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

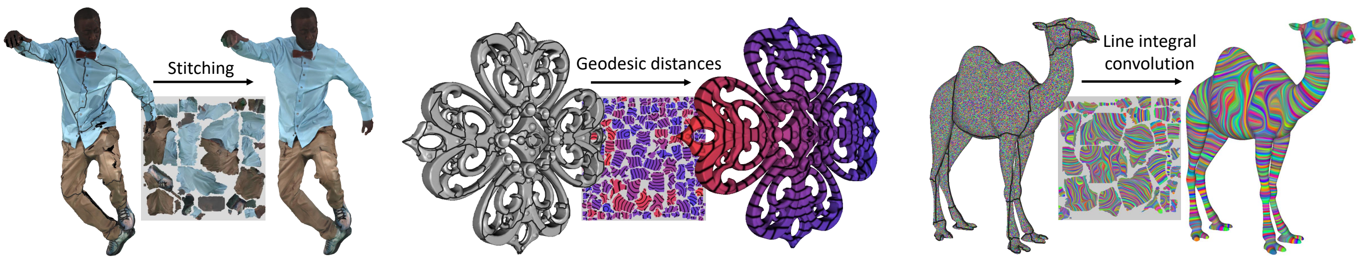


Fig. 1. We process surface signals directly in the texture atlas domain, thereby exploiting the regularity of the 2D grid sampling. Example applications include multiview stitching, computation of geodesic distance maps, and curvature-guided line integral convolution. (Black curves indicate chart boundaries.)

Processing signals on surfaces often involves resampling the signal over the vertices of a dense mesh and applying mesh-based filtering operators. We present a framework to process a signal directly in a texture atlas domain. The benefits are twofold: avoiding resampling degradation and exploiting the regularity of the texture image grid. The main challenges are to preserve continuity across atlas chart boundaries and to adapt differential operators to the non-uniform parameterization. We introduce a novel function space and multigrid solver that jointly enable robust, interactive, and geometry-aware signal processing. We demonstrate our approach using several applications including smoothing and sharpening, multiview stitching, geodesic distance computation, and line integral convolution.

CCS Concepts: • **Computing methodologies** → *Computer graphics*;

Additional Key Words and Phrases: signal processing, Laplacian filtering, mesh parameterization, multigrid solver, domain decomposition

ACM Reference Format:
Fabián Prada, Misha Kazhdan, Ming Chuang, and Hugues Hoppe. 2018. Gradient-Domain Processing within a Texture Atlas. *ACM Trans. Graph.* 37, 4, Article 154 (August 2018), 14 pages. https://doi.org/10.1145/3197517.3201317

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 ψ, target texture differential ω, and a screening weight $\alpha \geq 0$ balancing fidelity to ψ and ω, the goal is to find the texture values ϕ that minimize the energy

$$\mathrm{E}(\phi; h, \alpha, \psi, \omega) = \alpha \cdot \|\phi - \psi\|_h^2 + \|d\phi - \omega\|_h^2 . \quad (1)$$

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.

Authors' addresses: Fabián Prada; Misha Kazhdan, Johns Hopkins University, 3400 N Charles St, Baltimore, MD, 21218, USA, fabianprada@gmail.com, misha@cs.jhu.edu; Ming Chuang, PerceptIn Inc. 4633 Old Ironsides Dr, Snata Clara, CA, 95054, USA, mingchuang82@gmail.com; Hugues Hoppe, Google Inc. 601 N 34th St, Seattle, WA, 98103, USA, hhoppe@gmail.com.

0730-0301/2018/8-ART154 $15.00
https://doi.org/10.1145/3197517.3201317

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; Tasdizen 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 tesselation, 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 [Lefebvre 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.

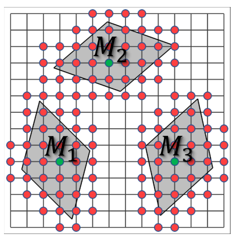
3 PRELIMINARIES

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 $\Phi : V \longrightarrow \mathbb{R}^3$ to the map $\Phi : M \longrightarrow \mathbb{R}^3$ 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 $p \sim q$ and that a function $\phi : M \longrightarrow \mathbb{R}$ 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 $\mathcal{T}$. 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, $h_p : T_pM \times T_pM \longrightarrow \mathbb{R}$. We recall several facts about h:

- Given tangent vectors $v, w \in T_pM$, 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 $\psi : M \longrightarrow \mathbb{R}$, the integral of ψ with respect to the metric h is defined to be
$$\int_M \psi \, dh \equiv \int_M \sqrt{|\mu^{-1} \circ h|} \cdot \psi \, d\mu \,,$$
with μ 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 μ denote the standard (in this case, 3D) Euclidean metric, we define the *immersion metric* g as:
$$g_p(v, w) \equiv \mu_{\Phi(p)}\Big(d\Phi|_p(v), d\Phi|_p(w)\Big)\,, \qquad \forall v, w \in T_pM\,.$$

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

[2] Note 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.

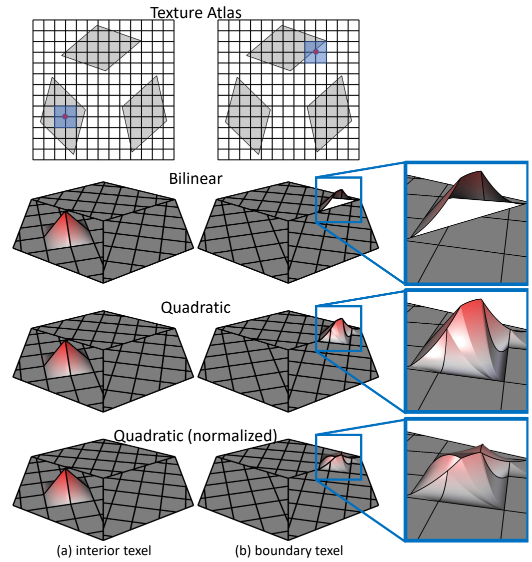


Fig. 2. Visualization of finite-element basis functions. The top row shows texture space with interior and boundary texels selected. Texel footprints are highlighted in blue. The lower rows show the corresponding bilinear, quadratic, and normalized-quadratic functions on the planar surface.

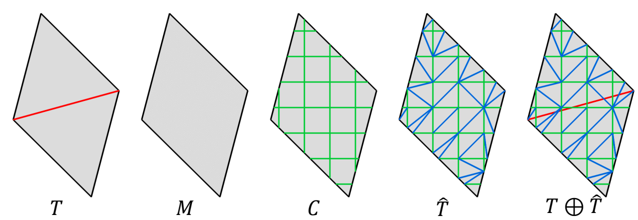


Fig. 3. Illustration of the different triangulations and polygonizations. Starting with a triangulation T, we compute the connected components M, clip these to the texture lattice to get a quad-dominant tesselation C, compute a constrained Delaunay triangulation $\widehat{T}$, and then compute the mutual refinement $T \oplus \widehat{T}$ of the initial and constrained triangulations.

Please see Table 3 in the appendix for a summary of the notation used throughout the paper.

4 INTER-CHART CONTINUITY

Our goal is to associate a basis function to each texel $t \in \mathcal{T}$ so that a set of discrete texture values can be interpreted as a function that can be evaluated anywhere on M.

Perhaps the simplest approach is to associate texel $t \in \mathcal{T}$ with the bivariate, first-order B-spline B_t centered at t. This conforms to the bilinear rasterization performed by graphics hardware. While such functions are well-behaved for interior texels, they are not seam-continuous for boundary texels, dropping to zero on the opposite side of the seam (Figure 2, second row).

4.1 Continuity by construction

Our approach is to define a basis $\{\phi_t\}_{t \in \mathcal{T}}$ consisting of seam-continuous functions that *approximate* the bilinear kernels $\{B_t\}$.

