Eurographics Symposium on Geometry Processing 2015
Mirela Ben-Chen and Ligang Liu
(Guest Editors)

Volume 34 (2015), Number 5

Perfect Laplacians for Polygon Meshes

Philipp Herholz Jan Eric Kyprianidis Marc Alexa

TU Berlin

Abstract

A discrete Laplace-Beltrami operator is called perfect if it possesses all the important properties of its smooth counterpart. It is known which triangle meshes admit perfect Laplace operators and how to fix any other mesh by changing the combinatorics. We extend the characterization of meshes that admit perfect Laplacians to general polygon meshes. More importantly, we provide an algorithm that computes a perfect Laplace operator for any polygon mesh without changing the combinatorics, although, possibly changing the embedding. We evaluate this algorithm and demonstrate it at applications.

1. Introduction

Discrete approximations of the Laplace-Beltrami operator are at the heart of many mesh processing techniques, such as representation [Sor06], deformation [BS08], spectral processing [ZVKD10], or descriptors [BBGO11]. For triangle meshes the cotan-Laplacian [PP93, DMSB99] is a common choice because it enjoys many desirable properties. However, as discussed by Wardetzky et al. [WMKG07], discrete Laplace operators cannot be *perfect* in the sense that they may fail to posses all properties of their smooth counterpart for an arbitrary triangle mesh. We recall the desired properties in Section 2 and explain why they are important for mesh processing applications.

The notion of perfect Laplace operators can be linked to force networks in equilibrium that only pull vertices along edges. This viewpoint allows the characterization of triangle meshes that admit perfect Laplace operators, namely, so called *regular* or *weighted Delaunay* meshes. Vertex weights can be used to generalize the cotan-Laplacian [Gli05, Gli07], and for any weighted Delaunay mesh there exists a perfect Laplace operator of this form [dGMMD14].

The situation is far less developed for general polygon meshes, i.e., meshes with face degrees larger than three. Alexa and Wardetzky [AW11] provide a construction that reduces to the cotan-Laplace for triangle meshes, but it is unclear for which meshes this operator is perfect. More importantly, there is no obvious generalization for weighted meshes, and therefore the question of existence of perfect Laplace operators for polygon meshes is open. Using the connection to orthogonal duals or force networks we are able to close this gap and generally describe (planar) meshes that admit perfect Laplacians (see Section 4).

Our main contribution is an algorithm that generates perfect Laplace operators for *any* polygonal mesh *without changing the combinatorics* but possibly *changing the embedding*. The main ideas for this algorithm are as follows:

1. We identify a perfect Laplace operator for a mesh with given combinatorics with a unique embedding by fixing the boundary. This defines a mapping from the coefficients of the Laplace operator to the edge lengths.
2. By analogy to force networks, we derive a simple update rule for the coefficients such that the edge lengths converge to the desired ones (if possible).

We describe and discuss this algorithm in Section 3. An important point of the algorithm is that it is based only on defining boundary conditions and measuring edge lengths of the embedding and, thus, has a natural extension to (possibly closed) non-planar meshes embedded in higher dimension.

We demonstrate the construction of perfect Laplace operators for polygon meshes by means of self-supporting surfaces with polygonal tiles as well as by computing parameterizations in Section 6 and discuss the properties of our construction, in particular in relation to other approaches, in Section 7.

2. Discrete Laplace operators

The smooth Laplace-Beltrami operator has several properties that naturally translate into properties of a discrete analogue. Loosely following Wardetzky et al. [WMKG07], we call a discrete Laplace operator *perfect* if it is locally defined, symmetric, non-negative, affinely invariant (the constant functions

are in the kernel), and has linear precision. In the following, we detail these properties and discuss implications.

Let $\mathcal{M} = (V, E, F)$ be a polygon mesh. A *(discrete) Laplace operator* $\mathbf{L}$ on $\mathcal{M}$ is defined by symmetric coefficients $\omega_{ij} = \omega_{ji}$ associated to each edge $(i, j) \in E$, acting on real-valued functions $\mathbf{u} : V \mapsto \mathbb{R}$, $i \mapsto u_i$ by:

$$(\mathbf{Lu})_i = \sum_{(i,j)\in E} \omega_{ij}(u_j - u_i)\,. \tag{1}$$

This definition identifies all discrete Laplacians that are local and symmetric in the sense of Wardetzky. Being defined on the differences of vertex values, the constant functions are always in the kernel, making the operator affinely invariant.

Non-negativity A Laplace operator is said to be non-negative, if all its coefficients are non-negative and if there is a spanning tree consisting of edges with positive coefficients. In combination with symmetry, this property ensures that (i) the operator is positive semi-definite and (ii) harmonic functions (with respect to the operator and appropriate boundary conditions) obey the maximum principle. This principle states that harmonic functions take their extremal values at the boundary. In the context of our work (and more generally for the application of computing parameterizations) it ensures that Laplacians can be used to obtain embeddings (with convex boundary) of planar graphs [Tut63].

