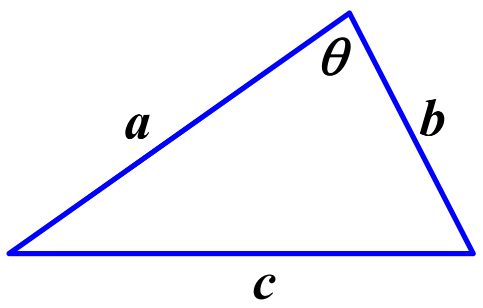
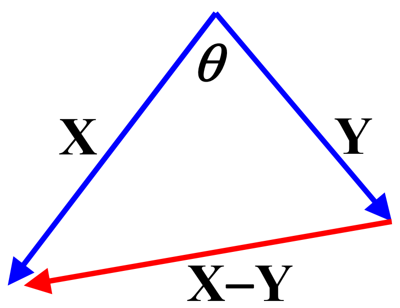
Mesh Simplification

The Law of Cosine

- **Here are some commonly used formulas.**
- **First, we learn that $c^2 = a^2 + b^2 - 2ab\cos(\theta)$, where θ is the angle opposite to side c.**
- **Vector form: $|X-Y|^2 = |X|^2+|Y|^2-2|X|\cdot|Y|\cos(\theta)$.**
- **Note that $|X|^2 = X\cdot X$, where $\cdot$ is the inner product.**
- **Since $(X-Y)\cdot(X-Y) = X\cdot X+Y\cdot Y-2X\cdot Y$, we have $X\cdot Y = |X|\cdot|Y|\cos(\theta)$.**

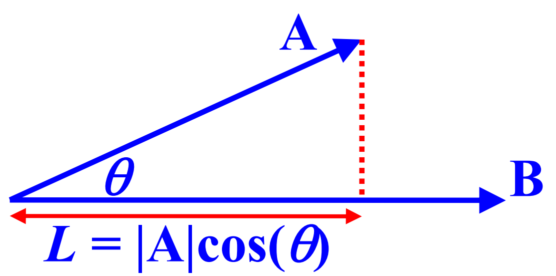




Projection of a Vector to Another

- Let **A** and **B** be two vectors. We wish to compute the length of projecting **A** to **B**.
- It is obvious that the length is $L = |\mathbf{A}|\cos(\theta)$.
- Since $\mathbf{A}\cdot\mathbf{B}=|\mathbf{A}|\cdot|\mathbf{B}|\cos(\theta)$, we have

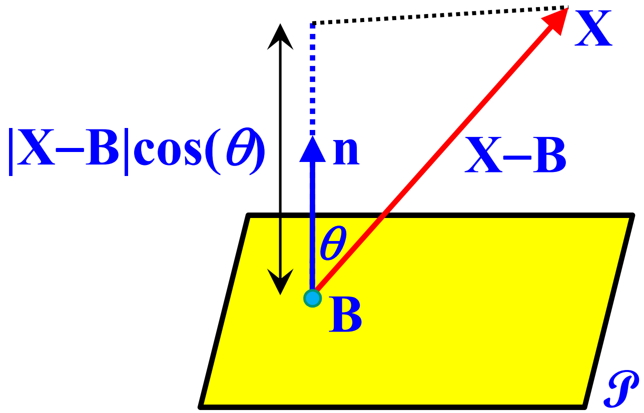
$$L = |\,\mathbf{A}\,|\cos(\theta) = |\,\mathbf{A}\,|\frac{\mathbf{A}\cdot\mathbf{B}}{|\,\mathbf{A}\,|\cdot|\,\mathbf{B}\,|} = \frac{\mathbf{B}}{|\,\mathbf{B}\,|}\mathbf{A}$$



Point to a Plane Distance: 1/2

- Let a plane $\mathscr{P}$ be represented by a base point **B** and a normal vector **n**, where $|\mathbf{n}| = 1$.
- Compute the distance from a point **X** to $\mathscr{P}$.
- Projecting **X** to **n** yields the distance $|\mathbf{X}-\mathbf{B}|\cos(\theta)$.
- Since $\cos(\theta)=(\mathbf{X}-\mathbf{B})\cdot\mathbf{n}/(|\mathbf{X}-\mathbf{B}|\cdot|\mathbf{n}|)=(\mathbf{X}-\mathbf{B})\cdot\mathbf{n}/|\mathbf{X}-\mathbf{B}|$, the distance is simply $(\mathbf{X}-\mathbf{B})\cdot\mathbf{n}$.

Compute the perpendicular foot from **X** to plane $\mathscr{P}$. Easy!



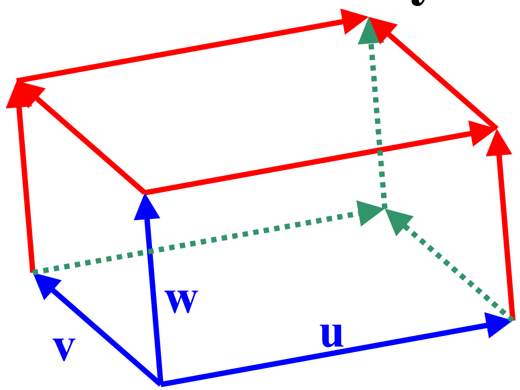
Point to Plane Distance: 2/2

- Sometimes the plane is given by $ax+by+cz+d = 0$, where $a^2 + b^2 + c^2 = 1$ (*i.e.*, normalized).
- The normal vector of this plane is $\mathbf{n} = < a, b, c >$.
- If $\mathbf{B} = < u, v, w >$ is a point in this plane, we have $au + bv + cw + d = 0$ and $au + bv + cw = -d$.
- The distance from $\mathbf{X} = < x, y, z >$ to this plane is $(\mathbf{X} - \mathbf{B})\bullet\mathbf{n}$.
- Plugging $\mathbf{B}$ and $\mathbf{n}$ into this equation yields:

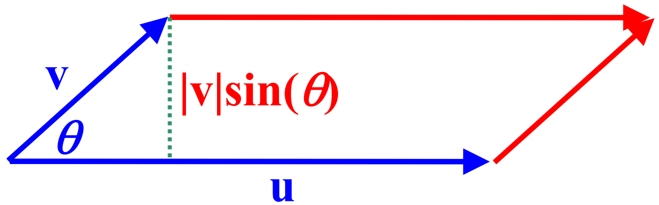
$$\begin{aligned}(\mathbf{X} - \mathbf{B})\bullet\mathbf{n} &= (<x,y,z> - <u,v,w>) \bullet <a,b,c> \\ &= <x,y,z> \bullet <a,b,c> - <u,v,w> \bullet <a,b,c> \\ &= (ax+by+cz) - (au+bv+cw) \\ &= (ax+by+cz) - (-d) \\ &= ax + by + cz + d\end{aligned}$$

Volume of a Parallelepiped: 1/2

- **A parallelepiped is defined by three vectors u, v and w.**



- **The parallelogram defined by u and v has an area of $|\mathbf{u}|\cdot|\mathbf{v}|\sin(\theta)$, which is the length of vector $\mathbf{u}\times\mathbf{v}$, where θ is the angle between u and v.**

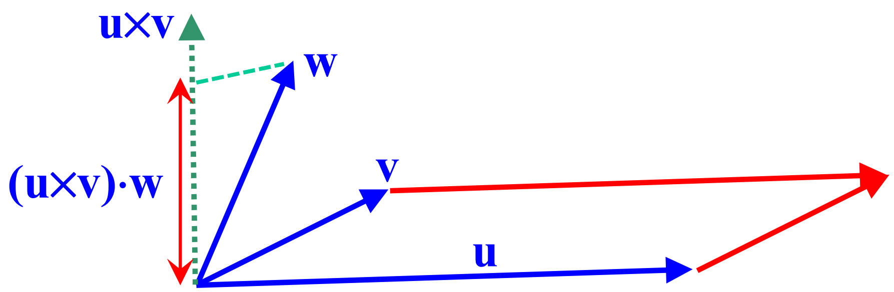


$$\text{area} = \text{base}\times\text{height} = |\mathbf{u}|(|\mathbf{v}|\sin(\theta)) = |\mathbf{u}||\mathbf{v}|\sin(\theta) = |\mathbf{u}\times\mathbf{v}|$$

Volume of a Parallelepiped: 2/2

- The volume of a parallelepiped is the product of its base area and its height.
- The base area is $|\mathbf{u}\times\mathbf{v}|$.
- Projecting $\mathbf{w}$ to $\mathbf{u}\times\mathbf{v}$ yields the height $(\mathbf{u}\times\mathbf{v})\cdot\mathbf{w}/|\mathbf{u}\times\mathbf{v}|$.
- Therefore, the volume is:

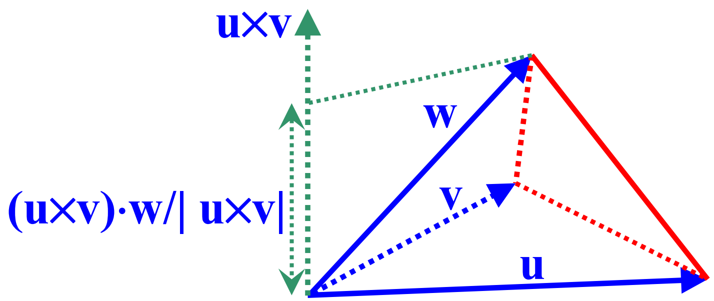
$$\begin{aligned}\text{Volume} &= \text{BaseArea}\times\text{Height}\\ &= |\mathbf{u}\times\mathbf{v}|\frac{(\mathbf{u}\times\mathbf{v})\cdot\mathbf{w}}{|\mathbf{u}\times\mathbf{v}|}\\ &= (\mathbf{u}\times\mathbf{v})\cdot\mathbf{w}\end{aligned}$$



Volume of a Tetrahedron

- A tetrahedron is also defined by three vectors **u**, **v** and **w**.
- The volume of a tetrahedron is (BaseArea×Height)/3.
- Base area is half of the parallelogram defined by **u** and **v**, and is equal to |u×v|/2.
- Height is our old friend, projecting **w** to **u×v**, which is (u×v)·w/|u×v|.
- Therefore, the volume is

$$\begin{aligned} \text{Volume} &= \frac{1}{3}\text{BaseArea}\times\text{Height} \\ &= \frac{1}{3}\left(\frac{1}{2}|\mathbf{u}\times\mathbf{v}|\right)\frac{(\mathbf{u}\times\mathbf{v})\cdot\mathbf{w}}{|\mathbf{u}\times\mathbf{v}|} \\ &= \frac{1}{6}(\mathbf{u}\times\mathbf{v})\cdot\mathbf{w} \end{aligned}$$



Mesh Simplification: 1/2

- ***Mesh simplification/decimation*** **is a class of algorithms that transform a given polygonal mesh into another with fewer faces, edges, and vertices.**
- **The simplification process is usually controlled by a set of *user-defined quality criteria* that can preserve specific properties of the original mesh as much as possible (*e.g*., geometric distance, visual appearance , etc).**
- **Mesh simplifications *reduces the complexity* of a given mesh.**

Mesh Simplification: 2/2

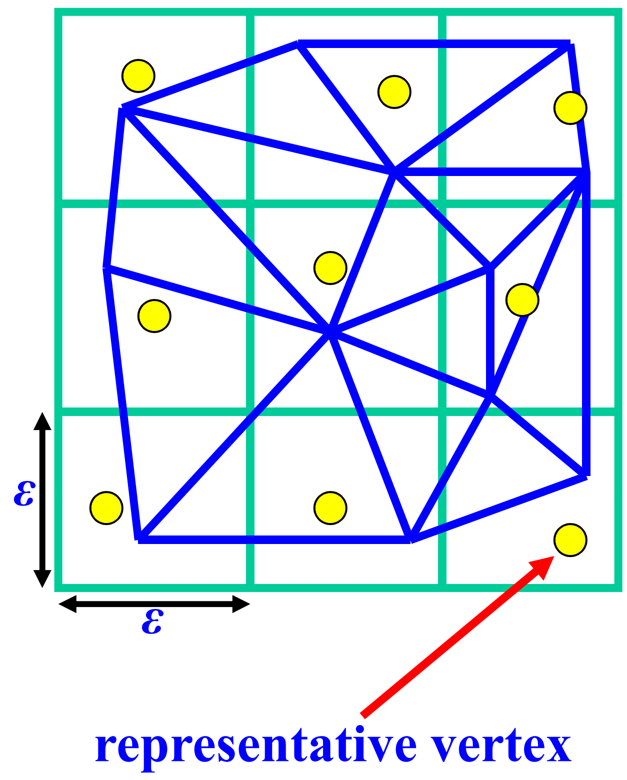
- **Simplification schemes usually work iteratively (*i.e.*, removing a vertex/edge at a time) and can be reversed. Thus, one can transmit the final result followed by the "reversed" operators.**
- **A mesh simplification scheme can be viewed as a *decomposition operator* to obtain a low frequency component (*i.e.*, the decimated mesh) and a high frequency component (*i.e.*, the difference between the original and decimated meshes). Then, a *reconstruction operator* can perform the inverse decimation to recover the original data from its low frequency component.**

Mesh Simplification Approaches

- **Vertex Clustering: It is in general fast, robust and of $O(n)$, where n is the number of vertices; however, quality is not always satisfactory.**
- **Incremental Decimation: It can deliver higher quality meshes in most cases, and can take arbitrary user-defined criteria into account according to how the next removal operation is chosen. However, complexity may be $O(n\log_2 n)$ or even $O(n^2)$.**
- **Resampling: The most general approach; however, new samples may be freely distributed.**

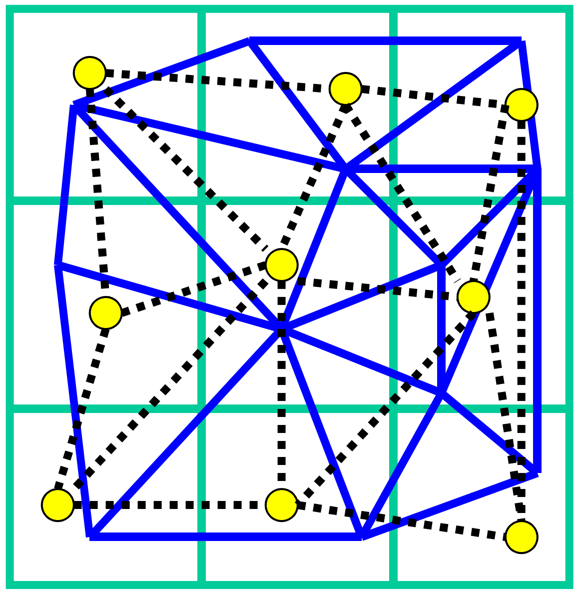
Vertex Clustering: 1/4

- **Given a tolerance $\varepsilon > 0$, the bounding space of the given mesh is partitioned into cells with diameter $\leq \varepsilon$.**
- **For each cell a *representative vertex* is computed (will talk about this later). If a cell has more than one vertices, they are all mapped to this representative vertex.**



