INSTITUTE OF VISUAL COMPUTING

TU Graz

SCIENCE
PASSION
TECHNOLOGY

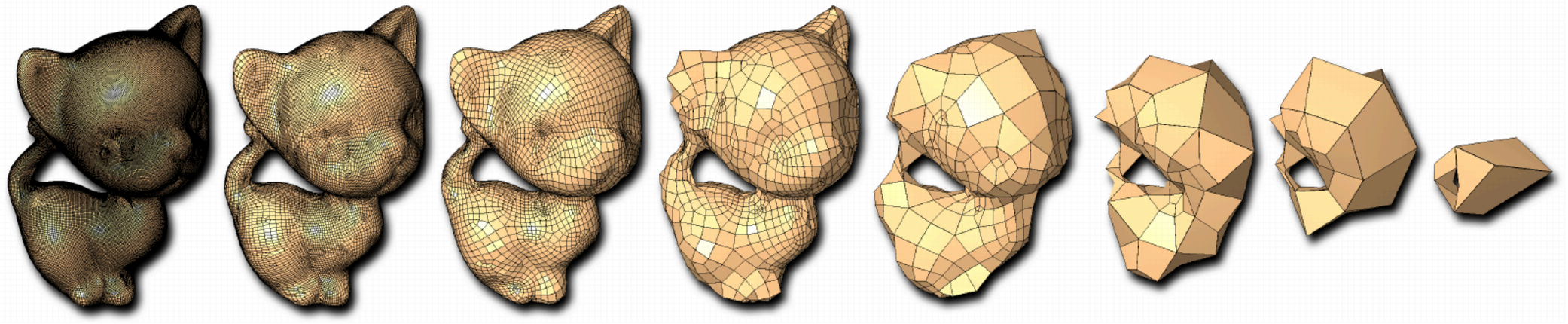
VU Fundamentals of Geometry Processing

Mesh Decimation

Julian Rakuschek

26.03.2026

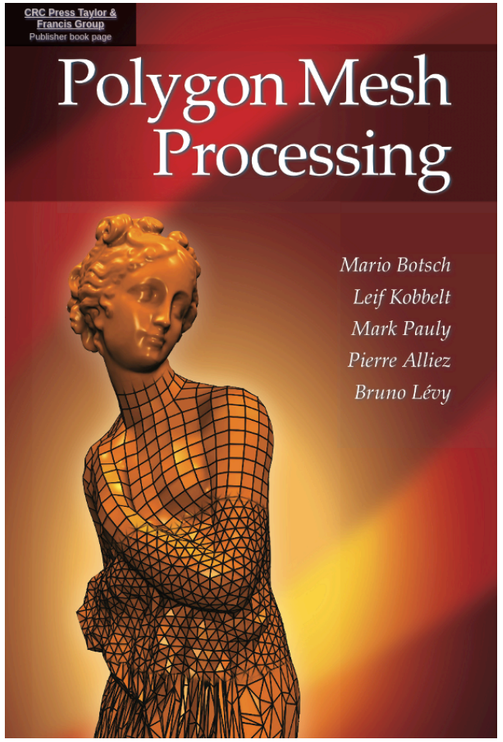
In this lecture



[Daniels, Joel et al. “Quadrilateral mesh simplification.” ACM Trans. Graph. 27 (2008): 148.]

Given a highly dense mesh how can we reduce
the amount of triangles while preserving features?

Supplemental Reading

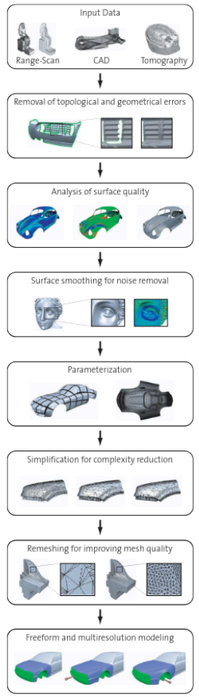


pmp-book.org

7.1 Vertex Clustering

7.2 Incremental Decimation

Acknowledgment



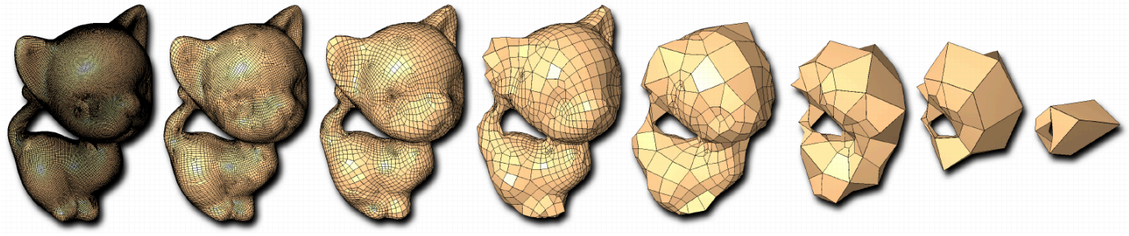
ETH APPLIED GEOMETRY GROUP

Mesh Decimation

Mark Pauly

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

SIGGRAPH2006



Mesh Decimation

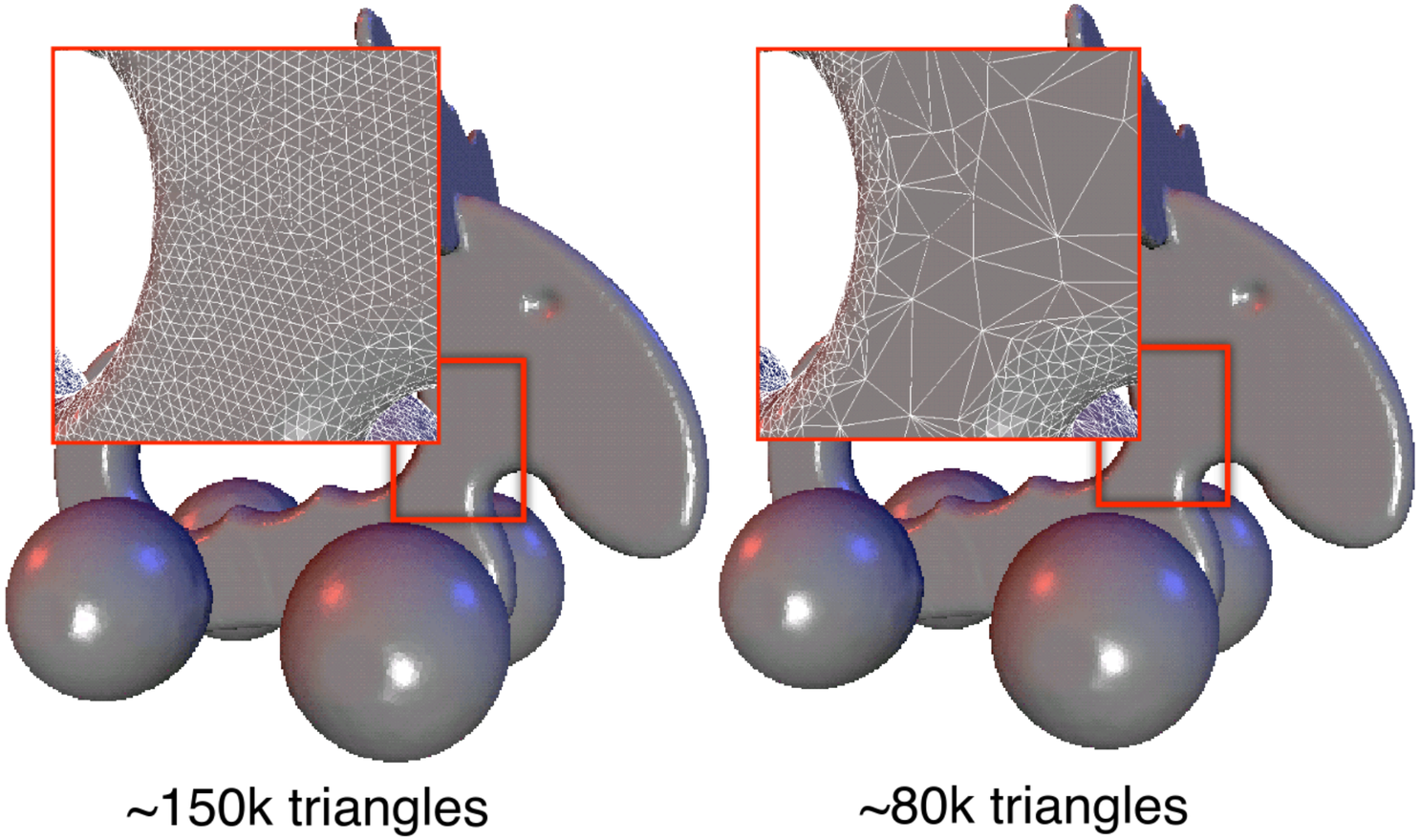
Ursula Augsdörfer

TU Graz Augsdörfer Mesh Decimation CGV

This lecture is based on a talk by Mark Pauly on Mesh Decimation
and a previous lecture by Prof.in Dr.in Ursula Augsdörfer.

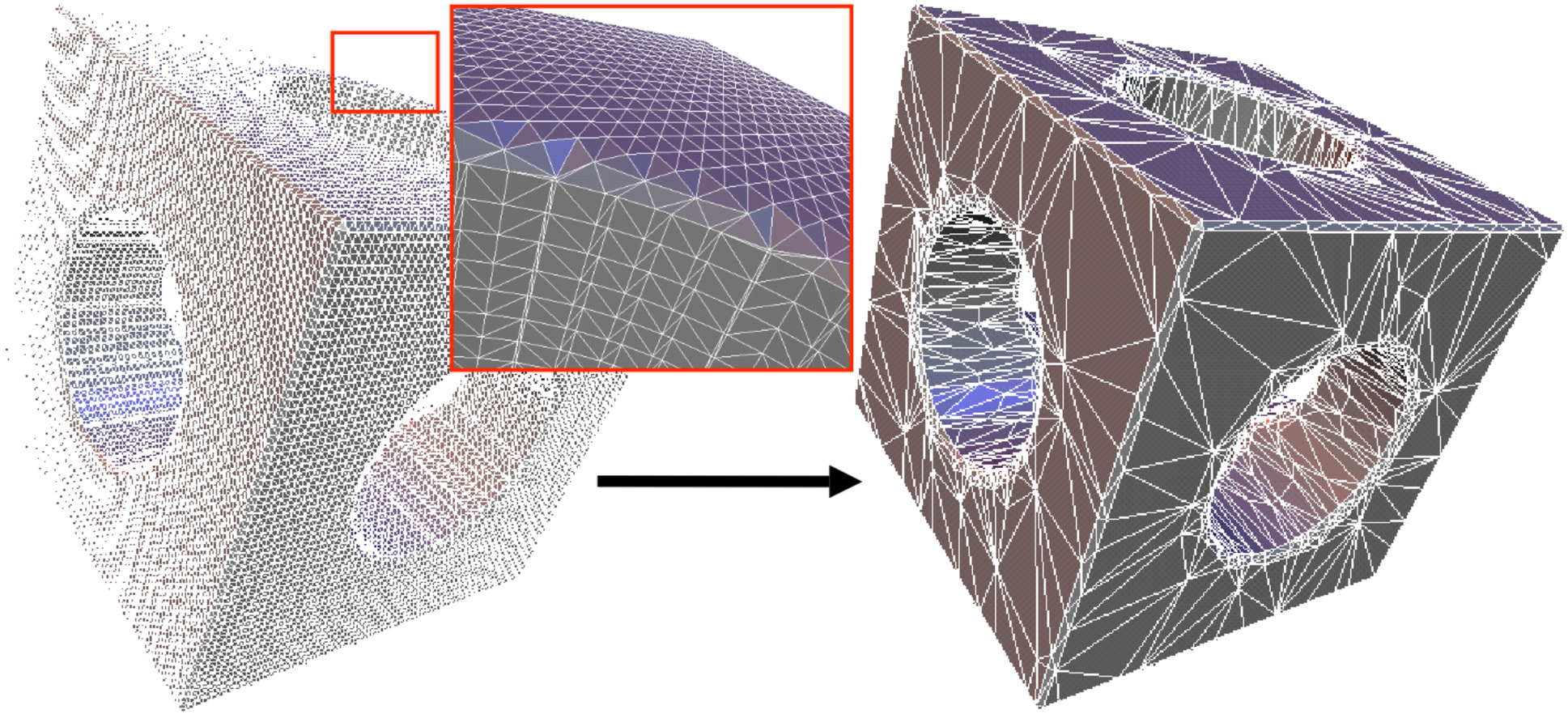
Introduction

Application: Oversampled 3D scan data



[M. Pauly, "Mesh Decimation", 2006]

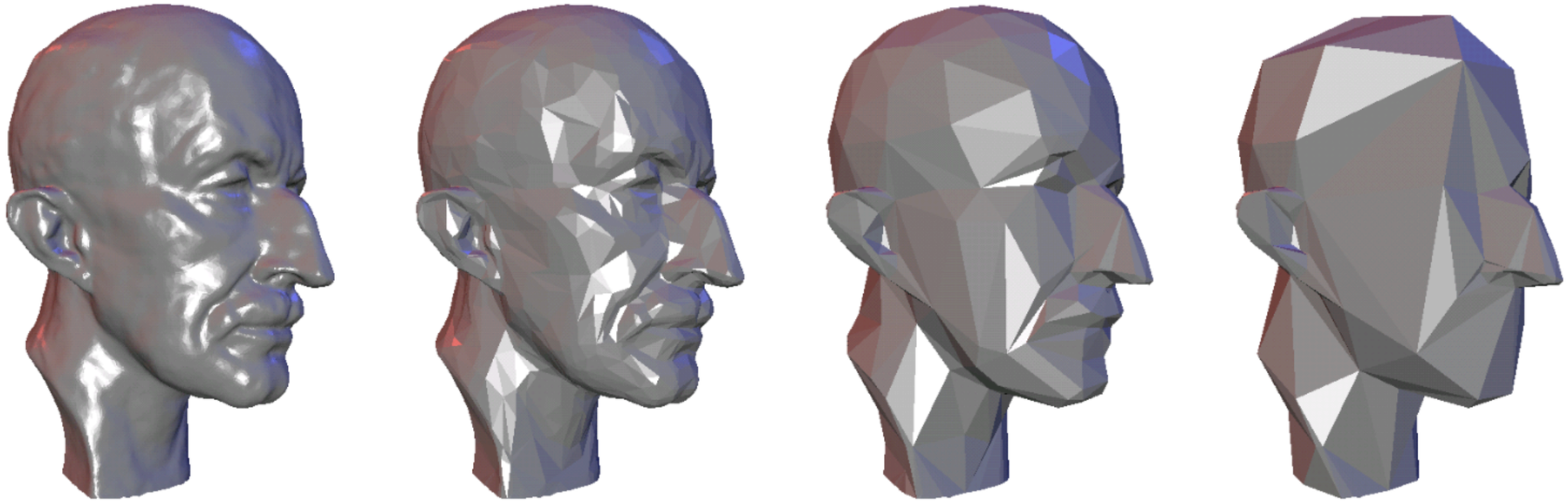
Application: Overtesselation, e.g. iso surface extraction



[M. Pauly, "Mesh Decimation", 2006]