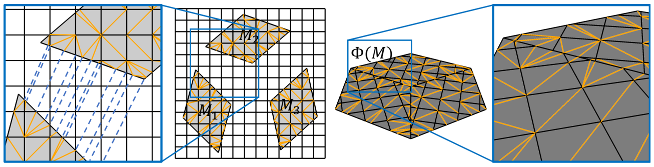


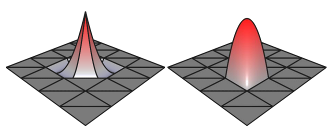
Fig. 4. Consistently clipping the M to the texture lattice and triangulating (left) gives a triangulation of the surface $\Phi(M)$ without T-junctions (right).

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 $\{\widetilde{Q}_{\tilde{n}}\}$ on M; (3) defining bilinear-like texel functions $\{\phi_t\}$ as linear combinations of the $\{\widetilde{Q}_{\tilde{n}}\}$.

(1) Triangulating the texture domain. We decompose the atlas domain M into a set of polygonal cells 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 $\widetilde{\mathcal{N}} = \mathcal{N}/\sim$ 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 $\widetilde{Q}_{\tilde{n}}$ to each equivalence class $\tilde{n} \in \widetilde{\mathcal{N}}$ by summing the quadratic Lagrange elements associated to nodes in the equivalence class:

$$\widetilde{Q}_{\tilde{n}} = \sum_{n\in\tilde{n}} Q_n .$$

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 $\overline{\phi}_t : M \to \mathbb{R}$ to be the linear combination of $\{\widetilde{Q}_{\tilde{n}}\}$, with coefficients given by evaluating the bilinear function B_t at the node positions:

$$\overline{\phi}_t(p) \equiv \sum_{\tilde{n}\in\widetilde{\mathcal{N}}} \left(\sum_{n\in\tilde{n}} B_t(n) \right) \cdot \widetilde{Q}_{\tilde{n}}(p) .$$

By construction, the $\{\overline{\phi}_t\}$ 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 $\overline{\phi}_t$ reproduces the bilinear function B_t whenever t is an interior texel (Figure 2a). Generally, the functions $\overline{\phi}_t$ 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 $\{\overline{\phi}_t\}$ 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:

$$\phi_t(p) \equiv \sum_{\tilde{n}\in\widetilde{\mathcal{N}}} \left(\frac{1}{|\tilde{n}|} \sum_{n\in\tilde{n}} B_t(n) \right) \cdot \widetilde{Q}_{\tilde{n}}(p) ,$$

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 ϕ_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 ϕ, the energy $\mathrm{E}_C(\phi; h)$ measures the integrated squared difference between the values of ϕ on opposite sides of a seam. We can include this continuity energy into Equation (1) as an additional term:

$$\mathrm{E}(\phi; h, \alpha, \psi, \omega) + \alpha \cdot \lambda \cdot \mathrm{E}_C(\phi; h) ,$$

where λ 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[4] (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.

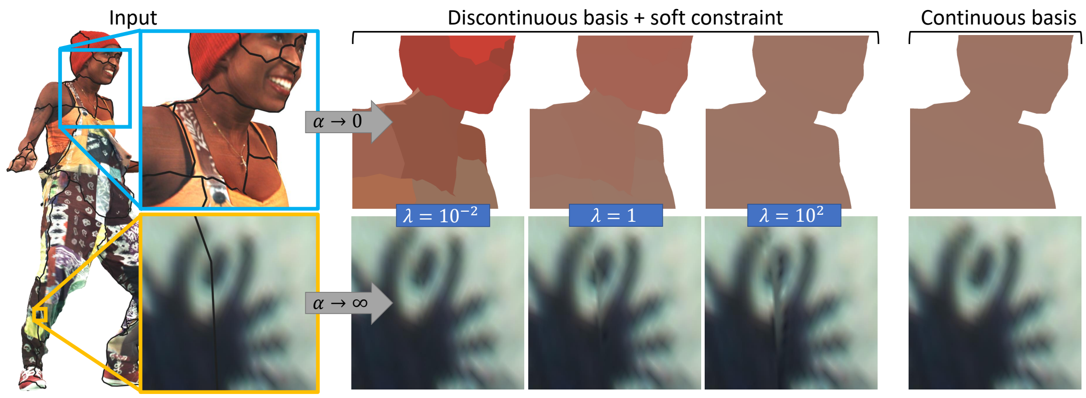


Fig. 5. We compare diffusion using the standard bilinear basis with soft constraints (middle) to diffusion using our continuous basis (right), showing results for large ($\alpha \longrightarrow 0$) and short ($\alpha \longrightarrow \infty$) time-scales. With soft constraints, we must tune the continuity weight λ as no single value works for all cases. With our continuous basis, no such tuning is required.

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 μ 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.

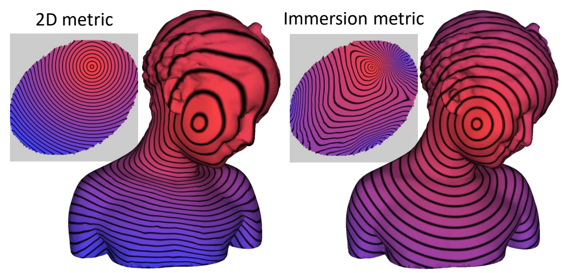


Fig. 6. We compare geodesic distances obtained using the 2D Euclidean metric (left) and the immersion metric (right).

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 adjacent texels and is defined as the symmetric difference of the product of the scalar function at one texel times the differential of the scalar function at the other. Because the functions $\{\phi_t\}$ form a partition of unity, we obtain a discretization of the exterior derivative, given in terms of finite differences [Bossavit 1988].

Whitney Basis. Let "$\prec$" be some precedence operator on texels, and let $\mathcal{A}$ denote the set of adjacent texels, i.e., pairs of texels whose basis functions have overlapping support:

$$\mathcal{A} \equiv \left\{ (s, t) \in \mathcal{T} \times \mathcal{T} \,\middle|\, s \prec t \text{ and } \mathrm{supp}(\phi_s) \cap \mathrm{supp}(\phi_t) \neq \emptyset \right\}.$$

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

$$\omega_a = \phi_s \cdot d\phi_t - d\phi_s \cdot \phi_t, \qquad \forall a = (s, t) \in \mathcal{A}.$$

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

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

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

$$\begin{aligned} d\phi_t &= d\phi_t \cdot \left(\sum_{s \in \mathcal{T}} \phi_s \right) - \phi_t \cdot d \left(\sum_{s \in \mathcal{T}} \phi_s \right) \\ &= \sum_{s \in \mathcal{T}} \left(d\phi_t \cdot \phi_s - \phi_t \cdot d\phi_s \right) \\ &= \sum_{a \in \mathcal{A}} \mathbf{d}_{ta} \cdot \omega_a . \end{aligned}$$

5.2 Metric-dependent discretization

Given a Riemannian metric h, we would like to compute the Hodge 0-star $\star_h^0 \in \mathbb{R}^{|\mathcal{T}| \times |\mathcal{T}|}$ and Hodge 1-star $\star_h^1 \in \mathbb{R}^{|\mathcal{A}| \times |\mathcal{A}|}$:

$$\begin{aligned} \left(\star_h^0 \right)_{s,t} &= \int_M \sqrt{|\mu^{-1} \circ h|} \cdot \phi_s \cdot \phi_t \, d\mu , \\ \left(\star_h^1 \right)_{a,b} &= \int_M \sqrt{|\mu^{-1} \circ h|} \cdot h^{-1}(\omega_a, \omega_b) \, d\mu . \end{aligned}$$

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 T and the triangulation $\widehat{T}$ obtained by tessellating M using the texel lattice.

By assumption, the matrix expression for h is constant on each triangle in T. By construction, the basis functions $\{\phi_t\}$ and $\{\omega_\alpha\}$ are polynomial on each triangle in $\widehat{T}$. Thus, computing a mutual refinement $T \oplus \widehat{T}$ of the two triangulations (Figure 3) and summing the integrals over the faces of $T \oplus \widehat{T}$, 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 $\{\omega_\alpha\}$ are piecewise cubic, computing $\star_h^0$ requires integrating fourth-order polynomials and computing $\star_h^1$ requires integrating sixth-order polynomials.)

