CENG 789 – Digital Geometry Processing

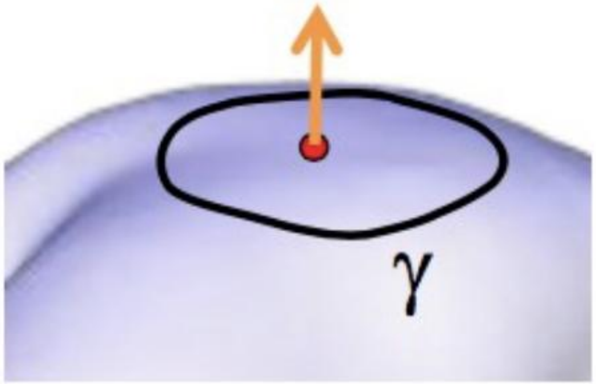
12- Laplacian with Applications in Geometry Processing

Prof. Dr. Yusuf Sahillioğlu

Computer Eng. Dept., MIDDLE EAST TECHNICAL UNIVERSITY, Turkey

Laplacian

- ✓ Continuous Laplace-Beltrami operator is approximated by the discrete Graph Laplacian.
- ✓ Customary to call Laplace-Beltrami operator as a Laplacian.
- ✓ Linking geometry of manifold to the heat flow.
- ✓ Intuition: encodes deviation of a point from its neighbors.

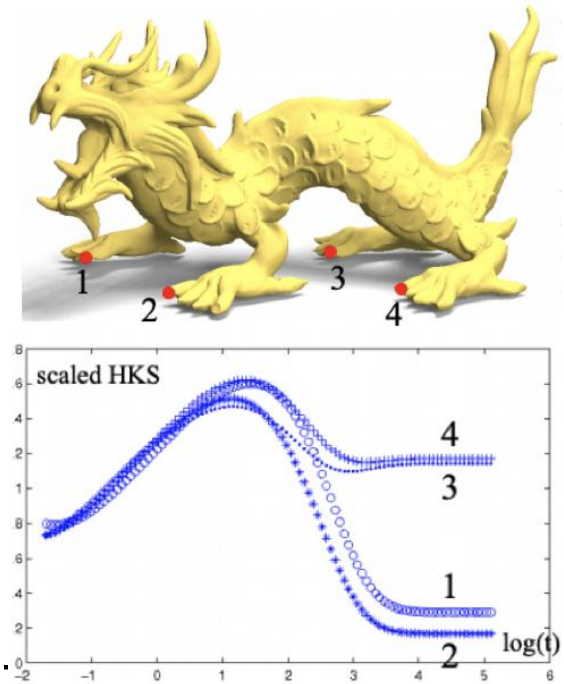


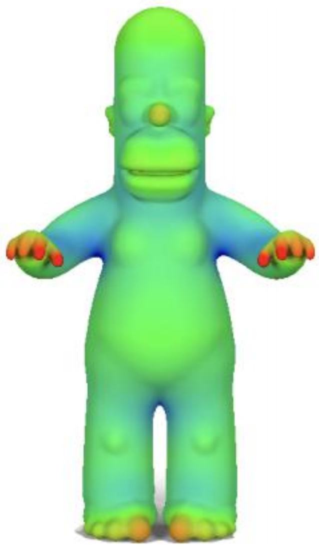
Laplacian Applications

- ✓ Laplace-Beltrami operator/Laplacian
- ✓ Applications:
  - ✓ Feature extraction (harmonic isocurves, HKS, GPS)
  - ✓ Segmentation (dot scissor, spectral clustering)
  - ✓ Deformation (differential coordinates)
  - ✓ Compression (least-squares meshes)
  - ✓ Parameterization (Tutte embedding)
  - ✓ Remeshing (spectral simplification)
  - ✓ Correspondence (functional maps)
  - ✓ Smoothing (surface fairing)
  - ✓ Sampling (local maxima)
  - ✓ Distance (biharmonic)

Laplacian Applications

- ✓ Laplace-Beltrami operator/Laplacian
- ✓ Applications:
  - ✓ Feature extraction (harmonic isocurves, HKS, GPS)

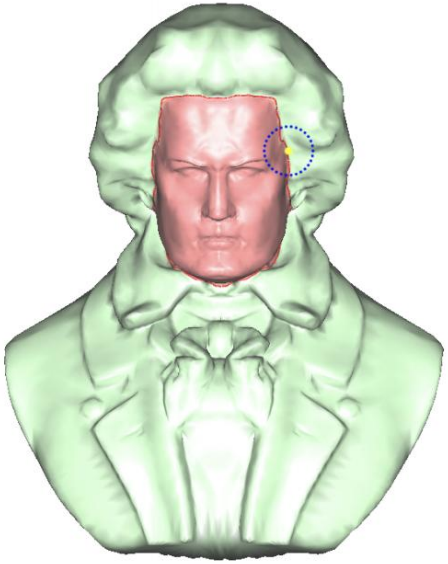




- ✓ Pairwise Harmonics for Shape Analysis, 2013.
- ✓ A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion, 2009.
- ✓ Laplace-Beltrami Eigenfunctions for Deformation Invariant Shape Representation, 2007.

Laplacian Applications

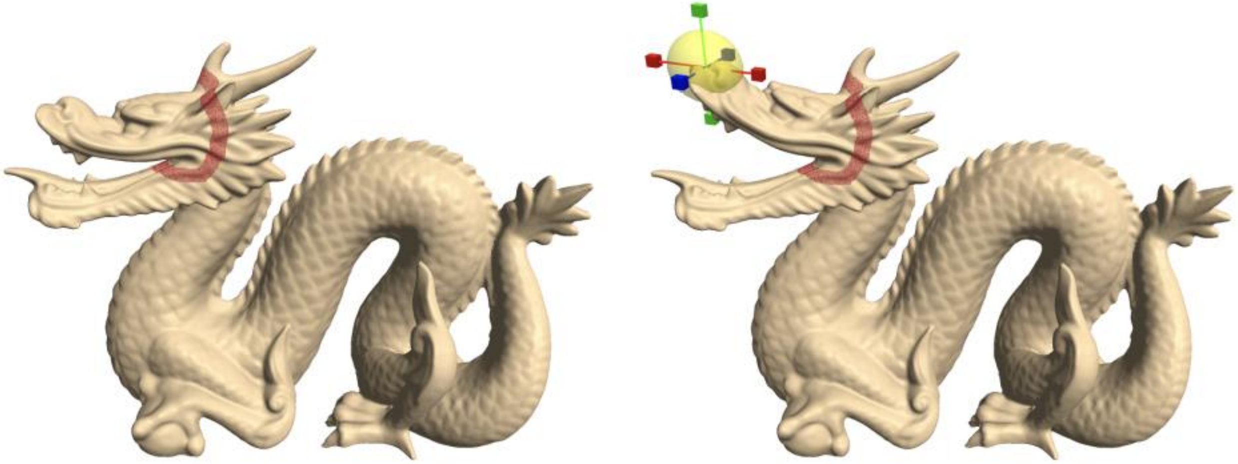
- ✓ Laplace-Beltrami operator/Laplacian
- ✓ Applications:
  - ✓ Segmentation (dot scissor, spectral clustering)



- ✓ Dot Scissor: A Single-Click Interface for Mesh Segmentation, 2011.
- ✓ Segmentation of 3D Meshes through Spectral Clustering, 2004.

Laplacian Applications

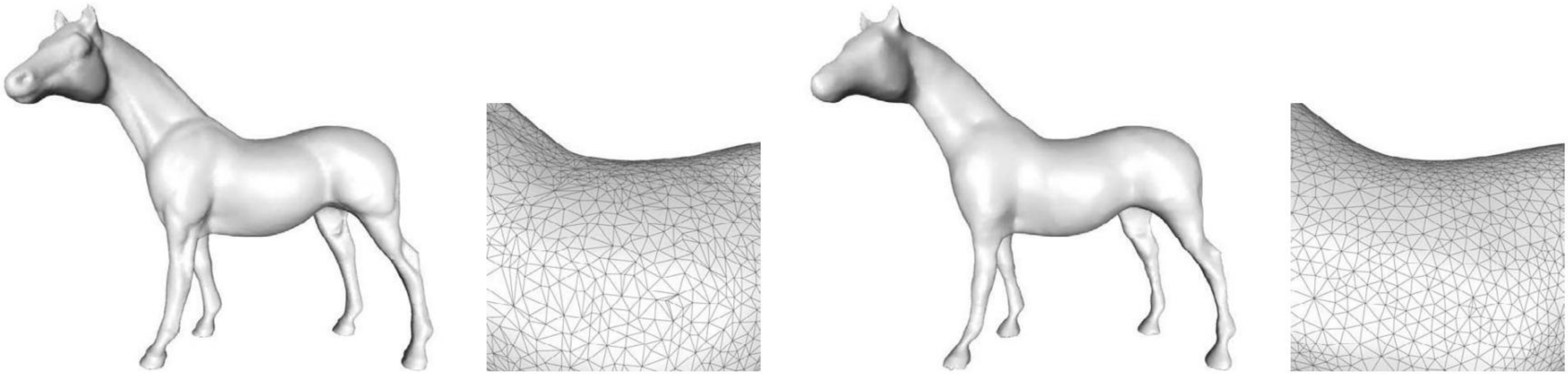
- ✓ Laplace-Beltrami operator/Laplacian
- ✓ Applications:
  - ✓ Deformation (differential coordinates)



- ✓ Laplacian Surface Editing, 2004.

Laplacian Applications

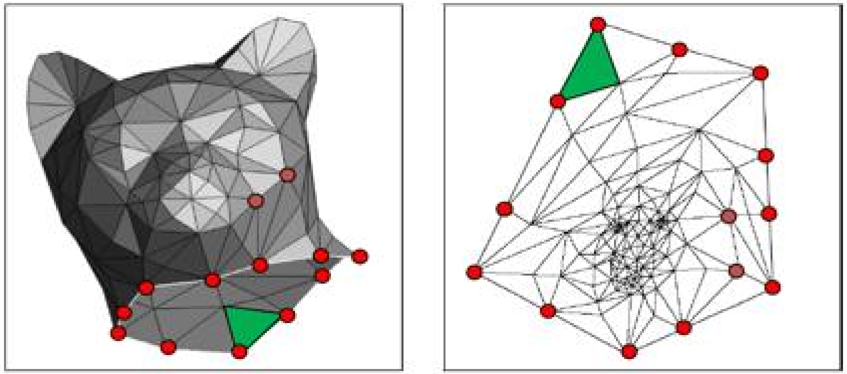
- ✓ Laplace-Beltrami operator/Laplacian
- ✓ Applications:
  - ✓ Compression (least-squares meshes)



- ✓ Spectral compression of mesh geometry, 2010.
- ✓ Least-squares Meshes, 2004.

Laplacian Applications

- ✓ Laplace-Beltrami operator/Laplacian
- ✓ Applications:
  - ✓ Parameterization (Tutte embedding)

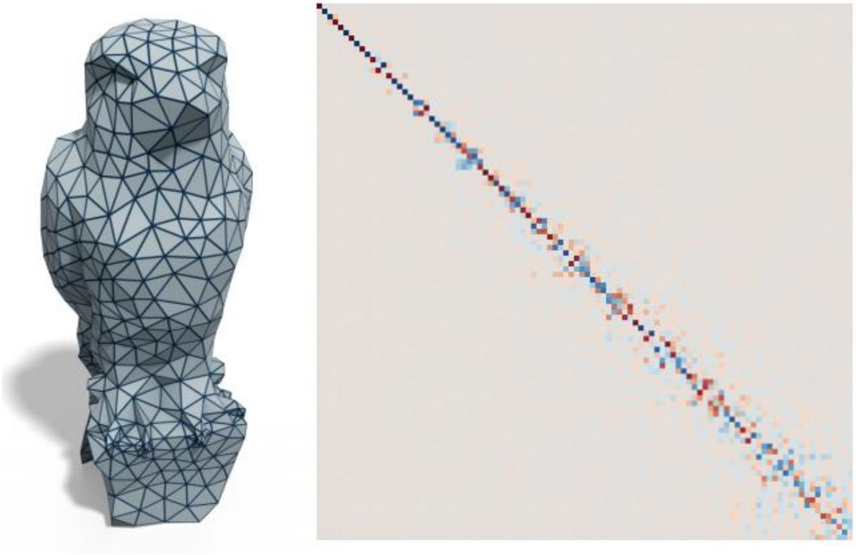


$$\sum_{(i,j)\in E} w_{ij}(v_i - v_j) = 0$$

- ✓ How to draw a graph, 1963.

