Eurographics Symposium on Geometry Processing (2004)
R. Scopigno, D. Zorin, (Editors)

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

Kun Zhou [1] John Synder [2] Baining Guo [1] Heung-Yeung Shum [1]

[1] Microsoft Research Asia [2] Microsoft Research

**Abstract**

*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" or MDS (multi-dimensional scaling) parameterization, based on an eigen-analysis of the matrix of squared geodesic distances between pairs of mesh vertices. We show that only a few iterations of nonlinear stretch optimization need be applied to the MDS parameterization to obtain low-stretch atlases. The close relationship we discover between these two parameterizations also allows us to apply spectral clustering based on MDS 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. 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 (according to ACM CCS): I.3.3 [Computer Graphics]: Mesh, parameterization

## 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 [MYV93, Ped95, LPRM02, SSGH01]. The target surface is first partitioned into a set of charts, called chartification, which are parameterized and packed into the texture domain.

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 [SSGH01]'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 [SSGH01], and is asymptotically related to geometric accuracy under piecewise-constant reconstruction [SGSH02].

Unfortunately, minimizing stretch requires nonlinear optimization. This leads to two difficulties: stretch minimization is slow, and it disregards stretch when forming charts in favor of unrelated heuristics that cut across sharp features or cluster based on chart compactness and planarity. The latter is true because if computing a stretch-minimizing embedding for a chart is costly then computing it over all possible chart partitionings is completely impractical.

To solve this problem, we apply a form of nonlinear dimensionality reduction called IsoMap [TSL00] 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 with little stretch. Without any extra computation, it also supplies an initial pa-

rameterization for each part. In fact, we show that it provides an excellent starting point for stretch minimization, so that a few iterations of nonlinear stetch optimization quickly converge and remove problem "fold-overs" easily.

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 [KL96]. Early work [MYV93] clusters triangles according to their normals. Other methods produce charts with convex boundaries [EDD*95, GVSS00, KLS03, LSS*98, SSGH01], a restriction that can significantly increase stretch.

The least squares conformal map (LSCM) [LPRM02] 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 [DMA02] is another free-boundary, conformal atlas approach. Though it preserves angles, conformal parameterization yields area and distance distortion that is undesirable in many applications. [SWG*03] 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.

[SCOGL02] 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 [GGH03] parameterize an entire surface over a single 2D square using cuts through vertices having extreme stretch. *Seamster* [SH02] 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.

Many algorithms can parameterize charts over planar regions [BVI91, CS98, DMA02, EDD*95, Flo97, Flo03] [HG99, LM98, LPRM02, MYV93, SSGH01]. We refer the interested reader to the recent survey by [FH04]. Like our method, [ZKK02] 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.

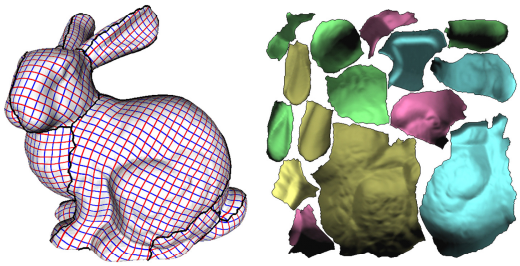


**Figure 1:** *Iso-chart atlas for the Stanford bunny. The model is partitioned into 15 large charts, which can be flattened with lower stretch than previous methods ($L^2$ = 1.01, $L^\infty$ = 2.26).*

A form of spectral analysis has been used for geometry compression and smoothing [KG00, Tau95]. These methods are based on the mesh's Laplacian which depends only on its connectivity, rather than geodesic distance. Recently, [GGS03] applies spectral graph theory to solve the spherical parameterization problem.

We apply stretch-minimization [SSGH01] to optimize parameterizations created by spectral analysis. We also use geometric stretch [SSGH01] or signal stretch [SGSH02] to determine when to terminate chart subdivision.

[KT03] 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 [SSGH01].

