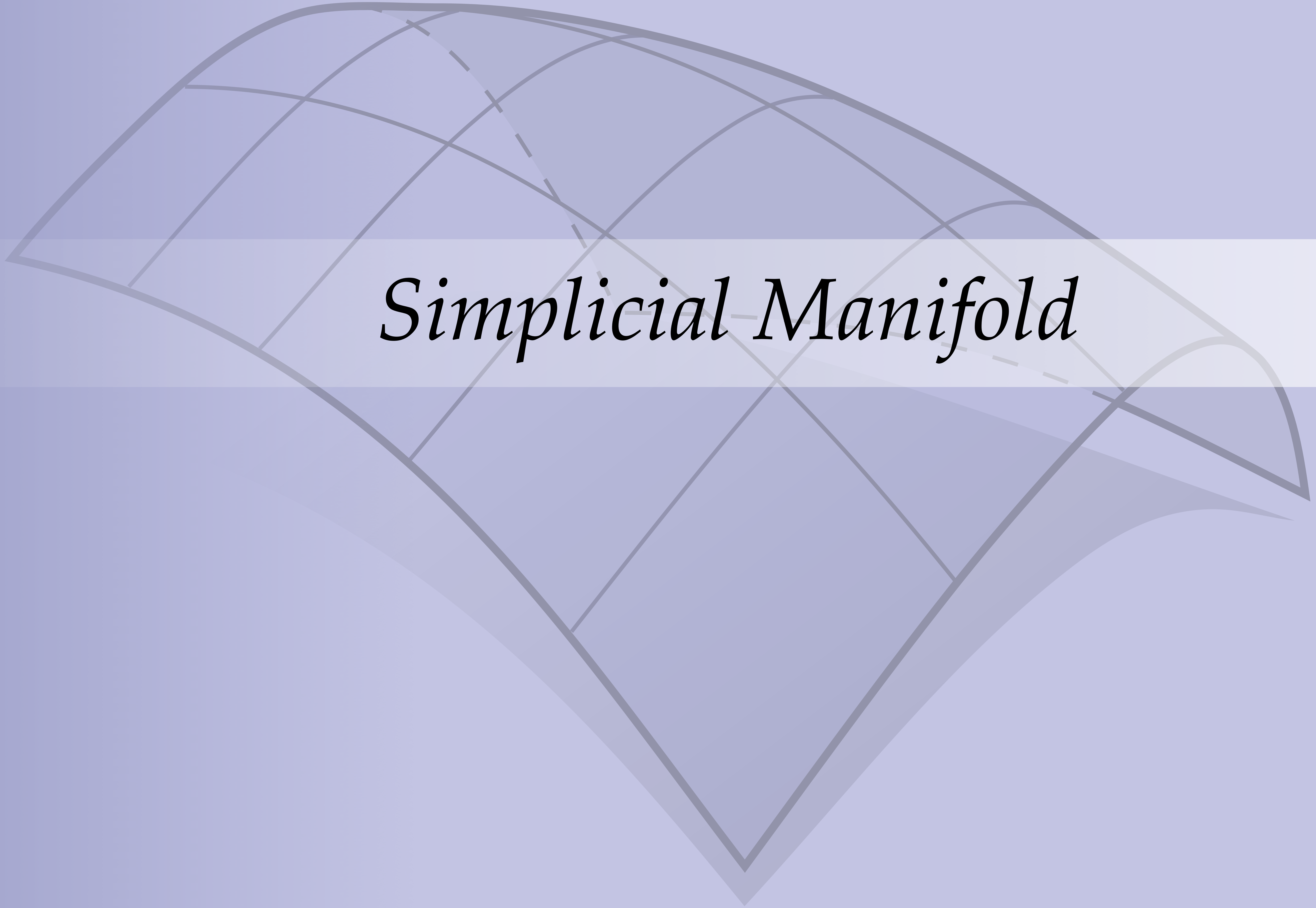
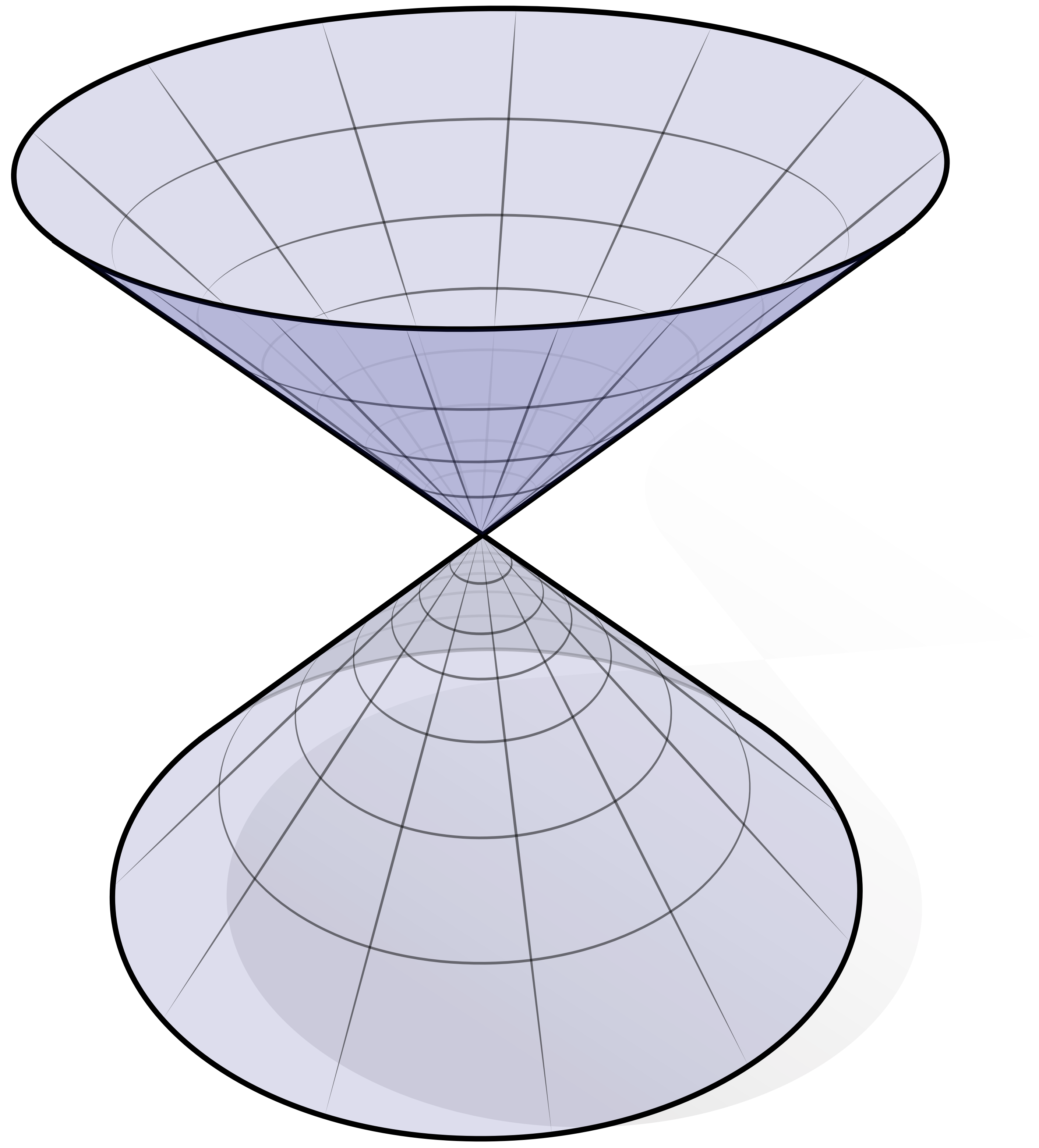
Simplicial Manifold



Manifold—First Glimpse

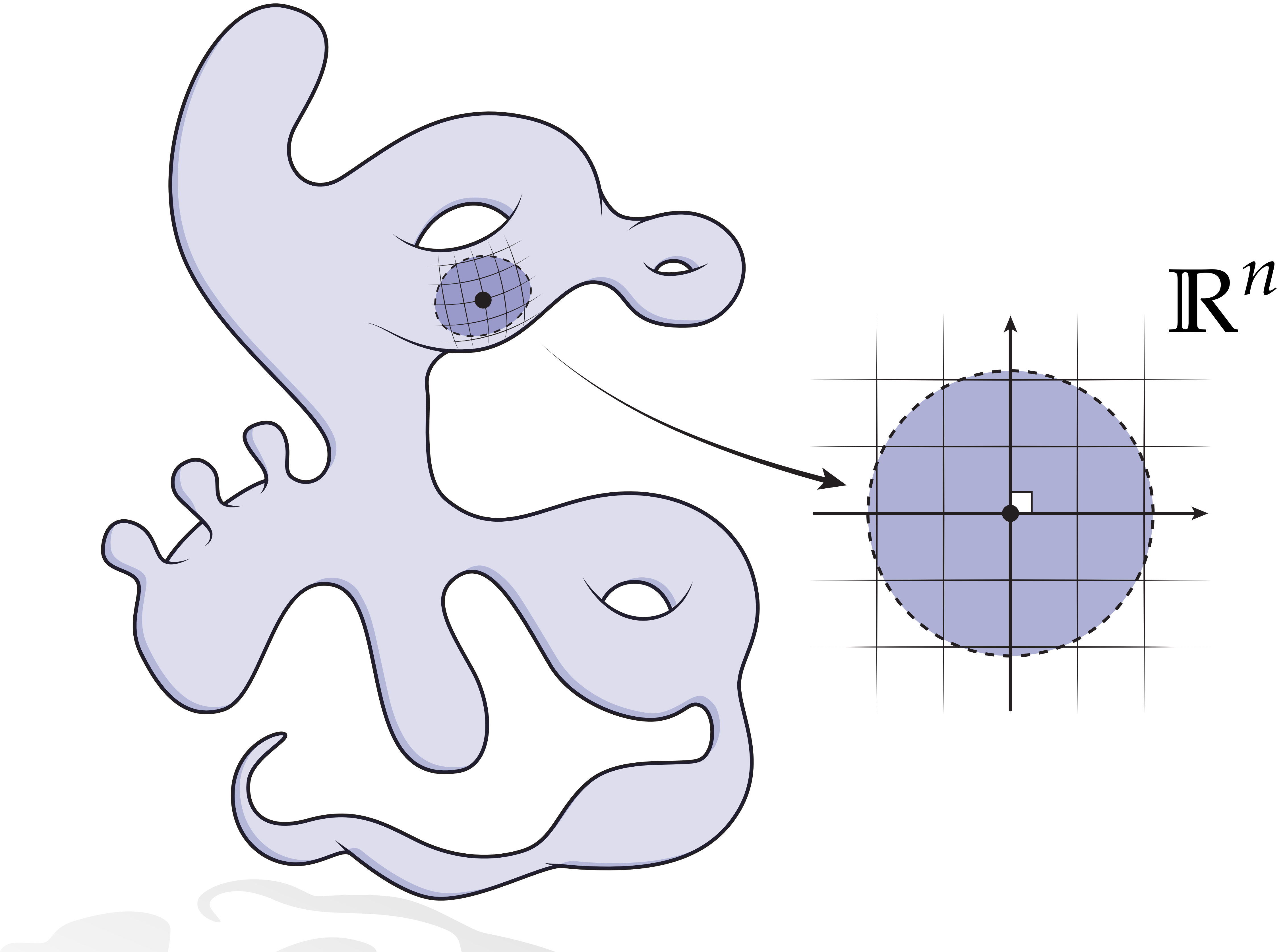
Very rough idea: notion of "nice" space in geometry.



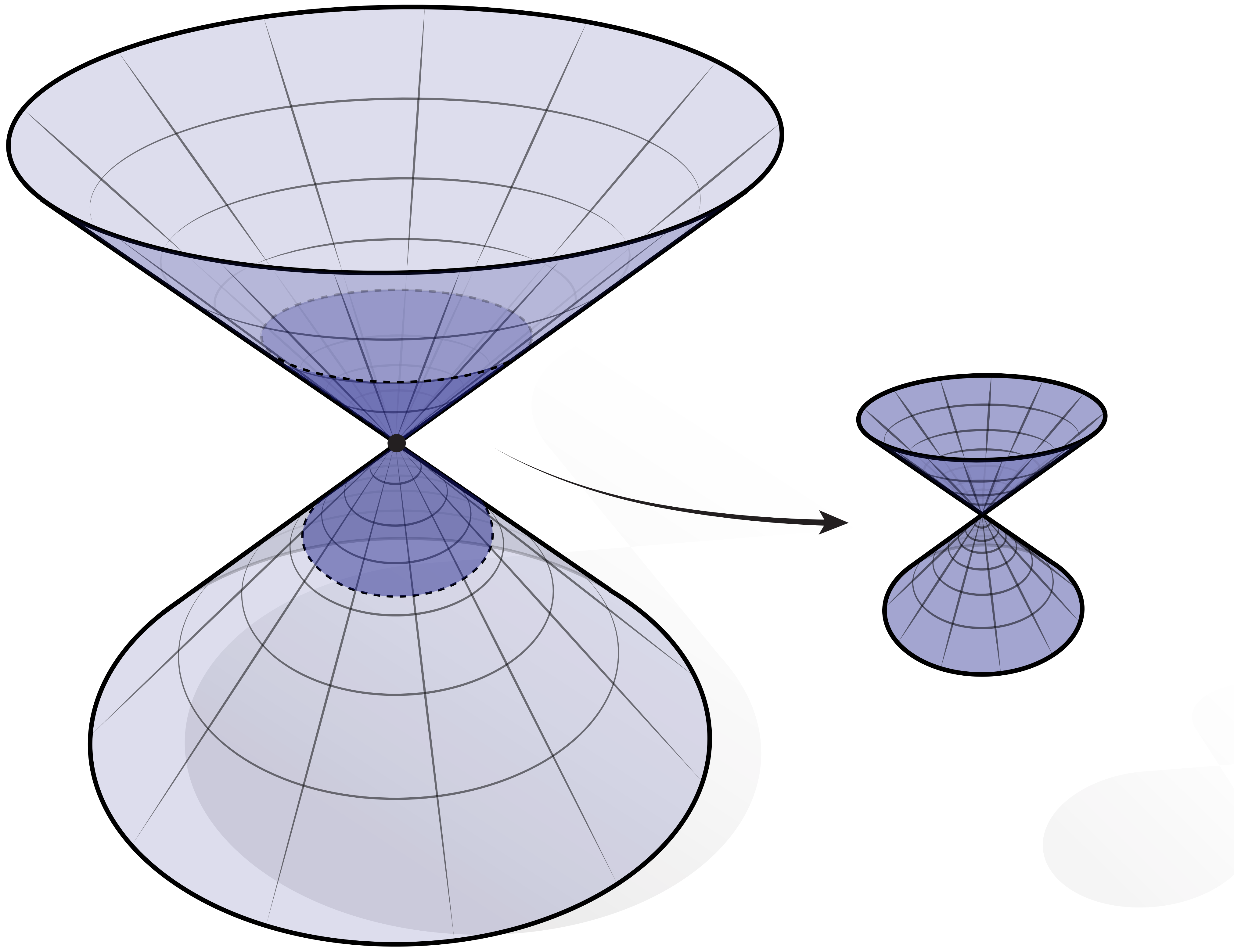
(Which one is "nice"?)

Manifold—First Glimpse

Key idea: manifold locally "looks like" $\mathbb{R}^n$



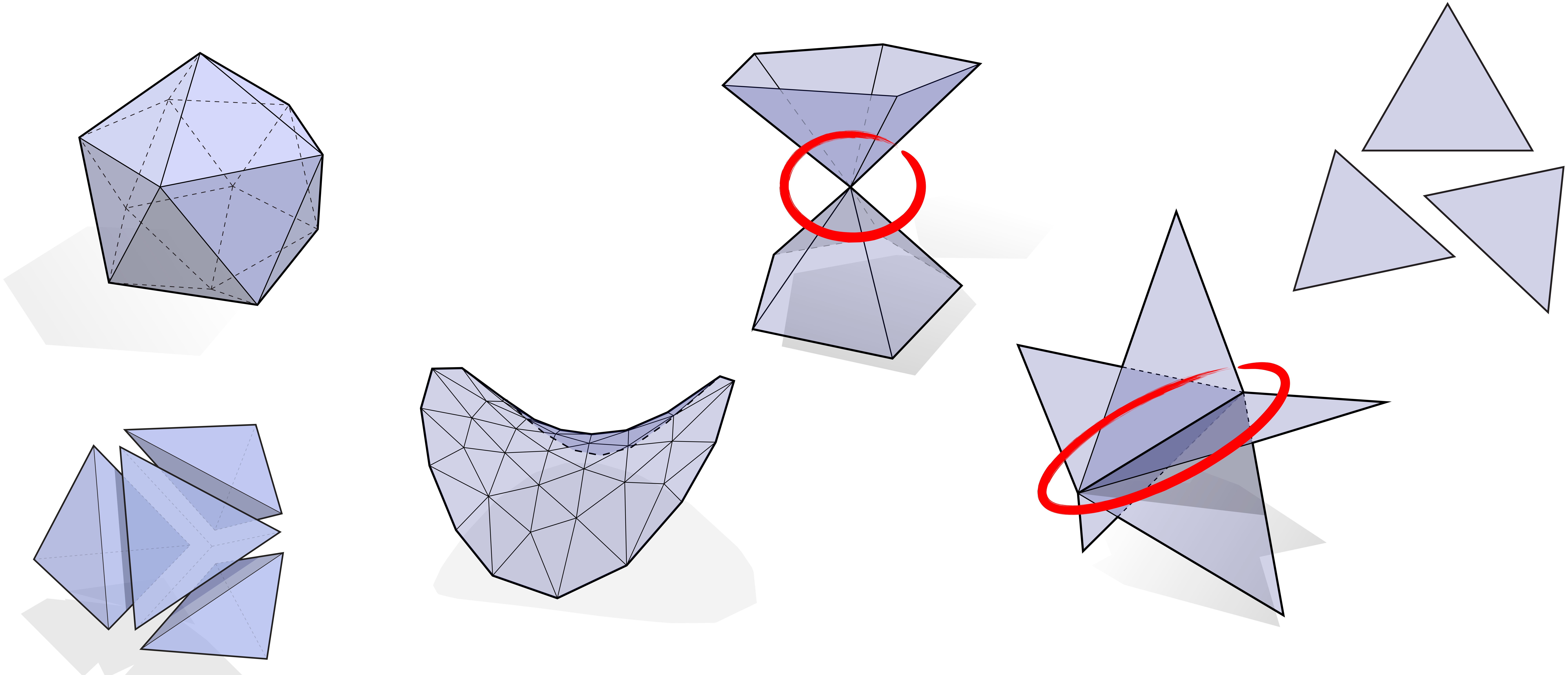
manifold



nonmanifold

Simplicial Manifold—Visualized

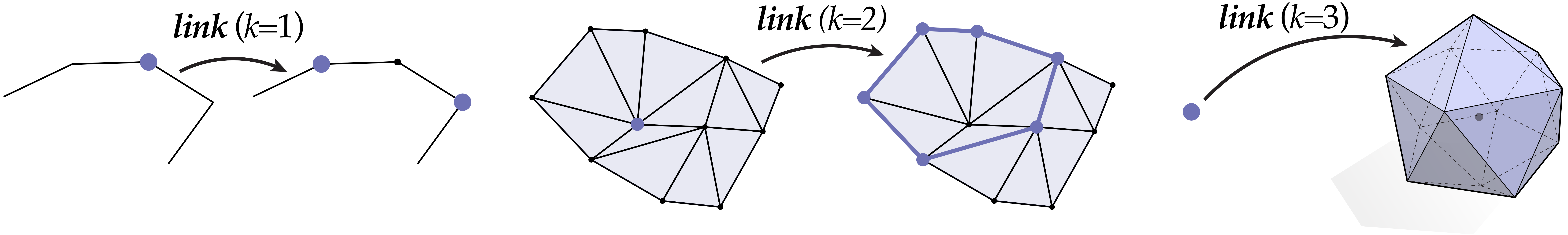
Which of these simplicial complexes look *"manifold?"*



(*E.g.*, where might it be hard to put a little *xy*-coordinate system?)

