# Laplacian Mesh Optimization

Andrew Nealen[1], Takeo Igarashi[2], Olga Sorkine[1], Marc Alexa[1]
1.TU Berlin, 2.The University of Tokyo

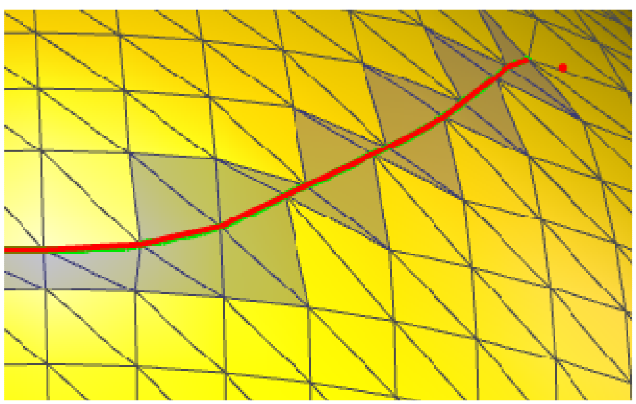
Presented by **Ming Chuang** on 11/04/2007

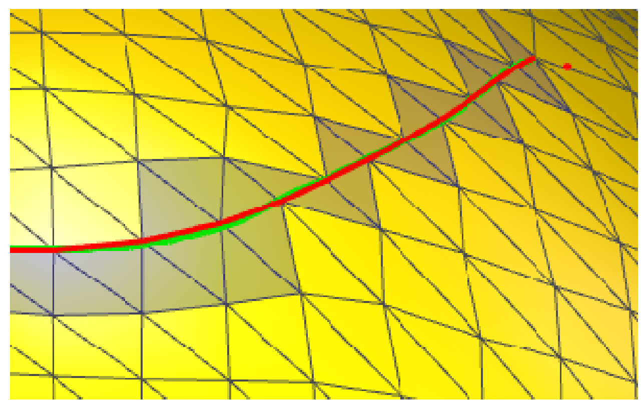
# Outline

- What is the goal?
- Basics and Notation
- Related Work
- Framework
- Global Triangle Shape Optimization
- Mesh Smoothing
- Conclusion

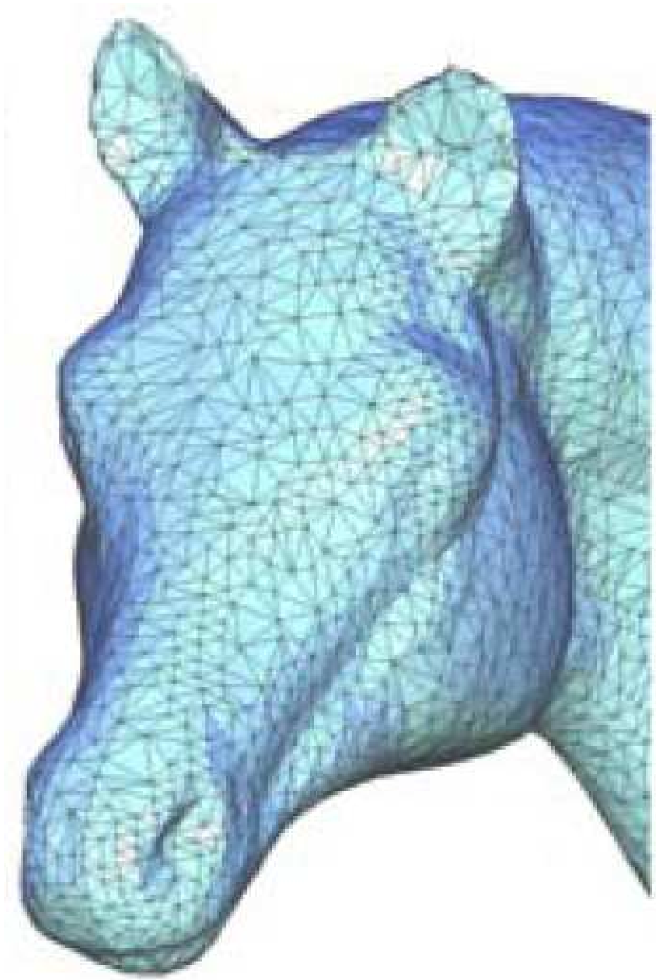
# What is the goal?

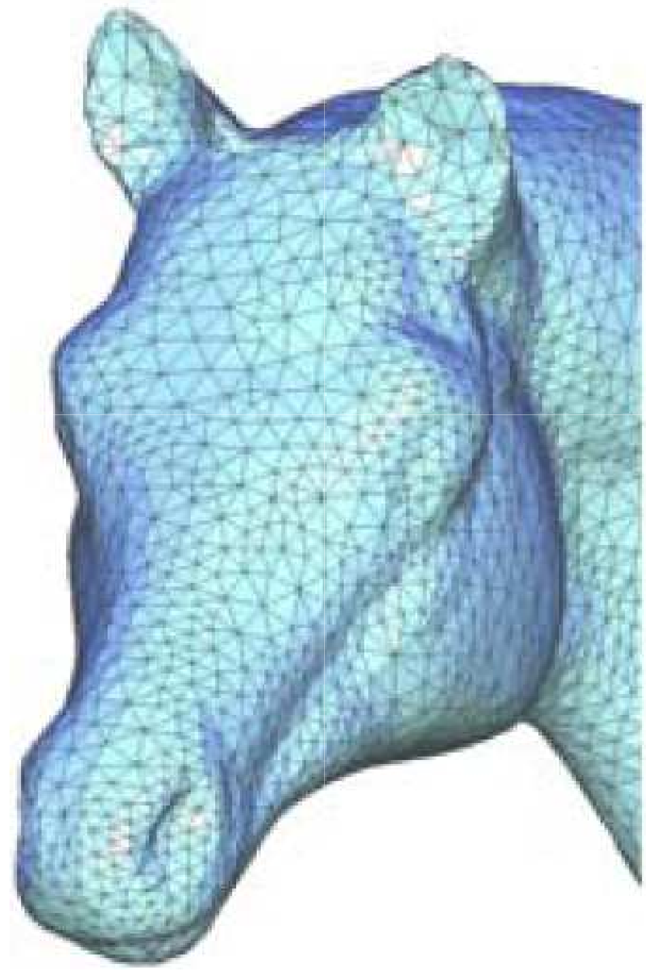
- A simple and efficient framework for:
  - Triangle Shape Optimization
  - Feature Preserving Smoothing





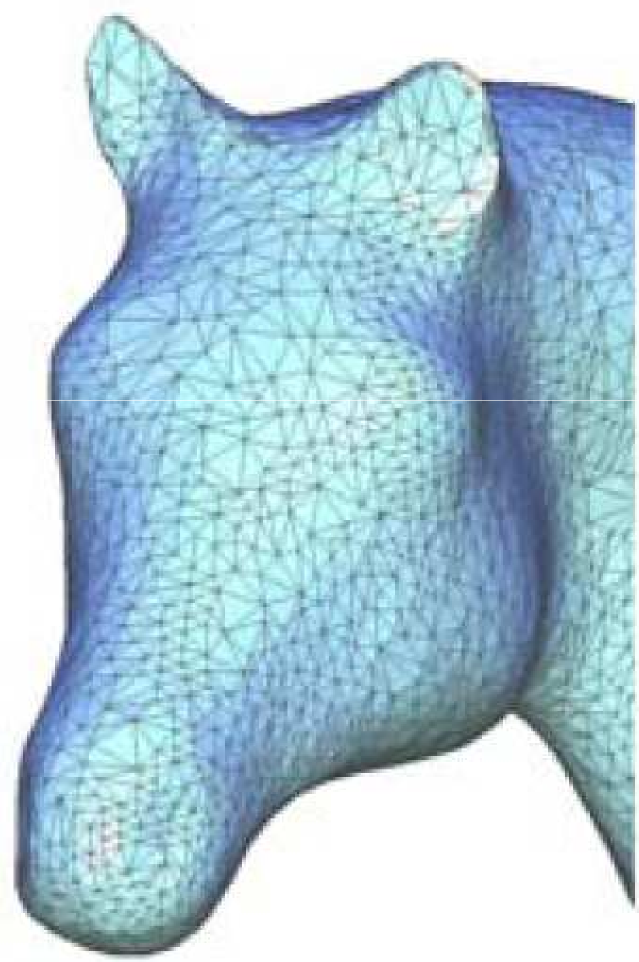
- Original Mesh





- Triangle shape optimization

- Feature Preserving Smoothing



# Basics and Notation

$$\mathbf{V} = [\mathbf{v}_1^T, \mathbf{v}_2^T, \ldots, \mathbf{v}_n^T]^T,\ \mathbf{v}_i = [v_{ix}, v_{iy}, v_{iz}]^T \in \mathbb{R}^3$$

$\delta_i$ is the Laplacian of $\mathbf{v}_i$ $\qquad \sum_{\{i,j\}\in E} w_{ij} = 1$

$$\delta_i = \sum_{\{i,j\}\in\mathbf{E}} w_{ij}(\mathbf{v}_j - \mathbf{v}_i) = \left[\sum_{\{i,j\}\in\mathbf{E}} w_{ij}\mathbf{v}_j\right] - \mathbf{v}_i, \tag{1}$$

$$w_{ij} = \frac{\omega_{ij}}{\sum_{\{i,k\}\in\mathbf{E}} \omega_{ik}} \tag{2}$$

$$\omega_{ij} = 1, \tag{3}$$

$$\omega_{ij} = cot\ \alpha + cot\ \beta, \tag{4}$$

# Basics and Notation

- To obtain the Laplacian for the entire mesh, we use the n by n Laplacian matrix L, with elements:

$$\mathbf{L}_{ij} = \begin{cases} -1 & i = j \\ w_{ij} & (i,j) \in \mathbf{E} \\ 0 & \text{otherwise} \end{cases} , \qquad (5)$$

- $L_u$ and $L_c$ denote the matrices with uniform and cotangent weights respectively

# Basics and Notation

- $V_d = [v_{1d}, v_{2d}, \ldots, v_{nd}]^T, d \in \{x, y, z\}$
- $\Delta_d = [\delta_{1d}, \delta_{2d}, \ldots, \delta_{nd}]^T, d \in \{x, y, z\}$

$$\Delta_d = \mathbf{L}\mathbf{V}_d. \qquad (6)$$

- When properly scaled by the Voronoi region as Meyer et al. [2003], we obtain:

$$\overline{\kappa}_i \mathbf{n}_i = \delta_{i,c\overline{\kappa}} = \frac{1}{4A(\mathbf{v}_i)} \sum_{\{i,j\} \in \mathbf{E}} (cot\ \alpha + cot\ \beta)(\mathbf{v}_j - \mathbf{v}_i), \qquad (7)$$

- which is the discrete mean curvature normal

# Previous Work

- Least Square Meshes
  - Sorkine and Cohen-Or [2004] use a small subset $\mathbf{C} \subset \mathbf{V}$ of geometrically constrained vertices (anchors) to construct the mesh.
  - They try to solve

$$\mathbf{V}'_d = [v'_{1d}, v'_{2d}, \ldots, v'_{nd}]^T, d \in \{x, y, z\}$$

by minimizing the quadratic energy:

$$\|\mathbf{L}_u \mathbf{V}'_d\|^2 + \sum_{s \in \mathbf{C}} w_s^2 |v'_{sd} - v_{sd}|^2, \qquad (8)$$

where $v_{sd}$ are the anchors

$w_s^2$ are weighting factors

# Previous Work

– In practice, with the anchors as the first m vertices, the (n + m) x n overdetermined linear system $\mathbf{A}\mathbf{V}'_d = \mathbf{b}$

$$\left[\begin{array}{c|c} \multicolumn{2}{c}{\mathbf{L}_u} \\ \hline \mathbf{I}_{m\times m} & \mathbf{0} \end{array}\right] \mathbf{V}'_d = \left[\begin{array}{c} \mathbf{0} \\ \hline \mathbf{V}_{(1\ldots m)d} \end{array}\right] \tag{9}$$

can be solved in the least squares sense using normal equation: $\mathbf{V}'_d = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$

– Note that the first n rows are Laplacian constraints, while the last m rows are positional constraints.

