To appear in Proceedings of Graphics Interface 2007

Constrained Planar Remeshing for Architecture

Barbara Cutler
Department of Computer Science
Rensselaer Polytechnic Institute

Emily Whiting
Departments of Architecture and Computer Science
Massachusetts Institute of Technology

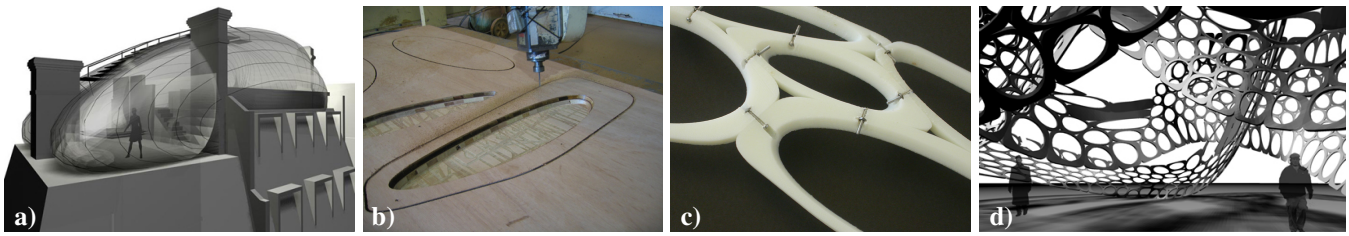


Figure 1: Beginning with a) an architect's complex curved design for a rooftop greenhouse, a planar remeshing of the surface is generated using the techniques described in this paper. Using automated milling equipment, b) the panels can be fabricated out of planar plywood sheets and c) assembled with simple hardware fasteners to d) create a unique and inspiring outdoor sculpture.

ABSTRACT

Material limitations and fabrication costs generally run at odds with the creativity of architectural design, producing a wealth of challenging computational geometry problems. We have developed an algorithm for solving an important class of fabrication constraints: those associated with planar construction materials such as glass or plywood.

Starting with a complex curved input shape, defined as a NURBS or subdivision surface, we use an iterative clustering method to remesh the surface into planar panels following a cost function that is adjusted by the designer. We solved several challenging connectivity issues to ensure that the topology of the resulting mesh matches that of the input surface.

The algorithm described in this paper has been implemented and developed in conjunction with an architectural design seminar. How the participants incorporated this tool into their design process was considered. Their important feedback led to key algorithmic and implementation insights as well as many exciting ideas for future exploration. This prototype tool has potential to impact not only architectural design, but also the engineering for general fabrication problems.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

Keywords: Curve, surface, solid, and object representations; Geometric algorithms, languages, and systems

1 INTRODUCTION

Physical limitations of construction materials and fabrication constraints can place undesirable restrictions on the freedom of architectural design. Usually the basic building materials are selected early in an architectural design process and can have significant impact on the shapes and styles that may be considered.

One particularly challenging construction material is glass. Although glass can be bent into curved panels (e.g., an automobile windshield or a grocery display case), flat planar sheets are more cost effective to manufacture [2]. This essentially limits all-glass construction to planar meshes. Furthermore, as a brittle material, glass is also susceptible to cracking when cut at extreme acute angles. Glass atriums, greenhouses, and other *curtain wall* buildings are typically made in simple, symmetric, planar shapes so these constraints can be satisfied by hand with pen and paper using rectangular panels. Examples of these shapes include the classical groin vault (the intersection of two cylindrical vaults) and the spherical lens shown in Figure 2a&b. Stephan et al. [21] describe a special subclass of surfaces that are common in architectural design that may be fit with a regular quadrilateral network through a series of *homothetic and dilative translation* transformations.

In general, any arbitrary surface can be represented by a triangle mesh and thus constructed from triangular panes of glass. One example of a *regular triangular mesh* applied to a gently curved surface is the modern roof of the British Museum in London (Figure 2c). This mesh can be simply parameterized by projecting it to a horizontal plane and is tiled with a valence-six, regular triangulation of near-equilateral triangles. Similar techniques were used in the structures shown in Figure 2d&e.

For more complex surfaces where a minimal distortion parameterization is not possible, *an irregular triangulation* can be used. This approach has created fantastic structures, such as the Fiera Milano (Figure 2f). This work by Jörg Schlaich uses the flexibility of an irregular network not only to allow arbitrary geometry, but also to define the building's structure. The edges between the triangular facets have been carefully arranged to form solid continuous ribs that carry the forces from the roof down to the foundation [10, 19].

Although current meshing techniques can produce visually striking results, designers are still limited by the basic triangular (or quadrilateral) primitive used in faceting. Creative opportunities still exist in the exploration of new patterns and shape variations. We provide a tool to enable such explorations.

1.1 Structural and Fabrication Considerations

Beyond an aesthetic desire to break from a triangular network, there are several disadvantages from the structural analysis and fabrication perspective. To analyze the forces carried by a network of struts and pinned joint connections, an engineer writes a force equation requiring the forces at each node to sum to zero [26]. Each strut carries an unknown tensile or compressive force to the node. If

Figure 2: Examples of several stunning architectural designs for curved forms from glass: a) Chadstone Shopping Center; Melbourne, Australia; RTKL Associates Inc, 1999. b) Lens Ceiling, U.S. Courthouse; Phoenix, Arizona; James Carpenter Design Associates, 2000. c) The Great Court; London, England; Norman Foster and Partners, 2000. d) Swiss Re Building (Gherkin); London, England; Normal Foster and Partners, 2004. e) DG Bank Building; Berlin, Germany; Gehry Partners and Schlaich, Bergermann and Partner, 2000. f) Fiera Milano; Milan, Italy; Schlaich, Bergermann, and Partner, 2005.

more than three struts meet at a joint the problem becomes over-determined and the actual forces cannot be computed. Analogously, a three-legged stool is stable, but a four-legged stool will wobble with the slightest manufacturing imprecision. If one leg is shortened by just a millimeter the amount of weight it carries will go from one quarter of the object weight to zero. To ensure a stable structural analysis that matches the forces in the constructed pinned frame, it is advantageous to have nodes with exactly three struts; i.e., a valence-three mesh.

