Computational Interlocking Furniture Assembly

Chi-Wing Fu*
Nanyang Technological University

Peng Song*
University of Science and Technology of China

Xiaoqi Yan Lee Wei Yang Pradeep Kumar Jayaraman
Nanyang Technological University

Daniel Cohen-Or
Tel Aviv University

Figure 1: *Some snapshots showing the assembly of* MULTI-FUNCTION TABLE. *Our method can plan a network of joints (e.g., Figure 2) that globally interlocks the component parts in the assembly; the input component parts are just simple 3D shapes without joint geometry.*

Abstract

Furniture typically consists of assemblies of elongated and planar parts that are connected together by glue, nails, hinges, screws, or other means that do not encourage disassembly and re-assembly. An alternative approach is to use an interlocking mechanism, where the component parts tightly interlock with one another. The challenge in designing such a network of interlocking joints is that local analysis is insufficient to guarantee global interlocking, and there is a huge number of joint combinations that require an enormous exploration effort to ensure global interlocking. In this paper, we present a computational solution to support the design of a network of interlocking joints that form a globally-interlocking furniture assembly. The key idea is to break the furniture complex into an overlapping set of small groups, where the parts in each group are immobilized by a local key, and adjacent groups are further locked with dependencies. The dependency among the groups saves the effort of exploring the immobilization of every subset of parts in the assembly, thus allowing the intensive interlocking computation to be localized within each small group. We demonstrate the effectiveness of our technique on many globally-interlocking furniture assemblies of various shapes and complexity.

CR Categories: I.3.5 [Computational Geometry and Object Modeling]: Curve, surface, solid, and object representations

Keywords: interlocking structure, furniture, joint, assembly

1 Introduction

Furniture generally refers to movable objects designed for supporting common human activities such as seating, storage, and office work. Furniture is typically an assembly of elongated and planar parts that are connected together in various ways, for example, by glue, nails, hinges, and screws. However, these common connectors do not encourage furniture disassembly and re-assembly, and often harm the external appearance of the furniture and, in general, the aesthetics of the design [Postell 2012].

An alternative approach is to use an interlocking mechanism and design *interlocking furniture*, where the component parts tightly interlock with one another by a network of joints (see Figure 2). This can be achieved only by a *global* interlocking scheme rather than by an aggregate of local components [Lau et al. 2011]. Such interlocking furniture can be disassembled only through certain orders, starting from a specific *single key*, which is the only free-to-move part in the furniture assembly. This key should be taken out from the assembly in order to release the global interlocking.

Interlocking furniture has several advantages. First, the furniture can be easily assembled and disassembled repeatedly without excessive wear on its parts. Second, external fixtures such as fasteners (e.g., screws) and bearings (e.g., hinges) are not required, thus keeping the intended aesthetics of the design. In addition, since the global interlocking scheme limits the parts removal and restricts the assembly and disassembly order, component parts can tightly interlock with one another, thus facilitating the stability and strength of the furniture [Laurajaxs 2014; Jones 2014].

Given a 3D furniture design, the goal of this work is

> *To plan and construct a network of joints in the design of interlocking furniture, so that the component parts in the furniture assembly can tightly interlock with one another in a global interlocking manner.*

The input of our computational method is a furniture design consisting of just a set of simple 3D parts (e.g., rectangular boxes), whereas the output is the modified 3D parts (see Figure 1) with appropriate joint geometry that interlocks the component parts.

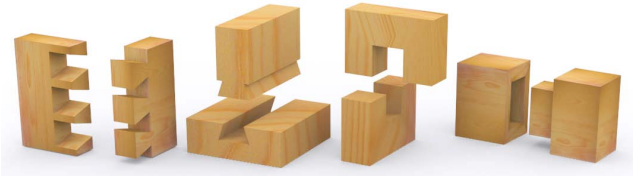


Figure 2: *Examples of four common joints: English dovetail, French dovetail, halved joints, and mortise and tenon (left to right).*

*joint first authors

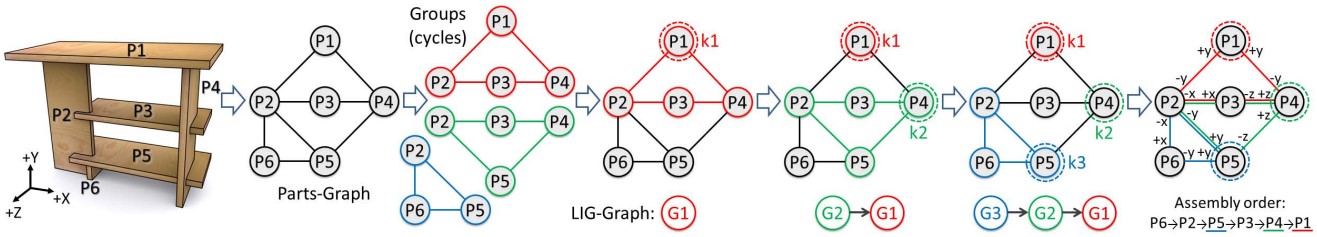


Figure 3: *Overview of our approach. Given a furniture model, we first identify overlapping cycles in its parts-graph. Then, we iteratively construct local interlocking groups (LIGs), i.e.,* G_1, G_2, *and* G_3, *with order dependency over the parts-graph by sharing certain parts (in particular a key) between adjacent LIGs. Assuming each LIG is interlocked, we can achieve global interlocking over the entire assembly.*

Given a furniture model, it is not straightforward to design a network of joints, or a network of connectors, to form a globally-interlocking assembly. State-of-the-art methods for interlocking puzzles [Xin et al. 2011; Song et al. 2012] perform well for solid compact shapes, but they cannot be used effectively for general furniture models, which are typically frame structures with elongated and planar parts. Designing the network of joints requires a *global analysis* over the component parts. Independently considering local joints is insufficient to achieve global interlocking, and this will involve a huge number of joint combinations that require an enormous effort to explore and guarantee global interlocking.

To achieve our goal, we develop a novel computational solution with the following key ideas. First, we formulate a formal model to define global interlocking. It is a group-based interlocking scheme backed up by a mathematical proof. Using this model, we can construct an overlapping set of small interlocking groups over the furniture complex, where the parts in each group are locked by a local key, and adjacent groups are further locked with dependencies. As a result, we can achieve global interlocking over all the parts with only a single mobile key, and save the enormous effort of exploring the immobilization of every subset of parts in the assembly. To put this model into practice, we first analyze the connections between adjacent parts, and determine possible joint geometries that could be deployed at each connection. We then develop an iterative method to construct small interlocking groups by assigning appropriate local joints, and progressively achieve global interlocking by further creating dependency between adjacent groups. By this means, we can create appropriate joint geometry between adjacent parts and produce an interlocking furniture assembly.

2 Related Work

3D Fabrication. Below we mainly focus on the design and fabrication of furniture. In this topic, the interest of the computer aided design community lies mainly on traditional engineering aspects that facilitate models fabrication (e.g., Gustafsson [1995] and Smardzewski [1998]), while the interest of the computer graphics community lies mainly on the furniture design: interactive methods [Saul et al. 2011] and computational tools [Lau et al. 2011].

Among the works, Lau et al. [2011] converted furniture models into fabricatable parts and connectors for constructing corresponding physical objects. Saul et al. [2011] developed a sketch-based interface for designing and building chairs fabricated from sheet materials. Umetani et al. [2012] proposed an interactive design framework for exploring valid furniture shapes under geometric and physical constraints. Schwartzburg and Pauly [2013] proposed a computational approach to generate 3D models (e.g., a chair) composed of intersecting planar pieces. Schulz et al. [2014] presented an interactive design-by-example system for fabricating objects using a parameterized template representation. Koo et al. [2014] developed an interactive system to create functioning works-like prototypes based on high-level functional relationships among 3D parts.

Fabrication by Parts. A number of related works focus on partitioning a solid 3D shape into parts for 3D fabrication. Medellín et al. [2007] subdivided a 3D shape into parts based on a regular grid. Hao et al. [2011] decomposed a large complex model into simpler 3D parts by using curvature-based partitioning. Luo et al. [2012] partitioned a large 3D object into smaller parts by planar cuts, and then connected the parts by male and female connectors. Hildebrand et al. [2013] determined the optimal printing orientation to address the directional bias issue in 3D printing. Cignoni et al. [2014] represented an input shape as planar pieces with slit-based fabrication and assembly. Vanek et al. [2014] improved the 3D printing efficiency by converting 3D objects into shells and breaking them into parts that can be glued together. Zhou et al. [2014] converted a 3D object into a set of jointed parts that can be folded into a box. Hu et al. [2014] decomposed a 2D/3D shape into a small number of parts that are approximately pyramidal.

3D Interlocking Assembly. In this work, we focus on designing an interlocking scheme to connect the furniture's component parts in a globally-interlocking fashion rather than shape (furniture) decomposition, since the furniture is given as a collection of simple logical parts. Below, we survey some recent computational methods on interlocking. The pioneering work by Cutler [1978; 1994] is based on exhaustive search to discover new six-piece interlocking burr structures; it took around two and a half years to complete the entire analysis. Instead of exhaustive search, some puzzle designers employed software such as BurrTools [Röver 2011] to assist the manual design process, i.e., using the software to test if their designs are interlocked and can be assembled. The local mobility of parts can be analyzed by blocking analysis [Li et al. 2008].

