Mesh Smoothing

USTC, 2024 Spring

Qing Fang, fq1208@mail.ustc.edu.cn

https://qingfang1208.github.io/

Introduction

Smoothing – from wiki

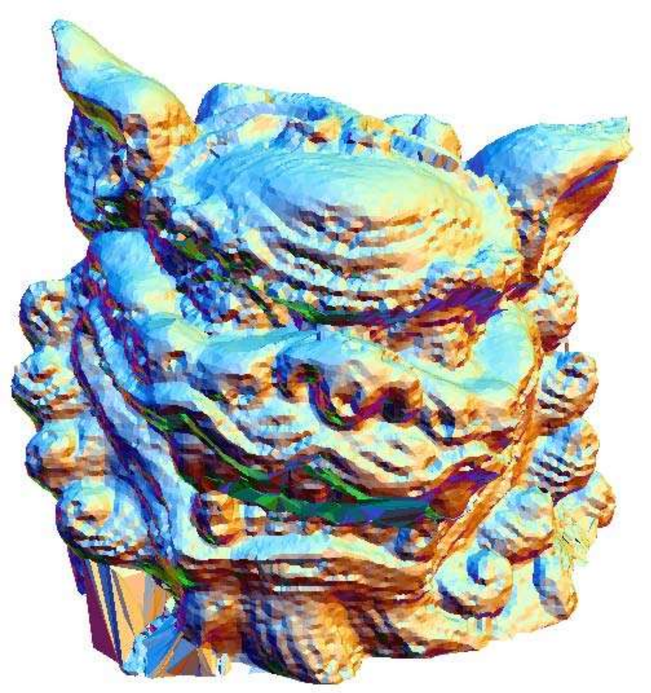
- In statistics and image processing, to smooth a data set is to create an approximating function that attempts to capture important patterns in the data, while leaving out noise or other fine-scale structures/rapid phenomena.
- Denoising and fairing

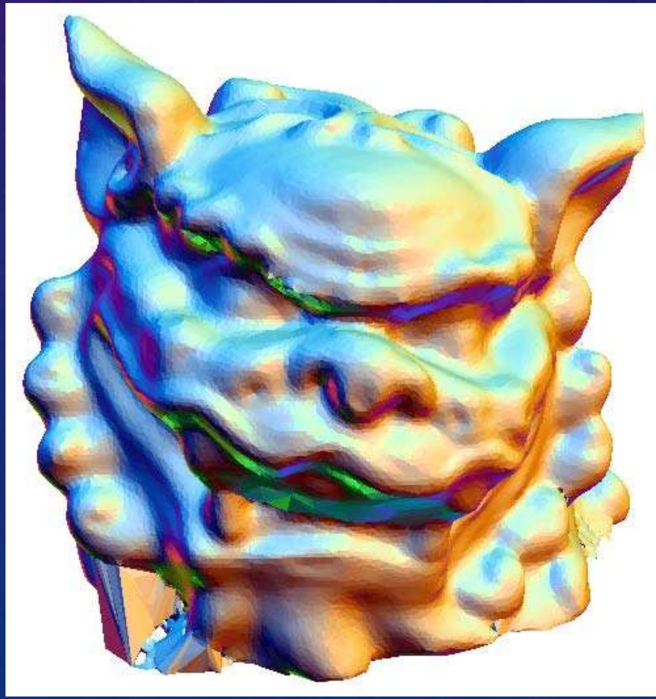
Denoising

- Meshes obtained from real world objects are often noisy.

Denoising

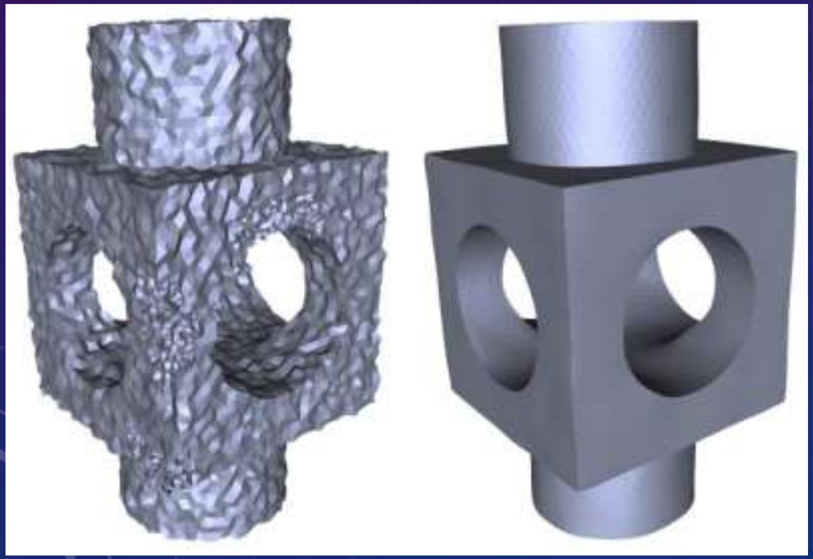
- Eliminate noises in high frequency and preserve features.

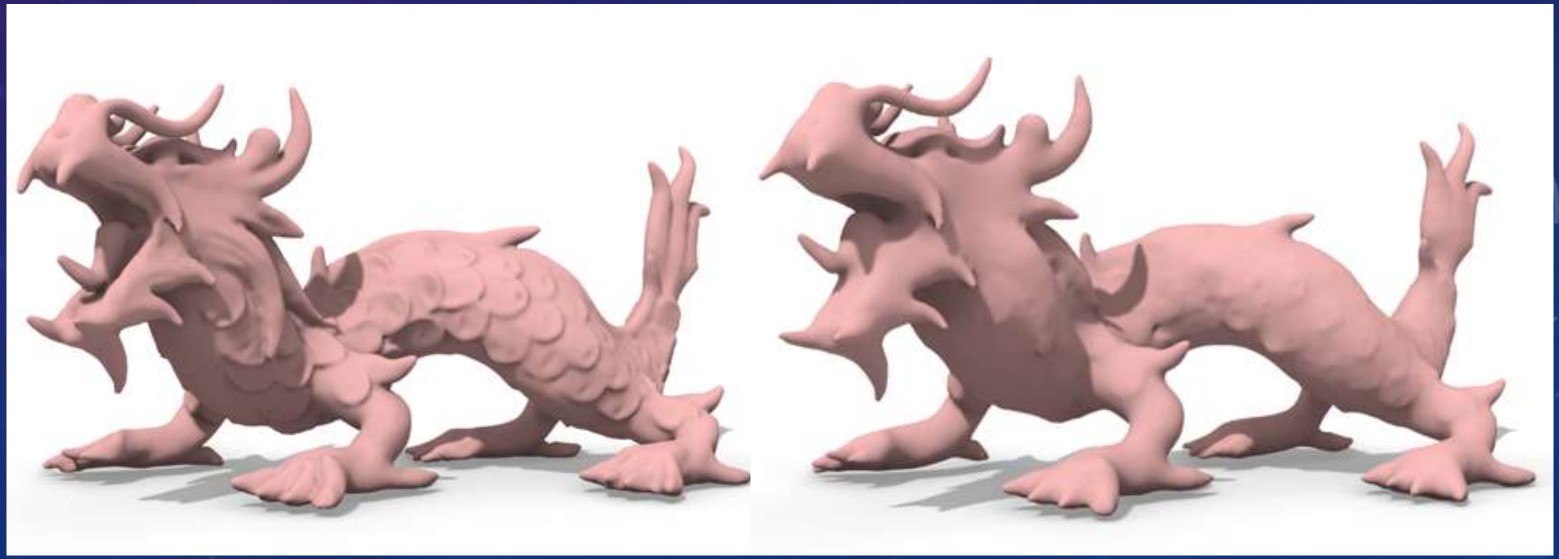




Noises and features

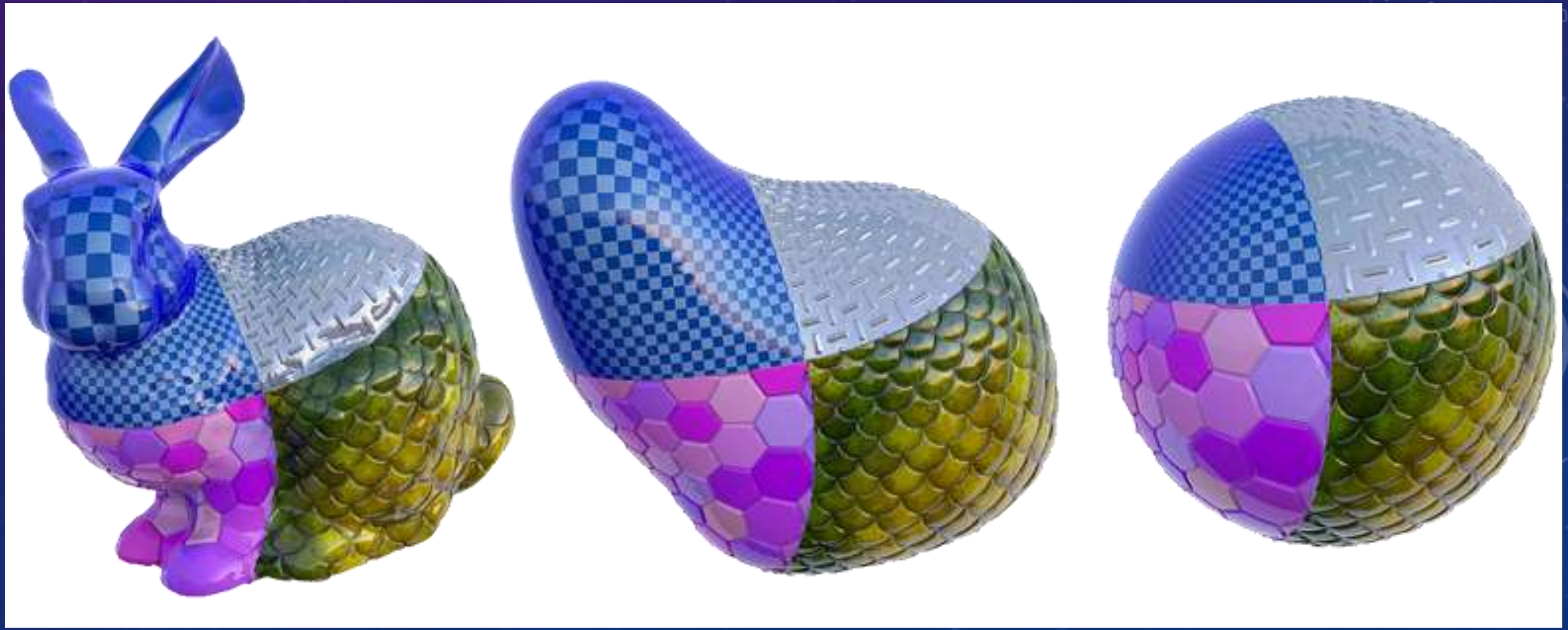
- What is noise on a surface? What is feature on a surface?
- High frequencies and low frequencies





Fairing

- Compute shapes that are as smooth as possible



Mesh smoothing

- Which part to be removed/preserved?
- Certain prior assumptions
  - Geometric
  - Semantic

Methods

Outline

- Filter-based methods
  - Laplacian smoothing
  - Bilateral denoising
  - Spectral filters
- 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: $\frac{\partial f(x,t)}{\partial t} = \lambda \Delta f(x, t)$

Laplacian smoothing

- Diffusion equation: $\frac{\partial f(x,t)}{\partial t} = \lambda \Delta f(x,t)$
  - A second-order linear partial differential equation
  - Smooth an arbitrary function f on a manifold surface by using Laplace-Beltrami operator
  - Discretize the equation both in space and time

