19th century mathematics

**EXTRINSIC**
(embedding/parameterized surface)

20th century mathematics

**INTRINSIC**
(Riemannian manifold/atlas of charts)
20th century computing

Extrinsic Triangulation
(connectivity + vertex positions)

$\mathbf{x}_i \in \mathbb{R}^3$

$\mathbf{x}_j \in \mathbb{R}^3$

21st century computing?

Intrinsic Triangulation
(connectivity + edge lengths)

$\ell_{ij} \in \mathbb{R}_{>0}$
Intrinsic Triangulations of Extrinsic Polyhedra

**Extrinsic Triangulation**

**Intrinsic Triangulation**

---

**Rough idea:** allow “bent” triangles that are *intrinsically* flat.
Extrinsic

Correspondence

Data structures

Useful algorithms

Intrinsic
Why is it useful to “draw” one triangulation on top of another?
Motivation from three different angles…

- geometry processing
- computational geometry
- scientific computing
Motivation—Computational Geometry (2D)

- Input triangulation
- Improve (preserve vertices)
- Improve (preserve both vertices & edges)
Motivation—Computational Geometry (3D)

input triangulation

extrinsic improvement

intrinsic improvement

To exactly preserve input geometry, must always preserve vertices & edges.

Only option is to refine!

...Or is it?

Key idea: think of input surface as “background domain” (just like $\mathbb{R}^2$)
Motivation—Computational Geometry

By adopting this viewpoint, can “port” algorithms from 2D to surfaces:

- Steiner tree approximation
- Delaunay refinement
- Constrained Delaunay triangulation (CDT)
- Steiner tree approximation
Motivation from three different angles…

- Geometry processing
- Computational geometry
- Scientific computing
Traditionally, mesh must serve two conflicting goals.
# Finite Element Meshing—No Free Lunch?

## Table of Mesh Quality Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Same Geometry</th>
<th>Same # Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element Quality</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Mesh Size</td>
<td>🚗</td>
<td>🚗</td>
</tr>
<tr>
<td>Geometric Quality</td>
<td>✔️</td>
<td>🚗</td>
</tr>
</tbody>
</table>

*Note: The table shows the comparison of mesh quality parameters for different scenarios.*

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Number of Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>$F = 3k$</td>
</tr>
<tr>
<td>Same Geometry</td>
<td>$F = 330k$</td>
</tr>
<tr>
<td>Same # Elements</td>
<td>$F = 3k$</td>
</tr>
</tbody>
</table>

*Note: The image shows a comparison of mesh quality parameters for different scenarios.*

---

*Example image description: The diagram illustrates the comparison of mesh quality parameters for different scenarios. The mesh size and geometric quality are consistent across all scenarios, while the element quality changes depending on the scenario.*
Intrinsic Basis Functions

standard basis functions
(piecewise linear)

intrinsic basis functions
(de-coupled from extrinsic triangles)

(can still use any standard bases, e.g., quadratic Lagrange)
Surface Meshes—Free Lunch

- geometric quality
- element quality
- mesh size

| F | = 20k
Motivation from three different angles…

- Computational geometry
- Geometry processing
- Scientific computing
Motivation—Geometry Processing

- Many geometric algorithms expressed purely in terms of intrinsic data:
  - Laplace-Beltrami operator $\Delta$
  - harmonic spectrum
  - (Gaussian) curvature
  - geodesic distance
  - Wasserstein distance
  - functional maps
  - …

Why then process the mesh itself from an extrinsic viewpoint?
Motivation—Robust Geometry Processing

Geometry encountered in real datasets can be really, really bad:

“Hell is other people’s meshes.”
—Jean-Paul Sartre
Motivation—Robust Geometry Processing

Data keeps getting worse over time…

"bad mesh" (ca. 1990)

"bad mesh" (ca. 2000)

"bad mesh" (ca. 2020)

"bad mesh" (ca. 2030)

How can we abstract away the mesh, and focus on the geometry?
Preconditioning in Numerical Linear Algebra

Key idea: user can forget about matrix, and just focus on what it represents.
"Preconditioning" in Geometry Processing

Intrinsic triangulations provide bridge between "bad meshes" and non-robust algorithms:

Key idea: user can forget about mesh, and just focus on what it represents.
Motivation from three different angles…

