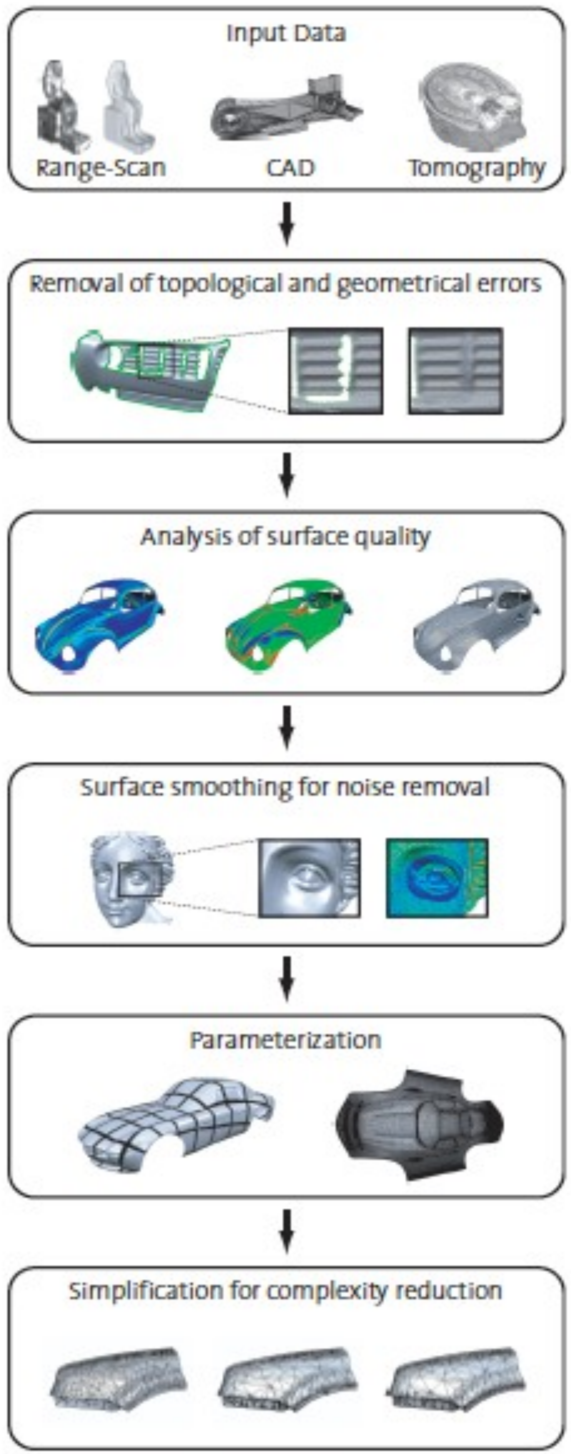
Geometric Modeling

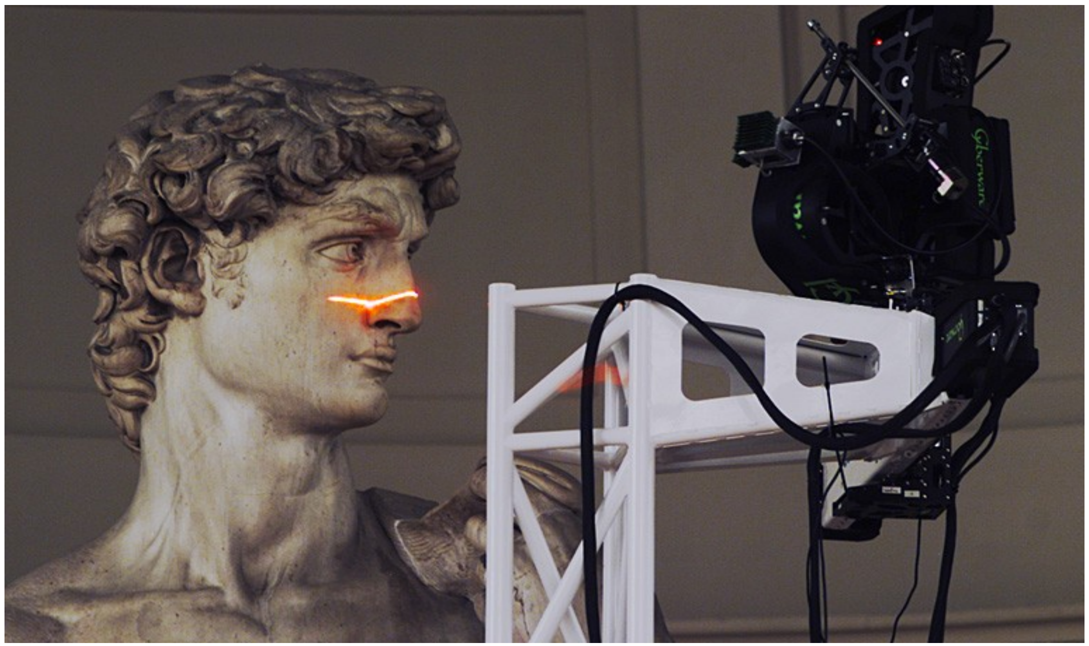
Sources of 3D data

- Modeling programs
  - Autodesk
  - CATIA
  - Bryce
  - Pro/E
- 3D Scanning
  - LiDAR/Range Scanning
- Photogrammetry
  - Structure from motion
- Procedural methods
  - Program creates model algorithmically
- Typing it in….

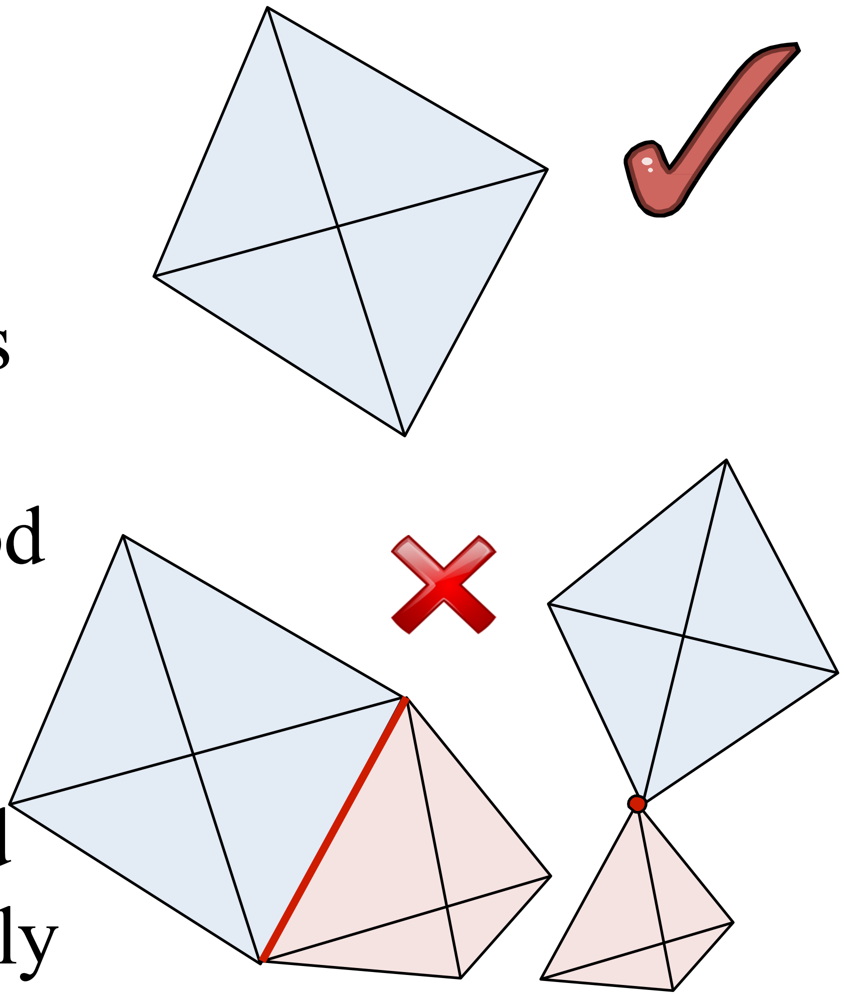


Digital Michelangelo

In 1998 Marc Levoy of Stanfor
  scanned several of Michelangelo sculptures
David at 1mm resolution
St. Matthew at 290 μm resolution
  about 200M triangles
What does "scanning resolution" mean?



Good Meshes



Manifold: 1.Every edge connects exactly two faces
2. Vertex neighborhood is "disk-like"

Orientable: Consistent normals

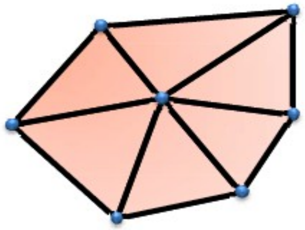
Watertight: Orientable + Manifold

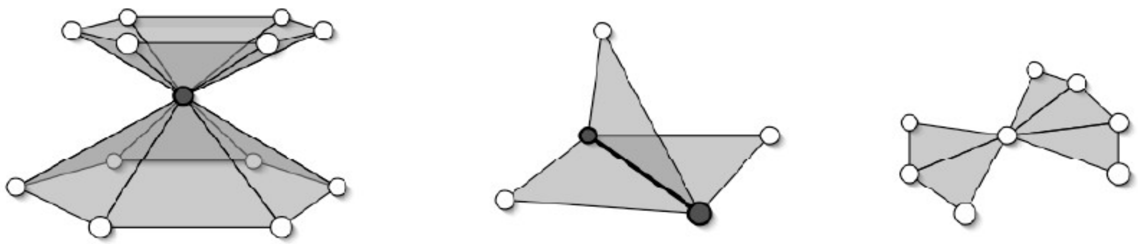
Boundary: Some edges bound only one face

Ordering: Vertices in CCW order when viewed from normal

2-Manifold Meshes

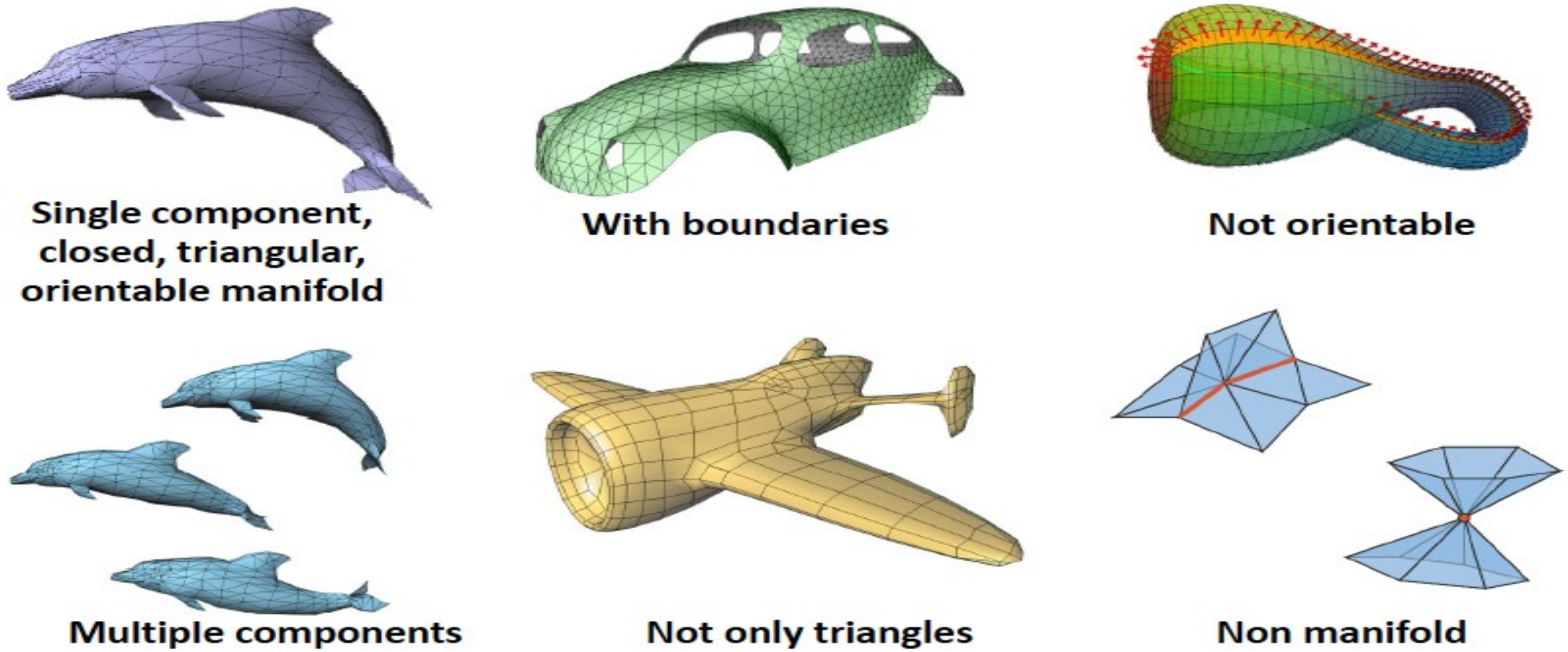
Disk-shaped neighborhoods





non-manifolds

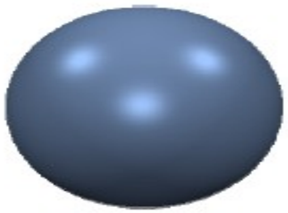
Mesh Characteristics



Genus

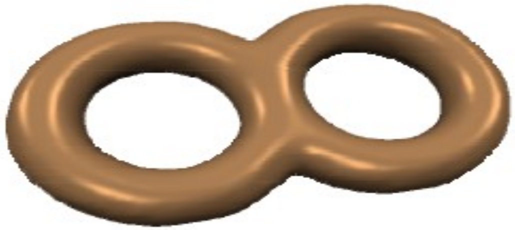
Genus:

Half the maximal number of closed paths that do not disconnect the mesh (= the number of holes)



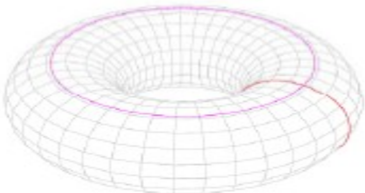
Genus 0

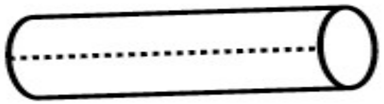
Genus 1



Genus 2

Genus ?





