Surface Simplification Using Quadric Error Metrics (QEM)

Michael Garland
Paul Heckbert
SIGGRAPH 97

Demo

Objectives

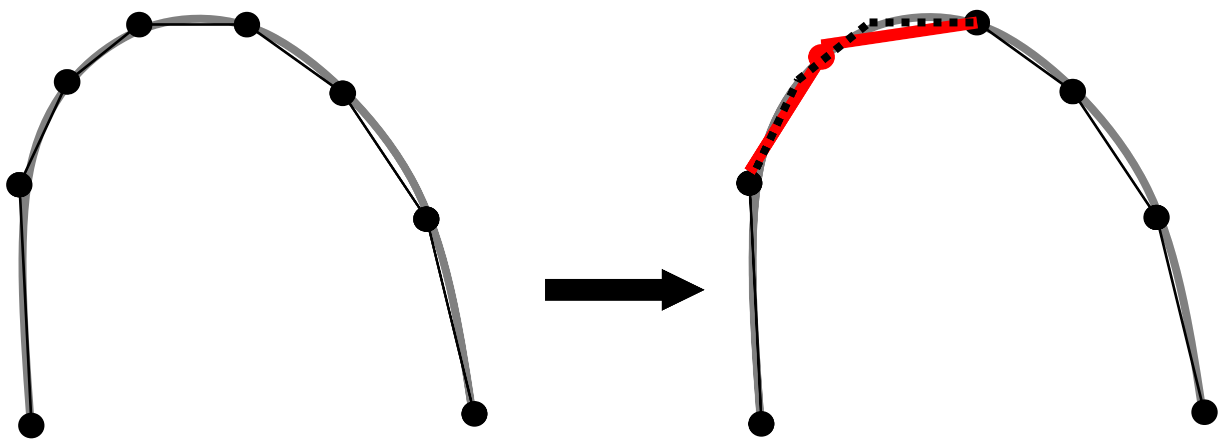
- Fast and good quality
  - Efficiency and quality trade-off

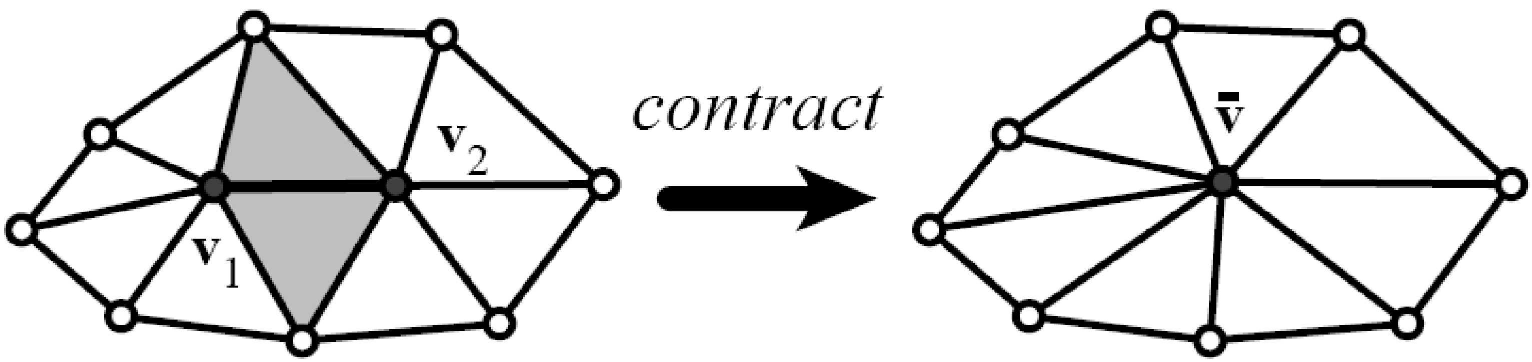
- Convenient characterization of error/shape
  - Compact and efficient to compute

