Design and Optimization of Orthogonally Intersecting Planar Surfaces

Yuliy Schwartzburg and Mark Pauly

1 Introduction

We present a method for the design of 3D constructions from planar pieces that can be cut easily and cheaply with laser cutters and similar Computer Numerical Control (CNC) machines. By cutting tight slits in intersecting pieces, they can be slid into each other forming stable configurations without any gluing or additional connectors. These constructions enable quick prototyping and easy exploration of shapes, and are particularly useful for education. We propose a constraint-based optimization method and computational design framework to facilitate such structures.

Planar surfaces can be connected, without any other materials or bindings, by cutting slits of the width of the material in the direction of the surface normal. These constructions are often found in cardboard and wooden 3-D puzzles (see Fig. 2). As laser cutters do not permit cuts that are not orthogonal to the surface, the slits would need to be larger to accommodate intersecting pieces at non-right angles (see Fig. 3). This would lead to unstable configurations and the need for connecting structures. Our goal is to avoid this complication and retain the simplicity of orthogonally intersecting pieces.

As shown in Fig. 3, when considering a pair of intersecting orthogonal pieces, each piece has one degree of freedom in rotation. This introduces a constrained design space for composing more complex shapes from multiple interconnected pieces. The common method for generating such objects is using parallel or semantic cross-sections (as in Fig. 2), but this severely limits the design space. Slightly more complex methods grow the structure generatively, starting with a skeleton and ensuring orthogonality step by step, in either a manual or procedural fashion. However, these approaches quickly become infeasible due to combinatorial complexity

Yuliy Schwartzburg · Mark Pauly
Computer Graphics and Geometry Laboratory (LGG) EPFL Lausanne, Switzerland

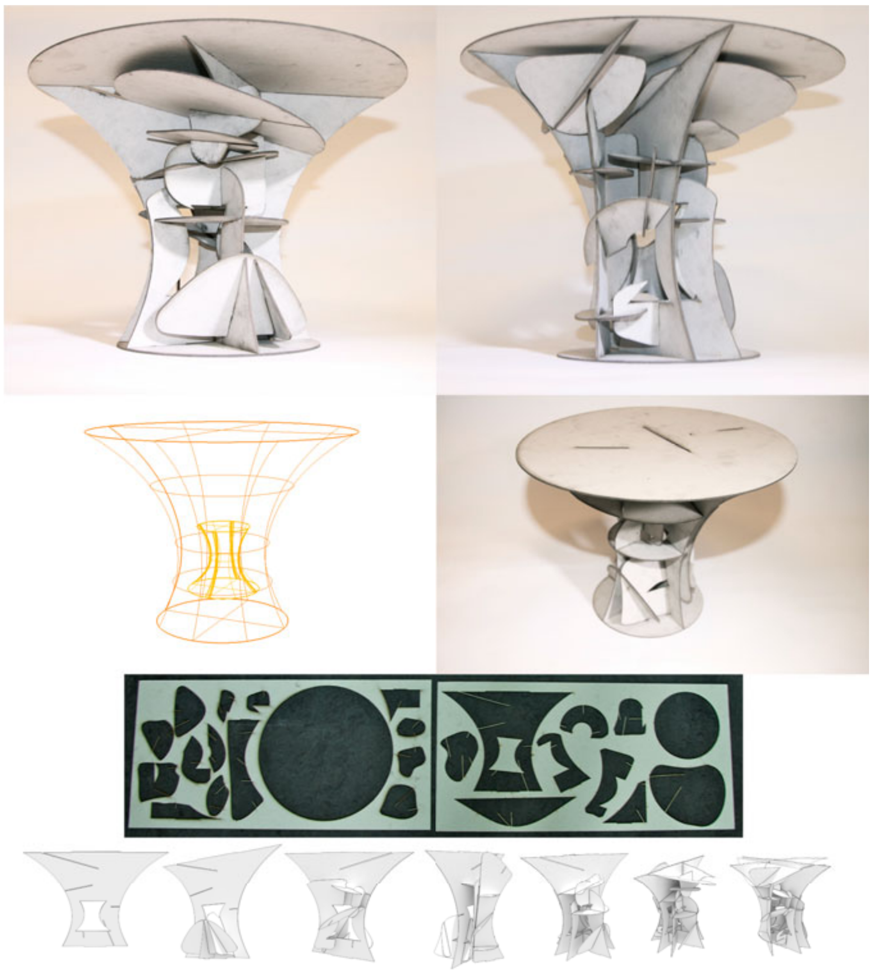

Fig. 1 A physical table built with orthogonally intersecting planar pieces using our design method. Also pictured are the hull wireframe, the negatives of the corresponding pieces, and the building process.

inherent in the global coupling of orthogonality constraints. The design space is difficult to predict and one design step can have unforeseeable implications for the end product (as seen in Fig. 5).