If a thick construction material such as plywood is used, designing the connection between panels of material is challenging. The edges of the panels will be chamfered so they form the appropriate angle when joined. The exact chamfer angle is defined per edge as one half the angle between the pair of panels. We can compute the vertices of the *interior* face of each panel by intersecting the planes parallel to the panels but offset inward by the thickness of the material. If four or more panels meet at a node in our remeshed model we must examine how the offset planes intersect. If not constrained to be a *conical mesh* [15], these planes will not intersect at a single interior point, and thus the panels will require additional chamfering as seen in Figure 3. This extra chamfering is expensive and aesthetically undesirable.

Thus it is often preferable to have joints with exactly three panels. Depending on the specific material chosen some of these issues are less important or irrelevant. In practice, fabrication and structural considerations are not tightly coupled with architectural design. We have been working closely with a structural engineering firm to finalize the details for a sculpture designed with this system. We will learn from this analysis and incorporate these constraints in the system as part of a tighter feedback loop in future work.

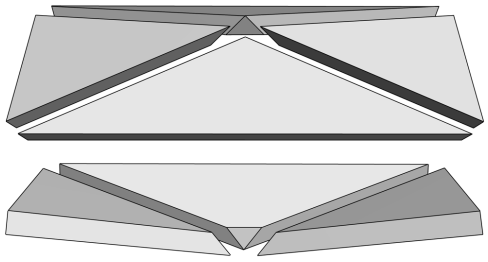


Figure 3: In the top image we show a detail where four panels meet crisply at a single exterior vertex. The panels have been pulled apart slightly to show the chamfer detail. The bottom image is shown from the inside of the model and one of the panels has been removed for clarity. Because the inwardly offset planes do not intersect at a single point, additional chamfering (of the center panel) is necessary.

1.2 Voronoi Cell Remeshing

An example of a prototype valence-three structural system is shown in Figure 4. In this architectural student design project, a complex curved surface was first texture-mapped with a pattern of Voronoi cells. Voronoi cells are attractive as a design tool because they typically result in meshes where exactly three facets meet at a joint *and* the angles within each cell are greater than 90°. A modular framework consisting of struts and three-way joint connectors was then applied to the corners and edges of the cells. The construction system was completed by stretching a rubber sheet over the network of struts and joints.

Because the Voronoi cells are not constrained to be planar, the lingering question from the student's project was how the structural system could be extended for use with a rigid planar infill material, such as glass or plywood. The student needed a modeling tool that could solve for vertex positions that resulted in planar facets. It was uncertain whether a simple post-process of the geometry could move the vertices such that the facets were planar. These very questions motivate the research described in this paper.

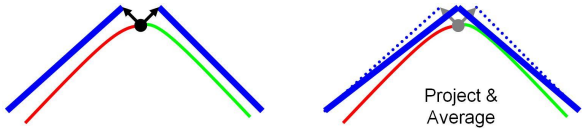
1.3 Related Graphics and Geometry Research

The problem this paper addresses is inherently a task of remeshing, a well-studied area of computational geometry. For example, to construct uniform triangulations similar to those in Figure 2c&f

Figure 4: Images of the scale model built by seminar participant and architecture student Mike Powell as a prototype structural system. In this early model the facets are not planar, and a flexible rubber sheet is necessary to skin the strut and joint framing system.

the work by Turk [23], Alliez et al. [1], and Surazhsky et al. [22] (among others) would be very applicable.

The work of Cohen-Steiner et al. [6] was particularly motivating to us, and on first glance is the exact tool to solve the non-planarity issues that challenged the project described in the previous section. In their work a pre-specified number of face clusters are fit to the original surface geometry using Lloyd's relaxation (a k-means clustering technique). Then planar proxies are fit to these clusters and connected to create a polygonal mesh. This work is effective for simplification and working with polygons rather than triangles reduces the number of vertices that must be sent to the graphics hardware. Unfortunately, the final polygon vertices are computed by *averaging* the projection of an original mesh vertex onto each proxy as illustrated below and in general the final vertex position does not lie on the proxy planes. Thus, facets which have more than three vertices will likely be non-planar.



Furthermore, for any but the simplest shapes the vertices from this remeshing cannot simply be moved such that the all facets are planar. Attempting to do so for non-trivial inputs will require vertex and neighborhood shuffling (discussed in Sections 2.2 & 2.3 and illustrated in Figures 6 & 7). Furthermore the positions cannot be trivially optimized to be planar since the solution space is highly discontinuous. The neighborhood of a particular facet is not stable from iteration to iteration preventing the incorporation of optimization terms to constrain the vertices appropriately.

Alternatively, we could view this planar remeshing task as an incremental simplification problem and attempt to use an edge or face collapse technique [8, 11, 12, 20] to incrementally converge on a polygonal model. This strategy has the advantage of creating a continuum of legally constructable intermediate meshes, but may quickly become stuck in a local minimum due to complications with the polygonal planarity requirement. We plan to explore this avenue in future work.