Euler Formula

For a closed (no boundary), manifold, connected surface mesh:

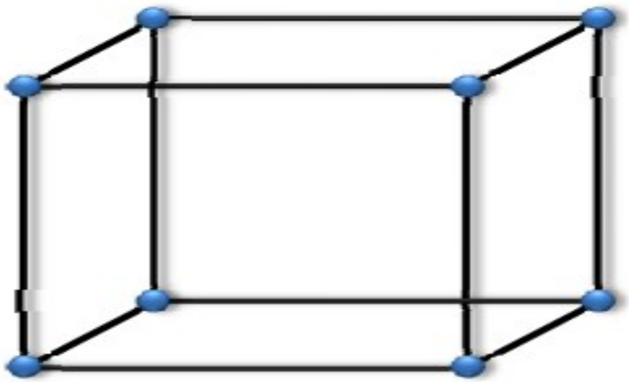
$V\text{-}E\text{+}F\text{=}\ 2(1\text{-}G)$

V = number of vertices
E = number of edges
F = number of faces
G = genus (number of holes in the surface)

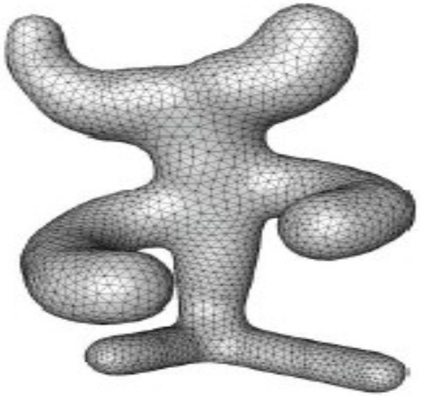
A **2-manifold** is a surface (locally like a plane)

For Closed 2-Manifold Polygonal Meshes

$$V + F - E = \boxed{\chi}$$ **Euler characteristic**



$V = 8$
$E = 12$
$F = 6$
$\chi = 8 + 6 - 12 = \mathbf{2}$

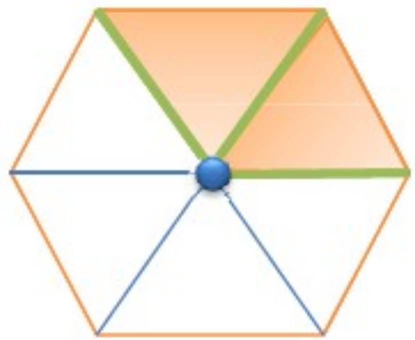


$V = 3890$
$E = 11664$
$F = 7776$
$\chi = \mathbf{2}$

…and if they are triangle meshes

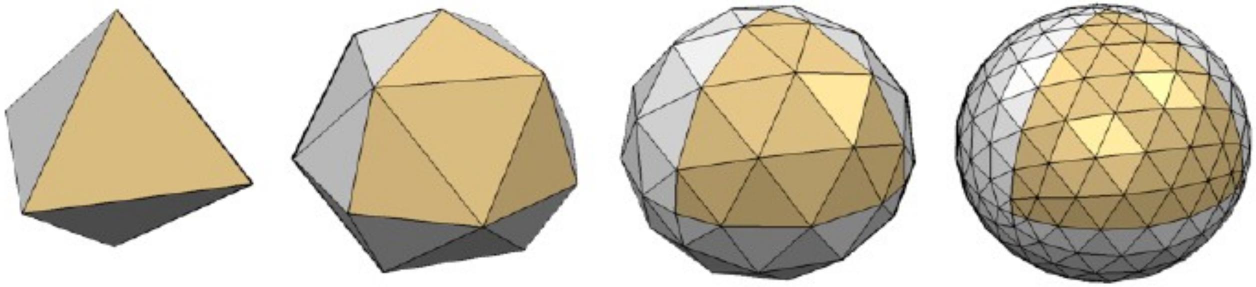
- *Triangle* mesh statistics

$$E \approx 3V$$
$$F \approx 2V$$



- Avg. valence ≈ 6
  Show using Euler Formula

Mesh Data Structures



Need to store

- Geometry
- Connectivity

Can be used as file formats or internal formats

Considerations

- Space
- Efficient operations

Mesh processing has different requirements than rendering

- Example: Smoothing by averaging a vertex with neighbor vertices

Face Set (STL)

- face:
  - 3 positions

Triangles		
x_{11} y_{11} z_{11}	x_{12} y_{12} z_{12}	x_{13} y_{13} z_{13}
x_{21} y_{21} z_{21}	x_{22} y_{22} z_{22}	x_{23} y_{23} z_{23}
...	...	...
x_{F1} y_{F1} z_{F1}	x_{F2} y_{F2} z_{F2}	x_{F3} y_{F3} z_{F3}

36 B/f = 72 B/v

no connectivity!

Indexed Face Set (OBJ)

- vertex:
  - position
- face:
  - vertex indices

Vertices
x_1 y_1 z_1
...
x_V y_V z_V

Triangles
v_{11} v_{12} v_{13}
...
...
...
...
v_{F1} v_{F2} v_{F3}

12 B/v + 12 B/f = 36 B/v

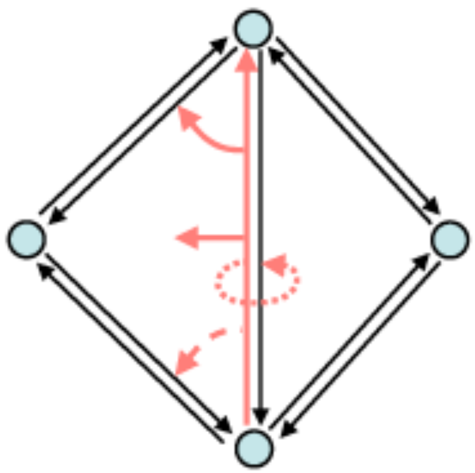
no neighborhood info

Half-Edge Data Structure

Consists of 3 tables (or arrays)

- Vertex table
- Halfedge table
- Face table

- vertex
  - position
  - 1 halfedge
- halfedge
  - 1 vertex
  - 1 face
  - 1, 2, or 3 halfedges
- face
  - 1 halfedge

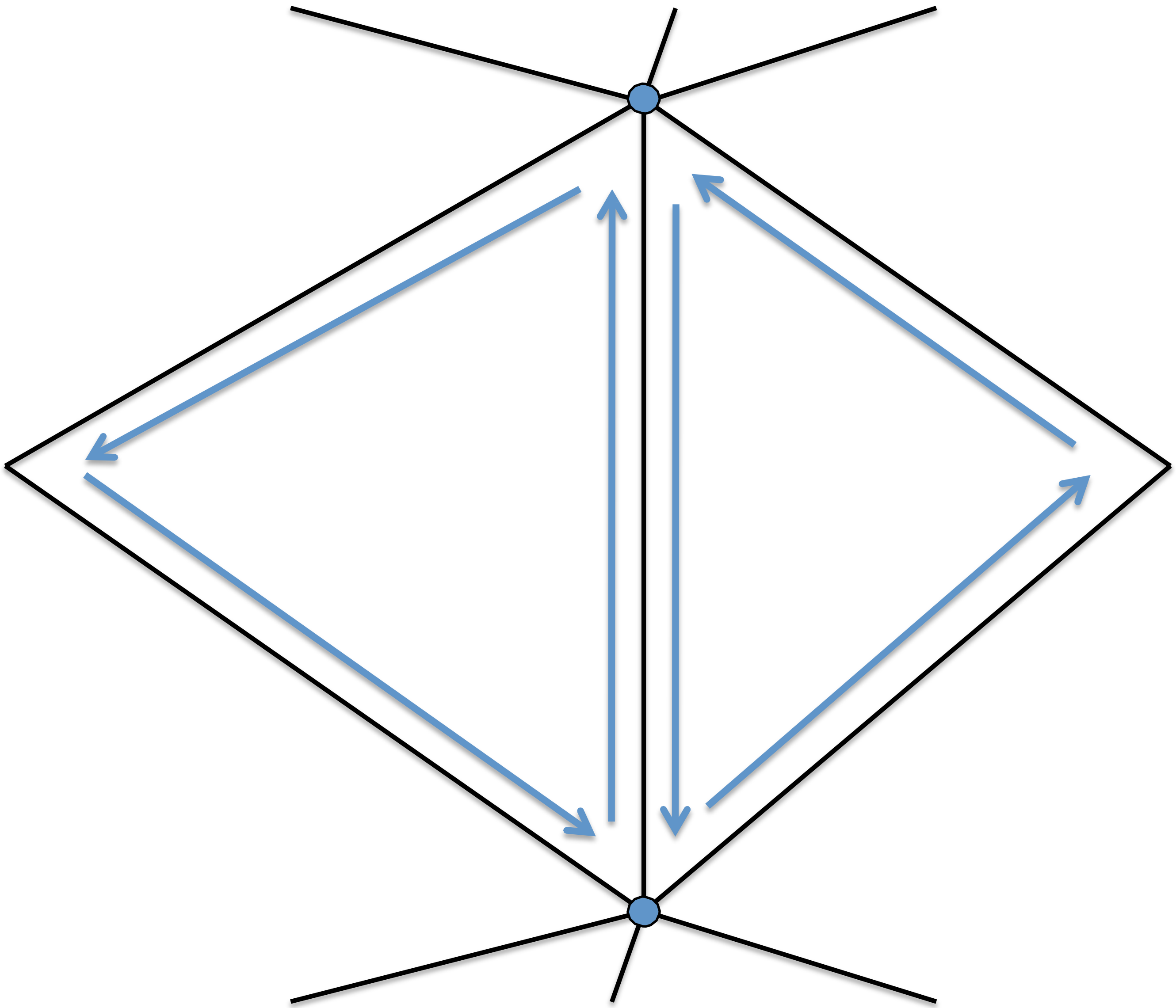


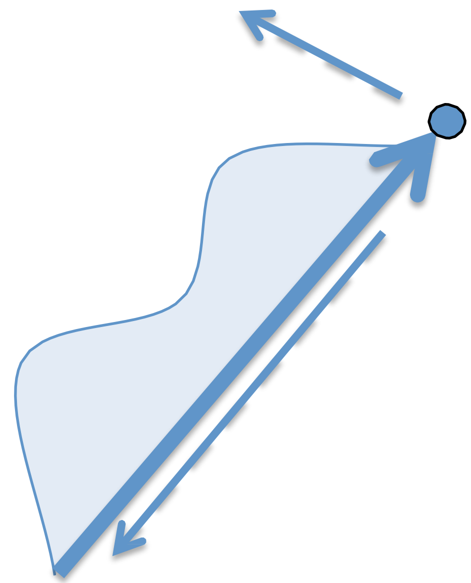
96 to 144 B/v

no case distinctions during traversal

Half Edge

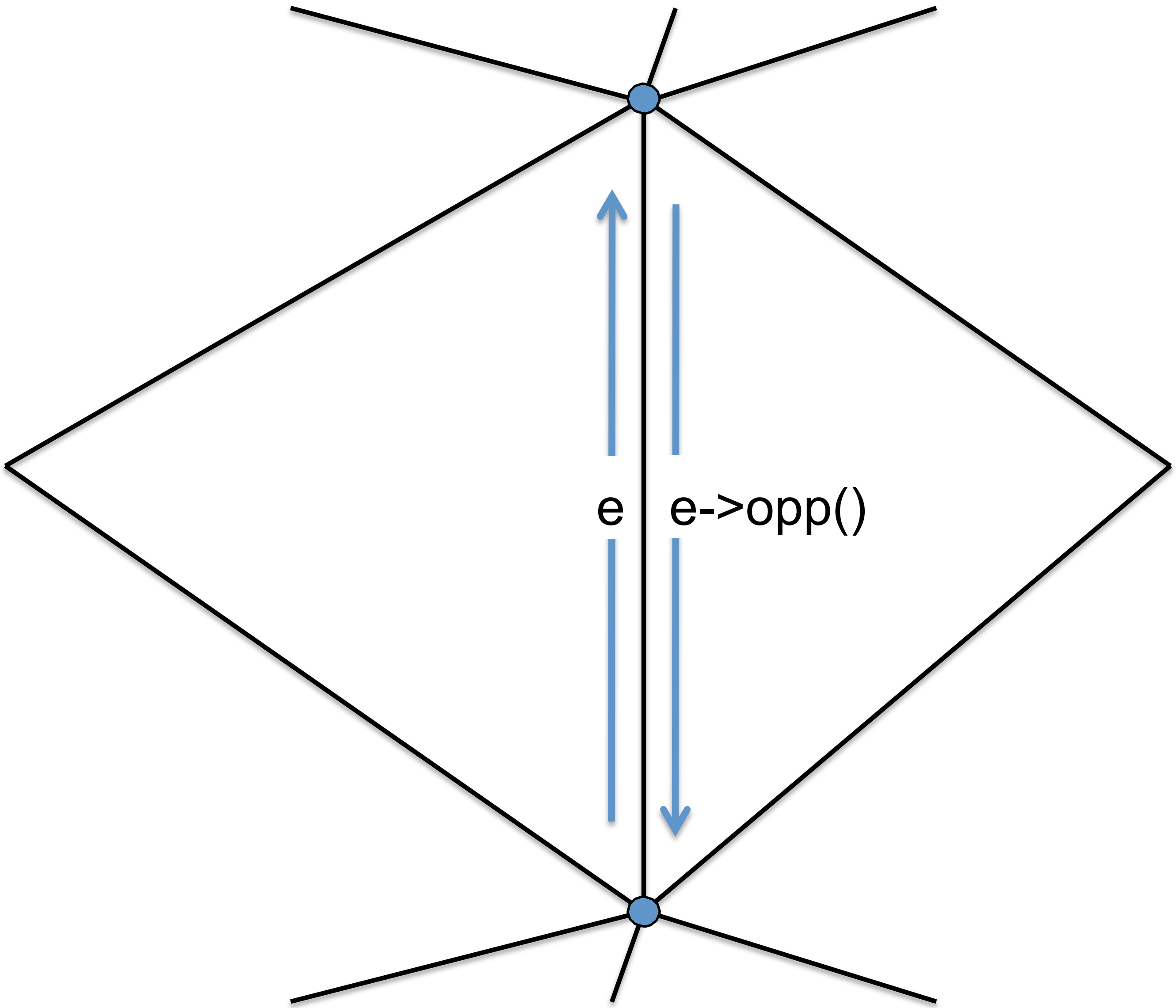
```
class HalfEdge {
  HalfEdge   *opp;
  Vertex     *end;
  Face       *left;
  HalfEdge   *next;
};

HalfEdge e;
```





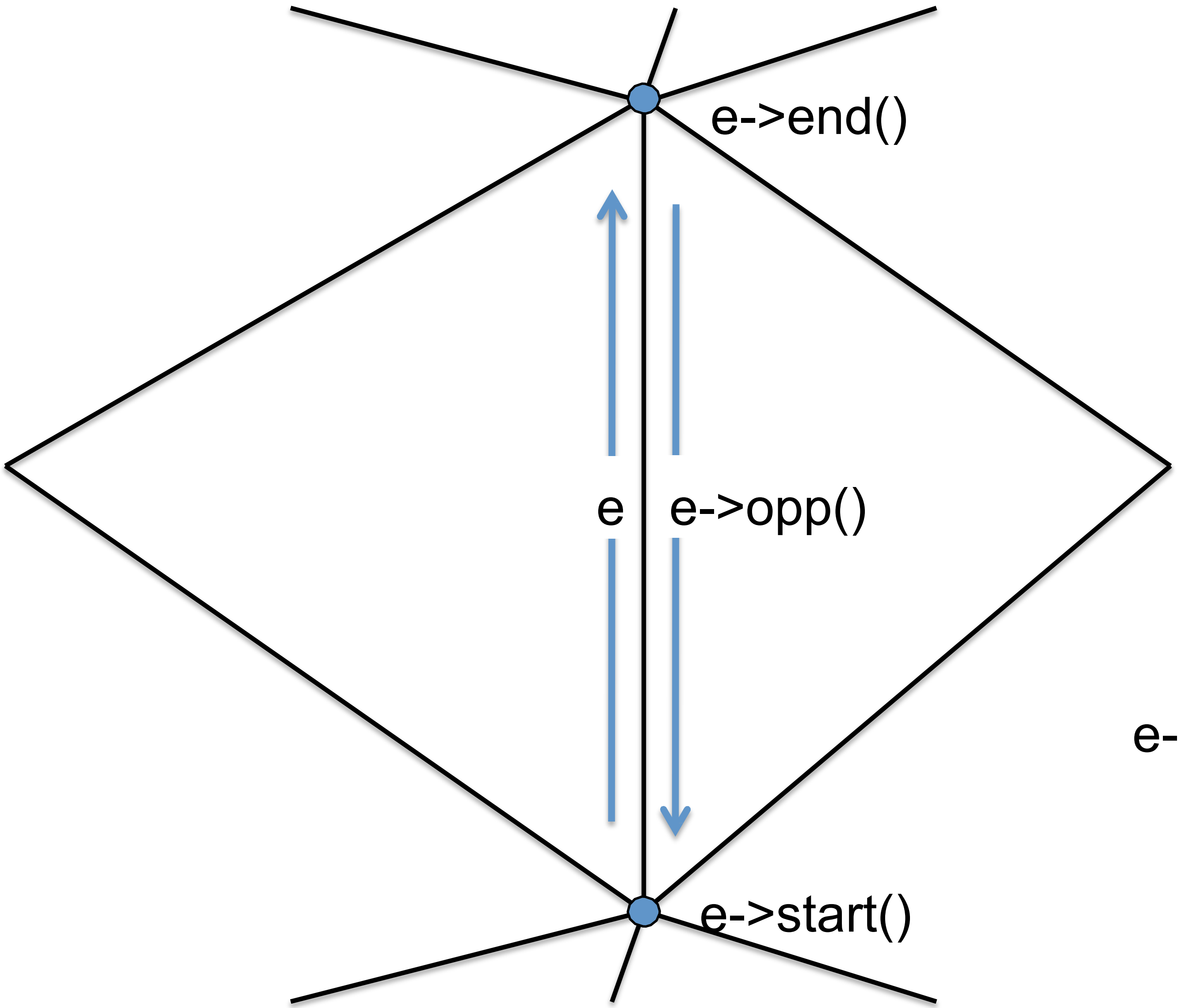
Half Edge

```
class HalfEdge {
  HalfEdge   *opp;
  Vertex     *end;
  Face       *left;
  HalfEdge   *next;
};

HalfEdge e;
```



