Thesis Proposal

Intrinsic Triangulations in Geometry Processing

Nicholas Sharp
nsharp@cs.cmu.edu

Last updated: February 9, 2021

Committee
Keenan Crane, Chair (CMU)
Ioannis Gkioulekas (CMU)
Anupam Gupta (CMU)
Maks Ovsjanikov (École Polytechnique)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA

Contents

1 Introduction

Triangular surface meshes are the cornerstone data structure in 3D geometry processing, typically represented by a collection of faces with a position for each vertex. What if instead of positions, geometry is encoded by a length associated with each edge? This small change yields an *intrinsic triangulation*, which has just the right structure to enable powerful new algorithms for a variety of tasks. This thesis will study and advance the theory and practice of intrinsic triangulations, from their basic representation, to new algorithmic procedures, to applications in geometry processing.

Intrinsic triangulations bring together a multitude of concepts from topology and differential geometry in a discrete, computational setting. The name "intrinsic" itself arises from a core notion in differential geometry: many properties on a surface do not really depend on its embedding in space, but merely on the measurement of lengths and angles along the surface; such properties are termed *intrinsic*. The main utility of intrinsic triangulations comes from the ability to construct and manipulate triangulations which do not have any associated embedding in 3D space, yet still support a variety of useful computations. We will see how working in this more general space of triangulations enables simple algorithms for hard problems in surface geometry.

One important practical reason to study intrinsic triangulations is their benefits for *robust geometry processing*. With the ever-growing availability of 3D data, it has become painfully apparent that seemingly-effective geometry processing algorithms often fail on poor-quality inputs encountered in the wild; one common underlying cause is poor numerical conditioning of finite element problems. Intrinsic triangulations offer a much-needed remedy, with powerful retriangulation schemes to improve the numerical quality of a mesh. In fact, these solutions come with hard algorithmic guarantees which are available only in the intrinsic setting.

Despite the great potential of intrinsic triangulations, they have not yet seen significant uptake amongst practitioners, remaining largely confined to the mathematical geometry community. There are several reasons for this, from the abstractness of the formulation to the apparent difficulty of basic computational operations. One key goal of this thesis is to "fill the gaps" by developing the missing algorithms and data structures to make these tools useful in practice. Additionally, we describe many basic operations for the first time, and throughout release easily-usable software implementations.

Beyond gathering and unifying past research on intrinsic triangulations across mathematics and computer science, this thesis will present the following contributions to the subject:

- The *signpost data structure*, the first fully-informative representation for intrinsic triangulations which offers constant-time updates, and supports new operations [SSC19].
- A new algorithm for computing geodesics on surfaces via edge flips in an intrinsic triangulation, which can also generate constrained intrinsic triangulation [SC20b].
- A covering space, called the *tufted cover*, which generalizes intrinsic triangulations to nonmanifold meshes and even point clouds [SC20a].

Section 7 gives an in-depth summary of these contributions and proposed ongoing work.

2 Basic formulation

2.1 Surface geometry

Geometry processing develops algorithms for computing with surfaces, ranging from simple polyhedra, to mechanical parts used in computed-aided design (CAD), to the output of 3D scanning systems (inset). There are many representations for surfaces, but the most direct is the piecewise-flat triangle mesh, comprised of a collection of vertices in space connected by triangular faces. Note that triangle meshes are also sufficient to represent polyhedral surfaces with higher-degree planar polygonal faces, by arbitrarily triangulating each face.

Figure 1: Applied geometry processing develops algorithms to compute with surface meshes, such as 3D scans (*left*) and mechanical CAD models (*right*).

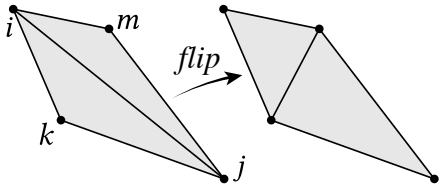
Formally, we define a triangular surface mesh $T := (M, p)$ by its connectivity $M = (V, E, F)$, a simplicial complex of vertices, edges, and triangular faces, and a position $p_i : V \to \mathbb{R}^3$ associated with each vertex. Each edge $ij \in E$ is then the unique straight line connecting two vertices, and each face $ijk \in F$ is a planar triangle defined by three vertices. Recall that restricting M to a simplicial complex means that the endpoints of each edge (and likewise vertices of each face) are always distinct—this will not necessarily be the case as we consider more general intrinsic triangulations.

2.2 Intrinsic triangulations

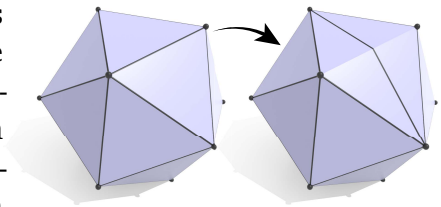
Much like the embedded triangle mesh, an intrinsic triangulation $\mathcal{T} := (M, l)$ is given by connectivity $M = (V, E, F)$, a cell complex of vertices, edges, and triangular faces, and geometry l. However, rather than vertex positions, the geometry is defined only by lengths $l_{ij} : E \to \mathbb{R}_{>0}$ associated with each edge ij. We require that l_{ij} satisfy the triangle inequality in each face; these edge lengths then determine a Euclidean metric on the interior of each triangle.

Topologically, we define the connectivity M to be a 2-manifold Δ-*complex* in the sense of Hatcher [Hat02, Section 2.1], allowing *e.g.* self-edges which connect a vertex to itself. Although polyhedral surfaces are most naturally modeled as a *simplicial* complex, formalizing intrinsic triangulations in the larger space Δ-complexes will be a necessary generalization for key results to hold. However, one must take care, as working in this more general space demands new nontrivial proofs even for seemingly intuitive properties—see *e.g.* Bobenko and Springborn [BS07] and Sharp and Crane [SC20b].

2.3 Edge flips



Intrinsic triangulations can be transformed via the *edge flip* operation, replacing some edge ij with the opposite diagonal km. In planar geometry, an edge flip would generally be implemented by measuring the distance between the endpoints k, m of the new edge (inset). However, one can observe that this operation is equivalently defined entirely from edge lengths, without any notion of vertex positions, by specifying the new edge length as the shortest path through the glued metric of the two triangles [Riv94]. Computationally, the new intrinsic edge length can be evaluated from the five already-known lengths in the diamond, via elementary geometry. This operation coincides with the ordinary planar notion of an edge flip for flat triangulations, but now generalizes directly to intrinsic triangulations which lack vertex positions, and may describe curved geometry. Traditionally, edge flips have been mainly used to achieve the Delaunay property (see Section 4.2), but in fact the edge flip is a general operation which can have other uses (*e.g.* Section 5.1).



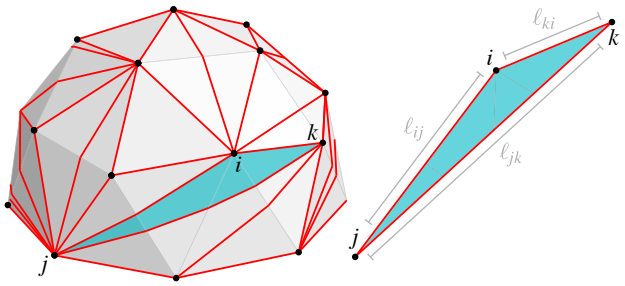
The remarkable property of the these intrinsic edge flips is that they exactly preserve Euclidean metric of the surface described by the triangulation. As concrete examples, quantities like surface area, distance along the surface, and Gaussian curvature—intrinsic geometric quantities—are invariant under the edge flip operation. An alternate perspective which may elucidate this property is to view intrinsic triangulations as sitting along the surface of an abstract manifold equipped with a *cone metric*; edges flips simply transform between different triangulations of the manifold. We thus have the freedom to flip edges to generate desirable intrinsic triangulations, without any cost of approximation error in the representation of the underlying surface. Section 4 explores how this operation allows us to robustly construct triangulations with excellent numerical conditioning, and Section 5 uses a similar mechanism to introduce special geodesic edges into a triangulation.

The edge flip operation induces the *flip graph* on the set of intrinsic triangulations, where two triangulations are connected if they are related by a single edge flip. Analyzing the correctness and complexity of edge flip-based algorithms will amount to proving various properties of the flip graph. As a basic example, it is known that the flip graph is connected: all possible intrinsic triangulations of a given surface are related by edge flips.