- Geometry processing
- Computational geometry
- Scientific computing
Overview

I. Today:
   1. Background & history
   2. Data structures I: signposts
   3. Data structures II: normal coords.

II. Tomorrow:
   4. Discrete Laplace operator
   5. Geodesics
   6. Open questions
Sneak Preview

Some highlights:

- Guaranteed quality surface meshing
  - extend Delaunay refinement, constrained Delaunay triangulation (CDT) to surfaces

- Extremely robust injective mapping
  - enabled by integer-based encoding of correspondence

- Laplacian for nonmanifold meshes + point clouds
  - provides bridge between mesh & point processing

- Greedy edge flip algorithm for geodesics
  - unifies geodesics + surface retriangulation
Background & History
Recall: allow “bent” triangles that are intrinsically flat.
Hierarchy of Triangulations

extrinsic

geodesic

intrinsinc triangulations

geodesic triangulations

extrinsic triangulations

extrinsic

geodesic*

*Requirement: no vertices inside triangles.

intrinsinc (connectivity + edge lengths)
Intrinsic Triangulation—Example

Not all intrinsic triangulations are geodesic triangulations:

Perfectly good metric space; does not come from a polyhedron in $\mathbb{R}^3$. 
An intrinsic triangulation consists of:

- (topology) A surface triangulation* $K = (V, E, F)$
- (geometry) A discrete metric $\ell: E \rightarrow \mathbb{R}_{>0}$
  - lengths must satisfy triangle inequality $\ell_{ij} + \ell_{jk} \geq \ell_{ki} \ \forall ijk \in F$

Need not be simplicial:

(*Formally: a 2-manifold $\Delta$-complex)
Evaluating Geometric Quantities

Can compute basic geometric quantities directly from lengths:

\[
\theta_{ij} = \arccos \left( \frac{\ell_{ij}^2 + \ell_{ik}^2 - \ell_{ij}^2}{2\ell_{ij}\ell_{ik}} \right)
\]

\[
\text{area} = \sqrt{s(s - \ell_{ij})(s - \ell_{jk})(s - \ell_{ki})}
\]

\[
s := \frac{1}{2}(\ell_{ij} + \ell_{jk} + \ell_{ki})
\]

(…and so on.)
Geometrically, intrinsic triangulation describes a *polyhedral cone metric*:

- zero Gaussian curvature at faces, *edges*
- singular distribution of curvature at vertices ("cones") — given by angle defect
Cone Metric

Geometrically, intrinsic triangulation describes a *polyhedral cone metric*:

- zero Gaussian curvature at faces, *edges*
- singular distribution of curvature at vertices ("cones") — given by angle defect

**Key idea:** *many* triangulations encode *identical* cone metric—nothing special about input!
From Extrinsic to Intrinsic

**SMOOTH**

Parameterized surfaces

Riemannian manifolds
General Relativity without Coordinates.

T. Regge

Palmer Physical Laboratory, Princeton University - Princeton, N. J. (*)

(ricevuto il 17 Ottobre 1960)

Summary. — In this paper we develop an approach to the theory of Riemannian manifolds which avoids the use of co-ordinates. Curved spaces are approximated by higher-dimensional analogs of polyhedra.

Also (5) is formally suitable for relaxation methods or it could be programmed on a computer if the question of the existence of a solution is first met.
It is interesting to notice that the intrinsic geometry of $M$ is completely fixed by the connection matrix and the length of all edges. The connection matrix is essentially a list of all faces, edges and vertices of $M$ and a list of their mutual relationship i.e. by reading it one can decide which vertices, edges belong to a given face, etc. The connection matrix supplies us with all the topological information needed in the construction of $M$.

Since we are chiefly interested in the intrinsic geometry of manyfolds we are not particularly interested in the edges of $M$ and we regard them as a rather immaterial convention for dividing $M$ into triangles, any other convention being just as good.

Once we have constructed a simplectic net on $M$ (in plain words a division of $M$ into triangles) the knowledge of the lengths of all edges of $M$ implies the knowledge of all angles $\sigma_i$, and therefore of the deficiencies $\epsilon_n$.