There are only a few computational methods that can construct 3D interlocking structures. Xin et al. [2011] created interlocking puzzles from 3D models by replicating and connecting a specific six-piece burr configuration, but since the method requires a central bulky structure to achieve interlocking, it is highly restrictive, and cannot deal with complex shapes of general topology. Later, Song et al. [2012] developed a recursive method to construct 3D interlocking structures by iteratively extracting polycube-shaped puzzle pieces from a general voxelized 3D shape.

The above works assume a solid compact 3D shape as input, so the internal volume of the shape is used to create the blocking geometry for achieving the interlocking. Hence, these methods cannot be applied to furniture since furniture models are typically frame structures with elongated and planar parts. Designing interlocking furniture is a more challenging task since the designed furniture should be interlocking, stable, and aesthetic for daily use, thereby involving more constraints on the design and creation.

Fabricating Self-Supporting Structures. An interlocking furniture is a self-supporting structure, since like other self-supporting structures, it holds together by itself without external support. A general self-supporting structure is composed of parts or blocks like bricks, stones, and rods. Vouga et al. [2012] proposed to interac-

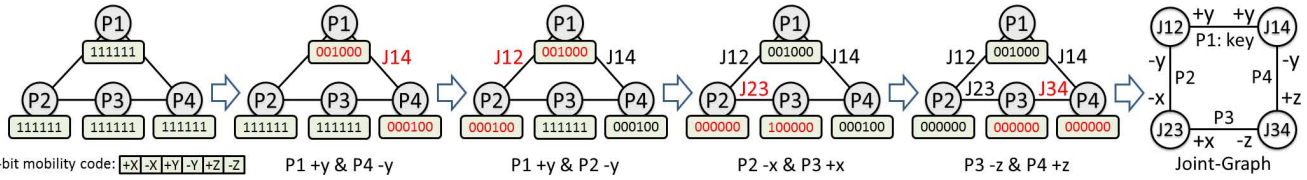


Figure 4: *We iteratively assign joints between adjacent parts, e.g., group* G_1 *in Figure 3, such that each joint immobilizes all directions of associated parts, except for one certain direction, e.g., joint* J_{14} *immobilizes* P_1 *and* P_4 *except +y and -y, respectively. The number of ON bits (in each six-bit mobility code) associated with the parts are reduced incrementally, indicating the parts interlocking in the assembly.*

tively design self-supporting structures by using the thrust network method. Panozzo et al. [2013] generated self-supporting masonry models from a user-provided height field. Liu et al. [2013] represented a 3D self-supporting surface using a 2D regular triangulation for improving its geometry processing. De Goes et al. [2013] introduced a discrete theory of equilibrium for simplicial masonry structures to facilitate their analysis and design. Song and Fu et al. [2013] proposed an interactive method for designing large reciprocal frame structures with many sloping rods. Tang et al. [2014] solved geometric and structural form-finding with self-supporting polyhedral meshes for architectural and industrial design. Deuss et al. [2014] constructed freeform self-supporting structures by using a sparse set of chains connected to fixed anchor points.

The design of all these self-supporting structures, just like interlocking furniture, requires a global analysis of all the components (i.e., blocks and parts). It should be stressed that blocks in self-supporting structures are connected by friction or mortar, while parts in interlocking furniture are connected by blocking using their own geometries in the joints. Furthermore, assembling self-supporting structures requires a dense formwork or chains, which are difficult and tedious to construct, while assembling interlocking furniture can be done easily with far less manual effort.

3 Overview

The input furniture design is given as a set of simple 3D *parts*, where adjacent parts may contact or intersect, see Figure 3 (left). To prepare such an input, we look for furniture designs by Google image search and use 3D Studio Max to arrange furniture parts with intersecting rectangular boxes to mimic a given furniture design.

The computational interlocking method that we present considers a simplified problem, where the parts potentially connect and only move along the main axes. Thus, each part is associated with a code of six bits, representing the six main axial directions, where an ON bit indicates that the part is free to move along the associated axial direction, and an OFF bit indicates that the part is immobilized along the associated axial direction. Initially, all parts of the assembly are associated with a 111111 code. Our algorithm iteratively assigns a joint between adjacent parts. Each joint immobilizes certain directions, so that *all parts*, except for a specific key, eventually possess a 000000 code, implying that they are immobilized in the assembly. Figure 4 illustrates this procedure on a group of parts, and shows how the assignment of joints incrementally reduces the number of ON bits associated with the parts of the assembly. Later on, we shall elaborate on this algorithm and formalize it.

A *parts-graph* is an undirected graph, where nodes represent furniture parts and edges connect two contacting/intersecting parts (see Figure 3 (left)). Given an initial parts-graph, we merge degree-1 nodes in the graph with their adjacent parts since these dangling nodes cannot be interlocked. We also analyze and identify groups of overlapping cycles in the parts-graph (see again Figure 3).

A *local interlocking group* (in short, LIG) is a connected subgraph in parts-graph, where parts are locked by a specific *key* in the group. That is, after we assemble the parts in a LIG (in the absence of other parts in the assembly), we cannot take out any part or any subset of parts from the LIG, unless we first take out the key, which is the only mobile part in the LIG. Thus, LIGs should include a cycle of at least three parts, so that the parts in a LIG can be interlocked.

A *joint-graph* is an undirected graph with joints as nodes and parts as edges (see Figure 4 (right)). In the joint-graph, the direction next to a node (joint) indicates the free axial direction that a joint imposes on the part. Since we consider joints that restrict parts to move along a specific axial direction, adjacent parts at a joint are restricted to move in opposite directions, e.g., $+y$ and $-y$ for P_1 and P_4, respectively, at joint J_{14} shown in the figure. Moreover, adjacent joints connected along an edge in the joint-graph should have different axial directions in order to immobilize the associated part (except the key), e.g., P_2 in Figure 4 (right): $-y$ and $-x$.

The Formal Model. We formulate the problem of computing an interlocking furniture as a problem of (i) forming LIGs that overlap one another over the parts-graph, and (ii) creating order dependency to lock adjacent LIGs (see Figure 3).

Our objective is to achieve a *global interlocking* state, where all parts and all subsets of connected parts (except the key itself) in the assembly are immobilized. This requires a global analysis to ensure that (i) every furniture part is locked by a local key in its own LIG(s), and (ii) every LIG is locked by some other adjacent LIG(s), except for the *primary LIG* (denoted by G_1). Hence, G_1 holds the *primary key* k_1 that locks the entire assembly.

An *LIG-graph* is a directed acyclic graph (denoted by $\mathbb{G}$) with LIGs as nodes (see Figure 3). Given $\mathrm{G_i}$ and $\mathrm{G_j}$ as two adjacent LIGs, there is a directed edge from $\mathrm{G_j}$ to $\mathrm{G_i}$ in $\mathbb{G}$, if $\mathrm{G_j}$ depends on (and is locked by) $\mathrm{G_i}$. That is, no part in $\mathrm{G_j}$ can be taken out before certain part(s) in $\mathrm{G_i}$ (in particular, $\mathrm{G_i}$'s key) has/have been taken out. In practice, such a dependency is created by overlapping adjacent LIGs with certain common parts (see Section 4 for detail). Except for G_1, every node in $\mathbb{G}$ should have at least one outgoing edge. Moreover, $\mathbb{G}$ should not have cycles, or cyclic dependency.

A Three-level Scheme. The formal model is a three-level global interlocking scheme: *parts*, *LIGs*, and the *entire assembly*. Based on it, we can progressively assemble the furniture based on the dependency order in $\mathbb{G}$: starting from the LIGs without incoming edges till reaching G_1 and then ending with the primary key k_1.

To approach the problem, we could ignore the middle level, and directly compute global interlocking with parts. However, directly computing interlocking is highly nontrivial since for n parts, we have to consider 2^n combinations (subsets) of parts, and ensure their immobilization (except the *key itself*) in the assembly, e.g., subsets $\{\mathrm{P}_1,\mathrm{P}_2\}$ and $\{\mathrm{P}_3,\mathrm{P}_4\}$ in Figure 4. Note that if we can partition the parts-graph into two subgraphs, such that all edges across the two subgraphs (i.e., joints between parts across the two partitions) share the same axial direction, the associated parts in a subgraph can be taken out altogether to break the assembly. Moreover, we have to explore many different choices of joints for blocking each part in the assembly. This would result in an overly large num-

ber of possible joints configurations and interlocking computation for each configuration. Hence, directly working with parts involves a huge search space, and is highly non-scalable.

Our three-level formal model overcomes the above issues by *forming and connecting small LIGs with order dependency* (see Figure 3). We can, therefore, moderate the search space in computing the interlocking locally in each LIG. Unlike the parts-graph, whose edges are undirected connections, we cannot arbitrarily break $\mathbb{G}$ since its edges are constrained to the assembly/disassembly order between adjacent LIGs. In other words, the order dependency avoids the need to consider the mobility of all LIG subsets, and the exploration of subset (of parts) immobilization. Please refer to Appendix for the related mathematical proof.

High-level Algorithm. Our computational solution has three major steps: (i) we construct a parts-graph to represent the input furniture, and then analyze and identify possible free axial directions between every pair of adjacent parts (see Section 5); (ii) we iteratively construct LIGs with appropriate joints to achieve local interlocking and order dependency among the overlapping LIGs, starting from the primary key to the entire parts-graph (see Section 6); and (iii) we generate the 3D model of each part by adapting the chosen joint geometry to modify the part (see Section 6).

