Sharpening and Sparsifying with Surface Hessians

DYLAN ROWE, University of Southern California, USA
ALEC JACOBSON, University of Toronto, Canada
ODED STEIN, University of Southern California, USA

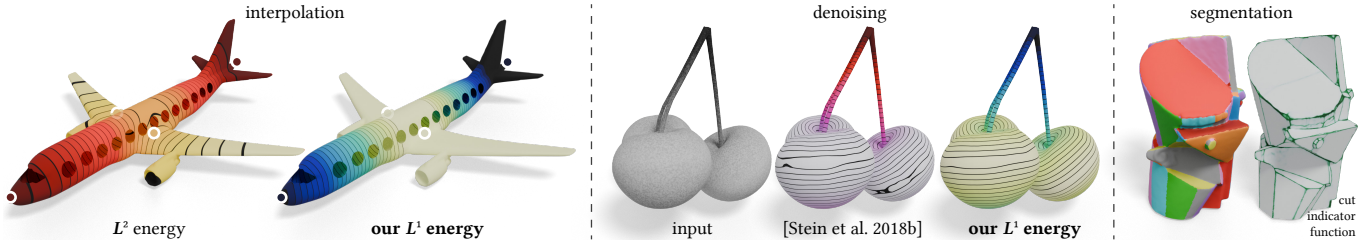


Fig. 1. Our intrinsic L^1 smoothness energy constructs piecewise smooth functions. Unlike L^2 energies, which fit a smooth function without any ridges to solve an interpolation problem (and thus have to hallucinate a saddle), we construct a piecewise flat function *(left)*. When denoising, our discretization produces clean isolines compared to previous, non-intrinsic approaches *(center)*. Our L^1 energy can construct sparse cut indicator functions to segment meshes *(right)*.

The L^1 Hessian energy measures the norm of the Hessian of a function on a surface (and *not* the squared norm, as is common with many geometry applications that employ L^2). Its minimizers tend to be locally linear with a sparse set of curved ridges. We introduce a fully-intrinsic discretization of this energy for triangle meshes and show that it can be optimized using off-the-shelf conic program solvers. We apply it to stylization, denoising, interpolation, hole-filling, and segmentation tasks. Our L^1 approach exhibits multiple important differences from its more-familiar L^2 counterpart: it preserves ridge-like features in the input, it naturally incorporates a flatness prior for reconstruction, and, at its extreme, it distills its input to an abstract, angular form.

CCS Concepts: • **Computing methodologies** → *Computer graphics*; *Mesh geometry models*; • **Mathematics of computing** → **Partial differential equations**.

ACM Reference Format:
Dylan Rowe, Alec Jacobson, and Oded Stein. 2024. Sharpening and Sparsifying with Surface Hessians. In *SIGGRAPH Asia 2024 Conference Papers (SA Conference Papers '24), December 3–6, 2024, Tokyo, Japan.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3680528.3687666

1 INTRODUCTION

Many tasks in geometry processing can be formulated as an optimization problem involving a smoothness energy on a surface Ω, such as interpolation, denoising, or hole-filling. The optimization problem's goal is to find the smoothest function fulfilling some constraint. These smoothness energies are usually L^2 derivative energies, i.e., they integrate the square of derivatives of a function, such as the Laplacian and Hessian energies [Stein et al. 2018b]

$$E_{\Delta^2}(u) = \frac{1}{2}\int_\Omega |\Delta u|^2 \, dx\,, \qquad E_{\mathrm{H}^2}(u) = \frac{1}{2}\int_\Omega \|\mathrm{H}_u\|^2 \, dx\,, \quad (1)$$

where $\|\cdot\|$ is the Euclidean or Frobenius norm.

These L^2 energies will find the smoothest overall solution to a given problem, but will fail when the goal is to obtain a *piecewise smooth* function, since the ridges between smooth regions are heavily penalized by the square integral (see Fig. 2).

The L^1 norm, on the other hand, is a well-known tool for sparsifying functions (see, e.g., [Bruckstein et al. 2009]). An L^1 optimization will reduce the value of a function overall, but allow for spikes in measure-zero regions of the surface, leading to the concentration of features on a few ridges (curves with gradient discontinuities in the normal direction). This behavior can be used to *sharpen* and *sparsify* functions on surfaces. Not every L^2 smoothness energy can be used as an L^1 energy, however. The vertex-based discretization of the Laplacian energy E_{Δ^2}, for example, leads to noisy results if naively transformed into an L^1 energy by simply changing the exponent of the integral [He and Schaefer 2013].

We present an L^1 smoothness energy based on the surface Hessian of a function that generalizes the *flat* L^1 Hessian energy of Stein et al. [2018b] to *curved* surfaces,

$$E_{\mathrm{H}}(u) = \frac{1}{2}\int_\Omega \|\mathrm{H}_u\| \, dx\,. \quad (2)$$

Our L^1 Hessian energy creates smooth results without influence from the boundary of the domain, like its L^2 counterpart, but it constructs *piecewise smooth* results. This allows it to be used in a variety of applications where the global smoothness of L^2 energies is not desirable, such as to interpolate/denoise data that is known to be only piecewise smooth (like distance functions), to fill holes of shapes with sharp features, and to sparsify functions on surfaces. We also introduce an intrinsic discretization of the L^1 Hessian energy for triangle meshes and its accompanying optimization problem, as existing discretizations are either not intrinsic or not L^1. Our discrete operator depends only on the edge lengths of the triangles, and the resulting minimization problem can be efficiently solved by any conic solver.

Our energy can be plugged into any constrained quadratic geometry processing optimization problem to encourage piecewise

SA Conference Papers '24, December 3–6, 2024, Tokyo, Japan

ACM ISBN 979-8-4007-1131-2/24/12.
https://doi.org/10.1145/3680528.3687666

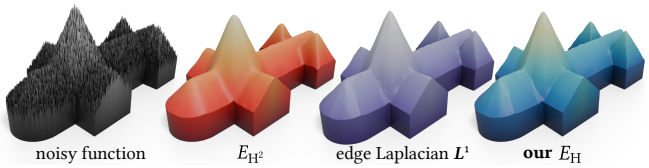


Fig. 2. Denoising a heightfield reminiscent of Stein et al. [2018b]'s Fig. 16. The L^2 energy E_{H^2} smooths out sharp ridges. An L^1 energy based on an anisotropic edge Laplacian [He and Schaefer 2013] leads to slightly better results. The L^1 Hessian energy E_{H} reconstructs a cathedral with sharp ridges.

smooth results with sparse, sharp ridge features. In this article we apply it to a wide variety of geometry processing tasks: (1) scattered data interpolation, (2) denoising, (3) hole-filling, (4) piecewise flat stylization, (5) mesh segmentation, and (6) automatic drawing of diffusion curves. Furthermore, we provide a generalized analysis of our operator on surfaces, analyze its low-frequency modes, and showcase how its minimizers differ from those of L^2 energies. Our L^1 Hessian energy is a useful addition to any general-purpose geometry processing toolbox alongside existing L^2 smoothness energies.

2 RELATED WORK

Our work, the discussion of an L^1 Hessian smoothness energy, its discretization, and its applications, is related to multiple old research topics in image and geometry processing: smoothness energies, L^1 norms, and the discretization of non-scalar functions on surfaces.

2.1 Smoothness energies

Common smoothness energies include the squared L^2 norm (or square integral) of the gradient ($\frac{1}{2}\int_\Omega \|\nabla u\|^2\, dx$) or the Laplacian ($\frac{1}{2}\int_\Omega |\Delta u|^2\, dx$) of a function. Smoothness energies have many applications, e.g., in surface fairing [Desbrun et al. 1999], scattered data interpolation [Jacobson et al. 2012], data smoothing [Weinkauf et al. 2010], cartoonization [Sýkora et al. 2014], and computing skinning weights for animation [Baran and Popović 2007]. Higher-order smoothness energies (such as those featuring the Laplacian) are preferred over low-order smoothness energies (featuring the gradient) in interpolation and smoothness applications where a very smooth result without spiky artifacts is desired, for example at constraints [Jacobson et al. 2011], or in surface flows [Crane et al. 2013]. Such higher-order energies require a choice of boundary condition. Beyond the popular zero Neumann conditions, one can choose, e.g., natural Hessian boundary conditions [Stein et al. 2018b, 2020a] arising from the minimization of a different higher-order energy measuring the squared L^2 norm of the Hessian of a function ($\frac{1}{2}\int_\Omega \|\mathrm{H}_u\|^2\, dx$). Our smoothness energy is a generalization of the Hessian smoothness energy of Stein et al. [2018b].

2.2 L^1 norms

The work of Stein et al. [2018b] not only features an L^2 Hessian energy, but also an L^1 Hessian energy. L^1 norms measure the integral of the norm of a function instead of the integral of its square, and they tend to encourage sharp ridges and sparse structures. Their

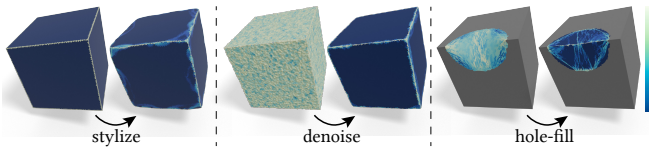


Fig. 3. Shapes with sharp ridges are preferred by our L^1 Hessian energy E_{H}. When stylizing *(left)*, the cube is preserved; when denoising *(center)*, the cube is restored; and when hole-filling *(right)*, the cube's ridge is reconstructed.

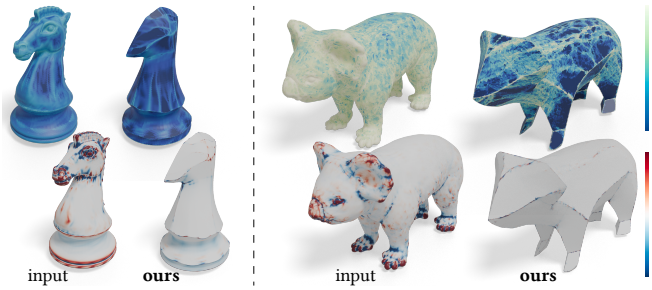


Fig. 4. E_{H} can be used in a flow to stylize a surface to appear piecewise flat. As the norm of the Hessian decreases (*top row*), the Gaussian curvature (visualized by the scaled angle defect, *bottom row*) concentrates on ridges.

affinity for ridges makes them a popular tool in image processing, where they preserve sharp image features during operations such as denoising better than L^2 approaches [Bergounioux and Piffet 2010; Chambolle and Lions 1997; Chan et al. 2007; Fu et al. 2006; Lysaker et al. 2003; Lysaker and Tai 2006; Steidl 2006; Steidl et al. 2005; You and Kaveh 2000; Yuan et al. 2009]. In geometry processing, L^1 norms have been employed, among other things, for surface reconstruction [Avron et al. 2010; Lai et al. 2013], for surface registration [Achenbach et al. 2015], to fracture objects [Sellán et al. 2023], to construct polycubes [Huang et al. 2014; Li et al. 2021; Zhang et al. 2020], to encourage sparsity in barycentric coordinates [Zhang et al. 2014], and to denoise surfaces [He and Schaefer 2013; Wu et al. 2015]. Of special interest is the recent work of Liu et al. [2019] who employ L^1 norms of Laplacians that are, like our method, a generalization of image processing L^1 smoothing approaches to triangle meshes. Additionally, Bronstein et al. [2016] propose a higher-order discretization and optimization procedure for the continuous L^1 norm on manifolds.

2.3 Discretizing vectors and matrices on surfaces

The Hessian is a matrix, and cannot be intrinsically discretized using usual scalar methods like Lagrangian finite elements (which is why the discretization of [Stein et al. 2018b] using coordinate-wise scalar functions is not intrinsic). There are many ways to discretize differentiable vector and matrix fields intrinsically on triangle meshes. We extend an approach of Grinspun et al. [2006, (4)] to L^1 optimization, where vectors and matrices are based on faces, and the differentiation of a vector field happens over edges. Alternative approaches include the application of discrete exterior calculus [de Goes et al. 2014; Fisher et al. 2007], finite elements [Boksebeld and Vaxman

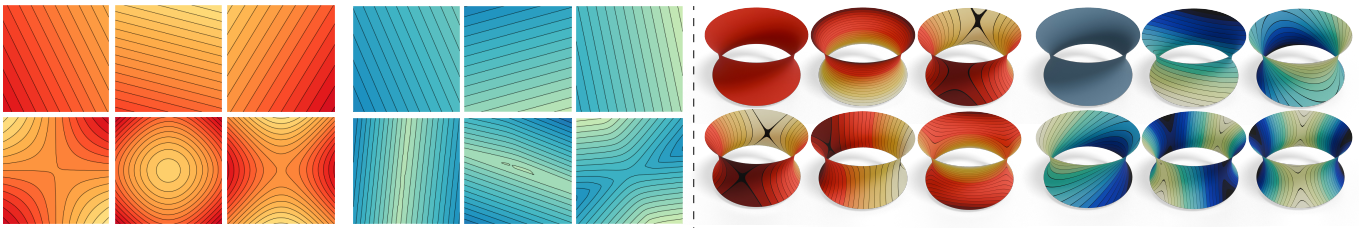


Fig. 5. The lowest six eigenmodes of E_{H^2} and E_{H} on a flat shape *(left)* and a curved surface *(right)*. While linear functions are the lowest modes for both flat examples, the L^1 energy exhibits sharp ridges that are not present in the L^2 modes.

