DISCRETE TRIANGULATED MESHES FOR ARCHITECTURAL DESIGN AND FABRICATION

A Dissertation

by

MAYANK SINGH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2011

Major Subject: Computer Science

DISCRETE TRIANGULATED MESHES FOR ARCHITECTURAL DESIGN AND FABRICATION

A Dissertation

by

MAYANK SINGH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Scott Schaefer
Committee Members,	John Keyser
	Andruid Kerne
	Richard Davison
Head of Department,	Valerie E. Taylor

May 2011

Major Subject: Computer Science

ABSTRACT

Discrete Triangulated Meshes for Architectural Design and Fabrication. (May 2011)

Mayank Singh, B.S., Indian Institute of Technology at Roorkee;

M.S., Marquette University

Chair of Advisory Committee: Dr. Scott Schaefer

Recent innovations in design and construction of architectural buildings has led us to revisit the metrics for discretizing smooth freeform shapes in context with both aesthetics and fabrication. Inspired by the examples of the British Museum Court Roof in Britain and the Beijing Aquatic Centre in China, we propose solutions for generating aesthetic as well as economically viable solutions for tessellating smooth, freeform shapes.

For the purpose of generating an aesthetic tessellation, we propose a simple linearized strain based metric to minimize dissimilarity amongst triangles in a local neighborhood. We do so by defining an error function that measures deformation required to map a pair of triangles onto each other. We minimize the error using a global non-linear optimization based framework.

We also reduce the complexity associated with prefabricating triangulated panels for a given shape. To do so, we propose a global optimization based framework to approximate any given shape using significantly reduced numbers of unique triangles. By doing so, we leverage the economies of scale as well as simplify the process of physical placement of panels by manual labor.

To everyone who helped me along the way

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

A. Architectural Design Process

Architectural design process begins with a set of requirements and associated constraints presented to an architect by a client. The architect proposes a creative design solution, that best satisfies the initial set of requirements. The proposed design is then handed over to a group of domain experts. These domain experts then analyze various aspects of the preliminary design. This analysis covers the study of structural integrity, adequate lighting, thermal comfort, acoustic reverberations, and cost limitations of the design. Based on expert recommendations the architect reviews and revises the initial design. This cycle continues until all involved groups are satisfied. Figure 1 is a diagrammatic representation of this process.

It is noteworthy that the ever-increasing computational-processing power is promoting the rapid decline of feedback time for simulation and analysis. The increase in processing power has directly contributed to increase in the number of design improvement iterations. Axel Kilian [Kil06], in his doctoral research, elaborates in detail how modern design practices are being influenced by bidirectional simulation and fabrication constraints.

This iterative improvement process also occurs in the construction phase of the project. The focus of our work lies in the latter part of design process, that is the analysis and optimization of design for construction purposes.

This dissertation follows the style of *Computational Aesthetics.*

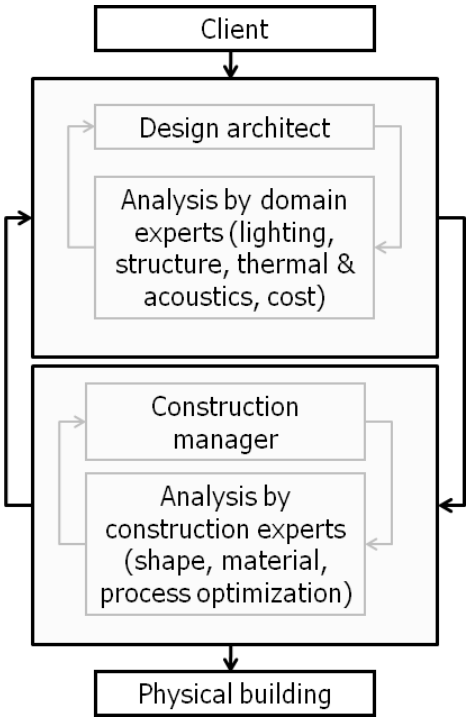


Fig. 1. Overview of architectural design process.

B. Digital Representation and Fabrication of Freeform Shapes for Architecture

The design process described above is common to all building shapes. We however focus on a small subset of design shapes referred to as freeform surfaces. Unlike traditional architecture where building shapes are generated using orthogonal planes, freeform shapes are created using smoothly curving surfaces. Figure [2] shows an example of a freeform design created by Frank O' Ghery. Freeform shapes are inspired by various sources such as naturally occurring organic shapes, continuous mathematical functions, or other aerodynamic shapes.

Advances in digital modelling has made it possible to generate complex freeform architectural designs with minimal input. Until recently such shapes were limited to other industries such as automotive, aviation, and shipping. This recently adopted architectural design practice has brought forward a new set of shape representation

Fig. 2. Guggenheim Museum Bilbao, Spain is an example of free-form architectural design using developable surface (©Bastian).

and fabrication challenges. Beyond digital representation of freeform shapes, we also need to consider the equally important aspect of physical fabrication of these shapes. Aided by cheap memory and the ease of digital data exchange between systems, we now have exceptional computer aided fabrications techniques. However, unlike other similar industries, the scale and material requirements of an architectural design brings a set of unresolved problems to the existing system of fabrication.

We aim to identify and address practical problems related to the virtual design representation as well as physical construction of freeform architectural shapes.

C. Discrete Representation of Freeform Shapes

Smart Geometry Group (SGG), a small collection of people from academia and construction industry, regularly conducts workshops and symposiums to push forward the state-of-the-art in computational tools for architectural design. Broadly speaking, the idea of optimizing geometry for any given discrete constraint falls in the area of *discrete differential geometry (DDG)* [AIB08, GS08, KHW05, Pol02]. Recently, a

new field of study has emerged from the intersection of DDG and architectural engineering called *architectural or smart geometry* [HPK07]. The idea behind this field is to computationally engineer smart geometric structures that conform to a new set of metrics defined by architectural design process.

Digital representation of a smooth shape requires decomposing the same into discrete parts. The spatial union of these parts approximate the underlying shape. This is an extensively studied problem in Computer Graphics [Far01]. Current CAGD systems can efficiently segment a smooth surface (implicit or parametric) into discrete triangulated representations. The triangles thus formed are nicely shaped and well suited for finite element simulations [She96, She01, She02b]. Since the end-goal for architectural tessellation is not to conduct simulations, these engineering based tessellation metrics may not apply to freeform architectural shapes.

Aesthetics are vital to architectural shapes since they are built on a large-scale and are visible to public scrutiny. This requirement guides a new set of metrics for shape rationalization. Defining aesthetic rationalization can be a subjective choice that varies from one project to another. The subjective nature of the problem makes it very transient and challenging. Some challenges may also be unique to a certain project such as packing a set of circles over a curved surface(example shown in Figure [SHWP09]). On the other hand, Carlos Séquin [SÓ5] defines aesthetics using quantifiable geometric attributes. He defines shape aesthetics by either using procedural rules or via shape optimization of constrained curves and surfaces with some associated global cost function.

Alternatively, rationalization of a shape can be guided by other factors such as structural stability, economic concerns or merely to simplify construction related complexities. Structural considerations are relatively well-defined mathematical problems with expected numerical tolerances. Examples of such problems can be planarization

Fig. 3. An example of triangulated (left) and quadrilateral (right) glass panels (©Drew Leavy (left) and Sakena (right)).

of polygonal panels [LPW*06, CW07], generating multi-layer geometry for a given shape [PLW*07], producing torsion free support structure for panels [SHWP09], or discretization of surface into developable parts [KFMP08, PSB*08]. The following chapter describes these problems and associated solutions in depth.

Dennis Sheldon [She02a] in his doctoral research first studied the problems associated with rationalization of freeform shapes in the context of various material properties. His discretization metric was guided by the analysis of mean curvature of the underlying shape. The analysis presented by him was closely tied to the manufacturing and construction tolerances of bendable metallic panels. This pioneering work presented by Dennis Sheldon was later used by the engineering team at Ghery Partners for the construction of Guggenheim Museum in Bilbao, Spain as shown in Figure 2.

Rationalization of surface can also be guided by the economics of the project. For example Eigensatz et al. [EKS*10] approximated a surface using a small set of molds. Each mold has an associated cost function. Planar panels are cheaper to manufacture and transport compared to spherical and cylinderical panels. Their method aims to balance two potentially conflicting goals - the aesthetic quality and the final cost of

the built shape. The method is flexible and the final decision can be left for the designer. Apart from using a set of molds for creating panels, the designer can also choose between using triangles, quadrilaterals or other n-sided polygons for the shape. Figure 3 shows an example of freeform surface using triangle (left) and quadrilateral (right) panels.

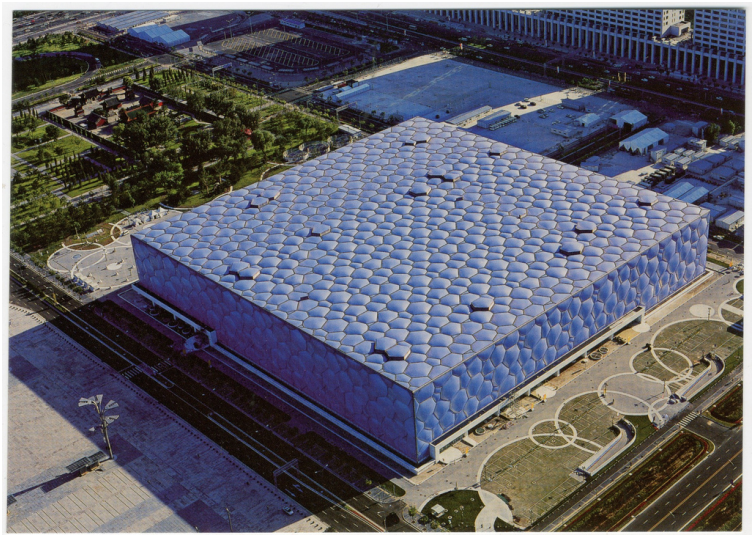


Fig. 4. Beijing Aquatic Center. An example where number of unique panels were greatly reduced(©neutralSurface).

Another example of cost cutting is the panelization of Beijing Aquatic Center in Beijing, China as shown in Figure 4. The initial design consisted of a set of randomly sized panels placed over the planar facade. However due to prohibitive cost of fabricating panels and complex construction process, the team was forced

---

Figure "The Great Courtyard of the British Museum" is copyright (c) 2007 by Drew Leavy available under a Attribution-NonCommercial-NoDerivs 2.0 Generic license (http://www.flickr.com/photos/drewleavy/786235580/) and Figure "Color changing roof" is copyright (c) 2010 by Sakena available under a Attribution 2.0 Generic license (http://www.flickr.com/photos/sakena/4832972895/)

Figure "National Aquatic Center (Water Cube)" is copyright (c) 2010 by neutralSurface available under a Attribution-NonCommercial-NoDerivs 2.0 Generic license (http://www.flickr.com/photos/jbergen/4521655448/)

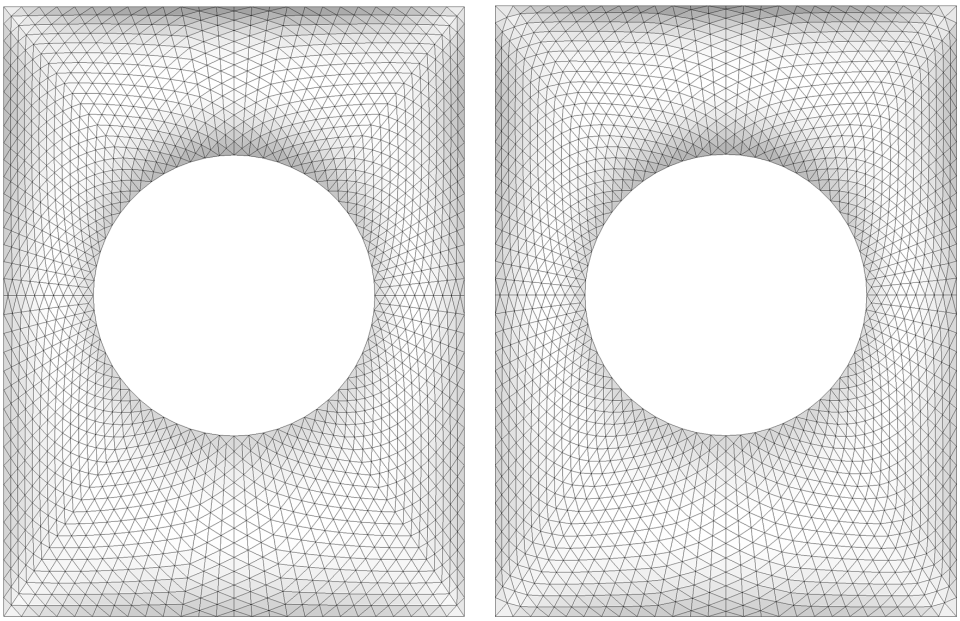
to redesign. The facade now consists of fewer number of unique panels repeated over the surface. This redesign reduced manufacturing cost as well as simplified the construction complexity.

Another principle defined for architectural aesthetics is the repetition in shape as mentioned by E. L. Garbett's [Gar67] treatise on the Principles of Design in Architecture. Repetition of panels for a planar surface is trivial. In past, famous architects such as Mies van der Rohe and other fellow modernists relied on repetition via geometric symmetry (rotational, reflective and translational). Construction experts have leveraged the idea of reuse, even when the parts had varied performance requirement. For example in case of a high-rise buildings the panels at the top would require more structural integrity than the ones at bottom. However in view of cost it is more effective to use the same panel everywhere.

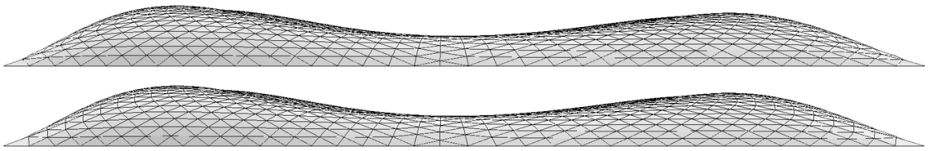
A successful real life example of using a computational engine to improve the aesthetics of a shape is the British Museum Great Court Roof. The roof is designed by the architectural firm Foster and Partners along with Buro Happold as the structural engineering firm. It was fabricated and installed by Waagner Brio. The roof consists of a triangulated grid with double glazed glass panels supported by steel members.

The circular inner boundary of the roof as shown in Figure 5 is supported by the Reading Room, while the rectangular outer boundary stands on the existing structure. This roof shape is generated by sampling points in x, y plane using polar coordinates. For each point a z value is derived using the summation of three different height functions. Profile of the shape is shown at the bottom of the Figure 5. This produces artifacts along the parameter lines (sharp turns) at the diagonals of the rectangular boundary.

Williams et al. [Wil01] suggested a relaxation based method that computes forces along the edges of the mesh. Each edge is treated as a spring in tension connecting



Top view of the mesh (input on the left and output on the right).



Side view of the mesh (input mesh on the top and output mesh on the bottom).

Fig. 5. Mesh for British Museum Court Roof.

vertices of the mesh. He is able to remove these artifacts by relaxing the mesh. Spring based relaxation methods are numerically unstable (as mentioned by the authors) and frequently explode unless the time step for integration as well as the tension coefficients for the spring are kept sufficiently small.

Inspired by the example of British Museum roof where the design team aim to create an aesthetic mesh, and the construction feasibility analysis done in case of Beijing Aquatic Centre we define the scope of our research as follows:

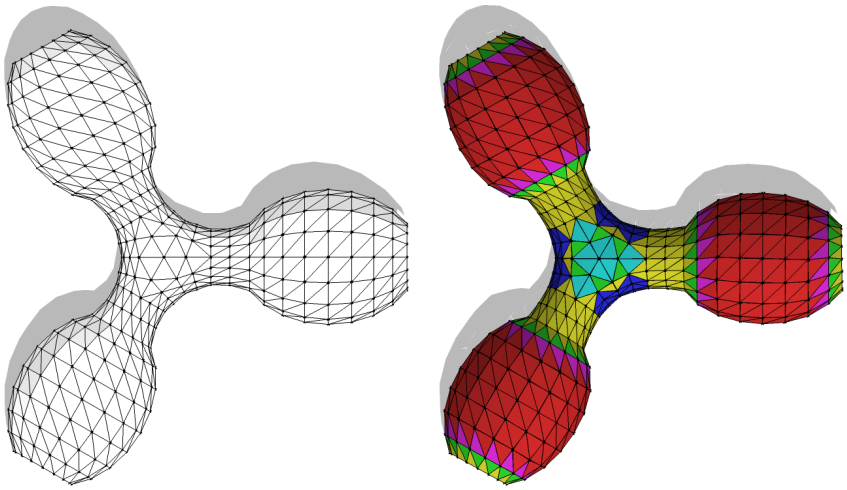
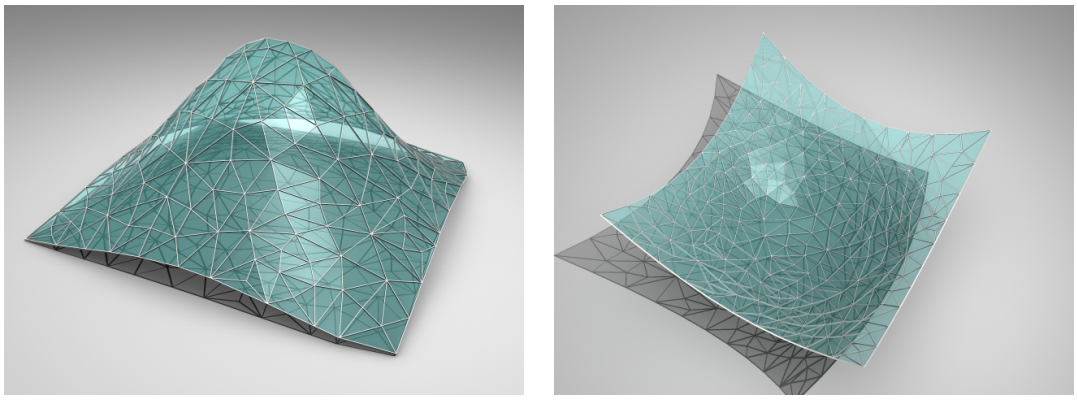


Fig. 6. An example of a dome shaped architectural design. Right: the black lines shows the rationalization of the original shape. Left: the same shape approximated using only six unique triangulated panels.

D. Objectives

- The first concern is how best to create an economically viable tessellation. Our goal is to approximate the input shape as best as possible using a very small number of unique triangles. Along with we ensure that the spacing or overlap between these set of triangles remains below prescribed tolerance levels. We begin the process of reducing the number of unique triangles by detecting clusters of similar triangles in the shape. For each cluster we compute a representative triangle. We then replace each triangle in the shape with the representative triangle of the cluster it belongs to. We do so while maintaining the output shape as similar as possible to the input shape. Figure 6 shows an example of a dome shaped architectural shape. The input shape is shown on the left and the optimized output shape on the right. Each cluster is depicted using a unique color.

- The second goal is to approximate the input shape using a set of triangles that



Gaussian Bump Paraboloid

Fig. 7. Example of simple shapes generated using our method for creating a set of locally similar triangles.

are similar in a local neighbourhood, thus creating an aesthetic tessellation for the shape. To do so we build an error metric that measures dissimilarity between triangles sharing a common edge. By doing so we can write an error function over the entire mesh. We then minimize this non-linear error function to create aesthetically pleasing triangles. Figure 7 illustrates the result of our method using a Gaussian bump and a Paraboloid shaped mesh.

CHAPTER II

PREVIOUS WORK IN RATIONALIZATION OF FREEFORM ARCHITECTURAL SHAPES

In the following chapter I will briefly cover the relevant background work in the optimization of free-form shapes.

A. Types of Rationalization

The choice of rationalization for a given shape resides with the chief designer of the project. For a given rationalization the engineering team produces a unique beam layout along with its structural requirements. Rationalization using triangles is a well studied problem in Computer Graphics. Triangulated panels are inherently planar, easy to fabricate and transport. Hambleton [DH09], while comparing triangle versus quadrilateral based rationalization, mentions that triangles provide the highest design flexibility at the cost of structural overhead of higher node complexity. Pottmann et al. [PLW*07] argue in favor of quadrilaterals over triangles based on its simplified node complexity, reduced cost per panel, lighter support structure, and torsion free nodes.

Quadrilateral discretization can be achieved using any one of the following methods [DH09] (a relative comparison of the following methods is provided in Table I):

- *Fitted Rotational Surface*: Translational and rotational surfaces are surfaces that are generated by translating or revolving one curve around another [section 1].

- *Principal Curvature Mesh*: Parametrization of surface using conjugate network of curves, typically principal curvature lines. Liu et al. [LPW*06] use such a

parametrized surface to generate a planar quad mesh [section 2].

- *Developable Strip Model*: Use one family of parameter lines to discretize surface into ruled strips with zero gaussian curvature. Kilian et al. [KFMP08] describe an optimization based method to generate corresponding planar quad mesh for an underlying developable surface [section 6].

Design Type	Node Simplicity	Structural Transparency	Design Intent	Material Efficiency
Triangulated Mesh	1	1	3.5	2.5
Quadrilateral Mesh	1.5	3	2	3.5
Fitted Rotational Surface	4	3.5	2	3.5
Principal Curvature Mesh	4	4	4	3.5
Developable Strip Model	3	4	4.5	2.5

Table I. Relative comparison of various types of surface rationalization as described by Hambleton [DH09].

B. Rationalization in Context to Structural Constructability

1. Rationalization using Gaussian Curvature

Dennis Shelden [She02a] first attempted to solve the problems related to construction of free-form shapes designed by Frank O' Ghery. He approached the problem of discretization by analysing the Gaussian curvature of the smooth surface in context to the flexibility of the material. He studied the structural limits of bendable metallic

panels for cladding the entire surface. Deforming sheets aid in accommodating given curvature, although this may come at the cost stretching the sheet in certain directions. The stretch leads to overlap and / or discontinuity amongst adjoining sheets. This discontinuity among sheets always lies within pre-established construction tolerance levels.

In cases where Gaussian curvature of the surface patch exceeds the prescribed material limits the shape is cut into disjoint pieces. The cutting is done such that the curvature of each of resulting piece is less than curvature of the original surface patch. Cutting the surface, or rationalization, reduces the limits on curvature, although this comes at a cost. As more pieces are introduced, fabrication cost rises. Tangential discontinuities introduced by rationalization also have aesthetic implications. Thus it is critical to have the designer input when introducing breaks in the surface. Highly rationalized surface will lead to smaller discrete pieces or panels. Higher number of panels will add to the complexity of physical placement of panels. Also, higher number of sheet pieces will require more attention to waterproofing strategies, since such cases may have poor inter-sheet connectivity.

2. Planarization of Panels and PQ-Mesh

In cases where the material of choice is non-bendable such as glass, the rationalization requires all resulting panels to be planar. Quadrangulation of freeform shapes produces a set of non-planar quadrilaterals.

