DOI: 10.1111/cgf.13931
EUROGRAPHICS 2020 / U. Assarsson and D. Panozzo
(Guest Editors)

*Volume 39 (2020), Number 2*

# Polygon Laplacian Made Simple

Astrid Bunge[1†] Philipp Herholz[2†] Misha Kazhdan[3] Mario Botsch[1]

[1]Bielefeld University, Germany [2]ETH Zurich, Switzerland [3]Johns Hopkins University, USA

**Abstract**

*The discrete Laplace-Beltrami operator for surface meshes is a fundamental building block for many (if not most) geometry processing algorithms. While Laplacians on triangle meshes have been researched intensively, yielding the cotangent discretization as the de-facto standard, the case of general polygon meshes has received much less attention. We present a discretization of the Laplace operator which is consistent with its expression as the composition of divergence and gradient operators, and is applicable to general polygon meshes, including meshes with non-convex, and even non-planar, faces. By* virtually *inserting a carefully placed point we implicitly refine each polygon into a triangle fan, but then hide the refinement within the matrix assembly. The resulting operator generalizes the cotangent Laplacian, inherits its advantages, and is empirically shown to be on par or even better than the recent polygon Laplacian of Alexa and Wardetzky [AW11] — while being simpler to compute.*

**CCS Concepts**

• ***Computing methodologies*** → ***Mesh geometry models;*** • ***Theory of computation*** → *Computational geometry;*

## 1. Introduction

The Laplace-Beltrami operator, or Laplacian for short, plays a prominent role in geometric modeling and related fields. As a generalization of the second derivative to functions defined on surfaces it is intimately related to the notion of curvature and signal frequencies. With triangle meshes being the standard surface representation in computer graphics and geometry processing, the discretization of the Laplace operator for triangular elements has received a lot of attention over the years, with the classical cotangent discretization [Mac49, Dzi88, PP93, DMSB99] being the de-facto standard.

Discrete Laplacians for *polygon meshes* have been much less well investigated, even though polygon meshes, in particular quad meshes, are ubiquitous in modern production pipelines (and higher-order polygons are considered more noble in the 1884 novel *Flatland* [Abb84]). A straightforward approach would be to triangulate all polygons and apply the well-defined triangle-based operators. Unfortunately this approach does not work well in many cases. Polygon meshes are commonly designed to align to certain surface features and capture symmetries of the shape. Introducing an arbitrary triangulation can break these properties and lead to noticeable artifacts. Alexa and Wardetzky [AW11] proposed a discrete Laplacian that directly operates on two-manifold polygon meshes and avoids these problems. However, their operator relies on a suitable choice of parameter, which can considerably influence the results and might not be easy to find, as we show in Section 5.

In this paper we propose a surprisingly simple, yet very effective discretization of the polygon Laplace operator. Inserting a carefully placed vertex for every polygon allows us to define a refined triangulation that retains the symmetries and defines a sensible surface consistent with the polygon mesh. Using the Galerkin method, we then coarsen the cotangent Laplacian defined over the triangulation to obtain a Laplacian on the original polygon mesh, completely hiding the auxiliary points from the user by encapsulating them within the matrix assembly stage.

Our discrete Laplacian operator acts directly on functions defined on the vertices of a general polygon mesh. It works accurately and robustly even in the presence of non-planar and non-convex faces. By leveraging the cotangent Laplacian in the core of our discretization, we inherit all its benefits. The absence of tweakable parameters makes our operator intuitive and easy to use in practice. Even though our method is quite simple and easy to implement, we will make our source code freely available at `https://github.com/mbotsch/polygon-laplacian`.

## 2. Related Work

**Laplacians for triangle meshes** The main goal when constructing a discrete Laplacian is to retain as many properties from the smooth setting as possible. While the classical cotangent Laplace operator [Mac49] is negative semi-definite, symmetric, and has linear precision—meaning that the operator yields zero for linear functions defined over a planar domain—it fails to maintain the maximum principle. As a consequence, parametrizations obtained

† the first two authors contributed equally

with this discretization can suffer from flipped triangles. In contrast, the combinatorial Laplacian [Tau95, Zha04] guarantees the maximum principle while failing to have linear precision. Bobenko and Springborn [BS07] introduced a discrete Laplacian based on the intrinsic Delaunay triangulation. While guaranteeing the maximum principle, this operator is defined over an intrinsic mesh with different connectivity. Recently, an efficient data structure for the representation of these meshes has been introduced [SSC19]. The idea of intrinsic triangulations does not extend to the case of non-planar polygon meshes due to the lack of a well defined surface. Other discretizations include the Laplacian of Belkin *et al.* [BSW08] that provides point-wise convergence and the octree-based Laplacian [CLB*09] defined to support multigrid solvers. In general, Wardetzky *et al.* [WMKG07] have shown that there cannot exist a discretization that fulfills a certain set of properties for all meshes, explaining the variety of approaches in the literature.

**Laplacians for polygon meshes** Alexa *et al.* [AW11] construct a discrete Laplacian for general polygon meshes. They circumvent the problem that (non-planar) polygons in 3D do not bound a canonical surface patch by considering the projection of the polygon onto the plane that yields the largest projection area. Their polygon Laplace operator yields the gradient of this area when applied to the vertex coordinates. However, the action of this operator on the component orthogonal to this projection is defined algebraically rather than geometrically, involving parameters without an obvious interpretation. Another potential problem with this construction is a relatively large number of negative coefficients, causing the maximum principle to be violated. Xiong *et al.* [XLH11] define a discrete Laplace operator for the special case of quadrilateral meshes by averaging over both triangulations of each quad.

**Virtual refinement in geometry processing** To extend the cotangent Laplacian to polygon meshes, we refine the mesh by inserting a virtual vertex for each face and using these to tessellate each polygon into a triangle fan. Similar to recent work on Subdivision Exterior Calculus [dGDMD16], we define a *prolongation* operator expressing functions on the coarser polygonal mesh as linear combinations of the triangle hat basis functions on the finer mesh. Then, leveraging the Galerkin method [Fle84], we define the Laplacian on the polygon mesh by coarsening the Laplacian from the triangle mesh. As in the method of de Goes *et al.* [dGDMD16], this gives us the benefit of working over a refined triangle mesh (where discrete Laplacians are well understood) without incurring the computational complexity of working on a finer mesh.

**Sample applications** Applications of Laplacians include the approximation of conformal parametrizations [DMA02], mesh deformation [SCOL*04], and signal processing on meshes [CRK16], to name a few. Replacing the usual cotangent Laplacian with a Laplacian defined on polygon meshes directly enables many of these algorithms to work in this more general setting. However, the quality of the results depends on the specific construction and properties of the polygon Laplacian. In this work we compare different variants with respect to a set of applications including parametrization [Flo97, GGT06], mean curvature flow [DMA02, KSBC12], spectral analysis [LZ10], and geodesics in heat [CWW13].

## 3. Math Review

Our approach for defining a polygon Laplacian proceeds in two steps. First, we refine the polygon mesh to define a triangle mesh on which the standard cotangent Laplacian is defined. Then, to obtain a Laplacian on the initial polygon mesh, we use the Galerkin method to coarsen the cotangent Laplacian. In the following we briefly review both the derivation of Laplacians on triangle meshes [BKP*10] and the Galerkin method [Fle84].

### Laplacians on triangle meshes

Consider a triangle mesh $\mathcal{M} = (V, T)$ with vertices $V$ and triangles $T$. Let $|V|$ and $|T|$ be the numbers of vertices and triangles, respectively, and let $\{\phi_1, \dots, \phi_{|V|}\}$ be the hat basis (with $\phi_i$ the piecewise linear function that is equal to one at vertex $v_i$ and zero at all other vertices). The Laplace operator is discretized as

$$\mathbf{L} = \mathbf{M}^{-1} \cdot \mathbf{S}, \tag{1}$$

with $\mathbf{M}, \mathbf{S} \in \mathbb{R}^{|V| \times |V|}$ denoting the mass and stiffness matrices

$$\mathbf{M}_{ij} = \int_{\mathcal{M}} \phi_i \cdot \phi_j = \begin{cases} \frac{|t_{ijk}| + |t_{jih}|}{12} & \text{if } j \in \mathcal{N}(i) \\ \sum_{k \in \mathcal{N}(i)} \mathbf{M}_{ik} & \text{if } j = i \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

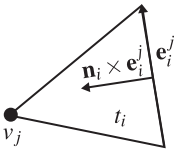
$$\mathbf{S}_{ij} = -\int_{\mathcal{M}} \langle \nabla \phi_i, \nabla \phi_j \rangle = \begin{cases} \frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2} & \text{if } j \in \mathcal{N}(i)\,, \\ -\sum_{k \in \mathcal{N}(i)} \mathbf{S}_{ik} & \text{if } j = i\,, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Here $t_{ijk}$ and $t_{jih}$ are the triangles incident on edge $(v_i, v_j)$ and $|t_{ijk}|$ and $|t_{jih}|$ are their areas; $\alpha_{ij}$ and $\beta_{ij}$ are the angles opposite edge $(v_i, v_j)$; $\mathcal{N}(i)$ denotes the index set of vertices in the one-ring neighborhood of vertex $v_i$. Note that our stiffness matrix $\mathbf{S}$ is *negative* semi-definite (like the Laplace matrix $\mathbf{L}$). This sign choice makes it consistent with the cotangent matrix in graphics. FEM literature typically uses $-\mathbf{S}$ as the stiffness matrix.

