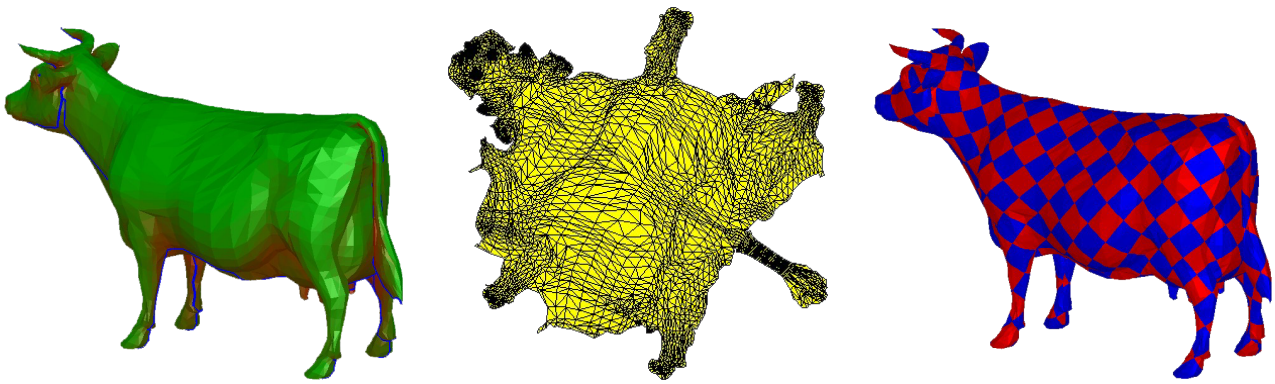
Seamster: Inconspicuous Low-Distortion Texture Seam Layout

Alla Sheffer *
Department of Computer Science
Technion I.I.T, Haifa, Israel

John C. Hart †
Department of Computer Science
University of Illinois at Urbana-Champaign



(a) (b) (c)

Figure 1: Seamster efficiently figures out where to cut an object to flatten it. Figure (a) shows that the seam (in blue) sneaks through low visibility regions of the model (in red) to cut into high curvature vertices, reducing the distortion of the flattening (b) and texturing (c).

ABSTRACT

Surface texturing aids the visualization of polygonal meshes by providing additional surface orientation cues and feature annotations. Such texturing is usually implemented via texture mapping, which is easier and more effective when the distortion of the mapping from the surface to the texture map is kept small.

We have previously shown that distortion occurs when areas of high surface curvature are flattened into the texture map. By cutting the surface in these areas one can reduce texture map distortion at the expense of additional seam artifacts.

This paper describes a faster technique for guiding a texture map seam through high distortion regions, while restricting the seam to regions of low visibility. This results in distortion reducing seams that are less visually distracting and take less time to compute. We have also observed that visibility considerations improve the speed of a recent method that adds cuts to reduce a surface genus.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture.

Keywords: Texture Mapping, Visibility Classification

1 INTRODUCTION

Texture mapping is a useful tool in the visualization of surfaces. Texturing capitalizes on the perceptual cue of size constancy to aid the user in perceiving depth. The higher the spatial frequency of a texture, the farther away the surface appears. Texturing can also be used to annotate a surface, with grid lines, text, markers, glyphs or other features.

Texture mapping requires the assignment of texture coordinates to a meshed surface, but meshes in visualization applications are often unparameterized and unstructured. A variety of techniques have been devised to automatically generate texture coordinates for such meshes. Many of these techniques strive to minimize the distortion of the texture mapping, preserving the proportions of features in the texture as they are mapped to the object surface.

A majority of these techniques generate the texture by mapping the surface into the plane and using it as a two dimensional texture map. These techniques assume that the surface mesh has already been cut into a simply connected two-dimensional manifold-with-boundary. Once such a cut has been performed, the mesh can be flattened into the domain of the texture map. The selection of this cut is often either arbitrary or left to the intuition of the user.

The cut results in a texture seam, which can appear as a discontinuity in the appearance of the texture on the object surface. Hence cuts should be made judiciously and kept short when possible. Users often hide the seam by placing it in an area of the model that is least seen. For example, texture maps for animals typically have their seam along the underbelly.

The layout of a two-dimensional texture map onto the surface of a given three-dimensional object inevitably creates distortion in all but special cases. It is a well-known differential geometry result that for a general surface patch there is no distance-preserving (isometric) parameterization in the plane [Ahlfors and Sario 1960]. Distance-preserving parameterizations exist only for developable surfaces, i.e. surfaces with zero Gaussian curvature such as the cylinder.

The strategic placement of seams can reduce this distortion. Cut-

*sheffa@cs.technion.ac.il
†jch@uiuc.edu

ting a seam through areas of high Gaussian curvature can relieve the tension induced by flattening, reducing the local distortion of the area's texture at the expense of introducing a texture seam. For example, user-designed cuts of animal models often extend to the high-curvature extremeties (e.g. fingers, feet, tail).

We have previously developed a seam-cutting method that finds vertices with high Gaussian curvature and forces the seam to pass through these vertices [Sheffer 2002]. This method tried to minimize the impact of the seam by finding an approximate shortest cutting path (actually a tree) through the high curvature vertices. Unfortunately, this approximate solution was still slow, taking up to an hour on the large meshes found in visualization applications, and the seam it generated passed conspicuously through highly visible regions of the mesh.