Half Edge

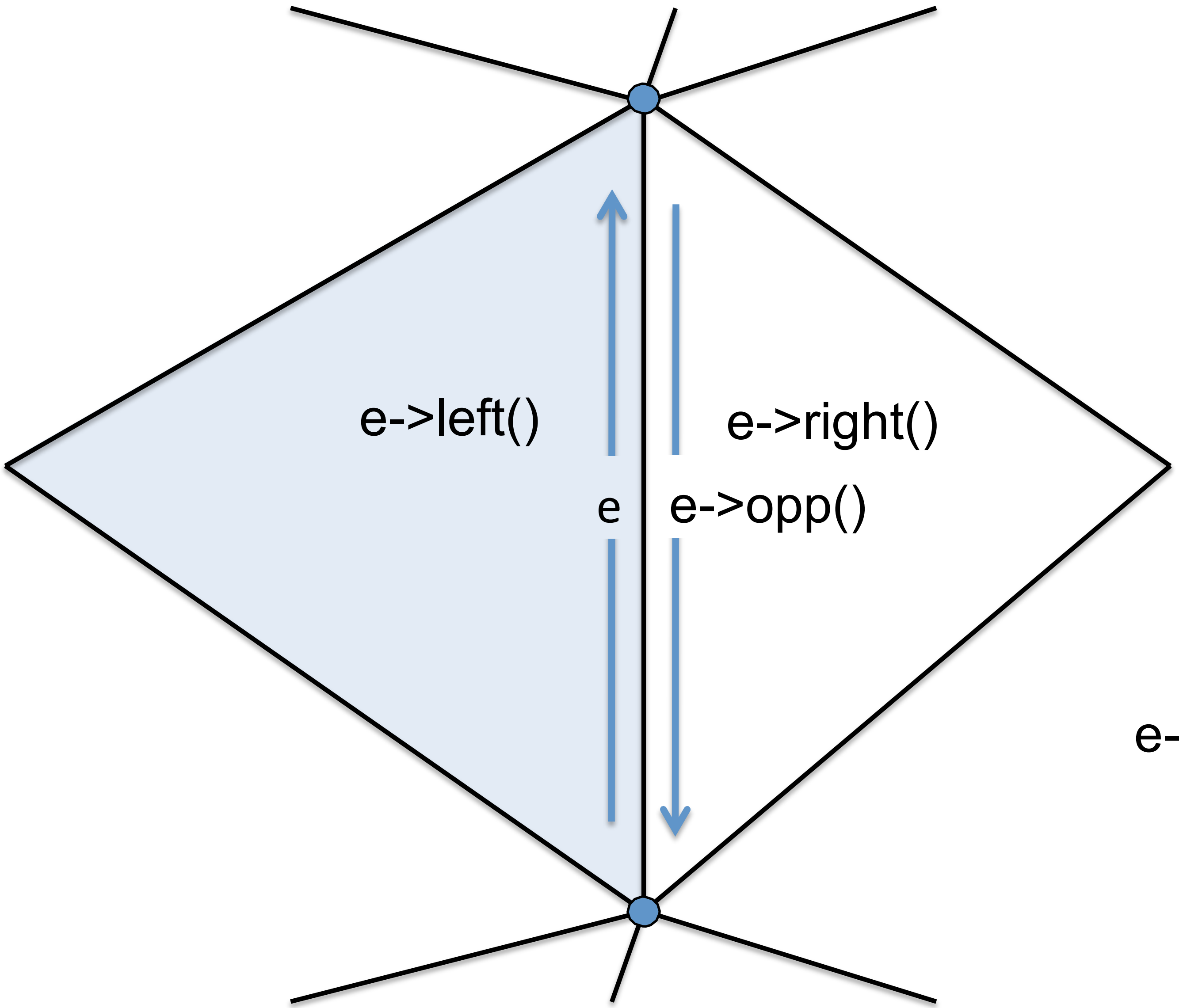
```
class HalfEdge {
  HalfEdge   *opp;
  Vertex     *end;
  Face       *left;
  HalfEdge   *next;
};

HalfEdge e;
```



e->start() = e->opp()->end();

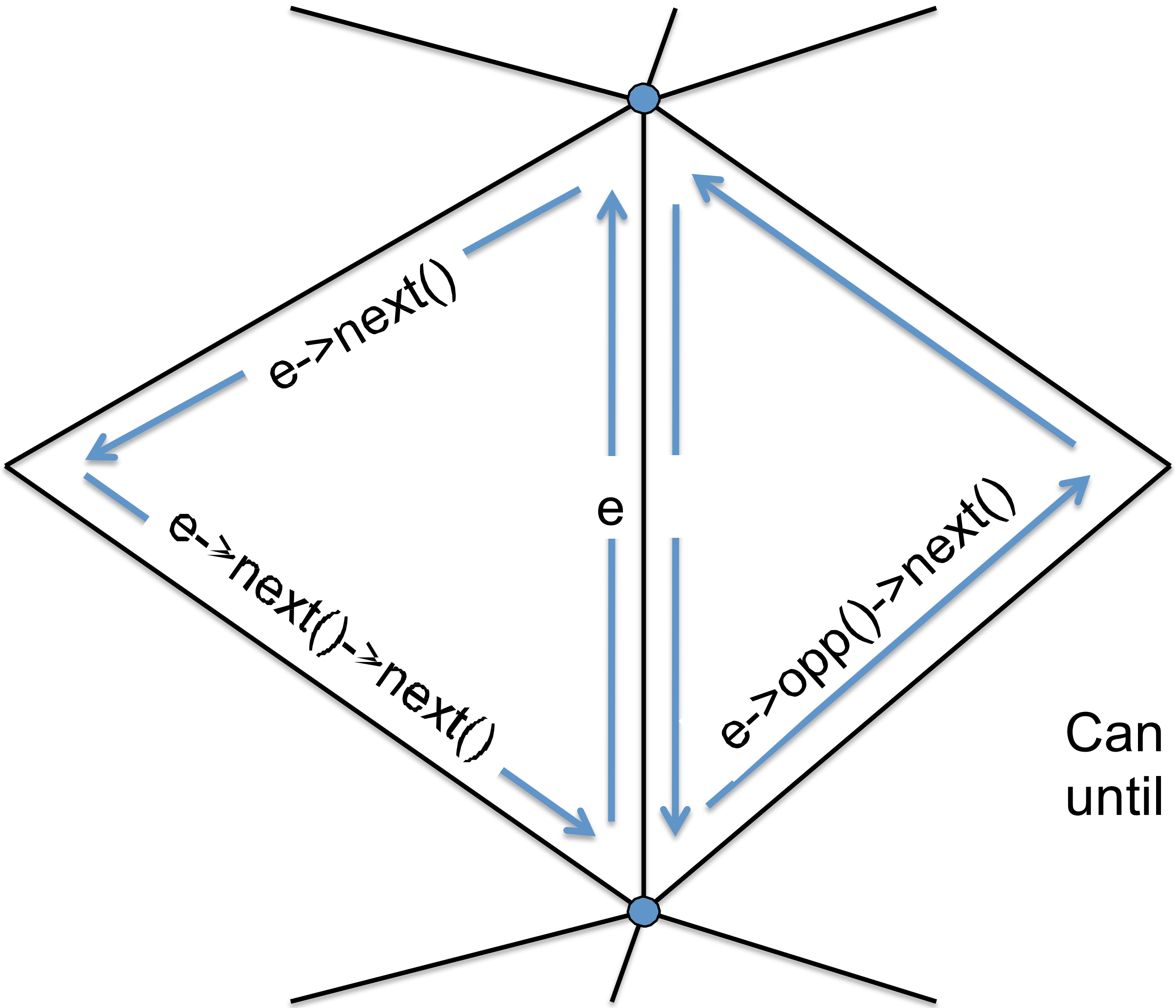
Half Edge

```
class HalfEdge {
  HalfEdge   *opp;
  Vertex     *end;
  Face       *left;
  HalfEdge   *next;
};

HalfEdge e;
```



e->right() = e->opp()->left();

Half Edge




```
class HalfEdge {
  HalfEdge   *opp;
  Vertex     *end;
  Face       *left;
  HalfEdge   *next;
};

HalfEdge e;
```

Can walk around left face until e(->next)^n = e

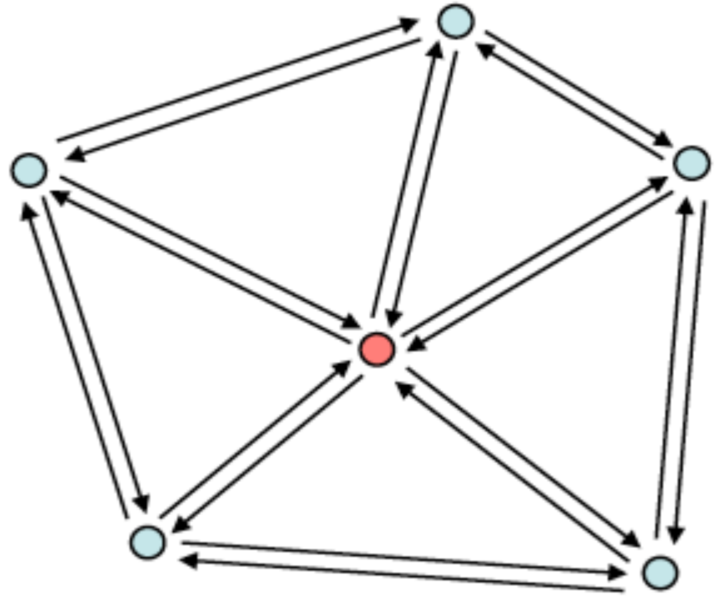
Vertex Star and 1-Rings

Vertex Star: A set consisting of

- the vertex
- its incident edges,
- neighboring vertices

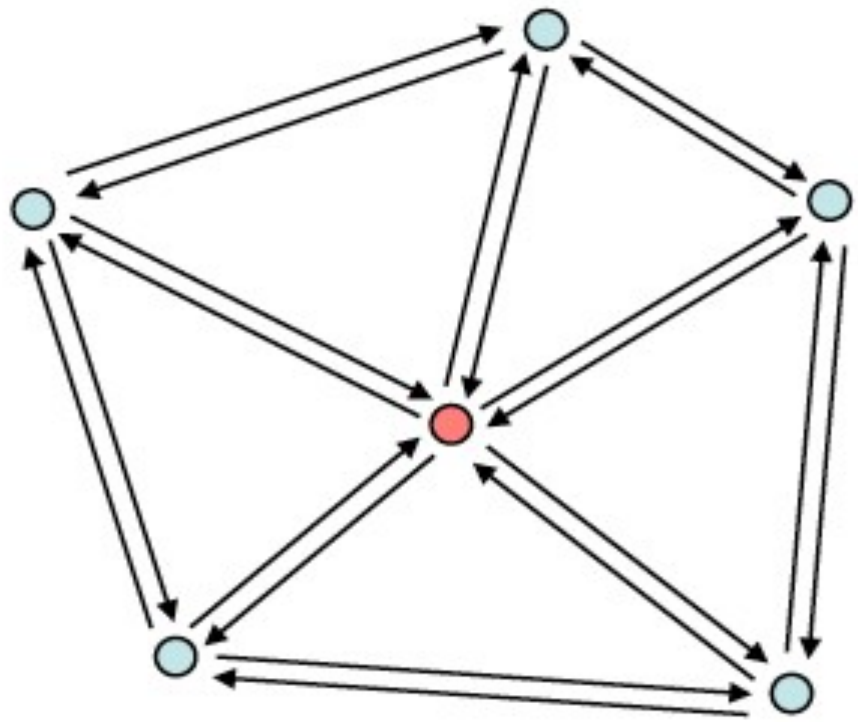
1- Ring: A set consisting of all incident

