DOI: 10.1111/cgf.14790

COMPUTER GRAPHICS *forum*
Volume 42 (2023), number 6, e14790

ARAP Revisited Discretizing the Elastic Energy using Intrinsic Voronoi Cells

Ugo Finnendahl, Matthias Schwartz and Marc Alexa

TU Berlin, Berlin, Germany
finnendahl@tu-berlin.de, matthias.schwartz@campus.tu-berlin.de, marc.alexa@tu-berlin.de

Abstract

As-rigid-as-possible (ARAP) surface modelling is widely used for interactive deformation of triangle meshes. We show that ARAP can be interpreted as minimizing a discretization of an elastic energy based on non-conforming elements defined over dual orthogonal cells of the mesh. Using the intrinsic *Voronoi cells rather than an orthogonal dual of the extrinsic mesh guarantees that the energy is non-negative over each cell. We represent the intrinsic Delaunay edges extrinsically as polylines over the mesh, encoded in barycentric coordinates relative to the mesh vertices. This modification of the original ARAP energy, which we term* iARAP*, remedies problems stemming from non-Delaunay edges in the original approach. Unlike the spokes-and-rims version of the ARAP approach it is less susceptible to the triangulation of the surface. We provide examples of deformations generated with iARAP and contrast them with other versions of ARAP. We also discuss the properties of the Laplace-Beltrami operator implicitly introduced with the new discretization.*

Keywords: modelling, deformations, polygonal modelling

CCS Concepts: • Mathematics of computing → Mesh generation; Discrete optimization; • Computing methodologies → Mesh geometry models

1. Introduction

As-rigid-as-possible (ARAP) surface modelling [SA07] is a popular and practically relevant approach for interactive deformation of triangle meshes with significant impact on research in geometry processing [LZX*08, BKP*10, TCL*13, BDS*12]. The basic approach is to minimize a deformation energy subject to a set of constrained vertices. The deformation energy is governed by penalizing the deviation of a vertex star from transforming rigidly:

$$E_i(\mathbf{V}, \mathbf{R}_i) = \sum_{j \in \mathcal{N}_i} \frac{w_{ij}}{2} \|\mathbf{e}_{ij} - \mathbf{R}_i \hat{\mathbf{e}}_{ij}\|^2. \tag{1}$$

This energy is minimized over the set of vertex positions $\mathbf{V}$ and the per-vertex rigid transformations $\mathbf{R}_i$ simultaneously. Deviation from rigidity is measured by comparing the rotated original edge vectors $\hat{\mathbf{e}}_{ij}$ to the edge vectors $\mathbf{e}_{ij}$ in the deformed mesh among the edges incident on a vertex. The weights w_{ij} control the influence of each edge. Sorkine and Alexa suggest to use the *cotan* weights known from discrete Laplace operators for triangle meshes [PP93, MDSB03]. Their argument is heuristic and based on the observation that the choice of diagonals in a rectangular grid should not affect the deformation (cf. Figure 1). The fact that the weights are negative for non-Delaunay edges is a well known problem, because it may cause the per-vertex energy term to become negative.

This may explain why it has become more common to use a modified version of the ARAP energy, following a proper discretization of the continuous deformation energy suggested by Chao et al. [CPSS10]. The idea of this energy, similar to ARAP, is to penalize the deviation of the gradient (coordinate wise, i.e. Jacobian) of a deformation mapping $\mathbf{f} : \mathbb{R}^3 \mapsto \mathbb{R}^3$ from a rigid transformation. The smooth version of this energy can be evaluated exactly in closed form for a linear deformation function and constant rigid transformation over a triangle [PP93]:

$$E_t = \frac{1}{2} \int_t \|d\mathbf{f} - \mathbf{R}\|^2 = \frac{1}{4} \sum_{(i,j) \in t} \cot \alpha_{ij}^t \|\mathbf{e}_{ij} - \mathbf{R}\hat{\mathbf{e}}_{ij}\|^2. \tag{2}$$

Here, α_{ij}^t is the interior angle in triangle t opposite edge (i, j), as in the definition of the *cotan* Laplace operator.

Modifying the ARAP energy to sum over the triangles incident on vertex i leads to the so-called *spokes-and-rims* version (Figure 2)

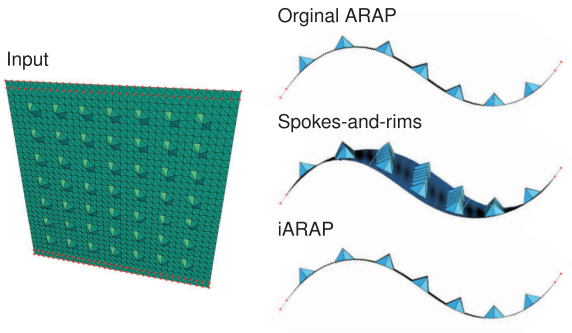

Figure 1: *Deformation of a mesh [SA07, LSHXY22]. The constrained vertices (coloured in red) are displaced orthogonal to the plane. The results exhibit the asymmetry in the energy of the spokes-and-rims approach. As the initial mesh is Delaunay the iARAP approach coincides with the original ARAP approach.*

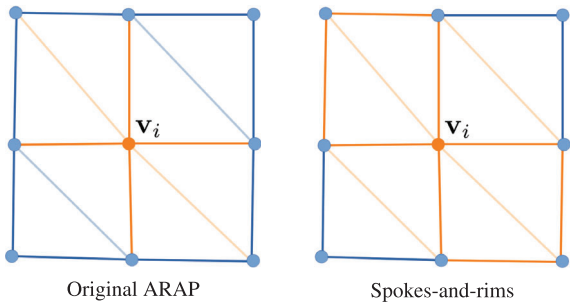

Figure 2: *Edges contributing to the ARAP energy associated with vertex i coloured in orange. Line thickness indicates magnitude of the* cotan *weights – thin lines have weights close to 0.*

where in addition to the edges incident on vertex i also the edges in the link (i.e. the boundary of the vertex star) contribute. While the *cotan* weights still may be negative for non-Delaunay edges, the energy exactly expresses the non-negative contribution of each triangle and cannot become negative in any part.

The addition of the *rim* edges introduces a potentially detrimental asymmetry in the cells that are governed by a common rigid transformation (see Figure 1). In Section 3, we interpret the different versions of ARAP as different discretizations of the elastic deformation energy by Chao et al. [CPSS10]. This shows that the spokes-and-rims version of ARAP is based on the mesh triangles, while the original ARAP may be considered using orthogonal dual cells. If the mesh is Delaunay the orthogonal dual cells are intrinsic Voronoi regions and this works well. Non-Delaunay edges, however, lead to dual cells that are not properly immersed. We ensure valid dual cells by considering the *intrinsic* Delaunay triangulation, whose dual is the intrinsic Voronoi diagram and the dual cells are properly immersed. This construction leads to the *iARAP* energy. The necessary computation of local rotations requires that we need to make the intrinsic triangulation available extrinsically. In Section 4, we explain how to represent the intrinsic Delaunay edges as polylines living on the original triangle mesh using barycentric coordinates.

The modified ARAP discretization differs from the original version in that it is guaranteed to be non-negative for each vertex star. This remedies the artefacts observed in the original ARAP version. It behaves similar to the spokes-and-rims variant, yet avoids unwanted asymmetries for meshes whose discretization fails to reflect the symmetries. We provide visual results and quantitative data in Section 5.

A particular appeal of ARAP surface modelling is a straightforward block-descent scheme for minimizing the energy: Rotations are computed locally by non-linear optimization, vertex positions are computed globally by solving a linear system with fixed system matrix. The system matrix is a discrete Laplace operator, for both the original and spokes-and-rims version of ARAP. The matrix resulting from our approach is different. In other words, the original ARAP and the spokes-and-rims variant have the same left-hand side in the linear system, and the problems resulting from the possibly negative energy are resolved by modifying the right-hand side of the system by spokes-and-rims. In the iARAP approach, the right-hand side is similar in spirit to the original ARAP, but the left-hand side is modified. We analyse the properties of the left hand side as a discrete Laplace operator in the spirit of the discussion by Wardetzky et al. [WMKG07] in Section 6. As it turns out, the new construction allows trading symmetry for a maximum principle.

Lastly, we briefly discuss how our approach relates to other modifications of the energy, in particular those that address the dependence of bending on the discretization. We also provide an outlook on using the implicitly introduced Laplacian operator for other applications.

