Triangle Surfaces with Discrete Equivalence Classes

Mayank Singh*
Texas A&M University

Scott Schaefer†
Texas A&M University

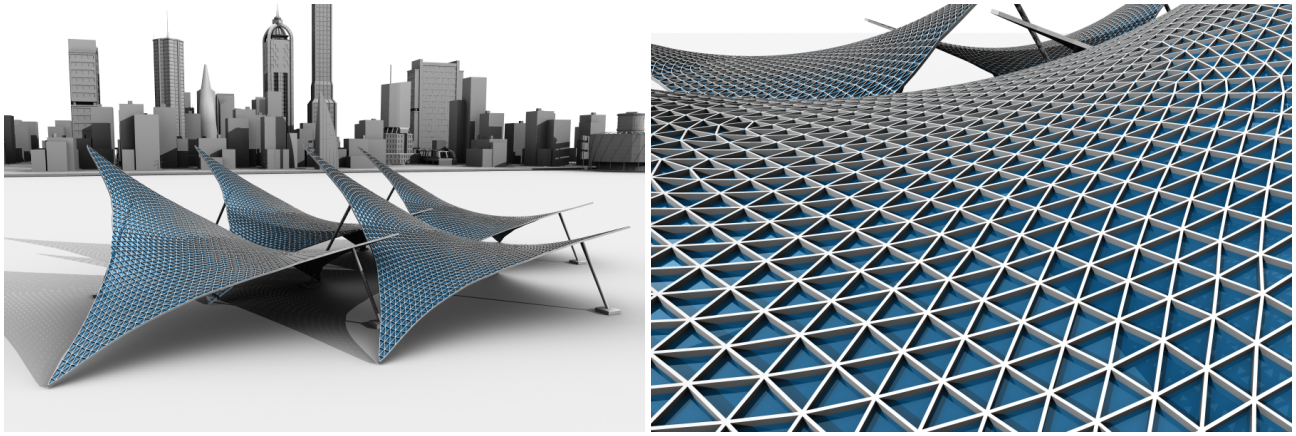

Figure 1: *An architectural rendering of a four-point tensile surface whose original shape contained* 3200 *unique triangles optimized to* 10 *unique polygons (only* 0.3% *of the total polygons) using our method. Left shows a close-up of the tessellation of the surface.*

Abstract

We propose a technique that takes a triangulated surface as input and outputs a surface with the same topology but altered geometry such that each polygon falls into a set of discrete equivalence classes. We begin by describing an error function that measures how close the polygons are to satisfying this criteria. To optimize this error function, we first cluster triangles into discrete sets such that the assignment of sets minimizes our error. We then find canonical polygons for each set using nonlinear optimization. Next, we solve a Poisson equation to find positions of vertices such that the surface polygons match the canonical polygons as close as possible. We also describe how to incorporate a fairness criteria into the optimization to avoid oscillations of the surface. We iterate this entire process until we reach a user specified tolerance, possibly adding clusters during iteration to guarantee convergence. We have been able to successfully reduce the number of unique triangles to lie within a small percentage of the total number of triangles in the surface and demonstrate our technique on various examples.

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

Keywords: mesh discretization, discrete sets, equivalence classes

1 Introduction

Modeling freeform shapes has many uses such as representing characters in digital movies, modeling the body of a car for manufacturing purposes, or depicting the shape of a building for architectural applications. The last application, architectural modeling, has seen much work recently in applying optimization to enforce various geometric properties that can aid in the practical construction of these shapes. For example, creating shapes with planar quad meshes [Liu et al. 2006; Cutler and Whiting 2007], constructing offset meshes for beam layout [Pottmann et al. 2007], modeling shapes with curved panels [Pottmann et al. 2008] and tiling surfaces with circles [Schiftner et al. 2009] have all been studied recently.

However, one problem that has been overlooked thus far is that the pieces that make up these shapes require a great deal of customization. For example, when modeling a freeform building from triangles or planar quad panels, it is very likely that each individual polygon is unique when compared to the other polygons. In terms of manufacturing, this uniqueness ignores economies of scale and requires each piece to be custom manufactured.

An alternative solution is to somehow model the shape with panels that fall into a small number of discrete sets. In this case, there are only a small number of unique panels with many of each individual panel type used to tile the surface. Such an approach was recently used in the construction of the Beijing National Aquatic Center where the originally proposed surface consisted of a very high number of unique shapes. In order to simplify the construction process, 4000 panels were reduced to a small number of sets [Drew 2008, p. 190]. However, this example was simple in that the walls and roof were planar. We wish to automate this reduction process and apply it to arbitrary freeform shapes.