The planar quadrilateral meshing work of Liu et al. [15] shares many challenges with our project. Their solution involves determining an appropriate local parameterization along which they align a regular quad mesh. Their mesh is further constrained to also fit an offset surface suitable for fabrication (avoiding the chamfer problem discussed earlier). Furthermore, their work extends nicely to generate developable surfaces which are of significant interest for architectural applications.

Other research projects have tackled various fabrication challenges. Chen et al. [5], Mitani et al. [16], and Haeberli [9] examine how to subdivide a target shape into strips or patches that are well approximated by a *developable surface*, which can be formed from a material such as sheet metal or paper that allows bending but not stretching. Branco and Soares [3] modify a developable surface by cutting and separating or overlapping the material (similar to a tailor's darts) to build surfaces with double curvature. Julius et al. [13] generate quasi-developable patches within a specified tolerance for applications where minor stretching is allowed. Wu and Kobbelt [25] further generalize the cluster based remeshing strategy to fit portions of the model to spheres and cylinders. A complex blend operation is required to fuse the different elements of the model together.

The area of dimensionality reduction, and specifically the Locally Linear Embedding framework [4, 18], contains interesting parallels to our project. The work of Kirasanov and Gortler [14] can be used to obtain a piecewise planar approximation of an arbitrary mesh, but the memory usage and performance could be prohibitively high. Furthermore, the output of these techniques are irregular triangular meshes that do not meet the requirements discussed earlier and lack the user control that is necessary for applications in architectural design.

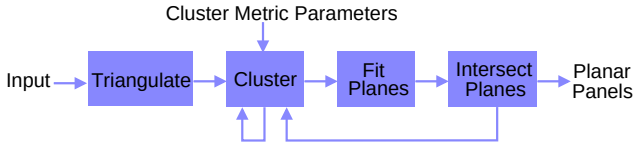
1.4 Overview

The contributions of this paper are:

- An algorithm to fit planar polygonal panels to arbitrary input geometry;
- A user-controllable metric for clustering the surface that controls the distribution and shape of the resulting panels;
- Control of relative panel shape and density through a paint brush metaphor or procedural scripting;
- Consideration of important structural and fabrication issues including offset planes for chamfering; and
- A full-featured prototype system implementation that was used in an architectural design seminar and integrated into a fabrication pipeline.

2 Planar Remeshing Algorithm

The reluctant acceptance of non-planar facets by the architects for the project described in Section 1.2 emphasizes the challenges of fabrication remeshing. In general it is not possible to simply adjust the facet vertices in a post-process. We illustrate some of these difficulties in images throughout this paper. Instead, we approach the problem by choosing planes and place vertices where the planes intersect. The initial stages of our method closely follow that of Cohen-Steiner et al. [6] to select a good distribution of planes using the randomized optimization method. Here is the basic pipeline of the system:



Initially we choose n random seed triangles on the surface of the mesh, where n is the target number of planes specified by the user. Then we divide the remaining triangles in the model into clusters around these seeds, ensuring that the triangles in each cluster are contiguous. The clustering step is iterated by computing a new center/seed for each cluster and then re-clustering. For an initial configuration of well-distributed random seeds, ten to twenty iterations is generally sufficient for convergence and to evenly distribute the triangles across the surface and into clusters of similar area (Figure 5).

In the following sections we describe our contributions; in particular, how to intersect the resulting proxy planes and form a closed model. If the intersection fails, it is necessary to pipe this information back into the clustering stage and try again.

2.1 Determining the Local Neighborhood

Once the clustering has reached a fixed point (or converged sufficiently), we must intersect the planes with their neighbors to find a planar polygonal remeshing. For convex shapes we could use any standard $O(n \log n)$ algorithm to compute the half space intersection [7]. Since the neighborhood information is available in our data structure, we can actually do this in $O(n)$ time by walking the boundary of each cluster and intersecting it with the proxy planes corresponding to neighboring clusters. This method works for both

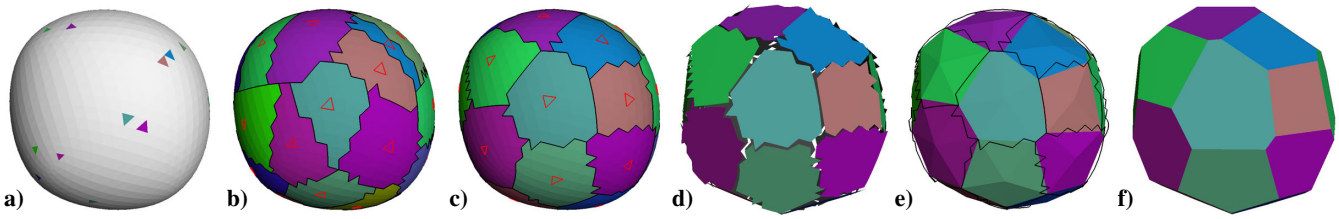


Figure 5: The basic algorithm begins by a) selecting random seed triangles from the original mesh. Next the mesh is b&c) iteratively clustered about these seeds and the seeds are repositioned; d) shows a visualization of the original mesh triangles projected onto the corresponding proxy planes; e) using the vertices and neighbors from the clusters we render non-planar polygons, similar to Cohen-Steiner et al. [6]; and f) the proper intersection of the neighboring planes.

convex and non-convex objects. In convex portions of the model the result is identical to a half-space intersection. In areas of negative curvature, the concept of a cluster center and the local neighborhood facilitates the construction of a corresponding polygonal remeshing.

