Eurographics Symposium on Geometry Processing 2020
Q. Huang and A. Jacobson
(Guest Editors)

Volume 39 (2020), Number 5

A Laplacian for Nonmanifold Triangle Meshes

Nicholas Sharp and Keenan Crane

Carnegie Mellon University

Abstract

We describe a discrete Laplacian suitable for any triangle mesh, including those that are nonmanifold or nonorientable (with or without boundary). Our Laplacian is a robust drop-in replacement for the usual cotan matrix, and is guaranteed to have nonnegative edge weights on both interior and boundary edges, even for extremely poor-quality meshes. The key idea is to build what we call a "tufted cover" over the input domain, which has nonmanifold vertices but manifold edges. Since all edges are manifold, we can flip to an intrinsic Delaunay triangulation; our Laplacian is then the cotan Laplacian of this new triangulation. This construction also provides a high-quality point cloud Laplacian, via a nonmanifold triangulation of the point set. We validate our Laplacian on a variety of challenging examples (including all models from Thingi10k), and a variety of standard tasks including geodesic distance computation, surface deformation, parameterization, and computing minimal surfaces.

CCS Concepts

• ***Mathematics of computing*** → ***Discretization; Partial differential equations;***

1. Introduction

The Laplacian Δ measures the degree to which a given function u deviates from its mean value in each local neighborhood; it hence characterizes a wide variety of phenomena such as the diffusion of heat, the propagation of waves, and the smoothest interpolation of given boundary data. Such phenomena play a central role in algorithms from geometry processing and geometric learning [BKP*10, BBL*17]. However, it remains challenging to construct *discrete* Laplacians that are accurate, efficient, and reliable—especially since contemporary data often fails to satisfy the preconditions of classical geometric algorithms (*e.g.*, being well-sampled, manifold, or exhibiting good triangle shape) [ZJ16, QSMG17].

Nonmanifold Laplacian. Though the Laplacian is often formally defined in local coordinates on a manifold, there is no fundamental reason why the manifold assumption is necessary. Physically, for instance, one can weld together metal plates in a nonmanifold fashion—here, one can still view the Laplacian as the time derivative of a heat diffusion process: $\Delta u := \frac{d}{dt}|_{t=0} u(t)$ (reversing the usual heat equation). Likewise, one can take any nonmanifold domain and "thicken" it by a tiny radius $\varepsilon > 0$, so that it becomes a manifold of one dimension higher (Figure 1); here again the Laplacian is well-defined. Alternatively, one can still think of the Laplacian as giving the deviation from the average in a small metric ball, or relate it to the variation of total area [PP93]—both of which are meaningful for nonmanifold domains. And so on. For us, it will be natural to view the Laplacian as the Hessian of the *Dirichlet energy*—different triangulations of the input then just provide different piecewise linear basis functions for approximating the underlying continuous energy.

Discrete Laplacians. For triangle meshes, the de facto standard is the *cotan Laplacian* (Section 3.3), equivalent to the usual linear finite element stiffness matrix. This operator is very sparse, easy to build, and generally works well for unstructured meshes with irregular vertex distributions. It can also be used on nonmanifold meshes by just summing up per-triangle contributions (as famously done by Pinkall & Polthier for minimal surfaces [PP93]). However, cotan-Laplace has well-known problems, chiefly that it does not provide a *maximum principle* [WMKG07], which can lead to severe defects in common algorithms. An important development was the *intrinsic Laplacian* of Bobenko & Springborn [BS07], obtained by flipping

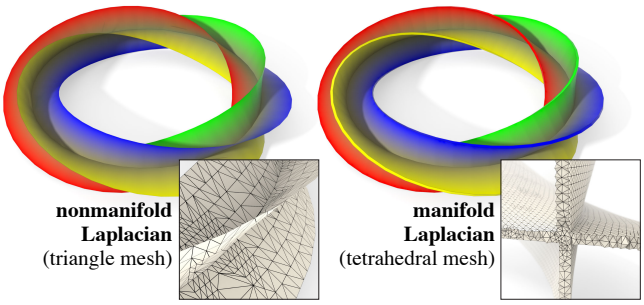


Figure 1: *We define a Laplacian for nonmanifold triangle meshes, which generally behaves like the Laplacian on a slightly thickened domain. Here for instance we get a near-identical harmonic interpolation of boundary values using our Laplacian on a triangle mesh* (left) *or with the standard Laplacian on a tetrahedral mesh* (right).

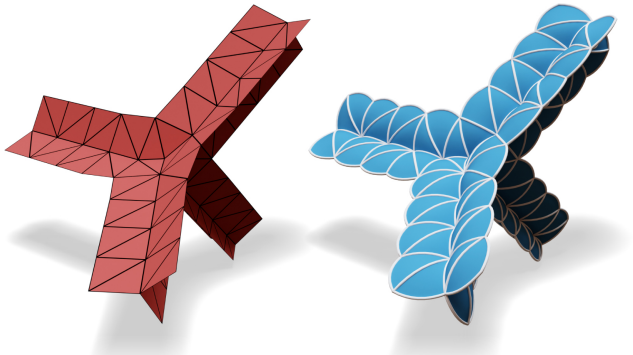


Figure 2: *A nonmanifold mesh (*left*) and its tufted cover (*right*). Since the cover is edge-manifold, we can freely flip edges in order to improve the Laplacian. (Note that we draw curved triangles only to help visualize connectivity; actual triangles remain flat.)*

edges to an *intrinsic Delaunay triangulation (iDT)* before building cotan-Laplace (Section 3.2). This operator retains most properties of cotan-Laplace, but guarantees a maximum principle, and improves the accuracy and reliability of many geometric algorithms [FSSB07, LFXH17, SSC19b, SSC19a]. However, the original construction applies only to manifold meshes, limiting its practical use.

Approach. Our key observation is that the intrinsic Laplacian can be constructed as long as all *edges* are manifold, since edge flips then remain well-defined. We therefore build a "tufted" version of the input, where all edges are manifold—and all interior vertices are nonmanifold (Figure 2). Though it may seem strange to intentionally introduce nonmanifold vertices, they ultimately have no effect on the definition of our Laplacian: we simply flip to Delaunay, then build cotan-Laplace as usual (Section 4).

Contributions. Overall, our contributions are as follows:

- We extend the notion of intrinsic Delaunay triangulations to nonmanifold geometry.
- We define a discrete Laplacian that exhibits all the same properties as the intrinsic Laplacian but (i) can be used with triangle meshes of arbitrary connectivity and (ii) has nonnegative weights even at boundary edges.
- Finally, we define a new point cloud Laplacian which inherits key properties of the intrinsic Laplacian.

In particular, both our mesh and point cloud Laplacian satisfy all the criteria for a "perfect" discrete Laplacian outlined by Wardetzky *et al.* [WMKG07], except for locality (though see Section 3.3). For inputs where no Delaunay flips are needed, our Laplacian is identical to summing up the per-triangle cotan Laplacian—though in practice such meshes are quite rare.

Since our Laplacian preserves the given vertex set, it can be used in a "black box" fashion: one simply provides an ordinary triangle mesh with n vertices as input, and gets a standard $n \times n$ Laplace matrix as output (Algorithm 3). From here, existing code can often be used without modification—see Section 5.4 for an example. The computational overhead is on par with just building cotan-Laplace and solving a linear system; we use simple arrays to represent the tufted cover (Section 4.1) and do not require more general nonmanifold data structures [DFGH04].

2. Related Work

Nonmanifold Geometry Processing. A variety of work from geometry processing considers nonmanifold triangle meshes [SG95, HG00, WLG03, GBTS99, YZ01], but does not explore the Laplacian; more recent work handles arbitrary, nonmanifold geometry by converting it to a volumetric, tetrahedral representation [JKSH13, HZG*18, SCM*19]. By working directly with *triangle* meshes we keep the problem dimension low, avoid hard tetrahedral meshing problems, and preserve self-intersections that are geometric but not topological. Most importantly, we can improve the robustness of existing triangle-based algorithms by just replacing cotan-Laplace with our Laplacian (see Section 5).

Covering Spaces. In geometry processing, *branched covers* have been used for, *e.g.*, mesh generation [KNP07, NPPZ11, NRP11], pattern synthesis [KCPS15], and topological visualization [RKG*17]. The branched cover we consider is just a trivial double cover, glued back together at all vertices (and any boundary curves)—which ensures that all edges are manifold. Though this construction is quite natural, it does not appear to have been previously used in geometry processing—perhaps because it is not always representable as an ordinary simplicial complex (Section 4.1).

Laplacians for Triangle Meshes. A fairly active question [AW11, HKA15, BSW08, HP11, MMdGD11] is how to accurately approximate the smooth *Laplace-Beltrami* operator while retaining key properties such as symmetry, locality, *etc.* [WMKG07]. For triangle meshes, a common choice is the *cotan Laplacian*, which arises naturally via linear finite elements, discrete exterior calculus [CDGDS13], electrical networks [Mac49, Duf59], and minimal surfaces [PP93]. As noted in Section 1, cotan-Laplace can be used on nonmanifold meshes by accumulating per-triangle matrices, but does not guarantee a maximum principle, and exhibits poor behavior on meshes with large obtuse angles (see [She02] and Section 5).

