Automated Motion Planning for Robotic Assembly of Discrete Architectural Structures

by

Yijiang Huang

B.S. Mathematics,
University of Science and Technology of China (2016)

Submitted to the Department of Architecture
in partial fulfillment of the requirements for the degree of

Master of Science in Building Technology

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

©Yijiang Huang, 2018. All rights reserved.
The author hereby grants to MIT permission to reproduce and to distribute publicly paper and electronic copies of this thesis document in whole or in part in any medium now known or hereafter created.

Author .................................................................
Department of Architecture
May 24, 2018

Certified by ..............................................................
Caitlin T. Mueller
Associate Professor of Architecture
and Civil and Environmental Engineering
Thesis Supervisor

Accepted by .............................................................
Sheila Kennedy
Professor of Architecture,
Chair of the Department Committee on Graduate Students
Automated Motion Planning for Robotic Assembly of Discrete Architectural Structures

by

Yijiang Huang

Submitted to the Department of Architecture on May 24, 2018, in partial fulfillment of the requirements for the degree of Master of Science in Building Technology

Abstract

Architectural robotics has proven a promising technique for assembling non-standard configurations of building components at the scale of the built environment, complementing the earlier revolution in generative digital design. However, despite the advantages of dexterity and precision, the time investment in solving the construction sequence and associated robotic motion grows increasingly with the topological complexity of the target design. This gap between parametric design and robotic fabrication congests the overall digital design/production process and often confines designers to geometries with standard topology.

In the goal of filling this gap, this research presents a new robotic assembly planning framework called Choreo, which eliminates human-intervention for parts that are typically arduous and tedious in architectural robotics projects. Specifically, Choreo takes discrete spatial structure as input, and then assembly sequence, end effector pose, joint configuration, and transition trajectory are all generated automatically. Choreo embodies novelties in both algorithm design and software implementation. Algorithm-wise, a three-layer hierarchical assembly planning framework is proposed, to gradually narrow down the computational complexity along the deep and branched search tree emerging in this combined task and motion planning problem. Implementation-wise, Choreo’s system architecture is designed to be modularized and adaptable, with the emphasis on being hardware-agnostic and forging a smooth integration into existing digital design-build workflow. Case studies on fabrication results of robotic extrusion (also called spatial 3D printing) are presented to demonstrate Choreo’s power on efficiently generating feasible robotic instructions for assembling shapes with non-standard topology and across the scales.

Thesis Supervisor: Caitlin T. Mueller
Title: Associate Professor of Architecture and Civil and Environmental Engineering
“Le vent se lève!...”

- Paul Valéry, “Le Cimetière Marin”
Acknowledgments

First of all, I would like to express my thanks to my advisor, Caitlin Mueller, for her generosity, encouragement, and support throughout my application, research, and my life at MIT. She showed me many important facets of academia, and taught me to always do research and teaching with a big picture in mind, and most importantly with curiosity, enthusiasm, and joy. The work presented in this thesis originated from the intellectual freedom and flexibility that she gave me. I am excited and honored to continue exploring my interest in computational geometry and robotic planning with her as a Ph.D. student in the next few years.

I would like to thank all of my friends and colleagues at MIT, past and present, for making this place such a open-minded and warm place for research and living. I especially wish to thank my Building Technology family for their love and intellectual support; Caelan Garrett at MIT CSAIL for his patience and passion on many productive discussions since the very beginning of this research; Prof. Michel Goemans at MIT Mathematics and Prof. Brian Williams at MIT CSAIL for their brilliant teaching and research discussions, from which I was exposed to many ingredients of this thesis; my students at 4.S48 Design for Robotic Assembly for their passion, talent and creativity that showed me the joy of teaching, and their endurance on their first-time TA; Justin Lavellee, Jen O’Brien, and Chris Stewart for their patience and help on supporting the physical making involved in this studies; Duncan Kincaid and his colleagues at CRON for their efforts on maintaining the computing and plotting machines; MIT’s soccer community for always being there and ready to play when I get stuck in my research.

I want to express my gratitude to my fiancée Stella Zhijing Zhang, for her love, caring and faith in my abilities. She was always present even when I was deeply lost in my research with my unblinking gaze at the screen or unrestful pacing in front of the white board. Her purity, gentleness and love towards nature and humanity shines a special beam of light on my life and my work. I wish that we can continue to push the boundary of knowledge together forever. Finally, it is hard to adequately express
my gratitude and thanks to my parents for giving me the resources, support and love to pursue my dream.
# Contents

1 Introduction ............................................. 11
   1.1 Background ........................................... 11
       1.1.1 Overview .................................... 11
       1.1.2 Scope and assumptions ....................... 12
       1.1.3 Challenges ................................ 13
   1.2 Related work ......................................... 13
       1.2.1 Robotic assembly for architecture .......... 14
       1.2.2 Classic assembly planning ................... 15
       1.2.3 Computer graphics ............................ 18
       1.2.4 Manipulation planning ....................... 21
       1.2.5 Task and motion planning .................... 24
   1.3 Thesis outline ...................................... 28

2 Methodology ........................................... 29
   2.1 Conceptual overview ............................... 29
   2.2 Model input ........................................ 33
   2.3 Sequence planning module ....................... 33
       2.3.1 Problem formulation ......................... 34
       2.3.2 Solving the CSP ............................. 40
   2.4 Motion planning module ......................... 43
       2.4.1 Semi-constrained Cartesian planning .......... 44
       2.4.2 Retraction planning ......................... 51
       2.4.3 Transition planning ......................... 51
Chapter 1

Introduction

1.1 Background

1.1.1 Overview

Architectural robotics has proven a promising technique for assembling nonstandard configurations of building components at the scale of the built environment, complementing the earlier revolution in generative digital design. In recent years, the sharp reduction of industrial robotics' cost has made investment in these advanced manufacturing machines more accessible than ever, converting the industrial robot into a cost efficient tool to materialize bespoke design [UNESCE, 2005].

However, despite the advantages of the decreasing hardware cost, dexterity, and precision of these multi-axis machines, the time investment in solving the construction sequence and associated robotic motion grows increasingly with the topological complexity of the target design. The level of automation in this design-assemble workflow is still comparably low, due to the technical challenge of finding a feasible assembly sequence and generating trajectories for the multi-axis robots. While transitioning between a digital design model and machine code for a 3-axis gantry machine is easy and direct, for multi-axis robots, gaining fine levels of control and bypassing the complexity of generating collision-free robotic trajectory is much more nuanced and subtle, which requires significant effort. Existing investigations in the
field of architectural robotics often involve manual planning of a path guidance for the robot’s end effector, followed with tedious diagnosis for potential problems in a trial-and-error manner. This slow and convoluted workflow deviates from the initial purpose of having such a digital design-assemble workflow: to forge a smooth and direct transition from digital design to real-world machine materialization; instead, the current process requires a complete re-program for the robot whenever the target geometry has a small change. This technical challenge in the assembly planning and programming of the robots congests the overall digital design / production process and often confines designers to geometries with standard topology with repetitive patterns. In order to close this gap and enable more possibilities for discrete architectural robotic assembly, an automated assembly planning system is needed, which calls for a more systematic and explicit computational exploration of assembly constraints and robotic motion planning.

This thesis presents a new algorithmic framework for robotic assembly planning, which embodies an hierarchical algorithm to integrate assembly sequence and motion planning. The planning framework is implemented as a flexible assembly planning tool, called Choreo, that allows users to input unconstrained spatial structures, and harvest automatically generated feasible assembly sequence and robotic trajectories. Case studies are presented to show the computational planning system’s power in enabling automated planning for robotic assembly of complex structures with non-standard topologies, which hasn’t been shown possible before.

1.1.2 Scope and assumptions

Discrete spatial structure’s definition broadly includes all the 3D structures that are consist of individual elements, which are connected to each other via structural joints and behave as a system when load is applied. The assembly planning problem is defined as: given a discrete spatial structure’s design model, the robot needs to be assigned a coordinated sequence of transition and assemble actions, to manipulate raw or sorted individual elements in a specific order to construct the designated design. There are essentially two main classes of robotic assembly applications: (1) spatial
extrusion (also called spatial 3D printing) and (2) spatial positioning (also called pick-and-place). In this thesis, for demonstration purpose, all the algorithmic framework description and case studies use spatial extrusion with a fix-base industrial robot arm as a concrete problem instance. However, the planning framework and the assembly planning tool can be configured to work with spatial positioning and the author is currently working on the implementation to support this.

In this thesis, the computation of the planning is all performed offline before execution. The robot is assumed to work in a deterministic world, which means that it optimistically believe that the environment that it is interacting with will behave exactly as specified in the computed plan. The planning is constrained to be purely geometric that the generated plan does not have a meaningful time parametrization embedded, and thus does not address the dynamics of the system. A fixed-base robot is assumed to be used, and support for mobile or aerial robots is left as future work.

1.1.3 Challenges

The fundamental challenges of assembly planning problems addressed by this thesis is that they requires finding solutions to large-scale problems that inherently depend on constraints at a much finer level of detail. The long planning horizon and intricate 3D configurations of elements involved in the design introduce computational challenges that has not been shown solvable before. Scattered in several different fields, separate techniques exist for either assembly planning for intricate design without considering the robots, or robot manipulation planning on rather small-scale problems compared to the problems discussed here. The resolution of the problem requires integrating insights from all of the existing work to a new comprehensive robotic assembly planning framework.

1.2 Related work

In response to the need for automation and integration described above, this section summarizes previous efforts in the area of automatic assembly. Key research from
five distinct fields, (1) robotic assembly for architecture, (2) mechanical assembly, (3) computer graphics, (4) manipulation planning and (5) task and motion planning is presented, with contributions and drawbacks highlighted. The aim of this section is to demonstrate why an integrated planning system, which combines features from all five of the above fields, is needed for robotic assembly to be fully accessible to architectural designers. In addition, during the composition of this research, the author found that the disconnection between disciplines caused similar ideas emerging in different contexts without referencing or acknowledging each other. The literature review presented here is intended to unify the efforts across a broad range of spectrum and provide future researchers a unified review on the related work in robotic assembly planning.

1.2.1 Robotic assembly for architecture

The exploration of robotic assembly in different architecture-scale application contexts, such as spatial positioning and extrusion [Gramazio et al., 2014], has focused on the design of application-specific processes and associated hardware systems. In all of these applications, researchers have encountered a similar problem: the generation of feasible robotic trajectories that do not collide with objects in the workspace [Parascho et al., 2017, Eversmann et al., 2017, Sondergaard et al., 2016]. Current solutions to this problem typically involve an intuition-based trial-and-error method. For a robot’s configuration during assembly, designers generate end-effector poses on the assembly geometry to achieve linear end-effector movement. For transition trajectories, designers manually generate guiding curves for the end-effector to follow, which hover over the workspace within a safety distance [Sondergaard et al., 2016]. Utilizing industrial robot’s built-in commands like Linear movement (LINE) or Point-To-Point (PTP) [wikipedia KRL, 2018], users rely on the built-in interpolation method to translate end-effector assignment to joint trajectories that are free of collisions, respect joint limits, and avoid singularities. As a result, this requires much extra effort to diagnosis the planning failure in a trial-and-error manner. Furthermore, while this manual planning process is possible for design with repetitive topological pattern,
such as spatial lattices with layers in zig-zag pattern [Hack and Lauer, 2014, Helm et al., 2015, BranchTech, 2018], planning the construction sequence and robotic motion is much more nuanced for designs with arbitrary topologies. The algorithmic difficulty of this assembly planning problem is preventing researchers from fully utilizing robots for fabrication with complicated geometry, which requires an explicit reasoning of the geometrical relationship between assembly elements and full robotic configuration.

In recent years, there has been some success in tackling this problem using automated motion planning by using a single-query robotic motion planning algorithm [Parascho et al., 2017] or an online control strategy [Giftthaler et al., 2017b, Giftthaler et al., 2017a] to compute transition trajectory between pre-programmed assembly primitives. However, the construction sequence in the existing work was still assigned manually, taking advantage of either the sparse or the repetitive topological nature of the target geometry. Recent progress in the field has lead to sensor-enabled online robotic control [Jeffers, 2016, Giftthaler et al., 2017b, Sandy and Buchli, 2018, Giftthaler et al., 2017a]. However, a global planning tool, which combines assembly sequence planning and associated robotic motion planning, is still in absence. As pointed out in [Giftthaler et al., 2017b], the combination of an autonomous control scheme with a “higher-level planner” that is “able to negotiate cluttered environment” is a key step to enable robotic assembly systems to safe operate in densely populated workspaces [Eversmann et al., 2017].

1.2.2 Classic assembly planning

There is a large body of work in classic assembly planning, also called mechanical assembly planning, dating back to the 1980s and 1990s with the influential work by Homem de Mello [De Mello and Sanderson, 1990], Wilson [Wilson, 1992], and others [De Fazio and Whitney, 1987, Woltery, 1989]. The focus of this line of research was generating of sequences that allow robots in an industrial assembly line setting to assemble a product based on design CAD files. The primary concerns in this work were to satisfy low-level constraints such as mutual blocking relationship during
assembly, mating constraints, and tolerances, etc. The methods involved focus on
the product itself, but the robots that perform the assembly are not considered. This
assumption is equivalent to assume that there are several magical hands that are
floating in the air and can achieve whatever poses that the assembly needs. In general,
this might be a valid assumption for this domain since the environment and assembly
robot can be specifically engineered to the task at hand. However, given a specific
robotic setup and a set of mechanical parts to be assembled, the robot's configuration
space significantly constrains the reachable end effector poses [Lozano-Pérez, 1983]
and thus requires considering the robots and the assembly object simultaneously in
the assembly planning (see section 1.2.4).

The key contributions in the area of classic assembly planning include (1) the
mathematical formulation of assembly planning problem under certain assumptions
(2) the proposal of several compact and efficient representations of intermediate as-
sembly states and (3) theoretical characterization of computational hardness of sub-
classes of the involved problems. A few key research is highlighted here and inter-
ested readers are referred to the survey papers [Jiménez, 2013, Ghandi and Masehian,
2015, Bahubalendruni and Biswal, 2016] and section 2.3.2 of Heger’s thesis [Heger,
2010] for a more detailed and extensive overview of the problems explored and meth-
ods used in classic assembly planning approaches.

Common across the literature of classic assembly planning is the approach of
planning assembly by disassembly. While early work uses precedence graph to de-
terminethe relationship between parts, Hommm de Mello and Sanderson first intro-
duced AND/OR graph to enrich the encoding of different disassemblies [De Mello
and Sanderson, 1990]. This work provided a compact representation of all possible
assembly sequences of a given product and it considers a disassembly planning prob-
lem that was then executed in reverse. The fully assembled goal configuration of an
assembly is maximally constrained and thus the branching factor during disassembly
searching is smaller than it would be for the direct assembly planning. Started from
an undirected graph of component connections generated from a CAD model of the
desired assembly (graph of liaisons [Bourjault, 1984, De Fazio and Whitney, 1987]),
several algorithms has been proposed to check all cutsets of this graph of liaisons with feasibility checking, in order to construct an AND/OR graph representing all valid assembly sequences [Röhrdanz et al., 1997, Homem de Mello and Sanderson, 1988, De Mello and Sanderson, 1991]. Notice that AND/OR graph can encodes the general type of assembly where multiple subassemblies are allowed to be assemble in one step (non-monotonic assembly), while the work presented in this thesis is limited to binary, monotonic [Wolter, 1989] assembly where only a single component is added at each step.