- Edges
- Faces
- Neighboring verCces



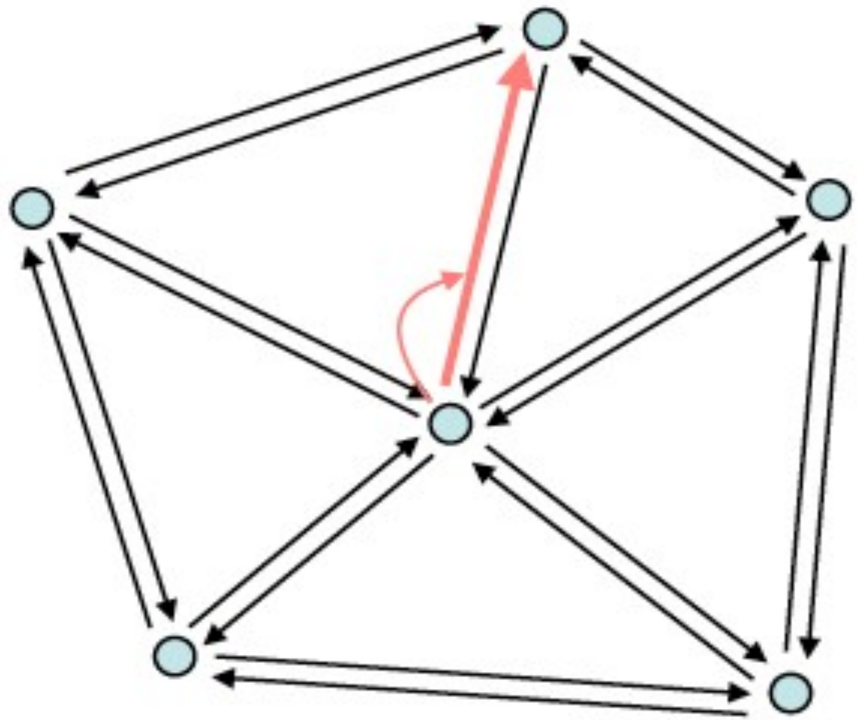
1-Ring Traversal

1. Start at vertex



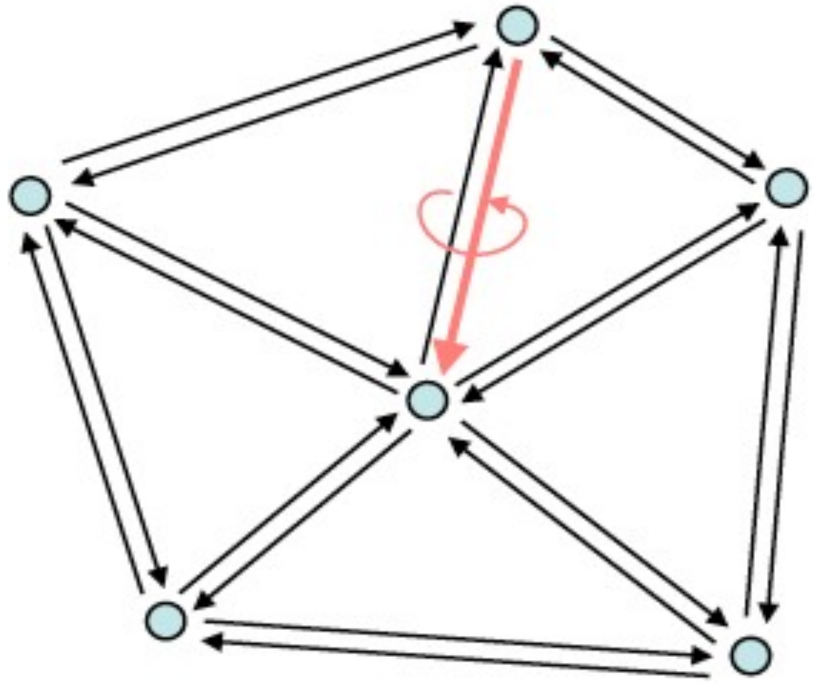
1-Ring Traversal

1. Start at vertex
2. Outgoing halfedge



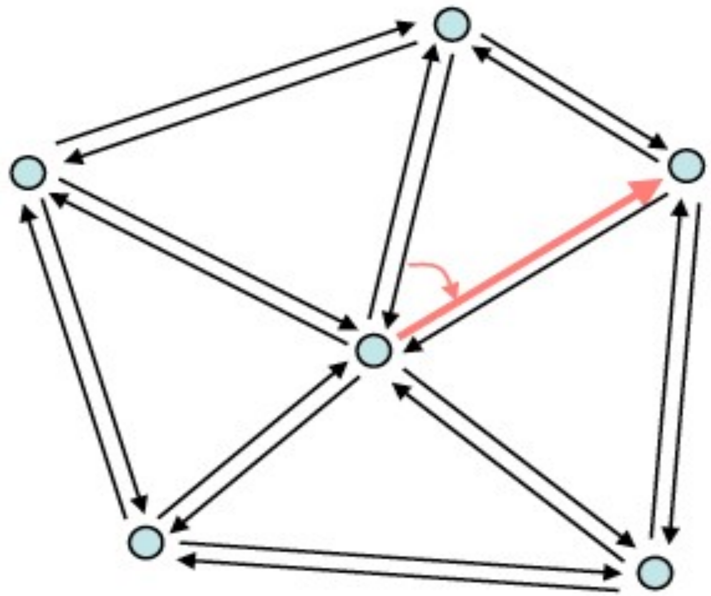
1-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge



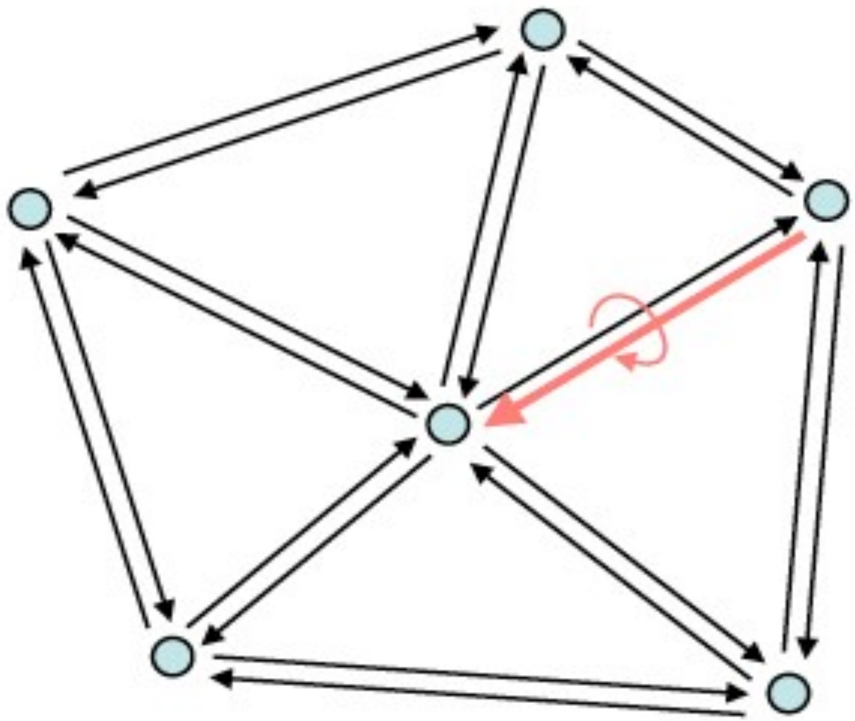
1-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge



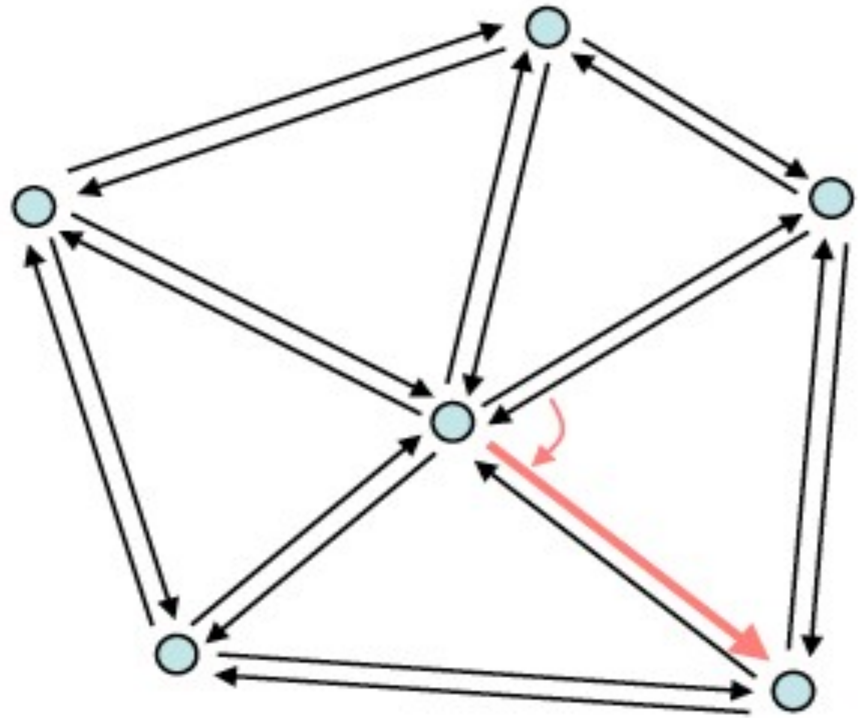
1-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge
5. Opposite



1-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge
5. Opposite
6. Next
7. ...

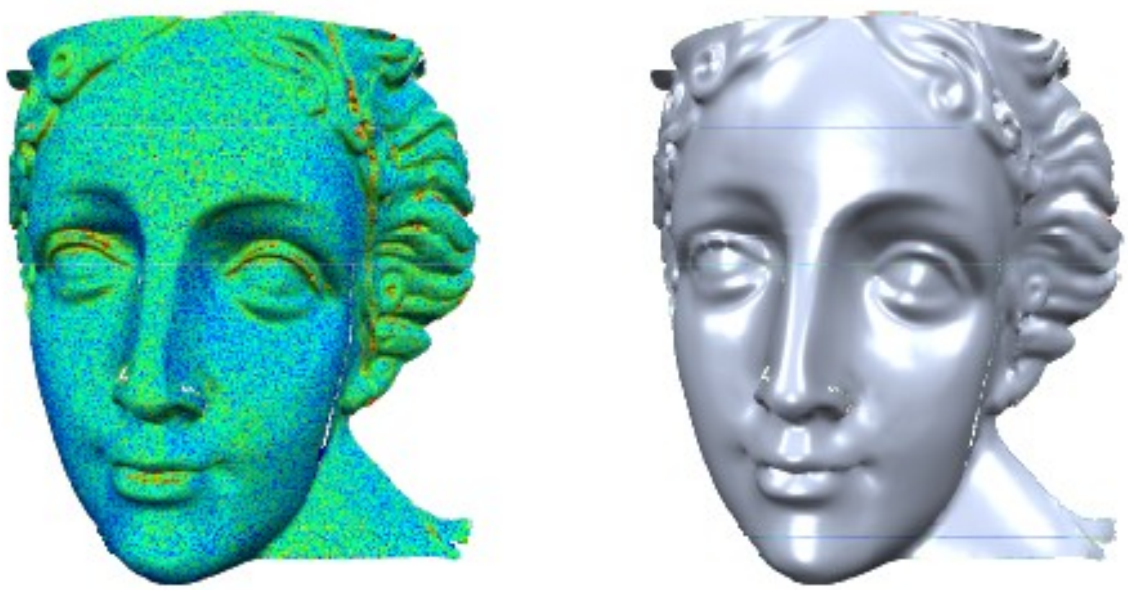


Mesh Smoothing
(aka Denoising, Filtering, Fairing)

Input: Noisy mesh (scanned or other)

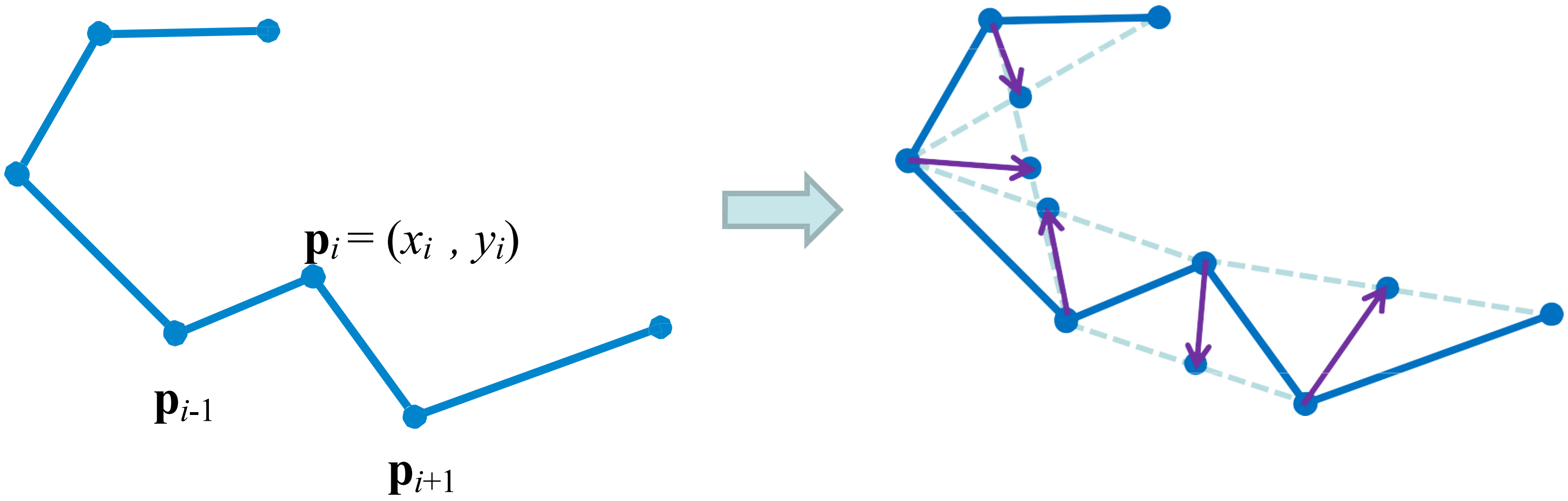
Output: Smooth mesh

How: Filter out high frequency noise



Laplacian Smoothing

An easier problem: How to smooth a curve?

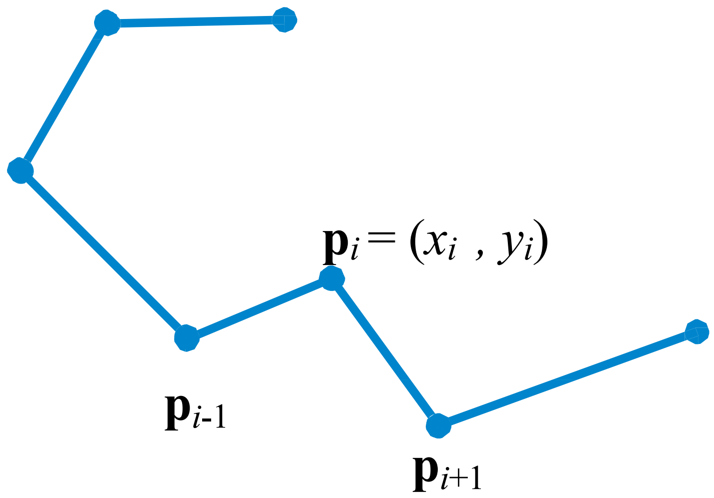


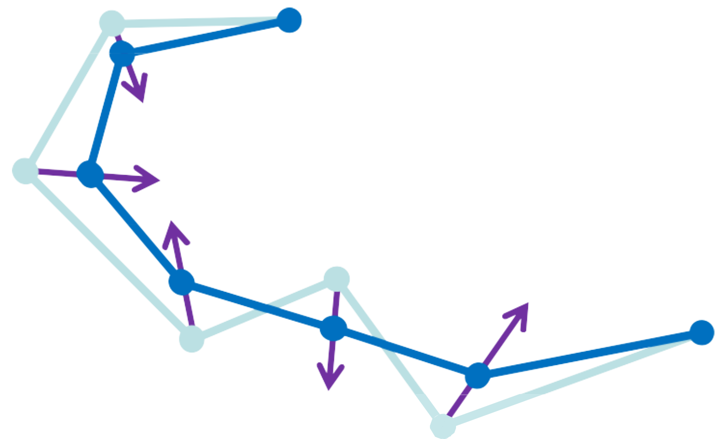
$$(\mathbf{p}_{i-1} + \mathbf{p}_{i+1})/2 - \mathbf{p}_i$$

$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i)$$

Laplacian Smoothing

An easier problem: How to smooth a curve?





