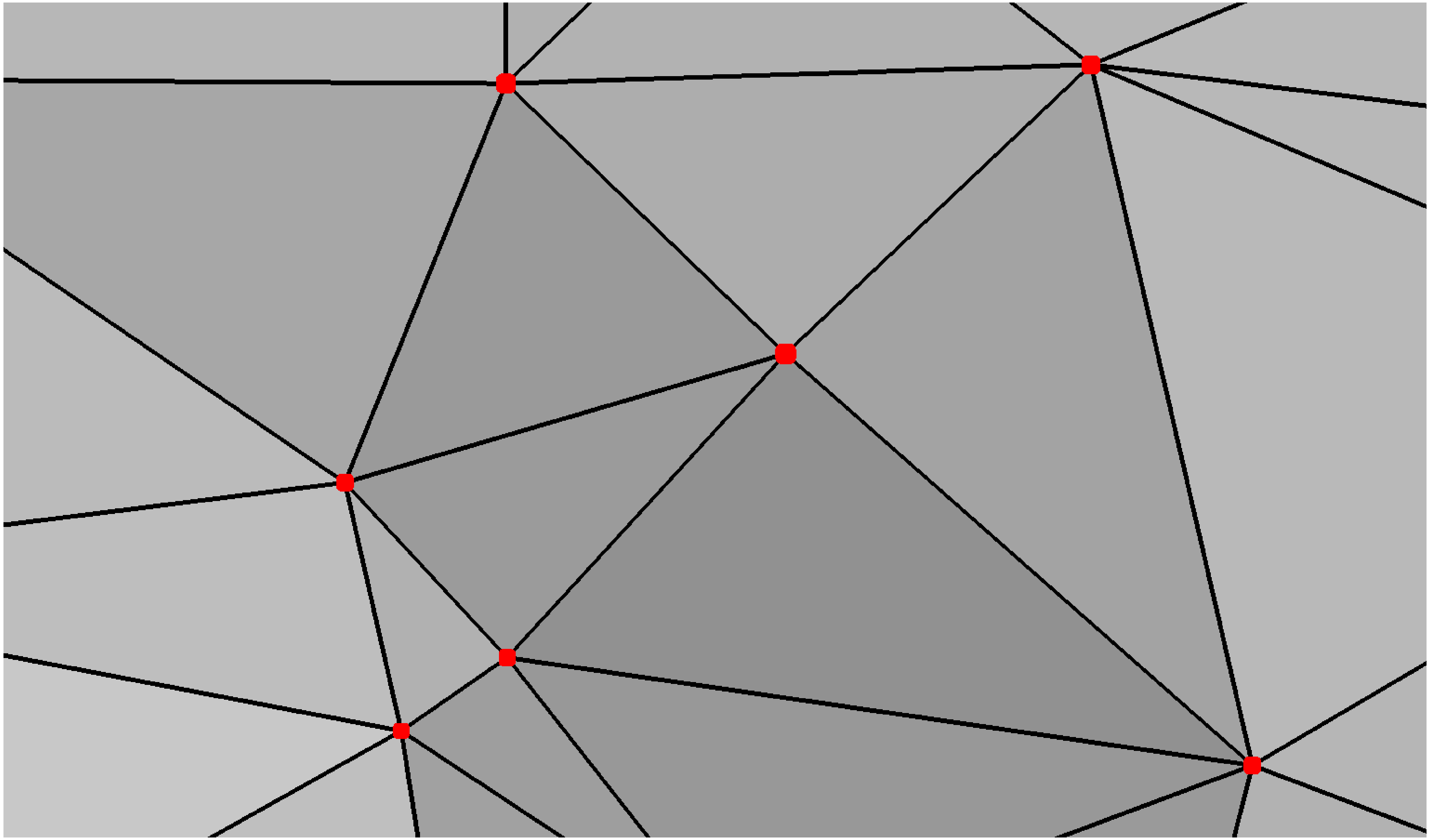
Surface Cleaning and Smoothing

Gianpaolo Palma

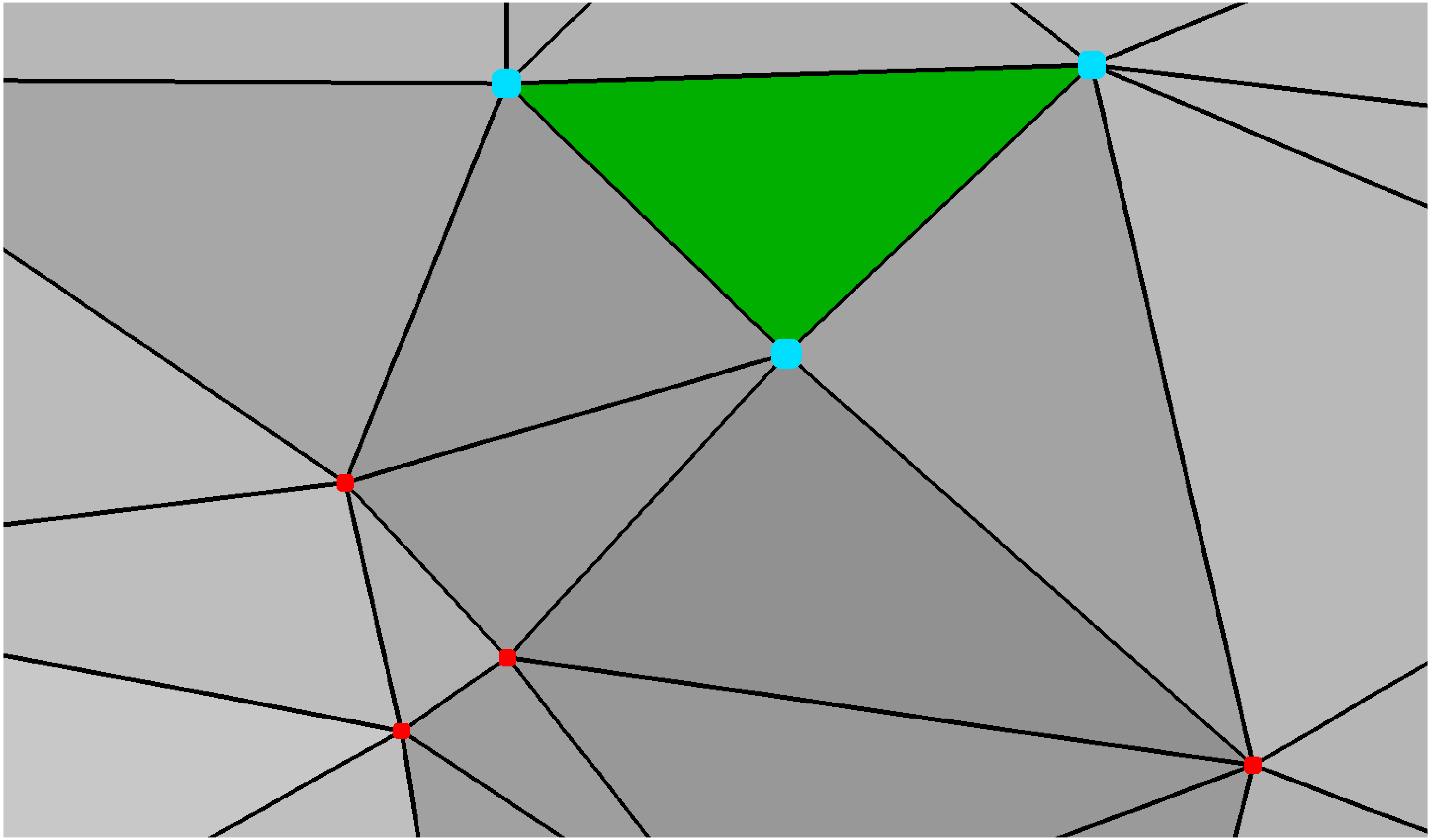
Triangle Mesh

List of vertices + List of triangle as triple of vertex references



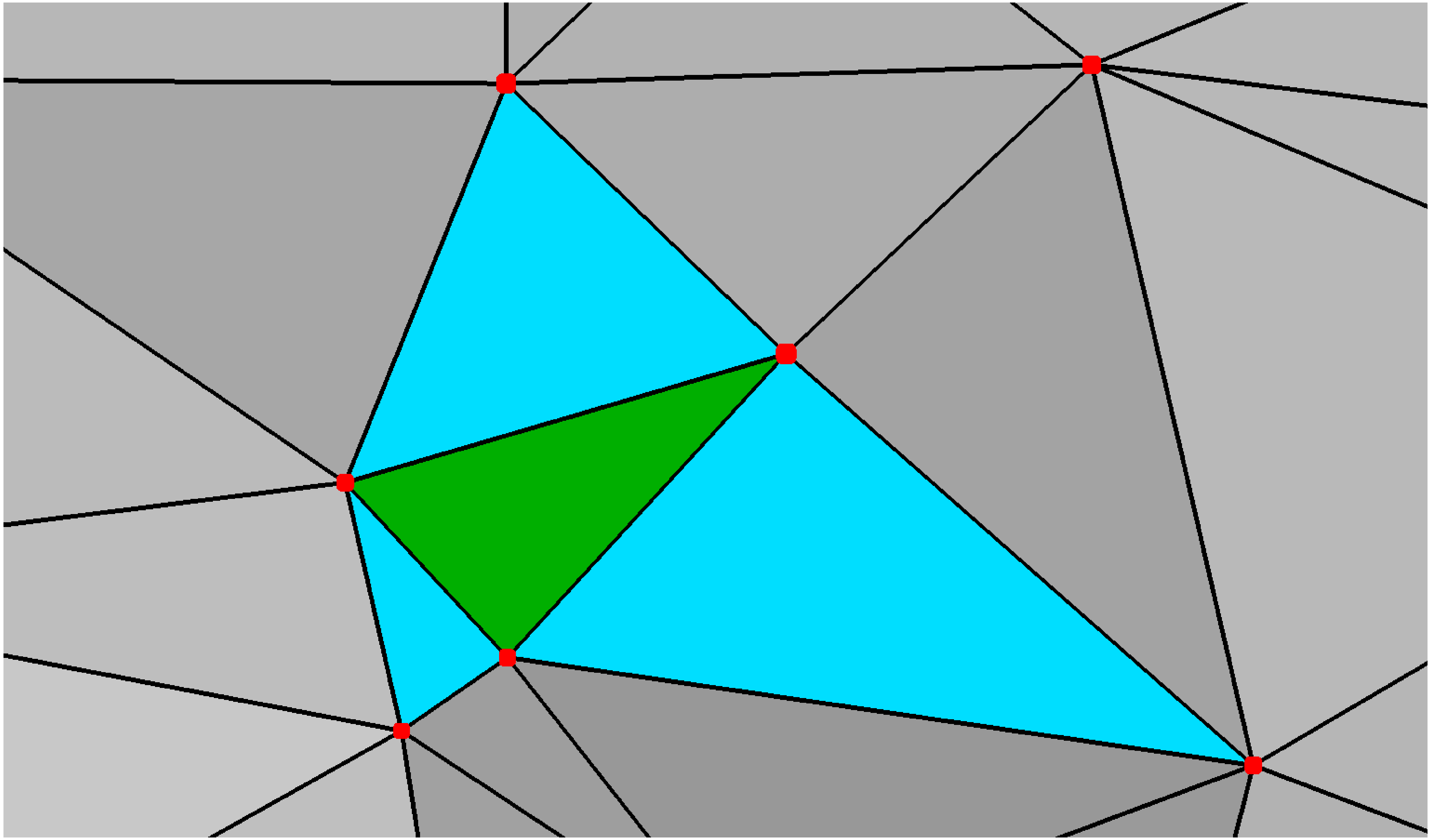
Mesh Adjacency Relation

FACE-VERTEX



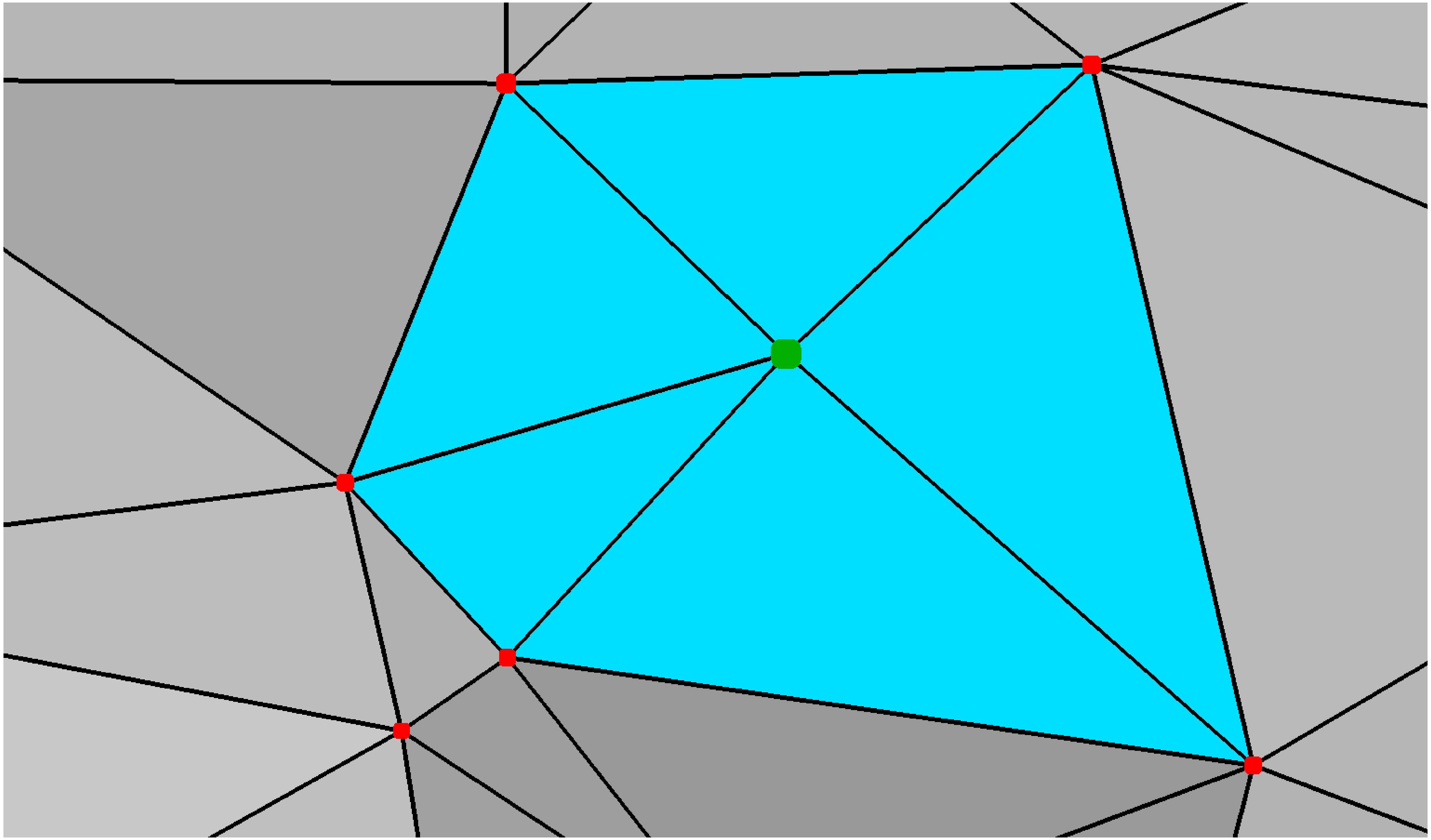
Mesh Adjacency Relation

FACE-FACE



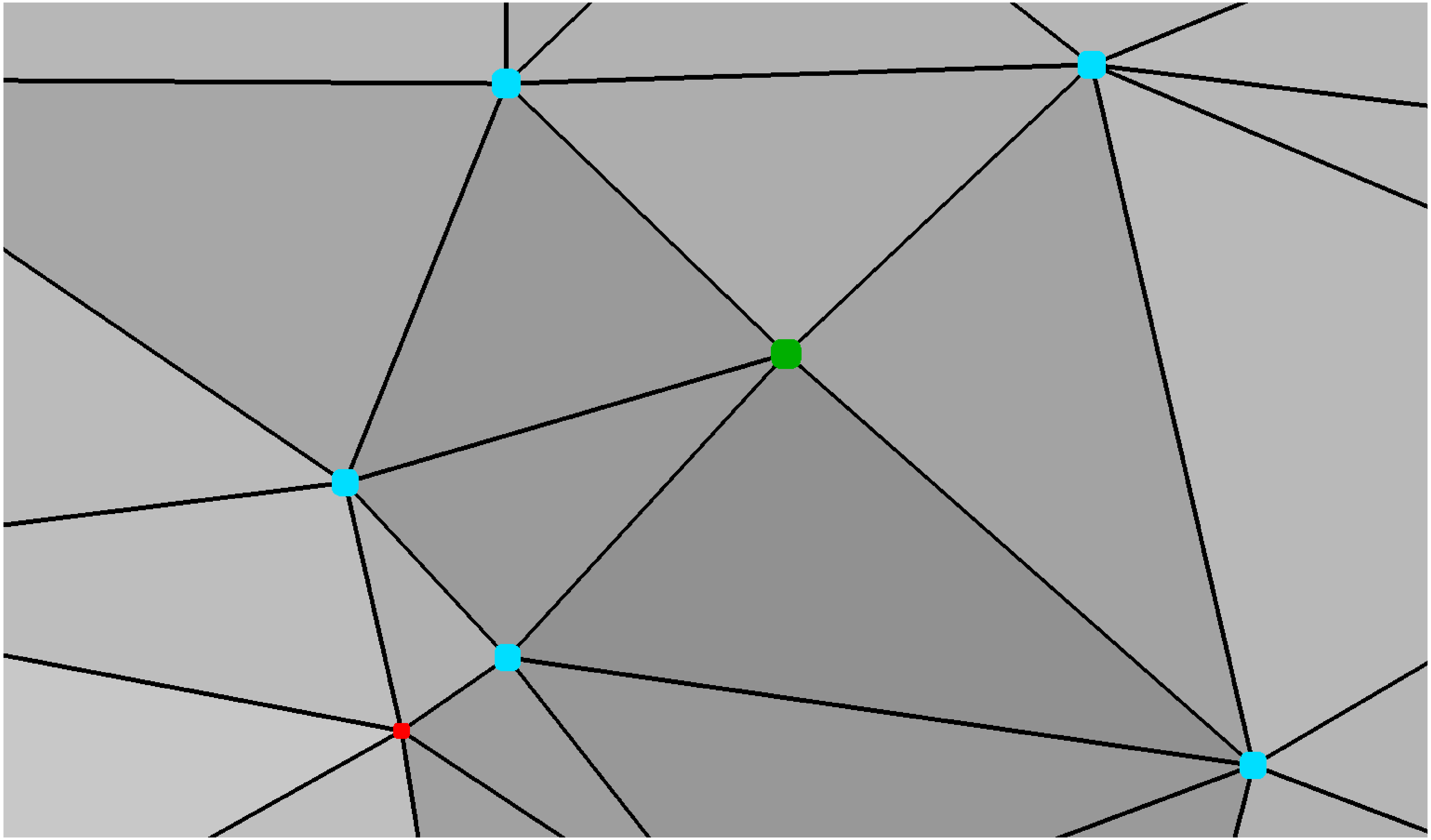
