Iso-charts: Stretch-driven Mesh Parameterization using Spectral Analysis

KUN ZHOU
Microsoft Research Asia
JOHN SNYDER
Microsoft Research
BAINING GUO and HEUNG-YEUNG SHUM
Microsoft Research Asia

We describe a fully automatic method, called iso-charts, to create texture atlases on arbitrary meshes. It is the first to consider stretch not only when parameterizing charts, but also when forming charts. The output atlas bounds stretch by a user-specified constant, allowing the user to balance the number of charts against their stretch. Our approach combines two seemingly incompatible techniques: stretch-minimizing parameterization, based on the surface integral of the trace of the local metric tensor, and the "IsoMap" parameterization, based on an eigen-analysis of the matrix of squared geodesic distances between pairs of mesh vertices (spectral analysis). We show that only a few iterations of nonlinear stretch optimization need be applied to the IsoMap parameterization to obtain low-stretch atlases. The close relationship we discover between these two parameterizations also allows us to apply spectral clustering based on IsoMap to partition the mesh into charts having low stretch. We also novelly apply the graph cut algorithm in optimizing chart boundaries to further minimize stretch, follow sharp features, and avoid meandering. Overall, our algorithm creates texture atlases quickly, with fewer charts and lower stretch than previous methods, providing improvement in applications like geometric remeshing and texture synthesis. We also describe an extension, signal-specialized atlas creation, to efficiently sample surface signals, and show for the first time that considering signal stretch in chart formation produces better texture maps.

Categories and Subject Descriptors: I.3.3 [**Computer Graphics**]: mesh, parameterization

Additional Key Words and Phrases: eigenanalysis, chartification, graph cutting, multi-chart geometry image, signal-specialized parameterization, texture synthesis

1. INTRODUCTION

Parameterization forms the basis of many geometry processing algorithms, such as texture mapping, morphing, editing, remeshing and compression. For parameterizing arbitrary meshes, a popular technique is to build texture atlases [Maillot et al. 1993; Pedersen 1995; Lévy et al. 2002; Sander et al. 2001]. The target surface is

Authors' addresses: K. Zhou, B. Guo and H.-Y. Shum, Microsoft Research Asia, 3F, Beijing Sigma Center, No. 49, Zhichun Road, Haidian District, Beijing 100080, P.R.C.; J. Snyder, Microsoft Research, Redmond, WA, 98052, USA.

first partitioned into a set of charts, called chartification, which are parameterized and packed into the texture domain to form an atlas.

Because a 3D surface is not isometric to a 2D plane, parameterization causes distortion. Distortion can be measured in many ways, including how well angles or areas are preserved, or how much parametric distances are stretched or shrunk onto the surface. We focus on distance distortion, specifically [Sander et al. 2001]'s definition of geometric stretch, which measures the average and worst-case stretching of local distances over the surface. Minimizing geometric stretch uses texture samples more efficiently than other measures [Sander et al. 2001], and is asymptotically related to geometric accuracy under piecewise-constant reconstruction [Sander et al. 2002].

Stretch reduction is intimately related to the partition of meshes into charts. Generally speaking, the smaller the charts, the less the distortion of any kind, including stretch. In the limit, distortion can be eliminated by making each mesh triangle its own chart [Carr and Hart 2002]. However, excessively fine partitioning is disadvantageous. It constrains mesh simplification, adds extra inter-chart gutter space, causes mipmap artifacts, and removes continuity across charts [Sander et al. 2001]. A good mesh atlas creation tool must therefore allow a careful choice of its charts, balancing chart distortion against the cost of more charts.

We achieve this balance by driving mesh partitioning from a user-specified stretch value. [Lévy et al. 2002] alluded to this possibility. However, stretch-driven parameterization is challenging, requiring nonlinear optimization which greatly slows the computation. If computing a stretch-minimizing embedding for a single chart is costly, then computing it over all possible chart partitionings becomes completely impractical. This difficulty has led previous approaches to disregard stretch when forming charts in favor of unrelated heuristics that cut across sharp features or cluster based on chart compactness or planarity. These simple heuristics produce charts that are either smaller or more stretched than necessary. Our approach, iso-charts, is the first algorithm for generating a texture atlas with large charts and bounded stretch.

To do this, we apply a form of nonlinear dimensionality reduction called IsoMap [Tenenbaum et al. 2000] which minimizes geodesic distance distortion between pairs of vertices on the mesh. The key to this application is our new observation that geodesic distance distortion is closely related to stretch, though they are defined quite differently. IsoMap thus provides two benefits for atlas generation. It provides an effective way, called *spectral analysis*, to decompose the model into large, geometrically meaningful parts like animal appendages that can be parameterized as charts with little stretch. Without any extra computation, it also supplies an initial parameterization for each chart. In fact, we show that it provides an excellent starting point for stretch minimization, so that a few iterations of nonlinear stetch optimization quickly remove problem "fold-overs" that result from applying IsoMap by itself. In essence, IsoMap provides a much better heuristic for creating and parameterizing low-stretch charts than previous methods.

Our main contribution is a new stretch-driven chartification method which clusters based on a spectral analysis of the matrix of geodesic distances and allows the user to bound stretch while keeping the number of charts small. We also show that such spectral analysis simultaneously obtains a low-stretch parameterization of charts more quickly than previous methods. We introduce the notion of graph cut to optimize chart boundaries, and modify the capacity metric to consider geodesic distance distortion and therefore stretch, given the relationship discovered between the two metrics. We propose "special spectral clustering" to create better charts in cases when geometric parts have periodic (tubular) structure. Finally, we generalize our approach to signal-specialized atlas creation. Our atlases are the first whose chart partitioning, as well as parameterization, is adapted to a particular signal such as a normal or color map.

2. RELATED WORK

There are many ways to build a texture atlas. A straightforward method is to partition the model by hand [Krishnamurthy and Levoy 1996]. Early work [Maillot et al. 1993] clusters triangles according to their normals. Other methods produce charts with convex boundaries [Eck et al. 1995; Guskov et al. 2000; Khodakovsky et al. 2003; Lee et al. 1998; Sander et al. 2001], a restriction that can significantly increase stretch.

The least squares conformal map (LSCM) [Lévy et al. 2002] parameterizes charts with arbitrarily shaped borders. It finds curves through high curvature zones and then grows charts to meet at these curves. The Intrinsic Parameterization [Desbrun et al. 2002] is another free-boundary, conformal atlas approach. Though it preserves angles, conformal parameterization yields area and distance distortion that is undesirable for many applications in which texture map storage should be conserved. [Sander et al. 2003] presents a chartification method based on stretch which forms compact charts that also cut along high-curvature features. None of these methods bounds stretch, and all neglect parameterization distortion during chartification.

[Sorkine et al. 2002] directly considers stretch during chartification. But it greedily adds triangles to a growing parameterization without further adjustment, and so is unable to form large charts.

Alternatively, an arbitrary surface can be cut into a single chart instead of an atlas. Geometry Images [Gu et al. 2003] parameterize an entire surface over a single 2D square using cuts through vertices having extreme stretch. *Seamster* [Sheffer and Hart 2002] is another cutting algorithm that reduces distortion by mapping to a free boundary. Although they can be regarded as stretch-driven cutting/parameterization algorithms, these methods do not allow the user to control cut length or stretch.

By observing that many objects consist of relatively simple regions, each of which has a natural parameterization, [Zhang et al. 2004] proposed a feature-based patch creation method. They make use of distance-based Morse functions to reduce genus and identify features. Each feature region will be unfolded to one or two patches based on a covariance matrix of the feature's surface points. To reduce the stretch during patch unfolding, the *Green-Lagrange tensor* is used in an optimization process. Although this method can decompose the models into large charts, it does not allow user control of the parameterization stretch. You can see from the texture layouts (refer to Figure 11 in that paper) that the texture colors are greatly distorted in some regions. Another advantage of our method is that it runs much faster. For the Stanford bunny (70k faces), our method takes about 1 minute on a 3.0 GHz PC, while the running time for the 10K bunny model using their method is over 9 minutes on a 2.4 GHz PC. Furthermore, our algorithm can also be generalized to produce signal-specialized atlases which represents a given surface signal using textures as compact as possible.

Many algorithms can parameterize charts over planar regions [Bennis et al. 1991; Campagna and Seidel 1998; Desbrun et al. 2002; Eck et al. 1995; Floater 1997; 2003; Hormann and Greiner 1999; Lévy and Mallet 1998; Lévy et al. 2002; Maillot et al. 1993; Sander et al. 2001]. We refer the interested reader to the recent survey by [Floater and Hormann 2004].

Like our method, [Zigelman et al. 2002] also applies IsoMap to mesh parameterization, but allows only simple (disk-topology) meshes and often produces triangle flips. Our method handles an arbitrary mesh, automatically divides it into multiple charts, guarantees no triangle flips, and extends to signal-specialized parameterization. We also show that the IsoMap parameterization is connected to a stretch minimizing one.

Recently, [Peyré and Cohen 2004] presented a surface flattening method based on another nonlinear data dimensionality reduction method, called *Locally Linear Embedding* (LLE) [Roweis and Saul 2000]. LLE aims at finding a mapping which preserves the relationship between neighboring data points. Therefore the LLE-based flattening approach is more local than the IsoMap-based approach [Zigelman et al. 2002], which tends to preserve the pairwise geodesic distances between surface vertices. However, both methods cannot guarantee validity of the flattening.

A form of spectral analysis has been used for geometry compression and smoothing [Karni and Gotsman 2000; Taubin 1995]. These methods are based on the mesh's Laplacian which depends only on its connectivity, rather than geodesic distance. Recently, [Gotsman et al. 2003] applies spectral graph theory to solve the spherical parameterization problem.

We apply stretch-minimization [Sander et al. 2001] to optimize parameterizations created by spectral analysis. We also use geometric stretch [Sander et al. 2001] or signal stretch [Sander et al. 2002] to determine when to terminate chart subdivision.

[Katz and Tal 2003] decompose meshes using geodesic distance and optimize boundaries using graph cut. We apply these ideas to atlas creation, but also introduce spectral analysis to aid decomposition and produce geodesic distance preserving parameterizations. We extend their notion of graph cut "flow capacity" to respect parameterization distortion, and their notion of distance to build signal-specialized atlases.

3. ALGORITHM OVERVIEW

Our approach is a top-down, stretch-driven method. Given a surface and a user-specified stretch value, it performs the following steps:

(1) Compute the surface spectral analysis, providing an initial parameterization (Section 4.1).
(2) Perform a few iterations of stretch optimization [Sander et al. 2001].
(3) If the stretch of this derived parameterization is less than the threshold, stop.

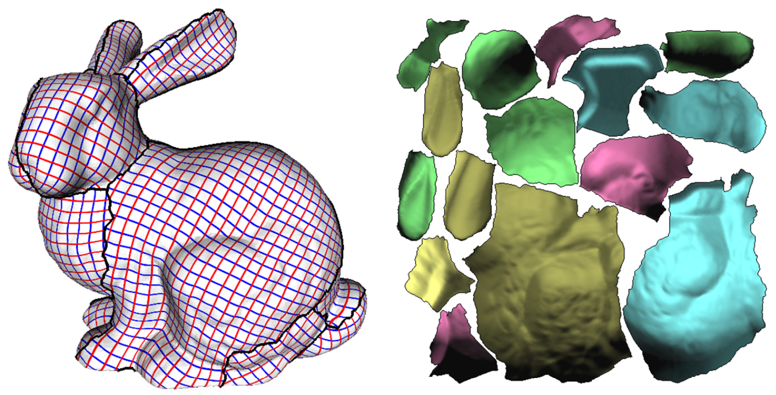


Fig. 1. Iso-chart atlas for the Stanford bunny. The model is partitioned into 15 large charts, which can be flattened with lower stretch than previous methods ($L^2 = 1.01$, $L^\infty = 2.26$).