# Previous Work

- Detail Preserving Triangle Shape Optimization:
  - Nealen et al. [2005] show how a least square optimization can improve triangle quality in a small mesh region, by modifying the above equation (9):

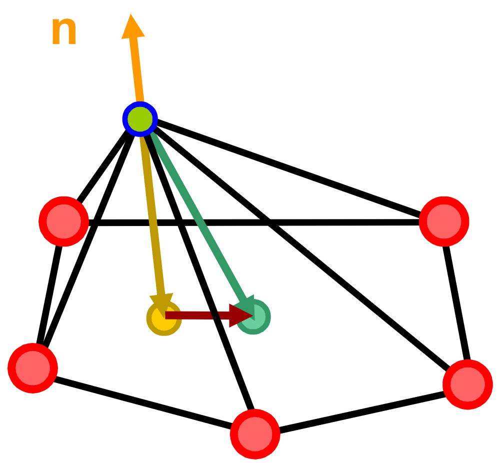
$$\left[\begin{array}{c|c} \multicolumn{2}{c}{\mathbf{L}_u} \\ \hline \mathbf{I}_{m\times m} & \mathbf{0} \end{array}\right] \mathbf{V}'_d = \left[\begin{array}{c} \Delta_{d,c} \\ \hline \mathbf{V}_{(1\ldots m)d} \end{array}\right], \qquad (10)$$

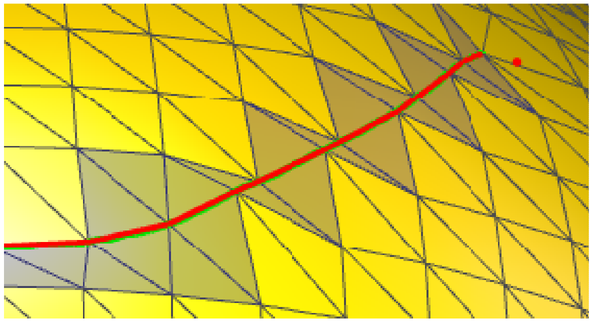
where the uniform Laplacian of each new vertex position is asked to resemble its undeformed cotangent Laplacian as closely as possible

# Previous Work

- Detail Preserving Triangle Shape Optimization:
  - Nealen et al. [2005] show how a least square optimization can improve triangle quality in a small mesh region, by modifying the above equation (9):

$$\left[\begin{array}{c|c} \multicolumn{2}{c}{\mathbf{L}_u} \\ \hline \mathbf{I}_{m\times m} & \mathbf{0} \end{array}\right] \mathbf{V}'_d = \left[\begin{array}{c} \Delta_{d,c} \\ \hline \mathbf{V}_{(1\ldots m)d} \end{array}\right], \qquad (10)$$

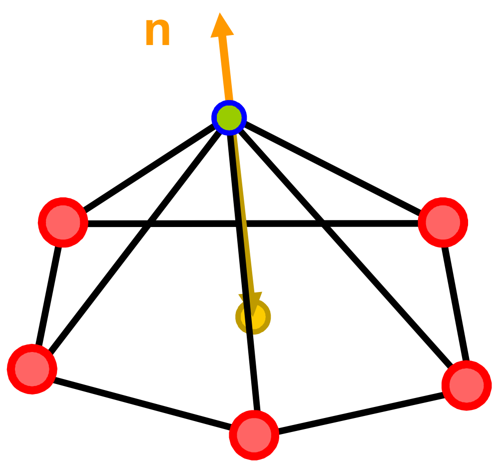


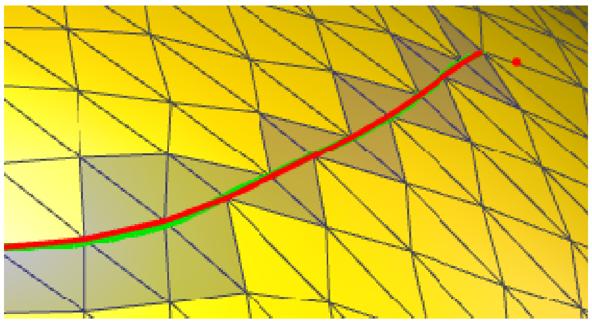



# Previous Work

- Detail Preserving Triangle Shape Optimization:
  - Nealen et al. [2005] show how a least square optimization can improve triangle quality in a small mesh region, by modifying the above equation (9):

$$\left[\begin{array}{c|c} \multicolumn{2}{c}{\mathbf{L}_u} \\ \hline \mathbf{I}_{m\times m} & \mathbf{0} \end{array}\right] \mathbf{V}'_d = \left[\begin{array}{c} \Delta_{d,c} \\ \hline \mathbf{V}_{(1\ldots m)d} \end{array}\right], \tag{10}$$



# Previous Work

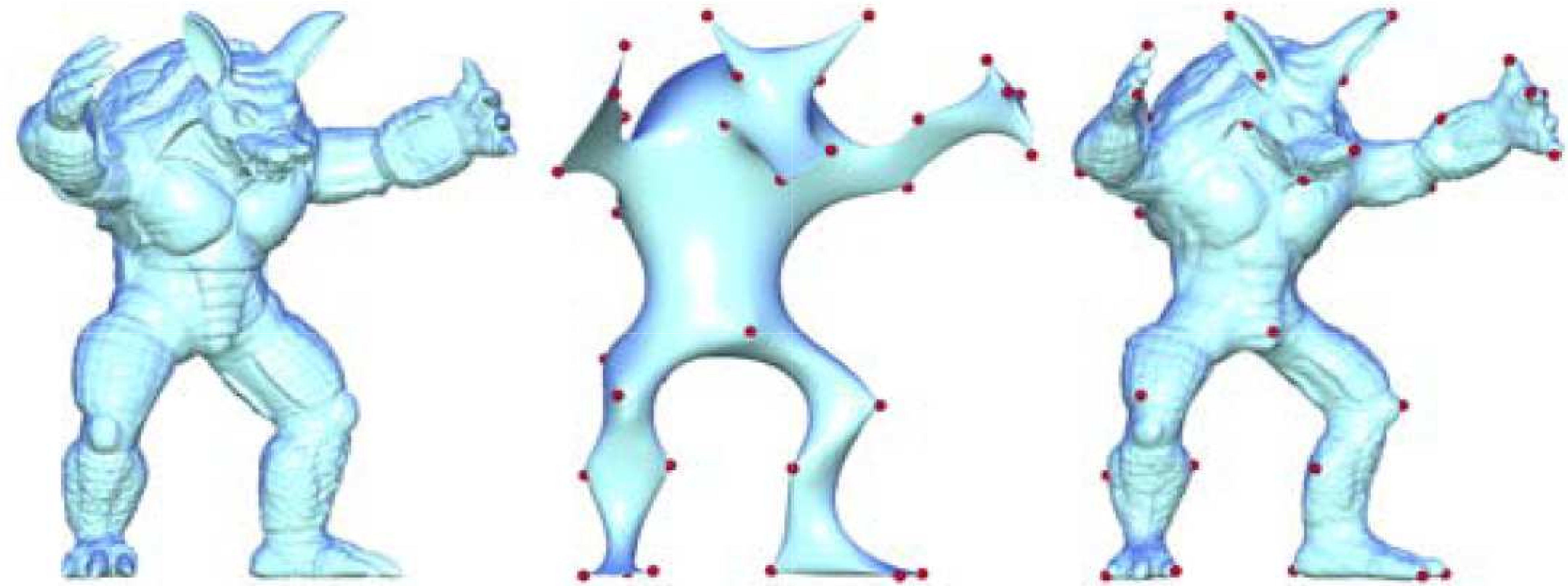
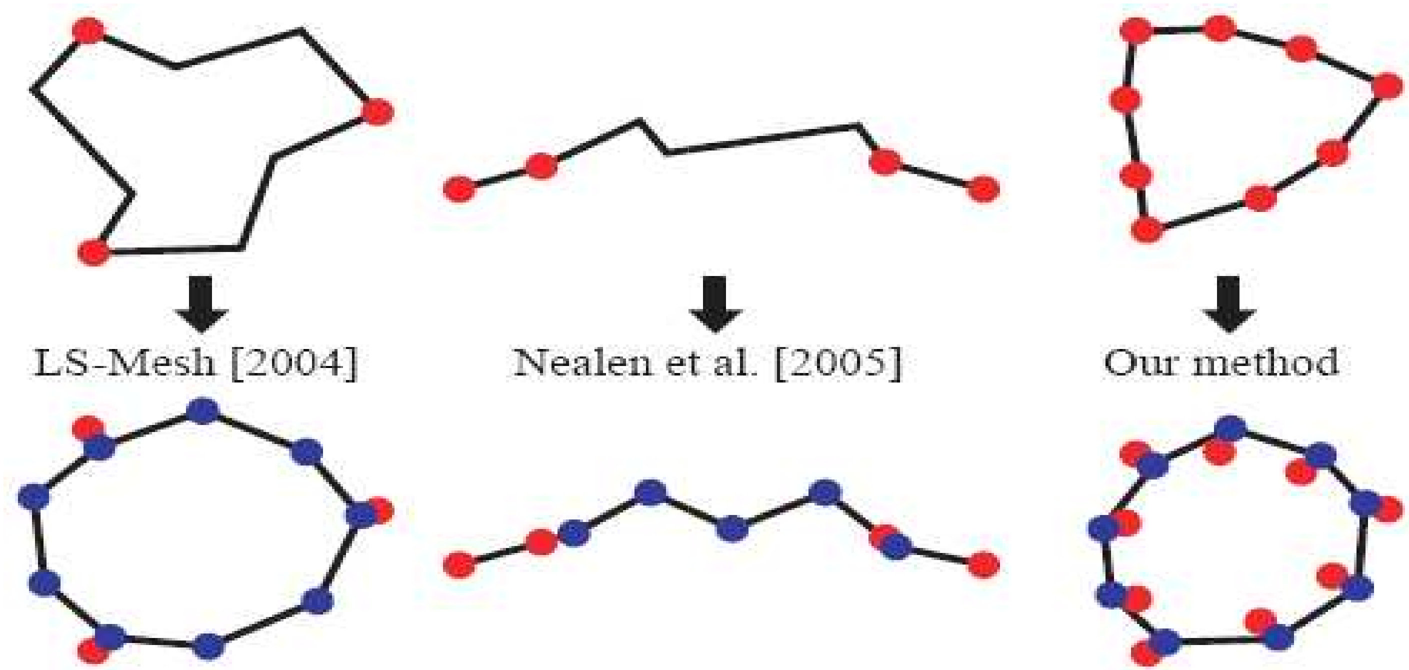


Figure 3: Least squares mesh [Sorkine and Cohen-Or 2004] (middle) and detail preserving triangle shape optimization [Nealen et al. 2005] (right) of the ARMADILLO (left), each with the same set of 40 control vertices (red).

# Framework

- The main idea in this paper is that we no longer have a subset of positional constraints, instead, all vertices appear both as Laplacian and positional constraints....



