Design Validation and Conflict Resolution for Robotic Fabrication: A Multi-Stage Framework for Complex and Non-Repetitive Processes

Pok Yin Victor Leung*[1][0000−0003−1536−8636] and Yijiang Huang*[2][0000−0003−1820−2535]

* Both authors have contributed equally to this work.

Abstract. The use of robotics in construction offers transformative possibilities but also unique computational challenges, especially for bespoke and non-repetitive assembly processes. Fabrication constraints such as robotic reachability, material grasp stability, collision, and structural stability have to be validated before construction begins. However, validating all constraints is computationally expensive and the current approaches are linear black-boxes that simply run until they detect a conflict. When attempting to resolve the conflict, the current approach provides no indication of which variables to adjust and how to adjust them. This paper introduces a multi-stage constraint-validation and conflict-resolution approach as part of a computer-aided design process. The approach is developed, and its feasibility is validated by the robotic construction of a complex timber pavilion with numerous interdependent design variables. These variables include the shapes and position of individual timber elements, choices of structural joints, and robotic tools. Our design approach organizes these variables and their associated constraints into three conflict-resolution stages: (1) Assembly Design Stage - where real-time feedback supports creative exploration, (2) Process Design Stage - where fabrication decisions are made, and (3) Motion Planning Stage - where collision-free trajectories are computed. We introduce the concept of proxy checks to provide faster feedback and more interpretable results in each stage, enhancing the iterative design process. Our approach marks a step toward a more intelligent, computer-aided design system that can provide suggestions to resolve conflicts, advancing the field of computational design for robotic construction.

Keywords: Design Validation · Robotic Fabrication · Spatial Assembly · Design to Production · Fabricatability · Distributed Robotic Tools

1 Introduction

Design validation has a long history in computer-aided design and manufacturing, which uses computer simulation to predict and quantify the feasibility of a given design. In robotic fabrication, the main concerns are (1) whether a given robot system can perform all the assembly motions without collision, and (2) stability during construction, among many other constraints. The current method

to perform such assessments (design validation) is to plan the robotic tasks and motions (trajectories) with a detailed simulation. Depending on the complexity of the process (e.g. total number of motions, density of collision environment), Task and Motion Planning (TAMP) can be time-consuming, resulting in a slow design-evaluation cycle.

This paper focuses on the design method for a robotically assembled timber pavilion called *HyperHut* using an autonomous robotic process with Distributed Robotic Tools (DiRT) previously introduced in [14]. In this robotic process, 4 robotic screwdrivers (of 2 different types) and 3 pneumatic parallel grippers (of different lengths) were used in a non-repetitive sequence. We initially employed the flowchart method in [12] to plan the action sequence, which resulted in a long list of more than 1000 individual movements. However, we encountered difficulty when planning the trajectories according to this task sequence [1], where constraint violation require labourously adjusting design parameters (conflict resolution) and re-planning from the beginning. We found that a substantial amount of time was spent on adjusting the design because of the large number of computational steps between input (design decisions) and the output (generated motions). This makes it difficult for the designer to understand the input sensitivity (which parameter to adjust) and its gradient (how to adjust it) to resolve conflict. Therefore, each adjustment is not far from random and all the tasks and motions have to be recomputed for validation. Furthermore, the process involved motions in tight spaces, which further lengthened motion planning times.

The goal of this paper is to improve the computational design workflow, specifically to reduce the time (manual and computational) needed for design validation and to reduce the number of iterations needed for conflict resolution. Our contribution can be summerized with following approaches:

1. Grouping design parameters and constraints into three design stages, prioritizing faster computation to be performed at early stage.
2. Using proxy constraints (simplified rules) that are less accurate but faster to compute at early stage, to cull infeasible design decisions.
3. Calculating inverse kinematics (IK) as a quick reachability check.
4. Grouping robotic motions as Linear Motion Group (LMG) and Free Motion Group (FMG) and planning them separately in stage two and three.

By employing all these strategies, we found that it is possible to iterate the initial design to a buildable state at a much faster pace than by going through the full validation process every single time. The parameters and constraints shown in our case study are common to many robotic assembly processes. Thus, the staged adjustment approach, the grouping strategy, and the proxy constraints are expected to be applicable to a wide range of processes.

[1] Previous approach in [12] treats task and motion planning as two separate stages. In this paper, we treat task planning and motion planning as part of the design process, their result being the action sequence and trajectories are decisions that is part of the design.

1.1 Related work

Industrial robots have been used in several architectural projects to create bespoke spatial assemblies [5,7,9,19]. Many of these early works adopt a trial-and-error approach for designing the robotic actions and motions. Starting with an assembly design, this planning process involves manually assigning and adjusting (1) an action pattern (e.g. pick-transfer-place-transit pattern), (2) an assembly sequence,(3) robot targets and (4) joint configurations at keyframes. This process can be assisted by software packages that perform point-wise kinematic checks [1,16,20] and plan motion trajectory from configuration to configuration [6,18]. However, for complex assemblies, a poorly chosen action pattern or assembly sequence might lead to a "stuck" situation where the robot cannot assemble a workpiece because other pieces block it. When such a dead-end happens, a trial-and-error approach is typically used to find out which of the many parameters to adjust and how to adjust it. Furthermore, there is often a desire to adjust parameters that do not affect the appearance of the pre-conceived architectural design, which we refer to as process parameters (such as the choice of gripper or grasp poses). This conflict resolution process is crucial for the design of complex robotic processes, such as our case study, where motion planning dead-ends appear frequently. Yet, the literature on this topic is sparse. This paper aims to fill this gap by presenting a workflow for designing these structures.

