

# **CENG 789 – Digital Geometry Processing**

## **09- Mesh Parameterization**

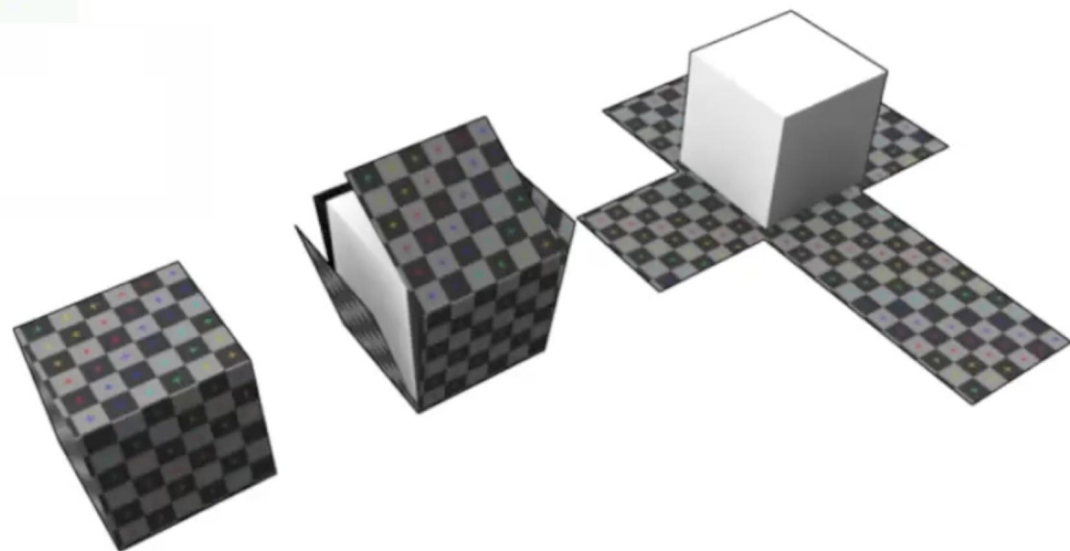
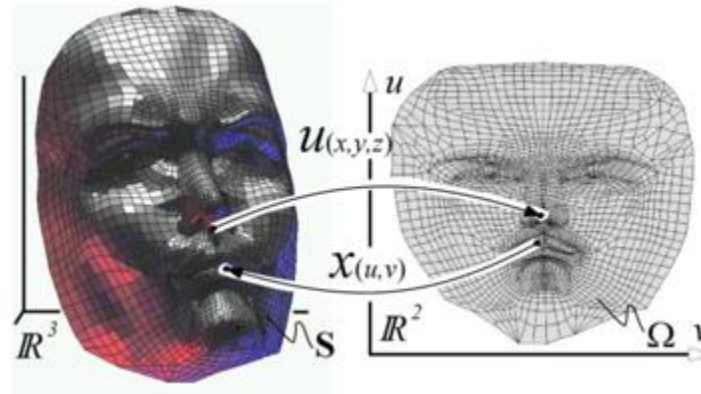
**Prof. Dr. Yusuf Sahillioğlu**

Computer Eng. Dept,  MIDDLE EAST TECHNICAL UNIVERSITY, Turkey

# Mesh Parameterization

2 / 77

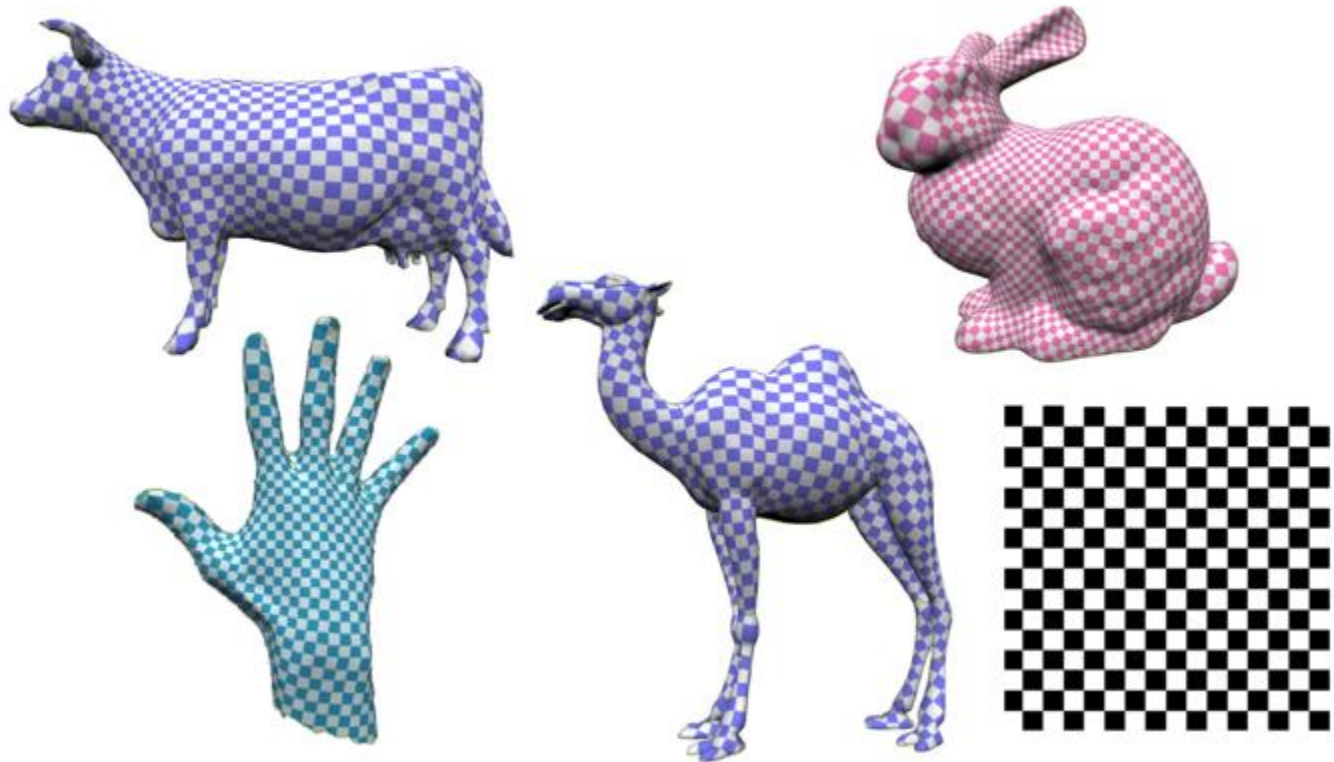
- ✓ Given a surface mesh in 3D, and a domain (mostly in 2D, e.g., for texture mapping), find a bijective mapping between them.



# Mesh Parameterization

3 / 77

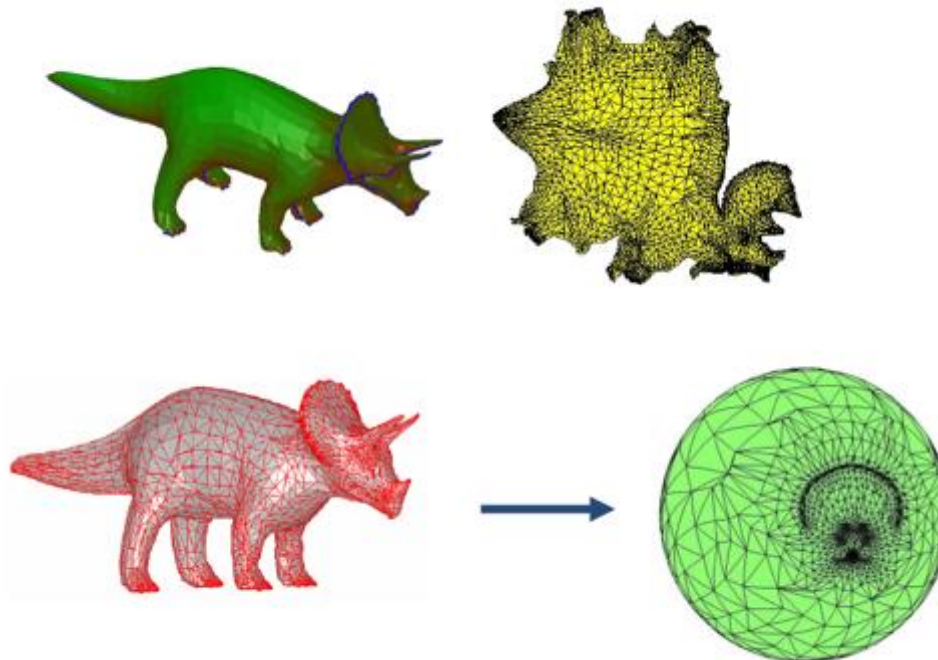
- ✓ Motivation: parameterization enables transferring of nice features from one domain to the other, e.g., main app: Texture Mapping: transfer of textures from 2D to the surface in 3D.



# Mesh Parameterization

4 / 77

- ✓ Motivation: parameterization enables transferring of nice features from one domain to the other, e.g., main app: Texture Mapping: transfer of textures from 2D to the surface in 3D.
- ✓ Top: more useful parameterization, e.g., texture mapping.
- ✓ Bottom: spherical parameterization (more in Slides 63-71).



# Mesh Parameterization

5 / 77

- ✓ Motivation: parameterization enables transferring of nice features from one domain to the other, e.g., main app: Texture Mapping: transfer of textures from 2D to the surface in 3D.
- ✓ Top: more useful parameterization, e.g., texture mapping.
- ✓ Bottom: spherical parameterization.
  - ✓ Apps for spherical parameterization.

Correspondence:



Morphing (map  
computed by param.):

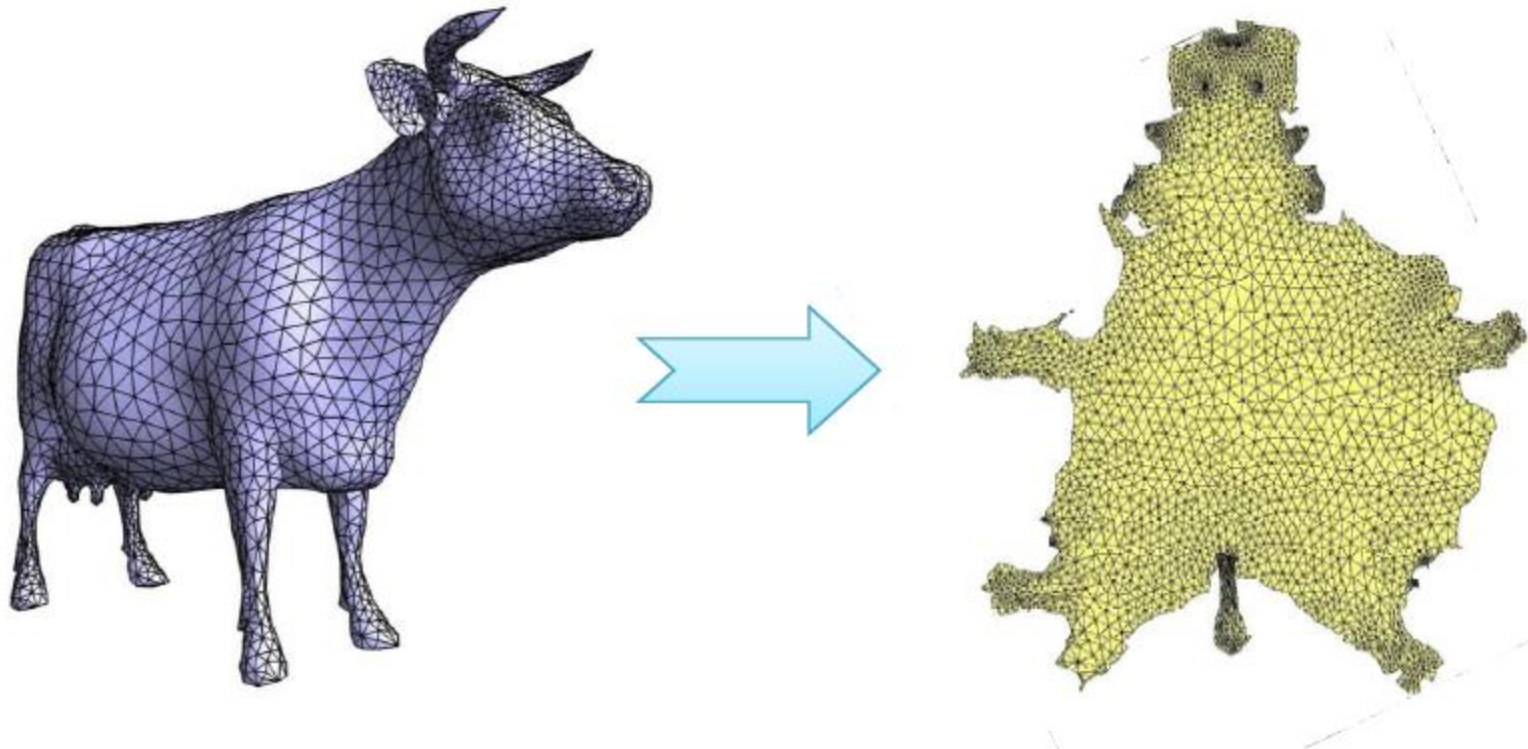




# Mesh Parameterization

6 / 77

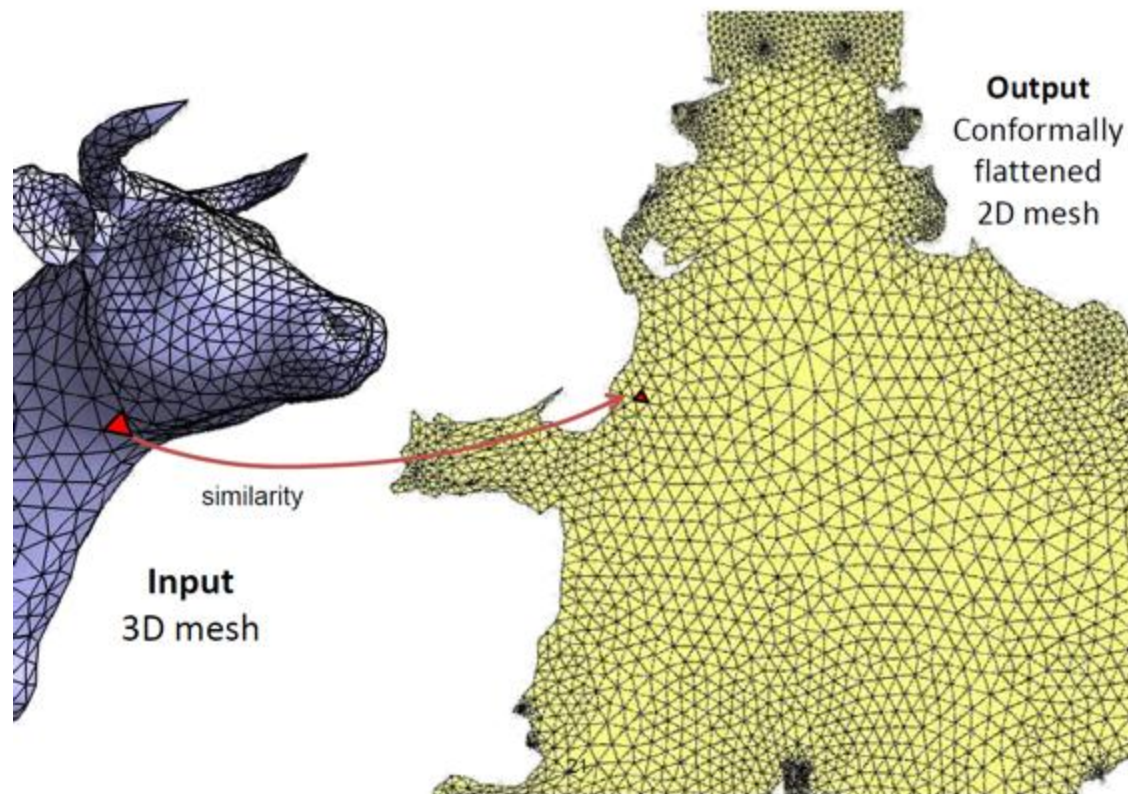
- ✓ Motivation: parameterization enables transferring of nice features from one domain to the other, e.g., main app: Texture Mapping: transfer of textures from 2D to the surface in 3D.



