# Evaluation of an Appearance-Preserving Mesh Simplification Scheme

Rasmus Hedin

Department of Electrical Engineering
Linköpings Universitet

2018-06-15

# Outline



LINKÖPINGS UNIVERSITET

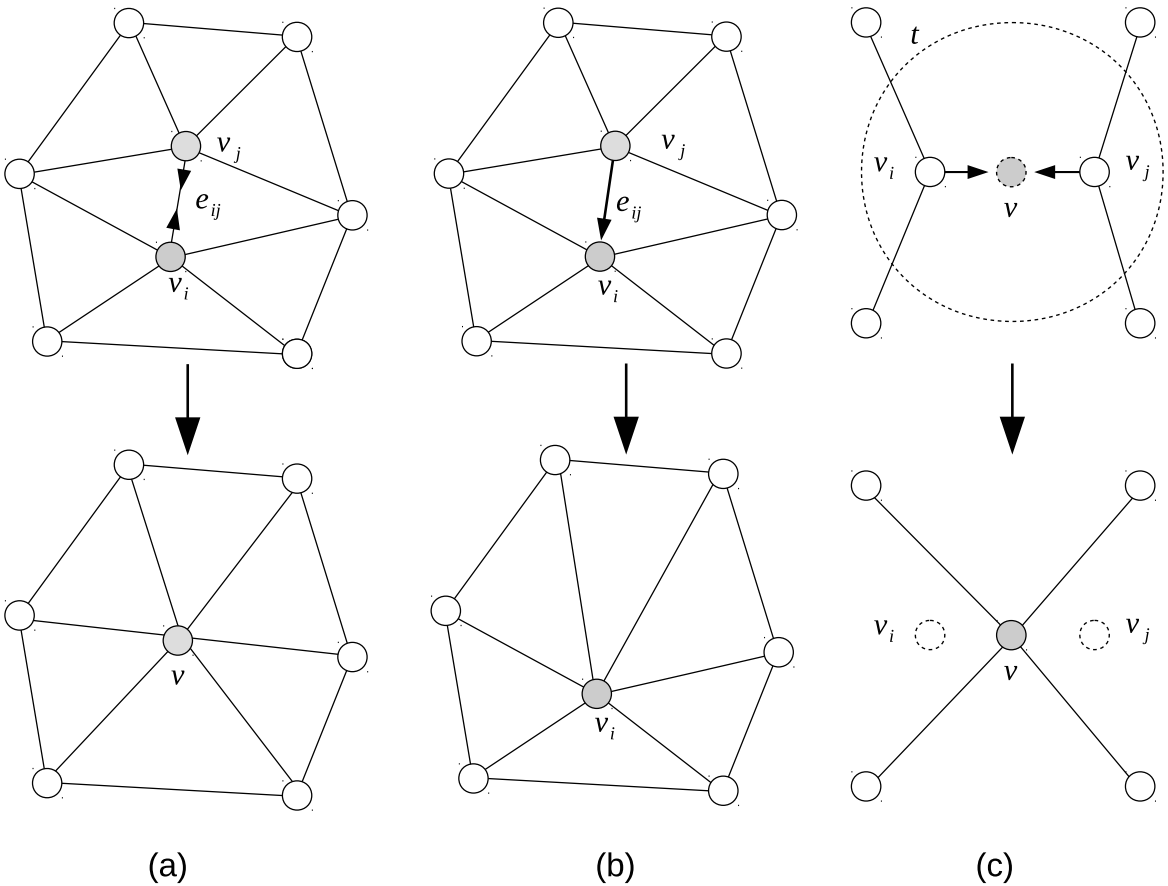
# Outline



LINKÖPINGS UNIVERSITET

# Vertex Merging

- Mesh can be simplified by merging vertices
- (a) edge collapse, (b) vertex removal, and (c) pair contraction