However, since the cluster neighborhoods are a function of both the underlying triangular mesh and the cluster metric, this method is not robust and often requires some cleanup. This is due to the fact that the set of cluster neighbors is not necessarily the same as the set of neighbors that define consistently intersected panels. Figure 6 shows an example where the polygonal boundary of a facet self-intersects, we call this geometric error a *flipped edge*.

2.2 Complex Curvature and Panel Shape

Trying the basic algorithm on non-convex target surface shapes uncovers a key challenge in this fabrication problem. To mesh the neighborhood at a saddle point (or generally, areas with *negative curvature*) we must either a) allow concave panels or b) have more than three panels meeting at some vertices. A simple non-convex shape is shown in Figure 8. Depending on the material to be used for construction, one of these two solutions may be more desirable. For example, it is impractical to make concave inner corner cuts in glass; thus, non-convex facets must be split at the concave corners.

If we choose to allow four or more planes at these vertices, we must ensure that all planes intersect at a single point (within the required construction tolerance). Furthermore, we will encounter an interesting offset problem in fabrication. The panels will have non-zero thickness and we need to make sure that not only do the four planes intersect at the single surface vertex, but that the four offset planes also intersect cleanly. If this offset problem is not solved, the construction process will require extra chamfering and

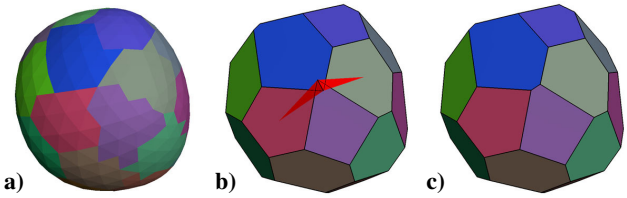


Figure 6: In the initial clustering of this object, a) four patches meet at a single original mesh vertex. The proxy intersection routine randomly chooses how to structure the local neighborhood to define vertices at the intersection of three planes. b) In this example, the random choice is incorrect, causing self-intersection of the polygon boundaries for the pink and gray facets. This is detected as a mis-oriented or flipped edge and visualized with a red triangle. c) Once detected, this problem is easily fixed by switching the neighbor assignments.

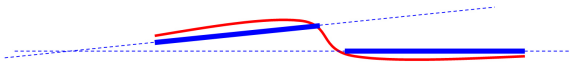
be more complicated and expensive (Figure 3).

In our system we allow concave facets. This decision made the implementation of various tasks from rendering to area computation far more challenging. We do restrict all panels to be star-shaped (that is, have a *non-null kernel*). By comparing the orientation of each facet edge relative to a point in the kernel, we can quickly detect geometric anomalies such as flipped edges (Figure 6). We have not found this restriction to limit the surfaces we are able to remesh with planar panels. Furthermore, with the implementation of alternate methods for rendering and detection of flipped edges this restriction can be lifted to allow general concave facets.

2.3 Challenging Proxy Plane Configurations

The largest challenge for the algorithm is that not every clustering of the original triangles will lead to a legal planar remeshing. In some cases the problem is simply that too few clusters were specified by the user to solve the target geometry. For example, it is impossible to intersect three planes to achieve a closed remeshing that approximates a genus zero shape. Some of these cases could be resolved by analyzing the surface (or a portion of the surface) to determine the minimum number of panels necessary to match the topology and genus of the input shape. However, this will not handle all cases. Consider the configuration below in cross section. The red curved input shape has been approximated by three blue planar proxy surfaces that happen to be parallel. Since these planes do not intersect, we cannot use them to find a planar remeshing.

Also consider the more general case below where the planes do intersect, but not *where they are supposed to intersect*. The intersection is on the wrong side of the patch and though the algorithm can successfully compute vertices for each facet, the facets will have many flipped edges and will be rendered inconsistently.



We detect this "spike" problem vertex (Figure 7b) by computing the distance between the cluster boundary vertex and the planar intersection vertex. An intersection vertex is flagged if this distance is greater than a reasonable threshold (we use 0.2 * the average cluster diameter). The algorithm attempts to resolve these problems with two different techniques.

First, the three proxy planes defining a flagged vertex are *tilted* to bring the intersection closer to the cluster boundary vertex. Typically, the normal of a planar proxy is computed for a collection of triangles as the eigenvector of the smallest eigenvalue (L^2) OR the average of the triangle normals weighted by area ($L^{2,1}$) [6]. For

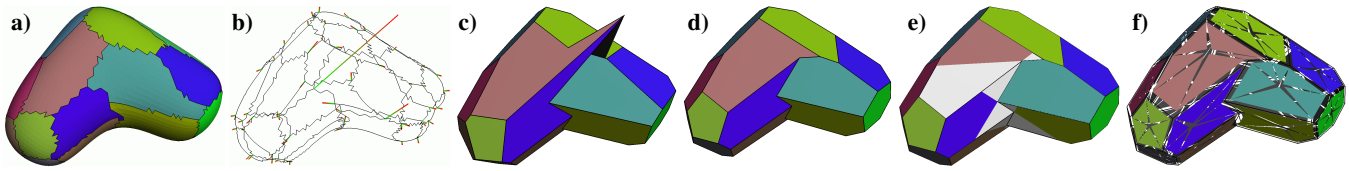


Figure 7: One way to detect potential problems in the a) triangle clustering is to b) study the difference vector from the vertex where three or more clusters meet (green) to the vertex found when the corresponding proxy planes are intersected (red). When this vector is relatively large, we may have c) difficulty resolving this particular set of planes into a topologically consistent mesh. This information is used to refine the solution and create d) a constructable planar remeshing. To simplify rendering we require that all panels be star-shaped, i.e., that they have e) a non-null *kernel*. Any point within the kernel can be used to f) draw the star-shaped panels and easily detect flipped edges (Figure 6).

planes surrounding a flagged vertex we average this result with the plane that passes through the proxy center and the cluster boundary vertex. We do not include this term for non-spike intersection vertices because it could inappropriately bias the solution.

