

Signal-Specialized Parametrization

Pedro V. Sander¹, Steven J. Gortler¹, John Snyder² and Hugues Hoppe²

¹Harvard University, Cambridge, MA, USA

²Microsoft Research, Redmond, WA, USA

Abstract

To reduce memory requirements for texture mapping a model, we build a surface parametrization specialized to its signal (such as color or normal). Intuitively, we want to allocate more texture samples in regions with greater signal detail. Our approach is to minimize signal approximation error — the difference between the original surface signal and its reconstruction from the sampled texture. Specifically, our signal-stretch parametrization metric is derived from a Taylor expansion of signal error. For fast evaluation, this metric is pre-integrated over the surface as a metric tensor. We minimize this nonlinear metric using a novel coarse-to-fine hierarchical solver, further accelerated with a fine-to-coarse propagation of the integrated metric tensor. Use of metric tensors permits anisotropic squashing of the parametrization along directions of low signal gradient. Texture area can often be reduced by a factor of 4 for a desired signal accuracy compared to non-specialized parametrizations.

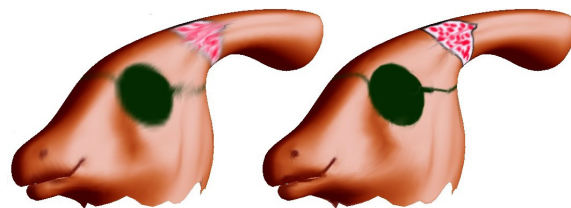
Categories and Subject Descriptors: I.3.7 [Computer Graphics]: Color, shading, shadowing, and texture.

1. Introduction

Real-time rendering is making increasing use of texture-mapping features provided by graphics hardware. Advanced rasterization features include multi-texturing, shader expression trees, and programmable pixel shaders. Surface signals are used to achieve a variety of rendering effects, including traditional color mapping, bump mapping (where surface normals are the signal), displacement mapping (geometry), and self-shadowing (horizon maps²¹ and polynomial texture maps²⁰).

While these rendering effects can also be computed in vertex shaders, texture mapping is advantageous because storing and processing texture images is generally more efficient than refining the carrier geometry to represent the detailed signal at the vertices of a dense mesh.³

To allow texture mapping, a surface must be parametrized onto a texture domain by assigning texture coordinates to its vertices. Given this parametrization, the surface signal is sampled into a texture image of the appropriate resolution. Texture memory can become a scarce resource in complex scenes with many textured objects. In this work, we examine how to construct a parametrization to best represent a given surface signal using textures as compact as possible. We construct such a parametrization as an off-line, automatic preprocess.



(a) geometry-based parametrization

(b) signal-specialized parametrization

Figure 1: Comparison of geometric-stretch and signal-stretch parametrizations. The 3D-painted surface signal is captured into the 128x128 textures shown above. The blurriness on the left is due to the parametrization's under-sampling of detail, which is improved on the right.

Most surface parametrization schemes to date assume no *a priori* knowledge of the signal, and instead minimize various geometric distortion measures. In this paper, we consider the problem of building a surface parametrization optimized for a specific signal. Intuitively, we want the parametrization to automatically allocate more texture samples to regions with greater signal detail. Heuristics having this behavior could be defined in various ways. We follow a principled approach, which is to minimize the *signal approximation error* — the difference between the reconstructed signal and the original surface signal.

Our first contribution is a signal-specialized metric:

- The *signal-stretch* metric integrates signal approximation error over the surface (§3.2). It is derived using a Taylor expansion of signal error (see appendix).
- During a pre-computation, each mesh face is assigned an *integrated metric tensor* (IMT), which encapsulates how much the signal varies over the face, and in what primary direction (§3.3).
- For fast evaluation, *affine transformation rules* can exactly transform these triangle-based IMTs (§3.4).

Our second contribution is an efficient parametrization algorithm to minimize this metric using a multiresolution hierarchy:

- The IMTs are computed on the fine mesh triangles, and propagated fine-to-coarse in the hierarchy (§4.1).
- The chart is parametrized using a coarse-to-fine optimization, based on affinely transformed IMTs (§4.2).
- The chart boundary parametrization is allowed to move during optimization, while still preserving an embedding (§4.4).
- As a post-process, the chart is relaxed to its tightest bounding rectangle, to exploit unused texture space (§4.5).

The new metric and algorithm are incorporated in a system for creating signal-specialized parametrizations of meshes. By allocating greater texture density to surface regions with signal detail, the resulting parametrizations reduce signal approximation error for a given texture size (Figure 1), or permit smaller textures for a given approximation error (Figure 2). For our examples, the same signal accuracy is achieved with a factor 3-5 savings in texture samples compared to a signal-independent parametrization.

2. Previous work

Signal-independent chart parametrization. Most schemes for flattening a surface chart into 2D minimize a geometric distortion metric, which assumes no knowledge of the surface signal. Many of these distortion metrics are tailored for the authoring problem of mapping an existing image onto a surface mesh, rather than the problem of sampling a given surface signal.

Eck et al.⁵ propose the discrete harmonic map, which assigns non-uniform springs to the mesh edges. Duchamp et al.⁴ investigate multiresolution solution methods for computing harmonic maps. Floater⁶ proposes a similar metric with a different edge-spring weighting that guarantees an embedding for convex boundaries. Haker et al.⁹ compute conformal maps onto the sphere. Hormann and Greiner¹² propose the MIPS parametrization, which attempts to preserve the ratio of singular values over the parametrization. All four of these metrics disregard absolute stretch scale over the surface, with the result that small domain areas can map to large regions on the surface.

Maillot et al.¹⁹ base their metric on mesh edge springs of nonzero rest length, where rest length corresponds to edge length on the surface. Lévy and Mallet¹⁸ use a metric that combines orthogonality and isoparametric terms.

Sander et al.²³ focus on making textures as small as possible for an unknown surface signal. Their stretch metric minimizes undersampling by integrating the sum of squared singular values over the map. Intuitively, this measures how distances in the domain get stretched when mapped onto the surface. We refer to this metric as *geometric-stretch*, and review it in more detail in Section 3.1, as it contributes essential elements to our signal-specialized metric.

Another contribution of Sander et al.²³ is to construct a single parametrization over all meshes in a progressive mesh sequence. In this paper we parametrize only a single mesh, focusing instead on signal specialization. Parametrizing a single mesh allows us to optimize the chart boundaries to arbitrary shapes, resulting in more optimal parametrizations. Chart boundary linearity constraints could be re-incorporated to allow parametrization of PM's.

Signal-specialized chart parametrization. There has been relatively little work in exploiting knowledge of the surface signal in optimizing the parametrization.