To overcome the generate-and-check inefficiency of cutset methods, Wilson introduced blocking graphs as a data structure that allowed computation of assembly partitioning in polynomial time based on the geometric feature of the input components in his seminal Ph.D. thesis [Wilson, 1992]. A directional blocking graph encodes blocking relationship between components for a given direction as directed edges. Connected components of a blocking graph represent removable connected components as a subassembly in the specified direction. Exploiting the fact that blocking relationships change only discretely and remain constant over ranges of potential directions, a non-directional block graph (NDBG) is proposed as a graph of directional blocking graphs, which enables a compact encoding of physical constraints and feasible separability of assemblies. Guibas et al. extended this work to allow parts to be moved by multi-step and rotational motions [Guibas et al., 1995].

Given a representation of the relationship between components, finding valid decompositions (cutsets) of the assembly into subassembly, also called as assembly decomposition problem, is a difficult computation problem and has been proven to be NP-hard [Kavraki and Kolountzakis, 1995]. Kavraki and Kolountzakis proved that partitioning a planar assembly into two connected sub-assemblies that can be separated is an NP-complete problem. Goldwasser et al. proved that even finding an approximate solution to the decomposition problem is very hard to solve [Goldwasser et al., 1996]. In the problem’s most general case, the assembly-planning problem is PSPACE-hard [Natarajan, 1988]. However, for a special class of the problem (monotone, binary assemblies, infinite translations or infinitesimal rigid motions only), there
exists algorithms to solve the problem in polynomial time [Wilson, 1992].

Note that nearly all work in mechanical assembly planning dates back to the 1990s and earlier. This area of research has gone somewhat out of fashion with many problems left unsolved - specifically that of how to integrate the constraints of the assembler, the robots, into the planning scheme as addressed by this thesis. Nevertheless, this line of research laid out the fundamental mathematical framework for describing the mutual blocking and separability of an assembly. Its ideas, methods, and analysis hasn’t been fully explored and should be resurrected and acknowledged with the recent increasing interest in the digital fabrication and assembly.

1.2.3 Computer graphics

Existing assembly-related research in computer graphics can be summarized into two categories: (1) computational design methods with assembly sequence as a physical constraint (2) fabrication techniques that require resolving assembly sequence for input object.

In contrast to classical assembly planning where assembly planning works like a technical assessment tool for arbitrary input mechanical parts, work in the computational assembly design focuses on the generation of interesting objects with the existence of assembly sequence as a constraint. Existing work addresses the problem for objects with specific features, such as 3D polynomino puzzles [Lo et al., 2009], 3D burr puzzles [Xin et al., 2011], voxelized recursive interlocking puzzles [Song et al., 2012], furnitures with interlocking joints [Fu et al., 2015] and planar interlocking pieces [Schwartzburg and Pauly, 2013]. Many of this work uses a constraint graph to represent the physical and separability constraints between components, tries to identify a direct acyclic component in the constraint graph, and uses data structure similar to AND/OR graph to represent assembly sequences. Although addressing very similar problems to the ones discussed in classic assembly planning (section 1.2.2), only a few computer graphics work in assembly-related topic builds on or acknowledges the existing work in classic assembly planning.

Agrawala et al. presents a computational framework to automatically generate
assembly instructions, given assembly object geometry, orientations, and optional grouping and ordering constraints on the object’s parts [Agrawala et al., 2003]. Their method integrates visualization cost functions into the existing assembly planning scheme to facilitate the generation of understandable, clear, and concise assembly instructions. Moving towards assembly in an architectural scale, Deuss et al. have studied the physical construction of self-supporting structures with masonry blocks by leveraging external cables to help finding self-supported partially constructed structures [Deuss et al., 2014]. However, they ignores potential collision problems among the cables, which is equivalent to abstracting away the constraints brought by the “assemblers” in the classic assembly planning.

The other line of research invents new hardware systems with assembly sequencing algorithms to enable new fabrication and assembly opportunities. Existing efforts focus on adding more degrees of freedom to the 3D printing technology. Traditional 3D printers have 3 degree of freedom (DoF): the printer’s hotend can move to any point in the gantry workspace but cannot rotate. To avoid collisions between the hotend and the printed part, input object is sliced into layers and printed layer by layer. WirePrint [Mueller et al., 2014] proposes an efficient way to print wireframe meshes, where edges in the mesh are directly extruded in 3D space. Compared to traditional 3D printed objects, WirePrint generates the wireframe of a model by slicing it horizontally and filling each slice with zigzaging filaments. This approach constraints the types of meshes that can be printed. To improve flexibility, Peng et al. introduce a 5-dof printer, which modifies a standard delta 3D printer by adding two rotation axes to the print bed [Peng et al., 2016]. Following up this work, Wu et al. present a printing sequence planning algorithm for this 5-dof printer [Wu et al., 2016]. They separate the consideration of collision constraint and connectivity constraint and use a heuristic constraint removal technique to identify a subgraph in the full constraint graph to balance the existence of feasible extruder orientations and printing sequences. However, there are two main drawbacks of this method: (1) the constraint removal phase has no guarantee on the existence of feasible order that satisfies connectivity constraint, which might lead to unguided backtracking,
especially for spatial trusses that have dense internal structures as considered in this thesis; (2) their algorithm abstracts away the potential kinematics constraint of the machine and is only applicable to the machine that has a rotatable print bed and thus impossible to be extended to extrusion at a larger scale. Recently, Dai et al. extend the degrees of freedom of the print bed by using an industrial robot to hold it, and present two successive shape decomposition algorithms to handle volume-to-surfaces and surfaces-to-curves decomposition, while taking into account the support and robotics constraints [Dai et al., 2018].

Huang et al. work on printing sequence planning problem for robotic spatial printing [Huang et al., 2016]. Instead of extending desktop 3D printer like in [Peng et al., 2016, Wu et al., 2016], they mount a 3D printing extruder on an industrial robot and extrude plastic directly in 3D space. They consider the printing sequence planning problem as a constrained search problem and developed a constrained graph cut algorithm to divide the input wireframe into subgroups, which enables a tractable searching with backtrack in each subgroups. However, for simplicity, their method only considers collision constraint between robot’s end effector and the partially printed structure and abstracts away the robot’s kinematic constraint. They use an ad-hoc method to generate feasible guiding curve for the robot’s end effector to follow, which results in very slow computation and trajectory with no guarantee on satisfying kinematics and collision constraint in transition motion. This thesis extends this work to a more general sequence searching algorithm that takes into account robot’s kinematic constraint using constraint satisfaction encoding (see section 2.3). Empirical comparison to this heuristic planning method used in [Huang et al., 2016] can be found in section 3.1.

Related problems There are other work that is not produced in computer graphics community, but also addresses the assembly sequence planning problem to connect digital design and physical assembly process. In architectural design context, Tai presents a computational design framework to design interlocking wooden frames with consideration of assembly sequence [Tai, 2012]. This work uses non-directional
free graph, a variant of non-directional block graph [Wilson, 1992], to represent the blocking relationship between elements and identified feasible subassemblies by recursively finding strong connected component in the graph. It uses genetic algorithm to optimize elements’ placement to satisfy both fabrication constraint and assembly sequence constraint. In the context of 3D printing in bio-engineering, Gelber et al. presented a heuristic backtrack searching algorithm to generate printing sequence to enable micro-scale freeform 3D printing on a purpose-built isomalt 3D printer [Gelber et al., 2018a]. They are the first to identify that the joint positioning errors are caused by beam compliance, and include it as a cantilever constraint in the sequence searching process [Gelber et al., 2018a, Gelber et al., 2018b]. This finding influenced the printing direction routing part in the sequence planning module presented in this thesis (see section 2.3.2).

In computer-human interaction (CHI) community, existing work explores new interface between human designers and machine fabrication. The assembly sequence planning problem has been one of the major technical barriers that constraints the design space but the resolution of this hard planning problem is usually not the focus of the work in CHI community. Recently, following the seminar work WirePrint [Mueller et al., 2014] and On-the-Print [Peng et al., 2016], Wu et al. explore synthesizing advanced reality (AR) technology and robotic spatial extrusion [Peng et al., 2018]. Readers are referred to [Baudisch and Mueller, 2017] for a recent survey on the fabrication-related work in CHI.

1.2.4 Manipulation planning

Moving into the world of robotics and planning, the focus of the research shifts from design’s end to the robot’s end. The robotic planning community has developed many approaches for motion planning that identify trajectories by searching in the continuous space of robot joint angles. Recent approaches perform this search using either sampling [LaValle, 2006] or optimization [Ratliff et al., 2009, Schulman et al., 2014]. In manipulation planning, the goal is not only to move robot without colliding with objects in the environment, as in classical motion planning, but
also to contact, operate, and interact with objects in the world to achieve high-level goal. This problem has been addressed from the earliest days of algorithmic motion planning [Lozano-Pérez, 1981, Lozano-Pérez et al., 1987, Wilfong, 1991]. Alami et al. pioneered the modern treatment of this problem, which involves continuous grasps as well as continuous placements [Alami et al., 1990, Alami et al., 1994]. They introduce manipulation graph that breaks the problem of one robot moving one object in a potentially complex environment into several problems of moving between connected components of the combined configuration space where each component shares the same grasp. A solution is an alternating sequence of transit and transfer paths corresponding to the robot moving while its hand is empty and the robot moving while holding an object. Siméon et al. expand this work to a more realistic setting that requires multiple regrasps, by using probabilistic roadmaps [Kavraki et al., 1996] to approximate each component of the robot’s configuration space. However, maintaining an explicit characterization of the free configuration space can be prohibitively expensive in high-dimensional planning problems.

Stilman and coauthors address a version of manipulation problem called navigation among movable obstacles (NAMO), where the robot must reach a specified location among a field of movable obstacles [Stilman and Kuffner, 2008, Stilman et al., 2007]. In order to solve monotonic NAMO problem instances, where each object can be moved at most once, they plan backwards from the goal and use swept volumes to recursively determine which additional object must be moved recursively. Den Berg et al. [Berg et al., 2009] provide a probabilistically complete algorithm for NAMO. However, the algorithm assumes the ability to fully characterize the connected components of the configuration space of the robot at each planning step, which is computationally prohibitive for robotic configuration space with more than two dimensions.

Hauser and coauthors identify a generalization of manipulation planning as multi-modal motion planning, i.e. planning for systems with multiple modes, representing different sub-manifolds of the configuration space under constraints [Hauser and Latombe, 2010, Hauser and Ng-Throw-Hing, 2011]. The key insight in this approach is that, as in the manipulation graph, one can conceptualize the planning process
as alternating between moving in a single mode, where the constraints are constant, and switching between modes. So the solution requires being able to plan within a single mode and identifying configurations where modes can change, which is in general specific to the task. Hauser and coauthors provided a probabilistically complete algorithm to solve this type of problem assuming that effective single-mode planners and mode-transition samplers are available. However, this pure uni-directional sampling-based algorithm has trouble solving high-dimensional problem as encountered in architectural robotic assembly problems. In this thesis, an hierarchical planning approach is proposed to efficiently address the assembly planning problem, which is a specific class of multi-modal motion planning problem, by first solving the assembly sequence and then using a purpose-built single-mode planners to obtain robot’s motion.

Barry et al. defined a bidirectional rapidly-exploring random tree (RRT) search of the combined configuration space [Barry et al., 2013]. The individual moves of this algorithm consider complete plans to reach the goal, ensuring the suggested motions have some chance of being on the path to reach the goal. They proposed a two-layer hierarchy planner, where the higher level plans only for the manipulated object without considering the robot, with the objective of identifying relevant mode transitions to guide the full planner. However, this planner was limited to domains with only one movable object and had running times on the order of minutes.

Rearrangement planning is a special instance of pick-and-place planning where all objects have explicit goal poses. These problems are very similar to robotic assembly planning problems addressed in this thesis, where object goal poses are specified in the input design model. Extending Stilman et al.’s work on NAMO [Stilman et al., 2007] to non-monotone problem instances, Krontiris and Bekris provided an algorithm that constructs a probabilistic roadmap (PRM) [Kavraki et al., 1996] in the combined configuration space [Krontiris and Bekris, 2015, Krontiris and Bekris, 2016]. To overcome the inefficiency caused by backtracking search, they propose a faster approximate by performing a topological sort on a constraint graph between objects using minimum removal paths (MCR) [Hauser, 2012, Hauser, 2013]. The use
of the PRM is able to recover completeness for problems that could not be solve by greedy backtrack search planner, but the lack of search guidance forces the planner to explore a large number of arrangements.

Dogar et al. propose an algorithm for multi-robot grasp planning using a constraint satisfaction problem (CSP) formulation [Dogar et al., 2015]. They use a domain-specific assumptions, regrasps, to remove constraints in the constraint graph, balancing between solvability and number of regrasps performed.

1.2.5 Task and motion planning

While motion planners deal beautifully with geometric constraints in high-dimensional configuration spaces, they cannot work with abstract features of the domain; they can plan how to move the robot’s joints to pick up an object but not to decide the order of tasks to satisfy logistic constraints. Motion planners also have limited ability to deal with partially-specified states. Symbolic planners, on the other hand, show its strength in reasoning over very large sets of discrete states by manipulating partial descriptions. However, these task planners work by enumeration: all possible operations are considered in a state to expand them in a search. In geometric domains, enumeration of possible operations and complete symbolic descriptions of states is difficult or impossible, depending on selected vocabulary and desired resolution.

Recent work in task and motion planning (TAMP) combines discrete task planning and continuous motion planning to simultaneously plan for discrete objectives as well as robot motions. This work aims to enable robots to operate in applications such as cooking, which require discrete choices of which objects to grasp or cook as well as continuous choices of which joint angles and object poses can physically perform each task. A key challenge is that often physical constraints such as collision, kinematic, and visibility constraints can restrict which high-level actions are feasible. The pioneering work Asymov system conducts an interleaved search at the symbolic and geometric levels [Cambon et al., 2009]. Their approach can be viewed as using the task planner as a heuristic to guide the motion planning approach. However, since the task-level planner has no knowledge on the geometry, its value as a heuristic is
limited. Plaku and Hager took a similar approach [Plaku and Hager, 2010].

