The new analog: A protocol for linking design and construction intent with algorithmic planning for robotic assembly of complex structures

Yijiang Huang
yijiangh@mit.edu
Massachusetts Institute of Technology
Cambridge, USA

Pok Yin Victor Leung
leung@arch.ethz.ch
ETH Zurich
Zurich, Switzerland

Caelan Garrett
caelan@csail.mit.edu
Massachusetts Institute of Technology
Cambridge, USA

Fabio Gramazio
gramazio@arch.ethz.ch
ETH Zurich
Zurich, Switzerland

Matthias Kohler
kohler@arch.ethz.ch
ETH Zurich
Zurich, Switzerland

Caitlin Mueller
caitlinm@mit.edu
Massachusetts Institute of Technology
Cambridge, USA

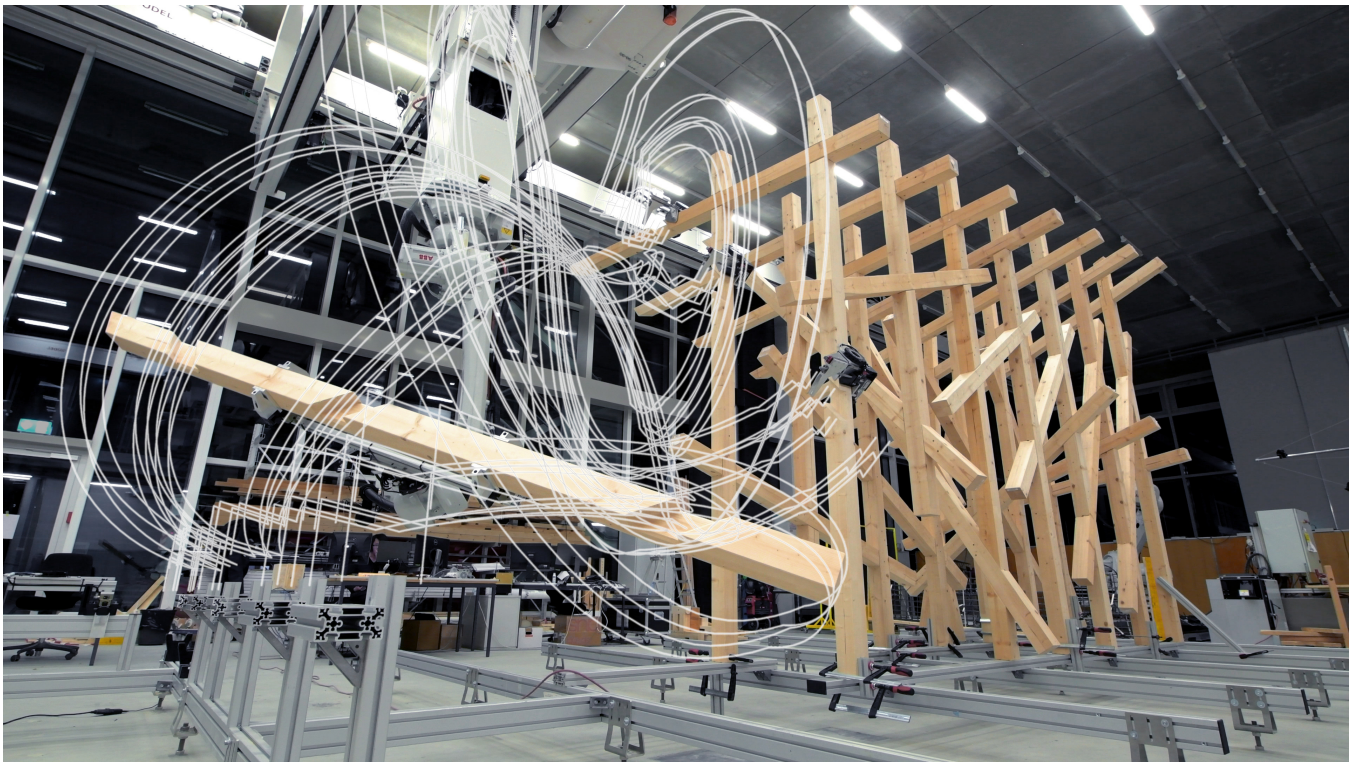


Figure 1: Image showing the final timber element being assembled in our case study. Three distributed clamps can be seen in the background already attached to the structure by the robotic arm. The assembly process is modeled with our flowchart and solved using our solver. Two linear and one free motion trajectory are used to bring the element from pickup to the clamps, shown here as overlaid white curves.

SCF '21, October 28–29, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-9090-3/21/10.
https://doi.org/10.1145/3485114.3485122

ABSTRACT

Construction robotics are increasingly popular in the architectural fabrication community due to their accuracy and flexibility. Because of their high degree of motion freedom, these tools are able to assemble complex structures with irregular designs, which advances architectural aesthetics and structural performance. However, automated task and motion planning (TAMP) for a robot to

assemble non-repetitive objects can be challenging due to (1) a non-repetitive assembly pattern (2) the need for a continuous robotic motion throughout a sequence of movement (3) a congested construction scene and (4) occasional robot configuration constraints due to taught positions. Recent work has already begun to address these challenges for repetitive assembly processes, where the robot repeats a pattern of primitive behaviors (e.g. brick stacking or spatial extrusion). Yet, there are many assembly processes that can benefit from a non-repetitive pattern. For example, processes can change tools on an element-by-element level to accommodate a wider range of geometry.

Our work is motivated by the necessity of robotic modeling and planning for a recently published timber assembly process which utilizes distributed robotic clamps to press together interlocking joints. In addition to pick-and-place operations, the robot needs to move numerous tools within the construction scene, similar to a tool-change operation. In order to facilitate an agile process for architectural design, construction process design, and TAMP, we introduce a flowchart-based specification language which allows various designers to describe their design and construction intent and knowledge. A compiler can then translate the assembly description, sequence, process flowchart, and robotic setup into a plan skeleton. Additionally, we present a linear and a non-linear solving algorithm that can solve the plan skeleton for a full sequence of robot motions. This algorithm can be customized to take into account designer intuition, which can speed up the planning process. We provide a comparison of the two algorithms using the timber assembly process as our case study. We validate our results by robotically executing and constructing a large-scale real-world timber structure. Finally, we demonstrate the flexibility of our flowchart by showing how custom assembly actions are modeled in our case study. We also demonstrate how other recently published robotic assembly processes can be formulated using our flowcharts to demonstrate generalizability.

CCS CONCEPTS

• **Applied computing** → **Computer-aided design**; **Computer-aided manufacturing**.

KEYWORDS

Digital Fabrication, Robotic Assembly, Task and Motion Planning, Spatial Timber Structure, Distributed Robotic Tools

ACM Reference Format:

Yijiang Huang, Pok Yin Victor Leung, Caelan Garrett, Fabio Gramazio, Matthias Kohler, and Caitlin Mueller. 2021. The new analog: A protocol for linking design and construction intent with algorithmic planning for robotic assembly of complex structures . In *Symposium on Computational Fabrication (SCF '21), October 28–29, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3485114.3485122

1 INTRODUCTION

An emerging trend in architectural construction is to use industrial robotic arms for discrete-element assembly tasks. This approach is used for different material systems, such as timber, steel, masonry, fiber, plastic, and can be applied on different scales, ranging anywhere from interior finishes to structural elements. These projects take advantage of the precision and accuracy of robot to create *non-repetitive* (i.e. irregular) assemblies that have complex architectural expression or high structural efficiency, which are difficult to assemble manually [Gramazio et al. 2014].

Non-Repetitive Robotic Assembly. Programming robots to build non-repetitive assemblies requires a very different paradigm from what is typically considered in industrial robotics, where the robot performs repetitive tasks in a production line and programmers can manually program and optimize a routine (often referred to as teaching). In the context of creating non-repetitive architectural assemblies, it is desirable to have general-purpose and flexible robots that can quickly adapt to evolving designs and construction concepts. In a high variation assembly, where the assembly movements do not have similar pick-and-place patterns or where the workpiece varies substantially in geometry, the time required to manually program or dry-run each individual motion can often be longer than the execution process itself. Still, it is important to validate that all the robotic motions are feasible during the design stage of an architectural scheme.

In this paper we focus on the most difficult and crucial component of validation: ensuring robot reachability and collision-free motion. Unlike a Cartesian robot (such as a 3D printer or a laser cutter), most robotic arms have six or more degrees of freedom (DOF) and their movements cannot be easily validated without performing full robotic motion planning for all of the involved steps. In fact, validation planning is so detailed such that, upon completion, the robotic execution trajectories are produced as a by-product. This nontrivial process requires formulating the assembly process as a planning scene, which defines the involved robots, tools, constraints and collision objects, and developing planning algorithms, often referred to as *planners*.

Task and Motion Planning. Within the robotics planning literature, there are predominantly two types of planners: (1) *task planners* plan discrete decisions such as the order in which to perform various types of robot and tool motions and (2) *motion planners* plan a trajectory for a single robotic motion. Recently, researchers in architecture-scale digital fabrication have started to use planners to generate instructions for robotic assembly. However, the currently available motion planning tools for the architectural community (such as compas_fab [Rust et al. 2018], moveit! [Sucan and Chitta 2018]) are only able to plan trajectories connecting two configurations. In order to apply these tools to *multi-step manipulation*, which involves several movements (e.g. picking up an object, transferring it to a different location, and inserting the object into a hole), the user needs to perform many motion planning calls, while ensuring the overlapping configurations are the same.

In the robotics literature, each movement is referred to as an *action template* [Huang et al. 2021]. An action template contains three types of parameters, which can be assigned manually or automatically: (1) discrete parameters (e.g. which object to pick or place), (2) continuous parameters (such as poses and grasps of the movable objects), and (3) continuous motion paths. Chaining multiple action templates together results in a *plan skeleton* — a high-level description of a multi-step manipulation process [Garrett et al. 2021].

Repetitive vs Flexible Plan Skeletons. Robotic processes such as material extrusion or pick-and-place assembly sometimes have repetitive plan skeletons. For example, in spatial extrusion, the robot alternates between two actions: (1) extrude and (2) transit. For pick-and-place assembly, the robot repetitively performs four actions in order: (1) transit, (2) pick, (3) transfer, and (4) place. In these situations, both a construction sequence and corresponding robotic motions can be planned together automatically [Huang et al. 2021].

However, many architectural assembly processes require more flexible plan skeletons. Some examples of this include when a tool-change is needed to accommodate different workpiece geometry or if the assembly actions change depending on a variable propriety of the workpiece. In these cases, the designer traditionally must manually create case-specific planning software in order to bridge the gap between this custom behaviour and the standard motion planners.

In our view, this approach wastes human time and expertise. Process designers bring high-level intentionality and knowledge about fabrication and assembly processes that should be communicated to planning systems without the need for low-level or case-specific programming. An alternative approach could take inspiration from other complex planning problems in robotics. In general, planning for non-repetitive plan skeletons that involve both high-level actions (e.g. tool-change) as well as low-level robotic motions (e.g. linear and free-space movements) is a sub-class of task and motion planning (TAMP) from the robotics planning literature [Garrett et al. 2021]. While many algorithms have been developed by the robotics community to automatically plan for both actions and motions, the formulation of such problems requires domain-specific expert knowledge that can be unfamiliar to many users. We believe this expertise gap will be closed by empowering designers and engineers to formulate their problems using a protocol that bridges between high-level intentions and low-level planning algorithms. Such a formulation can be seen as a new *digital* version of the traditional, *analog* architect-engineer-contractor conversations of relevance suited for the new era of digital fabrication.