Given an existing parametrization, Sloan et al.²⁴ warp the square texture domain onto itself to more evenly distribute a scalar importance field. Unlike importance, our metric is derived directly from signal approximation error, and is integrated over the surface. Our IMT captures signal directionality, which allows the parametrization to squash in the direction across the signal gradient. We do not restrict the chart boundary to be a square.

Terzopoulos and Vasilescu²⁵ approximate a 2D image using a warped grid of sample values. The warping is achieved using a dynamic simulation where grid edge weights are set according to local image content. We consider signals mapped onto surfaces in 3D, define the parametrization on a coarser, irregular mesh, and store the signal in a texture image mapped onto this mesh.

Hunter and Cohen¹⁴ compress an image as a set of texture-mapped rectangles, obtained by k-d tree subdivision of the image based on frequency content. In contrast, our parametrization is continuous, considers a signal defined on a surface in 3D, and better adapts to high frequencies along diagonal directions.

Multi-chart parametrization. To avoid excessive distortion, an arbitrary mesh is generally parametrized using multiple charts. In the limit, distortion can be driven to zero by making each triangle its own chart. However, partitioning the surface into many charts has drawbacks. It constrains mesh simplification, requires more inter-chart gutter space, leads to mipmap artifacts, and fails to exploit continuity across charts. Thus a balance must be made between parametrization distortion and the drawbacks of charts.

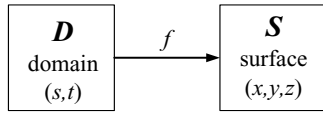
Several approaches^{5,8,16,19,23} directly partition the original mesh, while other schemes^{2,17,22} define the charts using mesh simplification.

For an interactive 3D painting system, Igarashi and Cosgrove¹⁵ construct charts based on the history of drawing operations. The texture resolution on a surface region is selected using the viewing resolution at the time the region was last painted. Our parametrization automatically adapts to the content of the signal, and scales the charts based on this content.

The main focus of this paper is the parametrization of a single chart using a signal-specialized metric. To process arbitrary meshes, we manually delineate the surface charts as in Krishnamurthy and Levoy¹⁶, and pack their parametrizations as in Sander et al.²³, but our method is applicable to any “chartification” scheme.

3. Parametrization metric

3.1 Review of geometric-stretch metric



To find a good parametrization f of a given surface \mathcal{S} onto a texture domain \mathcal{D} , Sander et al.²³ introduce the geometric-stretch metric $E_f(\mathcal{S})$, which is derived as follows.

The Jacobian of the function f is

$$J_f(s, t) = \begin{bmatrix} \partial f / \partial s(s, t) & \partial f / \partial t(s, t) \end{bmatrix} = \begin{bmatrix} f_s(s, t) & f_t(s, t) \end{bmatrix}.$$

The singular values Γ and γ of this 3×2 Jacobian matrix represent the largest and smallest length obtained when mapping unit-length vectors from the texture domain \mathcal{D} to the surface \mathcal{S} , i.e. the largest and smallest local “stretch”.

They are obtained as

$$\Gamma(s, t) = \sqrt{\frac{1}{2} \left((a_f + c_f) + \sqrt{(a_f - c_f)^2 + 4b_f^2} \right)} \quad \text{max sing. value}$$

$$\gamma(s, t) = \sqrt{\frac{1}{2} \left((a_f + c_f) - \sqrt{(a_f - c_f)^2 + 4b_f^2} \right)} \quad \text{min sing. value}$$

where

$$\begin{bmatrix} a_f(s, t) & b_f(s, t) \\ b_f(s, t) & c_f(s, t) \end{bmatrix} = \begin{bmatrix} f_s \bullet f_s & f_s \bullet f_t \\ f_s \bullet f_t & f_t \bullet f_t \end{bmatrix} = J_f^T J_f = M_f(s, t)$$

is called the *metric tensor* of the function f at (s, t) . The concept of metric tensor will become important in the derivation of our new signal-stretch metric.

From the singular values Γ and γ two norms corresponding to average and worst-case local stretch are defined as

$$L_2(s, t) = \sqrt{\frac{1}{2}(\Gamma^2 + \gamma^2)} = \sqrt{\frac{1}{2}(a_f + c_f)} \quad \text{and} \quad L_\infty(s, t) = \Gamma.$$

The L_2 stretch norm can also be expressed using the *trace* of the metric tensor as $L_2(s, t) = \sqrt{\frac{1}{2} \text{tr}(M_f)}$.

The squared L_2 stretch norm is integrated over the surface \mathcal{S} to obtain the geometric-stretch metric

$$E_f(\mathcal{S}) = \iint_{(s, t) \in \mathcal{D}} (L_2(s, t))^2 dA_s(s, t),$$

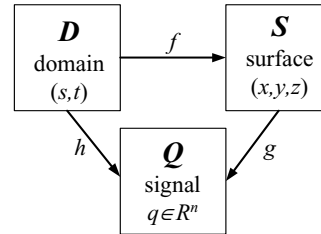
where $dA_s(s, t) = \sqrt{|M_f(s, t)|} ds dt$ is differential surface area.

In our setting where the surface \mathcal{S} is a triangle mesh, f is piecewise linear and thus its Jacobian J_f is constant over each triangle. Therefore the integrated metric can be rewritten as a finite sum:

$$E_f(\mathcal{S}) = \sum_{\Delta_i \in \mathcal{D}} \left(\frac{1}{2} \text{tr}(M_f(s_i, t_i)) A_s(\Delta_i) \right),$$

where $A_s(\Delta_i)$ is the surface area of triangle Δ_i , and $M_f(s_i, t_i)$ is the (constant) value of the metric tensor at any point $(s_i, t_i) \in \Delta_i$.

3.2 Signal-stretch metric



Unlike the geometric-stretch metric, our new metric considers a signal defined on the surface. Let this surface signal be denoted by the function $g : \mathcal{S} \rightarrow \mathcal{Q}$ where the signal-space \mathcal{Q} can be vector-valued (e.g. *RGB* color is a 3-vector in \mathcal{Q}).

To find a good surface parametrization f , we examine how well the function $h = g \circ f$ (from the texture domain \mathbf{D} to the signal \mathbf{Q}) is approximated when reconstructed from a discrete sampling over \mathbf{D} . For example, if the signal \mathbf{Q} varies greatly on a region of the surface \mathbf{S} , and the region is not allocated adequate space in the texture domain \mathbf{D} , then the local texture resolution on that region may be insufficient to accurately represent the signal.

In the appendix, we derive a metric for signal approximation error, $E_h(s, t)$, defined as the difference between h and its reconstruction \tilde{h} from a discrete sampling with spacing δ in \mathbf{D} . Our derivation makes two assumptions: (1) \tilde{h} is a piecewise constant reconstruction, and (2) the sampling is asymptotically dense. Under these assumptions, the squared signal approximation error at a point is

$$E_h(s, t) = \left(\frac{\delta^2}{3} \right) \text{tr}(M_h(s, t)),$$

$$\text{where } M_h(s, t) = J_h^T J_h = \begin{bmatrix} h_s \bullet h_s & h_s \bullet h_t \\ h_s \bullet h_t & h_t \bullet h_t \end{bmatrix}$$

is the metric tensor of the signal function h . (Here h_s and h_t are n -vectors where n is the dimension of the signal space \mathbf{Q} .)

