CSE 554

Lecture 7: Simplification

Fall 2016

Geometry Processing

Washington University in St. Louis
School of Engineering & Applied Science

- Fairing (smoothing)
  - Relocating vertices to achieve a smoother appearance
  - Method: centroid averaging



Why do we care?

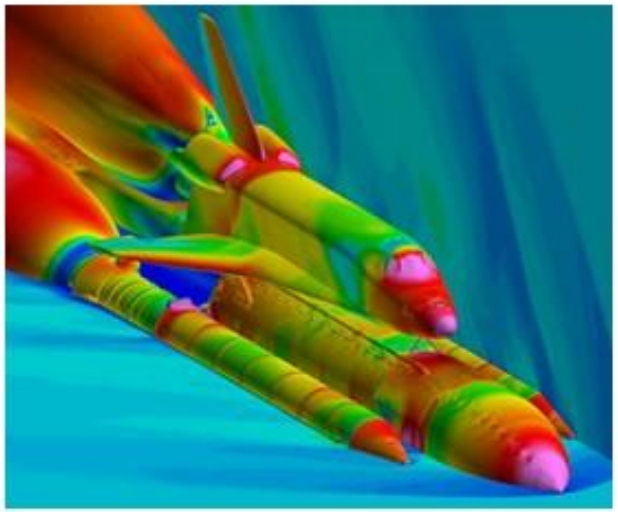
- Polygon count is a major performance factor
  - Animation (deformation), rendering, simulation, interaction



Character animation
(Pixar)



Ray tracing rendering
(POV-Ray)



Space shuttle aerodynamics
(NASA)

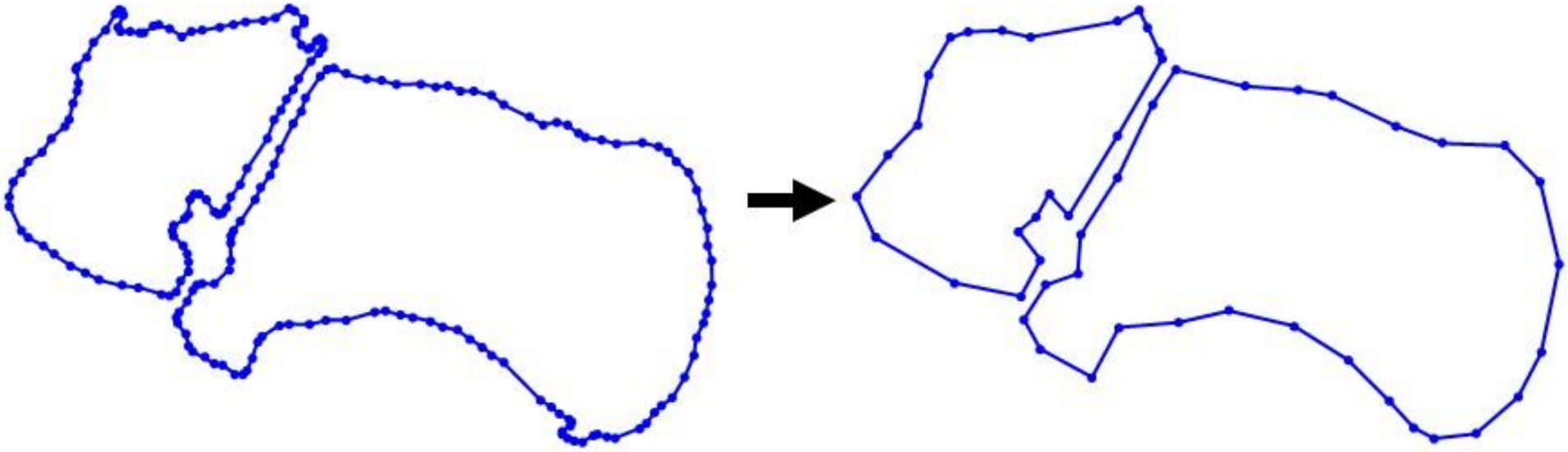
Simplification (2D)

- Replacing a 2D polygon by another with fewer vertices
  - Preserves the shape and topology of the input polygon

Simplification (2D)

Washington University in St. Louis
School of Engineering & Applied Science

- An optimization problem:
  - Given an input polygon and a target vertex count N
  - Find the coordinates of N vertices that minimize the distance from the new polygon formed by these vertices and the input polygon

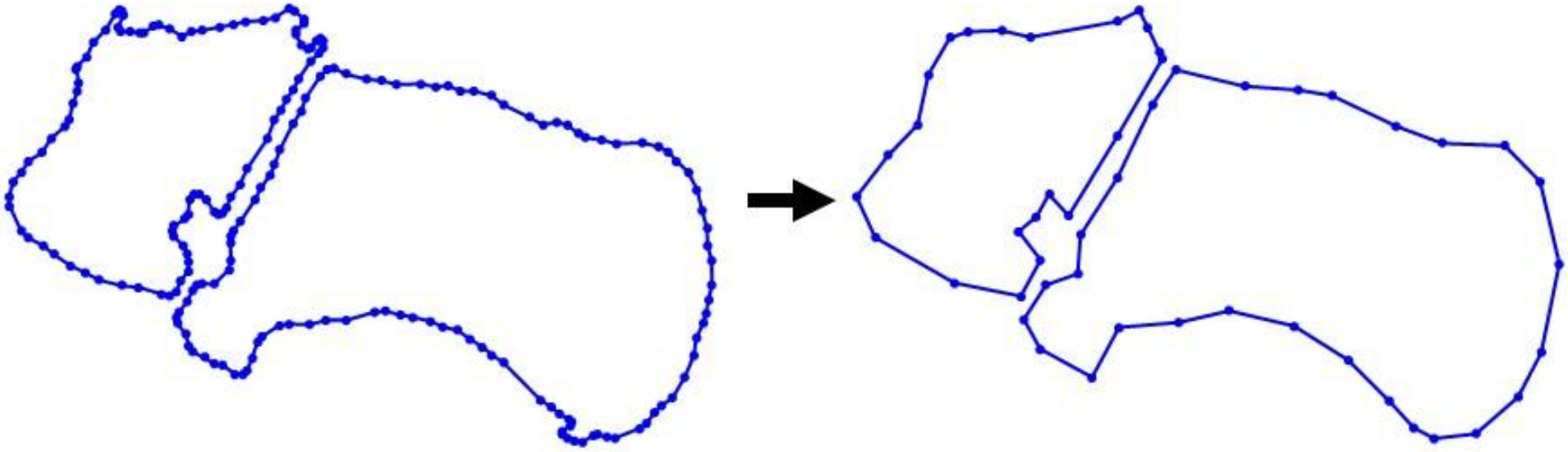


200 vertices

50 vertices

Simplification (2D)

- A greedy heuristic:
  - Iteratively merges two neighboring vertices into one, until the target vertex count is reached
  - Each merging minimizes the error to the original polygon