Seamster is an improvement designed to increase the speed and the visual quality of automatic texture coordinate synthesis. Like its predecessor, Seamster cuts the surface into a low distortion patch by cutting through high curvature regions of the surface (reviewed in Section 5), though Seamster uses a faster technique, described in Section 6, for finding approximate minimal paths that runs in a few minutes on the large meshes used in visualization applications. Unlike its predecessor, Seamster minimizes the visual impact of the seam by guiding the seam through less visible parts of the surface, using a technique described in Section 4. Seamster can handle manifolds of any genus by using a cutting technique recently introduced by Erickson and Har-Peled [2002]. Section 6.3 shows that visibility considerations can significantly accelerate the process of finding genus reducing cuts.

2 Related Work

The layout of a texture map onto a given object surface induces a mapping of the object surface into a texture domain. Most existing automatic and interactive techniques (e.g. [Samek 1986; Ma and Lin 1988; Bennis et al. 1991; Maillot et al. 1993; Levy and Mallet 1998; Hurdal et al. 1999; Levy et al. 2002; Sheffer and de Sturler 2002; Desbrun et al. 2002]) generate texture coordinates by providing a mapping to a two dimensional texture domain. These techniques require such a mapping to exist, i.e. they expect the surface to be an open two-manifold, topologically equivalent to a disk. Using this assumption, the methods focused only on the geometry of the mapping, leaving the mesh conectivity as is. They treated distortion as an energy functional, and collectively applied a variety of different energy minimization techniques to assign texture coordinates to mesh vertices.

Several works (e.g. [Haker et al. 2000]) suggested using a spherical texture domain, which allows seamless mapping for closed surfaces. However, these methods require a spherical texture, which is rarely available, and do not necessarily lead to lower distortions than those achieved by planar mapping.

The problem of providing planar texture map for closed surfaces was raised by Piponi and Borshukov [2000]. The Pelting method they propose generates textures for simply connected closed meshes. The method expects the user to define the seams necessary to generate the planar mapping. Pelting also contributed a blending mechanism used to feather the texture map near its boundary, blending the boundary of the texture at its seam. This blending removes the texture discontinuity artifact that ordinarily appears along a seam, but generates blurring artifacts in low frequency and structured textures.

Lapped textures avoid distortion by creating numerous small texture maps [Praun et al. 2000]. Like pelting, lapped texture also used texture blending to avoid seams. In addition, its small texture swatches had irregular boundaries to further disguise the seams. Lapped textures are not texture maps, and hence do not efficiently support texture-map applications like surface painting. The blended swatches also exhibit blurring artifacts when applied to low frequency or highly structured textures. The ability to layout a large texture instead of a bunch of small ones better supports surface painting and textures with low-frequency features.

Sander *et al.*, [2001] and Levy *et al.* [2002] subdivide the surface into multiple small patches and texture map the patches separately. Carr and Hart [2002] have even texture mapped individual polygons. These procedures have many of the advantages and drawbacks of the lapped texture method. In a recent work, Gu *et al.* [Gu et al. 2002] suggest using seams for reducing mapping distortion. The authors apply parameterization repetitively to find regions of maximal distortion and connect the regions to the boundary one at a time. The procedure does not provide minimal seams in terms of length and visibility. It can be time consuming as surface parameterization has to be performed repetitively to locate each region of high distortion.

3 Algorithm Overview

Our algorithm receives as input a 2-manifold triangulated surface of any genus (closed or open) in three dimensions. It then cuts seams into the surface along the existing edges of the mesh to satisfy the following goals. First, the seams have to convert the surface into an open genus zero 2-manifold with a single loop of boundary edges. This requirement converts the surface into a valid input to most texture mapping methods. The majority of the methods mentioned above can not handle inputs with multiple boundary loops (e.g. a cylinder). Hence the seams we generate will have to be connected and will have to join the existing multiple boundary loops on the surface into a single boundary loop.

The second goal is to provide a surface which has acceptable mapping distortion. The actual texture distortion will depend on the parameterization method used. But as explained above it will be bounded from below by an inherent distortion functional of the surface itself. The purpose of the seams is to reduce this part of the distortion to a level acceptable to the user.

While satisfying those two goals the visual impact of the seams is minimized by minimizing their length and hiding them in less visible parts of the model.

The algorithm has two main stages. First it pre-processes the model by computing a visibility measure for the mesh edges (Section 4) and assigning a distortion measure to the mesh nodes (Section 5). In the second step nodes which contribute a significant part of the distortion are selected. Those nodes are then connected by minimally visible seams (Section 6).

4 Visibility

The seams generated by Seamster need to cut through high distortion areas of the model to reduce distortion. This constraint still leaves a lot of freedom in the choice of the seam paths. For example, fingers and other extremities have high curvature and are usually cut from the base to the tip, but often a direct cut from any point on the base to the tip will work.

We use the visibility of the seam as an additional criterion for determining its path. The incorporation of visibility allows the seam to follow hidden paths that are the most obstructed from view.

Seamster classifies the visibility of edges using a faster, hardware accelerated version of Interior/Exterior Classification [Nooruddin and Turk 2000]. Nooruddin and Turk classified regions as interior or exterior based on visibility. They performed this classification by ray tracing a polygonal object orthographically, from a variety of uniformly sampled viewpoints on a hemisphere, into layered depth images. They then classified segments as interior or exterior based on their average visibility.