2.4 Intrinsic triangulations of embedded surfaces

In practice, we will often obtain intrinsic triangulations by starting with some mesh embedded in $\mathbb{R}^3$ (*e.g.*, a 3D model to analyze), reading off the edge lengths to form an intrinsic triangulation, then transforming the triangulation via edge flips. Because these flips preserve the metric, there is always an isometric bijection to the original surface, and the resulting triangulations can be drawn along the initial mesh with each edge as a geodesic. A key computational challenge is to develop data structures which encode not just the edge lengths of the intrinsic triangulation, but

furthermore encode the paths of these geodesic edge along the surface, and allow the bijection to be queried (Section 3).



From an algorithmic perspective, it may seem that this intrinsic triangulation is an abstract object, with little value for practical computation. Indeed, the intrinsic representation is by definition a less informative description of a surface, discarding geometric data like surface normals and dihedral angles. Furthermore, performing intrinsic edge flips implicitly constructs bent intrinsic triangles which lay across the surface, and whose edges are certainly not straight lines between vertices (inset). However, since the intrinsic metric is preserved, and thus intrinsic quantities like geodesic distances, tangent vectors, Laplacians, *etc.* remain perfectly well-defined, one can in fact evaluate many useful algorithms directly on an intrinsic triangulation, to great practical benefit (*e.g.* Section 4.3).

2.5 Historical roots

Intrinsic triangulations have their origins in the work of Regge [Reg61], as a representation for the study of general relativity. Modern mathematical study defined the notion of the intrinsic Delaunay triangulation [Riv94], and established that it can be constructed by edge flipping [Ind+01; BS07]. This construction leads to the intrinsic Delaunay Laplacian, a canonical Laplace matrix associated with any polyhedron which is a function only of its geometry, not its tessellation [BS07]. Additional research has also studied deep connections between intrinsic triangulations and discrete uniformization [Luo04], as well as circle packings and conformal parameterization [KSS06].

Prior to the work of this thesis, there has been relatively little study of practical aspects of intrinsic triangulations, such as appropriate data structures, robustness concerns, or integration with larger algorithmic pipelines in geometry processing. One notable exception is the work of Fisher et al. [Fis+07], which introduced a crossing-based data structure and demonstrated its use.

3 Representations

The most obvious representation, or data structure, for an intrinsic triangulation is a list of edge lengths l, along with some mesh data structure to encode the connectivity M. However, this representation leaves two open questions.

First, there are many mesh data structures, but because M may be a Δ-complex, not all will suffice (Section 3.1). Second, and more fundamentally, storing only M and l is inherently abstract. In the common case where our intrinsic triangulation is defined atop an embedded mesh, it does not allow for reconstructing the trajectory of an intrinsic edge along the embedded

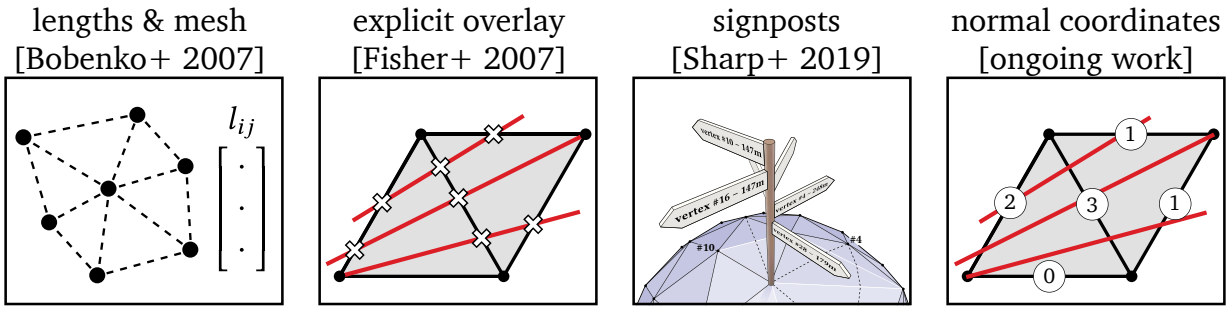


Figure 3: The representations for intrinsic triangulations considered in this thesis.

surface. More generally, the abstract representation does not support computations which involve tangent-valued data, or require querying the bijection between the intrinsic triangulation and the underlying mesh. Accordingly, we also consider richer representations which support a full range of operations on the intrinsic triangulation (Sections 3.2-3.4).

3.1 Representing connectivity

The connectivity among $M = (V, E, F)$, the vertices, edges, and faces in an intrinsic triangulation, can be represented by a mesh data structure; these structures have been widely studied in applied geometry (see *e.g.* [Bot+10]). However, while most traditional triangle meshes are formalized as a simplicial complex, intrinsic triangulations demand a Δ-complex which may contain repeated elements (Section 2.2). Using, *e.g.*, a naive listing of the vertex indices for each face will not suffice, between it is impossible to distinguish faces with identical vertices.

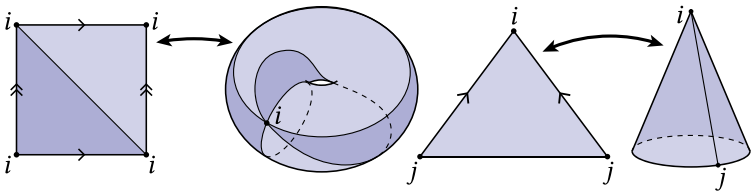


Figure 2: Two intrinsic triangulations which are Δ-complexes but not simplicial complexes.

Fortunately, many simple and standard mesh data structures represent general Δ-complexes without modification, such as edge-based winged-edge and halfedge structures [Bau75; Wei85; Ket99]. In fact, one can easily mimic these structures by augmenting a face-vertex list with an additional "gluing" array holding two values per face-side, as described in Sharp and Crane [SC20a, Section 4.1] in the context of intrinsic triangulations.

3.2 Explicit crossings

Even with a proper mesh data structure, additional data must be stored to encode how an intrinsic triangulation sits atop some initial embedded mesh (Section 2.4). The most direct approach is to explicitly store ordered lists of *crossings*, the locations where each intrinsic edge crosses an edge of the underlying mesh [Fis+07]. However, even just flipping an edge in this

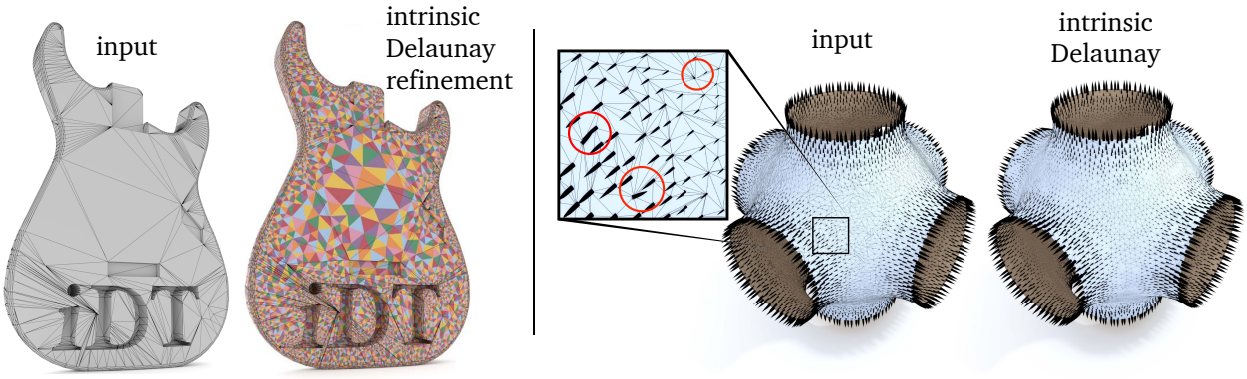


Figure 4: *Left*: The signpost data structure enables intrinsic Delaunay refinement for the first time, generating triangulations with good angle bounds. The black wireframe denotes the input mesh, while colored triangles give the intrinsic triangulation. *Right*: Signposts also enable vector field processing; the intrinsic Delaunay Laplacian offers a maximum principle for tangent vector fields, which here avoids unexpected flipped vectors when generating a smooth field.

representation is a nontrivial operation which must traverse and update the crossing graph, as opposed to the formulaic constant-time updates offered by other representations. Additional operations beyond edge-flips have not been described for this representation, although in principle they might be implemented with some traversal of the crossing graph.

