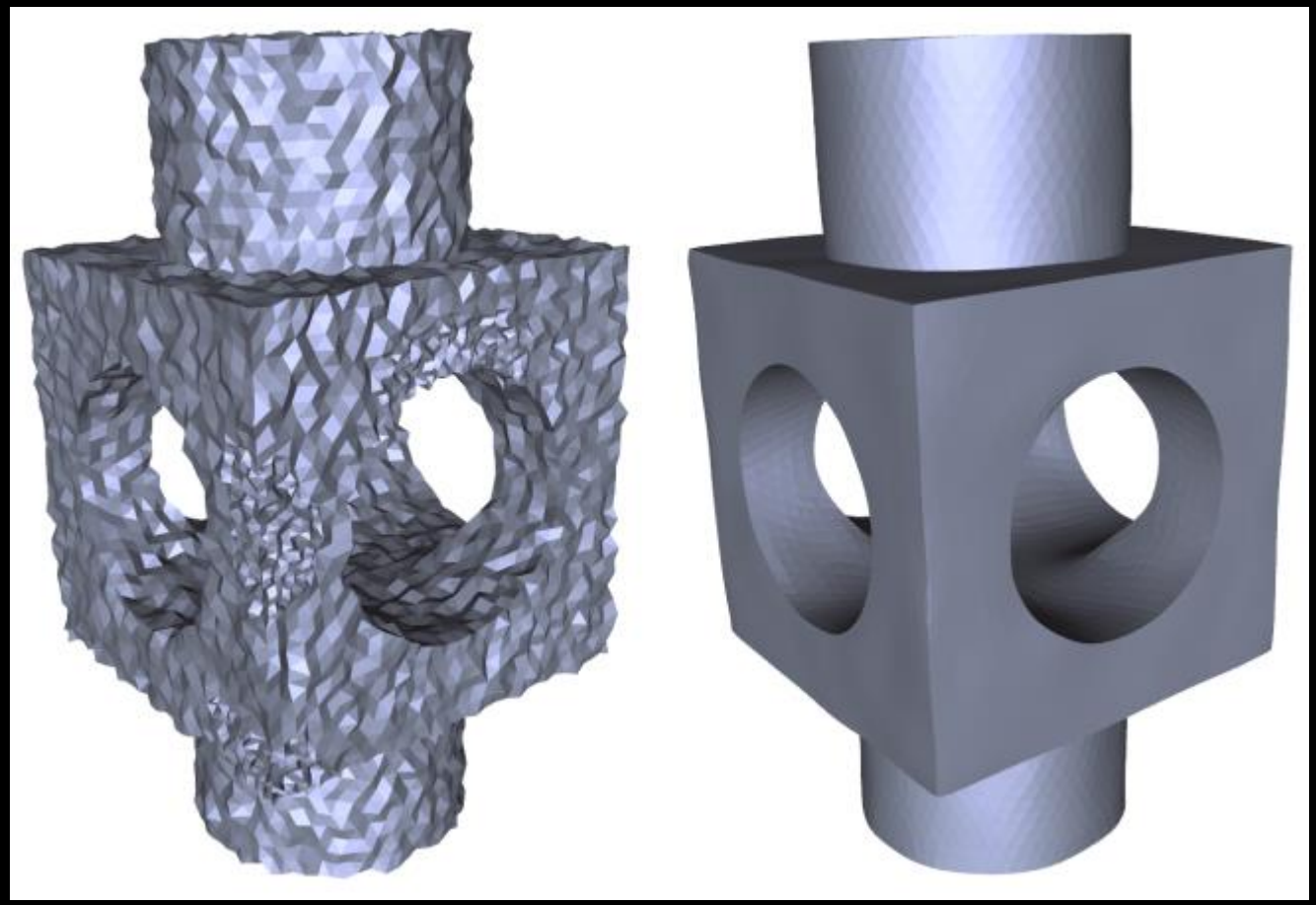
Mesh Smoothing

Xiao-Ming Fu

Denoising

- Removing the noise (the high frequencies) and keeping the overall shape (the low frequencies)
- Physical scanning process
- Feature VS Noise



Smoothing – From wiki

- In statistics and image processing, to smooth a data set is to create an approximating function that attempts to capture important patterns in the data, while leaving out noise or other fine-scale structures/rapid phenomena.

- In smoothing, the data points of a signal are modified so individual points (presumably because of noise) are reduced, and points that are lower than the adjacent points are increased leading to a smoother signal.

Outline

- Filter-based methods
- Optimization-based methods
- Data-driven methods

Outline

- Filter-based methods
- Optimization-based methods
- Data-driven methods

Laplacian smoothing

- Diffusion flow: a mathematically well-understood model for the time-dependent process of smoothing a given signal $f(\boldsymbol{x}, t)$.
  - Heat diffusion, Brownian motion
- Diffusion equation:

$$\frac{\partial f(\boldsymbol{x}, t)}{\partial t} = \lambda \Delta f(\boldsymbol{x}, t)$$

1. A second-order linear partial differential equation;
2. Smooth an arbitrary function f on a manifold surface by using Laplace-Beltrami Operator.
3. Discretize the equation both in space and time.

Spatial discretization

- Sample values at the mesh vertices $\boldsymbol{f}(t) = (f(v_1, t), \dots, f(v_n, t))^T$
- Discrete Laplace-Beltrami using either the uniform or cotangent formula.
- The evolution of the function value of each vertex:

$$\frac{\partial f(v_i, t)}{\partial t} = \lambda \Delta f(\boldsymbol{x}_i, t)$$

Matrix form:

$$\frac{\partial \boldsymbol{f}(t)}{\partial t} = \lambda \cdot L\boldsymbol{f}(t)$$

Temporal discretization

- Uniform sampling: $(t, t+h, t+2h, \dots)$
- Explicit Euler integration:

$$\boldsymbol{f}(t+h) = \boldsymbol{f}(t) + h\frac{\partial \boldsymbol{f}(t)}{\partial t} = \boldsymbol{f}(t) + h\lambda \cdot L\boldsymbol{f}(t)$$

1. Numerically stability: a sufficiently small time step h.

- Implicit Euler integration:

$$\boldsymbol{f}(t+h) = \boldsymbol{f}(t) + h\lambda \cdot L\boldsymbol{f}(t+h)$$
$$\Leftrightarrow (\boldsymbol{I} - h\lambda \cdot L)\boldsymbol{f}(t+h) = \boldsymbol{f}(t)$$

Laplacian smoothing

- Arbitrary function $\Longrightarrow$ vertex positions
  - $\boldsymbol{f} \Longrightarrow (\boldsymbol{x_1}, \ldots, \boldsymbol{x_n})^T$

- Laplacian smoothing:

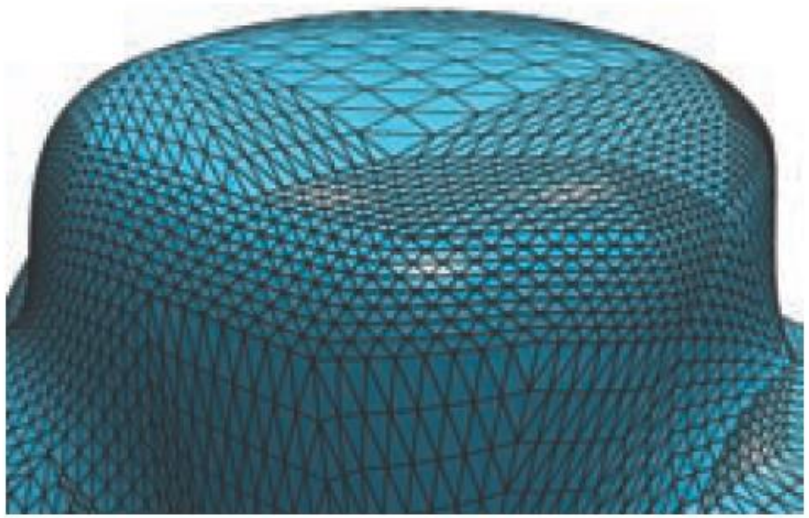
$$\boldsymbol{x}_i \longleftarrow \boldsymbol{x}_i + h\lambda \cdot \Delta \boldsymbol{x}_i$$

1. $\Delta \boldsymbol{x} = -2H\boldsymbol{n} \longrightarrow$ vertices move along the normal direction by an amount determined by the mean curvature H.
2. mean curvature flow.

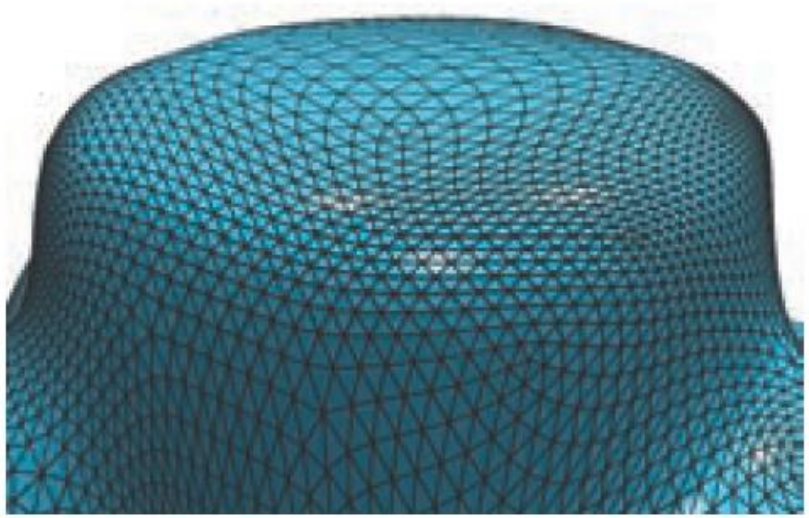
Figure 4.5. Curvature flow smoothing of the bunny mesh (left), showing the result after ten iterations (center) and 100 iterations (right). The color coding shows the mean curvature. (Model courtesy of the Stanford Computer Graphics Laboratory.)

Different Laplace-Beltrami operators

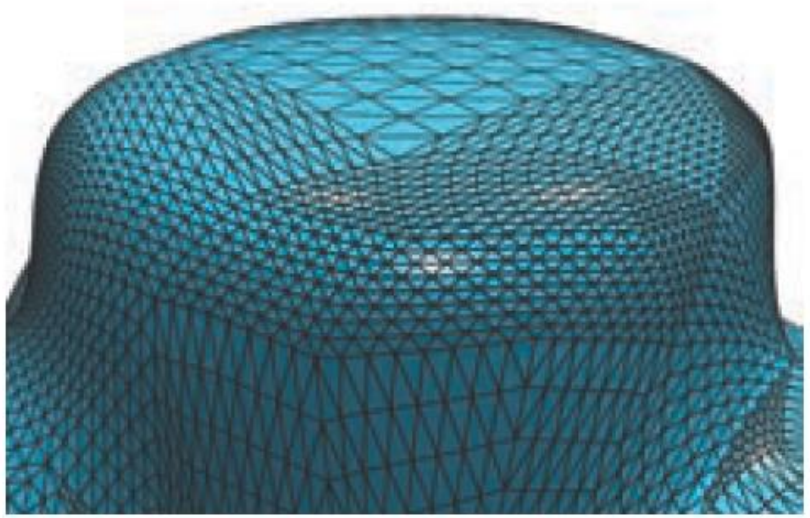
- Cotangent Laplacian.
  - the movement in the normal direction is true.
- Uniform Laplacian
  - move each vertex to the barycenter of its one-ring neighbors.
  - smooths the mesh geometry and a tangential relaxation of the triangulation.



Input



Uniform



Cotangent

Fairing

- Goal: compute shapes that are as smooth as possible
- Steady-states of the flow:
  - $L\boldsymbol{x} = 0$
  - $L^k\boldsymbol{x} = 0$
  - …

Figure 4.8. The blue region is determined by minimizing a fairness functional: membrane surface ($\Delta\mathbf{x} = 0$, left), thin-plate surface ($\Delta^2\mathbf{x} = 0$, center), and minimum variation surface ($\Delta^3\mathbf{x} = 0$, right). The order k of the Euler-Lagrange equation $\Delta^k\mathbf{x} = 0$ determines the maximum smoothness C^{k-1} at the boundary. (Image taken from [Botsch and Kobbelt 04a]. ©2004 ACM, Inc. Included here by permission.)

Gaussian Image Denoising

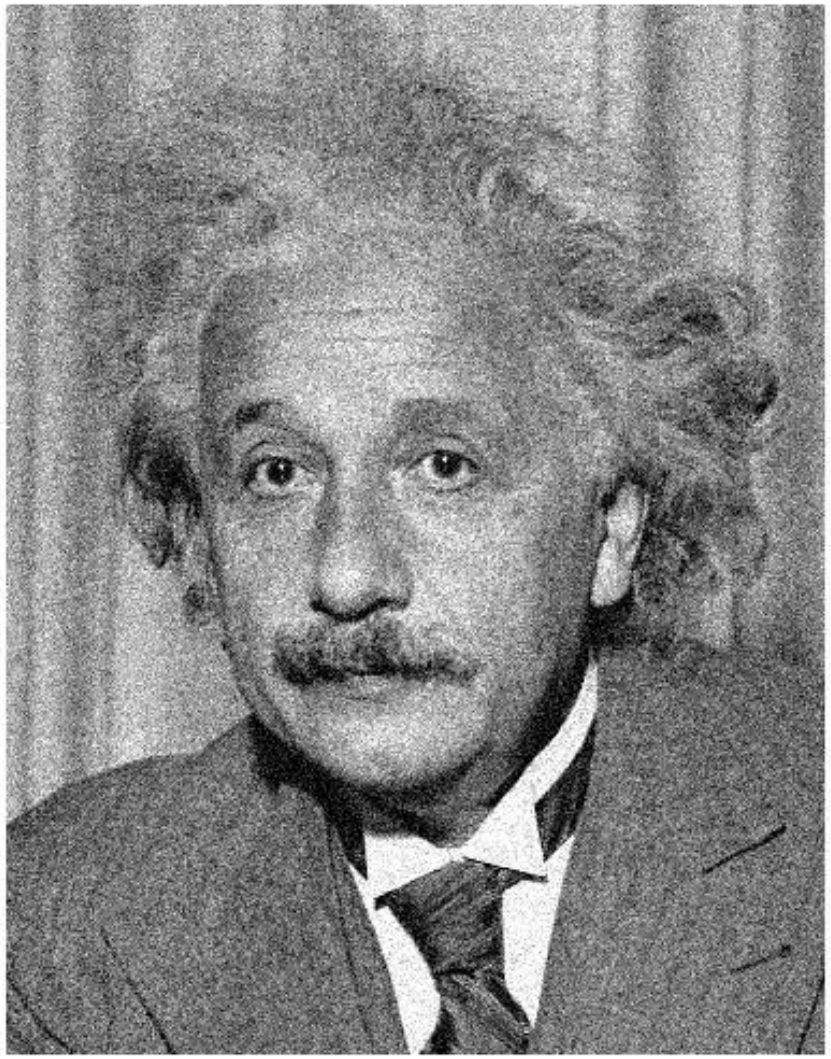
- The Gaussian filter for an image pixel $I(\boldsymbol{p})$, at coordinate $\boldsymbol{p} = (x, y)$, is defined as:

$$I(\boldsymbol{p}) \leftarrow \frac{1}{K_{\boldsymbol{p}}} \sum_{\boldsymbol{q} \in \Omega(\boldsymbol{p})} W_s(\|\boldsymbol{p} - \boldsymbol{q}\|) I(\boldsymbol{q})$$

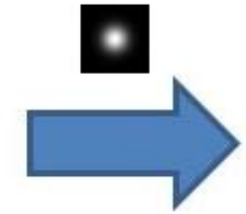
1. $\Omega(\boldsymbol{p})$: neighborhood of $\boldsymbol{p}$.

