

Scalable Assembly of General Objects

by

Yunsheng Tian

B.Eng., Nankai University (2019)

S.M., Massachusetts Institute of Technology (2021)

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2025

© 2025 Yunsheng Tian. All rights reserved.

The author hereby grants to MIT a nonexclusive, worldwide, irrevocable, royalty-free license to exercise any and all rights under copyright, including to reproduce, preserve, distribute and publicly display copies of the thesis, or release the thesis under an open-access license.

Authored by: Yunsheng Tian
Department of Electrical Engineering and Computer Science
August 15, 2025

Certified by: Wojciech Matusik
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: Leslie A. Kolodziejwski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Scalable Assembly of General Objects

by

Yunsheng Tian

Submitted to the Department of Electrical Engineering and Computer Science
on August 15, 2025 in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

ABSTRACT

In this thesis, I present a scalable system towards fully automated and flexible robotic assembly that generalizes over diverse geometries and complex structures. Most real-world objects are assemblies composed of multiple parts. Assembly presents significant challenges for robots to execute long-horizon, contact-rich manipulation with both reliability and generalization. However, most manufacturing facilities today still rely heavily on manually programmed assembly lines, which require significant labor, time, and setup costs yet offer no flexibility to object variations. My proposed system synergizes global multi-step planning with local reactive learning-based control to enable generalizable and precise assembly. Such an integrated paradigm effectively leverages the best of both worlds, accomplishing results that neither planning nor learning could achieve alone. For planning, I leverage guidance from physical simulation and learned feasibility networks to efficiently search for part sequences, precise motions, and stable grasps for dual-arm robots over long horizons. For learning-based control, I train robust policies via reinforcement learning for submillimeter-level insertion across different part geometries, assembly paths, and grasp poses. I introduce and open-source the largest-scale assembly dataset to date and demonstrate my system’s generalization on thousands of simulated assemblies as well as through end-to-end real robot experiments. By integrating planning and learning, I showcase the first system to achieve complete and generalizable real-world multi-part assembly without domain knowledge or human demonstrations. Although the system plans and learns purely in simulation, it transfers zero-shot to the real world and achieves 80% successful steps. Finally, I will share insights that further scale up robotic assembly and opportunities to extend to general manipulation, and discuss future directions to equip general-purpose robots with multi-step, precise manipulation capabilities.

Thesis supervisor: Wojciech Matusik

Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

Finishing a PhD feels like living through a century compressed into a handful of years. The world around me has changed so quickly, and so too have I — both in my research and in my personal life. I could not have made it through this journey without the support, guidance, and love of so many people.

First and foremost, I am deeply grateful to my advisor, Prof. Wojciech Matusik. His ambition, vision, and unwavering support gave me the courage to take on challenges I initially felt unprepared for. He taught me how to think and act independently, how to persevere when projects hit dead ends, and how to pursue ideas with both rigor and creativity. I will always carry his mentorship with me, and I hope to tackle greater and greater challenges with the resilience gained over the years.

I would also like to thank my thesis committee members, Prof. Pulkit Agrawal and Prof. Mina Konakovic Lukovic, for their invaluable feedback and encouragement — especially at the critical moments when my thesis was taking shape. I am grateful to Prof. Fredo Durand, Prof. Stefanie Mueller, and Prof. Leslie Pack Kaelbling for their sincere advice on my thesis, qualification exams, and on navigating PhD life and career in general. I am especially thankful to Prof. Leslie Kolodziejski for all the information, advice, and encouragement in my graduation, and being responsive all the time. It was your guidance and tireless support made my path to graduation clear and manageable. I feel fortunate to learn wisdom from all of you.

I owe much to my senior mentors — Prof. Shinjiro Sueda, Dr. Karl Willis, Dr. Rodger Luo, Dr. Hui Li, and Dr. Sachin Chitta — for shaping my thesis research and hosting my internship at Autodesk Pier 9, which remains a highlight in my PhD, not just for the research, but for the community of kind, energetic colleagues who made it unforgettable. I am grateful to Dr. Jie Chen and Prof. Chuang Gan for opening doors to collaboration with IBM and for their long-term continued mentorship. I also thank Prof. Fei Sha, Dr. Yashraj Narang and Dr. Wenhao Yu for sharing their perspectives on both my research and career paths. And I remember with gratitude my earliest mentors — Dr. Zherong Pan, Prof. Bo Ren, and Prof. Ming-Ming Cheng — who believed in me during my undergraduate years and opened the door that eventually brought me to MIT.

The Computational Design and Fabrication Group has been more than just a lab — it has been a family. I am thankful for Tao Du, Jie Xu, Liang Shi, Allan Zhao, Pingchuan Ma, Beichen Li, Yiyue Luo, Peter Yichen Chen, Minghao Guo, Yifei Li, Yichen Li, Liane Makatura, Alexandre Kaspar, Crystal Owens, Benjamin Jones, DongHun Ryu, Kui Wu, Wan Shou, Michael Foshey, Andrew Spielberg, Joshua Jacob, Bohan Wang, Bolei Deng, Yuanming Hu, Timothy Erps, Lara Zlokapa, Megan Tjandrasuwita, Wenqiang Chen, Young Lee, Ruitao

Su, Alston Lo, Michael Sun, and Devin Murphy. Our countless discussions, hangouts, BBQs, and retreats on the coast made the PhD years not only productive but joyful. I thank the administrative staff — Roger White, Megan Farmer, Felicia Raton, and Deborah Goodwin — for making daily life in the lab smooth and for organizing memorable group retreats.

I am grateful to my many collaborators and colleagues outside the lab who expanded my horizons: Yijiang Huang, Jialiang Zhao, Edward Gu, Annan Zhang, Farhad Javid, Yipu Chen, Jamison Meindl, Branden Romero, Bassel Omari, Ane Zuniga, Xinwei Zhang, Johannes Durholt, Daniela Rus, Veronika Thost, Tony Cui, Zhichu Ren, Jagdeep Bhatia, Holly Jackson, Zhou Xian, Yufei Wang, Yian Wang, Tsun-Hsuan Wang, Yiling Qiao, Zhiyang Frank Dou, Pavle Konakovic, Ashim Sitoula, Zhang-Wei Hong, Chao Liu, Xiang Zhang, Youcan Yan, Anthony Simeonov, and Lars Ankile. Thank you for the collaborations, technical discussions, and encouragement that pushed my work further.

No PhD is survivable without friends. I am grateful to Ruihan Yang, Bohan Zhang, Daqian Cheng, Jinsong Hao, Bingjie Tang, and Chen Wang for their constant emotional support across the years, even from afar. I thank Bowen Pan, Peiqi Wang, Rundong Ge, Yanwei Wang, Sixian Wang, Chang Liu, Xuanguai Huang, Ching-Yun Ko, Michael Lin, Xiaolin Fang, Yang Liu, Tao Chen, and Molin Zhang for the joyful dinners, parties, and the laughter that kept me deeply grounded in Boston. I still remember my first Thanksgiving here with Rui Qian, Tianjian Meng, Yongyuan Liang, and Yudian Li — a moment of warmth in the freezing winter when I needed it most. I enjoyed the company of neighbors and friends from Ted Adelson’s lab, including Haoshu Fang, Yuxiang Ma, Megha Tippur, Laurence Willemet, Shaoxiong Wang, and Branden Romero, whose conversations and laughter brightened many days. Before graduating, I was fortunate to build deeper connections with Zhenyu Jiang, Jingjing Chai, Zhutian Yang, and Lujie Yang, among others, whose friendships made the last stretch much lighter. To my old buddies from undergrad — Yujun Shi, Yuhao Zhang, Jiaxian Wang, Zhengwang Yu, Tingsheng Wang, Jiuqi Wei, Yingzheng Wang, Yunfei Sun, He Wang — and my high school friends — Wentao Yu, Jiayi Wu, Wentao Wang, Shiwangjue Shi, Xiaoyu Tong, Le Zhu, Jinglin Ji, Murong Xu, Luchuan Deng, Yuci Zhou, Danrong Zhang, Hao Suo, Qikun Guo — thank you for still keeping in touch and reminding me of the good old memories even as life scattered us across the world.

As my PhD drew to a close, I had the privilege of joining Amazon FAR, where I thank Peter Chen and Rocky Duan for recruiting me and sharing your vision on the future exciting journey, and Liang-Chieh Chen, Yang Liu, Qihang Yu, and Guanqi He for welcoming me and helping me begin this next chapter smoothly. It has been a wonderful experience working with you.

Most importantly, I thank my family. To my parents, who have always given me unconditional love and support despite the distance — your sacrifices and encouragement at every stage of my life carried me through moments of doubt and exhaustion. As I grow older, I come to understand more deeply how incredibly hard your journey was in raising me—the challenges you carried quietly, and the strength it took to give me the life and opportunities I have today. Even though I was too often busy and far away, you never once asked for anything in return. You reminded me to rest when I worked too hard, and supported me in silence when I struggled. From now on, I hope to bring you more laughter and more brightness in life. I am equally grateful to my wife’s parents, who welcomed me into their family as if I were their own son. Their kindness, wisdom, care, and understanding have

constantly supported me throughout this journey. I feel truly blessed to have gained not only parents-in-law, but another set of parents who care for me wholeheartedly and prayed for my success. My gratitude also extends to my grandparents. From childhood, they surrounded me with warmth and love that shaped who I am today. They cared deeply about my education and well-being, and their presence gave me a sense of belonging and security that I still carry with me. I especially hold close the memory of my paternal grandfather, who passed away during my PhD. His quiet yet profound love, and the values he instilled in me, will forever guide the way I live my life.

Finally, my greatest and deepest thanks go to my wife, Elsie, for her extreme patience, understanding, and never-ending support during the many sleepless nights and setbacks of this PhD. This PhD belongs as much to her as to me. She has been my anchor through every failed experiment and every moment of self-doubt. I am deeply grateful for her constant trust, for growing with me through every stage of life, and for patiently standing by me through all these years. To some extent, she has been my advisor in life—sharing her wisdom, teaching me how to live better and healthier, and helping me make the right decisions. At the same time, she is also my best friend and my closest therapist. She cooked for me every single day, filling our home with incredible recipes that carried not just flavor but her thoughtfulness and love. During the most intense seasons of deadlines and graduation, she would carefully prepare lunch and dinner boxes so that I could stay focused without worrying about meals. I deeply regret that in the busyness of my PhD I often could not even accompany her on something as ordinary as grocery shopping, moments that I now realize could have been cherished as time together. I wish I had managed my time better, so that I could have shared more of those small but meaningful parts of life with her. We created so many wonderful memories together in Boston and in our travels around the world—moments that made my life richer, more colorful, and filled with joy. With you, every day became something worth remembering. Words can hardly capture the depth of my gratitude for the warmth and happiness you brought into this journey. Finally, the hardest part of pursuing a PhD was not just the research itself, but also the search for what my ideal future life should look like. Over time, through your companionship, I gradually found clarity in that answer. I look forward with excitement to the years ahead with you by my side, and to building a life together filled with meaning, love, and countless more memories.

This thesis is not only the result of my own work, but also the collective effort of all the people who inspired, supported, and challenged me along the way. I am humbled by the generosity I have received and will carry it with me into the next chapter of my life.

Contents

<i>List of Figures</i>	13
<i>List of Tables</i>	15
1 Introduction	17
1.1 Motivations	17
1.1.1 Is Traditional Assembly Line All We Need?	17
1.1.2 Towards Autonomous, Flexible, and Generalizable Assembly	18
1.1.3 Necessity of Building a System	19
1.2 Overview of the Proposed System	19
1.2.1 Assembly by Disassembly	20
1.2.2 Assembly Path Planning	20
1.2.3 Assembly Sequence Planning	21
1.2.4 Assembly Grasp Planning	21
1.2.5 Assembly Policy Learning	21
1.2.6 Integrated Planning and Learning	21
1.3 Thesis Contributions and Impact	22
2 Background and Related Works	25
2.1 Path Planning for Assembly	25
2.2 Sequence Planning for Assembly	26
2.3 Physics Simulation for Assembly	27
2.4 Robotic Control for Assembly	27
2.5 Gap Analysis	28
3 Physics-Based Planning for Generalizable Virtual Assembly	31
3.1 Problem Formulation	33
3.2 Physics-Based Assembly Planning	33
3.2.1 Physics Simulation for Assembly	34
3.2.2 Disassembly Path Planning	36
3.2.3 Disassembly Sequence Planning	37
3.2.4 Assembly-by-Disassembly	38
3.3 Assembly Dataset	39
3.3.1 Dataset Pre-Processing	39
3.4 Evaluation	43
3.4.1 Baseline Methods	43

3.4.2	Two-Part Assembly	44
3.4.3	Multi-Part Assembly	47
3.4.4	Sub-Assembly and Interlocking Assembly	49
3.4.5	Failure Cases	50
3.5	Discussion and Conclusion	51
4	Efficient Assembly Sequence Planning with Physical Feasibility	53
4.1	Physically Feasible Sequence Planning	54
4.1.1	Disassembly Tree Search	54
4.1.2	Node Selection	56
4.1.3	Part Selection	57
4.1.4	Pose Selection	58
4.2	Feasibility Check for Assembly Sequences	58
4.2.1	Physics-Based Simulation for Assembly	58
4.2.2	Disassembly Path Planning	58
4.2.3	Gravitational Stability Check	59
4.3	Evaluation	60
4.3.1	Dataset for Assembly Sequence Planning	60
4.3.2	Baseline Methods	60
4.3.3	Experimental Setup	61
4.3.4	Quantitative Success Rate Comparison	61
4.3.5	Qualitative Assembly Sequence Comparison	62
4.3.6	Ablation Studies	62
4.3.7	Robotic Demonstration	64
4.4	Discussion and Conclusion	65
5	Real-World Dual-Arm Assembly via Integrated Planning and Learning	67
5.1	Planning Multi-Step Dual-Arm Assembly	68
5.1.1	Problem Formulation	68
5.1.2	Part Precedence Planning	71
5.1.3	Dual-Arm Grasp Filtering	72
5.1.4	Dual-Arm Sequence-Grasp Optimization	73
5.1.5	Grasp-Aware Pickup Fixture Generation	75
5.1.6	Motion Planning for Transit and Transfer	76
5.2	Learning General Single-Step Assembly Policy	76
5.2.1	Path-Centric Coordinate Transformation	77
5.2.2	Plan-Guided Residual Action	77
5.2.3	Minimalist Reward Design and Sim-to-Real Transfer	77
5.3	Experiments	78
5.3.1	Benchmark Suite and Experimental Setup	78
5.3.2	Real-World Deployment	78
5.3.3	Planning Multi-Step Assembly in Simulation	79
5.3.4	Learning Single-Step Assembly in Simulation	80
5.3.5	Executing Multi-Step Assembly in Real World	81
5.3.6	Integrating Vision Feedback: VLM for Insertion Alignment	87

5.4	Lessons Learned from Sim-to-Real Deployments	90
5.5	Discussion and Conclusion	91
6	Conclusion	93
6.1	Key Insights	93
6.2	Industrial Baseline Comparison	94
6.3	Future Directions	95
6.4	Thesis Impact	97
A	Hyperparameters of Physics-Based Path Planning	99
B	Algorithmic Details of Multi-Part Dual-Arm Assembly Planning	101
B.1	Part Precedence Planning	101
B.1.1	Precedence Tier Generation	101
B.1.2	Precedence Graph Generation	101
B.2	Dual-Arm Grasp Filtering	102
B.2.1	Single-Pose Grasp Feasibility Check	102
B.2.2	Assembling Grasp Feasibility Check	103
B.2.3	Holding Grasp Feasibility Check	103
B.2.4	Grasp Pair Filtering	103
B.3	Dual-arm Sequence-Grasp Optimization	105
	<i>References</i>	107

List of Figures

1.1	A simplified process of traditional assembly lines that heavily build on manual procedures to operate robots.	18
3.1	Our physics-based planning algorithm that determines the feasible assembly order (colored numbers) and searches for a feasible assembly motion path (colored curves) for each assembly step.	31
3.2	Motivation example: The washer can be disassembled from the shaft as long as the force applied has a positive projection along the true disassembly direction.	32
3.3	Overview of our physics-based assembly planning algorithm.	34
3.4	The signed distance field constructed (right) for each collision shape (left).	35
3.5	Overview of our dataset containing thousands of physically valid assemblies suitable for use with assembly planning tasks.	40
3.6	Screw category of the rotational dataset.	41
3.7	Puzzle category of the rotational dataset.	42
3.8	Others category of the rotational dataset.	42
3.9	More successful examples of two-part disassembly motion produced by our method while failing on baseline methods.	45
3.10	A tightly-fitted two-part disassembly (left: assembled, right: disassembled) solved by our method.	46
3.11	Screw disassembly (left: assembled, right: disassembled) solved by our method.	47
3.12	53-part engine disassembly (left: 53 parts left, middle: 11 parts left, right: 4 parts left) solved by our method.	48
3.13	A 3-part interlocking assembly disassembled by the movement of 2 parts (left: assembled, right: disassembled).	49
3.14	A failure case of our method disassembling a maze (geometries shown in the top row).	50
3.15	Large assemblies with several hundreds of parts and rich contacts where all tested methods reach the timeout of 2 hours.	51
4.1	Assembly plans generated autonomously from ASAP for a desk positioned on a rotary table including physically feasible assembly sequences, collision-free paths, gravitationally stable poses, gripper grasps, and robot arm motion.	54
4.2	An example disassembly tree where nodes represent partial assemblies and edges represent feasible (green) and infeasible (red) disassembly actions.	56

4.3	Network architecture for learning part disassembly priority. Given an assembly as input, a graph neural network is used to predict the next part to be removed.	57
4.4	Runtime usage by assembly size (number of parts).	62
4.5	Assembly sequence comparison between ASAP and <i>Assemble Them All</i> [125] on camera, train, and gear set.	63
4.6	Robotic assembly plans of 3D printed beams generated by ASAP executed in simulation (top row) and real-world (bottom row).	65
5.1	Our proposed dual-arm robotic system demonstrates adaptive manipulation and assembly capabilities for diverse multi-part objects.	67
5.2	Overview of Fabrica, our proposed robotic system with integrated planning and learning.	69
5.3	Illustration of our part precedence planning procedure.	71
5.4	Illustration of the four stages of the dual-arm grasp filtering process.	72
5.5	State tree for optimizing the part-grasp sequence.	73
5.6	Top: benchmark assemblies. Bottom: the auto-generated pickup fixtures in Section 5.1.5.	75
5.7	Step-by-step rendered assembly executions on different assemblies with different robots.	84
5.8	Step-by-step real-world assembly executions on different assemblies with Panda robots, with side-by-side correspondences between simulation and real.	85
5.9	Common failure cases during multi-step assembly execution.	86
5.10	Physical setup for integrating vision feedback. Left: Camera details. Right: The mounted configuration on the robot wrist.	88
5.11	Example outputs from VLM during corrective alignment.	90

List of Tables

3.1	Statistics of our assembly dataset.	40
3.2	Success rate (%) comparison on the entire two-part assembly dataset along with specific comparison on different rotational assemblies.	46
3.3	Success rate comparison (%) on multi-part assembly dataset.	48
3.4	Mean disassembly time per part (s) comparison on multi-part assembly dataset with different sizes of assemblies.	48
3.5	Distribution of failure cases in Assemble-Them-All aggregated over all benchmark assemblies.	51
4.1	Success rate (%) comparison of ASAP on the assembly test dataset against several baseline methods.	60
4.2	Accuracy and speed up of the proposed stability check algorithm compared to the combinatorial ground truth.	64
4.3	Success rate (%) comparison of ASAP on the assembly test dataset with reduced size of training dataset.	64
5.1	Key RL Hyperparameters.	79
5.2	Planning runtime breakdown of each assembly. Stages marked with * are parallelized, while others have yet to be parallelized.	80
5.3	Objective comparisons between optimal and random sequences. Higher is better for f_1, f_4 ; lower is better for f_2, f_3	80
5.4	% of successful steps without intervention in simulation evaluations.	80
5.5	Ablation studies on the impact of coordinate and action design choices on the assembly specialist (AS) policy’s performance across 1024 randomized simulation evaluations.	82
5.6	Scaling effect of number of policy trials across 3 complete end-to-end multi-step real-world evaluations.	83
5.7	Distribution of real-world failure cases in Fabrica aggregated over all benchmark assemblies.	84
5.8	% of successful steps without intervention in real-world evaluations.	87
5.9	Multi-step cumulative success rate with 0/1/2 interventions in real-world evaluations.	87
6.1	Qualitative comparison between the proposed framework and Siemens Process Simulate.	95

A.1	Hyper-parameters of physics-based simulation.	99
A.2	Hyper-parameters of physics-based path planners.	100
A.3	Hyper-parameters of geometric-based path planners.	100

Chapter 1

Introduction

"We realized that the true problem, the true difficulty, and where the greatest potential is – is building the machine that makes the machine."

- Elon Musk

1.1 Motivations

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. To transfer from digital designs to physical assemblies, robotic assembly systems are becoming increasingly essential in modern manufacturing, driven by the ongoing demand for automation, efficiency, and adaptability. As products become more complex and varied, industries face significant challenges in automating tasks that involve intricate geometries, precise alignment requirements, and dynamic, contact-rich interactions between parts.

1.1.1 Is Traditional Assembly Line All We Need?

Traditional assembly methods, built primarily for repetitive and predictable tasks, struggle significantly when adapting to new or modified products. These conventional systems typically rely on fixed assembly lines, where robots follow manually programmed paths and fixed procedures, see Figure 1.1. This rigidity demands extensive setup time and specialized human knowledge whenever there's a change in product design or when entirely new items are introduced, leading to production delays and higher costs.

Additionally, traditional assembly lines lack the flexibility needed for modern manufacturing approaches, which increasingly value adaptability and quick response to changing market demands. Practices such as mass customization, rapid prototyping, and high-mix, low-volume production require assembly systems that can easily handle frequent design adjustments. Traditional setups, with their predefined tasks and inflexible configurations, often fall short of these requirements.

Beyond inflexibility, traditional methods also struggle with today's complex products that often involve detailed parts, precise alignment, and dynamic interactions between components.

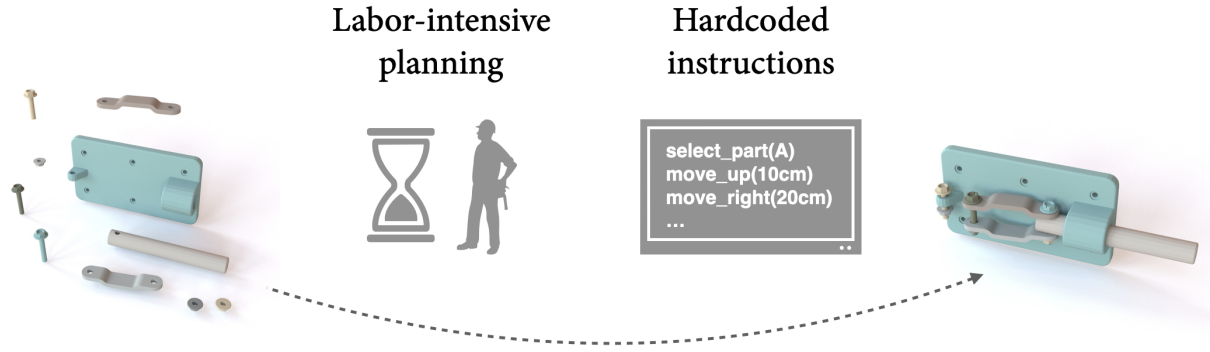


Figure 1.1: A simplified process of traditional assembly lines that heavily build on manual procedures to operate robots.

Dealing with varying part shapes and sequences autonomously is especially challenging for standard robotic systems. Frequent manual adjustments and reprogramming not only drive up costs but also slow down overall production, putting manufacturers at a competitive disadvantage.

As the industry shifts towards personalized products and shorter production runs, relying solely on traditional assembly lines becomes increasingly impractical. Today’s manufacturing needs systems that combine accuracy, speed, and intelligence. In other words, we need systems capable of adapting quickly and autonomously to diverse production scenarios without constant manual involvement. This reveals the limitations of traditional assembly lines and emphasizes the importance of developing next-generation robotic assembly techniques.

1.1.2 Towards Autonomous, Flexible, and Generalizable Assembly

Replacing traditional assembly lines poses significant challenges. A next-generation assembly system must excel in three critical dimensions: autonomy, flexibility, and generalization. Ensuring my technical approaches effectively meet these criteria requires thoughtful design decisions and robust methodologies.

Autonomy Autonomy in assembly systems promises a substantial reduction in manual involvement, both during system setup and in operation. Achieving true autonomy means matching or surpassing the runtime efficiency and consistency of traditional assembly lines while significantly reducing manual setup, pre-programming, and oversight. It also involves robustness, ensuring high-quality outputs consistently meet six-sigma industry standards. A practical approach toward high autonomy is to leverage a hybrid strategy, combining model-based methods that utilize known information about assembly geometry, poses, robot configurations, and workspace setups, with model-free, data-driven techniques to manage unpredictable uncertainties. The ultimate goal of autonomy in assembly systems is to seamlessly transition directly from digital assembly designs to fully realized physical assemblies, minimizing human involvement at every step.

Flexibility Flexibility in assembly contexts refers to the ease with which systems adapt to varying products and assembly tasks without extensive reconfiguration or significant customization. Traditional assembly methods rely heavily on rigid, specialized hardware infrastructure tailored to specific products, making repurposing costly and inefficient. Advances in artificial intelligence and dexterous robotic manipulation enable a simplified setup, potentially reduced to desktop-scale workspaces equipped with standard robotic manipulators. Flexibility is enhanced when the assembly system only requires basic inputs (such as CAD meshes and target poses) that are readily provided by designers. Ideally, the assembly process should demand no additional human expertise and minimal customization to accommodate new tasks or product variations.

Generalization Generalization addresses the capability of an assembly system to reliably and efficiently handle entirely new and previously unseen assembly tasks. A system that generalizes well can rapidly scale across diverse product lines and assembly configurations without performance degradation. Although recent breakthroughs in machine learning have enabled significant generalization improvements in simple robotic manipulation tasks, these methods are not yet mature enough to handle the complexities and precision required for long-horizon assembly tasks consistently at an industrial scale. Current generalization strategies, primarily driven by extensive data collection and training, still fall short in achieving the reliability and performance guarantees demanded by industrial standards. Addressing these challenges will require novel approaches combining accurate physical modeling, holistic planning, data-driven learning, to enable robust generalization capabilities.

1.1.3 Necessity of Building a System

While end-to-end learning has demonstrated remarkable successes in domains like vision and language, its applicability in complex robotic tasks such as assembly remains limited. Achieving industrial-level assembly performance in autonomy, flexibility, and generalization demands the integration of learning methods with traditional model-based engineering. Building a comprehensive, integrated assembly system is therefore essential, as relying exclusively on current end-to-end learning methods alone cannot yet meet the rigorous standards required by industrial robotics applications. Such a unified system, combining the best of both classical robotics and contemporary data-driven approaches, will be critical in bridging this capability gap and realizing fully autonomous, flexible, and generalizable robotic assembly solutions.

1.2 Overview of the Proposed System

The proposed system takes CAD meshes and assembled poses as input, and outputs a complete assembly plan. However, integrating recent advancements in robotic planning, control, machine learning, and physics-based simulation into a cohesive system capable of generalizable and robust assembly remains challenging. Existing systems often struggle with physically constrained spaces, precise positioning, multi-step sequencing, and simultaneous manipulation tasks involving multiple robot arms. Moreover, they lack comprehensive

benchmarks or large-scale datasets, hindering the systematic evaluation and comparison of methods and limiting progress within the research community.

This thesis directly addresses these critical gaps by introducing a unified framework that seamlessly integrates physics-based planning, automated sequence generation, and coordinated dual-arm manipulation. The proposed framework emphasizes physical realism and feasibility by employing physics-based simulations to guide assembly planning and learning-based methods to optimize robotic control policies. The central motivation is to significantly enhance the scalability and adaptability of robotic assembly systems, allowing rapid reconfiguration and efficient handling of diverse and complex assembly scenarios without extensive reprogramming.

Specifically, my system takes as input a rigid assembly of multiple parts, along with their meshes and assembled goal poses, which are readily available from CAD files in industrial manufacturing scenarios. The system autonomously generates and executes complete assembly plans, determining part paths and sequences, robot grasps and motions, fixture designs, and control policies to move each part sequentially from its disassembled pose to its assembled pose. This process accounts for key physical constraints, including kinematics, collisions, and dynamic stability. By addressing these fundamental issues, the framework proposed in this thesis aims to provide a practical and broadly applicable solution for the growing complexity and diversity of modern manufacturing tasks. The following sections provide the overview of main principles and methodologies used in this thesis.

1.2.1 Assembly by Disassembly

To reduce the inherent complexity involved in planning assembly tasks, we leverage the well-established *assembly-by-disassembly* principle [1]. This approach is grounded in the observation that disassembly is often simpler to plan than assembly due to clearly defined spatial constraints in the fully assembled state. By first planning the process in reverse by starting from the fully assembled product and determining a viable disassembly sequence, we inherently respect the critical geometric and physical constraints required in the final assembled configuration. Once a feasible disassembly sequence is obtained, we can reverse this sequence, systematically deriving a corresponding assembly plan. This method not only simplifies the computational burden of sequence planning but also naturally ensures collision-free, physically realistic, and efficient assembly paths, significantly speeding up the generation of practical robotic assembly solutions.