2022; Djerbetian 2016; Knöppel et al. 2013; Stein et al. 2020b], subdivision surfaces [Custers and Vaxman 2020; de Goes et al. 2016], and other discretization strategies [Knöppel et al. 2015; Liu et al. 2016; Sharp et al. 2019b]. Walker [2024] approximates the shape operator on triangle meshes using a Hellan–Herrmann–Johnson-element-based discretization of the surface Hessian.

3 BACKGROUND

Our domain is a smooth two-dimensional surface Ω. The gradient of a scalar function u on Ω, ∇u, is a vector field on Ω. The covariant derivative of a vector field $\mathbf{v}$ on Ω, $\nabla \mathbf{v}$, is a $(1,1)$ tensor field on Ω. This is a generalization of a matrix field to curved surfaces: A $(1,1)$ tensor can be interpreted as a map that takes a tangent vector to another tangent vector [Petersen 2006]. The *Hessian* of a scalar function u is the covariant derivative of its gradient [Petersen 2006],

$$\mathrm{H}_u = \nabla\nabla u \,, \tag{3}$$

and is a symmetric tensor when written in an orthonormal basis with respect to the Riemannian metric g.

This Hessian is the curved surface analog of the well-known matrix of partial derivatives in $\mathbb{R}^2$. For orthonormal[1] coordinates X, Y on Ω, the Hessian can be written as

$$\mathrm{H}_u(\mathbf{v}) \cdot \mathbf{w} = \mathbf{w}^\mathsf{T} \begin{pmatrix} \partial_X \partial_X u & \partial_Y \partial_X u \\ \partial_X \partial_Y u & \partial_Y \partial_Y u \end{pmatrix} \mathbf{v} \,. \tag{4}$$

The metric-induced norm of $(1,1)$ tensors on Ω is defined via the trace. For a tensor S,

$$\|S\| = \sqrt{\mathrm{trace}\,(S^* S)} \,, \tag{5}$$

where S^* is the adjoint of the tensor S with respect to the metric g. For matrices, this is also known as the Frobenius norm,

$$\left\| \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} \right\|_{\mathrm{F}} = \sqrt{S_{11}^2 + S_{12}^2 + S_{21}^2 + S_{22}^2} \,. \tag{6}$$

The Hessian H_u measures all second-order variation of a function, and its norm $\|\mathrm{H}_u\|$ quantifies this change. Unlike the Laplacian Δu, the norm of the Hessian detects *all* second-order variation: E.g., $|\Delta(x^2 - y^2)| = 0$, but $\|\mathrm{H}_{x^2-y^2}\| = 2\sqrt{2}$.

[1]For non-orthonormal coordinates, one might have to consider additional cross-terms.

3.1 L^p norms

The L^p norm of a function u on Ω is defined via the integral

$$\|u\|_p = \left(\int_\Omega \|u(x)\|^p \, dx \right)^{\frac{1}{p}} \,, \tag{7}$$

where the norm $\|u(x)\|$ inside the integral is the absolute value of a scalar, or the metric-induced norm of vectors and tensors.

The L^1 and L^2 norms have a variety of interesting properties relevant to optimization applications. It holds true by Hölder's inequality that, on compact domains,

$$\|u\|_1 \leq \sqrt{\mathrm{area}(\Omega)} \|u\|_2 \,. \tag{8}$$

This means that, in a sense, the L^1 norm is more permissive than the L^2 norm: Functions that have a small L^1 norm can have a large L^2 norm, but not the other way around. A good example of this on $(0,1) \subseteq \mathbb{R}$ is the function $u(x) = x^{-\frac{1}{2}}$:

$$\|u\|_1 = \int_0^1 x^{-\frac{1}{2}} \, dx = 2 \,, \qquad \|u\|_2 = \left(\int_0^1 \frac{1}{x} \, dx \right)^{\frac{1}{2}} = \infty \,. \tag{9}$$

This means that L^2 norms will discourage functions that have even very small unbounded segments, while L^1 norms will allow functions that are unbounded on small parts of the domain. Measuring the L^2 norm of the second derivative of a function approaching a simple ridge on Ω (which is unbounded) produces a massive penalty on that ridge, and the optimization results in a rounding-off of that ridge immediately. L^1 does not penalize unbounded functions on sparse line segments as much, and will balance them with other parts of the function, thus optimizing the L^1 norm will preserve and even encourage the formation of sparse ridges (see Fig. 1). Hölder's inequality holds for subsets of Ω as well, suggesting that minimizers of the L^1 norm can be upper bounded by a local area-reweighing of the L^2 norm which permits large norm values as long as they only exist over small-area regions of the domain.

Furthermore, the L^1 norm is also well-known as a tool for encouraging sparsity when it is used to approximate an actual sparsity norm that measures the area where a function is nonzero. These norms are difficult to approximate, but it can be shown that in some cases solving an L^1 optimization problem will also produce sparse results [Candès et al. 2008].

Other uses of the term L^1. Some use the term L^1 to describe the l^1 norm of vectors. In contrast to the Euclidean norm $\left\|\binom{x}{y}\right\| = \sqrt{x^2 + y^2}$, the l^1 vector norm is $|x| + |y|$. We never use the l^1 norm

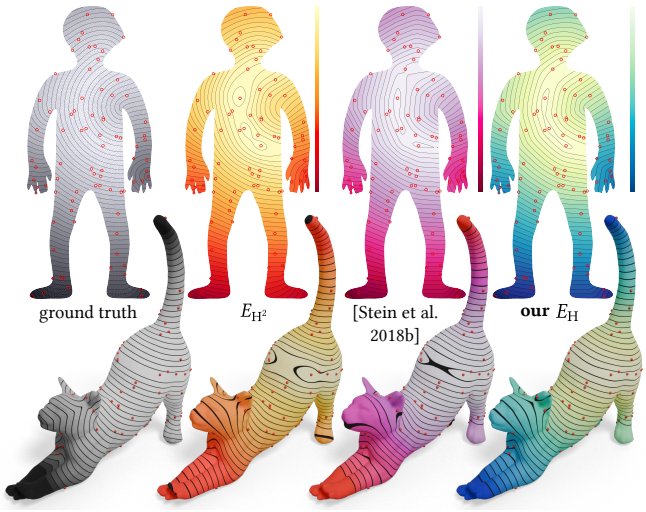


Fig. 6. A ground truth function is sampled at the red points *(left)*. These samples are then used for scattered data interpolation. The L^2 energy smooths out the sharp ridges of the ground truth *(center left)*, while Stein et al. [2018b] shows axis alignment artifacts. Our L^1 energy reconstructs the sharp ridge.

on any pointwise vectors or matrices in this article, and we only use the terms L^1 and L^2 to refer to integrated quantities (or discretized quantities that are summed over a mesh). Some refer to integrals of the l^1 norm of vectors as $L^{1,1}$ norms, and integrals of the Euclidean norm of vectors as $L^{2,1}$ norms. These terms are not relevant to us, as we only ever encounter the integrals of Euclidean norms (which we call L^1) and of squared Euclidean norms (which we call L^2).

3.2 Smoothness energies

Smoothness energies quantify the derivatives of a function on Ω as a measure of its smoothness: The smaller the derivatives are, the smoother a function is. While the first-order Dirichlet energy $\frac{1}{2}\int_\Omega \|\nabla u\|^2 \, dx$ is popular for many applications, we focus on higher-order smoothness energies, as first-order energies have a variety of drawbacks that make them unsuitable in combination with point constraints [Jacobson et al. 2011, Fig. 9], such as those used in scattered data interpolation.

The two smoothness energies most relevant to our discussion are the two higher-order energies of (1): The *Laplacian energy* E_{Δ^2}, which measures the squared integrated Laplacian of a function, and the squared *Hessian energy* E_{H^2}, which measures the squared integrated Hessian norm of a function.

E_{Δ^2} and E_{H^2} originate in the study of classical mechanics. The perpendicular displacement u from the rest plane in a physical model of a bending plate minimizes a smoothness energy subject to boundary constraints [Courant and Hilbert 1989]. In this model, fixing the plate to a particular location corresponds to the Dirichlet boundary condition, and applying a clamp at the boundary additionally applies a Neumann boundary condition. E_{Δ^2} and E_{H^2} display similar behavior, but differ in their natural boundary conditions – the conditions that apply at the boundary when no further constraints are applied to an optimization problem. The natural boundary conditions of the

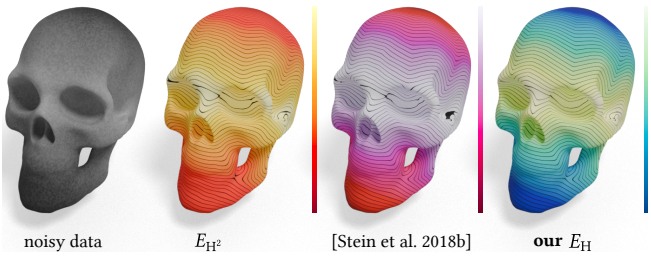


Fig. 7. Denoising scalar data on a surface *(left)*. The L^2 energy *(center left)* produces a smooth function with curved isolines. Stein et al. [2018b]'s L^1 energy *(center right)* produces slightly sharper isolines, but no corners. Our L^1 energy *(right)*, due to its intrinsicness, produces a sharp ridge with a corner.

Hessian L^2 energy correspond to no physical restraints at all, while the natural boundary conditions of the Laplacian energy result in noisy artifacts at the boundary [Stein et al. 2018b].

Relation to partial differential equations. It can be shown that minimizers of E_{Δ^2} and E_{H^2} solve the biharmonic equation $\Delta^2 u = 0$ on Ω, although the Hessian energy needs to be augmented with an additional curvature term on curved surfaces [Stein et al. 2020a].

4 THE L^1 HESSIAN ENERGY

In order to capture the benefits of L^1 energies (sharpening and sparsifying behavior) and apply them to smoothness energies, we employ an L^1 *Hessian energy*:

$$\boxed{E_{\mathrm{H}}(u) = \int_\Omega \|\mathrm{H}_u\| \, dx} \tag{10}$$

$E_{\mathrm{H}}(u)$ can be used as a smoothness energy in any geometry processing application that the L^2 energies can, but it will display a different behavior: instead of encouraging the *overall* smoothness of a function, it produces functions that are piecewise smooth, with nonsmooth parts concentrated on isolated sparse ridges (see Fig. 2). The energy favors sharp ridges and other sharp features. Fig. 3 shows that the cube acts like a target point of the energy – the energy preserves its shape. In Fig. 5 we see that the lowest-order modes of E_{H} feature sharp creases.

Previous works [Achenbach et al. 2015; He and Schaefer 2013] suggest that sparsity-encouraging norms provide stabler results when applied to anisotropic energies, rather than isotropic ones. For instance, an L^1 version of the vertex-based Laplacian energy naturally leads to noisy spiky results all over the domain; He and Schaefer [2013] combat this by approximately minimizing an L^0 energy based on the (anisotropic) *edge* cotangent Laplacian. Fig. 2 shows the result of minimizing the ℓ_1 norm of their edge Laplacian; our energy may benefit from similar stability due to the inherent anisotropy of the surface Hessian.

We can derive a partial differential equation (PDE) that is solved by minimizers of our L^1 Hessian energy E_{H} with the principle of variation. We vary u in the direction of a test function η:

$$\frac{d}{d\varepsilon}\int_\Omega \| \, \mathrm{H}_{u+\varepsilon\eta} \, \| \, dx \, \Big|_{\varepsilon=0} = 0 \, . \tag{11}$$

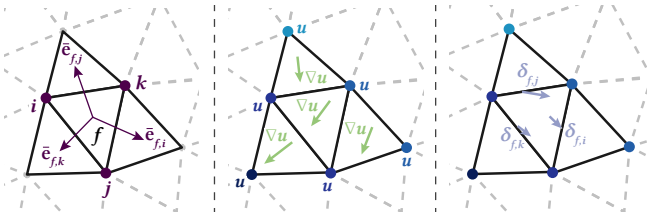


Fig. 8. The dual halfedges $\bar{\mathbf{e}}_{f,\cdot}$ connect the barycenter of the face f with the barycenter of its neighbors. u is a piecewise linear per-vertex function, and ∇u is a piecewise constant per-face vector *(center)*. The per-halfedge edge vector derivatives $\delta_{f,\cdot}$ are intrinsic to each face *(right)*.