# Mesh Parameterization

7 / 77

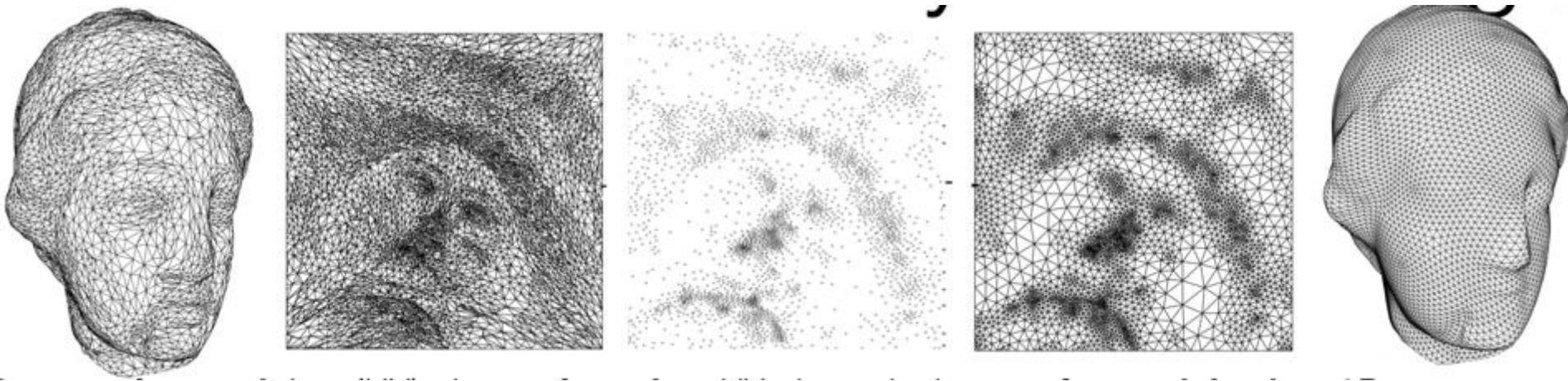
- ✓ Motivation: parameterization enables transferring of nice features from one domain to the other, e.g., main app: Texture Mapping: transfer of textures from 2D to the surface in 3D.



# Remeshing

8 / 77

- ✓ Unfold/flatten/unwrap/parameterize a surface mesh in 3D to 2D plane.
- ✓ Pull back whatever structure you need: texture, vector field, trianglition, etc. Here we pull back triangulation for remeshing application.



- ✓ Parametrize mesh into (UV) plane, triangulate UV plane nicely (e.g., Delaunay, self-design\*), transfer mesh back to 3D.
- ✓ <http://asdf.geometry.caltech.edu/pubs/AMD02.pdf>

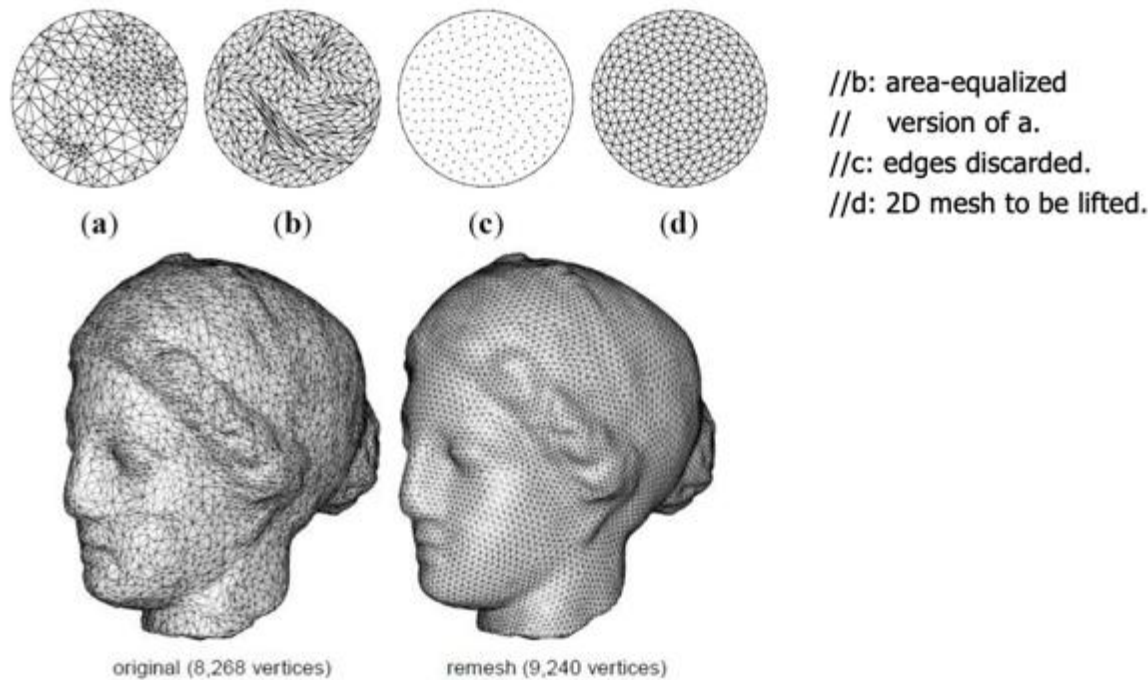
\* Minimize post-transfer distortion: detect creases in original mesh, constrain these to be edges in triangulation. Increase sampling in important, e.g., high-curvature, regions. And so on.



# Remeshing

9 / 77

- ✓ Unfold/flatten/unwrap/parameterize a surface mesh in 3D to 2D plane.
- ✓ Pull back whatever structure you need: texture, vector field, trianglition, etc. Here we pull back triangulation for remeshing application.

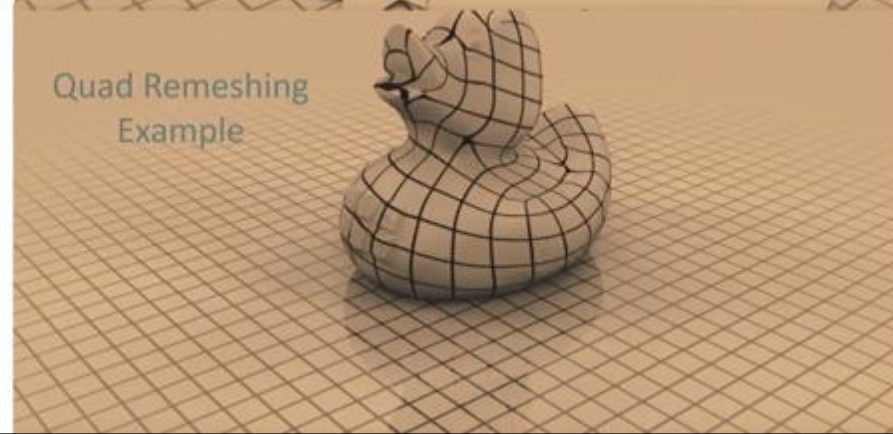
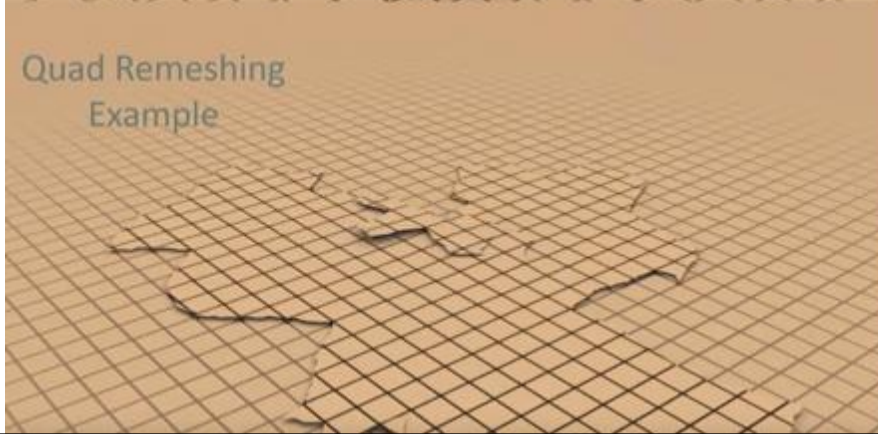
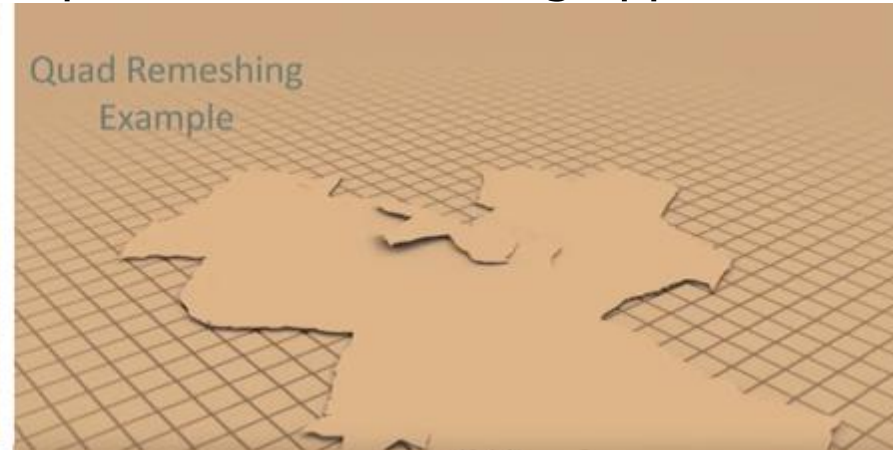


- ✓ Parametrize mesh into (UV) plane, triangulate UV plane nicely (e.g., area-equalized), transfer mesh back to 3D. Explicit surface remeshing, SGP'2003.

# Remeshing

10 / 77

- ✓ Unfold/flatten/unwrap/parameterize a surface mesh in 3D to 2D plane.
- ✓ Pull back whatever structure you need: texture, vector field, grid pattern, etc. Here we pull back grid pattern for remeshing application.



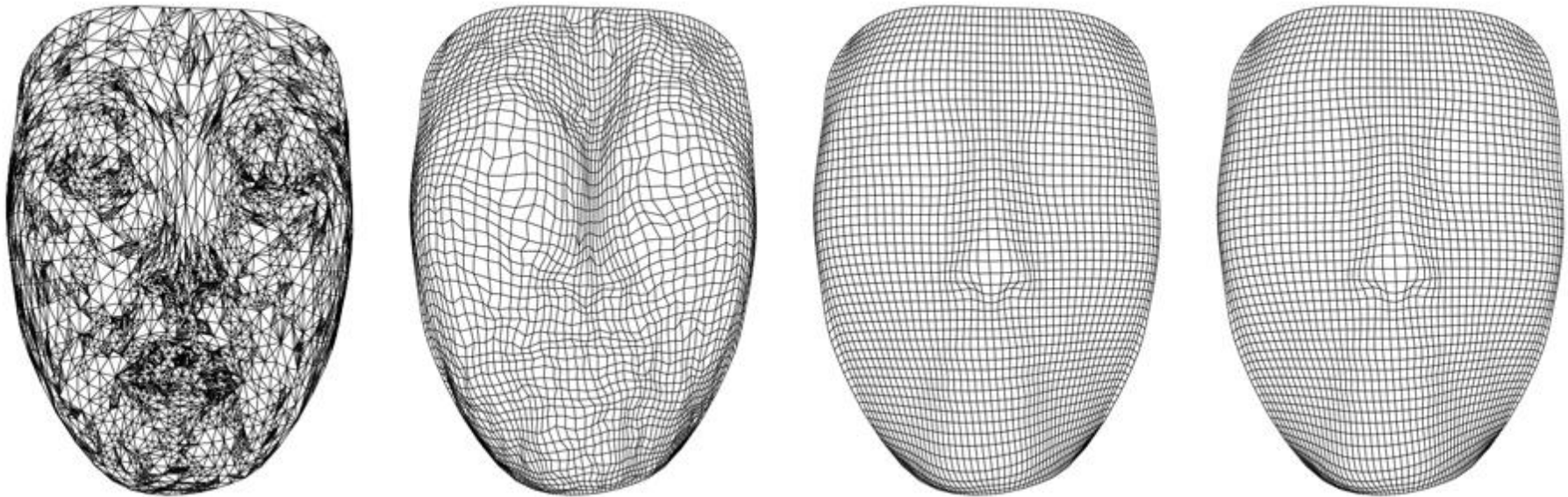
# Remeshing

11 / 77

- ✓ Unfold/flatten/unwrap/parameterize a surface mesh in 3D to 2D plane.
- ✓ Pull back whatever structure you need: texture, vector field, grid pattern, etc. Here we pull back grid pattern for remeshing application.

Input

Outputs w/ 3 different methods



- ✓ Find the 3D coordinate of each grid vertex  $g$ : Suppose 2D triangle  $t$  contains  $g$ . Find barycentric coords of  $g$  in  $t$ . Apply these barycentric coords to the 3D vertex coordinates of 3D version of  $t$ .



# Texture Mapping

12 / 77

- ✓ Unfold/flatten/unwrap/parameterize a surface mesh in 3D to 2D plane.
- ✓ Pull back whatever structure you need: texture, vector field, grid pattern, etc. Here we pull back color for texture mapping application.

Base skin texture.



Artist paints in some highlights & make-up.