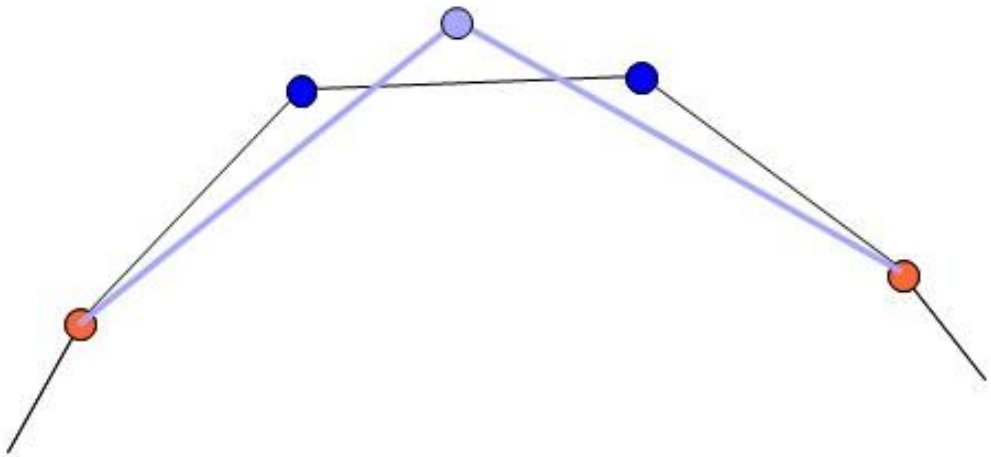
200 vertices

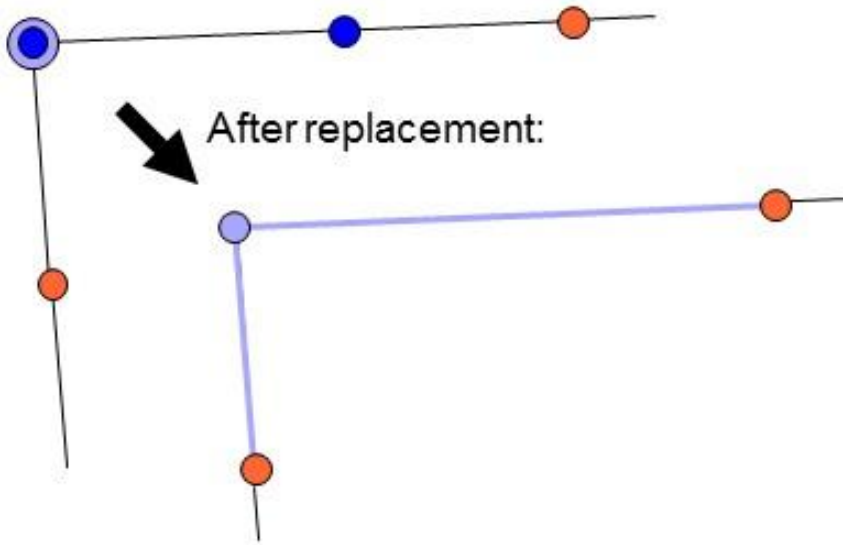
50 vertices

Simplification (2D)

Washington University in St. Louis
School of Engineering & Applied Science

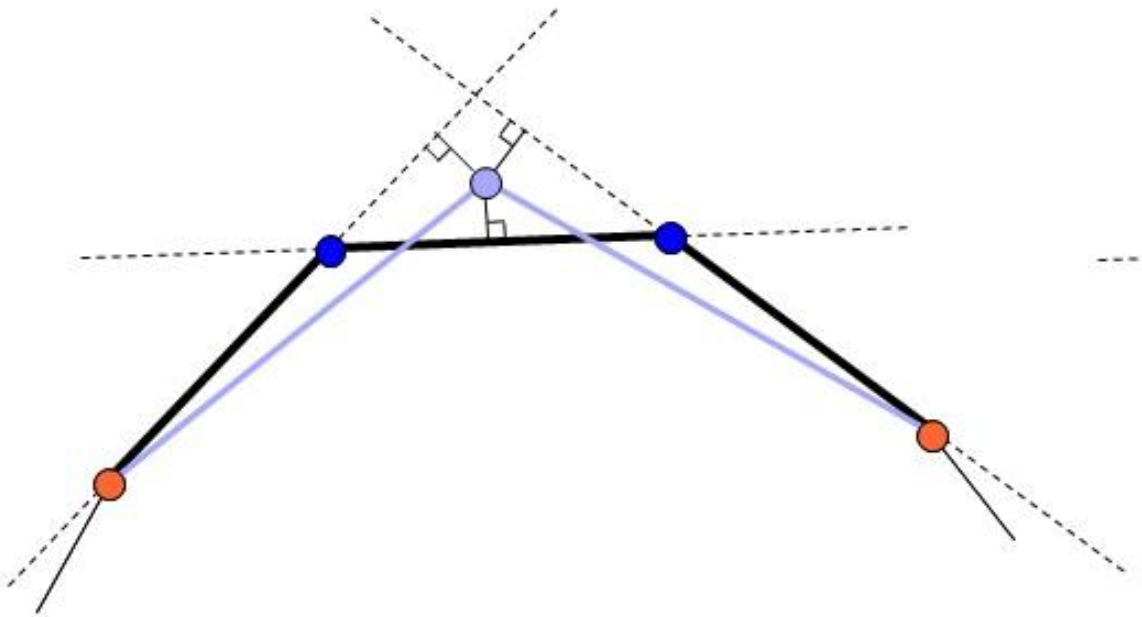
- If I want to replace two vertices with one, where should it be?

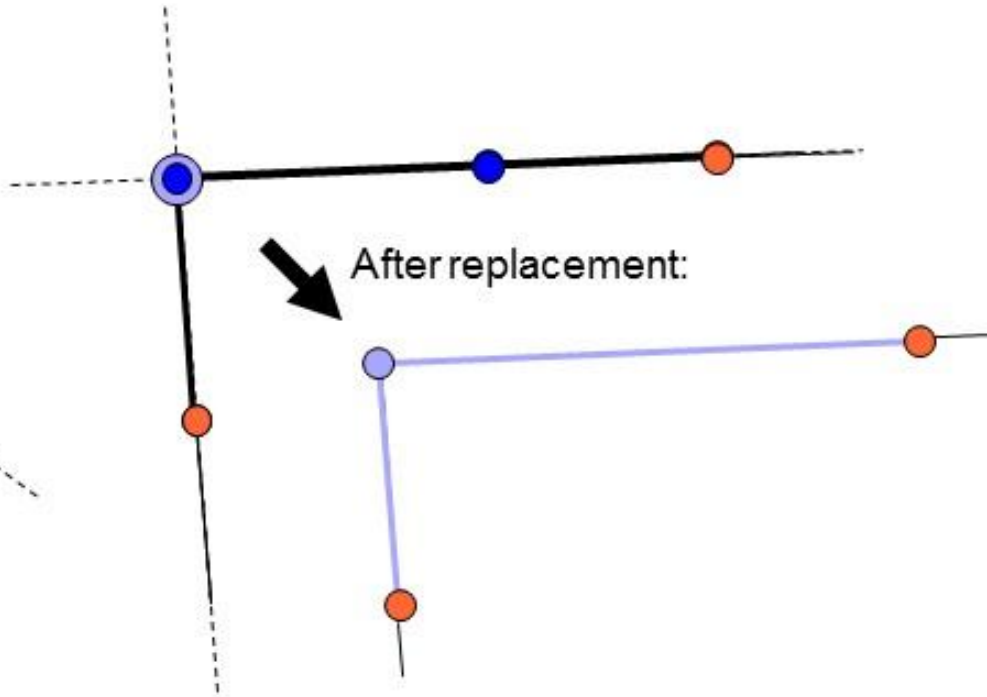




Simplification (2D)

- If I want to replace two vertices with one, where should it be?
  - Shortest distances to the supporting lines of involved edges

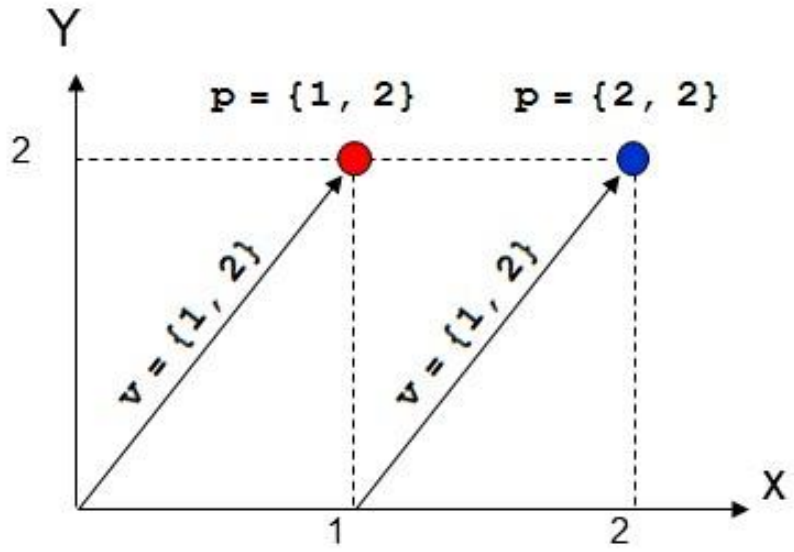




Points and Vectors

- Same representation

```
{x, y} (or {x, y, z})
```



- Different meaning:
  - Point: a fixed **location** (relative to {0,0} or {0,0,0})
  - Vector: a **direction** and **magnitude**
    - No location (any location is possible)

Point Operations

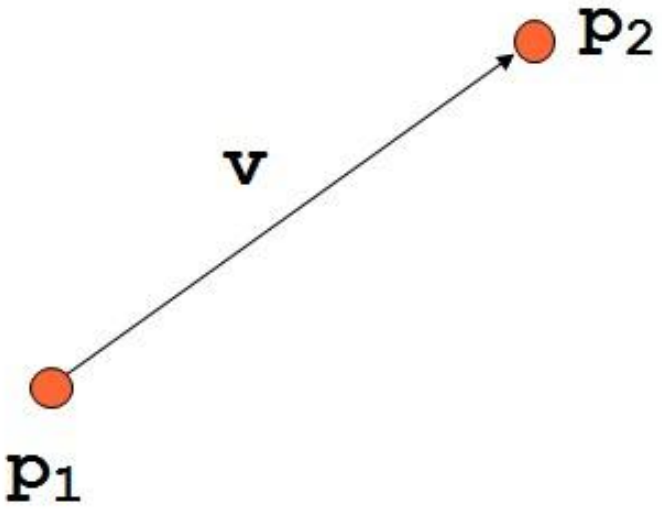
- Subtraction
  - Result is a vector

$$p_2 - p_1 = v = \{p_{2_x} - p_{1_x}, p_{2_y} - p_{1_y}\}$$

- Addition with a vector
  - Result is a point

$$p_1 + v = p_2 = \{p_{1_x} + v_x, p_{1_y} + v_y\}$$

- Can points add?