(4) Perform spectral clustering to partition the surface into charts (Section 5.1).

(5) Optimize chart boundaries using the graph cut technique (Section 5.2).

(6) Recursively split charts until the stretch criterion is met.

The result is a set of charts whose parameterizations have bounded stretch. Chart topology need not be explicitly checked; the stretch-driven process ensures that all charts are eventually subdivided into topological disks since otherwise parameterization stretch is infinite. We do check that the parameterization domain does not overlap itself and subdivide in that rare case. As a post-processing step, we merge small charts together if the parameterization stretch of the merged chart is less than the user specified stretch value.

Distortion is bounded using two norms on geometric or signal stretch, proposed in [Sander et al. 2001]. The L^2 norm integrates $(\gamma_{max}^2 + \gamma_{min}^2)/2$ over the surface, followed by an overall square root. The L^∞ norm maximizes $\max\{\gamma_{max}, 1/\gamma_{min}\}$ over the entire surface. Here, γ_{max} and γ_{min} are scalar functions over the surface representing the largest and smallest singular values of the Jacobian of the affine mapping from texture space to model/signal space at any point. The inclusion of shrink, $1/\gamma_{min}$, in the L^∞ norm is a modification which penalizes undersampling.

Figure 1 shows the chartification and parameterization results for the Stanford bunny. Notice how meaningful parts of the model like its head, ears, and body are decomposed into large charts. The whole computation takes about 1 minute.

4. SPECTRAL ANALYSIS

4.1 Applying IsoMap

Our algorithm builds upon the dimensionality reduction method called IsoMap (isometric feature mapping) [Tenenbaum et al. 2000]. Given a set of points on a high-dimensional manifold, IsoMap computes geodesic distances along the manifold as sequences of hops between neighboring points. It then applies MDS (multidimensional scaling) to find a set of points embedded in a low-dimensional space whose Euclidean distances are similar to the corresponding geodesic distances on the original manifold, as shown in Figure 2. Since geodesic rather than Euclidean

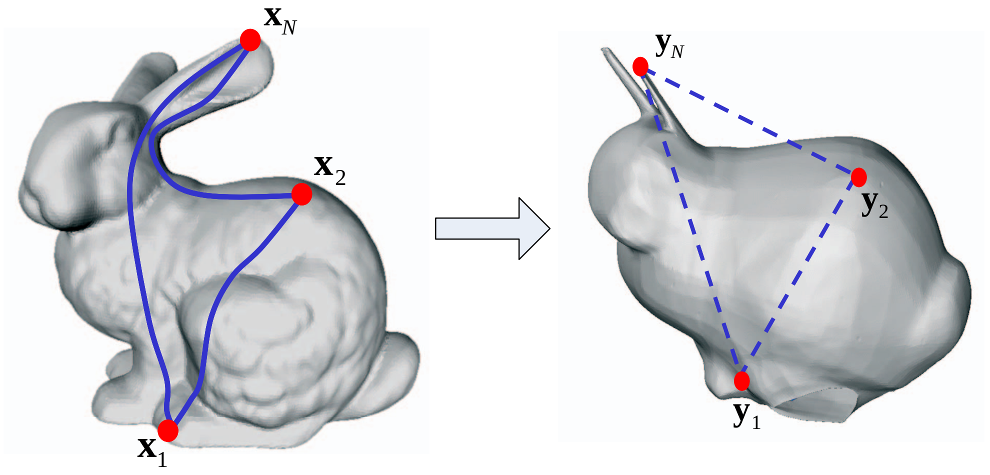


Fig. 2. Spectral analysis for the Stanford bunny. The left image is the original bunny; the right image is a visualization of the result of spectral analysis showing just the first three embedding coordinates. These embedding coordinates, $\vec{\mathbf{y}}_i$, are chosen so that Euclidean distances between pairs match geodesic distances between points on the original surface, $\vec{\mathbf{x}}_i$.

distance represents the true geometry of the manifold, IsoMap can discover the nonlinear manifold structure underlying complex data like 3D surface geometry.

We refer to this application of IsoMap as *surface spectral analysis* and outline its computation. Given a surface with N vertices, $\vec{\mathbf{x}}_i$,

—Compute the symmetric matrix $\mathbf{D}_N$ of squared geodesic distances between surface vertices.

—Apply double centering and normalization to $\mathbf{D}_N$ to yield $\mathbf{B}_N = -\frac{1}{2}\mathbf{J}_N\,\mathbf{D}_N\,\mathbf{J}_N$, where $\mathbf{J}_N$ is a $N \times N$ centering matrix defined by $\mathbf{J}_N = \mathbf{I} - \frac{1}{N}\,\mathbf{1}\,\mathbf{1}^{\mathbf{T}}$, $\mathbf{I}$ is the identity matrix, and $\mathbf{1}$ is a vector of ones of length N. This constrains the center of gravity of the computed point set to lie at the origin.

—Compute the eigenvalues λ_i and their corresponding eigenvectors $\vec{\mathbf{v}}_i$ of $\mathbf{B}_N$, $(i = 1, 2, ..., N)$.

—For each vertex i of the original surface, its embedding in the new space is an N-dimensional vector $\vec{\mathbf{y}}_i$ whose j-th component is given by $\vec{\mathbf{y}}_i^j = \sqrt{\lambda_j}\,\vec{\mathbf{v}}_j^i$ $(j = 1, 2, ..., N)$.

The eigenvalues λ_i and their corresponding eigenvectors $\vec{\mathbf{v}}_i$ of $\mathbf{B}_N$, $(i = 1, 2, ..., N)$ form the spectral decomposition of the surface shape. Eigenvectors corresponding to large eigenvalues represent global, low-frequency features on the surface while eigenvectors corresponding to small eigenvalues represent high-frequency details. It is natural to consider the high-energy, low-frequency components as a basis of chartification and parameterization.

TODO: possible diagram visualizing embedding from different eigenvalues. Could show bunny embedding with first three components, then next three, then next three, etc.

In general, N eigenvalues (and thus an N-dimensional embedding) are needed to fully represent a surface with N vertices. In practice, a small number of them typically dominate the energy. For the bunny model shown in Figure 1, the top 5 eigenvalues constitute over 85% of the squared energy; in other words, $(\sum_{i=1}^{5} \lambda_i)/(\sum_{i=1}^{N} \lambda_i) > 85\%$. Therefore we calculate only the $n \ll N$ largest eigenvalues and corresponding eigenvectors, leading to a n-dimensional embedding for all vertices.

The distortion of this n-dimensional embedding can be calculated as the sum of the geodesic distance distortion over all vertices. For each vertex i, its *geodesic distance distortion* (GDD) under the embedding is defined as:

$$GDD(i) = \sqrt{\frac{1}{N-1} \sum_{j=1}^{N} \left(\|\vec{\mathbf{y}}_i - \vec{\mathbf{y}}_j\| - d_{geo}(i,j) \right)^2} \tag{1}$$

where $\vec{\mathbf{y}}_i$ is the n-dimensional embedding coordinate of vertex i, and $d_{geo}(i,j)$ is the geodesic distance between vertex i and j. This definition can be extended from a vertex to a triangle by averaging the distortions of its three vertices, and is used in Section 5.2.

When $n = 2$, surface spectral analysis yields a surface parameterization minimizing the sum of squared GDD over all vertices. This is the key idea of the mapping algorithm in [Zigelman et al. 2002]. Our observation is that the same technique, surface spectral analysis, can be simultaneously applied to two critical problems: decomposition necessary for chartification, and parameterization.

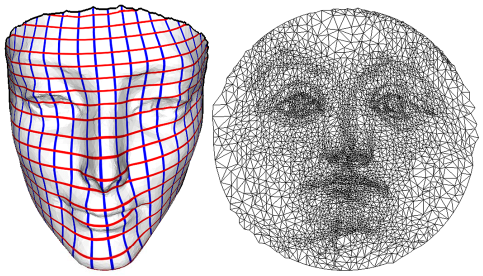
To calculate the geodesic distances between surface points of polygonal models, we use the fast matching method [Kimmel and Sethian 1998], which runs at $O(N^2 \lg N)$ and obtains more precise results than the Dijkstra graph search method, since it allows paths that cut across mesh triangles.

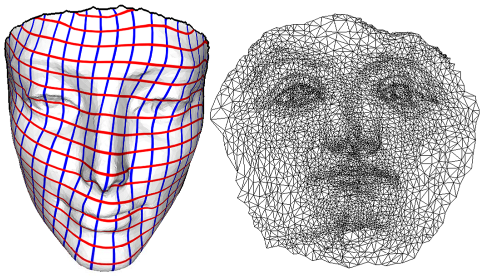
4.2 Analyzing Isomap Stretch

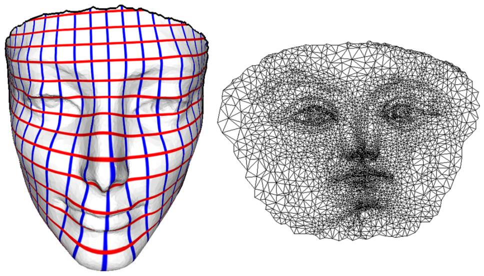
GDD-minimizing [Zigelman et al. 2002] and stretch-minimizing [Sander et al. 2001] parameterizations both focus on distance distortion. Still, GDD differs from stretch in several ways. It is *global* rather than *local*, since it considers distance between vertices that are arbitrarily far on the surface, rather than local stretch induced by the Jacobian at a point. It is *difference-based* rather than *ratio-based* since it penalizes differences between the original and parametric distances rather than how much unit-length tangent vectors are stretched. And it is *discrete* rather than *continuous* since it only considers distance distortions between vertex pairs rather than stretch in every triangle and in every direction.

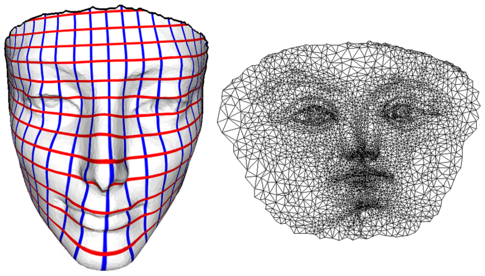
The discrete nature of spectral analysis, which measures distance distortion only between vertex pairs, gives rise to the main problem in [Zigelman et al. 2002]: triangle flips. Our algorithm provides a simple solution. Since triangle flips are defined to cause infinite stretch, and our algorithm always splits charts whose stretch is above the user's threshold, any finite threshold guarantees the final atlas will contain no flips.

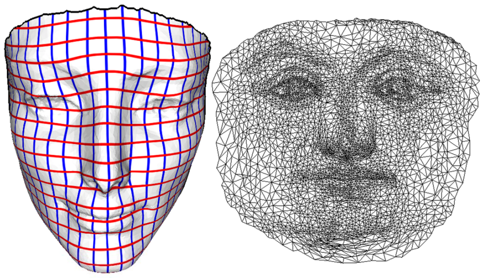
Spectral analysis requires solution of a low-dimensional eigenvalue problem rather than general nonlinear optimization. We accelerate the computation even further using the "landmark" extension (see Section 5). Despite differences between stretch and GDD, we find that spectral analysis reduces stretch very effectively.