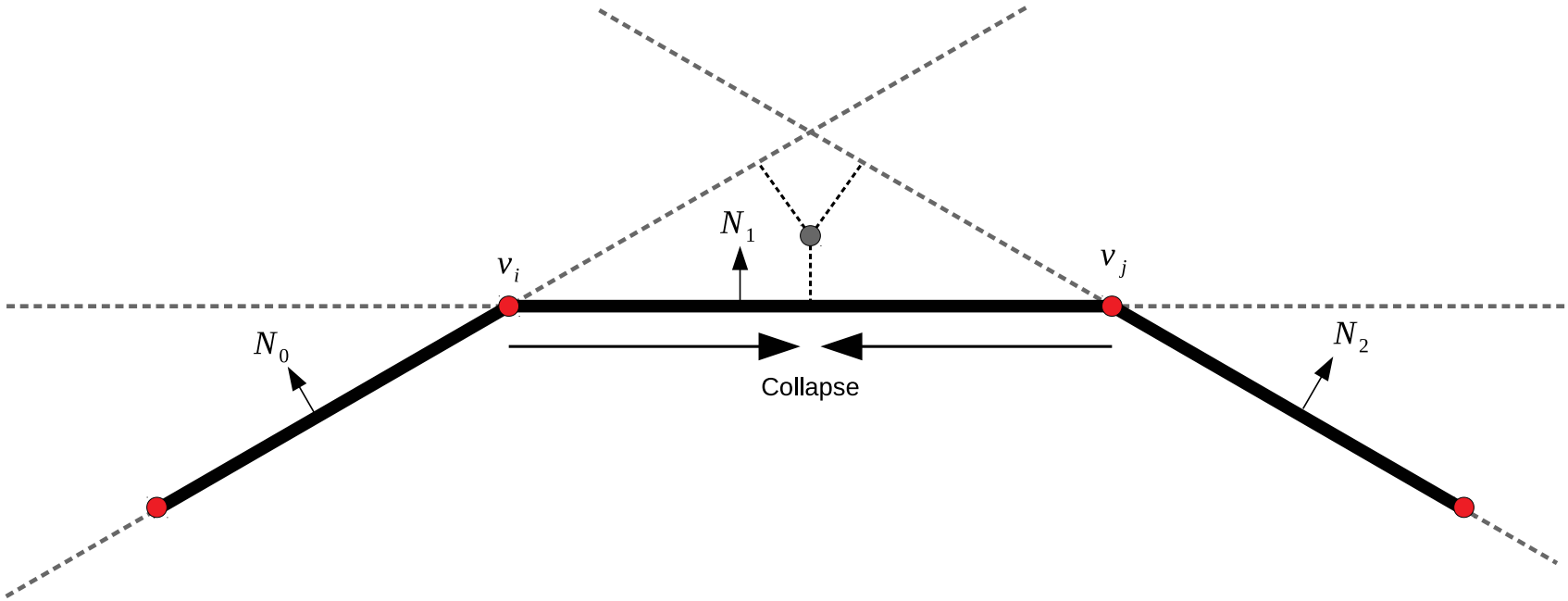
# Mesh Simplification Methods

- Progressive meshes
- Quadric Error Metrcs
- Appearance-preserving Simplification

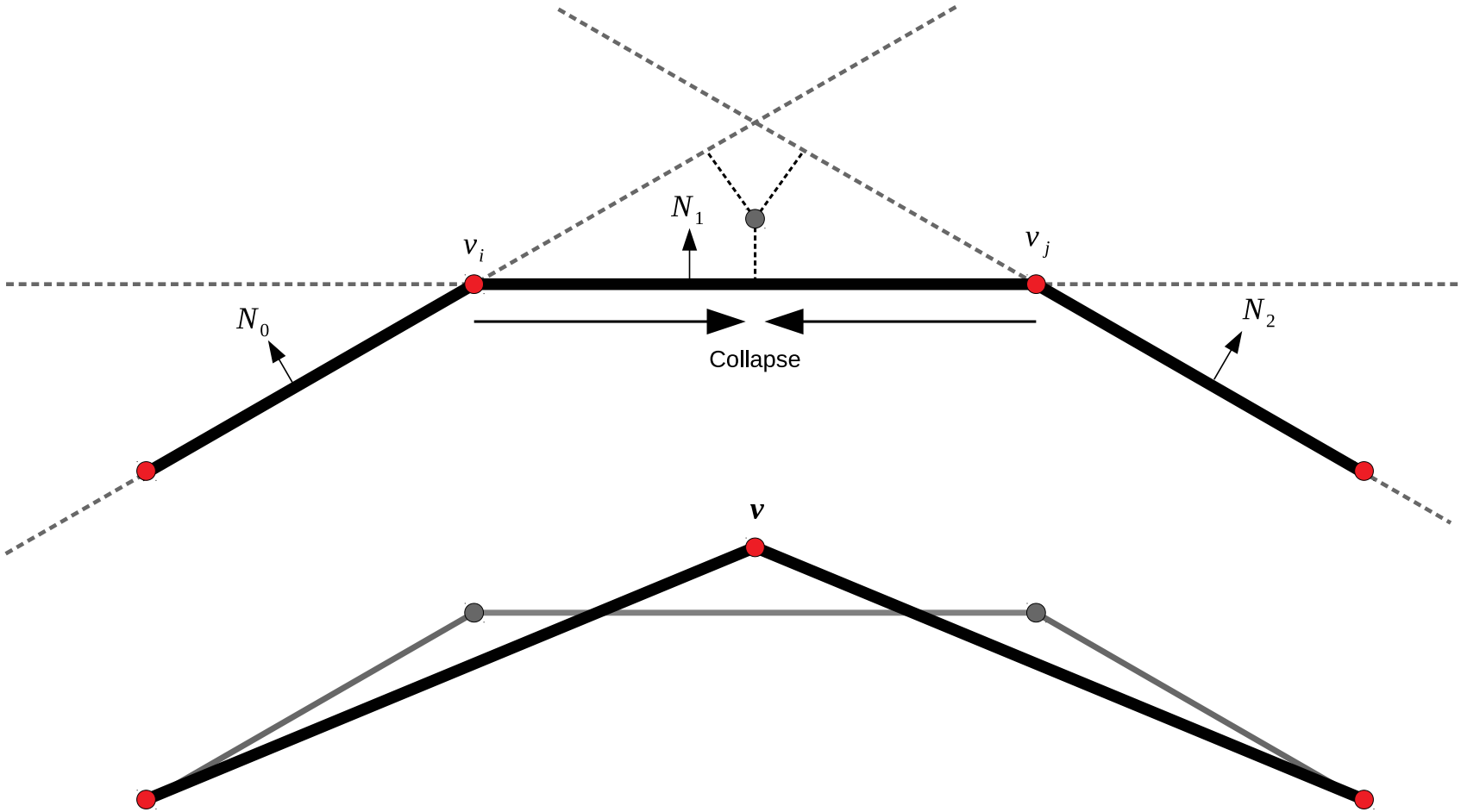
# Outline

# Quadric Error Metric

- Iteratively perform edge collapses $(\mathbf{v}_i, \mathbf{v}_j) \rightarrow \mathbf{v}$
- Cost based on distance to neighboring faces' planes



# Quadric Error Metric



# Calculating cost

Squared distance from point $v$ to plane $f$

$$\mathbf{v} = [x, y, z, 1]^\mathsf{T}, \ \mathbf{f} = [a, b, c, d]^\mathsf{T}$$

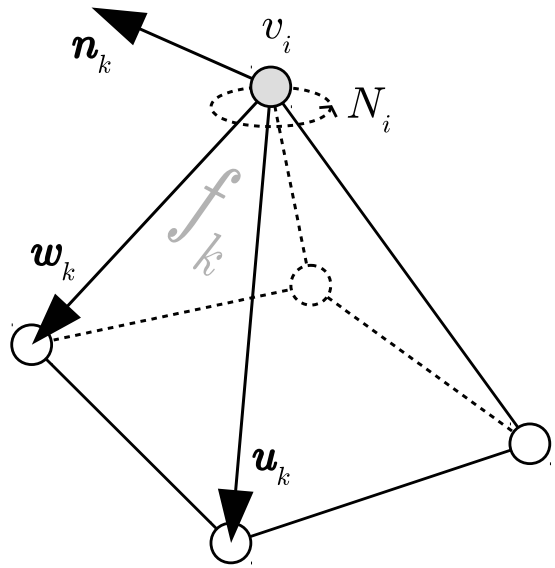
$$\begin{aligned} D^2 &= (\mathbf{f}^\mathsf{T}\mathbf{v})^2 \\ &= \mathbf{v}^\mathsf{T}(\mathbf{f}\mathbf{f}^\mathsf{T})\mathbf{v} \\ &= \mathbf{v}^\mathsf{T}\mathbf{Q}\mathbf{v} \end{aligned}$$