1.2.2 Assembly Path Planning

Assembly path planning involves determining collision-free and physically realistic trajectories for individual parts during assembly. Utilizing the assembly-by-disassembly principle, my system employs physics-based simulations to efficiently explore feasible assembly paths. This approach significantly reduces the complexity typically associated with traditional geometric motion planning methods, especially for assemblies involving complex geometries, tight clearances, and constrained spaces. By discretizing the search space into canonical actions and applying physics-based feedback, my method quickly finds viable motion paths that ensure robust and reliable assembly operations. Extensive evaluations on a large-scale dataset

of thousands of industrial assemblies demonstrated that my physics-based planning achieves state-of-the-art success rates and computational efficiency, effectively handling assemblies with as many as 80 parts within minutes.

1.2.3 Assembly Sequence Planning

Sequence planning addresses the combinatorial complexity of determining the order in which parts should be assembled. My method, ASAP [2], systematically generates physically feasible assembly sequences by leveraging tree-search algorithms enhanced with physics-based feasibility checks. It explicitly accounts for gravitational stability and collision constraints, ensuring each sub-assembly state is physically executable. By integrating geometric heuristics and graph neural networks trained on extensive simulation data, my approach efficiently guides sequence search, substantially outperforming traditional approaches in computational efficiency and generalizability. Experimental results demonstrated that ASAP achieved superior performance on a dataset comprising hundreds of complex product assemblies, consistently producing feasible sequences that meet realistic real-world physical constraints and significantly reducing the search complexity.

1.2.4 Assembly Grasp Planning

Grasp planning is crucial for stable and effective robotic manipulation during assembly tasks. As introduced in Fabrica [3], my approach precomputes valid grasp poses through parallelized offline inverse kinematics and collision checks, significantly simplifying online computations. The grasp selection process ensures compatibility with planned assembly paths and sequences, evaluating robotic arm configurations for stability, robustness, and collision avoidance. We demonstrated the practical efficacy of my method through rigorous experiments, where the optimized grasp poses consistently enabled precise assembly with maximal success rates.

1.2.5 Assembly Policy Learning

Policy learning is employed to handle the uncertainties and complexities inherent in contact-rich assembly tasks. My reinforcement learning framework [3] learns robust and generalizable assembly skills by exploiting equivariant representations and residual action frameworks, guided by pre-planned motions. This approach ensures that policies effectively generalize across various object geometries, grasp configurations, and insertion orientations. Experimental validations highlighted the policy’s high transferability, achieving an 80% success rate in zero-shot sim-to-real scenarios. The learned policies proved lightweight, computationally efficient, and significantly improved precision and adaptability in real-world robotic assembly tasks, demonstrating remarkable robustness to real-world variations and uncertainties.

1.2.6 Integrated Planning and Learning

My integrated planning and learning framework, Fabrica [3], bridges the gap between simulation and real-world execution. By combining an efficient hierarchical planner with generalist reinforcement learning policies, my system achieves reliable zero-shot sim-to-real

transfer, successfully executing long-horizon, dual-arm assembly tasks. This integrated approach systematically addresses sequence optimization, grasp selection, motion planning, and fixture generation, while the learned reactive control policies adaptively handle real-world uncertainties. Comprehensive benchmark evaluations demonstrated that my integrated system reliably performed across diverse multi-part assemblies such as furniture, electronic devices, and mechanical components, outperforming traditional methods. Detailed timing analyses and extensive real-world trials confirmed the system’s efficacy, robustness, and generalizability, marking a significant advancement toward fully autonomous robotic assembly in dynamic industrial and everyday scenarios.

1.3 Thesis Contributions and Impact

This thesis makes the following key contributions:

1. Developed a comprehensive assembly-by-disassembly method (Assemble-Them-All). It leverages physics-based simulations to plan complex assembly sequences and handle multi-part assemblies with intricate geometries and constrained motion paths.
2. Formulated the ASAP algorithm. It integrates geometric heuristics and physics-based feasibility checks to generate robust sequence plans for contact-rich, multi-step assemblies.
3. Designed and implemented the Fabrica framework for dual-arm assembly. It optimizes grasp filtering, sequence-grasp coordination, and motion planning to efficiently handle simultaneous multi-part manipulations, reducing assembly time and improving task success rates.
4. Introduced a large-scale open-sourced dataset for assembly planning. It includes thousands of assemblies with varying part geometries, assembly sequences, and contact dynamics, serving as a benchmark for evaluating robotic assembly algorithms.
5. Demonstrated the proposed system through extensive simulated and real-world robotic assembly tasks. Showcased its effectiveness in handling diverse objects, complex interactions, and tight tolerances.

This thesis presents a framework that significantly advances the state-of-the-art in robotic assembly, providing substantial benefits across multiple dimensions critical to industrial automation and general-purpose robotic manipulation. By integrating physics-based planning, automated sequence generation, and sophisticated dual-arm coordination into a cohesive and unified system, the proposed approach substantially enhances the adaptability, flexibility, and scalability of robotic assembly operations.

Firstly, this work addresses the critical challenge of variability and complexity inherent in modern manufacturing environments. Rigid, predefined motion plans and costly reprogramming limit traditional robotic systems, especially when designers introduce new products or modify existing ones. Our approach greatly reduces reliance on manual interventions and custom scripting by enabling robotic systems to autonomously plan and execute assembly

tasks based on realistic physics simulations and adaptive learning methods. This increased autonomy and adaptability significantly lowers the setup and operational overhead, facilitating rapid deployment and adjustment of assembly tasks across varied production scenarios.

Secondly, the integration of dual-arm robotic assembly significantly expands the range and complexity of tasks that can be reliably automated. Dual-arm coordination not only enhances the precision and robustness of handling complex geometries and intricate alignments but also broadens the spectrum of feasible assemblies. By optimizing grasp selection, sequencing, and fixture design, the proposed dual-arm framework addresses critical aspects of simultaneous manipulation that have traditionally posed significant challenges. These advancements open new possibilities for robotic systems to engage effectively in high-precision, multi-step assemblies that were previously inaccessible or required significant manual labor, thus expanding the practical applicability of robotic assembly to a wider range of industries.

Furthermore, the creation and dissemination of a large-scale, diverse dataset and benchmark is a critical contribution of this thesis. Previously, the scarcity of standardized benchmarks and comprehensive datasets often constrains research efforts. By providing a systematically designed dataset that encompasses thousands of assembly scenarios with various complexities, part geometries, and contact interactions, this thesis facilitates more rigorous and transparent comparisons among different methods. Such a resource is invaluable for advancing both theoretical research and practical algorithm development, enabling researchers worldwide to benchmark new approaches, identify strengths and weaknesses clearly, and iteratively improve upon existing methodologies.

Moreover, the detailed empirical validation performed through both simulation-based and real-world robotic experiments reinforces the practical impact and reliability of the proposed methods. Real-world validation is crucial in the field of robotics, where theoretical solutions must be robust enough to handle uncertainties, noise, and dynamic real-world conditions. My comprehensive testing in this thesis provides strong evidence that the developed framework can not only generalize well across diverse assembly scenarios but also perform reliably under realistic, industrially relevant conditions. This validation bridges the gap between theoretical advancements and practical, real-world applicability, significantly enhancing the credibility and adoption potential of the research within industrial settings.

In broader terms, the proposed framework is poised to catalyze a transformative shift in manufacturing and robotic manipulation paradigms, particularly benefiting applications in mass customization, rapid prototyping, and high-mix, low-volume production scenarios. As industries increasingly demand flexibility and responsiveness, the ability of robotic systems to quickly adapt to new tasks and environments becomes paramount. By directly addressing these requirements through intelligent planning and adaptive control, this thesis lays critical groundwork for future robotic systems that are not only highly capable but also inherently flexible and responsive to changing industrial needs.

Finally, this thesis impacts the broader field of robotics research by demonstrating a successful integration of machine learning, physics-based simulation, and traditional robotic planning methods. It highlights how leveraging complementary strengths across these domains can lead to superior system performance and generalization. This interdisciplinary approach is likely to inspire further research into hybrid robotic systems that combine the best of data-driven and classical engineering methodologies, thereby driving forward innovations across robotics, artificial intelligence, and industrial automation more generally.

Chapter 2

Background and Related Works

Building on the system-level motivations and objectives outlined in Chapter 1, we now ground our work in the broader context of existing research. In this chapter, I review the background and related works on assembly automation from the perspectives of path planning, sequence planning, physics simulation, and robotic control.

2.1 Path Planning for Assembly

Assembly automation begins with understanding the paths for assembling parts, typically done manually by experienced manufacturing teams. The goal is to compute 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. For example, Wilson et al. [4] proposed a directional blocking graph (DBG) and non-directional blocking graph (NDBG). Halperin et al. [5] 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 [6]. Probabilistic Roadmap Method (PRM) [7] and Rapidly-exploring Random Tree (RRT) [8] are the two notable sampling-based algorithms. PRM, RRT, and their variants [9–12] have proven to be more efficient at solving general motion planning than geometric reasoning methods on assembly problems [13–15]. But they usually suffer from the ‘narrow passage’ problem [16], especially in high-dimensional space where the free space is tight and rotation needs to be considered.

Physics-based motion planning [17] was developed as a competitive alternative to geometric-based motion planning (i.e. geometric-reasoning and sampling-based) and showed success with robotic navigation [18] and manipulation [19]. However, physics-based motion planning has not been fully explored for assembly where the motion is much more constrained. In this thesis, we demonstrate the benefit of using physics-based planning for efficient exploration of assembly paths to overcome the ‘narrow passage’ problem.

2.2 Sequence Planning for Assembly

Beyond path planning, assembling a product requires arranging an optimal sequence in assembling all the components and is a critical step for assembly system design [20]. This sequence planning problem can be represented as a graph such as AND/OR graph [21] to establish feasible assembly sequences. Different optimization techniques are applied to find the optimal assembly sequence among the feasible sequences [22–25]. 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 [26, 27]. Although many attempts have been made to reduce the search space, the number of potential assembly sequences can grow exponentially [28] which makes identifying a feasible sequence impossible even for complex assemblies.

The idea of assembly-by-disassembly has been proposed as an important strategy for speeding up, because assembled parts have better defined precedence and motion constraints than disassembled parts [1], 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. [29] The assembly-by-disassembly approach has been adopted in several applications in manufacturing and construction, including mechanical product assembly [1, 30], kit assembly [31], and robotic additive construction [32]. However, if the assembling directions 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.

In addition to path feasibility constraints, for real-world assemblies, gravitational stability is also considered during planning [33, 34]. However, evaluating the physical feasibility of complex-shaped assemblies accurately is difficult and time-consuming. To speed up the feasibility check, Rodriguez et al. [35] propose an iterative procedure on planar profile assembly by checking complex dynamic constraints later only if simpler static constraints are all satisfied. To tackle the combinatorial complexity of sequence optimization, evolutionary methods have been proposed, such as genetic algorithms [36–38], ant colony optimization [30]. However, these methods require evaluating massive sequences in order to be successful; hence their evaluation metrics are usually simple and do not consider gravitational stability, preventing the direct deployment of their optimized sequences in a real-world setup. More efficient search techniques have been explored in [32, 39] to specifically showcase stable assembly of frame shapes and bar structures. In contrast to prior work, I demonstrate my sequence planner on a complex general-shaped 3D assembly dataset derived from the Fusion 360 Gallery Assembly Dataset [40] and a real mechanical assembly dataset [41], featuring more complex shapes than previously considered. In this thesis, I consider more physical constraints, such as gravity and friction, as well as robotic constraints, in determining the feasibility of each assembly step. Additionally, we take into account the use of a flat support surface, stable poses, and grippers for holding unstable parts.

Learning-based methods are promising to tackle the combinatorial problem of assembly sequence generation. Self-supervised learning [31] collects data by trial-and-error kit disassem-

bly but is limited to 2D shape-matching tasks. Reinforcement learning methods [42–44] can train effective 3D assembly planners but demonstrate on simplified physics or geometries due to high sample complexity. My approach employs supervised learning inside planning, using a graph neural network (GNN) to predict disassembly sequences with physical feasibility for complex 3D assemblies. I gather training data from diverse assemblies generated via physics-based simulation, and the network shows effective guidance for planning disassembly sequences on unseen assemblies.

However, planning alone struggles with execution uncertainties, and stability- or efficiency-optimal plans remain underexplored. While robust and efficient robotic systems have been built for tasks like assembling IKEA chairs [45], LEGO blocks [46], and structural elements [32, 47–52], these are domain-specific and lack generalizability. In contrast, my planner generalizes across diverse multi-part assemblies, employs hierarchical structure and parallelization for efficiency, and explicitly optimizes stability to enhance downstream control robustness.

2.3 Physics Simulation for 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 [53–55] and motion planning [17, 18, 56]. It has been successfully applied to locomotion [57, 58], flying [59, 60], swimming [61], and manipulation [62].

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 3.4, requires accurately representing the object surface while efficiently resolving contact constraints. Most existing simulators [63–65] 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 [66, 67] or box stacking [34, 68]. In Kim et al. [69], physical simulation constructs a physically reasonable graph to speed up searches, tested on box-on-table and peg-in-hole tasks. Another category of contact-rich simulators uses penalty-based contact models [70, 71], but most only support collision detection between contact points and primitive shapes. I adopt the rigid body simulation with a penalty-based frictional contact model from Xu et al. [70], and extend it with a signed-distance-field (SDF) representation of collision body in order to simulate the high-accuracy assembly tasks with complex object geometries. Concurrent to my work, Narang et al. [72] announced a GPU-based simulator for assembly tasks, which also utilizes the SDF shape representation.

2.4 Robotic Control for Assembly

Motion planning has been widely studied in robotic assembly for manufacturing [73, 74] and building construction [32, 75], albeit in less realistic scenarios where planners are limited to structured environments. However, even with given assembly plans, executing contact-rich assembly remains challenging due to tight clearances, system uncertainties, and the need for generalization. Reinforcement Learning (RL) [76], on the other hand, has been applied to

more industrial and general-purpose assembly tasks [77–80] in unstructured environments [81]. My real-world dataset and SDF-based simulation can aid RL-based methods in acquiring more generalized policies. Likewise, my physics-based planner can generate complex assembly motions that can directly guide robots or serve as demonstrations to learn from [82, 83].

RL has shown promise in addressing these issues, combining motion planning with policy learning from CAD models or supervised trajectories [77, 80], and leveraging accurate simulations for motion generation and policy training [2, 72]. Sim-to-real transfer [84–87] and real-world RL [88] have enabled high-precision insertion, while some efforts [89] explore multi-step tasks. However, they all primarily work under simplified settings like top-down insertions and fixed grasps, which are insufficient for multi-part assembly where side-way insertions or tilted grasps are necessary. Varying grasps is particularly challenging for contact-rich assembly because changing the grasp frame alters the relationship between observed forces, measured poses, and the intended motion in the assembly frame. The same force vector or positional offset can correspond to very different physical effects depending on grasp orientation, and contact dynamics can shift due to changes in leverage and surface engagement. Without explicitly handling this variability, a policy trained for one grasp often fails for another, as it must implicitly relearn the action–effect mapping for each configuration. Imitation learning approaches [90, 91] support complex multi-step skill learning but lack robustness and generality. While spatial-equivariant techniques have improved generalization in other domains [92–96], they remain underexplored for assembly. Notably, Seo et al. [97] learn an SE(3)-equivariant gain-scheduling policy for contact-rich insertion by behavior cloning from expert demonstrations, achieving invariance to global translations and rotations of a single task. However, their formulation assumes a fixed object geometry, a fixed approach direction, and a fixed grasp frame, so the learned controller generalizes only to pose-transformed versions of the same assembly. As a result, it does not address the challenges posed by varying contact geometries, side-entry or tilted insertion paths, or changes in the grasp pose. Moreover, existing benchmarks focus on narrow tasks [98–102], limiting the evaluation of generalization.

In contrast, I propose a RL-based controller designed to handle diverse object geometries, varying grasp poses, and multiple assembly directions within a single generalist policy. I introduce a path-centric SE(3)-equivariant transformation that maps each planned assembly path into a canonical top-down insertion problem regardless of its approach direction or grasp frame, simplifying policy learning across variations. The controller operates as a residual policy on top of an open-loop motion plan, enabling fine-grained corrections that adapt to changes in contact conditions. Crucially, the policy is trained on a wide range of parts, grasps, and approach angles, and is integrated into a dual-arm assembly system where sequence and grasp planning further ensure stability and collision avoidance. This design allows the learned policy to generalize to novel assemblies and grasp configurations that would break the assumptions of previous fixed-geometry, fixed-grasp approaches.

2.5 Gap Analysis

Despite significant progress in assembly-related research, existing methods fall short of meeting the combined requirements of autonomy, flexibility, and generalization at an industrial scale, as defined in Sections 1.1.2 and 1.1.3.

From an autonomy standpoint, most prior systems still rely heavily on manual programming, task-specific tuning, or curated demonstrations, limiting their ability to operate without ongoing human intervention. Even approaches that leverage learning-based control or physics-based planning often focus on narrow scenarios with fixed setups, rather than end-to-end pipelines capable of transitioning directly from digital designs to physical assemblies with minimal human input, as envisioned in Section 1.1.2.

In terms of flexibility, current solutions are often tailored to specific object geometries, predefined assembly sequences, or constrained environments. This rigidity makes them costly to adapt to new products, high-mix low-volume production, or rapid design changes. While some methods explore modular planners or learning policies, they typically lack a unified framework that can seamlessly accommodate new parts, robot embodiments, and spatial constraints without extensive reconfiguration, which is a key requirement outlined in Section 1.1.2.

For generalization, many existing learning-based approaches achieve promising results on benchmark tasks but fail to scale reliably to unseen assemblies with different shapes, tolerances, and contact dynamics. These methods are often trained on limited datasets or under simplified physical models, making them brittle when transferred to realistic, contact-rich industrial settings. Similarly, existing model-based planning methods may handle novel geometries in principle but can become computationally infeasible when confronted with the combinatorial complexity of large assemblies. Achieving the level of robust performance across diverse products called for in Section 1.1.3 remains an open challenge.

These limitations indicate a critical gap: there is no existing framework that simultaneously integrates accurate physical reasoning, scalable sequence and grasp optimization, and robust reactive control to deliver industrial-grade performance across diverse assemblies. Addressing this gap is the central focus of this thesis, which proposes an integrated planning and learning system designed explicitly to achieve autonomy, flexibility, and generalization in real-world robotic assembly.

Chapter 3

Physics-Based Planning for Generalizable Virtual Assembly

Having established the relevant background and identified limitations of existing approaches in Chapter 2, we now present our first major technical contribution: a physics-based planning framework for assembly-by-disassembly. This chapter focuses on generating physically valid part motions and sequences in virtual environments, forming the foundation for the high-level sequence optimization and integration with real-world systems in the subsequent chapters.

The ability to automatically plan the steps required to assemble and disassemble designs created in computer-aided design (CAD) is an enabling technology for a number of downstream applications. In the design phase, Design for Assembly (DFA) checks can be performed to correct potential issues before manufacture [103]; assembly planning for high mix, low volume products can be greatly simplified with automatic generation of assembly instructions [104]; and at end-of-life, robotic disassembly systems [105] can be quickly adapted to recycle product components.

Despite significant research on assembly planning [29, 106], 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

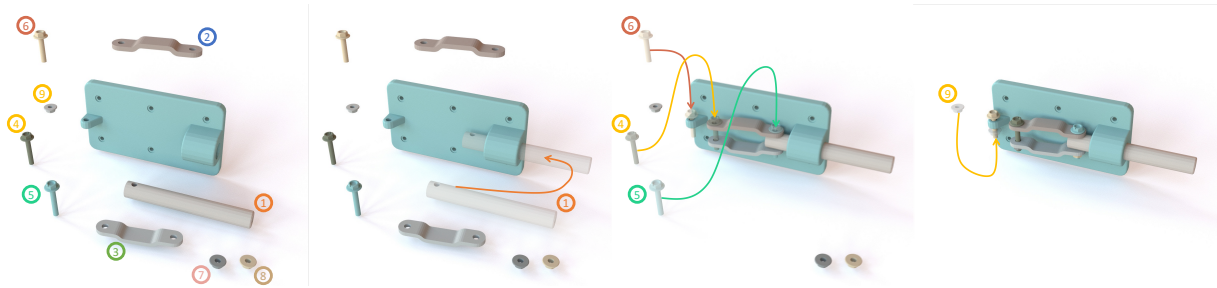


Figure 3.1: Our physics-based planning algorithm that determines the feasible assembly order (colored numbers) and searches for a feasible assembly motion path (colored curves) for each assembly step.

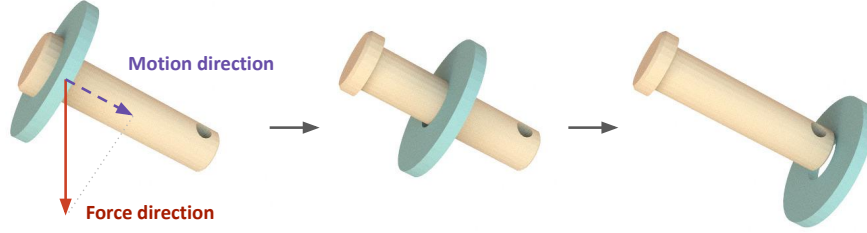


Figure 3.2: Motivation example: The washer can be disassembled from the shaft as long as the force applied has a positive projection along the true disassembly direction.

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 3.1). 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 3.2 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:

1. We introduce an accurate and efficient physics-based simulator that is customized for assembly.
2. We propose a novel physics-based assembly-by-disassembly planning method for translational and rotational assembly motion for arbitrary-shaped assemblies.
3. We define a large-scale dataset and benchmark for assembly planning including thousands of physically valid assemblies.
4. 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.

Our code, data, and videos are available at assembly.csail.mit.edu.

3.1 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 $\{s_0^i, \dots, s_n^i\}$ that connects the assembled state s_0^i , given as input and a disassembled state s_n^i . Correspondingly, an assembly path $P_A^i = \{s_n^i, \dots, s_0^i\}$ 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.

3.2 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 3.3. 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).

In the following sections, we first introduce our physics simulation for assembly in Section 3.2.1 as the key component to our algorithm. Next, we illustrate our disassembly

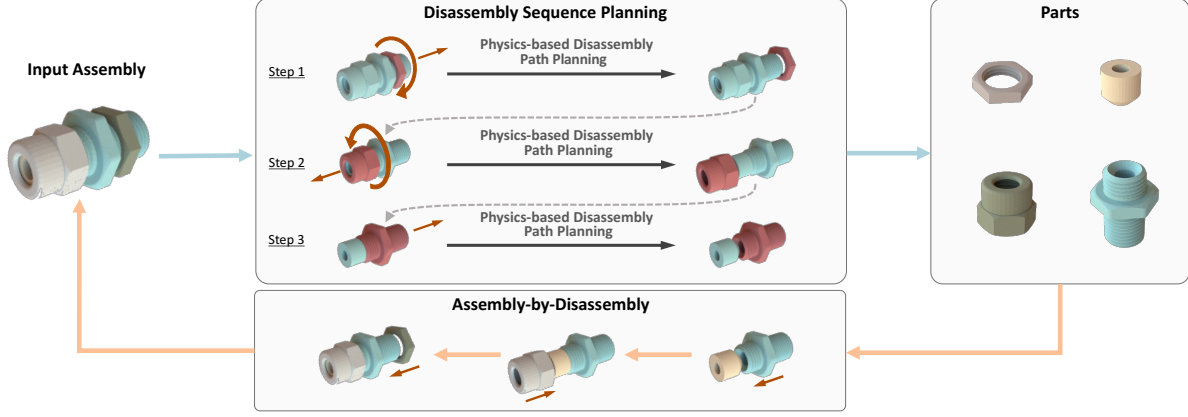


Figure 3.3: Overview of our physics-based assembly planning algorithm.

path planning algorithm for two-part disassembly in Section 3.2.2, then our disassembly sequence planning algorithm for multi-part disassembly in Section 3.2.3. Finally, we present in Section 3.2.4 how to obtain the assembly sequence and path by reversing solutions from disassembly planning, following the assembly-by-disassembly strategy (as defined in Section 1.2.1).

3.2.1 Physics Simulation for Assembly

The existing robotics simulators [63–65] 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. [70] and extend it with the signed distance field (SDF) representation of collision shapes to support complex collision geometries.

Specifically, the simulator developed by Xu et al. [70] is based on the reduced coordinate rigid body dynamics formulation of RedMax [107], 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:

$$\begin{array}{ll} d, \dot{d} & \text{penetration distance and speed} \\ \vec{n} & \text{unit-length contact normal direction} \end{array}$$

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

$$\vec{f}_c = (-k_n + k_d \dot{d}) d \vec{n}, \quad (3.1)$$

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

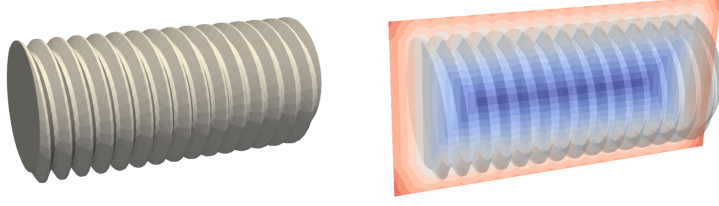


Figure 3.4: The signed distance field constructed (right) for each collision shape (left).

not affect the assembly motion under the assumption that every part is rigid. To efficiently acquire the required distance function, Xu et al. [70] restrict one of each pair of collision shapes to be a primitive shape with an analytical distance function.

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, see Figure 3.4. The right figure shows the level set of the cross-section signed distance field of a fine-thread screw. The distance is negative when the point is inside the geometry while the distance is positive if the point is outside.

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.

We construct the SDF grid for an object using the Fast Sweeping Algorithm [108] 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 be acquired in $\mathcal{O}(1)$ by trilinearly interpolating among surrounding grid cells and the gradients are computed in $\mathcal{O}(1)$ 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.

3.2.2 Disassembly Path Planning

For a given part in the assembly state s_0 , we look for a sequence of states $\{s_0, \dots, s_n\}$ 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 3.2, 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. In translational-motion-only cases, $A = \{[\pm 1, 0, 0], [0, \pm 1, 0], [0, 0, \pm 1]\}$ contains 6 actions corresponding to 6 translational degrees of freedom (DoF). When rotational motion is enabled, A has 12 actions for the 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 the Euclidean distance $\|s_{a_t} - s_{b_t}\|$ between the linear components of the two states. The rotational distance is the geodesic distance $\|\ln(s_{a_r}^{-1}s_{b_r})\|$ between their quaternion representations of the angular components. 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.

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) [109] along with carefully designed heuristics might be applied to speed up the search, though it might not perform equally well on simpler assemblies.

Algorithm 1 Disassembly Path Planning Guided by BFS

Require: Assembled state s_0 , timeout $t_{\max}$, max BFS depth $d_{\max}$

Ensure: A disassembly path $P_D = \{s_0, \dots, s_n\}$

```

1:  $Q \leftarrow \text{EmptyQueue}()$ 
2:  $Q.\text{Enqueue}(s_0)$ 
3: while  $t < t_{\max}$  and BFS depth  $d \leq d_{\max}$  do
4:    $s \leftarrow Q.\text{Dequeue}()$ 
5:   for all  $a_i$  in action space  $A$  do
6:      $s_i \leftarrow s$ 
7:     while  $s_i$  is not similar to any past states do
8:        $s_i \leftarrow \text{Simulate}(s_i, a_i, \Delta t)$ 
9:       if  $\text{IsDisassembled}(s_i)$  then return  $\text{GetPath}(s_0, s_i)$ 
10:     $Q.\text{Enqueue}(s_i)$ 
return failed

```

3.2.3 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 [110].

We propose a method to efficiently plan the disassembly sequence by progressively expanded BFS, see Algorithm 2. Given the assembled states $\mathbf{s} = \{s_0^1, \dots, s_0^M\}$ of all M parts, we look for an ordered sequence of disassembly paths $\mathbf{P}_D = \{\{s_0^{I_1}, \dots, s_n^{I_1}\}, \dots, \{s_0^{I_M}, \dots, s_n^{I_M}\}\}$ that connect the assembled state and an disassembled state for each part and satisfy precedence relationships. Here, $I_1, \dots, I_M$ 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 $d_{\max} = 1$. 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 are two main reasons for this failure. First, this part may be blocked by other parts, i.e., conflict with the precedence relationships. Second, this part may be unblocked 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 3.4.3.

Algorithm 2 Disassembly Sequence Planning with Progressive BFS

Require: Assembled states $\mathbf{s} = \{s_0^1, \dots, s_0^M\}$ of all M parts, path planning timeout $t_{\max}$, sequence planning timeout $T_{\max}$

Ensure: An ordered sequence of disassembly paths $\mathbf{P}_D = \{P_D^{I_1}, \dots, P_D^{I_M}\}$

- 1: $\mathbf{P}_D \leftarrow \{\}$
- 2: max BFS depth $d_{\max} \leftarrow 1$
- 3: **while** $t < T_{\max}$ **do**
- 4: **for all** part i in all remaining parts of the assembly **do**
- 5: $P_D^i \leftarrow \text{DisassemblyPathPlanning}(s_0^i, t_{\max}, d_{\max})$
- 6: **if** $P_D^i \neq \text{failed}$ **then**
- 7: $\mathbf{P}_D.\text{Append}(P_D^i)$
- 8: Remove part i from the assembly
- 9: **if** all M parts are disassembled **then return** $\mathbf{P}_D$
- 10: $d_{\max} \leftarrow d_{\max} + 1$
- return** failed

3.2.4 Assembly-by-Disassembly

Following the assembly-by-disassembly strategy (see Section 1.2.1), we reverse the disassembly sequence and paths and connect with the initial states to construct the entire assembly sequence and paths, 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 [111] 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.