Linear precision In the smooth setting, the Laplace-Beltrami operator $\mathbf{L}$ applied to the embedding $\mathbf{x}$ of a smooth 2D manifold yields the area gradient [dC76]. In particular, it vanishes on planar domains. Hence, in the discrete setting we ask that $(\mathbf{Lu})_i = \mathbf{0}$ if i is an interior vertex and all edges $(i, j) \in E$ are contained in a plane. Linear precision implies that curvature flows modify the mesh only in normal direction (i.e., there is no tangential smoothing) and parameterization based on the Laplacian preserves meshes in the plane.

Force networks For symmetric Laplace operators of the form in Eq. (1), the consequences of linear precision and non-negativity can be made more intuitive by considering the (planar) mesh as a force network [Max64]: each edge is a spring pulling or pushing the incident vertices together or apart from each other. In this picture, the coefficients ω_{ij} take the role of spring constants.

Consider the mesh to be embedded with vertex positions $\{\mathbf{x}_i\}$ and edges $\mathbf{e}_{ij} = \mathbf{x}_i - \mathbf{x}_j$. Then Hook's law takes the form

$$\mathbf{f}_{ij} = \omega_{ij}\,\mathbf{e}_{ij}\,, \tag{2}$$

describing the force vector $\mathbf{f}_{ij}$ acting on the spring (i, j). Comparing this to Eq. (1), we see that applying the Laplace operator to the vertex positions yields the sum of the forces for each node. Hence, linear precision means that forces cancel in each interior node or, in other words, the force network is in equilibrium. The sign of the coefficients ω_{ij} distinguishes between pulling or pushing forces. If all coefficients are positive, vertices are in equilibrium and the springs along all edges are pulling.

No free lunch Based on the force network interpretation of Laplacians, Wardetzky et al. [WMKG07] prove that there are triangle meshes without a discrete Laplacian of the form in Eq. (1) obeying both linear precision and non-negativity (see Figure 2 for an example). However, some applications call for a perfect Laplacian: parameterization techniques based on the Laplacian, for example, require an injective mapping; yet this is only guaranteed if the coefficients are non-negative (and the boundary is fixed in convex position). Asking additionally that planar meshes are mapped to themselves requires linear precision. See Section 6 for details on parameterization. Further applications include the computation of self supporting surfaces (Section 6) and the fitting of discrete minimal surfaces with fixed convex boundaries.

3. Method

In this section, we derive an algorithm for constructing a perfect Laplace operator for a given mesh. As not all meshes admit such an operator, the mesh vertices may be perturbed by the algorithm. The idea of our approach for planar meshes can be intuitively described in terms of force networks: starting from an arbitrary set of positive spring constants, we compute vertex positions by fixing the boundary. Then we compare the resulting edge lengths to the desired ones. If an edge is too short we lower the spring constant, yet keeping it positive. If the edge is too long we increase the spring constant.

In the following we describe this idea rigorously, providing a principled way for updating the coefficients. By introducing different boundary conditions, we can extend this idea to meshes embedded into $\mathbb{R}^3$. An analysis of the steady state yields a characterization of the solution.

3.1. Planar polygon meshes

Let $\mathcal{M} = (V, E, F)$ be a planar 3-connected polygon mesh with single boundary $\partial\mathcal{M}$, and let the geometry be given by vertex positions $\hat{\mathbf{X}} \in \mathbb{R}^{|\mathcal{V}|\times 2}$, where the positions of vertices appear as row vectors. The i-th row is referred to as $\hat{\mathbf{x}}_i$, i.e., the vertex position of vertex i. Edge lengths of the input mesh are denoted by $\|\hat{\mathbf{e}}_{ij}\|$. We assume that the boundary vertices are in strictly convex position[†].

Let the coefficients of the Laplace operator and the edge lengths at the k-th iteration of the algorithm be given by

$$\omega_{ij}^{(k)} > 0 \quad \text{and} \quad \left\|\mathbf{e}_{ij}^{(k)}\right\|. \tag{3}$$

We start with $\{\omega_{ij}^{(0)} = 1\}$ (or any other positive choice). Then we repeatedly embed the mesh with the given Laplace operator and update the coefficients. The updates preserve positivity of the coefficients, ensuring their positivity for all

iterations. This algorithm is summarized as pseudo code in Algorithm 1. The details are discussed in the remainder of this section.

Embedding Since we assume that the polygon mesh $\mathcal{M}$ has a convex boundary, a solution for the equilibrium state of the spring network can be obtained by solving Laplace's equation given by the coefficients ω_{ij} subject to the constraint that the boundary $\{\hat{\mathbf{x}}_i : i \in \partial\mathcal{M}\}$ is held fixed. More specifically, at each iteration k, we solve

$$\Omega^{(k)}\mathbf{X}^{(k)} = \mathbf{b} \tag{4}$$

with

$$\Omega_{ij}^{(k)} = \begin{cases} \omega_{ij}^{(k)} & (i,j) \in E,\ i \notin \partial\mathcal{M} \\ -\sum_j \omega_{ij}^{(k)} & i = j,\ i \notin \partial\mathcal{M} \\ 1 & i = j,\ i \in \partial\mathcal{M} \\ 0 & \text{otherwise} \end{cases} \qquad \mathbf{b}_i = \begin{cases} 0 & i \notin \partial\mathcal{M} \\ \hat{\mathbf{x}}_i & i \in \partial\mathcal{M} \end{cases}$$