Applying the chain rule and evaluating at $\varepsilon = 0$ gives

$$\int_\Omega \frac{1}{2}(\| \mathrm{H}_u \|^2)^{-\frac{1}{2}} \cdot 2\langle \mathrm{H}_u, \mathrm{H}_\eta \rangle \, dx = 0 \,, \tag{12}$$

where $\langle A, B \rangle = \mathrm{trace}(A^\intercal B)$ is the Frobenius inner product. Using integration by parts and ignoring boundary terms for now (see Supplemental Material B for further details), this is equivalent to

$$\int_\Omega \left(\mathrm{H}^* \frac{\mathrm{H}_u}{\| \mathrm{H}_u \|} \right) \eta \, dx = 0 \,, \tag{13}$$

where $\mathrm{H}^* = \nabla^* \nabla^*$ is the adjoint to the Hessian operator defined using the adjoint to the covariant derivative ∇^* with respect to the metric pairing [Petersen 2006]. Since η is an arbitrary test function, the only functions integrated over a region satisfying the above equation are those in which

$$\mathrm{H}^* \frac{\mathrm{H}_u}{\| \mathrm{H}_u \|} = 0 \,. \tag{14}$$

(14) is to our L^1 Hessian energy what the biharmonic equation $\Delta^2 u = 0$ is to the L^2 Hessian energy.

5 DISCRETIZATION

5.1 Computing the energy

Discretizing the Hessian L^1 energy E_H is not straightforward. Stein et al. [2018b] use mixed finite elements to take coordinate-wise partial second derivatives. We do not adopt this approach, as it does not result in an *intrinsic* operator, an operator which depends only on the edge lengths of each triangle, and not on the coordinate positions of the vertices. Intrinsic operators have a variety of advantages. Firstly, the norm of the Hessian is a property which a discretization should strive to preserve; secondly, intrinsic operators are preserved under isometric deformations of a shape; and thirdly, intrinsic operators allow the use of automatic intrinsic remeshing such as intrinsic Delaunay triangulation [Sharp et al. 2019a].

We present a discretization of the Hessian tailor-made for L^1 optimization. Let $\mathbf{u}$ be a per-vertex scalar function on a triangle mesh M with n vertices. $\mathbf{u}$ can be interpreted as a piecewise linear function on each face by barycentric interpolation. This allows us to directly and exactly compute $\nabla\mathbf{u}$, the gradient of $\mathbf{u}$, an intrinsic vector field that is constant per-face. Since the vector $(\nabla\mathbf{u})_f$ is tangent to the face f, it can be written using an intrinsic orthogonal basis for f defined by the first halfedge for each face in its face-list (see Fig. 8 *(center)* and Supplemental Material A).

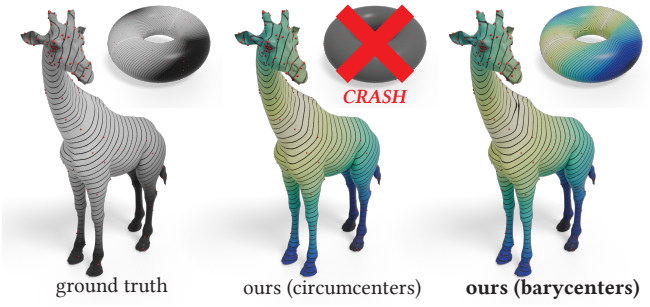
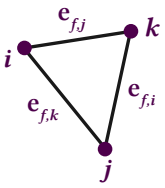


Fig. 9. Scattered data interpolation using different choices of triangle center. While the circumcenter variant of our method (*center*) gives results more similar to the ground truth (*left*), it may crash on meshes with zero dual edge lengths (*top center*), unlike the barycenter variant (*right*).



To discretize the covariant derivative in $\nabla\nabla u$ we start by computing the difference between the gradient vectors on each halfedge. We adopt the convention that a halfedge $\mathbf{e}_{f,i}$ of a face f is identified by the index i of the vertex it is opposite to. Let $\mathcal{N}_{f,i}$ be the neighbor of face f across halfedge i. We then define the difference of gradient vectors across the halfedge i as

$$\delta_{f,i}(\mathbf{u}) = I_{f,i}\,(\nabla\mathbf{u})_{\mathcal{N}_{f,i}} - (\nabla\mathbf{u})_f \,, \tag{15}$$

where the *hinge map* $I_{f,i}$ embeds a vector from the intrinsic coordinate system of face $\mathcal{N}_{f,i}$ to the intrinsic coordinate system of face f, defined by isometrically mapping the face $\mathcal{N}_{f,i}$ into the plane of f while keeping the shared edge fixed (see Fig. 8 *(right)*).

We now use the per-halfedge difference $\delta_{f,i}$ to compute finite-difference derivatives over each dual halfedge. Let $\bar{\mathbf{e}}_{f,i}$ be the dual vector to halfedge i in face f that goes from the center of f to the center of its neighbor $\mathcal{N}_{f,i}$ (since we later divide by the length of this, circumcenters could cause divisions by zero; we default to barycenters). Similar to how Knöppel et al. [2015] measure the transport of vectors along an edge from vertex to vertex, we measure the transport of vectors across an edge from face to face to compute our derivative over the halfedge:

$$d_{f,i}(\mathbf{u}) = \frac{1}{\|\bar{\mathbf{e}}_{f,i}\|^2} \delta_{f,i} \cdot \bar{\mathbf{e}}_{f,i} \,. \tag{16}$$

These per-halfedge derivatives are then accumulated like in the triangle-averaged operator of Grinspun et al. [2006]. Let $\mathcal{E}(f)$ be the set of halfedges of face f. Then

$$\mathcal{H}_f(\mathbf{u}) = \sum_{i \in \mathcal{E}(f)} \frac{d_{f,i}}{\|\bar{\mathbf{e}}_{f,i}\|^2} \, \bar{\mathbf{e}}_{f,i} \otimes \bar{\mathbf{e}}_{f,i} \,. \tag{17}$$

where $\otimes$ denotes the tensor product of vectors. We ignore boundary halfedges during accumulation in order to not make any explicit assumptions on boundary conditions.

$\mathcal{H}_f(\mathbf{u})$ is a symmetric 2×2 matrix *(by construction)* in the intrinsic coordinate system of f that corresponds to the Hessian H_u. We now need to take its Frobenius norm and integrate it over the entire

	Ours time	L^2 time	[Stein et al. 2018b] L^1 time	$\lvert V\rvert$
Octopus (2D)	0.367s	0.052s	0.343s	9142
Person (2D)	1.268s	0.087s	5.252s	9551
Frog (3D, original)	0.515s	–	–	12491
Cat (3D)	4.617s	0.261s	309.522s	31790
Frog (3D, 1 subd.)	2.142s	–	–	49964
Cathedral (2D)	6.168s	0.608s	9.337s	59833
Skull (3D)	26.374s	1.202s	26.615s	82494
Cherries (3D)	36.329s	10.505s	116.680s	126303
Frog (3D, 2 subd.)	9.645s	–	–	199856
Triceratops (3D)	20.639s	3.723s	42.133s	387664

Table 1. Runtimes for our method on a few of the examples in this article.

triangle mesh.

$$\mathbf{E}_{\mathrm{H}}(\mathbf{u}) = \sum_{\text{faces } f} A_f \|\mathcal{H}_f(u)\|_{\mathrm{F}}\,, \tag{18}$$

where A_f is the area of the face f, and $\|B\|_{\mathrm{F}} = \sqrt{\sum_i \sum_j B_{ij}^2}$ is the Frobenius norm of a matrix.

5.2 Representation by matrices

Given a scalar function represented as a vector of per-vertex values $\mathbf{u}$, we can explicitly write our surface Hessian as $\mathbf{Hu} = \mathbf{M}_F\mathbf{TCGu}$, where the matrices $\mathbf{G}$, $\mathbf{C}$, $\mathbf{T}$, and $\mathbf{M}_F$ represent taking the intrinsic gradient on faces, the covariant derivative on halfedges, the tensor product operation on faces, and the scaling by face areas respectively.

The intrinsic gradient on face f with vertices i, j, k and tip angle θ_i at vertex i takes the explicit form

$$(\mathbf{Gu})_f = \begin{pmatrix} \frac{u_j - u_i}{l_{ij}} & \frac{(l_{ki}\cos\theta_i - l_{ij})u_i - (l_{ki}\cos\theta_i)u_j + l_{ij}u_k}{l_{ij}l_{ki}\sin\theta_i} \end{pmatrix}^{\mathsf{T}}, \tag{19}$$

where the intrinsic basis has unit-length $\hat{\mathbf{x}}$ pointing along edge ij.

The hinge map covariant derivative $\mathbf{C}$ takes vectors on faces to vectors on halfedges; for halfedge h_{ij} on face f_i corresponding to the edge shared between f_i and f_j, $\mathbf{C}$ has entries

$$(\mathbf{C})_{h_{ij},f_j} = \begin{bmatrix}\mathbf{R}(\theta)\end{bmatrix}, \quad (\mathbf{C})_{h_{ij},f_i} = \begin{bmatrix}-\mathbf{I}_{2\times 2}\end{bmatrix}, \tag{20}$$

where $\mathbf{R}(\theta)$ rotates the basis at face f_i onto the basis at face f_j.

The tensor product matrix $\mathbf{T}$ takes halfedge vectors to the matrix corresponding to their scaled summed tensor products on faces; its entries can be found in Supplemental Material A. $\mathbf{M}_F$ scales the Hessian entries by corresponding face areas. The discrete L^1 Hessian energy is evaluated by taking the ℓ^2 norm of the four Hessian entries corresponding to each face, and then summing over all faces.

5.3 Optimization

An objective involving the L^1 Hessian energy is more difficult to optimize than the usual L^2 objectives because of the presence of square roots in each term of the sum (18). L^2 energies consist of a sum of squares, which makes the optimization a simple quadratic program whose solution can be found by finding a critical point – this only requires the solution of a single linear equation. The square roots in the L^1 Hessian energy make this impossible.

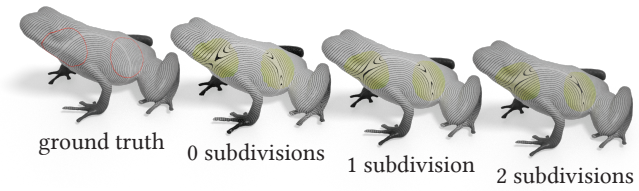


Fig. 10. Scalar hole filling with our method on a surface with increasing mesh resolutions. Higher-subdivision results visually match the ground truth better.

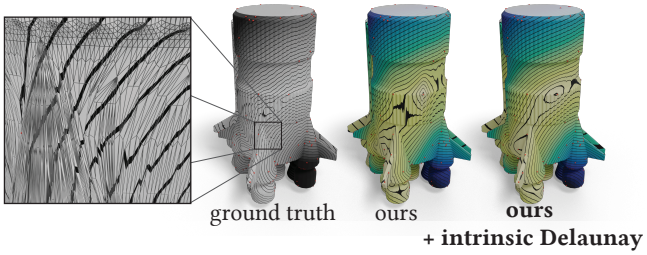


Fig. 11. Scattered data interpolation with and without intrinsic Delaunay remeshing. Our method without preprocessing (*center*) is sensitive to bad triangles (*left, inset*). Since our operator is intrinsic, we can mitigate this issue by preprocessing with intrinsic Delaunay remeshing (*right*).

However, $\mathcal{H}_f(u)$ is a linear function of u, so the resulting $\mathbf{E}_{\mathrm{H}}(\mathbf{u})$ is a sum of norms of linear functions. As a result, the L^1 optimization problem can be transformed into a conic programming problem that is amenable to standard black-box solvers. For this, we employ an approach similar to Stein et al. [2018b, Appendix C] that can efficiently solve problems of the following form:

$$\operatorname*{argmin}_{\mathbf{u}} \mathbf{E}_{\mathrm{H}}(\mathbf{u}) + \alpha\|\mathbf{u} - \mathbf{u}_0\|^2 \qquad \text{s.t. } \mathbf{Au} = \mathbf{b}\,. \tag{21}$$

See Supplemental Material D for details.

To understand the kinds of functions favored by E_{H}, we compute its compressed vibration modes [Brandt and Hildebrandt 2017], an analogon to eigenfunctions (see Fig. 5). We observe that the lowest modes of the L^1 energy E_{H} are characterized by sharp ridges, while those of E_{H^2} are smooth with saddle-like features. See Supplemental Material C for the algorithm used to compute the modes.

