Automatic CAD-model Repair: Shell-Closure

Jan Helge Bøhn and Michael J. Wozny

Rensselaer Design Research Center
Rensselaer Polytechnic Institute
Troy, New York 12180–3590
(518) 276–6751
bohn@rdrc.rpi.edu

Abstract

Shell-closure is critical to the repair of CAD-models described in the .STL file-format, the *de facto* solid freeform fabrication industry-standard. Polyhedral CAD-models that do not exhibit shell-closure, i.e. have cracks, holes, or gaps, do not constitute valid solids and frequently cause problems during fabrication. This paper describes a solution for achieving shell-closure of polyhedral CAD-models. The solution accommodates non-manifold conditions, and guarantees orientable shells. There are several topologically ambiguous situations that might arise during the shell-closure process, and the solution applies intuitively pleasing heuristics in these cases.

1 Introduction

It is critical that the CAD-models used in solid freeform fabrication (SFF) are robust to avoid failure during manufacturing. Unfortunately, CAD-models described in the .STL file-format [1], the *de facto* industry standard, frequently lack this robustness [2, 3, 4]. Typical problems include punctured shells, inconsistent facet-orientations, and internal walls and structures.

With the .STL file-format, the CAD-model solid is defined by its surface, i.e. the solid is defined by a closed shell enclosing its material. If this shell is punctured (i.e. open), then it can no longer properly contain the material, and this consequently causes the failures commonly associated with cracks, gaps, and holes (Figure 1.1). Similar problems arise when the facet-orientations across a shell are inconsistent. This too causes uncertainty regarding what side of the shell contains material. Finally, structural problems might arise from internal walls and structures within the solid. These internal features can create irregularities in the solidified material during the processing. For instance, with stereolithography [5] and selective laser sintering (SLS) [6], excess amounts of unsolidified material might get trapped inside the solidified matrix, and consequently result in uncontrolled shrinkage and surface warping (Figure 1.2).

The .STL file-format is capable of representing a robust, well-formed solid. Unfortunately, contemporary CAD-systems frequently fail to achieve this goal. There are two

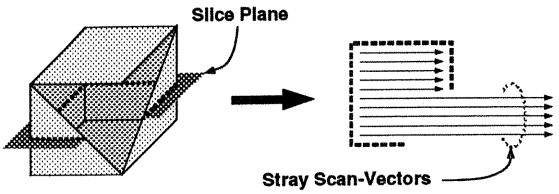

Figure 1.1 Shell-punctures causes the laser to produce stray scan-vectors.

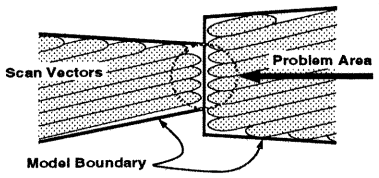

Figure 1.2 Internal walls cause structural discontinuities that can lead to excessive shrinkage.

design environments currently available for CAD; solid- and surface-modelers. Solid-modelers generally avoid punctured shells, but frequently generate needless internal walls and structures, and inconsistent orientations. In addition to these problems, surface-modelers frequently fail to properly mate adjacent surfaces; this results in punctured shells.

These problems are commonly resolved by the SFF-equipment operators prior to fabrication. The operator might attempt to edit the .STL file directly, or, if available, edit the original CAD-model to eliminate the problems. However, manual model repair becomes excessively expensive when the number of facets reaches 10,000 to 100,000+, and the number of errors run into the hundreds. This has lead to the development of automatic CAD-model repair.

Brock Rooney & Associates [7] perform limited CAD-model repair in their .STL translators, ensuring that their .STL output does not contain punctured shells. Their proprietary shell-closure solution is not complete. It does not handle orientability-problems satisfactorily, and it has limited success with certain simple structures. However, they report few problems in general with their .STL output when run on the software-robust stereolithography process of 3D Systems, Inc. In the rare cases where their shell-closure algorithms fails, the CAD-model must either be regenerated at the design-level, or be repaired manually as described above.

3D Systems, Inc. [8] provides its customers with a limited, proprietary model-repair facility. It too is intended to ensure that the model does not fail during fabrication, or if the repair fails, to provide indications to where it must be repaired manually. The program attempts to repair simple orientation problems, minor punctures, and facet-intersections, but leaves internal walls, and fails for large punctures and non-orientable shells. It performs the repair under the restriction of several user-specified parameters, and if these are set too conservatively, an intermediate result is generated. This output can be processed iteratively until a valid, though not necessarily correct result occurs.