Since the coefficients are non-negative by construction, the resulting vertex geometry $\mathbf{X}^{(k)}$ is a *unique embedding*. This follows from Tutte's spring-embedding theorem [Tut63] and its extension by Floater [Flo97]. In particular, uniqueness implies that we have a mapping

$$\omega_{ij} \mapsto \|\mathbf{e}_{ij}\| \tag{5}$$

from the Laplace operator coefficients to edge lengths. Note that this mapping is invariant to multiplying all ω_{ij} with a constant positive scalar.

Update rule We want to update the coefficients $\omega_{ij}^{(k)}$ such that $\|\mathbf{e}_{ij}^{(k)}\|$ gets closer to the edge length $\|\hat{\mathbf{e}}_{ij}\|$ of the input mesh. Since the coefficients are positive, we can derive from Hook's law that

$$F_{ij} = \|\mathbf{f}_{ij}\| = \omega_{ij}^{(k)} \left\|\mathbf{e}_{ij}^{(k)}\right\|. \tag{6}$$

Assuming that forces stay constant for small changes in the spring constants suggests the following update for the spring constants:

$$\omega_{ij}^{(k+1)} = \frac{F_{ij}}{\left\|\hat{\mathbf{e}}_{ij}\right\|}. \tag{7}$$

Combining Eqs. (6) and (7) eliminates the dependance on forces and yields our simple multiplicative update rule for the coefficients:

$$\omega_{ij}^{(k+1)} = \omega_{ij}^{(k)} \frac{\left\|\mathbf{e}_{ij}^{(k)}\right\|}{\left\|\hat{\mathbf{e}}_{ij}\right\|}. \tag{8}$$

After updating all coefficients, we scale them uniformly to avoid that the coefficients become arbitrarily small or large, potentially causing numerical problems. Note that the fraction of current to original edge lengths is always positive so that $\omega_{ij}^{(k)} > 0$ for all k, as desired.

Algorithm 1: Compute perfect Laplacian

Input: A mesh $\mathcal{M} = (V,E,F)$, coordinates $\hat{\mathbf{x}}$.
Output: Coordinates $\mathbf{x}'$, coefficients ω_{ij}.

$\omega_{ij} \leftarrow 1$
for $k = 1, \ldots,$ *max_iters* **do**
  solve $\Omega\mathbf{x}^{(k)} = \mathbf{b}$
  $\omega_{ij}^{(k+1)} \leftarrow \omega_{ij}^{(k)} \frac{\|\mathbf{x}_j^{(k)} - \mathbf{x}_j^{(k)}\|}{\|\hat{\mathbf{x}}_j - \hat{\mathbf{x}}_i\|}$
  $\omega_{ij}^{(k+1)} \leftarrow \omega_{ij}^{(k+1)} / \sqrt{\sum_{ij} \left(\omega_{ij}^{(k+1)}\right)^2}$
  if $\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\| < \varepsilon$ **then break**
end
return $\mathbf{x}^{(k)}, \omega_{ij}^{(k)}$

3.2. Polygon meshes in space

The algorithm can be readily generalized to polygonal meshes in 3D. The property of non-negativity carries over directly. Linear precision on the other hand must be replaced by a condition on the mean curvature normal

$$(\mathbf{L}\mathbf{x})_i = H_i\mathbf{n}_i, \tag{9}$$

with H_i being the mean curvature and $\mathbf{n}_i$ being the normal at vertex i. The mean curvature normal may be computed using the cotan-Laplacian or the area gradient [AW11]. Condition (9) is consistent with the planar case, since discrete mean curvature vanishes for planar vertices. Hence, to compute perfect Laplacians for closed 3D meshes, we replace the right hand side of Eq. (4) with

$$b_i = \begin{cases} H_i\mathbf{n}_i & \notin \partial\mathcal{M} \\ \hat{x}_i & i \in \partial\mathcal{M} \end{cases}. \tag{10}$$

3.3. Steady state of the solution

We have applied the algorithm to numerous meshes and in all our examples it converged quickly and uniformly to a steady state. An empirical study of convergence will be presented in Section 5. Based on this experimental evidence we conjecture that Algorithm 1 converges for arbitrary input meshes with prescribed convex boundary.

It is instructive, however, to inspect the steady state of the solution. Recall that we scale the ω_{ij} uniformly and denote this factor μ. The update in Eq. (8) reaches a steady state if, for any edge, either

$$\frac{\left\|\mathbf{e}_{ij}^{(\infty)}\right\|}{\left\|\hat{\mathbf{e}}_{ij}\right\|} = \mu^{-1} \quad \text{or} \quad \omega_{ij}^{(\infty)} = 0. \tag{11}$$

In particular, if all $\omega_{ij} > 0$ then all original edge lengths are preserved. This means that if a mesh admits a perfect Laplace operator, it is a steady state of our algorithm. If the mesh admits no perfect Laplace operator, a subset of edges

(in planar regions) have zero Laplace coefficients while the rest is scaled by a constant factor relative to the original geometry. In the latter case, the lengths of edges corresponding to zero Laplace coefficients are determined by the embedding based on the edges with positive coefficients. In a sense, the algorithm resorts to a subset of the edges, i.e., a coarsening of the mesh.