6 IMPLEMENTATION

This article was implemented in Python using NumPy [Harris et al. 2020] for linear algebra, SciPy [Virtanen et al. 2020] for sparse linear algebra, and libigl [Jacobson et al. 2018] and Gpytoolbox [Sellán and Stein 2023] for geometry processing routines and quadratic optimization. Conic programs were solved using Mosek [MOSEK ApS 2024], and Cholesky decompositions were performed using scikit-sparse, a Python wrapper for SuiteSparse [Davis and Hu 2011]. We used Blender and BlenderToolbox [Liu 2023b] for 3D renderings and matplotlib [Hunter 2007] for visualizing 2D results. All results

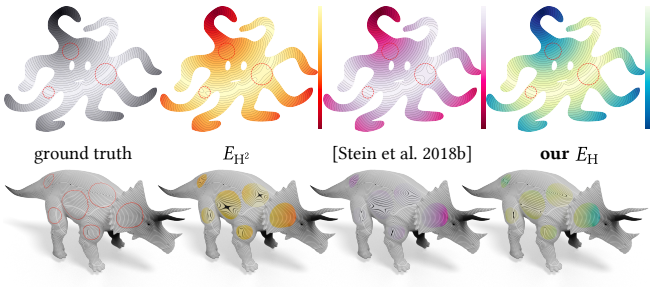


Fig. 12. Filling in missing data using our L^1 Hessian energy E_H. The data inside the red regions is unknown and needs to be recovered. The L^2 energy E_{H^2} destroys the sharp ridge features of the input. Stein et al. [2018b]'s L^1 energy's axis alignment also destroys the ridges. Our L^1 energy reconstructs the sharp feature.

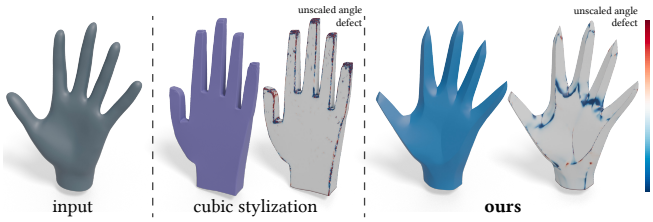


Fig. 13. Our method can stylize shapes to be piecewise flat *(right)*. Unlike the method of Liu and Jacobson [2019]'s *(center)*, our result shapes are not aligned to predefined normal directions, and are fit to the actual geometry of the input shape.

were run and rendered on a workstation with an Intel i7-13700K 16-core processor and Nvidia 4080 graphics card. Runtime statistics can be found in Table 1. In general, the runtimes of our method are on the same order as those of Stein et al. [2018b], and we found that our solver consistently converged for manifold-with-boundary meshes with reasonable triangles.

6.1 Effects of discretization

Fig. 10 shows the behavior of our method under mesh refinement. While our discretization is not guaranteed to converge, we empirically observe convergence to a limit with Loop subdivision.

Without any preprocessing, our discretization struggles with the kinds of triangle shapes that are difficult for many finite element methods. Since our discrete energy is purely intrinsic, however, we can intrinsic Delaunay remesh [Fisher et al. 2006] in a preprocessing step to mitigate this issue. Fig. 11 shows that intrinsic remeshing preprocessing makes our method work even on challenging meshes.

We offer the option for our dual edges $\hat{\mathbf{e}}_{f,i}$ to be either barycentric or circumcenter dual, and default to barycentric for robustness. Fig. 9 shows the difference between these two choices: The circumcenter dual is less robust, and can fail on some triangulations, while performing slightly better on very good triangulations.

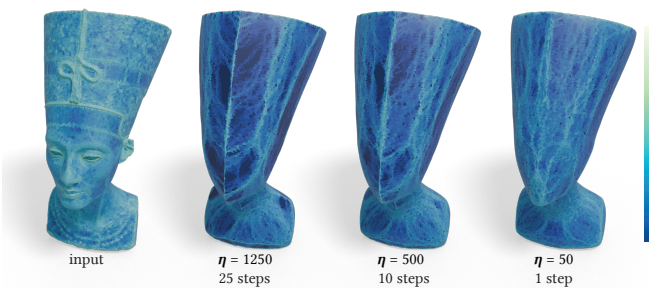


Fig. 14. Our flow under different step sizes. Step counts are chosen to sum to 0.01 (using $\frac{1}{2dt} = \eta$). For large η with more steps, the surface becomes more piecewise-flat. The surface is colored using the local Hessian norm.

7 APPLICATIONS

7.1 Scattered data interpolation

To interpolate scattered data provided at isolated points over a surface, we solve

$$\operatorname*{argmin}_{\mathbf{u}} \mathbf{E}_H(\mathbf{u}) \qquad \text{s.t. } (\mathbf{u})_i = (\mathbf{u}_0)_i \quad \forall i \in S\,. \tag{22}$$

Figs. 1 *(left)* and 6 show our method applied to scattered data interpolation problems on flat and curved surfaces. Unlike the L^2 energy, our L^1 energy reconstructs sharp ridges. Unlike Stein et al. [2018b]'s L^1 energy, our intrinsic approach reconstructs ridges even where they do not align with axes on the flat shape, and on the curved parts of a curved shape.

7.2 Data denoising

We can use our energy as a regularizer that encourages piecewise smoothness when denoising scalar data. Specifically, we can ask for functions which minimize our energy while staying close (in an L^2 sense) to noisy input data:

$$\operatorname*{argmin}_{\mathbf{u}} \mathbf{E}_H(\mathbf{u}) + \alpha\|\mathbf{u} - \mathbf{u}_0\|^2\,. \tag{23}$$

Fig. 2 shows our method applied to a noisy heightfield. Compared to an L^2 energy, our energy results in sharp ridges in the result. Figs. 1 *(center)* and 7 show that our method produces sharp ridges on curved surfaces, unlike the method of Stein et al. [2018b].

7.3 Stylization

Our energy can be used in a geometric flow which encourages *extrinsic* piecewise flatness by applying it to the three coordinate functions x, y, z. Piecewise smooth stylization is a popular task in geometry processing: beyond the L^1 approaches mentioned in Section 2, other approaches include encouraging developability [Binninger et al. 2021; Ion et al. 2020; Rabinovich et al. 2018; Sellán et al. 2020; Stein et al. 2018a] and manipulating the normals of the shape directly [Jadon et al. 2022; Liu and Jacobson 2019, 2021].

For stylization, we repeatedly smooth the coordinate functions, recalculating our Hessian at every iteration on the new geometry.

$$\begin{pmatrix}\mathbf{x}^{(i+1)}\\ \mathbf{y}^{(i+1)}\\ \mathbf{z}^{(i+1)}\end{pmatrix} = \operatorname*{argmin}_{\mathbf{x},\mathbf{y},\mathbf{z}} \mathbf{E}_H(\mathbf{x})+\mathbf{E}_H(\mathbf{y})+\mathbf{E}_H(\mathbf{z})+\eta \left\| \begin{pmatrix}\mathbf{x}\\ \mathbf{y}\\ \mathbf{z}\end{pmatrix} - \begin{pmatrix}\mathbf{x}^{(i)}\\ \mathbf{y}^{(i)}\\ \mathbf{z}^{(i)}\end{pmatrix} \right\|^2, \tag{24}$$

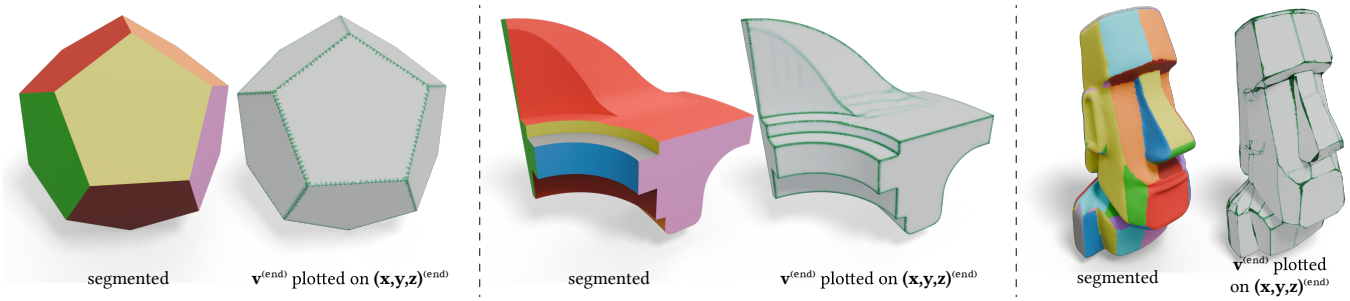


Fig. 15. Our L^1 Hessian energy can be used to segment meshes by repeatedly sharpening and sparsifying a cut indicator function $\mathbf{v}$. The resulting segmentations are plotted on the geometry that is to be segmented, while $\mathbf{v}$ is plotted on top of the sharpened geometry $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ produced during the segmentation process.

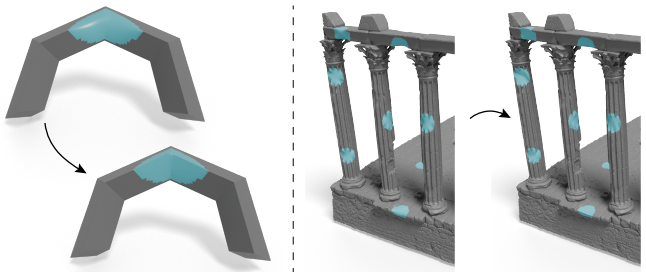


Fig. 16. Filling in holes in the geometry using our L^1 Hessian flow. The hole is first filled with an ad-hoc hole-filling method [Liepa 2003], and then our flow is run to fill the hole with a shape that has a sharp ridge.

where η is a step size that determines the speed and stability of the geometric flow. Additionally, at each time step, the Hessian matrix is constructed using a version of the mesh which is scaled to have maximum edge length 1. Figs. 3 (left) and 4 show a variety of piecewise linear stylizations using our flow with multiple steps. Fig. 3 (center) shows the method applied to noisy coordinates of a surface; the result is a denoising of the geometry. Fig. 14 shows the effect of varying η while scaling the step count to maintain the total time. Larger ηs (and step counts) tend to better minimize the energy over the shape, leading to a sharper appearance. It is worth noting that our stylization effect differs from Liu and Jacobson [2019]'s, which aligns mesh normals to a predefined set of directions (Fig. 13).

7.4 Hole filling

Our L^1 Hessian energy can be used to fill holes on shapes with sharp features. Filling holes in meshes is a popular geometry processing task that has been extensively studied, as is detailed in the survey of Guo et al. [2016]. Examples of more recent methods are the works of Calatroni et al. [2022]; Centin et al. [2015]; Feng et al. [2020].

7.4.1 Scalar data. Smoothness energies can be used as a regularizer for completing missing regions on a surface. To this end, we use (22), but rather than choosing S to be a sparse point set on the surface, it is instead chosen to be the known ground truth data outside of the missing regions. Fig. 12 shows our method used for data hole-filling on both a flat and curved surface. Unlike Stein et al. [2018b]'s L^1 energy, our intrinsic energy does not suffer from axis alignment, and manages to fill in the holes with sharp ridges.

7.4.2 Geometry. In situations where the missing hole data is geometric, we can employ our stylizing flow to fill in the gaps, restricting to an initial hole-filled geometry. We first fill relevant boundary loops using a coarse hole-filling method [Liepa 2003] before remeshing [Botsch and Kobbelt 2004] with a fine-grained target edge length. We then initialize the filled hole using minimizers of the L^2 interpolation problem, and flow the mesh with our stylizing flow, constraining data outside of the missing regions to remain the same between iterations.

Figs. 3 and 16 shows this process in action.

7.5 Segmentation

E_{H} can be used to segment meshes. This is another and well-studied problem in geometry processing; recent surveys on image and mesh segmentation [Minaee et al. 2022; Rodrigues et al. 2018] provide an overview over the large body of work on the topic. Especially relevant for us are approaches using the Mumford-Shah model [Chan and Vese 2001; Gao and Bui 2005; Kiefer et al. 2020; Mumford and Shah 1985], its Ambrosio-Tortorelli approximation [Ambrosio and Tortorelli 1990; Coeurjolly et al. 2016], and its generalizations to triangle meshes [Bonneel et al. 2018; Liu et al. 2020; Weill–Duflos et al. 2023], as our segmentation method is based on inserting an L^1 Hessian term into a Mumford-Shah-like formulation.