2. W_s: position similarity between $\boldsymbol{p}$ and $\boldsymbol{q}$, Gaussian function with standard deviations σ_s

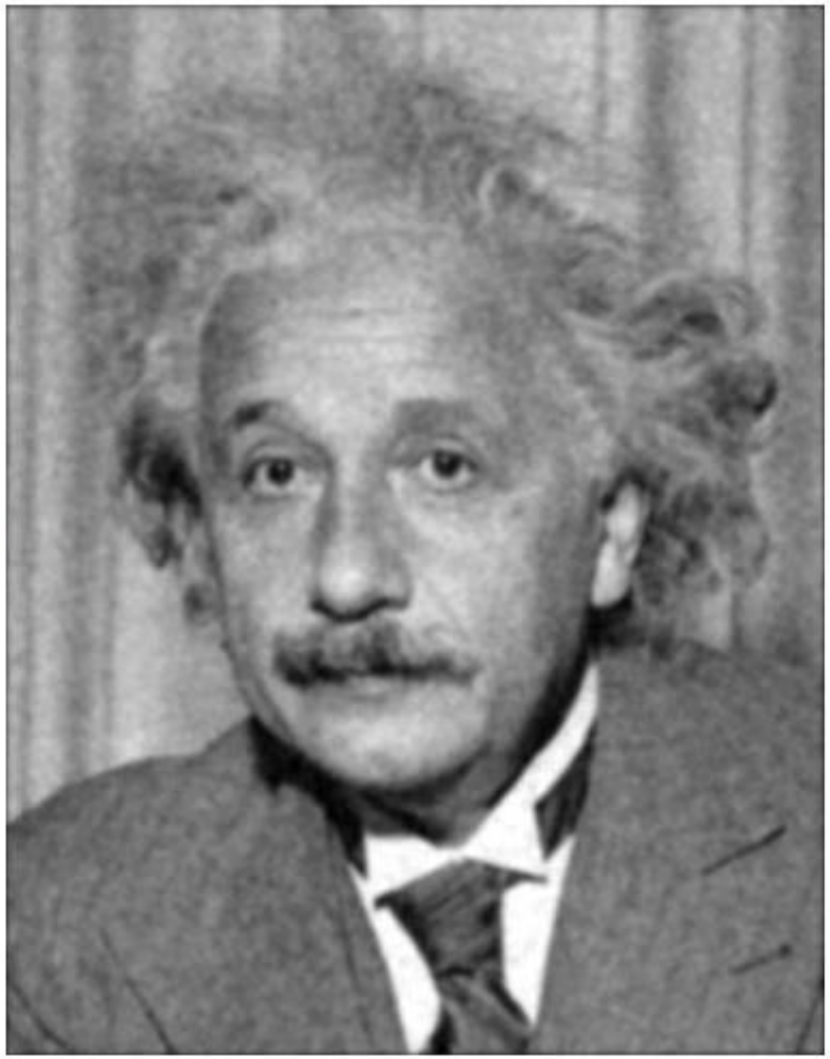
3. $K_{\boldsymbol{p}}$ is the normalization term, the summation of weights.



Additive Gaussian Noise



Gaussian
Filter



Edge is not preserved!!!

Bilateral Image Denoising

- The bilateral filter for an image pixel $I(\boldsymbol{p})$, at coordinate $\boldsymbol{p} = (x, y)$, is defined as:

$$I(\boldsymbol{p}) \leftarrow \frac{1}{K_{\boldsymbol{p}}} \sum_{\boldsymbol{q} \in \Omega(\boldsymbol{p})} W_s(\|\boldsymbol{p} - \boldsymbol{q}\|) W_r(\|I(\boldsymbol{p}) - I(\boldsymbol{q})\|) I(\boldsymbol{q})$$

1. $\Omega(\boldsymbol{p})$: neighborhood of $\boldsymbol{p}$.
2. W_s and W_r: monotonically decreasing weighting functions and often be Gaussian functions, with standard deviations σ_s and σ_r.
3. W_s: position similarity between $\boldsymbol{p}$ and $\boldsymbol{q}$.
4. W_r: color similarity between $\boldsymbol{p}$ and $\boldsymbol{q}$.
5. $K_{\boldsymbol{p}}$ is the normalization term, the summation of weights.

Bilateral Image Denoising

- non-linear
- edge-preserving
- noise-reducing smoothing

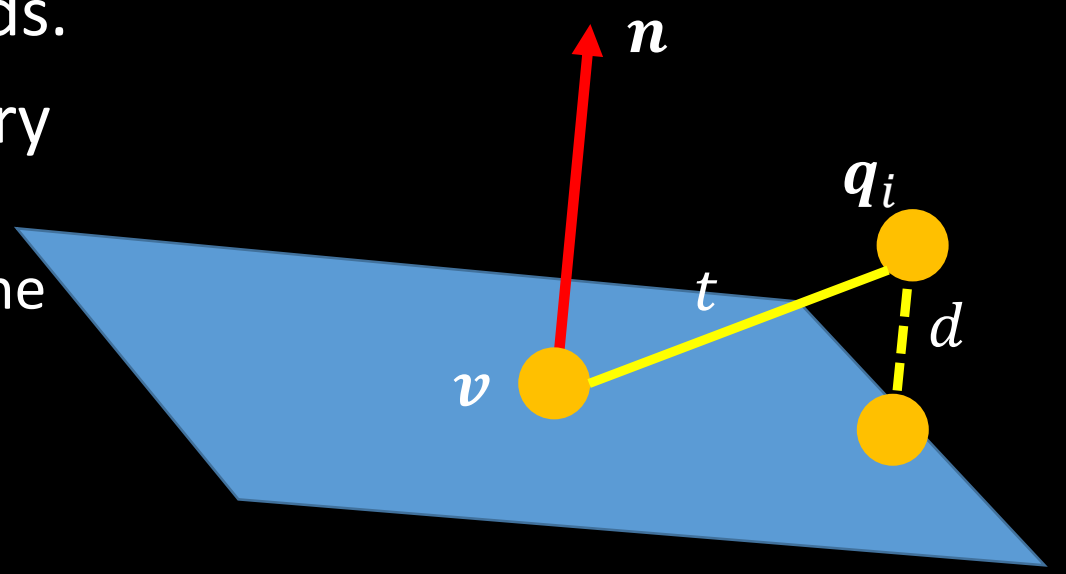
- **Extended to mesh case!!!**

Bilateral Mesh Denoising
Paper: Bilateral Mesh Denoising

- Vertex positions cannot simply be considered as the signal to be processed.
- Filtering a mesh using local neighborhoods.
- Main idea: local parameter space for every vertex using the tangent plane.
  - The heights of vertices over the tangent plane $\Leftrightarrow$ gray-level values of an image.
- Update $\boldsymbol{v}$:

$$\boldsymbol{v}^{new} = \boldsymbol{v} + d \cdot \boldsymbol{n}$$

Once $\boldsymbol{n}$ is given, we need to compute the new d to update $\boldsymbol{v}$.



Pseudo-code

Denoise_Point(Vertex $\boldsymbol{v}$, Normal $\boldsymbol{n}$)

$\{q_i\} = \Omega(\boldsymbol{v})$, N: number of neighbor vertices, d_{sum}, $K_{\boldsymbol{v}} = 0$

for $i \coloneqq 1\ to\ N$

$t = \|\boldsymbol{v} - \boldsymbol{q}_i\|$

$d = \langle \boldsymbol{n}, \boldsymbol{v} - \boldsymbol{q}_i \rangle$

$w_S = \exp(-t^2/(2\sigma_s))$

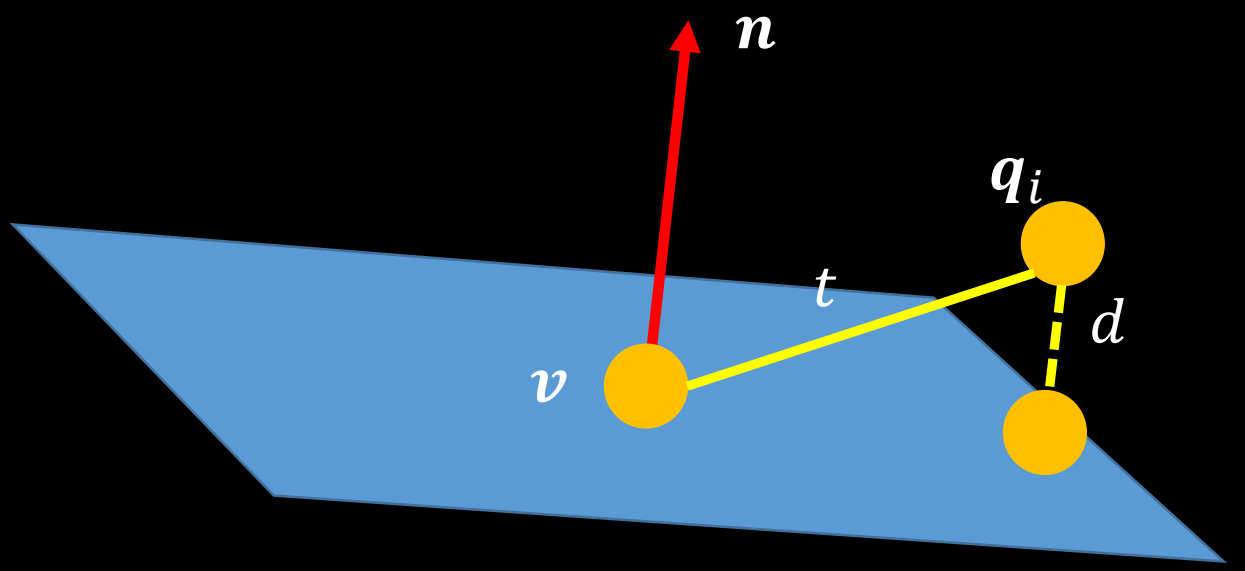
$w_r = \exp(-d^2/(2\sigma_r))$

$d_{\text{sum}} \mathrel{+}= w_S \cdot w_r \cdot d$

$K_{\boldsymbol{v}} \mathrel{+}= w_S \cdot w_r$

end

return $\boldsymbol{v}^{new} = \boldsymbol{v} + \boldsymbol{n} \cdot d_{\text{sum}}/K_{\boldsymbol{v}}$



Detail

- Normal: weighted average of the normals
  - 1-ring neighborhood of the vertex.
  - k-ring neighborhood for extremely noisy data.
  - weight: the area of the triangles.
- Mesh shrinkage: volume preservation technique.
  - Scale the updated mesh to preserve the volume.
  - How to compute the volume?
- Boundary.
- Parameters: σ_s, σ_r, number of iterations.

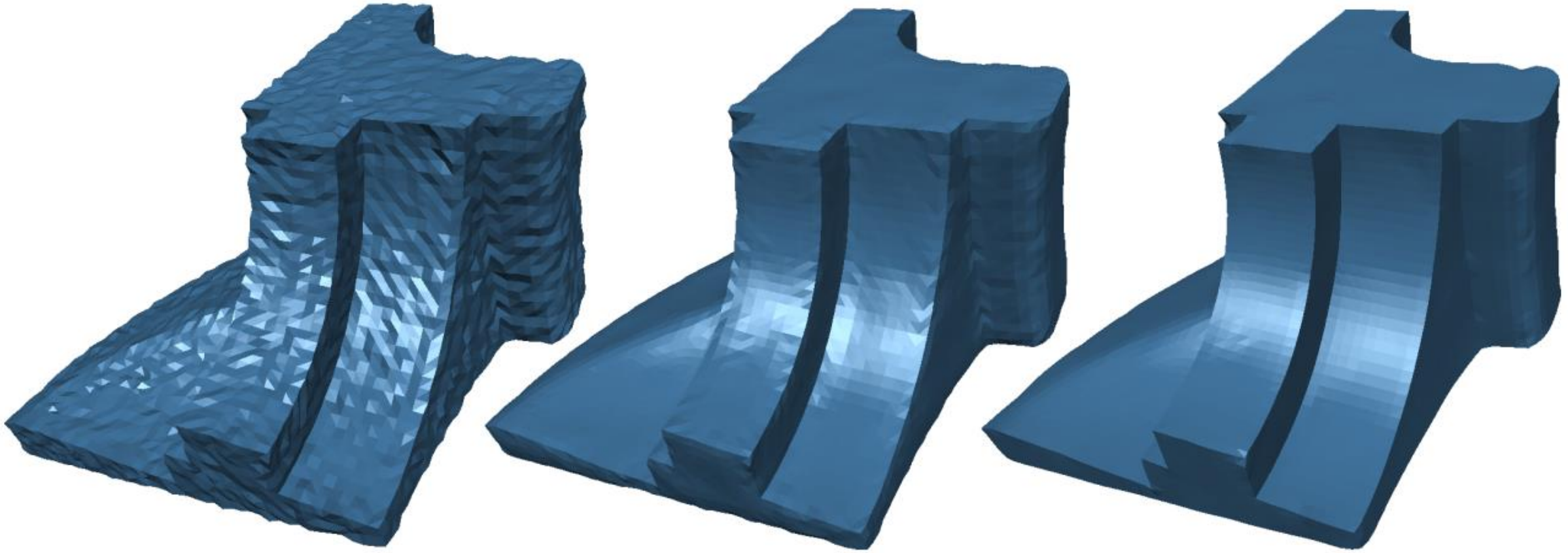


Figure 5: Results of denoising the Fandisk model. On the left is the input noisy model, in the middle is the results of [Jones et al. 2003], and on the right is our result.