Recent work on applying automated planning for robotic fabrication aims to alleviate designer's burden on resolving part of the robotic process parameters automatically, mostly assuming the architectural design is fixed. For processes with simple robot action patterns, such as pick-and-place assembly, specialized planning algorithms have been developed to plan the assembly sequence and motions for a single robot [11] and asynchronous motions of multiple assembly robots [3,8]. For generic robot fabrication processes, prior work has demonstrated PDDL-based task planning [17], flowchart-based task planning with automated motion planning [12], and integrated task and motion planning [15]. However, these automated planners can only provide a solution when the assembly task is feasible, but we often do not know that a priori and thus need an iterative process to update the inputs to the planner, i.e., architectural and process design variables, until we find a feasible plan. When a plan cannot be found, most planners today can only return a rather uninformative, cold "no plan is found" or "planning time limit reached" message after a lengthy computation. Designers are left to themselves to decompose or simplify the planning problem to get more granular information and to diagnose the problem. Our staged approach presents a systematic and computationally efficient framework to achieve this.

When it is relatively easy and quick to find a feasible robot plan, the planner can be used as a black-box performance evaluation subroutine in a larger optimization loop in a parameterized design space. Previous works derive score functions from measuring the structural behaviour during construction [10] or approximated assembly time of mobile robots [22]. However, for complex robotic processes, high-fidelity planning is needed to validate the design, which is too computationally expensive to be used in an optimization loop. Furthermore,

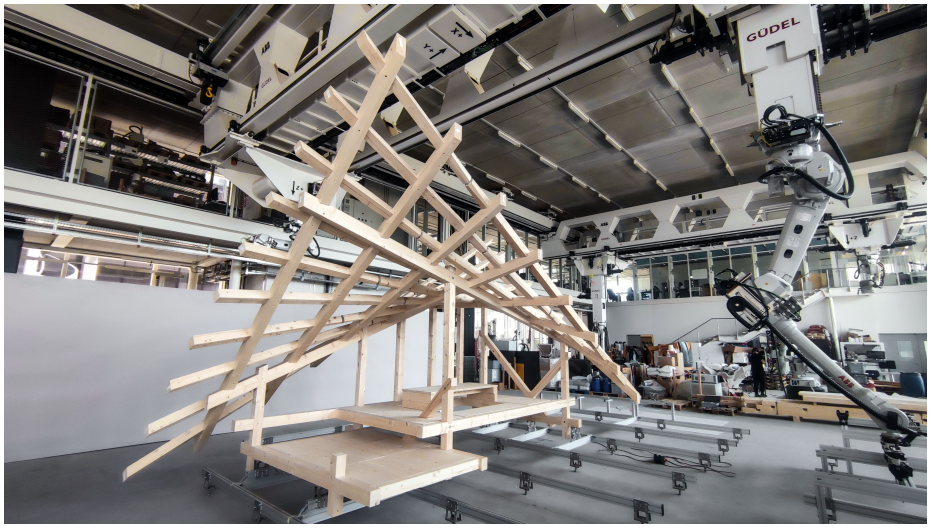

Fig. 1. Image of the HyparHut Pavilion after successful assembly under the robotic platform. The robotic arm and gantry used for construction can be seen.

since the planner is likely to fail, as noted in the last paragraph, this failure does not provide any useful signal, e.g., gradient, on how to nudge the design towards the feasible subspace. Although this work focuses on finding a constraint-satisfying design rather than optimization, our strategies enable the evaluation process to fail faster and to return more detailed conflict information, which has the potential to be integrated into an automated design optimization mechanism.

2 Methodology

2.1 Case Study

This paper is based on the case study project called *HyperHut*, named after its hyperbolic roof geometry. It consists of 57 timber elements that are assembled by a single 6 DOF robotic arm, invertedly mounted under a large 3 DOF gantry robot (Fig. 1). The combined setup has a working envelope that is larger than the whole pavilion, allowing the pavilion to be assembled in one continuous process as a simulation of an on-site assembly scenario. The robotic arm works collaboratively with 3 parallel grippers (Fig. 2a) and 4 distributed robotic screwdrivers (Fig. 2b). The role of the screwdrivers is to assemble the tight-fitting lap joints by pulling them together with a screw mechanism. In order to avoid jamming when closing multiple joints, multiple screwdrivers are operated synchronously via wireless communication with the robotic arm to complete the assembly.

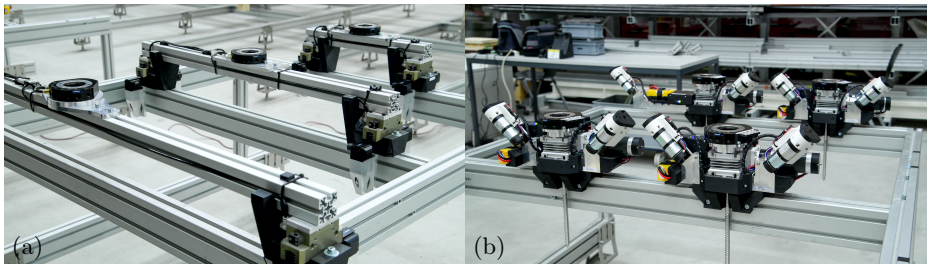

Fig. 2. Images of the robotic tools in their automated storage station (a) three robotic grippers, and (b) four robotic screwdrivers.