Laplacian Applications

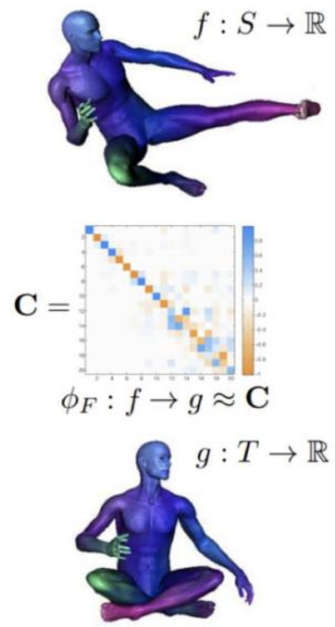
- ✓ Laplace-Beltrami operator/Laplacian
- ✓ Applications:
  - ✓ Remeshing (spectral simplification)



- ✓ Spectral Mesh Simplification, 2020.

Laplacian Applications

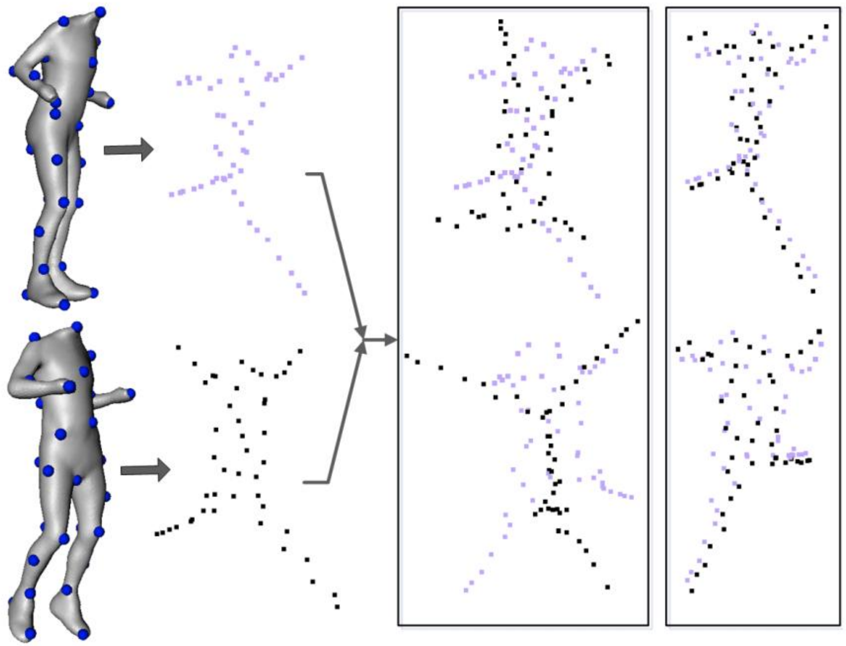
- ✓ Laplace-Beltrami operator/Laplacian
- ✓ Applications:
  - ✓ Correspondence (functional maps)



- ✓ Functional Maps: A Flexible Representation of Maps Between Shapes, 2012.

Laplacian Applications

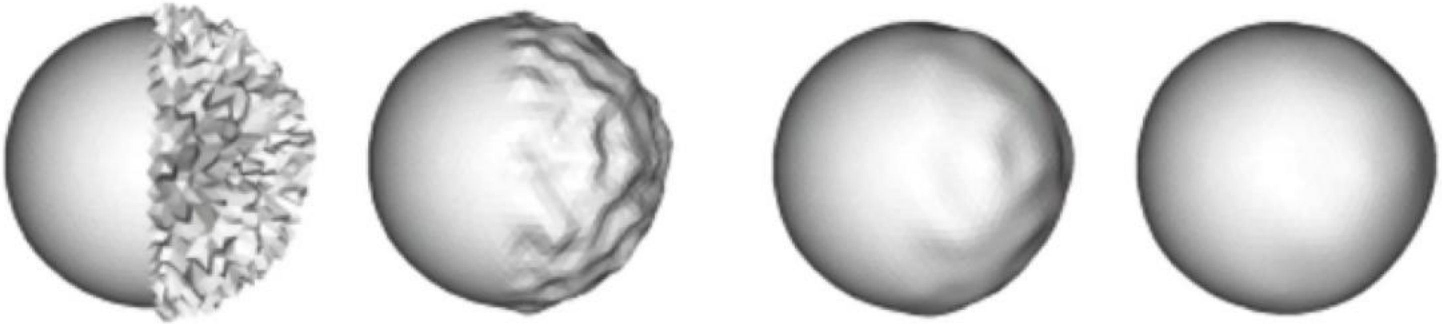
- ✓ Laplace-Beltrami operator/Laplacian
- ✓ Applications:
  - ✓ Correspondence (initialization – matching spectral embeddings)



- ✓ Minimum-Distortion Isometric Shape Correspondence Using EM Algorithm, 2012.

Laplacian Applications

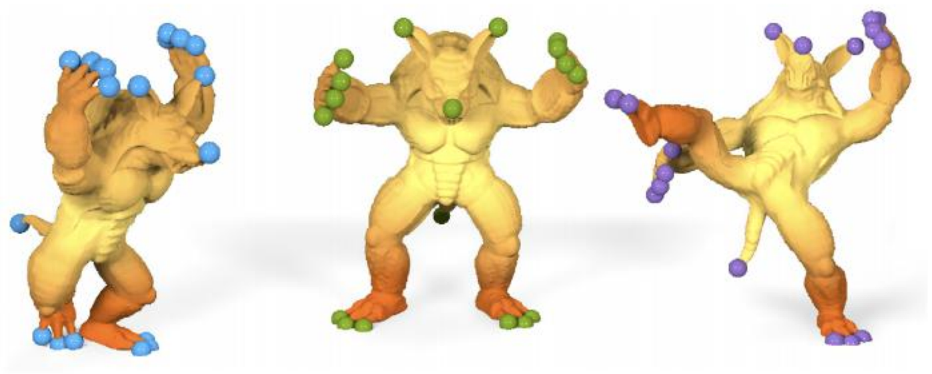
- ✓ Laplace-Beltrami operator/Laplacian
- ✓ Applications:
  - ✓ Smoothing (surface fairing)



- ✓ A Signal Processing Approach To Fair Surface Design, 1995.

Laplacian Applications

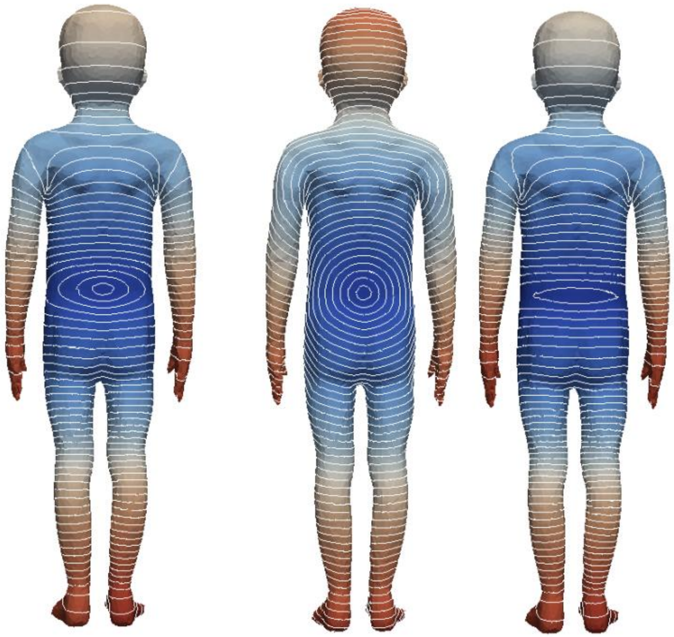
- ✓ Laplace-Beltrami operator/Laplacian
- ✓ Applications:
  - ✓ Sampling (local maxima of the HKS at a large time)



✓ A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion, 2009.

Laplacian Applications

- ✓ Laplace-Beltrami operator/Laplacian
- ✓ Applications:
  - ✓ Distance (biharmonic)



(a) Biharmonic (this paper) (b) Geodesic (c) Diffusion

- ✓ Biharmonic distance, 2010.
- ✓ The Heat Method for Distance Computation, 2017.

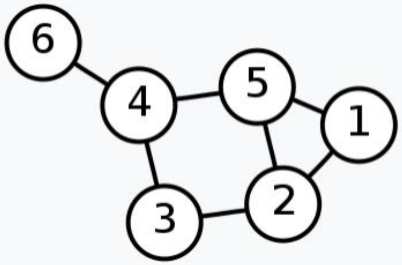
Discretization

- ✓ Discretization of the Laplace-Beltrami operator, Laplacian matrix.
- ✓ Laplacian matrix can be constructed in various ways.
- ✓ Intuition: encodes deviation of a point from its neighbors.

Discretization

- ✓ Uniform Laplacian
- ✓ Symmetric

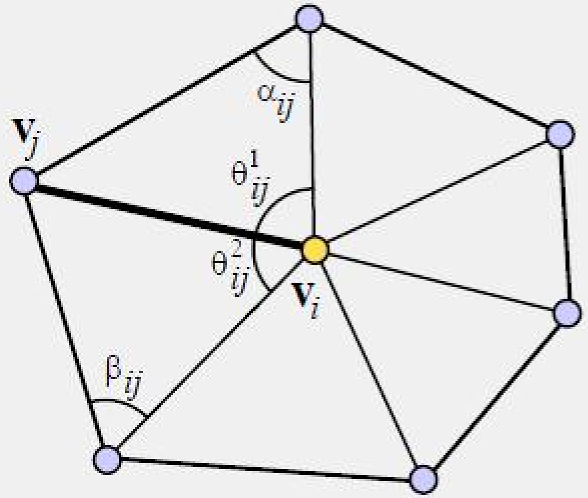
L = D - A

Labelled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2&0&0&0&0&0\\ 0&3&0&0&0&0\\ 0&0&2&0&0&0\\ 0&0&0&3&0&0\\ 0&0&0&0&3&0\\ 0&0&0&0&0&1 \end{pmatrix}$	$\begin{pmatrix} 0&1&0&0&1&0\\ 1&0&1&0&1&0\\ 0&1&0&1&0&0\\ 0&0&1&0&1&1\\ 1&1&0&1&0&0\\ 0&0&0&1&0&0 \end{pmatrix}$	$\begin{pmatrix} 2&-1&0&0&-1&0\\ -1&3&-1&0&-1&0\\ 0&-1&2&-1&0&0\\ 0&0&-1&3&-1&-1\\ -1&-1&0&-1&3&0\\ 0&0&0&-1&0&1 \end{pmatrix}$

$$L_{i,j} := \begin{cases} \deg(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

Discretization

- ✓ Cotangent Laplacian
- ✓ Symmetric



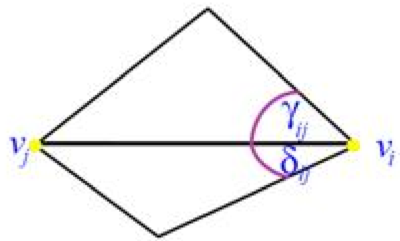
$$w_{ij} = \frac{1}{2}(cot\alpha_{ij} + cot\beta_{ij})$$

$$L_{i,j} := \begin{cases} \sum_j w_{ij} & \text{if } i = j \\ -w_{ij} & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

Discretization

- ✓ Mean-Value Laplacian
- ✓ Not symmetric any more ☹ but guaranteed positive weights

$$\boxed{w_{ij} = \frac{\tan(\gamma_{ij}/2) + \tan(\delta_{ij}/2)}{2\,\|v_i - v_j\|}}$$



$$L_{i,j} := \begin{cases} \sum_j w_{ij} & \text{if } i = j \\ -w_{ij} & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

Discretization

- ✓ Belkin Laplacian
- ✓ Symmetric
- ✓ No connectivity needed (point cloud Laplacian)
- ✓ Needs dense sampling
- ✓ Every entry is non-zero (not sparse)
  - ✓ Make it sparse by setting weights for k-nearest neighbors only

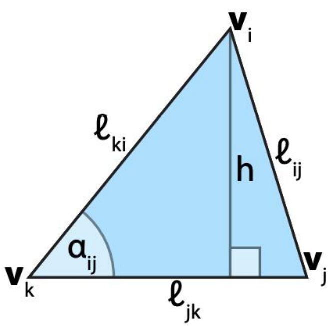
$$W_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{t}\right)$$

Tricky parameter to choose

$$L_{i,j} := \begin{cases} \sum_j w_{ij} & \text{if } i = j \\ -w_{ij} & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

Discretization

✓ Revisiting cotangent version (derivation)



Consider a triangle with vertices $\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k \in \mathbb{R}^d$. The triangle area A_{ijk} is defined intrinisically by [Heron 60]:

$$A_{ijk} = \sqrt{r(r - l_{ij})(r - l_{jk})(r - l_{ki})}$$

where l_{ij} is the length of the edge between $\mathbf{v}_i$ and $\mathbf{v}_j$, and r is the semi-perimeter $\frac{1}{2}(l_{ij} + l_{jk} + l_{ki})$.