A natural extension to the classic symbolic planning system paradigm is to introduce *semantic attachments*, where predicates whose truth value is established not via logic assertion but by calling external program that operates on a geometric representation of the state [Dornhege et al., 2012]. A motion planner can serve to implement such a predicate to determine the reachability of one configuration from another. However, the computational expense of calling a motion planner is the major challenge of this approach. This leads to a desire to minimize the considered set of object placements to limit the branching factor of the search, but considering only a sparse number of placements will limit the generality of the planner. Additionally, semantic attachments are ignored during heuristic generation, which leads to a geometrically uninformed symbolic search and may result in considerable inefficiency due to heuristic plateaus. Erdem et al. augment a task planner that is based on explicit causal reasoning with the ability to check for the existence of a feasible robot path [Erdem et al., 2011].

Lozano-Pérez and Kaelbling introduce a HPN approach, a regression-based symbolic planner uses *generators*, which perform fast approximate motion planning to select geometric parameters [Kaelbling and Lozano-Pérez, 2011]. Garrett et al. give an algorithm (HBF) for planning hybrid spaces by using approximation of the planning problem to guide the backward generation of successor actions to be considered in the forward search [Garrett et al., 2015]. Both of these two approaches require an inverse model to specify the generators, in order to be compatible with their backward searches.

The FFRob algorithm of Garrett et al. samples a set of object poses and robot configurations and then plans with them using a search algorithm that incorporates geometric constraints in its heuristic [Garrett et al., 2018a]. An iterative version of the algorithm has been proposed to have probabilistically complete guarantee and exponentially convergence. Their recent work generalizes this strategy of iteratively sampling then searching from pick-and-place domains to domains with arbitrary conditional samplers [Garrett et al., 2018b].
Pandey et al. and de Silva et al. use hierarchical task networks [Erol et al., 1994] to guide a search over plan skeletons, which are discrete action sequences with unbound continuous variables, using knowledge about the task decomposition [Pandey et al., 2012, de Silva et al., 2013]. The search over plan skeleton backtracks in the event that is unable to bind the free variables of a skeleton. Lagriffoul et al. proposed a constraint-satisfaction approach to interleaves the symbolic and geometric searches and focus on limiting the amount of geometric backtracking [Lagriffoul et al., 2014]. They generate a set of approximate linear constraints on robot configurations and object poses that allow them to efficiently determine which assignments are feasible and rule out a large amount of useless branches and thus significantly limiting backtracking. For each step in the plan skeleton, they call an RRT to determine a feasibility of the transit motion. Viewed from constraint satisfaction perspective, their approach can be thought of doing backtracking with forward checking of the kinematic constraints and a fixed value ordering. Lozano-Pérez and Kaelbling take a similar approach but leverage constraint satisfaction problem (CSP) operating on discretized variable domains to bind free variables [Lozano-Pérez and Kaelbling, 2014]. The sequence planning module (section 2.3) proposed in this thesis adopts a similar technique by using CSP to bind free geometric variables on a plan skeleton. However, it relaxes the requirement on feasible whole path’s existence and uses black-box feasibility checker to allow flexibility and scalability.

Erdem et al. plan at the task-level using a boolean satisfiability (SAT) solver, initially ignoring geometric constraints, and then attempt to produce motion plans satisfying task-level actions [Erdem et al., 2011]. If an induced motion planning problem is infeasible, the task-level description is updated to indicate motion infeasibility using domain-specific diagnostic interface. Dan tam et al. extend this approach by formulating task and motion planning problem more generally as a satisfiability modulo theories (SMT) problem [T. Dantam et al., 2016]. They use an incremental constraint solver to add motion constraints to the task-level logical formula when a candidate task plan is found. Their approach adjusts to motion planning failure automatically and allows previously failed motion planning queries to be reconsidered. The algo-
rithms proposed in Erdem et al. [Erdem et al., 2011] and Dantam et al. [T. Dantam et al., 2016] both assume a priori discretization of all continuous values apart from configurations. Srivastava et al. remove this restriction by using symbolic references to continuous parameters [Srivastava et al., 2014]. Lagriffoul and Andres propose to use answer set programming (ASP) [Lifschitz, 2008] to enable a richer failure explanation mechanism (culprit detection) at the interface between symbolic and geometric search spaces [Lagriffoul and Andres, 2016]. They use a domain-specific interfaces to bind values for symbolic references and update update the task-level description when none is available. Toussaint and coauthors formulate the binding of geometric variables as a nonlinear constrained optimization problem and use a hierarchy of bounds on the nonlinear program to prune plan skeletons [Toussaint, 2015, Toussaint and Lopes, 2017].

Existing research in TAMP is generally not user-friendly. As many planning problems involve complex relationships defined on a high-dimensional, continuous parameter space that requires specialized procedures to sample and evaluate satisfying values, there are very few systems that expose a black-box-like functionality for users to plug in customized samplers to work with their planning domains. Recently, Garrett et al. propose STRIPStream that extends the STRIPS planning language [Fikes and Nilsson, 1971] to support a generic, declarative specification for specialized sampling procedures that are treated as blackboxes, which marks an attempt in the TAMP community to increase the usability of its algorithms [Garrett et al., 2018c].

**Related problems** Beyond the theoretical approaches described above, there also exists some research in applied robotics relevant to architectural construction. For example, researchers in autonomous robotic assembly have been expanding possibilities by incorporating physical constraints into their planning algorithm and designing novel robotic systems [Cortsen et al., 2012, McEvoy et al., 2014]. Other researchers have studied the distributed assembly of structures using robot teams [Komendera, 2014, Yun, 2010]. However, this research usually focuses on assembly sequencing and scheduling problems for purpose-built hardware systems and ignores the geometric
motion planning.

1.3 Thesis outline

In summary, there is a rich literature of work related to robotic assembly for architecture, ranging from theoretical research in robotic task and motion planning to examples of built work of considerable intricacy. However, the field is nevertheless lacking an integrated, general-purpose method that can be applied systematically across many project types while also handling the geometric and topological complexity of contemporary architectural design. This work addresses this gap by presenting a new assembly planning algorithm framework and a modularized implementation that is adaptable to various assembly applications and hardware setups, and provides an intuitive modeling interface for designers. Chapter 2 presents the assembly planning algorithm, starting from model input (section 2.2), and goes through layers of its planning hierarchy: first sequence planning layer (section 2.3) and then motion planning layer (section 2.4). An engineering module, post processing module, is presented in section 2.5 to increase usability and adaptability of the computed results. Section 2.6 presents the engineering ideas behind the implementation of the assembly planning tool Choreo. Chapter 3 shows three case studies with computation statistics and fabrication results to demonstrate Choreo’s efficiency and power. Chapter 4 concludes the thesis and points to directions of future work.
Chapter 2

Methodology

This chapter introduces a new computation framework that can efficiently handle the problem of robotic assembly planning. First, section 2.1 gives a conceptual overview of the entire framework’s hierarchy and introduces its three main modules. Then, detailed problem formulations and associated solution strategies are described for sequence planning module (section 2.3), motion planning module (section 2.4) and post processing module (section 2.5).

2.1 Conceptual overview

In response to the issues addressed in chapter 1, a general robotic assembly planning system should have following capabilities:

1. Can take general discrete structures as input, with minimal possible restriction on the geometry

2. Can generate assembly sequence and associated poses for assembly operation, while satisfying constraints

3. Can solve for collision-free transition trajectory between assembly operations

4. Should be robot model and hardware agnostic
5. Should provide interface to synthesize generated trajectory and end-effector control commands and possibilities to be integrated into online control system

Creating an assembly planning system that meets the above capabilities is a challenge because (1) the computational complexity inherent from assembly planning problem (2) the engineering complexity for creating an interface bridging design to robotic planning.

Assembly planning is technically a subclass of high-dimensional robot manipulation problems, or more generally, task and motion planning (TAMP) problems (see section 1.2.4 and 1.2.5), which requires planning a coordinated sequence of motions that involve extrusion, picking, placing or manipulating specific type of construction materials, as well as moving through free space. Compared to general manipulation planning problems, architectural robotic assembly problems differ from typically studied TAMP problem in three key aspects. First, the discrete horizon of the assembly problems is much longer than many TAMP benchmarks [Lagriffoul, 2016], which often only require manipulating a couple of objects. Because each element must be assembled once (monotonic assembly) and the goal object poses are specified by the input design geometry, the assembly horizon is known in advance, compared to general rearrangement problems where each object is allowed to be moved more than once and thus planning horizon can be infinitely long [Krontiris and Bekris, 2016]. Thus, assembly planning requires identifying an ordering for object manipulation and then fitting this order to a fixed plan skeleton and binding the required geometric parameters. In contrast, problems in TAMP domains generally have unsettled action plans - it is not initially clear which actions are needed and in which order to perform these actions to complete a task.

Second, assembly problems involve physical constraints such as stiffness and stability that are not typically found in TAMP benchmarks. These constraints impact many state variables at once, making them challenging to effectively incorporate in many discrete task planning algorithms. Rather than directly using existing TAMP algorithms, a specialized system is developed in this research that incorporates several existing ideas but, because of its specialization to assembly planning, can scale
to complex models.

Third, common task specification languages for planning systems, such as planning domain Planning Domain Definition Language (PDDL) [McDermott et al., 1998] are not intuitive for architects and designers. The requirement of specifying task domains, predicates, action’s preconditions and effects departs from the architectural language of shape and geometry, and thus creates a gap between an architect’s geometric model and robotic task specification for planning. This gap in the modeling interface prohibits these algorithms to be easily adapted to architectural robotic assembly applications.

To harness the computational and engineering challenges posed by the assembly planning problem, this thesis proposes a planning framework that uses a hierarchical task and motion planning approach. The proposed planning framework incorporates three key modules as shown in figure 2-1. Instead of searching for a solution considering all parts of the searching tree at once, the proposed approach identifies and breaks the problem into two key sub-problems, sequence planning and motion planning, and isolates the sub-problems. First, the sequence planner (section 2.3) takes a discrete structure as input and outputs the assembly sequence and associated feasible end effector directions. The generation of assembly sequence and pruned end effector directions cuts the sequence-dependent ties between the sequence and motion planning subproblems, narrows down the searching space, and thus enables efficient solution searching. Next, with fixed assembly sequence and focused end effector directions, the motion planner (section 2.4) finalizes the choice of end effector pose for each assembly and plans for the robot’s entire joint trajectory during and between assemblies. Finally, the post processor (section 2.5) tags the computed trajectory plan with associated assembly information and outputs a complete assembly plan. After the framework is completed, it is possible for the user to insert more fine-tuned detail related to their application and hardware setup through the tagging system added by the post processor.

These modules, along with the framework inputs and outputs, are described in greater detail in the following sections. An example problem of using fixed-base six-
Figure 2-1: Overview of the assembly planning framework.

axis robot to spatially extrude a discretized linear frame structure is used to illustrate and exemplify the details of each module, but the system is general and can also apply to other robotic assembly tasks, for example, spatial positioning of discretized surfaces or volumetric elements.

**Key assumptions** In this thesis, the planning starts with an assembly plan skeleton, or action sequence, that has a pre-defined repetitive pattern on the actions: for example, pick element \( o_i \) from material rack - move - place element \( o_i \) at position \( p_i \) - move or extrude element \( o_i \) at position \( p_i \) - move. Only one fixed-base robot is considered to be working on the assembly task, and the relative position between the machine and the assembly object is defined by users. There is no other constraint on the robot - it can has any degrees of freedom and any type of joint (rotational or prismatic). The planner needs to assign a correct order to assign object \( o_i \) to each action in the plan skeleton, and bind variables to fully specify robot’s configurations during and between assembly steps. The generated plan is purely **geometric** - it has
no time stamps assigned (so the speed of movement is unspecified) and does not take into account of any rigid body dynamics of the system [Giftthaler et al., 2017a]. The plan is generated completely **offline** before execution and it assumes the environment is **deterministic**, where the robot and the assembly will behave as specified in the plan. The robot is not equipped with any sensor ability during execution.

### 2.2 Model input

The assembly planning framework starts with an input 3D model from a designer. The model type is flexible, but should represent overall geometry, topology, and discretization for robotic assembly. Discretization can be performed by designer intuition or through an algorithmic meshing or decomposition approach; the framework is agnostic to how this step is carried out.

For the discretized linear frame structure, a standard node-member data representation is used. Nodes are described with 3D spatial coordinates in an indexed list, and linear members are described by their start and end node indices. In this way, the geometry and topology of the structure are efficiently described. Different cross sections and material properties could be assigned referring to specific member’s index.

In addition to the design model, users need to input the robot’s data that includes the geometry of its links, limitation (min-max value) and type (rotational or prismatic) of its joints. Users also need to model and input static collision objects in the work environment, which are assumed to be invariant during the entire planning horizon. Relative position between the robot and the assembly object is set by a 3D translation vector.

### 2.3 Sequence planning module

In this module, a sequence planner takes any discrete geometry as input and solve for the order of the assembly operation and associated feasible end effector poses.
Globally, the sequence planner computes assignment of objects to each action in the predefined plan skeleton, which requires reasoning on the geometric and physical constraints in this combinatorial search. Locally in each assembly task, the planner resembles a grasp planner that computes all collision-free end effector poses, given all the collision objects in the target assembly stage.

This section first identifies the key constraints arisen in the sequence planning problem and formulates the problem as a Constraint Satisfaction Problem (CSP) (Section 2.3.1). Then, a solving technique is proposed to solve this CSP problem (Section 2.3.2), which embodies two main techniques: (1) user-guided model decomposition (2) backtracking search with 1-level forward checking and value ordering.

### 2.3.1 Problem formulation

The assembly sequence planning problem requires assigning every assembly action with an element from the model and find the geometric configuration of feasible end effector path\(^1\) for each action. Before introducing the sequence planning problem’s formulation, axis conventions are described in figure 2-2. The end effector frame is positioned at the 3D printing extruder’s tip. An end effector’s pose is defined by such a frame, which can be uniquely determined by (1) frame’s origin, (2) z-axis, and (3) rotation angle around the z-axis. Every trajectory point gets a local frame \(\{p\}\) assigned to define the position and orientation of the end effector in that trajectory point. All of these coordinate frames are described in a common reference frame \(\{\text{base}\}\). In the following discussions, \(n\) denotes the total number of elements to be assembled in the model.

The input frame model contains a set of linear elements, \(O_1, \ldots, O_n\). Each element specifies a linear trajectory that the end effector’s tip needs to traverse while extruding material. A discretized representation of the linear trajectory is used, which divides the trajectory into a sequence of points under certain discretization resolution. These

\(^1\)In this thesis, the term path refers to a sequence of end effector poses. In general, a path is a geometrical description of a robot’s configuration, which can be used to describe joint configuration [Siciliano et al., 2009].
points only specify the end effector poses’ origins, which still have an infinite number of possible end effector’s orientation. In order to have a good printing result, the end effector is required to maintain its orientation when printing. Thus, the robot’s path for printing an element is determined by (1) point origins specified by the element’s linear path (2) orientation of the end effector.