The basic problem of CAD-model repair is that of inferring information that has been lost somewhere in the design process. While it might be easy in some cases to detect the presence of a problem, it is frequently impossible to determine with certainty what the correct solution is. This is especially true if the designer's intent is lost. Without it, an educated guess must be made whenever an ambiguity is encountered. The goal of our work is to match the quality of manual repair that is performed without the knowledge of the designer's intent.

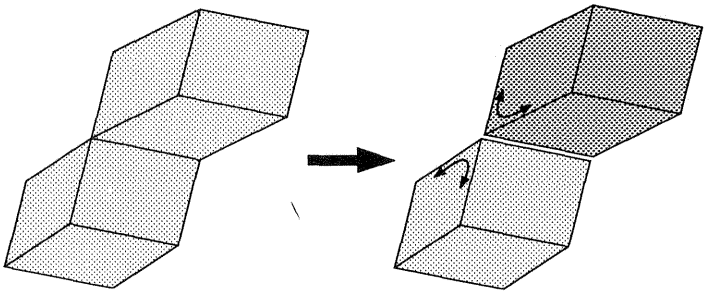

Figure 2.1 A non-manifold condition can be viewed locally as a set of two-manifold conditions.

2 Shell-Closure

This paper presents a complete solution for shell-closure of polyhedral shells. In doing so, it also solves the orientability problems, and it simplifies the complexity of the model-topology for more efficient processing during fabrication. In particular, it ensures a pseudo two-manifold topological organization of the facets to facilitate high-speed slicing [4, 9]. This topological organization also simplifies the subsequent resolution of shell-intersection and shell-nesting problems, which are necessary for the removal of needless internal walls.

A (two-) manifold structure is homeomorphic to a two-dimensional disk at every point across its surface, i.e., each such point has a neighborhood that is topologically equivalent to a two-dimensional disk. A non-manifold structure violates this constraint; two cubes that touch along a common edge is one such example. However, if the facets at a non-manifold location are paired up properly, then this situation can be described as a set of (pseudo) two-manifold conditions [10] (Figure 2.1).

Our solution for shell-closure follows a three-stage strategy: First, we identify the lamina edges. Second, we combined these edges into closed, non-self-intersecting loops of boundary-curves; commonly known as directed Jordan curves. And finally, we create lids that fit these Jordan curves and that close the open shells.

3 Lamina Edges

A lamina edge is generally considered a special case of a non-manifold edge; it has only a single adjacent face [11]. Lamina edges are of interest because they make up the boundaries of the shell-punctures, and of the corresponding lids which we will create to close the shells. However, if the definition of a lamina edge is generalized, in particular generalizing the definition of a puncture, then we can expand the use of creating lids for shell-closure to also solve erroneous facet-orientations and complicated non-manifold conditions.

Expanded definition: A lamina edge exists along each face-edge that does not mate and pair properly this particular face with any other face along this edge. These lamina edges can be found as follows: Let all face-edges be candidates for lamina edges. Then, for each face-edge, attempt to find an adjacent face of compatible orientation such that

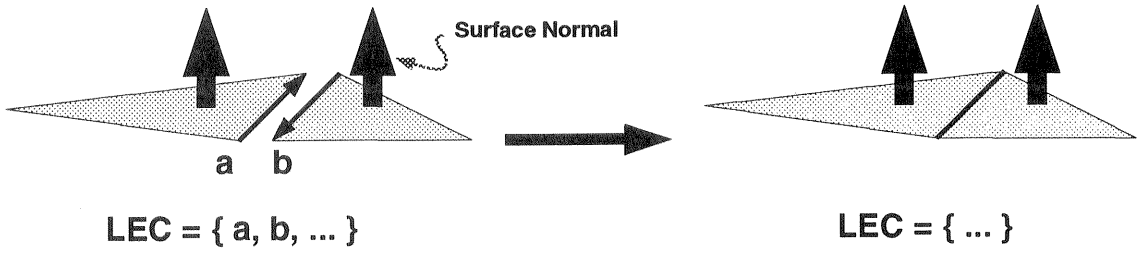

Figure 3.1 Two lamina edge candidates (a,b) are removed from the set of candidates (LEC) after a successful pairing of two faces about a common edge.