Fig. 3. Image showing the manual actions (a) fixing ground elements to the platform, (b) loading timber element onto the gripper, and (c) attaching screwdriver onto the timber element.

2.2 Task Planning with Flowchart

The assembly process follows a non-repetitive cycle by assembling one element at a time. Typically, the workflow includes (1) pick-up and attaching a gripper, (2) loading timber elements onto the gripper, (3) attaching screwdrivers to the timber element, (4) transferring the timber element to the assembly location, (5) synchronous joint closure, (6) retracting the gripper from the timber, (7) detaching the gripper and leave it at the tool-storage, (8) docking with a screwdriver on the structure, (9) retracting that screwdriver, and (10) detaching that screwdriver at the tool-storage. All actions are automated in the experiment and demo, with the exception of (2) and (3), which are performed by the operator (Figs. 3b and 3c).

An alternative assembly routine is used for elements connected to the ground (3 out of 57 elements), where the operator fixes the timber elements to a temporary platform (Fig. 3a). A third alternative routine was used, where a screwdriver was used (instead of a gripper) to hold and transfer the timber element (Fig. 4b). This mode is selected by the process designer on a case-by-case basis (34 out of 57 elements) to speed up the assembly process because it skips the steps to change to the parallel gripper. This mode also reduces the weight carried by the robot when multiple screwdrivers are attached to the timber element. However, this mode is only applicable when one of the joints is close to the middle of the timber element or the timber element is short, as shown in Fig. 4b.

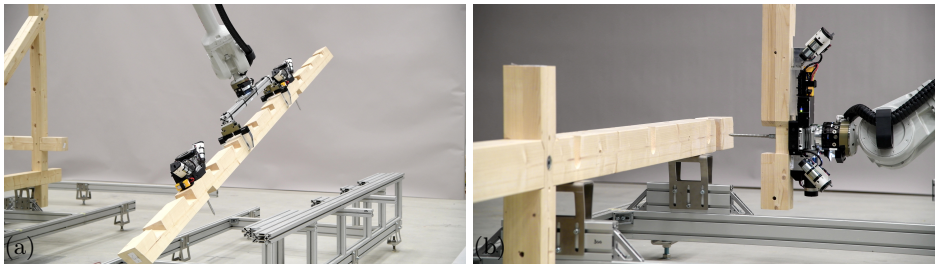

Fig. 4. Image showing the difference between using (a) a parallel gripper and (b) a screwdriver to hold the timber element during transfer and assembly.

The distribution of (1) how many joints are assembled for each timber element (hence how many screwdrivers are used), and (2) their gripper choices are shown in Table 1. This matrix of possibilities explains the origin of the non-repetitive action sequence. Note that (1) is determined by the arrangement of timber elements in the structure (architectural design), and (2) is chosen to fulfil engineering constraints (process design).

Table 1. Number of times each tool was used as a gripper for transferring a timber element, the occurrences are counted separately for elements with different numbers of simultaneously assembled joints. The total count adds up to 57 elements.

Gripper Choice	Ground Element	1 Joint	2 Joints	3 Joints	4 Joints
Parallel Gripper (500 mm)	0	2	1	0	0
Parallel Gripper (1000 mm)	3	2	8	0	0
Parallel Gripper (1500 mm)	0	0	5	2	0
Screwdriver	0	1	0	0	0
Screwdriver with long arm	0	1	16	11	5

We implemented the flowchart method introduced in [12] to create the non-repetitive action sequence (see Fig. 5) by analyzing the property of each timber element as shown in Table 1. The evaluation result, based on the final pavilion design with 57 elements, resulted in 811 high-level actions. If we expand these actions into their primitive movements (Fig. 5 shows part of the expansion process), there are 1087 Linear Movements (LM) (linear in Cartesian space) and 410 Free Movements (FM) (unconstrained joint-space movement).

2.3 Design Model Initialization

In this paper, 'design decisions' broadly encompass all aspects of the design including architectural geometry, spatial arrangement, structural joints, materials, and robotic assembly parameters." For example, one of the design decision is the

