EUROGRAPHICS 2013 / I. Navazo, P. Poulin
(Guest Editors)

Volume 32 (2013), Number 2

Fabrication-aware Design with Intersecting Planar Pieces

Yuliy Schwartzburg and Mark Pauly

École Polytechnique Fédérale de Lausanne, Switzerland

Figure 1: *3D designs composed of planar intersecting pieces fabricated using laser cutting and CNC milling.*

Abstract

We propose a computational design approach to generate 3D models composed of interlocking planar pieces. We show how intricate 3D forms can be created by sliding the pieces into each other along straight slits, leading to a simple construction that does not require glue, screws, or other means of support. To facilitate the design process, we present an abstraction model that formalizes the main geometric constraints imposed by fabrication and assembly, and incorporates conditions on the rigidity of the resulting structure. We show that the tight coupling of constraints makes manual design highly nontrivial and introduce an optimization method to automate constraint satisfaction based on an analysis of the constraint relation graph. This algorithm ensures that the planar parts can be fabricated and assembled. We demonstrate the versatility of our approach by creating 3D toy models, an architectural design study, and several examples of functional furniture.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling

1. Introduction

3D constructions composed of interlocking planar pieces (Figure 1) are popular for creating 3D toys made from wood or cardboard, but are also of great interest in architecture and interior design (Figure 2). The popularity of these models is mainly due to the ease of fabrication and assembly. Planar pieces can be cut easily and cheaply from many different materials, including cardboard, wood, metal, plastic, glass, or stone, using simple machinery such as saws or laser cutters. Compared to 3D printing, 2D cutting not only allows for a wide variety of materials, but also a much larger range of scale, enabling constructions of the size of buildings at comparatively low cost.

In this paper we define geometric constraints for 3D objects composed of planar interlocking pieces and propose an optimization approach that solves for a physically realizable solution for a given user input of planar piece orientations, positions, and outlines. A feasible solution satisfies geometric conditions that directly relate to fabrication, stability, and assembly.

If two pieces intersect this introduces an angle constraint

on the plane normals. We derive sufficient conditions on the relation of these constraints that guarantee rigidity of the final assembled structure. To guarantee a valid assembly sequence, each pair of intersecting planar pieces is constrained to be displaced only in the direction of the slit created along their intersection line. These slit constraints can easily create a locked state in which a subset of the pieces cannot be moved at all. Our optimization automatically avoids such configurations and ensures that the 3D design can be assembled. Beyond the orientation and placement of planar pieces, we also optimize their geometric shape to avoid collisions during assembly.

The space defined by angle and slit constraints can quickly become difficult to navigate in, since cycles in the graph can introduce complex non-local dependencies. Modifications of the orientation of a single piece to satisfy a certain design intent can propagate through the entire object, making manual control of all constraints cumbersome and virtually impossible for more advanced designs. This complexity is reflected in the fact that existing manual designs are often limited to non-cyclic graphs or grid-like structures (see Figure 2). We show that these restrictions are not necessary and introduce a more flexible constraint space that enables a variety of designs that cannot be achieved by current methods.

Contributions. The core contributions of this paper are:

- The definition of a constraint space of 3D objects composed of planar intersecting pieces that is derived from requirements for fabrication, stability, and assembly,
- an optimization algorithm for finding feasible configurations that satisfy all constraints, and
- an interactive system that enables effective design exploration within the space of feasible solutions.

The input to our system is a set of initial plane orientations and positions given by the user or created by some generative procedure. These can be specified at the start or during the design process in any order. Planar piece outlines are either defined explicitly or found by intersecting an infinite plane with a 3D hull. During interactive editing, the designer can freely modify the position or orientation of planar pieces, add or remove pieces, or change the constraint graph. The solution is automatically updated to ensure that all constraints are satisfied.

The output is a set of 2D stencil curves and a corresponding assembly plan. The curves can be transformed to machining instructions and sent to standard planar cutting machines. The fabricated planar pieces can be flat-packed efficiently and assembled without glue or screws. We believe that the simplicity, flexibility, and low cost of this type of construction makes our system interesting both for casual users who want to create their own 3D toys or freeform furniture, as well as architects and designers who want to realize ambitious large-scale productions.

Figure 2: *Intersecting planar pieces can lead to compelling 3D forms with applications in interior design, toy making, sculptural art, and architecture.*

2. Related Work

We discuss related work grouped into three categories: shape abstraction, paper craft, and fabrication-aware design. Our literature review does not aim for completeness, but rather selects typical papers in the different categories to position our work in the context of previous research.

Shape abstraction. Approximating a 3D shape by planar pieces has been studied in the context of geometry processing and rendering. Variational shape approximation [CSAD04] uses Lloyd clustering to distribute a collection of plane proxies over a surface to best approximate a given 3D shape. Billboard clouds [DDSD03] provide highly efficient rendering by optimizing for a small set of textured planes that provide a faithful visual representation of the 3D object. Slices [MSM11] use planar contours for creating shape abstractions guided by perceptual cues derived from a user study. While using the same geometric primitives as we do to describe a shape, these methods are not intended as design tools for physical models, but rather focus on geometric or visual approximation of purely digital representations.

Paper craft. Various forms of paper craft modeling have been investigated in computer graphics research. Origami, the art of paper folding without introducing cuts or using glue, has been studied extensively in recent years [DO07, Tac10]. Hart [Har07] introduced modular kirigami, a technique to create intricate symmetric structures from identical paper pieces. Miller and Akleman [MA08] built upon this work and propose a recipe for creating slide-together paper sculptures based on a polyhedral base surface. Similar to our approach, these methods connect pieces through slits. However, the constructions focus on specific classes of shapes, such as Platonic or Archimedean solids, and operate on a surface representation, not a 3D volume.

Pop-up designs [Gla02, LSH*10, LJGH11] aim at converting a 3D scene into a stable representation supported by two base planes that can be folded flat without stretching or bending the paper. Our approach is related to these paper craft methods in the sense that we also consider a complex constraint space defined by a set of geometric constraints that result from material properties or construction limitations. However, the specifics of our constraint space are fundamentally different and therefore require a different computational strategy.

Fabrication-aware design. Digital design tools that incorporate fabrication constraints are becoming popular in mechanical engineering, product design, and architecture [Kil06]. Recently, commercial tools have become available, such as Autodesk 123D, that aim at making fabrication-aware design accessible to non-professional users. Spatial Sketch [WLMI10] is a system that takes sketching input from an infrared light pen, transforms the line drawings into a solid shape, and approximates that shape with a collection of radial or parallel planes. SketchChair [SLMI11] combines intuitive user interfaces, simulation, and automated fabrication to enable personalized chair design. The system generates a set of planar pieces that can be slit into each other to create the framing for the chair, using a predefined grid pattern to specify the combinatorial structure of the chair. Lau and coworkers [LOMI11] propose an algorithm for generating physical realizations of man-made objects. They introduce a formal grammar for furniture models and combine lexical and structural analysis to automatically create parts and connectors for fabrication.

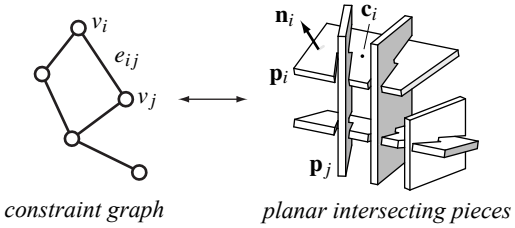
Schwartzburg and Pauly [SP11] provide a sketch of a design process for orthogonally intersecting planar pieces. While promoting the use of optimization for design support, they do not consider assembly constraints, rigidity, or collisions and provide no details on the optimization.