For spatial 3D printing of linear frame structure, only one assembly action type is considered: extrude. The sequence of assembly actions are defined in an alternating pattern: extruder-move-. . .-move-extrude. An important simplification is made to eliminate the move action concatenating adjacent extrude actions, which differs from general plan skeleton that couples the task-level reasoning with full geometric-level reasoning [Lozano-Pérez and Kaelbling, 2014]. In this way, the checking of transition paths’ existence is simplified to kinematics feasibility check and collision check during assembly. This simplification is equivalent to assume that if robot has a collision-free kinematic configuration at the start and the end of each assembly step, transition motion planner (2.4.3) can always find a feasible transition trajectory. This assumption can be found in many work in TAMP [Lagiffoul et al., 2014] and is generally valid through all of the performed experiments.
Each action in the predefined assembly plan skeleton is specified with a constraint variable and a set of geometric parameters. A constraint variable is a symbol that names an assembly element. A geometric parameter ranges over a continuous geometric quantity that defines end effector’s pose. To bind these variables, a CSP planner is called to verify if the assembly plan skeleton is satisfiable. The correctness of an assembly plan skeleton is enforced by the constraints, which are expressed as relationships between assembled elements at each assembly step and end effectors’ pose during assembly.

To formulate a problem domain as a discrete constraint satisfaction problem (CSP), it is necessary to specify a set of constraint variables, a discrete domain of values for each variable, and a set of constraints. Constraints are specified by a set of variables to which they apply and a test function that maps an assignment of variable values to true or false [Dechter, 2003].

**Constraint variables and geometric variables** The CSP is encoded using constraint variables $O_i, i = 1, \ldots, n$, which represent the assembly element assignment for $i$th assembly action in the assembly action skeleton. Its value domain is $1, \ldots, n$, which represents the indices of elements in the input model.

Though not explicitly expressed as constraint variables, the geometric variables are pruned by the CSP solver and used to guide the solver’s search. The pruned geometric domains will be output as a part of the solution. Geometric variables used in this problem are $V_i, i = 1, \ldots, n$, which represent end effector’s direction for $i$-th assembly action in the assembly action skeleton. Its value domain is $1, \ldots, m$ and represents the indices of directions. The indices of those directions are referencing an ordered list of unit vectors sampled on a semi-sphere. The sampling size $m$ is set according to desired discretization granularity. Thus, choice of end effector directions can be described by indices referencing to this shared list.

Notice that an assigned value $v$ of $V_i$ alone cannot uniquely determine the pose of an end effector. One needs to determine rotation angle $r$ around the assigned direction value $v$ to determine the end effector’s pose for assembly (see figure 2-2). This
degree of freedom remains undetermined during the entire sequence planning process and its determination is postponed until the motion planning process (Section 3.4). Notice that the domain definition of this rotation angle $r$ is application-dependent. While all the assembly task share a continuous rotation angle domain in interval $[0, 2\pi]$ for spatial extrusion due to the application and end effector’s z-axis symmetric nature, general assembly tasks, for example spatial positioning, might need different rotation angle domain $r_i$ to be assigned to each assembly element, depending on the grasp relationship between end effector’s geometry and the target assembly element’s geometry.

**Constraints** Constraints relate the constraints variables to one another and restrain them to constant quantities. If all the constraints are collectively satisfiable, then an assembly plan skeleton is valid, and the pruned geometric variable domains specify the geometric details for subsequent motion planning. In the spatial printing domain, the following types of constraints are used:

**AllDiff($O_1, \ldots, O_n$):** Each assembly element is used only once by an assembly action. No disassembling and reassembling is allowed. Thus all assembly element assignment $O_i$’s value is different.

**Connectivity($O_1, \ldots, O_k$), $k = 1, \ldots, n$:** At each assembly step, the newly added element must be connected to existing structure or connected to the ground. Let Boolean matrix $A \in R^{m \times m}$ denotes the adjacency matrix of the input spatial truss design model:

$$A[i][j] = \begin{cases} 1, & \text{if element } O_i \text{ and } O_j \text{ share a node;} \\ 0, & \text{otherwise.} \end{cases}$$

And ground connectivity matrix $G \in R^{m \times 1}$:

$$G[i] = \begin{cases} 1, & \text{if element } O_i \text{ has a grounded node;} \\ 0, & \text{otherwise.} \end{cases}$$
Then the connectivity constraint can be expressed as:

\[
\text{FORALL } 1 \leq i \leq m, \text{ EXIST } 1 \leq j < i,
\]
\[
A[O_i][O_j] = 1 \text{ OR } G(O_i) = 1
\]

**ExistValidEndEffectorPose**\((S_1, \ldots, S_k), \ k = 1, \ldots, n: \)** This constraint checks if there exist a valid end effector pose for each assembly action in the assembly plan skeleton. At each assembly step, existing assembly elements \(O_1, \ldots, O_{k-1}\) are considered as collision objects. These collision objects will collide with the end effector in some of the poses specified by direction \(V_i\)’s value and rotation angle around the direction. \(V_i\)’s domain is pruned by the collision objects, eliminating the values that has no valid rotation angle. For spatial printing, a symmetric cone that encloses the end effector is used to avoid explicit check or sample all rotational values around the chosen direction. A graphical demonstration of this geometric pruning is shown in figure 2-3. This constraint can be expressed as:

\[
\text{FORALL } 1 \leq i \leq n, \\
\text{EXIST } a, 1 \leq a \leq m, \\
(\text{FORALL } 1 \leq j < i, \ T[O_i][O_j][a] = 1) \text{ AND } (\text{ExistValidKinematics}(a, O_1, \ldots, O_{i-1}, O_{static}))
\]

where the three-dimensional matrix \(T \in R^{n \times n \times m}:\)

\[
T[i][j][a] = \begin{cases} 1, & \text{if printing element } i \text{ with direction } a \\
& \text{does not collide with element } j. \\
0, & \text{otherwise}. \end{cases}
\]

and **ExistValidKinematics** is a function that returns true if there exist one rotation angle around the chosen direction \(a\) that enables whole-body kinematic solutions for the robot to traverse the path points of the current element, without colliding into already assembled object \(O_1, \ldots, O_{i-1}\) and static world collision objects \(O_{static}\), and return false otherwise. The kinematics checking function keeps sampling rotation
Figure 2-3: Illustration of the geometric pruning. The existence of already assembled element (the element on the top in (c)) restricts the collision-free end effector pose in current assembly task, which prunes out values in associated end effector rotation angle $R_i$’s domain in (a) and direction $V_i$’s domain in (b). The green regions are the valid region that does not collide with the collision objects.

angle in $[0, 2\pi]$ around direction $a$ and checking the existence of a feasible joint solution, until it finds the first solution, and return true immediately or return false if it succeeds the sampling timeout without finding a feasible rotation angle. This function does not guarantee the existence of feasible kinematic solution for all the rotation angles - it is used only to eliminate the case where collision-free end effector poses exist but without associated feasible kinematic solutions. Note that the computation involved in checking the end effector’s collision (FORALL $1 \leq j < i$, $T[O_i][O_j][a] = 1$) is much lighter than checking the existence of a feasible kinematic solution, thus might enable faster pruning in the search.

**Stiffness**$(S_1, \ldots, S_k)$, $k = 1, \ldots, n$: The stiffness constraint makes sure that the partial assembly at each assembly step is stiff and the maximal deformation due to gravity (or other constantly presented load) is bounded by a predefined tolerance. In the case of spatial 3D printing, the deformation of all the nodes can be calculated using finite element analysis. The constraint test function returns true if the maximal node deformation is smaller than the tolerance. Otherwise, it returns false.

**Stability**$(S_1, \ldots, S_k)$, $k = 1, \ldots, n$: The stability constraint checker returns true if the gravitational center’s projection on the supporting plane lies in the convex hull of all the grounded nodes, and returns false otherwise. It guarantees that the partially
assembled structure does not tip over as a rigid body.

Notice that for different type of discrete structure’s assembly, masonry vault’s assembly for example, since the mechanics involved is completely different to the one in spatial trusses, a different evaluation scheme for checking stability constraint can be added to check the static equilibrium of the partial assembly structure [Deuss et al., 2014, Livesley, 1992]. Also notice that the stability constraint and stiffness constraint are rarely encountered in classic constraint processing problems. To date, no existing techniques can efficiently evaluate this type of constraint in the CSP framework. The evaluation of the stability constraint commonly induce a large amount of overhead as they will be called many times by the CSP planner. Finding an efficient constraint encoding to accelerate computation speed is currently in the author’s investigation.

2.3.2 Solving the CSP

One key advantage of a CSP formulation is that it reduces our job to picking variables and constraints to represent the problem, and use a generic solver to do the search. It is generally easier to articulate and check constraints for a given assignment of the variables than to construct a problem-specific search strategy, which is particularly important for the variety of problems emerging in architectural assembly. However, although in general efficient representations of the variables and constraints are preferred, as they will be called many times by the solver, the ExistValidEndEffectorPose, Stiffness and Stability constraints described in the last section is usually significantly non-linear and impossible to find an analytical expression. A simple backtracking search with 1-level forward checking and dynamic variable ordering is proposed in this work as a baseline solver to the assembly sequencing problem. In addition, to limit the computation in a reasonable amount of time, an additional user-guided task decomposition is introduced before running the search algorithm. More in-depth investigation of more efficient formulation of constraints and the use of more advanced CSP solving strategies are left for future research.
**User-guided model decomposition**  Model decomposition involves grouping the discrete input model into several connected components. Taking advantage of user’s intuition on the geometric relationship, the decomposition breaks the whole assembly sequencing problem into several smaller ones, and then searching is confined in each of these small sub-problems, which scales down the size of the search space and leads to more efficient CSP solving in each component.

Existing automatic model decomposition techniques in the literature only demonstrate the ability to handle the decomposition of the surface mesh. Huang et al. developed a constrained graph decomposition method to iteratively decompose an input frame model into connected groups, while guaranteeing the physical stability of the group that is connected to the ground at each graph cut iteration [Huang et al., 2016]. However, this method uses elements’ height and mutual collision relationship as graph cut’s cost heuristic and only performs well with surface mesh. For models with a volumetric geometric feature, such as spatial lattices, it usually fails to scale down the search space in subgroups with evenly distributed element number.

Thus, in order to accelerate the computation for model with a large number of elements, the planning framework offers users the choice to manually group the elements to guide the search in CSP, based on their intuition on the geometric occlusion between the decomposed groups. This process can be easily done on standard 3D modeling software. The resulting decomposition has been proven to be effective in handling the task planning for many geometry instances that have not been shown feasible to be 3D printed by a robot in existing literature (see chapter 3). A more general automatic model decomposition is currently under investigation by the authors.

**Backtracking search with 1-level forward checking and dynamic variable ordering**  A backtracking search with dynamic variable ordering is applied to solve the CSP problem (chapter 5.3, [Dechter, 2003]). A domain-dependent heuristic is proposed to assist the variable ordering. Similar to the searching cost used in [Huang et al., 2016], a weighted-sum of three types of costs is used:
**Position cost:** An element $o$ that has lower $z$ coordinate and forms sharper angle with global $z$ axis is preferred. The position cost is defined as:

$$E_p(o, O_{\text{printed}}^k) = \frac{\text{center}(o).z - \min_z}{\max_z - \min_z} e^{\text{Angle}(o)}$$

where $O_{\text{printed}}^k = o_1, \ldots, o_k$, $\min_z$ and $\max_z$ is the minimal and maximal node $z$ coordinate in current layer. $\text{Angle}(o)$ evaluates the angle between $z$ axis and the vector formed by $o$, using one of its printed node as the vector’s origin.

**Collision cost:** Denote $\bar{O}_{\text{printed}}^k = O_{\text{printed}}^k \cup \{ o \}$ and $\bar{O}_{\text{unprinted}}^k = O_{\text{unprinted}}^k \setminus \{ o \}$. Although $o$ is printable, it might cause some remaining unprinted elements in $\bar{O}_{\text{unprinted}}^k$ to have no feasible orientation for the end effector in the following stage. Thus, the element $o$ with the minimal value of the following collision cost is preferred:

$$E_c(o, O_{\text{printed}}^k) = \frac{1}{|\bar{O}_{\text{unprinted}}^k|} \sum_{\bar{o} \in \bar{O}_{\text{unprinted}}^k} \exp(-B^2(\Omega(\bar{o}, \bar{O}_{\text{printed}}^k))),$$

where $B(\Omega) = \frac{A(\Omega)}{2\pi}$ and $A(\Omega)$ is the area of $\Omega$ on the unit sphere, or in its discretized expression, the number of sampled vectors that fall into $\Omega$. A small value of $E_c(o, O_{\text{printed}}^k)$ indicates that the remaining unprinted struts still have a wide range of orientations after $o$ is printed.

**Distance-to-base cost:** A grounded element $o$ that is further from the robot’s base should be printed first. The Distance-to-base cost is defined as: The Position cost is defined as:

$$E_d(o, O_k) = \begin{cases} \frac{\text{distance (robot base, center}(o))}{\text{max_to_base_distance}}, & o \text{ is grounded} \\ 0, & \text{otherwise.} \end{cases}$$

**Overall cost:** Then the overall cost function is defined as

$$\min_{o \in \mathcal{C}} E(o, O_{\text{printed}}^k) = \omega_p E_p(o, O_{\text{printed}}^k) + \omega_c E_c(o, O_{\text{printed}}^k) + \omega_d E_d(o, O_{\text{printed}}^k),$$

where $\omega_p, \omega_c$, and $\omega_d$ are weights for the
terms. In all the experiments included in this thesis, \( \omega_p = 1, \omega_c = 3, \omega_d = 1 \). This cost function is used as a heuristic to guide the variable ordering in the backtracking search.

In addition, every time the search assigns a new element to an action, the constraint caused by this newly added element is propagated one level down the search tree, pruning the geometric domains of unassigned elements.

**Routing printing directions** After the CSP planner finishes its searching and produces an assembly order, an additional process can be taken to optimize the printing directions. For the assembly steps that connect two existing nodes, there is an extra degree of freedom to choose the assignment start node and the end node. This assignment has recently been proven to be critical for the physical execution of spatial extrusion, due to the molten joint’s incapability to resist bending moment and elastic recoil effect [Gelber et al., 2018b]. Gelber et al. introduce a cantilever constraint to their assembly planning algorithm to address this problem: new elements cannot be connected to node \( p \), if any previously printed element connected to \( p \) is cantilevered [Gelber et al., 2018a]. An slightly relaxed version of this constraint is used here to route the printing direction: the direction that starts from the node with larger valence (number of connected elements) is preferred. Based on the experiments carried out by the author, the introduction of this direction routing process dramatically increased the rate of empirical printing success, although it does not make a big difference on the geometric planning level.

