Gradient-Domain Processing within a Texture Atlas

FABIÁN PRADA and MISHA KAZHDAN, Johns Hopkins University, USA
MING CHUANG, PerceptIn Inc., USA
HUGUES HOPPE, Google Inc., USA

1 INTRODUCTION

In computer graphics, detailed surface fields are commonly stored using a texture atlas parameterization. The approach is to partition a surface mesh into chart regions, flatten each chart onto a polygon, and pack these polygons into a rectangular atlas domain by recording texture coordinates at vertices. The resulting texture map efficiently captures high-resolution content and is natively supported by hardware rasterization. A key benefit is that texels lie on a regular image grid, enabling efficient random-access, strong memory coherence, and massive parallelism.

As reviewed in Section 2, most techniques for processing signals on surfaces involve sampling the signal over a dense triangle mesh and defining a discretization of the Laplace operator that adapts to the nonuniform structure and geometry of mesh neighborhoods. We instead explore a framework to perform gradient-domain processing directly in a texture atlas domain, thereby (1) eliminating the need for resampling and (2) mapping computation to a regular 2D grid.

**Goal.** Our gradient-domain processing objective is designed to solve a broad class of problems. Given a texture-atlased triangle mesh with metric \( h \), and given target texture values \( \psi \), target texture differential \( \omega \), and a screening weight \( \alpha \geq 0 \) balancing fidelity to \( \psi \) and \( \omega \), the goal is to find the texture values \( \phi \) that minimize the energy

\[
E(\phi; h, \alpha, \psi, \omega) = \alpha \cdot \| \phi - \psi \|^2_h + \| d\phi - \omega \|^2_h.
\]

Formulating this least squares minimization over the texel values in a texture atlas poses several challenges:

- Using standard bilinear interpolation, texture maps represent functions that do not (in general) align across chart boundaries. As a result, continuity can only be enforced by constraining the texture signal to have constant value along the seams.
- Evaluating the texture near chart boundaries requires the use of both interior and exterior texels. Because exterior texels are not associated with positions on the surface, defining discrete derivatives across chart boundaries is non-trivial.
- Although(texels lie on a uniform grid, their corresponding locations on the surface are distorted by the parameterization. The nonuniform metric must be taken into account.

ACM Trans. Graph., Vol. 37, No. 4, Article 154. Publication date: August 2018.
Approach. To address these challenges, we use an intermediate representation involving continuous basis functions that approximate the bilinear basis. Specifically, we introduce:

- A novel function space spanned by basis functions that reproduce the bilinear reconstruction kernel in the interior of a chart and are continuous across chart boundaries.
- A basis for cotangent vector fields to represent the target texture differential.
- Metric-aware Hodge stars for constructing the mass and stiffness matrices in the discretization of Equation (1) over texels.

In effect, we form a linear system over the texel values of an ordinary texture atlas, but using system matrix coefficients derived from an approximating continuous function space.

To efficiently solve this system, we present a novel multigrid algorithm that exploits grid regularity within chart interiors while correctly handling irregularity across chart boundaries.

Our work does not address seamless texturing. Because the output representation, like the input, is a general texture atlas evaluated using bilinear hardware rasterization, continuity can only be attained by blurring the signal along chart boundaries. However, we find that formulating signal processing operations using an intermediate continuous representation yields results in which chart seams are usually imperceptible. Our strategy is more effective than introducing inter-chart continuity constraints (Section 4.2).

We demonstrate the effectiveness of our approach in applications including signal smoothing and sharpening, texture stitching, geodesic distance computation, and line integral convolution.

2 RELATED WORK

We begin by reviewing foundational work in gradient-domain image processing. Then we survey extensions to the processing of signals on surfaces and the discretizations of the Laplace operator that enable this. Finally, we discuss works that address inter-chart continuity in texture atlases.

Gradient-domain image processing. The regularity of image grids makes it easy to define discrete differential operators for filtering. Applications include gradient-domain smoothing and sharpening [Bhat et al. 2008], dynamic range compression [Fattal et al. 2002; Weyrich et al. 2007], and image stitching [Agarwala et al. 2004; Levin et al. 2003; Pérez et al. 2003]. Although much of the work focuses on homogeneous filtering, seminal early work by Perona and Malik [1990] demonstrates the power of incorporating anisotropy.

Gradient-domain processing on surface meshes. Applications in geometry processing often take the surface signal to be the coordinates of the embedding. Taubin [1995] introduces surface smoothing using the combinatorial Laplacian. Desbrun et al. [1999] extend this approach to perform isotropic smoothing using the cotangent Laplacian. Later works incorporate anisotropy into the formulation to support edge-aware smoothing [Bajaj and Xu 2003; Chuang et al. 2016; Clarenz et al. 2000; Taslizen et al. 2002]. Laplacian-based geometry processing is also used for more general surface editing [Sorkine et al. 2004; Yu et al. 2004].

Discretizing the Laplace operator. Due to the irregular tessellation, gradient-domain processing on surface meshes requires a geometry-aware discretization of the Laplace operator. The cotangent Laplacian [Dziuk 1988; Pinkall and Polthier 1993] is the standard discretization for triangle meshes with linear elements, and extensions have been proposed for general polygon meshes [Alexa and Wardetzky 2011]. For parametric surfaces, the Jacobian of the parameterization is used to define a Laplace operator through pointwise evaluation [Stam 2003; Witkin and Kass 1991]. When the parameterization is conformal, the Jacobian corresponds to an isotropic scale, facilitating the definition of a pointwise operator [Lui et al. 2005]. For implicit surfaces, the restriction of the 3D Laplacian on the Cartesian grid is used to define the operator on the surface [Bertalmio et al. 2001; Chuang et al. 2009; Osher and Sethian 1988].

Texture atlas parameterization. A large body of work has focused on optimizing seam-placement and minimizing parametric distortion [Lévy et al. 2002; Poranne et al. 2017; Sander et al. 2002, 2001; Sheffer and Hart 2002; Zhang et al. 2005; Zhou et al. 2004]. In this work we assume that the parameterization is given and should not be changed.

Inter-chart continuity. Some earlier representations allow access to texel values across chart boundaries. These include indirection maps [Lefèvre and Hoppe 2006] which pad the chart boundaries with pointers to texels on the opposite side of the seam, and the Traveler’s Map [González and Patow 2009] which encodes the affine transformation taking a seam texel to the corresponding texel location on the opposite side of the seam. González and Patow achieve seamless rendering by zippering the seams within a pixel shader, in effect adding a thin fillet of triangles over which standard bilinear sampling is replaced with linear sampling. Our approach also introduces a triangulation to define a seamless function space. We use refinement rather than zippering, allowing us to represent the signal using all active texels (including those immediately outside the chart). Our intermediate triangulation is created to assist signal processing and does not redefine the rendering representation.

An alternative approach is to enforce inter-chart continuity by projecting the surface signal onto the space of seam-continuous functions [Liu et al. 2017]. This also supports texture evaluation using standard hardware sampling. However, the projection can give rise to visible smearing artifacts when the signal has a large gradient parallel to a chart boundary, as discussed in Section 4.2.

Specialized atlas constructions. Another approach to attain inter-chart continuity is to constrain the atlas to map clusters of mesh faces to axis-aligned rectangular charts in the texture domain with matching numbers of texels across chart boundaries [Carr and Hart 2002, 2004; Yuksel 2017]. Our work supports general texture atlases.