# Framework

- This paper propose a modification of the previous equation (9) and (10). The new 2n x n system is:

$$\left[\frac{\mathbf{W}_L \mathbf{L}}{\mathbf{W}_p}\right] \mathbf{V}'_d = \left[\frac{\mathbf{W}_L \mathbf{f}}{\mathbf{W}_p \mathbf{V}_d}\right]. \tag{11}$$

- In general, larger weights in $\mathbf{W}_p$ enforce positional constraints and preserve more original geometry
- On the other hand, $\mathbf{W}_L$ enforce regular triangle shapes and/or surface smoothness

# Framework

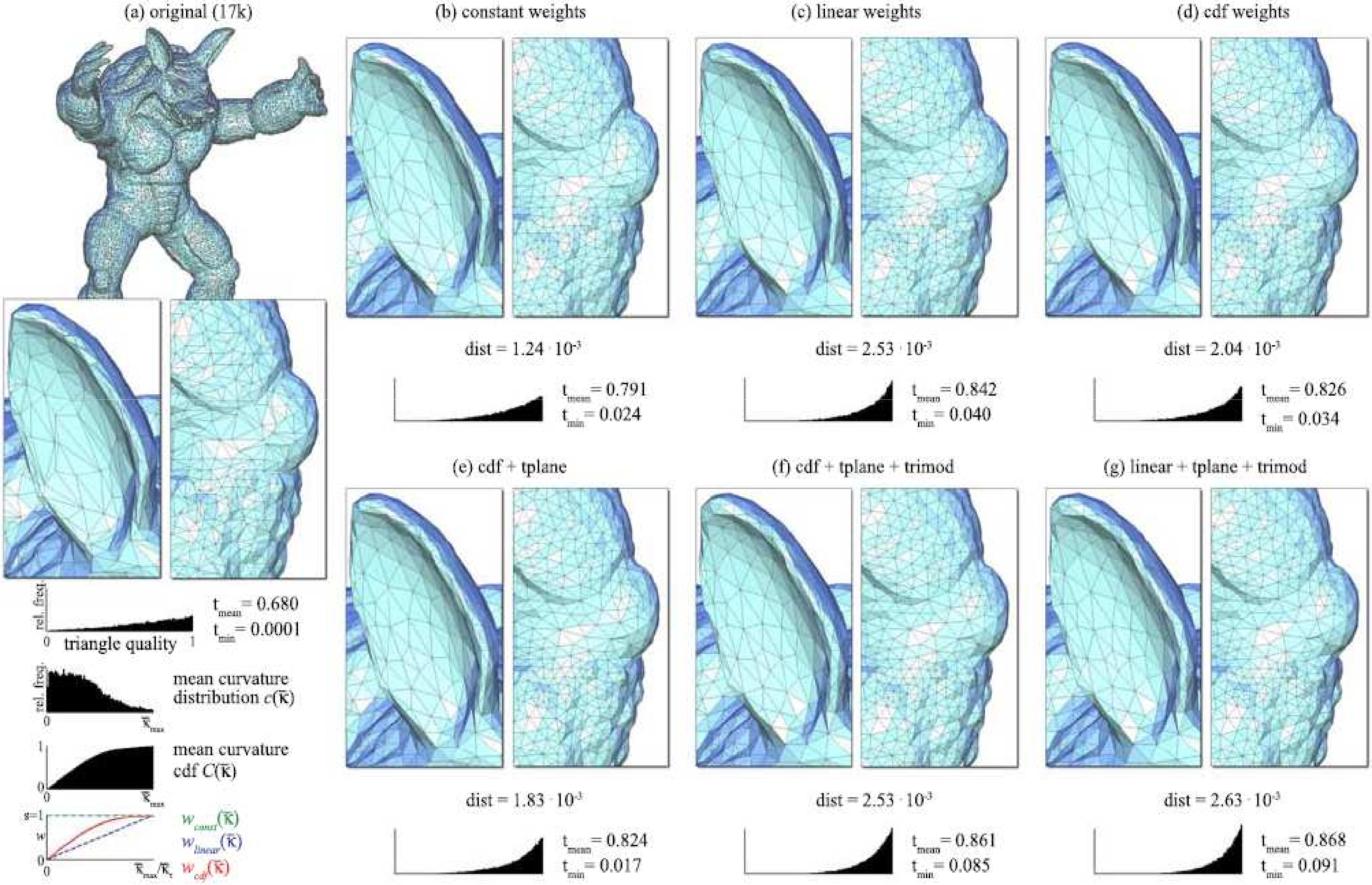
- As we will describe later, solving this system results in either:
  - Detail Preserving Triangle Shape Optimization
    - when setting $\mathbf{L} = \mathbf{L}_u$ and $\mathbf{f} = \Delta_{d,c}$,
  - Mesh Smoothing
    - when setting $\mathbf{L} = \mathbf{L}_{c\overline{K}}$ or $\mathbf{L}_c$ (outer fairness) and $\mathbf{f} = \mathbf{0}$
    - Or $\mathbf{L} = \mathbf{L}_u$ (outer and inner fairness) and $\mathbf{f} = \mathbf{0}$

# Global Triangle Shape Optimization

- We measure the triangle quality using the *radius ratio* [Pebay and Baker 2003], mapped to [0, 1]

$$t_i = 2\frac{r}{R}, \tag{12}$$

- R and r are the radii of the circumscribed and inscribed circles respectively
- $t_i$ =1(regular triangle) indicates it's well shaped, while $t_i$ =0 is degenerate.



(a) original (17k)
(b) constant weights
(c) linear weights
(d) cdf weights
dist = 1.24 · 10⁻³
dist = 2.53 · 10⁻³
dist = 2.04 · 10⁻³
$t_{mean}$ = 0.791
$t_{min}$ = 0.024
$t_{mean}$ = 0.842
$t_{min}$ = 0.040
$t_{mean}$ = 0.826
$t_{min}$ = 0.034
(e) cdf + tplane
(f) cdf + tplane + trimod
(g) linear + tplane + trimod
rel. freq.
0
triangle quality
1
$t_{mean}$ = 0.680
$t_{min}$ = 0.0001
rel. freq.
0
$\bar{\kappa}_{max}$
mean curvature distribution $c(\bar{\kappa})$
1
0
0
$\bar{\kappa}_{max}$
mean curvature cdf $C(\bar{\kappa})$
s=1
w
0
0
$\bar{\kappa}_{max}/\bar{\kappa}_{t}$
$w_{const}(\bar{\kappa})$
$w_{linear}(\bar{\kappa})$
$w_{cdf}(\bar{\kappa})$
dist = 1.83 · 10⁻³
dist = 2.53 · 10⁻³
dist = 2.63 · 10⁻³
$t_{mean}$ = 0.824
$t_{min}$ = 0.017
$t_{mean}$ = 0.861
$t_{min}$ = 0.085
$t_{mean}$ = 0.868
$t_{min}$ = 0.091

- $W_L = I$, $f = \Delta_{d,c}$
- $W_p = ?$

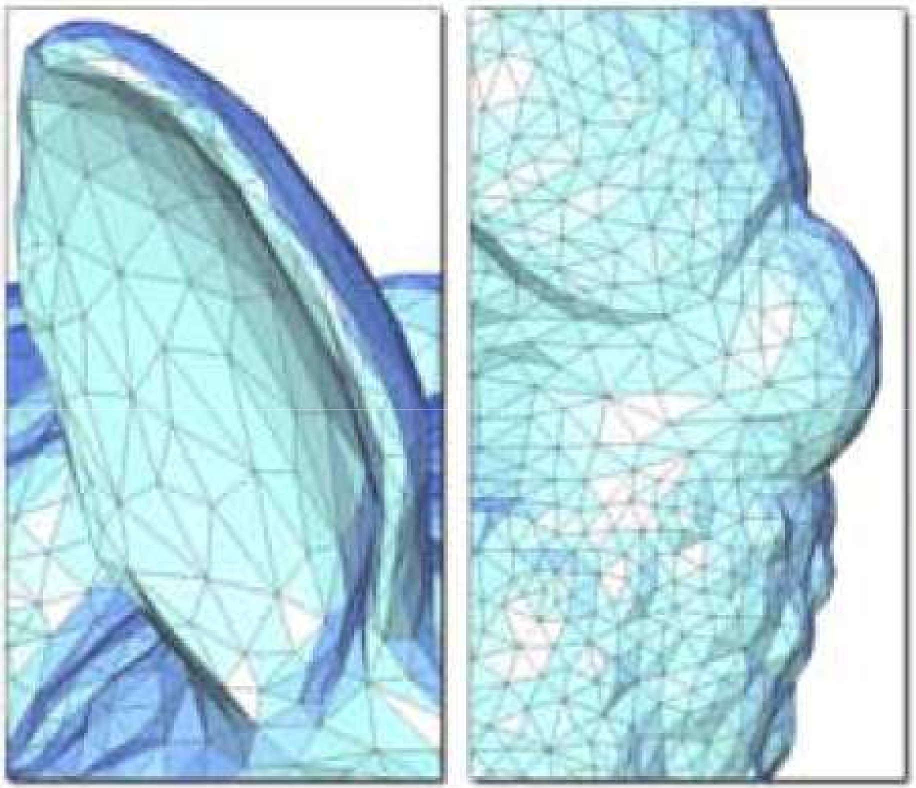
$$\mathbf{W}_p = \mathbf{W}_{const} = s\mathbf{I}$$

  - Lowest geometric error (Hausdorff distance [Cignoni et al. 1998])

$$\mathbf{W}_p = \mathbf{W}_{linear}$$

  - Using a linear curvature-to-weight transfer function, which maps the discrete mean curvature $\overline{\kappa}$ to weight $w_i$ for each vertex.
  - More precisely, we map the interval $[\overline{\kappa}_{min}, \overline{\kappa}_{max}]$ to $[0, s]$
  - Threshold $\overline{\kappa}_\tau$ is also used to truncate outliers