Things to consider

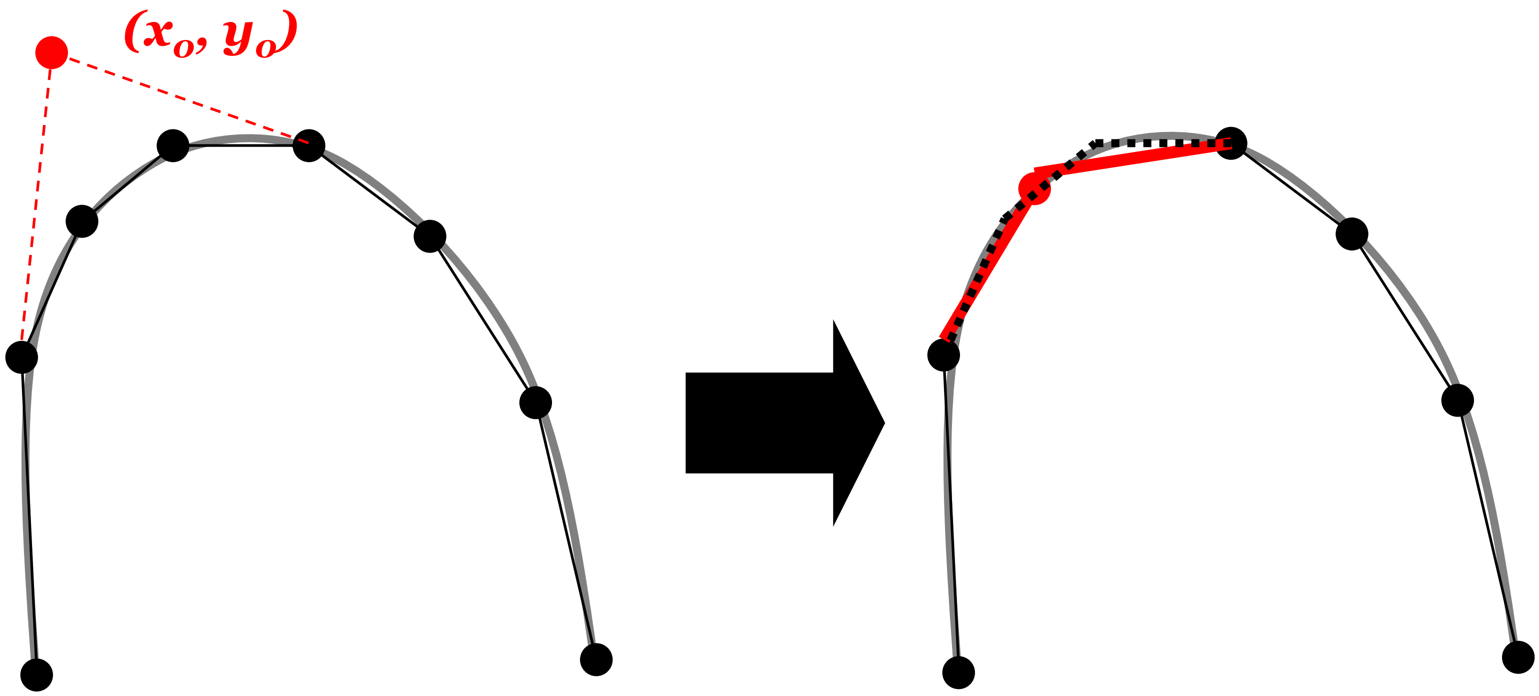
1. **Defining the basic operation**
2. **Defining the error metrics**
3. Primitives removal strategy
4. Remaining updates

Basic operation

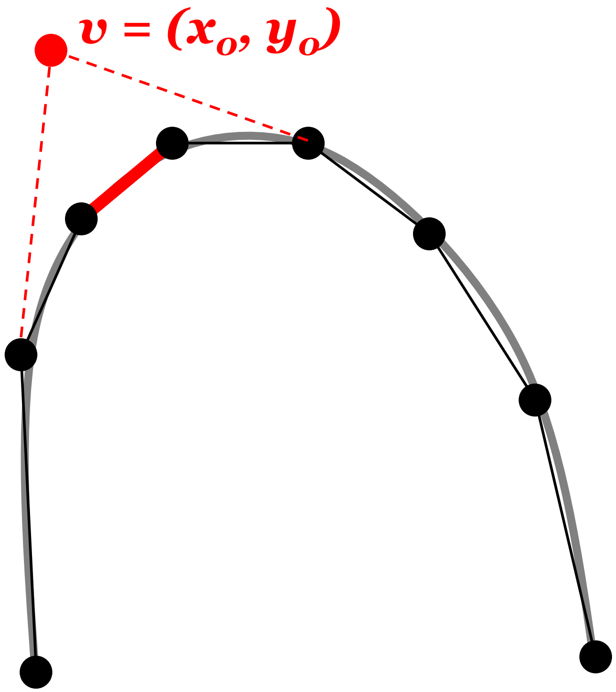




The error metrics – 2D



The error metrics – 2D



Matrix form

$$[a_1 \;\; b_1 \;\; c_1] * \begin{bmatrix} x_o \\ y_o \\ 1 \end{bmatrix}$$

$$L_1 = \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} \quad v = \begin{bmatrix} xo \\ y_o \\ 1 \end{bmatrix}$$