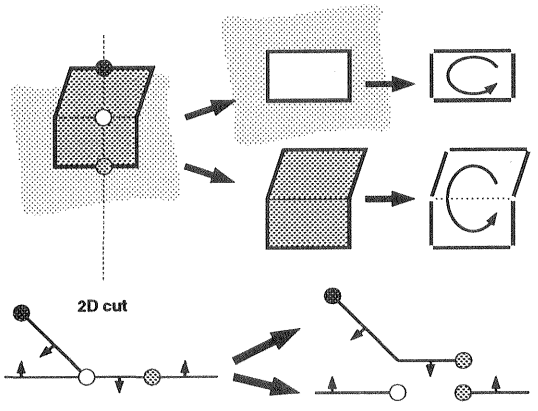

Figure 3.2 The orientability problem is solved by creating two sets of (partially) coincident lamina edges.

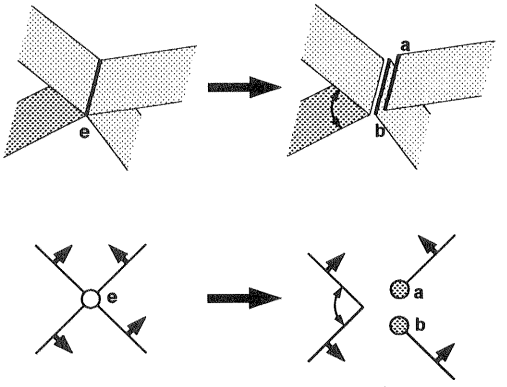

Figure 3.3 Two unpaired facets along a non-manifold edge yield two coincident lamina edges.

these two faces can be paired off and identified as adjacent about the common edge. Whenever this happens, the two coincident face-edges are removed from the set of lamina edge candidates (Figure 3.1). The face-edges that remain after this process is exhausted, are the lamina edges we seek.

The expanded definition of lamina edges allows for the possibility of multiple, coinciding lamina edges along a single physical edge; this simply represents multiple shell-puncture boundaries partially or fully coinciding along this particular edge. In particular, Figures 3.2 and 3.3 illustrate the effect of this definition on areas with orientation and non-manifold problems. Note how allowing multiple coinciding lamina edges ensures that the boundary of a shell-puncture always forms a closed, directed non-self-intersecting loop, or more explicitly, a directed Jordan curve. These Jordan curves are essential for the subsequent lid-creation and shell-closure.

4 Directed Jordan Curves

The first step towards extracting directed Jordan curves from the set of lamina edges, is to link the lamina edges into a set of non-intersecting boundary curve-segments, or more precisely, arcs. Each arc is directional, is adjacent to a single shell, and is terminated

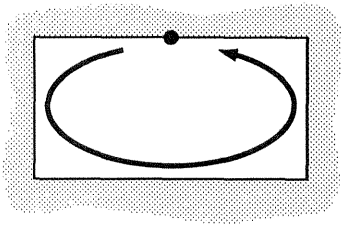

Figure 4.1 Simple, self-connecting loop (unambiguous).

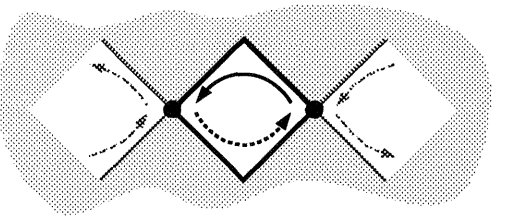

Figure 4.2 Linear combination (unambiguous).

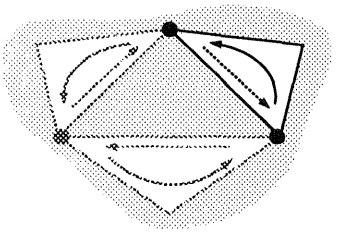

Figure 4.3 Cyclic with single direct returns.

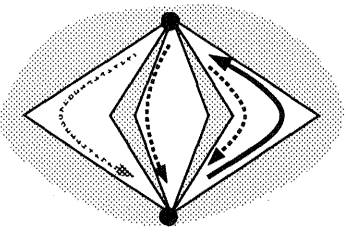

Figure 4.4 Cyclic with multiple direct returns.