Simplicial Manifold—Definition

Definition. A simplicial k-complex is *manifold* if the **link** of every vertex looks like* a (k-1)-dimensional sphere.



Aside: How hard is it to check if a given simplicial complex is manifold?

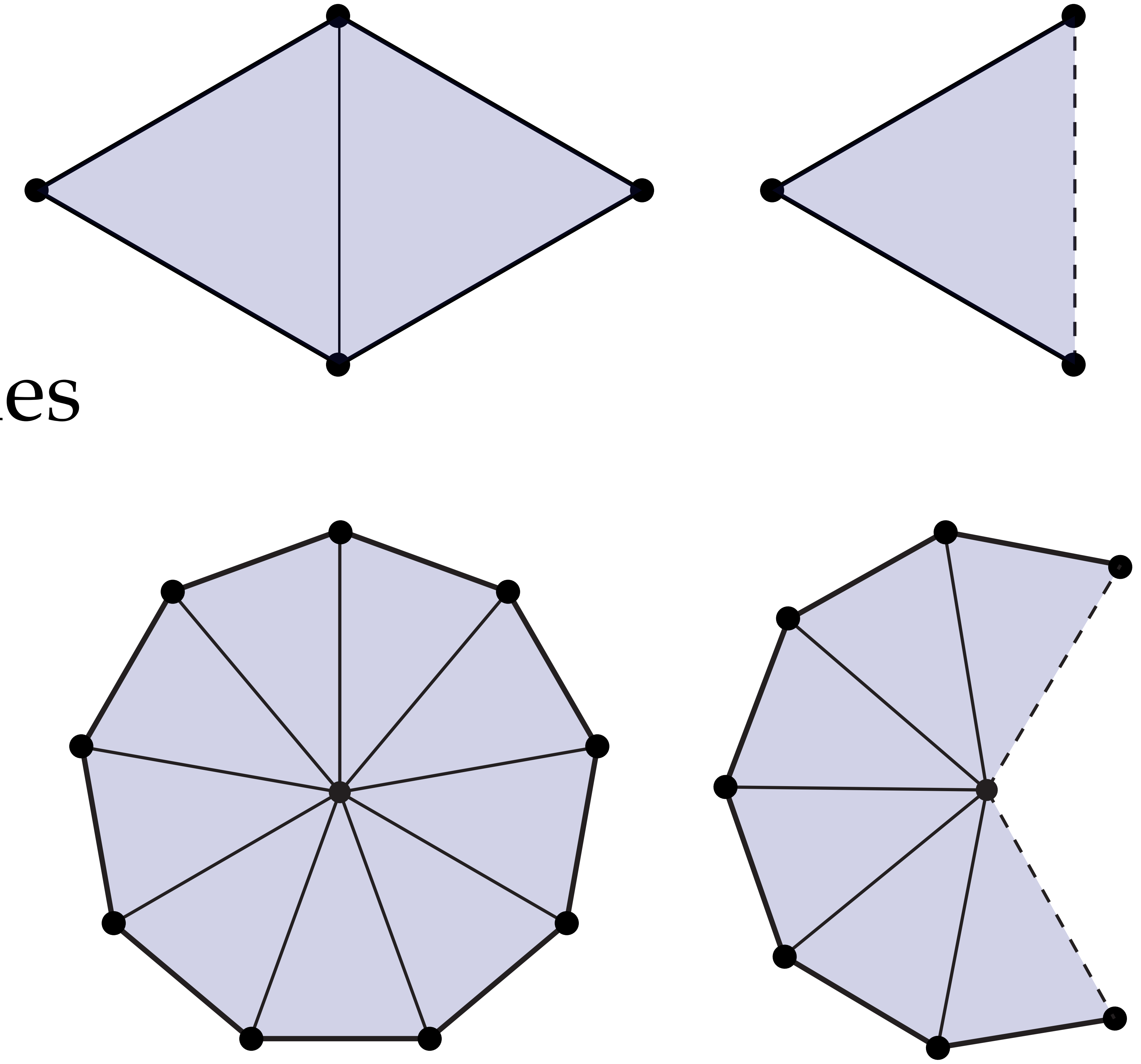
- (k=1) *easy*—is the whole complex just a collection of closed loops?
- (k=2) *easy*—is the link of every vertex a closed loop?
- (k=3) *easy*—is each link a 2-sphere? Just check if V-E+F = 2 (Euler's formula)
- (k=4) is each link a 3-sphere? …Well, it's known to be in NP! [S. Schleimer 2004]

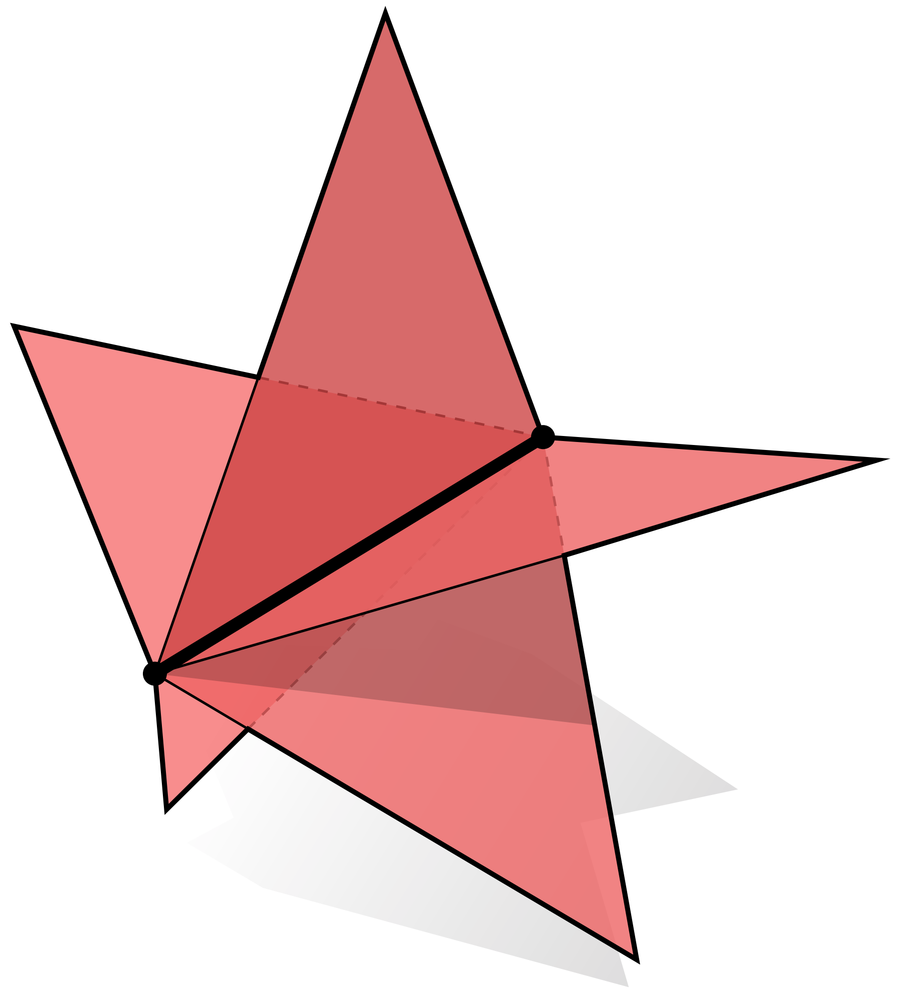
*i.e., is *homeomorphic* to.

Manifold Triangle Mesh

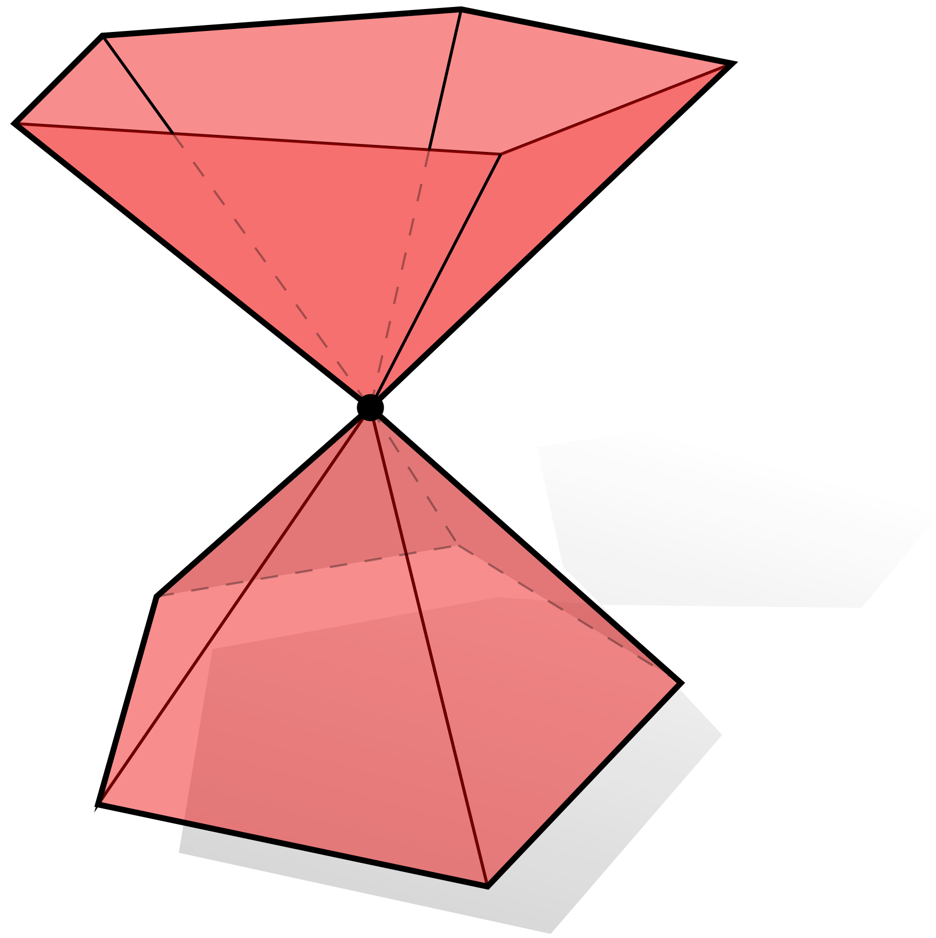
Key example: manifold triangle mesh (k=2)

- every edge is contained in exactly two triangles
  - …or just one along the boundary
- every vertex is contained in a single "loop" of triangles
  - …or a single "fan" along the boundary





