INTERLOCKER

DESIGN DOCUMENT

By: Jiongjian Chen & Mohamed Soudy

Based on:

"Computational Interlocking Furniture Assembly"
Chi-Wing Fu, Peng Song
Xiaoqi Yan, Lee Wei Yang, Pradeep Kumar Jayaraman, Daniel Cohen-Or
SigGraph 2015

PROJECT SUMMARY

Our authoring tool will allow users to create an interlocking furniture assemblies' procedure with given furniture parts.

In order to start, we'll need to 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. After obtaining a set of simple 3D parts, 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. Later, 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. Finally, we generate the 3D model of each part by adapting the chosen joint geometry to modify the part.

The user will be able to select different furniture (bookshelf, baby bed, shoe rack, bench, bed stand, console table, sofa, chair, child bed and multi-function table, etc.) to start the assembly procedure. An animation guidance will then be displayed where the user may choose to assemble the whole furniture by themselves.

Our tool can fulfill the needs of artists, manufacturers and users. It can be used by artists to create novel interlocking furniture based on various building blocks, without worrying about how to assemble or dissemble them. For manufacturers, it could possibly set up an industrial standard for 3D fabrication as well as fabrication by parts. Being able to digitally construct 3D interlocking structures will allow for better accuracy in the production process. For consumers, using interlocking furniture will ease the assembly/disassembly process and also the animation procedure our tool produces will serve as a guidance on how to correctly assemble the furniture.

Our alpha version is set to be completed by 3/20/2017 and our beta version will be ready by 4/24/2017. Our final version will be done by 5/8/2017.

1. AUTHORING TOOL DESIGN

1.1. Significance of Problem or Production/Development Need

Different parts of traditional furniture are usually connected with glue, nails, hinges, screws, etc. However, connecting parts of furniture this way makes it difficult to take apart the pieces again.
This conventional approach also harms the external appearance of the furniture and, in general, the aesthetics of the design. Interlocking furniture encourages assembly and disassembly and allows the components to be tightly interlocked with one another. The furniture can be easily assembled and disassembled repeatedly without excessive wear on its parts.

1.2. Technology

[FSYYJC15] Computational Interlocking Furniture Assembly

This is the paper we will mainly follow as it is the one we plan to implement for our authoring tool. This paper presents a computational solution to support the making of interlocking furniture assemblies. Taking a design composed of just simple 3D shapes, this 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.

The following four papers are the secondary papers we'll be using. What makes them unique is that they follow one another in terms of the main steps we'll need to take for our tool, which are, 3D Fabrication, Fabrication by Parts, 3D Interlocking Assembly and Fabricating Self-Supporting Structures.

[LOMI11] Converting 3D furniture models to fabricatable parts and connectors.

This paper solves the problem of taking an inputted 3D model of a man-made object, and automatically generating the parts and connectors required to build the corresponding physical object. The paper focuses on furniture models and the defining of formal grammars for IKEA cabinets and tables. This will prove to be useful with **3D fabrication** and when trying to take apart a 3d furniture into parts which is going to be the input of our tool.

[ZSMS14] Boxelization: Folding 3D objects into boxes.

In this paper the authors introduce a technique for transforming a 3D object into a cube or a box using a continuous folding sequence. The method suggested produces a single, connected object that can be physically fabricated and folded from one shape to the other. The method introduced in this paper will serve as guidance for **fabrication by parts**.

[LACS08] Automated generation of interactive 3D exploded view diagrams.

This paper introduces an automatic technique for organizing 3D models into layers of an explosion graph that shows how parts explode with respect to each other. The explosion graph also handles the most common classes of interlocking parts. It also introduces two algorithms for exposing user-selected target parts, one that explodes the necessary portions of the model in order to make the targets visible. This paper will help with **3D interlocking assembly**, to test if our outputs are interlocked and can be assembled.

[DPWLBSP14] Assembling self-supporting structures

This paper builds from the research in the creation and exploration of self-supporting freeform designs where the authors recognize the difficulty in constructing such structures which need extensive formwork during assembly that leads to high construction costs on a building scale. The authors propose a gradual construction method of the masonry model in stable sections which drastically reduce the material requirements and construction costs. The method is then checked on 3D printed models and also designs of self-supporting puzzles. Research provided in this paper will help in understanding the **fabrication of self-supporting structures** which is a necessity for furniture assembly.