3 PRELIMINARIES

The input to our algorithm is an atlas parameterization of a 2-manifold immersed in 3D. It consists of a triangle mesh \((V, T)\) residing in the unit-square, an equivalence relation \(\sim\) on \(V\) indicating if two boundary vertices correspond to the same point on the manifold, and a map \(\Phi : V \to \mathbb{R}^3\) giving the immersion.
3.1 Texture atlas

The mesh atlas induces a partition of triangles into connected components, each defining a chart domain $M_i \subset [0,1] \times [0,1]$ formed by the union of its triangles. We let $M = \bigcup M_i$ denote the parameterization domain. We extend the map $\Phi : M \rightarrow \mathbb{R}^3$ to the map $\Phi : M \rightarrow \mathbb{R}^3$ by linear interpolation within triangles. We extend the equivalence relation $\sim$ to $M$ by linear interpolation along boundary edges, setting $p \sim q$ if there exists boundary edges $(v_1, v_2)$ and $(w_1, w_2)$ and interpolation weight $\alpha \in [0,1]$ such that $v_1 \sim w_1$, $v_2 \sim w_2$, $p = (1-\alpha)v_1 + \alpha v_2$, and $q = (1-\alpha)w_1 + \alpha w_2$.

We say that points $p, q \in M$ are on opposite sides of a seam if $p \not\sim q$ and that a function $\phi : M \rightarrow \mathbb{R}$ is seam-continuous if it is continuous on $M$ and has the same values on opposite sides of a seam.

Given a $W \times H$ texture image, we partition the unit square into $W \times H$ cells and compute the dual graph (shown in black in the inset). As our goal is to define a function space which mimics the bilinear functions, we define the footprint of a node to be the four incident quads (the support of the bilinear kernel centered at the node).

We define a texel to be any node whose footprint overlaps $M$ and denote the set of texels by $T$. We assume that the footprint intersects exactly one $M_i$ and say a texel is interior if its footprint is contained within a chart (green nodes) and boundary otherwise (red nodes).\(^1\)

3.2 Riemannian structure

To integrate functions over the triangulation, we require a Riemannian metric $h$ on $M$. This function associates to every point $p \in M$ a symmetric, positive-definite bilinear form on the tangent space, $h_p : T_p M \times T_p M \rightarrow \mathbb{R}$. We recall several facts about $h$:

- Given tangent vectors $v, w \in T_p M$, the inner product of the vectors is defined to be $h_p(v, w)$.
- Given cotangent vectors $v^*, w^* \in T^*_p M$, the inner product of the vectors is defined to be $h_p^{-1}(v^*, w^*)$.$^2$
- Given a function $\psi : M \rightarrow \mathbb{R}$, the integral of $\psi$ with respect to the metric $h$ is defined to be
  \[
  \int_M \psi \, dh = \int_M \sqrt{|\mu^{-1} \circ h|} \cdot \psi \, d\mu,
  \]
with $\mu$ the standard (2D) Euclidean metric.

In the context of gradient domain processing, the metric needs only be integrable. Therefore we restrict ourselves to the set of piecewise-constant metrics. That is, given the canonical coordinate frame on the unit square containing $M$, and given a triangle $t \in T$, we consider metrics for which the matrix expression of $h_p$ is the same for all $p \in t$.

Letting $\mu$ denote the standard (in this case, 3D) Euclidean metric, we define the immersion metric $g$ as:

\[
g_p(v, w) = \mu_{\phi(p)} \left( \partial \phi_p | v, \partial \phi_p | w \right), \quad \forall v, w \in T_p M.
\]

1Charts can always be translated by different integer offsets to ensure that the footprint of a texel intersects exactly one chart.

2Note that since a bilinear form on a vector space is equivalent to a linear map from the vector space to its dual, the inverse is well-defined when the bilinear form is definite.
Since the bilinear kernels are piecewise-quadratic polynomials, we define the \( \{ \phi_t \} \) to be piecewise-quadratic as well.

We proceed in three steps: (1) computing a new triangulation \( \widehat{T} \) of the texture domain; (2) using \( \widehat{T} \) to define a seam-continuous basis of piecewise-quadratic functions \( \{ \tilde{Q}_n \} \) on \( M \); (3) defining bilinear-like texel functions \( \{ \phi_t \} \) as linear combinations of the \( \{ \tilde{Q}_n \} \).

1) Triangulating the texture domain. We decompose the atlas domain \( M \) into a set of polygonal cells \( \mathcal{C} \) by tessellating \( M \) using the texel lattice (Figure 3). For each vertex introduced along a seam, we insert a corresponding vertex on the opposite side of the seam (shown as dashed lines in Figure 4). Then, we compute a constrained Delaunay triangulation \( \widehat{T} \) of these polygons.

2) Defining a quadratic seam-continuous function basis. We associate a quadratic Lagrange basis function to each vertex and each edge in the triangulation \( \widehat{T} \) [Heckbert 1993]. These functions form a partition of unity, reproduce continuous piecewise-quadratic polynomials, and are interpolatory, i.e., a function centered at a node evaluates to 1 at that node and to 0 at all other nodes. (The inset shows elements centered on a vertex and edge of a triangle mesh.) We denote the set of nodes (vertices and edges) by \( \mathcal{N} \) and the basis as \( \{ Q_n \}_{n \in \mathcal{N}} \).

To obtain a seam-continuous function-space, we merge the \( \{ Q_n \} \) across seams into a single function. Specifically, let \( \widehat{\mathcal{N}} = \mathcal{N}/\sim \) be the set of equivalence classes in \( \mathcal{N} \) modulo seam-equivalence. (We implicitly treat a node \( n \in \mathcal{N} \) as a point on \( M \), using the vertex position if \( n \) is a vertex and the midpoint if \( n \) is an edge.) We associate a seam-continuous function \( \tilde{Q}_n \) to each equivalence class \( \tilde{n} \in \widehat{\mathcal{N}} \) by summing the quadratic Lagrange elements associated to nodes in the equivalence class:

\[
\tilde{Q}_n = \sum_{n \in \tilde{n}} Q_n .
\]

These functions also form a partition of unity, reproduce seam-continuous piecewise quadratic polynomials, and are interpolatory.

3) Defining a bilinear-like seam-continuous basis of texel functions. Given a texel \( t \in \mathcal{T} \), we define the function \( \tilde{B}_t : M \to \mathbb{R} \) to be the linear combination of \( \{ \tilde{Q}_n \} \), with coefficients given by evaluating the bilinear function \( B_t \) at the node positions:

\[
\tilde{B}_t(p) = \sum_{\tilde{n} \in \widehat{\mathcal{N}}} \left( \sum_{n \in \tilde{n}} B_t(n) \right) \cdot \tilde{Q}_n(p) .
\]

By construction, the \( \{ \tilde{B}_t \} \) are seam-continuous since they are the linear combinations of seam-continuous functions. Furthermore, due to the interpolatory property of the Lagrange elements, the function \( \tilde{B}_t \) reproduces the bilinear function \( B_t \) whenever \( t \) is an interior texel (Figure 2a). Generally, the functions \( \tilde{B}_t \) and \( B_t \) agree on the intersection of \( M \) with the footprint of \( t \).3 Compare Figure 2b, second and third rows.) Please see Claim 1 in the appendix.

The limitation of using the functions \( \{ \tilde{B}_t \} \) is that they do not form a partition of unity. To address this, we normalize the coefficients by the number of seams on which the node is located:

\[
\phi_t(p) = \sum_{\tilde{n} \in \widehat{\mathcal{N}}} \left( \frac{1}{|\tilde{n}|} \sum_{n \in \tilde{n}} B_t(n) \right) \cdot \tilde{Q}_n(p) ,
\]

where \( |\tilde{n}| \) is the cardinality of the equivalence class \( \tilde{n} \). Please see Claim 3 in the appendix.

This still associates a seam-continuous, piecewise quadratic function to each texel and reproduces the bilinear functions at interior texels. However, for a boundary texel \( t \), the functions \( \tilde{B}_t \) and \( B_t \) no longer agree on the intersection of \( M \) with the footprint of \( t \). (See Figure 2b, bottom row.)