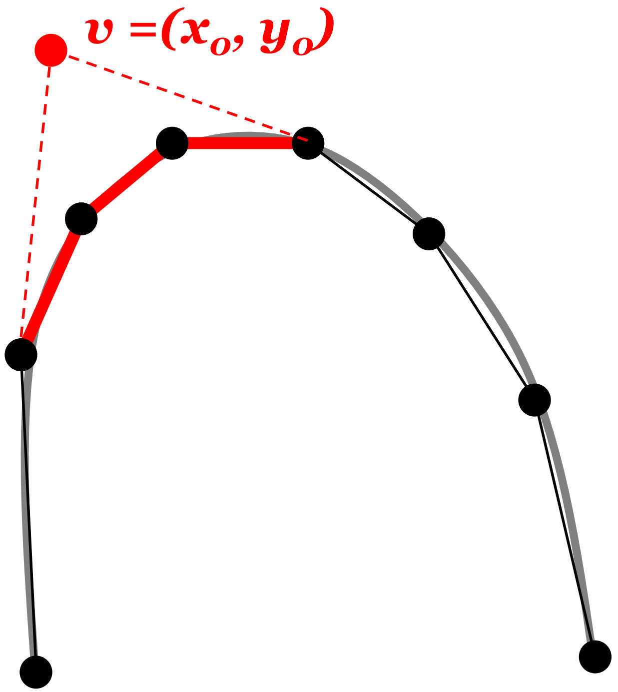
$$d_1^2 = [x_o \;\; y_o \;\; 1] * \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} * [a_1 \;\; b_1 \;\; c_1] * \begin{bmatrix} x_o \\ y_o \\ 1 \end{bmatrix}$$

$$d_1^2 = v^T * (L_1 * L_1^T) * v = v^T * (Q_1) * v$$

Line segment L_1: $a_1x + b_1y + c_1 = 0$

(x_o, y_o) to L_1 is $d_1 = |a_1 * x_o + b_1 * y_o + c_1|$

The error metrics – 2D



Total squared distance

$$E_v = \Sigma_i\, v^T * (L_i * L_i^T) * v = v^T * (\Sigma_i Q_i) * v$$

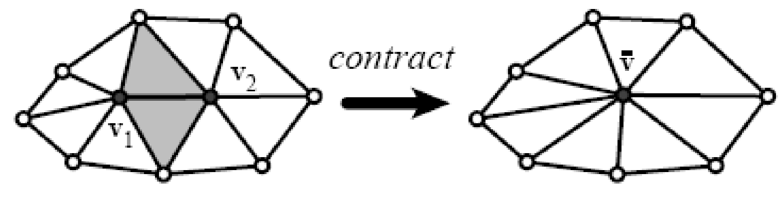
To better fit the curve minimize squared distance

$$dE_v/dv = (\Sigma_i\, Q_i) * v = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

So best v is at $(\Sigma_i\, Q_i)^{-1} * \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

For meshes

- Use plane equation instead of line equation
  - $ax + by + cz + d = 0$
    - $a^2 + b^2 + c^2 = 1$
  - $v = [v_x \;\; v_y \;\; v_z \;\; 1]^T$
  - $p = [a \;\; b \;\; c \;\; d]^T$



$$
\begin{aligned}
\Delta(\mathbf{v}) &= \sum_{\mathbf{p}\in\text{planes}(\mathbf{v})} (\mathbf{v}^{\mathsf{T}}\mathbf{p})(\mathbf{p}^{\mathsf{T}}\mathbf{v}) \\
&= \sum_{\mathbf{p}\in\text{planes}(\mathbf{v})} \mathbf{v}^{\mathsf{T}}(\mathbf{p}\mathbf{p}^{\mathsf{T}})\mathbf{v} \\
&= \mathbf{v}^{\mathsf{T}}\left(\sum_{\mathbf{p}\in\text{planes}(\mathbf{v})} \mathbf{K}_{\mathbf{p}}\right)\mathbf{v}
\end{aligned}
$$

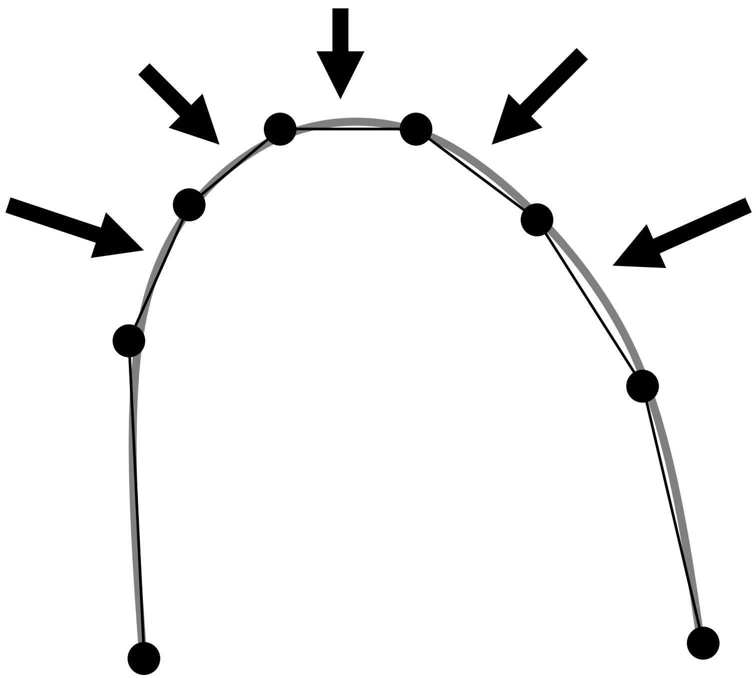
$$
\mathbf{K}_{\mathbf{p}} = \mathbf{p}\mathbf{p}^{\mathsf{T}} = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}
$$

Things to consider

1. Defining the basic operation
2. Defining the error metrics
3. **Primitives removal strategy**
4. **Remaining updates**

Primitives removal strategy

But which one?