Vertex Clustering: 2/4

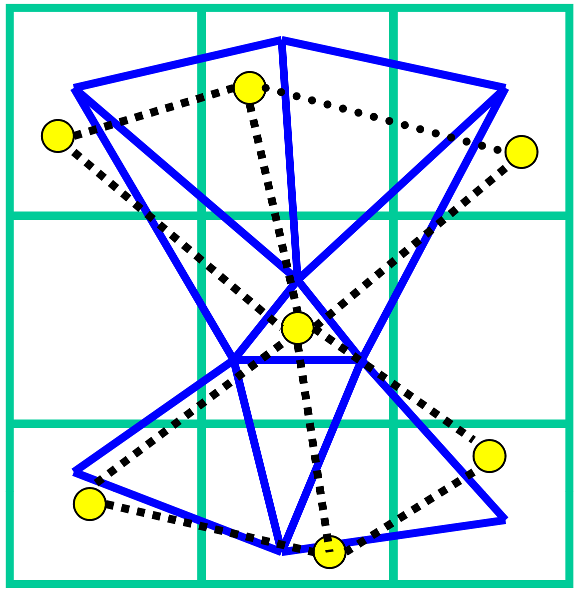
- Then, degenerate triangles are removed.
- If **P** and **Q** are the representative vertices of $p_0, p_1, \ldots, p_m$ and $q_0, q_1, \ldots, q_n$, respectively, **P** and **Q** are connected in the decimated mesh if at least one pair of vertices (p_i, q_j) was connected in the original mesh.



solid: original mesh
dotted: new mesh

Vertex Clustering: 3/4

- The resulting mesh may not be a 2-manifold even though the original one is, because **a portion of a surface could collapse to a point.**
- However, it can reduce the complexity of a mesh significantly, and guarantee a global approximation of the original mesh.



solid: original mesh
dotted: new mesh

Vertex Clustering: 4/4

- **How to compute those representatives?**
  - **The easiest way is to average the vertices in the same cell. If $\mathbf{P}_1$, $\mathbf{P}_2$, …, $\mathbf{P}_k$ are vertices in the same cell, then the representative is $\mathbf{P} = (\mathbf{P}_1 + \mathbf{P}_2 + \ldots + \mathbf{P}_k)/k$.**
  - **Or, depending on the importance of each vertex (of the mesh) one might assign a weight $w_i \geq 0$ to vertex $\mathbf{P}_i$. Then, the representative of $\mathbf{P}_1$, $\mathbf{P}_2$, …, $\mathbf{P}_k$ in the same cell is their *weighted* average:**

$$\mathbf{P} = \frac{w_1\mathbf{P}_1 + w_2\mathbf{P}_2 + \ldots + w_k\mathbf{P}_k}{w_1 + w_2 + \ldots + w_k}$$

Incremental Decimation: 1/2

- **Incremental algorithms remove one vertex or edge at a time based on user-specified criteria.**
- **Criteria can be binary or continuous.**
- **Binary criteria determine if a vertex is allowed to remove (*i.e.*, yes or no), while a continuous one rates the quality of the mesh (*i.e.*, roundness of triangles, small normal changes between neighboring triangles) before/after removal.**

Incremental Decimation: 2/2

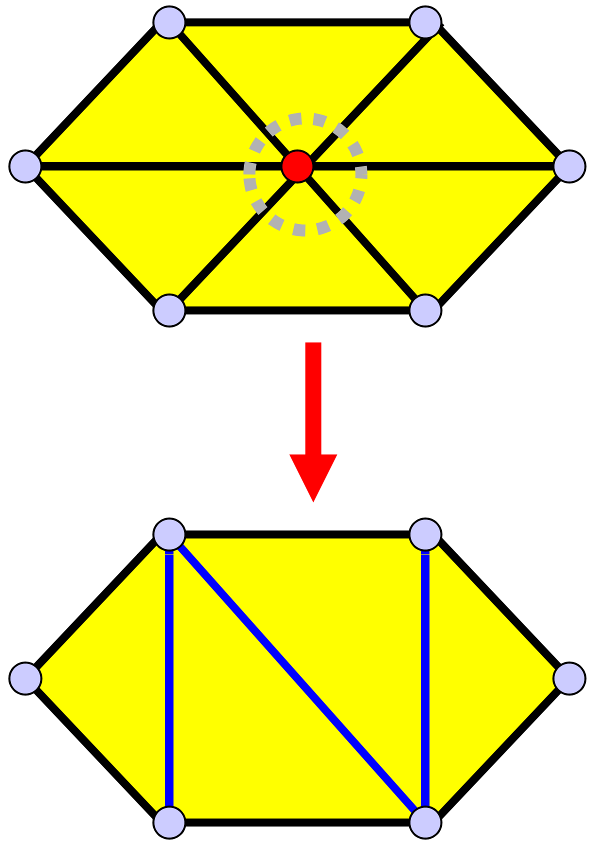
- **The surface geometry changes in the neighborhood of the removed vertex/edge, and the quality criteria have to be re-evaluated.**
- **To make the re-evaluation process more efficient, the candidates for removal are usually stored in a heap with the best removal operation on top.**
- **In this way, each update only costs $O(\log n)$ for large meshes if the criteria evaluation has constant time complexity.**

Topological Operators

- **There are a number of removal operators, some of which can preserve the mesh topology. These decimation operators are referred to as *Euler-Operators*. See CS3621 course page.**
- **Commonly used topological operators include:**
  - **Vertex removal (inverse: vertex insertion)**
  - **Edge collapse (inverse: edge split)**
  - **Half edge collapse (inverse: restricted vertex split)**

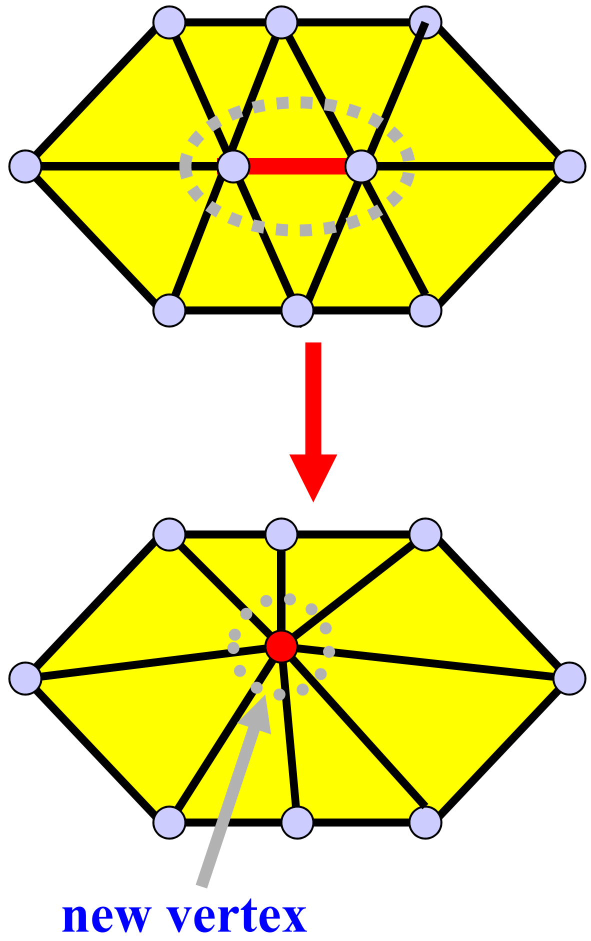
Vertex Removal

- **Vertex removal** deletes a vertex and its adjacent edges and faces, creating a ***k***-side hole, where ***k*** is the valence of the vertex.
- **This hole is triangulated by adding *k*-2 triangles back.**
- **Thus, the # of vertices and # of triangles are reduced by 1 and 2, respectively.**



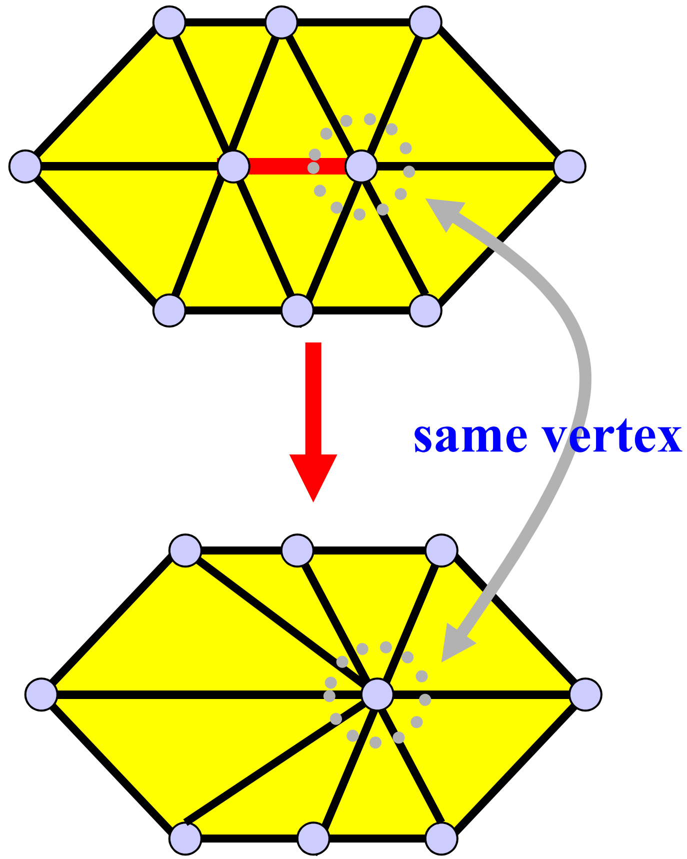
Edge Collapse

- **Edge collapse** selects an edge and collapses it to a new vertex. Its two adjacent triangles also collapse to two edges.
- **Thus, the # of vertices and # of triangles are reduced by 1 and 2, respectively.**
- **However, we are allowed to choose a *new* vertex!**



Half-Edge Collapse

- Given a selected edge with adjacent vertices p and q, the **half-edge collapse** operator moves p to q or q to p.
- This is a special case of the edge collapse operator.
- Note that moving p to q and moving q to p are *two* different operations.
- Note also that no degree of freedom is available.



Decimation Operator Notes: 1/2

- While the **half-edge collapse** operator is a special case of the **edge collapse** operator, its effect becomes noticeable only for extremely strong decimation where the exact location of individual vertices really matters.
- The global optimization that uses user specified criteria to make selections is completely separate from the decimation operator. This makes the design of decimation more orthogonal.

Decimation Operator Notes: 2/2

- **All three operators preserve mesh topology and the topology of the underlying surface may change near the end of decimation.**
- **Non-Euler operators CAN change mesh topology.**
- **The vertex contraction operator merges two *arbitrary* vertices into one even if they are not connected by an edge is a good example.**
- **The vertex contraction operator reduces the # of vertices by 1 but preserves the # of faces/edges.**

A Vertex Decimation Algorithm for Triangular Mesh: 1/15

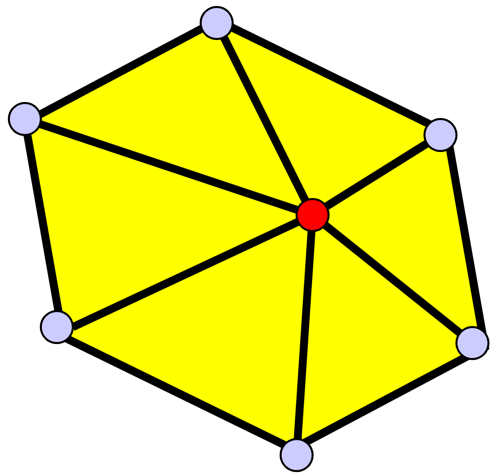
- **One of the earliest decimation algorithm was due to Schroeder, Zarge and Lorensen published in *SIGGRAPH 1992*.**
- **This algorithm uses vertex removal only and has a scheme as follows.**

```
while there is a vertex X that can be removed do
   begin
      apply the vertex removal operator to X;
      this creates a hole, not necessary planar;
      re-triangulate the hole;
   end
```

A Vertex Decimation Algorithm for Triangular Mesh: 2/15

- Not all vertices are candidates for decimation.
- Each vertex is assigned one of five possible classifications: simple, complex, boundary, interior edge, or corner vertex.
- A simple vertex is surrounded by a closed fan of triangles.