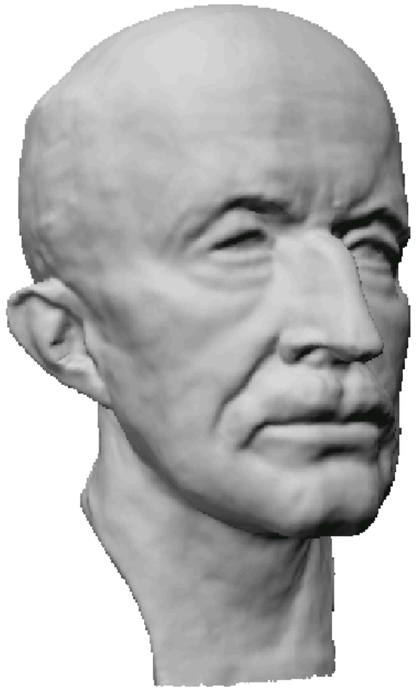
Application: Multiresolution hierarchies

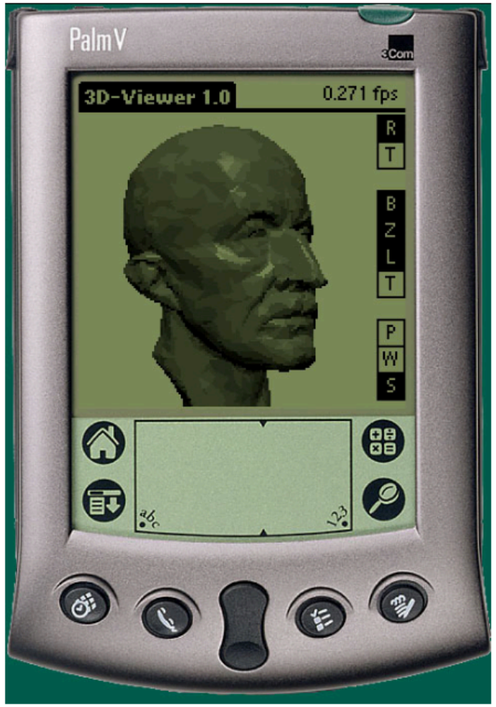
- efficient geometry processing
- level-of-detail (LOD) rendering



[M. Pauly, "Mesh Decimation", 2006]

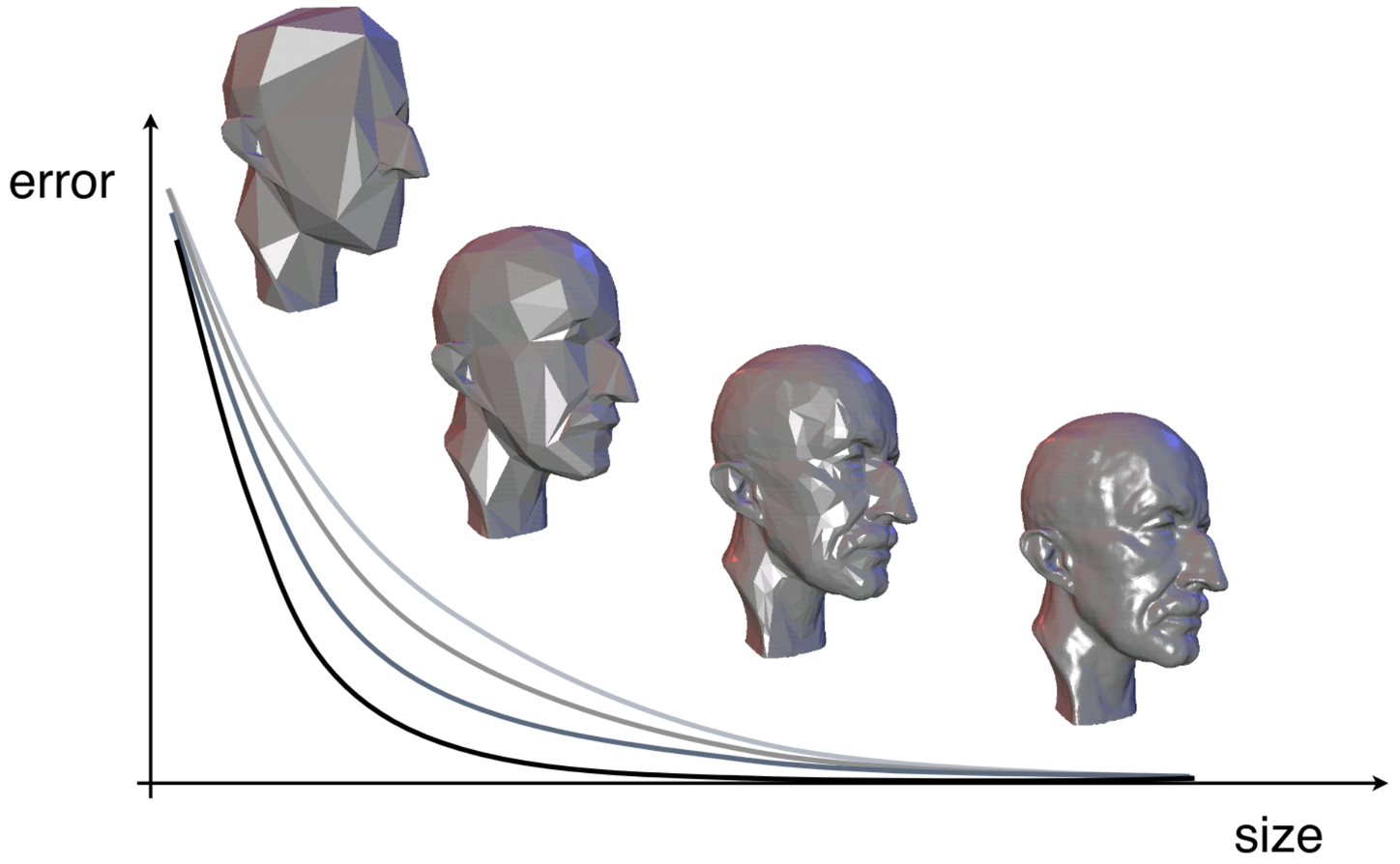
Application: Adaptation to hardware capabilities





[M. Pauly, "Mesh Decimation", 2006]

Size-Quality Tradeoff



[M. Pauly, "Mesh Decimation", 2006]

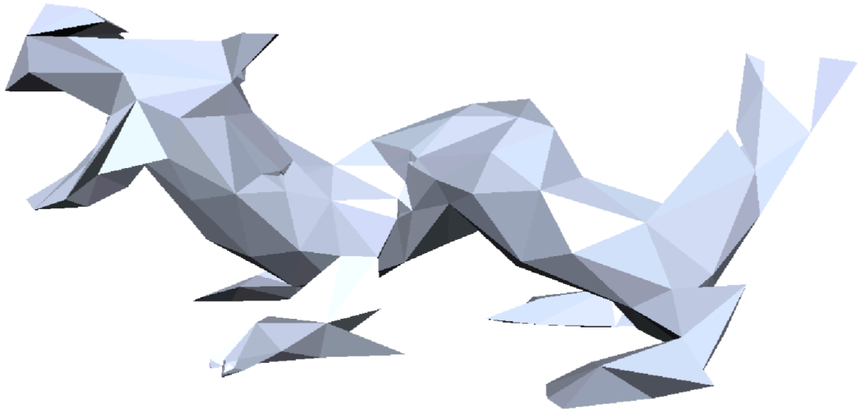
Problem Statement

Given $\mathcal{M} = (\mathcal{V}, \mathcal{F})$ find $\mathcal{M}' = (\mathcal{V}', \mathcal{F}')$ s.t.

1. $|\mathcal{V}'| = n < |\mathcal{V}|$ and $||\mathcal{M} - \mathcal{M}'||$ is minimal, or
2. $||\mathcal{M} - \mathcal{M}'|| < \epsilon$ and $|\mathcal{V}'|$ is minimal

$\mathcal{M}$

$\rightarrow$

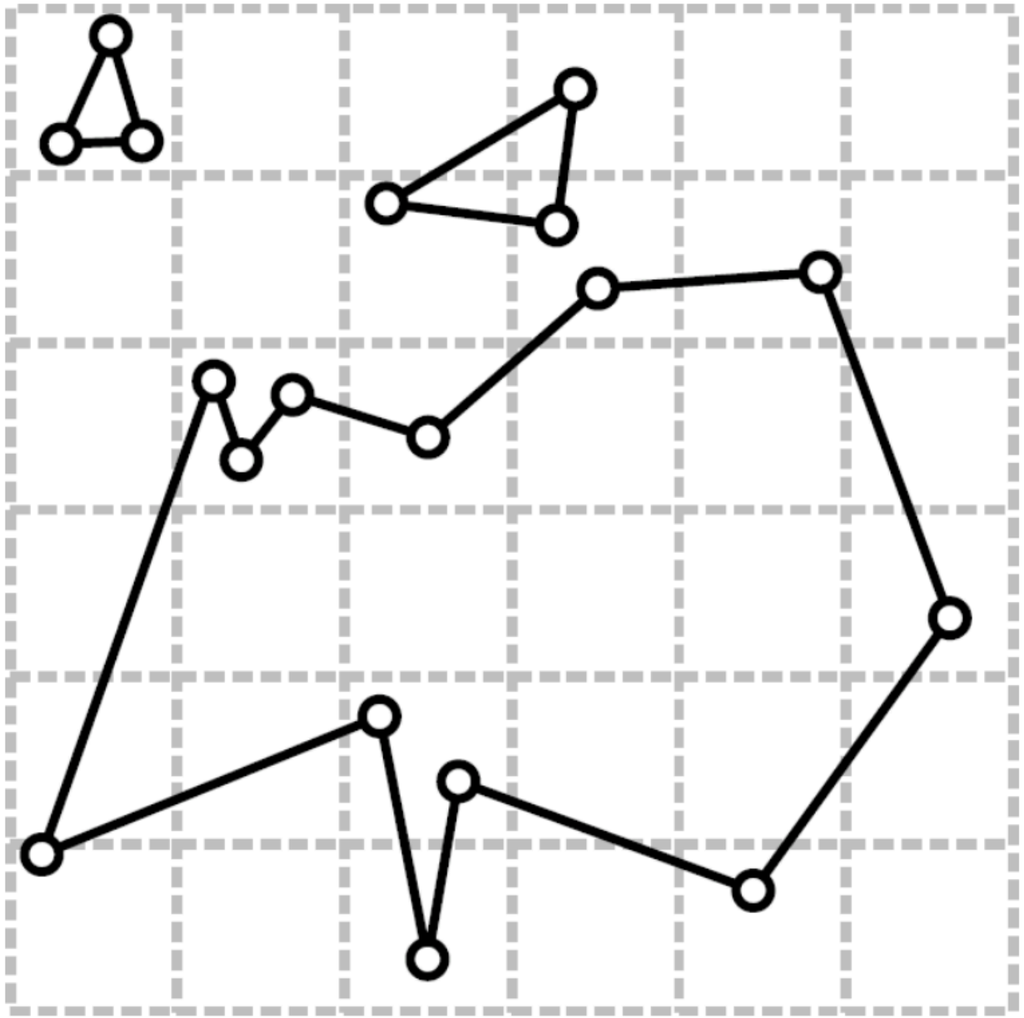


$\mathcal{M}'$

[M. Pauly, "Mesh Decimation", 2006]

Vertex Clustering

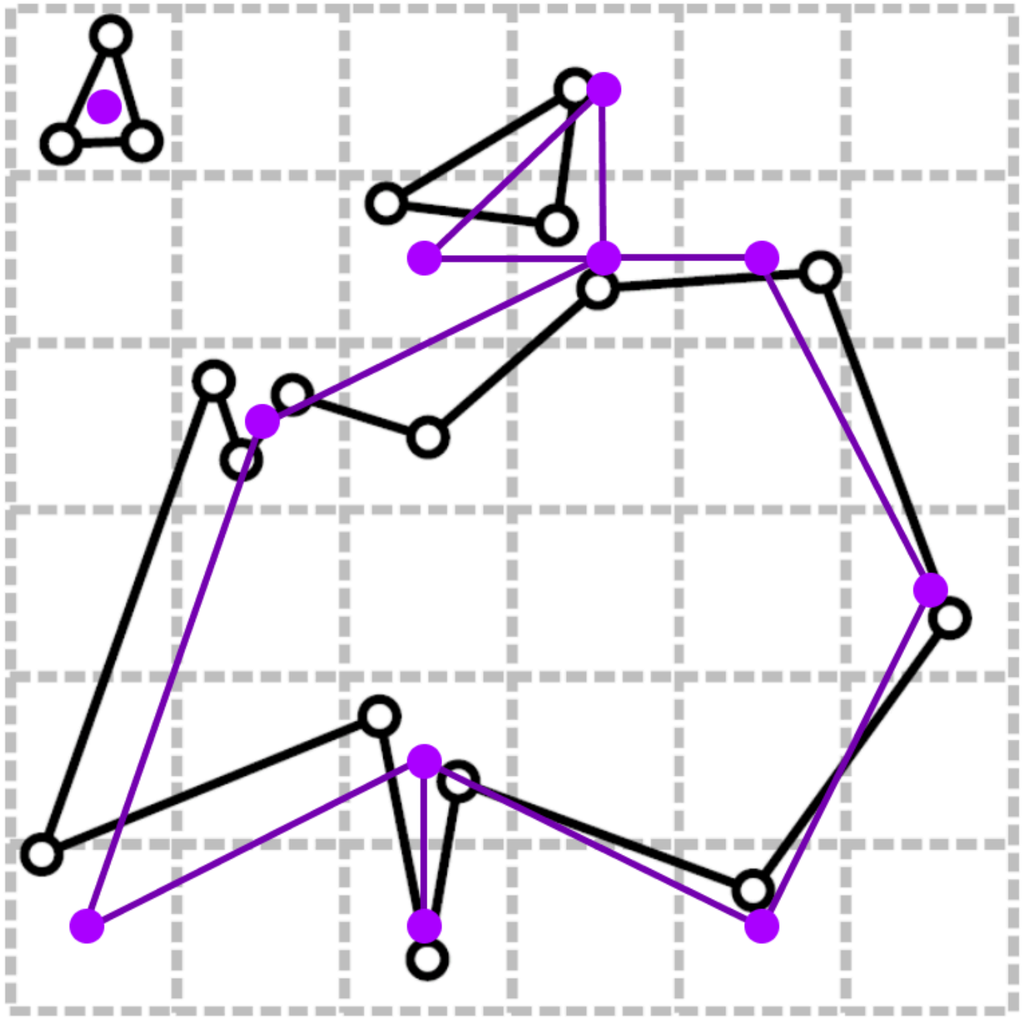
Step 1



Divide space around mesh into cells, $\epsilon \times \epsilon$, where ϵ is the *approximation tolerance*.