Bilateral Normal Filtering

Paper: Bilateral Normal Filtering for Mesh Denoising

- The normals on facets are well-defined.
- Considers normals as a surface signal defined over the original mesh.
- A novel bilateral normal filter that depends on both spatial distance and signal distance.
- Recover vertex positions in global and non-iterative manner.

Bilateral Normal Filtering

$$\boldsymbol{n}(f_i) \leftarrow \frac{1}{K_{\boldsymbol{p}}} \sum_{f_j \in \Omega(f_i)} A_j W_s\left(\left\|\boldsymbol{c}_i - \boldsymbol{c}_j\right\|\right) W_r\left(\left\|\boldsymbol{n}(f_i) - \boldsymbol{n}(f_j)\right\|\right) \cdot \boldsymbol{n}(f_j)$$

1. $\boldsymbol{n}(f_i)$: the normal of facet f_i.
2. $\boldsymbol{c}_i$: the center of facet f_i.
3. $\Omega(f_i)$: the neighbor facets of f_i.
4. $W_s\left(\left\|\boldsymbol{c}_i - \boldsymbol{c}_j\right\|\right)$: spatial distance.
5. $W_r\left(\left\|\boldsymbol{n}(f_i) - \boldsymbol{n}(f_j)\right\|\right)$: normal difference.
6. A_j: the area of facet f_j.

Mesh Denoising

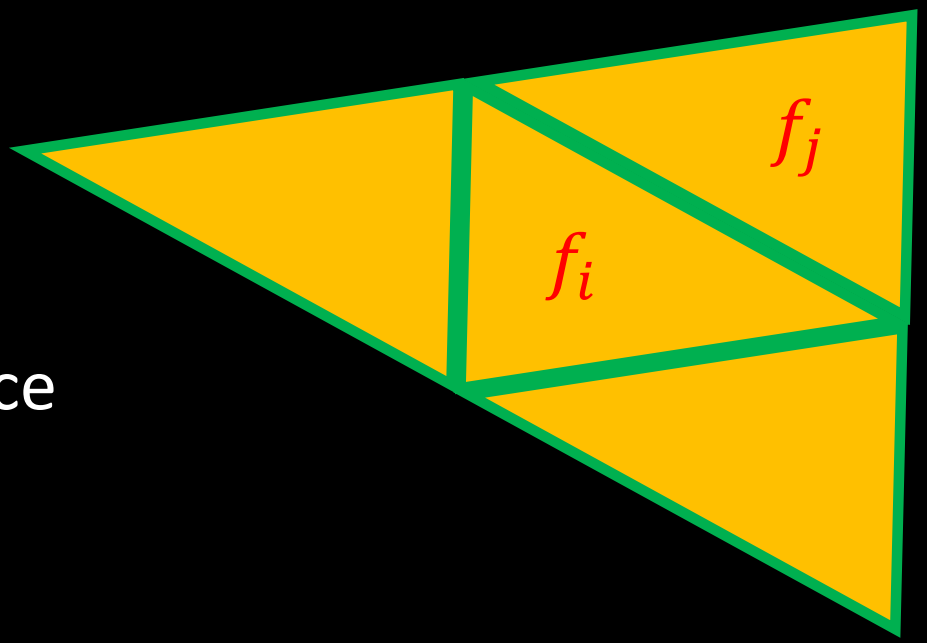
- Given the normal on each facet, determine the vertex positions to match the normal as much as possible.

- Local and Iterative Scheme
  - update the normal field.
  - update the vertex positions.

- Global and Non-Iterative Scheme
  - Energy minimization.

Local and Iterative Scheme

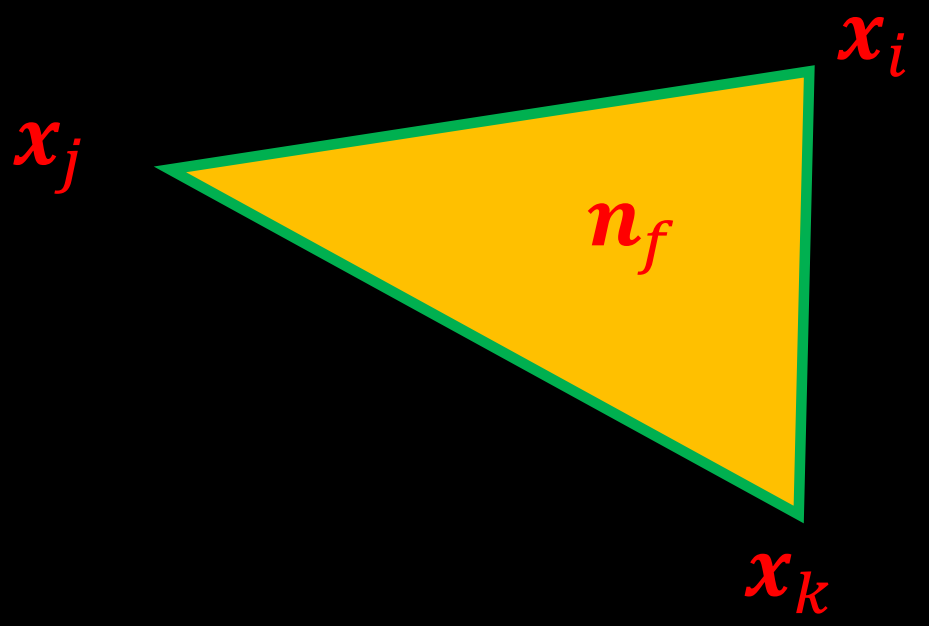
- Normal Updating:
  - $\boldsymbol{n}_i^{t+1} \leftarrow \frac{1}{K_{\boldsymbol{p}}} \sum_{f_j \in \Omega(f_i)} A_j W_s W_r \cdot \boldsymbol{n}_j^t$
  - Normalize the new normal after each iteration.
  - Multiple iterations: increase the influence from a 1-ring neighborhood to a wider region, leading to a smoother mesh.
  - Iteration number: user controls.



Local and Iterative Scheme

Vertex Updating:

$$\begin{cases} \boldsymbol{n}_f^T \cdot (\boldsymbol{x}_j - \boldsymbol{x}_i) = 0 \\ \boldsymbol{n}_f^T \cdot (\boldsymbol{x}_k - \boldsymbol{x}_j) = 0 \\ \boldsymbol{n}_f^T \cdot (\boldsymbol{x}_i - \boldsymbol{x}_k) = 0 \end{cases}$$



Energy:

$$E = \sum_{f_k} \sum_{i,j \in f_k} \left(\boldsymbol{n}_k^T \cdot (\boldsymbol{x}_j - \boldsymbol{x}_i) \right)^2$$

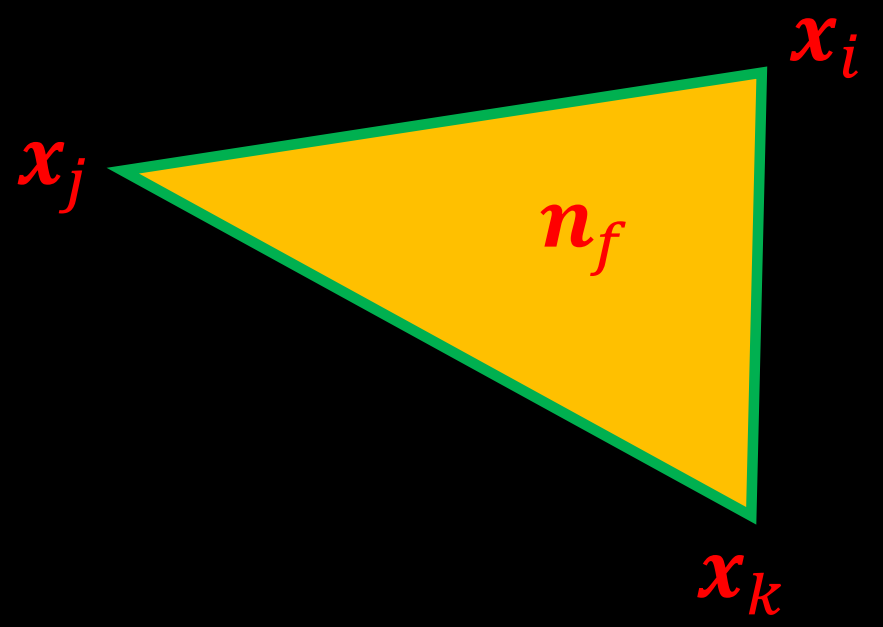
Linear system.

Local and Iterative Scheme

Each time, fix other vertex, update one vertex (Gauss–Seidel iteration). Why???

$$\boldsymbol{x}_i^{new} = \boldsymbol{x}_i + \frac{1}{N_i} \sum_{f_j \in \Omega(i)} \boldsymbol{n}_j \cdot \left(\boldsymbol{n}_j^T \cdot (\boldsymbol{c}_j - \boldsymbol{x}_i) \right)$$

1. No need to determine a suitable step size.
2. Not computationally expensive. No need to solve a linear system.
3. Iteration number: user control.



Global and Non-Iterative Scheme

- Energy minimization:

$$E(n) = (1-\lambda)E_s + \lambda E_a$$

$$E_s = \sum_i A_i \| (L\boldsymbol{n}^{new})_i \|_2^2$$

$$E_a = \sum_i A_i \| \boldsymbol{n}_i^{new} - n_i \|_2^2$$

1. L: Uniform weighting or cotangent weighting.
2. λ: a parameter to balance the E_s and E_a.

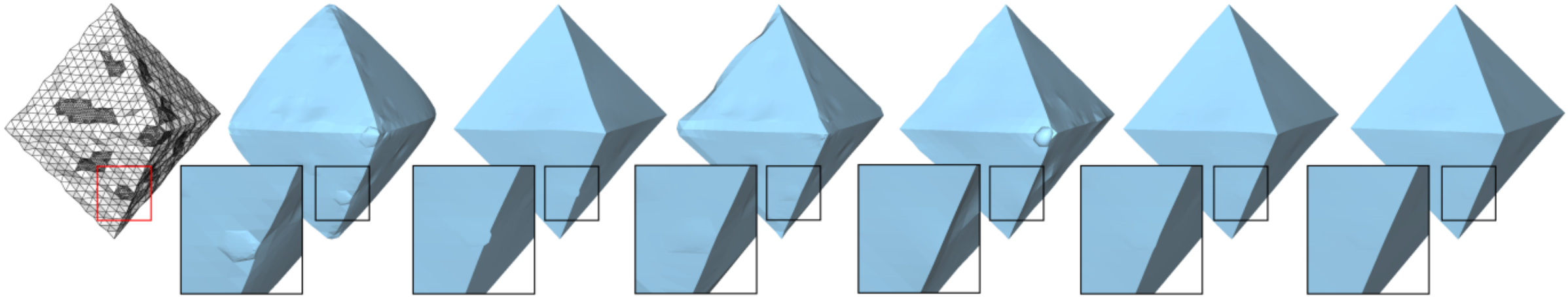


Fig. 1: Our mesh denoising schemes based on bilateral normal filtering produce better results than the state-of-the-art methods at challenging regions with sharp features or irregular surface sampling. From left to right: an input CAD-like model with random subdivision, denoising results with bilateral mesh filtering (vertex-based) [1], unilateral normal filtering [2], probabilistic smoothing [3], prescribed mean curvature flow [4], our local, iterative scheme, and our global, non-iterative scheme. All the meshes in the paper are flat-shaded to show faceting.

Manifold Harmonics

Paper: Spectral Geometry Processing with Manifold Harmonics

- 1D Fourier Transform:

$$F(\omega) = \int_{-\infty}^{+\infty} f(x) e^{-2\pi i \omega x} dx$$

$$f(x) = \int_{-\infty}^{+\infty} F(\omega) e^{2\pi i \omega x} d\omega$$

spatial domain $f(x)$ $\Leftrightarrow$ frequency domain $F(\omega)$

1D Fourier Transform

- An intuitive geometric interpretation of $f(x)$:
  - an element of a certain vector space
  - $\langle f, g \rangle = \int_{-\infty}^{+\infty} f(x)\overline{g(x)}dx$
  - $g(x) = e_\omega(x) := e^{2\pi i\omega x} = \cos(2\pi\omega x) + i \cdot sin(2\pi\omega x)$
- $e_\omega(x)$: frequency-related orthogonal basis
  - $f(x) = \int_{-\infty}^{+\infty} \langle f(x), e_\omega(x) \rangle \, e_\omega(x) d\omega$
  - It describes how much of the basis function $e_\omega(x)$ is contained in $f(x)$.
- Low-pass filter:
  - cutting off all frequencies above a user-defined threshold ω_{max}
  - $\tilde{f}(x) = \int_{-\omega_{max}}^{+\omega_{max}} \langle f(x), e_\omega(x) \rangle \, e_\omega(x) d\omega$

Manifold Harmonics

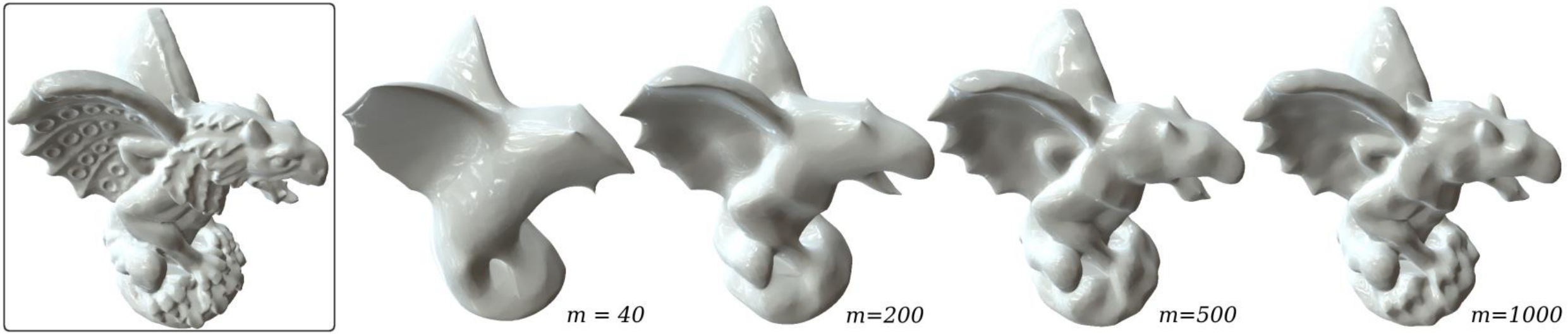
- How to generalize the 1D Fourier Transform to 2-manifold surface?
- sine and cosine functions ⟺ eigenfunctions of the Laplace operator
  - $\Delta e_\omega(x) = -(2\pi\omega)^2 e_\omega(x)$
  - Definition of eigenfunctions of the Laplace operator:
    - $\Delta e_i = \lambda_i e_i$, similar to the eigenvector of a matrix
- Choose eigenfunctions of the Laplace-Beltrami operator on 2-manifold surfaces as generalized basis functions.