4. Meshes admitting perfect Laplacians

Unless we can characterize the meshes that admit perfect Laplace operators, it is difficult to claim that the algorithm always performs as expected. In the following, we will therefore extend know results for triangle meshes [WMKG07] to the general polygonal case.

Recall that a perfect Laplace operator induces a force network that is in equilibrium where all forces are pulling. Note that, conversely, the existence of such a force network implies the existence of a perfect Laplacian, noting that the spring constants must be positive. So the existence of perfect Laplacians corresponds exactly to the existence of pulling forces such that the nodes are in equilibrium. There is a useful characterization for such meshes: the vertices can be lifted such that their convex hull provides the given combinatorics [Aur87]. In particular, triangle meshes with this property are referred to as *regular*[‡], yielding the characterization of meshes that admit perfect Laplace operators in the sense of Wardetzky et al. [WMKG07].

However, the notion of meshes admitting lifts into convex position applies more generally to any planar mesh and the meshes that do admit such a lift are called *regular subdivisions*. Jaume and Roter [JR13] analyze the case of regular subdivisions in detail and in particular distinguish between convex and strictly convex lifts. This distinction also appears in the solutions provided by our algorithm (see Section 3.3), as it translates to distinguishing zero and strictly positive coefficients in the Laplace operator. Jaume and Roter also apply their results to the case of force networks, and we can adapt their Proposition 4.4 to derive the following characterization:

> A polygon mesh in the plane admits a perfect Laplace operator if there exists a lift of the vertex positions such that the combinatorics of the convex hull of the vertices is a 3-connected spanning subgraph of the mesh.

Any edge not part of the subgraph corresponds to a dual edge of zero length or, equivalently, to a zero coefficient of the Laplacian.

The polygons in regular subdivisions are quite restricted. In particular, since they are projections of polyhedral faces, they are necessarily convex. In this view, it is not surprising that Alexa and Wardetzky [AW11] implicitly consider a more general construction: any two vertices that are part of the same face may carry a non-zero coefficient. We may characterize the meshes that admit a perfect Laplacian with such additional diagonals as follows: there exists a lift of the vertex positions such that *a subgraph of* the combinatorics of the convex hull of the vertices is a 3-connected spanning subgraph of the mesh. In particular, a polygon mesh in this case admits a perfect Laplace operator if it is contained in a regular triangulation of the vertices.

Note that the natural notion of allowing non-zero coefficients means that faces with higher degree are an obstruction for perfect Laplace operators, while allowing diagonals means that faces with higher degree relax the situation. We discuss the practical consequences of this distinction further in Section 7.

5. Results

In order to verify the validity of our approach, we performed a series of experiments covering different classes and types of meshes. We check for convergence as described in Algorithm 1 with a value of $\varepsilon = 10^{-6}$. All meshes have been scaled to fit a unit bounding box. For all experiments we conducted, the algorithm converged numerically.

Triangle meshes Our first test case concerned triangular meshes. To this end, we created random triangle meshes by computing tessellations of the unit square and unit circle using the software Triangle [She02]. In addition, we created convex tessellations of random point clouds using CGAL [The15]. Since the tessellation algorithms create Delaunay triangulations, we also further modified the tessellations by randomly flipping approximately 20% of the edges. While flipping, we ensured that the boundary is preserved and edges are not flipped if the flip would introduce inverted triangles. We tested the algorithm on 1000 random meshes, each with about 50 vertices sampled on a square and a disc. For all cases, we observed that each iteration produces a valid embedding together with an accompanying perfect Laplacian. Moreover, the vertices reached steady state and converged towards the input mesh, achieving good approximations after 5–10 iterations. This indicates that for all these cases the randomly generated meshes were regular, admitting a perfect Laplace operator. Figures 1(a) and (b) show two typical examples.

A more challenging test case is given by meshes that do not admit a perfect Laplacian. To this end, we created five manual examples with 8 to 15 vertices, such as the shadow of the Schönhardt polyhedron shown in Figure 2. The algorithm handles these cases as well; however, necessarily by changing some edge lengths and thereby moving the vertices (see Section 3.3 for details).

While we observe that our approach reaches steady state

[‡] We avoid the notion of *weighted Delaunay* here because it is limited to triangulations with a single boundary, see for example [dGAOD13].