Contribution. This paper presents the formalization of general-purpose robotic assembly planning with non-repetitive plan skeletons. Our method uses abstracted actions arranged in a flowchart to enable designers to easily describe complex, non-repetitive assembly processes. The other benefits include:

- The process description is *decoupled* from the implementation of automatic solvers and motion planners.
- The process description is fully *parametric* (e.g. geometry, joints, neighbour relationships and number of elements). Different architectural schemes can be evaluated without reformulation.
- It establishes a *protocol* between architectural design, process design and planning, allowing better separation of work and promoting collaboration between different expertise.
- The formulation is compatible with *non-sequential* motion planning, allowing difficult motions to be planned first. Process designers can easily control the planning priority based on experience and intuition, which can dramatically improve planning efficiency.
- The outputted trajectories from motion planning can be directly used for robotic *execution*.

We use a recently published spatial timber assembly process [Leung et al. 2021] as a case study to demonstrate the benefit of the proposed flexible planning framework. In particular, this assembly process has multiple grippers and fastening tools, a non-repetitive plan skeleton, and requires manipulation of long timber elements in a dense, congested environment. This allows us, a team of architects and engineers, to use the plan skeleton formulation as a protocol to effectively collaborate and to perform this case study. The method and planning results are validated by the real-world robotic construction of a spatial frame structure (4.8 x 3.0m footprint, 3.4m tall) comprised of 40 pieces of 100x100mm profile timber elements.

2 CHALLENGES AND RELATED WORK

Early Work in Assembly and Manipulation Planning. Automatic assembly of mechanical parts or structures is among the first few envisioned applications of industrial robots [Ayres and Miller 1983]. Investigations into generating assembly sequences that allow humans or robots to assemble mechanical parts based on design CAD files dates back to 1980s [De Fazio and Whitney 1987; De Mello and Sanderson 1990; Wilson 1992]. This line of work focuses on low-level constraints such as mutual blocking relationship during assembly, but ignores the geometric constraints introduced by robot manipulators.

Manipulation planning problems in which the goal is not just move the robot without collision but also to operate on the objects in the world have been addressed from the earliest days of motion planning to this day, for example [Alami et al. 1990; Garrett et al. 2015; Hauser and Ng-Thow-Hing 2011; Krontiris and Bekris 2015; Lozano-Pérez 1981; Siméon et al. 2004; Stilman and Kuffner 2008]. However, the lack of open-source implementations of these algorithms and a proper modeling interface to connect them with practical design and construction problems makes their usage rare in architectural digital fabrication projects.

Early Work in Architectural Robotics. A number of architectural projects have used industrial robots to create bespoke spatial assemblies [Eversmann et al. 2017; Hack and Lauer 2014; Helm et al. 2015; Thoma et al. 2018]. However, many of these early works adopt a trial-and-error method for planning the actions and robotic motions, often by manually (1) assigning a fixed plan skeleton, (2) guessing a construction sequence, (3) guessing robot target configurations. Although existing software packages can support these basic operations through performing point-wise kinematics checks [Braumann and Brell-Cokcan 2011; Schwartz 2012] and configuration-to-configuration motion planning [Gandia et al. 2018; Sucan and Chitta 2018], applying a strict ordering on the actions and solving linearly for trajectories can lead to a "stuck" situation in a dense, congested environment. For example, grasp poses or configurations that are feasible in the earlier steps might lead to infeasible situations in the subsequent actions. This leads to a highly inefficient solver that requires a lot of manual backtracking.

Sequence and Motion Planning. In assembly problems that have a fixed plan skeleton, such as single-robot spatial extrusion and pick-and-place assembly, the robot repetitively performs certain action

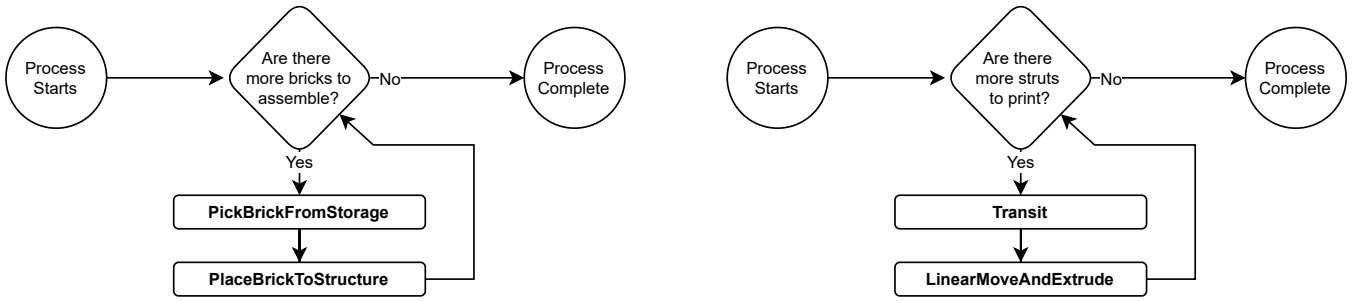


Figure 2: Plan skeleton for repetitive assembly processes: brick wall assembly (left) and 3D extrusion (right)

primitives in a fixed order (Fig. 2). The plan skeleton thus has a fixed length and pattern, which removes any challenge of high-level action planning. In these cases, the planning problems can be reduced to Sequence And Motion Planning (SAMP) problems, where the planner only needs to fill in the construction sequence, i.e. which element to assemble at each step, and the robotic motions. Algorithmic investigation of SAMP began in the robotic extrusion of bar structures with arbitrary geometries and topologies. Early work along this direction addressed sequence planning for a disembodied end effector, ignoring the robot arm [Gelber et al. 2018; Huang et al. 2016; Wu et al. 2016; Yu et al. 2016]. One recent example is Choreo, which plans both the assembly sequence and robotic motions for extrusion processes [Huang et al. 2018]. However, Choreo separates planning into separate sequence and transit phases, which can prevent it from finding solutions to feasible problems in some cases. Recent research has proposed scalable planning algorithms to solve large SAMP problems effectively without the use of human guidance [Garrett et al. 2020a; Huang et al. 2021]. However, it is hard to generalize these specialized search algorithms to general assembly domains without a pre-assigned, fixed plan skeleton. This limits the problems that these algorithms can address to a small category that is overly simplified, compared to more realistic construction processes.

Task and Motion Planning. Task and motion planning (TAMP) bridges both symbolic reasoning of actions to achieve goals and geometric reasoning in search of a collision-free robotic motions [Garrett et al. 2021]. Research in this area seeks to combine discrete task planning from the artificial intelligence (AI) community [Ghallab et al. 2004] and continuous motion planning from the robotic community [LaValle 2006] to allow reasoning on both levels simultaneously [Garrett et al. 2018; Srivastava et al. 2014; Toussaint 2015]. In order to solve a broad class of TAMP problems, Garrett et al. [2020b] proposed PDDLStream, a modular, domain-agnostic planning language for formulating robotic problems with symbolic task definitions. By using logical predicates to describe system states and symbolic operators to represent actions, PDDLStream and its solvers can automatically reason about the order of actions, while also planning valid robotic motions. However, PDDLStream modeling requires the users to formulate their planning problem in the format of symbolic states and actions, which is rarely used in the architectural community. In contrast, working directly with representations of high-level assembly and construction actions is more relevant to those working in architectural robotics. The missing gap is not one of solvers, but one of the challenge of problem formulation.

This paper aims to help bridge the gap by demonstrating the TAMP-based plan skeleton formulation process with a realistic and complex case study, bringing robotic construction closer to the problem encoding used by the TAMP solvers, and thereby enabling flexible, efficient planning for a wide range of complex and realistic structures.

3 FORMULATING A CONSTRUCTION PROCESS INTO A PLAN SKELETON: FLOWCHART INTERFACE

In order to simplify the formulation of an assembly process for a wider user group, this paper proposes the use of a two-step construction planning process, by first arranging high-level *actions* in a flowchart and then breaking down the actions into lower-level *movements*. This flowchart method is particularly intuitive for a designer to use and the result can be compiled automatically into plan skeletons and subsequently used to plan motions for the entire assembly process. An overview of the design to execution workflow is shown in Fig. 3.

3.1 High-level Actions

The *actions* in our formulation are user-defined, high-level abstractions of robotic manipulations skills. Each action consists of one or more atomic *movements* that can involve a combination of robots and tools. For example, the brick stacking process in Fig. 2 consists of two high-level actions: PickBrickFromStorage and PlaceBrickToStructure. The two actions can be repeated to assemble as many elements as required for a given assembly sequence. During each iteration, only the parameters specific to that step need to be changed, for example, the location to pickup and place a specific brick.

In order to illustrate how actions play an important role in more complex scenarios, we will extend our discussions based on a prior spatial timber assembly process [Leung et al. 2021] (Fig. 1). This process utilized a group of distributed robotic clamps and grippers as well as a single 6-DOF industrial robotic arm inversely-mounted on a 3-DOF gantry. The process is designed to automatically assemble timber structures consisting of linear timber elements connected with carpentry lap joints. While in the previous publication Leung

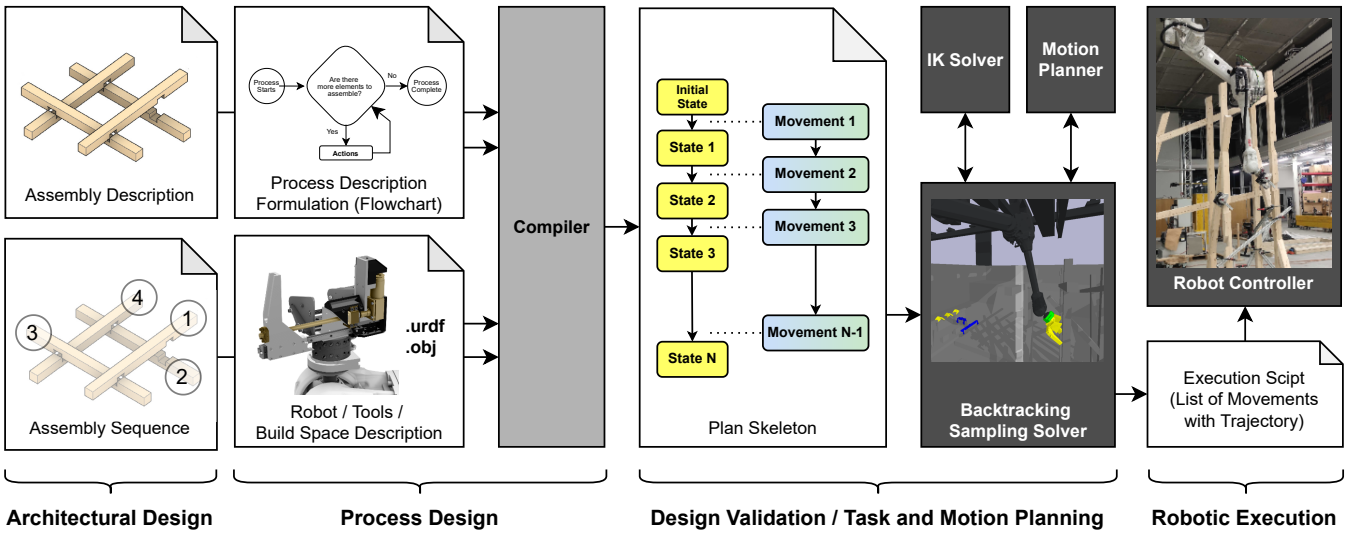


Figure 3: Generalized design, validation, and execution workflow for robotic assembly processes envisioned by the authors. The "Design Validation / TAMP" portion is highly automatic. Images are symbolic references to our case study.

et al. [2021] only used the robot for picking and placing the elements and had human operators to place the clamps, the case study in this paper presents an updated version of the process, tested physically, by using the robot to automatically perform all the actions. Of particular interest for TAMP is that a variable number of robotic clamps are used during each step to apply large assembly forces while the robotic arm manipulates and supports the weight of a timber element in space.