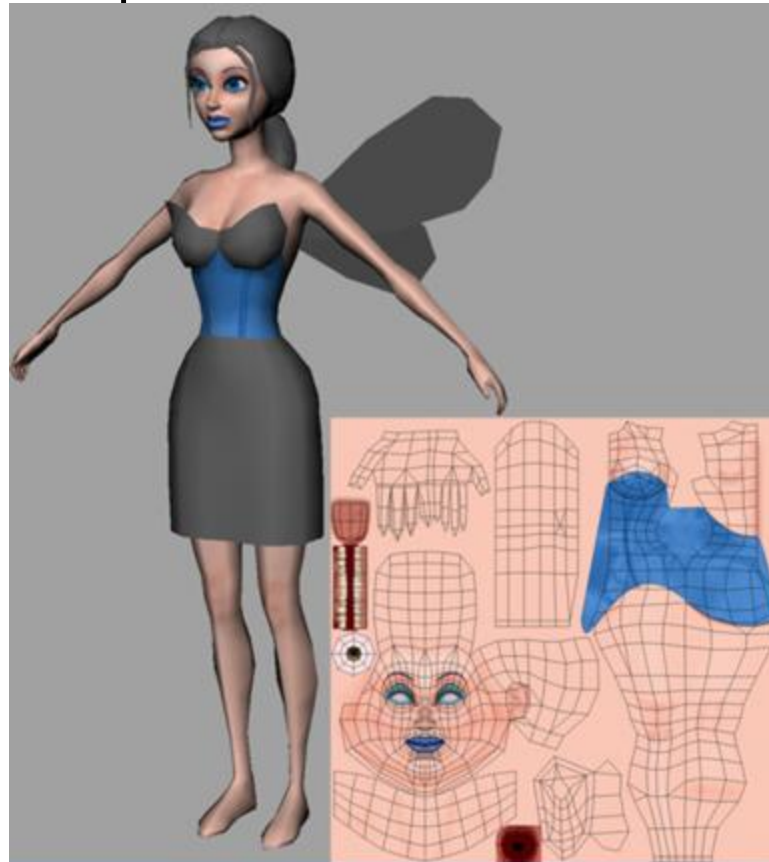




# Texture Mapping

13 / 77

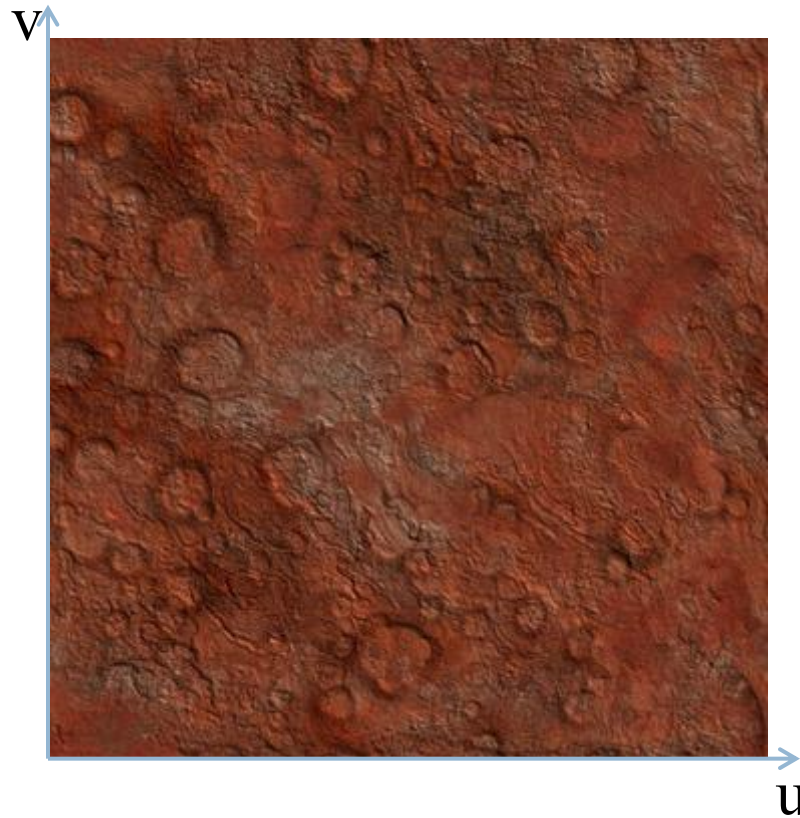
- ✓ Unfold/flatten/unwrap/parameterize a surface mesh in 3D to 2D plane.
- ✓ Pull back whatever structure you need: texture, vector field, grid pattern, etc. Here we pull back color for texture mapping application.



# Texture Mapping Roadmap

14 / 77

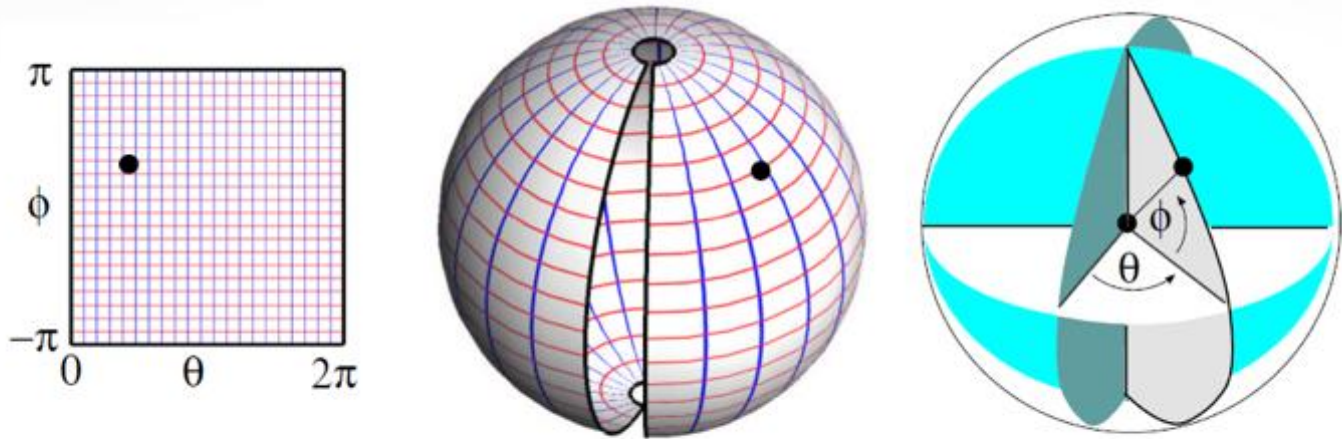
- ✓ Step 1: Associate an  $(u, v)$  coordinate system with the texture image where  $(u, v) \in [0,1] \times [0,1]$



# Texture Mapping Roadmap

15 / 77

- ✓ Step 2: **Parameterize** the surface to be texture mapped using two coordinates. For instance, a sphere can be parametrized by 2 params:

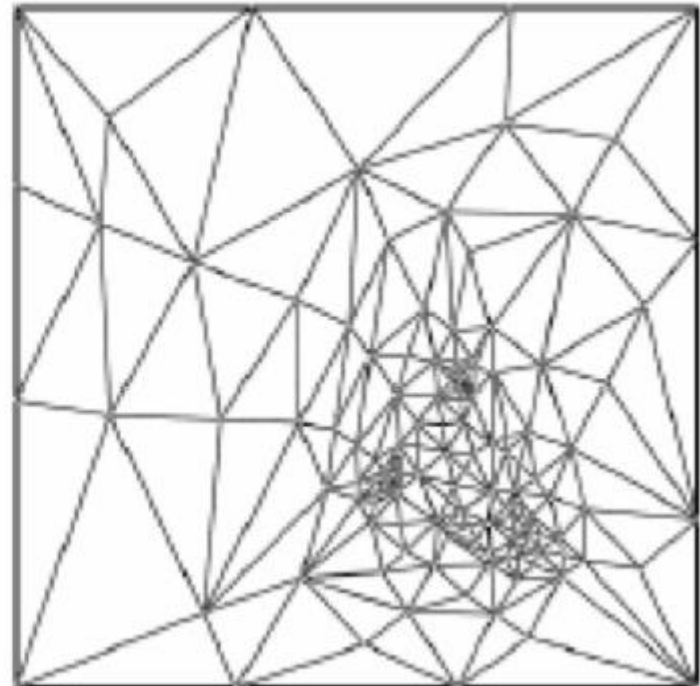
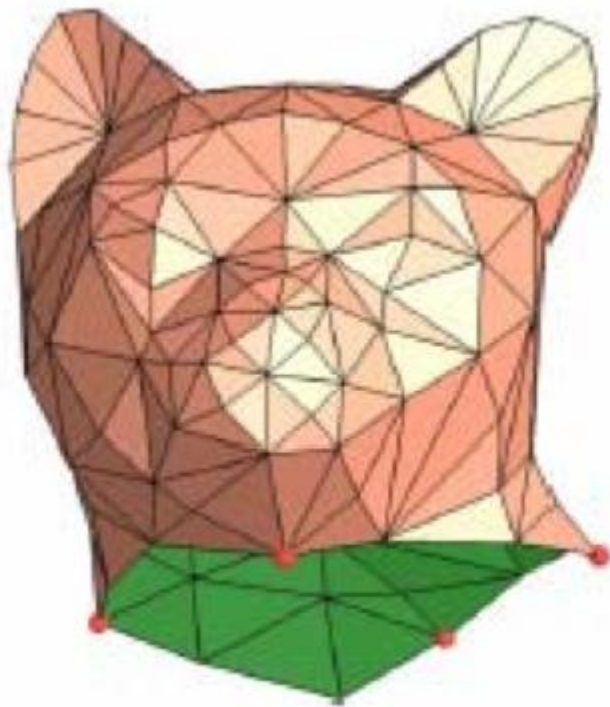


$$\begin{bmatrix} \theta \\ \phi \end{bmatrix} \mapsto \begin{bmatrix} \sin \theta \sin \phi \\ \cos \theta \sin \phi \\ \cos \phi \end{bmatrix}$$

# Texture Mapping Roadmap

16 / 77

- ✓ Step 2: We will learn this **parameterization** for generic shapes, where there is no analytical representation like the sphere case.

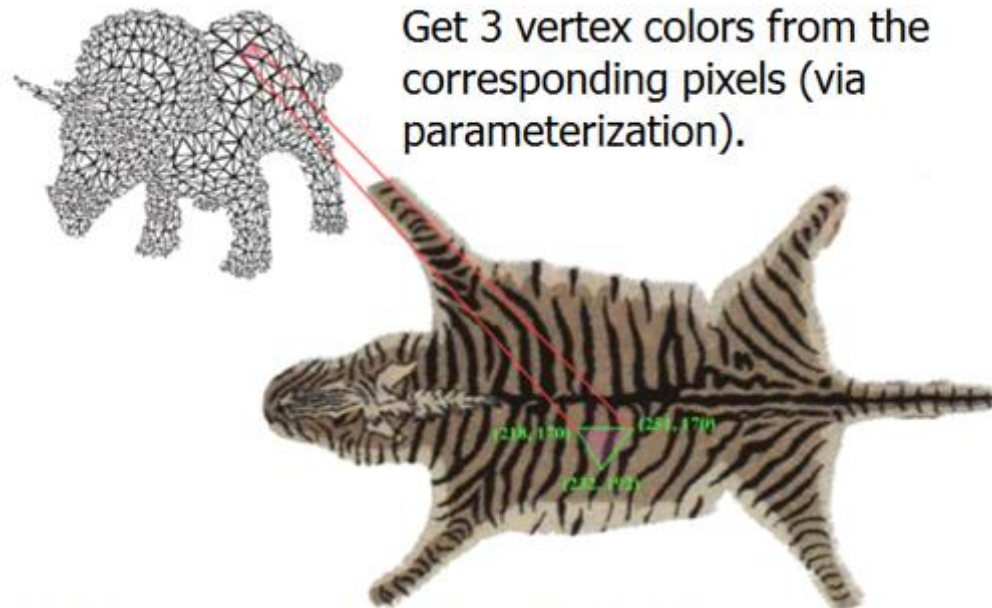




# Texture Mapping Roadmap

17 / 77

- ✓ Step 2: We will learn this **parameterization** for generic shapes, where there is no analytical representation like the sphere case.



For a non-vertex triangle point, find its barycentric coordinate in 3D; apply the same barycentric coords to the corresponding 2D green triangle.

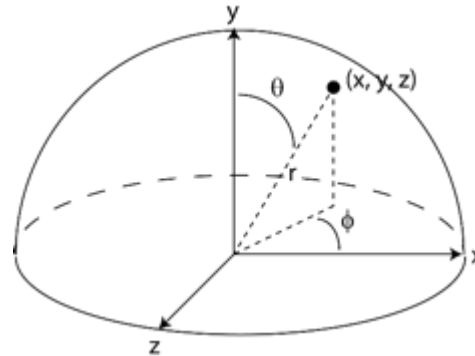
# Texture Mapping Roadmap

18 / 77

- ✓ Step 3: Compute a  $(u, v)$  value for every surface point. For instance, a sphere:

$$u = \phi / (2\pi)$$

$$v = (\pi - \Theta) / \pi = 1 - \Theta / \pi$$



- ✓ Step 4: Find the texture image coordinate  $(i, j)$  at the given  $(u, v)$  coordinate:

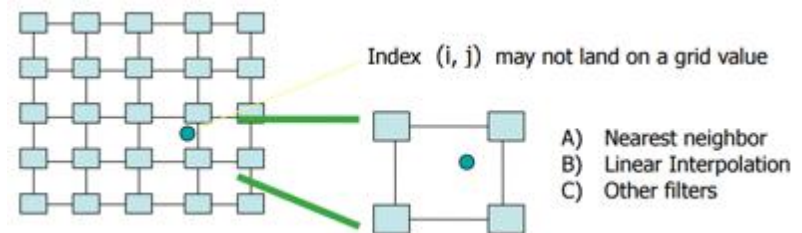
$$i = u \cdot n_x$$

$$j = v \cdot n_y$$

Note that  $i, j$  can be fractional!

$n_x$  = texture image width

$n_y$  = texture image height



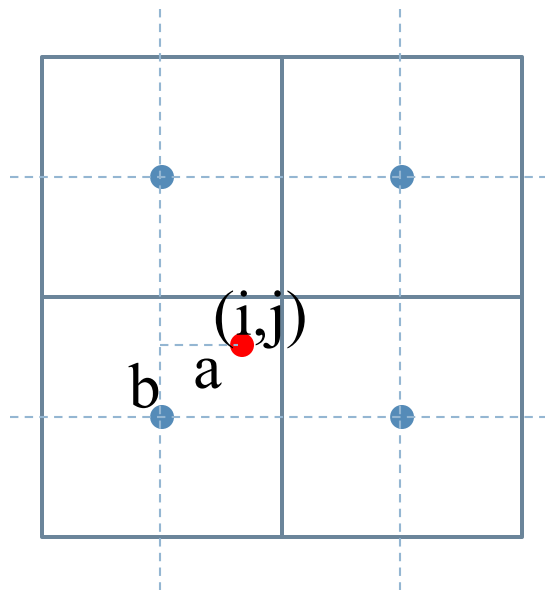
# Texture Mapping Roadmap

19 / 77

- ✓ Step 5: Choose the texel color using a suitable interpolation strategy.
- ✓ Nearest Neighbor: fetch texel at nearest coordinate

$$\text{Color}(x, y, z) = \text{fetch}(\text{round}(i, j))$$

- ✓ Bilinear Interpolation: Average four closest neighbors:



$$\begin{aligned} \text{Color}(x, y, z) = & \text{fetch}(\mathbf{round(i, j)}).(1 - a).(1 - b) + \\ & \text{fetch}(\mathbf{round(i+1, j)}).(a).(1 - b) + \\ & \text{fetch}(\mathbf{round(i, j+1)}).(1 - a).(b) + \\ & \text{fetch}(\mathbf{round(i+1, j+1)}).(a).(b) \end{aligned}$$

# Texture Mapping Roadmap

20 / 77

- ✓ Step 5: Choose the texel color using a suitable interpolation strategy.
- ✓ Nearest Neighbor: fetch texel at nearest coordinate  
$$\text{Color}(x, y, z) = \text{fetch}(\text{round}(i, j))$$
- ✓ Bilinear Interpolation: Average four closest neighbors:

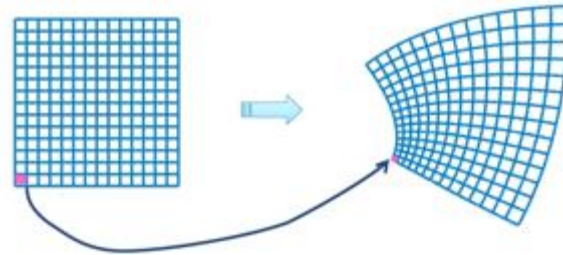




# Parameterization Types

21 / 77

- ✓ Conformal (angle preserving): if the angle of every pair of intersecting curves on surface is the same as that of the corresponding preimages.

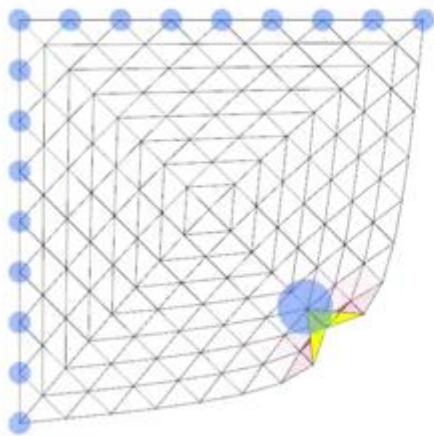


- ✓ Equiareal (area preserving): if every part of one domain is mapped onto a part in the other domain w/ the same area.
- ✓ Isometric (length preserving): if the length of any arc on surface is the same as that of its preimage. Isometric = conformal + equiareal.
- ✓ Overall deviation from these features (angle, area, length) defines the parameterization distortion (Slide 75).