(a) Uniform + fine-level stretch
[Sander et al. 2001]
$L^2 = 1.04$, $L^\infty = 1.68$
PSNR=76.8dB, running time 222s


(b) Uniform + coarse-to-fine stretch
[Sander et al. 2002]
$L^2 = 1.03$, $L^\infty = 2.16$
PSNR=76.8dB, running time 39s


(c) LSCM
[Lévy et al. 2002]
$L^2 = 1.11$, $L^\infty = 2.27$
PSNR=74.9dB, running time 3s


(d) LSCM + fine-level stretch
[Lévy et al. 2002] + [Sander et al. 2001]
$L^2 = 1.09$, $L^\infty = 2.27$
PSNR=75.5dB, running time 26s


(e) IsoMap
[Zigelman et al. 2002]
$L^2 = 1.04$, $L^\infty = 2.78$
PSNR=77.0dB, running time 2s

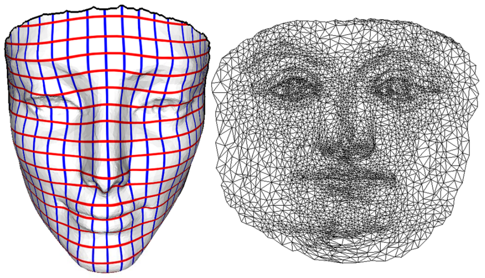

(f) IsoMap + fine-level stretch
[Zigelman et al. 2002]+[Sander et al. 2001]
$L^2 = 1.03$, $L^\infty = 2.32$
PSNR=77.0dB, running time 6s

Fig. 3. Comparison between different parameterization algorithms. PSNRs are measured using a geometry image that keeps the number of defined (within charts) samples constant at about 13,500. (d) applies 100 iterations of stretch minimization to the result of (c); similarly, (f) applies 20 stretch-minimizing iterations to (e).

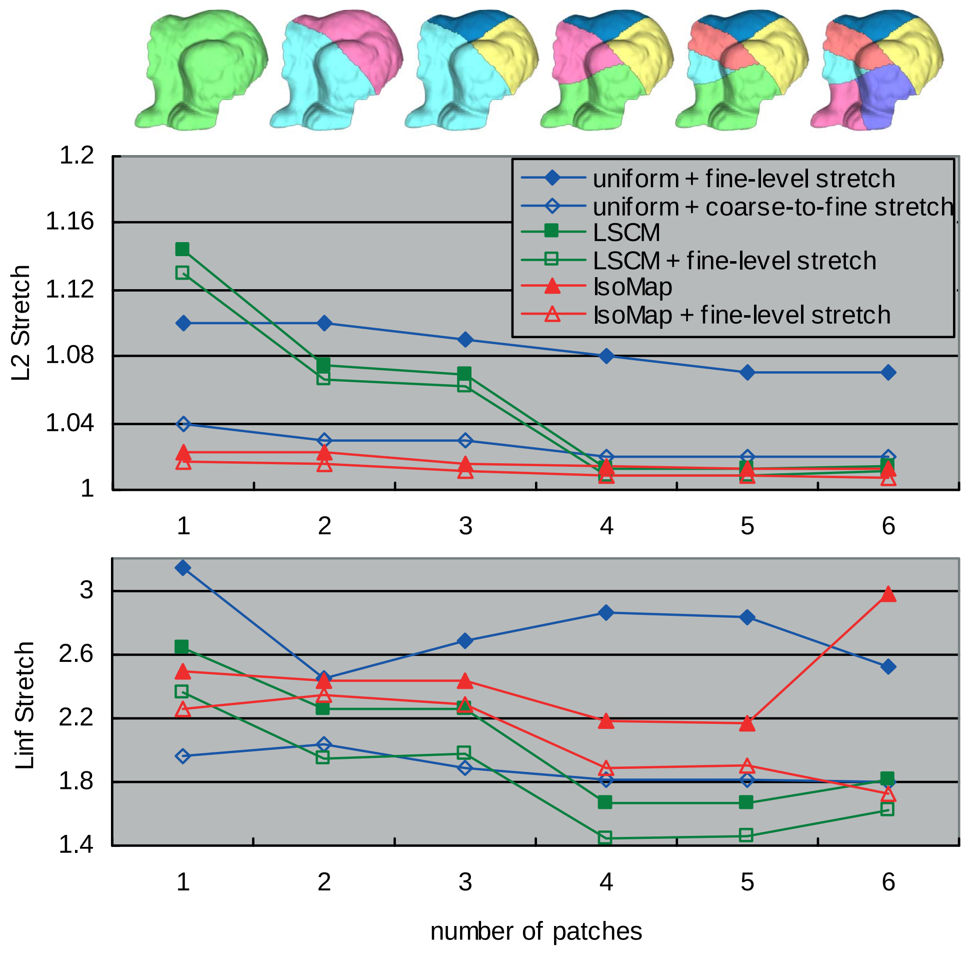


Fig. 4. Stretch vs. number of patches for different parameterizations.

Figure 3 shows parameterization results for a single chart model of a face. While LSCM (3c-d) is fast, it is conformal and so limits stretch poorly. IsoMap (3e) quickly provides a parameterization having little L^2 stretch – in fact, it's stretch is as good as the slow optimization method of [Sander et al. 2001] (3a). Furthermore, it provides a starting point which a few L^2-stretch-minimizing iterations improves even further (3f), yielding the best L^2 stretch result of all methods, including ones explicitly designed to minimize stretch. The PSNR numbers are for a geometry image constructed with the resulting parameterizations, and confirms the relationship between L^2 stretch and geometric accuracy discussed in [Sander et al. 2002]. Since it depends on only a single, worst-case point in the domain, L^∞ stretch is difficult to control for any method, but especially for a discrete method like IsoMap. Even there, our result is comparable after applying a few stretch-minimizing iterations.

Figure 4 compares results on a multi-chart model of a bunny, focusing on the trade-off between stretch and the number of charts. We partitioned the bunny into a series of chart sets, containing from 1 to 6 charts, and shown at the top of the

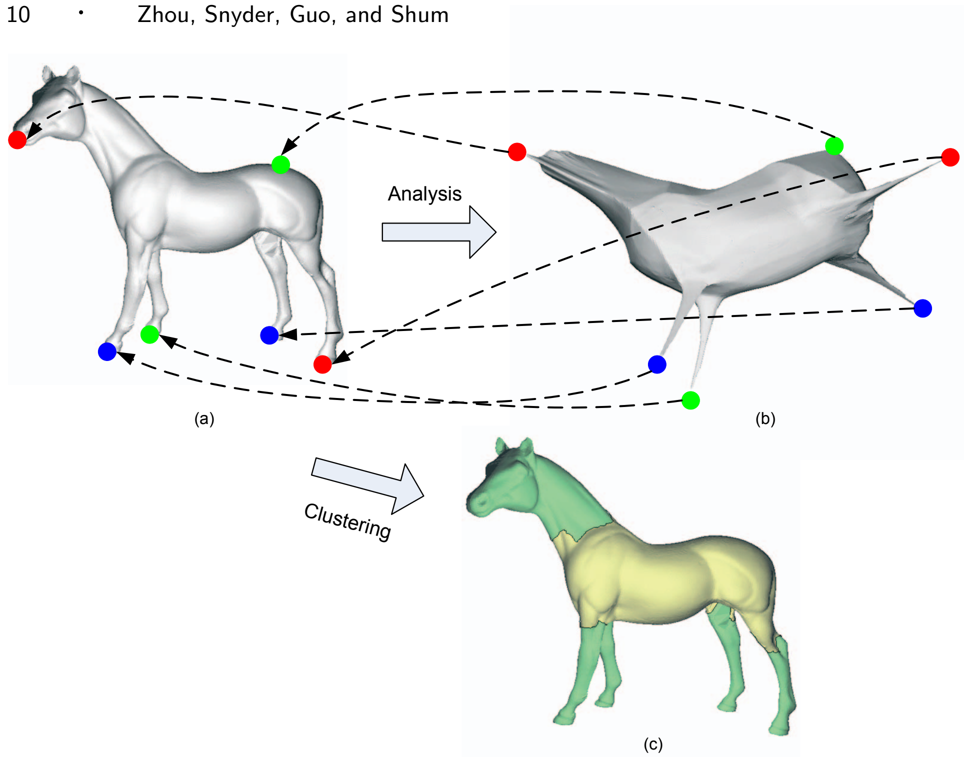


Fig. 5. Spectral analysis/clustering method for the horse model.

figure. As in Figure 3, the best L^2 stretch result is obtained by our method. We can also see that as the number of charts increases, it becomes easier to parameterize these smaller charts with any method. Only "uniform+fine-level stretch" ([Sander et al. 2001]) lags behind, because its initial 2D parameterization domain is a circle while the other methods adopt more natural domain shapes. As in Figure 3, L^∞ stretch is more haphazard, and a few L^2 stretch-minimizing iterations improve our result significantly.

5. SPECTRAL CLUSTERING AND BOUNDARY OPTIMIZATION

5.1 Spectral Clustering

If the parameterization induced by spectral analysis fails to satisfy the user's stretch threshold, it is partitioned into several smaller charts. Recall that global features of a model such as the head, ears, legs, and tails of animals correspond to the larger eigenvalues, so we use them to partition. We compute a few representative vertices using the spectral analysis results and then grow charts simultaneously around these representatives, a method we call *surface spectral clustering* and illustrated in Figure 5. The algorithm is as follows:

(1) Rank the eigenvalues λ_i and corresponding eigenvectors $\vec{\mathbf{v}}_i$ from surface spectral analysis such that $(\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N)$.

(2) Get the top n eigenvalues and eigenvectors such that λ_n/λ_{n+1} is maximized.

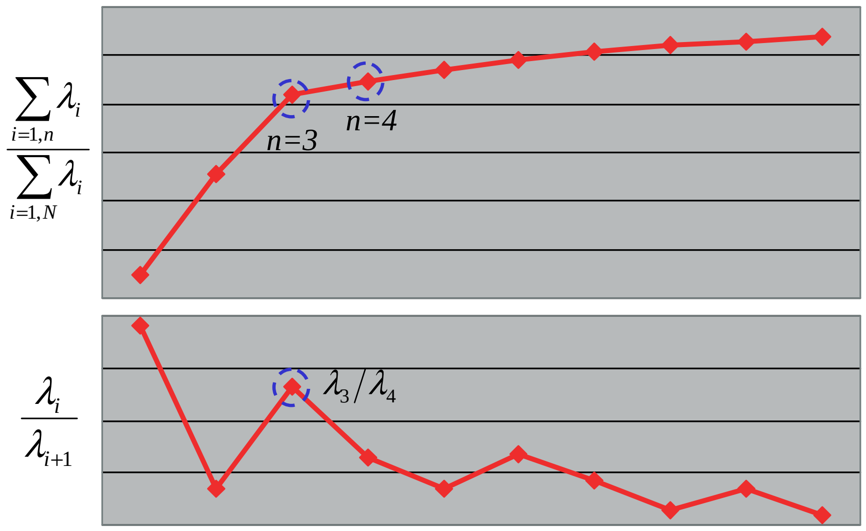


Fig. 6. Eigenvalues for the horse model.

(3) For each vertex i of the mesh, compute its n-dimensional embedding coordinates: $\vec{\mathbf{y}}_i^j = \sqrt{\lambda_j}\,\vec{\mathbf{v}}_j^i$ $(j = 1, 2, ..., n)$.

(4) For each of the n embedding coordinates, find the two vertices with maximum and minimum coordinate values and set them as $2n$ representatives, shown as pairs of colored dots in Figure 5.

(5) Remove representatives which are too close together, yielding $m \leq 2n$ representatives.

(6) Partition the mesh into m parts by growing charts simultaneously around the representatives using the geodesic distance calculated in surface spectral analysis. Each triangle is assigned to the chart whose representative is closest to the triangle.