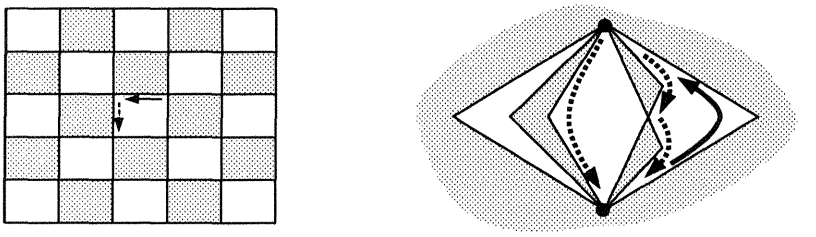

Figure 4.5 Trivial and nontrivial cases of cyclic with non-direct returns.

by two endpoints that might coincide. This can be achieved as follows: Initially, consider each lamina edge as an arc. Then, iteratively, for each vertex with only two adjacent arcs, merge those two arcs into a single arc, causing this vertex to no longer have an adjacent arc. The final set of arcs form a set of directed graphs (as defined in graph theory [12]) of various complexities that describe the shell-puncture boundaries, and these puncture-boundaries must be separated prior to lid-creation.

We have identified five classes of puncture-boundaries of increasing identification-complexity, which when combined, form the above mentioned sets of directed graphs. Among these classes, three occur in our industrial .STL test-suite, while the remaining two, the most complex classes, persist as unobserved possibilities: The most trivial class of all is the *simple* combination; a single self-connecting arc. It is unambiguous and trivial to detect, and it forms a complete puncture-boundary by itself (Figure 4.1).

The *linear* combination is also unambiguous; two arcs that are adjacent to the same shell and that form a closed loop. These arcs form a link between two separate subgraphs, each of which is less complex that their combination (Figure 4.2).

The remaining three classes are ambiguous because they offer multiple closed-loop alternatives, and because the Jordan curves are located in three-dimensional space rather than in a simplifying two-dimensional plane. Fortunately, one can make some reasonable,

basic assumptions that limit the number of loop-completion alternatives: For instance, we assume that, whenever it is an option, we should merge separate shells. This is based on the observation that the .STL file-format separates solids; consequently, all facets within a definition for a single solid should form a single shell. This dictates that, for the following three classes, two consecutive arcs may not be adjacent to the same shell prior to merging.

The class *cyclic with single direct return* is characterized by the formation of cyclic patterns by its punctures, and by the existence of a single returning arc for each of its arcs, once the option of connecting arcs along the same shell has been eliminated as described above (Figure 4.3). We have found this class to be rare, but present, in our industrial .STL test-suite.

The remaining two classes are theoretically feasible, though we have not yet encountered them: *Cyclic with multiple direct returns* is an inherently ambiguous problem (Figure 4.4); even after the elimination of one returning arc, multiple options remain. It must consequently be solved with additional heuristics, utilizing not only topological-, but also geometrical information. We choose to view it as a matching problem where the goal is to minimize the sum of the angles formed by the end-tangents of the connecting arcs. This heuristic finds the optimal choice in the planar case, and it has a graceful degradation as the situation becomes less planar and consequently more ambiguous. Furthermore, this problem can be solved as a *perfect matching with minimum weight* problem in $O(n!)$ time; specifically, in the case of two, three, and four arc-pairs, we end up with one, two, and six matching-combinations respectively: Relating back to the example in Figure 4.4, we observe the presence of six arcs (two upward arcs are not shown), yielding three arc-pairs. Once an arc is selected (solid black upward arc), the above shell-merger assumption eliminates one of the returning arcs (dotted gray downward arc) as an acceptable loop-forming match, leaving the other two returning arcs as possible matches to contend with. The selection of either returning arc will cause the merger of two shells, thus simplifying the remaining problem to the point where further loop-extraction is unambiguous. As a consequence, the overall problem has two possible combinations.

Finally, there is the class of *cyclic with non-direct returns*; these are combinations that require a sequence of more than two arcs to form a closed loop (Figure 4.5). Some of these cases can be solved trivially by incrementally merging arcs into longer parts, but one can also easily imagine cases that are more complex and ambiguous. One possible solution in the later case, is to project the arcs onto a suitable two-manifold structure, e.g. a plane, and resolve the matching-problem there.

More effort could clearly be invested in the loop-extraction from the more complex arc-combinations, but based on our industrial .STL test-suite, these problems appear to be few and far between. Instead, the central theme of our strategy is to identify the less complicated structures first, and extract their arcs from the original graph. This simplifies the remaining graph and will in general reveal additional unambiguous constructions. As a result, one can solve most complex graphs by solving trivial sub-problems.