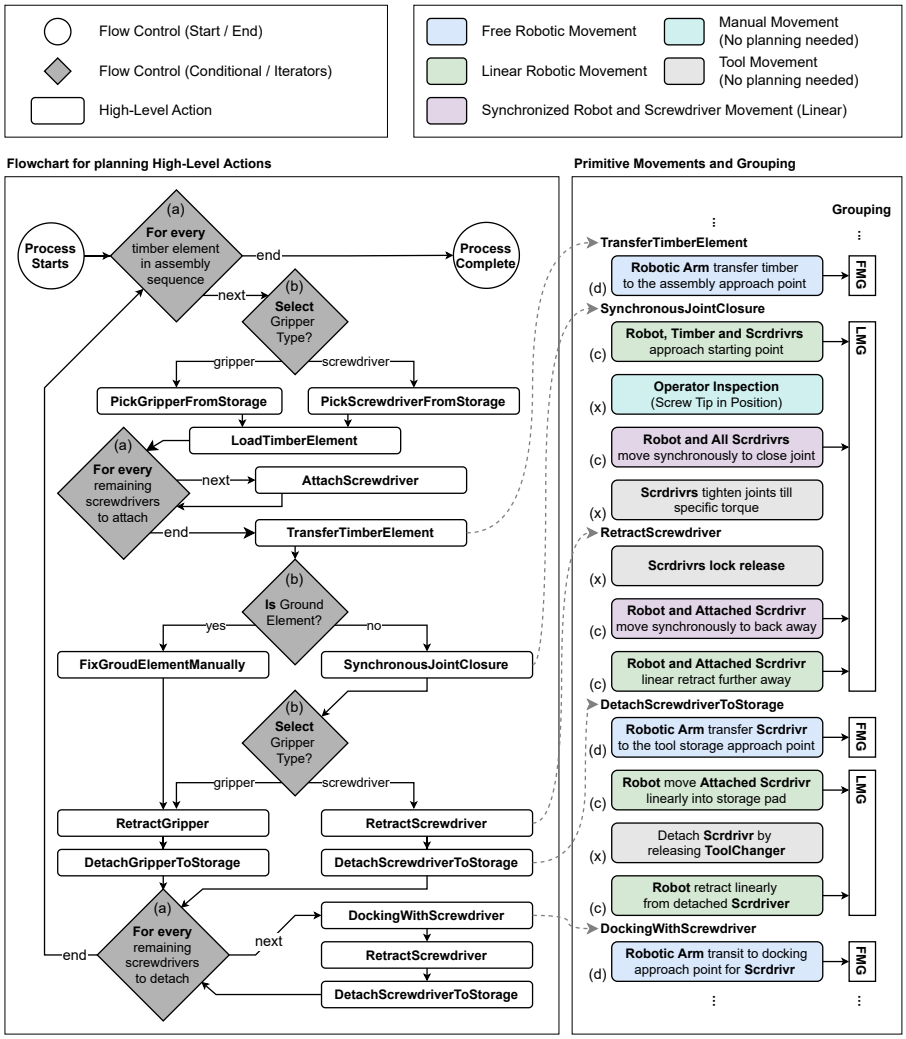

Fig. 5. Flowchart for planning the high-level action sequence showing (a) iterators and (b) conditional statements. The primitive actions are shown for five selected actions, revealing their constituent (c) LM, (d) FM, and (x) other movements that do not require planning. LM and FM are grouped for motion planning, which will be explained in Section 2.5.

movement of each timber element throughout the robotic assembly process. Due to the specificity of timber construction and the lack of existing precedence, we created a custom digital model as a complete representation of the design. The model includes parameters for **architectural design** decisions (such as the geometry and location of each timber element) and **process design** decisions (such

as assembly sequence, gripper choice and grasp pose for each element). These parameters are considered top-level decisions because the designers can modify them. In addition, the model also contains derived decisions (such as connectivity between elements) and sampler-based decisions (such as robot configurations computed from Inverse Kinematics). Both of these two types are computed from a function that takes input from other variables, with the difference that the derivation function is deterministic and the sampler is stochastic. A constraint function also takes input from other variables, but instead of generating a new value, it provides a binary certification of whether a constraint is satisfied. The detailed categorization of parameters and functions is shown in Fig. 7.

The designer initializes the model by converting an existing solid model (BReps created in Rhino3D) to a custom data structure that represents timber elements as rectangular boxes (width, depth and length) and their pose in space (global transformation). After the initial conversion, other top-level parameters are also initialized by filling them out with a default value. In some cases, the default value is generated automatically by a sampler function. For example, a gripper with a suitable size is selected based on the length of the timber element. While the automatic functions are designed to compute desirable default values, there is no guarantee of satisfying other constraints in the overall process. We rely on the designer to adjust these initial values in the conflict resolution step that will be described in Section 2.5.

One special modelling technique worth mentioning is related to how the grasp pose is selected. The grasp pose is a 3D frame (with 6 DOF) representing the affine transformation of the gripper relative to the timber element's origin. In order to ensure that the gripper pads have full contact with the sides of the timber element, a geometrically associative model was introduced as the sampler for the grasp pose (Fig. 6). It also introduces two new top-level parameters, 'face' and 'position'. Using this model, the designer can make straightforward adjustments to the face and position values while being certain that the gripper pads will satisfy the contact constraint. With this modelling, the grasp pose is no longer a top-level decision but a derived one. We used a simple rule to pick the top-facing face and a central position for their initial values, which minimizes the torque on the robot's wrist.

The relationships between decisions, sampler, derivation and constraint functions are represented as a graph in Fig. 7. The direction of data flow is represented using directed edges in the graph, which reveals the overall derivation order. The directional edges can be used (1) to trace potential upstream causes when a constraint is violated and (2) to identify downstream effects when an upstream decision is modified. For example, the latter helps identify which derived decisions have to be recomputed and which constraints have to be revalidated.

2.4 Conflict Detection

After parameters are initialized, the conflict resolution process begins. The goal is to check and resolve conflicts by modifying their upstream decisions. Conflicts can manifest in two different ways: (1) computed by the constraint-checking

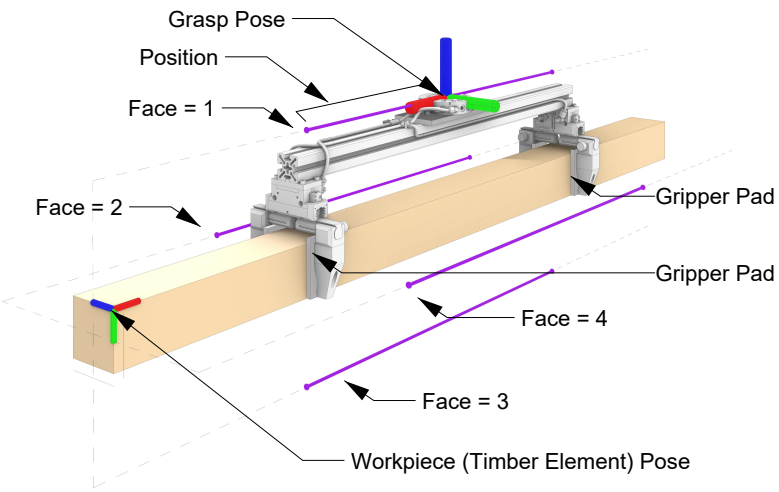

Fig. 6. The associative model within the grasp pose sampler. The input variables 'face' and 'position' are shown, the output of the sampler is the grasp pose relative to the workpiece pose.