The integrated squared error over the surface \mathbf{S} is therefore

$$E_h(\mathbf{S}) = \left(\frac{\delta^2}{3} \right) \text{tr}(\mathcal{M}_h(\mathbf{S})),$$

$$\text{where } \mathcal{M}_h(\mathbf{S}) = \iint_{(s,t) \in \mathbf{D}} M_h(s, t) dA_S(s, t)$$

is the *integrated metric tensor* (IMT) of the signal function h . Reducing this integral to a sum over domain triangles, we obtain

$$\mathcal{M}_h(\mathbf{S}) = \sum_{\Delta_i \in \mathbf{D}} \mathcal{M}_h(\Delta_i),$$

$$\text{where } \mathcal{M}_h(\Delta_i) = \iint_{(s,t) \in \Delta_i} M_h(s, t) dA_S(s, t) .$$

Our signal-specialized metric $E_h(\mathbf{S})$ can be seen as analogous to the geometric stretch from Section 3.1 (neglecting the globally constant factor $\delta^2/3$), but using the metric tensor of the signal mapping h rather than the surface mapping f . This is the reason we refer to it as the *signal-stretch* metric. In fact, if the signal reproduces the geometry (i.e. g is the identity map), then signal-stretch is equivalent to geometric stretch.

3.3 Computing the IMT

To compute the integrated metric tensor $\mathcal{M}_h(\Delta_i)$ of the signal on each triangle, we consider two cases.

First, the signal may be a **piecewise linear interpolant** of per-vertex attributes, as for example an *RGB* color specified at each vertex. In this case, the metric tensor $M_h(s, t)$ is constant over the triangle, just like the geometric tensor $M_f(s, t)$. The IMT is therefore the simple product

$$\mathcal{M}_h(\Delta_i) = M_h(s_i, t_i) A_S(\Delta_i) .$$

For completeness, we derive the Jacobian J_h for a triangle whose vertices have parametrizations p_1, p_2, p_3 in \mathbf{D} and signal q_1, q_2, q_3 in \mathbf{Q} . It is obtained by solving the linear system

$$(q_1 \quad q_2 \quad q_3) = (J_h \quad o) \begin{pmatrix} p_1 & p_2 & p_3 \\ 1 & 1 & 1 \end{pmatrix}$$

where $o \in \mathbf{Q}$ completes the affine transform matrix. Then, the metric tensor is $M_h = J_h^T J_h$ as previously defined.

The second case is that of a more **general signal** which has more detail than can be described at vertices. One example is a detailed image projected onto the triangle mesh. In that case, we compute $\mathcal{M}_h(\Delta_i)$ using numerical integration. Specifically, we apply a number of regular 1-to-4 subdivisions to the triangle, evaluate the signal at all the introduced vertices, and then sum the metric tensors of the resulting piecewise linear interpolant, just as described above. In our examples, we subdivided each mesh triangle into 64 sub-triangles to compute the IMT.

3.4 Affine transformation rule for the IMT

Within our optimization algorithm (Section 4), we repeatedly modify the parametrizations (s, t) of the mesh vertices, and examine the change in the signal-stretch functional $E_h(\mathbf{S})$. The straightforward implementation would be to recompute the integrated metric tensors of the mesh triangles based on the modified parametrization. This is prohibitive for two reasons:

(1) For the case of a general (nonlinear) signal on the mesh, the IMT is computed using expensive numerical integration.

(2) In our hierarchical solver (Section 4), coarse faces are used to represent regions of the original surface. Even if the signal is linear on the fine mesh faces, it becomes nonlinear on coarse mesh faces, and thus expensive to compute.

Fortunately, modifying the parametrization results in an affine transform of each mesh triangle, so we can *exactly* compute the IMT of a transformed triangle from its original IMT using a simple rule. (Although the domain Δ_i of the triangle changes, the integral $\mathcal{M}_h(\Delta_i)$ is still integrating the same region of the surface, due to the differential area term dA_S .)

Let $e: \mathbf{D} \rightarrow \mathbf{D}: (s', t') \rightarrow (s, t)$ be the local affine transform from the new triangle parametrization to the old, resulting in the new maps $f' = f \circ e$ and $h' = h \circ e$. Using the derivative chain rule, the new Jacobian is $J_{h'}(s', t') = J_h(s, t) J_e(s', t')$ where J_e is the Jacobian of the map e . Therefore, the new metric tensor is

$$M_{h'}(s', t') = J_e^T J_h J_e = J_e^T J_h^T J_h J_e$$

$$= J_e^T(s', t') M_h(s, t) J_e(s', t')$$

If e maps triangle vertices from p_1', p_2', p_3' to p_1, p_2, p_3 in \mathbf{D} , its Jacobian J_e can be obtained by solving the linear system

$$(p_1 \ p_2 \ p_3) = (J_e \ o) \begin{pmatrix} p_1' & p_2' & p_3' \\ 1 & 1 & 1 \end{pmatrix}$$

where $o \in \mathbf{D}$ completes the affine transform matrix.

For the IMT pre-computation in Section 3.3, we always store $\mathcal{M}_h(\Delta_i)$ with respect to a canonical parametrization f of the triangle Δ_i , e.g. one that maps the triangle onto a right-isosceles triangle in \mathbf{D} . Then, the affinely transformed IMT is

$$\mathcal{M}_h(\Delta_i) = J_e^T \mathcal{M}_h(\Delta_i) J_e \quad (1)$$

since the constant multiplication by matrices J_e^T and J_e is a linear operator that can be factored out of the integration, and the integration over the triangle area in 3D is unaffected by the transform.

To summarize, we pre-compute the integrated metric tensors on the original mesh faces with respect to canonical face parametrizations. During optimization, we apply the affine transform rule to quickly evaluate the modified signal-stretch metric.

4. Chart parametrization algorithm

Our optimization algorithm minimizes the nonlinear signal stretch $E_h(\mathbf{S})$ over the parametrizations (s,t) of the mesh vertices, while maintaining an embedding.

The basic strategy is similar to that of Sander et al.²³. After obtaining some initial chart parametrization, we minimize the nonlinear metric by repeatedly updating individual vertex (s,t) coordinates using line searches in the domain. Of course, other nonlinear optimization algorithms could be substituted, including those exploiting analytic derivatives. To prevent parametric folding, we only consider perturbing the vertex within the *kernel* of the polygon formed by its neighboring vertices. (The kernel of a polygon is the intersection of the interior half-planes defined by its boundary edges.)