3.3 Signposts

Rather than explicitly representing trajectory of each intrinsic edge, we can instead implicitly represent the trajectory by storing not just intrinsic edge lengths, but also the direction of each edge, as a tangent vector at the incident vertices $v_{ij} \in T_i M$ [SSC19]. This data is referred to as *signposts*, because it gives the direction and distance to walk along the original mesh to trace out the geodesic trajectory of each edge. This implicit representation retains the simple, constant-time edge flip updates from the lengths-only case.

Sharp, Soliman, and Crane [SSC19] define a wide variety of mesh-processing algorithms on the signpost representation of intrinsic triangulations, such as mesh refinement, repositioning vertices, and querying the bijection between the intrinsic triangulation and the underlying surface (Figure 4). This in turn enables higher-level retriangulation algorithms, like intrinsic Delaunay *refinement* and optimal Delaunay relaxation for the first time (Section 4).

The signpost representation implicitly encodes the connectivity of crossings via directions and lengths defined at vertices, recovering paths by tracing out geodesics along the surface. This encoding is discretely exact: in proper real arithmetic, it would always perfectly reconstruct the paths and the corresponding crossing connectivity. However, when implemented in inexact floating-point arithmetic, the recovery may fail in nearly-degenerate cases. Although this behavior is typical of geometry processing algorithms, it is particularly worrisome for intrinsic triangulations, for which a key application is robust computation on degenerate 3D models.

3.4 Normal coordinates

Normal coordinates have a long history in computational geometry and topology for representing curves through a complex [Kne29; Hak61; SSŠ02]; ongoing work in this thesis will investigate using normal coordinates to represent edge trajectories in intrinsic triangulations. The basic idea of normal coordinates is to store a count along each edge of a triangulation, encoding how many times a curve crosses that edge. These crossing counts are sufficient to recover the topological path of a curve on the surface—for Euclidean intrinsic triangulation edges which are geodesic by definition, curve geometry can then be recovered as the unique straight-line path through an unfolded sequence of triangles. Although recovering the geometry of paths along the surface will still ultimately require imperfect real-valued computation, normal coordinates offer important basic guarantees about properly representing connectivity, and will serve as a valuable tool for robust computing with intrinsic triangulations.

4 Intrinsic retriangulation

The intrinsic representation offers a large space of possible triangulations of a surface, all of which exactly encode the underlying intrinsic geometry. We can then design retriangulation algorithm which perform edge flips, as well as other operations like inserting new vertices, to construct triangulations with improved numerical conditioning or other desirable properties.

4.1 Comparison to traditional remeshing

Remeshing of surfaces meshes is widely studied in geometry processing [CDS12; CH11a; All+08], but such methods must inevitably trade off between element quality and geometric approximation of the input surface. The intrinsic approach escapes this tradeoff, operating in a space of triangulations which all exactly represent the underlying geometry. In fact, intrinsic algorithms offer concrete algorithmic guarantees about the quality of the resulting meshes, which generally are not otherwise available for surface remeshing rountines.

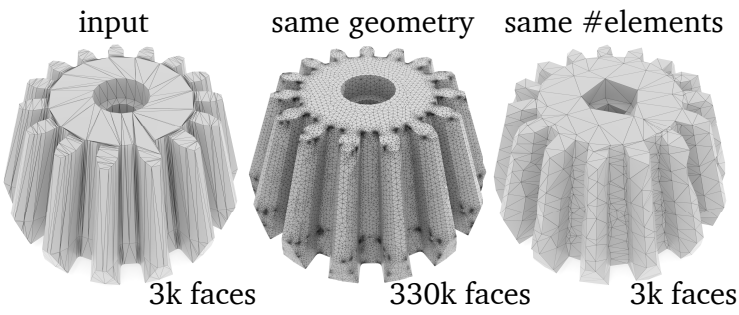


Figure 5: Traditional remeshing cannot improve element quality without increasing mesh size or disturbing the geometry; intrinsic triangulations escape this tradeoff.

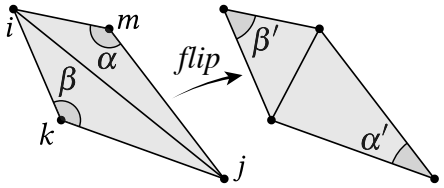
Of course, the price for this algorithmic power is that the output of intrinsic retriangulation is an intrinsic object with only edge lengths, not a traditional mesh with vertex positions. Fortunately, it is straightforward to adapt subsequent computations to this paradigm; generally one simply needs to evaluate geometric quantities from edge lengths, rather than vertex positions.

As one example, the GEOMETRY-CENTRAL C++ library contains a growing collection of routines which seamlessly support this paradigm [SC+19].

4.2 Delaunay flipping

In the plane, there are many equivalent characterizations of so-called Delaunay triangulations, such as the convex hull of a parabolic lifting, all triangles having empty circumcircles, or each edge having opposite corner angles $\alpha+\beta \leq \pi$. The last definition generalizes directly to intrinsic triangulations, computing corner angles via edge lengths, and defines the *intrinsic Delaunay triangulation* (IDT) [Riv94]. Just like the planar Delaunay triangulation associated with a point set, the intrinsic Delaunay triangulation is a fundamental discrete construction defined uniquely[1] for any polyhedron. Much like in planar geometry, the intrinsic Delaunay triangulation has many benefits for numerical computation; in particular it generates the *intrinsic Delaunay Laplacian*. This Laplace matrix has many computational benefits: it uniquely offers a discrete maximum principle, and the corresponding basis functions maximize the minimum angle in any triangle (see *e.g.* Shewchuk [She97]).

The foundational result in the study of intrinsic triangulations is that any triangulation can be transformed to be the intrinsic Delaunay triangulation by repeatedly flipping any edge with $\alpha + \beta > \pi$ (inseet) [Ind+01; BS07]. This mirrors Lawson's edge flipping algorithm in the plane [Law77], though significantly more sophisticated machinery is needed to prove correctness in the intrinsic setting.



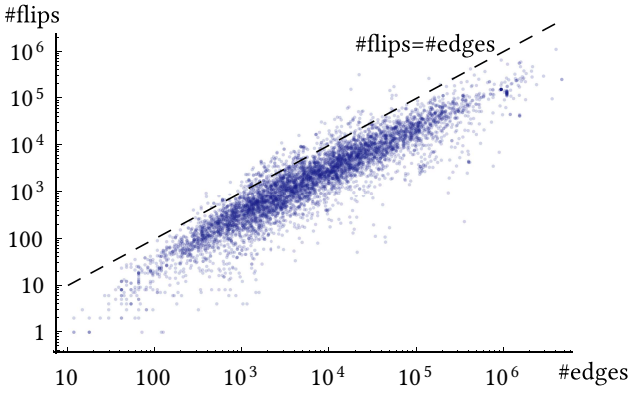


Figure 6: Number of edge flips to Delaunay on the Thingi10k dataset [ZJ16]; each point is a 3D model.

Delaunay edge flipping is then the most basic and essential intrinsic retriangulation scheme. Given any standard mesh with vertex positions, one can read off edge lengths to define an intrinsic triangulation, perform edge flips to generate the IDT, and construct the corresponding Laplace matrix for subsequent computation. Alternate approaches have constructed meshes satisfying the intrinsic Delaunay property as the dual of the geodesic voronoi diagram [Liu+17b], or by splitting edges [Liu+15], though these method are significantly more complicated and expensive than simply flipping edges. In terms of runtime, Delaunay edge flipping typically takes just milliseconds in practice for typical inputs, and an empirical study in Sharp, Soliman, and Crane [SSC19] (Figure 6) showed linear scaling on a challenging dataset of real-world models [ZJ16]. However, there is a wide gap between

[1] Up to general position. There exists an equivalence class induced by cocircular triangles with $\alpha + \beta = \pi$.

