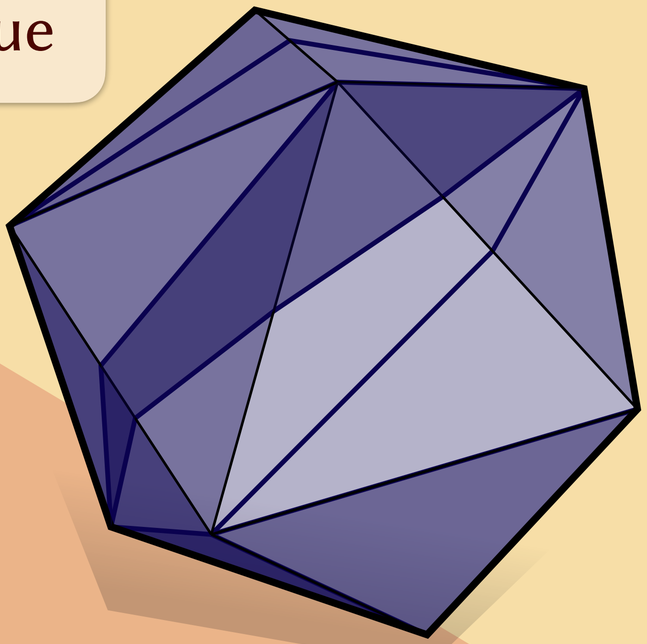
CGAL
DEVELOPER MEETING

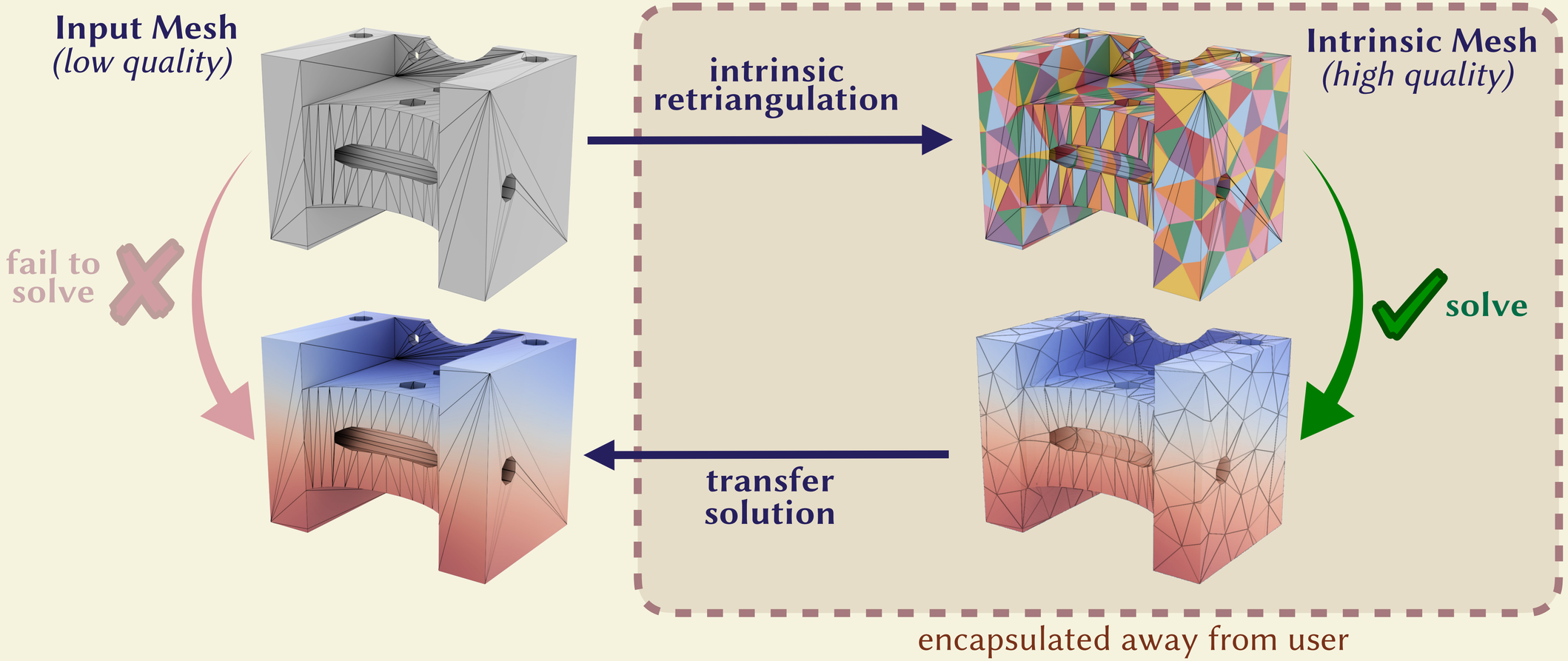
Geometry Processing with Intrinsic Triangulations

Mark Gillespie, INRIA / École Polytechnique

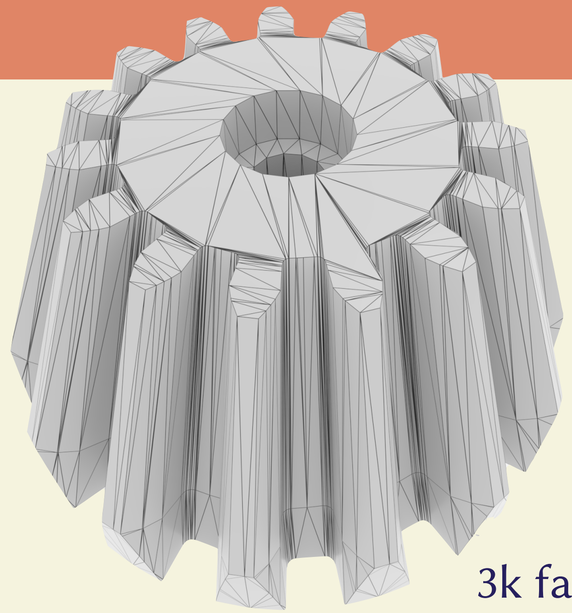


Wed, 16 April 2025

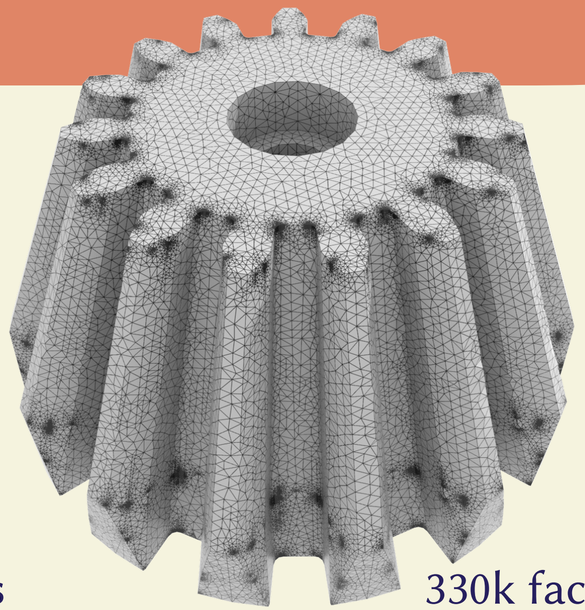
Working with low-quality triangle meshes



Trade offs of extrinsic remeshing



3k faces



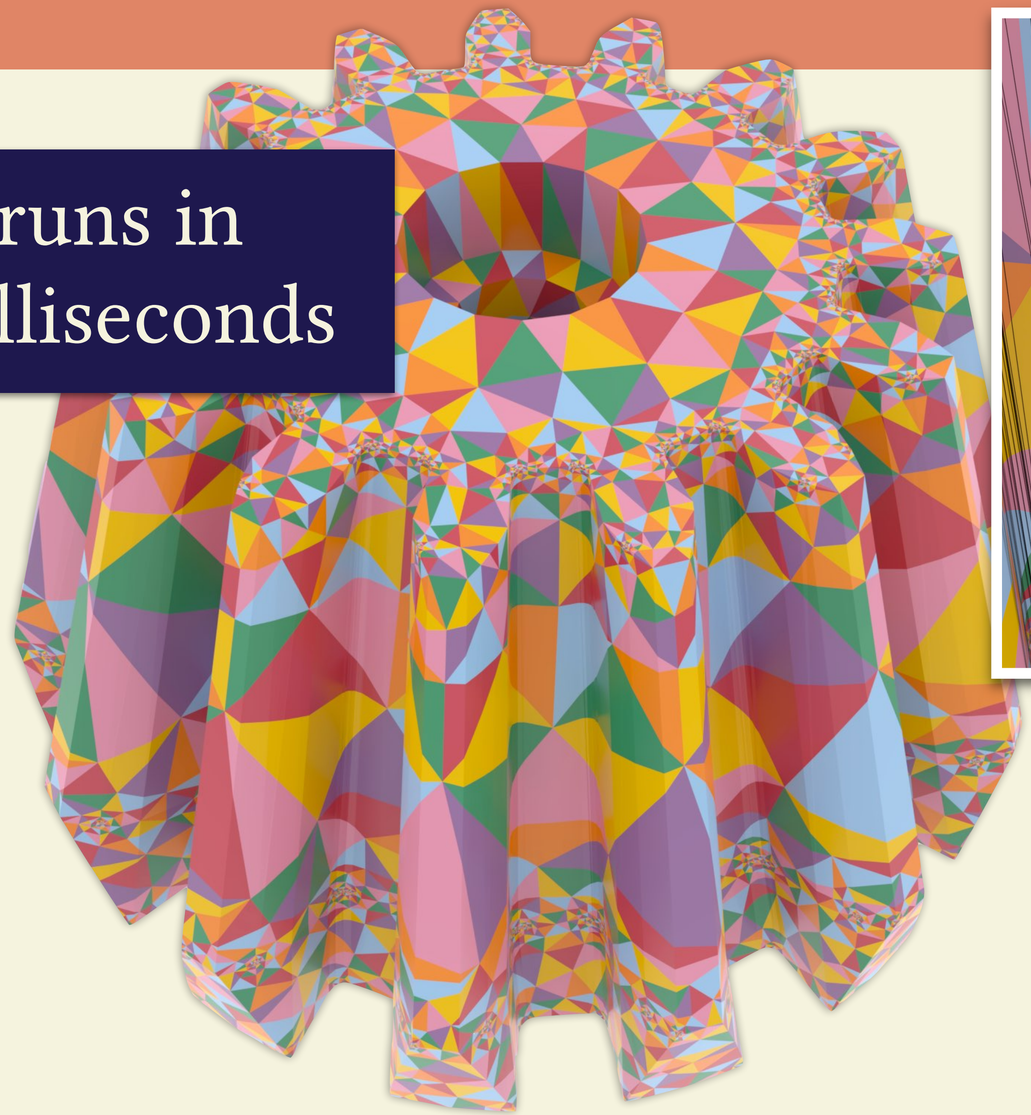
330k faces

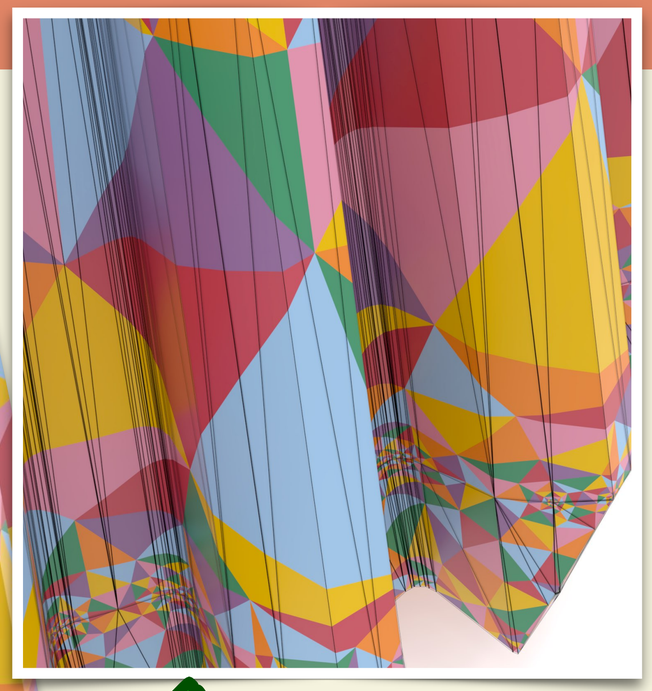
3k faces

triangle quality	✗	✓	✓
mesh size	✓	✗	✓
geometric fidelity	✓	✓	✗

Intrinsic triangulations sidestep the trade offs

runs in milliseconds

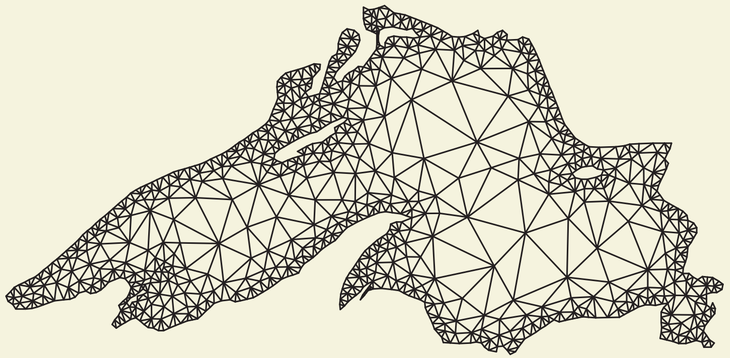




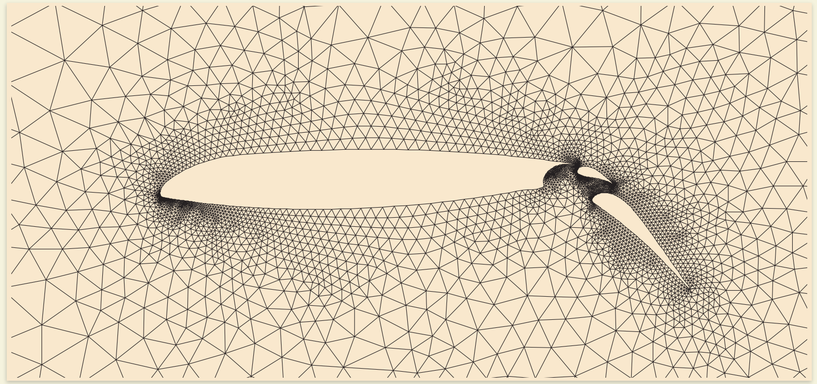
- ✓ high triangle quality
- ✓ exact same geometry
- ✓ without too many more triangles

Extending 2D computational geometry to surfaces

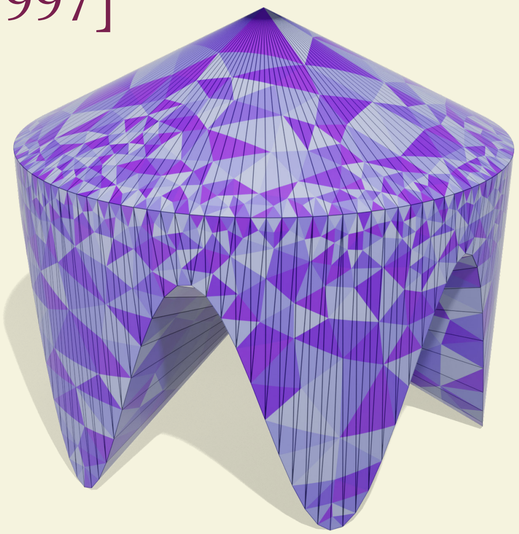
Delaunay refinement [Chew 1993; Shewchuk 1997]



[Shewchuk 1997]



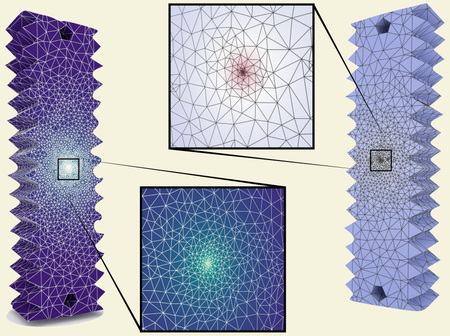
[Shewchuk 1997]



[Sharp *et al.* 2019]

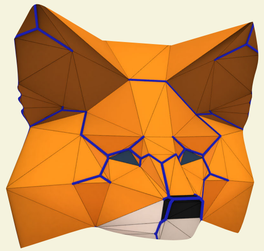
Optimal Delaunay triangulation [Chen & Xu 2004]

[Sharp *et al.* 2019]



Adaptive mesh refinement

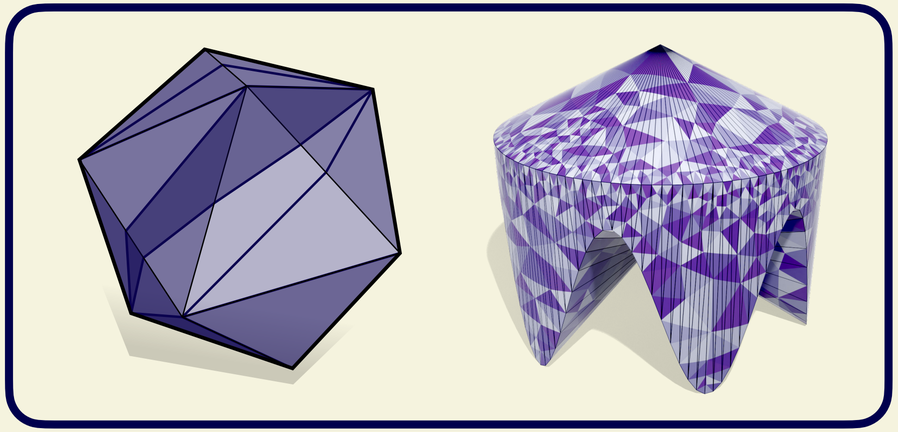
[Sharp *et al.* 2019]



Steiner tree approximation [Smith *et al.* 1981]

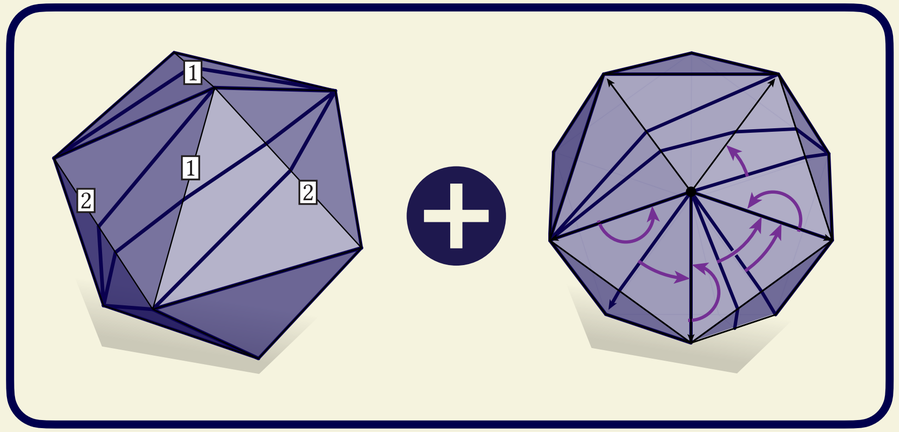
Today's Talk

I. Intrinsic Geometry Processing



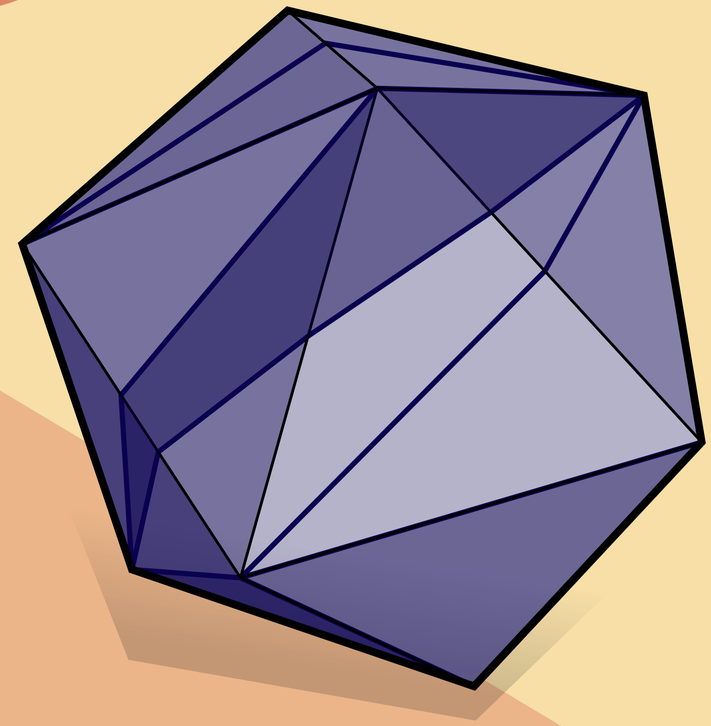
- intrinsic triangulations
- intrinsic Delaunay triangulation
- intrinsic Delaunay refinement

II. Data Structures

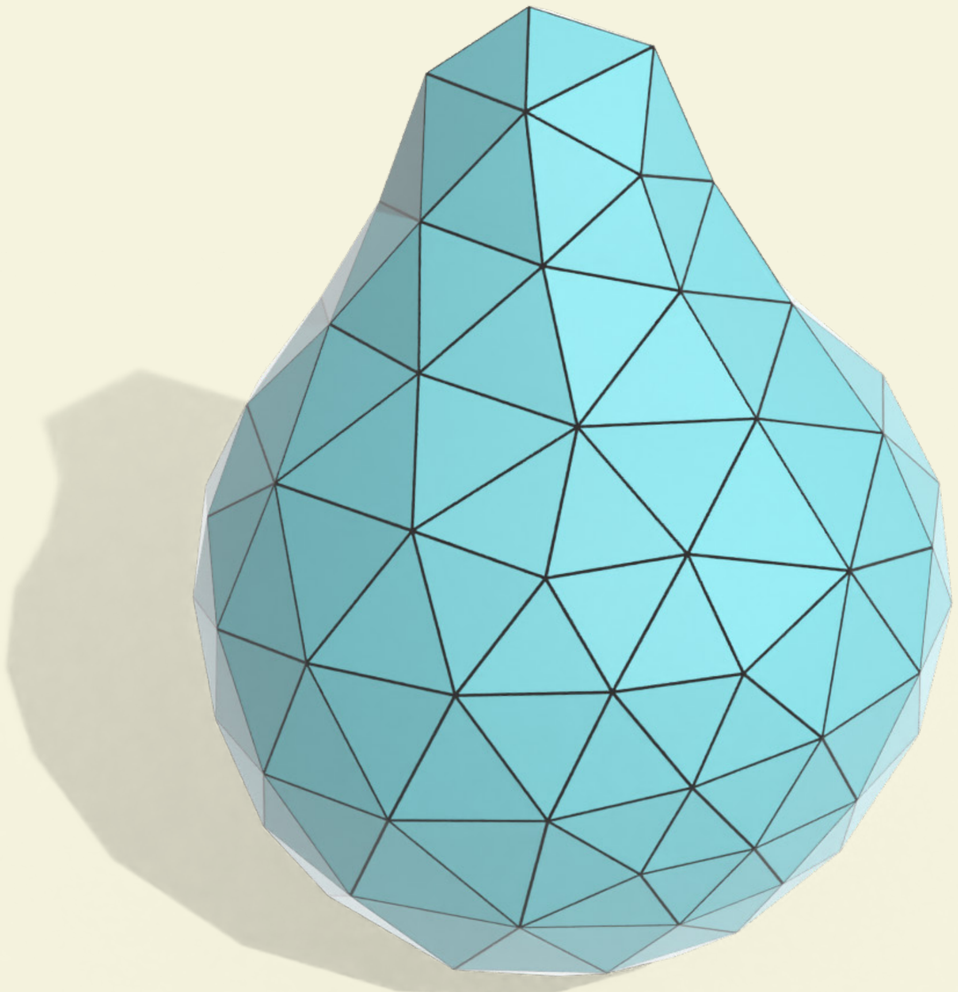


integer coordinates

1 Intrinsic Geometry Processing



Triangle meshes



Data Structure

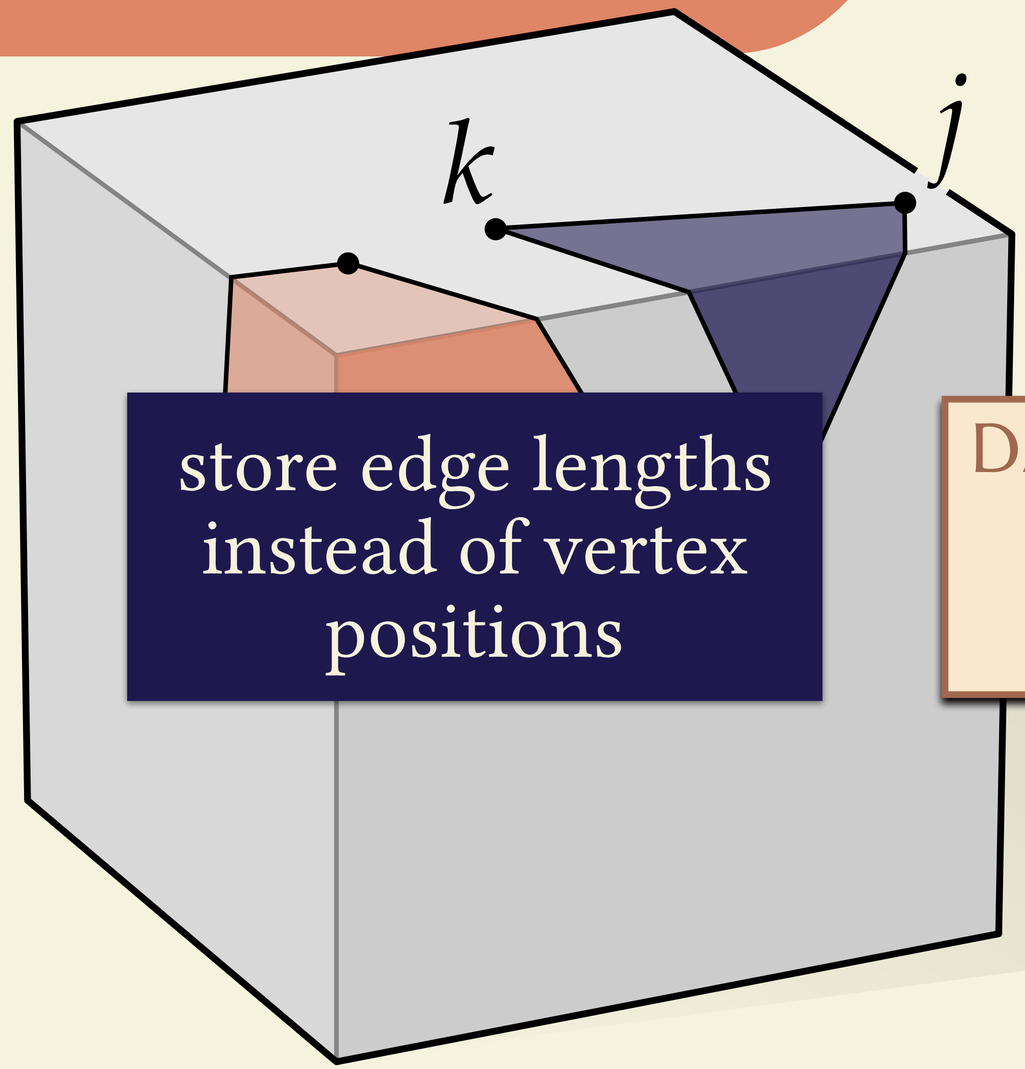
Position Data

```
vertices = {
    ( 1.2, 1.4, 0.7 ),
    ( 1.3,-2.1, 4.7 ),
        . . .
}
```

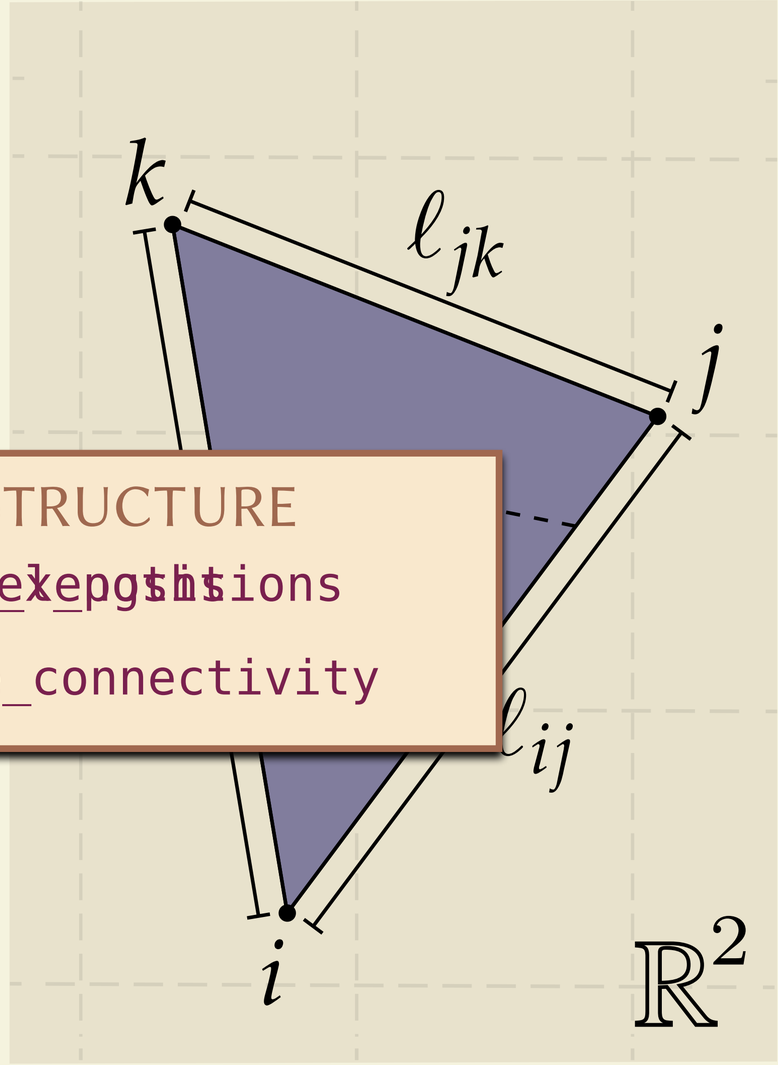
Connectivity Data

```
faces = {
    ( 0, 1, 2 ),
    ( 2, 1, 3 ),
    ( 5, 8, 9 ),
        . . .
}
```