[M. Pauly, "Mesh Decimation", 2006]

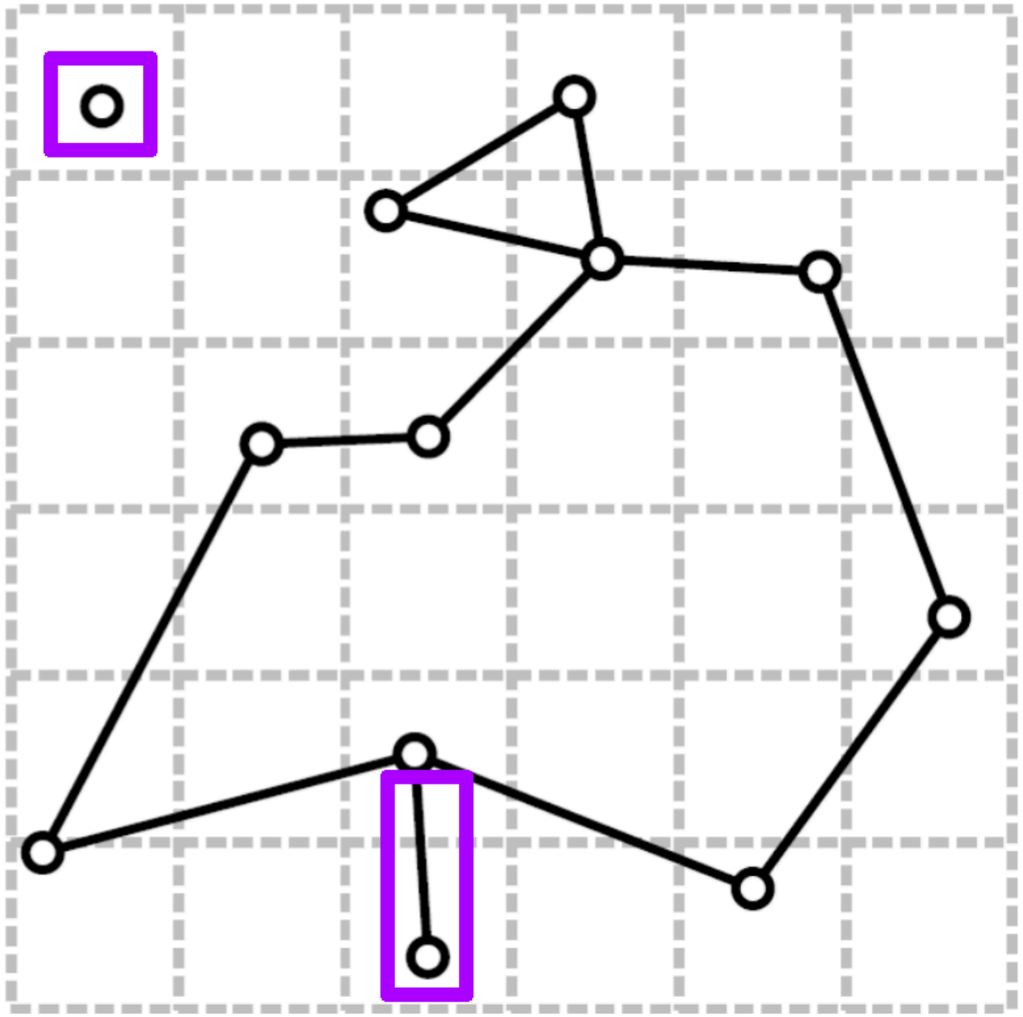
Step 2



Compute the cell representative p for each cell cluster: Assign all vertices $p_1, \ldots, p_k$, within this cell to p.

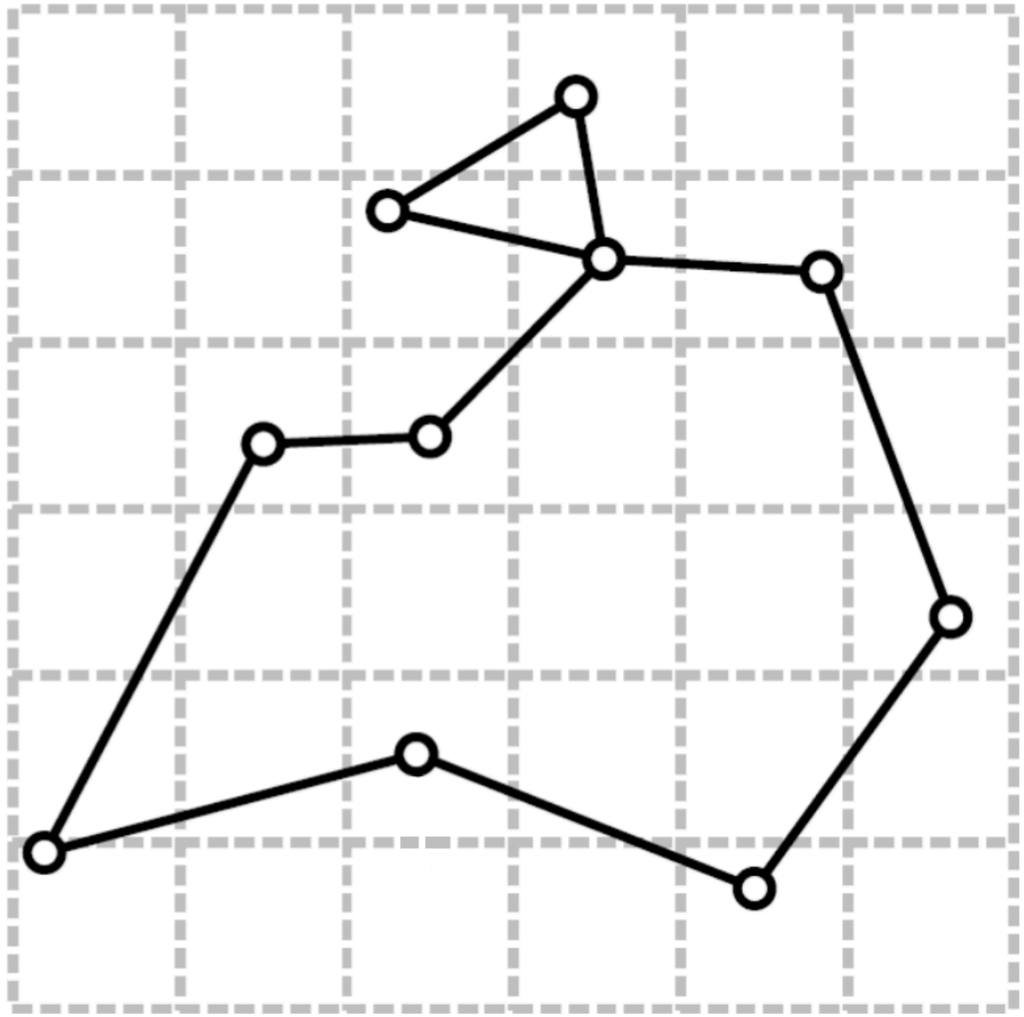
Then connect p and x if $p_i \in p_1, \ldots, p_k$ was connected to any $x_i \in x_1, \ldots, x_l$

Step 3



Many cells can become degenerate faces.

Step 4

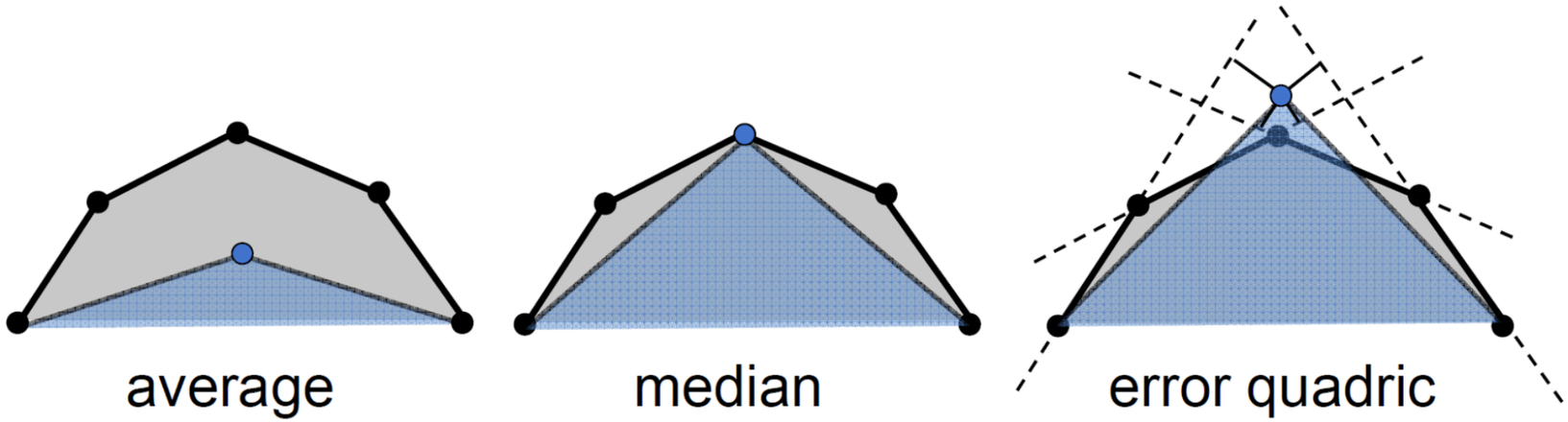


In the final step we therefore remove any degenerate faces.

Cell Representatives

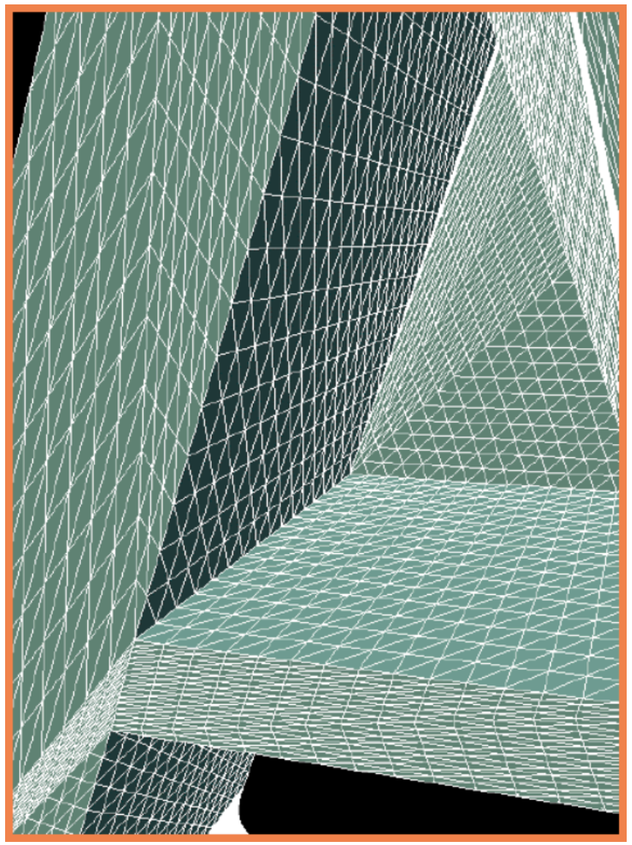
There are several options to position the new cell reprsentative p:

- Taking the **cell center**.
- Taking the **average position** of the associated vertices.
- Taking the **median** of the associated vertices.
- Finding optimal vertex position as a **least-square approximate**.

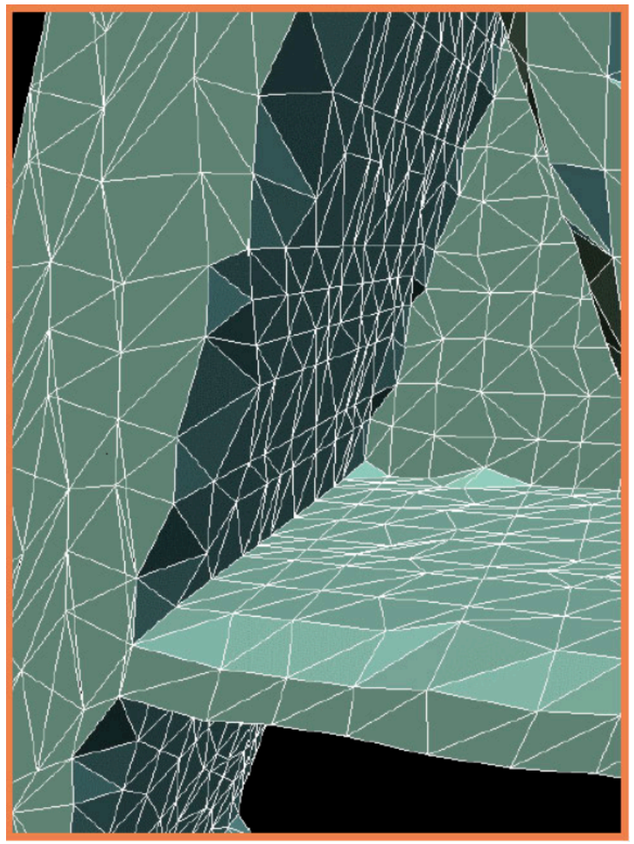


[M. Pauly, "Mesh Decimation", 2006]

When using the average position:



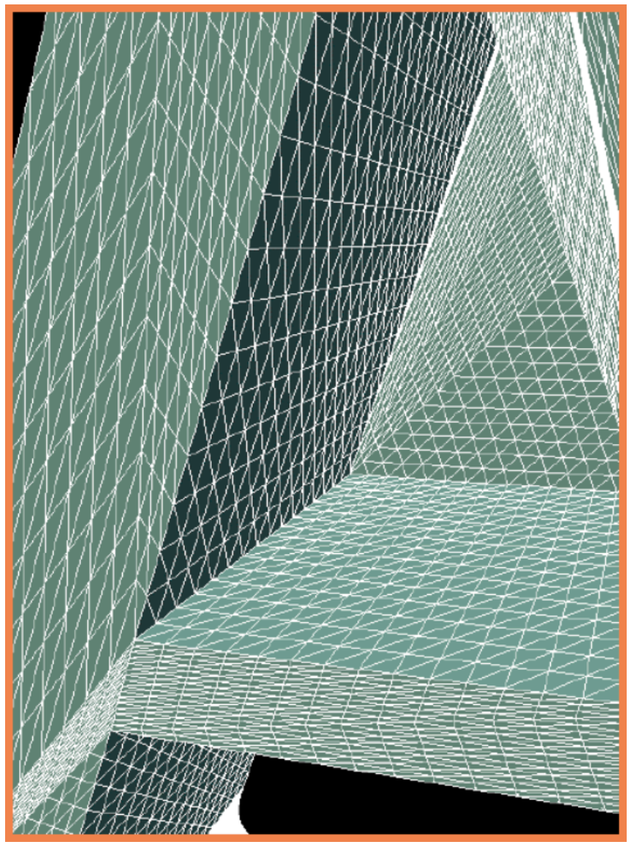
Original



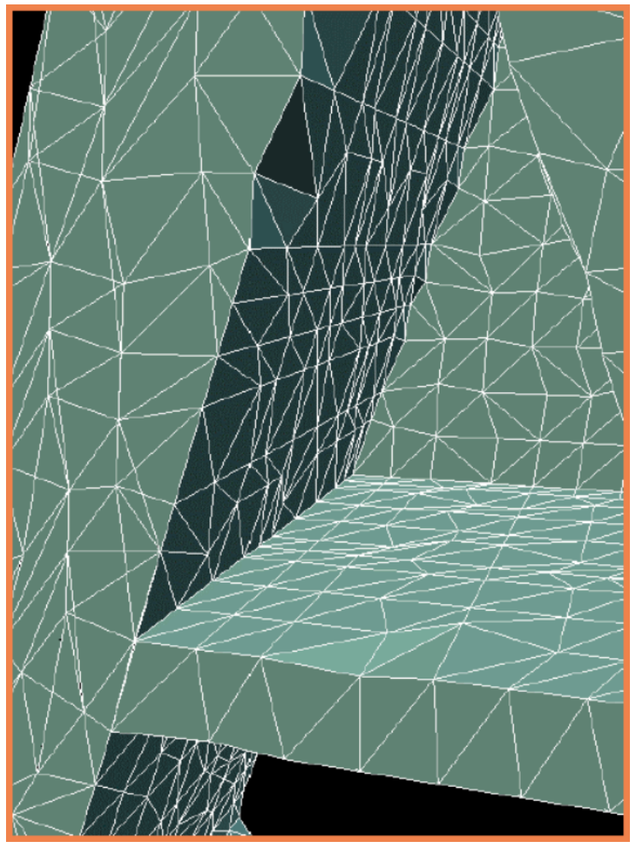
Decimated

[M. Pauly, "Mesh Decimation", 2006]