2. Background and Notation

We consider the immersion of a fixed triangle mesh $\widehat{\mathcal{M}}$ that is being deformed by a function $\mathbf{f} : \widehat{\mathcal{M}} \mapsto \mathbb{R}^3$. The immersion $\widehat{\mathcal{M}}$ is defined by the positions $\hat{\mathbf{v}}_i$ of the vertices $\mathcal{V}$ and the assumption that triangle $ijk \in \mathcal{T}$ is realized as the convex hull of its vertices.

The deformed mesh $\mathcal{M}$ is the result of applying $\mathbf{f}$ to the vertex positions, that is $\mathbf{v}_i = \mathbf{f}(\hat{\mathbf{v}}_i)$, and then realizing the triangles as the convex hull of their mapped vertices, identical to the definition of $\widehat{\mathcal{M}}$. Note that we explicitly avoid asking that $\mathbf{f}$ also maps the interiors of the triangles to the convex hull of the mapped vertices. This gives us the freedom to consider functions $\mathbf{f}$ that are not necessarily piecewise linear or continuous on the given mesh.

Given a realization of the mesh, edge vectors can be computed as $\mathbf{e}_{ij} = \mathbf{v}_j - \mathbf{v}_i$. This can be done for both, the original $\mathbf{e}_{ij}$ and the deformed version of the edge $\hat{\mathbf{e}}_{ij}$. The circumcentre of triangle ijk in the original mesh is $\hat{\mathbf{c}}_{ijk}$. The dual cell $\star i$ of vertex i is a polygon formed by the circumcentres of the triangles incident on i. See Figure 3 for an illustration. The dual cell is immersed if all primal edges ij are Delaunay; otherwise, it is self-intersecting, with dual edges $\star\hat{\mathbf{e}}_{ij}$ corresponding to non-Delaunay edges having opposite orientation.

Discrete laplace operators. It will be useful to recall the constructions of the *cotan* Laplace operator. There are two substantially different constructions leading to the same result. They have recently

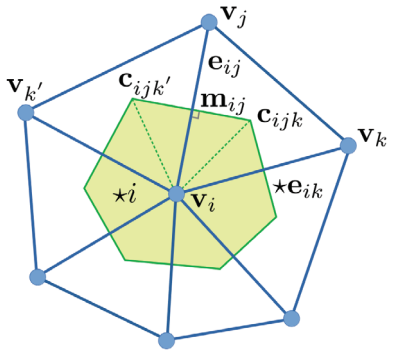

Figure 3: *Notation used for primal and dual elements of the mesh.*

been contrasted by Alexa el al. [AHKSH20] in the context of pointing out that the constructions differ for tetrahedralizations. The *primal* discretization is based on linear basis functions associated with the triangles of the mesh. We provide slightly more detail on the *dual* approach for context of our derivation of the ARAP energy; for more detail we ask the reader to consult the extensive literature on the subject and also see the derivation in Section 6.

Assuming the mesh consists of Delaunay edges, consider the dual cell $\star i$. The integrated Laplacian over $\star i$ turns into a sum over triangles spanned by vertex i and consecutive circumcentres, e.g. $\mathbf{v}_i, \mathbf{c}_{ijk}, \mathbf{c}_{ijk'}$. For each triangle we need to make an assumption of the function. The common approach is to consider only the (values in the) vertices along the primal edge, for example i and j. Other vertices, such as k and k', only contribute to the triangles associate with corresponding primal edges. This means, the function value assumed in $\mathbf{c}_{ijk}$ generally differs depending on the primal edge being considered. Therefore the underlying finite element space is *non-conforming*: The function is discontinuous along the edges of the triangles comprising the dual cell. In contrast, the finite element space for the primal construction is based on linear elements on triangles that agree on edges and is *conforming*. In general there is no reason for two different discretizations to agree and the fact that that the primal and dual constructions do lead to the same discrete operator for the Laplacian on triangle meshes is a coincidence. More generally, there are no inherent advantages or disadvantages of conforming or non-conforming finite element spaces and both are commonly used in different domains.

Intrinsic triangulations. The dual edge $\star \mathbf{e}_{ij}$ of non-Delaunay edges points in the "opposite" direction and leads to negative edge "lengths". This results in negative weights for the *cotan* Lapace-Beltrami operator and may even result in negative cell areas. This can cause problems when the *cotan* discretization is used [SA07, CWW17]. The intrinsic Delaunay triangulation of the mesh yields a principled solution to this problem. An intrinsic edge between two vertices on a polyhedron is a geodesic edge on the piecewise linear surface. In the realization of the surface it may be considered a polyline (compare Figure 7). Bobenko and Springborn [BS07] define the Laplace-Beltrami operator based on intrinsic Delaunay edges. They show that any polyhedron has a unique intrinsic Delaunay triangulation that can be constructed by intrinsically flipping non-Delaunay edges. Other constructions of the intrinsic Delaunay triangulation are possible [LFXH17] and they differ in their worst-case time complexity. In practice the computation is observed to be linear in the number of elements of the mesh for real-world models [SSC19, LFXH17]. For representation there exist several data structures [FSBS06, SSC19, GSC21] with support for different demands in applications.

3. Derivation of the ARAP Energies

We start from the elastic deformation energy introduced by Chao et al. [CPSS10]:

$$E(\mathbf{f}, \mathbf{R}) = \frac{1}{2} \int_t \|d\mathbf{f} - \mathbf{R}\|^2. \tag{3}$$

The main idea of ARAP is to restrict the rotations $\mathbf{R}$. It seems that all published variants of ARAP restrict $\mathbf{R}$ to be piecewise constant. The variants differ in the regions over which $\mathbf{R}$ is considered constant, and in the restrictions imposed on the deformation function $\mathbf{f}$.

As shown by Chao et al. [CPSS10], a necessary condition for minimizers of the energy in Equation (3) is

$$\Delta \mathbf{f} = \operatorname{div} \mathbf{R}, \tag{4}$$

which can be thought of as analogous to the situation for minimizers of Dirichlet energy having vanishing Laplacian. This is, in fact, the global step common to different versions of ARAP, and may be used as an explanation why all of them are based on the *cotan* Laplace-Beltrami operator on the LHS in the linear system.

The simplest version of ARAP results from using constant rotations per triangle. This choice has the drawback that bending across edges is not penalized, but it may be useful when other constraints are considered simultaneously [LG15, ZG18, LJ21]. In this case, $\mathbf{R}$ is computed in the local step based on the three vertices of a triangle, and divergence is computed in the usual way [PP03, Hir03], considering the columns of the rotation matrices as vector-valued functions for each of the three coordinates.

Explaining the original ARAP energy and the spokes-and-rims version in this framework requires making somewhat "unusual" choices. Both are based on fixing a rotation for each vertex and it seems natural that the boundary of this region intersects each edge at its midpoint

$$\mathbf{m}_{ij} = \frac{1}{2}\big(\mathbf{v}_i + \mathbf{v}_j\big), \tag{5}$$

which we define for both the original and the deformed geometry. In this way we have set $\mathbf{m}_{ij}$ to be the midpoint of the mapped vertices $\mathbf{v}_i, \mathbf{v}_j$, meaning we assume that $\mathbf{f}$ maps edge midpoints to edge midpoints.

With this assumption, we can interpret the rotations of spokes-and-rims ARAP as being defined over the polygon formed by the edge midpoints. While it was originally suggested to compute $\mathbf{R}_i$ from the vertex star of vertex i, using the polygon of edge midpoints is identical, except for an irrelevant scale factor. This means, we can interpret spokes-and-rims ARAP as considering the energy over only 3/4 of each triangle as depicted in Figure 4. Looking at derivations for spokes-and-rims ARAP, we see that divergence of

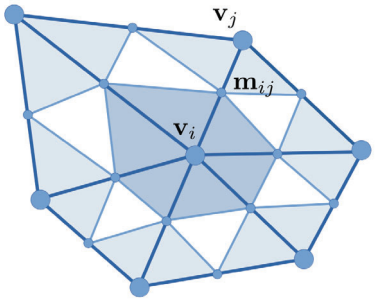

Figure 4: *In spokes-and-rims ARAP rotations are fitted per vertex star. The weighting can be interpreted as using only the dark-blue triangles for defining the rotation* $\mathbf{R}_i$*. This means, spokes-and-rims ARAP effectively integrates over* $3/4$ *of the surface.*

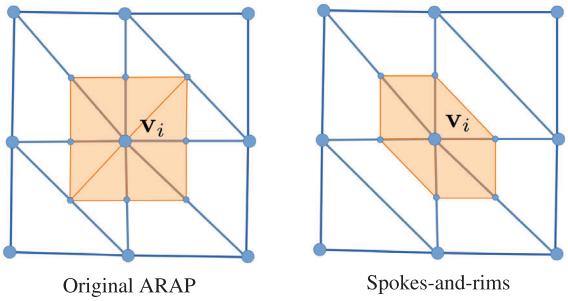

Figure 5: *Both the original and spokes-and-rims ARAP assume constant rotations per vertex. The orange indicates the different regions associated to the vertices. Note that the area in the original ARAP is independent of the chosen diagonal, whereas in spokes-and-rims ARAP the triangulation is reflected in the cells.*

the rotations in vertex i is computed by first averaging the three rotations over each triangle, and then computing the divergence from the triangle-wise quantities as usual.