Conditional Statements. Although a different number of clamps are needed for each step, the flowchart method can still be applied by adding extra conditional statements to create inner loops within each step. In our case study, an iterative loop is added (Fig. 4.b) for the robotic arm to perform as many PickClampFromStorage and PlaceClampToStructure actions as necessary. Similarly, another conditional loop is added (Fig. 4.c) for the robotic arm to detach all the clamps after they have been used. On the other hand, conditional statements can also be used in the flowchart to trigger different actions based on specific properties of that step. In our case study, a conditional statement (Fig. 4.d) allow for automatically deciding whether to perform PlaceElementWithoutClamps or PlaceElementWithClamps depending on whether clamps are used for that element.

The use of a flowchart allows a process designer to focus on arranging high-level actions and defer their implementation details to a later stage. Combined with the use of conditional statements, the high-level actions reduce unnecessary specificity during process design and reduce code redundancy when later implemented. As a counterexample, it would be possible to list out all possible clamp and no-clamp scenarios in our case study as sequential scripts. However, this implementation would be very hard to maintain, reuse, and alter.

3.2 Low-level Movements

The second step after creating the flowchart is to break down each action into a sequence of low-level movements. The movements refer to the primitive skills that a robotic system or the tools can perform. They should be formulated to be highly atomic for maximum modularity and reusability across different actions. The list below shows three common types of movements. In practice, custom movements can be formulated for a specific robotic setup (see Section 6).

- **Robotic Movement** - actions of the robotic system that requires a motion planner for computing trajectory. During a robotic movement, tools and workpieces that are attached to the robot move together with the robot. It is possible to impose additional constraints on the robot, such as constraining the end effector to follow a linear path in 3D workspace (linear robotic movement). If no additional constraints are imposed, the movement is referred to as a free robotic movement. Section 4 provides more information on motion constraints.
- **Tool Movement** - discrete actions executed by tools that are either stationary or attached to the robotic system, such as opening or closing a gripper, locking or unlocking a tool changer, or turning an electric spindle on or off. These types of movements do not require computing a trajectory for the robot but may change the shape of a tool or change the attachment status of a workpiece or a tool (whether they are attached to the robot).
- **Manual Movement** - any other type of movements that are not executed automatically, such as a human manually fixing elements, making structural connections, or inspecting the structure. If these movements change the state of the objects in the scene, it is important to update the corresponding planning scene for the motion planner. Instead of

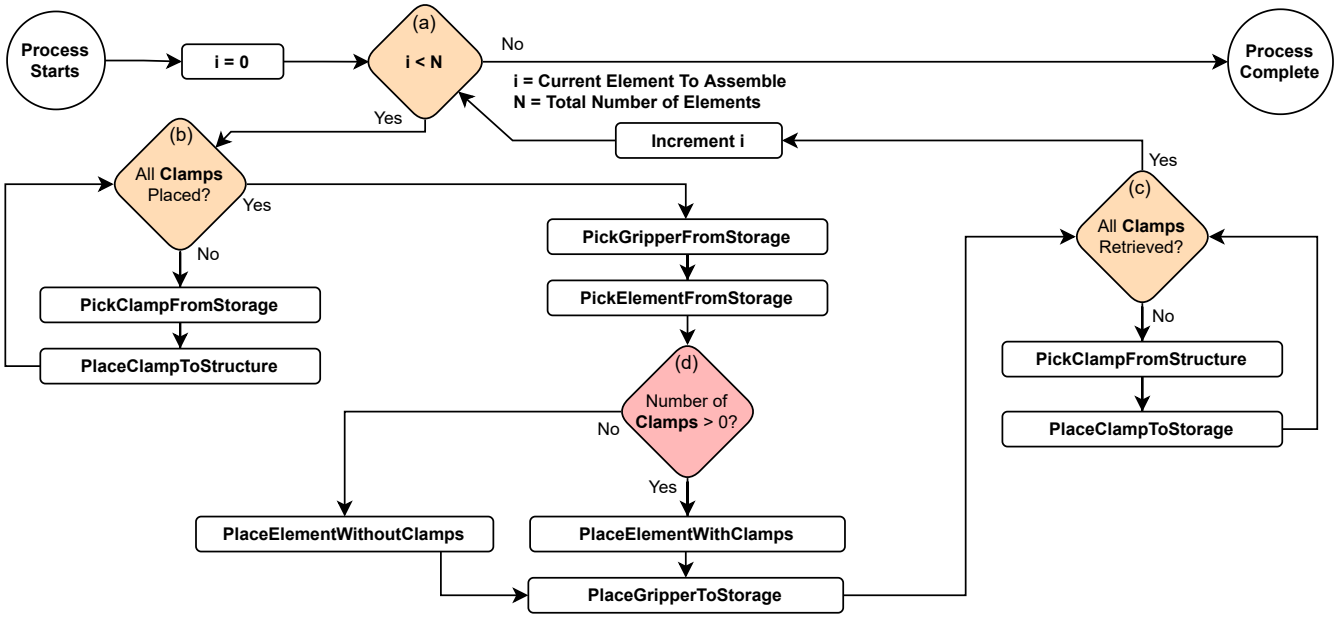


Figure 4: The flexible planning skeleton of the case study expressed as a flowchart, showing loops (a, b, c in orange) and conditional statements (d in red) that are evaluated for each element.

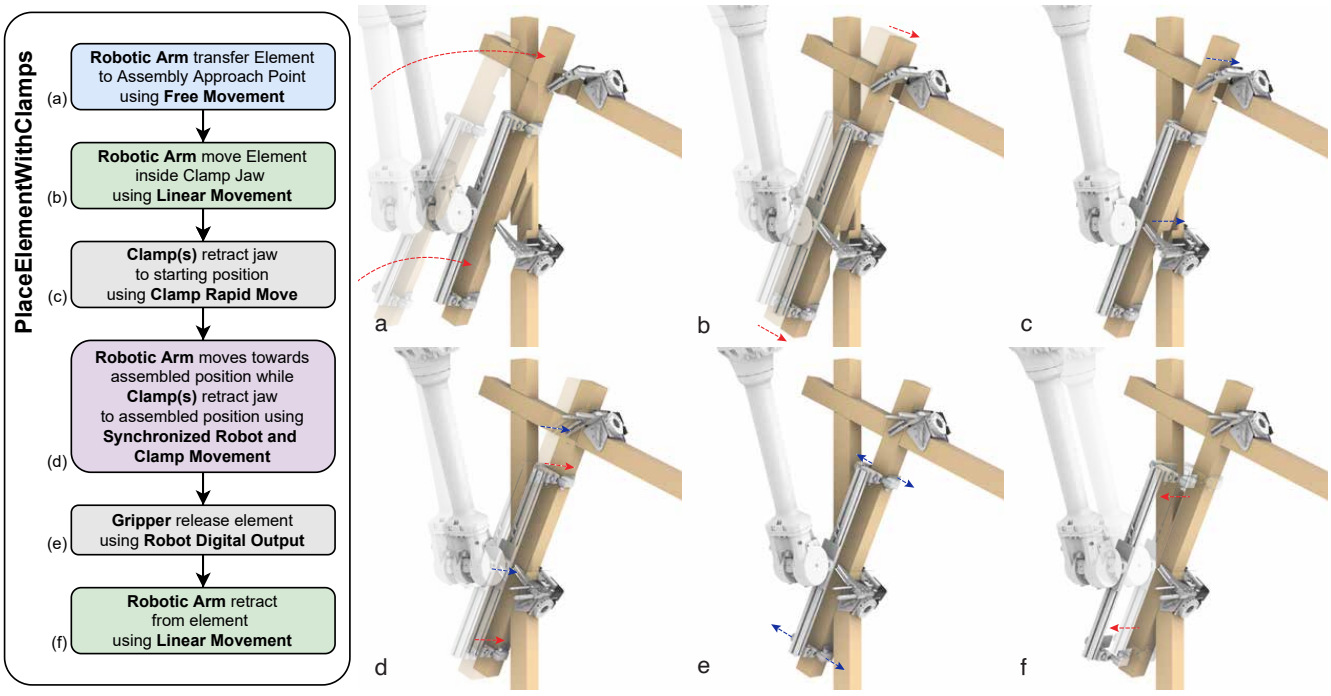


Figure 5: Movement decomposition of action PlaceElementWithClamps in our case study. Please refer to the legend of Fig 6 for the color coding of movements. Drawings on the right (a - f) shows an example of the planning scene after each Movement. Red arrows indicate movements caused by the robotic arm, blue arrows indicate tool movements. Note that step (d) is a custom movement that requires robot arm and clamps to move synchronously.

executing machine code, Manual Movements simply trigger an Operator Stop and wait for human confirmation.

The decomposition of an action into movements require a level of granularity that is determined by the motion planners at hand,

specifically, the motion constraint (see Section 3.4). For example, we cannot have a free motion and a linear motion combined as a single movement because they require two different planners to solve them. On the other hand, a free motion can be subdivided into several smaller chuck of concatenated free motions.

Fig. 5 shows the decomposition of one of the Actions (PlaceElementWithClamps) used in our case study. Note that a custom-formulated "Synchronized Robot & Clamp Movement" is used to model the synchronized movement unique to our case study. This movement requires a corresponding planner (in our case, similar to a linear motion planner) for motion planning.

3.3 Compiling a Flowchart Into a Plan Skeleton

The final step before performing motion planning is to compile the flowchart into a *plan skeleton.* A plan skeleton is a sequential list of movements (Fig. 6), each with an associated packet of information passed to the motion planner. The compiler is a piece of software that is created by the process designer to combine assembly description, assembly sequence, process description (flowchart, actions and movements) with a specific robotic setup. A typical compiling process involves the following tasks:

(1) **Gather the assembly sequence**: Depending on the type of assembly, the sequence can be manually specified (our case study) or automatically computed from the assembly description using heuristics (e.g. a brick wall).
(2) **Gather the sequence for other loops within each construction step**: For example, in our case study, we have two additional inner loops within each construction step regarding the sequence of attaching and detaching clamps. These are automatically assigned based on available tools.
(3) **Evaluate the conditional statements**: All the conditional statements in the flow chart can be evaluated from the assembly sequence and properties in the assembly description. The flowchart will therefore turn into a linear sequential list of Actions.
(4) **Gather action-specific parameters**: Actions may contain parameters that are computed from the assembly description, such as the target frame for the robot or tool to reach or the grasp pose. These parameters can either be constant (for example, always holding a brick on its center from its top face) or variable (for example, in our case study, holding a timber element along various locations and directions along its longitude axis).
(5) **Decompose actions into movements and gather movement-specific parameters** - Movements may contain parameters that are copied from its parent action (e.g. target frame, tool id) or computed directly from the assembly description (e.g. allowed collision pairs between the workpiece and its to-be-connected neighbors, see Section 3.4).