# Parameterization Bijection

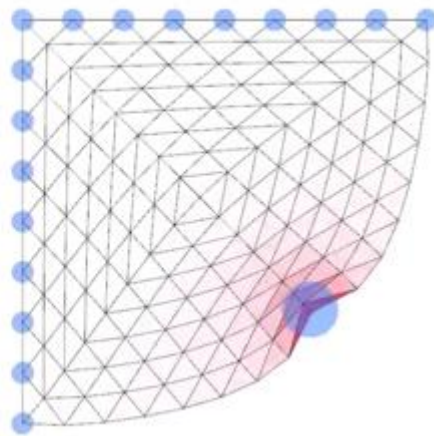
22 / 77

- ✓ Bijective parameterization: each mapped 2D point corresponds to exactly one 3D point.
- ✓ Desired property.
  - ✓ Non-bijective one will, for instance, give the same texture color (at 2D coordinate  $c$ ) to two distant 3D points that correspond to  $c$ .

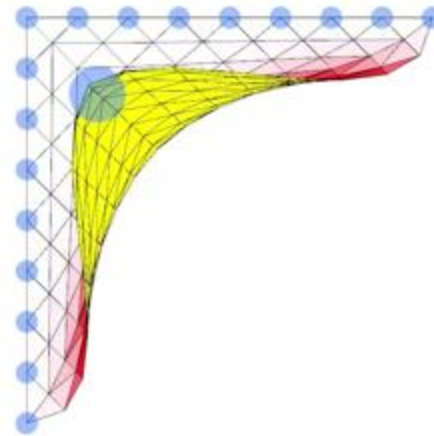


Local overlap

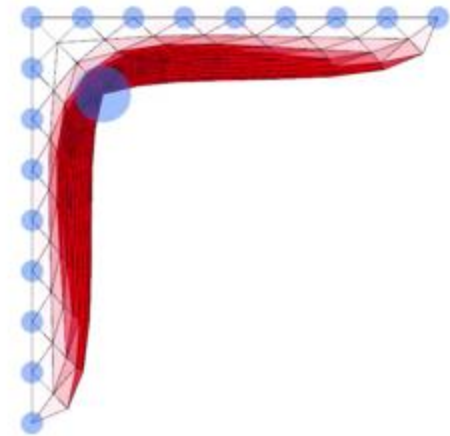
(normal of the highlighted flipped triangle is inverted w.r.t. the other triangle normals).



No local overlap



Local overlap

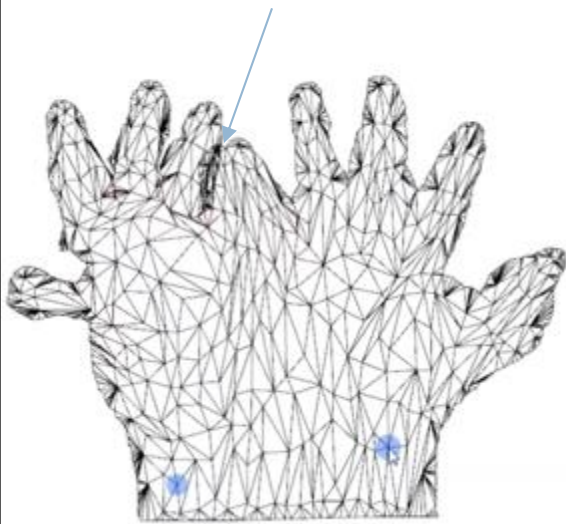


No local overlap

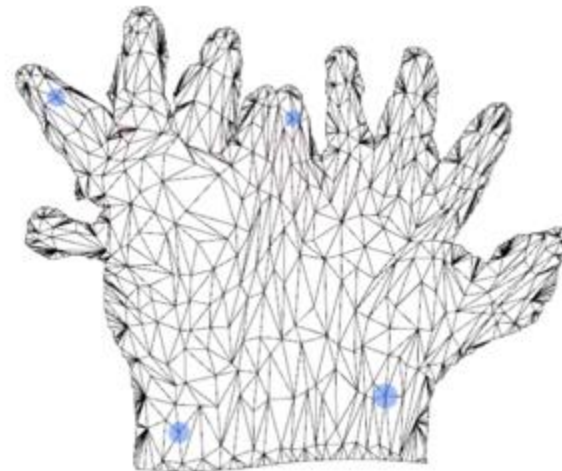
# Parameterization Bijection

23 / 77

- ✓ Bijective parameterization: each mapped 2D point corresponds to exactly one 3D point.
- ✓ Desired property.
  - ✓ Non-bijective one will, for instance, give the same texture color (at 2D coordinate  $c$ ) to two distant 3D points that correspond to  $c$ .



Global overlap.



Global overlap fixed.



# Linear Parameterization Methods

24 / 77

- ✓ In the next parameterization approaches, we will see 2 stages:
  - ✓ Boundary vertices of the mesh are mapped to the boundary of a convex region in 2D, e.g., disk.
  - ✓ Then the positions of the remaining vertices are obtained by solving a linear system:  $Wx = b \rightarrow x = W^{-1}b$ .
- ✓ We won't cover non-linear methods that are typically slower but more accurate in terms of minimizing parameterization distortion (Slide 75).
  - ✓ Read: Mesh Parameterization Methods and Their Applications.



# Disk Parameterization

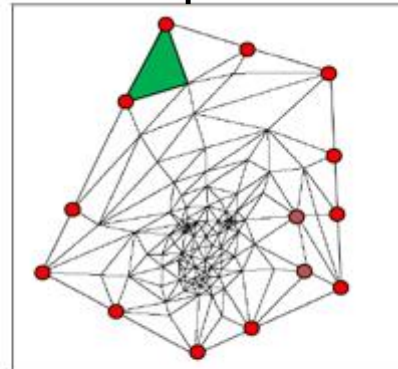
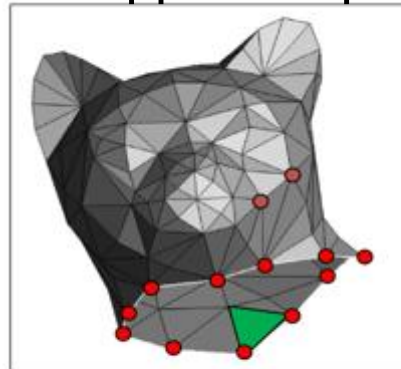
25 / 77

- ✓ Common to all methods: boundary loop vertices  $\rightarrow$  convex curve.
- ✓ Use a circle of radius  $r$  for an automatic placement as follows, respecting chord lengths.
- ✓ Consider boundary loop vertices:  $V_B = \{b_1, b_2, \dots, b_n\}$ .
  - ✓ Start w/ a boundary edge, visit an endpoint  $v$ , advance to an unvisited 1-ring neighbor of  $v$ , repeat. We have an order now ☺.
- ✓ Consider lengths of boundary edges:  $\|e_i\| = \|b_{i+1} - b_i\|$ .
  - ✓ The sector angle (how much angle an edge occupies on the circle; think pie diagrams) allocated to each edge is:  $\psi_i = \frac{2\pi \|e_i\|}{\sum_i \|e_i\|}$ .
- ✓ Desired uv coords for  $b_i$ :  $(u_i, v_i) = r (\cos(\Psi_i), \sin(\Psi_i))$ ,  $\Psi_i = \sum_{j=1}^i \psi_j$ .
- ✓ Note that, the last vertex goes to  $(r, 0)$ .
  - ✓ Sum of angles is  $2\pi$ , leading to  $r(\cos 2\pi, \sin 2\pi) = r(1, 0) = (r, 0)$ .

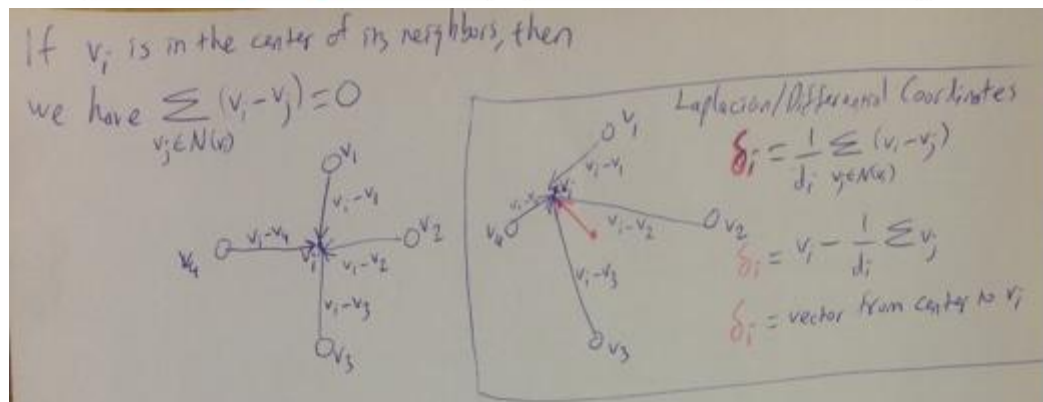
# Disk Parameterization

26 / 77

- ✓ Method # 1: map with uniform weights.
- ✓ Sets the new coordinate of non-boundary vertices as the simple center of the vertices of its 1-ring neighborhood.
- ✓ Bijective: each mapped 2D point corresponds to exactly one 3D point.



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



# Disk Parameterization

27 / 77

- ✓ Method # 1: map with uniform weights.
- ✓ Sets the new coordinate of non-boundary vertices as the simple center of the vertices of its 1-ring neighborhood.

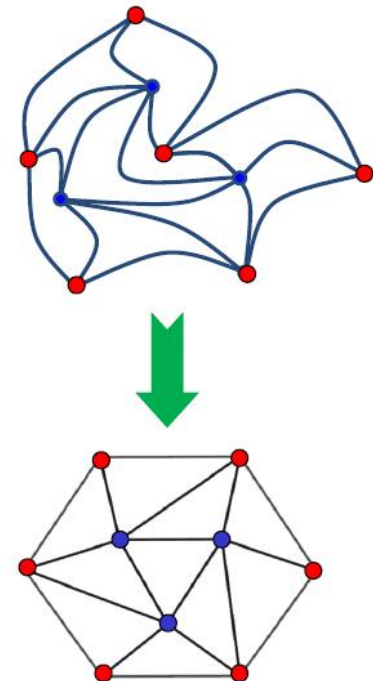
Fix 2D boundary to **convex polygon**.

Define drawing as a solution of

$$\begin{aligned} Wx &= b_x \\ Wy &= b_y \end{aligned} \quad w_{ij} = \begin{cases} > 0 & (i, j) \in E \quad i \notin B \\ -\sum_{j \neq i} w_{ij} & (i, i), i \notin B \\ 1 & (i, i), i \in B \\ 0 & \text{otherwise} \end{cases}$$

Weights  $w_{ij}$  control triangle shapes

$B$  = Boundary vertices



# Disk Parameterization

28 / 77

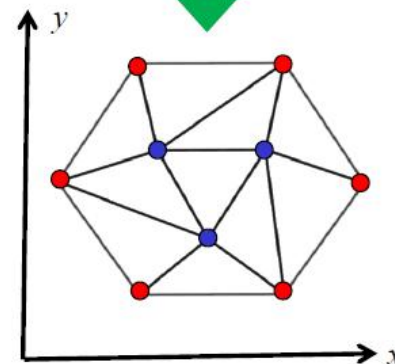
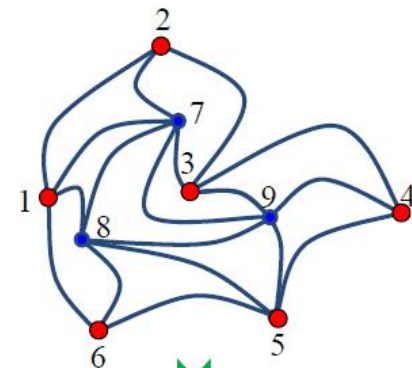
- ✓ Method # 1: map with uniform weights.
- ✓ Sets the new coordinate of non-boundary vertices as the simple center of the vertices of its 1-ring neighborhood.

$$w_{ij} = 1$$

$$W = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & -5 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & -5 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & -5 \end{pmatrix}$$

$$b_x = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 3 \\ 2 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

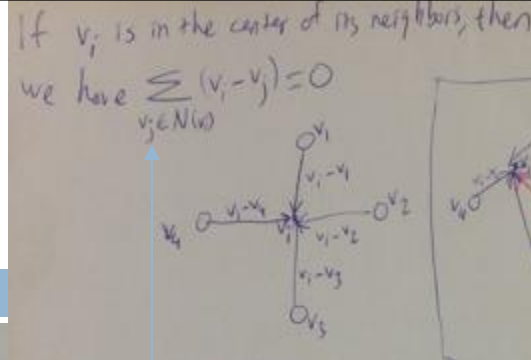
$$b_y = \begin{pmatrix} 2 \\ 3 \\ 3 \\ 2 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$



# Disk Parameterization

29 / 77

- ✓ Method # 1: map with
- ✓ Sets the new coordinate center of the vertices of

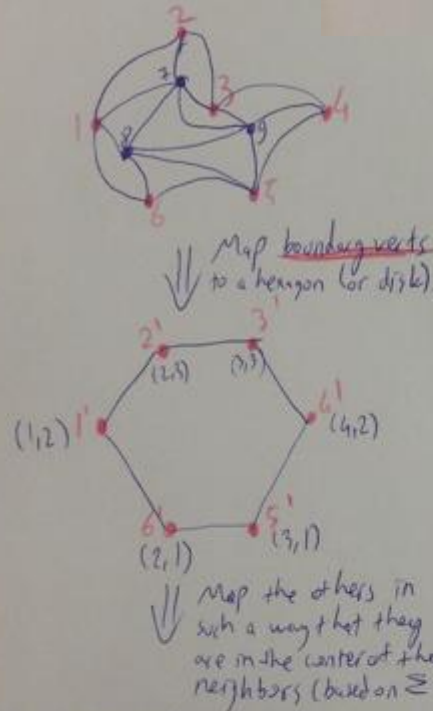
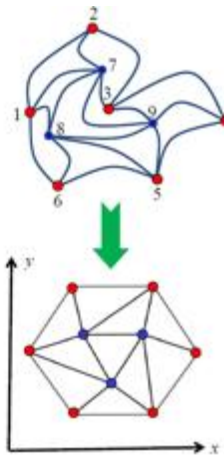


$$w_{ij} = 1$$

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & -5 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & -5 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & -5 \end{bmatrix}$$

$$b_x = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 3 \\ 2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$b_y = \begin{bmatrix} 2 \\ 3 \\ 3 \\ 2 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



$$W \begin{bmatrix} v_1^x \\ v_2^x \\ v_3^x \\ v_4^x \\ v_5^x \\ v_6^x \\ v_7^x \\ v_8^x \\ v_9^x \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 3 \\ 2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$v_7^x$  must be in the center of  $v_1^x, v_2^x, v_3^x, v_4^x, v_5^x$

$$\Rightarrow (v_1^x - v_7^x) + (v_2^x - v_7^x) + (v_3^x - v_7^x) + (v_4^x - v_7^x) + (v_5^x - v_7^x) = 0$$

$$(-v_7^x + v_1^x) + (-v_7^x + v_2^x) + (-v_7^x + v_3^x) + (-v_7^x + v_4^x) + (-v_7^x + v_5^x) = 0$$