Using the polygon of edge midpoints makes the discretization dependent on not only the positions of vertices, but also the connectivity of the mesh. A natural alternative, avoiding at least the dependence of the choice of edges, would be to use the Voronoi regions of the vertices. We depict the area considered for the two cases in Figure 5. Indeed, it is possible to interpret the original version of ARAP in this way. Note that the Voronoi cell around vertex i is spanned by a set of circumcentres, *if the mesh is Delaunay*. Our assumption will be that the deformation $\mathbf{f}$ may map these circumcentres freely and also differently depending on what triangle incident on the circumcentre we consider. In other words, we allow $\mathbf{f}$ to be discontinuous.

Consider the primal edge (i, j) and assume it has the Delaunay property. The part of the dual (Voronoi) cell $\star i$ corresponding to the edge is the triangle $\hat{\mathbf{v}}_i, \hat{\mathbf{c}}_{ijk}, \hat{\mathbf{c}}_{ijk'}$. This triangle is intrinsically flat, but in its realization it is comprised of two triangles, connected along the edge $\hat{\mathbf{v}}_i, \hat{\mathbf{m}}_{ij}$. Let us focus on the triangle $\hat{\mathbf{v}}_i, \hat{\mathbf{c}}_{ijk}, \hat{\mathbf{m}}_{ij}$. By our choice of $\mathbf{f}$, it will be mapped to the triangle $\mathbf{v}_i, \mathbf{c}_{ijk}, \mathbf{m}_{ij}$. Note that $\mathbf{c}_{ijk}$ is the result of mapping $\hat{\mathbf{c}}_{ijk}$ considering the sub-triangle incident on vertex i and edge (i, j). When considering the sub-triangle on the same edge but incident on vertex j the edge is considered in the opposite direction and the mapped circumcentre is $\mathbf{c}_{jik}$. Likewise, sub-triangles incident on edges (j, k) and (k, i) lead to four more circumcentres $\mathbf{c}_{jki}, \mathbf{c}_{kji}$ and $\mathbf{c}_{kij}, \mathbf{c}_{ikj}$. Because we allow f to be discontinuous, all six mappings may be chosen different. This degree of freedom can be exploited in order to minimize the deformation energy. This means, applying $\mathbf{f}$ to the piecewise linear surface $\widehat{\mathcal{M}}$ results in decomposing each triangle into six triangles, not necessarily connected across edges, illustrated in Figure 6. Note that although $\mathbf{f}$ is discontinuous our definition of $\mathcal{M}$ means that its triangles are flat and different from the original triangles mapped by f (except for the edges).

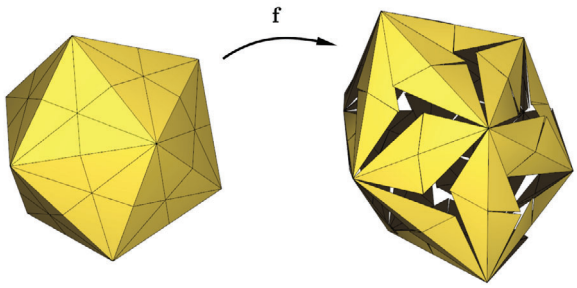

Figure 6: *ARAP [SA07] may be interpreted as the elastic deformation energy of a discontinuous deformation mapping* f*. The illustration shows how* f *would affect the input triangulation if it were applied the surface and not only the vertices of the mesh.*

Equation (2) describes the contribution of this smaller triangle to the deformation energy. Notice that edge $\mathbf{c}_{ijk} - \mathbf{v}_i$ is opposite to a right angle, so its contribution is zero. The edge $\hat{\mathbf{m}}_{ij} - \hat{\mathbf{c}}_{ijk}$ transforms to $\mathbf{m}_{ij} - \mathbf{c}_{ijk}$. For any choice of $\mathbf{R}_i$ minimizing the energy, we can set $\mathbf{c}_{ijk}$ so that $\mathbf{R}_i(\hat{\mathbf{m}}_{ij} - \hat{\mathbf{c}}_{ijk}) = \mathbf{m}_{ij} - \mathbf{c}_{ijk}$. In this way we have chosen f so that the contribution of this edge to the energy is minimal, namely zero.

This leaves only the transformed edge $\mathbf{v}_i - \mathbf{m}_{ij}$ contributing to the energy. This is true for both sub-triangles of $\hat{\mathbf{v}}_i, \hat{\mathbf{c}_{ijk}}, \hat{\mathbf{c}}_{ijk'}$. The overall contribution of this triangle to the energy then is:

$$\begin{aligned} E_{ij} &= \cot \angle \mathbf{m}_{ij}\mathbf{c}_{ijk}\mathbf{v}_i \left\| \frac{\mathbf{v}_i + \mathbf{v}_j}{2} - \mathbf{v}_i - \mathbf{R}_i\left(\frac{\hat{\mathbf{v}}_i + \hat{\mathbf{v}}_j}{2} - \hat{\mathbf{v}}_i\right)\right\|^2 \\ &= \frac{1}{4}\cot \alpha_{ij} \left\| \mathbf{e}_{ij} - \mathbf{R}_i \hat{\mathbf{e}}_{ij} \right\|^2. \end{aligned} \tag{6}$$

Here we have exploited the fact that the angles $\angle \mathbf{v}_i\mathbf{c}_{ijk}\mathbf{m}_{ij}$ and $\angle \mathbf{v}_i\mathbf{v}_k\mathbf{v}_j$ are identical ($\angle \mathbf{v}_i\mathbf{c}_{ijk}\mathbf{m}_{ij}$ is half the central angle $\angle \mathbf{v}_i\mathbf{c}_{ijk}\mathbf{v}_j$, which is twice the inscribed angle $\angle \mathbf{v}_i\mathbf{v}_k\mathbf{v}_j$). Considering also the triangle $\mathbf{v}_i, \mathbf{m}_{ij}, \mathbf{c}_{ijk'}$ we get the suggested choice for the weights w_{ij} in Equation 1.

Lastly, for computing divergence of $\mathbf{R}$ consistent with the original ARAP formulation we consider the *diamonds* $(\hat{\mathbf{v}}_i\hat{\mathbf{c}}_{ijk}\hat{\mathbf{v}}_j\hat{\mathbf{c}}_{ijk'})$ around vertex i (instead of the triangles). Each diamond consists of the surface spanned by two primal vertices incident on an edge and the corresponding two dual vertices (circumcentres). The rotation per diamond is computed by averaging the two rotations associated

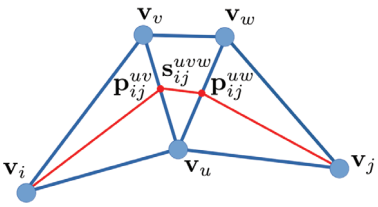

Figure 7: *Notation used for points along the intrinsic polyline. Two adjacent extrinsic triangles where the diagonal edge (u, v) is flipped in the intrinsic triangulation coloured in red.*

with the primal vertices. Then divergence is computed over the diamonds, similar to computing it over triangles [PP03].

Note that this derivation is based on the assumption that dual cells are *immersed* and that basing the realization on circumcentres yields Voronoi cells. If the mesh is not Delaunay, these assumptions are invalidated. Our idea to overcome this problem is to use the *intrinsic* Delaunay triangulation to define intrinsic Voronoi cells as the elements used for the discretization. This, in contrast to the approach of Chao et al. [CPSS10], keeps the spirit of computing divergence from the diamonds (albeit of the intrinsic Delaunay triangulation) for the RHS in Equation 4, but it changes the Laplace operator on the LHS.