Spatial discretization

- Sample values at the mesh vertices $f(x,t) = \{f(v_i,t), i = 1, \dots, n\}$
- Discrete Laplace calculated on vertices.
- Matrix form: $\vec{F}(t) = \left(f(v_1,t), \dots, f(v_n,t)\right)^T$

$$\frac{\partial \vec{F}(t)}{\partial t} = \lambda L \vec{F}(t)$$

Temporal discretization

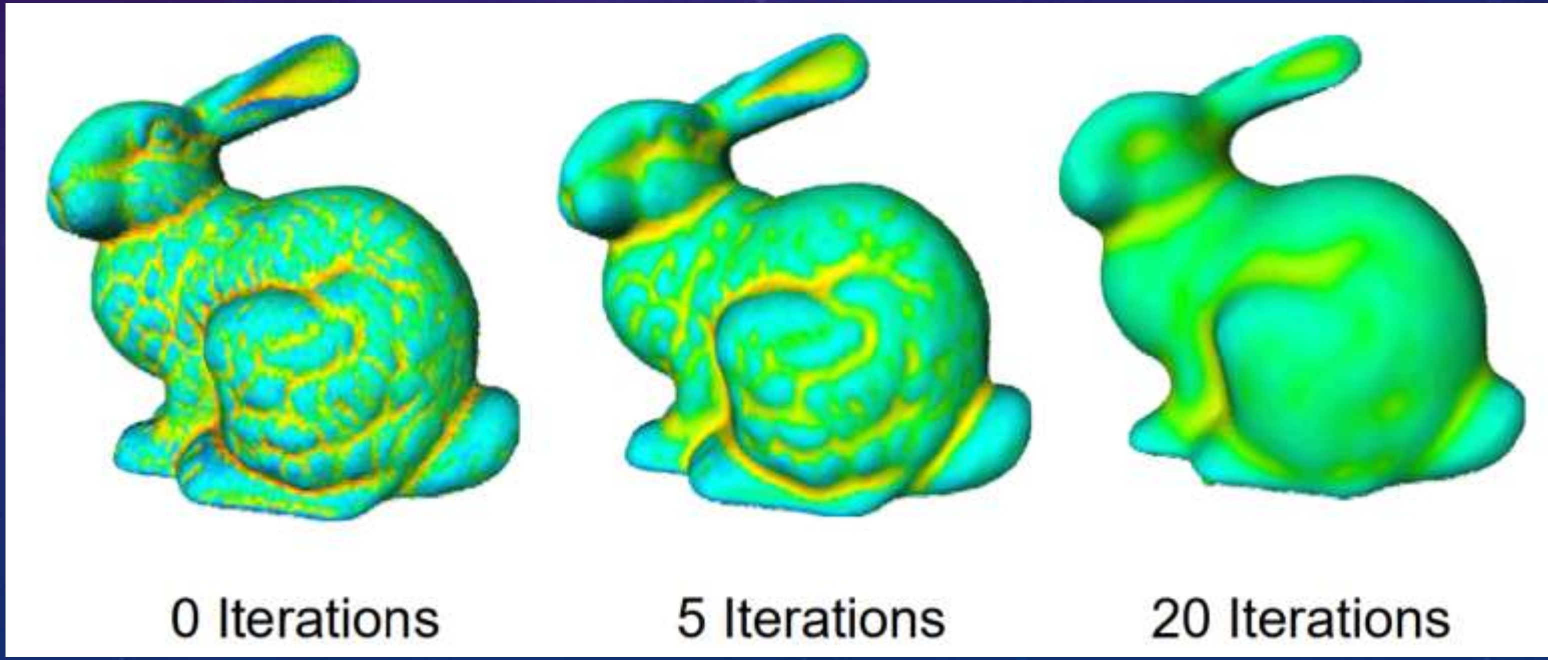
- Uniform sampling: $(t, t+h, t+2h, \dots)$
- Explicit Euler integration ($h \to 0$): $\vec{F}(t+h) = \vec{F}(t) + h\frac{\partial \vec{F}(t)}{\partial t} = \vec{F}(t) + h\lambda L\vec{F}(t)$
- Implicit Euler integration:

$$\vec{F}(t+h) = \vec{F}(t) + h\lambda L\vec{F}(t+h)$$

$$(I - h\lambda L)\vec{F}(t+h) = \vec{F}(t)$$

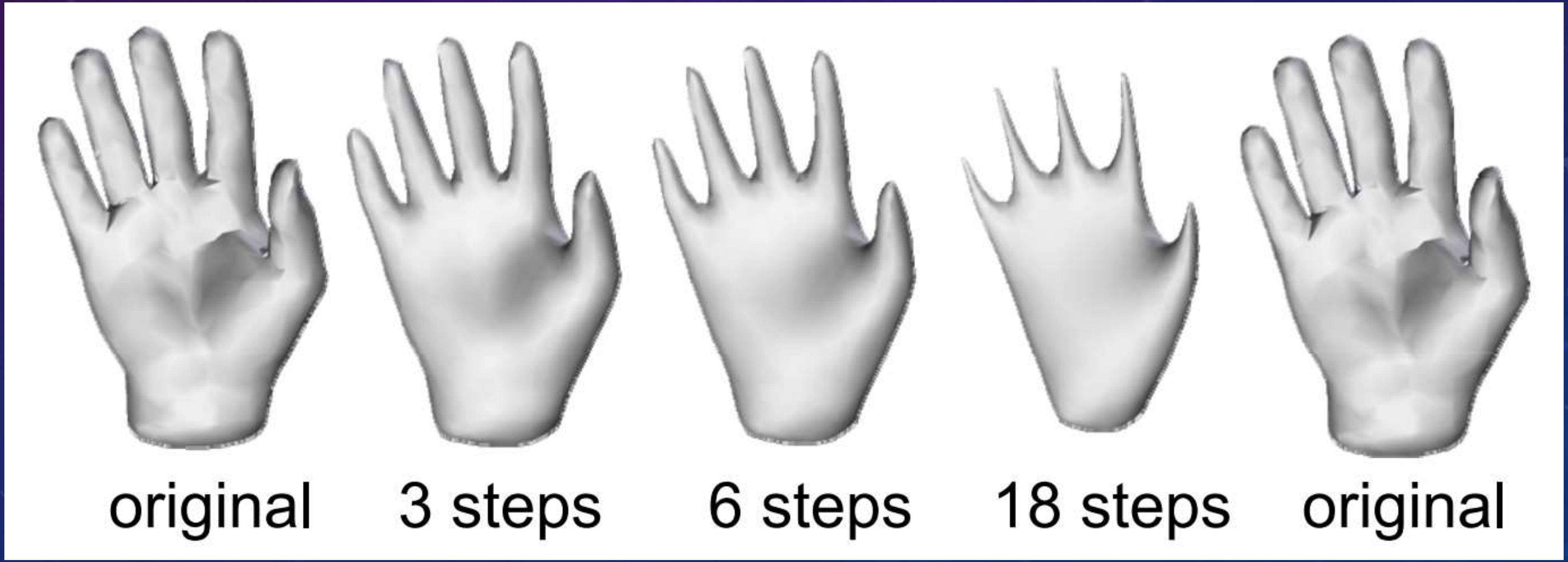
Laplacian smoothing

- Function: $\vec{F}(t) = \vec{p}(t) = (p_1(t), \dots, p_n(t))^T \quad n \times 3$
- Laplacian smoothing: $p_i \longleftarrow p_i + h\lambda(L\vec{p})_i$



Problem - shrinkage

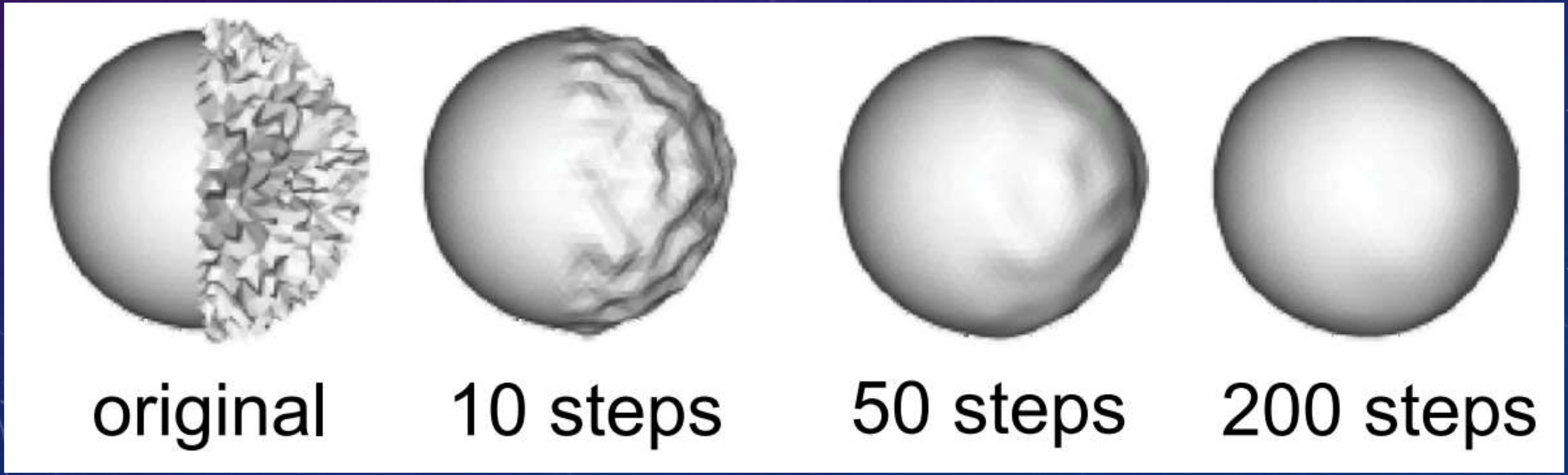
- Repeated iterations of Laplacian smoothing shrinks the mesh



Improved Laplacian

- Taubin smoothing: Laplacian + expansion

$$p_i \leftarrow p_i + h\lambda(L\vec{p})_i, \lambda > 0; p_i \leftarrow p_i + h\mu(L\vec{p})_i, \mu < 0$$



Improved Laplacian

- Mean curvature : $2H_iN_i = (L\vec{p})_i$

$$\vec{F}(t+h) = \vec{F}(t) + h\lambda L\vec{F}(t), \lambda > 0;\ \vec{F}(t+h) = \vec{F}(t) + h\mu L\vec{F}(t), \mu < 0$$

Fairing

- Steady-states of the flow:

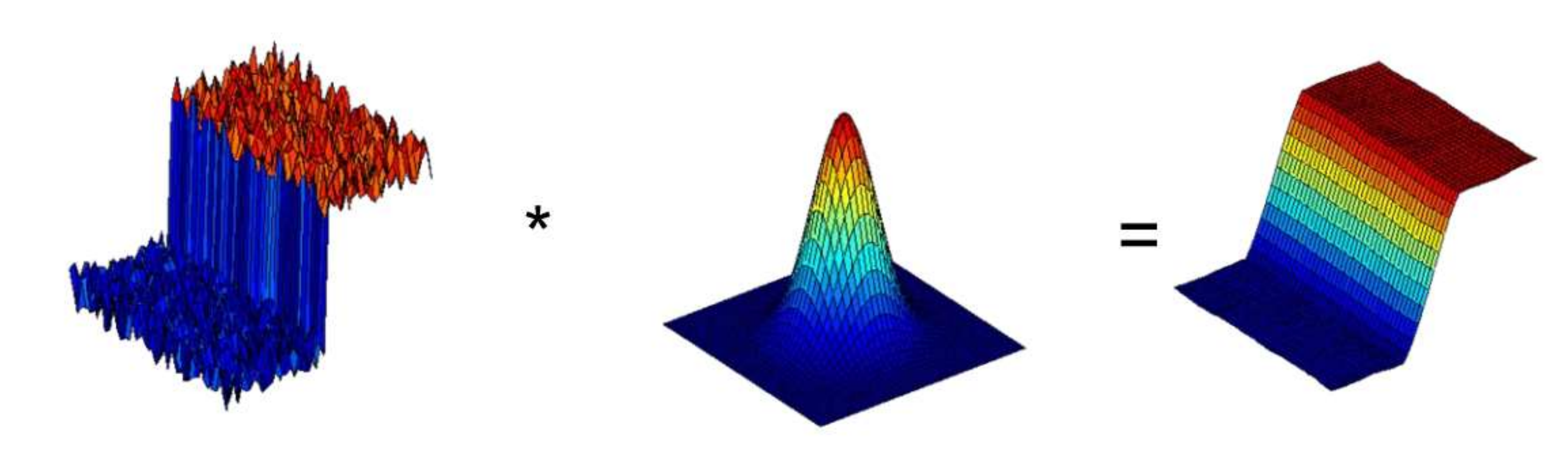
$Lx = 0$ $L^2\boldsymbol{x} = 0$ $L^3\boldsymbol{x} = 0$

Bilateral mesh denoising

- Gaussian filter: $I(p) \longleftarrow \frac{1}{K_p} \sum_{q \in \Omega(p)} W_s(||p - q||) I(q)$
  - $\Omega(p)$ neighborhood of p
  - W_s position similarity between $\boldsymbol{p}$ and $\boldsymbol{q}$, Gaussian function with standard deviations
  - K_p is the normalization term, the summation of weights