Second, we shuffle the planes (Cohen-Steiner et al. [6] refer to this as teleportation). We split a plane (seed a new cluster) near the problem area and merge two planes elsewhere in the model where perhaps the extra detail is less important. To ensure that the new cluster does not float away immediately, we flag the new cluster seed and its immediate neighbors as *sticky* and restrict its movement for several iterations. Figure 9 shows examples of challenging surfaces which require these techniques.

The detection and correction of these problems bound the error of our planar approximation, although we have not formalized the error bounds.

2.4 Multiple Clusters per Proxy Plane

Occasionally the optimal partitioning by a particular cluster metric suggests an impossible facet neighborhood. While the clustering method ensures contiguous clusters, it does not guarantee that a cluster is not completely surrounded by another cluster, or that a cluster does not have a hole, etc. Examples of neighborhoods that cannot be resolved are shown in Figure 10. These issues cannot always be corrected simply by adjusting or changing the cluster metric. It is necessary to automatically detect these conditions and break the cluster into separate pieces. As above, we make this change sticky to prevent it from being un-done immediately.

To prevent numerical issues when computing the intersection of nearly coplanar proxies, neighboring clusters whose normal differs by less than an angular threshold (we use 1°) will be snapped to the same *master plane*, as shown in Figure 11.

2.5 Boundary Conditions

Our prototype implementation also handles non-closed shapes, as shown in many of the examples in this paper. Normally, a panel vertex is defined where three clusters intersect. The boundary is treated as a special NULL cluster object. Thus, where two clusters meet at the boundary, we define a panel vertex. This point is placed on the intersection line between the two neighboring planes and closest to the boundary. Each cluster that touches the boundary is assigned one or more extra vertices along the border to ensure that the chamfered edge detail of the boundary is planar. More than one extra vertex is used as necessary to ensure that the overall area of the polygonal remeshing closely matches the area of the input shape (Figure 11).

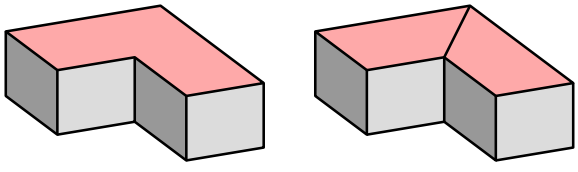


Figure 8: For non-convex input geometry, the intersection of proxy planes becomes more challenging. At the saddle point in the center of this example we must have either three facets of which at least one is non-convex (left) or four or more convex facets (right).

3 User Control

The algorithms described above were implemented in parallel with an architectural design seminar consisting of a half dozen design students. The participants applied the software and the various visualization tools for developing designs with complex curved geometry. They were exposed to and struggled with real-world fabrication constraints as they tried to coax their visions into shapes that could be constructed within the projected budget. Their experiences were critical in pushing the limits of the algorithm's robustness.

3.1 Surface Preparation

The input NURBS or subdivision surface must first be discretized: in our system the discrete mesh elements are triangles. Furthermore, the input geometry should be re-triangulated as necessary to ensure an appropriate resolution and density for the target shape and the number of panels that will be fit to its surface. The maximum edge length of the triangulated mesh bounds the quality of the iterative clustering optimization. This algorithm is a *discrete optimization* both in partitioning triangles into clusters and in determining the local neighborhood of the panels. Longer edge lengths reduce the number of possible discrete partitionings and possible

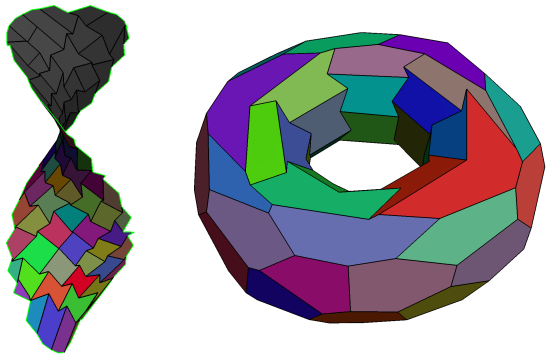


Figure 9: Computing a proper planar remeshing of surface twists like the helix example on the left is challenging. In particular, note the sharp, non-convex, *bowtie*-shaped facets created where the surface has double curvature.

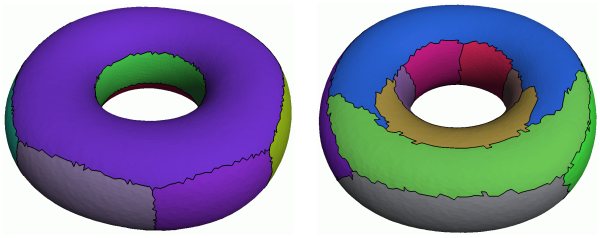


Figure 10: In the left image, the dark purple cluster is contiguous, but has a hole in the center. This cluster cannot be resolved with a single panel in our system and will be broken into multiple panels. In the right image, we see an alternate partitioning (using a greater number of seeds) that requires the blue, orange, and green clusters to intersect at two different points, which is impossible unless the planes are degenerate.

solutions. The total number of triangles dictates the performance. A single iteration of the clustering stage runs in $O(m \log m)$ time to assign m triangles, using a priority queue containing as many as m elements.