The notion of simplectic net replaces therefore the notion of a co-ordinate net. The metric tensor on the other hand is replaced by the lengths of the edges.
Incomplete History of Intrinsic Triangulations & Geometry Processing

- Alexandrov (1942) — embeddability of convex polyhedral metrics
- Regge (1961) — use intrinsic triangulations to model general relativity
- Thurston et al (1970s) — hyperbolic triangulations, 3-manifolds, geometrization
- Rivin (1994) — first to consider intrinsic Delaunay triangulations, embedding *ideal hyperbolic polyhedra*
- Indermitte (2001) — show that intrinsic version of Lawson’s Delaunay flip algorithm works
- Luo (2004) — discrete uniformization via intrinsic triangulations, edge flips
- Bobenko & Springborn (2005) — intrinsic Laplace operator via intrinsic Delaunay
- Kharevych, Springborn, Schröder (2005) — compute circle patterns w/ intrinsic Delaunay
- Bobenko & Izwestiev (2006) — give algorithm for embedding convex polyhedra
- Bobenko, Pinkall, Schröder, Springborn, Gu, Sun, Wu, Yau … (2000’s/2010’s) — discrete uniformization

Not much use (so far!) for practical geometry processing. Still some missing ingredients…
Intrinsic Edge Flip

**Basic operation:** *intrinsic* edge flip.

**Key idea:** intrinsic edge flips do not change geometry.
**Intrinsic Delaunay Triangulations**

- *Delaunay property* ensures good behavior of many geometric algorithms
- *Many* equivalent characterizations, e.g., empty circumcircle property
- More useful definition in intrinsic case: angles $\alpha, \beta$ opposite each edge sum to $\leq \pi$
- **Lawson’s algorithm**: flip any edge until all edges are (intrinsic) Delaunay.
- Useful fact: you can *always* flip a non-Delaunay edge (even in a $\Delta$-complex)
- **Theorem.** *Intrinsic flip algorithm terminates!* [Indermitte 2001, Bobenko & Springborn 2005]
Intrinsic Laplacian

- Laplace-Beltrami operator $\Delta$ fundamental in geometry processing algorithms
- Canonical discrete Laplace operator [Bobenko & Springborn 2005]:
  1. Flip to intrinsic Delaunay triangulation.
  2. Build usual cotan-Laplace matrix.
- **Fact.** Cotan weights are nonnegative if and only if edge is Delaunay.
  - Zero for concyclic triangles ⇒ operator is canonical
  - Get discrete maximum principle (no spurious extrema)
- In practice: more robust geometry processing

\[
\alpha + \beta \leq \pi \iff \cot \alpha + \cot \beta \geq 0
\]
Robustness of Intrinsic Laplacian — Example

- **Example:** geodesic distance via *heat method* [Crane, Weischedel, Wardetzky 2013]
  
  - compute distance via heat diffusion $\frac{d}{dt} u = \Delta u$
  - here, intrinsic Laplacian $\Delta$ gives much better accuracy on highly anisotropic meshes

\[
\frac{d}{dt} u = \Delta u
\]
Conformal Surface Flattening

- Conformal maps widely used for texture mapping, fabrication, machine learning...
- Two metrics are *discretely conformally equivalent* if length cross ratios $c_{ij}$ are preserved
- **Given:** $K = (V, E, F), \ell : E \to \mathbb{R}_{>0}$
- **Find:** discretely conformally equivalent metric $\tilde{\ell}$ with zero angle defect
  - Flow toward flat configuration; lay out in plane
- **Theorem.** Always possible—*if you allow intrinsic edge flips!*

Surveys: Crane 2019, “Conformal Geometry of Simplicial Surfaces”
Gu, Luo, & Yau 2020, “Computational conformal geometry behind modern technologies”
Other Intrinsic Algorithms

• A few other algorithms have used intrinsic triangulations:
  – [Kharevych et al 2005] Use intrinsic Delaunay to compute guaranteed-injective conformal maps via circle patterns
  – [Bobenko & Izmestiev 2006] Use intrinsic edge flips to embed convex polyhedra in $\mathbb{R}^3$
  – [Campen & Zorin 2017] Field-aligned parameterization; use intrinsic triangulation to represent vertices with large curvature

That’s about it! Not much more use (so far…) of intrinsic triangulations.
Why So Few Practical Algorithms?