Algorithm 3 Assembly by Disassembly

Require: Initial states $\mathbf{s}_{\text{init}} = \{s_{\text{init}}^1, \dots, s_{\text{init}}^M\}$, assembled states $\mathbf{s}_{\text{goal}} = \{s_{\text{goal}}^1, \dots, s_{\text{goal}}^M\}$ of all M parts

Ensure: An ordered sequence of assembly paths $\mathbf{P}_A = \{P_A^{I_M}, \dots, P_A^{I_1}\}$

- 1: Compute ordered disassembly paths $\mathbf{P}_D$ from Alg. 2 given $\mathbf{s}_{\text{goal}}$
 - 2: $\mathbf{P}_A \leftarrow \{\}$
 - 3: **for** $i \leftarrow M$ to 1 **do**
 - 4: Obtain the disassembled state $s_{\text{dis}}^{I_i}$ from disassembly path $P_D^{I_i}$
 - 5: Compute a collision-free path $P_C^{I_i}$ connecting $s_{\text{init}}^{I_i}$ and $s_{\text{dis}}^{I_i}$ using RRT-Connect
 - 6: $P_A^{I_i} \leftarrow P_C^{I_i} + \text{Reverse}(P_D^{I_i})$
 - 7: $\mathbf{P}_A.\text{Append}(P_A^{I_i})$
 - return** $\mathbf{P}_A$
-

3.3 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 [112, 113] with part mobility labels [114–117]. 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 [40, 118–120] 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. [40] as well as assemblies from Ebinger et al. [110] and Zhang et al. [15]. 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 3.1 lists the total number of assemblies and their distribution by number of parts in each assembly. Figure 3.5 shows a random sampling of assemblies in the dataset. Figure 3.6, 3.7 and 3.8 show a complete overview of our rotational dataset, which consists of rotational assemblies that require various types of assembly motion. Our dataset is open-sourced at: <https://github.com/yunshengtian/Assemble-Them-All>.

3.3.1 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:

Table 3.1: Statistics of our assembly dataset.

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

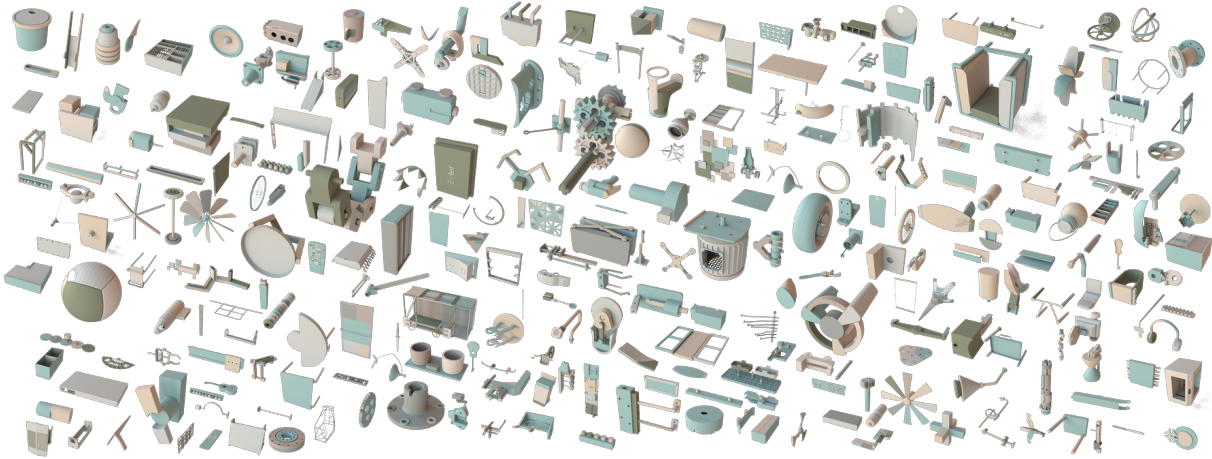


Figure 3.5: Overview of our dataset containing thousands of physically valid assemblies suitable for use with assembly planning tasks.

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

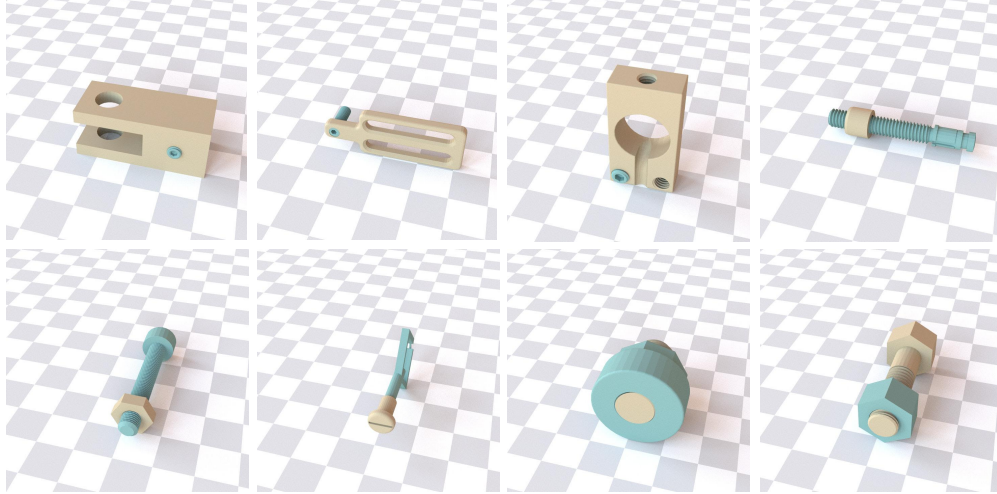


Figure 3.6: Screw category of the rotational dataset.

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.

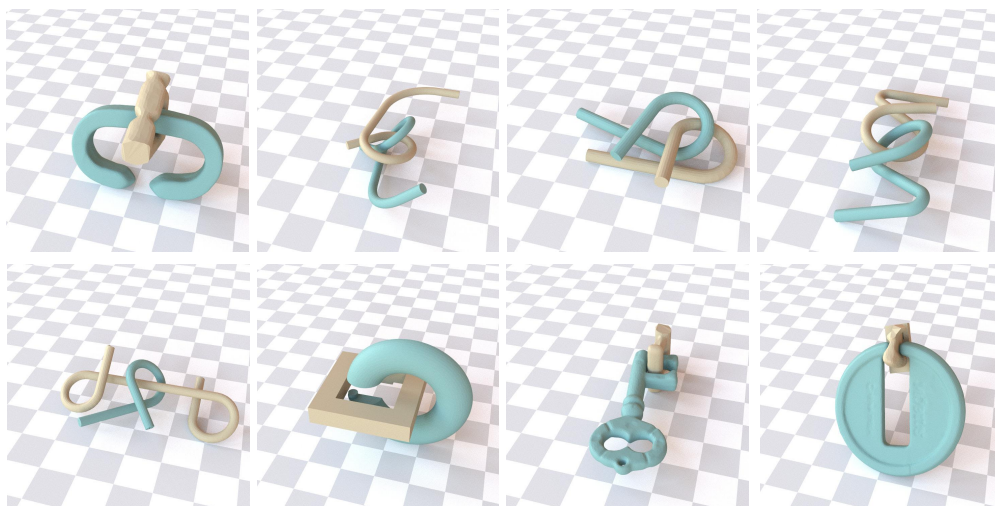


Figure 3.7: Puzzle category of the rotational dataset.

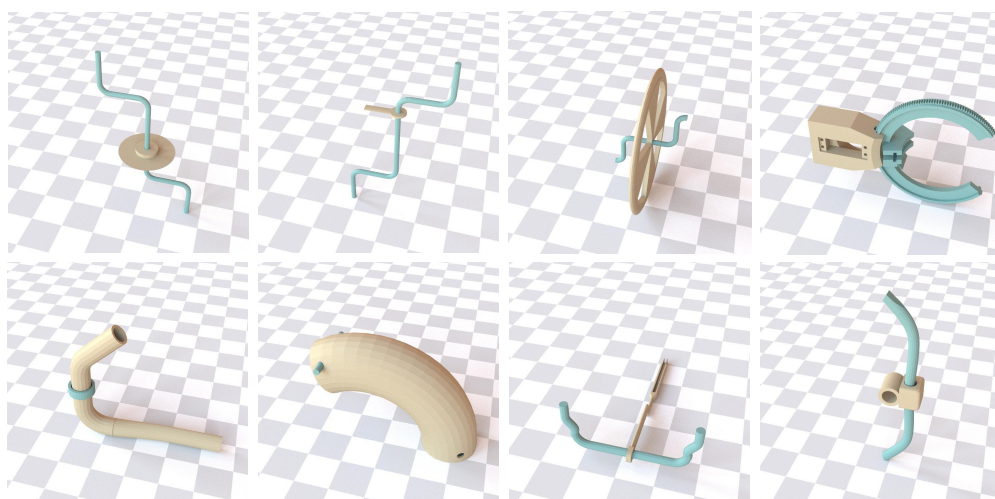


Figure 3.8: Others category of the rotational dataset.

3.4 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 3.4.4, 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 experiments can be found in Appendix A, including detailed hyper-parameters for physics-based simulation, physics-based path planners, and geometric-based path planners.

3.4.1 Baseline Methods

We choose the following path planning methods which are the most relevant baselines to compare:

RRT : Rapidly-Exploring Random Tree [8] is one of the most standard geometric path planning approaches. 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 : Targetless-RRT [11] is a modified version of RRT that is specifically designed for disassembly planning, which does not require a specified specific goal state. 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.

MV+T-RRT : Mating Vector + Targetless-RRT, a hybrid approach proposed in [110] 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 [17] 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. See Algorithm 4 for a complete illustration of BK-RRT in disassembly path planning.

Algorithm 4 BK-RRT for Disassembly Path Planning

Require: Assembled state s_0 , timeout $t_{\max}$, simulation time step Δt

Ensure: A disassembly path $P_D = \{s_0, \dots, s_n\}$

```
1:  $T \leftarrow \text{EmptyTree}()$ 
2:  $T.\text{AddNode}(s_0)$ 
3: while  $t < t_{\max}$  do
4:    $s_r \leftarrow \text{SampleRandomState}()$ 
5:    $s_i \leftarrow \text{NearestNeighbor}(T, s_r)$ 
6:    $a_i \leftarrow \text{RandomAction}()$ 
7:    $s_{i+1} \leftarrow \text{Simulate}(s_i, a_i, \Delta t)$ 
8:    $T.\text{AddNode}(s_{i+1})$ 
9:    $T.\text{AddEdge}(s_i, s_{i+1}, a_i)$ 
10:  if  $\text{IsDisassembled}(s_{i+1})$  then return  $T.\text{GetPath}(s_0, s_{i+1})$ 
    return failed
```

3.4.2 Two-Part Assembly

We evaluate our path planning algorithm and baseline approaches on the two-part dataset and report the performance comparison in Table 3.2 (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 3.2 (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 3.10. 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 3.10, 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 3.11. See Figure 3.9 for more assembly motion produced by our method which have a low success rate on other baseline methods.

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

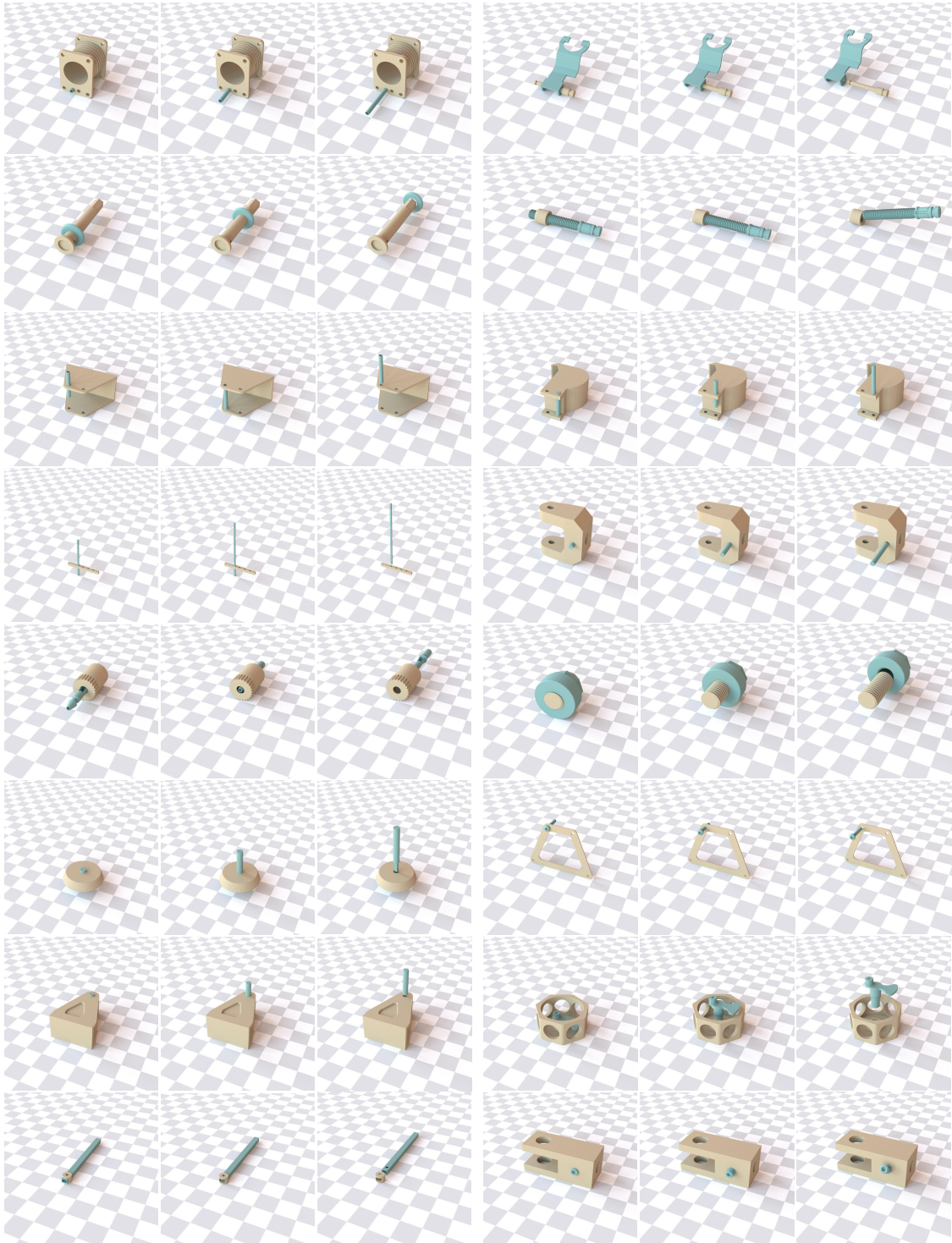


Figure 3.9: More successful examples of two-part disassembly motion produced by our method while failing on baseline methods.

Table 3.2: Success rate (%) comparison on the entire two-part assembly dataset along with specific comparison on different rotational assemblies.

Method	Two-Part Overall	Rotational			
		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

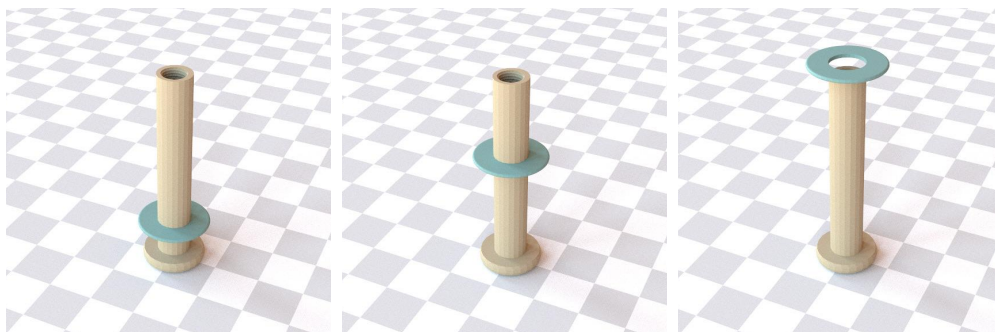


Figure 3.10: A tightly-fitted two-part disassembly (left: assembled, right: disassembled) solved by our method.

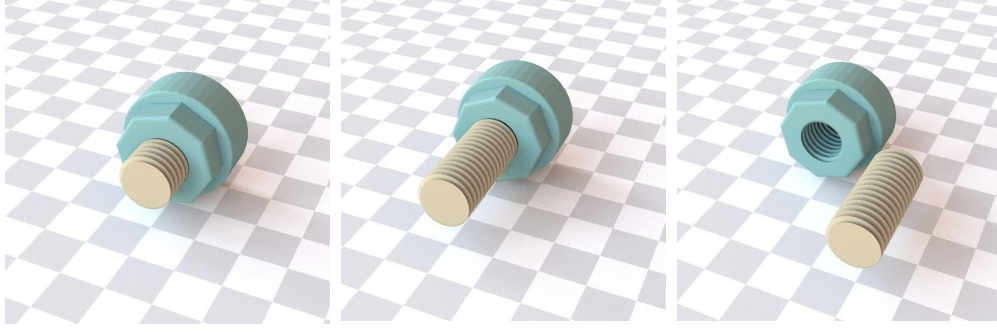


Figure 3.11: Screw disassembly (left: assembled, right: disassembled) solved by our method.

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 [15]; 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.

We report the results in Table 3.2 (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 margin in **Screw** and **Puzzle** assemblies while being worse than BK-RRT in the **Others** category. Figure 3.11 shows a screw example which can be successfully solved by our approach but fails on all other baseline methods. Our method successfully generates a long sequence of screwing motions and a short translation at the end to disassemble the screw from the nut.

For puzzle assemblies, Zhang et al. [15] 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.

3.4.3 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 3.1. 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 3.3. The results are produced with a 120s timeout for each path planning attempt and a 7200s timeout for the

Table 3.3: Success rate comparison (%) on multi-part assembly dataset.

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 3.4: Mean disassembly time per part (s) comparison on multi-part assembly dataset with different sizes of assemblies.

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

entire sequence planning, averaged from 3 random seeds. Table 3.3 shows that our method (either Full BFS or Prog. BFS) outperforms other baselines on all different assembly sizes.

Furthermore, our method is capable of disassembling 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 3.12) from [110] in 5 minutes while their MV+T-RRT takes hours to complete. We also empirically demonstrate in Table 3.4 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.

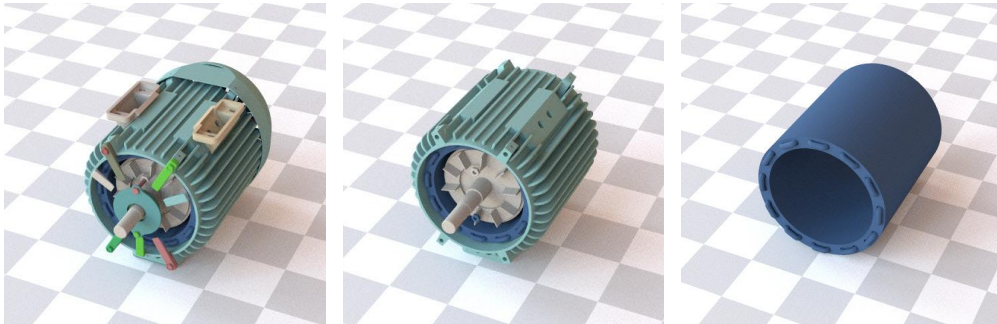


Figure 3.12: 53-part engine disassembly (left: 53 parts left, middle: 11 parts left, right: 4 parts left) solved by our method.

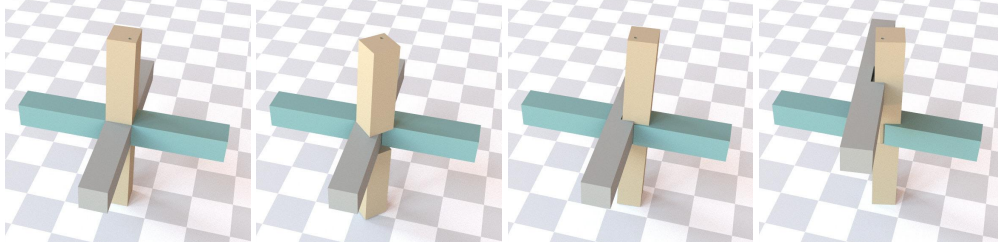


Figure 3.13: A 3-part interlocking assembly disassembled by the movement of 2 parts (left: assembled, right: disassembled).

3.4.4 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 3.13 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. 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.

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 interlocks. In this case, heuristics or learning-based methods can potentially be adopted to speed up the sequence search.

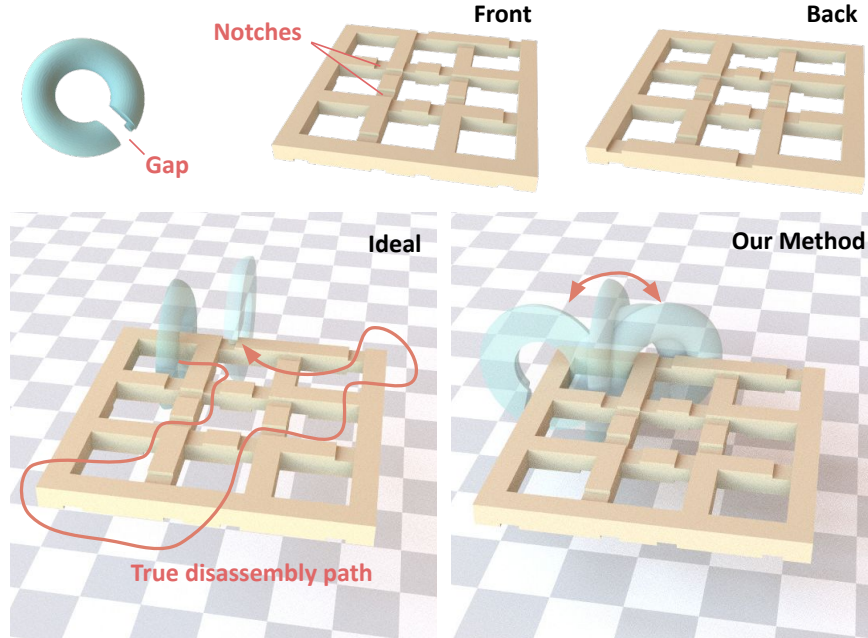


Figure 3.14: A failure case of our method disassembling a maze (geometries shown in the top row).

3.4.5 Failure Cases

In this section, we analyze the main failure cases of our method from the experimental results shown in Section 3.4.

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 3.14, we test our method on a challenging maze-like assembly from Zhang et al. [15]. 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 (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).

Computational complexity with dense contacts Both our method and the baseline approaches encounter failures on assemblies with excessively dense contacts. Such contact configurations often lead to a super-linear increase in computational cost for both simulation and sequence planning, causing the algorithms to exceed the time limit. Dense contacts can arise from two main factors: (1) large assemblies with many parts naturally produce numerous contact points, and (2) assembly meshes with extremely fine geometric details, such as threads on screws or highly detailed surface features, result in an excessive number of contact vertices. Figure 3.15 illustrates several cases in which all methods exceeded a 2-hour runtime limit.

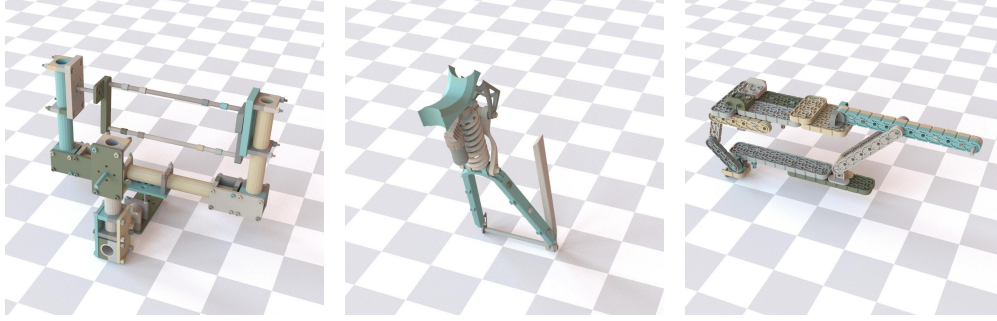


Figure 3.15: Large assemblies with several hundreds of parts and rich contacts where all tested methods reach the timeout of 2 hours.

Table 3.5: Distribution of failure cases in Assemble-Them-All aggregated over all benchmark assemblies.

Failure Case	Count	Percentage
a) Complex action sequences	27	23.7%
b) Dense contacts	87	76.3%

Summary and potential solutions Table 3.5 summarizes the distribution of failure cases across the benchmark assemblies for our algorithm. While this distribution may be influenced by biases in the dataset as it is potentially not representative of real-world assembly scenarios, it is noteworthy that dense contacts account for the majority of failures. This is largely because most assemblies in the dataset are designed to be straightforward to assemble and do not require long and complex action sequences. However, the assembly structure or mesh geometry details can still be arbitrarily complex, leading to excessively dense contact scenarios. It is also important to note that our benchmark does not include deformable objects, which represent another class of failure not addressed by our algorithm and thus are omitted from the table.

Failures of type (a) could potentially be mitigated by incorporating stronger geometric or visual reasoning to generate meaningful coarse action sequences, rather than relying solely on blind search within a physics-based framework. Failures of type (b) could benefit from more efficient and robust physics-based simulators capable of handling a larger number of contacts or adaptively selecting critical contact pairs for resolution.

3.5 Discussion and Conclusion

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, it is interesting to explore further the direction of physics-based planning for generalization to deformable assemblies, such as the snap-fit assembly.

Second, our current search method focuses exclusively on sequential, single-part assembly, without exploring opportunities for grouping parts or handling subassemblies—scenarios where multiple parts must move together for feasible assembly. Additionally, it does not support partial assembly states, such as partially assembling one component, transitioning to another task, and later returning to complete the initial component, a common pattern in complex assemblies.

Third, 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.

Fourth, for assemblies with many parts (e.g., > 100 parts), runtime rises significantly due to the quadratic time complexity w.r.t. the number of parts, and even explodes to factorial when sub-assemblies or interlocking motions are considered. The dominant factor in our method for such a high complexity comes from the naive enumerative sequence search. To address this, future work may develop more efficient heuristics or learned guidance for empirical speed up (which I will describe in the next section), or better leverage prior information from the designer or CAD structures beyond geometric meshes.

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. It is promising to extend the capability of performing complex assembly autonomously and flexibly on real robots, and we demonstrate convincing results in the next two chapters.

Chapter 4

Efficient Assembly Sequence Planning with Physical Feasibility

Chapter 3 demonstrated how powerful and accurate physics-based planning can generate reliable part motions for virtual assemblies. Transferring the method in Chapter 3 to the real world requires generating physically feasible assembly sequences (i.e., assembly sequences executable in the real world), a challenging research problem for several reasons: (1) Sequence Planning: The number of potential assembly sequences grows exponentially with part count, with not all sequences being equally good or feasible. (2) Physical Feasibility: Planned sequences may not be physically executable due to unsatisfied physical constraints. For example, a collision-free path might not exist for a given assembly order, or parts could fall during assembly due to instability. (3) Geometric Complexity: The geometry of objects can be highly complex, such as gears with numerous faces. Planning collision-free paths or assessing gravitational stability accurately in these scenarios is non-trivial.

We tackle these challenges with **ASAP** (**A**utomated **S**equences Planning for Complex **R**obotic **A**ssembly with **P**hysical **F**easibility). To guarantee physical feasibility of assembly sequences, we use physics-based simulation to check for collision-free paths. We also introduce an efficient method to verify gravitational stability and identify parts to be held for stability. Additionally, we find the most stable pose when placing the assembly on a support surface. To speed up planning and reduce the combinatorial search space, we employ the assembly-by-disassembly principle and efficient tree search algorithms. We guide the sequence search using geometric heuristics or graph neural networks (GNNs) trained on simulation-based data.

In summary, our work makes the following contributions:

1. An efficient assembly sequence planning algorithm for generating physically feasible sequences for complex-shaped, contact-rich assemblies, ensuring both collision-free paths and gravitational stability.
2. A learning-based approach for selecting the next part to disassemble, using GNNs trained on a large dataset of product assemblies.
3. A greedy approximate gravitational stability checking algorithm that more efficiently determining unstable parts for stabilization compared to the combinatorially expensive naive approach.

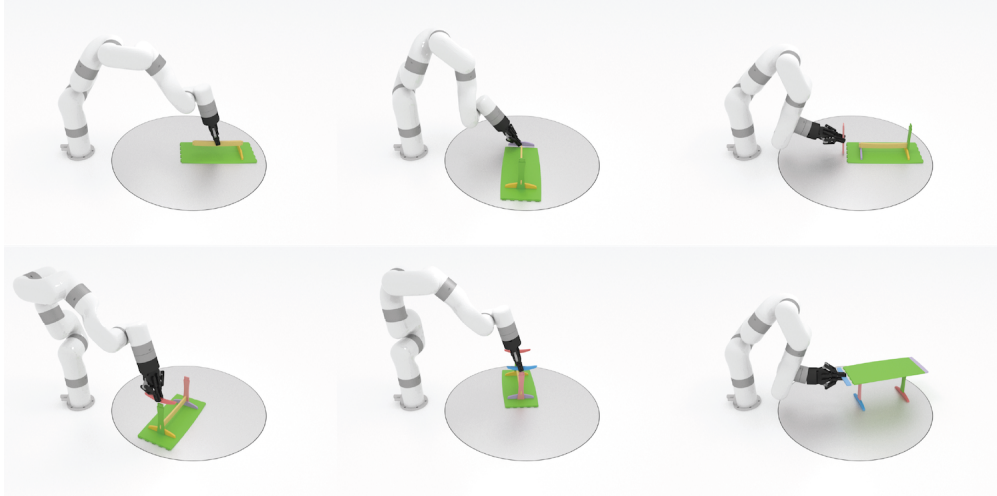


Figure 4.1: Assembly plans generated autonomously from ASAP for a desk positioned on a rotary table including physically feasible assembly sequences, collision-free paths, gravitationally stable poses, gripper grasps, and robot arm motion.