We may similarly define the cotangent of the angle opposite each edge. First we can derive the cosine and sine. Recall the law of cosines:

$$l_{ij}^2 = l_{jk}^2 + l_{ki}^2 - 2l_{jk}l_{ki}\cos\alpha_{ij} \;\rightarrow\; \cos\alpha_{ij} = \frac{-l_{ij}^2 + l_{jk}^2 + l_{ki}^2}{2l_{jk}l_{ki}}.$$

For sine, we employ the familiar area formula treating the $\overline{\mathbf{v}_j\mathbf{v}_k}$ as base:

$$A_{ijk} = \frac{1}{2}l_{jk}l_{ki}\sin\alpha_{ij} \;\rightarrow\; \sin\alpha_{ij} = \frac{2A_{ijk}}{l_{jk}l_{ki}}.$$

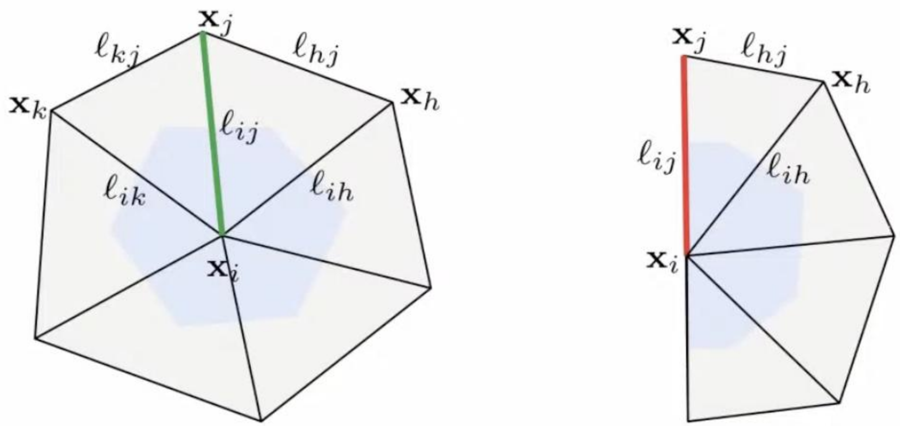
Finally putting these together we have:

$$\cot\alpha_{ij} = \frac{\cos\alpha_{ij}}{\sin\alpha_{ij}} = \frac{-l_{ij}^2 + l_{jk}^2 + l_{ki}^2}{2l_{jk}l_{ki}}\frac{l_{jk}l_{ki}}{2A_{ijk}} = \frac{-l_{ij}^2 + l_{jk}^2 + l_{ki}^2}{4A_{ijk}}.$$

Note that a similar intrinsic derivation is given in Equations 7 and 13 of [Meyer et al. 2003].

Discretization

✓ Revisiting cotangent version w/o angles (see slide23 for matrix A effect)



Cotangent Laplacian $\mathbf{\Delta} = \mathbf{A}^{-1}\mathbf{W}$ expressed in terms of discrete metric $\ell_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$ where

$$w_{ij} = \begin{cases} \frac{-\ell_{ij}^2+\ell_{jk}^2+\ell_{ki}^2}{8A_{ijk}} + \frac{-\ell_{ij}^2+\ell_{jh}^2+\ell_{hi}^2}{8A_{ijh}} & \text{if } e_{ij} \in \mathcal{E}_\text{i} \\ \frac{-\ell_{ij}^2+\ell_{jh}^2+\ell_{hi}^2}{8A_{ijh}} & \text{if } e_{ij} \in \mathcal{E}_\text{b} \\ -\sum_{k\neq i} w_{ik} & \text{if } i = j \end{cases} \qquad \mathbf{A} = \begin{pmatrix} a_1 & & \\ & \ddots & \\ & & a_n \end{pmatrix}$$

where A_{ijk} is area of triangle ijk and $a_i = \frac{1}{3} \sum_{ijk:ij,ik\in\mathcal{E}} A_{ijk}$

Discretization

- ✓ Active research area
  - ✓ Laplacian for polygon meshes
    - ✓ Polygon Laplacian Made Simple, 2020.
  - ✓ Laplacian for non-manifolds
    - ✓ A Laplacian for Nonmanifold Triangle Meshes, 2020.

Laplacian Apps Revisited (w/ Math)

- ✓ Let's put the spectrum of the cotan Laplacian in action.

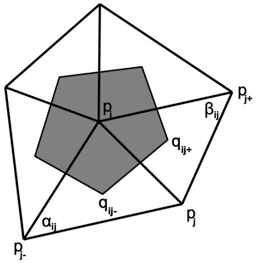
$$L\phi_i = \lambda_i \phi_i$$

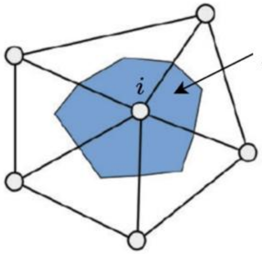
Laplacian Apps Revisited (w/ Math)

- ✓ Let's put the spectrum of the cotan Laplacian in action.
- ✓ We sometimes want eigenvectors to be orthonormal w.r.t. the diagonal area matrix A, which leads to the <u>generalized eigenvalue problem</u>:

$$A^{-1}L\phi_i = \lambda_i\phi_i \;\rightarrow\; L\phi_i = \lambda_i A\phi_i$$

- ✓ A_ii stores the Voronoi area of the ith vertex: approximate via 1/3 of the total surrounding area.

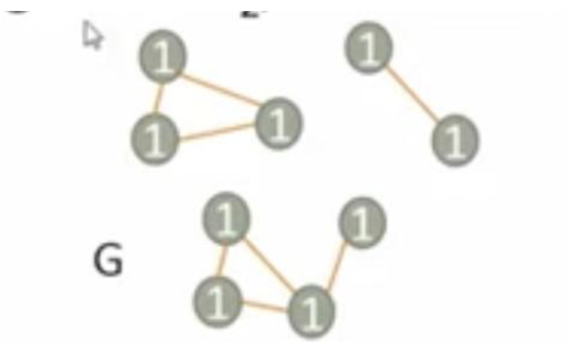




$$\frac{1}{3}\sum_{T_j \in N(i)} area(T_j)$$

Laplacian Apps Revisited (w/ Math)

- ✓ Since cotangent Laplacian, common choice, is symmetric positive:
  - ✓ Eigenvectors form a set of good bases, e.g., orthonormal.
  - ✓ Eigenvalues reveal global properties not apparent from edges:
    - ✓ $$0 = \lambda_1 < \lambda_2 \le \lambda_3 \le \ldots \le \lambda_n.$$
    - ✓ Graph/mesh has k connected components, if the first k eigenvalues are 0.

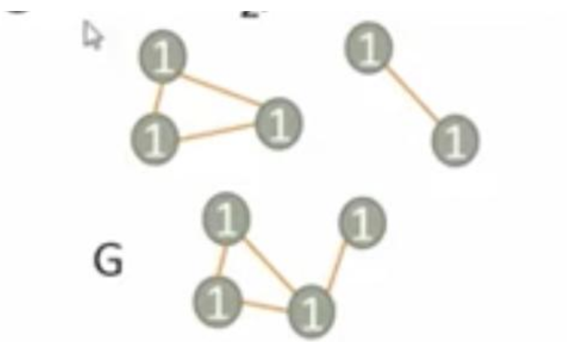


L_G has $0 = \lambda_1 = \lambda_2 = \lambda_3$ and $\lambda_4 > 0$

Both L_{G1} and L_{G2} have $0 = \lambda_1$ and $\lambda_2 > 0$. $\lambda_2(L_{G1}) < \lambda_2(L_{G2})$

Laplacian Apps Revisited (w/ Math)

- ✓ Since cotangent Laplacian, common choice, is symmetric positive:
  - ✓ Eigenvectors form a set of good bases, e.g., orthonormal.
  - ✓ Eigenvalues reveal global properties not apparent from edges:
    - ✓ $0 = \lambda_1 < \lambda_2 \leq \lambda_3 \leq \ldots \leq \lambda_n$.
    - ✓ if the graph is connected, lambda_2 > 0, called algebraic connectivity.
    - ✓ the greater lambda_2, the more connected graph is.

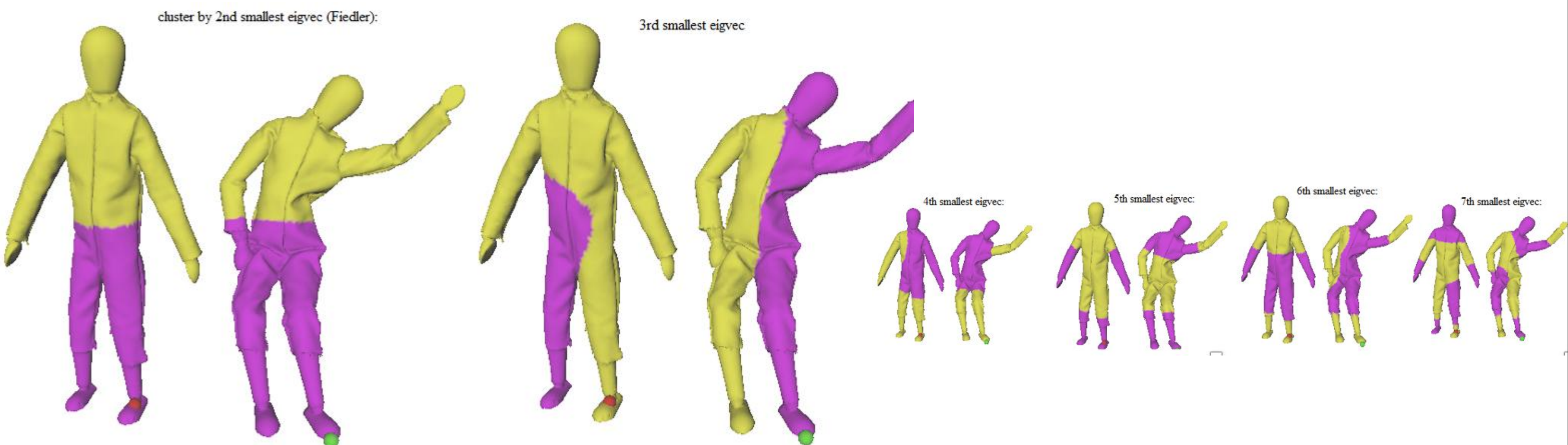


L_G has $0 = \lambda_1 = \lambda_2 = \lambda_3$ and $\lambda_4 > 0$

Both L_{G1} and L_{G2} have $0 = \lambda_1$ and $\lambda_2 > 0$. $\lambda_2(L_{G1}) < \lambda_2(L_{G2})$

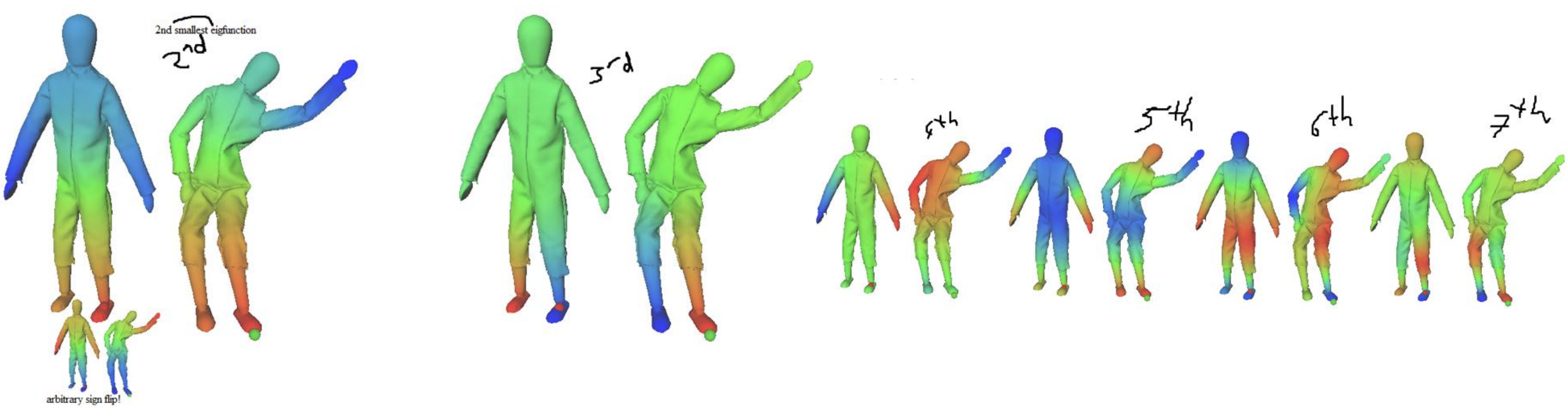
Laplacian Apps Revisited (w/ Math)

- ✓ Since cotangent Laplacian, common choice, is symmetric positive:
  - ✓ Eigenvectors form a set of good bases, e.g., orthonormal.
  - ✓ Eigenvalues reveal global properties not apparent from edges:
    - ✓ $0 = \lambda_1 < \lambda_2 \leq \lambda_3 \leq \ldots \leq \lambda_n.$
    - ✓ Eigenvector corresponding to lambda_2 is Fiedler vector and useful for binary partitioning: negative terms pink, +ve yellow.