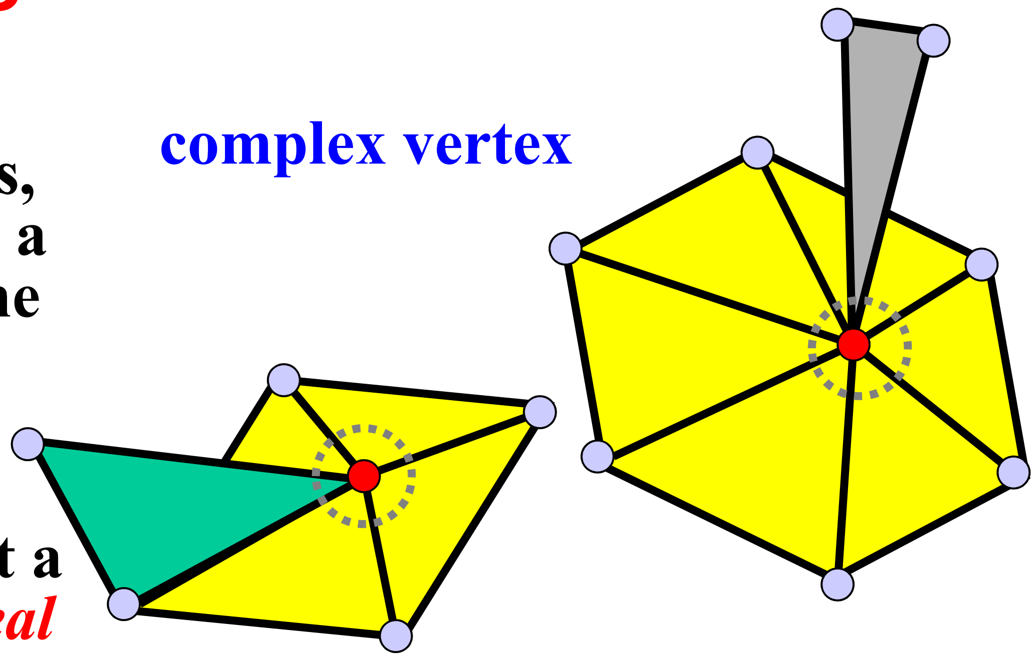
a simple vertex



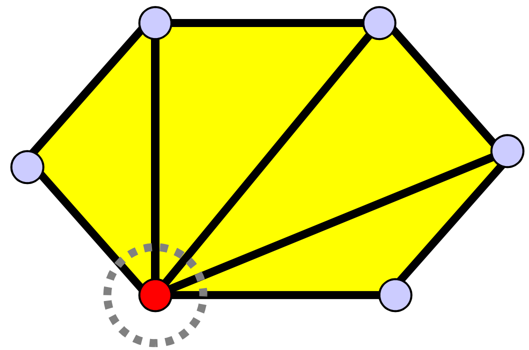
A Vertex Decimation Algorithm for Triangular Mesh: 3/15

- If an edge is shared by more than two triangles, or if a vertex is used by a triangle that is not in the fan, this vertex is a **complex vertex**.
- If a mesh contains a complex vertex, it is not a 2-manifold. ***We only deal with 2-manifolds in this course.***
- If a vertex is on the boundary of a mesh, it is a **boundary vertex**.

complex vertex

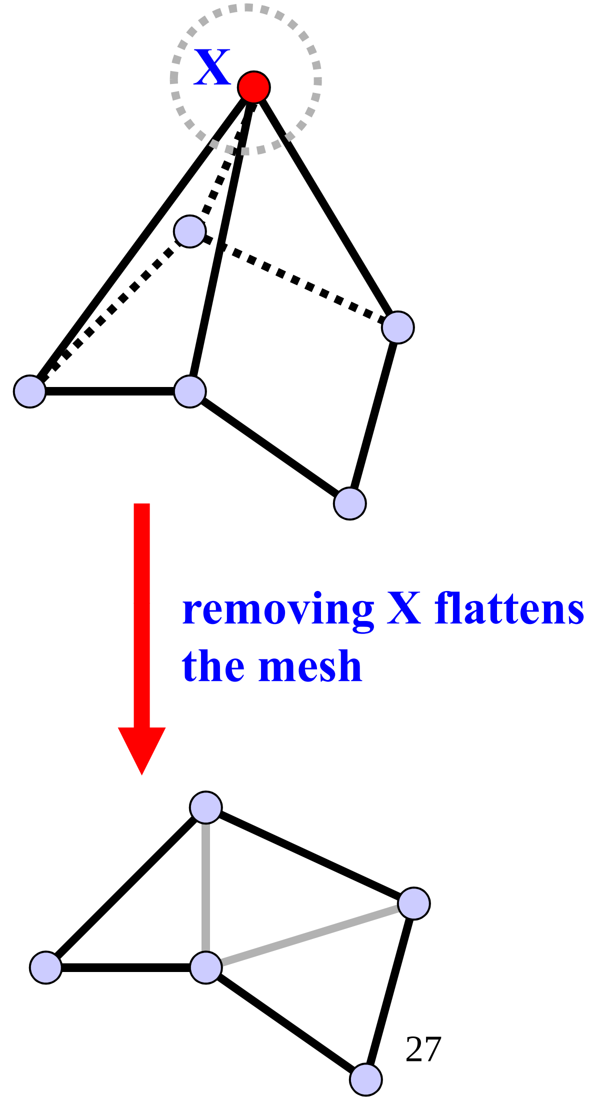


boundary vertex



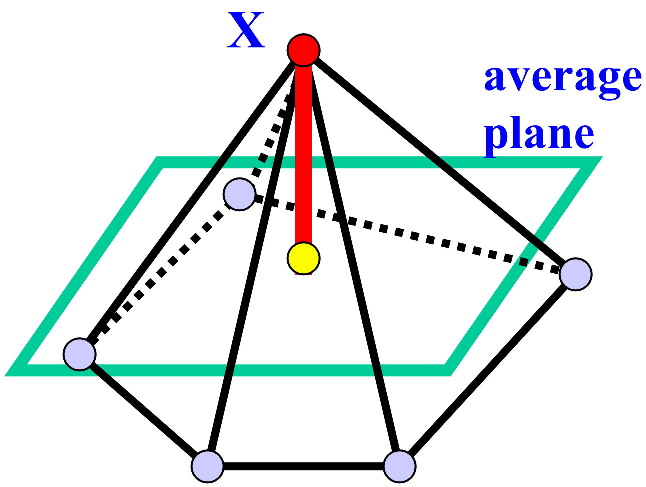
A Vertex Decimation Algorithm for Triangular Mesh: 4/15

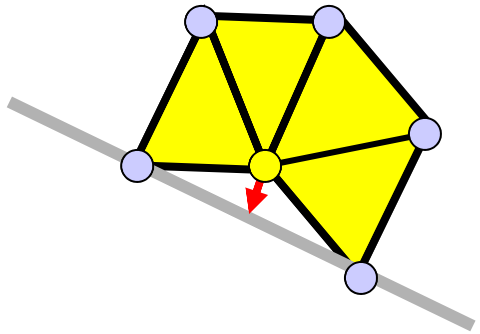
- **User Specified Criteria (Basic Idea):**
  - ***Do not remove sharp corners***
  - **If vertex X is “far” away from its adjacent vertices, X should not be removed because removing X flattens the vicinity of vertex X.**
  - **Thus, good candidates should be vertices in “flat” regions.**
  - **The “flatness” is measured by a plane, an average plane, representing the vicinity of X’s adjacent vertices.**



A Vertex Decimation Algorithm for Triangular Mesh: 5/15

- **User Specified Criteria:**
  - **If X is a simple vertex, the distance from X to an "average" plane is computed. If this distance is smaller than the given distance (*i.e.*, reasonably flat), X is removed.**
  - **If X is a boundary vertex, then use the distance from this vertex to the *boundary edge line*.**





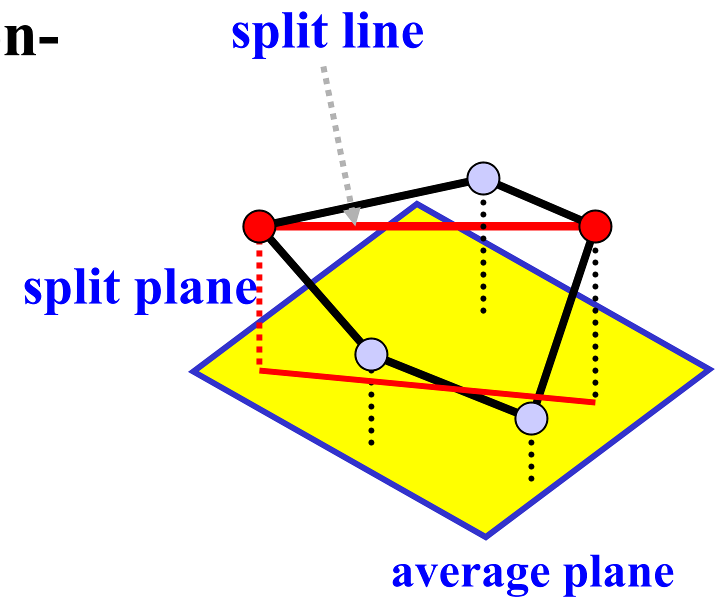
A Vertex Decimation Algorithm for Triangular Mesh: 6/15

- **Compute the "Average Plane":**
  - **Let X be the vertex under consideration.**
  - **Let T_i be a triangle in the fan of X.**
  - **Let c_i, A_i and n_i be the center, area and normal vector of triangle T_i, respectively.**
  - **The base point B and normal vector n of the average plane are calculated as follows:**

$$B = \frac{\sum A_i \times c_i}{\sum A_i} \qquad n = \frac{\sum A_i \times n_i}{\sum A_i}$$

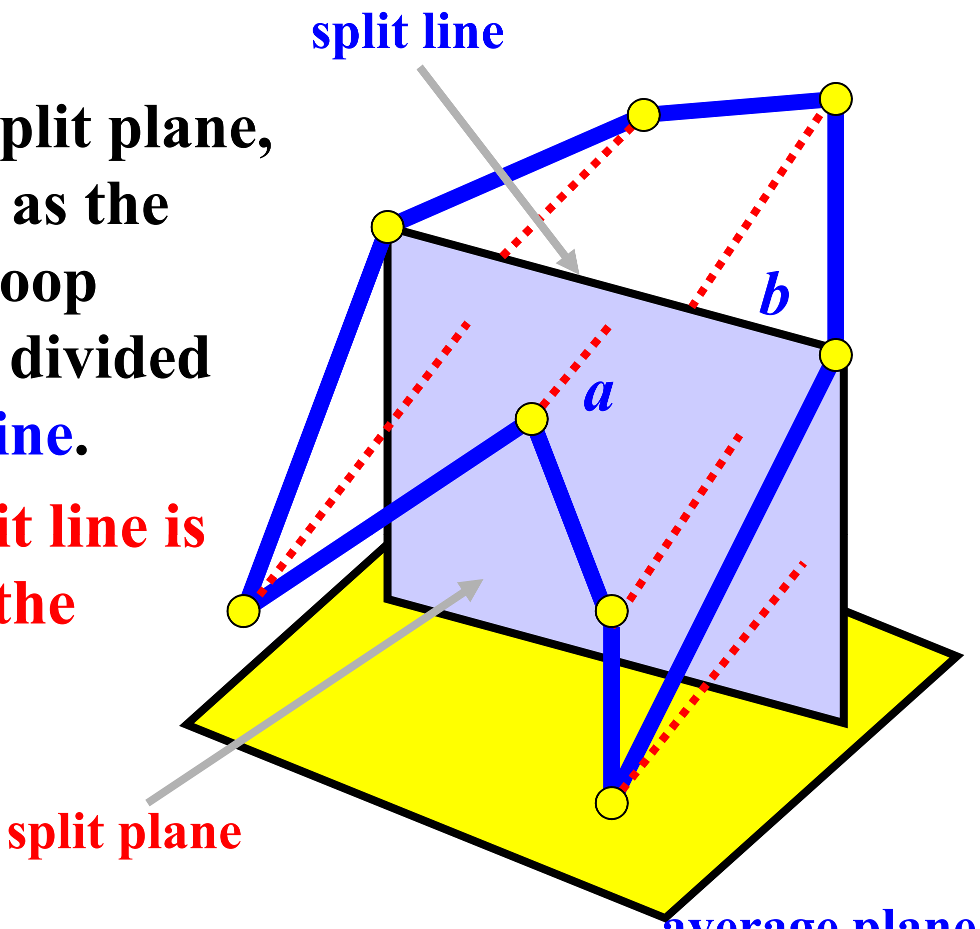
A Vertex Decimation Algorithm for Triangular Mesh: 7/15

- **Split Line and Split Plane:**
  - **A split line is a line joining two non-adjacent vertices.**
  - **A split plane is the plane that satisfies two conditions:**
    1) **it contains a split line and is perpendicular to the chosen average plane**
    2) **it divides the loop into two separate links such that all vertices of one link are in one side of the split plane and the remaining vertices are in the other.**



A Vertex Decimation Algorithm for Triangular Mesh: 8/15

- **Aspect Ratio:**
  - Given a split line and its split plane, the aspect ratio is defined as the *minimum* distance of the loop vertices to the split plane, divided by the length of the split line.
  - The "best" choice of a split line is the one that can produce the maximum aspect ratio.

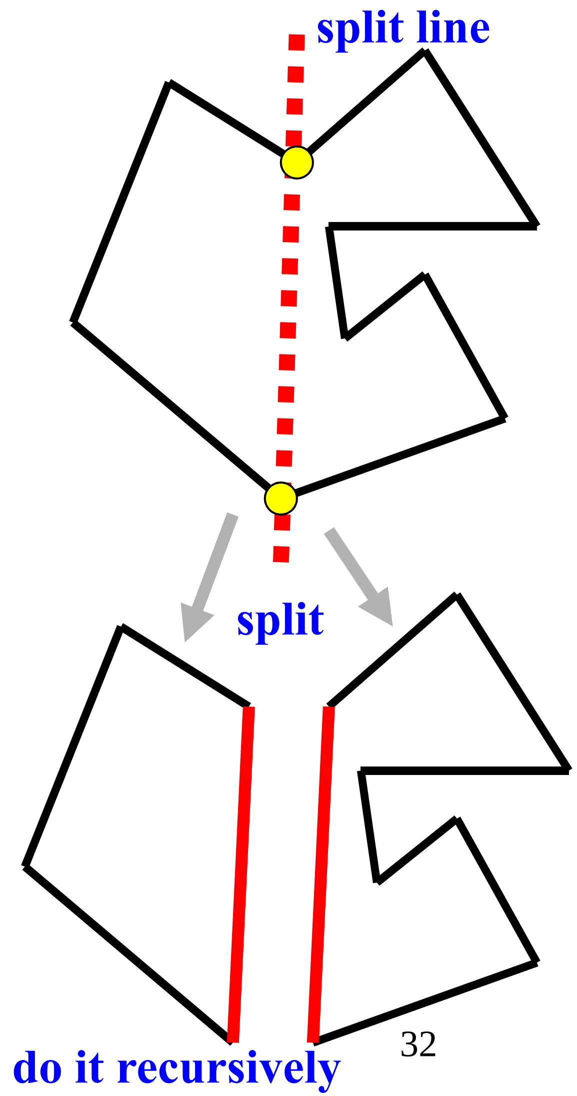


aspect ratio = a/b

A Vertex Decimation Algorithm for Triangular Mesh: 9/15

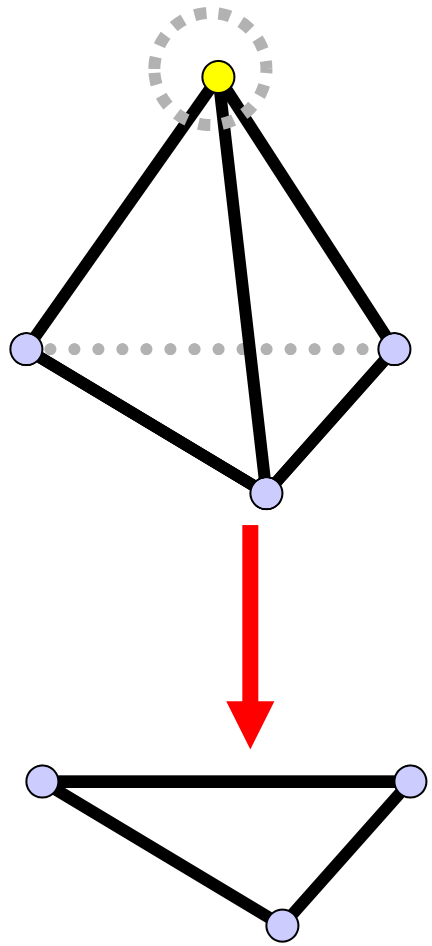
Re-triangulation

- Find a split line with a maximal aspect ratio.
- Each of these two links and the split line forms a loop.
- Recursively re-triangulate each loop.
- If re-triangulation fails, do not remove this vertex.