4. Evaluation on a large dataset of hundreds of complex product assemblies, demonstrating state-of-the-art performance compared to established baselines.
5. Integration of grasp planning and robot arm inverse kinematics, guaranteeing feasible robotic execution in both simulation and real-world experiments.

Our code, data, and videos are available at asap.csail.mit.edu.

4.1 Physically Feasible Sequence Planning

Our objective is to plan an assembly sequence under physical constraints using individual parts and their assembled states as input. The output sequence details each assembly step, including the part to assemble, its motion path, assembly pose, and other parts to be held. Solving them jointly yields a complete and physically feasible sequence.

Searching in a combinatorial sequence space is challenging, especially with realistic physical constraints. To reduce the complexity, we first employ the assembly-by-disassembly strategy (see Section 1.2.1). Next, in this section, we propose a tree search algorithm that efficiently prioritizes part disassembly by leveraging geometric heuristics or neural guidance, and identifies stable poses that satisfy physical constraints.

4.1.1 Disassembly Tree Search

We formulate the sequence planning as a tree-search framework, utilizing established techniques for efficient exploration. The core data structure is a disassembly tree, as shown in Figure 4.2. Each path of the tree represents a particular way of how the assembly is disassembled from top to bottom.

Algorithm 5 Physics-Based Disassembly Tree Search

Require: Assembly $G_0 = \{p_1, \dots, p_n\}$, evaluation budget N

Ensure: A physically-feasible disassembly sequence

```
1:  $T \leftarrow \text{EmptyTree}()$ 
2:  $i \leftarrow 0$ 
3:  $T.\text{AddNode}(G_0, \text{valid}=\text{true})$ 
4: while  $i < N$  and search not complete do
5:    $G \leftarrow \text{SelectNode}(T)$  ▷ §4.1.2
6:   for all part  $p$  in  $\text{SelectPart}(G)$  do ▷ §4.1.3
7:      $G' \leftarrow G \setminus \{p\}$ 
8:     if  $T.\text{HasEdge}(G, G')$  then
9:       continue
10:    for all pose  $s$  in  $\text{SelectPose}(G)$  do ▷ §4.1.4
11:       $P_a \leftarrow \text{CheckAssemblable}(G', p, s)$  ▷ §4.2.2
12:       $P_g \leftarrow \text{CheckStable}(G', s)$  ▷ §4.2.3
13:       $i \leftarrow i + 1$ 
14:      if  $P_a \neq \text{null}$  and  $P_g \neq \text{null}$  then
15:         $T.\text{AddNode}(G', \text{valid}=\text{true})$ 
16:         $e \leftarrow T.\text{AddEdge}(G, G', \text{valid}=\text{true})$ 
17:         $e.\text{StoreInfo}(s, P_a, P_g)$ 
18:        break ▷ feasible step
19:      if not  $T.\text{HasNode}(G')$  then
20:         $T.\text{AddNode}(G', \text{valid}=\text{false})$ 
21:      if not  $T.\text{HasEdge}(G, G')$  then
22:         $T.\text{AddEdge}(G, G', \text{valid}=\text{false})$ 
23:      else if  $|G'| = 1$  then ▷ complete sequence return  $T.\text{FindPath}(G_0, G')$ 
return null
```

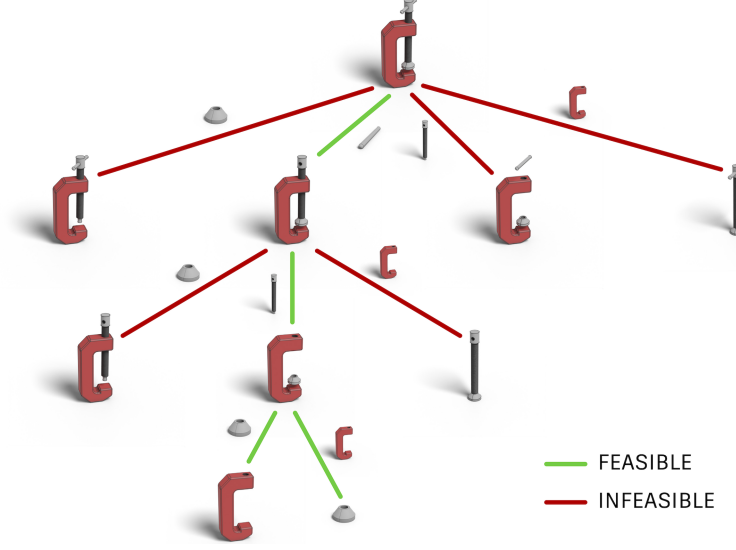


Figure 4.2: An example disassembly tree where nodes represent partial assemblies and edges represent feasible (green) and infeasible (red) disassembly actions.

Algorithm 5 outlines our disassembly tree search framework. Given the fully assembled assembly G_0 with n parts, our goal is to find a physically feasible disassembly sequence within the total simulation evaluation budget N . In our algorithm, we build and iteratively update a disassembly tree until a feasible sequence is found. The success of the algorithm relies on the following implementations to expand the tree in each iteration, as highlighted in Algorithm 5:

- **SelectNode(T)**: which node to expand tree T from (i.e., which partial assembly to disassemble).
- **SelectPart(G)**: which part to disassemble from node G .
- **SelectPose(G)**: under which pose to disassemble G .

In each iteration, we first select a node G from the tree T to expand, guided by the specific tree search method in **SelectNode(T)**. After that, we decide which part p to disassemble from G using **SelectPart(G)**. The remaining assembly without p is referred to as G' . Finally, we select a pose s to orient G and attempt to disassemble part p .

The feasibility of disassembly is determined by: (1) **CheckAssemblable()**, which gives a disassembly motion plan P_a (Section 4.2.2) and (2) **CheckStable()**, which gives a part holding plan P_g to ensure stability (Section 4.2.3). If both procedures return feasible solutions, the tree expansion is successful. Otherwise, we continue searching for new poses and different paths to expand the tree until either a feasible sequence is found, or the budget is reached, or the tree is fully expanded.

4.1.2 Node Selection

Any standard tree traversal method can be applied to select a node to expand in each iteration, i.e., **SelectNode()** in Algorithm 5. For simplicity, we apply a standard implementation of

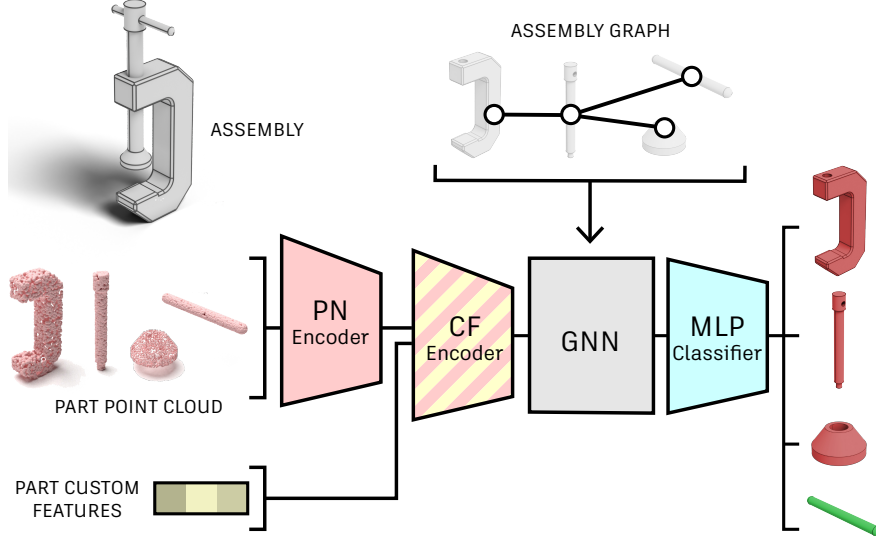


Figure 4.3: Network architecture for learning part disassembly priority. Given an assembly as input, a graph neural network is used to predict the next part to be removed.

depth-first search (DFS) that explores the tree as far as possible along each branch before backtracking.

4.1.3 Part Selection

Geometric Heuristic

To wisely select a part to disassemble, we leverage the insight that exterior parts are generally easier to disassemble due to fewer precedence constraints. We use the distance between a part’s center of mass and the assembly center (bounding box center) to prioritize outside-in part selection.

Learning-Based

We also implement a learning-based approach to accelerate the tree search described in Algorithm 5. Our network is trained by supervision on the task of node classification to predict the next part to disassemble. 6,368 disassembly sequence labels are generated from 1,906 training assemblies out of 2,146 total assemblies from the dataset described in Section 4.3.1 using simulation following a dynamic beam search to balance data diversity and computational efficiency. Inspired by previous works on learning for furniture assembly [121, 122] and computer-aided design (CAD) modeling [40], we implement a graph neural network (GNN) model that takes as input a graph G with n parts $\{p_1, p_2, \dots, p_n\}$ forming graph nodes connected by adjacency. The network outputs a probability distribution over all nodes, indicating the likelihood of each part being next in the disassembly sequence.

Figure 4.3 illustrates the network architecture consisting of four modules: a PointNet encoder (PN encoder) [123], a custom feature encoder (CF encoder), GNN message passing layers, and a multi-layer perceptron (MLP) classifier. The two encoders extract two kinds of

per-part features. The PN encoder extracts part geometric features from uniformly sampled point clouds (1K points per part), and the CF encoder provides additional information, such as part volume, part distance to the assembly center, and the number of adjacent parts. The two kinds of features together form a graph node embedding. Two rounds of message passing are used to update the graph node features between connected neighboring parts. Finally, the MLP classifier predicts the next part to disassemble from the final graph node embedding. We use a standard binary cross-entropy loss as the supervision signal between the predicted probability to disassemble a given part next $\hat{p}_i$ and the ground truth label p_i^{gt} indicating the next part to disassemble.

4.1.4 Pose Selection

Since it is impractical to evaluate infinite poses in a 3D space for a given assembly, we propose to use quasistatic stable poses [124] of assembly G as the pose candidates to select from (top-5 is empirically good enough), which provides a much reduced and reasonable search space. Furthermore, reusing poses during the assembly generates more intuitive sequences and also improves success rate.

4.2 Feasibility Check for Assembly Sequences

This section describes how we leverage physics-based simulation to check whether an assembly sequence is physically feasible. First, Section 4.2.1 describes our physics-based assembly simulation; Then, Section 4.2.2 focuses on finding collision-free paths for parts to be (dis)assembled using a physics-based path planning approach. Finally, Section 4.2.3 describes our method to check whether a given (partial) assembly can be gravitationally and frictionally stable, taking into account the support surface and a subset of parts to be held.

4.2.1 Physics-Based Simulation for Assembly

Our physics-based model is designed to work robustly with complex geometry for assembly and is able to handle minor problems in the input geometry, including small overlaps, potentially conflicting collision normals, and slight gaps. We use the penalty-based contact model for the contact normal force and the frictional forces [70, 71]. We use implicit Euler to integrate the system forward, which allows us to handle these penalty forces robustly and stably. The penetration distance is computed accurately and efficiently with pre-computed signed distance fields (SDF).

4.2.2 Disassembly Path Planning

To check whether a part can be assembled, we leverage again the idea of assembly-by-disassembly where disassembly path planning can be applied to connect the part’s assembled state to a disassembled state. Motivated by the recent success of physics-based path planning, we adopt [125] to generate a collision-free assembly path for each part. Specifically, the

planner starts from the assembled state and searches for a sequence of forces to apply until a disassembled state or a predefined budget is reached.

4.2.3 Gravitational Stability Check

Existing works using physics simulation to check stability are limited to simple block-stacking scenarios with a fixed pose on the support surface [34, 68]. However, complex real-world assemblies usually require at least two hands for assembly and re-orientation since the stable pose might change as more parts are assembled. To address this challenge, we develop an efficient physics-based stability check algorithm for a given assembly G with n parts under a specific pose s , represented as a homogeneous transformation matrix. The algorithm outputs the set of maximum M parts to be held to make G stable under pose s or returns null if it needs to hold more than M parts to stabilize G .

Algorithm 6 Physics-Based Stability Check

Require: Assembly $G = \{p_1, p_2, \dots, p_n\}$ with pose s , max M parts to be held, max simulation steps N , stable moving distance threshold d_{th}

Ensure: Parts to be held to make G stable with pose s

```

1:  $P_g \leftarrow \{\}$ 
2: while  $|P_g| \leq M$  do
3:   ResetSimulation( $G, s, P_g$ )
4:   for  $i$  in  $1, \dots, n$  do
5:      $\mathbf{q}_{i_0} \leftarrow \text{GetPosition}(p_i)$ 
6:      $stable \leftarrow \text{true}$ 
7:     for  $j$  in  $1, \dots, N$  do ▷  $N$ -step stability check
8:       Simulate( $\Delta t$ )
9:       for  $i$  in  $1, \dots, n$  do
10:         $\mathbf{q}_{i_j} \leftarrow \text{GetPosition}(p_i)$ 
11:        if  $\|\mathbf{q}_{i_j} - \mathbf{q}_{i_0}\| > d_{th}$  or  $\text{IsDisconnected}(p_i, G)$  then
12:           $stable \leftarrow \text{false}$ 
13:           $P_g \leftarrow P_g \cup \{p_i\}$ 
14:          break
15:        if not  $stable$  then
16:          break
17:   if  $stable$  then return  $P_g$ 
return null

```

A naive way to discover feasible part-holding plans is by trying all $O(n^M)$ combinations of parts to hold, which is impractical for large assemblies. Therefore, we introduce a greedy approximation with $O(M)$ part-holding plans to search in Algorithm 6. The key idea is to start with no parts held and only hold parts when we observe a part falling until M parts are held. We use P_g to denote the set of parts we hold to ensure stability, which is an empty set initially and gets iteratively updated. In each iteration, we obtain the initial position of all the parts, denoted as $\mathbf{q}_{i_0}$ for part p_i . Next, we run the physics simulation for maximum N time steps to check stability. In each step j , we obtain the updated position $\mathbf{q}_{i_j}$, and

Table 4.1: Success rate (%) comparison of ASAP on the assembly test dataset against several baseline methods.

Method		Success Rate (%) (Low Budget)			Success Rate (%) (High Budget)		
		2 Parts Held	3 Parts Held	4 Parts Held	2 Parts Held	3 Parts Held	4 Parts Held
ASAP (Ours)	Heuristics Learning	51.25	61.25	68.75	66.67	74.17	80.83
		54.58	62.92	69.58	67.08	76.25	82.08
Baseline	Random Permutation	14.58	25.42	41.25	27.92	43.33	55.42
	Genetic Algorithm [36]	14.17	25.83	40.00	30.83	41.25	51.25
	Assemble Them All [125]	19.17	27.08	35.42	30.42	46.25	56.67

calculate the Euclidean distance between $\mathbf{q}_{i_j}$ and initial position $\mathbf{q}_{i_0}$. If the distance is larger than a threshold d_{th} , this means part p_i is falling and thus not stable. Apart from that, we also check if part p_i loses contact with its neighboring parts; if so, it is also unstable. If any part is unstable in step j , we add part p_i to P_g and restart the whole process. The algorithm stops until either a feasible P_g is found, or we cannot hold extra parts ($|P_g| > M$). Note that we use position change as the stability check criterion because it is smoother in time for contact-rich simulation, as opposed to numerically noisy velocity or acceleration in previous works [34, 68].

4.3 Evaluation

4.3.1 Dataset for Assembly Sequence Planning

We build a large complex 3D assembly dataset derived from the Fusion 360 Gallery Assembly Dataset [40] and a real mechanical assembly dataset [41], consisting of 2,146 assemblies from 3 to 50+ parts with more complex shapes than those used previously for this problem. To ensure feasibility, we perform data filtering processes to make sure part meshes are watertight and able to be (dis)assembled without physical deformation, and remove parts that intersect with others in the assembly greater than a given distance. Among these assemblies, we select 240 assemblies to be the test set for benchmarking using the following procedures: First, we filter out all potentially unstable assemblies in the dataset by loading them into simulation and checking for stability using the top stable poses generated by Section 4.1.4. Next, we group the remaining dataset by the number of parts within the assembly and randomly sample up to 10 assemblies from each group to ensure a suitable representation of complex assemblies in the distribution. Finally, we combine all sampled assemblies to form the test set and the remaining 1,906 assemblies become our training set.

4.3.2 Baseline Methods

We compare ASAP with a naïve baseline and two established methods that generate and optimize (dis)assembly sequences: (1) *Random Permutation* generates completely random (dis)assembly sequences and evaluates them until the budget is met, to serve as a lower bound on performance. (2) *Genetic Algorithm* [36, 38] starts from randomly permuted sequences and

iteratively optimizes the sequences based on a *fitness function*, defined as the number of steps of the sequence that are physically feasible. (3) *Assemble Them All* [125] similarly leverages physics simulation to find assembly sequences by randomly selecting parts to disassemble and checking for collision-free disassembly paths. However it does not consider stability as a criterion, but assumes a gravity-free setting with no supporting surface and parts to be held. For comparison, we run *Assemble Them All* repeatedly to generate sequences until the budget is met. To take into account physical feasibility, we use our stability check algorithm in Section 4.2.3 under the top poses searched by the quasistatic pose estimator in Section 4.1.4 to evaluate the feasibility of the generated sequences for all baseline methods.

4.3.3 Experimental Setup

We benchmark two variants of our ASAP algorithm with different part selection methods (see Section 4.1.3). We run our experiments in parallel on Intel Xeon Platinum 8260 CPUs with 4GB RAM per core, where each experiment for a single assembly runs on a single CPU core. We set a 2-hour timeout budget for each experiment. For network implementation, all modules use 512 hidden neurons, ReLU activation, and batch normalization. The GNN uses two GATv2 [126] message-passing layers with eight heads. We withhold the test set described in § 4.3.1 and split the remaining data 90:10 for training and validation. Training takes 5 hours using an NVIDIA V100 GPU for 150 epochs, with a batch size of 160 and a learning rate of 0.001. We use the Adam optimizer with a scheduler that reduces the learning rate on the plateau.

4.3.4 Quantitative Success Rate Comparison

We now compare the performance of our algorithms against the baseline methods using the *success rate* metric: the percentage of assemblies from the test dataset that can be successfully (dis)assembled, given a fixed simulation evaluation budget and a set number of parts to be held. To represent a realistic scenario, we show results for both a low budget (50 evaluations) and a high budget (400 evaluations) in Table 4.1, holding 2, 3, and 4 parts respectively.

The results show that both variants of our algorithm outperform the three baseline methods by a significant margin, thanks to the tree search formulation and efficient guidance on part selection that greatly reduces the search space. We observe improvements for all methods if allowing more parts to be held or given more evaluation budget, though a setup with two part holders and a limited budget is more practical. The outstanding performance of our *Learning* variant shows that the neural guidance trained on a diverse set of assemblies with simulation-generated labels generalizes well to unseen and more challenging assemblies in the test set.

Theoretically, the runtime and memory usage may scale exponentially with respect to the number of parts in the worst case when the tree search has to be exhausted. In practice, we recorded the median runtime of ASAP with a single CPU thread on different sizes of assemblies, as also shown in Figure 4.4: ≤ 5 parts: 43s; (5,10] parts: 97s; (10, 20] parts: 431s; (21, 30] parts: 1705s; > 30 parts: 4190s. Significant speed up can be achieved through parallelization in several components of Algorithm 5. In terms of memory usage, each node in the tree search consumes approximately 800 bytes, while each edge requires $350 + N_G * 250$

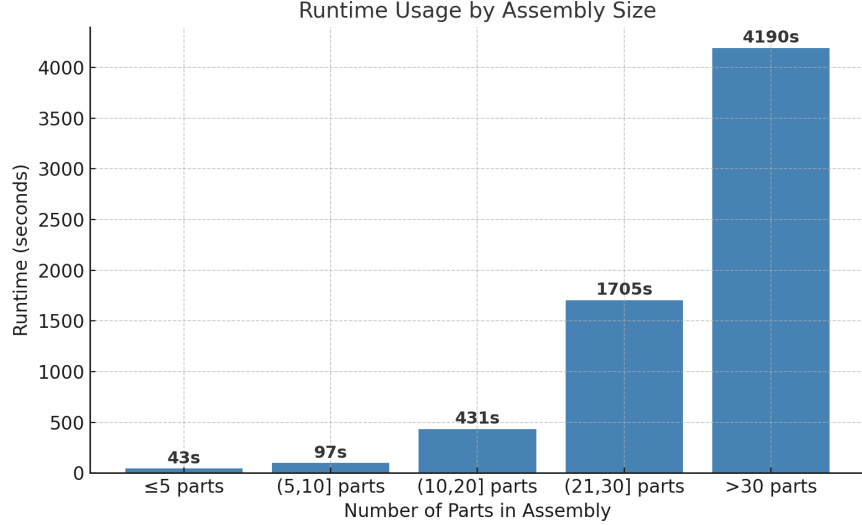


Figure 4.4: Runtime usage by assembly size (number of parts).

bytes, where N_G is the number of planned grasps (tunable by the user). Using $N_G = 25$ as a practical setting, each edge consumes roughly 6.6 KB. In the theoretical worst case (assuming unlimited compute), the memory footprint for a 20-part assembly could easily reach several GBs. In practice, however, feasible paths are often found before fully exploring the tree. Also, early termination from failed expansions prevents the tree from growing combinatorially, keeping memory usage for grasp storage in the KB–MB range. Overall, runtime is a more critical bottleneck than memory usage for our current implementation.

We conduct quantitative comparisons without constraints from robotic manipulators, which isolates the impact of planning strategies on assembly success while abstracting away the nuances of robotic hardware and specialized tools. For methods and results on assembly planning with ASAP using a parallel gripper and a robot arm, please see Section 4.3.7.

4.3.5 Qualitative Assembly Sequence Comparison

Figure 4.5 shows a comparison between assembly sequences generated by our method and *Assemble Them All*. As described in Section 4.3.2, *Assemble Them All* does not take gravitational stability into consideration in assembly sequence generation, thus producing sequences with many floating parts and apparently unstable poses. In contrast, ASAP generates physically feasible sequences with a stability guarantee, which is more applicable to a real-world assembly setup.

4.3.6 Ablation Studies

We compare our greedy stability check algorithm (Algorithm 6) to the full combinatorial check as the ground truth. We perform the comparison using 1,000 pairs of partial assemblies and poses randomly sampled from the data collected during simulation label generation. Table 4.2 shows the accuracy of our greedy approximation (TP = true positive, TN = true negative, FN = false negative). Due to the complexity of the combinatorial algorithm, we

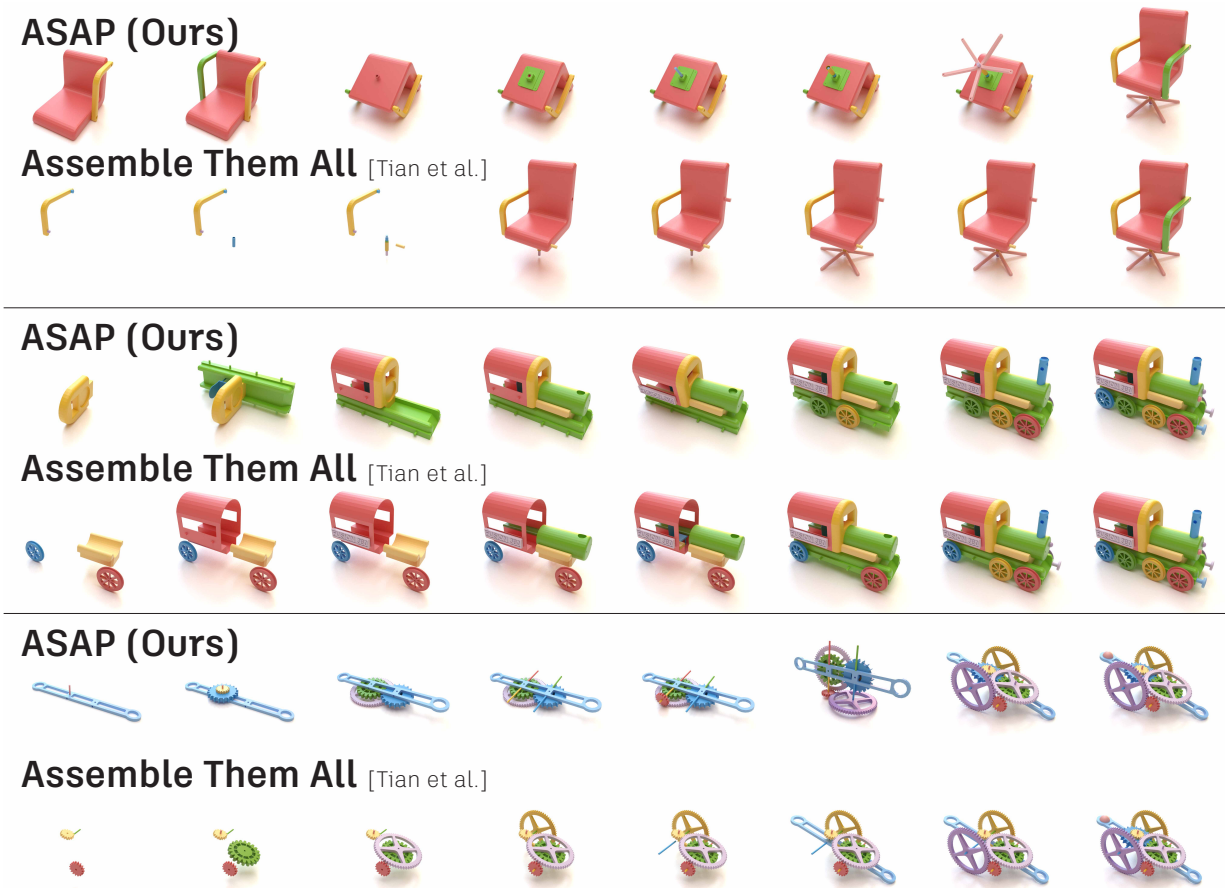


Figure 4.5: Assembly sequence comparison between ASAP and *Assemble Them All* [125] on camera, train, and gear set.

limit each comparison with a 1-hour timeout and report the percentage of timeouts. The greedy algorithm achieves around 90% accuracy and an order of magnitude speed-up, which becomes more significant as the number of parts that can be held increases due to a more expensive combinatorial check.

We also implemented a custom beam search for node selection with dynamically growing beam width, which performs similarly as DFS. Regarding geometric heuristics for part selection, we found the outside-in selection scheme introduced in Section 4.1.3 offers a slight advantage over prioritizing parts with small volume or few adjacent parts.

We conduct ablation studies to examine how training dataset size affects the GNN’s performance, with results summarized in Table 4.3. Specifically, we randomly sample 50% and 80% of the training data and evaluate in a 2-part held-out setting under both low-budget (50 evaluations) and high-budget (400 evaluations) regimes. Overall, the results reveal a positive scaling trend: more training data generally improves performance. However, the gain is not strictly monotonic. For example, in the low-budget scenario, using 50% of the data achieves the same performance as using 80%. This variability arises from the large diversity of assembly structures and geometries. Because of this vast design space, the training data distribution cannot be guaranteed to align closely with test distributions or real-world cases.

Table 4.2: Accuracy and speed up of the proposed stability check algorithm compared to the combinatorial ground truth.

# Parts to Hold	TP (%)	TN (%)	FN (%)	Timeout (%)	Acc. (%)	Speed Up
2	58.5	19.2	9.6	12.7	89.0	13.90x
3	66.3	8.4	7.8	17.5	90.5	17.03x
4	69.8	4.0	4.1	22.1	94.7	23.04x

Table 4.3: Success rate (%) comparison of ASAP on the assembly test dataset with reduced size of training dataset.

Method	Success Rate (%) (Low Budget)	Success Rate (%) (High Budget)
	2 Parts Held	2 Parts Held
100% data	54.58	67.08
80% data	49.16	65.00
50% data	49.16	62.92

Nevertheless, incorporating more diverse data typically enhances robustness and improves generalization to unseen assemblies.

4.3.7 Robotic Demonstration

We integrate ASAP with a robotic setup for real-world deployment. The motion of the robotic arm and gripper is governed by grasp planning, inverse kinematics (IK) and collision detection. To determine feasible grasping points on an object using a parallel gripper, we sample 100 pairs of antipodal points on its surface through ray casting and surface normal comparisons to ensure force closure. Each pair of antipodal points guides the placement of the gripper’s fingertips, along with the selection of 5 orientations that positively align with the assembly direction. Next, we execute the assembly path with the chosen grasping pose. At each step of the path, we calculate the robot arm’s joint angles via IK, using the gripper pose as input. A rotary table can be used to improve arm reachability, with the rotation angle optimized based on the desired gripper location and orientation. This process also includes thorough collision checking to ensure the robot arm, gripper, assembly parts, and the support surface remain collision-free. We prioritize grasp planning with poses that enhance stability and iterate until a collision-free and IK-feasible assembly plan is achieved.

