- Digital Geometry Processing
- Smoothing

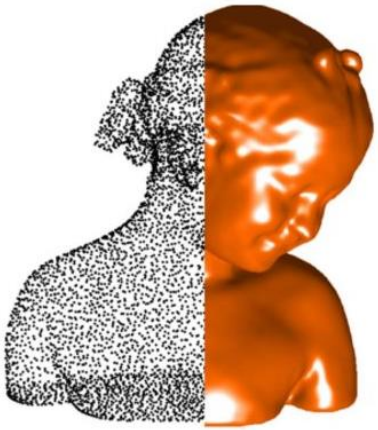
Outline

- Introduction to digital geometry processing
- Smoothing as signal processing (Taubin)
- Uniform weighting
- Cotangent weighting
- Smoothing as diffusion, Implicit smoothing

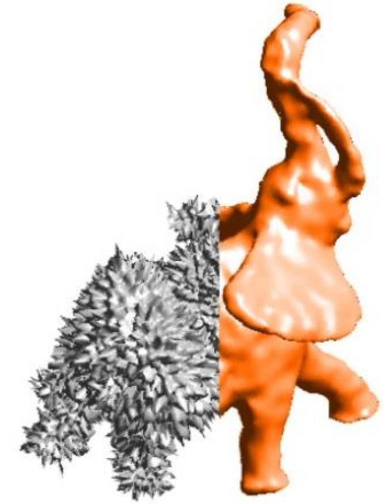
Geometry Processing

- Extend traditional digital signal processing (audio, video, etc.) to deal with geometric signals:
  - upsampling / downsampling / resampling / filtering / aliasing …
- They are basic building blocks for many areas/algorithms in graphics (rendering, animation, shape analysis, correspondence, etc)

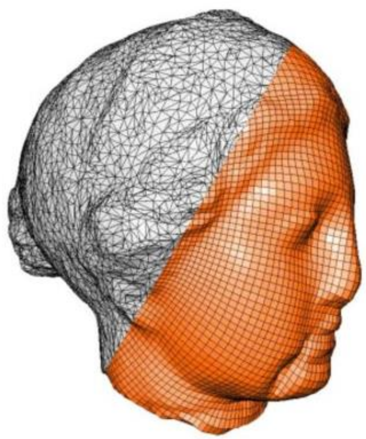
Geometry Processing Tasks



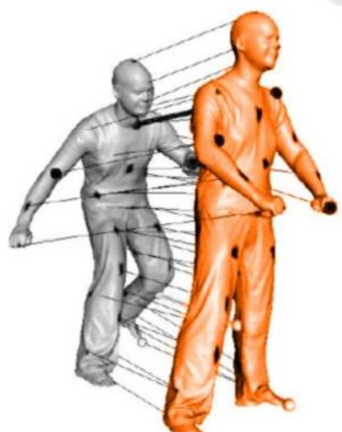
reconstruction



filtering

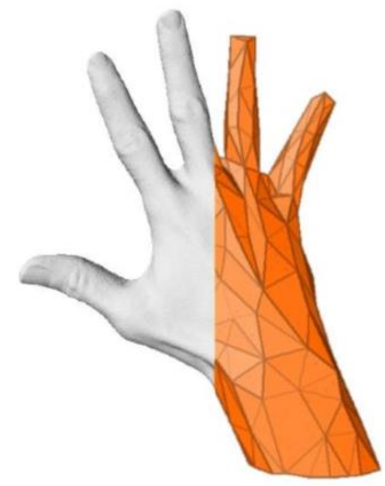


remeshing



shape analysis

parameterization



compression

Geometry Processing: Reconstruction

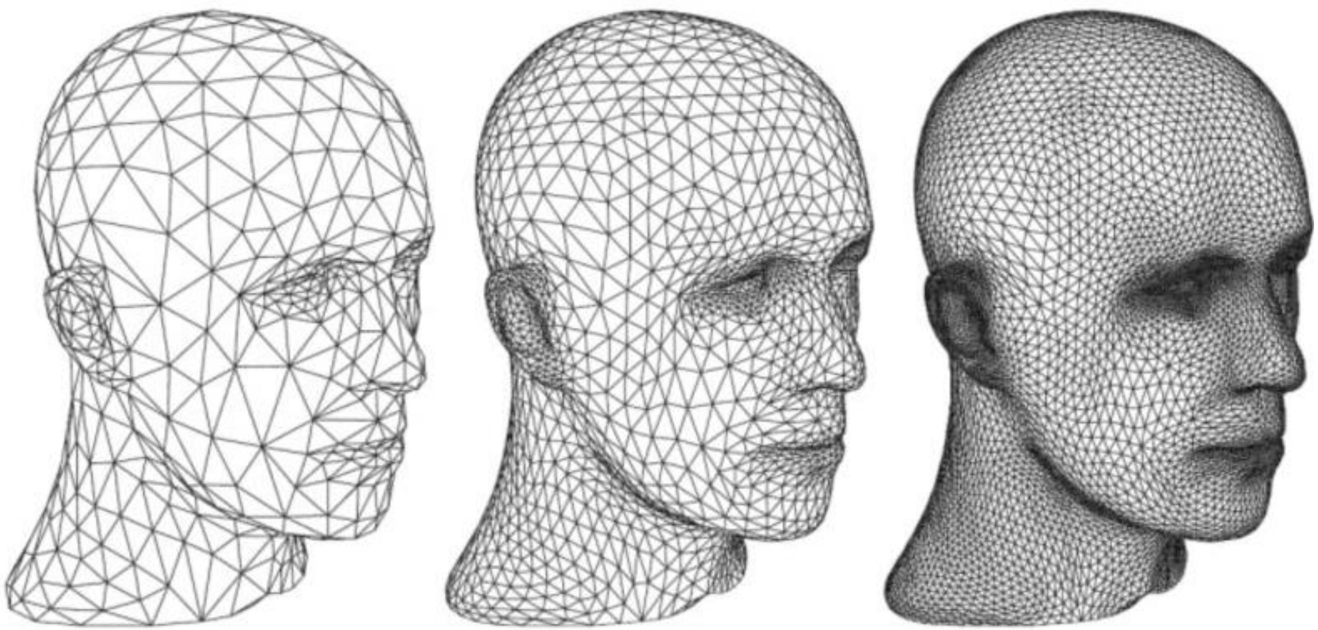
- Given samples of geometry, reconstruct surface

- What are samples? Many possibilities:
  - Points, points & normal, …
  - Image pairs / sets (multi-view stereo)
  - Line density integrals (MRI/CT scans)

- How do you get a surface? Many techniques:
  - Silhouette-based (visual hull)
  - Voronoi-based (e.g., power crust)
  - PDE-based (e.g., Poisson reconstruction)
  - Iso-surfacing (marching cubes)

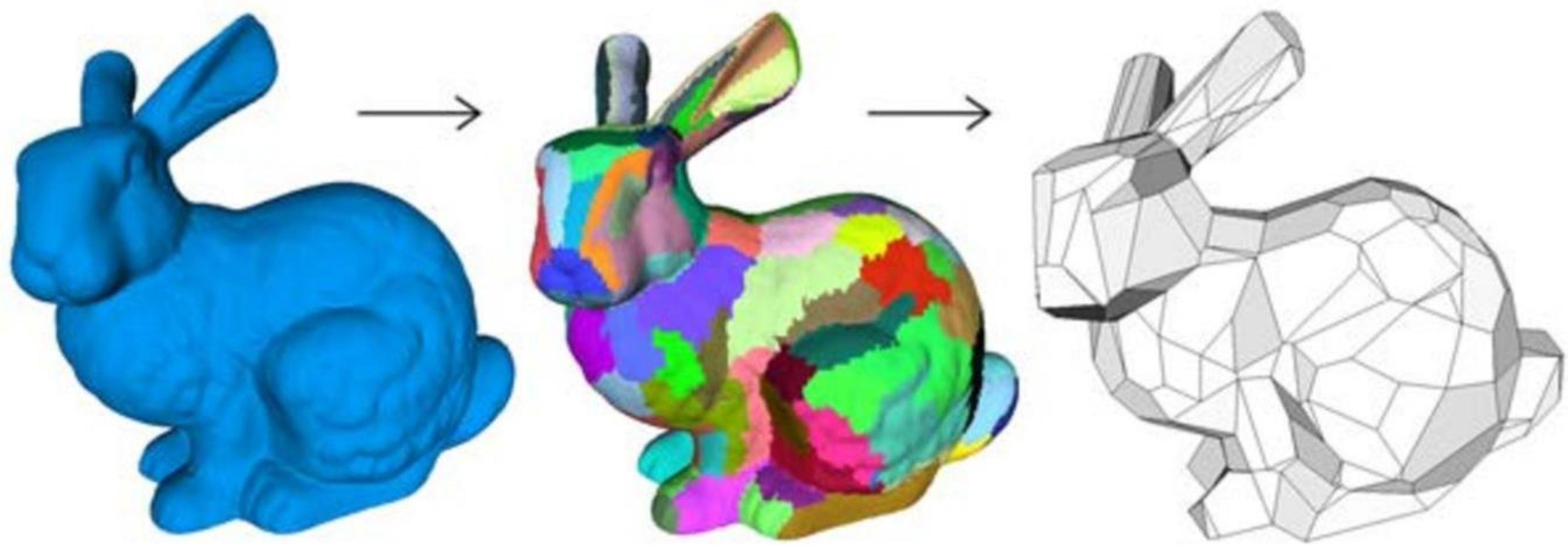
Geometry Processing: Upsampling

- Increase resolution via interpolation
- Images: e.g., bilinear, bicubic interpolation
- Polygon meshes:
  - Subdivision
  - Bilateral upsampling
  - …



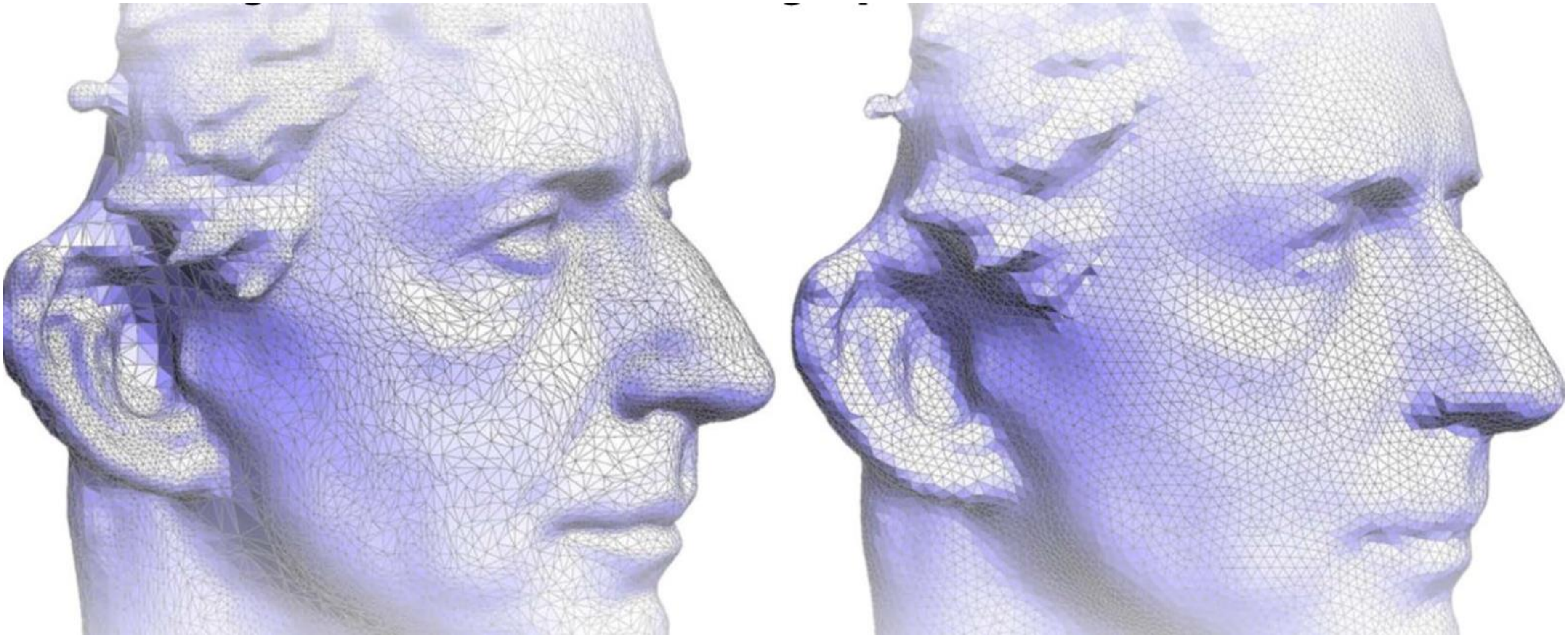
Geometry Processing: Downsampling

- Decrease resolution; try to preserve shape/appearance
- Images: nearest-neighbor, bilinear, bicubic interpolation
- Polygon meshes:
  - Iterative decimation, variational shape approximation, …