Mesh Adjacency Relation

VERTEX-FACE



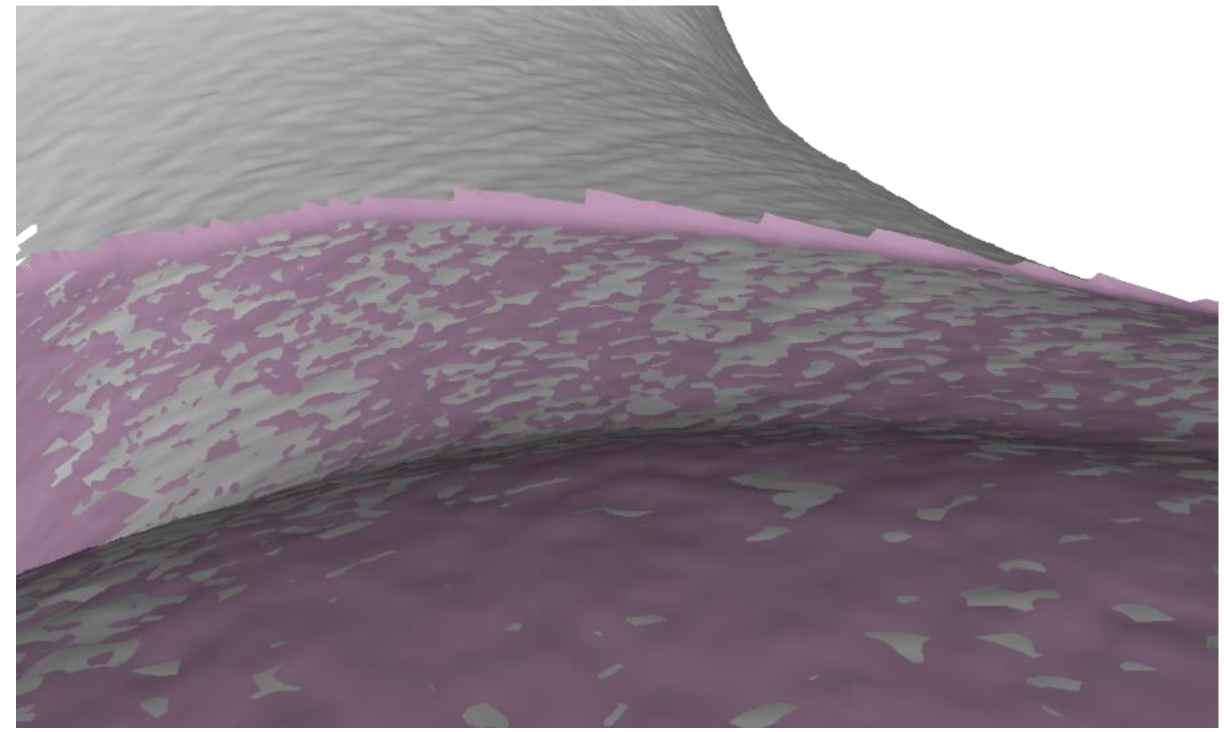
Mesh Adjacency Relation

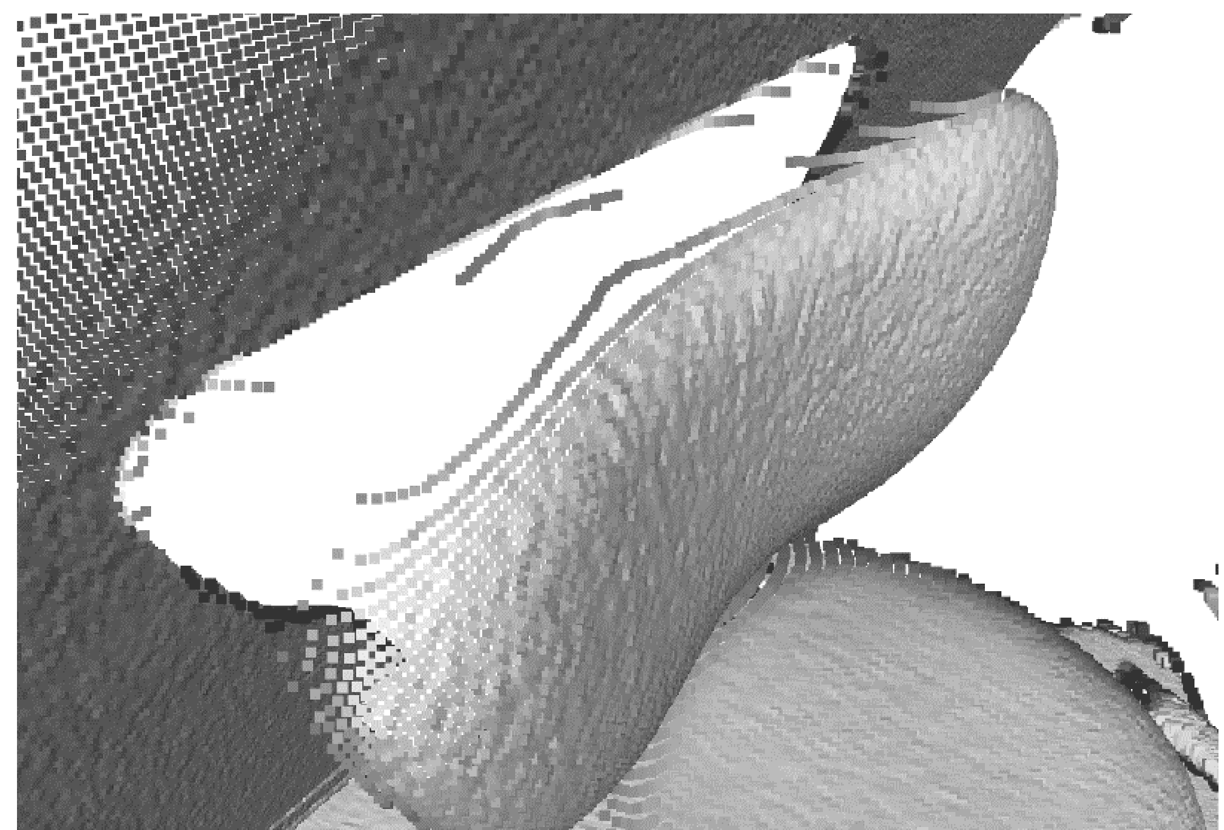
VERTEX-VERTEX



Surface Cleaning

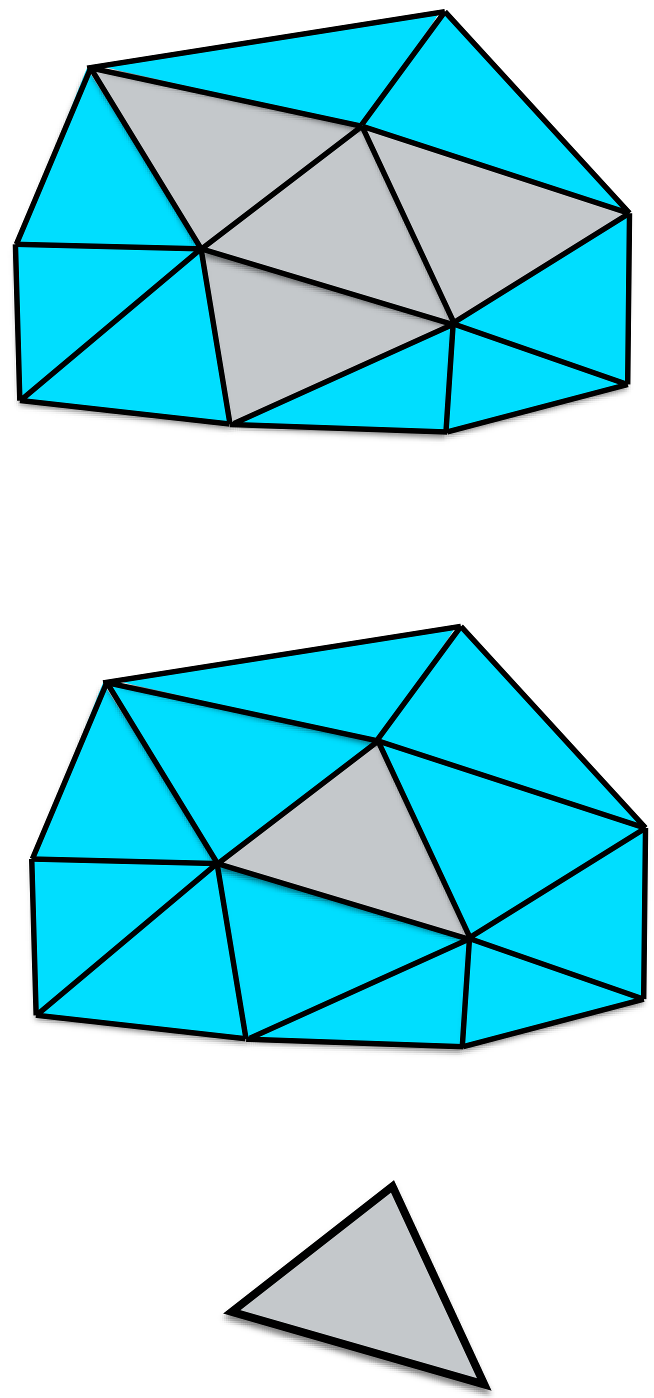
- Remove scanning artifact
  - Remove bad border triangle (triangle mesh)
  - Remove outliers (point cloud)
    Kriegel et al. "LoOP: Local Outliers Probability" CIKM 2009