Our visibility classification implementation uses a simple rasterization z-buffer. We repeatedly render the polygons of the object viewed orthographically from viewpoints uniformly sampled across the entire sphere. Each polygon is plotted with a color corresponding to the index of its face in the object mesh database. The framebuffer is then accessed, and a visibility counter associated with each polygonal face is incremented once if its color is present in the framebuffer. These face visibility counters are divided by the number of renderings to yield an average visibility of each face.

The visibility $V(f)$ of a face f will range from zero, for faces of interior or heavily occluded structures, to 50% (or more) for prominent faces. (The visibility can exceed 50% due to the stochastic nature of the finite number of samples.) Adjacent face visibilities are averaged to approximate the visibility of their shared edge, which we denote $V(e)$.

A more accurate determination of edge visibilities $V(e)$ can be obtained by rendering a hidden line drawing. As before the object is rendered orthographically from viewpoints uniformly sampled from the sphere. This time a hidden-line rendering is performed where each edge is drawn using a color corresponding to its index in the object mesh database. As before, the framebuffer is accessed and a visibility counter is incremented for each edge whose index appears as a pixel color in the stored framebuffer.

The choice of the sphere as the sampling domain assumes that the user is equally likely to view the model from any direction. This is true in a modeling environment, but is too general for many animation scenarios. For example, when modeling animated creatures which walk on the ground the view from below the animal is usually not very likely. By limiting the sampling to parts of the sphere, we can modify the visibility measurements to reflect the more restricted set of viewpoints. This will prioritize seam placement in regions which are less visible from more common viewpoints.

5 DISTORTION

To estimate the distortion generated when the surface is mapped to the plane we can measure the Gaussian curvature of the surface [Ahlfors and Sario 1960]. Zero Gaussian curvature corresponds to a developable surface which can be mapped to the plane with zero distortion. The curvature at any surface point is an indicator of how far a surface is locally from being developable. The distortion is in fact a monotone increasing function of the curvature's magnitude. Note that while the actual distortion depends on the parameterization method used, the distortion estimate should always provide a lower bound for the actual distortion. For the purpose of distortion measurement only the curvature at the mesh vertices is of interest. The edges do not contribute to the distortion, since a surface is locally developable around each edge. A pair of faces sharing an edge can always be projected to the plane with no distortion (Figure 2(a)). This is not true for vertices (Figure 2(b)).

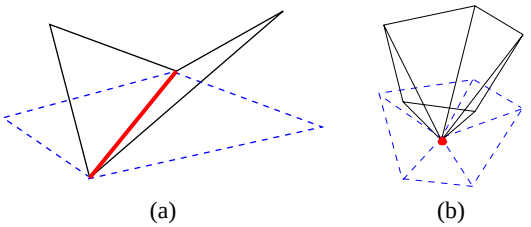


Figure 2: Local mapping to a plane: (a) the immediate neighborhood of an edge can be flattened with no distortion, (b) but the neighborhood of a vertex generally cannot.

When considering distortion at vertices, we can consider the distortion within the immediate neighborhood of the vertex or across a larger surrounding region. In the later case we better capture the curvature for fine meshes. Consider for example a hand model with a fine mesh. If only local curvature is considered, the vertices on the finger tips will not stand out with respect to curvature compared to other vertices. However, if the curvature across a surrounding region is measured, the value at the tips will be significantly higher than on the palm.

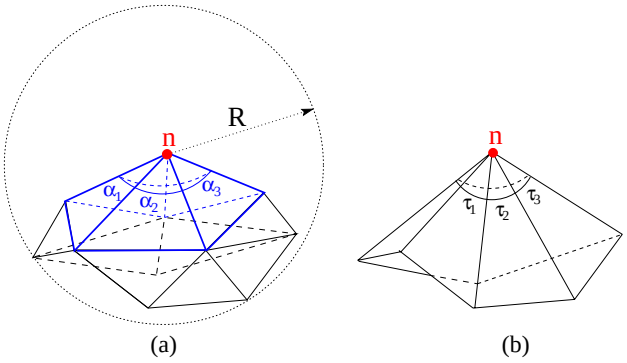


Figure 3: Measuring distortion. (a) Submesh M_r. The blue lines indicate the triangles used to measure $D'_0(n)$; (b) Measuring $D'_r(n)$ using the angles τ_j formed by n and the boundary edges of M_r.

We define the region $M_r(n)$ around vertex n as follows. Let $r \geq 0$ be the region magnitude. For $r = 0$, the region contains the triangles adjacent to n. For $r > 0$ we define the region radius R as the product of r and the length of the longest edge emanating from n

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

Given the radius, the region M_r contains all the triangles of the mesh within distance R from n (measuring distance between vertices). We then consider the triangles t_j formed by the boundary edges of M_r and n (Figure 3(b)). Region distortion [Sheffer 2002] is defined as

$$D'_r(n) = \frac{2\pi - \sum_j \tau_j}{2\pi}, \quad (1)$$

where τ_j are the angles at n of triangles t_j. This measure varies significantly as r changes. To capture the distortion both for coarse and fine mesh levels we use

$$D_r(n) = \max_{0 \leq i \leq r} (D'_i(n)) \quad (2)$$

instead. The radius i corresponding to the maximal distortion value is stored for later processing. The distortion, as defined, can be negative (at saddle points). The measure above applies only to interior mesh vertices. At the mesh boundary we need to distinguish between two cases. If the sum of angles around a boundary vertex n is less than 2π, then the sub-mesh around it is locally developable and $D_r(n) = 0$. If the sum is above 2π, $D_r(n) < 0$ and has to be computed as above.