Similarly  $v_8^x$  &  $v_9^x$  must be in their respective centers

$$\Rightarrow (v_1^x - v_8^x) + (v_2^x - v_8^x) + (v_6^x - v_8^x) + (v_3^x - v_8^x) + (v_5^x - v_8^x) = 0$$

$$\Rightarrow (v_3^x - v_9^x) + (v_4^x - v_9^x) + (v_5^x - v_9^x) + (v_6^x - v_9^x) + (v_7^x - v_9^x) = 0$$

Boundary vertices know their  $x$  (and  $y$ ) coordinates already

$$\Rightarrow v_1^x = 1, v_2^x = 2, v_3^x = 3, v_4^x = 4, v_5^x = 3, v_6^x = 2$$

9 equations, 9 unknowns can be represented as a linear system, and solved by  $x^x = W^{-1} b^x$

// do the same for  $y$ -coordinates:  $x^y = W^{-1} b^y$

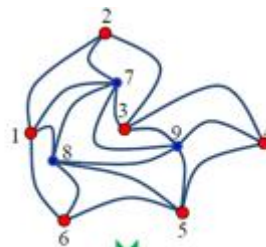


# Disk Parameterization

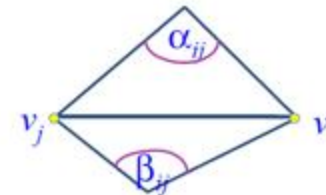
30 / 77

- ✓ Method # 2: map with harmonic (or cotangent) weights.
- ✓ Sets the new coordinate of non-boundary vertices as the weighted average of vertices of its 1-ring neighborhood.
- ✓ Angel-preserving parameterization achieved ☺.
- ✓ If mesh contains obtuse angles, the weights can be negative, and, as a result, parameterization can be non-bijective: 2D triangles overlap ☹.

$$w_{ij} \neq 1$$



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

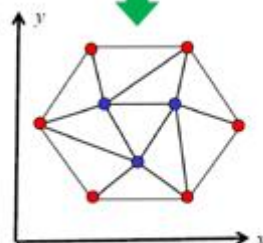


If  $ij$  is a border edge,  $w_{ij} = \cot \alpha_{ij} / 2$

$$W = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & -5 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & -5 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & -5 & 0 \end{pmatrix}$$

$$b_x = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 3 \\ 4 \\ 3 \\ 2 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$b_y = \begin{pmatrix} 2 \\ 3 \\ 3 \\ 2 \\ 2 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

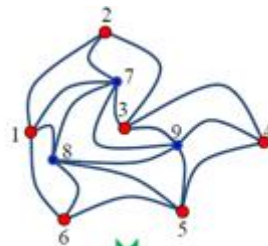


# Disk Parameterization

31 / 77

- ✓ Method # 3: map with mean-value weights.
- ✓ Sets the new coordinate of non-boundary vertices as the weighted average of vertices of its 1-ring neighborhood.
- ✓ Angel-preserving parameterization achieved ☺.
- ✓ Bijective 'cos weights are certainly positive inside convex polygons ☺.
- ✓ Weights vary non-smoothly as vertex move inside its 1-ring polygon ☹.
- ✓ In some cases introduce larger angular distortion than harmonic w's ☹.

$$w_{ij} \neq 1$$

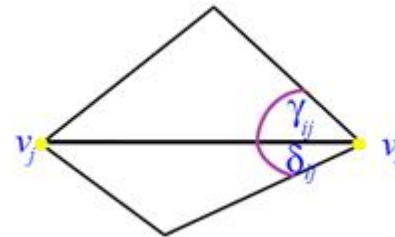
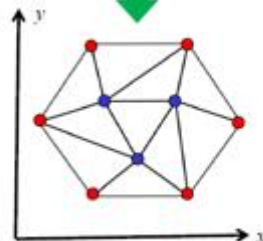


$$w_{ij} = \frac{\tan(\gamma_{ij} / 2) + \tan(\delta_{ij} / 2)}{2 \|v_i - v_j\|}$$

$$W = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & -5 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & -5 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & -5 \end{pmatrix}$$

$$b_x = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 3 \\ 4 \\ 3 \\ 3 \\ 2 \\ 0 \end{pmatrix}$$

$$b_y = \begin{pmatrix} 2 \\ 3 \\ 3 \\ 3 \\ 2 \\ 2 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

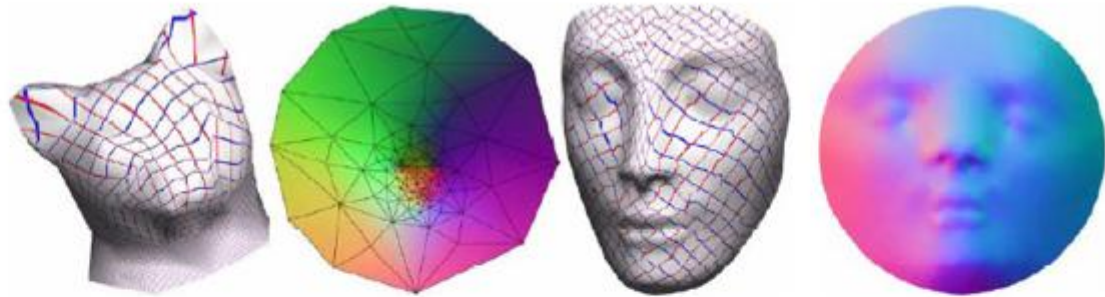


# Disk Parameterization

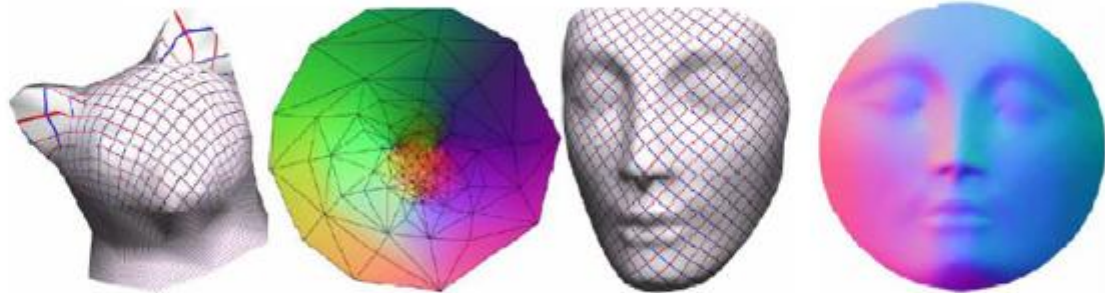
32 / 77

- ✓ Methods compared.

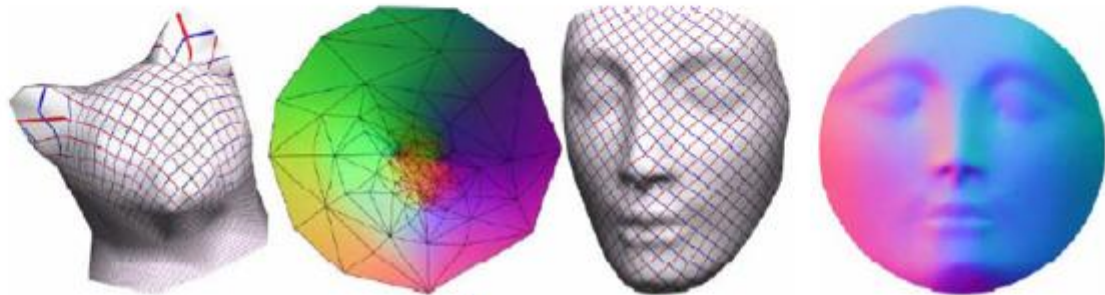
uniform



harmonic



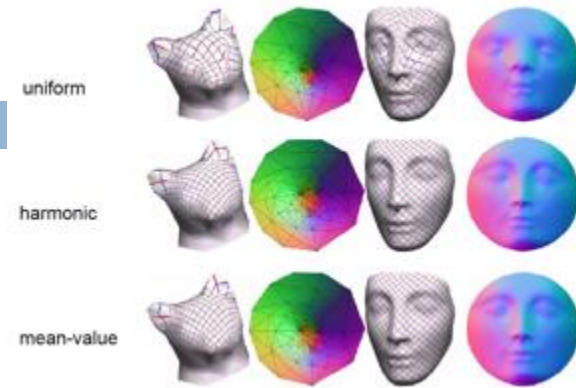
mean-value



# Disk Parameterization

33 / 77

- ✓ Why the distortions on grid lines of the texture?



- ✓ Didactic example: texture and mesh are



and



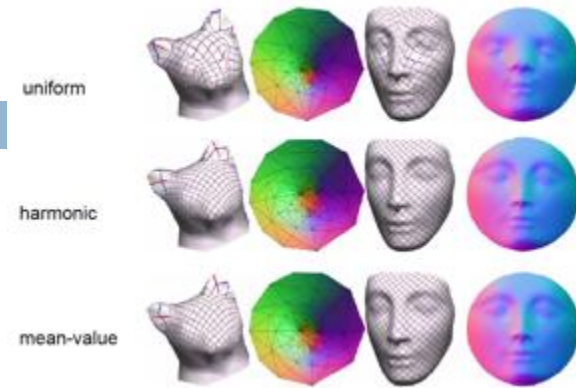
- ✓ Pyramid's (no floor) pole slightly moved so its projection not in center.
- ✓ 2D parameterization overlayed w/ texture (uniform, harmon, mean-val):



# Disk Parameterization

34 / 77

- ✓ Why the distortions on grid lines of the texture?



- ✓ Didactic example: texture and mesh are



and



- ✓ Pyramid's (no floor) pole slightly moved so its projection not in center.
- ✓ 3D w/ pulled-back texture (birdeye view); note the distortion at the left.

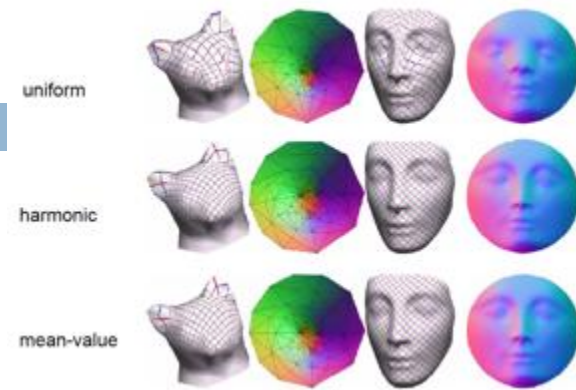




# Disk Parameterization

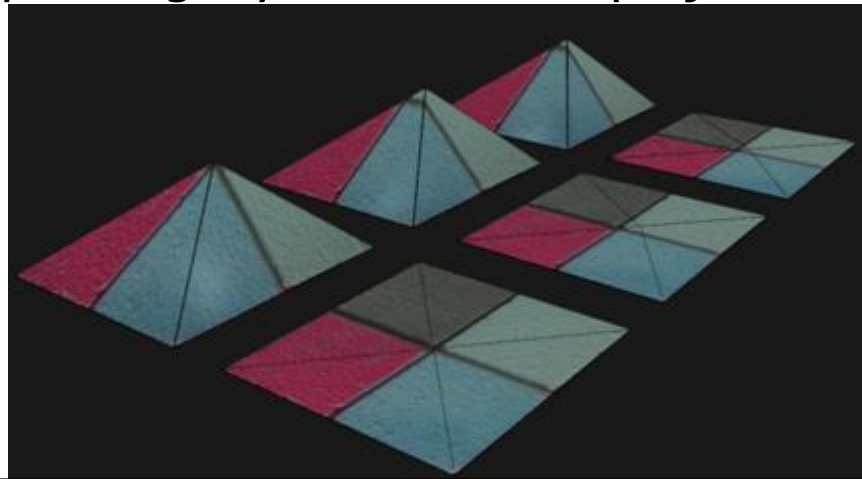
35 / 77

- ✓ Why the distortions on grid lines of the texture?



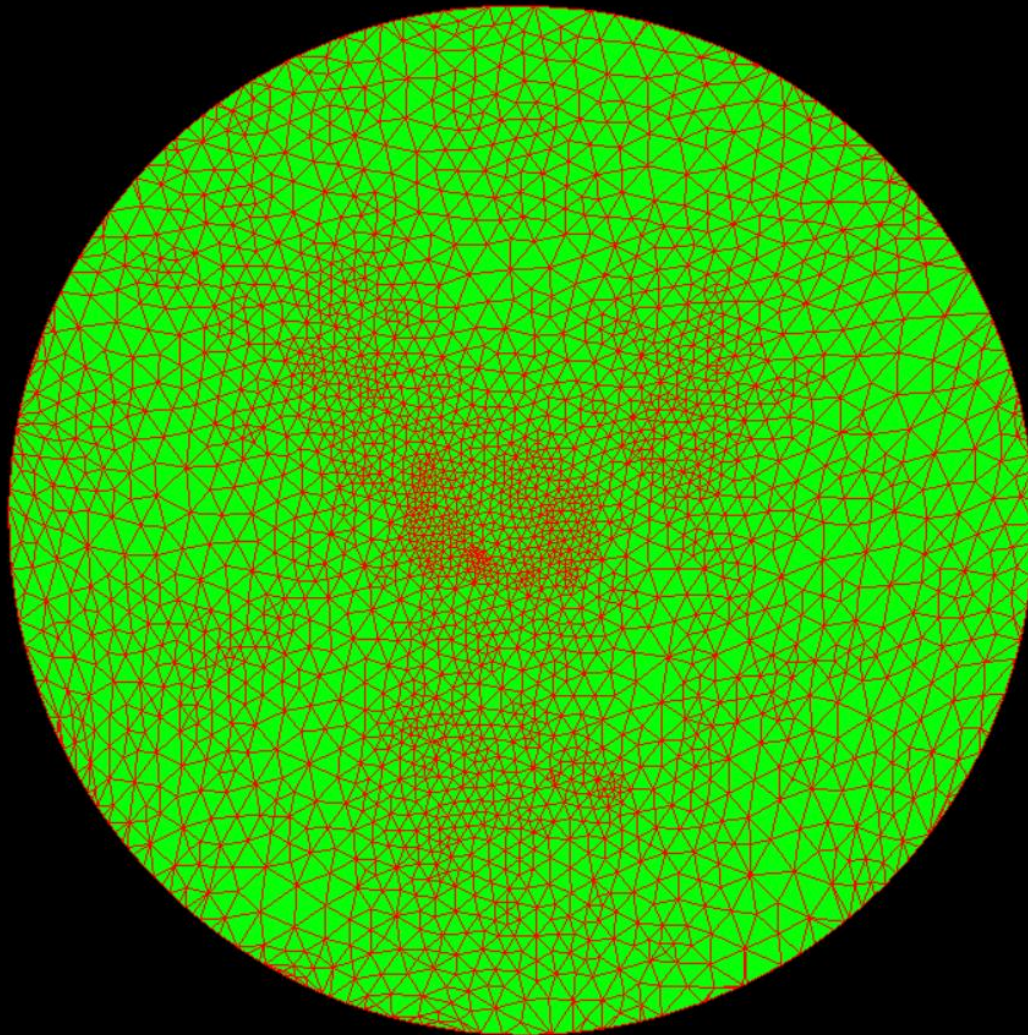
- ✓ Didactic example: texture and mesh are  and .