A Vertex Decimation Algorithm for Triangular Mesh: 10/15

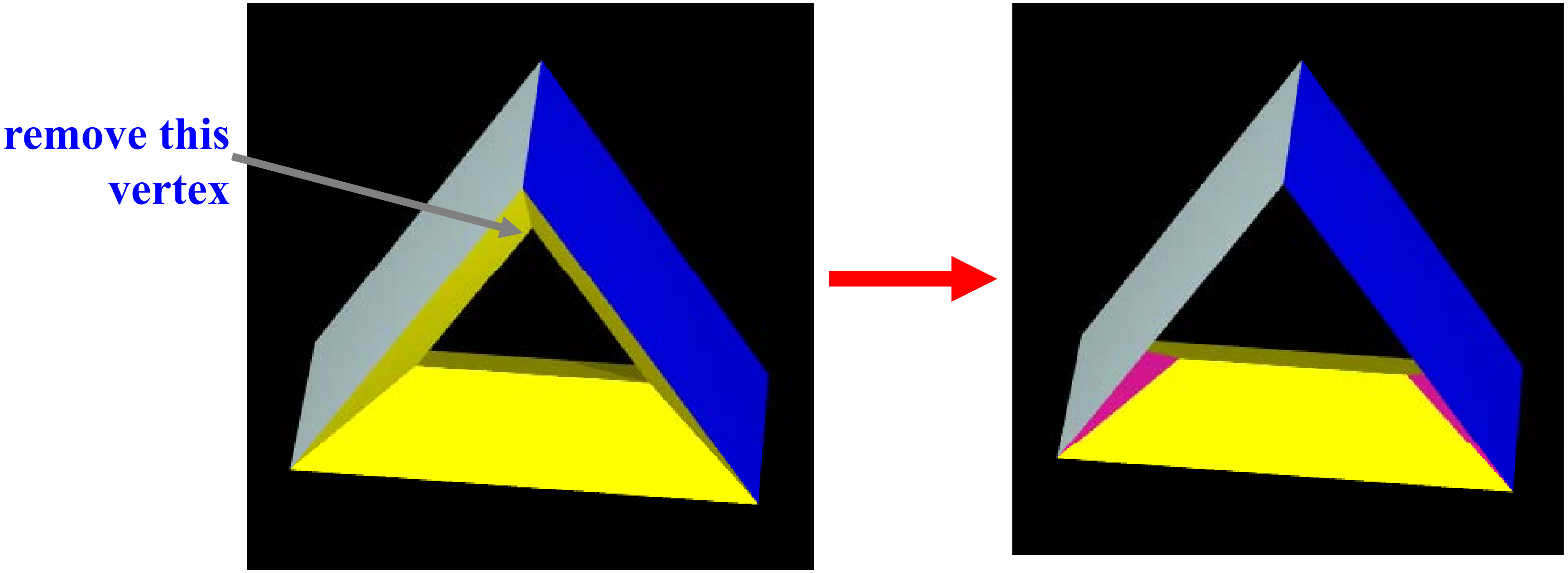
- **A Few Notes: 1/5**
  - **Repeated decimation may produce a tetrahedron. Further decimation reduces it to a triangle. So, we have *two* identical triangles!**
  - **This is a change of topology.**



A Vertex Decimation Algorithm for Triangular Mesh: 11/15

❑ A Few Notes: 2/5

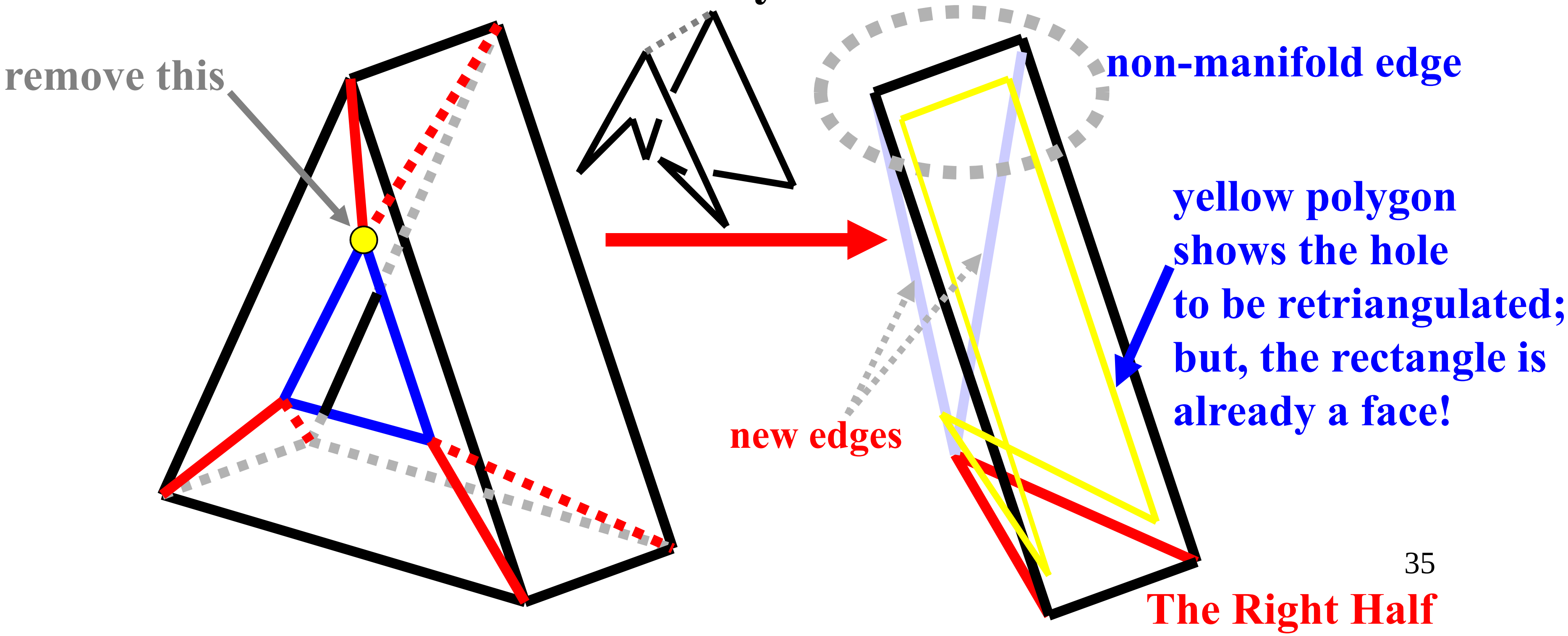
- ❖ If a mesh has holes like a torus, the boundary of a hole could reduce to a triangle (*i.e.*, triangular hole). Removing a vertex from the boundary could create a non-manifold.



A Vertex Decimation Algorithm for Triangular Mesh: 12/15

❑ A Few Notes: 3/5

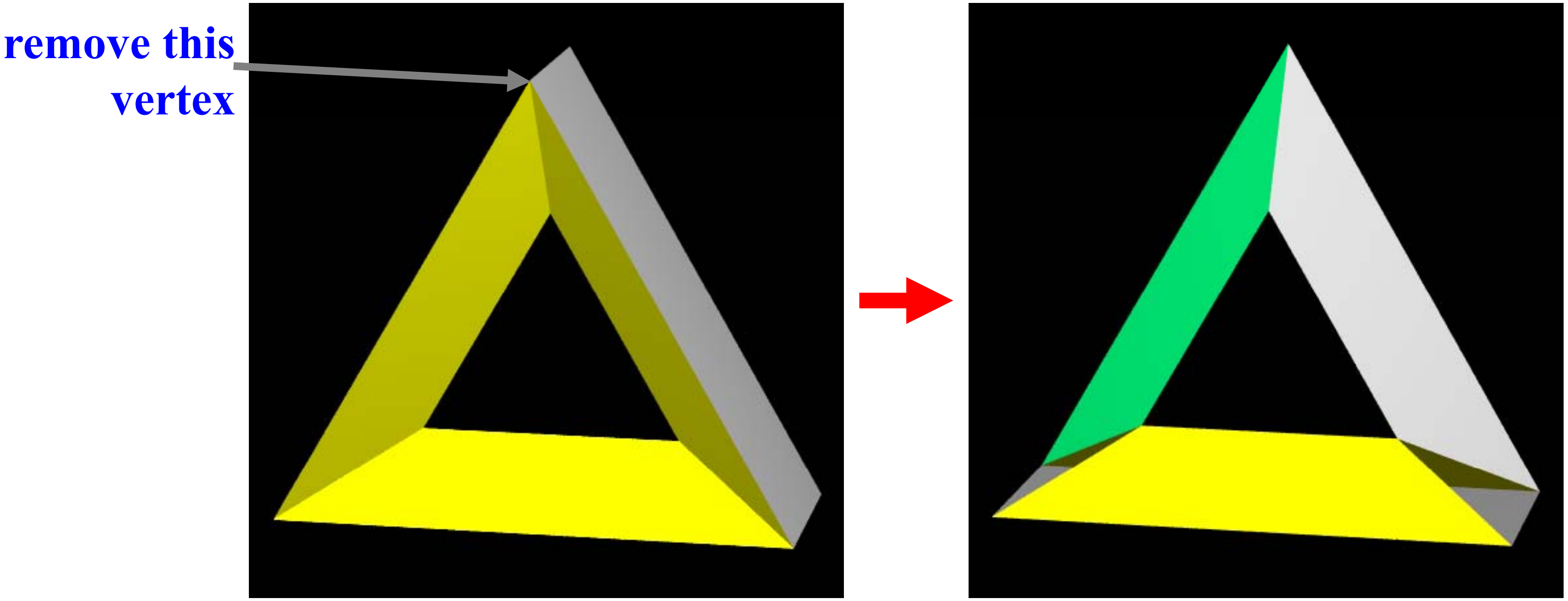
- ❖ If a mesh has holes like a torus, the boundary of a hole could reduce to a triangle (*i.e.*, triangular hole). Removing a vertex from the boundary could create a non-manifold.



A Vertex Decimation Algorithm for Triangular Mesh: 13/15

❑ A Few Notes: 4/5

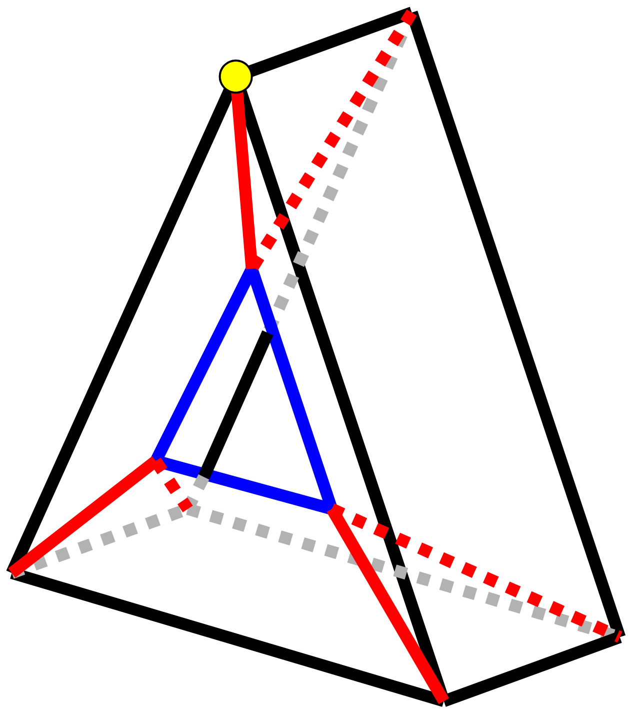
- ❖ **If a mesh has holes like a torus, the boundary of a hole could reduce to a triangle (*i.e.*, triangular hole). Removing a vertex from the boundary could create a non-manifold.**

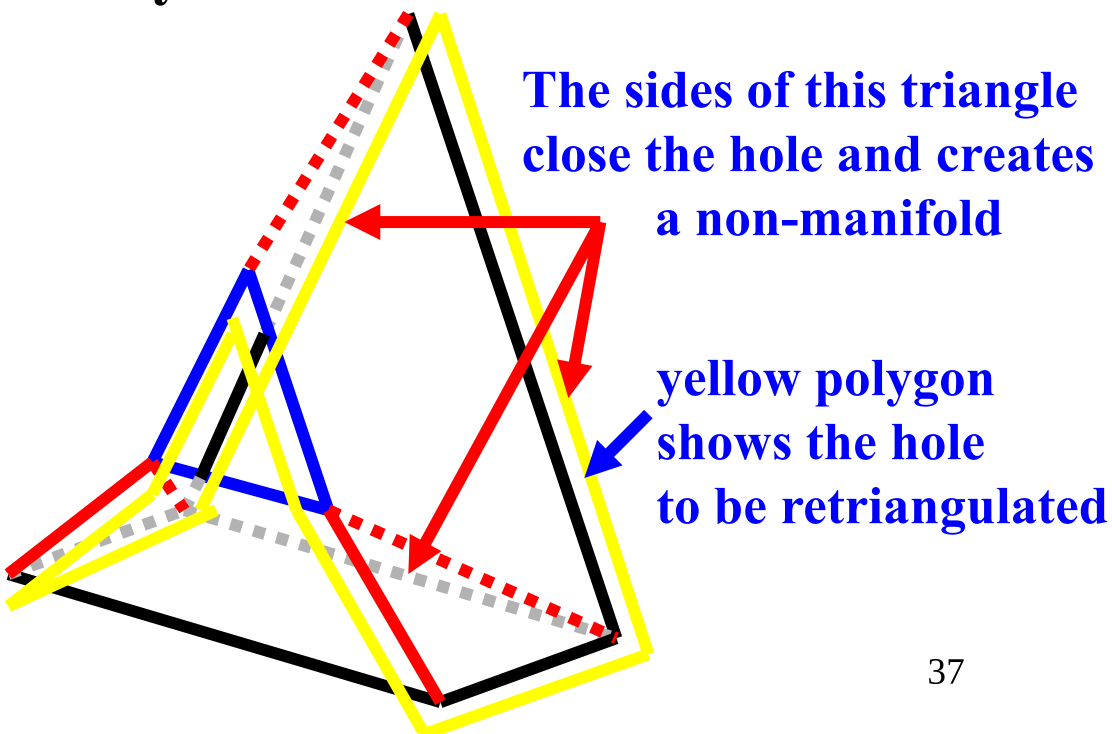


A Vertex Decimation Algorithm for Triangular Mesh: 14/15

A Few Notes: 5/5

- If a mesh has holes like a torus, the boundary of a hole could reduce to a triangle (*i.e.*, triangular hole). Removing a vertex from the boundary could create a non-manifold.