Our method works by constructing an approximation to the input mesh which simultaneously optimizes E_{H} as well as the set of regions over which the energy is evaluated. We employ a variation on the popular Mumford-Shah functional where we minimize

$$\mathcal{M}(u, \Gamma) = \alpha \int_{\Omega} \|u(x) - u_0(x)\|^2 \, dx + \int_{\Omega \setminus \Gamma} \| \mathrm{H}_u \| \, dx + \lambda \int_{\Gamma} dx \, .$$

The first term penalizes deviation from the original function, the second term encourages linearity away from the sparse set of segmentation boundary curves Γ, and the third term penalizes the total length of Γ. As this functional is highly nonlinear and difficult to optimize, we instead employ the approximation of *Ambrosio-Tortorelli*

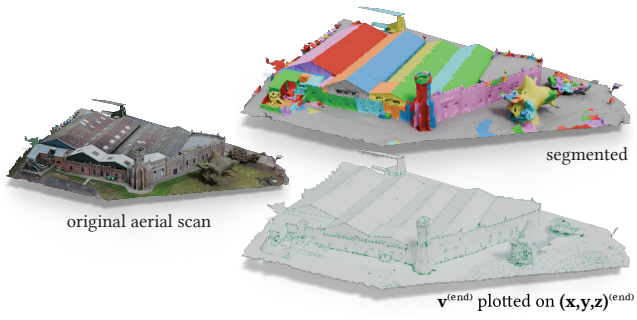


Fig. 17. Our segmentation strategy applied to a mesh reconstructed from an aerial scan of a building complex. The L^1 Hessian energy enforces a piecewise linear prior, selecting the mostly-flat angled roofs and side walls as segments.

[Ambrosio and Tortorelli 1990], but insert our L^1 Hessian:

$$A_\varepsilon(u,v) = \alpha \int_\Omega \|u(x) - u_0(x)\|^2 \, dx + \int_\Omega v^2 \| \, \mathrm{H}_u \, \| \, dx \\ + \lambda \int_\Omega \varepsilon \|\nabla v\|^2 + \frac{1}{4\varepsilon}(1-v)^2 \, dx \tag{25}$$

Here, v is a functional which represents the true Γ as an (inverse) indicator function which gets progressively sparser as $\varepsilon \to 0$. We follow Weill–Duflos et al. [2023]'s discretization and solve using a variation on their alternating optimization method; see Supplemental E for more details. Figs. 1 *(right)*, 15, and 17 show our segmentation procedure in action. We recover clean cuts and segments that follow the ridges in a piecewise smooth approximation. Figure 18 shows our results for varying λ, where smaller λs encourage solutions with many segments.

7.6 Diffusion curves

Diffusion curves [Orzan et al. 2008] extend the concept of vector graphics to colors. Instead of assigning a color value to every point of the domain, colors are assigned to sparse curves only, and smoothly interpolated to the rest of the domain. Our L^1 Hessian optimization can be used to solve the inverse diffusion curve problem: for a surface with colors, find a set of sparse curves and color values on those curves so that, when reconstructed via the diffusion curves method, we obtain a result that is close to the original input.

The inverse diffusion curves problem asks for a set of curves Γ and color values c_L and c_R along the "left" and "right" sides Γ_L and Γ_R of Γ such that the solution to

$$\Delta u\big|_{\Omega\setminus\Gamma} = 0, \quad u\big|_{\Gamma_L} = c_L, \quad u\big|_{\Gamma_R} = c_R \tag{26}$$

approximates a ground truth color distribution u_0.

To solve this problem, we split it into a curve discovery step and an approximate color assignment step. In the first step, we sample an artist-provided texture at the vertices of the mesh, and then choose Γ as the segment boundaries resulting from our segmentation procedure. In the second step, we solve the quadratic program

$$\mathbf{u}^* = \operatorname*{argmin}_{\mathbf{u}} \mathbf{u}^\mathsf{T}\mathbf{L}\mathbf{u} + \alpha\|\mathbf{u} - \mathbf{u}_0\|^2 \tag{27}$$

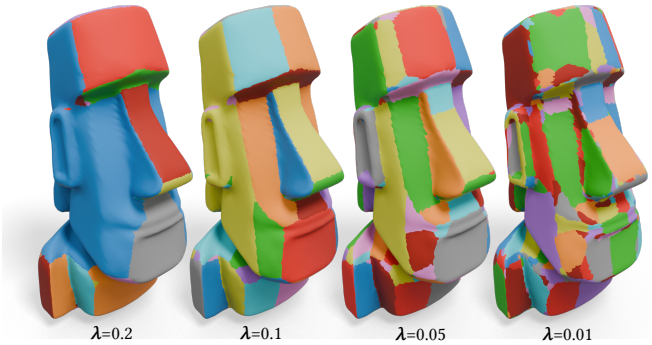


Fig. 18. Varying the parameter λ in the segmentation tasks controls for the number of segments in the final segmentation.

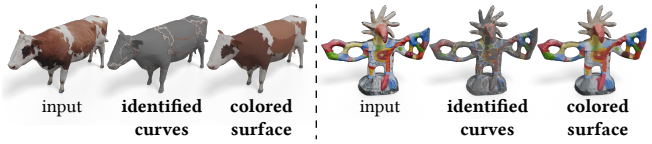


Fig. 19. Using our L^1 Hessian energy to construct diffusion curves on a surface. For an input surface with vertex colors, we identify sparse, salient curves to reconstruct color on the entire surface.

where $\mathbf{L}$ is the cotangent Laplacian, over each region of the segmentation. c_L and c_R are assigned by evaluating $\mathbf{u}^*$ at the segment boundaries. We recover approximate diffusion colors by solving (26) over each segment. We show the results of this procedure in Figure 19 based on colors u_0 sampled from an artist-provided texture.

8 LIMITATIONS & CONCLUSION

Our method smooths out some sharp ridges on highly curved segments (Fig. 20, *left*). This could potentially be addressed by adding an L^1 curvature term to the energy [Stein et al. 2020a]. The discretization can also introduce mesh-dependence: Since the problem of piecewise mesh stylization is under-determined, the initial direction of mesh edges can influence the results (Fig. 20, *right*). Moreover, our discretization of the L^1 Hessian energy E_H does not converge to the exact result under mesh refinement. While we capture some qualitative L^1 behavior, our discretization cannot be used to compute actual energy values. Interesting future work directions are to improve our discretization by explicitly accounting for curvature, employing higher-order FEM, and using different dual edges $\hat{\mathbf{e}}_{f,i}$. Lastly, while the smooth Hessian energy is well-defined on solid volumes, a discretization over tetrahedral meshes remains unknown.

Ridge-like regions appear in our results, but we do not explicitly control the locations of these ridges. It could be beneficial to use a weighting scheme to promote ridge development in specified regions, similar to how cut locations are encouraged to form in regions of high ambient occlusion in Weill–Duflos et al. [2023].

We use a black-box conic solver for our optimization procedure, which negatively impacts performance and scaling. We leave the development and implementation of more-efficient optimization procedures for the L^1 Hessian energy to future work.

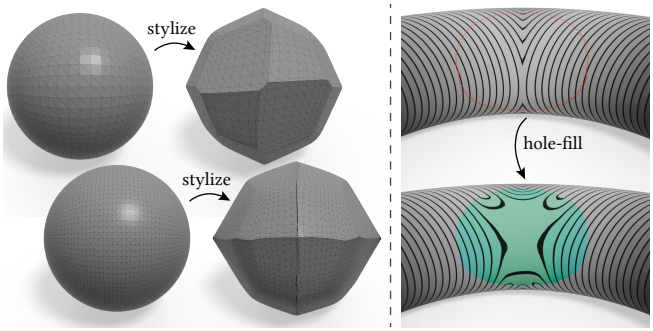


Fig. 20. Different initial mesh layouts lead to different stylizations *(left)*. On strongly curved segments, our method does not preserve this X-shaped sharp feature during hole-filling *(right)*.

ACKNOWLEDGMENTS

Our research is funded in part by NSERC Discovery (RGPIN – 2022 – 04680), the Ontario Early Research Award program, the Canada Research Chairs Program, a Sloan Research Fellowship, the DSI Catalyst Grant program and gifts by Adobe Inc. We thank Silvia Sellán and Yuta Noma for proofreading.

We acknowledge and thank the authors of the 2D and 3D assets used in this paper. These models included the Plane [YahooJAPAN 2019b], Cherries [siotech2011 2021], Tete [Svay 2015], Springer [TimEdwards 2023], Koala [YahooJAPAN 2019a], Person [Jacobson 2013], Cat [billyd 2023], Skull [YahooJAPAN 2019c], Giraffe [SDPM Esare 2021], Frog [Liu 2023a], Rocket [Microsoft 2014], Octopus [Jacobson 2013], Triceratops [wojciechmiedziocha 2019], Hand [Max Planck Society e.V via SMPL 2021], Nefertiti [Al-Badri and Nelles 2023], Fandisk [Hoppe et al. 1994], Moai [zames1992 2024], Evora [Global Digital Heritage and GDH-Afrika 2019], Building [Blayne Jackson 2016], Cow [Josué Boisvert 2015], and Stravinsky [Emm 2021].

REFERENCES

Jascha Achenbach, Eduard Zell, and Mario Botsch. 2015. Accurate Face Reconstruction through Anisotropic Fitting and Eye Correction. In *Vision, Modeling & Visualization*, David Bommes, Tobias Ritschel, and Thomas Schultz (Eds.). The Eurographics Association. https://doi.org/10.2312/vmv.20151251

Nora Al-Badri and Jan Nikolai Nelles. 2023. Nefertiti. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/nefertiti..

Luigi Ambrosio and Vincenzo Maria Tortorelli. 1990. Approximation of functional depending on jumps by elliptic functional via t-convergence. *Communications on Pure and Applied Mathematics* 43, 8 (1990), 999–1036.

Haim Avron, Andrei Sharf, Chen Greif, and Daniel Cohen-Or. 2010. l1-Sparse reconstruction of sharp point set surfaces. *ACM Trans. Graph.* 29, 5, Article 135 (2010).

Ilya Baran and Jovan Popović. 2007. Automatic rigging and animation of 3D characters. *ACM Trans. Graph.* 26, 3 (2007), 72–es.

Maïtine Bergounioux and Loïc Piffet. 2010. A second-order model for image denoising and/or texture extraction. *Set-Valued and Variational Analysis* 18 (2010), 277–306.

billyd. 2023. Cat. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/cat, originally from thingiverse.com/thing:1565405. Asset licensed under CC BY 4.0..

Alexandre Binninger, Floor Verhoeven, Philipp Herholz, and Olga Sorkine-Hornung. 2021. Developable Approximation via Gauss Image Thinning. *Computer Graphics Forum* 40, 5 (2021), 289–300.

Blayne Jackson. 2016. Building test Phantom 4 Pro - 20mp cam. https://sketchfab.com/3d-models/building-test-phantom-4-pro-20mp-cam-372e3c847ee74c0bb3942e22a3d0a6b8

Iwan Boksebeld and Amir Vaxman. 2022. High-Order Directional Fields. *ACM Trans. Graph.* 41, 6, Article 254 (2022).

Nicolas Bonneel, David Coeurjolly, Pierre Gueth, and Jacques-Olivier Lachaud. 2018. Mumford-Shah Mesh Processing using the Ambrosio-Tortorelli Functional. *Computer Graphics Forum* 37, 7 (2018), 75–85.

Mario Botsch and Leif Kobbelt. 2004. A remeshing approach to multiresolution modeling. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. 185–192.

Christopher Brandt and Klaus Hildebrandt. 2017. Compressed vibration modes of elastic bodies. *Computer Aided Geometric Design* 52-53 (2017), 297–312.

Alexander M. Bronstein, Yoni Choukroun, Ron Kimmel, and Matan Sela. 2016. Consistent Discretization and Minimization of the L1 Norm on Manifolds. *CoRR* abs/1609.05434 (2016). arXiv:1609.05434 http://arxiv.org/abs/1609.05434

Alfred M. Bruckstein, David L. Donoho, and Michael Elad. 2009. From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images. *SIAM Rev.* 51, 1 (2009), 34–81.

Luca Calatroni, Martin Huska, Serena Morigi, and Giuseppe Antonio Recupero. 2022. A Unified Surface Geometric Framework for Feature-Aware Denoising, Hole Filling and Context-Aware Completion. *J. Math. Imaging Vis.* 65, 1 (2022), 82–98.

Emmanuel J. Candès, Michael B. Wakin, and Stephen F. Boyd. 2008. Enhancing Sparsity by Reweighted l1 Minimization. *Journal of Fourier Analysis and Applications* 14 (2008), 877–905.

Marco Centin, Nicola Pezzotti, and Alberto Signoroni. 2015. Poisson-driven seamless completion of triangular meshes. *Computer Aided Geometric Design* 35-36 (2015).

Antonin Chambolle and Pierre-Louis Lions. 1997. Image recovery via total variation minimization and related problems. *Numer. Math.* 76 (1997), 167–188.

