Spanning Tree Seams for Reducing Parameterization Distortion of Triangulated Surfaces

Alla Sheffer
Department of Computer Science
Technion, Haifa, Israel
e-mail:sheffa@cs.technion.ac.il

Abstract

Providing a two-dimensional parameterization of three-dimensional tesselated surfaces is beneficial to many applications in computer graphics, finite-element surface meshing, surface reconstruction and other areas. The applicability of the parameterization depends on how well it preserves the surface metric structures (angles, distances, areas). For a general surface there is no mapping which fully preserves these structures. The distortion usually increases with the rise in surface complexity. For highly complicated surfaces the distortion can become so strong as to make the parameterization unusable for application purposes. One possible solution is to subdivide the surface or introduce seams in a way which will reduce the distortion.

This article presents a new method for introduction of seams in three-dimensional tesselated surfaces. The addition of seams reduces the surface complexity and hence reduces the metric distortion produced by the parameterization. Seams often introduce additional constraints on the application for which the parameterization is used, hence their length should be minimal. The new method we present minimizes the seam length while reducing the parameterization distortion.

1. Introduction

A two-dimensional parameterization of three-dimensional surfaces is useful for many applications. These include texture mapping for computer generated images [8, 18, 13], finite-element surface meshing [17, 14], surface reconstruction [7], multiresolutional analysis[5], generation of clothing patterns [15], and metal forming.

When the surface is represented using an analytic description, this description can often be used to provide the parameterization. However, many computer graphics and CAD models are represented by a triangular tessellation and the analytic representation of the surface is often unavailable. A triangulated mesh is the "natural" description of surfaces constructed from scattered data such as input from laser range scanners or sampling on a regular three-dimensional grid.

An algorithm for two-dimensional parameterization or *flattening* of tessellated surfaces first constructs a two-dimensional mesh with a similar connectivity to the three-dimensional surface. Then, a parametric function is defined between the two-dimensional mesh.

Multiple approaches for parameterization of tessellated surfaces have been suggested [2, 5, 7, 14, 13, 10, 17, 16]. Most provide good results for relatively simple surfaces, but can generate invalid parameterization for some inputs. A few methods [7, 10, 17] guarantee that the resulting flat mesh is valid for any legal input (where a legal input is a surface which is homeomorphic to a disk with a single boundary loop).

The major criteria when considering the suitability of the parameterization for different applications is the preservation of the surface metric structures (angles, distances, areas). Angular and distance/area distortions are detrimental to most applications which use the parameterization, since features in two dimensions are not preserved when projecting them onto the original three-dimensional surface. The best example is texture mapping, where the quality of the texture relies directly on the metric preservation in the parameterization.

For a general triangulated surface there is no mapping which fully preserves distances or areas [1]; neither is there a mapping that for a faceted surface with non-zero curvature fully preserves angles. Hence, parameterization methods can only attempt to minimize both types of distortion. Note that minimizing the two types of distortion independently will provide different parameterizations and it is up to the specific parameterization method to decide which combination of the distortions should be preferred. The distortion depends directly on the Gaussian curvature of the surface. A surface with zero Gaussian curvature is developable, i.e. it has a two-dimensional parameterization with zero distortion. The Gaussian curvature and the distortion can be reduced by cutting the surface, especially across regions (points) of high curvature (Figure 4). This influence can be seen directly in the formulation of many of the flattening or parameterization methods. For example, the method of Sheffer and de Sturler [17] (Section 4) uses a constrained minimization formulation to obtain the flat triangulation and each seam edge added to the surfaces reduces the number of constraints in the formulation. Cutting of surfaces, or *segmentation* has had numerous applications [5, 11, 6] in the past. However the main purpose of these has been generation of fair sub-meshes suitable for decimation or surface reconstruction. In the case of distortion reduction the main concern is the generated mesh boundary shape in terms of curvature and length. Hence these methods cannot be applied directly to this problem. Moreover, parameterization does not require the surface to be segmented into several parts. Seams which cut partially into the surface are often acceptable.