Following the above approach, ASAP generates feasible robotic assembly plans for diverse assemblies (e.g., Figure 4.1). While comprehensive simulated results can be found on our [project website](#), here we demonstrate the sim-to-real transfer on a real-world hardware setup with a 3D printed beam assembly with 5 parts. The central component of the hardware setup is a UFACTORY xArm 7 robotic arm, which utilizes a Robotiq 2F-140 gripper. To aid

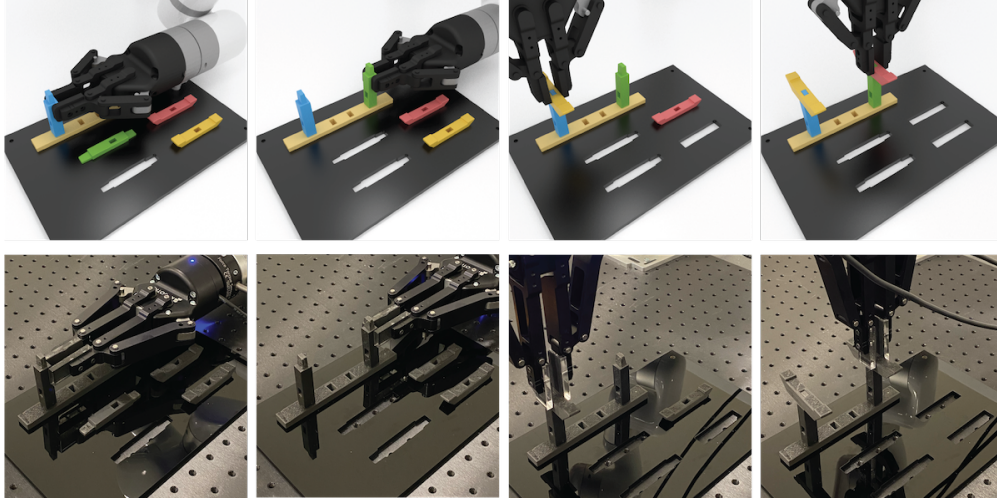


Figure 4.6: Robotic assembly plans of 3D printed beams generated by ASAP executed in simulation (top row) and real-world (bottom row).

in the spatial localization of assembly components, we use a laser-cut placemat that allows the robot to determine the precise positioning of parts, thereby reducing potential assembly errors.

The direct sim-to-real transfer is non-trivial because due to tight millimeter-level clearances in assembly joints for stability and inherent errors in part fabrication and arm localization. After extensive calibrations, in Figure 4.6, we show a successful step-by-step correspondence between simulated ASAP plans and real hardware execution. The sim-to-real transfer can be made more robust by incorporating vision or force feedback and adaptive manipulation skills [127, 128].

4.4 Discussion and Conclusion

We introduced ASAP, an efficient sequence planning algorithm for automated assembly that generates physically-feasible assembly plans for complex-shaped assemblies. We observe three notable failure cases of ASAP with ratios of 50%/45%/5% respectively:

1. Fully explored disassembly tree with no feasible sequence: This occurs in loosely assembled parts where stability is difficult to ensure or additional grippers are needed to handle unstable components. Such cases may arise from modeling inaccuracies, simplified product designs not intended for actual manufacturing, or assemblies requiring specialized fixtures or dedicated assembly lines.
2. Slow physics evaluation in complex contact dynamics: When assemblies contain many interacting components, the number of contact pairs can grow quadratically with the number of parts. Solving these contact constraints numerically becomes both computationally expensive and prone to instability as contact complexity increases.
3. Strict precedence constraints in large search spaces: Some assemblies permit only a

small number of feasible sequences within a vast search space. Under these conditions, ASAP may fail to identify a valid solution within a limited computation budget.

In future work, we plan to explore generating and executing more diverse assembly sequences in the real world using a more flexible robotic setup. To this end, we are interested to integrate different manipulation tools such as suction grippers and screwdrivers, and explore imitation learning or reinforcement learning to train physical assembly skills that are adaptable to real-life assembly tasks with uncertainties. In the next chapter, we will demonstrate a more powerful dual-arm robot setup with integrated planning and reinforcement learning for achieving more complex real-world assemblies.

Chapter 5

Real-World Dual-Arm Assembly via Integrated Planning and Learning

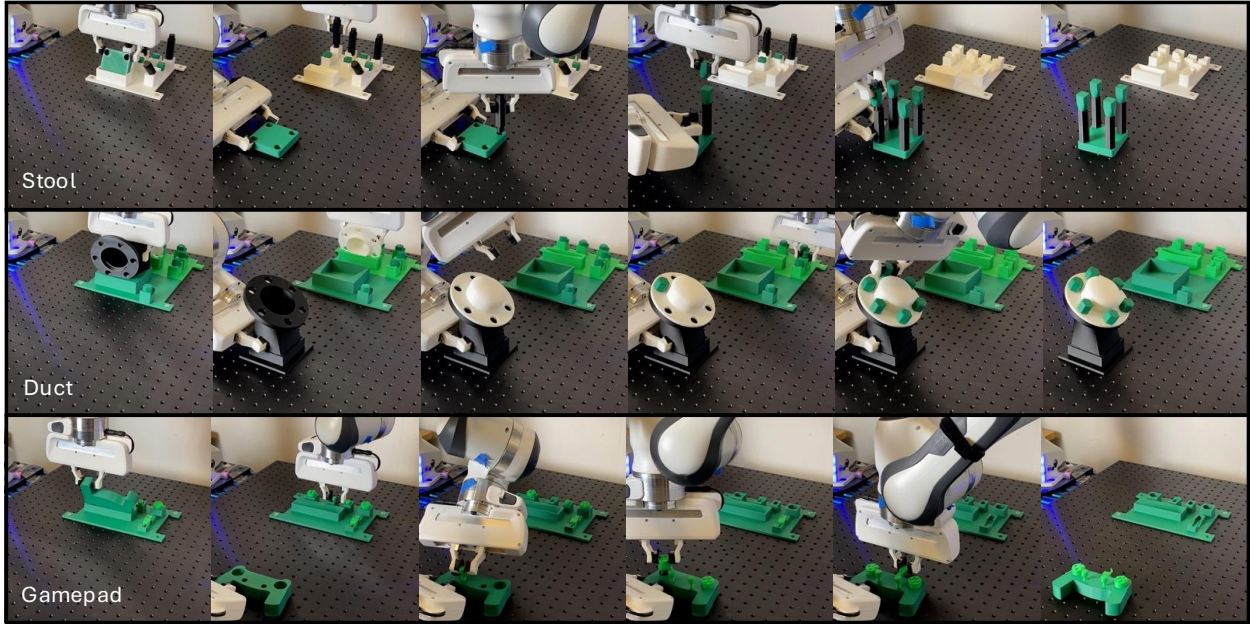


Figure 5.1: Our proposed dual-arm robotic system demonstrates adaptive manipulation and assembly capabilities for diverse multi-part objects.

With the ability to generate physically-feasible motions and optimized sequences from Chapters 3 and 4, we next turn to the challenge of executing these plans on real robotic platforms. This chapter presents an integrated planning and learning framework for dual-arm assembly, enabling robust multi-step manipulation under real-world uncertainties.

Robotic assembly of multi-part objects presents a longstanding challenge: long-term planning to map computer-aided design (CAD) models to robot programs and robust control skills to achieve high precision and adaptivity during contact-rich interactions. Despite recent progress in sim-to-real transfer of contact-rich part insertion skills [85, 87, 129], current robotic systems are still not capable of assembling general multi-part objects. Prior research

has primarily focused on two-part, top-down insertion using a single robot arm, such as demonstrated in Chapter 4, but multi-part assembly requires diverse insertion and grasping poses and a bi-manual operation that frequently changes which part to hold to counter-balance the insertion force from the other hand. This presents new challenges to planning and control. First, jointly finding an assembly-hold sequence, physically stable grasps, and collision-free robot motion presents a hybrid (discrete-continuous) optimization problem in a large search space. Second, control policies for part insertion must be robust to misalignment and uncertainty, while being able to generalize across a wide range of part geometries.

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

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

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

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

5.1 Planning Multi-Step Dual-Arm Assembly

5.1.1 Problem Formulation

Given a n -part assembly with parts indexed by $o \in O$, we compute a plan to manipulate all parts from the initial poses p_O^0 to the goal poses $p_O^G \in SE(3)$ under all physical constraints. We focus on sequential, collaborative manipulation that alternates between robot R_a assembling a part and another robot R_h holding a part to stabilize the sub-assembly. Then, we train control policies for precise contact-rich assembly steps. Finally, our system execution alternates between open-loop planned motions and closed-loop reactive policies. Figure 5.2 provides an overview of the system. Our system takes part meshes and hardware configurations as inputs. It plans sequences, grasps, fixture designs, and motions through a multi-stage planner, and learns RL policies for all insertion steps, which are deployed together on real robots to complete the assembly.

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

$$\min_{\phi, \sigma, \pi} E(\Phi_{i=1}^n \vec{f}(\phi_{1:i}, \sigma_{i-1:i}, \pi_i)) \quad \text{s.t.} \quad C_{\text{prec}}(\phi) \leq 0, C_{\text{kin}}(\phi, \sigma, \pi) = 0, C_{\text{col}}(\phi, \sigma, \pi) \leq 0 \quad (5.1)$$

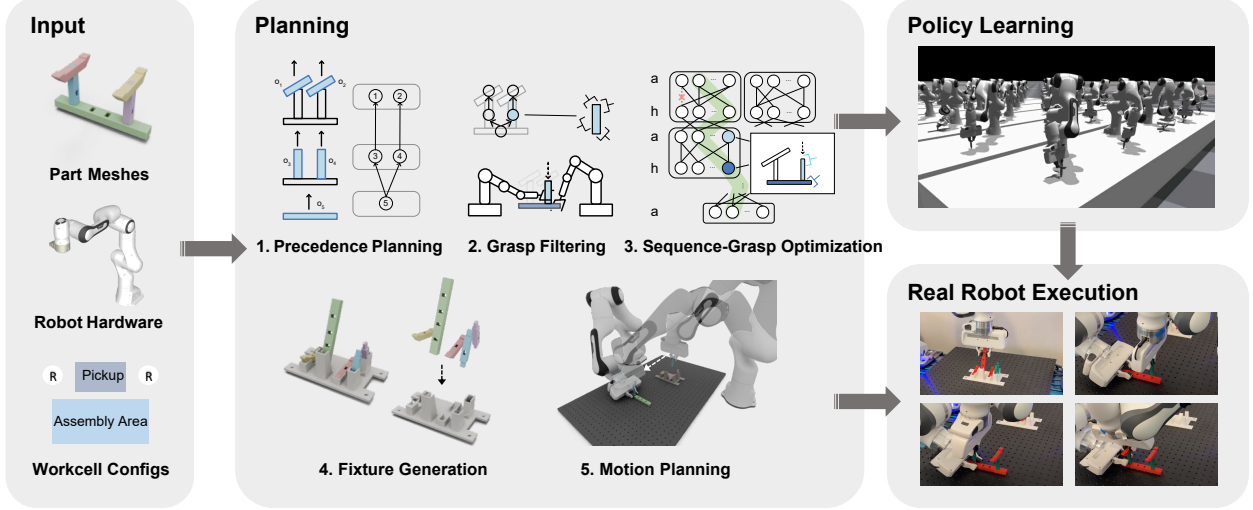


Figure 5.2: Overview of Fabrica, our proposed robotic system with integrated planning and learning.

The sequence $\phi = [o_{a,1}, o_{h,2}, o_{a,2}, \dots, o_{h,n}, o_{a,n}]$ is an ordering of parts to be held (h) and assembled (a), with $\sigma = [g_{a,1}, g_{h,2}, g_{a,2}, \dots, g_{h,n}, g_{a,n}]$ including grasp $g \in SE(3)$ for each step. The robot motion π is divided based on the mode families [130, 131] and skills: $\pi = [\underbrace{\tau_{a,1}^f, \tau_{a,1}^g, \tau_{a,1}^a}_{\pi[a,1]}, \underbrace{\tau_{h,2}^f, \tau_{a,2}^f, \tau_{a,2}^g, \tau_{a,2}^a}_{\pi[h,2]}, \dots]$ where each assembly task $\pi[a, i]$ for R_a contains

(1) a transit motion $\tau_{a,i}^f$ with its hand free, (2) a transfer motion $\tau_{a,i}^g$ grasping an object, and (3) an assembly motion $\tau_{a,i}^a$ for part insertion. A hold task $\pi[h, i]$ only involves a hold transfer motion $\tau_{h,i}^f$. The cost function composes step-wise objective vectors $\vec{f}$ that evaluate the quality of each step, $\Phi : \mathbb{R}^{|\vec{f}| \times n} \rightarrow \mathbb{R}^{|\vec{f}|}$ aggregates objectives across steps (e.g., sum or max), and E maps the result to a scalar (e.g., weighted sum). $C_{\text{prec}}, C_{\text{kin}}, C_{\text{col}}$ represent part precedence, kinematic, and collision constraints respectively. A more detailed breakdown of constraints is shown below.

$$\forall i \in [1, n],$$

$$C_{\text{prec}}(\phi_{1:i}) \leq 0 \quad (5.2a)$$

$$C_{\text{kin}}(o_{a,i}, g_{a,i}, \tau_{a,i}^g(0), p_{o_i}^0) = 0 \quad (5.2b)$$

$$C_{\text{kin}}(o_{a,i}, g_{a,i}, \tau_{a,i}^f(1), p_{o_i}^G) = 0 \quad (5.2c)$$

$$C_{\text{kin}}(o_{h,i}, g_{h,i}, \tau_{h,i}^f(1), p_{o_i}^G) = 0 \quad (5.2d)$$

$$C_{\text{col}}(\phi_{1:i}, \pi[a, i], \pi[h, i]) \leq 0 \quad (5.2e)$$

where each motion $\tau : [0, 1] \rightarrow \mathcal{Q}$ is a time-parametrized joint trajectory in the robot's configuration space $\mathcal{Q}$. eq. (5.2a) are the precedence constraints that ensure the partially assembled parts are connected and do not collapse under gravity. eqs. (5.2b) to (5.2d) ensure the robot configuration, grasp, and the grasped object pose are kinematically consistent when picking, assembling, and holding. eq. (5.2e) ensures that both robots' trajectories do not collide with previously assembled parts and other static obstacles.

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

Assumptions

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

- (A1) *Insertion-only assembly*: We assume that the mating between two parts only involves an insertion motion, without requiring skills like screwing or sliding.
- (A2) *No subassembly reorientation*: The final assembled positions of all parts are assumed to be given and fixed during the assembly process. This means that no further movement or re-orientation is allowed once a part is assembled.
- (A3) *Monotonic assembly*: Each part is only moved once, without considering regrasps, in-hand manipulation, or handovers.
- (A4) *No force and torque constraints for the robots*: We assume all parts are light compared to the robot payload.
- (A5) *A finite grasp set for each part*: $\forall g \in \sigma, g \in \mathcal{G}[o], |\mathcal{G}[o]| = N$, where $\mathcal{G}[o]$ can be computed by any grasp generator.

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

Planning Stages

In the following sections, we expand on the key planning stages:

- Section 5.1.2: Starting from a multi-part mesh model, we construct a precedence graph representing a minimum constraint set for any feasible sequence, considering only collision among parts.
- Section 5.1.3: To reduce online collision checking overhead during search, we precompute a discretized, collision-free grasp set for assembling and holding for all part pairs. Each feasible grasp is associated with a corresponding robot trajectory.
- Section 5.1.4: We leverage the precedence graph and precomputed grasp pairs to expand a state tree that contains all feasible part-grasp sequences and then search for an optimal part-grasp sequence that minimizes a grasp stability cost.
- Section 5.1.5: After the grasps are determined, we develop an automatic design algorithm to generate a pickup fixture, so that the planned grasp can be achieved easily without the need of a re-grasp.

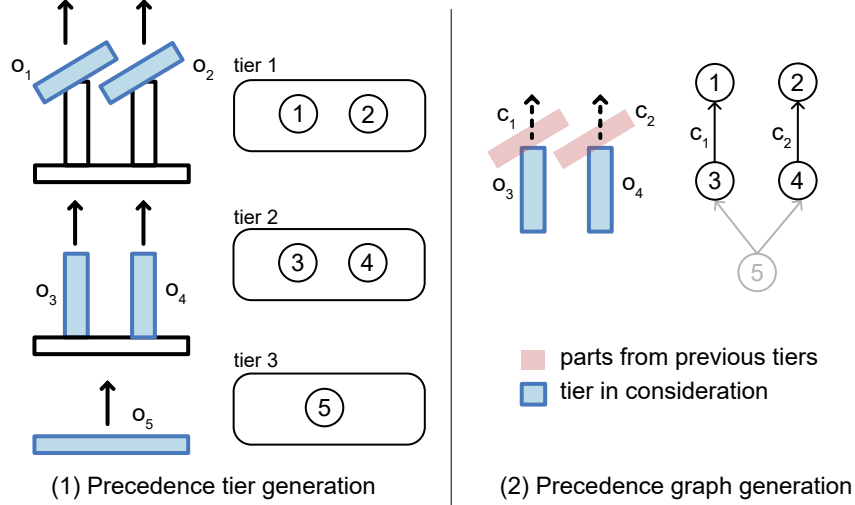


Figure 5.3: Illustration of our part precedence planning procedure.

- Section 5.1.6: With the assembly sequence fixed and all robot configurations determined for kinematic switches, i.e., pick-up, assembly, and hold, we plan for all transit and transfer motions.

5.1.2 Part Precedence Planning

While we introduced the part sequence planning in earlier chapters, we focused on discovering a *single* precedence-satisfying sequence. Here, to evaluate constraint C_{prec} dynamically without additional costs, we propose an algorithm to determine the *complete* precedence relationships for assembling all parts O . The process consists of two key steps, as shown in Figure 5.3: (1) Precedence tier generation, where parts are sequentially categorized into disassemblable tiers based on physical feasibility, and (2) Precedence graph generation, which analyzes fine-grained blocking relationships between tiers to construct a directed acyclic graph representing the disassembly dependencies. We now describe the complete technical details in the following paragraph.

First, we define precedence tier as a group of parts that can be removed independently of one another. Tiers are ordered so that parts in earlier tiers must be disassembled before those in later ones. To iteratively construct all tiers, we use a physics-based motion planner [125] to find all parts that can be disassembled without interfering with the rest, which are grouped into the current tier. We then remove these parts and repeat the process on the remaining assembly until each part is assigned to a tier. Next, we build a precedence graph G_{prec} that encodes the minimal set of ordering constraints that any collision-free assembly sequence must follow. Each node in G_{prec} is a part, and a directed edge $o_i \rightarrow o_j$ means that o_i must be assembled before o_j because o_j blocks the (dis)assembly path τ_{o_i} planned during tier generation. For each part o , we define its precedence set $O_{\text{prec}}[o] = \{o' | (o' \rightarrow o) \in G_{\text{prec}}\}$ as all parts that must be assembled before it.

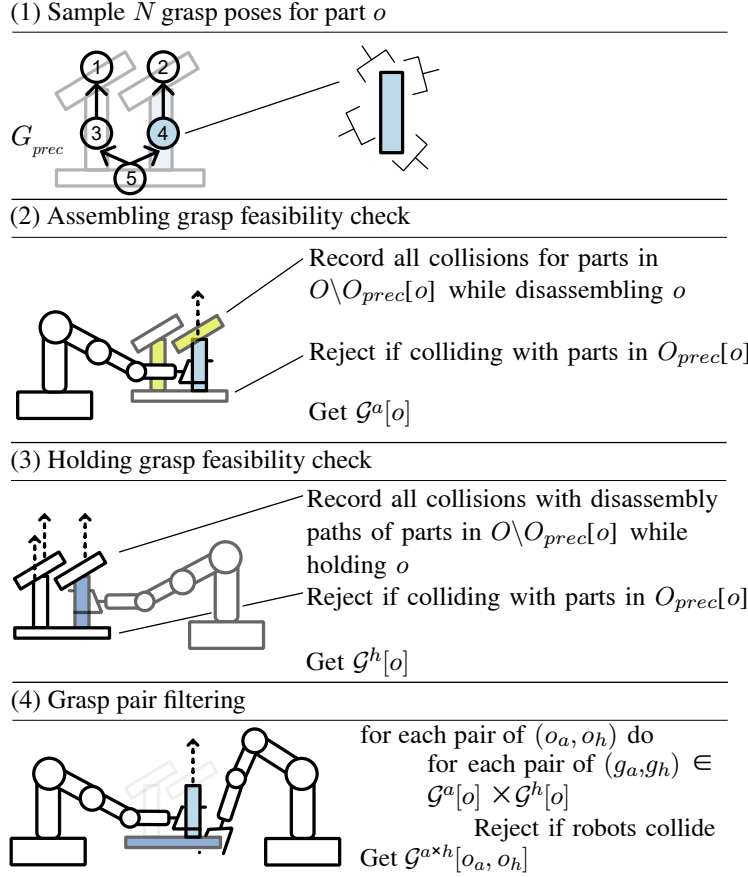


Figure 5.4: Illustration of the four stages of the dual-arm grasp filtering process.

5.1.3 Dual-Arm Grasp Filtering

We aim to identify valid grasp pairs $\mathcal{G}^{a \times h}[o_a, o_h]$ for each assembly-hold part pair (o_a, o_h) which support insertion and holding without colliding with preceding parts of o_a and o_h . Since searching the full 6-DoF space is infeasible, we assume feasible grasps exist in a dense, finite set of grasps, see (A5). Because each grasp must be checked for collisions with the current subassembly $\psi_{1:i}$, doing this online results in repeated and expensive checks against many part combinations. To speed up, we precompute valid grasps offline by sampling grasp candidates and performing parallelized inverse kinematics (IK) and collision checks for both arms. In practice, we simulate the robot following the part’s disassembly path τ_o and reject it if the motion collides with the precedence set $O_{prec}[o]$. All collisions and motions are recorded for reuse in later stages. Similarly, we check if a grasp can securely hold a part while disassembling other parts without collision. Finally, for each part pair (o_a, o_h) , we filter feasible grasps (g_a, g_h) by checking interarm collisions under computed IK. See Figure 5.4 for visual illustrations.

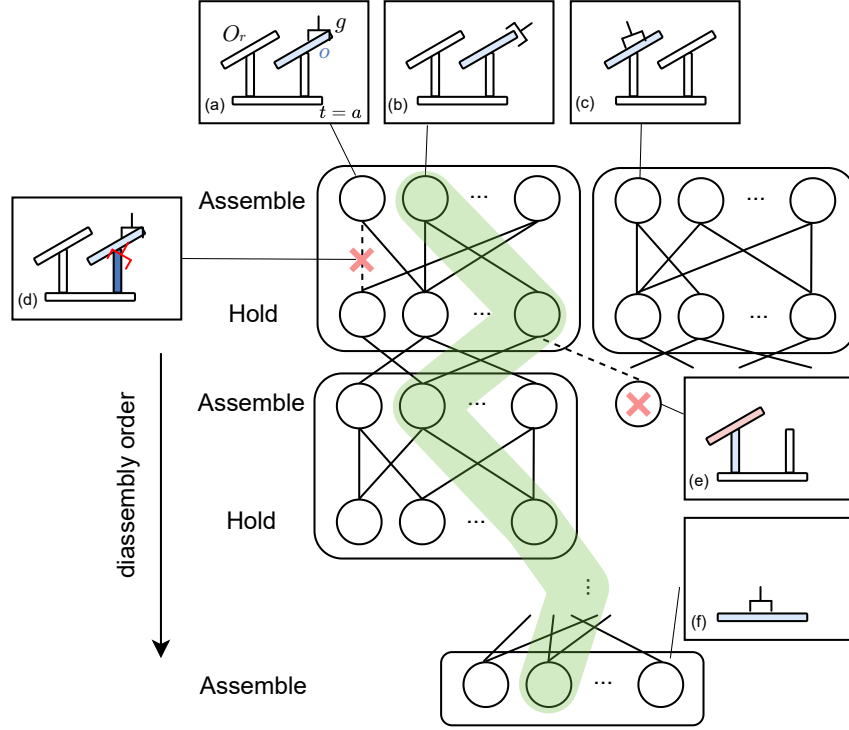


Figure 5.5: State tree for optimizing the part-grasp sequence.

5.1.4 Dual-Arm Sequence-Grasp Optimization

With all valid grasps computed, we now solve for the optimal sequence ϕ^* and grasps σ^* in Eq. 5.1. We formulate this as a state-space search problem and construct a directed state tree T_G , where each node represents a partial assembly state $s = (t, O_r, o, g)$ consisting of robot task t (assemble or hold), assembled parts O_r , part being grasped o , and grasp pose g . Starting from root nodes (complete assembly), we recursively expand the tree by alternating between assembling and holding, pruning states that violate constraints. Valid transitions must also respect precomputed grasp feasibility between successive steps. All collision and motion feasibility checks are reused from the earlier filtering stage. Each transition is scored by a grasp stability vector $\vec{f}$, capturing objectives such as supportiveness of the held part, frequency of grasp switches, torque stability, and contact area, which are designed to be lightweight yet effective for downstream control. We apply dynamic programming (DP) to propagate the best cumulative scores through the tree and identify the optimal solution. Please find more details in App. B.3.

Figure 5.5 presents a visual illustration of the process. (a-c): example root states (t, O, o, g) , with the task assignment t , the partial assembly O (white parts), selected assembly part o (blue part), and grasp g . (d): an edge pruned by an invalid assembly-hold pair that does not belong to the precomputed feasible set $\mathcal{G}^{a \times h}[o_1, o_{h,1}]$. (e): a state pruned due to violation of precedence constraint. (f): a leaf state node that represents the last part being disassembled. The optimal sequence that minimizes the summation of edge costs is highlighted in green.

We use the following formulas to evaluate the transition edge cost:

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

$$\frac{\|\boldsymbol{\tau}_{\text{part}} + \boldsymbol{\tau}_{\text{grasp}}\|}{N_{\text{grasp}}} \quad (5.3)$$

Where:

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

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

Where:

- $\mathbf{r}_i, \mathbf{r}_j$ are the position vectors of contact points on the part and grasp, respectively.
- $\mathbf{c}_{\text{part}}$ is the center of mass of the part.
- $\mathbf{d}_{\text{contact}}$ is the contact direction vector.
- N_{grasp} and N_{part} are the number of contact points for the grasp and part, respectively.
- Maximizing part contact area for holding grasps, weighted by the orientation difference from the paired assembling grasps (f_4):

$$\Delta\theta_{\text{rotation}} \times N_{\text{hold}} \quad (5.4)$$

Where:

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

Where:

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

5.1.5 Grasp-Aware Pickup Fixture Generation

For precise pickup, we develop a software-hardware co-design approach to automatically generate a fixture that stabilizes and orients each part for top-down pickup, based on planned grasps in Section 5.1.4. This removes the need for reorientation or regrasping between pickup and assembly, allowing the system to focus on the core assembly challenges. We first determine each part’s pickup pose in the world frame, with its orientation defined by the rotation from the assembly grasp to a top-down grasp. We then compute pickup positions by packing parts on the XY plane to avoid collisions between parts and the gripper. To reduce material usage and workspace area, we model this as a bin-packing problem and solve it with an iterative algorithm that alternates between packing, collision checking, and resolution. Finally, we generate the fixture by creating mold cavities based on the part geometries and poses, ensuring stable placement. See examples in Figure 5.6.

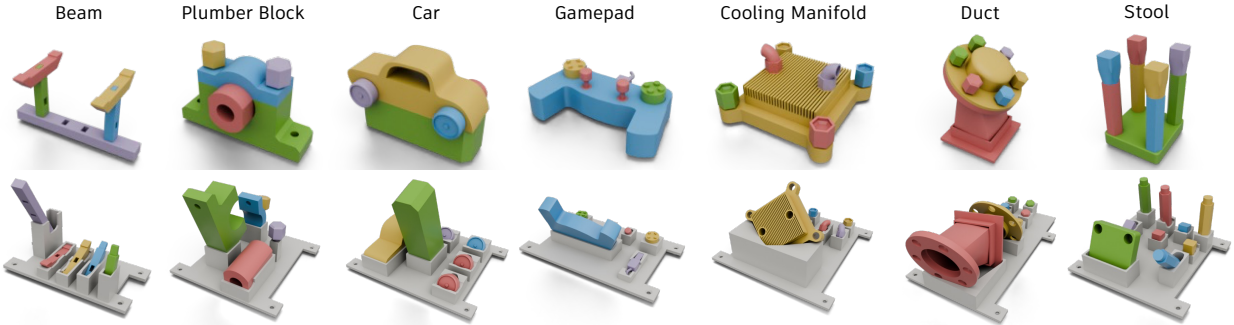


Figure 5.6: Top: benchmark assemblies. Bottom: the auto-generated pickup fixtures in Section 5.1.5.

Algorithmic Details

The fixture generation process begins with the computation of an appropriate pickup pose for each part. Our goal is to enforce a top-down pickup grasp without requiring re-grasping during the transition from pickup to assembly. Therefore, we can derive the pickup orientation of each part given the final assembled pose of them and the optimal grasp planned at the assembled pose, since we maintain the same relative transformation between the part and the gripper from pickup to assembly.

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

Determining the XY positions of the parts is more challenging, as the layout directly impacts the fixture area, which should ideally be minimized to reduce material costs for fixture fabrication and maximize the available workspace within the workcell. Additionally, incorrect part layout on the XY plane can lead to potential collisions between the gripper

and the remaining parts during pickup. Since the orientation of each object is locked, we can represent each part using a rectangular bounding box of its 2D-projected contour. Then, the problem becomes a 2D bin-packing problem, a classic problem with both heuristic and exact algorithms exist [132]. We use a simple algorithm that iterates through the following phases:

1. We use the Maximal Rectangles algorithm [133] to pack the bounding boxes into an initial bounding area;
2. We check the collisions between the gripper and the precedent parts of the grasped part;
3. If any collisions are detected, the colliding parts are buffered with additional spacing and the bin-packing process is performed again;
4. We increase the bin area once it is not enough to find a packing solution given the increased rectangle sizes.