• *All* intrinsic algorithms so far assume a fixed vertex set
  ‣ triangulation changed purely via *edge flips*

• Classic algorithms use many operations: *insert vertex, move vertex, split edge, …*

How can we implement a much broader range of algorithms in the intrinsic setting?
Note on Visualization

colored triangles = intrinsic triangulation
black wireframe = input mesh
# Data Structures I: Signposts

<table>
<thead>
<tr>
<th>Day I</th>
<th>Day II</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. BACKGROUND</td>
<td>4. LAPLACIAN</td>
</tr>
<tr>
<td>2. SIGNPOSTS</td>
<td>5. GEODESICS</td>
</tr>
<tr>
<td>3. NORMAL COORDS</td>
<td>6. OPEN QUESTIONS</td>
</tr>
</tbody>
</table>
Data Structures for Intrinsic Triangulations

- A brief history of intrinsic data structures:
  - **Basic data structure** (connectivity + lengths)
  - **Overlay data structure** [Fisher et al 2006]
  - **Signpost data structure** [Sharp et al 2019]
  - **Normal coordinates + roundabouts** [Gillespie et al 2021]
Basic Data Structure

- Minimal encoding of intrinsic triangulation:
  1. List of $|E|$ edge lengths
  2. Some kind of topological mesh data structure

- Must be able to describe any $\Delta$-complex
  - ✔ halfedge mesh, signed incidence matrices
  - ✗ vertex-face adjacency list

- Example: $\{\{i,i,i\}, \{i,i,i\}\}$ is not a valid simplicial complex

Limitation: doesn’t encode correspondence between original & new triangulation.
Evaluating Geometric Quantities

- Can compute basic geometric quantities directly from lengths
- Cost, accuracy comparable to extrinsic expressions
- Note: using vertex positions generally gives wrong result!
- Can still think of embedding \( f : V \to \mathbb{R}^3 \) as “signal” on surface
  - Mixing intrinsic & extrinsic can be quite useful—e.g., **extrinsic** mean curvature vector via **intrinsic** Laplacian

\[
\theta_{jk}^i = \arccos \left( \frac{\ell_{ij}^2 + \ell_{ik}^2 - \ell_{ij}^2}{2\ell_{ij}\ell_{ik}} \right)
\]

\[
\text{area} = \sqrt{s(s - \ell_{ij})(s - \ell_{jk})(s - \ell_{ki})}
\]

\[
s := \frac{1}{2}(\ell_{ij} + \ell_{jk} + \ell_{ki})
\]
Implementing Intrinsic Edge Flip

• To flip edge $ij$:
  1. compute $\ell_{kl}$ from known lengths
  2. replace triangles $ijk, jil$ with $ilk, ljk$

• Only valid if
  – triangles form a convex quadrilateral
  – both endpoints have degree $\geq 2$
Overlay Data Structure

- Ok, but what if we want correspondence between triangulations?
- (Fisher et al. 2006) Track where original & new edges cross
  - explicitly store common subdivision (e.g., via halfedge)
  - edge segments marked as original, new, or both
  - crossing locations stored as 1D coordinates

**Pros:**
- provides correspondence ⇒ interpolation
- guaranteed to be topologically valid

**Cons:**
- only supports edge flips (& not constant time)
- expensive to compute/store crossings

\[ O(n^2) \text{ crossings} \]
Signpost Data Structure

Basic idea: implicitly encode triangulation via direction & distance to neighbors
Signpost Data Structure

- Signpost data structure consists of:
  - A topological triangulation $K = (V, E, F)$
  - A discrete metric $\ell : E \rightarrow \mathbb{R}^3$
  - **Outgoing edge directions** $\varphi : H \rightarrow [-\pi, \pi)$
    - $H$ is the set of oriented edges or halfedges
  - Possibly: barycentric coordinates of new vertices

- **Note:** storage cost is now fixed; crossings can be lazily evaluated “on-demand”
  - …but are almost never actually needed!

**Key idea:** maintain description of tangent spaces, not just metric
Encoding Edge Directions

• Can encode a unit direction $X$ at a vertex as angle $\phi$ relative to some fixed reference direction $e$

• Angle $\phi \in [-\pi, \pi)$ describes angle around intrinsic cone

• E.g., to get direction of each outgoing edge:
  – normalize interior angles so they sum to $2\pi$
  – take cumulative sums of interior angles

• Reference direction stays fixed for all time (even after doing edge flips)

\[
\begin{align*}
\Theta_i &:= \sum_{ijk} \theta_{jk}^i \\
\tilde{\theta}_i^k &:= 2\pi \frac{\theta_{jk}^i}{\Theta_i} \\
\phi_{ija} &:= \sum_{n=0}^{a-1} \tilde{\theta}_{jn+1}^i
\end{align*}
\]

angle sum around $i$
normalized interior angles
edge directions


Signpost Data Structure—Tracing Query

• Can trace intrinsic edges across extrinsic mesh (and vice versa) “on demand” by evaluating discrete exponential map