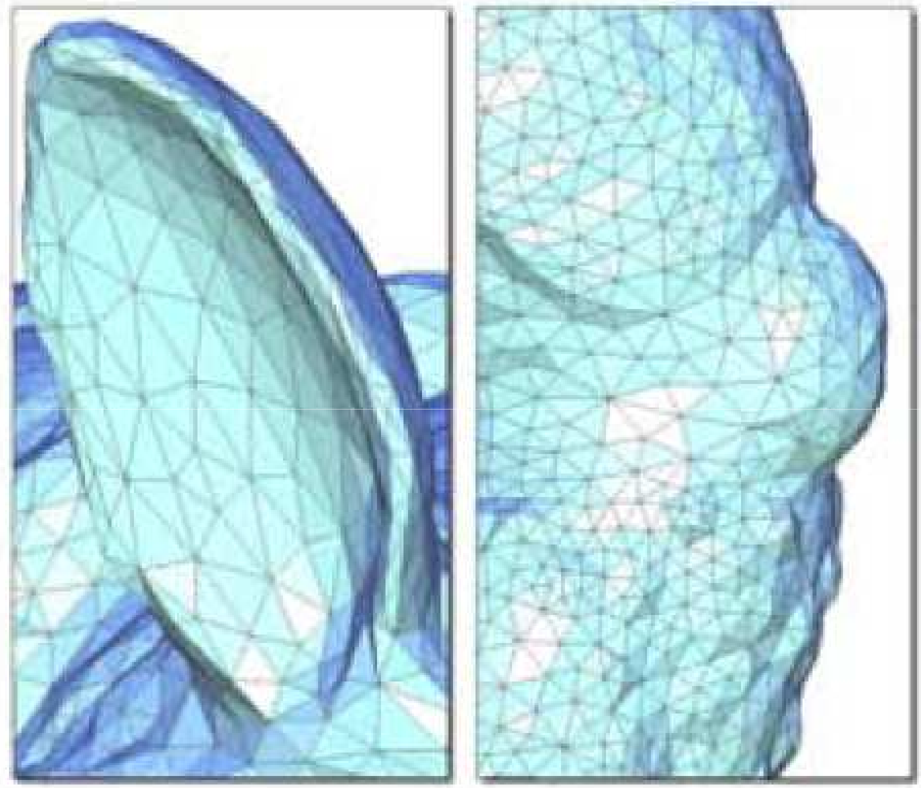
(b) constant weights



dist = $1.24 \cdot 10^{-3}$

$t_{mean} = 0.791$
$t_{min} = 0.024$

(c) linear weights



dist = $2.53 \cdot 10^{-3}$

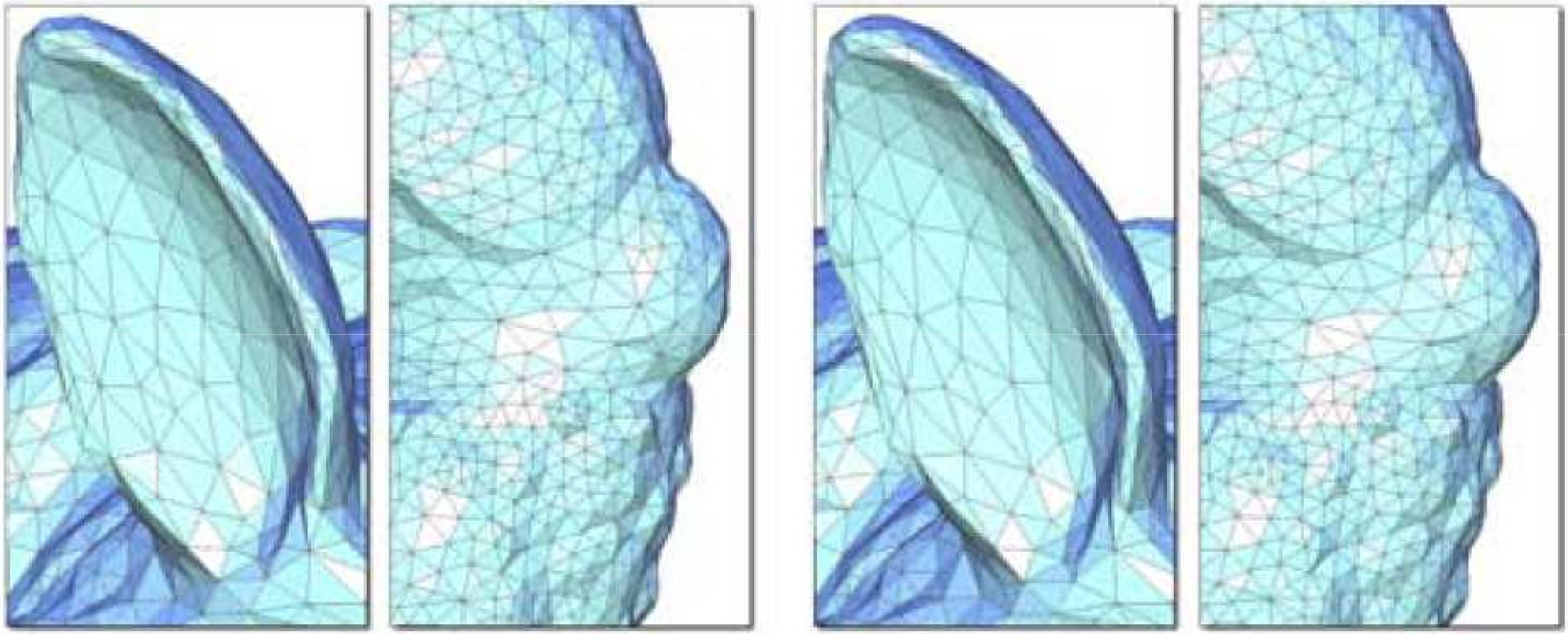
$t_{mean} = 0.842$
$t_{min} = 0.040$

- $\mathbf{W}_p = \mathbf{W}_{cdf}$
  - While $\mathbf{W}_{const}$ constrains low curvature vertices too much, $\mathbf{W}_{linear}$ tends to give them to much freedom
  - For those mesh a large amount of low curvature vertices, we use the cumulative density function (cdf) of $\overline{\kappa}$ to assign them larger weight.

$$C(\overline{\kappa}) = \int_0^{\overline{\kappa}} c(t)\, dt, \qquad (13)$$

(c) linear weights

(d) cdf weights



dist = $2.53 \cdot 10^{-3}$

dist = $2.04 \cdot 10^{-3}$

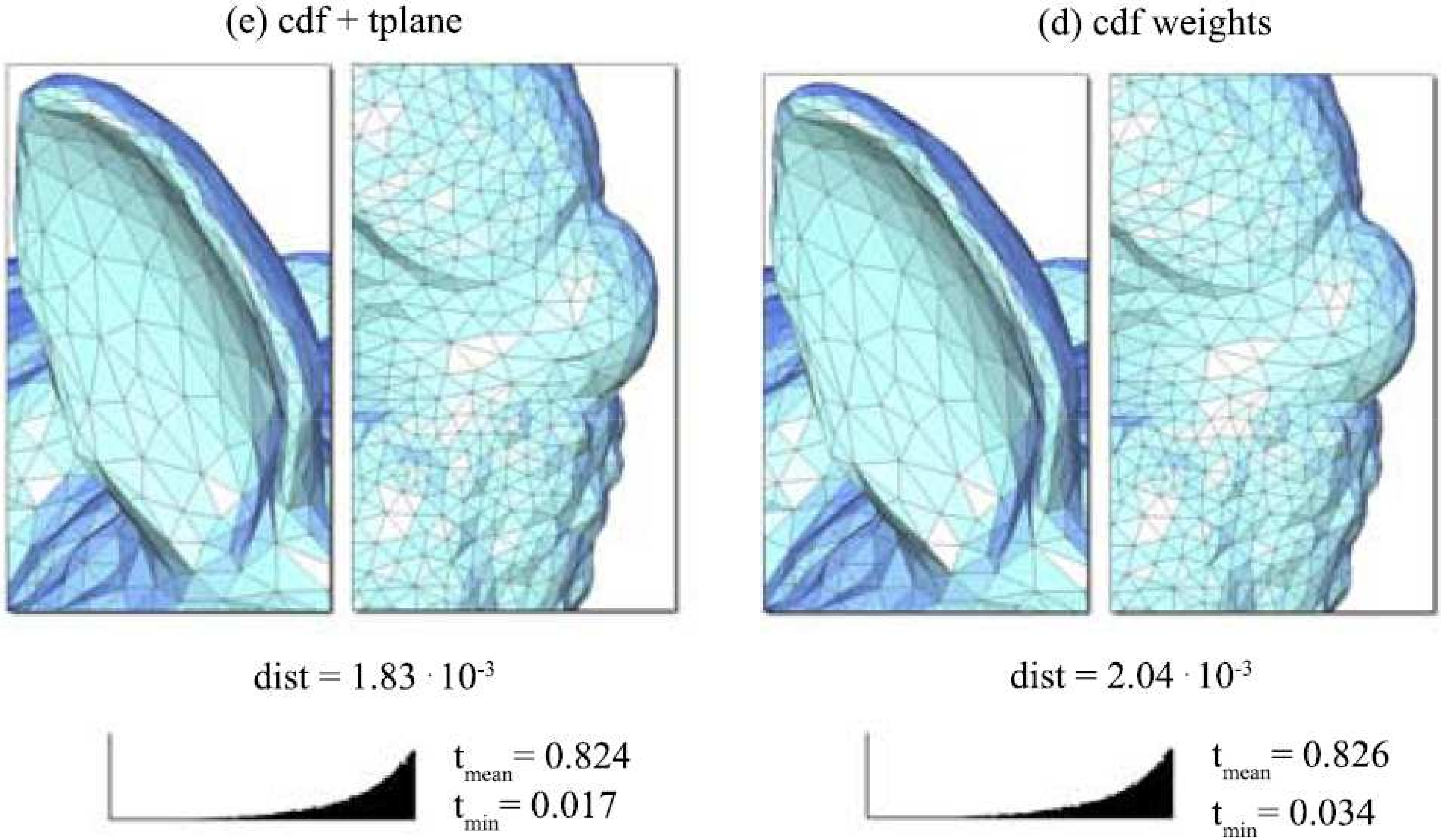
$t_{mean} = 0.842$
$t_{min} = 0.040$

$t_{mean} = 0.826$
$t_{min} = 0.034$

- To further reduce the geometric error, we can add the following term to the energy function:

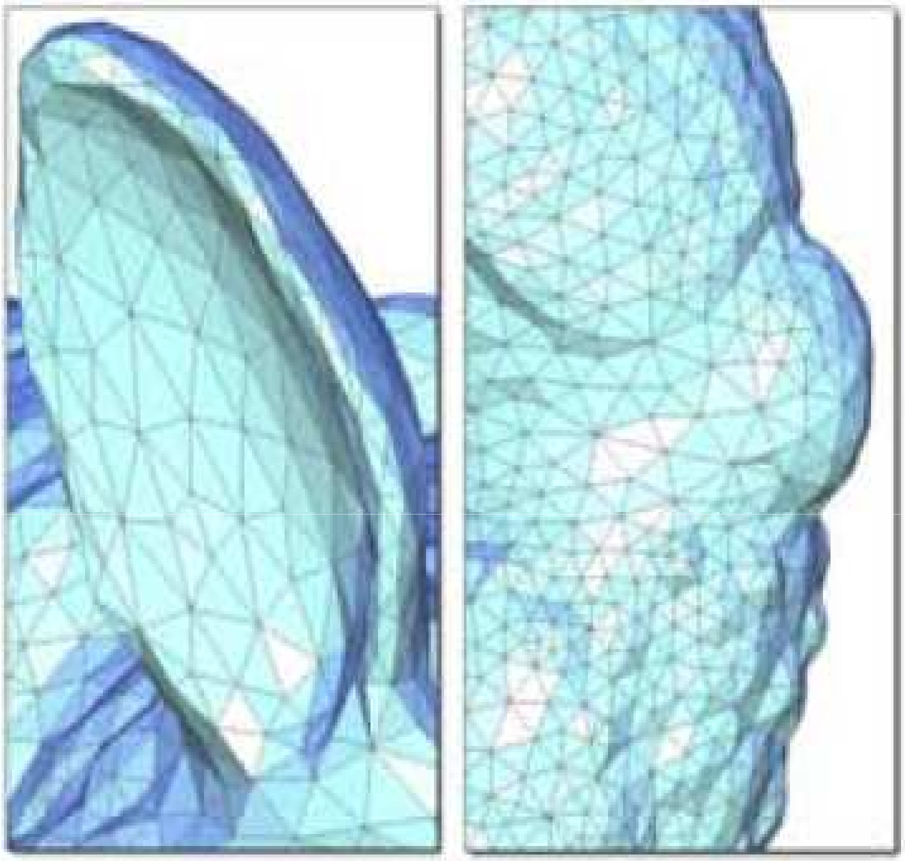
$$\sum_{i=1}^{n} |\mathbf{n}_i \cdot (\mathbf{v}'_i - \mathbf{v}_i)|^2. \qquad (14)$$

- This penalizes the displacement which is perpendicular to the tangent plane defined by the original vertex and the local surface normal given in the equation (7).