1.3. Design Goals

1.3.1 Target Audience.

The target audience for our tool are:

- furniture design artist
- Manufacturers
- Consumers

1.3.2 User goals and objectives

User can create interlocking furniture assemblies' procedure with given furniture parts.

1.3.3 Tool features and functionality

<u>Furniture selection/upload</u>
User will be able to select from a list of different furniture for the assembly process, while they're also encouraged to input their own furniture model to generate the assembly procedure.

<u>Furniture Assembly</u>

User will preview the animation of assembly/disassembly process, with close attention to the order of assembling/disassembling each part. User will also be able to assemble the parts manually within Maya viewport.

1.3.4 Tool input and output

Input: a set of simple 3D parts, where adjacent parts may contact or intersect
Output: computational solution (step animation/user manual) to support the making of interlocking furniture assemblies

1.4. User Interface

1.4.1 GUI Components and Layout

Overall our plugin doesn’t require much user input to operate. The user will be able to select different furniture they would like to generate an interlocking output for (See figure 1). Additionally, the user can adjust some of these building parameters for the scene.

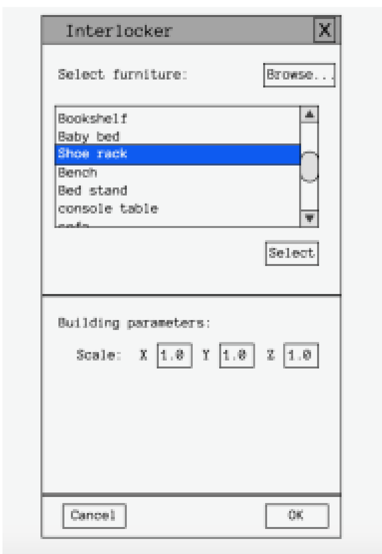


Figure 1: The options dialog

1.4.2 User Tasks

The user should have some familiarity with the Maya interface and how to change attributes. The user must select which furniture he/she wants to work on, and also be able to upload their own compatible furniture model.

1.4.3 Work Flow

1. User will interact with GUI to select the furniture type, and assign corresponding parameters if needed.
2. Maya viewpoint displays the animation of assembling the interlocking furniture with correct order; it starts off with a bunch of 3D furniture parts.
3. Output is an interlocking furniture that's assembled by all these parts.

The tool provides a visual procedure of how interlocking furniture should be assembled in suitable order, it also supports the making of interlocking furniture assemblies. Taking a design composed of just simple 3D shapes, it can automatically plan a joints network over the furniture, so that its component parts tightly interlock with one another in a global interlocking assembly.

2. AUTHORING TOOL DEVELOPMENT

2.1. Technical Approach

2.1.1 Algorithm Details

The computational solution has three major steps:

- Parts-Graph construction
- LIGs construction
- 3D model generation

A parts-graph is an undirected graph, where nodes represent furniture parts and edges connect two contacting/intersecting parts (see Figure 2). For the first step, 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. 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.

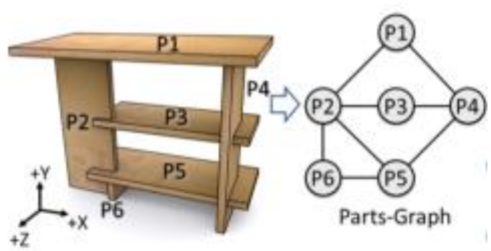


Figure 2: *Parts-graph*

After obtaining parts-graph, the next step is the construction of LIGs (local interlocking group), which is a connected subgraph in parts-graph, where parts are locked by a specific key in the group. 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.

For the last step, we generate the 3D model of each part by adapting the chosen joint geometry to modify the part. The process is illustrated in figure 3.

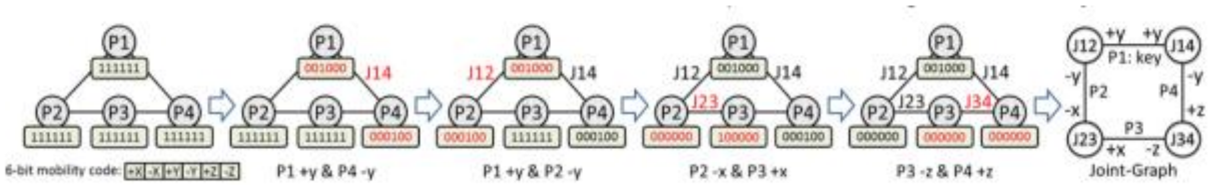


Figure 3: *We iteratively assign joints between adjacent parts,*

Time permitting; we may also create additional furniture apart from what's provided previously. For this project, all building blocks (3D furniture parts will be modeled and provided beforehand, instead of generating by algorithm)

The full pipeline is outlined in Figure 4.