4 Formal Model

This section details our formal model and the various conditions for achieving global interlocking. In Section 6, we present the procedure that plans and assigns joints in a given furniture complex based on these conditions. Let us start with the following notations:

- $\mathrm{G_i}$ is the ith LIG that we construct over a parts-graph; it has a local key $\mathrm{k_i}$ and a set of parts $\mathrm{S_i}$, for $i \in \{1, 2, ..., n\}$, where n is the total number of LIGs (nodes) in $\mathbb{G}$.
- S is the set of all the parts in G, i.e., $\mathrm{S} = \cup_{i=1}^{n} \mathrm{S_i}$, and $\mathrm{R_i}$ is the (residual) set of parts from $\mathrm{S_i}$ to $\mathrm{S_n}$, so $\mathrm{R_1} = \mathrm{S}$ and $\mathrm{R_n} = \mathrm{S_n}$.
- $\grave{\mathrm{S}}_\mathrm{i}$ and $\acute{\mathrm{S}}_\mathrm{i}$ are the subsets of parts in $\mathrm{S_i}$ that share with preceding and succeeding LIGs, respectively, i.e., $\grave{\mathrm{S}}_\mathrm{i} = \mathrm{S_i} \cap (\cup_{j=1}^{i-1} \mathrm{S_j})$ and $\acute{\mathrm{S}}_\mathrm{i} = \mathrm{S_i} \cap \mathrm{R_{i+1}}$. So, $\grave{\mathrm{S}}_1 = \emptyset$ and $\acute{\mathrm{S}}_\mathrm{n} = \emptyset$.
- $\dot{\mathrm{S}}_\mathrm{i} = \mathrm{S_i} \backslash (\grave{\mathrm{S}}_\mathrm{i} \cup \acute{\mathrm{S}}_\mathrm{i})$ are parts in $\mathrm{G_i}$, not shared with other LIGs.
- Given two adjacent parts $\mathrm{P_A}$ and $\mathrm{P_B}$ connected by a joint (see Figure 9 for examples), we denote $d(\mathrm{P_A}, \mathrm{P_B})$ as the free axial direction of $\mathrm{P_A}$ imposed by its joint connection with $\mathrm{P_B}$, so $d(\mathrm{P_A}, \mathrm{P_B})$ = -$d(\mathrm{P_B}, \mathrm{P_A})$, since $\mathrm{P_A}$ and $\mathrm{P_B}$ move in opposite directions in order to separate from each other.

4.1 Conditions for Local Interlocking

To achieve local interlocking, each LIG should include a cycle of parts. In this subsection, we present conditions for a cycle of parts to be local interlocking with k as its key (see below). Here we focus on 3- and 4-part cycles since they dominate in common furniture. Note that we may still formulate conditions for cycles with five or more parts, but in practice, these cycles are rare in common furniture and they may be decomposed into cycles with fewer parts.

Case i: a 3-part cycle (see Figure 5 (left)):

- $d(\mathrm{k}, \mathrm{P_A}) = d(\mathrm{k}, \mathrm{P_B})$, so k is mobile in the assembly.

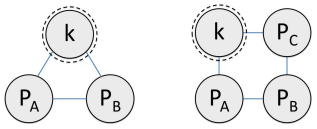


Figure 5: *Groups (cycles) with 3 parts (left) and 4 parts (right).*

- Every non-key part (i.e., $\mathrm{P_A}$ and $\mathrm{P_B}$) should be immobilized. Hence, $d(\mathrm{P_A}, \mathrm{P_B}) \neq d(\mathrm{P_A}, \mathrm{k})$ to immobilize $\mathrm{P_A}$, so that $\mathrm{P_A}$ cannot be separated from $\{\mathrm{P_B}, \mathrm{k}\}$ in the 3-part assembly. Similarly, $d(\mathrm{P_B}, \mathrm{P_A}) \neq d(\mathrm{P_B}, \mathrm{k})$ to immobilize $\mathrm{P_B}$.

Case ii: a 4-part cycle (see Figure 5 (right)):

- Again, $d(\mathrm{k}, \mathrm{P_A}) = d(\mathrm{k}, \mathrm{P_C})$, so k is mobile in the assembly.
- Every non-key part should be immobilized: For $\mathrm{P_A}$, $d(\mathrm{P_A}, \mathrm{k}) \neq d(\mathrm{P_A}, \mathrm{P_B})$; for $\mathrm{P_B}$, $d(\mathrm{P_B}, \mathrm{P_A}) \neq d(\mathrm{P_B}, \mathrm{P_C})$; and for $\mathrm{P_C}$, $d(\mathrm{P_C}, \mathrm{P_B}) \neq d(\mathrm{P_C}, \mathrm{k})$.
- Every subset of two adjacent parts should be immobilized. We immobilize $\{\mathrm{k}, \mathrm{P_A}\}$ by $d(\mathrm{k}, \mathrm{P_C}) \neq d(\mathrm{P_A}, \mathrm{P_B})$, so k and $\mathrm{P_A}$ cannot be separated together from the assembly. Similarly, $d(\mathrm{k}, \mathrm{P_A}) \neq d(\mathrm{P_C}, \mathrm{P_B})$ for $\{\mathrm{k}, \mathrm{P_C}\}$. Due to the subset complement[1], we do not need to check $\{\mathrm{P_B}, \mathrm{P_C}\}$ and $\{\mathrm{P_A}, \mathrm{P_B}\}$.

For 4-part groups, we do not need to consider triplets since triplets are complements of a single part in the assembly (similarly, two-part subsets in 3-part groups). Moreover, we do not consider subsets of non-adjacent parts, e.g., $\{\mathrm{P_A}, \mathrm{P_C}\}$, since these subsets are immobilized by their (immobilized) individual parts.

4.2 Expanding a LIG

We discover a rule for expanding an existing LIG: given a LIG $\mathrm{G_i}$ and a part $\mathrm{P'} \notin \mathrm{S_i}$ of $\mathrm{G_i}$, $\mathrm{S_i'} = \mathrm{S_i} \cup \{\mathrm{P'}\}$ is also a LIG if the following conditions are fulfilled in the assembly of $\mathrm{S_i'}$: i) $\mathrm{P'}$ is immobilized in all directions by some parts in $\mathrm{S_i}$; ii) $\{\mathrm{P'}, \mathrm{k_i}\}$ is also immobilized in $\mathrm{S_i}$, meaning that $\mathrm{P'}$ and $\mathrm{k_i}$ cannot be taken out together from the assembly of $\mathrm{S_i'}$; and iii) $\mathrm{k_i}$ is still mobile in $\mathrm{S_i}$.

By this rule, after we construct a LIG, we may add a new part to expand the LIG without intensive interlocking computation that involves all the parts of the LIG. Note that this rule can be proved by considering three disjoint groups of subsets of $\mathrm{S_i'}$: i) subsets of $\mathrm{S_i'}$ without $\mathrm{P'}$; ii) $\{\mathrm{P'}\}$ and $\{\mathrm{P'}, \mathrm{k_i}\}$; and iii) $\{\mathrm{P'}\} \cup \mathrm{X}$, where X is any nonempty subset of $\mathrm{S_i}$, excluding $\{\mathrm{k_i}\}$ and $\mathrm{S_i} \backslash \{\mathrm{k_i}\}$.

4.3 Conditions in LIG Construction

Next, we formulate the conditions in our formal model for constructing LIGs with dependency in assembly/disassembly order. The formulation is nontrivial since we have to guarantee *local interlocking* within each LIG, *global interlocking* over all the parts in the assembly, as well as *assemblability*, meaning that the structure can be assembled part by part (see Appendix for the proof). Below, we present the conditions in three cases:

Case i: Construct $\mathbf{G_1}$. The first one is a basic case: since $\mathrm{G_1}$ is the first LIG being constructed, it does not depend on others.

- **Local Interlocking:** $\mathrm{G_1}$ is local interlocking with key $\mathrm{k_1}$.
- **Avoid Mutual Dependency:** $\mathrm{k_1}$ should not be shared with any subsequent LIG, i.e., $\mathrm{k_1} \notin \acute{\mathrm{S}}_1$, to avoid mutual dependency or cycles in $\mathbb{G}$.
- **Assemblability[2]:** First, we should be able to take out $\mathrm{k_1}$ from the full assembly. Second, we should be able to subsequently take out from the assembly all the parts in "$\mathrm{S_1} \backslash \{\mathrm{k_1}\}$" that are not in $\mathrm{R_2}$, so the remaining parts are just $\mathrm{R_2}$.

[1] If a subset of parts is immobilized (cannot move together to separate from the remaining parts) in an assembly, its complement in the assembly is also immobilized (cannot move together in the opposite direction).

[2] Note that if we are able to disassemble the entire assembled structure part by part started from $\mathrm{k_1}$, we should also be able to assemble the same structure part by part until putting in $\mathrm{k_1}$, as the last assembly step. Hence, we explore assemblability by looking at the parts disassembly sequence.

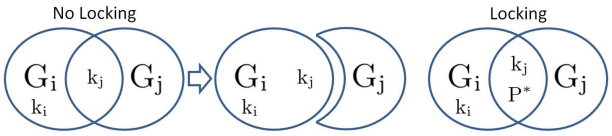


Figure 6: *Left: There is no dependency (locking) between* G_i *and* G_j *if they share only a key; we may separate* G_i *altogether from* $G_j \backslash \{k_j\}$. *Right: we propose to share a key and at least one non-key part* P^*, *so that one of the groups* (G_i) *can lock the other* (G_j).