Intrinsic triangles



store edge lengths instead of vertex positions



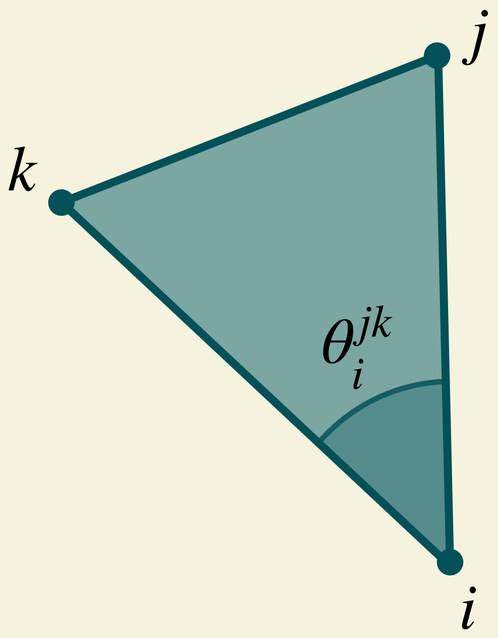
DATA STRUCTURE

~~vertex_positions~~

~~edge_lengths~~

face_connectivity

Performing geometric calculations



Face Area

Extrinsic setting

Given: vertex positions $p_i, p_j, p_k \in \mathbb{R}^3$

$$A_{ijk}^2 = \frac{1}{4}\left\langle \left(p_j - p_i\right) \times \left(p_k - p_i\right), \left(p_j - p_i\right) \times \left(p_k - p_i\right) \right\rangle$$

Intrinsic setting

Given: edge lengths $\ell_{ij}, \ell_{jk}, \ell_{ki} \in \mathbb{R}_{>0}$

$$A_{ijk}^2 = \frac{1}{16}\left(\ell_{ij} + \ell_{jk} + \ell_{ki}\right)\left(-\ell_{ij} + \ell_{jk} + \ell_{ki}\right)\left(\ell_{ij} - \ell_{jk} + \ell_{ki}\right)\left(\ell_{ij} + \ell_{jk} - \ell_{ki}\right)$$

Corner angle

Extrinsic setting

Given: vertex positions $p_i, p_j, p_k \in \mathbb{R}^3$

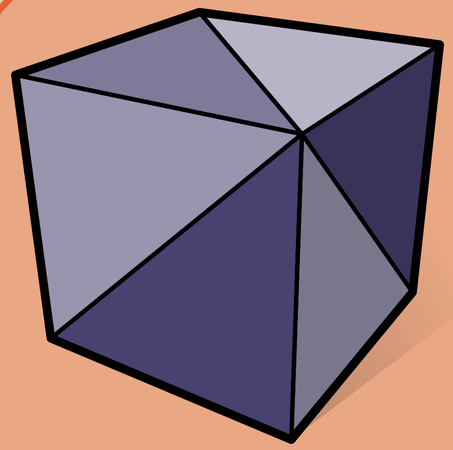
$$\cos\theta_i^{jk} = \left\langle p_j - pi, p_k - p_i \right\rangle \Big/ \|p_j - p_i\| \, \|p_k - p_i\|$$

Intrinsic setting

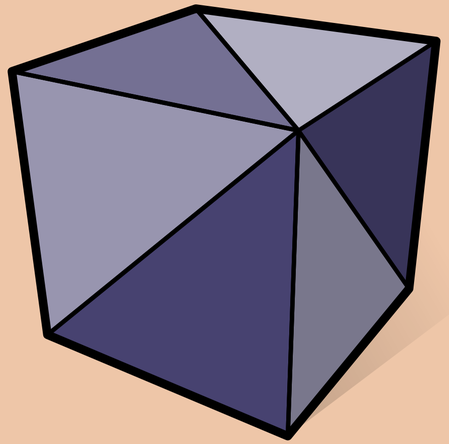
Given: edge lengths $\ell_{ij}, \ell_{jk}, \ell_{ki} \in \mathbb{R}_{>0}$

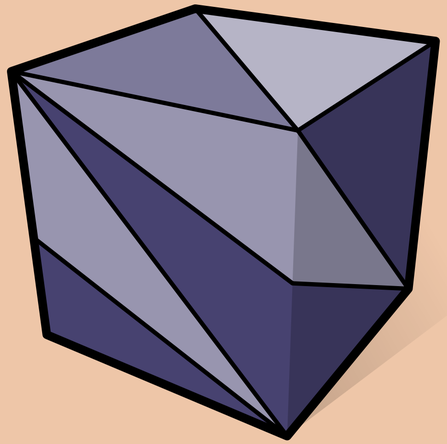
$$\cos\theta_i^{jk} = \frac{\ell_{ij}^2 - \ell_{jk}^2 + \ell_{ki}^2}{2\ell_{ij}\ell_{ki}}$$

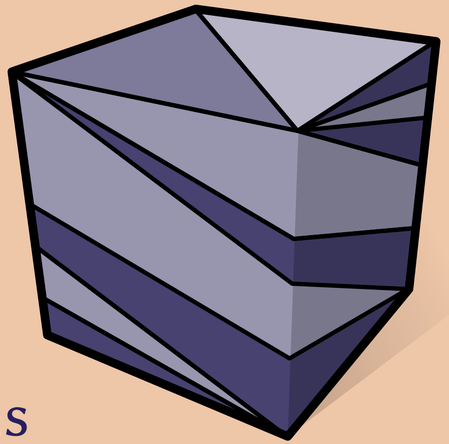
The space of intrinsic triangulations is large

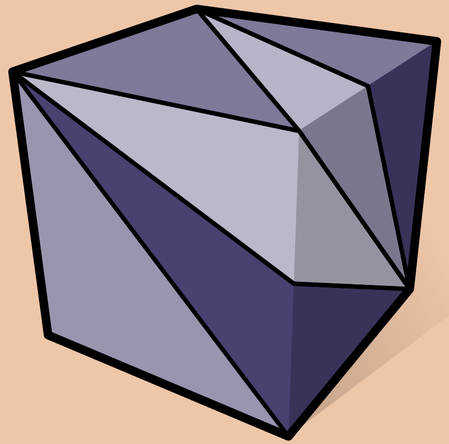


extrinsic triangle meshes







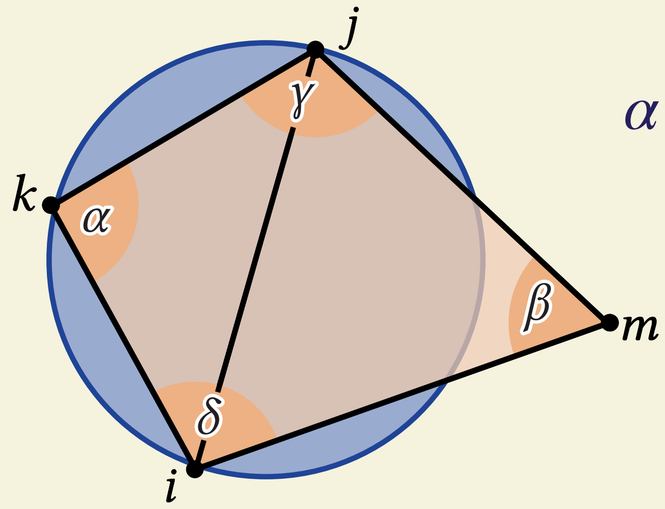


...

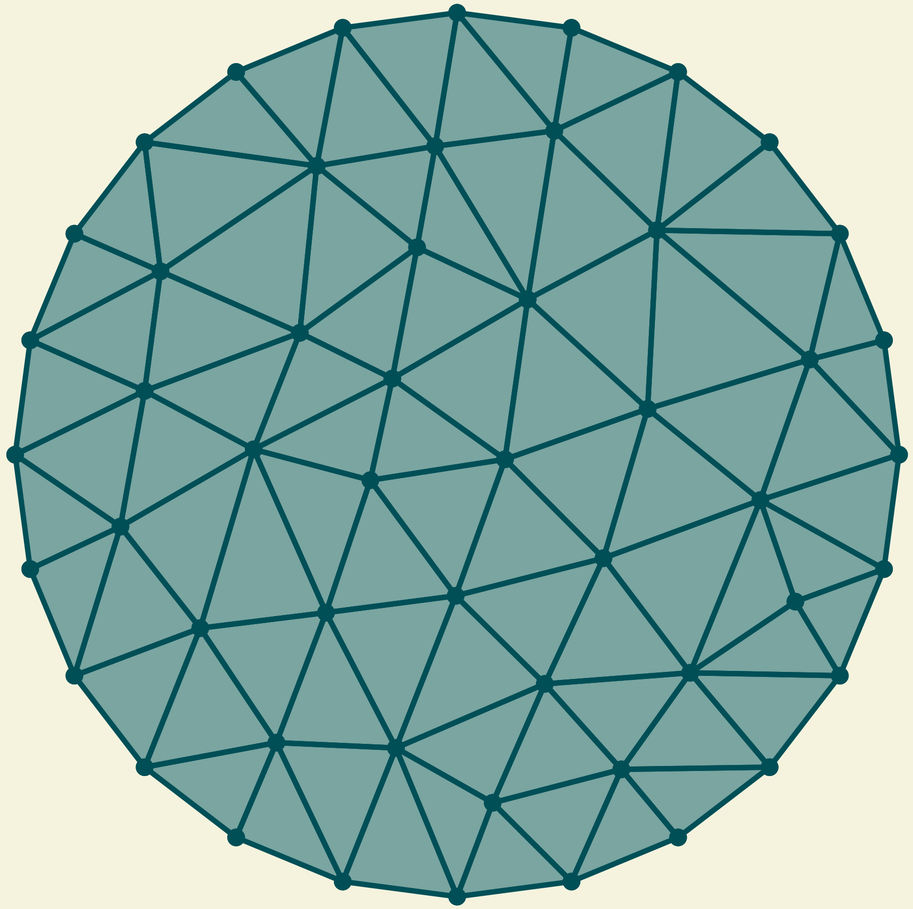
intrinsic triangulations

Delaunay triangulations

- Countless useful properties:
  - Maximize angles lexicographically, minimize spectrum lexicographically, smoothest interpolation, positive cotan weights…
- Key to successful 2D remeshing algorithms
- Characterized by *empty circumcircle condition*

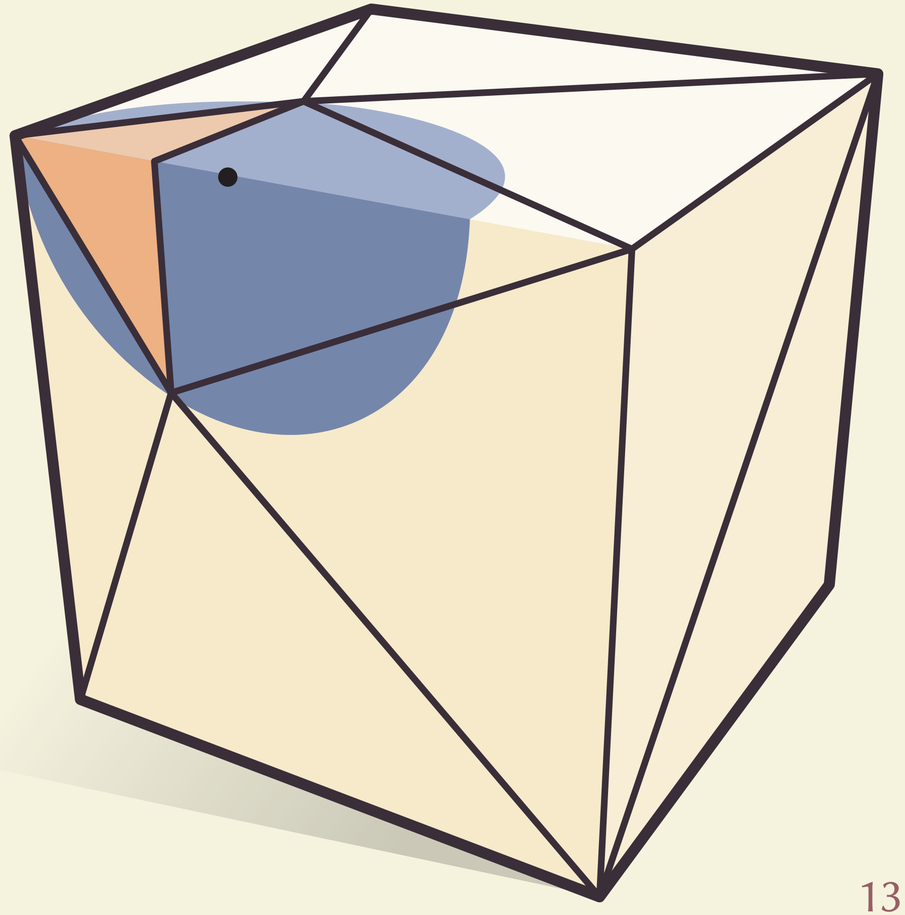


$$\alpha + \beta \leq \gamma + \delta$$



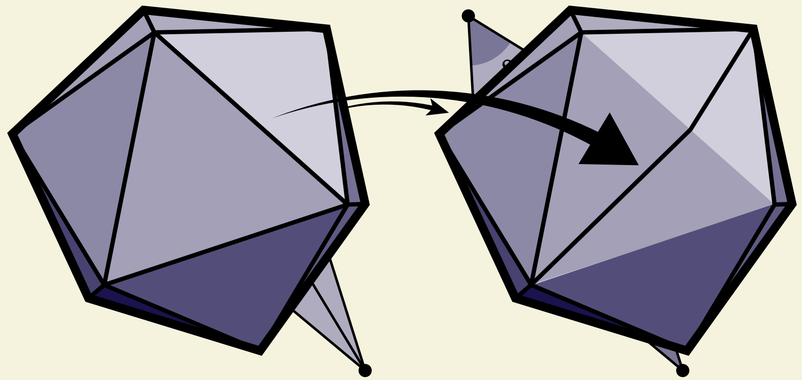
Intrinsic Delaunay triangulations

- [Masur & Smillie 1991, Rivin 1994, Indermitte *et al.* 2001, Bobenko & Springborn 2007]: intrinsic Delaunay
  - Maintain useful mathematical properties.
    [Sharp, **Gillespie** & Crane 2021]

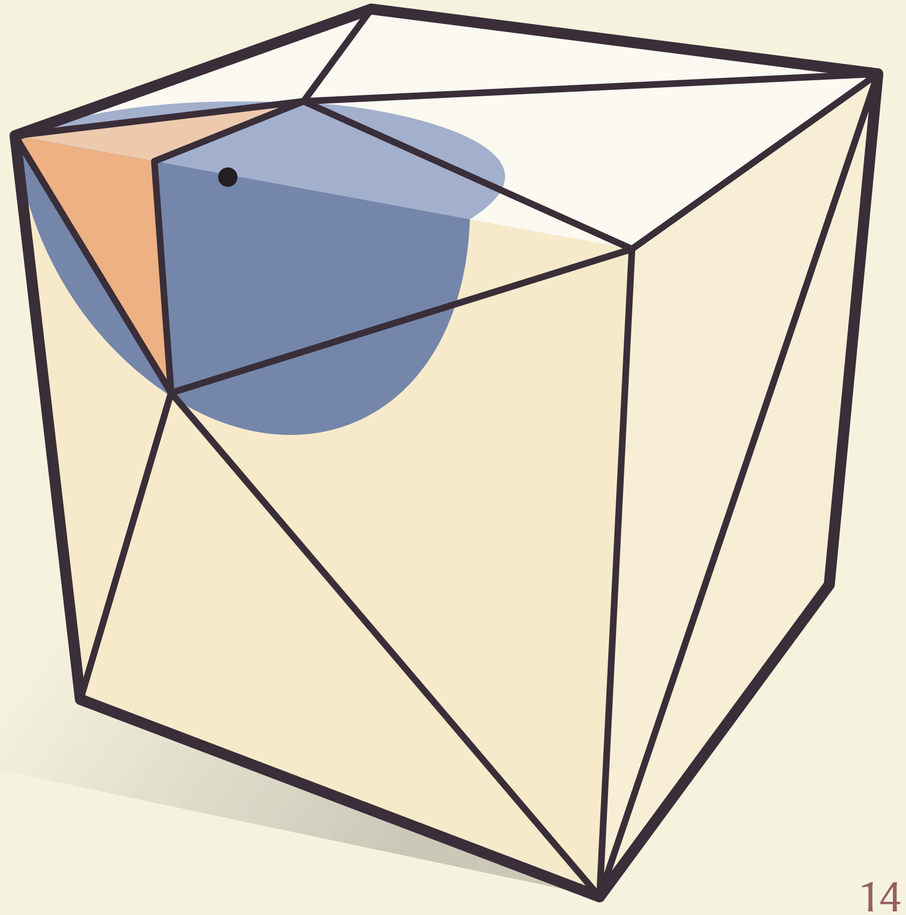


Intrinsic Delaunay triangulations

- [Masur & Smillie 1991, Rivin 1994, Indermitte *et al.* 2001, Bobenko & Springborn 2007]: intrinsic Delaunay
  - Maintain useful mathematical properties.
    [Sharp, **Gillespie** & Crane 2021]
- Compute by a simple algorithm:
  - Flip any non-Delaunay edge until none remain

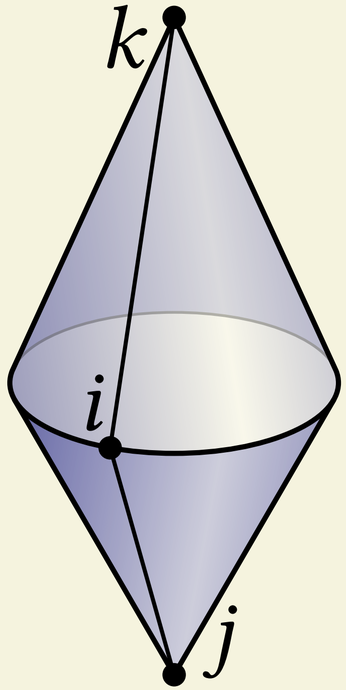


Faster than reading
the mesh from disk



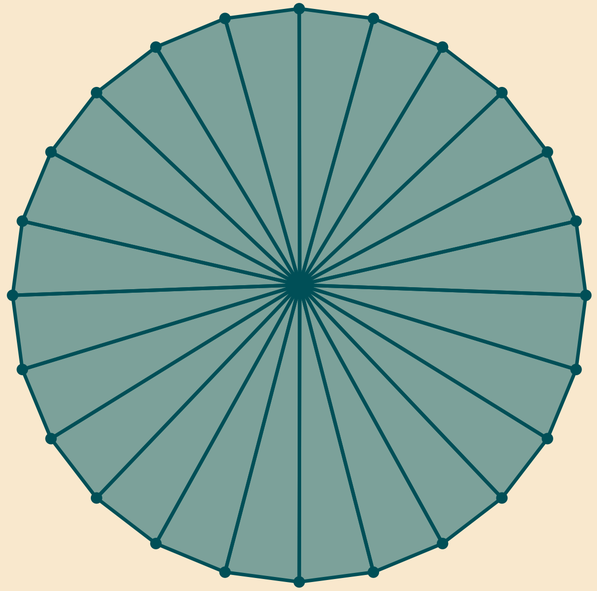
Intrinsic Delaunay triangulations

May be *irregular* (*e.g.*, two edges of a face may be glued together)



not a *simplicial complex*

Problem: cannot guarantee high triangle quality

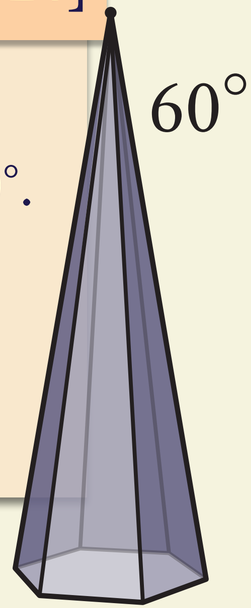


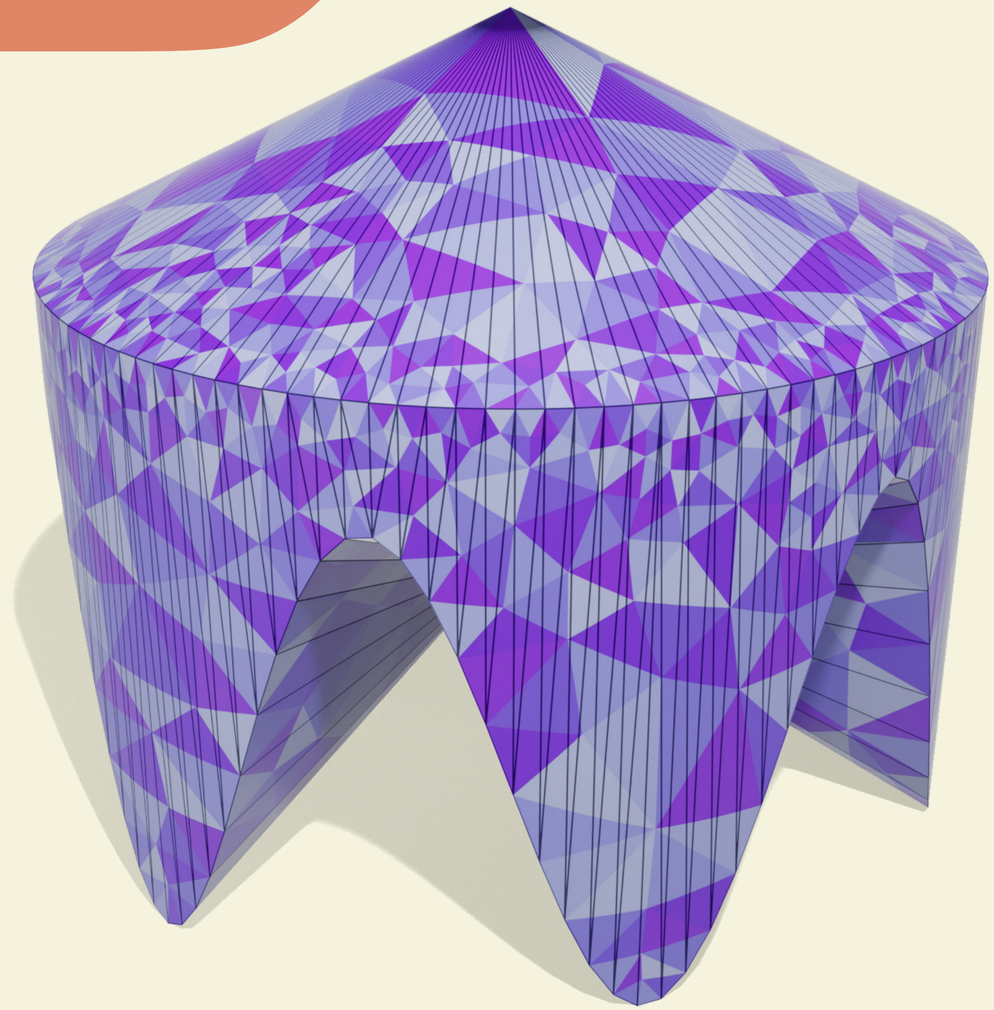
low quality Delaunay triangulation

Intrinsic Delaunay refinement

Theorem [**Gillespie**, Sharp & Crane 2021]

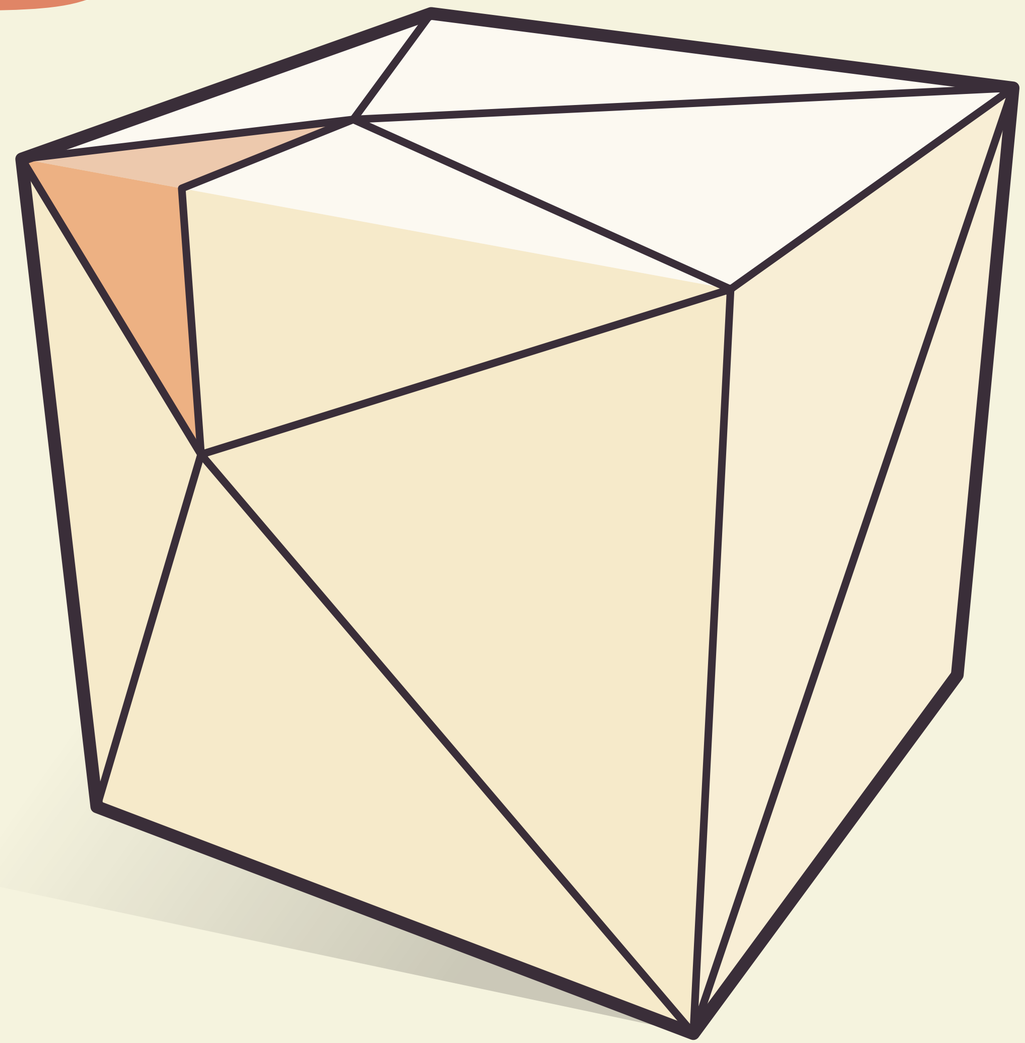
Let M be a mesh without boundary whose cone angles are all at least 60°. Then intrinsic Delaunay refinement produces a Delaunay mesh with triangle corner angles at least 30°





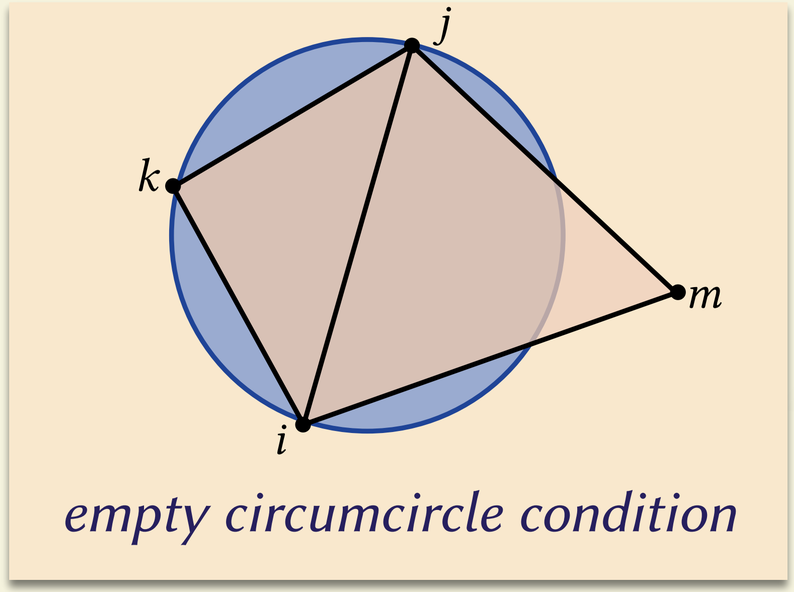
Intrinsic Delaunay refinement — algorithm