functions, as mentioned before, and (2) by the failure of a sampling function to generate a value. Sampling failure is possible when the function cannot make a meaningful output when given the input values, for example:

- Gripper choice sampler may fail to find a suitable gripper when a timber element is too long or too short.
- Inverse kinematic sampler may fail when the target is out of reach for the robot.
- Motion Planner (MP) may fail to find a motion trajectory when the target is blocked by an obstacle.

Maximum Planning / Searching Time When validating with MP, the planning algorithm has to be given a reasonable planning time to conduct the search. Too short, and it may not find a solution (even though the solution may exist); Too long, and the wait is too long for the designer to consider that trajectory impossible and to start making adjustments. We empirically found a suitable value for our process, using 2 seconds for LM and 600 seconds for FM. Using an automatic backtracking and retrying method described in [12], we can plan all the actions (approximately 1000 LM and 400 FM) while maintaining configuration continuity. If successful, the whole process takes 6 to 10 hours to plan. Unfortunately, during the conflict resolution stage, the planner will continue to fail. In the beginning, the planning will fail relatively fast as the first problem is encountered. However, as the designer fixes the problem and tries again, the problematic moment will occur later, meaning that the planner will run for a

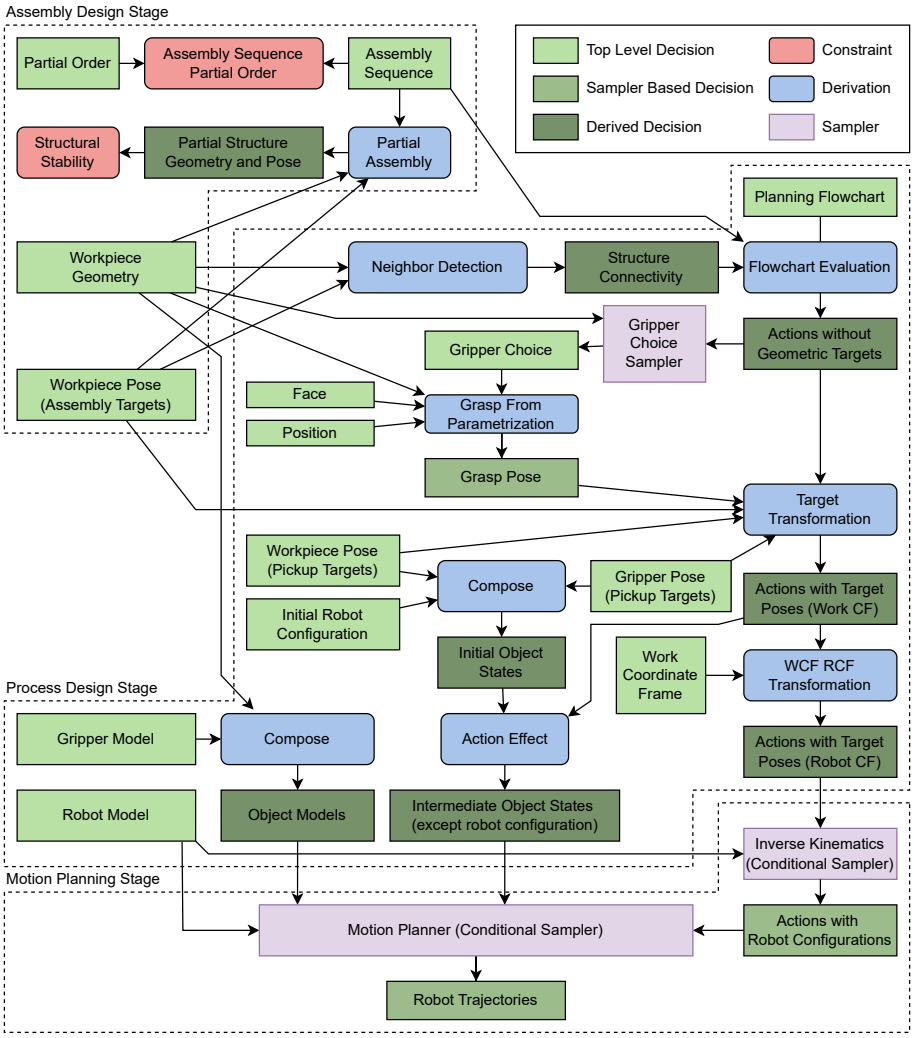

Fig. 7. Decision network represented as a graph, showing data flow between sampler, derivation and constraint functions.

long time before revealing the next problem. For the designer trying to resolve conflicts, it is very time-consuming to change one parameter each time to observe its effect.