(e) cdf + tplane
(d) cdf weights
dist = 1.83 · 10^-3
dist = 2.04 · 10^-3
t_mean = 0.824
t_min = 0.017
t_mean = 0.826
t_min = 0.034

- In some application, we may want to maximize $T_{min}$
- In this case, the positional weights $W_p$ can be modulated with the diagonal matrix $W_t$, where the entry $w_{t,i}$ for vertex $\mathbf{v}_i$ is set to the minimal triangle quality t of its adjacent triangles

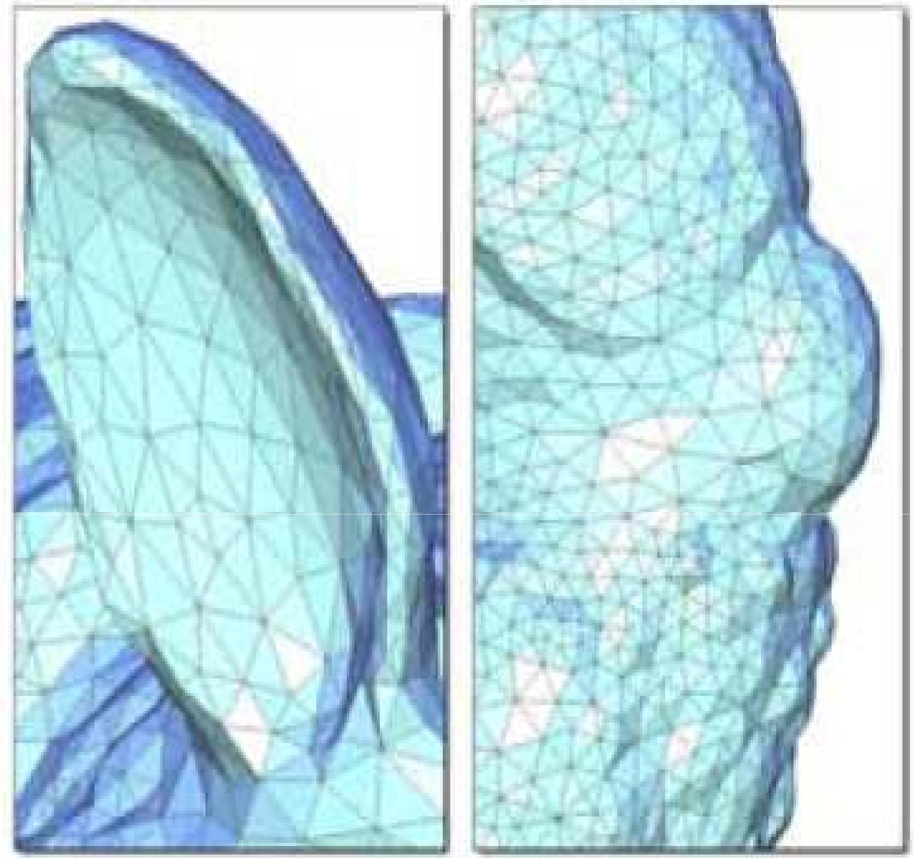
(f) cdf + tplane + trimod



dist = $2.53 \cdot 10^{-3}$

$t_{mean} = 0.861$
$t_{min} = 0.085$

(g) linear + tplane + trimod

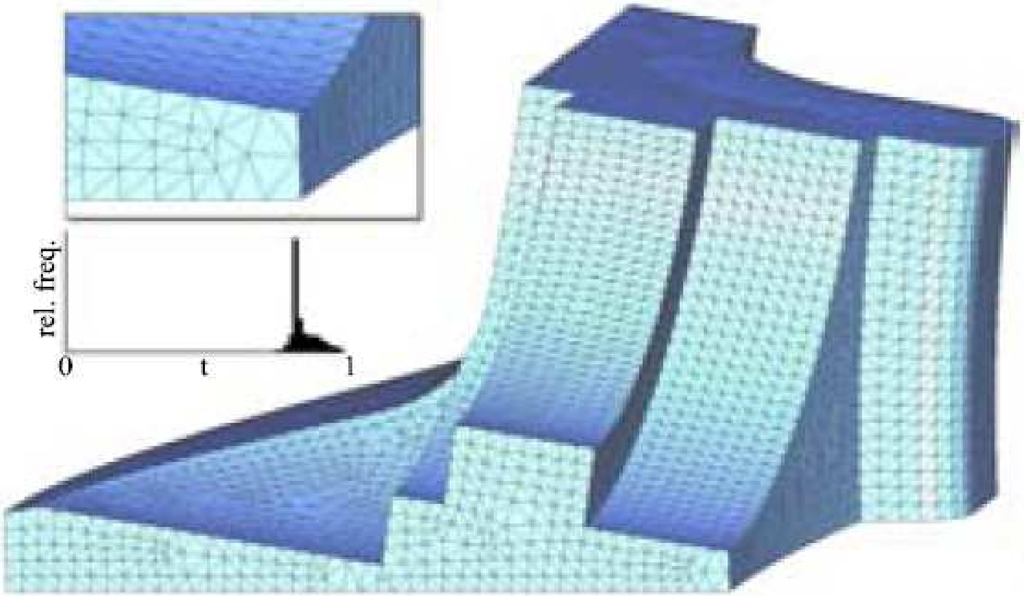


dist = $2.63 \cdot 10^{-3}$

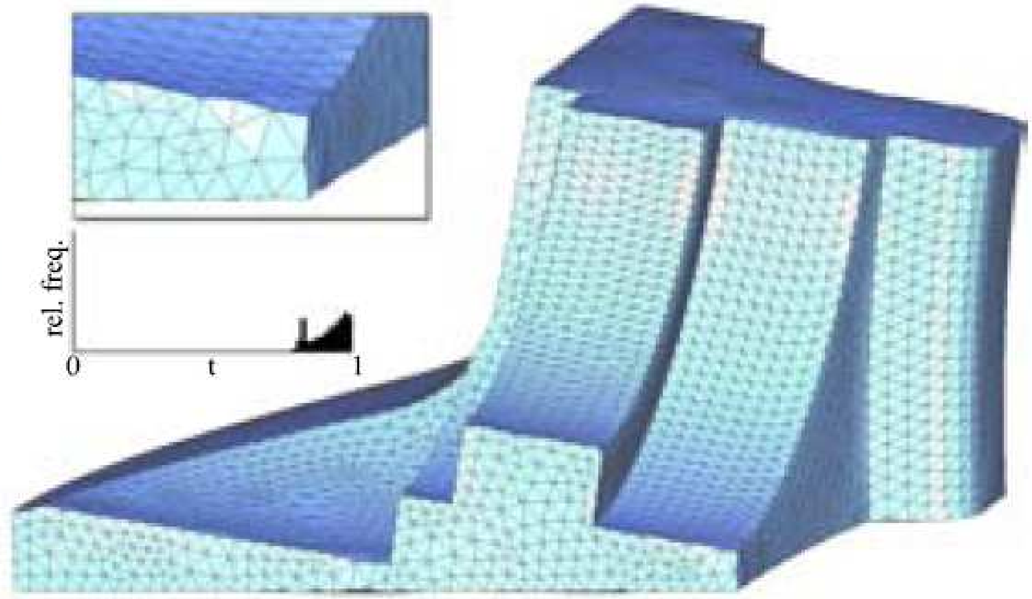
$t_{mean} = 0.868$
$t_{min} = 0.091$

- For meshes with distinct sharp features, we reduce the weight on the Laplacian constraint of high curvature vertices

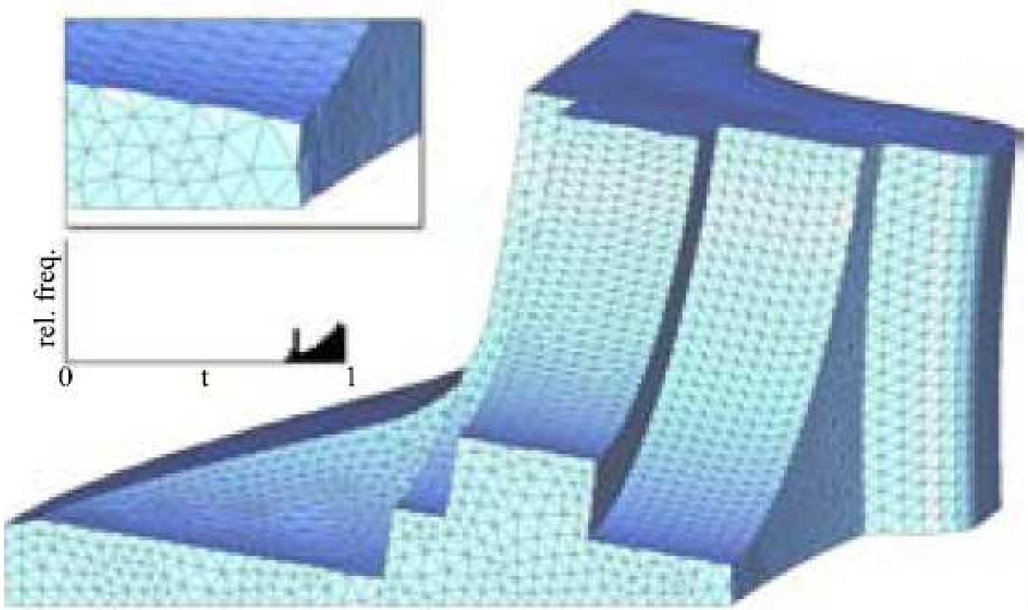
$$\mathbf{W}_L = s\mathbf{I} - \mathbf{W}_{linear}$$



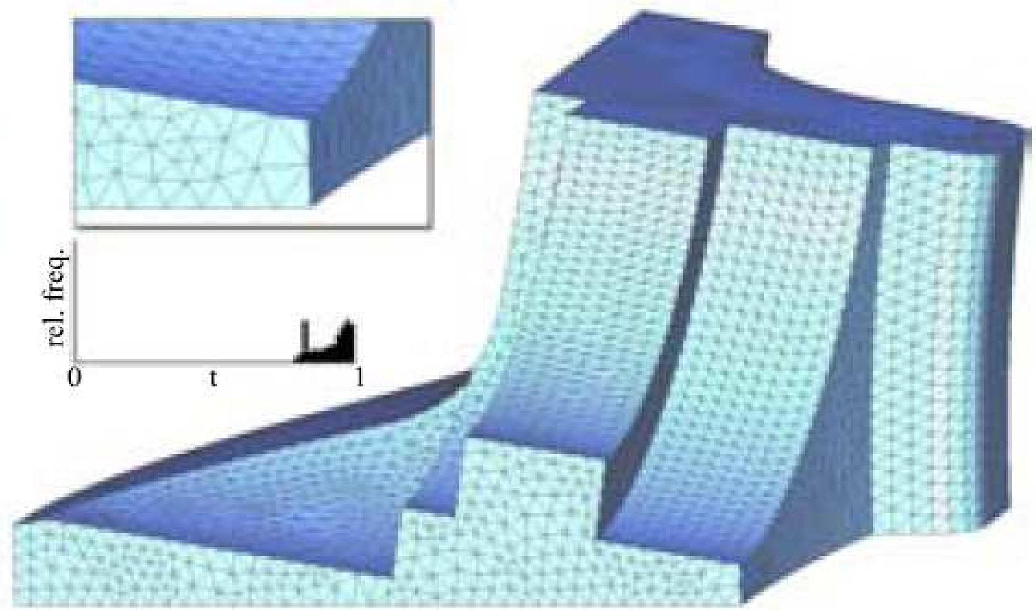
(a) original



(b) linear weights



(c) linear + tplane



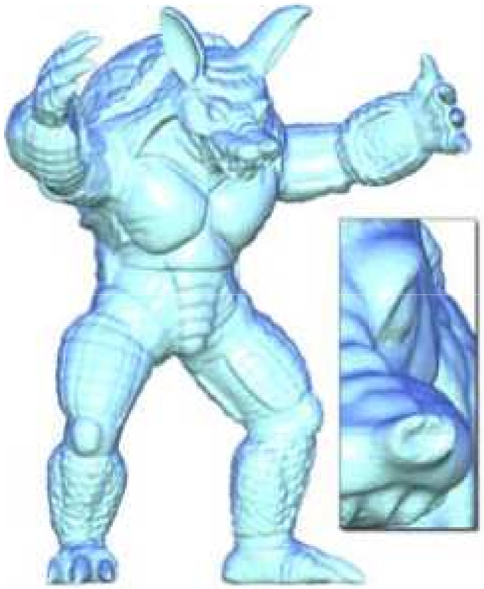
(d) linear + reduced Laplacian

- In some application, we may want to maximize $T_{min}$
- In this case, the positional weights $W_p$ can be modulated with the diagonal matrix $W_t$, where the entry $w_{t,i}$ for vertex $\mathbf{v}_i$ is set to the minimal triangle quality t of its adjacent triangles