When using the median position:



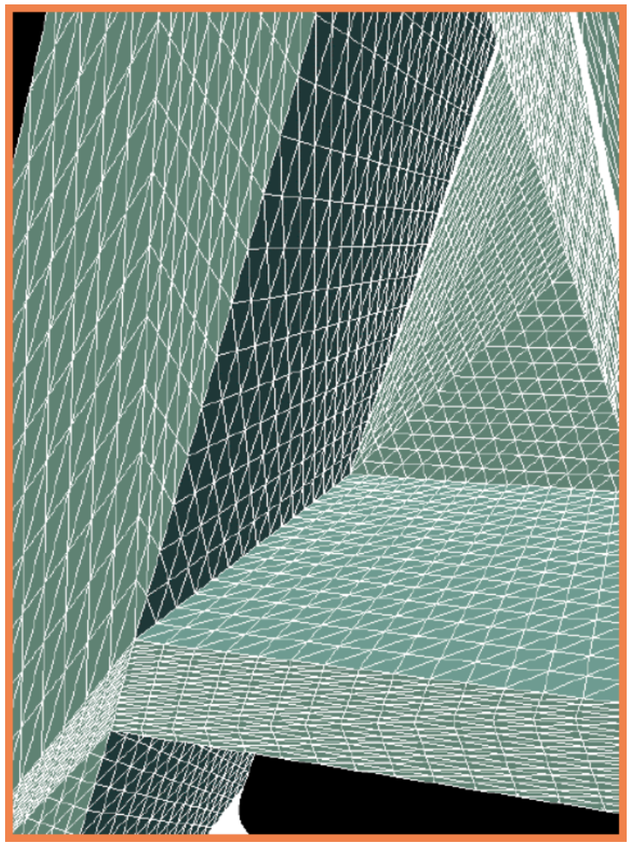
Original



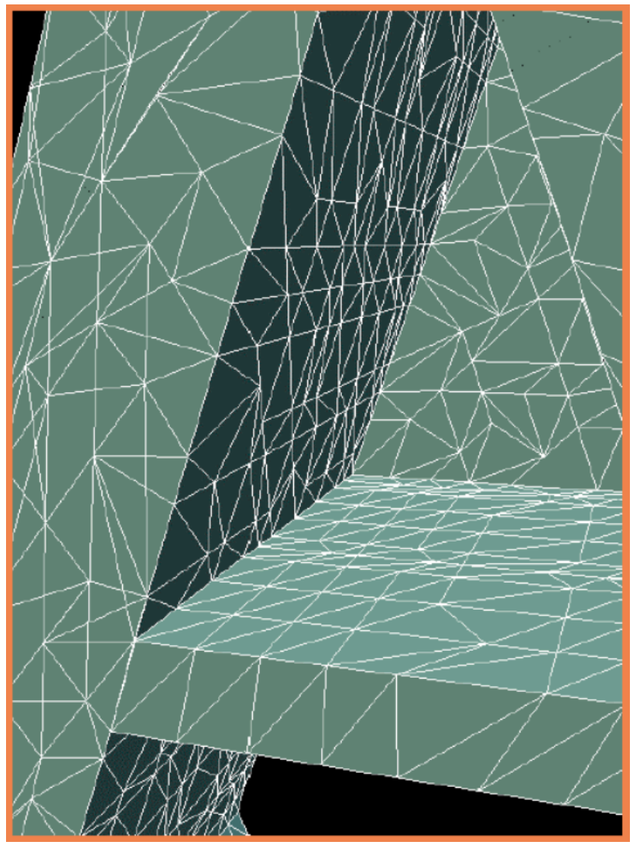
Decimated

[M. Pauly, "Mesh Decimation", 2006]

When using the error quadric position:



Original



Decimated

[M. Pauly, "Mesh Decimation", 2006]

Computing Error Quadrics (1)

Squared distance of a point p to a plane $q = (x, n)$ can be computed as

$$\text{dist}(p, q)^2 = (n^T p - n^T x)^2$$

Where x is an arbitrary vertex on the plane and n is the unit normal vector of this plane.

Using homogeneous coordinates, $p = (p, 1)$ and $n = (n, -n^T x)$ this can expressed more simply as

$$\text{dist}(p, q)^2 = (n^T p)^2 = p^T (nn^T) p =: p^T Q p$$

Computing Error Quadrics (2)

Squared distance of a point p to a plane $q = (x, n)$

$p = (x, y, z, 1)^T, n = (a, b, c, d)^T$, where $d = n^T x$

$$\text{dist}(p, q)^2 = (n^T p)^2 = p^T (n n^T) p \ =: \ p^T Q p$$

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

Computing Error Quadrics (3)

The sum of the quadratic distances to all supporting planes q_i of all triangles t_i within a cell is given by

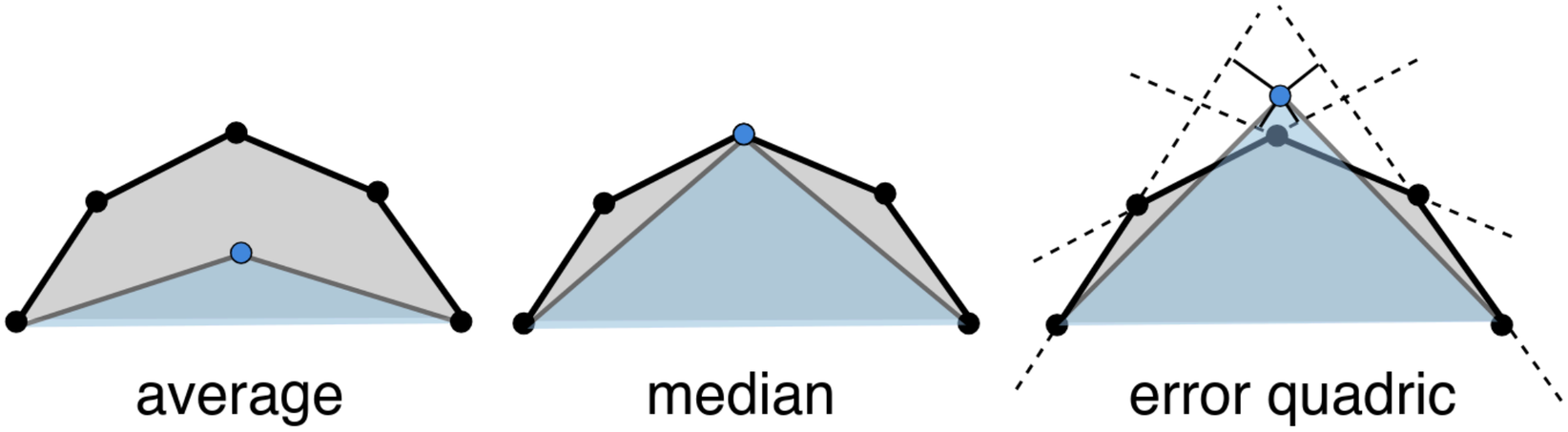
$$E(x) = \sum_{t_i} \text{dist}(p, q_i)^2 = \sum_{t_i} p^T Q_i p = p^T \left(\sum_{t_i} Q_i \right) p \;=:\; p^T Q p$$

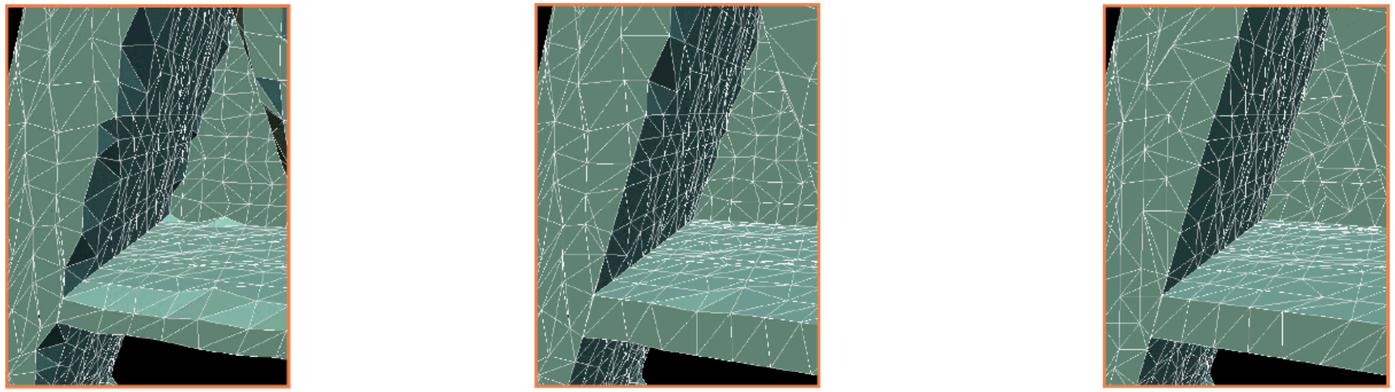
The optimal point p that minimizes the error is computed as the solution of the least-squares system:

$$\left(\sum_i n_i n_i^T \right) p = \sum_i n_i \left(n_i^T x_i \right)$$

$$Ap = n$$

Comparison



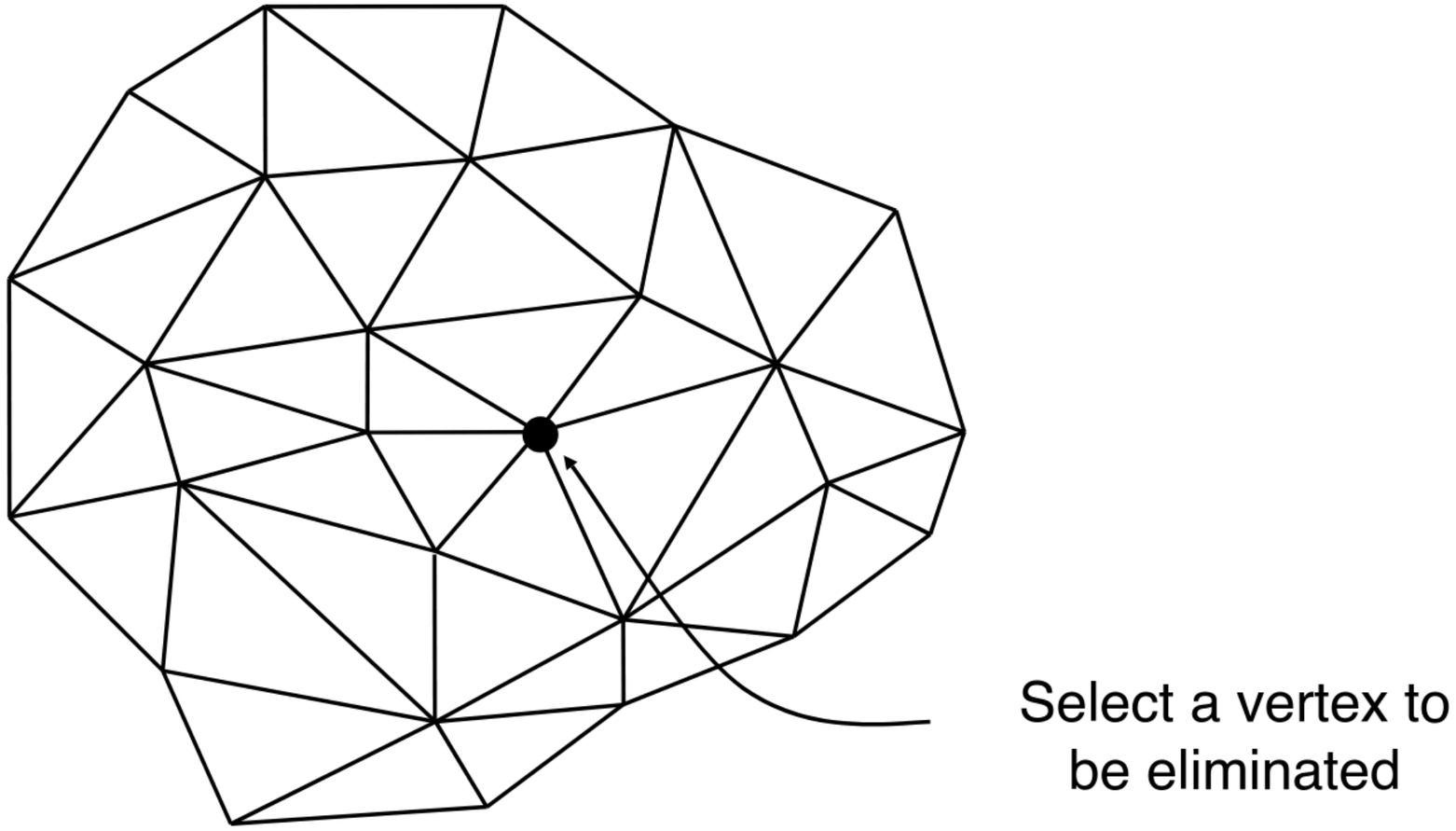


Discussion

- Quality of result depends on choice of representative
- Not always efficient
  - vertex clustering always keeps one vertex for every ϵ-cell, i.e. planar mesh could be decimated down to a triangle
- Topological changes may occur:
  - Problem: Non-manifold
  - Advantage: i.e. Sponge