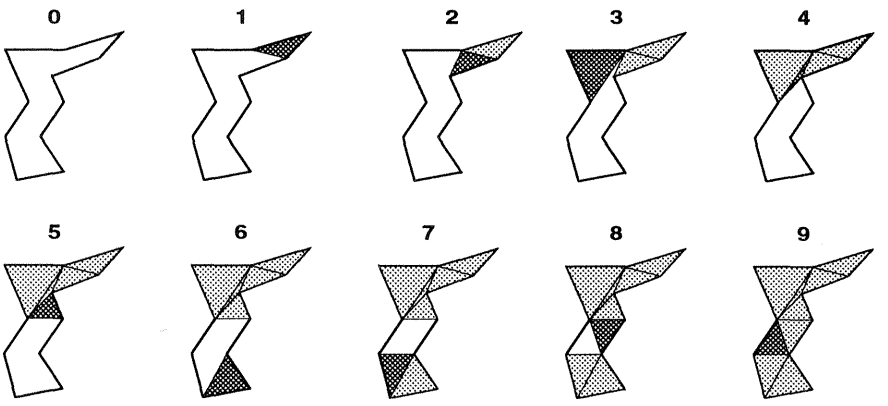

Figure 5.1 Lid-creation by iterative removal of the vertex with the smallest angle.

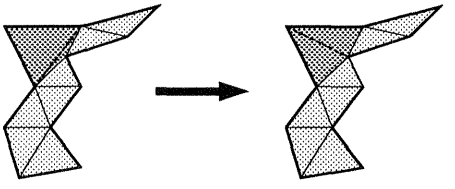

Figure 5.2 Iterative post-processing to improve the quality of the lid-surface.

5 Creating Lids

The task of creating lids is greatly simplified once the corresponding directed Jordan curves have been identified: The puncture is closed by an iterative algorithm that decrements the Jordan curve by eliminating one vertex at a time; the vertex is eliminated from further consideration by adding a facet that spans it and its two neighboring vertices along the shrinking Jordan curve. At each iteration, we remove the vertex along the Jordan curve that spans the smallest angle with its two neighboring vertices along the remaining, decremented Jordan curve (Figure 5.1). This avoids the clearly undesirable splitting of angles, and it will in general produce excellent results, especially for long narrow punctures which abound in industrial .STL files [7, 8].

However, this heuristic algorithm is not without drawbacks. It is a time-local algorithm in that it finds the best selection at any given time, without keeping the overall optimal solution in mind. An example of this is the selection of the third facet in Figure 5.1; it forces the subsequent undesirable creation of facet number four. Fortunately, such problems are easily rectified with a simple, inexpensive iterative post-processing step as illustrated in Figure 5.2: For each adjacent pair of lid-facets, flip their common diagonal edge if this increases their minimum facet-angle.

A more serious problem is the difficulty of finding the angles spanned about the vertices. While angles between vectors in 3D-space are in the range 0–180 degrees, it is clear, from the 2D-plane, that the angles between facet-edges are in the range 0–360 degrees. It turns out, however, that a conservative decision to assign the range 180–360 degrees to a vertex whenever there is a doubt to which half of the total range it belongs to, simply postpones its elimination. Eventually, one of the vertex's neighboring vertices

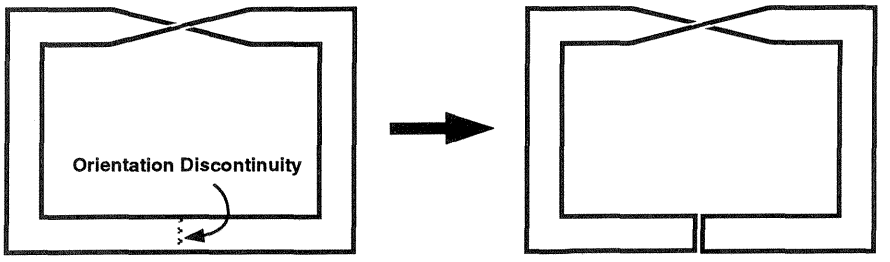

Figure 5.3 Cutting a non-orientable Möbius-strip along its orientation-discontinuity, results in a orientable panel.

will be eliminated, and its angles must be recomputed. Usually, this clarifies the situation, and the vertex might be assigned the range 0–180 degrees instead.