See Figure 3 for an example G_1, whose key is not shared with other LIGs to avoid mutual dependency; see the joints configuration in Figure 4, G_1 is local interlocking while it can be assembled.

Case ii: Construct G_j that shares parts with only one previously-constructed LIG, say G_i (j>i), but not others. Besides the first two conditions in case i, we have the following conditions for constructing G_j; in particular, G_j should depend on G_i:

- **New Part(s):** At least one part in G_j is not included in any previously-constructed LIGs, i.e., $|S_j| > |\grave{S}_j|$, so that we can construct LIGs that eventually cover all the parts.
- **Assemblability:** Before we unlock (take out local key k_j) and disassemble G_j, we should have unlocked all the LIGs that are constructed before G_j and taken out from the entire assembly all the parts that are not in R_j. Now, similar to case i, we ensure that k_j can be taken out from R_j, subsequently followed by all the parts (taken out one by one) in "$S_j \backslash \{k_j\}$" that are not in R_{j+1}, so the remaining parts are just R_{j+1}.
- **Dependency on G_i:** G_j should share at least two parts (including k_j) with G_i, so that every subset of parts in G_j becomes immobilized until we unlock G_i by taking out k_i.

Take an assembly of two LIGs (G_i and G_j) as an example. If they share only one part, which is k_j, parts in G_i may move altogether with k_j and separate from G_j even though we have not unlocked G_i by taking out its key (see Figure 6 (left)). By sharing an additional non-key part between them (see P^* in Figure 6 (right)), we cannot separate G_i from G_j since $\{k_j, P^*\}$ is an immobilized subset in G_j, which is a LIG. Moreover, k_i should not be shared with G_j to avoid mutual dependency (see the second condition in case i).

Case iii: Construct G_j that shares parts with multiple previously-constructed LIGs. This case is the same as case ii, except for the following condition:

- **Dependency.** G_j should depend on at least one previously-constructed LIG by sharing its local key k_j and at least one non-key part of it with a previously-constructed LIG.

Note that G_j may depend on more than one previously-constructed LIG by sharing its key and a non-key part as above; here we may share different non-key parts of G_j with these previous LIGs.

Assembly/Disassembly order. By constructing LIGs based on the formal model, the parts disassembly order naturally follows the LIG construction order: after taking out k_1 from the full assembly, we can take out parts in $S_1 \backslash R_2$, k_2, parts in $S_2 \backslash R_3$, etc., until S_n. Reversing this gives the parts assembly order, see Figure 3 (right).

5 Joint Analysis

To apply the formal model to create an interlocking furniture assembly given a furniture design, we first need to analyze the part connections in the design to find out possible joint geometries (each with a free axial direction) that can be deployed at each connection (Section 5) before we proceed to construct LIGs (Section 6).

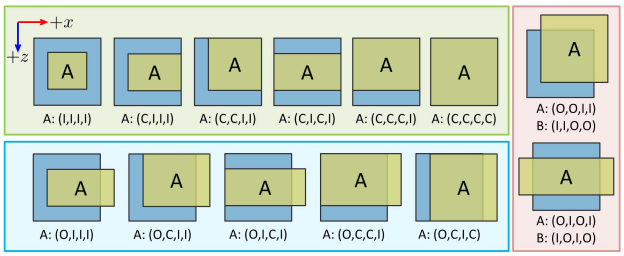


Figure 7: *The* 13 *cases of ICO vector* (A), *in which 2D boxes* A *(yellow) and* B *(blue) intersect; ordered by* $+x$, $-z$, $-x$, *and* $+z$.

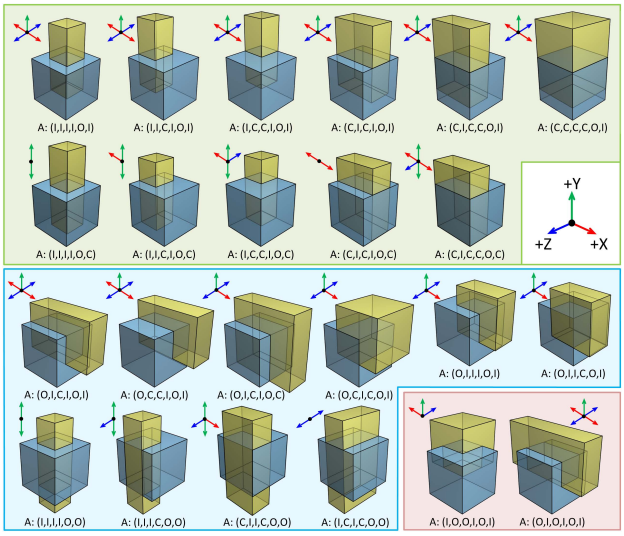


Figure 8: *The* 23 *cases of ICO vector* (A), *in which 3D boxes* A *(yellow) and* B *(blue) intersect: ordered by* $+x$, $-z$, $-x$, $+z$, $+y$, *and* $-y$. *The arrows show the possible axial movement directions of* A *over all joint geometries that we may choose to use.*

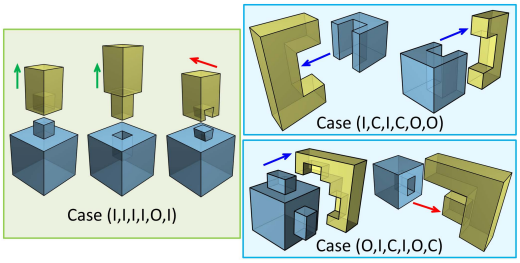


Figure 9: *Example joint geometries for some cases in Figure 8.*

5.1 Joint Lookup Table

To support efficient joint analysis, we prepare a precomputed table called the *joint lookup table*, which is indexed by the ICO vector:

ICO (Inside-Common-Outside) Vector. Given two axis-aligned boxes A and B with intersection volume V_{AB}, their ICO vectors have six elements, corresponding to main axial directions. Taking the $+x$ direction as an example with f_A and f_B as the $+x$ faces of A and B, respectively, the $+x$ element of A's ICO vector is:

$$\begin{cases} \text{I} & \text{if } f_A \text{ (but not } f_B\text{) is on } +x \text{ face of } V_{AB}, \\ \text{C} & \text{if both } f_A \text{ and } f_B \text{ are on } V_{AB}, \text{ and} \\ \text{O} & \text{if } f_A \text{ is not on } V_{AB}. \end{cases}$$

By repeating the above check for other faces, we can form A's ICO vector (see Figure 7 for cases with 2D boxes). Note that the ICO vectors of A and B are complements of each other (see Figure 7 (right)): interchanging I and O and keeping C unchanged.

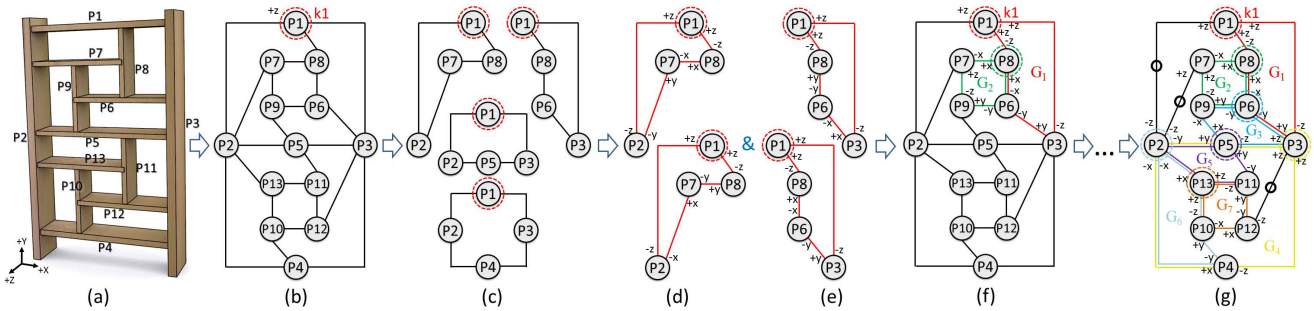


Figure 10: *(a) a given furniture design; (b) in its parts-graph,* P_1 *is the primary key* (k_1) *and* $+z$ *is* P_1*'s removal direction* ($\mathrm{d}(\mathrm{k}_1)$)*; (c) cycles that contain* P_1 *in the parts-graph; (d)&(e) some candidates of* G_1 *(red) produced from two of the cycles; the assigned axial directions among the joints make each of these cycles into a LIG; (f)* G_2 *(green) constructed after one of the candidate of* G_1*; and (g) one of the global interlocking configurations produced by our method; we iteratively construct the seven LIGs over the graph:* G_1 *to* G_7 *(in different colors).*

Returning to 3D boxes A and B, if they intersect, there are 23 unique and valid cases of ICO vector (see Figure 8). These cases are obtained by considering i) rotational symmetry and mirror reflection, and ii) ICO vector complement: if A's ICO vector has more O than I, we swap the role of A and B to reduce the number of cases. For 3D boxes, cases without O are invalid since one of the boxes is contained by the other.

On the other hand, if A and B just contact (without intersection), we compute A's ICO vector in the 2D space extended from the contact surface. By the same considerations as the 3D cases, we can obtain 13 cases (see Figure 7).