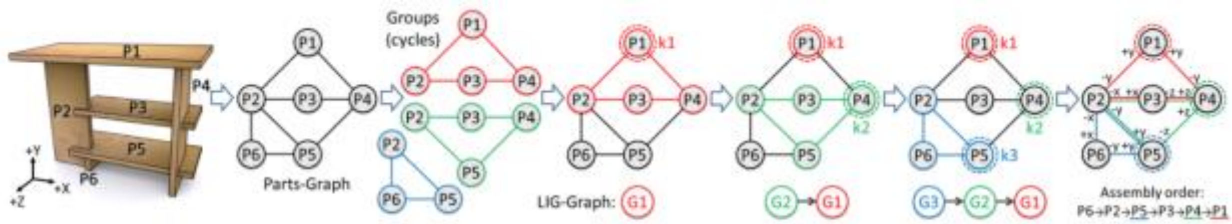


Figure 4: *Overview of our approach.*

2.1.2 Maya Interface and Integration

For this project, all GUI components such as dialogs and user settings will be implemented in MEL. The interlocking furniture assembly will be implemented as a C++-based command plugin accessible from MEL.

It will be using these Maya objects and classes:

MObjectArray - General data structure for storing various objects or textures
MAnimMessage - Animation messages
MAttributeIndex - The index information for an attribute specification
MAttributeSpec - An attribute specification
MFnData - Parent class for dependency graph data function sets
MItDependencyGraph - Dependency Graph Iterator
MNodeClass - A class for performing node class-level operations in the dependency graph

2.1.3 Software Design and Development

It will be implemented as a C++ plugin. The core data structures for calculating the tree model are given below:
Interlocker: Main class, returns the final result of parts-graphs with LIGs and joint information
SourceFurnitureSet: maintains a list of 3D interlocking objects that a specific furniture needs.
SourceFurniture: maintains a list of interlocking furniture available for assembly
PartsGraphCreate: Implement parts-graph given a furniture model with interlocking object,
PartsGraphCreate: Identify overlapping cycles in its parts-graph.
LIGsConstruction: construct local interlocking groups iteratively, with order dependency over the parts-graph by sharing certain parts (in particular a key) between adjacent LIGs.
JoinGraphGeneration: Implement joints assignment between adjacent parts, such that each joint immobilizes all directions of associated parts, except for one certain direction.
Assembly: Determine the assembly order based on LIGsConstruction. Assuming each LIG is interlocked, we can achieve global interlocking over the entire assembly.

2.2. Target Platforms

2.2.1 Hardware

• (32-bit) Intel® Pentium® IV or higher, AMD Athlon® XP processor (must be SSE capable)
• 1 GB RAM
• Hardware-accelerated OpenGL® graphics card

2.2.2 Software

• (32-bit) Windows XP Professional (SP2)
• Maya 8.5
• OpenGL 1.2+
• Microsoft Visual C++ 2005 or higher

2.3. Software Versions

2.3.1 Alpha Version Features (first prototype)

Interlocker will contain the following features in the alpha version:

- GUI
- Various 3D furniture
- various 3D interlocking parts made in 3D Max/Maya
- Interlocking furniture composition and corresponding parts-graph

The demo application will show the GUI and show examples of the 3D furniture inputted into the tool such as tables and chairs. It'll also display the corresponding parts of the selected furniture in Maya accompanied by its Parts-Graph.

2.3.2 Beta Version Features

Interlocker will contain the following additional features in the beta version:

- Generation of joint graph
- LIG graphs
- Assembly order of furniture

In the demo application, we'll have a working app that takes in furniture and displays the interlocked version of the furniture in Maya. Joint graphs and LIG graphs will also be displayed.

2.3.3 Description of any demos or tutorials

Alpha Demo: This demo mainly makes user become familiar with 3D furniture building blocks, with the connecting joints shown on each part, user will be able to understand how interlocking furniture works and order does matters in assembly procedure. At this stage, user is encouraged to use existing blocks to randomly assemble whatever they want just, but user may experience certain difficulty without certain goal of final prototype and corresponding user manual for assembly.