Bilateral mesh denoising

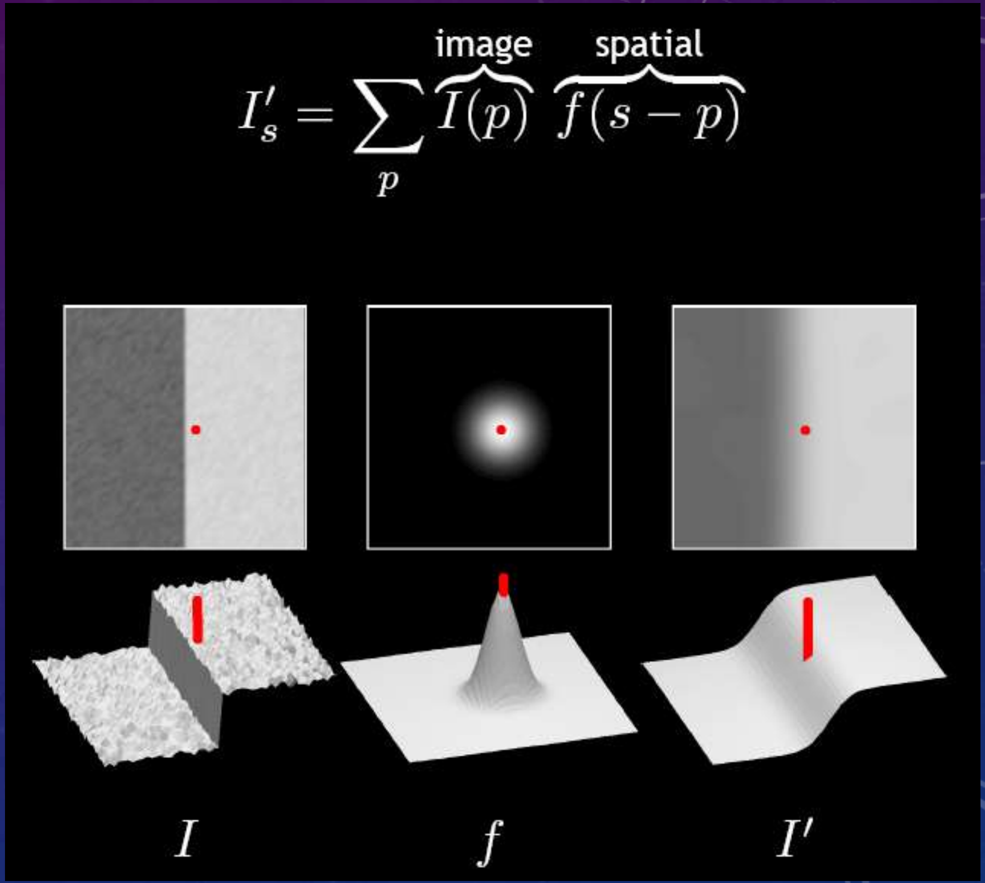
- Gaussian filter: $I(p) \longleftarrow \frac{1}{K_p} \sum_{q \in \Omega(p)} W_s(||p-q||) I(q)$



Bilateral mesh denoising

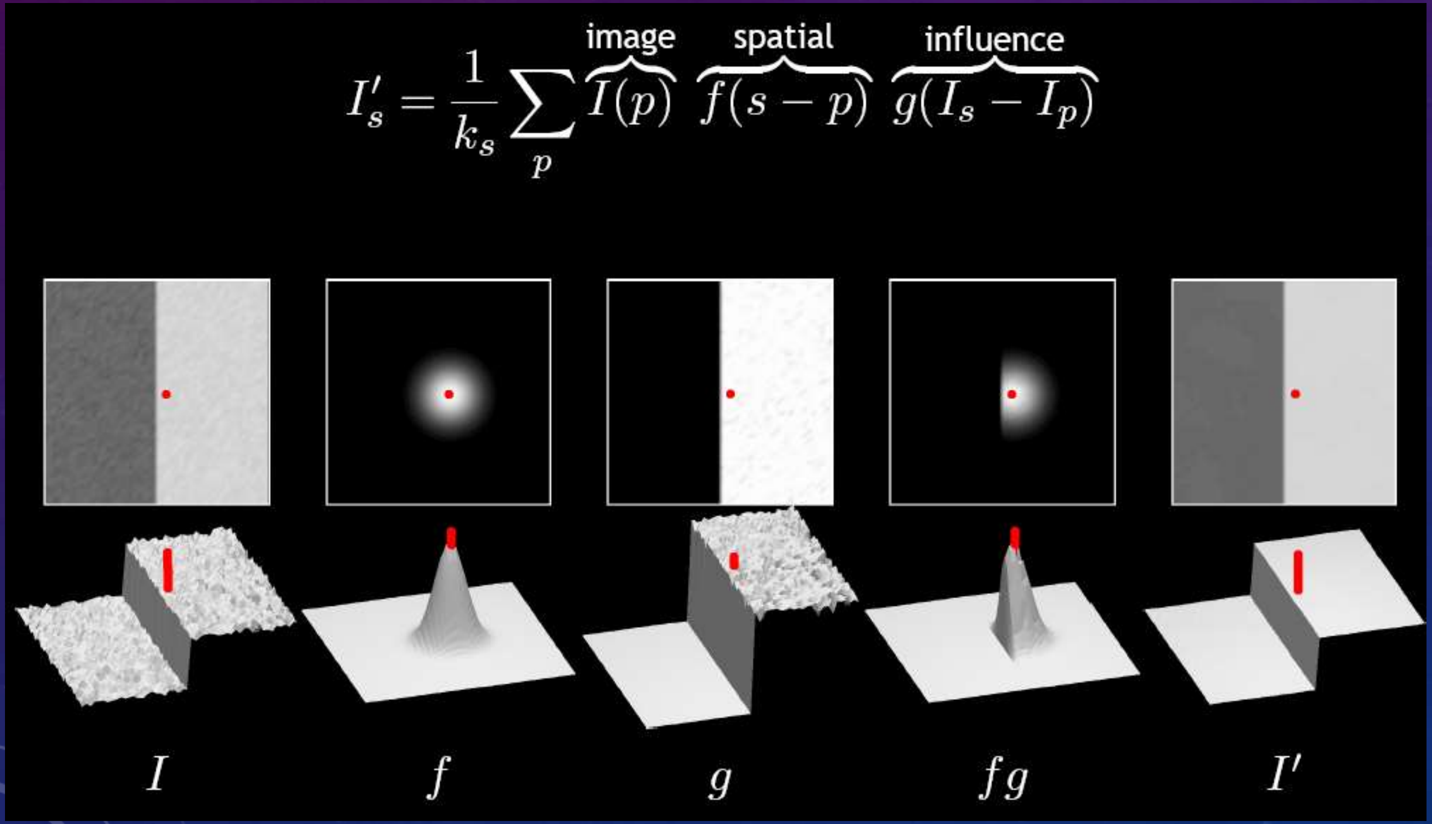
- Gaussian filter:

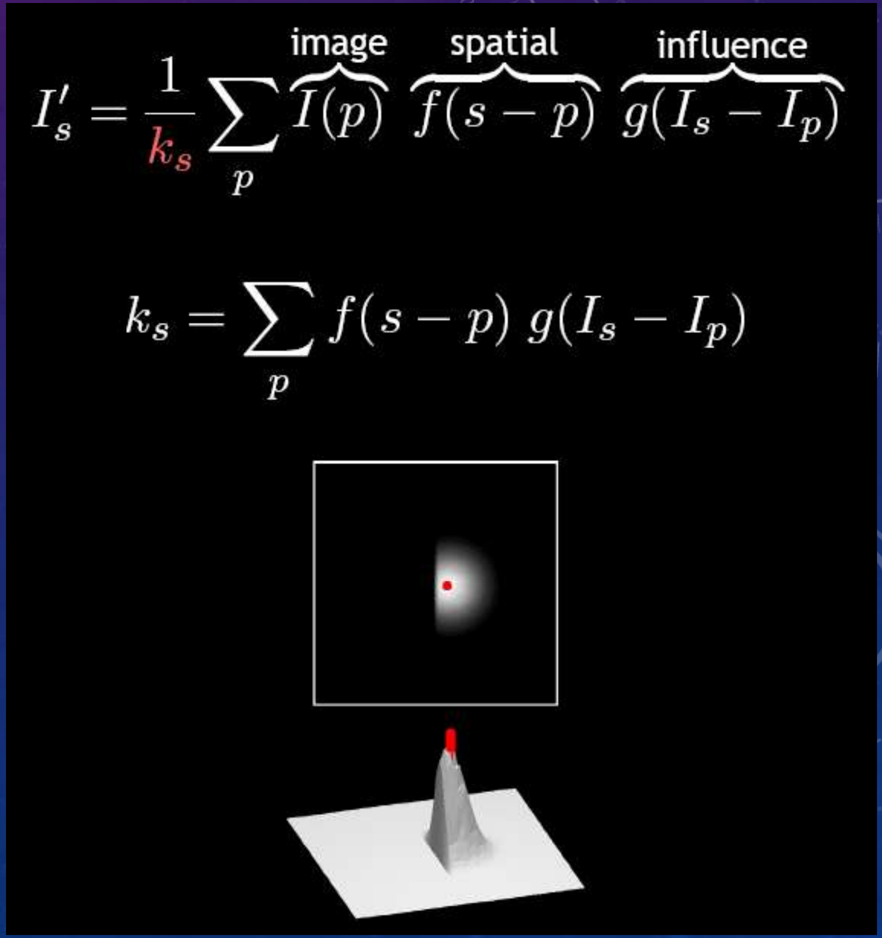
$$I(p) \leftarrow \frac{1}{K_p} \sum\nolimits_{q \in \Omega(p)} W_s(||p-q||) I(q)$$



$$I(p) \leftarrow \frac{1}{K_p} \sum\nolimits_{q \in \Omega(p)} W_s(||p-q||) W_r(||I(p)-I(q)||) I(q)$$ bilateral filter

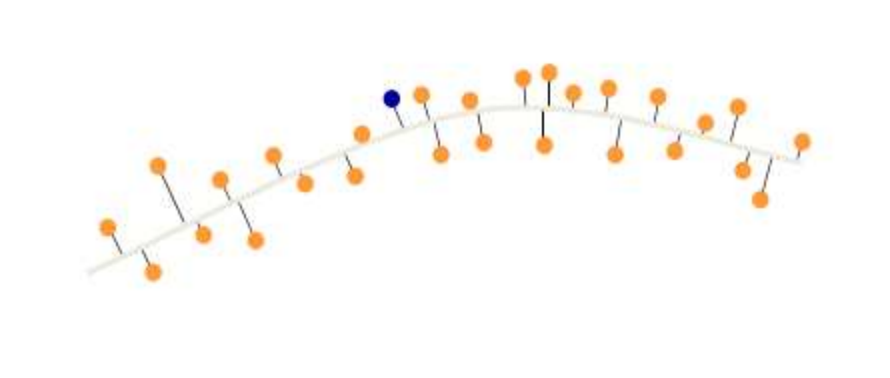
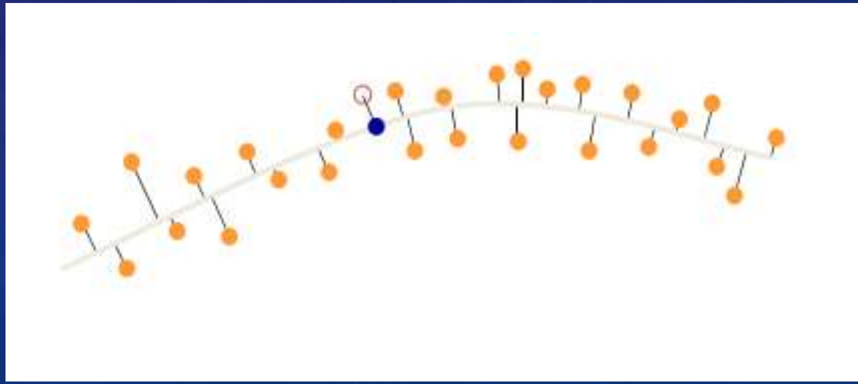
Bilateral mesh denoising