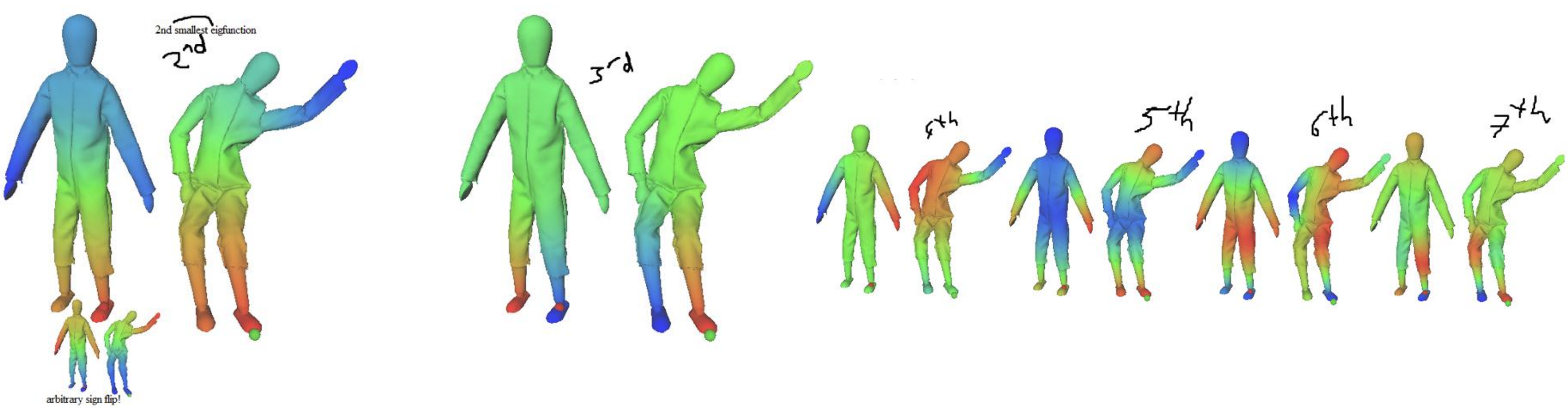
Laplacian Apps Revisited (w/ Math)

- ✓ Since cotangent Laplacian, common choice, is symmetric positive:
  - ✓ Eigenvectors form a set of good bases, e.g., orthonormal.
  - ✓ Eigenvalues reveal global properties not apparent from edges:
    - ✓ $$0 = \lambda_1 < \lambda_2 \leq \lambda_3 \leq \ldots \leq \lambda_n.$$
    - ✓ First eigenvectors (corresponding to small values) are smooth slowly varying functions on the mesh.
    - ✓ Last eigenvectors have high frequency (rapid oscillations).



Laplacian Apps Revisited (w/ Math)

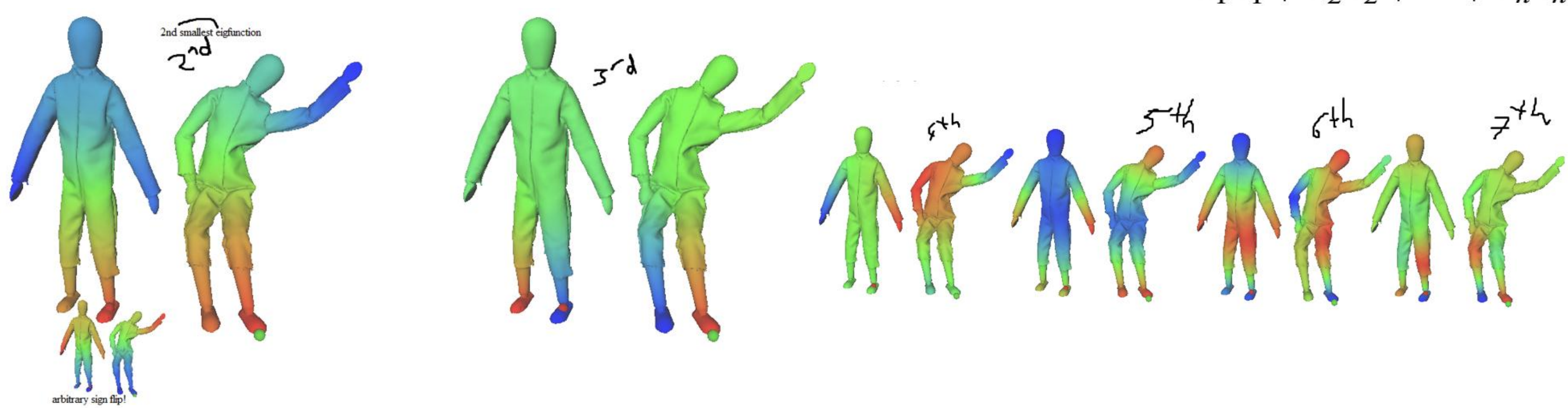
- ✓ Since cotangent Laplacian, common choice, is symmetric positive:
  - ✓ Eigenvectors form a set of good bases, e.g., orthonormal.
  - ✓ Eigenvalues reveal global properties not apparent from edges:
    - ✓ $$0 = \lambda_1 < \lambda_2 \leq \lambda_3 \leq \ldots \leq \lambda_n.$$
    - ✓ First eigenvector is the constant, smoothest, function that does not vary at all, hence not informative at all. Simply discard.



Laplacian Apps Revisited (w/ Math)

- ✓ Since cotangent Laplacian, common choice, is symmetric positive:
  - ✓ Eigenvectors form a set of good bases, e.g., orthonormal.
  - ✓ Eigenvalues reveal global properties not apparent from edges:
    - ✓ $0 = \lambda_1 < \lambda_2 \leq \lambda_3 \leq \ldots \leq \lambda_n.$
    - ✓ Laplacian eigenbasis is an extension of discrete cosine basis where the eigenvalues are considered as mesh frequencies.
    - ✓ Similar to JPEG compression, we've mesh compression [Karni'00]

$$\mathbf{x} = \alpha_1 \mathbf{e}_1 + \alpha_2 \mathbf{e}_2 + \ldots + \alpha_n \mathbf{e}_n$$

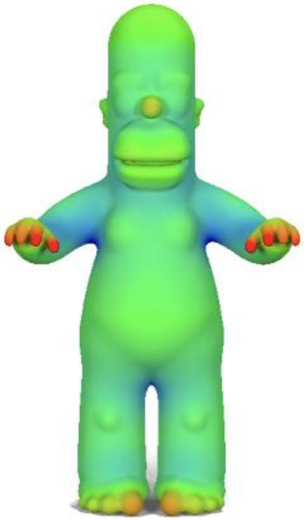


Laplacian Apps Revisited (w/ Math): Descriptor

- ✓ Amount of heat diffused from x to y in time t.
  - ✓ Set y=x to get the heat kernel signature (HKS) at point x.

$$k_t(x,y) = \sum_{i=0}^{\infty} e^{-\lambda_i t} \phi_i(x)\phi_i(y)$$

  - ✓ In practice, use the smallest 20 eigenvalues/eigenfunctions.
  - ✓ Sparse eigendecomposition takes 0.17 secs for a 12K-vertex mesh.



- ✓ A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion, 2009.

Laplacian Apps Revisited (w/ Math): Distance

- ✓ Heat-related global point signature (GPS)
  - ✓ No time parameter ☺
  - ✓ Arbitrary sign flip ☹

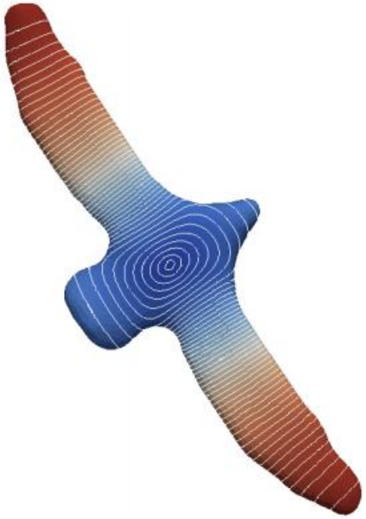
$$GPS(\mathbf{p}) = \left(\frac{1}{\sqrt{\lambda_1}} \phi_1(\mathbf{p}), \frac{1}{\sqrt{\lambda_2}} \phi_2(\mathbf{p}), \frac{1}{\sqrt{\lambda_3}} \phi_3(\mathbf{p}), \ldots \right)$$

- ✓ Heat-related biharmonic distance
  - ✓ No time parameter, unlike diffusion distance:

$$d_D(x,y)^2 = \sum_{k=1}^{\infty} e^{-2t\lambda_k} \left(\phi_k(x) - \phi_k(y)\right)^2$$

  - ✓ Topologically-robust (more paths considered)

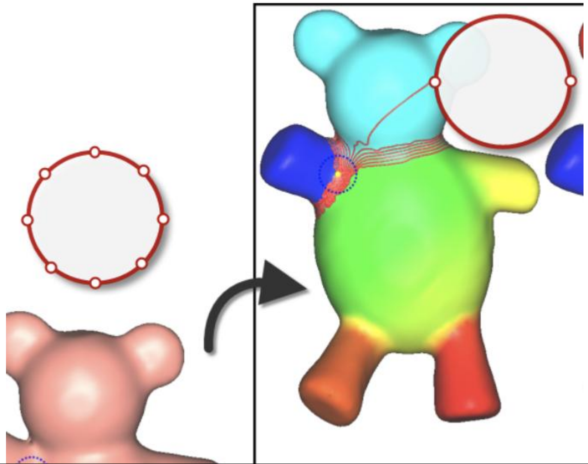
$$\tilde{d}_B(x,y)^2 = \sum_{k=1}^{K} \frac{\left(\phi_k(x) - \phi_k(y)\right)^2}{\lambda_k^2}.$$



Laplacian Apps Revisited (w/ Math): Segment

- ✓ K-means clustering on GPS coordinates

- ✓ Or get harmonic fields from the Laplacian for clustering



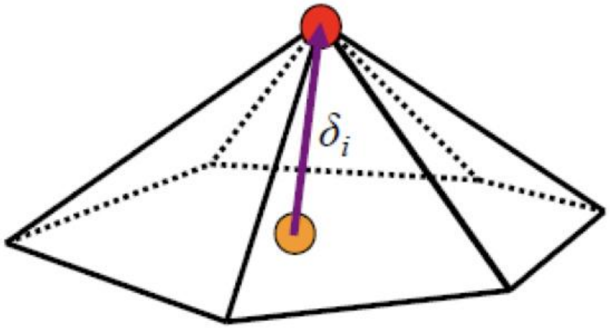
or symmetry axis extraction.

Laplacian Apps Revisited (w/ Math): Deformation

- ✓ Need a slightly different Laplacian, called General Laplacian, for deformation, as it would produce the differential coordinates when multiplied by the vertex vector.
- ✓ Differential coordinates encode the local structure, i.e., normal scaled by the mean curvature, of the mesh.