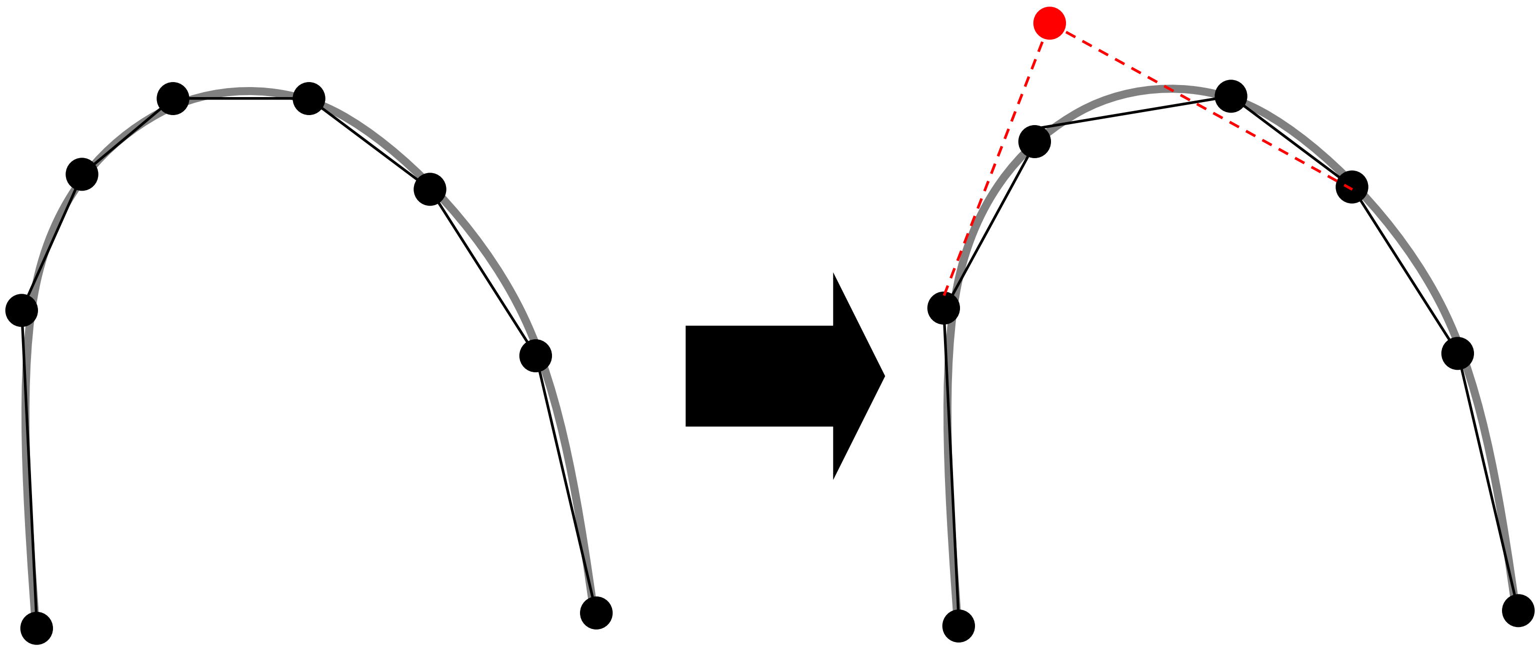


Total squared distance

$$E_v = \Sigma_i\, v^T * (L_i * L_i^T) * v = v^T * (\Sigma_i Q_i) * v$$

Use priority queue!