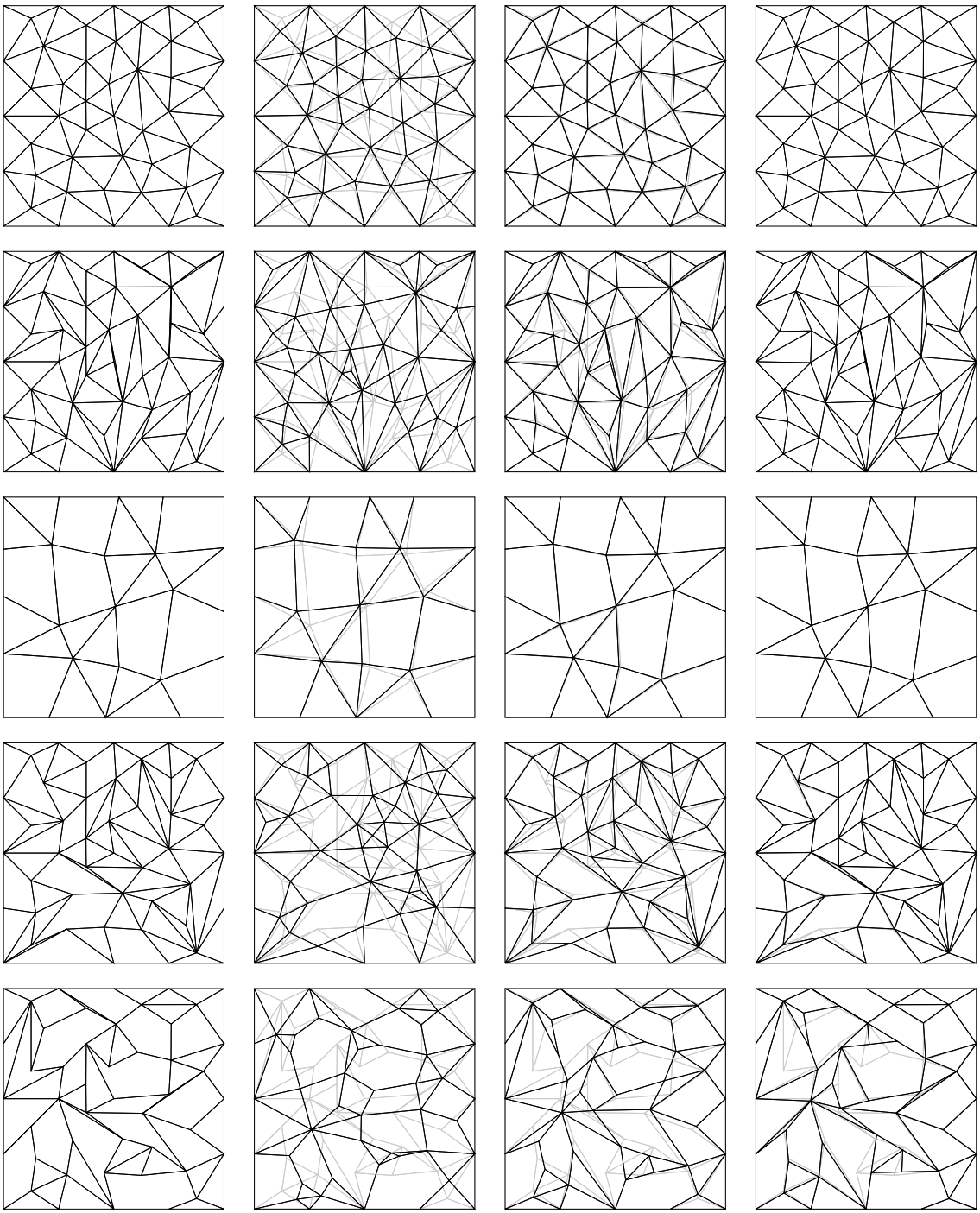

Figure 1: *Results created by our approach for different classes of polygon meshes. Each row depicts iterations of our algorithm starting with an input tesselation.*
(a) Delaunay triangulation: Original, 1, 5, and 20 iterations.
(b) Regular triangulation that is not Delaunay: Original, 1, 5, and 20 iterations.
(c) Polygonal mesh with convex faces: Original, 1, 5, and 20 iterations.
(d) Polygonal mesh with non-convex faces: Original, 1, 5, and 50 iterations.
(e) Polygonal mesh with non-convex faces: Original, 1, 5, and 50 iterations.

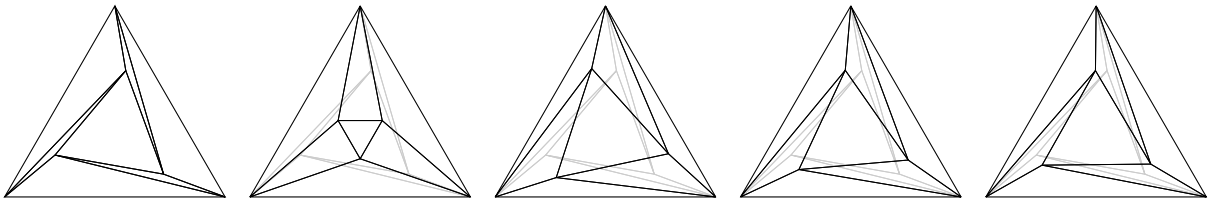

Figure 2: *The shadow of the Schönhardt polyhedron, a well-known examples for triangular mesh not admiting a perfect Laplacian. From left to right: Original, 1, 5, 10, 50 iterations of our algorithm.*

in all cases, the obtained solutions are not unique. Choosing different initial coefficients typically leads to different results. In particular, for Delaunay meshes with convex boundary, our approach does not recover the cotan-Laplacian (which is perfect in this case).