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

# Calculating cost

Sum distances to planes $f_k$ of triangles in neighborhood $N_i$ of $v_i$

$$
\begin{aligned}
D^2 &= \sum_k \mathbf{v}_i^\mathsf{T} \mathbf{Q}_k \mathbf{v}_i \\
&= \mathbf{v}_i^\mathsf{T} \left( \sum_k \mathbf{Q}_k \right) \mathbf{v}_i \\
&= \mathbf{v}_i^\mathsf{T} \mathbf{Q}_i \mathbf{v}_i
\end{aligned}
$$



# Finding Optimal Position

Optimal position $\bar{\mathbf{v}}$ after collapse $(\mathbf{v}_i, \mathbf{v}_j) \rightarrow \mathbf{v}$

$$(\mathbf{Q}_i + \mathbf{Q}_j)\bar{\mathbf{v}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

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

# Overview of Algorithm

Initialization

1. Compute matrix $Q$ for each vertex
2. Compute optimal vertex position for each edge collapse
3. Compute *cost* of each edge collapse
4. Store edge collapses in min-heap with *cost* as key

Simplification

1. Collapse edge on top of min-heap
2. Recompute optimal positions and costs
3. Repeat

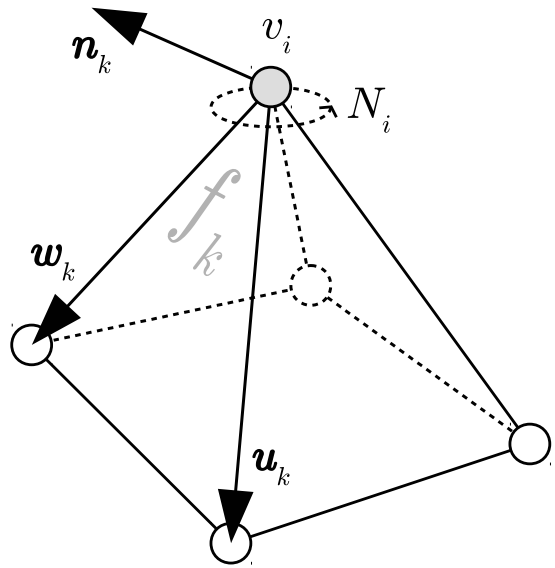
# More convenient notation

$$D^2 = (\mathbf{n}^\mathsf{T}\mathbf{v} + d)^2$$
$$= \mathbf{v}^\mathsf{T}(\mathbf{n}\mathbf{n}^\mathsf{T})\mathbf{v} + 2d\mathbf{n}^\mathsf{T}\mathbf{v} + d^2 \quad (1)$$

$$Q = (\mathbf{n}\mathbf{n}^\mathsf{T}, d\mathbf{n}, d^2)$$
$$= (\mathbf{A}, \mathbf{b}, c) \quad (2)$$

$$Q(\mathbf{v}) = \mathbf{v}^\mathsf{T}\mathbf{A}\mathbf{v} + 2\mathbf{b}^\mathsf{T}\mathbf{v} + c \quad (3)$$

$$\mathbf{A}\bar{\mathbf{v}} + \mathbf{b} = 0 \quad (4)$$



# Quadric Error Metric with Attributes

Extend **Q** to include attributes

$$Q = (\mathbf{A}, \mathbf{b}, c) = \left( \left[ \begin{array}{c|c} \mathbf{n}\mathbf{n}^T & \begin{matrix} \ddots & 0 & \ddots \end{matrix} \\ \hline \begin{matrix} \ddots & 0 & \ddots \end{matrix} & \begin{matrix} \ddots & 0 & \ddots \end{matrix} \end{array} \right], \left[ \begin{array}{c} d\mathbf{n} \\ \hline 0 \end{array} \right], d^2 \right)$$

# Quadric Error Metric with Attributes

Expected attribute value at point $p$:

$$\hat{s}_j(\mathbf{p}) = \mathbf{g}_j^{\mathsf{T}}\mathbf{p} + d_j$$

$\hat{s}_j$ interpolate the vertices of face $\left(\binom{p_1}{s_1}, \binom{p_2}{s_2}, \binom{p_3}{s_3}\right)$

$\mathbf{g}_j$ and $d_j$ obtained by solving:

$$\begin{bmatrix} \mathbf{p}_1^{\mathsf{T}} & 1 \\ \mathbf{p}_2^{\mathsf{T}} & 1 \\ \mathbf{p}_3^{\mathsf{T}} & 1 \\ \mathbf{n}^{\mathsf{T}} & 0 \end{bmatrix} \begin{bmatrix} \\ \mathbf{g}_j \\ \\ d_j \end{bmatrix} = \begin{bmatrix} s_{1,j} \\ s_{2,j} \\ s_{3,j} \\ 0 \end{bmatrix}$$