Glymph et al. [GSC*04] used a very simple technique to generate planar quads from smooth surface via a set of conjugate curves called *Directrix* and *Generatix*. The directrix curves were translated along generatix without any rotation, thus producing planar quadrilaterals, which can be used to fabricated glass and steel panels. Figure 8 shows an example of a simple curved shape along with its corresponding directrix and

generatix curves.

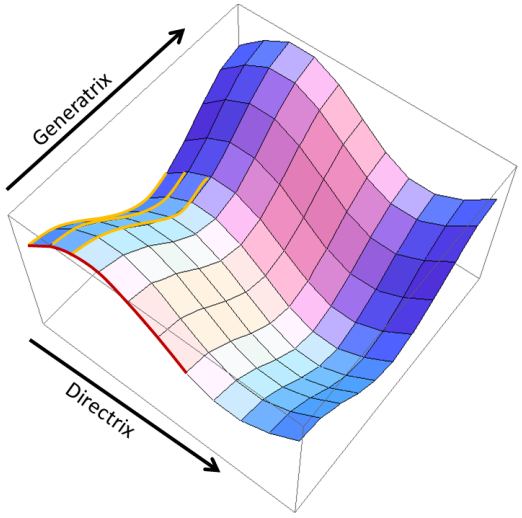


Fig. 8. Example of transforming a parametric shape into planar quadrilaterals using conjugate curves for installation of glass panels.

The need for planar panels come not only from limitations on fabrication methods but also for facilitating transport and storage. Planar panels, triangles or quadrilaterals, can be easily stacked on top of one another. Liu et al. [LPW*06] introduced a non-linear optimization based algorithm that would slightly perturb an input quadrilateral mesh and create a planar quadrilateral mesh (PQ mesh). In differential geometry the idea PQ meshes was first introduced by Sauer [Sau70], although its discrete counterpart came in much later.

A quadrilateral Q can be made planar by enforcing a simple constraint such that the sum of interior angles is equal to 2π. Additionally, Liu et al. [LPW*06] enforces an additional constraints to maintain proximity to the given reference surface. An important aesthetic consideration for architectural freeform shapes is maintaining a set of streamlines over the surface. Liu et al. [LPW*06] do not explicitly account for preservation of streamlines over the surface, they however have a Laplacian term that accounts for the fairness of the surface for their non-linear optimization. Finally,

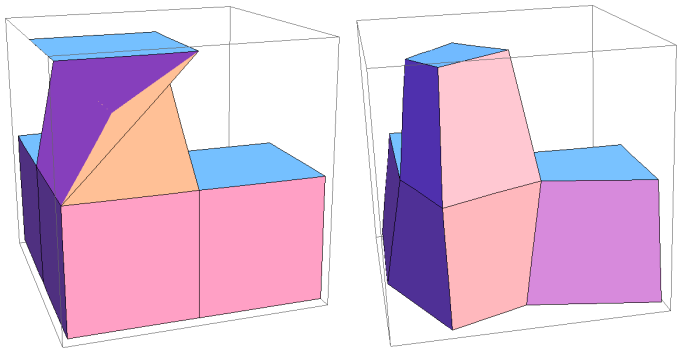


Fig. 9. Planarization of Quadilateral panels. The quadilateral mesh on the left is a planarized, resulting producing the mesh on the right hand side.

the function for minimization is expressed as a sum of angle constraints, closeness to original surface and laplacian term. Mathematically,

$$\begin{aligned} f_{PQ} &= f_{angle} + w_1 f_{laplacian} + w_2 f_{closeness} \\ f_{angle} &:= \sum_{i,j} (\phi^1_{i,j} + \cdots \phi^4_{i,j} - 2\phi)^2 \\ f_{laplacian} &:= \sum_{i,j} \left[(\mathbf{v}_{i+1,j} - 2\mathbf{v}_{i,j} + \mathbf{v}_{i-1,j})^2 + (\mathbf{v}_{i,j+1} - 2\mathbf{v}_{i,j} + \mathbf{v}_{i,j-1})^2 \right] \\ f_{closeness} &:= \sum_{i,j} \|\mathbf{v}_{i,j} - \mathbf{y}_{i,j}\| \end{aligned}$$

where $\mathbf{v}_{i,j}$ is unknown and $\mathbf{y}_{i,j}$ is the footpoint on the original surface Φ. w_1 and w_2 are user tunable coefficients. They also demonstrated the results of their work in context to hierarchical subdivision surfaces by alternating the use of Catmull Clark subdivision and applying PQ perturbation. Optimizing a quadrilateral mesh such that it forms a PQ mesh is highly dependent upon the input mesh. In order to produce aesthetic PQ meshes, rationalization of the surface should be done such that the network of discrete conjugate curves follow the principal curvature lines. Figure 9 shows an example of a simple quad mesh which when optimized has planar quadrilaterals.

3. Circular and Conical Mesh

As an extension to PQ meshes Liu et al. [LPW*06] describe another family of meshes known as Circular (in $2d$) or Conical (in $3d$) meshes (CP meshes). A subset of PQ meshes are referred to as *Circular meshes* if the planar quadrilateral in mesh have the circumcircle property [DMP93, AIB08]. A given vertex v of a quadrilateral mesh is a conical vertex if all four face planes touching v are tangent to a common sphere. This is equivalent to having all four faces tangent to a common oriented cone of revolution whose axis of revolution is collinear with the normal at the vertex v. Conical meshes can be generated from PQ meshes by adding another constraint to the optimization such that the opposite angles of the quad sum to π, i.e.

$$\omega^1_{i,j} + \omega^3_{i,j} - \omega^2_{i,j} - \omega^4_{i,j} = 0.$$

With the aid of various examples the authors demonstrate that the combination of subdivision and conical/circular perturbation produces high quality meshes that are suitable for aesthetic architectural design.

4. Multilayer Freeform Structures

A resulting implication based upon the choice of rationalization is how offset geometry is computed for a given shape. According to Pottmann et al. [PLW*07], computing an offset mesh for triangulated disretization is trivial (translation and scaling), however this is not true for a quadrilateral mesh. They also strongly advocate the use of quadrilateral meshes for free-form architectural structures citing the following reasons:

- A triangulated structure built using glass along with steel frame will have higher node complexity (six, as versus to four for a quadrilateral mesh).
- Cost per panel for triangulated panels is higher than quadrilaterals.

- Torsion free nodes are preferred for construction of steel beams. According to Pottmann et al. [PLW*07] torsion free nodes do not exist for triangle meshes.

- Disregarding trivial cases, triangle meshes do not posses offset at constant face-face or edge-edge distance.

Working with quadrilateral meshes, Pottmann et al. [PLW*07] states that two meshes $(\mathcal{M}, \mathcal{M}')$ are combinatorially equivalent if there is a $1-1$ correspondence between vertices and edges. Also the meshes are parallel if the distance between them is constant throughout the mesh. In a discrete setting, parallel meshes can lead to three different variations - vertex, edge, and faces offsets meshes. In each case the distance between corresponding elements of $(\mathcal{M}, \mathcal{M}')$ will be a constant value d. An alternative description of offset meshes is achieved via using discrete Gauss image mesh(S).

$$\begin{aligned} \mathcal{S} &= \frac{(\mathcal{M}' - \mathcal{M})}{d} \quad \text{whose vertices} \\ s_i &= \frac{(\mathbf{m}_i - \mathbf{m}_i)}{d} \quad \text{can be regarded as discrete normal vectors} \end{aligned}$$

Now, for

- $\mathcal{M}'$ to be a vertex offset mesh of $\mathcal{M} \iff$ the vertices $\mathcal{S}$ lie on the unit sphere S^2. Circular meshes hold this property.

- $\mathcal{M}'$ to be a edge offset mesh of $\mathcal{M} \iff$ the edges of $\mathcal{S}$ are tangent to S^2.

- $\mathcal{M}'$ to be a vertex face mesh of $\mathcal{M} \iff$ the faces of $\mathcal{S}$ are tangent to S^2. Conical meshes hold this property.

In order to simultaneously achieve all of the above conditions in a discrete setting Pottmann et al. [PLW*07] suggest minimizing the following function F_{offset}. Note

that this only produces an approximately offset mesh.

$$
\begin{aligned}
F_{offset} &= \lambda_1 f_{laplacian}(\mathcal{M}) + \lambda_2 f_{faces} + \lambda_3 f_{vert} + \lambda_4 f_{close,1} + \lambda_5 f_{close,2} + \lambda_6 f_{par} \\
f_{laplacian}(\mathcal{M}) &= \sum_{vertices\ \mathbf{m}_i} (\mathbf{m}_i - \frac{1}{deg(\mathbf{m}_i)} \sum_{\mathbf{m}_j \in star(\mathbf{m}_i)} \mathbf{m}_j)^2 \\
f_{faces} &= \sum_{faces\ F \in \mathcal{M}} (\mathbf{n}_F \cdot (\mathbf{s}_{i(F)} - \mathbf{n}_F))^2 \\
f_{vert} &= \sum_{vertices\ \mathbf{m}_i} (\mathbf{s}_i - \tilde{\mathbf{n}}_i)^2 \\
f_{close,1} &= \sum_i dist(\mathbf{m}_i, \Phi)^2 \\
f_{close,2} &= \sum_i \|\mathbf{s}_i\|^2 \\
f_{par} &= \sum_{edges\ \mathbf{m}_i, \mathbf{m}_j} \| \frac{\mathbf{m}_i - \mathbf{m}_j}{\|\mathbf{m}_i - \mathbf{m}_j\|} \times (\mathbf{s}_i - \mathbf{s}_j) \|^2.
\end{aligned}
$$

The fairness term $f_{laplacian}$ produces edge offset properties for a given mesh $\mathcal{M}$. For the face offset, the function f_{faces} minimizes the angle difference between normal of the face and its corresponding set of vertices on the Gauss image. The vertex offset function f_{vert} minimizes the difference between the vertex on Gauss image and the corresponding vertex normal ($\tilde{\mathbf{n}}_i$) on input mesh $\mathcal{M}_i$. $f_{close,1}$ and $f_{close,2}$ maintain proximity to the input mesh($\mathcal{M}$) and its Gauss image(S) respectively. f_{par} expresses the parallelism of input mesh($\mathcal{M}$) and its Gauss image(S). Finally, $\lambda_{1,2,3,4,5,6}$ are coefficients to manipulate the weight of each function. This non-linear system is solved using Gauss-Newton method with Levenberg–Marquardt (LM) regularization.

5. Freeform Surfaces from Single Curved Panels

In order to simplify the construction of architectural freeform shapes, Pottmann et al. [PSB*08] introduced the use of *semi-discrete* surfaces. These semi-discrete surfaces are continuous in one direction and discrete in the conjugate direction, i.e. the surface has continuous parametrizations in one direction and discrete in the other. The

surfaces are generated using an optimization based framework in conjugation with a B-spline representation.

A coarse PQ mesh under refinement produces a conjugate curve network. However, if the PQ mesh is refined in only one direction it produces developable strip model (*d-strip*). A D-strips model when joined together along edge curves $\mathbf{p}_i(u)$ can be interpreted using a b-spline

$$\begin{aligned} x_i(u,v) &= (1-v)\mathbf{p}_i(u) + v\mathbf{p}_{i+1}(u) \\ \mathbf{p}_i(u) &= \sum_j B^3(u-j)\mathbf{b}_{i,j} \end{aligned}$$

where parameter v is parallel to the ruling on the strip and u in the conjugate direction. B is the b-spline basis function for integer knots. $\mathbf{b}_{i,j}$ is the control point closest to the point $\mathbf{p}_i(j)$ on the curve.

In order to approximate a given surface Φ by a D-strip model, the control points are subjected to a non-linear optimization, described as follows:

$$\lambda_1 f_{prox} + \lambda_2 f_{boundary-prox} + \lambda_3 f_{dev} + \lambda_4 f_{fair-edge} + \lambda_5 f_{fair-rulings}$$

This minimization function accounts for closeness to Φ, closeness to the boundary curve of Φ, developability of the strips, and the fairness of the optimized surface. The proximity terms compute the distance to the tangent plane and the boundary of Φ. Developability ($\lambda_3 f_{dev}$) along a strip is maintained by keeping the quad planar. Fairness along the edges of a developable strip is maintained by minimizing the bending energy (or in this case the second derivative).

$$\lambda_4 f_{fair-edge} = \sum_i \int \|\ddot{p}_i(u)\|^2 du,$$

And fairness of rulings is achieved by minimizing via discrete differencing

$$\lambda_5 f_{fair-rulings} = \int (\sum \|\mathbf{p}_{i+1} - 2\mathbf{p}_i + \mathbf{p}_{i-1}\|) du.$$

6. Curved Folding

A developable surface is a ruled patch that can either be planar or curved (cylinder, cone or tangent surfaces). All developable surfaces have a property of zero curvature in the direction of the rules, thus producing a zero gaussian curvature. The freeform shapes of Guggenheim Museum in Bilbao, Spain in Figure 2 is an examples of curved developable surface.

Inspired by idea of approximating a freeform surface with a sequence of ruled surfaces, Kilian et al. [KFMP08] introduced an optimization based computational framework for the design and construction of developable surfaces using PQ meshes. Their method practices a minimization that alternates between maintaining planarity of quadrilaterals and generating minimally bending rules for a developable patch based upon user defined creases. To do so, the authors define bending energy that encapsulates curvature in the direction perpendicular to the rules, using discrete differences of normals. The method also estimates appropriate direction of rules based upon curvature of the patch. As a measure of convergence the optimization stops when the distance between patches is below a user-defined threshold.

Mathematically a vertex is considered developable if the distance between points on its geodesic circle and euclidian circle lie within a given threshold. The proposed method estimates rulings based upon predominant curvature direction. Ridges and valleys [OBS04] are marked as curve folds, or boundaries for developable parts. Ruling direction computed at each vertex is extended until it hits a crease (or boundary). Once the direction of rulings is determined for a given patch, it is flattened into a

Fig. 10. Selfridges Building, Birmingham exemplifying the use of circle packing (©Welshdan).

plane, using as rigid as possible parametrization. This planar patch is now used to generate an initial coarse quad dominant mesh.

The shape of the planar faces (P) and the position and orientation of corresponding polygonal soup (M) are optimized in an alternating manner. Optimization converges once the spacing between vertices of M are within a prescribed threshold. Optimization for M minimizes the following function:

$$\text{minimize } F = F_{vert} + \lambda F_{fit} + \mu F_{fair}$$

where F_{vert} is vertex laplacian, F_{fit} approximates the input surface and F_{fair} minimizes the bending energy of the developable patch. The bending energy is computed using the curvature of the patch (E_{bend} = difference of normals in gauss image / distance between vertices).

7. Surface Packing Design

Inspired by the circle packing like arrangement in the facade of Selfridges Building, Birmingham as shown in Figure 10 , Schiftner et al. [SHWP09] introduced the concept of Circle Packing (CP) meshes. CP-meshes exhibit the property of aesthetic triangles (more or less equilateral) along with producing a torsion free structural support for panel inserts. The idea behind CP-mesh is to have the incircle of a triangle touching the incircles of the adjacent triangles at a common point along the shared edge. For a given pair of triangles v_1, v_2, v_3 and v_1, v_3, v_4, we can achieve circle packing if,

$$l_{12} + l_{34} = l_{23} + l_{14}$$

where l_{ij} is the edge length between vertex v_i and v_j. Using this simple constraint along with maintaining proximity to the input reference mesh, Schiftner et al. [SHWP09] suggesting using a nonlinear optimization to transform an ordinary triangulated mesh into a CP-mesh. The optimization function is stated as:

$$F = \lambda_1 f_{cp} + \lambda_2 f_{prox} + \lambda f^{\partial}_{prox}$$

where f_{cp} measures the incircle packing property, f_{prox} maintains the input shape, and f^{∂}_{prox} maintains the boundary to the input shape.

$$f_{cp} = \sum_{\text{interior edges } \mathbf{v}_i, \mathbf{v}_j} (l_{ik} + l_{jm} - l_{kj} - l_{im})^2$$

where $(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k)$ and $(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_m)$ are faces adjacent to the edge $\mathbf{v}_i, \mathbf{v}_j$. Alternatively, the radii for incircle can be precribed and the constraint condition can be rewritten

---

as:

$$f_{bp} = \sum_{\text{interior edges } \mathbf{v}_i, \mathbf{v}_j} (\|\mathbf{v}_i - \mathbf{v}_j\| - r_i - r_j)^2$$

Derived Structures: Apart from producing aesthetic meshes, CP triangulated meshes produce torsion free beam layout. Dual of CP-meshes leads to torsion free hexagonal meshes. The hexagons can be filled by inserting a vertex in the center and splitting the face such that it has three quads. Subsequent optimization can then move vertices in such a manner that it penalizes non-planarity of quads. CP-meshes can also be used to generate another aesthetic variant called *tri-hex meshes*. As stated in their explanation, the main limitation of this process is that the optimization can succeed only if the mesh is topologically equivalent to a sphere or disk, and fails for surface with higher topological complexity.

8. Discretization of Freeform Surface into Finite Set of Panels

Rationalization of any freeform surface will produce a discrete set of panels. Physical construction of thus generated panels could be prohibitively expensive if each panel would require a custom mold. Eigenstaz et al. [EKS*10] suggested an optimization based framework that would minimally deform the shape such that each panel would be a part of a small subset of predefined molds, thereby reducing the production cost of panels. According to Eigenstaz et al. [EKS*10] each panel would fall in one of the following categories : planar, cylindrical, paraboloid, torus patch or generic cubic patch. Obviously, each mold will have an associated cost.

Any given freeform surface F can be approximated with the union of $\mathcal{P} = \{P_1, \cdots, P_n\}$ panels. According to Eigenstaz et al. [EKS*10] the quality of approximation depends upon two critical factors - *divergence* (spatial gap between panels)

and *kink angle* (jump in normal across adjacent panels). Let us assume that we have $\mathcal{M} = \{M_1, \cdots, M_m\}$, molds where m is significantly less than n, along with an assignment function A that assigns each one of the n panels to one of the m molds. The assignment function ($A : [1, n] \rightarrow [1, m]$) consists of a distance metric and an associated rigid transformation. Each mold can an associated cost function $c(M_k)$. The total cost of manufacturing the freeform shape can computed as follows:

$$cost(F, \mathcal{P}, \mathcal{M}, A) = \sum_{k=1}^{m} c(\mathcal{M}_k) + \sum_{i=1}^{n} c(M_{A(i)}, P_i).$$

Entire framework alternates between discrete (local) and continuous (global) optimization steps. The local optimization computes the assignment function A while the global optimization slowly deforms the underlying surface to match the assigned panels.

In order to minimize the error when using the assigned panels, the non-linear global optimization attempts to minimize the following: deviation from the input surface, the divergence from the curve network, the kink angle between panels, fairness of the curve network, and panel centering. Mathematically, it can be expressed as linear combination of above mentioned constraints:

$$E = \alpha_{fit} E_{fit} + \alpha_{div} E_{div} + \alpha_{kink} E_{kink} + \alpha_{fair} E_{fair} + \alpha_{cen} E_{cen}$$

Each one there terms are further described as follows. E_{fit} refers to the deviation of curve network from the reference input surface.

$$E_{fit} = \sum_{l=1}^{L} \|\mathbf{c}_l - \mathbf{f}_l\|^2,$$

where $\mathbf{c}_l$ are the sample points on the curve network $\mathcal{C}$ and $\mathbf{f}_l$ is the closest point on F from $\mathbf{c}_l$. L is the number of samples used. Constraint for minimizing divergence between the curve network samples and adjacent aligned mold surfaces is expressed

as

$$E_{div} = \sum_{l=1}^{L} \|\mathbf{c}_l - \mathbf{x}_{i(l)}\|^2 + \|\mathbf{c}_l - \mathbf{x}_{j(l)}\|^2$$

$\mathbf{x}_{i(l)}$ is the closest point to $\mathbf{c}_l$ on the aligned mold surface $M^*_{i(l)}$. Point $\mathbf{x}_{j(l)}$ is defined analogously. The condition for tangential continuity (or kink angle) is described as:

$$E_{kink} = \sum_{l=1}^{L} \|\mathbf{n}(\mathbf{x}_{i(l)}) - \mathbf{n}(\mathbf{x}_{j(l)})\|$$

where $\mathbf{n}(\mathbf{x}_{i(l)})$ is the normal vector of the aligned mold $M^*_{i(l)}$. Although E_{fit} may keep the curve network close to the underlying surface, it may cause undesirable undulations and large tangential motion of the boundary curves. A fairness term (E_{fair}) is added to account for normal displacement.

$$E_{fair} = \sum_{(j1,j2)\in C} (d_{j1} - d_{j2})^2$$

where $(j1, j2)$ is an index pair denoting an edge of the polygonal representation of the curve network $\mathcal{C}$. Along with maintaining quality curve network, we would like to center the panel. Thus an additional term for centering panels is added as

$$E_{cen} = \sum_{i=1}^{n} \|\mathbf{b}_i - \mathbf{p}_i\|^2$$

Here, $\mathbf{b}_i$ is an approximation of the barycenter of the segment S_i computed as the average of all the adjacent curve segment samples, and $\mathbf{p}_i$ is the projection of $\mathbf{b}_i$ onto the surface normal at the center of the aligned mold $M^*_{i(l)}$.

An exhaustive search for an assignment function A can be a critical bottleneck. This search lies in non-linear space. In order to ease the computational complexity, the authors introduce a $6D$ function that computes the distance between a panel and a mold type.

9. Equivalence Classes for Quadrilaterals

The method described by Eigenstaz et al. [EKS*10] reuses a small subset of molds to generate non-congruent panels. A logical extension is to be able to take advantage of economy of scale and design a set of congruent panels that can cover the entire surface. This is slightly more difficult since it would require prior knowledge of size of panels as well as the number of unique panels.

Fu et al [FLHCO10] resolved this problem using non-planar quads. They initiate their non-linear global optimization with large number of unique panels (equal to the number of quads in the shape) and reduce the number such that the spacing between panels lie within construction tolerance. The entire optimization can be termed as a linear combination of following terms:

$$minF = \mu_1 F_d + \mu_2 F_f + \mu_3 F_e + \mu_4 F_a + \mu F_o$$

where F_d is distance to original surface term that helps in closely approximating the input shape, F_f is a laplacian fairness term (common to all such optimization techniques), F_e minimizes the difference between edges lengths of adjacent panels, F_a is minimizes the difference between the sum of lengths of diagonals of the quad to the canonical quad and finally F_o is a signed volume term used to maintain the final shape of non-planar quad as close as possible to the canonical quad. The canonical quads are computed as an average of all the quads in the same cluster.