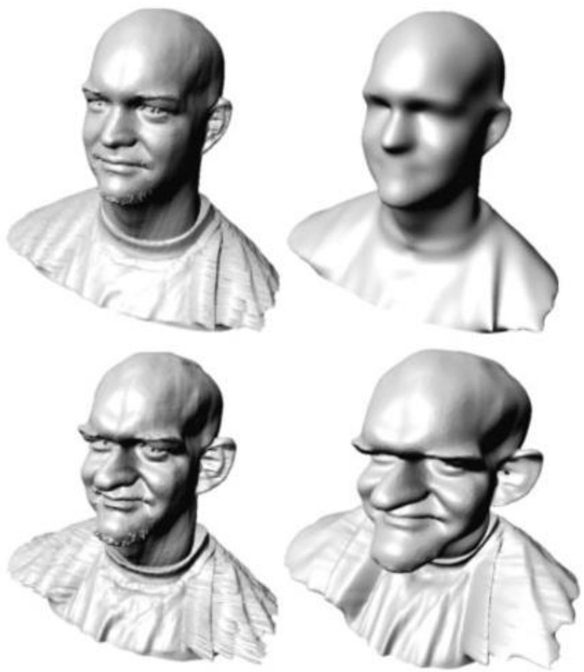
Geometry Processing: Resampling

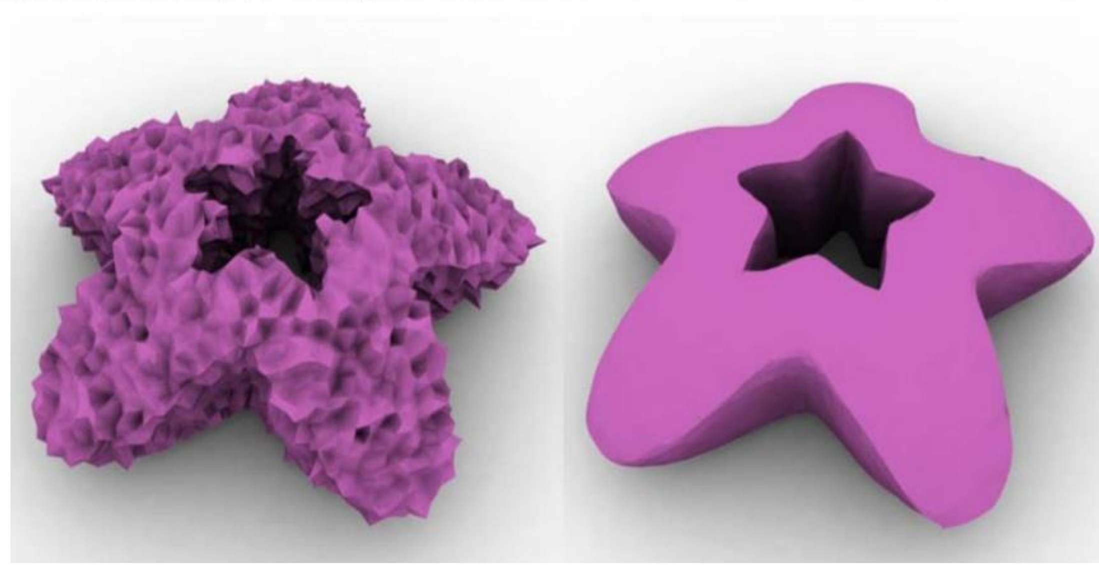
- Modify sample distribution to improve quality
- Images: not an issue! (pixels always stored on a regular grid)
- Meshes: shape of polygons is extremely important!
  - Different notion of "quality" depending on task
  - E.g., visualization vs. solving equations



Geometry Processing: Filtering

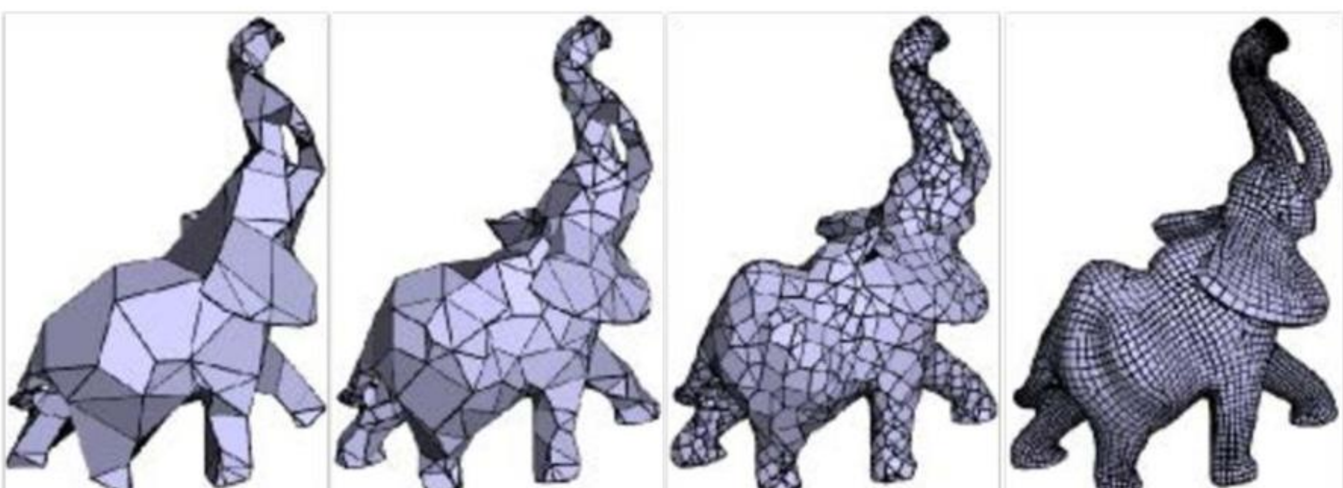
- Remove noise, or emphasize important features
- Images: blurring, bilatera; filter, edge detection, …
- Polygon meshes:
  - Curvature flow
  - Bilateral filter
  - Spectral filter





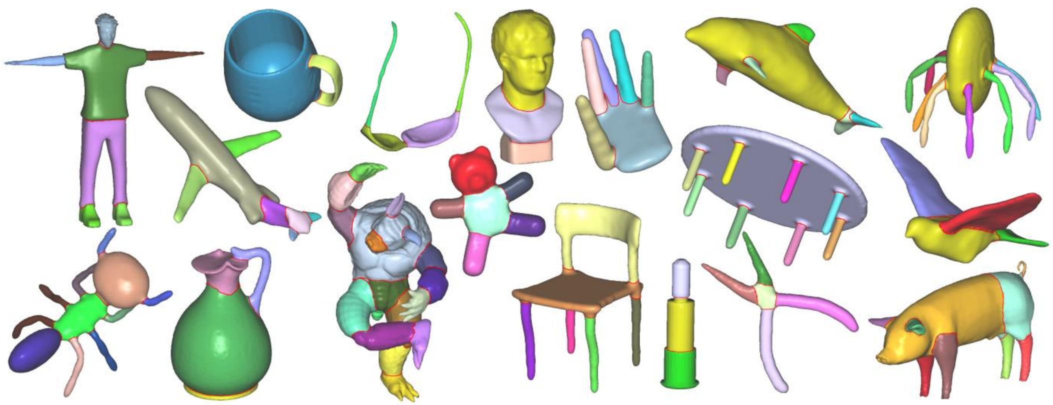
Geometry Processing: Compression

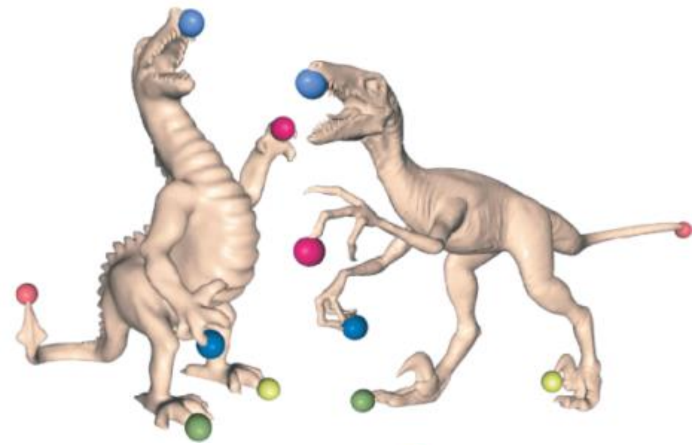
- Reduce storage size by eliminating redundant data/approximating unimportant data
- Images:
  - Run-length, Huffman coding – lossless
  - Cosine/wavelet (JPEG/MPEG) – lossy
- Polygon meshes:
  - Compress geometry and connectivity
  - Many techniques (lossy & lossless)

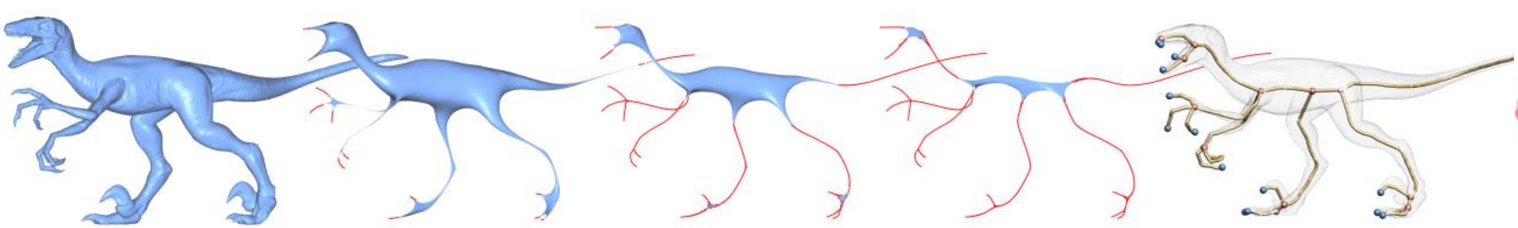


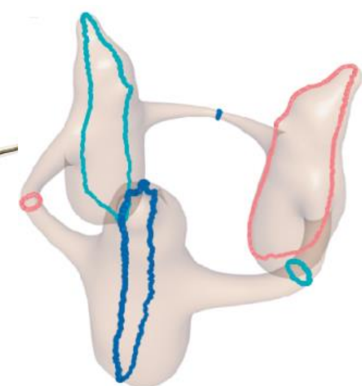
Geometry Processing: Shape Analysis

- Identify/understand important semantic features
- Images: computer vision, segmentation, face detection,…
- Polygon meshes:
  - Segmentation, correspondence, symmetry detection, skeletonization,….