4. Intrinsic ARAP Energy

The intrinsic Delaunay triangulation has been used mostly in a context where it is sufficient to compute the length of the edges [CWW17]. In order to measure the deformation of the surface we need to consider the realization of the intrinsic Delaunay edges. Note that each intrinsic Delaunay edge (i, j) is a polyline living on the piecewise linear surface. We denote the line segments of this polyline as $\mathbf{s}_{ij}^t$, where t denotes the triangle of the original mesh that carries the segment. In case the intrinsic edge coincides with an edge of the mesh the choice of t is not unique, but this has no consequences. The intersections $\mathbf{p}_{ij}^{uv}$ of the polyline with the extrinsic edge (u, v) are the start and end points of a segment:

$$\hat{\mathbf{s}}_{ij}^{uvw} = \hat{\mathbf{p}}_{ij}^{uw} - \hat{\mathbf{p}}_{ij}^{uv}. \tag{7}$$

For an illustration see Figure 7.

Note that this notation is not well defined, as a single intrinsic edge may intersect the same extrinsic edge multiple times [SSC19]. To avoid clutter we omit this complication in our presentation – it has no impact on the calculations.

We want that the energy depends on the vertex positions in the deformed state (and not the segment vectors). Note that we can express the intersections of an intrinsic edge with a mesh edge in terms of barycentric coordinates relative to the mesh vertices. A segment is the difference of two such barycentric coordinates, living on the same triangle. This means we can write for the segments *in the rest state*

$$\hat{\mathbf{s}}_{ij}^t = \hat{\mathbf{V}}\mathbf{b}_{ij}^t, \tag{8}$$

where $\hat{\mathbf{V}} = (\hat{\mathbf{v}}_0\ \hat{\mathbf{v}}_1\ \ldots)$ is the matrix that stores the vertex positions in the rest state column wise and $\mathbf{b}_{ij}^t$ is a sparse vector representing the segment in barycentric coordinates. The fact that $\mathbf{b}_{ij}^t$ encodes a vector (and not a point) means its elements sum to zero instead of one; and since both endpoints are on the same triangle it has at most three non-zero elements. If the segment coincides with a mesh edge it has only two non-zero elements. In particular, this is how original mesh edges are represented if they are Delaunay.

Similar to fixing the weights w_{ij} based on the rest state geometry we also fix the barycentric representations $\mathbf{b}_{ij}^t$. In this way we now simply replace the straight segment $\mathbf{e}_{ij}$ in the ARAP energy with the corresponding set of line segments $\{\mathbf{s}_{ij}^t\}$:

$$\begin{aligned} E_i(\mathbf{V}, \mathbf{R}_i) &= \sum_{j\in\mathcal{N}_i}\sum_{t\in\mathcal{T}_{ij}} \sigma_{ij}^t \frac{w_{ij}}{2} \|\mathbf{s}_{ij}^t - \mathbf{R}_i\hat{\mathbf{s}}_{ij}^t\|^2 \\ &= \sum_{j\in\mathcal{N}_i}\sum_{t\in\mathcal{T}_{ij}} \sigma_{ij}^t \frac{w_{ij}}{2} \|\mathbf{V}\mathbf{b}_{ij}^t - \mathbf{R}_i\hat{\mathbf{s}}_{ij}^t\|^2. \end{aligned} \tag{9}$$

The scalar factors σ_{ij}^t account for the fact that the intrinsic edge has been divided into smaller segments. We determine these factors by asking that the energy contribution of an edge is independent of its subdivision into segments (we drop subscripts referring to the edge to avoid clutter). Consider the intrinsic view, in which the edge is straight, we have $\mathbf{s}^t = \frac{\|\mathbf{s}^t\|}{\|\mathbf{e}\|}\mathbf{e}$ and note that $\sum_t \|\mathbf{s}^t\| = \|\mathbf{e}\|$. Equality of energy for the edge and the sum of the segments yields

$$\begin{aligned} \|\mathbf{e} - \mathbf{R}\hat{\mathbf{e}}\|^2 &= \sum_t \sigma^t \left\| \frac{\|\mathbf{s}^t\|}{\|\mathbf{e}\|}\mathbf{e} - \mathbf{R}\frac{\|\mathbf{s}^t\|}{\|\mathbf{e}\|}\hat{\mathbf{e}} \right\|^2 \\ &= \sum_t \sigma^t \frac{\|\mathbf{s}^t\|^2}{\|\mathbf{e}\|^2} \|\mathbf{e} - \mathbf{R}\hat{\mathbf{e}}\|^2. \end{aligned} \tag{10}$$

This suggests the choice

$$\sigma_{ij}^t = \frac{\|\mathbf{e}_{ij}\|}{\|\mathbf{s}_{ij}^t\|} \tag{11}$$

because this ensures

$$\sum_t \sigma^t \frac{\|\mathbf{s}^t\|^2}{\|\mathbf{e}\|^2} = \sum_t \frac{\|\mathbf{e}\|}{\|\mathbf{s}^t\|}\frac{\|\mathbf{s}^t\|^2}{\|\mathbf{e}\|^2} = \sum_t \frac{\|\mathbf{s}^t\|}{\|\mathbf{e}\|} = \frac{\sum_t \|\mathbf{s}^t\|}{\|\mathbf{e}\|} = 1, \tag{12}$$

as required. We further define $w_{ij}^t = \sigma_{ij}^t w_{ij}$.

4.1. Derivatives

For the optimization it is necessary to calculate derivatives. The gradient of the energy with respect to the vertex position $\mathbf{v}_k$ is

$$\begin{aligned} \nabla_{\mathbf{v}_k} E(\mathbf{V}, \{\mathbf{R}_i\}) &= \nabla_{\mathbf{v}_k} \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\sum_{t\in\mathcal{T}_{ij}} \frac{w_{ij}^t}{2} \|\mathbf{s}_{ij}^t - \mathbf{R}_i\hat{\mathbf{s}}_{ij}^t\|^2 \\ &= \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\sum_{t\in\mathcal{T}_{ij}} w_{ij}^t (\mathbf{s}_{ij}^t - \mathbf{R}_i\hat{\mathbf{s}}_{ij}^t) \nabla_{\mathbf{v}_k}\left(\mathbf{s}_{ij}^t\right) \\ &= \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\sum_{t\in\mathcal{T}_{ij}} w_{ij}^t (\mathbf{b}_{ij}^t)_k (\mathbf{s}_{ij}^t - \mathbf{R}_i\hat{\mathbf{s}}_{ij}^t), \end{aligned} \tag{13}$$

where $(\mathbf{b}_{ij}^t)_k$ is the k-th element of $\mathbf{b}_{ij}^t$, as $\mathbf{s}_{ij}^t = \mathbf{V}\mathbf{b}_{ij}^t$.

As $(\mathbf{b}^t_{ij})_k$ is only non-zero for segments within the star of $\mathbf{v}_k$ and each segment $\mathbf{s}^t_{ij}$ also appears negated in the energy, as segment $\mathbf{s}^t_{ji}$, with the same weight $w^t_{ij} = w^t_{ji}$, we can write the derivative as

$$\begin{aligned}\nabla_{\mathbf{v}_k} E(\mathbf{V}, \{\mathbf{R}_i\}) &= \sum_{\mathbf{s}^t_{ij} \in \mathcal{S}_k} w^t_{ij} (\mathbf{b}^t_{ij})_k (\mathbf{s}^t_{ij} - \mathbf{R}_i \hat{\mathbf{s}}^t_{ij}) \\ &= \frac{1}{2} \left(\sum_{\mathbf{s}^t_{ij} \in \mathcal{S}_k} w^t_{ij} (\mathbf{b}^t_{ij})_k (\mathbf{s}^t_{ij} - \mathbf{R}_i \hat{\mathbf{s}}^t_{ij}) \right. \\ &\quad \left. + \sum_{\mathbf{s}^t_{ji} \in \mathcal{S}_k} w^t_{ji} (\mathbf{b}^t_{ji})_k (\mathbf{s}^t_{ji} - \mathbf{R}_j \hat{\mathbf{s}}^t_{ji}) \right) \\ &= \sum_{\mathbf{s}^t_{ij} \in \mathcal{S}_k} w^t_{ij} (\mathbf{b}^t_{ij})_k \left(\mathbf{s}^t_{ij} - \left(\frac{\mathbf{R}_i + \mathbf{R}_j}{2} \right) \hat{\mathbf{s}}^t_{ij} \right),\end{aligned} \tag{14}$$

where $\mathcal{S}_k$ defines the set of directed segments within the extrinsic star of k, not the intrinsic.

4.2. Optimization

We can use the same block-descent optimization method as the original ARAP method [SA07]. The optimization is divided into a local step that optimizes $\{\mathbf{R}_i\}$ given $\mathbf{V}$ and a global linear step optimizing $\mathbf{V}$ given $\{\mathbf{R}_i\}$.