### 2.4 Motion planning module

The suspended plan skeleton obtained from the sequence planner contains the order of the assembly tasks \( o_1, o_2, \ldots, o_n \) and a range of collision-free end effector directions \( v_j, j \in [m_i] \) for each assembly task. In order to obtain a full kinematic solution for the robot, the motion planner needs to (1) determine the robot’s trajectory during each assembly task (2) plan for robot’s trajectory between assembly tasks. The
problem encountered here is a hybrid motion planning problem: Cartesian motion planning with constraints on end effector’s poses during assembly task and free motion planning without constraints on end effector’s pose in transition. In the proposed planning framework, this hybrid motion planning problem is solved in two phases: semi-constrained Cartesian planning (section 2.4.1) to resolve the constrained end effector poses and associated robot kinematic redundancy during each assembly task. Then, retraction planning (section 2.4.2) is added between the Cartesian motion and transition motion to enable a safer robot trajectory. Finally, transition planning (section 2.4.3) is used to compute robot’s trajectory in between adjacent assembly tasks. The sequential layout of transition motion, retraction motion, and Cartesian motion is explained in figure 2-4.

Figure 2-4: Illustration for sequential layout of transition motion, retraction motion, and Cartesian motion for each single assembly process.

2.4.1 Semi-constrained Cartesian planning

In many robotic assembly applications, the robot’s end effector is required to move in a linear movement, where the path points that the end effector’s tip needs to traverse is designated but its orientation has certain degrees of freedom [De Maeyer et al., 2017]. For example, spatial extrusion requires that the robot’s end effector’s tip, or the tip of the nozzle, to traverse the path points on the linear path formed by the element, but has freedom in choosing its direction and rotational angle around the direction. In addition, even when the end effector’s poses are fully determined, there
is still redundancy in choosing corresponding kinematic solutions. The planning for this type of motion is called semi-constrained Cartesian planning.

In this section, a graph-based semi-constrained Cartesian planner is proposed to resolve the end effector’s orientation and robot’s kinematics redundancy to fully determine robot’s joint configuration during each assembly process. As has discussed above, for spatial extrusion, in order to fully determine robot’s configuration in each individual extrusion task, the planner has three variables’ value to determine for each assembly task: (1) end effector direction \( \mathbf{v}_k \) (2) end effector’s rotation angle around its z-axis direction \( r_k \) and (3) kinematic solution while satisfying collision and speed limitation constraints:

\[
\mathbf{p}_k^i = (x_k^i, y_k^i, z_k^i), i \in \{1, \ldots, \text{number of path points in task } k\}
\]
\[
\mathbf{v}_k \in \text{direction domain } V_k
\]
\[
r_k \in [0, 2\pi)
\]

Solving robot’s kinematic solution problem means finding feasible joint position \( \theta_i \) for each pose \( \mathbf{p}_k \):

\[
\theta_k = (\theta_1^k, \ldots, \theta_d^k), d = \text{robot’s degrees of freedom}
\]
\[
\mathbf{p}_k^i = f(\theta_k)
\]

Where \( f \) represents robot’s kinematics. Notice that the inverse kinematics solution \( \theta_k \) for target end effector pose \( \mathbf{pose}_k \) is not unique and needs to be determined by the planning algorithm. Meanwhile, the computed joint solutions have to be collision-free with objects in the environment in corresponding assembly stage. In addition, the motion between consecutive joint solutions should adhere to robot’s maximum velocity and acceleration limitations so that the joint solution sequence is physically executable by the robot.

Graph-based path planning approach is the most popular approach so far to address these problems, because it produces determinate and globally optimal joint
solution without stochasticity [De Maeyer et al., 2017, ROS-I, 2018a]. The graph-based path planning algorithm starts with a list of given end effector poses for the robot to traverse and each end effector pose is assigned with parameters with tolerance ranges. With the tolerance, each given path pose represents a family of parameterized end effector poses and each pose in this family corresponds to a family of robot’s joint poses according to its inverse kinematics. These joint poses can be organized as vertices in a planning graph and edges only exist between joint poses that belongs to adjacent path pose families. Nodes that represent joint poses in collision can be pruned and edges that represents sharp turns of adjacent joint poses will not be added to the planning graph. Cost is assigned to each edge in the graph as the $L_1$ norm of the difference of the two adjacent joint poses. In this way, the semi-constrained Cartesian path planning problem is converted to a shortest path searching problem on a directed ladder graph and resulting path represents a sequence of joint poses with minimal joint difference between adjacent joint poses [De Maeyer et al., 2017].

However, the planning problems encountered in architectural robotic assembly usually involve very long planning horizon and two degrees of additional dimension on end effector choice per assembly. These features make a direct application of existing graph-based path planning algorithm impractical. For example, for spatial extrusion of a truss model with 300 elements, the storage of the corresponding planning graph will take 362 Gigabyte to store, which exceeds the RAM capacity of a common desktop computer. To harness this memory issue, a key observation is that this full expansion stores many joint poses that do not contribute to the optimal path and the solution’s existence. The quality of a solution is heavily dependent on the end effector discretization, which is inversely proportional to the computing time and memory. In this section, a sampling-based optimization algorithm is proposed to first search on a sparse representation of the planning graph and then expand this sparse graph representation into a full graph to apply shortest path search, when the graph’s size has been significantly reduced.
Extracting sparse ladder graph  This section first introduce a sparse representation of the planning graph, called sparse ladder graph, to help compress and locate the optimal part of the planning graph. Then, a sampling-based optimization algorithm is presented to help extracting an asymptotically optimal sparse representation. An important observation here is that the memory overhead is mainly caused by the storage of the graph edges connecting across all pairs of adjacent assembly tasks’ kinematics families.

This observation leads to the thoughts of incrementally build a sparse graph structure that describes the cost between end effector pose families, abstract away detailed information stored in the original edges connecting vertices representing joint poses. In the sparse ladder tree, each kinematics family is represented with a compact data structure called capsule, where only start and end path point’s corresponding joint poses are recorded (figure 2-5). For example, extrusion of element $i$ are discretized into 5 path points, 30 feasible end effector directions and 20 sampled rotation angles, then element $i$’s extrusion task has $30 \times 20 = 600$ capsules, instead of $30 \times 20 \times 5 = 3000$ vertices and associated explicit edges stored in the original full planning graph.

Edges in the sparse ladder graph are assigned with the optimal cost between capsules. When constructing edges, the minimal $L_1$ norm of joint pose difference between kinematic solutions of source capsule’s last path point and kinematic solutions of target capsule’s first path point. In this way, edges in the sparse ladder graph captures the optimal cost distance between the represented kinematics families and abstracts away the detailed connection between joint pose nodes.

Computing an optimal capsule path on the sparse ladder graph  The purpose of using sparse ladder graph is to find an optimal path of capsules to traverse all the assembly tasks. This path represents a fraction of the original planning graph that contains the optimal path of joint poses. Sampling-based algorithms are perfect to be used in this scenario, which allows an incremental construction of the sparse graph with an asymptotically optimality guarantee [Karaman and Frazzoli, 2011]. However, special initialization, sampling, feasibility checking, and connecting func-
Figure 2-5: Demonstration of capsules in the sparse ladder graph.

Implementations need to be implemented to make existing sampling-based planning algorithms comply with the hybrid discrete-continuous state space and the sequential layout of the sparse ladder graph.

Let $X \subseteq [n] \times [m] \times [0,2\pi)$ be the state space of the sparse planning problem, where $[m] \times [0,2\pi)$ parameterizes the end effector’s pose by assigning end effector’s direction with index $i \in [m]$ in a precomputed list of directions and rotation angle $\theta \in [0,2\pi)$. Note that each state correspond to a capsule. Let $X_{\text{obs}} \subseteq X$ be the state where capsule does not have feasible joint poses for some of the path points for the corresponding task. Let $X_{\text{free}}[i] = X - X_{\text{obs}}[i]$ be the resulting set of permissible states in assembly step $i$. Let $\delta : [n] \mapsto X$ be a sequence of states and $\Sigma$ be the set of all paths. The optimal path planning problem on a sparse ladder graph can be defined as the search for the path $\delta^*$ that minimizes the accumulated cost of the path while traversing each assembly task in a chronological order:

$$
\delta^* = \arg\min_{\delta \in \Sigma} \{ c(\delta) \mid \delta[i] = x(i, \cdot, \cdot), \forall i \in [n], \delta[i] \in X_{\text{free}}[i] \} \quad 48
$$
The cost denotes:

\[
c : X \times X \mapsto R_+
\]

\[
c(x, x') = \min_{J_1, J_2} ||J_1 - J_2||_{L_1}
\]

s.t. \( J_1 \in \text{InvKm}(\text{ExtractPose}(x)[\text{last path point}]) \)

\( J_2 \in \text{InvKm}(\text{ExtractPose}(x')[\text{first path point}]) \)

where InvKm denotes the inverse kinematic solver that returns all collision-free robot’s joint poses corresponding to given end effector pose. The function ExtractPose: \( X \mapsto \) end effector poses returns all the end effector poses that state \( x \) (capsule) encodes.

In this thesis, RRT* algorithm is applied on the sparse ladder graph for demonstration purpose (figure 2-6), while other probabilistically optimal sampling-based algorithm, e.g. PRM* can also be used. The complete description of these algorithms can be found in [Karaman and Frazzoli, 2011]. Key modifications of subfunctions are highlighted below to enable these algorithms to operate on the sparse ladder graph:

**InitSparseLadderGraph:** Instead of directly engaging in sampling across assembly steps, the algorithm starts with a sequential traversal along the chronological order of assembly actions to obtain a feasible initial solution for each assembly step. It sequentially samples \( X(i, \cdot, \cdot) \) to find a feasible state \( x_i \) from \( i = 1 \) to \( n \), each with a given timeout. For each \( i \in [n] \), if at least one feasible state is found in \( X(i, \cdot, \cdot) \) within a given timeout, \( X_{\text{free}}(i, \cdot, \cdot) \) is claimed to be non-empty and the planner continues to sample \( X(i + 1, \cdot, \cdot) \), otherwise the sequence plan is claimed to be infeasible and algorithm is terminated.

**Sample:** The sampler operates on a hybrid discrete-continuous state space, which returns a state \( x \in X(\cdot, \cdot, \cdot) \) that is generated from three different and stand-alone samplers. Each one of these three samplers generates independent and identically distributed samples from the corresponding state space. The generated samples uniquely determines (1) assembly task index \( i \) (2) end effector direction index \( j \) in assembly task \( i \)'s direction list, and (3) rotation angle \( \theta \in [2, 2\pi) \), which all together determine end effector’s poses along the path points in assembly task \( i \).
**CheckFeasibility:** State $x$’s feasibility can be verified by checking if all the encoded path points have feasible robot kinematics solutions. For state $x \in X(i, \cdot, \cdot)$ with task index $i$, a kinematic solution for a given end effector pose is pruned if it causes collision. Notice that each task has different set of collision objects, as elements assembled in previous tasks become collision objects in subsequent assembly tasks.

**Nearest and Rewiring:** Given a state $x \in X(i, \cdot, \cdot), 1 < i \leq n$, the function nearest returns the vertices with smallest cost to $x \in \mathcal{G} \cap X(i - 1, \cdot, \cdot)$, where $\mathcal{G}$ is the sparse ladder graph at the moment. Note that in this case, edge connections are confined to only vertices in adjacent assembly tasks, as skipping assembly tasks is not allowed. In contrast, general sampling-based algorithm operating on a continuous state space usually uses the nearest vertices that are contained in a ball of given radius centered at the new vertex $x$. For the same reason, the rewiring process are confined in adjacent assembly task $\mathcal{G} \cap X(i + 1, \cdot, \cdot)$, while RRT* rewires all the vertices in a ball centered at $x$.

![Figure 2-6: Demonstration on applying RRT* on sparse ladder graph. The optimal capsule path is highlighted and expansion of two adjacent capsules is depicted.](image)

**Extracting trajectory solution** The sampling-based algorithm returns a path $\sigma$ in the sparse ladder graph. The path is then expanded as a subgraph of the original planning graph to enable the use of standard shortest path search algorithms to find the sequence of joint poses with minimal cost. Each state (capsule) in the returned
path $\sigma$ is expanded by adding the intermediate path points’ kinematics solutions as nodes on the corresponding rungs and then constructing edges between all nodes on adjacent rungs, which corresponds to two successive path points (figure 2-6).

The expanded graph is a directed acyclic graph (DAG). Its topologically sorted feature makes it possible to compute shortest path in linear time, by processing the vertices on rungs in a topological order and chose the vertex with the minimum length (section 24.2 in [Cormen, 2009]). The resulting path gives a discretized joint trajectory for each assembly task in the assembly action sequence, which fully determines robot’s configuration during each individual assembly task.

2.4.2 Retraction planning

Retraction motion is a short segment of slow linear motion that is inserted between transition motion and cartesian motion as a buffer to allow the robot to safely transit from high speed to low speed when it’s approaching (or departing) the workpiece (figure 2-4). It is also called approaching and departing motion in grasp planning. In Choreo, the retraction planning constructs the linear segment by sampling in the set of feasible end effector’s directions that is constructed by the sequence planner and construct a line along this vector with a user-defined distance. The same feasibility checking strategy used in the sampling-based algorithm in the last section is applied here to verify the sampled direction’s feasibility. End effector’s orientation is kept unchanged during this retraction motion, same as the Cartesian extrusion motion. Notice that for more general assembly problems, for example spatial positioning, the direction of this retraction motion is related to the assembly and the end effector’s geometry, or the geometry of the joint formed by the assembly elements (e.g. interlocking joint between two wood elements).

2.4.3 Transition planning

Following semi-constrained Cartesian planning and retraction planning in the last two sections, transition planning computes a collision free joint trajectory connecting the
last joint pose in the departing retraction motion in assembly task $i$ and the first joint pose in the approaching retraction motion in assembly task $i + 1$. Note that this is a classic single-query motion planning call, taking into account of the present collision objects in assembly task $i$. The transition planner utilizes existing state-of-the-art motion planner for its transition trajectory planning (figure 2-7). The transition planner first tries to call the motion planner for directly connecting the target start and goal configurations. Upon failure, it replans by inserting a reset home pose between the start and goal configurations. The transition trajectories generated from three state-of-the-art motion planners are shown in figure 2-7. The results in figure 2-7 (b) shows that the CHOMP planner [Ratliff et al., 2009] finds it hard to generate a feasible transition plan at the first direct motion planning call at many assembly steps, and thus resetting itself to the home pose quite often.

![Figure 2-7: Transition planning with different planners: (a) STOMP [Kalakrishnan et al., 2011] (b) CHOMP [Ratliff et al., 2009] (c) RRT* [Karaman and Frazzoli, 2011]](image)

### 2.5 Post processing module

After the robot’s joint trajectories are generated, the post processing module groups and tags the results with associated information. This tagging process enables an easier importing and parsing of the results into various programming systems for application- and hardware-specific adjustment and fine-tuning, which is important to
make the planning framework usable in various robotic assembly applications with
different hardware setups. Three specific ways that the tagging process can be used
are described in this section.

2.5.1 Synthesis of hardware IO commands and trajectories

The generated robotic trajectory from the proposed planning framework is geomet-
rical and without timestamp information. In order to generate instructions for the
robot to interact with the physical world, the users need to weave IO commands to
synthesize the robot’s motion and its end effector’s behavior. Many existing robotic
assembly projects involve an offline programming process. In these projects, the
insertion of IO commands takes advantage of the inherent indexing of these traject-
ories, which is embedded on their generation on specific programming platform, for
example, the tree paths in the Grasshopper programming platform. To increase the
computed trajectory’s compatibility to programming platforms, Choreo’s trajectory
is formatted in a customized JSON format, which contains a hierarchical information
structure to maximize its readability and usability. Each element’s assembly process
contains several subprocesses, each of which is tagged with a subprocess type, for
example, transition, retraction_approach, extrusion, and retraction_depart (figure
2-8).