Recently, Hildebrand and coworkers [HBA12] introduced an algorithm for creating cardboard models based on sliding planar pieces. Their approach focuses on an automatic process that successively adds one element at a time while observing assembly constraints implicitly. In contrast, our system is designed as a dynamic design tool that allows the user to adjust and modify all pieces during the design. Instead of a static data structure that is built incrementally, we propose a continuous optimization that minimally alters the entire configuration to satisfy the constraints. By not limiting the data structure to sequential trees, our work allows for structures with cycles with non-trivial assemblies and assembly orders (such as cases when multiple pieces have to be moved at the same time during assembly, see also the supplementary materials). Additionally, [HBA12] does not consider the angles between planar pieces, keeping them fixed to 90 degrees.

A central component of our optimization is concerned with ensuring a valid assembly sequence of the intersecting planar pieces. Constraints on assembly are also the focus of methods for generating geometric puzzles, such as Polyomino puzzles [LFL09] or Burr puzzles [XLF*11]. While we do not explicitly consider puzzles in this work, we comment on the potential of our method for this type of application in Section 6 (see also accompanying video). Similarly, [Séq12] deals with multi-hand assemblies, which we do not consider in this paper due to the physical difficulty of construction, but this extension would be interesting to explore in future work.

3. Problem Formulation

We first introduce some notation to provide a formal description of our optimization and design objectives. In Section 4, we will discuss a method to realize these objectives. We represent an arrangement of intersecting planar pieces by a graph $G = (\mathcal{V}, \mathcal{E})$ consisting of a set $\mathcal{V} = \{v_1, \ldots, v_n\}$ of n nodes or vertices, and a set $\mathcal{E} = \{e_1, \ldots, e_m\}$ of m edges with $e_k \in \mathcal{V} \times \mathcal{V}$. Each node v_i represents a planar piece $\mathbf{p}_i$ with a normal $\mathbf{n}_i$, and the centroid of the piece $\mathbf{c}_i$ which lies on the plane (see illustration below).



constraint graph *planar intersecting pieces*

We write e_{ij} to denote an edge that connects the vertices v_i and v_j. An edge $e_{ij} \in \mathcal{E}$ defines a slit connection between two nodes v_i and v_j. We denote with $\alpha_{ij} = \arccos(|\mathbf{n}_i \cdot \mathbf{n}_j|)$ the corresponding intersection angle of the two plane normals $\mathbf{n}_i$ and $\mathbf{n}_j$, where $\alpha_{ij} \in [0, \pi/2]$ always measures the smaller of the two intersection angles.

3.1. Rigidity

To ensure stability of the structure, we must introduce constraints on the angles at which the pieces intersect. These angles restrict possible rotations (see Figure 3(e)) and contribute to the stress at the intersection slit. The angle at which a slit can be cut into a planar piece is constrained by machining limitations. Many tools (e.g. most laser cutters) can only cut a planar piece in a direction orthogonal to the plane normal, while other machinery (e.g. CNC milling devices or 3D laser cutters) have bounds on how far they can deviate from orthogonality (typically by about 40 degrees for a five-axis milling machine). We denote the maximum possible cutting angle relative to the plane normal as $t_\alpha \in [0, \pi/2)$. This cutting angle has direct consequences on the width h_{ij} of the slits: Let σ be the thickness of the material. Then for an edge e_{ij} with corresponding intersection angle α_{ij} the minimal re-

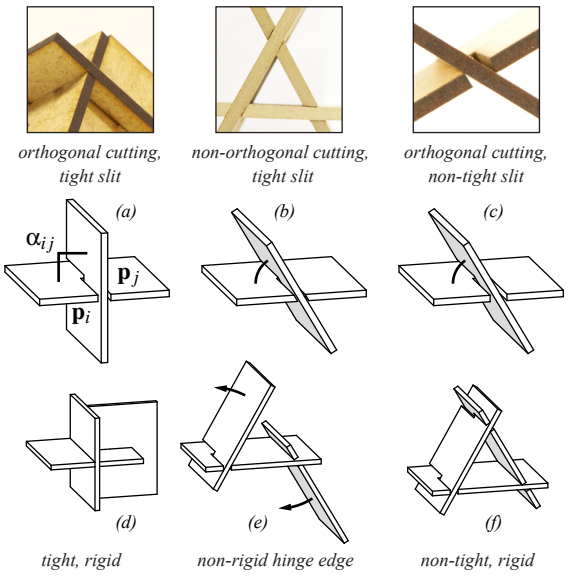
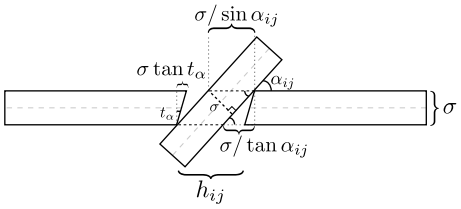


Figure 3: *Machining restrictions impose constraints on the maximal cutting angle of slits. Most devices (e.g. laser cutters) can only produce cuts orthogonal to the surface plane (a, c), while others (e.g. 5-axis milling) can accommodate slits up to some angular threshold (b). Tight slits (a, b, d) are generally preferable in terms of stability, but can significantly restrict the possible intersection angle of pieces. Wide slits (c) allow more freedom, but can introduce undesirable hinge edges that might lead to instabilities (e). Even when resorting to wide slits, our optimization is guaranteed to find a locally rigid configuration as illustrated in (f).*

quired slit width is

$$h_{ij} = \sigma/\sin\alpha_{ij} + \max(\sigma/\tan\alpha_{ij} - \sigma\tan t_\alpha, 0). \quad (1)$$



Angle constraints. We call a slit *tight*, if the pieces are touching along the entire intersection, or $\sigma\tan t_\alpha \geq \sigma/\tan\alpha_{ij}$. It follows that the slit of edge e_{ij} can only be tight if $\alpha_{ij} \geq \pi/2 - t_\alpha$. We call this constraint on the intersection of two planar pieces an *angle constraint*. In our optimization we aim for tight slits as these provide the most stable configurations. In particular for dense graphs, however, it is not always possible to obtain large enough intersection angles to achieve tight slits everywhere. Fortunately, we can achieve stable configurations even when widening the slit width.

We say a node $v_i \in \mathcal{V}$ is *rigid*, if the corresponding piece has no free rotational motion space, i.e. if the slit connections with other intersecting nodes prevent any rotational motion of the piece with respect to its adjacent pieces. We call a graph G *locally rigid*, if all nodes $v_i \in \mathcal{V}$ are rigid. As illustrated in Figure 3, tight slits trivially ensure rigidity of a piece. Alternatively, several slits can work together to eliminate any free rotational motion, thus ensuring rigidity even in the absence of tight slits. For a single piece, we can see that any two (non-collinear) slits already guarantee rigidity if the two connected pieces are themselves fixed, as the piece cannot rotate without pushing on the connected pieces (see Figure 4(a)).