3.2 Clustering Metric Parameters

The designer using the tool has significant influence over the ultimate planar remeshing. First and foremost the user controls the number of panels, which is the dominating factor in the cost of the physical object. Using more panels will result in a surface that more accurately matches the original surface, but will most likely cost more to machine and assemble.

The designer also controls the general shape of the panels using the cluster metric. This is done by offering the designer a selection of simple metrics that can be used singly or combined into a larger function by weighting the different terms. The first metric we implemented was the Euclidean distance measured from the center of each triangle to the proxy centroid. This metric results in clusters that are generally round with uniform size, similar to the early Voronoi cell remeshing project (Section 1.2). We also implemented the two metrics demonstrated by Cohen-Steiner et al. [6]: the L^2 and $L^{2,1}$ metrics, which orient cells along the surface based on local curvature. With the fourth metric, the user can *paint* a density map over the surface to control the relative size of the panels. These four terms are applicable during *all* clustering iterations. Once the neighborhood of a cluster has been determined and the boundaries of a facet have been computed, our fifth metric is available. We project the centroid of a triangle to the target proxy plane and compare it to the boundary of the corresponding facet. If this point lies within the boundary the value of the final metric is zero. If it lies outside, the value is the distance to the boundary.

In order to combine these separate terms into a single useful metric, it is crucial that we normalize the terms so they can be compared appropriately. The $L^{2,1}$ metric is unit-less and can be used as is, but the other four metrics must be scaled relative to the anticipated average cluster diameter.

3.3 Procedural Scripting

Simple procedural scripting can be used to control the patterns formed through clustering. Figure 12 shows examples of patterns in nature one might wish to reproduce. In Figure 13 we demonstrate one such script that controls the density procedurally. We plan to expose this scripting interface to our users to allow procedural modeling, similar in spirit to the Renderman language [24]. Our architectural collaborators make heavy use of several modeling packages including the Rhino NURBS modeling system [17]. The

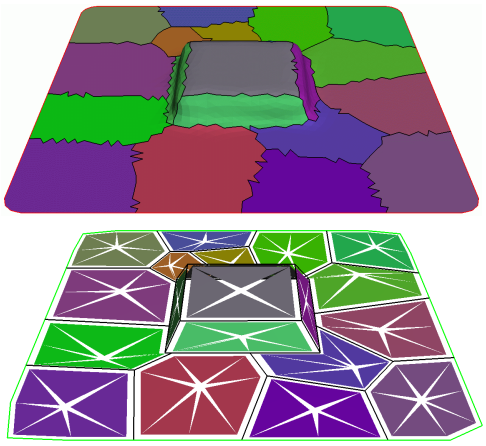


Figure 11: Although the wide band of triangles surrounding the bump in this model match very well using the L^2 and $L^{2,1}$ metrics, they cannot all be grouped together to form a single panel in our system, which requires that the planar facets be star-shaped. To ensure numerical stability, neighboring proxy planes with similar normal are snapped to a single master plane. Also, note that panels on the boundary of the geometry are assigned one or more extra vertices to ensure adequate coverage (Section 2.5).

algorithms described in this paper could be packaged as a plug-in to this modeling program and its companion scripting language.

4 Algorithm Summary and Parameters

The following pseudo-code summarizes our algorithm:

main
  place n seeds
  for $i = 1 \rightarrow k_1$ *initial distribution phase*
    lloyd_iter
  for $i = 1 \rightarrow k_2$ *planar intersection phase*
    determine facet neighborhood (Section 2.1)
    intersect neighbors
    "fix" neighborhood (Section 2.2)
    lloyd_iter

sub **lloyd_iter**
  assign triangles to clusters (Section 3.2)
  fit proxy plane to each cluster
  choose new seed
  snap/unsnap facets to master planes (Section 2.3)
  split/merge planes (Section 2.4)

The clustering parameters (Section 3.2) and n are set by the user. In our examples for this paper we have weighted the available metrics equally. The parameters k_1 and k_2 dictate the number of iterations of Lloyd's relaxation to be performed. For input models with larger polygon counts, larger values of these numbers should be used. Typically we find that 10-20 iterations for each loop is sufficient. If time is not a factor, these parameters can be removed and each loop is simply run until convergence (i.e., the assignment of each triangle to a cluster is constant).

5 Results

The performance of the system is primarily dependent on the number of triangles used to represent the original geometry. The surface

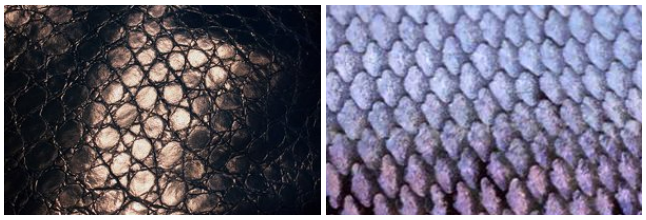


Figure 12: Organic patterns in nature such as the elephant and snake skins above inspired the architecture students participating in the seminar. The varying cell size in the left image can be achieved by varying a scaling term on the Euclidean distance cluster metric. The directional quality in the right image can be achieved with an anisotropic scaling that is specified procedurally or relative to surface properties such as curvature.

needs sufficient resolution to allow an even distribution of clusters. We recommend that architects aim for roughly 20-100 times as many triangles as desired clusters. For simple shapes the solver is quite efficient and can produce a mesh with 30 planar faces (shown in Figure 5) in seconds on a standard desktop machine. For complex geometry like that shown in Figures 15 & 16, where the solver must perform many iterative feedback loops, it can take several minutes to fully resolve the surface. The entire process remains quite interactive, allowing the user to adjust the total number of facets, the cluster metric, and the painted surface density as the possible surface tilings are explored.