3. If the stretch of this derived parameterization is less than the threshold, stop.
4. Perform spectral clustering to partition the surface into charts (Section 4.3).
5. Optimize chart boundaries using the graph cut technique (Section 4.4).
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 [SSGH01]. The $L^2$ norm integrates $(\gamma_{max}^2+\gamma_{min}^2)/2$ over the surface, followed by an overall square root. The $L^\infty$ norm maximizes $\max\{\gamma_{max}, 1/\gamma_{min}\}$ over the entire surface. Here, $\gamma_{max}$ and $\gamma_{min}$ 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 $L^\infty$ 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. Chartification and Parameterization

### 4.1. Surface Spectral Analysis

Our algorithm builds upon the dimensionality reduction method IsoMap (isometric feature mapping) [TSL00]. Given a set of high-dimensional points, IsoMap computes the geodesic distances along a manifold as sequences of hops between neighboring points. It then applies MDS (multidimensional scaling) to these geodesic distances to find a set of points embedded in low-dimensional space with similar pairwise distances.

We refer to this application of IsoMap as *surface spectral analysis* and outline its computation. Given a surface with $N$ vertices,

- 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 is used to constrain the center of gravity of the set of pairwise distances to lie at the origin.
- Compute the eigenvalues $\lambda_i$ 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 $\lambda_i$ 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.

Although $N$ eigenvalues are needed to fully represent a surface with $N$ vertices, 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. Note that we are trying to find an embedding in a higher dimensional space such that Euclidean distances in that higher dimensional space match geodesic distances on the manifold. This generally can not be done in a 3-dimensional space for arbitrary models.

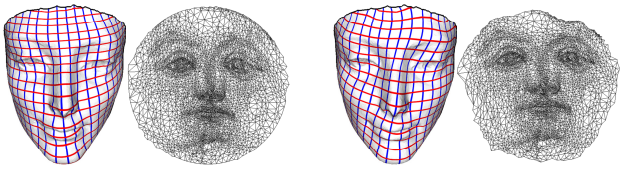
The distortion of this $n$-dimensional embedding can be calculated as the sum of the geodesic distance distortion over all vertices. For each vertex $i$, its *geodesic distance distortion* (GDD) under the embedding is defined as:

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

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 4.4.

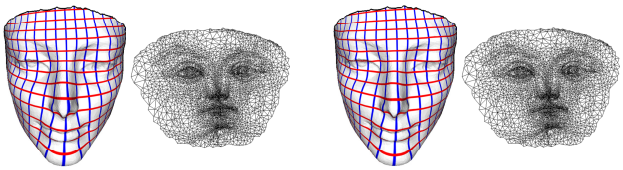
When $n = 2$, surface spectral analysis yields a surface parameterization minimizing the sum of squared GDD over all vertices. This is the key idea of the mapping algorithm in [ZKK02]. Our observation is that the same technique, surface spectral analysis, can be simultaneously applied to two critical problems: decomposition necessary for chartification, and parameterization.

To calculate the geodesic distances between surface points of polygonal models, we use the fast matching method [KS98], 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.



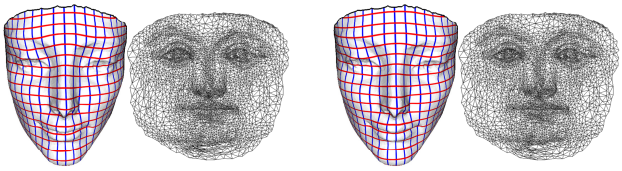
(a) Uniform + fine-level stretch [SSGH01]
$L^2 = 1.04, L^\infty = 1.68$
PSNR=76.8dB, running time 222s

(b) Uniform + coarse-to-fine stretch [SGSH02]
$L^2 = 1.03, L^\infty = 2.16$
PSNR=76.8dB, running time 39s



(c) LSCM [LPRM02]
$L^2 = 1.11, L^\infty = 2.27$
PSNR=74.9dB, running time 3s

(d) LSCM + fine-level stretch [LPRM02] + [SSGH01]
$L^2 = 1.09, L^\infty = 2.27$
PSNR=75.5dB, running time 26s



(e) IsoMap [ZKK02]
$L^2 = 1.04, L^\infty = 2.78$
PSNR=77.0dB, running time 2s

(f) IsoMap + fine-level stretch [ZKK02]+[SSGH01]
$L^2 = 1.03, L^\infty = 2.32$
PSNR=77.0dB, running time 6s