The *intrinsic Laplacian* overcomes these difficulties by replacing the input mesh with an *intrinsic Delaunay triangulation (iDT)* (Section 3.2). The iDT changes only the connectivity, using "bent" edges that run across the exact input geometry (Figure 3). On this

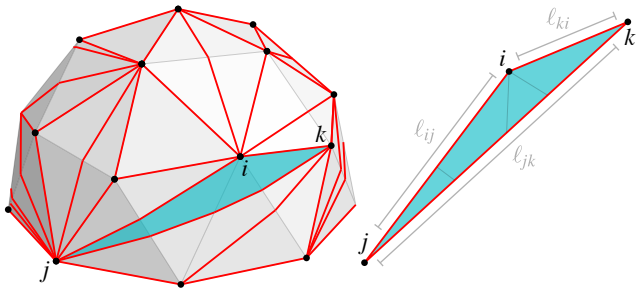


Figure 3: Left: *Much as a planar point set can be triangulated in many different ways, an intrinsic triangulation allows the vertices of a polyhedron to be connected by many different straight (geodesic) edges along the surface.* Right: *Each intrinsic triangle can be flattened into the plane without distortion; its geometry (area, angles,* etc.*) is hence completely described by just three edge lengths.*

new triangulation, cotan-Laplace is guaranteed to have nonnegative edge weights—at least for interior edges (see Section 5.5). To date, however, the intrinsic Laplacian, and more broadly, the notion of intrinsic triangulations, has been limited to manifold surface meshes. Our goal is to develop a Laplacian that is more reliable than cotan-Laplace, yet can be used on meshes of arbitrary connectivity.

Liu *et al.* [LXFH15] achieve the intrinsic Delaunay property via extrinsic *splits* rather than intrinsic flips; though their algorithm (and code) assumes manifold input, it may be possible to extend this scheme to nonmanifold meshes. However, as seen in [LFXH17, Figure 20] and [SSC19a, Figure 25], splits generate numerous additional triangles, and skinny angles which can worsen conditioning of the Laplacian [She02]. Moreover, since splits add new vertices, the resulting Laplacian cannot be used directly on the input mesh.

Global Remeshing. A traditional way to improve finite element quality is to globally remesh the domain. However, standard *extrinsic* Delaunay criteria such as *restricted Delaunay* [CDS12, Chapter 13] and *optimal Delaunay* [CH11] do not guarantee that the Laplacian will have nonnegative edge weights, even if triangles are otherwise nice (Figure 4). Global remeshing is also far more expensive than building our Laplacian—which takes just milliseconds (Section 5).

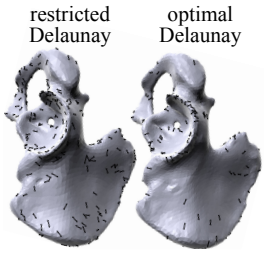


Figure 4: *Standard extrinsic Delaunay conditions yield many edges with negative weights (in black).*

Higher-Order Finite Elements. The Laplacian can also be improved by adopting higher-order basis functions [GKS02, RWP06, SHD*18], which preserves the input geometry but adds more degrees of freedom. A practical downside is that this larger Laplace matrix cannot be used as a drop-in replacement for cotan-Laplace; it also consumes additional time and memory. Here, little has been said about the nonmanifold case, nor about the maximum principle.

Point-Based Laplace Operators. To handle nonmanifold geometry one could also discard connectivity completely and build a *point cloud Laplacian* directly on the input vertex set [BSW08, BSW09, LPG12]. Traditional point cloud Laplacians (*à la* [BSW09]) exhibit some nice features like pointwise convergence (under certain sampling conditions) and a maximum principle, but lose other key properties (like symmetry, or linear precision). Moreover, small/thin features may get joined together erroneously, and quality will suffer unless the input vertices densely and uniformly sample the domain. There is also a large computational cost: the average neighborhood size is typically far greater than for a mesh-based Laplacian (around 30 points, rather than 6 vertices)—and must *grow* as sampling density increases [HP11], resulting in rather dense systems that are expensive to solve. In practice, picking parameters that balance quality and sparsity often requires extensive hand-tuning. For all these reasons, building point cloud Laplacians *on top of* classic schemes for triangle meshes (like cotan-Laplace) presents an attractive alternative [CRT04, CTO*10], providing sparsity and accuracy even for irregularly sampled points. Our nonmanifold construction takes this approach a step further by also providing nonnegative edge weights—see Section 5.7 for further discussion.

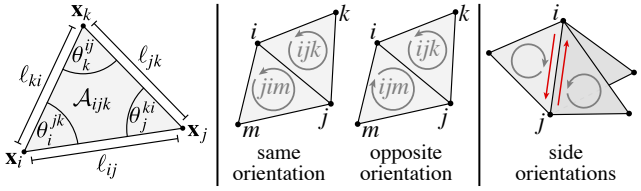


Figure 5: *Notation and orientation conventions for triangle meshes.*

3. Background

3.1. Triangle Meshes

We assume the input is a triangle mesh M, and use V, E, and F to denote its vertices, edges, and faces, *resp.* (Polygons with more sides can simply be triangulated, though the choice of triangulation may slightly change the result [AW11].) We denote vertices by a single index $i \in \mathsf{V}$, edges by two indices ij, and faces by three indices ijk. The *orientation* of a mesh element is given by the cyclic ordering of its vertices—for instance, $ijk = jki$, but $ijk \neq jik$. Two faces sharing an edge have the same orientation if shared vertices appear in the *opposite* order—*e.g.*, ijk and jim have the same orientation; ijk and ijm have opposite orientation. Similarly, each oriented triangle ijk has three oriented *sides* with compatible orientations ij, jk, ki. Finally, we use $\mathbf{x} : \mathsf{V} \to \mathbb{R}^3$ to denote the input vertex coordinates, $\ell_{ij} := |\mathbf{x}_i - \mathbf{x}_j|$ for the length of edge ij, $\mathcal{A}_{ijk}$ for the area of triangle ijk, and θ_i^{jk} for the interior angle at vertex i of triangle ijk.

Manifold Condition. An interior (or boundary) edge ij is *manifold* if it is contained in exactly two (or one) triangles; an interior (or boundary) vertex i is manifold if the boundary of all triangles incident on i forms a single loop (or path) of edges. For example, an edge contained in three faces is not manifold, and a vertex contained in two "cones" is also not manifold (see Figure 6). An *edge-manifold* or *vertex-manifold* mesh has all manifold edges or vertices, *resp.*

3.2. Intrinsic Triangulations

Intuitively, an *intrinsic triangulation* is another triangulation "drawn on top of" a given mesh, with the same vertex set; edges of the intrinsic triangulation are straight (*i.e.*, geodesic) paths (Figure 3, *left*). In reality, the intrinsic triangulation is completely determined by its connectivity and edge lengths [SSC19a, Section 2]; exactly where and how the intrinsic edges cross the input edges is not relevant for our Laplacian (though this additional correspondence

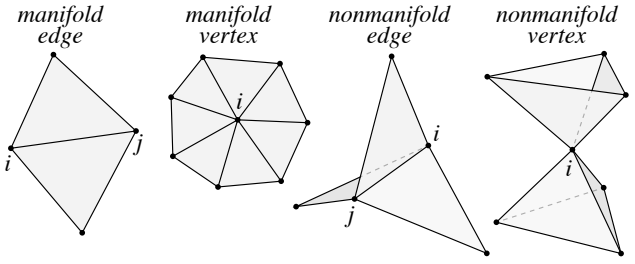


Figure 6: *Examples of manifold and nonmanifold edges/vertices.*

information can be useful for other applications [FSSB07, SSC19a]). In particular, since the two triangulations have the same vertex set, an intrinsic triangle ijk cannot have any vertices on its interior. Its geometry is therefore completely determined by three ordinary edge lengths (Figure 3, *right*), which in turn determine its area $\mathcal{A}_{ijk}$ and interior angles θ_i^{jk}.

Intrinsic Delaunay. A planar triangulation is *Delaunay* if no triangle circumcircle has vertices in its interior; such triangulations play an important role throughout computational science [AKL13]. Rivin [Riv94] generalizes this notion to manifold triangulations—in particular, a triangulation is *intrinsic Delaunay* if every interior edge ij shared by triangles ijk, jim satisfies the *local Delaunay condition*

$$\theta_k^{ij} + \theta_m^{ji} \leq \pi. \tag{1}$$

For planar triangulations this definition agrees with the usual one, but can also be applied to triangulated surfaces since the angles θ can be determined directly from the (intrinsic) edge lengths ℓ. Importantly, an interior edge satisfies Equation 1 if and only if

$$\cot\theta_k^{ij} + \cot\theta_m^{ji} \geq 0, \tag{2}$$

in other words, if and only if its so-called *cotan weight* is nonnegative [BS07]. Hence, an intrinsic Delaunay triangulation always yields a Laplacian with positive weights on interior edges.

Intrinsic Edge Flips. An *intrinsic edge flip* updates a triangle pair as depicted in Figure 7, computing the new edge length from a planar layout of the original triangles—and *not* from the distance between endpoints in $\mathbb{R}^3$. An intrinsic Delaunay triangulation can always be obtained by greedily flipping any non-Delaunay edge (Algorithm 2). This algorithm terminates in a finite number of operations [BS07], and typically requires no more than about $|E|$ total flips [SSC19a, Figure 10]. Importantly, intrinsic flips exactly preserve the original geometry—from a finite element perspective, changing the triangulation effectively just provides a different set of linear basis functions for the same polyhedral domain, which in turn improves the quality of the Laplace operator. The only challenge is that edge flips—and hence the iDT—are well-defined only for *edge-manifold* meshes, which is why we must first build the tufted cover (Section 4.2).