Manifold Harmonics

- Discrete Laplace-Beltrami operator: L
  - Symmetry, uniform or cotangent
- Eigenfunctions become the eigenvectors of L
  - an eigenvector can be considered a discrete sampling of a continuous eigenfunction on each vertex.
  - natural vibrations: eigenvectors of L
  - natural frequencies: eigenvalues of L
- L: symmetry and positive semi-definite
  - its eigenvectors build an orthogonal basis
  - For each vector $\boldsymbol{f} = (f_1, \dots, f_n)^T$: $\boldsymbol{f} = \sum_{i=1}^{n} \langle \boldsymbol{f}, \boldsymbol{e}_i \rangle \boldsymbol{e}_i$

Manifold Harmonics

- Low-pass filter:
  - $\boldsymbol{f} = \sum_{i=1}^{m} \langle \boldsymbol{f}, \boldsymbol{e}_i \rangle \boldsymbol{e}_i$ where $m < n$.
  - Replace $\boldsymbol{f}$ with vertex coordinates.



Other filters

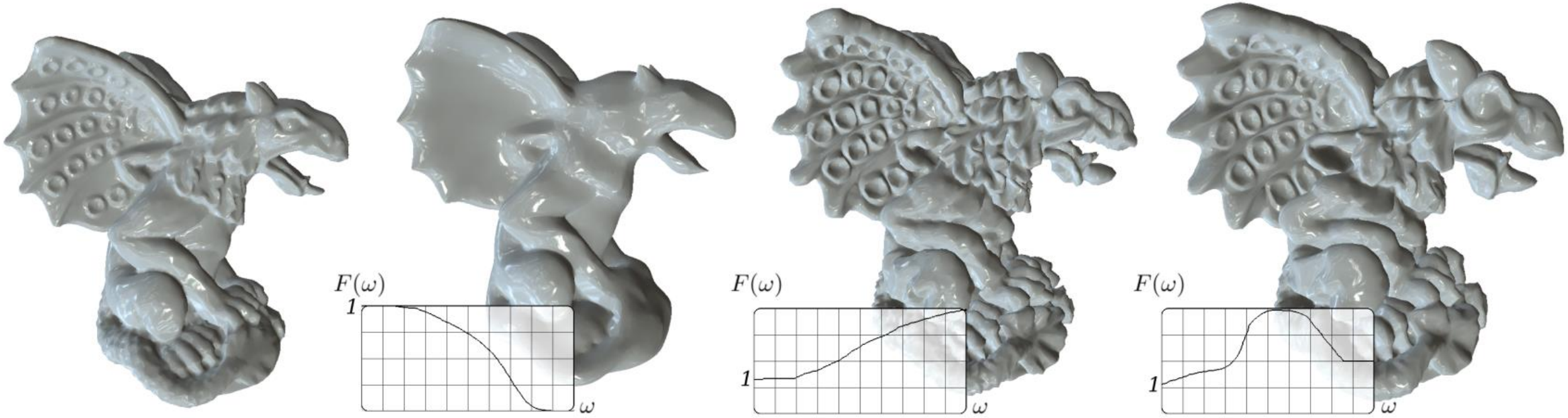


Figure 5: *Low-pass, enhancement and band-exaggeration filters. The filter can be changed by the user, the surface is updated interactively.*

Discussion

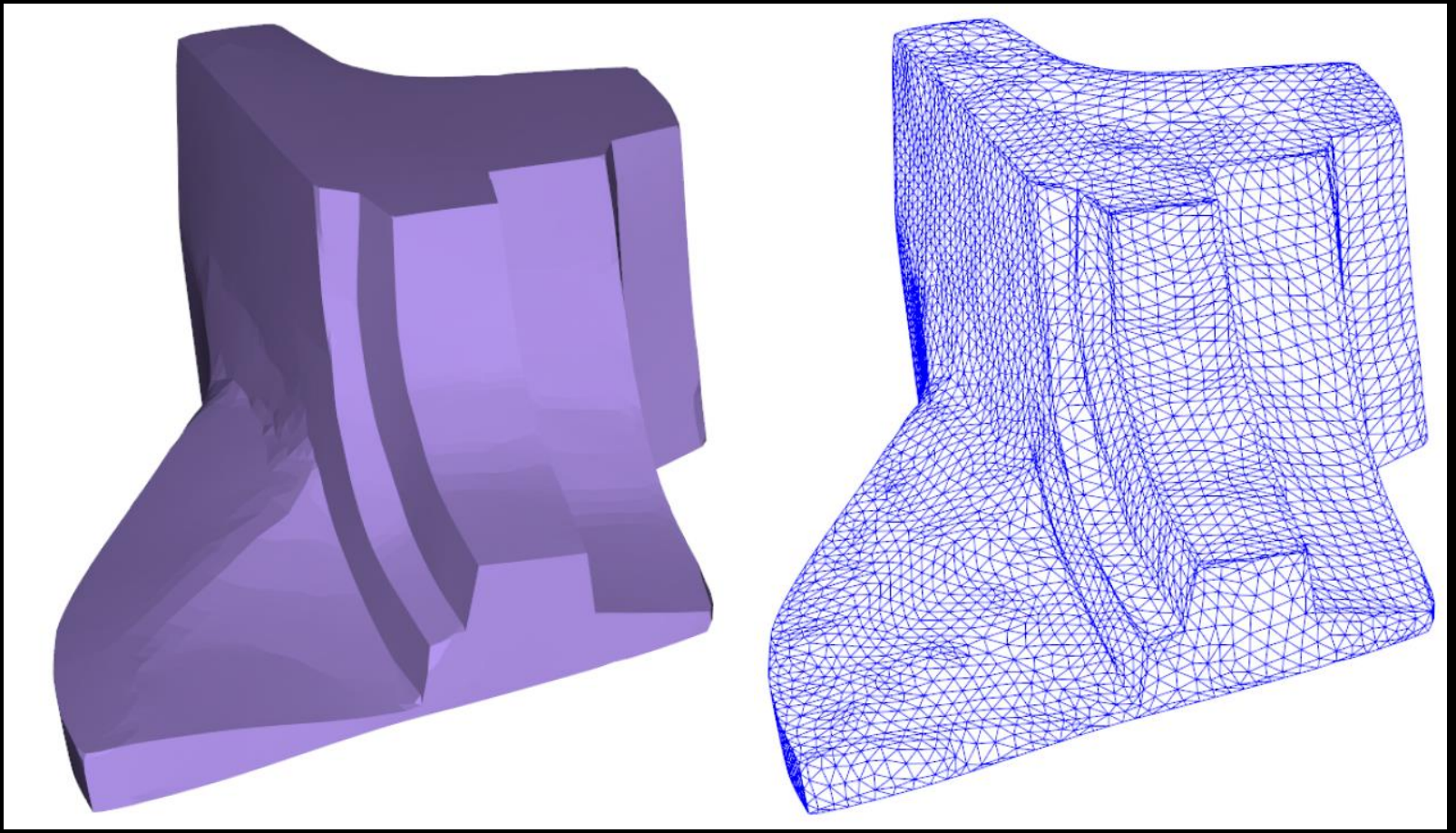
- Computationally expensive
  - eigenvector and eigenvalue
  - Paper: Fast Approximation of Laplace-Beltrami Eigenproblems, SGP 2018
- A very useful representation of triangle mesh:
  - 3D printing
    - Reduced-Order Shape Optimization Using Offset Surfaces, SIGGRAPH 2015
    - Non-Linear Shape Optimization Using Local Subspace Projections, SIGGRAPH 2016
  - Face modeling
    - Use small m to represent the basic shape
    - Use Laplacian coordinate to represent the details
  - …

Outline

- Filter-based methods
- Optimization-based methods
- Data-driven methods

Prior

- The model consists of flat regions.



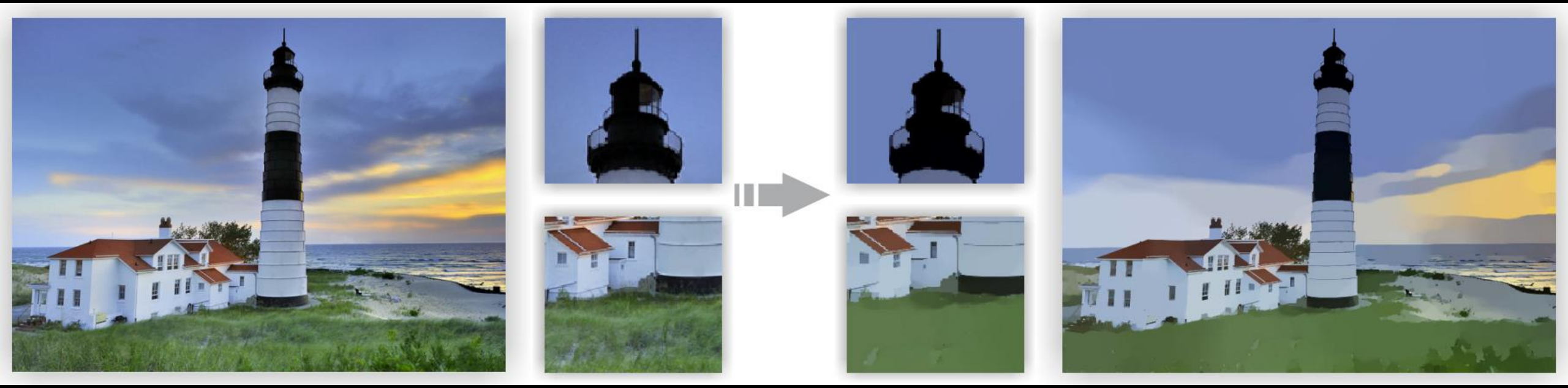
L_0 smoothing

Paper: Mesh denoising via L_0 minimization

- The method maximizes the flat regions of the model and gradually removes noise while preserving sharp features.

L_0 minimization for images

Paper: Image smoothing via L_0 gradient minimization



L_0 minimization for images

Paper: Image smoothing via L_0 gradient minimization

- Energy:

$$|c - c^\star|^2 + |\nabla c|_0$$

- c: a vector of pixel colors
- $c^\star$: original image colors
- ∇c: a vector of gradients of these colors
- $|\nabla c|_0$: L_0 norm of ∇c

Difficult to solve!

Optimization method

- Auxiliary variables δ:

$$\min_{c,\delta} |c - c^{\star}|^2 + \beta|\nabla c - \delta|^2 + \lambda|\delta|_0$$

- Alternating optimization:
- 1. Fix c, solve δ – subproblem:

$$\min_{\delta} \beta|\nabla c - \delta|^2 + \lambda|\delta|_0$$

Analytic solution

- 2. Fix δ, solve c – subproblem:

$$\min_{c,\delta} |c - c^{\star}|^2 + \beta|\nabla c - \delta|^2$$

Quadratic

δ – subproblem

$$\delta_i = \begin{cases} 0, \text{if } \beta(\nabla c_i)^2 \leq \lambda \\ \nabla c_i, \qquad \text{otherwise} \end{cases}$$

1. If $\beta(\nabla c_i)^2 \leq \lambda$, non-zero δ_i yields:

$$\beta(\nabla c_i - \delta_i)^2 + \lambda|\delta_i|_0 = \beta(\nabla c_i - \delta_i)^2 + \lambda \geq \lambda \geq \beta(\nabla c_i)^2$$

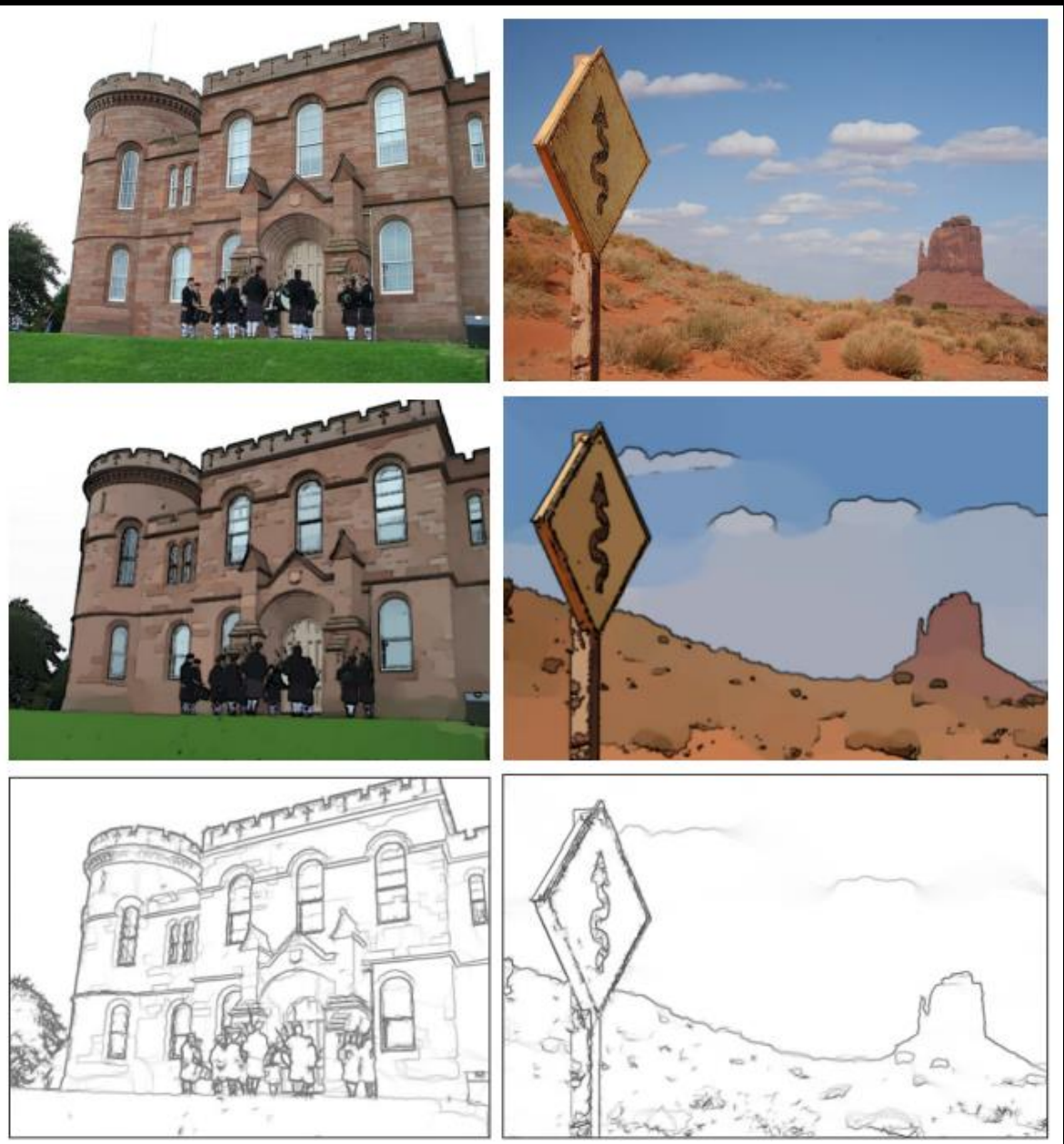
When $\delta_i = 0$, $\beta(\nabla c_i - \delta_i)^2 + \lambda|\delta_i|_0 = \beta(\nabla c_i)^2$.