Beta Demo: In this demo the user will be able to select different furniture and get the suggesting order of assembly with the help from Interlocker. This will make it much easier to assemble the interlocking furniture.

Final Demo: User can preview how different interlocking furniture is assembled, they're encouraged to try it they own as did in beta demo, also they may choose to upload their own furniture design, which consists of

3D parts, and Interlocker will generate a method to assemble. By this approach, users can focus on the design and appearance of the furniture rather than on the joints scheme and their geometry.

Tutorial: The final project will include code, sample source data, and how-to document. Users will be able to run the final demo out of the box.

Quick Guide: A Quick Guide will be provided to show the process of using the tool from when the user first interacts with the tool to the final outputted interlocked furniture.

3. WORK PLAN

3.1. Tasks

1. Build Framework – 1 week
   1.1. Model interlocking furniture parts in 3d Max or Maya – Mohamed
   1.2. Implement user GUI interface dialog (MEL) - Jiongjian
   1.3. Implement command plugin framework (C++) – Mohamed & Jiongjian
      1.3.1. Write code stubs
2. Construct Parts-graph – 2 week
   2.1. Generate initial parts-graph with given furniture model – Mohamed
   2.2. Merge degree-1 nodes in the graph with their adjacent parts – Mohamed
   2.3. Analyze and identify groups of overlapping cycles in the parts-graph - Jiongjian
3. Construct Joint Graph – 3 week
   3.1. Implement a precomputed joint lookup table, which is indexed by the ICO vector – Mohamed & Jiongjian
   3.2. Analyze each of its connections between adjacent parts by using the two cases – Mohamed & Jiongjian
      3.2.1. Implement two parts intersect case
      3.2.2. Implement two parts contact case
   3.3. Obtain Joint graph for further use – Mohamed
4. Construct LIG-graph – 3 weeks
   4.1. Implement conditions for a cycle of parts to be local interlocking (considering 3- and 4-part cycles) – Mohamed
   4.2. Implement rules for expanding an existing LIG – Jiongjian
   4.3. Implement three conditions for constructing LIGs with dependency in assembly / disassembly order. – Mohamed & Jiongjian
      4.3.1. Construct base case G1, which is the first one
      4.3.2. Construct Gj that shares parts with only one previously-constructed LIG, say Gi (j>i), but not others
      4.3.3. Construct Gj that shares parts with multiple previously-constructed LIGs.
   4.4. Obtain Assembly/Disassembly order
5. Integration & Polish – 1 week
   5.1. Test and refine workflow – Mohamed & Jiongjian
   5.2. Bug Fixing, Enhancements – Mohamed & Jiongjian
6. Overhead
   6.1. Authoring Tool Design Review Alpha (3/20/2017)
   6.2. Authoring Tool Design Review Beta 1 (4/5/2017)
   6.3. Authoring Tool Design Review Beta 2 (4/24/2017)
   6.4. Authoring Tool Final Version (5/8/2017)
   6.5. Write Authoring Tool Tutorial and Quick Guide

3.1.1 Alpha Version

Complete its corresponding tasks 1,2,3

3.1.2 Beta Version

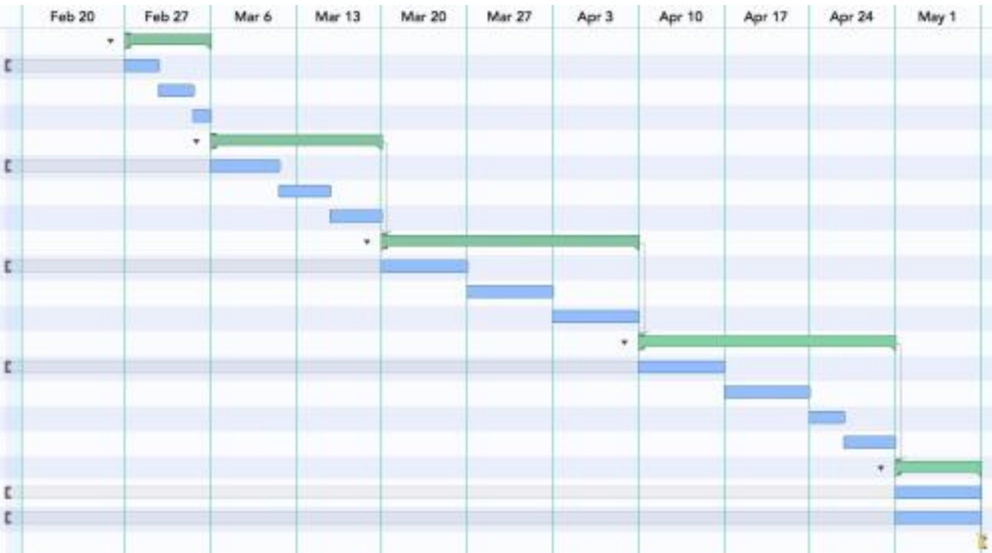
Complete its corresponding tasks 1,2,3,4,5