### Decomposition into divergence and gradient

Following the expression of the Laplacian as the divergence of the gradient in the continuous case, a similar expression is obtained in the discrete case. The gradient can be expressed as the matrix $\mathbf{G} \in (\mathbb{R}^3)^{|T| \times |V|}$ with

$$\mathbf{G}_{ij} = \begin{cases} \frac{\mathbf{n}_i \times \mathbf{e}_i^j}{2|t_i|} & \text{if } v_j \in t_i\,, \\ 0 & \text{otherwise}\,, \end{cases}$$




where $\mathbf{n}_i$ is the unit outward normal of triangle $t_i$ and $\mathbf{e}_i^j$ is the counter-clockwise oriented edge of triangle $t_i$ opposite vertex $v_j$. Then the divergence is

$$\mathbf{D} = -\mathbf{G}^{\mathsf{T}} \cdot \widetilde{\mathbf{M}}, \tag{4}$$

where $\widetilde{\mathbf{M}}$ is a block diagonal matrix whose $i$-th block consists of the $3 \times 3$ identity matrix multiplied by the area of the $i$-th triangle.

The product of divergence and gradient gives the stiffness matrix

$$\mathbf{S} = \mathbf{D} \cdot \mathbf{G} = -\mathbf{G}^{\mathsf{T}} \cdot \widetilde{\mathbf{M}} \cdot \mathbf{G} \tag{5}$$

and the Laplacian becomes

$$\mathbf{L} = -\mathbf{M}^{-1} \cdot \mathbf{G}^{\mathsf{T}} \cdot \widetilde{\mathbf{M}} \cdot \mathbf{G}. \tag{6}$$

In practice, the mass matrix is often approximated by a diagonal (lumped) mass matrix with the $i$-th diagonal entry in the lumped matrix set to the sum of entries in the $i$-th row of the original matrix. This makes inversion of $\mathbf{M}$ straightforward.

### The Galerkin method

Assume that we are given two finite-dimensional nesting function spaces $\mathcal{F}^c \subset \mathcal{F}^f$. Given a *prolongation* operator $P$ injecting the coarser space into the finer, $P\colon \mathcal{F}^c \hookrightarrow \mathcal{F}^f$, and given a symmetric positive semi-definite operator $Q$ defining a quadratic energy on the space of functions, we can define a symmetric positive semi-definite operator on the space $\mathcal{F}^f$ through restriction

$$Q^f(\phi,\psi) := Q(\phi,\psi), \qquad \forall \phi,\psi \in \mathcal{F}^f. \tag{7}$$

In a similar manner, we can define a symmetric positive semi-definite operator $Q^c$ on the space $\mathcal{F}^c$.

The Galerkin method tells us that the operators are related by

$$Q^c = P^* \circ Q^f \circ P, \tag{8}$$

where $P^*$ is the dual of $P$. In particular, given bases for $\mathcal{F}^c$ and $\mathcal{F}^f$ and letting $\mathbf{P}$, $\mathbf{Q}^c$, and $\mathbf{Q}^f$ be the matrices representing the operators with respect to these bases, we have

$$\mathbf{Q}^c = \mathbf{P}^{\mathsf{T}} \cdot \mathbf{Q}^f \cdot \mathbf{P}. \tag{9}$$

## 4. Polygon Laplacian

When working with general polygon meshes, the finite elements method cannot be applied directly for two reasons. First, when the polygon is not planar, it is not clear what the underlying surface is over which functions should be integrated. Second, even for simple planar polygons, it is not clear how to associate basis functions with the vertices of the mesh.

A simple approach would be to triangulate all the polygons, defining a piecewise-linear integration domain, and use the hat basis functions to define the finite elements. This would have the advantage of defining a finite elements system whose dimension equals the number of vertices in the input mesh. Unfortunately, the introduction of diagonal edges would also break the symmetry structure of the polygonization (e.g., in quad meshes where edges are aligned with principal curvature directions).

An alternate approach would be to refine the polygon mesh by introducing a new vertex in the middle of each face. This would preserve the symmetry structure, but would come at the cost of an increase in the finite elements system dimension. Also, as we show in Section 5, such an approach can result in poor performance due to the introduction of negative edge weights in the stiffness matrix. (For example, refining a rectangle by adding the mid-point creates two triangles with angles larger than $\pi/2$.)

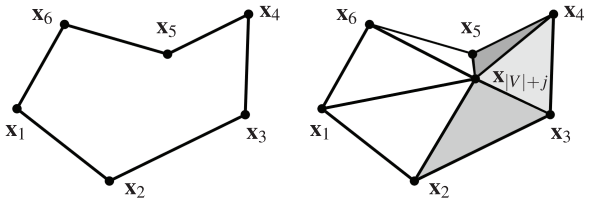



**Figure 1:** *Spanned triangle fan on the virtual mesh after inserting the virtual vertex $\vec{x}_{|V|+j}$ at the $j$-th face.*

We propose an in-between approach – introducing a *virtual* vertex in the middle of each face, expressed as the affine combination of the face's vertices. Naively, the new vertex produces a refined triangle mesh with a finite elements system given by the hat basis functions, as before. However, we then coarsen the refined system, expressing the vertex functions on the coarse mesh as linear combinations of the hat basis functions on the finer one.

This new system has dimension equal to the number of vertices in the original polygon mesh (preserving the finite elements system dimension) and has the property that basis functions have overlapping support only if the associated vertices lie on a common face (defining a sparse system that preserves the symmetry structure). A further advantage of our approach is that we easily obtain a consistent factorization of the stiffness matrix as the product of divergence and gradient matrices.

### 4.1. Construction of the finite elements system

Given a *polygon* mesh $\mathcal{M} = (V,F)$, we construct our finite elements system by defining a *refined triangle* mesh $\mathcal{M}^f = (V^f, T^f)$. Vertices in the refined mesh, $V^f$, are obtained by introducing a new vertex, $v_{|V|+j}$, for every face $f_j \in F$ and setting the position of $v_{|V|+j}$ to an affine combination of the positions of vertices in $f_j$

$$\vec{x}_{|V|+j} = \sum_{v_i \in f_j} w_{ji} \vec{x}_i, \quad \text{with} \quad \sum_{v_i \in f_j} w_{ji} = 1. \tag{10}$$

Triangles in the refined mesh, $T^f$, are obtained by replacing each face $f_j \in F$ with the triangle fan connecting the inserted vertex $v_{|V|+j}$ to the edges in $f_j$, as can be seen in Figure 1.

Using the refined triangle mesh, we can define the stiffness matrix, $\mathbf{S}^f \in \mathbb{R}^{|V^f| \times |V^f|}$, using the standard cotangent weights of Equation (3). Aggregating the affine weights, we get the *prolongation* matrix $\mathbf{P} \in \mathbb{R}^{|V^f| \times |V|}$, with

$$\mathbf{P}_{ij} = \begin{cases} 1 & \text{if } i = j, \\ w_{ki} & \text{if } i = |V| + k \text{ and } v_j \in f_k, \\ 0 & \text{otherwise}. \end{cases} \tag{11}$$

And finally, using the Galerkin method, we obtain an expression for the coarsened polygonal stiffness matrix as

$$\mathbf{S} = \mathbf{P}^{\mathsf{T}} \cdot \mathbf{S}^f \cdot \mathbf{P}. \tag{12}$$

We use the same approach to obtain the coarsened polygonal mass matrix $\mathbf{M}$, but in addition to restricting the refined triangle mass matrix $\mathbf{M}^f$, we also lump it to a diagonal matrix:

$$\mathbf{M} = \text{lump}\left(\mathbf{P}^\mathsf{T} \cdot \mathbf{M}^f \cdot \mathbf{P}\right) \quad \text{with} \tag{13}$$

$$\text{lump}(\mathbf{M})_{ij} = \begin{cases} \sum_k \mathbf{M}_{ik} & \text{if } i = j, \\ 0 & \text{otherwise}, \end{cases} \tag{14}$$

setting the diagonal entry to the sum of the entries in the row.