1. Pick any triangle violating angle bound

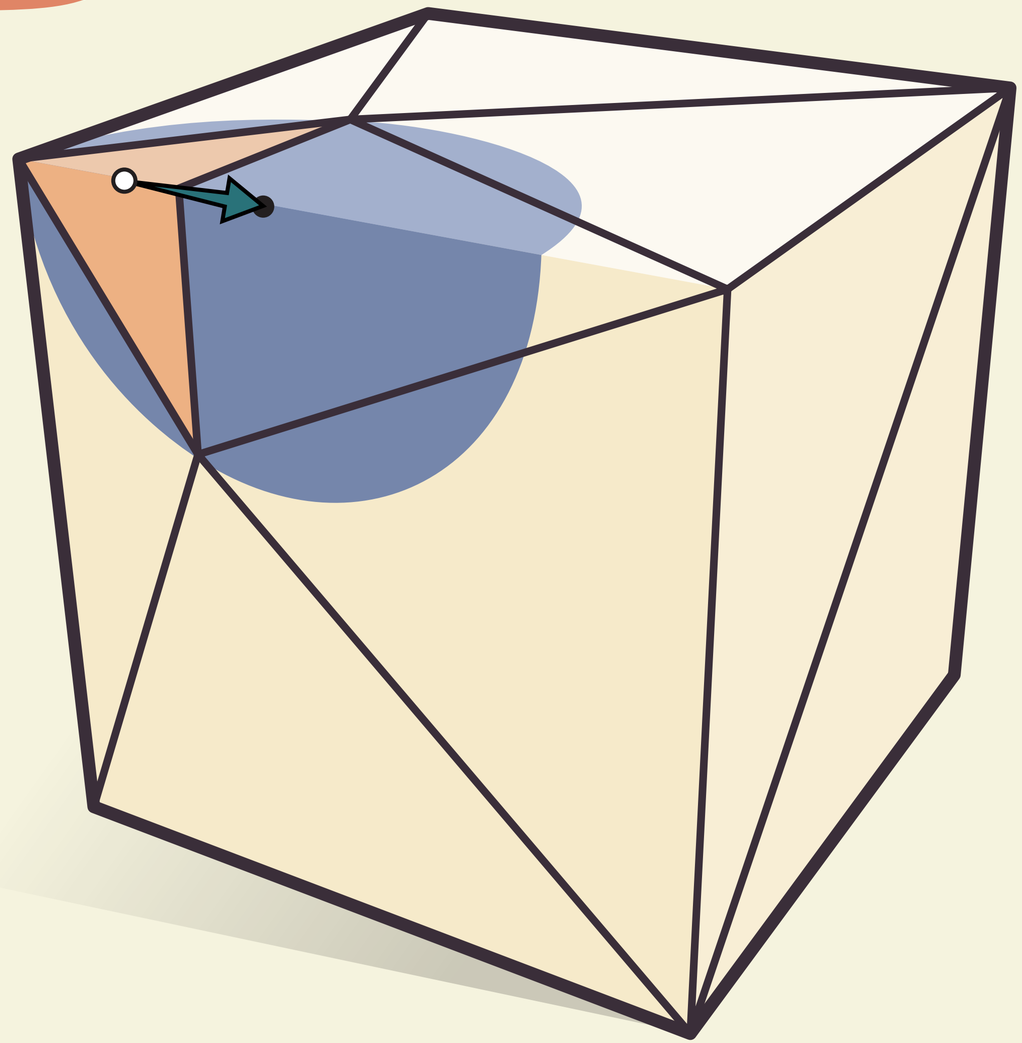


Intrinsic Delaunay refinement — algorithm

1. Pick any triangle violating angle bound
2. Locate its circumcenter

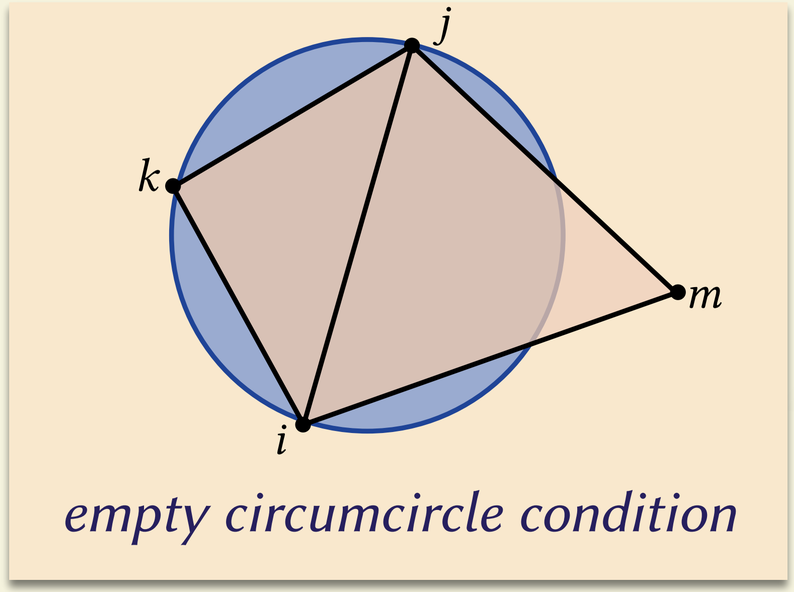


empty circumcircle condition

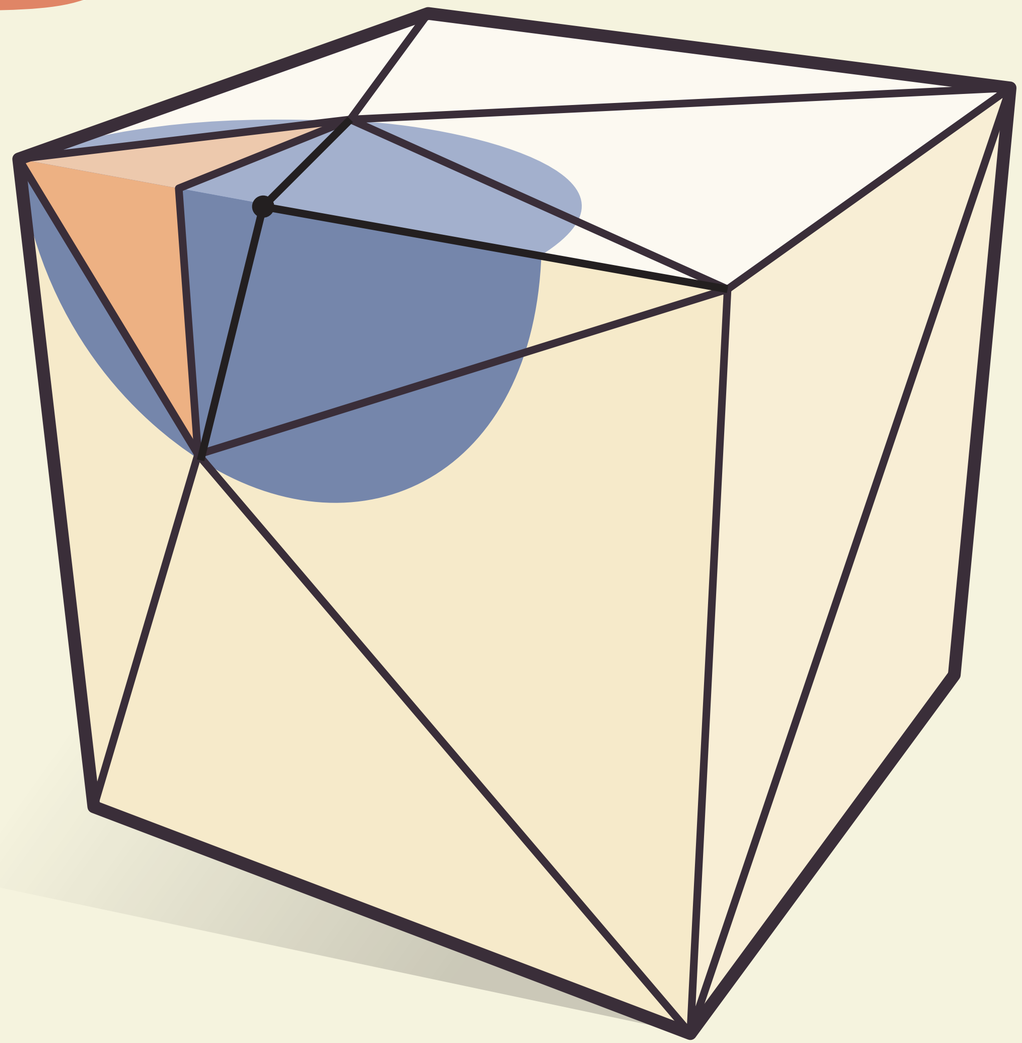


Intrinsic Delaunay refinement — algorithm

1. Pick any triangle violating angle bound
2. Locate its circumcenter
3. Insert a vertex at its circumcenter

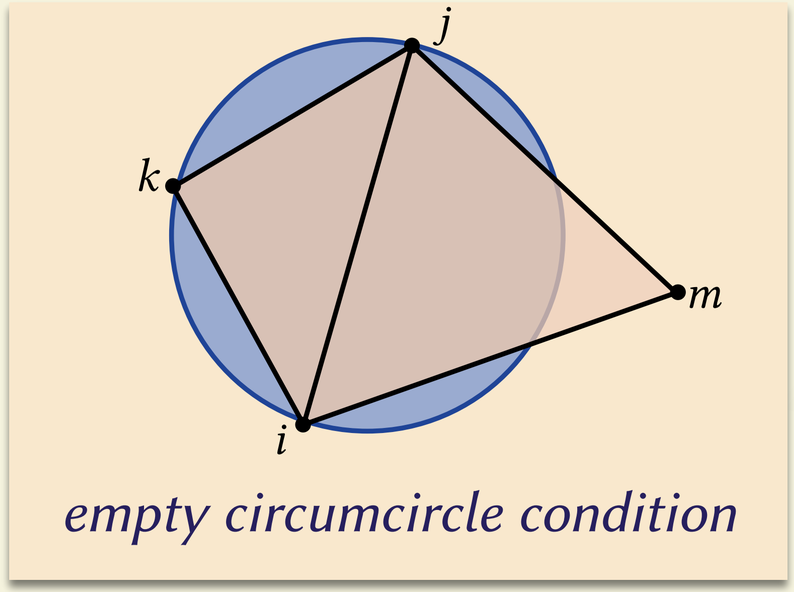


empty circumcircle condition

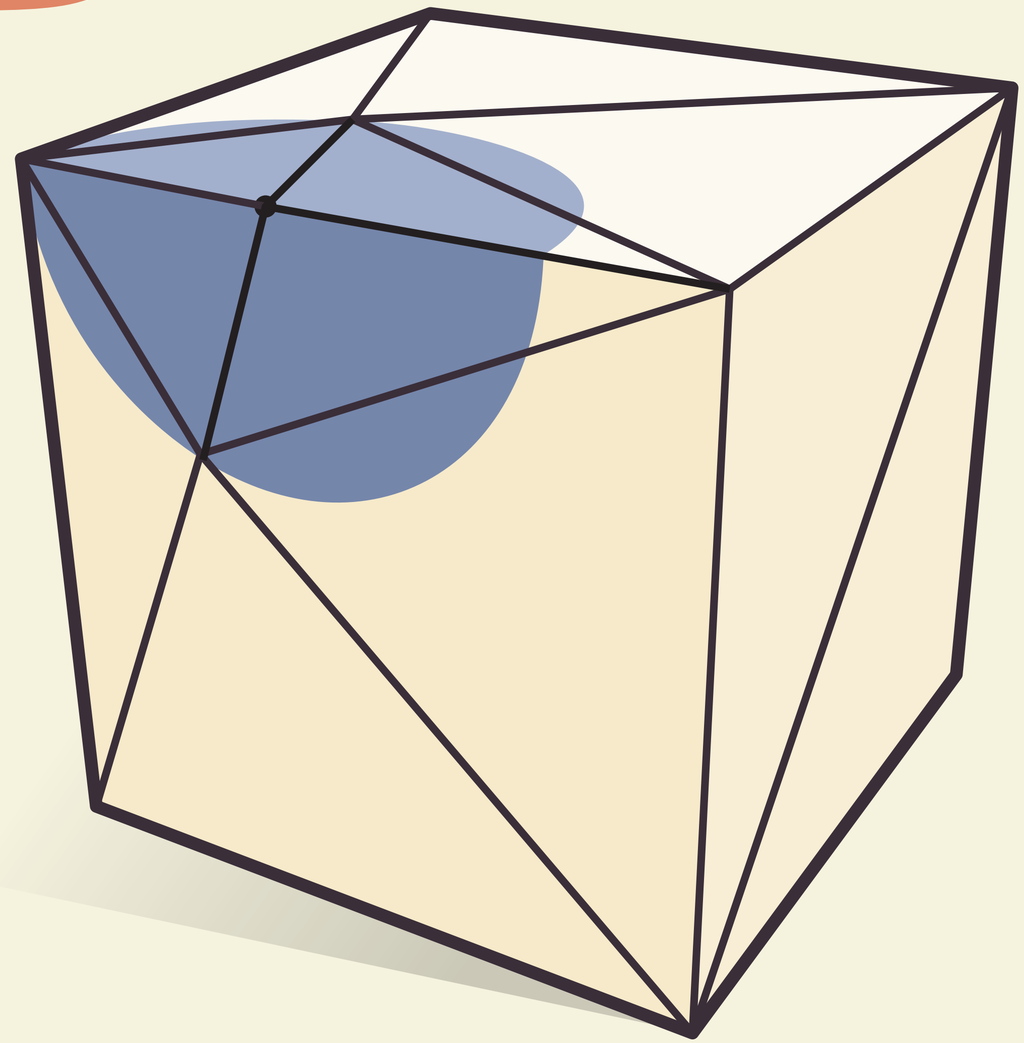


Intrinsic Delaunay refinement — algorithm

1. Pick any triangle violating angle bound
2. Locate its circumcenter
3. Insert a vertex at its circumcenter
4. Flip to Delaunay

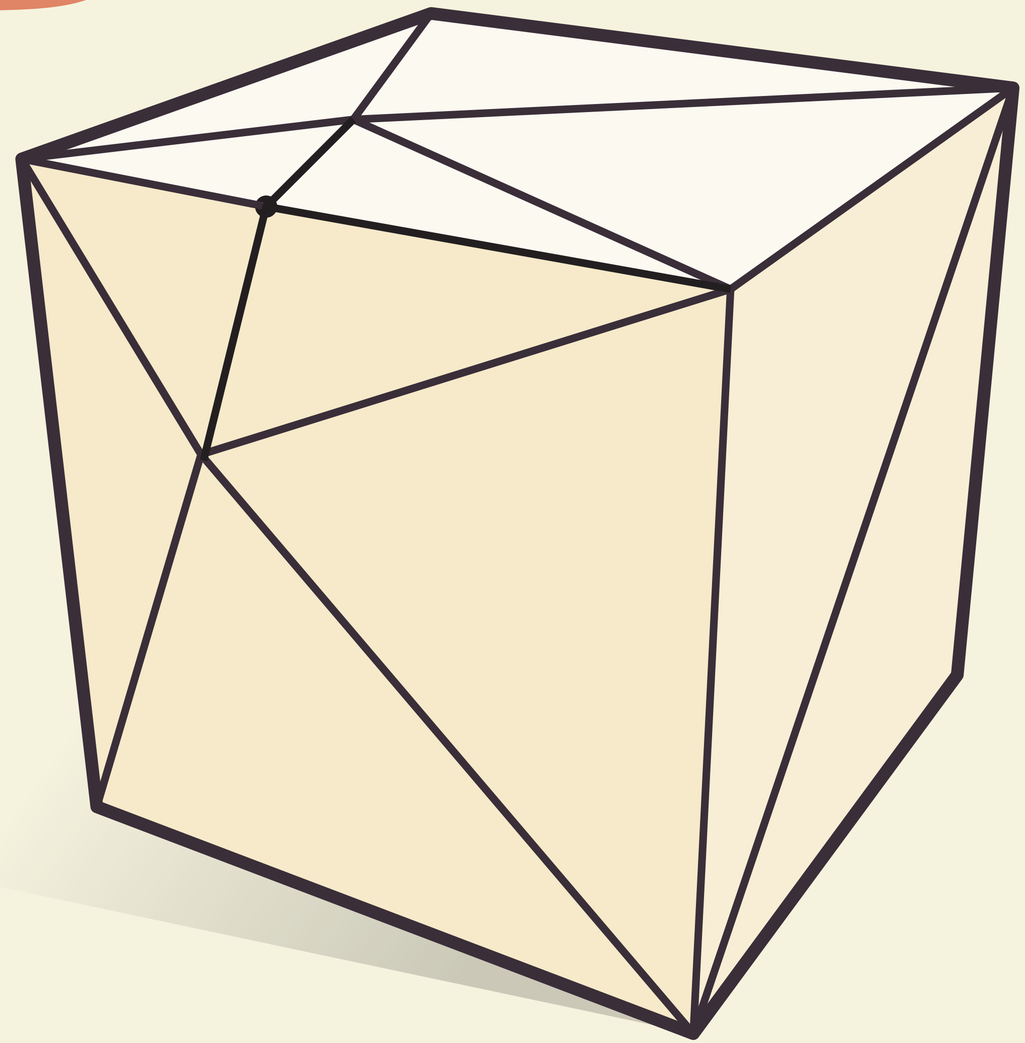


empty circumcircle condition

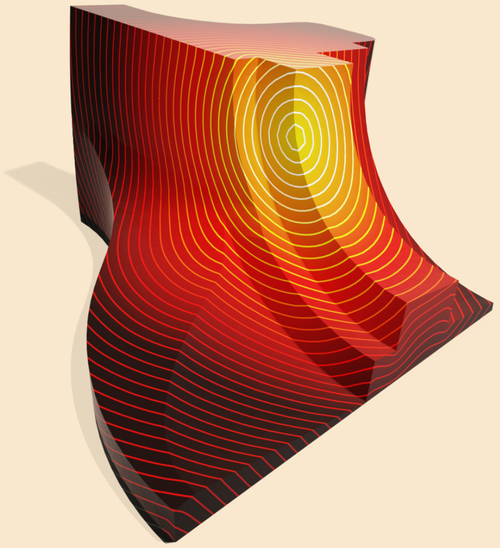


Intrinsic Delaunay refinement — algorithm

1. Pick any triangle violating angle bound
2. Locate its circumcenter
3. Insert a vertex at its circumcenter
4. Flip to Delaunay
5. … repeat until all triangles obey angle bound

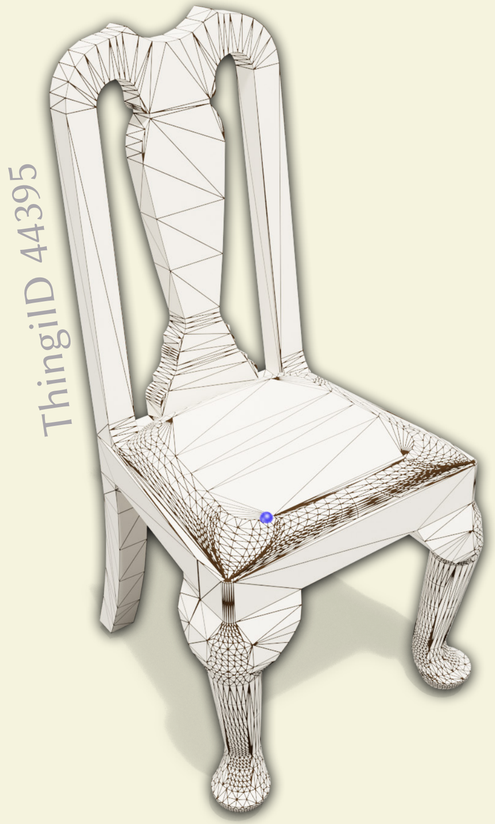


Example: Approximate Shortest Paths



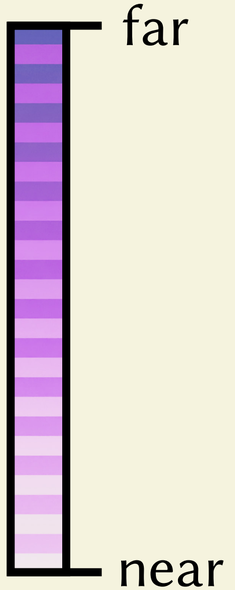
heat method for surface distances

[Crane, Weischedel & Wardetzky 2013]

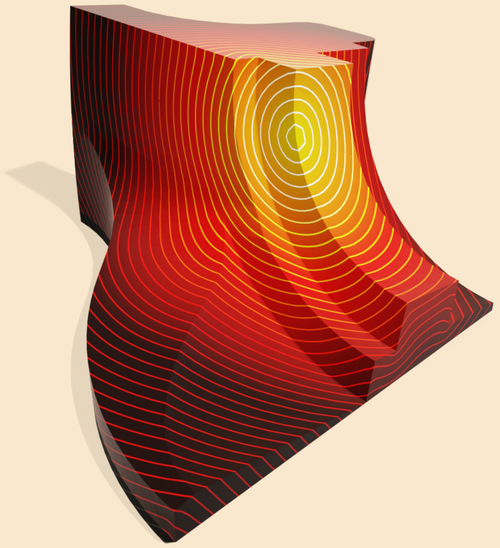


mean error: 28%

result on input mesh

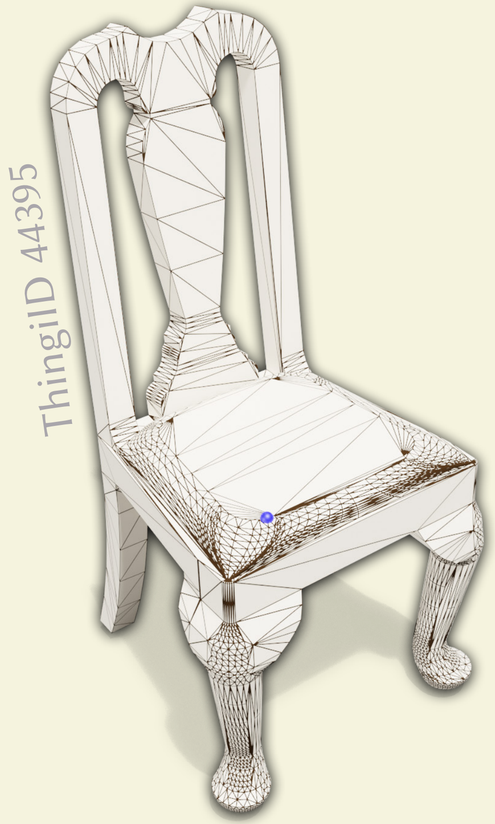


Example: Approximate Shortest Paths



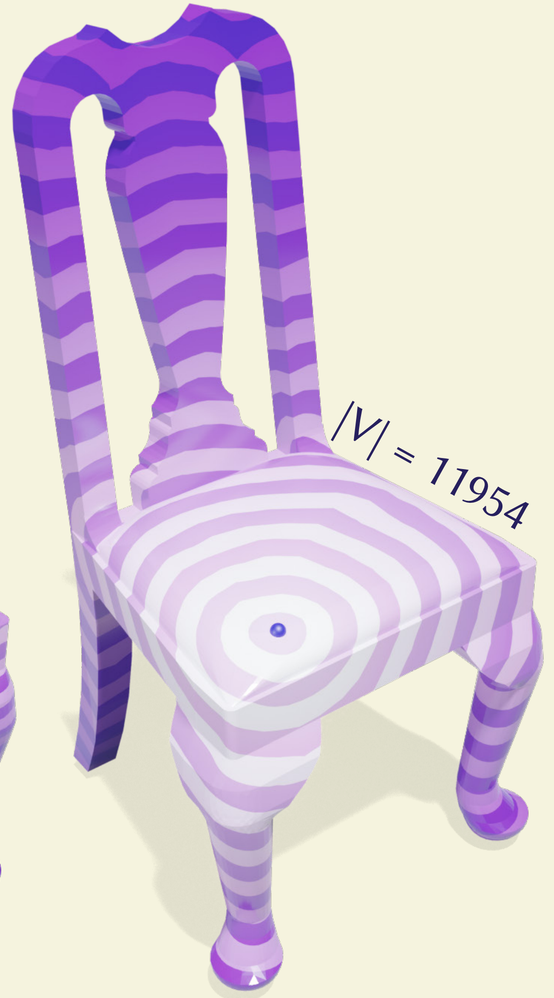
heat method for surface distances

[Crane, Weischedel & Wardetzky 2013]



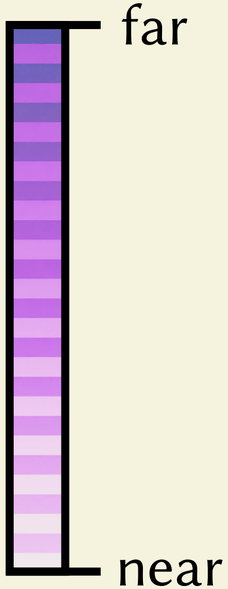
mean error: 28%

result on input mesh

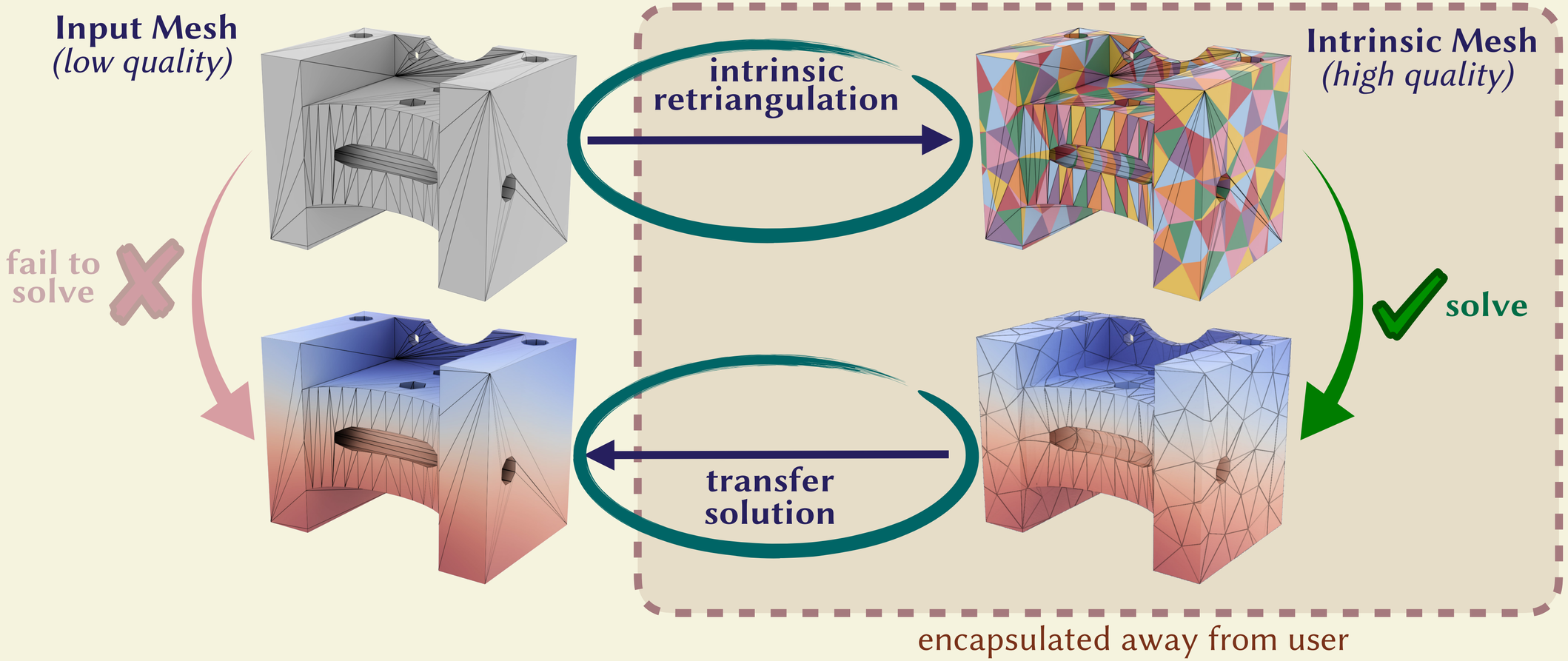


mean error: 2%

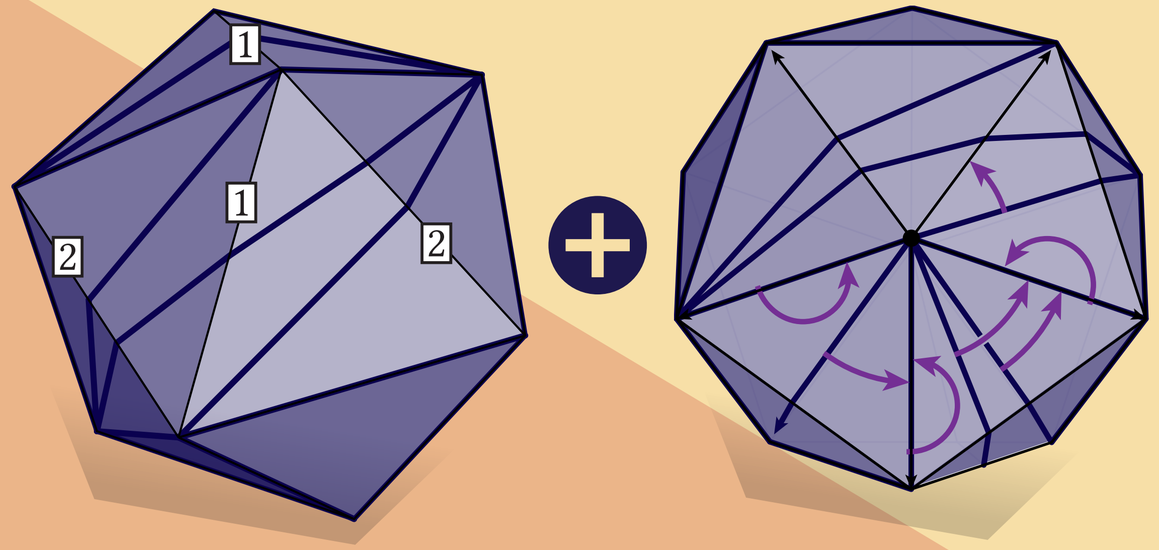
result on Delaunay refinement



Tracking Correspondence

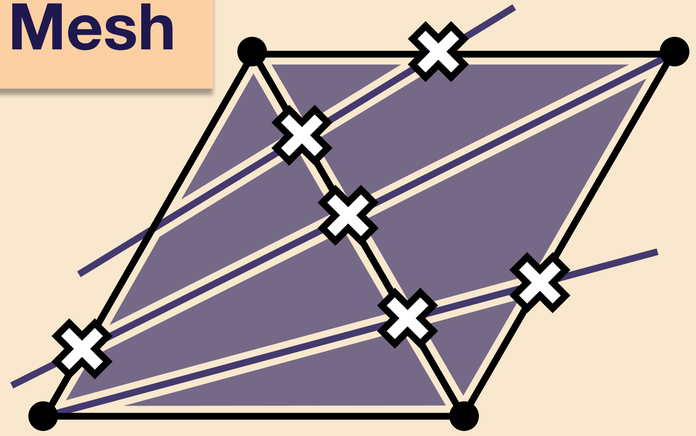


2 Data Structures



Correspondence data structures

Overlay Mesh

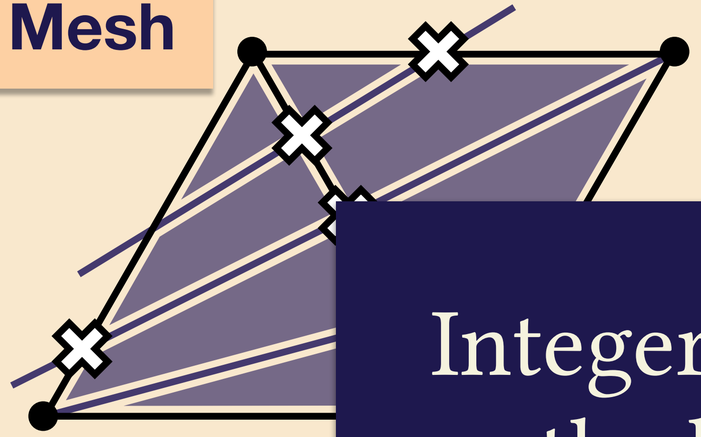


[Fisher, Springborn, Bobenko & Schröder 2006]