Proxy Collision Check Through practice, we have found that it is better to introduce sanity checks at an earlier stage before MP. One important discovery is the introduction of the collision check (CC) for all the intermediate states at the start and end of each motion. Contrary to the common approach to performing

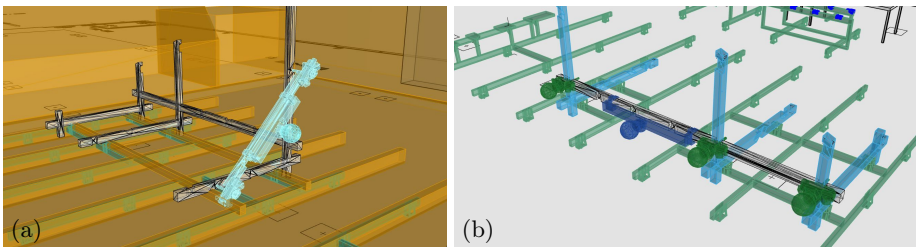

Fig. 8. Two examples of the disembodied collision check at different steps: (a) timber element with two screwdrivers approaching the assembly location, and (b) another timber element with three screwdrivers after joint closure. A sphere is used as a geometrical proxy for the robot wrist.

CC within the MP workflow, we performed CC before MP by excluding the robot body, which is still in an unknown configuration. We included all the other elements that have deterministic poses (the timber element, gripper, attached screwdrivers and a sphere as a proxy of the robot wrist) to perform CC with the stationary obstacles in the scene (Fig. 8). We called this sanity check a 'disembodied collision check' before the tool is detached from the robot. We also call this a 'proxy check' because it is a proxy of the actual MP as validation. Although this proxy check can only identify a subset of cases where MP would be infeasible, there are many benefits of this check:

- It is order-of-magnitude faster to compute than MP.
- It can be computed in deterministic time.
- It can be performed at an earlier stage, even before IK and other derived decisions are computed.

In practice, it takes less than one second to compute CC for each motion when computed in Rhino and less than 0.1 seconds when computed in PyBullet [4]. This means that the designer can receive near real-time feedback when inspecting the assembly process in a step-by-step manner in Rhino (with interactive 3D visualization) or receive a batch-checking report for all actions within a few minutes using PyBullet (without visualization).

2.5 Conflict Resolution

When a conflict is detected, the designer must decide which of its upstream decisions can be modified to resolve the situation. As mentioned before, the decision network can be used to identify top-level decisions or samplers that can be modified. However, since the MP is often the failing sampler, the number of upstream candidates is almost everything. This presents a challenge for the designer to decide which parameter to adjust and how to adjust it.

Decision Grouping by Design Stage From practice, we found that it is useful to group the design decisions, constraints and samplers into three design stages: Assembly Design Stage, Process Design Stage and Motion Planning Stage (grouped by dotted lines in Fig. 7). The designer follows a general guideline to resolve as much conflict as possible within one stage before progressing to the next stage. Backtracking to an earlier stage is done only when necessary because adjusting an earlier parameter often leads to new conflicts. This multi-stage iterative workflow is summarised in Fig. 9.

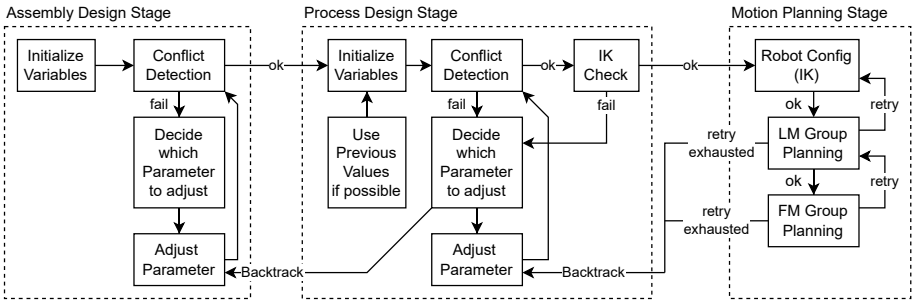

Fig. 9. Conflict resolution workflow with stage separation

Assembly Design Stage The first stage is mostly concerned with architectural and structural design decisions, such as the arrangement of timber elements. However, we also included the assembly sequence decision in this stage because it is often part of the consideration when designing a timber structure with integral timber joints. We perform (1) a "stability check" for the complete structure and all the partially assembled states, and (2) a disembodied assembly CC (a proxy check), which only includes the timber element's geometry in its final movement when the joints are closed. These checks are very fast to compute, allowing the designer to receive instant feedback while adjusting their design.

Process Design Validation The second stage is concerned with how the structure is constructed, such as the position of the structure relative to the robot, tool choice and grasp poses. We implemented the following checks within Rhino for an interactive design session, note that all CC involved in this stage do not include the robot:

- Grasp Stability Check
- Gripper Grasp CC for all motions while holding the timber element
- Screwdriver CC when it is attached to the timber element that is being assembled
- Screwdriver CC when the robot retrieves the screwdriver

– Tool storage CC for tool pickup and return motions

Finally, we also implemented an IK check at the end of this stage to check that all motion targets have IK solutions (confirming the targets to be collision-free and reachable). The collision detection was implemented using PyBullet for its better performance, allowing the designer to check IK for all motions (∼1400) within a few minutes. This provides the final confirmation that the process is ready for the next stage.