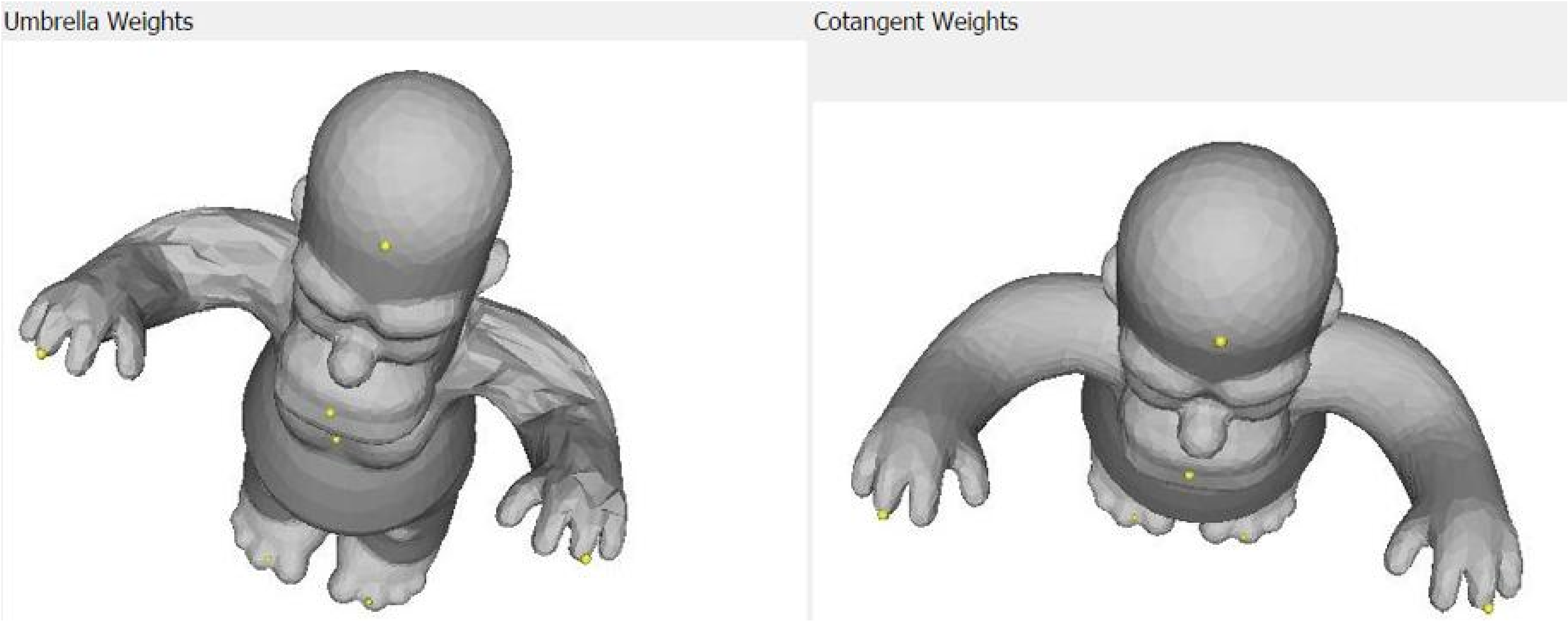
$$\delta_i = \frac{1}{d_i} \sum_{j \in N(i)} (\mathbf{v}_i - \mathbf{v}_j) \qquad \delta_i^c = \frac{1}{|\Omega_i|} \sum_{j \in N(i)} \frac{1}{2} (\cot \alpha_{ij} + \cot \beta_{ij}) (\mathbf{v}_i - \mathbf{v}_j)$$

$$L_{ij} = \begin{cases} 1 & i = j \\ -\frac{1}{d_i} & (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$



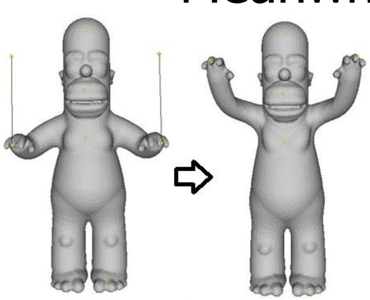
Laplacian Apps Revisited (w/ Math): Deformation

- ✓ Need a slightly different Laplacian, called General Laplacian, for deformation, as it would produce the differential coordinates when multiplied by the vertex vector.
- ✓ Differential coordinates encode the local structure, i.e., normal scaled by the mean curvature, of the mesh.



Laplacian Apps Revisited (w/ Math): Deformation

- ✓ Need a slightly different Laplacian, called General Laplacian, for deformation, as it would produce the differential coordinates when multiplied by the vertex vector.
- ✓ Differential coordinates encode the local structure, i.e., normal scaled by the mean curvature, of the original "good" mesh.
- ✓ Compute the differential coordinates of the deformed mesh too and keep it as close to the original diff cords as possible, regularization.
- ✓ Meanwhile let the deformed cords go towards to the data points, data

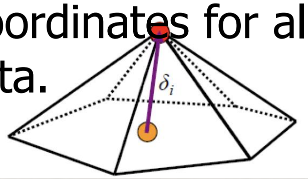


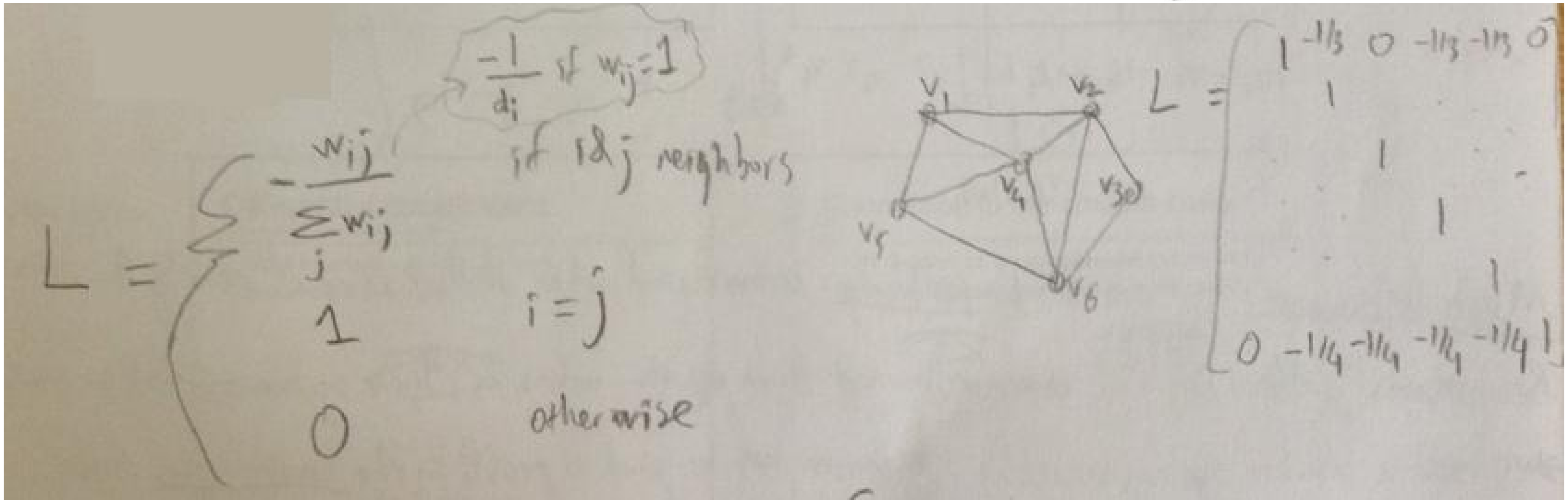
$$E(v) = \|v - c\|^2 + \alpha \|Lv - L \cdot v_0\|^2$$

$$L_{ij} = \begin{cases} 1 & i = j \\ -\frac{1}{d_i} & (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$ is the uniform one. Cotan version is created similarly.

Laplacian Apps Revisited (w/ Math): Deformation

- ✓ Differential Coordinates by General Laplacian Matrix **L**.
  - ✓ Since a differential coordinate is a linear combination of a vertex and its neighbors, the process of constructing differential coordinates for all vertices can be represented as a matrix **L** such that **L**v = delta.



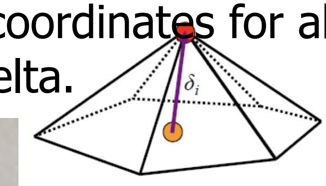


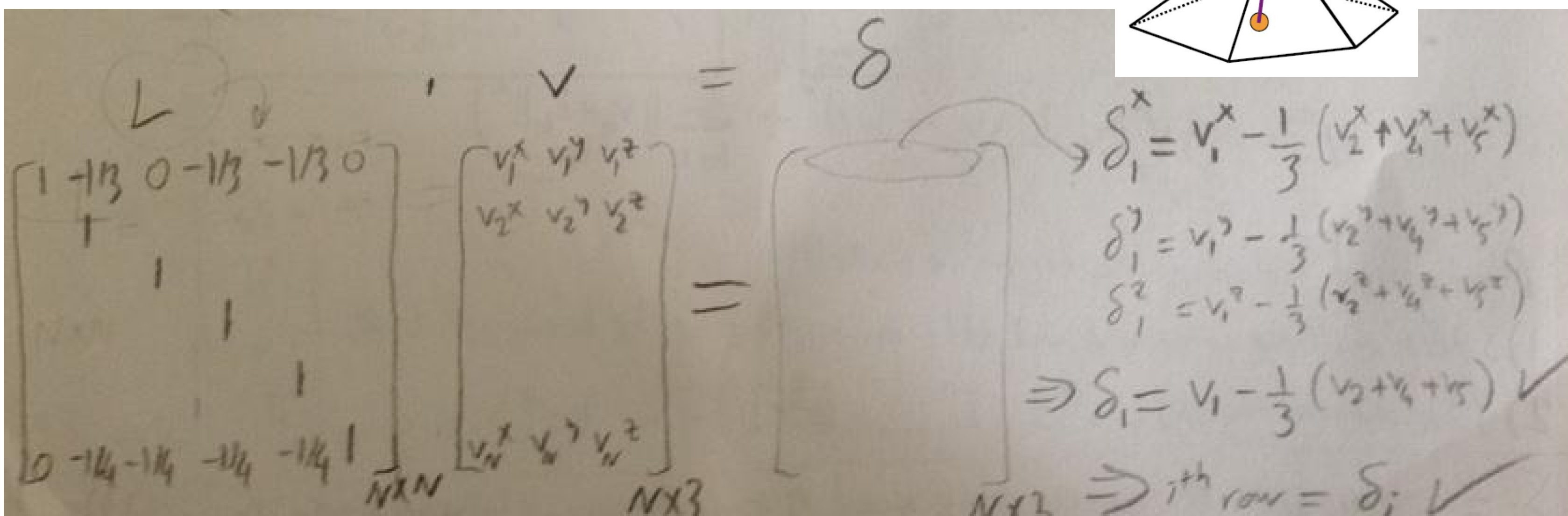
$$L_{ij} = \begin{cases} 1 & i = j \\ -\frac{1}{d_i} & (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

Use cotan weights in my handwriting above.

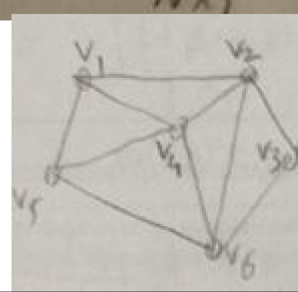
Laplacian Apps Revisited (w/ Math): Deformation

- ✓ Differential Coordinates by General Laplacian Matrix **L**.
  - ✓ Since a differential coordinate is a linear combination of a vertex and its neighbors, the process of constructing differential coordinates for all vertices can be represented as a matrix **L** such that **L**v = delta.





$$L_{ij} = \begin{cases} 1 & i = j \\ -\frac{1}{d_i} & (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$



Recall, $\vec{\delta_i} = \vec{v_i} - \frac{1}{d_i} \sum_{j \in N(i)} \vec{v_j}$.

Laplacian Apps Revisited (w/ Math): Deformation

- ✓ No exact solution: least-squares minimization (***all*** points constrained).

$$E(v) = \|v-c\|^2 + \alpha\|Lv - L\cdot v_0\|^2$$

where $L\cdot v_0 = \delta_0$, initial delta coordinates. $L\cdot v = \delta$, current delta coordinates.

$$= (v-c)^T(v-c) + \alpha(Lv-Lv_0)^T(Lv-Lv_0)$$

$$= v^Tv - v^Tc - c^Tv + c^Tc + \alpha\left((Lv)^T(Lv) - (Lv)^T(Lv_0) - (Lv_0)^T(Lv) + (Lv_0)^T(Lv_0)\right)$$

$$= v^Tv - 2v^Tc + c^Tc + \alpha\left(v^TL^TLv - 2(Lv)^T(Lv_0) + v_0^TL^TLv_0\right)$$

$$- 2v^TL^TLv_0$$

$$\frac{\partial E}{\partial v} = \cancel{2}v - \cancel{2}c + \cancel{2}\alpha L^TLv - \cancel{2}L^TLv_0 = 0$$

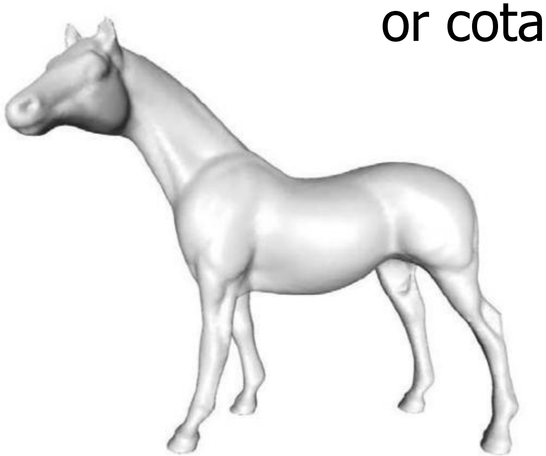
Eigen Library: SimplicialCholesky (solve by Cholesky)

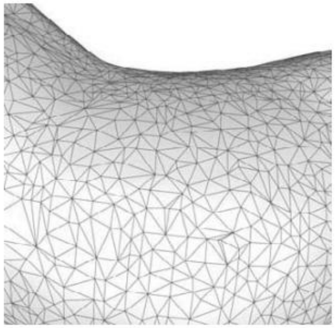
$$(I+\alpha L^TL)v = c + L^TLv_0 \Rightarrow Ax=b \text{ sparse linear sys}$$

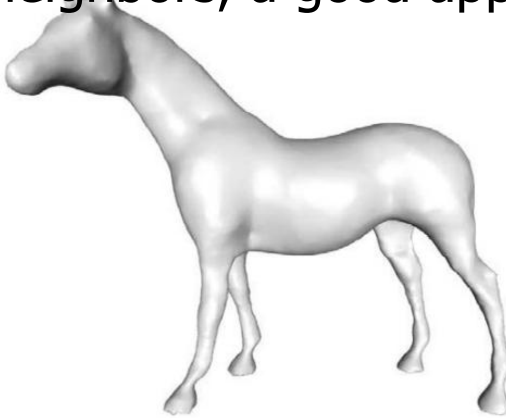
- ✓ Do this for x-, y-, z-coords separately, i.e., L is nxn, v is nx1 w/ the x-coords of all verts; and similarly for y-, z-.