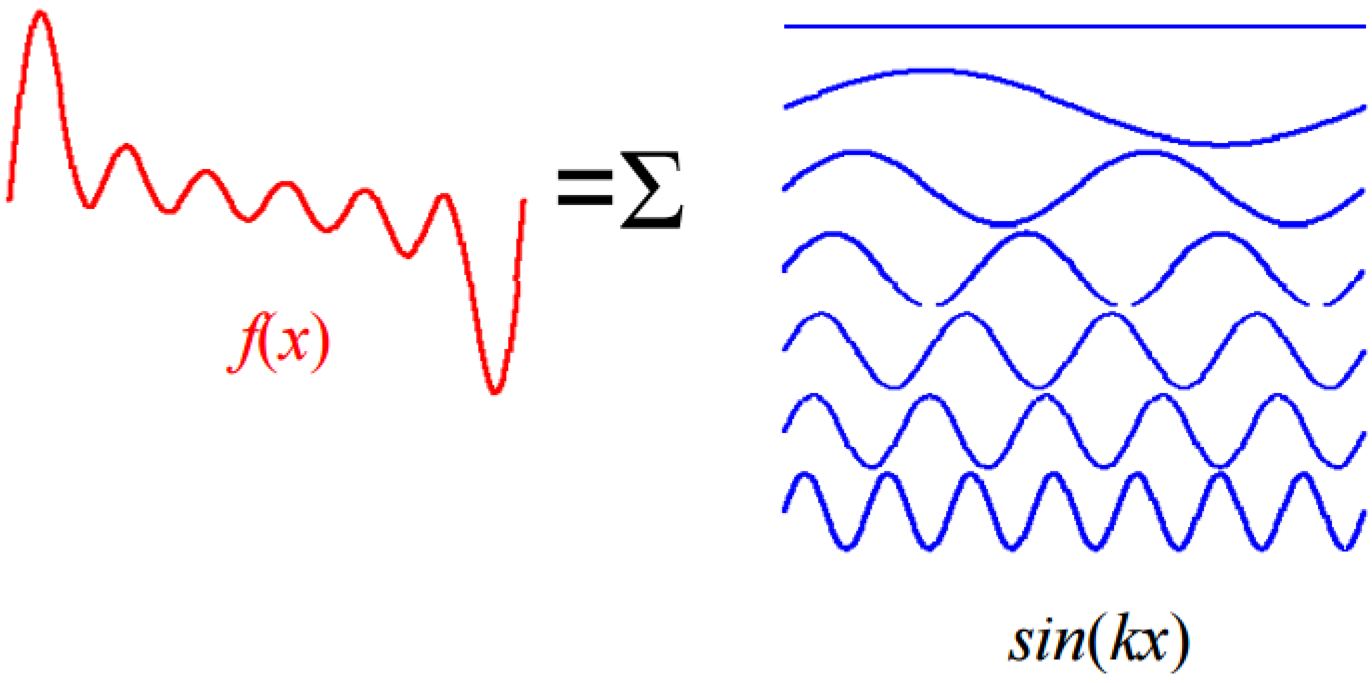






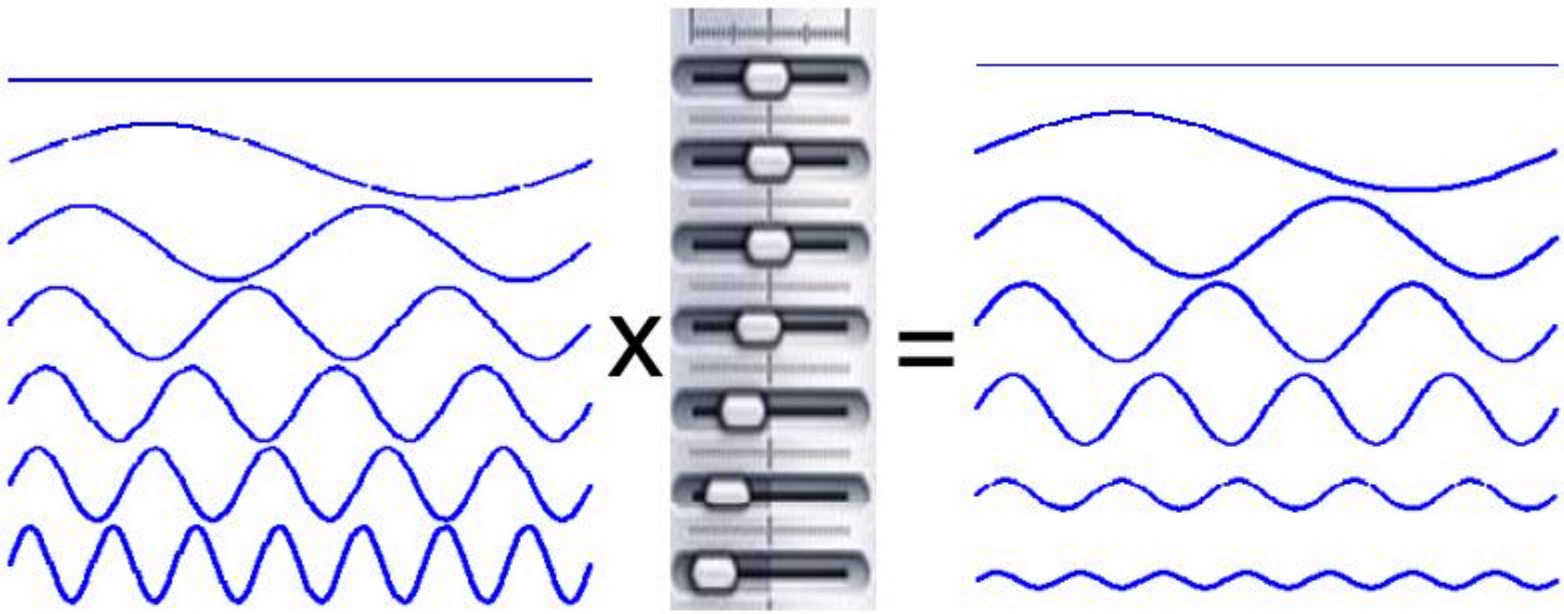


Smoothing as signal processing (Taubin 95)