2 Related Work

The utilization of computational techniques during the design process has been studied extensively, and in particular, in the context of performative and construction-aware design. Axel Killians doctoral thesis focuses on the interrelation between

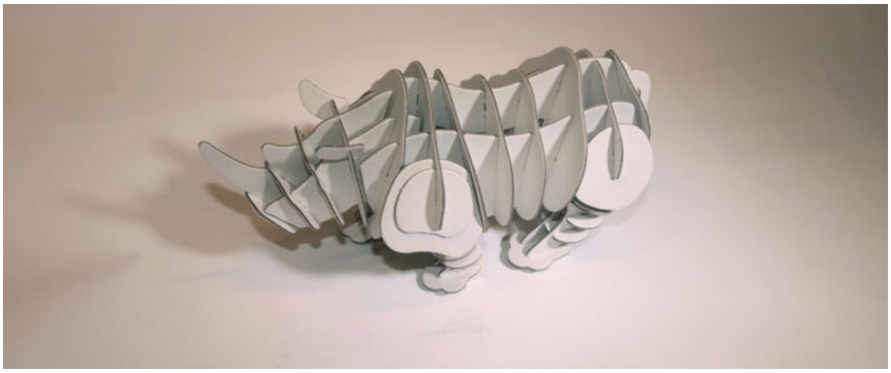

Fig. 2 An existing commercial cardboard artwork. Note that this was designed according to the skeleton of the rhinoceros and its cross-sections do not exhibit much variation in orientation.

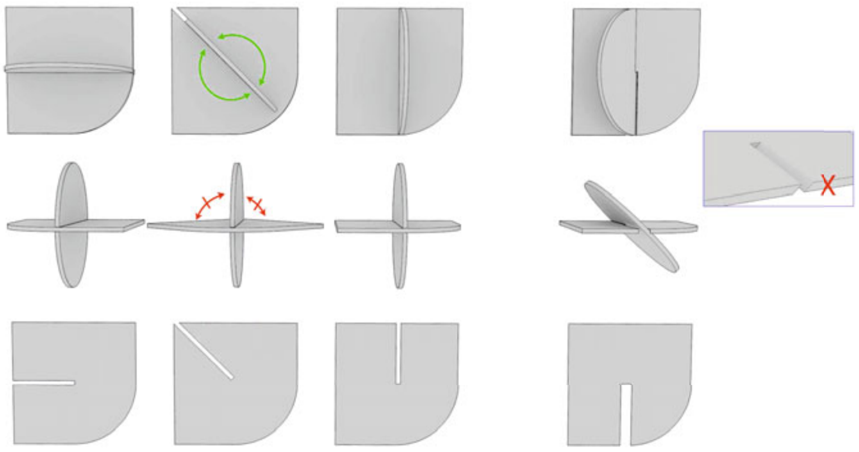

Fig. 3 Orthogonal intersection of two planar pieces allows rotation of the pieces relative to each other along one axis without violating orthogonality constraints. The first row shows assembled pieces while the second row shows a side view, in which the 90-degree intersection is apparent. The bottom row shows the cutting curve of the bottom piece. Fixing the bottom piece, the green arrow represents valid rotations of the second piece, as can be seen in the first three columns, which still form intersections of 90 degrees. All other rotations (red arrow) are not allowed. The fourth column represents one such invalid orientation. To produce tight, stable slits, the result should look like the inset piece. However, this is not easily accomplished with a laser cutter. Therefore, much wider, unstable slits are introduced.

design and constraints [3]. Oxman explores the link between performative design and computational geometry [6]. Mangelsdorf explores different strategies to deal with complex geometries and praises a hybrid approach that enables a high degree of freedom in the development of the form, but integrates concepts based on physics, form description and fabrication. Particularly, the Médiacité Liège consists of intersecting (but not orthogonal) planar rib sections that were designed with a mixture of physical form-finding and mathematical descriptions ([4], pp. 41-45). The Sphere Project ([1], pp. 103-111)) explores intersections of planar surfaces. They generate planar intersecting rings using an evolutionary process, however, their rings do not intersect orthogonally, leading to the introduction of welds, joints, and discontinuities. Pottman in ([7], p. 74)emphasizes the importance of“the development of efficient optimization algorithms and the incorporation into user-friendly rationalization software tools.” While our design space is too simplistic to fully capture the complexity of large-scale architectural projects, it nevertheless allows study of how computation and optimization can be leveraged to enable an effective design process.