4.2 Comparison with soft continuity constraints

We compare our construction of a continuous function space \( \{ \phi_t \} \) to the approach of Liu et al. [2017] which enforces continuity on the traditional bilinear basis by introducing a soft constraint \( E_C \).

For a general function \( \phi \), the energy \( E_C(\phi; h) \) measures the integrated squared difference between the values of \( \phi \) on opposite sides of a seam. We can include this continuity energy into Equation (1) as an additional term:

\[
E(\phi; h, \alpha, \psi, \omega) + \alpha \cdot \lambda \cdot E_C(\phi; h) ,
\]

where \( \lambda \) modulates the importance of continuity across the seam.

Figure 5 shows examples of signal diffusion using two different screening weights (Section 7.1), comparing the results obtained using the bilinear basis with soft constraints to the results obtained using our continuous basis. Renderings are obtained using the texture mapping hardware, with basis coefficients used as texel values. For large-time-scale diffusion (top), a low continuity weight results in insufficient cohesion between charts, and colors do not diffuse across chart boundaries. For short-time-scale diffusion (bottom), a high continuity weight encourages the function to be constant along the seam, resulting in perceptible color “smearing”. Our continuous basis provides correct results for both scenarios and does not require any parameter tuning.

5 DISCRETIZING DIFFERENTIAL OPERATORS

Performing gradient domain processing requires choosing a basis for representing the target vector field and discretizing the derivative operator. Following the Discrete Exterior Calculus approach [Crane et al. 2013a], we do this in two steps. We first define a cotangent vector field basis and a discrete derivative operator, both of which depend only on the triangulation connectivity. Then, we define the

3 A rare exception is if the footprint of a texel contains nodes that are on opposite sides of a seam, e.g., at the poles of the sinusoidal projection.

4 Note that when solving a diffusion equation, i.e., \( \omega = 0 \), the screening weight is inversely proportional to the time-scale of diffusion.
metric-dependent Hodge stars, giving inner-products on the spaces of scalar functions and vector fields. We combine these to obtain the system matrices for gradient domain processing.

We highlight the importance of capturing the surface metric by comparing the solution to the single-source geodesic distance problem (Section 7.3) using the 2D Euclidean metric $\mu$ of the texture domain and using the immersion metric $g$ of the surface mesh. As seen in Figure 6, the 2D Euclidean metric (left) produces concentric and uniformly spaced circles in the texture domain, but these do not correspond to geodesic circles on the surface due to parametric distortion. In contrast, our metric-aware approach (right) generates a set of distorted contours in the texture domain that map to uniformly spaced geodesic circles on the surface.

5.1 Metric-independent discretization

We use the Whitney basis to represent cotangent vector fields (i.e., 1-forms). Each basis element is associated to an unordered pair of derivative as a set of adjacent texels, i.e., pairs of texels whose basis functions have overlapping support:

$$\mathcal{A} \equiv \{ (s, t) \in \mathcal{T} \times \mathcal{T} \mid s < t \text{ and } \text{supp}(\phi_s) \cap \text{supp}(\phi_t) \neq \emptyset \}.$$ 

Given the scalar function basis $\{\phi_t\}$ and denoting the exterior derivative as $d$, the Whitney 1-form basis $\{\omega_o\}$ is defined as:

$$\omega_o = \phi_b \cdot d\phi_t - d\phi_b \cdot \phi_t, \quad \forall o = (s, t) \in \mathcal{A}.$$ 

Discrete exterior derivative. We denote by $d \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ the matrix giving the signed incidence of texels along adjacent texel pairs:

$$d_{(s, t)} = \begin{cases} -1 & \text{if } r = s \\ 1 & \text{if } r = t \\ 0 & \text{otherwise} \end{cases} \quad \forall r \in \mathcal{T} \text{ and } (s, t) \in \mathcal{A}.$$ 

5.2 Metric-dependent discretization

We recall that since the $\{\phi_t\}$ form a partition of unity, the matrix $d$ gives the discretization of the exterior derivative in the bases $\{\phi_t\}$ and $\{\omega_o\}$. That is, for a given scalar basis function $\phi_t$, we have:

$$d\phi_t = d\phi_t \cdot \left( \sum_{b \in \mathcal{S}} \phi_b \right) - \phi_t \cdot d \left( \sum_{b \in \mathcal{S}} \phi_b \right)$$

$$= \sum_{b \in \mathcal{S}} (d\phi_t \cdot \phi_b - \phi_t \cdot d\phi_b)$$

$$= \sum_{o \in \mathcal{A}} d_{o,s} \cdot \omega_o.$$ 

We compute the integrals by using the canonical coordinate frame for $[0, 1] \times [0, 1] \supset M$ and combining the two triangulations described earlier: the parameterized surface triangulation $\mathcal{T}$ and the triangulation $\mathcal{T}$ obtained by tessellating $M$ using the texture lattice.
By assumption, the matrix expression for $h$ is constant on each triangle in $T$. By construction, the basis functions $\{\phi_t\}$ and $\{\omega_o\}$ are polynomial on each triangle in $\hat{T}$. Thus, computing a mutual refinement $T \oplus \hat{T}$ of the two triangulations (Figure 3) and summing the integrals over the faces of $T \oplus \hat{T}$, the computation of the Hodge stars reduces to integrating polynomials over 2D polygons.

We compute the integrals over each face in the refinement by triangulating the face and using 11-point quadrature [Day and Taylor 2007], which is exact for polynomials up to degree six. (Since $\{\phi_t\}$ are piecewise quadratic polynomials and $\{\omega_o\}$ are piecewise cubic, computing $\star_h^1$ requires integrating fourth-order polynomials and computing $\star_h^6$ requires integrating sixth-order polynomials.)

5.3 Defining the linear system

In our applications, we are interested in computing functions minimizing the quadratic energy $E(\phi; h, \alpha, \psi, \omega)$ from Equation (1). Using the Euler-Lagrange formulation and discretizing with respect to the function basis, the coefficients of the minimizing $\hat{x}^* \in \mathbb{R}^{\hat{T}}$ are given as the solution to the linear system

$$\left( \alpha \cdot M_h + S_h \right) \cdot \hat{x}^* = \alpha \cdot \text{mass}_h(\psi) + \text{div}_h(\omega).$$

Here $M_h$ and $S_h$ are the mass and stiffness matrices, given by:

$$M_h \equiv \star_h^0 \quad \text{and} \quad S_h \equiv \text{d}^T \cdot \star_h^1 \cdot \text{d},$$

and $\text{mass}_h(\psi) \in \mathbb{R}^{\mathbb{S}^2}$ and $\text{div}(\omega)h \in \mathbb{R}^{\mathbb{S}^2}$ are obtained by integrating against the (differentials of the) basis functions:

$$\text{mass}_h(\psi)_t = \int_M [\mu^{-1} \circ h] \cdot \phi_t \cdot \text{d} \mu,$$

$$\text{div}(\omega)_t = \int_M [\mu^{-1} \circ h] \cdot h^{-1}(d\phi_t, \omega) \text{ d} \mu.$$

When $\psi$ or $\omega$ cannot be expressed as a linear combination of basis functions (e.g., in smoothing and sharpening, stitching, and line integral convolution applications), the constraints simplify:

$$\psi = \sum_{t \in \mathbb{T}} \bar{x}_t \cdot \phi_t \quad \Rightarrow \quad \text{mass}_h(\psi) = M_h \cdot \bar{x}, \quad (2)$$

$$\omega = \text{d} \left( \sum_{t \in \mathbb{T}} \bar{y}_t \cdot \phi_t \right) \quad \Rightarrow \quad \text{div}_h(\omega) = S_h \cdot \bar{y}, \quad (3)$$

$$\omega = \sum_{o \in \mathbb{O}} \bar{z}_o \cdot \omega_o \quad \Rightarrow \quad \text{div}(\omega) = \text{d}^T \cdot \star_h^1 \cdot \bar{z}. \quad (4)$$

When $\psi$ or $\omega$ cannot be expressed as a linear combination of basis functions (e.g., in computing single-source geodesic distances), we approximate the integrals using quadrature.