Thus, the minimum is achieved when $\delta_i = 0$.

2. If $\beta(\nabla c_i)^2 > \lambda$,

When $\delta_i = 0$, $\beta(\nabla c_i - \delta_i)^2 + \lambda|\delta_i|_0 = \beta(\nabla c_i)^2 > \lambda$.

When $\delta_i \neq 0$, the minimum is achieved when $\delta_i = \nabla c_i$ and is λ.



(a) (b)

Figure 13: *Image abstraction and pencil sketching results. Ou method removes the least important structures.*

Mesh denoising

- $c \rightarrow$ points p of input triangle mesh
- $\nabla c \rightarrow$ discrete differential operator?

- It is the key.

∇c

- Requirements:
  - $\nabla c = 0$ when the surface is flat
  - It is irrespective of the rotation or translation of the surface.

- Cot Laplacian operator.

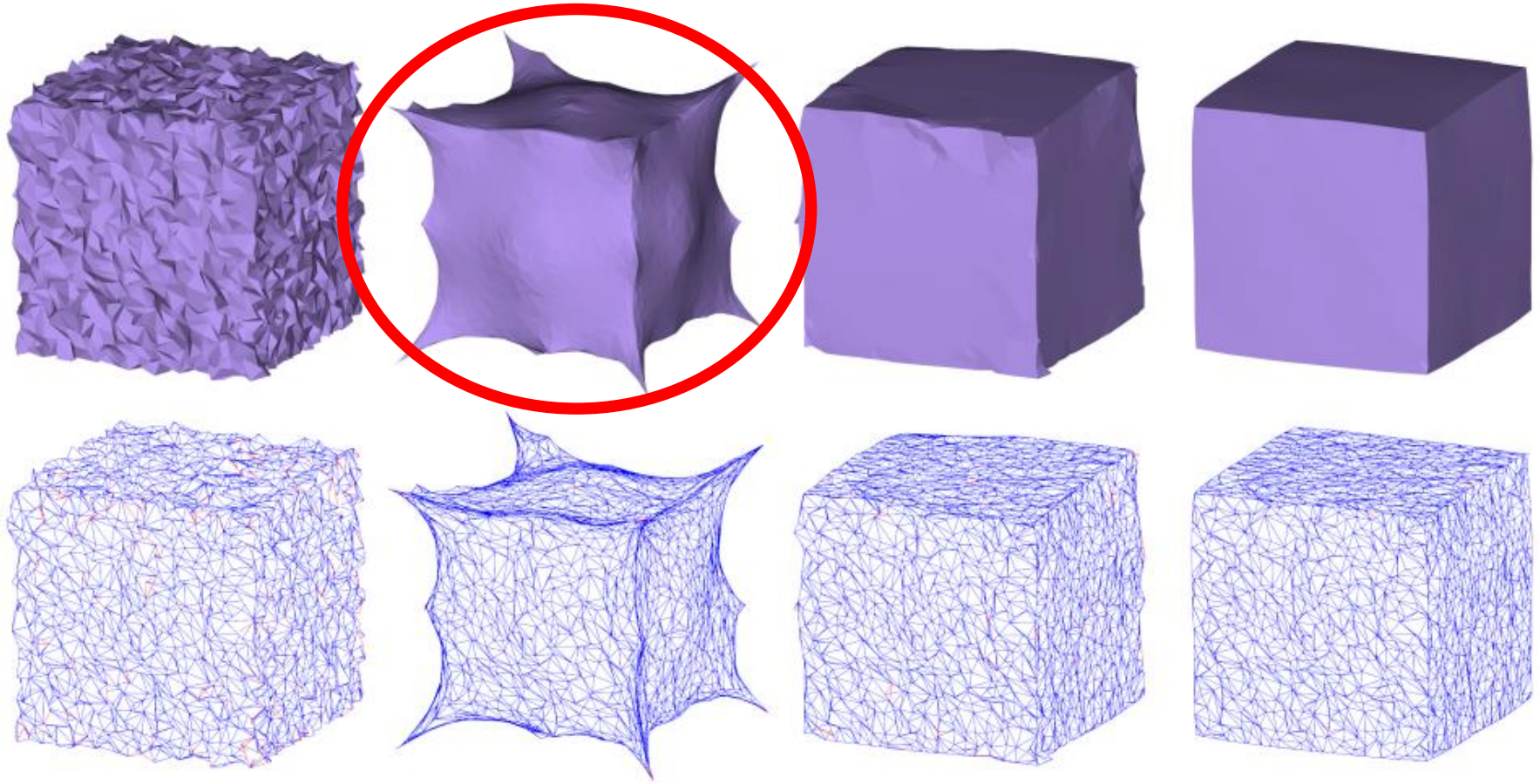
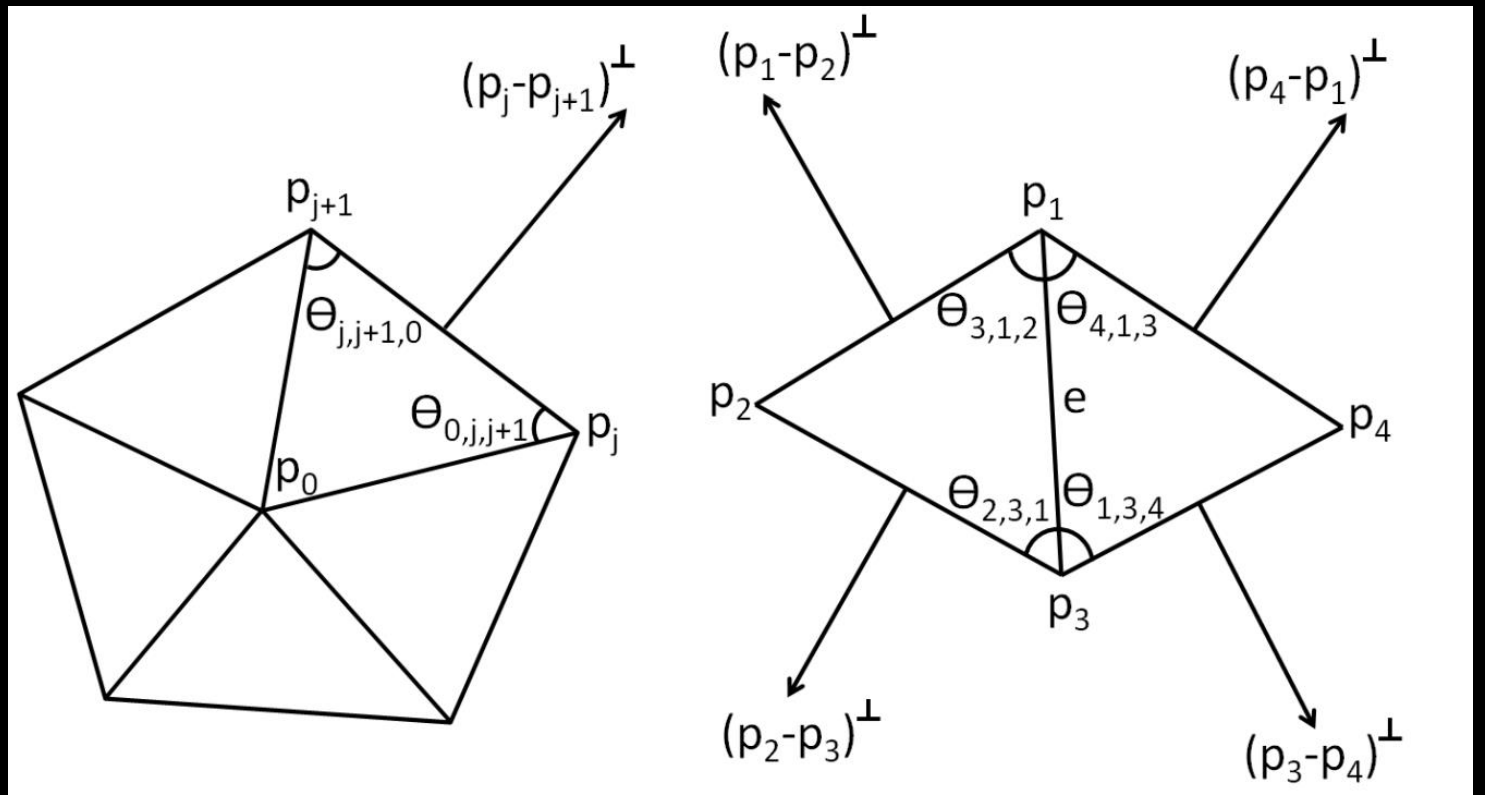


Figure 2: *From left to right: noisy input surface with* $\sigma = 0.3l_e$, *vertex-based cotangent operator, our cotangent edge operator, our area-based edge operator. The bottom row shows wireframes with flipped triangles denoted by red edges. None of these results use regularization.*

1. Fail to reproduce sharp feature
2. Shrink the surface

Differential Edge Operator

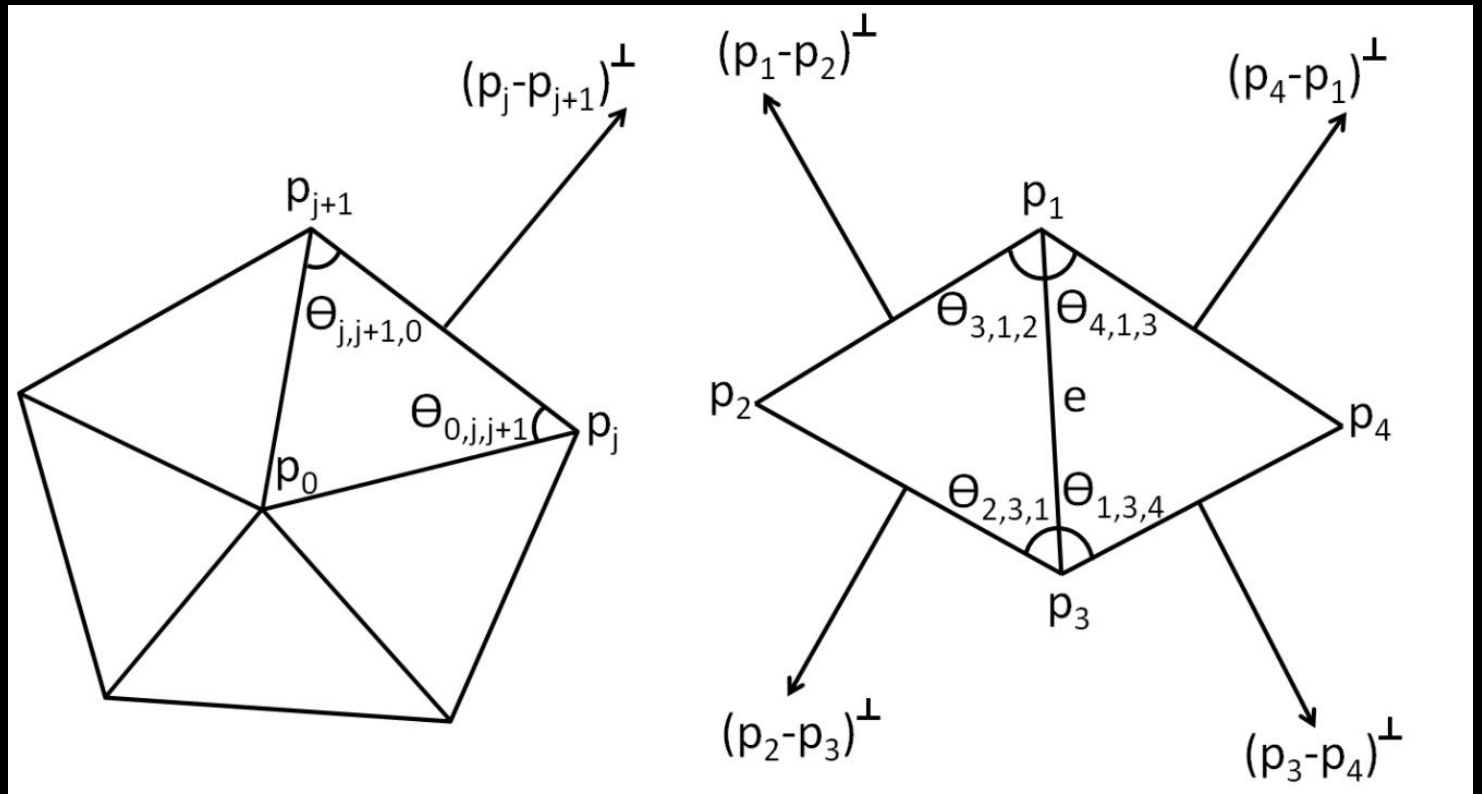
- $(p_j - p_{j+1})^{\perp} = (p_{j+1} - p_0)\cot\theta_{0,j,j+1} + (p_j - p_0)\cot\theta_{j,j+1,0}$
- Vertex-based cot Laplacian operator: $\sum_{j\in\Omega(p_0)}(p_j - p_{j+1})^{\perp}$