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

5.1.6 Motion Planning for Transit and Transfer

Finally, RRT-Connect [111] plans for all remaining transit and transfer motions, i.e., $\tau_{a,i}^f, \tau_{a,i}^g, \tau_{h,i}^f$, which can be parallelized since all start and goal states of motions are provided from earlier stages.

5.2 Learning General Single-Step Assembly Policy

Once the full assembly plan is computed, the next challenge is to track it reliably in the real world. We use a hybrid controller that alternates between tracking the pre-planned transit and transfer motions, and an RL-based reactive controller for contact-rich assembly steps. The controller must generalize across variations in object geometry, grasp poses, and assembly directions. To this end, we design a lightweight yet highly effective RL framework for training a generalist assembly policy. Given a pre-planned insertion path τ_o , our goal is to guide the part from its noisy initial pose $\hat{\tau}_o[0]$ to the goal pose $\hat{\tau}_o[1]$, accounting for uncertainty in grasp and mating part geometry. We frame this as a Markov decision process (MDP) and learn a policy $\pi : \mathcal{O} \rightarrow \mathbb{P}(\mathcal{A})$ that maximizes expected rewards over time $\mathbb{E}_\pi[\sum_{t=0}^{T-1} \gamma^t r(s_t)]$. We use

proximal policy optimization (PPO) [134] to train a stochastic policy in simulation, which is then transferred to the real world without additional fine-tuning.

5.2.1 Path-Centric Coordinate Transformation

Humans naturally reuse the same assembly skills across different objects, regardless of their poses or motions. We emulate this ability by designing a problem space transformation [135] with SE(3)-equivariance, which maps all straight-line assembly motions in the world frame into equivalent top-down insertions in the task frame, allowing the RL agent to perceive them in a unified way.

Formally speaking, given an assembly path τ with a pre-assembly position p_d and the assembled position p_a , we define a path-centric transformation $\mathbf{T}$ conditioned on τ such that $\mathbf{T}(p_d) = (0, 0, 0)$ and $\mathbf{T}(p_d) - \mathbf{T}(p_a) = (0, 0, \|p_d - p_a\|)$. Thus, the agent’s observation is the transformed position of the part being assembled (under unknown noise), and its action is the transformed delta position, i.e., the ideal position change in the path-centric frame. Thanks to equivariance, the agent only needs to learn top-down insertions and can omit orientation from both observations and actions, simplifying the learning setup. We use a task-space impedance (TSI) controller to enable smooth and compliant insertions, with impedance gains similarly transformed into the path-centric frame to maintain consistent behavior across different assembly directions. This design ensures that the observation and action spaces are minimal yet essential, facilitating generalizability among different assembly tasks, and is transferable to different robot arms and end effectors.

5.2.2 Plan-Guided Residual Action

Guidance from the planned open-loop action improves learning by injecting prior knowledge about the coarse assembly direction. Thus, we adopt the idea of residual action [91, 136] in RL, where the policy outputs only the corrective action on top of the open-loop action, allowing the policy to focus on refining the assembly rather than learning the full assembly behavior. In practice, residual action warm-starts policy learning and typically leads to faster and better convergence.

5.2.3 Minimalist Reward Design and Sim-to-Real Transfer

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

Due to the sim-to-real gap from misaligned dynamics, we adopt 1) domain randomization with 3mm-noised initialization on object pose during training and 2) Policy-Level Action Integrator (PLAI) [84] during deployment to ease the sim-to-real transfer of RL policies, which improves action consistency by incrementally applying policy outputs to the last desired state instead of the current state. PLAI applies policy actions as $s_{t+1}^d = s_t^d \oplus \Pi(o_t)$, where s_t^d represents the desired state at time t , $\Pi(o_t)$ is the policy action computed based on the

current observation o_t , and $\oplus$ denotes the composition operation, instead of the nominal approach $s_{t+1}^d = s_t \oplus \Pi(o_t)$ which is prone to error accumulation.

5.3 Experiments

5.3.1 Benchmark Suite and Experimental Setup

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

Hardware Setup

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

Simulation Environment

We use RedMax for simulation-based planning, and Isaac Gym for reinforcement learning. Grasp feasibility is determined by sampling 100 candidate grasps per part using an antipodal grasp planner, followed by inverse kinematics validation and collision checking in RedMax. Precedence tier planning uses a physics-based disassembly planner [125] with orthogonal force directions for motion planning.

Training Configuration

Assembly policies are trained using PPO from the RL Games framework with key hyperparameters for RL presented in Table 5.1, which we use for all reported RL experiments. Our generalist policies are trained for a maximum of 5×10^7 steps (or equivalently 1500 iterations) with a parallel rollout setup using 1024 environments, which takes less than 2 hours on a single NVIDIA RTX A6000 GPU.

5.3.2 Real-World Deployment

Robot control alternates between executing planned transit motions in joint space and reactive policy execution for insertion steps. For policy execution, a task-space impedance controller

Table 5.1: Key RL Hyperparameters.

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

is used with gains $K_p = [800 \text{ N/m}, 800 \text{ N/m}, 400 \text{ N/m}]$ and $K_d = 2\sqrt{K_p}$, transformed into the path-centric frame to ensure consistent compliance. Control runs at 30 Hz. During deployment, we use the leaky Policy-Level Action Integrator (leaky PLAII) scheme for improved stability with an action scale of 0.001 and an error threshold of 0.02. We use the same set of control parameters across all benchmark assemblies. Interventions are requested after three consecutive failures to insert, detected via joint deviation thresholds and motion stagnation.

5.3.3 Planning Multi-Step Assembly in Simulation

We evaluate our planner from three distinct perspectives: efficiency, optimality, and generality.

Efficiency Table 5.2 shows the breakdown of planning time by stages for different assemblies. Our overall speed is on the order of minutes to solve for optimal plans given efficient parallelization. The runtime exhibits a general scaling trend with respect to the number of parts in an assembly. However, this relationship is not strictly monotonic and assemblies with more parts are not always harder to plan. Planning time is distributed across multiple stages, each with its own notion of difficulty. For example, grasp planning is more efficient for assemblies with broad, smooth surface areas compared to those with highly curved geometries; motion planning is easier for loosely coupled assemblies with decent free space; and sequence planning is simpler when strict precedence constraints are present rather than when many alternative assembly orders exist. As a result, understanding runtime scaling requires a deeper analysis or definition of assembly complexity beyond part count alone. Still, part count remains an informative and influential factor that strongly correlates with runtime.

Optimality Table 5.3 shows the objective scores of optimized sequences surpassing the random ones by a large margin with priorities from f_1 to f_4 (see App. B.3 for details on the score definitions).

Table 5.2: Planning runtime breakdown of each assembly. Stages marked with * are parallelized, while others have yet to be parallelized.

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

Table 5.3: Objective comparisons between optimal and random sequences. Higher is better for f_1 , f_4 ; lower is better for f_2 , f_3 .

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

Table 5.4: % of successful steps without intervention in simulation evaluations.

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

Generality Please see App. 5.3.5 for visual demonstrations of planning with different robot arms (Panda, xArm7, UR5e) and grippers. Our planning framework is general to any given hardware.

5.3.4 Learning Single-Step Assembly in Simulation

We use Isaac Gym [65] for training RL policies and performing simulation evaluations, with the PPO code from RL Games [137]. Table 5.4 presents the average % of successful steps for assembling our benchmark assemblies in simulation across 1024 random trials using different methods:

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

The results show that open-loop tracking exhibits the lowest success rates across all assemblies, indicating its limitations in handling uncertainties and variations. The AS policy

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

5.3.5 Executing Multi-Step Assembly in Real World

Quantitative Results

Table 5.8 shows the % of successful steps on benchmark assemblies evaluated in the real world (step-wise statistics), and Table 5.9 shows the multi-step cumulative success rates with 0/1/2 total interventions for failure recovery (overall statistics). All numbers are averaged across three complete multi-step real experiments, which translate to thousands of total assembly steps. We deploy stochastic policies with state-based success detection and allow up to three trials per step until success.

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

Baseline Since Fabrica is the first to assemble general multi-part objects with only CAD input, identifying a comparable SOTA baseline is challenging. The closest work is ASAP [2], which performs single-arm kinematic feasibility search without sequence/grasp optimization or closed-loop control. To compare, we adapted it by planning with dual-arm and adding our RL policy. Without optimized part sequencing and grasping, ASAP performed substantially worse, often struggling even with two interventions, which emphasizes the contributions of our planning optimizations.

Ablation: Effectiveness of Sequence-Grasp Optimization To understand how much optimizing part sequences and grasps helps, we conduct ablation studies on our method by removing those optimizations respectively. We observed that suboptimal sequencing often caused instability due to inadequate support of critical neighboring parts and more part drifts due to unnecessary re-grasps. Meanwhile, suboptimal grasp selection frequently caused part slippage due to insufficient contact area or inadequate resistance to external torques. Thus, our planner inherently accounts for control-level uncertainties, and results demonstrate that selecting effective part sequences and grasps significantly enhances assembly reliability. For

Table 5.5: Ablation studies on the impact of coordinate and action design choices on the assembly specialist (AS) policy’s performance across 1024 randomized simulation evaluations.

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

control, we observed that our path-centric transformation is crucial for generalizing across varying assembly directions. Policies trained without it perform significantly worse when multiple directions are involved.

Ablation: Impact of Coordinate and Action Design We conducted ablation studies to evaluate the impact of coordinate frame selection (world vs. path-centric) and action representation (nominal vs. residual) on the assembly specialist (AS) policy’s performance in simulation. Table 5.5 presents the average % of successful steps without intervention across 1024 randomized simulation evaluations for each assembly task under four configurations.

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

Ablation: Effect of Number of Trials In real-world experiments, we evaluate the assembly specialist policy with varying numbers of allowed trials (1, 2, or 3) per step, due to the stochastic nature of the policy. Table 5.6 summarizes the average % of successful steps without intervention across three end-to-end multi-step assembly runs. Results indicate that permitting additional trials substantially boosts success rates. The policy consistently improves as trial count increases, suggesting that the policy benefits from repeated attempts to refine alignment and correct minor positional errors.

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

Table 5.6: Scaling effect of number of policy trials across 3 complete end-to-end multi-step real-world evaluations.

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

Qualitative Results

In addition to Figure 5.1, Figure 5.7 and Figure 5.8 show qualitative results of our system assembling a variety of multi-part objects using different robot arms, both in simulation and in the real world. Once the hardware configuration is provided, our planner works seamlessly with a wide range of robot arms and end-effectors without requiring additional tuning.

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

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

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

Failure Case Analysis

We observed a noticeable gap between simulation and real-world performances. Thus, we present a detailed analysis of common failure cases shown in Figure 5.9 with a quantitative breakdown in Table 5.7: a) Parts slip from gripper pads during insertion attempts; b) Cumulative displacement error accrued during the multiple steps of assembly; c) The holding gripper is not modeled during RL training, blocking the part insertion attempts in the real world; d) Assembled parts become unstablized (partially disassembled) during insertion of other parts. Among all failure cases, the cumulative assembly shift is the predominant issue. We assume a 3mm noise in the simulation given that the base part is stably held. However, many sources of real-world uncertainties and multi-step accumulation lead to more significant errors than those simulated. We propose one remedy in Section 5.3.6, but robustly eliminating

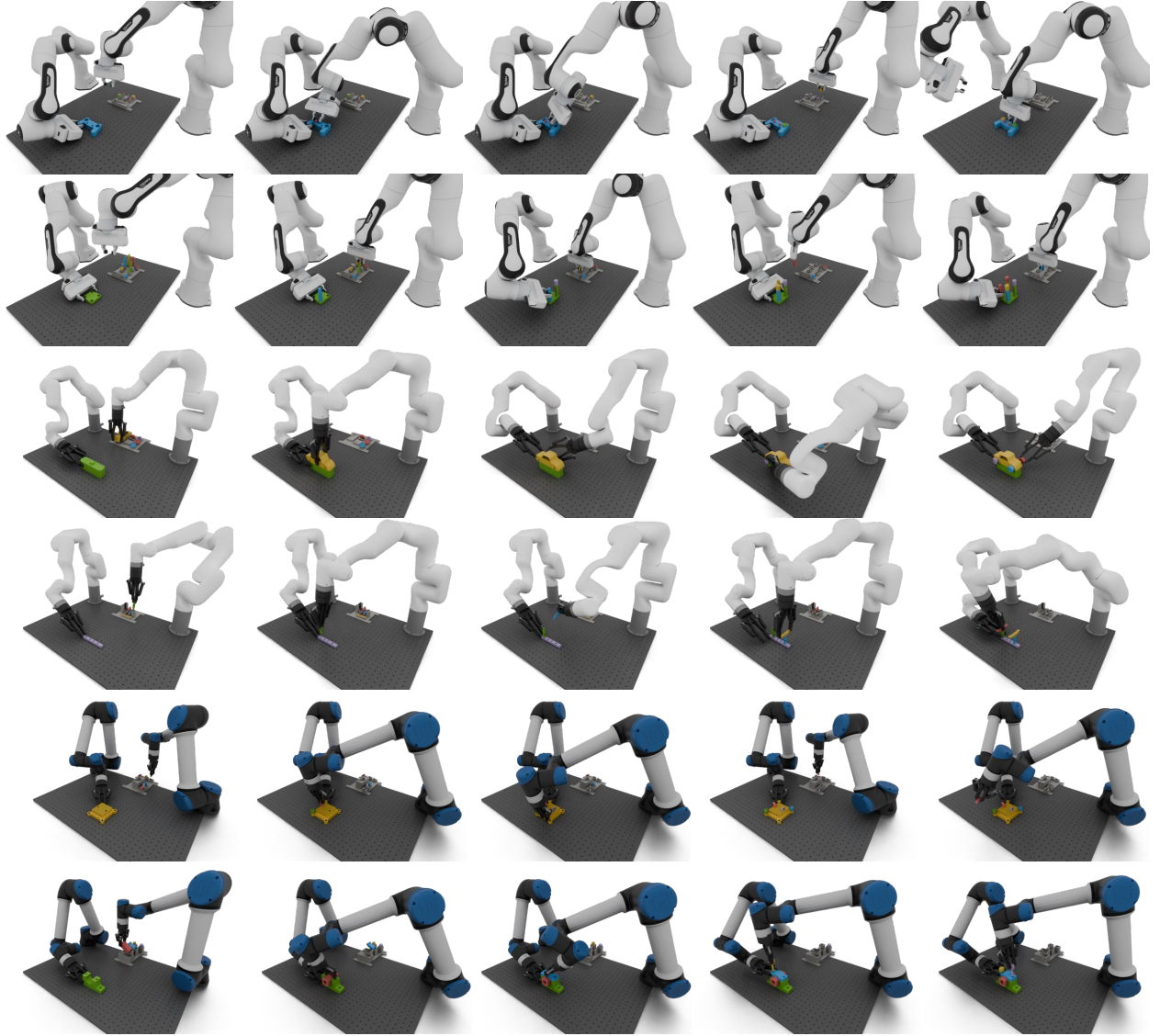


Figure 5.7: Step-by-step rendered assembly executions on different assemblies with different robots.

Table 5.7: Distribution of real-world failure cases in Fabrica aggregated over all benchmark assemblies.

Failure Case	Count	Percentage
a) Part slip from gripper	3	13.0%
b) Cumulative assembly shift	13	56.5%
c) Blocked by the holding gripper	5	21.7%
d) Unstablized assembled parts	2	8.7%

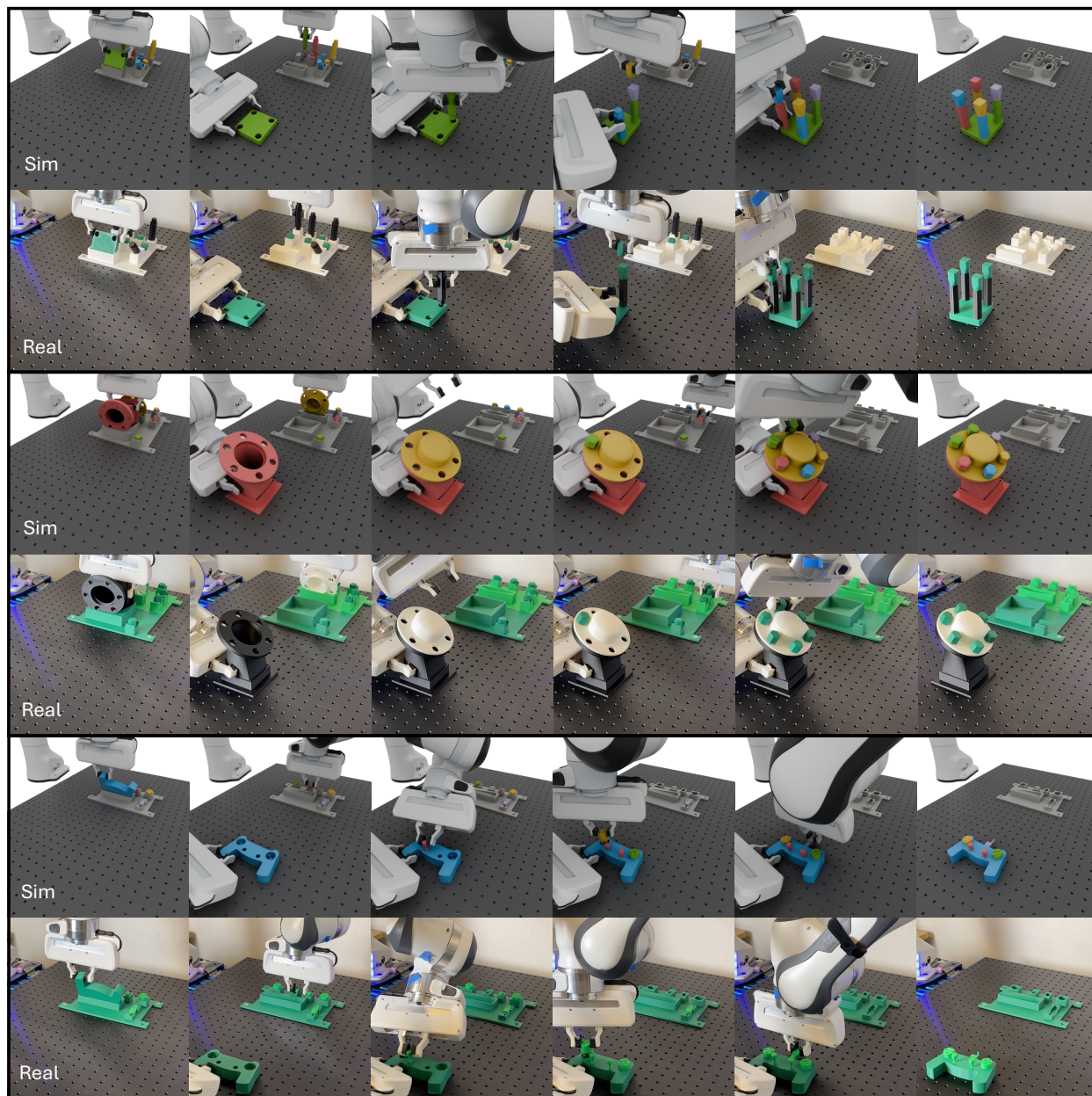


Figure 5.8: Step-by-step real-world assembly executions on different assemblies with Panda robots, with side-by-side correspondences between simulation and real.

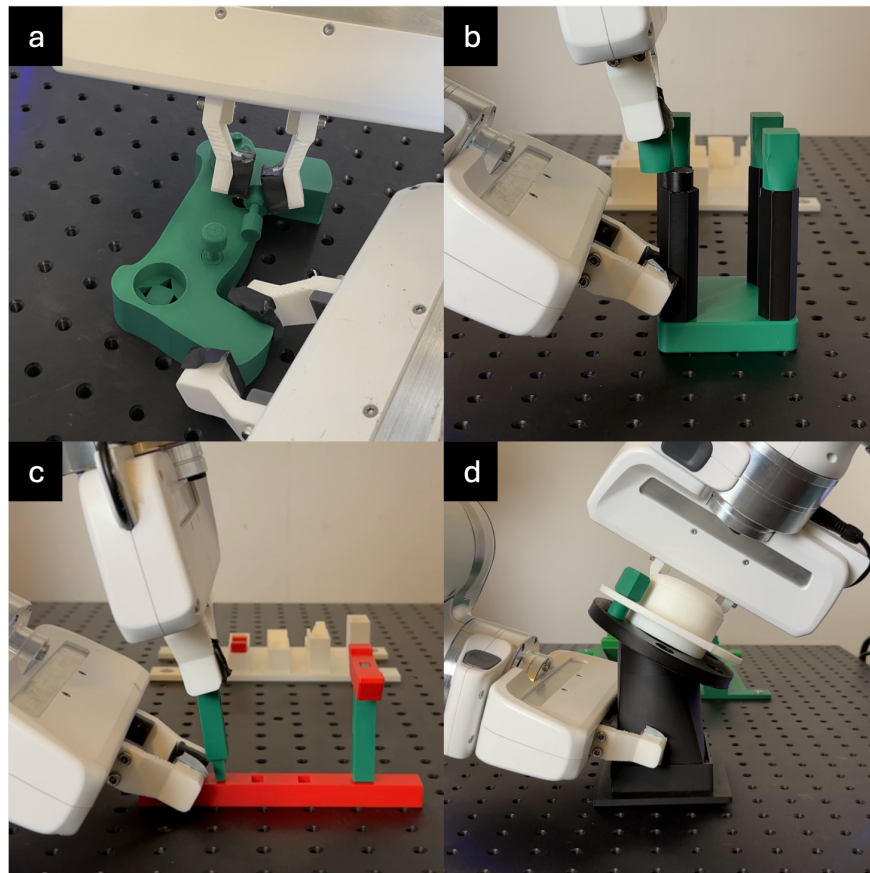


Figure 5.9: Common failure cases during multi-step assembly execution.

Table 5.8: % of successful steps without intervention in real-world evaluations.

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

Table 5.9: Multi-step cumulative success rate with 0/1/2 interventions in real-world evaluations.

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

or adapting to this issue requires non-trivial challenges for future work. Due to these failures, all methods achieve near-zero multi-step success rates without intervention due to inherently challenging steps causing consistent failures. However, with minimal interventions, our method significantly outperforms others, reaching 81% success with one intervention and 95% with two.

5.3.6 Integrating Vision Feedback: VLM for Insertion Alignment

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

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

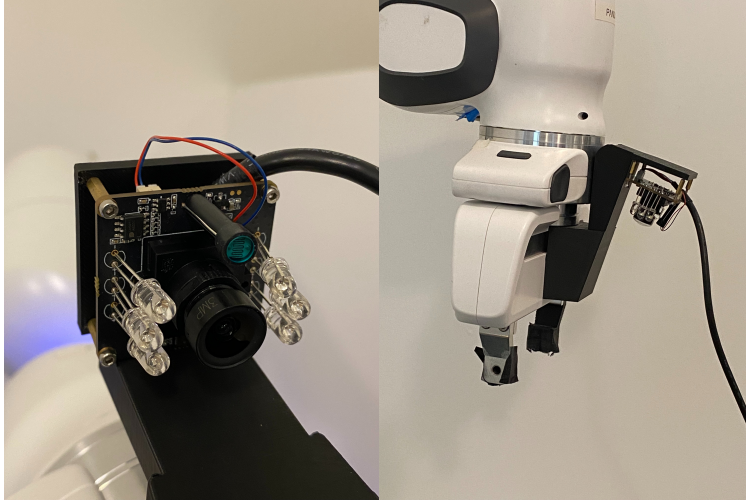


Figure 5.10: Physical setup for integrating vision feedback. Left: Camera details. Right: The mounted configuration on the robot wrist.

robustness and transparency in alignment tasks, especially under low-cost, fixed-focus camera setups.

Physical Setup

The vision integration requires only RGB input without high imaging quality, allowing for the use of low-cost cameras. We utilize an Arducam B0205 USB camera (\$34.99), shown in Figure 5.10, mounted on the robot wrist at an angle optimized to capture the insertion area. The camera is equipped with an IR-CUT filter and infrared LEDs for low-light conditions, but lacks focus adjustment, resulting in reduced image clarity when the insertion part is out of focus.

Methodology

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

'''

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

Task:

- The part is grasped by the robot and can move in four directions:
["up", "down", "left", "right"], each by 2 mm in the camera frame.

- The goal is to move the part to align it precisely with the hole for insertion.

Instructions:

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

Response format:

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

Only output the single best action based on spatial cues. If the part is already aligned, output "hold".

What is the best action to move the part to align with the hole?
 ,,,

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

Results and Analysis

The VLM integration demonstrated notable improvements in alignment accuracy, particularly in cases where the insertion policy initially failed due to misalignment. Figure 5.11 illustrates two representative examples where the VLM provided corrective actions that successfully guided the arm to the intended alignment. Specifically, the VLM identifies spatial misalignments in the camera frame and recommends discrete corrective actions (e.g., “right” and “up”) with concise reasoning.

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

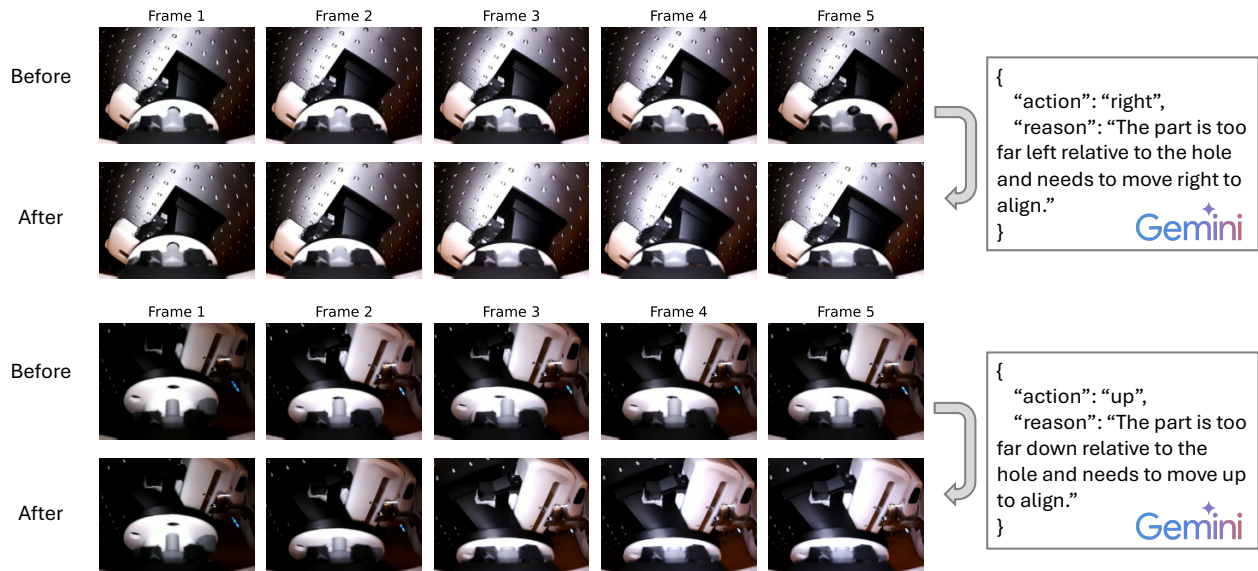


Figure 5.11: Example outputs from VLM during corrective alignment.

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

5.4 Lessons Learned from Sim-to-Real Deployments

From the deployment of our system in real-world settings, we identify several practical lessons and mitigation strategies for common pitfalls in sim-to-real transfer of robotic assembly systems.

Lesson 1: Calibration quality and its role in planning and control Accurate robot calibration is a foundational step that greatly simplifies both planning and control. While perfect calibration is not always necessary, achieving academic-grade calibration with only a few millimeters of error ensures that the robot’s position and orientation are sufficiently accurate for most tasks. The required level of calibration precision depends on the robustness of the closed-loop policy and the desired success rate. Since calibration is a well-studied problem, we recommend achieving reasonably high accuracy, even if perfection is not critical.

Lesson 2: Start simple with observation and action spaces Simpler observation and action spaces generally transfer more easily from simulation to reality. It is advisable to verify the performance of a minimal, functional system before introducing additional complexity. For example, using position-based observations often yields more reliable sim-to-real transfer than velocity-based observations, due to the difficulty of accurately modeling

a robot’s dynamics. Although optimal performance may ultimately require richer state or action representations, such expansions should be introduced gradually and validated through careful experimentation.

Lesson 3: Collision avoidance margins in planning In motion and grasp planning for contact-rich tasks, complex geometries, or constrained workspaces, leaving a small positional buffer (on the order of a few millimeters) can reduce the risk of collisions caused by unmodeled errors or uncertainties. However, overly conservative margins may eliminate feasible solutions. We recommend applying conservative collision avoidance during high-speed free-space motions, where positional accuracy is less critical, while using less conservative margins during contact-rich phases to preserve the feasibility of the plan.

Lesson 4: Tuning compliance gains for assembly Appropriate compliance gains are essential for enabling the robot to apply the correct amount of force during assembly and fine adjustments. Gains that are too high or too low can cause the part become stuck during assembly. Ideally, the optimal compliance parameters should adapt to the object’s physical properties and the tolerance between parts. While adaptive compliance tuning has been explored in research, applying and training such methods in practice remains non-trivial. Nevertheless, this is a promising area for future investigation.