Polygon meshes In a second series of tests we considered planar polygonal meshes. Proceeding as in the previous section, we created random triangle meshes but removed some of the edges. As for triangle meshes, we observed convergence for all 1000 random polygonal meshes tested. The algorithm enforces convex faces in all iterations even in case of non-convex input meshes. Two examples produced by our approach are shown in Figures 1(c), (d) and (e).

3D meshes For meshes embedded in 3-space the algorithm performs similar to the 2D case. Figure 3 illustrates the convergence for a polygonal input mesh. We tested the algorithm on a number of standard meshes and added artificial noise to assess robustness. For all examples the algorithm converged and produced a valid perfect Laplacian. The next section details applications involving meshes in 3-space.

6. Applications

Self-supporting surfaces Several recent works investigate self-supporting surfaces in equilibrium [dGAOD13, PBSH13, LPS*13]. The central idea is that a network of rigid struts can withstand its own dead load if it is in force equilibrium at the points where struts meet (vertices), except for supported vertices where force can be dissipated.

Denoting the gravitational force acting on vertex $\mathbf{x}_i$ along the z-direction by F_i and the force along the edge $\mathbf{e}_{ij}$ by f_{ij} the condition for structural stability at vertex i becomes

$$\sum_{(i,j)\in E} f_{ij}\frac{\mathbf{x}_j-\mathbf{x}_i}{||\mathbf{x}_j-\mathbf{x}_i||} = \begin{pmatrix}0\\0\\-F_i\end{pmatrix}, \quad \text{with } f_{ij}\geq 0. \qquad (12)$$

All forces have to be compressive as indicated by the positivity of the forces. Considering $\omega_{ij} = f_{ij}/||\mathbf{x}_j-\mathbf{x}_i||$ we are faced with the problem of computing a perfect Laplacian for a polygon mesh embedded in 3-space. Hence, we can apply our algorithm with appropriate boundary conditions

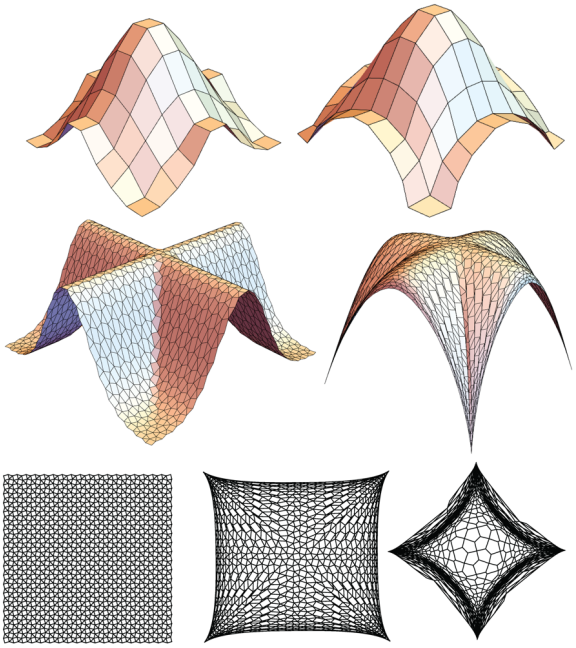

Figure 4: *Fitting of self-supporting surfaces to several polygonal meshes. The last row shows the planar projection of the input mesh above, the projection of the self-supporting surface and its dual.*

for supported vertices. In contrast to recent work, our algorithm does not rely on interior-point solvers and can be easily implemented using a sparse linear solver. Moreover, our system handles polygon meshes naturally. The results of our algorithm match results of existing methods. Self-supporting surfaces for several input meshes and comparisons to state of the art methods are shown in Figures 4 and 5, respectively.

In contrast to related work, we do not modify the combinatorics of the mesh which limits the direct applicability of our approach. However, our algorithm can serve as a central building block in a more sophisticated algorithm, driving a remeshing step based on the force distribution or in situations where a certain frame topology needs to be fixed.