- In some application, we may want to maximize $T_{min}$
- In this case, the positional weights $W_p$ can be modulated with the diagonal matrix $W_t$, where the entry $w_{t,i}$ for vertex $\mathbf{v}_i$ is set to the minimal triangle quality t of its adjacent triangles

# Mesh Smoothing

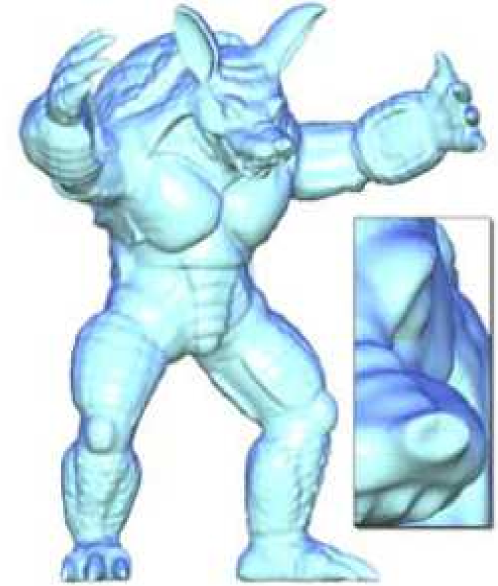
- Our frame work can be easily adjusted to perform global mes smoothing, optionally with feature preservation, simply by setting f=0, and adjusting the positional and Laplacian weights.
- Three parameters can be adjusted:
  - Positional Weights: $W_{const}$, $W_{linear}$, and $W_{cdf}$
  - Scale factor **s**
  - **Laplacian weights $W_L$**
    - **Any variants of the function mentioned above can be used.**



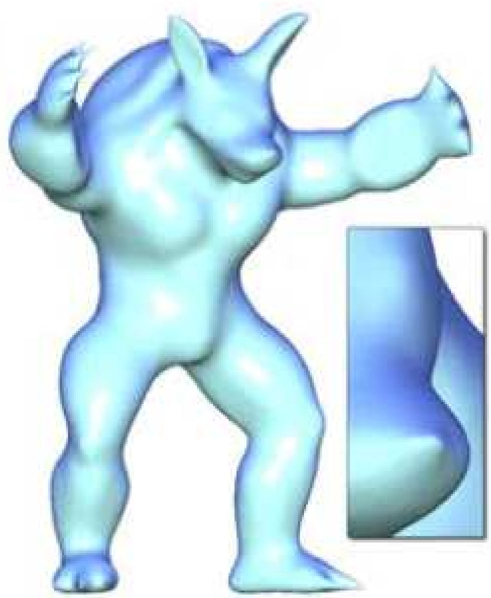
(a) original (173k)

(b) constant weights (s = 0.1)

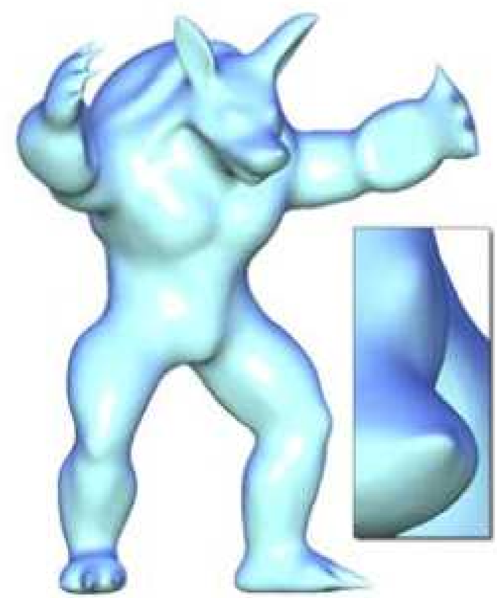
(c) cdf weights (s = 0.2)



(d) cdf + Laplacian weights (s = 0.2)



(e) constant weights (s = 0.01)



(f) cdf weights (s = 0.02)

(g) cdf + Laplacian weights (s = 0.02)

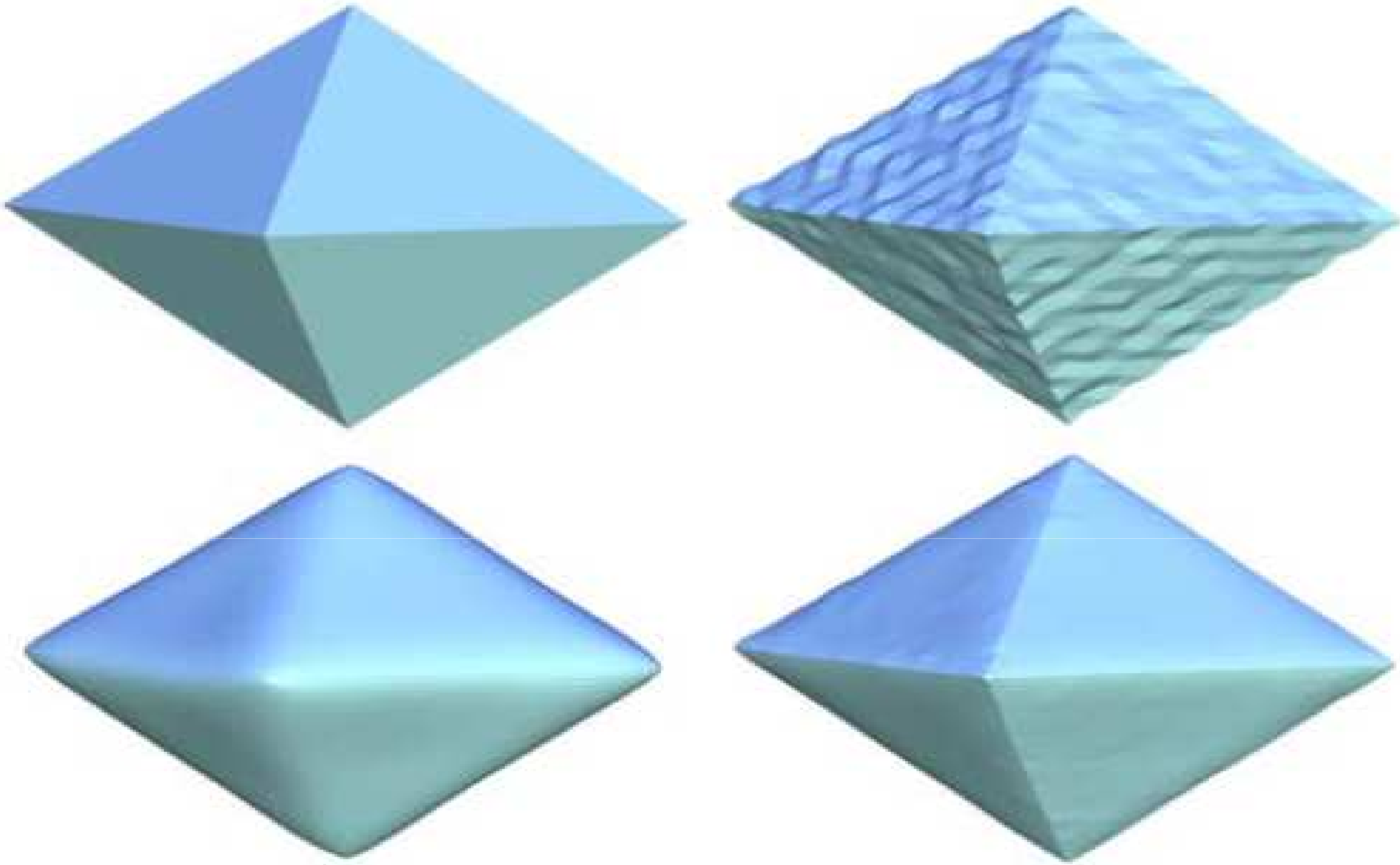
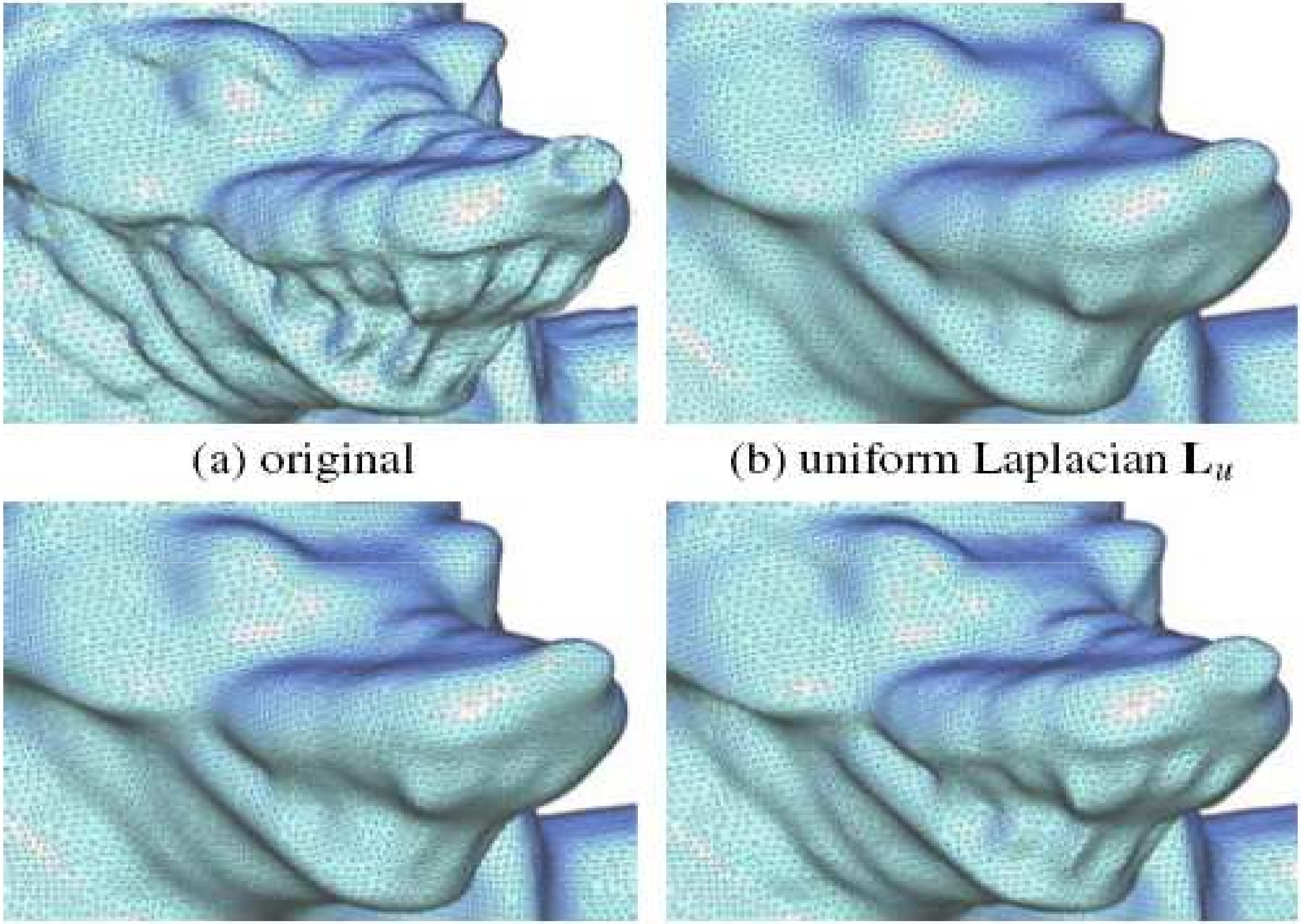


Figure 8: Smoothing a pyramid. Top row: original DOUBLEPYRAMID and the noisy version. Bottom row: Using $\mathbf{W}_p = \mathbf{W}_{linear}$ alone smoothes out sharp features, while additionally reducing the weights on Laplacian constraints of feature vertices recovers most of the original shape.