$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \frac{1}{2} L(\mathbf{p}_i)$$

Finite difference discretization of second derivative = Laplace operator in one dimension

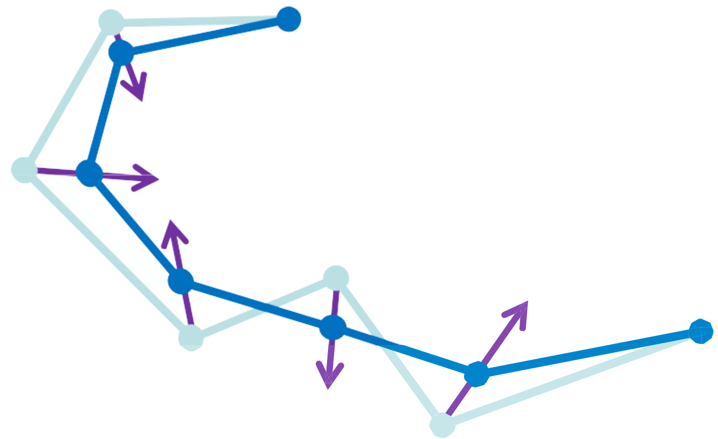
$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i)$$

Laplacian Smoothing

<u>Algorithm:</u>

Repeat for *m* iterations (for non boundary points):

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \lambda L(\mathbf{p}_i)$$



For which λ?

$0 < \lambda < 1$

Closed curve converges to?

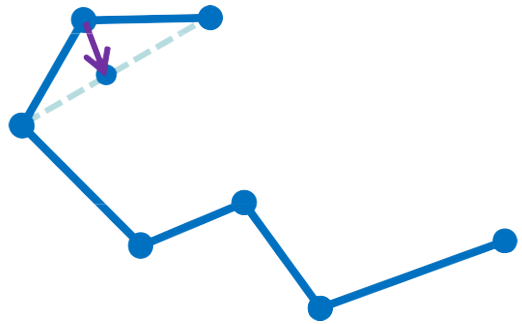
Single point

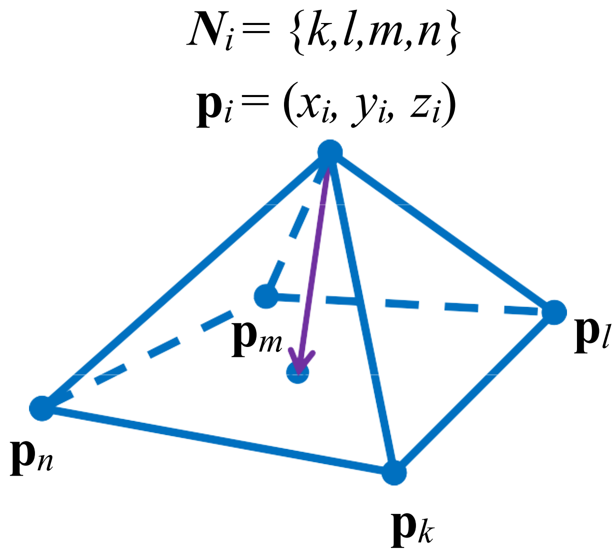
Laplacian Smoothing on Meshes

Same as for curves:

$$\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} + \lambda \Delta \mathbf{p}_i^{(t)}$$

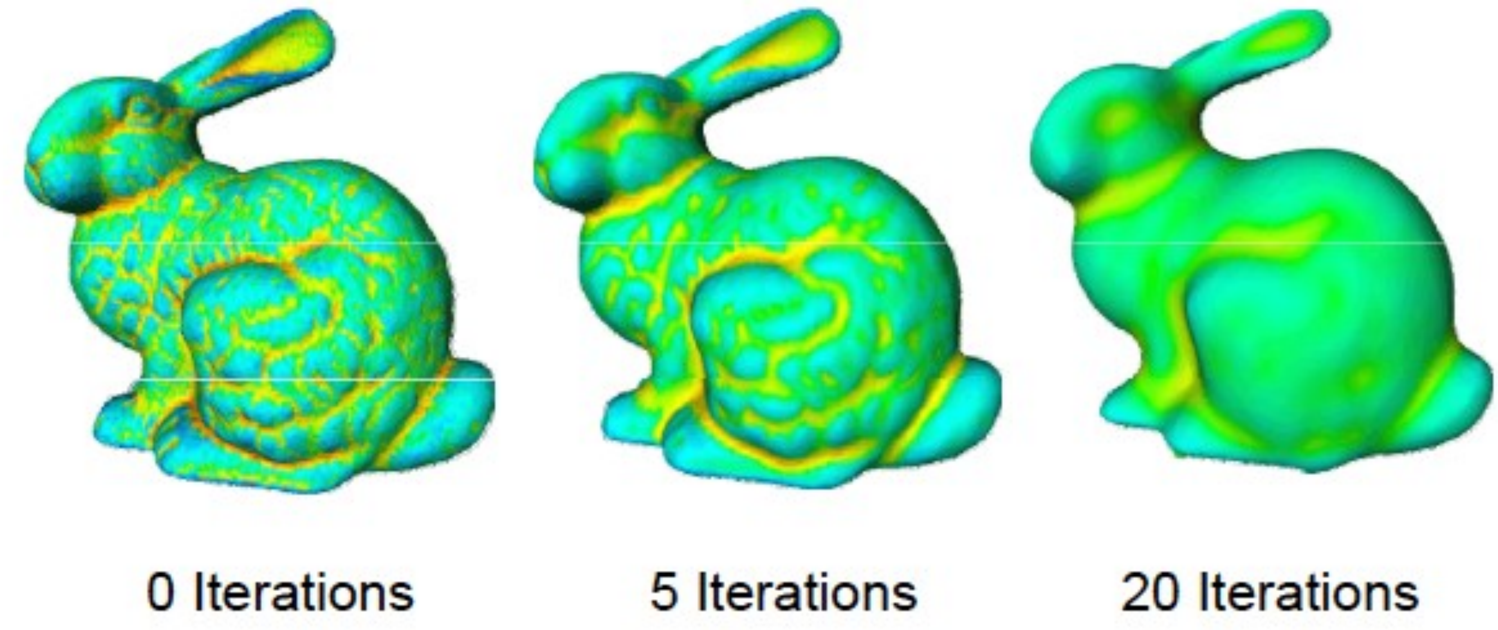
What is $\Delta \mathbf{p}_i$?





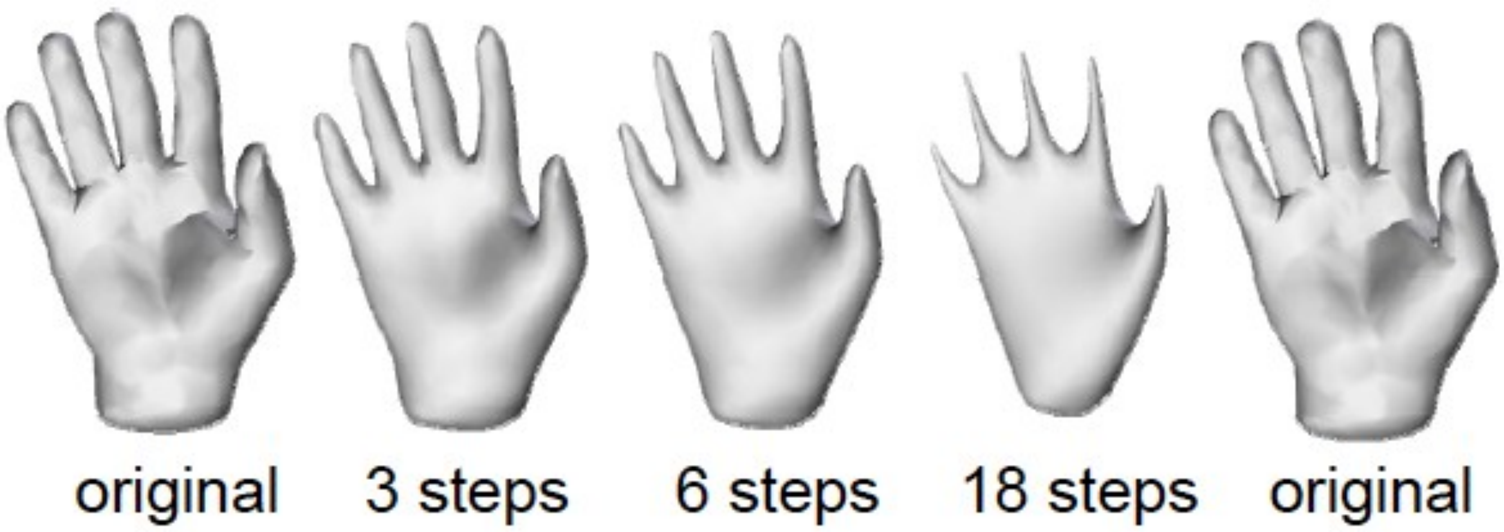
$$\frac{1}{2}\left(\mathbf{p}_{i+1} + \mathbf{p}_{i-1}\right) - \mathbf{p}_i \qquad \frac{1}{|N_i|}\left(\sum_{j \in N_i} \mathbf{p}_j\right) - \mathbf{p}_i$$

Laplacian Smoothing on Meshes



Problem - Shrinkage

Repeated iterations of Laplacian smoothing shrinks the mesh



Taubin Smoothing

Iterate:

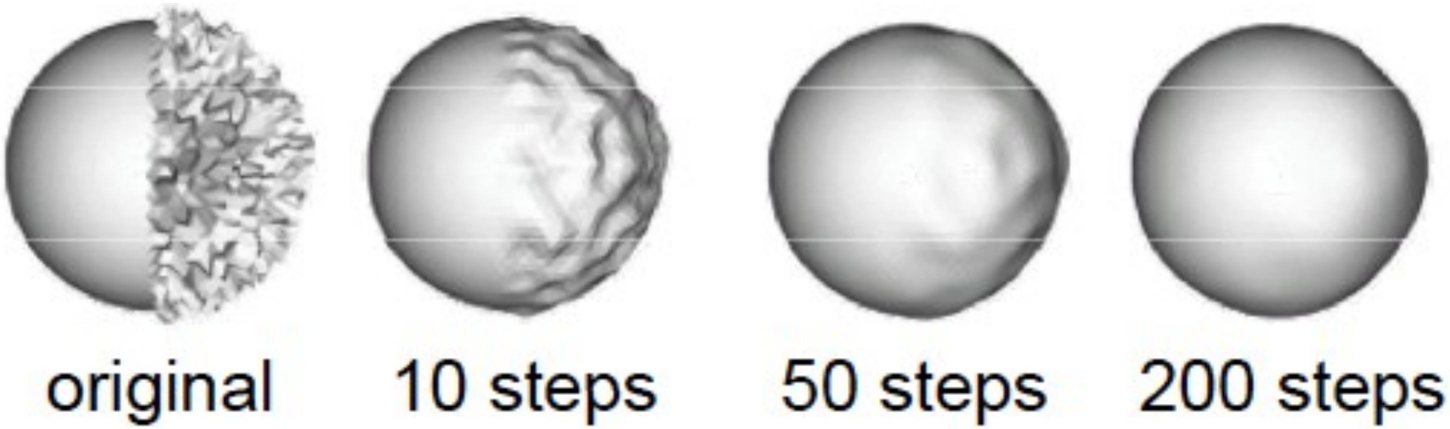
$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \lambda \Delta \mathbf{p}_i$$

Shrink

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \mu \Delta \mathbf{p}_i$$

Inflate

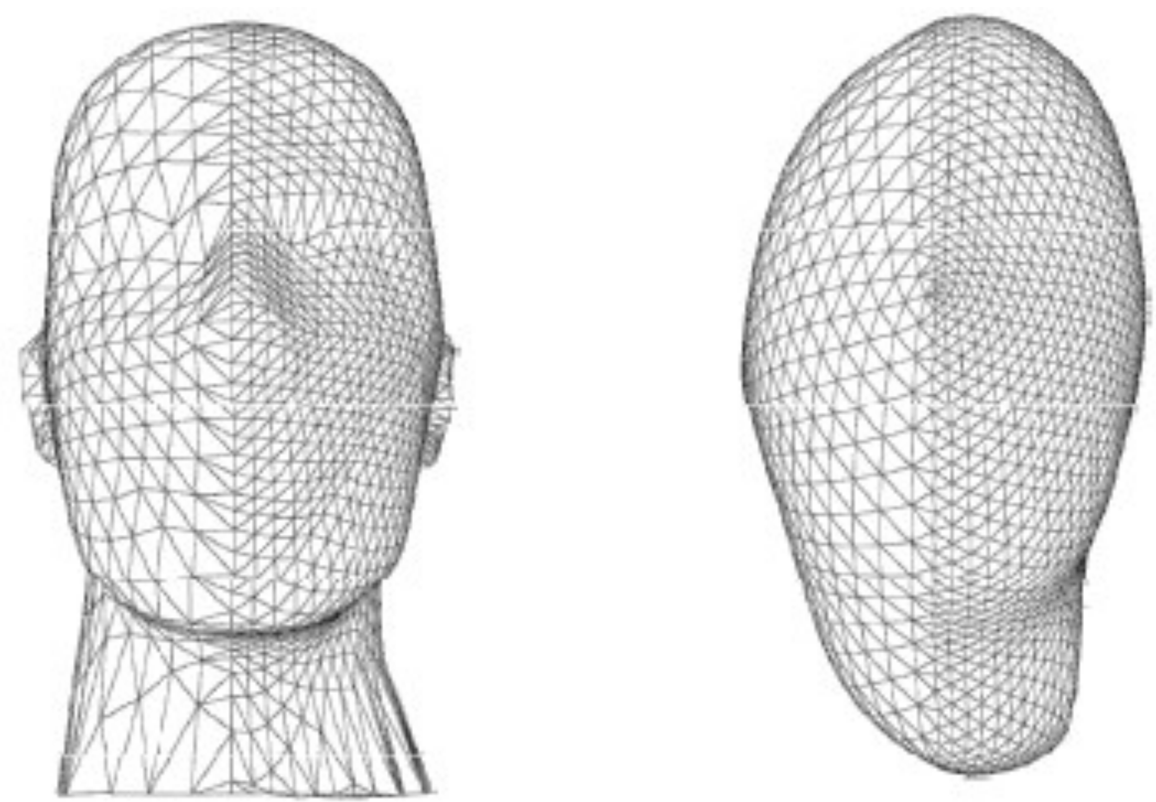
with $\lambda > 0$ and $\mu < 0$



From Taubin, Siggraph 1995

Laplace Operator Discretization
The Problem

The result should not depend on triangle sizes



From Desbrun et al., Siggraph 1999

Laplace Operator Discretization

What Went Wrong?