5.3 Defining the linear system

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

$$(\alpha \cdot \mathbf{M}_h + \mathbf{S}_h) \cdot \vec{x}^* = \alpha \cdot \mathrm{mass}_h(\psi) + \mathrm{div}_h(\omega)\,.$$

Here $\mathbf{M}_h$ and $\mathbf{S}_h$ are the mass and stiffness matrices, given by:[5]

$$\mathbf{M}_h \equiv \star_h^0 \qquad \text{and} \qquad \mathbf{S}_h \equiv \mathbf{d}^\top \cdot \star_h^1 \cdot \mathbf{d}\,,$$

and $\mathrm{mass}_h(\psi) \in \mathbb{R}^{|\mathcal{T}|}$ and $\mathrm{div}(\omega)_h \in \mathbb{R}^{|\mathcal{T}|}$ are obtained by integrating against the (differentials of the) basis functions:

$$\mathrm{mass}_h(\psi)_t \equiv \int_M \sqrt{|\mu^{-1} \circ h|} \cdot \phi_t \cdot \psi \, d\mu\,,$$

$$\mathrm{div}_h(\omega)_t \equiv \int_M \sqrt{|\mu^{-1} \circ h|} \cdot h^{-1}(d\phi_t, \omega)\, d\mu\,.$$

When ψ or ω can be expressed as a linear combination of basis functions (e.g., in smoothing and sharpening, stitching, and line integral convolution applications), the constraints simplify:

$$\psi = \sum_{t \in \mathcal{T}} \vec{x}_t \cdot \phi_t \quad \Rightarrow \quad \mathrm{mass}_h(\psi) = \mathbf{M}_h \cdot \vec{x}\,, \tag{2}$$

$$\omega = d\left(\sum_{t \in \mathcal{T}} \vec{y}_t \cdot \phi_t\right) \quad \Rightarrow \quad \mathrm{div}_h(\omega) = \mathbf{S}_h \cdot \vec{y}\,, \tag{3}$$

$$\omega = \sum_{\alpha \in \mathcal{A}} \vec{z}_\alpha \cdot \omega_\alpha \quad \Rightarrow \quad \mathrm{div}_h(\omega) = \mathbf{d}^\top \cdot \star_h^1 \cdot \vec{z}\,. \tag{4}$$

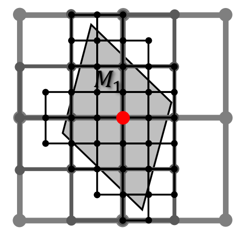
When ψ or ω cannot be expressed as a linear combination of basis functions (e.g., in computing single-source geodesic distances), we approximate the integrals using quadrature.

6 MULTIGRID

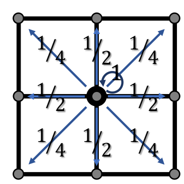
The applications we consider are formulated as solutions to sparse symmetric positive-definite linear systems. On domains with irregular connectivity like triangle meshes, these type of systems are commonly solved either through direct methods, like sparse Cholesky factorization, or through iterative 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 $\mathcal{T}^l$ 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 $\mathbf{P}^l$ 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 $\mathbf{R}^l \equiv (\mathbf{P}^l)^\top$. Then, given a matrix $\mathbf{A}$ defined at the finest resolution, we recursively construct the restriction of this matrix to the coarser levels of the hierarchy, setting $\mathbf{A}^0 = \mathbf{A}$ and $\mathbf{A}^{l+1} = \mathbf{R}^l \cdot \mathbf{A}^l \cdot \mathbf{P}^l$.

6.2 Solution update

To solve the system $\mathbf{A} \cdot \vec{x} = \vec{b}$ (with known constraints $\vec{b}$ and unknown coefficients $\vec{x}$), 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

[5] In practice $\mathbf{S}_h$ is computed directly by integrating the dot products of the differentials of the scalar functions $\{d\phi_t\}$. This is more efficient because $|\mathcal{A}| \approx 4|\mathcal{T}|$ and more stable because the integrands are only second-order polynomials.

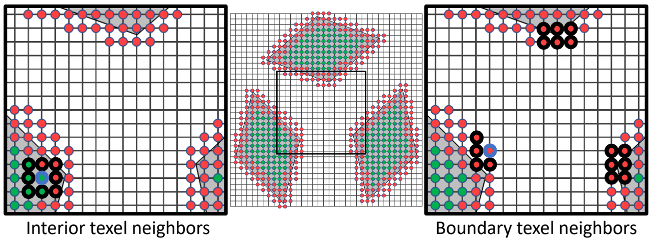


Fig. 7. Visualization of the neighbors of an interior texel (left) and a boundary texel (right). The selected texel is highlighted in blue and the neighbors are highlighted in black.

R.1 for $l = 0, \ldots, L-1$
R.2 $\quad \vec{r}^{\,l}_{\mathrm{i}} \leftarrow \vec{b}^{\,l}_{\mathrm{i}} - \mathbf{A}^{l}_{\mathbf{ib}} \cdot \vec{x}^{\,l}_{\mathrm{b}}$ *// boundary-relative residual*
R.3 $\quad \vec{x}^{\,l}_{\mathrm{i}} \leftarrow$ **GaussSeidelRelax**($\mathbf{A}^{l}_{\mathbf{ii}}$, $\vec{r}^{\,l}_{\mathrm{i}}$, $\vec{x}^{\,l}_{\mathrm{i}}$, n)
R.4 $\quad \vec{r}^{\,l}_{\mathrm{b}} \leftarrow \vec{b}^{\,l}_{\mathrm{b}} - \mathbf{A}^{l}_{\mathbf{bi}} \cdot \vec{x}^{\,l}_{\mathrm{i}}$ *// interior-relative residual*
R.5 $\quad \vec{x}^{\,l}_{\mathrm{b}} \leftarrow$ **Solve** ($\mathbf{A}^{l}_{\mathbf{bb}}$, $\vec{r}^{\,l}_{\mathrm{b}}$)
R.6 $\quad \vec{b}^{\,l+1} \leftarrow \mathbf{R}^{l} \cdot \left(\vec{b}^{\,l} - \mathbf{A}^{l} \cdot \vec{x}^{\,l}\right)$ *// restricted residual*
C.1 $\vec{x}^{\,L} \leftarrow$ **Solve**($\mathbf{A}^{L}$, $\vec{b}^{\,L}$)
P.1 for $l = L-1, \ldots, 0$
P.2 $\quad \vec{x}^{\,l} \leftarrow \vec{x}^{\,l} + \mathbf{P}^{l} \cdot \vec{x}^{\,l+1}$ *// prolonged correction*
P.3 $\quad \vec{r}^{\,l}_{\mathrm{b}} \leftarrow \vec{b}^{\,l}_{\mathrm{b}} - \mathbf{A}^{l}_{\mathbf{bi}} \cdot \vec{x}^{\,l}_{\mathrm{i}}$ *// interior-relative residual*
P.4 $\quad \vec{x}^{\,l}_{\mathrm{b}} \leftarrow$ **Solve**($\mathbf{A}^{l}_{\mathbf{bb}}$, $\vec{r}^{\,l}_{\mathrm{b}}$)
P.5 $\quad \vec{r}^{\,l}_{\mathrm{i}} \leftarrow \vec{b}^{\,l}_{\mathrm{i}} - \mathbf{A}^{l}_{\mathbf{ib}} \cdot \vec{x}^{\,l}_{\mathrm{b}}$ *// boundary-relative residual*
P.6 $\quad \vec{x}^{\,l}_{\mathrm{i}} \leftarrow$ **GaussSeidelRelax**($\mathbf{A}^{l}_{\mathbf{ii}}$, $\vec{r}^{\,l}_{\mathrm{i}}$, $\vec{x}^{\,l}_{\mathrm{i}}$, n)