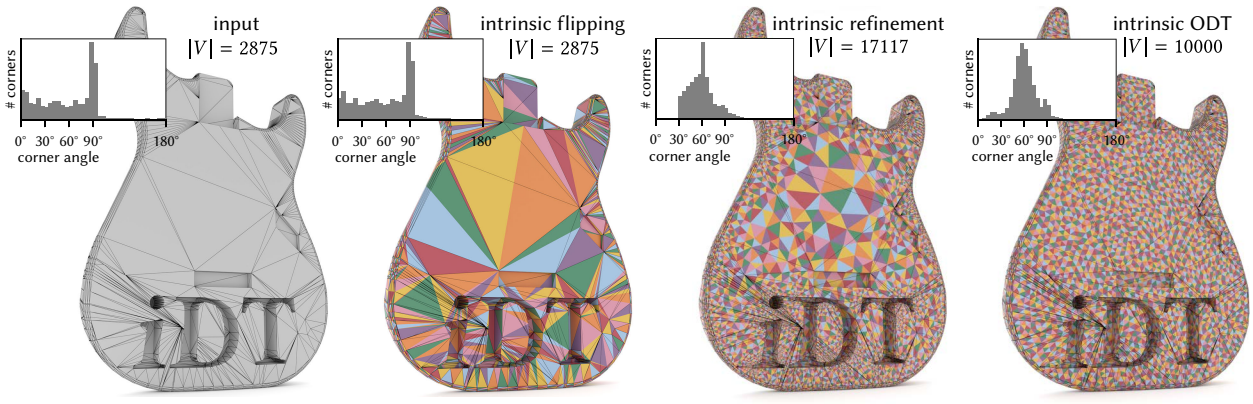


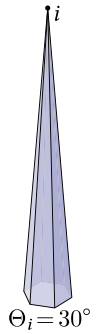
Figure 7: Intrinsic retriangulation schemes applied to a computed-aided design model with poor triangle quality. The black wireframe denotes the input mesh, colored triangles give the intrinsic triangulation. Delaunay flips achieve the Delaunay property for a fixed vertex set, while refinement and repositioning further improve triangle quality and vertex distribution.

this practical efficiency and asymptotic analysis: the only known runtime bound is exponential [Ind+01; BS07], although the most difficult known counterexample requires only $O(n^2)$ flips.

4.3 Delaunay refinement

In practice one often seeks triangulations which satisfy criteria beyond the Delaunay property, such as bounds on angles or edge lengths. *Delaunay refinement* progressively inserts vertices to achieve user-specified criteria, while maintaining the Delaunay property. Sharp, Soliman, and Crane [SSC19] describe *intrinsic* Delaunay refinement for the first time, making use of the insertion operations offered by the signpost data structure. The method uses an intrinsic variant of *Chew's 2nd algorithm* [Che93; She97], which essentially amounts to the following steps:

- Until a specified minimum angle bound is satisfied:
  - Flip to Delaunay
  - Find any triangle ijk that violates the angle bound
  - Insert the circumcenter p of ijk



In the intrinsic setting one must decide where to insert the circumcenter p if it is not contained in its triangle; we use a geodesic strategy, tracing tracing out the vector pointing to the circumcenter along the surface to an insertion location. Note also that in the intrinsic setting, the underlying domain may have skinny needle vertices where the total angle is already smaller than the user-specified minimum angle bound (see inset). The only change we make here is that we do not perform refinement when the skinny angle is incident on a degree-1 vertex, and hence cannot possibly be improved.

In practice intrinsic Delaunay refinement is quite effective, reproducing the behavior of the planar algorithm and consistently generating meshes with an interior angle bound of 30°. This matches the guarantee made in the planar case [Che87], though formally extending the analysis to the intrinsic setting is an area of ongoing work. The practical utility of this approach should not be understated: it automatically generates a surface triangulation with guaranteed quality bounds, and furthermore it does not require the tuning of any numerical parameters. Such behavior is extremely valuable for robust PDE-based geometry processing in practice, generating high-quality meshes for downstream applications without any tradeoff of approximation error, and only modestly increasing element counts (Figure 8).

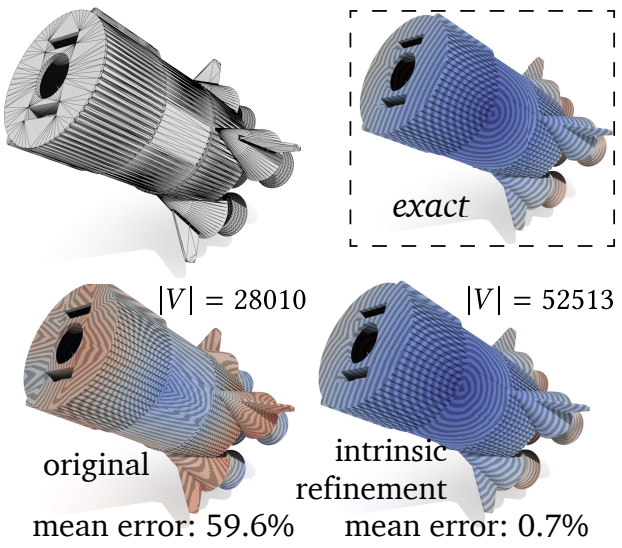


Figure 8: A fast distance algorithm [CWW17] is dramatically more accurate after intrinsic refinement.

4.4 Optimal Delaunay triangulation

An *optimal Delaunay triangulation* seeks to improve quality not just by refining the triangulation, but also by adjusting the placement of vertices [CX04]. Sharp, Soliman, and Crane [SSC19] apply this idea in the intrinsic setting by optimizing the location of inserted vertices—modifying the input vertices is of course undesirable, since it would change the surface geometry rather than just the triangulation. The basic strategy is to iteratively move all vertices toward the triangle area weighted sum of the circumcenters of incident triangles, again performing edge flips after each iteration to maintain the Delaunay property [CH11b, Equation 4.13]. In the intrinsic setting we can locate circumcenters as in the previous section; rather than averaging these locations, we simply average the vectors *to* these locations, then use this average as our update direction. We insert new vertices on each iteration by splitting edges longer than a user-defined target length (*à la* Tournois et al. [Tou+09]). In general we observe the same behavior as in the Euclidean case: in contrast to Delaunay refinement, we get a better distribution of areas, at the cost of some skinnier angles (Figure 7). Crucially, in this process we do not reposition the initial vertices, only those which we previously inserted, and thus continue to exactly preserve the geometry.

4.5 Simplification

Thus far, we have only discussed retriangulation routines which add vertices to the triangulation, or modify edges; the initial vertex set has always remained fixed. Another very important class

of retriangulation algorithms are simplifcation schemes, which generate a simpler version of the surface with similar geometry, often by collapsing edges. One widely-used method is quadric error simplification [GH97]. Simplification is an important geometry processing tool, with uses from compression [GGK02] to multigrid hierarchies [RL03].

Ongoing work in this thesis will explore *intrinsic* simplification routines. It is reasonable to expect that performing simplification in the larger space of intrinsic triangulations will yield new algorithms with improved properties, just as in the other retriangulation settings above. The key technical challenge is identifying a meaningful geometric notion of edge collapse in the intrinsic setting: how should an edge be selected for collapse to introduce minimal geometric error, and how should the intrinsic geometry be updated after the collapse operation? An additional hurdle is that many of our data structures for intrinsic triangulations (Section 3) implicitly leverage the fact that all triangles have no curvature on their interior, *i.e.* no vertices have been removed. Modifying these data structures to support removing vertices will likely be a necessary contribution.

5 Geodesics

Geodesic curves generalize the notion of a straight line to curved surfaces; they can be defined formally as locally-shortest paths, or curves of zero tangential acceleration. Fast and accurate geodesics are essential for tasks across science and engineering [Bos+11], and the ability to construct "straight lines" on polyhedral surfaces helps generalize classic algorithms from 2D computational geometry to curved surfaces.

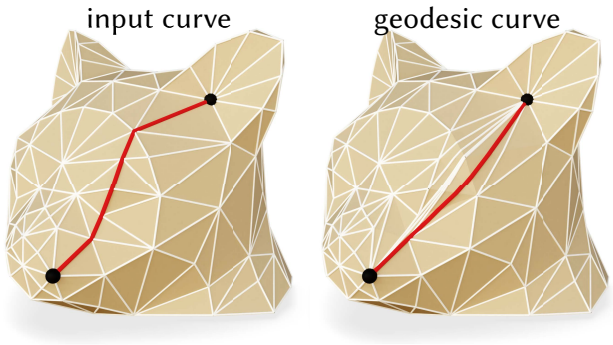
Computationally, geodesics are most widely studied in the context of finding *globally shortest* geodesics from a source point; such paths are useful for defining distance along a surface. The resulting algorithms have roots in the strategy of Mitchell, Mount, and Papadimitriou [MMP87]: start at the source, and use a Dijkstra-like traversal to track "windows" of geodesic paths sharing a common history. Many improvements, generalizations, and approximations have since been developed along this line of research [KS98; Sur+05; BK07; XW09; CWW13; YWH13; Xu+15; YXH14; Qin+16; Wan+17; Yin+19; AFH20; Cao+20].