Many robotic assembly project needs robot to have different speed in different
phases of its motion. For example, for robotic spatial extrusion, the robot needs
to extrude material in a constant speed following a straight linear movement. It is
in general impossible for users themselves to generate end effector’s constant speed
movement by assigning speed for robot’s joints. Fortunately, most of the industrial
robots provide linear movement commands that take a tool center point (TCP) plane
to generate linear movement with a user-defined constant end effector speed. Thus,
this requires that the produced result contains both robot’s joint trajectory and the
associated TCP planes to allow users to choose according to the subprocess’s defini-
tion. To support this feature, when exporting computed trajectories, the planning
system performs forward kinematics to every joint configuration to compute corre-
sponding TCP plane. Both of these joint array and TCP array are packed with assembly task id, subprocess id, and subprocess type. In addition, main data type can be specified to indicate what kind of motion the subprocess is using - TCP data should be used if end effector linear movement with constant speed is desired, and joint data should be used if there is no constraint on the end effector’s speed.

Then the formatted trajectories can be imported into any programming environment, such as Grasshopper, with a simple customized parser, to decode the JSON file and allow a direct and visually friendly IO commands insertion. Users can easily insert robot commands, such as digital IO, analog IO, and wait time, into designated processes or in between processes. Then, existing robot simulation packages can be used to visualize and simulate robot’s trajectory and export brand-specific executable robot instruction code.

![Figure 2-8: Demonstration of synthesizing the robot’s trajectories with IO commands to control end effector’s behavior, using post processor’s tagging system.](image)

2.5.2 Application-oriented path modification

For many robot assembly processes, especially spatial extrusion, the variety of end effector design and material properties requires the incorporation of ad-hoc fabrication logic to achieve the desired visual results [Hack and Lauer, 2014, Helm et al., 2015] or
increase the product’s structural performance [Tam et al., 2018]. These fabrication logics, which are derived from physical extrusion experiments, usually involve local modification of an end effector’s pose, such as pressing or extruding following small circular movements at structural joints to create local “knots.”

The meta-data associated with the computed trajectories makes it easy for user to locate where to insert such micro path modifications. Such fine-tuning and path modifications usually require users to iterate on the parameters controlling robot’s and end effector’s behavior, until they find the best parameter setting based on experimental observations. For spatial extrusion, one needs to perform many experiments to find the delicate balance between robot’s linear moving speed while extruding, cooling air’s pressure, and extrusion rate. With the tagging system offered here, the fabrication parameter calibration process can go back and forth between the fine-tuning programming platform and physical tests, keeping the overall robot trajectory unchanged.

2.5.3 High-level base plan for online robotic control systems

Finally, the tagged trajectory generated from this system can be used as a base plan and be integrated with an online control system such as [Giftthaler et al., 2017b]. Such integration might enable robust execution of robotic construction with the instruction from a high-level planner and is currently under the author’s investigation.

2.6 Implementation

The proposed hierarchical assembly planning framework has been implemented in a proof-of-concept planning tool called Choreo. This tool allows users to compute feasible robotic assembly trajectories using unconstrained target assembly geometry, and it can be configured to work with customized hardware and work environment setup. In this thesis, Choreo is configured to work with spatial extrusion applications. Through the lens of this specific class of robotic assembly problems, this section first presents the general system architecture (section 2.6.1) and then presents an overview
of the user experience of Choreo along each of its computation state (section 2.6.2 - 2.6.7) and shows how this tool can be fit into existing digital design-robotic assembly workflow.

### 2.6.1 System architecture overview

Choreo is implemented in C++ on Robot Operating System Kinetic Release on Linux 16.04 [Quigley et al., 2009]. The C++ code is open-source and available online\(^2\). Drawing inspiration of the Godel system from ROS industrial [ROS-I, 2018b], Choreo’s system architecture is designed to be modularized and flexible: graphical user interface (GUI) module, data IO module, visualization module, and core planning engine modules are all implemented in standalone ROS nodes. Instead of directly communicating to each other, the communication between these modules is coordinated by a central core node, in formatted ROS messages and services (figure 2-9). This enables a clean decoupling between modules that can offer users the flexibility to plug in and experiment with their customized sequence or motion planner without changing the rest of the codebase. The GUI is implemented as a simple Qt plugin for the Rviz visualization platform to provide buttons, sliders, and data IO to help users input/output their data and navigate them through the planning process. The separation of GUI from the core module initiate the possibility to have a web browser-based user interface.

### 2.6.2 Assembly problem setup

Robot and end effector is set up via Unified Robot Description Format (URDF) data\(^3\) in Choreo, which is an XML format data that contains robot’s link geometry, joint limitation and other related data. Although users can specify the robot by creating their own URDF, Choreo takes advantage of the off-the-shelf URDF setups by ROS-Industrial, which covers almost all the major industrial robot’s brands, e.g. KUKA, ABB, and UR. To specify customized end effector, users need to have the STL mesh

---

2 https://github.com/yijiangh/Choreo
3 https://wiki.ros.org/urdf
Figure 2-9: Choreo’s system architecture. The grey components are not implemented at the moment.

for the end effector, and create an URDF file to link the imported geometry (used for collision checking) to some designated joints of the robot. Static collision objects in the work environment can be imported as STL meshes, and linked to the robot URDF.

For spatial trusses, geometry can be specified using in the node-connectivity format described in section 2.2 and the diameter of the truss element is defined by the extruder’s nozzle. Decomposition can be added to the geometry model by simply assigning a layer index to each element. The author developed a simple parser based on Rhino and Grasshopper, to have a visual friendly layer tagging workflow. Other customized decomposition algorithm, for example the constrained graph cut decomposition algorithm in [Huang et al., 2016], can also be used. Support for general discrete geometry, for example volumetric element, is currently under the author’s development. The relative position between the assembly geometry and the robot’s base is specified by a 3D vector and input through the GUI widget.
2.6.3 Sequence planning

Currently, Choreo’s sequence planner is powered by a customized backtracking search engine (section 2.3.2). The kinematics check and collision check between robot and the environment is performed through the ikfast kinematics plugin [Diankov and Kuffner, 2008] and the collision checking interface provided by Moveit! [Sucan and Chitta, 2018]. Though not implemented at the moment, more advanced constraint solver, e.g. Gecode [Gecode, 2018], or TAM system that allows blackbox conditional samplers like STRIPStream [Garrett et al., 2018c], can be configured to export task sequence that comply with Choreo’s assembly sequence specification. This plugin-and-experiment feature offered by Choreo makes it a fantastic testbed for deeper investigation in the assembly sequence planning in the future.

The sequence planner exports an assembly sequence file that contains a sequence of assembly elements, and each assembly element in the sequence is associated with a set of feasible end effector directions. Then the core module will coordinate with the parsing and visualization module to visualize the sequence result with feasible end effector directions displayed, answering the queries sent by the GUI.

2.6.4 Motion planning

The semi-constrained Cartesian planner is implemented based on the Descartes planning package from ROS-Industrial [ROS-I, 2018a, De Maeyer et al., 2017]. The sparse ladder graph and the RRT* algorithm is implemented by the author using Descartes package’s ladder graph data structure. Retraction planner is a direct application of the Descartes package with direction vector sampling.

For transition planner, utilizing the motion planner plugin interface of the Moveit! motion planning framework [Sucan and Chitta, 2018], Choreo currently supports 25 motion planners in total, which includes 23 sampling-based planners from the Open Motion Planning Library [Sucan et al., 2012], CHOMP from Moveit! [Ratliff et al., 2009, Sucan and Chitta, 2018], and STOMP from ROS-industrial [Kalakrishnan et al., 2011, ROS-I, 2018c]. Switching between these planners only requires changing a single
parameters in the system launch file. However, at this moment, Choreo does not support using multiple motion planners at the same time, switching planners requires changing the system launch file and relaunching the system.

### 2.6.5 Simulation

After the motion planning is finished, trajectory is tagged with meta-data associated to the assembly task (section 2.5). The core module coordinates with the simulation module to simulate the chosen assembly task that the user chooses at GUI. A library of the computed trajectory, each tagged with assembly task’s id, is presented to the user. The user can choose to export the trajectories associated with the chosen assembly tasks as a json file.

![Figure 2-10: Screenshot of Choreo at trajectory simulation stage.](image)

### 2.6.6 Post-processing and fine-tuning

After the computed trajectory json file is exported, extra post processing and fine-tuning can be performed in other programming platforms. In all the case studies in
this thesis, a customized C# JSON file parser is implemented in Grasshopper and KUKA | PRC package [Braumann and Brell-Cokcan, 2011] is used to post-process the trajectory into a KUKA Robot Language (KRL) file for execution. Utilizing the tree data structure in Grasshopper, hardware IO command can be easily weaved in the trajectories, such as digital output command to switch on/off the extruder and the cooling air, analog output to control extrusion rate of the extrusion motor, wait command and local path modifications. The synthesis of all of these hardware control command with the trajectory with the right speed parametrization, is crucial for the empirical success of the extrusion, despite the fact that the generated plan itself is feasible in a pure geometrical planning domain. The tuning of all these fabrication-related parameters are carried out back and forth between grasshopper and physical experiments, without having to recompute the trajectory from Choreo.

Note that the exported trajectory can be configured easily to work in other parametric design platform, for example Dynamo, and be adapted to other robotic simulation packages like HAL [Schwartz, 2012] and Alpha [Dritsas, 2015].

2.6.7 Robotic execution

KUKA KR6-R900 is used for all of the case studies presented in this thesis. The URDF data of this robot is from ROS-Industrial\(^4\). A customized extrusion system is designed and assembled by the author, together with collaborators from Archisolution workshop\(^5\). Detailed description of the end effector, extrusion system, and cooling system can be found in [Yu et al., 2016] and the online supplementary materials of [Huang et al., 2016].

\(^4\)https://github.com/ros-industrial/kuka_experimental
\(^5\)http://www.asworkshop.cn/
Chapter 3

Case studies

To illustrate the capabilities of Choreo, this chapter introduces three case studies that utilize Choreo’s power to automatically plan for feasible robotic trajectories for spatial extrusion of complex spatial trusses with non-standard topology. The presented case studies has fundamentally different topologies: 3D Voronoi (section 3.1), Mars habitat design (section 3.2), and topology optimized simply-supported beam (section 3.3). Model-related data is presented in table 3.1. The user-guided decomposition of the three models are show in figure 3-1. Computation statistics on assembly planning and fabrication time are presented in table 3.2. All computational experiments were performed on a Linux virtual machine with 4 processors and 16 GB setup on desktop PC with a quad-core Intel Xeon CPU. These case studies, along with more topology optimized shapes’ assembly planning and fabrication results, have been presented in [Huang et al., 2018b, Huang et al., 2018a].

| Model                  | Node count | Element count | Layer count | Size [mm]     |
|------------------------|------------|---------------|-------------|---------------|
| 3D Voronoi (sec 3.1)   | 148        | 292           | 10          | 200 * 200 * 200 |
| Mars habitat (sec 3.2) | 86         | 214           | 9           | 150 * 150 * 320 |
| Topopt beam (sec 3.3)  | 121        | 271           | 53          | 400 * 100 * 100 |

Table 3.1: Input model information of the case studies. The layer count is the number of layers used in the user-generated decomposition (figure 3-1).
Table 3.2: Computation statistics of the case studies.

| Model                  | Sequence planning time [s] | Extrusion planning time [s] | Transition planning time [s] | Fabrication time [hr] |
|------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------|
| 3D Voronoi (sec 3.1)   | 1346                        | 1200                        | 1286 (RRT*)                 | 3.2                   |
| Mars habitat (sec 3.2) | 1498                        | 1200                        | 918 (RRT*)                  | 3                     |
| Topopt beam (sec 3.3)  | 2170                        | 1271                        | 893 (STOMP)                 | 3.6                   |

3.1 3D Voronoi

The 3D Voronoi design was generated by randomly sampling points within a rectangular solid, and then using the 3D Voronoi component in Grasshopper together with Kangaroo 2\(^1\). A sphere collision algorithm was used to force the elements lengths to have a distribution with lower variance. Figure 3-2 shows the design and fabrication of this case. Because of the Voronoi-generating algorithm, there is low variation in node valence, with most nodes having four elements, a relatively low number, connecting. In this design, elements are well supported during each construction step, and there are few very long elements. However, the long elements at the boundary have

\(^1\)http://kangaroo3d.com/
smaller node valences, and the resulting material warping and sagging can sometimes prevent the robot from locating and connecting to these elements even though the computed trajectory is feasible. In terms of motion planning, the internal topology does not have a trivial layer-based pattern. Thus, it is unintuitive for humans to find a sequence manually, and the Choreo platform proves helpful.

Comparison to Framefab [Huang et al., 2016] [Huang et al., 2016] is the only existing literature that demonstrate the ability to use an industrial robot to spatially print irregular meshes without manually predefined assembly logic. Their printing sequence generation algorithm generates a printing sequence without considering the whole robot, but only the end effector. To solve the transition trajectory between adjacent extrusion steps, they force the robot to reset to a home pose every time, and use a genetic algorithm to optimize the guiding curve for the robot’s end effector to follow, with collision check as a binary fitness function. To compare this heuristic approach with the motion planning module used in Choreo, the same as-
semly sequence generated by Choreo’s sequence planning module was used for both of the motion planner. The heuristic motion planning approach used in [Huang et al., 2016] took 86 hours to finish, with 7 planning failure, while Choreo’s motion planner only takes 1286 seconds with no planning failure. Choreo’s computational power and efficiency can be clearly observed from this comparison.

### 3.2 Mars habitat design

The third case study is a model of a pressurized habitat designed for a human colony on Mars. An outer dome membrane, discretized into a mesh-like structure, is helped structurally by an internal tree structure that acts like a tension spoke system to anchor the membrane to the ground. Figure 3-3 shows the design and fabrication of this case.

![Figure 3-3: Mars habitat design, robotic trajectories with RRT*, and final extruded result.](image)

The internal tree structure needs the outer membrane in order to contain its deformation under gravity within certain tolerance. The construction sequence thus alternates between outer and inner structure to gradually close the membrane at the
Nodes on the stem of the internal tree structure have highest node valence. The outer layer needs to be built before the internal tree elements to support them, but introducing more surrounding collision objects leaves narrow pathways for the robot to enter. This forces the planner to find long trajectories to allow specific joints to have sufficient rotation in open space to approach the desired joint trajectory and “build a tree branch” within the increasingly populated workspace inside the spherical membrane.