Fig. 8. Our V-cycle algorithm with domain-decomposition updates interior and exterior texels separately in both restriction and prolongation phases.

system $\mathbf{A}^{l} \cdot \vec{x}^{\,l} = \vec{b}^{\,l}$ at each level as

$$\begin{pmatrix} \mathbf{A}^{l}_{\mathbf{ii}} & \mathbf{A}^{l}_{\mathbf{ib}} \\ \mathbf{A}^{l}_{\mathbf{bi}} & \mathbf{A}^{l}_{\mathbf{bb}} \end{pmatrix} \begin{pmatrix} \vec{x}^{\,l}_{\mathrm{i}} \\ \vec{x}^{\,l}_{\mathrm{b}} \end{pmatrix} = \begin{pmatrix} \vec{b}^{\,l}_{\mathrm{i}} \\ \vec{b}^{\,l}_{\mathrm{b}} \end{pmatrix}.$$

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. (**Solve**$(\mathbf{A}, \vec{b})$ computes the solution to the system $\mathbf{A} \cdot \vec{x} = \vec{b}$ using a direct solver and **GSRelax**$(\mathbf{A}, \vec{b}, \vec{x}, n)$ performs n Gauss-Seidel relaxations with $\vec{x}$ as the initial guess.)

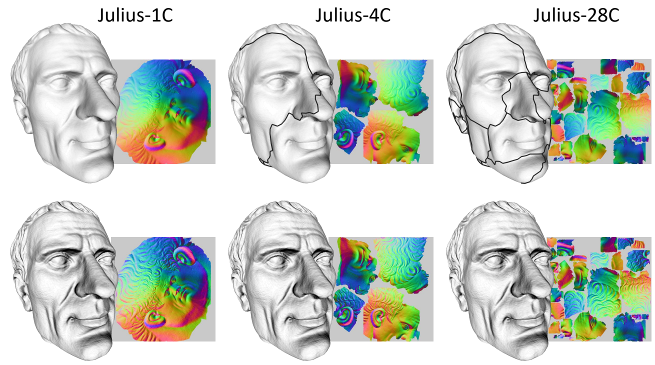


Fig. 9. We compare the performance of normal-map sharpening ($\beta = 3$) using atlases with 1, 4, and 28 charts, showing the input normal-maps (top) and the sharpened results after one V-cycle (bottom).

6.3 Performance

We analyze the performance of our multigrid solver by sharpening normal-maps over three different chartifications of the Julius model, shown in Figure 9. Sharpening is done by solving the gradient-domain problem in Equation (1), setting $h = g$, $\alpha = 10^4$, ψ equal to the input normal map, and $\omega = 3 \cdot d\psi$, 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.

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 $\mathbf{A}^0$. For multigrid, this is the symbolic factorization of the boundary and coarse systems $\{\mathbf{A}^{l}_{\mathbf{bb}}\}$ and $\mathbf{A}^{L}$.
- **Update**: For direct solvers, this is the numerical factorization of $\mathbf{A}^0$. For multigrid, this is the numerical factorization of $\{\mathbf{A}^{l}_{\mathbf{bb}}\}$ and $\mathbf{A}^{L}$ as well as the computation of the intermediate linear systems $\{\mathbf{A}^{l+1} = \mathbf{R}^{l} \cdot \mathbf{A}^{l} \cdot \mathbf{P}^{l}\}$.
- **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.

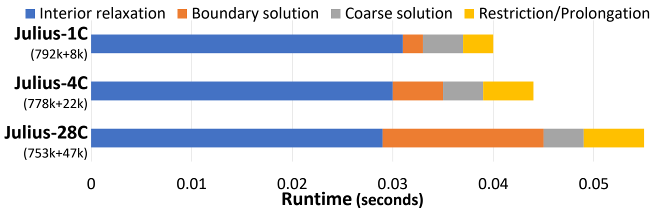


Fig. 10. Breakdown of V-cycle computations times: The numbers of interior and boundary texels at the finest resolution are specified under each model name. Interior relaxation is ~ 8× faster than boundary solution (per texel).

Model	CHOLMOD	PARDISO	Our multigrid
Julius-1C	3.8 : 1.2 : 0.2	2.8 : 0.8 : 0.2	0.4 : 0.1 : 0.04
Julius-4C	4.0 : 1.4 : 0.2	2.9 : 0.8 : 0.3	0.5 : 0.2 : 0.04
Julius-28C	4.0 : 1.3 : 0.2	3.1 : 0.9 : 0.3	0.6 : 0.4 : 0.06

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.

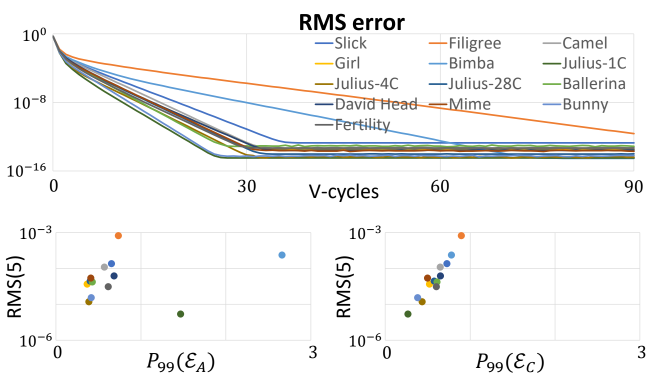


Fig. 11. Analysis of convergence: We show the RMS error as a function of the number of V-cycles, for the different models in this paper (top), and we plot the RMS after five V-cycles, as a function of the distortion (bottom). The two models with slowest convergence, Filigree and Slick, are also the ones whose parameterizations are least conformal.

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 ($h = g$, $\alpha = 10^4$, $\omega = 0$, and ψ set to random

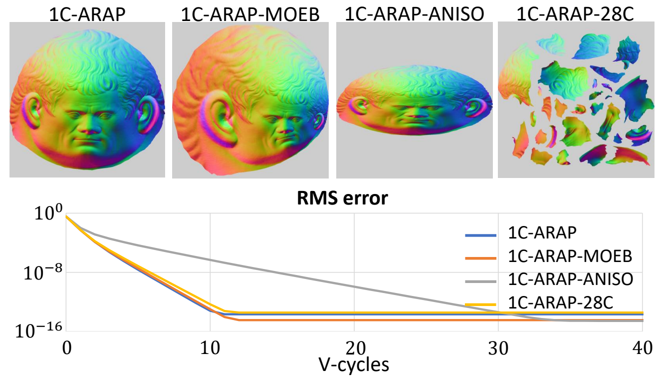


Fig. 12. Four different atlases for the Julius head (top), and the associated convergence plots (bottom).

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:

$$\mathcal{E}_A(\sigma_1, \sigma_2) = \frac{1}{2}\Big(\log(\sigma_1) + \log(\sigma_2)\Big)^2 = \frac{1}{2}\log^2(\sigma_1 \cdot \sigma_2)$$
$$\mathcal{E}_C(\sigma_1, \sigma_2) = \frac{1}{2}\Big(\log(\sigma_1) - \log(\sigma_2)\Big)^2 = \frac{1}{2}\log^2(\sigma_1 / \sigma_2).$$

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

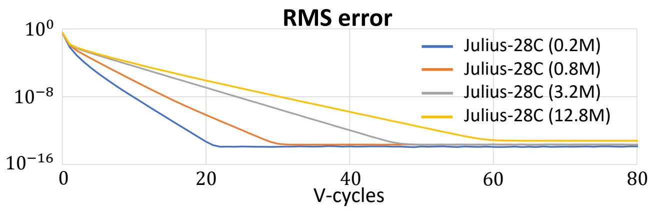


Fig. 13. Convergence of our solver for a single texture atlas using texture images with 0.2M, 0.8M, 3.2M, and 12.8M texels.