T.F. Chan and L.A. Vese. 2001. A level set algorithm for minimizing the Mumford-Shah functional in image processing. In *Proceedings IEEE Workshop on Variational and Level Set Methods in Computer Vision*. 161–168.

Tony F. Chan, Selim Esedoglu, and Frederick E. Park. 2007. Image decomposition combining staircase reduction and texture extraction. *Journal of Visual Communication and Image Representation* 18, 6 (2007), 464–486.

David Coeurjolly, Marion Foare, Pierre Gueth, and Jacques-Olivier Lachaud. 2016. Piecewise smooth reconstruction of normal vector field on digital data. *Computer Graphics Forum* 35, 7 (2016), 157–167.

Richard Courant and David Hilbert. 1989. *The Calculus of Variations*. John Wiley I& Sons, Ltd, Chapter 4, 164–274. https://doi.org/10.1002/9783527617210.ch4 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/9783527617210.ch4

Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Robust fairing via conformal curvature flow. *ACM Trans. Graph.* 32, 4, Article 61 (2013).

Bram Custers and Amir Vaxman. 2020. Subdivision Directional Fields. *ACM Trans. Graph.* 39, 2, Article 11 (2020), 20 pages.

Timothy A. Davis and Yifan Hu. 2011. The University of Florida Sparse Matrix Collection. *ACM Trans. Math. Software* 38, 1 (2011).

Fernando de Goes, Mathieu Desbrun, Mark Meyer, and Tony DeRose. 2016. Subdivision exterior calculus for geometry processing. *ACM Trans. Graph.* 35, 4, Article 133 (2016).

Fernando de Goes, Beibei Liu, Max Budninskiy, Yiying Tong, and Mathieu Desbrun. 2014. Discrete 2-Tensor Fields on Triangulations. *Computer Graphics Forum* 33, 5 (2014), 13–24.

Mathieu Desbrun, Anil N. Hirani, Melvin Leok, and Jerrold E. Marsden. 2005. Discrete Exterior Calculus. arXiv:math/0508341 [math.DG]

Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. 317–324.

Alexandre Djerbetian. 2016. *Tangent Vector Fields on Triangulated Surfaces - An Edge-Based Approach*. Master's thesis. Technion.

Emm. 2021. Day 203: Stravinsky Fountain, Paris - 1scanaday. https://sketchfab.com/3d-models/day-203-stravinsky-fountain-paris-1scanaday-f8b960d4cef1432a8dbd44315e3811e9

Chao Feng, Jin Liang, Maodong Ren, Gen Qiao, Wang Lu, and Shifan Liu. 2020. A Fast Hole-Filling Method for Triangular Mesh in Additive Repair. *Applied Sciences* 10 (2020).

Matthew Fisher, Peter Schröder, Mathieu Desbrun, and Hugues Hoppe. 2007. Design of tangent vector fields. *ACM Trans. Graph.* 26, 3 (2007), 56–es.

Matthew Fisher, Boris Springborn, Alexander I. Bobenko, and Peter Schroder. 2006. An algorithm for the construction of intrinsic delaunay triangulations with applications to digital geometry processing. In *ACM SIGGRAPH 2006 Courses* (Boston, Massachusetts) *(SIGGRAPH '06)*. Association for Computing Machinery, New York, NY, USA, 69–74. https://doi.org/10.1145/1185657.1185668

Haoying Fu, Michael K. Ng, Mila Nikolova, and Jesse L. Barlow. 2006. Efficient Minimization Methods of Mixed l2-l1 and l1-l1 Norms for Image Restoration. *SIAM Journal on Scientific Computing* 27, 6 (2006), 1881–1902.

Song Gao and T.D. Bui. 2005. Image segmentation and selective smoothing by using Mumford-Shah model. *IEEE Transactions on Image Processing* 14, 10 (2005), 1537–1549.

Global Digital Heritage and GDH-Afrika. 2019. Roman Temple of Evora. https://sketchfab.com/3d-models/roman-temple-of-evora-935f17a3824d49f7b2505a0686450d51

Eitan Grinspun, Yotam Gingold, Jason Reisman, and Denis Zorin. 2006. Computing discrete shape operators on general meshes. *Computer Graphics Forum* 25, 3 (2006), 547–556.

Xiaoyuan Guo, Jun Xiao, and Ying Wang. 2016. A survey on algorithms of hole filling in 3D surface reconstruction. *The Visual Computer* 34 (2016), 93–103.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (2020), 357–362.

Lei He and Scott Schaefer. 2013. Mesh denoising via L0 minimization. *ACM Trans. Graph.* 32, 4, Article 64 (jul 2013), 8 pages. https://doi.org/10.1145/2461912.2461965

Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. 1994. Piecewise smooth surface reconstruction. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '94)*. Association for Computing Machinery, New York, NY, USA, 295–302. https://doi.org/10.1145/192161.192233

Jin Huang, Tengfei Jiang, Zeyun Shi, Yiying Tong, Hujun Bao, and Mathieu Desbrun. 2014. l1-Based Construction of Polycube Maps from Complex Shapes. *ACM Trans. Graph.* 33, 3, Article 25 (2014).

J. D. Hunter. 2007. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* 9, 3 (2007), 90–95.

Alexandra Ion, Michael Rabinovich, Philipp Herholz, and Olga Sorkine-Hornung. 2020. Shape approximation by developable wrapping. *ACM Trans. Graph.* 39, 6 (2020), 12 pages.

Alec Jacobson. 2013. Algorithms and Interfaces for Real-Time Deformation of 2D and 3D Shapes.

Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. 2011. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.* 30, 4, Article 78 (2011).

Alec Jacobson, Daniele Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. https://libigl.github.io/.

Alec Jacobson, Tino Weinkauf, and Olga Sorkine. 2012. Smooth Shape-Aware Functions with Controlled Extrema. *Computer Graphics Forum* 31, 5 (2012), 1577–1586.

Elias Jadon, Bernhard Thomaszewski, Aleksandra Anna Apolinarska, and Roi Poranne. 2022. Continuous deformation based panelization for design rationalization. In *SIGGRAPH Asia 2022 Conference Papers (SA '22)*. Article 44.

Josué Boisvert. 2015. Cow. https://sketchfab.com/3d-models/cow-99d333e3b4e4470a8d7d38436489c001

Lukas Kiefer, Martin Storath, and Andreas Weinmann. 2020. An Efficient Algorithm for the Piecewise Affine-Linear Mumford-Shah Model Based on a Taylor Jet Splitting. *IEEE Transactions on Image Processing* 29 (2020), 921–933.

Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Globally optimal direction fields. *ACM Trans. Graph.* 32, 4, Article 59 (2013).

Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2015. Stripe patterns on surfaces. *ACM Trans. Graph.* 34, 4, Article 39 (2015).

Rongjie Lai, Xue-Cheng Tai, and Tony F. Chan. 2013. A Ridge and Corner Preserving Model for Surface Restoration. *SIAM Journal on Scientific Computing* 35, 2 (2013).

Lingxiao Li, Paul Zhang, Dmitriy Smirnov, S. Mazdak Abulnaga, and Justin Solomon. 2021. Interactive all-hex meshing via cuboid decomposition. *ACM Trans. Graph.* 40, 6, Article 256 (2021).

Peter Liepa. 2003. Filling Holes in Meshes . In *Eurographics Symposium on Geometry Processing (SGP 2003)*, Leif Kobbelt, Peter Schroeder, and Hugues Hoppe (Eds.).

Beibei Liu, Yiying Tong, Fernando De Goes, and Mathieu Desbrun. 2016. Discrete Connection and Covariant Derivative for Vector Field Analysis and Design. *ACM Trans. Graph.* 35, 3, Article 23 (2016), 17 pages.

Diana Liu. 2023a. Frog. https://sketchfab.com/3d-models/frog-436deb1555fa42b9bc146e1b6b0d1ebc

Hsueh-Ti Derek Liu. 2023b. BlenderToolbox. https://github.com/HTDerekLiu/BlenderToolbox.

Hsueh-Ti Derek Liu and Alec Jacobson. 2019. Cubic stylization. *ACM Trans. Graph.* 38, 6, Article 197 (2019).

Hsueh-Ti Derek Liu and Alec Jacobson. 2021. Normal-Driven Spherical Shape Analogies. *Computer Graphics Forum* 40, 5 (2021), 45–55.

Zheng Liu, Rongjie Lai, Huayan Zhang, and Chunlin Wu. 2019. Triangulated Surface Denoising using High Order Regularization with Dynamic Weights. *SIAM Journal on Scientific Computing* 41, 1 (2019).

Zheng Liu, Weina Wang, Saishang Zhong, Bohong Zeng, Jinqin Liu, and Weiming Wang. 2020. Mesh Denoising via a Novel Mumford–Shah Framework. *Computer-Aided Design* 126 (2020), 102858. https://doi.org/10.1016/j.cad.2020.102858

M. Lysaker, A. Lundervold, and Xue-Cheng Tai. 2003. Noise removal using fourth-order partial differential equation with applications to medical magnetic resonance images in space and time. *IEEE Transactions on Image Processing* 12, 12 (2003), 1579–1590.

Marius Lysaker and Xue-Cheng Tai. 2006. Iterative Image Restoration Combining Total Variation Minimization and a Second-Order Functional. *Int. J. Comput. Vision* 66, 1 (2006), 5–18.

Max Planck Society e.V via SMPL. 2021. Hand. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/hand, adapted from M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, M. Black "SMPL" (SIGGRAPH Asia 2015). Asset licensed under CC BY 4.0..

Microsoft. 2014. Rocket. https://sketchfab.com/3d-models/saturn-v-rocket-37f28664ac644b4f87a7bcfe0585ec20

Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. 2022. Image Segmentation Using Deep Learning: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 7 (2022), 3523–3542.

MOSEK ApS. 2024. *MOSEK Optimizer API for Python 10.1.31 (https:// docs.mosek.com/ latest/ pythonapi/ index.html)*.

David Mumford and Jayant Shah. 1985. Boundary detection by minimizing functionals. *Proc. IEEE Conf. Comp. Vis. Pattern Recognition* (1985).

Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. 2008. Diffusion curves: a vector representation for smooth-shaded images. In *ACM SIGGRAPH 2008 Papers*.

Peter Petersen. 2006. *Riemannian Geometry*. Springer.

Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2018. Discrete Geodesic Nets for Modeling Developable Surfaces. *ACM Trans. Graph.* 37, 2, Article 16 (2018).

Rui S. V. Rodrigues, José F. M. Morgado, and Abel J. P. Gomes. 2018. Part-Based Mesh Segmentation: A Survey. *Computer Graphics Forum* 37, 6 (2018), 235–274.

SDPM Esare. 2021. Giraffe. https://sketchfab.com/3d-models/giraffe-7c91bd719e594fcc930f3646db843265

Silvia Sellán, Noam Aigerman, and Alec Jacobson. 2020. Developability of heightfields via rank minimization. *ACM Trans. Graph.* 39, 4, Article 109 (2020).

Silvia Sellán, Jack Luong, Leticia Mattos Da Silva, Aravind Ramakrishnan, Yuchuan Yang, and Alec Jacobson. 2023. Breaking Good: Fracture Modes for Realtime Destruction. *ACM Trans. Graph.* 42, 1, Article 10 (2023).

Silvia Sellán and Oded Stein. 2023. gpytoolbox: A Python Geometry Processing Toolbox. https://gpytoolbox.org/.

Nicholas Sharp, Yousuf Soliman, and Keenan Crane. 2019a. Navigating intrinsic triangulations. *ACM Trans. Graph.* 38, 4, Article 55 (2019).

Nicholas Sharp, Yousuf Soliman, and Keenan Crane. 2019b. The Vector Heat Method. *ACM Trans. Graph.* 38, 3, Article 24 (2019).

siotech2011. 2021. Cherries. https://sketchfab.com/3d-models/cherries-a6a77f3d44d44c2dacab02b7a55c0b4e

Gabriele Steidl. 2006. A Note on the Dual Treatment of Higher-Order Regularization Functional. *Computing* 76 (2006), 135–148.

Gabriele Steidl, Stephan Didas, and Julia Neumann. 2005. Relations Between Higher Order TV Regularization and Support Vector Regression. In *Scale Space and PDE Methods in Computer Vision*. 515–527.

Oded Stein, Eitan Grinspun, and Keenan Crane. 2018a. Developability of triangle meshes. *ACM Trans. Graph.* 37, 4, Article 77 (jul 2018).

Oded Stein, Eitan Grinspun, Max Wardetzky, and Alec Jacobson. 2018b. Natural Boundary Conditions for Smoothing in Geometry Processing. *ACM Trans. Graph.* 37, 2, Article 23 (2018).