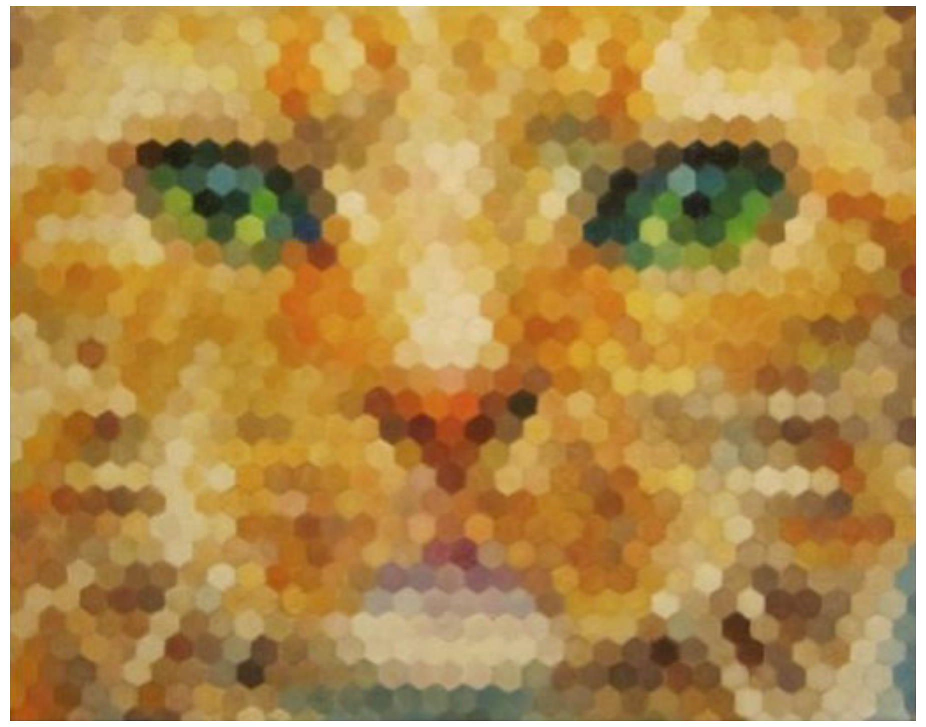
nonmanifold edge

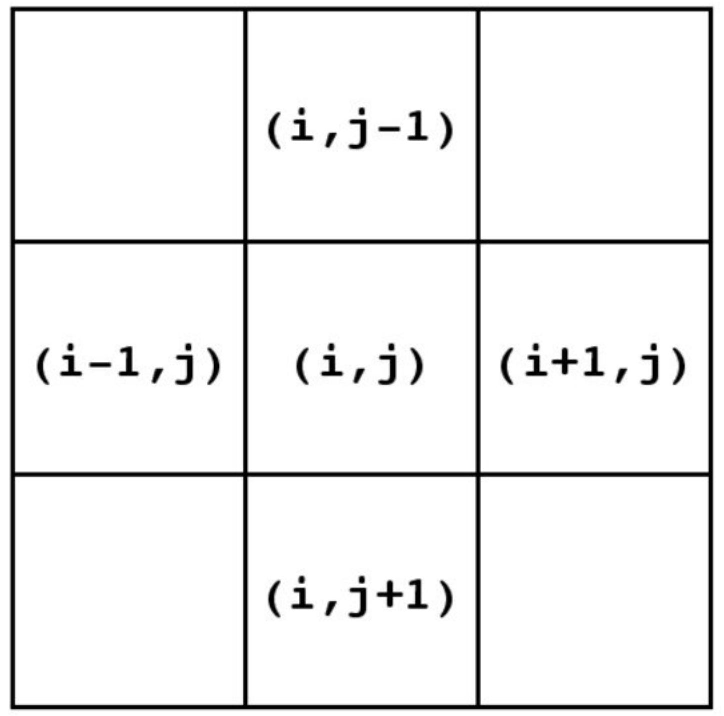


nonmanifold vertex

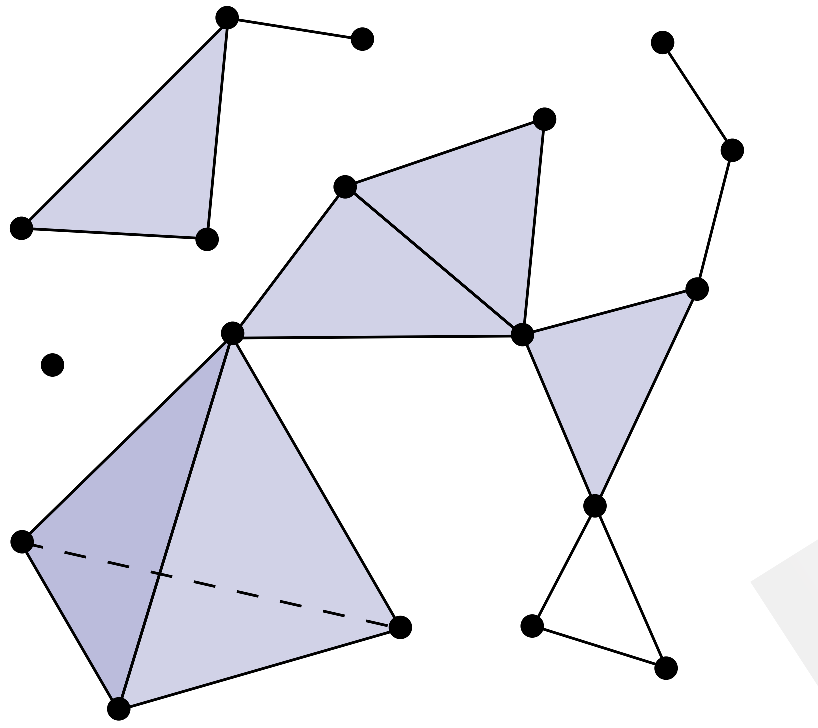
Manifold Meshes—Motivation

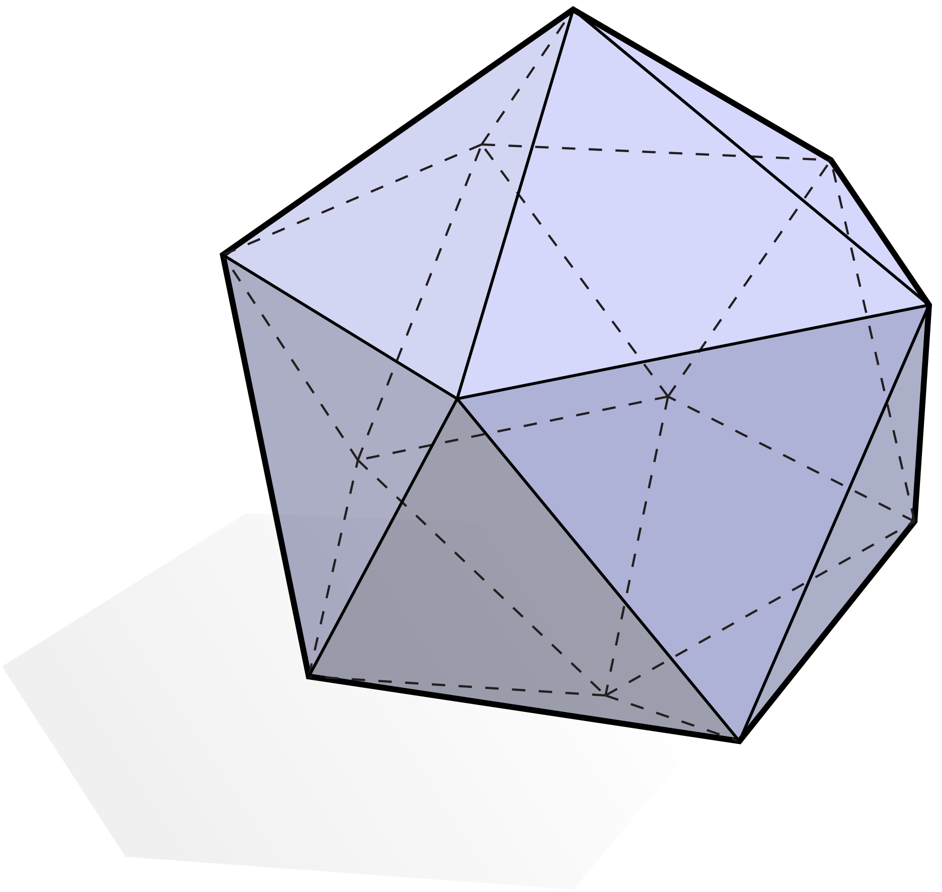
- Why might it be preferable to work with a *manifold* mesh?
- Analogy: 2D images
  - Lots of ways you *could* arrange pixels…
  - A regular grid does everything you need
  - Very simple (always have 4 neighbors)



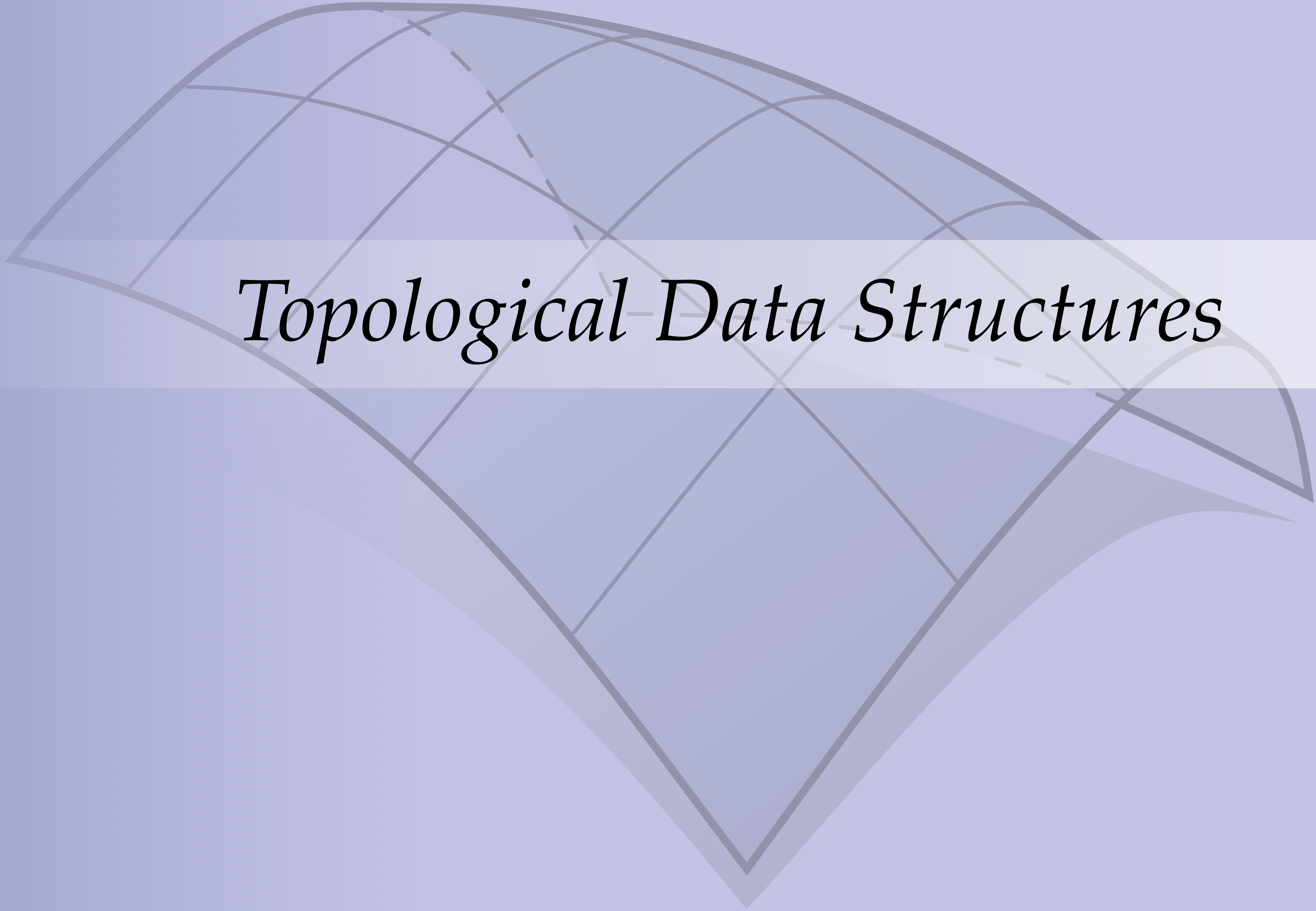


- Same deal with manifold meshes
  - *Could* allow arbitrary meshes…
  - Manifold mesh often does everything you need
  - Very simple (predictable neighborhoods)
  - *E.g.*, leads to nice **data structures**





Topological Data Structures

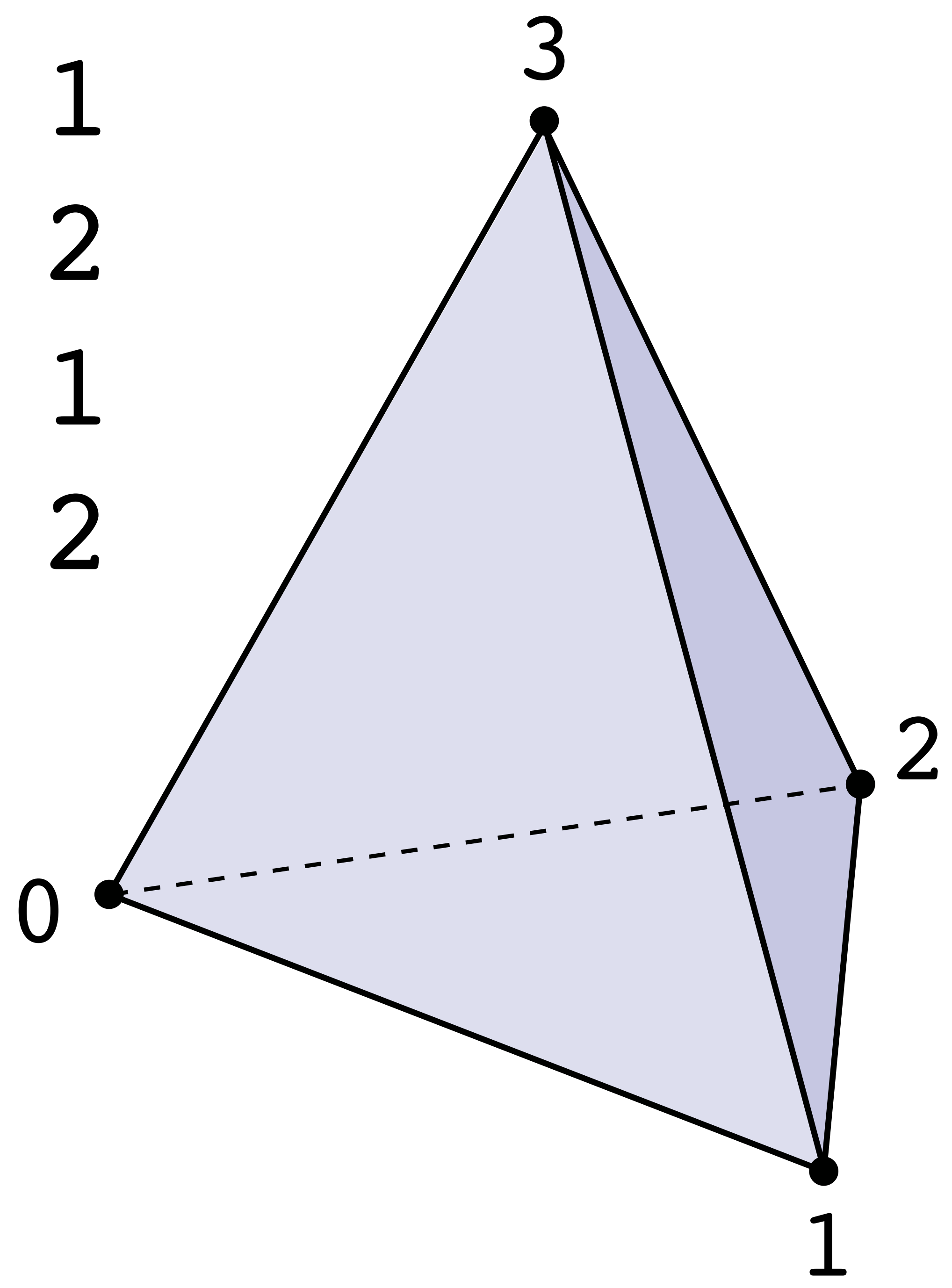


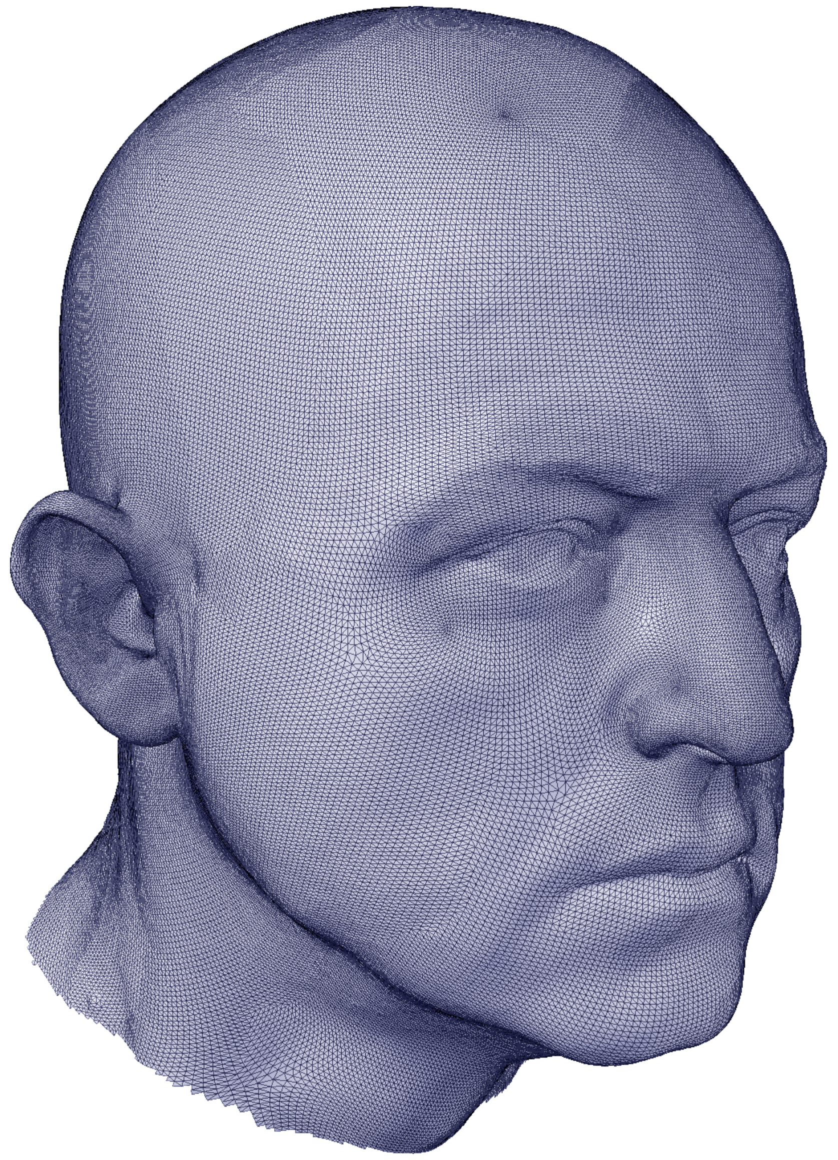
Topological Data Structures — Adjacency List

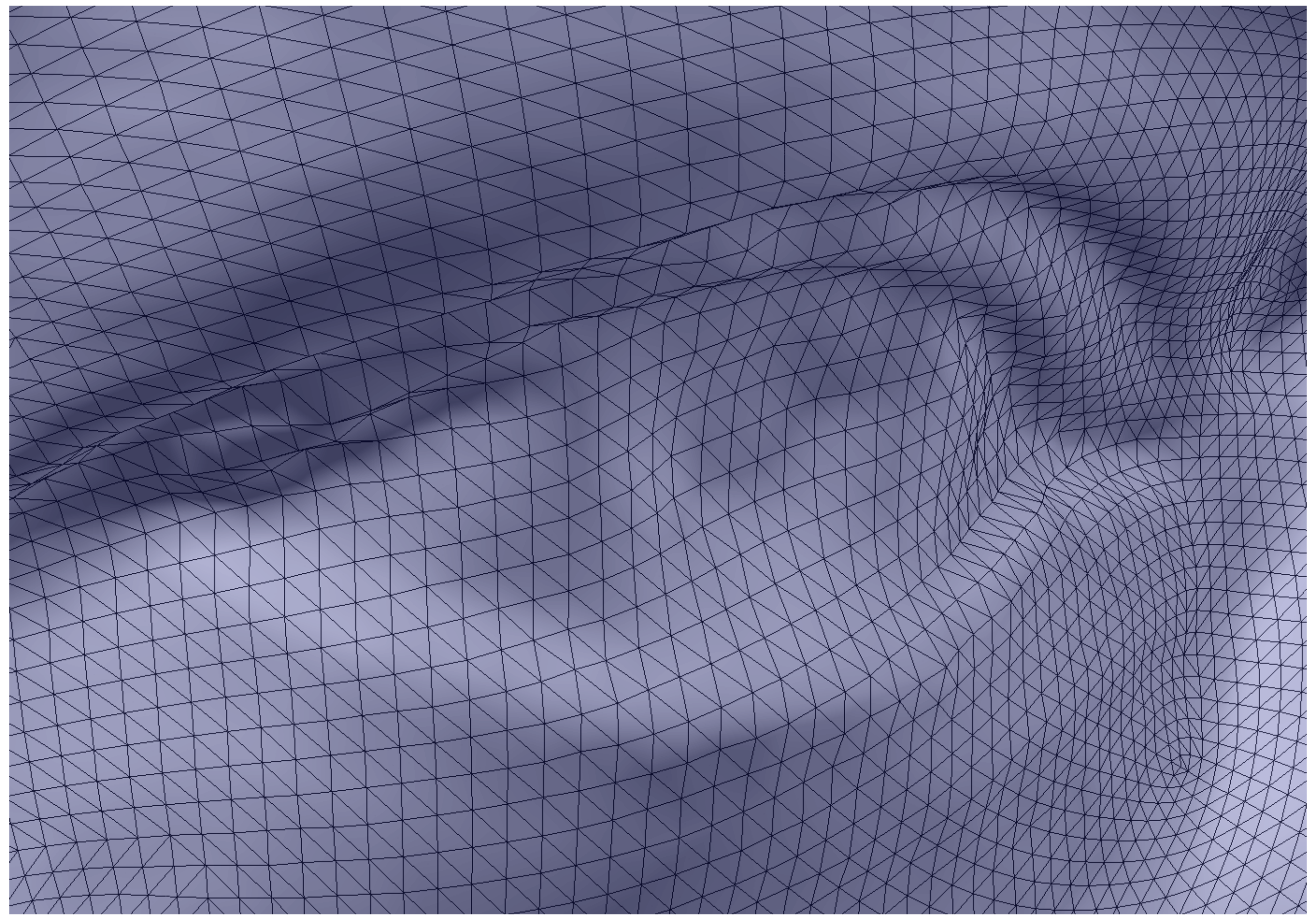
- Store only top-dimensional simplices
- Pros: simple, small storage cost
- Cons: hard to iterate over, *e.g.*, edges; expensive to access neighbors