We note that our stiffness matrix has non-zero entries if the corresponding vertices share a face, not just an edge. We also note that solving a linear system using our operators is different from solving the system on the refined mesh and only using the solution values at the original (i.e., non-virtual) vertices. Our approach corresponds to solving the system using a coarser function space. The alternative corresponds to sub-sampling a solution obtained over a refined function space, which could result in aliasing.

### 4.2. Choice of virtual vertex position

Choosing different virtual vertex positions, we provide a framework for defining a whole family of polygon Laplace operators. While many choices are possible we would like the virtual vertex to fulfill a set of properties in order to yield a geometric Laplacian:

- The vertex should be efficient to compute and uniquely defined.
- Since the choice of virtual vertex defines the surface of the polygon, we would like to find a point giving small surface area.
- For planar polygons, the vertex should be inside the polygon.
- To achieve linear precision, the virtual vertex should be an affine combination of polygon vertices (see Equation (10)).

A straightforward choice is to use the *centroid* of the polygon vertices. However, for non-convex (planar) polygons this point can be located outside the polygon. Moreover, it will be biased by an uneven distribution of polygon vertices.

Another choice is to find a point *minimizing the area* of the induced triangle fan, which is related to the minimization of the Dirichlet energy. Unfortunately, this point is not uniquely defined. For example, for convex planar polygons, the total area will be identical for every virtual vertex inside the polygon. Furthermore, since area is non-linear in the position of the virtual vertex, less efficient iterative solvers, e.g. Newton's method, are required to compute the position of the virtual vertex.

Instead, we opt for the minimizer of the sum of *squared* triangle areas of the induced triangle fan. Introducing a virtual vertex with position $\vec{x}_f$ into an $n$-gon with vertex positions $(\vec{x}_1, \dots, \vec{x}_n)$ allows us to define the triangle fan with $n$ triangles $(\vec{x}_i, \vec{x}_{i+1}, \vec{x}_f)$, where indices are interpreted modulo $n$. The position of the virtual vertex, $\vec{x}_f \in \mathbb{R}^3$, is then defined as the minimizer

$$\vec{x}_f = \arg\min_{\vec{x}} \sum_{i=1}^{n} \text{area}\left(\vec{x}_i, \vec{x}_{i+1}, \vec{x}_f\right)^2. \tag{15}$$

Compared to the minimizer of the area, the squared area minimizer has two advantages. First, the solution is unique even for planar convex polygons. Second, using the squared area, the objective function becomes quadratic in $\mathbf{x}_f$. Also, in contrast to the centroid, the minimizer of the squared area tends to be positioned in the interior of the polygon, even when the polygon is not convex.

While the position of our virtual vertex is unique, its expression as the linear combination of polygon vertices usually is not. Since linear precision requires $\vec{x}_f$ to be an affine combination of the polygon's vertices (see Section 4.3), finding the position of the virtual vertex can be formulated as a minimization directly over the weights $\vec{w} = (w_1, \dots, w_n)^\mathsf{T}$:

$$\begin{aligned} \vec{w}^f &= \arg\min_{\vec{w}} \sum_{i=1}^{n} \text{area}\left(\vec{x}_i, \vec{x}_{i+1}, \sum_{j=1}^{n} w_j \vec{x}_j\right)^2 \\ &\text{such that } \sum_{j=1}^{n} w_j = 1. \end{aligned} \tag{16}$$

Noting that when $n > 3$ the system is under-constrained (with multiple affine weights defining the same unique squared-area minimizer $\mathbf{x}_f$), we add the constraint that, of all the weights $\vec{w}$ defining the minimizing position $\mathbf{x}_f$, we prefer the one with minimal $L_2$-norm $\|\vec{w}\|$. This encourages the weights to be as uniform as possible, allowing each polygon vertex to contribute equally. These affine weights can be obtained by solving a single linear system, derived in Appendix A.

It is tempting to add non-negativity constraints $w_j \geq 0$ to Equation (16) since this would yield *convex weights* that guarantee a maximum principle for the virtual vertex and prevent negative coefficients in the coarsened stiffness matrix $\mathbf{S}$ (see Equation (12)) as long as they do not appear in $\mathbf{S}^f$. However, this comes at the cost of solving a quadratic program with inequality constraints.

We compare our choice of virtual vertex—minimizing the sum of squared triangle areas through affine weights—to the other options (centroid, min. area, convex weights) in Section 5.7.

### 4.3. Properties of the operator

One goal of our construction is to preserve the beneficial numerical properties of the cotangent Laplacian. **Symmetry** and **negative semi-definiteness** follow directly from our construction of the stiffness matrix $\mathbf{S} = \mathbf{P}^\mathsf{T} \cdot \mathbf{S}^f \cdot \mathbf{P}$, since the refined (cotangent) stiffness matrix $\mathbf{S}^f$ has these properties and since the prolongation matrix $\mathbf{P}$ has full rank. (Note that Wardetzky and Alexa [WMKG07, AW11] aim for *positive* definiteness, since they define their geometric Laplacian as the negative of ours.)

For **linear precision** we require $(\mathbf{S} \cdot \vec{u})_i = 0$ whenever vertex $v_i$ is not part of the boundary, all incident polygons $f_j \ni v_i$ are coplanar, and the values of $\vec{u}$ in the one-ring of $v_i$ are obtained by sampling a linear function on the plane. To see that this is the case, we note that by constraining ourselves to use affine weights in defining the prolongation, we ensure that prolonging $\vec{u}$ to the finer mesh, the values of $\vec{u}^f = \mathbf{P} \cdot \vec{u}$ sample a linear function at all vertices of the fan triangulations incident to $v_i$. Since the cotangent Laplacian has linear precision, this implies that $\mathbf{S}^f \cdot \vec{u}^f$ is zero at $v_i$ *and* all virtual centers $v_{|V|+j}$ with $f_j \ni v_i$. Since these are precisely the entries at which the $i$-th column of $\mathbf{P}$ is non-zero, it follows that $(\mathbf{P}^\mathsf{T} \cdot \mathbf{S}^f \cdot \vec{u}^f)_i = 0$. Or, equivalently, $(\mathbf{S} \cdot \vec{u})_i = 0$.

It follows that if the initial mesh $\mathcal{M}$ is a **triangle mesh**, then the derived stiffness matrix $\mathbf{S}$ is the standard cotangent Laplacian. This

is because using affine weights, the finite elements basis defined on the coarse mesh through prolongation is *precisely* the hat basis.

The property of positive off-diagonal values does not hold for the cotangent Laplacian and consequently cannot extend to our construction, resulting in the potential violation of the **maximum principle**. The lack of positivity is also present in the definition of [AW11], however, our operator tends to contain fewer negative off-diagonal coefficients (see Section 5).

### 4.4. Implementation

Implementing our system is simple, especially if code for the computation of the cotangent Laplacian is already available. Our implementation is based on the PMP-Library [SB19]. Algorithm 1 illustrates the procedure. Initially we set the prolongation matrix $\mathbf{P}$ to the identity of size $|V| \times |V|$ (line 3), reflecting the fact that all original vertices appear in the refined mesh. We then loop over all faces, collecting all vertex positions of the polygon, finding affine weights for the optimal virtual vertex, and constructing this point (lines 5–7, Section 4.2). Next we compute the area and cotangent weights for the resulting triangle fan and aggregate them in $\mathbf{M}^f$ and $\mathbf{S}^f$ (lines 9–10, Section 3). The final operators are constructed by multiplying the matrices $\mathbf{M}^f$ and $\mathbf{S}^f$ defined on the refined mesh with the prolongation matrix and its transpose (line 12, Section 4.1).

**Algorithm 1:** Assemble polygon stiffness and mass matrix

**Data:** Mesh vertices $V$ and faces $F$
**Result:** Mass matrix $\mathbf{M}$, stiffness matrix $\mathbf{S}$.

1 $\mathbf{S}^f = \mathbf{0}^{|V^f|\times|V^f|}$
2 $\mathbf{M}^f = \mathbf{0}^{|V^f|\times|V^f|}$
3 $\mathbf{P} = \mathbf{I}^{|V|\times|V|}$
4 **foreach** $f \in F$ **do**
5 $\quad \mathbf{X} \leftarrow \text{getVertexPositions}(f)$;
6 $\quad \mathbf{w} \leftarrow \text{findVirtualVertexWeights}(\mathbf{X})$;
7 $\quad \mathbf{x} \leftarrow \mathbf{X}^\mathsf{T} \cdot \mathbf{w}$;
8 $\quad \mathbf{P} \leftarrow \text{appendWeightRow}(\mathbf{P}, f, \vec{w})$;
9 $\quad \mathbf{M}^f \leftarrow \mathbf{M}^f + \text{buildTriangleFanMassMatrix}(f, \mathbf{X}, \mathbf{x})$;
10 $\quad \mathbf{S}^f \leftarrow \mathbf{S}^f + \text{buildTriangleFanCotan}(f, \mathbf{X}, \mathbf{x})$;
11 **end**
12 **return** $\mathbf{M} = \mathbf{P}^\mathsf{T} \cdot \mathbf{M}^f \cdot \mathbf{P}, \quad \mathbf{S} = \mathbf{P}^\mathsf{T} \cdot \mathbf{S}^f \cdot \mathbf{P}$