When the designer is satisfied with the planar remeshing, the output of the program is passed through several additional scripts to prepare a parts list for the automated milling equipment. We have prototyped several scale test models of the fabrication system (Figure 14) and we are preparing for a large commissioned outdoor sculpture installation.

6 Limitations and Future Work

Determining if a particular target shape can be remeshed with n planar elements, with matching topology and genus, and within some specified error, remains an open problem. Due to the nature of the randomized and iterative optimization process, the program is not deterministic or entirely predictable. For the participants of the architectural seminar, this was both an advantage and a disadvantage. To address this issue and give more control to the designer, we would like to expand the controls and make them more intuitive. In some instances the user would like to add, remove or adjust individual panels within the model. As we further develop the procedural scripting capabilities we will learn what level of control is possible within the tight material constraints.

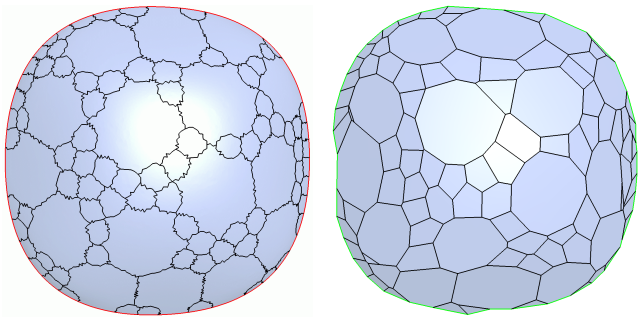


Figure 13: This patterning, which utilizes two distinct element sizes to mimic the left image of Figure 12, demonstrates the power of a procedural interface to control panel shape and structure.

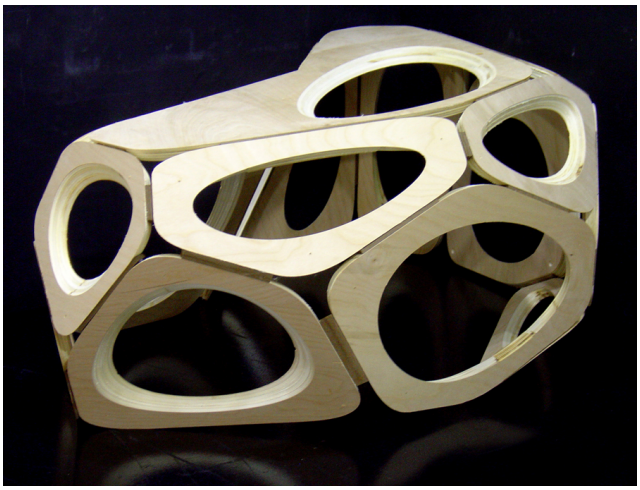


Figure 14: Photograph of a physical prototype to evaluate the edge connection detail. In this piece the facets were constructed from machine-milled plywood. A large hoop of material has been removed from the middle of each panel to emphasize the uniqueness of this remeshing, and the corners of each facet have been rounded to echo the curvature of the hoops.

Additionally, architects must adhere to certain practical constraints such as maintaining open interior space and building within the foot print specified by the site conditions. We have only begun to investigate the range of fabrication and structural challenges and geometry problems they generate. For example, an engineer may specify *varying* panel thickness based on a structural analysis of the form. To maintain a crisply defined interior surface, the chamfer angles between panels of different thicknesses must be adjusted.

7 Discussion and Conclusions

Our planar remeshing system is of great interest within the architecture community, and can be used not only to generate geometry or the final surface of a construction project, but can also be used to design the form-work for poured concrete, etc. The key for this and other interdisciplinary collaborations is to solve the hard computational problems while leaving appropriate controls in the hands of the designer.

Traditionally, there are significant construction cost savings to be gained when a design is modified so it can be realized with a series of tileable, identical, modular components. Now that computer-controlled milling machine technology (Figure 1b) has improved and is more affordable and available, it is more practical to consider designs composed of entirely unique and custom components. It is not unrealistic to imagine being able to design custom furniture on the internet and have it automatically machined and delivered to your door. The tools to do this must consider material and fabrication constraints while still allowing creative design. The work described in this paper is a significant step in that direction.

Acknowledgments

Much thanks and credit go to workshop organizer Professor Mark Goulthorpe and the workshop participants, including Michael Powell, John Rothenberg, Dennis Michaud, Matt Trimble, and Jeff Anderson, all from MITs Department of Architecture.

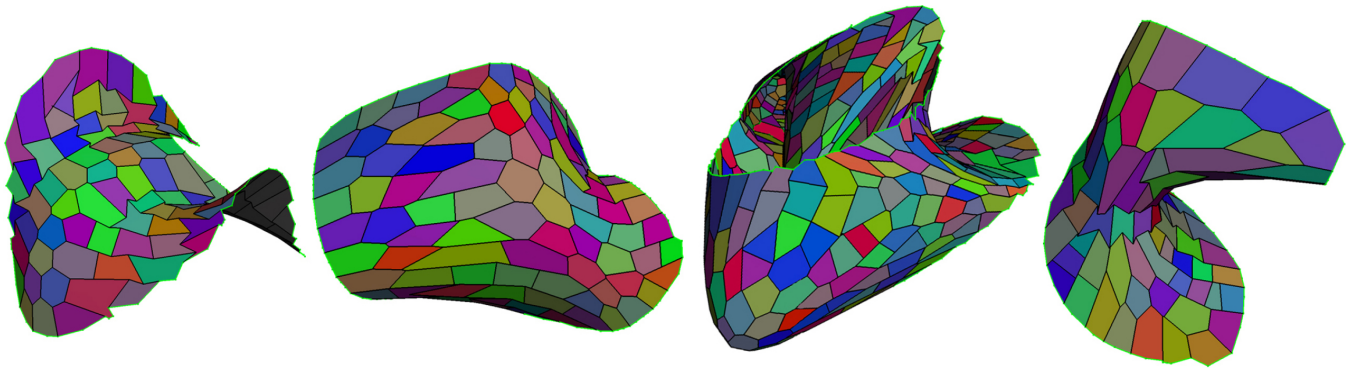


Figure 15: Additional designs created by workshop students to explore complex curvatures.