For $D_r(n)$ to be a correct indicator of distortion around a vertex, we need to handle two additional concerns. One concern is that the construction of M_r must avoid the generation of regions with multiple loops (Figure 4(a)). Otherwise, using D_r on a developable cylindrical region will wrongly indicate high distortion.

An even more important issue not handled by the previous development of this distortion measure [Sheffer 2002] is region overlap. As demonstrated in Figure 5, for any pair of nearby vertices

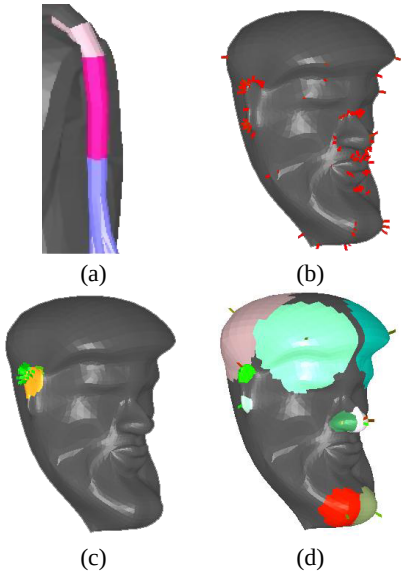


Figure 4: Region distortion. The spikes identify the vertices which represent high distortion regions. The colored regions around them identify the corresponding M_i's (the region for each vertex is colored in different shade). (a) Cylindrical regions generated on cow's (Figure 1) tail using basic D_r definition. (b)-(d) Vertices responsible for 25% of the head model distortion. (b) Regions and vertices generated with $r = 0$, i.e. based only on local curvature at the vertices. (c) Regions and vertices generated with $r = 5$ with no overlap avoidance. (d) Same, with overlap avoidance.

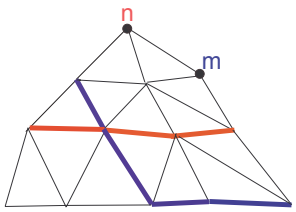


Figure 5: Overlapping vertex regions $M_r(n)$ and $M_r(m)$.

n and m and an r larger than the distance between the two, the regions $M_r(n)$ and $M_r(m)$ will overlap. This often causes an artificial increase in the curvature measure at one of the vertices, since its region partly incorporates the region around the second vertex. In Figure 5 $D_r(m)$ will be affected by the high curvature around n. If the distortion measure is used as is, this will generate artificial clustering of vertices with high D_r (Figure 4(c)). To avoid this the following test is performed for any pair of vertices n and m, where n is inside $M_i(m)$ (i is the index at which $D_r(m)$ is maximized).

- If $D_r(n)$ is larger than $D_r(m)$ we assume that n is the main contributor to the distortion. In this case we need to compute the distortion around m limiting it to a region not containing n. For efficiency reasons, we in this case limit ourselves to the immediate region around m and set $D_r(m) = D_0(m)$.
- If the two values are about equal, we need to consider the distortion in the non-overlapping regions around them. Yet again for efficiency reasons we consider only the $r = 0$ regions around them. If $D_0(m) > D_0(n)$ we keep $D_r(m)$ as is, and set $D_r(n) = D_0(n)$, or vice versa.

After the region "filtering" we obtain a much more correct indication of vertices which are the centers of regions of high distortion (Figure 4(d)). Note that using the region distortion provides a much better indicator than using the local distortion only (Figure 4(b)).

6 Seam Cutting Algorithm

The purpose of adding seams is to reduce the overall mesh distortion $D(M) = \sum_n D_r(n)$ by cutting through interior high distortion vertices and thus setting the distortion at these vertices to zero. In the discussion below we ignore the case where vertex distortion is negative (angle sum above 2π) for simplicity. A seam will reduce the distortion at a negative curvature point only if it passes through the saddle vertex. If the seam terminates at a negative curvature point, the flattened surface can overlap itself. A seam can reduce the distortion of a positive curvature region even if it terminates at a high curvature vertex without fear of overlap.

Another purpose of the seams is to reduce the genus of multiply connected surfaces, to enable planar parameterization (Section 6.3). As explained above, the vast majority of the parameterization methods require the parameterized surface to have a single boundary loop. Therefore all the seams have to be connected. The problem of finding optimal seams to reduce the distortion below a desired level D can be defined as follows on the graph of the mesh vertices and edges (N,E).

- Set the cost of each vertex n to $D_r(n)$.
- Set the weight of each interior mesh edge $W(e) = V(e)\|e\|$. (Recall $V(e)$ is the edge visibility). For boundary edges set $W(e) = \varepsilon > 0$ since boundary edges do not "cost" in terms of cutting, but for algorithm simplicity their weight has to be above zero.
- Find a connected sub-tree $S = (N_t, E_t)$ s.t.
  - $\sum_{n \notin N_t} D_r(n) \leq D$,
  - $\sum_{e \in E_t} W(e)$ is minimal.