3.4 Planning constraint for motion planners

The primary purpose of compiling a high-level flowchart into a plan skeleton is to convert a multi-stepped planning problem into atomic motion planning tasks that an off-the-shelf motion planner (MP) can individually solve. In general, motion planners search for a robotic trajectory within a feasible configuration space (described in joint positions) defined by constraints.

- **Joint Limits** - In general, an MP will stay within the joint limits of the robotic system. This limit is typically non-changing.
- **Robot Collision** - The MP will prevent the robotic system (including any attached tool and grasped workpieces) from colliding with itself or the stationary environment. Extra *allowable collision pairs* can also be specified for the MP to ignore expected contacts. Note that the environment and the expected contacts may change from step to step due to the progression of the assembly task.
- **Motion Constraint** - Constraints can be specified by the process designer to achieve a more controlled motion. For example, a linear movement constraint requires the tool tip to stay on a linear path in the workspace. In our case study, only free and linear movements are used. However, other constraints such as twisting and rotational motion constraints exist [Berenson et al. 2011].
- **Targets** - The MP requires start and end targets to be specified as input. These targets can be defined as a loosely constrained tool pose (also called a tool frame) or a prescribed robot configuration (joint position). Typically, a tool pose is specified and the MP can freely sample a valid robot configuration by internally calling an inverse kinematics (IK) sampler. A fixed, rigid robot configuration can be used in cases where a specific configuration is preferred by the designer, such as tool change positions and workpiece pickup positions [1] .

Fixed Constraint Within One Movement. By convention of most existing MPs, these planning constraints are fixed within one planning call. Our movement formulation thus also follows this convention and will not allow changing constraints within one movement. However, many assembly processes (including our case study) require changing the allowable collision pairs, such as during a tool change or when grasping a workpiece. In order to overcome the limitation, it is possible to add a linear movement shortly before making contact (e.g Fig 5.b) and shortly after breaking contact (e.g Fig 5.f) to specify different allowable collision pairs. Moreover, the linear motion constraint can be beneficial for creating a predictable trajectory for such operations.

4 PLAN SKELETON SOLVER

A single motion planning task can be solved using our movement formulation by passing the robot description, tools descriptions, object geometries (meshes), and planning constraints to a corresponding MP (based on the required motion constraint). However, a more involved solver is needed to solve all the sequentially coupled movements in a plan skeleton. Specifically, the "intersection" between two adjacent movements trajectory must share a common robot configuration to ensure joint-space continuity during

[1] The exact positions of these targets often depends on the physical setup and are often acquired by manually jogging the robotic system to alignment and then reading out the robotic configuration to achieve maximum repeatability. This technique is used throughout our case study for all the tool storage positions and the element pickup positions.

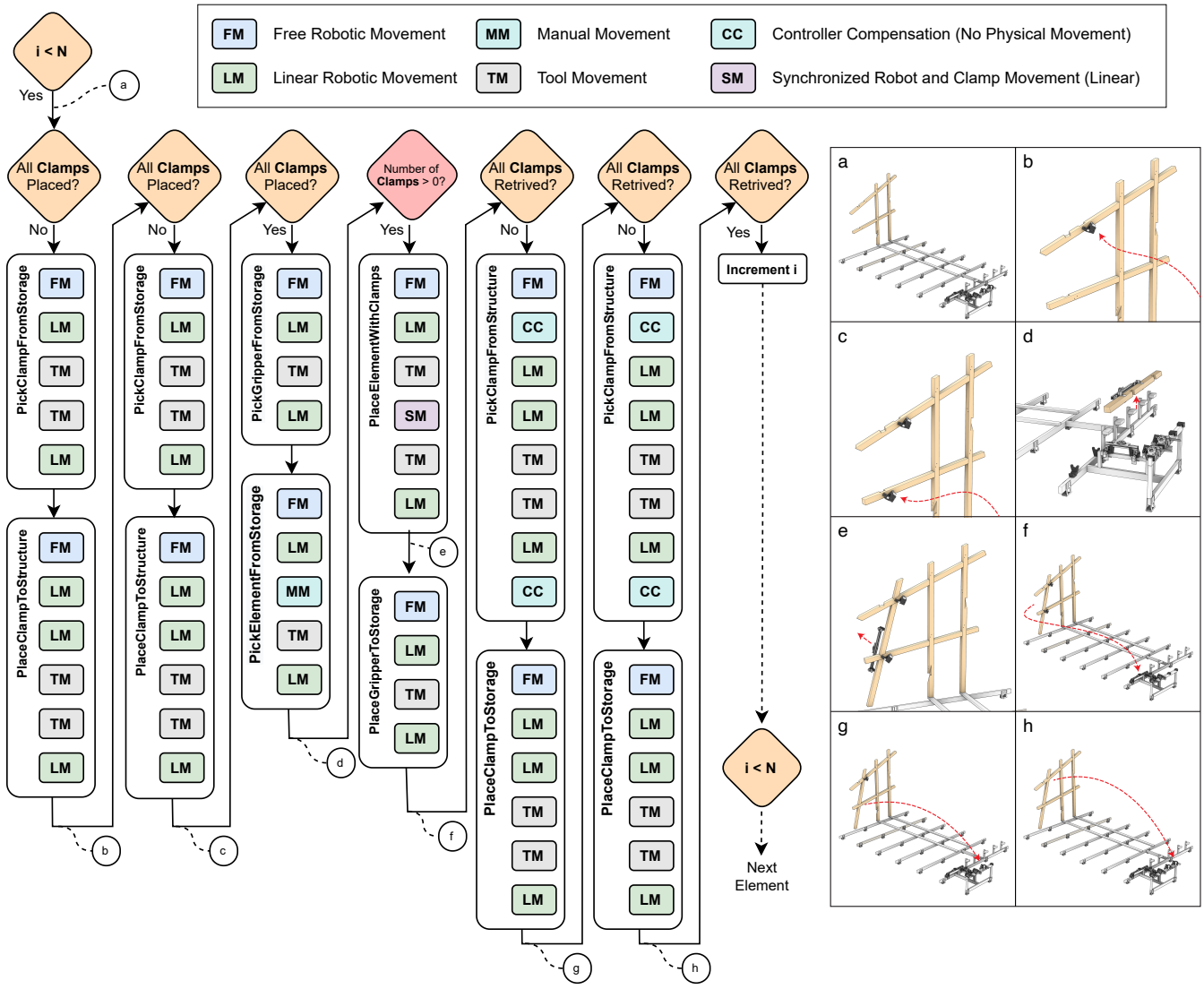


Figure 6: Diagrammatic output of the compiler: a full plan skeleton for one construction step (5th element in the sequence) in our case study, showing the 69 Movements created from 12 Actions. Images (a-h) on the right shows the state of the planning scene at selected moments. Red arrows indicate the general movement of the active tool before the state.

execution. Therefore, a solver needs to pass the ending robot configuration of the previous motion as the starting target for the next MP task.

4.1 Linear Sequential Solver

A simple method of solving all the movements in a plan skeleton is to solve individual movements in a fixed linear sequence. Starting from the first movement, a trajectory is planned by calling the corresponding MP. The robot configuration at the end of this trajectory is propagated as the starting target constraint for planning the next movement (Fig. 7). This is repeated for all the robotic movements (skipping over non-robotic movements) until all the movements are planned.

Due to the stochastic nature of MPs and the underlying IK samplers, a planning request may fail to find a solution within a reasonable time (i.e. timeout). In this case, the cause of the failure may lie in the planning result of previous movements. Specifically, the randomly sampled ending robotic configuration of the previous movement can create an unfavorable starting constraint for the current movement. Therefore it is necessary for the solver to adopt a backtracking or restarting mechanism for rewinding the planned movements and allow for another random attempt.

4.2 Nonlinear Solver

Unfortunately, the seemingly intuitive sequential planning strategy is highly inefficient at solving plan skeletons with scattered fixed

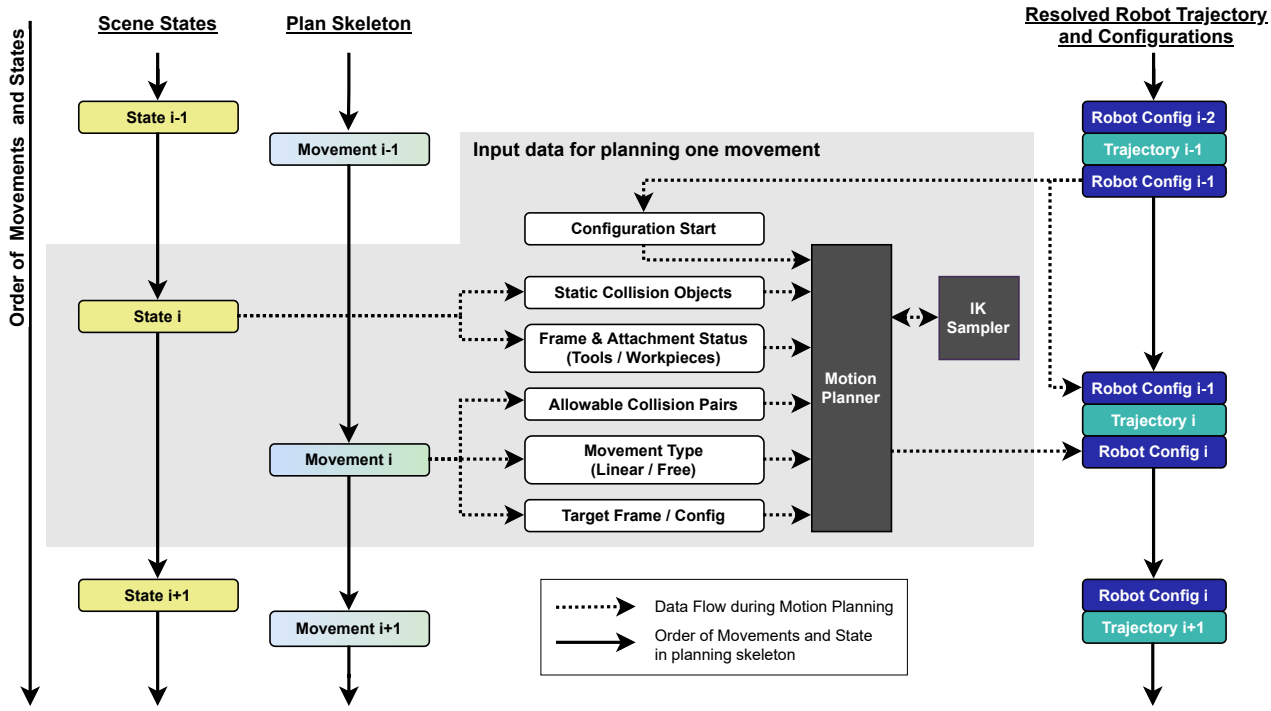


Figure 7: Diagram showing data flow for planning one movement within a plan skeleton. Note that the robotic configuration of the planning result is used for planning the next movement, this constraint ensures motion continuity.

robot configurations (e.g. for tool change and workpiece pickup) and with scattered difficult movements (e.g. narrow passages when manipulating long timber elements in congested environment). The backtracking or restarting method in a sequential solver will result in a lot of wasted effort solving the easier portion of the plan, only to be backtracked by a difficult movement.