Joint Lookup Table is indexed by the 23+13 ICO vectors above. To prepare it, we manually find out the possible joint geometries that we may choose to use at the connection of each case (see Figure 9 for examples). These joint geometries are constructed in the 3D space within or near V_{AB} based on common joint models: dovetail joint, halved joint, and mortise and tenon [Graubner 1992] (see Figure 2). Each joint geometry restricts the connected parts to separate from each other along a certain (single) axial direction.

5.2 Candidate Joints and Free Axial Directions

Given an input furniture design, we analyze each of its connections between adjacent parts by using the following two cases:

Case i: The two parts intersect. First, we compute the ICO vector of one of the two parts. If the vector has more O than I, we replace it by its complement. We then apply axis-to-axis transformations (rotation and mirror reflection) to obtain 24 copies of the vector in different 3D orientations, and match each copy against the six-elements ICO vectors in the joint lookup table. When a match is found, we can look up the joint geometries (with associated free axial directions), and then transform this information to the object space of the furniture by an inverse of the associated axis-to-axis transformation that we applied earlier. Hence, we can determine possible joint geometries and free axial directions for the given connection. If we cannot find a match, such a connection is invalid.

Case ii: The two parts contact. Case ii is similar to case i, except that we compute a four-elements ICO vector on the 2D contact surface and generate eight copies of the vector in 2D.

6 Iterative Procedure to LIG Construction

After the joint analysis, we know the possible joint geometries that can be deployed at each connection (joint) between parts. Moreover, we also know at each joint, a set of axial directions that each part at the joint can choose to have; we denote $\mathrm{D}(A, B)$ as the set of axial directions that part A can have at its joint with part B. Later on, when we plan a joint for forming a LIG (see Figure 4), we can select one axial direction from the joint's associated $\mathrm{D}(A, B)$; such a direction corresponds to a certain joint geometry (see again Figure 9). By this information and the formal model, the following procedure can plan the joints with appropriate geometry and iteratively construct small LIGs with order dependency.

6.1 G_1 Construction

Our procedure starts by forming the first LIG, which is G_1. This involves the following three steps:

Step 1) Confirm k_1. Given a user-specified primary key (k_1) with an intended removal direction, say $\mathrm{d}(\mathrm{k}_1)$, we need to examine the joints connected with k_1, and check to see if all these joints consistently allow $\mathrm{d}(\mathrm{k}_1)$ as the free axial direction to move k_1 out of the assembly. That is, whether $\mathrm{d}(\mathrm{k}_1) \in \mathrm{D}(\mathrm{k}_1, \mathrm{b})$ for every neighboring part b of k_1. For example, for P_1 in Figure 10(b): $\mathrm{d}(\mathrm{k}_1) = +z \in \mathrm{D}(\mathrm{P}_1, \mathrm{P}_2) \cap \mathrm{D}(\mathrm{P}_1, \mathrm{P}_8) \cap \mathrm{D}(\mathrm{P}_1, \mathrm{P}_3)$. In this way, we can choose the corresponding geometry for each of these joints, so that k_1 can be taken out from the assembly along $\mathrm{d}(\mathrm{k}_1)$.

Step 2) Construct G_1. We then obtain all the 3- and 4-part cycles that contain k_1 in the parts-graph. Since we have already chosen the geometry for the joints connected with k_1 in step 1, two of the joints in each of the cycles have been fixed and we only have one or two free joints left (depending on 3- or 4-part cycles). Taking the top-left cycle in Figure 10(c) as an example, joints P_1-P_2 and P_1-P_8 have been fixed, and so, we only need to decide the geometry of joints P_2-P_7 and P_7-P_8 to make the cycle into a LIG. Here we have to consider the following two requirements. First, we have to choose the free axial direction (and geometry) of the free joints from the associated $\mathrm{D}(A, B)$. Second, we have to ensure the cycle to be locally interlocking by the conditions formulated in Section 4.1. See Figure 10(d) for some results. After making a cycle into a LIG, we may apply the rule in Section 4.2 to expand the LIG.

Since there are only a few cycles that contain k_1, and only a few choices of axial directions ($\mathrm{D}(A, B)$) for each free joint in a cycle, we exhaustively find out all joint combinations that make each cycle to be a LIG. Hence, the output of this step is a set of LIGs as candidates of G_1 (see Figure 10(d)&(e) for examples).

Step 3) Verify Assemblability. Lastly, we test the assemblability of each candidate G_1 and prune away the candidates whose k_1 is blocked to be removed from the assembly. This is done by checking k_1's trajectory along $\mathrm{d}(\mathrm{k}_1)$ against other parts in the furniture. Note that we also consider and check secondary translational movement along axial directions perpendicular to $\mathrm{d}(\mathrm{k}_1)$, after k_1 is stopped by some parts in its primary movement along $\mathrm{d}(\mathrm{k}_1)$. After k_1, we also test other parts in the candidate LIG to ensure the fulfillment of the assemblability condition (see case i of Section 4.3).

Figure 11: *Interlocking furniture assemblies we produced (from left to right):* BOOKSHELF, BABY BED, SHOE RACK, BENCH, BEDSTAND, CONSOLE TABLE, SOFA, CHAIR, CHILD BED, *and* MULTI-FUNCTION TABLE. *Please refer to supp. video for their assembly animations.*

6.2 Construction of Subsequent LIGs

In practice, furniture assemblies do not possess an extended number of cycles in their individual parts-graph, even for more complicated furniture models, e.g., CHILD BED shown in Figure 11. Hence, by the formal model, we may exhaust every possible joint configuration that makes the furniture assembly to be global interlocking. By our method, the generation of each configuration is linear to the number of joints, and the computation load is not high, as evidenced by the timing statistics presented in Section 7. In detail, for each candidate G_1, we perform the following steps to iteratively produce subsequent LIGs based on the formal model and generate global interlocking joint configurations:

Step 1) Identify cycles and local keys. To construct G_2 (or in general G_{i+1} for $i \geq 1$), we find all 3- and 4-part cycles that share at least two parts with one of the groups we constructed previously, as well as contain at least one *new* part not in any previous group. This is to fulfill the dependency and new part(s) conditions for cases ii and iii in Section 4.3. Moreover, these cycles should not include local keys from any previous group to avoid mutual dependency.

For each cycle, we check each shared part to see if it can serve as a local key by examining its connecting joints. This is similar to step 1 when constructing G_1, but we test only the joints between parts in R_{i+1} since parts not in R_{i+1} should have been removed when we were about to disassemble the parts in G_{i+1} (based on the formal model); note also that some joints of the candidate key should have been fixed by the cycles of previous groups.

Step 2) Construct next LIG. For each shared part that can serve as a key in each cycle, we identify free joints in the cycle, plan each free joint, etc., by the procedure as in step 2 of G_1 construction. By then, we can produce a new LIG with order dependency, see Figure 10(f) for an example: G_2 (green) depends on G_1 (red).

Step 3) Assemblability. Similar to step 3 in G_1 construction, we verify the assemblability of each configuration, e.g., the one in Figure 10(f). However, when we examine successive groups after G_1, say G_i, we only need to explore parts in G_i and R_{i+1}, see the assemblability condition in case ii of Section 4.3.

After constructing G_2, we iterate the above steps to construct more LIGs until they cover all the furniture parts, see LIGs of different colors in Figure 10(g). Note that when we exhaust the choices of cycles and local keys, some branches we explore may fail to complete since we may not be able to find an appropriate cycle, a local key, or an interlocking joint configuration for every candidate cycle.

Post-processing. Some joints in a resulting configuration may not be included in any constructed group, see P_1-P_2 and edges marked with a circle in Figure 10(g) for examples. To complete the joint network, we assign a joint geometry to each of these free joints by examining the parts disassembly order and selecting an appropriate joint based on the removal direction of associated parts at the joint.

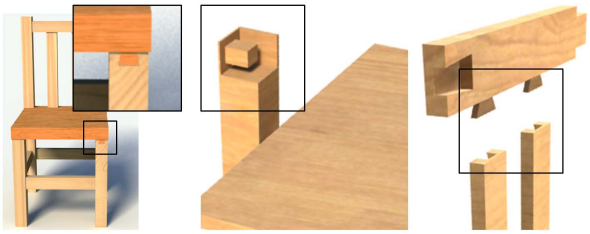


Figure 12: *Joint pattern (left) revealed on furniture exterior. Weak joint geometry with mortise and tenon (middle) and dovetail (right).*

6.3 Production of an Interlocking Furniture

After the above iterative procedure produces a large number of global interlocking configurations, our method looks for those with a large variation of joint types but without weak/thin joint geometry (see Figure 12 (middle & right)), and takes them as candidates (in practice, 20) for users to pick, e.g., users may avoid revealing joint patterns on furniture's exterior, especially the front, see Figure 12 (left). Moreover, we may label some parts for our method to choose the configurations, such that the labeled parts are assembled earlier, e.g., two of the long supporting parts in SHOE RACK, see Figure 11.

Lastly, to produce the 3D model of each part in the interlocking assembly, we use the constructive solid geometry (CSG) functions in 3DS Max script to adapt the joint geometries onto their associated parts as planned by the iterative procedure.

7 Results

Figure 11 showcases all the interlocking furniture assemblies produced by our method. Since this figure only shows static images of completed assemblies, we present in Figure 13 the disassembly sequence of SHOE RACK. Each snapshot image in the figure shows one disassembly step with the associated joint(s) highlighted in a small inset view next to the snapshot. Furthermore, we show the assembly sequences of CHAIR and CONSOLE TABLE in Figure 15, and snapshots of assembling CHILD BED in Figure 17. Please refer to the supplementary video for their animations.