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× 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

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

with h the metric, α the screening weighting, ψ the target scalar field, and ω 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 α 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 ψ 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 ϕ^* as the minimizer

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

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

$$\left(10^4 \cdot \mathbf{M}_g + \mathrm{S}_g\right) \cdot \vec{x}^* = \left(10^4 \cdot \mathbf{M}_g + \beta \cdot \mathrm{S}_g\right) \cdot \vec{x}.$$

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 β 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 β as a piecewise constant function, with a value associated to each cell $c \in C$. The minimizer is given by

$$\left(10^4 \cdot \mathbf{M}_g + \mathrm{S}_g\right) \cdot \vec{x}^* = \left(10^4 \cdot \mathbf{M}_g + \sum_{c \in C} \beta_c \cdot \mathrm{S}_{g,c}\right) \cdot \vec{x},$$

where $\mathrm{S}_{g,c}$ is the stiffness matrix with integration restricted to c,

$$\left(\mathrm{S}_{g,c}\right)_{s,t} = \int_c \sqrt{|\mu^{-1} \circ g|} \cdot g^{-1}(d\phi_s, d\phi_t)\, d\mu,$$

and β_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 β. We precompute the matrices $\mathrm{S}_{g,c}$. 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.

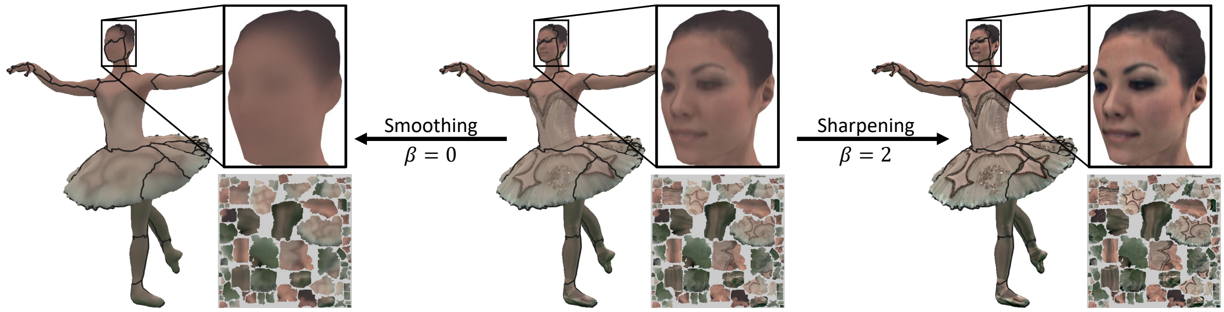


Fig. 14. We smooth (left) and sharpen (right) a texture by solving linear systems that dampen and amplify the local color variation.

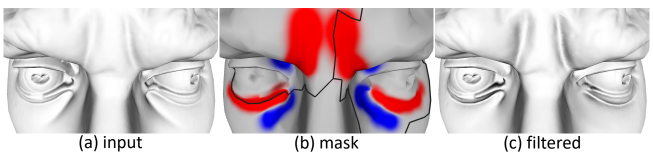


Fig. 15. We design a user interface for local filtering. In the middle we visualize the differential modulation mask used for this example, showing attenuation (smoothing) in blue and amplification (sharpening) in red.

Our input is a texture-atlased surface, together with a collection of *partial textures* $\{\psi_k\}$ and a *segmentation mask* ζ. 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 ψ 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:

$$\vec{z}_d \equiv \begin{cases} \psi(t) - \psi(s) & \text{if } \zeta(s) = \zeta(t) \\ 0 & \text{otherwise} \end{cases} \qquad \forall d = (s, t) \in \mathcal{A}.$$

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 $\omega = \sum_d \vec{z}_d \omega_d$, and solve for the signal with matching differential:

$$\phi^* = \arg\min_{\phi} \mathrm{E}(\phi; g, 10^2, \psi, \omega)\,.$$

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

$$\left(10^2 \cdot \mathbf{M}_g + \mathbf{S}_g\right) \cdot \vec{x}^* = 10^2 \cdot \mathbf{M}_g \cdot \vec{x} + \mathbf{d}^\top \cdot \star_g^1 \cdot \vec{z}\,.$$

Results of gradient-domain stitching are shown in Figures 1 (left) and 16 (middle right), where lighting differences between the cameras are removed, while details in the interior are preserved.

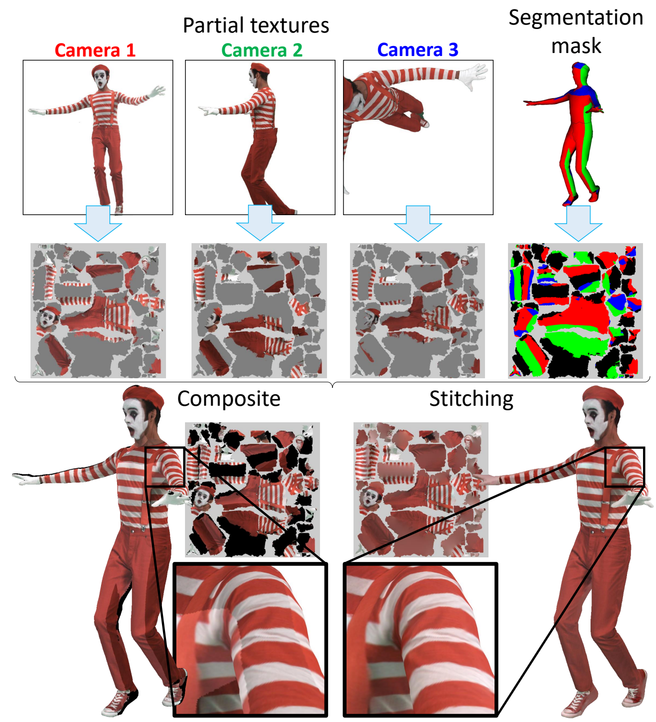


Fig. 16. Given a set of partial textures and a segmentation mask (top row), we stitch the partial textures into a single texture. Direct compositing produces a result that reveals the different lighting conditions (bottom left). Using gradient-domain stitching, the result is robust to illumination change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

$$\psi^* = \arg\min_{\psi} \mathrm{E}(\psi; g, 10^3, \delta_p, 0)\,. \tag{5}$$

The second solves for the function whose differential best matches the (negated) normalized differential of the diffused impulse:

$$\phi^* = \arg\min_\phi \mathrm{E}\left(\phi; g, 0, \bullet, -\frac{d\psi^*}{|d\psi^*|}\right). \quad (6)$$

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

In the texture domain, we associate the impulse with a texel $t \in \mathcal{T}$, defining the vector $\vec{x}$ 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 $(10^3 \cdot \mathbf{M}_g + \mathbf{S}_g) \cdot \vec{x}^* = 10^3 \cdot \mathbf{M}_g \cdot \vec{x}$.

The coefficients $\vec{y}^* \in \mathbb{R}^{|\mathcal{T}|}$ of the geodesic function are obtained by solving the system $\mathbf{S}_g \cdot \vec{y}^* = -\mathrm{div}_g(d\psi/|d\psi|)$. Leveraging the smoothness of ψ, we use one-point quadrature to approximate the values of $\mathrm{div}_g(d\psi/|d\psi|)$. 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: $\vec{y}^* \leftarrow \vec{y}^* - \vec{y}^*_t$.

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 $\vec{x}^*$ and $\vec{y}^*$ 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.