• Given an edge direction $X$ and length $\ell_{ij}$ at a vertex $i$, just “walk along surface”
  – computation: 2D ray-edge intersections
Signpost Data Structure—Edge Flip

- Edge flip same as before, but we also have to update signpost angles
- Easily computed from available data (lengths $\Rightarrow$ new lengths $\Rightarrow$ new angles)

\[
\varphi_{lk} \leftarrow \varphi_{lj} + \tilde{\theta}_{jk}^l \\
\varphi_{kl} \leftarrow \varphi_{ki} + \tilde{\theta}_{il}^k
\]

[Sharp et al 2018]
Signpost Data Structure—Vertex Insertion

- In contrast to earlier data structures, can also insert new vertices.
- Need tangent info/tracing to make this happen.
- **Basic idea**: given new **intrinsic** vertex $i$, trace along **extrinsic** mesh until we find the extrinsic triangle $abc$ containing $i$
  - record barycentric coordinates of new point.
  - insert new edges/faces in intrinsic mesh.
  - get signpost angles by adding interior angles of new triangles to incoming direction.
Signpost Data Structure—Many Other Operations

From there, many other operations—same basic pattern: update connectivity, edge lengths, and signpost angles via tracing & local arithmetic on known values.

**Key point:** full-blown intrinsic data structure, more “feature compatible” w/ standard mesh
Signpost Data Structure—Encapsulation

• Complexity hidden by simple interface

• Geometric queries:
  edge.length()
  triangle.area()
  corner.angle()
  ...

• Local remeshing:
  flipEdge( ij )
  insertVertex( ijk, p )
  moveVertex( i, v )
  ...

• Algorithms then implemented as usual

```java
// perform local remeshing
while( !done ) {
  done = false;
  for( Edge ij : mesh.edges() )
    if( !ij.isDelaunay() ) {
      done = false;
      mesh.flipEdge( ij );
    }
}

// access geometric quantities
Matrix L, M;
for( Vertex i : mesh.vertices() ) {
  for( HalfEdge ij : i.halfedges() ) {
    double wij = ij.cotan() + ij.twin.cotan();
    L( i, i ) += wij / 2.;
    L( i, j ) -= wij / 2.;
    M( i, i ) = i.dualArea();
  }
}
```
签柱数据结构—覆盖提取

- 也可以提取“覆盖网格”，即原始和内禀网格的公共细分
  - 需要的频率比你想象的要低！
  - 如用于可视化—或有时用于发送数据“下游”
- 简易两步过程:
  1. 按示性网格覆盖所有内禀边；存储交叉点
  2. 在每个三角形上运行简单局部算法以输出细分
    - 只有两种情况需要处理
- 不支付维护交叉点的成本在中间操作期间

- 案例 I
- 案例 II
Comparison to Explicit Overlay

signpost: 0.60s  
[Fisher et al]: 17.32s  
speedup: 28x

signpost: 0.20s  
[Fisher et al]: 4.27s  
speedup: 21x

(Also: explicit overlay only supports flips…)

Numerical Issues

- Finite precision presents challenges for all intrinsic triangulation data structures

- Basic data structure:
  - flipping near-degenerate triangles may yield lengths that do not describe a metric

- Overlay data structure:
  - even though guaranteed to be topologically valid, can get wrong geometry

- Signpost data structure:
  - tracing from $i$ to $j$ can fail to reach neighborhood of vertex $j$

- In practice, all these issues are essentially nitpicks
- All data structures work great, even on truly terrible meshes
Signpost Data Structure—Robustness

- **Robustness test:** flip to Delaunay & extract common refinement

- Succeeds on 100% of “ordinary” meshes (e.g., Princeton shape benchmark, Myles & Zorin dataset)

- for worst 3% of meshes in Thingi10k (i.e., really awful meshes), needed to perturb near-zero area triangles, or use quadruple precision

- In general: works in all but most pathological/adversarial cases

...gives new meaning to “bad mesh”...
Intrinsic Retriangulation Algorithms
Intrinsic Retriangulation—Overview

• So far just have a data structure. How do we actually get good intrinsic meshes?

