Automation in Construction 130 (2021) 103875

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

An integrated design tool for timber plate structures to generate joints geometry, fabrication toolpath, and robot trajectories

Nicolas Rogeau [a,b,*], Pierre Latteur [c], Yves Weinand [a,b]

[a] Laboratory for Timber Constructions (IBOIS), École Polytechnique Fédérale de Lausanne (EPFL), EPFL, ENAC, IIC, IBOIS, GC H2 711(Bâtiment GC), Station 18, CH-1015 Lausanne, Vaud, Switzerland

[b] The National Centre of Competence in Research (NCCR) Digital Fabrication, Swiss Federal Institute of Technology in Zurich (ETHZ), HIB E 25, Stefano-Franscini-Platz 1, CH-8093 Zurich, Zurich, Switzerland

[c] Institute of Mechanics, Materials and Civil Engineering (IMMC), Louvain School of Engineering (EPL), Université Catholique de Louvain, GCE, Place du Levant 1, 1348 Louvain-la-Neuve, Belgium

ARTICLE INFO

Keywords:
Integrated design
Timber plate structures
Wood joints
Digital fabrication
CNC machining
Robotic assembly
Robotic arm
Modular assembly
Assembly sequence
Insertion vector

ABSTRACT

This paper presents an integrated design tool for structures composed of engineered timber panels that are connected by traditional wood joints. Recent advances in computational architecture have permitted to automate the fabrication and assembly of such structures using Computer Numerical Control (CNC) machines and industrial robotic arms. While several large-scale demonstrators have been realized, most developed algorithms are closed-source or project-oriented. The lack of a general framework makes it difficult for architects, engineers and designers to effectively manipulate this innovative construction system. Therefore, this research aims at developing a holistic design tool targeting a wide range of architectural applications. Main achievements include: (1) a new data structure to deal with modular assemblies, (2) an analytical parametrization of the geometry of five timber joints, (3) a method to generate CNC toolpath while integrating fabrication constraints, and (4) a method to automatically compute robot trajectories for a given stack of timber plates.

1. Introduction

1.1. From traditional timber joints to Integrally Attached Timber Plate Structures (IATPS)

Early wood architecture traces have revealed timber joints dating back more than 7000 years ago [1]. Until the industrial revolution, it remained the most common technique to assemble wooden pieces. Two main structural purposes of timber joints can be distinguished: increasing the length of the elements to achieve longer spans (e.g., roof framing, bridge trusses) or increasing the width of the elements to cover a larger surface (e.g., stacked walls, roof cladding). For each situation, various joining techniques adapted to the local context were developed and transmitted by carpenters over time [2].

With the spread of metallic connectors (e.g. screws, bolts…) which can be easily mass-produced, and the rising cost of skilled labor in developed countries, timber joints have gradually been abandoned by construction companies in favor of less expensive practices that could be more easily automated [3]. New gluing processes have also made it possible to overcome the natural size of trees. Indeed, engineered wood products such as glued laminated timber (glulam) and cross-laminated timber (CLT) can cover large spans and wide surfaces with a single element whose dimensions are limited only by logistical constraints. With the industrialization of the construction sector, timber joints craft was therefore narrowed down mainly to vernacular architecture and cabinet making.

However, the recent emergence of robotics in the construction sector has made it possible to revive ancient techniques such as traditional timber joinery. Computer Numerical Control machines (CNC) and Industrial Robotic Arms (IRA) allow for automating complex fabrication and assembly processes that would otherwise be too time-consuming for cost-effective applications. This opportunity led to the development of so-called Integrally Attached Timber Plate Structures (IATPS) [4]. IATPS combine modern engineered wood panels with traditional timber joints that are digitally fabricated. Beyond their aesthetic appeal, these joints also fulfill a functional role as they allow each construction element to be precisely positioned with each other. They also reduce the need for additional connectors such as screws or nails as the assembly is

* Corresponding author at: GC H2 711 Station 18, 1015 Lausanne, Switzerland.
E-mail address: nicolas.rogeau@epfl.ch (N. Rogeau).

https://doi.org/10.1016/j.autcon.2021.103875
Received 27 May 2021; Received in revised form 26 July 2021; Accepted 2 August 2021
Available online 12 August 2021

performed solely by geometrical interlocking. Furthermore, experimental studies and numerical analysis of IATPS have demonstrated the positive influence of wood-wood connections on the structural performance of the system [5].

The main interest of IATPS lies in the fact that they allow a high degree of prefabrication. However, designing such structures is not straightforward as multiple constraints influence the shape of the connections between the panels. Several workflows have been showcased through the construction of demonstrators and real-scale flagship projects (Section 1.2) and several computational tools have been developed to assist in the 3D modeling of timber joints (Section 1.3). However, there is a lack of a general framework linking all constraints of the system together to allow non-experts designing IATPS and enable broader applications of the system. A general framework for the structural design of IATPS has been proposed by Rad and al. [6,7]. This contribution aims at completing this framework by integrating fabrication and assembly constraints in the 3D modeling of the connections.

1.2. Existing computational workflows for IATPS

IATPS have been showcased in several research pavilions and flagship buildings during the last decade (Fig. 1). Several large-scale demonstrators featuring finger joints have been built by researchers from the University of Stuttgart [8,9]. For the recent BUGA Wood pavilion [10], so-called "co-design" algorithms were developed to integrate structural requirements and robotic fabrication aspects into the design process. High-level of automation was reached as each unique module of the structure was carefully assembled by two collaborative robots.

Recent projects from the Digital Timber Construction chair of Kaiserslautern University have used wedge-joints [11] and butterfly joints [12,13] between hexagonal pieces to build doubly-curved timber vaults. For each project, global geometry and local connection details were controlled by custom scripts. Machining toolpath has also been automatically generated and simulated before exporting fabrication files to a 5-axis CNC.

Previous research conducted at the Laboratory for Timber Constructions at Ecole Polytechnique Fédérale de Lausanne has led to the construction of two full-scale projects: the theater of Vidy [14] in Lausanne and Annen head office in Manternach [15]. The first project is a double-layered folded structure inspired by Japanese origami patterns and the second consists of 23 doubly-curved vaults made out of timber boxes. Through-tenon joints were used in both structures as connections between timber panels. As for the pavilions mentioned above, parametric scripts were developed to integrate fabrication constraints in the design process.

For all projects presented above, the global shape was first discretized using a mesh where each mesh face represents a timber panel. A mesh data structure keeps track of the links between the elements. This adjacency information is essential for modeling joints between two timber plates as the shape of the connection is influenced by the relative position of the plates. As timber panels are planar elements, all mesh faces also need to be planar. Planarization algorithms are used to approximate curved surfaces with planar panels [16,17]. In addition, as timber panels are standardized products, it is often required to work with panels of constant thickness. Ensuring both planarity and constant thickness heavily constrains the design space to certain types of meshes. Hexagonal patterns with trivalent vertices (Fig. 2a) allow meeting both requirements [18].

However, mesh discretization is not the only possibility for modeling IATPS. One alternative is to rely on procedural generation algorithms such as the one developed by Rossi and al. [19]. Complex assemblies can be created from aggregation rules operating on a predefined set of tiles (Fig. 2b). Aggregation is the inverse operation of discretization. Instead of dividing a complex geometry into panels, panels are assembled iteratively to form a complex structure based on topologic constraints. As for meshes, aggregation methods produce an organized data structure that contains the adjacency information necessary to model the joints.

Another possibility is to manually draw the elements in 3D without relying on specific data structures (Fig. 2c.). While aggregation and discretization are powerful tools to tackle complex geometries, most architectural projects are drawn using standard CAD software without using any computational methods. In that case, the only information contained in the model is the plate geometry and its position in an arbitrarily defined frame. Therefore, automating the 3d modeling of the connections for such structures requires first to determine the relative position of the plates to each other.

The integrated design approach employed in all the projects cited above was based on a predefined data structure that facilitated the modeling of connections between elements and the integration of fabrication and assembly constraints. The drawback is that each architectural project is different and therefore, a unique algorithm has to be thought for each new construction system. This research aims to develop a holistic design tool to generalize joinery modeling between timber plates independently of the global geometry of a project.

1.3. Existing computational design tools for timber joints

The resurgence of timber joints in contemporary architecture has been facilitated by the development of 3D modeling tools to automatically generate geometry and fabrication files for these types of connections. Several Computer-Aided Design and Manufacturing (CAD/CAM) software such as CadWork® [20], Sema [21], and Lignocam [22] integrate dedicated joinery modules for timber frame structures and can export machining toolpath from 3D models. Similarly, open access web applications such as MakerCase [23], CutCad [24], Kyub [25], and Joinery [26] allow readily creating joints between planar pieces which can then be cut and assembled in 3D. Another approach based on voxels was taken by Larsson and al. [27] to develop Tsugite, an interactive interface to explore, design, and manufacture complex joints between timber beams. By restricting the design space to a 9x9x9 voxel grid, a multi-criteria analysis of the joint performance can be shown to the user after each iteration and the design can be accordingly improved.

Fig. 1. Three large-scale projects showcasing IATPS (from left to right): The folded structure of the Vidy theater (IBOIS, 2018) [14], the segmented shell of the Buga Wood Pavilion (ICD/ITKE, 2019) [10], the double-layered vault of the Annen head office (IBOIS, 2021) [15].

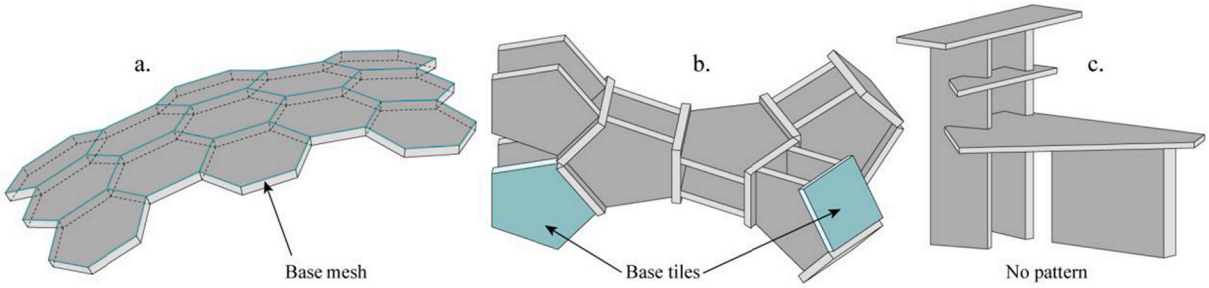


Fig. 2. Different methods for 3D modeling IATPS: using a mesh data structure (a), using aggregation rules for a set of two tiles (b), using standard CAD software without a specific data structure (c).

Custom plugins for Grasshopper, the visual scripting interface of Rhinoceros by McNeel and Associates [28], have also been developed. Timber Plate Structure plugin (TPS) [29] gives users the possibility to create origami patterns with dovetails, finger joints, or miter joints. Reindeer [30] open-source set of tools focuses on timber frame connections with structural analysis feedback. GluLamb [31] addresses the specific case of curved glued laminated timber (GLT) pieces. Finally, Emarf [32] offers various tools to generate timber joints between extruded solids and export fabrication files. The main advantage of those parametric solvers is the direct feedback that makes it possible to adapt the joint geometry according to fabrication constraints. Typically, machining toolpath can be visualized without leaving the design interface and its impact on the global project can therefore be considered ahead of execution.

Assembly constraints can also inform the shape of a joint. Most solvers leave it up to the user to assess the feasibility of the assembly as they are tailored for standard applications. When complexity rises, two strategies can be employed: (1) solving assembly constraints for a given set of joints and refining the design until the assembly works or (2) constraining the design space to generate only compatible joints. Tsugite solver belongs to the first category as it evaluates friction areas to assess the ease of insertion of each proposed joint. On the contrary, the TPS plugin computes a compatible vector of insertion for each piece in the structure before generating the joints to avoid any blocking situation.

The interlocking properties of timber joints can also be exploited in the design of a structure. DESIA framework [33] uses directed graphs to compute assembly sequences and vectors of insertion for a set of discrete elements where the last piece acts as a blocking key. Graph theory was also applied to compute assembly sequences of waffle structures composed of intersecting planar pieces connected by half-lap joints [34].

This research aims to provide a computational design tool for timber plate structures that may have a large number of connections. Therefore, fabrication and assembly constraints have to be treated automatically and cannot be left under the user's responsibility. The main challenge was to provide a high level of automation while keeping a design space as large as possible to accommodate different structural typologies: from structures with a regular pattern (shell, vaults, slabs…) to more irregular assemblies. By concentrating the research scope on timber plates, a systematic methodology could be employed to cover all topological cases.

The developed tool is introduced in the following sections. The implementation of the solver and its interface are presented in Section 2. Required inputs and working hypotheses are then detailed in Section 3. The algorithmic framework allowing the computation of compatible insertion vectors is explained in Section 4. The generation of the solver outputs (joints geometry, fabrication toolpath, and robotic trajectories) is covered in Section 5. Finally, the performance of the solver is demonstrated in Section 6 through three case studies.

2. Solver implementation and user interface

The main concept of this new integrated design tool is to convert a 3D model into an organized data structure allowing to generate joints geometry, fabrication toolpath, and robotic trajectories. Before detailing the different parts of the algorithm, its general implementation and its interface are here presented.

2.1. Implementation of a new data structure

The solver has been coded in Python 2.7 [35] and relies on the RhinoCommon [36] library for geometric operations. Concretely, the code is structured as four python classes:

- *Plate model*: methods and attributes applying to the full structure
- *Plate module*: methods and attributes applying to a group of plates
- *Plate*: methods and attributes applying to one single plate
- *Toolbox*: helper functions

Taking advantage of the concept of object-oriented programming (OOP) [37], a 3D model is converted into an instance of the plate model class. For each plate in the 3D model, an instance of the plate class is also created. Similarly, plate modules are instantiated for each group of plates specified by the user. This will be more detailed in Section 3.2 about the modular construction approach.

By converting 3D geometries into class instances, additional information can be attached to the elements of the structure. For example, a plate instance carries more data than just the plate geometry. The thickness of the plate, the plane in which the plate stands, and other specific attributes are all computed during the instantiation of the object and stored as variables that can be later accessed. Besides, those elements can be grouped in different hierarchic levels (Module, Model) to store topologic data about how the elements are connected.

Adding this layer of information during the conversion of the 3D model into a class instance also allows faster feedback when performing operations on the model. Typically, a modification of the joints geometry can be executed without the need to compute again all topologic information as it is already stored in the data structure.

2.2. User interface for iterative design workflows

To facilitate the manipulation of the integrated design tool by architects and structural designers, a custom plugin has been developed in the visual scripting environment of Grasshopper. This node-based interface makes the tool usable without any programming knowledge. Class methods and attributes can be accessed through different Grasshopper components. A parametric design workflow can be easily set up to generate different types of joints in an iterative process. This is further demonstrated in Section 6 through three case studies.

3. Solver inputs

The solver requires three types of inputs: a 3D model containing all the elements of the structure, an assembly sequence defining the order in which to assemble the different parts, and insertion constraints depending on the type of joints to generate. Each solver input is here described.

3.1. Hypothesis about the initial 3D model

To run the solver, a 3D model of the global geometry of a timber plate structure first needs to be provided. At this stage, joints are not modeled yet. The 3D model is a collection of timber plates that can be manually drawn or algorithmically generated in Rhinoceros CAD software. A plate is defined as a planar structural element with a small thickness compared to the planar dimensions [38]. It is geometrically represented as a collection of connected surfaces forming a closed polyhedron (see Fig. 3). Besides, a plate must follow the following requirements: (1) all surfaces composing the plate should be planar, (2) all vertices should be trivalent (having exactly three neighbors), and (3) each vertex of the top face should be directly connected to exactly one vertex of the bottom face and conversely. Plates with internal holes are included in this definition while those with curved edges or warped faces are excluded.