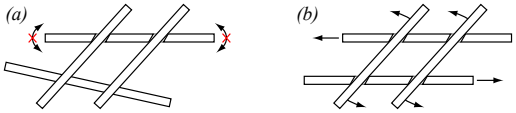


Figure 4: *A piece with non-tight slits (a) can still be rigid if its neighboring pieces are fixed. A configuration with four parallel non-tight slits in a cycle (b) is locally but not globally rigid. Therefore, our optimization always aims for tight slits if possible.*

Non-tight edges that are not part of a cycle of a graph will allow rotational motion around the slits. We call these *hinge edges* and write $\mathcal{E}^h \subseteq \mathcal{E}$ for the set of all hinge edges. Therefore, a necessary (but not sufficient) condition for local rigidity is that the slits of all hinge edges must be tight, i.e. $\alpha_{ij} \geq \pi/2 - t_\alpha \; \forall e_{ij} \in \mathcal{E}^h$.

Global rigidity, or preventing rotation in any cycle of G and not just per piece, is a difficult problem related to the rigidity of graphs [Jac07]. A non-globally rigid graph is shown in Figure 4(b). Rotation involving multiple pieces, as illustrated here, is unlikely to occur without specific initial conditions. Even so, to prevent this, we aim for tight slits wherever possible. Configurations with wide slits can also be less desirable for physical realization, since forces acting on these slits will be concentrated on the contact edges, which might lead to strong internal stresses and corresponding material wear [Dow93]. We call a graph *strongly rigid*, if all slits are tight, i.e. $\alpha_{ij} \geq \pi/2 - t_\alpha \; \forall e_{ij} \in \mathcal{E}$. Our optimization always tries to find a strongly rigid solution first. Only if such a solution cannot be obtained does the algorithm resort to wider slits, while still ensuring that the configuration remains locally rigid. If there is no feasible tight solution or if a tight solution is not desired, coupled rotations and material stress could be checked afterwards with a physics simulation in a similar manner to [UIM12].

3.2. Assembly

If we introduce cycles in the graph, similar to the cycle in Figure 5, then it can become impossible to separate the pieces and likewise impossible to assemble the object. Therefore, we consider the problem of disassembly: If a 3D

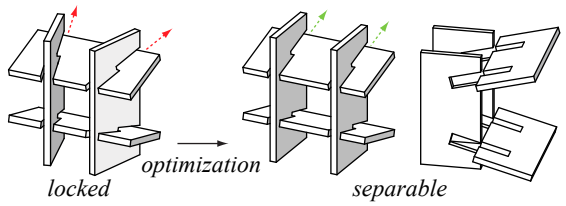
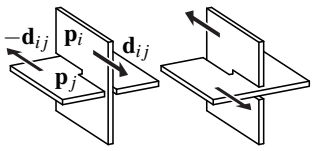


Figure 5: *Planar intersecting pieces can easily create locked configurations (left) in which no piece can be moved and hence the structure cannot be assembled. Our algorithm optimizes for plane orientations that ensure feasibility of assembly (right).*

object represented by a graph G can be separated into individual pieces, then the inversion of this disassembly sequence yields an assembly plan. In this section, we focus on how to define and detect if a structure is assemblable. We describe a solution to ensure this constraint for an invalid configuration in Section 4.1.

We need to satisfy two types of constraints to guarantee existence of a disassembly sequence: (i) Slit constraints and (ii) global collision constraints. Slit constraints restrict the relative motion for separating two interlocking pieces and are independent of the actual shape of the pieces. Effectively, these constraints are due to the collisions of the two pieces in every direction but that of the slit and the coupling of these collisions (see Figure 5). They therefore depend only on the intersection direction $\mathbf{d}_{ij}$ and the structure of the constraint graph. Global collisions, or ensuring collision-free paths during assembly, are discussed in Section 5.

Slit constraints. The physical connection defined by an edge $e_{ij} \in \mathcal{E}$ is realized by introducing straight slits into the planar pieces $\mathbf{p}_i$ and $\mathbf{p}_j$ (Figure 3). We assume that $\mathbf{p}_i$ and $\mathbf{p}_j$ can only be separated by translating them in opposite directions along the vector $\mathbf{n}_i \times \mathbf{n}_j$. We assume this is the only way to separate a connection, i.e. no rotation or deformation of a piece is allowed or necessary. Since both configurations are generally possible for a slit (see inset illustration), we define the edge direction vector as $\mathbf{d}_{ij} = \pm \mathbf{n}_i \times \mathbf{n}_j$, using the sign convention that piece $\mathbf{p}_i$ can only be moved in the direction $\mathbf{d}_{ij}$ relative to $\mathbf{p}_j$, and, analogously, $\mathbf{p}_j$ can only be moved in direction $-\mathbf{d}_{ij}$ relative to $\mathbf{p}_i$. We call these restrictions on the relative motion of pieces $\mathbf{p}_i$ and $\mathbf{p}_j$ the *slit constraint* of edge e_{ij}.



Assembly conditions. We call two edges e_{ij} and e_{kl} *parallel*, if $\mathbf{d}_{ij} = \mathbf{d}_{kl}$. We say a graph $G = (\mathcal{V}, \mathcal{E})$ can be *split* into two subgraphs G_1 and G_2, if there exists

Algorithm 1 isSeparable$(G = (\mathcal{V}, \mathcal{E}))$

```
if |V| = 1 then
  return true
else
  for all e ∈ E do
    // find all edges parallel to e
    E' ← {e' ∈ E | d(e') = d(e)}
    // if parallel cut exists, cut graph and recurse
    if E' contains a cut C ⊆ E' of G then
      split G along C into G1 and G2
      return isSeparable(G1) and isSeparable(G2)
    end if
  end for
  return false
end if
```

- a cut into two non-empty vertex sets $\mathcal{V}_1$ and $\mathcal{V}_2$ with $\mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{V}$ and $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$, and
- a cut-set of parallel edges $\mathcal{E}' \subseteq \mathcal{E}$ such that $v_i \in \mathcal{V}_1$ and $v_j \in \mathcal{V}_2$ for all $e_{ij} \in \mathcal{E}'$.

The two subgraphs are then given as $G_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2)$, where the edge sets $\mathcal{E}_1$ and $\mathcal{E}_2$ are the restrictions of $\mathcal{E}$ to $\mathcal{V}_1$ and $\mathcal{V}_2$, respectively. We call the edge set $\mathcal{E}'$ a *parallel cut* (as opposed to other cuts in the graph that do not consist of only parallel edges). The existence of a parallel cut means that we can displace all pieces in $\mathcal{V}_1$ along the common edge direction of $\mathcal{E}'$ to separate them from the pieces in $\mathcal{V}_2$ without violating any of the slit constraints (see for example the assembly of a cycle along a parallel cut in the accompanying video [3:45-]).

Definition I: We call a graph $G = (\mathcal{V}, \mathcal{E})$ *separable*, if

- $|\mathcal{V}| = 1$, or
- G can be split into two separable subgraphs G_1 and G_2.

It follows immediately from Definition I that any graph that does not contain cycles, i.e. is a tree, is separable, since any edge defines a parallel cut-set. Cycles in the graph, however, require the existence of a set of parallel edges that cuts the cycle into two or more parts.