Contribution

In this work we provide a method for automatic introduction of surface seams which cut through regions and points of high curvature. At the same time, since seams modify and extend the boundary of the domain, they often add constraints or adversely affect the application for which the parameterization is used. For texture mapping, for example, they cause discontinuities in the texture across the seam. For surface meshing they enforce conformal placement of mesh nodes along the seam boundary, making the process more cumbersome and often adversely affecting the mesh quality. Hence when introducing seams two considerations have to be taken into account: distortion reduction and minimization of seam length. The new method we present considers both. The method first selects a set of interior mesh nodes through which the seams will pass. The choice of the nodes is based on curvature considerations (Section 2). After the nodes are selected, a Minimal Spanning Tree (MST) algorithm is applied on the graph associated with the mesh to generate the shortest possible set of seams which will connect the selected nodes to the boundary. This step is described in Section 3. The first step is based on the parameterization requirement, in terms of mesh connectivity and parameterization distortion. The second step ensures the minimal length of the seams.

Most popular parameterization methods [5, 7, 17] require the surface to have a single boundary loop. Hence, to parameterize a surface with multiple loops, we need to cut it first in order to remove interior loops. The spanning tree method introduces seams which eliminate all interior loops based on the method definition.

As will be demonstrated in the examples given later on, the method dramatically reduces the parameterization distortion while adding minimal length seams. It is robust and efficient. To the author's knowledge this is the first attempt to introduce a fully automatic and systematic way of cutting surfaces to minimize the distortion which occurs when flattening surface.

The rest of the paper is organized as follows. The choice of surface nodes to be added to the boundary is explained in Section 2. Section 3 describes the spanning tree algorithm used to generate the seams connecting the nodes to the boundary. The combination of seam cutting with a texture mapping procedure is demonstrated in Section 4. Section 5 concludes the paper and discusses the method properties and future work.

2. Node Selection

The first step of the algorithm is the selection of interior nodes which should be part of the seams. The nodes are selected based on Gaussian curvature around each node. The node Gaussian curvature provides a lower bound on the local distortion during parameterization. Note that for distortion purposes only, the curvature at the nodes is of interest. The edges do not contribute to the distortion since any surface is locally developable around each edge (Figure 1).

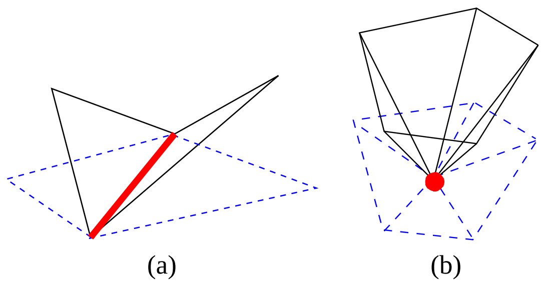

Figure 1. Local parameterization in a plane: surface is locally developable around an edge (a), around a vertex it generally is not (b).

Gaussian curvature is not formally defined for non-smooth surfaces, hence only a discrete approximation can be computed. Approximations for computing it across a mesh region tend to be costly [9, 12]. In this work we use an alternative measure of curvature which is easier to compute. The suggested measure is roughly based on the deviation of the sum of angles around a node from $360°$, i.e. on the local deviation of the surface from a plane.

We define a *region curvature* C_r of magnitude r at an interior mesh node n as follows (Figure 2).

- C_0 is the ratio between the sum of face angles α_i around the node and the sum of angles around an interior node in a planar mesh (Figure 2(a)):

$$C_0(n) = \frac{\sum_i \alpha_i}{2\pi}.$$

- To define a curvature over a region we first define a region radius R as a product of the magnitude r and the length of the longest edge emanating from n

  $$R = r \max_{e=(n,m)} (\|n - m\|).$$

  Given the radius, a sub-region of the mesh $M_r(n)$ is found such that all the nodes in $M_r(n)$ are within distance R from n (Figure 2(c)). We then consider the triangles t_j formed by the boundary edges of $M_r(n)$ and n (Figure 2(d)). The curvature is computed as

  $$C_r(n) = \frac{\sum_j \tau_j}{2\pi}$$

  where τ_j are the angles at n of t_j (Figure 2(d)). This gives a relative curvature around a node for a given region size.