- Explicit mesh of common subdivision
- Edge flips nonlocal & expensive
  - **No further operations**

Correspondence data structures

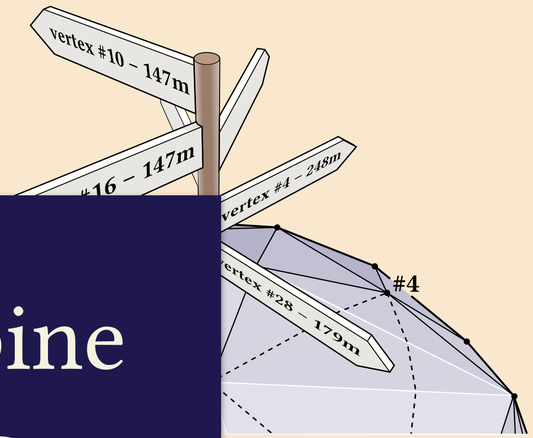
Overlay Mesh



[Fisher, Springb
& Schröc

- Explicit mesh of common subdivision
- Edge flips nonlocal & expensive
  - **No further operations**

Signposts

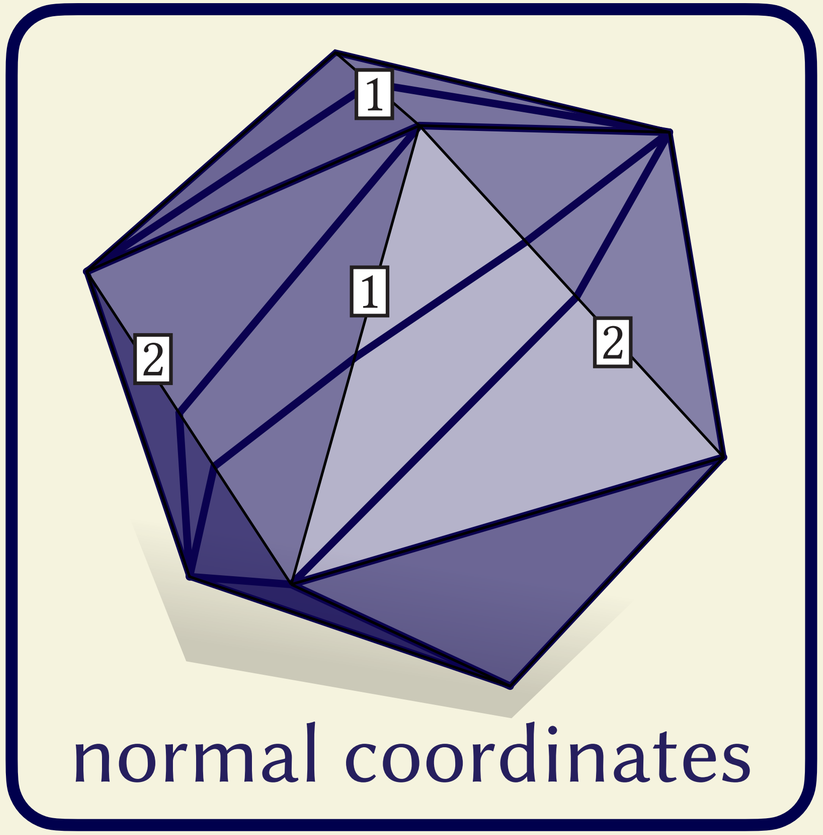


n & Crane 2019]

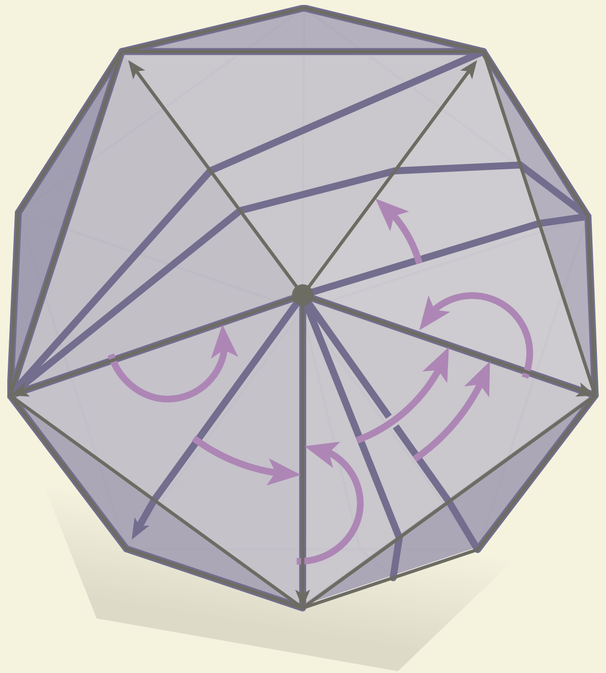
- Floating point quantities stored at vertices
- **Supports many local mesh operations**
- Common subdivision connectivity may be invalid

Integer coordinates combine the best of both worlds

Integer coordinates data structure



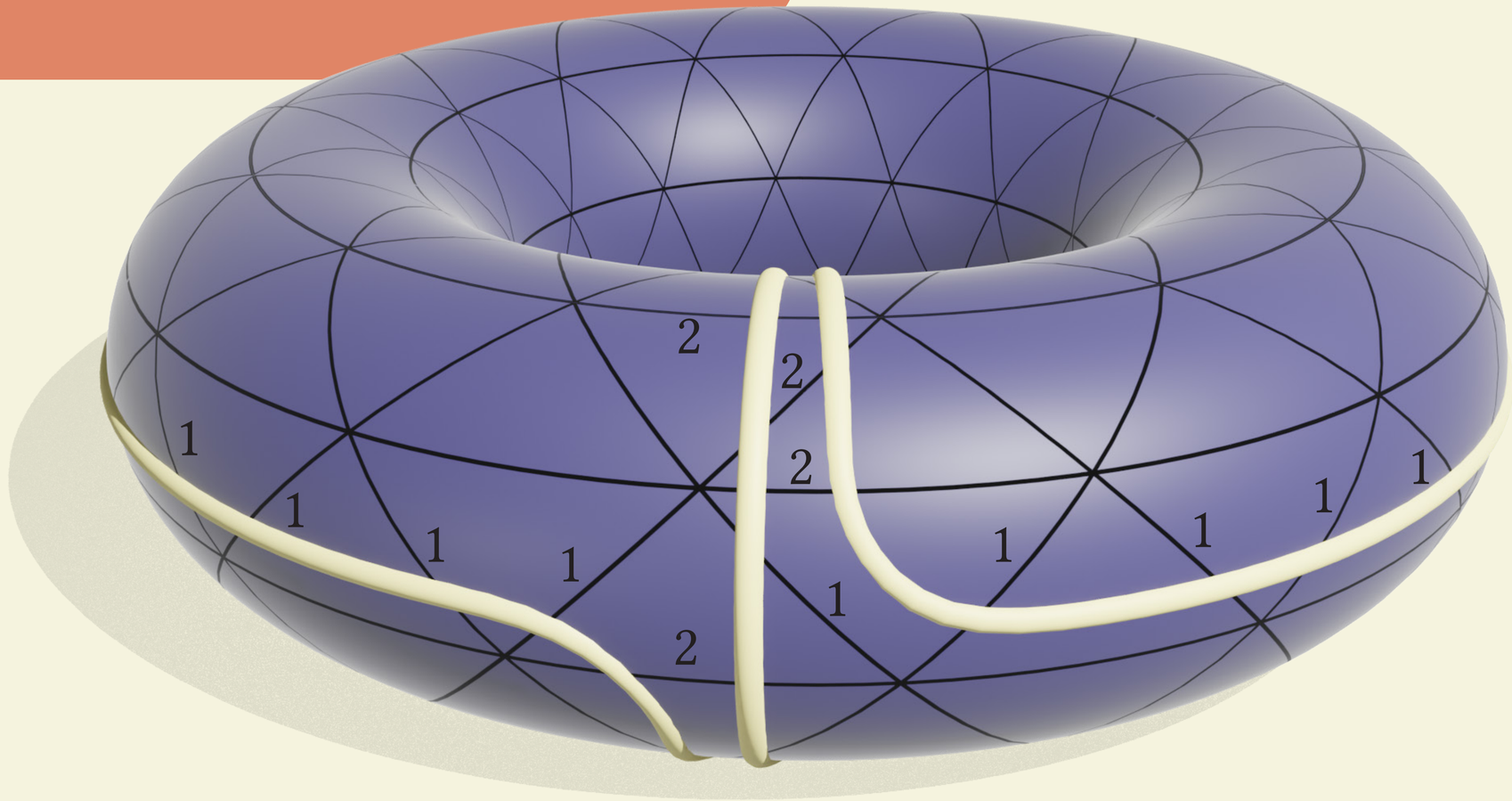
normal coordinates



roundabouts

(concretely, just 3 integers per edge)

Normal coordinates



Foundations: [Kneser 1929; Haken 1961]

Computational Topology: [Schaefer+ 2008; Erickson & Nayyeri 2013]

Geometry Processing: [Hass & Trnkova 2020]

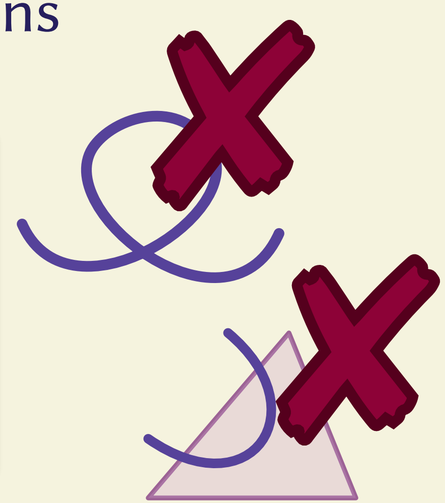
Encoding a curve with normal coordinates

- Just count intersections

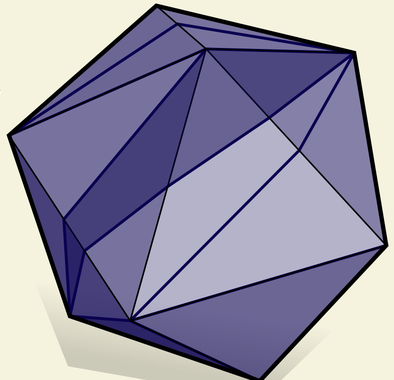
Rules

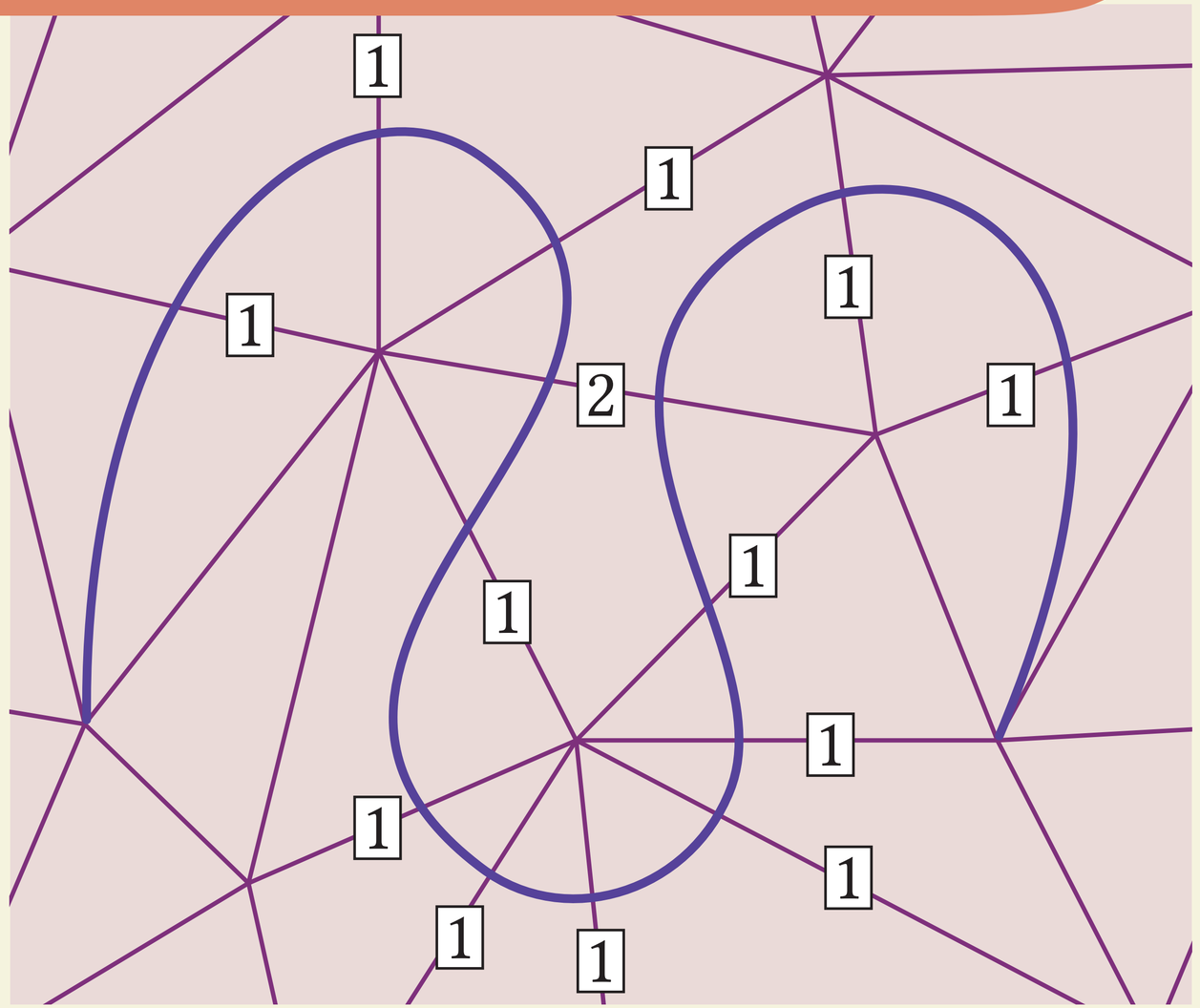
1. No self-crossings
2. No U-turns

(also curves may only start or end at vertices of the triangulation)