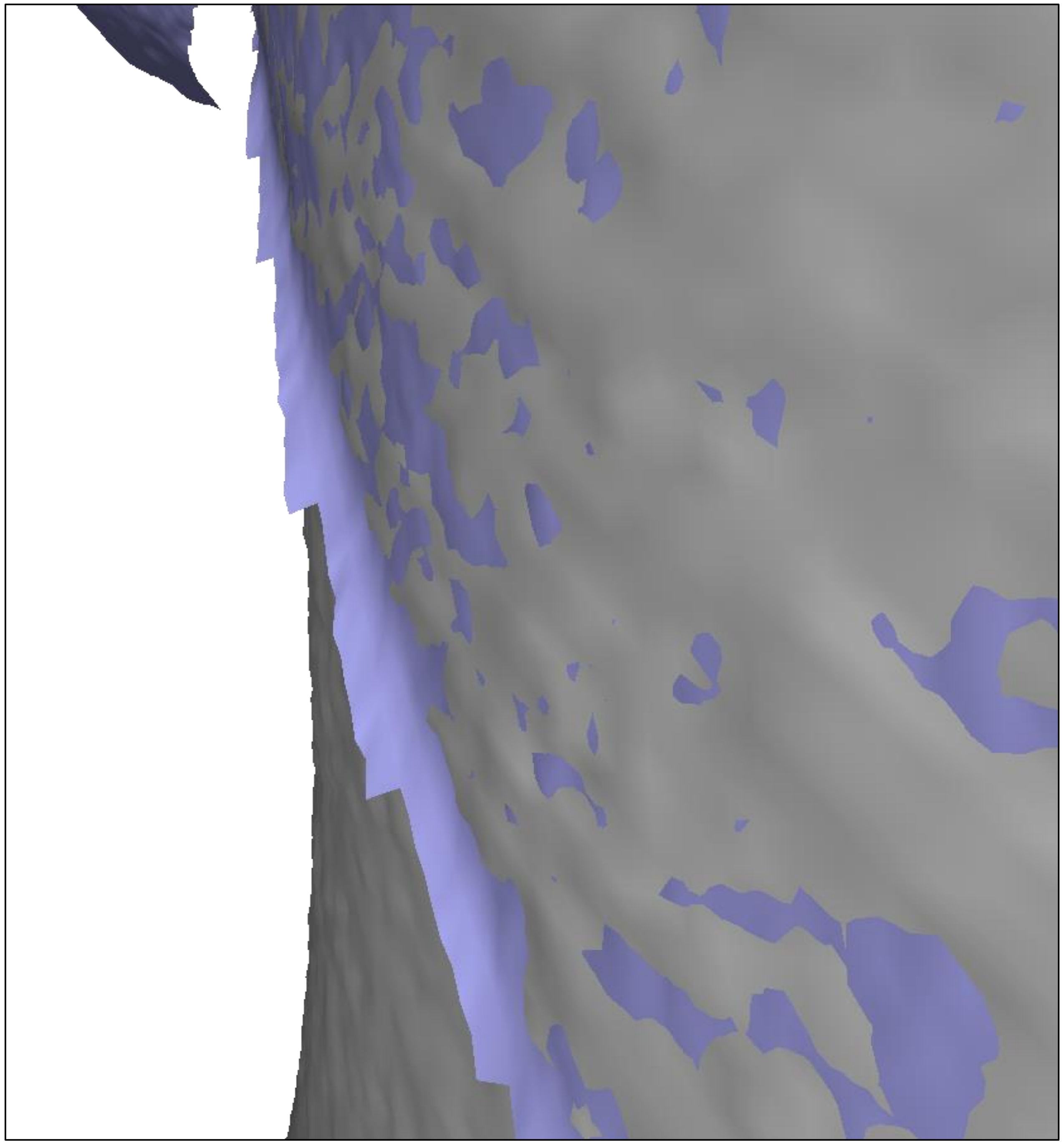


Surface Cleaning

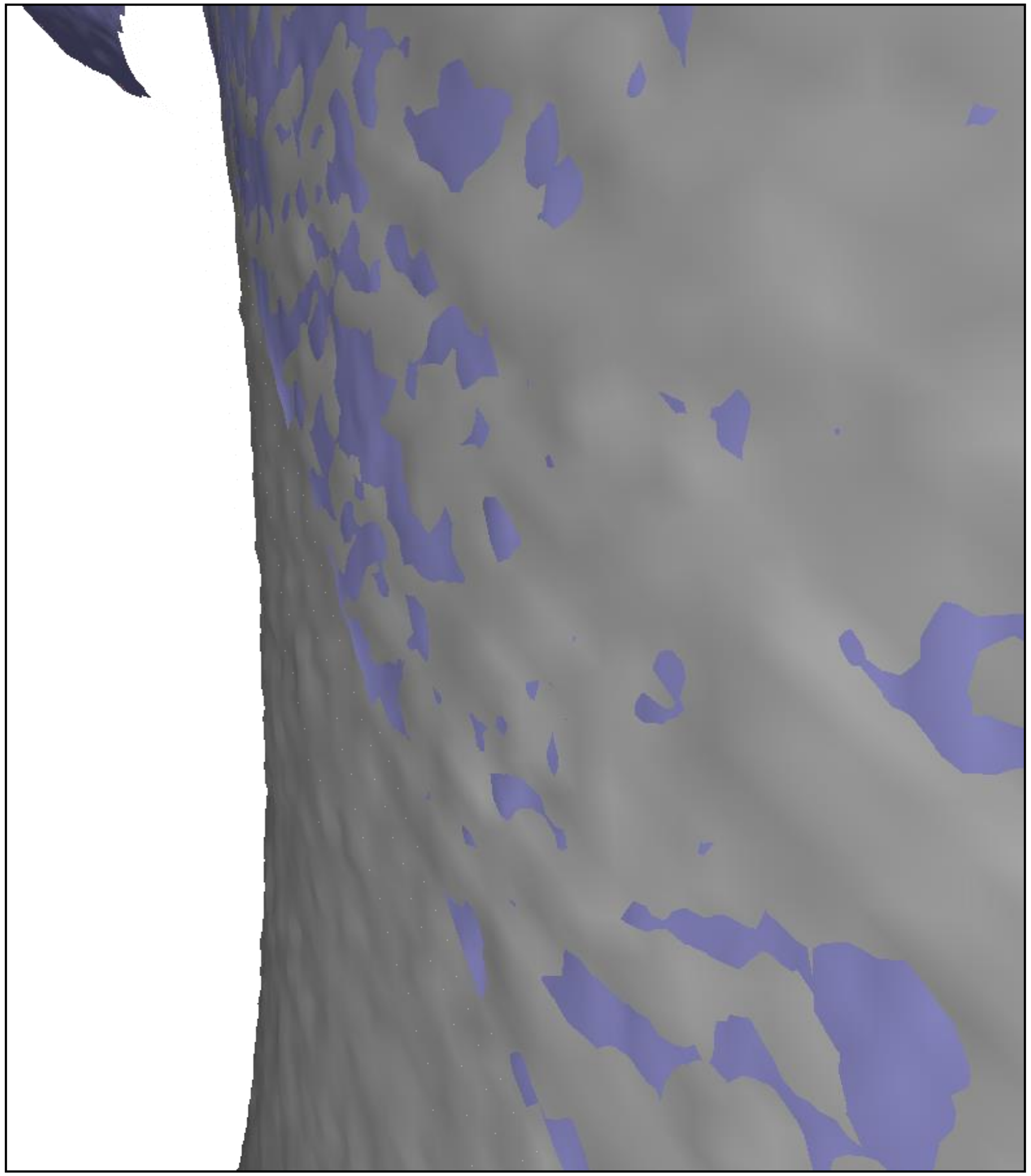
1. Select border triangles
   - Border triangle if an edge doesn’t have an adjacent face (using FF adjacency)
2. Dilate selection (eventually multiple times)
   - Add triangles that share and edge with the previous selection (using FF adjacency)
3. Remove selection



Surface Cleaning



BEFORE

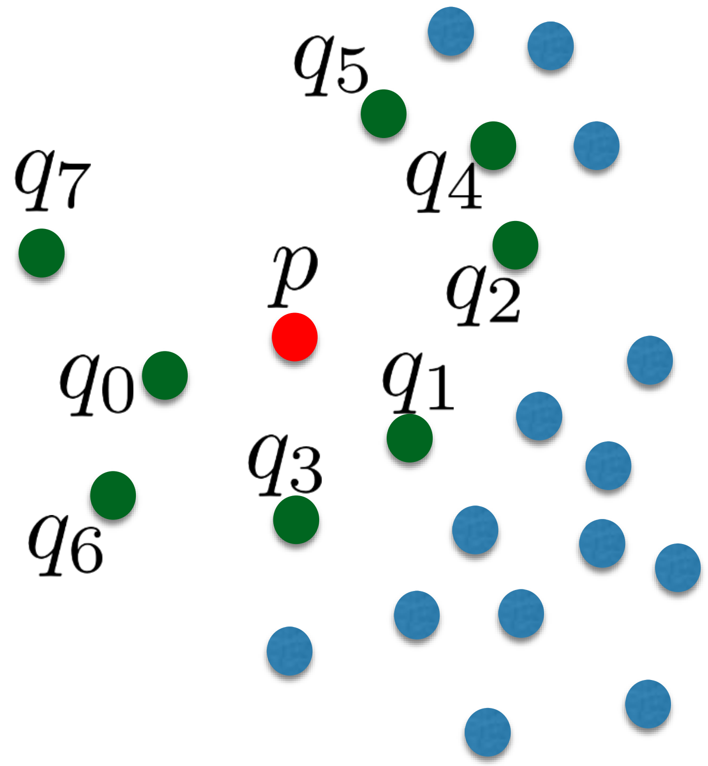


AFTER

Surface Cleaning

- Outliers removal based on local density in point cloud
    1. Compute density for each point using K-nearest point

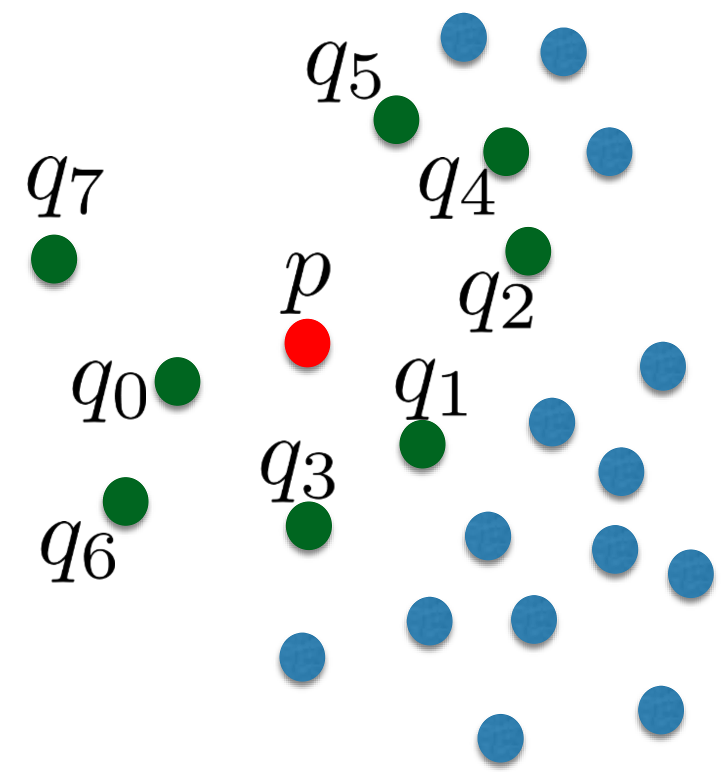
$$\sigma(p) = \sqrt{\frac{\sum_{q_i \in K_p} (p - q_i)^2}{\# K_p}}$$



Surface Cleaning

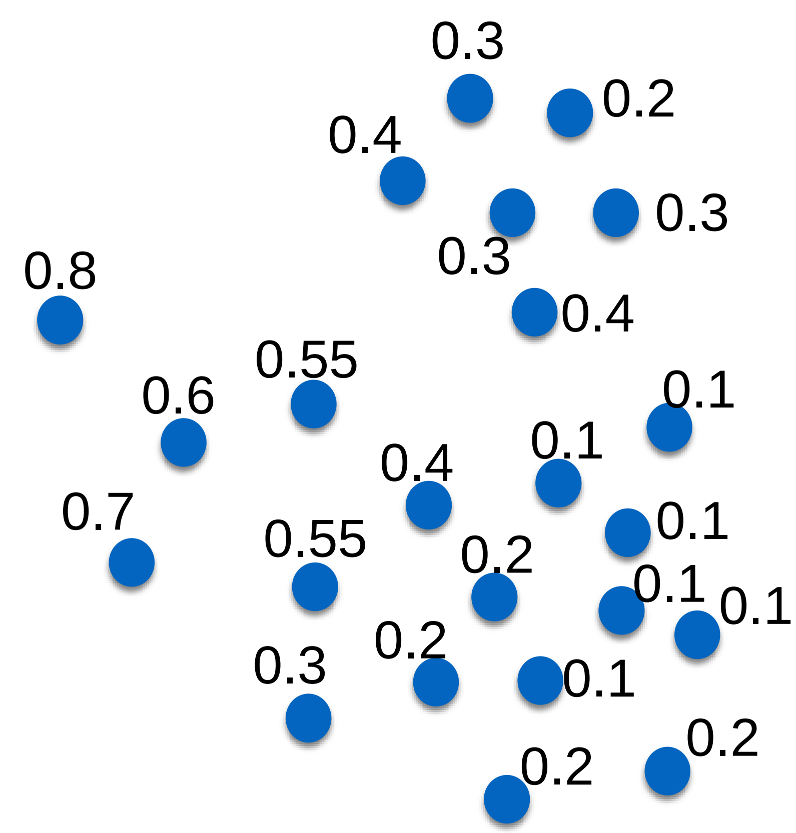
- Outliers removal based on local density in point cloud
  1. Compute density for each point using K-nearest point
  2. Comparison with the mean density of the neighbor point

$$\mathrm{PLOF}(p) = \frac{\sigma(p)}{\sum_{q_i \in K_p} \sigma(q_i)/\#K_p} - 1$$



Surface Cleaning

- Outliers removal based on local density in point cloud
  1. Compute density for each point using K-nearest point
  2. Comparison with the mean density of the neighbor point
  3. Probability computation with error Gaussian function

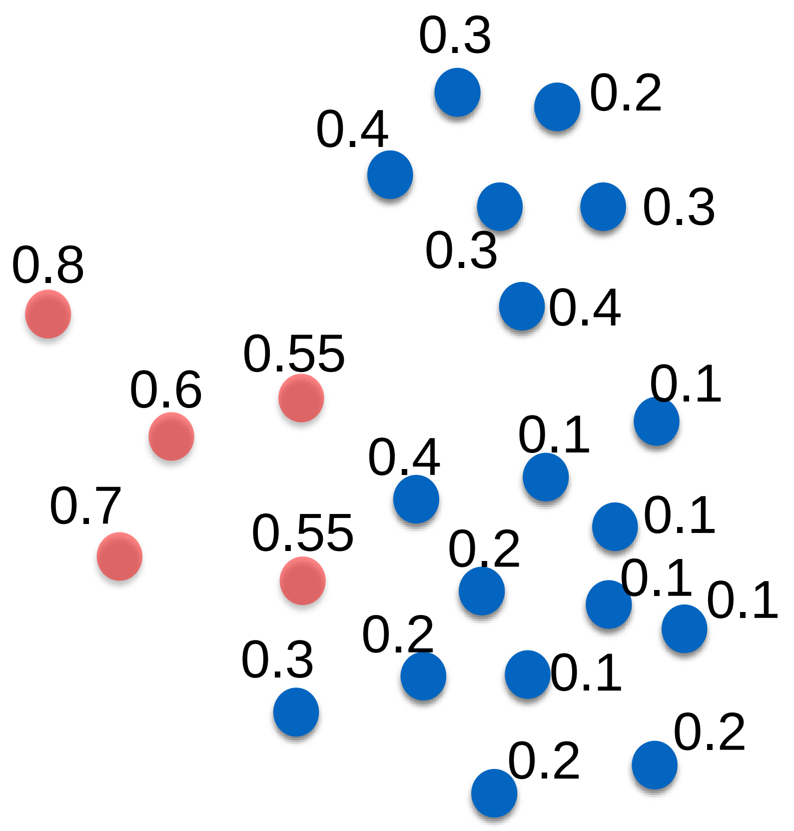


$$\mathrm{LoOP}(p) = \max\left\{0,\ \mathrm{erf}\left(\frac{\mathrm{PLOF}(p)}{nPLOF\sqrt{2}}\right)\right\}$$

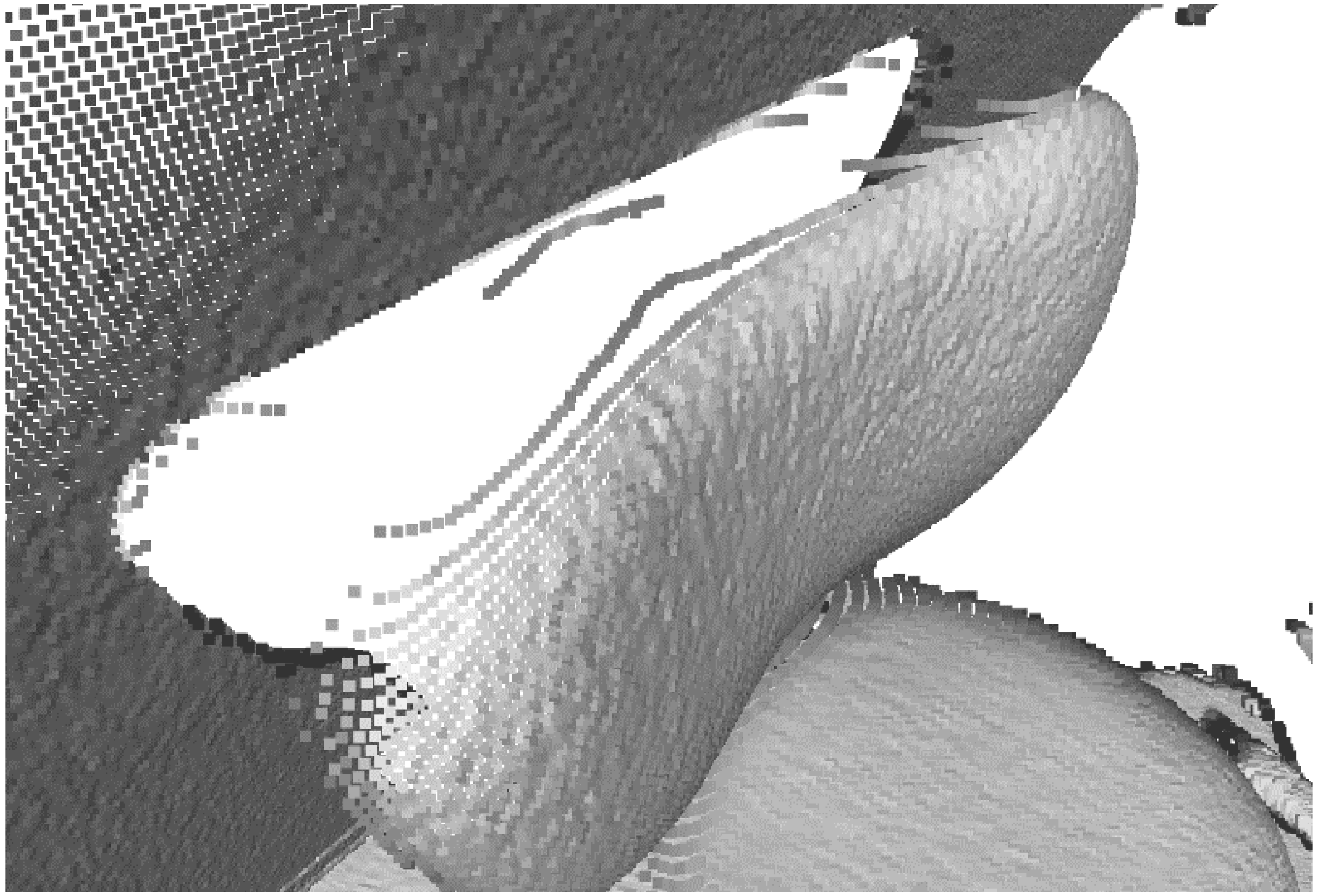
$$\mathrm{nPLOF} = \frac{\sum_{p_i \in \mathrm{Cloud}} \mathrm{PLOF}(p_i)^2}{\#\mathrm{Cloud}}$$

Surface Cleaning

- Outliers removal based on local density in point cloud
  1. Compute density for each point using K-nearest point
  2. Comparison with the mean density of the neighbor point
  3. Probability computation with error Gaussian function
  4. Remove point with probability higher than a threshold (typically 0.5)