One example of narrow passage manipulation can be seen in Fig 8. It shows two possible robot configurations after the robotic arm brings a clamp to the joint using free movement. However, one of them cannot proceed further due to an imminent collision. The second described difficulty, related to a fixed robotic configuration, can be seen in the tool changes in our case study. There, a free movement is used to bring the robot close to the tool storage and then a short linear movement is used to bring the robot to a predetermined fixed configuration. When solved linearly, the probability of the free movement to sample a configuration close enough to the fixed configuration is too low for the linear MP to bridge. In our case study, the success rate of solving the movements of a single element using a linear solver is practically zero (see results in Section 5).

In order to overcome this problem, we introduce a non-linear solving method (Fig. 9.a) based on a priority heuristics that describe the "difficulty" of a movement. In general, movements with more constraints are more difficult to plan, since more constraints leads to smaller feasible robot configuration space [Kingston et al. 2018]. Similarly, movements with more, larger attached objects or collision objects have smaller collision-free robot configuration space, thus harder to plan. These heuristics allows the solver to plan difficult movements first, thereby failing quickly before spending time on the other movements. The starting robotic state constraint propagation works essentially the same way as the sequential planner, but it propagates both forward and backwards to neighboring movements. This intuition essentially allows the backtracking algorithm to eliminate unsuitable configuration candidates earlier and can substantially improve success rates.

Case-Study Specific Priority Computation. The priority heuristics used in our case study can be seen in Fig.9b. We first use the general rule of thumb to classify movements that have more constraints (motion constraints and start configuration constraints) to be more difficult. In addition, the process designer can assign a priority flag (a Boolean value) to specific movements of an action to denote higher priority. The inclusion of designer intuition into our heuristics allows a process designer to help the solver making informed decisions. We also found that it is natural for a process designer to speculate which movements are the most constrained and thus most difficult to plan a motion for.

5 SOFTWARE IMPLEMENTATION AND RUNTIME RESULTS

In order to validate our method and to generate trajectories for executing the case study, we implement our algorithms and solvers using Pybullet [Coumans and Bai 2016] as a simulation platform, which takes care of collision checking, forward kinematics, and

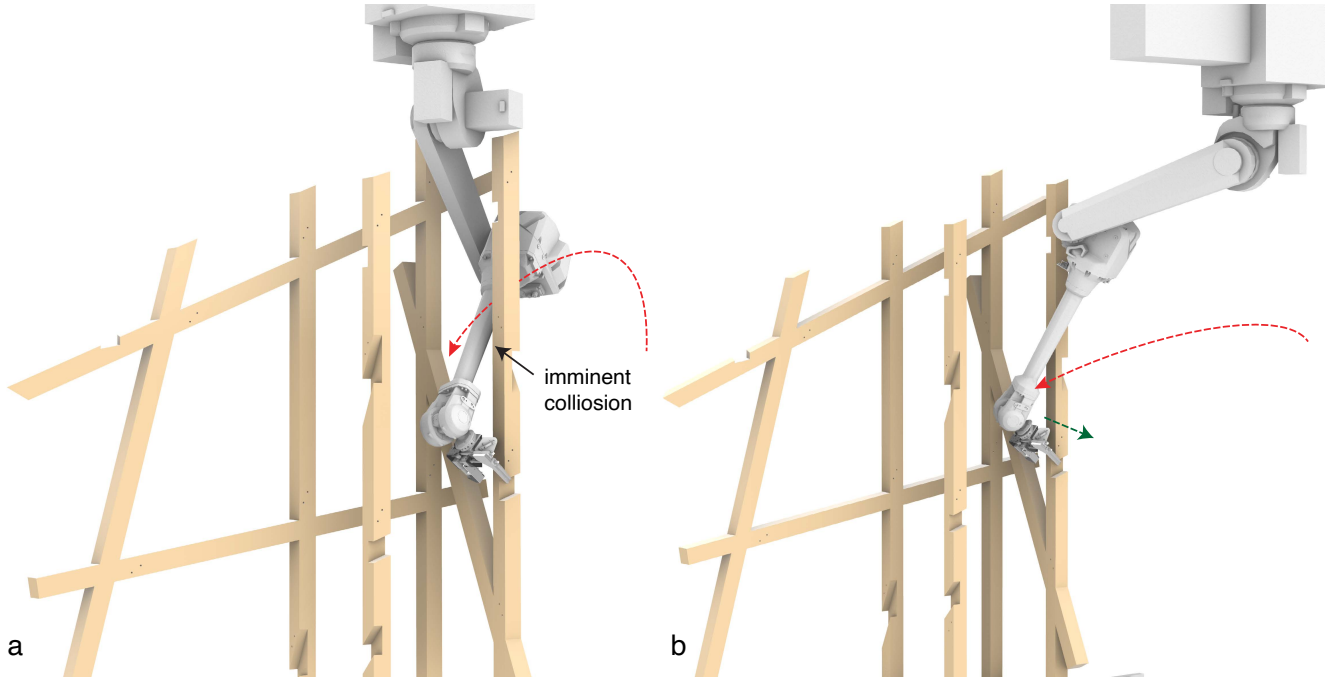


Figure 8: Comparison between a good and bad state after the robotic arm transfers a clamp to the structure (a free movement), before the final approaching movement (a linear movement). Due to the stochastic nature of the IK sampler, any good or bad scenarios are equally likely to happen. *(a)*: **The robotic arm body is sandwiched between the structure after its previous movement (red arrow). The linear MP for the next movement cannot find a collision free path.** *(b)*: **The robotic arm holds the clamp in a different configuration that is possible for next movement (green arrow).**

visualization during motion planning. Robot and tool description are based on Unified Robot Description Format (URDF) and meshes (.obj and .stl). Assembly description, robotic configuration and geometry classes are extended from the compas framework [Van Mele and many others 2021]. Flowchart, action-movement decomposition and compiler are implemented by the authors in Python, without the graphical visualization or user interface described in Fig. 4 or Fig. 6. Timber element geometry and robotic tool geometry are parsed from objects modeled in Rhinoceros 6 [McNeel et al. 2010]. Assembly sequence and assembly directions for individual element are specified manually using an interactive script within the Rhinoceros 6 canvas.

Our free motion planner implements the RRT-connect algorithm [Kuffner Jr. and LaValle 2000]. Our linear motion planner implements Randomized Gradient Descent [Stilman 2010; Yao and Gupta 2007], with Trac-IK as its gradient-based IK solver [Beeson and Ames 2015]. The three extra DOF of our robots is handled by first sampling the gantry position within a sphere near the target end-effector pose, and then using IKFast [Diankov and Kuffner 2008] (an analytical IK solver) to obtain all the configurations for the 6-DOF robot arm. Both MPs comply with the standard interfaces described in the compas_fab [Rust et al. 2018] API, the robotic fabrication package extending the compas framework.

To compare the effect of linear and nonlinear solving, we perform 40 random trials on solving a sequence of movements for a particular timber element (5^{th} in the sequence, the same construction step as shown in Fig. 6). For the linear sequential solver, we include forward (plan from the first movement to the end) and backward order (plan from the last movement to the start). The results shows that when fixed robot configuration constraints are included (for tool-change), the linear sequential solvers cannot solve the problem at all (Fig. 10.a.1). In contrast, the nonlinear planner can solve the problem with a 32.4% success rate. Fig. 10.a.2 shows the average time of successful and failed trials. In addition to the rollout experiment above, we perform 10 random trials of running the planner until success or timeout (1800 seconds) with automatic random restarts. Fig. 10.a.3 shows the planner's average solving time. The dots represents the individual data from the random trials.

To further compare the solvers, we remove the robot configuration constraints and perform the same experiments as above. The result (Fig. 10.b.1) shows that the nonlinear planner still have higher success rate (27.0%) than the linear sequential solvers (16.2%, 10.8%) . The nonlinear solver also has a lower averaged planning time until a solution is found (Fig. 10.b.3). Finally, comparing Fig. 10.a.3 and Fig. 10.b.3, it is interesting to see that the nonlinear planner solves the problem faster when robot configuration constraints are included. The fixed configurations provide useful hints for the solver because these configurations are conjured manually to be in a 'non-stretched' position and tested to be collision free.

Fig. 11 shows the planning time for all the elements along the construction sequence using the nonlinear solver. The variation in planning time roughly corresponds to the difficulty of planning in

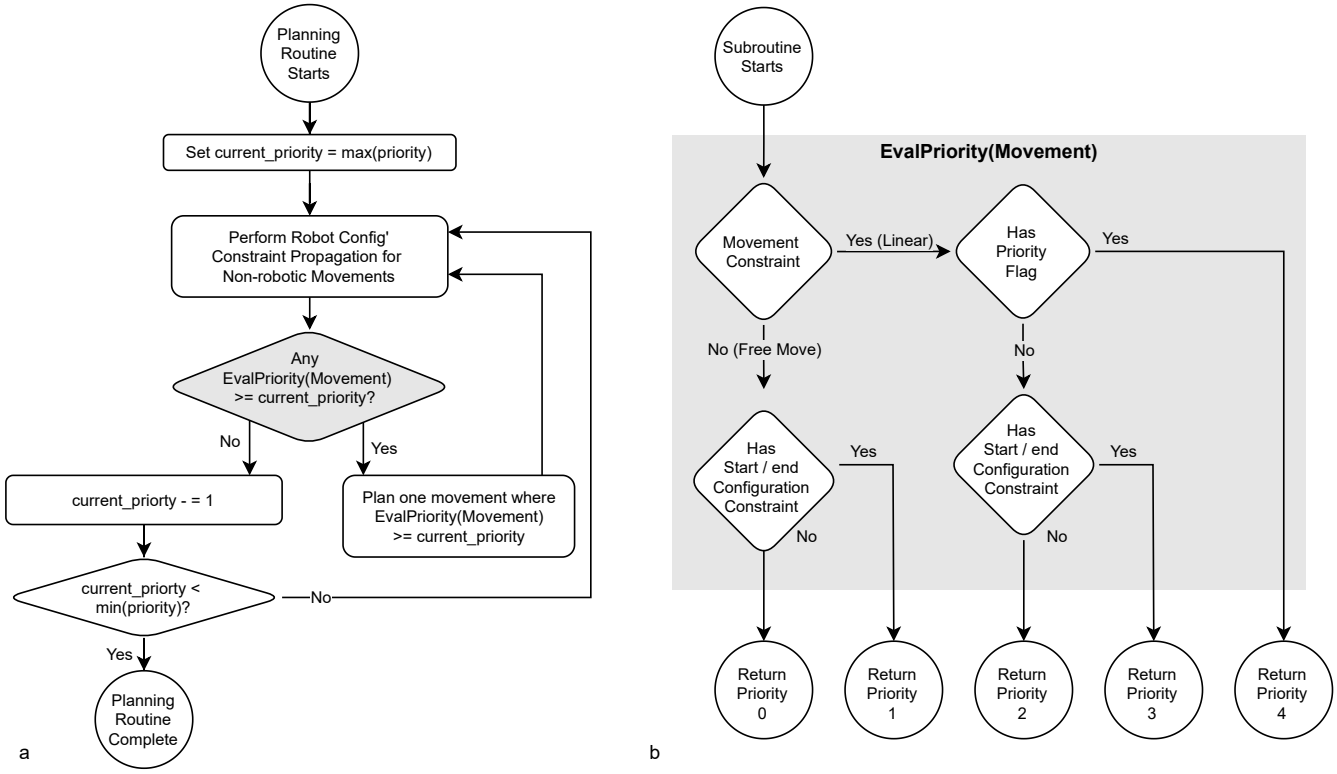


Figure 9: a: Nonlinear algorithm for solving plan skeleton based on a priority heuristics. b: Movement priority heuristics used in our case study, higher scores are considered more difficult and are planned earlier.

different construction steps. A timelapse video of the real-world construction experiments can be found in the supplementary material.