Step 2 computes n by finding the "knee" in the curve relating the sum of eigenvalues to the number of eigenvalues, as shown in the top graph of Figure 6. To find this knee, we look at the ratio of successive eigenvalues, findng the pair where the eigenvalues are dropping fastest; i.e., finding i maximizing λ_i/λ_{i+1}. In the case where the top three eigenvalues constitute 90% or more of the total eigenvalue sum, we perform binary subdivision. Otherwise, we find the $n \geq 3$ which maximizes the ratio of successive eigenvalues and proceed with this n using the above 6 steps. The bottom part of Figure 6 shows the ratio graph and the optimal value of $n = 3$ selected for this horse example.

The value of n is a measure of shape complexity: $n < 3$ implies a fairly flat shape; large n implies a complicated shape with significant detail. Eliminating the remaining $N - n$ eigenvalues ignores high frequency detail and avoids partitioning into too many charts. Our implementation also restricts $n \leq 10$ (see Section 6), which in turn restricts the maximum number of sub-charts.

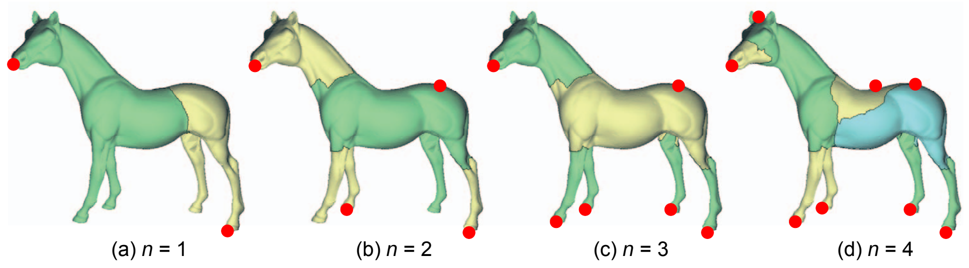


Fig. 7. Spectral clustering results for the horse model as a function of n.

Since representatives computed from different dimensions in Step 4 may be close and so redundant, Step 5 removes them. We use a distance threshold of 10 times the average edge length of the input mesh. In Step 6, the geodesic distance from a triangle to a representative vertex is computed as the average of the geodesic distances of the triangle's three vertices to the representative.

Figure 7 shows clustering results for different values of n on the horse model. Note how identifiable features, such as the legs, are partitioned using this method. The top row in Figure 13 demonstrates spectral clustering results for other models. Charts correspond to global features in the model such as the wings of the feline, and the neck, legs and tail of the dinosaur.

5.2 Computing Optimal Partition Boundaries with Graph Cutting

After splitting charts, we optimize the boundaries between them. Chart boundaries should satisfy two objectives: 1) they should cut through areas of high curvature without being too jaggy, and 2) they should minimize the embedding distortions of the charts they border.

The first objective has been addressed in previous chartification work [Sander et al. 2001; Lévy et al. 2002; Sander et al. 2003], which minimize various measures of chart compactness while choosing chart cuts of shortest length or along edges having high dihedral angle. Recently [Katz and Tal 2003] has used graph cut to decompose meshes, an idea we apply to the mesh parameterization problem. The second objective relates to our desire for a stretch-minimizing partition, and has never been addressed as far as we know.

Our solution is to formulate the optimal boundary problem as a graph cutting problem. For simplicity, we discuss the binary case which splits the surface into two. When subdividing into more than two charts, we consider each pair of neighboring charts in turn.

Figure 8a gives an example. Suppose we seek an optimal boundary between two charts **A** and **B**. The initial partition is generated by using surface spectral clustering. We then generate a *medial region*, **C**, by expanding an area to either side of the initial split boundary. The medial region's size is proportional to the total area of the unsplit patch; we use 30% for all examples. Now an undirected flow network graph (Figure 8b) can be constructed from **C** using an extension of the method in [Katz and Tal 2003]. We modify their definition of "capacity" between

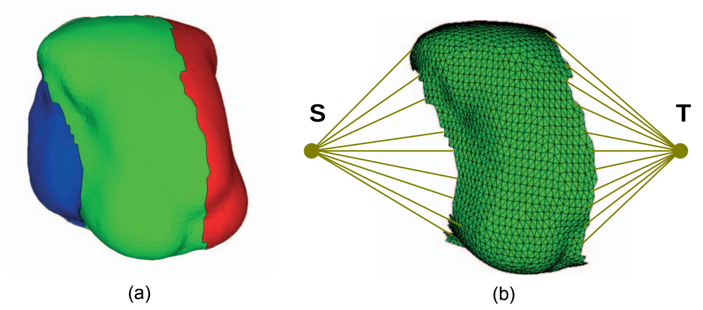
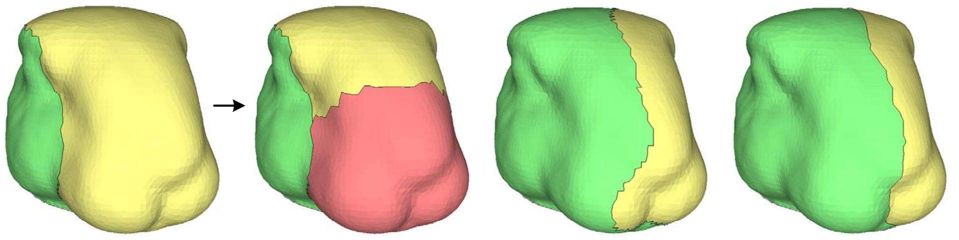


Fig. 8. Finding the optimal partition boundary is formulated as a graph cut problem. (a) the shape is decomposed into three parts, lateral areas **A** (red), **B** (blue) and medial area **C** (green). (b) constructing a graph for the medial area.



(a) Angular distance only (α=1)
L^2 = 1.02, L^∞ = 2.05

(b) GDD only (α=0)
L^2 = 1.01, L^∞ = 1.87

(c) Combined (α=0.5)
L^2 = 1.01, L^∞ = 1.94

Fig. 9. Comparing different graph-cut capacities.

the two adjacent triangles f_i and f_j as

$$c(f_i, f_j) = \alpha\, c_{ang}(f_i, f_j) + (1 - \alpha)\, c_{distort}(f_i, f_j) \tag{2}$$

The first term in equation (2) corresponds to the first objective of a nonjaggy cut through edges of high dihedral angle. We adopt the same formula as [Katz and Tal 2003]:

$$c_{ang}(f_i, f_j) = \left(1 + \frac{d_{ang}(f_i, f_j)}{avg(d_{ang})}\right)^{-1} \tag{3}$$

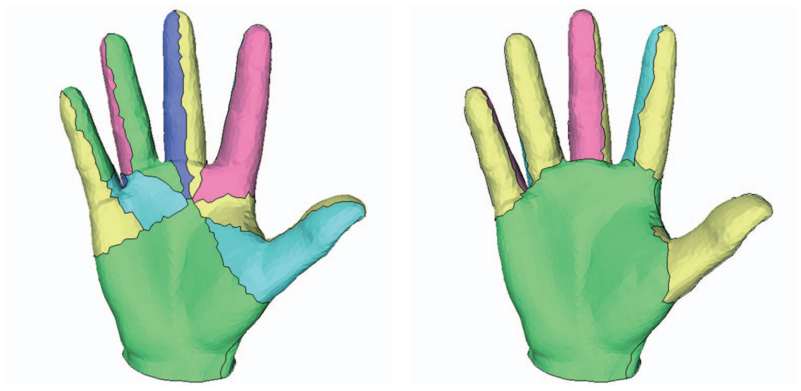
where $d_{ang}(f_i, f_j)$ is defined as $(1 - \cos\alpha_{ij})$, α_{ij} is the angle between normals of the triangles f_i and f_j, and $avg(d_{ang})$ is the average angular distance between adjacent triangles.

The second term in equation (2) measures embedding distortion, defined as

$$c_{distort}(f_i, f_j) = \frac{d_{distort}(f_i, f_j)}{avg(d_{distort})} \tag{4}$$

$$d_{distort}(f_i, f_j) = |GDD_A(f_i) - GDD_B(f_i)| + |GDD_A(f_j) - GDD_B(f_j)| \tag{5}$$

where $GDD_A(f_i)$ and $GDD_B(f_i)$ are the GDDs of triangle f_i under the embedding induced by **A** or **B**, respectively. $avg(d_{distort})$ is the average $d_{distort}(f_i, f_j)$ over all pairs of adjacent triangles. This definition of $c_{distort}(f_i, f_j)$ prefers boundary edges whose adjacent triangles balance GDD between embeddings determined by **A** and



(a) before boundary optimization
13 charts, $L^2 = 1.04$, $L^\infty = 2.40$

(b) after boundary optimization
11 charts, $L^2 = 1.01$, $L^\infty = 1.84$

Fig. 10. Boundary optimization results for the hand model.

B. In other words, the cut should avoid placing a triangle on the wrong side where it creates unnecessary distortion.

Given the graph in Figure 8b with edge weights defined via equation (2), the minimum cut algorithm generates a boundary satisfying our two objectives: it passes through areas of high curvature (if they exist) without meandering and yields low embedding distortion of its bordering charts.

The weight parameter α trades off the two objectives. α=1 defines capacity as in [Katz and Tal 2003] and achieves good results for models with sharp features. For shapes whose dihedral angles vary smoothly in the medial area, it tends toward a cut of shortest length (see Figure 9a), often generating unbalanced cuts that poorly control stretch. For example in Figure 9a (left), splitting by $\alpha = 1$ produces too much stretch in the yellow chart, requiring it to be split again to satisfy the user's threshold (right).

On the other hand, we can set $\alpha = 0$ to minimize GDD as Figure 9b illustrates, which avoids needless chart subdivision but makes the boundary jaggier. Figure 9c sets $\alpha = 0.5$. Although the parameterization stretch is a little larger than 9b, a smoother boundary is desirable for many applications.

Figure 10 shows the result of graph-cut boundary optimization on a hand model. Note how optimized chart boundaries follow creases in the model, such as the between the thumb and hand. Stretch of the resulting charts is also significantly decreased, even though fewer are required to satisfy the stretch threshold.

5.2.1 *Landmark IsoMap for Medial Region Embedding.* To compute the above optimal partition boundary, we require two embeddings over the unsplit chart: one corresponding to side **A** and one to side **B**. These two embeddings define GDD_A and GDD_B. Neither sub-chart "core", **A** or **B**, contains the inner vertices of the medial region **C**. So we can't compute the embedding coordinates of **C**'s vertices using spectral analysis on **A** or **B** alone. Since we don't yet know which triangles of **C** will be joined with **A** and which with **B**, we desire embeddings for each sub-chart that will not be too distorted by triangles that end up inserted in the other sub-chart. A recent extension of IsoMap [Silva and Tenenbaum 2002], called landmark IsoMap, solves this problem by embedding the medial region implicitly given only embeddings for each core and the geodesic distance relationship of **C**'s to each core's vertices.

Suppose there are N_A vertices in **A**. After performing surface spectral analysis, we get n_A eigenvalues λ_i and corresponding eigenvectors $\vec{\mathbf{v}}_i$. The n_A-dimensional embeddings of all vertices in **A** form the columns of an $n_A \times N_A$ matrix $\mathbf{L}_A$:

$$\mathbf{L}_A = \left[\sqrt{\lambda_1}\,\vec{\mathbf{v}}_1,\ \sqrt{\lambda_2}\,\vec{\mathbf{v}}_2,\ \cdots,\ \sqrt{\lambda_{n_A}}\,\vec{\mathbf{v}}_{n_A}\right]^T$$

A vertex p outside **A** can be located in its n_A-dimensional embedding space by using its known geodesic distances to the vertices in **A** as constraints. This same idea identifies geographic location using a finite number of distance readings in GPS [Silva and Tenenbaum 2002]. Let Δ_p denote the column vector of squared distances between p and the vertices in **A**. The n_A-dimensional embedding coordinate $\vec{\mathbf{v}}_p$ can be computed by the formula:

$$\vec{\mathbf{v}}_p = \frac{1}{2}\,\mathbf{L}_A^{\dagger}\,(\bar{\Delta} - \Delta_p)$$

where $\bar{\Delta}$ is the column mean of $\mathbf{D}_{N_A}$, and $\mathbf{L}_A^{\dagger}$ is the pseudoinverse transpose of $\mathbf{L}_A$:

$$\mathbf{L}_A^{\dagger} = \left[\vec{\mathbf{v}}_1/\sqrt{\lambda_1},\ \vec{\mathbf{v}}_2/\sqrt{\lambda_2},\ \cdots,\ \vec{\mathbf{v}}_{n_A}/\sqrt{\lambda_{n_A}}\right]^T$$

Now we can calculate GDDs for all vertices in **C** under the embedding induced by **A**, and similarly for **B**.