This is a variant of the *prize-collecting Steiner tree* problem [Hochbaum 1997], which is known to be NP-Complete. A factor of two approximation algorithm for the problem is given by Goemans and Williamson [1995]. Figure 6(a) shows the result of implementing a variant of this method on a cow model to reduce the distortion by one-third. This example demonstrates the main drawback of using this approach in our case. The surface distortion is not spread equally or even randomly throughout the model, but more commonly is concentrated in several regions of the model. Thus local methods [Goemans and Williamson 1995] produce inadequate results. To overcome this problem we separate the solution into

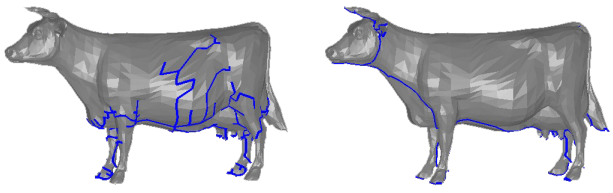


Figure 6: Seam generation methods. (a) Prize-collecting Steiner tree. (b) Seamster - minimal Steiner tree on a subset of vertices.

two tasks. First we select the vertices with high distortion measure, which we call *terminals*, and then construct a tree which connects them. The chosen method does not guarantee the factor of two approximation, however, in practice it generates shorter seams (in terms of weight) for the same amount of distortion reduction. The method finds the seam tree as follows.

1. Select a set of terminal vertices T to serve as the tree leaves.
2. Compute an approximate minimal Steiner tree of T.

The two stages are described in detail below. Note that the *minimal Steiner tree* problem, formally defined below, is also NP-Complete and hence only an approximate tree is computed.

6.1 Terminal Vertex Selection

Prior to selecting the vertices, the desired distortion level D needs to be selected. Given the total distortion $D(M)$ across the mesh, the acceptable distortion D is selected by the user, as it highly depends on specific application needs. We provided three choices of selecting D:

1. Set D to some constant, unrelated to the specific mesh properties.
2. Set D to a percentage of the overall distortion.
3. Set an acceptable distortion at a vertex d and set $D = d|N|$, where $|N|$ is the number of vertices in the mesh.

From the experiments, the choice of D as a percentage of the overall distortion is the most intuitive and easy to manipulate. This choice was used for all the examples in the paper. Given D the selection of terminal vertices T is performed as follows.

Algorithm 1 (VerticesSelect(N)).

1. **begin**
2. sort N in nonascending order of D_r ($D_r(n_i) \geq D_r(n_{i+1})$)
3. $i = 0$, $s = 0$, $T \leftarrow \emptyset$
4. **while** $s < D(M) - D$
5. $T \leftarrow T \cup n_i$
6. $i = i + 1$, $s = s + D_r(n_i)$
7. **endwhile**
8. **end**

This gives us the vertices with highest expected distortion around them. During the selection we can also consider the vertex visibility, by ordering vertices according to $D_r/V(n)$ where $V(n)$ is the vertex visibility. This will prioritize the selection of less visible vertices. Adding the visibility factor has a minor effect on models with prominent features, but hides the seams better for models where distortion is spread relatively equally. The selection guarantees that the distortion after the seam addition will be less than D. The final surface, after the seam cutting, should have a single boundary loop. Hence, if the surface *a priori* contains one or more boundary loops, we add all the boundary vertices to T. By assigning small $\varepsilon > 0$ weights to the boundary edges, as explained above, we guarantee that the boundary loops will automatically be incorporated into the seam by the Steiner tree.

6.2 Approximate Minimal Steiner Tree

A Steiner tree for a set of vertices (terminals) in a graph is a connected sub-graph containing all the terminal vertices [Skiena 1997]. A minimal Steiner tree is a Steiner tree with minimal sum of edge weights. The problem of finding the minimal Steiner tree is NP-Complete [Skiena 1997]. The following is a standard algorithm for approximation of the minimal Steiner tree. It is proven to be within factor of $2/\sqrt{3}$ of the optimum [Skiena 1997].

1. For each $n, m \in T$ compute the shortest path $P(n,m)$ between n and m.
2. Define a new graph where T are the vertices and there are edges between each pair of vertices. The new edge weights are set to weight of the shortest path between the two terminals.
3. Compute the minimal spanning tree on the new graph.

Our previous implementation [Sheffer 2002] of the approximate Steiner tree for generating seams used Dijkstra's shortest path algorithm [Cormen et al. 2000] for Step 1 and Kruskal's minima spanning tree algorithm [Cormen et al. 2000] for Step 3. That implementation explicitly computed the paths from all the terminal vertices to all the vertices in the mesh. However, only a fraction of those are used for the computation of the tree in Step 3.

This work provides a new algorithm for computing the MST which shortens the computational time to a fraction of its original cost (Table 1). The main idea of the algorithm is to use front propagation from a set of seeds. The initial seeds are the selected terminal vertices. the fronts around each seed are grown by adding the vertex with shortest path to one of the seeds n to its front. When two fronts meet, at a vertex n we perform the following. First the path from seed s_1 to n and the path from seed s_2 to n are stored as part of the MST. It is proven that the two sub-paths define the shortest path from s_1 to s_2. From now on the two fronts are considered a single joint front with the seed set $\{s_1, s_2\}$. This ensures that only paths from the two seeds to other terminal vertices will be considered. The process continues until all the fronts are merged. At this point we obtain a path connecting all the terminal vertices as required.