Adding Points

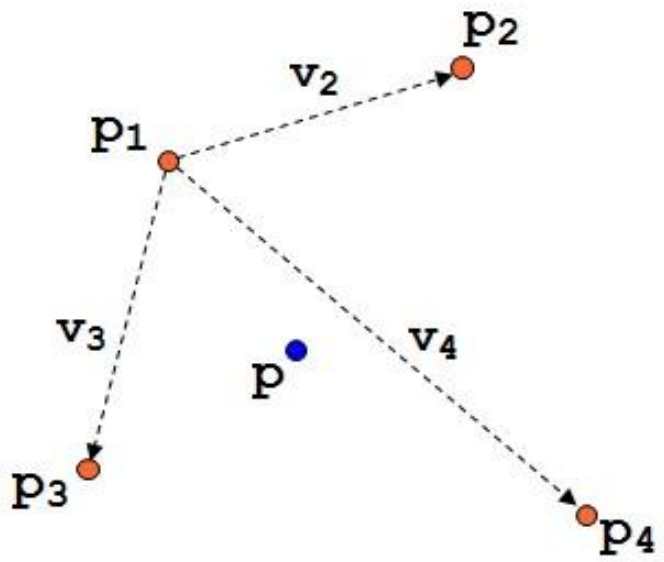
- Affine combinations
  - Weighted addition of points where all weights sum to 1

$$\sum_{i=1}^{n} w_i p_i = p, \quad \text{where} \sum_{i=1}^{n} w_i = 1$$

  - Result is another point
    - Same as adding scaled vectors to a point

$$\sum_{i=1}^{n} w_i p_i = \sum_{i=1}^{n} w_i (p_i - p_1) + \sum_{i=1}^{n} w_i p_1$$

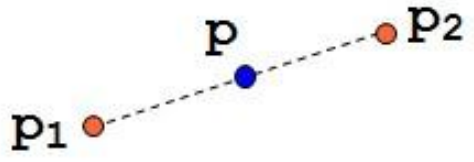
$$= \sum_{i=1}^{n} w_i v_i + p_1$$



Adding Points

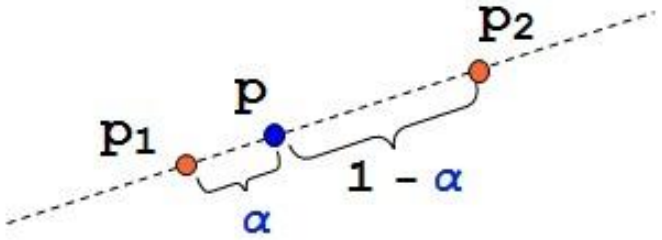
- Affine combinations: examples
  - Mid-point of two points

$$\mathbf{p} = \frac{\mathbf{p}_1 + \mathbf{p}_2}{2}$$



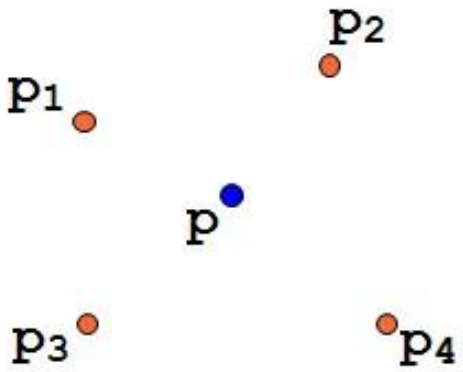
  - Linear interpolation of two points

$$\mathbf{p} = (1 - \alpha)\,\mathbf{p}_1 + \alpha\,\mathbf{p}_2$$



  - Centroid of multiple points

$$\mathbf{p} = \sum_{i=1}^{n} \frac{\mathbf{p}_i}{n}$$



Vector Operations

- Addition/Subtraction
  - Result is a vector

$$\mathbf{v}_1 \pm \mathbf{v}_2 = \left\{ \mathbf{v}_{1_x} \pm \mathbf{v}_{2_x}, \mathbf{v}_{1_y} \pm \mathbf{v}_{2_y} \right\}$$

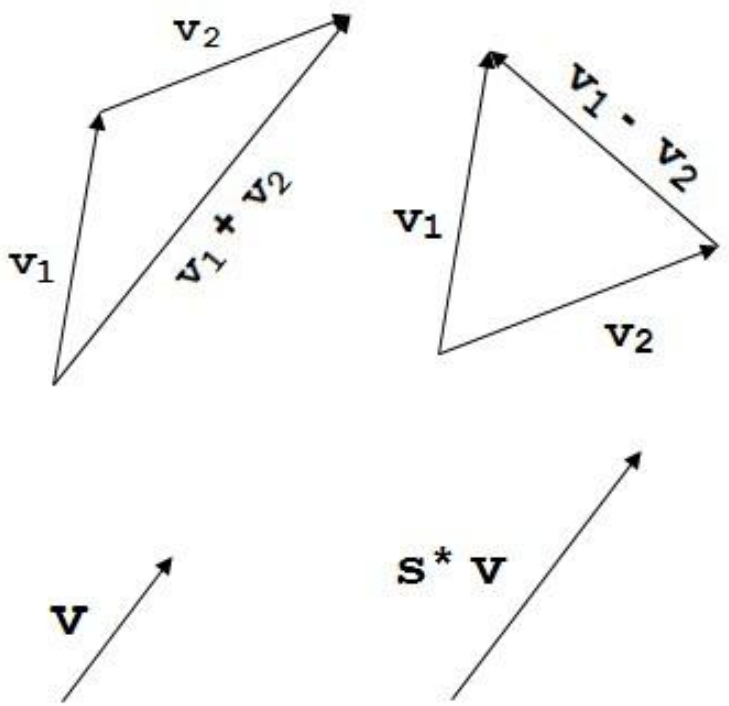
- Scaling by a scalar
  - Result is a vector

$$s * \mathbf{v} = \{ s * \mathbf{v}_x, s * \mathbf{v}_y \}$$

- Magnitude
  - Result is a scalar

$$| \mathbf{v} | = \sqrt{\mathbf{v}_x^2 + \mathbf{v}_y^2}$$

    - A unit vector: $| \mathbf{v} | = 1$
    - To make a unit vector (normalization): $\frac{\mathbf{v}}{| \mathbf{v} |}$

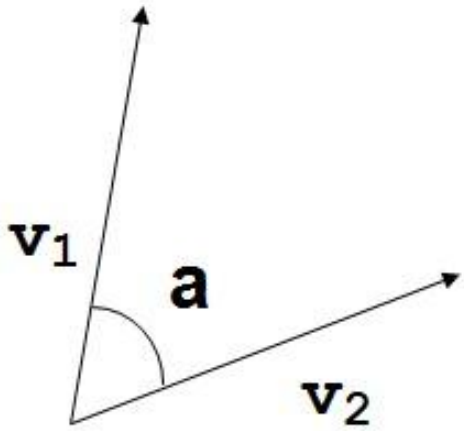


More Vector Operations

- Dot product (in both 2D and 3D)
  - Result is a scalar

    $\mathbf{v_1} \cdot \mathbf{v_2} = |\mathbf{v_1}||\mathbf{v_2}| \cos[\alpha]$

  - In coordinates (simple!)
    - 2D: $\mathbf{v_1} \cdot \mathbf{v_2} = \mathbf{v_{1_x}} * \mathbf{v_{2_x}} + \mathbf{v_{1_y}} * \mathbf{v_{2_y}}$
    - 3D: $\mathbf{v_1} \cdot \mathbf{v_2} = \mathbf{v_{1_x}} * \mathbf{v_{2_x}} + \mathbf{v_{1_y}} * \mathbf{v_{2_y}} + \mathbf{v_{1_z}} * \mathbf{v_{2_z}}$
    - Matrix product between a row and a column vector