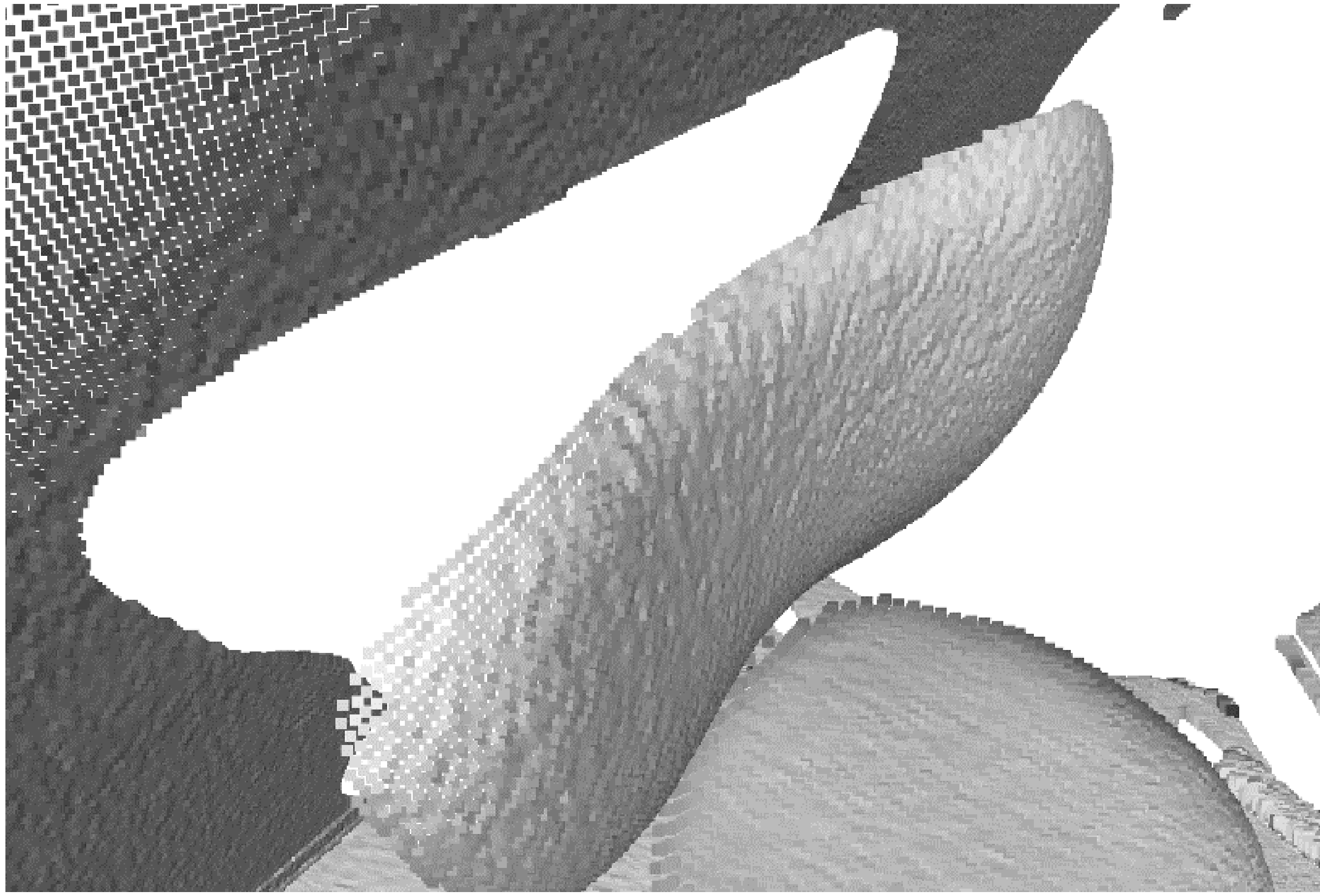


Surface Cleaning



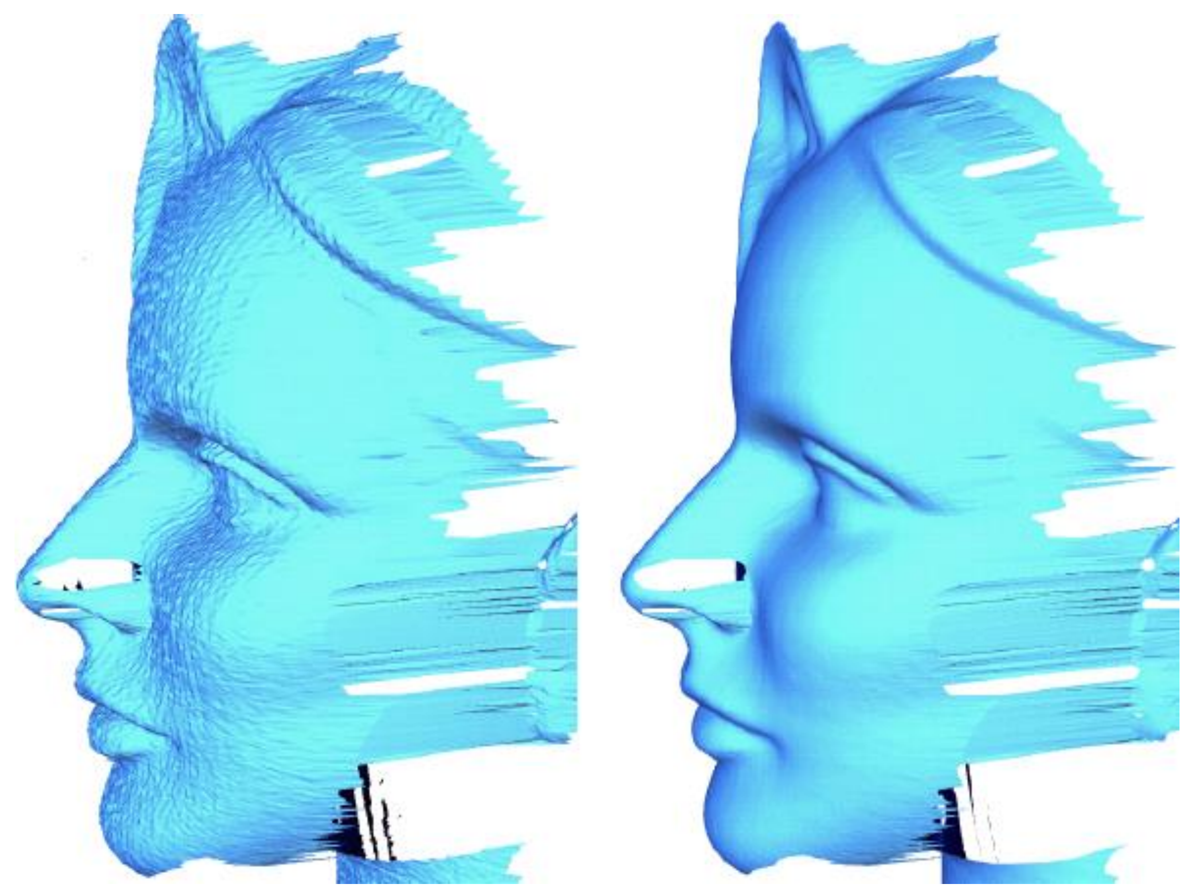
BEFORE

AFTER



Smoothing

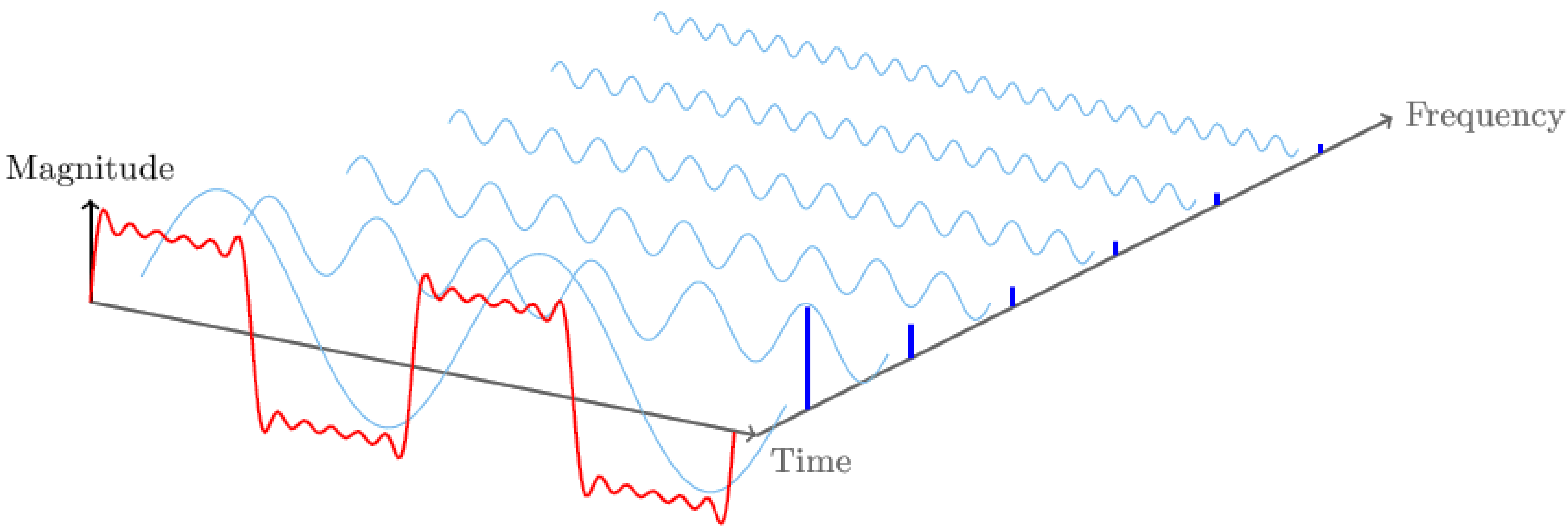
- Filtering out noise (high frequency components) from a mesh as in image



[Desbrun et al., SIGGRAPH 99]

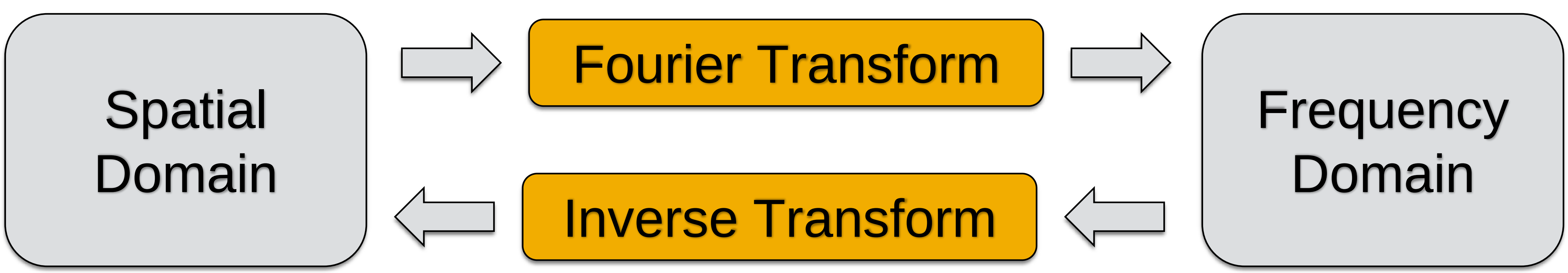
Fourier Transform

- Represent a function as a sum of sines and cosines



Fourier Transform

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



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

Filtering in Spatial Domain

- Smooth a signal by convolution with a kernel function (finite support kernel)

$$h(x) = (f * g)(x) = \int_{-\infty}^{+\infty} f(y)g(x-y)\mathrm{d}y$$

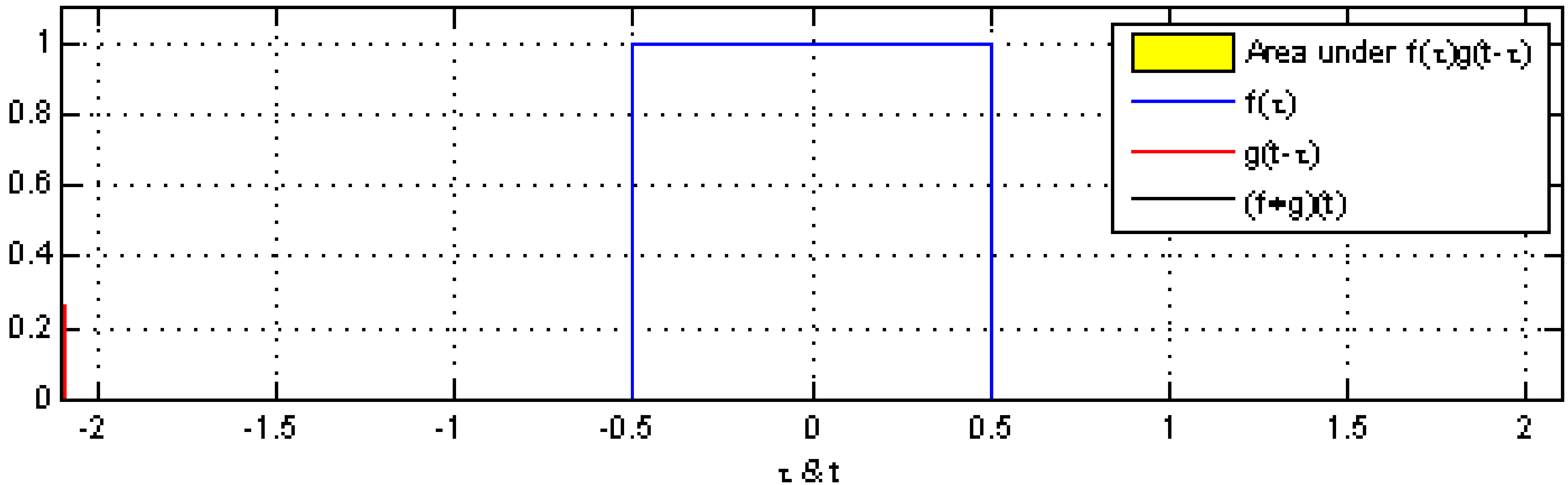
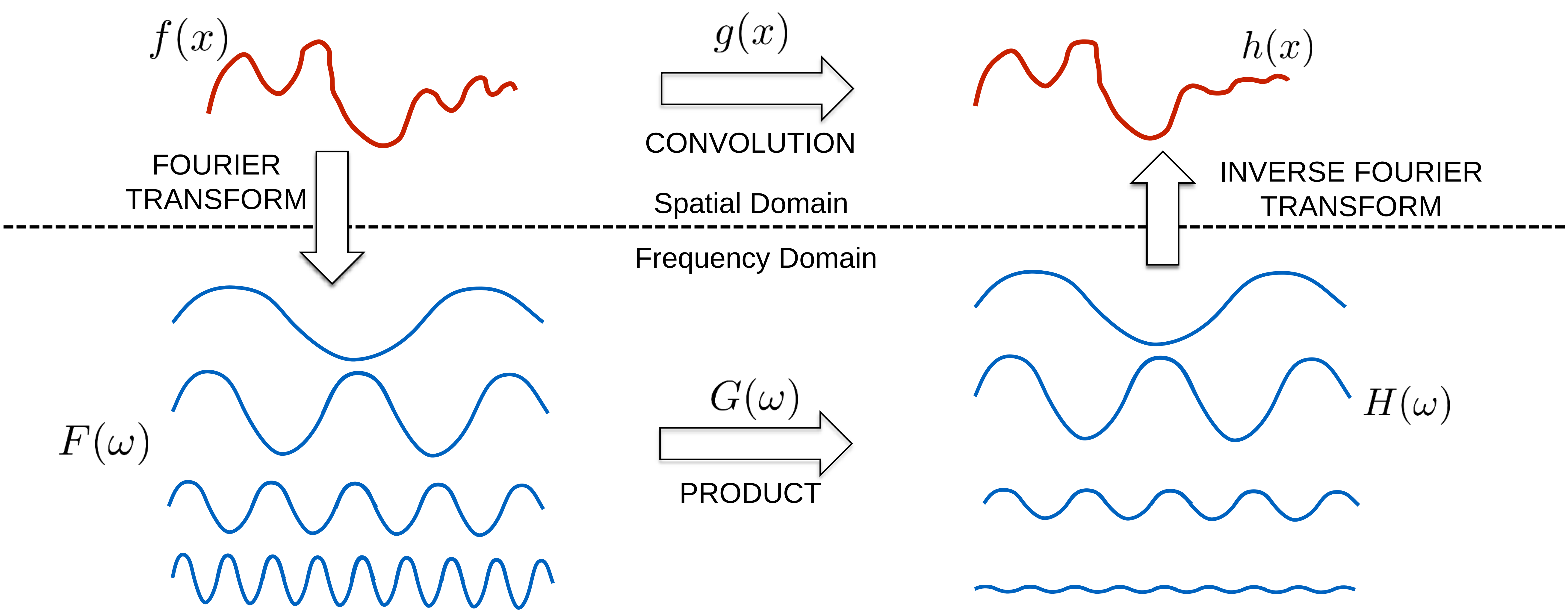