### 3.3 Topology optimized simply-supported beam

Using the ground structure topology optimization method described in [Huang et al., 2018a], a simply-supported beam was designed for the loads and boundary conditions shown in figure 3-4 (a), (b). With this approach, it was possible to remove 91% of the material initially included in the ground structure (the precise result is dependent on the stiffness constraint specified by the user). The resulting topology is fairly irregular; compared to a standardized mesh topology, this has the potential to be much more efficient both in terms of material use and fabrication time. The average element length is long, and element length variation is low because the design is generated from a regular base mesh. However, the geometric configuration generated from these elements is not trivial. The trajectory highlighted in figure 3-4 shows the corresponding tool center point traveling trajectory from the transition planning result, indicating that the robot’s configuration changes significantly between many pairs of adjacent extrusions, requiring the planner to output a long and unintuitive trajectory to stay within joint limitations and stay clear from collisions.

Even though the base mesh is a regular 3D grid with all diagonals connected in each unit cell, the output from topology optimization does not necessarily have a trivial assembly sequence. The high values of node valence combined with long elements tends to create very narrow pathways, slowing down the transition planning and producing long transition trajectory to adapt to the drastic joint change between adjacent extrusion processes.
Figure 3-4: Topology optimized simply-supported beam, with (a-b) topology optimization input and result, (c) robot trajectories with STOMP, and (d) final extruded result.
Chapter 4

Conclusions

It has been nearly a decade since the modern investigation of architectural robotic assembly started at Gramazio & Kohler research at ETH-Zurich [Gramazio et al., 2014]. However, despite the compelling new possibilities demonstrated by various promising architectural case studies and prototypical structures, the level of automation in this digital design-assembly workflow is still comparably low. The slow and convoluted workflow by manually planning for the robots deviates from the desire for a versatile and adaptable design-assembly workflow. The technical challenge in the assembly sequence planning and programming of the robots distracts architectural designers from focusing on the important aspects of their design, restricts their flexibility, and prevents them from fully realizing innovative architectural structures.

This thesis presents the first attempt in the field to rigorously formulate the architectural robotic assembly planning problems, provide an integrated algorithmic solution, and presents an proof-of-concept software system implementation. Three case studies are presented to demonstrate the proposed assembly planning framework’s power on enabling spatial extrusion of complex spatial trusses with non-standard topologies, which hasn’t been shown possible before.

This chapter summarizes the key contributions presented in this thesis, identifies the potential impacts, and proposes outlooks for future work.
4.1 Summary of contributions

The main theoretical contributions of this thesis includes:

- A new hierarchical assembly planning algorithm is proposed to solve the assembly planning that has long planning horizon and three-dimensional geometric complexity, which hasn’t been shown solvable before. The introduction of sequence planning layer and motion planning layer uses simplified feasibility checking in the abstract-level search to prune geometric variables’ domains on the plan skeleton, while avoiding explicit motion planning call.

- A constraint satisfaction problem (CSP) formulation of the assembly sequence planning problem that takes the robotic constraints into consideration is introduced. A backtracking search with dynamic variable ordering and 1-level forward checking is proposed as a baseline planner to solve the problem. (Section 2.3)

- A RAM-safe, asymptotically optimal semi-constrained Cartesian planner that is capable of solving large problem instance is proposed. A sparse ladder graph representation is proposed to abstract away details on the explicit planning graph, and a sampling-based optimization framework that operates on the sparse ladder graph to localize the optimal search region for extracting joint trajectory solutions using DAG shortest path search. (Section 2.4.1)

From engineering perspective, the implementation of Choreo embodies the following contributions:

- A modularized software system that allows users to implement and plug in customized sequence or motion planners to adapt the system to specific application and hardware without changing other modules in the system. (Section 2.6.3, 2.6.4)

- A hardware- and application-agnostic planning system that can be easily configured to work with industrial robots across brands, sizes and degrees of freedoms.
The planning system can also work with any customized end effector. (Section 2.3)

- A post processing system that tags the generated trajectories with assembly process meta-data, to maximize the flexibility in hardware IO command synthesis and micro-path modification on various programming platforms. (Section 2.6.6)

4.2 Potential impact

The case studies presented in this thesis has demonstrated this new assembly planning framework’s power in automatically generating feasible robotic instructions and how its integration into existing digital design workflow can resurrect topology as a fundamental design variable on designers’ palette for robotic assembly. The emergence of this automated planning system can provide a better way for designers to interact with robots, shifting the machine programming experience back to high-level tasks in the architectural language of shape and topology.

On the other hand, the open-source nature, the flexibility, and the efficiency of Choreo creates an inviting and encouraging testbed for educators, researchers, and practitioners to explore novel robotic fabrication and assembly applications more boldly, with the support by this enabling technology. It provides a general common playground for future research in assembly-related sequence planner and motion planner, and creates a bridge between architectural robotics research community and task and motion planning research community.

4.3 Limitations and future work

Extension to other robotic assembly applications and other robot types
All the algorithmic descriptions and case studies presented in this thesis is performed in the context of the specific application of robotic spatial extrusion. However, generalizing the proposed assembly planning framework to other assembly applications, for
example spatial positioning, requires very little modification on the algorithm itself. These modifications includes different predefined plan skeleton, different constraints on end effector’s orientation in the semi-constrained Cartesian planner, and some extensions on the assembly sequence data format. This generalization is currently under development by the author.

**Automatic decomposition method for sequence planner** The only human intervention in the entire assembly planning system is the shape decomposition to inform and accelerate the sequence planner. An automatic decomposition algorithm will eliminate this last bit of human intervention and fully automate the planning process.

**Backtrack between planning layers** Currently, when the planning system encounters a planning failure in any layer in the hierarchy, there is no backtracking mechanism provided to allow it to go back one level higher and try another branch to search. Thus, the hierarchical planning algorithm is not complete. Existing work in TAMP has devised various way to allow this geometric backtracking across planning layers. The integration of some of this research is left as future work.

4.4 Concluding remark

Although Choreo is still at its infant stage, its flexibility and speed has already suggested an exciting future possibility: fabrication and assembly logic related to robotic constructibility could be integrated as a driver in iterative conceptual design, pushing the role of technical assessment from checking a nearly finalized design to an early-stage design aid. This calls for a even more holistic investigation into the mutual constraining relationship between architectural geometry and robotics, and the ultimate proposal of construction-driven geometry guidance, where the planning system not only plans, but also diagnoses planning failure, explains causes for failure, and most importantly, provides suggestion for resolution.
Bibliography

[Agrawala et al., 2003] Agrawala, M., Phan, D., Heiser, J., Haymaker, J., Klingner, J., Hanrahan, P., and Tversky, B. (2003). Designing effective step-by-step assembly instructions. In ACM Transactions on Graphics (TOG), volume 22, pages 828–837. ACM.

[Alami et al., 1994] Alami, R., Laumond, J.-P., and Siméon, T. (1994). Two manipulation planning algorithms. In WAFR Proceedings of the workshop on Algorithmic foundations of robotics, pages 109–125. A. K. Peters, Ltd. Natick, MA, USA.

[Alami et al., 1990] Alami, R., Siméon, T., and Laumond, J.-P. (1990). A geometrical approach to planning manipulation tasks. the case of discrete placements and grasps. In The fifth international symposium on Robotics research, pages 453–463. MIT Press.

[Bahubalendruni and Biswal, 2016] Bahubalendruni, M. R. and Biswal, B. B. (2016). A review on assembly sequence generation and its automation. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 230(5):824–838.

[Barry et al., 2013] Barry, J., Pack, L., and Lozano-pérez, K. T. (2013). A hierarchical approach to manipulation with diverse actions. In 2013 IEEE International Conference on Robotics and Automation, pages 1799–1806.

[Baudisch and Mueller, 2017] Baudisch, P. and Mueller, S. (2017). Personal fabrication. Foundations and Trends in Human-Computer Interaction, 10(3-4):165–293.

[Berg et al., 2009] Berg, J. v. d., Stilman, M., Kuffner, J., Lin, M., and Manocha, D. (2009). Path planning among movable obstacles: A probabilistically complete approach. In Algorithmic Foundation of Robotics VIII, Springer Tracts in Advanced Robotics, pages 599–614. Springer, Berlin, Heidelberg.

[Bourgault, 1984] Bourjault, A. (1984). Contribution à une approche méthodologique de l’assemblage automatisé: élaboration automatique des séquences opératoires. Thèse d’Etat, Université de Franche-Comté.

[BranchTech, 2018] BranchTech (2018). Branch Technology. https://www.branch.technology/. Last accessed: 2018-03-14.
[Braumann and Brell-Cokcan, 2011] Braumann, J. and Brell-Cokcan, S. (2011). Parametric robot control: integrated cad/cam for architectural design. In Proceedings of the 31st Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA).

[Cambon et al., 2009] Cambon, S., Alami, R., and Gravot, F. (2009). A hybrid approach to intricate motion, manipulation and task planning. The International Journal of Robotics Research, 28(1):104–126.

[Cormen, 2009] Cormen, T. H. (2009). Introduction to algorithms. MIT press.

[Cortsen et al., 2012] Cortsen, J., Jørgensen, J. A., Solvason, D., and Petersen, H. G. (2012). Simulating robot handling of large scale deformable objects: Manufacturing of unique concrete reinforcement structures. In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 3771–3776. IEEE.

[Dai et al., 2018] Dai, C., Wang, C. C., Wu, C., Lefebvre, S., Fang, G., and Liu, Y. (2018). Support-free volume printing by multi-axis motion. ACM Transactions on Graphics (TOG).

[De Fazio and Whitney, 1987] De Fazio, T. and Whitney, D. (1987). Simplified generation of all mechanical assembly sequences. IEEE Journal on Robotics and Automation, 3(6):640–658.

[De Maey et al., 2017] De Maeyer, J., Moyaers, B., and Demeester, E. (2017). Cartesian path planning for welding robots: Evaluation of the descartes algorithm. In Proceedings of the 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation.

[De Mello and Sanderson, 1990] De Mello, L. H. and Sanderson, A. C. (1990). And/or graph representation of assembly plans. IEEE Transactions on robotics and automation, 6(2):188–199.

[De Mello and Sanderson, 1991] De Mello, L. H. and Sanderson, A. C. (1991). A correct and complete algorithm for the generation of mechanical assembly sequences. IEEE transactions on Robotics and Automation, 7(2):228–240.

[de Silva et al., 2013] de Silva, L., Pandey, A. K., Gharbi, M., and Alami, R. (2013). Towards combining htn planning and geometric task planning. arXiv:1307.1482 [cs]. arXiv: 1307.1482.

[Dechter, 2003] Dechter, R. (2003). Constraint processing. Morgan Kaufmann.

[Deuss et al., 2014] Deuss, M., Panozzo, D., Whiting, E., Liu, Y., Block, P., Sorkine-Hornung, O., and Pauly, M. (2014). Assembling self-supporting structures. ACM Transactions on Graphics (TOG), 33(6):214.
[Diankov and Kuffner, 2008] Diankov, R. and Kuffner, J. (2008). Openrave: A planning architecture for autonomous robotics. Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34, 79.

[Dogar et al., 2015] Dogar, M., Spielberg, A., Baker, S., and Rus, D. (2015). Multi-robot grasp planning for sequential assembly operations. In Robotics and Automation (ICRA), 2015 IEEE International Conference on, pages 193–200. IEEE.

[Dornhege et al., 2012] Dornhege, C., Eyerich, P., Keller, T., Trüg, S., Brenner, M., and Nebel, B. (2012). Semantic attachments for domain-independent planning systems. In Towards service robots for everyday environments, pages 99–115. Springer.

[Dritsas, 2015] Dritsas, S. (2015). A digital design and fabrication library. In Proceedings of the Symposium on Simulation for Architecture & Urban Design, pages 75–80. Society for Computer Simulation International.

[Erdem et al., 2011] Erdem, E., Haspalamutgil, K., Palaz, C., Patoglu, V., and Uras, T. (2011). Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation. In 2011 IEEE International Conference on Robotics and Automation, pages 4575–4581.

[Erol et al., 1994] Erol, K., Hendler, J., and Nau, D. S. (1994). Htn planning: Complexity and expressivity. In Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 2), AAAI’94, pages 1123–1128. American Association for Artificial Intelligence.

[Eversmann et al., 2017] Eversmann, P., Gramazio, F., and Kohler, M. (2017). Robotic prefabrication of timber structures: towards automated large-scale spatial assembly. Construction Robotics, 1(1-4):49–60.

[Fikes and Nilsson, 1971] Fikes, R. E. and Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. Artificial Intelligence, 2(3):189–208.

[Fu et al., 2015] Fu, C.-W., Song, P., Yan, X., Yang, L. W., Jayaraman, P. K., and Cohen-Or, D. (2015). Computational interlocking furniture assembly. ACM Transactions on Graphics (TOG), 34(4):91.

[Garrett et al., 2015] Garrett, C. R., Lozano-Pérez, T., and Kaelbling, L. P. (2015). Backward-forward search for manipulation planning. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 6366–6373.

[Garrett et al., 2018a] Garrett, C. R., Lozano-Pérez, T., and Kaelbling, L. P. (2018a). Ffrob: Leveraging symbolic planning for efficient task and motion planning. The International Journal of Robotics Research, 37(1):104–136.

[Garrett et al., 2018b] Garrett, C. R., Lozano-Pérez, T., and Kaelbling, L. P. (2018b). Sampling-based methods for factored task and motion planning. arXiv preprint arXiv:1801.00680.

73
[Garrett et al., 2018c] Garrett, C. R., Lozano-Pérez, T., and Kaelbling, L. P. (2018c). Stripstream: Integrating symbolic planners and blackbox samplers. arXiv:1802.08705 [cs]. arXiv: 1802.08705.

[Gecode, 2018] Gecode (2018). Gecode - a generic constraint development environment. http://www.gecode.org/. Last accessed: 2018-05-25.

[Gelber et al., 2018a] Gelber, M. K., Hurst, G., and Bhargava, R. (2018a). Freeform assembly planning. arXiv:1801.00527 [cs]. arXiv: 1801.00527.

[Gelber et al., 2018b] Gelber, M. K., Hurst, G., Comi, T. J., and Bhargava, R. (2018b). Model-guided design and characterization of a high-precision 3d printing process for carbohydrate glass. Additive Manufacturing, 22:38–50.

[Ghandi and Masehian, 2015] Ghandi, S. and Masehian, E. (2015). Review and taxonomies of assembly and disassembly path planning problems and approaches. Computer-Aided Design, 67:58–86.

[Giftthaler et al., 2017a] Giftthaler, M., Neunert, M., StÄduble, M., Frigerio, M., Semini, C., and Buchli, J. (2017a). Automatic differentiation of rigid body dynamics for optimal control and estimation. Advanced Robotics, 31(22):1225–1237. arXiv: 1709.03799.

[Giftthaler et al., 2017b] Giftthaler, M., Sandy, T., Dörfler, K., Brooks, I., Buckingham, M., Rey, G., Kohler, M., Gramazio, F., and Buchli, J. (2017b). Mobile robotic fabrication at 1: 1 scale: the in situ fabricator. Construction Robotics, 1(1-4):3–14.