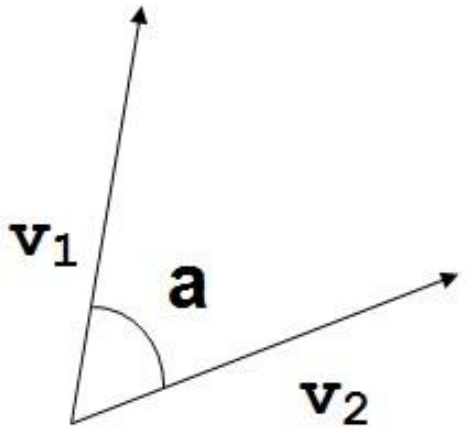


More Vector Operations

- Uses of dot products
  - Angle between vectors:

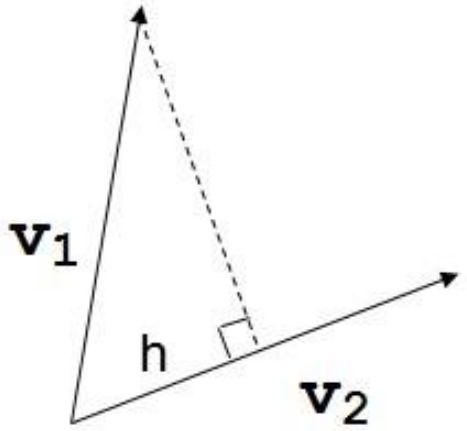
$$\alpha = \text{ArcCos}\left[\frac{v_1 \cdot v_2}{|v_1| * |v_2|}\right]$$

    - Orthogonal test: $v_1 \cdot v_2 = 0$



  - Projected length of v_1 onto v_2 :

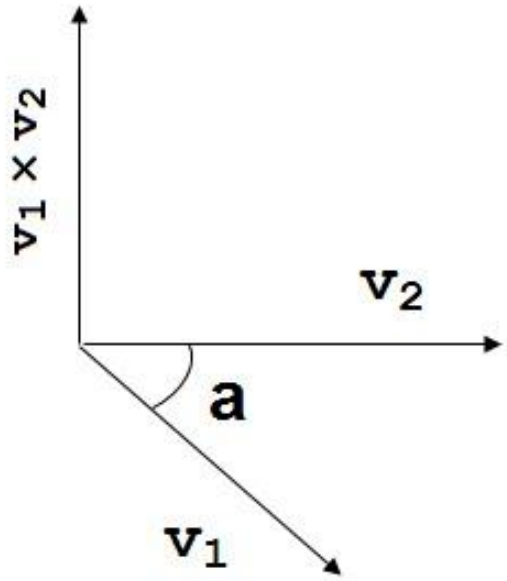
$$h = \frac{v_1 \cdot v_2}{|v_2|}$$



More Vector Operations

- Cross product (only in 3D)
  - Result is another 3D vector
    - Direction: Normal to the plane where both vectors lie (right-hand rule)
    - Magnitude: $| \mathbf{v}_1 \times \mathbf{v}_2 | = | \mathbf{v}_1 | | \mathbf{v}_2 | \, \mathrm{Sin}[\alpha]$
  - In coordinates:

$$\mathbf{v}_1 \times \mathbf{v}_2 = \{ \mathbf{v}_{1_y} \mathbf{v}_{2_z} - \mathbf{v}_{1_z} \mathbf{v}_{2_y}, \mathbf{v}_{1_z} \mathbf{v}_{2_x} - \mathbf{v}_{1_x} \mathbf{v}_{2_z}, \mathbf{v}_{1_x} \mathbf{v}_{2_y} - \mathbf{v}_{1_y} \mathbf{v}_{2_x} \}$$

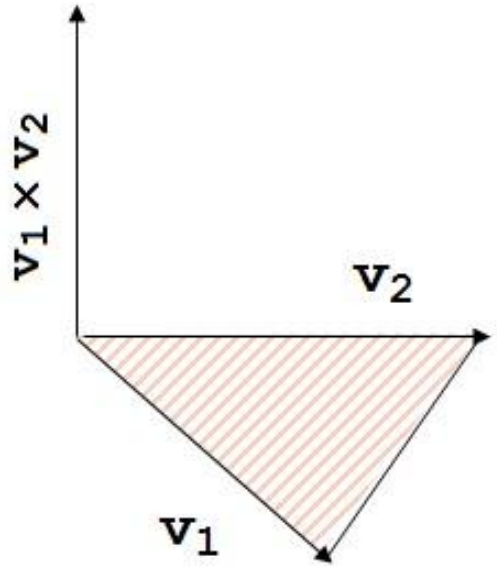


More Vector Operations

- Uses of cross products
  - Getting the normal vector of the plane
    - E.g., the normal of a triangle formed by $\mathbf{v}_1\mathbf{v}_2$
  - Computing area of the triangle formed by $\mathbf{v}_1\mathbf{v}_2$

$$\text{Area} = \frac{| \mathbf{v}_1 \times \mathbf{v}_2 |}{2}$$

    - Testing if vectors are parallel: $| \mathbf{v}_1 \times \mathbf{v}_2 | = 0$

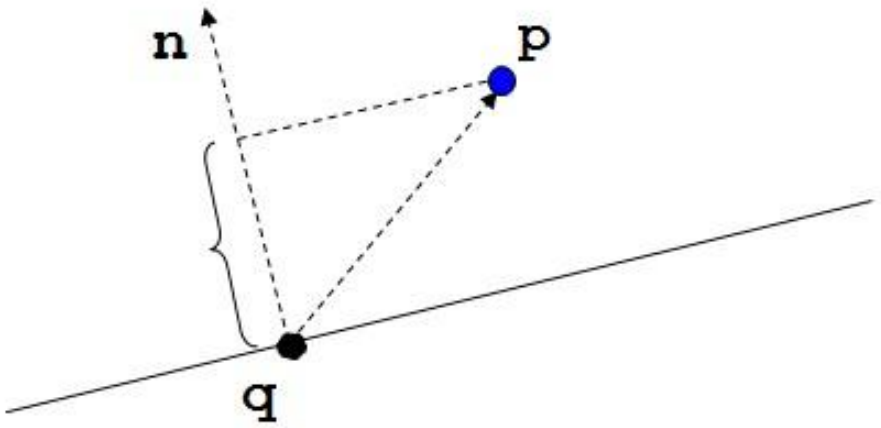


Properties

	Dot Product	Cross Product
Distributive?	$\mathbf{v} \cdot (\mathbf{v}_1 + \mathbf{v}_2) = \mathbf{v} \cdot \mathbf{v}_1 + \mathbf{v} \cdot \mathbf{v}_2$	$\mathbf{v} \times (\mathbf{v}_1 + \mathbf{v}_2) = \mathbf{v} \times \mathbf{v}_1 + \mathbf{v} \times \mathbf{v}_2$
Commutative?	$\mathbf{v}_1 \cdot \mathbf{v}_2 = \mathbf{v}_2 \cdot \mathbf{v}_1$	$\mathbf{v}_1 \times \mathbf{v}_2 = -\mathbf{v}_2 \times \mathbf{v}_1$ (Sign change!)
Associative?	~~$\mathbf{v}_1 \cdot (\mathbf{v}_2 \cdot \mathbf{v}_3)$~~	$\mathbf{v}_1 \times (\mathbf{v}_2 \times \mathbf{v}_3) \neq (\mathbf{v}_1 \times \mathbf{v}_2) \times \mathbf{v}_3$

Simplification (2D)

- Distance to a line
  - Line represented as a point **q** on the line, and a perpendicular unit vector (the *normal*) **n**
    - To get **n**: take a vector {x,y} along the line, **n** is {-y,x} followed by normalization
  - Distance from any point **p** to the line: $(\mathbf{p} - \mathbf{q}) \cdot \mathbf{n}$
    - Projection of vector (**p**-**q**) onto **n**
  - This distance has a sign
    - "Above" or "under" of the line
    - We will use the distance squared



Simplification (2D)