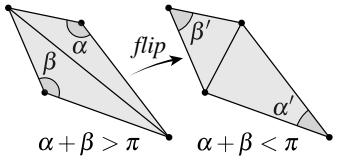


Figure 7: *A non-Delaunay edge* (left) *can always be made Delaunay (right) via an edge flip.*

3.3. Cotan Laplacian

A general way to build cotan-Laplace (which works for our tufted cover) is to define, for each triangle $ijk \in \mathsf{F}$, a local cotan matrix

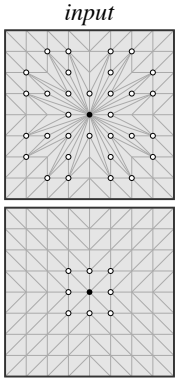
$$\mathsf{L}_{ijk} := \begin{bmatrix} w_{ki}+w_{ij} & -w_{ij} & -w_{ki} \\ -w_{ij} & w_{ij}+w_{jk} & -w_{jk} \\ -w_{ki} & -w_{jk} & w_{jk}+w_{ki} \end{bmatrix},$$

where $w_{ij} := \frac{1}{2}\cot\theta_i^{jk}$ (and similarly for w_{jk}, w_{ki}). Entries of local matrices are then summed up into the corresponding entries of a global Laplacian $\mathsf{L} \in \mathbb{R}^{|V|\times|V|}$. The associated mass matrix $\mathsf{B} \in \mathbb{R}^{|V|\times|V|}$ is a diagonal matrix with entries

$$\mathsf{B}_{ii} = \sum_{ijk\in F} \mathcal{A}_{ijk}/3,$$

i.e., for each triangle containing i, we contribute one-third of its area to the mass at vertex i. Using these matrices, a Poisson equation $\Delta u = f$ can be discretized as $\mathsf{Lu} = \mathsf{Bu}$.

Perfect Laplacians. Wardetzky *et al.* [WMKG07] outline several criteria for a "perfect" discrete Laplacian, which capture fundamental properties of the smooth Laplace operator Δ. Cotan-Laplace exhibits all of these properties, except for one: nonnegative edge weights, which are sufficient to guarantee a discrete *maximum principle*, *i.e.*, that any discrete harmonic function $\mathsf{Lu} = 0$ has local maxima/minima only at boundary vertices. This property is important in practice, preventing (for example) interpolated values from going out of bounds, or flipped triangles in a parameterization. For a manifold mesh, one can always obtain nonnegative weights by flipping to an intrinsic Delaunay triangulation (Equation 2), but edge flips have no clear definition for nonmanifold edges. Moreover, even an iDT can yield negative weights on boundary edges, causing problems for interpolation (Section 5.5). These shortcomings motivate our search for a Laplacian that is well-behaved on *any* triangle mesh.



A Note About Locality. According to Wardetzky *et al.* [WMKG07], a perfect discrete Laplacian should also exhibit a certain notion of *locality*. However, this criterion is worth revisiting. In the smooth setting, locality of the Laplacian Δ means that the value of Δu at any point p depends only on values of u in an arbitrarily small metric neighborhood of p. The intrinsic Laplacian generally does a good job of capturing this notion of **geometric locality**—for instance, every vertex in a Delaunay triangulation is guaranteed to be connected to its geometrically closest neighbors [MD10, Theorem T2]. Wardetzky's only complaint is that the intrinsic Laplacian can depend on arbitrarily large *combinatorial* neighborhoods, relative to a fixed input triangulation. More precisely, Wardetzky's notion of **combinatorial locality** asks that the value of Lu at a vertex i depend only on values of u at immediate neighbors j in the input triangulation (or more generally, on values k edges away, for some universally fixed integer k). However, combinatorial locality is often a poor proxy for geometric locality—consider for instance the inset triangulation (top) versus its Delaunay triangulation (bottom). Moreover, from a geometric point of view, there is nothing special about the neighborhood relationships in the input mesh: just as many different atlases describe the same manifold, many different triangulations can be used to specify the metric of the same polyhedron. On the whole, there is no clear reason to favor combinatorial locality over geometric locality if the goal is to approximate smooth solutions as accurately as possible. On the other hand, changing the mesh combinatorics can incur *computational* cost—for instance, it is not clear how to update the intrinsic Laplacian after moving even a single vertex, other than to rebuild the intrinsic Delaunay triangulation from scratch. See Section 6 for discussion of other practical trade-offs.

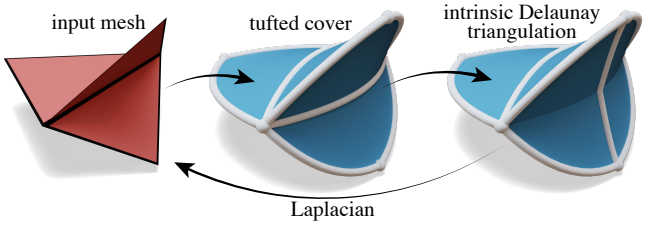


Figure 8: *Starting with the input mesh, we build a tufted cover, then flip to intrinsic Delaunay. Since the vertex set is preserved, a Laplacian built on the iDT can be used directly on the input mesh.*

4. Method

We now give the main steps needed to construct our Laplacian, outlined in Figure 8. For simplicity we will consider an input mesh given by a list of triangles with distinct vertices (as in common mesh file formats), but in principle our construction applies to any triangulation (formally: any pure 2-dimensional Δ-complex).

4.1. Data Structure

In general, a vertex-face adjacency list is not be sufficient to represent an iDT, which can contain, *e.g.*, multiple edges between the same two vertices (see [SSC19a, Figure 6]). Formally, we must be able to encode a so-called Δ-*complex* [Hat02, Section 2.1]. We therefore augment this list with a *gluing map* G that can represent any edge-manifold intrinsic triangulation (Figure 9). For each side of each triangle, G stores the corresponding side of the adjacent triangle. A side is encoded as a pair (f,s), where $f \in \{1,\dots,|F|\}$ is the index of a face, and $s \in \{1,2,3\}$ is the index of a side within that face. *E.g.*, for $f = ijk$, $s = 1,2,3$ correspond to ij, jk, and ki, *resp.* (One could also store a flag for boundary edges, though our tufted cover will never have boundary.)

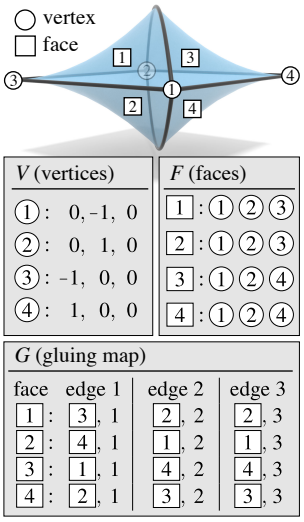


Figure 9: *A simple array-based data structure records, for each side of each face, which face and side it is glued to. A side is encoded by a face index, and an index $s \in \{1,2,3\}$.*

4.2. Tufted Cover

One idea for transforming nonmanifold input to be manifold is to take the boundary of a thickened "shell" around the input (as in Figure 1, *right*), but explicitly generating such a shell while avoiding self-intersection is cumbersome and expensive. Since a triangulation need only be edge-manifold to support the iDT (Section 4.3) on which we build the Laplacian (Section 4.4), we use a much simpler construction. In particular, we make two logical copies of each input triangle, and systematically glue along edges to form a closed, edge-manifold mesh. We call this mesh the *tufted cover* since duplicated triangles still share the same vertices, akin to two layers of upholstered furniture connected by buttons (see inset). Note that, purely for visualization (*e.g.*, in Figure 8), we "inflate" the cover outward to clearly distinguish front and back faces, but the actual geometry of each triangle remains flat.

More precisely, the tufted cover of an input mesh $\mathsf{M} = (\mathsf{V},\mathsf{E},\mathsf{F})$ is a triangle mesh $\widetilde{\mathsf{M}} = (\widetilde{\mathsf{V}},\widetilde{\mathsf{E}},\widetilde{\mathsf{F}})$ with the same vertices ($\widetilde{\mathsf{V}} = \mathsf{V}$), together with a gluing map $\widetilde{\mathsf{G}}$. For each face $f \in \mathsf{F}$, $\widetilde{\mathsf{M}}$ has two oppositely oriented copies $\sigma_F(f), \sigma_B(f) \in \widetilde{\mathsf{F}}$ which one can think of as the "front" and the "back" of f, respectively. Nonmanifold edges are resolved by the way we define the gluing map $\widetilde{\mathsf{G}}$. We first list the faces around each edge $e \in \mathsf{E}$ in a circular order $\rho^e := (f_1,\dots,f_k)$. If we imagine that these faces are consistently orientated relative to e, then we just glue them "front to back" along the shared edge, *i.e.*, we glue $\sigma_F(f_i)$ to $\sigma_B(f_{i+1 \bmod k})$ for $i = 1,\dots,k$ (Figure 10 gives an example). A more precise description of the gluing procedure which takes orientation into account is given in Algorithm 1; here $\textsc{Side}(e,f)$ just gives the side index of e within face f (1, 2, or 3).

Algorithm 1 $\textsc{ConstructTuftedCover}(\mathsf{M},\rho)$

Input: A (possibly nonmanifold) triangle mesh M and an ordering ρ^e of faces around each edge e.