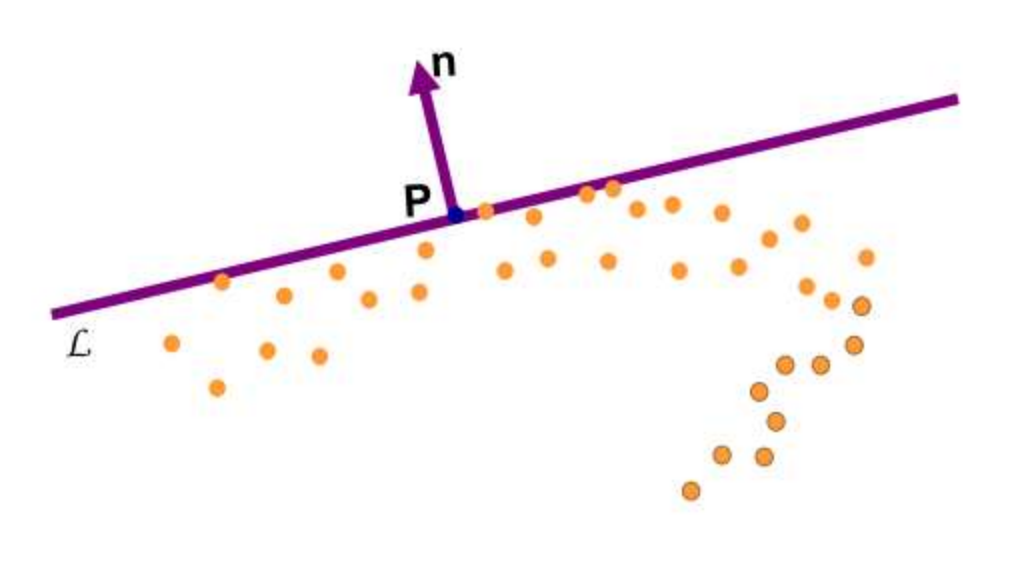
Bilateral filtering of meshes

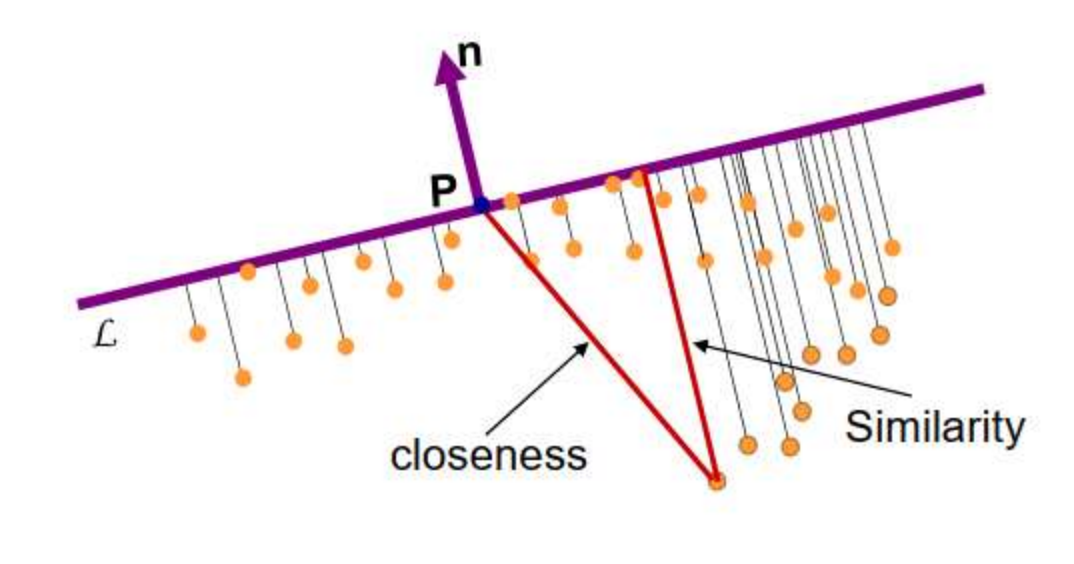
- Height above surface is equivalent to the gray level values in images
- Apply the bilateral filter to heights
- Move the vertex to its new height

How to represent noise-free surface

- A plane that passes through the point is the estimator to the smooth surface





How to represent noise-free surface

- Approximating plane : (1) a good approximation to surface, (2) preserve features
- For vertex p with normal n:

For $q \in \Omega(p)$:

$$d_q = \langle n, q - p \rangle$$

$$w_s = \exp(-\|q - p\|^2/(2\sigma_s))$$

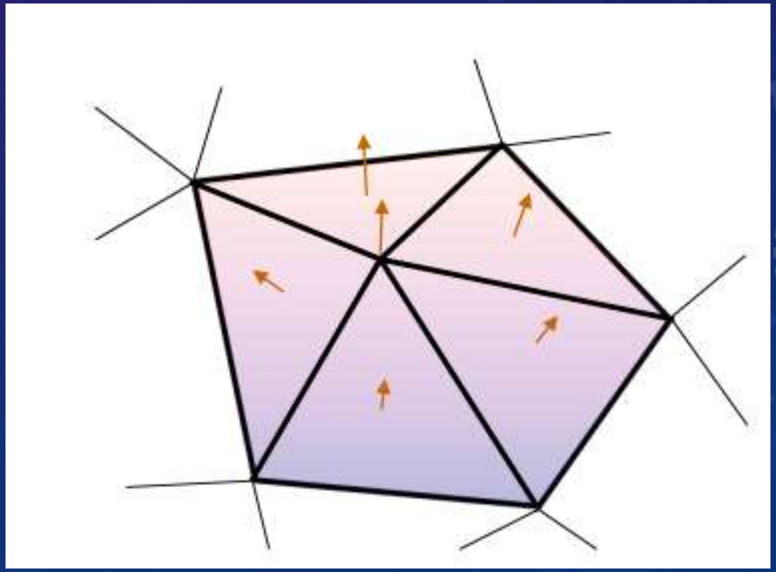
$$w_r = \exp(-d^2/(2\sigma_r))$$

$$d \mathrel{+}= w_s w_r d_q$$

$$w \mathrel{+}= w_s w_r$$

End

$$p = p + \frac{d}{w} \cdot n$$



Detail

- Normal: weighted average of the normal
  - 1-ring neighborhood of the vertex.
  - k-ring neighborhood for extremely noisy data
- Mesh shrinkage: volume preservation technique
  - Computing the volume
  - Sclae to preserve volume

Bilateral normal filtering

- 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

- $n_T \longleftarrow \frac{1}{K_p} \sum_{T' \in \Omega(T)} A_{T'} W_s(||c_{T'} - c_T||) W_r(||n_{T'} - n_T||) n_T$
  - n_T the normal of face T
  - c_T the center of face T
  - $\Omega(T)$ the neighbor of face T
  - A_T the area of face T

Bilateral normal filtering

- 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

Normal updating

- Local and iterative scheme:

$$n_T \longleftarrow \frac{1}{K_p} \sum_{T' \in \Omega(T)} A_{T'} W_s W_r n_T$$

1. Normalize the new normal after each iteration
2. Multiple iterations: increase the influence from a 1-ring neighborhood to a wider region, leading to a smoother mesh.

- Global and non-iterative scheme:

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

$$E_s = \sum_T A_T \|(Ln)_T\|_2^2$$

$$E_a = \sum_T A_T \left\| n_T - n_T^0 \right\|_2^2$$

Vertex updating

$$\begin{cases} \langle n_T, x_j - x_i \rangle = 0 \\ \langle n_T, x_k - x_j \rangle = 0 \\ \langle n_T, x_i - x_k \rangle = 0 \end{cases} \Longrightarrow E = \sum_T \sum_{ij \in T} \langle n_T, x_j - x_i \rangle^2$$

1. Solving linear system.
2. Gauss–Seidel iteration (fix other vertex, update one vertex)

$$x_i \leftarrow x_i + \frac{1}{N_i} \sum\nolimits_{T \in \Omega(i)} \langle n_T, c_T - x_i \rangle n_T$$

a) No need to determine a suitable step size.
b) Not computationally expensive. No need to solve a linear system.

Results

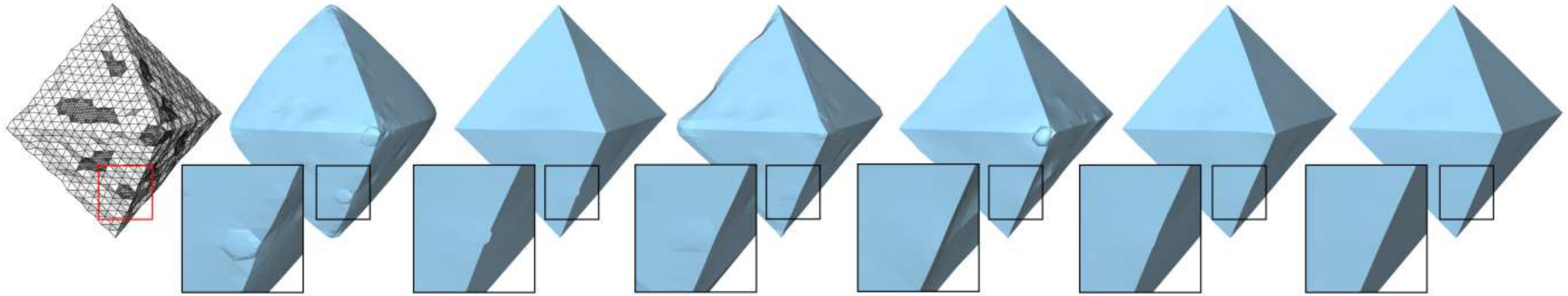


Fig. 1: Our mesh denoising schemes based on bilateral normal filtering produce better results than the state-of-the-art methods at challenging regions with sharp features or irregular surface sampling. From left to right: an input CAD-like model with random subdivision, denoising results with bilateral 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.

Spectral filters

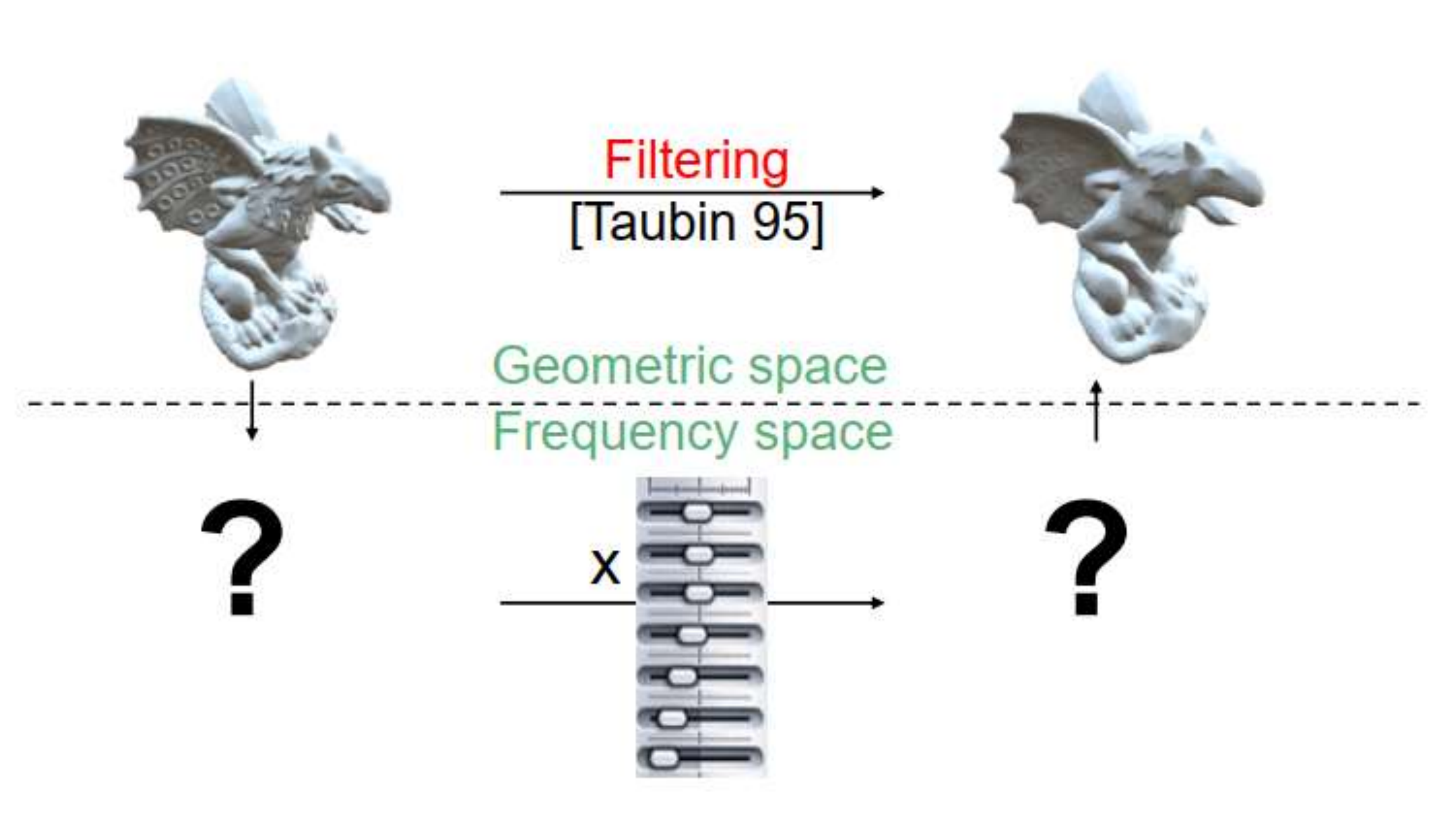
- 1D Fourier Transform:

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

$$f(x) = \int_{-\infty}^{\infty} F(w) e^{2\pi i w x} dw$$

Spatial domain $f(x) \Longleftrightarrow$ frequency domain $F(w)$

Spectral filters



Spectral filters

- 2-manifold surface:

Sine and cosine functions $\Leftrightarrow$ eigenfunctions of the Laplace operator