2 Background

2D periodic or aperiodic tilings of the plane have been well studied [Grünbaum and Shephard 1986]. However, freeform shapes that may possess arbitrary curvature are substantially more difficult to tile. To our knowledge, the problem of modeling 3D freeform shapes with discrete sets has not been addressed previously. However, we briefly discuss other similar work such as mosaic tiling i.e. repeated tiling of a small number of shapes covering a given space.

Mosaics attempt to cover space with a set of tiles, possibly leaving gaps in between the tiles. A prominent contribution for generating mosaic in 2D was done by Kim and Pellacini [Kim and Pellacini 2002], where a set of image tiles of arbitrary shape were used to recreate an given image. The tiles would fit the image nearly perfectly, leaving negligible space in between the tiles. Elber and Wolberg [Elber and Wolberg 2003] improved upon the idea by ex-

*e-mail: mayank@cs.tamu.edu
†e-mail:schaefer@cs.tamu.edu

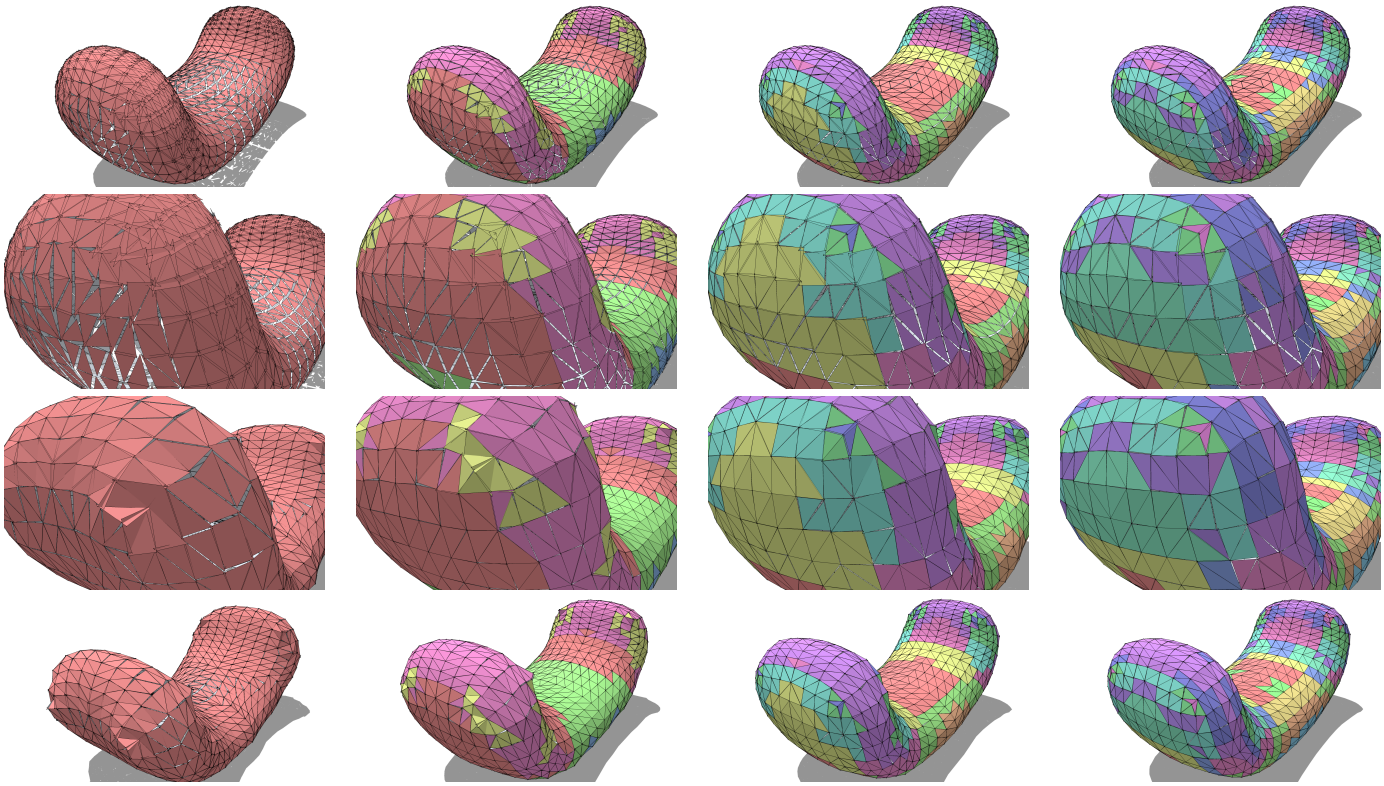

Figure 2: *The effect of cluster numbers on optimization. Each figure shows the mapping of clusters to polygons and show the canonical polygon for that cluster mapped onto the surface. The first two rows show the effect of only using clustering. The bottom two rows show the effect of global optimization on the vertex positions.*

tracting free-form curves in an image and placing tiles along these curves. More recent work [Pavic et al. 2009] takes a multiresolution approach to the problem.