3 Design Process

We present a construction-aware design tool for enabling effective design of structures containing orthogonally interconnected pieces through computational support. The tool facilitates the building of the described structures starting from cross-sections of an initial manifold mesh or boundary representation that defines the hull. It is designed to be time-saving and intuitive while giving freedom to the architect to prescribe or break necessary constraints. We use an iterative design process in which the designer places planar surfaces within the desired volume and a feasible solution is calculated using an optimization approach based on constraint satisfaction. As the design process continues, the planes are updated based on the coupling defined by the constraints. Finally, slits are inserted into the surfaces and curves are produced, ready to be sent to a laser cutter. Our tool guarantees satisfaction of constraints during the design process, and in contrast to scripting, facilitates intuitive design handles.

3.1 Optimization Constraints

For details on the constraint satisfaction solver and its implementation we refer to [2]. In this paper, we concentrate on the design process of utilizing such a solver. It is designed to find a feasible solution satisfying all necessary (hard) constraints (in our case, orthogonality) while remaining close to the original design. Additional design criteria can be specified in the form of soft constraints. These constraints will be satisfied as well as possible within the design space defined by the hard constraints. Additional constraints necessary for a specific model can be programmed and inserted into the solver (stability, tension, fixed boundaries, daylighting, heat distribution, etc.)

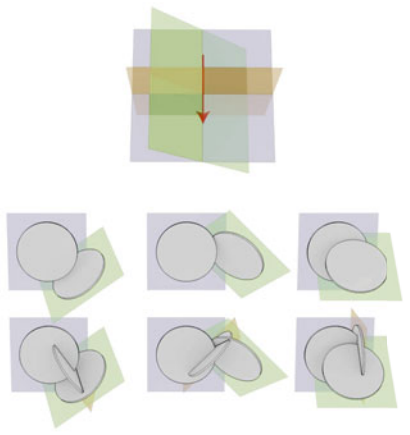

Fig. 4 If two planes are fixed, there is only one allowed orientation for a third plane connecting the two. The purple and green planes form an intersection line (red) that uniquely determines the normal direction of the plane connecting the two (yellow). Three examples are illustrated, before and after adding the third plane.

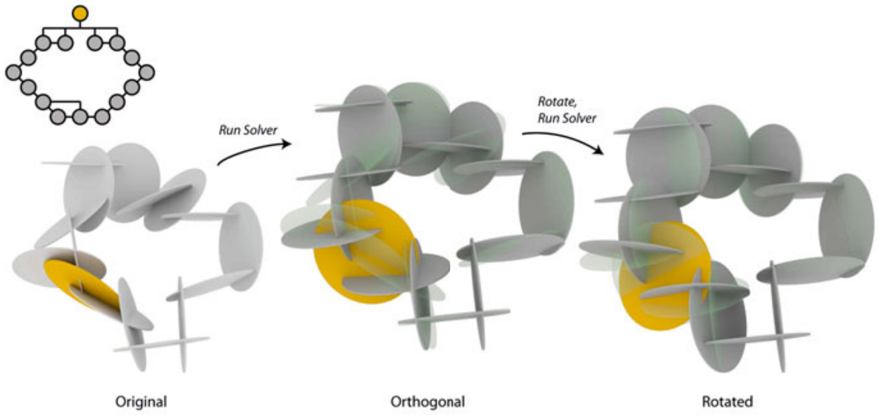

Fig. 5 A cyclic arrangement of pieces. The graph representing the connections is shown at top-left. Beginning with a non-orthogonal construction at left, we run the optimization solver. Then, as a user edit operation the yellow piece is rotated and the optimization runs again. The transparencies in green illustrate the previous locations of the pieces for comparison. Note that rotating the yellow piece produces changes in many neighboring pieces, and slight changes in far pieces.