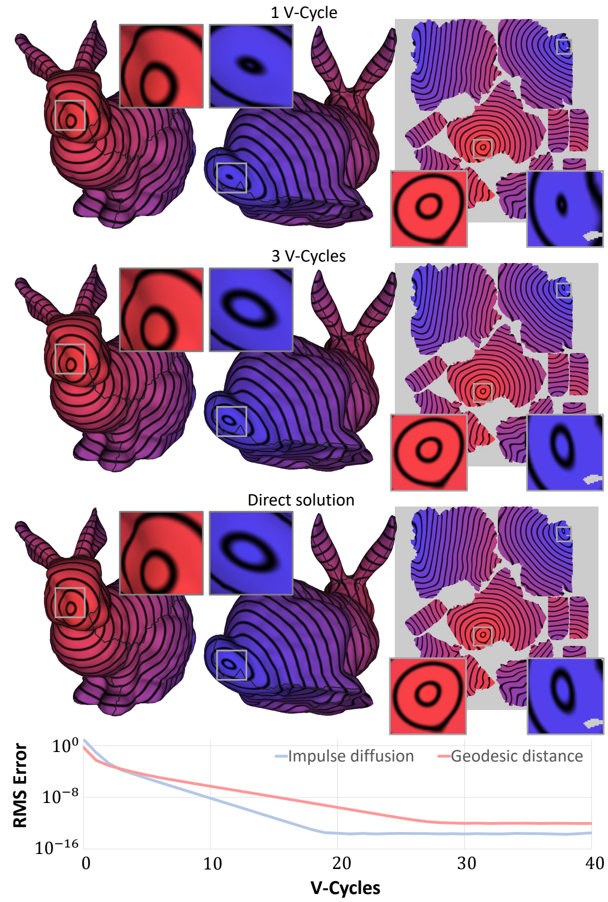


Fig. 17. Geodesic distance computation. The inset closeups show the source in the red region and the farthest point in the blue region. With a single V-cycle, the result is already similar to the exact solution from a direct solver (RMS= $5.2 \cdot 10^{-3}$). With three V-cycles, the result is almost indistinguishable (RMS= $2.7 \cdot 10^{-4}$). Despite the evolving right-hand side of the system, our multigrid solver exhibits exponential decay in the RMS error.

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 $\vec{u}$, we first define a metric $h_{\vec{u}}$ that shrinks distances along $\vec{u}$ and stretches them along the perpendicular direction. Then, we diffuse a random texture ψ along the streamlines by solving

$$\phi^* = \arg\min_\phi \mathrm{E}(\phi; h_{\vec{u}}, 1, \psi, 0).$$

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

$$h_{\vec{u},p}(v,w) \equiv g_p\left(v, \vec{u}^\perp(p)\right) \cdot g_p\left(w, \vec{u}^\perp(p)\right) + \epsilon \cdot g_p(v,w),$$

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

Figure 1 (right) and Figure 18 show visualizations of surface curvature. For these we define $\vec{u}$ 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.

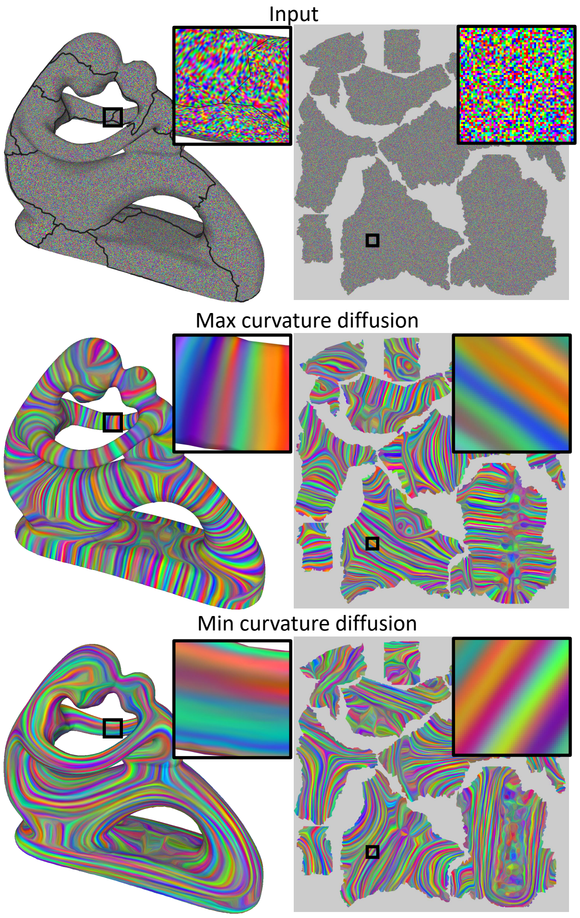


Fig. 18. We perform line integral convolution by modifying the surface metric to diffuse a random signal (top) along the directions of principal curvature (middle and bottom).

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 μ. This is the case for the applications in Sections 7.1–7.3 because texture atlasing 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

Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. 2004. Interactive digital photomontage. *ACM Trans. Graphics* 23 (2004), 294–302.

Marc Alexa and Max Wardetzky. 2011. Discrete Laplacians on general polygonal meshes. *ACM Trans. Graphics* 30 (2011), 102:1–102:10.

Chandrajit Bajaj and Guoliang Xu. 2003. Anisotropic diffusion of surfaces and functions on surfaces. *ACM Trans. Graphics* 22 (2003), 4–32.

Marcelo Bertalmio, Li-Tien Cheng, Stanley Osher, and Guillermo Sapiro. 2001. Variational problems and partial differential equations on implicit surfaces. *J. Computational Physics* 174 (2001), 759–780.

Pravin Bhat, Brian Curless, Michael Cohen, and Lawrence Zitnick. 2008. Fourier analysis of the 2D screened Poisson equation for gradient domain problems. In *10th European Conf. Computer Vision.* 114–128.

Alain Bossavit. 1988. Whitney forms: a class of finite elements for three-dimensional computations in electromagnetism. *IEE Proc. A (Physical Science, Measurement and Instrumentation, Management and Education, Reviews)* 135 (1988), 493–500.

William Briggs, Van Emden Henson, and Steve McCormick. 2000. *A multigrid tutorial (2nd ed.).* Society for Industrial and Applied Mathematics.

Brian Cabral and Leith Casey Leedom. 1993. Imaging vector fields using line integral convolution. In *20th Annual Conf. Computer Graphics and Interactive Techniques.* 263–270.

Nathan Carr and John Hart. 2002. Meshed atlases for real-time procedural solid texturing. *ACM Trans. Graphics* 21 (2002), 106–131.

Nathan Carr and John C. Hart. 2004. Painting detail. *ACM Trans. Graphics* 23 (2004), 845–852.

Yanqing Chen, Timothy Davis, William W. Hager, and Sivasankaran Rajamanickam. 2008. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Trans. Math. Softw.* 35 (2008), 22:1–22:14.

Ming Chuang, Linjie Luo, Benedict Brown, Szymon Rusinkiewicz, and Misha Kazhdan. 2009. Estimating the Laplace-Beltrami operator by restricting 3D functions. *Computer Graphics Forum (SGP '09)* (2009), 1475–1484.

Ming Chuang, Szymon Rusinkiewicz, and Michael Kazhdan. 2016. Gradient-domain processing of meshes. *J. Computer Graphics Techniques* 5 (2016), 44–55.

Ulrich Clarenz, Udo Diewald, and Martin Rumpf. 2000. Anisotropic geometric diffusion in surface processing. In *Conf. Visualization '00.* 397–405.

Keenan Crane, Fernando de Goes, Mathieu Desbrun, and Peter Schröder. 2013a. Digital Geometry Processing with Discrete Exterior Calculus. In *ACM SIGGRAPH 2013 Courses.* 7:1–7:126.

Keenan Crane, Clarisse Weischedel, and Max Wardetzky. 2013b. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Trans. Graphics* 32 (2013), 152:1–152:11.

David Day and Mark Taylor. 2007. A new 11 point degree 6 cubature formula for the triangle. *Sixth International Congress on Industrial Applied Mathematics (ICIAM07) and GAMM Annual Meeting* 7 (2007), 1022501–1022502.

Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan Barr. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *ACM SIGGRAPH Conf. Proc.* 317–324.