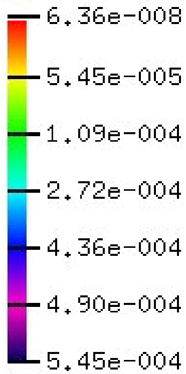




A Vertex Decimation Algorithm for Triangular Mesh: 15/15

- **A Few Notes: 4/4**
  - **Thus, in the decimation process, a check must be made to prevent duplicated triangles and triangle edges. In this way, the topology of the mesh can be preserved.**

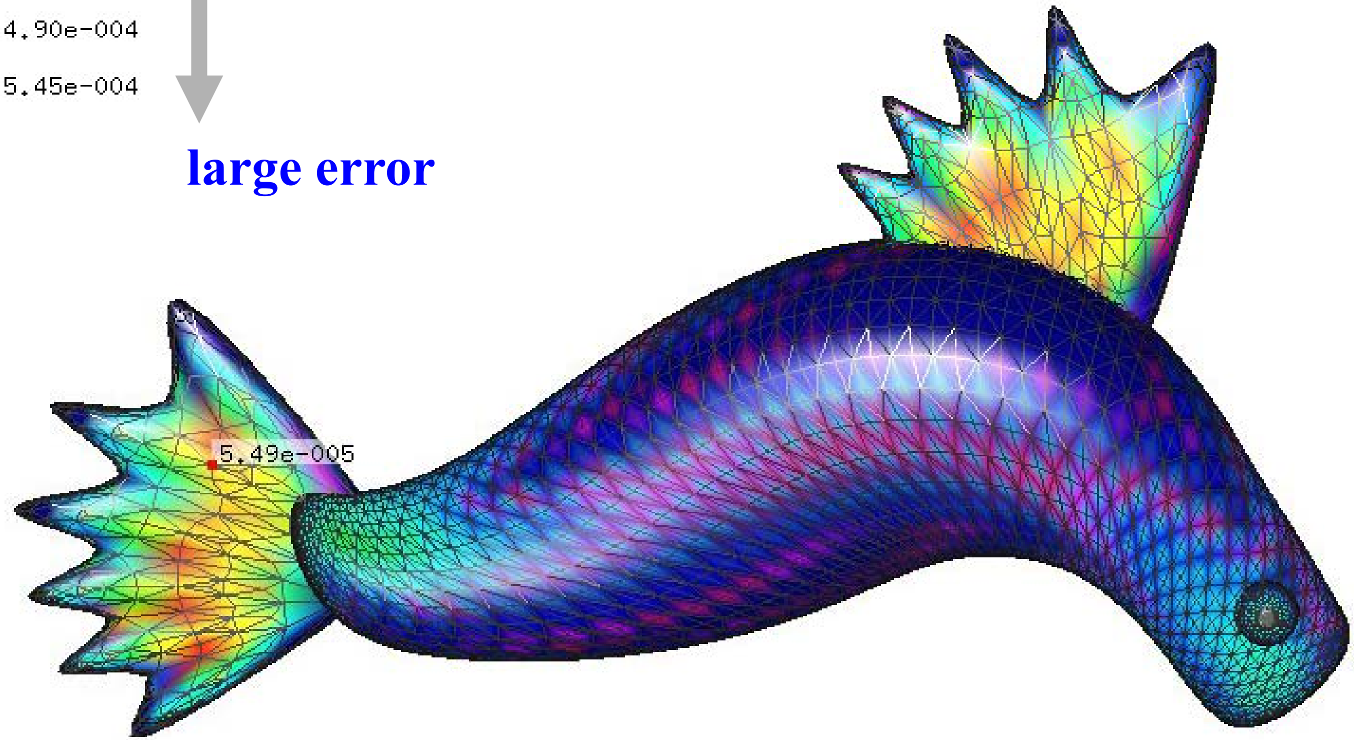
Vertex Error



small error

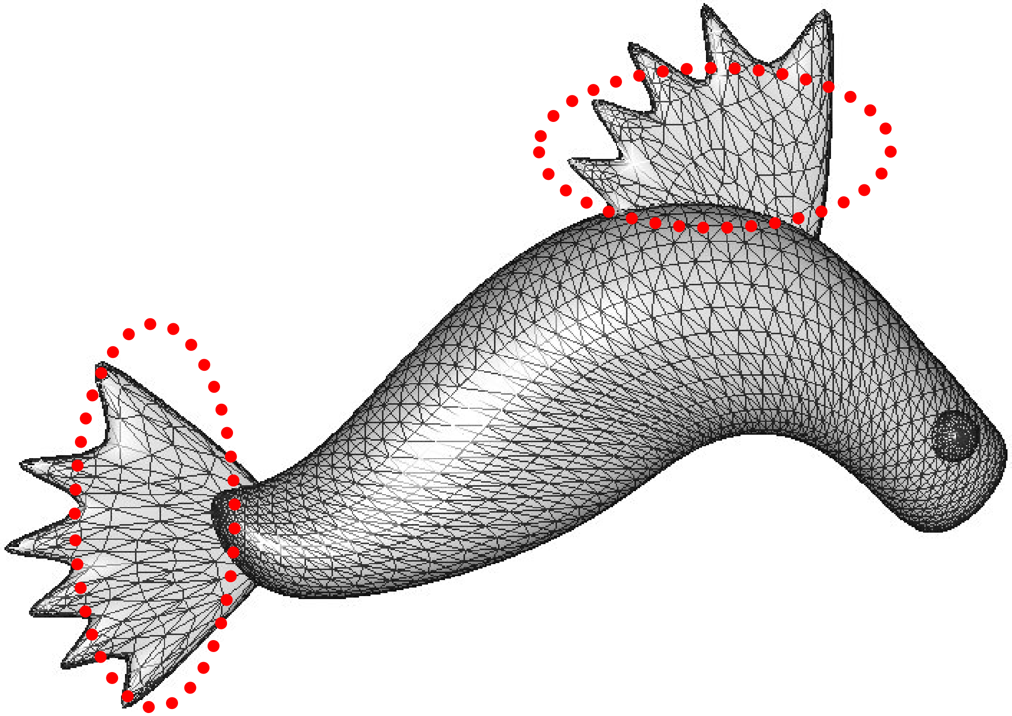
large error

Small vertex error means flat area, and can be simplified first.



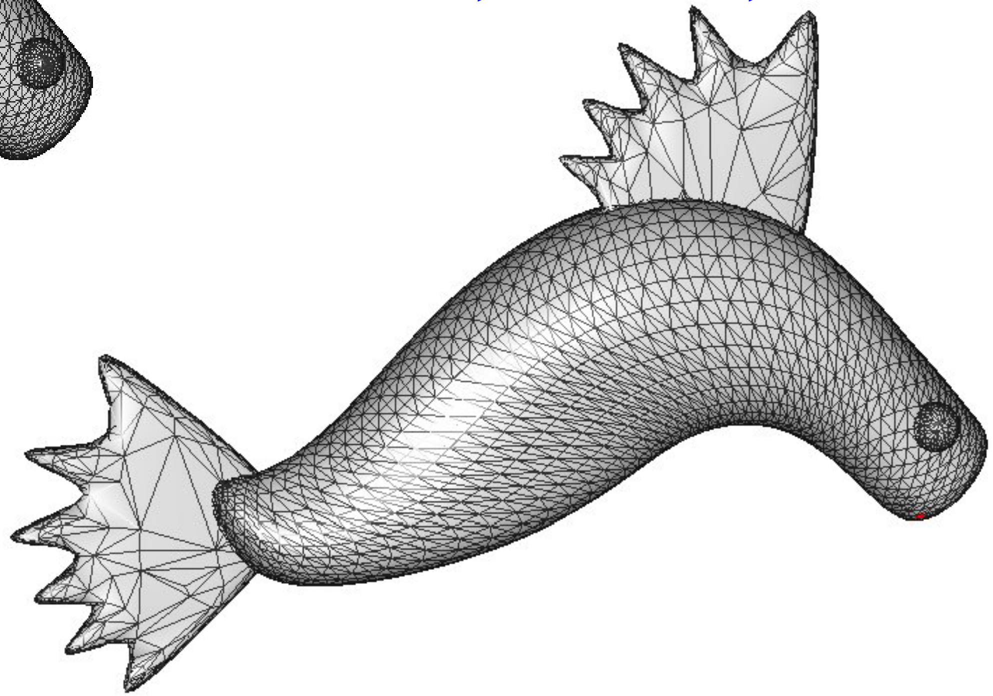
Some Results: 1/4

Note that flat portions are simplified first.



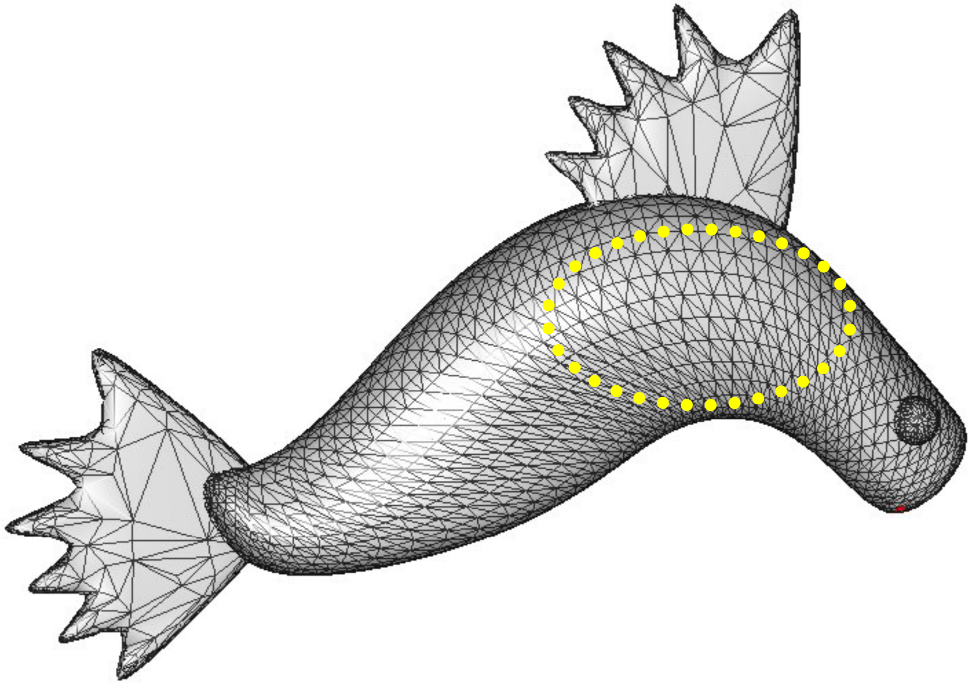
V=4632, E=13872, F=9248

V=3602, E=10776, F=7184



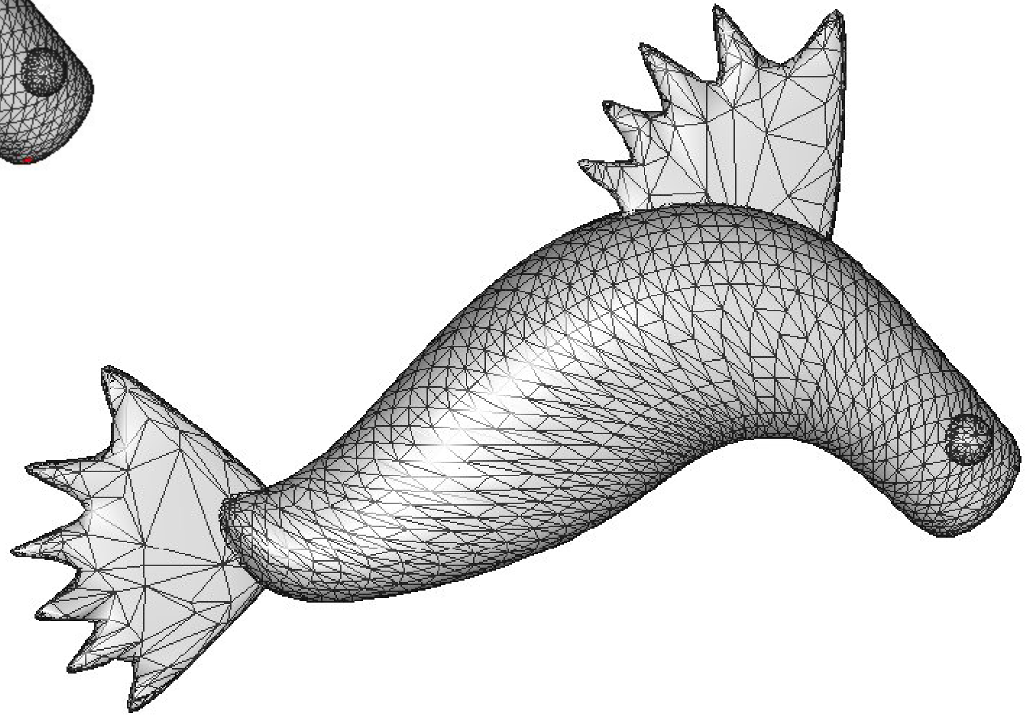
Some Results: 2/4

Note that flat portions are simplified first.