3.2 Orthogonality

The basic hard constraint of our system is that two intersecting pieces must meet at a right angle. If we look at a system of two pieces (A and B), if we fix piece A we leave one degree of freedom for piece B, rotation around the normal of A. Consider a system of three pieces (A, B, and C) with B intersecting A and C, but A and C independent. If we fix A and C and A is not parallel to C, then B has a fixed orientation (see Fig. 4) If A is parallel to C, then this can be treated as the first case.

To represent the global configuration, we can consider a bidirectional graph of connections $G = (V, E)$, where a vertex V represents a planar pieces and an edge E represents an intersection between two pieces. While it is not difficult to ensure orthogonality at the start of the design process in an iterative approach, as cycles are introduced into G, a new intersection or a change in orientation can propagate through the graph, requiring modification of a number of pieces. However, by using optimization techniques, we can automatically calculate the minimal necessary modification of the placed pieces in order to satisfy the constraints.

In order to satisfy orthogonality in the optimization, we need to rotate each non-orthogonal intersecting pair as explained in Figure 6. The solver iterates, each time satisfying constraints and then merging the results, until a consistent state is reached. It is often not possible to satisfy every intersection (or the results become uninterestingly parallel); an unwanted intersection can then be eliminated by cutting a piece at a set distance from the intersection.

We use an iterative approach of optimizing after placing each plane. The optimization can be run once at the end of the design process as a post-rationalization as well (see Fig. 5). However, there is less chance of deviating far from the intended design when optimizing iteratively.

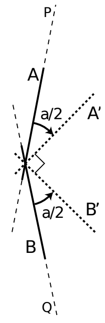

Fig. 6 Consider two pieces, A and B, defined on planes P and Q respectively. We take the intersection of P and Q as the rotation axis. Then, we rotate the pieces around their respective centroids (in opposite directions) by $a/2$, where a is 90 degrees subtracted by the angle between P and Q

3.3 Position

By itself, the orthogonality constraint would rotate each piece such that globally, the movement is minimal. However, a necessary constraint in many designs is to lock down certain integral pieces such that the optimization does not modify them unnecessarily. A position constraint locks a piece to its original position such that the algorithm will strive to solve the optimization without moving that piece. This can be useful to define load-bearing pieces or pieces designed to break other constraints. Silhouette.

As an example of a performative constraint, we consider pieces forming a set silhouette (see Fig. 8) when viewed or lit from a certain direction [5]. This soft constraint can be used for an application such as daylighting or can enable the designer to intuitively fill a section of a certain volume for semantic purposes. The constraint can also be prescribed to necessitate a certain shape or “skeleton” in order to maintain structural integrity. Consider two intersecting pieces A and B. The silhouette constraint is satisfied locally within the solver by translating A and B in opposite directions such that they maximally fill the space defined by the projected silhouette.

3.4 Movement

A piece can additionally be restrained to a certain volumetric region, disallowing extreme rotations, which can otherwise occur in some cases. Another possible type of movement constraint is to lock the rotation angle, disallowing movement by more than a certain angle away from the original normal of the piece.

The collection of all user specified constraints forms a complex design space that is difficult to manually navigate. Trying to satisfy one constraint may invalidate others. Optimization methods can deal with these complex constraint satisfaction problems effectively.

4 Implementation

Our design tool is implemented as a Rhinoceros 5 plugin using C#. It can be integrated into existing workflow with custom Rhinoceros commands and toolbar buttons, requiring little training. Rhino functions allow iterative building while optimizing for given hard constraints. The user can selectively remove and break constraints. More final modifications can be made as well: there exist functions to trim a planar piece before intersection with another piece, and to trim redundant parts of pieces based on a given silhouette (i.e. projecting the desired silhouette back on each piece and removing the resulting projection from all but one piece.) To prepare the output for printing, at any given intersection, slits are cut out of each piece at proper locations to enable construction. This is done naively, making sure that each piece can be slid into its partner, but the user must make sure that the construction can be assembled properly. We provide tools to aid in this process to identify where unbuildable cycles occur and either introduce an extra cut or rearrange the orientations of the slits according to user input. Finally, we perform rigid body simulations in Maya to check for stability of the final construct, as unwanted sliding in the slit direction can still occur.