### 4.5. Gradient and divergence

We are also able to factor the stiffness matrix as the product of divergence and gradient matrices. As described in Section 3, we can obtain a matrix expression for the gradient and divergence operators over the refined mesh, $\mathbf{G}^f$ and $\mathbf{D}^f$ and use those to factor the refined triangle stiffness matrix

$$\mathbf{S}^f = \mathbf{D}^f \cdot \mathbf{G}^f. \tag{17}$$

Coarsening, we obtain a factorization of the polygon stiffness matrix in terms of coarse polygon divergence and gradient matrices

$$\mathbf{S} = \underbrace{\mathbf{P}^\mathsf{T} \cdot \mathbf{D}^f}_{=\mathbf{D}} \cdot \underbrace{\mathbf{G}^f \cdot \mathbf{P}}_{=\mathbf{G}}. \tag{18}$$

Note that the derived gradient operator, $\mathbf{G} = \mathbf{G}^f \cdot \mathbf{P}$, is a map from functions defined on the vertices of the *original* polygon mesh to tangent vector fields that are constant on triangles of the *refined* mesh. These refined triangles, however, can uniquely be identified with half-edges of the original polygon mesh, so that the refined triangles never have to be constructed explicitly. Hence, the gradient operator maps from function values at vertices to vectors at half-edges (and conversely for the divergence operator, $\mathbf{D} = \mathbf{P}^\top \cdot \mathbf{D}^f$ ).

### 4.6. Finite Elements Exterior Calculus

An alternative approach is to define differential operators by extending Finite Elements Exterior Calculus to polygon meshes. We do this by using our coarsened basis to define Whitney bases for higher-order forms and then using these higher-order bases to define the Hodge star operators.

**Recall** Given a basis of zero-forms $\{\psi_i\}$ forming a partition of unity, one can define Whitney bases [Whi57, AFW06] for 1-forms $\{\psi_{ij}\}$ (with $i<j$ and $\text{supp}(\psi_i)\cap\text{supp}(\psi_j)\neq\emptyset$) and 2-forms $\{\psi_{ijk}\}$ (with $i<j<k$ and $\text{supp}(\psi_i)\cap\text{supp}(\psi_j)\cap\text{supp}(\psi_k)\neq\emptyset$) by setting:

$$\psi_{ij} = \psi_i \cdot d\psi_j - \psi_j \cdot d\psi_i\,,$$
$$\psi_{ijk} = 2\left(\psi_i \cdot d\psi_j \wedge \psi_k - \psi_j \cdot d\psi_i \wedge d\psi_k + \psi_k \cdot d\psi_i \wedge d\psi_j\right).$$

The exterior derivatives are then defined using the combinatorics

$$\mathbf{d}^0_{(ij)k} = \begin{cases} -1 & i=k \\ 1 & j=k \\ 0 & \text{otherwise} \end{cases}, \quad \mathbf{d}^1_{(ijk)(lm)} = \begin{cases} 1 & i=l,\ j=m \\ 1 & j=l,\ k=m \\ -1 & i=l,\ k=m \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

and the discrete Hodge stars are defined using the geometry

$$\star^0_{ij} = \langle\langle \psi_i, \psi_j \rangle\rangle, \tag{20}$$
$$\star^1_{(ij)(kl)} = \langle\langle \psi_{ij}, \psi_{kl} \rangle\rangle, \tag{21}$$
$$\star^2_{(ijk)(lmn)} = \langle\langle \psi_{ijk}, \psi_{lmn} \rangle\rangle, \tag{22}$$

with $\langle\langle\cdot,\cdot\rangle\rangle$ denoting the integral of the (dot-)product of $k$-forms. Using the hat basis on a triangle mesh, the basis functions are linear within each triangle so computing the coefficients reduces to integrating quadratic polynomials over a triangle.

**Prolonging higher-order forms** Given the hat basis on the refined triangle mesh $\{\phi^f_1,\ldots,\phi^f_{|V^f|}\}$, defining a prolongation operator $\mathbf{P}$ is equivalent to defining a coarsened basis $\{\phi_1,\ldots,\phi_{|V|}\}$ on the triangle mesh, with $\phi_j = \sum_i \mathbf{P}_{ij}\phi^f_i$.

As both bases form a partition of unity, we can define Whitney bases for higher-order forms. In doing so we get:

$$\phi_{ij} = \sum_{k,l} \mathbf{P}_{ki}\mathbf{P}_{lj}\phi^f_{kl} \quad \text{and} \quad \phi_{ijk} = \sum_{l,m,n} \mathbf{P}_{li}\mathbf{P}_{mj}\mathbf{P}_{nk}\phi^f_{lmn}, \tag{23}$$

which gives prolongation operators for 0-, 1-, and 2-forms:

$$\mathbf{P}^0_{ij} = \mathbf{P}_{ij}, \tag{24}$$
$$\mathbf{P}^1_{(ij)(kl)} = \mathbf{P}_{ik} \cdot \mathbf{P}_{jl}, \tag{25}$$
$$\mathbf{P}^2_{(ijk)(lmn)} = \mathbf{P}_{il} \cdot \mathbf{P}_{jm} \cdot \mathbf{P}_{kn}. \tag{26}$$

This allows us to define Hodge star operators for the coarsened basis through prolongation:

$$\star^k = (\mathbf{P}^k)^\mathsf{T} \cdot \star^{k,f} \cdot \mathbf{P}^k, \tag{27}$$

where $\star^k$ is the discrete Hodge star for $k$-forms defined using the coarsened basis and $\star^{k,f}$ is the discrete Hodge star for $k$-forms defined using the refined hat basis.

In particular, this gives a factorization of the stiffness matrix as $\mathbf{S} = -(\mathbf{d}^0)^\mathsf{T} \cdot \star^1 \cdot \mathbf{d}^0$, with gradients represented in terms of differences across polygon edges/diagonals. The minus sign is due to our design choice of a negative semi-definite stiffness matrix, as discussed in Section 3.

## 5. Evaluation

We evaluate our Laplacian in a variety of different geometry processing operations, applied to a selection of polygon meshes (Figures 2 and 3). These results are compared to two other methods. First, we triangulate each (potentially non-planar, non-convex) $n$-gon into $n-2$ triangles without inserting an extra point, and then use the standard cotangent Laplacian for triangle meshes. Using the dynamic-programming approach of Liepa [Lie03] we find the polygon triangulation that minimizes the sum of squared triangle areas (similar in concept to our squared area minimization). We also experimented with the triangulation that maximizes the minimum triangle angle (similar to planar Delaunay triangulations). While the latter yields better-behaved cotangent weights, it tends to produce flips or fold-overs for non-convex polygons, so we used the former for most experiments. Second, we compare our results to the ones obtained with the polygon Laplacian of Alexa and Wardetzky [AW11], based on their original implementation, using various values for their parameter $\lambda$ that assigns weight to the component perpendicular to the projection direction.

### 5.1. Conformalized mean curvature flow

A common approach for smoothing meshes is to use implicit integration to solve the diffusion equation [DMSB99]. In our application we use the conformalized Mean Curvature Flow introduced by Kazhdan *et al.* [KSBC12], which obtains the coordinates of the mesh vertices at the next time-step, $\mathbf{X}^{t+\varepsilon}$ from the coordinates at the current time-step, $\mathbf{X}^t$, iteratively solving the linear system

$$\left(\mathbf{M}^t - \varepsilon \mathbf{S}^0\right) \vec{X}^{t+\varepsilon} = \mathbf{M}^t \vec{X}^t \tag{28}$$

with $\varepsilon$ the time-step, $\mathbf{S}^0$ the initial stiffness matrix, and $\mathbf{M}^t$ the mass matrix at time $t$. After each iteration the mesh is translated back to the origin and re-scaled to its original surface area. Figure 4 demonstrates the resilience of our Laplacian to noise and non-planar as well as non-convex faces. The flow can recover the spherical shape after one iteration and converges after 10 iterations.