The resulting tree has the same weight sum as the one computed using the Dijkstra and Kruskal algorithms. As demonstrated in Table 1 the speedup increases with increase in model size and number of terminal vertices (model complexity). The speedup allows handling of significantly larger models (e.g. Fig. 11). However at those sizes the second part of a mapping procedure, namely the actual mapping computation becomes excessively time consuming. (This is true for any of the existing mapping methods).

6.3 Genus Reduction

A surfaces can be mapped to a planar domain only when it is homeomorphic to a disk. Hence surfaces of genus greater than zero have to be cut to enable the mapping. The genus of a connected two manifold can be computed using the Euler-Poincare formula

$$N + F - E = 2 - 2G - B,$$

where N, F, and E are the numbers of mesh vertices, faces and edges respectively, G is the genus and B is the number of boundary loops. The problem of finding a minimal set of cut edges is known to be NP-Hard [Erickson and Har-Peled 2002].

The algorithm proposed in [Erickson and Har-Peled 2002] repeatedly finds the shortest loop connecting a mesh vertex to itself using front propagation. The loops are then tested to see if they

reduce the surface genus or simply cut the surface into two pieces. After the loops of all the vertices are found, the shortest one which reduces the genus by one is selected. The process is repeated until the genus is reduced to zero. The procedure is proven to find cuts whose length is within a constant factor of the minimal cuts. However, due to the repetitive testing it can be quite time consuming.

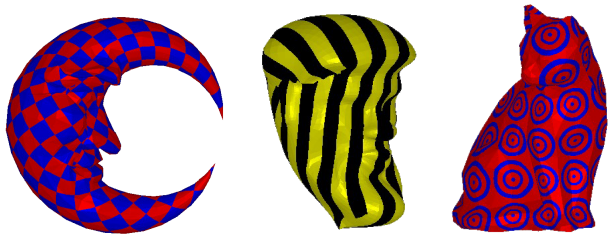
Figure 7: Seams reducing the genus of a figure eight model and the resulting texture. The two interior loops are found by the genus reduction method. The remaining seams were created by the spanning tree.

Seamster integrates visibility information into the genus reduction process. This information is helpful since the interior walls of a tunnel, across which the shortest cut should pass, are less visible than its exterior entryways. The modified procedure is performed as follows. We treat genus-reducing seams as manifold boundaries, and so genus reduction occurs after visibility determination but before seams are cut between high-distortion vertices. First the mesh nodes are sorted in increasing order of visibility. Starting from least visible vertex we find the shortest weighted loop which connects the vertex to itself. The weight is the visibility of the edge computed similarly to the tree cutting procedure above. As in [Erickson and Har-Peled 2002] we check if the loop reduces the genus. If it does, we introduce the necessary cut. Here we differ from [Erickson and Har-Peled 2002] where all loops are computed and tested. In theory this can produce longer cuts, but in practice due to the use of low visibility in the vertex and path selection, we find paths which are close to optimal. Similarly to [Erickson and Har-Peled 2002] we continue the testing of loops until the genus is reduced to zero. By avoiding the testing of all the loops at each stage we reduce the runtime by a factor of $|N|$. In practice it is reduced further since low visibility vertices are likely to occur on loops which reduce genus, hence less vertices need to be tested to find such loops. An example of genus reduction is shown in Figure 7.

7 Results

To generate surface texture Seamster needs to be combined with an actual mapping (parameterization) procedure. In the examples in this paper (e.g. Figure 8) it was combined with a texture mapping method developed by Sheffer and de Sturler [Sheffer and de Sturler 2002]. The method does not require the two-dimensional domain boundary to be predefined, and does not force it to be convex. Seamster can also be combined with other similar parameterization techniques [Hormann and Greiner 2000; Levy et al. 2002; Desbrun et al. 2002].

The examples in this section demonstrate the seams and the texture generated based on them. Except Figure 7 all the models are



(a) (b) (c)

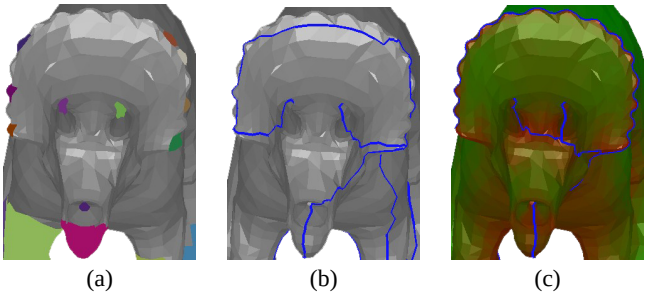
Figure 8: Texture examples: (a) moon (2K faces); (b) head (7K faces); (c) cat (670 faces).

of genus zero. We have demonstrated Seamster on both open and closed meshes (e.g. the head model shown in Figure 8(b) has a boundary loop at its neck). The visibility function on the models shown in Figures 1, 9 and 10 is visualized by a color spectrum from red to green, where red is low visibility and green is high. For the four-legged animals (cow, triceratops, and rabbit) the view sphere was constrained to views from above. The flat "skin" models show the two dimensional mapping of the cut surface. The statistics on some of the models and results are summarized in Table 1.