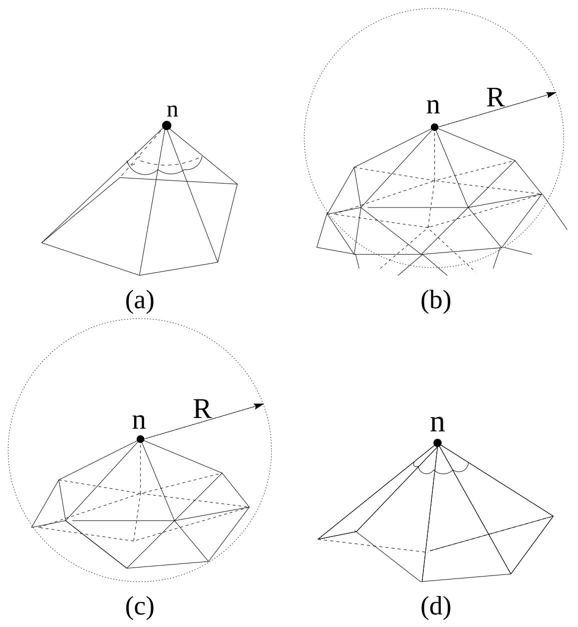

Figure 2. Computing $C_r(n)$. (a) Computing $C_0(n)$ - summation of angles around a node. (b)-(d) Computing $C_r(n), r > 0$. (b) The mesh around a node and the radius R sphere. (c) The mesh sub region $M_r(n)$ contained by the sphere. (d) The triangles formed by the boundary edges of $M_r(n)$ and n.

For $r = 0$ this measure is quite similar to the Gaussian curvature approximation suggested by Calladine [3]. The main difference is that it does not take into consideration the area of the surrounding triangles. Hence it is scale independent, and therefore, is more appropriate as a parameterization distortion indicator. We use the values of $C_r(n)$ to select the nodes which should be added to the boundary node set. Given user defined threshold values $\tilde{C_0}$, $\tilde{C_0'} > \tilde{C_0}$, r, and $\tilde{C_r}$, the selection is done as follows.

- Compute $C_0(n)$ at all interior nodes.
- Set the node set S' to include the interior mesh nodes where $C_0(n) < \tilde{C_0}$.
- Set the node set S'' to include the interior mesh nodes where $\tilde{C_0} < C_0(n) < \tilde{C_0'}$.
- For each node n in S'' compute $C_r(n)$. If $C_r(n) < \tilde{C_r}$, add n to S'.

This selection process gives all the nodes with very high local curvature, and nodes with both relatively high local and high region curvature. The values for $\tilde{C_0}$, $\tilde{C_0'}$, and $\tilde{C_r}$ used throughout the paper were $.9, .95$, and $.9$, respectively. The choice of r varies throughout the examples and is specified near each example figure.

3. Generating Seams

Once the nodes are selected the shortest possible seams which will add them to the boundary need to be computed. To do this we consider the weighted graph (V, E)associated with the mesh, where edge lengths correspond to arc weights w. To compute the shortest seams we construct the Minimal Spanning Tree (MST) of the chosen nodes together with the boundary nodes. The construction process is explained below.

3.1. Algorithm Steps

After the selection of a set of interior nodes S', the method analyzes the mesh graph $M = (N, E)$ to generate the seams. The main steps of the algorithm are described below and are demonstrated in Figure 3.

- Generate a node set S containing the selected interior nodes S' and all the boundary nodes of the mesh (Figure 3(a)).
- Compute the shortest paths between each pair of nodes in S (Section 3.2 (Figure 3(b)).
- Compute the Minimal Spanning Tree (MST) of S (Figure 3(c)). A *Minimal Spanning Tree (MST)* of a subset of the graph vertices S is the collection of paths that join all vertices in S together, with the minimum possible sum of arc weights.

- Add seams into the mesh along the MST edges.

After adding the seams, the surface can be opened up along them during parameterization (Figure 3(d)).