Back to curves:

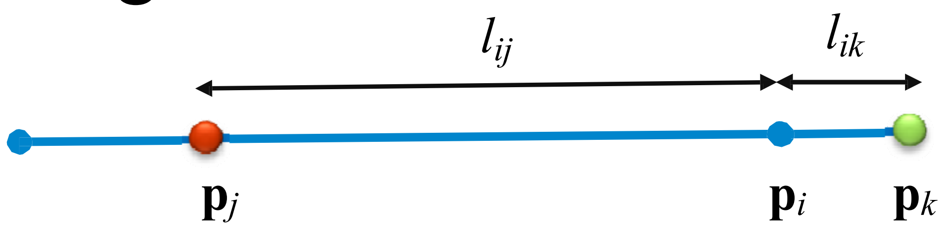
$$\boxed{\frac{1}{2}}(\mathbf{p}_{i+1}+\mathbf{p}_{i-1})-\mathbf{p}_i$$

Same weight for both neighbors, although one is closer

Laplace Operator Discretization

The Solution

Use a weighted average to define Δ



Which weights?

$$w_{ij} = \frac{1}{l_{ij}} \qquad w_{ik} = \frac{1}{l_{ik}}$$

$$L(\mathbf{p}_i) = \frac{w_{ij}\mathbf{p}_j + w_{ik}\mathbf{p}_k}{w_{ij} + w_{ik}} - \mathbf{p}_i$$

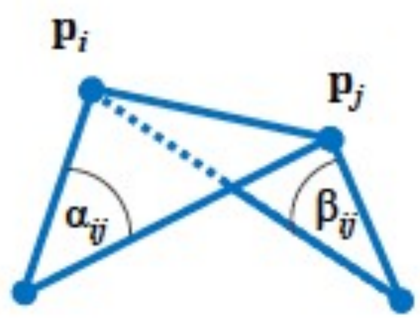
Straight curves will be invariant to smoothing

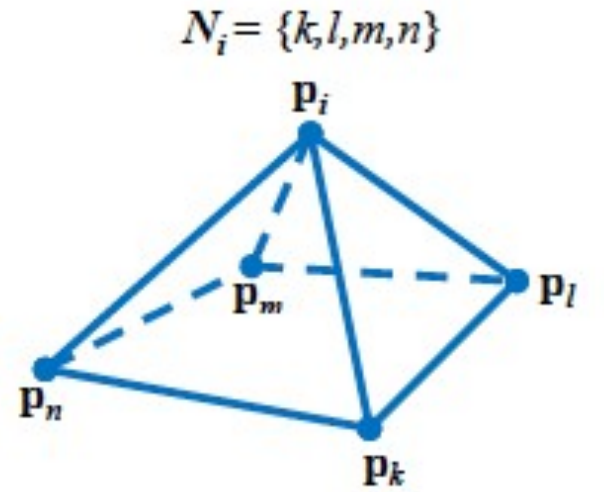
Laplace Operator Discretiztion

Cotangent Weights

Use a weighted average to define Δ

Which weights?





$$w_{ij} = \frac{h_{ij}^1 + h_{ij}^2}{l_{ij}} = \frac{1}{2}\left(\cot\alpha_{ij} + \cot\beta_{ij}\right) \qquad L(\mathbf{p}_i) = \frac{1}{\sum w_{ij}} \sum_{j \in N_i} w_{ij}\left(\mathbf{p}_j - \mathbf{p}_i\right)$$

Planar meshes will be invariant to smoothing

Contangent Weights

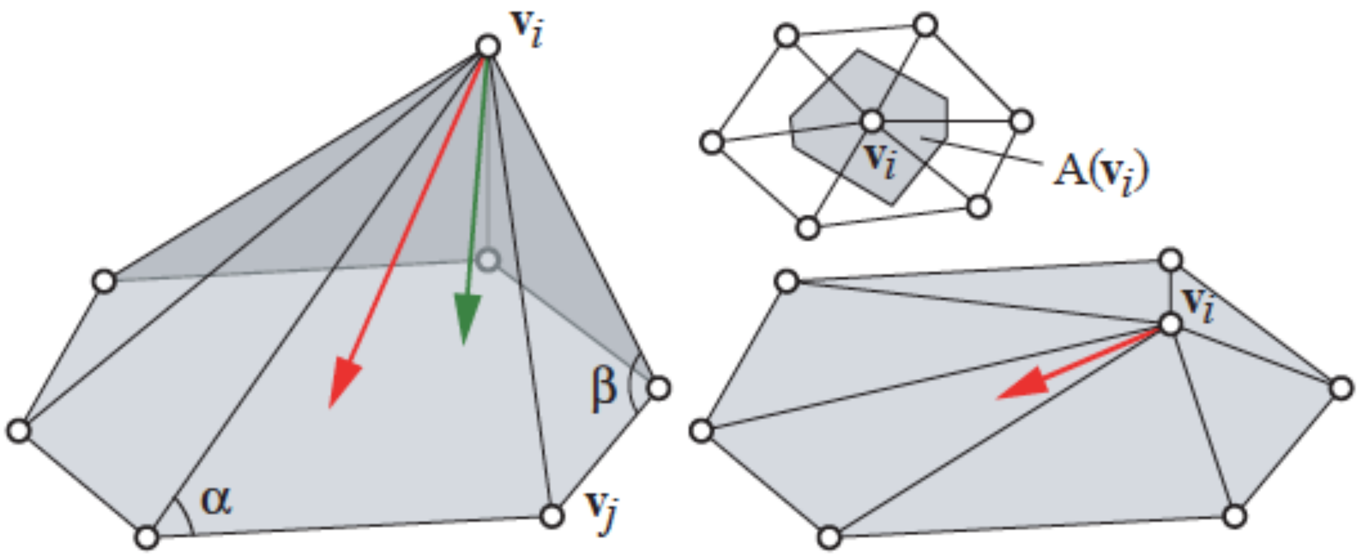
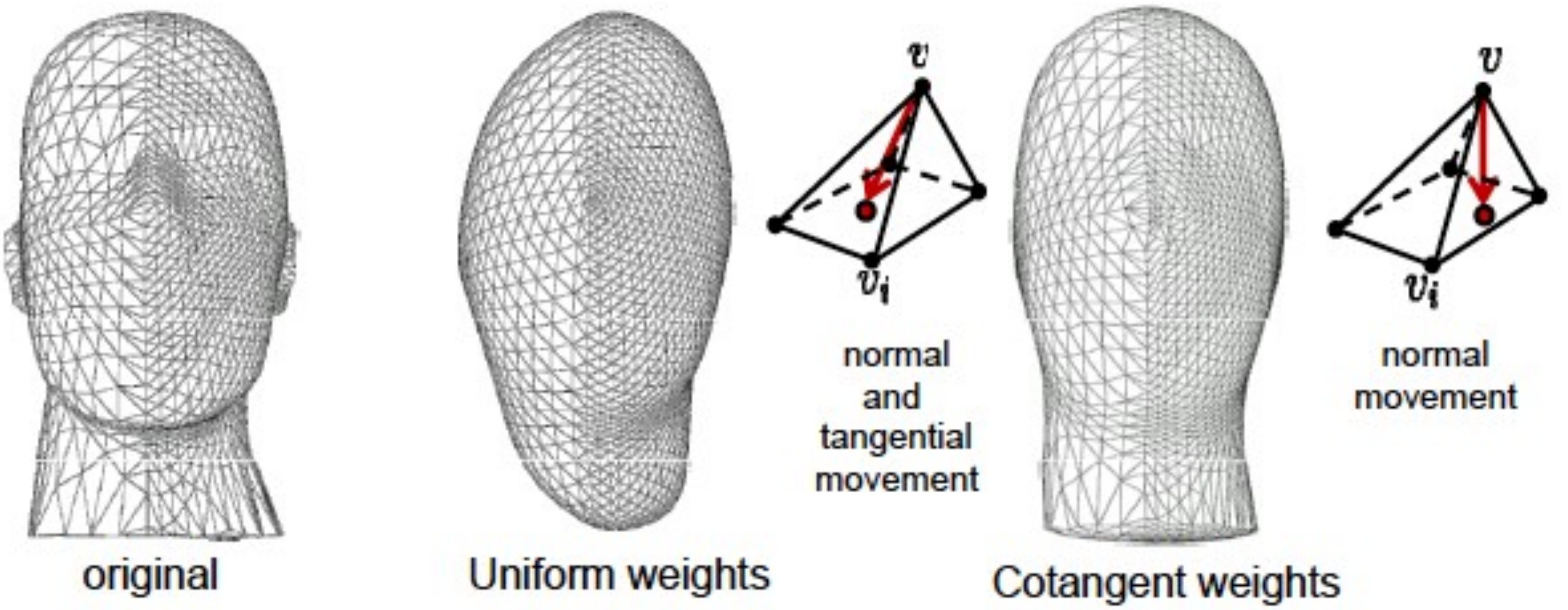


Figure 4: Left: uniform (red) and cotangent (green) Laplacian vectors for a vertex $\mathbf{v}_i$ and its (in this case planar) 1-ring, as well as the angles used in Eqn. 4 for one $\mathbf{v}_j$, Bottom right: the effect of flattening $\mathbf{v}_i$ into the 1-ring plane. While the cotangent Laplacian vanishes, the uniform Laplacian generally does not. Right top: the Voronoi region $A(\mathbf{v}_i)$ around a vertex.