6 MULTIGRID

The applications we consider are formulated as solutions to sparse symmetric positive-definite linear systems. On domains with irregular connectivity like triangle meshes, these type of systems are commonly solved either through direct methods, like sparse Cholesky factorization, or through iterative methods, like conjugate gradients. Both approaches have limitations within an interactive system: Cholesky factorization requires expensive precomputation and the back-substitution is inherently serial, while iterative methods like conjugate gradients converge too slowly.

To support interactivity, we implement a multigrid solver that exploits the regularity of the texture domain. The challenge in doing so is handling the irregularity that arises at the seams. We resolve this by using domain-decomposition [Smith et al. 1996], partitioning the degrees of freedom into interior, where we leverage regularity, and boundary, where the system is small enough to be handled by a direct solver. We start by describing the implementation of the multigrid solver and then discuss performance.

6.1 Hierarchy construction

Our input is a texture grid where charts are separated sufficiently so that the footprint of each texel intersects a single chart. The set of texels in the input grid defines the finest resolution of our hierarchy. We construct the coarser levels by generating a multiresolution grid for each chart independently, as shown in the inset. We select a texel in the finest resolution (level 0) as the origin (shown in red in the inset), and define the texels $\mathbb{T}^l$ at the $l$-th hierarchy level as the subset of finest-level grid nodes with indices $[2^l m, 2^l k]$ whose $[-2^l, 2^l] \times [-2^l, 2^l]$ footprints intersect the chart. Extending the definitions from Section 3, we classify texels at coarser levels of the hierarchy as interior or boundary by checking whether their footprint is entirely contained within a chart.

Each texel of the hierarchy indexes a basis function. Texels at the finest resolution are associated with the continuous basis $\{\phi_t\}$ introduced in Section 4. We implicitly construct the coarse function spaces using the Galerkin approach, defining a prolongation matrix $P^l = \{P^l_i\}$ that expresses basis functions at coarser level $l+1$ as linear combinations of (at most) 9 basis functions at level $l$. The coefficients are given by the bilinear up-sampling stencil, (see inset). The restriction matrix is defined as $R^l \equiv (P^l)^T$. Then, given a matrix $A$ defined at the finest resolution, we recursively construct the restriction of this matrix to the coarser levels of the hierarchy, setting $A^0 = A$ and $A^{l+1} = R^l \cdot A^{l} \cdot P^l$.

6.2 Solution update

To solve the system $A \cdot \bar{x} = \bar{b}$ (with known constraints $\bar{b}$ and unknown coefficients $\bar{x}$), we update the estimated solution by performing a V-Cycle [Briggs et al. 2000]. Starting at the finest resolution, we recursively relax the solution and restrict the residual to the next coarser level. At the coarsest resolution, we solve the small system using a direct solver. Then, we recursively add the prolonged correction to the estimated solution at the next finer level, and apply further relaxation.

To perform the V-cycle efficiently, we rearrange variables in blocks of interior (i) and boundary (b) texels, and rewrite the linear system:

$$A = \begin{bmatrix} A_{ii} & A_{ib} \\ A_{bi} & A_{bb} \end{bmatrix}, \quad \bar{x} = \begin{bmatrix} \bar{x}_i \\ \bar{x}_b \end{bmatrix}, \quad \bar{b} = \begin{bmatrix} \bar{b}_i \\ \bar{b}_b \end{bmatrix}.$$
interior relaxation, boundary solution, solution at the coarsest level, and restriction and prolongation. Memory coherence and parallelism make the average cost of relaxing an interior texel significantly lower than solving for a boundary texel. Thus, a V-cycle becomes less efficient as the atlas becomes more fragmented. The cost of solving at the coarse level and the cost of applying restriction and prolongation is a small fraction of the overall runtime. Evaluating sharpening is done by solving the gradient-domain problem in Equation (1), setting \( h = g, \alpha = 10^4, \psi \) equal to the input normal map, and \( \omega = 3 \cdot \delta \psi \), and using a multigrid system with \( L = 4 \) hierarchy levels and \( n = 3 \) Gauss-Seidel relaxations per level. The solutions for the boundary texels and for the full system at the coarsest level are obtained using CHOLMOD [Chen et al. 2008]. These tests are performed on a quad-core i7-6700HQ processor.

### 6.3 Performance

We analyze the performance of our multigrid solver by sharpening normal-maps over three different characterizations of the Julius model, shown in Figure 9. Sharpening is done using the gradient-domain problem in Equation (1), setting \( h = g, \alpha = 10^4, \psi \) equal to the input normal map, and \( \omega = 3 \cdot \delta \psi \), and using a multigrid system with \( L = 4 \) hierarchy levels and \( n = 3 \) Gauss-Seidel relaxations per level. The solutions for the boundary texels and for the full system at the coarsest level are obtained using CHOLMOD [Chen et al. 2008].

#### Runtime

Figure 10 shows the runtime decomposition for a single V-cycle using double precision. We plot the aggregate times for interior relaxation, boundary solution, solution at the coarsest level, and restriction and prolongation. Memory coherence and parallelism make the average cost of relaxing an interior texel significantly lower than solving for a boundary texel. Thus, a V-cycle becomes less efficient as the atlas becomes more fragmented. The cost of solving at the coarse level and the cost of applying restriction and prolongation is a small fraction of the overall runtime. Evaluating using texture maps with \( 0.2M, 0.8M, 3.2M, \) and \( 12.8M \) texels, we found that performance scales almost linearly with the number of texels, with improved parallelism at higher resolutions due to the increased per-thread workload.

#### Comparison to direct solvers

Table 1 compares the performance of our multigrid system with two direct solvers: CHOLMOD [Chen et al. 2008] and PARDISO [Petra et al. 2014a,b]. All solvers are run in double precision. For each one, we report three timings:

- **Initialization**: For direct solvers, this is the symbolic factorization of the fine system \( A^f \). For multigrid, this is the symbolic factorization of the boundary and coarse systems \( A^b_{bb} \) and \( A^b \).
- **Update**: For direct solvers, this is the numerical factorization of \( A^f \). For multigrid, this is the numerical factorization of \( A^b_{bb} \) and \( A^b \) as well as the computation of the intermediate linear systems \( (A^{i+1} = R^i \cdot A^i \cdot P^i) \).
- **Solution**: For direct solvers, this is back-substitution updating the three coordinates separately. For multigrid, this is a single (parallelized) V-Cycle pass updating the coordinates together.

---

**Figures**

- **Fig. 7.** Visualization of the neighbors of an interior texel (left) and a boundary texel (right). The selected texel is highlighted in blue and the neighbors are highlighted in black.
- **Fig. 8.** The V-cycle algorithm with domain-decomposition updates interior and exterior texels separately in both restriction and prolongation phases.
- **Fig. 9.** We compare the performance of normal-map sharpening (\( \beta = 3 \)) using atlases with 1, 4, and 28 charts, showing the input normal-maps (top) and the sharpened results after one V-cycle (bottom).