Differential Edge Operator

- Simialr to vertex version:

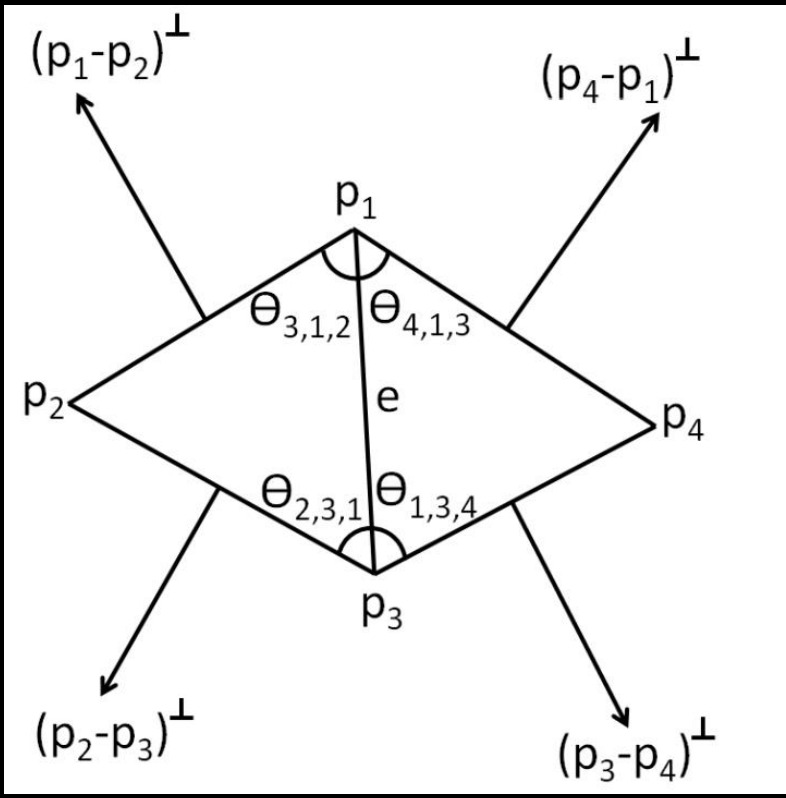
$$D(e) = \sum_{j \in \Omega(e)} \left(p_j - p_{j+1}\right)^{\perp}$$



Differential Edge Operator

- $D(e) = \begin{bmatrix} -\cot\theta_{2,3,1} - \cot\theta_{1,3,4} \\ \cot\theta_{2,3,1} + \cot\theta_{3,1,2} \\ -\cot\theta_{3,1,2} - \cot\theta_{4,1,3} \\ \cot\theta_{1,3,4} + \cot\theta_{4,1,3} \end{bmatrix}^T \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$

- $|D(e)| = 2\sin(\frac{\gamma}{2})\,|p_3 - p_1|$



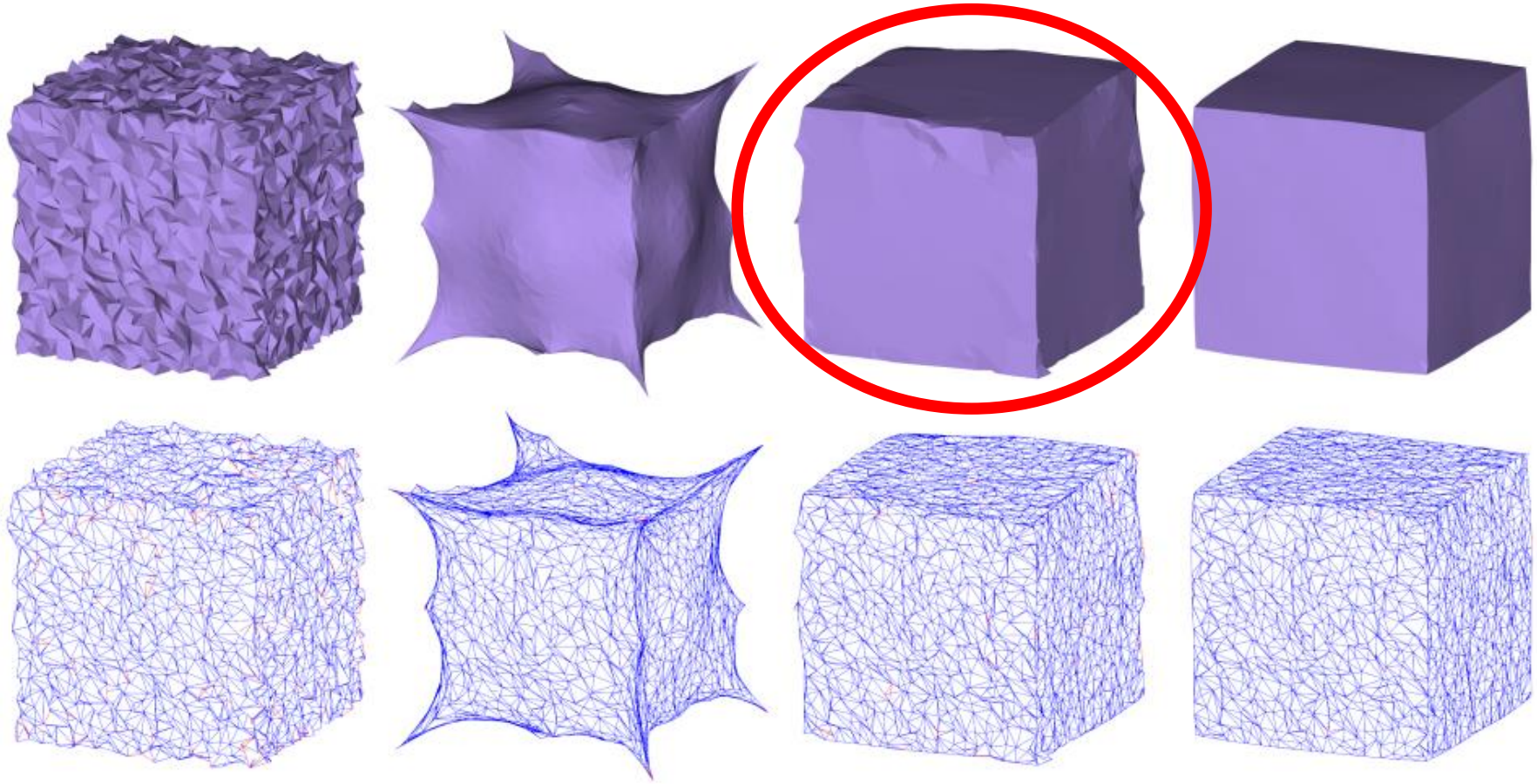


Figure 2: *From left to right: noisy input surface with* $\sigma = 0.3l_e$*, vertex-based cotangent operator, our cotangent edge operator, our area-based edge operator. The bottom row shows wireframes with flipped triangles denoted by red edges. None of these results use regularization.*

The issue stems from degenerate triangles where the cot weights approach infinity as an angle approaches zero.

Area-based edge operator

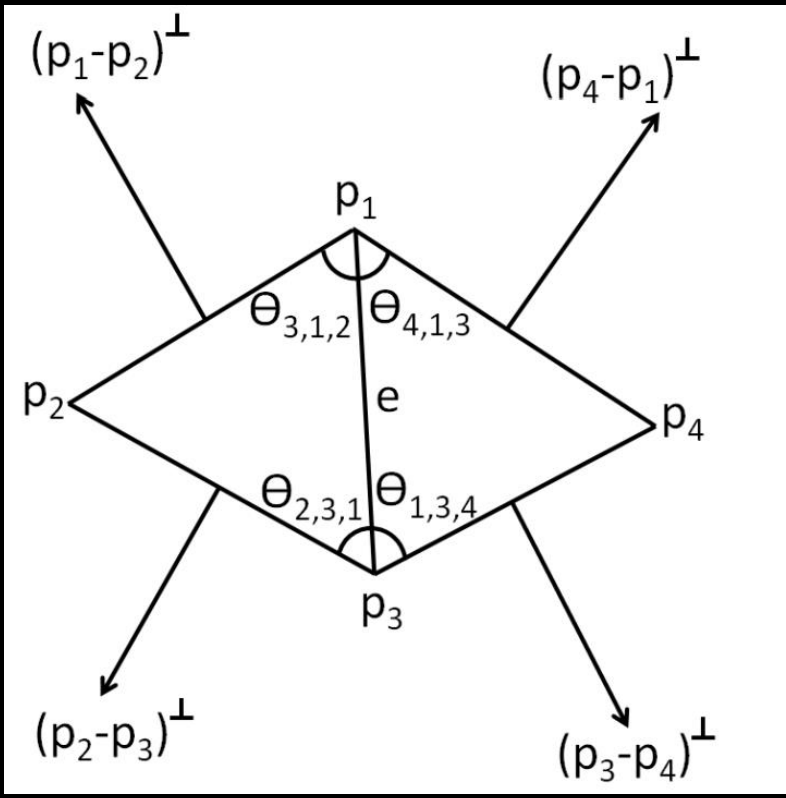
- LINEAR PRECISION:
- $0 = \sum_{j\in\Omega(i)} \omega_{ij}(p_i - p_j) = \sum_{k\in\Omega(i)\cup i} \omega'_{i,j} p_k$
- At the same time: $0 = \sum_{k\in\Omega(i)\cup i} \omega'_{i,j}$

- Similarly: when p_j are planar:

$$0 = \sum_j \omega_j p_j, 0 = \sum_j \omega_j$$

$\omega_1 = -\Delta_{2,3,4}, \omega_2 = \Delta_{1,3,4},$

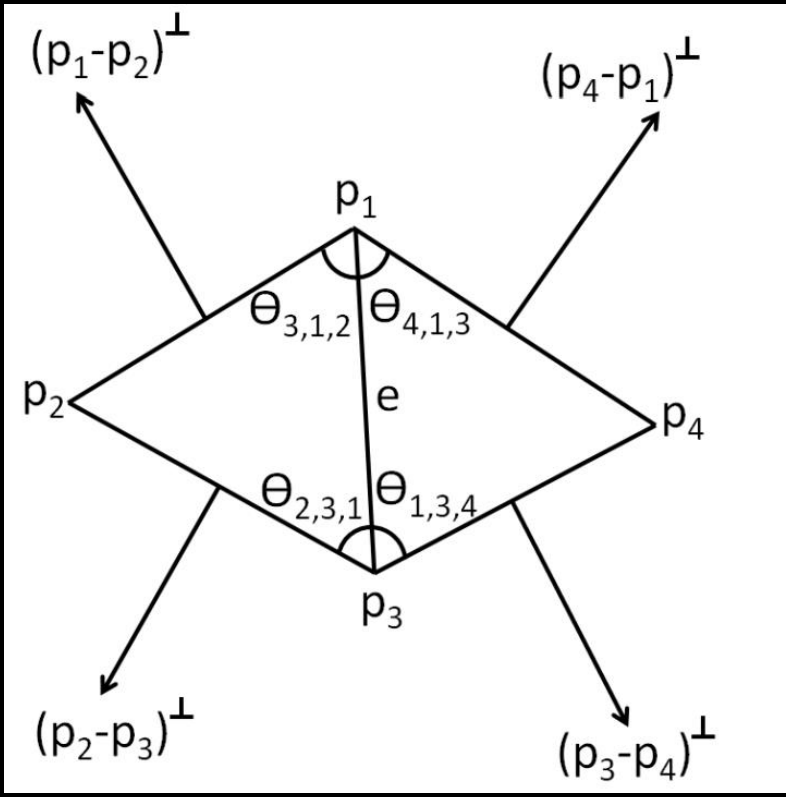
$\omega_3 = -\Delta_{1,2,4}, \omega_4 = \Delta_{1,2,3}$



Area-based edge operator

- It is not scale-independent.
- Scaled by $\Delta_{1,3,4} + \Delta_{1,2,3}$

- How to compute $\Delta_{2,3,4}$ and $\Delta_{1,2,4}$?
  - an isometric unfolding of the surface around the shared edge



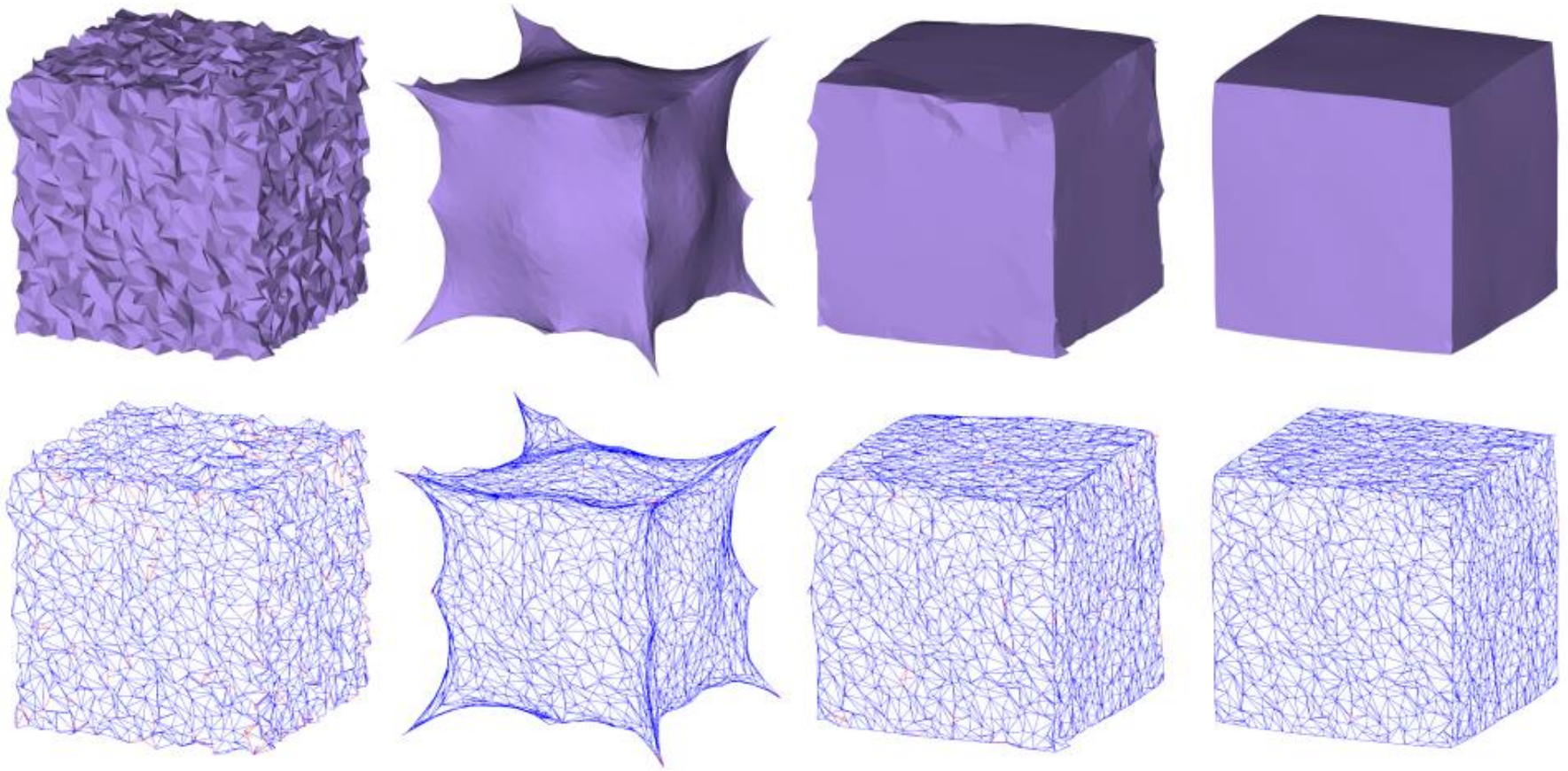


Figure 2: *From left to right: noisy input surface with* $\sigma = 0.3l_e$*, vertex-based cotangent operator, our cotangent edge operator, our area-based edge operator. The bottom row shows wireframes with flipped triangles denoted by red edges. None of these results use regularization.*