Oded Stein, Alec Jacobson, Max Wardetzky, and Eitan Grinspun. 2020a. A Smoothness Energy without Boundary Distortion for Curved Surfaces. *ACM Trans. Graph.* 39, 3 (2020).

Oded Stein, Max Wardetzky, Alec Jacobson, and Eitan Grinspun. 2020b. A Simple Discretization of the Vector Dirichlet Energy . *Computer Graphics Forum* (2020).

Maurice Svay. 2015. Tête by Henri Laurens. https://sketchfab.com/3d-models/tete-by-henri-laurens-a019a7f8a41b4e7fa24c8c2095e75d8b

Daniel Sýkora, Ladislav Kavan, Martin Čadík, Ondřej Jamriška, Alec Jacobson, Brian Whited, Maryann Simmons, and Olga Sorkine-Hornung. 2014. Ink-and-ray: Bas-relief meshes for adding global illumination effects to hand-drawn characters. *ACM Trans. Graph.* 33, 2, Article 16 (2014).

TimEdwards. 2023. Springer. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/springer, originally from thingiverse.com/thing:335658. Asset licensed under GPLv2.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272.

Shawn W. Walker. 2024. Approximating the Shape Operator with the Surface Hellan–Herrmann–Johnson Element. *SIAM Journal on Scientific Computing* 46, 2 (2024), A1252–A1275. https://doi.org/10.1137/22M1531968 arXiv:https://doi.org/10.1137/22M1531968

Colin Weill–Duflos, David Coeurjolly, Fernando de Goes, and Jacques-Olivier Lachaud. 2023. Joint optimization of distortion and cut location for mesh parameterization using an Ambrosio-Tortorelli functional. *Computer Aided Geometric Design* 105 (2023).

Tino Weinkauf, Yotam Gingold, and Olga Sorkine. 2010. Topology-based Smoothing of 2D Scalar Fields with C1-Continuity. *Computer Graphics Forum* 29, 3 (2010), 1221–1230.

wojciechmiedziocha. 2019. Triceratops dinosaur. https://sketchfab.com/3d-models/triceratops-dinosaur-87527079bad44917ab1b98a456b46c7e

Xiaoqun Wu, Jianmin Zheng, Yiyu Cai, and Chi-Wing Fu. 2015. Mesh Denoising using Extended ROF Model with L1 Fidelity. *Computer Graphics Forum* 34, 7 (2015), 35–45.

YahooJAPAN. 2019a. Koala. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/koala, originally from thingiverse.com/thing:182225. Asset licensed under CC BY 3.0..

YahooJAPAN. 2019b. Plane. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/plane, originally from thingiverse.com/thing:182252. Asset licensed under CC BY 3.0..

YahooJAPAN. 2019c. Skull. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/skull, originally from thingiverse.com/thing:182365. Asset licensed under CC BY 3.0..

Y.-L. You and M. Kaveh. 2000. Fourth-order partial differential equations for noise removal. *IEEE Transactions on Image Processing* 9, 10 (2000), 1723–1730.

Jing Yuan, Christoph Schnörr, and Gabriele Steidl. 2009. Total-Variation Based Piecewise Affine Regularization. In *Scale Space and Variational Methods in Computer Vision*. 552–564.

zames1992. 2024. Moai Free. https://sketchfab.com/3d-models/moai-free-bf7f8fbcdcd7408f9f95567bf9cde054

Juyong Zhang, Bailin Deng, Zishun Liu, Giuseppe Patanè, Sofien Bouaziz, Kai Hormann, and Ligang Liu. 2014. Local barycentric coordinates. *ACM Trans. Graph.* 33, 6, Article 188 (2014).

Paul Zhang, Josh Vekhter, Edward Chien, David Bommes, Etienne Vouga, and Justin Solomon. 2020. Octahedral Frames for Feature-Aligned Cross Fields. *ACM Trans. Graph.* 39, 3, Article 25 (2020), 13 pages.

SUPPLEMENTAL MATERIAL

A DETAILED DERIVATION OF OPERATORS

Our Hessian operator consists of 4 matrices which are multiplied together. The first just represents the gradient on faces, in the intrinsic orthonormal basis with $\hat{\mathbf{x}}$ pointing along the edge between edge ij:

$$(\mathbf{Gu})_f = \begin{pmatrix} \frac{-1}{l_{ij}} u_i + \frac{1}{l_{ij}} u_j \\ \frac{l_{ki}\cos\theta_i - l_{ij}}{l_{ij} l_{ki} \sin\theta_i} u_i - \frac{\cos\theta_i}{l_{ij}\sin\theta_i} u_j + \frac{1}{l_{ki}\sin\theta_i} u_k \end{pmatrix}, \tag{28}$$

where the face f has vertices i, j, k, and θ_i is the tip angle at vertex i.

The second matrix takes differences of the previously-found gradients across a halfedge between the two faces. Explicitly, if θ_{f_i,f_j} is the angle the basis at face f would need to be rotated by to align with the basis at face f_j, the halfedge h_{ij} (on f_i corresponding to the edge shared between f_i and f_j) gets the matrix entries

$$\begin{aligned} (\mathbf{C})_{h_{ij},f_j} &= \left[\mathbf{R}(\theta)\right] \\ (\mathbf{C})_{h_{ij},f_i} &= \left[-\mathbf{I}_{2\times 2}\right] \end{aligned} \tag{29}$$

The third matrix divides the previous difference by the length of the dual edge between two centroids and performs a tensor product. Let e.g. $t_{ij,x}$ correspond to the x entry of the unit vector pointing from the centroid of f to the centroid at the face across halfedge ij in triangle f, and $l^\star_{ij}$ correspond to the intrinsic distance between these centroids. Additionally, let e.g. $c_{ij,x}$ correspond to the x entry in the difference taken over halfedge ij in face f. Then

$$(\mathbf{T})_f (c)_f = \begin{pmatrix} T_{xx} \\ T_{xy} \\ T_{yx} \\ T_{yy} \end{pmatrix}_f (c)_f = \left(\hat{T}_1 \mid \hat{T}_2 \mid \hat{T}_3 \mid \hat{T}_4 \mid \hat{T}_5 \mid \hat{T}_6\right) \begin{pmatrix} c_{jk,x} \\ c_{jk,y} \\ c_{ki,x} \\ c_{ki,y} \\ c_{ij,x} \\ c_{ij,y} \end{pmatrix}_f$$

with

$$\begin{aligned} \hat{T}_1 &= \left(\tfrac{1}{l^\star_{jk}} t^3_{jk,x} \quad \tfrac{1}{l^\star_{jk}} t^2_{jk,x} t_{jk,y} \quad \tfrac{1}{l^\star_{jk}} t^2_{jk,x} t_{jk,y} \quad \tfrac{1}{l^\star_{jk}} t_{jk,x} t^2_{jk,y}\right)^\mathsf{T} \\ \hat{T}_2 &= \left(\tfrac{1}{l^\star_{jk}} t^2_{jk,x} t_{jk,y} \quad \tfrac{1}{l^\star_{jk}} t_{jk,x} t^2_{jk,y} \quad \tfrac{1}{l^\star_{jk}} t_{jk,x} t^2_{jk,y} \quad \tfrac{1}{l^\star_{jk}} t^3_{jk,y}\right)^\mathsf{T} \\ \hat{T}_3 &= \left(\tfrac{1}{l^\star_{ki}} t^3_{ki,x} \quad \tfrac{1}{l^\star_{ki}} t^2_{ki,x} t_{ki,y} \quad \tfrac{1}{l^\star_{ki}} t^2_{ki,x} t_{ki,y} \quad \tfrac{1}{l^\star_{ki}} t_{ki,x} t^2_{ki,y}\right)^\mathsf{T} \\ \hat{T}_4 &= \left(\tfrac{1}{l^\star_{ki}} t^2_{ki,x} t_{ki,y} \quad \tfrac{1}{l^\star_{ki}} t_{ki,x} t^2_{ki,y} \quad \tfrac{1}{l^\star_{ki}} t_{ki,x} t^2_{ki,y} \quad \tfrac{1}{l^\star_{ki}} t^3_{ki,y}\right)^\mathsf{T} \\ \hat{T}_5 &= \left(\tfrac{1}{l^\star_{ij}} t^3_{ij,x} \quad \tfrac{1}{l^\star_{ij}} t^2_{ij,x} t_{ij,y} \quad \tfrac{1}{l^\star_{ij}} t^2_{ij,x} t_{ij,y} \quad \tfrac{1}{l^\star_{ij}} t_{ij,x} t^2_{ij,y}\right)^\mathsf{T} \\ \hat{T}_6 &= \left(\tfrac{1}{l^\star_{ij}} t^2_{ij,x} t_{ij,y} \quad \tfrac{1}{l^\star_{ij}} t_{ij,x} t^2_{ij,y} \quad \tfrac{1}{l^\star_{ij}} t_{ij,x} t^2_{ij,y} \quad \tfrac{1}{l^\star_{ij}} t^3_{ij,y}\right)^\mathsf{T} \end{aligned}$$

The fourth matrix $\mathbf{M}_F$ is a matrix containing facewise areas, repeated 4 times for each face.

Then the four intrinsic Hessian entries corresponding to a function $\mathbf{u}$ are given by the entries of

$$\mathbf{Hu} = \mathbf{M}_F \mathbf{T} \mathbf{C} \mathbf{G} \mathbf{u} \tag{30}$$

Each matrix entry corresponding to boundary halfedges is zeroed out.

B FLAT L^1 HESSIAN BOUNDARY CONDITIONS IN 2D

Here, we derive the boundary conditions for the L^1 Hessian in a (nice enough) subset Ω of $\mathbb{R}^2$, assuming $\|Hu\|_F \neq 0$. We start with Equation (12), partially converting it to indices:

$$\int_\Omega \frac{1}{\|Hu\|_F} \partial_i \partial_j u \cdot \partial_i \partial_j \eta \, dx = 0 \,. \tag{31}$$

Integrating by parts twice (once in i and once in j) yields

$$\begin{aligned} \int_{\partial\Omega} \frac{1}{\|Hu\|_F} \mathbf{n}^\mathsf{T} (Hu)(\nabla\eta) - \eta \cdot \left(\nabla \cdot \frac{Hu}{\|Hu\|_F}\right) \cdot \mathbf{n} \, dS \\ + \int_\Omega \eta \cdot H^* \left(\frac{Hu}{\|Hu\|_F}\right) dx = 0 \,. \end{aligned} \tag{32}$$

The integration over Ω is precisely the interior pointwise condition given in Equation (13). Here we focus on understanding the boundary terms using two characteristic types of variation η, and subsequently drop the last term from (32).

B.1 Case 1: Variation is zero on $\partial\Omega$ and $\nabla\eta \propto \mathbf{n}$

First, we consider variations which are zero on the boundary, but which only have gradient in the normal direction. Such variations satisfy $\nabla\eta = g\mathbf{n}$ where g is a scalar function. The boundary integration becomes

$$\int_{\partial\Omega} \frac{1}{\|Hu\|_F} \mathbf{n}^\mathsf{T} (Hu)(g\mathbf{n}) - 0 \cdot \left(\nabla \cdot \frac{Hu}{\|Hu\|_F}\right) \cdot \mathbf{n} \, dS = 0 \tag{33}$$

and, since g is arbitrary,

$$\mathbf{n}^\mathsf{T} (Hu) \mathbf{n} = 0 \,. \tag{34}$$

This is the same condition as appears in the L^2 scenario of Stein et al. [2018b].