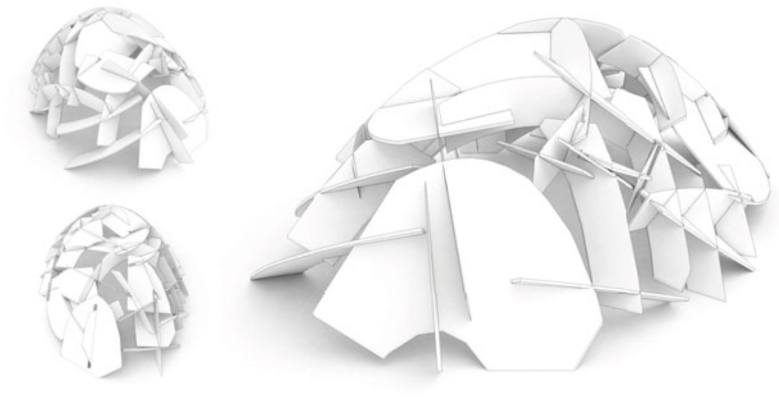

Fig. 7 A dome structure exhibiting many variations in shape and orientation.

5 Examples

Figs. 1, 7, and 8 present examples of the process showing the possibilities of utilizing computation to handle constraints and enable unexpected structures. Note the diversity of constructions allowed by these constraints and the use of optimization methods to help in the process.

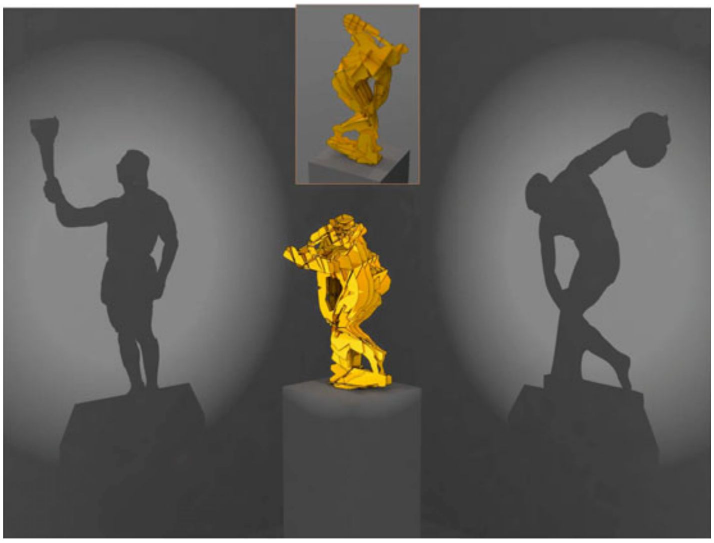

Fig. 8 A Shadow Art example (see [5]) with many more cycles. The example is constructed with silhouette constraints (the two shadows shown). In this case, the optimization is done as a post-rationalization step, and as there are many cycles, pieces tend to become more parallel.

6 Conclusions

As any project grows, the performance and construction constraints multiply. It is often impossible to foresee each issue from the initial stages of the design, and the complexity is such that the architect cannot account for every issue up front. This is a task that can be aided through the use of computation. A seemingly simple construction, orthogonally intersecting pieces, can quickly get complicated. We have presented an optimization-based design method that enables complex structures that are driven by their constraints while still allowing artistic freedom. As well as enabling a novel sculptural technique, since the results can be produced with a laser cutter, immediate applications of our method include prototyping and exploration of shapes, which can be of much use in the educational domain. Most importantly, by presenting a method of designing with the aid of optimization techniques to handle construction-aware constraints, we hope that this inspires further use of optimization during the design process rather than only as a post-rationalization step.

References

1. Bollinger, K., Grohmann, M., Tessmann, O.: The Sphere Project - Negotiate Geometrical Representations From Design to Production. In: Advances in Architectural Geometry, pp. 103–111 (2010)
2. Combettes, P.: Construction dun point fixe commun famille de contractions fermes. Comptes Rendus de lAcadmie des Sciences Srie I (1995)
3. Kilian, A., Nagakura, T.: Design exploration through bidirectional modeling of constraints. Massachusetts Institute of Technology, Cambridge (2006)
4. Mangelsdorf, W.: Structuring Strategies for Complex Geometries. Architectural Design 80(4), 40–45 (2010)
5. Mitra, N.J., Pauly, M.: Shadow art. ACM Transactions on Graphics 28(5), 156:1-156:7 (2009)
6. Oxman, N.: Get Real Towards Performance-Driven Computational Geometry. International Journal of Architectural Computing 5(4), 663–684 (2007)
7. Pottmann, H.: Architectural Geometry as Design Knowledge. Architectural Design 80(4), 72–77 (2010)