The type of joint which can be generated between two plates is conditioned by their relative position. Five topologic cases are supported by the solver (Fig. 4). The contact zone is defined as the surface (or the volume for intersecting plates) shared by two adjacent plates. Edgewise connections are a particular type of side-to-face connections with the contact zone being on the plate edge. For each relative position, a type of joint has been studied and parametrized (see Section 5.1). To ensure the 3D model can be properly interpreted by the solver, each pair of adjacent plates must respect one of the five contact types.

3.2. Assembly sequence and modular construction approach

In addition to the 3D model, the user is asked to provide an assembly sequence defining the order of assembly of the elements. Some solvers such as DESIA [33] can automatically compute an assembly sequence for a given set of elements. However, for real construction projects, assembly steps are often constrained by other factors than the geometry of the pieces. Besides, large-scale structures are often subdivided into smaller modules that can be preassembled off-site. The range of possibilities for a modular assembly sequence is extremely wide as plates can be grouped in a lot of different manners. To avoid constraining the solver to linear assemblies and enable the use of modules, the assembly sequence is here considered as a design variable under the responsibility of the user. The sequence input is written as a list of integers using Python syntax (Fig. 5). Each integer refers to a plate in the list of elements of the 3D model. To change the order of the assembly, the user can either change the order of the collection of plates, either swap integers in the assembly sequence.

3.3. Insertion constraints for timber joints

The goal of the solver is to generate timber joints that are compatible with the assembly sequence. To do so, the direction of the assembly has to be predetermined before generating the joints geometry. Otherwise, non-compatible joints could lead to blocking situations where pieces cannot be assembled (Fig. 6). To summarize, the shape of a connection is induced by the type of joint, by the assembly sequence, and by the adjacent plates. The algorithms leading to the determination of the assembly direction are detailed in Section 4.

Timber joints can be inserted in one or more directions depending on their geometry. The direction of the assembly is represented by an insertion vector while the insertion domain shows all possible insertion vectors for a given joint (Fig. 7a). Before a joint has been created between two plates, the assembly can potentially be performed from all directions. Therefore, the largest insertion domain is a hemisphere oriented to the normal of the contact zone (Fig. 7b). However, most types of joints cannot be adapted to work for any insertion vector. To ensure the generation of the joint geometry remains possible, it is necessary to constraint the insertion domain to a portion of that sphere before computing the insertion vectors (Fig. 7c). This is a way of informing the solver about the type of joint that will later be created.

Mathematically, an insertion constraint is defined as the locus of all the points on the unit sphere from which an insertion vector can be created by linking the sphere center. The locus can be tridimensional (a piece of the unit sphere), bidimensional (an arc on a unit circle) or unidimensional (a point on the surface of the sphere). The solver supports five contact types (Fig. 4). Per default, it provides a different insertion constraint for each of them (Fig. 8). Custom constraints can also be specified by the user via the Grasshopper plugin interface.

4. Solver algorithms

This section describes all the necessary steps to transform the 3D model into an instance of the “Model” class (see Section 2.1) using the inputs provided by the user. The topology of the structure is first computed by looking at the contacts between the plates. Then, the assembly sequence is parsed to extract the different assembly steps. Finally, insertion vectors are computed for each plate and module in the structure.

4.1. Computing adjacency information from plate contacts

The 3D model input only provides geometries without any attached data. Conversely, a mesh data structure stores information about the adjacency of its elements. Therefore, the first step of the algorithmic framework is to find all contact zones where a joint could be created. This is achieved by intersecting the plates with each other. If the intersection returns neither a surface nor a volume, the pair is discarded. Otherwise, the contact type is determined by comparing the orientation of the normal to each neighboring plate with the normal of the contact zone (Fig. 9). A plane is also generated at the center of the contact zone.

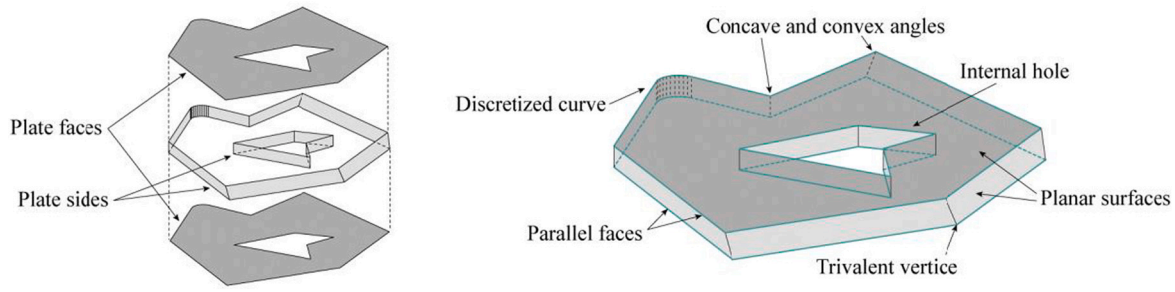


Fig. 3. Plate faces refer to the two largest surfaces of the poly-surface while other surfaces are designated as sides.

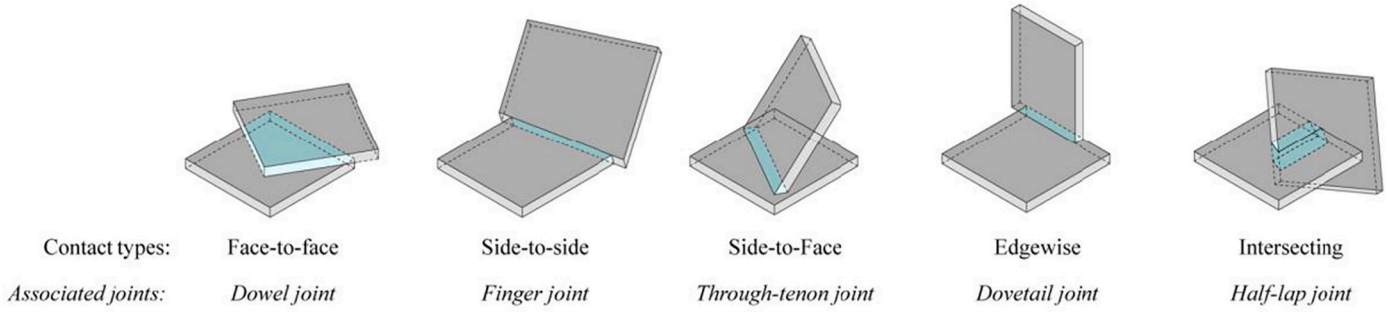


Fig. 4. Possible contact types for an assembly of two plates and their associated timber joints.

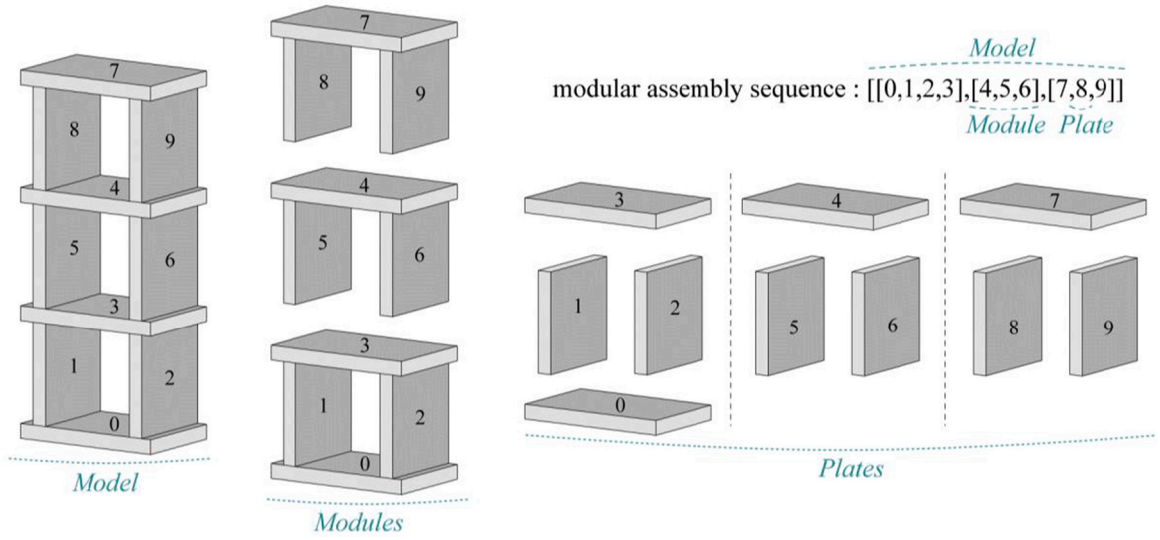


Fig. 5. A modular assembly sequence of timber plates expressed as a list of lists of integers.

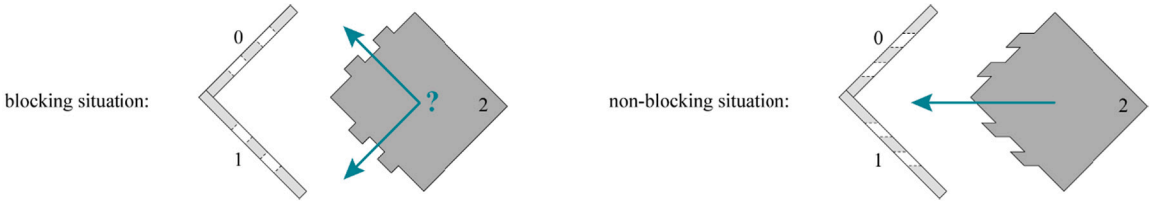


Fig. 6. Generating joints without considering the assembly sequence can lead to blocking situations (a). The goal of the solver is to first compute a compatible direction of assembly before generating joints geometry (b).

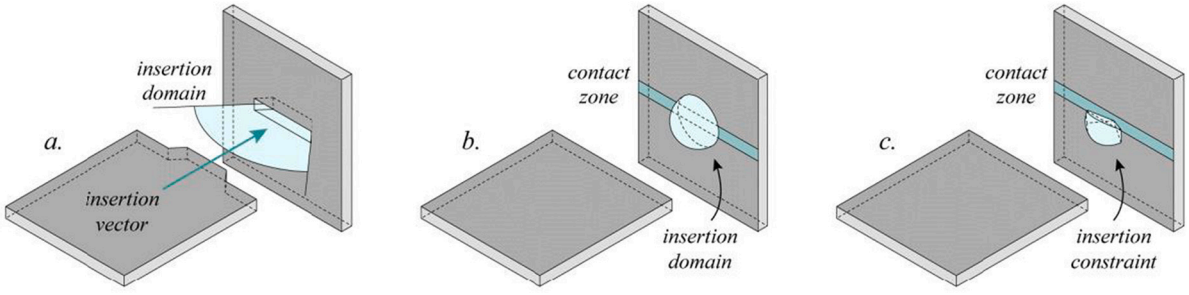


Fig. 7. Each timber joint has an insertion domain that represents all possible insertion vectors for the assembly of the pieces (a). Without joint, the insertion domain is represented by a half-sphere (b). Insertion constraints ensure joints can be generated once the insertion vector has been deduced.

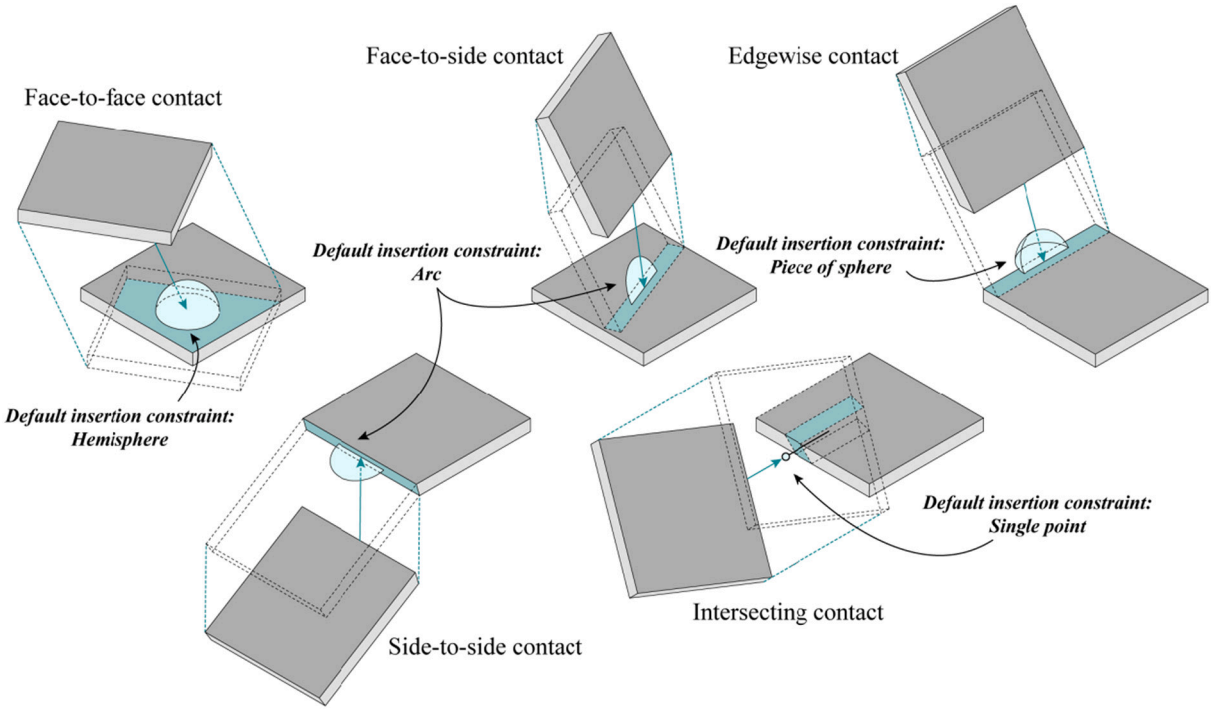


Fig. 8. The solver provides a default insertion constraint for each contact type.

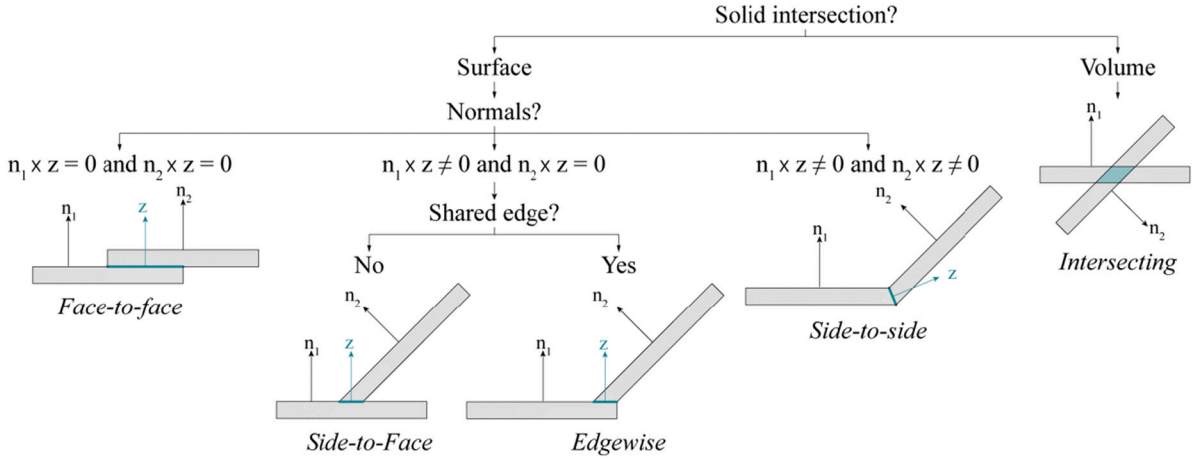


Fig. 9. Decision tree of the algorithm used to identify the different contact types.

It will be the frame of reference for the joint that will later be created between both plates.