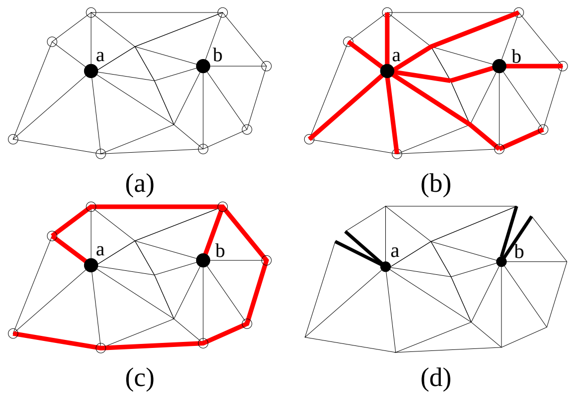

Figure 3. Cutting seams in the mesh. (a) The node set S containing the nodes a and b and all the boundary nodes. (b) The shortest paths from a to all other nodes in S. Only this subset of paths is shown for clarity. (c) The MST of S. (d) Splitting the mesh along the seams.

3.2. Computing the Shortest Paths

In this and the next section two well-known graph theory algorithms are applied to add the shortest possible seams to the mesh, given the set of nodes that the seams must contain. After the node set S containing the selected interior nodes and the boundary nodes is computed, the shortest paths between each pair of nodes in S are computed.

```
1. Let $T = S$
2. While $T \neq \emptyset$
    2.1 Get $t \in T$
    2.2 $T = T \setminus \{t\}$
    2.3 Compute the shortest paths from $t$ to all the nodes
        in $T$
```

To compute the shortest path from t to a set of nodes T, a modified Dijkstra's algorithm[4] is used. It uses a priority queue Q of paths and associated path weights. The operation $ExtractMin(Q)$ returns the path with the lowest weight in the queue. The algorithm is defined as follows.

```
1. $I = \{t\}$
2. For each $e = (t, t')$
    2.1 If $e$ on boundary $w(e) = \epsilon$,
        else $w(e) = \|e_1 - e_2\|$
    2.2   $Q \leftarrow (e, w(e))$
3. While $Q \neq \emptyset$
    3.1 $(P, w(P)) = ExtractMin(Q)$
    3.2 $P = P(t, p)$
    3.3 If $p \in I$ goto 3
    3.4 $I = I \cup \{p\}$
    3.5 If $p \in T$
        - $T = T \setminus \{p\}$
        - Record P(t,p) and w(P)
        - If $T = \emptyset$ **terminate**
    3.6 For each $e = (p, p')$ if $p' \notin I$
          $Q \leftarrow (P \cup \{e\}, w(P) + w(e))$
```

The setting of boundary edge weights to ϵ in step 2.1 is made to ensure that the boundary lengths do not affect the choice of seam edges (since they do not influence the final seam lengths). With the setting to ϵ we ensure that all boundary nodes are connected along the perimeter and then the shortest seams to interior nodes from the boundary are chosen.

Using a binary heap for the priority queue representation Dijkstra's algorithm runs in $O(|E|\log(|N|))$ time (where $|E|$ is the number of edges and $|N|$ number of vertices in the mesh). The standard algorithm computes the paths from a vertex to all other graph vertices. In our case only the cost of the paths from t to a set of vertices T is of interest, hence the modification of the standard Dijkstra in 3.5. Due to the modification, the code will terminate earlier, but the speedup depends on the mesh structure and the distribution of the chosen nodes.

3.3. Minimal Spanning Tree

After finding the shortest paths $P(S)$ between each pair of nodes in S we can compute the minimal spanning tree of the nodes which will provide the seams for the surface cutting. We use Kruskal's algorithm [4] to compute the tree $T(S)$. The algorithm uses a disjoint set data structure to maintain several disjoint sets of elements. Each set contains the vertices in a tree of the current forest. The operation $MakeSet(v)$ creates a set for a single node tree containing v. $FindSet(v)$ returns the set that contains v. The operation $Unite(v, u)$ combines the two trees, uniting the sets.

```
1. $T(S) = \emptyset$
2. For each $s \in S$, $MakeSet(s)$
3. Sort the paths $P(S)$ by nondecreasing weight order
```

```
4. For each path $P(s_1, s_2) \in P(S)$ in the order of non-decreasing weights
   4.1 If $FindSet(s_1) \neq FindSet(s_2)$
       – $T(S) = T(S) \cup P(s_1, s_2)$
       – $Unite(s_1, s_2)$
```

Choosing an efficient implementation of the disjoint set data structure [4] this algorithm can run in $O(|P(S)| \log(|P(S)|))$. Since $P(S)$ contains paths between every pair of nodes in S, $|P(S)| = |S|^2$. In practice the size of the node set $|S|$ is close to the number of boundary nodes, as only a few interior nodes are chosen in the first step of the algorithm. The generated minimal spanning tree contains both interior mesh edges and edges on the boundary (connecting the boundary nodes which are all included in S). To generate the seams we first remove all boundary edges and then proceed to split the mesh, duplicating each interior mesh edge which corresponds to an arc in the MST.