**Figure 2:** *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).*

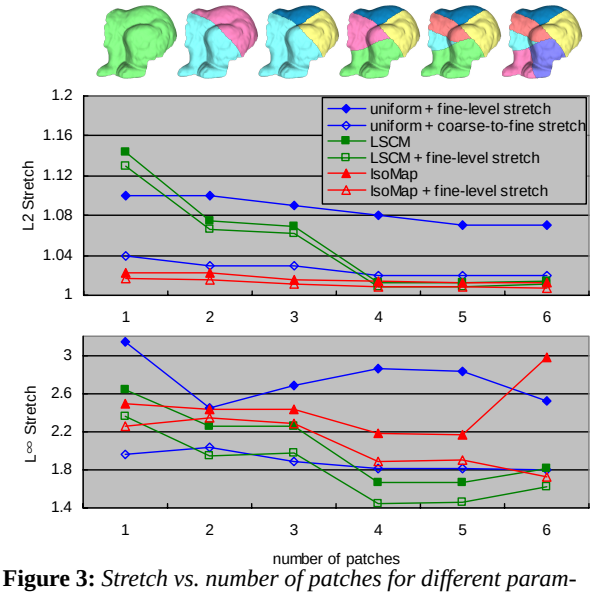



**Figure 3:** *Stretch vs. number of patches for different parameterizations.*

### 4.2. Spectral Analysis and Stretch

GDD-minimizing [ZKK02] and stretch-minimizing [SSGH01] 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 [ZKK02]: 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.

Figure 2 shows parameterization results for a single chart model of a face. While LSCM (2c-d) is fast, it is conformal and so limits stretch poorly. IsoMap (2e) quickly provides a parameterization having little $L^2$ stretch – in fact, it's stretch is as good as the slow optimization method of [SSGH01] (2a). Furthermore, it provides a starting point which a few $L^2$-stretch-minimizing iterations improves even further (2f), 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 [SGSH02]. Since it depends on only a single, worst-case point in the domain, $L^\infty$ 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 3 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 figure. As in Figure 2, the best $L^2$ stretch result is

obtained by our methods. 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" ([SSGH01]) lags behind, because its initial 2D parameterization domain is a circle while the other methods adopt more natural domain shapes. As in Figure 2, $L^\infty$ stretch is more haphazard, and a few $L^2$ stretch-minimizing iterations improves our result significantly.

### 4.3. Surface 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*. The algorithm is as follows:

1. Rank the eigenvalues $\lambda_i$ 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 $\lambda_n/\lambda_{n+1}$ is maximized.
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.
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 which has the closest representative to the triangle.

Step 2 amounts to a relative error threshold that finds the "knee" in the curve relating energy to the number of eigenvalues. 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 5), which in turn restricts the maximum number of sub-charts.

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.

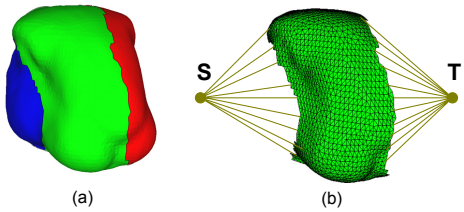



**Figure 4:** *Finding the optimal partition boundary is formulated as a graph cut problem. (a) the shape is decomposed into three parts, lateral areas* ***A*** *(red),* ***B*** *(blue) and medial area* ***C*** *(green). (b) constructing a graph for the medial area.*

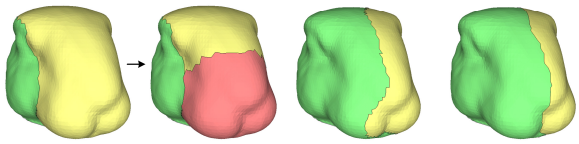



**Figure 5:** *Comparing different graph-cut capacities.*

### 4.4. Computing Optimal Partition Boundaries

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 [SSGH01, LPRM02, SWG*03], which minimize various measures of chart compactness while choosing chart cuts of shortest length or along edges having high dihedral angle. Recently [KT03] 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 4a 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 4b) can be constructed from **C** using an extension of the method in [KT03]. We modify their definition of

"capacity" between the two adjacent triangles $f_i$ and $f_j$ as

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

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 [KT03]:

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

where $d_{ang}(f_i, f_j)$ is defined as $(1-\cos\alpha_{ij})$, $\alpha_{ij}$ 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

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

$$\begin{aligned} d_{distort}(f_i, f_j) &= |GDD_A(f_i) - GDD_B(f_i)| \\ &+ |GDD_A(f_j) - GDD_B(f_j)| \end{aligned}$$

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

The weight parameter $\alpha$ trades off the two objectives. $\alpha = 1$ defines capacity as in [KT03] 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 5a). But this split produces too much stretch in the right chart, which must be split again to satisfy the user's threshold (right side of Figure 5a).