5.5 Discussion and Conclusion

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

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

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

Hardware capability Compared to the existing multipart assembly dataset with thousands of objects [125], our current benchmark is limited in its size and diversity. This is because we want to ensure that the benchmark tasks are achievable by commonly used parallel grippers, but these grippers have a limited grasp width and thus cannot establish stable antipodal grasps for parts with large and complex geometries. However, to broaden the assembly

capability, we can envision either a multi-finger hand or a multi-tool system in which the robots can switch specialized grippers according to the part geometry, and our planning and control system could be adapted to this setting.

Practical integration A key practical limitation of the current setup is sensitivity to localization calibration of the two arms. Each arm’s ability to reach commanded poses with millimeter-level precision depends on the fidelity of its own kinematic model and calibration. Small errors in joint encoder offsets, link dimensions, or tool center point definition can cause systematic deviations between the planned and actual end-effector poses. Regular in-situ calibration against a reference artifact or workspace-specific pose verification could mitigate this issue, but such steps introduce additional setup and maintenance effort not yet addressed in the present system. In addition, repeated insertions and extractions can cause gradual wear of the pickup fixture, leading to deviations from the planned motions. This sensitivity is amplified by the tight tolerances inherent to the assembly, where even sub-millimeter shifts can result in failures. Incorporating perception capabilities, as illustrated below, could enable the system to detect and adapt to such deviations in real time, compensating for both calibration errors and tolerance-induced misalignments.

Perception While the VLM integration demonstrated significant alignment improvements, occlusions remained a primary failure mode. Occlusions are common in complex assemblies, necessitating either a flexible/active camera setup or multiple cameras strategically positioned to cover the scene comprehensively. Furthermore, hallucination remains a concern, particularly in cluttered scenes where visual cues are ambiguous. Future work will explore improving prompt engineering and camera configurations, potentially leveraging multi-view setups and active camera movements akin to human head and body movements. Better perception integration could further enable direct bin-picking, allowing it to grasp parts from random, unknown initial poses instead of requiring a specialized pickup fixture. However, bin-picking remains a well-known challenge in industry, particularly in terms of robustness and generalizability across arbitrary part geometries and physical properties. Addressing these challenges requires substantial research and development efforts, but would significantly expand the practical applications of our approach.

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

Chapter 6

Conclusion

Chapter 5 illustrated how integrated planning and learning enable successful real-world assembly across diverse scenarios. In this concluding chapter, we synthesize the insights gained from the entire thesis, discuss broader implications for scalable robotic assembly, and outline promising directions for future research.

6.1 Key Insights

This thesis introduces a comprehensive framework for scalable robotic assembly, addressing the crucial challenges of variability, complexity, and adaptability in modern manufacturing processes. This thesis presents core innovations that include a robust physics-based planning methodology (Assemble-Them-All), a highly efficient and structured sequence planning algorithm (ASAP), and a complete dual-arm robotic assembly system capable of handling general multi-part assemblies in the real world with augmented planning and reinforcement learning.

Firstly, Assemble-Them-All effectively leverages simulation to discover physically feasible disassembly and subsequent assembly paths, significantly enhancing the scalability and adaptability of robotic assembly solutions. The approach provides robust and realistic planning outcomes, enabling automated systems to manage complex geometries and tightly constrained assembly spaces with minimal manual input.

Secondly, ASAP demonstrates substantial improvements in computational efficiency and assembly feasibility by integrating physics-based feasibility checks and geometric or learned heuristics. This integration significantly speeds up sequence planning and enables more realistic constraints, making the algorithm highly effective in managing complex, contact-rich assembly scenarios.

Thirdly, the dual-arm robotic framework optimizes grasp selection and sequence-grasp coordination to improve the robustness of real-world dynamic manipulation and proposes a minimal yet effective reinforcement learning framework to mitigate uncertainties in contact-rich steps. The innovations in the planning and learning structure, as well as hardware co-design, substantially improve the precision and reliability of robotic assembly tasks, ensuring successful outcomes even with intricate geometries and tight precision requirements.

Finally, the thesis makes a significant contribution through establishing a large-scale

dataset and benchmark. This comprehensive resource enables systematic evaluation, training, comparison, and further advancement of robotic assembly methods, supporting continued innovation and development within the research community.

6.2 Industrial Baseline Comparison

To contextualize the capabilities of the proposed framework, it is instructive to compare it with widely used industrial robotic assembly platforms such as Siemens Process Simulate. Such a commercial tool is well-established for offline robot programming, virtual commissioning, and cycle time optimization, and are widely used in industries such as automotive and machinery manufacturing. It is highly effective when the assembly process is stable over time, the robot and the entire workcell are precisely calibrated, and the geometry is precisely modeled.

However, their workflows are predominantly manual and CAD-model-centric, requiring: Complete and accurate CAD assemblies with all part geometries and mating features explicitly defined. Manual specifications of task sequences, robot motion paths (or waypoints), object poses, and grasps. Engineer-driven reprogramming when the product design changes or a new product is introduced. In practice, this leads to multi-day to multi-week setup times for new product lines, even when reusing prior configurations. Furthermore, Siemens Process Simulate focuses on deterministic execution of pre-validated sequences; it generally lack built-in mechanisms for autonomous sequence generation or policy adaptation to unseen assemblies, which are critical for high-mix, low-volume or rapidly changing production environments.

By contrast, the proposed framework developed in this thesis accepts raw CAD meshes (without explicit mating feature annotations) and automatically: 1) Infers physically feasible assembly sequences using physics-based simulation; 2) Selects grasps and optimizes sequence-grasp coordination; 3) Executes contact-rich steps with learning-based policies that adapt to geometric and physical uncertainties. This reduces setup time from weeks to hours, and enables zero-shot execution for assemblies not seen during system training.

Table 6.1 provides a summary of comparisons between our framework and Siemens Process Simulate along different dimensions, including **flexibility** for adaptation to new assemblies, **setup time** for new product introduction, **generalization** to unseen assemblies, and **adoption** for practical integration into assembly workflows. While Siemens Process Simulate remains the gold standard for enterprise-scale integration, production line optimization, and coordination across multiple robotic cells, the proposed framework targets a different but complementary niche: rapid, autonomous adaptation to new assemblies in high-mix or fast-changing manufacturing environments. In practice, these two approaches could be used in combination: Siemens Process Simulate providing the overall production planning, BOM management, and line layout validation, while the proposed framework handles automated sequence generation, grasp planning, and adaptive execution for individual workcells. This hybrid approach would combine the robustness and maturity of established industrial platforms with the agility and autonomy of next-generation robotic assembly systems.

Table 6.1: Qualitative comparison between the proposed framework and Siemens Process Simulate.

Performance Criteria	Proposed Framework	Siemens Process Simulate
Flexibility (adaptation to new assemblies)	Accepts raw CAD meshes without pre-defined planning information; automatically infers sequences and grasps; some tolerance to small modeling or calibration errors.	Requires complete CAD with manual programming of sequences, poses, grasps, and motion paths; limited automatic sequence generation; extensive calibration required.
Setup Time (new product introduction)	Hours to a day; automated physics-based planning eliminates most manual programming except for workcell design and reconfiguration.	Typically several days to weeks; extensive manual planning, authoring, and tuning required for each product variant.
Generalization (unseen assemblies)	High: integrates learning-based control and physics-based planning for zero-shot execution on assemblies not seen during training; does not work with deformable parts or complex assembly mechanisms yet.	Low–Moderate: can reuse prior programs only if geometry changes are minor; major changes require full reprogramming; deformable or complex assembly processes can be specified manually.
Adoption (practical workflow integration)	Emerging research prototype; not yet integrated into enterprise product lifecycle management (PLM) workflows; designed for flexible, high-mix scenarios.	Mature, widely adopted in industry; integrates with bill of materials (BOM) management, change control, and entire production line layout design, enabling coordinated planning across multiple workcells.

6.3 Future Directions

Flexible and general physics-based assembly simulation To effectively train robotic assembly agents that handle intricate behaviors, it is crucial to develop flexible and general physics-based simulations. Future simulation frameworks should robustly model sophisticated mechanical interactions such as snap-fit connections, press-fits, and threaded insertions, ideally in a unified framework. Additionally, these simulations must extend beyond rigid objects to accurately represent deformable materials [138], stiff parts [139], thin-shell structures [140], cables, and wires [141], which frequently appear in realistic assembly scenarios. Achieving high computational efficiency in simulating these complex interactions is essential to generate extensive datasets necessary for effective policy training. Moreover, the development of simulation twins constructed directly from real-world video demonstrations [142, 143] can further enhance the realism and accuracy of virtual training environments, ensuring seamless

transfer of learned policies to real-world applications.

3D assembly data generation and foundation models A truly generalist robotic assembly agent requires large-scale, diverse training data of 3D assemblies. While our released dataset [40] (containing 8,251 assemblies and 154,468 separate parts) is the largest 3D assembly dataset to date, its scale remains orders of magnitude smaller than image datasets like LAION-5B [144] (5B) or 3D shape datasets like Objaverse-XL [145] (10M). To bridge this gap, it is promising to develop generative models to produce synthetic yet high-fidelity 3D assemblies with precisely aligned parts, ensuring physical manipulability. By scaling up assembly data by orders of magnitude, we can unlock the potential of large foundation models for robotic assembly, leading to generalizable planning and manipulation at unprecedented levels of flexibility and efficiency.

Multi-skill, multi-scale, multi-sensory dexterous robotic assembly Beyond parallel grippers and insertion skills, an advanced assembly system must leverage a diverse set of tools, including suction cups, screwdrivers, wrenches, hammers, and fixtures [146]. Selecting the right tools and skills for a given assembly task can be guided by geometric reasoning [147], supervised learning [148], or language models [149].

In addition, assemblies naturally come with multi-scale parts and require distinct strategies such as pivoting [150] and reorientation [151] for precise in-hand manipulation of tiny parts, and whole-body or multi-arm coordination [73] to allow strength and reach in larger parts. To collectively advance robotic assembly, it is promising to establish standardized benchmarks and learning frameworks spanning diverse skills and scales.

Furthermore, industrial-grade robotic assembly requires enhanced perception through vision and tactile sensing. By integrating structured planning with multi-sensory policy learning, my goal is to enable robots to reason about tool use, master dexterous manipulation skills, and execute complex, long-horizon assemblies with high precision and reliability.

Practical industrial applications The capabilities developed for generalist robotic assembly have direct relevance to a wide range of industrial scenarios. In logistics and e-commerce warehouses, robots equipped with multi-sensory perception and adaptive manipulation can perform high-throughput bin picking, handling a constantly changing variety of SKUs with different shapes, weights, and packaging materials. In manufacturing settings, the ability to perform deformable insertions, such as routing cables through confined spaces, fitting rubber seals, or inserting flexible tubing, can streamline assembly processes that currently rely heavily on manual labor. Similarly, mixed-material assemblies, which combine rigid, semi-rigid, and deformable components (e.g., metal fasteners into composite housings or electronics into plastic enclosures), present challenges in robust perception, planning, and control that future systems must address. Demonstrating robust performance in these realistic, high-variability applications will be a critical milestone toward the widespread adoption of fully autonomous assembly systems in industry.

6.4 Thesis Impact

The proposed framework significantly advances robotic assembly capabilities, offering robust solutions to previously intractable challenges in industrial automation and robotic manipulation. By systematically integrating physics-based planning, automated sequence optimization, and dual-arm coordination, the framework dramatically enhances adaptability and scalability, enabling robotic systems to autonomously manage diverse assembly scenarios with minimal reconfiguration.

Industrially, this translates into substantial reductions in setup and reconfiguration times, enabling more agile and responsive production environments. The capability to autonomously adapt to new assembly tasks without extensive manual programming significantly reduces operational costs and enhances productivity. This adaptability is particularly impactful in sectors characterized by high-mix, low-volume production, such as aerospace, automotive, and consumer electronics, where frequent product updates and customizations are common.

Moreover, the comprehensive empirical validation conducted through extensive simulations and real-world robotic experiments bridges theoretical advancements with practical applicability. By demonstrating reliable performance under realistic industrial conditions, this thesis increases industry confidence in adopting advanced robotic assembly technologies.

Academically, the dataset and benchmarking suite introduced in this work offer an unprecedented opportunity for rigorous evaluation and comparison across various robotic assembly methodologies. Such resources facilitate incremental and collaborative innovation, driving forward research efforts across robotics, automation, and related fields. Additionally, by emphasizing a multi-disciplinary integration of physics simulation, algorithmic optimization, and practical robotics control, this thesis highlights a critical research methodology that could inspire future studies.

Ultimately, this thesis lays a foundational framework for next-generation robotic assembly systems that are inherently flexible, responsive, and scalable. The advancements presented not only promise significant productivity and efficiency improvements in manufacturing but also pave the way for broader robotic applications, significantly influencing the trajectory of future robotic research and industrial automation. The physical world will be built and refined at an unprecedented speed by extremely democratizing the effort of building machines that make machines.

Appendix A

Hyperparameters of Physics-Based Path Planning

We now present detailed hyper-parameter settings of experiments introduced in Section 3.4. Table A.1 summarizes the main hyper-parameters used in our physics-based simulation, Table A.2 shows the hyper-parameters for physics-based path planners (our method and BK-RRT), and Table A.3 shows the hyper-parameters for geometric-based path planners (RRT, T-RRT, MV+T-RRT).

Table A.1: 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 A.2: 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 A.3: 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

Appendix B

Algorithmic Details of Multi-Part Dual-Arm Assembly Planning

We now present detailed mathematical formulations of each algorithm introduced in Sec. 5.1.

B.1 Part Precedence Planning

Algorithm 7 and the following paragraphs provide the details of the part precedence planning algorithm.

B.1.1 Precedence Tier Generation

Initially, an empty list of tiers T_{prec} is initialized. We use O_r to represent all the remaining assembled parts, which starts from all parts O . Although O_r is not empty, the algorithm constructs a new tier t_{prec} by evaluating each part $o_i \in O_r$ to determine the feasibility of disassembly. Using motion planning, a disassembly path q_i is computed for o_i . Specifically, we apply Assemble-Them-All [125], a physics-based method for efficient disassembly motion planning given the highly constrained search space, and determine success based on a given timeout. If o_i can be disassembled, the pair (o_i, q_i) is added to t_{prec} . Once all feasible parts are processed, the constructed tier t_{prec} is added to T_{prec} , and all $o_i \in t_{\text{prec}}$ is removed from O_r . This process repeats until all parts are assigned to tiers.

B.1.2 Precedence Graph Generation

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

O_e . The algorithm proceeds until all parts in all precedence tiers are added to the precedence graph.

Algorithm 7 Part Precedence Planning

```

1: Input: All parts  $O$  with goal poses  $p_O^G$ 
2: Output: Directed graph  $G_{\text{prec}}$  representing precedence constraints
3: Initialize an empty list of tiers:  $T_{\text{prec}} \leftarrow []$ 
4: Initialize an empty directed graph:  $G_{\text{prec}} \leftarrow \text{DiGraph}()$ 
5: Initialize all remaining parts  $O_r \leftarrow O$ 
6: while  $O_r \neq \emptyset$  do ▷ Tier generation
7:   Initialize an empty tier:  $t_{\text{prec}} \leftarrow \{\}$ 
8:   for each part  $o_i \in O_r$  do
9:      $\tau_{o_i} \leftarrow \text{DisassemblyPath}(o_i, O_r)$ 
10:    if feasible  $\tau_{o_i}$  is found then
11:       $t_{\text{prec}} \leftarrow t_{\text{prec}} \cup \{(o_i, \tau_{o_i})\}$ 
12:     $O_r \leftarrow O_r \setminus \{o_i \mid (o_i, \tau_{o_i}) \in t_{\text{prec}}\}$ 
13:     $T_{\text{prec}}.\text{Append}(t_{\text{prec}})$ 
14: Initialize an empty set of parts in earlier tiers:  $O_e \leftarrow \emptyset$ 
15: for each tier  $t_{\text{prec}} \in T_{\text{prec}}$  do ▷ Graph construction
16:   Let  $O_t \leftarrow \{o_i \mid (o_i, \tau_{o_i}) \in t_{\text{prec}}\}$  ▷ Parts in tier  $t_{\text{prec}}$ 
17:   for each  $(o_i, \tau_{o_i}) \in t_{\text{prec}}$  do
18:      $G_{\text{prec}}.\text{AddNode}(o_i, \text{path} = \tau_{o_i})$ 
19:      $O_c \leftarrow \text{CheckPathCollision}(o_i, \tau_{o_i}, O_e)$  ▷  $O_c$ : colliding parts in  $O_e$ 
20:     for each  $o_c \in O_c$  do
21:        $G_{\text{prec}}.\text{AddEdge}(o_i, o_c)$ 
22:    $O_e \leftarrow O_e \cup O_t$  ▷ Update parts in earlier tiers
23: return  $G_{\text{prec}}$ 

```

B.2 Dual-Arm Grasp Filtering

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

B.2.1 Single-Pose Grasp Feasibility Check

algorithm 8 evaluates whether a specific grasp configuration g is feasible for a target part o in its current pose p_o . The algorithm first determines the set of preceding parts O_{prec} from the precedence graph G_{prec} . For each gripper aperture (a_{grasp} and a_{release}), collision checks are performed involving the robot body, the gripper, the target part o , the preceding parts O_{prec} , and the environment obstacles E . If any collision occurs between them, the grasp g is not feasible. Additional collision checks are performed between the gripper and other non-preceding parts $O_{\text{all}} \setminus O_{\text{prec}}$, and collision information is added to the grasp. Such collisions

are not hard constraints because the non-preceding parts may get assembled later than the target part depending on the specific assembly sequence. In this case, the collision does not matter, but the information has to be recorded to find collision-free assembly sequences in the later stage.

Algorithm 8 Single-Pose Grasp Feasibility Check

```

1: Input: Grasp  $g$ , robot  $R$ , target part  $o$  with pose  $p_o$ , all parts  $o \in O$  with their
   corresponding goal poses  $p_o^G$ , precedence graph  $G_{\text{prec}}$ 
2: Output: Feasibility (True/False), updated grasp  $g$  with collision and IK information
3:  $O_{\text{prec}} \leftarrow G_{\text{prec}}.\text{PrecedingParts}(o)$ 
4:  $E \leftarrow$  environment obstacles
5:  $q_g^R \leftarrow \text{IK}(R, g, o, p_o)$   $\triangleright$  IK for robot to grasp part  $o$  under pose  $p_o$  with grasp  $g$ 
6: if feasible  $q_g^R$  is found then
7:   Set robot  $R$  configuration to  $q_g^R$ 
8:   for each gripper aperture  $a \in \{a_{\text{grasp}}, a_{\text{release}}\}$  do
9:     Set gripper with aperture  $a$ 
10:    if  $\text{CheckCollision}(R, o, O_{\text{prec}}, E)$  then
11:      return False, g
12:     $\text{CheckCollision}(R, O \setminus O_{\text{prec}})$ 
13:    Record collision and IK information to  $g$ 
14:    return True, g
15: else
16:    return False, g

```

B.2.2 Assembling Grasp Feasibility Check

algorithm 9 determines the feasibility of using a grasp g to disassemble a target part o along its disassembly path τ_o , derived from G_{prec} . For each pose $p_{o,t}$ along τ_o , the grasp is transformed accordingly and its feasibility is validated using algorithm 8. The aggregated collision and IK information across all poses is stored in g . The grasp is feasible if all poses along τ_o are validated.

B.2.3 Holding Grasp Feasibility Check

algorithm 10 evaluates whether a grasp g can securely hold a part o while allowing other parts o_i to be disassembled. The feasibility of g is first validated using algorithm 8.

B.2.4 Grasp Pair Filtering

Putting the assembling and holding feasibility checks together, algorithm 11 generates and filters the dual-arm grasp pairs. For each part o_i , a set of candidate grasp poses $\{g_k\}_{k=1}^{N_g}$ is generated. Each grasp is evaluated for assembling and holding feasibility using algorithm 9 and ??, respectively, and feasible grasps are stored in $\mathcal{G}^a[o_i]$ and $\mathcal{G}^h[o_i]$. Finally, iterating

Algorithm 9 Assembling Grasp Feasibility Check

```
1: Input: Grasp  $g$ , robot  $R$ , target part  $o$ , all parts  $O$  with goal poses  $p_O^G$ , precedence graph  $G_{\text{prec}}$ 
2: Output: Feasibility (True/False), updated grasp  $g$  with collision and IK information
3:  $\tau_o \leftarrow G_{\text{prec}}.\text{GetPath}(o)$  ▷ Disassembly path of part  $o$ 
4: for each part pose  $p_{o,t} \in \tau_o$  do ▷ Disassembling  $o$  while grasping  $o$ 
5:   Transform grasp  $g$  according to  $p_o^t$  to obtain  $g_t$ 
6:    $\text{feas}, g_t \leftarrow \text{CheckGraspFeas}(g_t, o, p_{o,t}, \dots)$  ▷ Alg. 8
7:   if not  $\text{feas}$  then
8:     return False,  $g$ 
9:   Gather collision and IK information from  $g_t$  to  $g$ 
10: return True,  $g$ 
```

Algorithm 10 Holding Grasp Feasibility Check

```
1: Input: Grasp  $g$ , robot  $R$ , target part  $o$ , all parts  $O$  with goal poses  $p_O^G$ , precedence graph  $G_{\text{prec}}$ 
2: Output: Feasibility (True/False), updated grasp  $g$  with collision and IK information
3:  $\text{feas}, g \leftarrow \text{CheckGraspFeas}(g, o, p_o^G, \dots)$  ▷ Alg. 8
4: if not  $\text{feas}$  then
5:   return False,  $g$ 
6: for each part  $o_i \in O_{\text{all}} \setminus \{o\}$  do
7:    $q_{o_i} \leftarrow G_{\text{prec}}.\text{GetPath}(o_i)$  ▷ Disassembly path of  $o_i$ 
8:   for each part pose  $p_{o_i,t} \in q_{o_i}$  do ▷ Disassembling  $o_i$  while grasping  $o$ 
9:     CheckCollision( $R, o_i$ ) and gather collision information to  $g$ 
10: return True,  $g$ 
```

through all assembly-hold part pairs, the set of feasible assembly-hold grasp pairs $\mathcal{G}^{a \times h}[o_a, o_h]$ only contains those that do not lead to collisions between the two robots.

Algorithm 11 Dual-Arm Grasp Pair Filtering

```

1: Input: All parts  $O$  with goal poses  $p_O^G$ , robots  $R_a, R_h$ , precedence graph  $G_{\text{prec}}$ 
2: Output: Assembling grasps  $\mathcal{G}^a$ , holding grasps  $\mathcal{G}^h$ , assembling-holding grasp pairs  $\mathcal{G}^{a \times h}$ 
3: Initialize empty dictionaries  $\mathcal{G}^a, \mathcal{G}^h, \mathcal{G}^{a \times h} \leftarrow \{\cdot\}, \{\cdot\}, \{\cdot\}$ 
4: for each part  $o_i$  in  $O$  do ▷ Feasible grasp generation
5:    $\mathcal{G}^a[o_i], \mathcal{G}^h[o_i] \leftarrow \{\}, \{\}$ 
6:   Generate  $N_g$  grasp poses  $\{g_k\}_{k=1}^{N_g}$  on part  $o_i$ 
7:   for each grasp pose  $g_k$  do
8:      $\text{feas}_a, g_a \leftarrow \text{CheckAssemGraspFeas}(g_k, R_a, o_i, \dots)$ 
9:     if  $\text{feas}_a$  then
10:       $\mathcal{G}^a[o_i] \leftarrow \mathcal{G}^a[o_i] \cup \{g_a\}$ 
11:      $\text{feas}_h, g_h \leftarrow \text{CheckHoldGraspFeas}(g_k, R_h, o_i, \dots)$ 
12:     if  $\text{feas}_h$  then
13:       $\mathcal{G}^h[o_i] \leftarrow \mathcal{G}^h[o_i] \cup \{g_h\}$ 
14: for each part  $o_a \in O$  do ▷ Feasible grasp pair filtering
15:   for each part  $o_h \in O$  do
16:     if  $o_a \in G_{\text{prec}}.\text{PrecedingParts}(o_h)$  then
17:       continue
18:        $\mathcal{G}^{a \times h}[(o_a, o_h)] \leftarrow \{\}$ 
19:       for each grasp  $g_a \in \mathcal{G}^a[o_a]$  do
20:         for each grasp  $g_h \in \mathcal{G}^h[o_h]$  do
21:           Set robot  $R_a$  configuration to  $q_{g_a}^{R_a}$ 
22:           Set robot  $R_h$  configuration to  $q_{g_h}^{R_h}$ 
23:           if not  $\text{CheckCollision}(R_a, R_h)$  then
24:              $\mathcal{G}^{a \times h}[(o_a, o_h)] \leftarrow \mathcal{G}^{a \times h}[(o_a, o_h)] \cup \{(g_a, g_h)\}$ 
25: return  $\mathcal{G}^a, \mathcal{G}^h, \mathcal{G}^{a \times h}$ 

```

B.3 Dual-arm Sequence-Grasp Optimization

Algorithm 12 provides the pseudocode for the sequence-grasp optimization algorithm.

Algorithm 12 Dual-Arm Assembly Sequence Planning

```

1: Input: All parts  $O$ , precedence graph  $G_{\text{prec}}$ , assembling grasps  $\mathcal{G}^a$ , assembling-holding
   grasp pairs  $\mathcal{G}^{a \times h}$ 
2: Output: Optimal assembly part-grasp sequence  $S^*$ 
3: Initialize an empty tree  $T_G \leftarrow \text{DiGraph}()$ 
4:  $S \leftarrow []$  ▷ Initialize an empty search stack.
5: for each  $o \in O$  do
6:   if  $G_{\text{prec}}.\text{SucceedingParts}(o) = \emptyset$  then
7:     for each  $g \in \mathcal{G}^a[o]$  do
8:        $T_G.\text{AddNode}((a, O \setminus \{o\}, o, g))$  ▷ Root nodes
9:        $\text{push}(S, (a, O \setminus \{o\}, o, g))$ 
10: while  $S$  not empty do
11:    $(t_i, O_i, o_i, g_i) \leftarrow \text{pop}(S)$ 
12:    $t_{i+1} \leftarrow h$  if  $t_i = a$  else  $t_{i+1} \leftarrow a$ 
13:   for each  $o_{i+1} \in O_i$  do
14:      $O_{i+1} \leftarrow O_i \setminus \{o_i\}$  if  $t_{i+1} = a$  else  $O_{i+1} \leftarrow O_i$ 
15:     if  $(O_i \cup \{o_i\}) \cap G_{\text{prec}}.\text{PrecedingParts}(o_{i+1}) \neq \emptyset$  then
16:       continue ▷ Precedence check
17:     for each  $g_{i+1} \in \mathcal{G}^{t_{i+1}}[o_{i+1}]$  do
18:        $O_c \leftarrow \mathcal{G}^{t_{i+1}}[o_{i+1}][g_{i+1}].\text{CollidingSet}$  ▷ Precomputed collision set
19:       if  $O_c \cap O_{i+1} \neq \emptyset$  then
20:         continue ▷ State collision check
21:       if  $(g_{i+1}, g_i) \in \mathcal{G}^{t_{i+1} \times t_i}[(o_{i+1}, o_i)]$  then
22:          $T_G.\text{AddEdge}(((t_i, O_i, o_i, g_i), (t_{i+1}, O_{i+1}, o_{i+1}, g_{i+1})))$  ▷ Grasp pair validity
23:          $\text{push}(S, (t_{i+1}, O_{i+1}, o_{i+1}, g_{i+1}))$ 
24:   for each  $e_i \in T_G.\text{Edges}()$  do ▷ Objective evaluation
25:      $((t_i, O_i, o_i, g_i), (t_j, O_j, o_j, g_j)) \leftarrow e_i$ 
26:     if  $t_i = a, t_j = h$  then ▷ Assembling-holding step
27:        $e_i.\vec{f} \leftarrow \vec{f}(O_i, o_i, g_i, o_j, g_j)$ 
28:     else ▷ Holding transition step
29:        $e_i.\vec{f} \leftarrow \vec{0}$ 
30:    $S^* \leftarrow [o_{a,1}^*, g_{a,1}^*, o_{h,2}^*, g_{h,2}^*, \dots, o_{a,n}^*, g_{a,n}^*] \leftarrow \text{DP}(T_G, \Phi)$  ▷ Dynamic programming
31: return  $S^*$ 

```

References

- [1] L. Homem de Mello and A. Sanderson. “A correct and complete algorithm for the generation of mechanical assembly sequences”. In: *IEEE Transactions on Robotics and Automation* 7.2 (1991), pp. 228–240. DOI: [10.1109/70.75905](https://doi.org/10.1109/70.75905).
- [2] Y. Tian, K. D. Willis, B. Al Omari, J. Luo, P. Ma, Y. Li, F. Javid, E. Gu, J. Jacob, S. Sueda, et al. “Asap: Automated sequence planning for complex robotic assembly with physical feasibility”. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 4380–4386.
- [3] Y. Tian et al. *Fabrica: Dual-Arm Assembly of General Multi-Part Objects via Integrated Planning and Learning*. 2025. arXiv: [2506.05168](https://arxiv.org/abs/2506.05168) [cs.RO]. URL: <https://arxiv.org/abs/2506.05168>.
- [4] R. H. Wilson and J.-C. Latombe. “Geometric reasoning about mechanical assembly”. In: *Artificial Intelligence* 71.2 (1994), pp. 371–396.
- [5] D. Halperin, J.-C. Latombe, and R. H. Wilson. “A general framework for assembly planning: The motion space approach”. In: *Algorithmica* 26.3 (2000), pp. 577–601.
- [6] E. Masehian and S. Ghandi. “Assembly sequence and path planning for monotone and nonmonotone assemblies with rigid and flexible parts”. In: *Robotics and Computer-Integrated Manufacturing* 72 (2021), p. 102180.
- [7] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”. In: *IEEE transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.
- [8] S. M. LaValle et al. “Rapidly-exploring random trees: A new tool for path planning”. In: (1998).
- [9] M. Jordan and A. Perez. “Optimal bidirectional rapidly-exploring random trees”. In: (2013).
- [10] 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.
- [11] I. Aguinaga, D. Borro, and L. Matey. “Parallel RRT-based path planning for selective disassembly planning”. In: *The International Journal of Advanced Manufacturing Technology* 36.11 (2008), pp. 1221–1233.