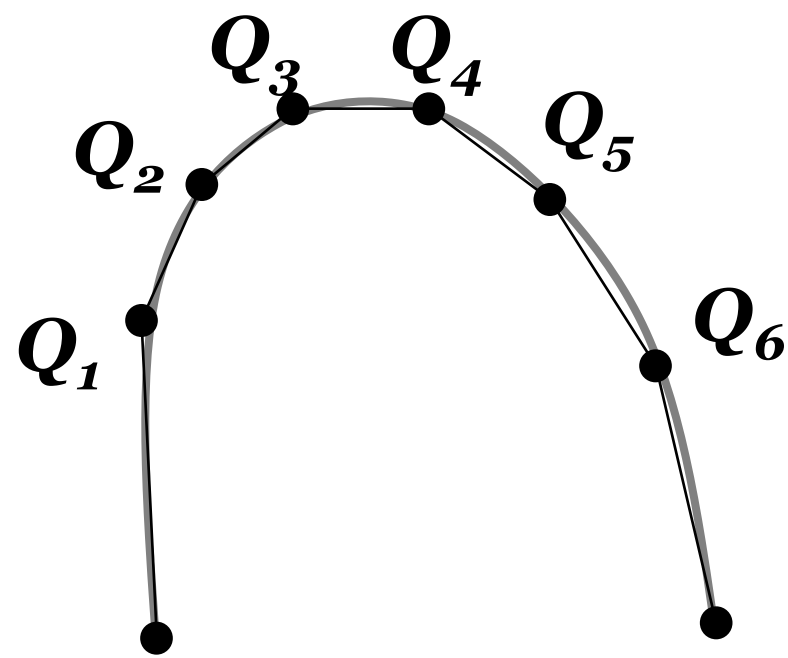
Efficiency consideration



- *Accumulated error w/ respect to ALL underlying segments!*
- *How many do we have in queue?*

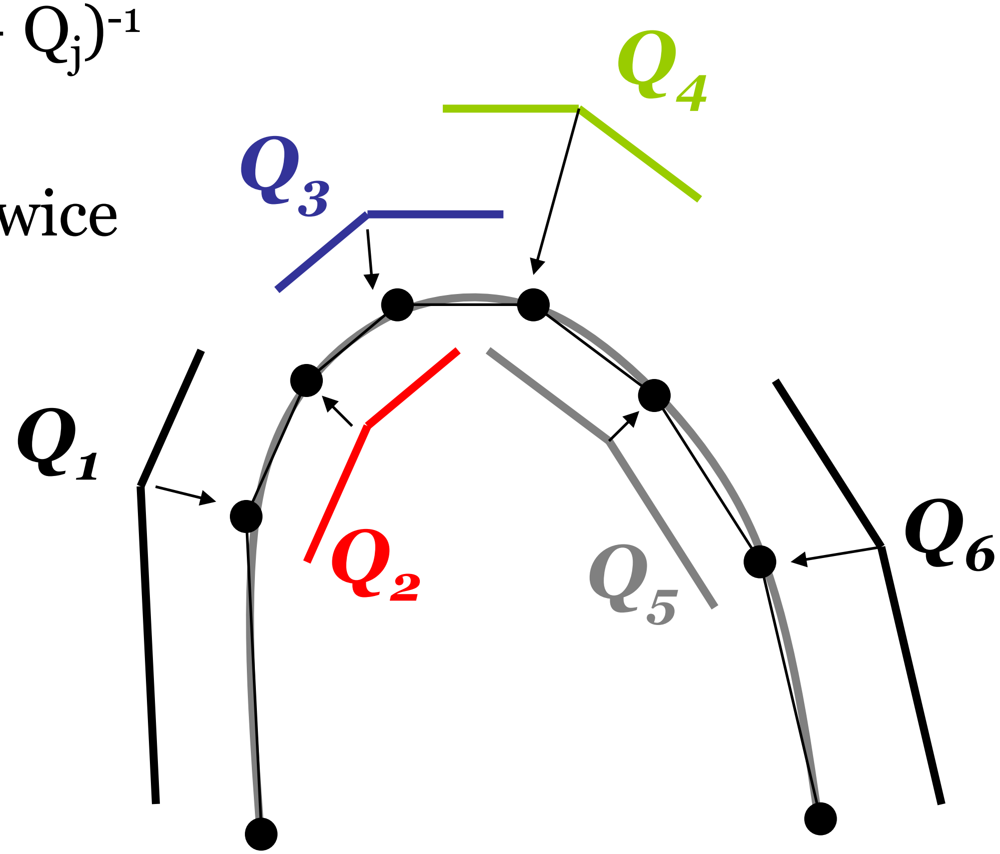
Cheats!

- Keep indices to all underlying segments!
  - Storage and indexing speed.
  - Sometimes you only compare errors
- So vertices remember their own incident segments as Q
  - V_{new} = minimize $(Q_i + Q_j)^{-1}$
  - Just sum two
  - The middle counted twice
  - For meshes, 3 times



Cheats!