Figure 5 shows an example of this flow after one and ten iterations respectively. The mean curvature is color-coded and shows that the mesh correctly converges to a sphere when using our operator. This example shows that even extreme differences in polygon scale are handled correctly. The results of Alexa and Wardetzky's

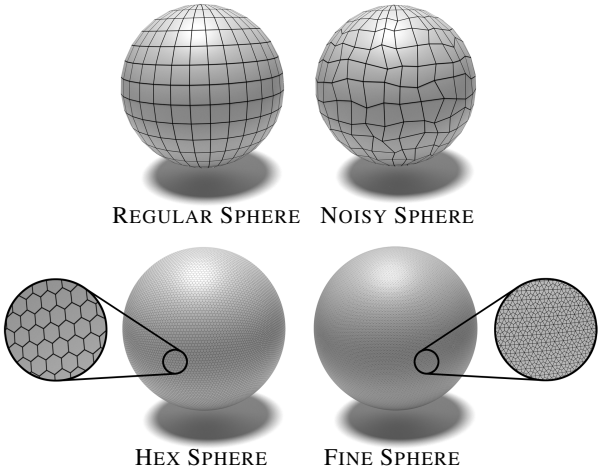



**Figure 2:** *Spherical meshes used for testing the accuracy of spherical harmonics and mean curvature.*

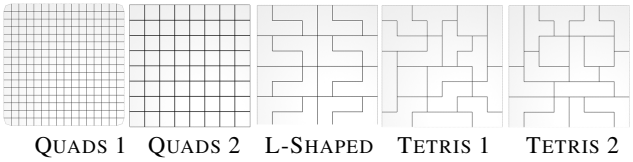



**Figure 3:** *Planar meshes used for the evaluation of geodesic distances, including non-convex and non-star shaped tessellations.*

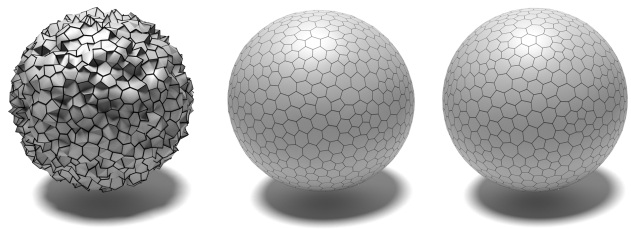


**Figure 4:** *Stress test for smoothing on a noisy sphere (left). After one (center) and ten iterations (right).*

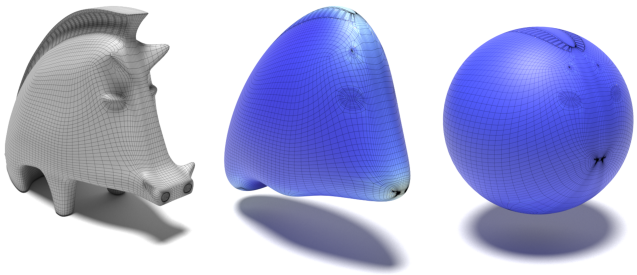


**Figure 5:** *Visualization of the mean curvature flow on a quad mesh. Mean curvature is color coded.*

| | Ours | [AW11] |
|---|---|---|
| Figure 4 | $5.075 \times 10^{-3}$ | $5.417 \times 10^{-3}$ |
| Figure 5 | $2.271 \times 10^{-3}$ | $2.69 \times 10^{-3}$ |
| Figure 6 | $1.157 \times 10^{-3}$ | $1.204 \times 10^{-3}$ |

**Table 1:** *Conformal distortion of conformal spherical [KSBC12] and spectral free-boundary [MTAD08] parametrizations.*

| | | | [AW11] | | |
|---|---|---|---|---|---|
| Mesh | Ours | [Lie03] | $\lambda = 0.1$ | $\lambda = 0.5$ | $\lambda = 1$ |
| REGULAR SPHERE | **0.0168** | 0.0469 | 0.0290 | 0.0290 | 0.0290 |
| NOISY SPHERE | **0.0469** | 0.1053 | 0.0562 | 0.0814 | 0.1551 |
| HEX SPHERE | **0.0016** | 0.3711 | 0.0023 | 0.0038 | 0.0075 |
| FINE SPHERE | 0.1334 | 0.0623 | 0.0279 | **0.0107** | 0.0141 |

**Table 2:** *Root-mean-square error (RMSE) of the curvature deviation for different meshes of the unit sphere.*

operator are qualitatively very similar, however, analyzing the conformal error shows a small advantage for our method (see Table 1). We evaluated the conformal distortion by computing the root mean square errors of area-weighted angle differences between the mesh and its spherical parametrization. We opted for this measure because it is not obvious how to compute conformal distortion for non-planar polygons.

### 5.2. Mean curvature estimation

Noting that the Laplacian of the coordinate function is the mean-curvature normal vector, we can use our Laplace operator to approximate the mean curvature $H$ at vertex $v_i$

$$H(v_i) := \frac{1}{2} \left\| \left( \mathbf{L}\vec{X} \right)_i \right\| \cdot \operatorname{sign}\left( \left\langle \left( \mathbf{L}\vec{X} \right)_i, \vec{n}_i \right\rangle \right) \tag{29}$$

with $\vec{X}$ the matrix of vertex coordinates and $\vec{n}_i$ the normal at $v_i$.

Since the mean curvature is 1 at every point on the unit sphere, we can measure the accuracy of our Laplacian by comparing the estimated mean curvature to the true one. Table 2 gives the root mean square error (RMSE), comparing curvature estimates over different polygonizations of the sphere, and using different definitions of the Laplace operator. As the table shows, our operator outperforms existing methods on most spherical meshes.

### 5.3. Parametrization

We compare our operator to [AW11] with respect to conformal parametrization based on Mullen *et al.*'s [MTAD08] spectral free-boundary parametrization. Figure 6 illustrates that both operators perform well for this quad mesh. In this case we obtain a conformal distortion (c.f. Section 5.1) of $1.157 \times 10^{-3}$ while the method of [AW11] achieves $1.204 \times 10^{-3}$. For [AW11] we systematically tried to find the best parameter $\lambda$ and consistently observed very similar behavior with a slightly lower conformal distortion for our operator, across a variety polygon meshes.

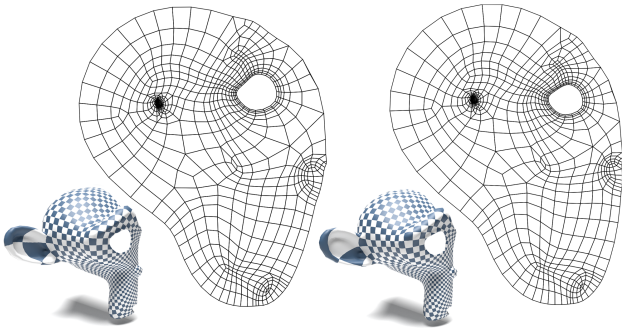


**Figure 6:** *Parametrization of the right half of a quadrangulated monkey head. The result of [AW11] (left) is similar to ours (right), but our operator has slightly lower conformal distortion.*

### 5.4. Reproducing the spherical harmonics

The eigenfunctions of the Laplacian form an orthonormal basis known as the "manifold harmonics" [VL08]. In the case that the surface is a sphere, these eigenfunctions are known to be the spherical harmonics. We evaluate the quality of our Laplacian by measuring the extent to which the true spherical harmonics are eigenvectors of the Laplacian.

We know that the spherical harmonic $Y_l^m \colon S^2 \to \mathbb{R}$ is an eigenfunction of the Laplacian with eigenvalue $-l \cdot (l+1)$. Sampling $Y_l^m$ at the vertices of the spherical mesh, we obtain $\vec{y}_l^m \in \mathbb{R}^{|V|}$. We measure the accuracy of our Laplacian by evaluating

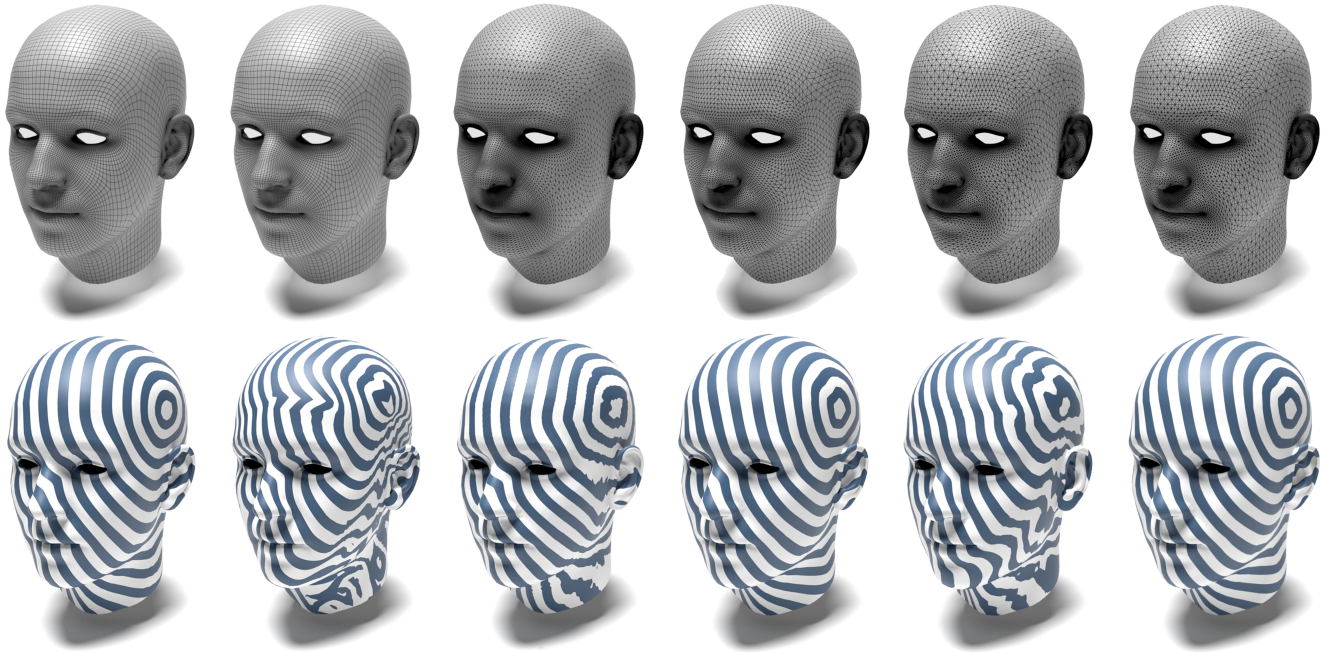
$$\left\| \vec{y}_l^m + \frac{1}{l \cdot (l+1)} \mathbf{L} \cdot \vec{y}_l^m \right\|_{\mathbf{M}}^2, \tag{30}$$