An alternate approach to geodesics is shortening a *given* curve to be a geodesic, akin to a curve shortening flow on the surface [Gag90]. Such procedures can construct a larger space of geodesics beyond merely shortest geodesics including even closed loops, and can preserve the topological class of curves—both of which are critical for tasks like modeling and fabrication. *Lagrangian* approaches to curve shortening represent curves via vertices that can move freely along the surface or in $\mathbb{R}^3$ [HS94; MVC05; XW07; ASS09; XHF11; Han+17; Liu+17a; RŠN19], whereas *Eulerian* methods encode curves as level sets of a scalar function [Set89; WT10; Zha+10].

The work of Sharp and Crane [SC20b] demonstrates that intrinsic triangulations can be used to shorten curves on surfaces to be geodesic by just flipping edges. In some sense, this is quite natural: the edges of an intrinsic triangulation are already geodesic arcs along a surface, so one just needs a flipping strategy to intentionally introduce particular long geodesic arcs. The resulting strategy is quite simple, and more interestingly represents a totally different

approach to finding geodesics, compared to the window-based and unfolding-based approaches prevalent in past work. The remainder of this section will outline this intrinsic flip-based geodesic algorithm, and show its utility for geometry processing.

5.1 Flip geodesics



To find geodesic curves with intrinsic triangulations, we will always represent the curves as a sequence of edges in an intrinsic triangulation, akin to a path in the graph theory sense. On a normal mesh, this representation would be hopelessly restrictive: geodesics curves would need to pass through the interior of faces rather than along edges. However, with intrinsic triangulations we can modify the edge set of a triangulation while preserving the underlying geometry. Precisely, the input γ will be a path along the edges of an intrinsic triangulation $\mathcal{T}$ (or more generally a loop or network of paths and loops). The output will be a geodesic path γ' which is isotopic to the input, along the edges of an updated intrinsic triangulation $\mathcal{T}'$, Furthermore, this algorithm operates in the space of noncrossing curves: the input curves must not cross transversely, and it is guaranteed that the output will not contain any new crossings.

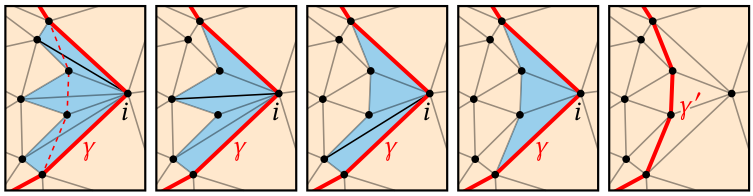


Figure 9: FLIPOUT shortens the curve γ by flipping edges.

The key algorithmic building block is the FLIPOUT subroutine, so-named because it flips edges out of a neighborhood to provably introduce a shorter path. At a vertex i where the path is not yet a geodesic, FLIPOUT simply repeatedly flips edges outgoing from i which form a convex diamond (see inset). In Sharp and Crane [SC20b, Theorem 4.1], we prove that this subroutine will always terminate with a shorter path along the perimeter; the proof for the general case of Δ-complex (Sharp and Crane [SC20b, Appendix A]) is nontrivial, but necessary because the triangulation may be reduced to a Δ-complex at intermediate stages of the algorithm. Shortening a path, loop, or curve network then simply amounts to repeatedly applying FLIPOUT, shortening the curve each time, until the curve is a geodesic. Since this routine involves only edge flip operations, it is easy to apply to a surface mesh using intrinsic triangulations. The rich data structures in Section 3 then recover the resulting geodesic as a polyline along the original surface.

5.2 Constrained intrinsic retriangulation

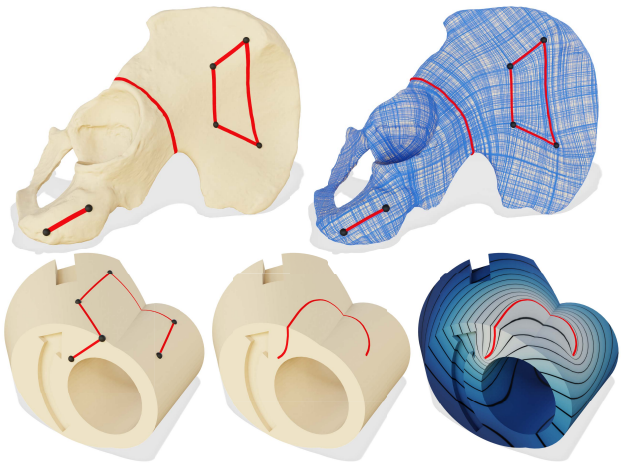


Figure 10: PDEs taking boundary conditions from constrained intrinsic triangulations.

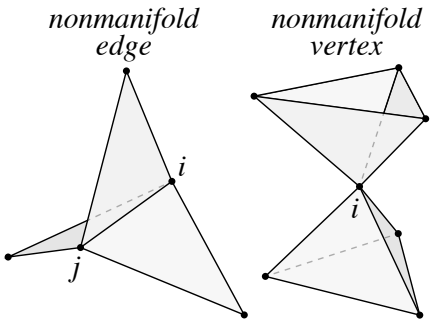
A key benefit of constructing geodesic curves via edge flips is that the procedure yields not just the curves, but also a triangulation of the domain which conforms to those curves. This allows the scheme to serve a much-needed role for generating constrained intrinsic triangulation of surfaces, akin to constrained Delaunay triangulation in the plane [Che89]. Such triangulations are a necessary ingredient for many applications in PDE-based geometry processing, where solutions must conform to some boundary conditions defined along the surface. Figure 10 demonstrates two practical applications of this technique, showcasing a cross field conforming to curves on a 3D scan of a pelvis [Knö+13], and a Poisson equation taking boundary conditions from a Bézier curve on a mechanical part (once one can construct geodesics, Bézier curves follow naturally via a de Casteljau-style subdivision [MCV08]).

5.3 Single-source geodesics

The methods described above are a kind of curve-shortening flow: they take some particular curve as a input, and shorten it to be a geodesic. It is natural to wonder whether the same techniques can be applied to the widely-studied single source all destination (SSAD) problem, where one seeks a geodesic path to all other vertices in a mesh.

Sharp and Crane [SC20b] also present simple algorithm for this task based on the FLIPOUT subroutine, the basic idea of which is to run a Dijkstra search outward from the source vertex, while constantly applying the FLIPOUT subroutine at the frontier to ensure all accepted paths are geodesic. Of course, this procedure is only guaranteed to yield geodesics, not necessarily globally-shortest geodesics, but in practice it is quite efficient and very often does find shortest geodesics (see inset). Perhaps the most interesting consequence of this algorithm is that it constructively implies the existence of a single triangulation of a surface, which contains among its edge set geodesics from a particular source vertex to every other vertex in the mesh—it is remarkable that such a triangulation exists at all.

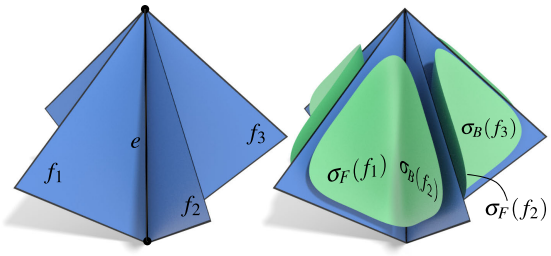
6 Generalized domains



Traditionally, the formulation of intrinsic triangulations begins with a strong assumption about the connectivity of the input domain: a manifold, oriented triangle mesh. *Manifold* connectivity requires local patches of the surface be homeomorphic to $\mathbb{R}^2$, that is, any neighborhood of the surface looks like the plane; examples of nonmanifold edges and vertices are shown inset. However, geometric data encountered in practice often has nonmanifold features, either intentionally or due to noise, or may have even less structure in the form of a point cloud—handling such inputs is crucial to leverage intrinsic triangulations for robust geometry processing. This section outlines the work of Sharp and Crane [SC20a], which presents a covering space generalizing intrinsic triangulations to nonmanifold meshes. The same technique can even be applied to point clouds.

6.1 Nonmanifold intrinsic triangulations

The key observation is that only nonmanifold *edges* obstruct the usage of intrinsic triangulations, because there is no notion of an edge flip at a nonmanifold edge. Nonmanifold *vertices* are a non-issue. One can then construct a particular double-covering space of the triangulation, called the *tufted cover*, for which all edges are manifold [SC20a]. The intrinsic triangulation of this covering space then supports edge flip operations, which can be used *e.g.* to construct the high-quality intrinsic Delaunay Laplacian, and use it as the Laplace matrix for the original nonmanifold mesh.