For given vertex positions, the local step consists of calculating optimal rotations $\mathbf{R}_i$ for each cell centred around $\tilde{v}_i$. Our approach differs from the original version only by the definition of the edges associated with cell of i. We arrive at the following covariance matrix:

$$\mathbf{S}_i = \sum_{j \in \mathcal{N}_i} \sum_{t \in \mathcal{T}_{ij}} w^t_{ij} (\mathbf{s}^t_{ij})^\mathsf{T} \hat{\mathbf{s}}^t_{ij}. \tag{15}$$

Optimal rotations $\mathbf{R}_i$ can be computed in different ways – we use the polar decomposition $\mathbf{S}_i = \mathbf{T}_i \mathbf{R}_i$.

The global step is directly given by setting the gradient w.r.t. vertex positions (Equation 14) to zero:

$$\begin{aligned}&\sum_{\mathbf{s}^t_{ij} \in \mathcal{S}_k} w^t_{ij} (\mathbf{b}^t_{ij})_k \left(\mathbf{s}^t_{ij} - \left(\frac{\mathbf{R}_i + \mathbf{R}_j}{2} \right) \hat{\mathbf{s}}^t_{ij} \right) = 0 \\ \Leftrightarrow &\sum_{\mathbf{s}^t_{ij} \in \mathcal{S}_k} w^t_{ij} (\mathbf{b}^t_{ij})_k \mathbf{s}^t_{ij} = \sum_{\mathbf{s}^t_{ij} \in \mathcal{S}_k} w^t_{ij} (\mathbf{b}^t_{ij})_k \left(\frac{\mathbf{R}_i + \mathbf{R}_j}{2} \right) \hat{\mathbf{s}}^t_{ij}.\end{aligned} \tag{16}$$

Note that this is the same linear system of equations for every dimension and the system matrix only depends on the initial mesh.

Recall that we can write $\mathbf{s}^t_{ij}$ as $\mathbf{V}\mathbf{b}^t_{ij}$. This suggests that the system matrix is

$$\mathbf{L} = \mathbf{B}\mathbf{D}_w\mathbf{B}^\mathsf{T}, \tag{17}$$

where $\mathbf{B}$ contains all the barycentric coordinate vectors $\mathbf{b}^t_{ij}$ of all segments and $\mathbf{D}_w$ is a diagonal matrix with the weights corresponding to the segments in $\mathbf{B}$. This representation is similar to the definition of the polygon Laplacian by Alexa and Wardetzky [AW11, Equation 3] for Delaunay meshes, as $\mathbf{B}$ reduces to the co-boundary operator in this case.

The matrix $\mathbf{L}$ is symmetric by construction and positive semi-definite because the weights in $\mathbf{D}_w$ are positive. So we can use the Cholesky factorization similar to the other ARAP methods.

Table 1: *Execution times for the precomputation (Cholesky factorization, plus intrinsic Delaunay triangulation for iARAP) and a single iteration (local rotations from extrinsic or intrinsic edges, plus solving) averaged over 100 runs for the different ARAP variants. All times are in milliseconds.*

Model	Bunny		Dachs		Arma	
Vertices	3K		20K		200K	
Times	Pre	Step	Pre	Step	Pre	Step
Original ARAP	55	3.2	336	23.5	943	72.7
Spokes-and-rims	46	3.2	280	22.7	825	75.9
iARAP	80	5.2	507	37.1	1405	95.5

Table 2: *Execution times for the local step in iARAP depending on the number of segments per intrinsic edge. All instances have the same number of vertices (35K) and edges (105K) but differ in the ratio of segments to edges. Times are in milliseconds.*

Segments/Edges	1.05	1.16	1.29
Execution Time	62.9	65.8	69.5

5. Results

We implemented the iARAP method using Eigen [GJ*10] and libigl [JP*18]. For the intrinsic Delaunay triangulation we use the signpost data structure [SSC19] implementation of geometry central[SC*19]. All results were computed in a single thread on an AMD Ryzen 7 2700X.

The iARAP discretization requires computing the intrinsic Delaunay triangulation in a preprocess, adding to the time required for precomputation (Table 1). After this, each step in the optimization is comparable to the original ARAP and spokes-and-rims ARAP. Only the calculations of the segments comprising an intrinsic edge are more involved. We store the precalculated sparse barycentre vectors $\mathbf{b}^t_{ij}$ and then the additional effort is a single sparse matrix-vector multiplication. The overhead grows with the number of intrinsic edge crossings as shown in Table 2. The resulting overhead is negligible in practice, see Table 1.

In Figure 8, we show the decay of the energy over the iterations. We observe that iARAP behaves similar to spokes-and-rims ARAP and slightly better than the original ARAP. Anecdotally, all methods seemed to offer the same degree of interactivity during modelling sessions.

The visual differences compared to the original ARAP on non-Delaunay meshes, as shown in Figure 9, are noticeable. Due to negative energy contributions the original ARAP exhibits creases, for example at the ears of the Bunny. In fat, the creases already appear

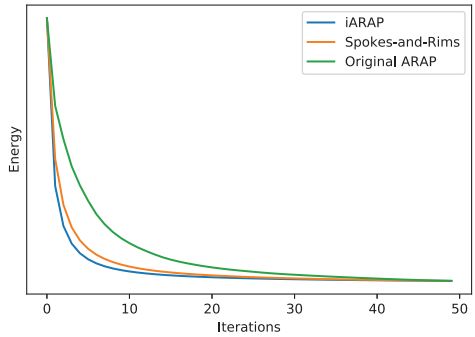

Figure 8: *Visualization of the energy decay per iteration on a mesh with $\approx 20K$ vertices. The three methods perform roughly similar, with the original ARAP method converging slightly slower.*

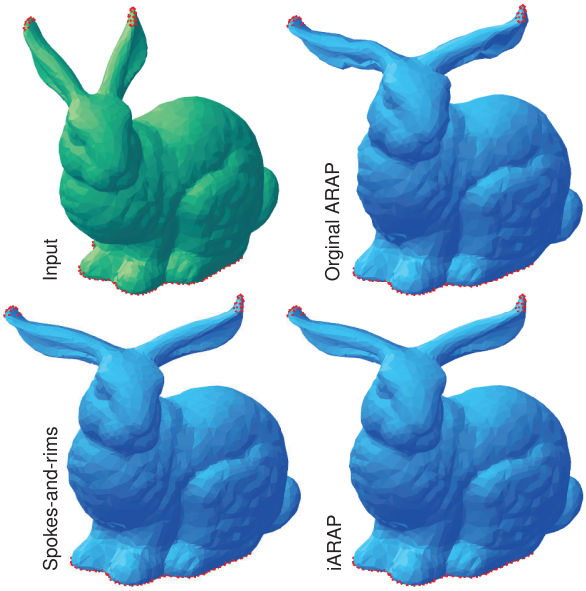

Figure 9: *Result for a non-Delaunay mesh. Constraints are coloured in red. The original BUNNY mesh has many ill-shaped triangles in the ears. This results in unwanted deformations for the original ARAP methods. Spokes-and-rims ARAP and iARAP generate similar and better results.*

without displacing the constraints. As the spokes-and-rims ARAP and iARAP energies are well defined, we regularly observe smooth and similarly looking deformed meshes, see Figure 10.

The difference between spokes-and-rims ARAP and iARAP clearly shows on meshes consisting of nearly regular quads that have been triangulated by inserting diagonals. Figure 11 shows the deformation of a cube with the top of the cube moved backwards. From the side all three methods look the same, which is why we only show one instance. In the front view we see asymmetries in the spokes-and-rims variant. This effect may also be observed on real world shapes, such as the balloon depicted in Figure 12.