- [12] J. Cortés, L. Jaillet, and T. Siméon. “Molecular disassembly with RRT-like algorithms”. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE. 2007, pp. 3301–3306.
- [13] S. Sundaram, I. Remmler, and N. M. Amato. “Disassembly sequencing using a motion planning approach”. In: *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*. Vol. 2. IEEE. 2001, pp. 1475–1480.
- [14] D. T. Le, J. Cortés, and T. Siméon. “A path planning approach to (dis) assembly sequencing”. In: *2009 IEEE International Conference on Automation Science and Engineering*. IEEE. 2009, pp. 286–291.
- [15] X. Zhang, R. Belfer, P. G. Kry, and E. Vouga. “C-Space tunnel discovery for puzzle path planning”. In: *ACM Transactions on Graphics (TOG)* 39.4 (2020), pp. 104–1.
- [16] D. Hsu, J.-C. Latombe, R. Motwani, and L. E. Kavraki. “Capturing the connectivity of high-dimensional geometric spaces by parallelizable random sampling techniques”. In: *Advances in randomized parallel computing*. Springer, 1999, pp. 159–182.
- [17] S. Zickler and M. M. Veloso. “Efficient physics-based planning: sampling search via non-deterministic tactics and skills.” In: *AAMAS (1)*. Citeseer. 2009, pp. 27–33.
- [18] I. A. Sucan and L. E. Kavraki. “A sampling-based tree planner for systems with complex dynamics”. In: *IEEE Transactions on Robotics* 28.1 (2011), pp. 116–131.
- [19] M. Moll, L. Kavraki, J. Rosell, et al. “Randomized physics-based motion planning for grasping in cluttered and uncertain environments”. In: *IEEE Robotics and Automation Letters* 3.2 (2017), pp. 712–719.
- [20] M. F. F. Rashid, W. Hutabarat, and A. Tiwari. “A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches”. In: *The International Journal of Advanced Manufacturing Technology* 59.1 (2012), pp. 335–349.
- [21] L. H. De Mello and A. C. Sanderson. “AND/OR graph representation of assembly plans”. In: *IEEE Transactions on robotics and automation* 6.2 (1990), pp. 188–199.
- [22] Y. Qin and Z. Xu. “Assembly process planning using a multi-objective optimization method”. In: *Proceedings of the 2007 IEEE international conference on mechatronics and automation, ICMA*. Vol. 4303610. 2007, pp. 593–598.
- [23] C. Sinanoğlu and H. R. Börlü. “An assembly sequence-planning system for mechanical parts using neural network”. In: *Assembly Automation* (2005).
- [24] G. Chen, J. Zhou, W. Cai, X. Lai, Z. Lin, and R. Menassa. “A framework for an automotive body assembly process design system”. In: *Computer-Aided Design* 38.5 (2006), pp. 531–539.
- [25] W.-C. Chen, P.-H. Tai, W.-J. Deng, and L.-F. Hsieh. “A three-stage integrated approach for assembly sequence planning using neural networks”. In: *Expert Systems with Applications* 34.3 (2008), pp. 1777–1786.
- [26] Q. Su. “A hierarchical approach on assembly sequence planning and optimal sequences analyzing”. In: *Robotics and Computer-Integrated Manufacturing* 25.1 (2009), pp. 224–234.

- [27] X. Niu, H. Ding, and Y. Xiong. “A hierarchical approach to generating precedence graphs for assembly planning”. In: *International Journal of Machine Tools and Manufacture* 43.14 (2003), pp. 1473–1486.
- [28] C. Ramos, J. Rocha, and Z. Vale. “On the complexity of precedence graphs for assembly and task planning”. In: *Computers in industry* 36.1-2 (1998), pp. 101–111.
- [29] S. Ghandi and E. Masehian. “Review and taxonomies of assembly and disassembly path planning problems and approaches”. In: *Computer-Aided Design* 67 (2015), pp. 58–86.
- [30] H. Wang, Y. Rong, and D. Xiang. “Mechanical assembly planning using ant colony optimization”. In: *Computer-Aided Design* 47 (2014), pp. 59–71.
- [31] K. Zakka, A. Zeng, J. Lee, and S. Song. “Form2fit: Learning shape priors for generalizable assembly from disassembly”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 9404–9410.
- [32] Y. Huang, C. R. Garrett, I. Ting, S. Parascho, and C. T. Mueller. “Robotic additive construction of bar structures: Unified sequence and motion planning”. In: *Construction Robotics* 5 (2021), pp. 115–130.
- [33] S. Abe, T. Murayama, F. Oba, and N. Narutaki. “Stability check and reorientation of subassemblies in assembly planning”. In: *IEEE SMC’99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028)*. Vol. 2. IEEE. 1999, pp. 486–491.
- [34] J. Aleotti and S. Caselli. “Efficient planning of disassembly sequences in physics-based animation”. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2009, pp. 87–92.
- [35] I. Rodriguez, K. Nottensteiner, D. Leidner, M. Kaßecker, F. Stulp, and A. Albu-Schäffer. “Iteratively refined feasibility checks in robotic assembly sequence planning”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1416–1423.
- [36] E. Kongar and S. M. Gupta. “Disassembly sequencing using genetic algorithm”. In: *The International Journal of Advanced Manufacturing Technology* 30 (2006), pp. 497–506.
- [37] R. M. Marian, L. H. Luong, and K. Abhary. “A genetic algorithm for the optimisation of assembly sequences”. In: *Computers & Industrial Engineering* 50.4 (2006), pp. 503–527.
- [38] H.-E. Tseng, C.-C. Chang, S.-C. Lee, and Y.-M. Huang. “A block-based genetic algorithm for disassembly sequence planning”. In: *Expert Systems with Applications* 96 (2018), pp. 492–505.
- [39] Y. Huang, J. Zhang, X. Hu, G. Song, Z. Liu, L. Yu, and L. Liu. “Framefab: Robotic fabrication of frame shapes”. In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), pp. 1–11.
- [40] K. D. Willis et al. “JoinABLE: Learning Bottom-up Assembly of Parametric CAD Joints”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022.

- [41] K. Lupinetti, F. Giannini, M. Monti, and J.-P. Pernot. “Content-based multi-criteria similarity assessment of CAD assembly models”. In: *Computers in Industry* 112 (2019), p. 103111.
- [42] K. Watanabe and S. Inada. “Search algorithm of the assembly sequence of products by using past learning results”. In: *International Journal of Production Economics* 226 (2020), p. 107615.
- [43] K. Kitz and U. Thomas. “Neural dynamic assembly sequence planning”. In: *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2021, pp. 2063–2068.
- [44] N. Funk, S. Menzenbach, G. Chaltatzaki, and J. Peters. “Graph-based Reinforcement Learning meets Mixed Integer Programs: An application to 3D robot assembly discovery”. In: (2022).
- [45] F. Suárez-Ruiz, X. Zhou, and Q.-C. Pham. “Can robots assemble an IKEA chair?” In: *Science Robotics* 3.17 (2018), eaat6385.
- [46] L. Nägele, A. Hoffmann, A. Schierl, and W. Reif. “LegoBot: Automated planning for coordinated multi-robot assembly of LEGO structures”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 9088–9095.
- [47] Y. Huang, P. Y. V. Leung, C. Garrett, F. Gramazio, M. Kohler, and C. Mueller. “The new analog: A protocol for linking design and construction intent with algorithmic planning for robotic assembly of complex structures”. In: *Proceedings of the 6th Annual ACM Symposium on Computational Fabrication*. 2021, pp. 1–17.
- [48] Z. Wang, F. Kennel-Maushart, Y. Huang, B. Thomaszewski, and S. Coros. “A temporal coherent topology optimization approach for assembly planning of bespoke frame structures”. In: *ACM Transactions on Graphics (TOG)* 42.4 (2023), pp. 1–13.
- [49] C.-J. Liang, S.-C. Kang, and M.-H. Lee. “RAS: a robotic assembly system for steel structure erection and assembly”. In: *International Journal of Intelligent Robotics and Applications* 1 (2017), pp. 459–476.
- [50] K. Dörfler, T. Sandy, M. Giftthaler, F. Gramazio, M. Kohler, and J. Buchli. “Mobile robotic brickwork: automation of a discrete robotic fabrication process using an autonomous mobile robot”. In: *Robotic Fabrication in Architecture, Art and Design 2016* (2016), pp. 204–217.
- [51] A. A. Apolinarska, M. Pacher, H. Li, N. Cote, R. Pastrana, F. Gramazio, and M. Kohler. “Robotic assembly of timber joints using reinforcement learning”. In: *Automation in Construction* 125 (2021), p. 103569.
- [52] A. Kramberger, A. Kunic, I. Iturrate, C. Sloth, R. Naboni, and C. Schlette. “Robotic assembly of timber structures in a human-robot collaboration setup”. In: *Frontiers in Robotics and AI* 8 (2022), p. 768038.
- [53] T. Chen, J. Xu, and P. Agrawal. “A Simple Method for Complex In-hand Manipulation”. In: *Conference on Robot Learning* (2021).

- [54] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. *OpenAI Gym*. 2016. eprint: [arXiv:1606.01540](https://arxiv.org/abs/1606.01540).
- [55] O. : M. Andrychowicz et al. “Learning dexterous in-hand manipulation”. In: *The International Journal of Robotics Research* 39.1 (2020), pp. 3–20.
- [56] Y. Zhou, S. Sueda, W. Matusik, and A. Shamir. “Boxelization: Folding 3D Objects into Boxes”. In: *ACM Trans. Graph.* 33.4 (July 2014). ISSN: 0730-0301.
- [57] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. “Sim-to-Real: Learning Agile Locomotion For Quadruped Robots.” In: *Robotics: Science and Systems*. 2018.
- [58] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. “Learning agile and dynamic motor skills for legged robots”. In: *Science Robotics* 4.26 (2019), eaau5872.
- [59] J. Xu, T. Du, M. Foshey, B. Li, B. Zhu, A. Schulz, and W. Matusik. “Learning to fly: computational controller design for hybrid UAVs with reinforcement learning”. In: *ACM Transactions on Graphics (TOG)* 38.4 (2019), pp. 1–12.
- [60] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter. “Control of a quadrotor with reinforcement learning”. In: *IEEE Robotics and Automation Letters* 2.4 (2017), pp. 2096–2103.
- [61] T. Du, J. Hughes, S. Wah, W. Matusik, and D. Rus. “Underwater Soft Robot Modeling and Control With Differentiable Simulation”. In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4994–5001. DOI: [10.1109/LRA.2021.3070305](https://doi.org/10.1109/LRA.2021.3070305).
- [62] OpenAI et al. *Solving Rubik’s Cube with a Robot Hand*. 2019. arXiv: [1910.07113](https://arxiv.org/abs/1910.07113) [[cs.LG](https://arxiv.org/abs/1910.07113)].
- [63] E. Coumans and Y. Bai. “Pybullet, a python module for physics simulation for games, robotics and machine learning”. In: (2016).
- [64] E. Todorov, T. Erez, and Y. Tassa. “Mujoco: A physics engine for model-based control”. In: *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2012, pp. 5026–5033.
- [65] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, et al. “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)*. 2021.
- [66] Z. Hou, J. Fei, Y. Deng, and J. Xu. “Data-efficient hierarchical reinforcement learning for robotic assembly control applications”. In: *IEEE Transactions on Industrial Electronics* 68.11 (2020), pp. 11565–11575.
- [67] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox. “Closing the sim-to-real loop: Adapting simulation randomization with real world experience”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8973–8979.

- [68] S. Rakshit and S. Akella. “The influence of motion paths and assembly sequences on the stability of assemblies”. In: *IEEE Transactions on Automation Science and Engineering* 12.2 (2014), pp. 615–627.
- [69] S.-K. Kim and M. Likhachev. “Parts assembly planning under uncertainty with simulation-aided physical reasoning”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 4074–4081.
- [70] J. Xu, T. Chen, L. Zlokapa, M. Foshey, W. Matusik, S. Sueda, and P. Agrawal. “An end-to-end differentiable framework for contact-aware robot design”. In: *Robotics Science Systems* (2021).
- [71] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, and S. Coros. “Add: Analytically differentiable dynamics for multi-body systems with frictional contact”. In: *ACM Transactions on Graphics (TOG)* 39.6 (2020), pp. 1–15.
- [72] Y. Narang, K. Storey, I. Akinola, M. Macklin, P. Reist, L. Wawrzyniak, Y. Guo, A. Moravanszky, G. State, M. Lu, et al. “Factory: Fast Contact for Robotic Assembly”. In: *arXiv preprint arXiv:2205.03532* (2022).
- [73] H. Chen, W. Wan, and K. Harada. “Planning to Build Soma Blocks Using a Dual-arm Robot”. In: *2021 IEEE International Conference on Development and Learning (ICDL)*. IEEE. 2021, pp. 1–7.
- [74] W. Wan, K. Harada, and K. Nagata. “Assembly sequence planning for motion planning”. In: *Assembly Automation* (2017).
- [75] V. N. Hartmann, A. Orthey, D. Driess, O. S. Oguz, and M. Toussaint. “Long-horizon multi-robot rearrangement planning for construction assembly”. In: *arXiv preprint arXiv:2106.02489* (2021).
- [76] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [77] 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.
- [78] M. Yu, L. Shao, Z. Chen, T. Wu, Q. Fan, K. Mo, and H. Dong. “RoboAssembly: Learning Generalizable Furniture Assembly Policy in a Novel Multi-robot Contact-rich Simulation Environment”. In: *arXiv preprint arXiv:2112.10143* (2021).
- [79] J. De Winter, I. El Makrini, G. Van de Perre, A. Nowé, T. Verstraten, and B. Vanderborght. “Autonomous assembly planning of demonstrated skills with reinforcement learning in simulation”. In: *Autonomous Robots* 45.8 (2021), pp. 1097–1110.
- [80] Y. Fan, J. Luo, and M. Tomizuka. “A learning framework for high precision industrial assembly”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 811–817.
- [81] J. Luo and H. Li. “A learning approach to robot-agnostic force-guided high precision assembly”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 2151–2157.

- [82] Z. Zhu and H. Hu. “Robot learning from demonstration in robotic assembly: A survey”. In: *Robotics* 7.2 (2018), p. 17.
- [83] J. J. Roldán, E. Crespo, A. Martín-Barrio, E. Peña-Tapia, and A. Barrientos. “A training system for Industry 4.0 operators in complex assemblies based on virtual reality and process mining”. In: *Robotics and computer-integrated manufacturing* 59 (2019), pp. 305–316.
- [84] B. Tang, M. A. Lin, I. Akinola, A. Handa, G. S. Sukhatme, F. Ramos, D. Fox, and Y. Narang. “Industreal: Transferring contact-rich assembly tasks from simulation to reality”. In: *arXiv preprint arXiv:2305.17110* (2023).
- [85] B. Tang, I. Akinola, J. Xu, B. Wen, A. Handa, K. Van Wyk, D. Fox, G. S. Sukhatme, F. Ramos, and Y. Narang. “AutoMate: Specialist and Generalist Assembly Policies over Diverse Geometries”. In: *Robotics: Science and Systems*. 2024.
- [86] X. Zhang, C. Wang, L. Sun, Z. Wu, X. Zhu, and M. Tomizuka. “Efficient sim-to-real transfer of contact-rich manipulation skills with online admittance residual learning”. In: *Conference on Robot Learning*. PMLR. 2023, pp. 1621–1639.
- [87] X. Zhang, M. Tomizuka, and H. Li. “Bridging the Sim-to-Real Gap with Dynamic Compliance Tuning for Industrial Insertion”. In: *ICRA*. 2024.
- [88] J. Luo, Z. Hu, C. Xu, Y. L. Tan, J. Berg, A. Sharma, S. Schaal, C. Finn, A. Gupta, and S. Levine. “Serl: A software suite for sample-efficient robotic reinforcement learning”. In: *arXiv preprint arXiv:2401.16013* (2024).
- [89] M. Noseworthy, B. Tang, B. Wen, A. Handa, N. Roy, D. Fox, F. Ramos, Y. Narang, and I. Akinola. “Forge: Force-guided exploration for robust contact-rich manipulation under uncertainty”. In: *arXiv preprint arXiv:2408.04587* (2024).
- [90] L. Ankile, A. Simeonov, I. Shenfeld, and P. Agrawal. “JUICER: Data-Efficient Imitation Learning for Robotic Assembly”. In: *arXiv* (2024).
- [91] L. Ankile, A. Simeonov, I. Shenfeld, M. Torne, and P. Agrawal. *From Imitation to Refinement – Residual RL for Precise Assembly*. 2024. arXiv: 2407.16677 [cs.RO]. URL: <https://arxiv.org/abs/2407.16677>.
- [92] H. Huang, D. Wang, A. Tangri, R. Walters, and R. Platt. “Leveraging symmetries in pick and place”. In: *The International Journal of Robotics Research* (2024), p. 02783649231225775.
- [93] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitzmann. “Neural descriptor fields: Se (3)-equivariant object representations for manipulation”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 6394–6400.
- [94] A. Simeonov, Y. Du, Y.-C. Lin, A. R. Garcia, L. P. Kaelbling, T. Lozano-Pérez, and P. Agrawal. “Se (3)-equivariant relational rearrangement with neural descriptor fields”. In: *Conference on Robot Learning*. PMLR. 2023, pp. 835–846.
- [95] J. Yang, C. Deng, J. Wu, R. Antonova, L. Guibas, and J. Bohg. “Equivact: Sim (3)-equivariant visuomotor policies beyond rigid object manipulation”. In: *arXiv preprint arXiv:2310.16050* (2023).

- [96] D. Wang, S. Hart, D. Surovik, T. Kelestemur, H. Huang, H. Zhao, M. Yeatman, J. Wang, R. Walters, and R. Platt. “Equivariant diffusion policy”. In: *arXiv preprint arXiv:2407.01812* (2024).
- [97] J. Seo, N. P. Prakash, X. Zhang, C. Wang, J. Choi, M. Tomizuka, and R. Horowitz. “Contact-rich SE (3)-Equivariant Robot Manipulation Task Learning via Geometric Impedance Control”. In: *IEEE Robotics and Automation Letters* (2023).
- [98] W. Lian, T. Kelch, D. Holz, A. Norton, and S. Schaal. “Benchmarking off-the-shelf solutions to robotic assembly tasks”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 1046–1053.
- [99] L. Fan, Y. Zhu, J. Zhu, Z. Liu, O. Zeng, A. Gupta, J. Creus-Costa, S. Savarese, and L. Fei-Fei. “Surreal: Open-source reinforcement learning framework and robot manipulation benchmark”. In: *Conference on robot learning*. PMLR. 2018, pp. 767–782.
- [100] Y. Lee, E. S. Hu, and J. J. Lim. “IKEA furniture assembly environment for long-horizon complex manipulation tasks”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 6343–6349.
- [101] M. Heo, Y. Lee, D. Lee, and J. J. Lim. “FurnitureBench: Reproducible Real-World Benchmark for Long-Horizon Complex Manipulation”. In: *Robotics: Science and Systems*. 2023.
- [102] J. Luo, C. Xu, F. Liu, L. Tan, Z. Lin, J. Wu, P. Abbeel, and S. Levine. “FMB: a Functional Manipulation Benchmark for Generalizable Robotic Learning”. In: *arXiv preprint arXiv:2401.08553* (2024).
- [103] I. Melckenbeeck, S. Burggraeve, B. Van Doninck, J. Vancraen, and A. Rosich. “Optimal assembly sequence based on design for assembly (DFA) rules”. In: *Procedia CIRP* 91 (2020), pp. 646–652.
- [104] M. Agrawala, D. Phan, J. Heiser, J. Haymaker, J. Klingner, P. Hanrahan, and B. Tversky. “Designing effective step-by-step assembly instructions”. In: *ACM Transactions on Graphics (TOG)* 22.3 (2003), pp. 828–837.
- [105] S. Ong, M. Chang, and A. Nee. “Product disassembly sequence planning: state-of-the-art, challenges, opportunities and future directions”. In: *International Journal of Production Research* 59.11 (2021), pp. 3493–3508.
- [106] M. Santochi, G. Dini, and F. Failli. “Computer aided disassembly planning: state of the art and perspectives”. In: *CIRP Annals* 51.2 (2002), pp. 507–529.
- [107] Y. Wang, N. J. Weidner, M. A. Baxter, Y. Hwang, D. M. Kaufman, and S. Sueda. “REDMAX: Efficient & flexible approach for articulated dynamics”. In: *ACM Transactions on Graphics (TOG)* 38.4 (2019), pp. 1–10.
- [108] H. Zhao. “A fast sweeping method for eikonal equations”. In: *Mathematics of computation* 74.250 (2005), pp. 603–627.
- [109] R. Coulom. “Efficient selectivity and backup operators in Monte-Carlo tree search”. In: *International conference on computers and games*. Springer. 2006, pp. 72–83.

- [110] T. Ebinger, S. Kaden, S. Thomas, R. Andre, N. M. Amato, and U. Thomas. “A general and flexible search framework for disassembly planning”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 3548–3555.
- [111] J. J. Kuffner and S. M. LaValle. “RRT-connect: An efficient approach to single-query path planning”. In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 2. IEEE. 2000, pp. 995–1001.
- [112] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. “Shapenet: An information-rich 3d model repository”. In: *1512.03012* (2015).
- [113] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su. “Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding”. In: 2019, pp. 909–918.
- [114] F. Xiang et al. “SAPIEN: A SimulATED Part-based Interactive ENvironment”. In: June 2020.
- [115] X. Wang, B. Zhou, Y. Shi, X. Chen, Q. Zhao, and K. Xu. “Shape2motion: Joint analysis of motion parts and attributes from 3d shapes”. In: 2019, pp. 8876–8884.
- [116] Z. Yan, R. Hu, X. Yan, L. Chen, O. van Kaick, H. Zhang, and H. Huang. “RPM-Net: Recurrent Prediction of Motion and Parts from Point Cloud”. In: 38.6 (2019), 240:1–240:15.
- [117] R. Hu, W. Li, O. Van Kaick, A. Shamir, H. Zhang, and H. Huang. “Learning to predict part mobility from a single static snapshot”. In: 36.6 (2017), pp. 1–13.
- [118] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, and D. Panozzo. “ABC: A big CAD model dataset for geometric deep learning”. In: 2019, pp. 9601–9611.
- [119] K. D. D. Willis, Y. Pu, J. Luo, H. Chu, T. Du, J. G. Lambourne, A. Solar-Lezama, and W. Matusik. “Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Construction from Human Design Sequences”. In: 40.4 (2021).
- [120] B. Jones, D. Hildreth, D. Chen, I. Baran, V. G. Kim, and A. Schulz. “Automate: A dataset and learning approach for automatic mating of cad assemblies”. In: *ACM Transactions on Graphics (TOG)* 40.6 (2021), pp. 1–18.
- [121] O. Aslan, B. Bolat, B. Bal, T. Tumer, E. Sahin, and S. Kalkan. “AssembleRL: Learning to Assemble Furniture from Their Point Clouds”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 2748–2753.
- [122] Y. Li, K. Mo, L. Shao, M. Sung, and L. Guibas. “Learning 3D Part Assembly from a Single Image”. In: *European conference on computer vision (ECCV 2020)* (2020).
- [123] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. “Pointnet: Deep learning on point sets for 3d classification and segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

- [124] K. Goldberg, B. V. Mirtich, Y. Zhuang, J. Craig, B. R. Carlisle, and J. Canny. “Part pose statistics: Estimators and experiments”. In: *IEEE Transactions on Robotics and Automation* 15.5 (1999), pp. 849–857.
- [125] 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”. In: *ACM Trans. Graph.* 41.6 (2022).
- [126] S. Brody, U. Alon, and E. Yahav. “How attentive are graph attention networks?” In: *arXiv preprint arXiv:2105.14491* (2021).
- [127] S. R. Chhatpar and M. S. Branicky. “Search strategies for peg-in-hole assemblies with position uncertainty”. In: *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*. Vol. 3. IEEE. 2001, pp. 1465–1470.
- [128] J. C. Triyonoputro, W. Wan, and K. Harada. “Quickly inserting pegs into uncertain holes using multi-view images and deep network trained on synthetic data”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 5792–5799.
- [129] J. Xu, S. Kim, T. Chen, A. R. Garcia, P. Agrawal, W. Matusik, and S. Sueda. “Efficient Tactile Simulation with Differentiability for Robotic Manipulation”. In: *6th Annual Conference on Robot Learning*. 2022.
- [130] R. Alami, J.-P. Laumond, and T. Siméon. “Two manipulation planning algorithms”. In: *WAFR Proceedings of the workshop on Algorithmic foundations of robotics*. AK Peters, Ltd. Natick, MA, USA. 1994, pp. 109–125.
- [131] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani. “Manipulation planning with probabilistic roadmaps”. In: *The International Journal of Robotics Research* 23.7-8 (2004), pp. 729–746.
- [132] A. Lodi, S. Martello, and M. Monaci. “Two-dimensional packing problems: A survey”. In: *European journal of operational research* 141.2 (2002), pp. 241–252.
- [133] J. Jylänki. *A thousand ways to pack the bin-a practical approach to two-dimensional rectangle bin packing*. 2010. URL: <https://raw.githubusercontent.com/juj/RectangleBinPack/master/RectangleBinPack.pdf>.
- [134] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).
- [135] K. Doshi, M. Bagatella, and S. Coros. “Problem Space Transformations for Generalisation in Behavioural Cloning”. In: *arXiv preprint arXiv:2411.04056* (2024).
- [136] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling. “Residual policy learning”. In: *arXiv preprint arXiv:1812.06298* (2018).
- [137] D. Makoviichuk and V. Makoviychuk. *rl-games: A High-performance Framework for Reinforcement Learning*. https://github.com/Denys88/rl_games. May 2021.
- [138] C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, and A. Selle. “The material point method for simulating continuum materials”. In: *Acm siggraph 2016 courses*. 2016, pp. 1–52.

- [139] L. Lan, D. M. Kaufman, M. Li, C. Jiang, and Y. Yang. “Affine body dynamics: Fast, stable & intersection-free simulation of stiff materials”. In: *arXiv preprint arXiv:2201.10022* (2022).
- [140] Y. Wang, J. Zheng, Z. Chen, Z. Xian, G. Zhang, C. Liu, and C. Gan. “Thin-shell object manipulations with differentiable physics simulations”. In: *arXiv preprint arXiv:2404.00451* (2024).
- [141] Y. She, S. Wang, S. Dong, N. Sunil, A. Rodriguez, and E. Adelson. “Cable manipulation with a tactile-reactive gripper”. In: *The International Journal of Robotics Research* 40.12-14 (2021), pp. 1385–1401.
- [142] T. Dai, J. Wong, Y. Jiang, C. Wang, C. Gokmen, R. Zhang, J. Wu, and L. Fei-Fei. “Acde: Automated creation of digital cousins for robust policy learning”. In: *arXiv e-prints* (2024), arXiv–2410.
- [143] H. Jiang, H.-Y. Hsu, K. Zhang, H.-N. Yu, S. Wang, and Y. Li. “Phystwin: Physics-informed reconstruction and simulation of deformable objects from videos”. In: *arXiv preprint arXiv:2503.17973* (2025).
- [144] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, et al. “Laion-5b: An open large-scale dataset for training next generation image-text models”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 25278–25294.
- [145] M. Deitke, R. Liu, M. Wallingford, H. Ngo, O. Michel, A. Kusupati, A. Fan, C. Laforte, V. Voleti, S. Y. Gadre, et al. “Objaverse-xl: A universe of 10m+ 3d objects”. In: *arXiv preprint arXiv:2307.05663* (2023).
- [146] L. Shao, T. Migimatsu, and J. Bohg. “Learning to scaffold the development of robotic manipulation skills”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 5671–5677.
- [147] R. H. Wilson. “Geometric reasoning about assembly tools”. In: *Artificial Intelligence* 98.1-2 (1998), pp. 237–279.
- [148] H. Shi, H. Xu, S. Clarke, Y. Li, and J. Wu. “RoboCook: Long-Horizon Elasto-Plastic Object Manipulation with Diverse Tools”. In: *arXiv preprint arXiv:2306.14447* (2023).
- [149] M. Xu, W. Yu, P. Huang, S. Liu, X. Zhang, Y. Niu, T. Zhang, F. Xia, J. Tan, and D. Zhao. “Creative Robot Tool Use with Large Language Models”. In: *2nd Workshop on Language and Robot Learning: Language as Grounding*. 2023.
- [150] X. Zhang, S. Jain, B. Huang, M. Tomizuka, and D. Romeres. “Learning generalizable pivoting skills”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 5865–5871.
- [151] T. Chen, J. Xu, and P. Agrawal. “A System for General In-Hand Object Re-Orientation”. In: *Conference on Robot Learning* (2021).