In particular, the tufted cover is defined by splitting each face f in to two copies $\sigma_F(f)$ and $\sigma_B(f)$, then cyclically gluing together consecutive faces around each original edge (see inset). By construction, the tufted cover will always be an edge manifold, closed, oriented triangulation, though it is nonmanifold at almost every vertex. The resulting triangulation will have twice as many faces, but the exact same vertex set as the original mesh, which makes it trivial to transfer functions and operators defined at vertices between the tufted cover and the original mesh. The name "tufted cover" arises because the nonmanifold vertices are reminiscent of the buttons on tufted upholstery.

It should be emphasized that the resulting triangulation still lacks vertex-manifold connectivity; in fact it is nonmanifold at nearly every vertex—we have constructed a triangulation that has exactly the right structure to apply intrinsic edge flips, but have not otherwise rectified the nonmanifoldness. Other intrinsic triangulation-based methods might not immediately apply (such as the geodesic algorithm of Section 5). However, the tufted cover provides enough structure for the core task of generating the intrinsic Delaunay Laplacian, and is a useful starting point for subsequent developments. The intrinsic Delaunay triangulation of the tufted cover

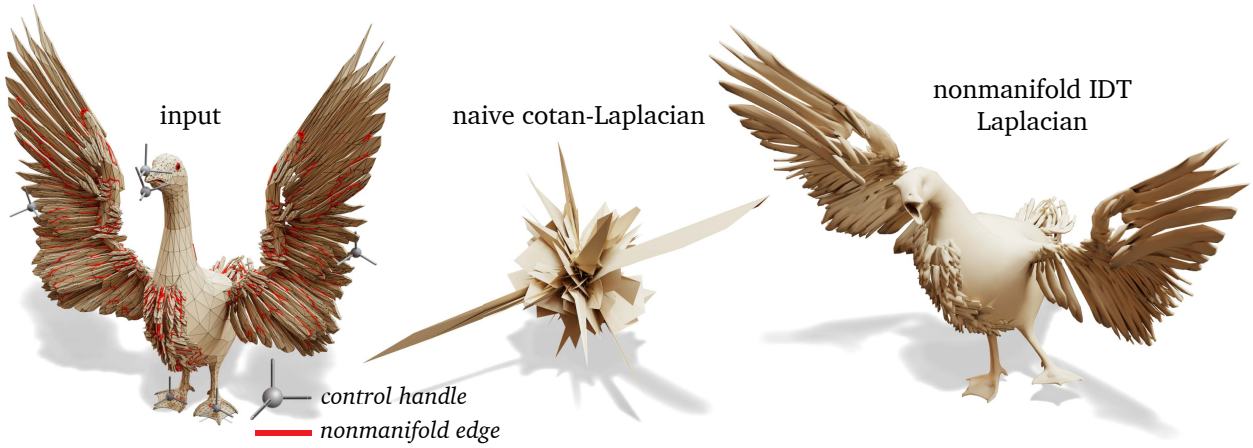


Figure 11: Robustness in the context of differential surface editing [Yu+04; Lip+04; Sor+04], where a system of equations involving a Laplacian is solved to deform a 3D model. Applying these techniques naively in the input mesh yields only numerical noise, but substituting the nonmanifold IDT Laplacian generates the expected smooth deformation.

offers a high-quality cotan-Laplace matrix for nonmanifold meshes for the first time, extending the benefits to *all* triangle meshes without restrictions on connectivity. Substituting this operator in to existing algorithms immediately improves there performance and robustness on low-quality input meshes (Figure 11).

6.2 Point clouds

Point clouds, simply a collection of points $p \in \mathbb{R}^3$, are a common alternative to meshes for representing surfaces in geometry processing, though the absence of connectivity makes many computations more difficult. A common strategy for building a Laplacian on a point cloud is to identify the k nearest neighbors of each point, (ii) project these neighbors onto an estimated tangent plane, and (iii) construct the planar Delaunay triangulation of the projected points. These local triangulations are then used to build a Laplacian in a variety of ways, though existing schemes involve difficult-to-tune numerical parameters [BSW08; LPG12], or do not guarantee the Delaunay property [CRT04; Cao+10]. In Sharp and Crane [SC20a], we observe that nonmanifold intrinsic Delaunay triangulations can be directly applied in this setting. The strategy is to take the union of all local triangulations,

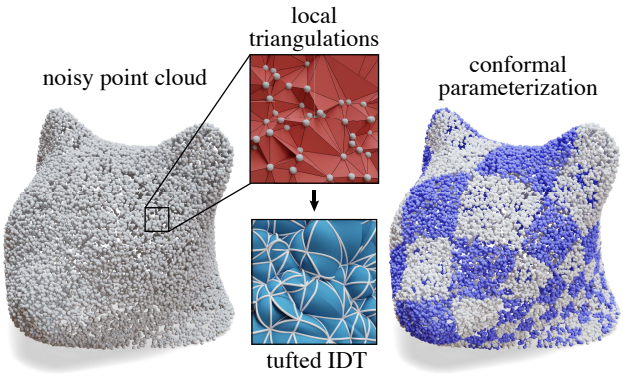


Figure 12: The intrinsic Delaunay point cloud Laplacian, demonstrated here for spectral conformal parameterization of a point cloud [Mul+08].

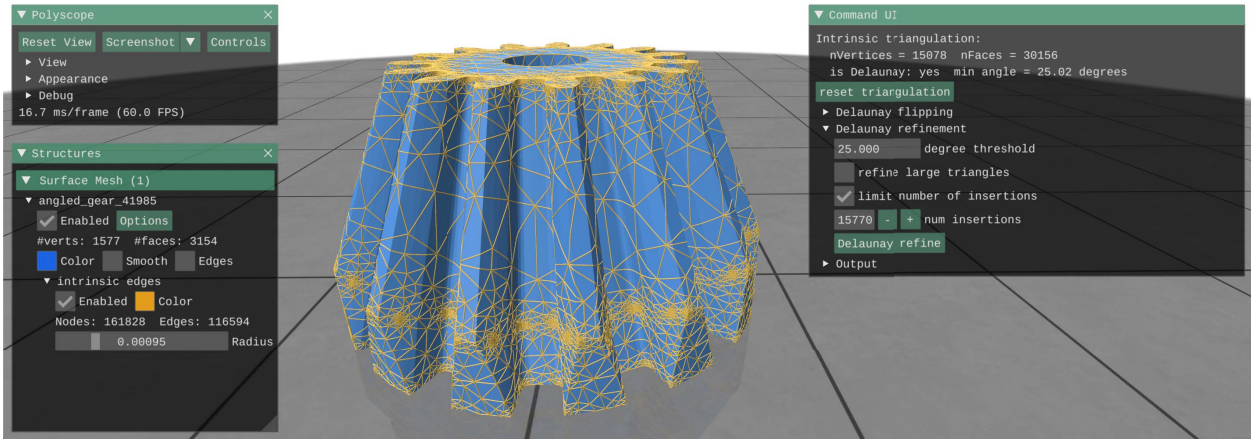


Figure 13: An intrinsic triangulation for robust computation on a poorly-triangulated engineering model, generated via the method of Sharp, Soliman, and Crane [SSC19]. This example is computed in the geometry-central library [SC+19].

yielding a triangulation which has highly highly irregular connectivity, is nonmanifold almost everywhere, and has duplicate copies of many faces. However, one can proceed as before, building the tufted cover, flipping to the intrinsic Delaunay triangulation, and reading off the cotan-Laplacian. The resulting Laplace matrix has excellent numerical properties and does not require tuning any parameters, Figure 12 shows the use of this operator for geometry processing on point clouds.

7 Outline and timeline

This thesis will explore the theory and practice of intrinsic triangulations in geometry processing. Intrinsic triangulations bring together formal aspects of discrete differential geometry with classical algorithm and data structure development, yielding new techniques with great practical utility, especially for robust geometry processing.

7.1 Summary of contributions

The novel technical contributions of this thesis are summarized as follows:

1. **Signpost intrinsic triangulations** [SSC19]: We present a new representation for intrinsic triangulations, the first to offer all key operations in constant time. In addition, we describe many important operations (*e.g.* refinement of intrinsic triangulations) for the first time, in the context of the signpost data structure.