[Goldwasser et al., 1996] Goldwasser, M., Latombe, J.-C., and Motwani, R. (1996). Complexity measures for assembly sequences. In Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on, volume 2, pages 1851–1857. IEEE.

[Gramazio et al., 2014] Gramazio, F., Matthias, K., and Willmann, J. (2014). The robotic touch. Park Books.

[Guibas et al., 1995] Guibas, L. J., Halperin, D., Hirukawa, H., Latombe, J.-C., and Wilson, R. H. (1995). A simple and efficient procedure for polyhedral assembly partitioning under infinitesimal motions. In Proceedings of 1995 IEEE International Conference on Robotics and Automation, volume 3, pages 2553–2560 vol.3.

[Hack and Lauer, 2014] Hack, N. and Lauer, W. V. (2014). Mesh-mould: Robotically fabricated spatial meshes as reinforced concrete formwork. Architectural Design, 84(3):44–53.

[Hauser, 2012] Hauser, K. (2012). The minimum constraint removal problem with three robotics applications. In Workshop on the Algorithmic Foundations of Robotics (WAFR).
Hauser, K. (2013). Minimum constraint displacement motion planning. In *Robotics: Science and Systems (RSS)*.

Hauser, K. and Latombe, J.-C. (2010). Multi-modal motion planning in non-expansive spaces. *The International Journal of Robotics Research*, 29(7):897–915.

Hauser, K. and Ng-Thow-Hing, V. (2011). Randomized multi-modal motion planning for a humanoid robot manipulation task. *The International Journal of Robotics Research*, 30(6):678–698.

Heger, F. W. (2010). *Assembly planning in constrained environments: Building structures with multiple mobile robots*. PhD thesis, Carnegie Mellon University.

Helm, V., Willmann, J., Thoma, A., Piskorec, L., Hack, N., Gramazio, F., and Kohler, M. (2015). Iridescence print: Robotically printed lightweight mesh structures. *3D Printing and Additive Manufacturing*, 2(3):117–122.

Homem de Mello, L. S. and Sanderson, A. C. (1988). Automatic generation of mechanical assembly sequences. Technical report, Robotics Institute, Carnegie-Mellon University, PA.

Huang, Y., Carstensen, J., and Mueller, C. (2018a). 3d truss topology optimization for automated robotic spatial extrusion. In *Proceedings of the International Association for Shell and Spatial Structures (IASS) Symposium 2018*.

Huang, Y., Carstensen, J., Tessmer, L., and Mueller, C. (2018b). Robotic extrusion of architectural structures with nonstandard topology. In *Robotic Fabrication in Architecture, Art and Design 2018*. Springer.

Huang, Y., Zhang, J., Hu, X., Song, G., Liu, Z., Yu, L., and Liu, L. (2016). Framefab: Robotic fabrication of frame shapes. *ACM Transactions on Graphics (TOG)*, 35(6):224.

Jeffers, M. (2016). Autonomous robotic assembly with variable material properties. In *Robotic Fabrication in Architecture, Art and Design 2016*, pages 48–61. Springer.

Jiménez, P. (2013). Survey on assembly sequencing: a combinatorial and geometrical perspective. *Journal of Intelligent Manufacturing*, 24(2):235–250.

Kaelbling and Lozano-Pérez, T. (2011). Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, pages 1470–1477.
[Kalakrishnan et al., 2011] Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., and Schaal, S. (2011). Stomp: Stochastic trajectory optimization for motion planning. In 2011 IEEE International Conference on Robotics and Automation, pages 4569–4574.

[Karaman and Frazzoli, 2011] Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. arXiv:1105.1186 [cs]. 01241 arXiv: 1105.1186.

[Kavraki and Kolountzakis, 1995] Kavraki, L. E. and Kolountzakis, M. N. (1995). Partitioning a planar assembly into two connected parts is np-complete. Information Processing Letters, 55(3):159–165.

[Kavraki et al., 1996] Kavraki, L. E., Svestka, P., Latombe, J. C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation, 12(4):566–580.

[Komendera, 2014] Komendera, E. (2014). Precise assembly of truss structures by distributed robots. PhD thesis, University of Colorado at Boulder.

[Krontiris and Bekris, 2015] Krontiris, A. and Bekris, K. (2015). Dealing with difficult instances of object rearrangement. In Robotics: Science and Systems (RSS).

[Krontiris and Bekris, 2016] Krontiris, A. and Bekris, K. E. (2016). Efficiently solving general rearrangement tasks: A fast extension primitive for an incremental sampling-based planner. In International Conference on Robotics and Automation (ICRA), pages 3924–3931. IEEE.

[Lagreiffoul, 2016] Lagriffoul, F. (2016). On benchmarks for combined task and motion planning. In RSS 2016 Workshop on Task and Motion Planning.

[Lagreiffoul and Andres, 2016] Lagriffoul, F. and Andres, B. (2016). Combining task and motion planning: A culprit detection problem. The International Journal of Robotics Research, 35(8):890–927.

[Lagreiffoul et al., 2014] Lagriffoul, F., Dimitrov, D., Bidot, J., Saffiotti, A., and Karlsson, L. (2014). Efficiently combining task and motion planning using geometric constraints. The International Journal of Robotics Research, 33(14):1726–1747.

[LaValle, 2006] LaValle, S. M. (2006). Planning algorithms. Cambridge university press.

[Lifschitz, 2008] Lifschitz, V. (2008). What is answer set programming? In Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3, AAAI’08, pages 1594–1597. AAAI Press.

[Livesley, 1992] Livesley, R. K. (1992). A computational model for the limit analysis of three-dimensional masonry structures. Meccanica, 27(3):161–172. 0014.
[Lo et al., 2009] Lo, K.-Y., Fu, C.-W., and Li, H. (2009). 3d polyomino puzzle. In ACM Transactions on Graphics (TOG), volume 28, page 157. ACM.

[Lozano-Pérez, 1981] Lozano-Pérez, T. (1981). Automatic planning of manipulator transfer movements. IEEE Transactions on Systems, Man, and Cybernetics, 11(10):681–698. 01067.

[Lozano-Pérez, 1983] Lozano-Pérez, T. (1983). Spatial planning: A configuration space approach. IEEE transactions on computers, (2):108–120.

[Lozano-Pérez et al., 1987] Lozano-Pérez, T., Jones, J., Mazer, E., O’Donnell, P., Grimson, W., Tournassoud, P., and Lanusse, A. (1987). Handey: A robot system that recognizes, plans, and manipulates. In 1987 IEEE International Conference on Robotics and Automation Proceedings, volume 4, pages 843–849.

[Lozano-Pérez and Kaelbling, 2014] Lozano-Pérez, T. and Kaelbling, L. P. (2014). A constraint-based method for solving sequential manipulation planning problems. In Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, pages 3684–3691. IEEE.

[McDermott et al., 1998] McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). Pddl-the planning domain definition language. Technical report, Yale Center for Computational Vision and Control.

[McEvoy et al., 2014] McEvoy, M., Komendera, E., and Correll, N. (2014). Assembly path planning for stable robotic construction. In Technologies for Practical Robot Applications (TePRA), 2014 IEEE International Conference on, pages 1–6. IEEE.

[Mueller et al., 2014] Mueller, S., Im, S., Gurevich, S., Teibrich, A., Pfisterer, L., Guimbretière, F., and Baudisch, P. (2014). Wireprint: 3d printed previews for fast prototyping. In Proceedings of the 27th annual ACM symposium on User interface software and technology, pages 273–280. ACM.

[Natarajan, 1988] Natarajan, B. K. (1988). On planning assemblies. In Proceedings of the fourth annual symposium on Computational geometry, pages 299–308. ACM.

[Pandey et al., 2012] Pandey, A. K., Saut, J. P., Sidobre, D., and Alami, R. (2012). Towards planning human-robot interactive manipulation tasks: Task dependent and human oriented autonomous selection of grasp and placement. In 2012 4th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob), pages 1371–1376.

[Parascho et al., 2017] Parascho, S., Gandia, A., Mirjan, A., Gramazio, F., and Kohler, M. (2017). Cooperative fabrication of spatial metal structures. In Fabricate 2017, pages 24–29. UCL Press.
[Peng et al., 2018] Peng, H., Briggs, J., Wang, C.-Y., Guo, K., Kider, J., Mueller, S., Baudisch, P., and Guimbretière, F. (2018). Roma: Interactive fabrication with augmented reality and a robotic 3d printer. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, pages 579:1–579:12. ACM.

[Peng et al., 2016] Peng, H., Wu, R., Marschner, S., and Guimbretière, F. (2016). On-the-fly print: Incremental printing while modelling. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, pages 887–896. ACM.

[Plaku and Hager, 2010] Plaku, E. and Hager, G. D. (2010). Sampling-based motion and symbolic action planning with geometric and differential constraints. In 2010 IEEE International Conference on Robotics and Automation, pages 5002–5008.

[Quigley et al., 2009] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In ICRA workshop on open source software, volume 3, page 5. Kobe, Japan.

[Ratliff et al., 2009] Ratliff, N., Zucker, M., Bagnell, J. A., and Srinivasa, S. (2009). Chomp: Gradient optimization techniques for efficient motion planning. In International Conference on Robotics and Automation, pages 489–494. IEEE.

[Röhrdanz et al., 1997] Röhrdanz, F., Mosemann, H., and Wahl, F. M. (1997). High-lap: A high level system for generating, representing, and evaluating assembly sequences. International Journal on Artificial Intelligence Tools, 6(02):149–163.

[ROS-I, 2018a] ROS-I (2018a). ROS Industrial - Descartes. https://github.com/ros-industrial-consortium/descartes. Last accessed: 2018-03-14.

[ROS-I, 2018b] ROS-I (2018b). ROS Industrial - Godel. https://github.com/ros-industrial-consortium/godel. Last accessed: 2018-03-14.

[ROS-I, 2018c] ROS-I (2018c). ROS Industrial - Industrial Moveit. https://github.com/ros-industrial/industrial_moveit. Last accessed: 2018-03-14.

[Sandy and Buchli, 2018] Sandy, T. and Buchli, J. (2018). Object-based visual-inertial tracking for additive fabrication. IEEE Robotics and Automation Letters, 3(3):1370–1377.

[Schulman et al., 2014] Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., and Abbeel, P. (2014). Motion planning with sequential convex optimization and convex collision checking. The International Journal of Robotics Research, 33(9):1251–1270.

[Schwartz, 2012] Schwartz, T. (2012). Hal. In Robotic Fabrication in Architecture, Art and Design 2012, pages 92–101. Springer.

[Schwartzburg and Pauly, 2013] Schwartzburg, Y. and Pauly, M. (2013). Fabrication-aware design with intersecting planar pieces. In Computer Graphics Forum, volume 32, pages 317–326. Wiley Online Library.
[Siciliano et al., 2009] Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2009). *Kinematics*. Springer.

[Sondergaard et al., 2016] Sondergaard, A., Amir, O., Eversmann, P., Piskorec, L., Stan, F., Gramazio, F., and Kohler, M. (2016). Topology optimization and robotic fabrication of advanced timber space-frame structures. In *Robotic Fabrication in Architecture, Art and Design 2016*, pages 190–203. Springer.

[Song et al., 2012] Song, P., Fu, C.-W., and Cohen-Or, D. (2012). Recursive interlocking puzzles. *ACM Transactions on Graphics (TOG)*, 31(6):128.

[Srivastava et al., 2014] Srivastava, S., Fang, E., Riano, L., Chitnis, R., Russell, S., and Abbeel, P. (2014). Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 639–646.

[Stilman and Kuffner, 2008] Stilman, M. and Kuffner, J. J. (2008). Planning among movable obstacles with artificial constraints. *The International Journal of Robotics Research*, 27(11-12):1295–1307.

[Stilman et al., 2007] Stilman, M., Schamburek, J. U., Kuffner, J., and Asfour, T. (2007). Manipulation planning among movable obstacles. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3327–3332.

[Sucan and Chitta, 2018] Sucan, I. A. and Chitta, S. (2018). Moveit! http://moveit.ros.org. Last accessed: 2018-03-14.

[Sucan et al., 2012] Sucan, I. A., Moll, M., and Kavraki, L. E. (2012). The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82. http://ompl.kavrakilab.org.

[T. Dantam et al., 2016] T. Dantam, N., Kingston, Z., Chaudhuri, S., and E. Kavraki, L. (2016). Incremental task and motion planning: A constraint-based approach. In *Robotics: Science and Systems (RSS)*.

[Tai, 2012] Tai, A. S.-C. (2012). Design for assembly: A computational approach to construct interlocking wooden frames. Master’s thesis, Massachusetts Institute of Technology.

[Tam et al., 2018] Tam, K.-M., Marshall, D. J., Gu, M., Kim, J., Huang, Y., Lavallee, J. A., and Mueller, C. T. (2018). Fabrication-aware structural optimisation of lattice additive-manufactured with robot-arm. *International Journal of Rapid Manufacturing*.

[Toussaint, 2015] Toussaint, M. (2015). Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI’15, pages 1930–1936. AAAI Press.
[Toussaint and Lopes, 2017] Toussaint, M. and Lopes, M. (2017). Multi-bound tree search for logic-geometric programming in cooperative manipulation domains. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4044–4051.

[UNESCO, 2005] UNESCO (2005). Press release ece/stat/05/p03, geneva, world robotics survey (october 11, 2005).

[wikipedia KRL, 2018] Wikipedia KRL (2018). Kuka robot language - wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/KUKA_Robot_Language. Last accessed: 2018-04-16.

[Wilfong, 1991] Wilfong, G. (1991). Motion planning in the presence of movable obstacles. *Annals of Mathematics and Artificial Intelligence*, 3(1):131–150.

[Wilson, 1992] Wilson, R. H. (1992). *On Geometric Assembly Planning*. PhD thesis, Stanford University.

[Woltery, 1989] Woltery, J. D. (1989). On the automatic generation of assembly plans. In *Proceedings, 1989 International Conference on Robotics and Automation*, pages 62–68.

[Wu et al., 2016] Wu, R., Peng, H., Guimbretière, F., and Marschner, S. (2016). Printing arbitrary meshes with a 5dof wireframe printer. *ACM Transactions on Graphics (TOG)*, 35(4):101.

[Xin et al., 2011] Xin, S., Lai, C.-F., Fu, C.-W., Wong, T.-T., He, Y., and Cohen-Or, D. (2011). Making burr puzzles from 3d models. In *ACM Transactions on Graphics (TOG)*, volume 30, page 97. ACM.

[Yu et al., 2016] Yu, L., Huang, Y., Liu, Z., Xiao, S., Liu, L., Song, G., and Wang, Y. (2016). Highly informed robotic 3d printed polygon mesh: A novel strategy of 3d spatial printing. In *Proceedings of the 36st Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*, pages 298–307.

[Yun, 2010] Yun, S.-k. (2010). *Coordinating construction by a distributed multi-robot system*. PhD thesis, Massachusetts Institute of Technology.