6 EXTENDING OUR FORMULATION FOR PRACTICAL ISSUES IN CONSTRUCTION

The following sections aim to show the compatibility of our process formulation method in coping with various practical situations that arise in the presented case study. Beyond the case study, our formulation method can be further applied to other architectural assembly scenarios. More examples can be found in Section A of the Appendix.

6.1 Inclusion of Temporary Scaffolding

Certain assembly processes require the addition of scaffolding in the middle of the process to temporarily support the structure. Using our formulation, both robotic and manual manipulation of scaffolding can be easily incorporated into the Action Flowchart (Fig. 12). It can also be used to remove workpieces. This allows all of the collision objects in the scene to be correctly modeled and avoids unexpected collisions during execution.

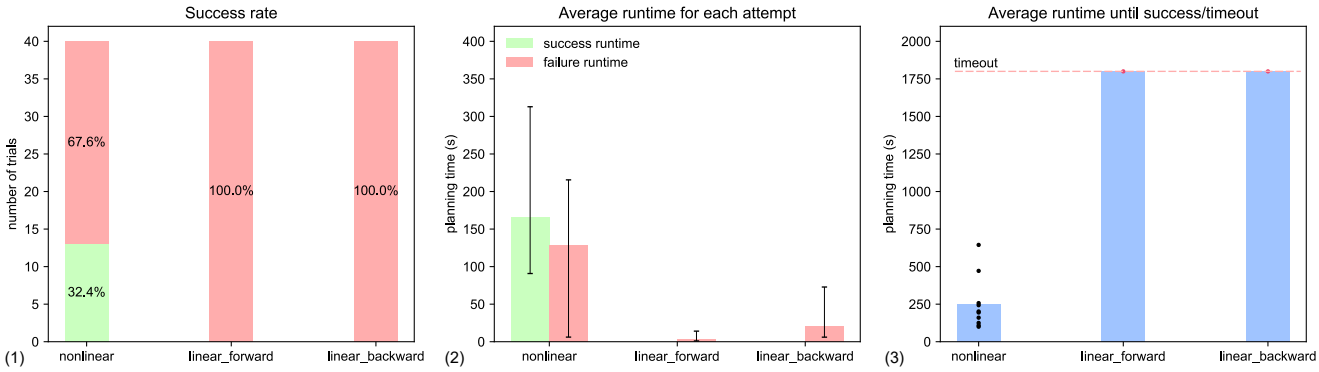
During the construction of our case study, we used manual carpentry clamps and aluminum profiles for temporary structural support. We initially assumed that their geometries were small enough to not cause collisions and therefore did not model them. However, collisions happened multiple times, causing substantial disruption to the process and proving that temporary scaffold modeling is necessary.

6.2 Executing Paths with Controlled Collision

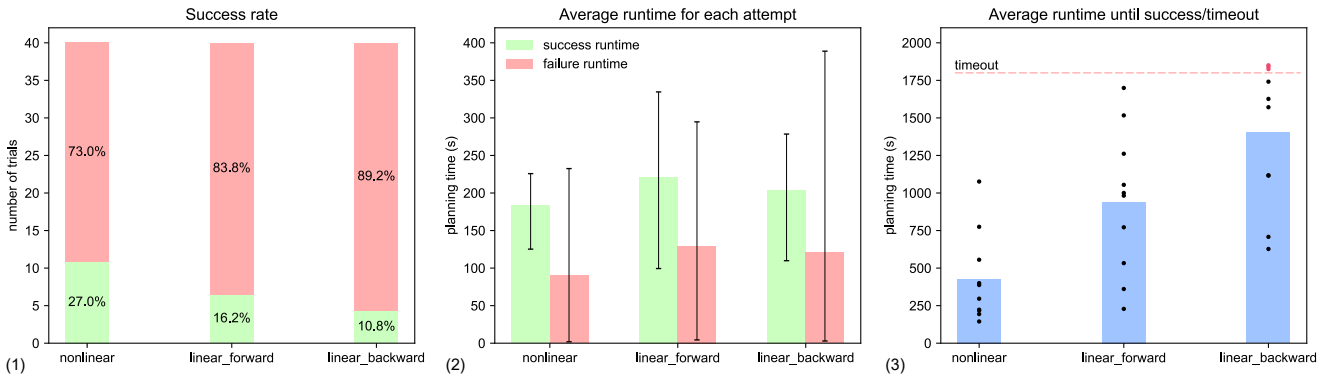
Many different industrial robotic arms now support compliant movement modes that assist alignment or provide contact forces. Depending on the manufacturer, performing such commands may require parameter values such as SoftDirections and PayloadInformation for each motion segment. Using our flowchart formulation method, we programmed these parameters as optional parameters in the corresponding robotic movements for execution with our ABB controller. The one-to-one relationship between the low-level movement formulation and the execution code allows parameters to be passed seamlessly from the designer to the machine during execution.

6.3 Online Visual Alignment

One of the actions in our case study requires the robotic arm to dock with and detach a clamp from the structure. This is a challenging movement as the clamp may have moved during the clamping process and deviate from the programmed position. By adding a camera at the robot flange next to the tool changer, we are able to detect the misalignment based on a captured image. Because the correction amount is small, we implemented it as a Cartesian



(a) Fixed robot configuration constraints (used in tool changing) are included.



(b) Fixed robot configuration constraints (used in tool changing) are ignored.

Figure 10: Comparison between linear and nonlinear solver for solving 45 robotic movement for the 5th timber element.

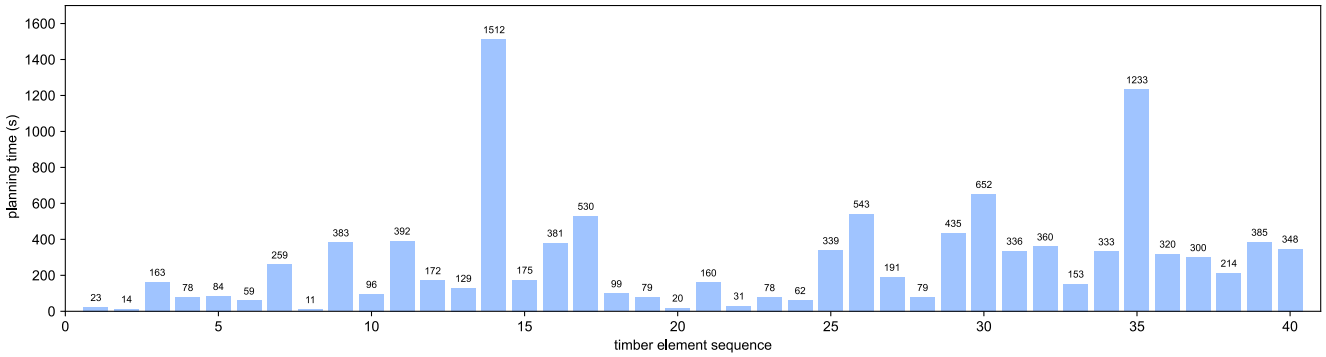


Figure 11: Computing time for nonlinear solver to solve movements for each of the 40 timber elements used in our case study. Element index indicates its place in the construction sequence.

offset instead of re-planning the motion. In our PickClampFromStructure action formulation (Fig. 13), we have added two special movements: AcquireDockingOffset to direct the camera to acquire the correction image, and apply the offset for subsequent movements and CancelDockingOffset to remove it after the dock and detach procedure is completed.

7 CONCLUSIONS AND FUTURE WORK

This paper contributes a new protocol for architectural and construction process designers to communicate intent and knowledge to robotic task and motion planners. This protocol allows for complex, realistic construction robotics applications with non-repetitive

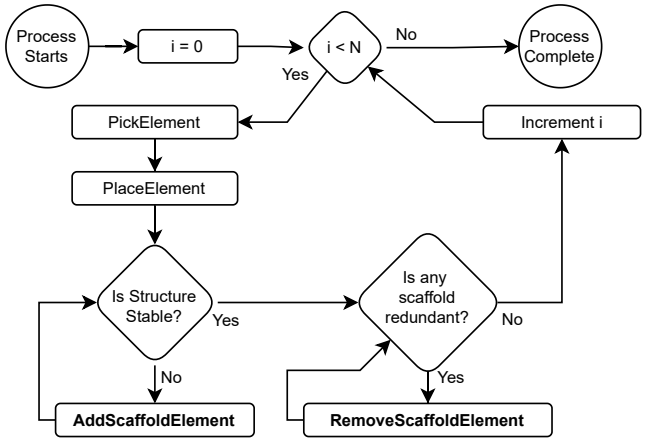


Figure 12: Addition of AddScaffoldElement and RemoveScaffoldElement (highlighted in bold) to a generic pick and place flowchart. This is compatible with our case study flowchart in Fig. 4.

assemblies to make use of efficient, state-of-the-art planning methods that offload low-level efforts from human designers. The method is demonstrated computationally and physically on a real-world case study. Key contributions and directions for future work are given briefly below:

Interface for designers. The computational fabrication field does not yet have a good interface to perform task and motion planning for general-purpose robotic assembly. This paper has presented a new interface concept that will allow designers to formulate and solve planning problems even with non-repetitive action patterns easily. The proposed flowchart-based visualization allows designers to formulate their problems visually using computational logic that understandable both by humans and computers. At the moment, our work did not implement a graphical user interface for manipulating the flowchart. Therefore, designers have to be well versed with Python code to adjust the flowchart logic.

However, we speculate an interactive icon-dragging interface that can be modeled after the visual programming platform Grasshopper [McNeel et al. 2010] . The major input form the architectural designer will be: (1) the geometry of the parts to be assembled (2) the position of the parts after they are assembled. (3) assembly sequence and assembly direction (can potentially be automatic). The process designer should be able to input (1) digital representations of available robots and tools, including their geometry, kinematics, actions and capability. (2) Assembly flowchart. In our demonstration, the architectural designer and the process designer role is filled by two person. However, future work can study whether this separation is useful or necessary. At the moment, the compiler is created specifically for our choice of tools (clamps and gripper) and parts (wood elements with half-lap joints. Future work should study how this can be generalized such that changing tools or part definition does not require adjustments to the compiler.

Extensibility. One key advantage of our formulation is its flexibility. In addition to the timber case study, the examples provided in the appendix demonstrate its versatility on a range of robotic construction problems. However, for repetitive assembly problems, our formulation only provides a generalized description framework but does not offer additional computational benefits (see Sequence and Motion Planning in Section 2). In future work, the method can be adapted to work on multi-robot, multi-element planning using the plan skeleton formulation. Our method is compatible with asynchronous multi-robotic movements (multiple robot agents operating at the same time), mobile robotic movements (e.g. robots with a nonholonomic mobile base [Dörfler et al. 2016]), or even non-discrete robotic movements (e.g. 3D printing [Mitropoulou et al. 2020]), as long as specialized motion planners are provided. For us, the creation of these planners falls onto the domain expert in planning.

Towards full TAMP. Another compelling next step is to open up the parameters that are set by the designers when compiling a flowchart into a plan skeleton (Section 3.3). In this work, the construction sequence and the grasp pose are provided by the architect, using their intuition. But these values can be filled in by a automatic planning algorithm as well, which needs the planner to solve for both symbolic parameters (e.g. the construction sequence) as well as geometric parameters (e.g. grasp poses, robot trajectories).