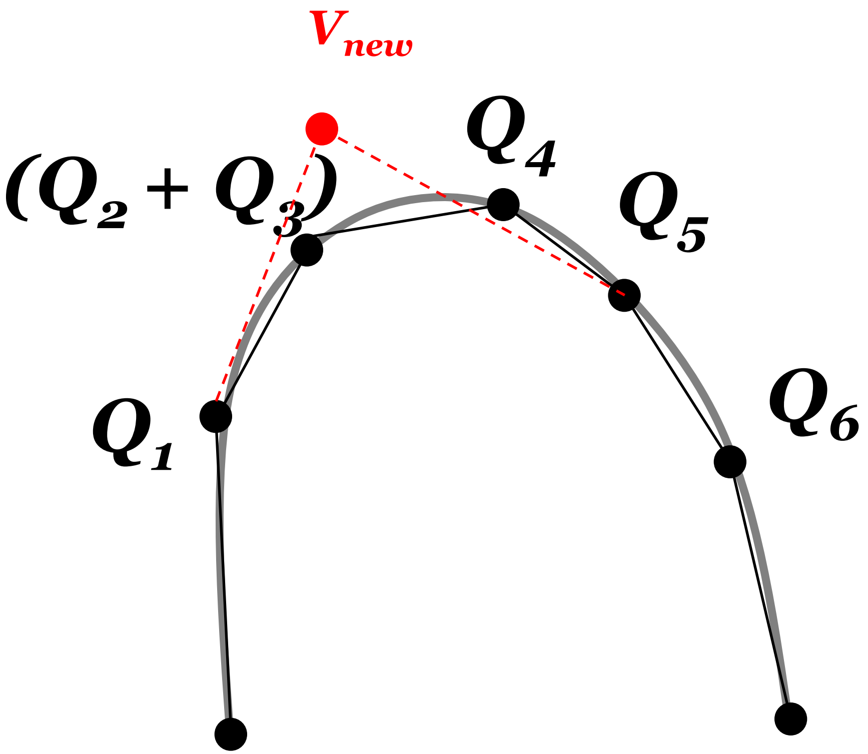
- So vertices remember their own incident segments as Q
  - V_{new} = minimize $(Q_i + Q_j)^{-1}$
  - Just sum two
  - The middle counted twice



Remaining Updates

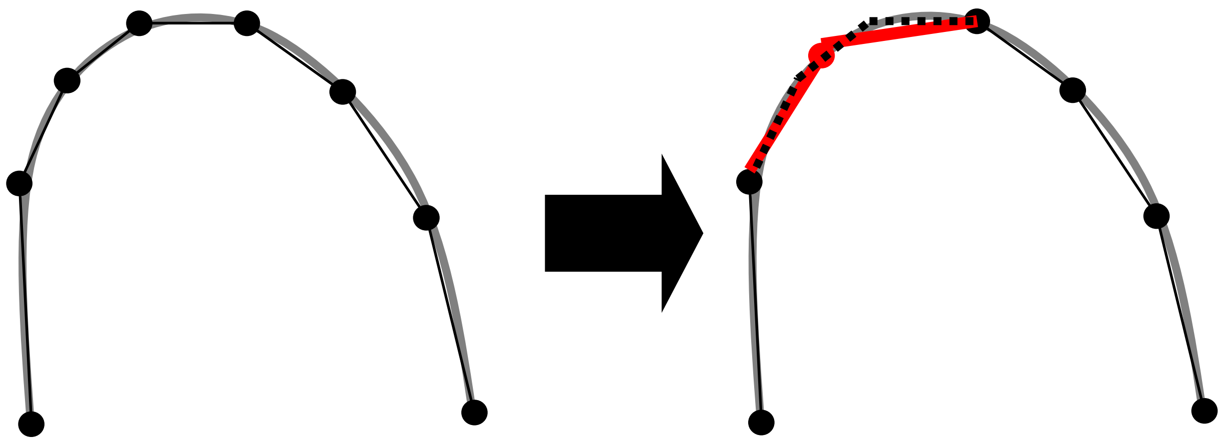
- *Error* (V_{new}) = $V_{new}^T(Q_2 + Q_3 + Q_4)V_{new}$

- *Minimized error to find* V_{new}



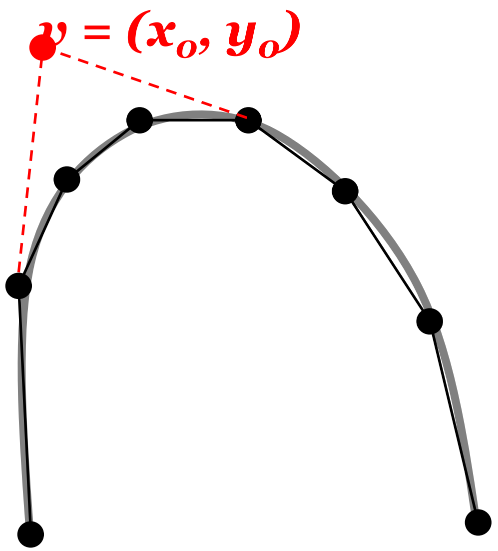
Quick review

1. **Defining the basic operation**
2. Defining the error metrics
3. Primitives removal strategy
4. Remaining updates