- Closed point to multiple lines
  - Sum of squared distances from **p** to all lines (Quadratic Error Metric, QEM)
    - Input lines: $\{q_1, n_1\}, \ldots, \{q_m, n_m\}$

$$QEM[p] = \sum_{i=1}^{m} ((p - q_i) \cdot n_i)^2$$

  - We want to find the **p** with the minimum QEM
    - Since QEM is a convex quadratic function of **p**, the minimizing **p** is where the derivative of QEM is zero, which is a linear equation

$$\frac{\partial QEM[p]}{\partial p} = 0$$

Simplification (2D)

- Minimizing QEM
  - Writing QEM in matrix form

$$QEM[p] = \sum_{i=1}^{m} ((p - q_i) \cdot n_i)^2$$

$$p = (p_x \;\; p_y) \leftarrow \text{Row vector}$$

Matrix transpose

$$QEM[p] = p \cdot a \cdot p^T - 2\, p \cdot b + c \qquad \text{[Eq. 1]}$$

Matrix (dot) product

$$a = \begin{pmatrix} \sum_{i=1}^{m} n_{i_x} n_{i_x} & \sum_{i=1}^{m} n_{i_x} n_{i_y} \\ \sum_{i=1}^{m} n_{i_x} n_{i_y} & \sum_{i=1}^{m} n_{i_y} n_{i_y} \end{pmatrix} \qquad b = \begin{pmatrix} \sum_{i=1}^{m} n_{i_x} (n_i \cdot q_i) \\ \sum_{i=1}^{m} n_{i_y} (n_i \cdot q_i) \end{pmatrix} \qquad c = \sum_{i=1}^{m} (n_i \cdot q_i)^2$$

2x2 matrix — 1x2 column vector — Scalar

Simplification (2D)

- Minimizing QEM

$$QEM[p] = p \cdot a \cdot p^T - 2\, p \cdot b + c$$

  - Solving the zero-derivative equation:

$$\frac{\partial\, QEM[p]}{\partial\, p} = 2\, a \cdot p^T - 2\, b = 0$$

$$\downarrow$$

$$a \cdot p^T = b \qquad \text{[Eq. 2]}$$

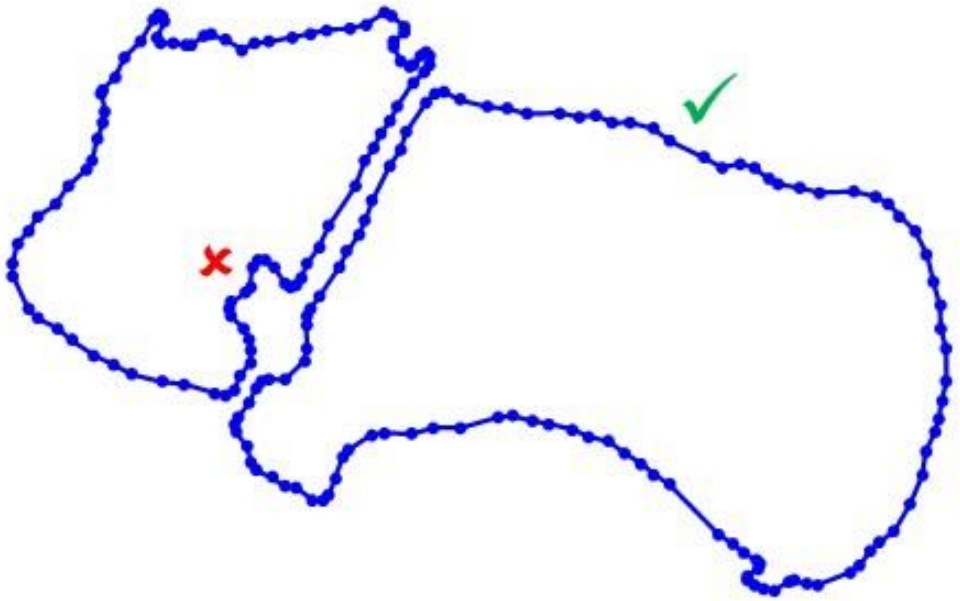
$$\downarrow$$

$$\begin{pmatrix} \sum_{i=1}^{m} n_{i_x} n_{i_x} & \sum_{i=1}^{m} n_{i_x} n_{i_y} \\ \sum_{i=1}^{m} n_{i_x} n_{i_y} & \sum_{i=1}^{m} n_{i_y} n_{i_y} \end{pmatrix} \begin{pmatrix} p_x \\ p_y \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{m} n_{i_x} (n_i \cdot q_i) \\ \sum_{i=1}^{m} n_{i_y} (n_i \cdot q_i) \end{pmatrix}$$

  - A linear system with 2 equations and 2 unknowns (p_x,p_y)
    - Using Gaussian elimination, or matrix inversion: $p^T = a^{-1} b$

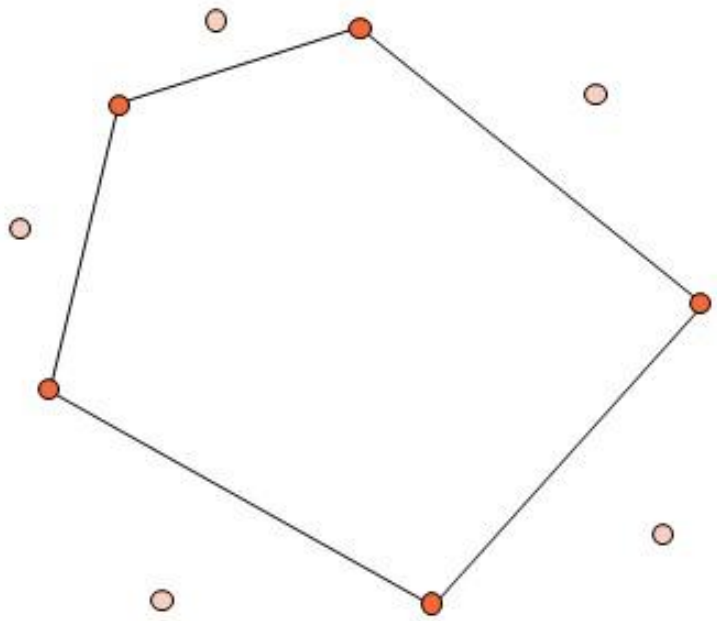
Simplification (2D)

- What vertices to merge first?
  - Pick the ones whose replacement vertex introduces least QEM error (usually lies in flat areas)



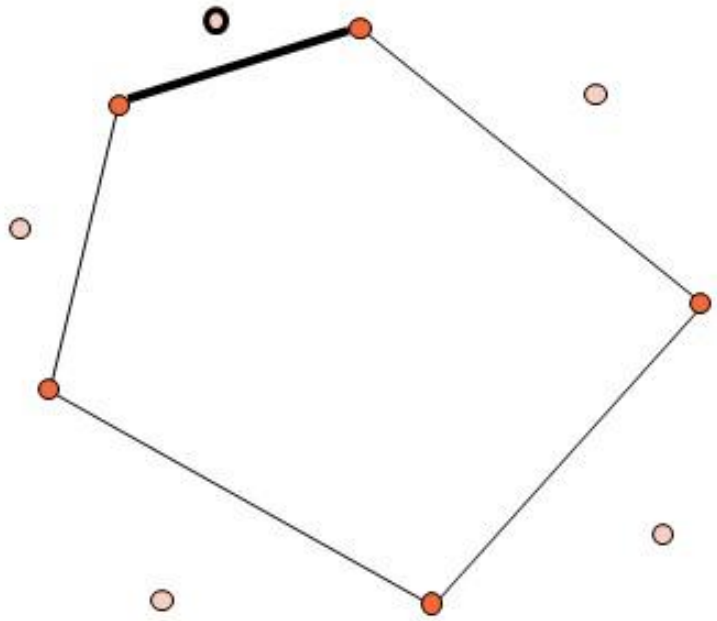
Simplification (2D)

- The algorithm
  - Step 1: For each edge, compute the best vertex location to replace that edge, and the QEM at that location.
    - Store that location (called minimizer) and its QEM with the edge.