Image by Wikipedia

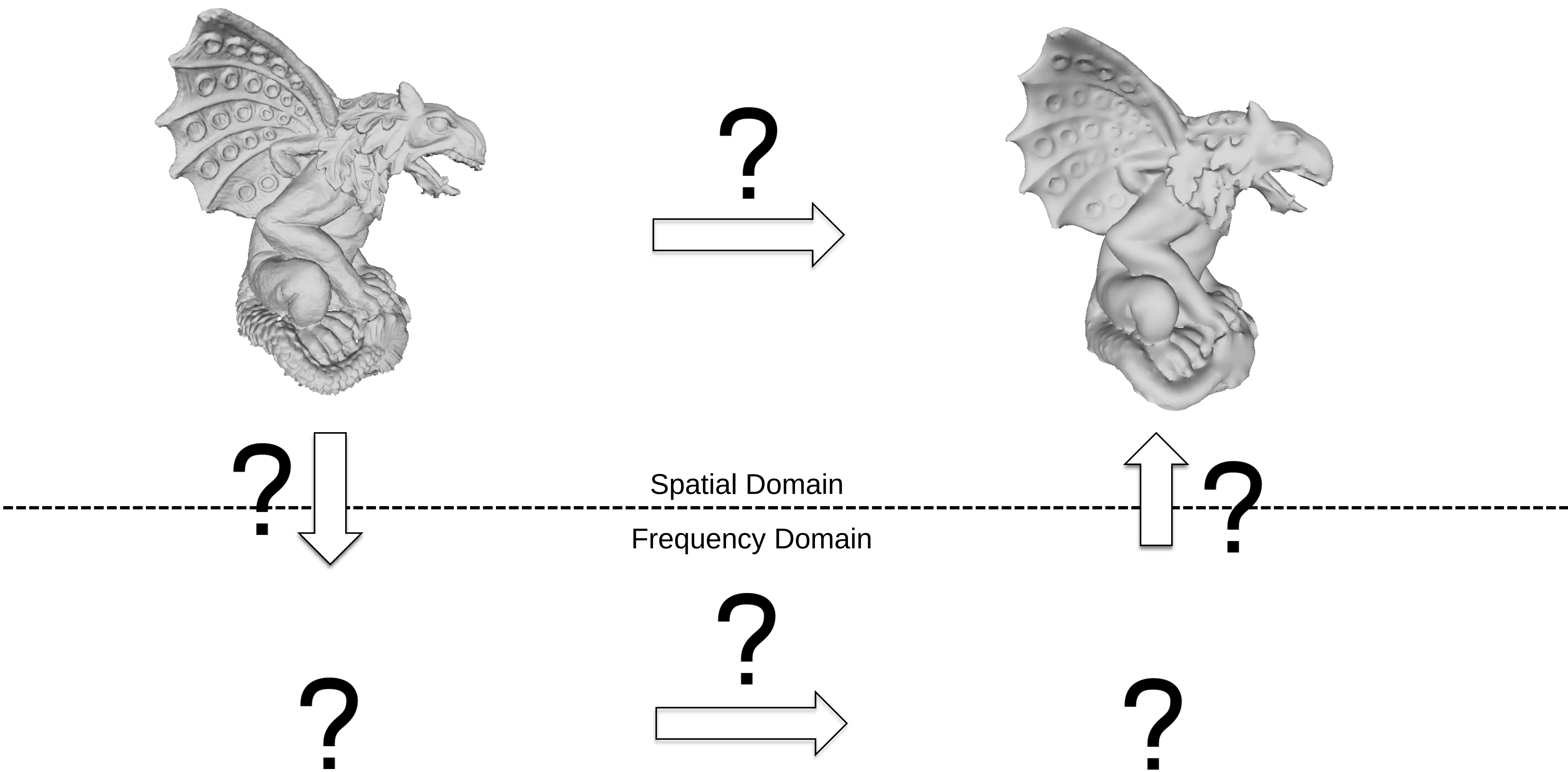
Filtering in Frequency Domain

- Convolution in spatial domain corresponds to multiplication in frequency domain

$$h(x) = (f * g)(x) \quad \Rightarrow \quad H(\omega) = F(\omega)G(\omega)$$



Filtering on Mesh?

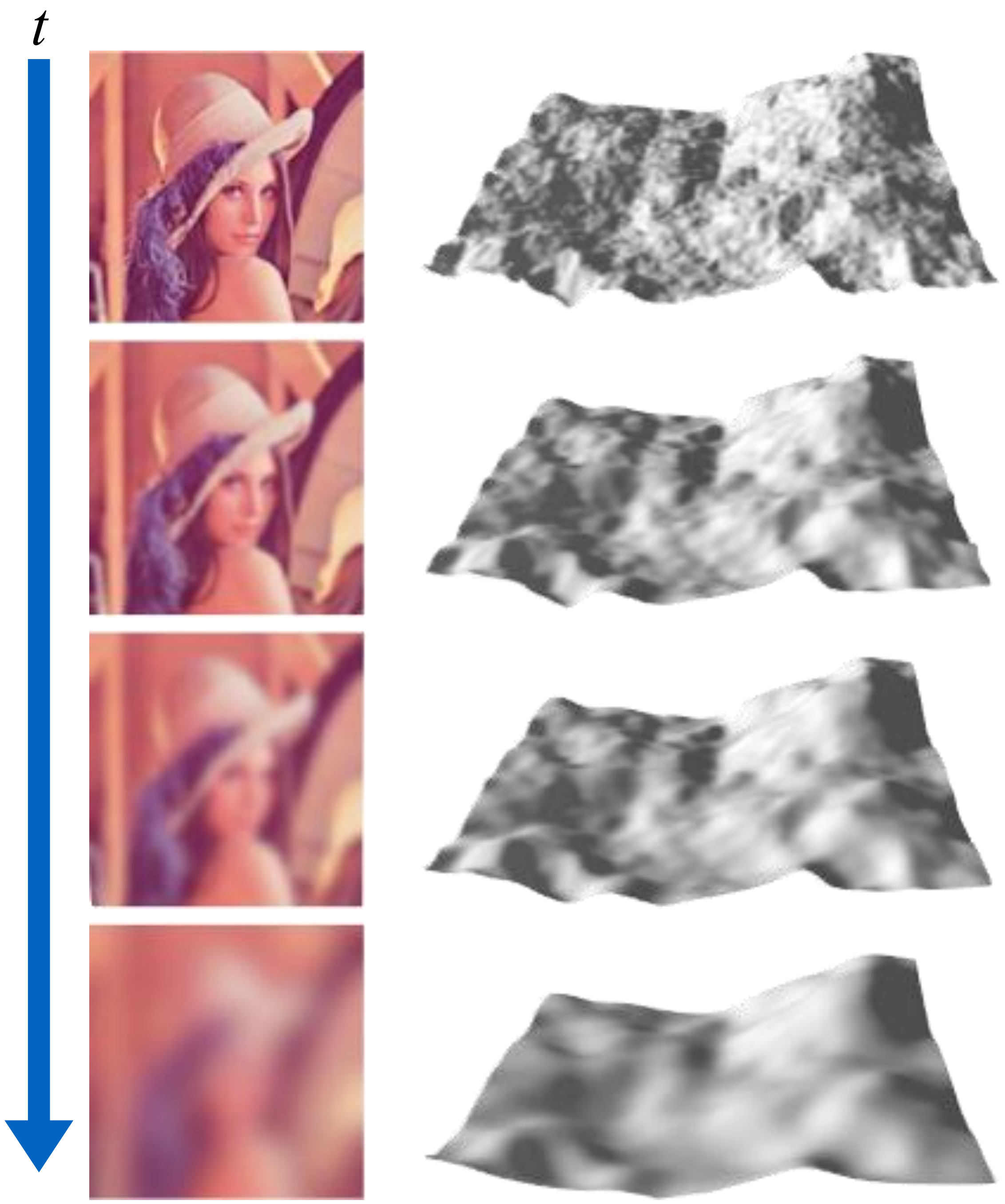


Diffusion equation

- Heat equation

$$\frac{\partial x}{\partial t} = \lambda \Delta x$$

- The function becomes smoother and smoother for increasing values of *t*



Laplacian Smoothing

- Discretization in space and time of the diffusion equation

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

$$\Downarrow$$

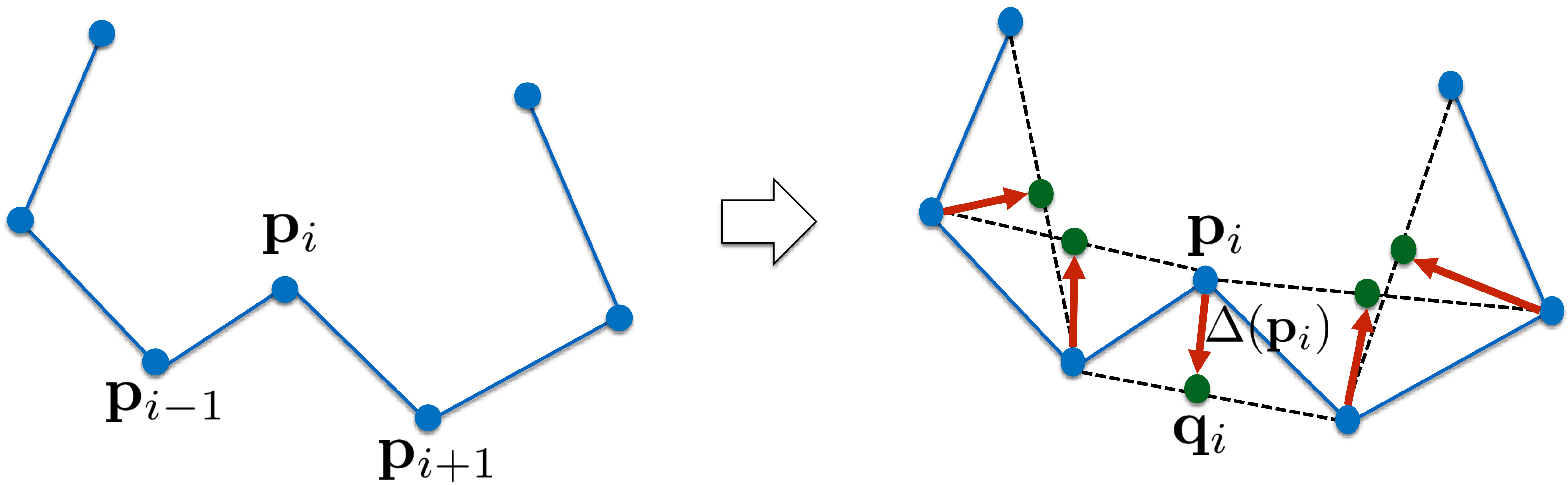
$$\frac{\partial f(x,t)}{\partial t} \approx \frac{f(x,t+h) - f(x,t)}{h}$$

$$\Downarrow$$

$$f(x,t+h) = f(x,t) + \lambda h \Delta f(x,t)$$

Laplacian Smoothing

- How to smooth a curve? Move each vertex in the direction of the mean of the neighbors



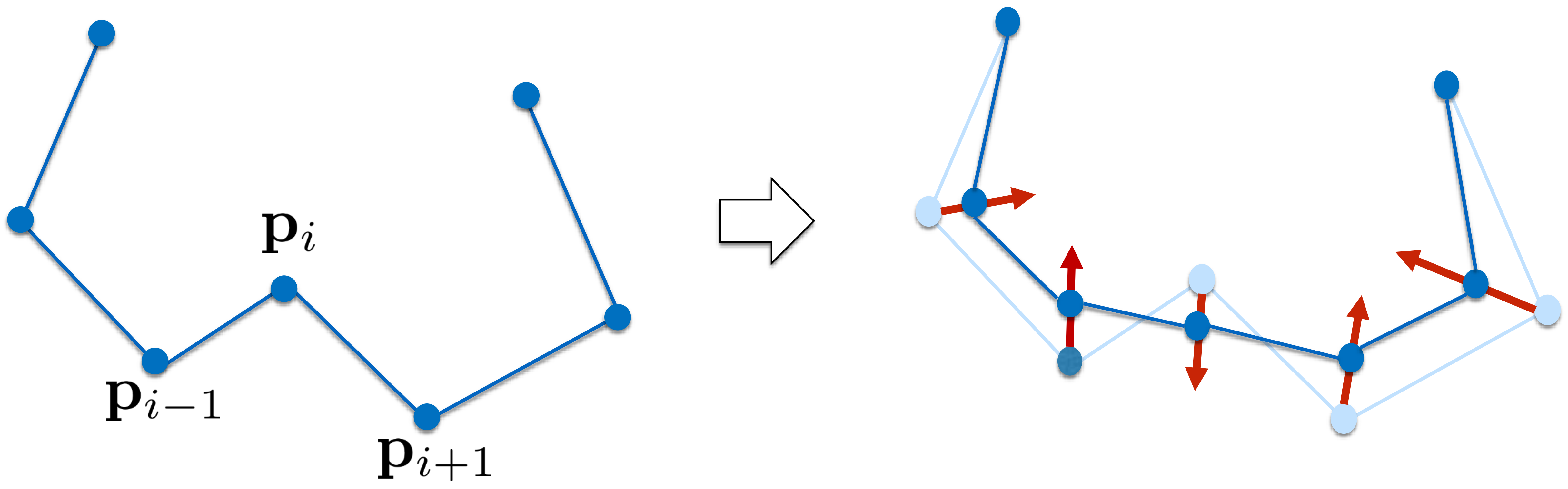
$$\mathbf{q}_i = (\mathbf{p}_{i+1} + \mathbf{p}_{i-1})/2$$

$$\boxed{\Delta(\mathbf{p}_i) = \mathbf{q}_i - \mathbf{p}_i = (\mathbf{p}_{i+1} + \mathbf{p}_{i-1})/2 - \mathbf{p}_i = (\mathbf{p}_{i+1} - \mathbf{p}_i)/2 + (\mathbf{p}_{i-1} - \mathbf{p}_i)/2}$$

Finite difference discretization of second derivative = Laplace operator

Laplacian Smoothing

- How to smooth a curve? Move each vertex in the direction of the mean of the neighbors



$$\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} + \lambda \Delta(\mathbf{p}_i^{(t)})$$

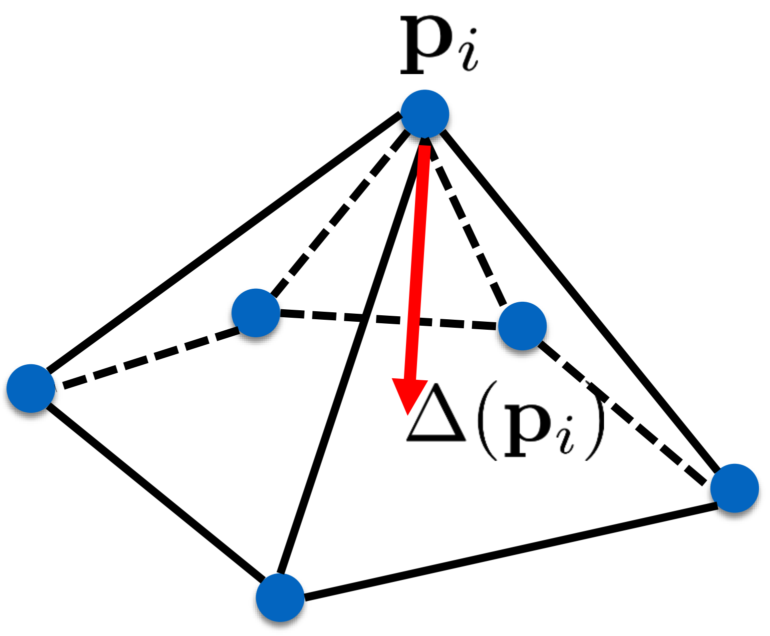
$$0 < \lambda < 1$$

Laplacian Smoothing on Mesh

- Same as for curve.
  1. For each vertex, it computes the displacement vector towards the average of its adjacent vertices.
  2. Move each vertex by a fraction of its displacement vector

$$\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} + \lambda \Delta(\mathbf{p}_i^{(t)})$$

$$0 < \lambda < 1$$

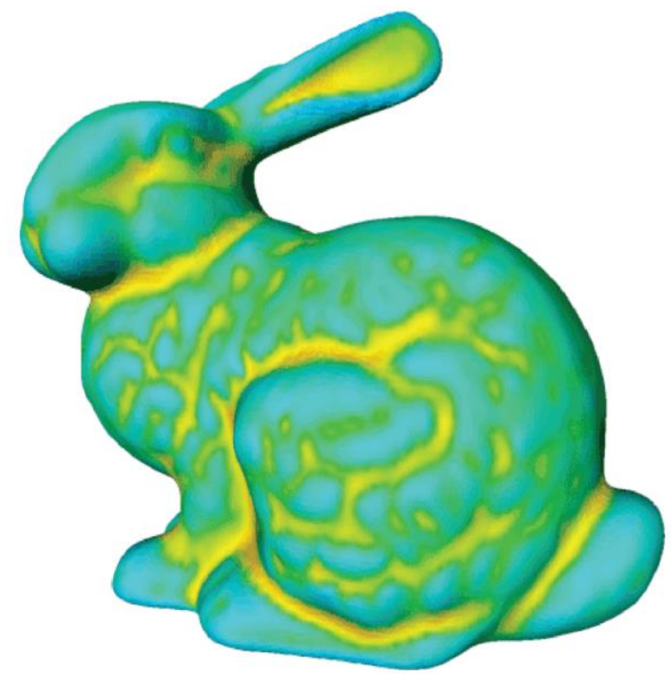


- Umbrella operator

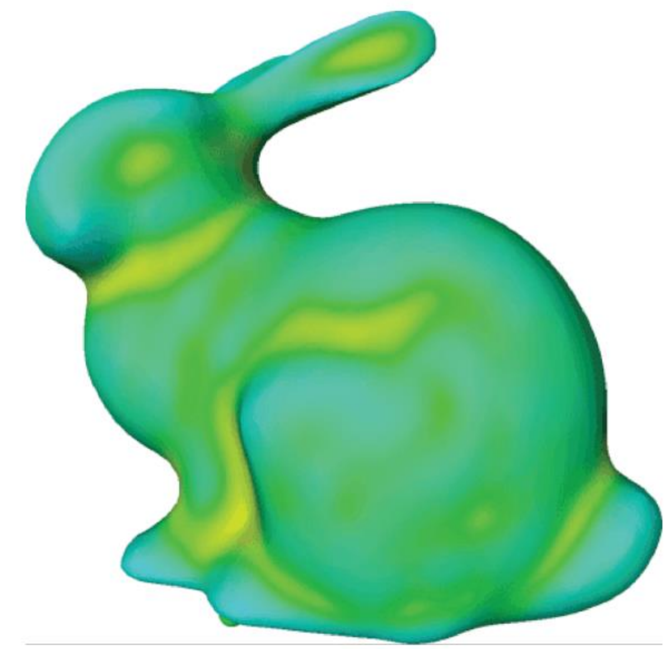
$$\Delta(\mathbf{p}_i) = \frac{1}{|N_i|}(\sum_{j \in N_i} \mathbf{p}_j) - \mathbf{p}_i$$