4. Seams and Texture Mapping

The algorithm described above was tested in conjunction with the parameterization method introduced by Sheffer and de Sturler [17, 16]. The parameterization method is briefly described below. The parameterization was then used to generate texture on geometric models in order to demonstrate the effect of seams on texture quality.

4.1. Angle Based Flattening

The Angle Based Flattening (ABF) algorithm [17] is based on the observation that a planar triangular mesh is defined by the angles within each triangle, up to global scaling, rotation or translation. Based on this observation, the method formulates the flattening problem solely in terms of the planar triangulation angles. The algorithm minimizes the relative distortion of the planar angles with respect to their counterparts in the three dimensional surface, while satisfying a set of constraints that ensure the validity of the flat mesh. This formulation provides a constrained minimization problem defined entirely in terms of flat mesh angles; the locations of the flat mesh nodes do not play a role. The solution of this problem provides the set of angles which determine the flat triangulation. The ABF formulation does not require the two-dimensional boundary to be predefined and does not place any restrictions on the boundary shape or the surface curvature. At the same time the solution method based on this formulation is provably correct, i.e. the minimization procedure is guaranteed to converge to a valid solution. Sheffer and de Sturler [16] adopted the method for use in texture mapping, by adding a post-processing step which minimizes the distortion of lengths in addition to angles.

4.2. Texture Mapping Examples

We demonstrated the seam generation procedure and its effect on surface parameterization on three models of varying complexity. After generating the seams the models were parameterized as described in [17]. The parameterization was then used for texture mapping [16]. The quality of the texture directly depends on the parameterization distortion. The texture quality is also affected by seam length and location, since texture discontinuities are generated along the seams.

Figure 4 shows a cat model (without the planar base). The flattening algorithm can handle the model successfully, but the parameterization distortion is high due to large Gaussian curvature over the surface. As a result, the texture distortion becomes unacceptable (Figure 4(c)). The seam introduction method with $r = 1$ generates a seam from the base to the cat's head, adding nodes on the ear tips as the extreme curvature points. The parameterization distortion is significantly reduced and the new texture mapping preserves lengths and angles pretty well (Figure 4 (d)-(f)). In (Figure 4 (g)-(i)) we increase the value of r to 2. This adds two more nodes on the cat's nose to the node set S'. The resulting texture is slightly better with more even texture pattern. However, the seams are slightly longer (with 60 edges instead of 53 as in the first example). Both runs took less than a second each.

In Figure 5 we generate texture for a rabbit model. The model has 898 faces. The node set S' is generated using $r = 1$ and contains 50 nodes. These include points on the rabbit's ears, nose, tail and feet . The final seams contain 100 edges and take about 2 seconds to generate. Figure 6 shows a model of a cow. It is the largest and most complicated of the three models, with 5803 faces. The node set for it contains 98 nodes (using $r = 1$) and includes such sharp features as horns, ears, tail and hoofs. The final seams contain 326 edges. The seam generation time increases quite significantly due to increases in both model and node set sizes and is close to two minutes (1:57). Such an increase is to be expected as Dijkstra's algorithm for computing the paths between each pair of nodes is $O(|E| \log(|N|))$ (E-edges and N-nodes) and the MST algorithm is nearly quadratic in the number of nodes in S'.