CHAPTER III

EQUIVALENT CLASSES FOR TRIANGLE MESHES

We aim to simplify the fabrication process for a given architectural freeform shape. Unlike the set of methods described in Chapter II our method operates only upon triangle rationalization. We do so since triangles are inherently planar, easier to fabricate from a single mold and can be stacked on top of each for ease of transportation. Figure 11 shows an example of freeform architectural surface approximated using only six unique triangles (originally the surface has 576 triangles). They also provide the designer with utmost design flexibility as argued by Hambleton [DH09].

We reduce the fabrication complexity by approximating a given shape with a reduced set of unique planar panels. We use a global linear optimization based method to generate a small set of congruent panels that approximate a given triangulated mesh. Figure 12 provides a brief overview of the entire method. We begin by detecting a set of triangles that are similar to each other. Each triangle in the set is assigned to a single cluster. There can be more than one cluster of similar triangles. For each one of such triangle clusters we compute a representative triangle. This representative or canonical triangle replaces all the triangles in the corresponding cluster. Thus creating a mesh of disconnected and/or overlapping canonical triangles. We use a Poisson based global optimization to reduce the spacing and overlap between canonical triangles. We continue this process of canonical triangle computation and surface optimization until the error falls below construction tolerance level.

Tiling or approximating a shape with repeated units is a well studied problem, computationally and otherwise. Grünbaum, Branko and Shephard [GS86] presented the combinatoric theory behind 2D periodic tiling. Their example set was limited to only planar patterns. Craig Kaplan in his doctoral dissertation [KS04] presented

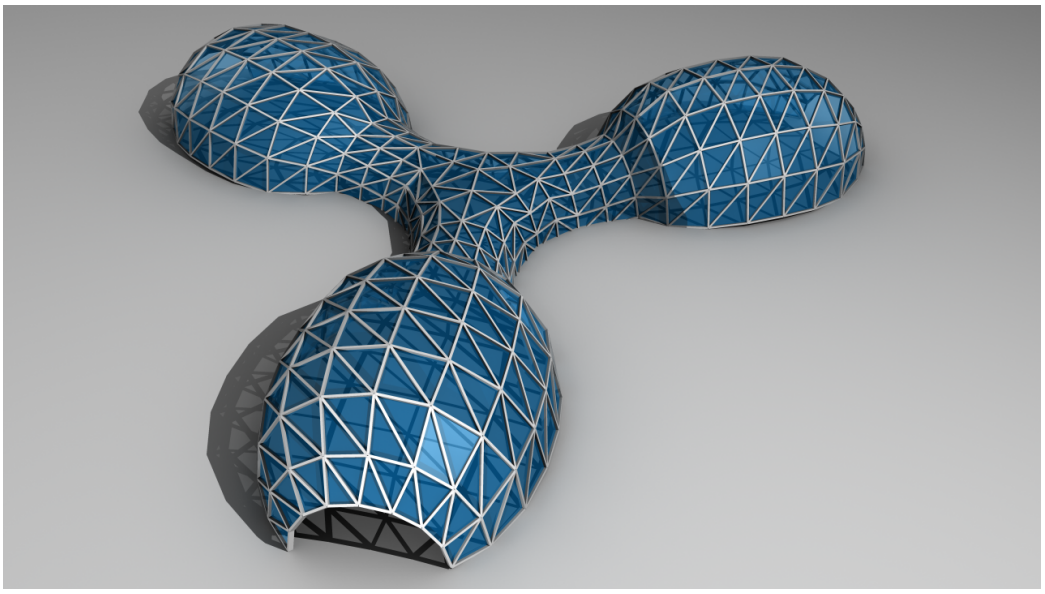


Fig. 11. Example of a dome shaped architectural roof originally consisting of 576 triangles. The shape shown above is approximated with only 6 unique triangles.

an interactive computational engine to create a variety of planar Islamic patterns. Creating 2/3-D mosaics is also a relevant body of work. Kim and Pellacini [KP02] first presented a technique to auto-generate 2D mosaics by covering the entire image with repeating units. Elber and Wolberg [EW03] improved upon 2d mosaics by taking into account the curvature lines. The concept of using tiles are extended to 3D by Lai et al. [LHM06]. They used a set of equi-size quadrilateral pieces along with grout to cover the entire 3D surface. Their approach even though produces aesthetic mosaics uses significant amount of grout. This may not be feasible for building construction purposes. Passos and Walter [DPW08] extended this work so as to use variable sized tiles. Even this approach requires significant amount of grout to hold the panels in place.

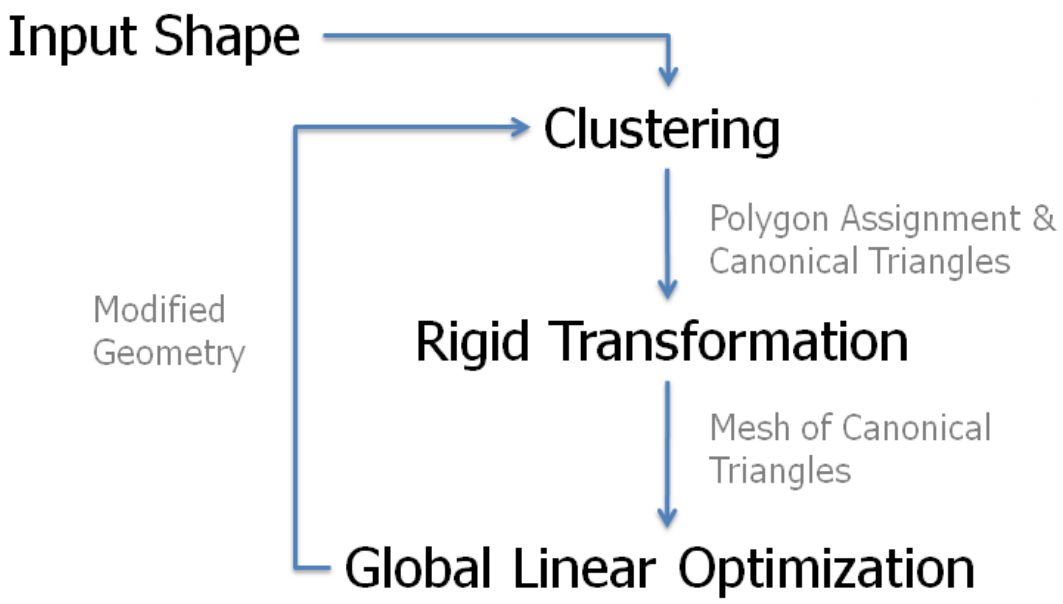


Fig. 12. Overview for creating a set of equivalence classes.

A. Distance Between Two Triangles

We begin with a closed or open (with boundary) triangle mesh along with an initial specification of number of clusters (n) and construction tolerance (ϵ). The ϵ-tolerance is specified as a percentage of the diagonal of the bounding box of the mesh. We aim to approximate the shape within the specified tolerance limit using n-distinct equivalence classes of triangles.

In order to compute clusters of similar triangles we need to define a similarity metric between two triangles. Given two triangles A and B with vertices a_1, a_2, a_3 and b_1, b_2, b_3 respectively we measure similarity as the sum of distance between the corresponding vertices. However before we can measure the distance between corresponding vertices we need compute rigid transformation that maps triangle A onto

---

Rigid transformation consists of rotation and translation. It maintains the angles and length of edges of the triangle under transformation.

the triangle B. Apart from computing the rigid transformation we also consider the set of six possible correspondences for mapping the vertices of triangle A to B. We select the correspondence that best minimizes the distance between the two triangles. The six correspondence include rotation as well as reflection of the triangle. Fu et al [FLHCO10] also used a similar metric for measuring the distance between two quadrilaterals.

Mathematically the distance between the two triangles P and Q is expressed as

$$D(P,Q) = \min_{R^T R=I,T,j} \sum_{l=1}^{3} |RP_{perm(j,l)} + T - Q_l|^2 \tag{3.1}$$

where R and T represent the rigid transformation. $perm(j,l)$ represent the l^{th} element of the j^{th} permutation of the indices $\{1,2,3\}$. We compute the best rigid transformation R and T using the method proposed by Arun et al [AHB87], where the translation is given by

$$T = \bar{Q} - R\bar{P}$$

where $\bar{P}$ and $\bar{Q}$ represent the centroids of the two triangles. We now represent the vertices as

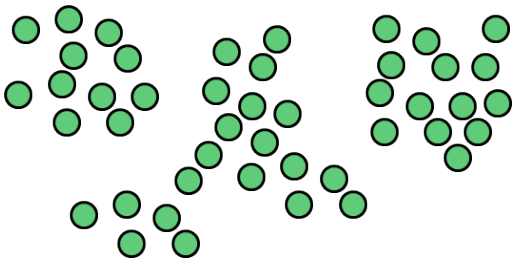
$$\begin{aligned} P_l^* &= P_l - \bar{P} \\ Q_l^* &= Q_l - \bar{Q} \end{aligned}$$

and define M to be

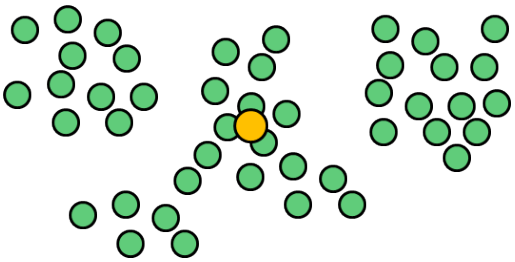
$$M = \sum_{l=1}^{3} P_l^* (Q_l^*)^T.$$

The rotation matrix R is given by

$$R = UV^T$$

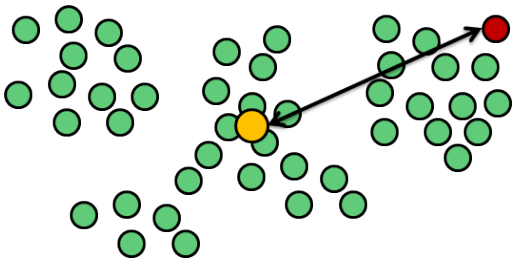


1: All points are assigned to a single cluster.

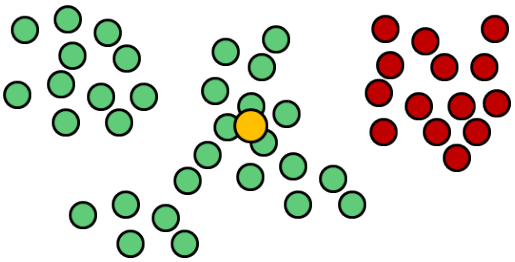


2: Using non-linear search we compute the canonical triangle for this cluster.

Fig. 13. First and second steps in clustering points.



3: Now we select the point that has the maximum distance from the center (canonical) of the cluster and assign it to a new cluster.



4: Recompute the distance of all points to the center of two clusters. Assign each point in the space appropriately.

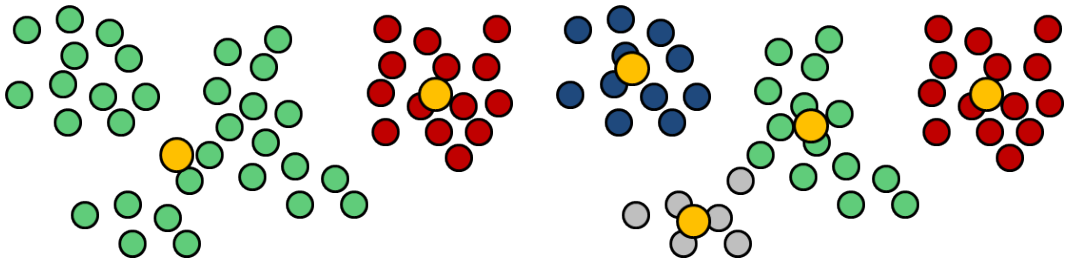
Fig. 14. Third and forth steps in clustering points.

where

$$M = U\Sigma V^T$$

is the singular value decomposition of M. If $Det[M] < 0$ we negate the vector in V corresponding to the smallest singular value.

This distance metric $D(P, Q)$ is symmetric in P, Q and has the property that if one is the rotation or reflection of the other, the distance is returned as zero. If the panels need to be oriented then we can drop the three correspondences that account of reflection of triangles.



5: Now we have split the space into two clusters.

6: We repeat this process of cluster generation until pre-defined number of clusters are created.

Fig. 15. Fifth and sixth steps in clustering points.

B. Clustering of Triangles

Now that we understand the distance metric between two triangles, we can detect a set of similar triangles and assign them to a single cluster. Once the mesh is partitioned into discrete clusters we can represent each cluster with a representative triangle referred to as *canonical triangle*. This canonical triangle attempts to minimize the squared sum of distance to all the triangles in the set. The function is expressed as

$$\min_{C_j, ind} \sum_i D(P_i, C_{ind(i)}) \tag{3.2}$$

where C_j is the canonical triangle for the j^{th} cluster and $ind(i)$ refers to the cluster that triangle P_i is assigned.

In order to identify clusters of similar triangles we use a well-known clustering algorithm called adaptive k-means clustering [AV07]. Figure 13, 14, and 15 illustrates the k-means clustering process via circles that represent triangles in $3d$ space. The clustering process is preformed 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.

The k-means clustering requires computation of cluster center. In our case the

center is represented by the canonical triangle. The distance metric in equation 3.1 is a highly non-linear function. Thus we employ a non-linear optimization using the Levenberg-Marquardt algorithm [FJNT04] to search for canonical triangle C_j for each cluster. The canonical triangle is a reference frame independent triangle. Instead

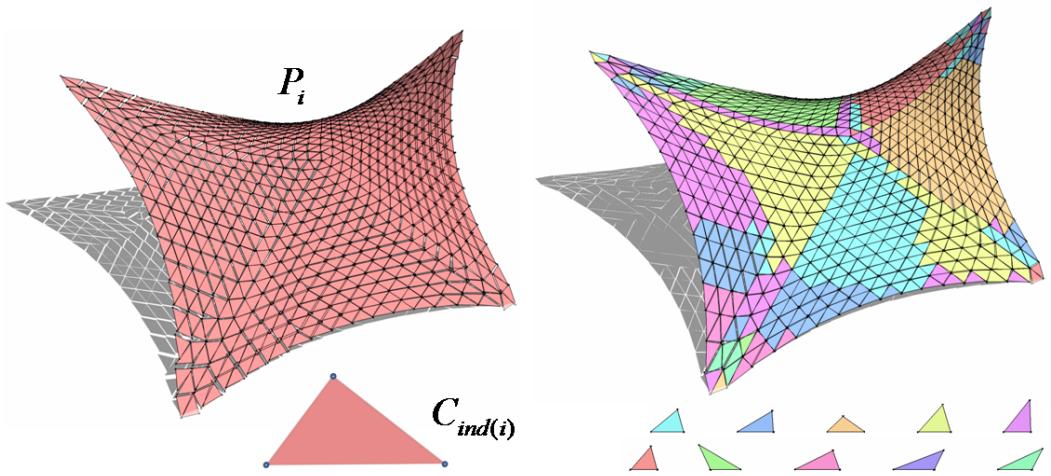


Fig. 16. A 5-Point tensile membrane with varying number of clusters. On the left hand side the entire shape is approximated with a single cluster. Beneath the shape is the single canonical triangle. On the right is the same shape with 10 clusters. The 10 canonical triangles are shown at the below the shape.

of nine-floating point numbers we represent the triangle using only three degrees of freedom. Now even though the search for canonical triangle is non-linear, our search space is limited to three unknowns, thus making the problem computationally tractable. There are multiple ways to represent a triangle with three degrees of freedom. We choose to do so using the vertices of the canonical triangle C_j as follows:

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

Typically k-means clustering begins with random seeds. However since we use the

variant proposed by [WYZG09], the process of clustering is deterministic and tends to produce better results. We begin our non-linear search 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 3.1 and repeat this process until n clusters have been added. Figure 16 demonstrate the results of varying clusters using a 5-point tensile roof. As shown in Figure Figure 17 the error drops rapidly as initial clusters are created but slows down significantly as more clusters are added. The spacing and other overlap between canonical triangles is significantly reduced as the number of clusters is raised from 1 to 10. Figure 18, 19 and 20 show similar results.

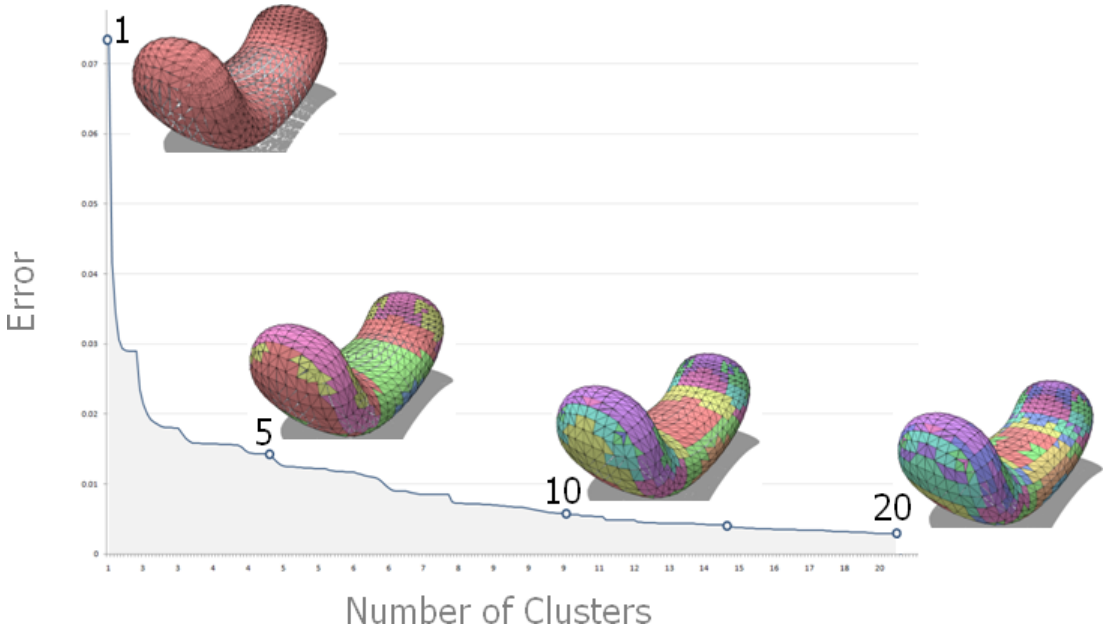


Fig. 17. A graph showing how the error drops as we increase the number of clusters. Note that the rate of error drop diminishes very rapidly after the 5^{th} cluster is created.

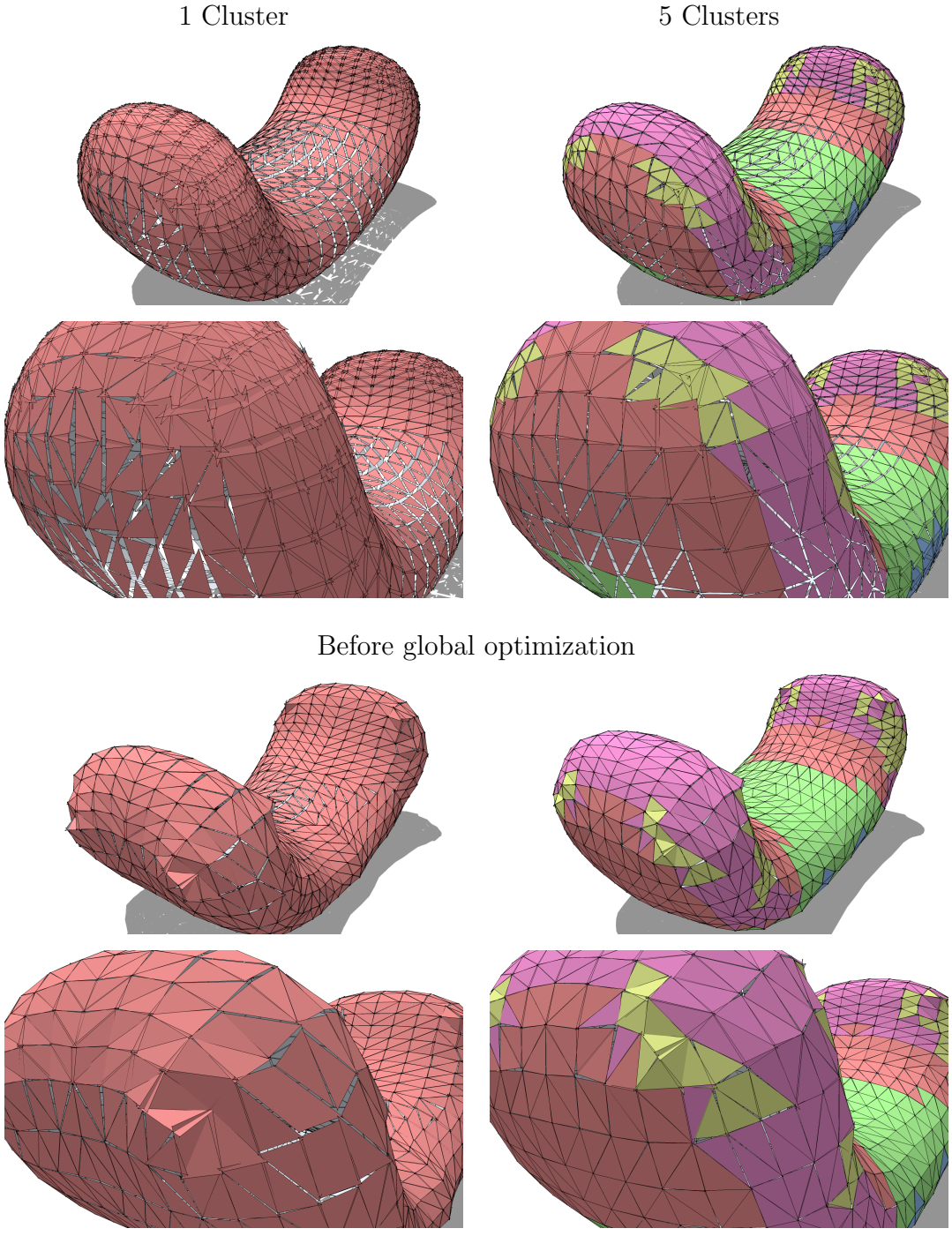


Fig. 18. Bean with 1 cluster on the left and 5 clusters on the right size both before and after optimization.