# Quadric Error Metric with Attributes

$$\mathbf{A} = \left[\begin{array}{c|c} \mathbf{n}\mathbf{n}^T + \sum_j \mathbf{g}_j\mathbf{g}_j^T & -\mathbf{g}_1 \cdots -\mathbf{g}_m \\ \hline -\mathbf{g}_1 & \\ \vdots & \mathrm{I} \\ -\mathbf{g}_m & \end{array}\right]$$

$$\mathbf{b} = \left[\begin{array}{c} d\mathbf{n} + \sum_j d_j\mathbf{g}_j \\ \hline -d_1 \\ \vdots \\ -d_m \end{array}\right]$$

$$c = d^2 + \sum_j d_j^2$$

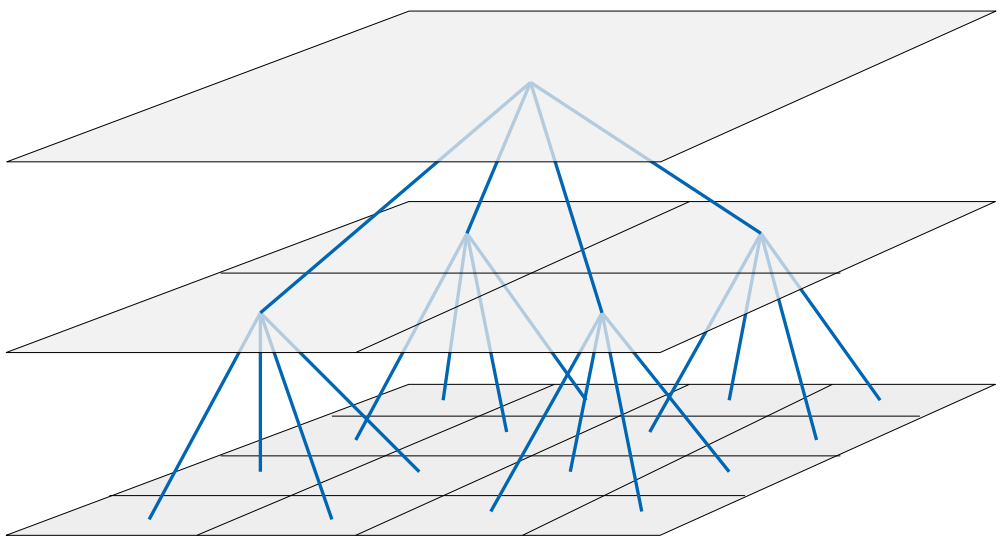
# Outline

# Problem with Texture Atlas

Bad texture values in seams

# Improving Texture

- Use a pull-push algorithm to fill invalid pixels
- Creates a pyramid of images with decreasing resolution
- Each pixel is assigned weight $w_i$ and color $x_i$



# Find valid pixels

- Create mesh with UV-coordinates as vertices
- Cast rays toward the mesh to find valid pixels

# Find valid pixels

- Could also be obtained with a threshold-filter
- Apply edge-filter to trim edges

# Pull Phase

- Create lower resolution level with Gaussian blur filter

$$w_i^{r+1} = \sum_k \tilde{h}_k \min(w_k^r, 1)$$

$$x_i^{r+1} = \frac{1}{w_i^{r+1}} \sum_k \tilde{h}_k \min(w_k^r, 1) x_i^r$$

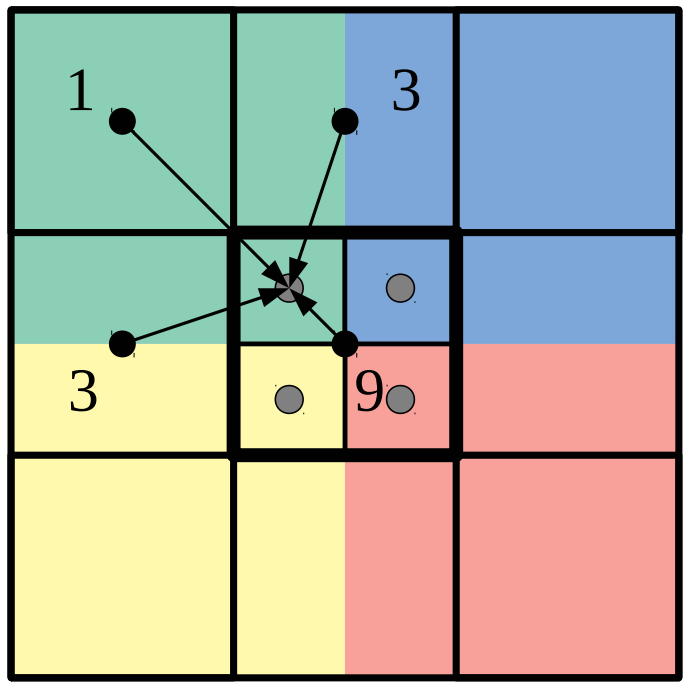
| | | | |
|---|---|---|---|
| 1 | 2 | 2 | 1 |
| 2 | 4 | 4 | 2 |
| 2 | 4 | 4 | 2 |
| 1 | 2 | 2 | 1 |