5. Conclusions

A method for automatic generation of seams in faceted surface models was presented. The method generates seams which reduce the distortion when parameterizing the models. At the same time it keeps the seam length minimal, given the distortion reduction requirements. The method had been tested successfully in conjunction with a parameterization scheme, and was used for texture mapping on

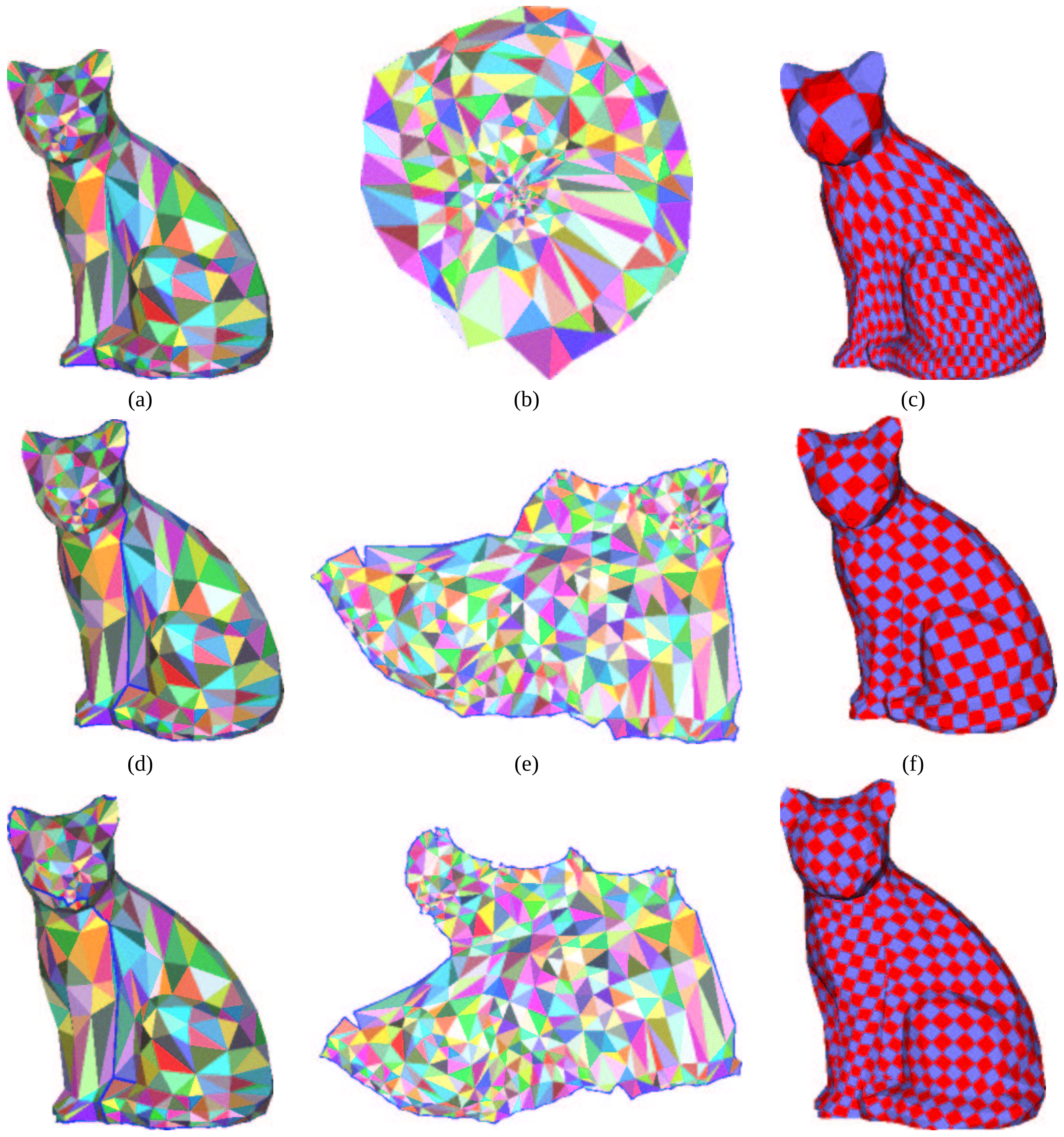

Figure 4. Texture mapping for a cat model (without the planar base). (a) The model (671 elements). (b) Flat parameterization. (c) Generated texture. (d) The model with seams (in blue) with $r = 1$. (e) Flat parameterization containing seams. (f) Surface texture. (g) The seams generated with $r = 2$. (h) Flat parameterization containing the seams. (i) Surface texture ($r = 2$).