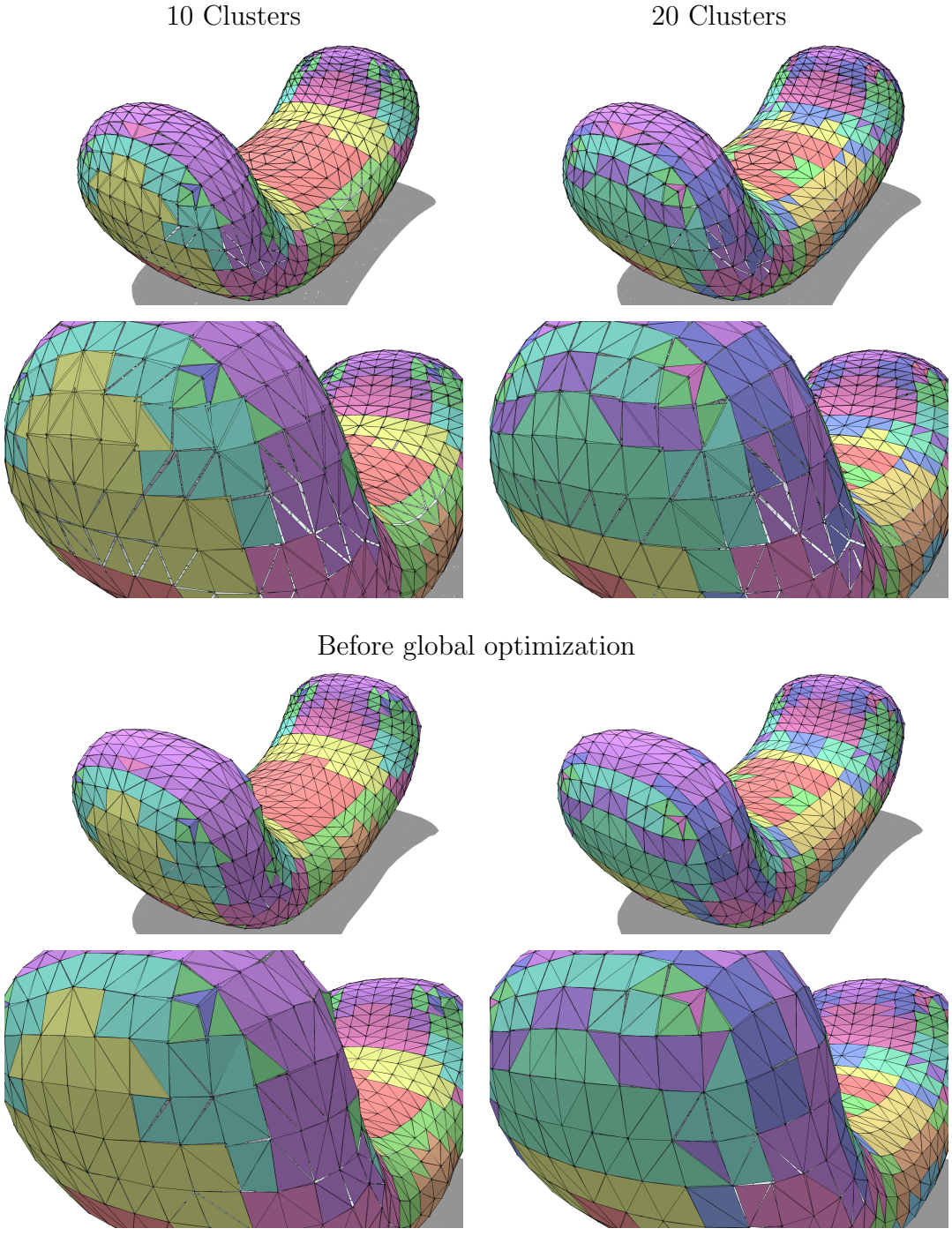


Fig. 19. Bean with 10 clusters on the left and 20 clusters on the right size both before and after optimization.

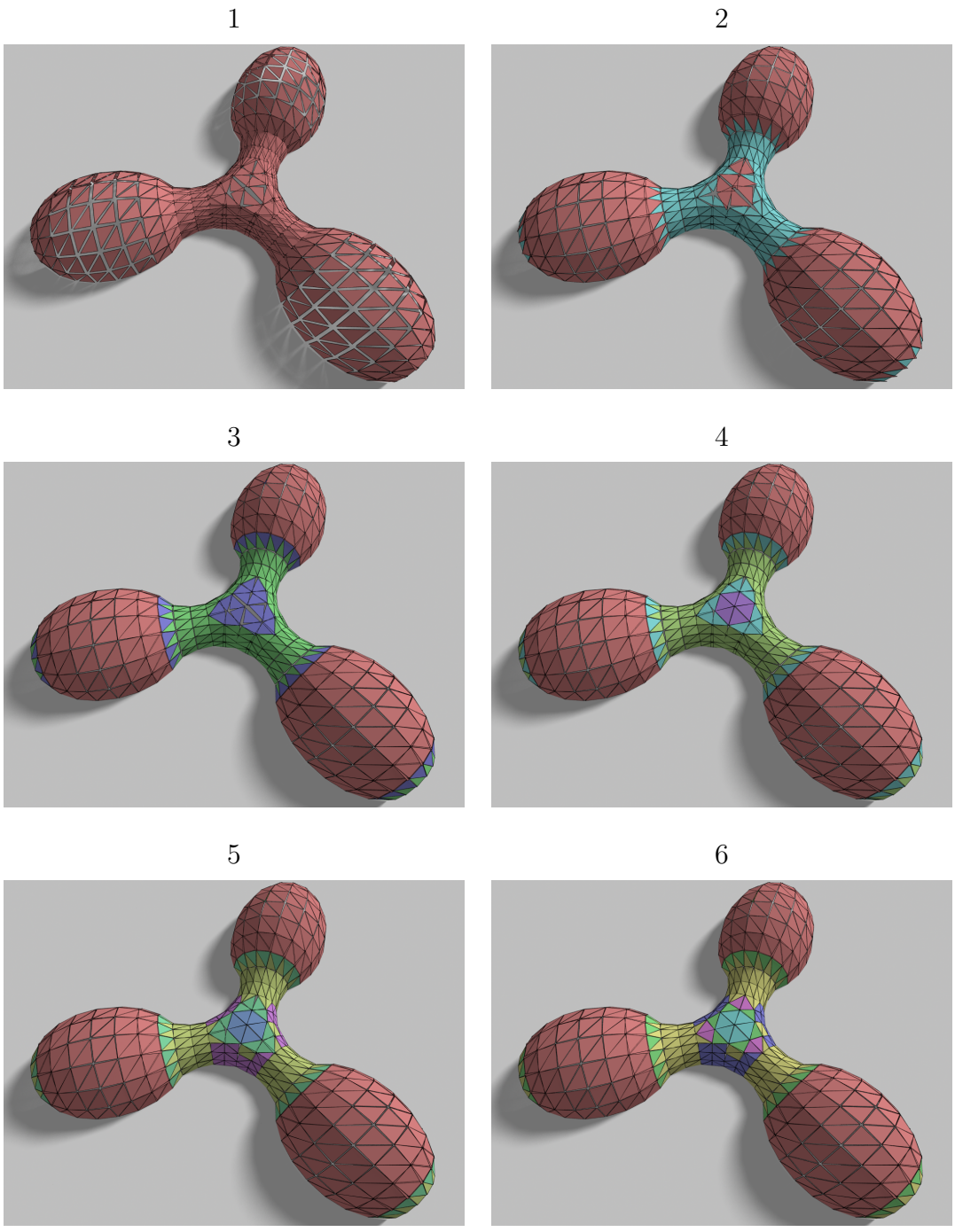


Fig. 20. Clustering for architectural roof. Clockwise from top shows the number of clusters in color along with the shape. Note that as the number of clusters grow, the spacing and overlap between triangles diminishes.

C. Global Optimization

Using the rigid transformation described in equation 3.1 we can replace each triangle in the mesh with its corresponding canonical triangle. This produces a disconnected mesh of canonical triangles, where the number of unique triangles is equal to the number of clusters. The spacing and overlap between transformed canonical triangles diminishes as we raise the number of clusters. Theoretically, the error would be zero if each triangle is assigned to its own cluster. However we aim to reduce this spacing and overlap between canonical triangles while keeping the number of clusters as low as possible.

1. Poisson Based Optimization

For a given assignment of triangles to clusters and the transformed canonical polygons $\hat{C}_{ind(i)}$ we attempt to a find a surface whose triangles match the canonical triangle mesh. Yu et al. [YZX*04] tackled a very similar problem in context of mesh deformation. They solved the problem using a global Poisson based optimization. We follow the same approach by minimally deforming the input mesh such that the gradient of triangle P_i matches the gradient of transformed canonical triangle $\hat{C}_{ind(i)}$. The Poisson based optimization can be expressed as a function that minimizes the sum of difference in gradients over the entire mesh.

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

where ΔP_i is the gradient of triangle P_i and Δ_i is the area of P_i.

2. Surface Proximity

As our global optimization attempts to reduce the difference in gradients, we would also like to approximate the input shape as closely as possible. Thus we define a proximity metric that penalizes the global optimization if the optimized surface deviates from the input mesh. This proximity metric also aids in avoiding a trivial zero solution. We build this proximity metric by computing a footpoint for each vertex in the mesh. For each triangle P_i of the current surface we compute the closest point ($\mathbf{x}_i$) and normal ($\mathbf{n}_i$) on the input shape from the centroid of triangle P_i. The proximity error is then defined as

$$E_c = \sum_i \left(\mathbf{n}_i \cdot (\mathbf{p}_i - \mathbf{x}_i)\right)^2.$$

The expression E_c allows the vertices to move freely move in the tangent plane but would penalize if the vertices move off the surface (along the normal). For shapes with boundaries we have an additional term that accounts for maintaining the shape of the boundary. For each boundary vertex we compute the distance to closest boundary edge from the input mesh and attempt to minimize this distance. For example if $\mathbf{p}_l$ is a vertex on the boundary of P and $\mathbf{y}_1$ and $\mathbf{y}_2$ are vertices of the boundary edge that is closest to $\mathbf{p}_l$, then the boundary proximity error is expressed as

$$E_b = \sum_i |\mathbf{p}_l - \mathbf{y}_1 - \frac{(\mathbf{p}_l - \mathbf{y}_1) \cdot (\mathbf{y}_2 - \mathbf{y}_1)}{(\mathbf{y}_2 - \mathbf{y}_1) \cdot (\mathbf{y}_2 - \mathbf{y}_1)}(\mathbf{y}_2 - \mathbf{y}_1)|^2.$$

E_b error is summed over all the boundary vertices of P. The final global minimization error is expressed as

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

where α and β are linear weight associated with the proximity metric. Higher value for α and β would preserve the original shape, however higher weights for α and β

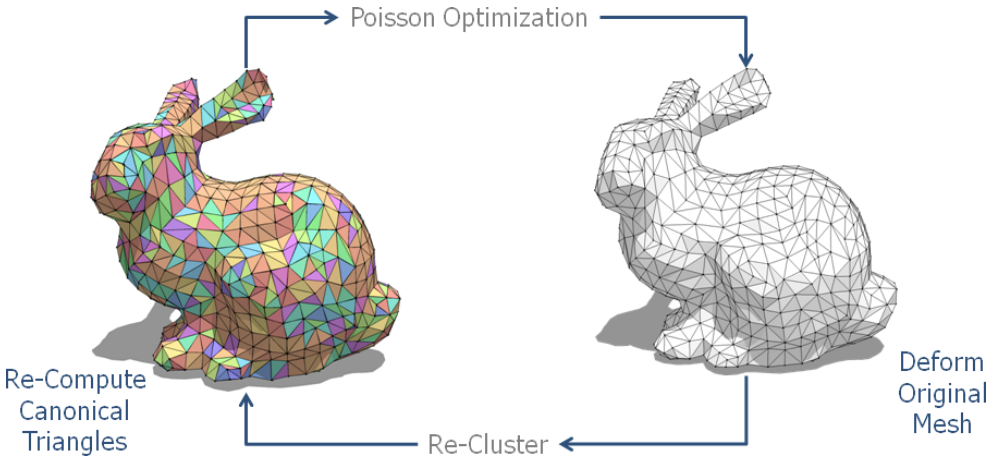


Fig. 21. Poisson based global optimization loops until convergence.

inhibit the rate of convergence. We are able to faithfully approximate the original shapes by keeping α and β to 0.001 and 0.01 respectively. We decrease these weights as the optimization progresses.

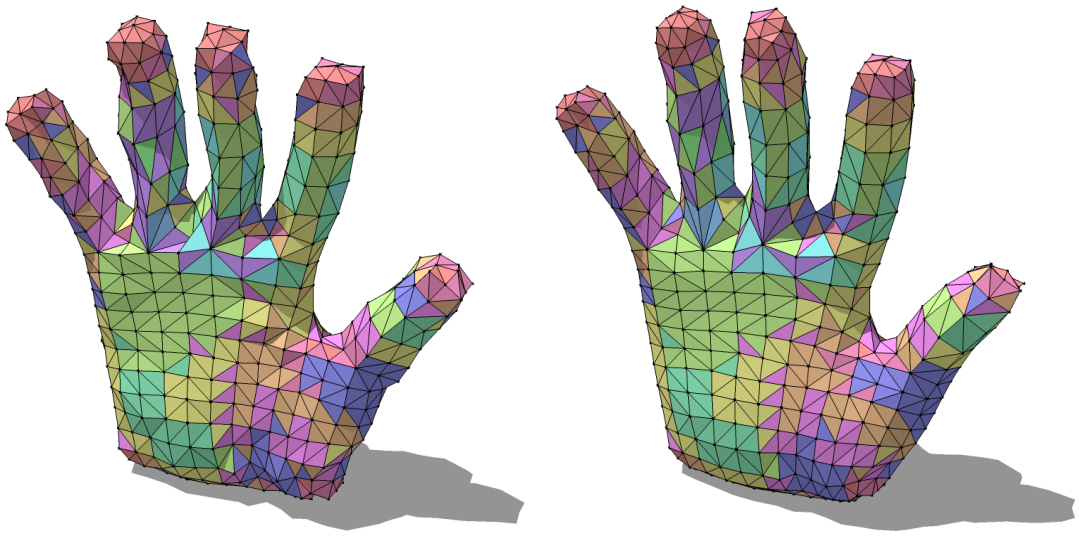
Our error function is quadratic and its minimum is given by the solution of a sparse system of linear equations. The entire minimization function is expressed a least square problem of the form

$$Af = b$$

and can be numerically solved using the conjugate gradient method. After solving for new vertex positions of P, as shown in Figure 21, we iterate the entire process of clustering and computing canonical triangles. This process loops until convergence.

3. Surface Fairness

The proximity metric is designed such that input mesh is closely approximated. Unfortunately this metric falls short of reproducing the input shape. Even though the vertices of the optimized mesh closely approximate the input shape, the normals of the optimized shape may not be faithful to the input shape. This causes undesired



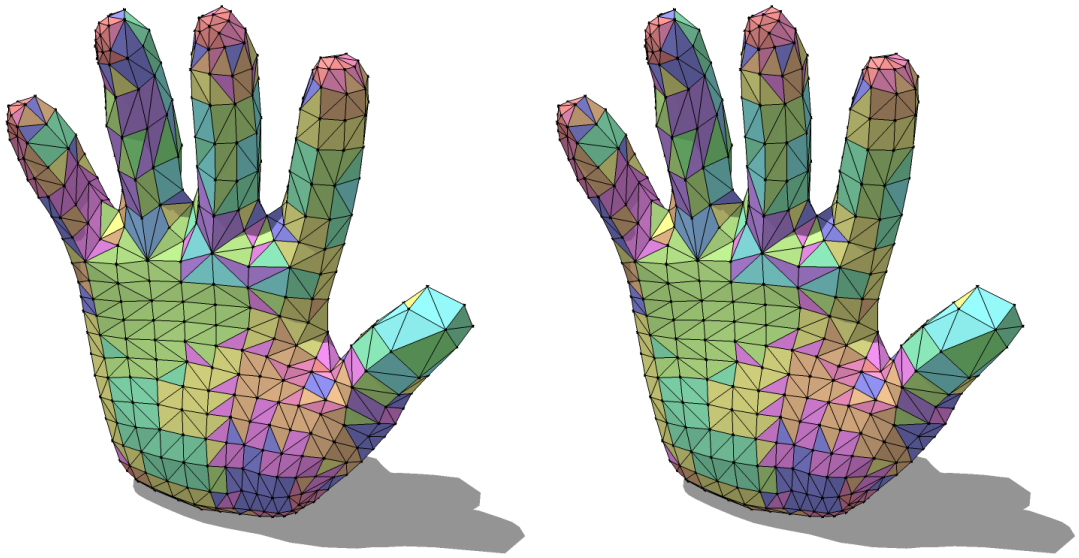
Without rotation 10% rotation

Fig. 22. 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.

undulation in the optimized mesh. Figure 22 (left) shows an example of how even though the geometry is closely approximated the shape has undesirable artifacts.

These artifacts can be removed if the optimized shape maintain the geometry as well as the normals of the input shape. We can match the normals of the optimized shape to the original shape by supplementing our our global optimization with a non-linear term that accounts for matching normals. Doing so will inhibit the rate of convergence of our global optimization.

Instead we maintain our global linear optimization as is and reproduce the normals of the input shape by slightly modifying the way we map a canonical triangle to its corresponding triangle in the mesh. Let $\mathbf{n}_i$ be the normal of the closest point $\mathbf{p}_0$ (on the input mesh) to $\mathbf{p}_i$ and let $\mathbf{m}_i$ be the normal of P^k, where P^k is the polygon of the mesh in current iteration of the optimization that $\mathbf{p}_i$ belongs to. When map-



50% rotation 100% rotation

Fig. 23. Example of an optimized mesh showing the effect of rotating canonical triangles towards the normal of the closest point on the surface. Notice that even rotating canonical polygons by half the way produces significantly improved results.

ping the canonical triangle $\hat{C}_{ind(i)}$ using rigid transformation, additionally we rotate the canonical triangle about the axis $\mathbf{m}_i \times \mathbf{n}_i$ by a small fraction. We do so before performing our Poisson optimization.

Figure 23 shows how the oscillating artifacts are removed by aligning the normal m_i with $\mathbf{n}_i$. Figure 22 demonstrates that we can achieve very good results by rotating the canonical triangle by only 1/10 of the angle between the two normals $(\mathbf{n}_i, \mathbf{m}_i)$. Larger rotations do not significantly improve the quality of results, although it does hamper the rate of convergence for our global optimization.

D. Discussion and Results

The optimization process alternates between assigning clusters/computing canonical triangles and performing global optimization. This process continues until the sum of

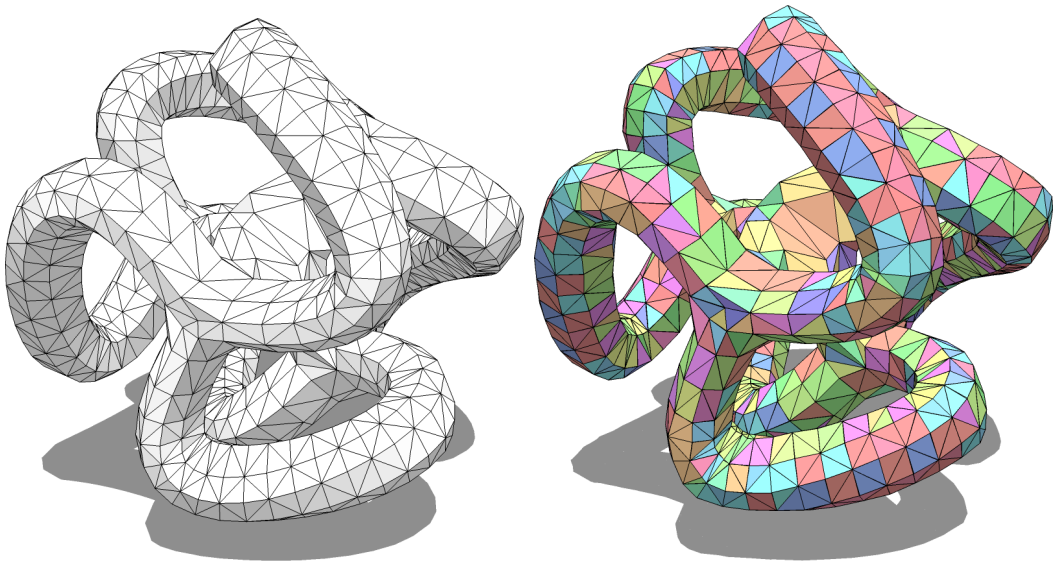


Fig. 24. A high genus shape with 2492 polygons. The initial shape is shown on the left and the optimized shape with 64 clusters is shown on the right.

distance between a triangle and its canonical counterpart (equation 3.2) falls below the prescribed tolerance level (ϵ). As shown in Figure 26 the error drop significantly during the initial stages of optimization. However the rate of error drop levels off during the latter part of the optimization. If in case the optimization fails to bring the error below the user-defined tolerance level, we simply add another cluster so as to better approximate the triangles in the input shape. The new cluster is initiated by the triangle with the worst error in the set. The number of clusters is highly dependent upon the shape as well as the tolerances expected. For example a closed shape (Figure 24) with high genus and curvature requires higher number of clusters in comparison to an open shape with zero genus (Figure 11). Table II shows a brief comparison of various shapes in terms of number of clusters required to approximate the shape along with the corresponding error levels.

Our optimization is meant to run before initiating the construction phase but only after the design has been finalized by the architect. The optimization is run

	Tris	**Clusters**	**Mean Error**	**RMS Error**
Figure 11	576	6	0.033%	0.063%
Figure 16 rhs	1280	10	0.029%	0.061%
Figure 21	1724	44	0.051%	0.099%
Figure 24	2492	64	0.026%	0.050%
Figure 27 top lhs	1396	84	0.055%	0.104%
Figure 27 mid rhs	676	19	0.023%	0.045%
Figure 27 btm lhs	1800	31	0.008%	0.017%
Figure 27 btm rhs	2880	58	0.012%	0.023%

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

as an offline process to generate approximate geometry. The 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. The bean shaped example shown in Figure 25 has 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 11, 16, 21, 24 and the figure 27 on p. 48 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 triangle surface P so that any gaps/overlaps are visible; though the error tolerance is set low enough that these

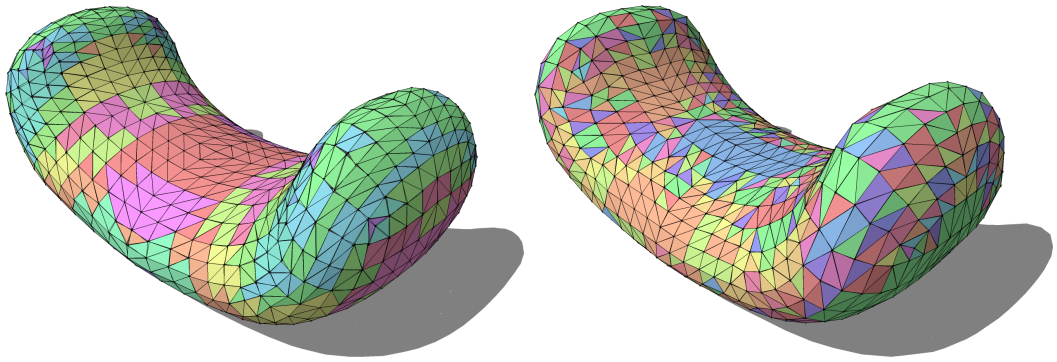


Fig. 25. Example of different clustering methods. Left: the shape is partitioned into 17 clusters before running global optimization. Right: the same shape starting with 1 cluster and incrementally adding clusters and running the optimization to convergence each time a cluster is added up to 17 clusters.

discrepancies are not visible. Figure 16 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 triangles, 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 21 and 24, 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 24). 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.