---

**System Equations**

\[
\begin{align*}
\text{Initialization:} & \quad A^f \cdot \hat{x}^f = \hat{b}^f \\
\text{Update:} & \quad A^f \cdot \hat{x}^f = A^b_{bb} \cdot \hat{x}^b + A^b \cdot \hat{x}^i \\
\text{Solution:} & \quad \hat{x}^f = \hat{x}^b + G^f \hat{x}^i
\end{align*}
\]

We update the solution at interior texels by locking the boundary coefficients, adjusting the constraints to account for the solution met at the boundary, and performing multiple passes of Gauss-Seidel relaxation over the interior coefficients. Leveraging the grid-regularity of texel adjacency (Figure 7, left), relaxation of interior texels can be done efficiently using multi-coloring (parallelization) and temporal-blocking (memory coherence) [Weiss et al. 1999].

As boundary texels have irregular adjacency patterns (Figure 7, right), Gauss-Seidel relaxation is less efficient. However, because the number of boundary texels is small, these can be updated using a direct solver at interactive rates. This time we lock interior coefficients, adjust the constraints to account for the solution met in the interior, and perform a direct solve for the boundary coefficients.

Our V-cycle algorithm (Figure 8) performs the interior relaxation before the boundary solution in the restriction phase, and after in the prolongation phase. (\texttt{Solve}(A, \hat{b})) computes the solution to the system \( A \cdot \hat{x} = \hat{b} \) using a direct solver and \texttt{GSSRelax}(A, \hat{b}, \hat{x}, n) performs \( n \) Gauss-Seidel relaxations with \( \hat{x} \) as the initial guess.)
Table 1. For each solver, we list, from left to right, the time for initialization, update, and solution (in seconds). The construction of the mass and stiffness matrices is the same for all solvers, and takes between 1 and 3 seconds.

<table>
<thead>
<tr>
<th>Model</th>
<th>CHOLMOD</th>
<th>PARDISO</th>
<th>Our multigrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Julius-1C</td>
<td>3.8 : 1.2 : 0.2</td>
<td>2.8 : 0.8 : 0.2</td>
<td>0.4 : 0.1 : 0.04</td>
</tr>
<tr>
<td>Julius-4C</td>
<td>4.0 : 1.4 : 0.2</td>
<td>2.9 : 0.8 : 0.3</td>
<td>0.5 : 0.2 : 0.04</td>
</tr>
<tr>
<td>Julius-28C</td>
<td>4.0 : 1.3 : 0.2</td>
<td>3.1 : 0.9 : 0.3</td>
<td>0.6 : 0.4 : 0.06</td>
</tr>
</tbody>
</table>

As Table 1 shows, direct solvers incur heavy initialization and update costs due to the factorization (symbolic and numerical, respectively) of large system matrices. In contrast, our approach only requires factorization of small matrices—the ones associated to the boundary nodes and the one at the coarsest resolution. Our multigrid approach also updates the solution at interactive rates, five times faster than a direct solver. In practice, we have found that it takes between two and four V-cycles to obtain a solution that is indistinguishable from a direct solver’s solution.

6.4 Convergence

We assess the convergence of our solver by analyzing how RMS error decreases with the number of V-cycles. Figure 11 (top) shows plots of the RMS error for the models shown in the paper, using the same linear system ($h = g, \alpha = 10^4, \omega = 0, \psi$ set to random texture), at the same resolution (texture images are rescaled to have 800K texels), with ground-truth obtained using a direct solver.

For all models the RMS error decays exponentially up to machine precision. To better understand the different convergence rates, we analyze the effects of parametric distortion on the solver.

Distortion. To measure distortion, we scale each 3D model so that its surface area equals the area of the triangulation in the parametric domain and then consider the singular values of the affine transformations mapping 3D triangles into 2D. As in the work of Smith and Schaefer [2015], we use a symmetric Dirichlet energy that equally penalizes singular values and their reciprocals.

Unlike the earlier work, we define this energy in log-space:

$$\mathcal{E}_D(\sigma_1, \sigma_2) = \log^2(\sigma_1) + \log^2(\sigma_2).$$

An advantage of this formulation is that we can express the energy as the sum $\mathcal{E}_D = \mathcal{E}_A + \mathcal{E}_C$ of authalic and conformal energies:

$$\mathcal{E}_A(\sigma_1, \sigma_2) = \frac{1}{2} \left[ \log(\sigma_1) + \log(\sigma_2) \right]^2 = \frac{1}{2} \log^2(\sigma_1 \cdot \sigma_2)$$

$$\mathcal{E}_C(\sigma_1, \sigma_2) = \frac{1}{2} \left[ \log(\sigma_1) - \log(\sigma_2) \right]^2 = \frac{1}{2} \log^2(\sigma_1/\sigma_2).$$

To better understand how distortion affects convergence rates, we plot the RMS error after five V-cycles against the 99-th percentile distortion in Figure 11 (bottom). Surprisingly, convergence is weakly correlated with area (authalic) distortion. Rather, it is the deviation from conformality, as reflected by larger values of $\mathcal{E}_C$, that correlates strongly with slower convergence.

We corroborate this empirical observation by computing an as-rigid-as-possible [Liu et al. 2008] parameterization of the Julius head (1C-ARAP). Then, we obtain new charts by applying a Möbius transformation (1C-ARAP-MOEB), applying an anisotropic scale (1C-ARAP-ANISO), and partitioning into 28 charts (1C-ARAP-28C). Note that 1C-ARAP, 1C-ARAP-MOEB, and 1C-ARAP-28C have the same conformal distortions while 1C-ARAP, 1C-ARAP-ANISO, and 1C-ARAP-28C have the same authalic distortions.

Figure 12 shows the convergence plots for the four different atlases. As the figure shows, neither the application of a Möbius...
transformation nor the introduction of new seams significantly affects the convergence rate of the solver. In contrast the introduction of anisotropy significantly degrades the solver’s performance.

Note that though it does not necessarily improve convergence, reducing area distortion is still important for ensuring that the discretization samples the function space uniformly.

Resolution. We also analyze the performance of our multigrid solver as a function of resolution. Fixing the parameterization, we up-sample the texture map and consider the convergence of the multigrid solver at different resolutions.

Figure 13 shows representative results for four different resolutions of the Julius-28C atlas. As the figure shows, though the RMS error decays exponentially, the convergence rate slows as resolution is increased. We do not have a satisfying explanation for this behavior and intend to continue studying this in the future.

Single precision solver. Using single precision, we obtain a roughly $2\times$ speedup for the interior relaxation and for the restriction and prolongation stages, though numerical precision limits the achievable accuracy. The error reduction is similar to that of double precision for the first 5-8 iterations, at which point the single precision solver plateaus to an RMS error of roughly $10^{-5}$.

Triangle quality. Though convergence efficiency depends on the parametric distortion, it is less dependent on the quality of the triangulation. For example, if there is no distortion, the discretization of the linear system depends only on the parameterization of the chart boundaries and not on the shapes of the triangles.

7 APPLICATIONS

We demonstrate the versatility of our approach by considering a number of applications of gradient-domain processing. For each of these, the solution is obtained by solving for the minimizer

$$\phi^* = \arg \min_{\phi} E(\phi; h, \alpha, \psi, \omega),$$

with $h$ the metric, $\alpha$ the screening weighting, $\psi$ the target scalar field, and $\omega$ the target differential.

We use single precision and, with the exception of the last application, results are obtained using our multigrid solver, with $L = 4$ hierarchy levels, $n = 3$ Gauss-Seidel iterations per level, and using CHOLMOD to solve for the boundary nodes and coarsest resolution system. Immersions are scaled so the surface has unit area (because the effects of $\alpha$ and $h$ are scale-dependent). All parameterizations, with the exception of those shown in Figure 12, are obtained using UVAtlas [Microsoft 2018]. Please see Table 2 in the appendix for performance statistics.

Source code for our texture-space gradient-domain processing can be found at https://github.com/mkazhdan/TextureSignalProcessing/.