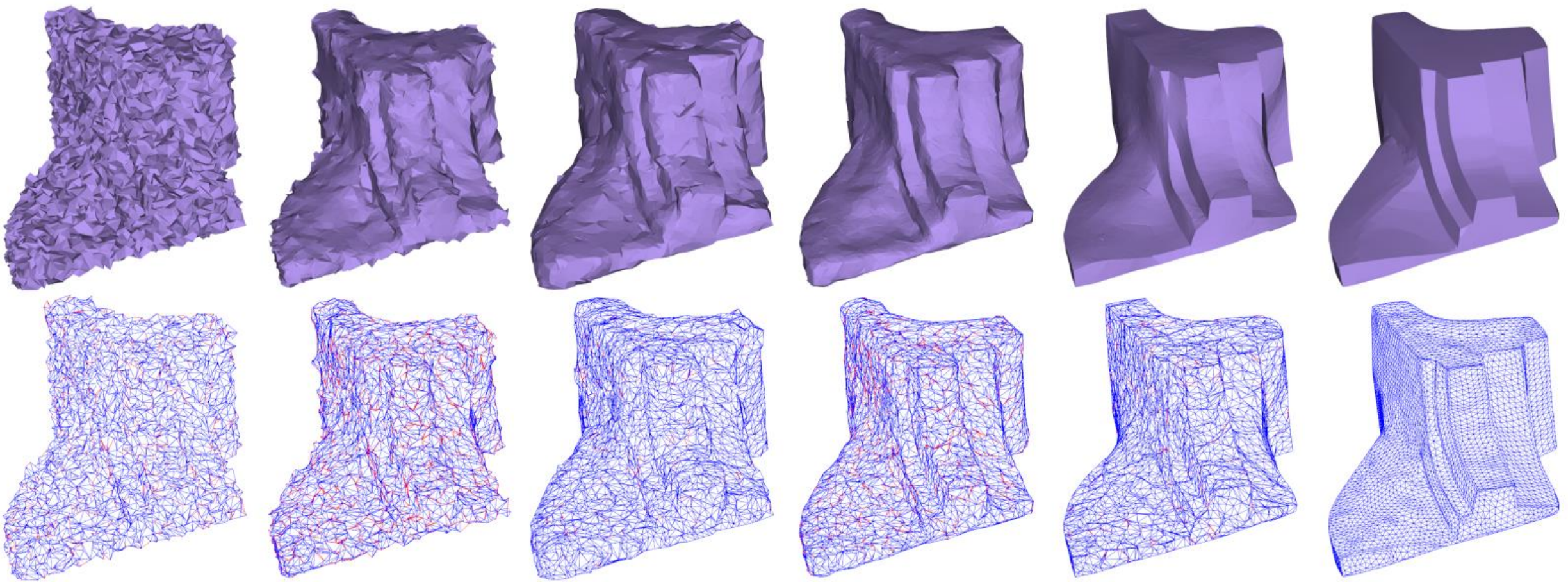


Figure 9: *From left to right: the input mesh with large noise in random directions, bilateral filtering [Fleishman et al. 2003], prescribed mean curvature flow [Hildebrandt and Polthier 2004], mean filtering [Yagou et al. 2002], bilateral normal filtering [Zheng et al. 2011], our result. We show the wireframe of each surface below.*

L_0 smoothing

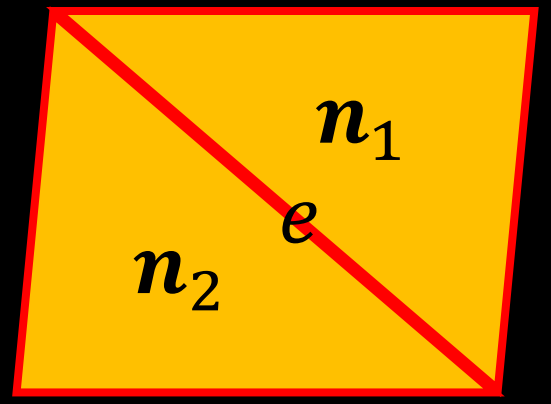
Paper: Mesh denoising via L_0 minimization

- Paper: http://faculty.cs.tamu.edu/schaefer/research/L0Smoothing.pdf
- Slides: http://faculty.cs.tamu.edu/schaefer/research/slides/L0Smoothing.pdf
- Blog: http://www.cnblogs.com/shushen/p/5113484.html

Total Variation-based method

Paper: Variational Mesh Denoising using Total Variation and Piecewise Constant Function Space

- Replace the vertex positions with the normals.
  - Facet normal filtering
    - Total Variation
  - Vertex updating
    - Iterative updating
- How to remove the noise and preserve the sharp feature?
  - Sharp feature is sparse.
  - Normal difference on edge is sparse.



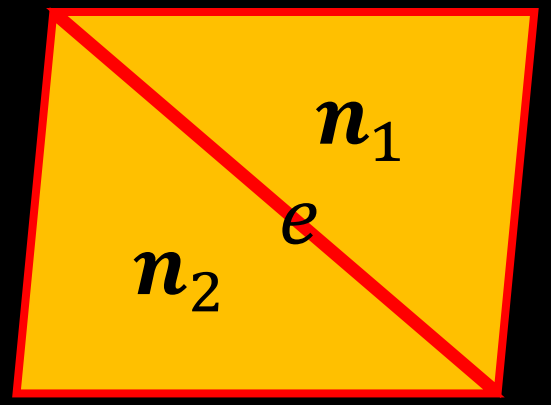
Total Variation

$$\min E_{TV} + \alpha E_a$$

$$E_{TV} = \sum_e \omega_e \cdot l_e \sqrt{\|\nabla \boldsymbol{n}_e\|_2^2}$$

$$E_a = \sum_f \|\boldsymbol{n}_f - \boldsymbol{n}_f^{in}\|_2^2$$

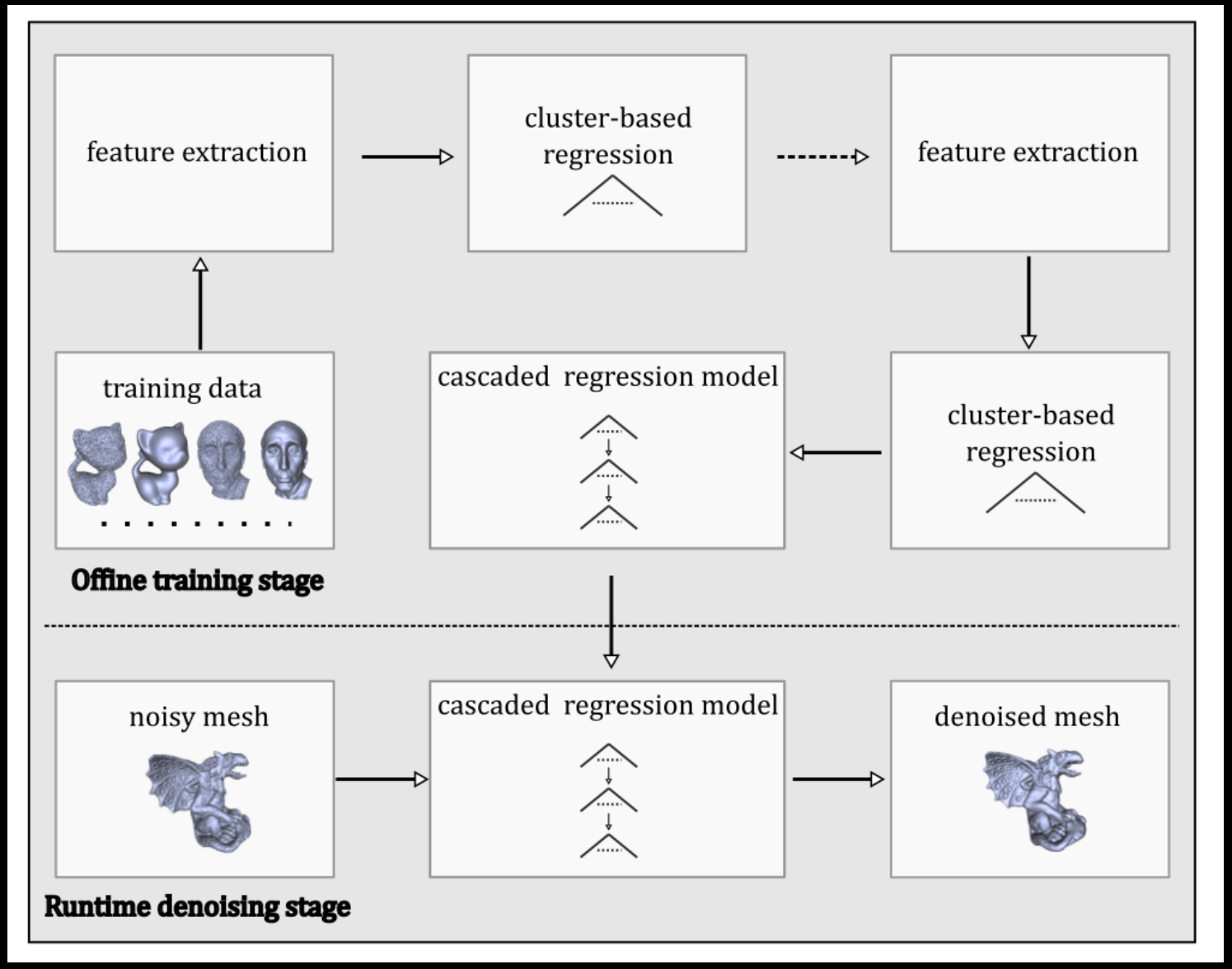
1. $\nabla \boldsymbol{n_e} = \boldsymbol{n}_1 - \boldsymbol{n}_2$
2. $\omega_e = \exp\left(-\|\boldsymbol{n}_1^{in} - \boldsymbol{n}_2^{in}\|_2^4\right)$



Outline

- Filter-based methods
- Optimization-based methods
- Data-driven methods

Mesh Denoising via Cascaded Normal Regression



A highly nonlinear function

Overview

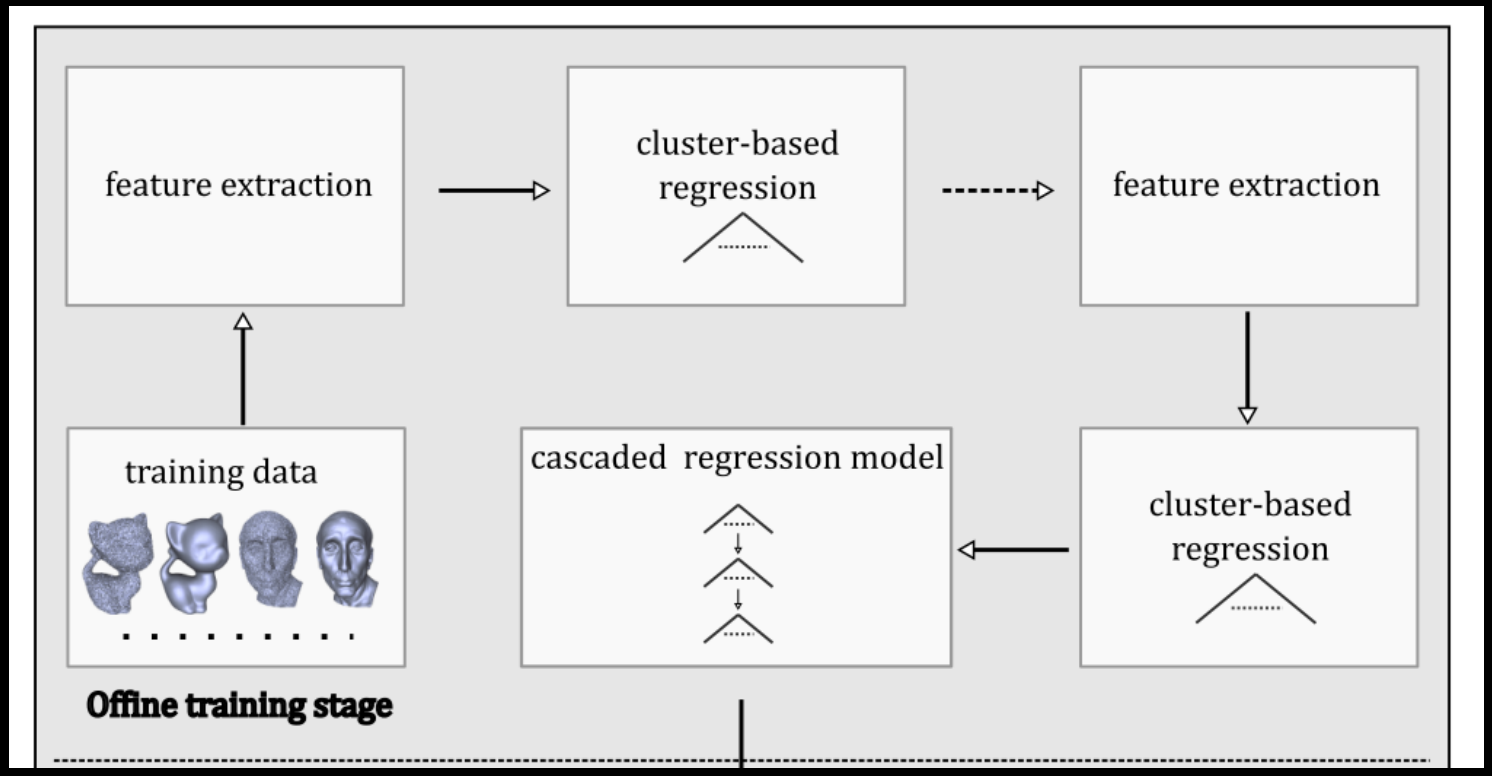
- Goal: learn the relationship between noisy geometry and the ground-truth geometry

$$n_f = \mathcal{F}(\Omega_f)$$

Ω_f: local noisy region

Cascaded Regression

- The output from the current regression function serves as the input of the next regression function.
- Each regression function: a neural network with a single hidden layer