Motion Planning Validation The final validation of the robotic process is to plan the robotic motions (trajectories). This is the complete validation because the MP respects the following constraints, making sure (1) the planned trajectory stays within the the robot's kinematic limits, (2) all moving objects are free from collision with stationary objects, and (3) the discretized trajectory satisfies joint speed and acceleration limits. Using the non-linear planning sequence with automatic backtracking and retrying method introduced in [12], the whole process would take 6 to 10 hours to plan, hence we would often leave the planner to run overnight. Unfortunately, if one of the motions fails to plan, the rest of the motions cannot continue. This means that the overnight run can only help identify one problematic motion per nightly run.

Grouped Motion Planning In order to overcome this problem, we have improved upon the non-linear planning sequence by splitting the entire motion planning process into two parts. We employed a grouping strategy to group consecutive linear motions and free motions into what we called Linear Motion Groups (LMGs) and Free Motion Groups (FMGs) (see Fig. 5 for a grouping example). We first plan the LMGs, make adjustments if necessary, and finally plan the FMGs. In our process, the LMGs are short motion segments representing approach and retract motions, each containing 2 to 6 linear motions, while the FMGs are much longer motions that move the end effector from one location to another.

Because there are only two types of groups, LMGs and FMGs always happen alternatively. This means that each of the LMGs is separated by unconstrained free motion that can bridge large gaps if necessary. This allows us to plan each of the LMGs separately without worrying about continuity or dependency on each other. Within each LMG, the same non-linear ordering logic was used to overcome the problem caused by taught configurations [12].

The main benefit of this method is that even if one of the LMGs fails, the other groups can still be planned, allowing a single batch run to reveal all the problematic motions. Furthermore, because each group is rather small, the backtracking depth is limited, and therefore the total time to give up can be rather short. In our case study, we implemented Randomized Gradient Descent [21] to plan each LM and planning all the LMGs for the whole assembly process would only take ten to twenty minutes, which is short enough for the designer to make some parameter adjustments and wait to see its effect.

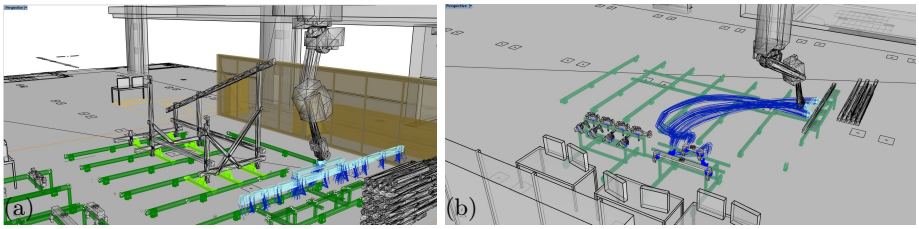

Fig. 10. Example of a successfully planned (a) Linear Motion and (b) Free Motion.

Once the LMGs are completely planned, the FMGs can be planned. The beginning and end configuration of the FMG are copied from the neighbouring LMGs to ensure continuity of joint configurations, and off-the-shelf motion planners, such as RRT-Connect [13], can be used to plan each FM. Note that FMGs can also be planned without dependency across the groups, meaning that parallel computing is possible. In our case study, all the FMGs can be planned when given enough time. However, for future work, if an FMG fails to plan due to its neighbouring LMGs are badly chosen, an automatic backtracking method could be implemented to re-plan its neighbouring LMGs and second-degree FMGs neighbours.

3 Results

Using the strategies described in the previous section, we successfully planned our case study structure by adjusting the initial design parameters until all constraints were fulfilled. The human designer provides important intuition during conflict resolution for selecting which parameters to adjust and how to adjust them. Thanks to the separation of design stages, the number of parameters to choose from is kept to a manageable number. Furthermore, the proxy checks in each stage make the evaluation time fast enough for an efficient adjustment process. The use of proxy collision checks at each design stage enables the final push for all the computationally difficult MP tasks to be accomplished automatically towards the end of the process. Table 2 provides an estimation of the number of times an adjustment-validation cycle was performed in each design stage. Without the strategies presented in this paper, each parameter adjustment would have to go through the entire planning process for validation. For hundreds of iterations, if each validation would take hours to compute, the whole design process would take an intractable amount of time.

4 Conclusion and Future Work

This paper contributes an iterative computational approach to validate and resolve conflicts for a design so that it can be robotically fabricated. We introduced

Table 2. Estimated number of adjustment-validation iterations in each design stages.

Design Stage	Validation Time (per iteration)	Iterations Performed
Assembly Design	Seconds for each element	Few Hundred
Process Design (No IK)	Seconds for each element	Few Hundred
Process Design (IK + CC in PyBullet Planner outside Rhino)	Few Minutes for the whole process	<50
Motion Planning (LMGs)	Tens of Minutes for the whole process	<20
Motion Planning (FMGs)	Five to ten hours for the whole process	<10

a hierarchical framework that groups decision parameters and feasibility checks into three stages and used fail-fast principles to enable faster and more localized design adjustment. This approach breaks down existing lengthy, black-box-style validation process into smaller pieces that are more interpretable and have a faster turnaround time. Thus, speeding up the overall design-to-production workflow. We demonstrated the feasibility of our approach by designing and constructing a complex robotic timber assembly process that involves tool changes and robot movement in a cluttered space.

Although our design validation approach can provide more granular information to the designer when a conflict arises (e.g., showing points of contact in a collision check), we emphasize that our framework differs significantly from classical 'wicked problems,' [2] which are characterized by ill-defined, subjective goals. In contrast, our problem context involves formally defined constraints that are more similar to constraint satisfaction problems. Nevertheless, our current system still relies on subjective human intuition to decide which parameter to adjust, how to adjust it, and when to backtrack to a previous design stage. Future work should aim to derive formalized models from this intuition-based conflict resolution to guide automated design systems more effectively.