3.1.3 Final Version

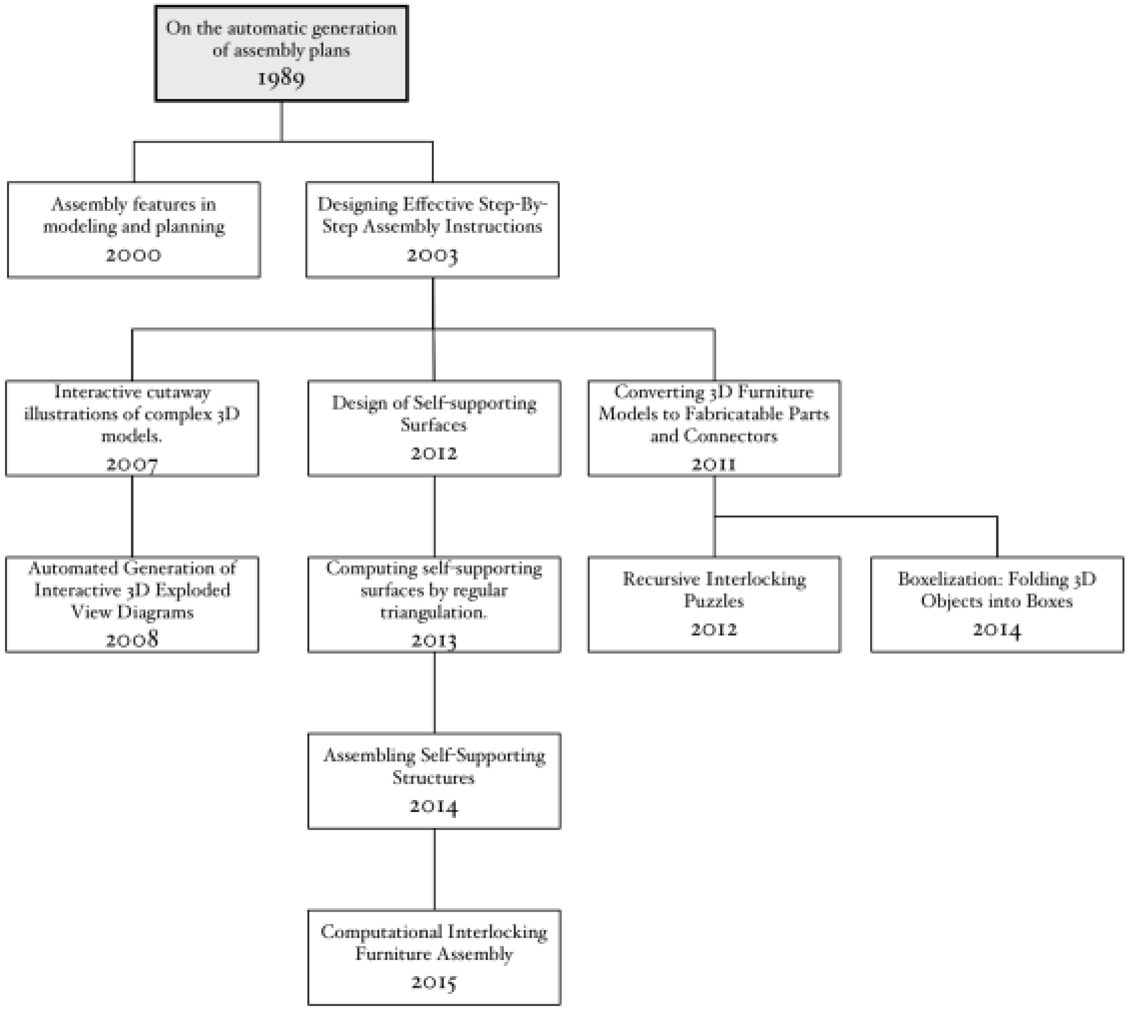
Complete its corresponding tasks 1,2,3,4,5,6