Finally, our flowchart-based formulation is designed for offline, pre-planning purposes and currently do not support change of design after the fabrication has started or adaptive online re-planning beyond a complete re-computation. An interesting future direction is to investigate how to dynamically re-plan given incremental changes of design or certain scene observations.

As construction robotics advances from boutique examples to real-world deployment, the thoughtful combination of automation and human expertise becomes increasingly important. The methods presented in this paper represent a key step towards this future.

REFERENCES

Rachid Alami, Thierry Siméon, and Jean-Paul Laumond. 1990. A geometrical approach to planning manipulation tasks. The case of discrete placements and grasps. In *International Symposium of Robotic Research (ISRR)*.

Robert U. Ayres and Steven M Miller. 1983. *Robotics, applications and social implications*. Ballinger Pub. Co., Cambridge, Mass.

Patrick Beeson and Barrett Ames. 2015. TRAC-IK: An open-source library for improved solving of generic inverse kinematics. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 928–935.

Dmitry Berenson, Siddhartha Srinivasa, and James Kuffner. 2011. Task space regions: A framework for pose-constrained manipulation planning. *The International Journal of Robotics Research* 30, 12 (2011), 1435–1460.

Johannes Braumann and Sigrid Brell-Cokcan. 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)*.

Erwin Coumans and Yunfei Bai. 2016. PyBullet, a Python module for physics simulation for games, robotics and machine learning. \url{http://pybullet.org}.

Thomas De Fazio and Daniel Whitney. 1987. Simplified generation of all mechanical assembly sequences. *IEEE Journal on Robotics and Automation* 3, 6 (1987), 640–658.

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

Rosen Diankov and James Kuffner. 2008. *OpenRAVE: A Planning Architecture for Autonomous Robotics*. Technical Report CMU-RI-TR-08-34. Robotics Institute, Carnegie Mellon University. https://pdfs.semanticscholar.org/c28d/3dc33b629916a306cc58cbff05dcd632d42d.pdf

Kathrin Dörfler, Timothy Sandy, Markus Giftthaler, Fabio Gramazio, Matthias Kohler, and Jonas Buchli. 2016. Mobile robotic brickwork. In *Robotic Fabrication in Architecture, Art and Design 2016*. Springer, 204–217.

Rebeca Duque Estrada, Fabian Kannenberg, Hans Jakob Wagner, Maria Yablonina, and Achim Menges. 2020. Spatial winding: cooperative heterogeneous multi-robot system for fibrous structures. *Construction Robotics* 4, 3 (2020), 205–215.

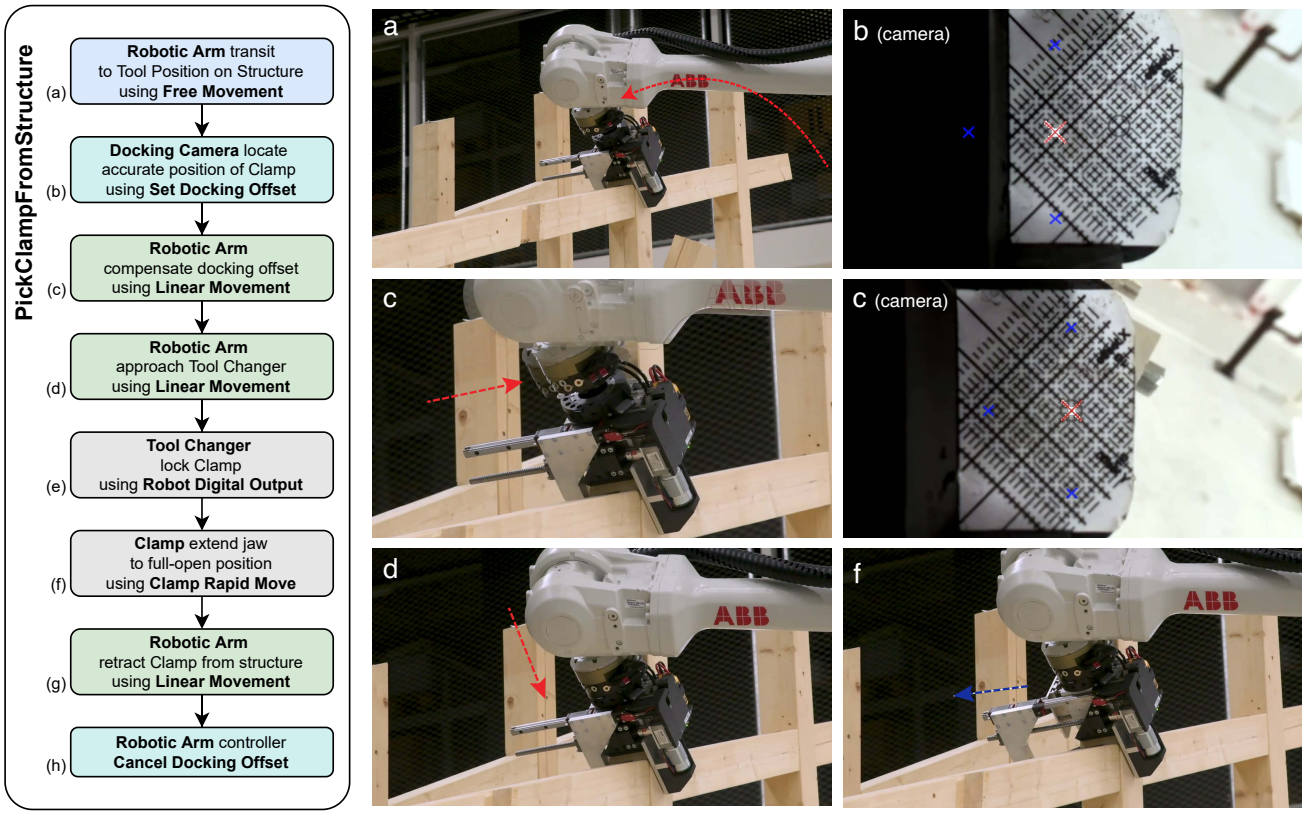


Figure 13: Visual alignment and docking procedures implemented by two special movements: AcquireDockingOffset and CancelDockingOffset. Left: movement decomposition of action PickClampFromStructure in our case study. Photos on the right shows the state after corresponding movements (a,b,c,d,f) are executed. Arrows indicate the movement of the robotic arm (red in a,c,d) and the clamp jaw (blue in f)

Philipp Eversmann, Fabio Gramazio, and Matthias Kohler. 2017. Robotic prefabrication of timber structures: towards automated large-scale spatial assembly. *Construction Robotics* 1, 1-4 (2017), 49–60.

Augusto Gandia, Stefana Parascho, Romana Rust, Gonzalo Casas, Fabio Gramazio, and Matthias Kohler. 2018. Towards Automatic Path Planning for Robotically Assembled Spatial Structures. In *Robotic Fabrication in Architecture, Art and Design*. Springer, 59–73.

Caelan Garrett, Yijiang Huang, Tomas Lozano-Pérez, and Caitlin Mueller. 2020a. Scalable and Probabilistically Complete Planning for Robotic Spatial Extrusion. In *Robotics: Science and Systems XVI*. Robotics: Science and Systems Foundation. https://doi.org/10.15607/RSS.2020.XVI.092

Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomas Lozano-Pérez. 2021. Integrated Task and Motion Planning. *Annual review of control, robotics, and autonomous systems* 4 (2021).

Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2015. Backward-Forward Search for Manipulation Planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vol. 2015-Decem. https://doi.org/10.1109/IROS.2015.7354287

Caelan R. Garrett, Tomás Lozano-Pérez, and Leslie P. Kaelbling. 2020b. PDDLStream: Integrating Symbolic Planners and Blackbox Samplers. In *International Conference on Automated Planning and Scheduling (ICAPS)*. https://arxiv.org/abs/1802.08705

Caelan Reed C.R. Garrett, Tomás Lozano-Pérez, and L.P. Leslie Pack Kaelbling. 2018. Sampling-based methods for factored task and motion planning. *The International Journal of Robotics Research* 37, 13-14 (2018). https://doi.org/10.1177/0278364918802962

Matthew K Gelber, Greg Hurst, and Rohit Bhargava. 2018. Freeform assembly planning. *IEEE Transactions on Automation Science and Engineering* 16, 3 (2018), 1315–1329.

Malik Ghallab, Dana S Nau, and Paolo Traverso. 2004. *Automated Planning: Theory and Practice*. Elsevier.

Fabio Gramazio, Kohler Matthias, and Jan Willmann. 2014. *The robotic touch*. Park Books.

Norman Hack and Willi Viktor Lauer. 2014. Mesh-Mould: Robotically Fabricated Spatial Meshes as Reinforced Concrete Formwork. *Architectural Design* 84, 3 (2014), 44–53.

Kris Hauser and Victor Ng-Thow-Hing. 2011. Randomized multi-modal motion planning for a humanoid robot manipulation task. *International Journal of Robotics Research (IJRR)* 30, 6 (2011), 676–698. http://journals.sagepub.com/doi/abs/10.1177/0278364910386985

Volker Helm, Jan Willmann, Andreas Thoma, Luka Piškorec, Norman Hack, Fabio Gramazio, and Matthias Kohler. 2015. Iridescence print: Robotically printed lightweight mesh structures. *3D Printing and Additive Manufacturing* 2, 3 (2015), 117–122.

Yijiang Huang, Caelan Garrett, Ian Ting, Stefana Parascho, and Caitlin Mueller. 2021. Robotic additive construction of bar structures: Unified sequence and motion planning. *Construction Robotics* (2021). https://doi.org/10.1007/s41693-021-00062-z

Yijiang Huang, Caelan R Garrett, and Caitlin T Mueller. 2018. Automated sequence and motion planning for robotic spatial extrusion of 3D trusses. *Construction Robotics* 2, 1 (12 2018), 15–39. https://doi.org/10.1007/s41693-018-0012-z

Yijiang Huang, Juyong Zhang, Xin Hu, Guoxian Song, Zhongyuan Liu, Lei Yu, and Ligang Liu. 2016. Framefab: Robotic fabrication of frame shapes. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 224.

Zachary Kingston, Mark Moll, and Lydia E Kavraki. 2018. Sampling-based methods for motion planning with constraints. *Annual review of control, robotics, and autonomous systems* 1 (2018), 159–185.

A Krontiris and K E Bekris. 2015. Dealing with Difficult Instances of Object Rearrangement. In *Robotics: Science and Systems (RSS)*. Rome, Italy. http://www.cs.rutgers.edu/~kb572/pubs/Krontiris_Bekris_rearrangement_RSS2015.pdf

James J Kuffner Jr. and Steven M LaValle. 2000. {RRT-Connect}: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Steven M LaValle. 2006. *Planning Algorithms*. Cambridge University Press. msl.cs.uiuc.edu/planning

Pok Yin Victor Leung, Aleksandra Anna Apolinarska, Davide Tanadini, Fabio Gramazio, and Matthias Kohler. 2021. Automatic Assembly of Jointed Timber Structure using Distributed Robotic Clamps. In *PROJECTIONS - Proceedings of the 26th CAADRIA Conference - Volume 1*. Springer, 583–592.

Tomás Lozano-Pérez. 1981. Automatic planning of manipulator transfer movements. *IEEE Transactions on Systems, Man, and Cybernetics* 11 (1981), 681–698. http://ieeexplore.ieee.org/document/4308589/

Robert McNeel et al. 2010. Rhinoceros 3D, Version 6.0. *Robert McNeel & Associates, Seattle, WA* (2010).