From Definition I we can derive a recursive algorithm that checks if a given graph G is separable and hence the corresponding 3D structure disassemblable (see Algorithm 1). This algorithm makes use of an important observation: If a graph is separable and contains multiple parallel cuts, then applying any of these cuts in any order will create separable subgraphs since the edges of the other cuts will define parallel cuts in one or more of the new subgraphs as well. Consequently, we can simply iterate through all edges in the edge set $\mathcal{E}$, apply the first parallel cut that we find (if one exists) to split the graph into two, and execute the same procedure recursively on the generated subgraphs. Any cycle in

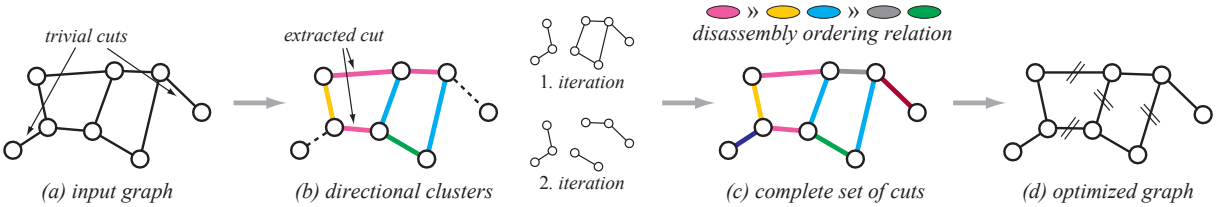


Figure 6: *Optimizing for assembly. Ignoring trivial cuts, edges are clustered according to the slit direction (illustrated here by the 2D graph edge direction) and a cut is extracted from the clusters (b). The corresponding edges are removed and clustering is repeated until a complete set of cuts (indicated by color) is obtained (c). The optimization then ensures that all edges of the same cut become parallel (d) so that the resulting object can be disassembled. The sequence of disassembly is constrained by the partial ordering relation resulting from the clustering method. The dark purple and red connections can be separated at any time and are thus not part of the ordering relation.*

the graph that does not contain a parallel cut is in a locked state, i.e. cannot be disassembled (see Figure 5).

4. Optimization

Section 3.1 and Algorithm I provide a means to test whether a given design graph G is rigid and can be assembled. Our goal is now to create such valid configurations. The complex coupling of constraints makes it imperative to augment the design process with an optimization method. By handling constraint satisfaction automatically, the user is relieved of this difficult task and can focus on high-level aspects of the design.

An inherent problem with optimization is that as the orientations and positions of the pieces change, edges would need to be added or removed from the intersection graph. This quickly leads to intractability. To avoid this, we make use of the observation that the angle and slit constraints only depend on the plane orientations, but not the spatial positioning of planar pieces nor their geometric shape. We can thus formulate the optimization as a two step procedure: Step one solves for a feasible solution using only the plane normal vectors $\mathbf{n}_1,\dots,\mathbf{n}_n$ as unknowns (Section 4.1). Step two then optimizes separately for the centroids $\mathbf{c}_1,\dots,\mathbf{c}_n$ and the boundaries of each piece to maintain the structure of the graph (Section 4.2).

4.1. Orientation Optimization

The input for the optimization is an initial arrangement of planes with the corresponding constraint graph $G=(\mathcal{V},\mathcal{E})$ that specifies which planes should be intersecting.

Satisfying angle constraints. To obtain tight slits we define an inequality constraint to force the intersection angles α_{ij} above the threshold mandated by fabrication (Section 3.1). For each edge e_{ij}, let $c_{\text{angle}}(e_{ij})$ denote how far the corresponding planar pieces are from satisfying the angle threshold measured as

$$c_{\text{angle}}(e_{ij}) = \left(\mathbf{n}_i \cdot \mathbf{n}_j\right)^2 - \sin^2 t_\alpha. \tag{2}$$

We use the square of the dot product to implicitly deal with the two possible orientations. It follows that the graph G is strongly rigid, if $c_{\text{angle}} < 0\ \forall e_{ij}$.

Satisfying slit constraints. In order to guarantee the existence of an assembly sequence, we need to create a series of parallel cuts in the graph that separate all planar pieces from each other. Our algorithm first finds a complete set of parallel or almost parallel cuts, the latter of which are then optimized to become parallel. The goal here is to induce minimal change to the current plane orientation to achieve edge parallelism and thus guarantee assembly.

Based on this objective, our algorithm proceeds as follows (see Figure 6): We first discard all hinge edges, i.e. edges that are not part of a cycle or collinear edges, since these edges provide trivial parallel cuts and thus do not impose any critical slit constraints. For the remaining edges that belong to a cycle, we perform k-means clustering based on edge direction, starting with k equal to the number of edges. We then progressively decrease k until one of the clusters contains a cut of the graph. To avoid unnecessary modifications to the plane normals when making the cluster parallel, we select the smallest subset of the cluster edges that define the cut. These edges are tagged and assigned to the same cut set for the optimization. We then continue the clustering recursively on the untagged edges of the generated subgraphs until all edges are covered.

This yields a decomposition of the edge set into disjoint sets $\mathcal{S}_1,\dots,\mathcal{S}_M, \mathcal{S}_k \subseteq \mathcal{E}$ that, when applied in sequence, form a complete series of cuts of G. Since these cuts are not necessarily parallel cuts, we align the directions of all edges within each cut set to a common direction to achieve parallelism. We introduce an auxiliary unit vector $\mathbf{s}_k$ for each set $\mathcal{S}_k$ to represent this common constraint direction and formulate equality constraints for each edge in $\mathcal{S}_k$ by quantifying the difference in alignment of the edge direction vectors to the vector $\mathbf{s}_k$. This difference is formulated for each $e_{ij} \in \mathcal{S}_k$ as

$$c_{\text{slit}}(e_{ij},\mathbf{s}_k) = \left\|(\mathbf{n}_i \times \mathbf{n}_j) \times \mathbf{s}_k\right\|^2. \tag{3}$$

Given the sequence of cut sets $\mathcal{S}_1,\ldots,\mathcal{S}_M$, it follows that G can be assembled, if $c_{\text{slit}}(e_{ij},\mathbf{s}_k)=0\ \forall\mathcal{S}_k$ and $\forall e_{ij}\in\mathcal{S}_k$.

Satisfying user constraints. Our goal is to find a configuration of plane normals that satisfies all angle and slit constraints, while being as close as possible to the design input of the user. We therefore introduce a closeness objective function $f_{\text{input}}(v_i)=\|\mathbf{n}_i-\mathbf{n}_i'\|_2$ that measures the deviation of the current plane normal to the initial input normal vector $\mathbf{n}_i'$. The corresponding energy term is given as

$$E_{\text{input}}=\sum_{i=1}^{n}\omega_i f_{\text{input}}(v_i)^2, \tag{4}$$

where the weights ω_i allow the user to specify important aspects of the design intent. Pieces that should not change substantially can be assigned a high weight ("fixing the piece"), while a low weight can be set for pieces that are free to deviate more strongly from the input configuration.