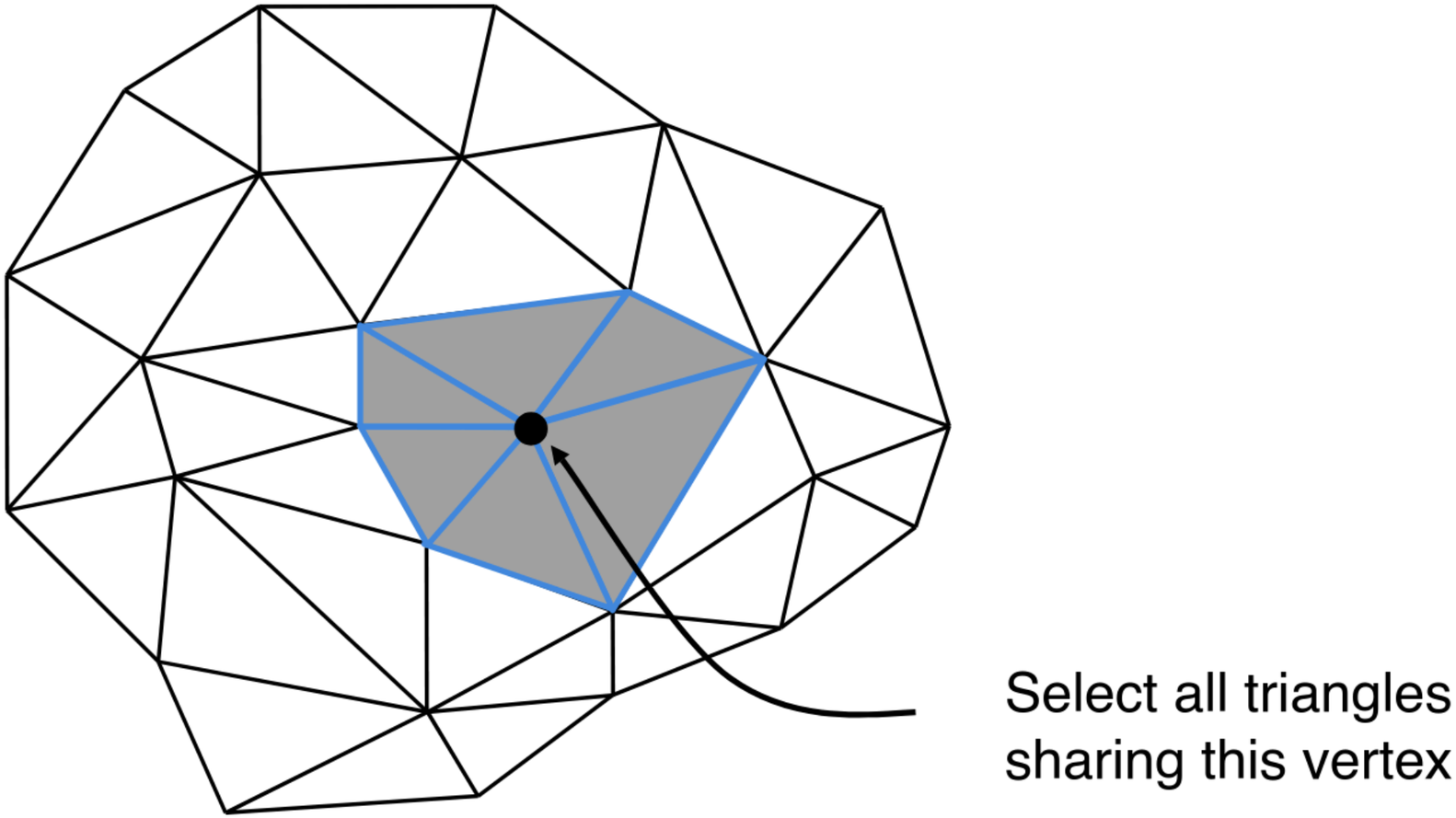
Incremental Decimation

The Basic Idea (1)



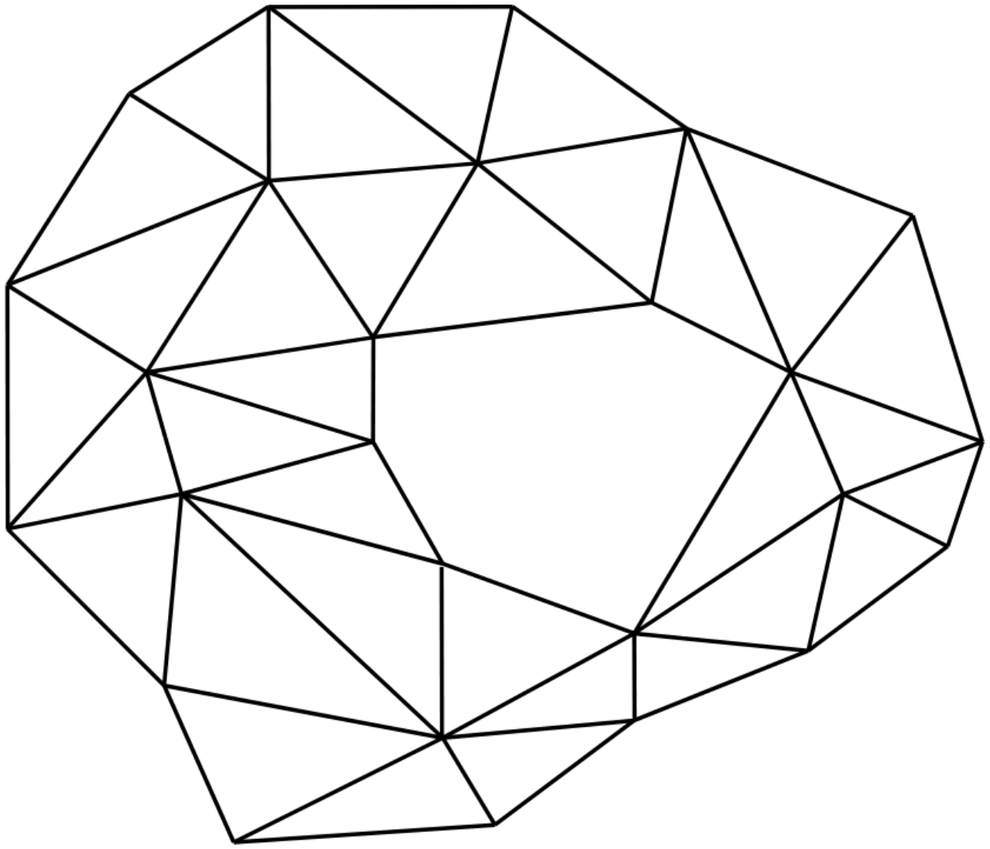
[M. Pauly, "Mesh Decimation", 2006]

The Basic Idea (2)



[M. Pauly, "Mesh Decimation", 2006]

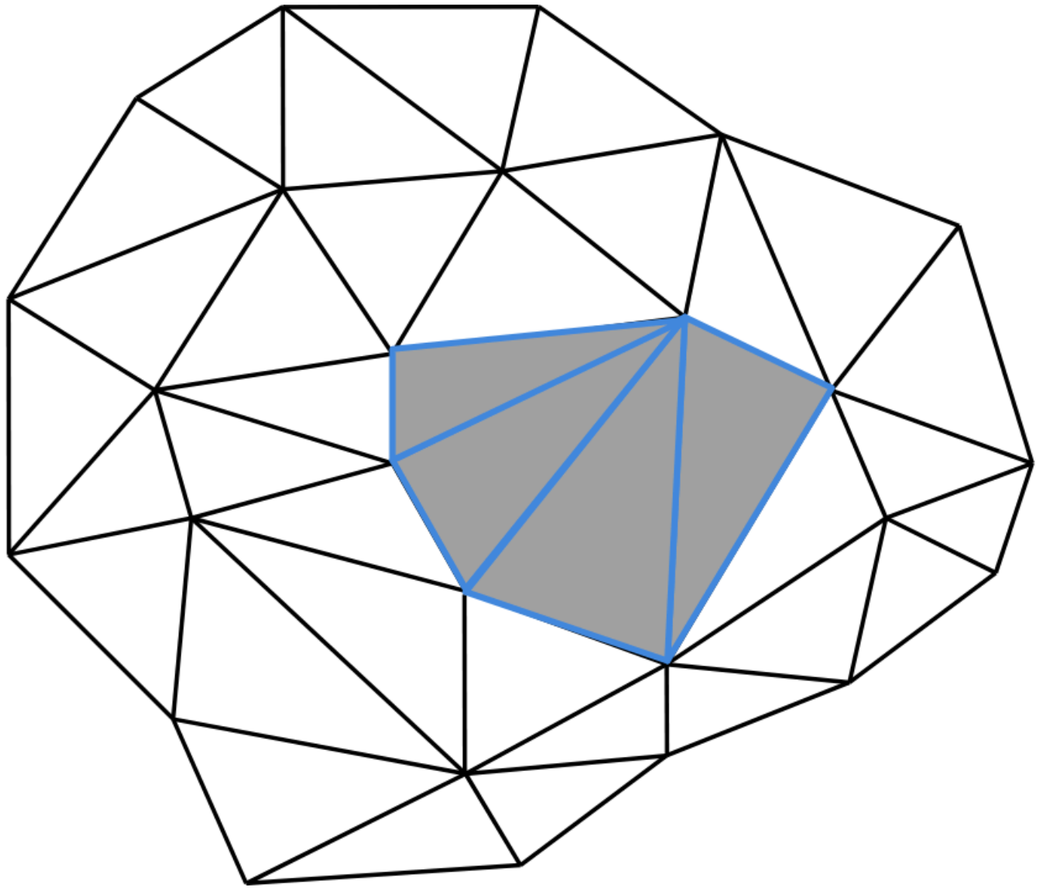
The Basic Idea (3)



Remove the selected triangles, creating the hole

[M. Pauly, "Mesh Decimation", 2006]

The Basic Idea (4)



Fill the hole
with triangles

[M. Pauly, "Mesh Decimation", 2006]

The Basic Idea (5)

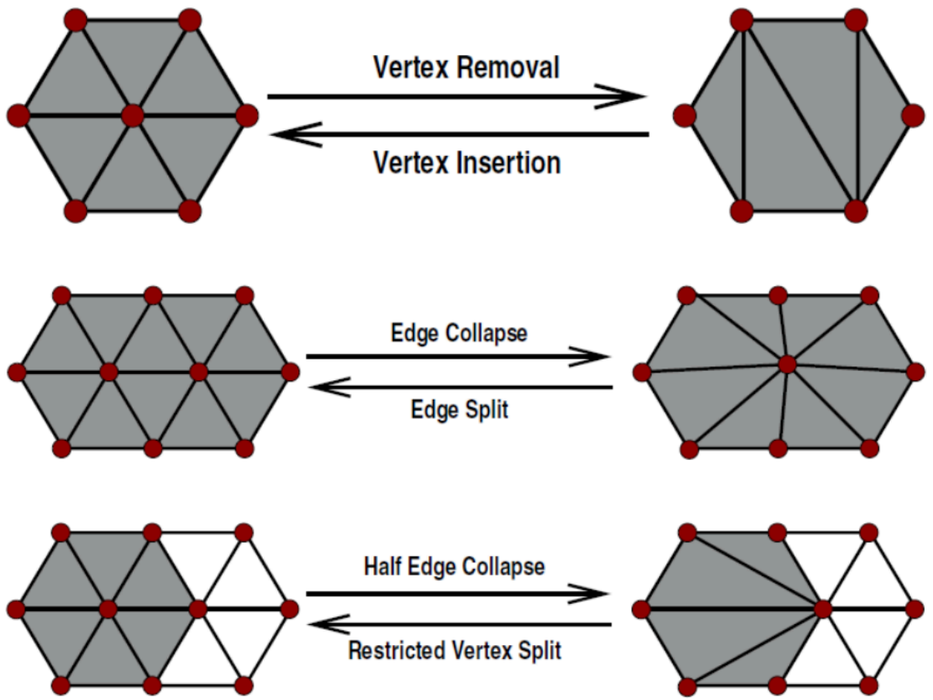
Remove one vertex at a time as long as

$$||\mathcal{M} - \mathcal{M}'|| < \epsilon$$

In each step the best candidate for removal is determined.

- Binary: Either we remove it or keep it (similar to the convexity check in the assignment).
- Continuous: We compute a measure to see how much effect the removal has on the resulting mesh.

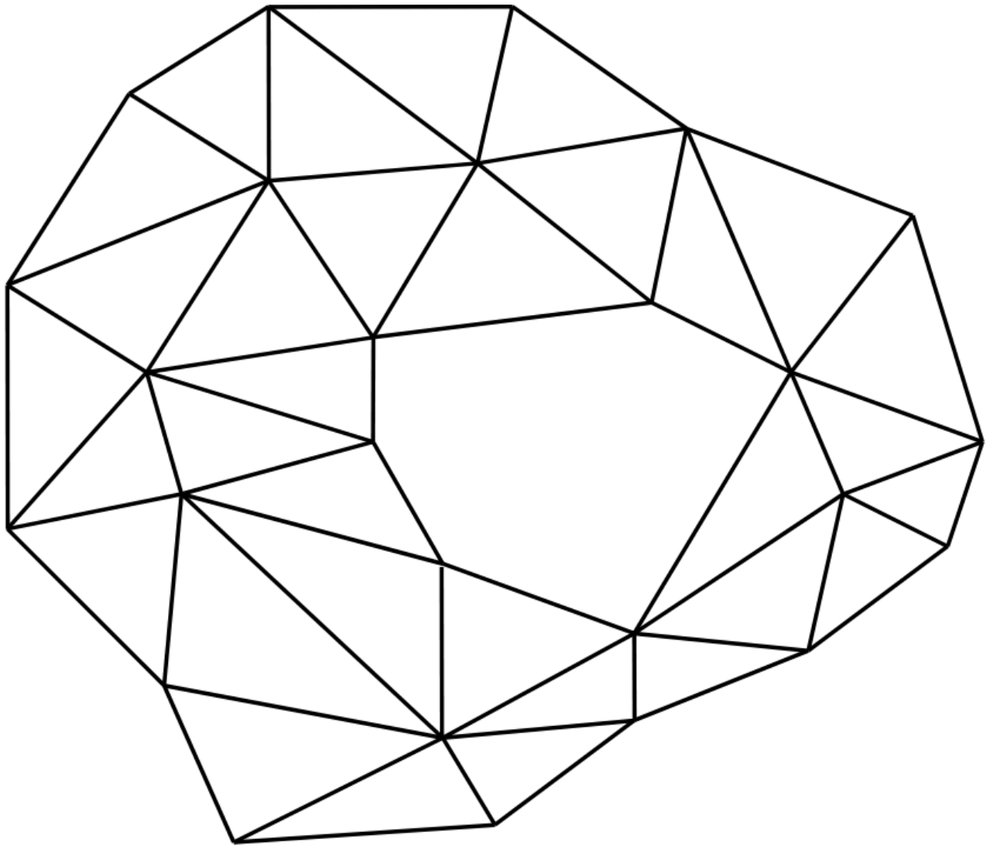
Decimation Operators



Each of the listed decimation operators is reversible.

[Botsch, Mario et al. “Polygon Mesh Processing.” (2010).]

Vertex Removal (1)



Remove the selected triangles, creating the hole

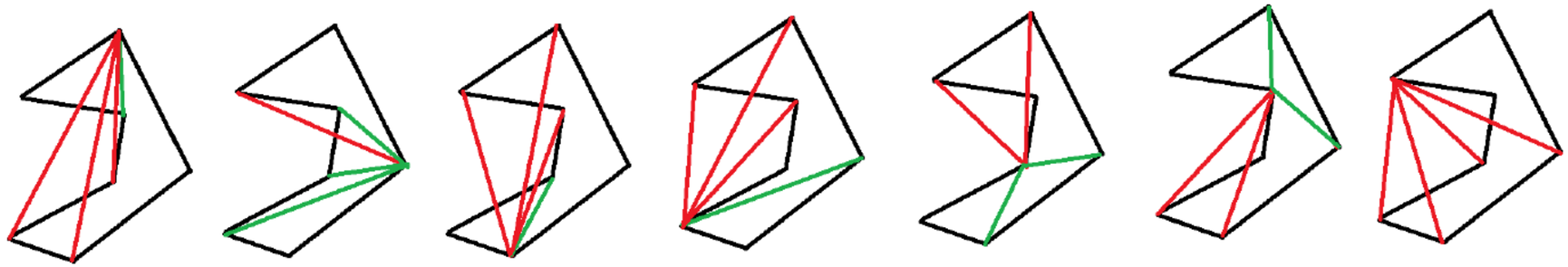
Coming back to this: How can we triangulate the hole?

Vertex Removal (2)

We strongly prefer **convex** holes.