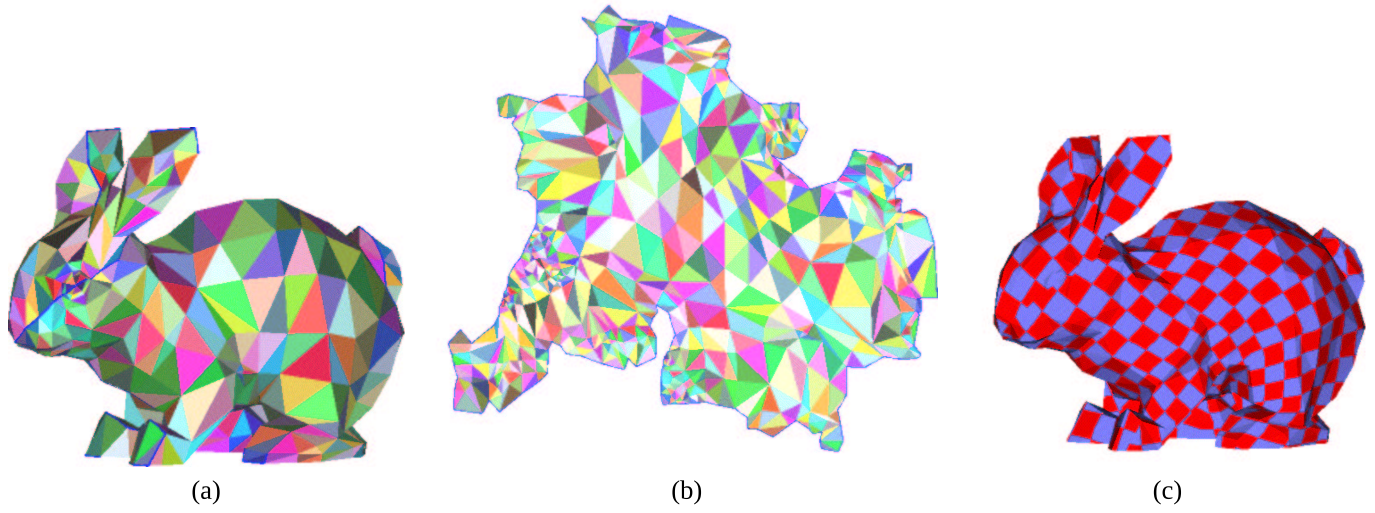

Figure 5. Texture mapping for a rabbit model (898 faces). (a) The model with seams (in blue). (b) Flat parameterization. (c) Generated texture.

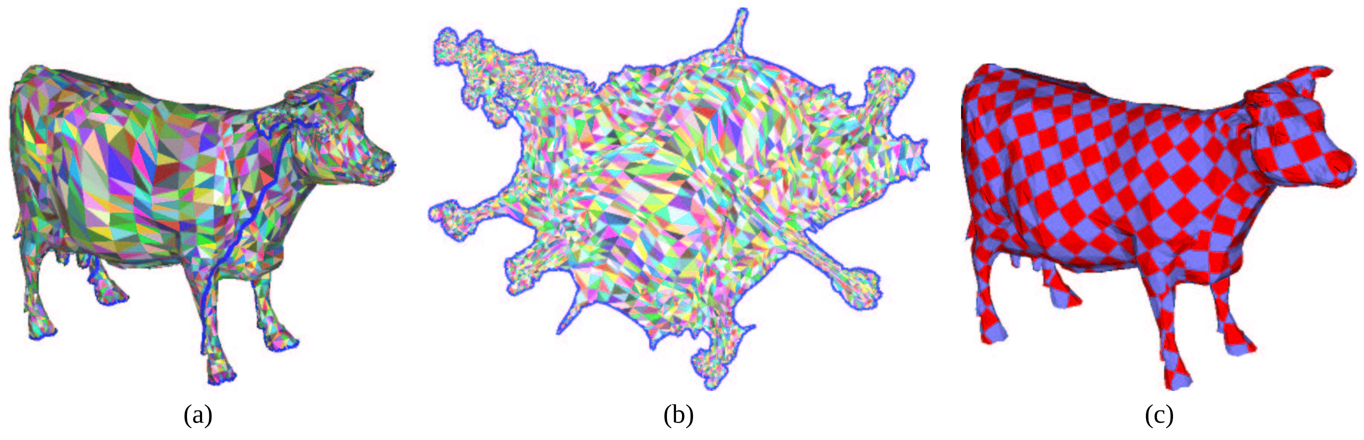

Figure 6. Texture mapping for a cow model (5803 faces) (a) The model with seams (in blue). (b) Flat parameterization. (c) Generated texture.

complex surface models.