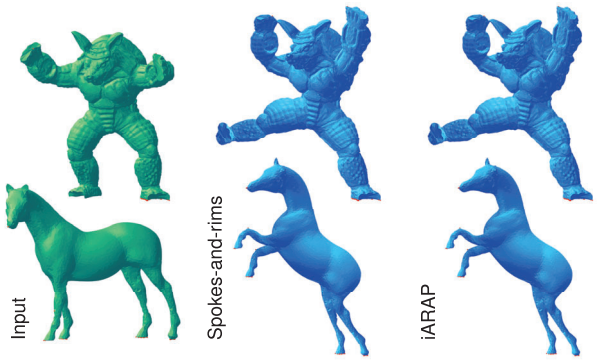

Figure 10: *The results of spokes-and-rims ARAP (mid) and iARAP (right) are often very similar for reasonably well triangulated surfaces.*

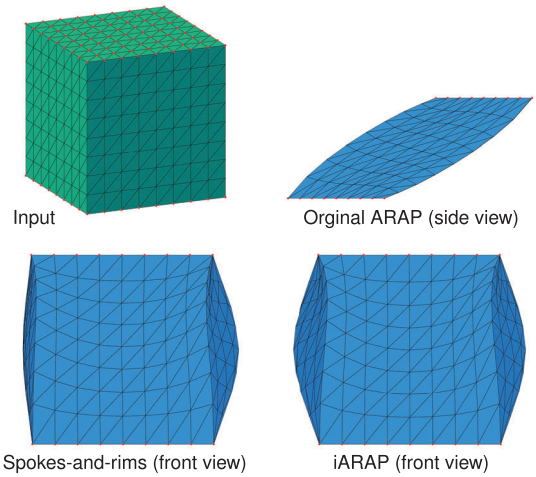

Figure 11: *Deformation of a subdivided cube mesh. The constrained vertices are shifted to the back. The results exhibit the asymmetry in the energy of the spokes-and-rims approach. As the input mesh is Delaunay the result of iARAP and original ARAP coincide.*

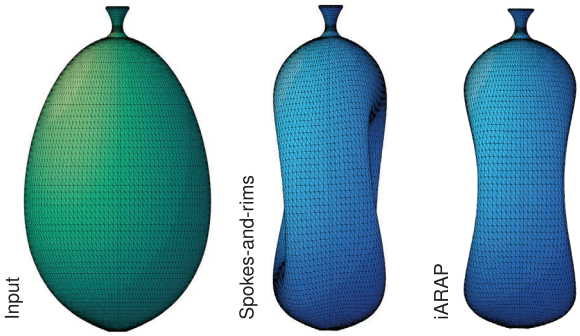

Figure 12: *A balloon mesh from Thingi10K [ZJ16] is squeezed. Spokes-and-rims ARAP rotates the entire mesh asymmetrically. Original ARAP is omitted as it behaves identical to iARAP on this mesh.*

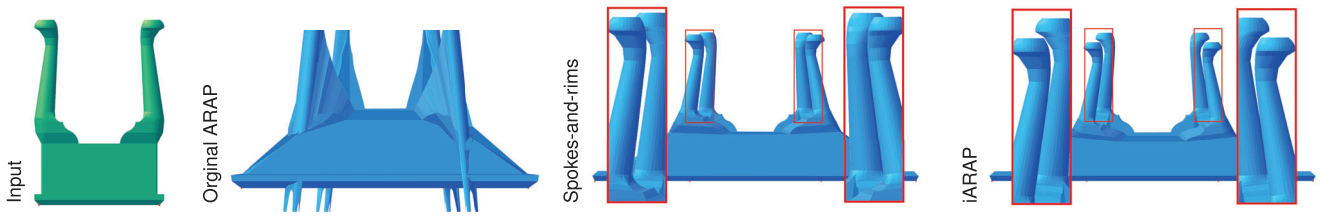

Figure 13: *Deformation of a stool from Thingi10K [ZJ16]. Notice the symmetric gap between the legs in the input. As the original ARAP energy diverges we stopped the optimization after a few steps (and the image shows only a subset of the resulting mesh). Spokes-and-rims ARAP and iARAP both converge, but only iARAP preserves the symmetry.*

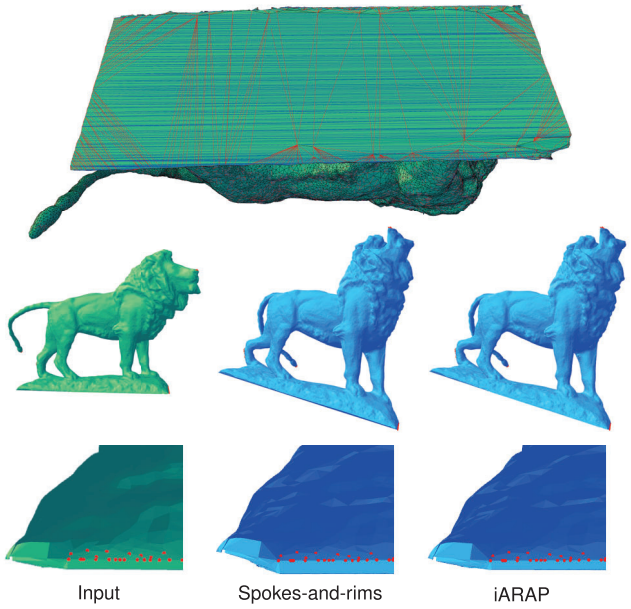

Figure 14: *On the base of the lion, intrinsic edges (red) cross a lot of extrinsic edges (blue). While the results of spokes-and-rims ARAP and iARAP are roughly similar, spokes-and-rims slightly bends the base.*

Asymmetries are also visible in symmetric shapes whose representation as a triangle mesh fails to represent the symmetries. The stool from thingi10K [ZJ16] has symmetric legs. Symmetric deformations should not remove this symmetry, yet only iARAP exhibits the desired behaviour (Figure 13).

Subtle differences can be observed in quite generally when the triangulation of the surface is far from being Delaunay or, in other words, if the intrinsic Delaunay edges intersect many extrinsic edges. The base of the lion in Figure 14 is triangulated with many non-Delaunay edges. While iARAP is oblivious to this triangulation, spokes-and-rims ARAP slightly bends the flat region, presumably due to numerical issues.

6. Laplace-Beltrami Operator

Both the original ARAP method and spokes-and-rims ARAP yield the *cotan* Laplace-Beltrami operator as the left hand side of the linear system of equations for the global optimization step. The iARAP discretization leads to a different matrix. We discuss this matrix interpreted as a Laplacian operator (Section 3) in terms of the list of properties introduced by Wardetzky et al. [WMKG07].

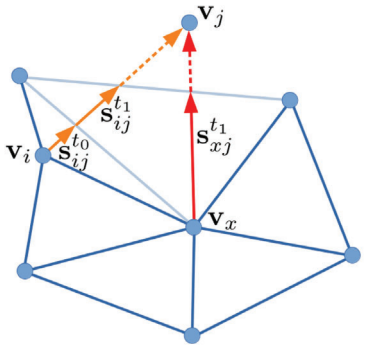

Figure 15: *Two cases of intrinsic edges intersecting a vertex star coloured in red and orange. Bold arrows follow intrinsic edges. Blue lines are extrinsic edges. Dashed lines are outside of the vertex star of $\mathbf{v}_x$. The solid red polyline starts at $\mathbf{v}_x$ and ends at the boundary of the vertex star of $\mathbf{v}_x$. The solid orange polyline starts and ends at the boundary of the vertex star.*

As mentioned before, Equation 17 shows that $\mathbf{L}$ is symmetric and positive semi-definite.

The operator is local in the sense that $\mathbf{L}_{ij} = 0$ if there is no edge between $\mathbf{v}_i$ and $\mathbf{v}_j$. This is the case because the derivative with respect to $\mathbf{v}_i$ only depends on the star of $\mathbf{v}_i$. On the other hand, the weights depend on the intrinsic Delaunay triangulation, which is global, so the operator is only weakly local in the sense that moving vertices may affect the weights of vertices that are far away [WMKG07, AHKSH20].

Linear precision is equivalent to $(\mathbf{Lv})_x = 0$ for each interior vertex $\mathbf{v}_x$ if all vertices are straight-line embedded into the plane. Given such a vertex $\mathbf{v}_x$ we prove linear precision by first grouping all segments $\mathbf{s}_{ij}^t \in \mathcal{N}_x$ in the star of vertex x according to their intrinsic edge $\mathbf{e}_{ij}$. It is clear that each group defines a straight polyline starting and ending at the boundary of the star or starting/ending at $\mathbf{v}_x$. The two cases are visualized in Figure 15. We represent a polyline as an ordered list of extrinsic points $(\mathbf{p}_0, \mathbf{p}_1, \ldots)$ using sparse barycentric

vectors (similar to segments, see Section 4):

$$\mathbf{p}_i = \mathbf{V}\mathbf{b}_i.$$

The contribution of a polyline $\mathbf{P}$ that starts and ends at the boundary of the star of $\mathbf{v}_x$ is

$$\sum_{\mathbf{s}^t_{ij}\in\mathbf{P}} w^t_{ij}(\mathbf{b}^t_{ij})_x\mathbf{s}^t_{ij} = \sum_{\mathbf{s}^t_{ij}\in\mathbf{P}} w^t_{ij}(\mathbf{b}^t_e - \mathbf{b}^t_s)_x\mathbf{s}^t_{ij}$$

$$= \sum_{\mathbf{s}^t_{ij}\in\mathbf{P}} (\mathbf{b}^t_e - \mathbf{b}^t_s)_x \frac{\| \star \mathbf{e}_{ij}\|_\pm}{\|\mathbf{s}^t_{ij}\|}\mathbf{s}^t_{ij},$$

where $\mathbf{p}^t_s$ is the start point and $\mathbf{p}^t_e$ the end point of the segment $\mathbf{s}^t_{ij}$. Since all points lie in a common plane, all segments of the polyline between i and j point in the same direction. Therefore

$$\frac{\| \star \mathbf{e}_{ij}\|_\pm}{\|\mathbf{s}^t_{ij}\|}\mathbf{s}^t_{ij}$$

is the same for all segments and equals

$$\frac{\| \star \mathbf{e}_{ij}\|_\pm}{\|(\mathbf{v}_j - \mathbf{v}_i)\|}(\mathbf{v}_j - \mathbf{v}_i).$$