Gerhard Dziuk. 1988. *Finite elements for the Beltrami operator on arbitrary surfaces.* Springer, 142–155.
Raanan Fattal, Dani Lischinski, and Michael Werman. 2002. Gradient domain high dynamic range compression. *ACM Trans. Graphics* 21 (2002), 249–256.
Francisco González and Gustavo Patow. 2009. Continuity mapping for multi-chart textures. *ACM Trans. Graphics* 28 (2009), 109:1–109:8.
Paul Heckbert. 1993. Introduction to finite element methods. In *ACM SIGGRAPH 1993 Courses.*
Sylvain Lefebvre and Hugues Hoppe. 2006. Appearance-space texture synthesis. *ACM Trans. Graphics* 25 (2006), 541–548.
Anat Levin, Assaf Zomet, Shmuel Peleg, and Yair Weiss. 2003. Seamless image stitching in the gradient domain. In *European Conf. Computer Vision.* 377–389.
Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least Squares Conformal Maps for Automatic Texture Atlas Generation. *ACM Trans. Graph.* 21 (2002), 362–371.
Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven Gortler. 2008. A Local/Global Approach to Mesh Parameterization. In *Symposium on Geometry Processing.* 1495–1504.
Songrun Liu, Zachary Ferguson, Alec Jacobson, and Yotam Gingold. 2017. Seamless: Seam Erasure and Seam-aware Decoupling of Shape from Mesh Resolution. *ACM Trans. Graph.* 36 (2017), 216:1–216:15.
Lok Lui, Yalin Wang, and Tony Chan. 2005. Solving PDEs on Manifolds with Global Conformal Parameterization. In *Proc. Third Int. Conf. on Variational, Geometric, and Level Set Methods in Computer Vision.* 307–319.
Microsoft. 2018. UVAtlas: isochart texture atlasing . https://github.com/Microsoft/UVAtlas. (2018).
Stanley Osher and James Sethian. 1988. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *J. Computational Physics* 79 (1988), 12–49.
Jonathan Palacios and Eugene Zhang. 2011. Interactive Visualization of Rotational Symmetry Fields on Surfaces. *IEEE Trans. Visualization and Computer Graphics* 17 (2011), 947–955.
Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson image editing. *ACM Trans. Graphics* 22 (2003), 313–318.
Pietro Perona and Jitendra Malik. 1990. Scale-space and edge detection using anisotropic diffusion. *Trans. on Pattern Analysis and Machine Intelligence* 12 (1990), 629–639.
Cosmin Petra, Olaf Schenk, and Mihai Anitescu. 2014a. Real-time stochastic optimization of complex energy systems on high-performance computers. *Computing in Science & Engineering* 16, 5 (2014), 32–42.
Cosmin G. Petra, Olaf Schenk, Miles Lubin, and Klaus Gäertner. 2014b. An augmented incomplete factorization approach for computing the Schur complement in stochastic optimization. *SIAM J. on Scientific Computing* 36 (2014), C139–C162.
Ulrich Pinkall and Konrad Polthier. 1993. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics* 2, 15–36.
Roi Poranne, Marco Tarini, Sandro Huber, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Autocuts: Simultaneous Distortion and Cut Optimization for UV Mapping. *ACM Trans. Graph.* 36 (2017), 215:1–215:11.
Pedro Sander, Steven Gortler, John Snyder, and Hugues Hoppe. 2002. Signal-specialized Parametrization. In *Proc. 13th Eurographics Workshop on Rendering.* 87–98.
Pedro Sander, John Snyder, Steven Gortler, and Hugues Hoppe. 2001. Texture Mapping Progressive Meshes. In *Proc. 28th Annual Conf. on Computer Graphics and Interactive Techniques.* 409–416.
Alla Sheffer and John Hart. 2002. Seamster: Inconspicuous Low-distortion Texture Seam Layout. In *Proc. Conf. on Visualization '02.* 291–298.
Barry Smith, Petter Bjørstad, and William Gropp. 1996. *Domain decomposition: Parallel multilevel methods for elliptic partial differential equations.* Cambridge Univ. Press.
Jason Smith and Scott Schaefer. 2015. Bijective Parameterization with Free Boundaries. *ACM Trans. Graph.* 34 (2015), 70:1–70:9.
Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and Hand-Peter Seidel. 2004. Laplacian surface editing. In *Symposium on Geometry Processing.* 175–184.
Jos Stam. 2003. Flows on surfaces of arbitrary topology. *ACM Trans. Graphics (SIGGRAPH '03)* 22 (2003), 724–731.
Tolga Tasdizen, Ross Whitaker, Paul Burchard, and Stanley Osher. 2002. Geometric surface smoothing via anisotropic diffusion of normals. In *Conf. Visualization '02.* 125–132.
Gabriel Taubin. 1995. A signal processing approach to fair surface design. In *ACM SIGGRAPH Conf. Proc.* 351–358.
Christian Teitzel, Roberto Grosso, and Thomas Ertl. 1997. Line Integral Convolution on Triangulated Surfaces. In *Conf. World Society for Computer Graphics 1997.* 572–581.
Christian Weiss, Wolfgang Karl, Markus Kowarschik, and Ulrich Rüde. 1999. Memory characteristics of iterative methods. In *1999 ACM/IEEE Conf. Supercomputing.*
Tim Weyrich, Jia Deng, Connelly Barnes, Szymon Rusinkiewicz, and Adam Finkelstein. 2007. Digital bas-relief from 3D scenes. *ACM Trans. Graphics* 26 (2007), 32:1–32:7.
Andrew Witkin and Michael Kass. 1991. Reaction-diffusion textures. In *ACM SIGGRAPH Conf. Proc.* 299–308.
Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. 2004. Mesh editing with Poisson-based gradient field manipulation. *ACM Trans. Graphics* 23 (2004), 644–651.
Cem Yuksel. 2017. Mesh color textures. In *High Performance Graphics.* 17:1–17:11.
Eugene Zhang, Konstantin Mischaikow, and Greg Turk. 2005. Feature-based Surface Parameterization and Texture Mapping. *ACM Trans. Graph.* 24 (2005), 1–27.
Kun Zhou, John Synder, Baining Guo, and Heung-Yeung Shum. 2004. Iso-charts: Stretch-driven Mesh Parameterization Using Spectral Analysis. In *Symposium on Geometry Processing.* 45–54.

APPENDIX

CLAIM 1. *The unnormalized and normalized texel functions $\overline{\phi}_t$ and ϕ_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 $\hat{t} \in \widehat{T}$). 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 $n \in \mathcal{N}$, we reproduce the function Q:

$$Q(p) = \sum_{n \in \mathcal{N}} Q(n) \cdot Q_n(p), \qquad \forall p \in M.$$

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

$$\widetilde{Q}_{[n]} \equiv \sum_{m \in [n]} Q_m = Q_n.$$

Thus, for an interior texel $t \in \mathcal{T}$ we get:

$$\overline{\phi}_t(p) \equiv \sum_{\tilde{n} \in \widetilde{\mathcal{N}}} \left(\sum_{n \in \tilde{n}} B_t(n) \right) \cdot \widetilde{Q}_{\tilde{n}}(p) = \sum_{n \in \mathcal{N}} B_t(n) \cdot Q_n(p) = B_t(p),$$

so that $\overline{\phi}_t$ reproduces B_t. Since for an interior texel t we have

$$\phi_t \equiv \sum_{\tilde{n} \in \widetilde{\mathcal{N}}} \left(\frac{1}{|\tilde{n}|} \sum_{n \in \tilde{n}} B_t(n) \right) \cdot \widetilde{Q}_{\tilde{n}} = \sum_{\tilde{n} \in \widetilde{\mathcal{N}}} \left(\sum_{n \in \tilde{n}} B_t(n) \right) \cdot \widetilde{Q}_{\tilde{n}} \equiv \overline{\phi}_t,$$

the function ϕ_t also reproduces B_t. □

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

$$\sum_{t \in \mathcal{T}} \sum_{n \in \tilde{n}} B_t(n) = |\tilde{n}|.$$

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

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

□

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

$$\sum_{t \in \mathcal{T}} \phi_t(p) = 1, \qquad \forall p \in M.$$

PROOF. Let $\mathbf{B} \in \mathbb{R}^{|\widetilde{\mathcal{N}}| \times |\mathcal{T}|}$ be the matrix whose $(\tilde{n}, t)$-th entry is the coefficient of $\widetilde{Q}_{\tilde{n}}$ in the expression of ϕ_t:

$$\phi_t = \sum_{\tilde{n} \in \widetilde{\mathcal{N}}} \mathbf{B}_{\tilde{n}t} \cdot \widetilde{Q}_{\tilde{n}}.$$