• **intrinsic Delaunay triangulation** (iDT)
  – already saw Lawson’s flip algorithm
  – makes mesh Delaunay, but nothing more
  – angle/area distribution can still be (very) bad

• **intrinsic Delaunay refinement** (iDR)
  – provides guaranteed bounds on angles
  – can also adapt to user-specified criteria

• **intrinsic optimal Delaunay triangulation** (iODT)
  – balance between element quality and vertex distribution
Intrinsic Delaunay Triangulation (iDT)

- **Lawson’s flipping algorithm:**
  - enqueue all edges
  - **while** queue is not empty:
    - pop an edge
    - flip it if it’s not Delaunay
    - enqueue neighbors if not already in queue
Intrinsic Delaunay Refinement (iDR)

- **Chew’s 2nd algorithm:**
  - Until min angle is above $\theta_{\text{min}}$:
    - flip to Delaunay (Lawson)
    - pick any “bad” triangle $ijk$
    - insert circumcenter $p$ of $ijk$
- What’s the intrinsic circumcenter?

![Graphs showing corner angles and corner counts for input and iDR](image)
Intrinsic Optimal Delaunay Triangulation (iODT)

- Unlike DR, allow vertices to move
- **Intrinsic**: non-cone vertices only!
- Until convergence:
  - split edges longer than $\ell_{\text{max}}$
  - move vertices to weighted average of circumcenters
  - flip to Delaunay (Lawson)

**Intrinsic**: use average vector to circumcenters
Intrinsic Delaunay Triangulation—Example
Intrinsic Delaunay Refinement—Examples

In practice: can always achieve $\theta_{\text{min}} = 30^\circ$

\[ \theta_{\text{min}}: 0.13^\circ \rightarrow 30^\circ \quad \theta_{\text{min}}: 2.3^\circ \rightarrow 30^\circ \quad \theta_{\text{min}}: 0.66^\circ \rightarrow 30^\circ \]
Intrinsic Optimal Delaunay — Examples
Comparison to Traditional Meshing

intrinsic Delaunay refinement

restricted Delaunay refinement [Engwirda 2018]

\[ |F| = 20k \]
\[ \ll 1 \text{ second} \]

\[ |F| = 330k \]
\[ 20 \text{ minutes} \]

Why? Extrinsic meshing must juggle element quality & geometric error…
Comparison to Extrinsic Delaunay

Delaunay: yes
$|V| = 5236$
$\theta_{\text{min}} = 0.30^\circ$

Delaunay: yes
$|V| = 360$
$\theta_{\text{min}} = 7.31^\circ$

Delaunay: yes
$|V| = 1766$
$\theta_{\text{min}} = 34.01^\circ$

[Liu et al. 2015] intrinsic flips intrinsic Delaunay refinement
Open Question: *Is Intrinsic Flipping Efficient?*

- Lawson’s algorithm in 2D: $O(n^2)$ flips

- Impossible to bound number of flips in terms of mesh size. [Bobenko & Springborn 2005]
  - Flip graph does not have bounded diameter!

- **Surprising observation:** [Sharp et al 2019] empirically, number of flips grows roughly like # of edges, even on “crazy” data like Thingi10k
  - Why? Intuition: longest edge of embedded mesh is bounded by extrinsic diameter
  - Goal: bound number of flips in terms of geometry, rather than number of elements.
Intrinsic Geometry Processing
How to Make a Round Trip

From easiest to hardest:

• Don’t make a round trip
  – *example*: eigenvalue computation
• Copy values at vertices
  – *example*: geodesic distance
• Get solution value at a point
  – *example*: Karcher mean
• Flip back to original triangulation
  – *example*: injective mapping
• Build explicit overlay mesh
  – *example*: visualization of intrinsic solution
Application—Finite Element Problems

- Still have triangle mesh → can still use ordinary **linear** basis functions
  - No change to existing solver code! Just build usual mass & stiffness matrix
  - Same matrix density, performance, guarantees, *etc.*, **but** better accuracy/stability
- If desired, can still use any other (*e.g.*, higher-order) triangular elements
- Example: solve Poisson equation $\Delta u = f$