Example. ("hollow" tetrahedron)

0	2	1
0	3	2
3	0	1
3	1	2



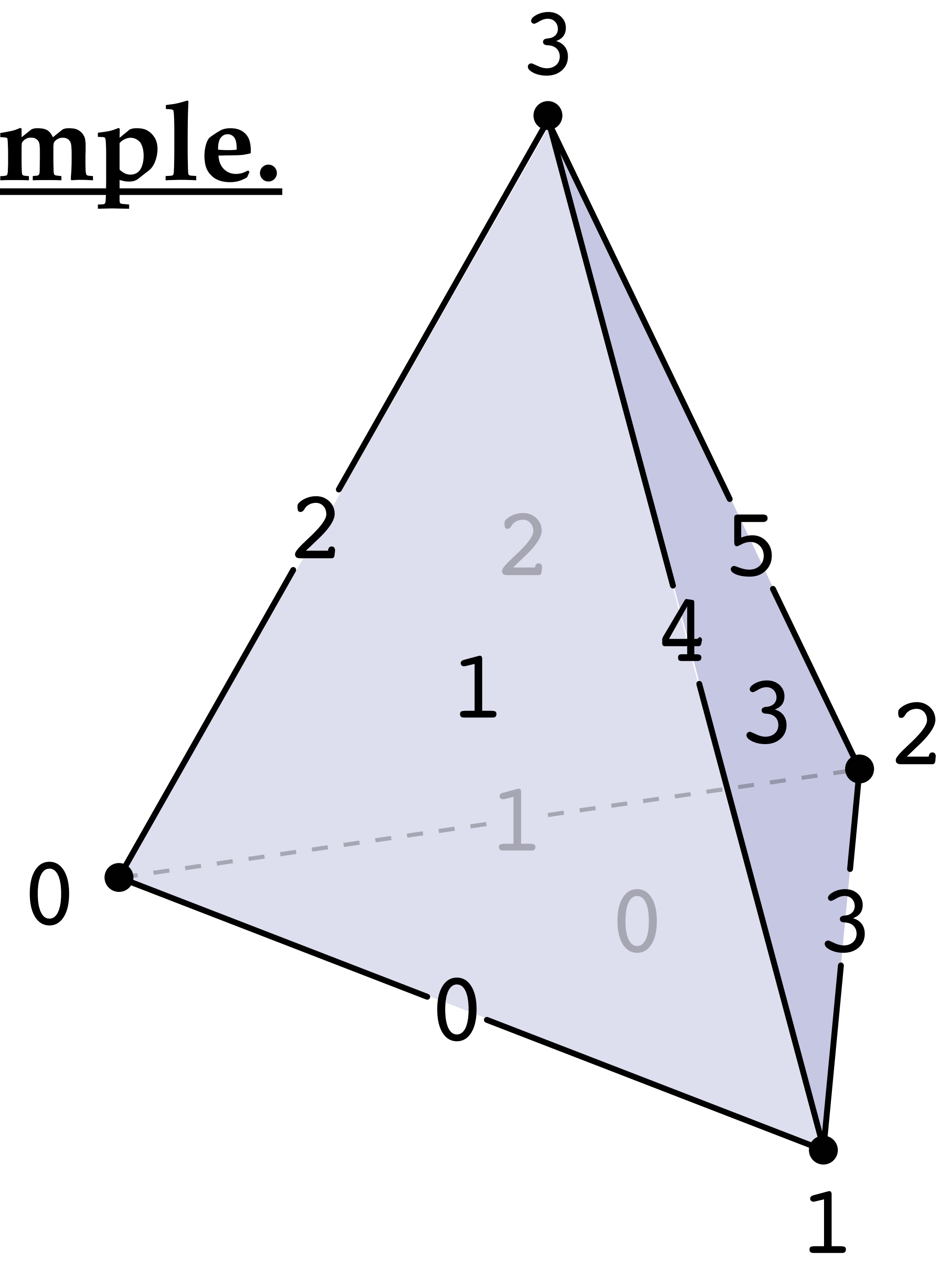




Q: How might you list all edges touching a given vertex? *What's the cost?*

Topological Data Structures—Incidence Matrix

Example.



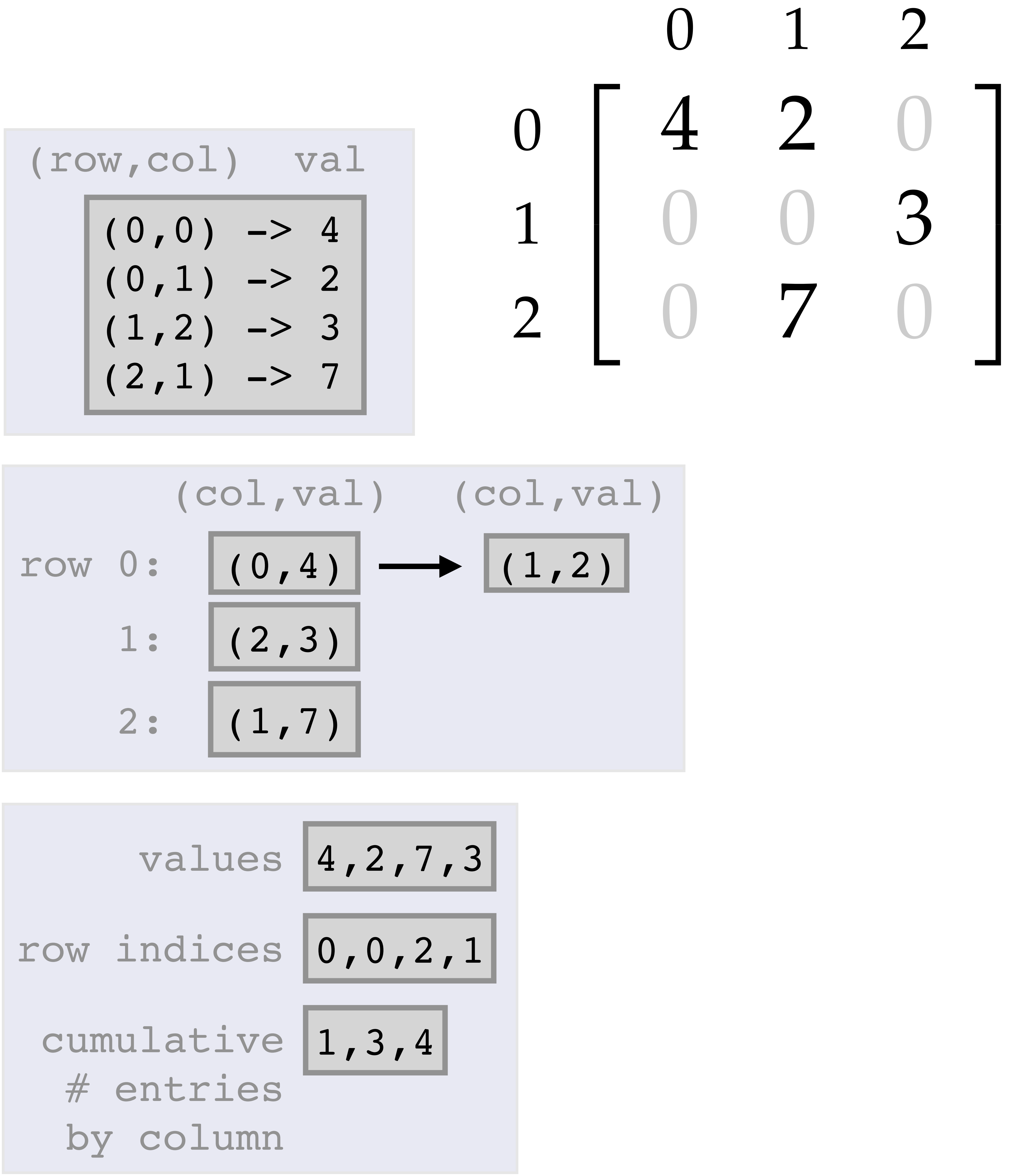
$$E^0 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

$$E^1 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

Definition. Let K be a simplicial complex, let n_k denote the number of k-simplices in K, and suppose that for each k we give the k-simplices a canonical ordering so that they can be specified via indices $1, \ldots, n_k$. The kth *incidence matrix* is then a $n_{k+1} \times n_k$ matrix E^k with entries $E^k_{ij} = 1$ if the jth k-simplex is contained in the ith $(k+1)$-simplex, and $E^k_{ij} = 0$ otherwise.

Aside: Sparse Matrix Data Structures

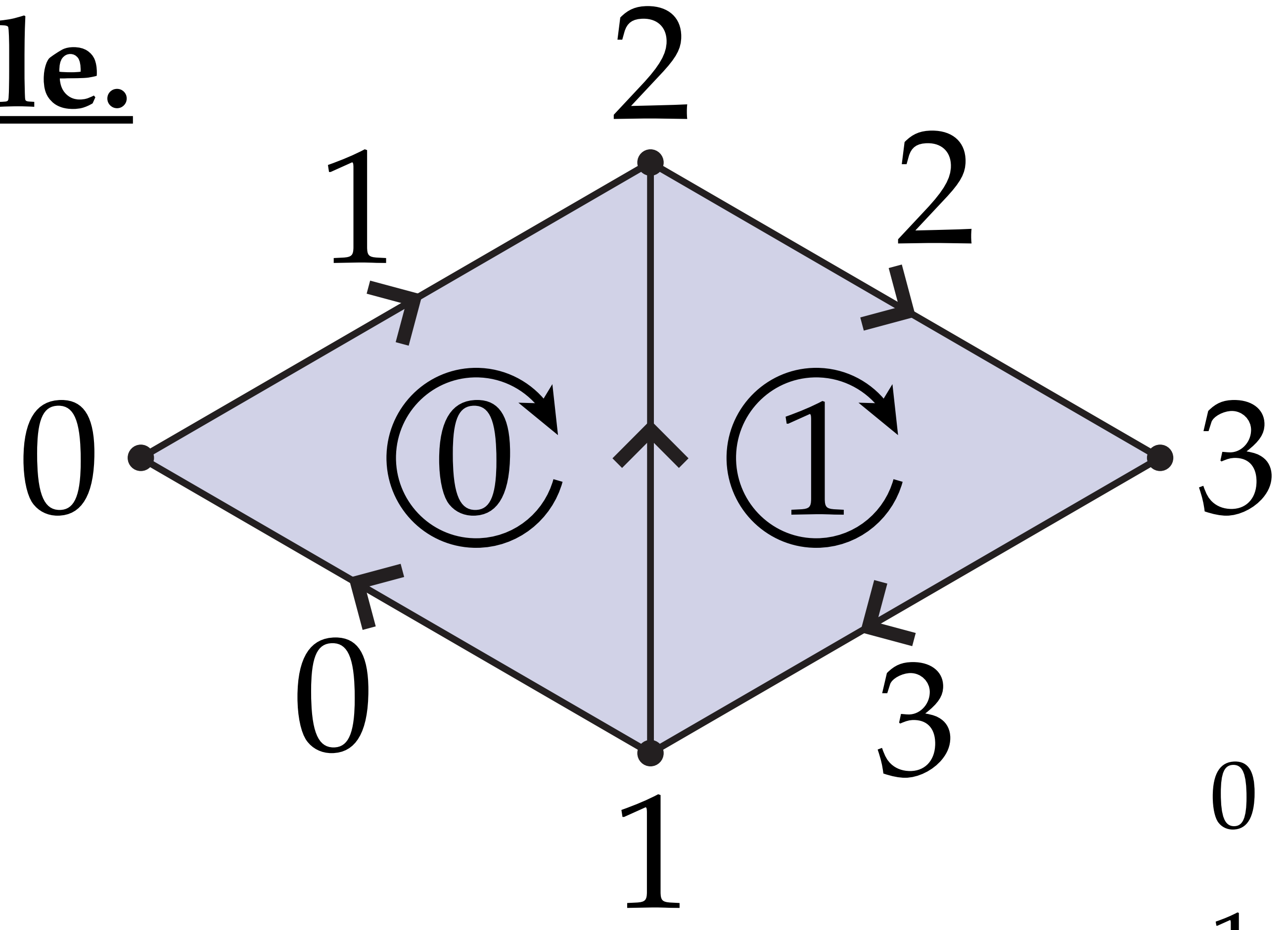
- **Enormous** waste to explicitly store zeros (O(n) vs. O(n^2))
- Instead use a *sparse matrix* data structure
- **<u>Associative array</u>** from (row, col) to value
  - easy to lookup/set entries (e.g., hash table)
  - harder to do matrix operations (e.g., multiply)
- **<u>Array of linked lists</u>**
  - conceptually simple
  - slow access time; incoherent memory access
- **<u>Compressed column format</u>**
  - hard to add/remove entries
  - fast for actual matrix operations (e.g., multiply)
- In practice: build "raw" list of entries first, then convert to final (e.g., compressed) data structure



Data Structures—Signed Incidence Matrix

A *signed incidence matrix* is an incidence matrix where the sign of each nonzero entry is determined by the relative orientation of the two simplices corresponding to that row/column.

Example.



$$E^0 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & -1 & 1 & 0 \end{bmatrix} \end{array}$$

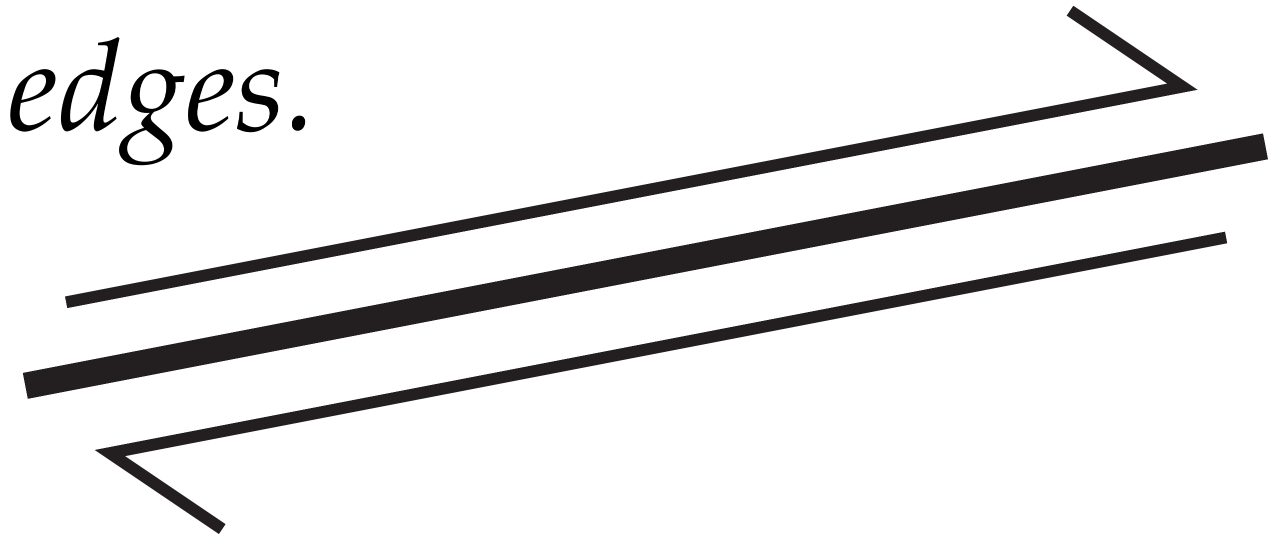
$$E^1 = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

(Closely related to *discrete exterior calculus*.)