In addition, to help reveal the global interlocking state in the interlocking assemblies, we present the parts-graphs together with the joint information (the chosen axial directions that each joint imposes on its associated parts) in various figures: CONSOLE TABLE

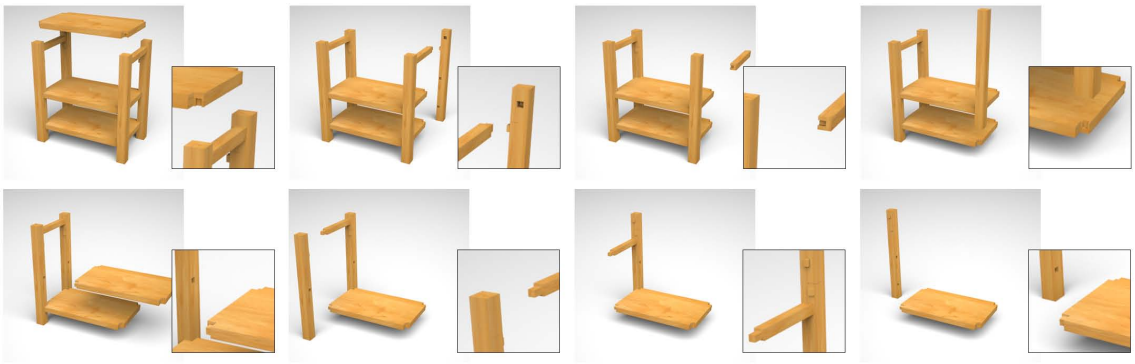


Figure 13: *Snapshots showing the disassembly of* SHOE RACK*; see the zoom-in views for the joint geometry in each disassembly step.*

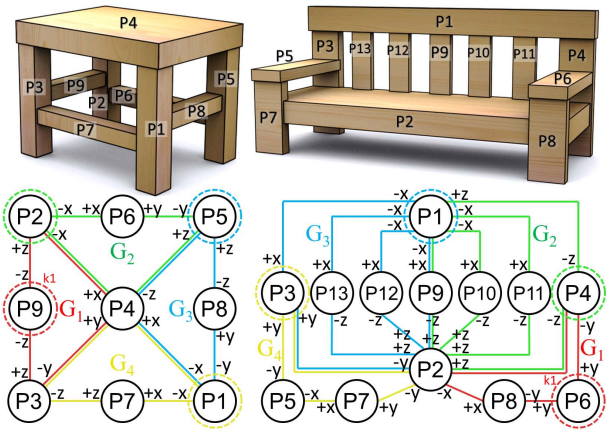


Figure 14: BENCH *and* SOFA. *The top row shows the labels of parts, while the bottom row shows the parts-graphs with LIGs and joint information using the same style as in Figure 10(g).*

	# Parts	# Joints	# LIGs	# Valid configurations	Timing (sec)
CONSOLE TABLE	6	8	3	87	0.02
BEDSTAND	6	9	3	46	0.07
BENCH	9	12	4	112	0.38
SHOE RACK	9	16	5	4488	18.89
MULTI-FUNCTION TABLE	10	18	5	124	2.29
CHAIR	12	18	6	320	13.16
SOFA	13	20	4	8766	5.9
BOOKSHELF	13	22	6	97808	62.3
BABY BED	23	42	6	43160	48.98
CHILD BED	32	66	8	15492	157.68

Table 1: *Statistics about the resulting furniture in Figure 11.*

	# Total Trial	# Non-interlocking	# Non-assemblable	# Valid configurations	Timing (sec)
CONSOLE TABLE	10^6	68184	30523	1293	6.18
BENCH	10^6	72049	27728	223	47.8
SHOE RACK	10^6	72573	27424	3	73.83
MULTI-FUNCTION TABLE	10^6	68481	31519	0	135.24
BOOKSHELF	10^6	82853	17147	0	1359.74
BABY BED	10^3	7937	2063	0	12906.2

Table 2: *Results: generating global interlocking (valid) configurations with the randomized baseline method.*

in Figure 3 (rightmost), BOOKSHELF in Figure 10(g), and BENCH and SOFA in Figure 14. These graphs show individual local interlocking groups (LIGs), how they share local keys, and how they lock with one another. Note also the parts-graph of BENCH shown in Figure 14; it reveals how the rule in Section 4.2 helps expand a LIG with additional parts in the assembly, e.g., G_2 and G_3.

Moreover, we may manually choose different parts of a furniture model as its primary key. For example, we choose the top part of CONSOLE TABLE (see Figure 3) as the key, but for MULTI-FUNCTION TABLE, its key is a bottom part on the lower-left (see Figure 1). For CONSOLE TABLE, we create another configuration of it with a lower part as the key: P_6 (see labels in Figure 3). We fabricated this alternative design and show it in Figure 16 (top).

Statistics and Performance. We implemented our method in C++ and executed it on a desktop PC with a 3.4GHz CPU and 8GB memory. Table 1 presents the statistics of our results (see corresponding models in Figure 11), including the number of parts, joints and LIGs, the number of global interlocking (valid) configurations produced, and the total time taken to produce each resulting interlocking furniture. In general, the performance depends on the number of parts and joints, as well as the complexity of the parts-graph.

Evaluation. Given a furniture model with n joints and $\sim k$ choices of valid joint geometry at each joint, we have $\sim k^n$ different joint configurations in total. Among them, only a small portion of configurations are valid, i.e., global interlocking while can be assembled. To evaluate our method against a baseline, we develop a simple randomized method to generate a large number of joint configurations by randomly picking valid joint geometries at each joint. By this, we can estimate the chance and time taken to produce global interlocking configurations with the randomized method and compare it with our method. From Table 2, we can see that the randomized method can only produce results for models with a few parts, and its time taken increases dramatically with the number of parts. Lastly, it is worth to note that since the randomized method tries out joint combinations randomly, it may discover global interlocking configurations that our method cannot produce, since our method generates global interlocking configurations by connecting small LIGs.

3D Fabrication. We fabricate physical models of some of the resulting interlocking assemblies, see Figure 16. These models are created by 3D-printing individual parts of the assembly by using the MakerBot FDM printer with the PLA plastic material. Upon the assembly, all the parts tightly interlock with one another according to the joints network planned by the LIG construction process.

Limitations. First, we cannot achieve global interlocking for certain furniture structures, e.g., a model with a long horizontal part in the middle and two independent groups of parts on each end of the long part. If parts between the two groups do not interact (contact/intersect) with one another, we cannot achieve global interlocking over the entire assembly, but can only arrange joints to locally interlock parts in each group. Second, our method currently considers only cycles of up to 4 parts and orthogonal connections between neighboring parts. Third, we employ only common joint models (e.g., see Figure 2) but not more advanced ones. Though we can handle general three-way joints by modeling them as three two-way joints, we cannot handle a special three-way joint situation: three orthogonal rods intersect one another within a common

Figure 15: *The assembly of* CHAIR *(top) and* CONSOLE TABLE *(bottom).*

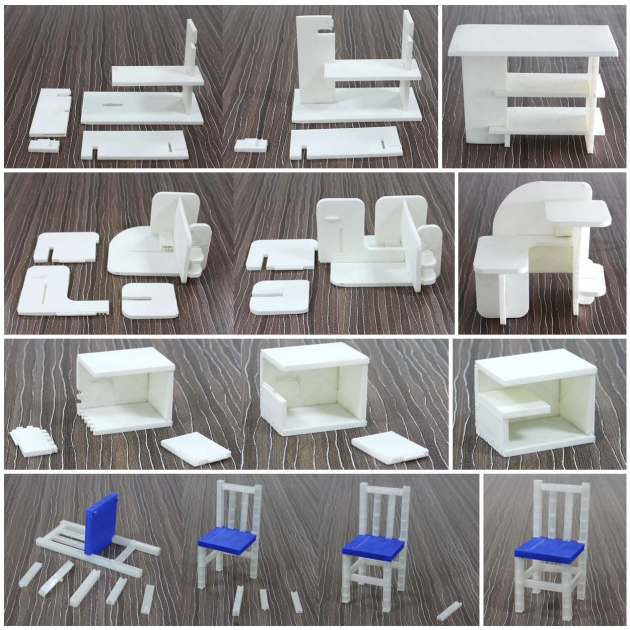


Figure 16: *3D fabrication results:* CONSOLE TABLE*,* MULTI-FUNCTION TABLE*,* BEDSTAND*, and* CHAIR *(from top to bottom).*

cubical volume in the middle. In this case, we need to employ a special joint model called the puzzle joint to interlock the three rods. Lastly, we did not consider structural stability in the assembly, e.g., putting a load on the structure; we leave this as a future work.

8 Conclusion

This paper presents a computational solution to support the making of interlocking furniture assemblies. Taking a design composed of just simple 3D shapes, our method can automatically plan a joints network over the furniture, so that its component parts tightly interlock with one another in a global interlocking assembly. By this approach, users can focus on the design and appearance of the furniture rather than on the joints scheme and their geometry.