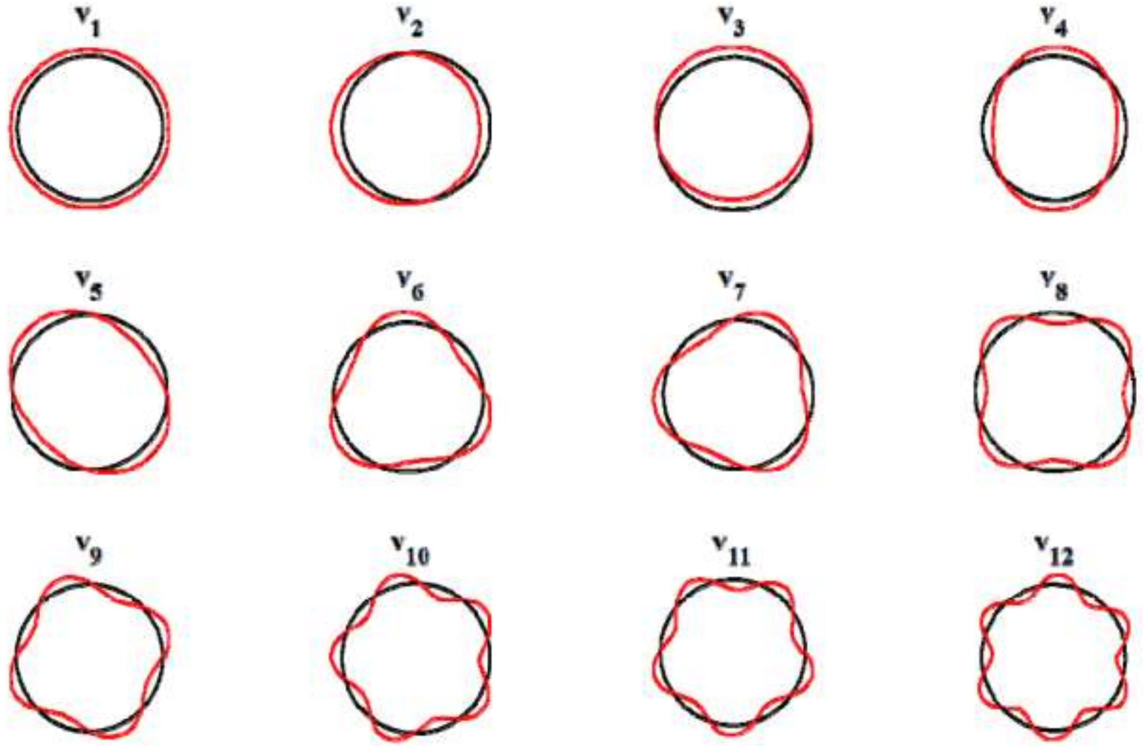
$$\Delta e_w(x) = -(2\pi w)^2 e_w(x), e_w(x) = e^{2\pi i w x}$$

Definition of eigenfunctions of the Laplace operator

$$\Leftrightarrow L e_i = \lambda_i e_i$$

Spectral filters

$$\mathbf{L} = \mathbf{V}\mathbf{D}\mathbf{V}^T \qquad \mathbf{V} = \begin{pmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \\ | & | & & | \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} k_1 & & & \\ & k_2 & & \\ & & \dots & \\ & & & k_n \end{pmatrix}$$



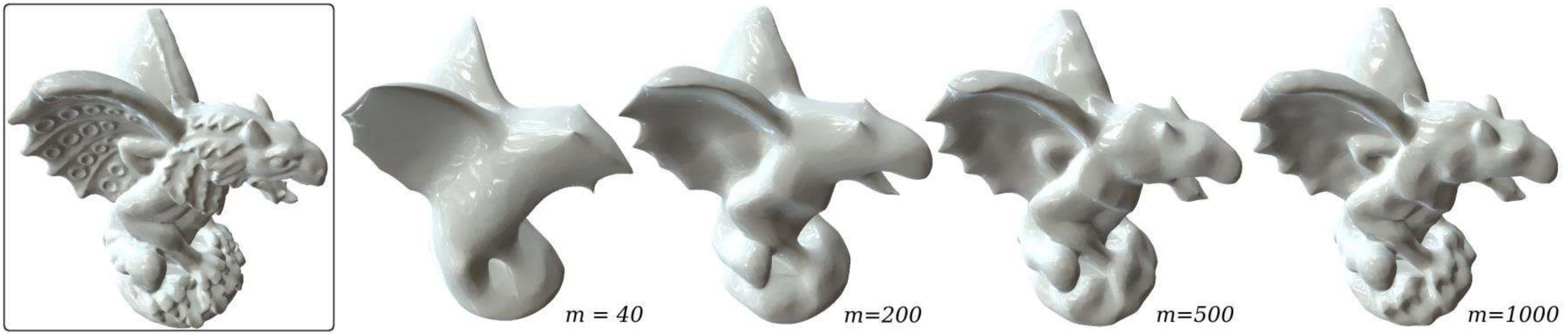
Spectral filters

$$\mathbf{L} = \mathbf{V}\mathbf{D}\mathbf{V}^T$$

$$\mathbf{V} = \begin{pmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \\ | & | & & | \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} k_1 & & & \\ & k_2 & & \\ & & \cdots & \\ & & & k_n \end{pmatrix}$$

Low-pass filter

- $f = \sum_{i=1}^{m} \langle f, e_i \rangle e_i \, , m < n$
- Replace f with vertex coordinates



Other filters

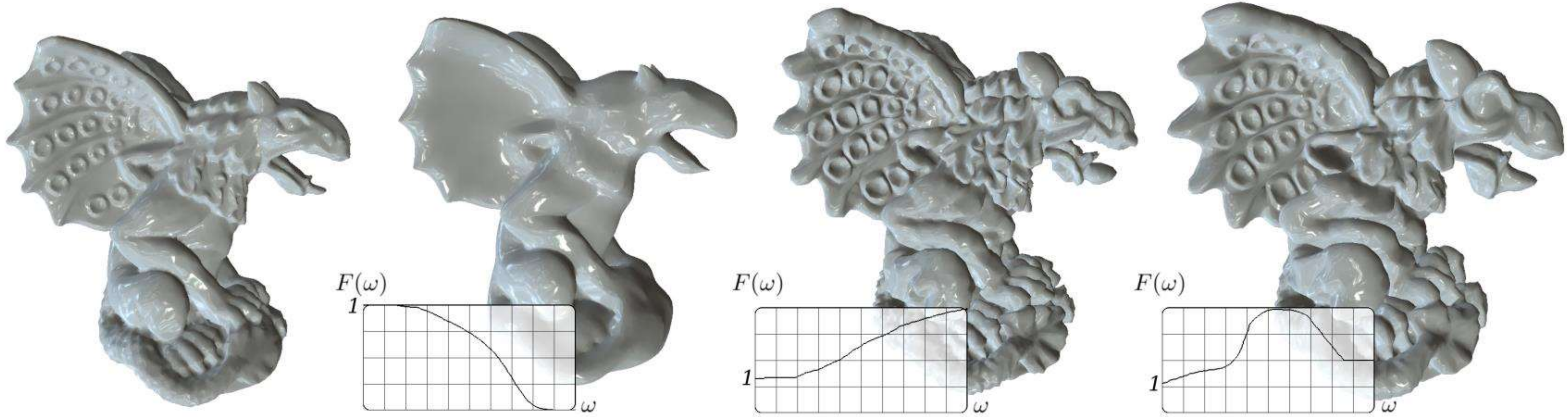


Figure 5: *Low-pass, enhancement and band-exaggeration filters. The filter can be changed by the user, the surface is updated interactively.*

Discussion

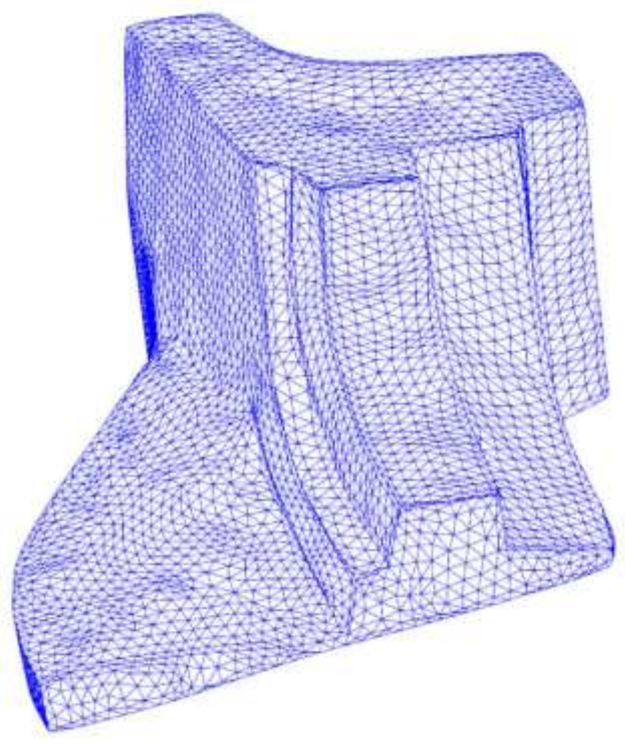
- Computationally expensive
  - Paper: Fast Approximation of Laplace-Beltrami Eigenproblems, SGP 2018
- A very useful representation of triangle mesh
  - 3D printing
    - Reduced-Order Shape Optimization Using Offset Surfaces, SIGGRAPH 2015
    - Non-Linear Shape Optimization Using Local Subspace Projections, SIGGRAPH 2016
  - Face modeling – simplification and Laplacian coordinate for details

Outline

- Filter-based methods
- Optimization-based methods
  - L_0 smoothing
  - Total Variation
- Data-driven methods

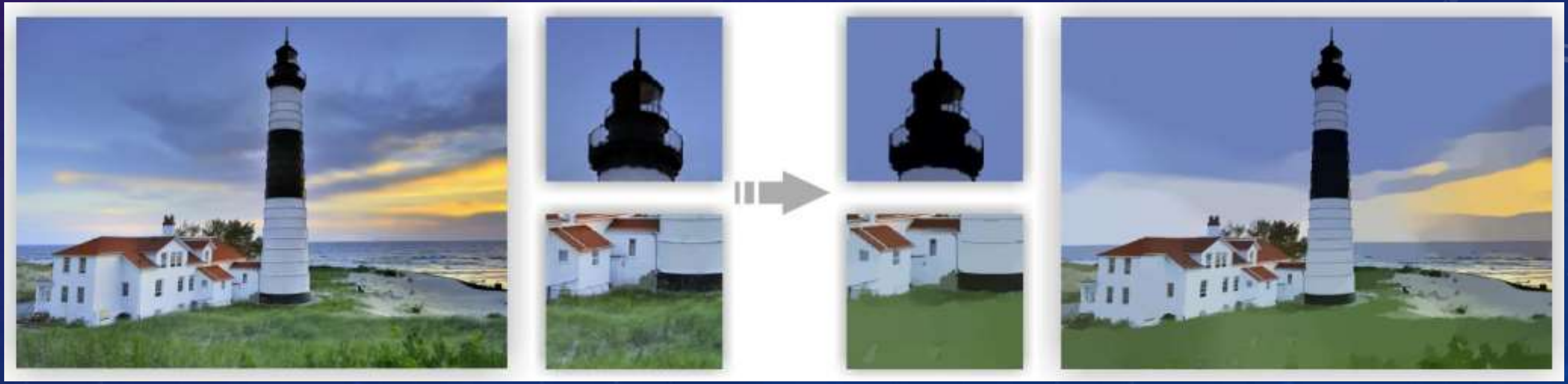
Prior

- The model consists of flat regions



L_0 Smoothing

- Paper: Mesh denoising via L_0 minimization
  - Maximizes flat regions and gradually removes noise while preserving sharp features.
- From image processing - Paper: Image smoothing via L_0 gradient minimization



L_0 minimization for images

- Energy: $|c - c^*|^2 + \lambda|\nabla c|_0$
  - c: a vector of pixel colors
  - c^*: original image colors
  - ∇c : a vector of gradients of these colors
  - $|\nabla c|_0$: L_0 norm of ∇c

Optimization method

- Auxiliary variables δ:

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

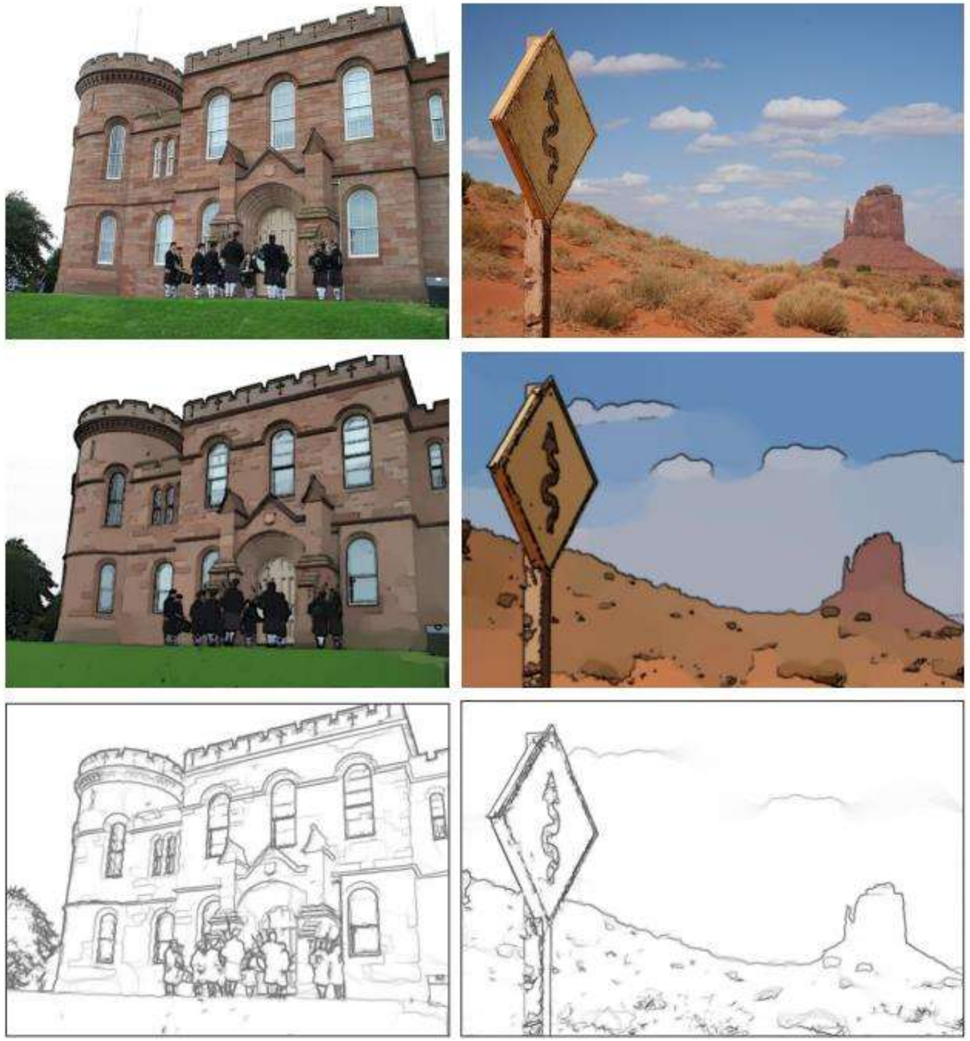
- Alternating optimization:
  - Fix c, solve δ – subproblem: $\min_{\delta} \beta|\nabla c - \delta|^2 + \lambda|\delta|_0$ **Analytic solution**
  - Fix δ, solve c – subproblem: $\min_{c} |c - c^*|^2 + \beta|\nabla c - \delta|^2$ **Quadratic**