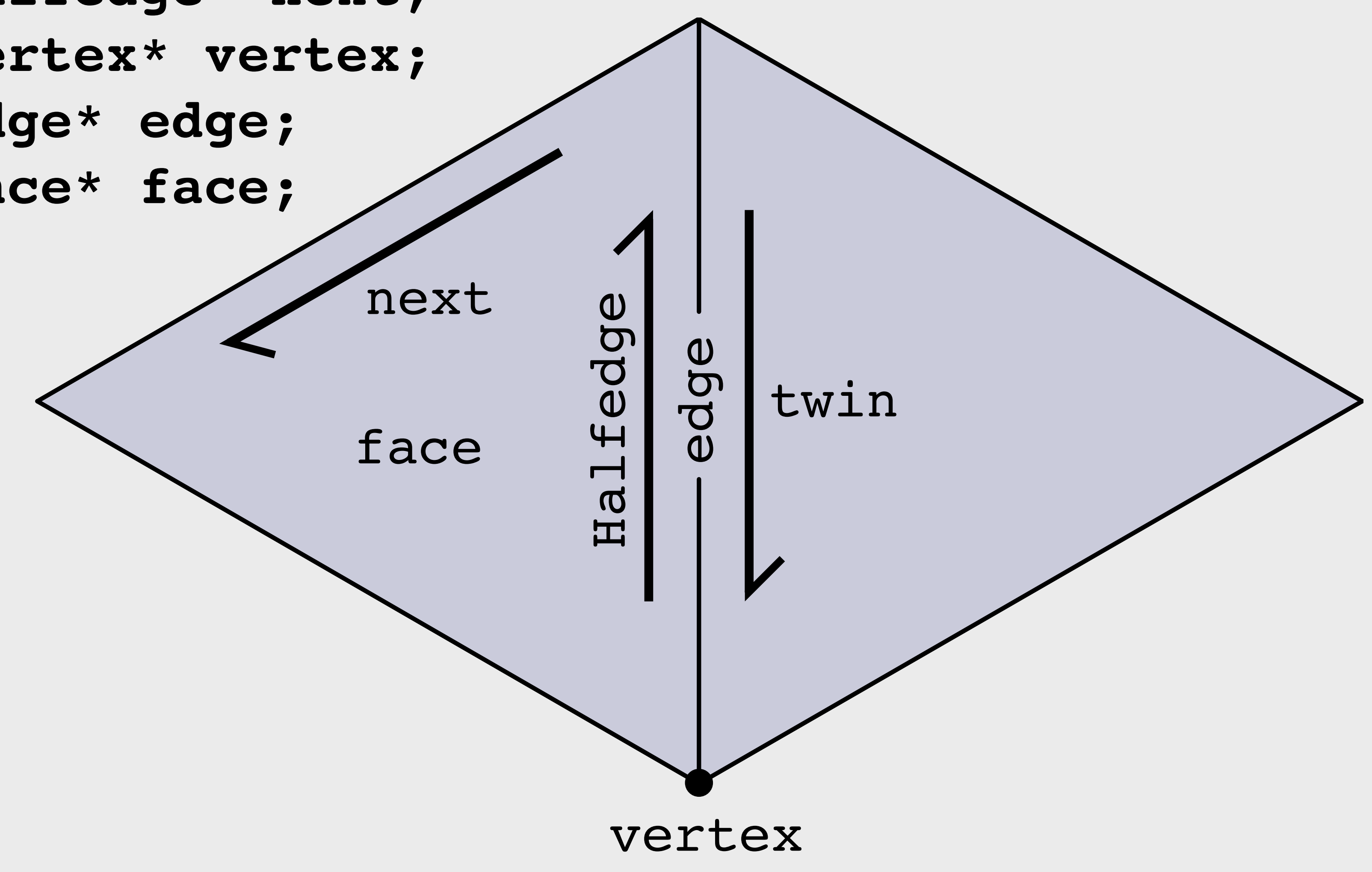
Topological Data Structures—Half Edge Mesh

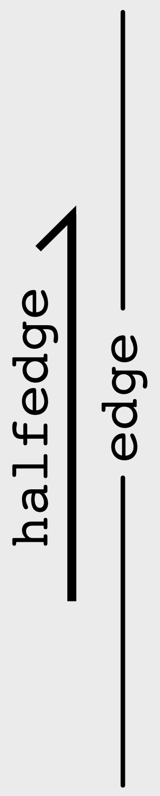
Basic idea: each edge gets split into two oppositely-oriented *half edges.*

- Half edges act as "glue" between mesh elements.
- All other elements know only about a single half edge.



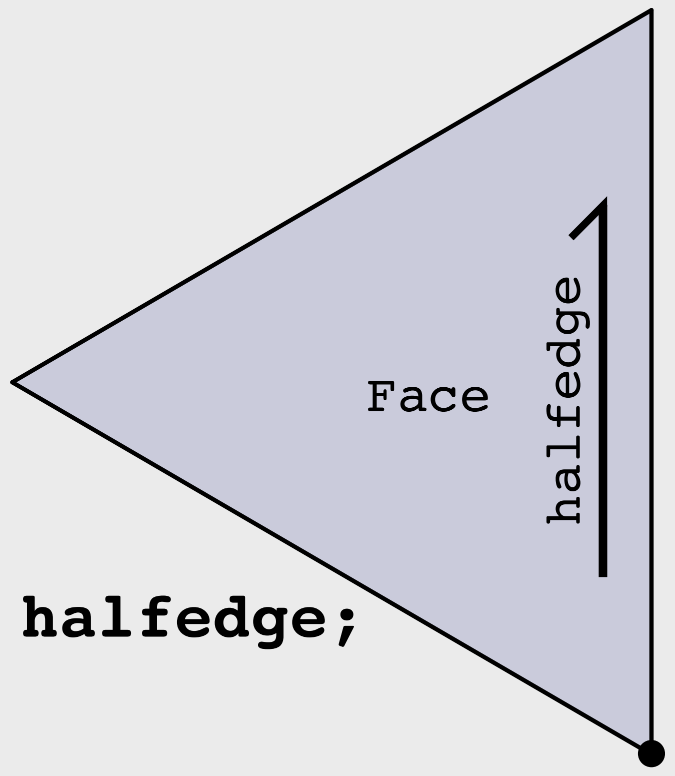
```
struct Halfedge
{
   Halfedge* twin;
   Halfedge* next;
   Vertex* vertex;
   Edge* edge;
   Face* face;
};
```

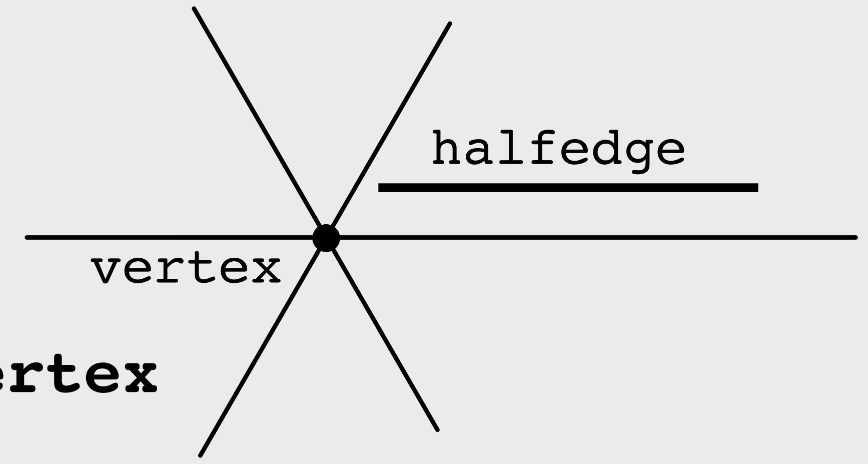





```
struct Edge
{
   Halfedge* halfedge;
};
```

```
struct Face
{
   Halfedge* halfedge;
};
```






```
struct Vertex
{
   Halfedge* halfedge;
};
```

(You will use a half edge data structure in your assignments!)

Half Edge—Algebraic Definition

Definition. Let H be any set with an even number of elements, let $\rho : H \to H$ be any permutation of H, and let $\eta : H \to H$ be an involution without any fixed points, *i.e.*, $\eta \circ \eta = \text{id}$ and $\eta(h) \neq h$ for any $h \in H$. Then (H, ρ, η) is a *half edge mesh*, the elements of H are called *half edges*, the orbits of η are *edges*, the orbits of ρ are *faces*, and the orbits of $\eta \circ \rho$ are *vertices*.

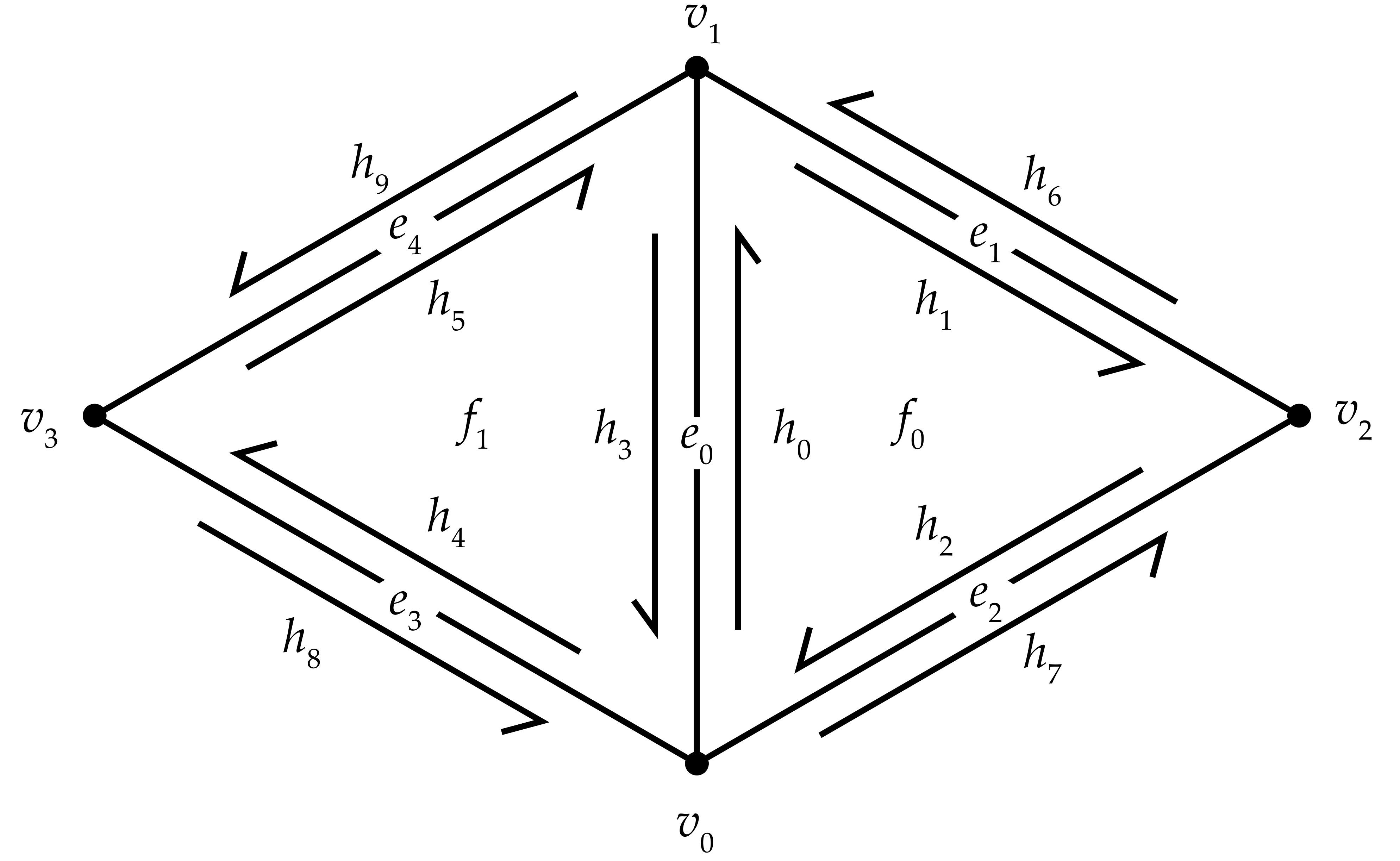
Fact. Every half edge mesh describes a compact oriented topological surface (without boundary).

$$(h_0, \ldots, h_9) \overset{\rho}{\mapsto} (h_1, h_2, h_0, h_4, h_5, h_3, h_9, h_6, h_7, h_8)$$

"next"

$$(h_0, \ldots, h_9) \overset{\eta}{\mapsto} (h_3, h_6, h_7, h_0, h_8, h_9, h_1, h_2, h_4, h_5)$$

"twin"