Constraint optimization. The goal of finding a configuration of planar pieces that is rigid and assemblable can now be formulated as a quadratic objective function with quadratic equality and inequality constraints:

$$\begin{aligned}&\arg\min_{\mathbf{n}_1,\ldots,\mathbf{n}_N} E_{\text{input}}\\ \text{subject to } &c_{\text{angle}}\le 0\ \ \forall\mathbf{e}_{ij},\\ &c_{\text{slit}}=0\ \ \forall\mathcal{S}_k\ \ \forall\mathbf{e}_{ij},\in\mathcal{S}_k,\\ &\|\mathbf{n}_i\|^2=1\ \ \forall\mathbf{n}_i,\\ &\|\mathbf{s}_i\|^2=1\ \ \forall\mathbf{s}_i.\end{aligned} \tag{5}$$

The unknowns are the auxiliary vectors $\mathbf{s}_1,\ldots,\mathbf{s}_M$, and the plane normals $\mathbf{n}_1,\ldots,\mathbf{n}_N$ that determine both the intersection angle of connected planes as well as their slit direction. We also add additional equality constraints to keep the vectors $\mathbf{n}_i$, $\mathbf{s}_i$ normalized. We use Sequential Quadratic Programming (SQP) to solve Equation 5. The accompanying video shows how an initial configuration is modified by the optimization [0:45-0:51].

Rigidity. If the optimization defined above yields a solution, the resulting 3D structure is guaranteed to be strongly rigid. If no solution is found, we relax the requirement of strong rigidity to introduce more degrees of freedom by allowing edges that are part of cycles to violate the angle constraint. This will introduce wider slits but still guarantees a locally rigid configuration. We simply select the non-hinge edge with the highest c_{angle}, remove the corresponding angle constraint, and re-run the optimization. We iterate this procedure until a feasible solution is found. Note that the optimization is guaranteed to find such a feasible solution. If all angle constraints are invalidated, strong rigidity is only enforced on the hinge edges. Since these can only be part of a trivial parallel cut, they are not in conflict with the slit constraints. Consequently, a solution always exists (but might be far from the input), for example when all edges become parallel. In this case, the configuration might need to be checked for global rigidity (as in Figure 4).

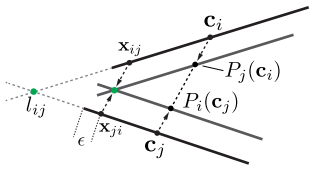
4.2. Position Optimization

The optimization described above solves for plane orientations that satisfy assembly and rigidity constraints. Changes in plane orientation for a piece v_i are effected by rotating around the centroids $\mathbf{c}_i$. As a result, some pieces that are connected by an edge in the graph G might no longer be intersecting, while others might now intersect even though there is no corresponding edge in G.

For each newly introduced intersection, if a corresponding edge can be introduced without violating the constraints, we add it to G. Then, since the intersection pattern defined by the edge set $\mathcal{E}$ is closely related to visual or functional semantics of the design, we apply a second optimization step to ensure that the graph structure is preserved. Recall that the $\mathbf{c}_i$ and the boundaries of the pieces have no direct influence on the angle and slit constraints as these only depend on the plane normals. Thus we can, independently of the above orientation optimization, modify the relative positioning of the planar pieces and their contours to maintain the graph intersection pattern. We can also handle collisions during the assembly process in this step.

During the design process, the user can define the contours explicitly or may optionally choose a 3D guiding volume (see for example Figure 8). Each piece would then be maximally intersected with the guiding volume to obtain its contours. This choice results in slight differences in the method that will be noted below.

Retaining intersections. Let us assume that the planar pieces v_i and v_j of a given edge e_{ij} no longer intersect after their normals have been modified by the orientation optimization. Let line l_{ij} be the intersection of the infinite planes of $\mathbf{p}_i$ and $\mathbf{p}_j$ (see inset). For $\mathbf{p}_i$, we find the furthest point lying on the piece in the direction of $\mathbf{c}_i$ to its projection on l_{ij} and subtract a small distance ε to account for the slit width and allow for stability. We call this point x_{ij}. A similar operation on $\mathbf{p}_j$ gives us $\mathbf{x}_{ji}$. The target points $P_j(\mathbf{c}_i)$ and $P_i(\mathbf{c}_j)$ are such that $\mathbf{x}_{ij}$ and $\mathbf{x}_{ji}$ coincide:



$$P_j(\mathbf{c}_i)=\mathbf{c}_i-\frac{\mathbf{x}_{ji}-\mathbf{x}_{ij}}{2} \tag{6}$$

If the pieces intersect then $P_j(\mathbf{c}_i)=\mathbf{c}_i$. If the user specifies a guiding volume, then we additionally project $\mathbf{x}_{ij}$ into the volume to ensure an intersection, again adding an ε tolerance. We formulate the optimization as a minimization of

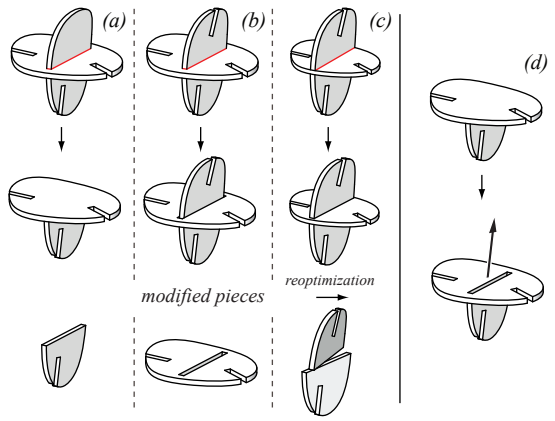


Figure 7: *After optimization of orientations, intersections may arise that prevent assembly (red lines). The system clips each piece such that no new intersections are introduced. (a) and (b) illustrate clipping without changing the intersection graph. If it is not possible to keep the same graph, we split a piece into two and re-optimize for orientation, as in (c). The bottom row shows only the modified pieces for illustration. Clipping can also be performed such that the assembly path is free of collisions (d). Afterwards, the piece can be taken out in the direction of assembly. Note this will also ensure that clipping of (b) will not introduce assembly constraints.*

the following energy:

$$E_{\text{pos}} = \sum_{i=1}^{n} \sum_{e_{ij} \in \mathcal{E}} \|\mathbf{c}_i - P_j(\mathbf{c}_i)\|, \tag{7}$$

where e_{ij} are the out-edges of v_i. We optimize for all positions simultaneously using a non-linear alternating projection scheme (see [BXM03]), iterating until convergence. The optimization converges to the closest local minimum to the input since each step weakly decreases the distance between each pair of pieces. While it still might be possible in degenerate cases that all $\mathbf{c}_i$ converge to a single point, we never observed this in our experiments and usually 10 iterations is sufficient.

Contouring. The position optimization displaces planar pieces to ensure that all intersections specified in the constraint graph are realized. This operation might lead to new intersections requiring new edges to be added. To avoid these and the resulting additional constraints, we apply a clipping algorithm that iteratively clips pieces, changing only contours, to avoid collisions. This leaves no new intersections, but can result in split pieces and possibly disconnected structures. Since the choice of which planes clip which and in what order can change the piece geometry and aesthetics, we perform only the necessary clipping in order for the collisions to be avoided, and default sensibly in ambiguous cases.

We aim to clip without changing the intersection graph: Say there are two intersecting pieces $\mathbf{p}_i$ and $\mathbf{p}_j$ where $e_{ij} \notin \mathcal{E}$. If the pieces do not intersect along the entire length of one piece, then we can simply clip without changing the graph since the pieces stay connected in relation to the slits (see Figure 7(a-b)). Otherwise, we split one piece by another such that the graph does not get disconnected. In ambiguous cases (neither split disconnects the graph or both do) we default to splitting the piece that has no slits on one side of the intersection or the one with most intersections. By clipping one piece with another, we effectively split a vertex in our intersection graph into two vertices, which introduces additional degrees of freedom in our constraint graph. We therefore rerun the orientation optimization to ensure constraint satisfaction (see Figure 7(c)).