In a planar **convex** polygon, all internal angles are at most 180°. Convex polygons can be easily triangulated from any vertex.

If a polygon is **non-convex**, that is there are internal angles > 180°, then extra care has to be taken to avoid creating overlaps and self-intersections.



[U. Augsdörfer, "Mesh Decimation", 2020]

green edges are safe, red edges cause problems

Vertex Removal (3)

Checking 3D polygons for convexity:

The m-sided hole which appears in a mesh by removing a vertex usually does not lie exactly in a 2D plane, so the notion of convexity does not make sense. Simple solution: Analyse 2D image of 3D hole.

We need to find a direction from which we look at the hole. For this we compute the average normal vector of all adjacent faces of a vertex.

$$n_{avg} = \frac{\sum_i A_i n_i}{\sum_i A_i}$$

Where A_i is the area of each adjacent face with n_i being the normal. This is an area-weighted average.

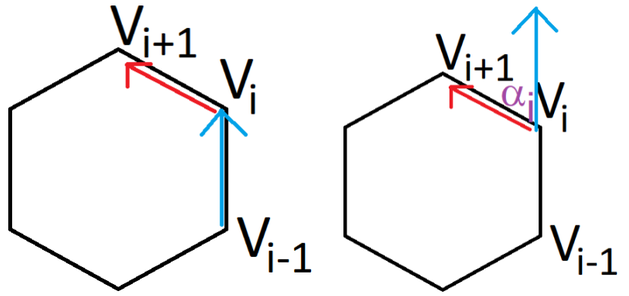
Vertex Removal (4)

Suppose the cyclic ordering of vertices $v_0, v_1, \ldots, v_{m-1}$ of a planar m-gon is such that the polygon is traversed counter-clockwise.

Then the internal angle α_i is computed from

$$\sin \alpha_i = \sin(180^\circ - \alpha_i) = \left\langle \left(\left(\frac{v_i - v_{i-i}}{|v_i - v_{i-i}|} \right) \times \left(\frac{v_{i+1} - v_i}{|v_{i+1} - v_i|} \right) \right), \frac{n}{|n|} \right\rangle$$

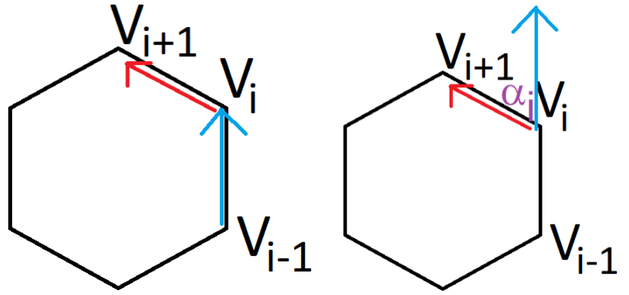
where the normal vector n is pointing towards the eye of the viewer.



[U. Augsdörfer, "Mesh Decimation", 2020]

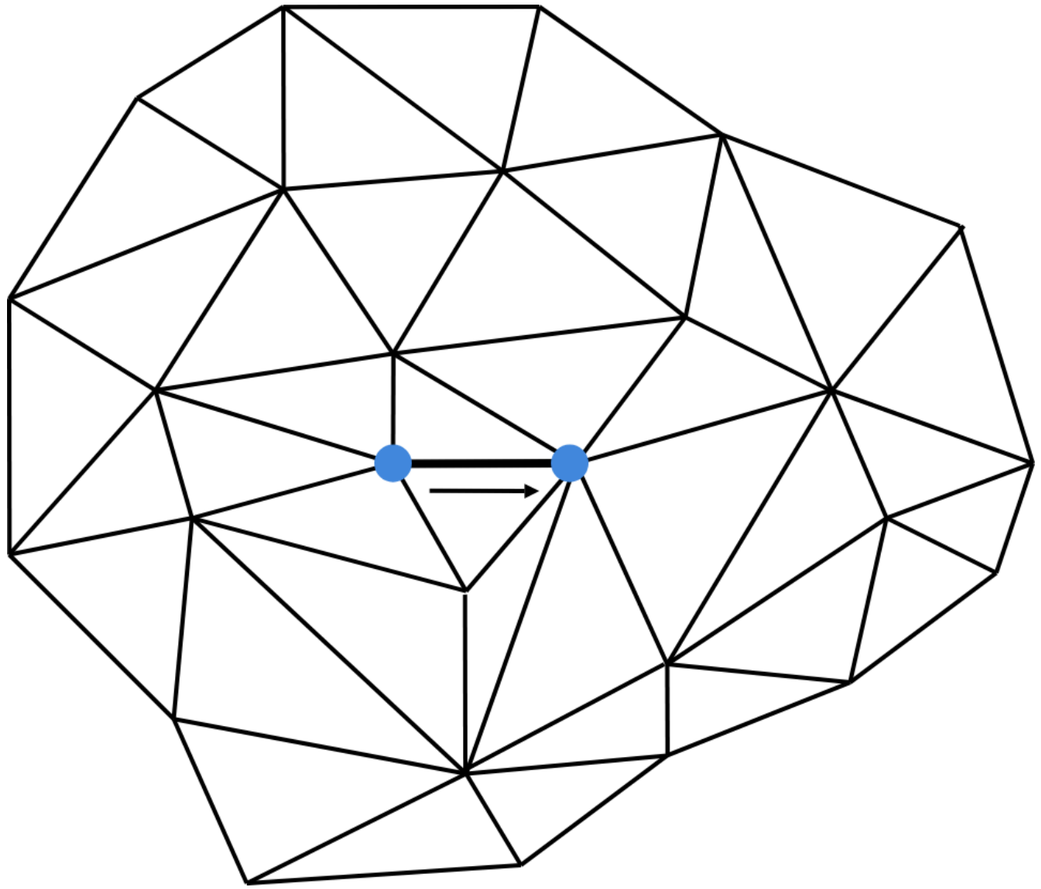
Vertex Removal (5)

- The cross product is a vector orthogonal to the successive edges $v_{i-1}v_i$ and v_iv_{i+1}, and its length coincides with $\left|\sin\alpha_i\right|$. The dot product with the normal vector recovers the true value of $\sin\alpha_i$.
- The vector $\vec{n}$ is likewise orthogonal to these two edges.
- The cross product points in the same direction as the normal vector if the vertex is convex, and it points in the other direction if the vertex is concave.



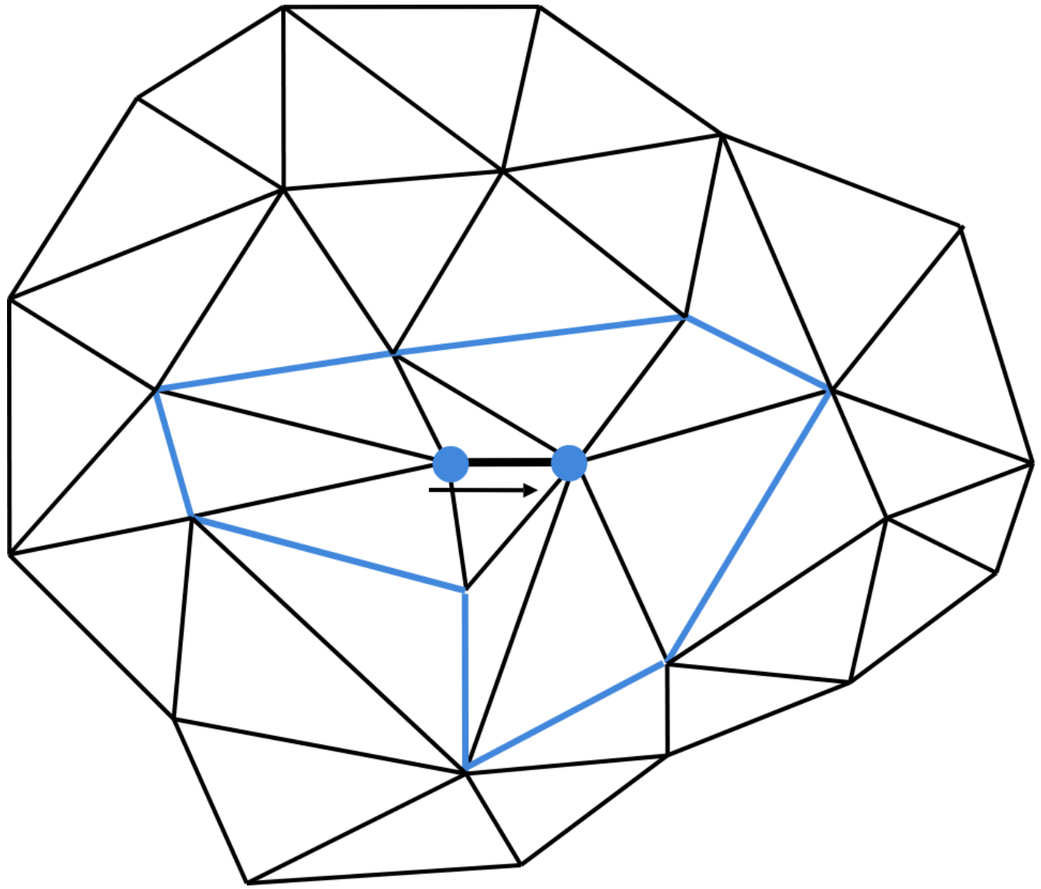
[U. Augsdörfer, "Mesh Decimation", 2020]

Halfedge Collapse (1)



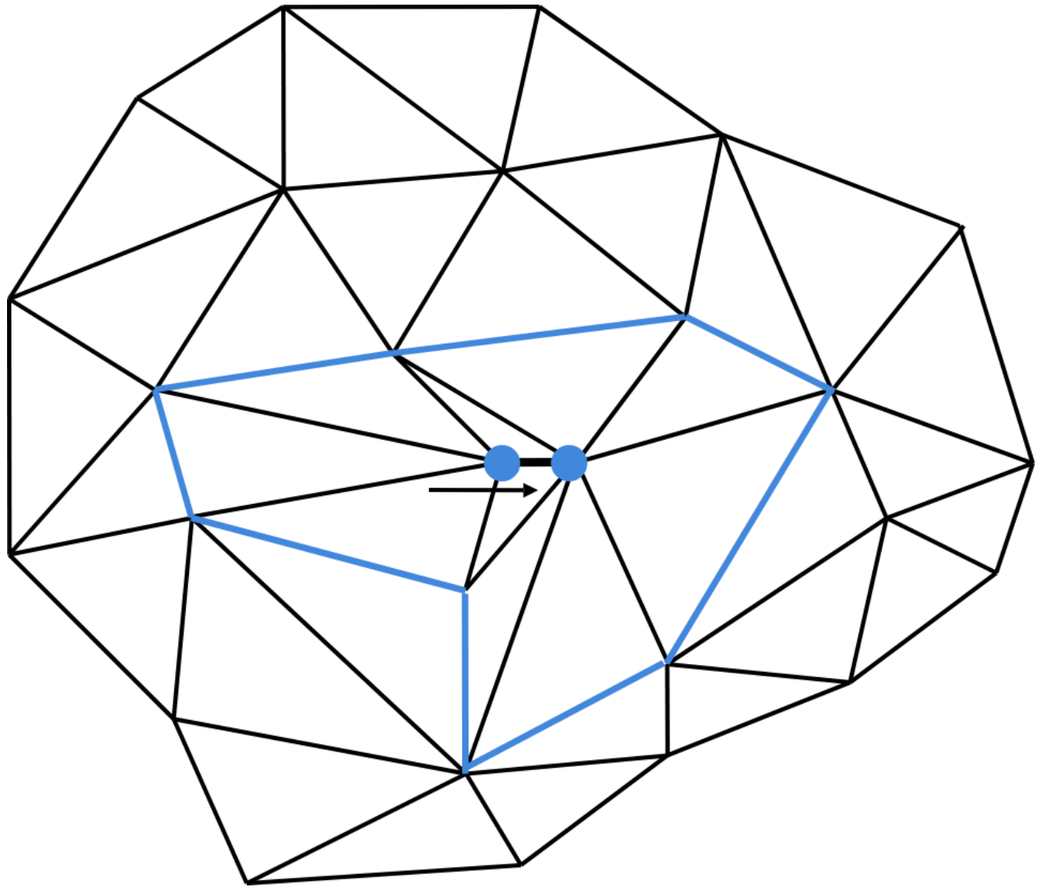
[M. Pauly, "Mesh Decimation", 2006]

Halfedge Collapse (2)



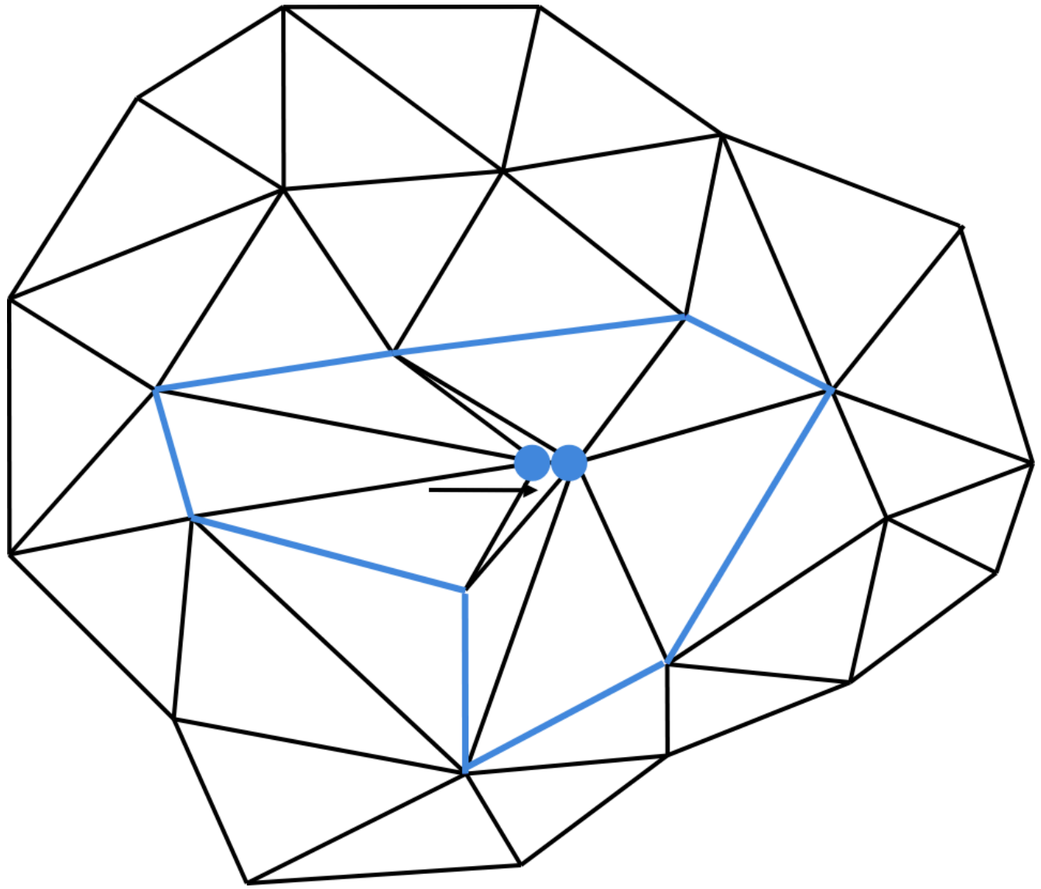
[M. Pauly, "Mesh Decimation", 2006]