# Push Phase

- Blend neighboring pixels

$$tw_i^r = \sum_k h_k \min(w_k^{r+1}, 1)$$

$$tx_i^r = \frac{1}{tw_i^r} \sum_k h_k \min(w_k^{r+1}, 1) x_i^{r+1}$$




- Blend higher resolution pixels with the computed value

$$x_i^r = tx_i^r(1 - w_i^r) + w^r x_i^r$$
$$w_i^r = tw_i^r(1 - w_i^r) + w^r$$

# Pyramid in Pull Phase

# Pyramid in Push Phase

# Outline



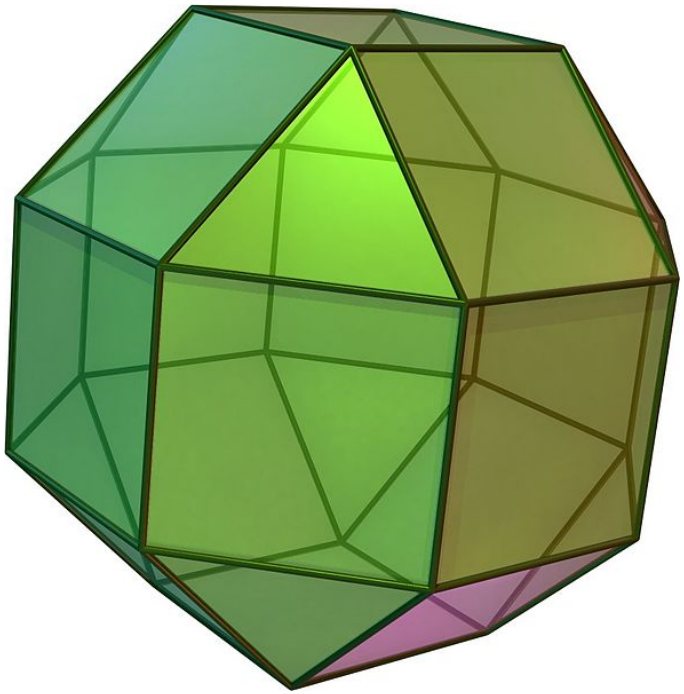
LINKÖPINGS UNIVERSITET

# Appearance Preservation

- Two methods for model comparison
  - Image comparison
  - Hausdorff distance
- The original model is compared to four LoD:s

# Image comparison

- Render the models to compare from multiple positions
- Measure RMS of luminance between the images



# Hausdorff distance

- Sample points on the surface of the model
- Measure the distance to the other surface
- The error is then the mean distance

$$e(p, S') = \min_{p' \in S'} \left\| p' - p \right\|$$

$$E(S, S') = \max_{p \in S} e(p, S')$$

# Volume Preservation

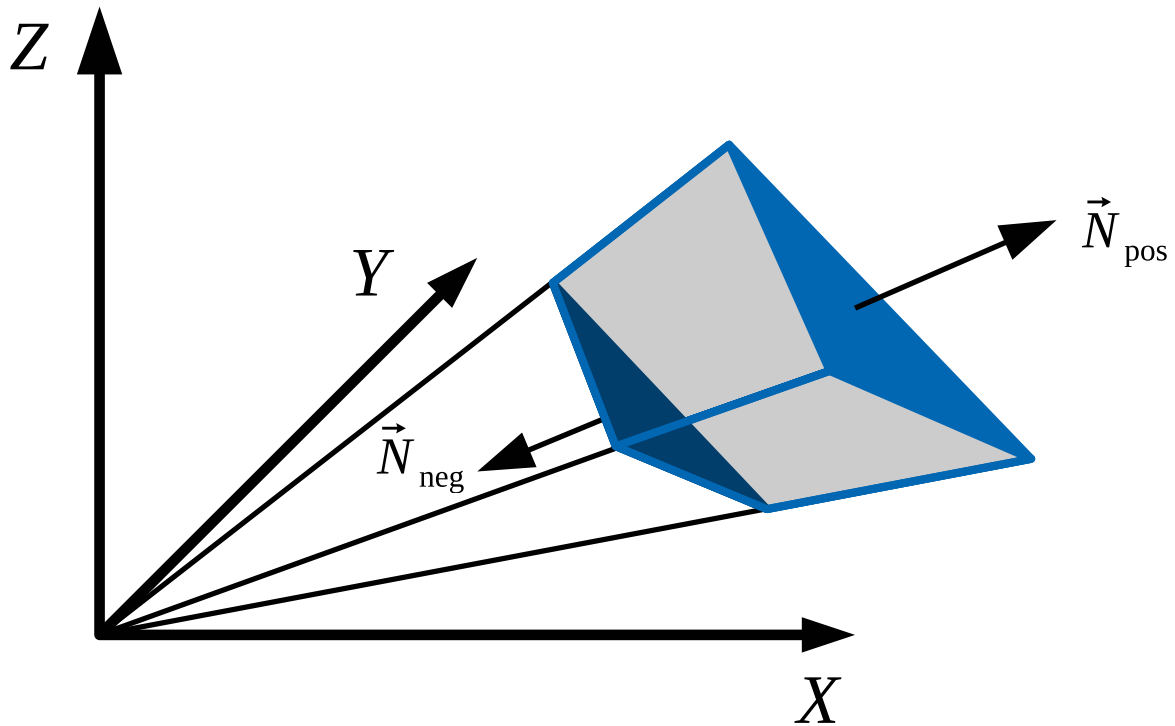
Vertices of a polygon and the origin creates a tetrahedron