5.3 Special Spectral Clustering for Tubular Shapes

Given the n dominant eigenvalues, surface spectral clustering partitions the shape into at most $2n$ charts. This works well for complex shapes but can produce too many charts for simple shapes with $n \leq 3$. As shown in Figure 11a, spectral clustering partitions the bunny ear into 5 charts and the feline wing into 6 charts. To avoid excessive partitioning, we can instead subdivide the chart into two, according to the first of the embedding coordinates. This simple approach often works, but it is not ideal for tubular/cylindrical protrusions, a common feature in typical meshes (see Figure 11b).

A better method (Figure 11d) is inspired by recent work in computer vision [Brand and Huang 2003; Elad and Kimmel 2003], which observes that dominant eigenpairs of the distance matrix can be used to detect and segment data points with cyclic distributions. The following heuristic has proven effective, which regards a shape as tubular if its eigenvalues λ_i meet the following conditions:

—$(\sum_{i=1}^{3}\lambda_i)/(\sum_{i=1}^{N}\lambda_i) > 0.9$, i.e. the top three eigenvalues represent the shape well.

—$\lambda_1/\lambda_2 > 3$, means the shape is long enough.

—$\lambda_2/\lambda_3 < 2$, means the shape is cyclic.

—$\lambda_3/\lambda_4 > 3$, i.e. the 4-th eigenvalue decreases quickly enough to be ignored.

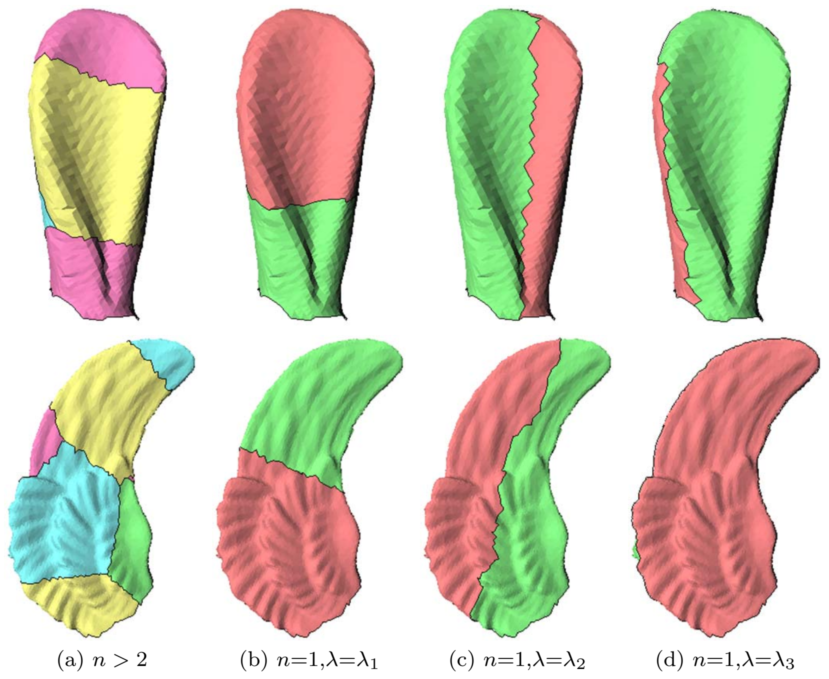

(a) $n > 2$
(b) n=1,λ=λ_1
(c) n=1,λ=λ_2
(d) n=1,λ=λ_3

Fig. 11. Partitioning tubular shapes. Column a shows general spectral clustering (Section 4.3), while columns b-d show binary clustering based on the first, second, and third eigenvalues.

As long as a shape is detected as a cylinder/tube, it is partitioned into two subcharts. As noted by [Brand and Huang 2003], the second and third dimensions can be regarded as cyclic axes. Partitioning the shape according to the third principal dimension, which corresponds to the shorter cyclic axis, produces more planar patches. Figure 11d shows the results using the third component, a more natural split than using the first or second component in Figure 6bc.

The overall chart subdivision algorithm may be summarized as follows. If the top three eigenvalues contain less than 90% of the energy, we perform "general" spectral clustering (Section 5.1). Otherwise, if the chart is tubular, we perform "special" spectral clustering described in this section. In all other cases, we perform binary spectral clustering, using the single embedding coordinate corresponding to the largest eigenvalue. In our experience, only a single nonbinary chart subdivision is performed (at the first iteration); thereafter, binary subdivision suffices.

6. IMPLEMENTATION DETAILS

A naive implementation of our stretch-driven chartification and parameterization algorithm is expensive, especially as the number of model vertices grows.

To accelerate the computation, we exploit landmark IsoMap [Silva and Tenenbaum 2002], which was used in the last section to compute the embedding coordinates for vertices in the medial region. Landmark IsoMap selects q vertices as landmark points, where $q \ll N$. Instead of computing the $N \times N$ matrix of squared geodesic distances, $\mathbf{D}_N$, an $q \times N$ matrix $\mathbf{D}_{q,N}$ is computed measuring distances from each vertex to the landmark points only. Embedding coordinates of the q landmark points are computed using surface spectral analysis while the remaining vertices can be computed using the method described in Section 5.2.1.

To get the landmark points, models are simplified by performing half edge collapse operations based on the quadric error metric [Garland and Heckbert 1997]. Progressive meshes [Hoppe 1996] free us from having to simplify each chart from scratch. We only need to perform enough vertex splits recorded in the PM to obtain enough landmark points within the chart.

For all charts, we use $q = 100$ landmark points, which makes the processing fast even on large charts. When the chart has fewer than 100 vertices, we simply include them all as landmark points. Though the landmark embedding can exhibit more stretch than the full analysis, this is likely only for large chart that have high stretch and will need to be refined anyway. Landmark embedding with q independent of chart size thus provides a fast but very reasonable heuristic.

Since the the top 10 eigenvalues constitute over 95% of the squared energy in our test models, another speed-up is to calculate only the first 10 eigenpairs in surface spectral analysis. In summary, the geodesic distance computation is reduced to $O(q\,N\log N)$ and spectral decomposition to $O(q^2)$.

To stretch-optimize the parameterization produced by spectral analysis, we perform a few iterations of the method in [Sander et al. 2002]. We first find each vertex whose L^2 stretch in its 1-ring neighborhood is greater than the user-specified limit, and rank these vertices in decreasing order of stretch. For each vertex, we then perform the following steps. If a vertex is not in the convex kernel of its neighborhood, we relocate it to the centroid of the kernel. We then take a random direction and minimize stretch along that direction. Directional minimizations are performed up to 6 times or until an iteration fails to improve stretch sufficiently (using a threshold of at least 10% improvement).

After generating an atlas, we perform a post-processing step that merges charts using a simple heuristic. We begin by ranking the charts in increasing order of number of faces belonging to the chart and computing an average normal for each chart. Only "small" charts are considered, having no more than 10% of the total number of faces. For each small chart, we consider merging each of its neighboring charts in decreasing order of dot products of average normals. Chart pairs are greedily merged in this order if the stretch after merging still satisfies the stretch thresholds.

Our packing algorithm is an extension of the "Tetris" algorithm in [Lévy et al. 2002] (refer to Figure 7 in that paper). In [Lévy et al. 2002], charts can only be introduced from the top; we can introduce them from the top, bottom, left or right directions. Our algorithm therefore keeps track of the current "horizons" for each of the four directions, given all the charts introduced so far. A horizon for a direction (up, down, left, right) is the distance in that direction to the nearest chart for each edge pixel. For example, the top horizon is the distance in the vertical direction from the top of the packing to the nearest chart inserted, for each pixel in the top row.

The user specifies the desired width and height of the atlas, W and H. Our packing algorithm then performs the following steps:

(1) Rescale each chart to make its area in the texture domain equal to its area in 3D domain.
(2) Sort the charts in decreasing order according to their area.
(3) For each chart:
    —Choose the direction pair in which to add the chart. Suppose that the current atlas height and width is H_{curr} and W_{curr} (initialized to zero). Compute $W_{expect} = H_{curr} * W/H$. If $W_{expect} > W_{curr}$, try introducing the chart from the left and right, otherwise try from the top and bottom.
    —Insert the chart into the packing. From the direction pair of the previous step (left/right or top/bottom), choose the single direction that wastes the least space. If the chart is introduced from the top (bottom), we adopt the method in [Lévy et al. 2002] to compute the horizontal translation of the chart minimizing lost space between the bottom (top) horizon of the chart and the current top (bottom) horizon of the evolving packing. Similarly, if the chart is introduced from the left (right), we minimize the lost space between the right (left) horizon of the chart and the current left (right) horizon of the packing.

Like [Sander et al. 2003], we also allow several rotations for each chart (16 in our implementation). Our method produces packings that waste little space between charts and have nearly the same aspect ratio as was requested.

7. SIGNAL-SPECIALIZED ATLAS CREATION

So far, we have used geometric stretch to drive chartification and parameterization. Our algorithm can also be generalized to produce a signal-specialized parameterization which represents a given surface signal using textures as compact as possible. To achieve this goal, [Sander et al. 2002] defines a signal-stretch metric and develops an iterated multi-grid strategy to minimize it over manually created charts. Our approach is to apply our IsoMap-based algorithm, but using signal distance rather than geometric distance.

For surface signals defined by interpolation over per-vertex samples, we compute the pairwise signal distances between vertices. Given two vertices i and j and the geodesic path between them, the signal distance between them is defined as the sum of signal differences between pairs of adjacent points along the path. Applying spectral analysis to a matrix of signal distances creates a parameterization that preserves these distances and therefore ties our algorithm to signal distortion in the same way as our unspecialized algorithm was tied to geometric distortion.

The rest of the algorithm is analogous. We use signal distance in the embedding distortion part of the graph cut capacity metric, perform signal-stretch minimizing iterations to refine the embedding [Sander et al. 2002], and use signal-stretch thresholds as a chart subdivision criterion.

For surface signals that exhibit more variation between vertices (such as one defined by a pre-existing parameterization that is to be optimized), we can exploit the idea of the *integrated metric tensor* (IMT) from [Sander et al. 2002] to encapsulate the signal's variation. The IMT is an integral over each triangle of the metric tensor

(Jacobian transpose times Jacobian) of the mapping from points on the triangle into the signal space. It is a symmetric, 2×2 matrix regardless of the signal's dimensionality and is therefore a compact measure of per-triangle signal variation, requiring only three numbers per triangle. For mapped signals, the Jacobian is not constant over each triangle, as it is for a signal defined by linear interpolation between triangle vertices, but varies spatially. Thus, computing IMTs for mapped signals requires numerical integration.

Given the geodesic path between vertices i and j, we can use the sequence of IMTs of each triangle traversed by the path to compute the signal distance along the path from i to j. Let M_k be the IMT of the k-th triangle traversed by this geodesic path, and p_k and q_k be the first and last points along the linear segment of the path within this triangle. Then the contribution to signal distance of the path segment from p_k to q_k is given by

$$(p_k - q_k)^T \, M_k \, (p_k - q_k).$$

In other words, the IMT is interpreted as a quadratic form that maps geometric difference vectors within a triangle to squared signal distances.