Output: The tufted cover mesh $\widetilde{\mathsf{M}}$ and edge glue map $\widetilde{\mathsf{G}}$

$\widetilde{\mathsf{F}} \leftarrow \bigcup_{ijk\in\mathsf{F}} \{ijk, jik\}$ ▷*two copies of each face*
$\widetilde{\mathsf{G}} \leftarrow \{\}$ ▷*assemble an edge glue map*
for each edge $e \in \mathsf{E}$ **do**
  if e and $\sigma_F(\rho^e_1)$ have the same orientation **then**
    $f \leftarrow \sigma_F(\rho^e_1)$
  else $f \leftarrow \sigma_B(\rho^e_1)$
  for $i = 1,\dots,k$ **do** ▷*letting $k := |\rho_e|$*
    $g_1 \leftarrow \sigma_F(\rho^e_{i+1 \bmod k})$
    $g_2 \leftarrow \sigma_B(\rho^e_{i+1 \bmod k})$
    if f and g_1 have different orientation along e **then**
      $\textsc{Swap}(g_1,g_2)$
    $\widetilde{\mathsf{G}}(f,\textsc{Side}(e,f)) \leftarrow (g_1,\textsc{Side}(e,g_1))$
    $\widetilde{\mathsf{G}}(g_1,\textsc{Side}(e,g_1)) \leftarrow (f,\textsc{Side}(e,f))$
    $f \leftarrow g_2$
return $\widetilde{\mathsf{M}},\widetilde{\mathsf{G}}$

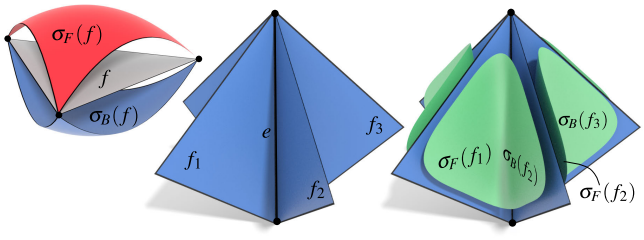


Figure 10: Left: *for each face f of the input mesh, we make a "front" and "back" copy. For each edge e* (center) *we then glue together alternating front/back pairs* (right).

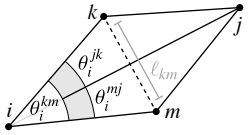
By construction, $\widetilde{\mathsf{G}}$ defines a symmetric bijection between faces of the same orientation; hence, the tufted cover $\widetilde{\mathsf{M}}$ is always edge-manifold and oriented. For input meshes with boundary, front and back faces are simply glued together along the boundary; $\widetilde{\mathsf{M}}$ is hence also closed. Note that we assume a unique ordering of faces around e, which may not be available in degenerate cases (*e.g.*, coplanar faces incident on e). As seen in Section 5.4, however, the ordering has little effect on the final operator—essentially we just need any reasonable space of linear elements for the region around the edge.

4.3. Intrinsic Delaunay Triangulation

At this point, summing up local cotan matrices on $\widetilde{\mathsf{M}}$ (as in Section 3.3) would just yield the ordinary cotan-Laplacian (times two). To improve the Laplacian, we must now flip $\widetilde{\mathsf{M}}$ to an iDT, as shown in Figure 11, *right*. Algorithm 2 gives the usual greedy flip algorithm in terms of the gluing map G; to implement the "while" loop we keep a queue of edges that are not yet Delaunay (possibly enqueuing some of the neighbors ki, im, mj, and/or jk after each flip). In practice, this algorithm takes a small fraction of a second, even on large meshes [SSC19a, Section 4]. Each flip updates G as detailed in Figure 12, and the new edge length is given by

$$\ell_{km} = \sqrt{\ell_{im}^2 + \ell_{ik}^2 - 2\ell_{im}\ell_{ik}\cos\theta_i^{km}}. \tag{3}$$

To get θ_i^{km} we sum of the two angles θ_i^{jk} and θ_i^{mj}, which can be computed from known edge lengths via the law of cosines. (Fisher *et al.* [FSSB07, Section 2.2] give a more robust floating point implementation.)



Algorithm 2 FLIPTODELAUNAY$(\mathsf{M}, \mathsf{G}, \ell)$

Input: An edge-manifold triangulation M, with edge gluing map G and edge lengths ℓ.
Output: An intrinsic Delaunay triangulation $\mathsf{M}, \mathsf{G}, \ell$

```
while any edge ij is not Delaunay do        ▷Equation 1
    Compute the new length ℓ_km             ▷Equation 3
    Flip edge ij to km, updating F and G    ▷see Figure 12
end while
return M, G, ℓ
```

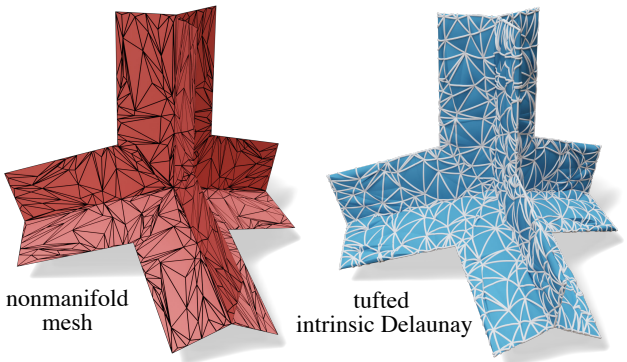


Figure 11: *Even for poor-quality nonmanifold meshes, we achieve the Delaunay criterion everywhere—without inserting new vertices.*

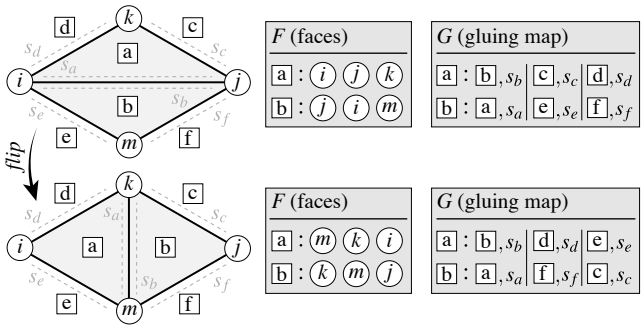


Figure 12: *An edge flip updates just two rows of the face and gluing tables. Here, values $s_f \in \{1,2,3\}$ index the sides of face f.*

4.4. Building the Laplacian

From here, our Laplacian is just the cotan-Laplacian L of $\widetilde{\mathsf{M}}$, scaled by a factor $1/2$ (since from a finite element perspective, we have effectively integrated the Laplacian *twice*, over the double cover). Algorithm 3 summarizes the entire process, and gives expressions for computing the cotan weights directly from the intrinsic edge lengths. The subroutine ANGULARSORTFACES simply sorts the triangles around each edge (in circular order); MEASUREEDGELENGTHS gets the initial edge lengths from the vertex positions.

Since the tufted cover $\widetilde{\mathsf{M}}$ is closed, edge-manifold, and gets flipped to an intrinsic Delaunay triangulation, the tufted Laplacian L will have nonnegative edge weights for an input mesh of arbitrary geometry and connectivity. Note that for a closed, manifold input mesh M, the tufted cover will just be two disjoint copies of M, and L is hence the usual intrinsic Laplacian of Bobenko & Springborn [BS07]. However, if M has boundary then we still get something extra: the boundary weights will also be nonnegative, providing a maximum principle relative to *any* set of pinned vertices (see Section 5.5 for further discussion).

Algorithm 3 BUILDTUFTEDLAPLACIAN$(\mathsf{M}, \mathbf{x})$

Input: Any triangle mesh M, with vertex positions $\mathbf{x}$
Output: A $|\mathsf{V}| \times |\mathsf{V}|$ Laplace matrix L for M

```
ρ ← ANGULARSORTFACES(M, x)
M̃, G̃ ← CONSTRUCTTUFTEDCOVER(M, ρ)
ℓ ← MEASUREEDGELENGTHS(M̃, x)              ▷see Section 3.1
ℓ ← MOLLIFY(M̃, ℓ)                         ▷optional, Section 4.5
M̃, G̃, ℓ ← FLIPTODELAUNAY(M̃, G̃, ℓ)
L ← 0 ∈ ℝ^{|V|×|V|}                       ▷initialize a sparse matrix
for each corner i of each ijk ∈ F̃ do
    s ← (l_ij + l_jk + l_ki)/2
    A ← √(s(s−a)(s−b)(s−c))               ▷Heron's area formula
    w_jk ← (1/8)(l_ij^2 + l_ki^2 − l_jk^2)/A   ▷computes (1/2)cot(θ_i^jk)
    L_jk ← L_jk − w_jk                    ▷accumulate entries
    L_kj ← L_kj − w_jk
    L_jj ← L_jj + w_jk
    L_kk ← L_kk + w_jk
return L/2                                ▷ 1/2 factor due to double cover
```

4.5. Intrinsic Mollification

Meshes encountered "in the wild" may have near-degenerate geometry (*e.g.*, near-zero angles or areas) that can impair even basic floating point arithmetic [ZJ16]. Delaunay flips sometimes fix degenerate triangles, but are not guaranteed to do so. Likewise, repairing defects *extrinsically* is hard to do with any kind of guarantee, since small vertex perturbations that fix one element can damage another. For the Laplacian, however, we need only intrinsic data (edge lengths), and can devise a simple *mollification* strategy that is guaranteed to work: just increase the length of all edges by a small, constant amount until no input triangle is degenerate. More precisely, for each corner of each triangle we want

$$\ell_{ij} + \ell_{jk} > \ell_{ki} + \delta, \tag{4}$$

for some user-defined tolerance $\delta > 0$, *i.e.*, we want the triangle inequality to hold with significant inequality, so that triangles are nondegenerate. Then

$$\varepsilon := \max_{ijk} \max(0, \delta - \ell_{ki} - \ell_{ij} + \ell_{jk})$$

is the smallest length we can add to *all* edge lengths to ensure that Equation 4 holds. Updating the edge lengths via $\ell_{ij} \leftarrow \ell_{ij} + \varepsilon$ yields a valid Laplacian even on pathological inputs, as validated in Section 5.2. Moreover, this strategy closely preserves the given geometry: at worst, ε can be just *slightly* larger than δ (due to floating point error); when M is already nondegenerate, $\varepsilon = 0$. Note that this strategy is unrelated to clamping edge weights, which is a non-geometric way of dealing with *non-Delaunay edges* rather than degenerate triangles—and may significantly distort the Laplacian.