On the other hand, we can set $\alpha = 0$ to minimize GDD as Figure 5b illustrates, which avoids chart proliferation but makes the boundary jaggier. Figure 5c sets $\alpha = 0.5$. Although the parameterization stretch is a little larger than 5b, a smoother boundary is desirable for many applications.

#### 4.4.1. 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 [ST02], 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 $\lambda_i$ 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 [ST02]. Let $\Delta_p$ 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 $\mathbf{L}_A$:

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

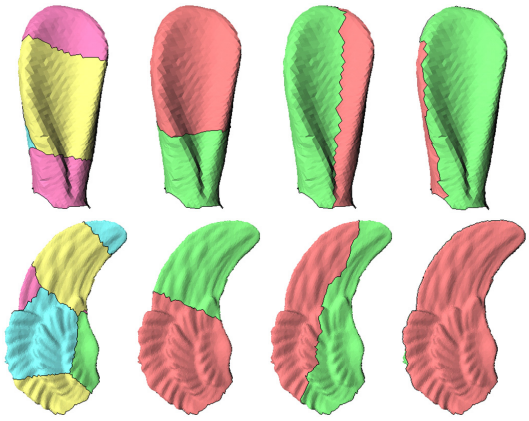
Now we can calculate GDDs for all vertices in **C** under the embedding induced by **A**, and similarly for **B**.

### 4.5. 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 6a, 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 6b).

A better method (Figure 6d) is inspired by recent work in computer vision [BH03, EK03], 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 $\lambda_i$ 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,$\lambda$=$\lambda_1$ (c) $n$=1,$\lambda$=$\lambda_2$ (d) $n$=1,$\lambda$=$\lambda_3$

**Figure 6:** *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 sub-charts. As noted by [BH03], 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 6d 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 4.3). 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 non-binary chart subdivision is performed (at the first iteration); thereafter, binary subdivision suffices.

## 5. 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 [ST02], 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 4.4.1.

To get the landmark points, models are simplified by performing half edge collapse operations based on the quadric error metric [GH97]. Progressive meshes [Hop96] 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 $O(qN\log N)$ and spectral decomposition to $O(q^2)$.

## 6. 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, [SGSH02] defines a signal-stretch metric and develops an iterated multi-grid strategy to minimize it over manually created charts.

We extend our approach to surface signals by computing 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 them, and therefore ties our algorithm to signal distortion in the same way as our previous algorithm was tied to geometric distortion.

Typical signals such as textured colors 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. This same problem led [SGSH02] to combine geometric and signal stretch; our solution defines distance with a similar combination of geodesic and signal distances:

$$d_{comb}(i,j) = \beta \frac{d_{geo}(i,j)}{avg(d_{geo})} + (1-\beta)\frac{d_{sig}(i,j)}{avg(d_{sig})} \qquad (5)$$

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

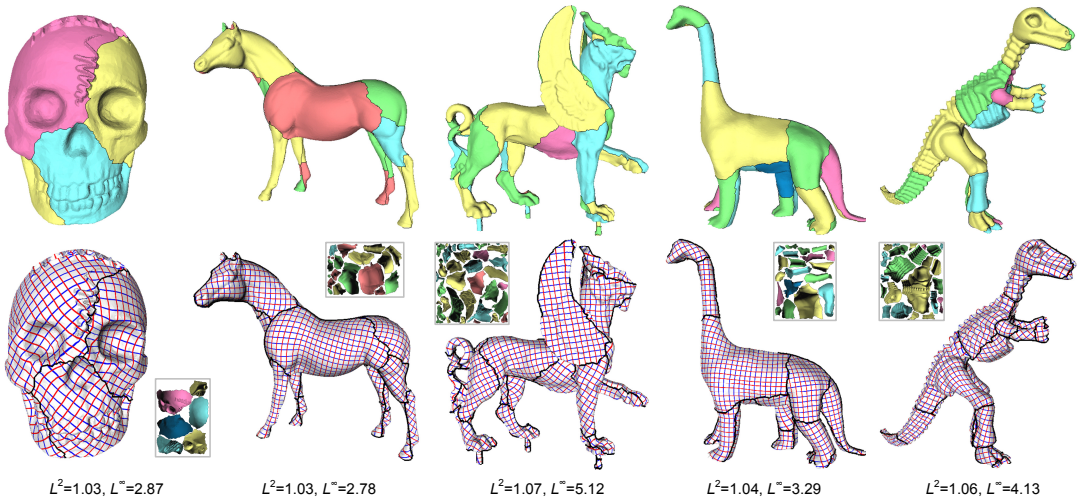



**Figure 7:** *Chartification and parameterization results produced by our algorithm. The top row shows the charts on the 3D surface; the bottom row shows iso-parameter lines over the surface and the texture atlases.*