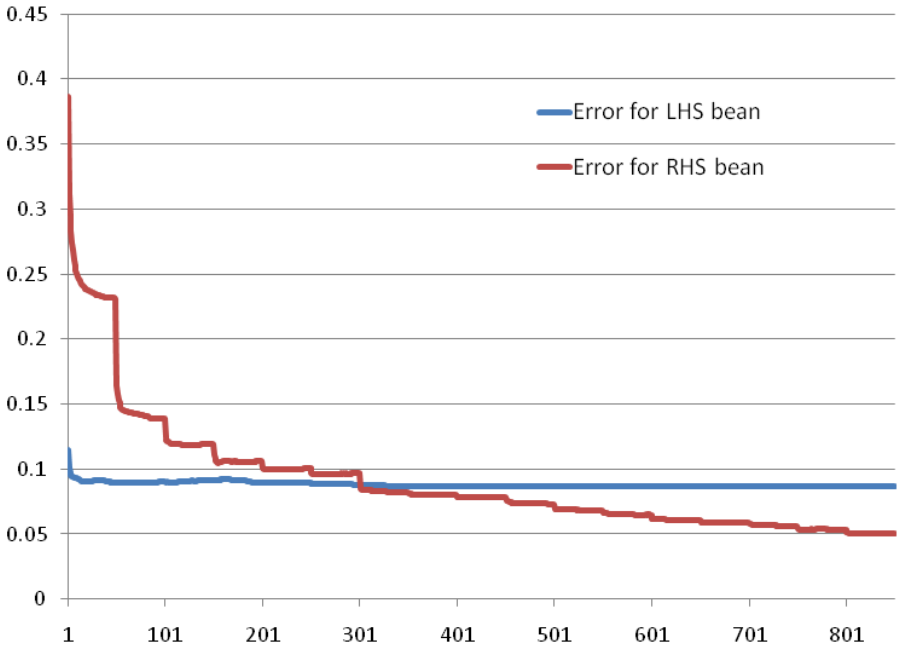


Fig. 26. Error graph for the performing clustering before global optimization (blue) and clustering while performing global optimization (red).

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 25 shows the difference between these two approaches and the plot of the error is shown in Figure 26. 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 25 where the polygons are more uniform in shape.

E. Conclusion and 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.

Unlike Eigensatz et al. [EKS*10], we do not explicitly account for aesthetic considerations such as preservation of streamlines over the surface. Conforming to the visual appearance of input mesh can be an additional constraint for our global optimization. Although this may come at a cost of slowing down the rate of convergence. Also for shapes such as Stanford Bunny (Figure 27 on p. 48 Top-RHS) defining as well as detecting streamlines will be a non-trivial task.

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 [LPW*06]. The constraints for planar quad meshes are already non-linear and will certainly harm the convergence of our method, possibly requiring substantially more clusters.

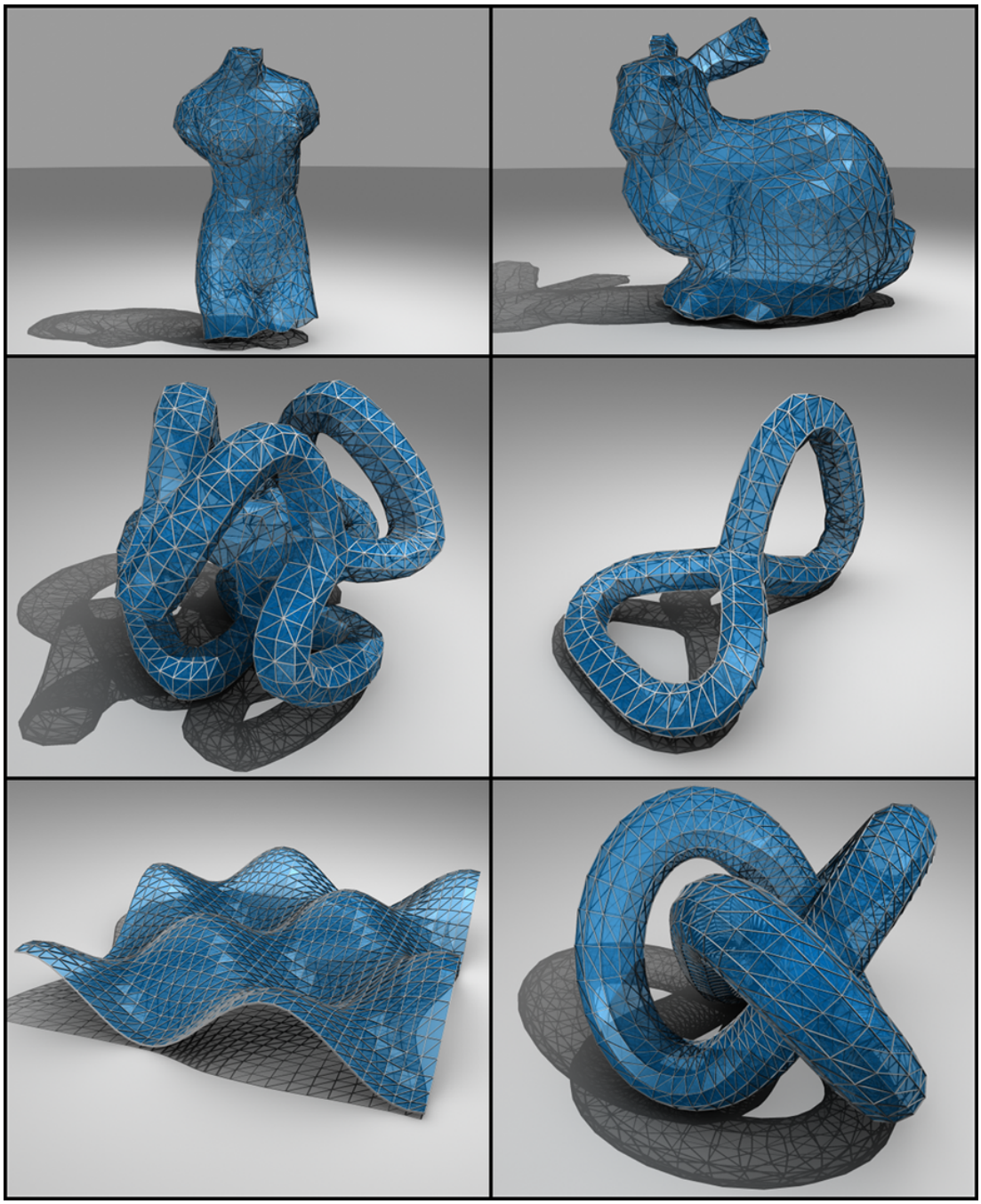


Fig. 27. Final results of various shapes with different number of clusters. The details of each shape are summarized in Table II.

CHAPTER IV

AESTHETIC TRIANGLES

An extensive collection of research work in Computer Graphics aim to create aesthetic meshes using a set of *good triangles* [She96, She02b]. The primary motivation behind creating such a mesh is to be able to conduct geometry dependent discrete finite element simulations (FES) in a stable and efficient manner. High quality meshes for FES aim to maximize the minimum angle of each triangle in the mesh. Such a mesh is isotropic in nature and visually all triangle aim to be roughly equilateral.

We wish to take an alternative approach to the problem of generating a set of good triangles by broadening the definition of aesthetic meshes. While maintaining the visual quality of input mesh we aim to alter triangles that are dissimilar in their local neighbourhood. For example if we have a skinny or small triangle adjacent to a large triangle, we aim to simultaneously deform both triangles minimally such that they are visually similar. We do so by introducing a *skew* error metric that measure dissimilarity of triangles across a shared edge. The error function for a given shape is defined as the cumulative sum of this skew error metric measured over all the edges sharing a pair of triangles (i.e. excluding boundary edges). We then optimized the shape so as to minimize our non-linear error function. Figure 28 shows two examples of an architectural shape optimized using our error metric. Note that in both cases the shape of triangles vary smoothly over the surface.

Our approach primarily focuses towards generating meshes for freeform architectural shapes. Unlike other applications in Computer Graphics (CG), freeform shapes for architectural buildings are massive in scale and the mesh stands in public eye, thus making the aesthetics an imperative aspect of the final shape. These architectural shapes are created by architects or designers and it is critical to maintain features

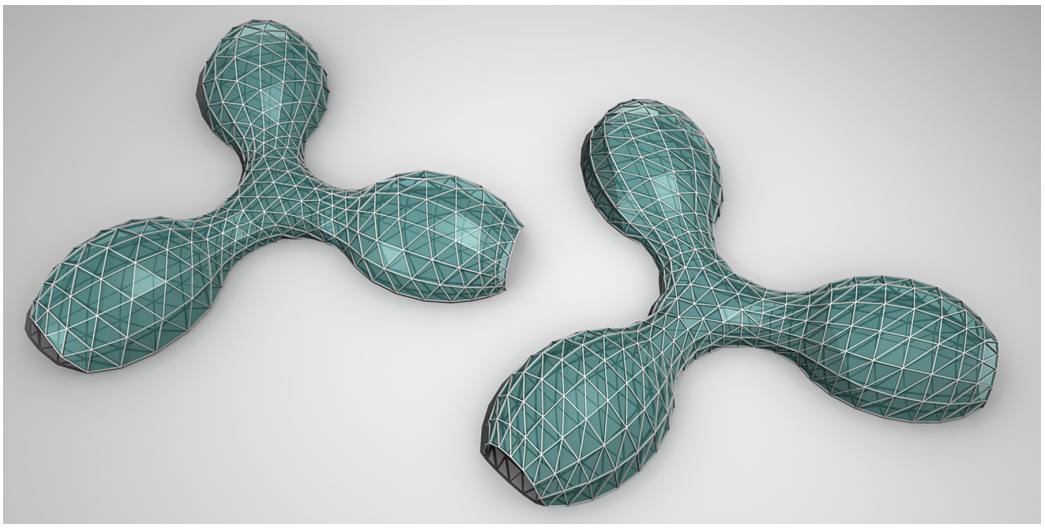


Fig. 28. Example of an optimized dome shape architectural roof highlighting the different between using the two alternatives (reflections only on the left and reflection and rotation on the right) for for our strain based error metric.

such as boundary of the shape, parameter lines, and topology of the original shape. Our method preserves both the input geometry and topology.

Our skew based error metric is similar to the *linearized strain* which is commonly used while studying the deformations of physical objects in Computer Graphics [ITF04, WRK*10]. Strain for a physical object is measured as deformation representing the displacement between particles in the body relative to its reference length or size. In our context, we measure deformation by mapping adjacent triangles, that share a common edge, onto each other. The deformation of one triangle is measured using the shape of the other as a default state or rest pose. This measurement is performed in a symmetric manner such that deformation of each triangle is measured using the other as the rest pose. We illustrate the effectiveness of our strain based mesh fairing method using a set of models with varying curvature and feature size in section H.

To summarize, as a part of this work we propose a simple edge based error metric

to quantify dissimilarity between a pair of triangles. We will also present a rotation minimizing simplified form of this metric expressed using a combination of edges and areas of the two triangles. We demonstrate the effectiveness of this metric via a non-linear optimization based solver. Additionally, we also provide a closed form analytic Jacobian of the error function so as to be computationally efficient for large meshes.

A. Relevant Work

Williams et al. [Wil01] proposed a simple spring relaxation based method for creating a set of aesthetic triangles. Initial mesh for the British Museum Court Roof was generated by tessellating a regularly sampled $2d$ parametric function with associated height value. The method relaxes the mesh by preforming a discrete differencing of vertices along the parameter lines. It also takes into account the stipulated maximum triangle size for the glass panels. Even though this method was successful in generating the resulting mesh, the authors mention that it is numerically unstable and explodes unless the time step as well as the spring tension is kept sufficiently small.

Shewchuk [She02b] describes a comprehensive set of attributes to measure the quality of triangles in a mesh. His work compare and contrast triangle quality both in terms of aesthetics and stability for performing finite element simulations. The most commonly used method for generating quality triangles is via Delaunay triangulation. This method maximizes the minimum angle for all the triangles in the mesh, hence avoiding skinny triangles. Detailed references for fundamental notions, algorithms, and applications are available with Aurenhammer [Aur91] and Okabe et al. [OBSC00],

Centroidal Voronoi Tessellation (CVT) [DFG99] is a very commonly used simple and stable method for generating high quality isotropic triangles. The method consists

of interleaving Delaunay triangulation with optimization of the triangle domain using Lloyd's algorithm. During Lloyd's iteration, each vertex in the mesh is moved to the center of its Voronoi cell. This iterative alternating technique reduces the quadratic error described in equation 4.1.

$$E_{CVT} = \sum_{i=1\cdots N} \int_{V_i} \|\mathbf{x} - \mathbf{x}_i\|^2 dx \tag{4.1}$$

where $\mathbf{x}_i$ are the vertex position and V_i is the local voronoi cell associated with the vertex $\mathbf{x}_i$. A variation of CVT was introduced by Chew [Che87] where so-called *Steiner points* were inserted to limit the size of triangle edges below a given user-defined threshold. Ruppert [Rup95] and Shewchuk [She01] extended Chew's [Che87] method to account for triangle size. The associated size factor for the triangles is modulate manually or defined using spatial characteristics such as distance to the medial axis of the mesh. In order to generate meshes over a $3d$ surface a variant of CVT is used referred to as *constrained* CVT (CCVT) [DGJ02] or *restricted* CVT (RCVT) [YLL*09].

CVT and variants thereof tend to produce triangles that are very close to being equilateral. We define our metric such that it would return a zero error if all triangles are equilateral. Our skew-based error metric will also return a zero error in cases where all triangles are not equilateral.

For example let the set of triangles shown on the left hand side of Figure 29 be input for both CVT and our method. The mesh in the middle of Figure 29 is generated as a result of performing CVT over the input domain. Note that all triangles are roughly equilateral. Our method will return a zero error for the mesh illustrated in the middle of Figure 29. Mesh on the right side of Figure 29 is produced as a result of performing our skew based optimization over the input domain. The error for this mesh is zero as well. However the triangles in the mesh on the right

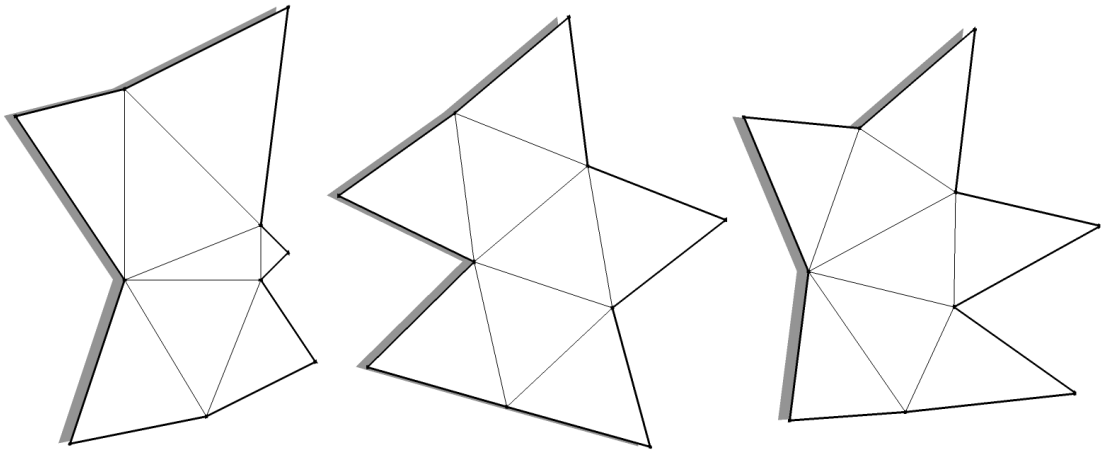


Fig. 29. The input mesh is shown on the left hand side. In the middle is the result of optimizing the input mesh using CVT like methods. On the right hand side is the input mesh optimized using our method. Our method returns a zero error for both meshes in the center as well the one on the right hand side. Note that the shape of triangles in the mesh on the right hand side and in the middle is distinctly different.

side of Figure 29 are not equilateral. In a sense we aim to broaden the definition of aesthetic triangles such that it would return a zero error not only for a set of equilateral triangles but for other shape and size triangles as well.

B. Skew Transformation

Presuming that we are given a correspondence between the vertices of two triangle P and Q as illustrated in Figure 30 we can map P onto Q in a congruent manner using an affine transformation defined as

$$A_{(P,Q)} = \sum_{\ell=1}^{3} p_\ell^* \left(q_\ell^*\right)^T \tag{4.2}$$

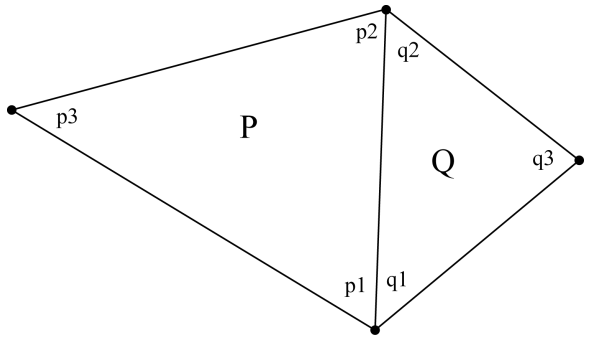


Fig. 30. The error metric operates upon a shared edge $e1, e2$ with triangle P and Q on either side.

where p^*_ℓ and q^*_ℓ are the centroids of triangle P and Q respectively. The affine transformation $(A_{(P,Q)})$ between P and Q is a set containing rotations, translations, dilations, and shears. Here the translation is given by

$$T_{(P,Q)} = q^* - p^* \cdot (A_{(P,Q)}).$$

We factor the affine transformation mapping P onto Q into two parts. The first part transforms P onto Q in a rigid manner, where the length of edges and area of triangle P are preserved. The second part deforms the triangle P such that the vertices of P are co-located with the vertices of triangle Q. Mathematically, this relationship is expressed as

$$A_{(P,Q)} = R_{(P,Q)} \cdot S_{(P,Q)},$$

here $R_{(P,Q)}$ is the rigid transformation that consists of only rotation and reflection, while $S_{(P,Q)}$ is referred to as *skew* transformation. A similar formation was used by and Irving et al. [ITF04] and Wicke et al. [WRK*10] while computing deformation gradient of a tetrahedron. We can compute the rigid and skew transformation via singular value decomposition of $A_{(P,Q)}$ expressed as

$$A_{(P,Q)} = UWV^T = (UV^T)(VWV^T)$$

where U and V are orthogonal and W is diagonal with all its diagonal entries positive. The rigid transformation (rotation and reflection) is computed as

$$R_{(P,Q)} = UV^T.$$

If $det(R) < 0$, we simply negate the vector in V corresponding to the smallest singular value. The left over skew transformation is defined as

$$S_{(P,Q)} = VWV^T$$

Note that linearized strain matrix is defined as $\epsilon = V(W - I)V^T$ by Wicke et al. [WRK*10]. Now our skew transformation can be simply expressed as

$$S_{(P,Q)} = R^{-1}_{(P,Q)} \cdot A_{(P,Q)} = R^T_{(P,Q)} \cdot A_{(P,Q)}$$

since $R^{-1} = R^T$.

C. Skew Error Metric

We define the error between triangle P and Q as the magnitude (Frobenius norm) of skew or deformation required to map one onto other. The error is expressed as

$$Err_{(P,Q)} = |I - S_{(P,Q)}|_{\mathcal{F}} = |I - R^T_{(P,Q)} \cdot A_{(P,Q)}|_{\mathcal{F}}. \tag{4.3}$$

Note that the error in equation 4.3 is zero if the triangle P maps perfectly onto Q. Thus if both triangle P and Q are equilateral the error will be zero. For a given edge i, we consider a symmetric error by taking into account the reverse mapping as well. The final error for an edge i shared by the triangles P and Q is expressed as

$$Err^i = Err^2_{(P,Q)} + Err^2_{(Q,P)}.$$

Based upon this error metric we define a minimization function over the entire mesh as

$$E_s = \min \sum_{i=0}^{n} Err^i, \tag{4.4}$$

where n is the number of edges in the mesh that share a pair of triangles on either side.

D. Minimizing Rigid Transformation

We simplify our error function as described in equation 4.3 by minimizing the rigid transformation. To do so we project the pair of $3d$ triangles P and Q into a $2d$ plane. We are able to perform such a projection because our error function is invariant under orthogonal transforms. By projecting the triangles into $2d$ plane we reduce the affine and the rigid transformation to a 2×2 matrix. Therefore rewriting the error function $Err_{(P,Q)}$ as defined in equation 4.3 between two triangles P and Q as

$$Err^2_{(P,Q)} = |I - \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \cdot \begin{bmatrix} a0 & a1 \\ a2 & a3 \end{bmatrix}|^2_{\mathcal{F}}. \tag{4.5}$$

We minimize the rigid transformation in the above error expression (equation 4.5) and compute the values of c and s as

$$\frac{a0 + a3}{\sqrt{(a0 + a3)^2 + (a1 - a2)^2}} \quad \text{and}$$
$$\frac{a1 - a2}{\sqrt{(a0 + a3)^2 + (a1 - a2)^2}}$$

respectively. Based upon the above values of c and s, we simplify the skew error function ($Err^2_{(P,Q)}$) between P and Q as

$$2 + a0^2 + a1^2 + a2^2 + a3^2 - 2\sqrt{(a1 - a2)^2 + (a0 + a3)^2}. \tag{4.6}$$

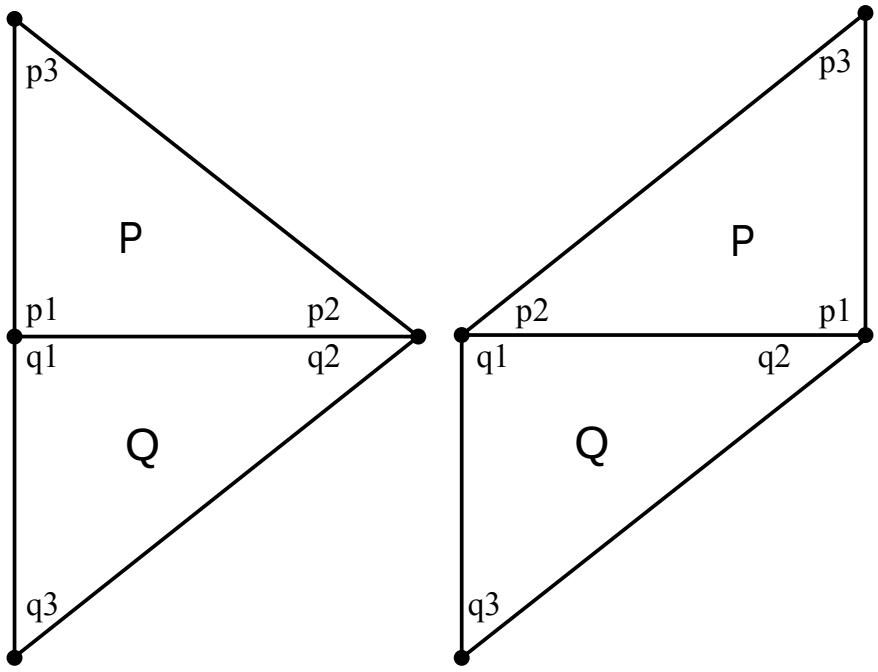


Fig. 31. Two different alternatives to map triangle $P(p1, p2, p3)$ onto $Q(q1, q2, q3)$. On the left is the *reflection* mapping and on the right side is the *rotation* mapping.