Global collisions. As opposed to only checking for introduced intersections, we can also look at the global configuration and apply a final step that aims at resolving collisions by cutting pieces if they block the motion path of another piece. This procedure works in the order of disassembly, i.e. we start with the piece that would be taken out first from the assembled object. To simplify the collision analysis, as in [HBA12], we only consider the straight-line motion space defined by the slit direction. Thus every piece defines a collision volume that can be generated by sweeping the piece geometry in the slit direction. We automatically detect collisions of this volume with all other pieces that have not yet been taken out and clip these pieces to allow full motion (see Figure 7(d)). If this splits a piece into two, we continue as described in the previous paragraph. Recall that every slit can be realized in two opposing orientations (see inset illustration in Section 3). Therefore, if the graph becomes disconnected, we also have the degree of freedom of choosing the opposite orientation for each parallel cut.

5. Design Process

With the optimization method in place, we now introduce an interactive design process that allows the user to create 3D objects composed of intersecting planar pieces. The interactive nature of the design process is best appreciated in the accompanying video and supplementary materials.

Constraint editing. In order to keep the problem tractable, we keep the intersection graph G fixed during a single optimization. However, at any stage during the design, the constraint graph can be modified by the user to explore different design options, adding possible intersections and reoptimizing to see the possible resulting structure. To increase the degrees of freedom the user can discard a piece, i.e. remove a node from the graph, or eliminate an intersection, i.e. delete an edge. One advantage of the clustering method illustrated in Figure 6 is that we can simply permute the ranking of the cuts to propose different design solutions to the user. This allows exploring design alternatives by simply browsing through a set of solutions that all satisfy the constraint

sets (see accompanying video [0:52-1:10]). For the ambiguous cases in Section 4.2, the user can choose between possible clippings.

Another handle that is at our disposal for avoiding collisions is the orientation of slits within each parallel cut set. Recall that every slit can be realized in two opposing orientations (see inset illustration in Section 3). After the optimization, we make sure that the slits are all facing the same direction for each parallel cut. However, one can freely choose the opposite orientation for each slit, as long as the parallel cuts remain in the same orientation. This will only change the global collision clipping and the user can choose the opposite orientations for aesthetic or static reasons. For example, if the wrong orientation is selected, the pieces might slide out due to gravity or other forces acting on the design.

6. Results

We show several designs created with our system to evaluate the versatility and effectiveness of our approach. Figure 8 shows the design of a wooden 3D toy based on a guiding surface mesh. This dinosaur toy was designed in about 20 minutes with a concrete idea of the intended structure. Figure 9 shows an application in architecture, highlighting the potential of our method for design exploration and the creation of scale models. A feasible construction was ensured with our method with the process finished in less than 15 minutes. The assembly process as prescribed by our framework can be seen in the accompanying video. Figure 10 illustrates how functional custom-designed furniture can be generated with our system. These examples were designed by exploration with only a minimal idea of the intended outcome. Even with a relatively simple geometry, the assembly of this model requires joining multiple pieces at the same time along non-trivial parallel cuts. Existing methods that incrementally add one piece at a time [HBA12] cannot deal with such configurations.

All examples in Figures 8 to 10 are strongly rigid, i.e. only contain tight slits. Figure 11 is an example with non-tight slits that still maintains rigidity. The assembly of such models is in general more complex, since non-tight slits can lead to non-rigid constellations during construction that only become rigid once the corresponding cycles are closed. In return, we obtain a substantially richer design space, in particular when working with orthogonal cutting devices such as laser cutters. This is an important advantage over previous methods such as [HBA12] that rely on tight slits for rigidity and therefore quickly lead to grid-like structures when dealing with dense constraint graphs. With the chair and table examples, milling took one hour and the assembly of each model required about half an hour. The geometric information created during the design, i.e. the contour curves of the planar pieces, is directly translated into machining instructions via scripting. To facilitate assembly, we can optionally create unique matching IDs for each slit that are laser-etched into the piece. An assembly plan can be created directly from the partial ordering generated by the clustering algorithm of Section 4 as illustrated in Figure 10 and the supplementary materials, and the system shows animations of the assembly to aid the user. All our examples have been physically fabricated and assembled, demonstrating that our system covers the full chain from digital design to production.

Limitations. We use a greedy clustering strategy to generate a series of parallel cuts in order to avoid the full combinatorial search that would be computationally intractable for all but the simplest graphs. Hence we are not guaranteed to find a global optimum in the sense of closest valid configuration to a given set of input constraints. In our experiments, however, we found that the greedy choice was as good or better than all subsequent clustering or user-assisted choices for cuts.

Currently, we do not incorporate statics or material physics into the optimization. Similarly, we ignore potential limitations of the production process, such as restrictions of the tool path for milling machines, or specific properties of the material, such as anisotropies (e.g. in wood) that would favor certain directional alignments. These limitations offer a trade-off between the effectiveness of the design approach and the computational complexity. They are a consequence of the specific abstraction of our model that focuses primarily on the geometric relations and properties resulting from the main fabrication and assembly constraints.

Future Work. The limitations discussed above are one immediate target for future research. Integrating static properties or other performance objectives into the optimization

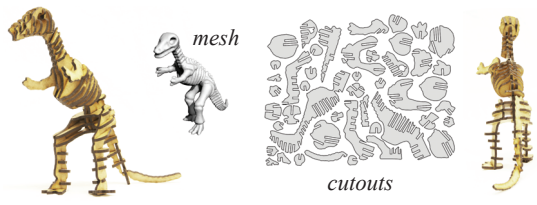


Figure 8: *A 3D toy composed of orthogonally intersecting planar pieces. The mesh of the dinosaur serves as a guiding volume for the design.*

Figure 9: *An architectural design study of an outdoor pavilion made from orthogonally intersecting pieces.*

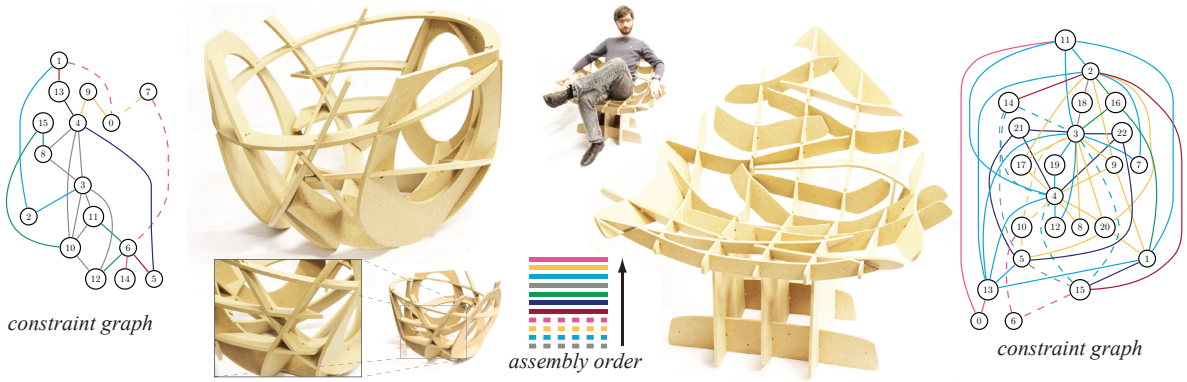


Figure 10: *The base of a coffee table and a freeform chair milled from medium density fiberboard (MDF) (see also Figure 1). Complex joints like the one illustrated in the zoomed image can easily be created with our method. The design graphs illustrate the complex coupling of constraints. The parallel cut sets computed by the optimization are indicated by color in the order of assembly. The smallest intersection angle is* 50°*, as mandated by the CNC milling machine.*

provides numerous opportunities to improve the effectiveness of the design process, perhaps with a method similar to [UIM12]. The process would also be aided with additional semantic controls and shape-aware operations. Another promising avenue for future research is to consider a broader spectrum of material behavior. For example, bendable pieces made of thin wooden or metal sheets can be fabricated with the same technology as the rigid pieces currently considered in our work. Developable surfaces lead to constructions that are much more general, but also more difficult to control.