7.1 Isotropic filtering

A signal $\psi$ is smoothed and sharpened by solving for a new signal with scaled differential. Following the approach of Bhat et al. [2008], we compute a filtered signal $\phi^*$ as the minimizer

$$\phi^* = \arg \min_{\phi} E(\phi; g, 10^4, \psi, \beta \cdot \psi),$$

with $\beta$ the differential scaling term (and $g$ the immersion metric). Setting $\bar{x}$ to the coefficients of the input signal and using Equations (2) and (3), the coefficients $\bar{x}'$ of the minimizer are given by

$$\left(10^4 \cdot M_g + S_g \right) \cdot \bar{x}' = \left(10^4 \cdot M_g + \sum_{c \in C} \beta_c \cdot S_{g,c} \right) \cdot \bar{x}.$$

When $\beta < 1$, the differential of the input signal is dampened and the signal is smoothed. When $\beta > 1$, the differential is amplified, and the signal is sharpened. Figure 9 shows results of sharpening a normal map and Figure 14 shows results of smoothing and sharpening a color texture.

Local filtering. Selective removal or enhancement of signal detail is obtained by allowing $\beta$ to vary spatially. Figure 15 shows an example of local filtering where an input texture (a) is filtered to produce both sharpening and smoothing effects (c). The spatially varying modulation mask (b) prescribes that the furrow should be amplified (red) while the bags under the eyes should be removed (blue). We represent $\beta$ as a piecewise constant function, with a value associated to each cell $c \in C$. The minimizer is given by

$$\left(10^4 \cdot M_g + S_g \right) \cdot \bar{x}' = \left(10^4 \cdot M_g + \sum_{c \in C} \beta_c \cdot S_{g,c} \right) \cdot \bar{x},$$

where $S_{g,c}$ is the stiffness matrix with integration restricted to c,

$$(S_{g,c})_{x,t} = \int_{\gamma} [g^{-1} \cdot \frac{\partial}{} \cdot \frac{\partial}{} (d\phi_x, d\phi_t)] d\mu,$$

and $\beta_c$ is the differential modulation factor at c.

We designed an interactive system for texture filtering using a spray-can interface to prescribe local modulation weights $\beta$. We precompute the matrices $S_{g,c}$. Then, at run-time, the user-specified modulation weights are transformed into linear constraints and our multigrid solver generates the new texture values at interactive rates, approximately 18 frames per second on the ballerina model (740k texels). Please refer to the accompanying video for a demonstration.

7.2 Texture stitching

Previous works in image and geometry processing merge multiple signals by formulating stitching as a gradient-domain problem [Agarwala et al. 2004; Levin et al. 2003; Pérez et al. 2003]. These approaches use the input signals to compute differences between pairs of adjacent elements and solve for a global signal that matches the differences in a least squares sense. Here, we describe how to use our framework to stitch together textures obtained by imaging a static object from multiple viewpoints.
Fig. 14. We smooth (left) and sharpen (right) a texture by solving linear systems that dampen and amplify the local color variation.

Fig. 15. We design a user interface for local filtering. In the middle we visualize the differential modulation mask used for this example, showing attenuation (smoothing) in blue and amplification (sharpening) in red.

Our input is a texture-atlased surface, together with a collection of partial textures \( \{ \psi_k \} \) and a segmentation mask \( \zeta \). Figure 16 shows an example with three partial textures. The partial textures sample the color of visible texels from each camera’s viewpoint, and the segmentation mask specifies the camera providing the best view. (The quality of a view is determined by visibility as well as the alignment of the surface normal to the camera’s view direction.)

A naive solution is to create a composite texture \( \psi \) by using the camera with the best view to assign a texel’s color. As shown in the middle left of Figure 16, this reveals abrupt illumination changes at the transitions between regions covered by different cameras.

These discontinuities are removed by solving for a texture that preserves the differential within the partial textures and is smooth across the boundaries. To achieve this, we set the target texel difference to zero for texel pairs residing on different partial textures:

\[
\tilde{z}_d = \begin{cases} 
\hat{\psi}(t) - \hat{\psi}(s) & \text{if } \zeta(s) = \zeta(t), \\
0 & \text{otherwise}
\end{cases} \quad \forall d = (s, t) \in \mathcal{A}.
\]

In regions not seen by any camera, differences are also set to zero to encourage a smooth fill-in. We construct the associated one-form \( \omega = \sum_{d \in \mathcal{A}} \tilde{z}_d d \), and solve for the signal with matching differential:

\[
\phi^* = \arg \min_{\phi} E(\phi; g, 10^3, 0),
\]

Setting \( \tilde{x} \) to the coefficients of the composite and using Equations (2) and (4), the coefficients \( \tilde{x}^* \) of the solution are given by

\[
\left( 10^2 \cdot M_g + S_g \right) \cdot \tilde{x}^* = 10^2 \cdot M_g \cdot \tilde{x} + d^{\top} \cdot \gamma^1 \cdot \tilde{z}.
\]

Fig. 16. Given a set of partial textures and a segmentation mask (top row), we stitch the partial textures into a single texture. Direct compositing produces a result that reveals the different lighting conditions (bottom left). Using gradient-domain stitching, the result is robust to illumination change between the cameras (bottom right).

7.3 Single-source geodesic distances

We demonstrate the robustness of our approach by computing single-source geodesic distances using the Geodesics-in-Heat method [Crane et al. 2013b]. The approach computes distances to a source point \( p \in M \) by solving two successive systems. The first solves for a short-time-scale diffusion of an impulse \( \delta_p \) at the surface point:

\[
\psi^* = \arg \min_{\psi} E(\psi; g, 10^3, \delta_p, 0).
\]

The second solves for the function whose differential best matches the (negated) normalized differential of the diffused impulse:
\[ \phi^* = \arg \min_{\phi} E \left( \phi; g, 0, \cdot, -\frac{d\phi^*}{|d\psi|} \right). \]  

(Setting \(\alpha = 0\), the target scalar field has no effect.)

In the texture domain, we associate the impulse with a texel \(t \in T\), defining the vector \(\bar{x}\) whose coefficients is one at the \(t\)-th texel and zero for all others. Using Equation (2) we obtain the coefficients of the smoothed impulse by solving \((10^3 \cdot M_g + S_p) \cdot \bar{x}^u = 10^3 \cdot M_g \cdot \bar{x}\).

The coefficients \(\bar{y}^u \in \mathbb{R}^{|T|}\) of the geodesic function are obtained by solving the system \(S_g \cdot \bar{y}^u = -\text{div}_g(d\psi /[d\psi])\). Leveraging the smoothness of \(\psi\), we use one-point quadrature to approximate the values of \(\text{div}_g(d\psi /[d\psi])\). Assuming a connected surface, the solution is unique up to a constant factor and we offset the solution so that the distance is 0 at the source: \(\bar{y}^u \leftarrow \bar{y}^u - \bar{y}_{\text{source}}\).

Figure 17 (top) shows the distance function for a source point selected on the cheek of the bunny. Note that after three V-cycles (second row), the result is indistinguishable from the result obtained with a direct solver (third row).

This application is unusual in that the constraint to the second linear system depends on the solution to the first, and hence evolves with the V-cycle iterations. Nonetheless, Figure 17 (bottom) shows that the RMS error for both \(\bar{x}^u\) and \(\bar{y}^u\) decays exponentially.

We also validated the efficiency of our multigrid solver in an interactive application in which a user picks a source texel and the application displays the estimated geodesic distances after each multigrid pass. When a source texel is selected the smoothed impulse and distance functions are initialized to zero. Then, at each frame, one V-cycle is performed for the impulse diffusion system and a second is performed for the distance estimation (using the solution from the first V-cycle to define the constraints for the second). We achieve an interactive rate of 17 frames per second on the bunny model (670k texels). Please refer to the accompanying video for a demonstration.