The concept of using tiles was extended to 3D by Lai et al. [Lai et al. 2006]. In this case, the authors place equal-sized quadrilateral tiles on the surface, oriented along the curvature lines. To place the tiles, they minimize a spring-like energy over the entire surface. Even though this method covers the entire surface with a single sized quad, the spacing between the tiles is significant and filled with grout, which makes this technique unsuitable for building construction purposes.

Recently, Passos and Walter [Dos Passos and Walter 2008] adapted this method to include variable sized tiles and achieved much larger coverage. This method places a number of user defined tiles on the surface based on curvature with larger number of tiles in regions of high curvature. The simulation then uses a relaxation procedure to move tiles away from each other leaving gap for grout. Even though this method improves upon the prior work, it may still fall short of the expectations in construction where the spacing between tiles must fall within a small tolerance.

In contrast to these mosaic methods, we do not allow the user to specify the tiles of the surface, but instead find the canonical polygons corresponding to each equivalence class through optimization. While 3D mosaics have been designed to have gaps between the tiles filled with grout, we aim to reduce the spacing between canonical polygons far below that used in these methods.

Fu et al. [2010] and Eigenstaz et al. [2010] developed methods to approximate surfaces with discrete types of objects in parallel to our paper. Eigenstaz et al. [2010] do not try to use small numbers of congruent shapes to represent a surface but address a related problem of what types of surfaces to use to minimize construction costs. Fu et al. [2010] also address a similar problem to ours except they use non-planar quadrilateral surfaces.

3 Discrete Equivalence Classes

We will assume that we are given a triangulated surface with or without boundary as input as well as an initial number of clusters n and a tolerance ϵ that we would like to converge to measured as a percentage of the bounding box diagonal of the shape. Our goal is to modify the geometry of the triangles such that we maintain the appearance of the input surface while having all of the triangles fall into one of the n equivalence classes.

We begin by defining a notion of similarity between two triangles, which will allow us to construct our equivalence classes. Given two triangles A, B with vertices a_1, a_2, a_3 and b_1, b_2, b_3 we measure the distance between triangles by considering rigid transformations of the two polygons under all possible correspondences of their vertices. Hence, their distance is

$$D(A,B) = \min_{R^T R=I, T, j} \sum_{\ell=1}^{3} \left| Rb_{perm(j,\ell)} + T - a_\ell \right|^2 \qquad (1)$$

where R, T represent a rigid transformation and $perm(j,\ell)$ represents the ℓ^{th} element of the j^{th} permutation of the indices $\{1,2,3\}$.

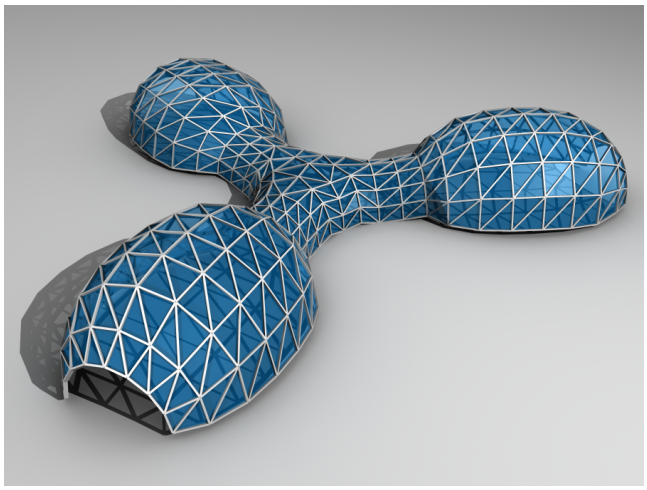

Figure 3: *A curved roof structure with 576 triangles, which was reduced down to* 6 *unique clusters with our method.*

There are six of these permutations and they alter the correspondence between the vertices of A and B to find the best match between the vertices of the triangle. We allow the orientation of the vertices of B to change and therefore consider reflections as well as rotations to find the best correspondence between the two polygons. We should note that Fu et al. [2010] developed an identical metric in parallel to our work but for quads rather than triangles.

To find the best rigid transformation between the two polygons, we use the method of [Arun et al. 1987]. The translation is given by

$$T = \bar{B} - R\bar{A}$$

where $\bar{A}, \bar{B}$ represent the centroids of the two polygons. Now, let $a^*_\ell = a_\ell - \bar{a}$, $b^*_\ell = b_\ell - \bar{b}$, and define M to be

$$M = \sum_{\ell=1}^{3} a^*_\ell \left(b^*_\ell\right)^T.$$

R is then given by $R = UV^T$ where $M = U\Sigma V^T$ is the singular value decomposition of M. If $det(R) < 0$, we simply negate the vector in V corresponding to the smallest singular value.