There are two major contributions in this work. First, we develop a formal model to define and achieve global interlocking in a joints network. This is a novel model, enabling us to break a furniture complex into an overlapping set of groups: parts in each group locally interlock with one another by a single key, and adjacent groups are further locked with dependencies. By this three-level model, we can save the effort of exploring the immobilization of every subset of parts in the assembly, and localize the intensive interlocking computation within each small group. Our mathematical proof shows that furniture assemblies constructed according to this model are guaranteed to be global interlocking. Second, we put the formal model into practice by developing a computational method. Given a furniture design, we analyze the connections of its parts efficiently by using the joint lookup table and ICO vectors. We then develop an iterative method that progressively constructs small interlocking groups with overlapping parts while considering other factors such as local joint geometry and joint appearance. By this method, we can construct appropriate joints over the furniture's component parts, and create many different global interlocking furniture assemblies of assorted appearance and complexity.

In the future, we would like to support more different joint models, and study their parameters and strength in connecting parts. Moreover, we plan to study the stability of the assembled structure based on the choice of joints, the joint parameters, the joints network, and the load that would be put onto the assembly. Lastly, we would like to support non-orthogonal connections among parts and apply our method to large-scale structures such as reciprocal frames.

Acknowledgments

We thank reviewers for the valuable comments, Michael Brown for his voice over, Hadar Elor for proofreading, and Zhongqi Fu for the 3D printing. This work is supported in part by the Singapore MOE Tier-2 grant (MOE2011-T2-2-041), Israel Science Foundation, and the National Natural Science Foundation of China (61403357).

References

CIGNONI, P., PIETRONI, N., MALOMO, L., AND SCOPIGNO, R. 2014. Field-aligned mesh joinery. *ACM Trans. Graph. 33*, 1. Article 11.

CUTLER, W. H. 1978. The six-piece burr. *Journal of Recreational Mathematics 10*, 4, 241–250.

CUTLER, W. H., 1994. A computer analysis of all 6-piece burrs. Self published.

DE GOES, F., ALLIEZ, P., OWHADI, H., AND DESBRUN, M. 2013. On the equilibrium of simplicial masonry structures. *ACM Trans. Graph. (SIGGRAPH) 32*, 4. Article 93.

DEUSS, M., PANOZZO, D., WHITING, E., LIU, Y., BLOCK, P., SORKINE-HORNUNG, O., AND PAULY, M. 2014. Assembling

Figure 17: *Some snapshots showing the assembly of* CHILD BED.

self-supporting structures. *ACM Trans. Graph. (SIGGRAPH Asia) 33*, 6. Article 214.

GRAUBNER, W. 1992. *Encyclo. of Wood Joints*. Taunton Press.

GUSTAFSSON, S. I. 1995. Furniture design by use of the finite element method. *Holz als Roh- und Werkstoff 53*, 4, 257–260.

HAO, J., FANG, L., AND WILLIAMS, R. E. 2011. An efficient curvature-based partitioning of large-scale stl models. *Rapid Prototyping Journal 17*, 2, 116–127.

HILDEBRAND, K., BICKEL, B., AND ALEXA, M. 2013. Orthogonal slicing for additive manufacturing. *Computers & Graphics (SMI) 37*, 6, 669–675.

HU, R., LI, H., ZHANG, H., AND COHEN-OR, D. 2014. Approximate pyramidal shape decomposition. *ACM Trans. Graph. (SIGGRAPH Asia) 33*, 6. Article 213.

JONES, D., 2014. Interlocking chair. `http://www.offi.com/products/offikids/PAS-CHAIR.php?p2c=679`.

KOO, B., LI, W., YAO, J., AGRAWALA, M., AND MITRA, N. J. 2014. Creating works-like prototypes of mechanical objects. *ACM Trans. Graph. (SIGGRAPH Asia) 33*, 6. Article 217.

LAU, M., OHGAWARA, A., MITANI, J., AND IGARASHI, T. 2011. Converting 3D furniture models to fabricatable parts and connectors. *ACM Trans. Graph. (SIGGRAPH) 30*, 4. Article 85.

LAURAJAXS, 2014. Interlocking furniture. `http://laurajaxs.hubpages.com/hub/interlocking-furniture`.

LI, W., AGRAWALA, M., CURLESS, B., AND SALESIN, D. 2008. Automated generation of interactive 3D exploded view diagrams. *ACM Trans. Graph. (SIGGRAPH) 27*, 3. Article 101.

LIU, Y., HAO, P., SNYDER, J., WANG, W., AND GUO, B. 2013. Computing self-supporting surfaces by regular triangulation. *ACM Trans. Graph. (SIGGRAPH) 32*, 4. Article 92.

LUO, L., BARAN, I., RUSINKIEWICZ, S., AND MATUSIK, W. 2012. Chopper: Partitioning models into 3D-printable parts. *ACM Trans. Graph. (SIGGRAPH Asia) 31*, 6. Article 129.

MEDELLÍN, H., LIM, T., CORNEY, J., RITCHIE, J. M., AND DAVIES, J. B. C. 2007. Automatic subdivision and refinement of large components for rapid prototyping production. *Journal of Comp. and Info. Science in Eng. 7*, 3, 249–258.

PANOZZO, D., BLOCK, P., AND SORKINE-HORNUNG, O. 2013. Designing unreinforced masonry models. *ACM Trans. Graph. (SIGGRAPH) 32*, 4. Article 91.

POSTELL, J. 2012. *Furniture Design*. Wiley, 2nd edition.

RÖVER, A., 2011. Burr tools. `burrtools.sourceforge.net`.

SAUL, G., LAU, M., MITANI, J., AND IGARASHI, T. 2011. SketchChair: An all-in-one chair design system for end users. In *TEI*, 73–80.

SCHULZ, A., SHAMIR, A., LEVIN, D. I. W., SITTHI-AMORN, P., AND MATUSIK, W. 2014. Design and fabrication by example. *ACM Trans. Graph. (SIGGRAPH) 33*, 4. Article 62.

SCHWARTZBURG, Y., AND PAULY, M. 2013. Fabrication-aware design with intersecting planar pieces. *Computer Graphics Forum (Eurographics) 32*, 2, 317–326.

SMARDZEWSKI, J. 1998. Numerical analysis of furniture constructions. *Wood Science and Technology 32*, 4, 273–286.

SONG, P., FU, C.-W., AND COHEN-OR, D. 2012. Recursive interlocking puzzles. *ACM Trans. Graph. (SIGGRAPH Asia) 31*, 6. Article 128.

SONG*, P., FU*, C.-W., GOSWAMI, P., ZHENG, J., MITRA, N. J., AND COHEN-OR, D. 2013. Reciprocal frame structures made easy. *ACM Trans. Graph. (SIGGRAPH) 32*, 4. Article 94. (* joint first authors).

TANG, C., SUN, X., GOMES, A., WALLNER, J., AND POTTMANN, H. 2014. Form-finding with polyhedral meshes made simple. *ACM Trans. Graph. (SIGGRAPH) 33*, 4. Article 70.

UMETANI, N., IGARASHI, T., AND MITRA, N. J. 2012. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph. (SIGGRAPH) 31*, 4. Article 86.

VANEK, J., GALICIA, J. A. G., BENES, B., MĚCH, R., CARR, N., STAVA, O., AND MILLER, G. S. 2014. PackMerger: A 3D print volume optimizer. *Computer Graphics Forum 33*, 6, 322–332.

VOUGA, E., HÖBINGER, M., WALLNER, J., AND POTTMANN, H. 2012. Design of self-supporting surfaces. *ACM Trans. Graph. (SIGGRAPH) 31*, 4, Article 87.

XIN, S.-Q., LAI, C.-F., FU, C.-W., WONG, T.-T., HE, Y., AND COHEN-OR, D. 2011. Making burr puzzles from 3D models. *ACM Trans. Graph. (SIGGRAPH) 30*, 4. Article 97.

ZHOU, Y., SUEDA, S., MATUSIK, W., AND SHAMIR, A. 2014. Boxelization: Folding 3D objects into boxes. *ACM Trans. Graph. (SIGGRAPH) 33*, 4. Article 71.

Appendix

Proof: The Formal Model

In this subsection, we show that assemblies constructed based on the formal model presented in Sections 3 and 4 are i) *global interlocking*, i.e., all parts and all subsets of parts are immobilized except the key, and ii) can be assembled (*assemblability*).

Proof: Global Interlocking. We prove this condition by mathematical induction with the following proposition $\mathrm{Q}(n)$:

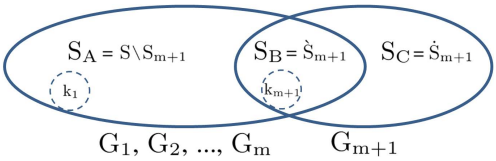
> *Given an assembly of n LIGs constructed according to the conditions in the formal model (particularly in Section 4.3), all subsets of parts in the assembly are immobilized except* $\mathrm{k_1}$ *(equivalently,* $\mathrm{S}\backslash\{\mathrm{k_1}\}$*).*

First, when n=1, the assembly has only one LIG, so the LIG's local interlocking property fulfills the proposition. So, $\mathrm{Q}(1)$ is true.

Then, we assume that $\mathrm{Q}(m)$ is true for some positive integer m. Hence, assemblies with m LIGs constructed according to the formal model are global interlocking with all subsets of parts being immobilized except $\mathrm{k_1}$ (equivalently, $\mathrm{S}\backslash\{\mathrm{k_1}\}$ due to complement).