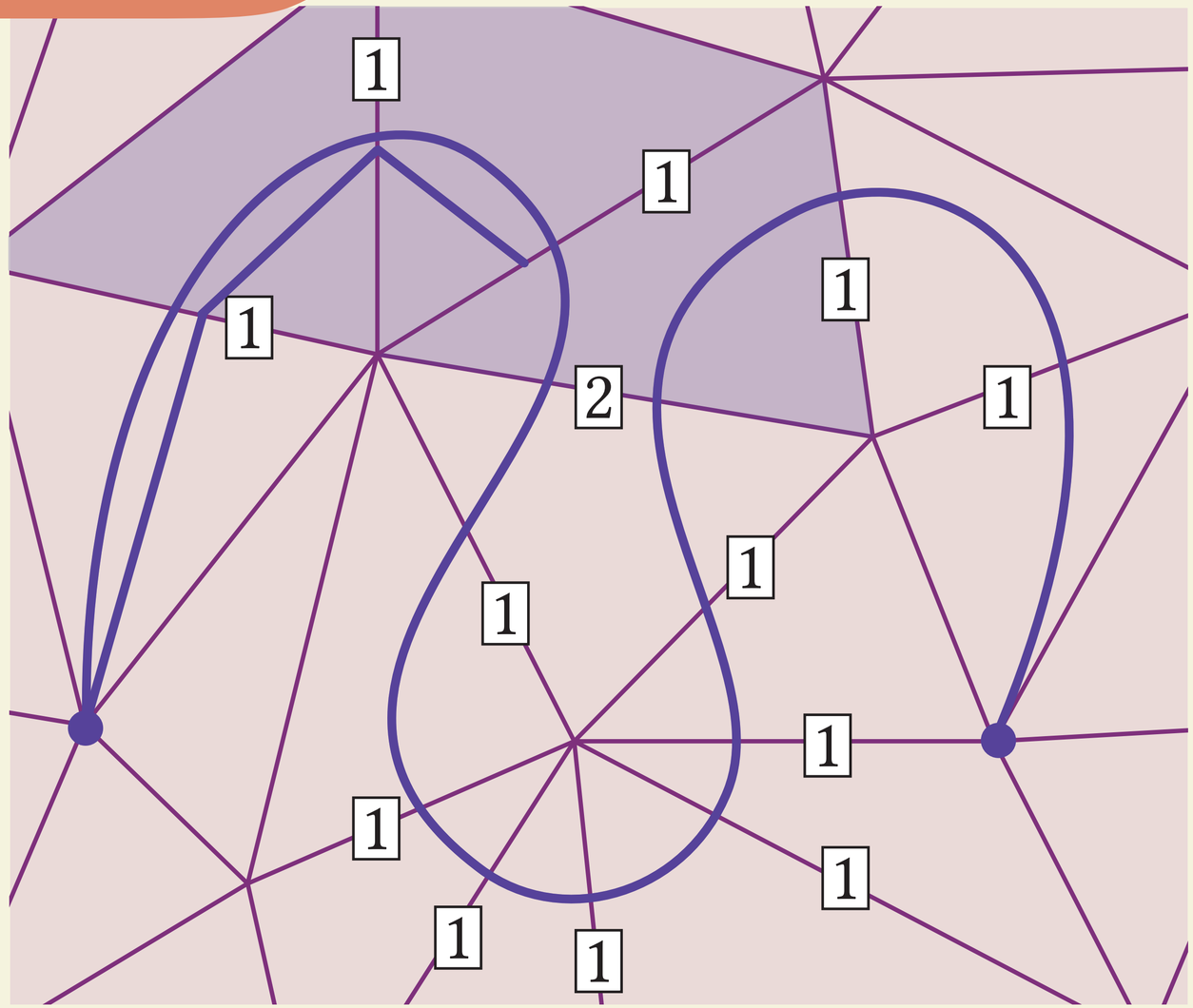


automatically satisfied for our triangulations

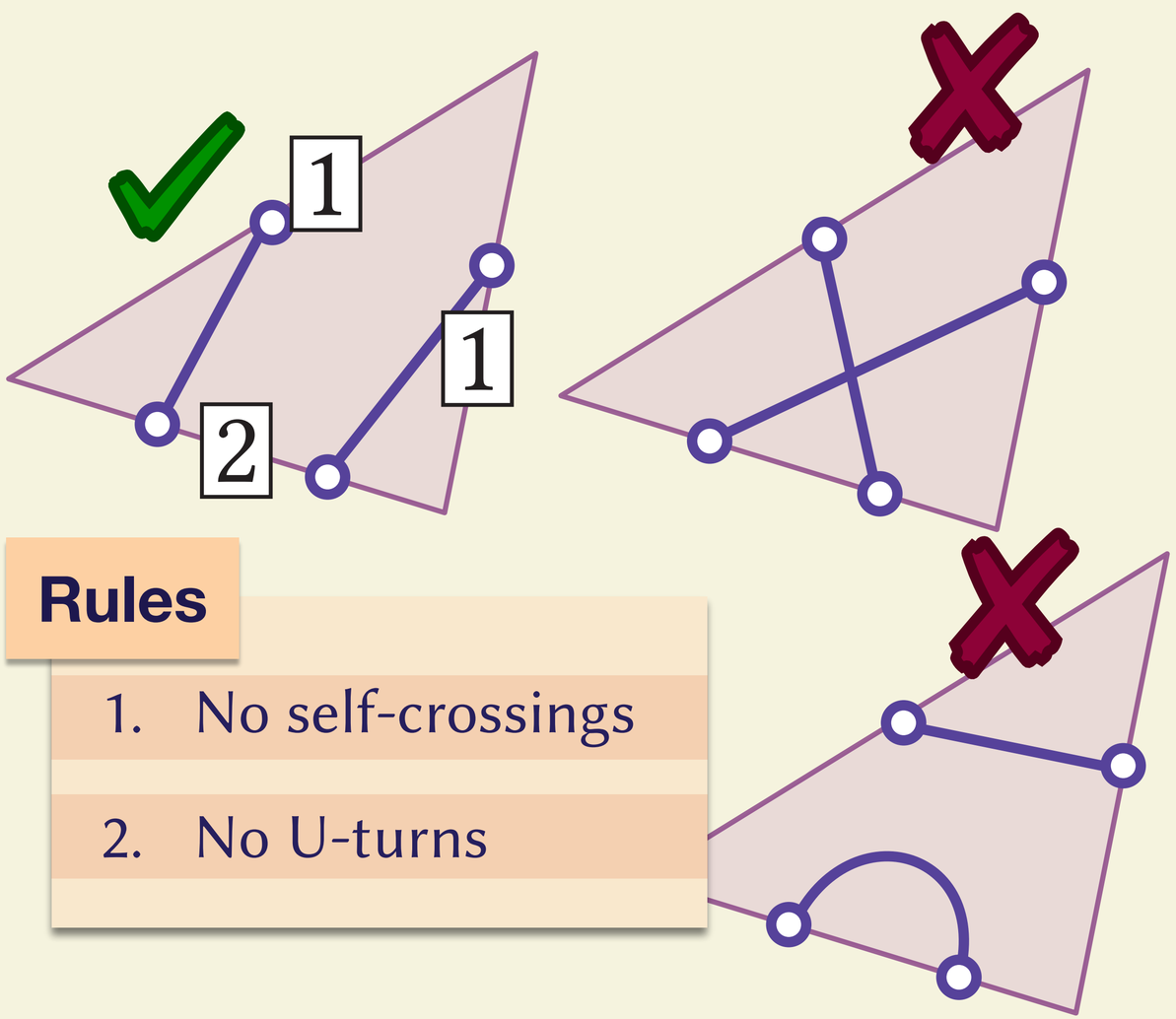




Reconstructing the curve

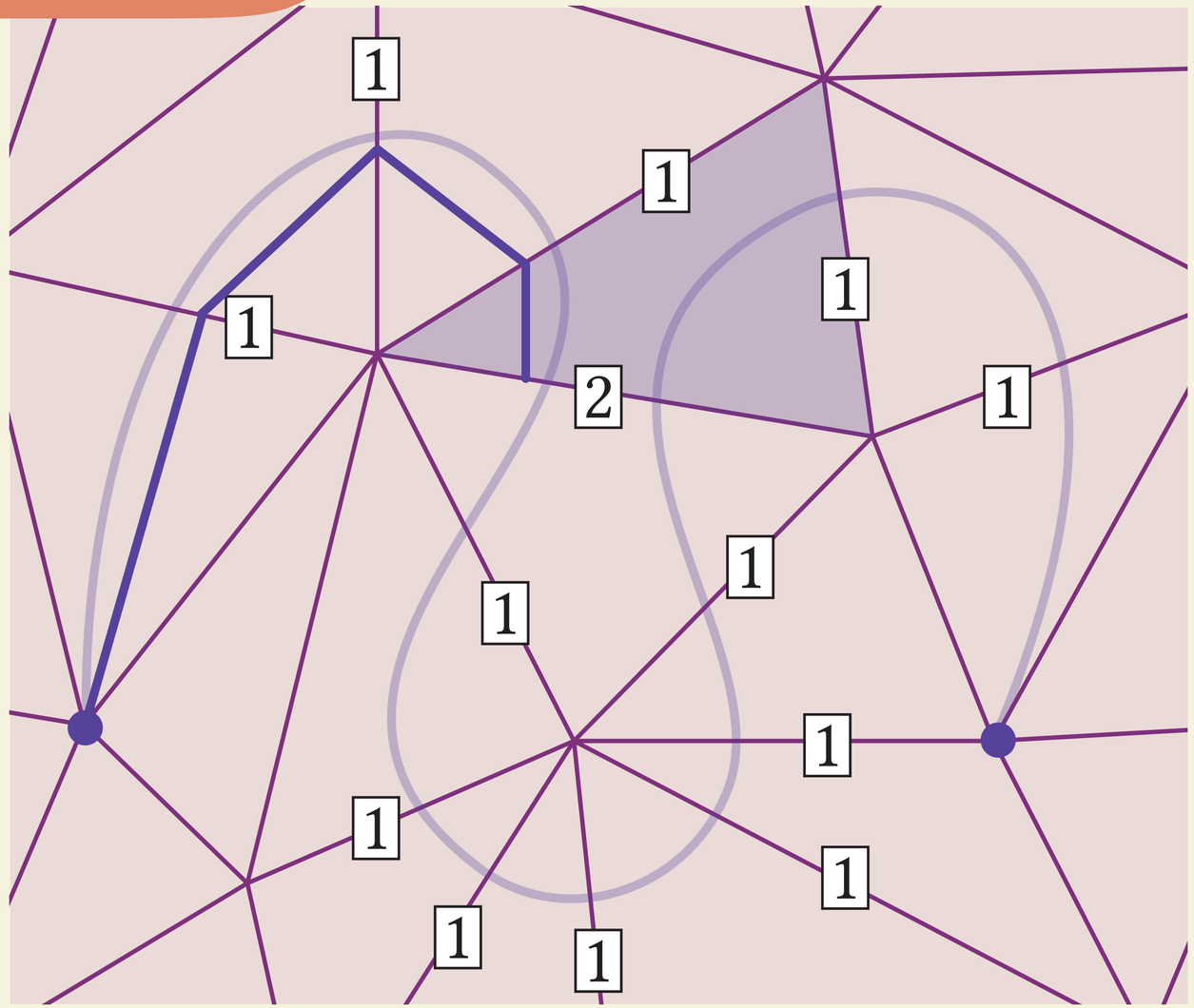


Reconstructing the curve

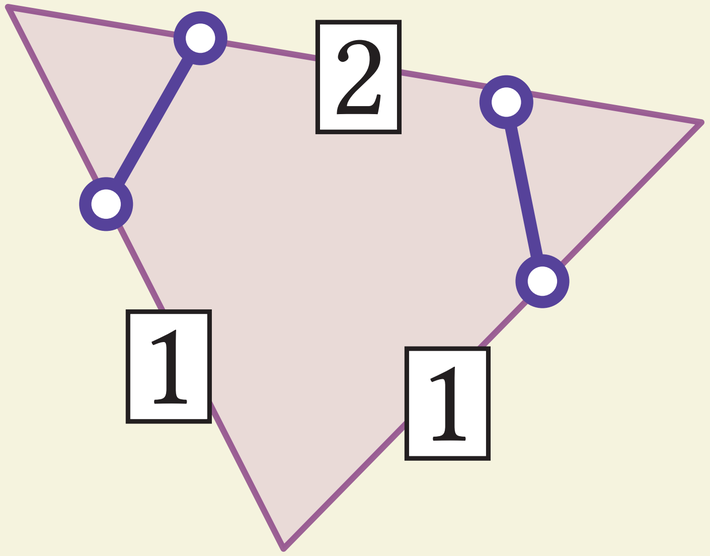


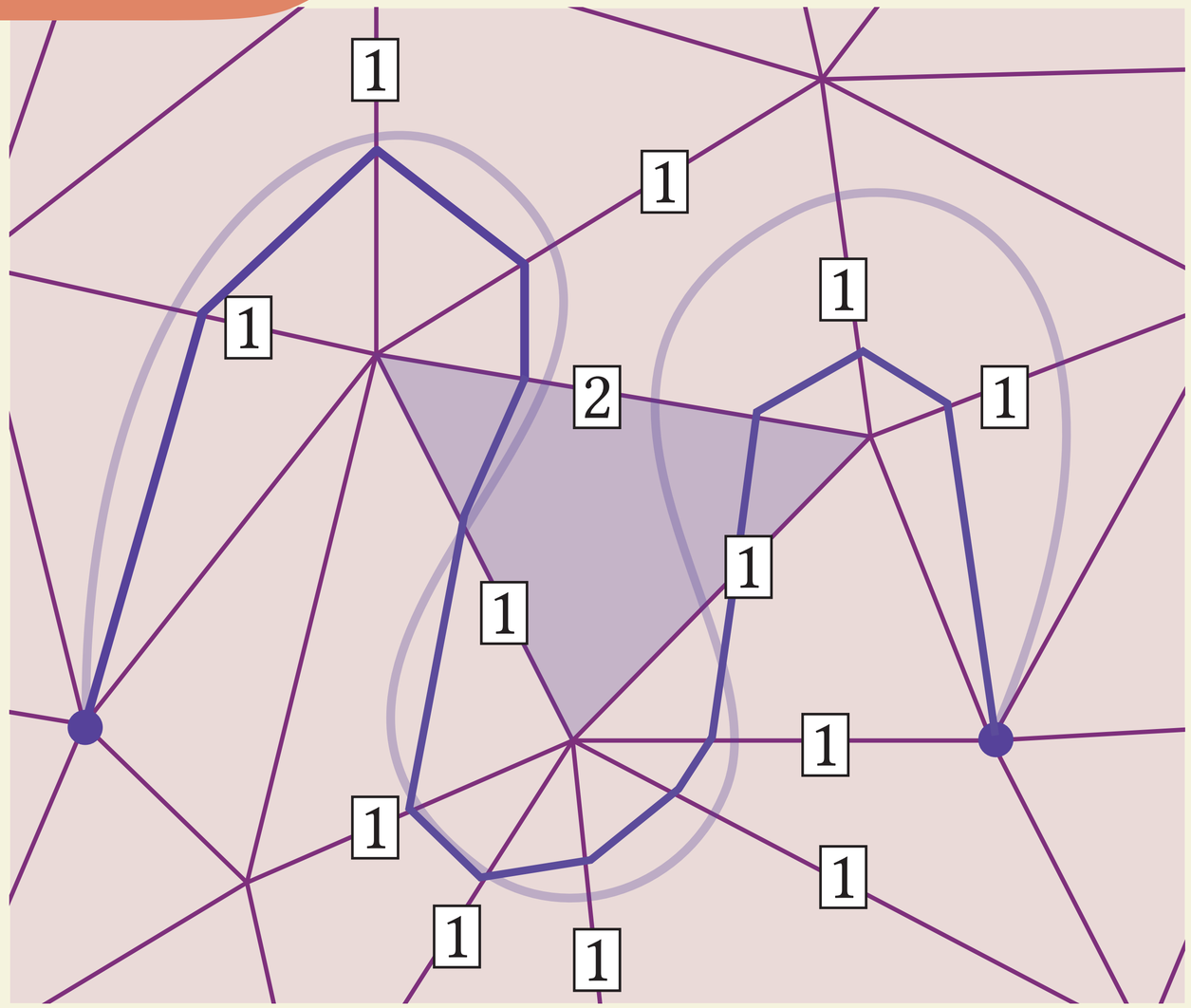
Rules

1. No self-crossings
2. No U-turns



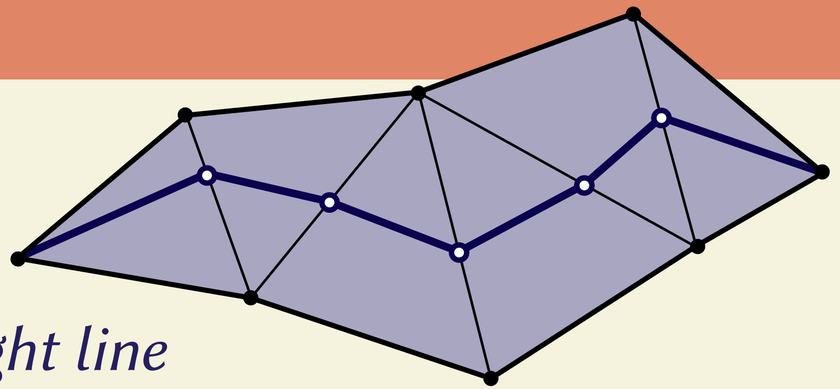
Reconstructing the curve

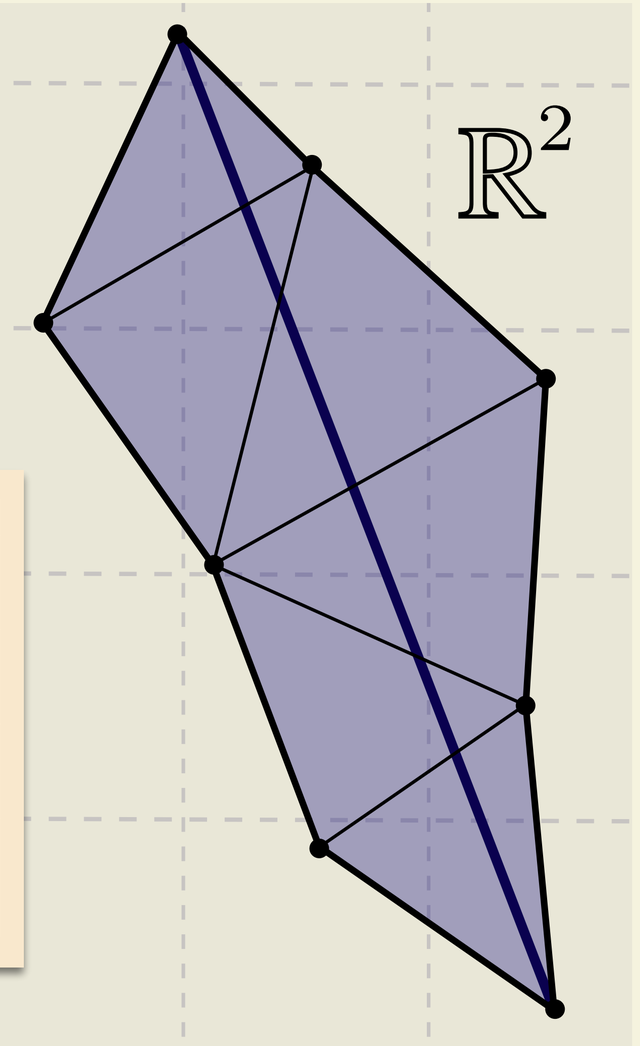




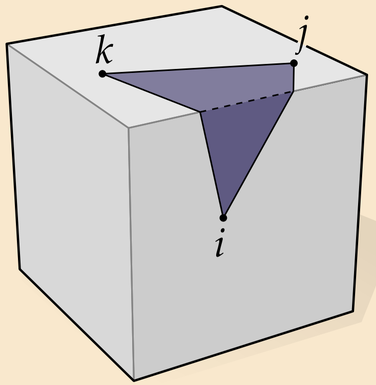
Finding the exact curve geometry

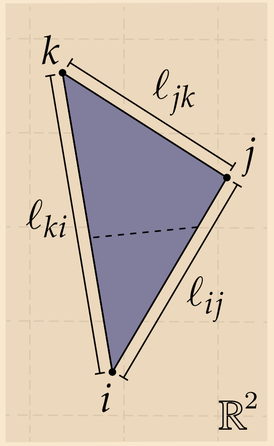
- So far: sequence of triangles
- True curve unfolds to a *straight line*
  - Lay out in plane for exact curve
- Normal coordinates determine edges exactly





Intrinsic edges

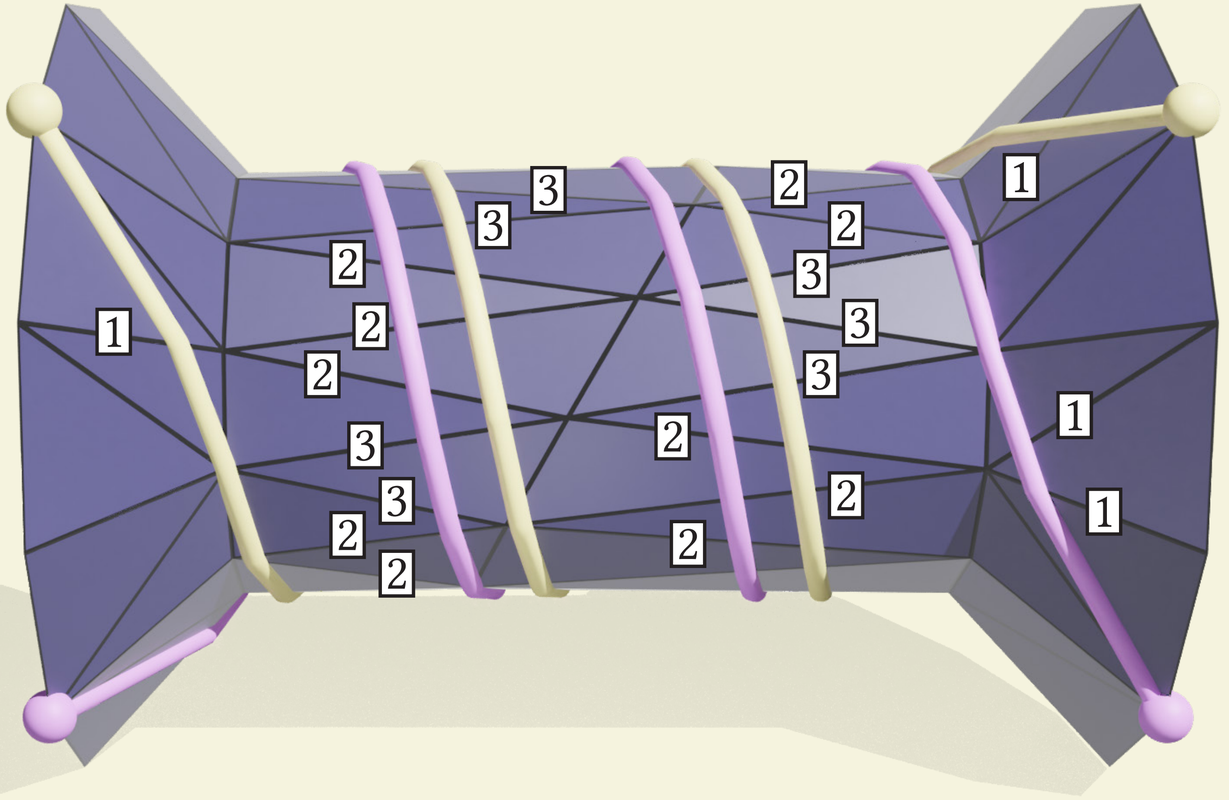




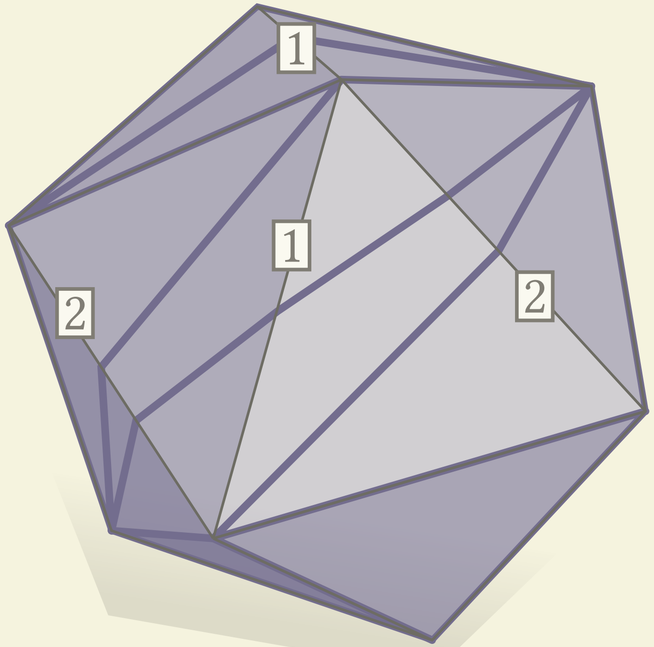
Collections of Curves

- *e.g.* edges of a triangulation
- Could store multiple sets of normal coordinates
  - Expensive 
- Instead, just store one set of normal coordinates

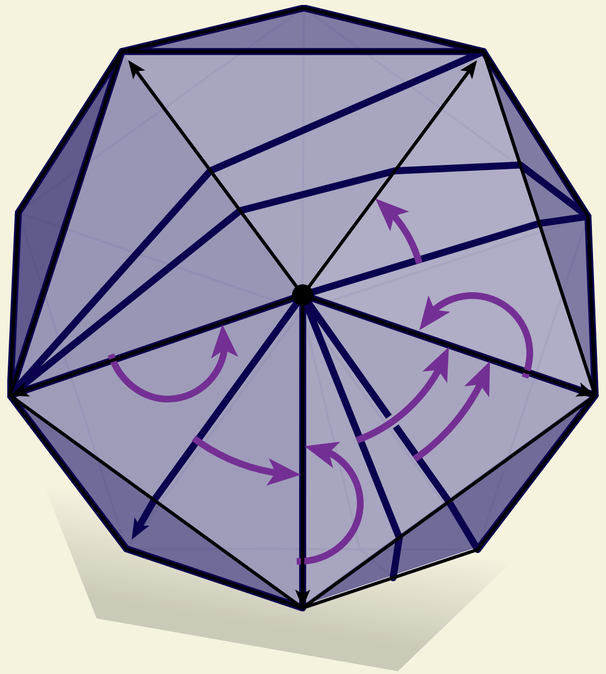
Store just one integer per edge



The integer coordinates data structure



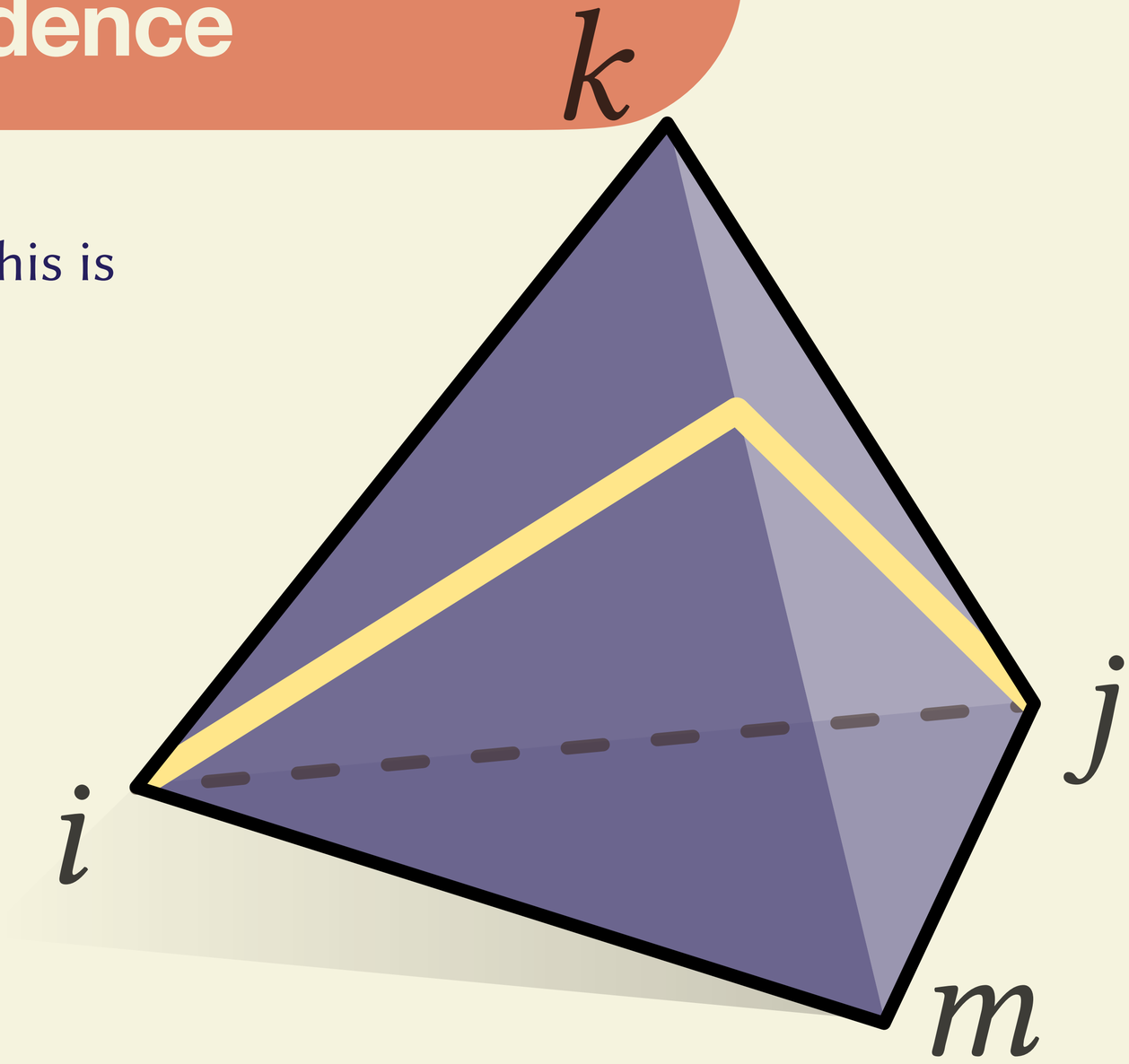
normal coordinates



roundabouts

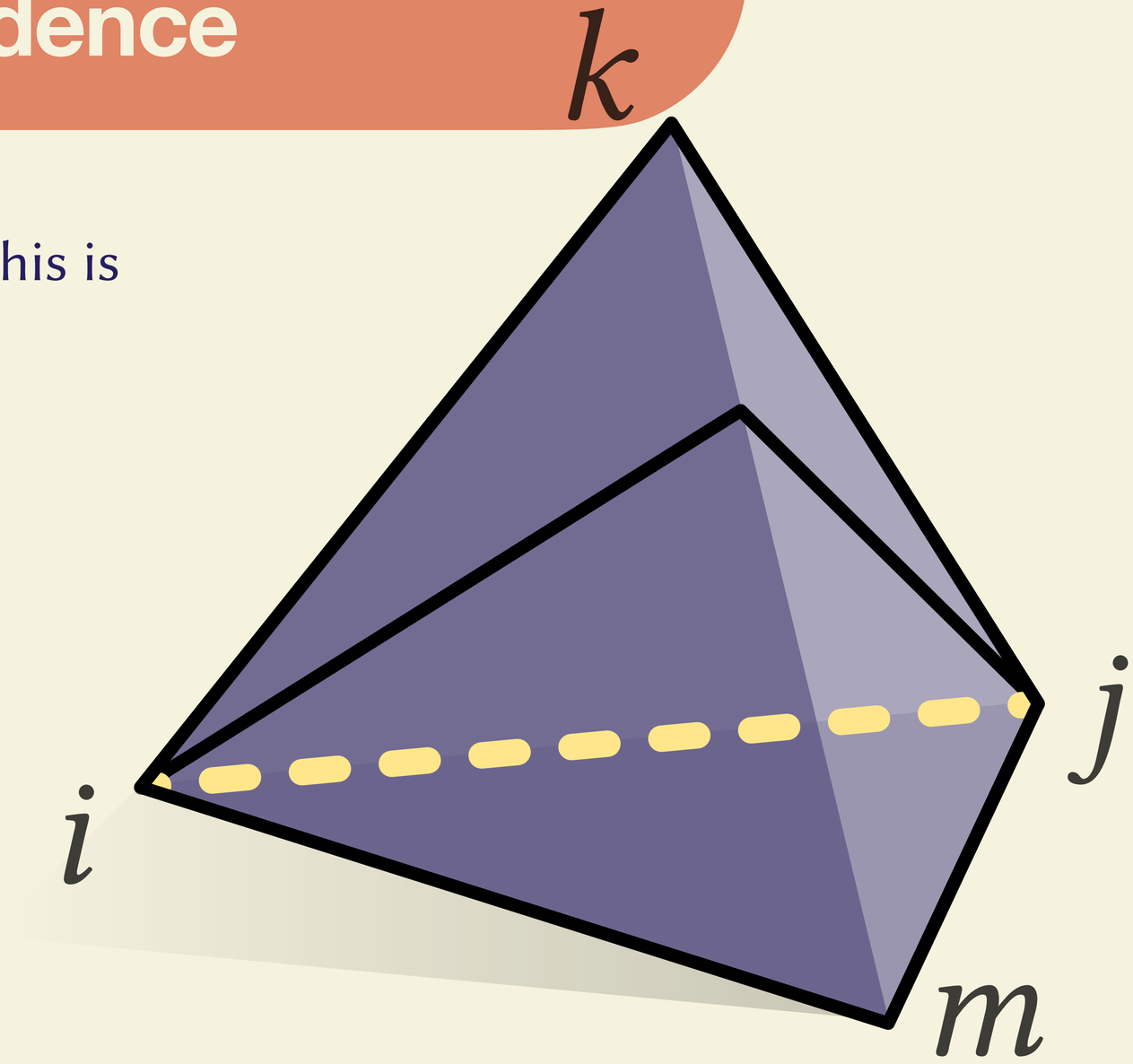
Normal coordinates are not enough to encode correspondence

- Can’t immediately tell which edge this is



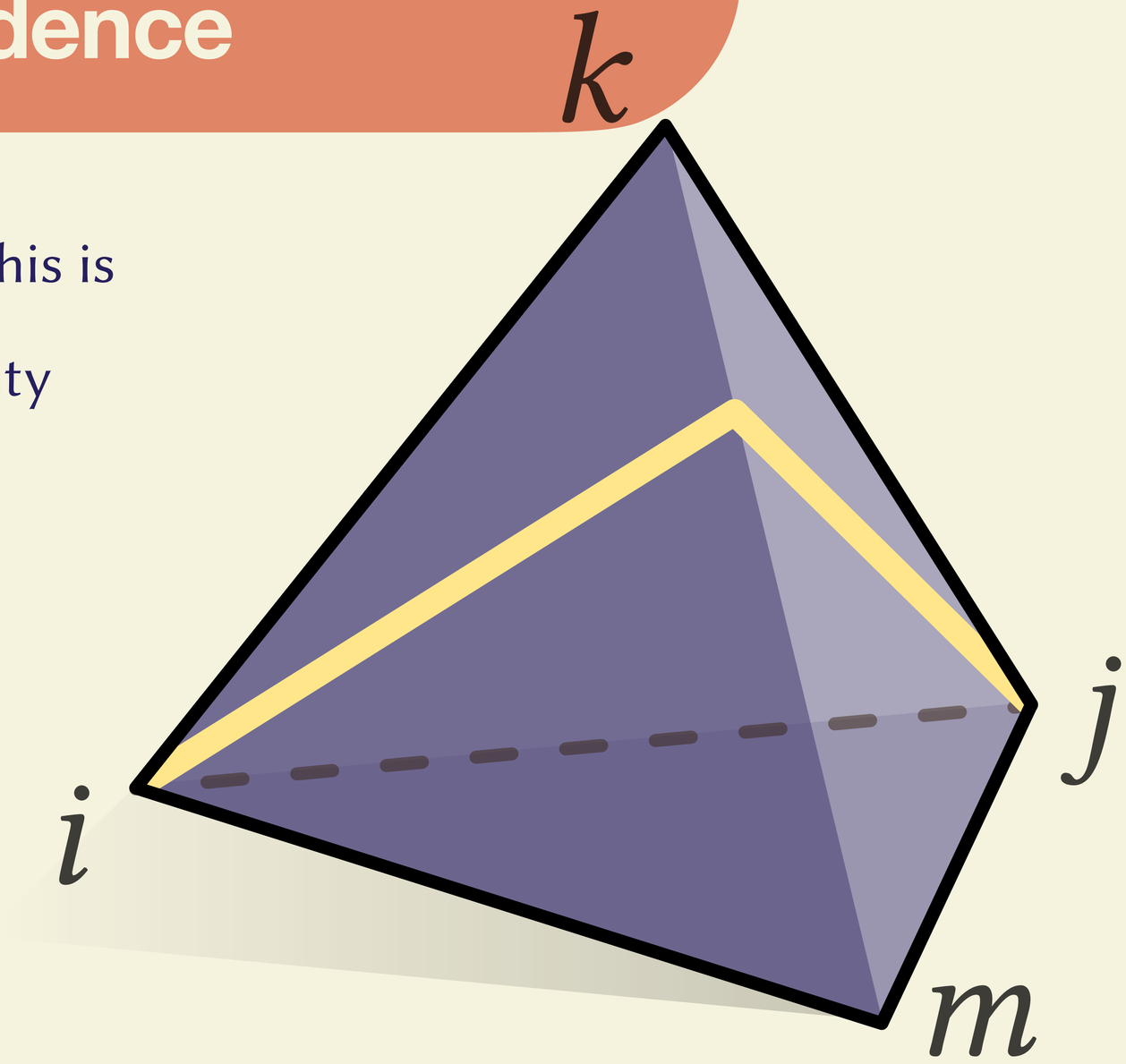
Normal coordinates are not enough to encode correspondence

- Can’t immediately tell which edge this is

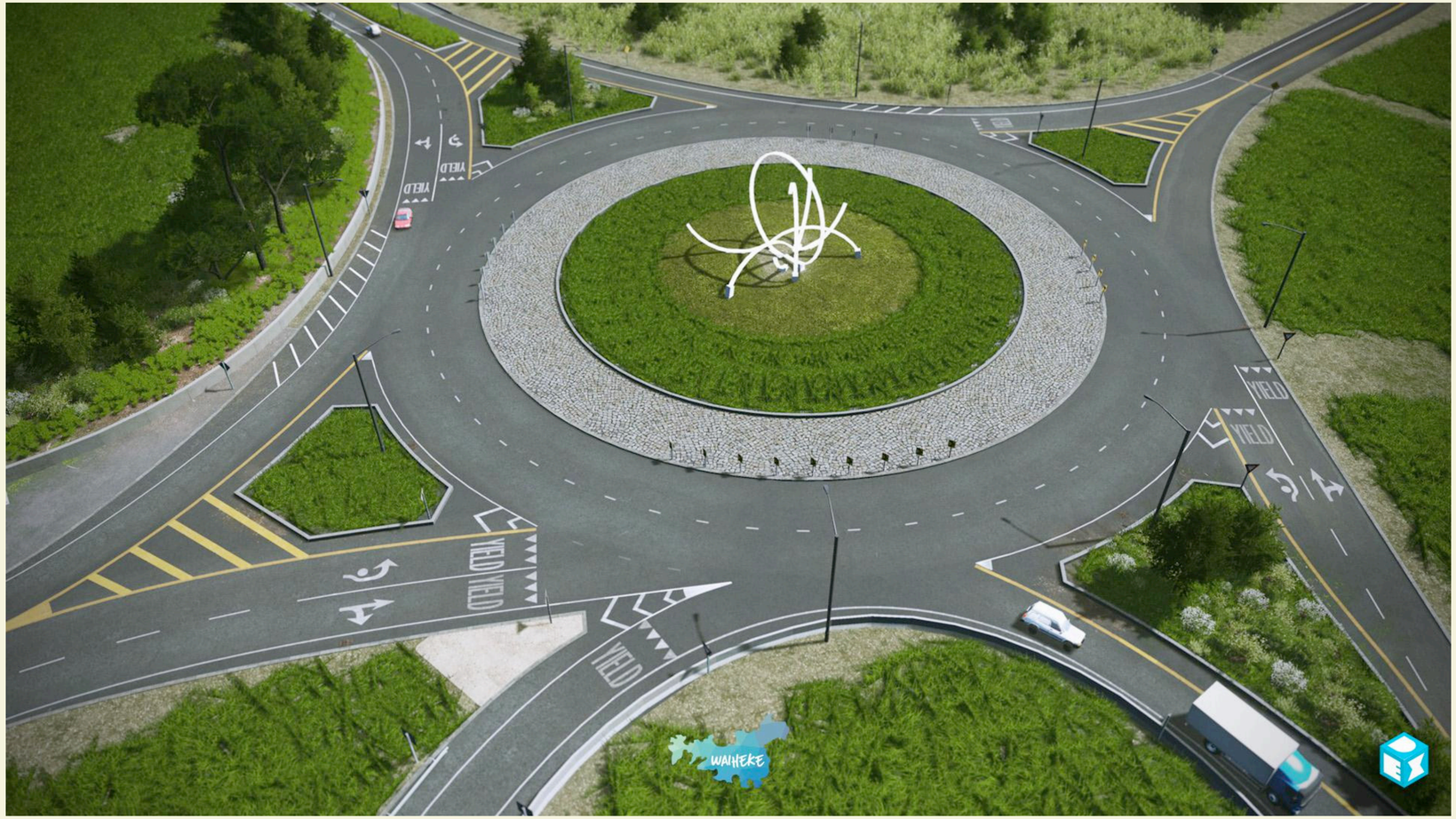


Normal coordinates are not enough to encode correspondence

- Can’t immediately tell which edge this is
  - Roundabouts resolve this ambiguity