REFERENCES

[1] Pierre Alliez, Mark Meyer, and Mathieu Desbrun. Interactive geometry remeshing. *ACM Transactions on Graphics*, 21(3):347–354, July 2002.

[2] Joseph S. Amstock. *Handbook of Glass in Construction*. McGraw Hill, 1997.

[3] J. N. Rodrigues Branco and C. Guedes Soares. Mapping of shell plates of double curvature into plane surfaces. *Journal of Ship Production*, 21(4):248–257, November 2005.

[4] Matthew Brand. Charting a manifold. *Neural Information Processing Systems (NIPS)*, December 2002.

[5] H.-Y. Chen, I.-K. Lee, S. Leopoldseder, H. Pottmann, T. Randrup, and J. Wallner. On surface approximation using developable surfaces. In *Graphical Models and Image Processing*, 1998.

[6] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. *ACM Transactions on Graphics*, 23(3):905–914, August 2004.

[7] Mark de Berg, Otfried Schwarzkopf, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2 edition, 1997.

[8] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of ACM SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 209–216, 1997.

[9] Paul Haeberli. Lamina Design LLC, 2005. `http://laminadesign.com/contact.html`.

[10] Alan Holgate. *The Art of Structural Engineering: The Work of Jörg Schlaich and his Team*. Edition Axel Menges, 1997.

[11] Hugues Hoppe. Progressive meshes. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 99–108, August 1996.

[12] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *Proceedings of SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, pages 19–26, August 1993.

[13] Dan Julius, Vladislav Kraevoy, and Alla Sheffer. D-charts: Quasi-developable mesh segmentation. In *Computer Graphics Forum, Proceedings of Eurographics 2005*, volume 24(3), pages 581–590, Dublin, Ireland, 2005. Eurographics, Blackwell.

[14] D. Kirasanov and S. J. Gortler. A discrete global minimization algorithm for continuous variational problems. Technical report, Harvard Computer Science Technical Report: TR-14-04, July 2004.

[15] Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. Geometric modeling with conical meshes and developable surfaces. *ACM Transactions on Graphics*, 25(3):681–689, July 2006.

[16] Jun Mitani and Hiromasa Suzuki. Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Transactions on Graphics*, 23(3):259–263, August 2004.

[17] Rhino. Rhinoceros: NURBS modeling for Windows, 2005. `http://www.rhino3d.com/`.

[18] Sam Roweis and Lawrence Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, Dec 2000.

[19] Hans Schober and Kai Kuerschner. Meraviglioso. *Civil Engineering Magazine*, 75(12), December 2005.

[20] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. *Computer Graphics*, 26(2):65–70, 1992.

[21] S. Stephan, J. Sanchez-Alvarez, and K. Knebel. Reticulated structures on free-form surfaces, 2003. `http://www.mero.de/bausysteme/downloads/artikel/free_ret_stru_e.pdf`.

[22] Vitaly Surazhsky, Pierre Alliez, and Craig Gotsman. Isotropic remeshing of surfaces: a local parameterization approach. In *Proceedings of the 12th International Meshing Roundtable*, pages 215–224, September 2003.

[23] Greg Turk. Re-tiling polygonal surfaces. In *Proceedings of ACM SIGGRAPH 92*, Computer Graphics Proceedings, Annual Conference Series, July 1992.

[24] Steve Upstill. *The Renderman Companion : A Programmer's Guide to Realistic Computer Graphics*. Addison-Wesley, 1990.

[25] Jianhua Wu and Leif Kobbelt. Partitioning to hybrid surfaces. In *Computer Graphics Forum, Proceedings of Eurographics 2005*, volume 24(3), pages 277 – 284, Dublin, Ireland, 2005. Eurographics, Blackwell.

[26] Waclaw Zalewski and Edward Allen. *Shaping Structures: Statics*. John Wiley & Sons, Inc., 1998.

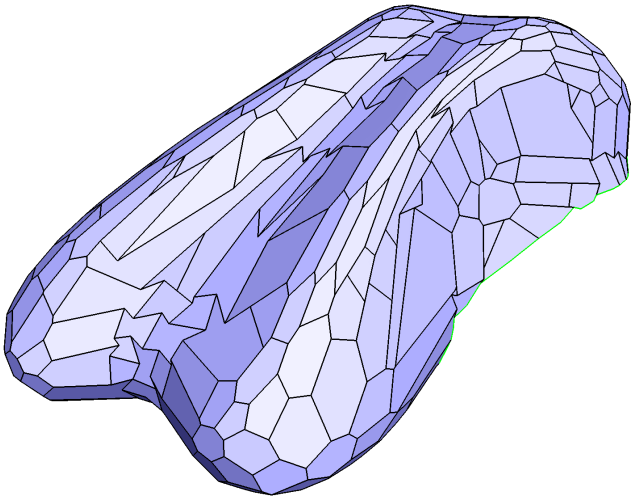


Figure 16: The freeform geometry of the glass rooftop greenhouse design from Figure 1a remeshed with planar panels.