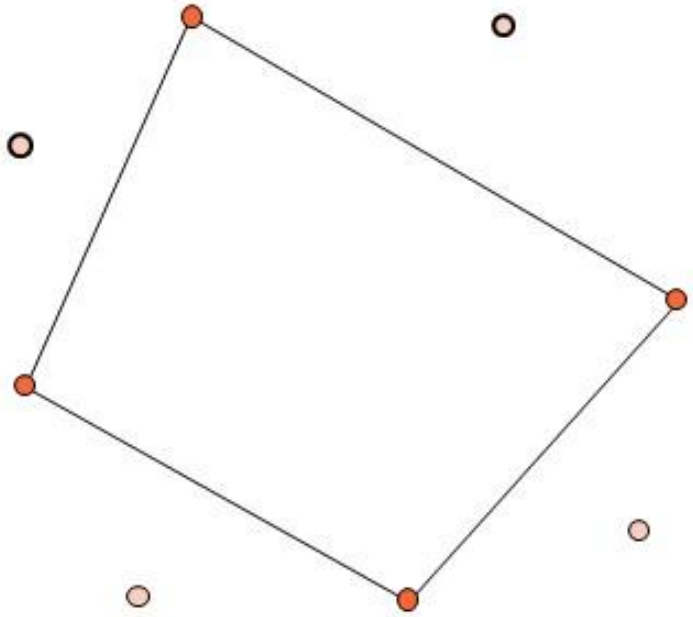
Simplification (2D)

- The algorithm
  - Step 1: For each edge, compute the best vertex location to replace that edge, and the QEM at that location.
    - Store that location (called minimizer) and its QEM with the edge.
  - Step 2: Pick the edge with the lowest QEM and collapse it to its minimizer.
    - Update the minimizers and QEMs of the re-connected edges.



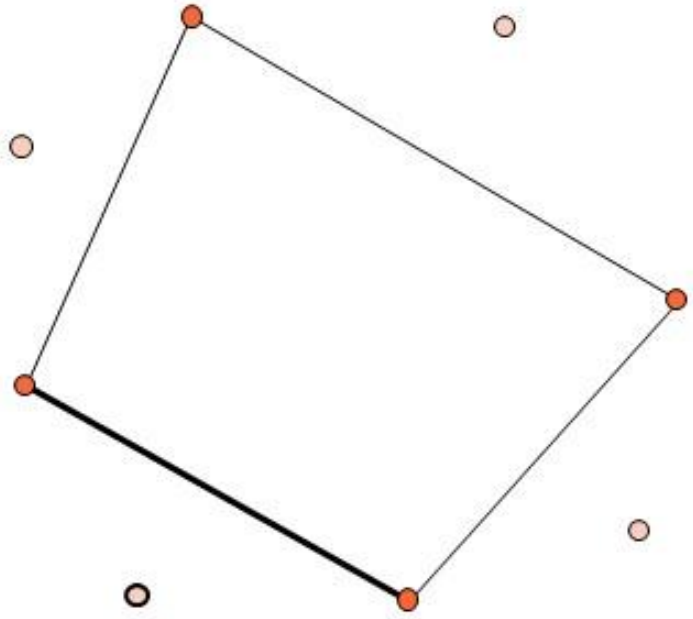
Simplification (2D)

- The algorithm
  - Step 1: For each edge, compute the best vertex location to replace that edge, and the QEM at that location.
    - Store that location (called minimizer) and its QEM with the edge.
  - Step 2: Pick the edge with the lowest QEM and collapse it to its minimizer.
    - Update the minimizers and QEMs of the re-connected edges.



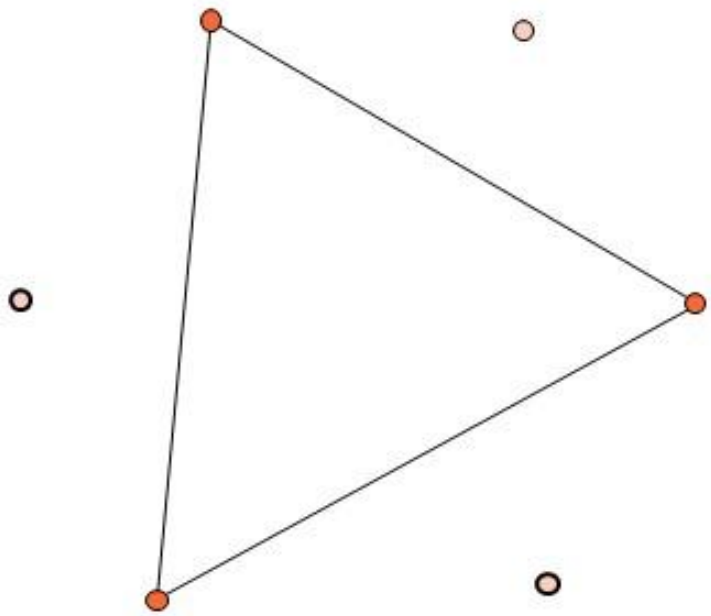
Simplification (2D)

- The algorithm
  - Step 1: For each edge, compute the best vertex location to replace that edge, and the QEM at that location.
    - Store that location (called minimizer) and its QEM with the edge.
  - Step 2: Pick the edge with the lowest QEM and collapse it to its minimizer.
    - Update the minimizers and QEMs of the re-connected edges.
  - Step 3: Repeat step 2, until a desired number of vertices is left.



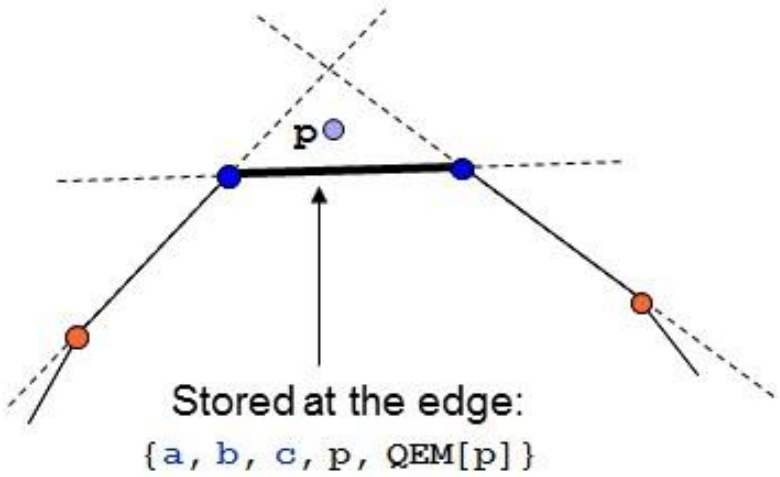
Simplification (2D)

- The algorithm
  - Step 1: For each edge, compute the best vertex location to replace that edge, and the QEM at that location.
    - Store that location (called minimizer) and its QEM with the edge.
  - Step 2: Pick the edge with the lowest QEM and collapse it to its minimizer.
    - Update the minimizers and QEMs of the re-connected edges.
  - Step 3: Repeat step 2, until a desired number of vertices is left.



Simplification (2D)

- Step 1: Computing minimizer and QEM on an edge
  - Consider supporting lines of this edge and adjacent edges
  - Compute and store at the edge:
    - The minimizing location **p** (Eq. 2)
    - QEM (substitute **p** into Eq. 1)
      - Used for edge selection in Step 2
    - QEM coefficients (**a**, **b**, **c**)
      - Used for fast update in Step 2