Adjacency information can be represented using a graph where each node of the graph is a plate and each link represents a contact between two plates (Fig. 10, left). In the model instance, this information is stored using a list of pairs of integers based on plate numbers (Fig. 10, right). Parallel lists are used to store the contact type, the contact zone, and the contact plane for each pair of plates. In addition, an insertion constraint is selected in function of the contact type and oriented according to the contact plane.

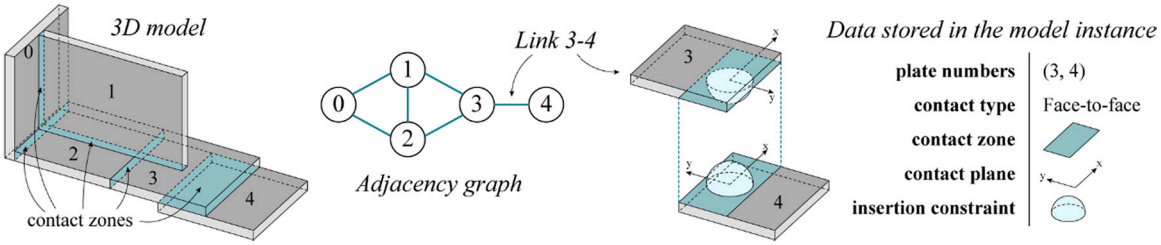


Fig. 10. Plate contacts can be represented with an adjacency graph where each link represents a contact between two plates. Contact types, contact zones, contact planes, and insertion constraints are stored in parallel lists inside the model instance.

4.2. Computing assembly steps from the assembly sequence

The second step of the algorithmic framework is to parse the assembly sequence input. Each pair of square brackets implies the instantiation of a module. A module is a pre-assembled group of plates that moves as one single object. Modules and plates can be combined to form larger modules. In the example of Fig. 11, the module "[1,2,–3]" is associated with the plates "0" and "4" to form the module "[0, [1,2,3],4]". This module is then assembled with the module "[[5,[6,7]], [8,9]]" to complete the model. As for a simple plate, an insertion vector needs to be associated with each module to specify the assembly direction. However, at this stage, insertion vectors are not yet determined.

The assembly sequence is successively divided into different subsequences that define the order of assembly of the elements constituting each module. A tree graph can be used to represent the development of the assembly sequence associated with the model into the subsequences associated with all modules (Fig. 11, top left). Since a plate can be part of several modules, smaller modules need to be assembled first. Therefore, the order of the assembly steps is obtained by navigating the tree from bottom to top (Fig. 11, top right). Consequently, in the example of Fig. 11, the first module to assemble is "[6,7]". Modules that are on the same level of the tree graph are assembled from left to right respecting the ascending order of the plate integers. Hence, the next module to assemble is "[1,2,3]" followed by "[5,[6,7]]" and "[8,9]". Several steps can be required to assemble a module. For the module "[1,2,3]", plate "2" is first inserted into plate "1". Then, plate "3" is inserted into plate "2". A total of nine steps are necessary to complete the assembly of the input geometry of Fig. 11.

4.3. Computing compatible insertion vectors for each plate and module

At this stage, contact types and assembly steps have been retrieved from user inputs. The purpose of this third part of the algorithmic framework is to compute insertion vectors for all the elements (plates and modules) of the model at each assembly step. With modular assemblies, a single element can be part of several subsequences. Therefore, while each contact zone needs to be associated with only one insertion vector to generate a joint, the contact zones of one plate can have different vectors of insertion (like C_1 and C_2 in Fig. 12). However, if two contact zones are involved in the same assembly step (like C_2 and C_3 in Fig. 12), the same insertion vector needs to be applied.

By representing the assembly sequence as a graph, adjacency and assembly information can be compared. To find the insertion vector associated with an assembly step, all contacts between the plate(s) to assemble and the plate(s) appearing before it in the subsequence are considered. In the example of Fig. 12, the first subsequence only implies plates 3 and 4. Therefore, only one contact will be involved in the determination of the insertion vector. On the contrary, for the second subsequence, the module "[3,4]" has four contacts reported on the adjacency graph with plates 0, 1, and 2. As those plate numbers appear before "[3,4]" in the subsequence, a common insertion vector will be generated for the four contacts.

Insertion constraints are used to compute an insertion vector for all contact zones involved in an assembly step. Four different cases can be considered depending on whether the element to be assembled is a plate or a module and the number of adjacent plates. (Fig. 13). When assembling a plate that has only one contact with one adjacent plate (Fig. 13, top left), the insertion vector is directly found by taking the average point on the domain of the insertion constraint. When assembling a plate that has more than one contact (Fig. 13, bottom left), the vector is found by intersecting the domains of all the insertion constraints associated with those contacts. A similar method is described in chapter 4 of Christopher Robeller's thesis [4]. This method is here extended to modular assemblies. When a module is inserted in the structure (Fig. 13, top and bottom right), all the insertion constraints associated with the contacts between the plates of the module and the plate(s) already in place are regrouped and intersected to compute a compatible insertion vector. The intersection process has also been optimized compared to the initial method as it does not rely on solid intersections but only on faster point-point, point-curve and point-surface intersections. Finally, if the intersection of the constraints returns no result for one assembly step, the solver asks the user to modify one of the solver inputs.

5. Solver outputs

Once an insertion vector has been computed for each assembly step and associated with each contact zone, joints can finally be generated. In this section, the parametrization of five different joints (one for each contact type) is presented. The generation of CNC toolpath and robotic trajectories to automate the fabrication and the assembly of the pieces is also detailed.

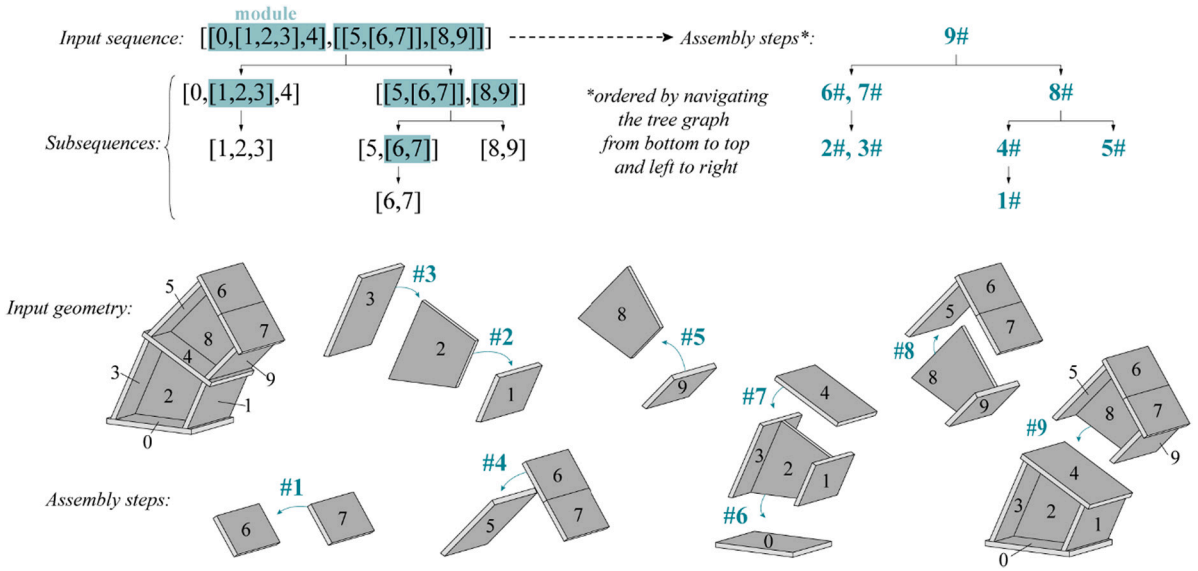


Fig. 11. Developing all subsequences from the input sequence allows ordering the assembly steps.

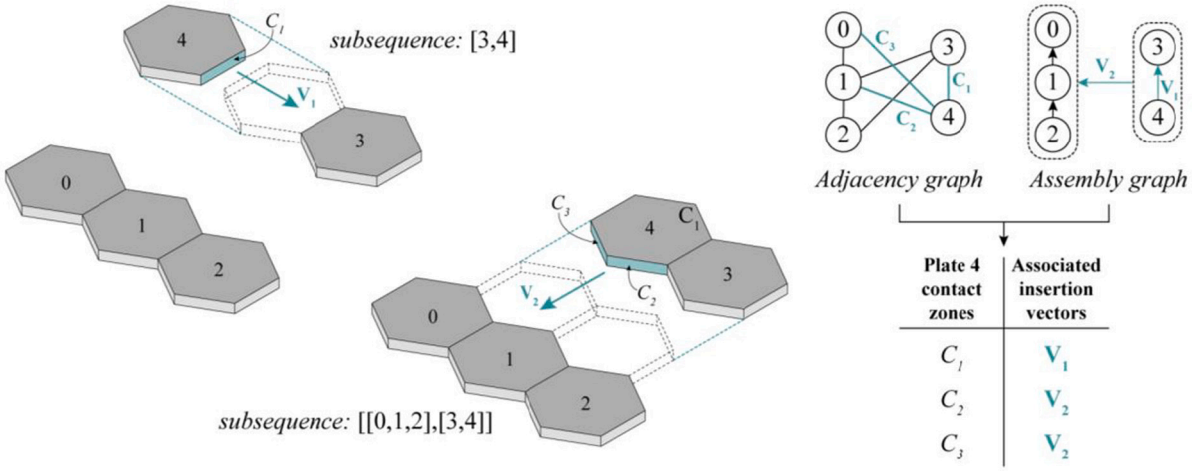


Fig. 12. In this example, plate 4 has three contact zones (C_1, C_2, C_3) and two insertion vectors (V_1 and V_2) since it is involved in two subsequences. The aim of the algorithm is to associate one insertion vector to each contact zone by crossing adjacency and assembly information.

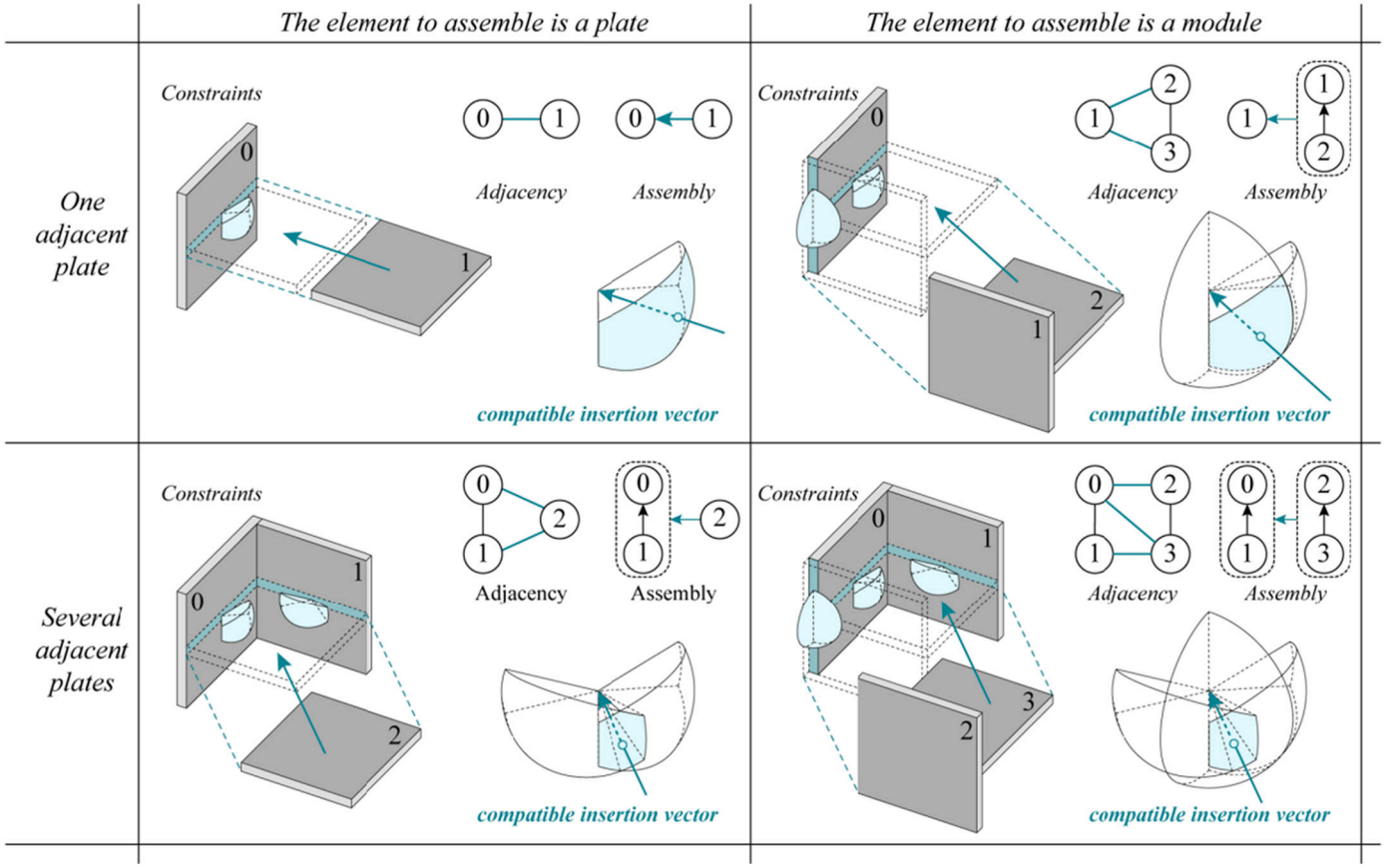


Fig. 13. A compatible insertion vector is found by intersecting the insertion constraints from the contact zones between the plate(s) to assemble and the plate(s) already in place.

5.1. Generating joints geometry

For each contact type, one timber joint has been studied and parametrized (Fig. 14). Those joints have been selected among others for their common use in timber construction or for their geometric and aesthetic interest. However, other joints could easily be parametrized by following a similar logic to the one presented here. The shape of each joint can be controlled by a set of user-defined parameters. For example, for the dowel joint, the number of dowels, their radius, and inclination can be adjusted. Similarly, for the through-tenon joint, parameters allow modifying the number of tenons as well as their dimensions and spacing.

The objective was to provide a purely analytical description of the shape of each joint to remain independent of any computational library or programming language. The parametric equations of all the points necessary to create four of those five joints are given in the appendix. The half-lap joint is here the only exception. Its geometric construction requires the use of solid Boolean operations that prevented a purely mathematical approach. For this particular type of joint, it is, therefore,

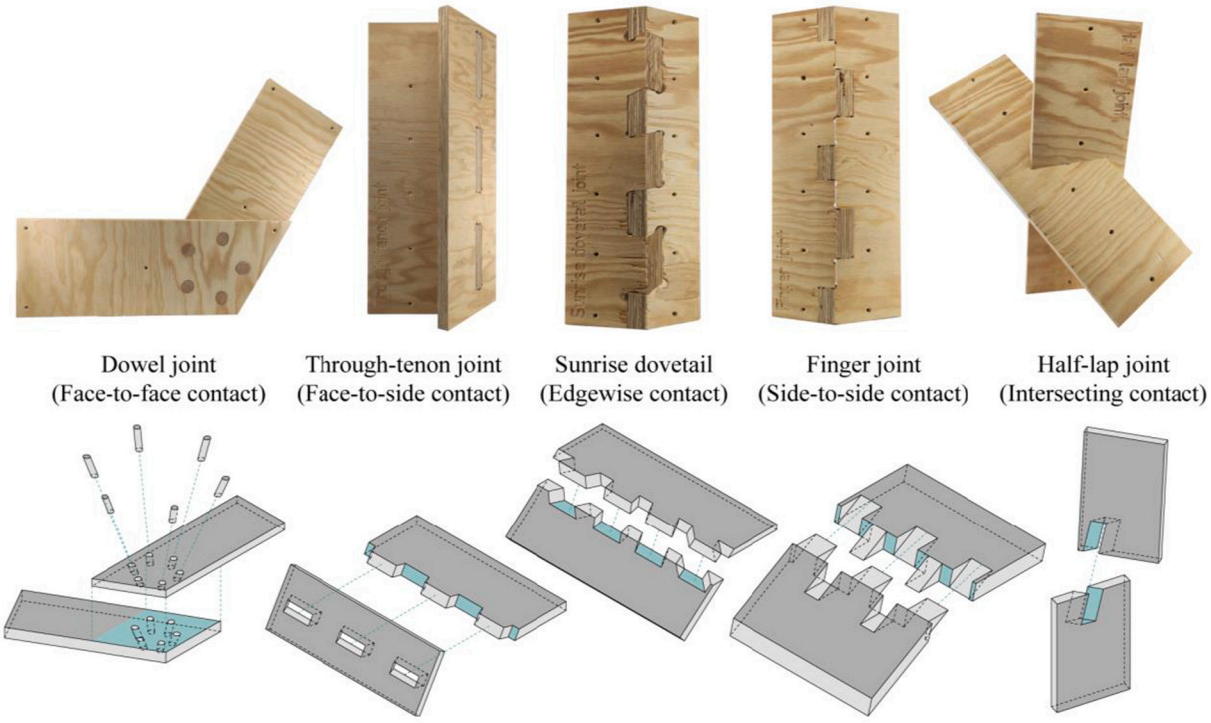


Fig. 14. Five types of timber joints have been parametrized and can be generated with the solver.

necessary to rely on an existing library of geometric algorithms. The parametrization of the half-lap joint is nevertheless detailed in Fig. 15.