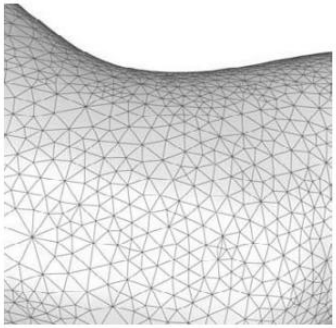
Laplacian Apps Revisited (w/ Math): Compress

- ✓ Already seen JPEG-like compression (Slide 29) via eigenbasis.
- ✓ Least-squares solution for the same problem:
  - ✓ Store just the xyz coordinates of the few anchor points
  - ✓ Store also the *sparse* General Laplacian
  - ✓ This is your compression. For recovery just solve the least-squares problem similar to the deformation case
    - ✓ Replace Lv_0 with 0 to save even more in your compression
    - ✓ Putting 0-vector recovers each vertex to the weighted (uniform or cotan) centroid of its neighbors, a good approximation









Laplacian Apps Revisited (w/ Math): Compress

- ✓ Already seen JPEG-like compression (Slide 29) via eigenbasis.
- ✓ Least-squares solution for the same problem:

$$E(v) = \|v - c\|^2 + \alpha \|Lv - \cancel{L} v_0\|^2$$ where $L.v_0 = \delta_0$, initial delta coordinates. $L.v = \delta$, current delta coordinates.

$$= (v-c)^T(v-c) + \alpha (Lv - Lv_0)^T (Lv - Lv_0)$$

$$= v^T v - v^T c - c^T v + c^T c + \alpha \left((Lv)^T(Lv) - (Lv)^T(Lv_0) - (Lv_0)^T(Lv) + (Lv_0)^T(Lv_0) \right)$$

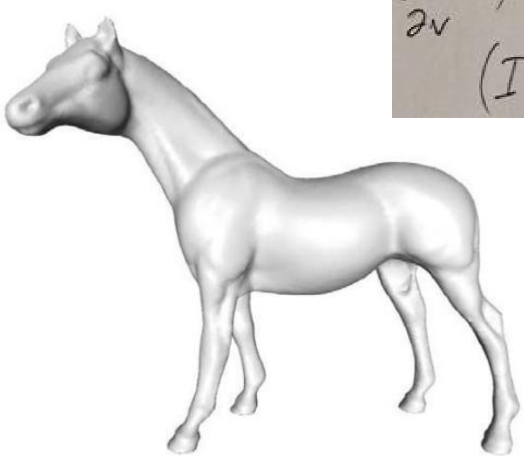
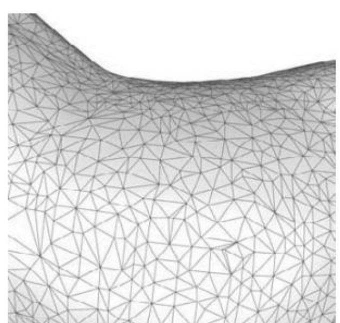
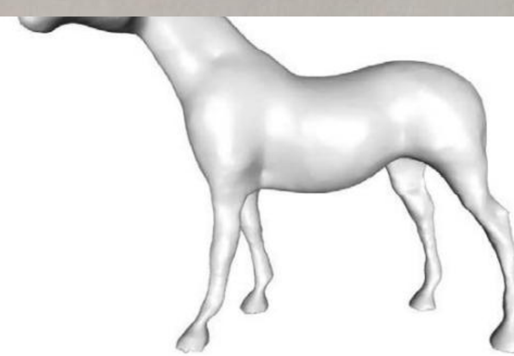
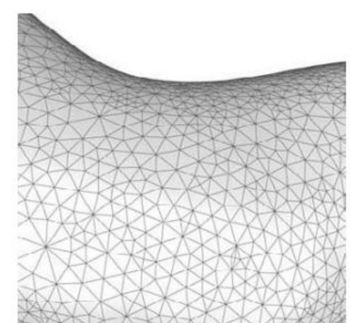
$$= v^T v - 2v^T c + c^T c + \alpha \left(v^T L^T L v - 2(Lv)^T(Lv_0) + v_0^T L^T L v_0 \right)$$

$$-2 v^T L^T L v_0$$

$$\frac{\partial E}{\partial v} = \cancel{2} v - \cancel{2} c + \cancel{2} \alpha L^T L v - \cancel{2} L^T L v_0 = 0$$

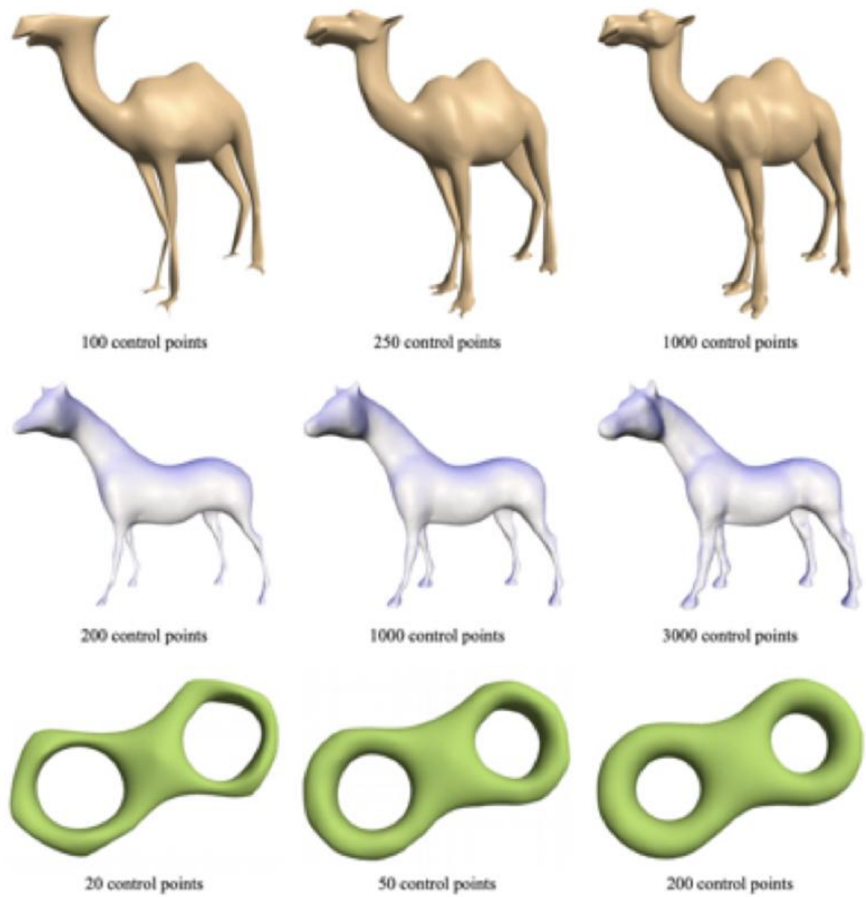
$$(I + \alpha L^T L) v = c + L^T L v_0 \Rightarrow Ax = b$$ sparse linear sys

Eigen Library: SimplicialCholesky (solve by Cholesky)

Laplacian Apps Revisited (w/ Math): Compress

- ✓ Seen JPEG-like compression (Slide29) via eigenbasis (Gen. Lap. in use).
- ✓ Least-squares solution for the same problem:



vertices: 39074

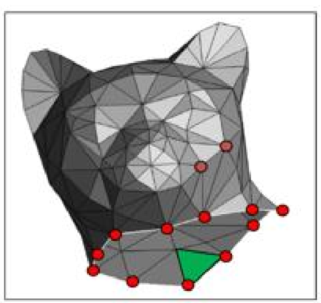
vertices: 19851

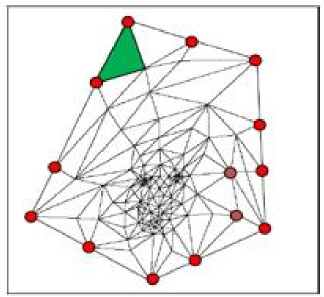
vertices: 2718

Laplacian Apps Revisited (w/ Math): Paramtrztn

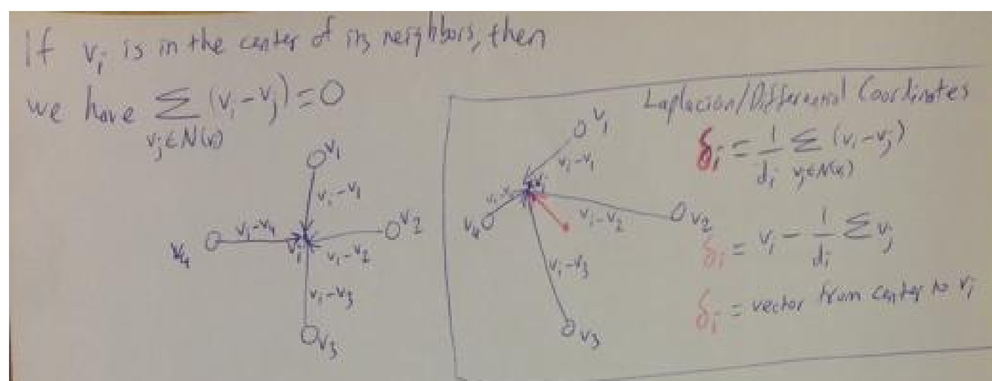
✓ Map boundary to a convex region, e.g., b = disk //square below:

✓ Map non-boundary so that each one is in the weighted (uniform or cotan) centroid of its neighbors: Wx = b ➔ x = W^{-1}b





$$\sum_{(i,j)\in E} w_{ij}(v_i - v_j) = 0$$

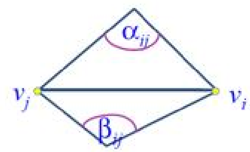


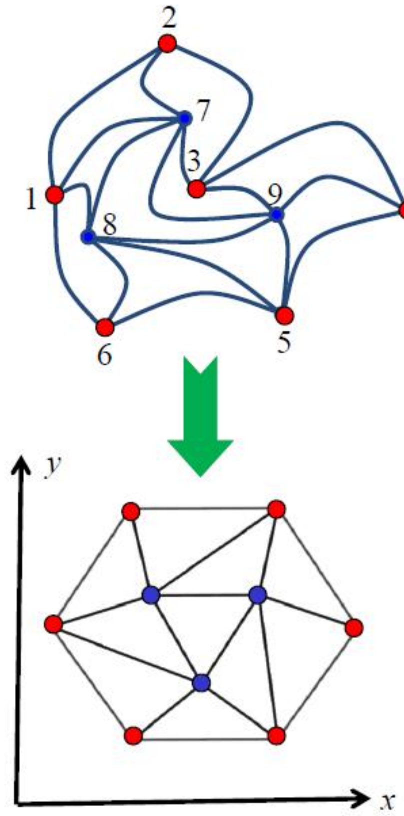
Laplacian Apps Revisited (w/ Math): Paramtrztn

✓ Bottom part of W is a negated subset of the uniform (or cot) Laplacian.