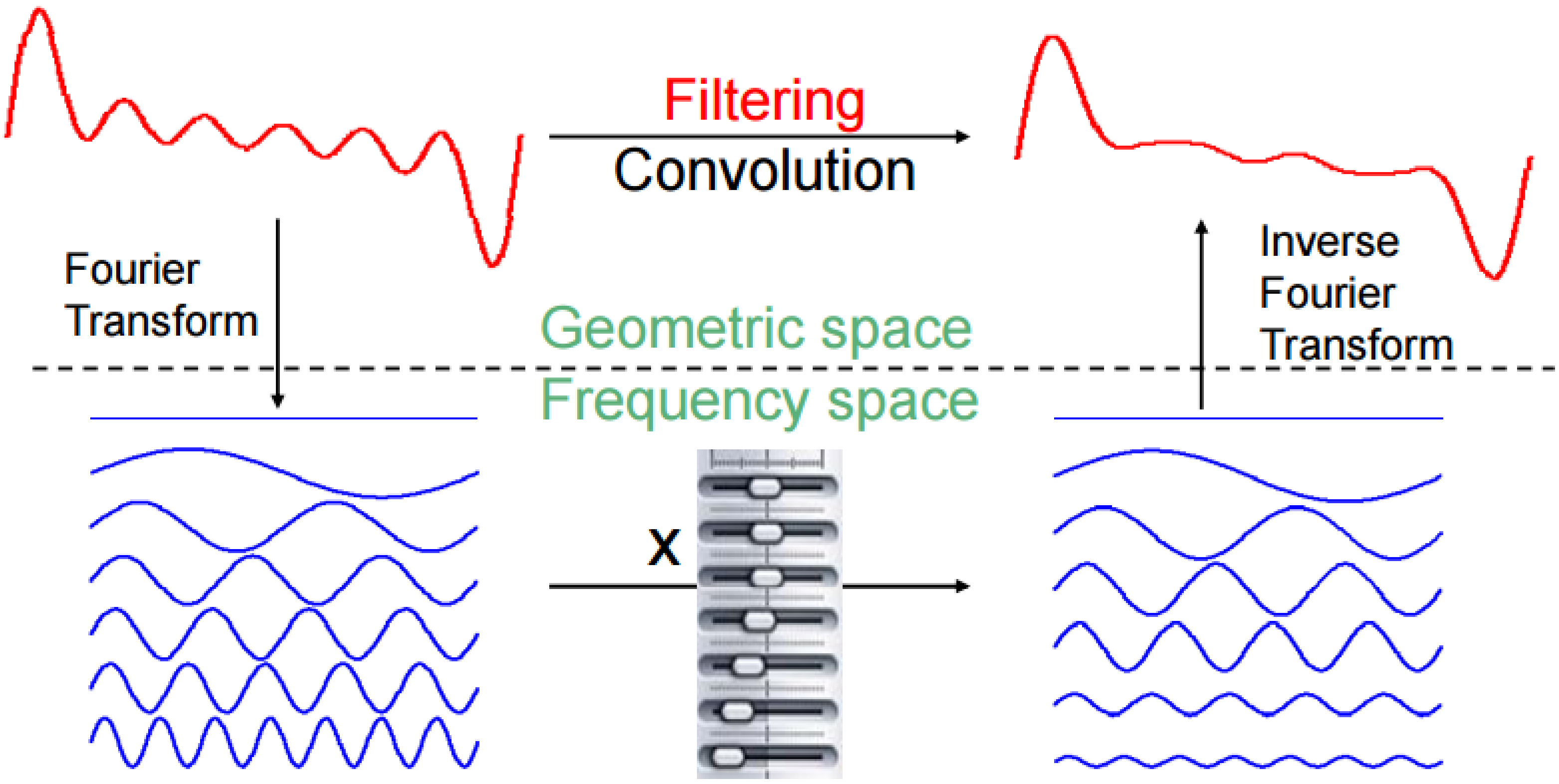
Fourier Transform

Smoothing as signal processing



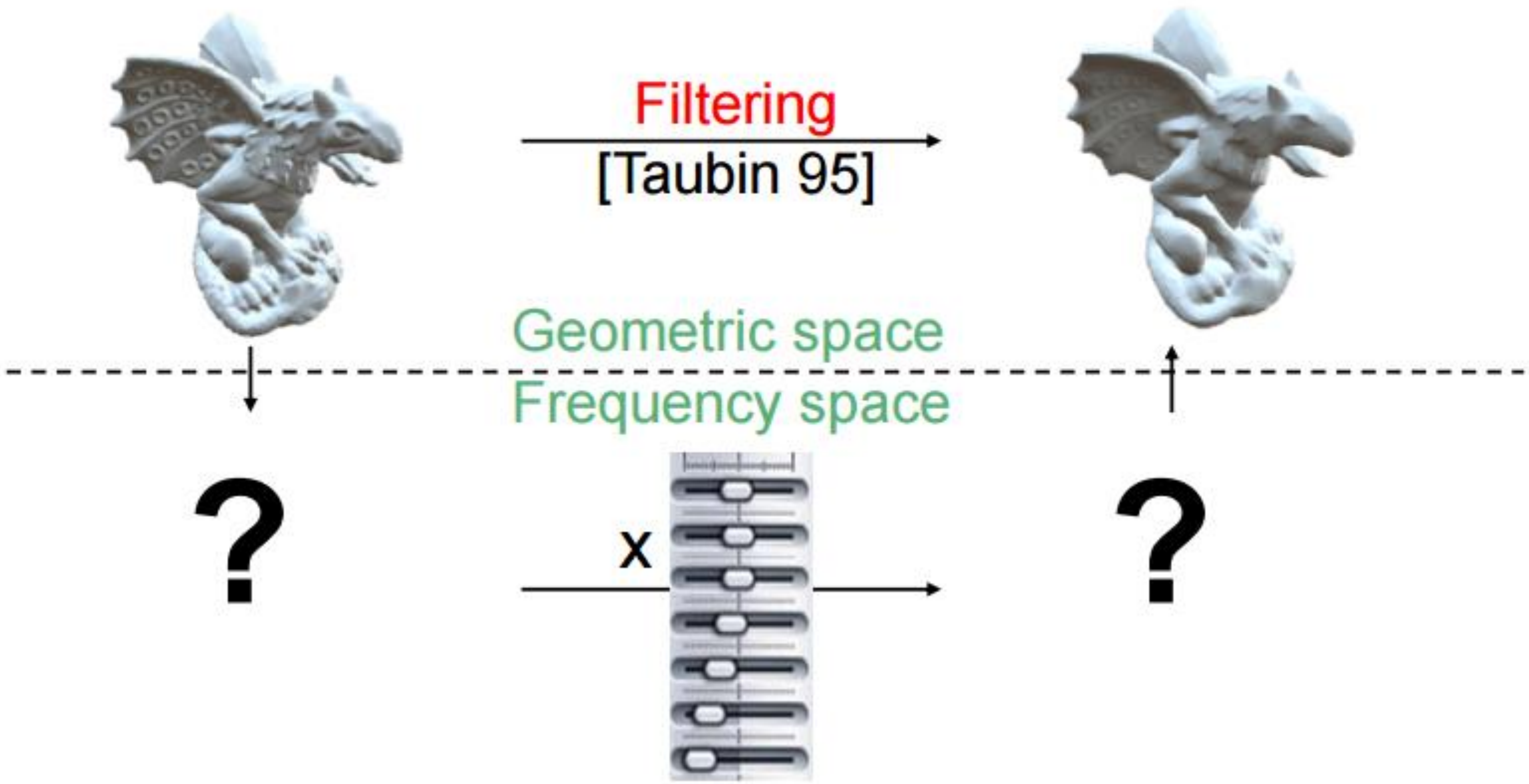
Fourier Transform

Smoothing as signal processing



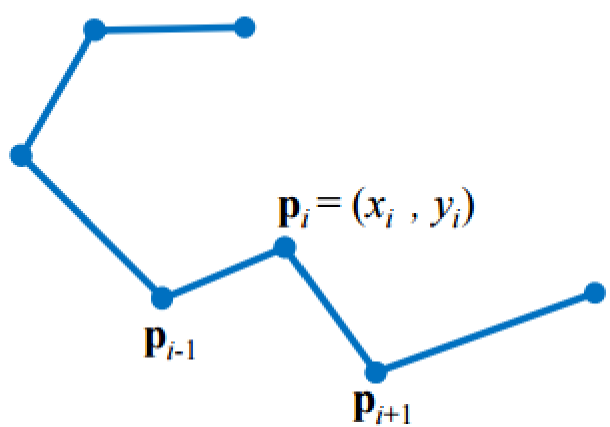
Fourier Transform

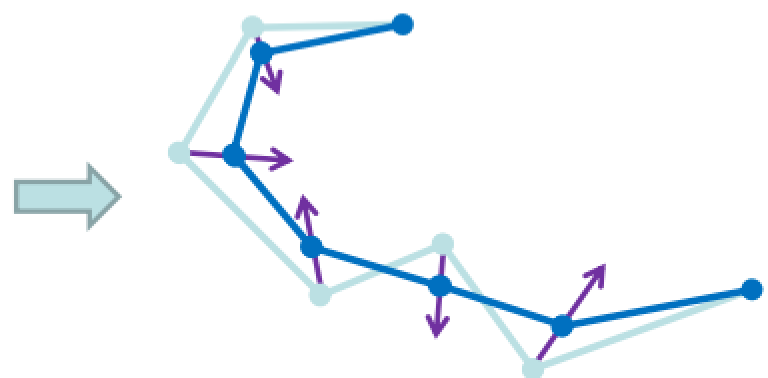
Signal processing on a Mesh



Laplacian Smoothing

Smoothing as a low-pass filter. Consider implemented as a convolution. For the case of curves:

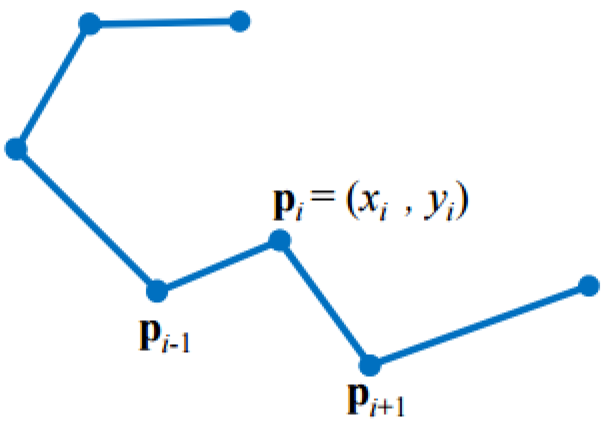


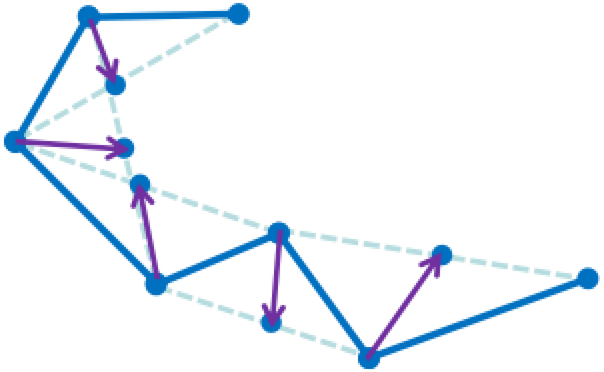


$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \frac{1}{2}\boxed{L(\mathbf{p}_i)}$$

Each vertex is displaced by half of its **1D discrete Laplacian**

Laplacian Smoothing





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

Finite difference discretization of second derivative

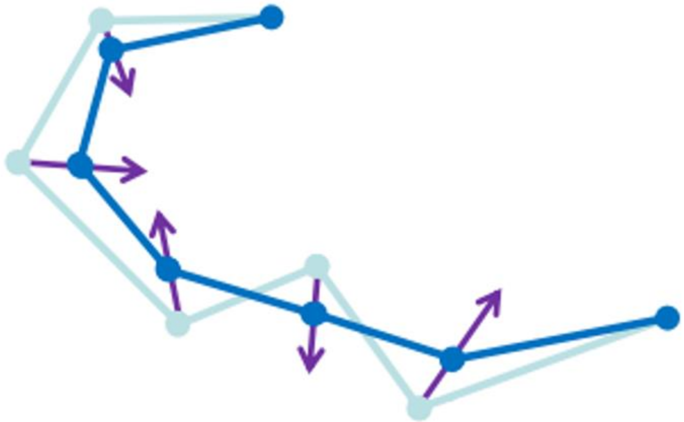
$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i)$$

Laplacian Smoothing

<u>Algorithm:</u>

Repeat for *m* iterations (for non boundary points):

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \lambda L(\mathbf{p}_i)$$



For what λ values?
$0 < \lambda < 1$

Closed curve converges to?
Single point

Spectral Analysis
Closed Curve

Re-write $\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} + \lambda L\left(\mathbf{p}_i^{(t)}\right)$

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

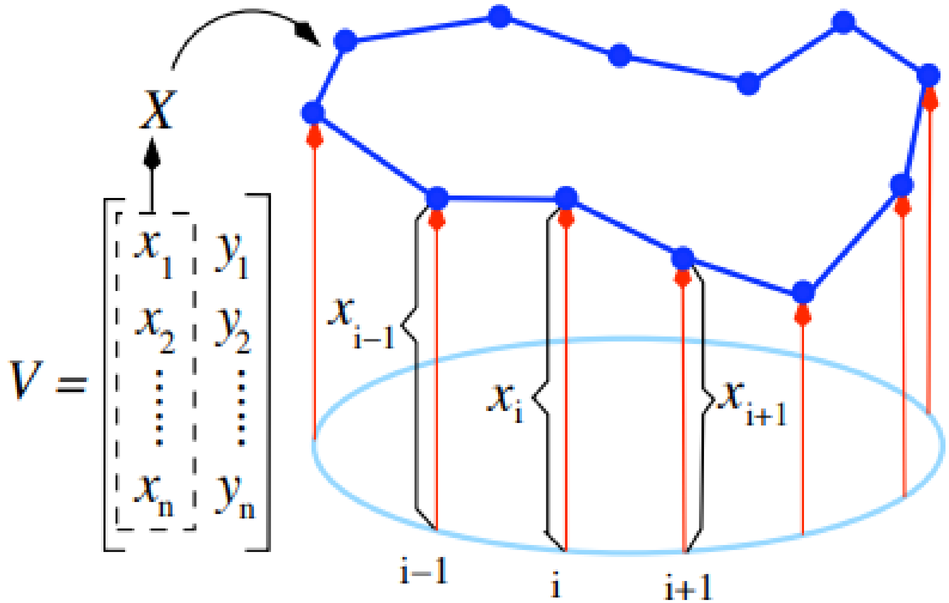
$$= \frac{1}{2}(\mathbf{p}_{i-1} - 2\,\mathbf{p}_i + \mathbf{p}_{i+1})$$

in matrix notation: $\mathbf{P}^{(t+1)} = \mathbf{P}^{(t)} - \lambda \mathbf{L}\mathbf{P}^{(t)}$

$$\mathbf{P} = \begin{pmatrix} x_1 & y_2 \\ \dots & \dots \\ x_n & y_n \end{pmatrix} \in \mathbb{R}^{n \times 2} \quad \mathbf{L} = \frac{1}{2}\begin{pmatrix} 2 & -1 & & & & -1 \\ -1 & 2 & -1 & & & \\ & & \dots & & & \\ & & & -1 & 2 & -1 \\ -1 & & & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{n \times n}$$

Sparse symmetric matrix

Spectral Mesh processing, Bruno Levy et al., SIGGRAPH Asia 2009 Course,



- The x-component of the coordinate vector V, i.e. X, can be viewed as a 1D periodic signal

Eigenvectors of L

- To analyze the behavior of Laplacian smoothing, we rely on the eigenvectors, which form a basis (since L is symmetric, it has real and orthogonal eigenvectors)

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

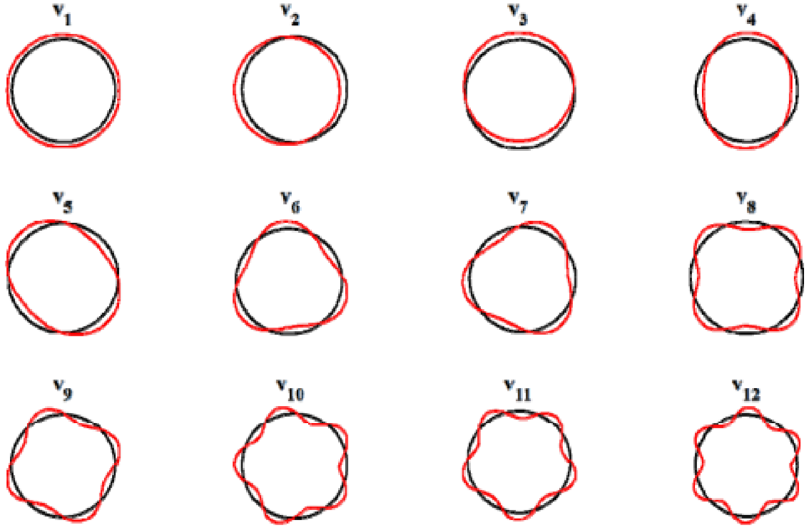
eigendecomposition

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

eigenvectors

eigenvalues

- Decompose signal into a linear combination of eigenvectors of Laplacian



Spectral Analysis

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

a sparse matrix multiplication
linear time complexity

After m iterations: $\mathbf{P}^{(m)} = (\mathbf{I} - \lambda \mathbf{L})^m \mathbf{P}^{(0)}$