For our nonlinear metric $E_h(\mathbf{S})$, we first compute IMTs on each triangle as discussed in Section 3.3. Perturbing a vertex during optimization induces an affine transform on each of its adjacent faces. We minimize the sum of the IMTs on these affinely transformed triangles using the formula of Section 3.4.

However, optimizing the chart parametrization using a uni-resolution algorithm has slow convergence, and often converges to bad local minima, particularly for our signal-stretch metric.

We improve both the speed and result of optimization using a novel multiresolution optimization algorithm. A hierarchy is established over the chart using a progressive mesh (PM) representation¹⁰. This PM is only used to accelerate the parametrization of the original (fine) mesh; the original mesh is not modified. The PM is constructed by simplifying the chart mesh using a sequence of half-edge collapses, with

a quadric error metric that seeks to preserve the surface signal¹¹.

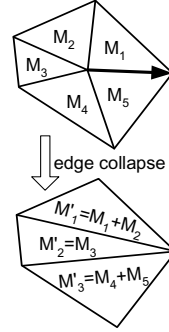
As described in the next sections, we use this multiresolution PM sequence to (1) propagate the signal IMT fine-to-coarse from the original mesh to all coarser meshes, and (2) apply a coarse-to-fine parametrization algorithm that uses these IMTs. In Section 4.4 we describe how we bootstrap this optimization process.

4.1 Fine-to-coarse metric propagation

For our hierarchical optimization technique, we need to redistribute the IMTs defined on triangles of the fine mesh to the triangles of the coarser meshes in the PM sequence. This redistribution must necessarily be inexact, because the triangles in the meshes at different resolutions lack any nesting property on the surface.

Transferring IMTs between faces requires expressing them with respect to a common coordinate system. We use the current parametrization to do this. We affinely transform the IMTs in the fine mesh triangles from their canonical frames to their shapes in the current parametrization.

Then, for each half-edge collapse in the PM sequence, we redistribute the IMTs using the simple scheme illustrated on the right. This heuristic scheme has the property that the sum of IMTs over mesh triangles is maintained at all levels of detail. Also, the redistribution weights are independent of the current parametrization. We experimented with more complex redistribution weights based on parametric overlap areas. However, these other schemes were worse, because the parametrization can initially be poor (i.e. contain highly stretched triangles).



4.2 Coarse-to-fine parametrization

Our coarse-to-fine algorithm has the following structure. We first create an initial embedding for the few faces in the PM base mesh using a brute-force optimization as in Sander et al.²³, but using the IMTs propagated from the fine mesh. Then, for each vertex split refinement operation in the PM sequence, we place the newly added vertex at the centroid of the kernel of its neighborhood polygon, and proceed to optimize it and its neighbors using IMTs:

```
// Parametrize the newly added vertex and its neighbors.
procedure optimize_vertex_split(Vertex vnew)
  // obtain initial (s,t) for neighborhood to be an embedding
  vnew.st := centroid(kernel(Neighborhood(vnew)))
  optimize_vertex_parametrization(vnew)
  repeat vertex_niter times
    for (v ∈ Neighbors(vnew))
      optimize_vertex_parametrization(v)
      optimize_vertex_parametrization(vnew)
```

Specifically, `optimize_vertex_parametrization(v)` minimizes $\sum_f tr(\mathcal{M}_h(\Delta_f))$ for faces f adjacent to v (Equation 1).

Unlike geometric stretch, the signal-stretch metric can have zero gradient since the signal may be locally constant on a region of the surface. Therefore, as a regularizing term we add a tiny fraction of geometric stretch to the energy functional minimized. This prevents the formation of degenerate triangles, and ensures that new vertices find non-degenerate neighborhood kernels.

4.3 Iterated multigrid strategy

The coarse-to-fine (CTF) optimization creates a new parametrization of the fine mesh. The new parametrization modifies the transformed IMTs on the fine mesh triangles. These transformed IMTs can be propagated fine-to-coarse (FTC), to be used in another iteration of CTF optimization.

This process bears some resemblance to the V-cycle commonly used in multigrid optimization¹, but applied here to irregular, non-nested grids. In classical multigrid, the coarse-level operation is uniquely defined using a restriction operation. In our setting, the mapping g is given at the fine resolution only. Thus, the energy on some coarse mesh can only be defined given some pointwise mapping between the fine and coarse meshes. We obtain this mapping implicitly by solving for a parametrization at the finest level and using this parametrization to propagate IMTs fine to coarse. Thus, the current fine-level solution is actually used to *define* the coarse-level problem.

To bootstrap this iterative optimization process, we require an initial parametrization to transform the IMTs on the finest mesh. We obtain this initial parametrization using a CTF optimization with the geometric stretch metric²³. Since we don't yet have IMTs, the CTF optimization refers to the geometry of the coarse meshes (x,y,z at each vertex). This CTF strategy was also used by Hormann et al.¹³ to accelerate geometric surface parametrization. The intuition is that a simplified mesh forms a good geometric approximation, and therefore its parametrization is a good starting state for parametrizing a finer mesh.

The high-level algorithm can be summarized as:

```

procedure optimize_chart_parametrization
  Pre-compute canonical IMTs on fine mesh faces.
  Construct progressive mesh of chart.
  // Initialize the parametrization:
  CTF optimize geometric metric without IMTs.
  // iteratively optimize using signal stretch:
  repeat ftc_ctf_niter times
    Transform fine mesh IMTs using current param.
    FTC propagate IMTs to all PM meshes. // (§4.1)
    CTF optimize signal stretch using IMTs. // (§4.2)

```

A single iteration (*ftc_ctf_niter*=1) is sufficient for many models. Further iterations (*ftc_ctf_niter*=3) yield significant additional improvement for some models, such as horse and gargoyle in Figure 5. Our multigrid strategy is significantly faster than brute-force, single-resolution optimization, as shown in Table 1.

Optimization method	Signal stretch $E_h(S)$	Signal error SAE	Timing (secs)
Floater + brute-force	552.0	38.7	7265
CTF (vertex signal)	90.0	33.4	23
CTF (vertex geometry)	89.1	35.9	21
+1x FTC-CTF IMTs	34.6	18.3	43
+2x FTC-CTF IMTs	32.9	17.7	65
+3x FTC-CTF IMTs	31.2	17.0	88
+ bound.-rect. FTC-CTF	28.7	15.6	118

Table 1: Comparison of parametrization methods on the model in Figure 1. Signal approximation error is measured using 128x128 textures. The final row includes the boundary-rectangle optimization of Section 4.5, and is shown in Figure 1b.