Halfedge Collapse (3)



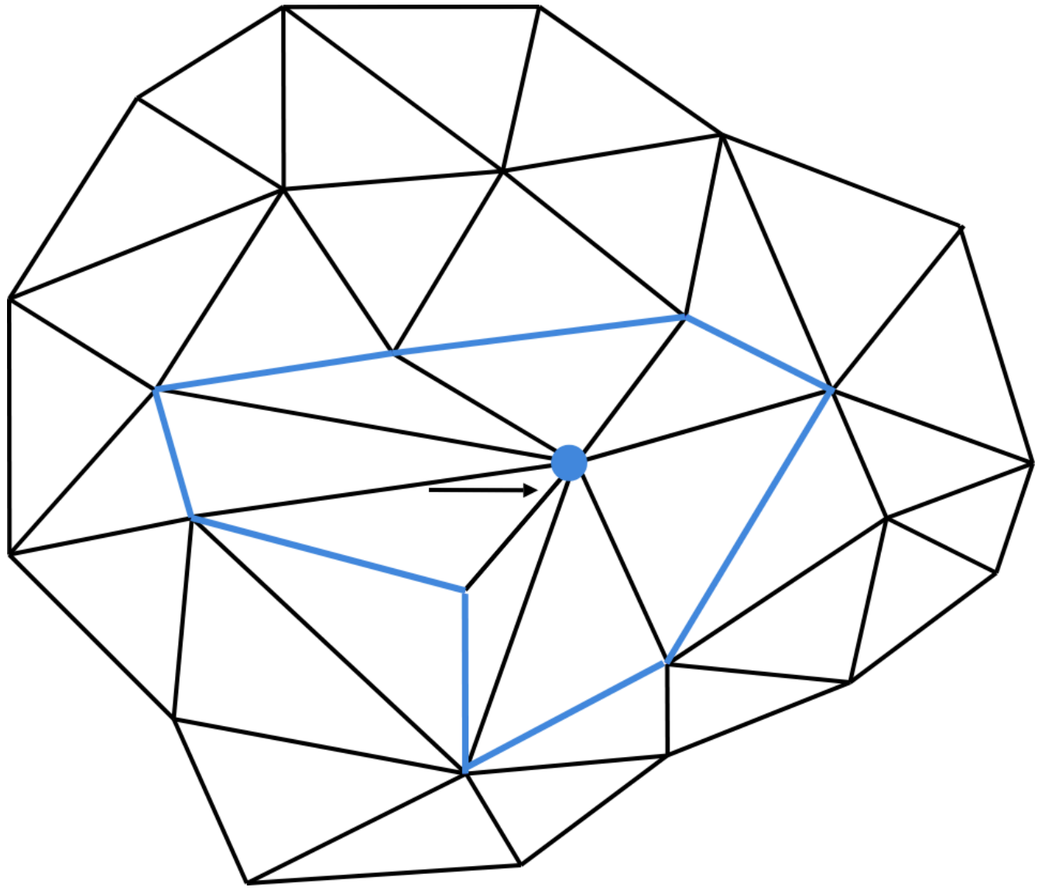
[M. Pauly, "Mesh Decimation", 2006]

Halfedge Collapse (4)



[M. Pauly, "Mesh Decimation", 2006]

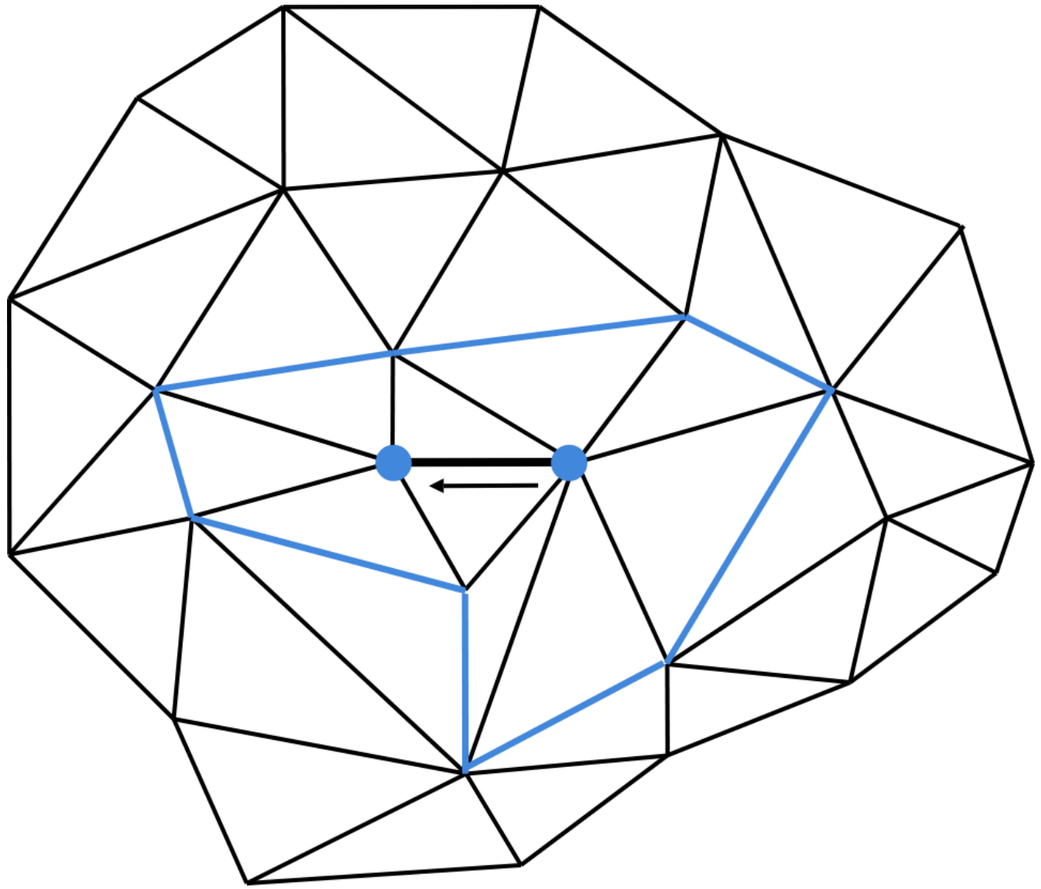
Halfedge Collapse (5)



This worked completely fine, but what if we do it in the other direction?

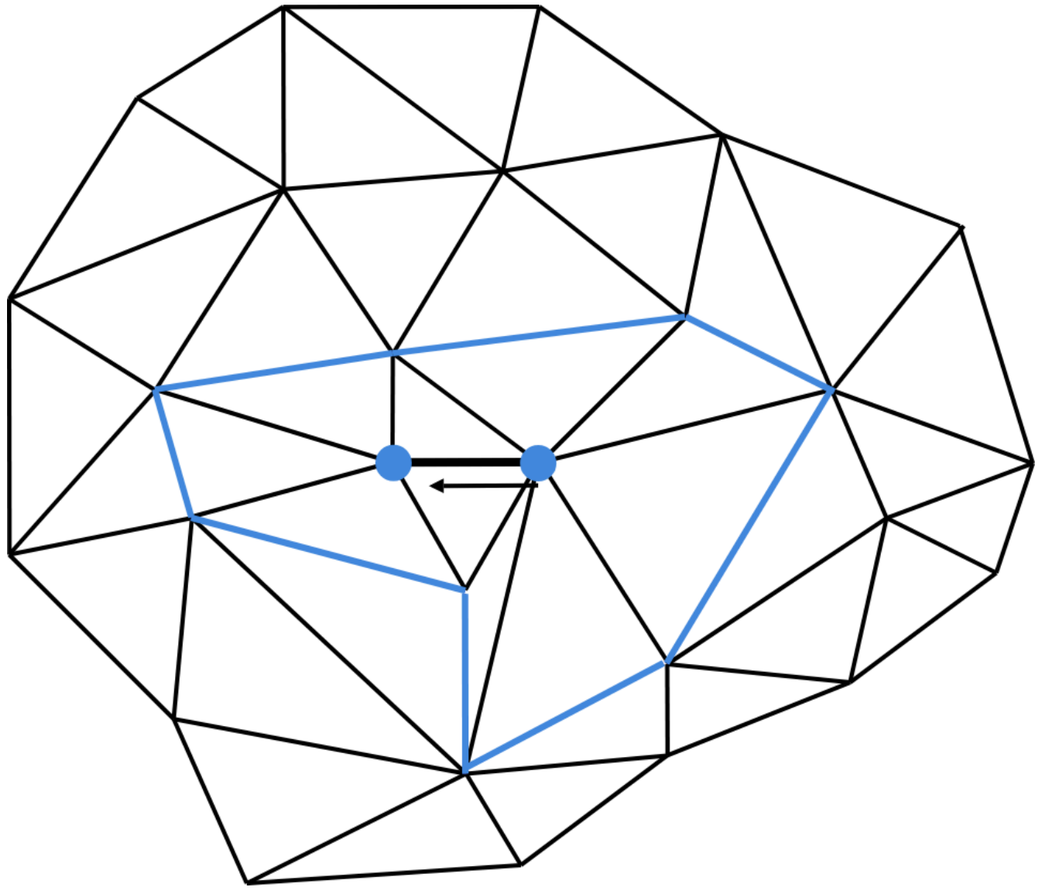
[M. Pauly, "Mesh Decimation", 2006]

Halfedge Collapse (6)



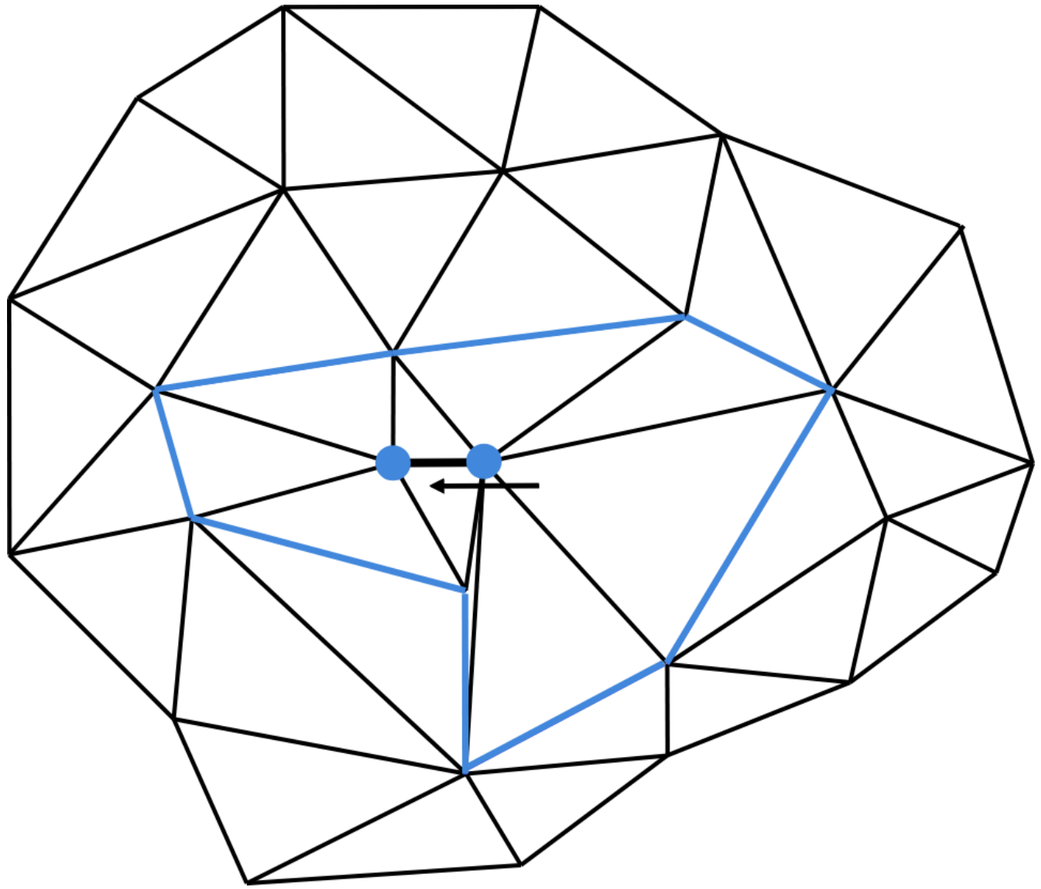
[M. Pauly, "Mesh Decimation", 2006]

Halfedge Collapse (7)



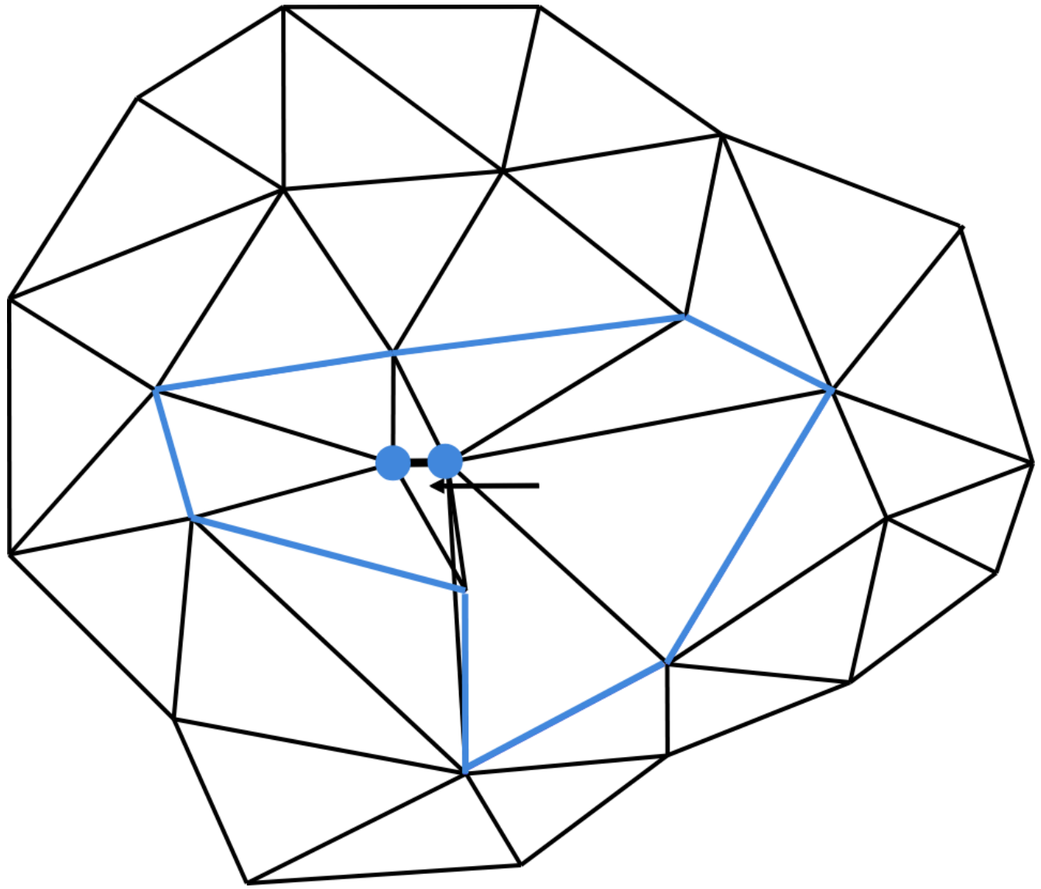
[M. Pauly, "Mesh Decimation", 2006]