Parameterization A classical application of discrete Laplacians is mesh parameterization. Given a fixed convex boundary in the plane and a discrete Laplacian of a polygon mesh

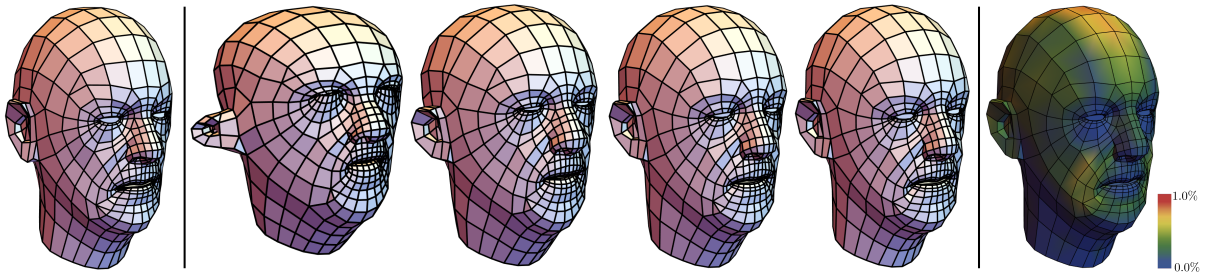

Figure 3: *Given a polygonal input mesh (left), our algorithm is applied. Iteration 1,2,5 and 20 are shown (center). The converged result is shown on the right. The color coding illustrates the vertex displacement from input mesh to the result. The maximal vertex displacement is about* 0.85% *of the bounding box diagonal.*

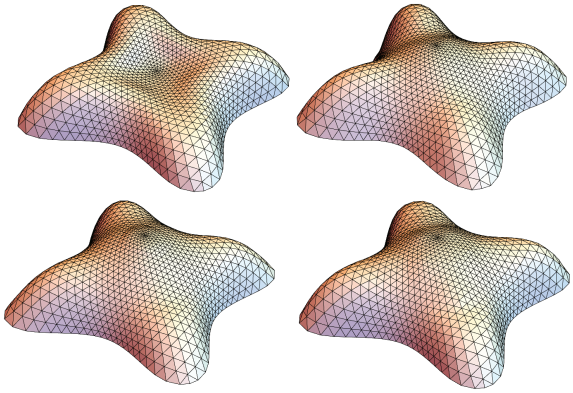

Figure 5: *Our algorithm produces self-supporting surfaces comparable to state of the art methods. The input mesh (top, left) is deformed towards a self-supporting surface by our method (top, right), the method by Vouga et al. [VHWP12] (bottom, left), and using the technique of de Goes et al. [dGAOD13].*

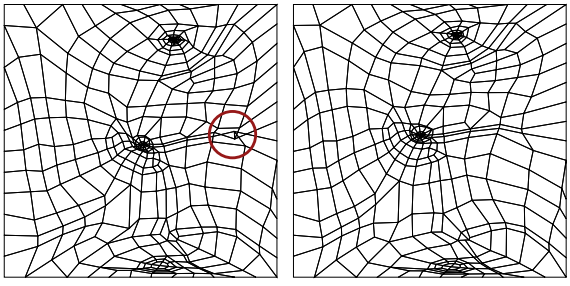

Figure 6: *Parameterizing using the Laplace operator of Alexa et al. [AW11] can introduce flipped faces (left, flipped faces marked red). Our operator ensures a flip-free embedding.*

embedded in $\mathbb{R}^3$ we can compute an embedding into $\mathbb{R}^2$ by solving Eq. (4). Using operators with negative coefficients can result in face flips, i.e., the embedding is not injective. For applications like texture mapping this behavior in not acceptable. With our algorithm a flip-free embedding is always guaranteed. This holds for the triangle as well as for the polygonal case. To our knowledge this is the first algorithm to introduce a geometry aware flip-free parameterization technique for polygon meshes.

7. Discussion

Our approach provides a perfect Laplace operator for any input mesh, albeit at the cost of potentially moving the vertices. In the following, we briefly discuss the consequences, particularly in light of related approaches.

7.1. Modifying combinatorics

If triangle meshes with convex boundary are Delaunay, all triangle angles are smaller than $\pi/2$ and the cotan-operator is non-negative. An interesting theorem due to Rippa [Rip90] states that general planar triangle meshes can be turned into Delaunay triangulations by incrementally flipping edges violating the Delaunay property. Bobenko and Springborn [BS07] construct perfect Laplacians for triangle meshes embedded in $\mathbb{R}^3$ by generalizing this theorem using intrinsic edge flips, i.e., they flip edges only for the generation of the Laplace operator—the operator may still be applied to the original mesh. Although these algorithm are guaranteed to terminate, it might take up to $|\mathcal{V}|^2$ edge flips [Ede01]. Perhaps more importantly, in some applications it might be important to fix combinatorics, such as for different vertex geometries that represent different instances of a mesh. In a sense both approaches are related: they apply an operator derived from different underlying combinatorics, while we may apply an operator derived from different geometry; so in both cases the Laplace operator was derived from a metric that is inconsistent with the input.

7.2. Modifying geometry

The work of Mullen et al. [MMdGD11] is similar in spirit to ours. Using non-linear optimization, they search for a weighted cotan-Laplacian exhibiting good numerical behavior and mostly non-negative weights. Optionally, vertex positions are optimized as well. Although non-negativity is not implied, the results are impressive and can significantly improve the accuracy of computations involving the discrete Laplacian. By contrast, our approach guarantees non-negativity and, perhaps more importantly, generalizes to polygonal meshes. We also think it is conceptually simpler and offers a more straight forward implementation.