This simplifies the contribution since successive segments share a point:

$$(\mathbf{b}^{t_n}_e - \mathbf{b}^{t_0}_s)_x \frac{\| \star \mathbf{e}_{ij}\|_\pm}{\|(\mathbf{v}_j - \mathbf{v}_i)\|}(\mathbf{v}_j - \mathbf{v}_i),$$

where $p^{t_0}_s$ is the start and $p^{t_n}_e$ is the end vertex of the polyline. As both are on the boundary of the star of x this contribution is 0 as $(b_i)_x = 0$ for all points i on the boundary of the 1-ring.

For the second case we only consider polylines $\mathbf{P} \in \mathcal{P}_x$ that start at $\mathbf{v}_x$, as the segments are directed and the opposite direction contributes the same way:

$$\sum_{\mathbf{P}\in\mathcal{P}_i}\sum_{\mathbf{s}^t_{ij}\in\mathbf{P}} (\mathbf{b}^t_{ij})_i w^t_{ij}\mathbf{s}^t_{ij}.$$

Note that $x = i$ and that all polylines in $\mathcal{P}_i$ that start at i contain only one segment. This is clear as the end point of a segment in a triangle t that starts at $\mathbf{v}_i$ must be on the edge opposite to $\mathbf{v}_i$. We can again rewrite the term due to the linearity of the vertex positions and the fact that the polylines start at $\mathbf{v}_i$ as

$$\sum_{j\in\mathcal{N}_i} (\mathbf{b}^{t_n}_e - \mathbf{b}_i)_i \frac{\| \star \mathbf{e}_{ij}\|_\pm}{\|(\mathbf{v}_j - \mathbf{v}_i)\|}(\mathbf{v}_j - \mathbf{v}_i),$$

where $\mathcal{N}_i$ describes the intrinsic Delaunay neighbourhood of i and $p^{t_n}_e$ is the end vertex of the polyline. As $p^{t_n}_e$ is on the boundary of the star of i by definition and $(\mathbf{b}_i)_i = 1$ we end up with:

$$\sum_{j\in\mathcal{N}_i} \frac{\| \star \mathbf{e}_{ij}\|_\pm}{\|(\mathbf{v}_j - \mathbf{v}_i)\|}(\mathbf{v}_j - \mathbf{v}_i).$$

In the planar embedding this is the same contribution as the intrinsic Laplace-Beltrami operator and according to Stokes' theorem this adds up to zero [AHKSH20]. Concluding, the iARAP operator yields the same result as the intrinsic *cotan* Laplace-Beltrami for all linear functions embedded in a plane, by weighting only the extrinsic neighbourhood.

Table 3: *Overview of the properties of the two Laplace-Beltrami operators. The operator* $\mathbf{L}$ *is the one used in iARAP,* $\mathbf{L}_+$ *is the asymmetric modification.*

		Loc				
Prop	Sym	WEAK	STRONG	Lin	Pos	Psd
$\mathbf{L}$	•	•	○	•	○	•
$\mathbf{L}_+$	○	•	○	•	•	•

From the results of Wardetzky et al. [WMKG07] it follows directly that the "positive" weight property cannot be satisfied or else we would have a contradiction with their main result. Note that the definition of the sign of the Laplacian varies in the literature. In our setup we would ask that all off-diagonal elements of $\mathbf{L}$ are negative. There is no reason why $\mathbf{B}\mathbf{B}^\mathsf{T}$ should not contain positive and negative weights, as $\mathbf{b}^t_{ij}$ contains negative and positive values.

An overview of the properties is shown in Table 3. Summarizing, the properties of the Laplacian implicitly defined by the intrinsic ARAP discretization of the elastic energy of Chao et al. [CPSS10] has properties that are identical to the *cotan* Laplacian. We want to stress, however, that the Laplacian corresponding to iARAP is generally *not* the *cotan* Laplacian. Experimentally we find that if both have (unwanted) negative off-diagonal entries, the iARAP discretization leads to smaller absolute values. This may help to alleviate unwanted effects of the wrong sign of these coefficients.

Alternative derivation. One may derive the operator implicitly defined by iARAP in a slightly more general way using Stokes's theorem:

$$\int_{\mathbf{C}} \Delta f = \int_{\partial\mathbf{C}} \mathbf{n}\cdot\nabla f. \tag{18}$$

Here, f is an arbitrary function, Δ the Laplace-Beltrami operator, ∇ the gradient operator, $\mathbf{C}$ a cell on the domain of f and $\mathbf{n}$ the outward pointing normal of the cell boundary $\delta\mathbf{C}$.

In a discrete setting the function f is represented as vector $\mathbf{f}$. If we define $\mathbf{C}$ to be the intrinsic Voronoi cell $\star i$, the dual edges $\star\mathbf{e}_{ij}$ resemble the boundary. By assuming a constant gradient along the dual edge $\star\mathbf{e}_{ij}$, we can approximate the integrated gradient in normal direction:

$$\int_{\star\mathbf{e}_{ij}} \mathbf{n}\cdot\nabla f \approx \| \star \mathbf{e}_{ij}\| \frac{(\mathbf{f}_j - \mathbf{f}_i)}{\|\mathbf{e}_{ij}\|}. \tag{19}$$

We arrive at the (integrated) intrinsic *cotan* Laplace-Beltrami operator:

$$\int_{\star i} \Delta f = \sum_{\star\mathbf{e}_{ij}\in\delta\star i} \int_{\star\mathbf{e}_{ij}} \mathbf{n}\cdot\nabla f \approx \sum_{\star\mathbf{e}_{ij}\in\mathcal{N}_i} \frac{\| \star \mathbf{e}_{ij}\|}{\|\mathbf{e}_{ij}\|}(\mathbf{f}_j - \mathbf{f}_i), \tag{20}$$

where $\mathcal{N}_i$ is the intrinsic neighbourhood of i.

In the case that i has only knowledge of the function values of its extrinsic neighbours this approximation is not possible anymore. If we still want to resemble the intrinsic Voronoi cell, we could approximate the intrinsic edges by tracing outwards. The intersections of the extrinsic link with the intrinsic Delaunay edges could be used

to approximate the function value of the intrinsic neighbour by linearly interpolating the two extrinsic neighbours on the link.

Using our notation for the barycentric coordinates of the traced edges we find the following term

$$\sum_{j \in \mathcal{N}_i} \frac{\| \star \mathbf{e}_{ij} \|_{\pm}}{\|(\mathbf{b}_p^j \mathbf{V} - \mathbf{v}_i)\|} (\mathbf{b}_p^j \mathbf{f} - \mathbf{f}_i),$$

where $\mathcal{N}_i$ is the intrinsic neighbourhood of i and $\mathbf{b}_p^j \mathbf{V}$ is the intersection of the intrinsic edge (i, j) and the boundary of the vertex star of i.

In fact this is exactly the iARAP Laplace-Beltrami formulation considering only polylines that start at i. If we set the contribution to all other polylines within the star of i to 0 we arrive at an extrinsic approximation of the intrinsic Delaunay Laplace-Beltrami operator derived from finite volumes. This new Laplace-Beltrami operator $\mathbf{L}_+$ has positive off-diagonal entries as all weights and interpolation coefficients are positive. Yet it is not symmetric: as the vertices used for the interpolation are not incident on the intrinsic edge (i, j) the influence is one-directional. If we want to overcome this asymmetry we need to consider those segments, as in the iARAP discretization.

Note that for symmetry it would be enough to only consider the first segment of an intrinsic edge, not all. That would lead to yet another operator defined by changing the iARAP energy to only depend on the first segment of the intrinsic edge.