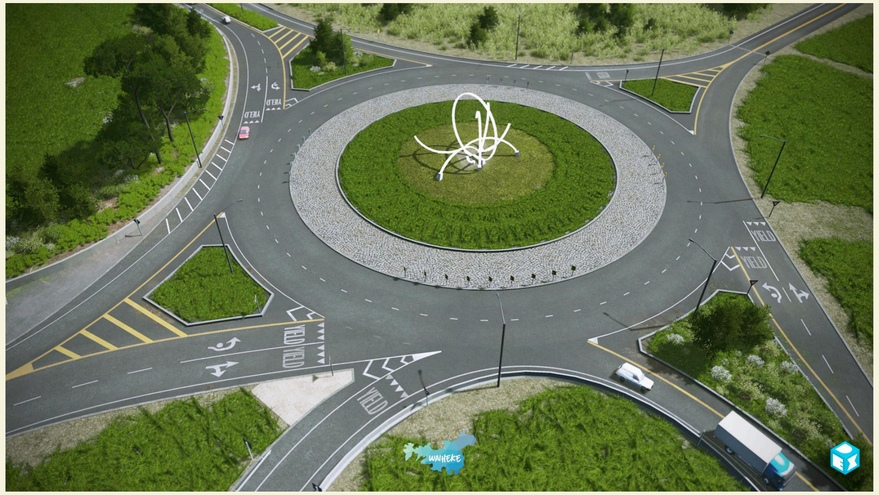


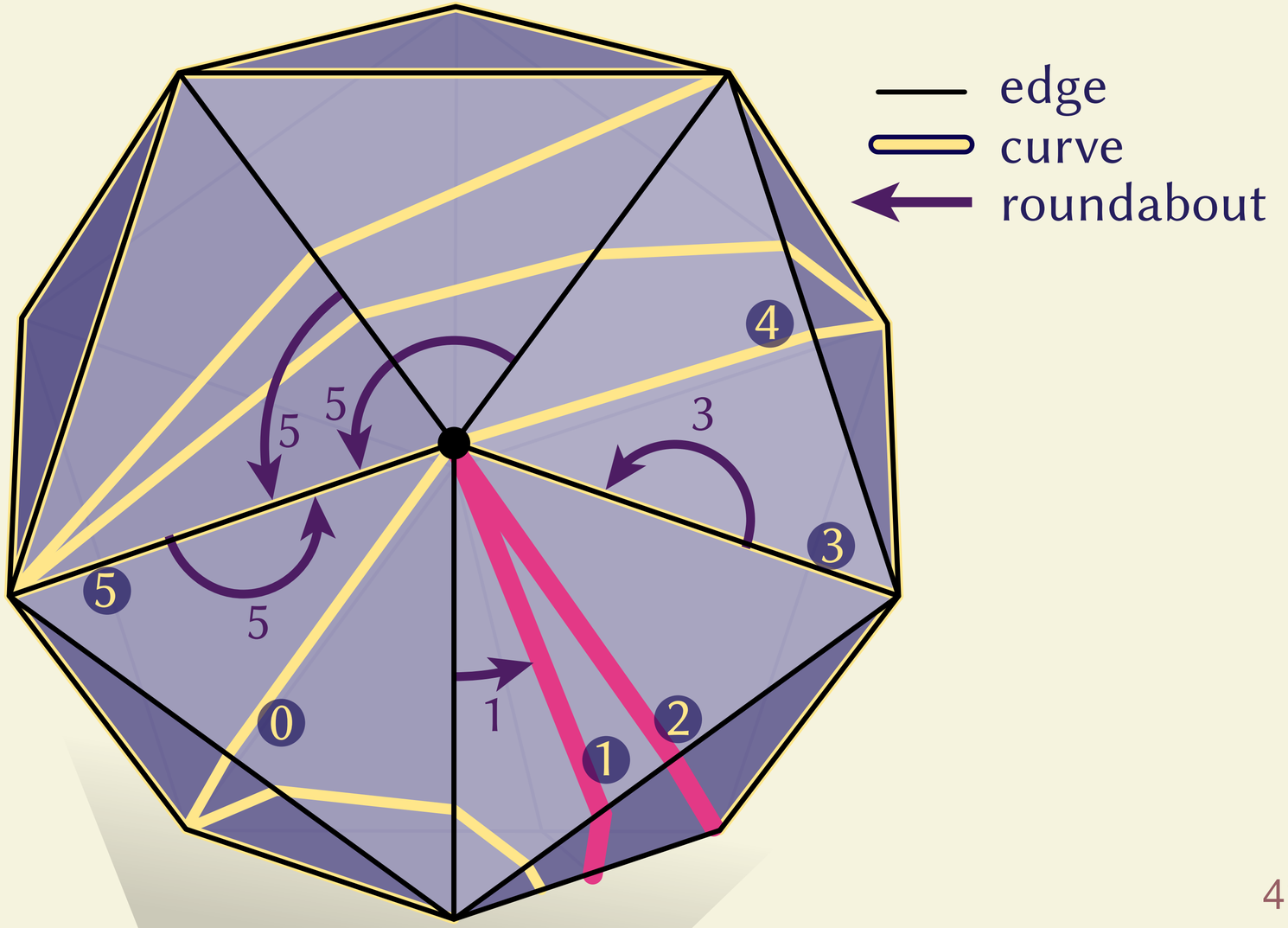
Roundabouts



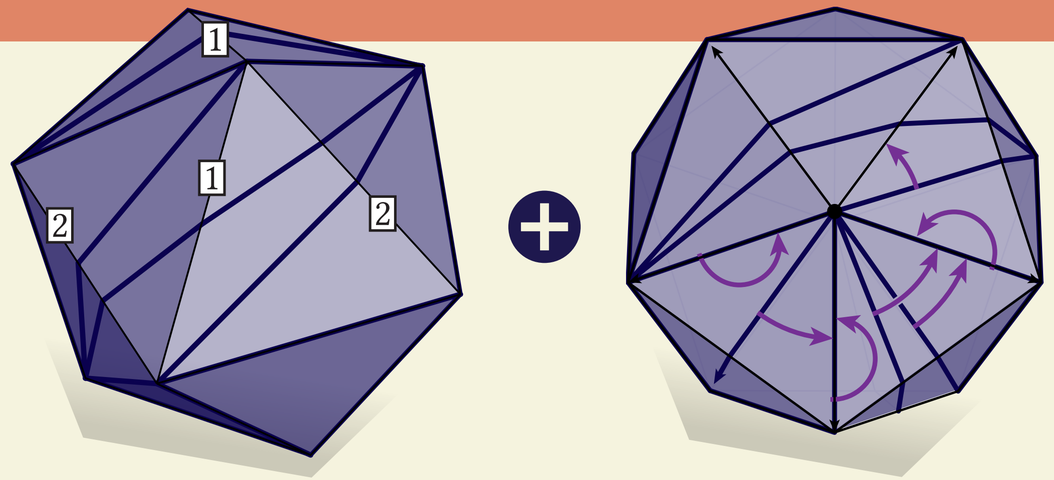
Roundabouts

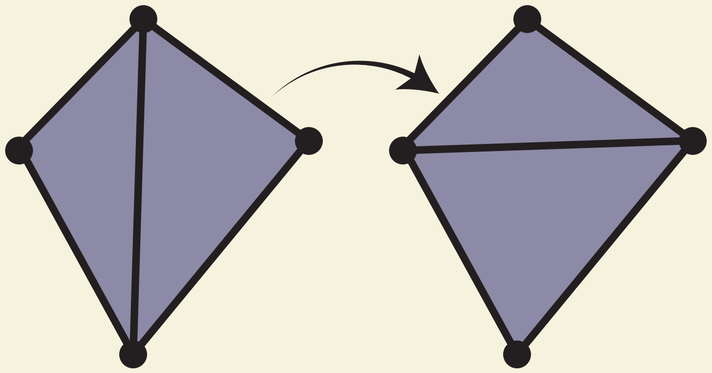
- Each black edge stores the index of the next yellow curve
- Resolves all ambiguity



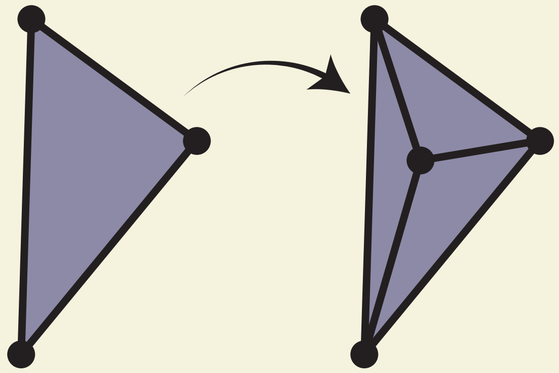


Data structure operations

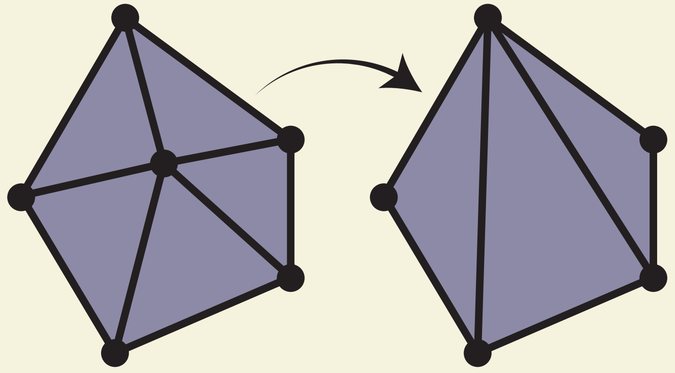




edge flips

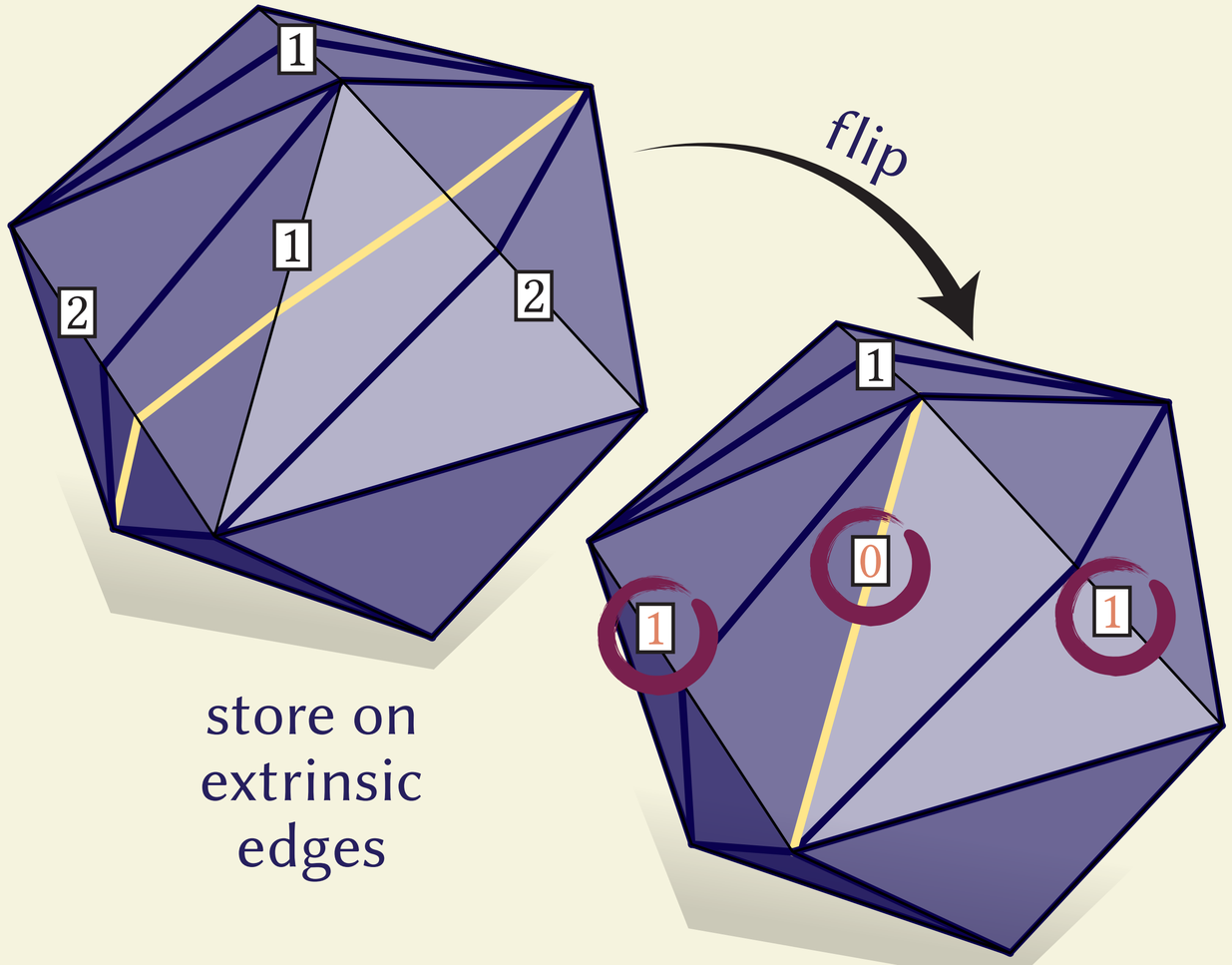


vertex insertion



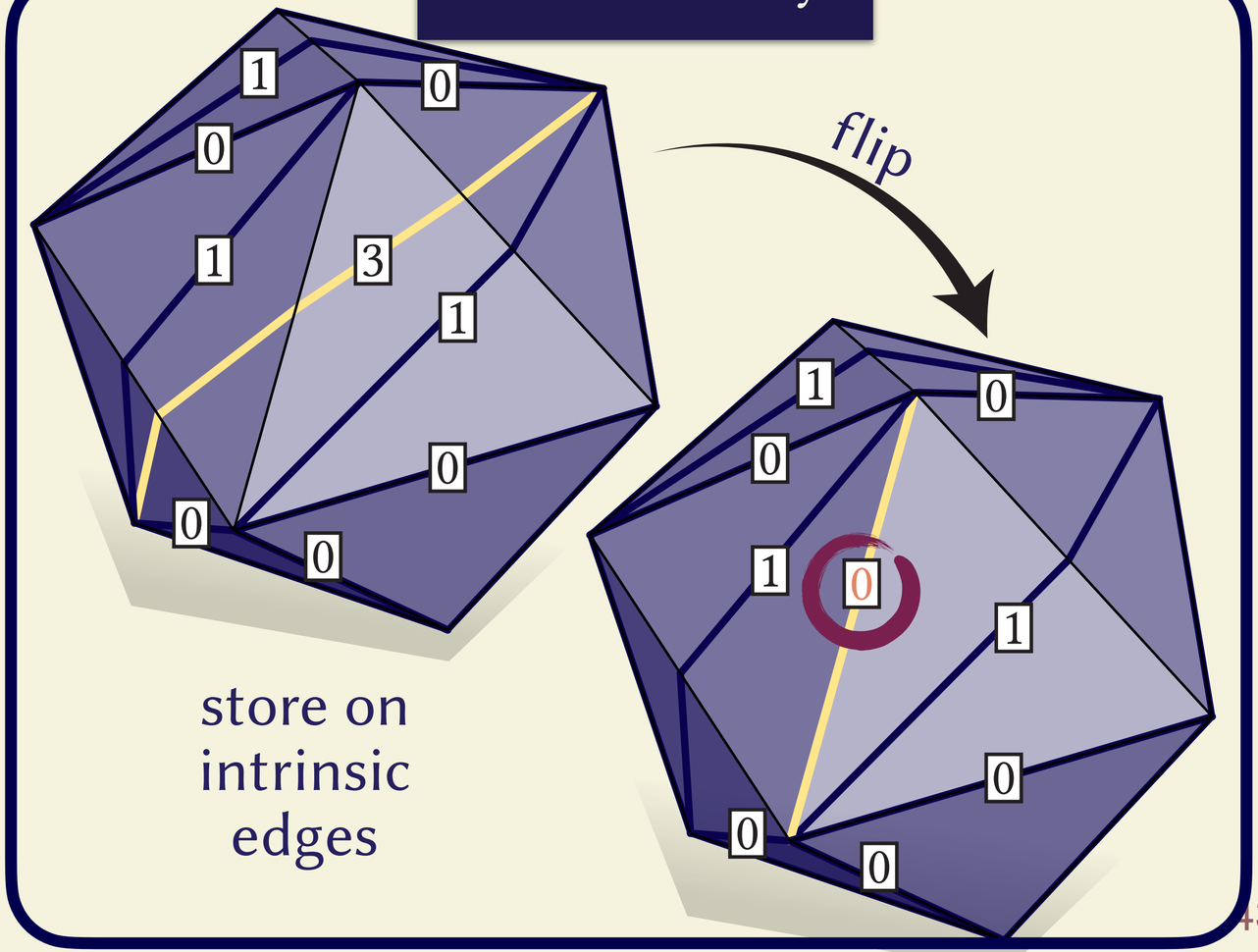
flat vertex removal

Where to store integer coordinates



store on extrinsic edges

Better locality

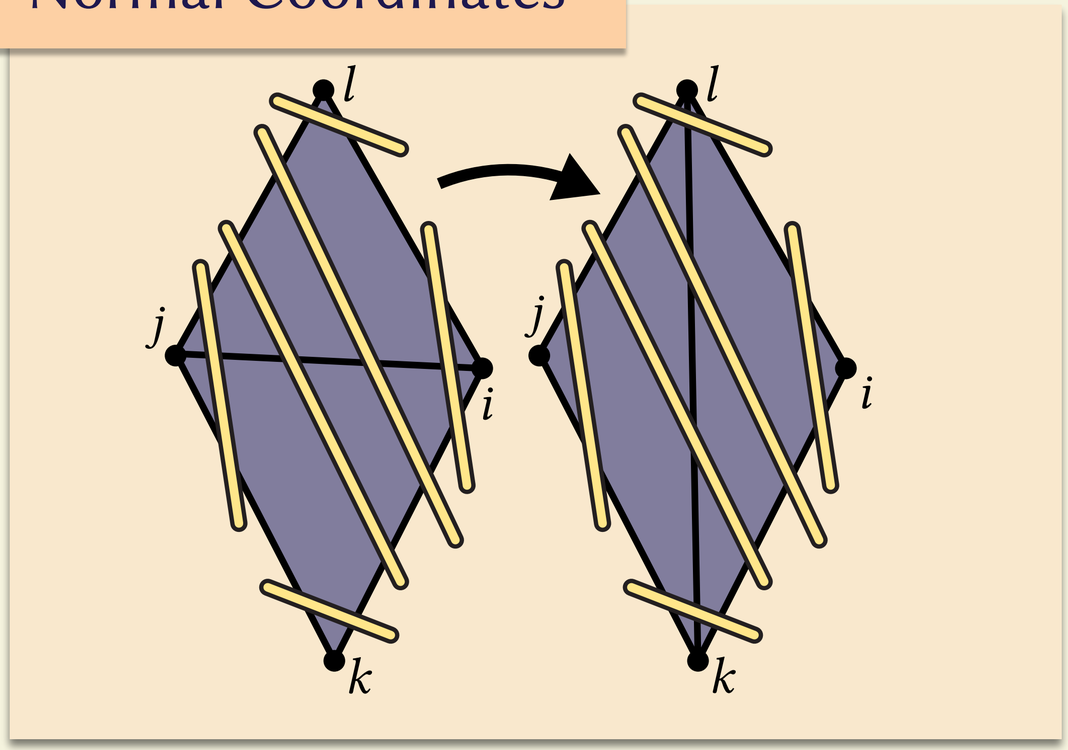


store on intrinsic edges

Edge flips

edge
curve

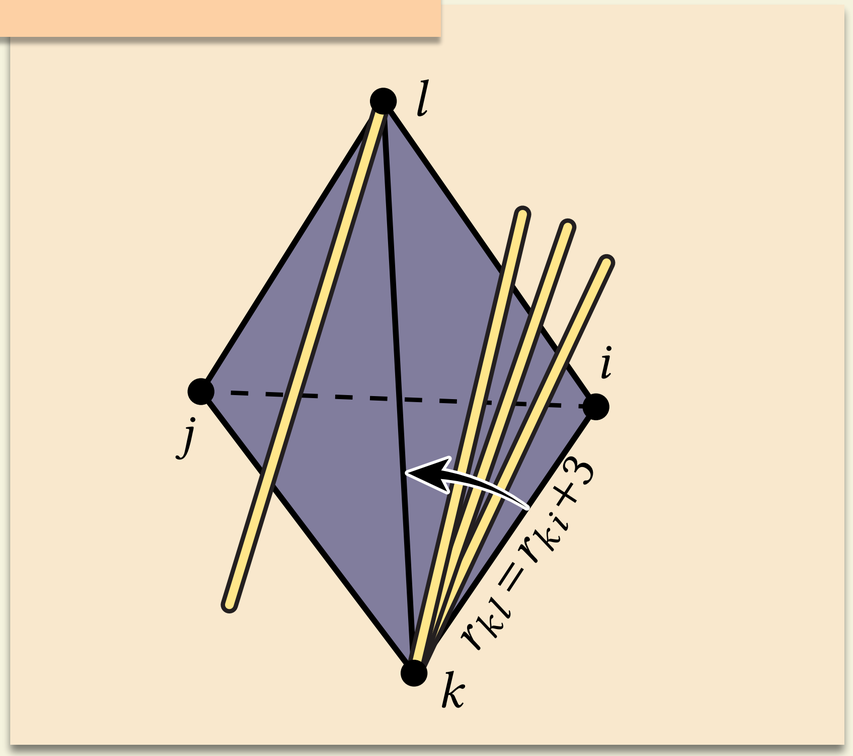
Normal Coordinates



$$n_{kl} = \max(n_{ki} + n_{lj}, n_{jk} + n_{li}) - n_{ij}$$

(if there are no endpoints)

Roundabouts



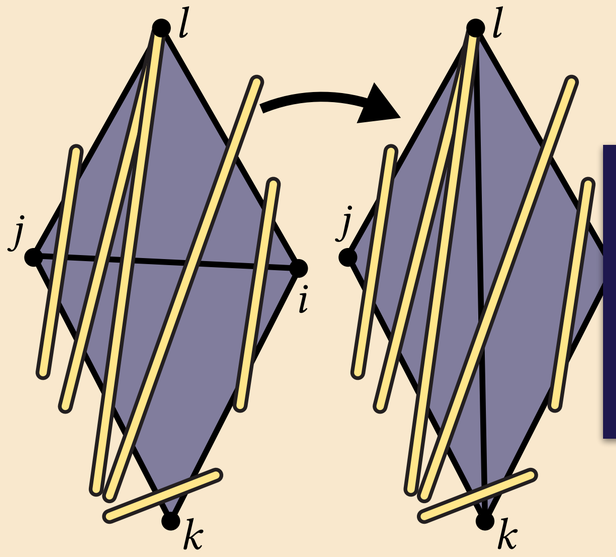
$$r_{kl} = r_{ki} + n_{ki}^{-} + \max(0,\, n_{il}^{+} - n_{lk}^{+} - n_{ki}^{+})$$

Edge flips

edge

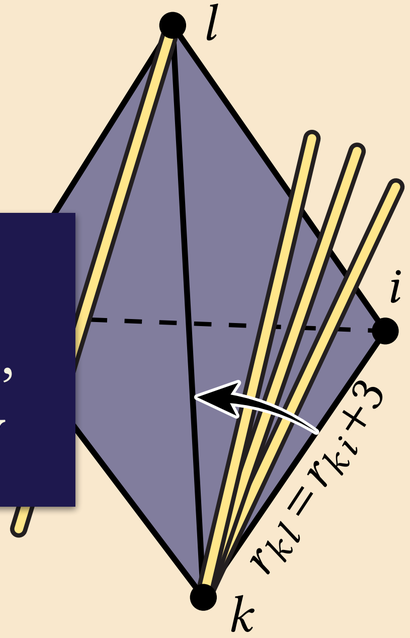
curve

Normal Coordinates



Key takeaway:
closed form flip formulas, independent of geometry

Roundabouts



$$n_{kl} = c_l^{jk} + c_k^{ij} + \frac{1}{2}\left|c_j^{il} - c_j^{ki}\right| + \frac{1}{2}\left|c_i^{lj} - c_i^{jk}\right| - \frac{1}{2}e_l^{ji} - \frac{1}{2}e_k^{ij} + e_i^{lj} + e_i^{jk} + e_j^{il} + e_j^{ki} + n_{ij}^{-}$$

(general case)

where

$$2c_k^{ij} := \max(0, n_{jk}^{+} + n_{ki}^{+} - n_{ij}^{+}) - e_i^{jk} - e_j^{ki}$$

$$e_k^{ij} := \max(0, n_{ij}^{+} - n_{jk}^{+} - n_{ki}^{+})$$

$$r_{kl} = r_{ki} + n_{ki}^{-} + \max(0, n_{il}^{+} - n_{lk}^{+} - n_{ki}^{+})$$

Vertex insertion

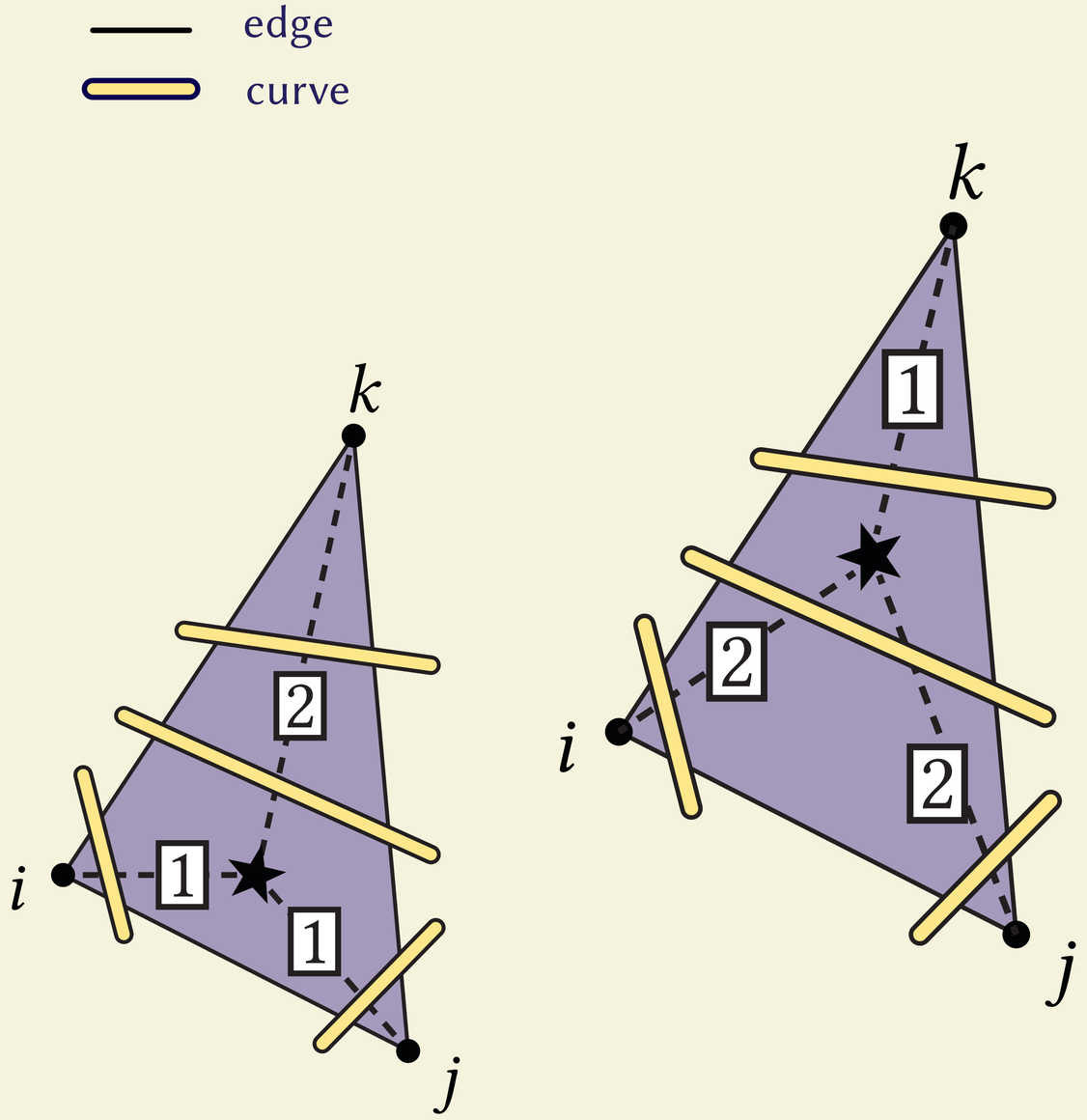
- *Unlike* classic normal coordinates, depends on geometry
- Not a computational challenge:

  1. Locate crossing curves using normal coordinates

  2. Count intersections

  3. Update roundabouts

  4. *(For convenience):* Record inserted vertex position on original mesh

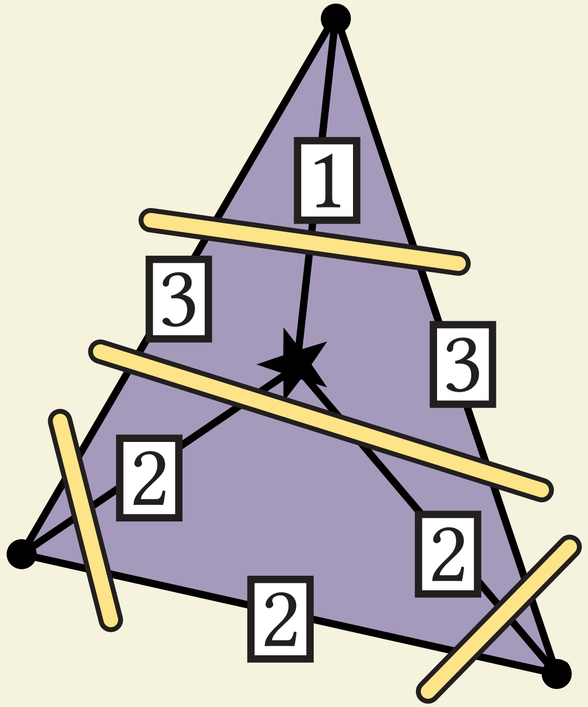


Vertex removal

edge

curve

- Only remove inserted vertices
- Strategy: reduce to degree-3 case

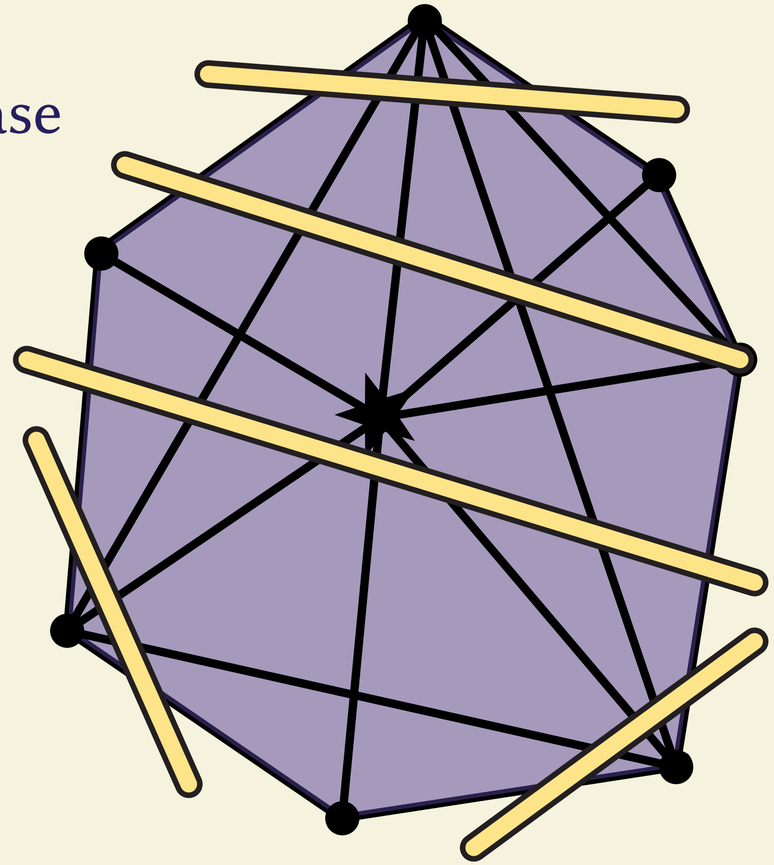


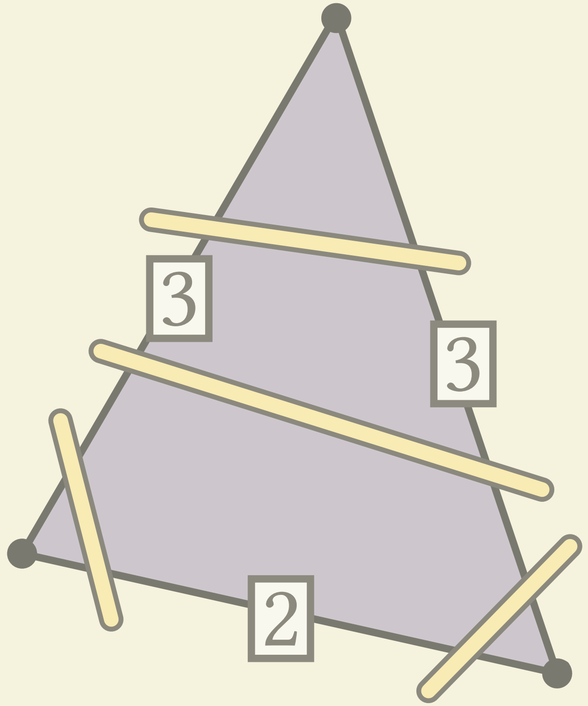
Vertex removal

edge

curve

- Only remove inserted vertices
- Strategy: reduce to degree-3 case



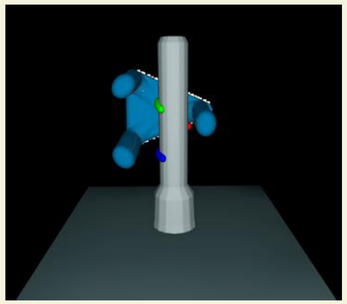


Robustness of intrinsic triangulations

Intrinsic Delaunay refinement — validation

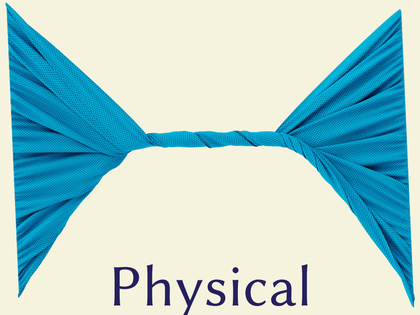
- Compute refinements for Thingi10k dataset [Zhou & Jacobson 2016]
  - 7696 manifold meshes
- 100% success rate
  - [Sharp *et al.* 2019] succeed on only 69.1% of meshes

Usage

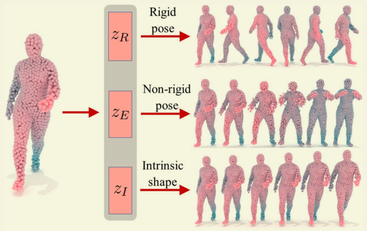


Robotics
[Lakshmipathy *et al.* 2024]

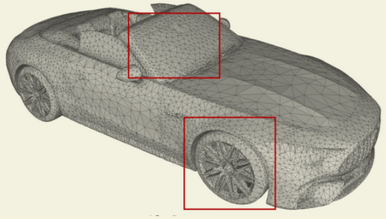
Cell Modeling
[Numerow *et al.* 2024]



Physical Simulation
[Zhang *et al.* 2023]

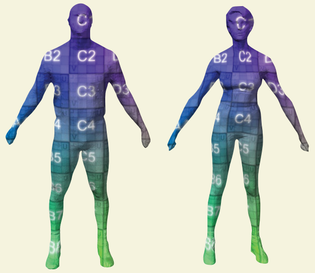


Geometric ML
[Aumentado-Armstrong *et al.* 2023]

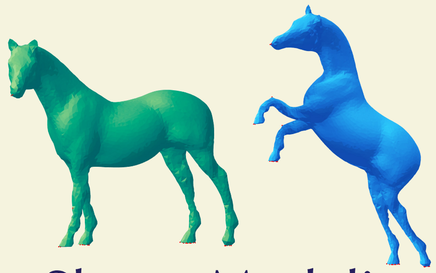


Extrinsic Remeshing
[Dai *et al.* 2024]

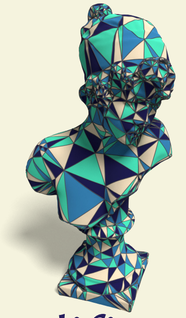
Parameterization
[Fargion & Weber 2022]
[Wang *et al.* 2022]
[Akalin *et al.* 2024]



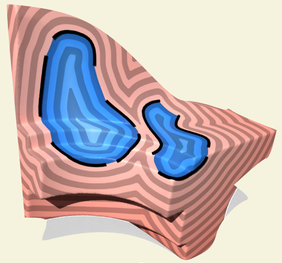
Surface Correspondence
[Takayama 2022]



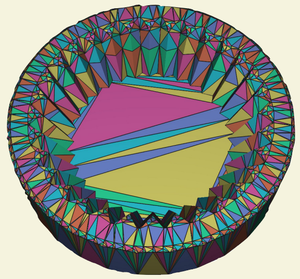
Shape Modeling
[Finnendahl *et al.* 2023]



Simplification
[Shoemaker *et al.* 2023]

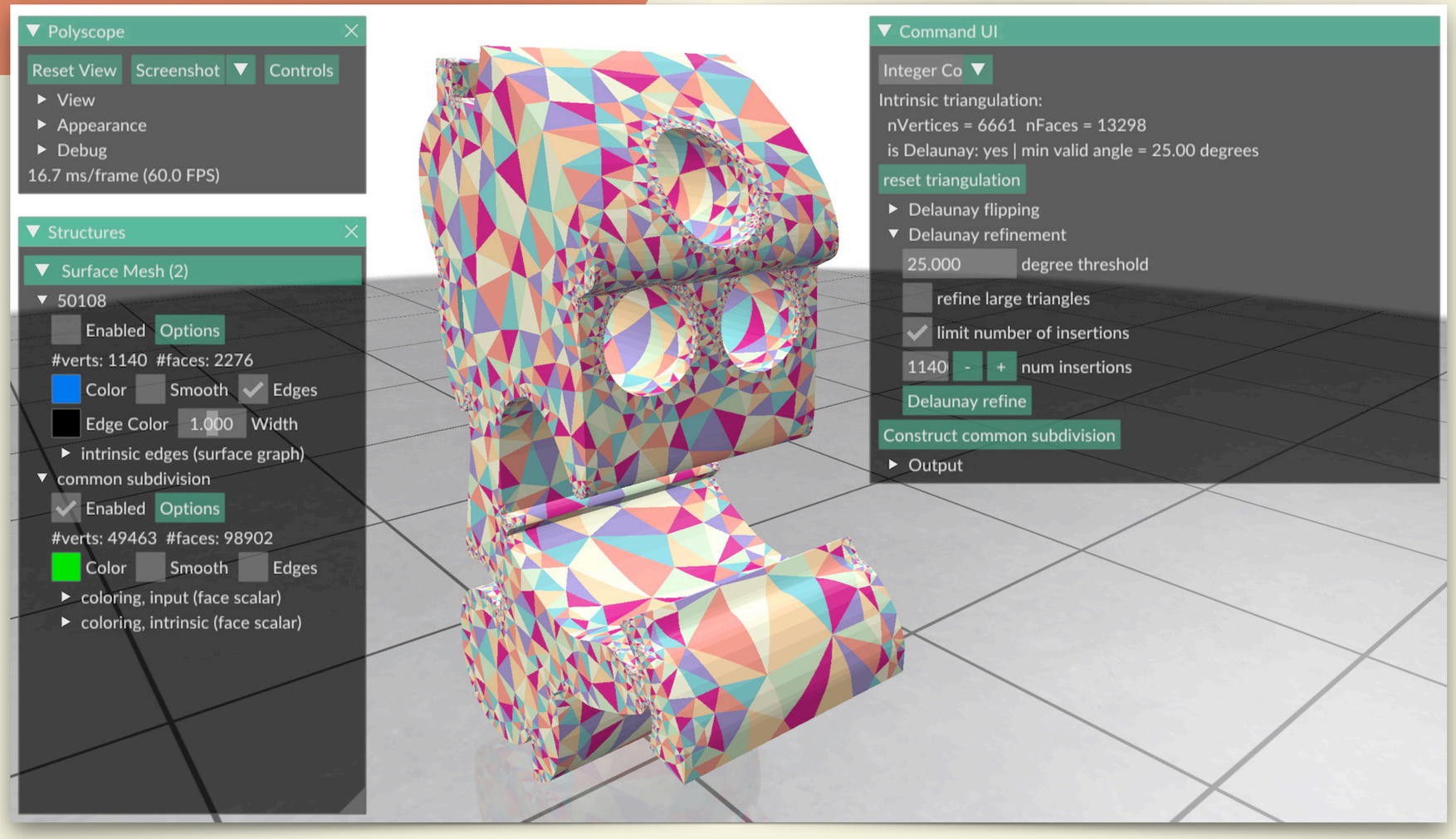


Surface Distances
[Feng & Crane 2024]



Rendering
[Celes 2025]

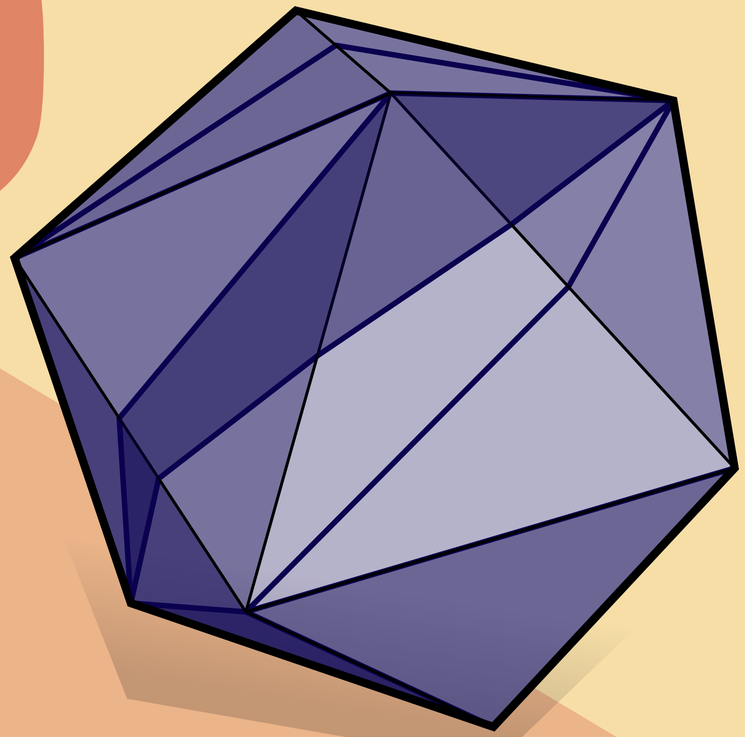
Try it out yourself



https://github.com/MarkGillespie/intrinsic-triangulations-demo

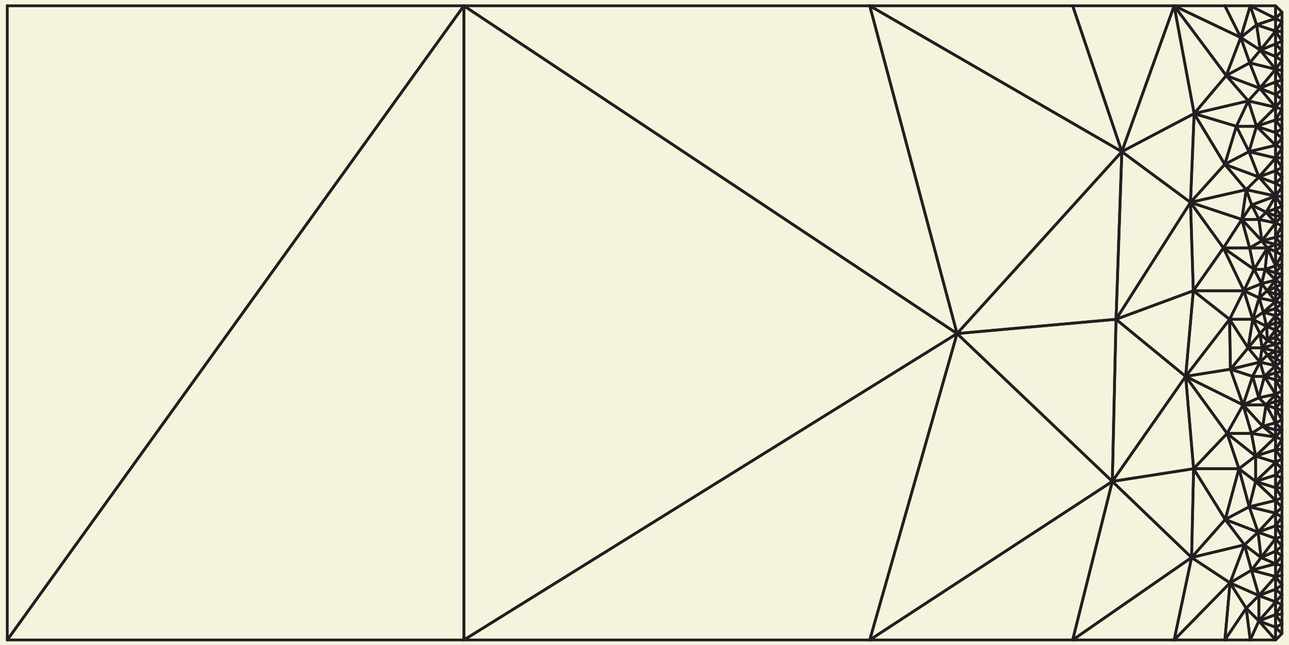
✉ mark.gillespie81@gmail.com
markjgillespie.com

Thanks for listening

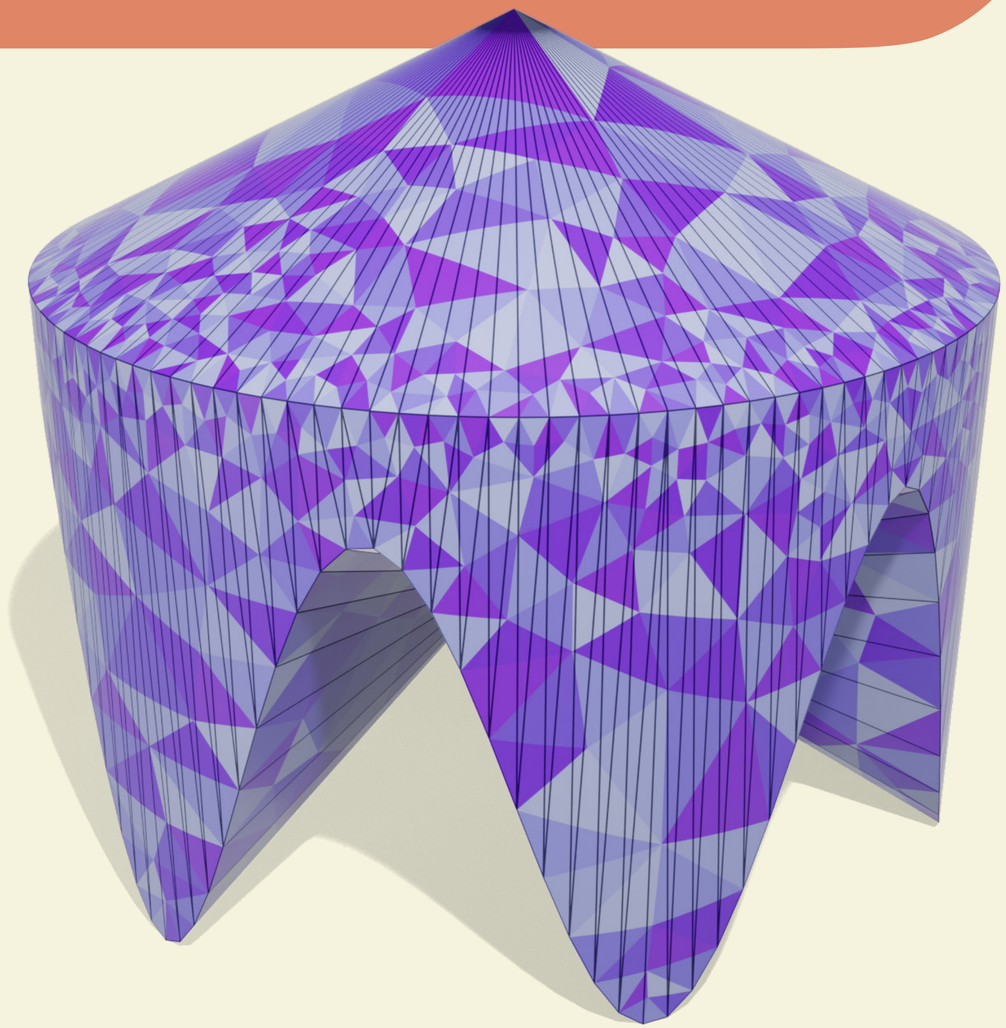


Supplemental Slides

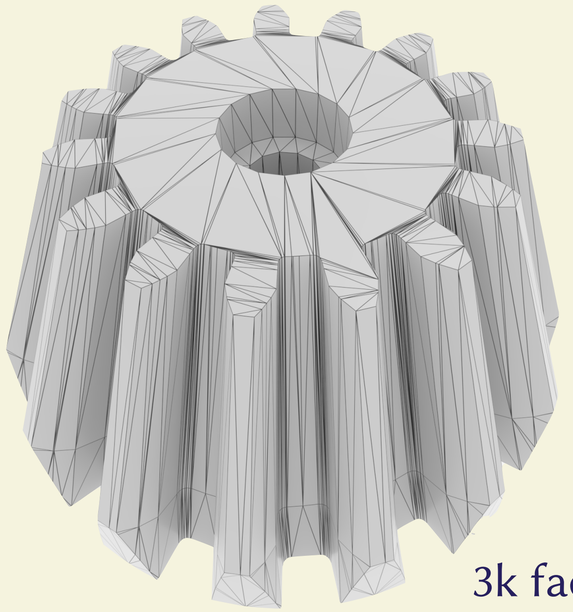
Intrinsic Delaunay refinement size grading



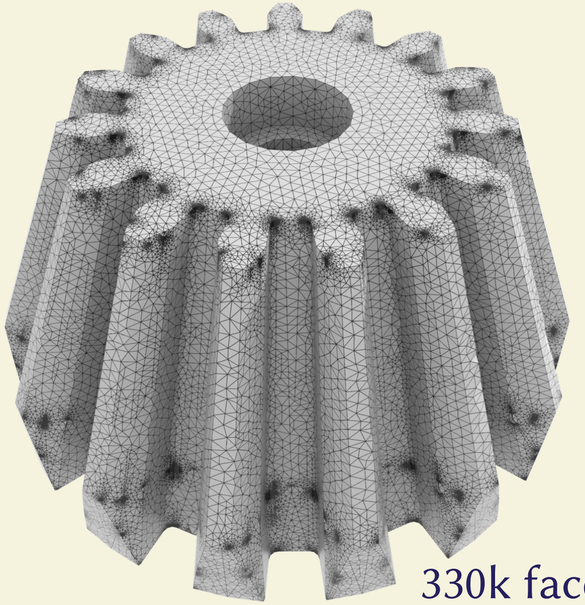
[Shewchuk 1997]



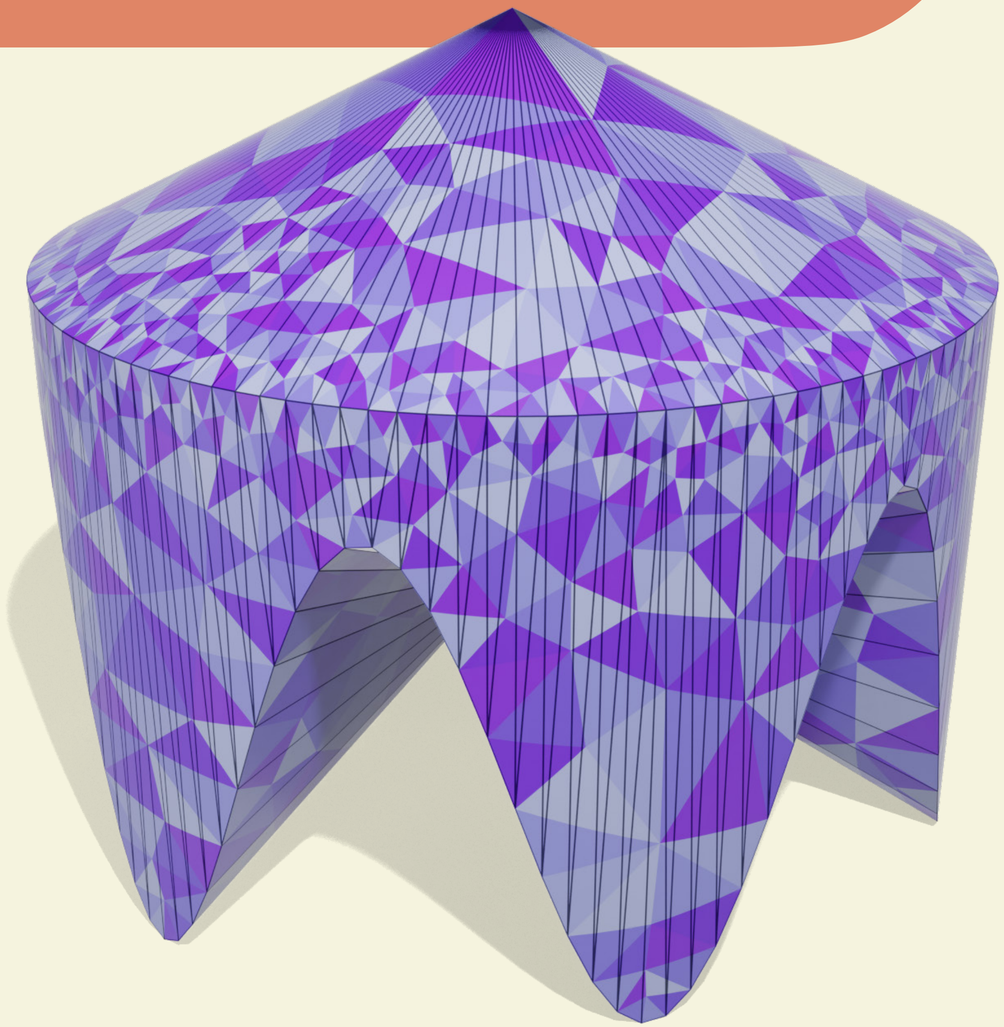
Intrinsic Delaunay refinement size grading



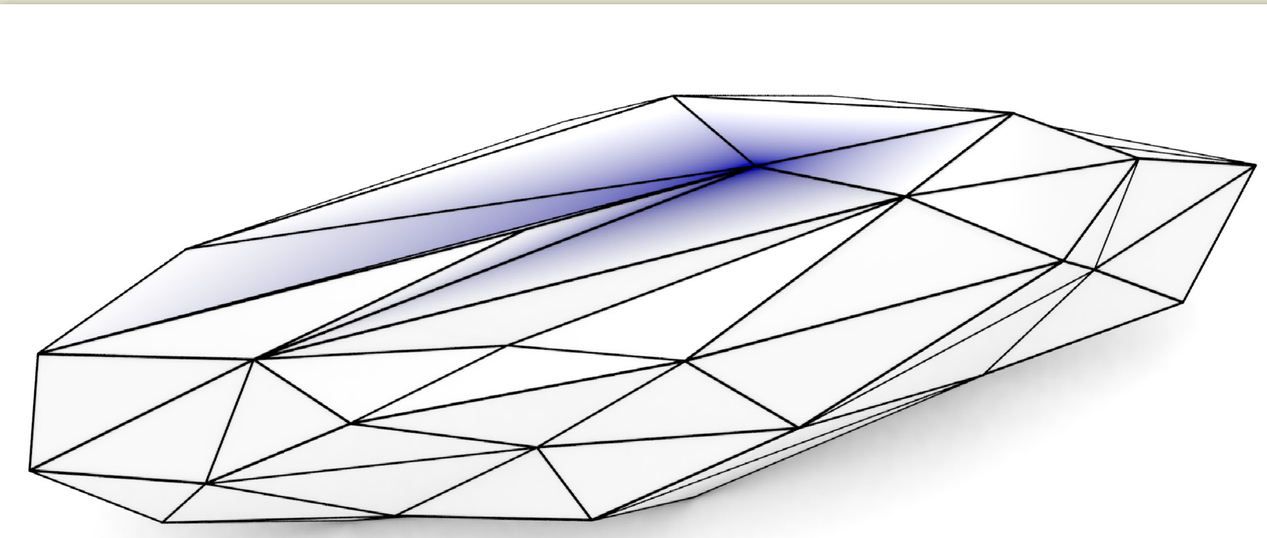
3k faces



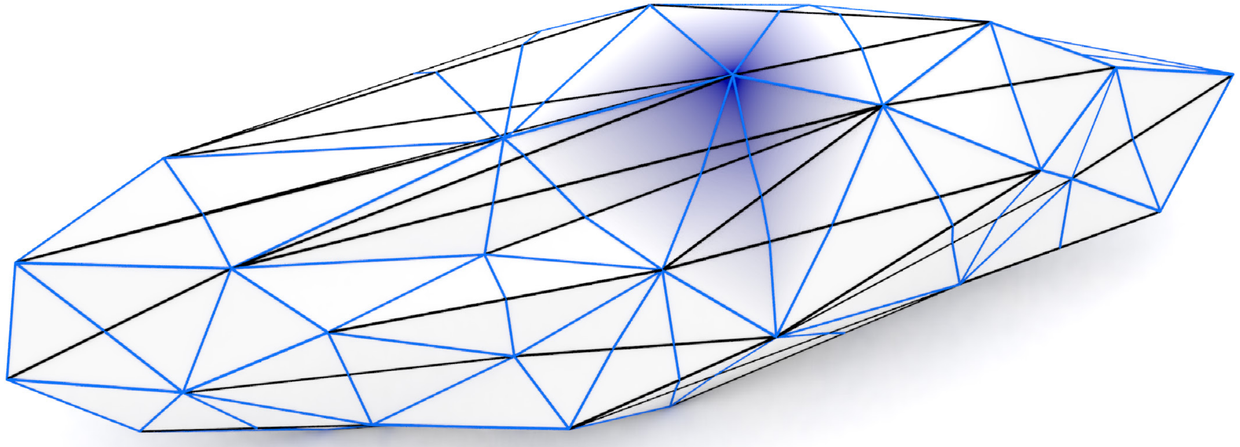
330k faces



Bad basis functions



Input basis function

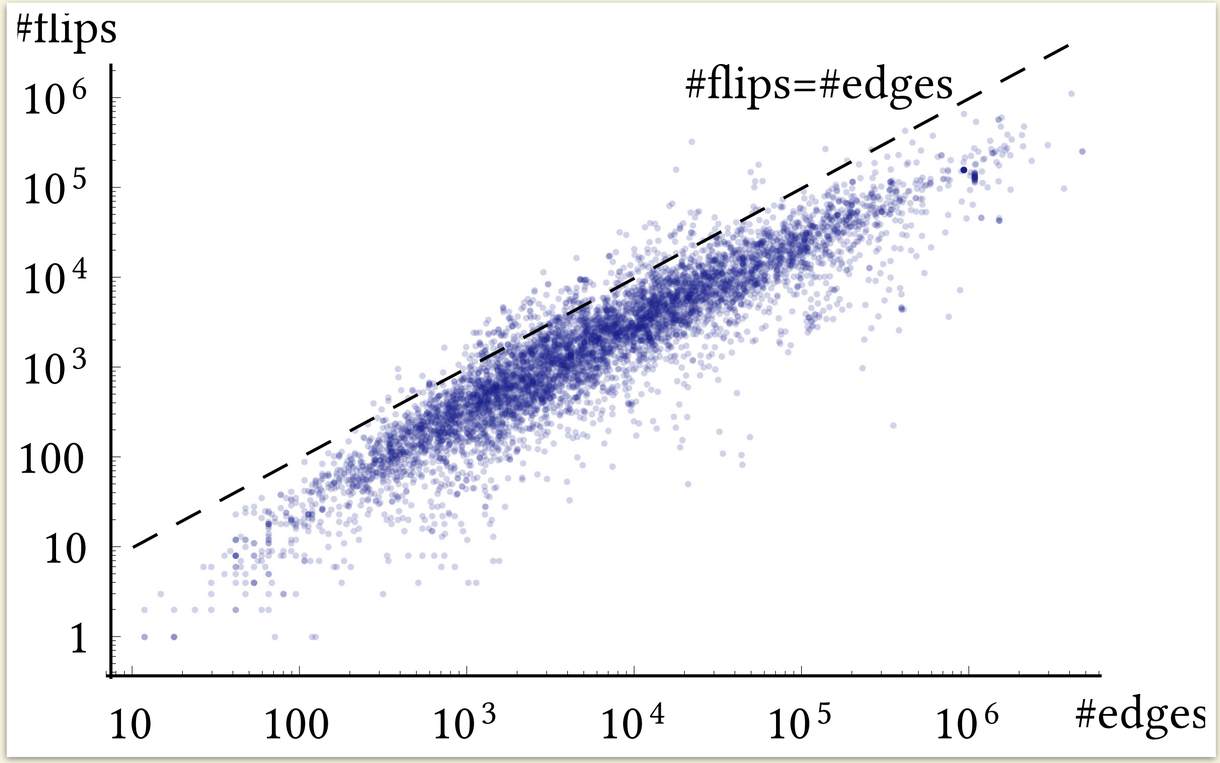


Intrinsic basis function

[Sharp, Soliman & Crane 2019]