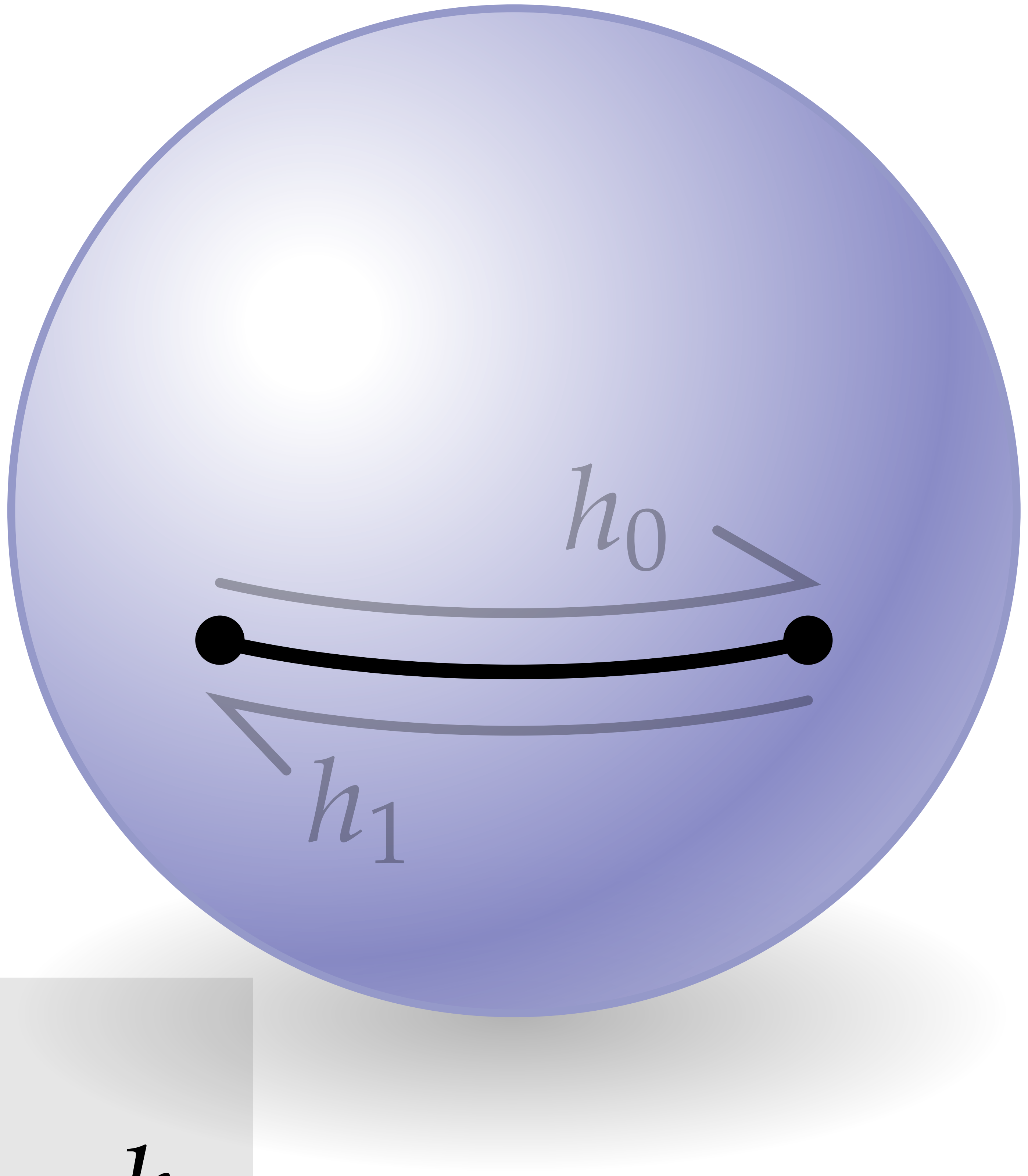
Half Edge—Smallest Example

Example. Consider just two half edges h_0, h_1

twin

$\eta(h_0) = h_1$

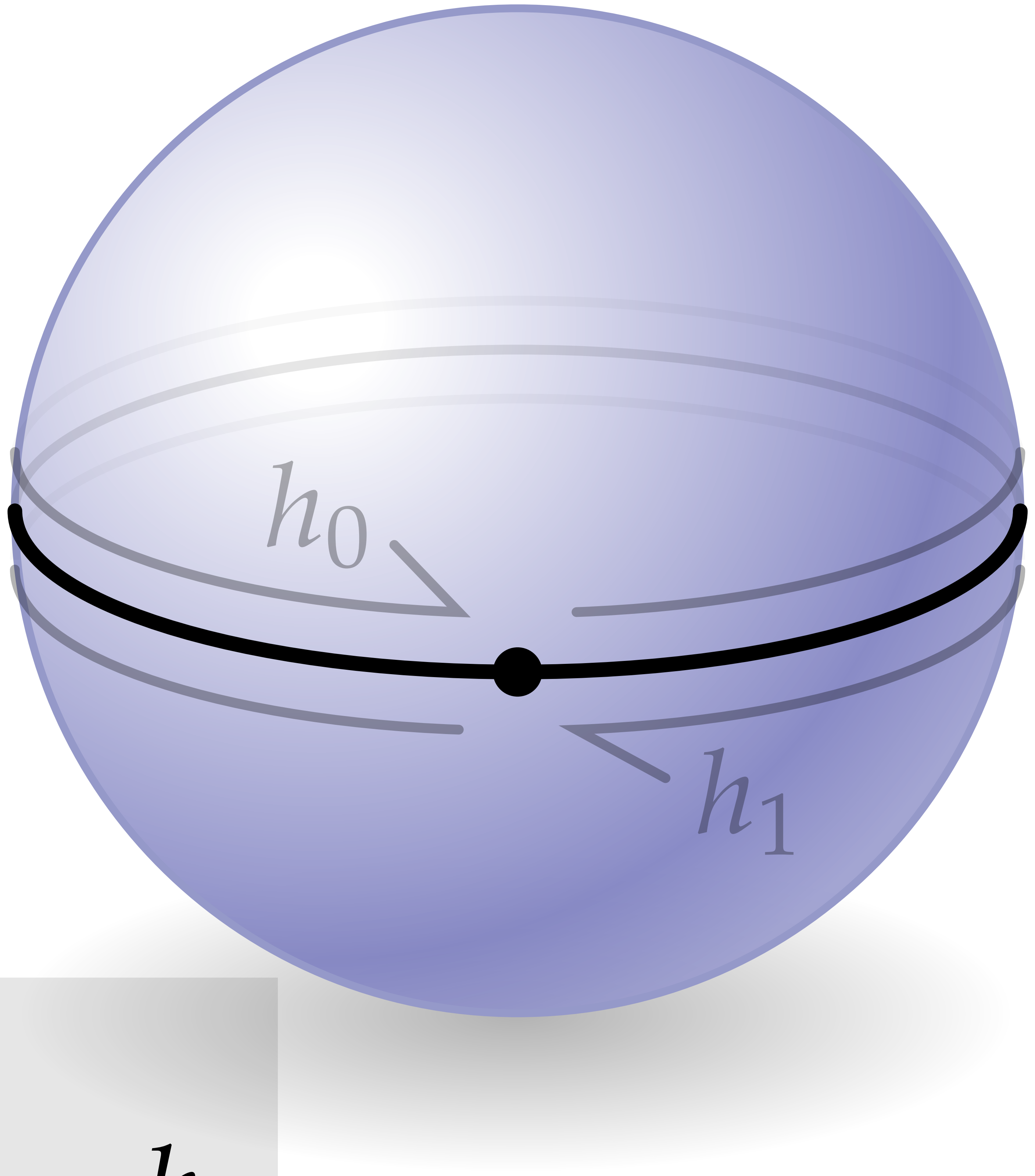
$\eta(h_1) = h_0$



next

$\rho(h_0) = h_1$

$\rho(h_1) = h_0$

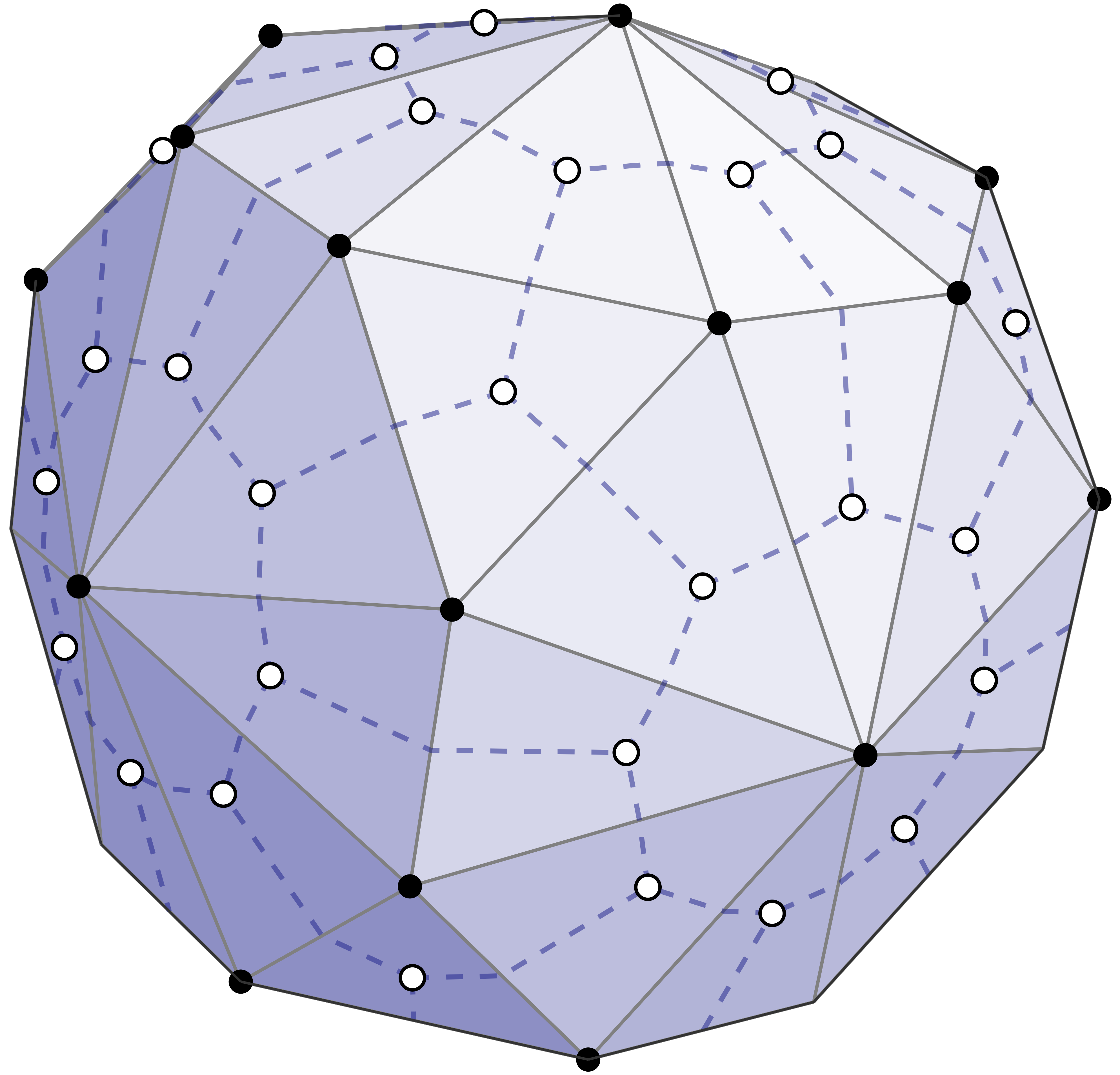


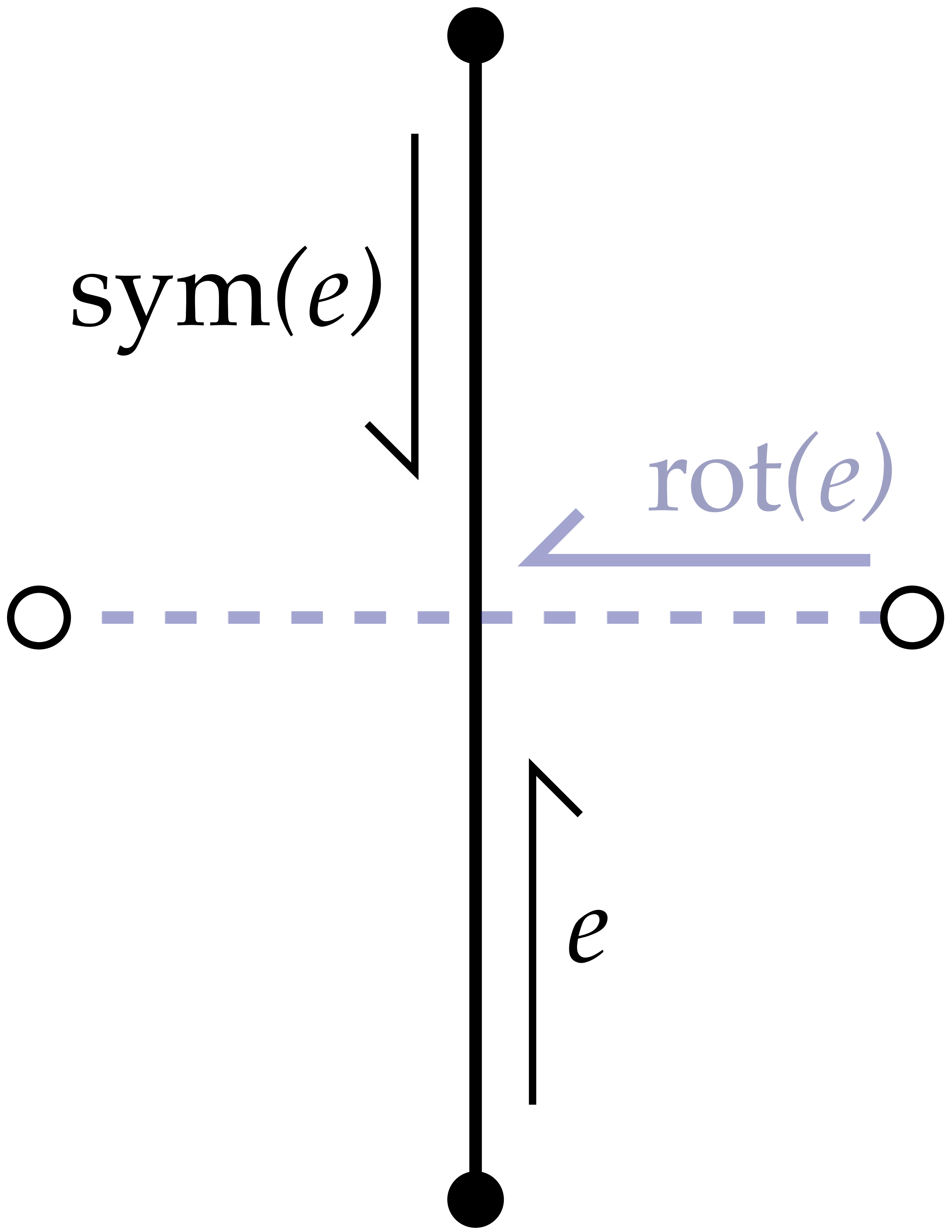
next

$\rho(h_0) = h_0$

$\rho(h_1) = h_1$

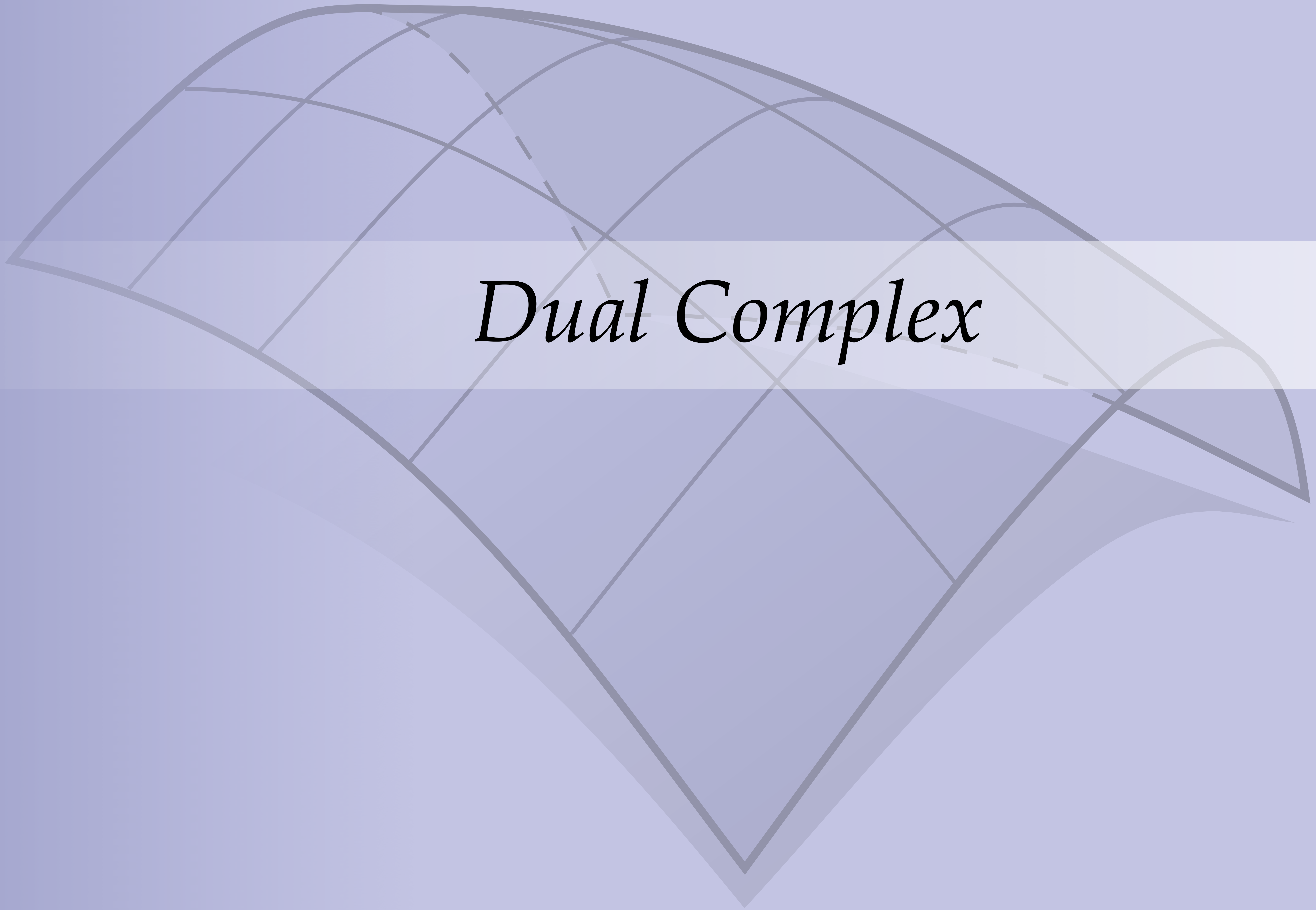
Other Data Structures—Quad Edge



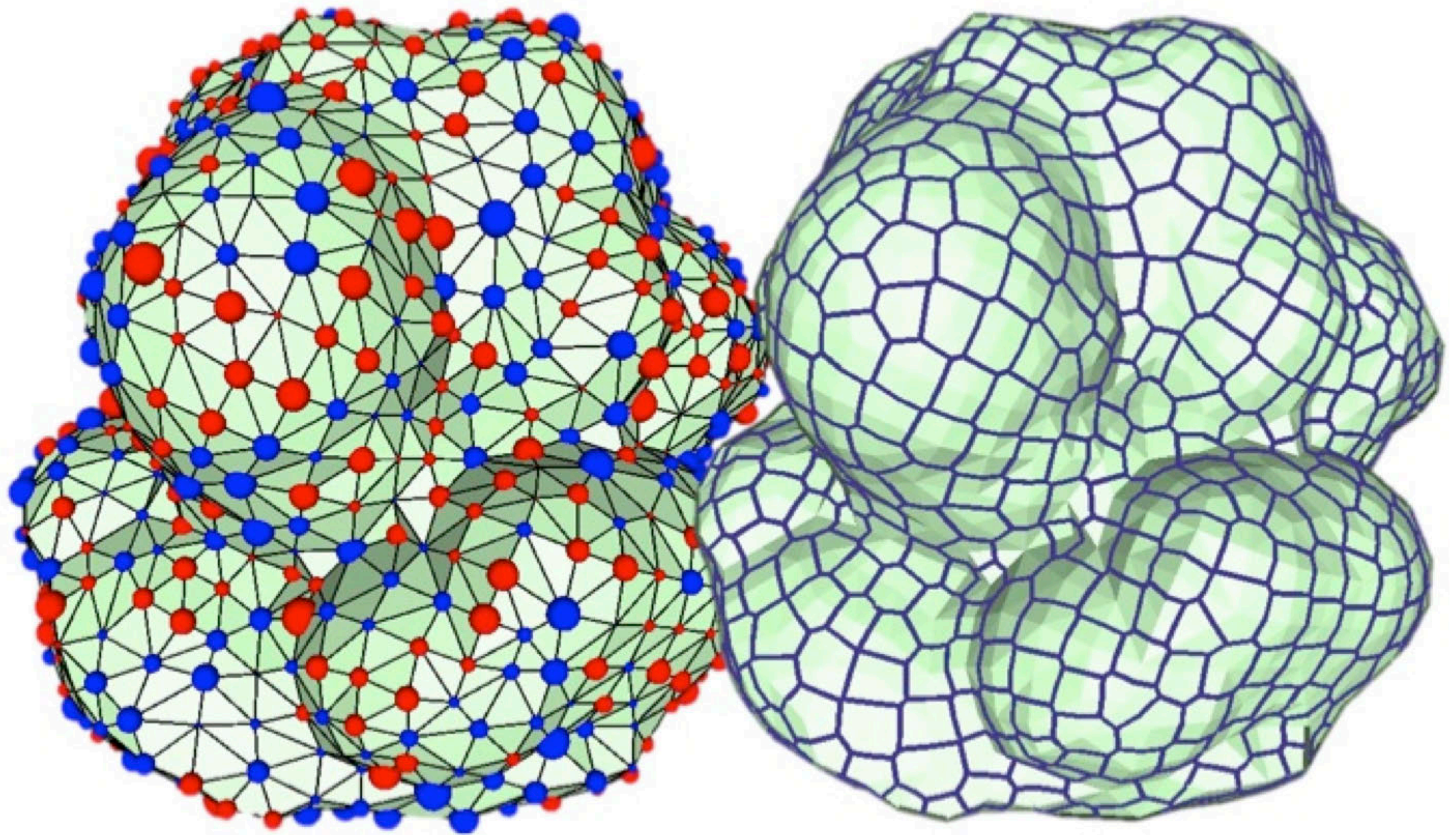


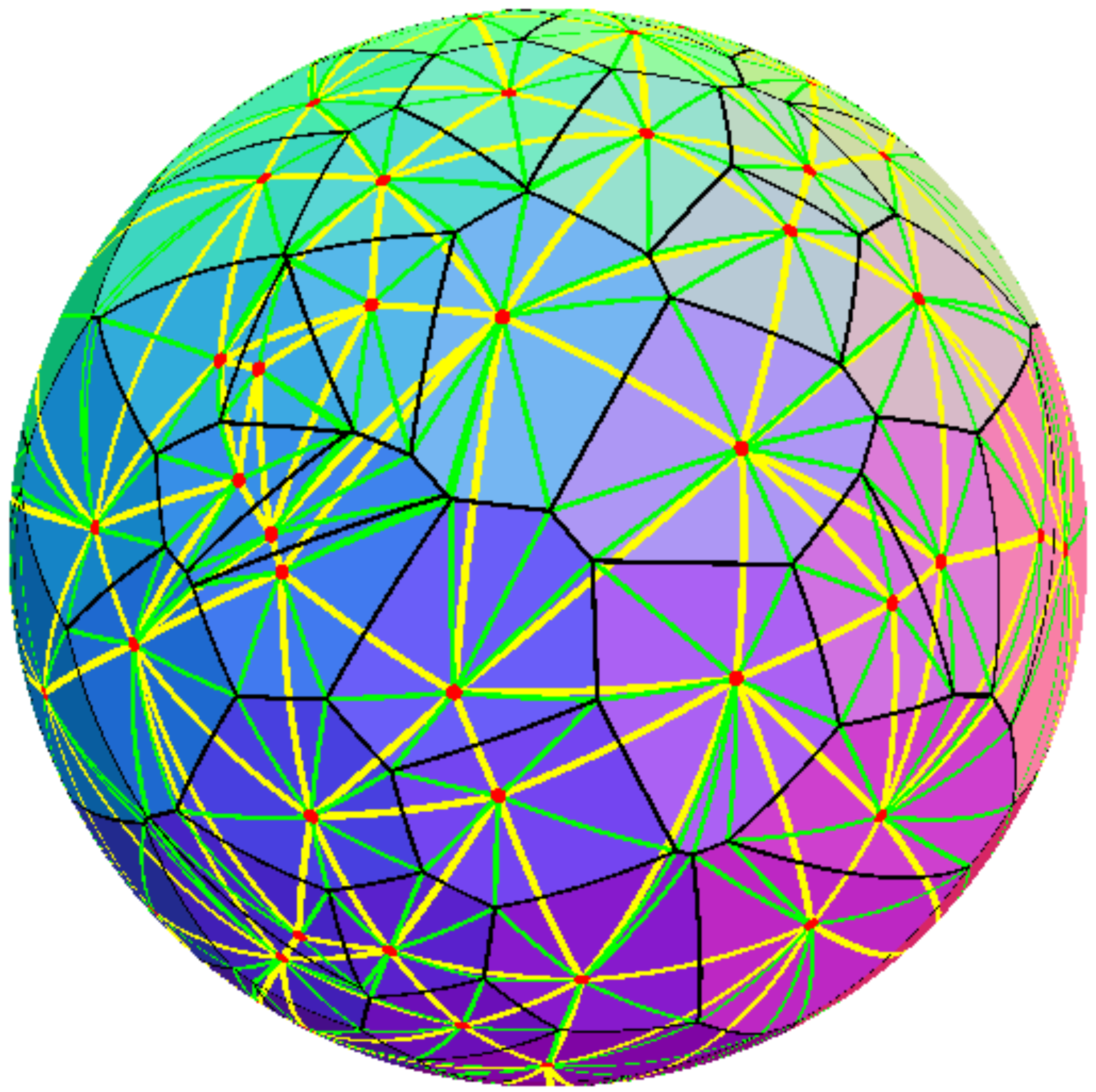
(L. Guibas & J. Stolfi, "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams")