5. Analysis and Validation

We examined the behavior of our Laplacian via several common geometry processing algorithms. Note that to visualize the tufted cover we subdivide each triangle into a curved patch; to trace edges of the intrinsic triangulation we use the signpost data structure of [SSC19a] (doing additional work to resolve nonmanifold vertices).

5.1. Performance

We implemented our method in double precision, via `geometry-central` [SC*19], and measured all timings on one thread of an i7-4790K CPU. In general, the extra cost of our method relative to using cotan-Laplace is far less than loading the input from disk, and about as expensive as solving one linear system $\mathsf{L}\mathsf{u} = \mathsf{f}$. For example, on the 57k face mesh in Figure 14 constructing the tufted cover and performing intrinsic Delaunay flips take 96ms and 25ms, respectively; building and solving a linear system using cotan-Laplace on the same mesh takes 106ms.

5.2. Dataset Validation

We were able to successfully construct a Laplacian satisfying the maximum principle on 100% of two real-world datasets: the *Princeton Shape Benchmark*, which contains 1814 low-quality meshes from computer graphics [SMKF04], and *Thingi10k*, comprised of 10,000 meshes from a 3D printing repository [ZJ16]. Both datasets include meshes with extremely degenerate geometry and connectivity, including 1216 and 1971 nonmanifold entries, *resp.*. Only trivial preprocessing was applied, discarding one entry that contained splines, ignoring unreferenced vertices, and omitting triangles with repeated vertices. We converted *Thingi10k* meshes from STL to PLY format via *Meshlab* [CCC*08].

Algorithm 1 successfully builds the tufted cover for all meshes in both datasets, always producing a closed, edge-manifold mesh. Without mollification, Algorithm 2 successfully constructs the iDT on all meshes except those with near-degenerate faces; it succeeds on 100% of meshes after intrinsic mollification with $\delta = 10^{-4}h$, where h is the mean edge length (see Section 4.5). Most importantly, *all* of our final Laplace matrices are symmetric and positive semidefinite, with nonnegative edge weights.

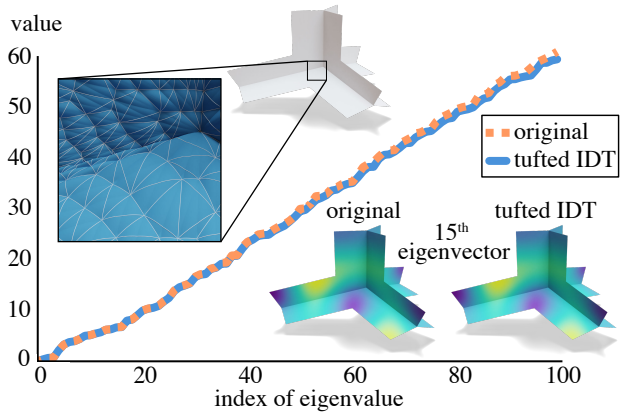


Figure 13: *To verify that our discrete Laplacian correctly approximates the (smooth) Laplace operator, we verify that it has essentially the same low-frequency eigenvectors/eigenvalues as the input—which should not depend strongly on the choice of tessellation.*

5.3. Spectral Validation

One might worry that the Laplacians we build, while structurally valid, might not correctly capture the original geometry. To verify that our Laplacian is indeed geometrically meaningful, we compute its first $k = 100$ eigenvalues, and compare them to the eigenvalues from a high-quality *extrinsic* refinement of the input mesh using the ordinary cotan Laplacian (Figure 13). These eigenvalues characterize the geometry in a mesh-independent way, confirming that our new triangulation still correctly captures the input geometry.

5.4. Face Ordering

The circular ordering ρ^e of faces around an edge e (Algorithm 3) is typically unambiguous. However, even if it is not uniquely determined (*e.g.*, due to coplanar triangles) we find that it has little effect on the Laplacian. The reason is simply that the different linear basis functions induced by different orderings (and hence triangulations) all provide reasonable approximations of the continuous solution space. For instance, in Figure 14 we apply several different ordering strategies (by angle, by face area, and random) on a mesh with

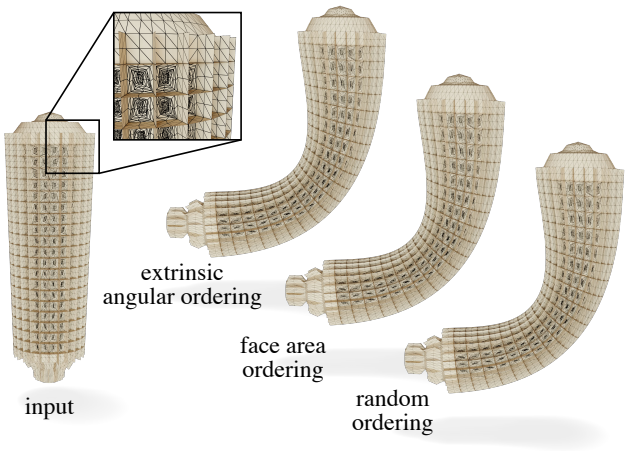


Figure 14: *Our Laplacian does not depend strongly on the ordering of faces around edges—here we compute a deformation using biharmonic weights for several ordering strategies, yielding virtually identical results.*

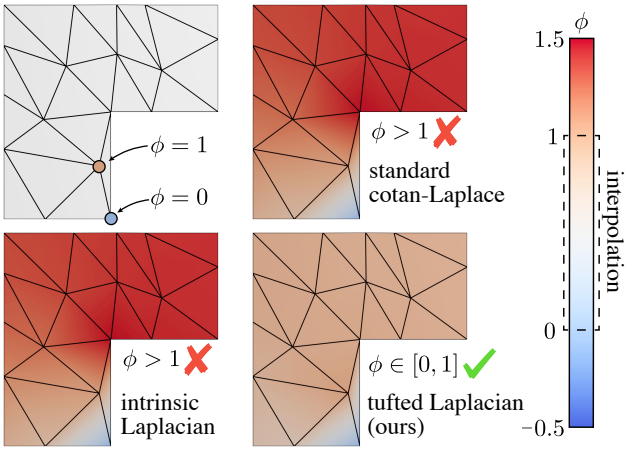
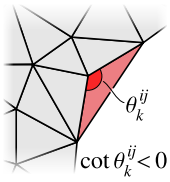


Figure 15: *Unlike previous discrete Laplacians, ours is guaranteed to have nonnegative edge weights even at the domain boundary. Here, harmonic interpolation of a set of pinned values stays inside the given range. In contrast, even the intrinsic Laplacian [BS07] (as well as ordinary cotan-Laplace) yields values far outside this range.*

about 3k nonmanifold edges. Deformations computed via *bounded biharmonic weights* [JBPS11] (which rely on the Laplacian) are essentially indistinguishable: final vertex positions differ by about 10^{-6} on average.

This experiment also demonstrates the value of providing a "drop in replacement" for cotan-Laplace: rather than implementing the biharmonic weight algorithm ourselves (which takes some work due to the need for a QP solver), we simply built our Laplace matrix and passed it to *libigl* [JP*18]. Since we preserve the input vertex set (and hence matrix dimensions), we were able to compute the weights with zero modification to the existing *libigl* code.

5.5. Meshes with Boundary



Our Laplacian has useful properties not only for nonmanifold meshes, but for any mesh with boundary. Both cotan-Laplace and the intrinsic Laplacian will have negative edge weights for a boundary edge ij opposite an obtuse angle θ_k^{ij} (see inset). Delaunay flips are no help here, since boundary edges cannot be flipped. Such weights pose no problem when Dirichlet boundary conditions are enforced along the entire domain boundary (since boundary weights do not enter into the equation), but can be problematic for interpolating other sets of pinned values (Figure 15). In contrast, since our tufted cover is always closed and Delaunay, *all* weights of our Laplacian are nonnegative—even for edges incident on the boundary. Hence, harmonically interpolating *any* set of pinned values will yield a function bounded within the range of these values (Figure 15, *bottom-right*). This property provides additional robustness for algorithms built on top of interpolated weights, Green's functions, *etc.*

5.6. Polygon Mesh Processing

The Laplacian is central to numerous algorithms in polygon mesh processing [BKP*10]. Our Laplacian serves a drop-in replacement for cotan-Laplace, improving accuracy and providing some basic guarantees on robustness—here we examine several examples.