This distance function is symmetric in A, B and has the property that if two triangles are a rotation or reflection of each other, then their distance will be zero. This metric makes sense for applications where we construct surfaces out of unoriented panels such as glass. However, if panels must be oriented (inside vs. outside faces), then we could easily incorporate this restriction by reducing the permutations in $perm(j, \ell)$ to the three oriented permutations of the vertices.

3.1 Clustering

In order to transform an existing triangulated mesh into a small finite set of representative triangles, we partition the triangles in the surface into a set of discrete clusters. For each cluster we will build a representative polygon that we refer to as the *canonical polygon*. This stage of the optimization seeks to minimize the following function

$$\min_{C_j, ind} \sum_i D(P_i, C_{ind(i)}) \tag{2}$$

where C_j is the canonical polygon for the j^{th} cluster and $ind(i)$ gives the cluster to which the polygon P_i is assigned.

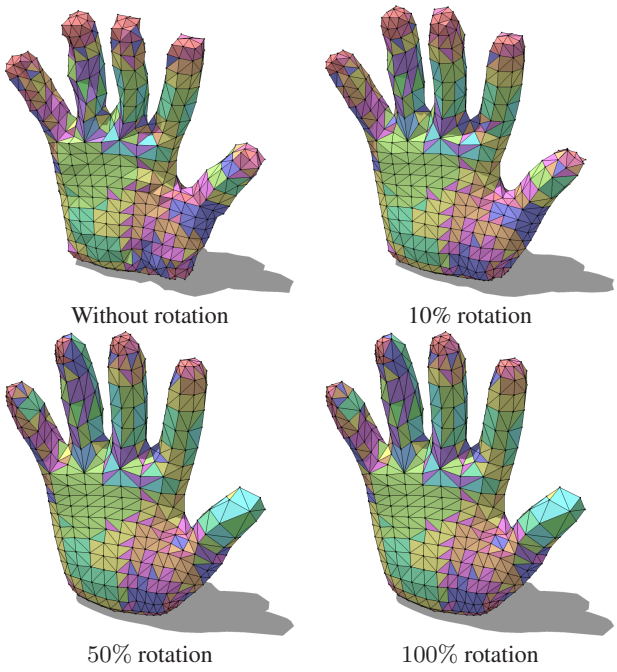

Figure 4: *Example of an optimized mesh showing the effect of rotating canonical triangles towards the normal of the closest point on the surface. Notice how even a small amount of rotation can fix most of the artifacts cause by surface oscillation.*

A commonly used algorithm to optimize functions of the form of Equation 2 is k-means clustering or some variant thereof [Arthur and Vassilvitskii 2007]. This clustering is performed in an alternating fashion. First we assume the C_j are fixed and optimize ind by assigning the P_i to the closest canonical polygon as measured by the distance function $D(P_i, C_j)$. Then we assume ind is fixed to optimize for C_j.

Since the distance function in Equation 1 is very nonlinear, finding the canonical polygon C_j may be difficult. Given an assignment ind of the polygons to clusters, we perform a nonlinear optimization of Equation 2 using the Levenberg-Marquardt algorithm [Frandsen et al. 2004]. We note that a 3D triangle under the class of orthogonal transformations does not have nine degrees of freedom, but only three. There are many ways of representing these three degrees of freedom in terms of lengths and angles of the triangle. We choose a simple representation of the vertices of C_j as

$$\begin{aligned} C_{j,1} &= (0,0,0) \\ C_{j,2} &= (x_2,0,0) \\ C_{j,3} &= (x_3,y_3,0). \end{aligned}$$

Hence, we need only optimize for the variables x_2, x_3, y_3, which greatly reduces the size of the optimization problem and leads to fast convergence.

While k-means clustering typically begins with random seeds, we use the variant proposed by [Wang et al. 2009] as the result is deterministic and tends to produce better results. We begin our optimization with a single cluster and run the k-means clustering to convergence. We then iteratively add a new cluster corresponding to the polygon with the worst error in the summation from Equation 1 and repeat this process until n clusters have been added.