It is interesting to note that the above lid-creation algorithm is capable of creating a non-orientable shell if the open shell is non-orientable. An example of this would be the closure of a Möbius-strip. However, with our definition of lamina edges, non-orientable shells are cut such that they become orientable prior to lid-creation. In particular, the Möbius-strip becomes a simple panel (Figure 5.3). As a consequence, since the lid-creation algorithm maintains the directionality of the diminishing Jordan curves, we are guaranteed to maintain orientability throughout the shell-closure; and more importantly, the final closed shells are therefore always orientable.

6 Conclusions

This paper has presented a complete solution for shell-closure of polyhedral shells, the first step towards well-formed and manufacturable solids. The solution is based on topological principals, and consequently resolves orientability and non-manifold problems in addition to eliminating shell-punctures (i.e. cracks, holes, and gaps). As a result, the facet-model output consists entirely of closed, orientable shells, and the resulting facet-models can therefore be viewed as a set of two-manifold shells.

This topological organization simplifies the subsequent resolution of shell-intersection and shell-nesting problems, and it is a prerequisite for the detection and removal of zero-volume parts and other needless internal walls that create structural discontinuities in the fabricated parts.

7 Acknowledgments

This paper is taken in part from a thesis to be submitted in partial fulfillment for the degree of Doctor of Philosophy in the Department of Electrical, Computer, and Systems Engineering at Rensselaer Polytechnic Institute.

This research was supported by NSF Grant DDM-8914212 as a subcontract through the University of Texas Solid Freeform Fabrication program, and other grants of the Rensselaer Design Research (RDRC) Industrial Associates Program. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of

the authors and do not necessarily reflect the views of the National Science Foundation or any of the industrial sponsors.

We would like to thank Dick Aubin, Pratt & Whitney (a division of United Technologies), for providing a number of industrial .STL models. Also, special thanks to the members of the RDRC: Stephen Rock for his early work on this project and his subsequent collaboration; James Miller and Robert O'Bara for their development of facet-model visualization tools; and Peter Wilson for his helpful review of this paper.

References

[1] "Stereolithography Interface Specification," 3D Systems, Inc., June 1988.

[2] S. J. Rock and M. J. Wozny, "A Flexible File Format for Solid Freeform Fabrication," in: *Solid Freeform Fabrication Symposium Proceedings*, H. L. Marcus, J. J. Beaman, J. W. Barlow, D. L. Bourell, and R. H. Crawford (eds.), The University of Texas at Austin (Austin, TX), August 12-14 1991, pp. 1–12.

[3] J. Arline, "A user survey of problems with rapid prototyping systems," *Prototyping Report*, vol. 1, no. 6, November 1991, pp. 7–8.

[4] S. J. Rock, "Solid Freeform Fabrication and CAD System Interfacing," Master's thesis, Rensselaer Polytechnic Institute, Troy, NY, December 1991.

[5] R. A. Schubert, "StereoLithography," in: *NCGA'89 Conference Proceedings*, vol. 2, (Philadelphia, PA), April 17-20, 1989, pp. 182–186.

[6] C. R. Deckard, *Selective Laser Sintering (CAD/CAM)*. Ph.D. thesis, The University of Texas at Austin, 1989.

[7] B. Rooney. Private communication, Brock Rooney & Associates, Birmingham, MI, May 27, 1992.

[8] C. Alexander. Private communication, 3D Systems, Inc., Valencia, CA, May 29, 1992.

[9] S. J. Rock and M. J. Wozny, "Utilizing Topological Information to Increase Scan Vector Generation Efficiency," in: *Solid Freeform Fabrication Symposium Proceedings*, H. L. Marcus, J. J. Beaman, J. W. Barlow, D. L. Bourell, and R. H. Crawford (eds.), The University of Texas at Austin (Austin, TX), August 12-14 1991, pp. 28–36.

[10] M. Mäntylä, "Boolean Operations of 2-Manifolds through Vertex Neighborhood Classification," *ACM Transactions on Graphics*, vol. 5, no. 1, January 1986, pp. 1–29.

[11] K. J. Weiler, *Topological Structures for Geometric Modeling*. Ph.D. thesis, Rensselaer Polytechnic Institute, Troy, NY, August 1986. (RDRC-TR 86032).

[12] F. S. Roberts, *Applied Combinatorics*. Prentice-Hall, 1984.