He et al. [HSH10] defined the same solution as described in equation 4.6 in context of as-rigid-as-possible surface parametrization. This formulation expresses our error function in a manner such that it only requires us to compute affine transformation between triangles P and Q as mentioned in equation 4.2. Now even though affine transformation requires inverting a matrix, it is only a 2×2 matrix and can be done easily and efficiently. We can compute the error $Err_{(Q,P)}$ for reverse mapping by simply inverting $A_{(P,Q)}$, a 2×2 matrix.

E. Mappings

So far we have presumed that a correspondence is provided between vertices of triangle P and Q. For a given pair of triangles P and Q as shown in Figure 30 with vertices $(p1, p2, p3)$ and $(q1, q2, q3)$ respectively let there be a shared edge between the vertices

$(p1, p2)$ and $(q1, q2)$. Figure 31 highlights the two alternatives for mapping vertices of P onto Q that take advantage of the shared edge. The left side of the Figure 31 illustrates a mapping where a reflection of triangle P is mapped onto Q such that $p1$ is mapped to $q1$, $p2$ is mapped to $q2$ and $p3$ is mapped onto $q3$. Henceforth we will refer to this mapping as *reflection mapping* between P onto Q. The right side of the Figure 31 shows a mapping between triangle P and Q where triangle P is rotated and reflected such that $p1$ is mapped to $q2$, $p2$ is mapped to $q1$ and $p3$ is mapped onto $q3$. We will refer this mapping as *rotation mapping* between P and Q.

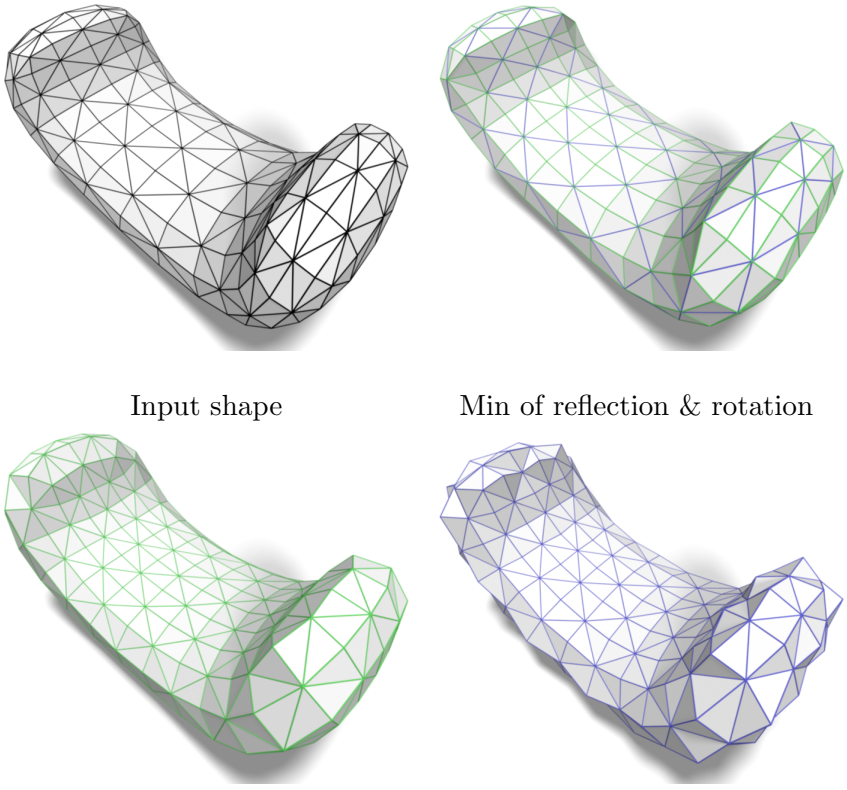


Fig. 32. Optimized model of bean using only reflection mapping (bottom left), only rotation (bottom right) and using minimum of reflection and reflection (top right) mapping. Note that high curvature features on either side of the bean are better preserved while using the minimum of both mappings.

Figure 32 shows an example of bean shaped model using reflection mapping on the left side of bottom row, rotation mapping on the right side of bottom row and minimum of reflection and rotation on the right side of top row. Note that the high curvature features on the either side of the bean are better preserved while using the minimum of two mappings (reflection and rotation). Also the minimum of two mappings provides visually pleasing variation amongst the triangles, unlike rotation or reflection where the optimization tends to make all triangles the same shape. Also, note that the parameter lines are well preserved while using the preferred minimum of reflection and rotation mapping.

F. Canonical Space

The error expression in as described in equation 4.6 requires us to map the two triangles P and Q into a $2d$ plane and compute an affine transformation between P and Q. By reducing our search space to only 2 mappings (reflection and rotation) we can further simplify our error function such that the skew error is expressed in a dimension independent manner using only the edge lengths and area of the two triangles.

To do so we begin by considering the default mapping between the two triangles P and Q, such that $p1$ maps to $q1$, $p2$ maps to $q2$ and $p3$ maps to $q3$. Subsequently we transform the two triangles to a canonical space where $p1$ and $q1$ are co-located at the origin and the edges $(p2 - p1)$ and $(q2 - q1)$ are aligned along the positive x-axis. Now we can rewrite the coordinates of the two triangles as

$$p1 = (0, 0) \quad p2 = (L, 0) \quad p3 = (x_1, y_1)$$
$$q1 = (0, 0) \quad q2 = (L, 0) \quad q3 = (x_2, y_2)$$

We define the edges for triangles P as

$$\begin{aligned} pE1 &= (p3 - p2) = \{x1 - L, 0\} \\ pE2 &= (p1 - p3) = \{-x1, 0\} \\ pE3 &= (p2 - p1) = \{L, 0\} \end{aligned}$$

and similarly for triangle Q as

$$\begin{aligned} qE1 &= (q3 - q2) = \{x2 - L, 0\} \\ qE2 &= (q1 - q3) = \{-x2, 0\} \\ qE3 &= (q2 - q1) = \{L, 0\} \end{aligned}$$

Using the $2d$ canonical coordinate space we simplify the error between triangles P and Q as expressed in equation 4.6 to

$$3 + \left(\frac{x_1 - x_2}{y_1}\right)^2 + \left(\frac{y_2}{y_1}\right)^2 - 2\sqrt{\left(\frac{x_1 - x_2}{y_1}\right)^2 + \left(1 + \frac{y_2}{y_1}\right)^2}$$

In context of canonical coordinates we also know that

$$\begin{aligned} pE3 \cdot pE2 &= -Lx_1 \\ qE3 \cdot qE2 &= -Lx_2 \end{aligned}$$

Also

$$\begin{aligned} \Delta_p &= \frac{Ly_1}{2} \\ \Delta_q &= \frac{Ly_2}{2} \end{aligned}$$

where Δ_p and Δ_q are the area of triangles P and Q respectively. Using the above

relationship we can rewrite the error function $(Err^i_{(P,Q)})^2$as

$$3 + \left(\frac{-pE3 \cdot pE2 + qE3 \cdot qE2}{2\Delta_p}\right)^2 + \left(\frac{\Delta_q}{\Delta_p}\right)^2 - 2\sqrt{\left(\frac{-pE3 \cdot pE2 + qE3 \cdot qE2}{2\Delta_p}\right)^2 + \left(1 + \frac{\Delta_q}{\Delta_p}\right)^2}. \quad (4.7)$$

Similarly for the reverse mapping of triangle Q onto P we can express $(Err^i_{(Q,P)})^2$ as

$$3 + \left(\frac{-pE3 \cdot pE2 + qE3 \cdot qE2}{2\Delta_q}\right)^2 + \left(\frac{\Delta_p}{\Delta_q}\right)^2 - 2\sqrt{\left(\frac{-pE3 \cdot pE2 + qE3 \cdot qE2}{2\Delta_q}\right)^2 + \left(1 + \frac{\Delta_p}{\Delta_q}\right)^2} \quad (4.8)$$

The equation 4.7 and 4.8 can be used for the reflection mapping as well. We will just need to recompute $pE3, pE2, qE3$ and $qE2$. Δ_p and Δ_q remains the same.

The skew error for triangles P and Q in equation 4.7 and 4.8 is simplified such that it does not require projection into $2d$ plane or computation of an affine transformation between P and Q. Note that the error is expressed in a dimension independent manner using only using edge lengths and area of the two triangles. Also, we can easily decompose the error expressed in equation 4.7 and 4.8 so as to compute analytic derivatives as shown in appendix B.

G. Global Optimization

In order to avoid a trivial zero error solution and also to maintain the original input shape we supplement our optimization function as described in equation 4.4 with two additional proximity terms. The first term E_{sp} represents closeness to the original surface and the second term E_{bp} accounts for proximity to the boundary of the original surface. Obviously the second term will be zero for a closed surface. We compute

proximity to the surface as

$$E_{sp} = \sum_i \left(\mathbf{n}_i \cdot (\mathbf{x}_i - \mathbf{p}_i)\right)^2 \tag{4.9}$$

where $\mathbf{x}_i$ is the position of i^{th} vertex of mesh in the current iteration of optimization, $\mathbf{p}_i$ is the footpoint of $\mathbf{x}_i$ on the original mesh and $\mathbf{n}_i$ is the normal of the original surface at point $\mathbf{p}_i$. This formulation not only keep the optimized surface close to the original surface it also allows the point $\mathbf{x}_i$ to move laterally within the tangent plane.

Additionally, while preserving the shape we would also prefer to reproduce the boundary of the original input shape. To do so for each boundary vertex we compute the distance to closest boundary edge from the input mesh and attempt to minimize this distance. For example if $\mathbf{p}_l$ is a vertex on the boundary of P and $\mathbf{y}_1$ and $\mathbf{y}_2$ are vertices of the boundary edge that is closest to $\mathbf{p}_l$, then the boundary proximity error is expressed as

$$E_{bp} = \sum_i |\mathbf{p}_l - \mathbf{y}_1 - \frac{(\mathbf{p}_l - \mathbf{y}_1) \cdot (\mathbf{y}_2 - \mathbf{y}_1)}{(\mathbf{y}_2 - \mathbf{y}_1) \cdot (\mathbf{y}_2 - \mathbf{y}_1)}(\mathbf{y}_2 - \mathbf{y}_1)|^2.$$

This error E_{bp} is summed over all the boundary vertices of the input mesh. The final global minimization error is expressed as

$$\min_p E_s + \alpha E_{sp} + \beta E_{bp} \tag{4.10}$$

where α and β are linear weight associated with the proximity metric. Higher values for α and β will closely maintain the original shape. We are able to faithfully approximate the original shapes by setting α and β to 50.0 and 50.0 respectively. We solve our cumulative error function mentioned in equation 4.10 using non-linear optimization via Levenberg–Marquardt algorithm as implemented in an open-source library available online [Lou04].

H. Discussion and Results

We demonstrate the effectiveness of our non-linear optimization based method against a set of simple shapes consisting of both positive and negative curvature. All examples shown in this section use the minimum of 2 mappings (default and reflection) as mentioned in section E, unless otherwise noted.

The example shapes in Figure 33 are created using a set of randomly distributed points in (x, y)-plane. For each sample point in $2d$ we compute the z-coordinate using a simple algebraic function. For the example test shapes in Figure 33 we use 200 interior vertices and 40 vertices lie along the boundary of the surface. We perform Delaunay triangulation to create the initial set of triangles. Figure 33 illustrates the shapes rendered using a simple linear false color mapping where green indicates high error while blue shows triangles with lower error. Error for a face is computed as a sum of error of the three edges. The right column of Figure 33 shows significantly improved tessellation for all shapes without transforming all the triangles into equilateral triangles. Note that as mentioned previously we do not edit the initial topology of the shape. For the Gaussian bump, Saddle and the Wavy surface we reduced the final error to less than 1% of the initial error. For the final shape of Parabloid we constrained the four boundary vertices such that their position in space is fixed. Therefore most of the error (green color triangles) is localized around the boundary of the shape. The error for the final shape of Parabloid is reduced to 2.5% of the error of the original shape.

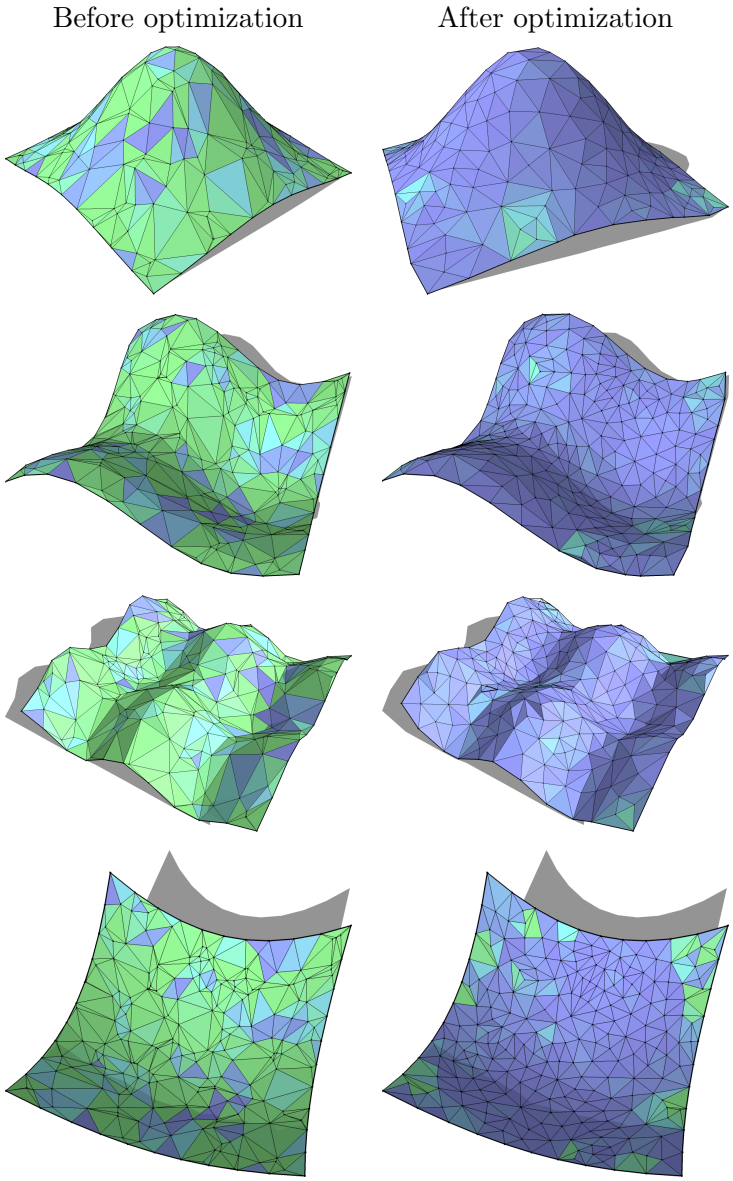
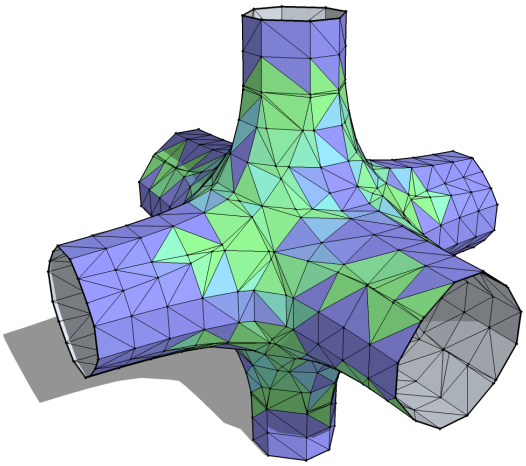
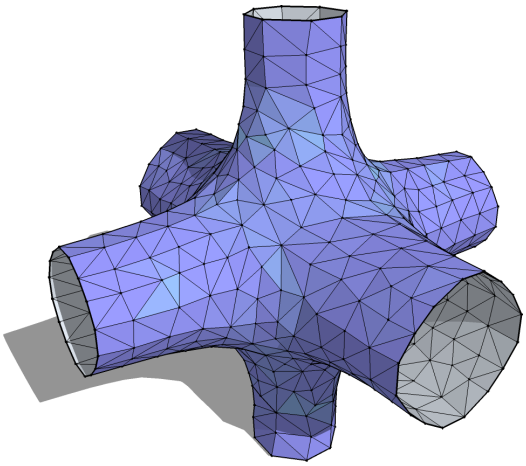


Fig. 33. Gaussian bump, Saddle, Wavy surface, and Parabloid generated using simple algebraic functions. The shapes are colored based on error, where green indicates higher error and blue indicates low error.



Surface before optimization.



Surface after optimization.

Fig. 34. Example of a tube shaped surface generated using *Marching Cubes* algorithm.

Along with simple algebraic shapes, we also tested our method on shapes generated by the Marching Cubes (MC) method [LC87]. The MC method tends to produce a set of bad skinny triangles along the edges of the grid cells. We illustrate the problem of bad triangles using an example shape in Figure 34 consisting of two intersecting hollow tubes. The final error for the optimized output shape as shown on the right side of Figure 34 is less than 0.5% of the input shape generated using the MC method.

Apart from maintaining the topology of the input mesh, our method optimizes the mesh such that it tends to reduce the dissimilarity amongst triangles in a local neighbourhood while editing the mesh in a minimal manner. This is illustrated via examples of Venus and Bunny in Figure 35 and 36 respectively where fewer dissimilar triangles are highlighted in green color. The final error for Venus drops to 18.55% of the original shape while making certain that the final position of vertices lie perfectly on the input shape. For the Bunny, this error drop to 17% of the input shape. Convergence for closed shapes is slower than shapes with flexible boundary, since a shifting boundary can provide extra degrees of freedom during the optimization. Note that unlike CVT and its variants our optimization does not tend to create isotopic triangles. For example, the top right side of Figure 35 shows the back of Venus where we have a set of long triangles. Note that the shape of these triangles remains approximately the same in the optimized shape as shown in the bottom right side of Figure 35.

We observe that the error in optimized shape tends to localize around low valence vertices (3 or 4). These vertices do not provide enough degrees of freedom to lower the error without sacrificing the quality of surface. Our optimization tends to move low valence vertices along the normal of the surface, thereby causing undesirable artifacts. We can avoid such artifacts by raising the value for proximity coefficients α and β. Higher values for proximity coefficients will inhibit the rate of convergence. We use 50 and 100 for α and β respectively, for both Venus (Figure 35) and Bunny (Figure 36).

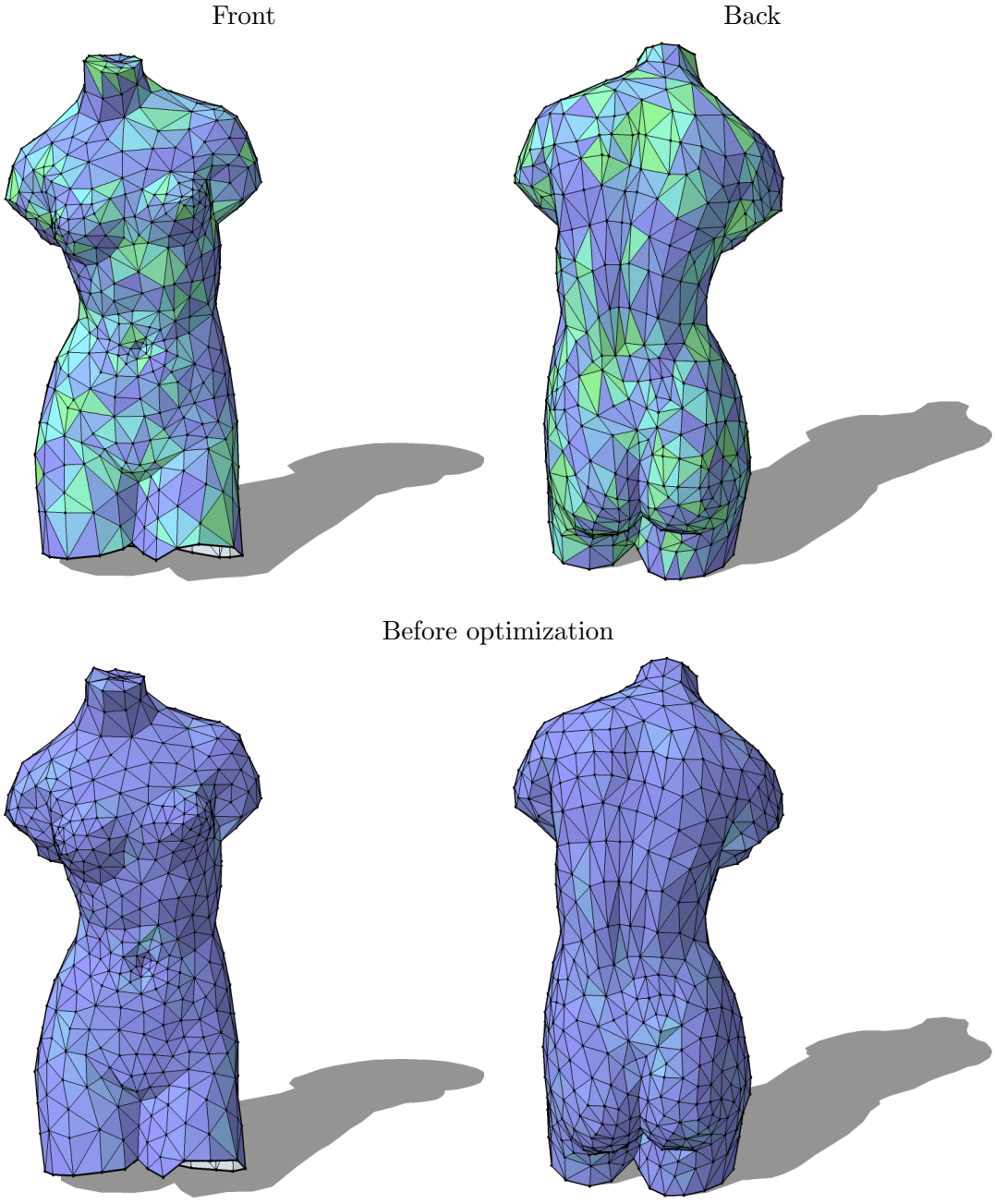


Fig. 35. Example of front (top) and back (bottom) of input (left) and optimized (right) model of Venus.