As we build more capable robotic fabrication systems to respond to more complex design demands, planning becomes increasingly important as the interface between design and robot capabilities. We believe that this paper is the first step towards a more intelligent computer-assisted design future, where computers are not limited to validating designs but also diagnosing design problems and suggesting remedies.

References

1. Braumann, J., Brell-Cokcan, S.: 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). pp. 242–251. Banff (Alberta), Canada (2011). https://doi.org/10.52842/conf.acadia.2011.242
2. Buchanan, R.: Wicked problems in design thinking. Design issues **8**(2), 5–21 (1992)

3. Chen, J., Li, J., Huang, Y., Garrett, C., Sun, D., Fan, C., Hofmann, A., Mueller, C., Koenig, S., Williams, B.C.: Cooperative task and motion planning for multi-arm assembly systems (arXiv:2203.02475) (Mar 2022), http://arxiv.org/abs/2203.02475
4. Coumans, E., Bai, Y.: PyBullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning (2016), http://pybullet.org
5. Eversmann, P., Gramazio, F., Kohler, M.: Robotic prefabrication of timber structures: Towards automated large-scale spatial assembly. Construction Robotics **1**(1–4), 49–60 (2017). https://doi.org/10/gg55ds
6. Gandia, A., Parascho, S., Rust, R., Casas, G., Gramazio, F., Kohler, M.: Towards automatic path planning for robotically assembled spatial structures. In: Robotic Fabrication in Architecture, Art and Design. pp. 59–73. Springer (2018)
7. Hack, N., Lauer, W.V.: Mesh-mould: Robotically fabricated spatial meshes as reinforced concrete formwork. Architectural Design **84**(3), 44–53 (2014). https://doi.org/10/gg55fz
8. Hartmann, V.N., Orthey, A., Driess, D., Oguz, O.S., Toussaint, M.: Long-horizon multi-robot rearrangement planning for construction assembly. IEEE Transactions on Robotics **39**(1), 239–252 (Feb 2023). https://doi.org/10.1109/TRO.2022.3198020
9. Helm, V., Willmann, J., Thoma, A., Piškorec, L., Hack, N., Gramazio, F., Kohler, M.: Iridescence print: Robotically printed lightweight mesh structures. 3D Printing and Additive Manufacturing **2**(3), 117–122 (2015)
10. Huang, Y., Garrett, C., Mueller, C.: Constructability-driven design of frame structures with state-space search methods. Automation in Construction **167**, 105711 (Nov 2024). https://doi.org/10.1016/j.autcon.2024.105711
11. Huang, Y., Garrett, C.R., Ting, I., Parascho, S., Mueller, C.T.: Robotic additive construction of bar structures: Unified sequence and motion planning. Construction Robotics **5**(2), 115–130 (Jun 2021). https://doi.org/10.1007/s41693-021-00062-z
12. Huang, Y., Leung, P.Y.V., Garrett, C., Gramazio, F., Kohler, M., Mueller, C.: 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. pp. 1–17. ACM, Virtual Event USA (Oct 2021). https://doi.org/10.1145/3485114.3485122, https://dl.acm.org/doi/10.1145/3485114.3485122
13. Kuffner, J.J., LaValle, S.M.: RRT-connect: An efficient approach to single-query path planning. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065). vol. 2, pp. 995–1001. IEEE (2000)
14. Leung, P.Y.: DiRT: Distributed Robotic Tools for Spatial Timber Assembly with Integral Timber Joints. Doctoral Thesis, ETH Zurich (2023). https://doi.org/10.3929/ethz-b-000647367, https://www.research-collection.ethz.ch/handle/20.500.11850/647367
15. Leung, P.Y.V., Huang, Y., Garrett, C., Gramazio, F., Kohler, M.: Planning non-repetitive robotic assembly processes with task and motion planning (TAMP). In: Robotic Fabrication in Architecture, Art and Design (2024)
16. Schwartz, T.: 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, pp. 92–101. Springer (2012)
17. Sherkat, S., Skoury, L., Wortmann, A., Wortmann, T.: Artificial Intelligence Automated Task Planning for Fabrication. In: Dörfler, K., Knippers, J., Menges, A., Parascho, S., Pottmann, H., Wortmann, T. (eds.) Advances in Architectural

Geometry 2023, pp. 249–260. De Gruyter (Oct 2023). https://doi.org/10.1515/9783111162683-019
18. Sucan, I.A., Chitta, S.: Moveit! (2018), http://moveit.ros.org
19. Thoma, A., Adel, A., Helmreich, M., Wehrle, T., Gramazio, F., Kohler, M.: Robotic fabrication of bespoke timber frame modules. In: Willmann, J., Block, P., Hutter, M., Byrne, K., Schork, T. (eds.) Robotic Fabrication in Architecture, Art and Design 2018. pp. 447–458. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-319-92294-2_34
20. visose: Robots (Jun 2022), https://github.com/visose/Robots
21. Yao, Z., Gupta, K.: Path planning with general end-effector constraints. Robotics and Autonomous Systems **55**(4), 316–327 (2007). https://doi.org/10/dtd3m6
22. Zargar, S.H., Leicht, R.M., Wagner, A.R., Brown, N.C.: Integrating early assessment of robotic constructability into design optimization of a standalone classroom. Automation in Construction **157**, 105175 (Jan 2024). https://doi.org/10.1016/j.autcon.2023.105175