Model	Size (faces)	$D(M)$ before	$D(M)$ after	Seam % (length)	Map dist.	Seamster ([Sheffer 2002])
Cat	671	4.76	3.93	1.82	1.043	0.19s (0.43s)
Rabbit	902	15.27	10.96	4.3	1.085	0.10s (0.94s)
Cow	5804	38.1	28.7	2.46	1.04	4.5s (18.5s)
Tricer.	5660	35.3	22.6	2.56	1.095	3.7s (26.4s)
Moon	2324	15.59	10.65	2.18	1.043	1.9s (2.5s)
Head	7232	9.59	7.44	1.97	1.017	3.9s (40.7s)
Dinosaur	28136	132.8	87.33	2.16	1.065	184s (3800s)
Hand	2000	13.7	8.66	2.79	1.055	2.26s (2.76s)

Table 1: Model Statistics. The seam length percent measures the length of the seam edges with respect to the total length of mesh edges. The mapping distortion is measured using stretch measure [Sander et al. 2001] (the value range is $[1,\infty]$). The execution times compare Seamster to our previous distortion-minimizing seam cutting implementation.

Figure 9 demonstrates the effect of the visibility factor on the seams in the triceratops model, zooming in on the head. Taking visibility into consideration, the seams become longer but move to less visible locations between the horns, under the chin and along the crown.



(a) (b) (c)

Figure 9: Seams with and without visibility factor. (a) Regions of extremum points on the head. (b) Shortest seams connecting them. (c) Visibility function (red shows less visible regions) and the seams generated using visibility.

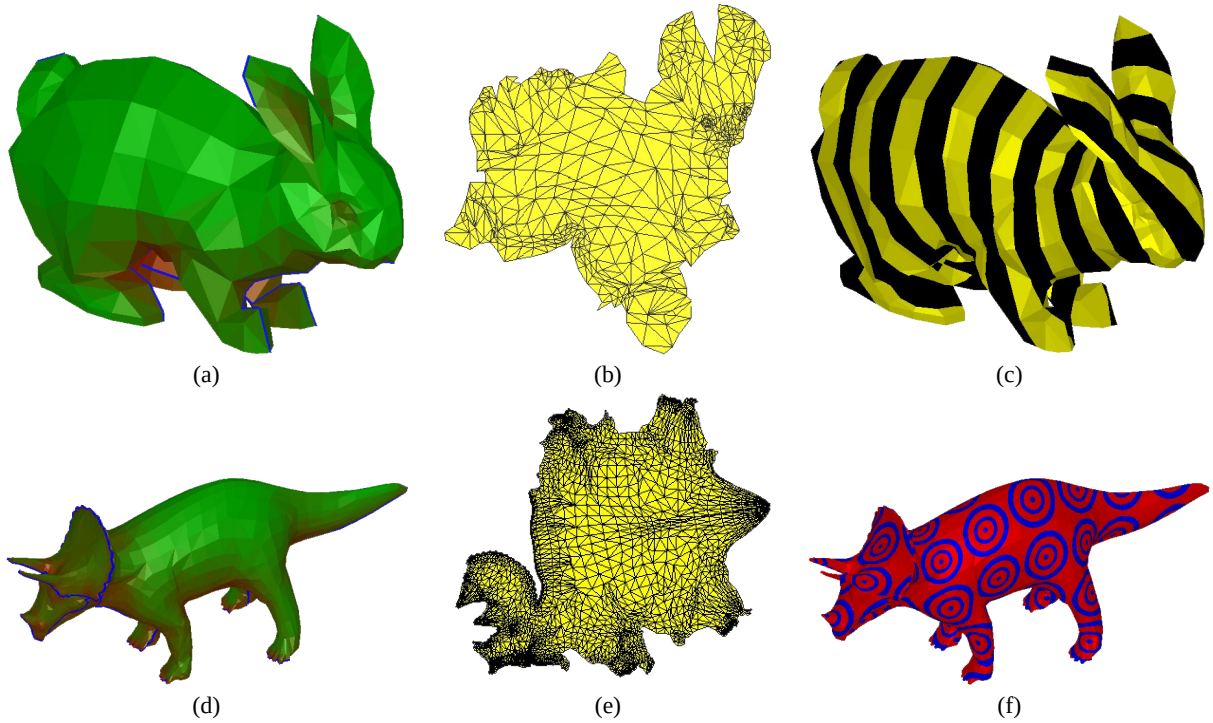


Figure 10: Seamster generated textures. (a,d) Visibility and seams. (b,e) Flat parameterization "skin." (c,f) Texture.

Figure 11 utilizes the efficiency of Seamster's fast approximate MST algorithm to generate the seam of a large dinosaur model. Seamster ran over twenty times faster on this model than did our previous MST implementation (which took over an hour).

Figure 11: Texturing a large model (28K faces).

Figure 12 compares the results of the optimal seams method with seams and texture generated using geometry images [Gu et al. 2002]. The main difference between the resulting seams is in the connection of the index finger and thumb and in the seam on the back of the hand. The seams generated by seamster are shorter and the final texture has less distortion (stretch metric of 1.4 for geometry images v. 1.055 for Seamster), thanks to the free boundary mapping.

8 Conclusions

We have developed a new method for non-distorted texture mapping by generating surface seams prior to the parameterization procedure. The new algorithm converts any closed or open surface into a patch homeomorphic to a disk, which can be parameterized in the plane. It reduces the parameterization distortion to a level acceptable by the user. All this is done while minimizing the visual impact of the seams, by minimizing their length and hiding them in less visible areas of the model.