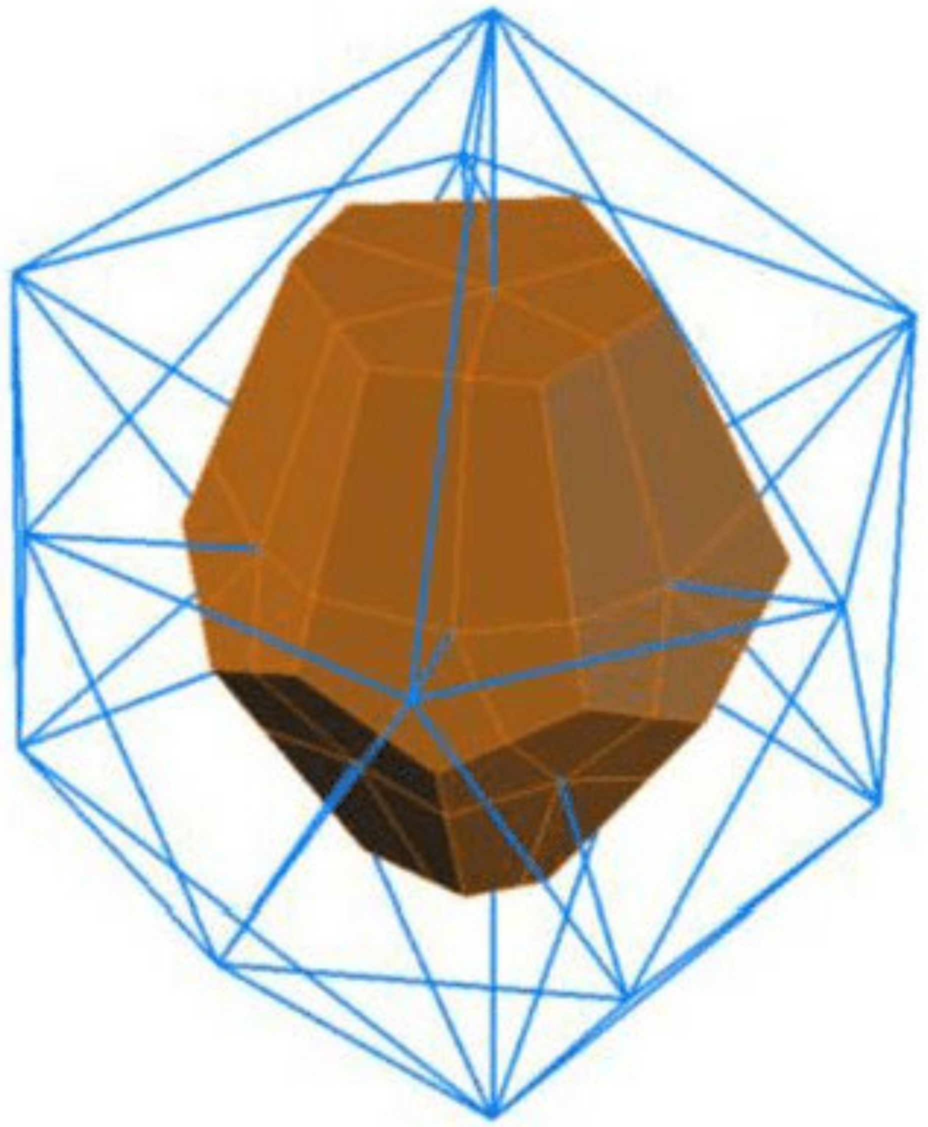
Dual Complex

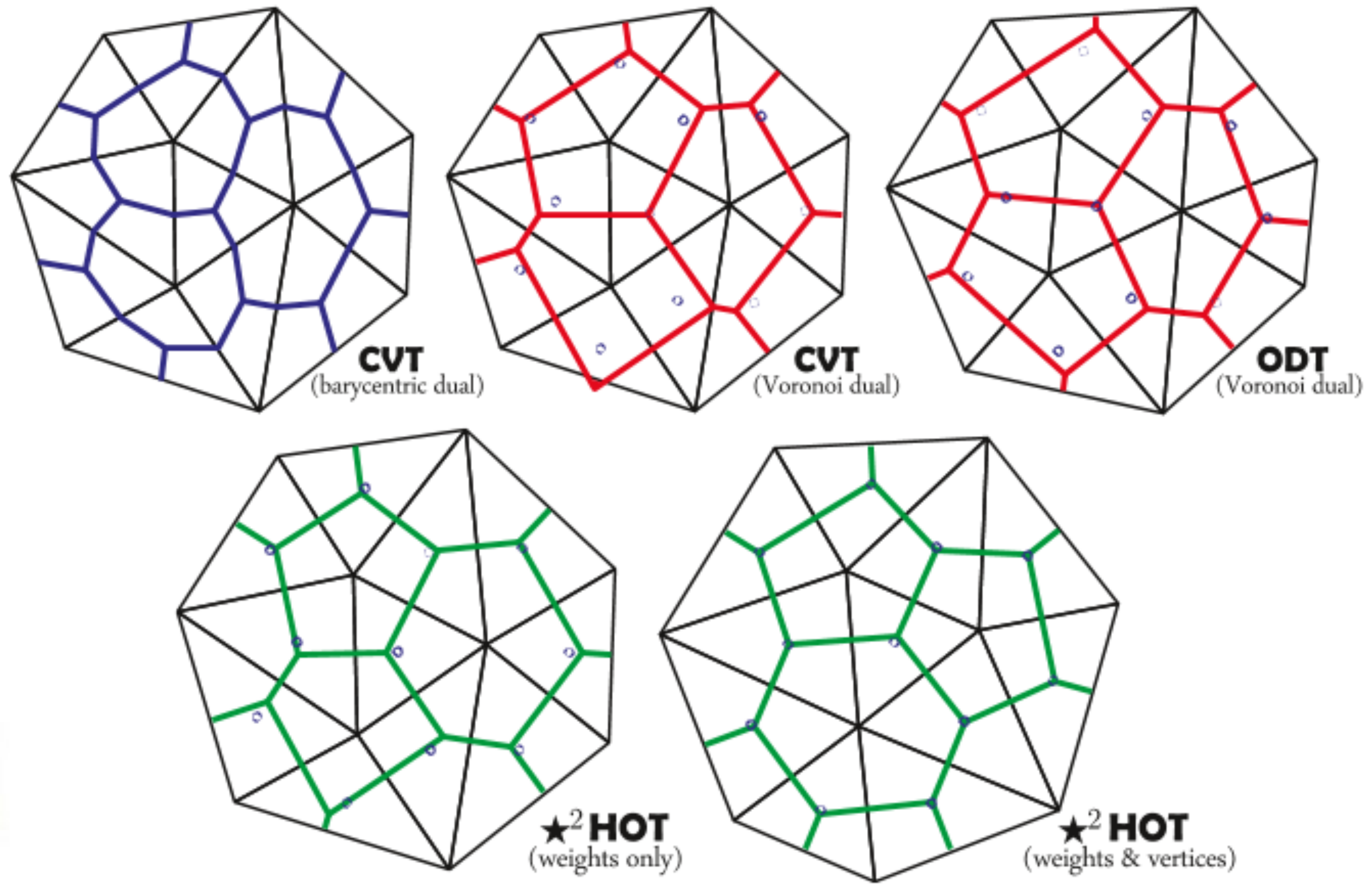


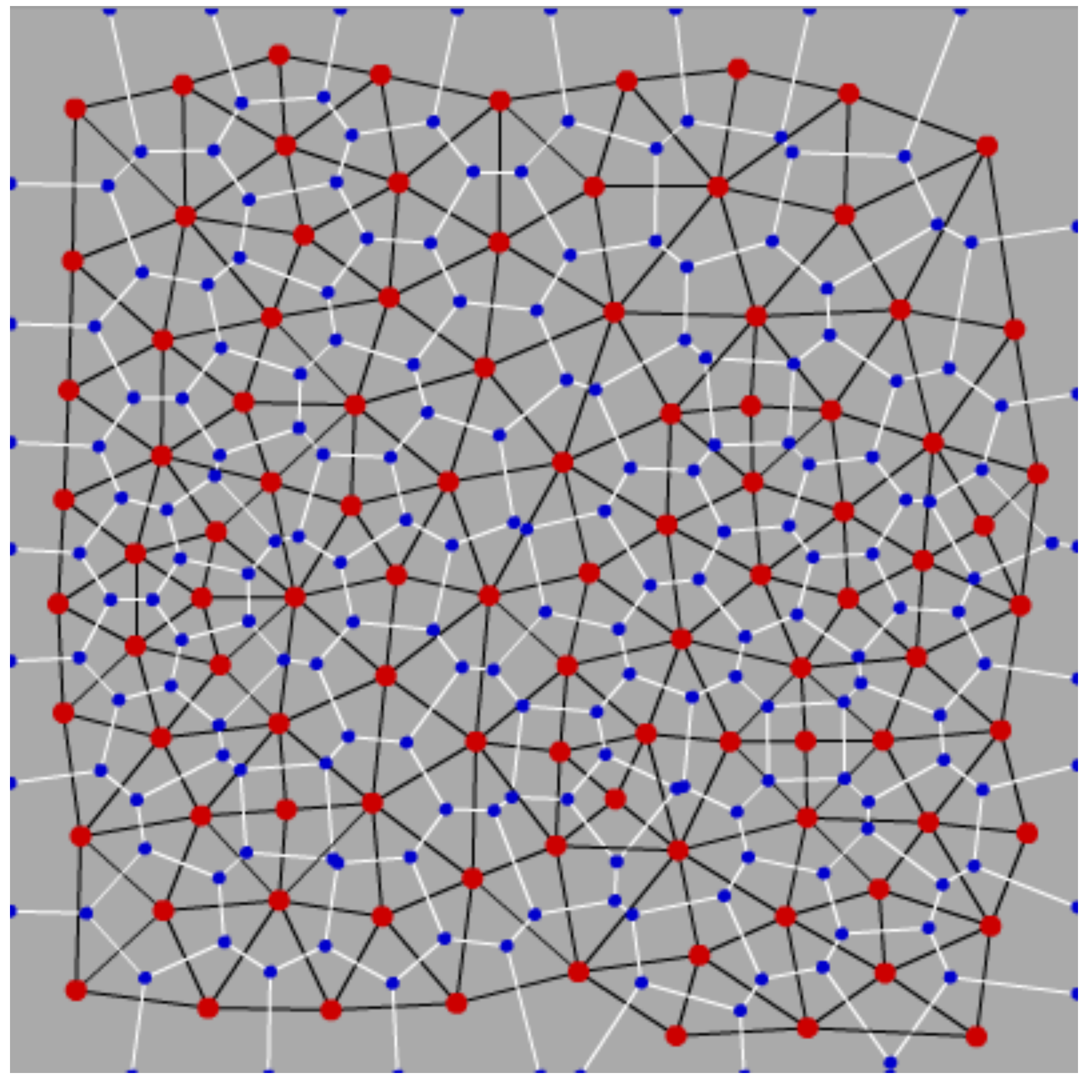
Dual Mesh — Visualized



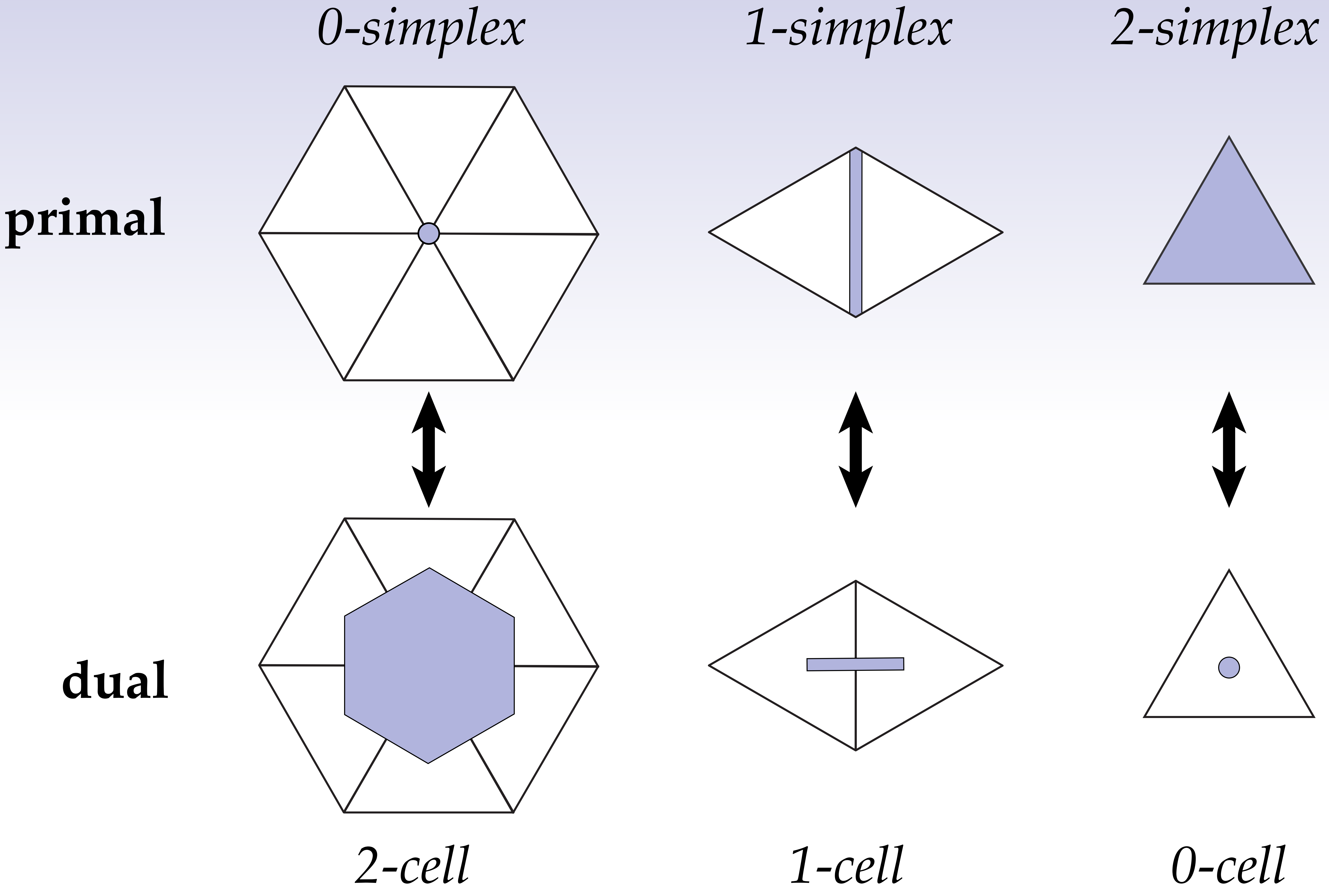






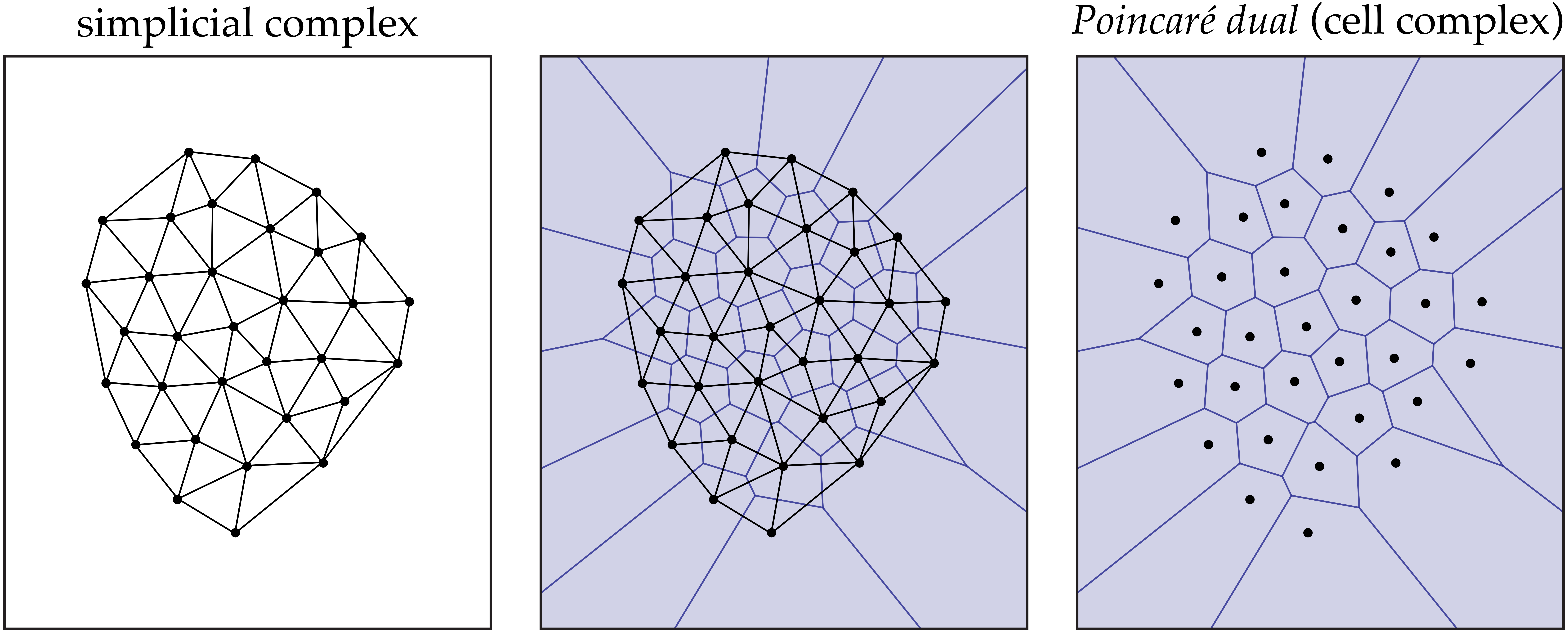


Primal vs. Dual



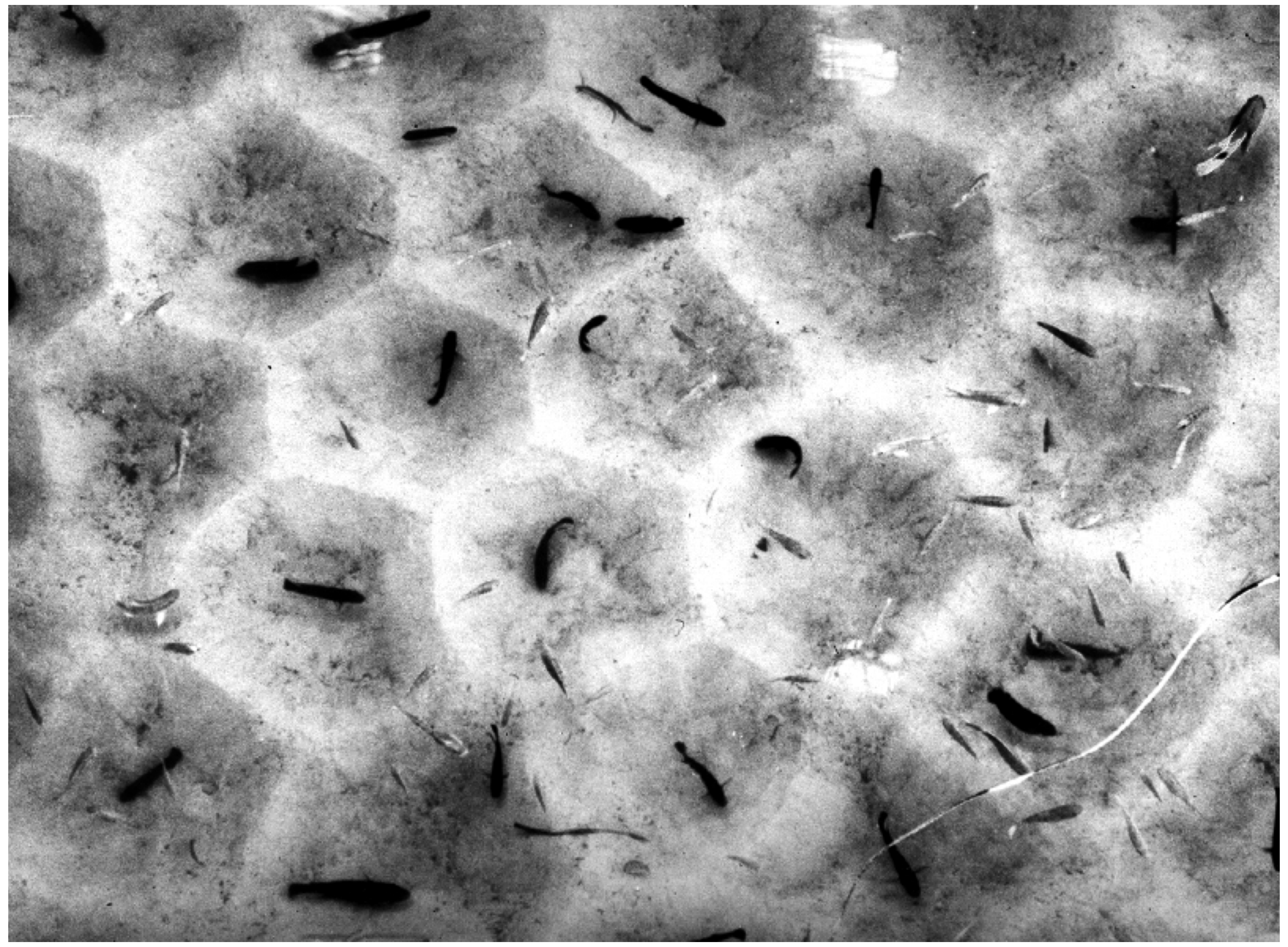
Motivation: record measurements of flux *through* vs. circulation *along* elements.