Typical signals such as textured colors can exhibit much more variation than the underlying geometry. Unsurprisingly, surface spectral analysis using signal distances produces a very complex embedding with many dominant eigenvalues and leads to excessive partitioning. The analogous problem led [Sander et al. 2002] to combine geometric and signal stretch; our solution defines distance with a similar combination of geodesic and signal distances:

$$d_{comb}(i,j) = \beta \, \frac{d_{geo}(i,j)}{avg(d_{geo})} + (1-\beta) \, \frac{d_{sig}(i,j)}{avg(d_{sig})} \tag{6}$$

where $d_{sig}(i,j)$ is the signal distance between i and j. We achieve good results with $\beta = 0.5$.

8. ISOCHARTS FOR TEXTURE SYNTHESIS

Surface texture synthesis algorithms [Wei and Levoy 2001; Turk 2001] build a mesh pyramid, or sequence of successively coarser meshes, to extend multi-resolution texture synthesis from 2D images to 3D surfaces. The retiling method [Turk 1991] is used for this purpose, which distributes points over the surface and then reconstructs their connectivity. To render the synthesis results, per-vertex colors are associated with the mesh. More efficient use of graphics hardware results from using a texture map, which can be created as a post-process.

We instead construct a good mesh pyramid and compact texture maps simultaneously. The idea is based on the *multi-chart geometry* image (MCGIM) [Sander et al. 2003] which uses an atlas to sample a surface's geometric coordinates and then zippers chart boundaries to ensure that a "watertight" mesh can be reconstructed.

Our method first creates a geometric stretch minimizing atlas. We then apply the sampling, zippering and packing of [Sander et al. 2003] to build a series of successively coarser MCGIMs (e.g., 128×128, 64×64, 32×32) all derived from this same atlas. Because these different resolution MCGIMs are packed and zippered separately, a natural correspondence does not exist between their samples. We

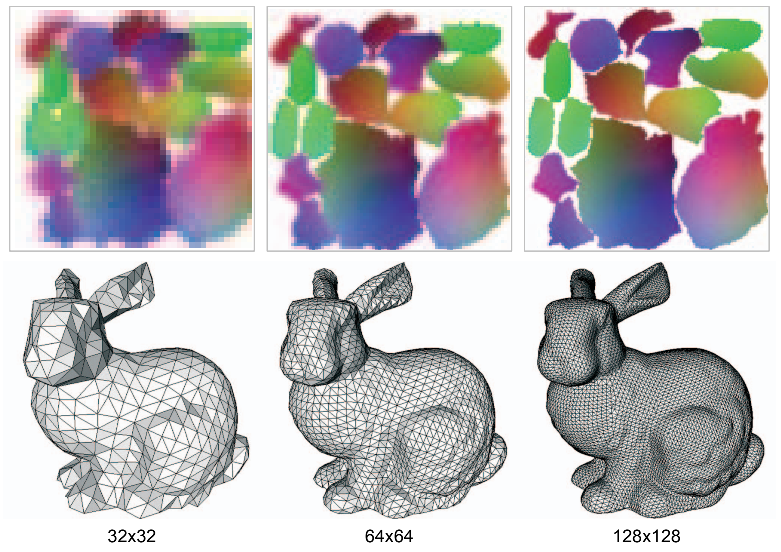


Fig. 12. Mesh LOD generation for texture synthesis.

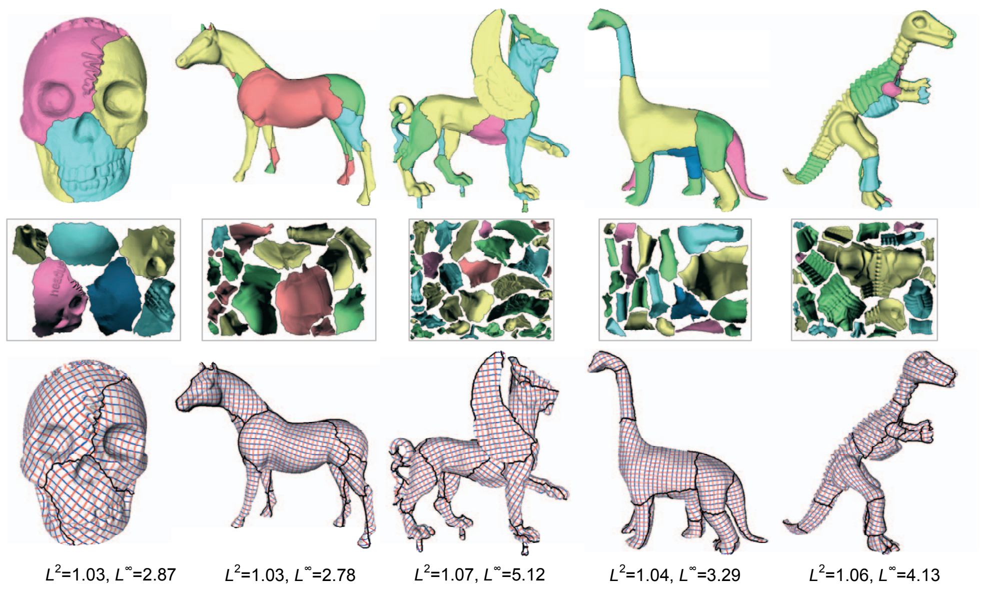
compute one explicitly using the normal-shooting method [Sander et al. 2002]. Figure 12 illustrates an example.

Our multiresolution texture synthesis algorithm extends [Wei and Levoy 2001]. The basic algorithm is similar to the sequence in which a picture is painted: long and thick strokes are placed first, and details are then added. Suppose M_{l-1} and M_l are two successive mesh LODs where M_{l-1} is the coarser one. Since the synthesis proceeds from coarse to fine LOD, assume M_{l-1} already has a synthesized texture. To synthesize the texture for M_l, first transform the colors from M_{l-1} to M_l. Then for each vertex in M_l, build its neighborhoods by flattening and resampling the mesh locally (refer to Figure 5 in [Wei and Levoy 2001]). The neighborhoods in M_{l-1} are built in a similar way by flattening and resampling the corresponding vertices and faces in M_{l-1}. Then the two layer neighborhoods are used to find the best match pixel color from the sample texture. We refer the readers to Table 3 in [Wei and Levoy 2001] for the pseudocode of the multiresolution synthesis algorithm.

Our use of MCGIMs for texture synthesis has two advantages. It quickly builds a mesh pyramid that has a one-to-one correspondence between vertices and samples in a texture map. This makes storing and rendering the resulting synthesized texture trivial. Furthermore, we can use the parameterization during texture synthesis to avoid local flattening of the mesh and the “pointer-chasing” costs of finding neighboring vertices on a 3D mesh. For samples in the interior of charts, the 2D neighborhood of the sample in the parameterization domain directly provides neighboring samples and a local flattening. The relatively few samples on chart boundaries are more complicated so we resort to the 3D mesh.

9. RESULTS

Figure 1 and Figure 13 illustrate results on several models. For all experiments, the stretch threshold was set at $L^2 = 1.1$, $L^\infty = 5.0$. Table I supplies additional



L^2=1.03, L^∞=2.87
L^2=1.03, L^∞=2.78
L^2=1.07, L^∞=5.12
L^2=1.04, L^∞=3.29
L^2=1.06, L^∞=4.13

Fig. 13. Chartification and parameterization results produced by our algorithm. The top row shows the charts on the 3D surface; the bottom row shows iso-parameter lines over the surface and the texture atlases.

	bunny	horse	dino	feline	skull	dinosaur
# vertices	35k	48k	24k	75k	20k	56k
# faces	69k	97k	48k	150k	40k	112k
# charts	15	19	20	38	6	23
# charts before merge	21	32	36	67	6	39
packing ratio	0.72	0.69	0.66	0.65	0.71	0.70
L^2 stretch	1.01	1.03	1.04	1.07	1.03	1.06
L^∞ stretch	2.26	2.78	3.29	5.12	2.87	4.13
chart&param (s)	40	68	20	165	14	87
merging (s)	15	10	26	72	0	32
packing (s)	15	20	10	50	3	43
total (s)	70	98	56	287	17	162

Table I. Statistics and timings for models presented in the paper. Times were measured on an Intel Xeon 3.0G machine.

statistics, including running times, for the resulting atlases. The results demonstrate that our method produces low-stretch atlases with a small number of charts. The following sections demonstrate applications of our method and compare it to previous work.

9.1 Insensitivity to Noise and Mesh Resolution

Since our algorithm is based on the surface spectral analysis, it relies only on the geodesic distances between surface vertices, which are fairly insensitive to noise and resolution change. Figure 14 shows the results of applying our algorithm to a noisy bunny and a low resolution bunny (3,000 triangles). In both cases, the chart numbers of the texture atlases are similar to that in Figure 1, although the

	horse	feline	gargoyle
remeshing dimension	215×309	245×272	296×218
# defined vertices	47,655	45,563	47,151
# unique vertices	45,768	42,690	45,167
geometry PSNR	88.7	82.9	85.6
[Sander et al. 2003]			
remeshing dimension	281×228	478×133	466×138
# defined vertices	48,389	48,038	46,724
# unique vertices	41,857	35,956	41,961
geometry PSNR	84.6	79.5	83.8

Table II. Comparing Iso-charts and MCGIMs of [Sander et al. 2003]. Our PSNR results are approximately 3-4dB higher.

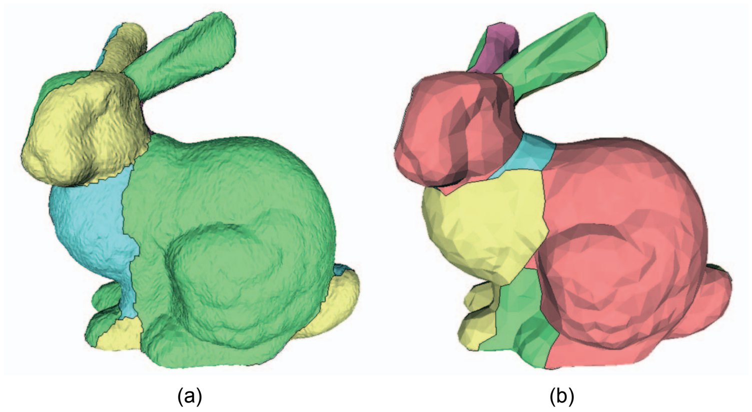


Fig. 14. Our algorithm is insensitive to noise addition and resolution change. (a) result for a bunny with added noise, with 17 charts and distortion $L^2 = 1.03$, $L^\infty = 3.44$. (b) result for a low resolution bunny, with 12 charts and distortion $L^2 = 1.06$, $L^\infty = 2.79$.

segmentation results exhibit minor differences.

9.2 Multi-Chart Geometry Images

Figure 15 and Table II compare our method to [Sander et al. 2003] for the construction of multi-chart geometry images. Results from [Sander et al. 2003] (middle column of Figure 15 and bottom 4 rows of Table II) are outputs of their software, obtained with their permission. The number of charts in their method is selected by hand, and was chosen by the authors to provide a reasonable balance between stretch and number of charts. Our method better preserves detail and exhibits fewer artifacts. Geometric accuracy numbers (PSNR, see [Sander et al. 2003] for a definition) confirm our advantage.

Following [Sander et al. 2003]'s terminology, *defined* vertices count those within charts, and so ignore wasted inter-chart space. *Unique* vertices count the number of unique MCGIM samples and so discount sampling redundancy along chart boundaries caused by "zippering" neighboring charts together. By bounding stretch using as few charts as possible, we create atlases having shorter boundaries yielding more unique vertices and thus better PSNR (3-4dB).