Sidenote. Earlier in the project, we explored using CTF optimization directly on the per-vertex signal instead of IMTs. However, the problem is that the surface signal varies too much. Unlike the geometric signal, it is not well approximated on a coarser mesh. As an example, a color-map signal usually zigzags across the unit *RGB* cube many times as one traverses the surface. Thus, an optimized coarse mesh often fails to adequately “reserve” space in the parametric domain for signal detail present in the finer meshes. The IMTs and their FTC propagation provide this lookahead capability, as shown by the results in Table 1.

4.4 Chart boundary optimization

To improve the parametrization quality, we allow chart boundary vertices to move in the texture domain, at all levels of the coarse-to-fine optimization algorithm. For this to work, we must overcome two problems.

First, the geometric-stretch and signal-stretch metrics are not scale-invariant. These functionals go to zero as the chart becomes infinitely large. We achieve scale-invariance by multiplying the functionals by total chart area. We find that this is preferable to multiplying per-triangle stretch by per-triangle area because it is computationally more stable.

Second, it is possible for the optimized chart boundary to self-intersect. To prevent this, when optimizing a chart boundary vertex we test for intersections between the two adjacent boundary edges and the remaining boundary edges. Since there are typically \sqrt{m} boundary elements for a chart of m vertices, this brute-force testing is fast.

One limitation of allowing the chart boundary to take on an arbitrary shape in \mathbf{D} is that it imposes constraints on subsequent mesh simplification. More vertices need to be retained on the simplified mesh to represent the boundaries, because their irregular parametric shapes are difficult to approximate with coarse polygons. In this paper our approach is to simplify the mesh prior to parametrizing it.

4.5 Growth to bounding rectangle

For a single chart, we embed its parametrization into a square texture image. For multi-chart meshes, we find the tightest bounding rectangle around each chart, and pack these rectangles within the texture as in Sander et al.²³. In either case, some texture regions within the bounding square or rectangle are left unused. To reduce these wasted regions, we encourage the chart to grow into the unused space. We achieve this using an additional FTC-CTF iteration where we remove the chart area penalty but constrain the chart boundary to remain within the original bounding rectangle. Examples are shown in Figure 5.

4.6 Relative chart scaling

When optimizing multi-chart meshes, we apply a separate, isotropic scale to each chart to minimize error over the entire mesh. Given N charts with domain areas a_i and error metrics E_i , we determine for each chart an area scale factor α_i (i.e., a 1D scale by $\sqrt{\alpha_i}$ in both s and t). Isotropically scaling a chart by α_i creates the new error $E'_i = E_i/\alpha_i$; for example, a chart 4 times bigger has $1/4$ the squared signal stretch. Therefore we seek to find

$$\operatorname{argmin}_{\alpha_1, \dots, \alpha_N} \left(\sum_{i=1}^N E_i / \alpha_i \right) \text{ such that } \sum_{i=1}^N \alpha_i a_i = 1$$

which minimizes summed error after the scaling, subject to the constraint that the total rescaled domain area be held constant.

The optimal chart area scalings can be derived in closed form using the method of Lagrange multipliers as

$$\alpha_i = \sqrt{E_i/a_i} / \sum_{k=1}^N \sqrt{E_k a_k}.$$

5. Results

We have created signal-specialized parametrizations for a number of models, as summarized in Table 2 and shown in Figure 4 and Figure 5. All models originated from 3D scanning. The signals on the parasaur and horse were created by projecting hand-painted images onto the surfaces. The signals on the gargoyle and cat are normal maps. The signal on the face is scanned color data.

The motivation for texture mapping is to represent detail at a finer level than can be represented at the mesh vertices. In our examples, we obtain this detail information either from a more detailed geometric model (e.g. normal map), or from an externally defined signal (e.g. painted color image). In the first case, we transfer the signal from the more detailed mesh by ray-shooting along the interpolated surface normal²². In the second case, we projected a high-resolution image onto the mesh.

For the gargoyle and horse models in Figure 5, we manually partitioned the mesh into 6 and 5 charts respectively.

	Model	#Vertices	#Charts	Timing (secs)
Figure 1	parasaur	3,870	1	49
Figure 5	gargoyle	2,500	6	68
Figure 5	horse	5,000	5	117
Figure 5	face	1,315	1	12
Figure 5	parasaur	3,870	1	49
Figures 3,4	cat	1,000	1	10
Figure 3	cat	53,197	1	556

Table 2: Model sizes and parametrization timings.

Table 2 shows that our parametrization scheme takes a few minutes per model. These are reasonable execution times for automatic preprocessing of a large suite of models.

Figure 4 qualitatively compares three different parametrizations of the cat surface. The Floater result is representative of parametrizations that ignore absolute surface stretch (e.g. also harmonic map, conformal map, MIPS). The geometric-stretch parametrization provides the most even distribution of texture samples over the surface, as is evident in the rightmost column. Our new signal-stretch parametrization adapts the sampling density to local signal detail. Note how the sharp signal transitions near creases are allotted more space in the texture domain. Thus, the reconstructed signal is significantly better. For Figure 4, we omitted the chart boundary growth of Section 4.5, to make a fair comparison with Floater, and to show the natural shapes that the charts adopt using our algorithm. Considering that 43% of the 64x64 texture samples are thus unused, the rendering quality is surprisingly good.

To quantify parametrization quality, we measure signal approximation error (SAE) as rms difference on a dense set of surface points, distributed uniformly according to surface area. For each point, we compute the difference between the original signal and the bilinear interpolation of the four adjacent texture samples. For vector-valued signals, we use the L_2 norm. Table 1 confirms a good correlation between our signal-stretch norm and SAE.

Figure 3 graphs this signal error as a function of the number of texture samples, compared for three parametrizations. The graphs show a notable reduction in error from the geometric stretch to the signal-specialized metric. In particular, a given approximation error can be obtained with a factor ~ 3 – 5 savings in texture size. In the top graph, the signal-stretch deteriorates at higher texture sizes because the curves uses the same 1,000-vertex parametrization for all texture resolutions. For high-resolution textures, it is beneficial to increase the parametrization complexity. By comparison, the bottom graph shows the result of using the detailed 53,197-vertex mesh to parametrize the signal. The improvement is then more uniform along the whole range of texture resolutions.

Most of our examples show the improvement in appearance quality for a given texture size. Figure 2 instead compares the texture size required for a given quality level.

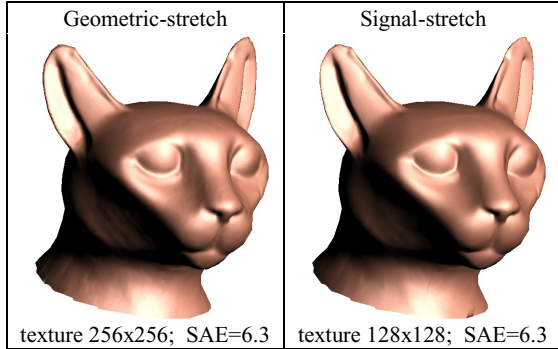


Figure 2: These nearly identical renderings demonstrate that our parametrization reduces texture size by a factor of 4 for the same signal approximation error.