Can be described using eigen-decomposition of $\mathbf{L}$

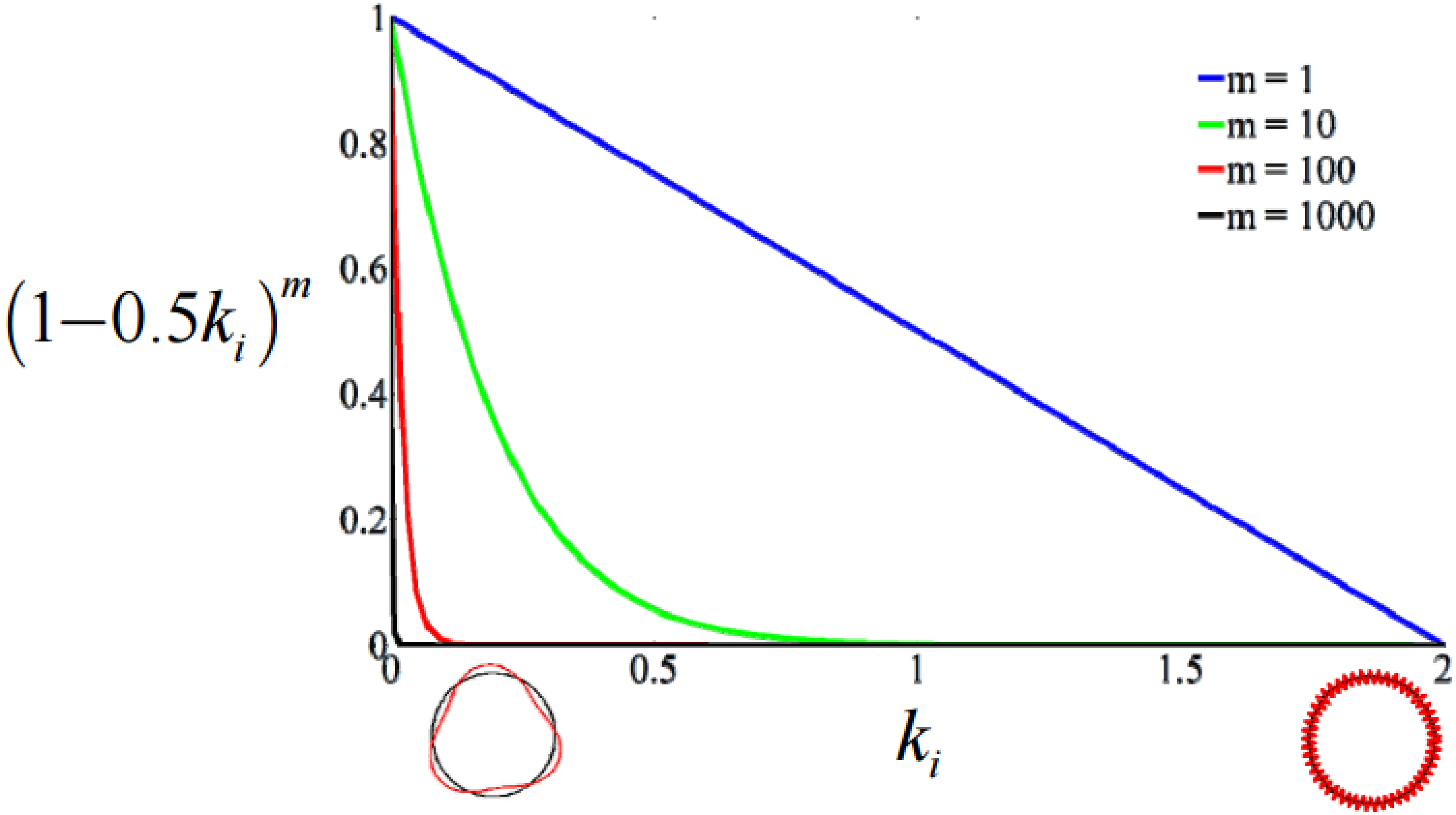
$$\mathbf{L} = \mathbf{V}\mathbf{D}\mathbf{V}^T$$

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

$$\Rightarrow \quad \mathbf{P}^{(m)} = \mathbf{V} \boxed{(\mathbf{I} - \lambda \mathbf{D})^m} \mathbf{V}^T \mathbf{P}^{(0)}$$