$u$  $f$  extrinsic ODT  intrinsic ODT

~2x more accurate
Application—Geodesic Distance

(Crane et al 2013) Heat method to compute geodesic distance via two Poisson-like equations

\[ |V| = 28010 \]
\[ \text{mean error: 59.6\%} \]

\[ |V| = 28010 \]
\[ \text{mean error: 20.4\%} \]

\[ |V| = 52513 \]
\[ \text{mean error: 0.7\%} \]

<i>Heat Method</i>

<2x more elements>
Adaptive Mesh Refinement (AMR)

• For the first time, apply *a posteriori* refinement to surface meshes (Nochetto 2005)
• Focus computational effort only where it’s needed (*can’t* be determined *a priori*)

18–54x smaller mesh
2.5–10x speedup
(for comparable accuracy)
Injective Surface Parameterization

- Compute harmonic map from surface to disk
- intrinsic Delaunay property avoids “flipped” triangles; AMR improves accuracy
Tangent Vector Field Processing

- Since signpost data structure maintains correspondence between tangent spaces, can use intrinsic triangulations to improve vector field processing

- **Flip-free property** [1]: For iDT, minimizer $X$ of vector Dirichlet energy will be "flip free"
  - i.e., if $\Delta^V X = 0$ (where $\Delta^V$ is the connection Laplacian) each vector is in convex cone of parallel-transported neighbors

- Helps prevent foldover in field-aligned parameterization / meshing

Flip-Free Vector Fields

input mesh

intrinsic Delaunay triangulation
Logarithmic Map

- Flip-free vector fields help avoid degeneracy in derived quantities (e.g., log map)

(Sharp, Soliman, Crane 2018)
Globally Optimal Direction Fields

- Intrinsic triangulation/refinement provides even smaller Dirichlet energy $E_D$ (smoother) than smallest Dirichlet energy on original triangulation

\[ E_D = 1200.2 \quad E_D = 123.5 \quad E_D = 47.0 \]
Open Question: Vector Rippa Theorem?

- **Rippa’s theorem.** Given function at vertices of a triangulation, Delaunay triangulation gives “smoothest” interpolation, i.e., minimizes Dirichlet energy of linear interpolation.

- Analogous theorem for tangent vectors at vertices?

- Many notions of discrete vector Dirichlet energy
  1. (Fisher et al 2007) 1-forms
  2. (Knoppel 2013) Finite elements
  3. (Knoppel 2015) Finite differences
  4. (Sharp et al 2018) Weitzenböck identity
  5. (Stein et al 2020) nonconforming finite elements

- Have found counterexamples for (2) and (3)!

(In practice: very close to smoothest)
Data Structures II: Normal Coordinates
Normal Coordinates for Intrinsic Triangulations

- Already saw two data structures for tracking correspondence:
  - **Overlay** *(explicit)* — store common refinement in mesh data structure. Guarantees correct combinatorics, but expensive to explicitly track every single local operation.
  - **Signpost** *(implicit)* — store only the direction & distance to each neighbor at each vertex. Output-sensitive and cheap to update, but can fail due to floating point error.

- Normal coordinates offer best of both worlds:
  - **exact combinatorics** — purely integer data encodes how two triangulations cross
  - **implicit** — don’t explicitly store common refinement; updated via simple local rule
  - **output sensitive** — work required to extract final refinement (by “tracing” edges) is proportional to size of output
Normal Coordinates for Curves

Basic idea: simply count how many times a curve crosses each edge of a triangulation.

(All other edges have normal coordinate 0.)

Important restriction (normal curve): not allowed to enter & exit same edge of triangle
Normal Coordinates—History & Perspectives

• Originally developed for normal surface theory

• Several different uses/perspectives:
  
  • topology — used to study, e.g., the mapping class group (among many other things…)
  
  • computer science — “compressed” representation of curves: can encode extremely long curve with far fewer bits than needed to explicitly store curve segments
  
  • geometry processing — provide robust data structure for tracking intrinsic triangulations

Think geometrically: will always imagine normal coordinates encode geodesic curves.
Normal Coordinates & Triangle Inequality

For a curve not passing through vertices, normal coordinates satisfy triangle inequality:

\[ n_{ij} + n_{jk} \geq n_{ki} \]

**Basic idea:** at most, arcs that go into the first two edges can come out the third edge.
In general, will allow curves to stop/start at vertices. Two useful numbers:

**edges leaving corner** $k$

$$e_{k}^{ij} = \max \left( 0, n_{ij} - n_{jk} - n_{ki} \right)$$

**edges crossing corner** $k$

$$c_{k}^{ij} = \frac{1}{2} \left( \max \left( 0, n_{jk} + n_{ki} - n_{ij} \right) - e_{i}^{jk} - e_{j}^{ki} \right)$$
Using this data, not hard to trace out sequence of crossed edges...