Halfedge Collapse (8)



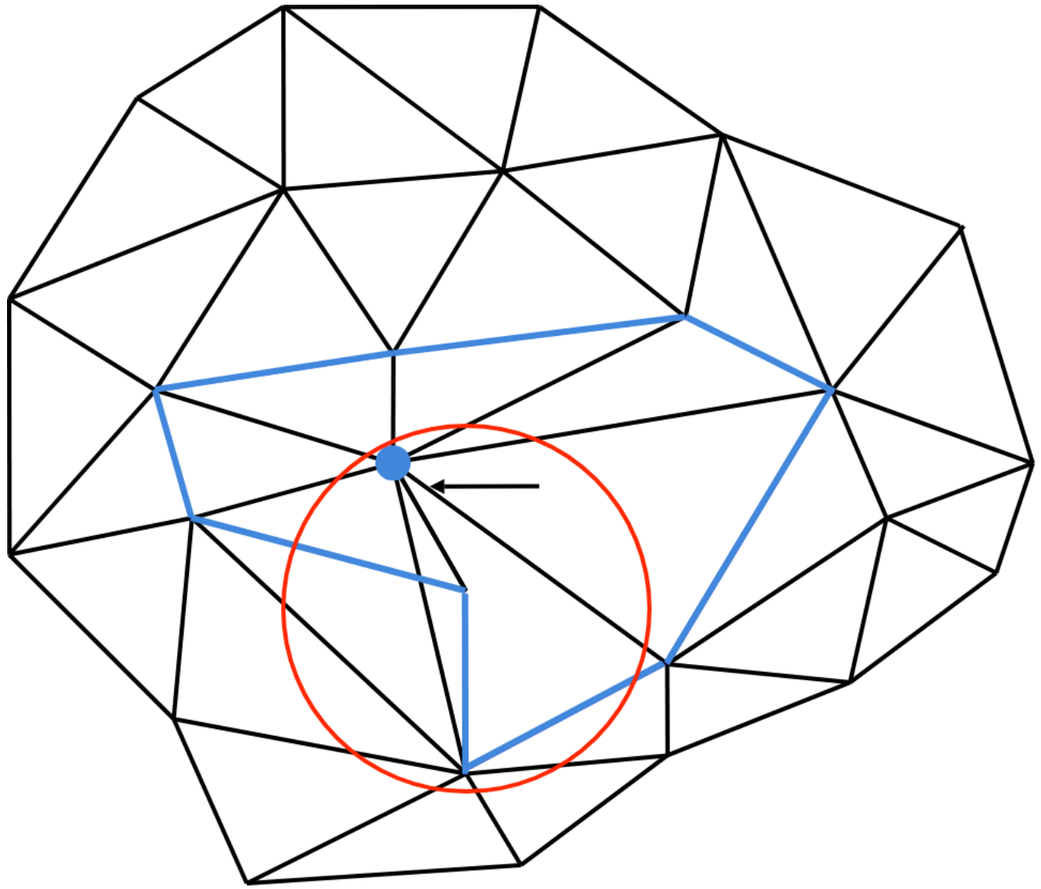
[M. Pauly, "Mesh Decimation", 2006]

Halfedge Collapse (9)



[M. Pauly, "Mesh Decimation", 2006]

Halfedge Collapse (10)



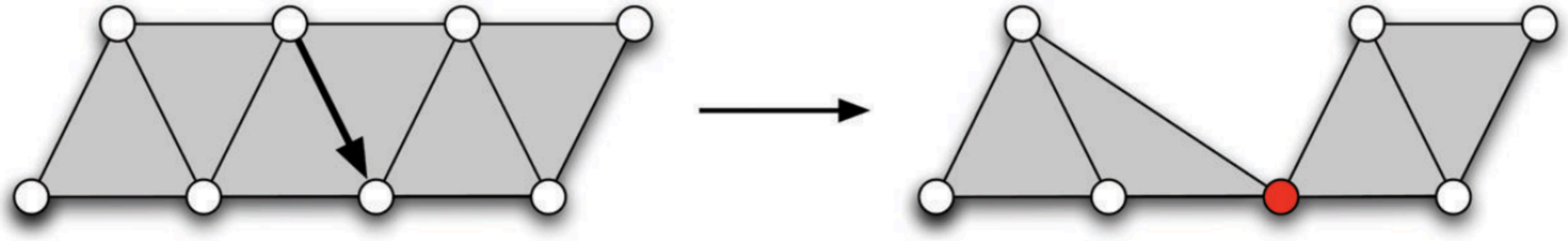
Problem: We have created a topological inconsistency - a self intersection!

[M. Pauly, "Mesh Decimation", 2006]

Illegal Halfedge Collapses (1)

Collapsing an edge (p, q) is a valid operation if and only if the following two criteria hold [Hoppe et al. 93]:

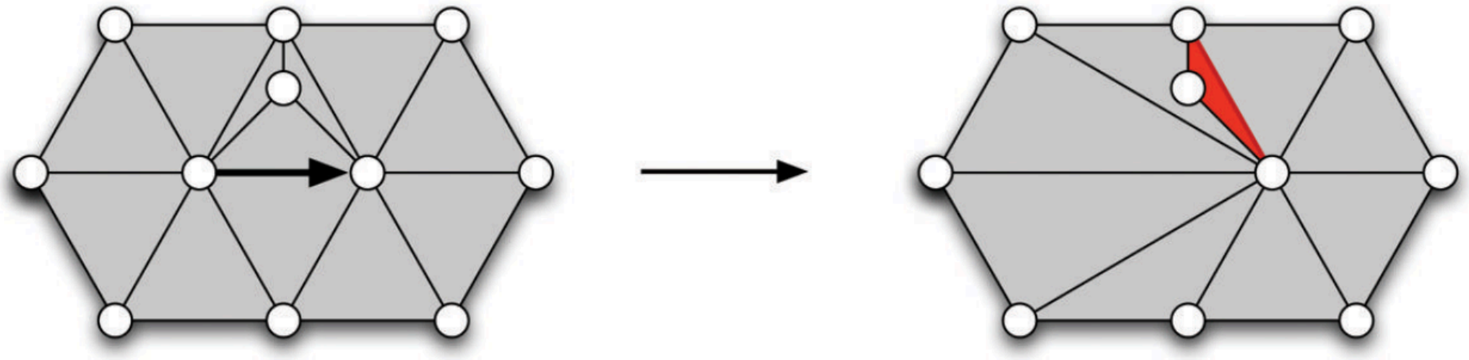
Illegal Halfedge Collapses (2)



[Botsch, Mario et al. “Polygon Mesh Processing.” (2010).]

If both p and q are boundary vertices, then the edge (p, q) has to be a boundary edge.

Illegal Halfedge Collapses (3)



[Botsch, Mario et al. “Polygon Mesh Processing.” (2010).]

For all vertices r incident to both p and q there has to be a triangle (p, q, r). In other words, the intersection of the one-rings of p and q consists of vertices opposite the edge (p, q) only.

Vertex Ranking (1)

How do we find the best removal candidate?

Incremental mesh decimation ranks all removal operations. Ranking may be defined by

- Distance measure
- Fairness measure
  - Triangle shape, i.e. rank by ratio of circum circle radius to shortest edge of all incident triangles after removal
  - Visual smoothness, i.e. rank by maximum or average normal jump. (We have to compute normal in any case to detect flipping!)
  - Colour decimation
  - Texture distortion

Vertex Ranking (2)

Re-evaluation after every vertex removal is expensive

Better:

- Candidates for removal are kept in a heap data structure, where the best removal operation is on top.
- If candidate has to be re-evaluated it is
  - deleted from the heap
  - re-inserted with new value

Incremental Decimation can be done in two ways

Greedy Reduction

For each region:

1. evaluate quality after decimation
2. enqueue (quality, region)

Repeat until no further reduction possible:

1. get best mesh region from queue
2. apply decimation operator
3. update queue

With Error Control

For each region:

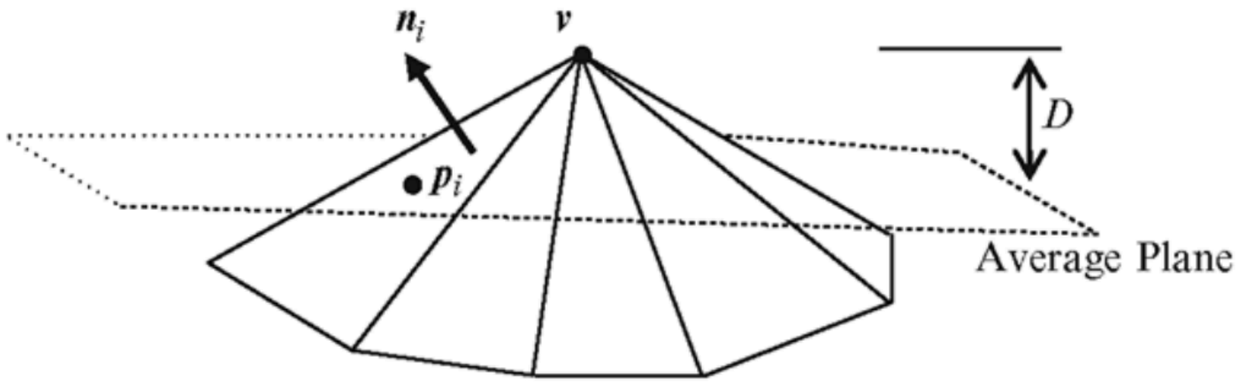
1. evaluate quality after decimation
2. enqueue (quality, region)

Repeat until no further reduction possible:

1. get best mesh region from queue
2. **if error $< \epsilon$**
   - **apply decimation operator**
   - **update queue**

The **error control** stops the decimation when the quality becomes too bad.

Error Metrics (1)



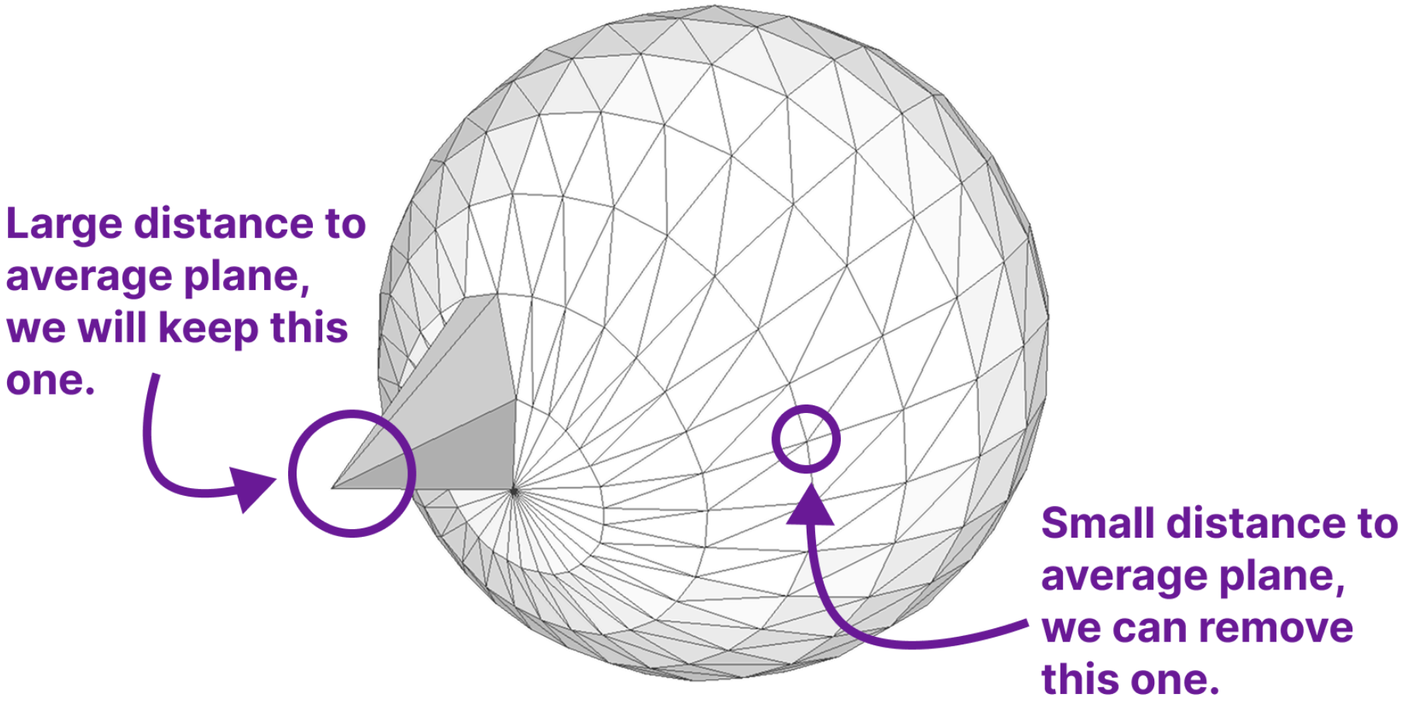
[U. Augsdörfer, "Mesh Decimation", 2020]

Formula for D can be found in the assignment sheet.

1. Compute the normal vector of each adjacent plane.
2. Compute the average normal vector n_{avg}.
3. Compute the barycentric average of all faces p_{avg}.
4. n_{avg} and p_{avg} define the average plane.
5. Finally compute distance D of vertex v to the average plane.

The distance to the average plane is a **local** error metric: Measures the distance between a face and a subpatch.

Error Metrics (2)



Remember: We want to remove vertices whose absence is barely noticeable. Removing the spike would remove an important feature that can make the difference in ML models (for example).

Error Metrics (3)

The **Hausdorff Distance** — a global error metric

This distance measure is defined to be the maximum minimum distance. If we have two sets A and B, then $H(A, B)$ is found by computing the minimum distance $d(a, B)$ for each point $a \in A$ and then taking the maximum of those values:

$$H(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|.$$

Notice that, in general, $H(A, B) \neq H(B, A)$. The symmetric Hausdorff distance is defined as the maximum of both values:

$$d_H(A, B) = \max\left(H(A, B),\ H(B, A)\right).$$

Error Metrics (4)

Computing the Hausdorff distance for a mesh:

- We compute the one-sided Hausdorff distance $H(A, B)$.
- A are the points of the original mesh.
- B are the points of the decimated mesh.
- **Goal:** Efficient computation of the Hausdorff distance during mesh decimation.
- **Track vertex–triangle assignments**
  - Maintain a mapping from *original mesh vertices* to *triangles of the decimated mesh*.
- **During an edge collapse**
  - Removed vertices **p** and **q** (or only **p** in the case of a half-edge collapse).
  - Assign these vertices to the *nearest triangle in the local neighborhood*.

Error Metrics (5)

- **Update affected regions**
  - An edge collapse changes the geometry of adjacent triangles.
  - Vertices previously assigned to these triangles must be *re-distributed* to appropriate nearby triangles.
- Every triangle t_i of the decimated mesh at any time maintains a list of original vertices belonging to the currently associated patch S_i.
- The Hausdorff distance is then evaluated by finding the most distant point in this list.

Discussion

Vertex Clustering

- fast, but difficult to control
- topology changes, non-manifold meshes
- global error bound, but often not close to optimum

Iterative Decimation

- good trade-off between mesh quality and speed
- explicit control over mesh topology

Questions?