Laplacian Smoothing on Mesh

0 Iterations



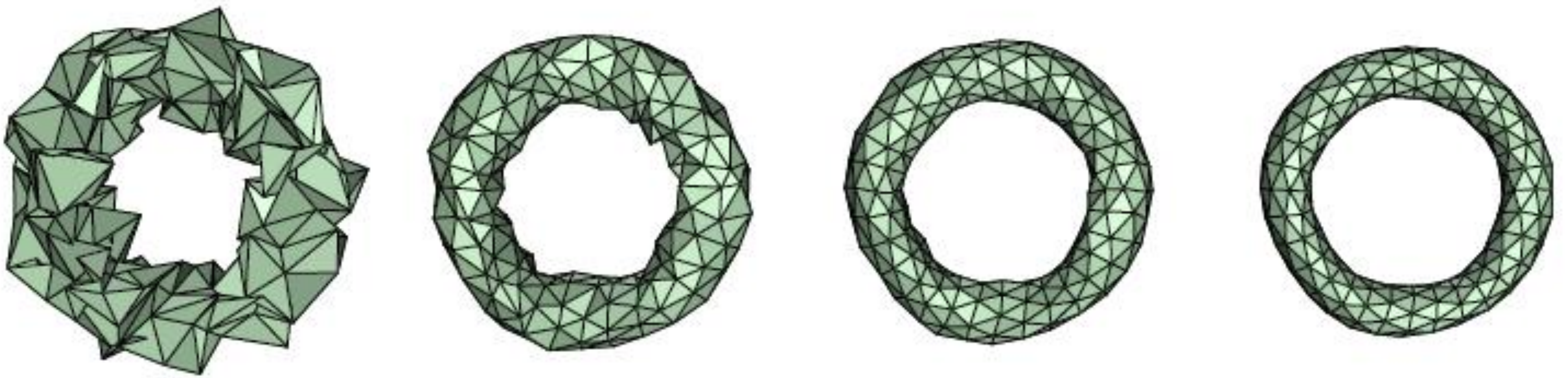
5 Iterations



20 Iterations

Laplacian Smoothing on Mesh

- Problem - Repeated iterations of Laplacian smoothing shrinks the mesh



Taubin Smoothing

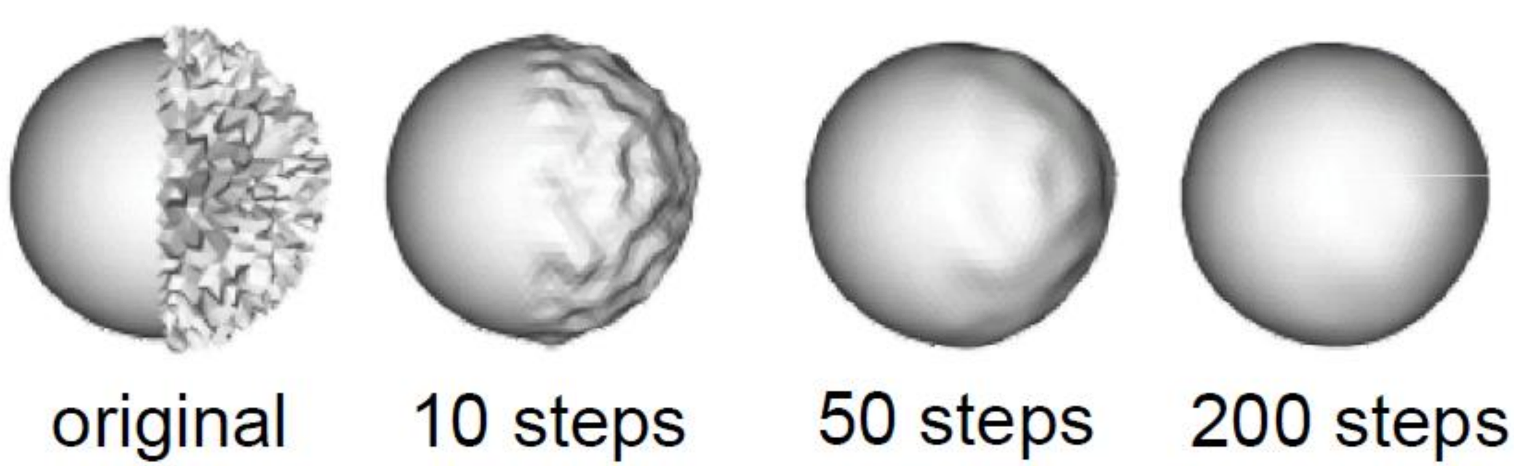
[Taubin et al., SIGGRAPH 95]

- For each iteration performs 2 steps:
  1. Shrink. Compute the laplacian and moves the vertices by λ times the displacement.

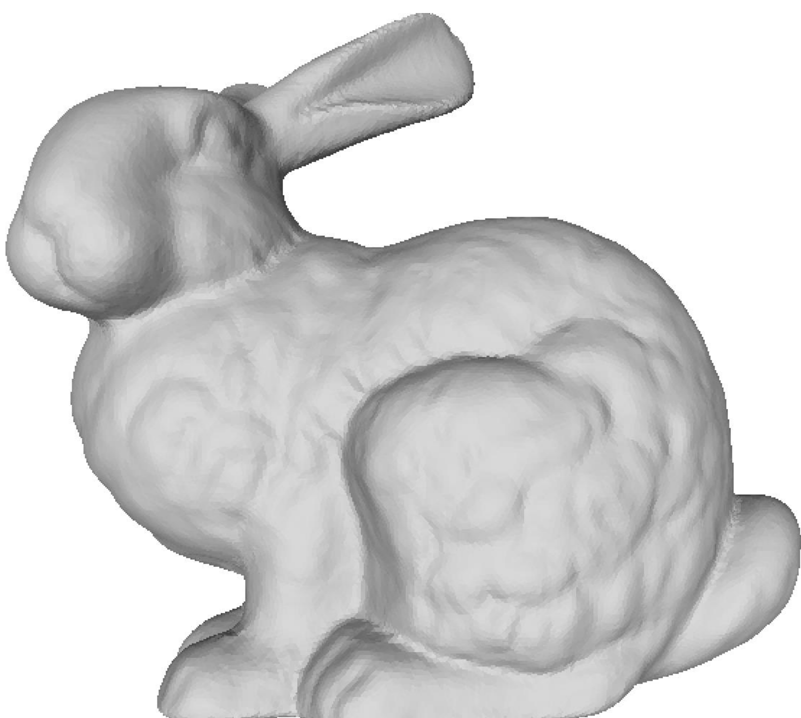
$$\mathbf{p}_i = \mathbf{p}_i + \lambda\Delta(\mathbf{p}_i) \quad \text{with} \quad \lambda > 0$$

  2. Inflate. Compute again the laplacian and moves back each vertex by μ times the displacement.

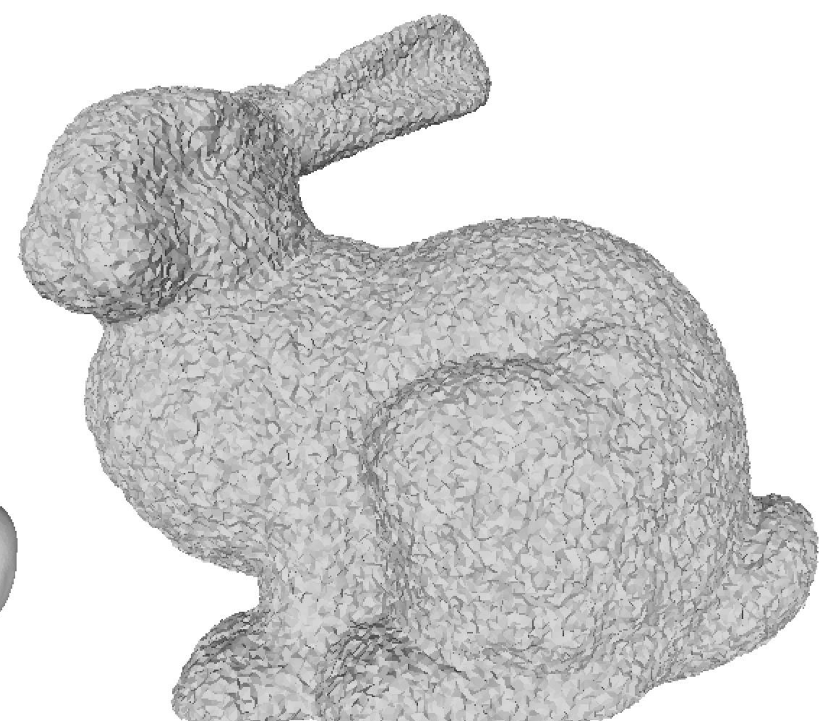
$$\mathbf{p}_i = \mathbf{p}_i + \mu\Delta(\mathbf{p}_i) \quad \text{with} \quad \mu < 0$$