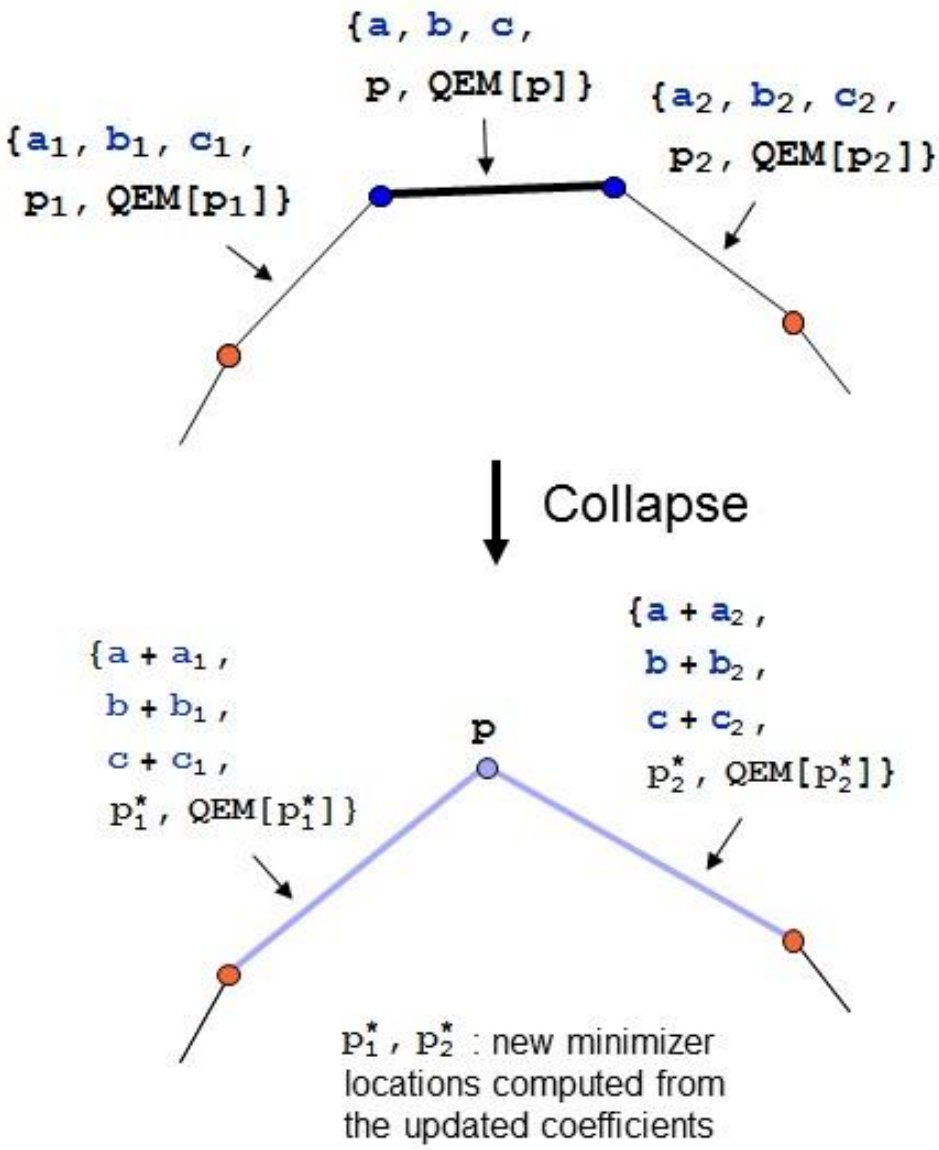


$$QEM[p] = p \cdot a \cdot p^T - 2\, p \cdot b + c \quad \text{[Eq. 1]}$$

Simplification (2D)

Washington University in St. Louis
School of Engineering & Applied Science

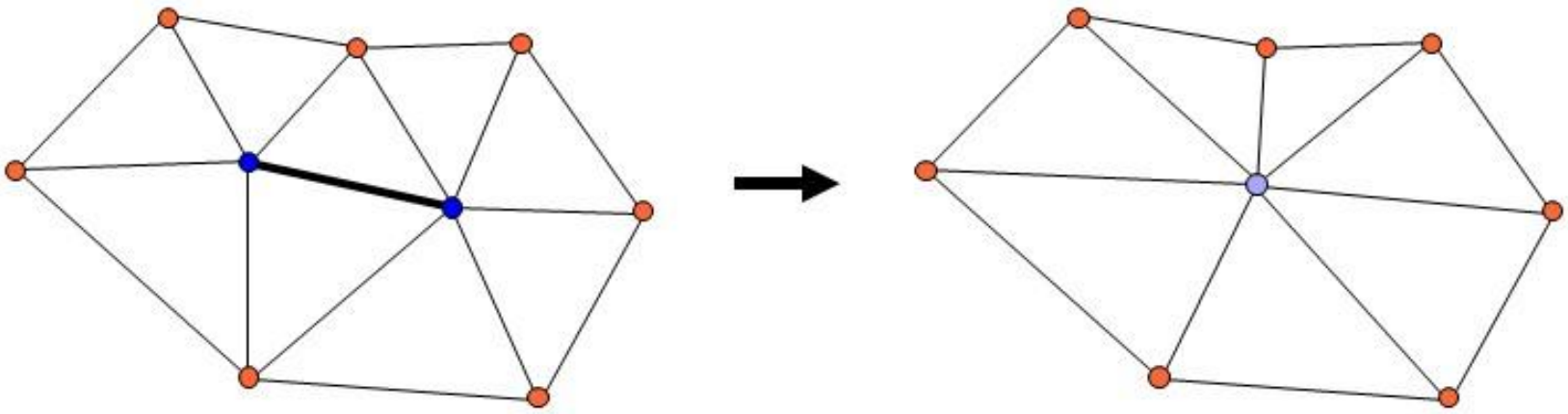
- Step 2: Collapses and updates
  - Remove the edge and its vertices
  - Re-connect two neighbor edges to the minimizer of the removed edge
  - For each re-connected edge:
    - Increment its coefficients by that of the removed edge
      - The coefficients are additive!
    - Re-compute its minimizer and QEM



Simplification (3D)

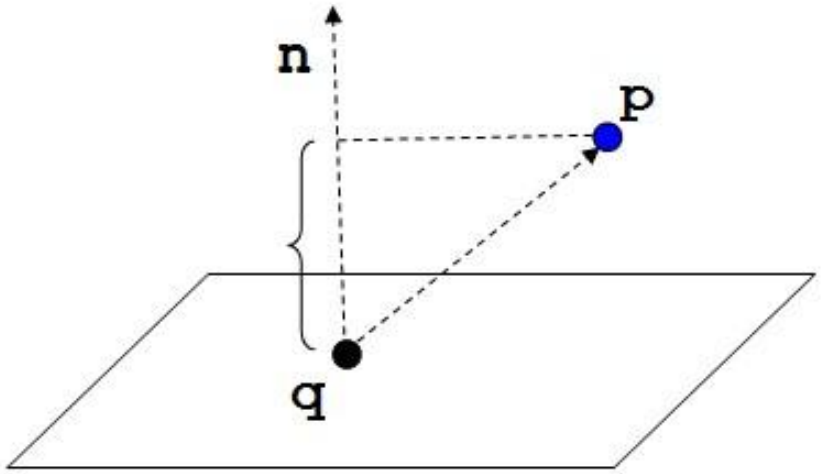
Washington University in St. Louis
School of Engineering & Applied Science

- The algorithm is similar to 2D
  - Replace two edge-adjacent vertices by one vertex
    - Placing new vertices closest to supporting planes of adjacent triangles
  - Prioritize collapses based on QEM



Simplification (3D)

- Distance to a plane (similar to the line case)
  - Plane represented as a point **q** on the plane, and a unit normal vector **n**
    - For a triangle: **n** is the cross-product of two edge vectors
  - Distance from any point **p** to the plane: $(\mathbf{p} - \mathbf{q}) \cdot \mathbf{n}$
    - Projection of vector (**p**-**q**) onto **n**
  - This distance has a sign
    - "above" or "below" the plane
    - We use its square



Simplification (3D)

- Closest point to multiple planes
  - Input planes: $\{q_1, n_1\}, \ldots, \{q_m, n_m\}$
  - QEM (same as in 2D)

$$QEM[p] = \sum_{i=1}^{m} ((p - q_i) \cdot n_i)^2$$

    - In matrix form: $p = (p_x \; p_y \; p_z)$

$$QEM[p] = p \cdot a \cdot p^T - 2\, p \cdot b + c$$

$$a = \begin{pmatrix} \sum_{i=1}^{m} n_{i_x} n_{i_x} & \sum_{i=1}^{m} n_{i_x} n_{i_y} & \sum_{i=1}^{m} n_{i_x} n_{i_z} \\ \sum_{i=1}^{m} n_{i_y} n_{i_x} & \sum_{i=1}^{m} n_{i_y} n_{i_y} & \sum_{i=1}^{m} n_{i_y} n_{i_z} \\ \sum_{i=1}^{m} n_{i_z} n_{i_x} & \sum_{i=1}^{m} n_{i_z} n_{i_y} & \sum_{i=1}^{m} n_{i_z} n_{i_z} \end{pmatrix}$$ 3x3 matrix

