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.

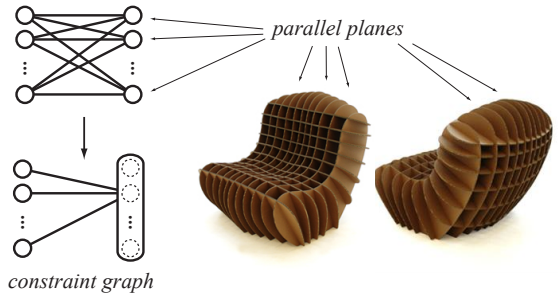


Figure 1: *A typical grid design commonly observed in existing cardboard models requires one family of planes to be parallel. In the constraint graph, these parallel planes can be collapsed to a single node, leading to a configuration without cycles that trivially ensures that all slit constraints can 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 1, 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 2). 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:

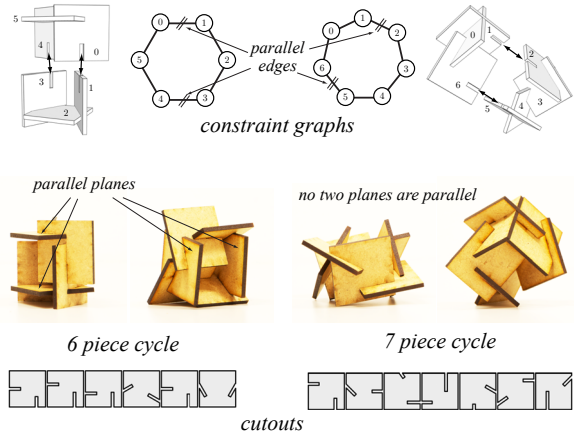
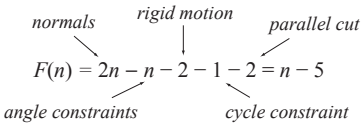


Figure 2: *Elementary cycles with orthogonally intersecting planes. A parallel cut consisting of two edges ensures that the cycles can be assembled. With six or less pieces, such cycles must contain at least two parallel planes. While not specifically designed as puzzles, the assembly of these models 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 3 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 4.

We present a comparison of plane orientations and positions before and after a single step of optimization in Figure 5. For inputs we use sample meshes and planes generated by the method of McCrae and colleagues [MSM11].

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 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.

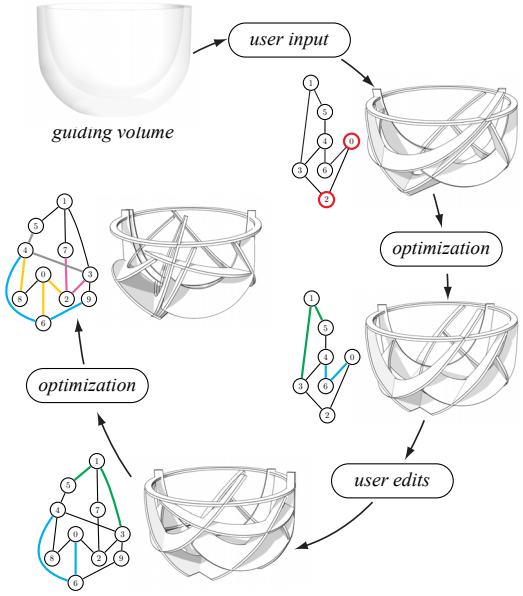


Figure 3: *Illustration of a typical design process. The user first specifies a 3D guiding volume, then positions and orients several initial pieces. The constraint graph automatically computed from the intersections typically violates some angle constraints (red nodes) and/or does not contain the required 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.*

References

[Joh10] JOHNSON S.: The nlopt nonlinear-optimization package, 2010. 2

[Kra94] KRAFT D.: Algorithm 733: Tomp–fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software (TOMS) 20*, 3 (1994), 262–281. 2

[MSM11] MCCRAE J., SINGH K., MITRA N.: Slices: A shape-proxy based on planar sections. *ACM Trans. Graph. 30*, 6 (2011), 464–470. to appear. 2