| | 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 |
| $L^\infty$ 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 1:** *Statistics and timings for models presented in the paper. Times were measured on an Intel Xeon 3.0G machine.*

| | 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 |
| [SWG*03] | | | |
| 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 2:** *Comparing Iso-charts and MCGIMs of [SWG*03]. Our PSNR results are approximately 3-4dB higher.*

## 7. Results

Figures 1 and 7 illustrate results on several models. For all experiments, the stretch threshold was set at $L^2 = 1.1, L^\infty = 5.0$. To generate the texture atlases shown in the bottom row, we use the method proposed by [SWG*03] to pack the charts together. Table 1 supplies additional 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.

**Multi-Chart Geometry Images** Figure 8 and Table 2 compare our method to [SWG*03] for the construction of multi-chart geometry images. Results from [SWG*03] (middle column of Figure 8 and bottom 4 rows of Table 2) 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 [SWG*03] for a definition) confirm our advantage.

Following [SWG*03]'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).

**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 param-

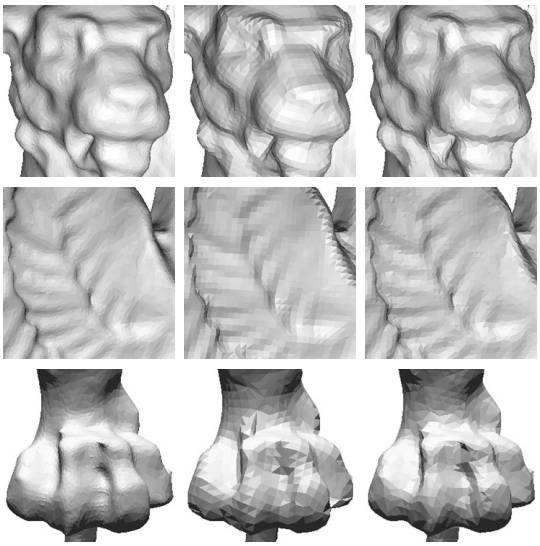


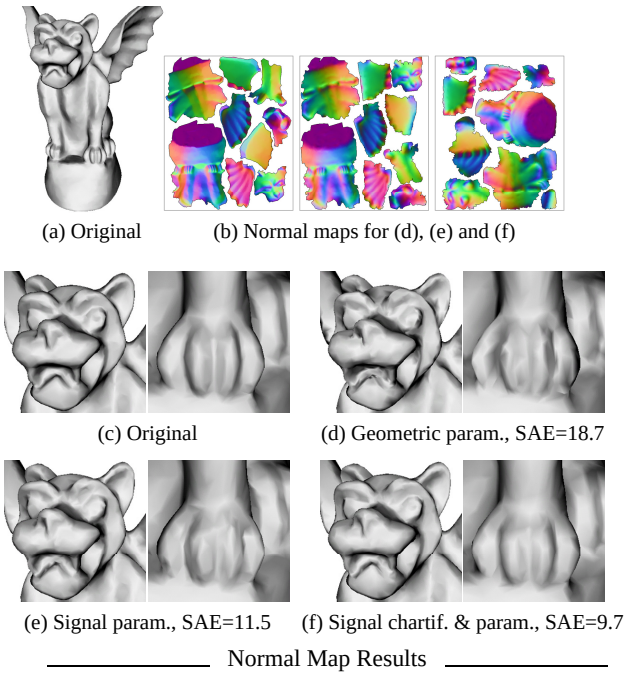

**Figure 8:** *Comparison of geometry remeshing methods. Geometry images have resolution* $256 \times 256$.

eterizes 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 [SWG*03], 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 9 (top), our algorithm reduces SAE for a normal map from 18.7 to 11.5 by parameterizing based on signal stretch (9e) rather than geometric stretch (9d). Visual fidelity is correspondingly improved. These results are similar to those reported in [SGSH02], which cut charts manually without considering their signal contents and is therefore analogous to our result in 9e (though our method is automatic). When chart partitioning is also adapted to the signal (9f), a texture atlas with even smaller SAE and better visual fidelity is produced. The bottom of the figure shows a similar result for a color signal.