2. **Edge flip geodesics** [SC20b]: We develop a new algorithm for constructing geodesic curves on surfaces, using edge flips in an intrinsic triangulation. This method is a new

perspective on an old problem, and enables the construction of constrained intrinsic triangulations.

3. **Tufted cover** [SC20a]: We define the tufted cover of a nonmanifold mesh, extending the benefits of intrinsic triangulations to all triangle meshes regardless of connectivity, and even point clouds.

4. **Software systems** [SC+19]: We design and implement a general and efficient software library for geometry processing, which for the first time allows intrinsic triangulations to be seamlessly used as a backend data structure, and supports all of the algorithms described in this thesis (Figure 13).

The existing contributions listed above already comprise the primary body of novel contributions for this thesis. In many ways, the most crucial advancement needed for intrinsic triangulations is approachable communication to a broader audience, and usable software tools to motivate widespread adoption. Hopefully, the content of this thesis will serve as a valuable reference to that effect. In addition, the following technical work is in-progress at the time of this proposal, and may provide further valuable additions:

1. **Normal coordinates.** Ongoing work will develop a framework for representing intrinsic triangulations using normal coordinates, and explore implications for robust geometry processing (Section 3.4). This work will be performed in collaboration with fellow PhD student Mark Gillespie, who already has already established promising initial results.

2. **Simplification.** We will investigate the possibility of performing intrinsic simplification, as well as algorithmic advances enabled by simplification (Section 4.5). This will require both extending data structures to represent intrinsic triangulations, as well as identifying geometrically appropriate simplification rules.

7.2 Timeline

Ongoing work will focus on producing the thesis document itself and contributing to accompanying software releases, as well as continued research towards future contributions.

- FALL 2018: developed signpost data structure [SSC19]
- SPRING-FALL 2019: initial work on edge flip geodesics
- SPRING 2020: culmination of edge flip geodesics [SC20b]
- SUMMER 2020: developed nonmanifold intrinsic triangulations [SC20a]
- FEB 2021: thesis proposal
- FEB-MAY 2021: work towards ongoing contributions
- FEB-MAY 2021: prepare and submit thesis
- JUN 2021: thesis defense

References

[AFH20] Y. Adikusuma, Z. Fang, and Y. He. “Fast Construction of Discrete Geodesic Graphs”. *ACM Trans. Graph.* 39.2 (2020).

[All+08] Pierre Alliez, Giuliana Ucelli, Craig Gotsman, and Marco Attene. “Recent advances in remeshing of surfaces”. *Shape analysis and structuring* (2008), pp. 53–82.

[ASS09] E. Appleboim, E. Saucan, and J. Stern. *Normal Approximations of Geodesics on Smooth Triangulated Surfaces*. Tech. rep. CCIT Report, 2009.

[Bau75] Bruce G Baumgart. “A polyhedron representation for computer vision”. *Proceedings of the May 19-22, 1975, national computer conference and exposition*. 1975, pp. 589–596.

[BK07] D. Bommes and L. Kobbelt. “Accurate Computation of Geodesic Distance Fields for Polygonal Curves on Triangle Meshes”. *VMV*. Vol. 7. 2007, pp. 151–160.

[Bos+11] P. Bose, A. Maheshwari, C. Shu, and S. Wuhrer. “A Survey of Geodesic Paths on 3D Surfaces”. *Comput. Geom. Theory Appl.* 44.9 (Nov. 2011). ISSN: 0925-7721.

[Bot+10] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon mesh processing*. CRC press, 2010.

[BS07] A. Bobenko and B. Springborn. “A Discrete Laplace–Beltrami Operator for Simplicial Surfaces”. *Discrete & Computational Geometry* 38.4 (2007).

[BSW08] Mikhail Belkin, Jian Sun, and Yusu Wang. “Discrete Laplace Operator on Meshed Surfaces”. *Symp. Comp. Geom.* 2008.

[Cao+10] Junjie Cao, Andrea Tagliasacchi, Matt Olson, Hao Zhang, and Zhinxun Su. “Point Cloud Skeletons via Laplacian Based Contraction”. *Shape Mod. Int. Conf.* IEEE. 2010.

[Cao+20] L. Cao, J. Zhao, J. Xu, S. Chen, G. Liu, S. Xin, Y. Zhou, and Y. He. “Computing Smooth Quasi-geodesic Distance Field (QGDF) with Quadratic Programming”. *Computer-Aided Design* (2020).

[CDS12] Siu-Wing Cheng, Tamal K Dey, and Jonathan Shewchuk. *Delaunay mesh generation*. CRC Press, 2012.

[CH11a] Long Chen and Michael Holst. “Efficient mesh optimization schemes based on optimal Delaunay triangulations”. *Computer Methods in Applied Mechanics and Engineering* 200.9-12 (2011), pp. 967–984.

[CH11b] Long Chen and Michael J. Holst. “Efficient Mesh Optimization Schemes based on Optimal Delaunay Triangulations”. *Comput. Methods Appl. Mech. Engrg.* 200 (2011).

[Che87] L. P. Chew. “Constrained Delaunay Triangulations”. *Proceedings of the Third Annual Symposium on Computational Geometry*. SCG ’87. ACM, 1987. ISBN: 0-89791-231-4.

[Che89] L. Chew. “Constrained Delaunay Triangulations”. *Algorithmica* 4.1-4 (1989).

[Che93] L. P. Chew. “Guaranteed-quality Mesh Generation for Curved Surfaces”. *Proceedings of the Ninth Annual Symposium on Computational Geometry*. SCG ’93. ACM, 1993. ISBN: 0-89791-582-8.

[CRT04] Ulrich Clarenz, Martin Rumpf, and Alexandru Telea. “Finite Elements on Point Based Surfaces”. *SPBG*. 2004.

[CWW13] K. Crane, C. Weischedel, and M. Wardetzky. “Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow”. *ACM Trans. Graph.* 32.5 (Oct. 2013). ISSN: 0730-0301.

[CWW17] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. “The Heat Method for Distance Computation”. *Commun. ACM* 60.11 (Oct. 2017), pp. 90–99. ISSN: 0001-0782. DOI: 10.1145/3131280. URL: http://doi.acm.org/10.1145/3131280.

[CX04] Long Chen and Jin-chao Xu. “Optimal delaunay triangulations”. *Journal of Computational Mathematics* (2004).

[Fis+07] M. Fisher, B. Springborn, P. Schröder, and A. Bobenko. “An Algorithm for the Construction of Intrinsic Delaunay Triangulations with Applications to Digital Geometry Processing”. *Computing* 81.2 (Nov. 2007). ISSN: 1436-5057.

[Gag90] M. Gage. “Curve Shortening on Surfaces”. en. *Annales scientifiques de l’École Normale Supérieure* Ser. 4, 23.2 (1990), pp. 229–256.

[GGK02] Craig Gotsman, Stefan Gumhold, and Leif Kobbelt. “Simplification and compression of 3D meshes”. *Tutorials on multiresolution in geometric modelling*. Springer, 2002, pp. 319–361.

[GH97] Michael Garland and Paul S Heckbert. “Surface simplification using quadric error metrics”. *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 1997, pp. 209–216.

[Hak61] Wolfgang Haken. “Theorie der normalflächen”. *Acta Mathematica* 105.3-4 (1961), pp. 245–375.

[Han+17] X. Han, H. Yu, Y. Yu, and J. Zhang. “A Fast Propagation Scheme for Approximate Geodesic Paths”. *Graphical Models* 91 (2017), pp. 22–29. ISSN: 1524-0703.

[Hat02] A. Hatcher. *Algebraic Topology*. Algebraic Topology. Cambridge University Press, 2002. ISBN: 9780521795401.

[HS94] J. Hass and P. Scott. “Shortening Curves on Surfaces”. *Topology* 33.1 (1994).

[Ind+01] C. Indermitte, T. Liebling, M. Troyanov, and H. Clémençon. “Voronoi Diagrams on Piecewise Flat Surfaces and an Application to Biological Growth”. *Theoretical Computer Science* 263 (2001).

[Ket99] Lutz Kettner. “Using generic programming for designing a data structure for polyhedral surfaces”. *Computational Geometry* 13.1 (1999), pp. 65–90.

[Kne29] Hellmuth Kneser. “Geschlossene Flächen in dreidimensionalen Mannigfaltigkeiten.” *Jahresbericht der Deutschen Mathematiker-Vereinigung* 38 (1929), pp. 248–259.