- ✓ Pyramid's (no floor) pole slightly moved so its projection not in center.
- ✓ Not so clear w/  
non-birdeye view:



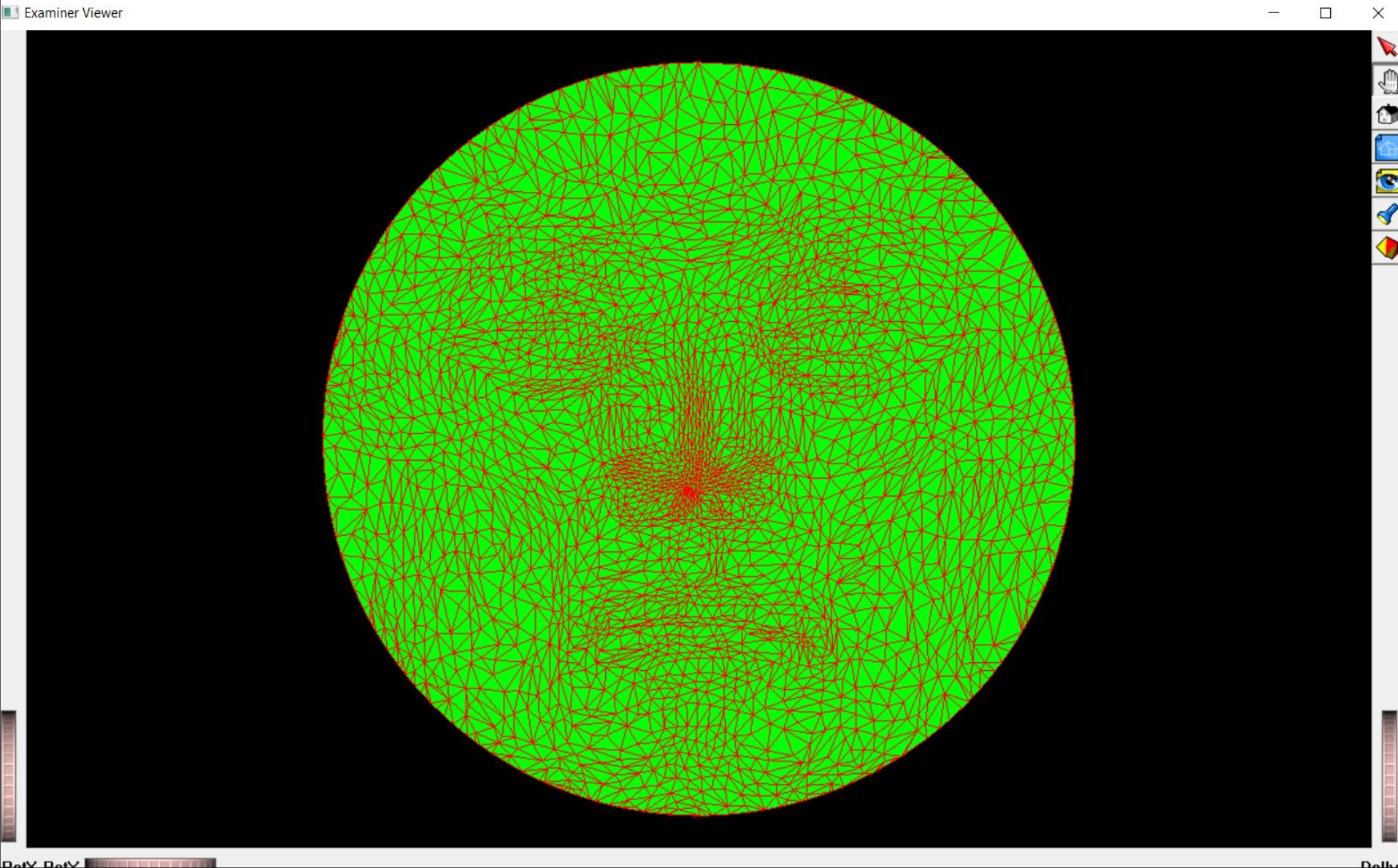
# Disk Parameterization // Uniform Weights

Examiner Viewer

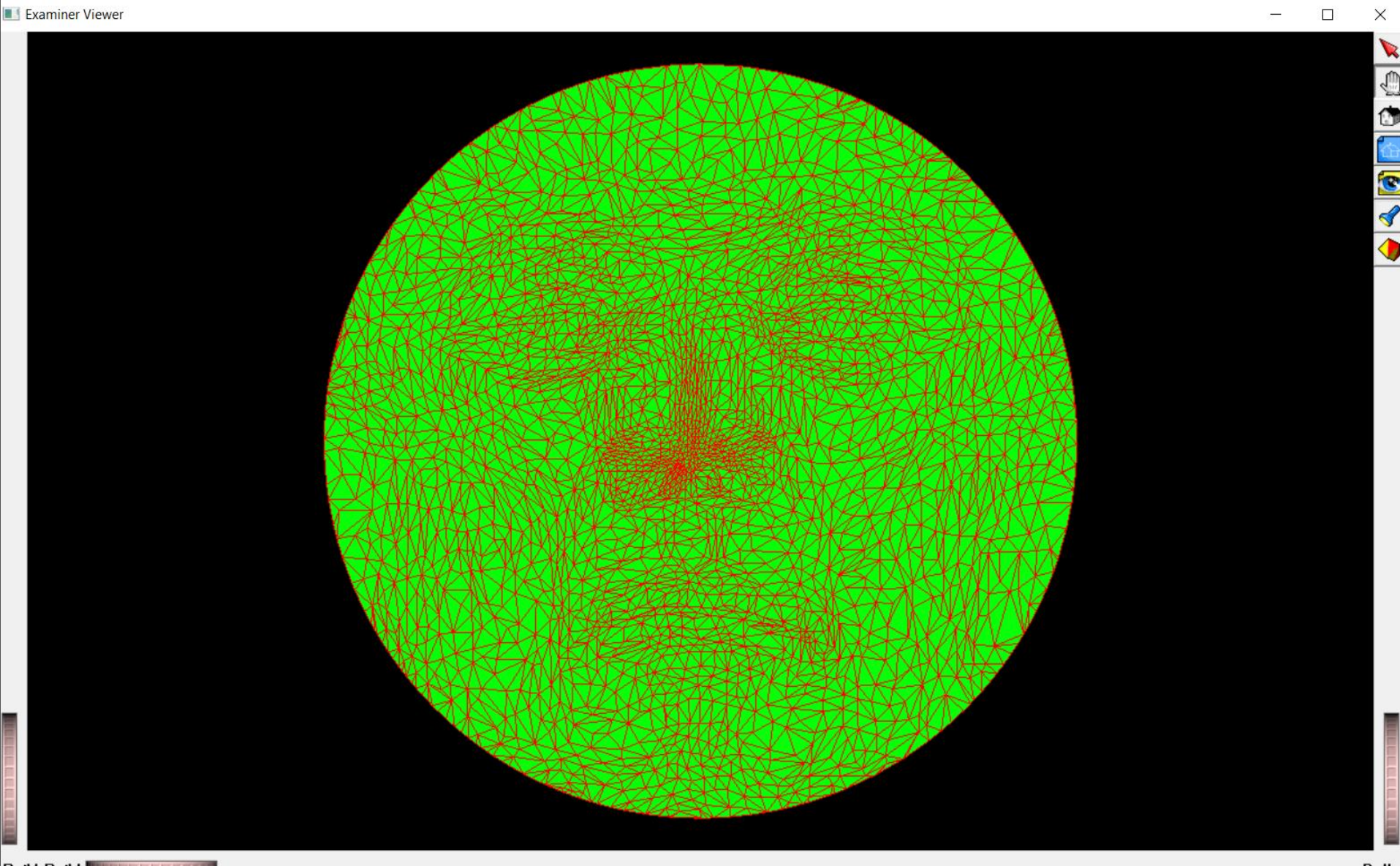




# Disk Parameterization // Harmonic (cot) Weights



# Disk Parameterization // Mean-value Weights

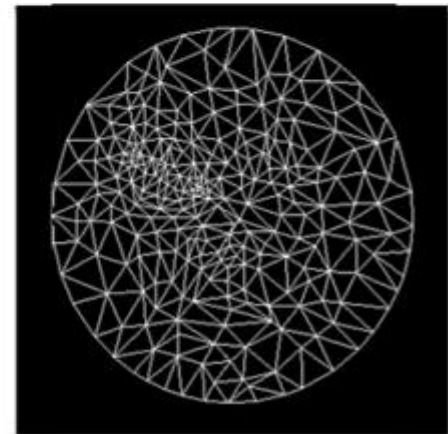
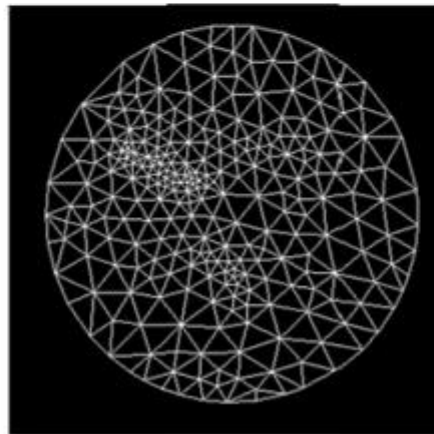
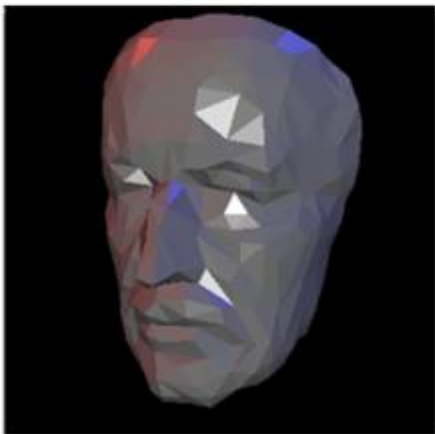
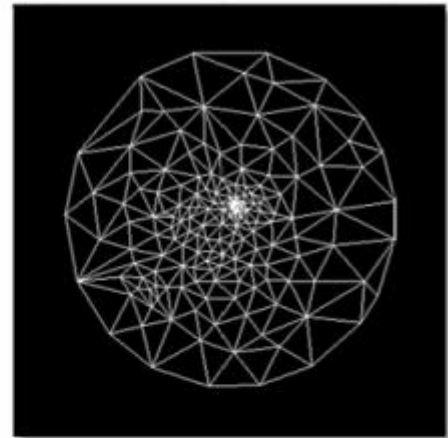
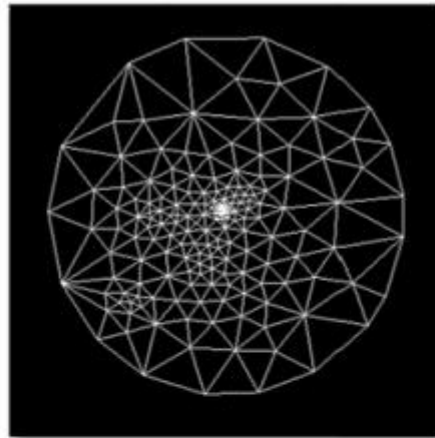




# Disk Parameterization

39 / 77

- ✓ Methods compared. Middle: uniform weights, right: harmonic weights.





# Disk Parameterization

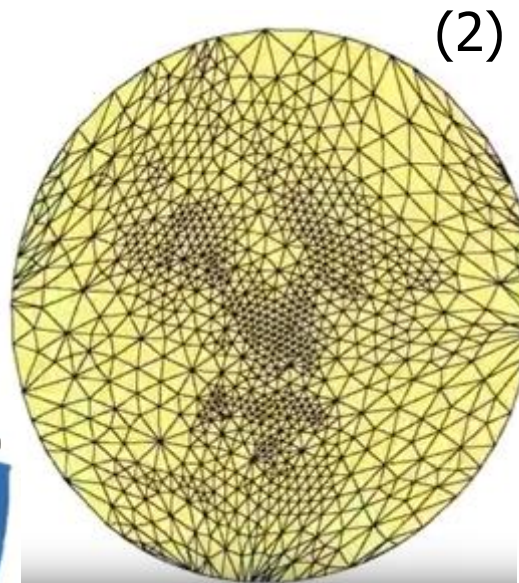
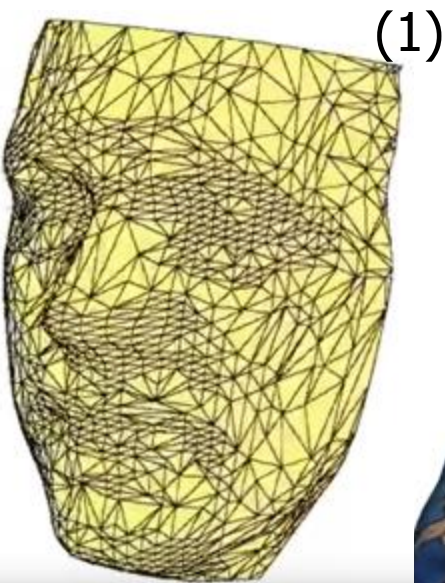
40 / 77

- ✓ Methods compared. Middle: uniform weights, right: harmonic weights.
- ✓ If you parameterize an input triangle mesh on a plane with a single convex boundary by prescribing the same boundary to itself, and solve a Laplacian with uniform edge weights for the inner vertices, do you reproduce the same mesh (input = output)?
  - ✓ No; uniform weights put every inner vertex in the perfect average of their neighbors.
  - ✓ Cotangent weights would achieve this reproduction (input=output).

# Disk Parameterization

41 / 77

- ✓ Method 1 is used for texture mapping application.



# Disk Parameterization

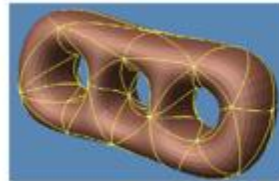
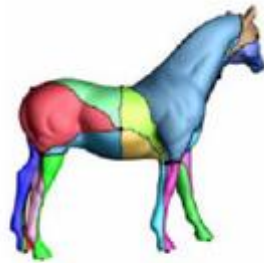
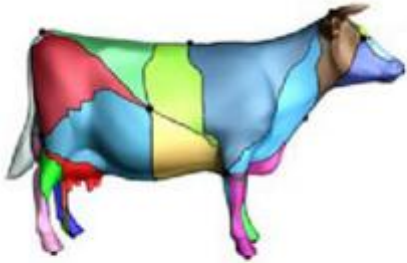
42 / 77

- ✓ Method 2 (harmonic weights) can be improved by making it bijective.
- ✓ To do that, simply make the 3D surface mesh Delaunay (simple greedy algorithm based on flipping non-Delaunay edges).
  - ✓ It has been proven that if the mesh satisfies the Delaunay criterion, even if it contains obtuse triangles, the parameterization obtained using the cotangent weights will always be bijective.
- ✓ In general, it is a good idea to improve the mesh quality, e.g., by bisecting obtuse angles or flipping edges, prior to applying any parameterization method.

# Fixed-Boundary Parameterization

43 / 77

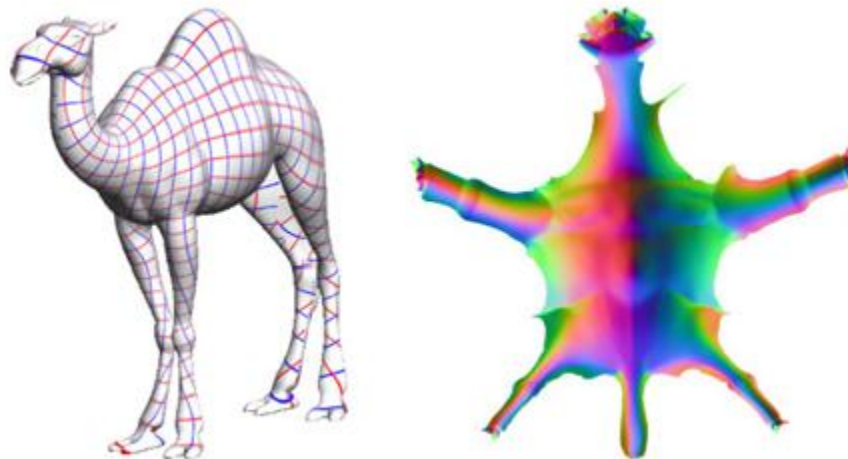
- ✓ Fixed-boundary techniques, e.g., disk parameterization, perform poorly when 3D surface meshes have
  - ✓ non-convex boundaries, or
  - ✓ boundaries that differ significantly from the specified boundary of the planar domain.
- ✓ One solution is to first segment the input mesh into triangular patches and then map them into 2D triangle boundaries.