(a) original

(b) uniform Laplacian $\mathbf{L}_u$

(c) cotangent Laplacian $\mathbf{L}_c$

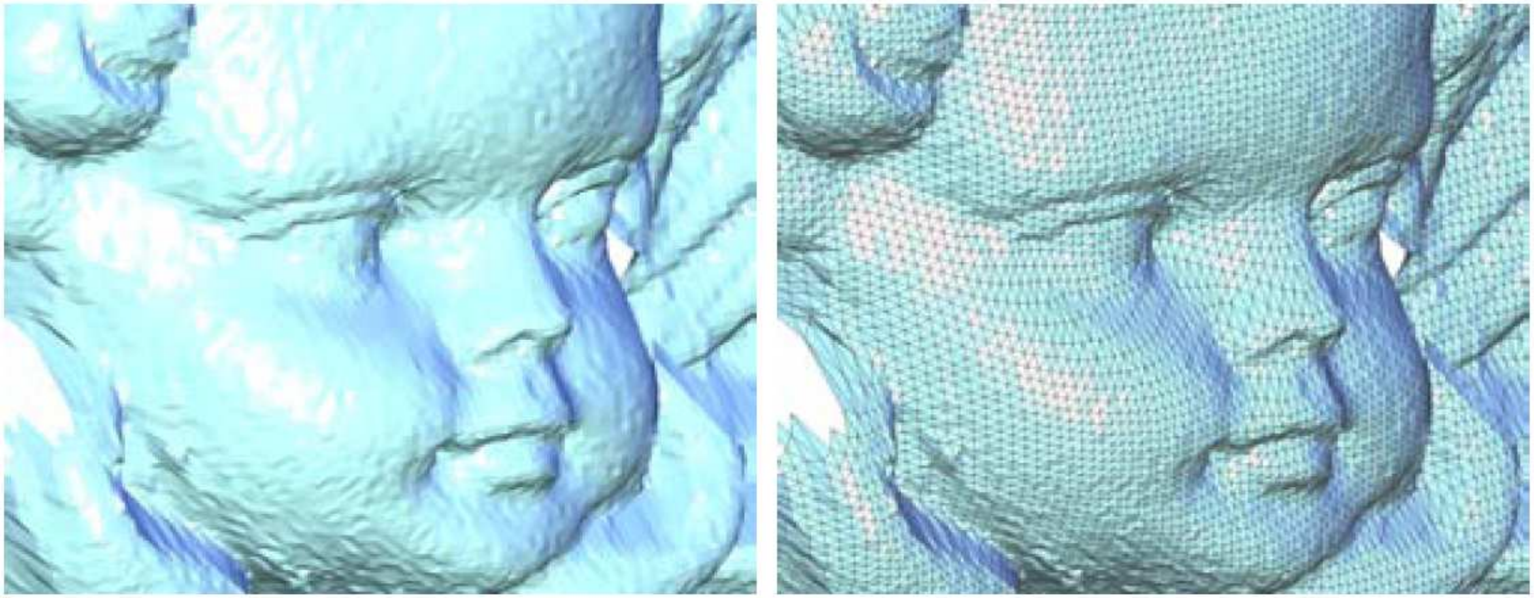
(d) weighted Laplacian $\mathbf{W}_L\mathbf{L}_c$

Figure 9: Comparison of umbrella (b) and cotangent (c) discretization of the $\mathbf{L}$ matrix. In (d), Laplacian constraints $\mathbf{L}_c\mathbf{V}'_d = 0$ are relaxed on feature vertices.

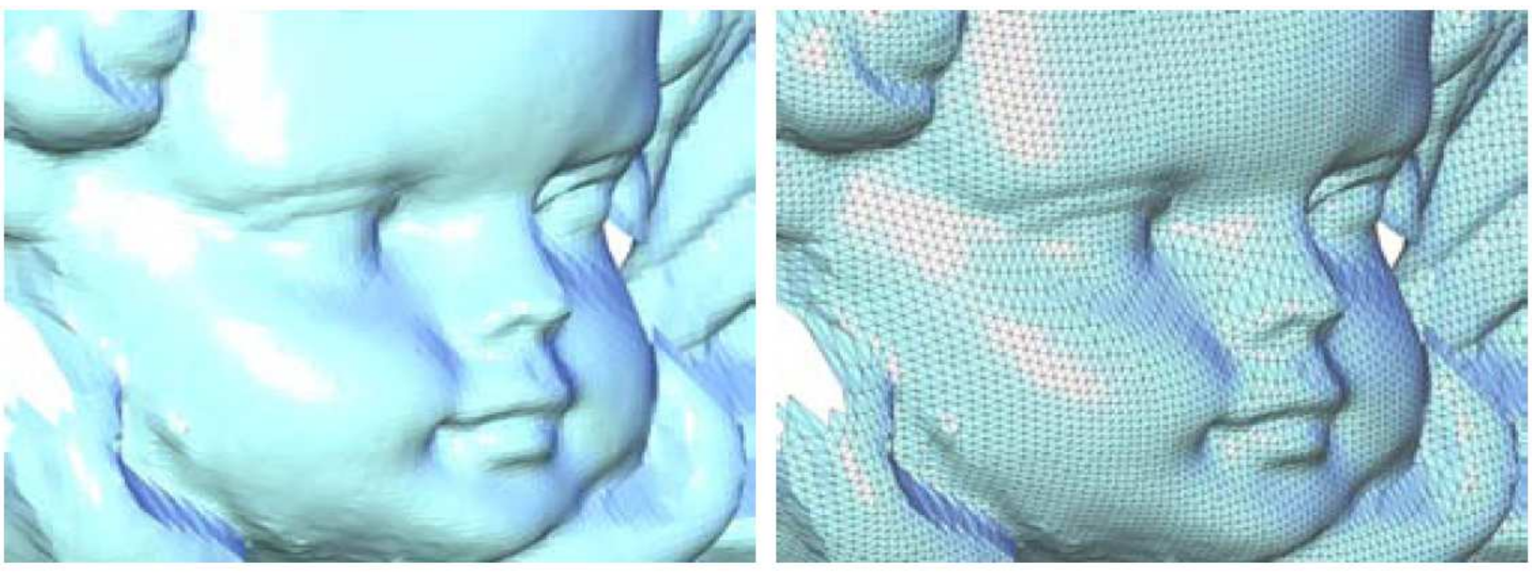
# Conclusion

- Pros
  - Efficient, non-iterative approach
  - Easily controllable triangle shape optimization and mesh smoothing
  - Leverages existing least squares frameworks
- Cons
  - Rely on some parameter tweaking to get what you want...
  - Euclidean distance is not Hausdorff distance, so error control is indirect...

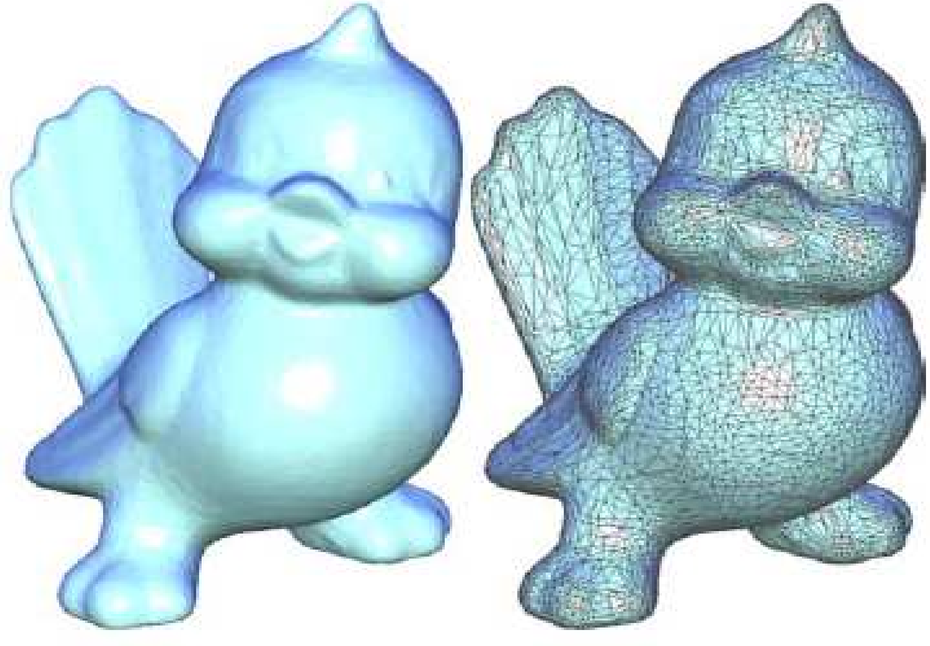
# More results....