Quick review

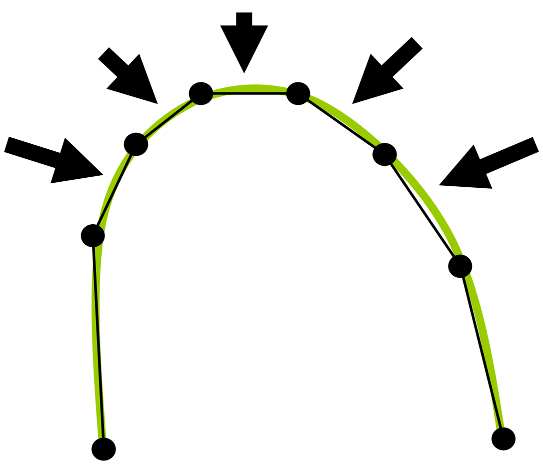
1. Defining the basic operation
2. **Defining the error metrics**
3. Primitives removal strategy
4. Remaining updates

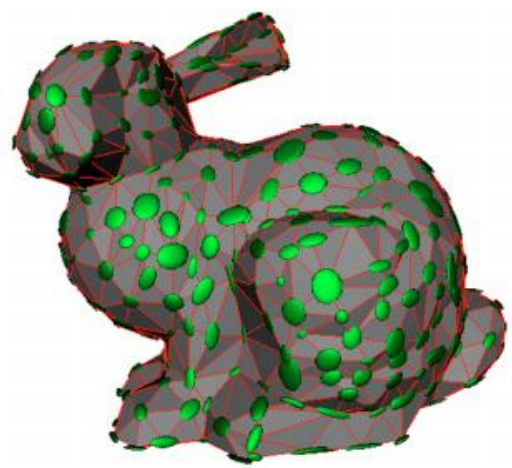


$$
\begin{aligned}
\Delta(\mathbf{v}) &= \sum_{\mathbf{p} \in \text{planes}(\mathbf{v})} (\mathbf{v}^{\mathsf{T}}\mathbf{p})(\mathbf{p}^{\mathsf{T}}\mathbf{v}) \\
&= \sum_{\mathbf{p} \in \text{planes}(\mathbf{v})} \mathbf{v}^{\mathsf{T}}(\mathbf{p}\mathbf{p}^{\mathsf{T}})\mathbf{v} \\
&= \mathbf{v}^{\mathsf{T}} \left(\sum_{\mathbf{p} \in \text{planes}(\mathbf{v})} \mathbf{K}_{\mathbf{p}} \right) \mathbf{v}
\end{aligned}
$$

Quick review

1. Defining the basic operation
2. Defining the error metrics
3. **Primitives removal strategy**
4. Remaining updates





Quick review

1. Defining the basic operation
2. Defining the error metrics
3. Primitives removal strategy
4. **Remaining updates**

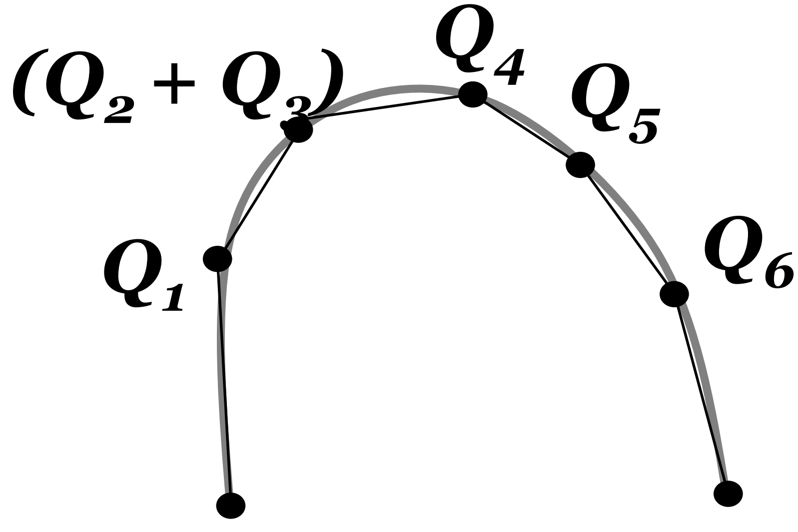


Illustration of operations

1. Initial Q for every vertices
2. Compute v_{new} and err(v_{new}) for every edges
   1. $V_{1\text{-}2}$, $V_{2\text{-}3}$, $V_{3\text{-}4}$, $V_{4\text{-}5}$ …
   2. $(Q_1+Q_2)^{-1}$, $(Q_2+Q_3)^{-1}$, $(Q_3+Q_4)^{-1}$, $(Q_4+Q_5)^{-1}$ …
   3. Put into min heap