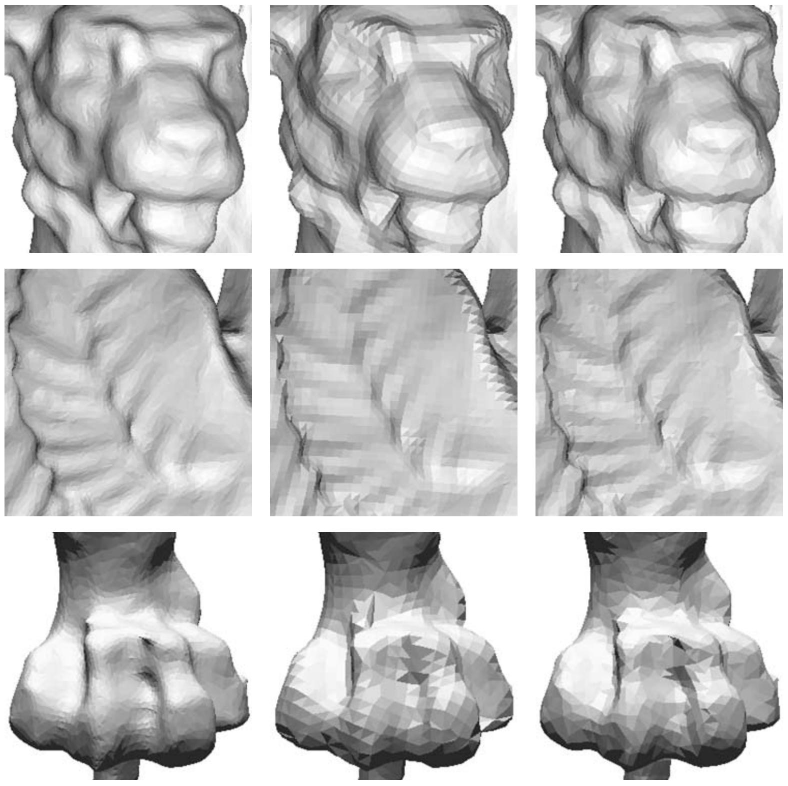


Fig. 15. Comparison of geometry remeshing methods. Geometry images have resolution 256×256.

9.3 Signal-Specialized Atlases

Our method provides a simple but effective way to specialize texture atlases to a particular signal in order to optimize use of limited texture samples. To show the effectiveness of both parts of our approach – parameterization and chartification – we applied three different atlas generation methods: one based only on geometric stretch (i.e., ignoring the particular signal), one that parameterizes based on signal stretch but still uses charts based on the geometry, and one that adapts both the charts and their parameterizations to the signal. Following [Sander et al. 2003], we measure *signal approximation error* (SAE) which integrates over the surface the difference between the original signal and one reconstructed from the texture map.

In Figure 16, our algorithm reduces SAE for a normal map from 18.7 to 11.5 by parameterizing based on signal stretch (16e) rather than geometric stretch (16d). Visual fidelity is correspondingly improved. These results are similar to those reported in [Sander et al. 2002], which cut charts manually without considering their signal contents and is therefore analogous to our result in 16e (though our method is automatic). When chart partitioning is also adapted to the signal (16f), a texture atlas with even smaller SAE and better visual fidelity is produced. Figure 17 shows a similar result for a color signal.

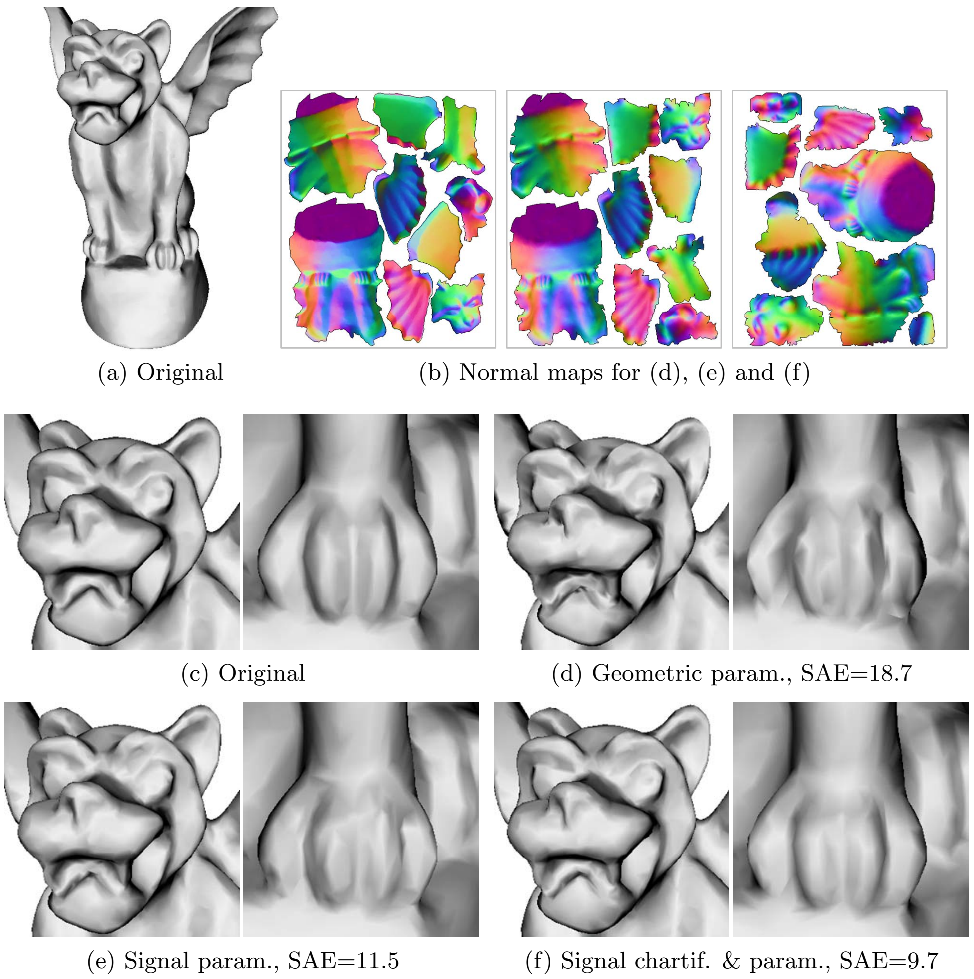

(a) Original
(b) Normal maps for (d), (e) and (f)
(c) Original
(d) Geometric param., SAE=18.7
(e) Signal param., SAE=11.5
(f) Signal chartif. & param., SAE=9.7

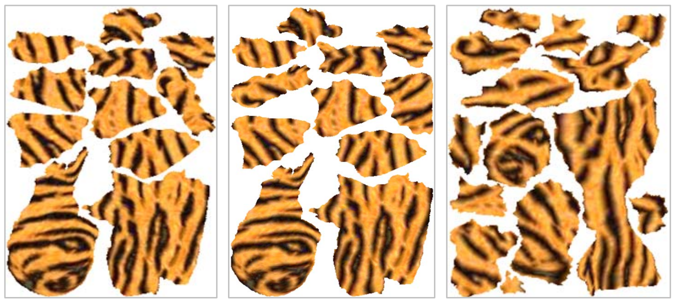
Fig. 16. Signal-specialized atlas results (normal map) (256×256 maps).

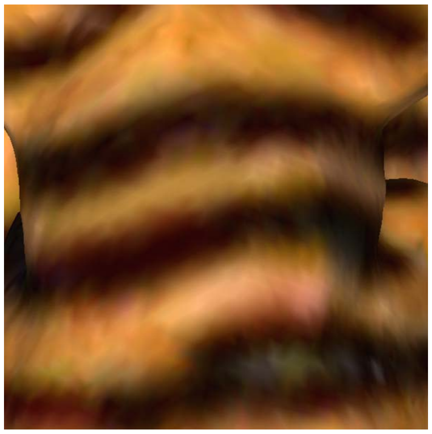
9.4 Texture Synthesis

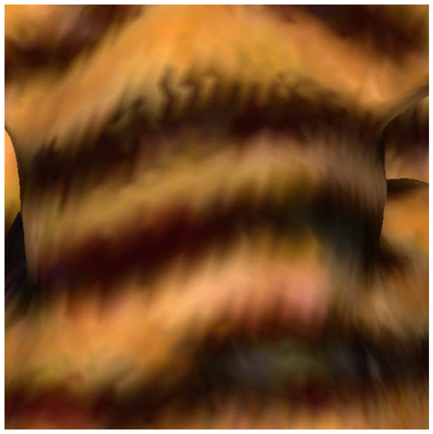
Figure 18 shows synthesis results from applying [Turk 2001] on the mesh pyramid generated by our method. Small stretch is crucial for generating a good synthesis result. To prevent undesirable distortion of the synthesized texture, we find the same stretch threshold $L^2 < 1.1$, $L^\infty < 5.0$ works well at the resolutions we use (256×256, 512×512). Our results look very natural with little shrinking or stretching of the texture pattern onto the surface.

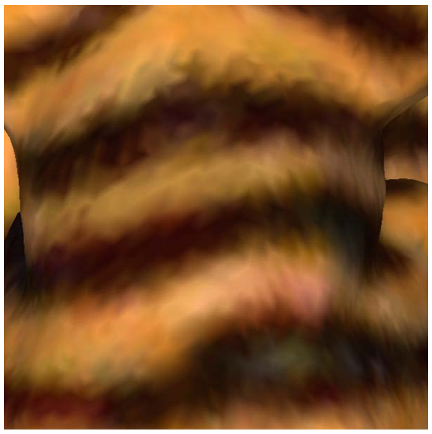
Figure 19 compares texture synthesis results from our atlases and the LSCM atlases of [Lévy et al. 2002]. Though we used the same stretch thresholds to terminate chart subdivision for both methods, LSCM does not limit stretch as well as our algorithm and produces more charts, wasting samples. The result is undesirable texture magnification over the bunny's nose, and a breaking up of the tile pattern

(a) Original


(b) Color maps for (d), (e) and (f)


(c) Original


(d) Geometric param., SAE=20.8


(e) Signal param., SAE=17.9

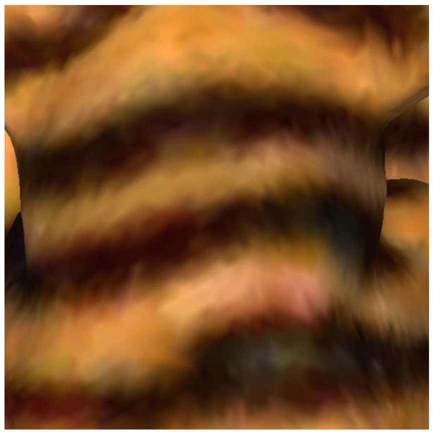

(f) Signal chartif. & param., SAE=16.5

Fig. 17. Signal-specialized atlas results (color map) (256×256 maps).