Taubin vs Laplacian

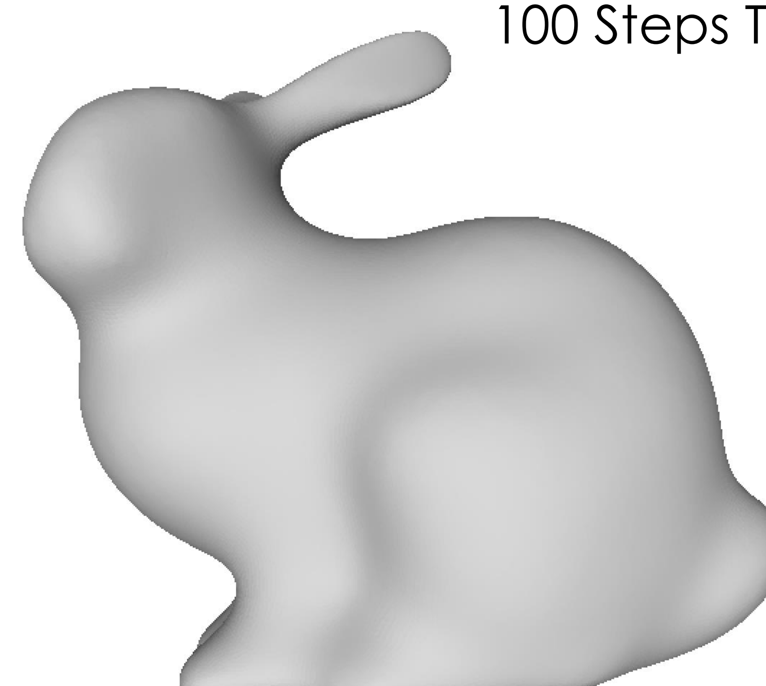


Original



Noise Added

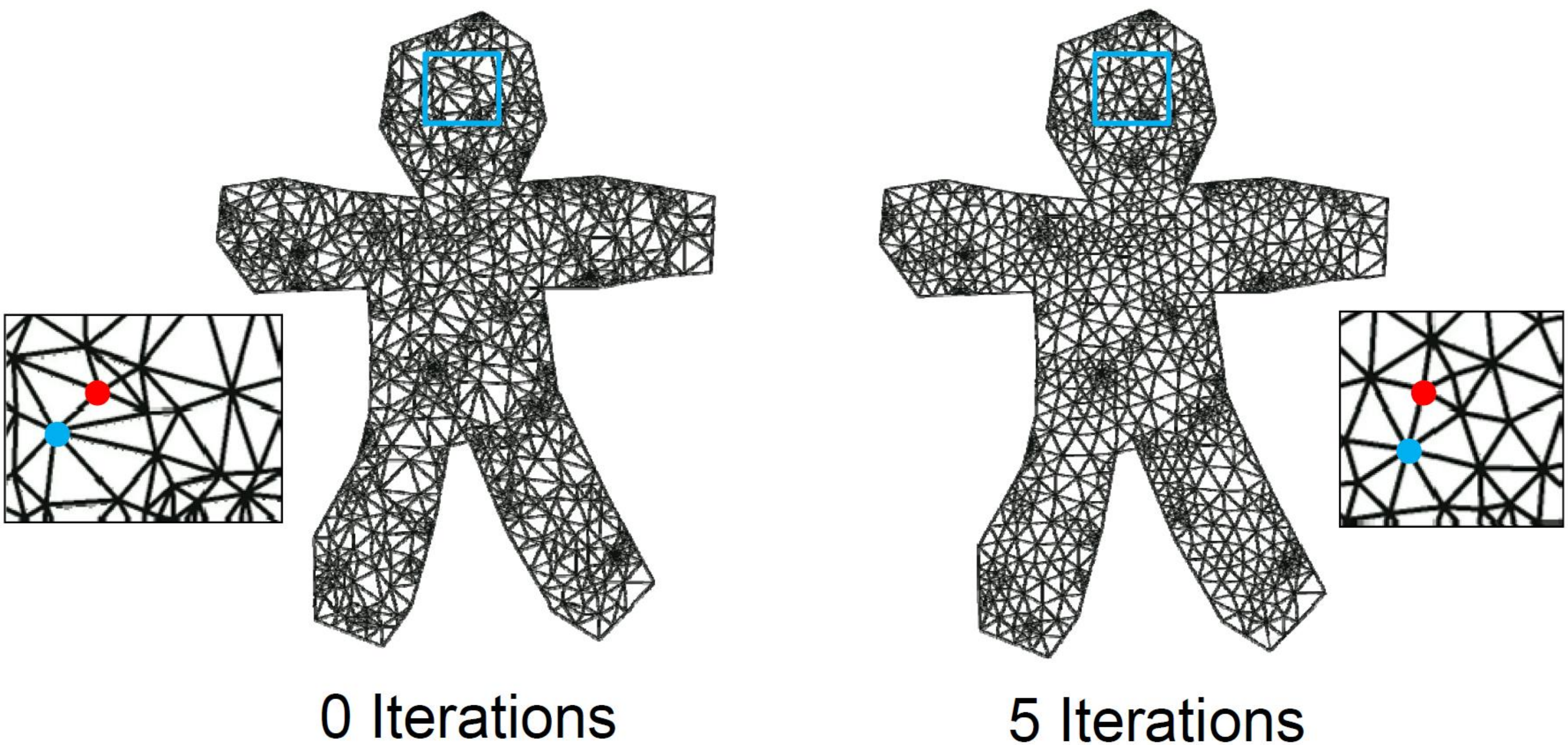
100 Steps Taubin



100 Steps Laplacian

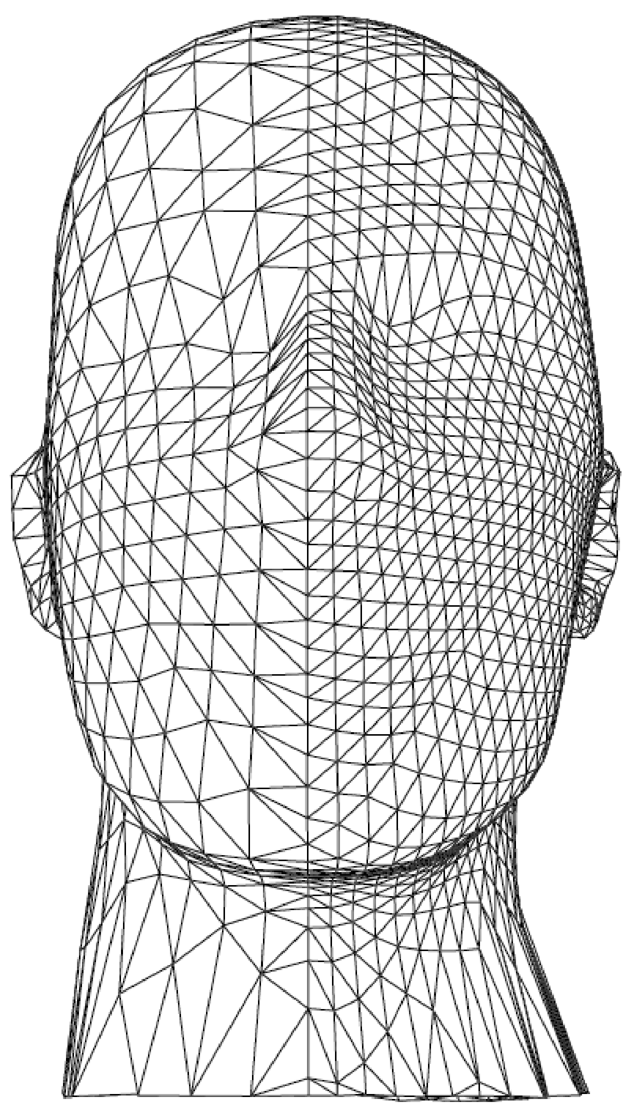
Laplace Operator -Problems

- Flat surface should stay the same after smoothing

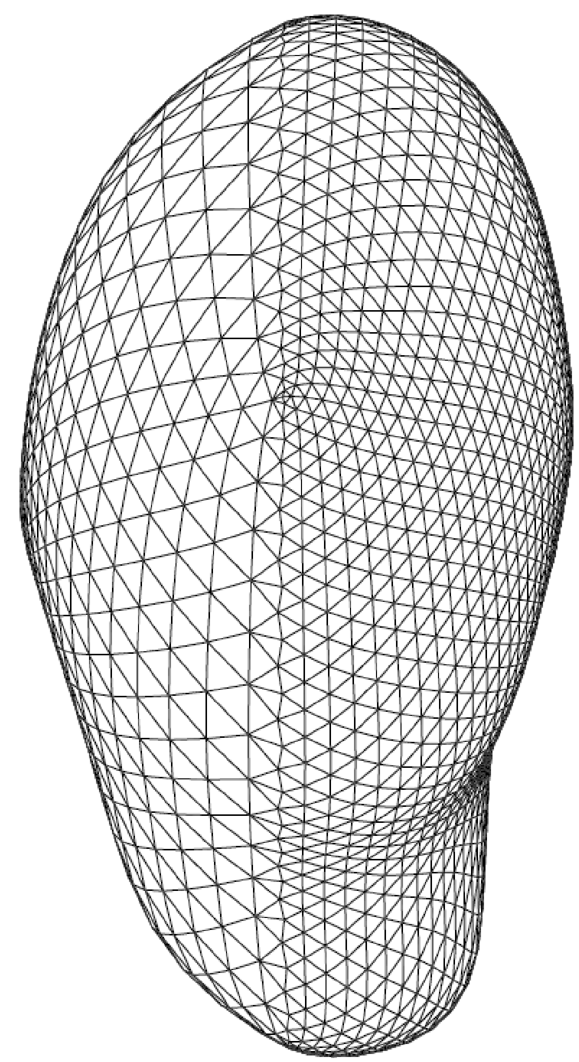


Laplace Operator Problem

- The result should not depend on triangle sizes



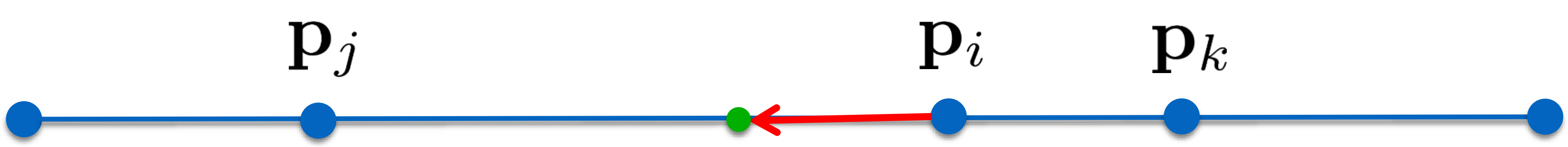
Original



Laplacian

Laplace Operator

- Back to curves



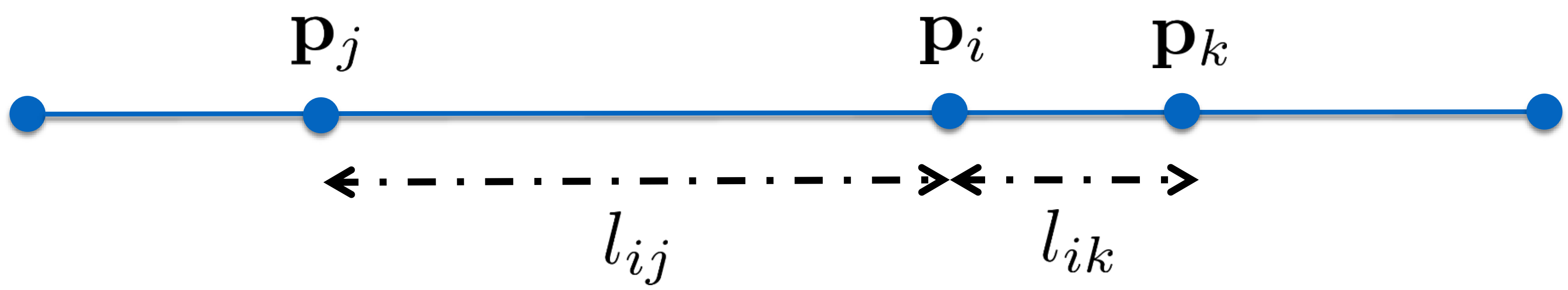
$$\Delta(\mathbf{p}_i) = (\mathbf{p}_j + \mathbf{p}_k)/2 - \mathbf{p}_i$$

- The same weight for both the neighbors, although one is closer

- The displacement vector should be null

Laplace Operator

- Use a weighted average to compute the displacement vector



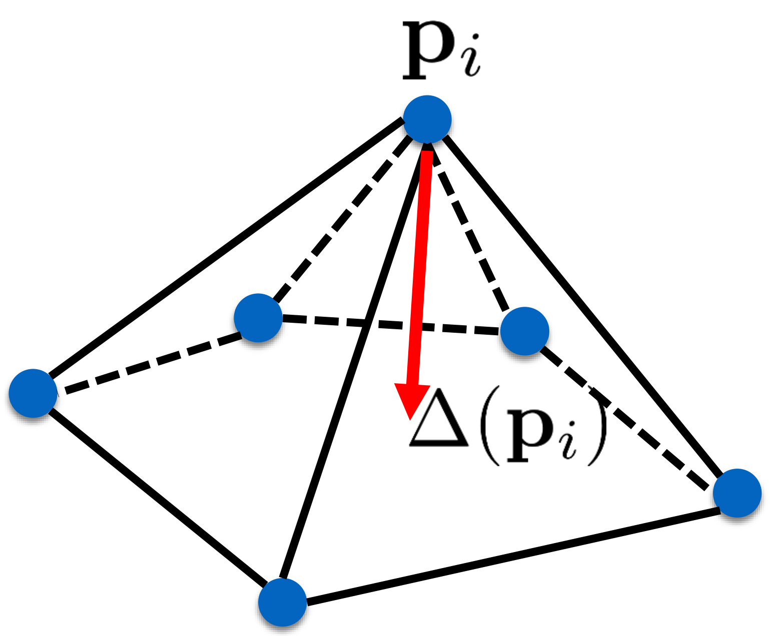
$$w_{ij} = \frac{1}{l_{ij}} \qquad w_{ik} = \frac{1}{l_{ik}} \qquad \Delta(\mathbf{p}_i) = \frac{w_{ij}\mathbf{p}_j + w_{ik}\mathbf{p}_k}{w_{ij} + w_{ik}} - \mathbf{p}_i$$

- Strait curve will be invariant to smoothing

Laplace Operator on Mesh

- Use a weighted average to compute the displacement vector

$$\Delta(\mathbf{p}_i) = \frac{1}{W} \sum_{j \in N_i} w_{ij} (\mathbf{p}_j - \mathbf{p}_i)$$



- Scale-dependent Laplace operator

- Laplace-Beltrami operator

[Desbrun et al., SIGGRAPH 99]

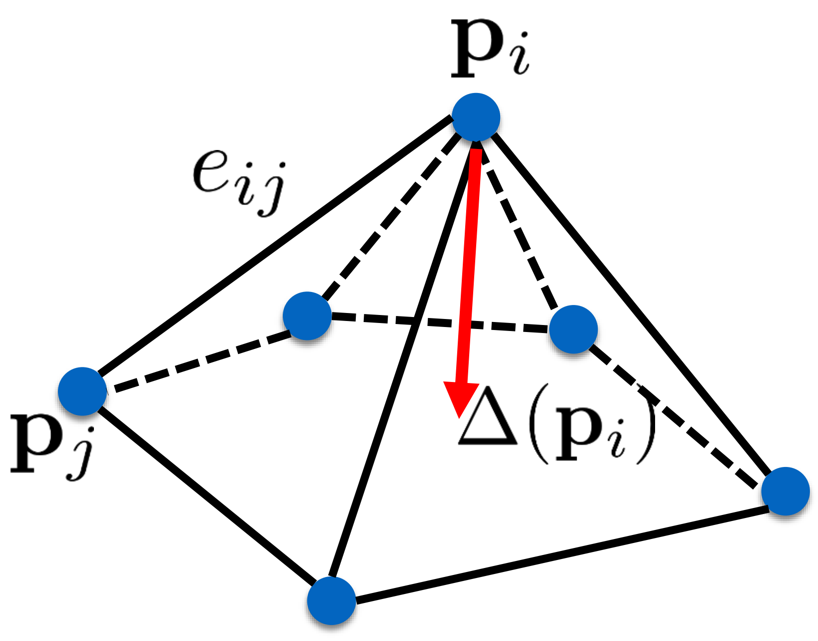
Scale-dependent Laplace Operator

- Substitute regular Laplacian with an operator that weights vertices by considering involved edges

$$\Delta(\mathbf{p}_i) = \frac{2}{E} \sum_{j \in N_i} \frac{\mathbf{p}_j - \mathbf{p}_i}{e_{ij}}$$

with

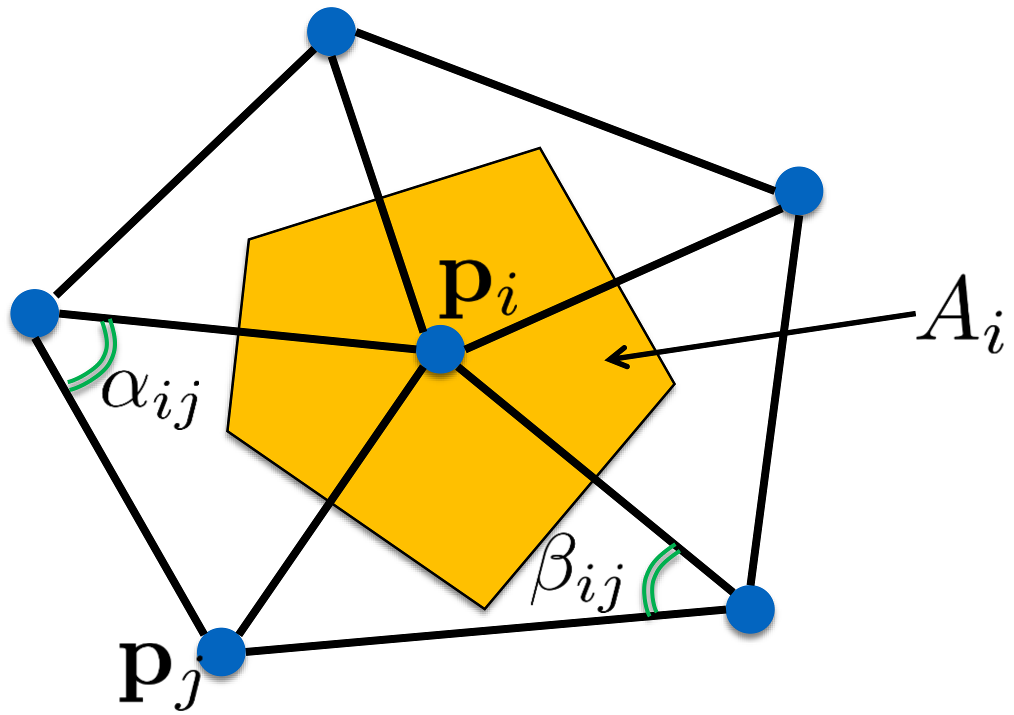
$$E = \sum_{j \in N_i} e_{ij} = \sum_{j \in N_i} |\mathbf{p}_j - \mathbf{p}_i|$$



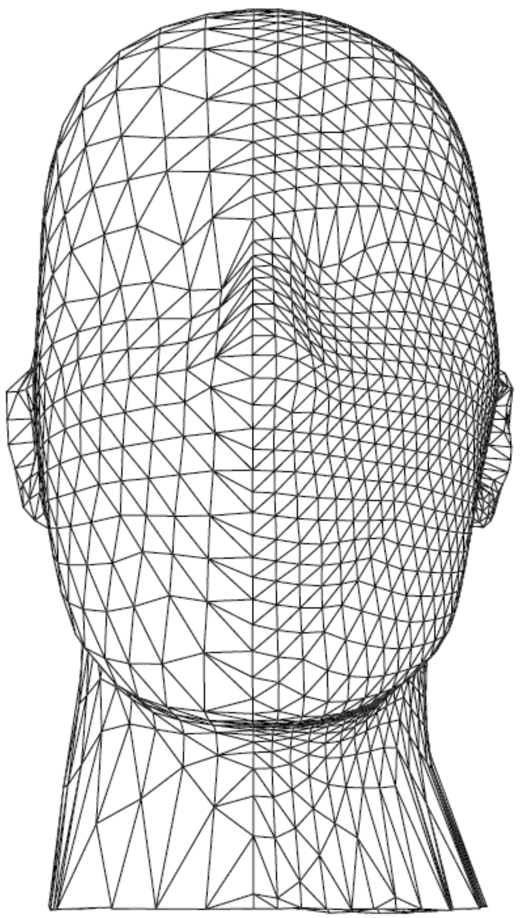
Laplace-Beltrami Operator

- Weight that depends on the difference of mean curvature (cotangent weight)

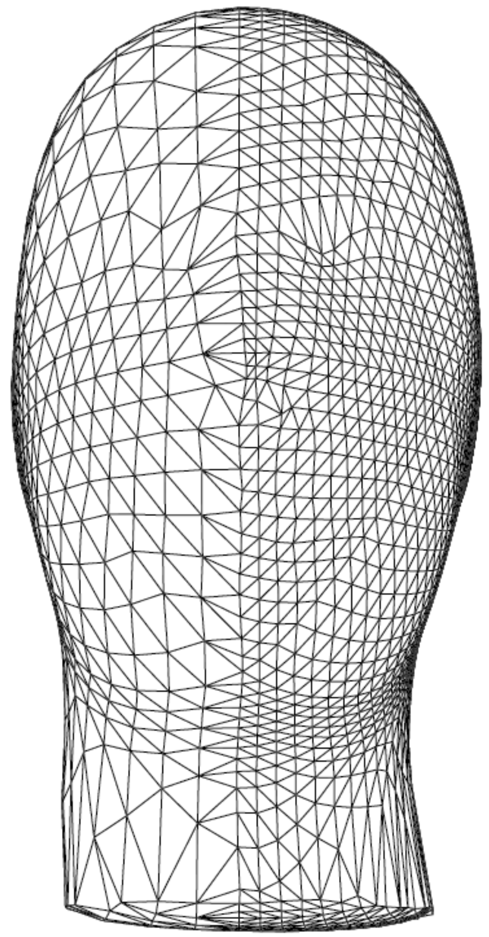
$$\Delta_S(\mathbf{p}_i) = \frac{1}{2A_i} \sum_{j \in N_i} (\cot \alpha_{ij} + \cot \beta_{ij})(\mathbf{p}_j - \mathbf{p}_i)$$