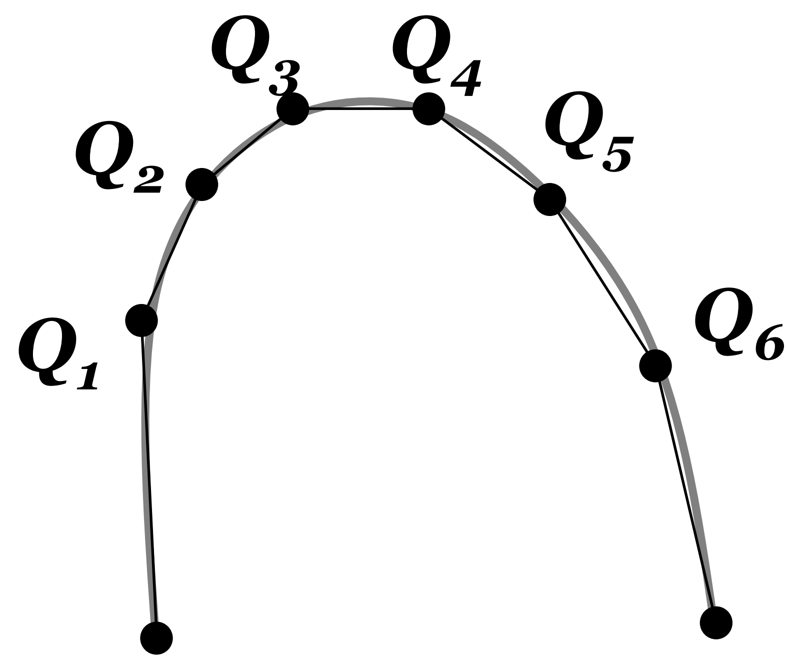
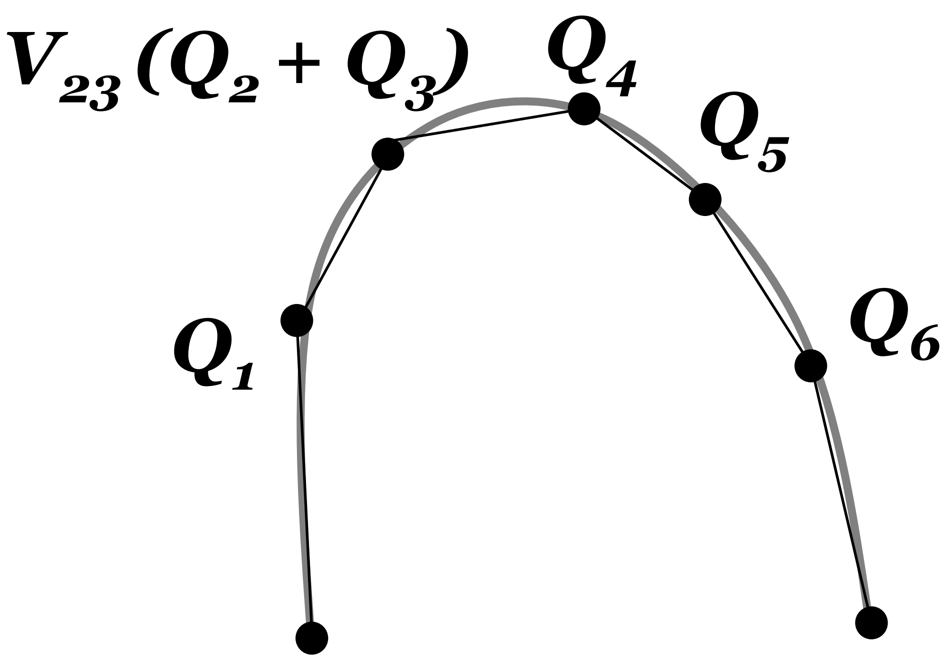


Illustration of operations

3. Select min err(V_{ij}) to contract
4. Update **necessary items** in the min heap
    1. V1-23, V23-4, V3-4, V4-5
    2. Their Q
5. Repeat step-3



Iterations from paper

1. Compute the Q matrices for all the initial vertices

2. Compute the optimal contraction target $\mathbf{v}_{\mathbf{new}}$ for each valid pair $\mathbf{v}_1$ $\mathbf{v}_2$. The error $\mathbf{v}_{\mathbf{new}}{}^{T}(\mathbf{Q}_1+\mathbf{Q}_2)\mathbf{v}_{\mathbf{new}}$ of this target vertex becomes the *cost* of contracting that pair.

3. Place all the pairs in a heap keyed on cost with the minimum cost pair at the top.

4. Iteratively remove the pair $\mathbf{v}_1$ $\mathbf{v}_2$ of least cost from the heap, contract this pair, and update the costs of all valid pairs involving v_{New}

Objectives

- Fast and good quality
  - Efficiency and quality trade-off
  - No indexing
  - Only require necessary updates

- Convenient characterization of error/shape
  - Compact and efficient to compute
  - 10 coefficient, 4x4 matrix inversion/addition

Something not addressed…

- Focused on 2-manifold
  - no joining
- Boundary condition
  - Eaten away
- Triangle inversion
  - Why have the case?