δ – subproblem

- If $\delta = 0, \beta|\nabla c - \delta|^2 + \lambda|\delta|_0 = \beta|\nabla c|^2$
- If $\delta \neq 0, \beta|\nabla c - \delta|^2 + \lambda|\delta|_0 \geq \lambda$

$$\beta|\nabla c - \delta|^2 + \lambda|\delta|_0 \geq \min(\beta|\nabla c|^2, \lambda)$$

$$\delta = \begin{cases} 0, & \beta|\nabla c|^2 \leq \lambda \\ \nabla c, & \beta|\nabla c|^2 > \lambda \end{cases}$$



(a) (b)

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

Mesh denoising

- c → vertex coordinate p
- ∇c → discrete differential operator
- Define on edges

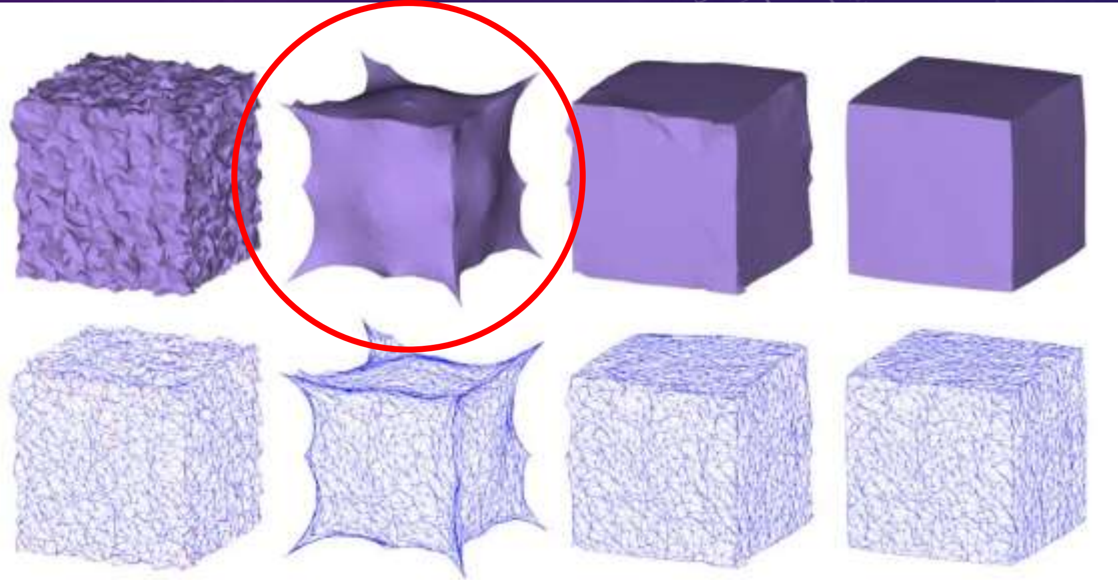
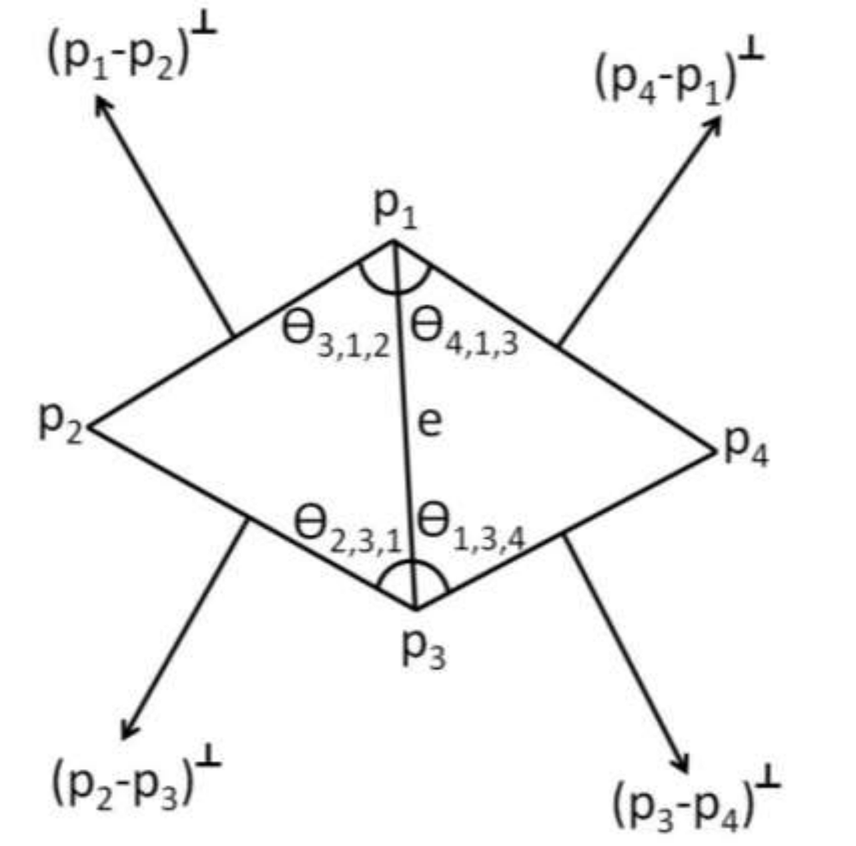


Figure 2: *From left to right: noisy input surface with* $\sigma = 0.3l_e$*, vertex-based cotangent operator, our cotangent edge operator, our area-based edge operator. The bottom row shows wireframes with flipped triangles denoted by red edges. None of these results use regularization.*

Differential edge operator

- Flat $\Leftrightarrow |\nabla c| = 0$
- $\nabla_{p_2} A_{p_1 p_2 p_3} + \nabla_{p_4} A_{p_1 p_3 p_4}$

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



Problem of cotan weights

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

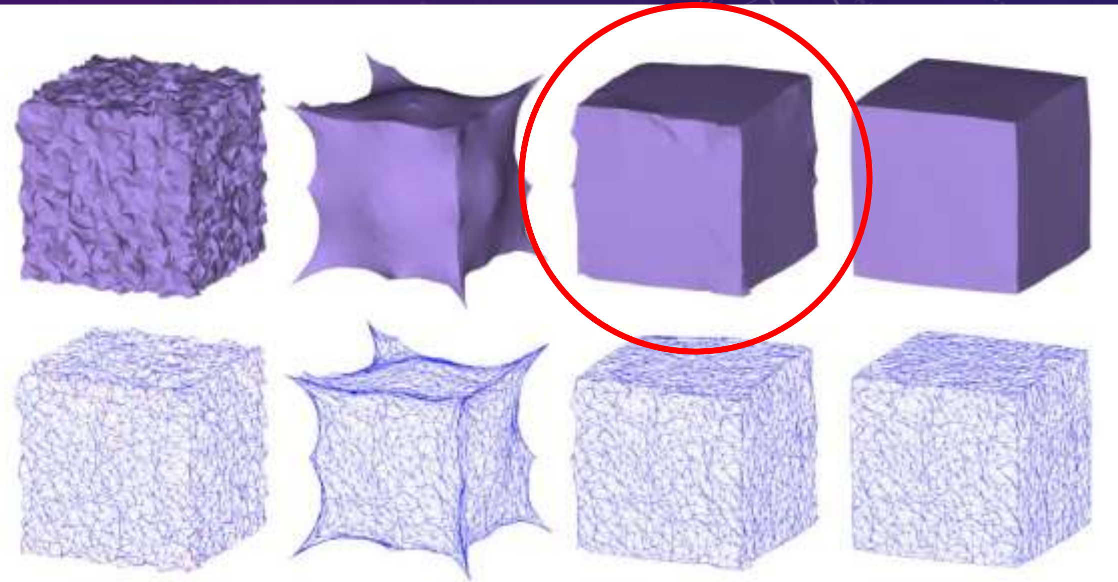


Figure 2: *From left to right: noisy input surface with $\sigma = 0.3l_e$, vertex-based cotangent operator, our cotangent edge operator, our area-based edge operator. The bottom row shows wireframes with flipped triangles denoted by red edges. None of these results use regularization.*

Area-based edge operator

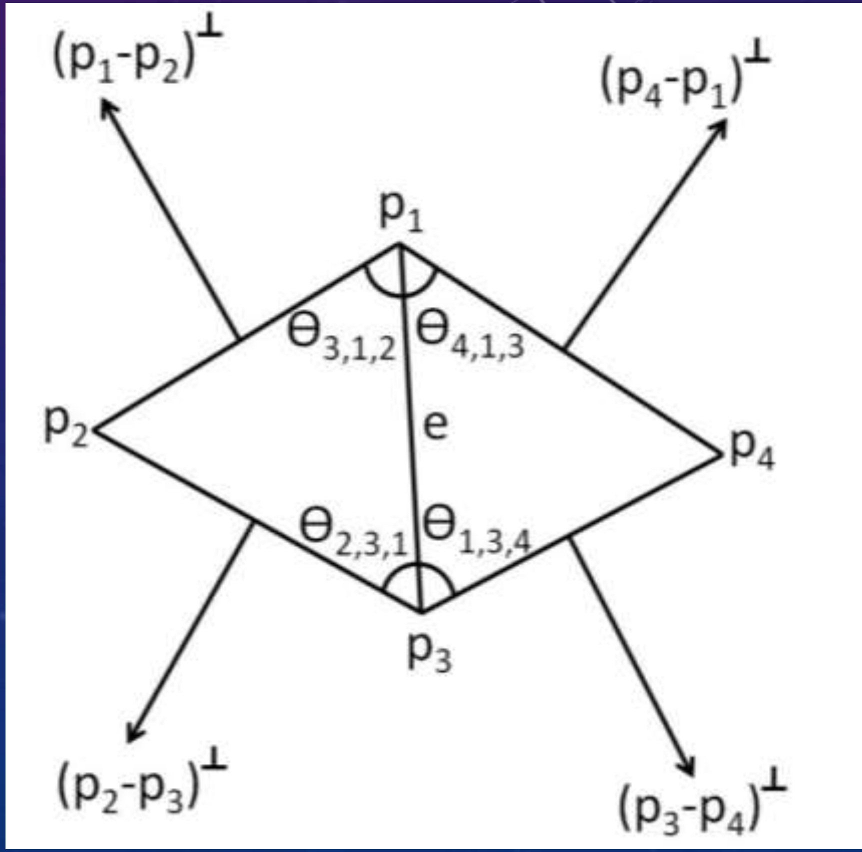
- $D(e) = \begin{bmatrix} -\cot\theta_{2,3,1} - \cot\theta_{1,3,4} \\ \cot\theta_{2,3,1} + \cot\theta_{3,1,2} \\ -\cot\theta_{3,1,2} - \cot\theta_{4,1,3} \\ \cot\theta_{1,3,4} + \cot\theta_{4,1,3} \end{bmatrix}^T \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} \times \|p_1 - p_3\|_2^2$

  - Similarly: when p_j are planar:

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

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

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



Results

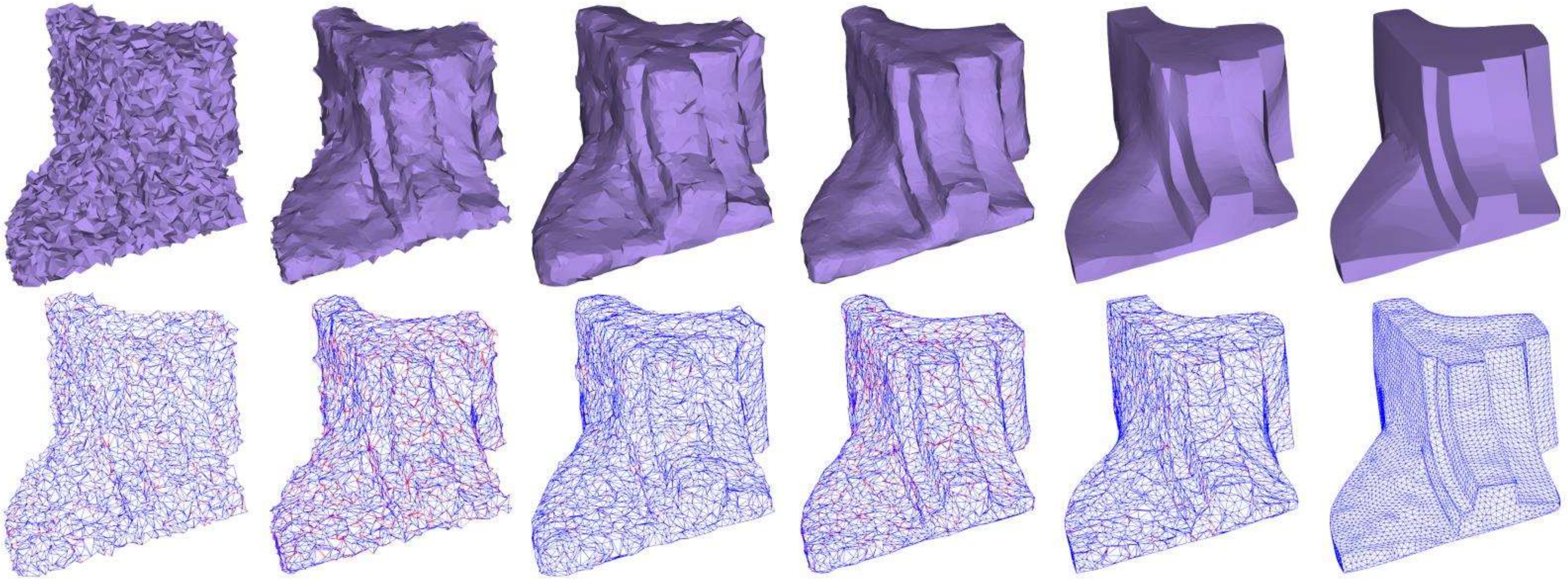
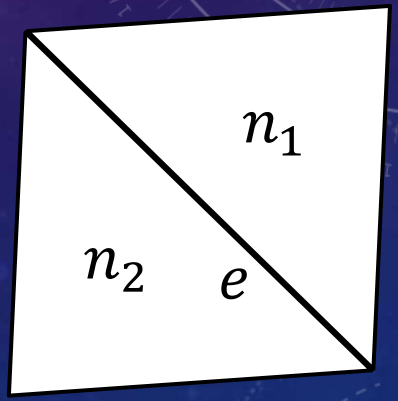


Figure 9: *From left to right: the input mesh with large noise in random directions, bilateral filtering [Fleishman et al. 2003], prescribed mean curvature flow [Hildebrandt and Polthier 2004], mean filtering [Yagou et al. 2002], bilateral normal filtering [Zheng et al. 2011], our result. We show the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

Total variation-based method

- 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



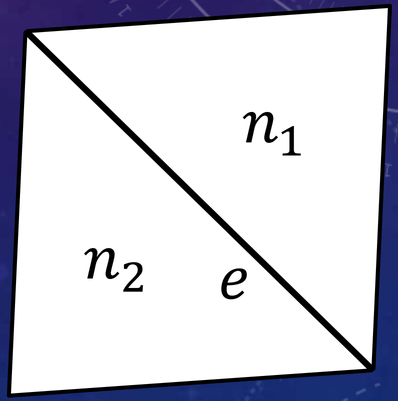
Total variation-based method

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

1. $E_{TV} = \sum_e w_e l_e \sqrt{\|\nabla n_e\|_2^2}$,

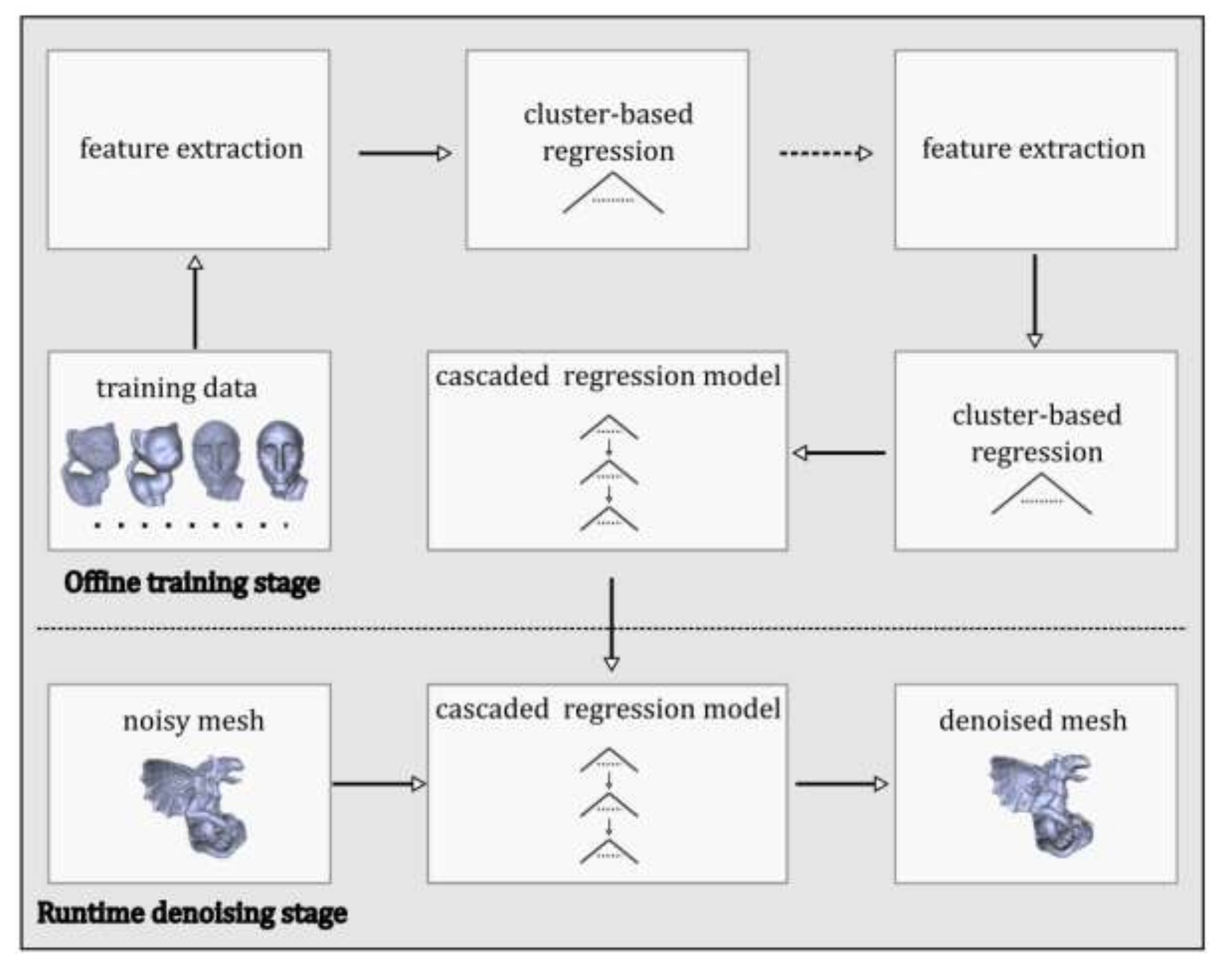
where $\nabla n_e = n_1 - n_2$, $\quad w_e = \exp(-\|n_1^* - n_2^*\|_2^4)$

2. $E_\alpha = \sum_f \left\|n_f - n_f^*\right\|_2^2$



Outline

- Filter-based methods
- Optimization-based methods
- Data-driven methods
  - Mesh Denoising via Cascaded Normal Regression



A highly nonlinear function

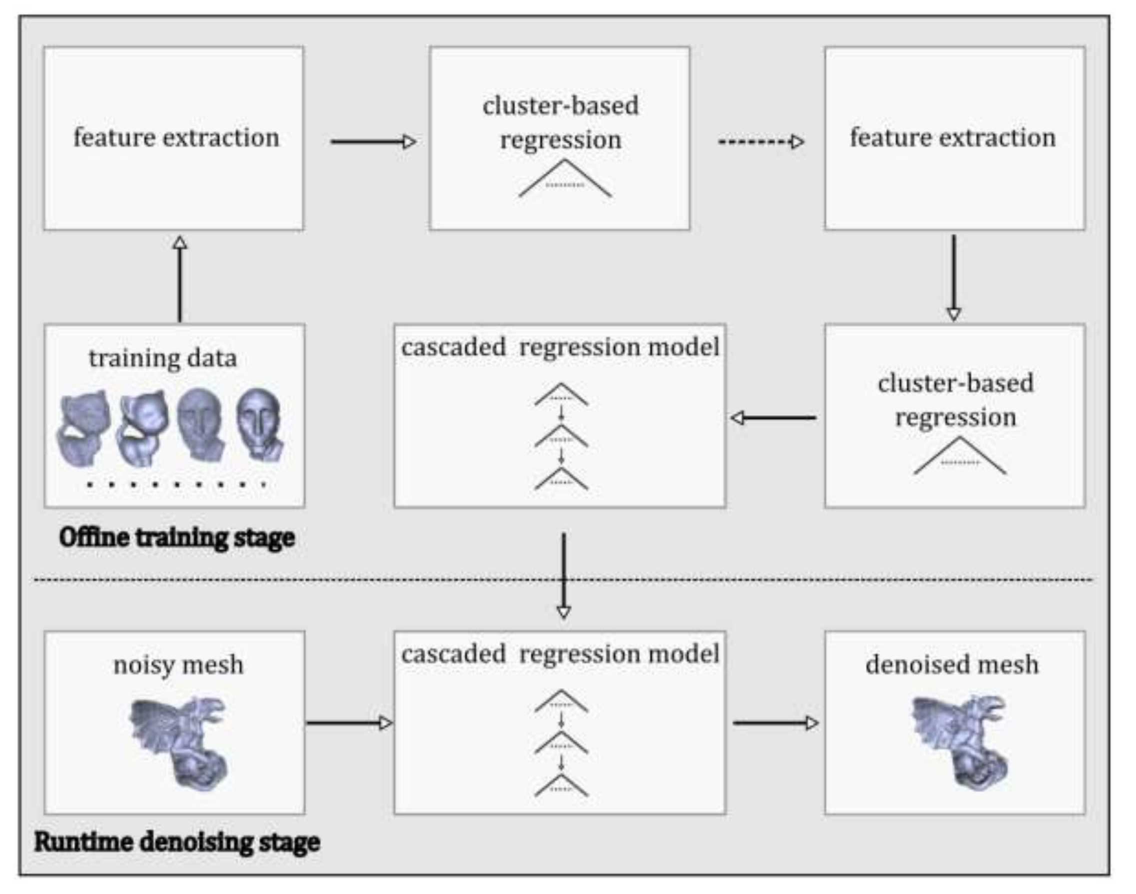
Mesh Denoising via Cascaded Normal Regression