7. Discussion

In this work, we show that the "missing" continuous picture of the original discrete ARAP energy can be found by discretizing the energy of Chao et. al. [CPSS10] over non-conforming elements of dual cells. This interpretation makes clear that the orthogonal dual cell associated with a vertex needs to be immersed. We ensure this by using intrinsic Voronoi cells. The corresponding Laplace-Beltrami operator approximates the intrinsic connectivity using the extrinsic connectivity. It has the same sparsity pattern (stencil) as the standard *cotan* Laplace-Beltrami operator (i.e. the adjacency matrix of the mesh) but approximates the intrinsic Laplacian. As the connectivity of the intrinsic Laplacian depends on the geometry and may change as the geometry changes, factorizations would have to be recalculated. THe iARAP Laplacian allows for a fixed symbolic factorization or sparse optimizations techniques that assume a fixed sparsity pattern [HTS*22] even after geometry changes.

Our general perspective on the different ARAP discretizations directly leads to further alternatives, which would be worthwhile to investigate in future work. Instead of making the map of the circumcentres part of the optimization, one may assume that they are mapped relative to the vertex positions (i.e. retain their barycentric coordinates). This would lead to another Laplace-Beltrami operator in case of using the extrinsic as well as intrinsic triangulation.

There are various other versions of ARAP modelling, defining the energy in slightly different ways to improve upon the original method. Problems of ARAP also arise from the rigid cells being defined per vertex. The energy penalizes stretching and bending differently because the rotations are defined over vertex stars. Chao et al. [CPSS10] found that the bending penalty decreases as the resolution grows and pointed out that the penalty arises from the vertex wise discretization. This problem can be solved by defining the rotations per triangle and then adding a second term that penalizes bending explicitly [ZG18, LG15]. Most of the methods that are based on ARAP or improving on it define energies that sum over edge sets. Those sets contain the edges of a vertex star or a single triangle, whether they define rotations vertex wise or triangle wise. We note that our approach based on the intrinsic Delaunay triangulation is orthogonal to these variations and could be incorporated by using the representation of an intrinsic edge as a sequence of extrinsic segments. It would be interesting to test whether such intrinsic versions of these methods further improve their results on poorly triangulated surfaces.

In a different direction, our derivation also motivates the investigation of other ways to restrict the rotations. Another way to address the above mentioned problems might be a richer function space for the rotations, lifting the restriction from piecewise constant functions. This would also require a proper discretizations of divergence.

Acknowledgements

The authors have nothing to report.

Open access funding enabled and organized by Projekt DEAL.

References

[AHKSH20] Alexa M., Herholz P., Kohlbrenner M., Sorkine-Hornung O.: Properties of laplace operators for tetrahedral meshes. *Computer Graphics Forum 39*, 5 (2020), 55–68. https://doi.org/10.1111/cgf.14068

[AW11] Alexa M., Wardetzky M.: Discrete laplacians on general polygonal meshes. In *ACM SIGGRAPH 2011 Papers* (New York, NY, USA, 2011), SIGGRAPH '11, Association for Computing Machinery. https://doi.org/10.1145/1964921.1964997

[BDS*12] Bouaziz S., Deuss M., Schwartzburg Y., Weise T., Pauly M.: Shape-up: Shaping discrete geometry with projections. *Computer Graphics Forum 31*, 5 (Aug 2012), 1657–1667. https://doi.org/10.1111/j.1467-8659.2012.03171.x

[BKP*10] Botsch M., Kobbelt L., Pauly M., Alliez P., Lévy B.: *Polygon Mesh Processing*. AK Peters/CRC Press, (2010). https://doi.org/10.1201/b10688

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

[CPSS10] Chao I., Pinkall U., Sanan P., Schröder P.: A simple geometric model for elastic deformations. In *ACM SIGGRAPH 2010 Papers*. SIGGRAPH '10, Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1833349.1778775

[CWW17] Crane K., Weischedel C., Wardetzky M.: The heat method for distance computation. *Communications of the ACM 60*, 11 (Oct 2017), 90–99. https://doi.org/10.1145/3131280

[FSBS06] Fisher M., Springborn B., Bobenko A. I., Schroder P.: An algorithm for the construction of intrinsic Delaunay triangulations with applications to digital geometry processing. In *ACM SIGGRAPH 2006 Courses*. SIGGRAPH '06, Association for Computing Machinery, New York, NY, USA (2006), pp. 69–74. https://doi.org/10.1145/1185657.1185668

[GJ*10] Guennebaud G., Jacob B., et al.: Eigen v3. http://eigen.tuxfamily.org, (2010).

[GSC21] Gillespie M., Sharp N., Crane K.: Integer coordinates for intrinsic geometry processing. *ACM Transactions of Graphics 40*, 6 (Dec 2021). https://doi.org/10.1145/3478513.3480522

[Hir03] Hirani A. N.: *Discrete Exterior Calculus*. PhD thesis, California Institute of Technology (2003).

[HTS*22] Herholz P., Tang X., Schneider T., Kamil S., Panozzo D., Sorkine-Hornung O.: Sparsity-specific code optimization using expression trees. *ACM Transactions of Graphics 41*, 5 (May 2022). https://doi.org/10.1145/3520484

[JP*18] Jacobson A., Panozzo D., et al.: libigl: A simple C++ geometry processing library, (2018). https://libigl.github.io/

[LFXH17] Liu Y.-J., Fan D., Xu C.-X., He Y.: Constructing intrinsic delaunay triangulations from the dual of geodesic voronoi diagrams. *ACM Transactions of Graphics 36*, 2 (Apr 2017). https://doi.org/10.1145/2999532

[LG15] Levi Z., Gotsman C.: Smooth rotation enhanced as-rigid-as-possible mesh animation. *IEEE Transactions on Visualization and Computer Graphics 21*, 2 (2015), 264–277. https://doi.org/10.1109/TVCG.2014.2359463

[LJ21] Liu H.-T. D., Jacobson A.: Normal-driven spherical shape analogies. *Computer Graphics Forum 40*, 5 (2021), 45–55. https://doi.org/10.1111/cgf.14356

[LSHXY22] Loriot S., Sorkine-Hornung O., Xu Y., Yaz I. O.: Triangulated surface mesh deformation. In *CGAL User and Reference Manual*, 5.4 ed. CGAL Editorial Board, (2022).

[LZX*08] Liu L., Zhang L., Xu Y., Gotsman C., Gortler S. J.: A local/global approach to mesh parameterization. *Computer Graphics Forum 27*, 5 (2008), 1495–1504. https://doi.org/10.1111/j.1467-8659.2008.01290.x

[MDSB03] Meyer M., Desbrun M., Schröder P., Barr A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*. Hege H.-C., Polthier K., (Eds.), Springer, Berlin, Heidelberg, Berlin, Heidelberg (2003), pp. 35–57.

[PP93] Pinkall U., Polthier K.: Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics 2*, 1 (1993), 15–36.

[PP03] Polthier K., Preuß E.: Identifying vector field singularities using a discrete hodge decomposition. In *Visualization and Mathematics III*. Hege H.-C., Polthier K., (Eds.), Springer, Berlin, Heidelberg, Berlin, Heidelberg (2003), pp. 113–134.

[SA07] Sorkine O., Alexa M.: As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*. SGP '07, Eurographics Association, Goslar, DEU (2007), pp. 109–116.

[SC*19] Sharp N., Crane K., et al.: geometry-central, (2019). https://geometry-central.net/

[SSC19] Sharp N., Soliman Y., Crane K.: Navigating intrinsic triangulations. *ACM Transactions of Graphics 38*, 4 (Jul 2019). https://doi.org/10.1145/3306346.3322979

[TCL*13] Tam G. K., Cheng Z.-Q., Lai Y.-K., Langbein F. C., Liu Y., Marshall D., Martin R. R., Sun X.-F., Rosin P. L.: Registration of 3d point clouds and meshes: A survey from rigid to nonrigid. *IEEE Transactions on Visualization and Computer Graphics 19*, 7 (2013), 1199–1217. https://doi.org/10.1109/TVCG.2012.310

[WMKG07] Wardetzky M., Mathur S., Kälberer F., Grinspun E.: Discrete laplace operators: No free lunch. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*. SGP '07, Eurographics Association, Goslar, DEU (2007), pp. 33–37.

[ZG18] Zhao H., Gortler S. J.: Shape deformation with a stretching and bending energy. In *Proceedings of the 31st International Conference on Computer Animation and Social Agents*. CASA 2018, Association for Computing Machinery, New York, NY, USA (2018), pp. 71–76. https://doi.org/10.1145/3205326.3205360

[ZJ16] Zhou Q., Jacobson A.: Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797* (2016).