Offline training stage

- A training pair: $(S_i, \bar{n}_i)$

S_i: filtered facet normal descriptor (FND) of i^{th} facet

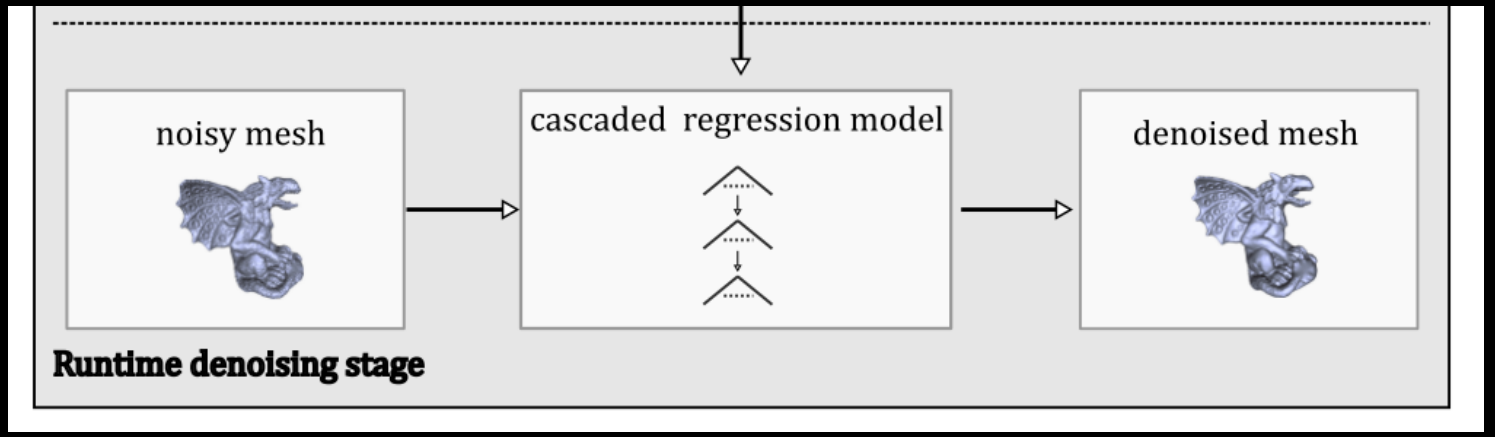
$\bar{n}_i$: ground-truth facet normal

- Goal: learn the function:

$$\mathcal{F}: S_i \rightarrow \bar{n}_i, \forall i$$

Runtime denoising stage

- Extract FND for each facet
- Apply $\mathcal{F}$ to obtain new normal for each facet
- Recover vertices with known normal



Bilateral Normal Filtering

$$\boldsymbol{n}_i^{k+1} \leftarrow \frac{1}{K_{\boldsymbol{p}}} \sum_{f_j} A_j W_s\left(\left\|\boldsymbol{c}_i - \boldsymbol{c}_j\right\|\right) W_r\left(\left\|\boldsymbol{n}_i^k - \boldsymbol{n}_j^k\right\|\right) \cdot \boldsymbol{n}_j^k$$

Parameters: σ_s, σ_r, iteration number K

- Bilateral filtered facet normal descriptor (B-FND)

S_i

$$:= \left(\boldsymbol{n}_i^1\left(\sigma_{s_1}, \sigma_{r_1}\right), \ldots, \boldsymbol{n}_i^1\left(\sigma_{s_L}, \sigma_{r_L}\right), \boldsymbol{n}_i^2\left(\sigma_{s_1}, \sigma_{r_1}\right), \ldots, \boldsymbol{n}_i^2\left(\sigma_{s_L}, \sigma_{r_L}\right), \ldots,$$

Guided bilateral filter (Joint bilateral filter)

$$\boldsymbol{n}_i^{k+1} \leftarrow \frac{1}{K_{\boldsymbol{p}}} \sum_{f_j} A_j W_s\left(\left\|\boldsymbol{c}_i - \boldsymbol{c}_j\right\|\right) W_r\left(\left\|g\left(\boldsymbol{n}_i^k\right) - g\left(\boldsymbol{n}_j^k\right)\right\|\right) \cdot \boldsymbol{n}_j^k$$

In this paper, $g\left(\boldsymbol{n}_i^k\right) = \frac{1}{K_{\boldsymbol{p}}} \sum_{f_j} A_j W_s\left(\left\|\boldsymbol{c}_i - \boldsymbol{c}_j\right\|\right) \cdot \boldsymbol{n}_j^k$ Gaussian normal filter

- Guided filtered facet normal descriptor (G-FND)

$$S_i^g \coloneqq \left(\boldsymbol{n}_{g,i}^1\left(\sigma_{s_1}, \sigma_{r_1}\right), \ldots, \boldsymbol{n}_{\boldsymbol{g},i}^1\left(\sigma_{s_L}, \sigma_{r_L}\right),\right.$$

Training data

- A dataset: $D = \{S_i, \bar{n}_i\}_{i=1}^{N}$
- First Partition the training data into K_c clusters via a k-means algorithm
- For each cluster D_l: 85% the training set D_{l1}, 15% validation set D_{l2}

Cluster-based regression

- Cost function:

$$E := \sum_{i \in D_{l1}} \left\| \Lambda\left(\Phi_l(S_i)\right) - \bar{n}_i \right\|^2 + \lambda E_{reg}$$

E_{reg}: commonly used L2 regularization term of unknown parameters

Φ_l: regression function as a single-hidden layer feed forward network N_r hidden nodes

$$\Phi_l(S) = \sum_{k=1}^{N_r} \exp\left(-\left\| W_{l,k}^T \bar{S} - b_{l,k} \right\|^2\right) \boldsymbol{a}_{l,k}$$

$\bar{S}$: feature standardization version of S

$W_{l,k}^T \in R^{3LK}$, $b_{l,k} \in R$, $\boldsymbol{a}_{l,k} \in R^3$

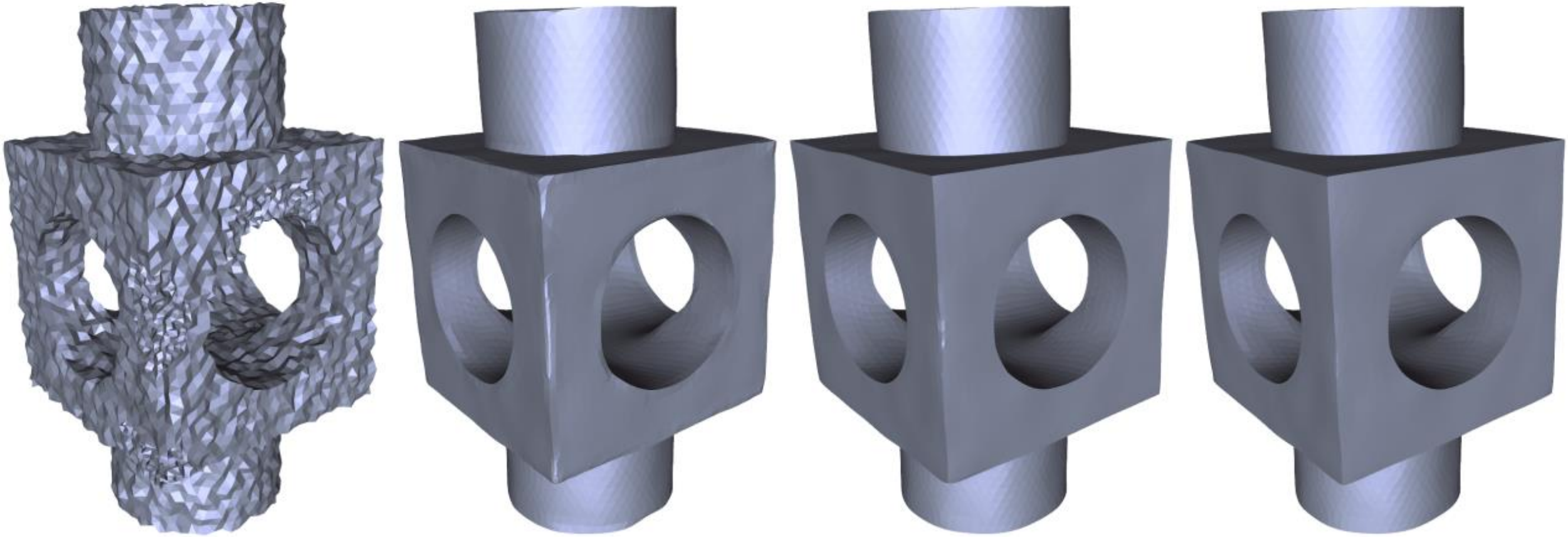
Regression function

$$\mathcal{F}(S) \coloneqq \Phi_l(S), \text{ if } \|S - c_l\| \leq \|S - c_k\|, \forall k.$$

c_l: cluster center of D_l.

Cascaded scheme

- G-FND in the first regression function

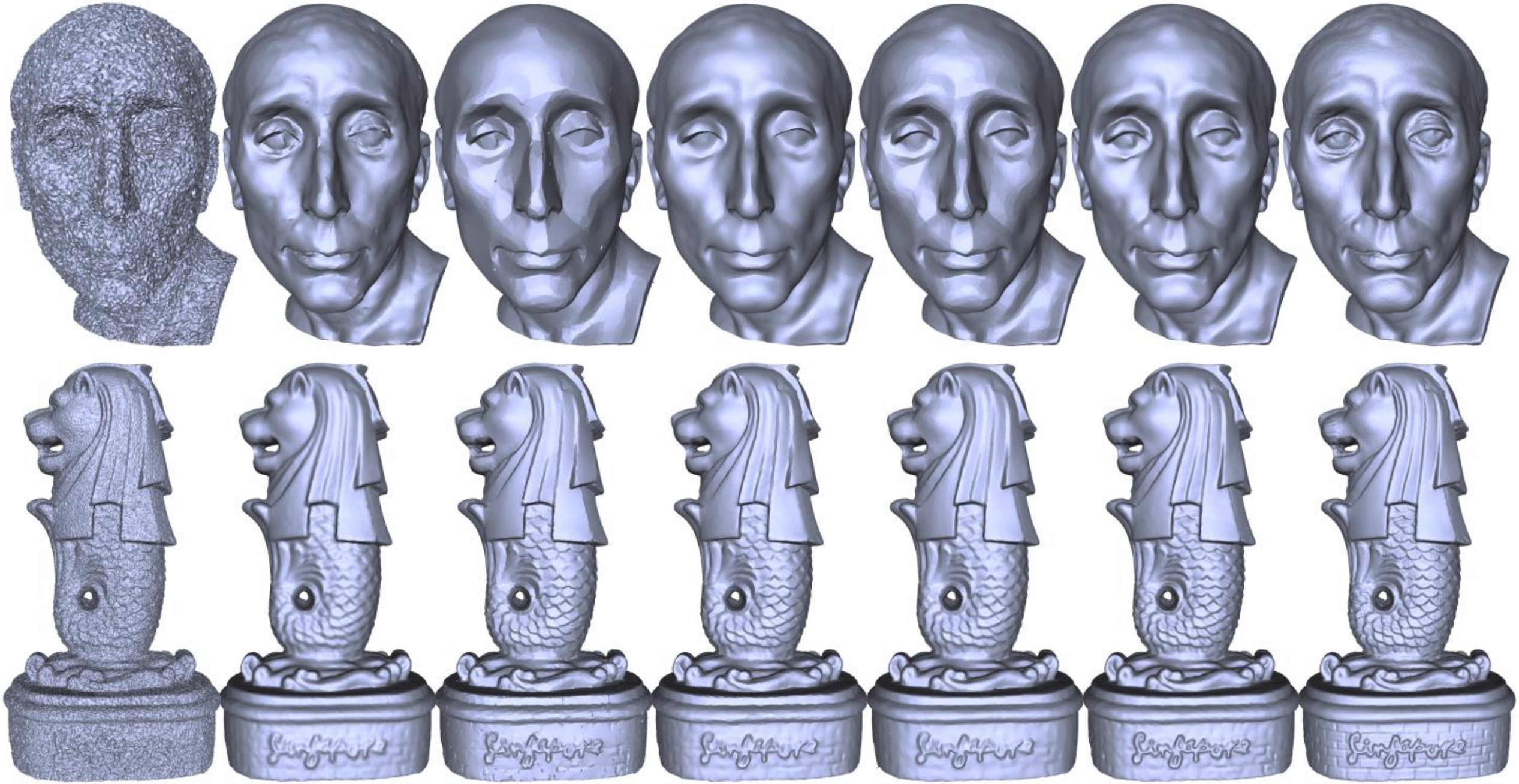


(a) Input (b) G-FND (c) B-FND (d) Both

Choice of hyperparameters

- σ_s: $\{\bar{l}_e, 2\bar{l}_e\}$, $\bar{l}_e$ is the average edge length.
- σ_r: $\{0.1, 0.2, 0.35, 0.5, \infty\}$
- $K = 1$

- 3 cascaded regressions are enough to generate good results.
- $K_c = 4$ after testing.
- $N_r = 20$ after testing.



(a) Noisy input **(b)** Bilateral normal **(c)** L_0 smoothing **(d)** Guided normal **(e)** Bayesian **(f)** Our method **(g)** Ground-truth