For all timber joints, the solver splits the output into three lists: one list of 3D solids to add to the plate geometry, another to subtract from the plate, and the last one for independent solids such as external keys or dowels (Fig. 16). The final geometry of each plate is obtained by performing Boolean operations with the solids of the first two lists (e.g. tenons are merged with the plate initial geometry while mortises are subtracted). However, those operations require a high computational time. To keep fast feedback and enable an iterative process, the solver keeps all joints as distinct geometries separated from the plate. When the user is satisfied with the design, joints are ultimately merged within the plate for visualization purposes.

5.2. Generating fabrication toolpath for CNC machining

The interest of developing an accurate 3D modeling tool for timber joints in IATPS is to enable a direct workflow from design to fabrication. The parameterization of each joint presented in this paper encompasses creating a 3D solid for visualization purposes and polylines and surfaces informing the milling process. This geometric data can be interpreted by CAD-CAM software to generate a machining toolpath which can then be exported to a CNC using a standard machine language as G-Code [39]. Although a CNC router allows cutting complex joints into standardized timber panels, it also brings some constraints to the design mainly due to axis reachability and cutting tools characteristics.

The “inside corner” problem is a well-known example of CNC fabrication constraints (see Fig. 17a). Due to the circular cross-section of CNC milling bits, it is impossible to cut sharp corners with a CNC. For timber joints, this is typically problematic to fabricate a mortise matching a rectangular tenon. To make both parts compatible, removing a bit more material is necessary. This can be done by extending the cutting toolpath to create notches in the corners. As those fabrication details can impact both the structural performance and the aesthetic of the connections, there is an interest in integrating them in an iterative design workflow [4]. Consequently, the developed tool automates the generation of those notches and updates the machining toolpath accordingly. It also provides the possibility to choose between two alternatives: dog-bone fillets (Fig. 17b) or T-bone fillets (Fig. 17c) [40].

Other functionalities have also been developed to ease the manipulation of the 3D model. All plates can be scaled and reoriented independently with their attributes. They can also be automatically laid flat in an array or a grid to visualize fabrication toolpath better or stacked on top of each other to prepare robotic trajectories (see Section 5.3). Per default, the bottom face of each plate is facing downward. However, the top and bottom faces can be inverted by the user to specify the best cutting approach for non-orthogonal machining (using more than three axes).

The solver returns two types of outputs for fabrication purposes: contours and surfaces. Contours are pairs of closed polylines that can be interpolated to create multiple milling passes. The orientation of the cutting tool is given by joining each pair of corresponding vertices from the top polyline to the bottom polyline. Surfaces are typically used to chamfer tenons (as illustrated in the appendix). The orientation of the cutting tool is here given by the normal of the surface, and a snake-like surfacing toolpath can then be generated according to the tool radius.

5.3. Generating robot trajectories for an automated assembly process

One objective of this research is to propose a general framework for the automated assembly of timber plate structures. By splitting the assembly sequence into modules, groups of plates can be preassembled offsite with an industrial robotic arm. A robotic insertion strategy for timber joints based on the visual detection of fiducial markers has already been detailed in a previous paper [41]. Therefore, only geometrical considerations about the robot trajectories will be discussed here. Tridimensional path planning usually requires a long computational time as checking for object collisions is a complex task, and the solution space is vast [42]. To enable fast feedback for designers, a more straightforward approach was taken. A potential trajectory based on nine predefined moves is extrapolated from the vector of insertion associated with each plate (Fig. 18). The open-source plugin Robots [43]

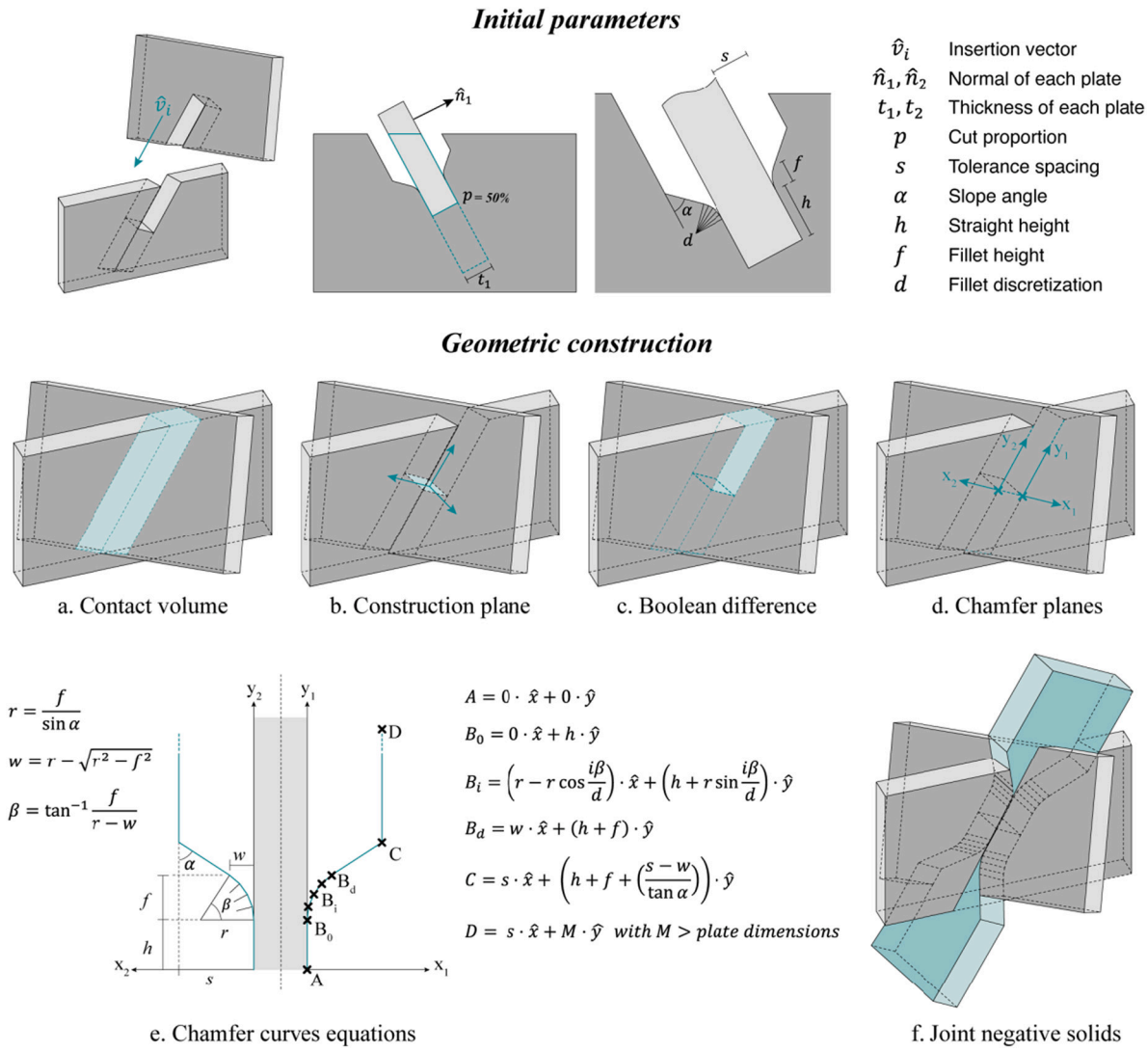


Fig. 15. Parametric construction of the half-lap joint as implemented in the solver. The use of solid Boolean operations to get the chamfer planes precluded a purely mathematical description of the joint geometry. For the other joints, the equations can be found in the appendix.

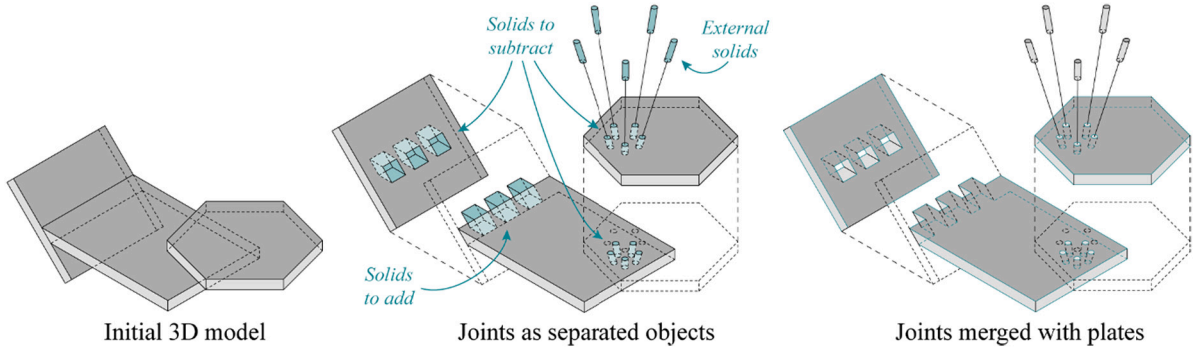


Fig. 16. Joint geometry is stored as three lists of solids and kept separated from the plate geometry until the design is fixed. Solid Boolean operations allow integrating the joints inside the plates to display the result.

is then used to check if there are no obstructions and if the robot can reach all positions. Note that other robotic simulation plugins could also be used in function of the model of the robot. If necessary, the user can tweak the trajectory with some additional parameters. The direction of each movement is fixed but lengths can be modified. While this semi-automated method eventually requires some manual adjustment, it simplifies the assembly process using a predictive but parametric trajectory.

6. Solver application and performance

The developed solver has been tested on different 3D models to

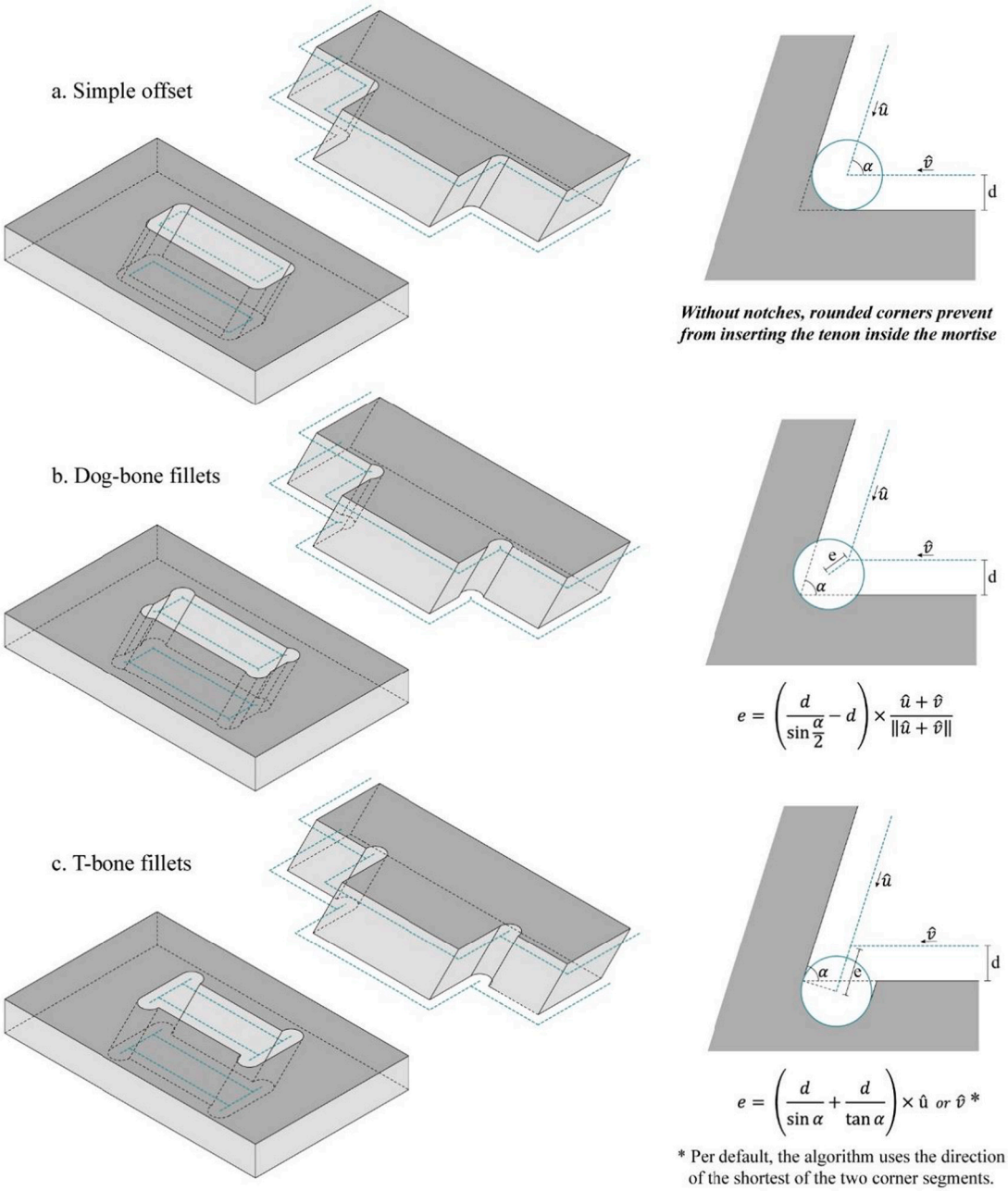


Fig. 17. Integrating corner notches into fabrication toolpath.

assess its performance. This section presents three structures for which joints geometry, fabrication toolpath, and robotic trajectories have been automatically generated. Further investigations are still required to tackle the physical assembly of those structures with a robotic arm. Major challenges remain to insert all types of joints with varying plate configurations [41]. The purpose of those case studies is exclusively to demonstrate the range of applications of the integrated design tool by applying the solver on different geometries.

6.1. Case study 1: curved beam

For the first case study (Fig. 19), a 4-m-long curved beam made out of 18 timber panels was considered. Through-tenons joints were applied on face-to-side contacts to connect the beam web with both flanges. The top flange is more segmented, being composed of 7 panels to obtain a curved shape. The bottom flange is composed of three longer panels. Therefore, different parameters were used for the top and bottom layers. A single 10 cm wide tenon was generated for each contact zone in the top flange while two tenons of 5 cm were used for the contact zones in the lower flange. Finger joints were used for side-to-side connections in both flanges. However, all contacts between web elements were discarded to avoid assembly issues. Due to the angle between the top flange plates, a slight chamfer needs to be surfaced on top of the fingers, as shown on the bottom right of Fig. 19. The fabrication toolpath also includes dog-bone notches for outer and inner milling curves. To simulate the robotic assembly of the structure, an ABB 6400 robot equipped with a gripping end-effector was reproduced in the parametric environment. The trajectory of each plate was validated after adjusting the position of the plate stack and the other parameters mentioned in the previous section.