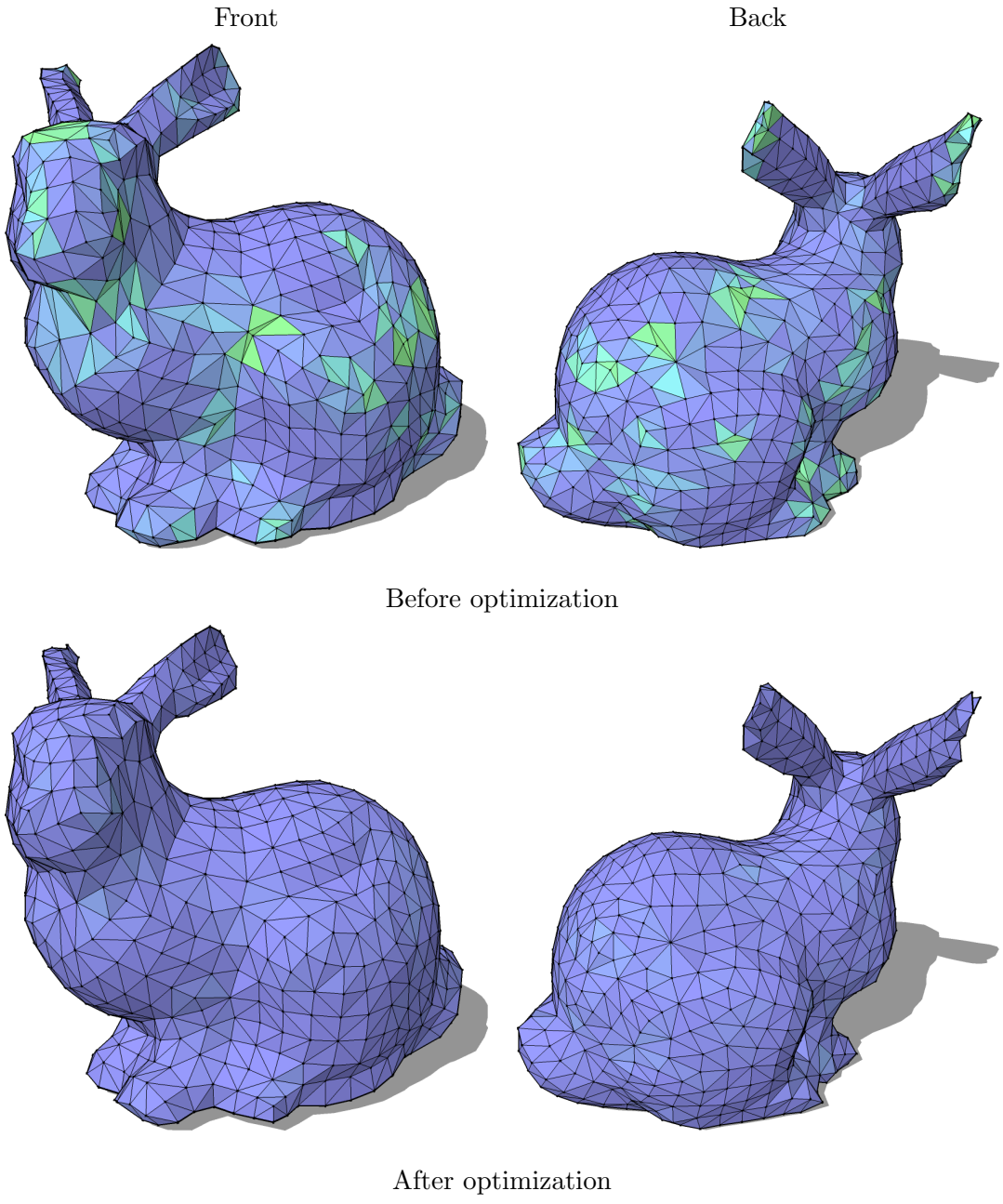


Fig. 36. Example of front (left) and back (right) of input (top) and optimized (bottom) model of Bunny.

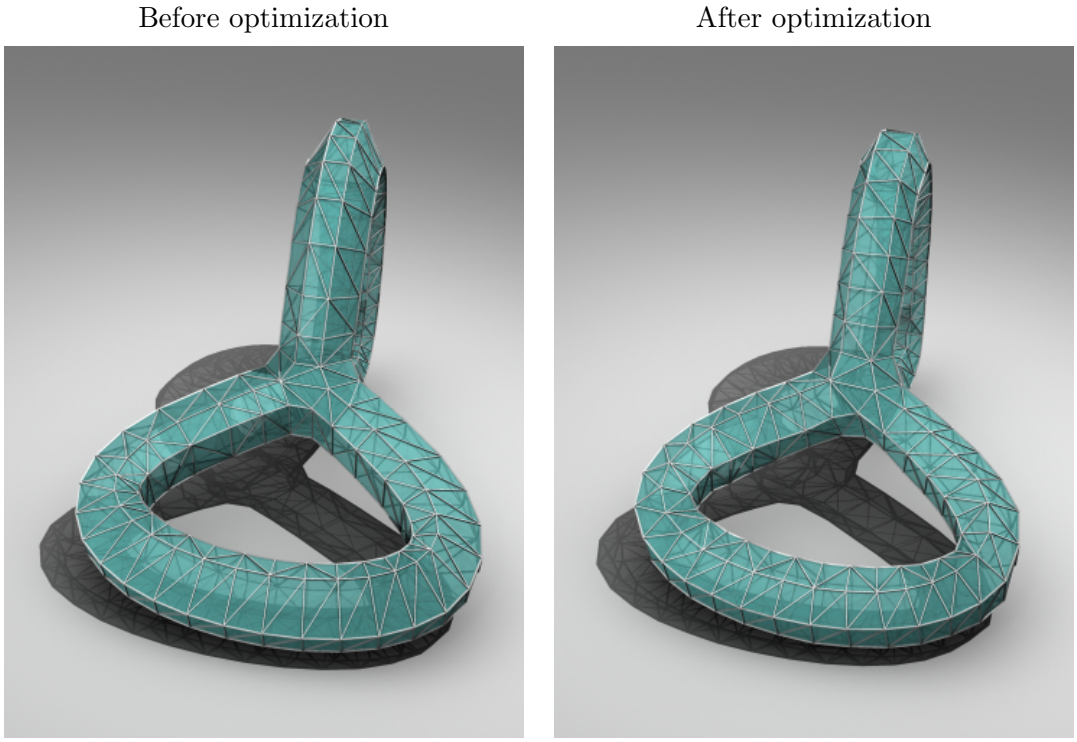


Fig. 37. Example of an abstract shape. The size of the triangles along the tube is significantly smaller in the optimized shape on the right.

Using the minimum of two mappings (reflection and rotation) as described in section E we are able to implicitly maintain parameter lines over the surface, as can be seen in the abstract shape in Figure 37. Note that along the tube, in the input shape on the left, the small size triangles are adjacent to larger triangles. The triangle size is approximately uniform in the optimized output shape as shown on the right side of Figure 37.

We also tested our method against a set of freeform roofing structures. Figure 38 shows a coarse approximation of the roof of British Museum, in London, England. The vertices of this mesh are sampled off the final shape of the roof. Notice that significant portion of the error lies along the boundary of the shape as can be seen in the side view of the roof (bottom of Figure 38). The optimized shape shown on the

right fixes the smaller triangles on the edge as well as maintains the parameter lines along with the interior and the exterior boundary of the shape.

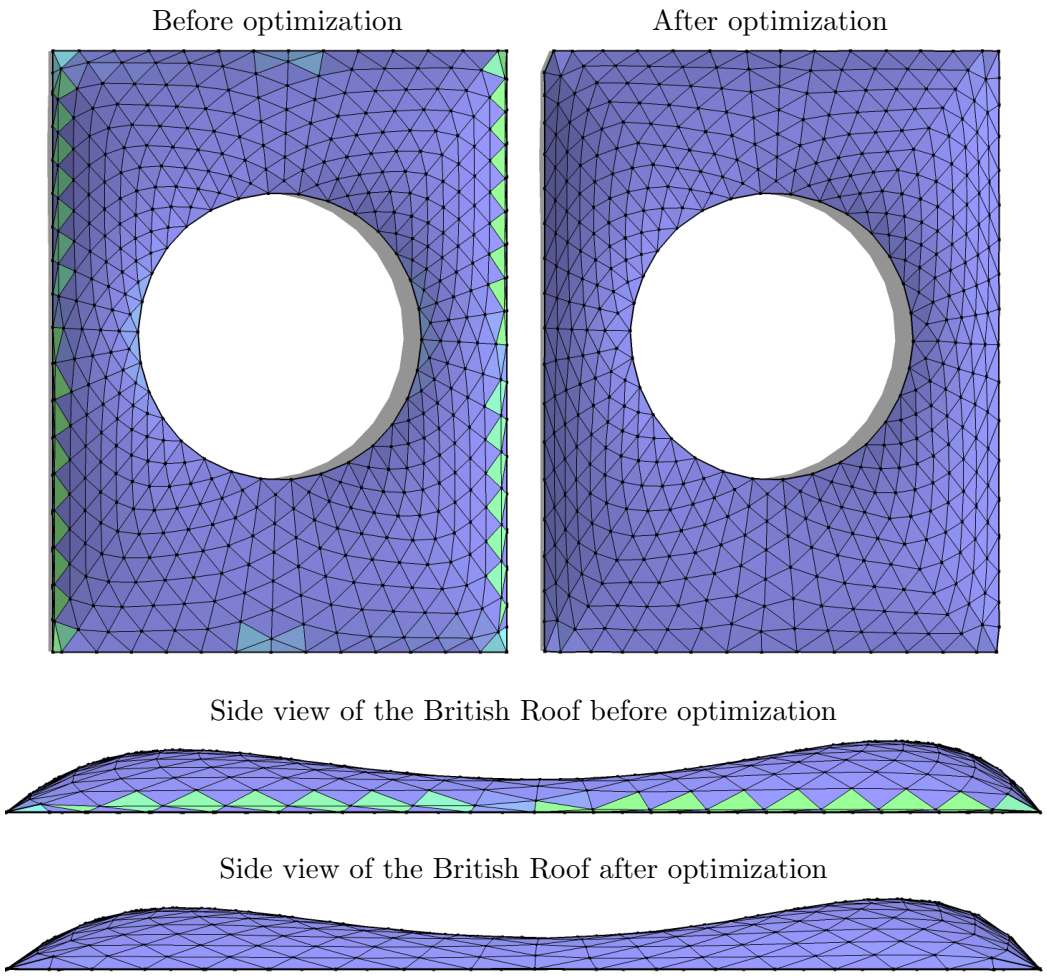


Fig. 38. A coarse approximation of British Roof. The points for the mesh are sampled off the original British Museum. Error shown in green mostly occurs on either side of the surface. The optimized shape on the right reduces the error while maintaining the aesthetic parameter lines. Final error is reduced to 1.98% of the original surface.

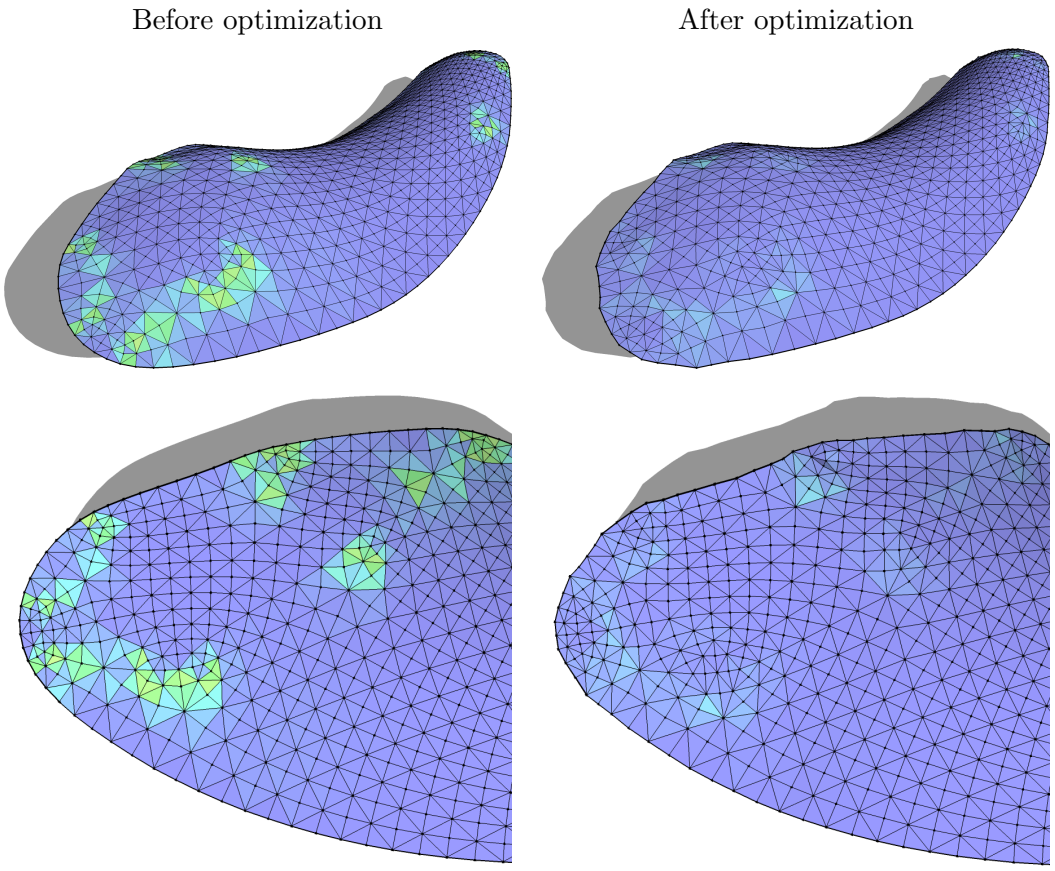


Fig. 39. An example of subdivided architectural roof. The error is localized around the extraordinary vertices. Note that parameter lines are very well maintained in the final shape.

Another roof shaped structure is shown in Figure 39. The final shape of the roof is created as a result of Catmull-Clark subdivision. While most of the shape has extremely good triangulation, there are high error areas around the extraordinary vertices. The shape of these triangles comes from the parameter lines generated from the characteristic map of the chosen subdivision scheme. Our method highlights such problem areas in green color as seen in Figure 39 and attempts to reduce dissimilarity amongst triangles around the extraordinary vertices. The bottom of the Figure 39 shows a zoom-in view of the problem areas. Note that the optimized shape on the

right reduces the error for these triangles while maintaining the boundary and the parameter lines of the original shape.

Additionally we also tested our method on another example of roof that has substantially higher curvature than the last two examples. Figure 40 shows the input dome shaped roof on the left and the optimized shape on the right. Notice that in this case the error (in green) is localized to areas where the smooth curving dome meets the central area. The final optimized shape attempts to smooth the change in triangle shape and size in the areas highlighted in green in the input shape as shown in the left side of Figure 40.

We have also tested our method using only the reflection mapping. Such a mapping tends to lose the parameter lines of the surface and creates triangles that are all similar in shape and size to each other. Figure 28, 32(left) and 41(right) are examples of shapes that are optimized using only the reflection mapping. The results generated using only reflection mapping is similar but significantly better than using a simple linear parallelogram based metric defined by Touma and Gotsman [TG98] in context to mesh compression. The linear parallelogram based error metric is scale independent and works well only for planar meshes. We illustrate the comparison between our non-linear skew-based error function and linear parallelogram based error metric via a couple of simple examples in Figure 41. Note that the parallelogram metric shown on the left side of Figure 41 tends to displace the parameter lines in a non-symmetric manner for both Gaussian bump and the domed architectural roof. While our skew-based metric as shown on the right side of Figure 41 tends to create triangles that vary smoothly in terms of shape and size.

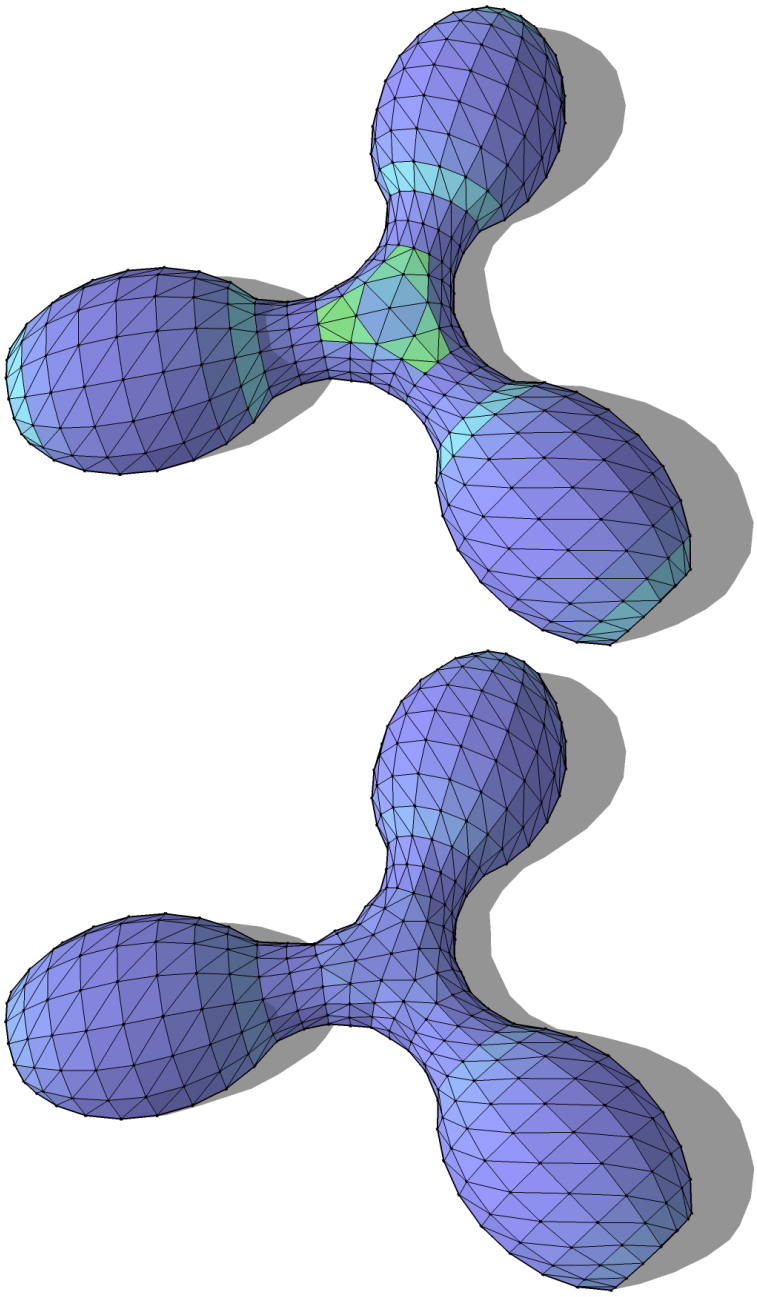


Fig. 40. An example of a dome shaped architectural shape, before (top) and after optimization (bottom). The dissimilar triangles are highlighted in green.

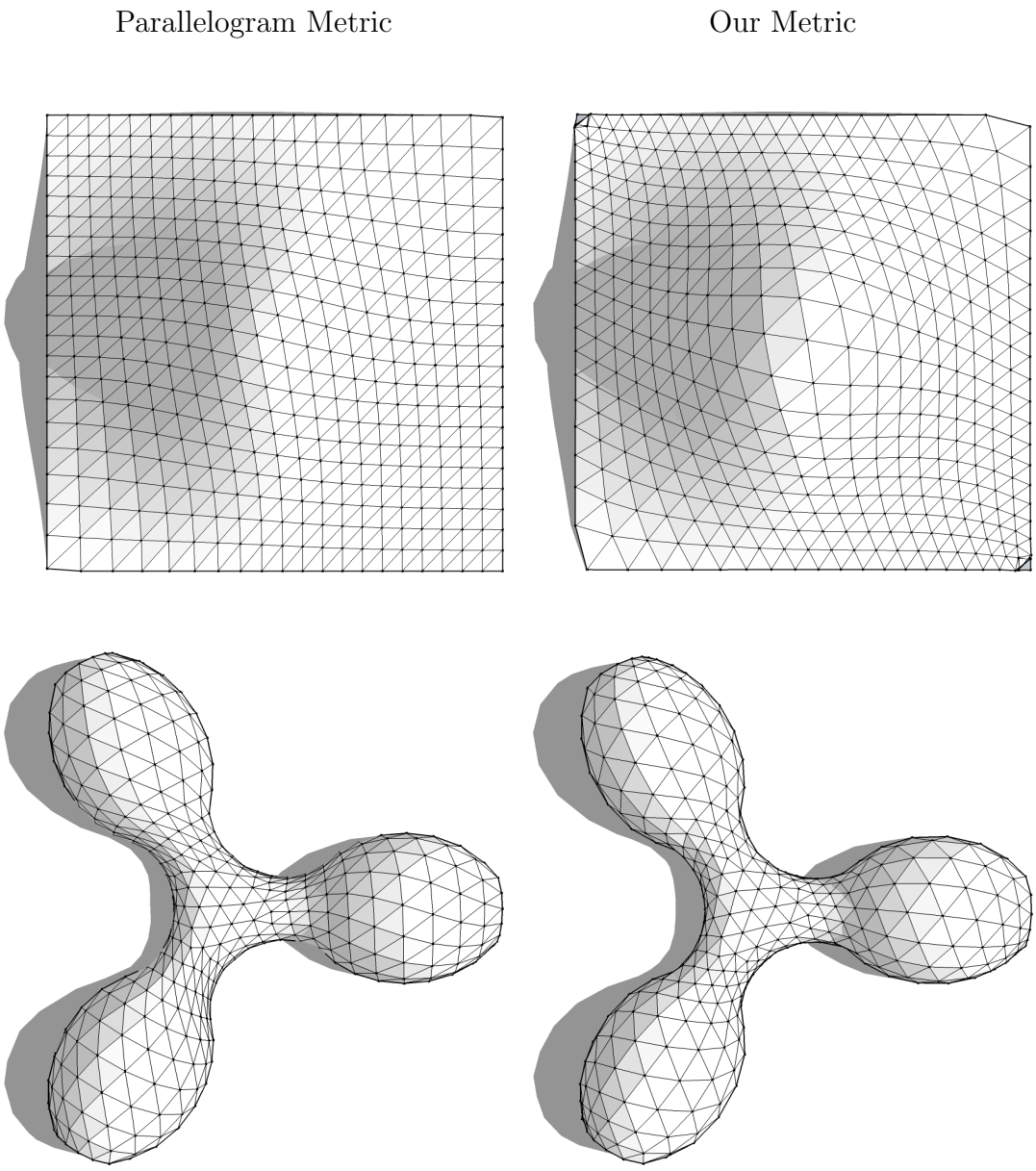


Fig. 41. Comparison of using a parallelogram based metric (left) versus using our skew based error metric with single mapping (right).