Basic question: go left, go right, or stop? (+which crossing)
Straightening Traced Path

• So far, path is just a sequence of edges

• To get geometry of path:
  – unfold corresponding triangle strip into plane
  – draw straight line between endpoints
  – find intersections with each edge as coordinates $s, t \in [0,1]

• Guaranteed to get right edge sequence

• Only possible error (floating point): line-line intersection

• Clamping to $[0,1]$ still gives path with right combinatorics

Compare: with signposts, trace could (in principle) fail to even enter final triangle…
Intrinsic Triangulations & Normal Coordinates

- So far we have been using normal coordinates to encode a single curve.

- **Observation 1**: union of disjoint normal curves represented by sum of normal coordinates.

- **Observation 2**: edges of an intrinsic triangulation are disjoint geodesic arcs (hence, normal curves).

- Can therefore encode entire triangulation with a single set of normal coordinates.
Edge Flip Formula for Normal Coordinates

- As with edge lengths, normal coordinates resulting from an edge flip can be updated using only values on the immediate neighbors.

- Especially simple if no curve terminates at a vertex.
  - “tropical” version of Ptolemy’s relation for cyclic quadrilateral.

- Can generalize this formula if we assume normal coordinates encode a triangulation.
Roundabouts

• For tracking correspondence of triangulations, normal coordinates are not enough!
  – after performing flips, may have two intrinsic edges with same endpoints

• Problem: how do you identify traced curve with logical edge of extrinsic mesh?

• Solution [Gillespie et al 2021]: “roundabouts”
  – at each vertex of extrinsic mesh, enumerate outgoing edges in counter-clockwise order.
  – each directed edge of intrinsic mesh then stores index of next outgoing extrinsic edge.
  – analogy: roundabouts on roadways

Notice: roundabouts are “combinatorial analogue” of signposts (quantized direction)
Common Refinement

- Common refinement of extrinsic & intrinsic mesh is needed for, e.g., texture mapping
- After tracing out edges, can recover the common refinement of the extrinsic + intrinsic meshes via procedure similar to one used for signposts

Connectivity always correct: ordering of intersections known from tracing (combinatorial).
Application: Discrete Conformal Mapping

• Powerful tool in geometry: **uniformization**
  – (near-)canonical map to constant-curvature domain
• Recent breakthrough: **discrete uniformization theorem**
  – *every* polyhedral mesh—*no matter how bad*—can be uniformized in exact sense
• Long history with many contributors (*incomplete list*):
  – Luo, Bobenko, Pinkall, Springborn, Gu, Guo, Sun, Wu, Yau…
  – *(hyperbolic realization)* Rivin, Fillastre, …

![Diagram](image-url)
Application: Discrete Conformal Mapping

- New paper [1]:
  1. robust scheme for correspondence (normal + roundabouts)
  2. simple interpolation scheme via layout in light cone
  3. handles genus-0 case (convex embedding in 2-sphere)
  4. simplify optimization: perform Ptolemy flips at any moment
- Ptolemy flips seem to be “right” computational approach [2]

Discrete Uniformization—Numerical Robustness

- Not enough to compute intrinsic description of flattened surface (scale factors) — also need to extract map
- **In theory**, no big deal:
  - construct common subdivision of three triangulations (input, intrinsic Delaunay, uniformized)
  - compute projective texture coordinates at vertices of common subdivision
- **In practice**: can’t escape floating-point!
  - proofs of correctness assume real numbers…
  - have three intersecting triangulations on “bad mesh”…!

signposts  normal coordinates

THE DIFFERENCE BETWEEN PRACTICE AND THEORY:

IN THEORY, THERE IS NO DIFFERENCE, IN PRACTICE, THERE IS
Discrete Conformal Mapping — Examples

Succeeds on all cone configurations from Myles-Pietroni-Zorin (MPZ).

Extracts common refinement on 97.7% of 32,744 connected components from Thingi10k.
Thanks!

**Discrete Differential Geometry:**
**An Applied Introduction**

Keenan Crane • CMU 15-458/858