There are several interesting future extensions to the method. The first is generating seams based on visibility criteria so as to hide the seam from view, when possible. Another one is handling models with genus higher than zero, such as the torus, by adding seams which will make the model homeomorphic to a disk.

6. Acknowledgments

The author would like to thank Sariel Har-Peled for his advice on the graph theory aspects of the algorithm. This research was partly supported by the Center for Process Simulation and Design (NSF DMS 98-73945) and the Center for Simulation of Advanced Rockets (DOE LLNL B341494).

References

[1] L. V. Ahlfors and L. Sario. *Riemann Surfaces*. Princeton University Press, Princeton, New Jersey, 1960.

[2] C. Bennis, J. M. Vézien, and G. Iglésias. Piecewise surfaces flattening for non-distorted texture mappinng. *Proc. SIGGRAPH 91*, 25:237–246, 1991.

[3] C. R. Calladine. Gaussian curvature and shell structures. In J. Gregory, editor, *Mathematics of Surfaces*, pages 179–196. Clarendon Press, Oxford, 1986.

[4] T. H. Cormen, C. E. Lieseron, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, 2000.

[5] M. Eck, T. DeRose, T. Duchamp, M. Hoppe, H.and Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. *Computer Graphics (Annual Conference Series, 1995. SIGGRAPH '95)*, pages 173–182, 1995.

[6] J.-T. Fan, G. Medioni, and R. Nevatia. Segmented descriptions of 3d surfaces. *IEEE Journal of Robotics and Automation*, 3 (6):527–538, 1987.

[7] M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14:231–250, 1997.

[8] S. Haker, S. Angenent, R. Tannenbaum, A. andKikinis, G. Sapiro, and M. Halle. Conformal surface parameterization for texture mapping. *IEEE Transactions on Visualization and Computer Graphics*, 6 (2):181–189, 2000.

[9] B. Hamann. Curvature approximation for triangulated surfaces. *Computing Suppl.*, 8:139–153, 1993.

[10] K. Hormann and G. Greiner. Mips: an efficient global parameterization method. In *Curve and Surface Design: St. Malo 1999*, pages 153–162. Vanderbilt University Press, 2000.

[11] A. D. Kalvin and R. Taylor. Superfaces: Polygonal mesh simplification with bounder error. *IEEE Computer Graphics and Applications*, 16 (3):64–77., 1996.

[12] L. Kobbelt. Discrete fairing and variational subdivision for free-form surface design. *Visual Computer*, 16(3/4):142–158, 2000.

[13] B. Levi and J. Mallet. Non-distorted texture mapping for sheared triangulated meshes. *Proc. SIGGRAPH 1998*, pages 343–352, 1998.

[14] D. L. Marcum and J. A. Gaiter. Unstructured surface grid generation using global mapping andphysical space approximation. *8th International Meshing Roundtable*, pages 397–406, 1999.

[15] J. McCartney, B. K. Hinds, and B. L. Seow. The flattening of triangulated surfaces incorporating darts and gussets. *Computer-Aided Design (CAD)*, 31:249–260, 1999.

[16] A. Sheffer and E. de Sturler. Non-distorted texture mapping usingangle based flattening. *Tech. Report UIUCDCS-R-2001-2257 / UILU-ENG-2001-1764, November 2001, Submitted to ACM TOG*, 2001.

[17] A. Sheffer and E. de Sturler. Surface parameterization for meshing using angle-based flattening. *Engineering with Computers*, 17 (3):326–337, 2001.

[18] G. Zigelman, R. Kimmel, and N. Kiryati. Texture mapping using surface flattening via multi-dimensionalscaling. *CIS report CIS-2000-01, submitted to IEEE Trans. on Visualization and Computer Graphics*, 2000.