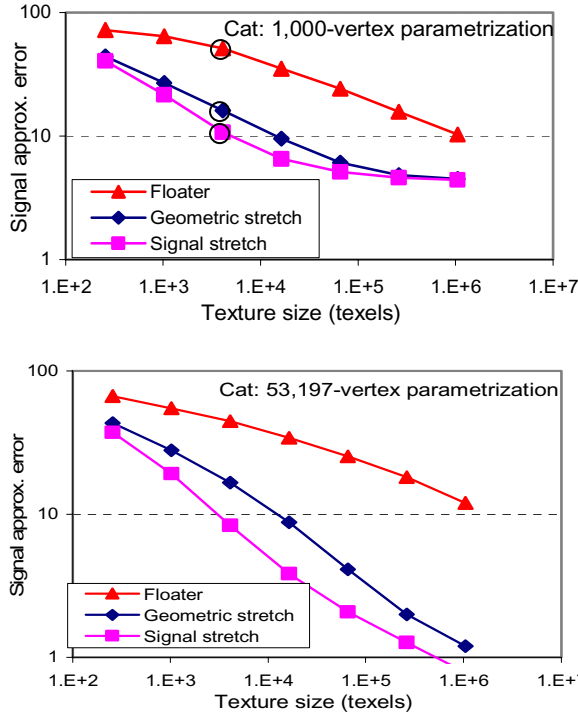


Figure 3: Signal approximation error (SAE) as a function of texture size for three parametrizations, using a 1000-vertex mesh (top), and using a 53,197-vertex mesh (bottom). The circled points indicate the 64x64 textures rendered in Figure 4.

6. Summary and future work

To reduce the size of texture maps for real-time rendering, we introduced the signal-stretch parametrization metric derived from a Taylor expansion of signal approximation error. To improve the efficiency and quality of the optimization, we developed a multiresolution algorithm that accumulates the fine signal variation onto the faces of coarser meshes, providing “lookahead” during coarse-to-fine optimization. By integrating a metric tensor, we capture signal directionality and therefore locally squash the parametrization perpendicularly to signal gradient. Our new optimization algorithm also accelerates the minimization of other parametrization metrics (both linear and nonlinear), such as geometric stretch.

Signal-specialized parametrizations allocate more texture samples to mesh regions with greater signal variation. Texture resolution can be selected based on the desired signal reconstruction accuracy. Often, a factor 4 reduction in texture space is possible, thus allowing more textured objects to be rendered in the same scene.

Specializing a parametrization to a signal offers the greatest benefit on surfaces with inhomogeneous signals, i.e. non-uniform detail. For homogeneous signals (e.g. the high-frequency grass texture in Figure 5), the signal-stretch metric effectively behaves like geometric-stretch because the averaged metric tensor (IMT/area) tends towards the same scaled identity matrix everywhere on the surface. In this case, geometric stretch is the best one can hope for.

Note that our metric applies to signals of arbitrary dimensionality. In particular, it is possible to specialize a parametrization to a combination of signals, such as normals and colors, and even the geometric signal itself.

There are several areas for future work. We are exploring better signal-specialized metrics derived from the Taylor expansion of a locally linear reconstruction, instead of the current constant-reconstruction assumption. Our signal-stretch metric $E_h(\mathcal{S})$ is proportional to δ^2 , so its minimum is independent of texture resolution. If the desired texture resolution is known a priori, a better parametrization might be attainable by specializing to that resolution. Also, better efficiencies might be achievable by adapting mesh chartification to the surface signal. Finally, rather than simply measuring signal-approximation error, we could propagate this error through the rendering process, and even consider perceptual measures.