3.2. Schedule

Title	Start	End	Duration
1) Build Framework	**2/27/17**	3/3/17	5d
1.1) Model interlocking furniture parts in 3d Max or Maya	2/27/17	2/28/17	2d
1.2) Implement GUI interface dialog (MEL)	3/1/17	3/2/17	2d
1.3) Implement command plugin framework (C++)	3/3/17	3/3/17	1d
2) Construct Parts-Graph	**3/6/17**	3/17/17	10d
2.1) Generate initial parts-graph with given furniture model	3/6/17	3/9/17	4d
2.2) Merge degree-1 nodes in the graph with their adjacent parts	3/10/17	3/14/17	3d
2.3) Analyze and identify groups of overlapping cycles in the parts-graph	3/15/17	3/17/17	3d
3) Construct Joint Graph	**3/20/17**	4/7/17	15d
3.1) Implement a precomputed joint lookup table, which is indexed by the ICO vector	3/20/17	3/24/17	5d
3.2) Analyze each of its connections between adjacent parts by using the two cases	3/27/17	3/31/17	5d
3.3) Obtain Joint graph for further use	4/3/17	4/7/17	5d
4) Construct LIG-Graph	**4/10/17**	4/28/17	15d
4.1) Implement conditions for a cycle of parts to be local interlocking	4/10/17	4/14/17	5d
4.2) Implement rules for expanding an existing LIG	4/17/17	4/21/17	5d
4.3) Implement three conditions for constructing LIGs with dependency	4/24/17	4/25/17	2d
4.4) Obtain Assembly/Disassembly order	4/26/17	4/28/17	3d
5) Integration & Polish	**5/1/17**	5/5/17	5d
5.1) Test and refine workflow	5/1/17	5/5/17	5d
5.2) Bug Fixing, Enhancements	5/1/17	5/5/17	5d
6) Overhead	5/8/17	5/8/17	0d



4. RELATED RESEARCH



5. References

[ZSMS14] ZHOU, Y., SUEDA, S., MATUSIK, W., AND SHAMIR, A. 2014. Boxelization: Folding 3D objects into boxes. ACM Trans. Graph. (SIGGRAPH) 33, 4. Article 71.

[VHWP12] VOUGA, E., HOBINGER, M., WALLNER, J., AND POTTMANN, H. 2012. Design of self-supporting surfaces. ACM Trans. Graph. (SIGGRAPH) 31, 4, Article 87.

[SFC12] SONG, P., FU, C.-W., AND COHEN-OR, D. 2012. Recursive interlocking puzzles. ACM Trans. Graph. (SIGGRAPH Asia) 31, 6. Article 128.

[LHSWG13] 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.

[LACS08] 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.

[LOMI11] 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.

[DPWLBSP14] DEUSS, M., PANOZZO, D., WHITING, E., LIU, Y., BLOCK, P., SORKINE-HORNUNG, O., AND PAULY, M. 2014. Assembling self-supporting structures. ACM Trans. Graph. (SIGGRAPH Asia) 33, 6. Article 214.

[W89] WOLTER J. D. On the automatic generation of assembly plans. IEEE International Conference on Robotics and automation, Scottsdale, 1989, pp. 62–62.

[APHHKHT03] AGRAWALA, M., PHAN, D., HEISER, J., HAYMAKER, J., KLINGNER, J., HANRAHAN, P., AND TVERSKY, B. 2003. Designing effective step-by-step assembly instructions. ACM Transactions on Graphics 22, 3, 828–837.

[FSYYJC15] Fu, C.W., Song, P., Yan, X., Yang, L.W., Jayaraman, P.K., and Cohen-Or, D. 2015. Computational interlocking furniture assembly. ACM Trans. Graph. 34, 4, Article 91 (July 2015), 11 pages.

[LRACS07] LI, W., RITTER, L., AGRAWALA, M., CURLESS, B., AND SALESIN, D. 2007. Interactive cutaway illustrations of complex 3D models. ACM Transactions on Graphics 26, 3 (July), 31:1–31:11.

[HB00] VAN HOLLAND W, BRONSVOORT WF. Assembly features in modeling and planning. Robotics and Computer Integrated Manufacturing 2000; 16:277–94.