Figure 2 (top) illustrates clustering with an example of a simple bean shaped surface and is clustered using varying number of

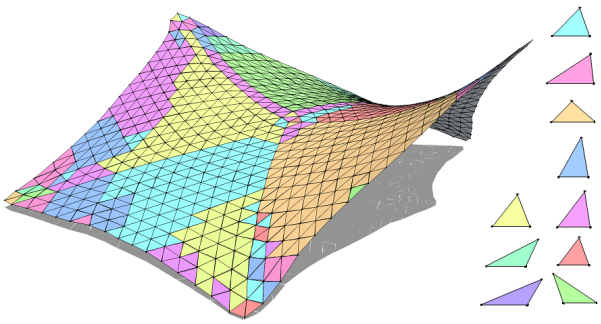

Figure 5: 5 *Point Tensile Membrane with* 1280 *triangles optimized with* 10 *clusters. The canonical polygons found by the optimization are shown on the right.*

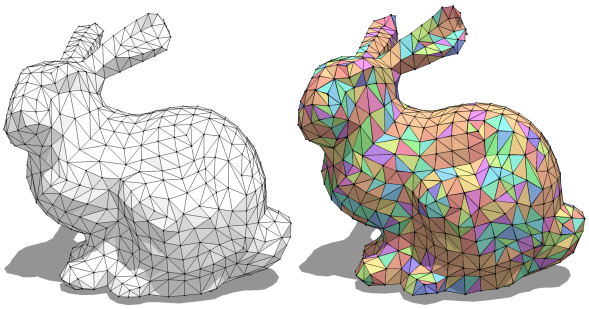

Figure 6: *A complex shape of a bunny with* 1724 *polygons optimized using* 42 *clusters with the clusters drawn as different colors. The initial shape of the object before optimization is shown on the left.*

clusters. For each figure we show the canonical polygon $C_{ind(i)}$ mapped onto the polygon P_i using the best rigid transformation and vertex correspondence from Equation 1. We denote this transformed polygon by $\hat{C}_{ind(i)}$. With higher numbers of clusters, the gaps and overlaps in the surface diminish even though these results lack the position optimization performed in Section 3.2, which improve the results significantly (see the lower part of the figure).

3.2 Global Optimization

So far we have attempted to minimize the error in Equation 2 by clustering similar triangles and replacing them with a best fit canonical polygon. However, this process does not guarantee a perfectly aligned mesh with canonical polygons mapped in place of the polygons of the mesh (as evident in Figure 2). Though clustering can trivially reduce the error in Equation 2 to zero by assigning each polygon to its own cluster, we aim to do so using only a small number of clusters.

Given assignment of triangles to clusters and the transformed canonical polygons $\hat{C}_{ind(i)}$, we must find a surface whose polygons match the canonical polygons. Yu et al. [2004] considered a very similar problem in the context of mesh deformation, which was solved using the Poisson equation. Similarly, we solve a Poisson equation to find the new positions of the vertices to match our canonical polygons. The Poisson equation attempts to find vertex positions for the shape P such that

$$E_g = \sum_i \left| \nabla P_i - \nabla \hat{C}_{ind(i)} \right|^2 \Delta_i$$

is minimized where ∇P_i is the gradient of the triangle P_i and Δ_i is the area of P_i.

To avoid the trivial global optimum of all zeros and to maintain the shape of the initial surface, we also add a closeness term to the Poisson optimization. For each polygon P_i of the current surface, we find the closest point and normal (x_i, n_i) on the initial shape P^0 to P_i's centroid $\bar{P}_i$ and define this error as the distance squared to the tangent planed formed by (x_i, n_i).

$$E_c = \sum_i \left(n_i \cdot (\bar{P}_i - x_i) \right)^2$$

By minimizing the distance to the tangent planes, we allow polygons to slide along the surface in nearly planar regions to fit the canonical polygons better while still maintaining the shape of the initial surface P^0.

For surfaces with boundary, we also add another term that measures the deviation of the vertices on the boundary of the current surface P compared with the initial shape P^0. If p_ℓ is a vertex on the boundary of P and y_1, y_2 are vertices on the boundary of P^0 such that their edge is the closest to p_ℓ, then the error is given by

$$E_b = \sum_\ell \left| p_\ell - y_1 - \frac{(p_\ell - y_1) \cdot (y_2 - y_1)}{|y_2 - y_1|^2} (y_2 - y_1) \right|^2$$

where this summation is over all the boundary vertices of P.

We use these three error functions in our global optimization and minimize

$$\min_P E_g + \alpha E_c + \beta E_b$$

where α and β begin at a small constants (we use 0.001 and 0.01) and decrease with each iteration of the optimization. This error function is quadratic and its minimum is given by the solution of a sparse system of linear equations. After solving for new positions for the vertices of P using this linear system, we then iterate the entire process by reclustering, finding new canonical polygons and solving the global optimization again until the process converges.