$$\boxed{w_{ij} = 1}$$

$$\boxed{w_{ij} = \frac{\cot(\alpha_{ij}) + \cot(\beta_{ij})}{2}}$$



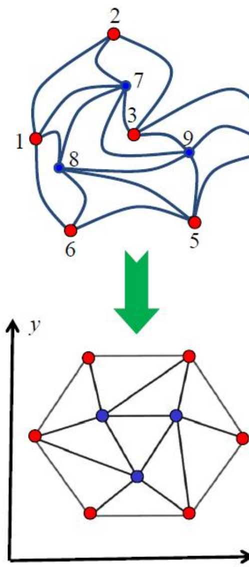


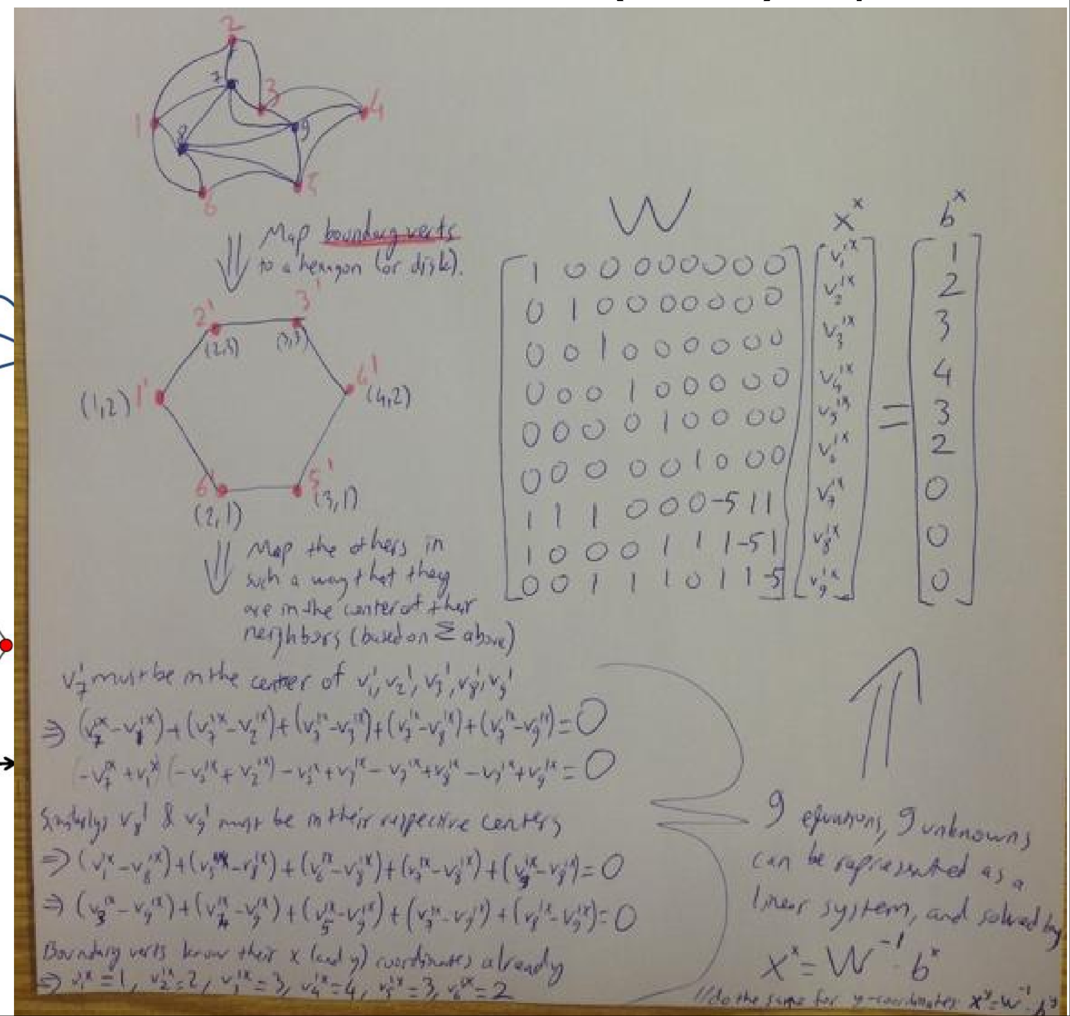
$$W = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & -5 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & -5 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & -5 \end{pmatrix} \quad b_x = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 3 \\ 2 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad b_y = \begin{pmatrix} 2 \\ 3 \\ 3 \\ 2 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Laplacian Apps Revisited (w/ Math): Paramtrztn

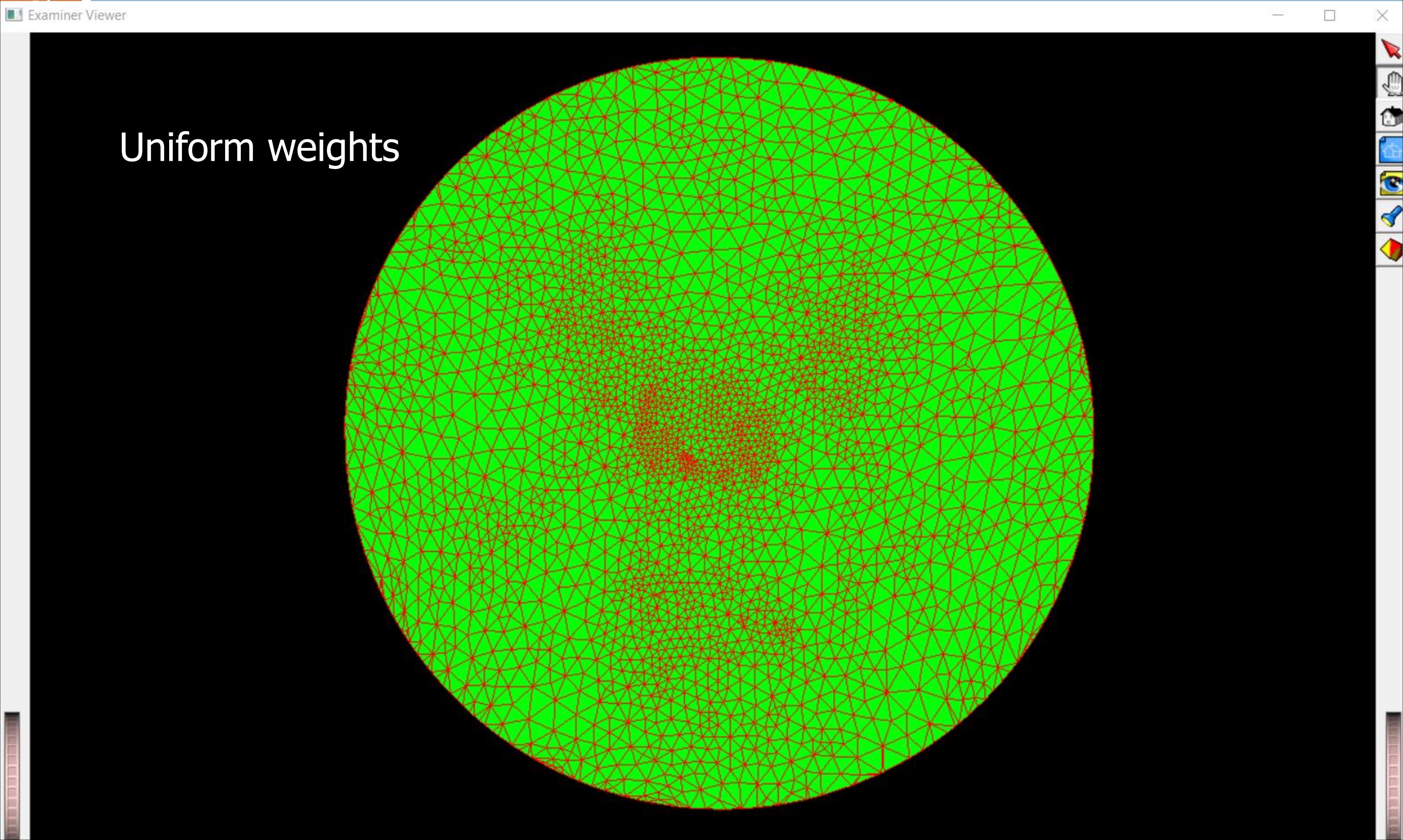
- ✓ Bottom part of W is a negated subset of the uniform (or cot) Laplacian.

$$W = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & -5 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & -5 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & -5 \end{pmatrix} \quad b_x = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 3 \\ 2 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad b_y = \begin{pmatrix} 2 \\ 3 \\ 3 \\ 2 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

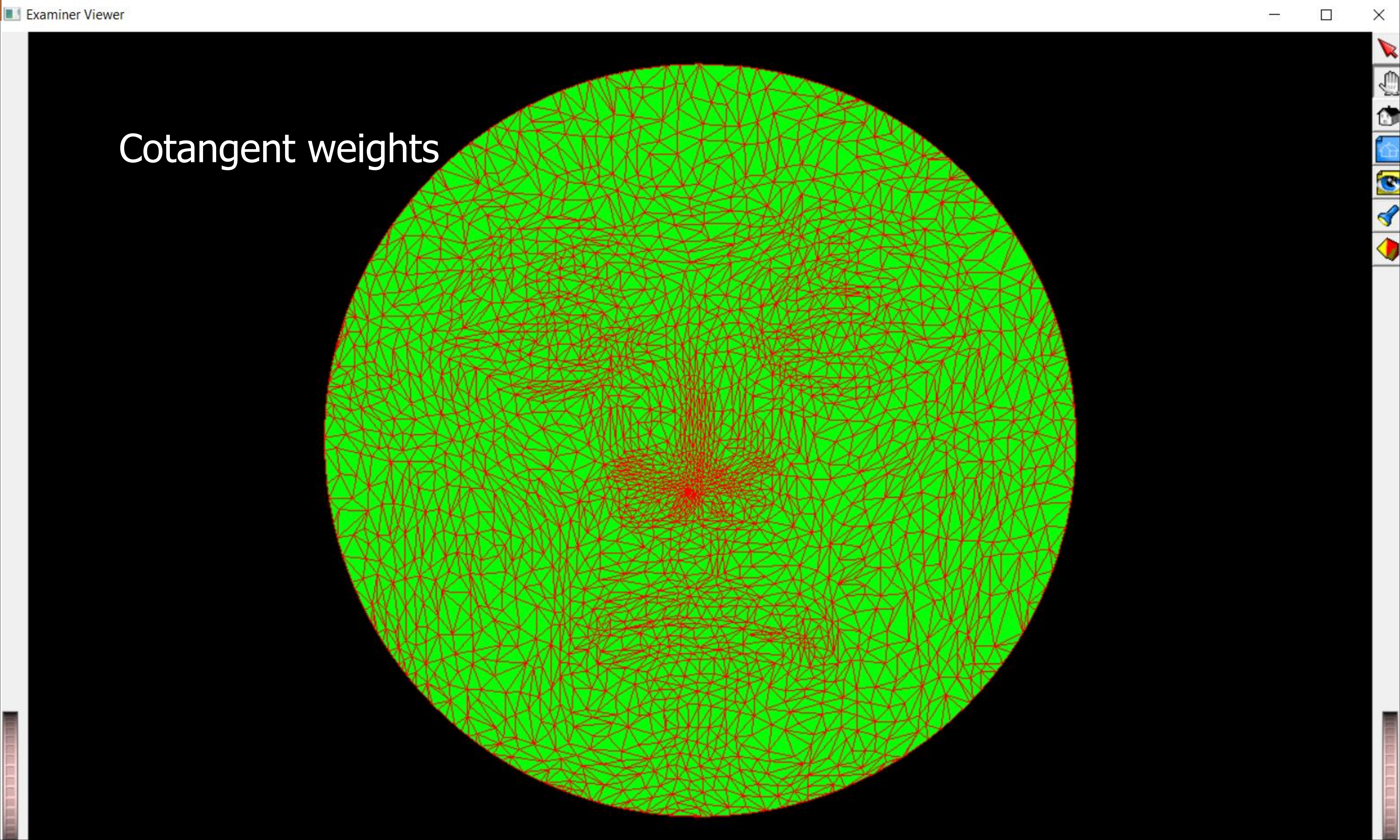




Laplacian Apps Revisited (w/ Math): Paramtrztn

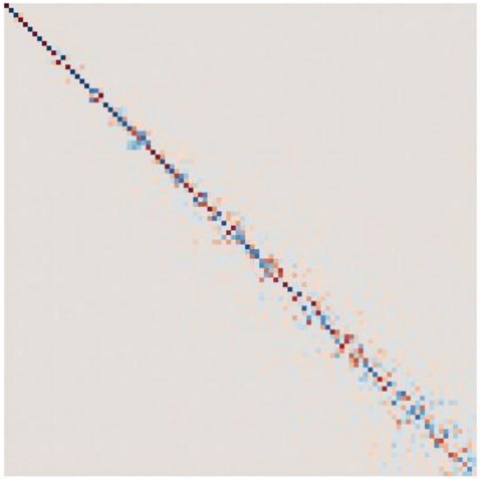


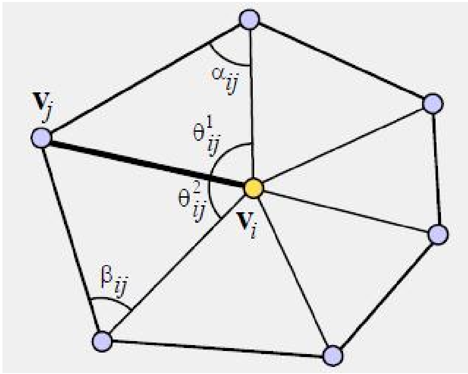
Laplacian Apps Revisited (w/ Math): Paramtrztn



Laplacian Apps Revisited (w/ Math): Remeshing

- ✓ Simplify the mesh by collapsing edges that will affect the spectrum of the Laplacian matrix the least.
- ✓ Leads to an appearance-preserving and also spectrum-preserving low-resolution mesh.





$$w_{ij} = \frac{1}{2}(cot\alpha_{ij} + cot\beta_{ij})$$

$$L_{i,j} := \begin{cases} \sum_j w_{ij} & \text{if } i = j \\ -w_{ij} & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

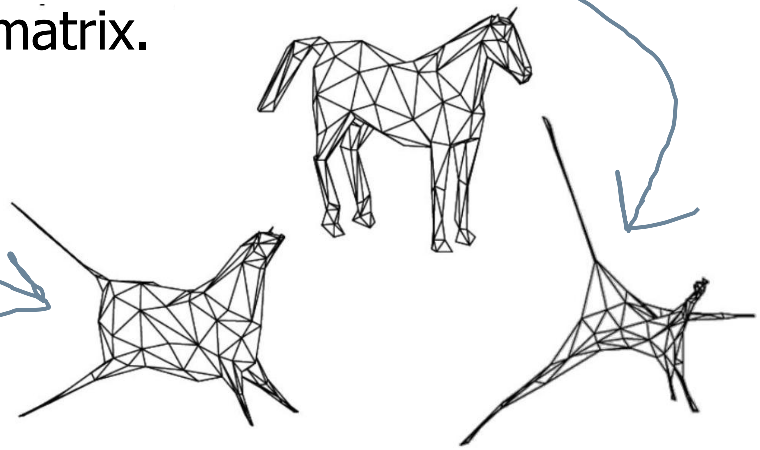
Laplacian Apps Revisited (w/ Math): Corresp.