Further potential for future work lies in the assembly process itself. Our goal previously was to make assembly as easy as possible. Yet some applications might target the opposite. For example, 3D puzzles are intriguing because finding an assembly sequence is difficult and often requires playful experimentation [LFL09, XLF*11]. We believe that interlocking planar pieces have a great potential for challenging puzzles, even when using only few pieces. For example, the 7-piece cyclic model shown in the accompanying video and supplementary materials was perceived by several test persons as very difficult to assemble without explicit assembly instructions. This example illustrates the potential of our approach for creating appealing 3D puzzles.

Figure 11: *A lampshade design uses non-tight slits to facilitate a complex intersection pattern. Our optimization still guarantees that the assembled structure is completely rigid.*

7. Conclusion

We show in this paper that interlocking planar pieces offer rich design possibilities for creating compelling 3D models that can be fabricated with low-cost machinery and assembled without glue, screws, or other means of support. We demonstrate that optimization is crucial for exploiting the full creative potential of this type of construction. Only through computational means can we handle the intricate coupling of fabrication and assembly constraints that lead to a highly complex design space. Our approach effectively hides this complexity, allowing even non-expert users to quickly create interesting 3D designs.

Acknowledgments. We thank Stephane Grandgirard and Russell Loveridge for help with fabrication of the initial tests, and Christopher Robeller, Hans Ulrich Buri, and the IBOIS lab, as well as Trevor Patt and Nathaniel Zuelke, for help with fabrication. We thank Sofien Bouaziz for the discussion and technical help, as well as Mario Deuss, Boris Neubert, Thibaut Weise, and all the members of LGG.

References

[BXM03] BOYD S., XIAO L., MUTAPCIC A.: Alternating projections. *Lecture notes of EE392o, Stanford University, Autumn Quarter 2004* (2003). 8

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. *ACM Trans. Graph. 23* (Aug. 2004), 905–914. 2

[DDSD03] DÉCORET X., DURAND F., SILLION F., DORSEY J.: Billboard clouds for extreme model simplification. *ACM Trans. Graph. 22*, 3 (2003), 689–696. 2

[DO07] DEMAINE E., O'ROURKE J.: *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, 2007. 2

[Dow93] DOWLING N.: *Mechanical behavior of materials: engineering methods for deformation, fracture, and fatigue*. Prentice Hall Englewood Cliffs, NJ, 1993. 4

[Gla02] GLASSNER A.: Interactive pop-up card design. 1. *Computer Graphics and Applications, IEEE 22*, 1 (2002), 79–86. 3

[Har07] HART G.: Modular kirigami. *Journal of Mathematics and the Arts 1*, 1 (2007), 21–28. 2

[HBA12] HILDEBRAND K., BICKEL B., ALEXA M.: crdbrd: Shape fabrication by sliding planar pieces. *Comput. Graph. Forum 31*, 2 (2012). 3, 8, 9

[Jac07] JACKSON B.: Notes on the rigidity of graphs. In *Levico Conference Notes* (2007). 4

[Joh10] JOHNSON S.: The nlopt nonlinear-optimization package, 2010. 12

[Kil06] KILIAN A.: *Design exploration through bidirectional modeling of constraints*. Massachusetts Institute of Technology, Dept. of Architecture, 2006. 3

[Kra94] KRAFT D.: Algorithm 733: Tomp–fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software (TOMS) 20*, 3 (1994), 262–281. 12

[LFL09] LO K.-Y., FU C.-W., LI H.: 3d polyomino puzzle. *ACM Trans. Graph. 28* (December 2009), 157:1–157:8. 3, 10

[LJGH11] LI X.-Y., JU T., GU Y., HU S.-M.: A geometric study of v-style pop-ups: Theories and algorithms. *ACM Trans. Graph. 30*, 4 (2011), 98:1–10. 3

[LOMI11] LAU M., OHGAWARA A., MITANI J., IGARASHI T.: Converting 3d furniture models to fabricatable parts and connectors. *ACM Trans. Graph. 30* (Aug. 2011), 85:1–85:6. 3

[LSH*10] LI X., SHEN C., HUANG S., JU T., HU S.: Popup: automatic paper architectures from 3d models. *ACM Trans. Graph. 29*, 4 (2010), 111. 3

[MA08] MILLER J., AKLEMAN E.: Edge-based intersected polyhedral paper sculptures constructed by interlocking slitted planar pieces. *Proceedings of the 2008 Bridges Conference on Mathematical Connections in Art, Music, and Science* (2008). 2

[MSM11] MCCRAE J., SINGH K., MITRA N.: Slices: A shape-proxy based on planar sections. *ACM Trans. on Graph.(Proc. SIGGRAPH Asia) 30*, 6 (2011). 2

[Séq12] SÉQUIN C.: Prototyping dissection puzzles with layered manufacturing. *Fabrication and Sculpture Track, Shape Modeling International Conference* (2012). 3

[SLMI11] SAUL G., LAU M., MITANI J., IGARASHI T.: Sketchchair: An all-in-one chair design system for end users. In *Proceedings of the fifth international conference on tangible, embedded, and embodied interaction* (2011), ACM, pp. 73–80. 3

[SP11] SCHWARTZBURG Y., PAULY M.: Design and optimization of orthogonally intersecting planar surfaces. *Computational Design Modelling* (2011), 191–199. 3

[Tac10] TACHI T.: Freeform rigid-foldable structure using bidirectionally flat-foldable planar quadrilateral mesh. In *Advances in Architectural Geometry 2010*, Ceccato C. et al., (Eds.). Springer, 2010, pp. 87–102. 2

[UIM12] UMETANI N., IGARASHI T., MITRA N. J.: Guided exploration of physically valid shapes for furniture design. *ACM Transactions on Graphics 31*, 4 (2012). 4, 10

[WLMI10] WILLIS K. D., LIN J., MITANI J., IGARASHI T.: Spatial sketch: bridging between movement & fabrication. In *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction* (2010), ACM, pp. 5–12. 3

[XLF*11] XIN S., LAI C., FU C., WONG T., HE Y., COHEN-OR D.: Making burr puzzles from 3d models. *ACM Trans. Graph. 30*, 4 (2011), 97. 3, 10

Appendix A: Orthogonal Intersection

We provide a brief analysis of the design space for orthogonally intersecting planar pieces with tight slits. The angle constraint for strong rigidity effectively eliminates a degree of freedom in the relative orientation of connected pieces. Only a rotation around the respective plane normals is possible. Consequently, any two non-parallel planar pieces $\mathbf{p}_i$ and $\mathbf{p}_k$ can be connected by a piece $\mathbf{p}_j$ with unique plane normal $\mathbf{n}_j = \mathbf{n}_i \times \mathbf{n}_k$. In the special case where $\mathbf{p}_i$ and $\mathbf{p}_k$ are parallel, i.e. $\mathbf{n}_i = \mathbf{n}_k$, any plane with $\mathbf{n}_j \cdot \mathbf{n}_i = 0$ will be a valid connection. These restrictions on the connection of pieces have interesting consequences on the design space.