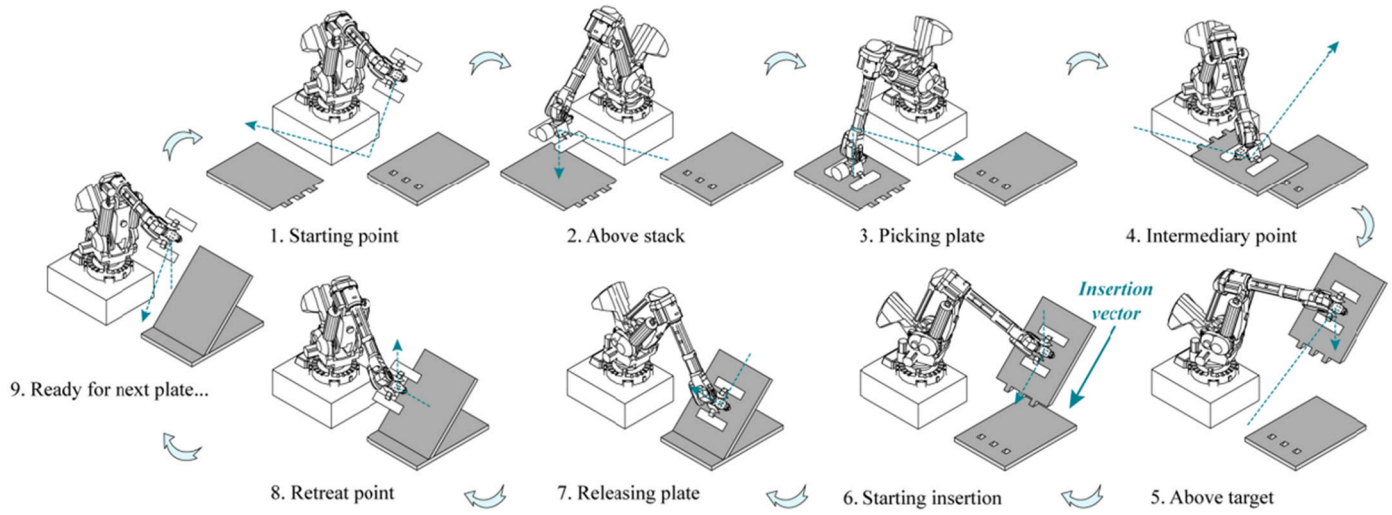


Fig. 18. The different steps of the robotic trajectory are similar for each plate. However, distances can be adjusted, and intermediary points can be added to work around obstacles and avoid collisions.

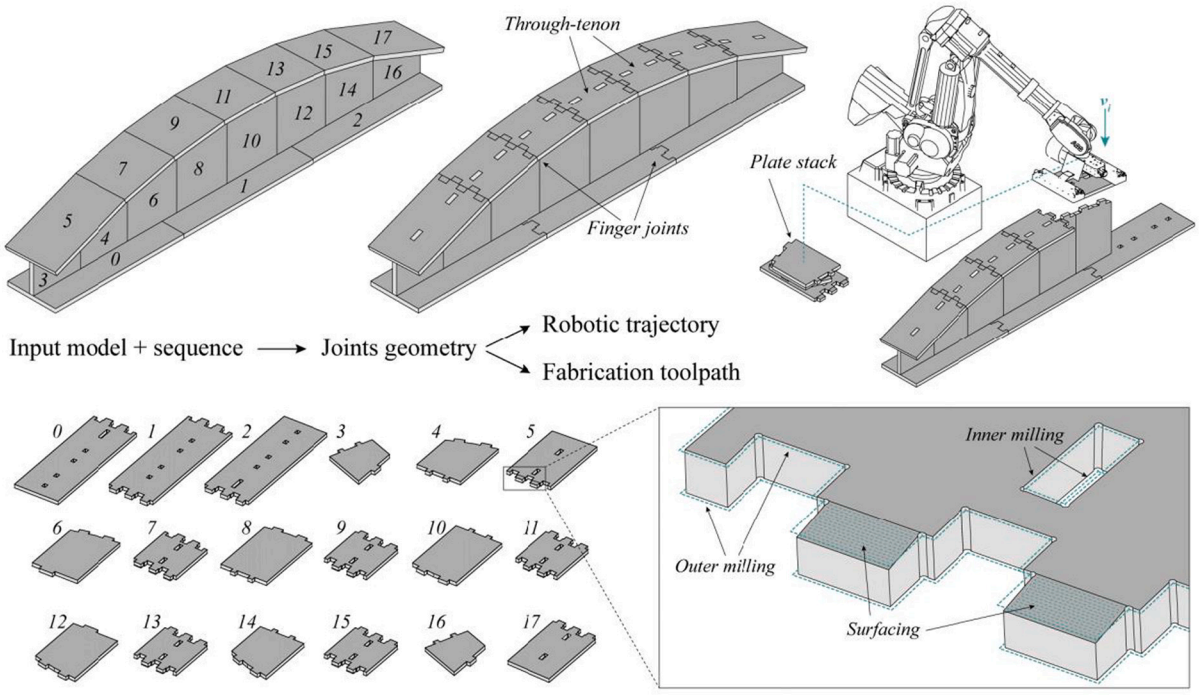


Fig. 19. Application of the solver on a segmented curved beam made out of 18 panels.

6.2. Case study 2: boxed vault

The second case study is a boxed vault composed of 20 hexagonal modules of 7 plates (Fig. 20). A doubly-curved NURBS surface was first segmented by projecting a hexagonal tiling pattern on it. Boxes were then extruded by intersecting bisector planes from each mesh edge [44]. Boxes are slightly shifted about each other, as no optimization algorithm was used to planarize the initial geometry. For this example, a modular assembly sequence was set as input: each group of seven plates forms a module to which a vector of insertion and a subsequence are assigned. Regarding the joints, sunrise dovetails were applied on all edgewise connections to connect the plates of the box, while dowel joints were used to connect the boxes. Fig. 20 highlights how the solver not only creates the joints geometry but also gives a complete overview of the different steps of the construction process. This goes from the preview of t-bone notches for the manufacturing process to the simulation of the insertion of the different elements (here performed manually) for the assembly of the structure.

6.3. Case study 3: timber frame

For the third case study, a modular assembly sequence was also used. The structure is composed of two arches of 13 plates each and 21 purlins (Fig. 21). In this scenario, both arches are preassembled with a robot while the purlins are added manually. Therefore, the plates of the arches are grouped in two distinct modules in the input sequence. Dowel joints were generated at each face-to-face contact to create a rigid connection between the pieces of the arches. Half-lap joints were used for the purlins with a guiding slope to ease the assembly and dog-bone notches in the corners. Robotic trajectories have been generated for one of the arch

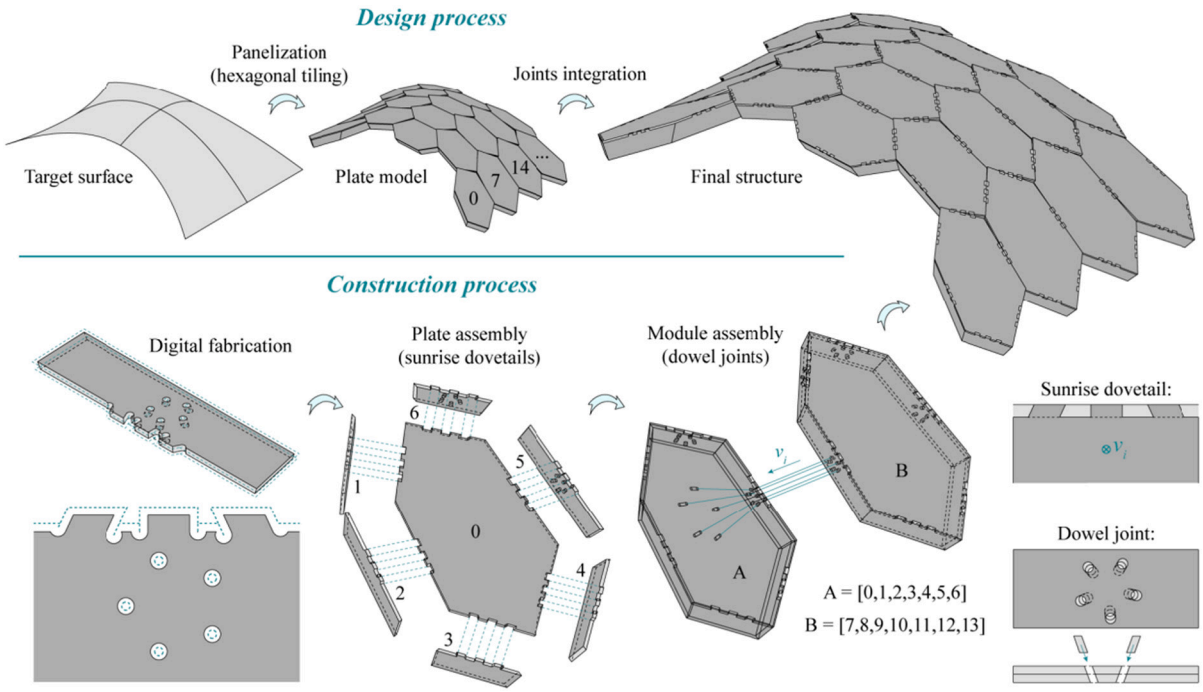


Fig. 20. Application of the solver on a boxed vault composed of 20 hexagonal modules of 7 plates.

modules after scaling it down according to the size of our robot model. The entire assembly sequence for this module was successfully simulated using the output of the solver.

6.4. Computational performance

The performance of the solver is presented in Fig. 22 for the three case studies. The computational time associated with each step of the algorithm is detailed. Boolean operations remain the most computationally intensive task, with more than 2 min required to merge the joints and the notches of the complete boxed vault. The model instantiation also requires a consequent time to compute since finding all contact zones relies also on Boolean operations. Besides, this part of the algorithm runs in the order of n^2 as each pair of plates needs to be tested. Therefore, computing the adjacency of an assembly of a thousand plates might take a considerable amount of time. However, only generating the joints geometry remains quite fast. For the second case study, 1365 solids were generated in less than 5 s. This maintains the possibility to explore different design options quickly before performing the Boolean operations. The time needed to create a joint depends on its geometric complexity. The simple cylinder of a dowel joint is typically much faster to generate than a half-lap joint with rounded guiding slopes. It should also be noted that working inside the Grasshopper environment requires creating a new copy of the model instance before executing any task which added between 1 and 2 s of delay to every step. Finally, it is rarely necessary to work on the entire model at once. Only performing Boolean operations on one plate or module greatly enhances the solver's responsivity.

The solver was also tested in a workshop with architecture students who had only very limited knowledge of parametric design. This was a good opportunity for testing the user experience of the developed Grasshopper plugin for non-experts. Surprisingly, students were rapidly capable of designing relatively complex assemblies of a dozen plates. The tool also proved to be particularly effective to explain and visualize assembly constraints. The documentation of the potential sources of error in the workflow was also greatly improved after students' feedback.

7. Conclusion and outlook

An integrated design tool for timber plate structures connected by wooden joints has been developed and tested on three case studies. It offers an effective way to integrate digital fabrication and robotic assembly constraints into the design of standard and bespoke structures. Compatible timber joints are automatically created by interpreting an assembly sequence set by the user. This sequence can also be divided into modules allowing a multi-step assembly. By generalizing the joinery system, more freedom is given to the designer, and the impact of construction constraints on the project can be better understood. Besides, this opens the possibility to easily extend the library of joints by following the same framework.

The application of the algorithm to three case studies resulted in a relatively high calculation time, especially for large assemblies. This is caused mainly by three factors. First, the solver needs to look for all contacts between the elements since no information about adjacency is provided. Second, Boolean operations are computationally-intensive tasks. Third, the Grasshopper environment requires creating a new copy of all properties stored in the model at each step to avoid conflicts when running parallel tasks. However, using this node-based visual scripting platform, it is possible to complete the complex modeling of all timber joints independently of the model instantiation. This drastically reduces the delay between two iterations, maintaining a high level of interactivity. Besides, it is not necessary to merge the joints with their parent plate to visualize the joints. Boolean operations can therefore be postponed after the end of the design process.

The solver could also be optimized to reduce calculation times. A list of all adjacent plates could optionally be provided by the user in addition to the assembly sequence, avoiding the need to compute plate intersections. If the global shape of the model is designed using a mesh data structure, adjacency information could easily be transferred from the mesh to the plate model. Finally, implementing external libraries optimized for Boolean operations could also speed up the process.

The joinery solver could also be extended to beam elements or more complex polyhedra. Similar logic to plates could be considered, although plates have the advantage of having only two potential orientations, which reduces the amount of user input required. The current

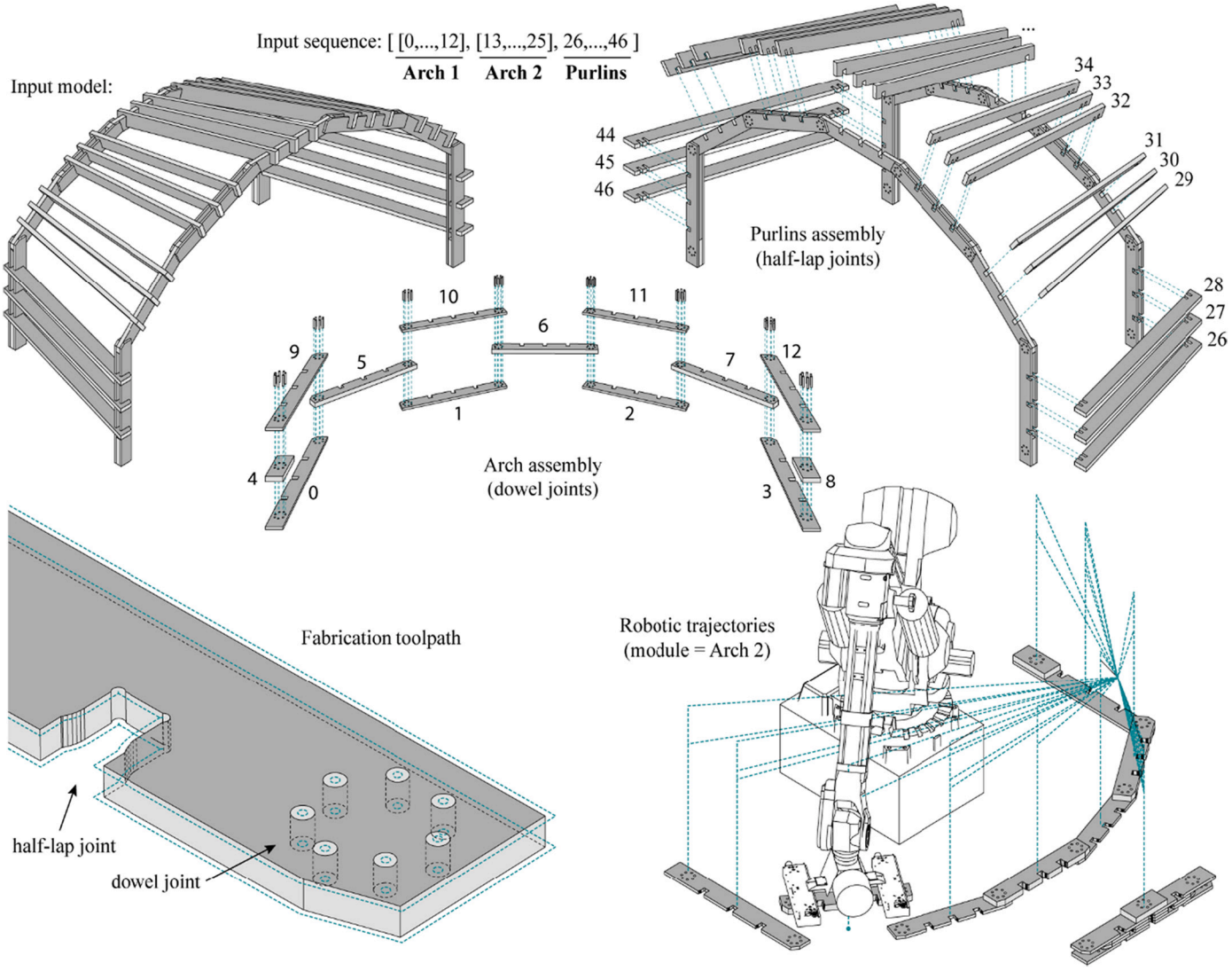


Fig. 21. Application of the solver on a timber frame composed of two arches of 13 pieces each and 21 purlins.