with the $L_2$-norm computed relative to the mass matrix $\mathbf{M}$.

Table 3 compares the spectral error, giving the sum of errors over the spherical harmonics in the first nine frequencies ($1 \leq l \leq 9$). As the table shows, in this experiment we are unable to reproduce the accuracy of Alexa and Wardetzky's Laplacian. However, our construction does not require selecting a parameter on a case-by-case basis as in [AW11].

| | | | [AW11] | | |
|---|---|---|---|---|---|
| Mesh | Ours | [Lie03] | $\lambda = 0.1$ | $\lambda = 0.5$ | $\lambda = 1$ |
| REGULAR SPHERE | 0.0393 | 0.0200 | 0.0599 | 0.0220 | **0.0175** |
| NOISY SPHERE | 0.0643 | 0.0722 | 0.0969 | **0.0589** | 0.1366 |
| HEX SPHERE | **7.41e-7** | 0.0037 | 9.93e-6 | 1.46e-6 | 1.13e-6 |
| FINE SPHERE | 0.0003 | 6.73e-5 | 0.0005 | **6.48e-5** | 8.98e-5 |

**Table 3:** *We measure the extent to which the spherical harmonics of frequencies $1 \leq l \leq 9$ are eigenvectors, with eigenvalue $-l \cdot (l+1)$, of the Laplacian on spherical meshes.*



a) ours b) [AW11] c) refinement d) max-angle e) min-area f) intrinsic Delaunay

**Figure 7:** *Computing geodesic distances on a quad mesh (top left) following the approach of Crane* et al. *[CWW13], using our operator (a) and Alexa and Wardetzky's (b, using $\lambda = 0.5$). Images (c–f) show results using different triangulation strategies: refining the mesh by inserting the virtual point (c), triangulating polygons to maximize the minimum angle (d), and triangulating polygons to minimize squared triangle areas (e). To improve the results of (e), we employ the Laplacian based on the intrinsic Delaunay triangulation [BS07] (f). The triangles of (d) are already Delaunay, therefore using the intrinsic triangulation does not further improve the result in (d).*

### 5.5. Geodesics in Heat

In [CWW13], Crane *et al.* proposed the heat method for computing geodesic distances from a selected vertex $v_i$ to all others on the mesh. Let $\mathbf{D}$ be the divergence operator, $\mathbf{G}$ the gradient operator and $\vec{e}_i \in \mathbb{R}^{|V^c|}$ the $i$-th unit vector. The geodesic distances are computed in four steps.

First, the impulse function $\vec{e}_i$ is diffused for a time-step of $\varepsilon$ by solving for $\vec{u} \in \mathbb{R}^{|V|}$ such that

$$(\mathbf{M} - \varepsilon\mathbf{S})\,\vec{u} = \mathbf{M} \cdot \vec{e}_i, \tag{31}$$

where we set $\varepsilon$ to the squared mean of edge lengths as suggested in [CWW13]. Next, the gradients of $\vec{u}$ are computed and normalized to have unit length, resulting in a vector field $\vec{g} \in \mathbb{R}^{|T^f|\times 3}$ with

$$\vec{g}_i = -\frac{(\mathbf{G}\cdot\vec{u})_i}{\left\|(\mathbf{G}\cdot\vec{u})_i\right\|}. \tag{32}$$

Finally, the geodesic distance, $\vec{d} \in \mathbb{R}^{|V|}$ is computed by solving for the scalar field whose gradient best matches $\vec{g}$

$$\mathbf{S}\cdot\vec{d} = \mathbf{D}\cdot\vec{g} \tag{33}$$

and additively offsetting so that it has value zero at $v_i$. Our polygon Laplacian offers a natural decomposition into divergence and gradient (c.f. Section 4.5). Crane *et al.* [CWW13] demonstrate how to obtain a normalized gradient for computing geodesic distances using the operator of Alexa and Wardetzky.

We qualitatively compare the results using our operator, the one from [AW11], and several polygon triangulation strategies in Figure 7. (a) Our construction gives a result with considerably fewer artifacts compared to the other approaches. (b) The Laplace operator of Alexa and Wardetzky fails independent of the choice of $\lambda$. (d) Triangulating polygons to maximize the minimal angle also gives good results, but this approach is not suitable for arbitrary meshes since it fails on non-convex polygons. (e) Minimum-area triangulations avoid this problem, but give worse results due to poor triangle shapes. (f) Combining minimum-area triangulations with the Laplacian based on the intrinsic Delaunay triangulation [BS07] fixes this problem, but is more complex to compute. In (c) we show the result obtained by using the cotangent Laplacian on the mesh explicitly refined by inserting the virtual vertices (this is equivalent to $\mathbf{S}^f$). Our construction is clearly different from just refining polygons.

The quality of geodesics is linked to the number of negative off-diagonal coefficients in the stiffness matrix. Analyzing the ratio of negative off-diagonal entries for Figure 7 confirms this correlation:

| ours | [AW11] | c) | d) | e) | f) |
|---|---|---|---|---|---|
| 11% | 21% | 17% | 5% | 15% | 0% |

We consistently observe a significantly smaller number of negative off-diagonal coefficients in our operator as compared to [AW11].

| Mesh | Ours | [BS07] | [AW11] $\lambda = 0.1$ | [AW11] $\lambda = 0.5$ | [AW11] $\lambda = 1$ |
|---|---|---|---|---|---|
| QUADS 1 | 0.0265 | 0.0394 | **0.0179** | 0.0415 | 0.1206 |
| QUADS 2 | **0.0356** | 0.0856 | 0.0403 | 0.0469 | 0.1233 |
| L-SHAPED | **0.0574** | 0.0736 | 0.4216 | 0.1115 | 1.6531 |
| TETRIS 1 | 0.2183 | **0.1408** | 0.4521 | 0.2304 | 0.2483 |
| TETRIS 2 | 0.0821 | **0.0665** | 0.4381 | 0.1177 | 0.1852 |

**Table 4:** *Root-mean-square error of computed geodesic distances for the different planar tessellations shown in Figure 3.*

| Mesh | Affine | Convex | Centroid | Abs. Area | [AW11] | [Lie03] |
|---|---|---|---|---|---|---|
| HEXAGON | 55 | 142 | 13 | 357 | 52 | 21 |
| QUADS | 171 | 573 | 50 | 1460 | 240 | 82 |
| HEXAGON | 472+25 | 483+25 | 476+26 | 469+25 | 477+26 | 77+8 |
| QUADS | 596+35 | 596+35 | 599+35 | 600+36 | 599+35 | 498+29 |

**Table 5:** *Timing (in ms) for constructing the Laplace matrix (top) and solving the linear system (bottom). The latter is split into the time needed for Cholesky factorization and for back-substitution.*

Moreover, the optimal $\lambda$ parameter for [AW11] has to be determined manually for each mesh.

Figure 8 shows another result of geodesics in heat. This example features highly anisotropic polygons that lead to severe distortions for [AW11], while our operator defines smooth geodesic distances.

Table 4 provides a quantitative evaluation of geodesic distances, by compare them to Euclidean distances on different planar meshes (Figure 3). Our operator yields smaller root-mean-square errors for most models, including the geodesic distances computed via intrinsic Delaunay triangulation [BS07,SSC19] (based on the implementation provided in libigl [JP*18]).