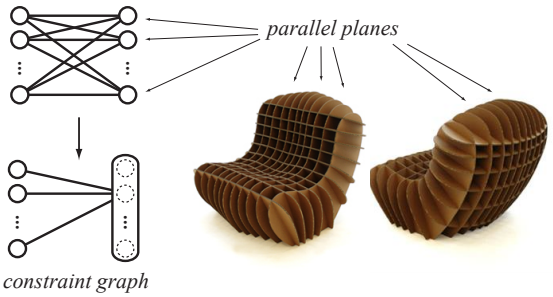


Figure 12: *A typical grid design commonly observed in existing cardboard models requires one family of planes to be parallel. In the constraint graph, these parallel planes can be collapsed to a single node, leading to a configuration without cycles that trivially ensures that all slit constraints can be satisfied.*

Conceptually, assuming consistent slit orientations, a set of parallel planes connected to the same pieces can be collapsed to a single piece. An example is given in Figure 12, where the constraint graph is a bipartite graph. The vertex set shown on the right can be collapsed into a single vertex, which eliminates all cycles and thus trivially ensures that the graph can be assembled. Densely connected graphs of this sort naturally lead to parallel planes in the arrangement. In this particular example, all nodes have valence $n/2$, where $n = |\mathcal{V}|$. As a result, any cut has at least $n/2$ edges. To ensure that the graph can be split along the cut, these edges need to be parallel. This leads to at least $n/2$ parallel planar pieces.

This example suggests that parallel edges of a cut lead to parallel planar pieces. However, this is not necessarily the case. Let us consider a simple elementary cycle consisting of n pieces, i.e. $2n$ unknown parameters for the n plane normals (see Figure 13). The available degrees of freedom are reduced by angle constraints, the global rigid motion of the structure, the constraint that the pieces form a cycle, and the existence of a parallel cut, which necessitates that (at least) two edge directions must be parallel. The formula below quantifies the degrees of freedom $F(n)$ of an elementary cycle of size n:

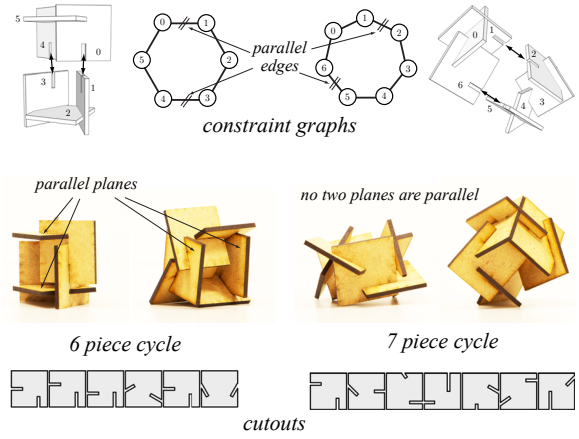
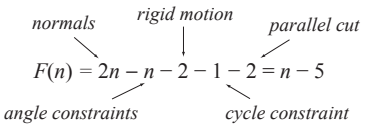


Figure 13: *Elementary cycles with orthogonally intersecting planes. A parallel cut consisting of two edges ensures that the cycles can be assembled. With six or less pieces, such cycles must contain at least two parallel planes. While not specifically designed as puzzles, the assembly of these models can be challenging without a given assembly plan, since the parallel cut is the only way to close the cycle. This illustrates the potential of our approach for generating recreational 3D puzzles.*



Consequently, for a cycle with 6 pieces, the single remaining degree of freedom is the rotation of the two pieces associated with a parallel edge around the axis defined by the edge direction. As it turns out though, in this case the two pieces connecting the parallel edges must be parallel. Thus the smallest cycle that can be strongly rigid and assemblable and does not contain any parallel pieces consists of 7 pieces.

Appendix B: Design Process Details

Figure 14 illustrates a typical design session. We deliberately chose an example where the algorithm leads to significant change of the plane orientations to better visualize the effect of the optimization. While these alterations are often more subtle, they can involve a large subset of pieces. Satisfying the constraints by manually moving and rotating the pieces would thus be very cumbersome, even for simple designs consisting of few pieces. At the end of the process, assembly instructions can be directly generated from the assembly graph, see Figure 15.

Implementation. Our framework is implemented in C++ using the Qt library. We use the Boost Graph Library for all graph operations and Boost Geometry for performing

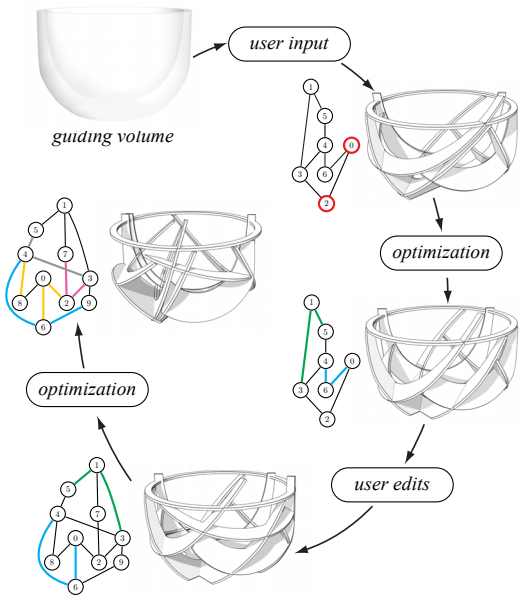


Figure 14: *Illustration of a typical design process. The user first specifies a 3D guiding volume, then positions and orients several initial pieces. The constraint graph automatically computed from the intersections typically violates some angle constraints (red nodes) and/or does not contain the required parallel cuts. The optimization then solves for a configuration that satisfy the constraints (colored edges are parallel). The user adds more planar pieces and iteratively refines the design, relying on the optimization method to ensure constraint satisfaction.*

CSG operations and collision detection. We also use the C Clustering Library written by Michiel Jan Laurens de Hoon for clustering the orientations. We use Sequential Least-Squares Quadratic Programming as implemented by NLopt [Kra94, Joh10] to solve Equation 5.

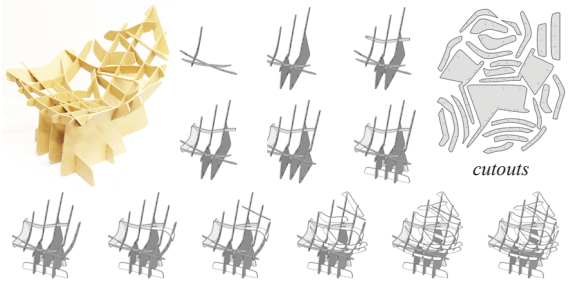


Figure 15: *Assembly sequence of the freeform chair model.*