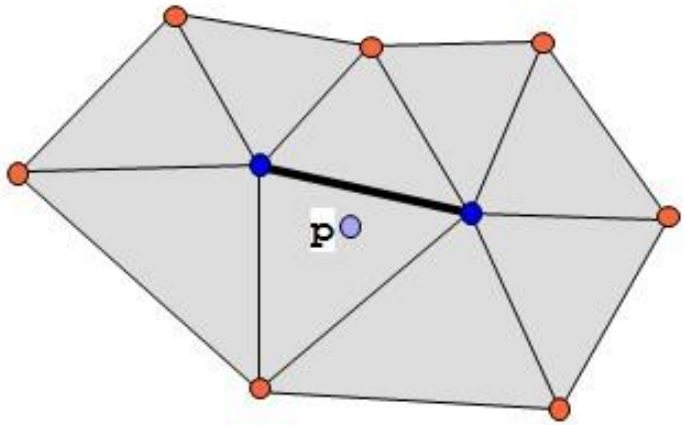
$$b = \begin{pmatrix} \sum_{i=1}^{m} n_{i_x} (n_i \cdot q_i) \\ \sum_{i=1}^{m} n_{i_y} (n_i \cdot q_i) \\ \sum_{i=1}^{m} n_{i_z} (n_i \cdot q_i) \end{pmatrix}$$ 1x3 column vector

$$c = \sum_{i=1}^{m} (n_i \cdot q_i)^2$$ Scalar

  - Find p that minimizes QEM: $\boxed{a \cdot p^T = b}$
    - A linear system with 3 equations and 3 unknowns (p_x, p_y, p_z)

Simplification (3D)

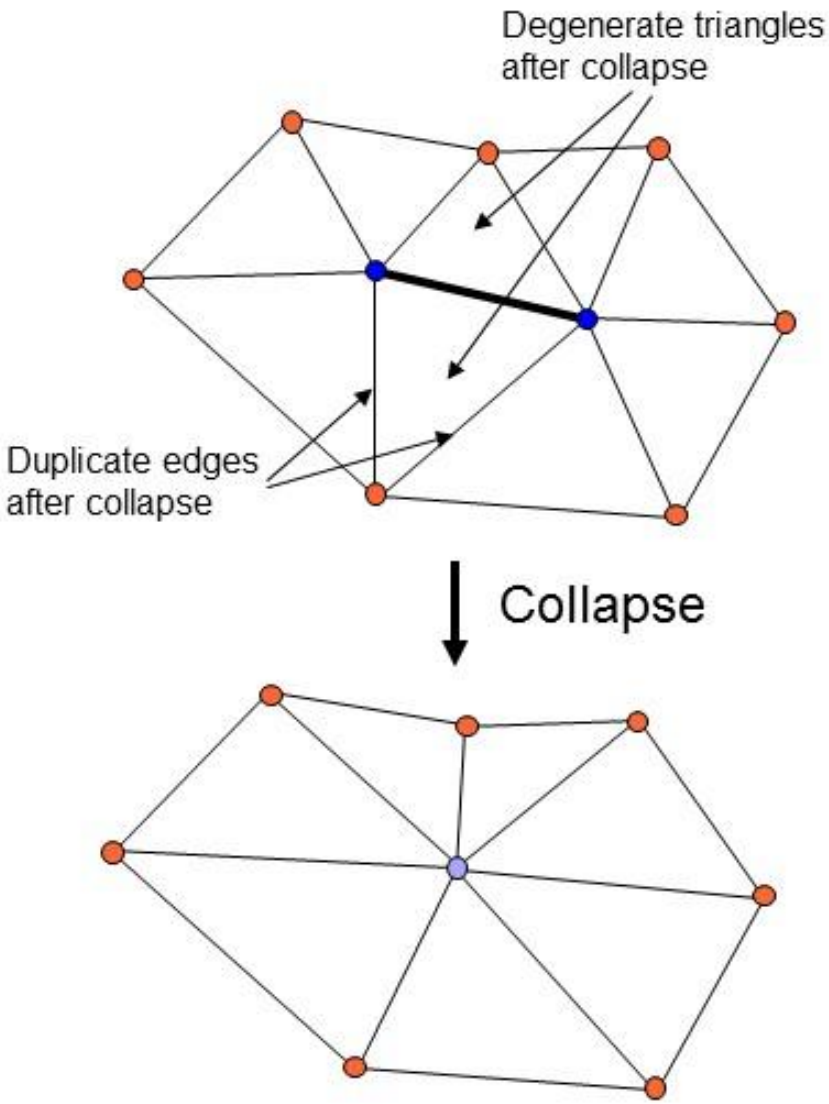
- Step 1: Computing minimizer and QEM on an edge
  - Consider supporting planes of all triangles adjacent to the edge
  - Compute and store at the edge:
    - The minimizing location **p**
    - **QEM[p]**
    - QEM coefficients (**a**, **b**, **c**)



The supporting planes for all shaded triangles should be considered when computing the minimizer of the middle edge.

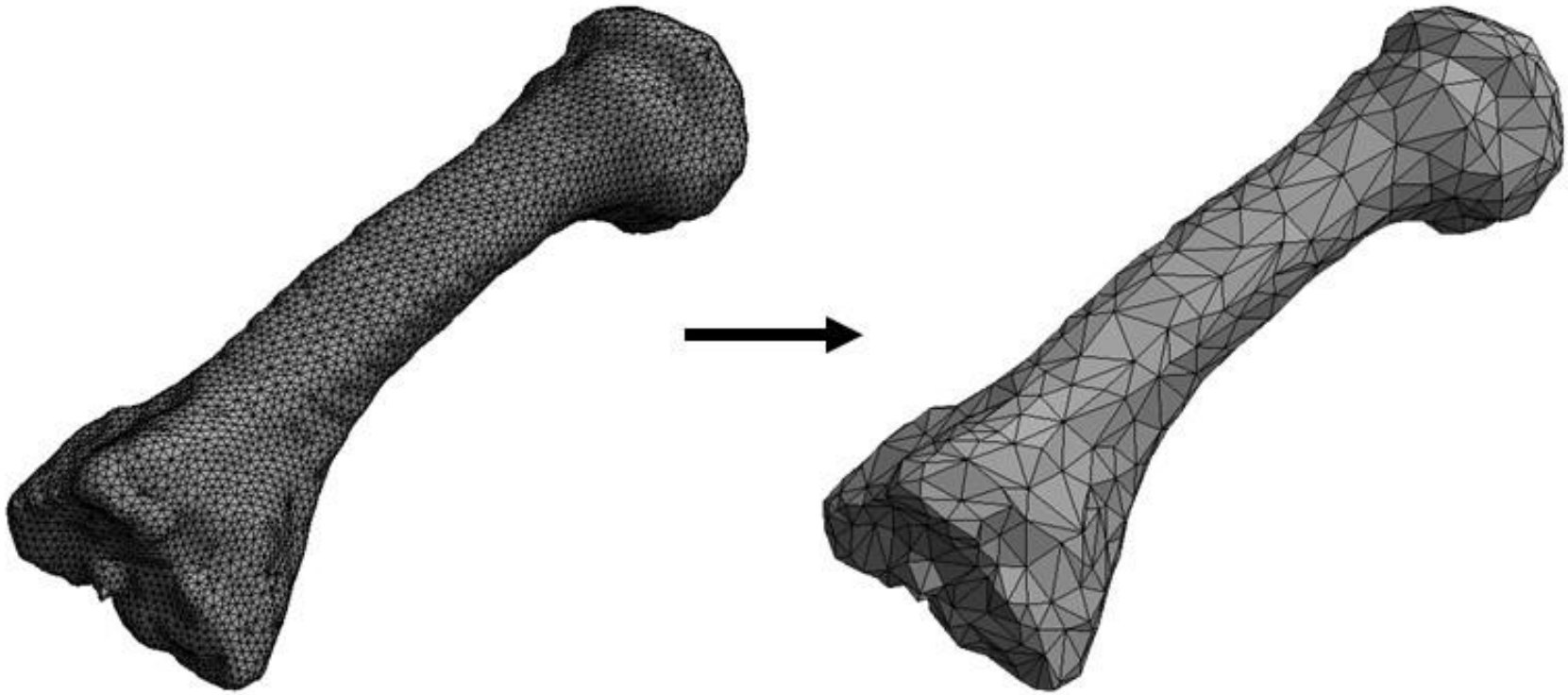
Simplification (3D)

- Step 2: Collapsing an edge
  - Remove the edge with least QEM
  - Re-connect neighbor triangles and edges to the minimizer of the removed edge
    - Remove "degenerate" triangles
    - Remove "duplicate" edges
  - For each re-connected edge:
    - Increment its coefficients by that of the removed edge
    - Re-compute its minimizer and QEM



Simplification (3D)

- Example:



5600 vertices

500 vertices

Further Readings

- Fairing:
  - "*A signal processing approach to fair surface design*", by G. Taubin (1995)
    - No-shrinking centroid-averaging
    - Google citations > 1000
- Simplification:
  - "*Surface simplification using quadric error metrics*", by M. Garland and P. Heckbert (1997)
    - Edge-collapse simplification
    - Google citations > 2000