7.3. Using diagonals

As mentioned earlier, Alexa and Wardetzky [AW11] implicitly consider all diagonals in a face. Interestingly, while this improves flexibility and makes it easier to generate non-negative Laplacians, their construction equally uses all diagonals, resulting in negative coefficients for almost all faces.

A downside of our approach is that using only the original edges, a perfect Laplacian requires convex faces. We could use all diagonals in our construction but at the expense of being able to guarantee an embedding in the planar case. This might be acceptable especially for the practical case of meshes in $\mathbb{R}^3$. In fact, we have experimented with using diagonals and the results look reasonable.

There is, however, a good reason for avoiding diagonals also in the case of meshes in $\mathbb{R}^3$: in a non-planar face, positive Laplace coefficients for the diagonals imply that the area gradient flow would reduce the area by folding the polygon rather than making it planar (as desired). This suggests that the choice of a Laplace operator may be application dependent.

Acknowledgment This work has been supported by the ERC through grant ERC- 2010-StG 259550 ("XSHAPE").

References

[Aur87] AURENHAMMER F.: A criterion for the affine equivalence of cell complexes in r^d and convex polyhedra in r^{d+1}. *Discrete & Computational Geometry 2*, 1 (1987), 49–64.

[AW11] ALEXA M., WARDETZKY M.: Discrete laplacians on general polygonal meshes. *ACM Trans. Graph. 30*, 4 (2011), 102:1–102:10.

[BBGO11] BRONSTEIN A. M., BRONSTEIN M. M., GUIBAS L. J., OVSJANIKOV M.: Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Trans. Graph. 30*, 1 (Feb. 2011), 1:1–1:20.

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

[BS08] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (Jan. 2008), 213–230.

[dC76] DO CARMO M.: *Differential geometry of curves and surfaces*. Prentice-Hall, 1976.

[dGAOD13] DE GOES F., ALLIEZ P., OWHADI H., DESBRUN M.: On the equilibrium of simplicial masonry structures. *ACM Trans. Graph. 32*, 4 (2013), 93:1–93:10.

[dGMMD14] DE GOES F., MEMARI P., MULLEN P., DESBRUN M.: Weighted triangulations for geometry processing. *ACM Trans. Graph. 33*, 3 (2014), 28:1–28:13.

[DMSB99] DESBRUN M., MEYER M., SCHRÖDER P., BARR A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of SIGGRAPH '99* (1999), ACM, pp. 317–324.

[Ede01] EDELSBRUNNER H.: *Geometry and Topology for Mesh Generation*. Cambridge University Press, 2001.

[Flo97] FLOATER M. S.: Parametrization and smooth approximation of surface triangulations. *Comput. Aided Geom. Des. 14*, 3 (1997), 231–250.

[Gli05] GLICKENSTEIN D.: Geometric triangulations and discrete laplacians on manifolds, 2005. arXiv:math/0508188.

[Gli07] GLICKENSTEIN D.: A monotonicity property for weighted delaunay triangulations. *Discrete & Computational Geometry 38*, 4 (2007), 651–664.

[JR13] JAUME R., ROTE G.: Recursively-regular subdivisions and applications, 2013. arXiv:1310.4372.

[LKL02] LEE Y., KIM H. S., LEE S.: Mesh parameterization with a virtual boundary. *Computers & Graphics 26*, 5 (2002), 677–686.

[LPS*13] LIU Y., PAN H., SNYDER J., WANG W., GUO B.: Computing self-supporting surfaces by regular triangulation. *ACM Trans. Graph. 32*, 4 (2013), 92:1–92:10.

[Max64] MAXWELL J. C.: On reciprocal figures and diagrams of forces. *Philosophical Magazine 27* (1864), 250–261.

[MMdGD11] MULLEN P., MEMARI P., DE GOES F., DESBRUN M.: HOT: Hodge-optimized triangulations. *ACM Trans. Graph. 30*, 4 (2011), 103:1–103:12.

[PBSH13] PANOZZO D., BLOCK P., SORKINE-HORNUNG O.: Designing unreinforced masonry models. *ACM Trans. Graph. 32*, 4 (2013), 91:1–91:12.

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

[Rip90] RIPPA S.: Minimal roughness property of the delaunay triangulation. *Computer Aided Geometric Design 7*, 6 (1990), 489–497.

[She02] SHEWCHUK J. R.: Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications 22* (2002), 21–74.

[Sor06] SORKINE O.: Differential representations for mesh processing. *Computer Graphics Forum 25*, 4 (2006), 789–807.

[The15] THE CGAL PROJECT: *CGAL User and Reference Manual*, 4.6 ed. CGAL Editorial Board, 2015.

[Tut63] TUTTE W. T.: How to draw a graph. *Proceedings of the London Mathematical Society s3-13*, 1 (1963), 743–767.

[VHWP12] VOUGA E., HÖBINGER M., WALLNER J., POTTMANN H.: Design of self-supporting surfaces. *ACM Trans. Graph. 31*, 4 (2012), 87:1–87:11.

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

[ZVKD10] ZHANG H., VAN KAICK O., DYER R.: Spectral mesh processing. *Computer Graphics Forum 29*, 6 (2010), 1865–1894.