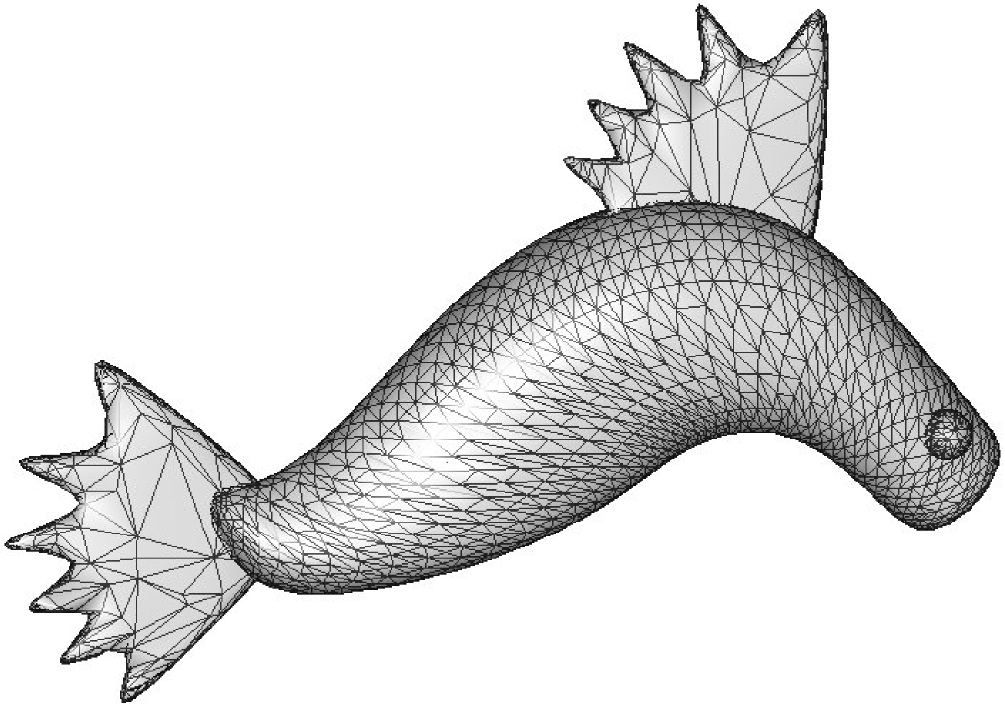
V=3602, E=10776, F=7184

V=2570, E=7680, F=5120



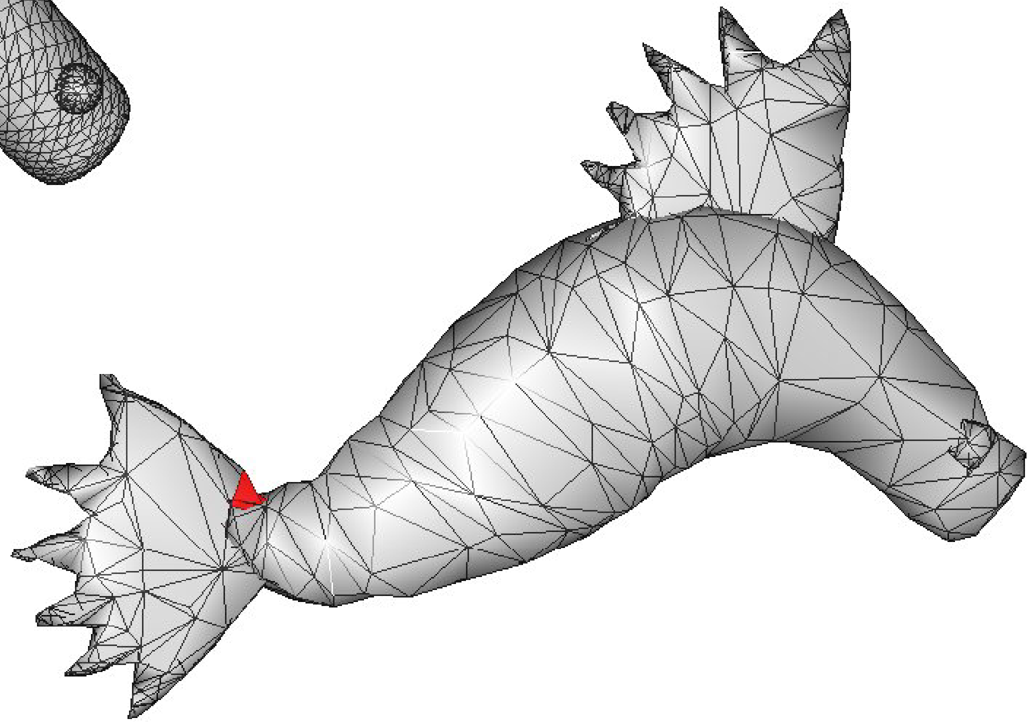
Some Results: 3/4

Note that flat portions are simplified first.



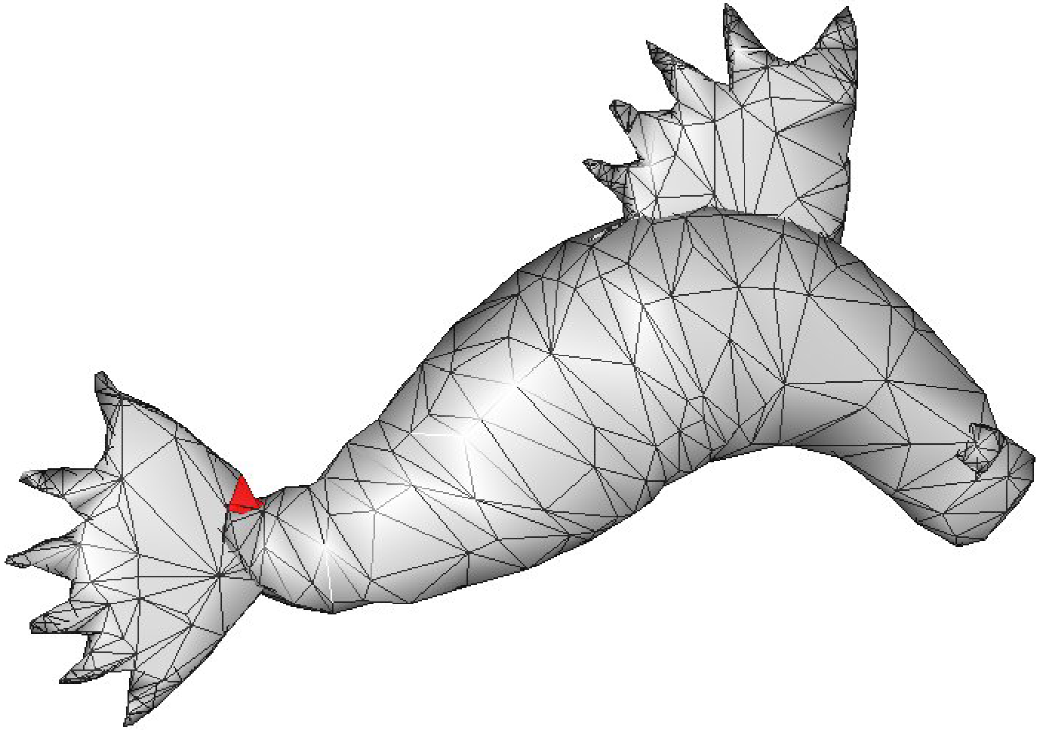
V=2570, E=7680, F=5120

V=506, E=1488, F=992



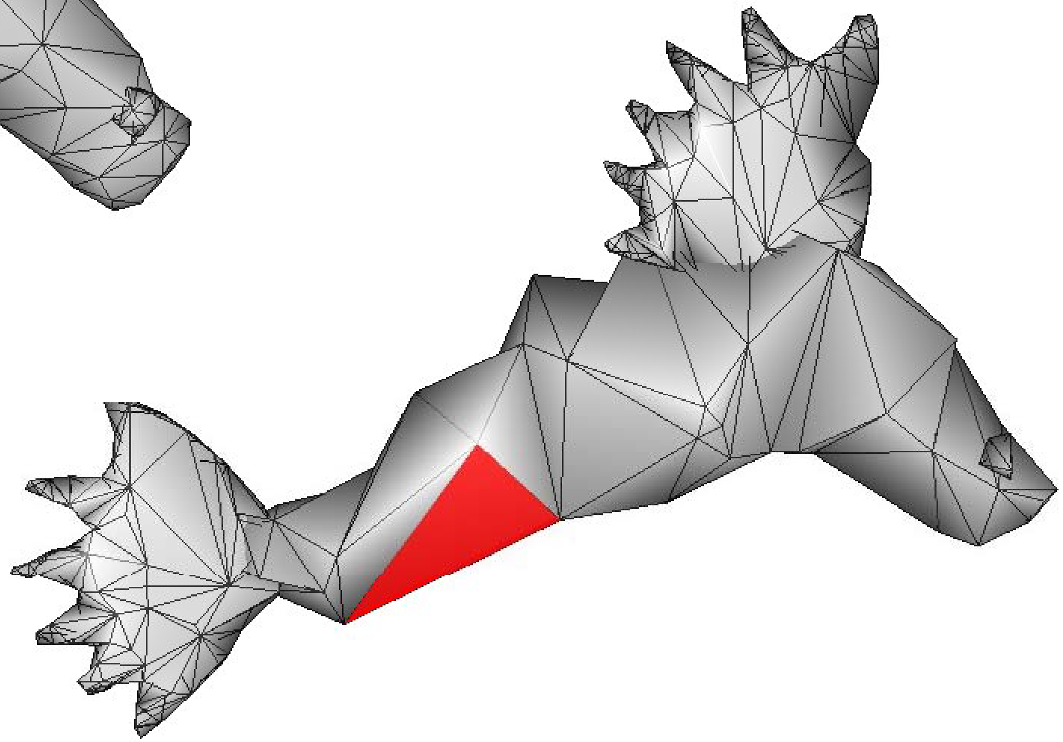
Some Results: 4/4

Note that flat portions are simplified first.



V=506, E=1488, F=992

V=306, E=888, F=592



Quadric Error Metric Decimation: 1/9

- **This algorithm is due to Michael Garland and Paul S. Heckbert, published in *IEEE Visualization 1998*.**
- **This algorithm uses the quadric error distance measure and the edge collapse operator.**
- **Each vertex of a given mesh is associated with an error metric, a 4×4 symmetric matrix, and a quadric (*i.e.*, second degree) error.**
- **For each edge, a new vertex with minimum error value (based on the error metric) is found and used for selecting an edge to be collapsed.**

Quadric Error Metric Decimation: 2/9

- **Since this algorithm uses edge collapse, we need a criterion for selecting an edge.**
- **Given two vertices, p and q, a pair (p,q) is a valid pair for collapsing, if**
  - **pq is an edge, or**
  - **$|p - q| < \varepsilon$, where ε is a user-defined constant**
- **If $\varepsilon > 0$, two very close vertices may be collapsed together (*i.e.*, vertex contraction), creating a non-manifold mesh. Thus, if vertex contraction is unwanted, set ε to 0!**

Quadric Error Metric Decimation: 3/9

- **What is an error metric?**
  - **It is a 4×4 symmetric matrix Q!**
  - **Each vertex v has an error metric matrix Q_v. We shall show how to find it later.**
  - **The error at a vertex $v = [v_1, v_2, v_3, 1]^T$, $\Delta(v)$, is defined as $v^T Q_v v$.**
  - **Since Q_v is a 4×4 matrix, $\Delta(v) = v^T Q_v v = \delta$ is a surface of second degree in v, where δ is a given value.**
  - **Hence, this error metric is referred to as a quadric error metric.**

Quadric Error Metric Decimation: 4/9

- **How do we collapse an edge?**
  - **If (p,q) is a valid pair, a simple way is to move p to q, move q to p, or move p and q to $(p+q)/2$.**
  - **However, there is a better way. We may move to a new point v that minimizes the error $\Delta(v) = (v^T Q_p v + v^T Q_q v)/2 = [v^T(Q_p+Q_q)v]/2$, where Q_p and Q_q are the error metric matrices of vertices p and q.**
  - **After v is computed, edge pq is collapsed and v receives the error value $\Delta(v)$ and error metric matrix Q_p+Q_q**

Quadric Error Metric Decimation: 5/9

compute error and error matrix for each vertex of the mesh;
select all valid edges pq such that $|p - q| < \varepsilon$;
for each selected edge pq do
begin
minimize $\Delta(r) = [r^T(Q_p + Q_q)r]/2$ to find r;
~~let $\Delta(r) = (\Delta(p) + \Delta(q))/2$ and~~ $Q_r = Q_p + Q_q$;
place all selected edges in a heap using $\Delta(r)$ as a key;
end;
while there are edges on the heap do
begin
remove the top edge pq;
collapse it to the computed r;
update the mesh and the keys;
end

Quadric Error Metric Decimation: 6/9

❑ How do we find Q_v for v, initially? 1/3

- ❖ Given a plane P: $ax+by+cz+d=0$, where $a^2+b^2+c^2=1$ (*i.e.*, normalized), and a point $v=(v_1,v_2,v_3)$, the error (*i.e.*, distance) from v to P is $\Delta_P(v) = av_1 + bv_2 + cv_3 + d$.
- ❖ Let $P = < a, b, c, d >$ and $v = < v_1,v_2,v_3,1 >$. Then, we have $\Delta_P(v) = av_1 + bv_2 + cv_3 + d = P \bullet v$.
- ❖ Thus, the error at v with respect to P is calculated by plugging v's coordinates into P's equation. If P•v is zero, v is in P. Otherwise, P•v gives the *signed* "distance" from v to P.

Quadric Error Metric Decimation: 7/9

- **How do we find Q_v for v, initially? 2/3**
  - **Since error may be negative, we use its square!**
  - **Since P•v can be rewritten into a matrix form $P^T v$, where P and v are row matrices, we have this**

$$
\begin{aligned}
(\mathbf{P}^T \cdot \mathbf{v})^2 &= (\mathbf{P}^T \cdot \mathbf{v})^T (\mathbf{P}^T \cdot \mathbf{v}) \\
&= (\mathbf{v}^T \cdot \mathbf{P})(\mathbf{P}^T \cdot \mathbf{v}) \\
&= \mathbf{v}^T (\mathbf{P}\mathbf{P}^T)\mathbf{v} \\
&= \mathbf{v}^T \underbrace{\begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}}_{M_P(\mathbf{v})} \mathbf{v}
\end{aligned}
$$

Quadric Error Metric Decimation: 8/9

- **How do we find Q_v for v, initially? 3/3**
  - **The error metric matrix of v w.r.t. P is the matrix shown earlier rather than the error value itself! Let this matrix be $M_P(v)$.**
  - **Now, for each vertex v in the given mesh, the error metric matrix of vertex v is the sum of all $M_P(v)$, where P is a plane that contains an incident triangle of v:**

$$Q_{\mathbf{v}} = \sum_{\text{all } \mathbf{P}\text{'s incident to } \mathbf{v}} M_{\mathbf{P}}(\mathbf{v})$$

Quadric Error Metric Decimation: 9/9

- **How do we find a v that minimizes v^TQ_vv?**
  - **Once Q_v is computed from Q_p and Q_q, where pq is the edge to be collapsed, we need to find a new vertex v such that v^TQ_vv is minimized.**
  - **Since v^TQ_vv is a second degree function in v, its minimum can easily be found. Compute and set the partial derivatives of v^TQ_vv to zero, and solve for x, y and z!**