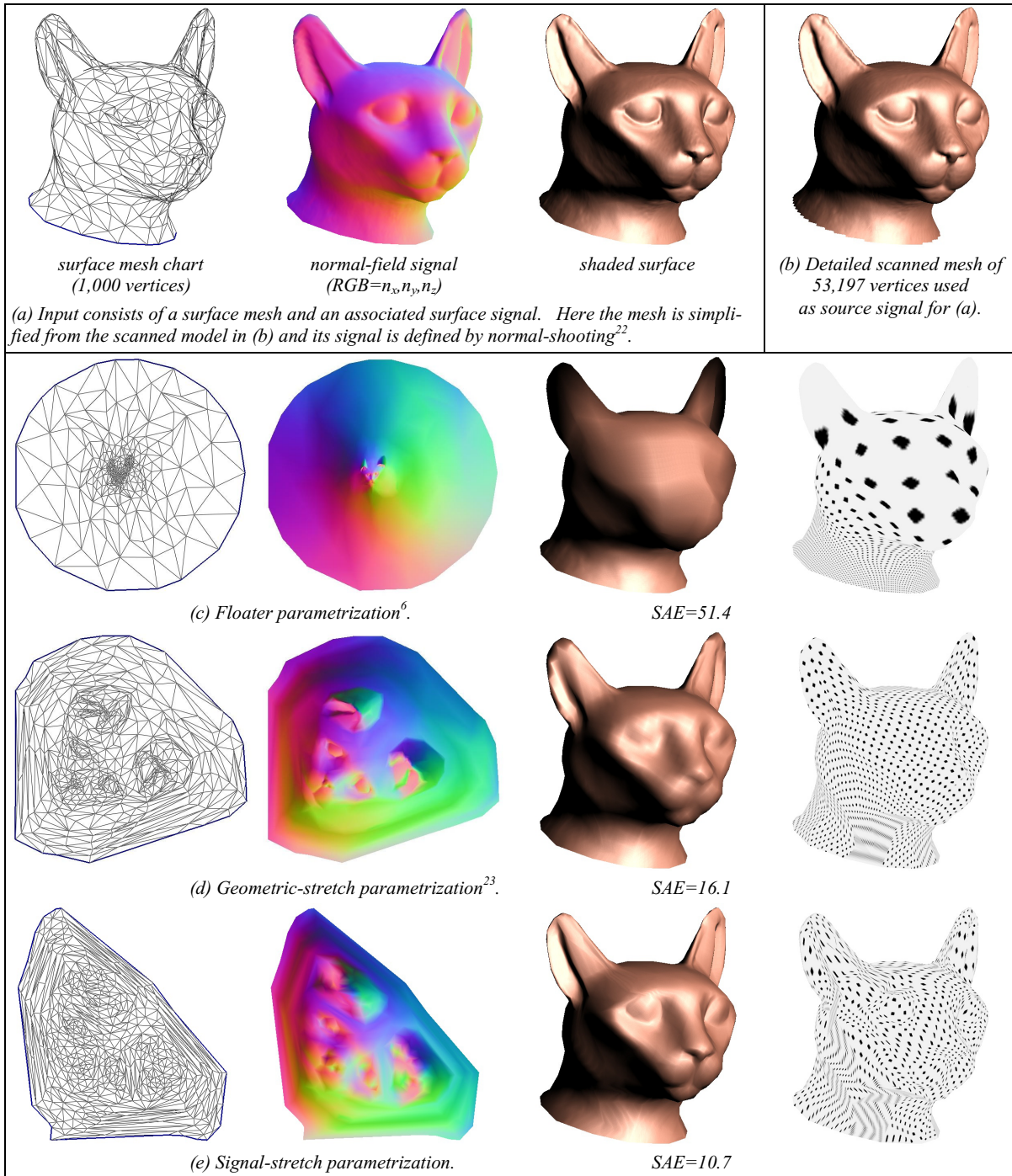


Figure 4: Comparison of parametrization schemes for a single chart, where the surface signal is a normal map. In (c-e), columns show: (1) parametrization of chart in texture domain, (2) normal-map signal transferred to texture domain, (3) shaded surface using normal-map reconstructed from texture with 64×64 samples, and (4) visualization of mapping a regular 64×64 texture grid pattern onto the surface. Signal approximation error (SAE) is the rms L_2 difference between the original color signal and its reconstruction over the surface. For this 8-bit/channel normal-map, SAE thus ranges from 0 to $255\sqrt{3}$.

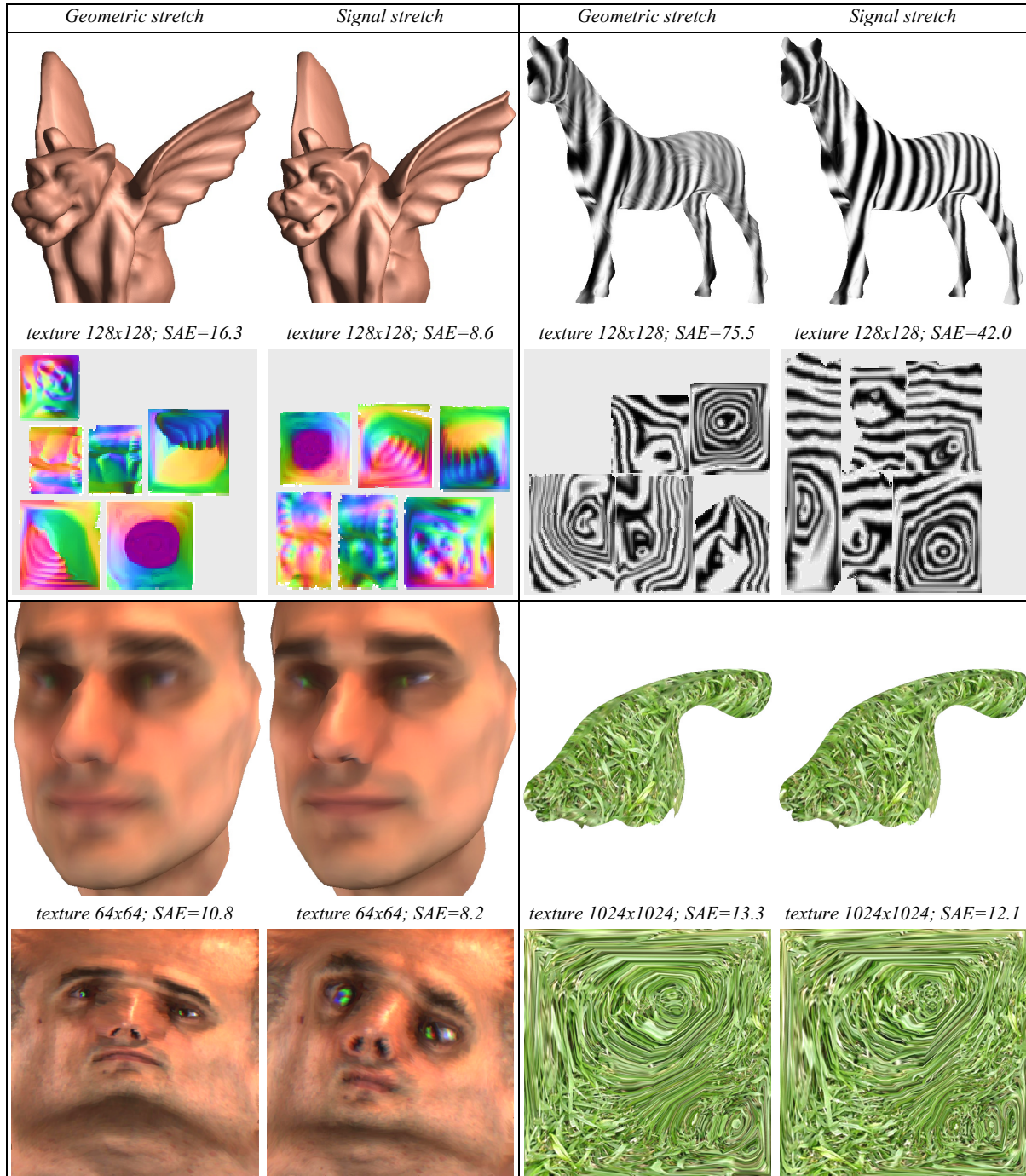


Figure 5: Results comparing traditional geometry-based surface parametrization with our new signal-stretch parametrization. For textures of the same size, the signal-stretch parametrization significantly reduces the signal approximation error (SAE) over the surface. The upper left example uses normal maps whereas the other examples use color maps. The lower right example demonstrates that for high-frequency signals, signal-stretch behaves much like geometric-stretch.