Minimal Surfaces. A classic geometric problem is to construct surfaces of minimal area that interpolate a given boundary curve—such surfaces model the physical behavior of *soap films*, and can be approximated by triangle meshes where the vertex positions themselves are harmonic, *i.e.*, where $\mathsf{L}\mathbf{x} = 0$ [PP93]. Here, nonmanifold configurations arise not due to erroneous mesh connectivity, but rather because physical solutions to this problem can have nonmanifold features. As shown in Figure 18, our Laplacian exhibits numerical robustness even for very poor-quality meshes. In particular, it guarantees that every vertex of the minimal surface is contained in the convex hull of its intrinsic neighbors [BS07, Section 4], preventing flipped triangles and other geometric artifacts that can occur with cotan-Laplace (Figure 18, *right*).

Geodesic Distance. The *heat method* [CWW17] approximates geodesic distance using both the Laplacian, as well as a discrete divergence operator that uses the same cotan weights. Approximating geodesic distance is a good test case, since polyhedral geodesic distance has an exact reference solution that does not depend on how the domain is triangulated [SSK*05], though is much harder to compute. Especially on poorly-triangulated nonmanifold models, we find that our tufted Laplacian comes much closer to this reference solution than cotan-Laplace (Figure 16).

Shape Editing. A common shape editing paradigm is to work in *differential coordinates*, *e.g.*, to modify first or second derivatives of the surface, then recover best-fit vertex positions by solving a system

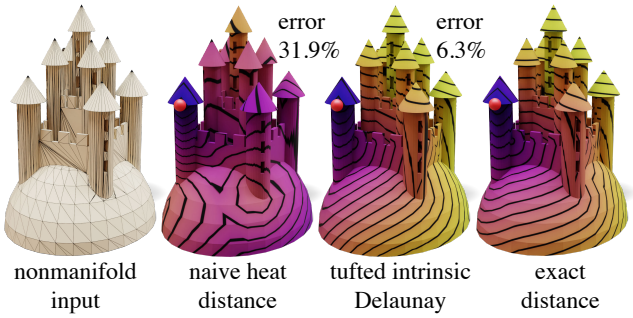


Figure 16: *Our Laplacian greatly improves the accuracy of PDE-based algorithms on poorly tessellated nonmanifold meshes. Here, we compute geodesic distance via the heat method [CWW17], which yields large error with the cotan Laplacian. Substituting our tufted Laplacian brings the result much closer to the exact solution.*

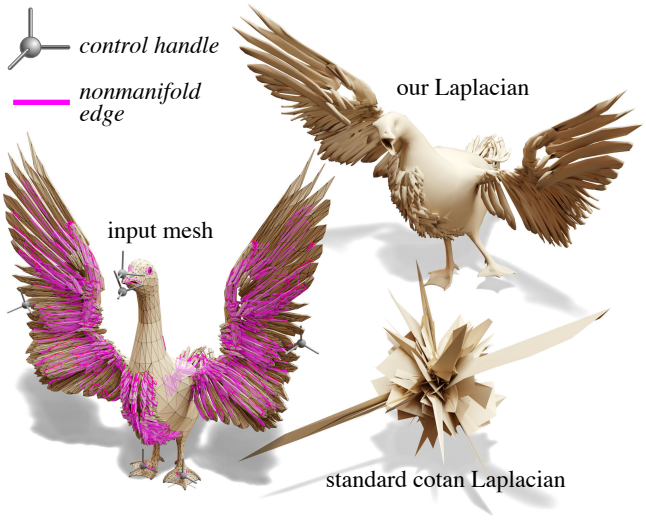


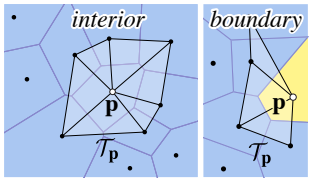
Figure 17: *A poor-quality nonmanifold mesh resulting from a boolean operation* (bottom left) *deformed in the spirit of [LSCO*04]. The standard cotan Laplacian yields nonsensical numerical output, whereas our Laplacian produces the expected deformation.*

involving the Laplacian [YZX*04, LSCO*04, SCOL*04]. Robustness is especially important in this context, since artist-designed models often have poor triangulation quality, and may intentionally include nonmanifold features to help represent a shape concisely. Figure 17 shows one example where our Laplacian dramatically improves robustness of an editing task on a low-quality mesh resulting from boolean operations; here we specify transformations at isolated handles, and interpolate them across the shape to obtain a global deformation. In contrast, cotan-Laplace completely fails to provide a satisfactory solution.

5.7. Point Cloud Laplacians

To obtain a point cloud Laplacian, we build a highly nonmanifold triangulation of the point cloud—our tufted cover then furnishes a Laplacian with all the usual guarantees. This example also provides an extreme "stress test" for our Laplacian, since these triangulations are nonmanifold almost everywhere (Figure 20, *top*).

A common strategy for building a Laplacian on a point cloud $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \subset \mathbb{R}^3$ is to (i) identify the k nearest neighbors of each point $\mathbf{p}$, (ii) project these neighbors onto an estimated tangent plane, and (iii) construct the planar Delaunay triangulation $\mathcal{T}_\mathbf{p}$ of the projected points. These local triangulations are then used to build a Laplacian in a variety of ways. For instance, both Belkin *et al.* [BSW08] and Liu *et al.* [LPG12] use the triangulations purely to determine the mass matrix B; edge weights are determined via a Gaussian function of the distance in $\mathbb{R}^3$ (and are hence always positive). Such schemes are accurate and have nice theoretical properties (*e.g.*, pointwise convergence for fairly uniform point distributions) but involve numerical parameters which are difficult to estimate, and matrices that are quite dense. Other schemes use the local triangulations only to determine connectivity; the original point locations are still used to accumulate cotan weights [CRT04, CTO*10]. A significant benefit of this approach is that (like mesh-based Laplacians) it accurately handles *nonuniform* point distributions, while still retaining a high degree of sparsity. However, since edges may not satisfy the local Delaunay property (Equation 1), the resulting Laplacian can have negative edge weights.



We likewise construct our point cloud Laplacian from local triangulations, but rather than accumulating weights independently, we form the union $\mathcal{T} = \bigcup_{\mathbf{p}\in P} \mathcal{T}_\mathbf{p}$ of triangles containing $\mathbf{p}$ in all local triangulations $\mathcal{T}_\mathbf{p}$; points contained in a noncompact cell of the local Voronoi diagram can be tagged as boundary vertices (see inset). The resulting global triangulation $\mathcal{T}$ has highly irregular connectivity, is nonmanifold almost everywhere, and has duplicate copies of many faces. However, we can simply proceed as before: build the tufted cover, flip to an intrinsic Delaunay triangulation, and read off the corresponding Laplace matrix L—which can then be used directly on the original point cloud P. (We also multiply L by $1/3$, since triangles triply-cover the local neighborhoods.) This Laplacian exhibits all the desired properties (symmetry, positive-definiteness, nonnegative edge weights, *etc.*) while remaining very sparse; unlike schemes based on Gaussian weights, there are no parameters to estimate or tune.

Comparison of Point Cloud Laplacians. We compared point cloud Laplacians by computing associated *Green's functions*; Green's functions of the Laplacian and other linear differential operators are often used as the basis for, *e.g.*, local shape descriptors in shape analysis [WS19]. The smallest number of neighbors that gave reasonable results for the Laplacian of Belkin *et al.* [BSW08] was $k = 30$; even with this minimal neighborhood size, the linear solve was 10x more expensive than with our tufted Laplacian. For both our Laplacian and the Laplacian of Clarenz *et al.* [CRT04], the triangulation has little dependence on the neighborhood size k—it merely needs to be large enough to resolve the triangulation of the center vertex. Though we use the same neighborhood size $k = 30$ to construct local triangulations; the final matrices still have just 7

Figure 18: Left: *Our Laplacian yields the correct geometry for nonmanifold minimal surfaces, even on near-degenerate meshes.* Right: *For a high-quality mesh, the ordinary cotan-Laplacian may still fail to exhibit the maximum principle, producing vertices that are not in the convex hull of their neighbors (as shown here, and in [PR01]), or flipped triangles whose normals differ from their neighbors by more than* 90°.