### 5.6. Timings

We evaluated the computational costs of the different operators on the HEX SPHERE (16070 faces) and the FINE SPHERE (96408 faces) shown in Figure 2. All timings were measured on a standard workstation with a six-core Intel Xeon 3.6 GHz CPU; no experiment exploited multi-threading. We analyze our approach with the different virtual vertex options described in Section 4.2: *centroid* of polygon vertices, minimizing the sum of (absolute) triangle areas (*Abs. Area*), minimizing the sum of squared triangle areas using either *affine* weights or *convex* weights. We compare these methods to Alexa and Wardetzky [AW11] and to the minimum area polygon triangulation [Lie03]. The timings are given in Table 5.

In terms of the construction time for the Laplace matrix, our approach is faster than [AW11] on fine-resolution meshes and comparable for coarser meshes. And, with the exception of *centroid*, it is also the fastest of the different virtual vertex choices, since the Newton optimization of *Abs. Area* (based on Eigen) and the QP solver of *convex* (based on CGAL) are computationally expensive. However, triangulating the mesh and constructing the cotangent Laplacian is faster than defining a polygon Laplacian. Also, since a vertex on the polygon mesh has at least as many face-adjacent neighbors as the same vertex on the triangulated mesh, the polygon Laplacians are less sparse, resulting in an increased solver time.

### 5.7. Comparison of virtual vertex choices

As stated in Section 4.2, there are several options for computing the virtual vertices and their weights. We compare the performance of these alternative constructions, using both lumped and un-lumped mass matrices, in a number of applications. For geodesic distances (see Section 5.5), our proposed version using affine weights gives the overall best results (see Table 6), although the strictly convex weights yield a smaller error for meshes with non-star-shaped faces. For most applications, using the lumped mass matrix gives better results, as also shown in the additional experiments in the supplementary material. Balancing numerical performance with efficiency, we found the minimizer of the squared triangle areas through affine weights to be the best choice.

### 5.8. Convergence behavior

We evaluate the convergence behavior of our Laplacian under mesh refinement by solving the Poisson equation $\Delta f = b$ with Dirichlet boundary conditions for the Franke test function [Fra79]

$$f(x,y) = \frac{3}{4}e^{-\frac{(9x-2)^2+(9y-2)^2}{4}} + \frac{3}{4}e^{-\frac{(9x+1)^2}{49}-\frac{9y+1}{10}} + \frac{1}{2}e^{-\frac{(9x-7)^2+(9y-3)^2}{4}} - \frac{1}{5}e^{-(9x-4)^2-(9y-7)^2}.$$

Figure 9 depicts the decrease of the $L_2$ error under mesh refinement for regular triangle, quad, and hex meshes. The identical slope in this log-log plot reveals that our operator inherits the quadratic convergence order of the triangle-based cotangent Laplacian.

**Summary** While our evaluations do not demonstrate that our Laplace operator is superior to existing state-of-the-art methods, it does show that it is competitive. We perform slightly better than triangulation [Lie03]. This is likely because our Laplacian is denser. And, while the polygon Laplacian of [AW11] outperforms ours in some cases, its behavior depends on the choice of the parameter $\lambda$, which cannot be fixed so as to perform well in all cases. In contrast, our method is parameter-free.

## 6. Conclusion

In this work we have proposed a novel polygon Laplacian, defined by first refining the polygon mesh to a triangle mesh and then coarsening the cotangent Laplacian from the triangle mesh back to the original polygon mesh. The derived polygon Laplacian exhibits numerous desirable properties including sparsity, symmetry, negative semi-definiteness, linear precision, and consistency with the divergence and gradient operators, without suffering from an increase in the dimensionality of the linear system. We have evaluated our Laplacian against other state-of-the-art methods and have demonstrated that it performs competitively, providing efficient and high-quality solutions without requiring parameter tuning.

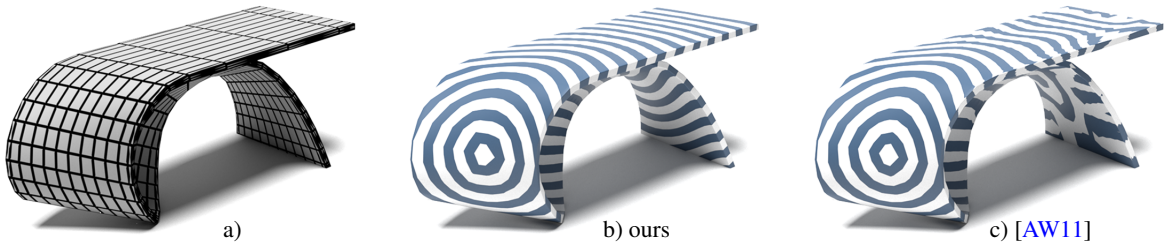



**Figure 8:** *Geodesic distances computed on a quad mesh (a) with our Laplacian (b) and the Laplacian proposed in [AW11] (c). Even with the manually determined best weight parameter ($\lambda = 0.4$), (c) exhibits distortions while ours remains stable despite highly anisotropic faces.*

| | *un-lumped mass matrix* | | | | *lumped mass matrix* | | | |
|---|---|---|---|---|---|---|---|---|
| Mesh | Affine | Convex | Centroid | Abs.Area | Affine | Convex | Centroid | Abs.Area |
| QUADS 1 | **0.025** | **0.025** | **0.025** | **0.025** | 0.026 | 0.027 | 0.027 | 0.027 |
| QUADS 2 | **0.031** | **0.031** | **0.031** | **0.031** | 0.036 | 0.036 | 0.036 | 0.036 |
| L-SHAPED | 0.141 | 0.165 | 0.134 | 0.134 | **0.057** | 0.063 | 0.068 | 0.068 |
| TETRIS 1 | 0.465 | 0.490 | 0.490 | 0.491 | 0.218 | **0.181** | 0.186 | 0.185 |
| TETRIS 2 | 0.406 | 0.346 | 0.151 | 0.153 | **0.082** | 0.089 | 0.089 | 0.088 |

**Table 6:** *RMSE of geodesic distances for the planar meshes in Figure 3 computed with different choices of virtual vertices and using un-lumped and lumped mass matrices.*

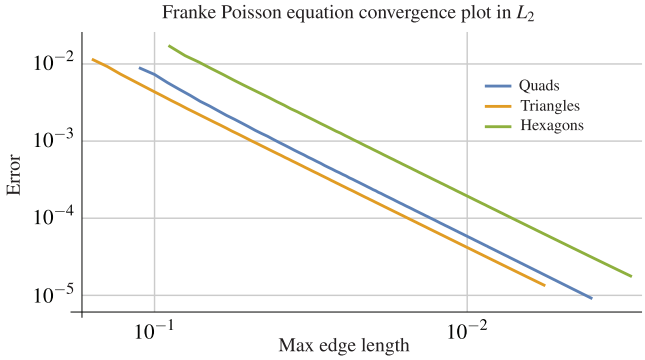



**Figure 9:** *$L_2$ convergence plots for the solution of the Poisson equation on triangle, quad, and hex meshes of increasing resolution.*

In the future, we would like to extend our approach in several ways. We would like to consider introducing multiple virtual vertices within a single face as this could allow for better triangulations of non-convex polygons. We would like to consider loosening the restriction that the virtual vertex needs to be an affine combination of the polygon vertices. For example, in the case of a planar polygonization of the sphere, this would allow us to introduce vertices on the surface of the sphere, not just on the planes containing the polygons. Finally, we would like to extend our approach to higher-order basis functions and volumetric polyhedral meshes.

**Acknowledgments** We thank the anonymous reviewers for their valuable comments and suggestions, and Marc Alexa and Fabian Prada for helpful discussions on Laplacians and discrete exterior calculus. This work was sponsored in part by NSF Award 1422325.

Open access funding enabled and organized by Projekt DEAL. [Correction added on 24 March 2021, after first online publication: Projekt Deal funding statement has been added.]