When n=m+1, we have an assembly of m+1 LIGs ($\mathrm{G_1}$, $\mathrm{G_2}$, ..., $\mathrm{G_m}$, and $\mathrm{G_{m+1}}$) constructed based on the formal model. Next, we need to show that all subsets in $\mathrm{S} = \mathrm{S_1}\cup...\cup\mathrm{S_{m+1}}$ are immobilized, except $\mathrm{k_1}$ (equivalently, $\mathrm{S}\backslash\{\mathrm{k_1}\}$). Since the last LIG $\mathrm{G_{m+1}}$ can only be constructed by either case ii or case iii in Section 4.3, we have to show that $\mathrm{Q}(m+1)$ is true for these two cases.

(i) $\mathrm{G_{m+1}}$ shares parts with one previously-constructed LIG only. Without loss of generality, we take $\mathrm{G_m}$ as such a group. See notations in Section 4: we denote $\mathrm{S_A} = \mathrm{S}\backslash\mathrm{S_{m+1}}$ as the set of parts in "$\mathrm{G_1}$, $\mathrm{G_2}$, ..., $\mathrm{G_m}$" not shared with $\mathrm{G_{m+1}}$; $\mathrm{S_B} = \grave{\mathrm{S}}_{\mathrm{m+1}}$ as the set of parts shared between $\mathrm{G_m}$ and $\mathrm{G_{m+1}}$ (including $\mathrm{k_{m+1}}$); and $\mathrm{S_C} = \dot{\mathrm{S}}_{\mathrm{m+1}}$ as the set of parts in $\mathrm{G_{m+1}}$ only.



To show that S is global interlocking, we first need to show that the primary key $\mathrm{k_1}$ remains mobile. This is achieved by the *assemblability* condition in case i of Section 4.3.

Next, to show that all subsets of parts in $\mathrm{S} = \mathrm{S_A}\cup\mathrm{S_B}\cup\mathrm{S_C}$ are immobilized, except $\mathrm{k_1}$ and $\mathrm{S}\backslash\{\mathrm{k_1}\}$, we divide the subsets into the following non-overlapping categories:

Category 1: Subsets of $\mathrm{S_A}\cup\mathrm{S_B}$, excluding $\mathrm{k_1}$. These subsets of parts are from the assembly of "$\mathrm{G_1}$, $\mathrm{G_2}$, ..., $\mathrm{G_m}$." Since such an assembly has m LIGs, only $\mathrm{k_1}$ and $\mathrm{S_A}\cup\mathrm{S_B}\backslash\{\mathrm{k_1}\}$ are mobile in this assembly by the assumption that $\mathrm{Q}(m)$ is true. Since $\mathrm{G_1}$, $\mathrm{G_2}$, ..., $\mathrm{G_m}$ is a sub-assembly of S, the immobilization of these subsets remains in S.

Furthermore, we can show that $\mathrm{S_A}\cup\mathrm{S_B}\backslash\{\mathrm{k_1}\}$ is immobilized in S because in the presence of $\mathrm{S_C}$ in S, $\mathrm{S_B}$, which consists of $\mathrm{k_{m+1}}$ and at least one non-key part of $\mathrm{G_{m+1}}$, cannot be separated from $\mathrm{S_C}$ since $\mathrm{G_{m+1}}$ is a LIG. Hence, all subsets of $\mathrm{S_A}\cup\mathrm{S_B}$ are immobilized in S, except $\mathrm{k_1}$.

Category 2: Subsets of $\mathrm{S_C}$. Since $\mathrm{G_{m+1}}$ is a LIG, all subsets of $\mathrm{S_{m+1}}$ are immobilized, except $\mathrm{k_{m+1}}$ and $\mathrm{S_{m+1}}\backslash\{\mathrm{k_{m+1}}\}$. Since subsets of $\mathrm{S_C}$ are in $\mathrm{S_{m+1}}$ without $\mathrm{k_{m+1}}$ and $\mathrm{S_{m+1}}\backslash\{\mathrm{k_{m+1}}\}$, all subsets of $\mathrm{S_C}$ are immobilized.

The subsets remaining after categories 1 and 2 are those with a nonempty portion of $\mathrm{S_A}\cup\mathrm{S_B}$ and a nonempty portion of $\mathrm{S_C}$. We further divide these subsets into categories 3 to 5.

Category 3: Subsets without $\mathrm{S_B}$, with a nonempty portion of $\mathrm{S_A}$, and with a nonempty portion of $\mathrm{S_C}$. Without $\mathrm{S_B}$, the nonempty portions of $\mathrm{S_A}$ and $\mathrm{S_C}$ are not adjacent to each other in S, so subsets in category 3 can only be mobile if both portions are mobile simultaneously. However, since the nonempty portion of $\mathrm{S_C}$ are simply subsets in category 2, such a portion is always immobilized (by the same argument in category 2), and so are the subsets in category 3.

Category 4: Subsets with $\mathrm{S_B}$, a nonempty portion of $\mathrm{S_A}$, and a nonempty non-full portion of $\mathrm{S_C}$. $\mathrm{S_B}\cup$ a nonempty non-full portion of $\mathrm{S_C}$ is simply a subset of $\mathrm{S_{m+1}}$ ($\neq\{\mathrm{k_{m+1}}\}$). Since $\mathrm{S_{m+1}}$ is a LIG, these subsets are all immobilized to be separated from the remaining parts of $\mathrm{S_{m+1}}$, so they together with a nonempty portion of $\mathrm{S_A}$ are still immobilized in S.

Category 5: Subsets with $\mathrm{S_B}\cup\mathrm{S_C}$ ($\mathrm{S_{m+1}}$) and a nonempty portion of $\mathrm{S_A}$, excluding $\mathrm{S}\backslash\{\mathrm{k_1}\}$. Let $\hat{\mathrm{S}}_\mathrm{A}$ be a nonempty portion of $\mathrm{S_A}$: $\hat{\mathrm{S}}_\mathrm{A}$ cannot be $\mathrm{S_A}$, otherwise the resulting subset is $\mathrm{S_A}\cup\mathrm{S_{m+1}}$, i.e., whole assembly. Moreover, $\hat{\mathrm{S}}_\mathrm{A}$ cannot be $\mathrm{S_A}\backslash\{\mathrm{k_1}\}$, otherwise the resulting subset is $\mathrm{S}\backslash\{\mathrm{k_1}\}$. Since the complement of category-5 subsets in S is $\mathrm{S}\backslash(\hat{\mathrm{S}}_\mathrm{A}\cup\mathrm{S_{m+1}})$ $= \mathrm{S_A}\backslash\hat{\mathrm{S}}_\mathrm{A}$, which must be nonempty and not equal to $\{\mathrm{k_1}\}$. Such a complement is a subset belonging to those immobilized subsets in category 1. Due to the fact that if a subset of parts is immobilized, its complement in S is also immobilized. Hence, category-5 subsets are all immobilized in S.

Summary: Since the above five categories cover all subsets of $\mathrm{S}=\mathrm{S_A}\cup\mathrm{S_B}\cup\mathrm{S_C}$ (excluding $\mathrm{k_1}$ and $\mathrm{S}\backslash\{\mathrm{k_1}\}$), we conclude that all subsets of parts (except $\mathrm{k_1}$ and $\mathrm{S}\backslash\{\mathrm{k_1}\}$) in the assembly of the m+1 LIGs (S) are immobilized. So, $\mathrm{Q}(m+1)$ is true.

(ii) $\mathrm{G_{m+1}}$ shares parts with more than one previously-constructed LIGs. This case is similar to case (i), except that $\mathrm{S_B}$ contains parts of $\mathrm{G_{m+1}}$ shared with multiple previous groups (including $\mathrm{k_{m+1}}$ and at least one non-key part of $\mathrm{G_{m+1}}$). If we go through the five categories above with such $\mathrm{S_B}$, we can still find that the same arguments apply, so subsets of categories 1 to 5 in this case are all immobilized, except $\mathrm{k_1}$ and $\mathrm{S}\backslash\{\mathrm{k_1}\}$.

For both cases above, $\mathrm{Q}(m+1)$ is true if $\mathrm{Q}(m)$ is true. Hence, by the principle of mathematical induction, $\mathrm{Q}(n)$ is true for any positive integer n.

Proof: Assemblability. We show that this condition can be achieved by the two *assemblability* conditions in Section 4.3.

By the *assemblability* condition in case i of Section 4.3, when we construct $\mathrm{G_1}$, we have ensured that $\mathrm{k_1}$ can be taken out from the full assembly, and after that, we can subsequently take out all parts in "$\mathrm{S_1}\backslash\{\mathrm{k_1}\}$" that are not in $\mathrm{R_2}$, so the remaining parts are $\mathrm{R_2}$.

By the *assemblability* condition in case ii of Section 4.3 (note: this condition applies to both cases ii and iii of Section 4.3), when we construct $\mathrm{G_i}$ ($2\leq i\leq n$), we have ensured that given $\mathrm{R_i}$ as the remaining parts, we can take out $\mathrm{k_i}$ and then all parts in "$\mathrm{S_i}\backslash\{\mathrm{k_i}\}$" that are not in $\mathrm{R_{i+1}}$, so the remaining parts are $\mathrm{R_{i+1}}$. Hence, we can remove parts in $\mathrm{G_2}$ until $\mathrm{R_3}$ remains, remove parts in $\mathrm{G_3}$ until $\mathrm{R_4}$ remains, etc., until we reach and disassemble $\mathrm{G_n}$.

Therefore, we can always disassemble the entire structure part by part. In other words, we can also assemble the same structure part by part from $\mathrm{G_n}$ using a reversed parts order. As a result, we can show that our formal model guarantees assemblability.