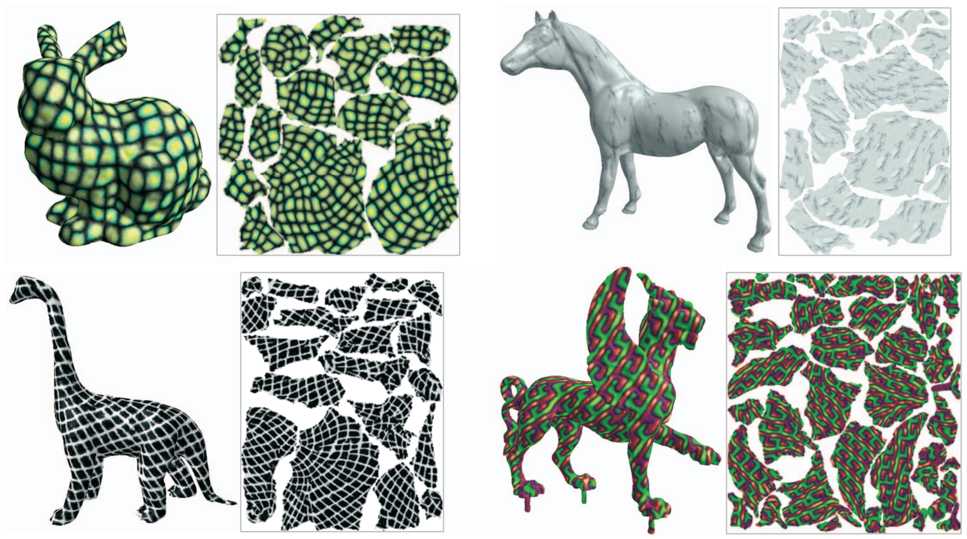


Fig. 18. Texture synthesis results using our multi-chart geometry images.

in the gargoyle's cheeks and ears. Our advantage would increase even further if the comparison was made using an equal number of charts.

10. CONCLUSION

Spectral analysis on the matrix of geodesic distances between points on a surface provides a fast, simple, and effective way to simultaneously solve two problems in atlas generation: partitioning the model into charts, and parameterizing those charts. It can also be simply generalized to account for signal rather than geometric distance, to optimize the atlas for a particular signal. We have shown that spectral analysis reduces stretch very well and provides a good starting point for further stretch minimization. Finally, we have shown how to incorporate these ideas in a stretch-driven atlas generator that improves results over previous techniques in geometry remeshing, texture map creation, and texture synthesis.

In future work, we would like to apply our algorithm to point geometry rather than dealing with global meshes, as in the original IsoMap paper. We are also interested in generalizing "special" spectral clustering to other categories of shapes besides tubes. Though we have thoroughly documented an empirical relation, the theoretical relation between stretch-minimizing and GDD-minimizing parameterizations has yet to be rigorously analyzed and understood. Intuitively though, both parameterizations are an attempt to limit distance distortions, and both heavily penalize mapping of large distances over the surface to small distances in the domain, rather than penalizing other distortion measures such as lack of conformality or area preservation.

REFERENCES

Bennis, C., Vezien, J. M., and Iglesias, G. 1991. Piecewise flattening for non-distorted texture mapping. In *Proceedings of SIGGRAPH 1991*. 237–246.

Brand, M. and Huang, K. 2003. A unifying theorem for spectral embedding and clustering. In

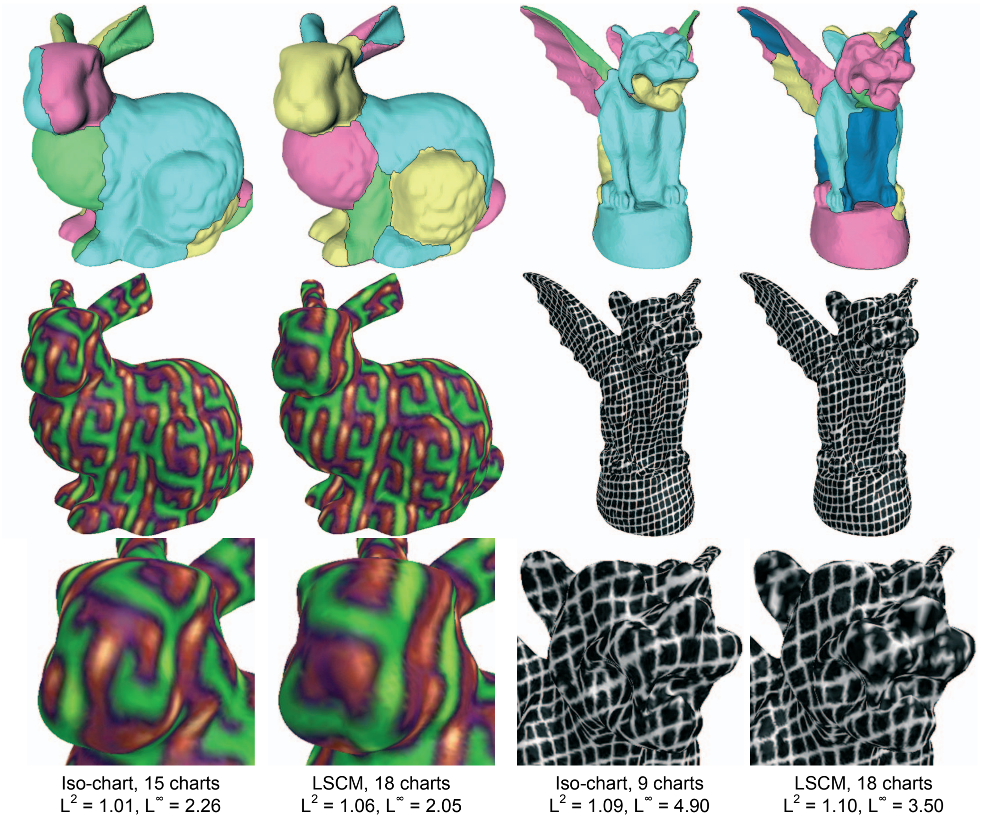

Iso-chart, 15 charts
L^2 = 1.01, L^∞ = 2.26
LSCM, 18 charts
L^2 = 1.06, L^∞ = 2.05
Iso-chart, 9 charts
L^2 = 1.09, L^∞ = 4.90
LSCM, 18 charts
L^2 = 1.10, L^∞ = 3.50

Fig. 19. Minimizing parameterization stretch without using too many charts is crucial for a high-quality synthesis result.

Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics 2003, C. M. Bishop and B. J. Frey, Eds.

Campagna, S. and Seidel, H. 1998. Parameterizing meshes with arbitrary topology. In *Proceedings of Image and Multidimensional Digital Signal*. 287–290.

Carr, N. and Hart, J. 2002. Meshed atlases for real-time procedural solid texturing. *ACM Transaction on Graphics 21,* 2, 106–131.

Desbrun, M., Meyer, M., and Alliez, P. 2002. Parameterizing meshes with arbitrary topology. In *Proceedings of Eurographics 2002*.

Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzle, W. 1995. Multiresolution analysis of arbitrary meshes. In *Proceedings of SIGGRAPH 1995*. 173–182.

Elad, A. and Kimmel, R. 2003. On bending invariant signatures for surfaces. *IEEE Transaction on PAMI 25,* 10, 1285–1295.

Floater, M. 1997. Parameterization and smooth approximation of surface triangulations. *CAGD 14,* 3, 231–250.

Floater, M. 2003. Mean value coordinates. *CAGD 20,* 1, 19–27.

Floater, M. and Hormann, K. 2004. Surface parameterization: a tutorial and survey. In *Advances in Multiresolution analysis of Geometric Modelling*, M. F. N.A. Dodgson and M. Sabin, Eds. Springer, 259–284.

Garland, M. and Heckbert, P. 1997. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH 1997*. 209–216.

Gotsman, C., Gu, X., and Sheffer, A. 2003. Fundamentals of spherical parameterization for 3d meshes. In *Proceedings of SIGGRAPH 2003*. 358–363.

Gu, X., Gortler, S., and Hoppe, H. 2003. Geometry images. In *Proceedings of SIGGRAPH 2002*. 355–361.

Guskov, I., Vidimce, K., Sweldens, W., and Schröder, P. 2000. Normal meshes. In *Proceedings of SIGGRAPH 2000*. 95–102.

Hoppe, H. 1996. Progressive mesh. In *Proceedings of SIGGRAPH 1996*. 99–108.

Hormann, K. and Greiner, G. 1999. Mips: An efficient global parameterization method. In *Curve and Surface Design: Saint-Malo*. Vanderbilt University Press, 219–226.

Karni, Z. and Gotsman, C. 2000. Spectral compression of mesh geometry. In *Proceedings of SIGGRAPH 2000*. 279–286.

Katz, S. and Tal, A. 2003. Hierachical mesh decomposition using fuzzy clustering and cuts. In *Proceedings of SIGGRAPH 2003*. 954–961.

Khodakovsky, A., Litke, N., and Schröder, P. 2003. Globally smooth parameterizations with low distortion. In *Proceedings of SIGGRAPH 2003*. 350–357.

Kimmel, R. and Sethian, J. 1998. Computing geodesics on manifolds. In *Proceedings of Nat'l Academy Sciences*. 8431–8435.

Krishnamurthy, V. and Levoy, M. 1996. Fitting smooth surfaces to dense polygon meshes. In *Proceedings of SIGGRAPH 1996*. 313–324.

Lee, A., Sweldens, W., Schröder, P., Cowsar, L., and Dobkin, D. 1998. Maps: multiresolution adaptive parameterization of surfaces. In *Proceedings of SIGGRAPH 1998*. 95–104.

Lévy, B. and Mallet, J.-L. 1998. Non-distortion texture mapping for sheared triangulated meshes. In *Proceedings of SIGGRAPH 1998*. 343–352.

Lévy, B., Petitjean, S., Ray, N., and Mallet, J.-L. 2002. Least squares conformal maps for automatic texture atlas generation. In *Proceedings of SIGGRAPH 2002*. 362–371.

Maillot, J., Yahia, H., and Verroust, A. 1993. Interactive texture mapping. In *Proceedings of SIGGRAPH 1993*. 27–34.

Pedersen, H. 1995. Decorating implicit surfaces. In *Proceedings of SIGGRAPH 1995*. 291–300.

Peyré, G. and Cohen, L. 2004. Geodesic computations for fast and accurate surface flattening. *Preprint CMAP n536*.

Roweis, S. and Saul, L. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 2323–2326.

Sander, P., Gortler, S., Snyder, J., and Hoppe, H. 2002. Signal-specialized parameterization. In *Proceedings of Eurographics Workshop on Rendering 2002*.

Sander, P., Snyder, J., Gortler, S., and Hoppe, H. 2001. Texture mapping progressive meshes. In *Proceedings of SIGGRAPH 2001*. 409–416.

Sander, P., Wood, Z., Gortler, S., Snyder, J., and Hoppe, H. 2003. Multi-chart geometry images. In *Symposium on Geometry Processing 2003*. 146–155.

Sheffer, A. and Hart, J. 2002. Seamster: inconspicuous low-distortion texture seam layout. In *Proceedings of IEEE Visualization 2002*. 291–298.

Silva, V. and Tenenbaum, J. 2002. Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems*. 705–712.

Sorkine, G., Cohen-Or, D., Goldenthal, R., and Lischinski, D. 2002. Bounded-distortion piecewise mesh parameterization. In *Proceedings of IEEE Visualization 2002*. 355–362.

Taubin, G. 1995. A signal processing approach to fair surface design. In *Proceedings of SIGGRAPH 1995*. 351–358.

Tenenbaum, J., Silva, V., and Langford, J. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science 290*, 2319–2323.

Turk, G. 1991. Generating textures for arbitrary surfaces using reaction-diffusion. In *Proceedings of SIGGRAPH 1991*. 289–298.

Turk, G. 2001. Texture synthesis on surfaces. In *Proceedings of SIGGRAPH 2001*. 347–354.

Wei, L. and Levoy, M. 2001. Texture synthesis over arbitrary manifold surfaces. In *Proceedings of SIGGRAPH 2001*. 355–360.

ZHANG, E., MISCHAIKOW, K., AND TURK, G. 2004. Feature-based surface parameterization and texture mapping. *to appear in ACM Transaction on Graphics*.

ZIGELMAN, G., KIMMEL, R., AND KIRYATI, N. 2002. Texture mapping using surface flattening via multidimensional scaling. *IEEE Transactions on Visualization and Computer Graphics 8,* 2, 198–207.