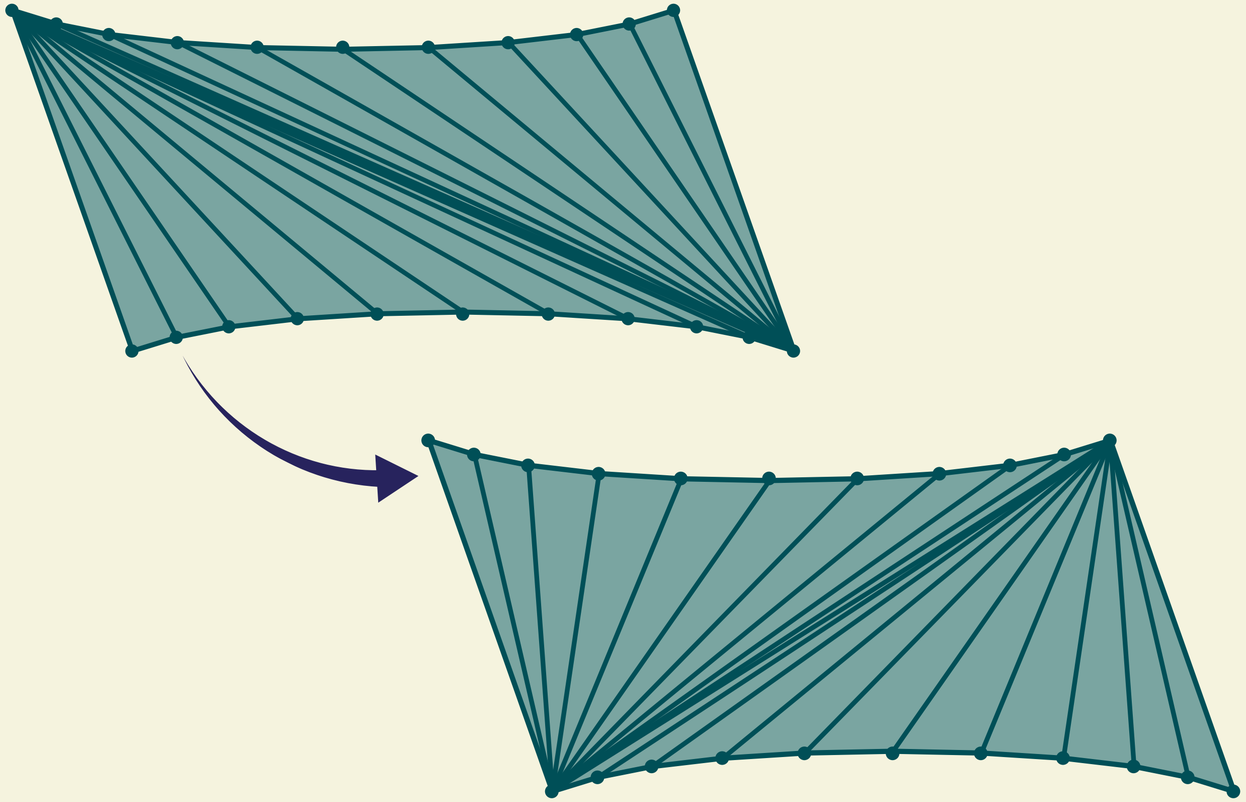
Intrinsic Delaunay flip complexity

Empirically, usually linear time



[Sharp, Soliman & Crane 2019]

$\Omega(n^2)$ worst-case examples in the plane



Exact isometric embeddings

(intrinsically) convex polyhedra

unique convex embedding into $\mathbb{R}^3$
[Alexandrov 1942]

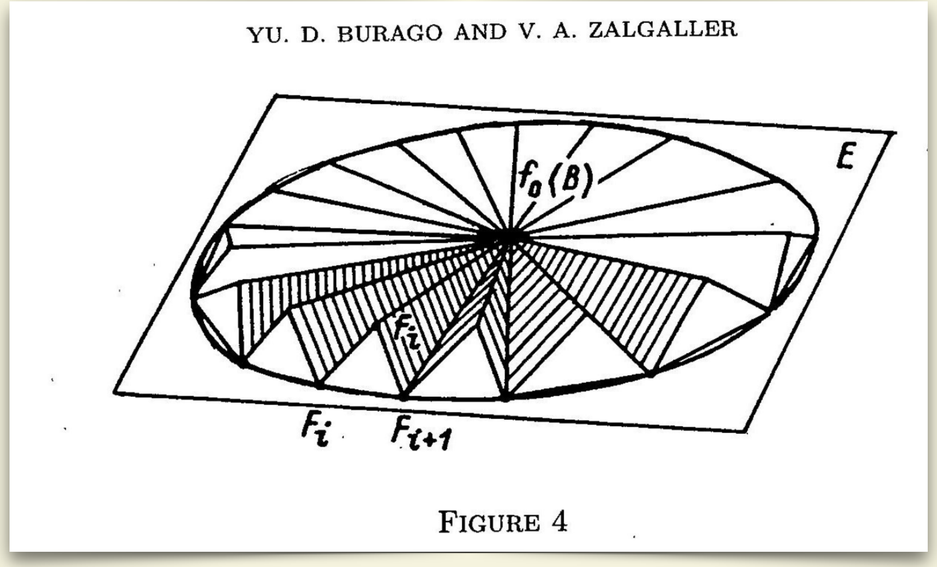
(may need to flip edges)

constructive proof/algorithm
[Bobenko & Izmestiev 2008]

general polyhedra

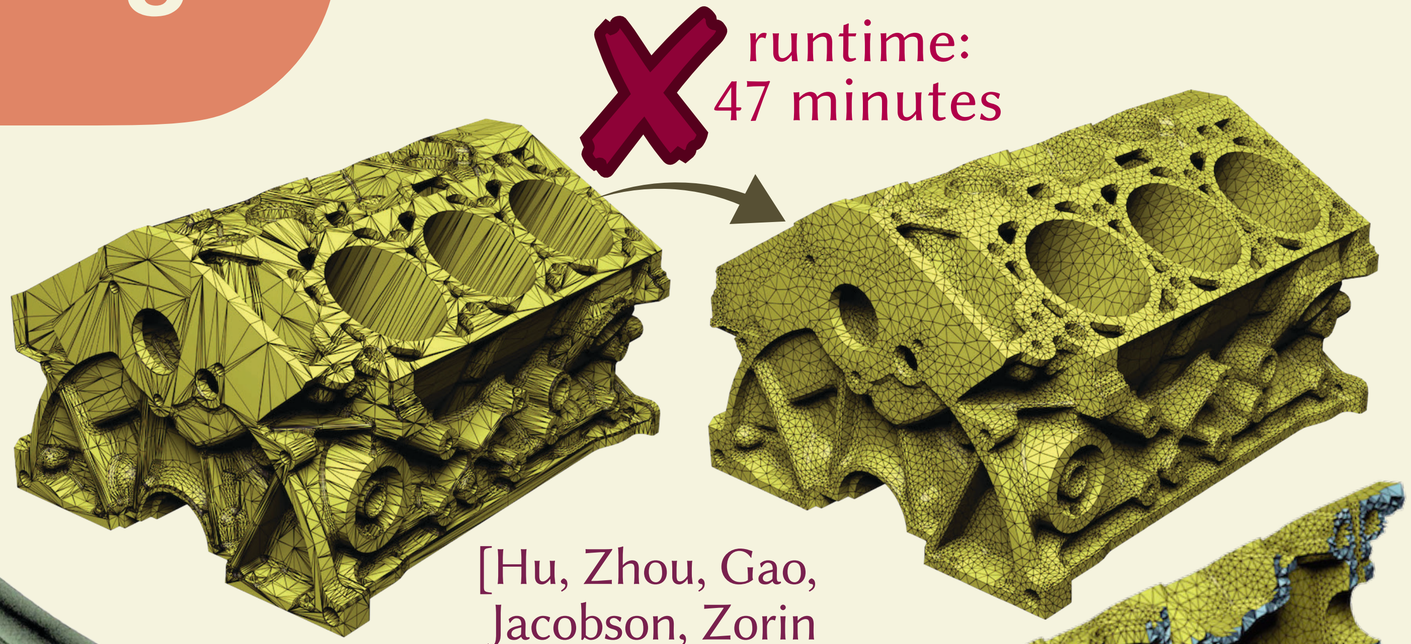
many embeddings into $\mathbb{R}^3$
[Burago & Zalgaller 1960, 1995]

(may need to subdivide mesh many times)



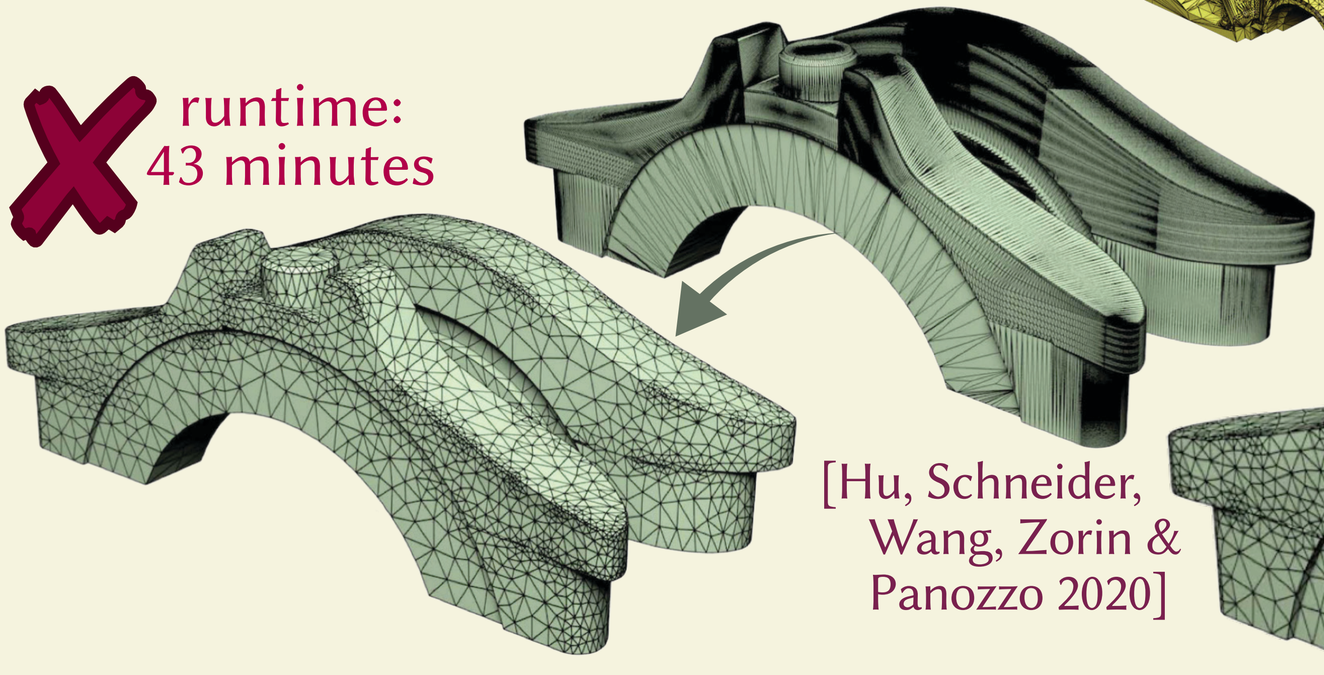
Status quo: remeshing

- State-of-the-art is robust but slow
  - Techniques work volumetrically



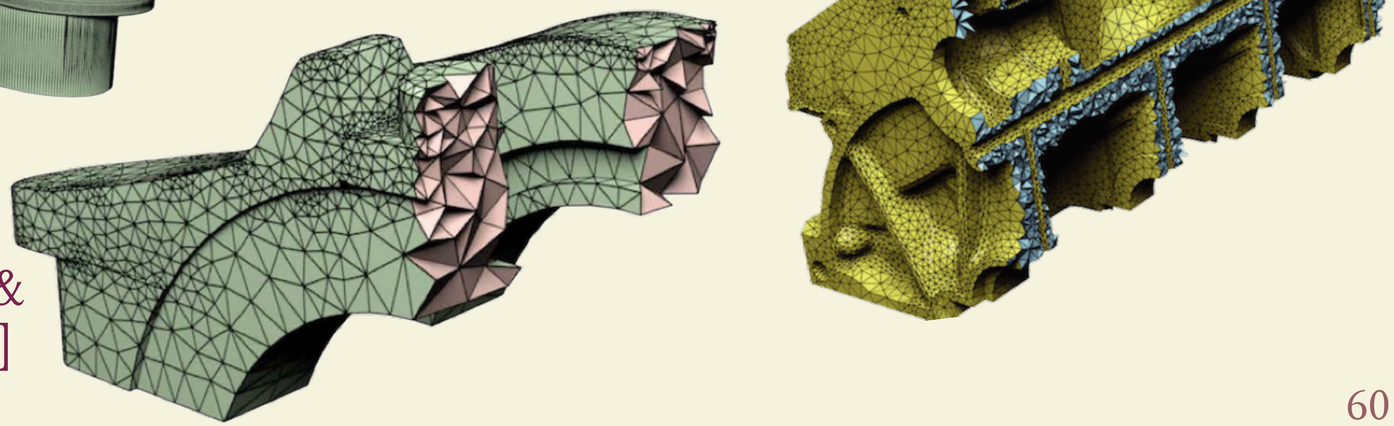
runtime:
47 minutes

[Hu, Zhou, Gao, Jacobson, Zorin & Panozzo 2018]



runtime:
43 minutes

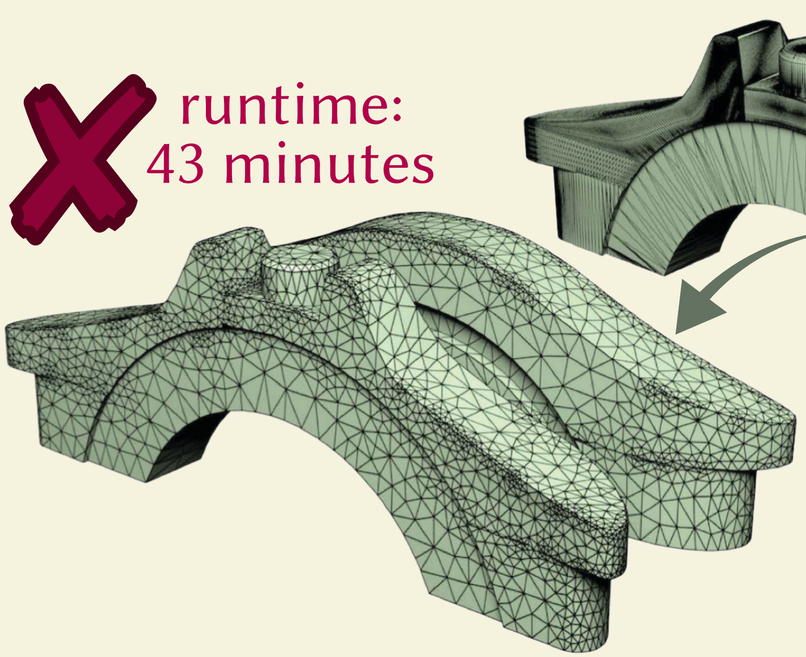
[Hu, Schneider, Wang, Zorin & Panozzo 2020]



Status quo: remeshing

- State-of-the-art is robust but slow
  - Techniques work volumetrically

runtime:
43 minutes



[Hu, Schneider, Wang, Zorin & Panozzo 2020]

runtime:
47 minutes

Meshing is much easier in 2D

strong guarantees on quality

Generate high-quality meshes in milliseconds

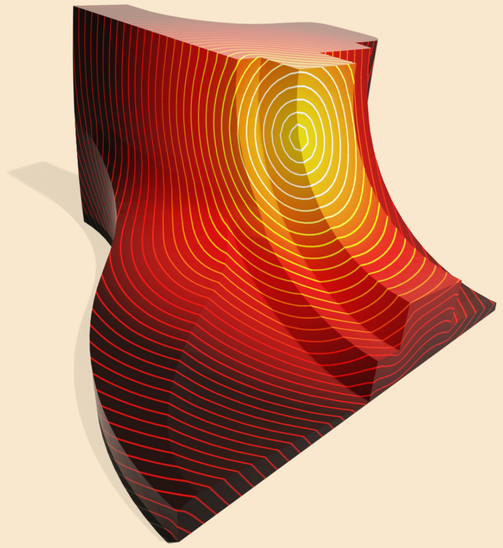
via Delaunay refinement [Chew 1993; Shewchuk 1997]

[Shewchuk 1997]

80 milliseconds

70 milliseconds

Example: Approximate Shortest Paths



heat method for surface distances

[Crane, Weischedel & Wardetzky 2013]

|V|=2948

input

mean error: 28%

|V|=2948

intrinsic Delaunay triangulation

mean error: 7%

|V|=11954

intrinsic Delaunay refinement

mean error: 2%

Boundary handling

[Shewchuk 1997]

3.4.1 Description of the Algorithm

Chew's algorithm begins with the constrained Delaunay triangulation of a segment-bounded PSLG, and uses Delaunay refinement with locked subsegments and a quality bound of $B = 1$, but there is no idea of encroached diametral circles. However, it may arise that a skinny triangle t cannot be split because t and its circumcenter c lie on opposite sides of a subsegment s (possibly with c outside the triangulation). Because s is locked, inserting a vertex at c will not remove t from the mesh. Instead, c is rejected for insertion, and all free vertices (but not input vertices or vertices that lie on segments) that lie in the interior of the diametral circle of s and are visible from the midpoint of s are deleted. Then, a new vertex is inserted at the midpoint of s. The Delaunay property is maintained throughout all deletions and insertions, except that locked subsegments are not flipped. Figure 3.20 illustrates a subsegment split in Chew's algorithm.

If several subsegments lie between t and c, only the subsegment visible from the interior of t is split.

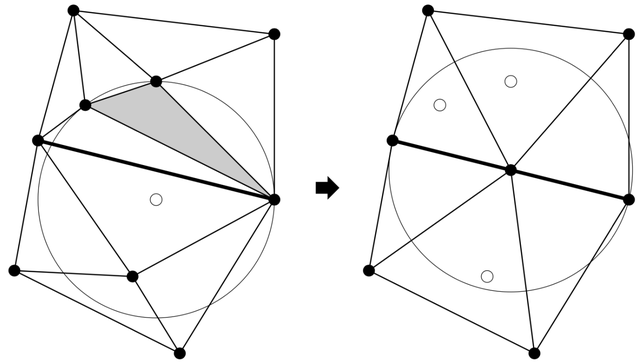
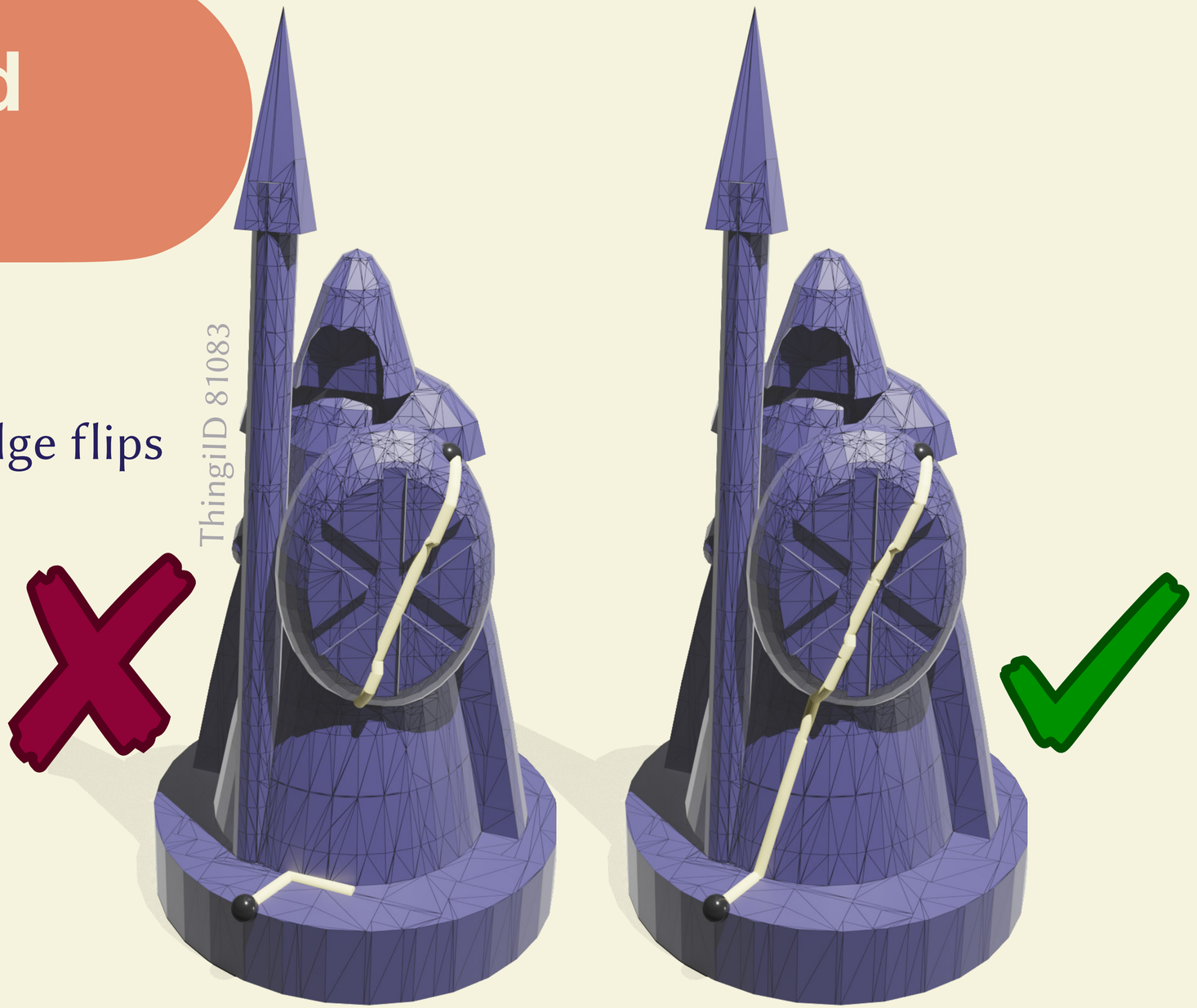


Figure 3.20: At left, a skinny triangle and its circumcenter lie on opposite sides of a subsegment. At right, all vertices in the subsegment's diametral circle have been deleted, and a new vertex has been inserted at the subsegment's midpoint.

Example: Flip-Based Geodesic Paths

- FlipOut [Sharp & Crane 2020]:
  - computes geodesic paths via edge flips

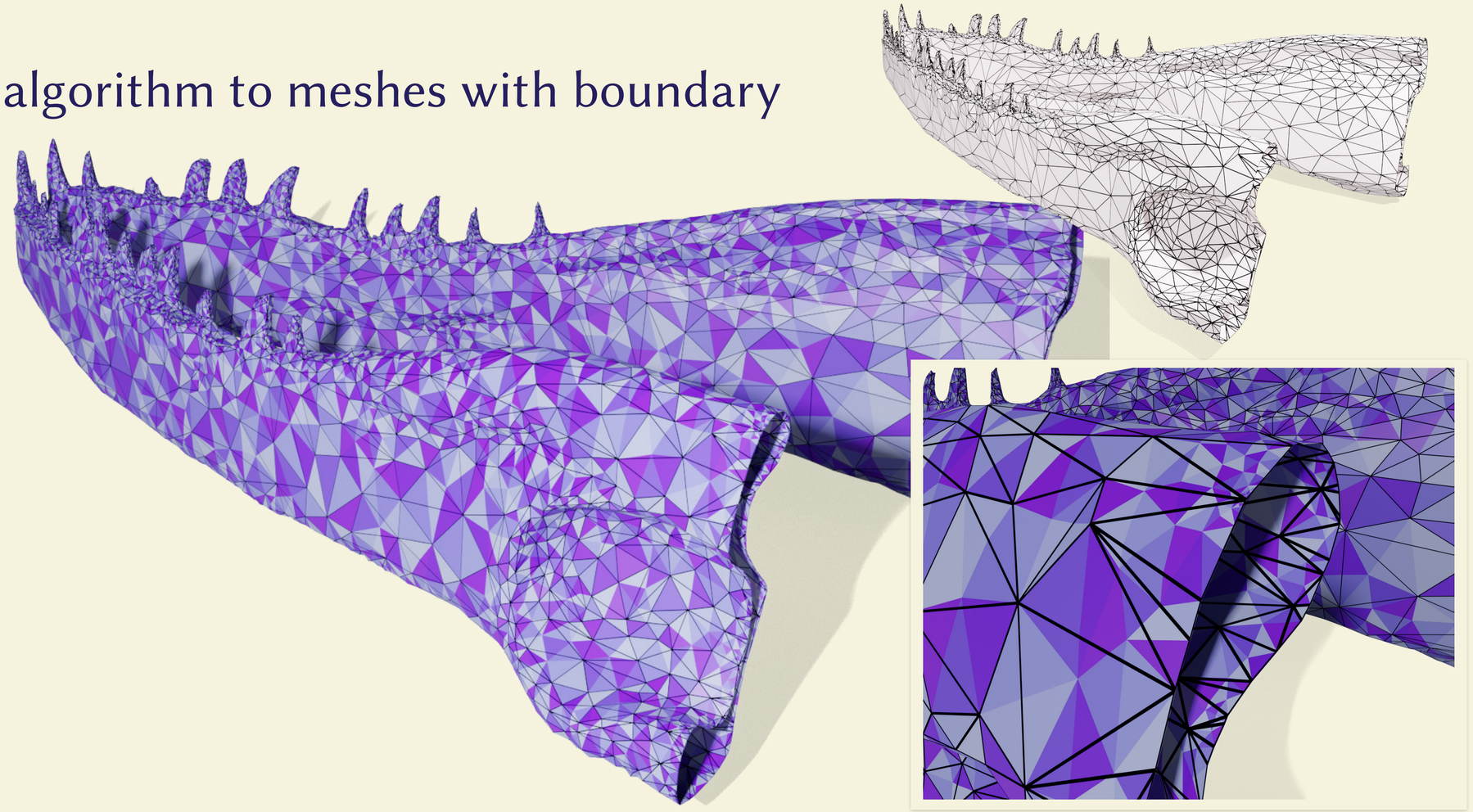


[Sharp, Soliman & Crane 2019]

[Ours]

Intrinsic Delaunay refinement of meshes with boundary

- Extend algorithm to meshes with boundary



ThingiID 48352