[Knö+13] F. Knöppel, K. Crane, U. Pinkall, and P. Schröder. “Globally Optimal Direction Fields”. *ACM Trans. Graph.* 32.4 (2013).

[KS98] R. Kimmel and J. Sethian. “Fast Marching Methods on Triangulated Domains”. *Proc. Nat. Acad. Sci.* 95 (1998).

[KSS06] L. Kharevych, B. Springborn, and P. Schröder. “Discrete Conformal Mappings via Circle Patterns”. *ACM Trans. Graph.* 25.2 (2006).

[Law77] C. Lawson. “Software for C1 Surface Interpolation”. *Mathematical software*. Elsevier, 1977, pp. 161–194.

[Lip+04] Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, Christian Rossi, and Hans-Peter Seidel. “Differential Coordinates for Interactive Mesh Editing”. *Proc. Shape Mod. App.* 2004.

[Liu+15] Yong-Jin Liu, Chun-Xu Xu, Dian Fan, and Ying He. “Efficient construction and simplification of Delaunay meshes”. *ACM Transactions on Graphics (TOG)* 34.6 (2015), pp. 1–13.

[Liu+17a] B. Liu, S. Chen, S. Xin, Y. He, Z. Liu, and J. Zhao. “An Optimization-driven Approach for Computing Geodesic Paths on Triangle Meshes”. *Computer-Aided Design* 90 (2017).

[Liu+17b] Yong-Jin Liu, Dian Fan, Chun-Xu Xu, and Ying He. “Constructing intrinsic Delaunay triangulations from the dual of geodesic Voronoi diagrams”. *ACM Transactions on Graphics (TOG)* 36.2 (2017), pp. 1–15.

[LPG12] Yang Liu, Balakrishnan Prabhakaran, and Xiaohu Guo. “Point-based Manifold Harmonics”. *IEEE Trans. Vis. Comp. Graph.* 18.10 (2012).

[Luo04] Feng Luo. “Combinatorial Yamabe flow on surfaces”. *Communications in Contemporary Mathematics* 6.05 (2004), pp. 765–780.

[MCV08] D. Morera, P. Carvalho, and L. Velho. “Modeling on Triangulations with Geodesic Curves”. *The Visual Computer* 24.12 (Dec. 2008).

[MMP87] J. Mitchell, D. Mount, and C. Papadimitriou. “The Discrete Geodesic Problem”. *SIAM J. Comput.* 16.4 (Aug. 1987). ISSN: 0097-5397.

[Mul+08] Patrick Mullen, Yiying Tong, Pierre Alliez, and Mathieu Desbrun. “Spectral Conformal Parameterization”. *Comp. Graph. Forum*. Vol. 27. 5. Wiley Online Library. 2008.

[MVC05] D. Martınez, L. Velho, and P. Carvalho. “Computing Geodesics on Triangular Meshes”. *Computers & Graphics* 29.5 (2005).

[Qin+16] Y. Qin, X. Han, H. Yu, Y. Yu, and J. Zhang. “Fast and Exact Discrete Geodesic Computation based on Triangle-oriented Wavefront Propagation”. *ACM Trans. Graph.* 35.4 (2016).

[Reg61] Tullio Regge. “General relativity without coordinates”. *Il Nuovo Cimento (1955-1965)* 19.3 (1961), pp. 558–571.

[Riv94] I. Rivin. “Euclidean Structures on Simplicial Surfaces and Hyperbolic Volume”. *Annals of mathematics* 139.3 (1994).

[RL03] Nicolas Ray and Bruno Lévy. “Hierarchical least squares conformal map”. *11th Pacific Conference onComputer Graphics and Applications, 2003. Proceedings.* IEEE. 2003, pp. 263–270.

[RŠN19] M. Remešíková, M. Šagát, and P. Novysedlák. “Discrete Lagrangian Algorithm for Finding Geodesics on Triangular Meshes”. *Applied Mathematical Modelling* (2019). ISSN: 0307-904X.

[SC+19] N. Sharp, K. Crane, et al. *geometry-central*. www.geometry-central.net. 2019.

[SC20a] Nicholas Sharp and Keenan Crane. “A laplacian for nonmanifold triangle meshes”. *Computer Graphics Forum*. Vol. 39. 5. Wiley Online Library. 2020, pp. 69–80.

[SC20b] Nicholas Sharp and Keenan Crane. “You Can Find Geodesic Paths in Triangle Meshes by Just Flipping Edges”. *ACM Trans. Graph.* 39.6 (2020).

[Set89] J. Sethian. “A Review of Recent Numerical Algorithms for Hypersurfaces Moving with Curvature Dependent Speed”. *J. Differential Geometry* 31 (1989), pp. 131–161.

[She97] Jonathan Richard Shewchuk. “Delaunay Refinement Mesh Generation”. Tech Report CMU-CS-97-137. PhD thesis. Carnegie Mellon University, 1997.

[Sor+04] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and H-P Seidel. “Laplacian Surface Editing”. *Symp. Geom. Proc.* 2004.

[SSC19] N. Sharp, Y. Soliman, and K. Crane. “Navigating Intrinsic Triangulations”. *ACM Trans. Graph.* 38.4 (2019).

[SSŠ02] Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. “Algorithms for normal curves and surfaces”. *International Computing and Combinatorics Conference*. Springer. 2002, pp. 370–380.

[Sur+05] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. Gortler, and H. Hoppe. “Fast Exact and Approximate Geodesics on Meshes”. *ACM Trans. Graph.* 24.3 (2005).

[Tou+09] Jane Tournois, Camille Wormser, Pierre Alliez, and Mathieu Desbrun. “Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation”. 28.3 (2009), p. 75.

[Wan+17] X. Wang, Z. Fang, J. Wu, S. Xin, and Y. He. “Discrete Geodesic Graph for Computing Geodesic Distances on Polyhedral Surfaces”. *Computer Aided Geometric Design* 52 (2017).

[Wei85] Kevin Weiler. “Edge-based data structures for solid modeling in curved-surface environments”. *IEEE Computer graphics and applications* 5.1 (1985), pp. 21–40.

[WT10] C. Wu and X. Tai. “A Level Set Formulation of Geodesic Curvature Flow on Simplicial Surfaces”. *IEEE Transactions on Visualization and Computer Graphics* 16.4 (July 2010).

[XHF11] S. Xin, Y. He, and C. Fu. “Efficiently Computing Exact Geodesic Loops within Finite Steps”. *IEEE Transactions on Visualization and Computer Graphics* 18.6 (2011).

[Xu+15] C. Xu, T. Wang, Y. Liu, L. Liu, and Y. He. “Fast Wavefront Propagation for Computing Exact Geodesic Distances on Meshes”. *IEEE transactions on visualization and computer graphics* 21.7 (2015).

[XW07] S. Xin and G. Wang. “Efficiently Determining a Locally Exact Shortest Path on Polyhedral Surfaces”. *Computer-Aided Design* 39.12 (2007).

[XW09] S. Xin and G. Wang. “Improving Chen and Han’s Algorithm on the Discrete Geodesic Problem”. *ACM Trans. Graph.* 28.4 (2009).

[Yin+19] X. Ying, C. Huang, X. Fu, Y. He, R. Yu, J. Wang, and M. Yu. “Parallelizing Discrete Geodesic Algorithms with Perfect Efficiency”. *Computer-Aided Design* 115 (2019).

[Yu+04] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. “Mesh Editing with Poisson-based Gradient Field Manipulation”. *SIGGRAPH*. 2004.

[YWH13] X. Ying, X. Wang, and Y. He. “Saddle Vertex Graph: a Novel Solution to the Discrete Geodesic Problem”. *ACM Trans. Graph.* 32.6 (2013).

[YXH14] X. Ying, S. Xin, and Y. He. “Parallel Chen-Han (PCH) Algorithm for Discrete Geodesics”. *ACM Trans. Graph.* 33.1 (2014).

[Zha+10] J. Zhang, C. Wu, J. Cai, J. Zheng, and X. Tai. “Mesh snapping: Robust Interactive Mesh Cutting Using Fast Geodesic Curvature Flow”. *Computer graphics forum*. Vol. 29. 2. Wiley Online Library. 2010.

[ZJ16] Q. Zhou and A. Jacobson. “Thingi10K: A Dataset of 10,000 3D-Printing Models”. *arXiv:1605.04797* (2016).