# Free-Boundary Parameterization

44 / 77

- ✓ Fixed-boundary techniques, e.g., disk parameterization, perform poorly when 3D surface meshes have
  - ✓ non-convex boundaries, or
  - ✓ boundaries that differ significantly from the specified boundary of the planar domain.
- ✓ Another solution is to compute the boundary of the 2D domain as part of the solution, hence free-boundary.





# Free-Boundary Parameterization

45 / 77

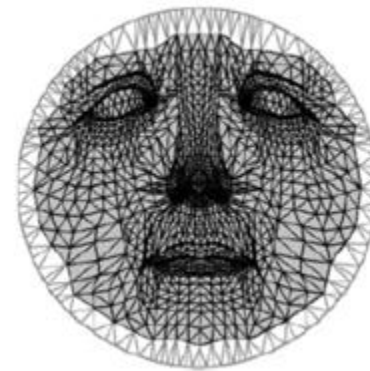
- ✓ Fixed-boundary techniques, e.g., disk parameterization, perform poorly when 3D surface meshes have
  - ✓ non-convex boundaries, or
  - ✓ boundaries that differ significantly from the specified boundary of the planar domain.
- ✓ Mesh Parameterization with a Virtual Boundary paper sets the boundary free as follows.



Virtual boundary (dashed) is added to the original mesh in 3D. Introduced 1+ layers of triangles around the original boundary (gray ones on the rightmost).



Mean-value weight parameterization (Method # 3) here has too much distortion.



Virtual boundary mapped to a fixed convex polygon and the rest mapped via Method # 3. Since the original boundary vertices are free to move, less distortion achieved here.

# Free-Boundary Parameterization

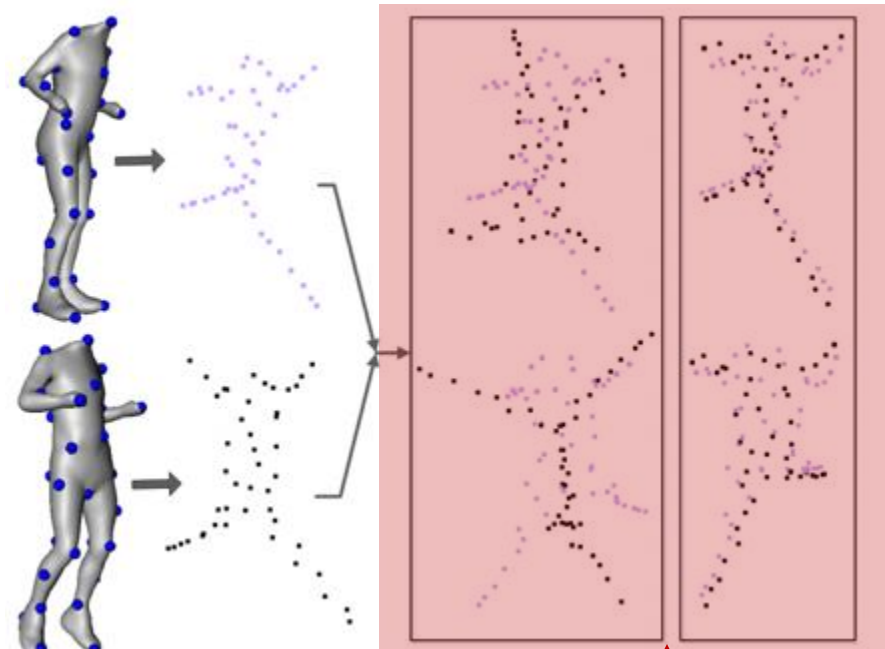
46 / 77

- ✓ All methods thus far were angle-preserving, i.e., minimize angular/conformal distortion during parameterization.
  - ✓ Method # 1 (uniform) was an exception as it minimizes nothing.
- ✓ Multi-dimensional scaling (MDS) based method minimizes length distortion, i.e., distance between two 3D points and their mapped 2D image points are kept as close as possible.
- ✓ This one is also free-boundary as it doesn't treat boundary differently.

# MDS-based Parameterization

47 / 77

- ✓ MDS: Embedding of the mesh into  $R^k$ ,  $k=2$  for planar parameterization, in such a way that pairwise dissimilarity values b/w points, e.g., geodesic distances, in  $R^3$  are replaced by the Euclidean distances in the embed space.
- ✓ Here are 2 different MDS transformations from  $R^3$  to  $R^3$ .



- ✓ Useful for non-rigid shape correspondence problem.
  - ✓ Robust 3D Shape Correspondence in the Spectral Domain
  - ✓ Minimum-Distortion Isometric Shape Correspondence Using EM Algorithm.

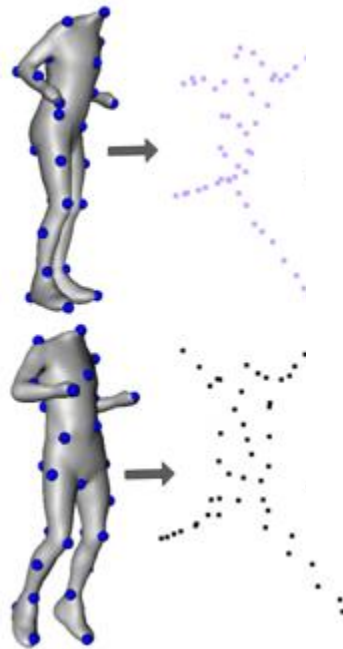
# MDS-based Parameterization

48 / 77

- ✓ pairwise dissimilarities are put in affinity matrix  $A$ .
- ✓ Classical method:  $k$  leading eigenvectors of  $A$  define MDS coords in  $R^k$ .
- ✓ Least-squares method: minimize  $E(\mathbf{v}) = \sum_{i < j} (\|\mathbf{v}_i - \mathbf{v}_j\| - g(i, j))^2$  using mass-spring system as described in [Detail-Preserving Mesh Unfolding for Nonrigid Shape Retrieval](#).
- ✓ Another stress function:

$$\sum_{i=1}^n \sum_{j=1}^n \left( 1 - \frac{\|\mathbf{e}(\mathbf{s}_i) - \mathbf{e}(\mathbf{s}_j)\|}{d_{\text{geo}}(\mathbf{s}_i, \mathbf{s}_j)} \right)^2$$

Weighted Averages on Surfaces.



# MDS-based Parameterization

49 / 77

- ✓ pairwise dissimilarities are put in affinity matrix  $A$ .
- ✓ Classical method is used in Texture Mapping Using Surface Flattening via Multidimensional Scaling to get parameterizations that are OK for non-complex surfaces such as:

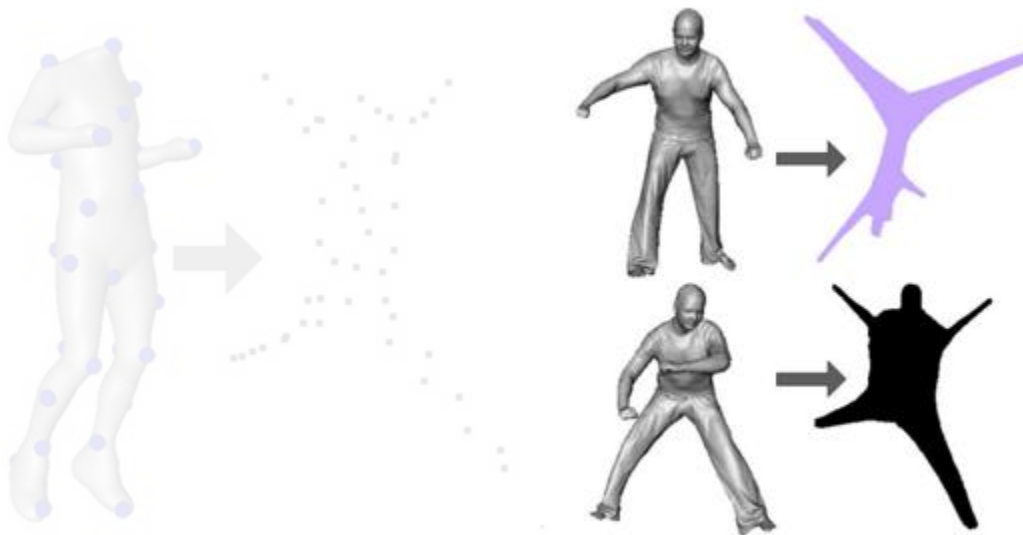




# MDS-based Parameterization

50 / 77

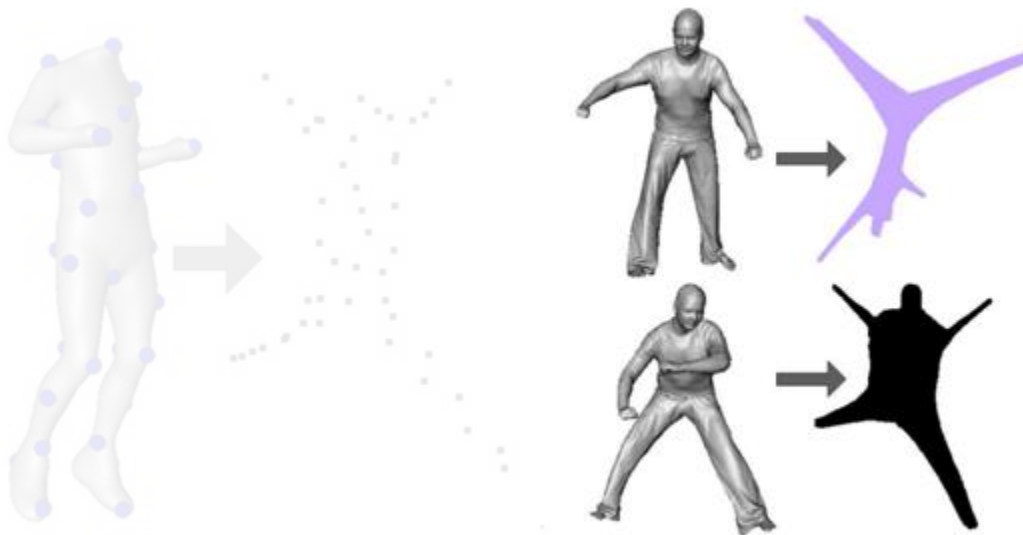
- ✓ pairwise dissimilarities are put in affinity matrix  $A$ .
- ✓ Classical or least-squares method slow for large input (all-pairs geodesics needed); accelerated via paper: Sparse Multidimensional Scaling Using Landmark Points.
- ✓ Given the embedded landmark points (geodesics b/w landmarks only), this paper (LMDS) computes embedding coordinates for the remaining data points based on their distances from the landmark points.



# MDS-based Parameterization

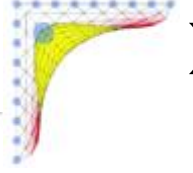
51 / 77

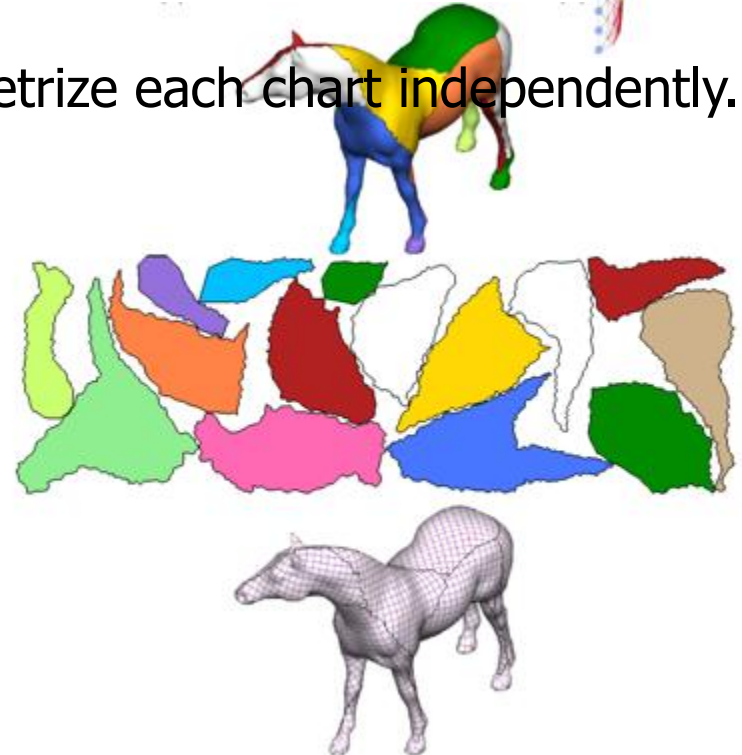
- ✓ pairwise dissimilarities are put in affinity matrix  $A$ .
- ✓ Alternative to LMDS based on Least-squares Meshes (see Deform. Slides).
- ✓ Idea: embed landmarks via classical or least-squares method, then use least-squares meshes to embed the remaining mesh vertices; resulting triangles'll be shaped similarly to the original mesh thanks to  $L$  matrix.



# MDS-based Parameterization

52 / 77

- ✓ pairwise dissimilarities are put in affinity matrix  $A$ .
- ✓ LMDS-like method is used in Iso-charts: Stretch-driven mesh parameterization using spectral analysis.
- ✓ If the parametrized planar mesh contains folds (local overlaps: ) ,  
they segment the surface and parametrize each chart independently.

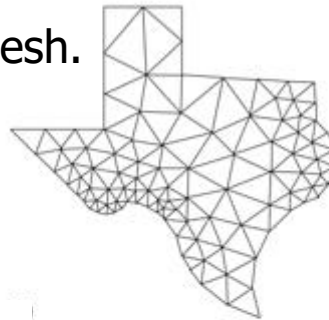


# Parameterization Refinement

53 / 77

- ✓ First, parameterize input mesh using an existing technique.
- ✓ Second, apply post-processing to improve the parameterization.
  - ✓ Untangling of 2D meshes in ALE simulations.

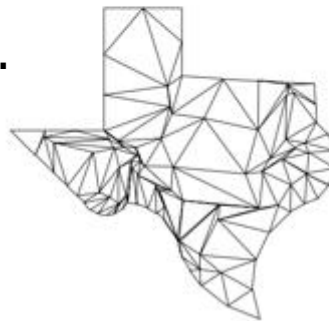
Original Texas mesh.



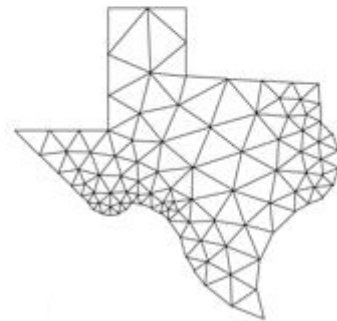
Tangled by random moves  
(not parameterization).



Untangled result.



Untangled result  
improved further.



# Parameterization Refinement

54 / 77

- ✓ First, parameterize input mesh using an existing technique.
- ✓ Second, apply post-processing to improve the parameterization.
  - ✓ Untangling of 2D meshes in ALE simulations.