Case study	*Model instanciation*	*Joints generation*	*Toolpath generation*	*Boolean operations*	*Robotic simulation*
#1 Curved beam 18 plates	39 contacts 2.9 s	136 solids (8 FJ, 22 TT) 2.9 s	322 notches 0.8 s	without notches / with notches 2.7 s / 22.6 s	234 planes 1.0 s
#2 Boxed vault 140 plates	409 contacts 39.1 s	1365 solids (43 DJ, 120 SD) 4.6 s	1440 notches 10.3 s	without notches / with notches 25.5 s / 126 s	/ /
#3 Timber frame 46 plates	196 contacts 6.6 s	804 solids (64 DJ, 66 HL) 6.3 s	264 notches 8.4 s	without notches / with notches 18.1 s / 30.8 s	169 planes (1 arch) 3.2 s

System specifications: *CPU: Intel(R) Xeon(R) E3-1505M v6 @ 3.00GHz, RAM:16 Go, GPU: NVIDIA Quadro M2200*

Fig. 22. Computational time of each operation for each case study. Joints abbreviations: DJ = Dowel joint, TT = through-tenon joint, SD = Sunrise Dovetails, FJ = Finger Joint, HL = Half-lap joint.

framework targets translational assemblies with pieces moving according to one insertion vector. More complex insertion trajectories involving rotations could also be considered.

In conclusion, this research constitutes a first attempt at generalizing the design process of IATPS. While the computational performance of the solver could still be improved, its application to three case studies demonstrated the interest in connecting architectural design, digital fabrication, and robotic assembly under one roof. Parallel investigations have been carried on the mechanical characterization of such structures [6], and a logical development would be to also integrate structural feedback into this general design framework.

This contribution opens new perspectives for IATPS. The development of this new tool facilitates the manipulation of this innovative structural system by architects and structural designers who are not necessarily familiar with coding. Consequently, this fosters the shift in the status of IATPS from iconic research pavilions to a more widespread and highly automated building system that allows for both standardized and bespoke structures.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

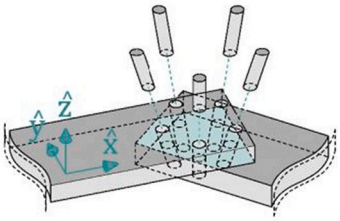
Acknowledgments

This research was supported by the NCCR Digital Fabrication, funded by the Swiss National Science Foundation (NCCR Digital Fabrication Agreement #51NF40-141853). The authors would like to thank Alexandre Flamant for his contribution to the equations describing joints geometry, and Dr. Julien Gamerro and Dr. Aryan Rezaei Rad for their support in reviewing the paper.

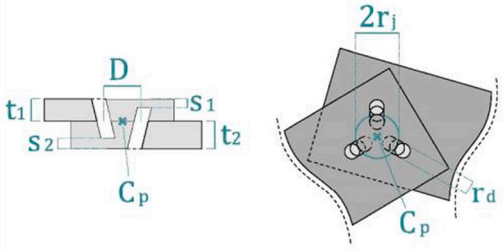
Appendix A. Parametrization of the dowel joint

Dowel joint

Dowel joints can be used to connect timber plates with coplanar faces. With one single dowel, the pieces can rotate freely and the connection is a pin. With more dowels distributed in a circle, a more rigid connection can be created. This mathematical definition gives the possibility to incline the dowels to better lock the parts. The depth of the holes can be controlled to hide the joint from one or both sides for aesthetic or practical purposes. The radius of the dowel and of circular pattern can also be specified by the user.



Initial parameters



$\hat{x}\,\hat{y}\,\hat{z}$	Construction plane of the joint
C_p	Center of the joint
t_1	Thickness of the top plate
t_2	Thickness of the bottom plate
α	Angle of insertion of the dowels
r	Radius of the dowels
D	Diameter the joint
s_1	Top dowel offset
s_2	Bottom dowel offset
n_d	Number of dowels
n_p	Number of points for discretization

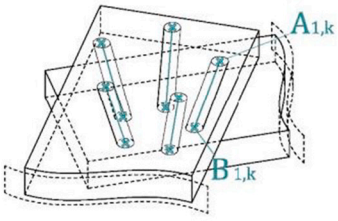
Intermediary steps

1 - Dowels extremities

$$A_{1,k} = C_p + \frac{D}{2} t_1 \tan(\alpha)\left(\cos\left(k\frac{2\pi}{n_d}\right)\hat{x} + \sin\left(k\frac{2\pi}{n_d}\right)\hat{y}\right) + t_1\hat{z}$$

$$B_{1,k} = C_p + \frac{D}{2} t_2 \tan(\alpha)\left(\cos\left(k\frac{2\pi}{n_d}\right)\hat{x} + \sin\left(k\frac{2\pi}{n_d}\right)\hat{y}\right) - t_2\hat{z}$$

$$\forall k < n_d, \in \mathbb{N}$$



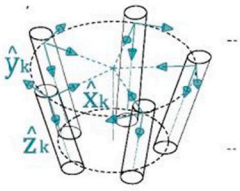
2 - Dowels construction axes

$$\hat{x}_k = \frac{\overrightarrow{A_{1,k}(C_p + t_1\hat{z})}}{\left\|\overrightarrow{A_{1,k}(C_p + t_1\hat{z})}\right\|}$$

$$\hat{y}_k = \frac{\hat{x}_k \times \hat{z}}{\|\hat{x}_k \times \hat{z}\|}$$

$$\hat{z}_k = \frac{\overrightarrow{A_{1,k}B_{1,k}}}{\left\|\overrightarrow{A_{1,k}B_{1,k}}\right\|}$$

$$\forall k < n_d, \in \mathbb{N}$$

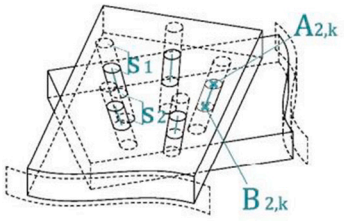


3 - Dowels offsets

$$A_{2,k} = B_{1,k} - s_1\hat{z}_k$$

$$B_{2,k} = A_{1,k} + s_2\hat{z}_k$$

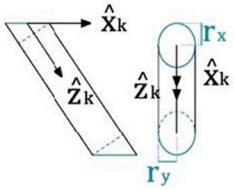
$$\forall k < n_d, \in \mathbb{N}$$



4 - Ellipse radius

$$r_x = \frac{r}{\cos(\alpha)}$$

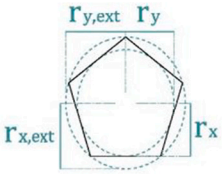
$$r_y = r$$

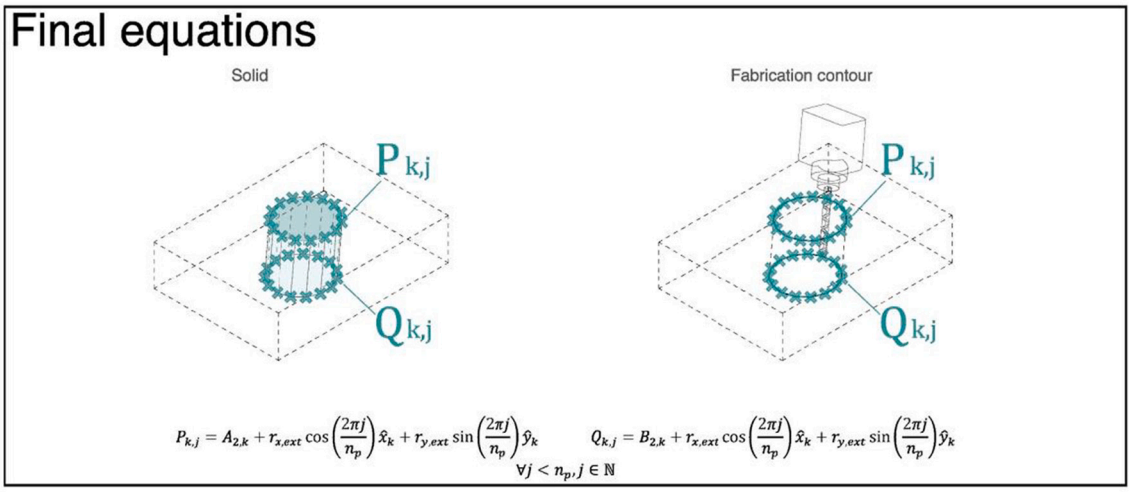


5 - Extern radius

$$r_{x,ext} = \frac{r_x}{\cos\left(\frac{\pi}{n}\right)}$$

$$r_{y,ext} = \frac{r_y}{\cos\left(\frac{\pi}{n}\right)}$$

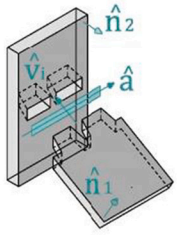




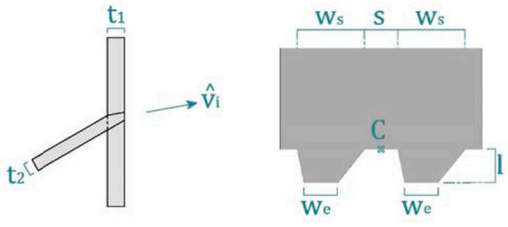
Appendix B. Parametrization of the through-tenon joint

Trough-tenon joint

Through-tenon or mortise-and-tenon joints are commonly used in woodworking to join two pieces with a rigth angle. This joint usually consists of inserting a rectangular tongue formed on one plate into a rectangular hole on the corresponding plate. This mathematical definition extends the insertion space by accepting in-plane and out-of-plane vectors of insertion. Depending on the angle between this vector and the normals of each plate, the joint is then sheared and/or chamfered. Among the different parameters, the length of the tenon can be adjusted to create a stub mortise with the tenon embedded in the plate.



Initial parameters

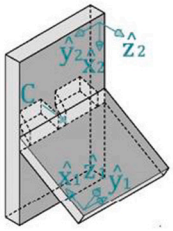


t_1	Thickness of the tenons plate
t_2	Thickness of the mortises plate
w_s w_e	Tenon width at start and end
s	Tenon spacing
C	Center of the joint
l	Length of the tenons
$\hat{v}_i$	Vector of insertion
$\hat{a}$	Axis of the contact zone
$\hat{n}_1$ $\hat{n}_2$	Planes normals

Intermediary steps

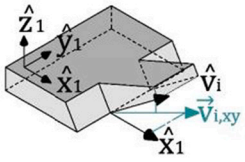
1 - Construction planes

$$\hat{x}_1 = \hat{n}_1 \times \hat{a} \quad \hat{x}_2 = \hat{a} \times \hat{n}_2$$
$$\hat{y}_1 = \hat{a} \quad \hat{y}_2 = -\hat{a}$$
$$\hat{z}_1 = \hat{n}_1 \quad \hat{z}_2 = \hat{n}_2$$



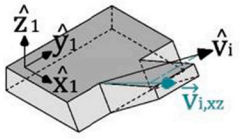
2 - Projection of vi in xy plane

$$\vec{v}_{i,xy} = \frac{\hat{v}_i - (\hat{v}_i \cdot \hat{z}_1)\hat{z}_1}{(\hat{v}_i - (\hat{v}_i \cdot \hat{z}_1)\hat{z}_1) \cdot \hat{x}_1}$$



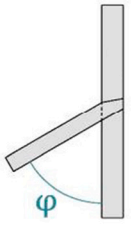
3 - Projection of vi in xz plane

$$\vec{v}_{i,xz} = \frac{\hat{v}_i - (\hat{v}_i \cdot \hat{y}_1)\hat{y}_1}{\|\hat{v}_i - (\hat{v}_i \cdot \hat{y}_1)\hat{y}_1\|}$$



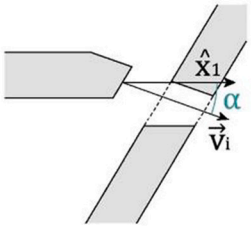
4 - Dihedral angle

$$\varphi = \text{acos}(\hat{n}_1 \cdot \hat{n}_2)$$



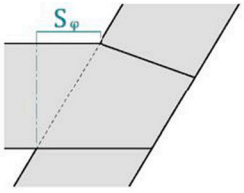
5 - Insertion angle in plane

$$\alpha = \text{acos}(\hat{v}_{i,xz} \cdot \hat{x}_1)$$



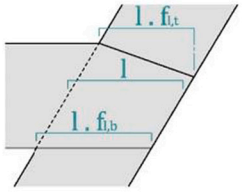
6 - Offset due to non-orthogonality

$$S_\varphi = \sqrt{t_1^2(\sin^{-2}(\varphi) - 1)}$$



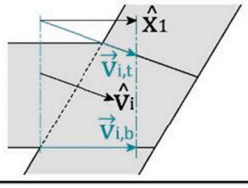
7 - Lenght factor

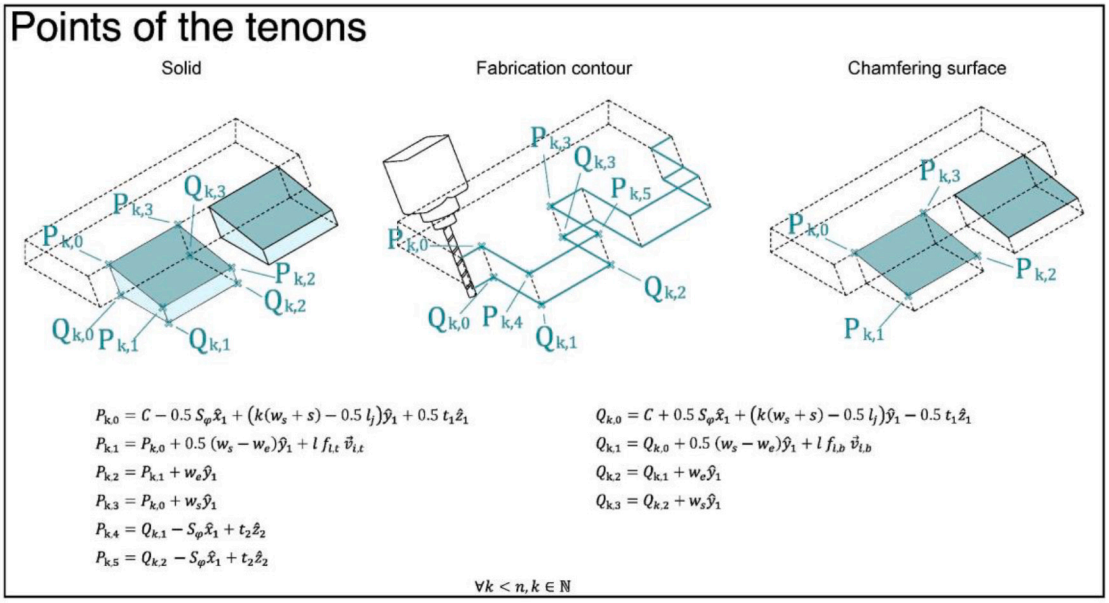
$$f_{l,b} = \frac{1}{2}\left(\frac{\sin(\varphi + \alpha)}{\sin(\varphi - \alpha)} + 1\right)$$
$$f_{l,t} = 1$$