Unfortunately, simply incorporating these closeness terms into the optimization does not maintain the shape of the surface, though they do prevent the optimized surface from drifting too far from the original shape. Figure 4 (top left) shows the result of using this closeness metric. The optimization can quickly push polygons away from the original shape and the optimized surface may take on a jagged appearance. This optimized shape is geometrically close to the original shape, yet the normals of the shape are far from that of the original surface.

Ideally, we would like both the geometry and the normals of the optimized surface to match that of the original shape. We provide a simple method for doing so by modifying the orientation of each of the transformed canonical polygons before solving the Poisson equation. Let n_i be the normal of the closest point on P^0 to $\bar{P}_i$ and let m_i be the normal of P_i. We then rotate the transformed canonical polygon $\hat{C}_{ind(i)}$ about the axis $m_i \times n_i$ by a small amount prior to solving the Poisson equation. Figure 4 shows the effect of rotating by 10% θ_i degrees (top right) where θ_i is the angle between m_i and n_i, 50% θ_i (bottom left) and θ_i (bottom right). Notice that even modifying the orientation of the canonical polygons slightly is enough to greatly improved the quality of the optimized surface. For our optimizations, we use 10% θ_i as the rotation amount since large amounts of rotation negatively affect the error in the optimization and require more clusters to obtain the same error tolerance.

	Tris	Clusters	Mean Error
Fig 1	3200	10	0.014%
Fig 3	576	6	0.033%
Fig 5	1280	10	0.029%
Fig 6	1724	44	0.051%
Fig 7	2492	64	0.026%
Fig 9 top lhs	1396	84	0.055%
Fig 9 mid rhs	676	19	0.023%
Fig 9 btm lhs	1800	31	0.008%
Fig 9 btm rhs	2880	58	0.012%

Table 1: *The number of clusters and error for various models. The error is of the form mean error in terms of a percentage of the length of the bounding box diagonal from the initial shape.*

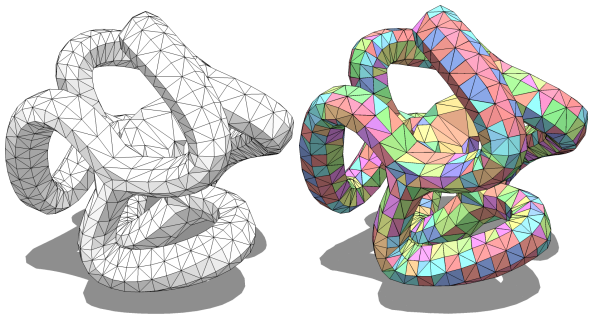

Figure 7: *A high genus shape with* 2492 *polygons. The initial shape is shown on the left and the optimized shape with* 64 *clusters is shown on the right.*

4 Discussion and Results

Our optimization iterates through clustering/finding canonical polygons and global optimization to reduce the error in Equation 2. If at any point during the optimization the maximal distance between vertices of the optimized mesh and the transformed canonical polygons is less than the user specified tolerance ϵ we terminate the optimization. As seen in Figure 8 the error drops dramatically as the optimization proceeds and gradually levels off. If the optimization does not achieve the specified tolerance, we simply insert another cluster corresponding to the polygon with the worst error in the summation in Equation 2 and continue the optimization. Hence, if ϵ is set very low, then a large number of clusters may be required to achieve this tolerance depending on the shape of the object. We summarize the results for many of our shapes in Table 1.

In terms of time, our optimization requires n nonlinear optimizations per iteration to find the canonical polygons each of which may be repeated several times in the k-means clustering until the assignments in ind converge. Furthermore, we also solve a global system of equations in each iteration. Hence, our optimization is not fast. However, we would like to emphasize that this optimization is only designed to be run as a post-processing step in the modeling process. As an example, the shape in Figure 8 contains 1792 polygons and we optimized to 17 clusters. Initial clustering takes 499 seconds and each step of the optimization (reclustering and global optimization) takes about 9 seconds on an Intel Core 2 6700. A typical optimization may run for hundreds or even a few thousand iterations, though this amount is strongly dependent on ϵ.

Figures 1, 2, 3, 4, 6, 7 and 9 show examples of shapes that we have run our optimization on. In all of these examples, we show the transformed canonical polygons $\hat{C}_{ind(i)}$ instead of the optimized polygonal surface P so that any gaps/overlaps are visible; though the error tolerance is set low enough that these discrepancies are not visible. Figure 5 shows an example of our optimization run on a tensile structure where the clustering is depicted by different color polygons. This shape has 1280 triangles and we use only 10 canonical polygons, shown on the right of the image. We terminate our optimization based on the maximal error, the average error for these shapes is typically far lower. For example, the mean error is only 0.029% of the length of the bounding box diagonal.