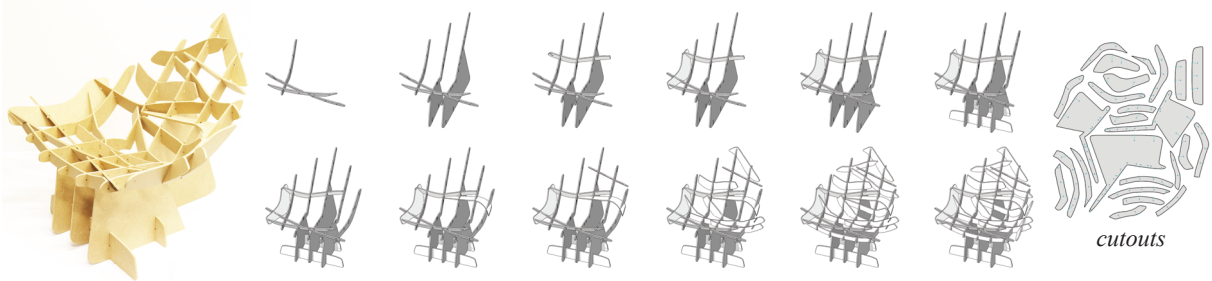


Figure 4: *Assembly sequence of the freeform chair model.*

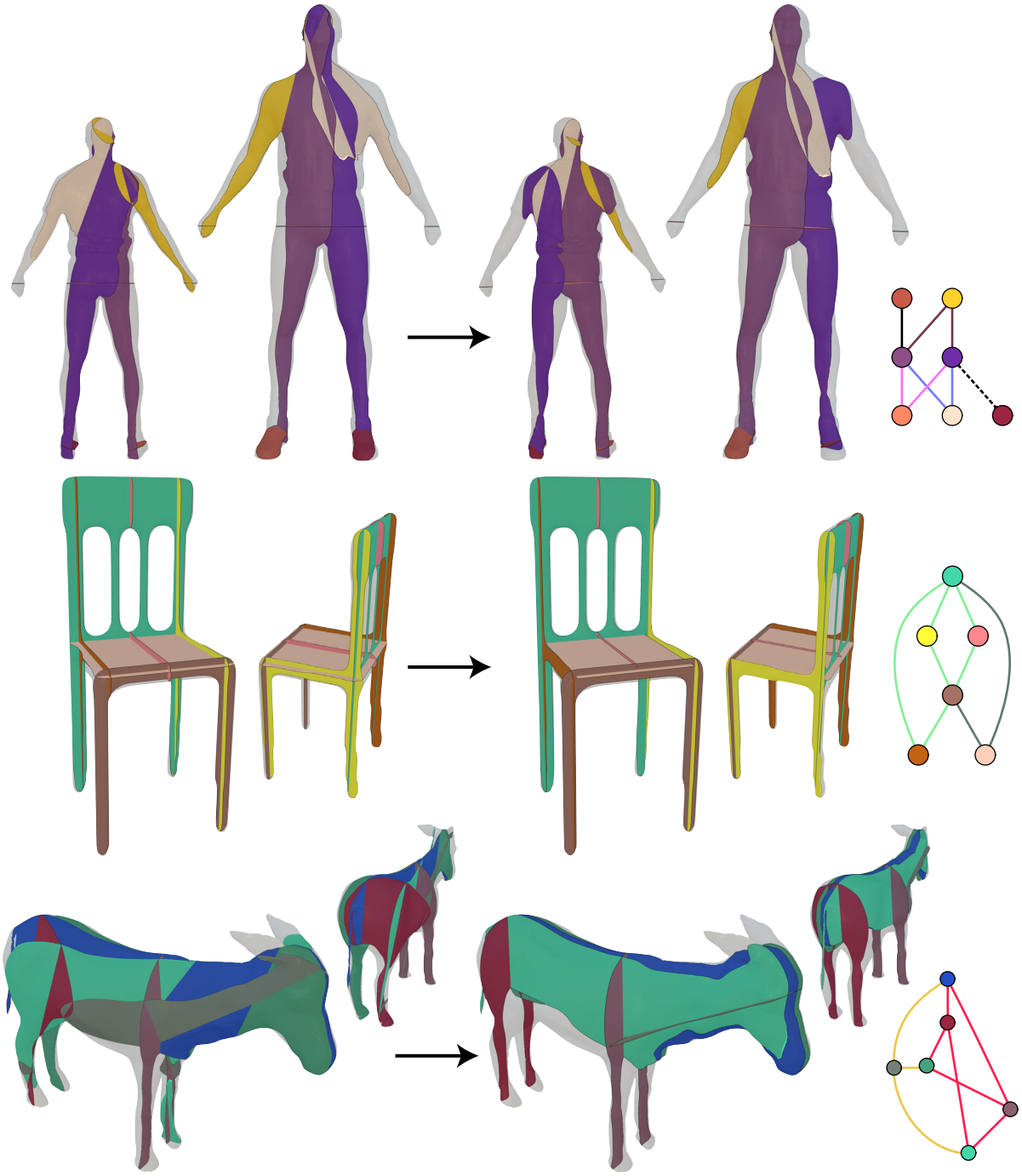


Figure 5: *We use the method of McCrae and colleagues to generate initial plane positions (left). After a single optimization step, the results at right are produced. The respective constraint graphs are shown at the far right. In a normal editing session, the user would continue adding planes and constraints from this point.*