## 8. 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 and texture map creation.



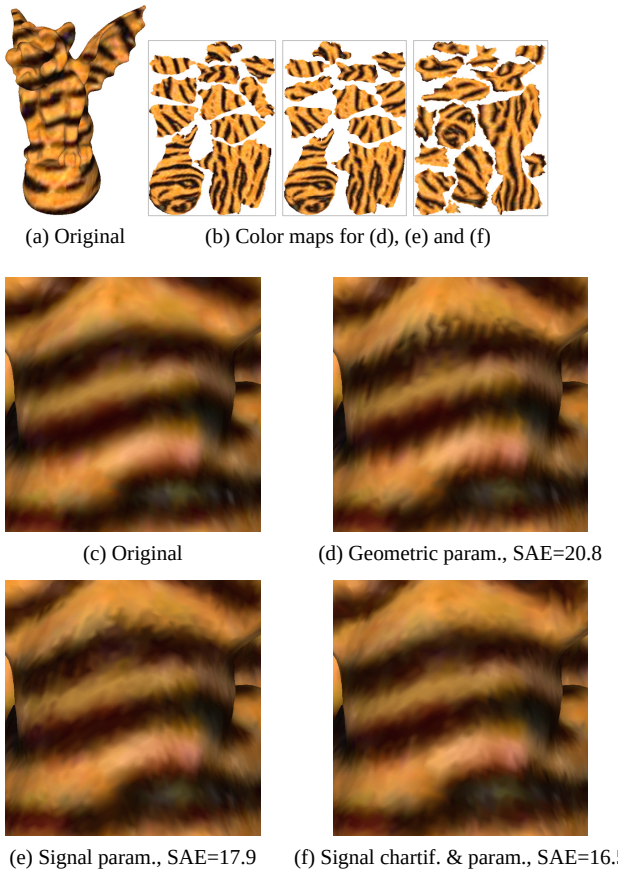




**Figure 9:** *Signal-specialized atlas results (*$256 \times 256$ *maps).*

## References


[BH03] BRAND M., HUANG K.: A unifying theorem for spectral embedding and clustering. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics 2003* (2003), Bishop C. M., Frey B. J., (Eds.).

[BVI91] BENNIS C., VEZIEN J. M., IGLESIAS G.: Piecewise flattening for non-distorted texture mapping. In *Proceedings of SIGGRAPH 1991* (1991), pp. 237–246.

[CS98] CAMPAGNA S., SEIDEL H.: Parameterizing meshes with arbitrary topology. In *Proceedings of Image and Multidimensional Digital Signal* (1998), pp. 287–290.

[DMA02] DESBRUN M., MEYER M., ALLIEZ P.: Parameterizing meshes with arbitrary topology. In *Proceedings of Eurographics 2002* (2002).

[EDD*95] ECK M., DEROSE T., DUCHAMP T., HOPPE H., LOUNSBERY M., STUETZLE W.: Multiresolution analysis of arbitrary meshes. In *Proceedings of SIGGRAPH 1995* (1995), pp. 173–182.

[EK03] ELAD A., KIMMEL R.: On bending invariant signatures for surfaces. *IEEE Transaction on PAMI 25*, 10 (2003), 1285–1295.

[FH04] FLOATER M., HORMANN K.: Surface parameterization: a tutorial and survey. In *Advances in Multiresolution analysis of Geometric Modelling* (2004), N.A. Dodgson M. F., Sabin M., (Eds.), Springer, pp. 259–284.

[Flo97] FLOATER M.: Parameterization and smooth approximation of surface triangulations. *CAGD 14*, 3 (1997), 231–250.

[Flo03] FLOATER M.: Mean value coordinates. *CAGD 20*, 1 (2003), 19–27.

[GGH03] GU X., GORTLER S., HOPPE H.: Geometry images. In *Proceedings of SIGGRAPH 2002* (2003), pp. 355–361.

[GGS03] GOTSMAN C., GU X., SHEFFER A.: Fundamentals of spherical parameterization for 3d meshes. In *Proceedings of SIGGRAPH 2003* (2003), pp. 358–363.