One important problem to be addressed in future research is fitting textures across seams. The problem is seen in Figure 8(b), where the texture discontinuity at the seam along the brow is very visible. The seam is necessary for distortion reduction and hence cannot be avoided. There is also no better (more hidden) path for it. However, it might be possible given a specific texture to find a local mapping stretching the texture so that the discontinuity is less visible. In the case of Figure 8(b) it is possible to move the texture on the two sides of the brow, making the yellow and black stripes meet, and thus hiding the seam. An automatic procedure to do this is an important future extension of the method.

A second problem we plan to address is a more rigorous way to measure surface distortion, which will more accurately capture the model shape. This will enable easier comparison between models with respect to their inherent distortion.

Acknowledgments. This work was sponsored in part by the NSF under grant NSG-9732379. Thanks to Hugues Hoppe for the hand model and statistics, and to Jerome Maillot for the head model. Thanks also to Sariel Har-Peled and Jeff Erickson for their help with genus reduction. The picture at the end of the references was created by Oded Fuhrmann.

(a) (b) (c) (d) (e) (f)

Figure 12: Seams and texture comparison. (a)-(b) Seams and texture generated using geometry images. (c)-(f) Seamster results. (c)-(d) Visibility functional and seams, (e)-(f) texture generated by ABF [Sheffer and de Sturler 2002].

REFERENCES

AHLFORS, L. V., AND SARIO, L. 1960. *Riemann Surfaces*. Princeton University Press, Princeton, New Jersey.

BENNIS, C., VEZIEN, J., AND IGLESIAS, G. 1991. Piecewise surface flattening for non-distorted texture mapping. *Proc. SIGGRAPH 91* (July), 237–246.

CARR, N. A., AND HART, J. C. 2002. Meshed atlases for real-time procedural solid texturing. *ACM Transactions on Graphics 21*, 2 (Apr.), 106–131.

CORMEN, T. H., LIESERON, C. E., AND RIVEST, R. L. 2000. *Introduction to Algorithms*. MIT Press, Cambridge.

DESBRUN, M., MEYER, M., AND ALLIEZ, P. 2002. Intrinsic parameterizations of surface meshes. *Proc. Eurographic 2002*, to appear.

ERICKSON, J., AND HAR-PELED, S. 2002. Cutting a surface into a disk. *Proc. ACM Symposium on Computational Geometry*, 244–253.

GOEMANS, M. X., AND WILLIAMSON, D. P. 1995. A general approximation technique for constrained forest problems. *SIAM Journal on Computing 24*, 296–317.

GU, X., GORTLER, S., AND HOPPE, H. 2002. Geometry images. *Proc. SIGGRAPH 2002* (July), 355–361.

HAKER, S., ANGENENT, S., TANNENBAUM, A., KIKINIS, R., SAPIRO, G., AND HALLE, M. 2000. Conformal surface parameterization for texture mapping. *IEEE Transactions on Visualization and Computer Graphics 6 (2)*, 181–189.

HOCHBAUM, D. S. 1997. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Company.

HORMANN, K., AND GREINER, G. 2000. Mips: an efficient global parameterization method. In *Curve and Surface Design: St. Malo 1999*, Vanderbilt University Press, 153–162.

HURDAL, M., BOWERS, P., STEPHENSON, K., SUMNERS, D., REHMS, I. K., SCHAPER, K., AND ROTTENBERG, D. 1999. Quasi-conformally flat mapping the human cerebellum. In *Proc. of MICCAI'99*, Springer-Verlag, vol. 1679 of *Lecture Notes in Computer Science*, 279–286.

LEVY, B., AND MALLET, J. 1998. Non-distorted texture mapping for sheared triangulated meshes. *Proc. SIGGRAPH 98* (July), 343–352.

LEVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least squares conformal maps for automatic texture atlas generation. *Proc. SIGGRAPH 2002* (July), 362–371.

MA, S., AND LIN, H. 1988. Optimal texture mapping. *Proc. Eurographics '88* (Sep.), 421–428.

MAILLOT, J., YAHIA, H., AND VERROUST, A. 1993. Interactive texture mapping. *Proc. SIGGRAPH 93* (Aug.), 27–34.

NOORUDDIN, F., AND TURK, G. 2000. Interior/exterior classification of polygonal models. *Proc. Visualization 2000* (Oct.), 415–422.

PIPONI, G., AND BORSHUKOV, D. 2000. Seamless texture mapping of subdivision surfaces by model pelting and texture blending. *Computer Graphics (Annual Conference Series, 2000. SIGGRAPH '00)*, 471 – 478.

PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2000. Lapped textures. *Proc. SIGGRAPH 2000* (July), 465–470.

SAMEK, M. 1986. Texture mapping and distortion in digital graphics. *The Visual Computer 2*, 5, 313–320.

SANDER, P., SNYDER, J., GORTLER, S., AND HOPPE, H. 2001. Texture mapping progressive meshes. *Proc. SIGGRAPH 01* (Aug.), 409–416.

SHEFFER, A., AND DE STURLER, E. 2002. Smoothing an overlay grid to minimize linear distortion in texture mapping. *ACM Transactions on Graphics 21*, 4 (Oct.).

SHEFFER, A. 2002. Spanning tree seams for reducing parameterization distortion of triangulated surfaces. *Proc. Shape Modelling International* (May), 61–66.

SKIENA, S. S. 1997. *The Algorithm Design Manual*. Springer Verlag.