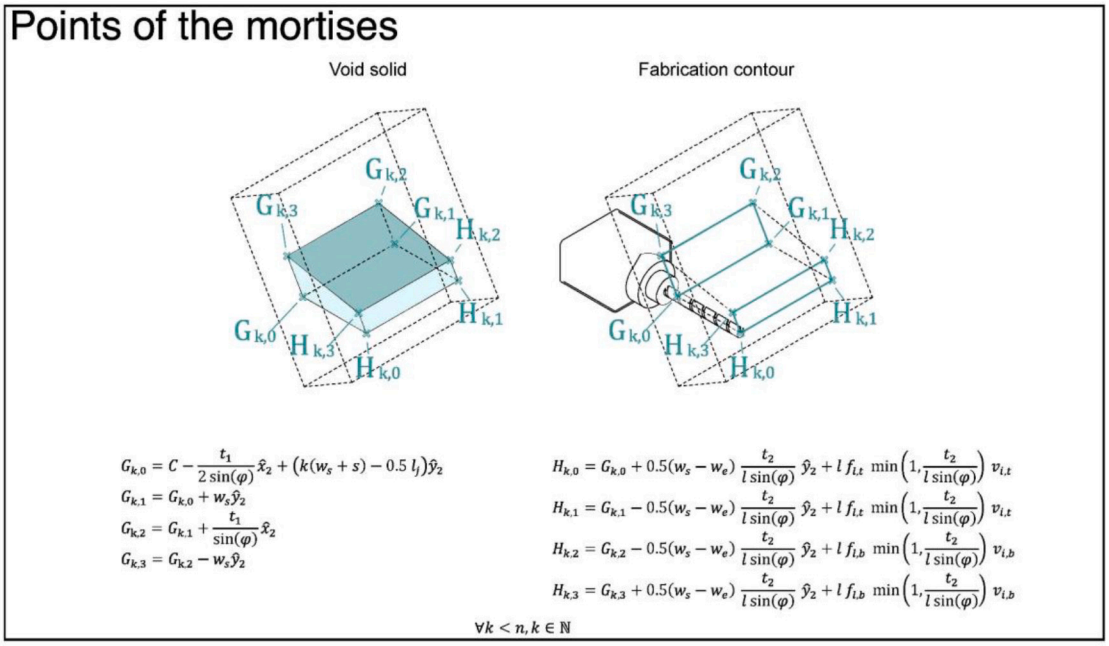


8 - Construction vectors

$$\begin{cases} \vec{v}^*_{i,b} = \vec{v}_{i,xy} - \hat{y}_1 \dfrac{\vec{v}_{i,xy} \cdot (\hat{v}_i \times \hat{y}_i)}{(\hat{v}_i \times \hat{y}_i) \cdot \hat{y}_1} \\ \vec{v}_{i,b} = \dfrac{\vec{v}^*_{i,b}}{\vec{v}^*_{i,b} \cdot \hat{x}_1} \end{cases}$$
$$\vec{v}_{i,t} = \frac{\hat{v}_i}{\hat{v}_i \cdot \hat{x}_1}$$



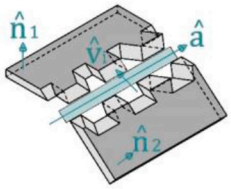




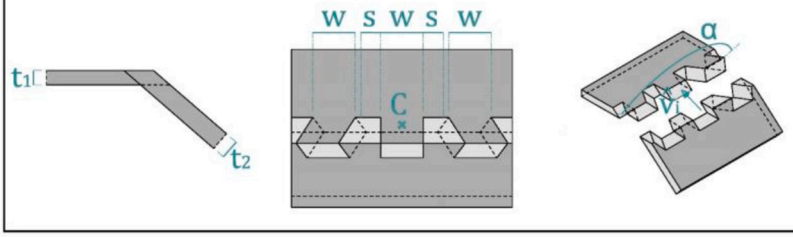
Appendix C. Parametrization of the sunrise dovetail joint

Sunrise dovetail joint

Sunrise dovetails, also called nejiri arigata in Japan, are a particular case of dovetail joints with multiple fingers oriented in different directions. This highly aesthetic joint has a unique vector of insertion which can be found by intersecting all the planes of the different fingers. This mathematical definition allows controlling the maximum angle between two planes as well as the number of fingers. The final shape of the joint is given by the orientation of the vector of insertion specified by the user.



Initial parameters

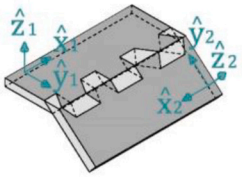


t_1	Thickness of the tenons plate
t_2	Thickness of the mortises plate
w	Finger width at start and end
s	Finger spacing
C	Center of the joint
$\hat{v}_i$	Vector of insertion
$\hat{a}$	Axis of the contact zone
$\hat{n}_1\ \hat{n}_2$	Planes normals
α	Spread angle around $\hat{V}_i$
n	Number of fingers

Intermediary steps

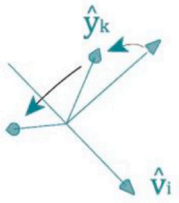
1 - Construction plane

$$\hat{x}_1 = \hat{n}_1 \times \hat{a} \quad \hat{x}_2 = \hat{a} \times \hat{n}_2$$
$$\hat{y}_1 = \hat{a} \quad \hat{y}_2 = -\hat{a}$$
$$\hat{z}_1 = \hat{n}_1 \quad \hat{z}_2 = \hat{n}_2$$



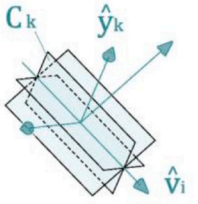
2 - Rotation matrix

$$R(\theta) = \begin{pmatrix} v_{i1}^2(1-c)+c & v_{i1}v_{i2}(1-c)-v_{i3}s & v_{i1}v_{i3}(1-c)+v_{i2}s \\ v_{i1}v_{i2}(1-c)+v_{i3}s & v_{i2}^2(1-c)+c & v_{i2}v_{i3}(1-c)-v_{i1}s \\ v_{i1}v_{i3}(1-c)-v_{i2}s & v_{i2}v_{i3}(1-c)+v_{i1}s & v_{i3}^2(1-c)+c \end{pmatrix}$$
$$c = \cos(\theta) \quad s = \sin(\theta) \quad \hat{v}_i = (\hat{v}_{i1}, \hat{v}_{i2}, \hat{v}_{i3})^t$$



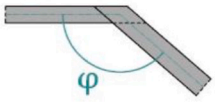
3 - Fingers planes

$$\hat{a}_{c,k} = R\left(\frac{\pi-\alpha}{2} + k\frac{\alpha}{n-1}\right)\cdot \hat{a}_1$$
$$C_k \equiv \{\hat{v}_i, \hat{a}_{c,k}\}$$
$$\forall k < n, k \in \mathbb{N}$$



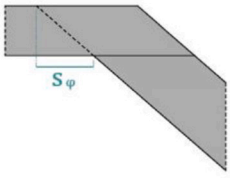
4 - Dihedral angle

$$\varphi = \text{acos}(\hat{x}_1 \cdot \hat{x}_2)$$



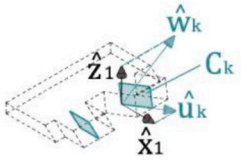
5 - Offset due to non-orthogonality

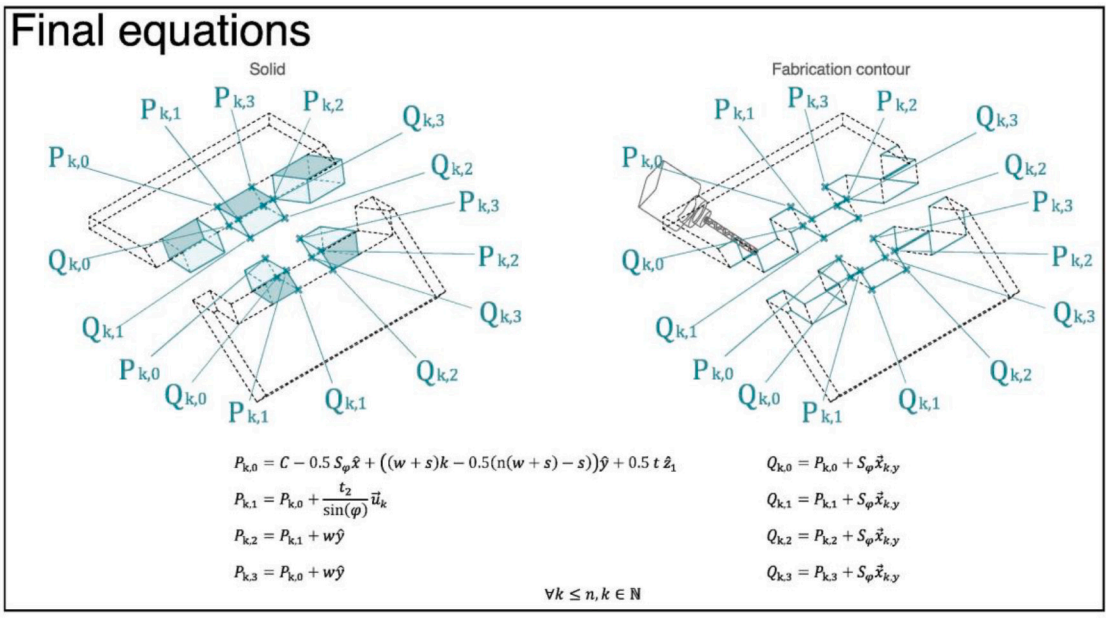
$$S_\varphi = \sqrt{t_1^2(\sin^{-2}(\varphi) - 1)}$$



6 - Construction vectors

$$\vec{u}_k^* = \hat{x}_1 - \frac{\hat{x}_1 \cdot (\hat{v}_i \times \hat{a}_{c,k})}{\hat{a} \cdot (\hat{v}_i \times \hat{a}_{c,k})}\hat{a} \quad \vec{w}_k^* = \hat{z}_1 - \frac{\hat{z}_1 \cdot (\hat{v}_i \times \hat{a}_{c,k})}{\hat{a} \cdot (\hat{v}_i \times \hat{a}_{c,k})}\hat{a}$$
$$\vec{u}_k = \frac{\vec{u}_k^*}{\vec{u}_k^* \cdot \hat{x}_1} \qquad \vec{w}_k = \frac{\vec{w}_k^*}{\vec{w}_k^* \cdot \hat{z}_1}$$
$$\forall k < n, k \in \mathbb{N}$$

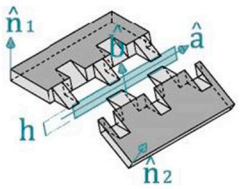




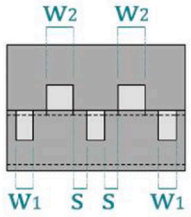
Appendix D. Parametrization of the finger joint

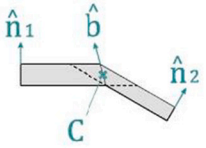
Finger joint

Finger joints or comb joints are traditionally used to interlock two pieces end to end. This can be done to cover a longer span or a larger area than what is permitted by the size of the pieces. The principle consists in creating a complementary pattern on both sides of the contact zone so that for each finger on one side, there is a corresponding hole on the other side. This mathematical definition generalizes the finger joint to non-orthogonal cases with potentially different thicknesses of plates. The number of fingers on each side, their width and their spacing can also be controlled.



Initial parameters



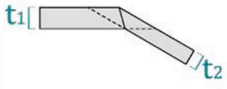


w_1 w_2	Widths of the fingers of the plates
s	Distance between fingers
h	Height of the contact zone
$\hat{a}$	Longitudinal axis of the contact zone
$\hat{b}$	Longitudinal axis of the contact zone
$\hat{n}_1$ $\hat{n}_2$	Planes normals
n_{c1} n_{c2}	Number of fingers of the plates

Intermediary steps

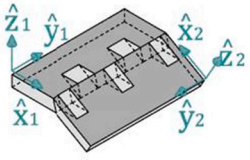
1 - Plates thickness

$$t_1 = d.|\hat{z}_1 \cdot \hat{b}|$$
$$t_2 = d.|\hat{z}_2 \cdot \hat{b}|$$



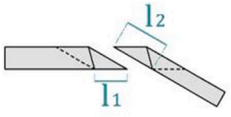
2 - Construction plane

$$\hat{x}_1 = \hat{n}_1 \times \hat{a} \quad \hat{x}_2 = \hat{a} \times \hat{n}_2$$
$$\hat{y}_1 = \hat{a} \quad \hat{y}_2 = -\hat{a}$$
$$\hat{z}_1 = \hat{n}_1 \quad \hat{z}_2 = \hat{n}_2$$

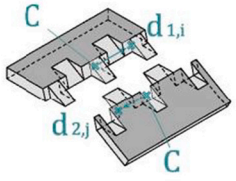


3 - Length of the fingers

$$l_1 = \frac{t_2}{|\hat{z}_1 \cdot \hat{x}_2|}$$
$$l_2 = \frac{t_1}{|\hat{z}_2 \cdot \hat{x}_1|}$$