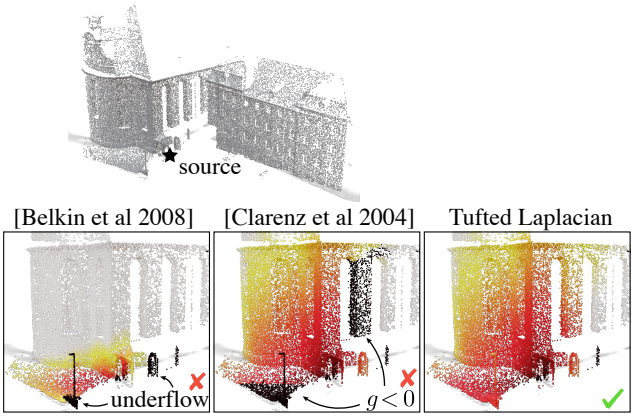


Figure 19: *Harmonic Green's function g on a point cloud. Here, standard Gaussian weights* à la *Belkin* et al. *[BSW08]* (left) *can yield numerical underflow for time steps h that are small enough to resolve the solution. Using Delaunay triangulations to determine connectivity* à la *Clarenz* et al. *[CRT04]* (center) *is more stable, but can still yield erroneous negative values. Our tufted point cloud Laplacian* (right) *guarantees a discrete maximum principle, yielding an accurate result. Scan data from [HSL*17].*

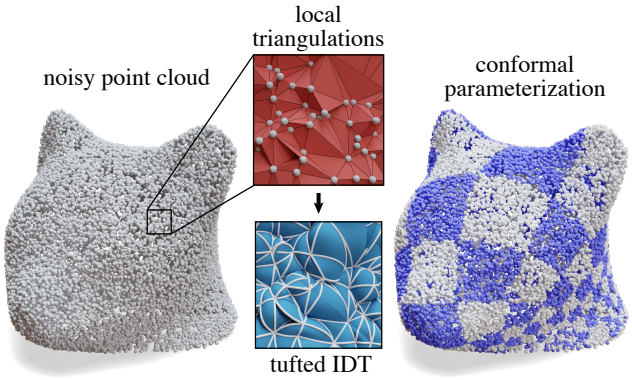


Figure 20: *To construct a high-quality point cloud Laplacian, we take the union of local triangulations about each point, resulting in a nonmanifold mesh on which we construct our tufted Laplacian. Here, we use this Laplacian to compute the spectral conformal parameterization of a surface [MTAD08].*

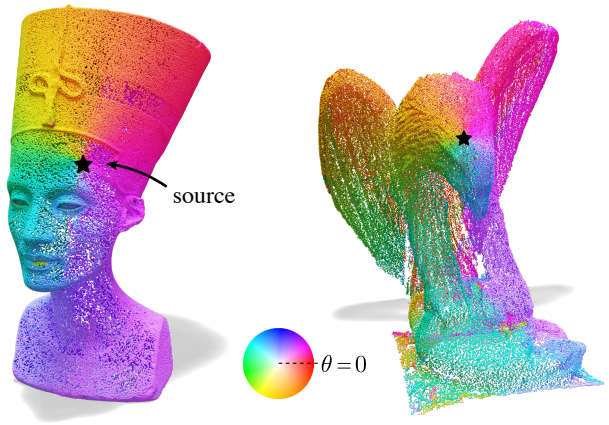


Figure 21: *Our point cloud Laplacian satisfies the same basic properties as cotan-Laplace, enabling mesh-based algorithms to be easily ported to point clouds. Here we plot the angular component* θ *of the logarithmic map, which provides a global surface parameterization around a given point [SSC19b].*

nonzeros in each row on average, as with a standard mesh Laplacian. As shown in Figure 19, the fact that our point cloud Laplacian has positive edge weights and requires no parameter tuning makes it a practical and useful alternative to existing schemes.

Point Cloud Parameterization. In Figure 20 we used our Laplacian to generate a conformal parameterization of a noisy point cloud, adapting the mesh-based method of [MTAD08]. This method also requires an expression for the area of the flattened surface, which we write as the sum over all oriented triangles in the tufted cover (encoded as a bilinear form). Estimated point normals are used to determine triangle orientations. We did not find it necessary to use a boundary-only mass matrix, though boundary points can be identified if desired—see above. Finally, Figure 21 shows a more sophisticated algorithm run directly on point clouds: parameterization by the logarithmic map, computed via the *vector heat method* [SSC19b, Section 8.2].

6. Limitations and Future Work

The usual trade-off with the intrinsic approach (not specific to this method) is that the piecewise linear bases used to perform computation differ from those of the input mesh. For some applications (such as computing eigenvalues, or geodesic distances) this fact is irrelevant: values on the input and intrinsic triangulation carry precisely the same meaning. When *interpolating* nodal values (as in parameterization, for instance) it may be more natural to use the so-called *overlay mesh* (see [SSC19a, Section 3.4]), though for many applications interpolating an intrinsic solution in the input basis still improves results dramatically (*e.g.*, Figure 17).

As noted in Section 3.3, our Laplace operator does not exhibit the "locality" property of Wardetzky *et al.*, *i.e.*, the local neighborhood may be different than in the input. However, one should carefully consider the practical implications: in the end, our Laplacian provides a principled discretization of the (continuous) Laplacian on the input domain, simply using a different—and often, higher-quality—piecewise linear finite element space.

Apart from some simple mollification (Section 4.5), our Laplacian does nothing to *repair* defective input (*e.g.*, by filtering outliers or removing topological noise); instead, it assumes that the input mesh correctly represents the desired domain. For instance, vertices that are coincident in space but have distinct indices will not be connected in the tufted cover. If such features are undesirable, one should seek out standard tools for mesh repair [ACK13].

Our tufted cover might be used to extend other operators to the nonmanifold setting—such as those from discrete exterior calculus [DHLM05], vector field processing [dGDT16, VCD*16], or even just other discretizations of the Laplacian (Section 2). The basic strategy is the same: build the tufted cover, flip to Delaunay, and use the resulting edge lengths to evaluate any local geometric quantities appearing in the definition of the operator. However, some thought is required to transfer vector-valued data to and from the intrinsic triangulation; Sharp *et al.* [SSC19b, Section 5.4] explore one possible scheme.

At a high level, our construction helps extend the Delaunay condition and its associated guarantees to nonmanifold meshes and point clouds—such guarantees undoubtedly provide new opportunities for robust geometry processing.

Acknowledgements

Thanks to Boris Springborn and Max Wardetzky for helpful discussions. This work was supported by NSF award 1943123, an NSF Graduate Fellowship, a Packard Fellowship, and gifts from Autodesk, Adobe, Activision Blizzard, Disney, and Facebook. Thanks to creators of meshes: Thingiverse user *Ramenspork*'s high school students (Figure 16) and TurboSquid user *Gatzegar* (Figure 17).

References

[ACK13] ATTENE M., CAMPEN M., KOBBELT L.: Polygon Mesh Repairing: An Application Perspective. *Comp. Surv. 45*, 2 (2013). 11

[AKL13] AURENHAMMER F., KLEIN R., LEE D.-T.: *Voronoi Diagrams and Delaunay Triangulations*. World Scientific, 2013. 4

[AW11] ALEXA M., WARDETZKY M.: Discrete Laplacians on General Polygonal Meshes. In *SIGGRAPH 2011*. 2011. 2, 3

[BBL*17] BRONSTEIN M. M., BRUNA J., LECUN Y., SZLAM A., VANDERGHEYNST P.: Geometric Deep Learning: Going Beyond Euclidean Data. *IEEE Signal Processing Magazine 34*, 4 (2017). 1

[BKP*10] BOTSCH M., KOBBELT L., PAULY M., ALLIEZ P., LÉVY B.: *Polygon Mesh Processing*. CRC press, 2010. 1, 8

[BS07] BOBENKO A. I., SPRINGBORN B. A.: A Discrete Laplace–Beltrami Operator For Simplicial Surfaces. *Disc. & Comp. Geom. 38*, 4 (2007). 1, 4, 6, 8

[BSW08] BELKIN M., SUN J., WANG Y.: Discrete Laplace Operator on Meshed Surfaces. In *Symp. Comp. Geom.* (2008). 2, 3, 9, 10

[BSW09] BELKIN M., SUN J., WANG Y.: Constructing Laplace Operator from Point Clouds in $\mathbb{R}^d$. In *Symp. Disc. Alg.* (2009), SIAM. 3

[CCC*08] CIGNONI P., CALLIERI M., CORSINI M., DELLEPIANE M., GANOVELLI F., RANZUGLIA G.: MeshLab: an Open-Source Mesh Processing Tool. In *Eurograph. Ital. Chap. Conf.* (2008). 7

[CDGDS13] CRANE K., DE GOES F., DESBRUN M., SCHRÖDER P.: Digital Geometry Processing with Discrete Exterior Calculus. In *ACM SIGGRAPH Courses*. 2013, pp. 1–126. 2

[CDS12] CHENG S.-W., DEY T. K., SHEWCHUK J.: *Delaunay Mesh Generation*. Chapman & Hall, 2012. 3

[CH11] CHEN L., HOLST M. J.: Efficient Mesh Optimization Schemes based on Optimal Delaunay Triangulations. *Comput. Meth. Appl. Mech. Engrg. 200* (2011). 3

[CRT04] CLARENZ U., RUMPF M., TELEA A.: Finite Elements on Point Based Surfaces. In *SPBG* (2004). 3, 9, 10

[CTO*10] CAO J., TAGLIASACCHI A., OLSON M., ZHANG H., SU Z.: Point Cloud Skeletons via Laplacian Based Contraction. In *Shape Mod. Int. Conf.* (2010), IEEE. 3, 9

[CWW17] CRANE K., WEISCHEDEL C., WARDETZKY M.: The Heat Method for Distance Computation. *Comm. ACM 60*, 11 (2017). 8, 9

[DFGH04] DE FLORIANI L., GREENFIELDBOYCE D., HUI A.: A Data Structure for Non-manifold Simplicial d-Complexes. In *Symp. Geom. Proc.* (2004). 2

[dGDT16] DE GOES F., DESBRUN M., TONG Y.: Vector Field Processing on Triangle Meshes. In *SIGGRAPH Courses*. 2016. 11

[DHLM05] DESBRUN M., HIRANI A. N., LEOK M., MARSDEN J. E.: Discrete Exterior Calculus. *arXiv preprint math/0508341* (2005). 11

[Duf59] DUFFIN R. J.: Distributed and Lumped Networks. *J. Math. Mech.* (1959). 2