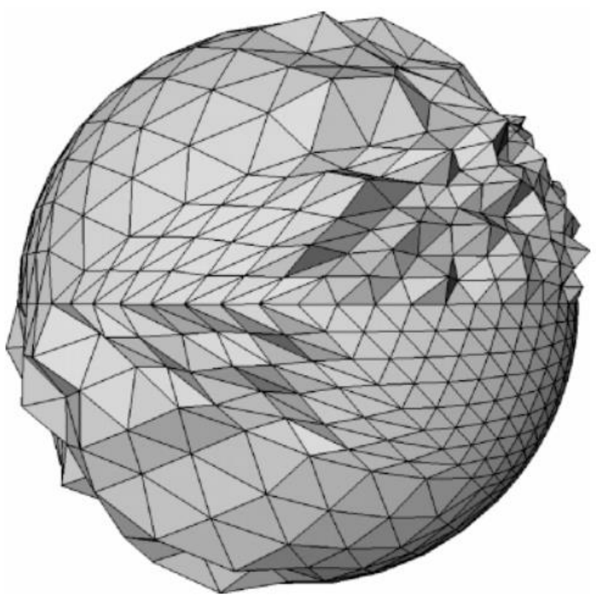
Original



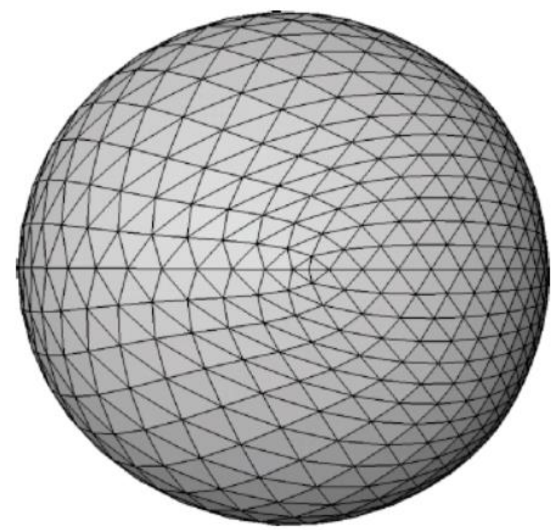
Cotangent weight



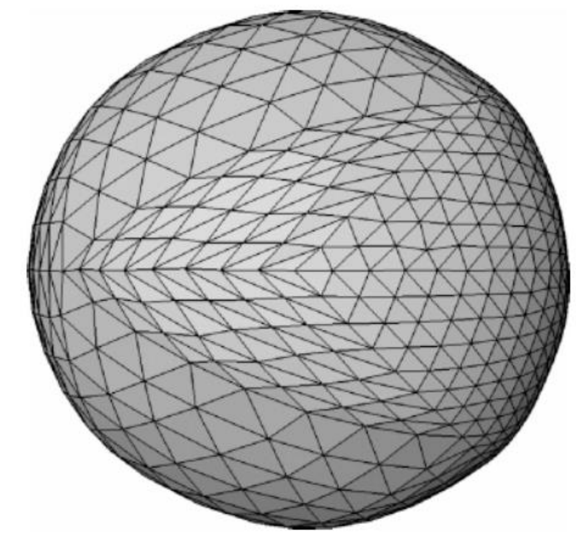
Umbrella Operator vs Laplace-Beltrami



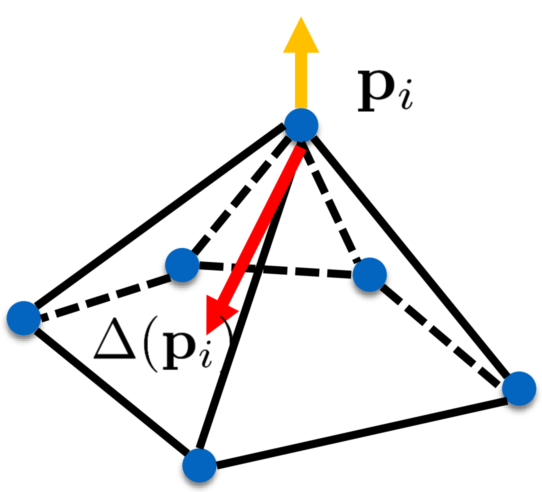
ORIGINAL



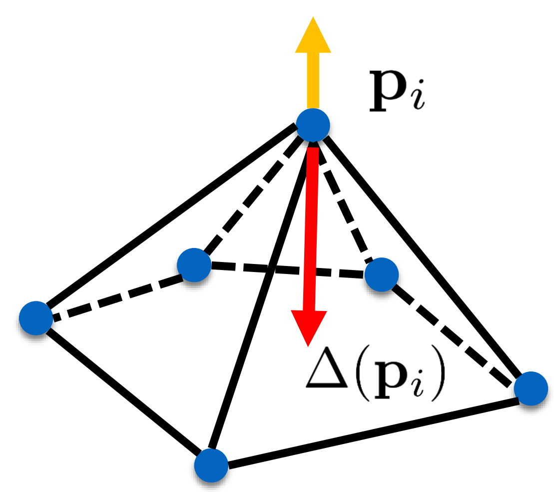
UMBRELLA



LAPLACE-BELTRAMI

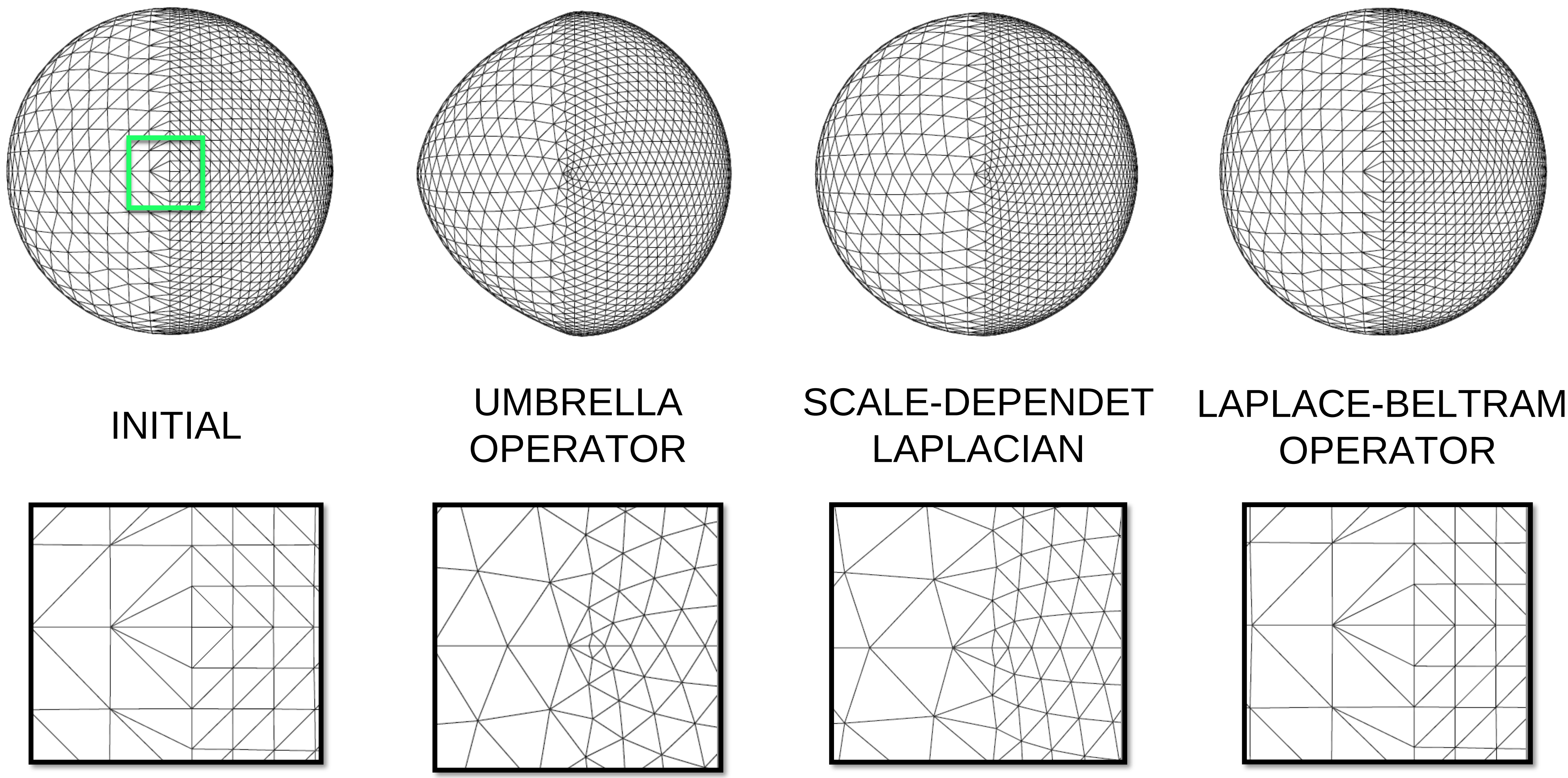


Tangential drift



Moves vertices along normal

Comparison



[Desbrun et al., SIGGRAPH 99]

Numerical Integration

- Write update in matrix form

$$\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} + \lambda\Delta(\mathbf{p}_i^{(t)}) \qquad \Delta(\mathbf{p}_i) = \frac{1}{W}\sum_{j\in N_i} w_{ij}(\mathbf{p}_j - \mathbf{p}_i)$$

$$\mathbf{P}^{(t)} = (\mathbf{p}_1^{(t)}, \ldots, \mathbf{p}_n^{(t)}) \in \mathbb{R}^{n\times 3}$$

- Laplacian Matrix $\mathbf{L} = \mathbf{DM} \in \mathbb{R}^{n\times n}$

$$\mathbf{M}_{ij} = \begin{cases} w_{ij} & \text{if} \quad i \neq j \\ \sum_{j\in N_i} w_{ij} & \text{if} \quad i = j \\ 0 & \end{cases} \qquad \mathbf{D} = \operatorname{diag}\Big(\ldots, \frac{1}{W}, \ldots\Big)$$

Numerical Integration

- **Explicit Euler integration**: resolve the system by iterative substitution requiring small λ for stability

$$\mathbf{P}^{(t+1)} = \mathbf{P}^{(t)} + \lambda \mathbf{L} \mathbf{P}^{(t)} = (\mathbf{I} + \lambda \mathbf{L}) \mathbf{P}^{(t)}$$

- **Implicit Euler integration**: resolve the following linear system (the system is very large but sparse)

$$(\mathbf{I} - \lambda \mathbf{L}) \mathbf{P}^{(t+1)} = \mathbf{P}^{(t)}$$

Spectral Analysis

- Eigen-decomposition of Laplacian matrix

$$\mathbf{L} = \mathbf{D}\mathbf{M} \in \mathbb{R}^{n \times n} \quad \Rightarrow \quad \mathbf{L}\mathbf{v} = \lambda\mathbf{v} \quad \Rightarrow \quad \mathbf{L} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{-1}$$

$$\mathbf{V} = \begin{bmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_1 & \dots & \mathbf{v}_n \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

Eigenvector are the natural vibrations

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \dots & \\ & & & \lambda_n \end{bmatrix} \in \mathbb{R}^{n \times n}$$

Eigenvalues are the natural frequencies

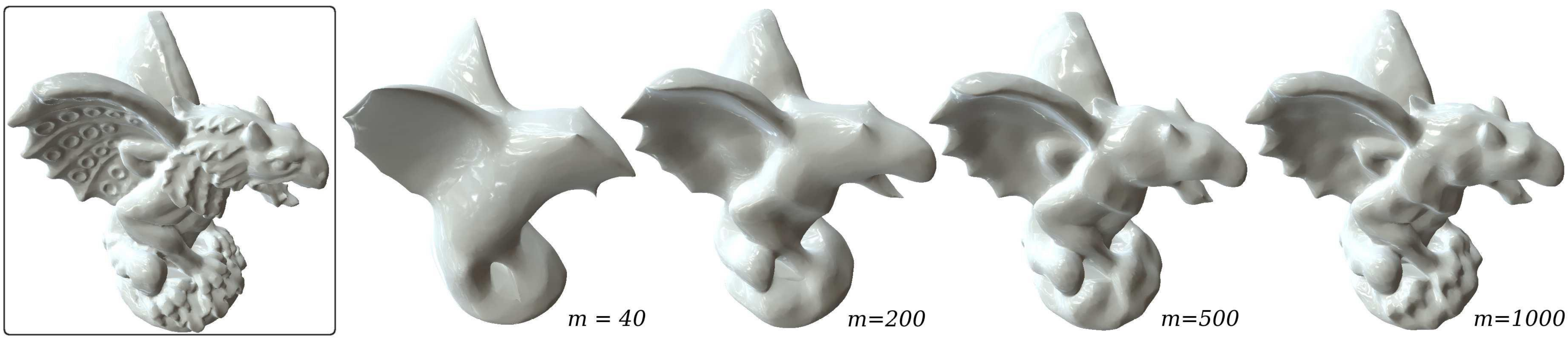
Spectral Analysis

- Visualization of the eigenvector of the Laplacian matrix

[Vallet et al., Eurographics 2008]

Spectral Analysis

- Smoothing using the Laplacian eigen-decomposition using the first m eigenvectors

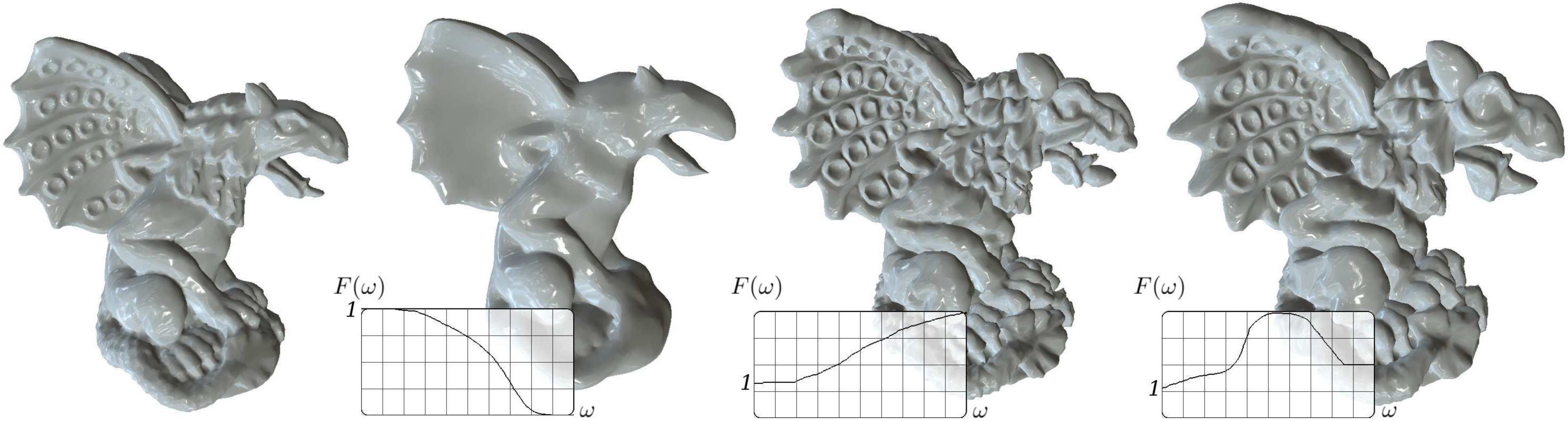


[Vallet et al., Eurographics 2008]

- The first functions captures the general shape of the functions and the next ones correspond to the details

Spectral Analysis

- Geometry filtering



[Vallet et al., Eurographics 2008]

Spectral Analysis

- Eigenvalues of Laplace matrix ≅ frequencies
- Low-pass filter ≅ reconstruction from eigenvectors associated with low frequencies
- Decomposition in frequency bands is used for mesh deformation
  - often too expensive for direct use in practice!
  - difficult to compute eigenvalues efficiently
- For smoothing apply diffusion

References

- Taubin, Gabriel. "A signal processing approach to fair surface design." *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM, 1995.

- Desbrun, Mathieu, et al. "Implicit fairing of irregular meshes using diffusion and curvature flow." *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 1999.

- Vallet, Bruno, and Bruno Lévy. "Spectral geometry processing with manifold harmonics." *Computer Graphics Forum*. Vol. 27. No. 2. Blackwell Publishing Ltd, 2008.