For complex shapes such as Figures 6 and 7, more clusters are typically required to achieve low error tolerances. Even though the number of clusters may be higher than for other shapes, the total number of clusters is still only a small fraction of the total number of polygons (about 2.4% in the case of Figure 6). Furthermore, closed shapes tend to be more difficult than shapes with boundaries to optimize as well. Closed shapes with zero error must satisfy a global constraint implied by the Gauss-Bonnet theorem, which states that the integral of curvature for closed shapes is a constant dependent on the genus of the surface. When boundaries are present, the surface has more degrees of freedom to shrink or expand even when we add an error term to maintain the shape of the boundary during optimization.

The initial number of clusters n the user specifies also has an effect on the result of the optimization. If we start with a single cluster, run the optimization to convergence and repeat by adding a cluster each time up to n clusters, we tend to achieve a lower error than simply beginning the optimization with n clusters. Figure 8 shows the difference between these two approaches as well as a plot of the error from Equation 2 during optimization. If we begin the optimization with a single cluster, the initial error is much higher than if we start with n. However, if we begin with n clusters, the error converges quickly. In contrast, if we begin with a single cluster and incrementally add clusters during optimization, we often achieve a lower total error after reaching n clusters. The disadvantage of incrementally adding clusters is that the canonical polygons will be very similar to one another. This effect can be seen in the optimized shapes in Figure 8 where the polygons are more uniform in shape.

5 Future Work

While our optimization works well, there are some areas that we would like to improve in the future. Outliers may be a problem for our optimization where the clusters are relatively tight except for a few triangles of the surface. By identifying and removing these outliers from the optimization we may be able to use fewer repeatable clusters at the cost of having a small percentage of the polygons as custom shapes.

Our method is also very dependent on the topology of the shape and we currently assume that the topology is fixed. Different triangulations of the same or similar shapes may produce different results in terms of the final error and number of clusters. We may be able to improve our optimization by allowing small changes to the topology of the shape such as edge flips. However, this strategy would need to take into account the shape of the initial object as well as its normals since the local curvature of the object may change significantly after such an edge flip.

Finally, we would like to incorporate n-gons into our optimization instead of simply restricting ourselves to triangles though doing so may be complex. In terms of practical construction, these n-gons will typically need to be planar surfaces [Liu et al. 2006]. The constraints for planar quad meshes are already non-linear and will certainly harm the convergence of our method, possibly requiring substantially more clusters.

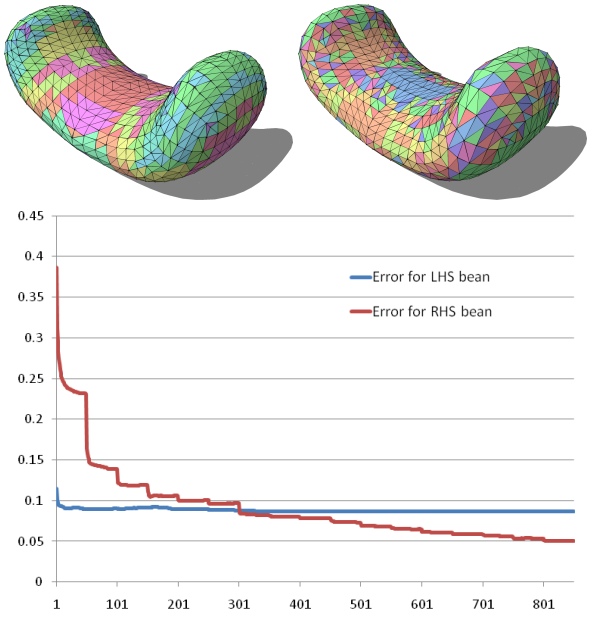

Figure 8: *Example of different clustering methods. Top Left: the shape is partitioned into* 17 *clusters before running global optimization. Top Right: the same shape starting with* 1 *cluster and incrementally adding clusters and running the optimization to convergence each time a cluster is added up to* 17 *clusters. Bottom: the error graph for the two optimizations (blue is for left shape, red is for right shape).*