### 7.4 Line integral convolution

Lastly, we consider the application of line integral convolution [Cabral and Leedom 1993] to surface vector field visualization. Teitzel et al. [1997] achieve this by tracing streamlines over the triangulation and averaging a random signal over these paths, obtaining a signal defined over the mesh. Palacios and Zhang [2011] interactively project the field onto the view-plane, obtaining a signal in screen-space. We show that surface-based line integral convolution can be computed in the texture domain by using gradient-domain processing with an anisotropic metric.

Given a vector field \(\bar{u}\), we first define a metric \(h_{\bar{u}}\) that shrinks distances along \(\bar{u}\) and stretches them along the perpendicular direction. Then, we diffuse a random texture \(\psi\) along the streamlines by solving

\[ \psi^* = \arg \min_{\psi} E(\psi, h_{\bar{u}}, 1, \psi, 0). \]

Given a vector field \(\bar{u}\) which is constant per triangle in the canonical coordinate frame of \(M\) and given tangent vectors \(v, w \in T_p M\), we define the anisotropic metric by setting

\[ h_{\bar{u},p}(v, w) \equiv g_p(v, \bar{u}^-(p)) \cdot g_p(w, \bar{u}^-(p)) + \epsilon \cdot g_p(v, w), \]

with \(\bar{u}^\perp\) the vector-field perpendicular to \(\bar{u}\) (relative to \(g\)) and \(\epsilon \cdot g_p(v, w)\) a regularizing term ensuring that \(h_{\bar{u},p}\) is non-singular.

Figure 1 (right) and Figure 18 show visualizations of surface curvature. For these we define \(\bar{u}\) by scaling principal curvature directions by the absolute difference in principal curvature values.

This application highlights the robustness of our finite-elements discretization, which provides high-quality vector-field visualizations despite significant distortion in the metric.

### 8 CONCLUSION AND DISCUSSION

We have presented a framework for gradient-domain processing directly in a texture atlas, avoiding the need for resampling. The framework introduces novel function spaces and Hodge stars to enable seamless, metric-aware computations. A multigrid algorithm leverages the grid regularity, in conjunction with domain decomposition, to attain interactive performance over megapixel domains. We apply the framework to a variety of applications.
we have not encountered this problem in practice.

which is time-consuming in our current implementation.

\[ \mu \text{ (approximately) a scalar multiple of the 2D Euclidean metric} \]

near-singular maps could lead to issues of numerical precision, but the system matrices requires inverting the metric tensors. Even

tions (e.g., at the poles of the equirectangular map) because defining

tations (Section 4) involves the computation and traversal of triangulations (Section 4) which is time-consuming in our current implementation.

Our implementation also does not support singular parameterizations (e.g., at the poles of the equirectangular map) because defining the system matrices requires inverting the metric tensors. Even near-singular maps could lead to issues of numerical precision, but we have not encountered this problem in practice.

Future work: We hope to extend our approach in several ways:

- complete the DEC picture by defining a sparse Hodge 2-star, enabling vector-field processing in the texture domain;
- consider more tailored prolongation matrices (as in algebraic multigrid) to extend our multigrid solver to the context of anisotropic filtering;
- extend our implementation to support interactive update of the system matrices (e.g., for processing signals on a deforming surface) by leveraging the constancy of the texture atlas;
- achieve temporally coherent filtering of time-varying signals on fixed-connectivity or evolving meshes.

ACKNOWLEDGMENTS

This work is supported by the NSF award 1422325. We thank Microsoft for the Slick, Girl, Ballerina, and Mime datasets. We thank the Stanford University Computer Graphics Laboratory for the Bunny and David models. We thank the AIM@SHAPE-VISIONAIR Shape Repository for the Julius, Bimba, Camel, Fertility, and Filigree models.

REFERENCES

Practical multigrid is known to converge slowly for inhomogeneous or anisotropic systems. In practice, we find that our multigrid solver converges efficiently when the metric \( h \) is (approximately) a scalar multiple of the 2D Euclidean metric \( \mu \). This is the case for the applications in Sections 7.1–7.3 because texture atlassing tends to construct parameterizations that preserve lengths, up to global scale. For line integral convolution (Section 7.4) the prescribed metric distortion is so severe that multigrid exhibits poor convergence and we instead use a direct solver.

Determining the coefficients of the system matrix (Section 5) involves the computation and traversal of triangulations (Section 4) which is time-consuming in our current implementation.

Our implementation also does not support singular parameterizations (e.g., at the poles of the equirectangular map) because defining the system matrices requires inverting the metric tensors. Even near-singular maps could lead to issues of numerical precision, but we have not encountered this problem in practice.

Future work: We hope to extend our approach in several ways:

- complete the DEC picture by defining a sparse Hodge 2-star, enabling vector-field processing in the texture domain;
- consider more tailored prolongation matrices (as in algebraic multigrid) to extend our multigrid solver to the context of anisotropic filtering;
- extend our implementation to support interactive update of the system matrices (e.g., for processing signals on a deforming surface) by leveraging the constancy of the texture atlas;
- achieve temporally coherent filtering of time-varying signals on fixed-connectivity or evolving meshes.

ACKNOWLEDGMENTS

This work is supported by the NSF award 1422325. We thank Microsoft for the Slick, Girl, Ballerina, and Mime datasets. We thank the Stanford University Computer Graphics Laboratory for the Bunny and David models. We thank the AIM@SHAPE-VISIONAIR Shape Repository for the Julius, Bimba, Camel, Fertility, and Filigree models.

REFERENCES


Min curvature diffusion

Max curvature diffusion

Min curvature diffusion

Fig. 18. We perform line integral convolution by modifying the surface metric to diffuse a random signal (top) along the directions of principal curvature (middle and bottom).

Limitations. Standard multigrid is known to converge slowly for inhomogeneous or anisotropic systems. In practice, we find that our multigrid solver converges efficiently when the metric \( h \) is (approximately) a scalar multiple of the 2D Euclidean metric \( \mu \). This is the case for the applications in Sections 7.1–7.3 because texture atlassing tends to construct parameterizations that preserve lengths, up to global scale. For line integral convolution (Section 7.4) the prescribed metric distortion is so severe that multigrid exhibits poor convergence and we instead use a direct solver.

Determining the coefficients of the system matrix (Section 5) involves the computation and traversal of triangulations (Section 4) which is time-consuming in our current implementation.

Our implementation also does not support singular parameterizations (e.g., at the poles of the equirectangular map) because defining the system matrices requires inverting the metric tensors. Even near-singular maps could lead to issues of numerical precision, but we have not encountered this problem in practice.

Future work: We hope to extend our approach in several ways:
Gradient-Domain Processing within a Texture Atlas • 154:13


APPENDIX

CLAIM 1. The unnormalized and normalized texel functions \( \tilde{\phi}_t \) and \( \phi_t \) reproduce the bilinear function \( B_t \) whenever \( t \) is an interior texel.