[FSSB07] FISHER M., SPRINGBORN B., SCHRÖDER P., BOBENKO A. I.: An Algorithm for the Construction of Intrinsic Delaunay Triangulations with Applications to Digital Geometry Processing. *Comp. 81*, 2 (2007). 2, 4, 6

[GBTS99] GUÉZIEC A., BOSSEN F., TAUBIN G., SILVA C.: Efficient Compression of Non-manifold Polygonal Meshes. *Comp. Geom. 14*, 1-3 (1999). 2

[GKS02] GRINSPUN E., KRYSL P., SCHRÖDER P.: CHARMS: a Simple Framework for Adaptive Simulation. 3

[Hat02] HATCHER A.: *Algebraic Topology*. Cambridge University Press, 2002. 5

[HG00] HUBELI A., GROSS M.: Fairing of Non-manifolds for Visualization. In *Proc. Vis.* (2000), IEEE. 2

[HKA15] HERHOLZ P., KYPRIANIDIS J. E., ALEXA M.: Perfect Laplacians for Polygon Meshes. In *Comp. Graph. Forum* (2015). 2

[HP11] HILDEBRANDT K., POLTHIER K.: On Approximation of the Laplace-Beltrami Operator and the Willmore Energy of Surfaces. In *Comp. Graph. Forum* (2011), vol. 30. 2, 3

[HSL*17] HACKEL T., SAVINOV N., LADICKY L., WEGNER J. D., SCHINDLER K., POLLEFEYS M.: Semantic3d.net: A New Large-scale Point Cloud Classification Benchmark. *arXiv:1704.03847* (2017). 10

[HZG*18] HU Y., ZHOU Q., GAO X., JACOBSON A., ZORIN D., PANOZZO D.: Tetrahedral Meshing in the Wild. *ACM Trans. Graph. 37*, 4 (2018). 2

[JBPS11] JACOBSON A., BARAN I., POPOVIC J., SORKINE O.: Bounded Biharmonic Weights for Real-time Deformation. *ACM Trans. Graph. 30*, 4 (2011). 8

[JKSH13] JACOBSON A., KAVAN L., SORKINE-HORNUNG O.: Robust Inside-outside Segmentation Using Generalized Winding Numbers. *ACM Trans. Graph. 32*, 4 (2013). 2

[JP*18] JACOBSON A., PANOZZO D., ET AL.: libigl: A simple C++ geometry processing library, 2018. https://libigl.github.io/. 8

[KCPS15] KNÖPPEL F., CRANE K., PINKALL U., SCHRÖDER P.: Stripe patterns on surfaces. *ACM Trans. Graph. 34*, 4 (2015), 39. 2

[KNP07] KÄLBERER F., NIESER M., POLTHIER K.: Quadcover: Surface Parameterization using Branched Coverings. In *Comp. Graph. Forum* (2007), vol. 26. 2

[LFXH17] LIU Y.-J., FAN D., XU C.-X., HE Y.: Constructing Intrinsic Delaunay Triangulations from the Dual of Geodesic Voronoi Diagrams. *ACM Trans. Graph. 36*, 2 (2017). 2, 3

[LPG12] LIU Y., PRABHAKARAN B., GUO X.: Point-based Manifold Harmonics. *IEEE Trans. Vis. Comp. Graph. 18*, 10 (2012). 3, 9

[LSCO*04] LIPMAN Y., SORKINE O., COHEN-OR D., LEVIN D., ROSSI C., SEIDEL H.-P.: Differential Coordinates for Interactive Mesh Editing. In *Proc. Shape Mod. App.* (2004). 9

[LXFH15] LIU Y.-J., XU C.-X., FAN D., HE Y.: Efficient Construction and Simplification of Delaunay Meshes. *ACM Trans. Graph.* (2015). 3

[Mac49] MACNEAL R.: *The Solution of Partial Differential Equations by Means of Electrical Networks*. PhD thesis, Caltech, 1949. 2

[MD10] MAUS A., DRANGE J. M.: All closest neighbors are proper delaunay edges generalized, and its application to parallel algorithms. *Proc. Nor. Inf.* (2010). 4

[MMdGD11] MULLEN P., MEMARI P., DE GOES F., DESBRUN M.: HOT: Hodge-optimized triangulations. In *SIGGRAPH*. 2011. 2

[MTAD08] MULLEN P., TONG Y., ALLIEZ P., DESBRUN M.: Spectral Conformal Parameterization. In *Comp. Graph. Forum* (2008), vol. 27, Wiley Online Library. 10

[NPPZ11] NIESER M., PALACIOS J., POLTHIER K., ZHANG E.: Hexagonal Global Parameterization of Arbitrary Surfaces. *IEEE Trans. Vis. Comp. Graph. 18*, 6 (2011). 2

[NRP11] NIESER M., REITEBUCH U., POLTHIER K.: Cubecover: Parameterization of 3D Volumes. In *Comp. Graph. Forum* (2011). 2

[PP93] PINKALL U., POLTHIER K.: Computing Discrete Minimal Surfaces and their Conjugates. *Exp. Math. 2*, 1 (1993). 1, 2, 8

[PR01] POLTHIER K., ROSSMAN W.: Counterexample to the Maximum Principle for Discrete Minimal Surfaces. *Electronic Geometry Models* (2001). 10

[QSMG17] QI C. R., SU H., MO K., GUIBAS L. J.: Pointnet: Deep Learning on Point Sets for 3d Classification and Segmentation. In *Proc. Comp. Vis. Patt. Rec.* (2017). 1

[Riv94] RIVIN I.: Euclidean Structures on Simplicial Surfaces and Hyperbolic Volume. *Ann. Math. 139*, 3 (1994). 4

[RKG*17] ROY L., KUMAR P., GOLBABAEI S., ZHANG Y., ZHANG E.: Interactive Design and Visualization of Branched Covering Spaces. *IEEE Trans. Vis. Comp. Graph. 24*, 1 (2017). 2

[RWP06] REUTER M., WOLTER F.-E., PEINECKE N.: Laplace–Beltrami Spectra as 'Shape-DNA' of Surfaces and Solids. *Comp.-Aid. Des. 38*, 4 (2006). 3

[SC*19] SHARP N., CRANE K., ET AL.: geometry-central, 2019. www.geometry-central.net. 7

[SCM*19] SELLÁN S., CHENG H. Y., MA Y., DEMBOWSKI M., JACOBSON A.: Solid Geometry Processing on Deconstructed Domains. *Comp. Graph. Forum* (2019). 2

[SCOL*04] SORKINE O., COHEN-OR D., LIPMAN Y., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian Surface Editing. In *Symp. Geom. Proc.* (2004). 9

[SG95] SHIMADA K., GOSSARD D. C.: Bubble Mesh: Automated Triangular Meshing of Non-manifold Geometry by Sphere Packing. In *Proc. Symp. Solid Mod. App.* (1995). 2

[SHD*18] SCHNEIDER T., HU Y., DUMAS J., GAO X., PANOZZO D., ZORIN D.: Decoupling Simulation Accuracy from Mesh Quality. *ACM Trans. Graph. 37*, 5 (2018). 3

[She02] SHEWCHUK J. R.: What is a Good Linear Element? Interpolation, Conditioning, and Quality Measures. In *IMR* (2002). 2, 3

[SMKF04] SHILANE P., MIN P., KAZHDAN M., FUNKHOUSER T.: The Princeton shape benchmark. In *Shape Mod. Int.* (2004). 7

[SSC19a] SHARP N., SOLIMAN Y., CRANE K.: Navigating Intrinsic Triangulations. *ACM Trans. Graph. 38*, 4 (2019). 2, 3, 4, 5, 6, 7, 11

[SSC19b] SHARP N., SOLIMAN Y., CRANE K.: The Vector Heat Method. *ACM Trans. Graph. 38*, 2 (2019). 2, 10, 11

[SSK*05] SURAZHSKY V., SURAZHSKY T., KIRSANOV D., GORTLER S. J., HOPPE H.: Fast Exact and Approximate Geodesics on Meshes. In *ACM Trans. Graph.* (2005), vol. 24, Acm. 8

[VCD*16] VAXMAN A., CAMPEN M., DIAMANTI O., PANOZZO D., BOMMES D., HILDEBRANDT K., BEN-CHEN M.: Directional Field Synthesis, Design, and Processing. In *Comp. Graph. Forum* (2016), vol. 35. 11

[WLG03] WAGNER M., LABSIK U., GREINER G.: Repairing Non-manifold Triangle Meshes using Simulated Annealing. *Int. J. Shape Mod. 9*, 02 (2003). 2

[WMKG07] WARDETZKY M., MATHUR S., KAELBERER F., GRINSPUN E.: Discrete Laplace Operators: No Free Lunch. In *Symp. Geom. Proc.* (2007). 1, 2, 4

[WS19] WANG Y., SOLOMON J.: Intrinsic and Extrinsic Operators for Shape Analysis. In *Handbook of Num. Anal.*, vol. 20. 2019. 9

[YZ01] YING L., ZORIN D.: Nonmanifold Subdivision. In *Proc. Vis.* (2001), IEEE. 2

[YZX*04] YU Y., ZHOU K., XU D., SHI X., BAO H., GUO B., SHUM H.-Y.: Mesh Editing with Poisson-based Gradient Field Manipulation. In *SIGGRAPH*. 2004. 9

[ZJ16] ZHOU Q., JACOBSON A.: Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv:1605.04797* (2016). 1, 7