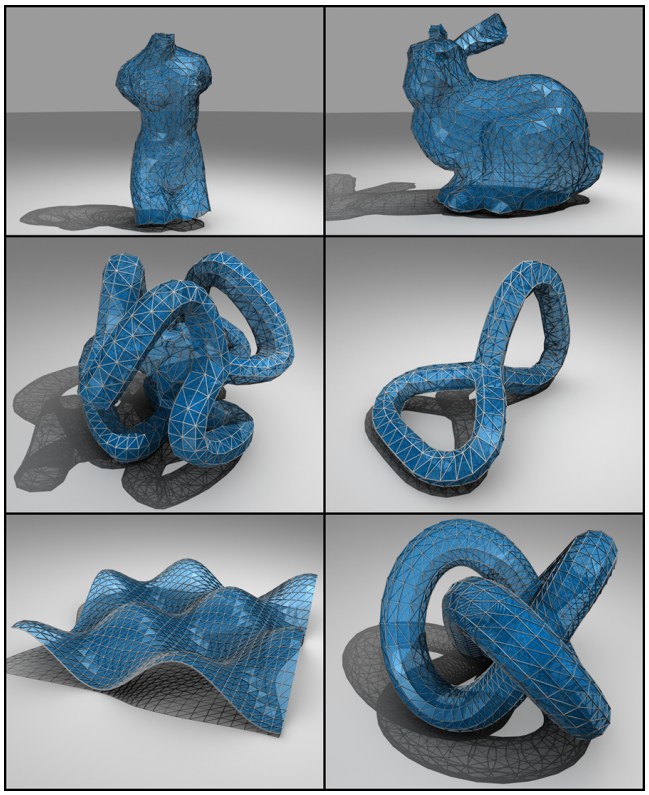

Figure 9: *Final results of various shapes with different number of clusters. The details of each shape are summarized in Table 1.*

References

ARTHUR, D., AND VASSILVITSKII, S. 2007. k-means++: the advantages of careful seeding. In *Proceedings of the* 18*th annual ACM-SIAM symposium on Discrete algorithms*, 1027–1035.

ARUN, K. S., HUANG, T. S., AND BLOSTEIN, S. D. 1987. Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell. 9*, 5, 698–700.

CUTLER, B., AND WHITING, E. 2007. Constrained planar remeshing for architecture. In *Proceedings GI '07*, ACM, New York, NY, USA, 11–18.

DOS PASSOS, V. A., AND WALTER, M. 2008. 3d mosaics with variable-sized tiles. *Vis. Comput. 24*, 7, 617–623.

DREW, P. 2008. *New Tent Architecture*. Thames & Hudson, June.

EIGENSATZ, M., KILIAN, M., SCHIFTNER, A., MITRA, N., POTTMANN, H., AND PAULY, M., 2010. Paneling architectural freeform surfaces. To appear in ACM SIGGRAPH.

ELBER, G., AND WOLBERG, G. 2003. Rendering traditional mosaics. *The Visual Computer 19*, 67–78.

FRANDSEN, P. E., JONASSON, K., NIELSEN, H. B., AND TINGLEFF, O., 2004. Unconstrained optimization, 3rd edition.

FU, C.-W., LAI, C.-F., HE, Y., AND COHEN-OR, D., 2010. K-set tilable surfaces. To appear in ACM SIGGRAPH.

GRÜNBAUM, B., AND SHEPHARD, G. C. 1986. *Tilings and patterns*. W. H. Freeman & Co., New York, NY, USA.

KIM, J., AND PELLACINI, F. 2002. Jigsaw image mosaics. In *ACM SIGGRAPH '02*, ACM, New York, NY, USA, 657–664.

LAI, Y.-K., HU, S.-M., AND MARTIN, R. R. 2006. Surface mosaics. *Vis. Comput. 22*, 9, 604–611.

LIU, Y., POTTMANN, H., WALLNER, J., YANG, Y.-L., AND WANG, W. 2006. Geometric modeling with conical meshes and developable surfaces. In *ACM SIGGRAPH '06*, ACM, New York, NY, USA, 681–689.

PAVIC, D., CEUMERN, U., AND KOBBELT, L. 2009. Gizmos: Genuine image mosaics with adaptive tiling. *Comput. Graph. Forum 28*, 8, 2244–2254.

POTTMANN, H., LIU, Y., WALLNER, J., BOBENKO, A., AND WANG, W. 2007. Geometry of multi-layer freeform structures for architecture. In *ACM SIGGRAPH '07*, ACM, New York, NY, USA, 65.

POTTMANN, H., SCHIFTNER, A., BO, P., SCHMIEDHOFER, H., WANG, W., BALDASSINI, N., AND WALLNER, J. 2008. Freeform surfaces from single curved panels. In *ACM SIGGRAPH '08*, ACM, New York, NY, USA, 1–10.

SCHIFTNER, A., HÖBINGER, M., WALLNER, J., AND POTTMANN, H. 2009. Packing circles and spheres on surfaces. In *ACM SIGGRAPH Asia '09*, ACM, New York, NY, USA, 1–8.

WANG, L., YU, Y., ZHOU, K., AND GUO, B. 2009. Example-based hair geometry synthesis. In *ACM SIGGRAPH '09*, ACM, New York, NY, USA, 1–9.

YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2004. Mesh editing with poisson-based gradient field manipulation. In *ACM SIGGRAPH '04*, ACM, New York, NY, USA, 644–651.