PROOF. Let \( Q : M \to \mathbb{R} \) be a continuous, piecewise quadratic function on \( M \) (i.e., quadratic on each triangle \( \tilde{t} \)). Because the functions \( \{Q_n\} \) are quadratic Lagrange elements on \( M \), if we linearly combine them with weights obtained by evaluating \( Q \) at the nodes \( n \in N \), we reproduce the function \( Q \):

\[
Q(p) = \sum_{n \in N} Q_n(p), \quad \forall p \in M.
\]

Now if \( t \) is an interior texel, the nodes in the support of \( B_t \) cannot lie on a seam. Specifically, let \( n \in N \) be a node and \( \{n_i\} \in N \) be the equivalence class containing \( n \). If \( B_t(n_i) \neq 0 \) then \( |n_i| = |n| \). Therefore, for any \( n \) in the support of \( B_t \), we have:

\[
\tilde{Q}[n] \equiv \sum_{m \in [n]} Q_m = Q_n.
\]

Thus, for an interior texel \( t \in \tilde{T} \) we get:

\[
\tilde{\phi}_t(p) = \sum_{n \in \tilde{n}} \left( \sum_{n \in n} B_t(n) \cdot \tilde{Q}_n(p) \right) = \sum_{n \in N} B_t(n) \cdot Q_n(p) = B_t(p),
\]

so that \( \tilde{\phi}_t \) reproduces \( B_t \). Since for an interior texel \( t \) we have

\[
\phi_t \equiv \sum_{n \in N} \left( \frac{1}{|n|} \sum_{n \in n} B_t(n) \right) \cdot \tilde{Q}_n = \sum_{n \in N} \left( \sum_{n \in n} B_t(n) \right) \cdot \tilde{Q}_n \equiv \tilde{\phi}_t,
\]

the function \( \phi_t \) also reproduces \( B_t \). □

LEMMA 2. For any equivalence class \( \tilde{n} \in \tilde{N} \), the sum of the bilinear basis functions evaluated at elements of \( \tilde{n} \) equals the cardinality of \( \tilde{n} \):

\[
\sum_{n \in \tilde{n}} B_t(n) = |\tilde{n}|.
\]

PROOF. Since the \( \{B_t\} \) form a partition of unity on \( M \) we have:

\[
\sum_{\tilde{t} \in \tilde{T}} \sum_{n \in \tilde{n}} B_t(n) = \sum_{n \in \tilde{n}} B_t(n) = \sum_{\tilde{t} \in \tilde{T}} \sum_{n \in \tilde{n}} B_t(n) = 1 = |\tilde{n}|.
\]

CLAIM 3. The functions \( \{\phi_t\} \) form a partition of unity on \( M \):

\[
\sum_{\tilde{t} \in \tilde{T}} \phi_t(p) = 1, \quad \forall p \in M.
\]

PROOF. Let \( B \in \mathbb{R}^{15 |\tilde{N}| \times 15 |\tilde{T}|} \) be the matrix whose \((\tilde{n}, \tilde{t})\)-th entry is the coefficient of \( \tilde{Q}_n \) in the expression of \( \phi_t \):

\[
\phi_t = \sum_{\tilde{n} \in \tilde{N}} B_{\tilde{n}, \tilde{t}} \cdot \tilde{Q}_n.
\]

ACM Trans. Graph., Vol. 37, No. 4, Article 154. Publication date: August 2018.
Since the functions \( \{ Q_k \} \) form a partition of unity on \( M \), the condition that the functions \( \{ \tilde{\varphi}_t \} \) form a partition of unity on \( M \) is equivalent to the condition that the sum of the \( \{ \tilde{\varphi}_t \} \) is equal to the sum of the \( \{ Q_k \} \). Or, equivalently, that the rows of \( B \) sum to one:

\[
\sum_{t \in C} B_{n,t} = 1.
\]

Let \( \tilde{B} \in \mathbb{R}^{|\mathcal{X}| \times |\mathcal{Y}|} \) be the matrix whose \((\tilde{n}, t)\)-th entry is the coefficient of \( \tilde{Q}_k \) in the expression of \( \tilde{\varphi}_t \). By construction, we have:

\[
\tilde{B}_{n,t} = \sum_{t \in C} B_{t}(n).
\]

We can define \( B \) by normalizing the entries of \( \tilde{B} \) by their row-sums.

By Lemma 2, the \( \tilde{n} \)-th row-sum of \( \tilde{B} \) is the cardinality of \( \tilde{n} \):

\[
\sum_{t \in C} \tilde{B}_{\tilde{n},t} = \sum_{t \in C} \sum_{n \in \tilde{n}} B_{t}(n) = |\tilde{n}|.
\]

Thus, normalizing \( \tilde{B} \) by its row-sums we can define

\[
B_{\tilde{n},t} = \sum_{n \in \tilde{n}} \frac{1}{|\tilde{n}|} B_{t}(n).
\]

Or equivalently,

\[
\tilde{\varphi}_t = \sum_{n \in \tilde{n}} \left( \frac{1}{|\tilde{n}|} \sum_{n \in \tilde{n}} B_{t}(n) \right) \cdot \tilde{Q}_n.
\]

While one could perform this normalization in other ways (e.g., by setting \( \tilde{\varphi}_t = \tilde{\varphi}_t / \sum \tilde{\varphi}_t \)), normalizing by the row-sum is particularly simple and ensures that the functions \( \{ \tilde{\varphi}_t \} \) are linear combinations of the \( \{ \tilde{Q}_k \} \), e.g., piecewise quadratic.

<table>
<thead>
<tr>
<th>Name</th>
<th>Fig.</th>
<th>Resolution</th>
<th>Texels</th>
<th>Charts</th>
<th>V-cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slick</td>
<td>1</td>
<td>1024 x 1024</td>
<td>734K</td>
<td>29</td>
<td>38</td>
</tr>
<tr>
<td>Filigree</td>
<td>1</td>
<td>1024 x 1024</td>
<td>701K</td>
<td>168</td>
<td>64</td>
</tr>
<tr>
<td>Camel</td>
<td>1</td>
<td>2048 x 2048</td>
<td>2837K</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>Girl</td>
<td>2</td>
<td>1024 x 1024</td>
<td>722K</td>
<td>36</td>
<td>39</td>
</tr>
<tr>
<td>Bimba</td>
<td>6</td>
<td>1024 x 1024</td>
<td>724K</td>
<td>1</td>
<td>23</td>
</tr>
<tr>
<td>Julius-1C</td>
<td>9</td>
<td>1018 x 1018</td>
<td>800K</td>
<td>1</td>
<td>23</td>
</tr>
<tr>
<td>Julius-4C</td>
<td>9</td>
<td>1089 x 1089</td>
<td>800K</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>Julius-28C</td>
<td>9</td>
<td>522 x 522</td>
<td>200K</td>
<td>28</td>
<td>14</td>
</tr>
<tr>
<td>Julius-28C</td>
<td>9</td>
<td>1055 x 1055</td>
<td>800K</td>
<td>28</td>
<td>39</td>
</tr>
<tr>
<td>Julius-28C</td>
<td>9</td>
<td>2127 x 2127</td>
<td>3200K</td>
<td>28</td>
<td>121</td>
</tr>
<tr>
<td>Julius-28C</td>
<td>9</td>
<td>4266 x 4266</td>
<td>12800K</td>
<td>28</td>
<td>409</td>
</tr>
<tr>
<td>Ballerina</td>
<td>14</td>
<td>1024 x 1024</td>
<td>740K</td>
<td>87</td>
<td>49</td>
</tr>
<tr>
<td>David head</td>
<td>15</td>
<td>1024 x 1024</td>
<td>749K</td>
<td>14</td>
<td>34</td>
</tr>
<tr>
<td>Mime</td>
<td>16</td>
<td>1024 x 1024</td>
<td>727K</td>
<td>35</td>
<td>41</td>
</tr>
<tr>
<td>Bunny</td>
<td>17</td>
<td>1024 x 1024</td>
<td>670K</td>
<td>9</td>
<td>29</td>
</tr>
<tr>
<td>Fertility</td>
<td>18</td>
<td>2048 x 2048</td>
<td>2770K</td>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Processing statistics, including the resolution of the texture image, the number of texels, the number of charts, and the time for a single V-cycle pass (in milliseconds). As line integral convolution requires a direct solver, we do not provide V-cycle times for the Camel and Fertility models.