The graph in Figure 42 shows a comparison of initial and final error for a set of models. We distinguish the algebraic surface (Gaussian bump, Saddle, Wavy and Parabloid) generated using random set of points from the other since the initial error is extremely high for them.

Our method is meant to be run only once as an off-line preprocessing step. The computational speed of our method is directly proportional to the size of input mesh. As the number of vertices increases the size for Jacobian matrix of the error function is doubles. Computing the inverse of sparse Jacobian matrix of the error function is a bottle neck for our method. Using the analytic Jacobian as described in Appendix B we can take large steps towards the minimum of the function and quickly converge to an approximate solution. Here onwards we can switch to discrete differencing for a finer grain solution. Our method is only as robust as the closest point computation is. This tends to cause problems in cases where two layers of the surface are very close to each other (for example the ears of Bunny in Figure 36).

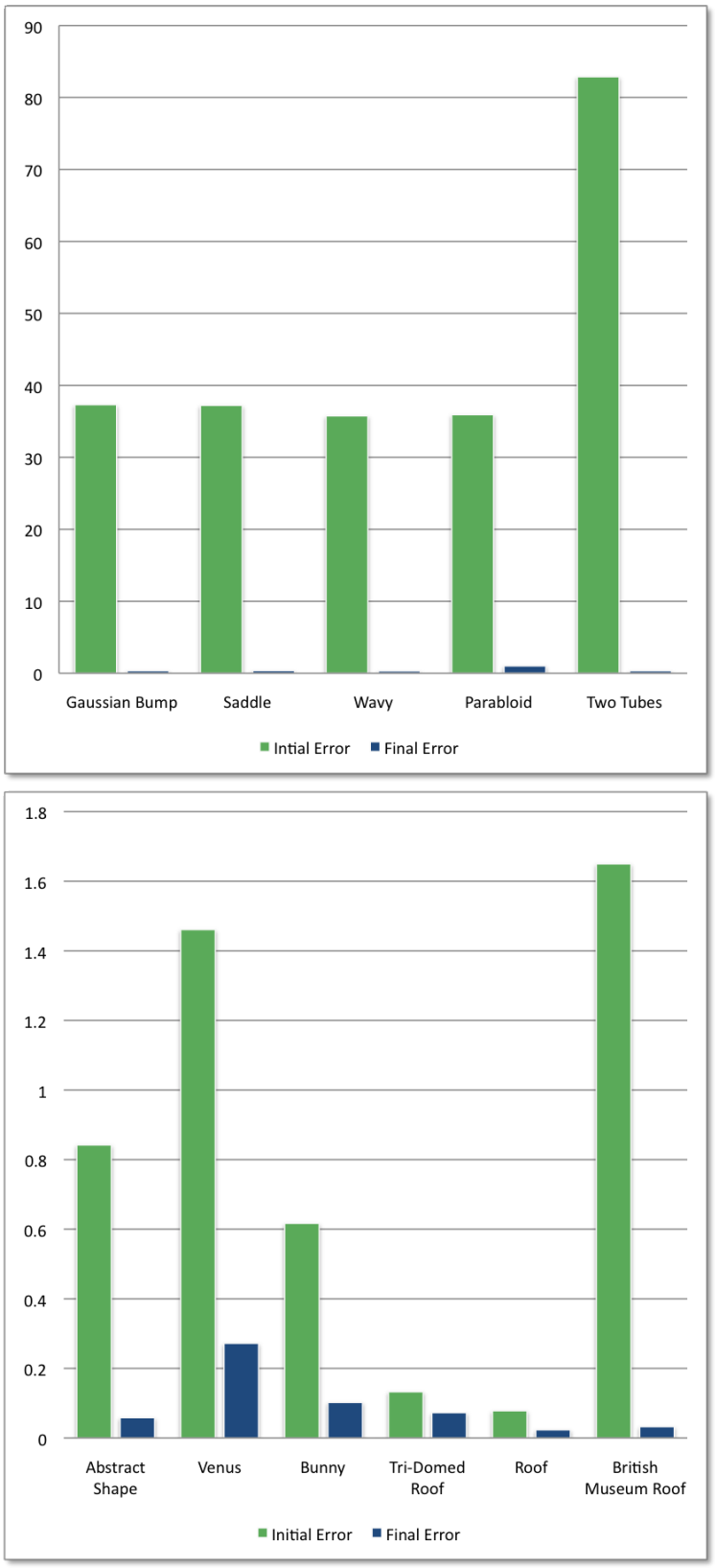


Fig. 42. Comparison of initial (in green color) and final error (in blue color).

I. Conclusion and Future Work

We are able to successfully reduce dissimilarity amongst triangles in a variety of models. The triangles without aiming to look equilateral tends to become visually harmonious as a result of our optimization. We are able to achieve these results while maintaining the geometry as well as the topology of the input shape.

Maintaining smooth parameter lines over the discretized surface is a critical aspect of freeform shapes. Eigenstaz et al. [EKS*10] optimizes freeform shapes for fewer panels while accounting explicitly for parameter lines. We would like to account for parameter lines as well. However since our method operates upon arbitrary shapes it may be difficult to accurately detect parameter lines. For example in shapes such as Venus (Figure 35) or Bunny (Figure 36) - there may not be any parameter lines to detect. How best to robustly account for parameter lines is an aspect we would like explore further in future.

While minimizing the skew metric norm, as the triangle size changes, it may no longer be suitable to depict high frequency details over the surface. Larger triangles may not be able to adequately represent areas of high curvature. A common approach used by CVT liked methods is to insert Steiner points and add more triangles to better approximate local features. An analogous approach in our case would be to perform local adaptive subdivision. This would insert additional set of triangle at high curvatures areas thus providing the non-linear optimization with enough degrees of freedom to better approximate the underlying shape.

CHAPTER V

CONCLUSION

As a part of this work we have developed methods to improve upon the aesthetics as well as to simplify the process of physical fabrication of triangle meshes. We have proposed a non-linear optimization based method to create harmonious triangles in a local neighbourhood. Along with creating an aesthetic mesh we can also cluster a set of triangles in a mesh so as to replace each triangle in the cluster with a single unique canonical triangle. This simplify the process of transport and physical placement of panels.

Advances in design representation and fabrication has opened new avenues for research in context of freeform architecture. By leveraging the improving computational efficiency we can now not only optimize shapes for geometric problems but also for associated physical limitations. For example along with optimizing for triangle panels we can compliment the function with other physically based constraints such as the weight associated with glass panels, the thickness required for structural support of panels.

Whiting et al [WOD09] attempted to look into structural stability of architectural forms. However they viewed the problem from the perspective of traditional masonry based orthogonal architectural shapes. We would like to adapt our geometric algorithms for modern freeform shapes such that the geometry not only adapts to visual or fabrication needs but also to the structural stability of the shape. In future, we would also like to explore metrics that adapt the geometry of a shape based upon feedback from simulations results from other analysis such as lighting, acoustics and energy usage etc.

REFERENCES

[AHB87] Arun K. S., Huang T. S., Blostein S. D.: Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell. 9*, 5 (1987), 698–700.

[AIB08] Alexander I. Bobenko Y. B. S.: *Discrete Differential Geometry : Integrable Structure*. American Mathematical Society, Technische Universität München, Garching bei München, Germany, 2008.

[Aur91] Aurenhammer F.: Voronoi diagrams — a survey of a fundamental geometric data structure. *ACM Comput. Surv. 23*, 3 (1991), 345–405.

[AV07] Arthur D., Vassilvitskii S.: K-means++: The advantages of careful seeding. *Proceedings of the* 18*th annual ACM-SIAM symposium on Discrete algorithms* (2007), 1027–1035.

[Che87] Chew L. P.: Constrained delaunay triangulations. In *Proceedings of symposium on Computational Geometry* (New York, 1987), ACM, pp. 215–222.

[CW07] Cutler B., Whiting E.: Constrained planar remeshing for architecture. In *Proceedings of Graphics Interface* (New York, 2007), ACM, pp. 11–18.

[DFG99] Du Q., Faber V., Gunzburger M.: Centroidal voronoi tessellations: Applications and algorithms. *SIAM Review 41*, 4 (1999), 637–676.

[DGJ02] Du Q., Gunzburger M. D., Ju L.: Constrained centroidal voronoi tessellations for surfaces. *SIAM J. Sci. Comput. 24*, 5 (2002), 1488–1506.

[DH09] Daniel Hambleton H. Y. P. I.: Study of panelization techniques to inform freeform architecture. In *Proceedings of Glass Performance Days* (2009).

[DMP93] Dutta D., Martin R. R., Pratt M. J.: Cyclides in surface and solid modeling. *IEEE Comput. Graph. Appl. 13*, 1 (1993), 53–59.

[DPW08] Dos Passos V. A., Walter M.: 3d mosaics with variable-sized tiles. *Vis. Comput. 24*, 7 (2008), 617–623.

[EKS*10] Eigensatz M., Kilian M., Schiftner A., Mitra N. J., Pottmann H., Pauly M.: Paneling architectural freeform surfaces. In *ACM SIGGRAPH 2010 papers* (New York, 2010), SIGGRAPH '10, ACM, pp. 45:1–45:10.

[EW03] Elber G., Wolberg G.: Rendering traditional mosaics. In *The Visual Computer* (2003), vol. 19, pp. 67–78.

[Far01] Farin G.: *Curves and Surfaces for CAGD*, 5th ed. Morgan Kaufmann, San Francisco, CA, 2001.

[FJNT04] Frandsen P. E., Jonasson K., Nielsen H. B., Tingleff O.: *Unconstrained Optimization, 3rd ed.* Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2004.

[FLHCO10] Fu C.-W., Lai C.-F., He Y., Cohen-Or D.: K-set tilable surfaces. In *ACM SIGGRAPH 2010 papers* (New York, July 2010), vol. 29 of *SIGGRAPH '10*, ACM, pp. 44:1–44:6.

[Gar67] Garbett E. L.: *Rudimentary Treatise on the Principles of Design in Architecture*, 3rd ed. Virtue & Co, London, 1867.

[GS86] Grünbaum B., Shephard G. C.: *Tilings and Patterns*. W. H. Freeman & Co., New York, 1986.

[GS08] Grinspun E., Secord A.: Introduction to discrete differential geometry: The geometry of plane curves. In *SIGGRAPH Asia '08: ACM SIGGRAPH ASIA 2008 courses* (New York, 2008), ACM, pp. 1–4.

[GSC*04] Glymph J., Shelden D., Ceccato C., Mussel J., Schober H.: A parametric strategy for free-form glass structures using quadrilateral planar facets. *Automation in Construction 13*, 2 (2004), 187 – 202. Conference of the Association for Computer Aided Design in Architecture.

[HPK07] H. Pottmann A. Asperl M. H., Kilian A.: *Architectural Geometry*. Bentley Institute Press, Exton, PA, USA, 2007.

[HSH10] He L., Schaefer S., Hormann K.: Parameterizing subdivision surfaces. *ACM Trans. Graph. 29* (July 2010), 120:1–120:6.

[ITF04] Irving G., Teran J., Fedkiw R.: Invertible finite elements for robust simulation of large deformation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2004), SCA '04, Eurographics Association, pp. 131–140.

[KFMP08] Kilian M., Flöry S., Mitra N. J., Pottmann H.: Curved folding. *ACM Trans. Graphics 27*, 3 (2008), 75:1–75:9. Proc. SIGGRAPH.

[KHW05] Klaus Hildebrandt K. P., Wardetzky M.: *On the Convergence of Metric and Geometric Properties of Polyhedral Surfaces.* Tech. rep., Zuse Institute Berlin., 2005.

[Kil06] Kilian A.: *Design Exploration Through Bidirectional Modeling of Constraints.* PhD thesis, Massachusets Institute of Technology, Cambridge, MA, 2006.

[KP02] Kim J., Pellacini F.: Jigsaw image mosaics. In *ACM SIGGRAPH '02* (New York, 2002), ACM, pp. 657–664.

[KS04] Kaplan C. S., Salesin D. H.: Islamic star patterns in absolute geometry. vol. 23, ACM, pp. 97–119.

[LC87] Lorensen W. E., Cline H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, 1987), SIGGRAPH '87, ACM, pp. 163–169.

[LHM06] Lai Y.-K., Hu S.-M., Martin R. R.: Surface mosaics. *Vis. Comput. 22*, 9 (2006), 604–611.

[Lou04] Lourakis M.: levmar: Levenberg-marquardt nonlinear least squares algorithms in C/C++. [web page] `http://www.ics.forth.gr/~lourakis/levmar/`, Jul. 2004.

[LPW*06] Liu Y., Pottmann H., Wallner J., Yang Y.-L., Wang W.: Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graphics 25*, 3 (2006), 681–689. Proc. SIGGRAPH.

[OBS04] Ohtake Y., Belyaev A., Seidel H.-P.: Ridge-valley lines on meshes via implicit surface fitting. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, 2004), ACM, pp. 609–612.

[OBSC00] Okabe A., Boots B., Sugihara K., Chiu D. S. N.: *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2 ed. Wiley, Hoboken, NJ, USA, August 2000.

[PLW*07] Pottmann H., Liu Y., Wallner J., Bobenko A., Wang W.: Geometry of multi-layer freeform structures for architecture. In *ACM SIGGRAPH 2007 papers* (New York, 2007), SIGGRAPH '07, ACM.

[Pol02] Polthier K.: *Polyhedral Surfaces of Constant Mean Curvature*. PhD thesis, Habilitationsschrift, TU-Berlin, Febr. 2002.

[PSB*08] Pottmann H., Schiftner A., Bo P., Schmiedhofer H., Wang W., Baldassini N., Wallner J.: Freeform surfaces from single curved panels. *ACM Trans. Graph. 27* (August 2008), 76:1–76:10.

[Rup95] Ruppert J.: A delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms 18*, 3 (1995), 548–585.

[SÓ5] Séquin C. H.: Cad tools for aesthetic engineering. *Comput. Aided Des. 37*, 7 (2005), 737–750.

[Sau70] Sauer R.: *Differenzengeometrie*. Springer, New York, 1970.

[She96] Shewchuk J. R.: Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, Lin M. C., Manocha D., (Eds.), vol. 1148 of *Lecture Notes in Computer Science*. Springer-Verlag, May

1996, pp. 203–222. From the First ACM Workshop on Applied Computational Geometry.

[She01] Shewchuk J. R.: Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications 22* (2001), 1–3.

[She02a] Shelden D.: *Digital Surface Representation and the Constructibility of Gehrys Architecture*. PhD thesis, Massachusets Institute of Technology, Cambridge, MA, September 2002.

[She02b] Shewchuk J. R.: *What Is a Good Linear Finite Element? - Interpolation, Conditioning, Anisotropy, and Quality Measures*. Tech. rep., In Proc. of the 11th International Meshing Roundtable, 2002.

[SHWP09] Schiftner A., Höbinger M., Wallner J., Pottmann H.: Packing circles and spheres on surfaces. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers* (New York, 2009), ACM, pp. 1–8.

[TG98] Touma C., Gotsman C.: Triangle mesh compression. In *Proceedings of Graphics Interface* (Vancouver, June 1998), pp. 26–34.

[Wil01] Williams C. J. K.: The analytic and numerical definition of the geometry of the british museum great court roof. In *Third International Conference on Mathematics & Design* (2001), Burry M., Datta S., Dawson A., Rollo A. J., (Eds.), pp. 434–440.

[WOD09] Whiting E., Ochsendorf J., Durand F.: Procedural modeling of structurally-sound masonry buildings. *ACM Trans. Graph. 28* (December 2009), 112:1–112:9.

[WRK*10] Wicke M., Ritchie D., Klingner B. M., Burke S., Shewchuk J. R., O'Brien J. F.: Dynamic local remeshing for elastoplastic simulation. *ACM Trans. Graph. 29*, 4 (2010), 1–11.

[WYZG09] Wang L., Yu Y., Zhou K., Guo B.: Example-based hair geometry synthesis. In *ACM SIGGRAPH '09* (New York, 2009), ACM, pp. 1–9.

[YLL*09] Yan D.-M., Lévy B., Liu Y., Sun F., Wang W.: Isotropic remeshing with fast and exact computation of restricted voronoi diagram. In *SGP '09: Proceedings of the Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2009), Eurographics Association, pp. 1445–1454.

[YZX*04] Yu Y., Zhou K., Xu D., Shi X., Bao H., Guo B., Shum H.-Y.: Mesh editing with poisson-based gradient field manipulation. In *ACM SIGGRAPH '04* (New York, 2004), ACM, pp. 644–651.

APPENDIX A

SHORT NOTE ON CONSTRUCTION OF FREE-FORM ARCHITECTURAL SHAPES

According to Dennis Shelden [She02a] fabrication strategy for each project depends upon the choice of materials. The cladding of a surface can be done using transparent rigid material such as glass or opaque flexible metallic sheets made up of stainless steel, aluminum or, for Bilbao titanium.

He mentions that the tooling of dies for extrusion molding or stamping fabrication technology can incur a high fixed cost of product manufacturing. Stamping with metal mold will require creating both negative and positive dies. Creating such a die is a multi-step process. The high cost of creating each mold can be offset if they are used to produce high volume of pre-fabricated parts. This is also true for life-time fixing or replacement of the parts.

He also points out that economies of scale are not only limited to physical production. A subtle implication is that of cost of information required to generate each part. Using Computer Numerically Controlled (CNC) production, unique parts can be custom created with a high degree of precision. Although each unique part comes with the overhead of shop drawing information required for fabrication. The cost of carrying information for each unique part ripples through all aspects of design and construction such as engineering, modelling, drafting even digital storage associated with the development of the part. Fewer unique parts also minimized the chances of discrepancies to occur in the process pipeline. Also, once the engineer has to work with fewer parts, greater attention can be given to each part. More time and effort can be spent towards configuration and fabrication of the mold, than it would be in case where all parts are unique and have to handled separately.

CNC based fabrication can achieve tolerances of 1 millimetre [She02a]. However higher tolerances are introduced during traditional manual assembly methods. Lower tolerance during construction requires higher premium and can be cost prohibitive. On-site digital surveys are carried out to maintain accuracy in placing components in their appropriate position. With higher number of unique parts, more survey samples are required to maintain conformance with the digital model. This can affect both the associated time and cost.

An alternative strategy is to take such construction tolerances into consideration while designing parts. Fewer unique parts will not only simplify the fabrication process but also reduce the sampling overhead of maintaining accuracy for physical placements of panels. Such construction tolerances also take advantage of adjustable connection strategies that are used for resolving dimensional discrepancies between parts. Shelden [She02a] also mentions that the cost of manufacturing primary structural elements to tight tolerance (such as glass) is far greater than that of fabricating secondary system to similar tolerances (steel frame holding the panels).

APPENDIX B

DERIVATIVES FOR SKEW BASED ERROR FUNCTION

The error function described in equation 4.7 and 4.8 consists of two parts. The first one is built using a set of edges and other consists of the area of the two triangles. By decomposing the error equation into parts we have

$$\begin{aligned} a &= -pE0 \cdot pE2 + qE0 \cdot qE2 \\ b &= \Delta_p \\ c &= \Delta_q \end{aligned} \tag{B.1}$$

Furthermore let $t_1 = \frac{a}{b}$ and $t_2 = \frac{c}{b}$. Now we can rewrite the error functions $Err_{(P,Q)}$ as

$$Err^2_{(Q,P)} = f^2 = 3 + \frac{1}{4}t_1^2 + t_2^2 - 2\sqrt{\frac{1}{4}t_1^2 + (1+t_2)^2}$$

For $Err_{(Q,P)}$ we simply switch the values of b and c. Now we can easily write the derivative of f with respect to any vertex X as

$$\frac{\partial f}{\partial X} = \frac{1}{2\sqrt{f}}\left(\frac{1}{2}(t_1 t_1') + 2(t_2 t_2') - \frac{\frac{1}{2}(t_1 t_1') + 2(1+t_2)t_2'}{\sqrt{\frac{1}{4}t_1^2 + (1+t_2)^2}}\right)$$

where

$$\begin{aligned} t_1' &= \left(\frac{a'b - ab'}{b^2}\right) \\ t_2' &= \left(\frac{c'b - cb'}{b^2}\right). \end{aligned}$$

In order to compute Jacobian of the function with respect to each vertex of triangle P and Q we need to compute derivatives of a, b and c.

$$\frac{\partial a}{\partial p_0} = 2p_0 - p_1 - p_2$$

$$
\begin{aligned}
\frac{\partial a}{\partial q_0} &= 2q_0 + q_1 + q_2 \\
\frac{\partial b}{\partial p_0} &= \frac{1}{2}(pE0\cot\theta_0 + pE2\cot\theta_2) \\
\frac{\partial b}{\partial q_0} &= \{0.0, 0.0, 0.0\} \\
\frac{\partial c}{\partial p_0} &= \{0.0, 0.0, 0.0\} \\
\frac{\partial c}{\partial q_0} &= \frac{1}{2}(qE0\cot\phi_0 + qE2\cot\phi_2) \\
\frac{\partial a}{\partial p_1} &= -p_0 - p_1 \\
\frac{\partial a}{\partial q_1} &= q_0 - q_2 \\
\frac{\partial b}{\partial p_1} &= \frac{1}{2}(pE1\cot\theta_1 + pE0\cot\theta_0) \\
\frac{\partial b}{\partial q_1} &= \{0.0, 0.0, 0.0\} \\
\frac{\partial c}{\partial p_1} &= \{0.0, 0.0, 0.0\} \\
\frac{\partial c}{\partial q_1} &= \frac{1}{2}(qE1\cot\phi_1 + qE0\cot\phi_0) \\
\frac{\partial a}{\partial p_2} &= -p_0 + p_2 \\
\frac{\partial a}{\partial q_2} &= q_0 - q_1 \\
\frac{\partial b}{\partial p_2} &= \frac{1}{2}(pE2\cot\theta_2 + pE1\cot\theta_1) \\
\frac{\partial b}{\partial q_2} &= \{0.0, 0.0, 0.0\} \\
\frac{\partial c}{\partial p_2} &= \{0.0, 0.0, 0.0\} \\
\frac{\partial c}{\partial q_2} &= \frac{1}{2}(qE2\cot\phi_2 + qE1\cot\phi_1)
\end{aligned}
$$

VITA

Name	Mayank Singh
Address	7355 Hillendale Drive,
	Franklin WI 53132
Email Address	mayanksingh@tamu.edu
Education	Ph.D. Computer Science, Texas A&M University, 2011
	M.S. Computing, Marquette University, 2002
	B.S. Architecture, Indian Institute of Technology, 2000