Filtering high frequencies

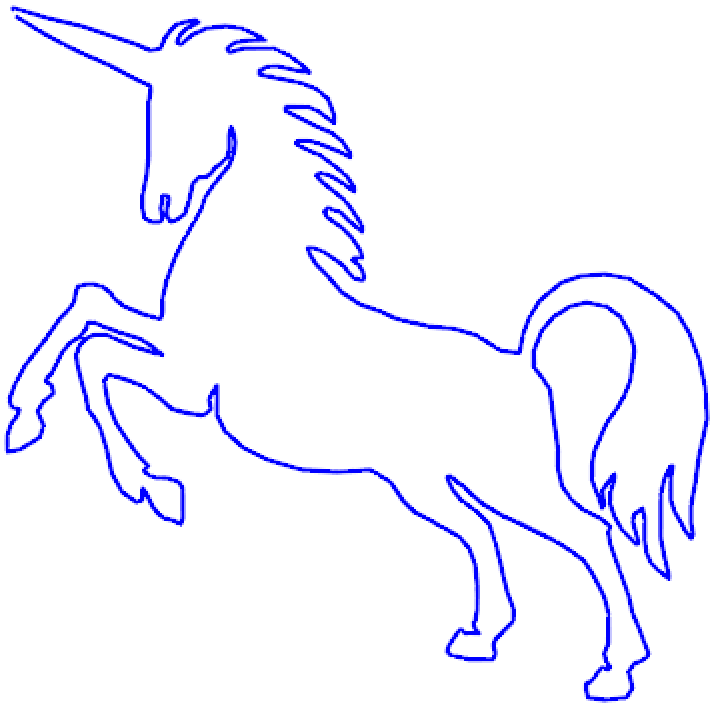
Spectral Analysis
Laplacian Smoothing



more iterations in the end shrink to nothing

Laplacian smoothing

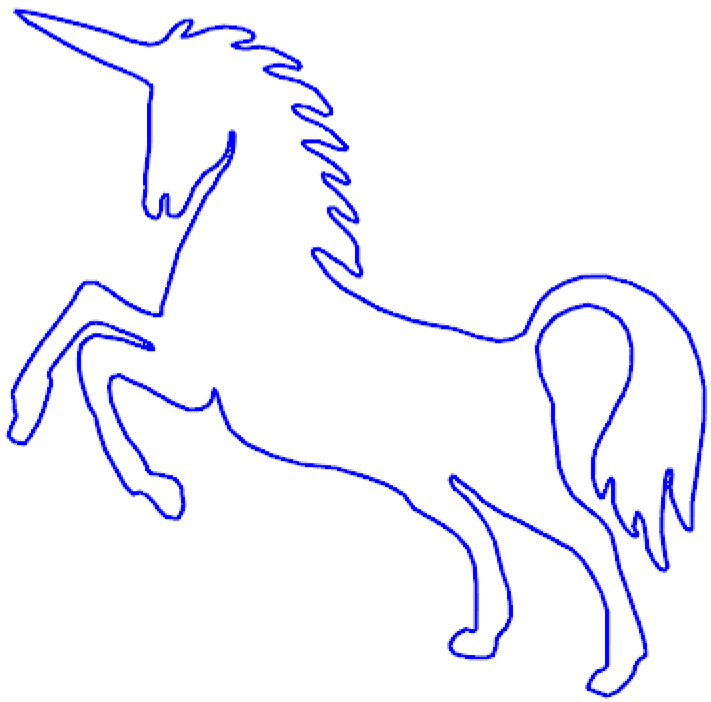
2D Curve – Example



Original curve

Laplacian smoothing

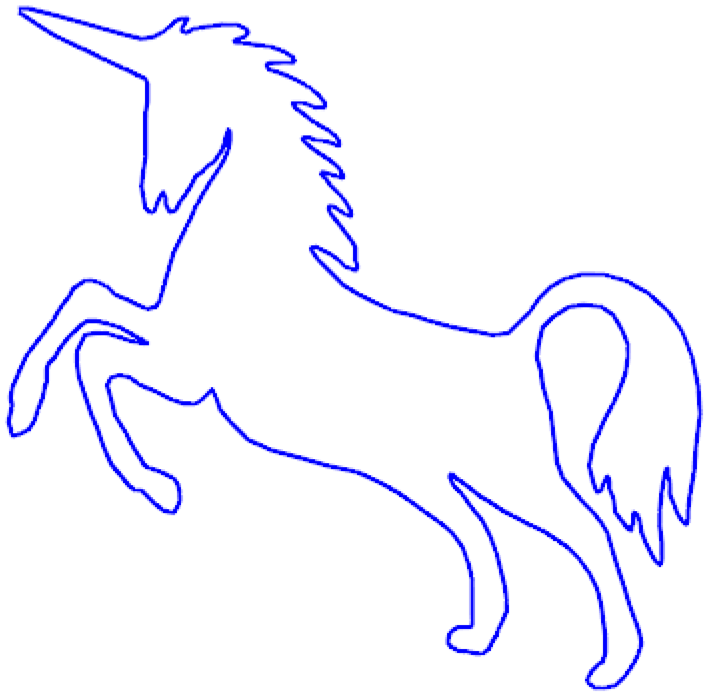
2D Curve – Example



1st iteration; $\lambda=0.5$

Laplacian smoothing

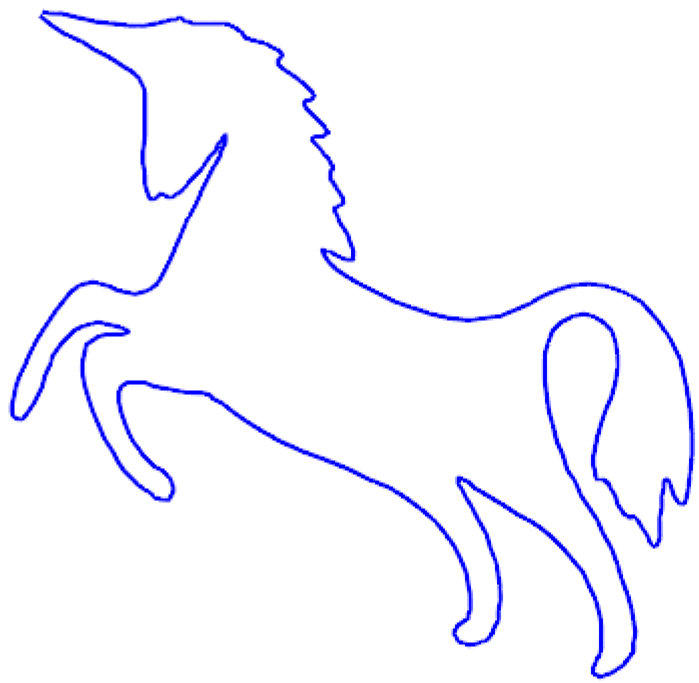
2D Curve – Example



2nd iteration; $\lambda=0.5$

Laplacian smoothing

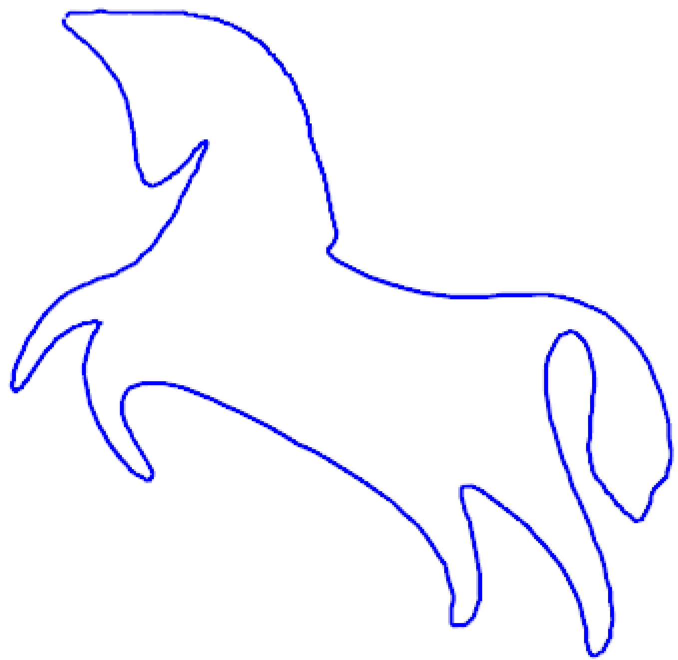
2D Curve – Example



8th iteration; $\lambda=0.5$

Laplacian smoothing

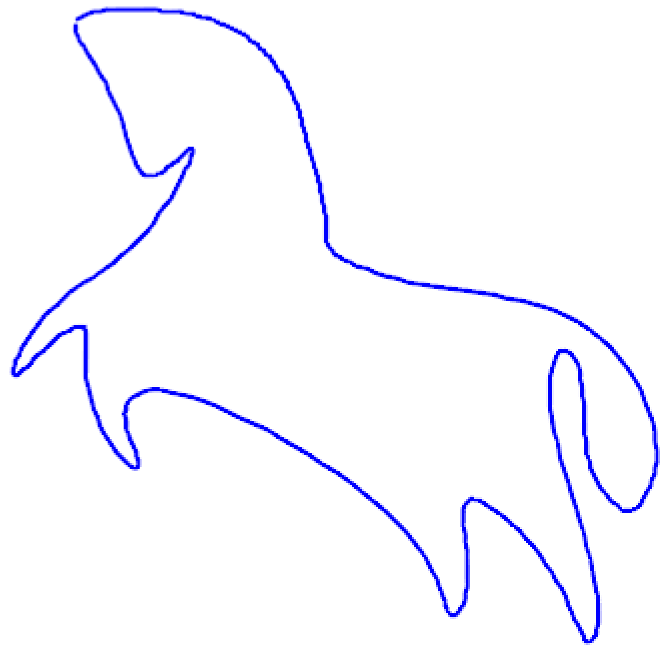
2D Curve – Example



27th iteration; $\lambda=0.5$

Laplacian smoothing

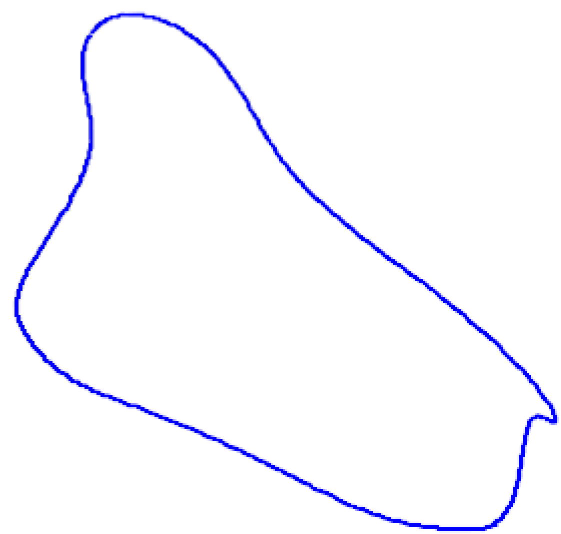
2D Curve – Example



50th iteration; λ=0.5

Laplacian smoothing

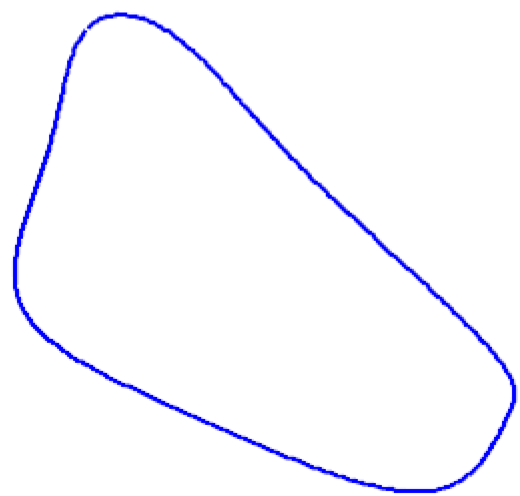
2D Curve – Example



500th iteration; $\lambda=0.5$

Laplacian smoothing

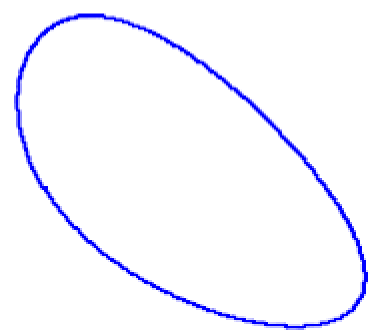
2D Curve – Example



1000th iteration; $\lambda=0.5$

Laplacian smoothing

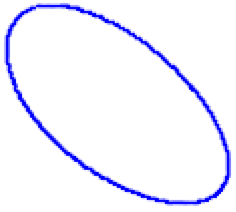
2D Curve – Example



5000th iteration; $\lambda=0.5$

Laplacian smoothing

2D Curve – Example



10000th iteration; λ=0.5

Laplacian smoothing

2D Curve – Example

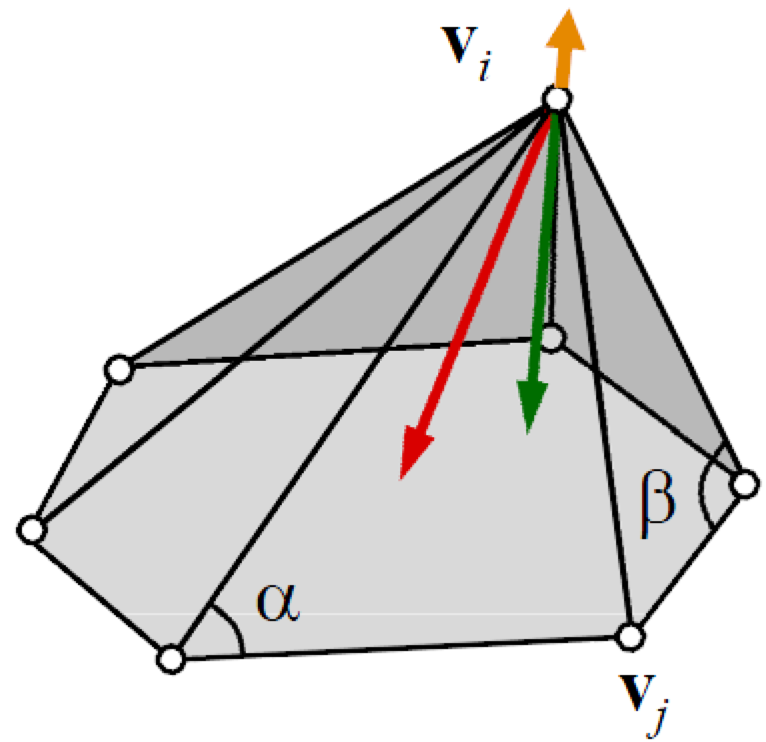
50000th iteration; $\lambda=0.5$

Discrete Laplace-Beltrami

- Uniform discretization: $L(\mathbf{v})$ or $\Delta\mathbf{v}$

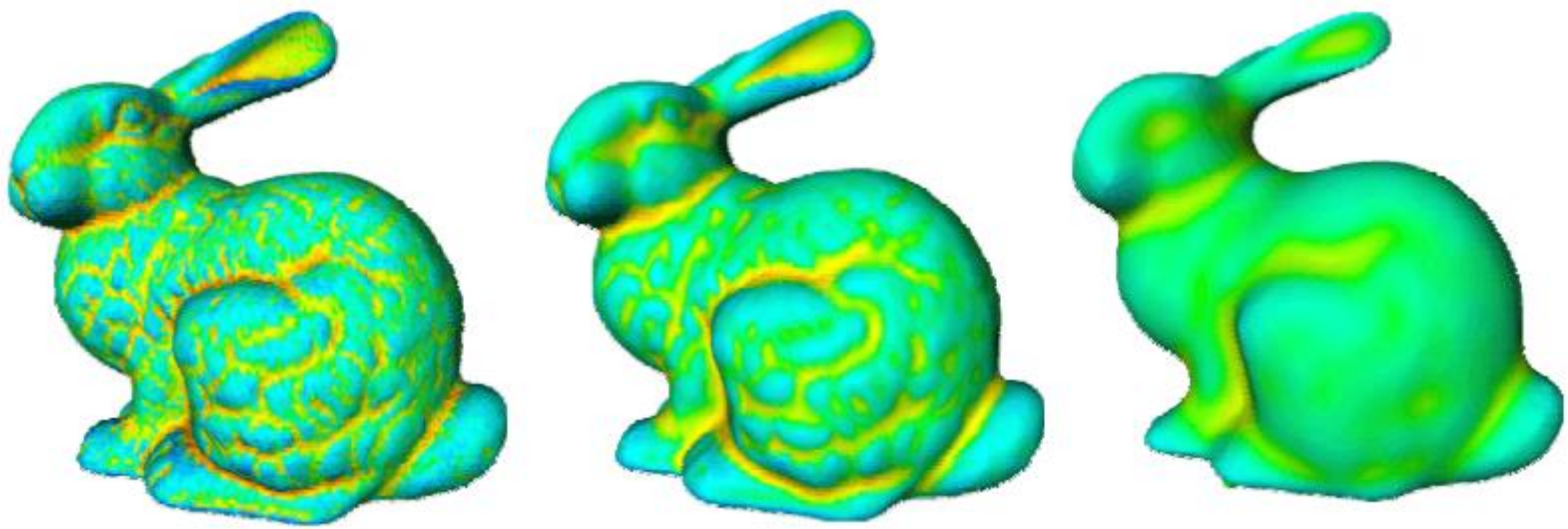
$$L_u(\mathbf{v}_i) = \frac{1}{|N_1(\mathbf{v}_i)|} \sum_{\mathbf{v}_j \in N_1(\mathbf{v}_i)} (\mathbf{v}_j - \mathbf{v}_i) \quad \text{where} \quad |N_1(\mathbf{v}_i)| = d_i$$

$$= \frac{1}{|N_1(\mathbf{v}_i)|} \left(\sum_{\mathbf{v}_j \in N_1(\mathbf{v}_i)} \right) - \mathbf{v}_i$$



- Uniform discretization is known as umbrella operator
- Always points to the centroid of base polygon (red vector)
- Uniform discretization gives bad approximation of Laplacian for irregular triangulations