- ✓ Match functions, e.g., eigenfunctions, over surfaces: Functional Maps'12.

$$f = \sum_{i=1}^{k_S} a_i \phi_i^S \text{ and } g = \sum_{i=1}^{k_T} b_i \phi_i^T$$

- ✓ Desired map hidden in **C a** ~ **b** can be extracted w/ a linear solve, provided enough number of **a b** pairs (100 in use).

- ✓ Laplacian embedding gives isometry-invariant coordinates that can initialize a correspondence method. Embeddings of other matrices are also useful here, e.g., geodesic affinity matrix.



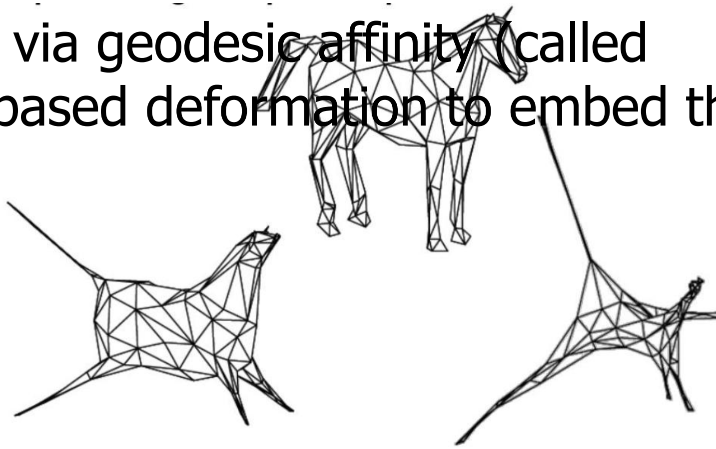
Laplacian Apps Revisited (w/ Math): Corresp.

- ✓ Match functions, e.g., eigenfunctions, over surfaces: Functional Maps'12.

$$f = \sum_{i=1}^{k_S} a_i \phi_i^S \text{ and } g = \sum_{i=1}^{k_T} b_i \phi_i^T$$

- ✓ Desired map hidden in **C a** ~ **b** can be extracted w/ a linear solve, provided enough number of **a b** pairs (100 in use).

- ✓ Unlike Laplacian, geodesic affinity matrix is not sparse so it's difficult to store and process (eigendecomposition).
- ✓ Solution: embed only few anchor points via geodesic affinity (called classical MDS), and then use Laplacian-based deformation to embed the rest (Slide 32).



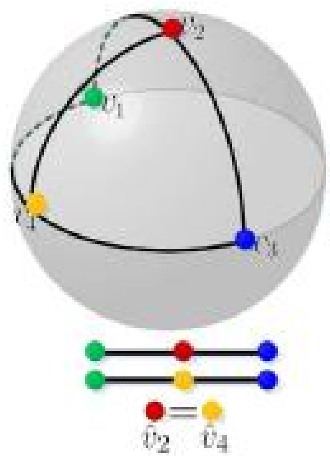
Laplacian Apps Revisited (w/ Math): Corresp.

- ✓ Match functions, e.g., eigenfunctions, over surfaces: Functional Maps'12.

$$f = \sum_{i=1}^{k_S} a_i \phi_i^S \text{ and } g = \sum_{i=1}^{k_T} b_i \phi_i^T$$

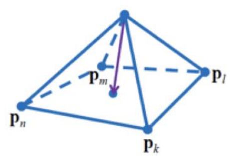
- ✓ Desired map hidden in **C a** ~ **b** can be extracted w/ a linear solve, provided enough number of **a b** pairs (100 in use).

- ✓ Geometric distortion is inevitable in such embeddings (details lost).

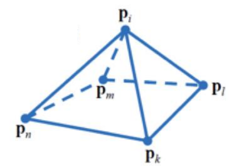


Laplacian Apps Revisited (w/ Math): Smoothing

✓ Cancel high-frequencies by moving each vert towards its neighbor's weighted (uni or cotan) centroid, and then inflate to prevent shrinkage.



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



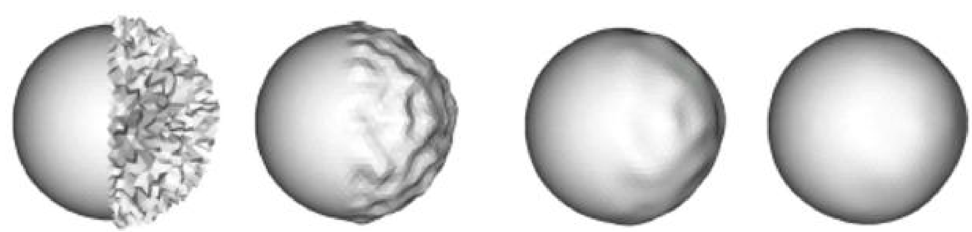
$$L(\mathbf{p}_i) = \frac{1}{\sum_{j\in N_i} w_{ij}}\left(\sum_{j\in N_i} w_{ij}\mathbf{p}_j\right) - \mathbf{p}_i$$

Iterate:

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

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

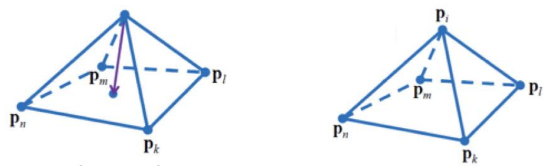
with $\lambda > 0$ and $\mu < 0$



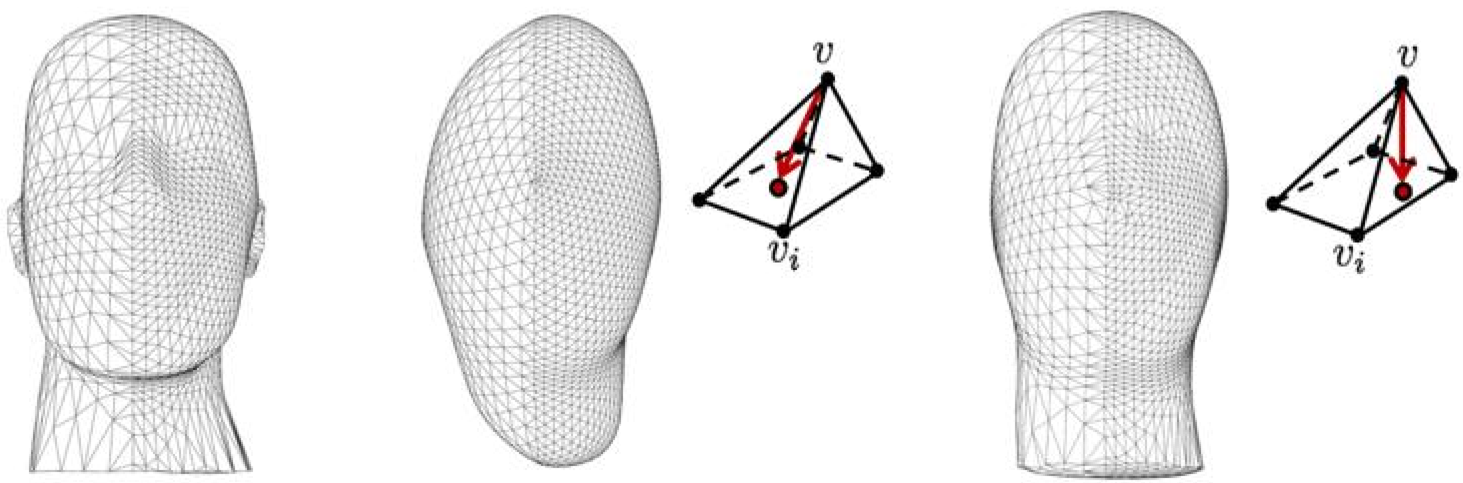
original 10 steps 50 steps 200 steps

Laplacian Apps Revisited (w/ Math): Smoothing

- ✓ Cancel high-frequencies by moving each vert towards its neighbor’s weighted (uni or cotan) centroid, and then inflate to prevent shrinkage.



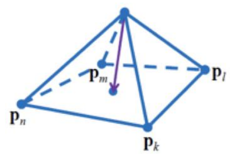
$$\frac{1}{|N_i|}\left(\sum_{j\in N_i}\mathbf{p}_j\right)-\mathbf{p}_i \qquad L(\mathbf{p}_i)=\frac{1}{\sum_{j\in N_i} w_{ij}}\left(\sum_{j\in N_i} w_{ij}\mathbf{p}_j\right)-\mathbf{p}_i$$

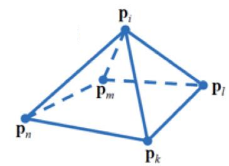


original · Uniform weights · Cotangent weights

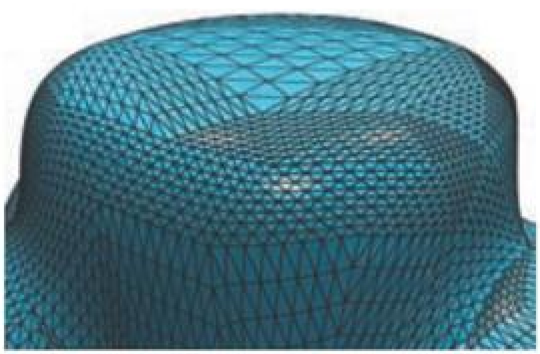
Laplacian Apps Revisited (w/ Math): Smoothing

✓ Cancel high-frequencies by moving each vert towards its neighbor's weighted (uni or cotan) centroid, and then inflate to prevent shrinkage.

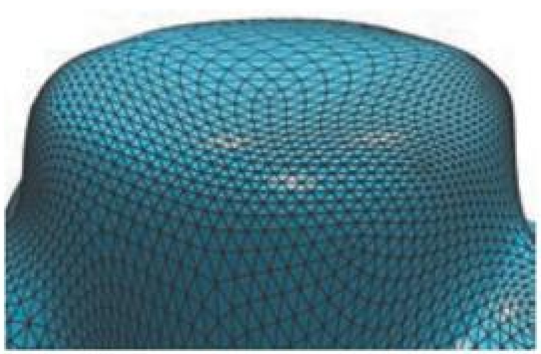




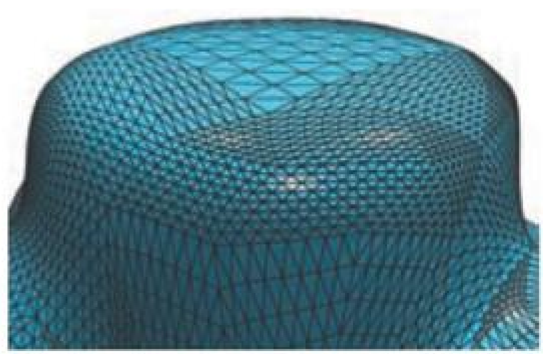
$$\frac{1}{|N_i|}\left(\sum_{j\in N_i}\mathbf{p}_j\right)-\mathbf{p}_i \qquad L(\mathbf{p}_i)=\frac{1}{\sum_{j\in N_i} w_{ij}}\left(\sum_{j\in N_i} w_{ij}\mathbf{p}_j\right)-\mathbf{p}_i$$



original



Uniform weights



Cotangent weights