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

$$n_f = \mathcal{F}(\Omega_f),\ \Omega_f\text{: local noisy region}$$

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

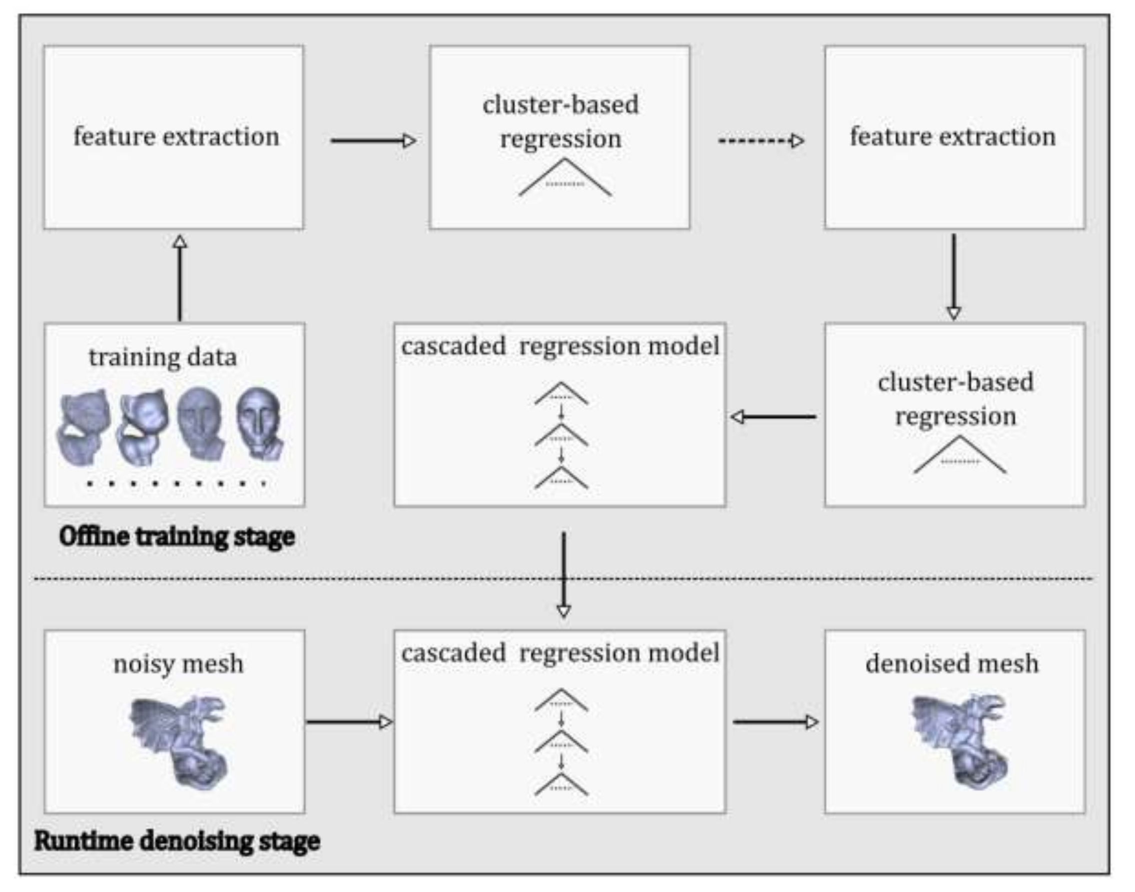
- A training pair: $(S_i, \bar{n}_i)$

S_i : filtered face normal descriptor (FND) of ith face

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

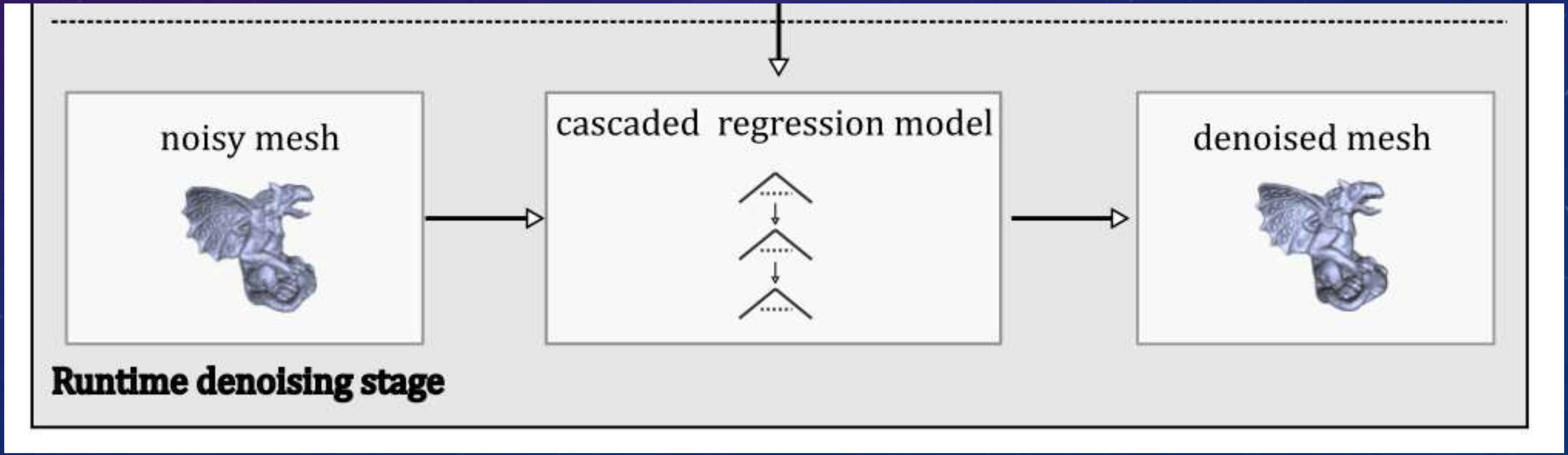
- Goal - learn the function:

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



Runtime denoising stage

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



Bilateral normal filtering

$$n_T^{(k+1)} \leftarrow \frac{1}{K_p} \sum\nolimits_{T' \in \Omega(T)} A_{T'} W_s(||c_{T'} - c_T||) W_r\left(\left\|n_{T'}^{(k)} - n_T^{(k)}\right\|\right) n_T^{(k)}$$

Parameters: σ_s, σ_r, iteration number M

Bilateral filtered face normal descriptor (B-FND)

$$S_T = (n_T^{(1)}(\sigma_{s_1}, \sigma_{r_1}), \dots, n_T^{(1)}(\sigma_{s_L}, \sigma_{r_L}), \dots, n_T^{(M)}(\sigma_{s_1}, \sigma_{r_1}), \dots, n_T^{(M)}(\sigma_{s_L}, \sigma_{r_L}))$$

Guided bilateral filter (joint bilateral filter)

$$n_T^{(k+1)} \leftarrow \frac{1}{K_p} \sum\nolimits_{T' \in \Omega(T)} A_{T'} W_s(\|c_{T'} - c_T\|) W_r\left(\left\|g(n_{T'}^{(k)}) - g(n_T^{(k)})\right\|\right) n_T^{(k)}$$

Gaussian normal filter: $g\left(n_T^{(k)}\right) = \frac{1}{K_p} \sum_{T' \in \Omega(T)} A_{T'} W_s(\|c_{T'} - c_T\|) n_T^{(k)}$

Guided filtered face normal descriptor (G-FND)

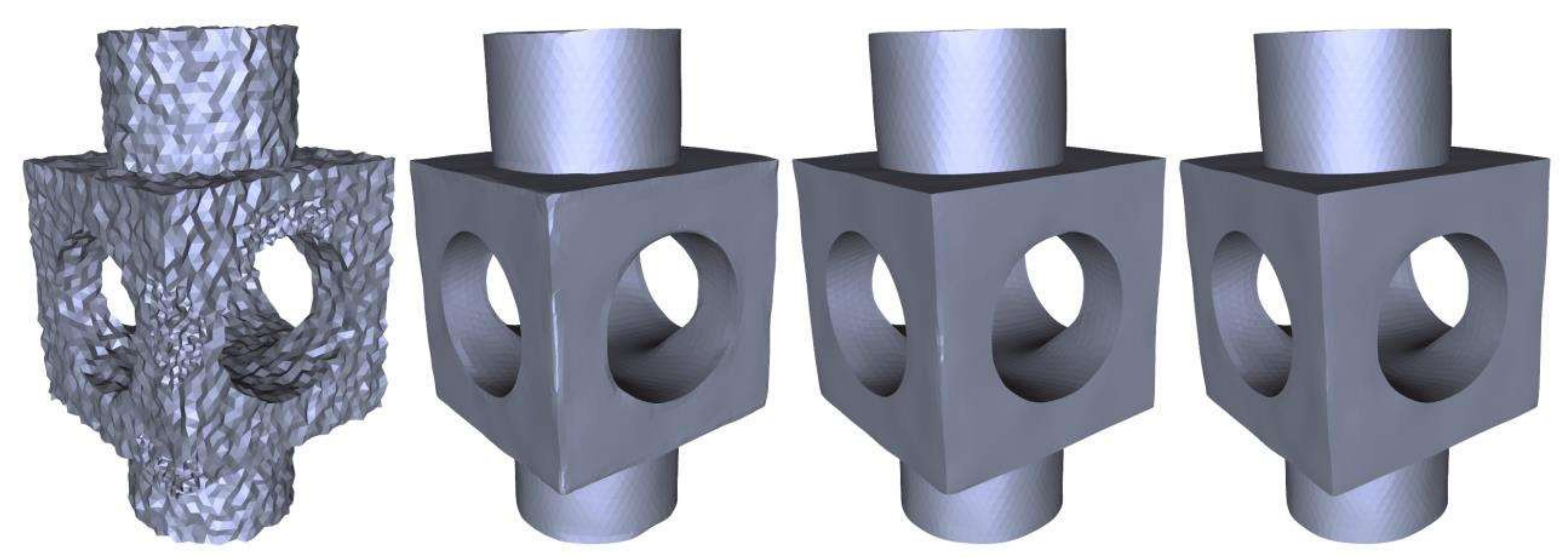
$$S_{g,T} = (n_{g,T}^{(1)}(\sigma_{s_1}, \sigma_{r_1}), \dots, n_{g,T}^{(1)}(\sigma_{s_L}, \sigma_{r_L}), \dots, n_{g,T}^{(M)}(\sigma_{s_1}, \sigma_{r_1}), \dots, n_{g,T}^{(M)}(\sigma_{s_L}, \sigma_{r_L}))$$

Training data

- A dataset: $D = \{S_i, \bar{n}_i\}_{i=1}^{N}, S_i = [S_T(i), S_{g,T}(i)]$
- 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}

Cascaded scheme

- G-FND in the first regression function



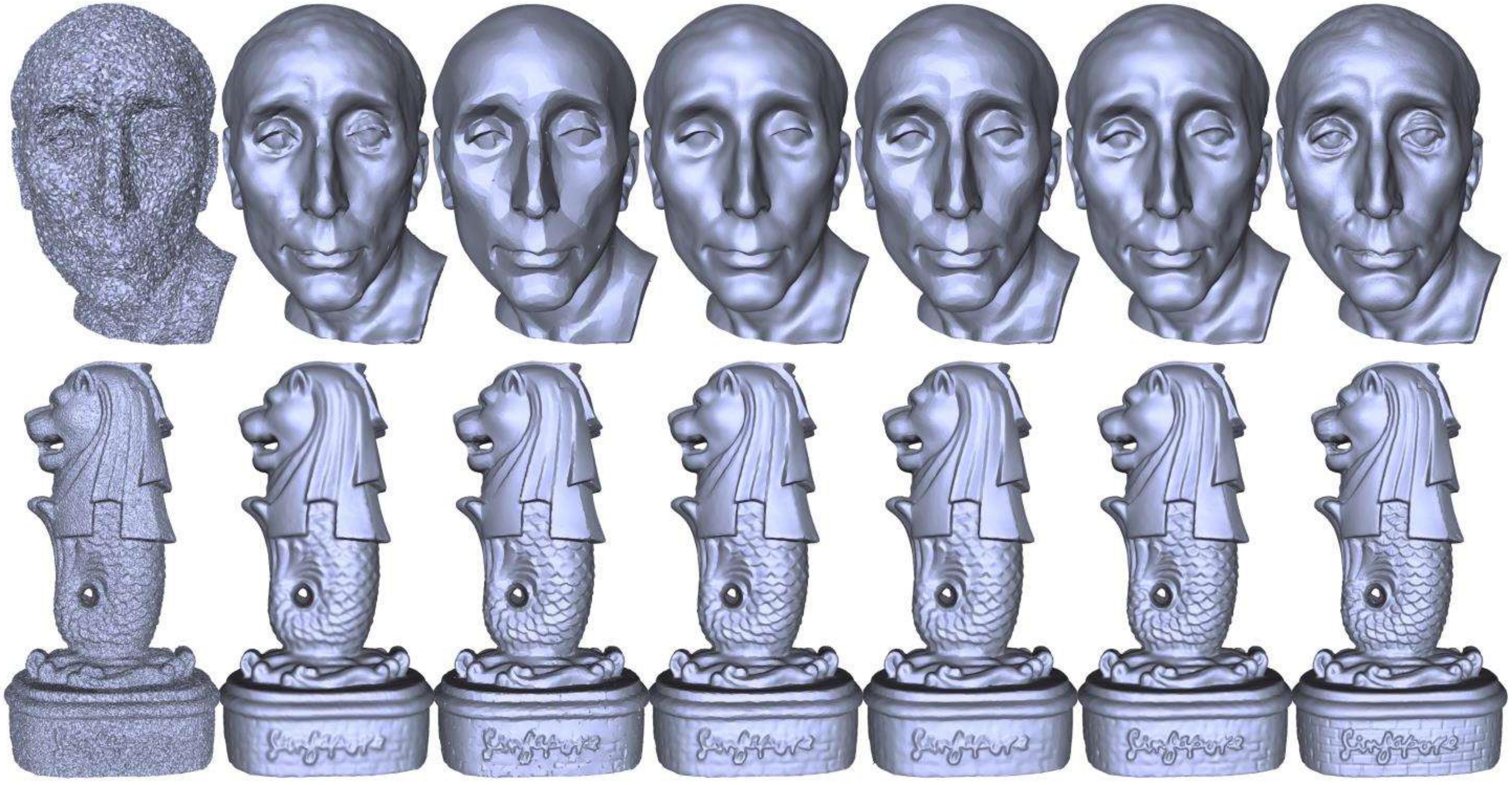
(a) Input (b) G-FND (c) B-FND (d) Both

Choice of hyperparameters

- σ_s: $\{l_e, 2l_e\}$, 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.

Results



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