B.2 Case 2: Variation is nonzero on $\partial\Omega$ and $\nabla\eta \propto \mathbf{t}$

In this case, the variation is nonzero on the boundary, and we also impose that the gradient of the variation is proportional to the tangent vector ($\nabla\eta = (\nabla\eta \cdot \mathbf{t})\mathbf{t}$ where $\mathbf{t}$ is unit length), and we impose the integrability condition $\oint_{\partial\Omega} \nabla\eta \cdot \mathbf{t} \, dS = 0$ ($\nabla\eta \cdot \mathbf{t}$ is continuous on the boundary). In coordinates, this gives

$$\int_{\partial\Omega} \left(\frac{1}{\|Hu\|_F} \mathbf{n}^\mathsf{T} (Hu) \mathbf{t}\right) (\mathbf{t}_i \cdot \partial_i \eta) - \eta \cdot \left(\nabla \cdot \frac{Hu}{\|Hu\|_F}\right) \cdot \mathbf{n} \, dS = 0 \,. \tag{35}$$

Integration by parts on the leftmost term (a line integral on the boundary, where the derivative is $\mathbf{t} \cdot \nabla$) results in an integral over $\partial\partial\Omega$ (the empty set) and an integral over $\partial\Omega$,

$$-\int_{\partial\Omega} \eta \cdot \left(\nabla \left(\mathbf{t}^\mathsf{T} \frac{Hu}{\|Hu\|_F} \mathbf{n}\right) \cdot \mathbf{t} + \left(\nabla \cdot \frac{Hu}{\|Hu\|_F}\right) \cdot \mathbf{n}\right) dS = 0 \,. \tag{36}$$

Finally, to convert this to a pointwise boundary condition, it remains to give a class of variations which satisfy $\nabla\eta = (\nabla\eta \cdot \mathbf{t})\mathbf{t}$ and $\oint_{\partial\Omega} \nabla\eta \cdot \mathbf{t} \, dS = 0$ and which converge to a point measure at some boundary point x_0. Let η be a smooth positive bump function on $\partial\Omega$ which is compactly supported in a ball of radius h around x_0 and which integrates to 1 over $\partial\Omega$; its gradient automatically fulfills the integrability condition. We extend η into the domain by keeping it constant in the inward normal direction over a small enough thickening of the boundary, and then by smoothly filling the interior of the domain from the inner boundary of this strip. This variation is

within the class of variations we have restricted ourselves to (even for arbitrarily small support around x_0), and still satisfies Equation (36). So, pointwise,

$$\nabla\left(\mathbf{t}^\top \frac{Hu}{\|Hu\|_F}\mathbf{n}\right)\cdot\mathbf{t} + \left(\nabla\cdot\frac{Hu}{\|Hu\|_F}\right)\cdot\mathbf{n} = 0\,. \tag{37}$$

This condition is similar to the L^2 scenario of Stein et al. [2018b], with the difference of an additional denominator appearing.

C COMPUTING MODES

To compute the modes in Fig. 5, we alternate between

$$\begin{aligned}\mathbf{u}_i \leftarrow \operatorname*{argmin}_{\mathbf{u}} \frac{1}{2}\mathbf{u}^\top\mathbf{L}\mathbf{u} + \mu\|\mathbf{H}\mathbf{u}\| \\ \text{s.t.}\mathbf{u}_i^\top\mathbf{M}\mathbf{u} = 0 \quad \forall j < i \\ \mathbf{c}_{i-1}^\top\mathbf{M}\mathbf{u} = 1\,,\ \text{and} \\ \mathbf{c}_i \leftarrow \frac{\mathbf{u}_i}{\sqrt{\mathbf{u}_i^\top\mathbf{M}\mathbf{u}_i}}\end{aligned} \tag{38}$$

until convergence, where $\mathbf{c}_0$ is initialized randomly, and $\mu = 1000000$. To find modes of the L^2 energy, we apply eigsh [Virtanen et al. 2020] to the L^2 matrix of Stein et al. [2018b].

D TRANSFORMING THE OPTIMIZATION PROBLEM INTO CONIC FORM

To optimize (21), we rearrange it into the form

$$\operatorname*{argmin}_{\mathbf{u}} \|\mathbf{H}\mathbf{u}\|_{\varphi,1} + \alpha\mathbf{u}^\top\mathbf{M}\mathbf{u} - 2\mathbf{u}_0^\top\mathbf{M}\mathbf{u} \qquad \text{s.t. } \mathbf{A}\mathbf{u} = \mathbf{b}\,, \tag{39}$$

where $\mathbf{H}$ is a matrix mapping $\mathbf{u}$ to its four intrinsic Hessian entries, and $\mathbf{M}$ is the (Voronoi) lumped mass matrix of Ω (see Supplemental Material A). The norm $\|\cdot\|_{\varphi,1}$ corresponds to taking the standard Euclidean norm (*not* the squared norm) of the vector with the four entries corresponding to each face, and then summing the norms over all faces. The nonlinear terms in the objective can be rewritten as standard conic problem constraints by introducing variables $r = 2\alpha\mathbf{u}^\top\mathbf{M}\mathbf{u}$ and $\mathbf{z} = \|(\mathbf{H}\mathbf{u})|_f\|$ (where $|_f$ represents taking the 4 entries of our linear Hessian operator that correspond to face f), and by Cholesky decomposing $2\alpha\mathbf{M} = \mathbf{L}\mathbf{L}^\top$

$$\begin{aligned}\operatorname*{argmin}_{\mathbf{u}} \mathbf{1}^\top\mathbf{z} + r - 2\mathbf{u}_0^\top\mathbf{M}\mathbf{u} \\ \text{s.t.}\begin{cases} A\mathbf{u} = \mathbf{b}, \\ (\mathbf{z})_f \geq \sqrt{\sum_i^4 \left((\mathbf{H}\mathbf{u})|_f\right)_i^2} \quad \forall f \\ r \geq \sum_j (\mathbf{L}^\top\mathbf{u})_j^2 \end{cases}\end{aligned} \tag{40}$$

The objective in this problem is linear in the variables $\mathbf{z}$, r, and $\mathbf{u}$, and the constraints correspond to a linear constraint, a quadratic conic constraint, and a rotated conic constraint respectively. It can be efficiently solved using black-box conic solvers.

E SEGMENTATION DISCRETIZATION AND OPTIMIZATION

We follow Weill–Duflos et al. [2023] by applying a smoothness energy to vertex-based functions in order to minimize a face-based quantity (v). Employing the face-based DEC Laplacian matrix $\mathbf{L}$ (using extrinsic distance between centroids) [Desbrun et al. 2005], the Voronoi mass matrix $\mathbf{M}$, and the diagonal matrix of face areas $\mathbf{M}_F$, we discretize the functional as

$$\begin{aligned}\mathbf{A}(\mathbf{u},\mathbf{v}) = \alpha(\mathbf{u}-\mathbf{u}_0)^\top\mathbf{M}(\mathbf{u}-\mathbf{u}_0) + (\mathbf{v}^2)^\top\|\mathbf{H}\mathbf{u}\|_\phi + \lambda\varepsilon\mathbf{v}^\top\mathbf{L}\mathbf{v} + \\ \frac{\lambda}{4\varepsilon}(\mathbf{v}^\top\mathbf{M}_F\mathbf{v} - 2^\top\mathbf{M}_F\mathbf{v})\,dx\,.\end{aligned} \tag{41}$$

where $\mathbf{v}^2$ is the elementwise square of $\mathbf{v}$, and $\|\cdot\|_\phi$ represents taking Frobenius norm over the 4 entries corresponding to a face. In this energy functional, λ penalizes length between segmentation regions, and α encourages solutions to closely approximate the original function. (41) is solved using an alternating scheme. For a fixed ε, we follow the procedure

$$\begin{aligned}\mathbf{v}^{(i+1)} \leftarrow \operatorname*{argmin}_{\mathbf{v}} \mathbf{v}^\top\left(\operatorname{diag}(\|\mathbf{H}\mathbf{u}^{(i)}\|_\phi) + \lambda\varepsilon\mathbf{L} + \frac{\lambda}{4\varepsilon}\mathbf{M}_F\right)\mathbf{v} - \frac{\lambda}{4\varepsilon}2^\top\mathbf{M}_F\mathbf{v} \\ \mathbf{u}^{(i+1)} \leftarrow \operatorname*{argmin}_{\mathbf{u}} \alpha\mathbf{u}^\top\mathbf{M}\mathbf{u} + \mathbf{1}^\top(\operatorname{diag}(\mathbf{v}^{(i+1)})^2\|\mathbf{H}\mathbf{u}\|_\phi - 2\alpha\mathbf{u}_0^\top\mathbf{M}\mathbf{u}\end{aligned}$$

before halving ε and resolving, repeating until ε is below a threshold ε_2. The fixed $\mathbf{u}$ iteration amounts to a quadratic solve in $\mathbf{v}$, whereas the fixed $\mathbf{v}$ iteration amounts to a conic program solve in $\mathbf{u}$. To apply the segmentation procedure to geometry, we set $\mathbf{u}$ to be the vertex coodinates $(\mathbf{x},\mathbf{y},\mathbf{z})$.

At last, we apply postprocessing to find a true segmentation of the mesh elements. We first take the mean of $\mathbf{v}$ between the faces adjacent to each edge. We then cut the mesh along edges with $\mathbf{v}$ smaller than some threshold. We then merge the resulting connected components with fewer than τ faces into their neighboring components based on a majority vote along their boundaries; τ is a hyperparameter that should be chosen dependent on mesh resolution.

F PARAMETERS

Figure 1. The Cherry parameters were $\alpha_{\text{ours}} = 6.1479$ for our energy and $\alpha_{L^1} = 16.17$ for [Stein et al. 2018b] L^1. The Tete segmentation used the parameters $\lambda = 0.25, \alpha = 1000, \varepsilon_{\text{init}} = 0.1, \varepsilon_{\text{end}} = 0.001, \varepsilon_{\text{inner loop}} = 0.00001$ and used a cut threshold of 0.5 and merge threshold τ of 20 faces.

Figure 2. The Cathedral parameters were $\alpha_{\text{ours}} = 100$ for our energy, $\alpha_{L^2} = 0.05$ for L^2, $\alpha_{L^1} = 100$ for [Stein et al. 2018b] L^1, and $\alpha_{\Delta^1} = 25$ for the L^1 edge Laplacian energy.

Figure 3. The flowed cube was run for 50 iterations at fidelity weight 100. The denoised cube ran for 10 iterations at fidelity weight 500. The hole filled cube ran for 100 iterations at fidelity weight 100.

Figure 4. For the Springer, we ran our flow for 20 iterations at fidelity weight 500. For Koala, we ran our flow for 40 iterations at fidelity weight 500.

Figure 5. For our compressed modes on the Square and Catenoid, we used $\mu = 1000000$ and $\varepsilon = 5\times10^{-10}$ for the inner loop stopping criterion.

Figure 7. For the Skull, we used $\alpha_{\text{ours}} = 44.675350189208984$ for our energy, $\alpha_{L^2} = 0.11566162109375$ for L^2, $\alpha_{L^1} = 78.5$ for [Stein et al. 2018b] L^1. For this figure, the smoothing parameters were searched for by evaluating our optimization procedure at 5 equally spaced points across a large starting range, then recursing on the 3-point subinterval whose center point admitted the closest L^2 solution to the ground truth.

Figure 13. We used the authors' command line C++ implementation of [Liu and Jacobson 2019], with a parameter $\lambda = 1$. Our Hand ran for 43 iterations at a fidelity weight 500.

Figure 14. For Nefertiti, the fidelity parameters are set to $\eta = 1250$ for 25 steps, $\eta = 500$ for 10 steps, and $\eta = 50$ for 1 step.

Figure 15. The Dodecahedron used $\lambda = 0.3, \alpha = 1000, \varepsilon_{\text{init}} = 0.1, \varepsilon_{\text{end}} = 0.001, \varepsilon_{\text{inner loop}} = 0.00001$, a cut threshold of 0.6, and a merge limit of 4. The Fandisk used $\lambda = 0.5, \alpha = 1000, \varepsilon_{\text{init}} = 0.1, \varepsilon_{\text{end}} = 0.01, \varepsilon_{\text{inner loop}} = 0.00001$, a cut threshold of 0.305, and a merge limit of 4. The Moai used $\lambda = 0.1, \alpha = 1000, \varepsilon_{\text{init}} = 0.1, \varepsilon_{\text{end}} = 0.0001, \varepsilon_{\text{inner loop}} = 0.00001$, a cut threshold of 0.7, and a merge limit of 30.

Figure 16. The Horseshoe flowed for 80 iterations at a fidelity weight 100. Evora flowed for 40 iterations at a fidelity weight 1000.

Figure 17. The Building used $\lambda = 0.03, \alpha = 0.5, \varepsilon_{\text{init}} = 0.1, \varepsilon_{\text{end}} = 0.001, \varepsilon_{\text{inner loop}} = 0.00001$, a cut threshold of 0.925, and a merge limit of 20.

Figure 18. For the Moai λ test, all parameters except λ were kept the same as in the Moai subfigure of Figure 15.

Figure 19. The segmentation for the Cow used $\lambda = 0.35, \alpha = 1000, \varepsilon_{\text{init}} = 0.1, \varepsilon_{\text{end}} = 0.0001, \varepsilon_{\text{inner loop}} = 0.0001$, a cut threshold of 0.95, and merge limit 4. The segmentation for the Stravinsky Fountain character used $\lambda = 2, \alpha = 10000, \varepsilon_{\text{init}} = 0.1, \varepsilon_{\text{end}} = 0.0001, \varepsilon_{\text{inner loop}} = 0.0001$, as well as a cut threshold of 0.95 and a merge threshold of 4.

Figure 20. The Cubesphere and UV Sphere both ran for 250 iterations at fidelity weight 500.

G SUPPLEMENTAL MATHEMATICA CALCULATIONS

A supplemental file is attached to this work giving the formula for the (intrinsic) distances and vector entries between a face's centroid and its adjacent faces' centroids as computed in Mathematica from only the edge lengths of a generic triangle.