[GH97] GARLAND M., HECKBERT P.: Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH 1997* (1997), pp. 209–216.

[GVSS00] GUSKOV I., VIDIMCE K., SWELDENS W., SCHRÖDER P.: Normal meshes. In *Proceedings of SIGGRAPH 2000* (2000), pp. 95–102.

[HG99] HORMANN K., GREINER G.: Mips: An efficient global parameterization method. In *Curve and Surface Design: Saint-Malo* (1999), Vanderbilt University Press, pp. 219–226.

[Hop96] HOPPE H.: Progressive mesh. In *Proceedings of SIGGRAPH 1996* (1996), pp. 99–108.

[KG00] KARNI Z., GOTSMAN C.: Spectral compression of mesh geometry. In *Proceedings of SIGGRAPH 2000* (2000), pp. 279–286.

[KL96] KRISHNAMURTHY V., LEVOY M.: Fitting smooth surfaces to dense polygon meshes. In *Proceedings of SIGGRAPH 1996* (1996), pp. 313–324.

[KLS03] KHODAKOVSKY A., LITKE N., SCHRÖDER P.: Globally smooth parameterizations with low distortion. In *Proceedings of SIGGRAPH 2003* (2003), pp. 350–357.

[KS98] KIMMEL R., SETHIAN J.: Computing geodesics on manifolds. In *Proceedings of Nat'l Academy Sciences* (1998), pp. 8431–8435.

[KT03] KATZ S., TAL A.: Hierachical mesh decomposition using fuzzy clustering and cuts. In *Proceedings of SIGGRAPH 2003* (2003), pp. 954–961.

[LM98] LÉVY B., MALLET J.-L.: Non-distortion texture mapping for sheared triangulated meshes. In *Proceedings of SIGGRAPH 1998* (1998), pp. 343–352.

[LPRM02] LÉVY B., PETITJEAN S., RAY N., MALLET J.-L.: Least squares conformal maps for automatic texture atlas generation. In *Proceedings of SIGGRAPH 2002* (2002), pp. 362–371.

[LSS*98] LEE A., SWELDENS W., SCHRÖDER P., COWSAR L., DOBKIN D.: Maps: multi-resolution adaptive parameterization of surfaces. In *Proceedings of SIGGRAPH 1998* (1998), pp. 95–104.

[MYV93] MAILLOT J., YAHIA H., VERROUST A.: Interactive texture mapping. In *Proceedings of SIGGRAPH 1993* (1993), pp. 27–34.

[Ped95] PEDERSEN H.: Decorating implicit surfaces. In *Proceedings of SIGGRAPH 1995* (1995), pp. 291–300.

[SCOGL02]SORKINE G., COHEN-OR D., GOLDENTHAL R., LISCHINSKI D.: Bounded-distortion piecewise mesh parameterization. In *Proceedings of IEEE Visualization 2002* (2002), pp. 355–362.

[SGSH02] SANDER P., GORTLER S., SNYDER J., HOPPE H.: Signal-specialized parameterization. In *Proceedings of Eurographics Workshop on Rendering 2002* (2002).

[SH02] SHEFFER A., HART J.: Seamster: inconspicuous low-distortion texture seam layout. In *Proceedings of IEEE Visualization 2002* (2002), pp. 291–298.

[SSGH01] SANDER P., SNYDER J., GORTLER S., HOPPE H.: Texture mapping progressive meshes. In *Proceedings of SIGGRAPH 2001* (2001), pp. 409–416.

[ST02] SILVA V., TENENBAUM J.: Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems* (2002), pp. 705–712.

[SWG*03] SANDER P., WOOD Z., GORTLER S., SNYDER J., HOPPE H.: Multi-chart geometry images. In *Symposium on Geometry Processing 2003* (2003), pp. 146–155.

[Tau95] TAUBIN G.: A signal processing approach to fair surface design. In *Proceedings of SIGGRAPH 1995* (1995), pp. 351–358.

[TSL00] TENENBAUM J., SILVA V., LANGFORD J.: A global geometric framework for nonlinear dimensionality reduction. *Science 290* (2000), 2319–2323.

[ZKK02] ZIGELMAN G., KIMMEL R., KIRYATI N.: Texture mapping using surface flattening via multidimensional scaling. *IEEE Transactions on Visualization and Computer Graphics 8*, 2 (2002), 198–207.