The Minimum of a Quadric Function

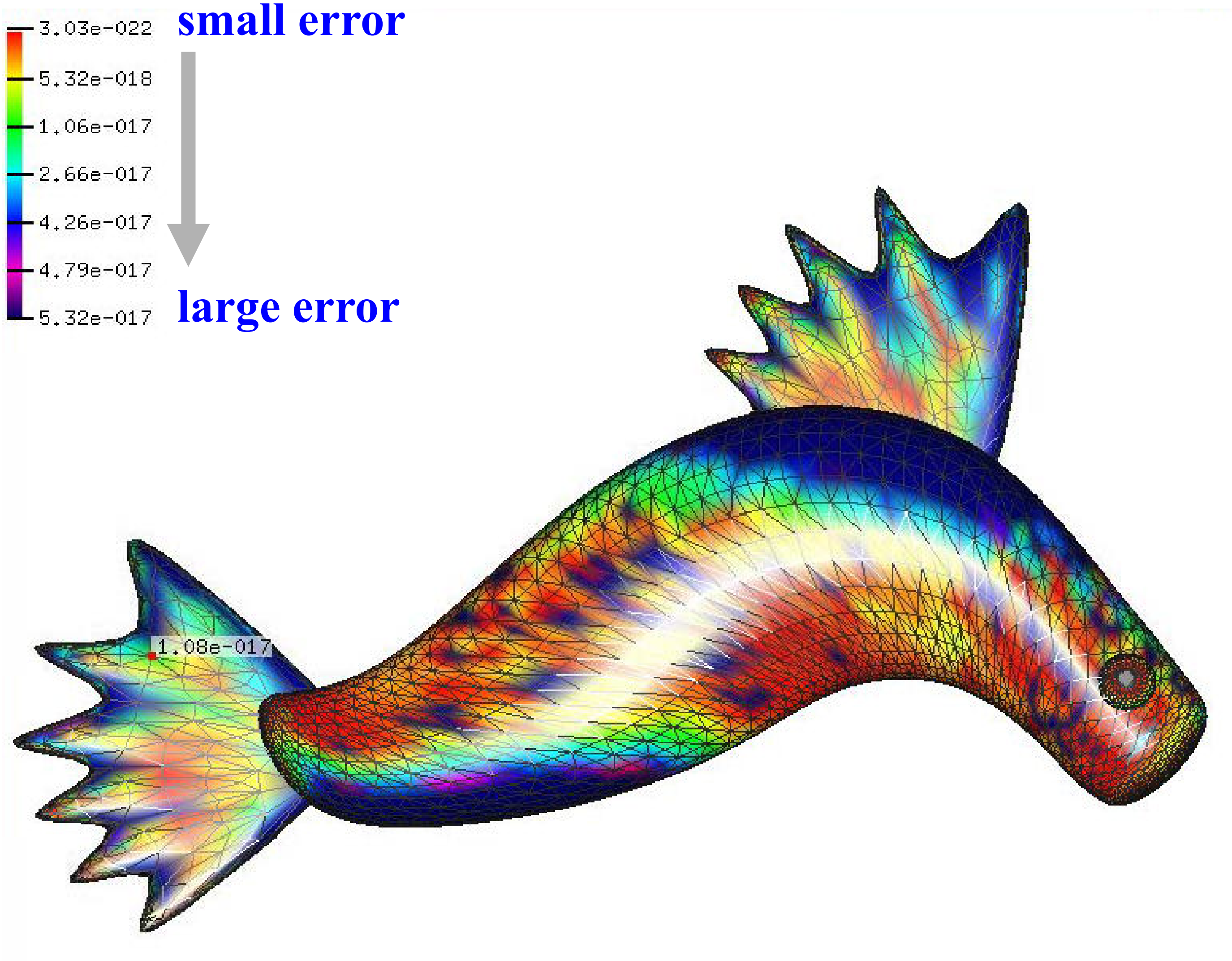
- The vector **v** in the function $\mathbf{v}^T Q_v \mathbf{v}$ has three variables, *i.e.*, $\mathbf{v} = (x, y, z)$, and the function itself is of second degree.
- Therefore, function $\mathbf{v}^T Q_v \mathbf{v}$ has a form of

$$F(x, y, z) = ax^2 + by^2 + cz^2 + 2dxy + 2exz + 2fyz + 2gx + 2hy + 2iz + j$$

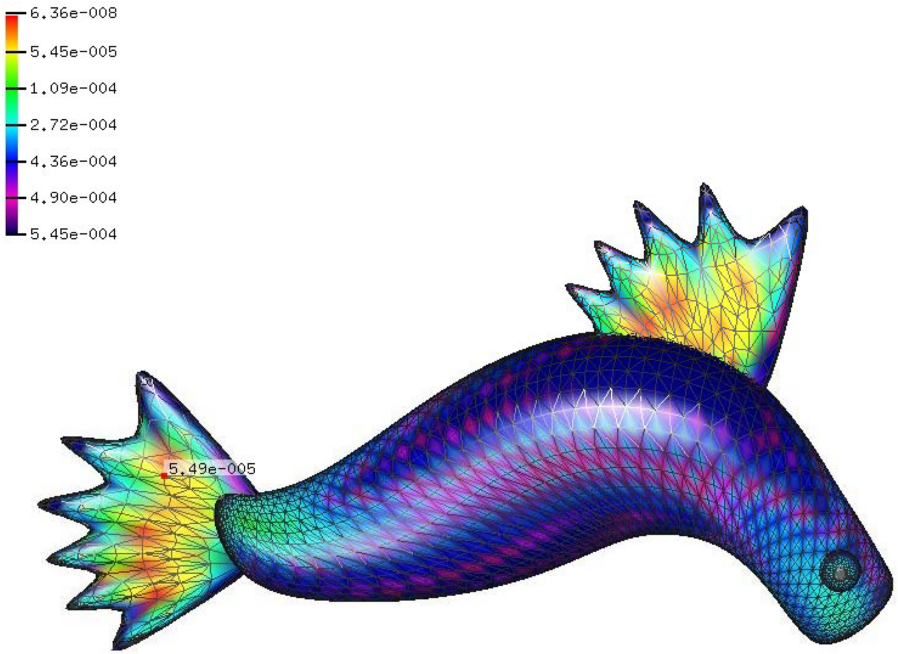
- Setting the partial derivatives to zero and solving for x, y and z yield the vector **v**.

$$\begin{aligned}
\frac{\partial F}{\partial x} &= ax + dy + ez + g = 0 \\
\frac{\partial F}{\partial y} &= dx + by + fz + h = 0 \\
\frac{\partial F}{\partial z} &= ex + fy + cz + i = 0
\end{aligned}$$