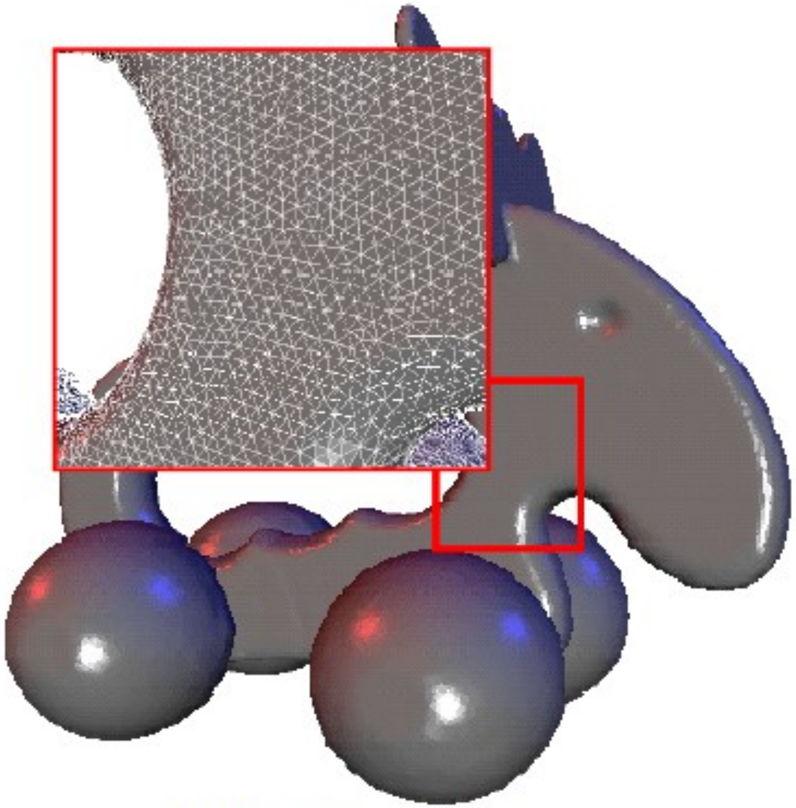
Smoothing with the Cotangent Laplacian



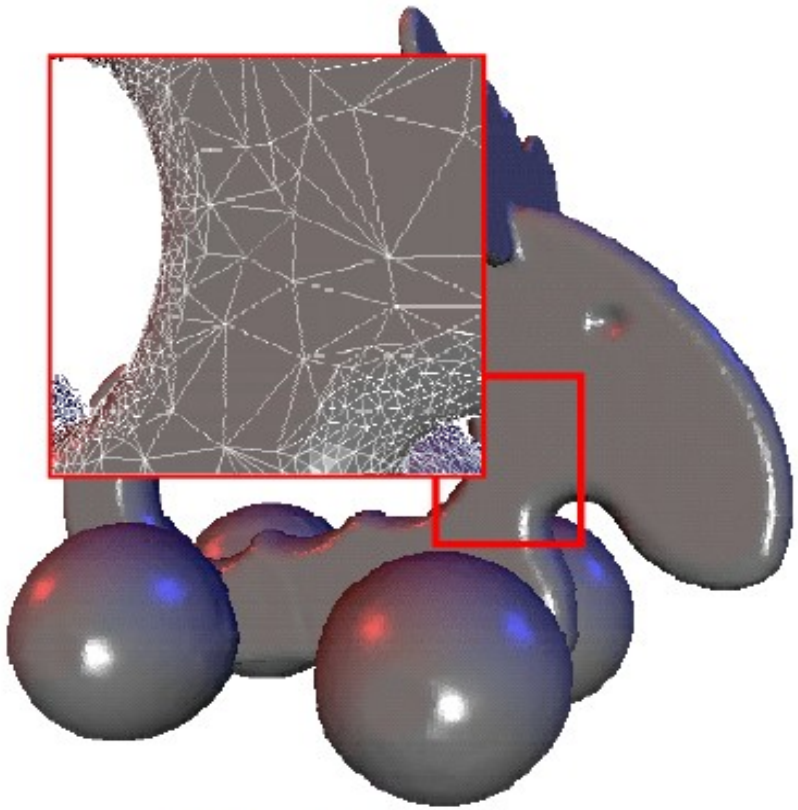
From Desbrun et al., Siggraph 1999

Mesh Simplification

- Oversampled 3D scan data



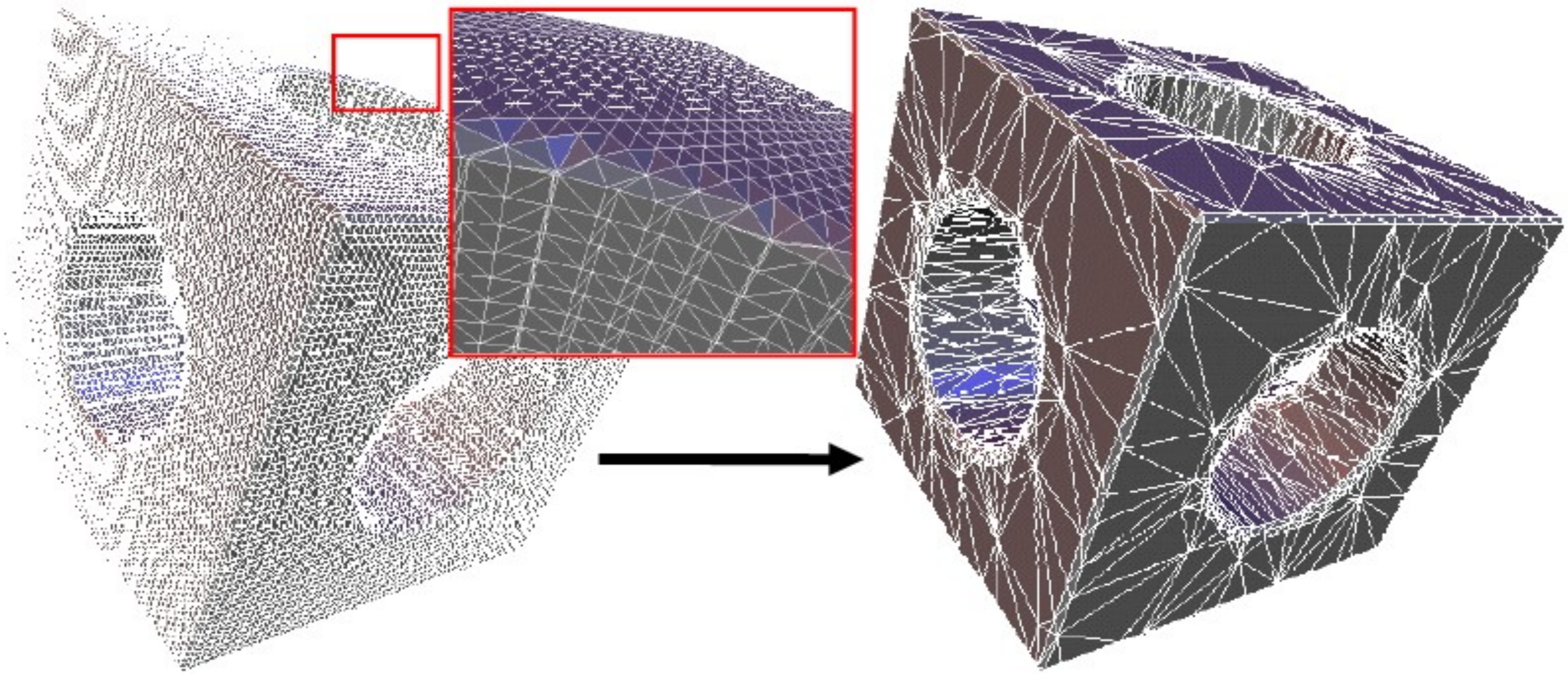
~150k triangles



~80k triangles

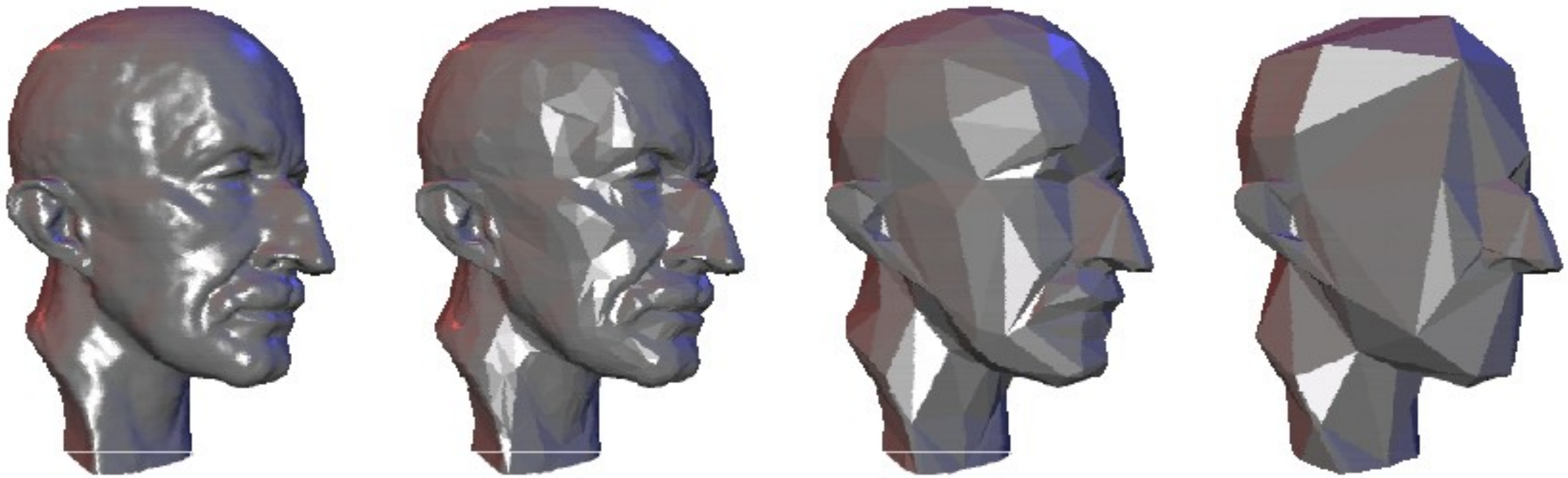
Simplification: Applications

- Overtessellation: E.g. iso-surface extraction



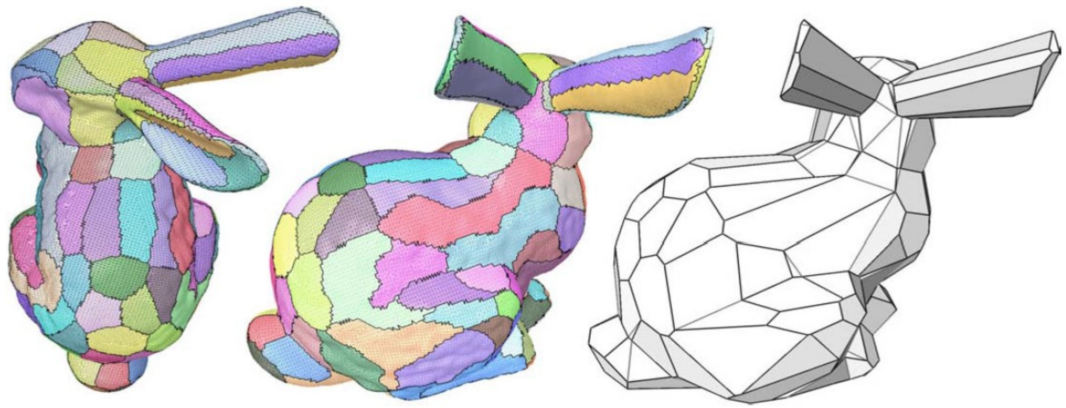
Simplifcation: Applications

- Multi-resolution hierarchies for
  - efficient geometry processing
  - level-of-detail (LOD) rendering



Mesh Decimation Methods

- **Vertex clustering**
- **Incremental decimation**
- Resampling
- Mesh approximation

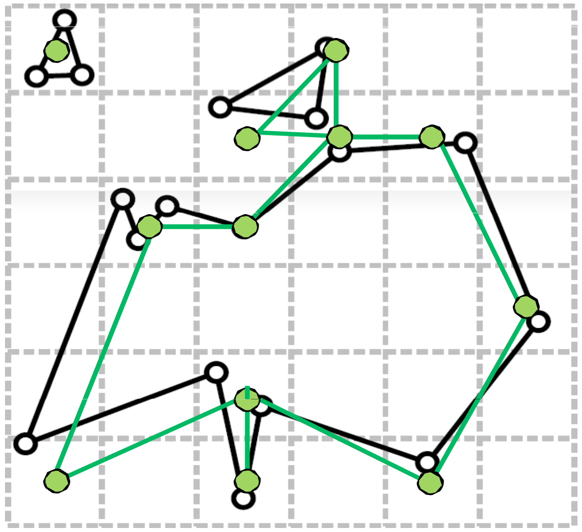


Vertex Clustering

- Cluster Generation
- Computing a representative
- Mesh generation
- Topology changes

Vertex Clustering

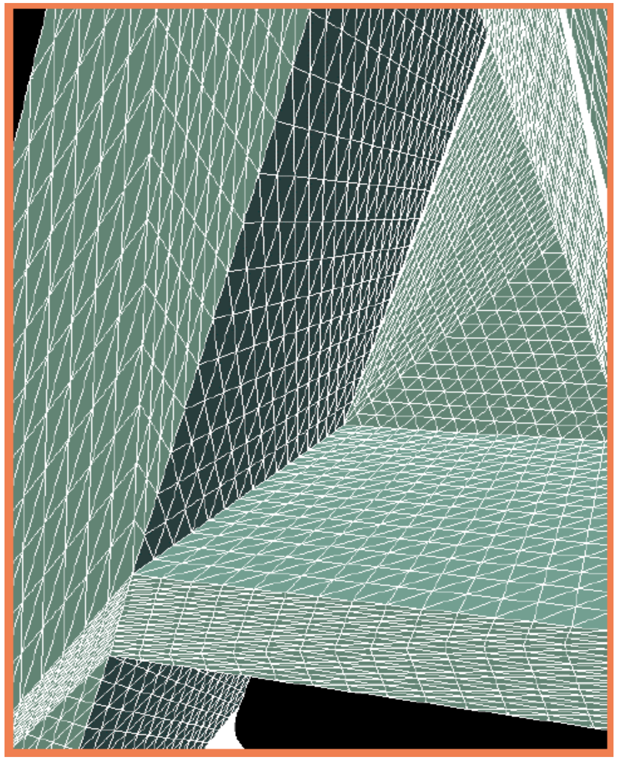
- Cluster Generation
  - Uniform 3D grid
  - Map vertices to cluster cells
- Computing a representative
- Mesh generation
- Topology changes

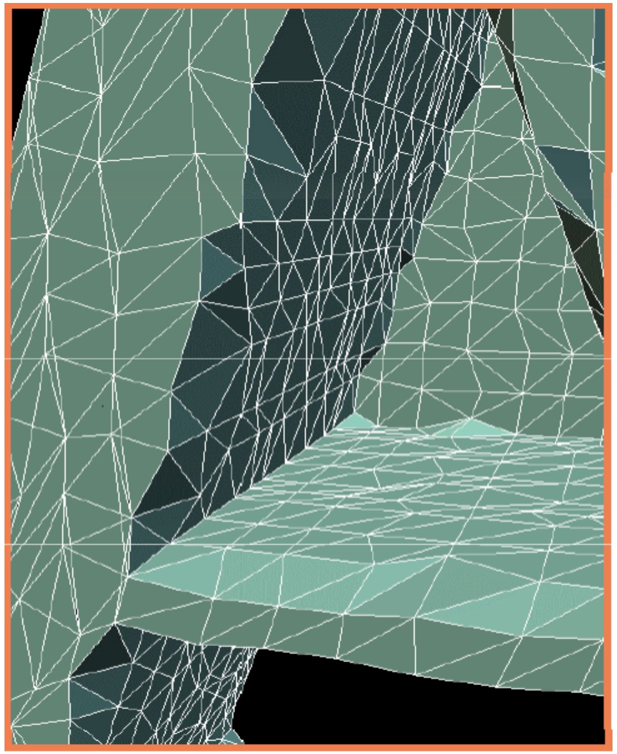


Vertex Clustering

- Cluster Generation
- Computing a representative
  - Average/median vertex position
  - Error quadrics
- Mesh generation
- Topology changes

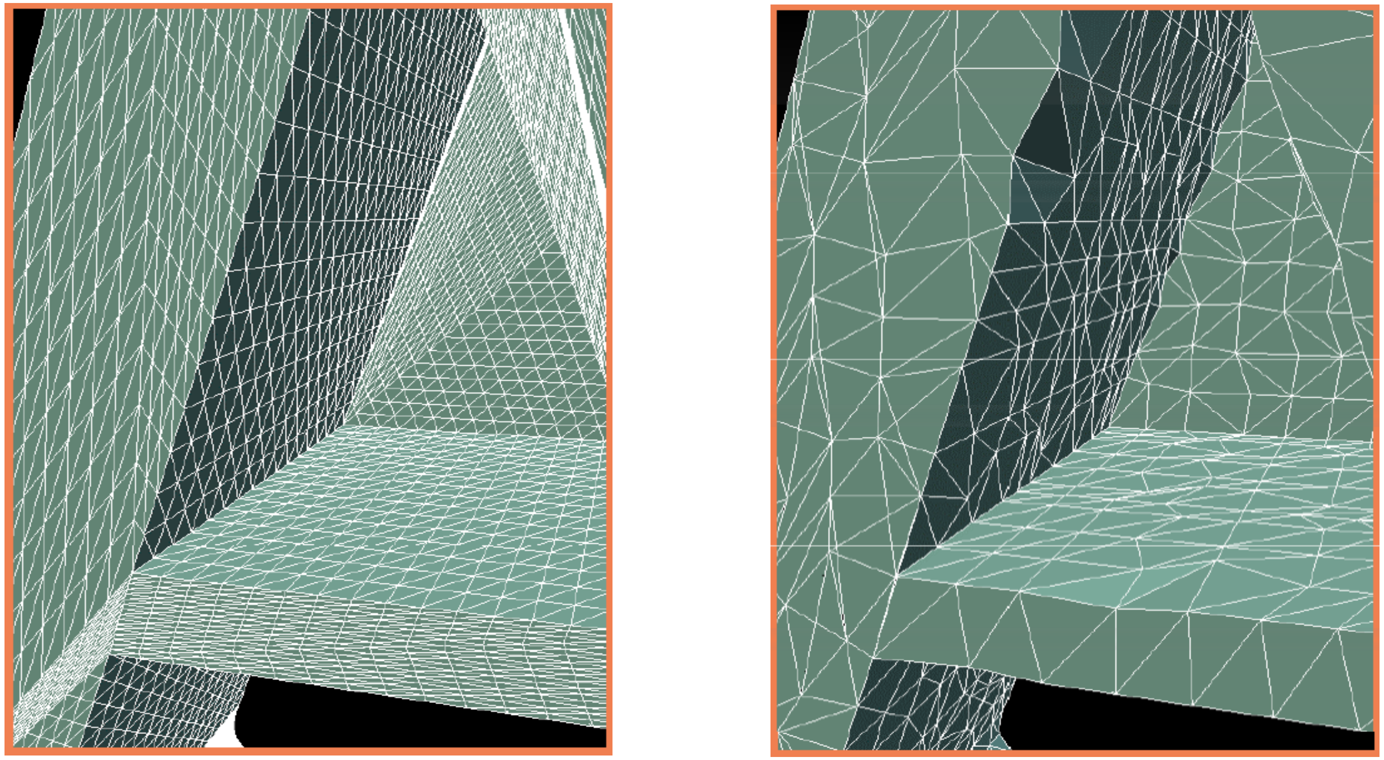
Computing a Representative





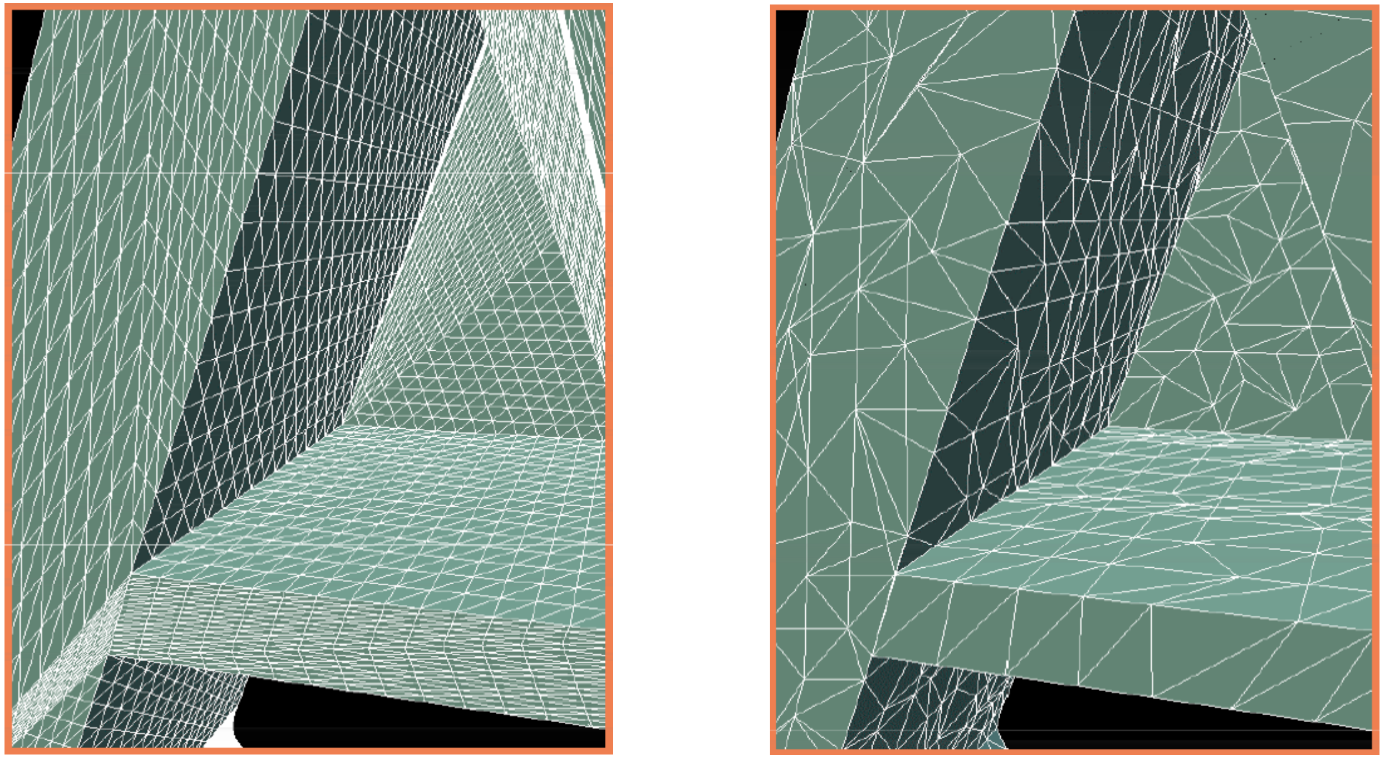
Average vertex position

Computing a Representative



Median vertex position

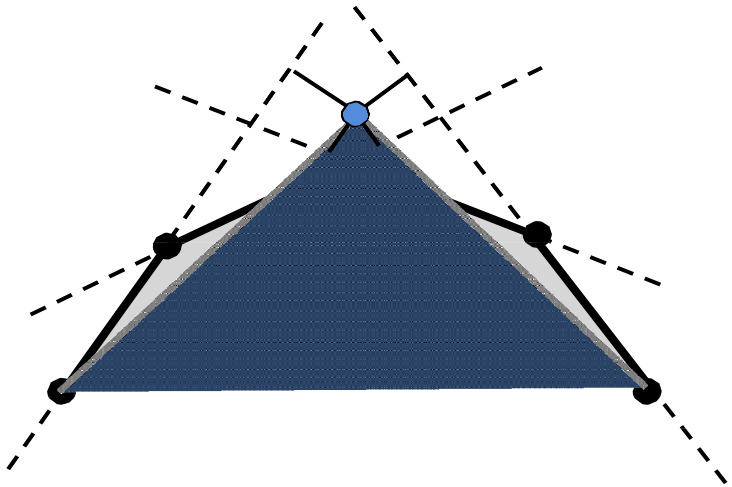
Computing a Representative



Error quadrics

Error Quadrics

- Cluster of vertices → set of triangles incident on the vertices
- Compute a new vertex position that minimizes distance to triangles planes