Laplacian Smoothing on Meshes



0 Iterations 5 Iterations 20 Iterations

Problem - Shrinkage

Repeated iterations of Laplacian smoothing shrinks the mesh

original 3 steps 6 steps 18 steps original

Taubin Smoothing

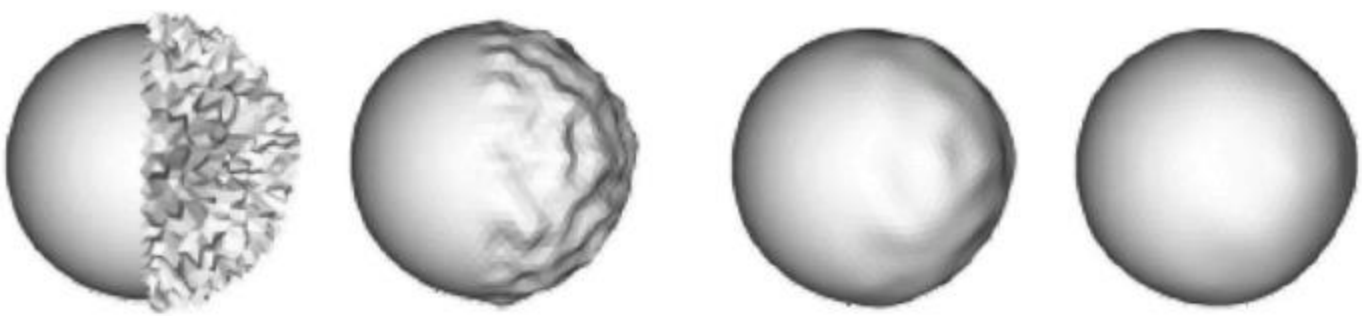
Shrinkage problem is remedied with an inflation term

Iterate:

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \lambda \Delta \mathbf{p}_i \qquad \text{Shrink}$$

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \mu \Delta \mathbf{p}_i \qquad \text{Inflate}$$

with $\lambda > 0$ and $\mu < 0$ and $|\mu| > \lambda$

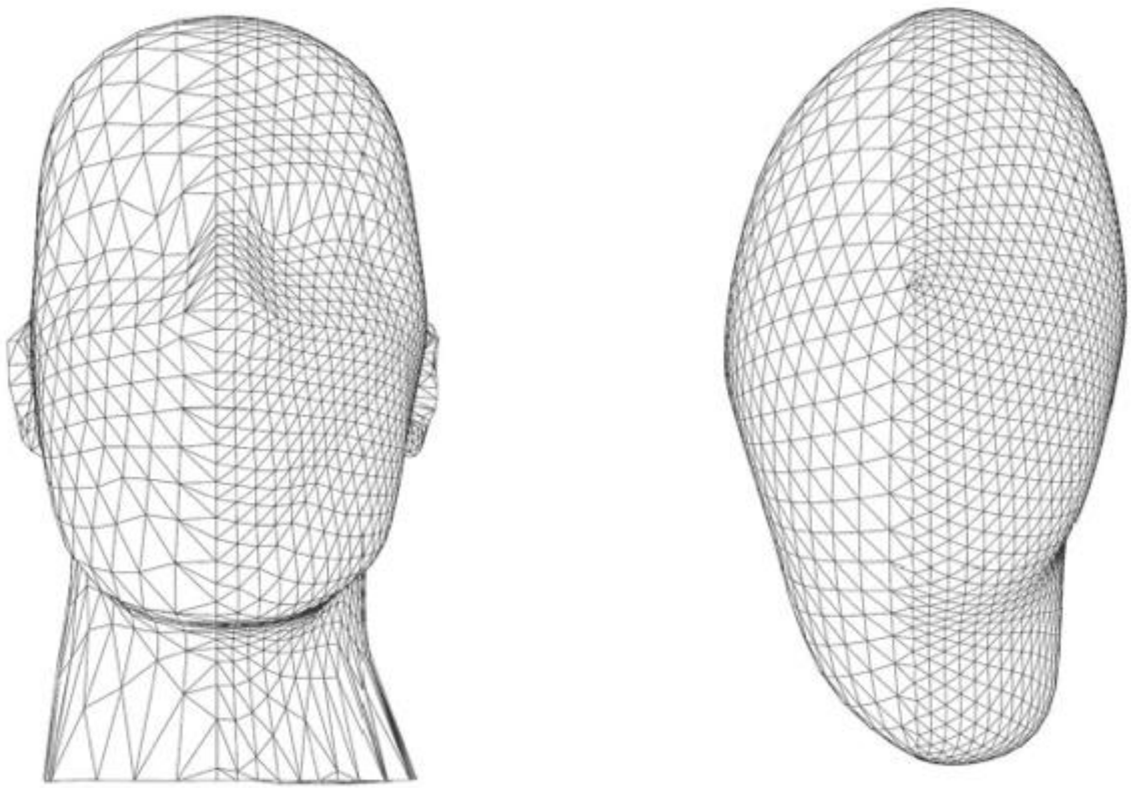


original 10 steps 50 steps 200 steps

From Taubin, Siggraph 1995

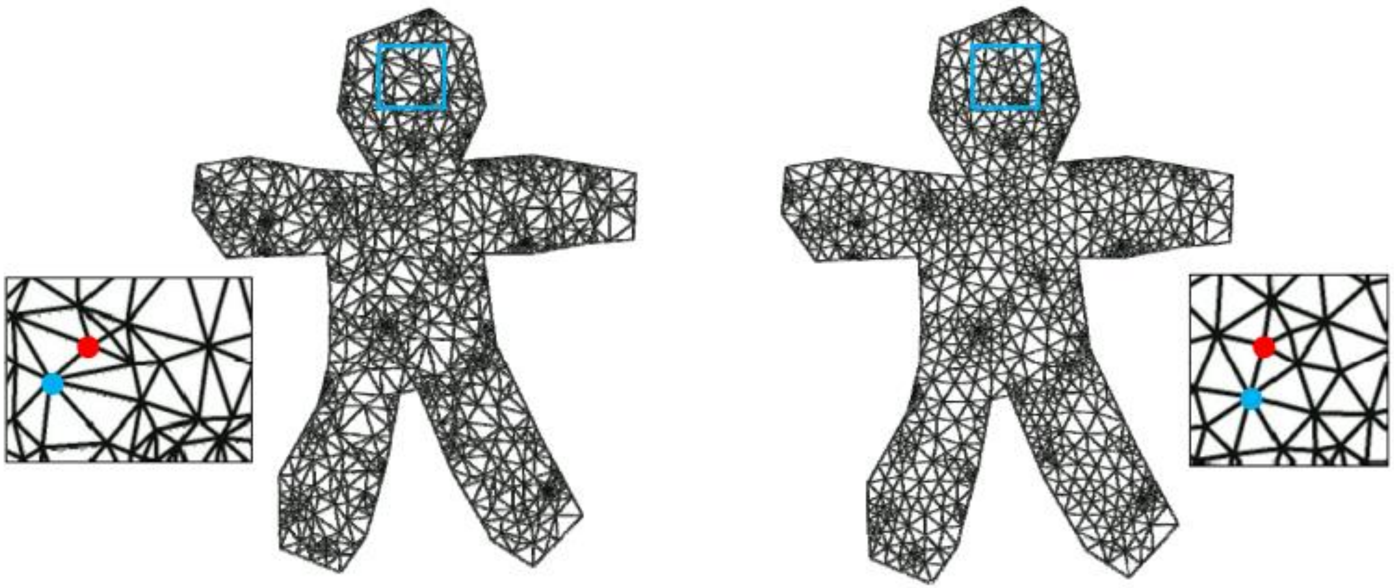
Laplace Operator discretization

Problem: Uniform weighting leads to distortion for meshes with irregular sampling.



Laplace Operator Discretization

Sanity check – what should happen if the mesh lies in the plane: $\mathbf{p}_i = (x_i, y_i, 0)$?



0 Iterations

5 Iterations

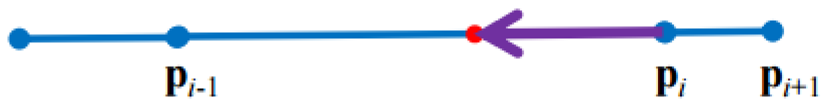
Not good – A flat mesh is smooth, should stay the same after smoothing

Laplace Operator Discretization

What went wrong?

Back to curves:

$$\boxed{\frac{1}{2}}\left(\mathbf{p}_{i+1}+\mathbf{p}_{i-1}\right)-\mathbf{p}_i$$

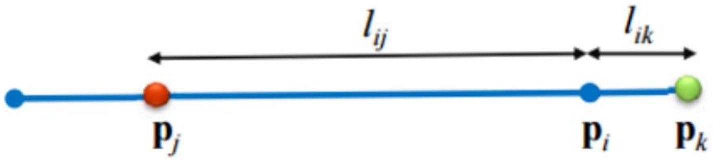


Same weight for both neighbors, although one is closer

Laplace Operator Discretization

The Solution (curve case)

Solution: use a smarter weight instead



Which weights?

$$w_{ij} = \frac{1}{l_{ij}} \qquad w_{ik} = \frac{1}{l_{ik}}$$

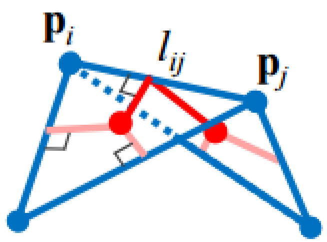
$$L(\mathbf{p}_i) = \frac{w_{ij}\mathbf{p}_j + w_{ik}\mathbf{p}_k}{w_{ij} + w_{ik}} - \mathbf{p}_i$$

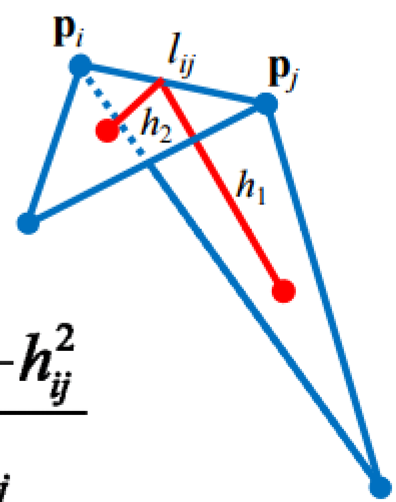
Straight 1D curves will be invariant to smoothing with this inverse distance weighting, i.e., $L(\mathbf{p}_i) = 0$.

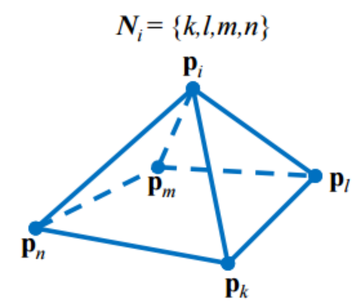
Laplace Operator Discretization
Cotangent Weights

Same idea extends to 2D triangular surfaces:

$$w_{ij} = \frac{1}{l_{ij}} \text{ in 1D is replaced as follows in 2D:}$$

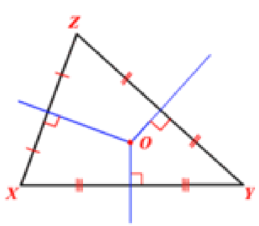






$$w_{ij} = \frac{h_{ij}^1 + h_{ij}^2}{l_{ij}}$$

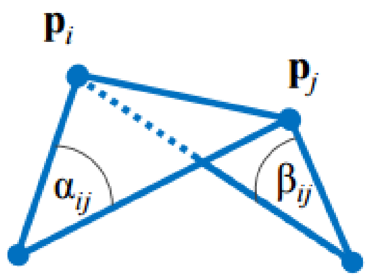
$$L(\mathbf{p}_i) = \frac{1}{\sum_{j \in N_i} w_{ij}} \left(\sum_{j \in N_i} w_{ij} \mathbf{p}_j \right) - \mathbf{p}_i$$



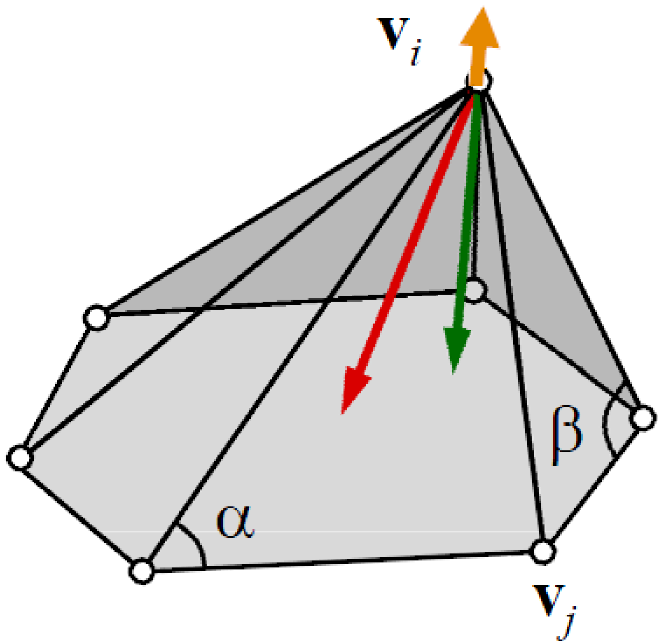
Here, O is the circumcenter of ΔXYZ .