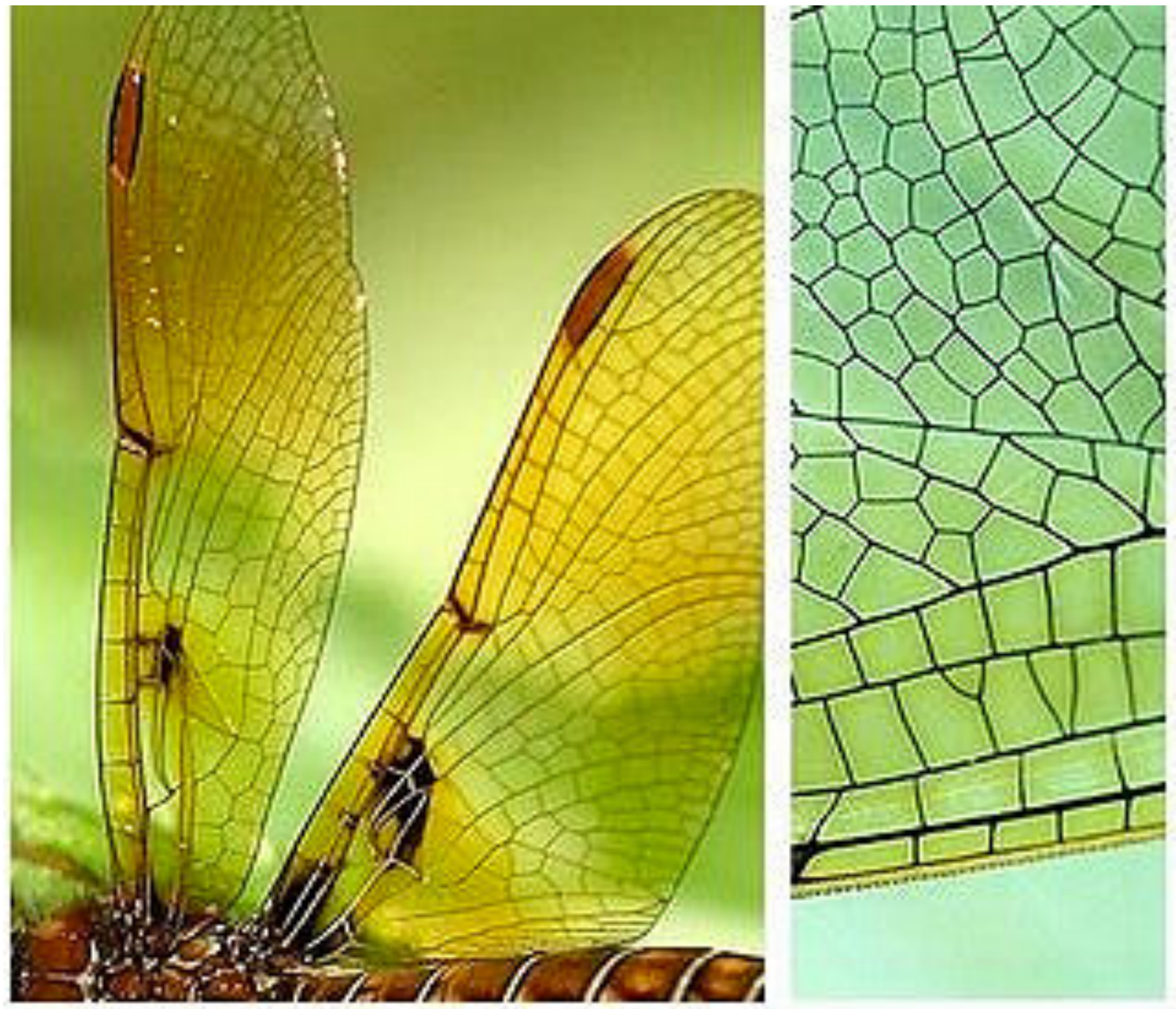
Poincaré Duality

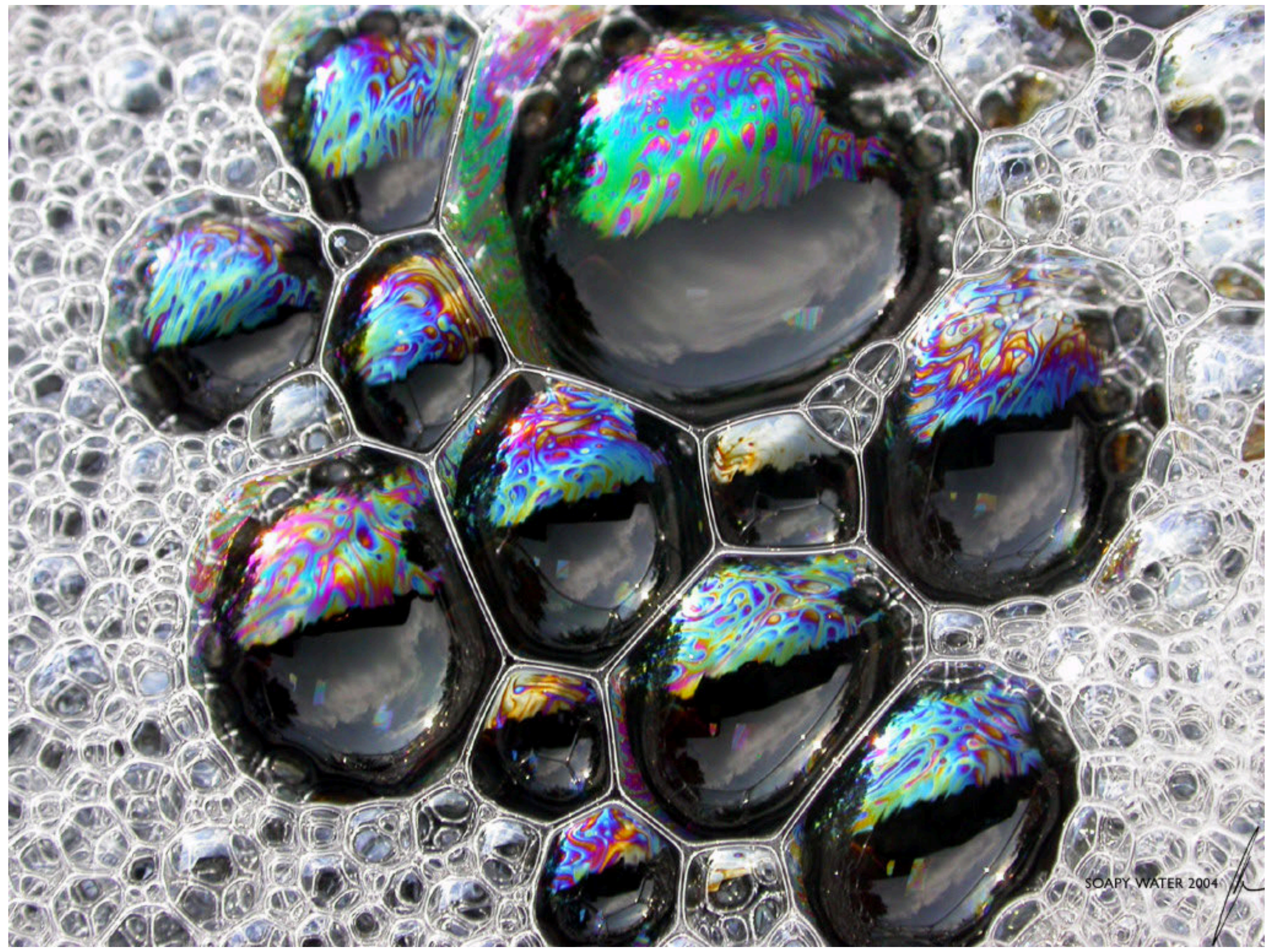


Note: we have said nothing (so far) about *where* the dual vertices are—only *connectivity*.

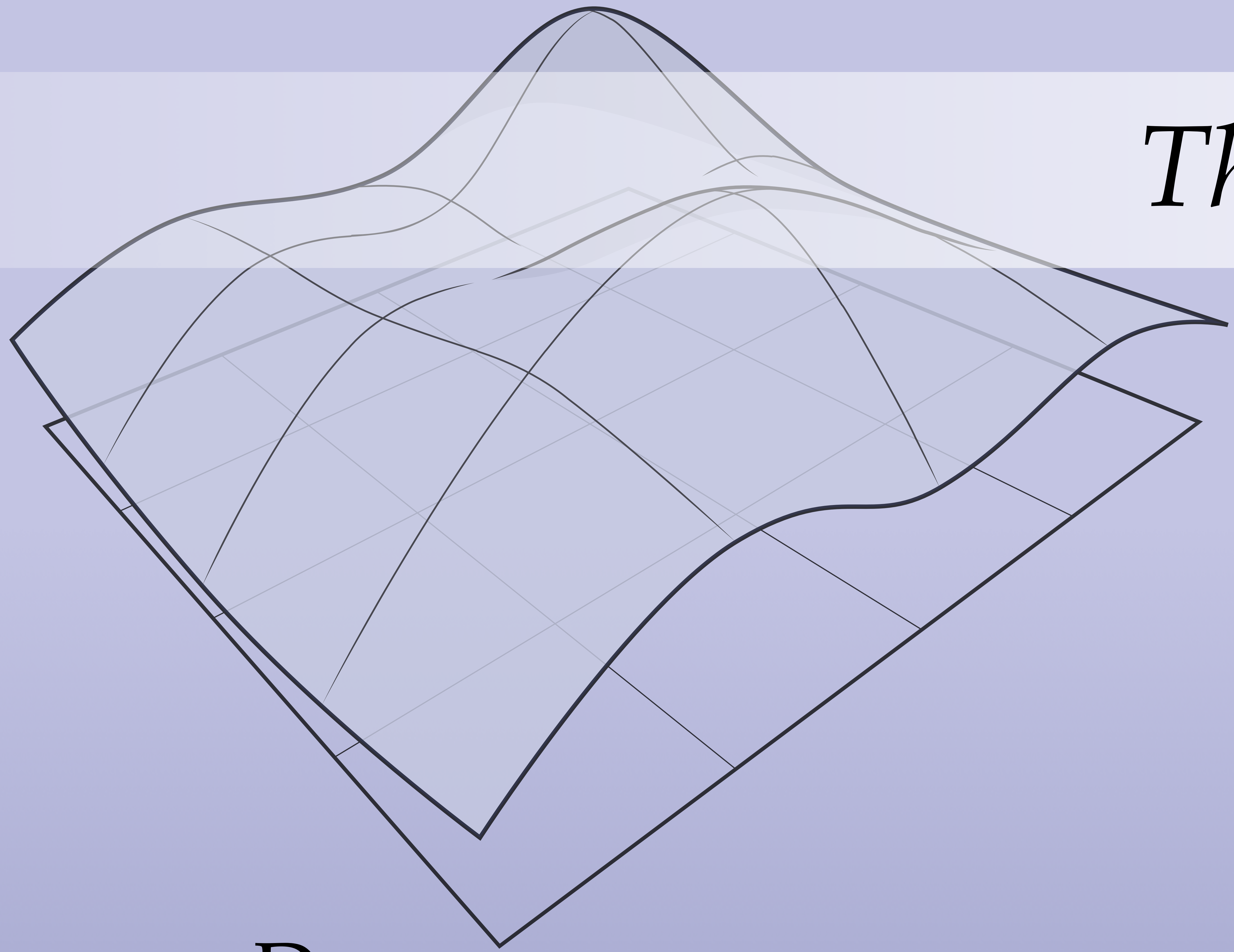
Poincaré Duality in Nature







Thanks!



Discrete Differential Geometry: An Applied Introduction

Keenan Crane • CMU 15-458/858