4 - Finger start location for n1 >= n2



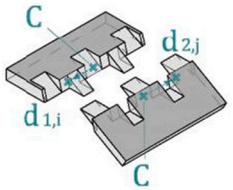
$$d_{1,i} = \begin{cases} i(w_1 + s) - \frac{l_j}{2} & k < \left\lfloor \frac{\Delta n}{2} \right\rfloor \\ \left\lfloor \frac{\Delta n}{2} \right\rfloor (w_1 + s) + \left(i - \left\lfloor \frac{\Delta n}{2} \right\rfloor\right)(w_1 + w_2 + 2s) - \frac{l_j}{2} & \left\lfloor \frac{\Delta n}{2} \right\rfloor \le k < \left\lfloor \frac{\Delta n}{2} \right\rfloor + n_2 \\ \left\lfloor \frac{\Delta n}{2} \right\rfloor (w_1 + s) + n_2(w_1 + w_2 + 2s) + \left(i - \left\lfloor \frac{\Delta n}{2} \right\rfloor - n_2\right)(w_1 + s) - \frac{l_j}{2} & \left\lfloor \frac{\Delta n}{2} \right\rfloor + n_2 \le k < n_1 \end{cases}$$

$$d_{2,j} = (\Delta n + 1)(w_1 + s) + j(w_1 + w_2 + 2s)$$

$$l_j = n_1(w_1 + s) + n_2(w_2 + s) - s$$

$$\Delta n = |n_1 - n_2|$$

4 - Finger start location for n1 < n2

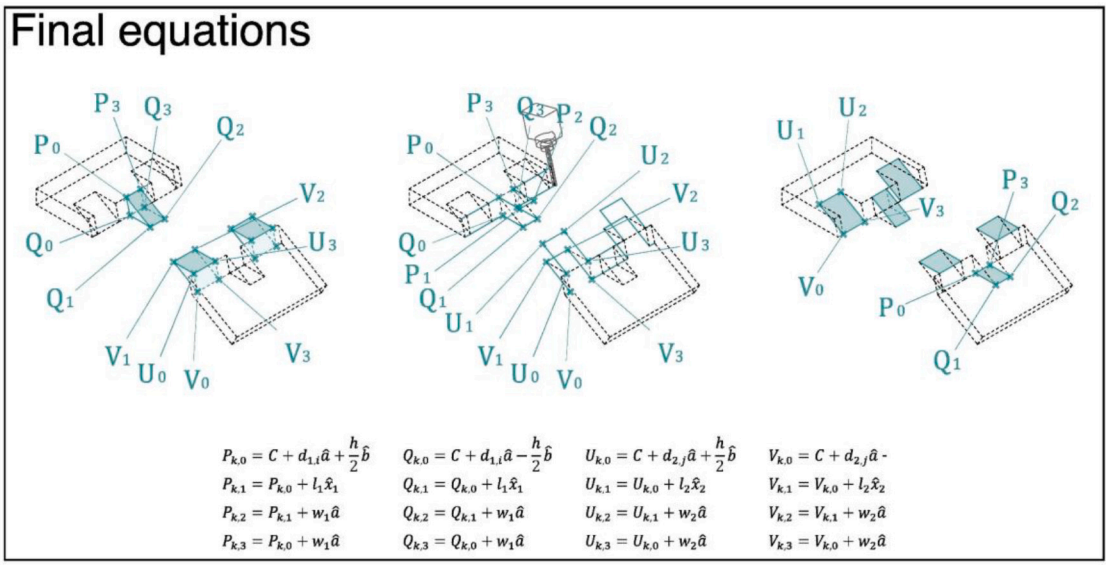


$$d_{1,i} = (\Delta n + 1)(w_2 + s) + i(w_1 + W_2 + 2s)$$

$$d_{2,j} = \begin{cases} j(w_2 + s) - \frac{l_j}{2} & k < \left\lfloor \frac{\Delta n}{2} \right\rfloor \\ \left\lfloor \frac{\Delta n}{2} \right\rfloor (w_2 + s) + \left(j - \left\lfloor \frac{\Delta n}{2} \right\rfloor\right)(w_1 + w_2 + 2s) - \frac{l_j}{2} & \left\lfloor \frac{\Delta n}{2} \right\rfloor \le k < \left\lfloor \frac{\Delta n}{2} \right\rfloor + n_1 \\ \left\lfloor \frac{\Delta n}{2} \right\rfloor (w_2 + s) + n_2(w_1 + w_2 + 2s) + \left(j - \left\lfloor \frac{\Delta n}{2} \right\rfloor - n_1\right)(w_2 + s) - \frac{l_j}{2} & \left\lfloor \frac{\Delta n}{2} \right\rfloor + n_1 \le k < n_2 \end{cases}$$

$$l_j = n_1(w_1 + s) + n_2(w_2 + s) - s$$

$$\Delta n = |n_1 - n_2|$$



References

[1] W. Tegel, R. Elburg, D. Hakelberg, H. Stäuble, U. Büntgen, Early neolithic water wells reveal the world's oldest wood architecture, PLoS One 7 (2012), https://doi.org/10.1371/journal.pone.0051374.

[2] W. Graubner, Encyclopedia of Wood Joints, Chrysalis Books, 1992 (ISBN 9780713470918).

[3] T. Benson, Building the Timber Frame House: The Revival of a Forgotten Craft, Touchstone, 1981 (ISBN 9781439107072).

[4] C.W.M. Robeller, Integral Mechanical Attachment for Timber Folded Plate Structures, Ecole Polytechnique Fédérale de Lausanne, 2015, https://doi.org/10.5075/epfl-thesis-6564.

[5] J. Gamerro, I. Lemaître, Y. Weinand, Mechanical characterization of timber structural elements using integral mechanical attachments, in: World Conference on Timber Engineering, Seoul, 2018, https://doi.org/10.5075/epfl-ibois-256646.

[6] A. Rezaei Rad, H. Burton, N. Rogeau, P. Vestartas, Y. Weinand, A framework to automate the design of digitally-fabricated timber plate structures, Comp. Struct. 244 (2021), https://doi.org/10.1016/j.compstruc.2020.106456.

[7] A. Rezaei Rad, H.V. Burton, Y. Weinand, Macroscopic model for spatial timber plate structures with integral mechanical attachments, J. Struct. Eng. 146 (2020), https://doi.org/10.1061/(ASCE)ST.1943-541X.0002726.

[8] O.D. Krieg, T. Schwinn, A. Menges, J.-M. Li, J. Knippers, A. Schmitt, V. Schwieger, Biomimetic lightweight timber plate shells: computational integration of robotic fabrication, architectural geometry and structural design, in: Advances in Architectural Geometry, London, 2014, pp. 109–125, https://doi.org/10.1007/978-3-319-11418-7_8.

[9] T. Schwinn, D. Sonntag, T. Grun, J.H. Nebelsick, J. Knippers, A. Menges, Potential applications of segmented shells in architecture, in: Biomimetics for Architecture, Birkhäuser, 2019, pp. 116–125, https://doi.org/10.1515/9783035617917-015.

[10] H.J. Wagner, M. Alvarez, A. Groenewolt, A. Menges, Towards digital automation flexibility in large-scale timber construction: integrative robotic prefabrication and co-design of the BUGA Wood Pavilion, Construct. Robot. 4 (2020) 187–204, https://doi.org/10.1007/s41693-020-00038-5.

[11] V.E. Tagliaboschi, Hexbox Canopy: A Rapid Assembly Segmented Timber Shell with Wedge Joints, Università di Pisa, 2020, https://doi.org/10.13140/RG.2.2.28481.10084.

[12] C. Robeller, V. Viezens, Timberdome: construction system for CLT-segmental plate shells without screws, in: 24th International Timber Construction Forum, Garmisch Partenkirchen, Germany, 2018. https://www.forum-holzbau.com/pdf/41_IHF2018_Robeller_Viezens.pdf (accessed May 3, 2021).

[13] C. Robeller, N. Von Haaren, Recycleshell: wood-only shell structures made from cross-laminated timber (CLT) production waste, J. Int. Assoc. Shell Spatial Struct. 61 (2020) 125–139, https://doi.org/10.20898/j.iass.2020.204.045.

[14] C. Robeller, J. Gamerro, Y. Weinand, Thétre Vidy Lausanne - a double-layered timber folded plate structure, J. Int. Assoc. Shell Spatial Struct. 58 (2017) 295–314, https://doi.org/10.20898/j.iass.2017.194.864.

[15] C. Robeller, M. Konakovic, M. Dedijer, M. Pauly, Y. Weinand, A double-layered timber plate shell: computational methods for assembly, prefabrication, and structural design, in: Advances in Architectural Geometry 2016, Zurich, 2016, pp. 104–122, https://doi.org/10.3218/3778-4_9.

[16] W. Wang, D.-M. Yan, Y. Liu, D. Yan, B. Chan, R. Ling, F. Sun, Hexagonal Meshes With Planar Faces, University of Honk Kong, 2008. https://www.researchgate.net/publication/268324026 (accessed July 10, 2020).

[17] R. Mesnil, A re-parameterization approach for the construction of domes with planar facets, J. Int. Assoc. Shell Spatial Struct. 59 (2018) 286–295, https://doi.org/10.20898/j.iass.2018.198.016.

[18] H. Pottman, A. Asperl, M. Hofer, A. Kilian, Architectural Geometry, 1st ed., Bentley Institute Press, Exton, 2007 (ISBN 9780934493045).

[19] A. Rossi, O. Tessmann, Collaborative assembly of digital materials, in: Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture, Cambridge, 2017, pp. 512–521 (ISBN 9780692965061).

[20] Cadwork, Variant. https://www.cadwork.com/cwen/Modules/Variant-parametric, 2018 (accessed January 30, 2020).

[21] SEMA, Prefabrication, (n.d.). https://www.sema-soft.de/en/software/timber-construction/prefabrication/ (accessed January 30, 2020).

[22] Lignocam, (n.d.). https://www.lignocam.com/index.php/fr/btlx (accessed December 2, 2019).

[23] J. Hollander, MakerCase. https://www.makercase.com/, 2019 (accessed March 20, 2019).

[24] F. Heller, J. Thar, D. Lewandowski, M. Hartmann, P. Schoonbrood, S. Sophy, S. Voelker, J. Borchers, CutCAD - an open-source tool to design 3D objects in 2D, in: Proceedings of the Designing Interactive Systems Conference, Association for Computing Machinery, Hong Kong, 2018, pp. 1135–1139 (ISBN 9781450351980).

[25] P. Baudisch, A. Silber, Y. Kommana, M. Gruner, L. Wall, K. Reuss, L. Heilman, R. Kovacs, D. Rechlitz, T. Roumen, Kyub: a 3D editor for modeling sturdy laser-cut objects, in: Proceedings of the Conference on Human Factors in Computing Systems, Association for Computing Machinery, Glasgow, 2019 (ISBN 9781450359702).

[26] C. Zheng, E.Y.L. Do, J. Budd, Joinery: parametric joint generation for laser cut assemblies, in: Proceedings of the Special Interest Group on Computer-Human Interaction Conference on Creativity and Cognition, Association for Computing Machinery, Singapore, 2017, pp. 63–74 (ISBN 9781450344036).

[27] M. Larsson, H. Yoshida, N. Umetani, T. Igarashi, Tsugite: interactive design and fabrication of wood joints, in: Proceedings of the 33rd Annual Symposium on User Interface Software and Technology, Association for Computing Machinery, 2020, pp. 317–327 (ISBN 9781450375146).

[28] McNeel and Associates, Grasshopper - Algorithmic Modeling for Rhino, (n.d.). https://www.grasshopper3d.com/ (accessed February 9, 2021).

[29] C. Robeller, Y. Weinand, Interlocking folded plate – integral mechanical attachment for structural wood panels, Int. J. Space Struct. 30 (2015) 111–122, https://doi.org/10.1260/0266-3511.30.2.111.

[30] J.H. Mork, M. Luczkowski, S.H. Dyvik, B. Manum, A. Rønnquist, A parametric toolkit for advanced timber structures, in: 6th Forum Wood Building Nordic, Trondheim, 2017. https://www.researchgate.net/publication/332621926 (accessed March 2, 2020).

[31] T. Svilans, Integrated Material Practice in Free-form Timber Structures, Royal Danish Academy of Fine Arts, 2020. https://adk.elsevierpure.com/en/publications/integrated-material-practice-in-free-form-timber-structures (accessed February 9, 2021).

[32] VUILD Architects, EMARF. https://emarf.co/, 2021 (accessed February 9, 2021).

[33] Z. Wang, P. Song, M. Pauly, Desia: a general framework for designing interlocking assemblies, in: Transactions on Graphics, Association for Computing Machinery, 2018, https://doi.org/10.1145/3272127.3275034.

[34] Y. Schwartzburg, M. Pauly, Fabrication-aware design with intersecting planar pieces, Comp. Graph. Forum 32 (2013) 317–326, https://doi.org/10.1111/cgf.12051.
[35] Python Software Foundation, Welcome to Python.org. https://www.python.org/, 2021 (accessed April 6, 2021).
[36] Robert McNeel & Associates, RhinoCommon API, (n.d.). https://developer.rhino3d.com/api/RhinoCommon/ (accessed January 31, 2020).
[37] D. Molloy, Chapter 1 - Introduction to Object-oriented Programming. http://ee402.eeng.dcu.ie/introduction/chapter-1—introduction-to-object-oriented-programming, 2020 (accessed April 29, 2021).
[38] S. Timoshenko, S. Woinowsky-Krieger, Theory of Plates and Shells, 2nd ed., McGraw-Hill Book Company, New York, 1959 (ISBN 9780070647794).
[39] International Organization for Standardization, ISO 6983-1:2009, Automation Systems and Integration, Numerical Control of machiNes, Program Format and Definitions of Address Words, Part 1: Data Format for Positioning, Line Motion and Contouring Control Systems. https://www.iso.org/standard/34608.html, 2009 (accessed April 6, 2021).
[40] N. Padfield, More Elegant CNC Dogbones, Fablab RUC. https://fablab.ruc.dk/more-elegant-cnc-dogbones/, 2017 (accessed March 16, 2021).
[41] N. Rogeau, V. Tiberghien, P. Latteur, Y. Weinand, Robotic insertion of timber joints using visual detection of fiducial markers, in: 37th International Symposium on Automation and Robotics in Construction, Kitakyushu, Japan, 2020, pp. 491–498, https://doi.org/10.22260/ISARC2020/0068.
[42] A. Gandia, S. Parascho, R. Rust, G. Casas, F. Gramazio, M. Kohler, Towards Automatic Path Planning for Robotically Assembled Spatial Structures, in: Robotic Fabrication in Architecture, Art and Design 2018, Zurich, 2018, pp. 60–73 (ISBN 9783319922942).
[43] V. Soler, Robots. https://github.com/visose/Robots/wiki, 2020 (accessed March 16, 2021).
[44] P. Vestartas, N. Rogeau, J. Gamerro, Y. Weinand, Modelling workflow for segmented timber shells using wood-wood connections, in: Impact: Design With All Senses - Proceedings of the 2019 Design Modelling Symposium, Berlin, 2020, pp. 596–607, https://doi.org/10.1007/978-3-030-29829-6_46.

Glossary

Adjacency list: List of lists containing all neighbors of each cell of a graph.
Assembly sequence: List of integers describing in which order building components have to be assembled. Integers can also be grouped in subsidiary lists to express intermediary steps in a modular assembly process (see subsequence).
B-rep (Boundary representation): Solid represented by its envelope.
CNC (Computer Numerical Control) machine: Milling machine that can be controlled by a computer to automate the cutting of flat panels allowing the realization of bespoke structures from standardized elements.
Contact type: Topologic relation between two plates describing if two plates intersect each other or are juxtaposed and, for the second case, which of their faces is in contact with the other plate (e.g. if the top face of plate A is connected to a side face of plate B, the contact type is labeled "Face-to-side"). The contact type has a direct impact on the type of joints that can be created between both plates.
Contact vector: a local vector of insertion associated with a contact zone between two plates. Contact vectors are determined in function of the contact type.
Contact zone: Planar surface resulting from the intersection of two juxtaposed plates.
Fabrication toolpath: Pair of curves representing the trajectory of a cutting tool. The lower curve gives the position while the upper curve gives the orientation.
G-code: Standard programming language which can be interpreted by most CNC machine to execute a fabrication toolpath.
IATPS (Integrally Attached Timber Plate Structures): Structure composed of standard timber panels which are connected by timber joints. The joints are integrated into the shape of the elements and prefabricated using a CNC machine.
Insertion domain: Set of vectors representing all possible directions to assemble two pieces. Timber joints geometry constraints the insertion domain to certain directions of assembly. Insertion domains can be represented in the 3D space as pieces of spheres.
Insertion vector: Vector representing the direction of assembly of one element or one group of elements (module).
Instantiation: In object-oriented programming, construction of a distinct object based on a class containing specific methods and attributes. All class instances share the same data structure but store different values.
IRA (Industrial Robotic Arm): Articulated robot composed of a chain of connected links which can be programmed to perform actions similar to a human arm.
Plate (class): A python class that stores the plate geometry and other attributes such as the plate contour, the plate thickness, and the geometry of the associated joints.
Plate (geometry): Closed polyhedron that has its two largest faces parallel to each other and all its vertices trivalent. Each vertex of the top face should also be directly connected to exactly one vertex of the bottom face and conversely.
Plate faces: Top and bottom faces of a plate, parallel to each other and larger than all other faces.
Plate model: A Python class representing a collection of plates. The adjacency graph and the assembly sequence are its main attributes.
Plate module: A Python class representing a portion of a plate model. In addition to the model attributes, the module is characterized by a subsequence and a vector of insertion.
Plate neighbor: Adjacent plate which shares a contact zone (surface or volume) with another plate.
Plate sides: Faces which are forming the edges of the plates between the top and bottom plate faces.
Robotic trajectory: Series of robot positions expressed as a list of frames.
Subsequence: Part of an assembly sequence defining the order of insertion for the plates of a specific module in the plate model.
Timber joinery: Technique to connect two pieces of wood solely through their geometry (form-closure) and without the use of mechanical (nails, screws) or chemical (glue) fasteners.