Signed volume of a tetrahedron can be calculated with

$$V_i = \frac{v_0 \cdot (v_1 \times v_2)}{6}$$

Volume of a mesh is then the sum of the signed volumes

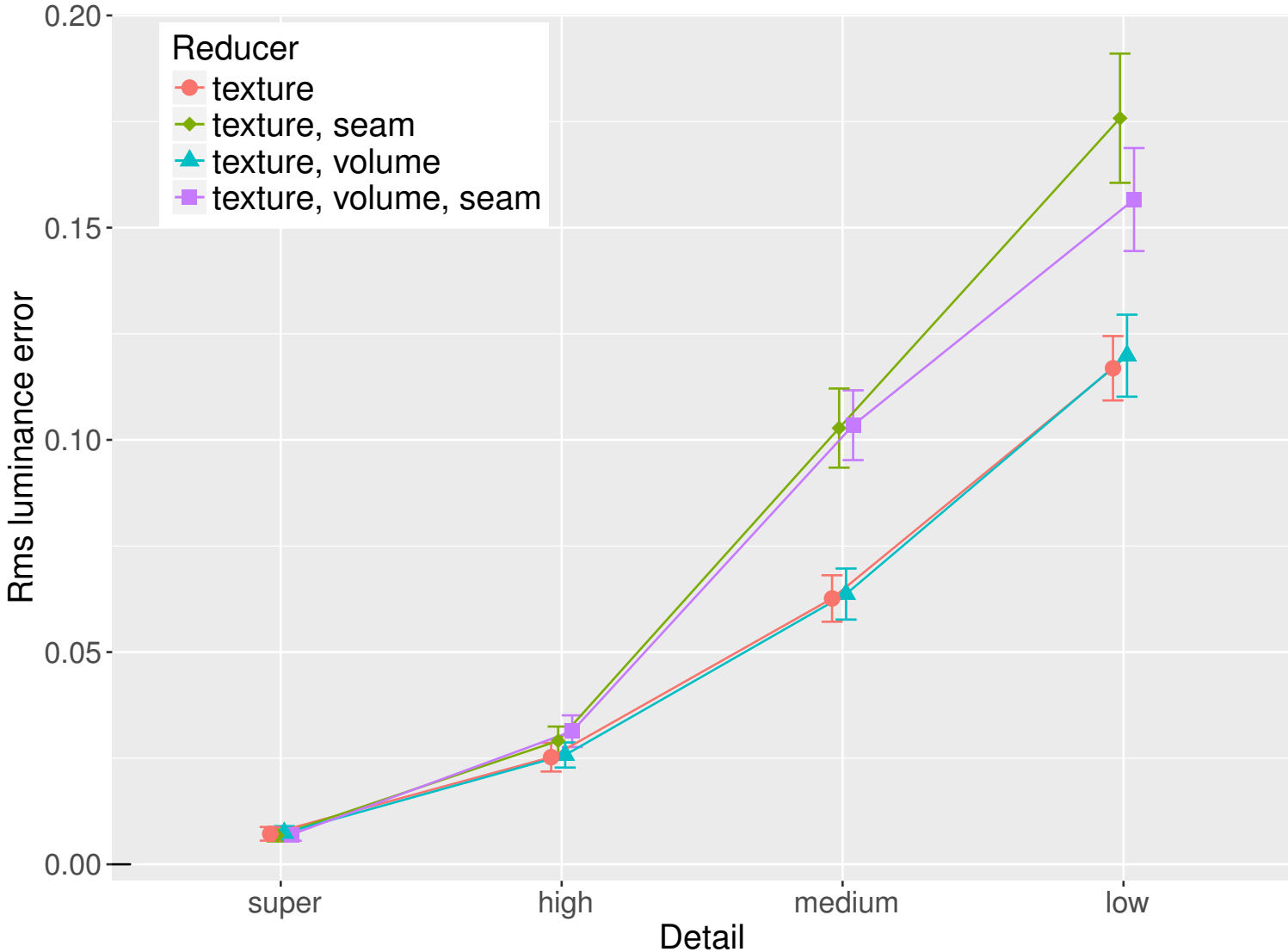
$$V_{total} = \sum_i V_i$$



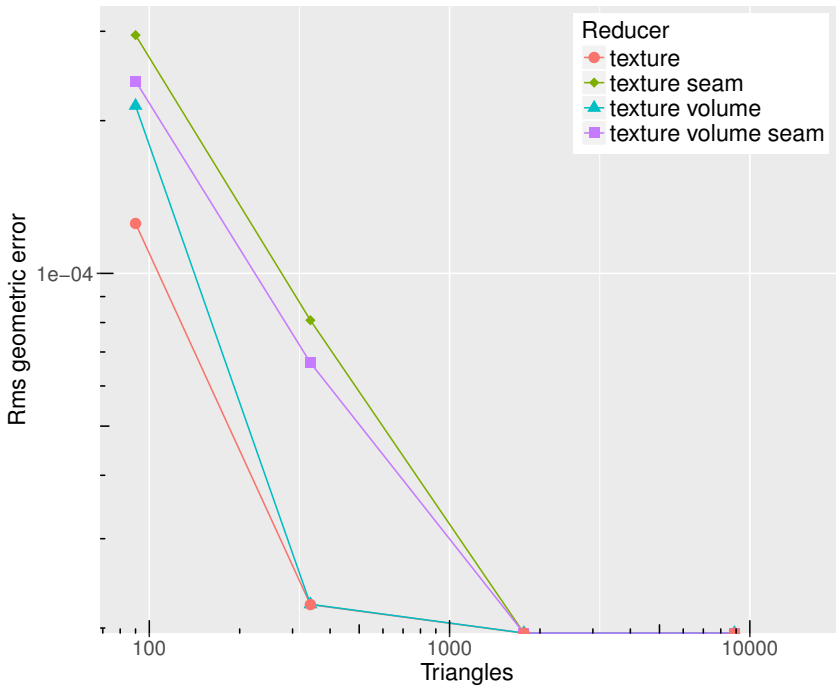
# Execution Time

# Outline

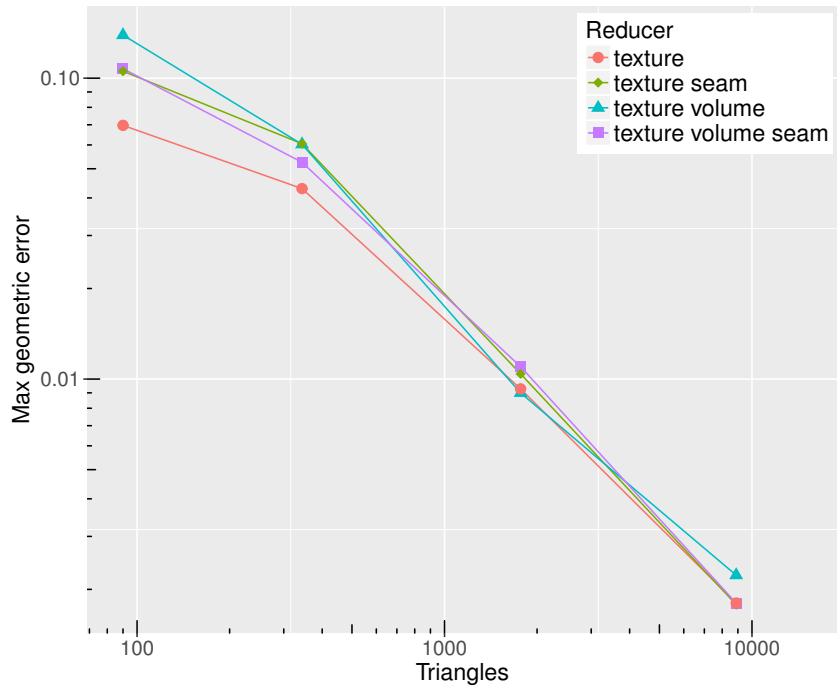
# Rms Luminance Error



# Geometric Error

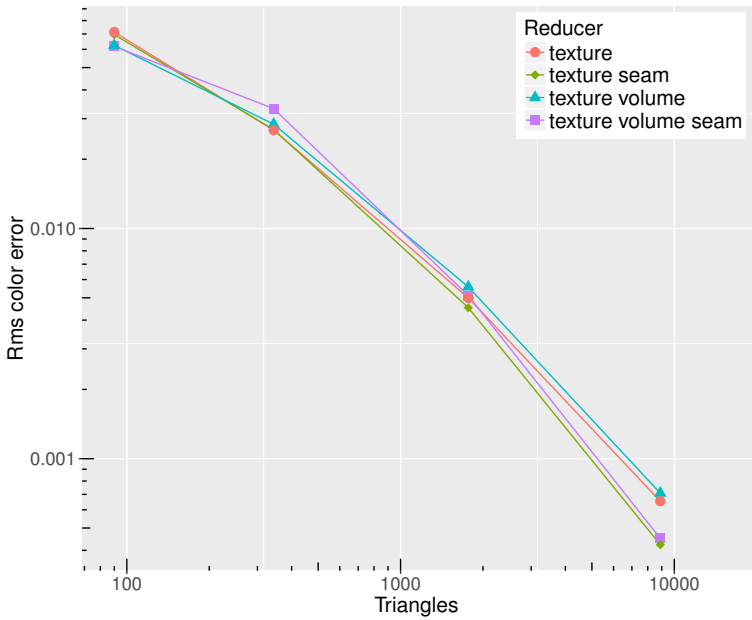



Rms geometric error