Since the functions $\{\widetilde{Q}_{\tilde{n}}\}$ form a partition of unity on M, the condition that the functions $\{\phi_t\}$ form a partition of unity on M is equivalent to the condition that the sum of the $\{\phi_t\}$ is equal to the sum of the $\{\widetilde{Q}_{\tilde{n}}\}$. Or, equivalently, that the rows of $\mathbf{B}$ sum to one:

$$\sum_{t\in\mathcal{T}} \mathbf{B}_{\tilde{n}t} = 1.$$

Let $\overline{\mathbf{B}} \in \mathbb{R}^{|\widetilde{\mathcal{N}}|\times|\mathcal{T}|}$ be the matrix whose $(\tilde{n},t)$-th entry is the coefficient of $\widetilde{Q}_{\tilde{n}}$ in the expression of $\overline{\phi}_t$. By construction, we have:

$$\overline{\mathbf{B}}_{\tilde{n}t} \equiv \sum_{n\in\tilde{n}} B_t(n)\,.$$

We can define $\mathbf{B}$ by normalizing the entries of $\overline{\mathbf{B}}$ by their row-sums.

By Lemma 2, the $\tilde{n}$-th row-sum of $\overline{\mathbf{B}}$ is the cardinality of $\tilde{n}$:

$$\sum_{t\in\mathcal{T}} \overline{\mathbf{B}}_{\tilde{n},t} = \sum_{t\in\mathcal{T}}\sum_{n\in\tilde{n}} B_t(n) = |\tilde{n}|\,.$$

Thus, normalizing $\overline{\mathbf{B}}$ by its row-sums we can define

$$\mathbf{B}_{\tilde{n},t} \equiv \sum_{n\in\tilde{n}} \frac{1}{|\tilde{n}|}\cdot B_t(n)\,.$$

Or equivalently,

$$\phi_t \equiv \sum_{\tilde{n}\in\widetilde{\mathcal{N}}} \left(\frac{1}{|\tilde{n}|}\sum_{n\in\tilde{n}} B_t(n)\right)\cdot \widetilde{Q}_{\tilde{n}}.$$

While one could perform this normalization in other ways (e.g., by setting $\phi_t = \overline{\phi}_t/\sum_s \overline{\phi}_s$), normalizing by the row-sum is particularly simple and ensures that the functions $\{\phi_t\}$ are linear combinations of the $\{\widetilde{Q}_{\tilde{n}}\}$, e.g., piecewise quadratic. □

Name	Fig.	Resolution	Texels	Charts	V-cycle
Slick	1	1024 × 1024	734K	29	38
Filigree	1	1024 × 1024	701K	168	64
Camel	1	2048 × 2048	2837K	32	—
Girl	5	1024 × 1024	722K	36	39
Bimba	6	1024 × 1024	724K	1	23
Julius-1C	9	1018 × 1018	800K	1	23
Julius-4C	9	1089 × 1089	800K	4	30
Julius-28C	9	522 × 522	200K	28	14
Julius-28C	9	1055 × 1055	800K	28	39
Julius-28C	9	2127 × 2127	3200K	28	121
Julius-28C	9	4266 × 4266	12800K	28	409
Ballerina	14	1024 × 1024	740K	87	49
David head	15	1024 × 1024	749K	14	34
Mime	16	1024 × 1024	727K	35	41
Bunny	17	1024 × 1024	670K	9	29
Fertility	18	2048 × 2048	2770K	11	—

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.

Symbol	Summary description
$p,q\in M$	points in parameterization domain
$v,w\in T_pM$	tangent vectors at $p\in M$
$v^*,w^*\in T_p^*M$	cotangent vectors at $p\in M$
$\Phi: M\to\mathbb{R}^3$	surface parameterization
$\mu_p: T_pM\times T_pM\to\mathbb{R}$	2D Euclidean metric at $p\in M$
$g_p: T_pM\times T_pM\to\mathbb{R}$	immersion metric at $p\in M$
$h_p: T_pM\times T_pM\to\mathbb{R}$	Riemannian metric at $p\in M$
$t\in T$	triangle in initial triangulation of M
$c\in C$	cell in grid-polygonization of M
$\hat{t}\in\widehat{T}$	triangle in the grid-triangulation of M
$r,s,t\in\mathcal{T},\mathcal{T}^l$	texels (at level l)
$a,b\in\mathcal{A}$	adjacent texel pairs
$n\in\mathcal{N}$	vertices / edges in $\widehat{T}$
$\tilde{n}\in\widetilde{\mathcal{N}}$	$\mathcal{N}$ modulo seam-equivalence
$\phi,\psi\in\Omega^0(M)$	scalar functions
$\omega\in\Omega^1(M)$	cotangent vector field / 1-form
$\{B_t\}\subset\Omega^0(M)$	bilinear basis
$\{Q_n\}\subset\Omega^0(M)$	quadratic Lagrange basis
$\{\widetilde{Q}_{\tilde{n}}\}\subset\Omega^0(M)$	seam-continuous quadratic Lagrange basis
$\{\overline{\phi}_t\}\subset\Omega^0(M)$	bilinear-like basis
$\{\phi_t\}\subset\Omega^0(M)$	like $\{\overline{\phi}_t\}$ but forming a partition of unity
$\{\omega_a\}\subset\Omega^1(M)$	1-form basis
$d:\Omega^0(M)\to\Omega^1(M)$	differential operator
$\mathbf{d}\in\mathbb{R}^{\lvert\mathcal{A}\rvert\times\lvert\mathcal{T}\rvert}$	differential matrix
$\star_h^0\in\mathbb{R}^{\lvert\mathcal{T}\rvert\times\lvert\mathcal{T}\rvert}$	Hodge 0-star matrix
$\star_h^1\in\mathbb{R}^{\lvert\mathcal{A}\rvert\times\lvert\mathcal{A}\rvert}$	Hodge 1-star matrix
$\mathbf{M}_h\in\mathbb{R}^{\lvert\mathcal{T}\rvert\times\lvert\mathcal{T}\rvert}$	mass matrix
$\mathbf{S}_h\in\mathbb{R}^{\lvert\mathcal{T}\rvert\times\lvert\mathcal{T}\rvert}$	stiffness matrix
$\mathrm{mass}_h:\Omega^0(M)\to\mathbb{R}^{\lvert\mathcal{T}\rvert}$	discrete mass operator
$\mathrm{div}_h:\Omega^1(M)\to\mathbb{R}^{\lvert\mathcal{T}\rvert}$	discrete divergence operator
$\mathbf{P}^l\in\mathbb{R}^{\lvert\mathcal{T}^{l+1}\rvert\times\lvert\mathcal{T}^l\rvert}$	prolongation matrix
$\mathbf{R}^l\in\mathbb{R}^{\lvert\mathcal{T}^l\rvert\times\lvert\mathcal{T}^{l+1}\rvert}$	restriction matrix
$\vec{x},\vec{y}\in\mathbb{R}^{\lvert\mathcal{T}\rvert}$	0-form coefficients
$\vec{z}\in\mathbb{R}^{\lvert\mathcal{A}\rvert}$	1-form coefficients
$\mathrm{E}:\Omega^0(M)\to\mathbb{R}$	gradient-domain energy
$\mathrm{E}_C:\Omega^0(M)\to\mathbb{R}$	seam continuity energy
$\alpha\in\mathbb{R}^{\geq 0}$	screening weight
$\beta\in\mathbb{R}^{\geq 0}$	gradient modulation
$l\in\{0,\ldots,L\}$	level in the multigrid hierarchy
$n\in\mathbb{N}$	Gauss-Seidel iterations

Table 3. Summary of notation.