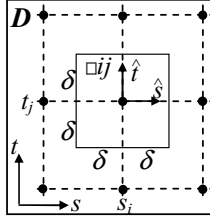
7. References

1. Briggs, W. A multigrid tutorial, SIAM, Philadelphia, 1987.
2. Cignoni, P., Montani, C., Rocchini, C., and Scopigno, R. A general method for recovering attribute values on simplified meshes. *IEEE Visualization 1998*, pp. 59-66.
3. Cohen, J., Olano, M., and Manocha, D. Appearance-preserving simplification. *SIGGRAPH 1998*, pp. 115-122.
4. Duchamp, T., Certain, A., DeRose, A., and Stuetzle, W. Hierarchical computation of PL harmonic embeddings. Technical report, University of Washington, 1997.
5. Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzle, W. Multiresolution analysis of arbitrary meshes. *SIGGRAPH 1995*, pp. 173-182.
6. Floater, M. Parametrization and smooth approximation of surface triangulations. *CAGD 14(3)*, pp. 231-250, 1997.
7. Garland, M., and Heckbert, P. Surface simplification using quadric error metrics. *SIGGRAPH 1997*, pp. 209-216.
8. Garland, M., Willmott, A., and Heckbert, P. Hierarchical face clustering on polygonal surfaces. *Symposium on Interactive 3D Graphics 2001*, pp. 49-58.
9. Haker, S., Angenent, S., Tannenbaum, S., Kikinis, R., Sapiro, G, and Halle, M. Conformal surface parameterization for texture mapping. *IEEE Trans. on Visual. and Comp. Graphics*, 6(2), 2000.
10. Hoppe, H. Progressive meshes. *SIGGRAPH 1996*, pp. 99-108.
11. Hoppe, H. New quadric error metric for simplifying meshes with appearance attributes. *IEEE Visualization 1999*, pp. 59-66.
12. Hormann, K., and Greiner, G. MIPS: An efficient global parametrization method. In *Curve and Surface Design: Saint-Malo 1999*, Vanderbilt University Press, pp. 153-162.
13. Hormann, K., Greiner, G., and Campagna, S. Hierarchical parametrization of triangulated surfaces. *Vision, Modeling, and Visualization 1999*, pp. 219-226.
14. Hunter, A., and Cohen, J. Uniform frequency images: adding geometry to images to produce space-efficient textures. *IEEE Visualization 2000*, pp. 243-250.
15. Igarashi, T., and Cosgrove, D. Adaptive unwrapping for interactive texture painting. *Symposium on Interactive 3D Graphics 2001*, pp. 209-216.
16. Krishnamurthy, V., and Levoy, M. Fitting smooth surfaces to dense polygon meshes. *SIGGRAPH 1996*, pp. 313-324.
17. Lee, A., Sweldens, W., Schröder, P., Cowsar, L., and Dobkin, D. MAPS: Multiresolution adaptive parametrization of surfaces. *SIGGRAPH 1998*, pp. 95-104.
18. Lévy, B., and Mallet, J.-L. Non-distorted texture mapping for sheared triangulated meshes. *SIGGRAPH 1998*, pp. 343-352.
19. Maillot, J., Yahia, H., and Verroust, A. Interactive texture mapping. *SIGGRAPH 1993*, pp. 27-34.
20. Malzbender, T., Gelb, D., and Wolters, H. Polynomial texture maps. *SIGGRAPH 2001*, pp. 519-528.
21. Max, N. Horizon mapping: Shadows for bump-mapped surfaces. *The Visual Computer*, July 1998, pp. 109-117.
22. Sander, P., Gu, X., Gortler, S., Hoppe, H., and Snyder, J. Silhouette clipping. *SIGGRAPH 2000*, pp. 327-334.
23. Sander, P., Snyder, J., Gortler, S., and Hoppe, H. Texture mapping progressive meshes. *SIGGRAPH 2001*, pp. 409-416.
24. Sloan, P.-P., Weinstein, D., and Brederson, J. Importance driven texture coordinate optimization. *Computer Graphics Forum (Eurographics '98) 17(3)*, pp. 97-104.
25. Terzopoulos, D., and Vasilescu, M. Sampling and reconstruction with adaptive meshes. *CVPR 1991*, pp. 70-75.

8. Appendix: Derivation of signal-stretch metric

As explained in Section 3.2, signal approximation error is the difference between the function h and its reconstruction \tilde{h} from a discrete sampling of the texture domain \mathbf{D} . In this appendix, we show that the squared pointwise error gives rise to the norm $E_h(s, t) = (\delta^2/3) \text{tr}(M_h(s, t))$ under the assumptions of piecewise constant reconstruction and asymptotically dense sampling.

We assume that the domain \mathbf{D} contains a regular grid of sample points (s_i, t_j) , spaced 2δ apart on each axis. As shown on the right, let $(\hat{s}, \hat{t}) \in [-\delta, +\delta] \times [-\delta, +\delta]$ be a local coordinate system within the grid square $\square ij$ about each sample, such that $(s, t) = (s_i + \hat{s}, t_j + \hat{t}) \in \square ij$.



Perhaps the simplest reconstruction function \tilde{h} in the neighborhood $\square ij$ of each sample is given by the constant function $\tilde{h}_{ij}(s, t) = h(s_i, t_j)$. With this reconstruction function, the pointwise squared error can be expressed as

$$E_h(s, t) = \|h(s, t) - \tilde{h}_{ij}(s, t)\|^2 = \|h(s_i + \hat{s}, t_j + \hat{t}) - h(s_i, t_j)\|^2$$

Using a Taylor expansion about (s_i, t_j) , E_h can be rewritten as

$$E_h(s, t) = E_{\square ij}(\hat{s}, \hat{t}) + O(\delta^3)$$

where the first term is defined via

$$\begin{aligned} E_{\square ij}(\hat{s}, \hat{t}) &= \left\| \begin{bmatrix} h_s(s_i, t_j) & h_t(s_i, t_j) \end{bmatrix} \begin{bmatrix} \hat{s} \\ \hat{t} \end{bmatrix} \right\|^2 = \left\| J_h(s_i, t_j) \begin{bmatrix} \hat{s} \\ \hat{t} \end{bmatrix} \right\|^2 \\ &= \begin{bmatrix} \hat{s} & \hat{t} \end{bmatrix} M_h(s_i, t_j) \begin{bmatrix} \hat{s} \\ \hat{t} \end{bmatrix} = \begin{bmatrix} \hat{s} & \hat{t} \end{bmatrix} \begin{bmatrix} a_h & b_h \\ b_h & c_h \end{bmatrix} \begin{bmatrix} \hat{s} \\ \hat{t} \end{bmatrix}. \end{aligned}$$

The average squared error over the grid square $\square ij$ is then

$$\begin{aligned} \bar{E}_{\square ij} &= \frac{1}{4\delta^2} \int_{-\delta}^{+\delta} \int_{-\delta}^{+\delta} (E_{\square ij}(\hat{s}, \hat{t}) + O(\delta^3)) d\hat{s} d\hat{t} \\ &= \delta^2 \left(\frac{1}{3} a_h + \frac{1}{3} c_h \right) + O(\delta^3). \end{aligned}$$

Normally, we are interested in the squared error integrated over surface area. The integral here ignores the variation of differential area $dA_s(\hat{s}, \hat{t})$ within the grid square $\square ij$, but that variation is insignificant for our asymptotic analysis ($\delta \rightarrow 0$).

The error converges to 0 at a rate $O(\delta^2)$. So, neglecting the higher-order terms $O(\delta^3)$, a measure of approximation error with piecewise constant reconstruction as $\delta \rightarrow 0$ is

$$E_h(s, t) = \left(\frac{\delta^2}{3} \right) \text{tr}(M_h(s, t)).$$

While this analysis assumes h is continuously differentiable over \mathbf{D} , it can also be applied heuristically to other functions such as piecewise linear ones.