Error Quadrics

- Squared distance of point *p* to plane *q*: remember plane equation: $ax + by + cz + d = 0$

$$p = (x, y, z, 1)^T, \quad q = (a, b, c, d)^T$$

$$dist(q, p)^2 = (q^T p)^2 = p^T (q q^T) p =: p^T Q_q p$$

$$Q_q = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

Error Quadrics

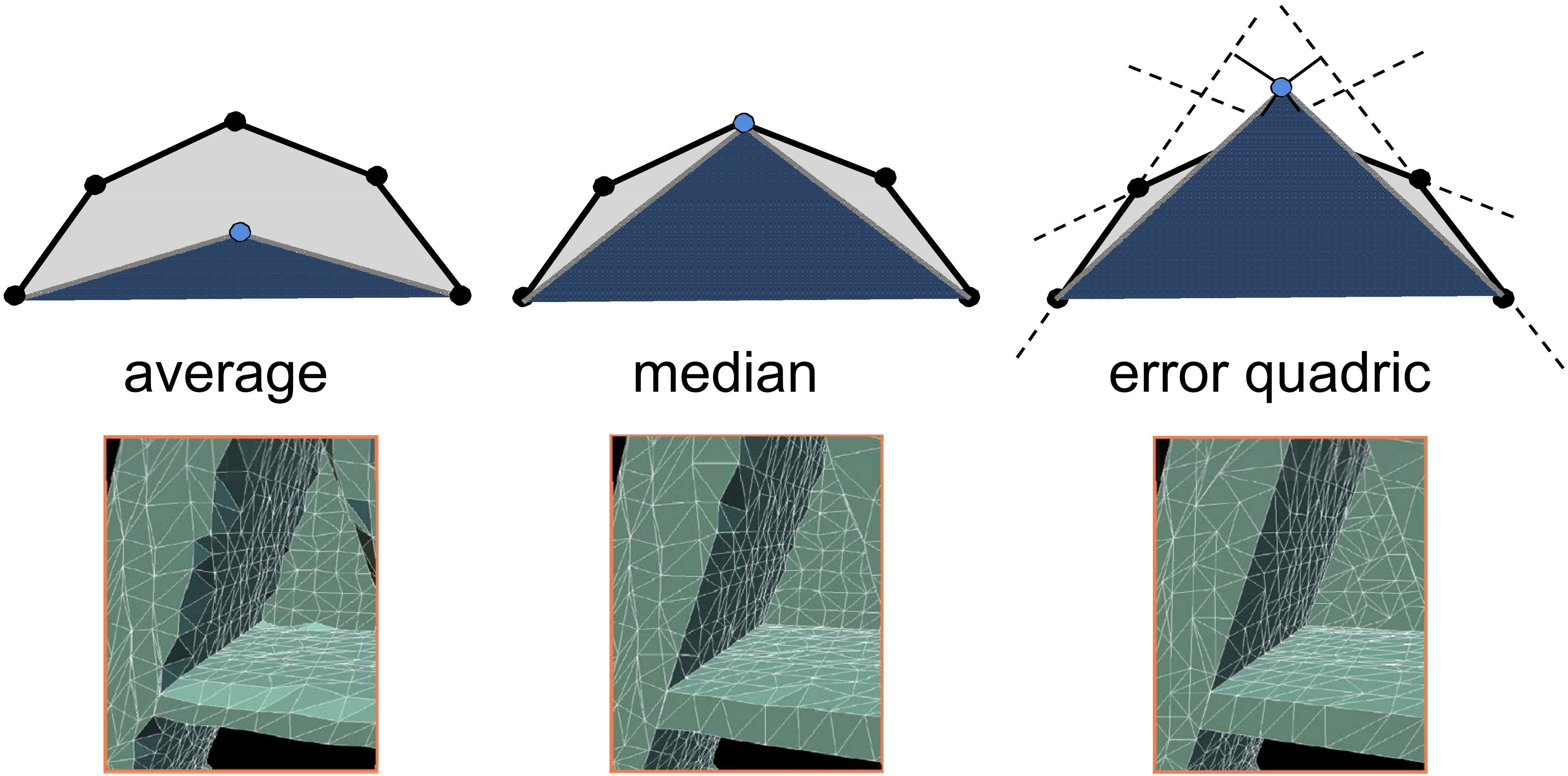
- Sum distances to planes q_i of vertex neighboring triangles:

$$\sum_i dist(q_i, p)^2 = \sum_i p^T Q_{q_i} p = p^T \left(\sum_i Q_{q_i} \right) p =: p^T Q_p p$$

- Point p^* that minimizes the error satisfies:

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} p^* = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Comparison

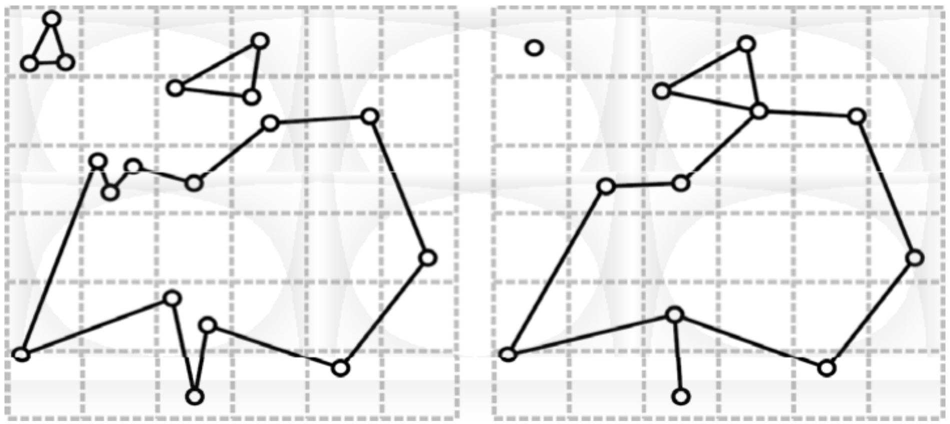


Vertex Clustering

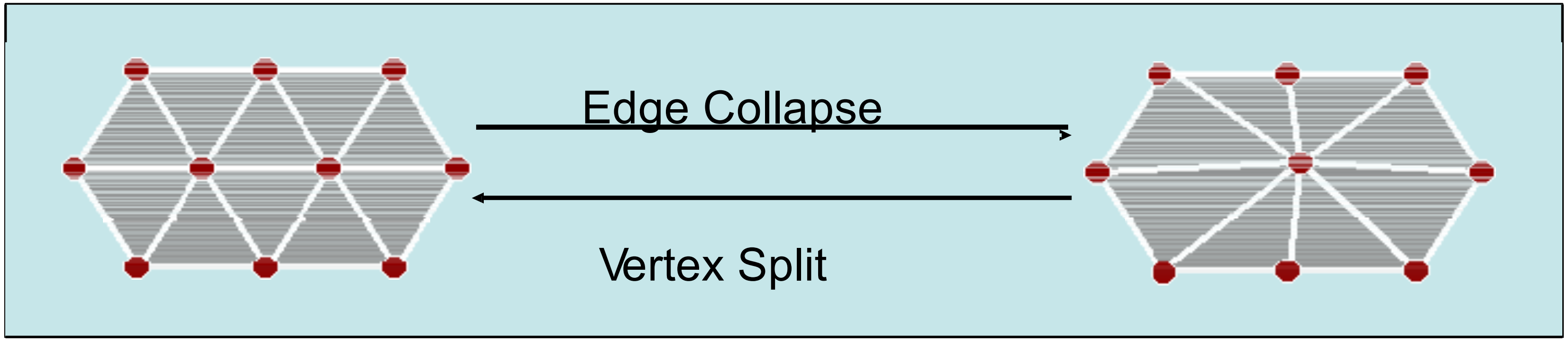
- Cluster Generation
- Computing a representative
- Mesh generation
  - Clusters $p \leftrightarrow \{p_0,\ldots,p_n\}$, $q \leftrightarrow \{q_0,\ldots,q_m\}$
  - Connect (p,q) if there was an edge (p_i,q_j)
- Topology changes

Vertex Clustering

- Cluster Generation
- Computing a representative
- Mesh generation
- Topology changes
  - If different sheets pass through one cell
  - Can be non-manifold

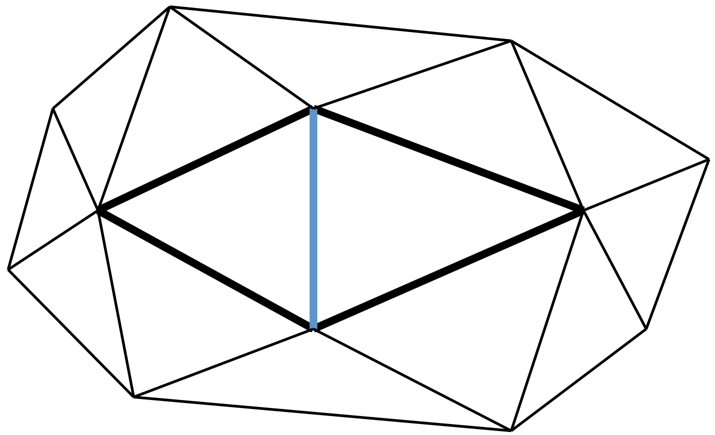


Edge Collapse (instead of vertex clustering)



- Merge two adjacent vertices
- Define new vertex position
  - Continuous degrees of freedom
  - Filter along the way

Edge Collapse

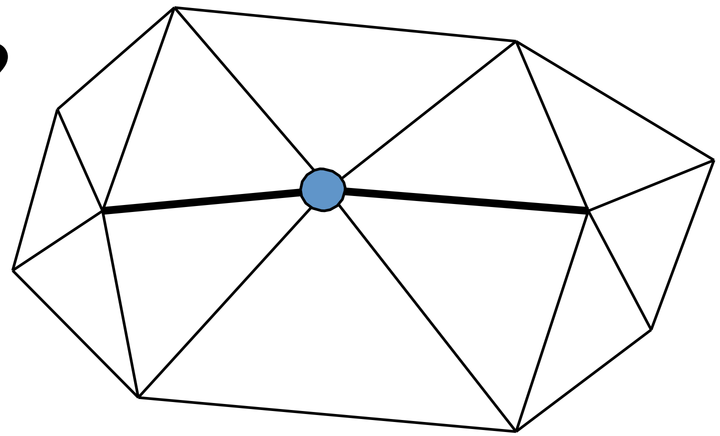


Removing an edge…

- Merges two vertices into one vertex
- Removes two faces
- Mesh still consists of triangles

Which edges should be removed first?

Where should the new vertices go?



Quadric Error Metric

Find Q for each vertex v

$$Q_{\mathrm{v}} = \Sigma\, Q_i$$

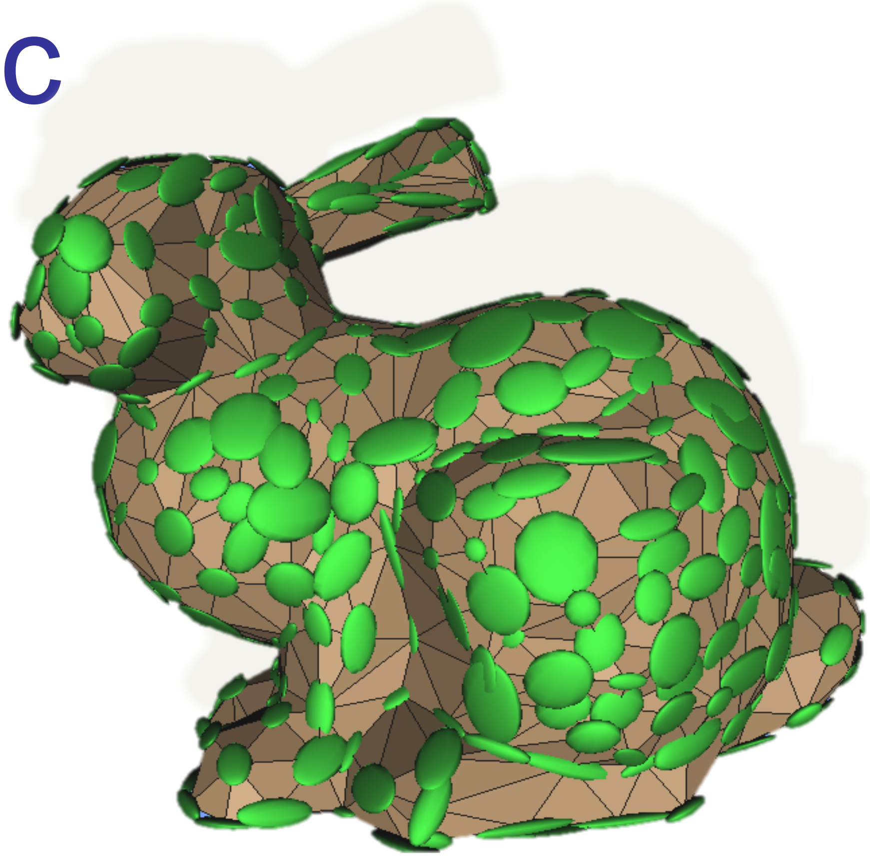
(for adjacent polygons i)

Create a priority queue of edge collapses:

Each collapse would create new vertex $\mathbf{v}_{\mathrm{new}}$

$Q_{vnew} = Q_{v1} + Q_{v2}$

Choose collapse with min $Q(\mathbf{v}_{\mathbf{new}})$



$$\mathbf{v}_{12} = -\begin{bmatrix} A^2 & AB & AC \\ AB & B^2 & BC \\ AC & BC & C^2 \end{bmatrix}^{-1} \begin{bmatrix} AD \\ BD \\ CD \end{bmatrix}$$

Examples