Laplace Operator Discretization

This is called **cotangent weighting**



$$w_{ij} = \frac{h_{ij}^1 + h_{ij}^2}{l_{ij}} = \frac{1}{2}\left(\cot\alpha_{ij} + \cot\beta_{ij}\right)$$



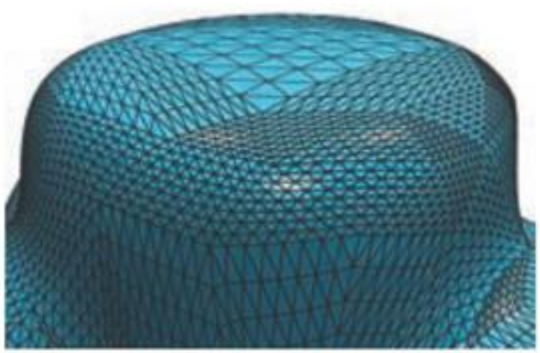
Planar meshes will be invariant to smoothing

With cotangent weights

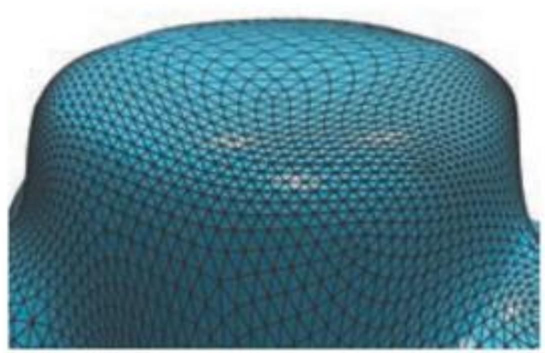
- The resulting Laplacian (green) is a good approximation of the normal in opposite direction (amber)
- magnitude is twice the mean curvature.

Thus also known as **curvature flow smoothing**.

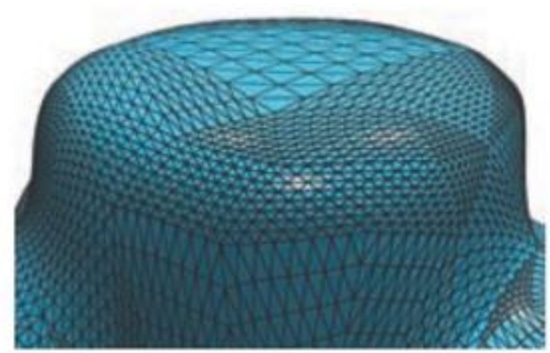
Smoothing with Cotangent weights



original

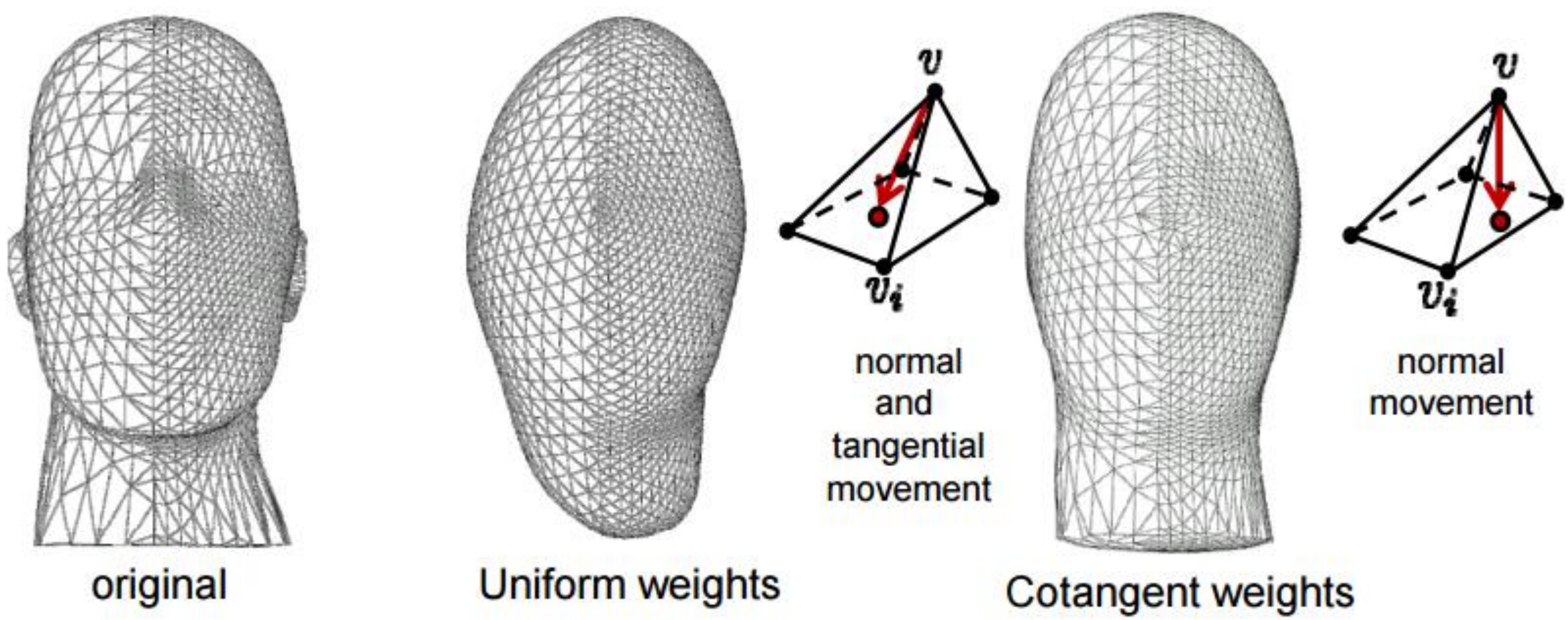


Uniform weights



Cotan weights

Smoothing with the Cotangent Laplacian



points always in centroids (original geometry ruined)

geometry-aware

From Desbrun et al., Siggraph 1999

Smoothing as diffusion

[Implicit fairing of irregular meshes using diffusion and curvature flow, Desbrun et al/ SIG 99]

- Model smoothing as a diffusion process

$$\frac{\partial \mathbf{x}}{\partial t} = \lambda L(\mathbf{x})$$

- Integrate over time will smooth the high frequencies

$$\frac{\mathbf{x}^{n+1} - \mathbf{x}^{n}}{\mathrm{d}t} = \lambda L\,\mathbf{x}^{n}$$

$$\mathbf{x}^{n+1} - \mathbf{x}^{n} = \lambda \mathrm{d}t\, L\,\mathbf{x}^{n}$$

$$\mathbf{x}^{n+1} = (I + \lambda \mathrm{d}t\, L)\,\mathbf{x}^{n}$$

explicit integration

Scale λ by simulation parameter time t

Taubin arrived at the same equation via signal processing

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

Implicit Smoothing

- Forward Euler integration

$$\mathbf{x}^{n+1} = (I + \lambda\, \mathrm{d}t\, L)\, \mathbf{x}^{n}$$

explicit integration

involves a matrix-vector multiplication in each iteration

Con:Forward Euler is not numerically stable. Can take only very small time steps

- Backward Euler for unconditional stability

$$\mathbf{x}^{n+1} = \mathbf{x}^{n} + \lambda\, \mathrm{d}t\, L(\mathbf{x}^{n+1})$$

implicit integration
(approximate derivative using the new mesh)

$$(I - \lambda\, \mathrm{d}t\, \mathrm{L})\mathbf{x}^{n+1} = \mathbf{x}^{n}$$

Solve a sparse linear system in each iteration

Pro: unconditional stability; can take larger time steps.

Implicit fairing of irregular meshes using diffusion and curvature flow
M. Desbrun, M. Meyer, P. Schroeder, A. Barr
ACM SIGGRAPH 99

Implicit fairing

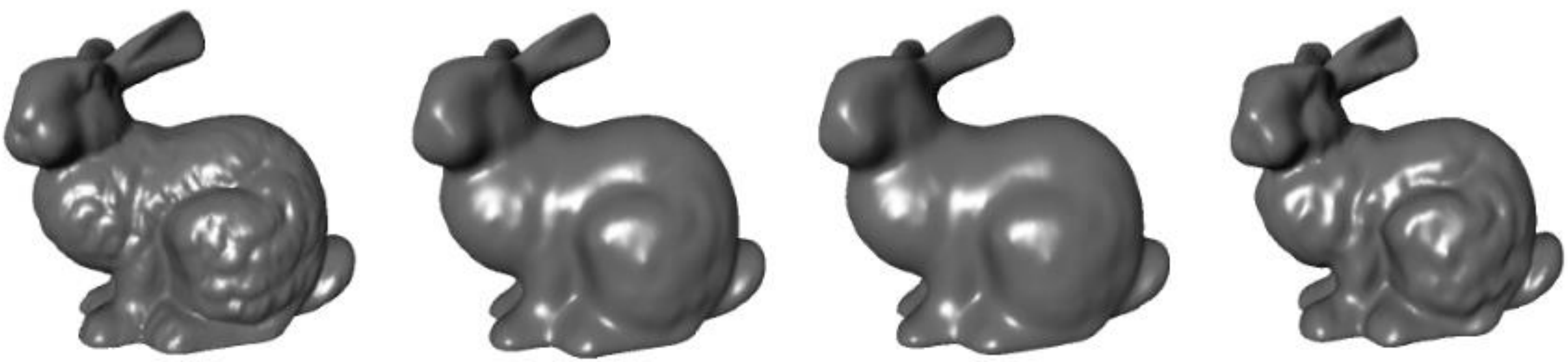


Figure 4: *Stanford bunnies: (a) The original mesh, (b) 10 explicit integrations with* $\lambda dt = 1$, *(c) 1 implicit integration with* $\lambda dt = 10$ *that takes only 7 PBCG iterations (30% faster), and (d) 20 passes of the* $\lambda|\mu$ *algorithm, with* $\lambda = 0.6307$ *and* $\mu = -0.6732$. *The implicit integration results in better smoothing than the explicit one for the same, or often less, computing time. If volume preservation is called for, our technique then requires many fewer iterations to smooth the mesh than the* $\lambda|\mu$ *algorithm.*

References

- “A Signal Processing Approach to Fair Surface Design”, Taubin, Siggraph ‘95
- “Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow”, Desbrun et al., Siggraph ’99