**Appendix A:** Finding the optimal virtual point

Given a polygon with $n$ vertices $(\vec{x}_1,\dots,\vec{x}_n)$, we want to insert a point $\vec{x} = \vec{x}(\vec{w})$, defined as an affine combination $\vec{x}(\vec{w}) = \sum_{j=1}^{n} w_j \vec{x}_j$ with $\sum_j w_j = 1$, such that the sum of squared triangle areas over the resulting triangle fan is minimized. This leads to the following optimization problem in the weights $\vec{w} = (w_1,\dots,w_n)^{\mathsf{T}}$:

$$\min_{\vec{w}} \sum_{i=1}^{n} \operatorname{area}\left(\vec{x}_i, \vec{x}_{i+1}, \sum_{j=1}^{n} w_j \vec{x}_j\right)^2 \quad \text{s.t.} \quad \sum_{j=1}^{n} w_j = 1. \tag{34}$$

The objective function can be rewritten as

$$E(\vec{w}) = \sum_{i=1}^{n} \frac{1}{2} \left\| (\vec{x}_i - \vec{x}_{i+1}) \times (\vec{x}(\vec{w}) - \vec{x}_i) \right\|^2. \tag{35}$$

Since $E$ is quadratic in $\vec{x}(\vec{w})$ and therefore also quadratic in $\vec{w}$, it can be written as

$$E(\vec{w}) = \frac{1}{2} \vec{w}^{\mathsf{T}} \mathbf{A} \vec{w} + \vec{b}^{\mathsf{T}} \vec{w} + c. \tag{36}$$

Minimizing with respect to $\vec{w}$, i.e., setting $\frac{\partial E}{\partial \vec{w}}$ to zero, leads to

$$\mathbf{A}\vec{w} = -\vec{b} \quad \text{with}$$

$$\begin{aligned} \mathbf{A}_{ij} &= 2 \sum_{k=1}^{n} \left(\mathbf{x}_j \times (\mathbf{x}_{k+1} - \mathbf{x}_k)\right) \cdot \left(\mathbf{x}_i \times (\mathbf{x}_{k+1} - \mathbf{x}_k)\right), \\ \vec{b}_i &= 2 \sum_{k=1}^{n} \left(\mathbf{x}_i \times (\mathbf{x}_{k+1} - \mathbf{x}_k)\right) \cdot \left((\mathbf{x}_{k+1} - \mathbf{x}_k) \times \mathbf{x}_k\right) \end{aligned} \tag{37}$$

We add one row to the matrix to enforce the partition of unity constraint $\sum_j w_j = 1$, turning it into the $(n+1) \times n$ linear system

$$\begin{pmatrix} \mathbf{A} \\ 1 \cdots 1 \end{pmatrix} \vec{w} = \begin{pmatrix} -\mathbf{b} \\ 1 \end{pmatrix}. \tag{38}$$

The matrix $\mathbf{A}$ has rank 2 or 3 for planar or non-planar polygons, respectively. Hence, the system in Equation (38) has rank 3 or 4 for planar/non-planar polygons. It is therefore fully determined for either (planar) triangles or non-planar quads, and is underdetermined otherwise. In the latter case, we aim for the least-norm solution, i.e., the solution $\vec{w}$ with minimal $\|\vec{w}\|$, because it distributes the influence of polygon vertices $\vec{x}_i$ equally. We handle both the fully-determined and the under-determined cases in a robust and unified manner through the matrix pseudo-inverse [GL89], which we compute through Eigen's complete orthogonal decomposition [GJ*10].

## References


[Abb84] ABBOT E. A.: *Flatland: A Romance of Many Dimensions*. Seeley & Co., 1884. 1

[AFW06] ARNOLD D. N., FALK R. S., WINTHER R.: Finite element exterior calculus, homological techniques, and applications. *Acta Numerica 15* (2006), 1–155. 5

[AW11] ALEXA M., WARDETZKY M.: Discrete Laplacians on general polygonal meshes. *ACM Transactions on Graphics 30*, 4 (2011), 102:1–102:10. 1, 2, 4, 5, 6, 7, 8, 9, 10

[BKP*10] BOTSCH M., KOBBELT L., PAULY M., ALLIEZ P., LEVY B.: *Polygon Mesh Processing*. AK Peters, 2010. 2

[BS07] BOBENKO A. I., SPRINGBORN B. A.: A discrete Laplace–Beltrami operator for simplicial surfaces. *Discrete & Computational Geometry 38*, 4 (2007), 740–756. 2, 8, 9

[BSW08] BELKIN M., SUN J., WANG Y.: Discrete Laplace operator on meshed surfaces. In *Proceedings of Symposium on Computational Geometry* (2008), pp. 278–287. 2

[CLB*09] CHUANG M., LUO L., BROWN B. J., RUSINKIEWICZ S., KAZHDAN M.: Estimating the Laplace-Beltrami operator by restricting 3d functions. *Computer Graphics Forum 28*, 5 (2009), 1475–1484. 2

[CRK16] CHUANG M., RUSINKIEWICZ S., KAZHDAN M.: Gradient-domain processing of meshes. *Journal of Computer Graphics Techniques 5*, 4 (2016), 44–55. 2

[CWW13] CRANE K., WEISCHEDEL C., WARDETZKY M.: Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics 32*, 5 (2013), 152:1–152:11. 2, 8

[dGDMD16] DE GOES F., DESBRUN M., MEYER M., DEROSE T.: Subdivision exterior calculus for geometry processing. *ACM Transactions on Graphics 35*, 4 (2016), 133:1–133:11. 2

[DMA02] DESBRUN M., MEYER M., ALLIEZ P.: Intrinsic parameterizations of surface meshes. *Computer Graphics Forum 21*, 3 (2002), 209–218. 2

[DMSB99] DESBRUN M., MEYER M., SCHRÖDER P., BARR A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of ACM SIGGRAPH* (1999), pp. 317–324. 1, 6

[Dzi88] DZIUK G.: *Finite Elements for the Beltrami operator on arbitrary surfaces*. Springer Berlin Heidelberg, 1988, pp. 142–155. 1

[Fle84] FLETCHER C.: *Computational Galerkin Methods*. Computational Physics Series. Springer-Verlag, 1984. 2

[Flo97] FLOATER M. S.: Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design 14*, 3 (1997), 231–250. 2

[Fra79] FRANKE R.: *A critical comparison of some methods for interpolation of scattered data*. Tech. rep., Naval Postgraduate School, 1979. 9

[GGT06] GORTLER S. J., GOTSMAN C., THURSTON D.: Discrete one-forms on meshes and applications to 3d mesh parameterization. *Comput. Aided Geom. Des. 23*, 2 (2006), 83–112. 2

[GJ*10] GUENNEBAUD G., JACOB B., ET AL.: Eigen v3. http://eigen.tuxfamily.org, 2010. 11

[GL89] GOLUB G. H., LOAN C. F. V.: *Matrix Computations*. Johns Hopkins University Press, Baltimore, 1989. 11

[JP*18] JACOBSON A., PANOZZO D., ET AL.: libigl: A simple C++ geometry processing library, 2018. https://libigl.github.io/. 9

[KSBC12] KAZHDAN M., SOLOMON J., BEN-CHEN M.: Can mean-curvature flow be modified to be non-singular? *Computer Graphics Forum 31*, 5 (2012), 1745–1754. 2, 6, 7

[Lie03] LIEPA P.: Filling holes in meshes. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (2003), pp. 200–205. 6, 7, 9

[LZ10] LÉVY B., ZHANG H. R.: Spectral mesh processing. In *ACM SIGGRAPH 2010 Courses* (2010), pp. 8:1–8:312. 2

[Mac49] MACNEAL R.: *The Solution of Partial Differential Equations by Means of Electrical Networks.* PhD thesis, California Institute of Technology, 1949. 1

[MTAD08] MULLEN P., TONG Y., ALLIEZ P., DESBRUN M.: Spectral conformal parameterization. *Computer Graphics Forum 27*, 5 (2008), 1487–1494. 7

[PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Experim. Math. 2* (1993), 15–36. 1

[SB19] SIEGER D., BOTSCH M.: The polygon mesh processing library, 2019. http://www.pmp-library.org. 5

[SCOL*04] SORKINE O., COHEN-OR D., LIPMAN Y., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *Proceedings of Eurographics Symposium on Geometry Processing* (2004), pp. 179–188. 2

[SSC19] SHARP N., SOLIMAN Y., CRANE K.: Navigating intrinsic triangulations. *ACM Transactions on Graphics 38*, 4 (2019). 2, 9

[Tau95] TAUBIN G.: A signal processing approach to fair surface design. In *Proceedings of ACM SIGGRAPH* (1995), pp. 351–358. 2

[VL08] VALLET B., LÉVY B.: Spectral geometry processing with manifold harmonics. *Computer Graphics Forum 27*, 2 (2008), 251–260. 7

[Whi57] WHITNEY H.: *Geometric Integration Theory*. Princeton University Press, 1957. 5

[WMKG07] WARDETZKY M., MATHUR S., KÄLBERER F., GRINSPUN E.: Discrete Laplace operators: No free lunch. In *Proceedings of Eurographics Symposium on Geometry Processing* (2007), pp. 33–37. 2, 4

[XLH11] XIONG Y., LI G., HAN G.: Mean Laplace-Beltrami operator for quadrilateral meshes. In *Transactions on Edutainment V*, Pan Z., Cheok A. D., Müller W., (Eds.). Springer Berlin Heidelberg, 2011, pp. 189–201. 2

[Zha04] ZHANG H.: Discrete combinatorial Laplacian operators for digital geometry processing. In *Proceedings of SIAM Conference on Geometric Design and Computing* (2004), pp. 575–592. 2