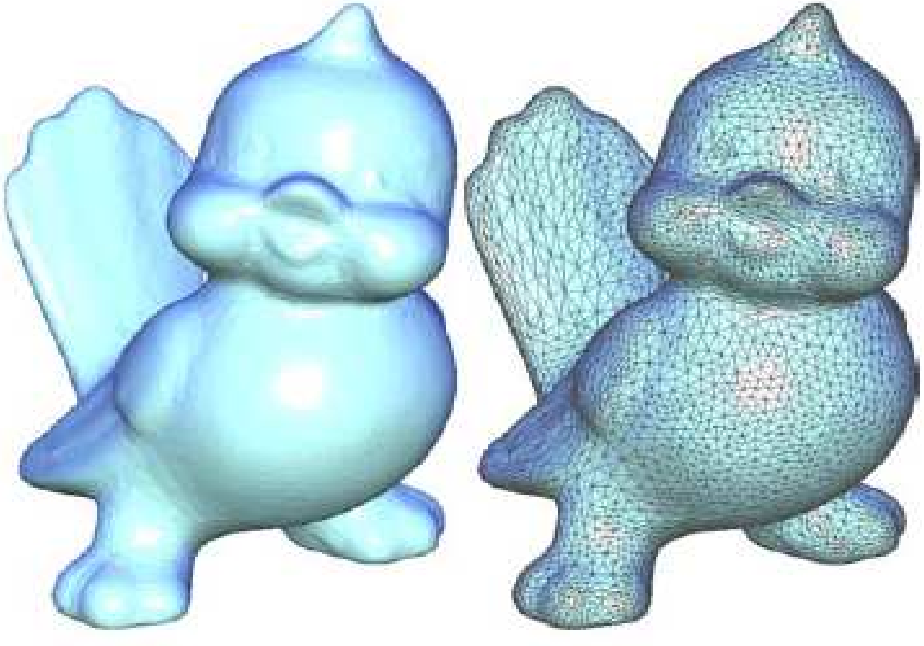
(a) original



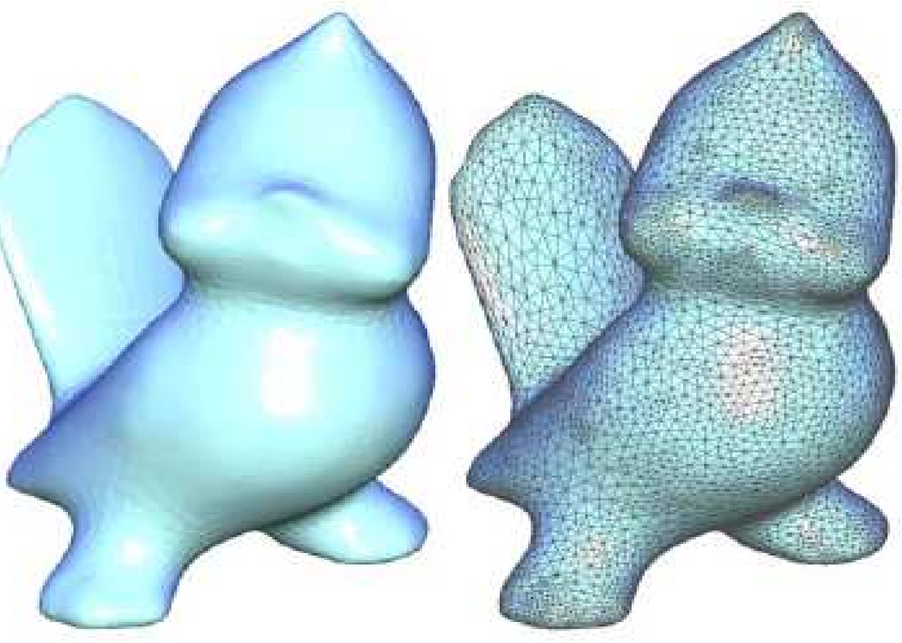
(b) smoothed



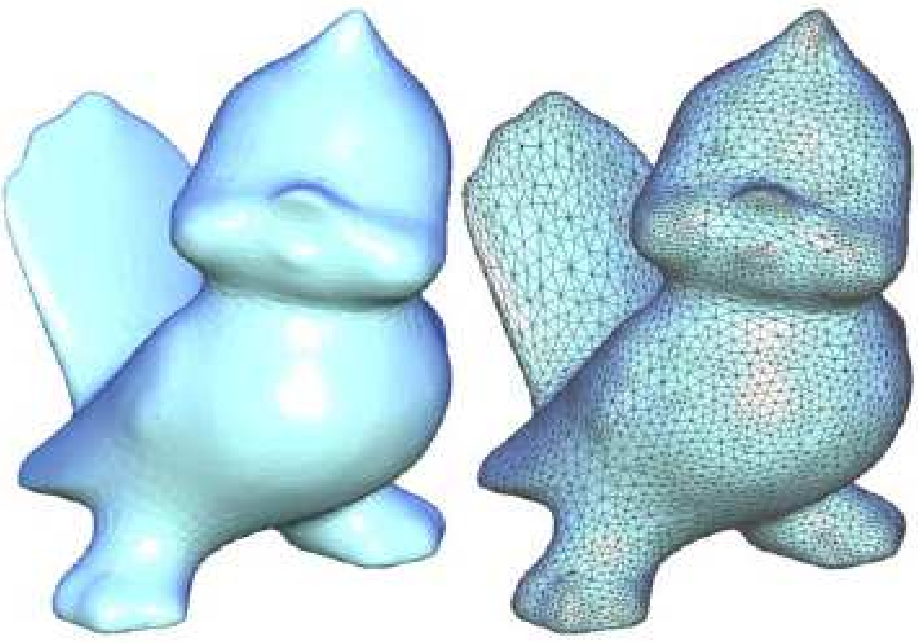
(a) original



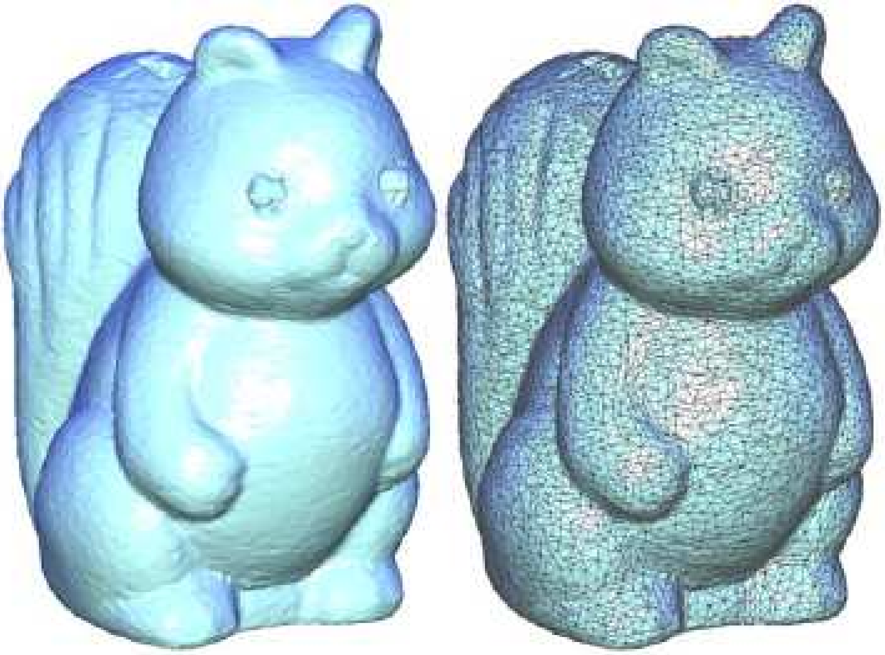
(b) triangle optimization with $\mathbf{W}_{cdf}$



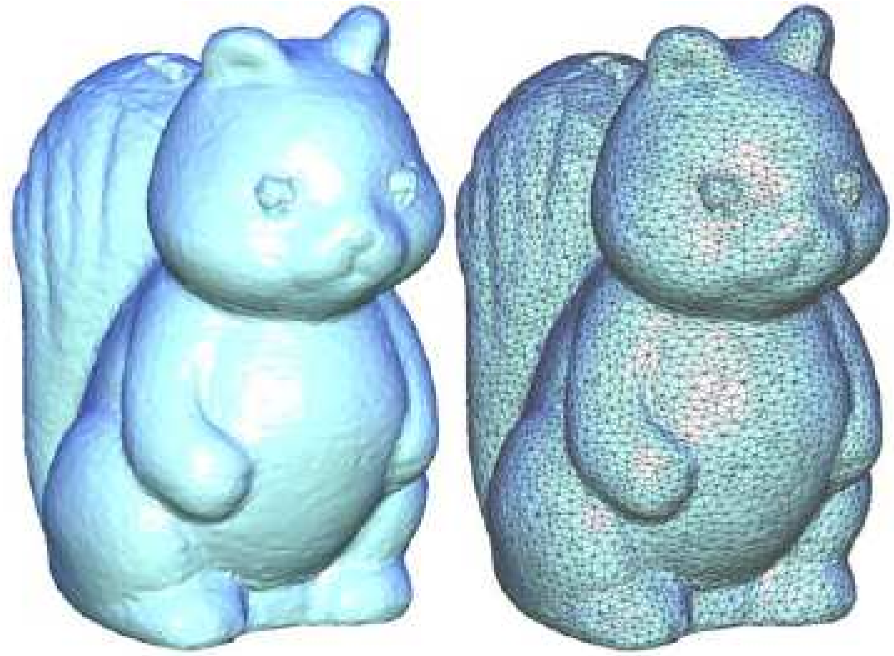
(c) smoothing with $\mathbf{W}_{cdf}$ and $\mathbf{L}_u$
(inner and outer fairness)



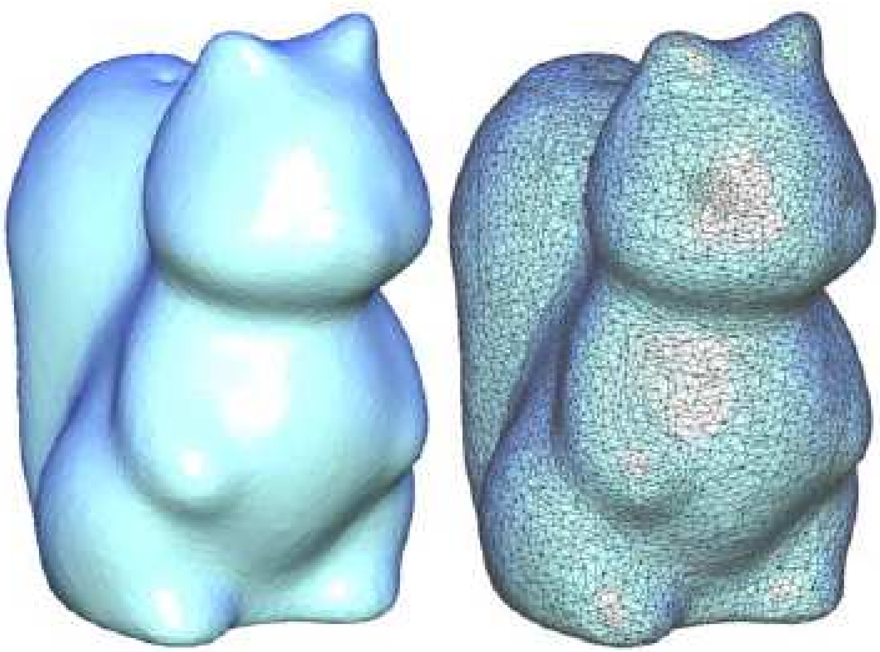
(d) feature preserving smoothing with $\mathbf{W}_{cdf}$ and $\mathbf{W}_L\mathbf{L}_u$
(inner and outer fairness)



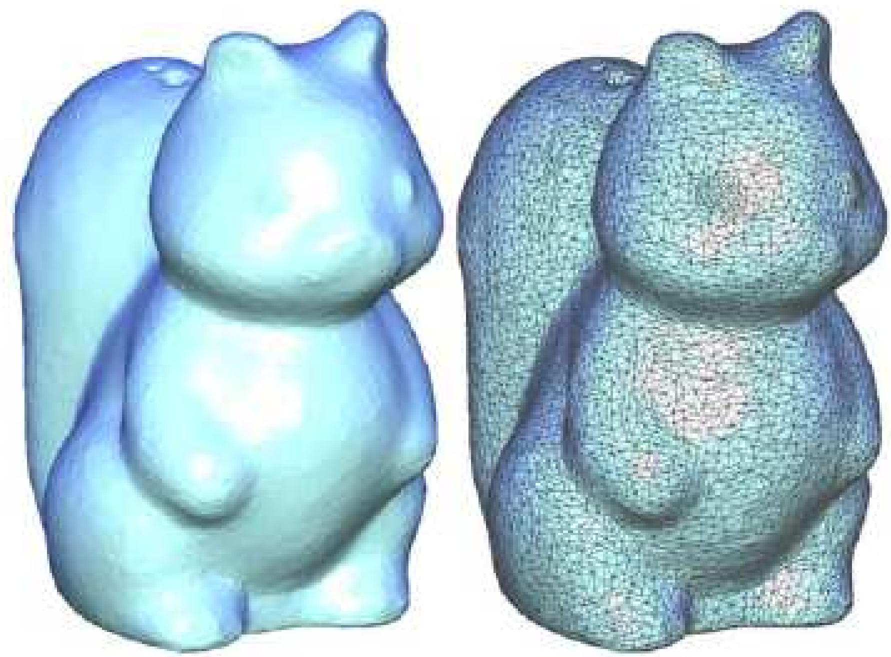
(a) original



(b) triangle optimization with $\mathbf{W}_{cdf}$



(c) smoothing with $\mathbf{W}_{cdf}$ and $\mathbf{L}_{c\overline{\kappa}}$
(outer fairness only, inner fairness untouched)



(d) feature preserving smoothing with $\mathbf{W}_{cdf}$ and $\mathbf{W}_{L}\mathbf{L}_{c}$
(outer fairness only, inner fairness untouched)

Thank you...