Ioanna Mitropoulou, Mathias Bernhard, and Benjamin Dillenburger. 2020. Print Paths Key-Framing: Design for Non-Planar Layered Robotic FDM Printing. In *Symposium on Computational Fabrication* (Virtual Event, USA) *(SCF '20)*. Association for Computing Machinery, New York, NY, USA, Article 6, 10 pages. https://doi.org/10.1145/3424630.3425408

Stefana Parascho. 2019. *Cooperative Robotic Assembly: Computational Design and Robotic Fabrication of Spatial Metal Structures*. Doctoral Thesis. ETH Zurich. https://doi.org/10.3929/ethz-b-000364322 Accepted: 2019-09-17T07:16:57Z.

R. Rust, G. Casas, S. Parascho, D. Jenny, K. Dörfler, M. Helmreich, A. Gandia, Z. Ma, I. Ariza, M. Pacher, B. Lytle, and Y. Huang. 2018. COMPAS FAB: Robotic fabrication package for the COMPAS Framework. https://github.com/compas-dev/compas_fab/. https://doi.org/10.5281/zenodo.3469478 Gramazio Kohler Research, ETH Zürich.

Thibault Schwartz. 2012. HAL: Extension of a visual programming language to support teaching and research on robotics applied to construction. In *Robotic Fabrication in Architecture, Art and Design 2012*. Springer, 92–101.

Thierry Siméon, Jean-Paul Laumond, Juan Cortés, and Anis Sahbani. 2004. Manipulation planning with probabilistic roadmaps. *International Journal of Robotics Research (IJRR)* 23, 7-8 (2004), 729–746.

Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. 2014. Combined Task and Motion Planning Through an Extensible Planner-Independent Interface Layer. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Mike Stilman. 2010. Global manipulation planning in robot joint space with task constraints. *IEEE Transactions on Robotics* 26, 3 (2010), 576–584.

Mike Stilman and James J Kuffner. 2008. Planning Among Movable Obstacles with Artificial Constraints. *The International Journal of Robotics Research* 27, 11-12 (2008), 1295–1307.

Ioan A Sucan and Sachin Chitta. 2018. Moveit! http://moveit.ros.org

Andreas Thoma, Arash Adel, Matthias Helmreich, Thomas Wehrle, Fabio Gramazio, and Matthias Kohler. 2018. Robotic fabrication of bespoke timber frame modules. In *Robotic Fabrication in Architecture, Art and Design*. Springer, 447–458.

Marc Toussaint. 2015. Logic-geometric programming: an optimization-based approach to combined task and motion planning. In *IJCAI International Joint Conference on Artificial Intelligence*. AAAI Press, 1930–1936.

Tom Van Mele and many others. 2017-2021. COMPAS: A framework for computational research in architecture and structures. https://doi.org/10.5281/zenodo.2594510 http://compas.dev.

Randall H Wilson. 1992. *On Geometric Assembly Planning*. Ph.D. Dissertation. Stanford University.

Rundong Wu, Huaishu Peng, François Guimbretière, and Steve Marschner. 2016. Printing arbitrary meshes with a 5DOF wireframe printer. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 101.

Zhenwang Yao and Kamal Gupta. 2007. Path planning with general end-effector constraints. *Robotics and Autonomous Systems* 55, 4 (2007), 316–327.

Lei Yu, Yijiang Huang, Zhongyuan Liu, Sai Xiao, Ligang Liu, Guoxian Song, and Yanxin Wang. 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)*. 298–307.

A APPENDIX: OTHER FORMULATION EXAMPLES

In this section, we provide two examples of using our flowchart interface to formulate non-repetitive robotic assembly processes published by other researchers. Figure 14 describes a multi-robot assembly process where some robots are used as temporary support. It is a generalized description from the two-robot steel tube assembly process by Parascho [2019]. This is different from the scaffolding approach described in Section 6.1, because the robots can take turns to support and transfer elements. Figure 15 describes a robotic fiber winding process by Estrada et al. [2020] where a robotic arm interfaces with a Cartesian machine.

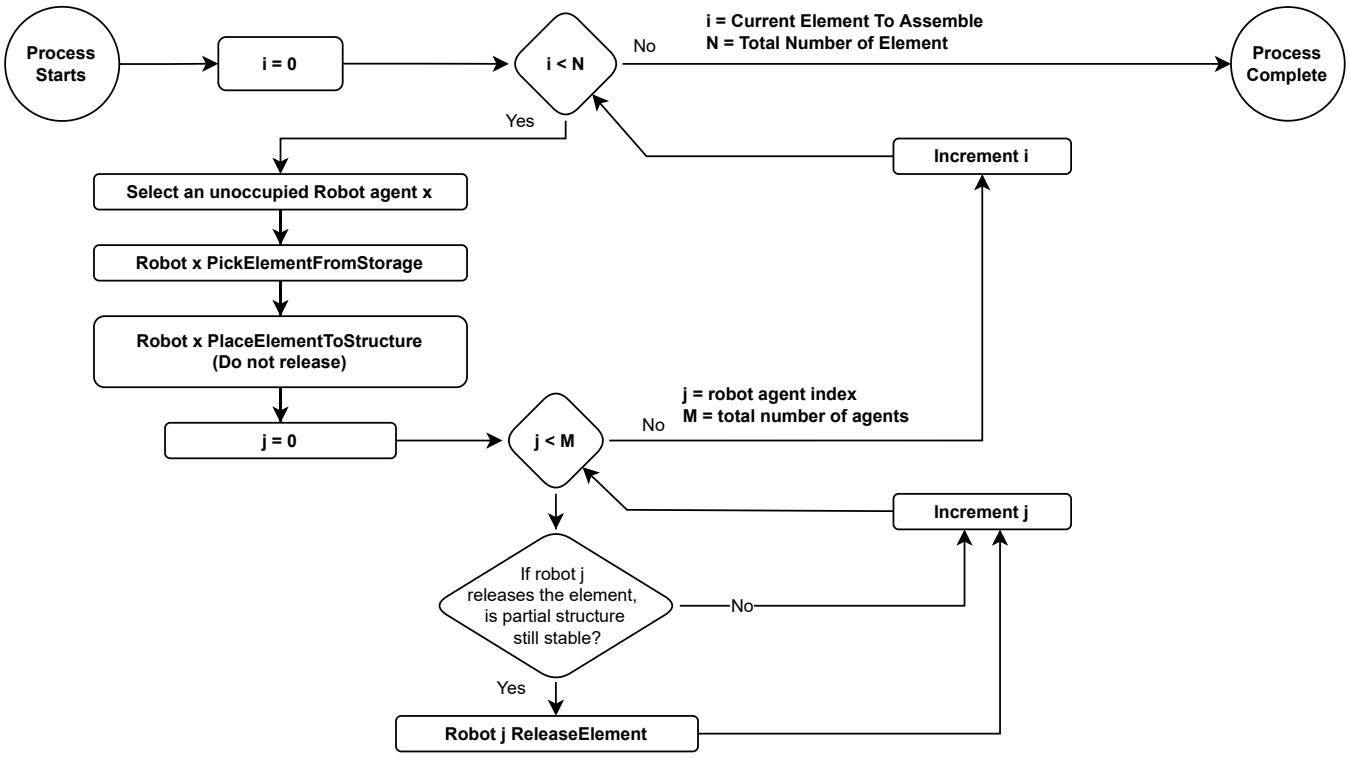


Figure 14: Construction process flowchart for multi-robot assembly processes. This is a generalized version of the process presented in [Parascho 2019].

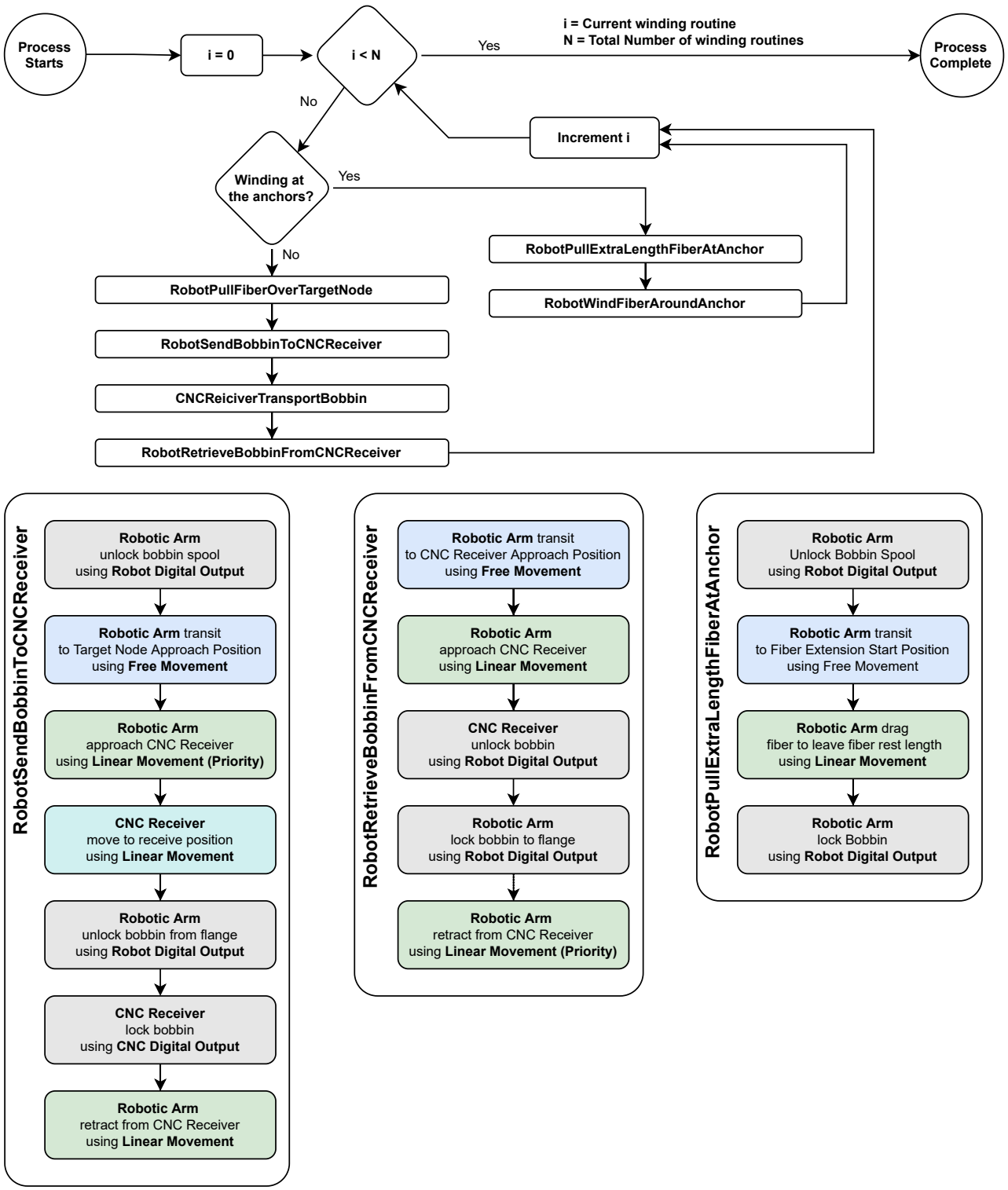


Figure 15: Construction process flowchart for spatial fiber winding based on process presented in [Estrada et al. 2020].