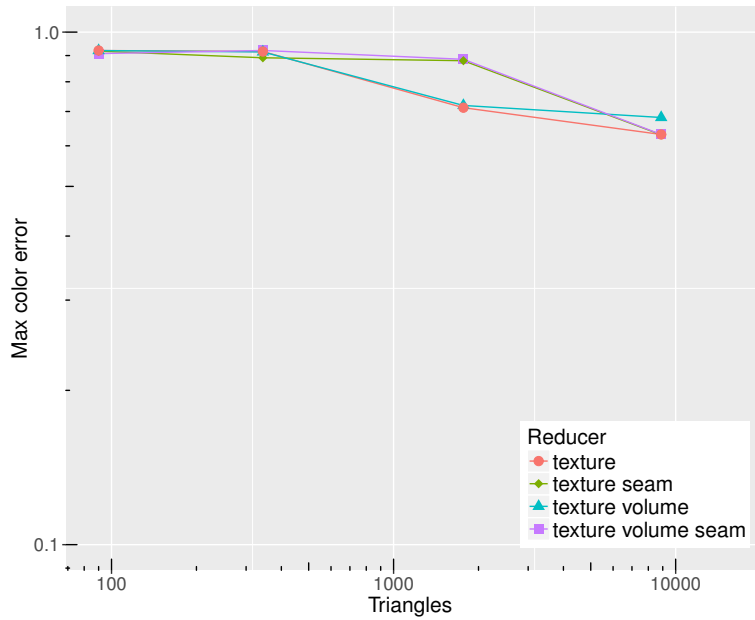



Max geometric error

# Color error

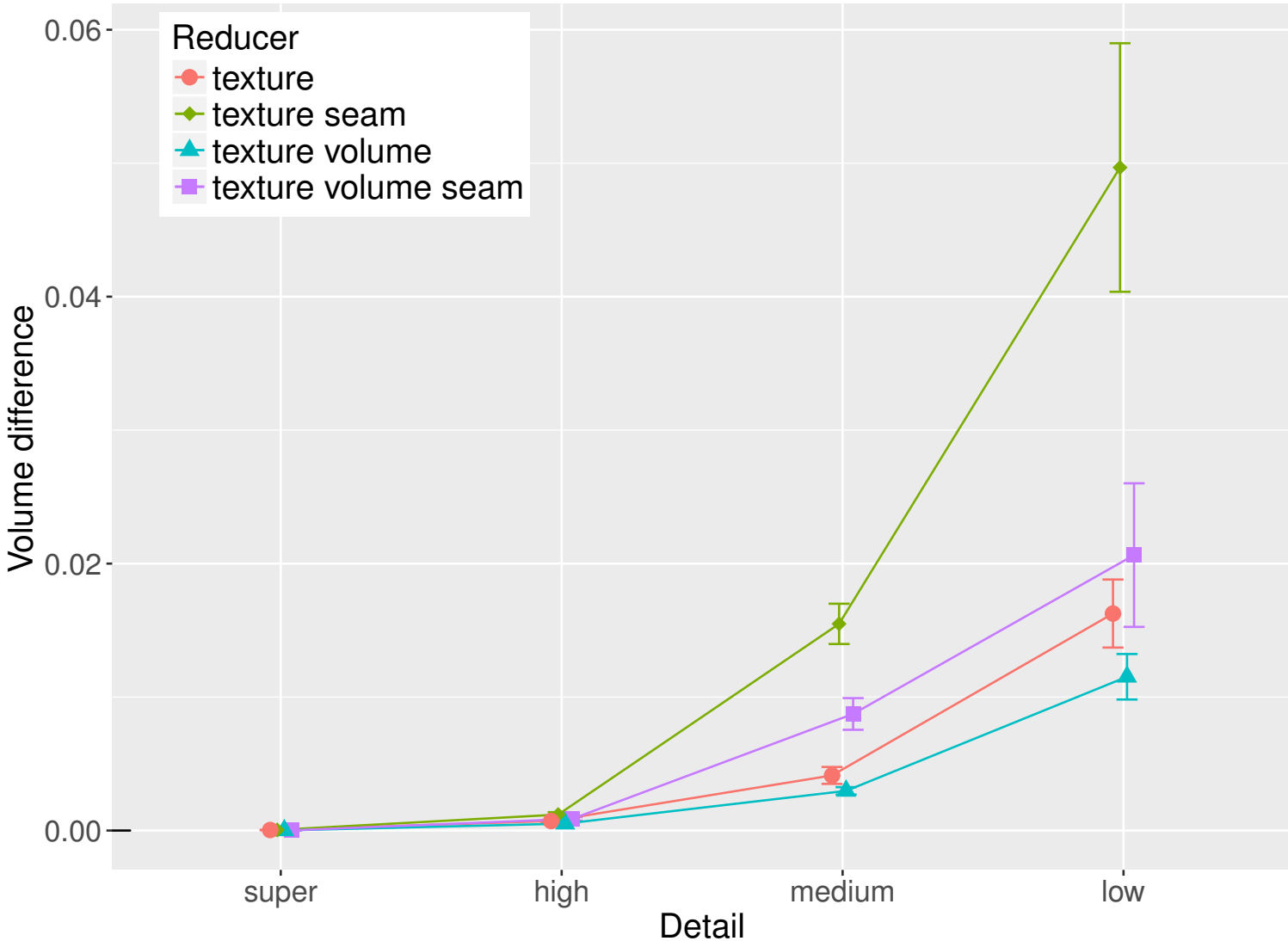



Rms color error

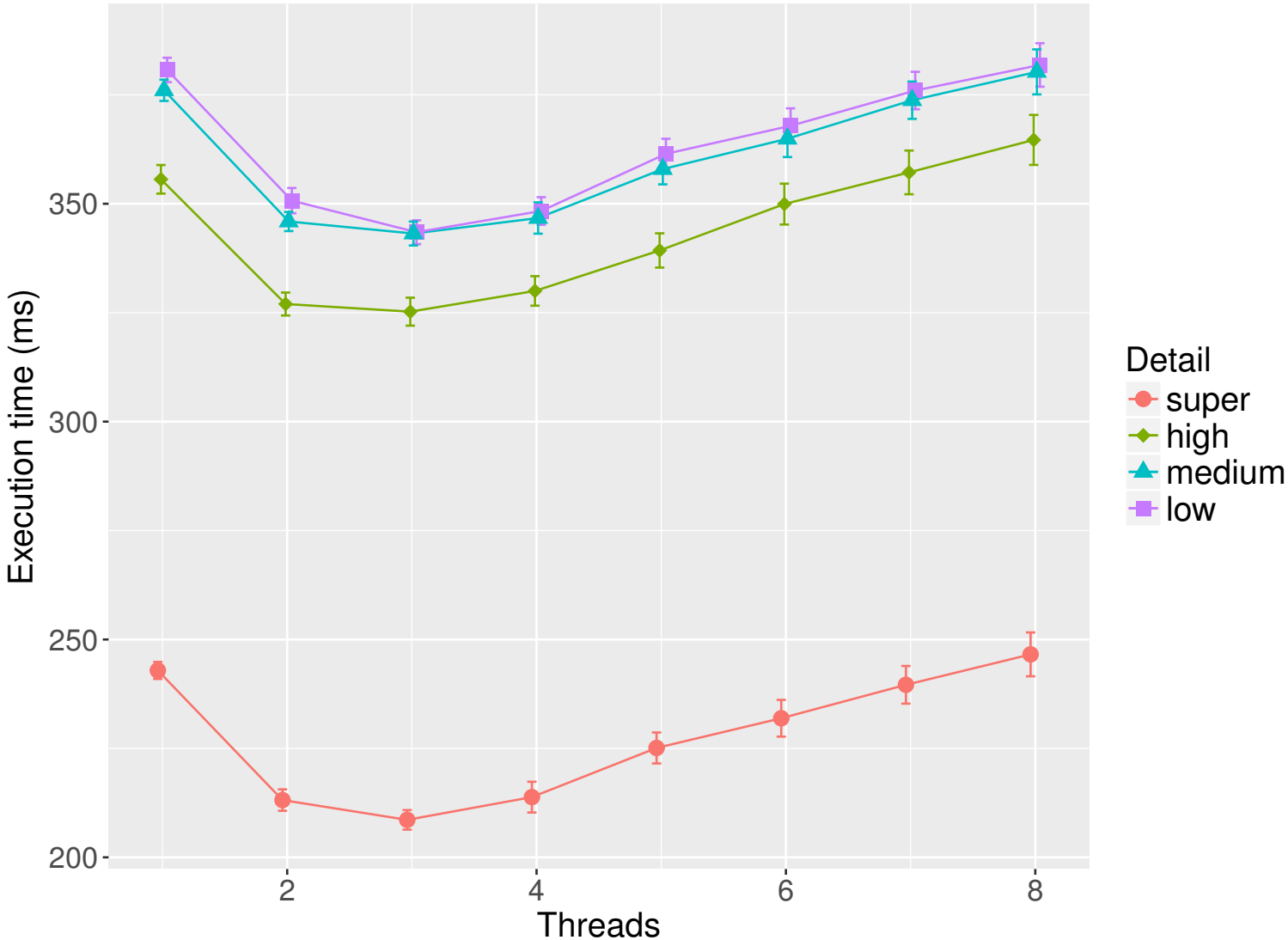



Max color error

# Volume


Reducer
texture
texture seam
texture volume
texture volume seam
Volume difference
0.06
0.04
0.02
0.00
super
high
medium
low
Detail

# Execution Time



# Improving Texture

Original

Bound

Improved

# Improving Texture

# LoD:s (only geometry)

Super High Medium Low

# LoD:s (geometry and texture)

Super High Medium Low

# LoD:s (geometry and texture)

Super

High

Medium

Low

# Summary

- The first main message of your talk in one or two lines.
- The second main message of your talk in one or two lines.
- Perhaps a third message, but not more than that.

- Outlook
  - Something you haven't solved.
  - Something else you haven't solved.

LINKÖPINGS UNIVERSITET