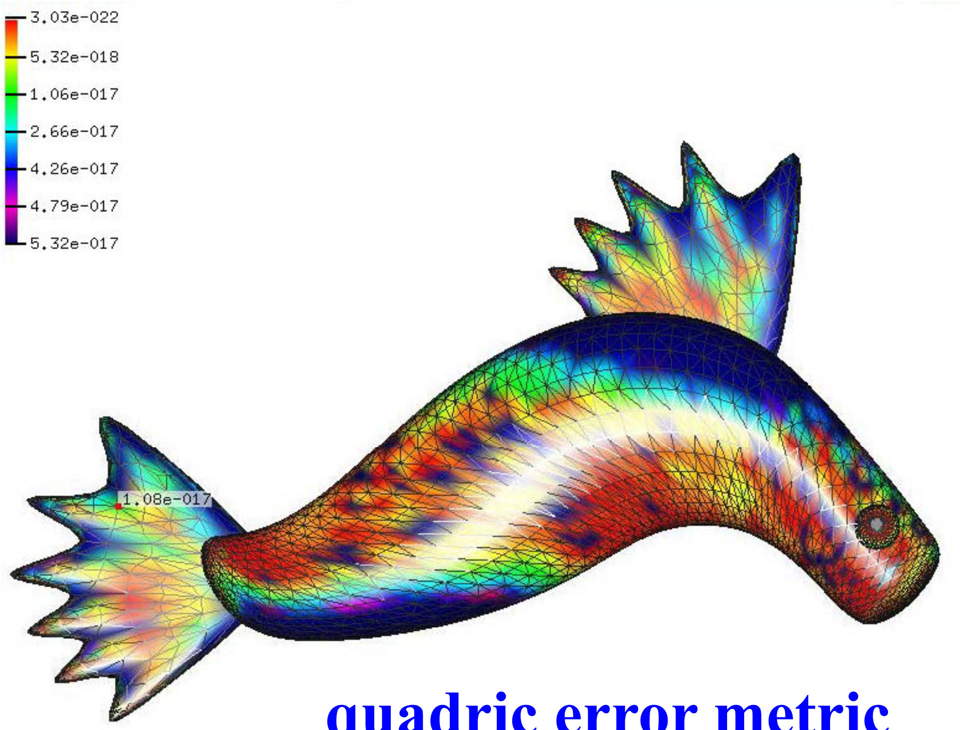
Quadric Error Metric



Decimation vs Error Metric

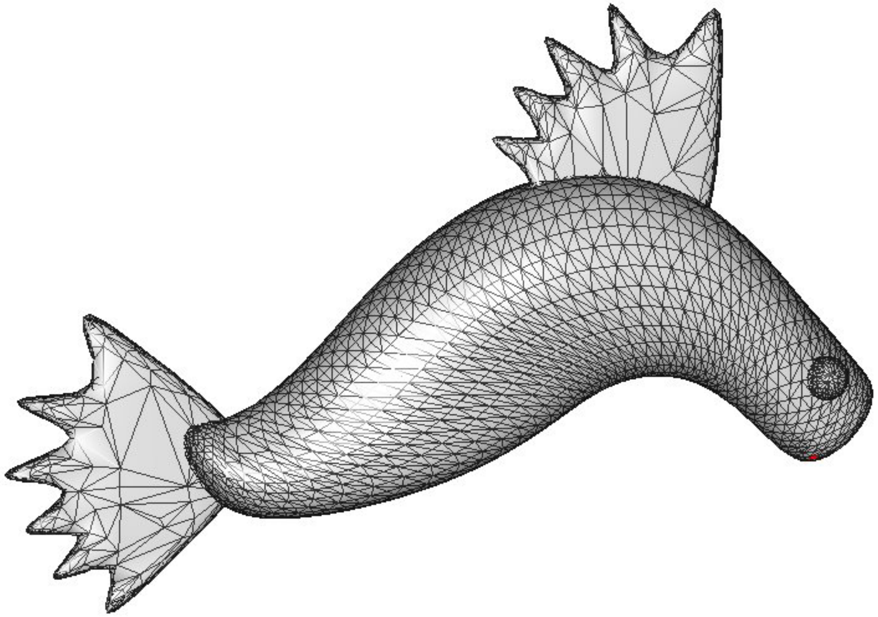


vertex decimation



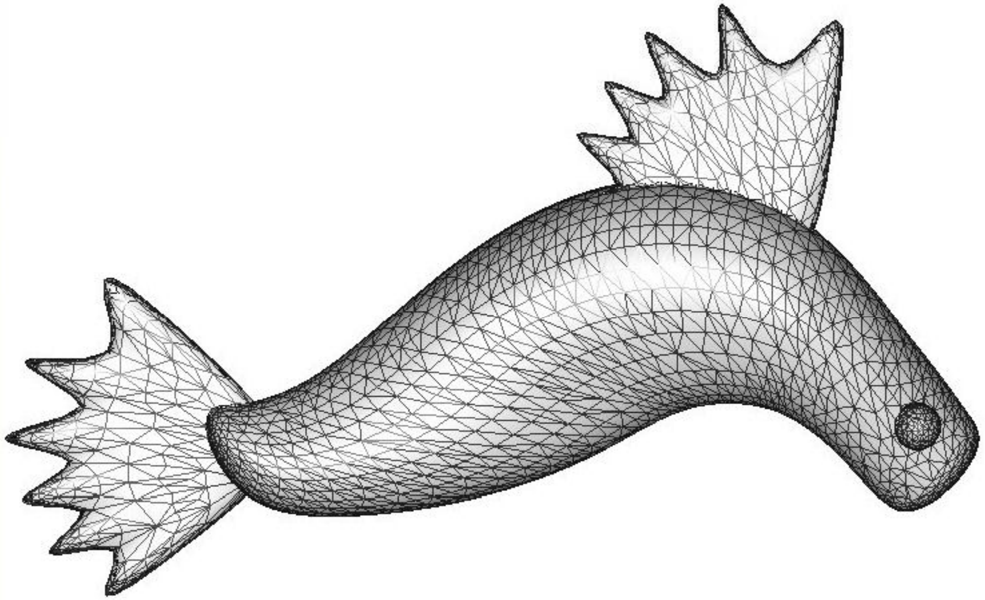
quadric error metric

Results and Comparisons: 1/3

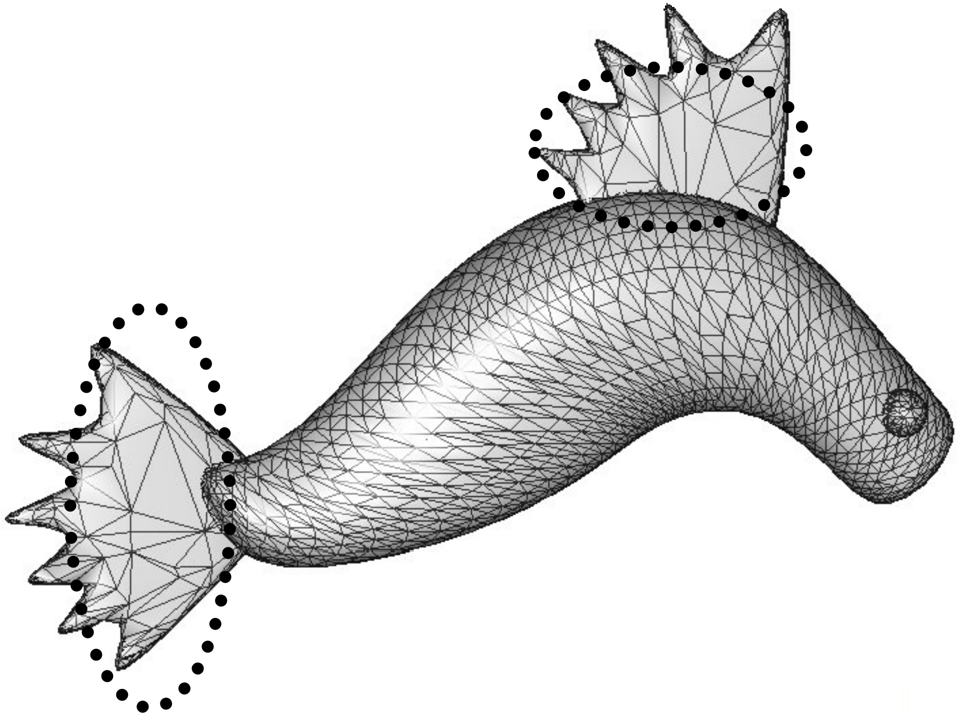


vertex decimation: V=3602

error metric: V=3555

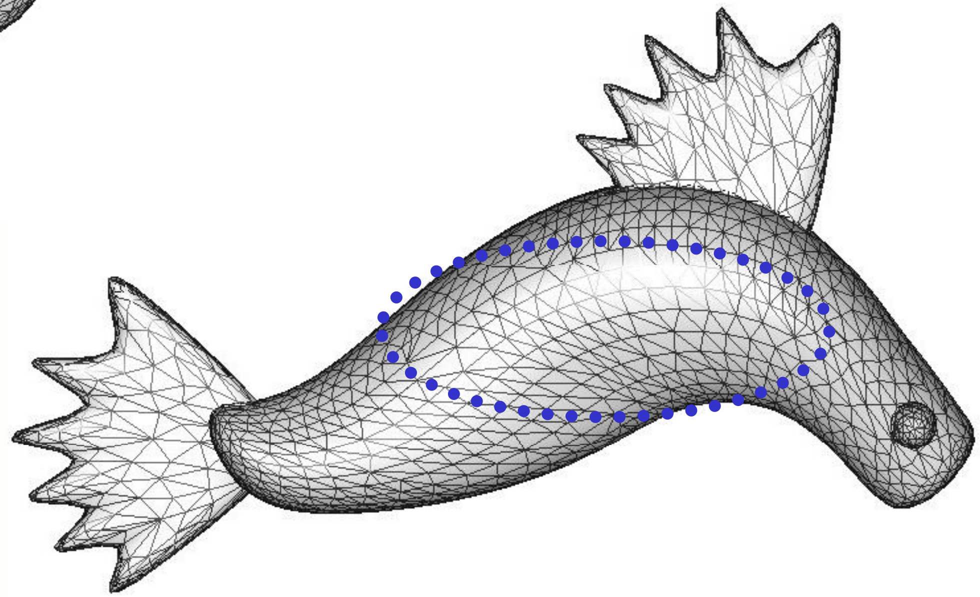


Results and Comparisons: 2/3

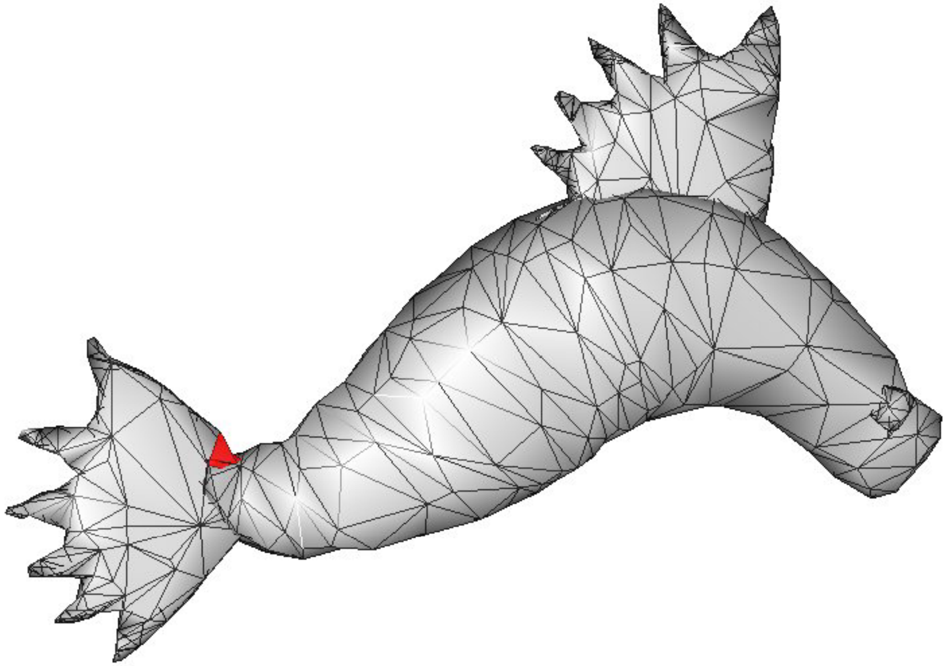


vertex decimation: V=2570

error metric: V=2476

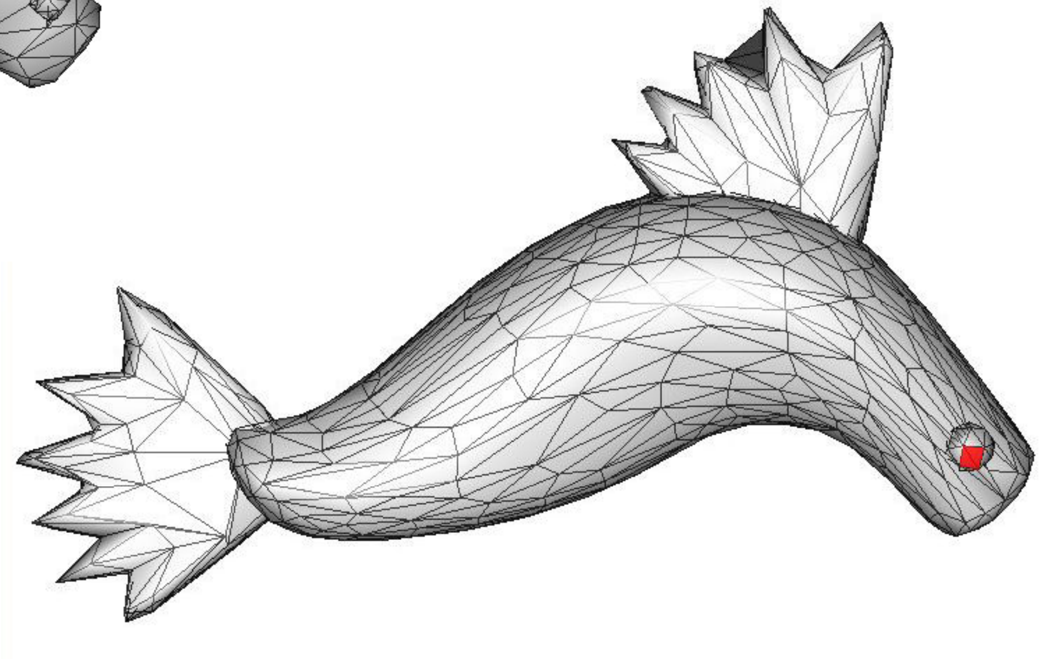


Results and Comparisons: 3/3

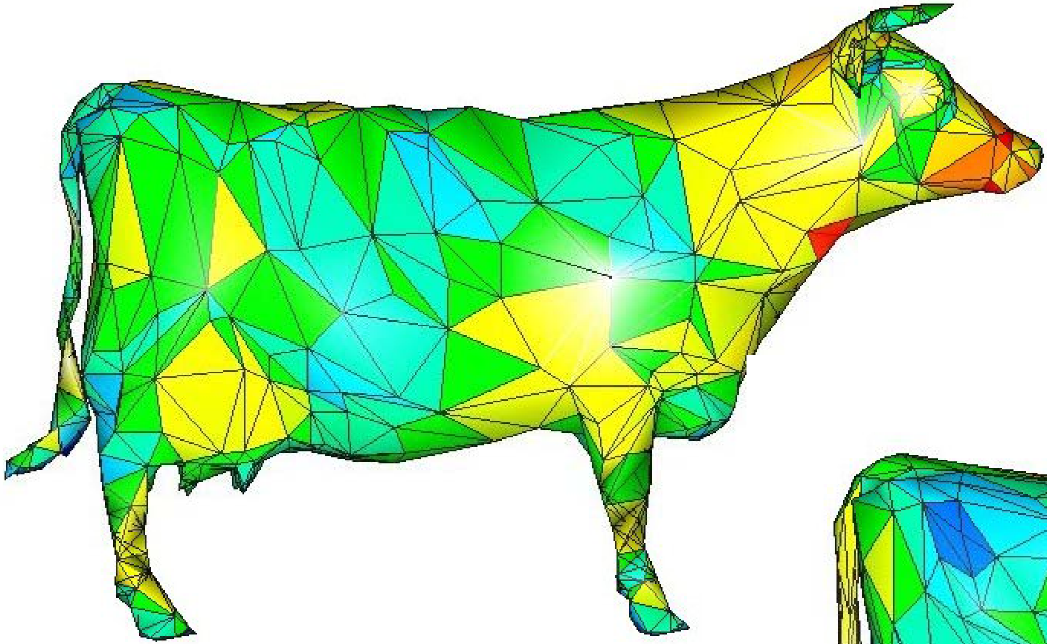


error metric: V=599

vertex decimation: V=1538

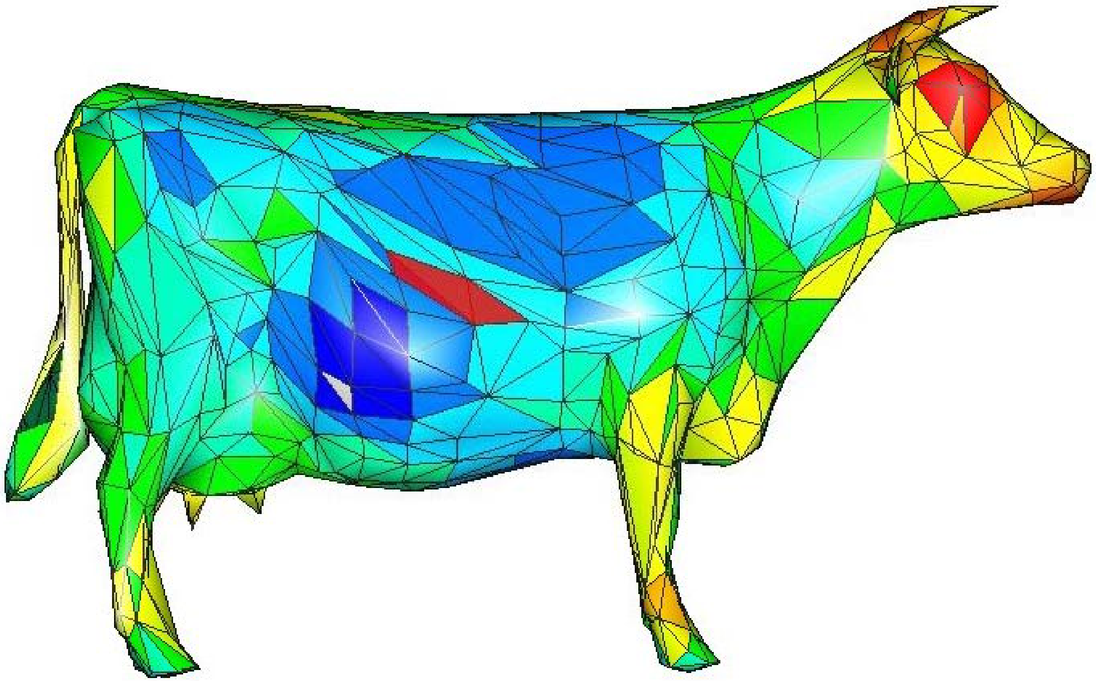


More Comparisons: 1/2



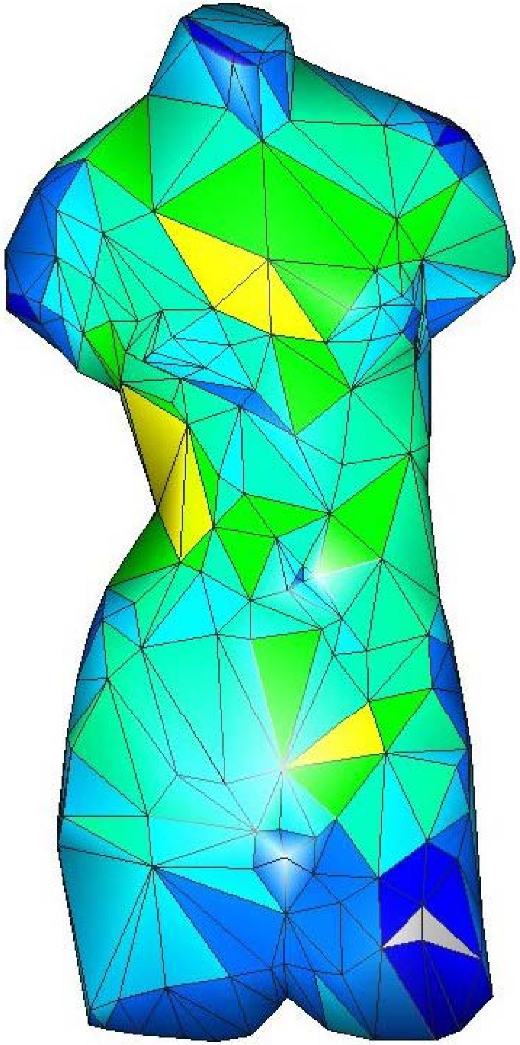
vertex decimation:
V=703,E=2106,F=1401

error metric:
V=703,E=2106,F=1398



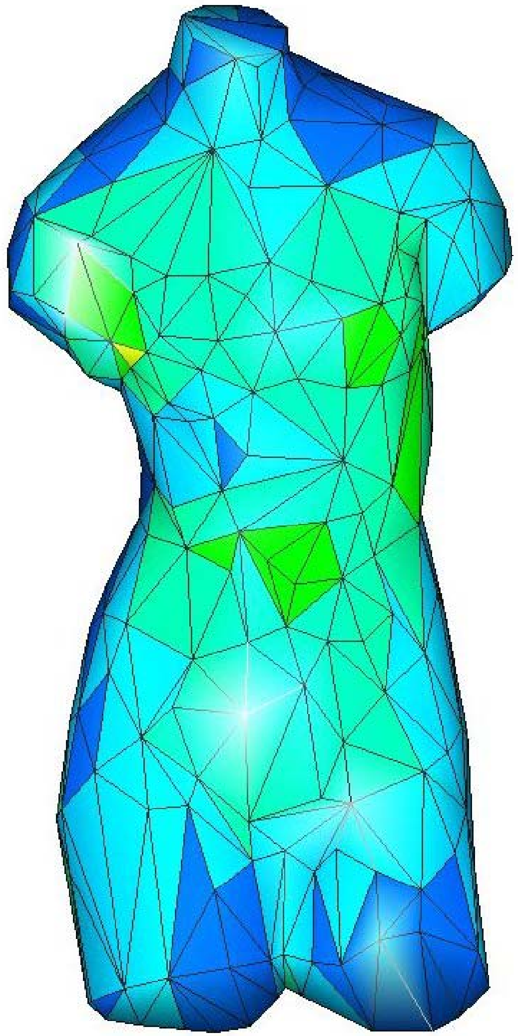
More Comparisons: 2/2

vertex decimation



V=311
E=927
F=618

error metric



The End