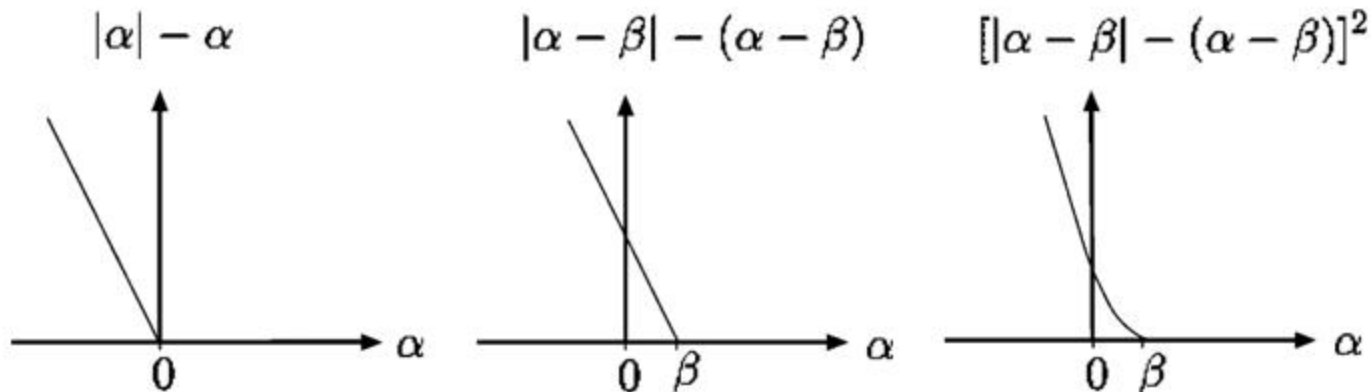
$$f(x) = \sum_i^n (|\alpha_i| - \alpha_i) \quad f(x) = \sum_i^n (|\alpha_i - \beta| - (\alpha_i - \beta)) \quad f(x) = \sum_i^n (|\alpha_i - \beta| - (\alpha_i - \beta))^2$$

$\alpha_i$ : area of the  $i^{\text{th}}$  element.

$\beta$  to prevent zero-area in the minimization of  $f(x)$ .

Quadratic version is differentiable/smooth for one element.

Untangle tetrahedral meshes by simply replacing areas with volumes.

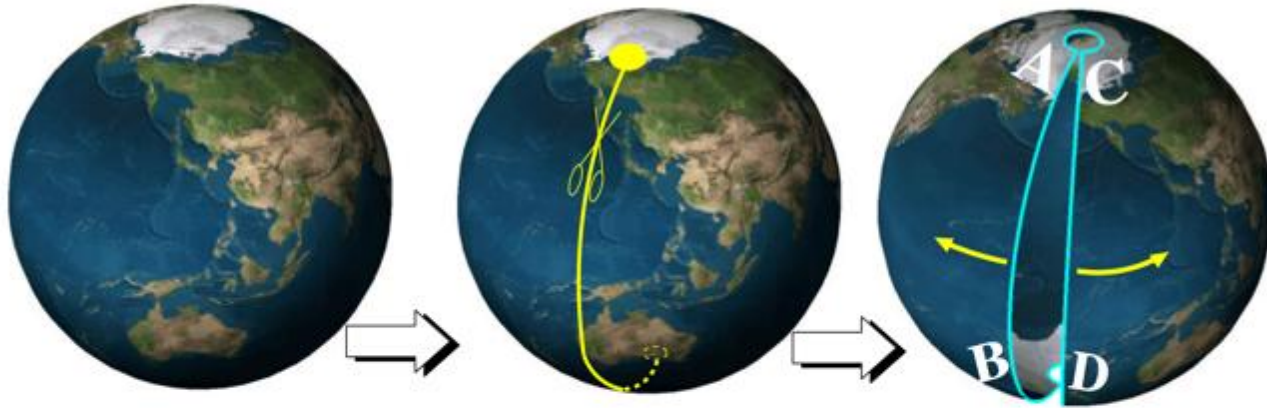




# Parameterization of Closed Meshes

55 / 77

- ✓ Closed meshes need to be cut open before mapped to a 2D plane.
- ✓ Artifacts are likely along the cuts.

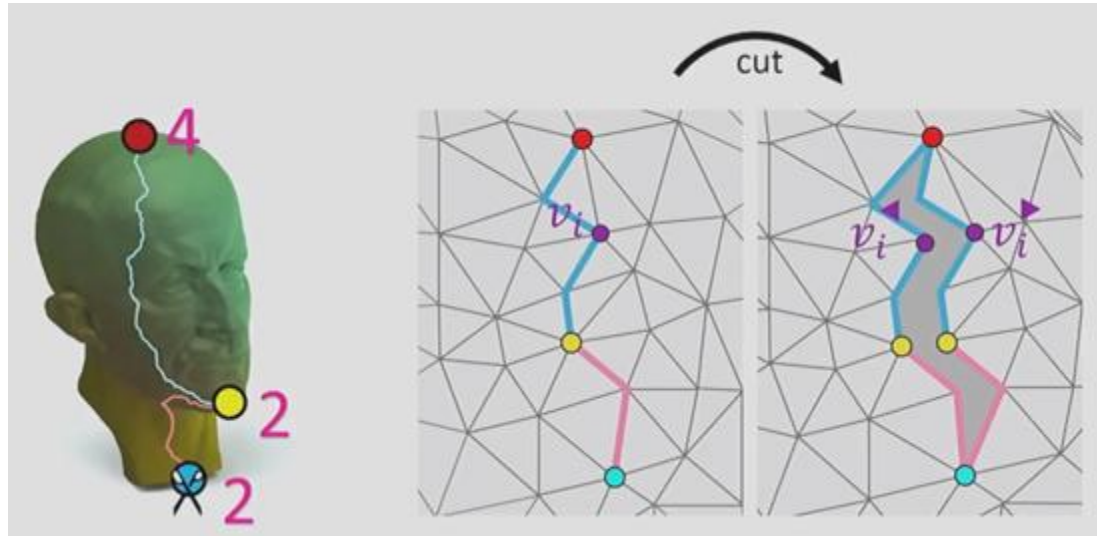


- ✓ Duplicate each vertex on the cut.
- ✓ Change the vertex index of the triangles on one side of the cut to reference those duplicate vertices.
- ✓ Now you've a disk topology surface that can be mapped to a 2D disk.

# Parameterization of Closed Meshes

56 / 77

- ✓ Closed meshes need to be cut open before mapped to a 2D plane.
- ✓ Artifacts are likely along the cuts.

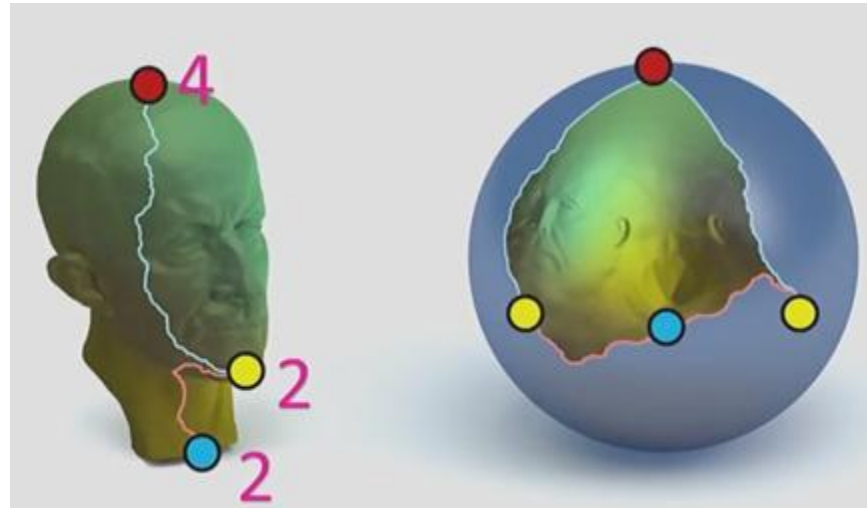
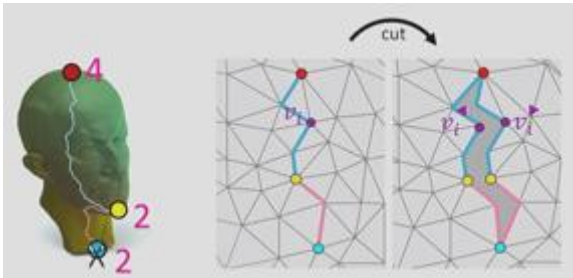


- ✓ Duplicate each vertex on the cut.
- ✓ Change the vertex index of the triangles on one side of the cut to reference those duplicate vertices.
- ✓ Now you've a disk topology surface that can be mapped to a 2D disk.

# Parameterization of Closed Meshes

57 / 77

- ✓ Closed meshes need to be cut open before mapped to a 2D plane.
- ✓ Artifacts are likely along the cuts.

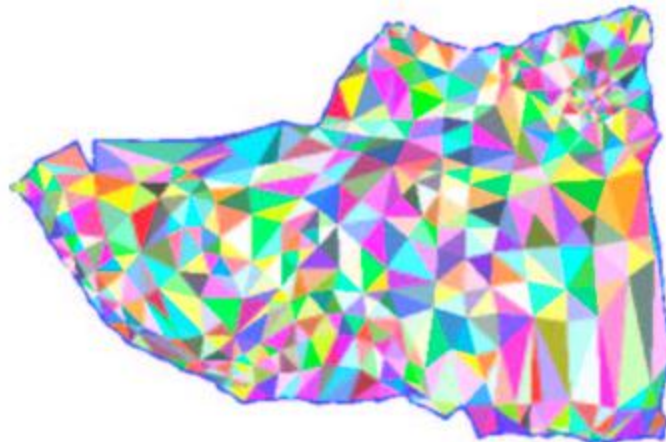


- ✓ Duplicate each vertex on the cut.
- ✓ Change the vertex index of the triangles on one side of the cut to reference those duplicate vertices.
- ✓ Now you've a disk topology surface that can be mapped to a sphere.

# Parameterization of Closed Meshes

58 / 77

- ✓ Closed meshes need to be cut opened before mapped to a 2D plane.
- ✓ Methods for cut/seam computation based on **Minimum Spanning Tree** computation (on high-curvature verts) to keep the cut **short** & **acyclic**.
  - ✓ E.g., Spanning Tree Seams for Reducing Parameterization Distortion of Triangulated Surfaces, 2002.



# Parameterization of Closed Meshes

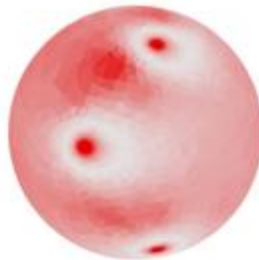
59 / 77

- ✓ Closed meshes need to be cut opened before mapped to a 2D plane.
- ✓ Methods for cut/seam computation based on **Minimum Spanning Tree** computation (on high-curvature verts) to keep the cut **short** & **acyclic**.
  - ✓ E.g., Sphere-Based Cut Construction for Planar Parameterizations, 2018. //high-curvature regions cluster together on sphere surface.

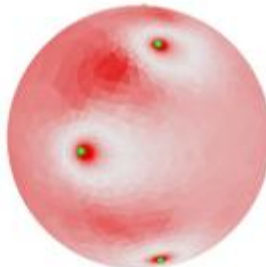
Input a closed  
genus-zero  
triangular mesh



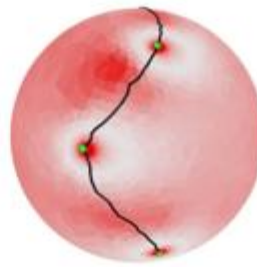
Step 1: parameterize  
to a sphere ACAP



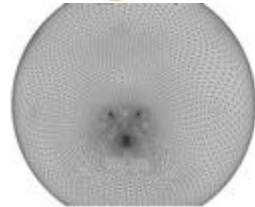
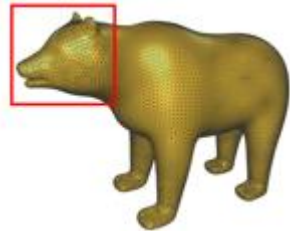
Step 2: find  
feature points by  
hierarchical clustering



Step 3: cut by  
a minimal  
spanning tree



Output an open mesh  
of disk topology

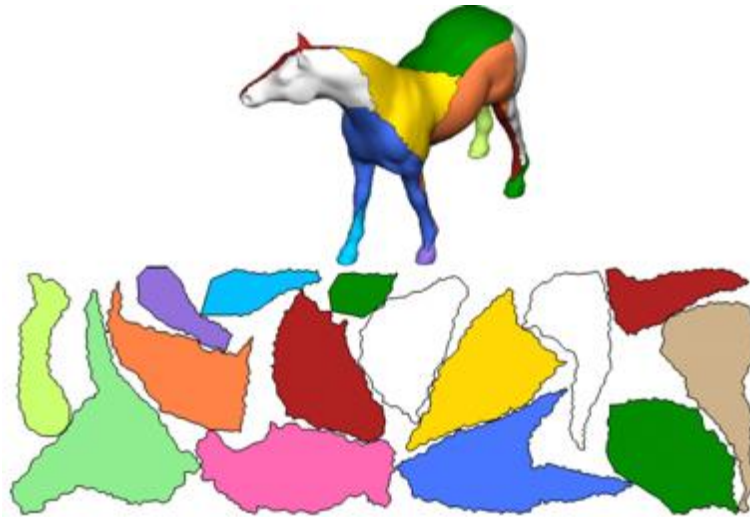




# Parameterization of Closed Meshes

60 / 77

- ✓ Closed meshes need to be cut opened before mapped to a 2D plane.
- ✓ Another kind of surface cutting algorithm segments the surface to many parts, each disk topology, that can be independently mapped charts.

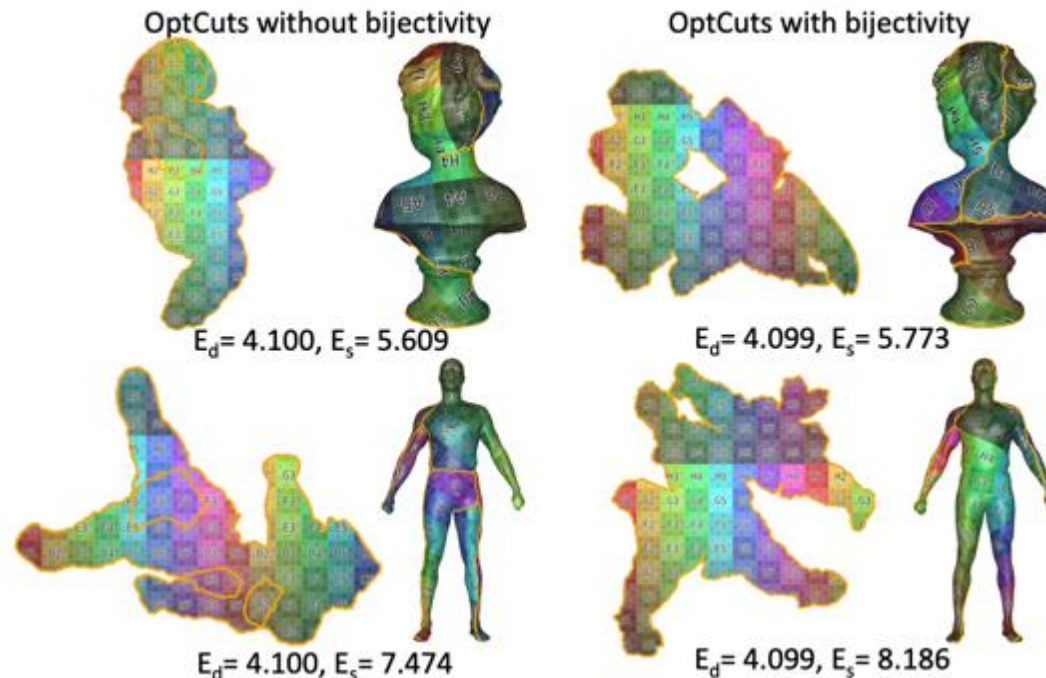


- ✓ More common because it is really hard to get a single connected cut that will map the surface to a single chart without large distortion.

# Parameterization of Closed Meshes

61 / 77

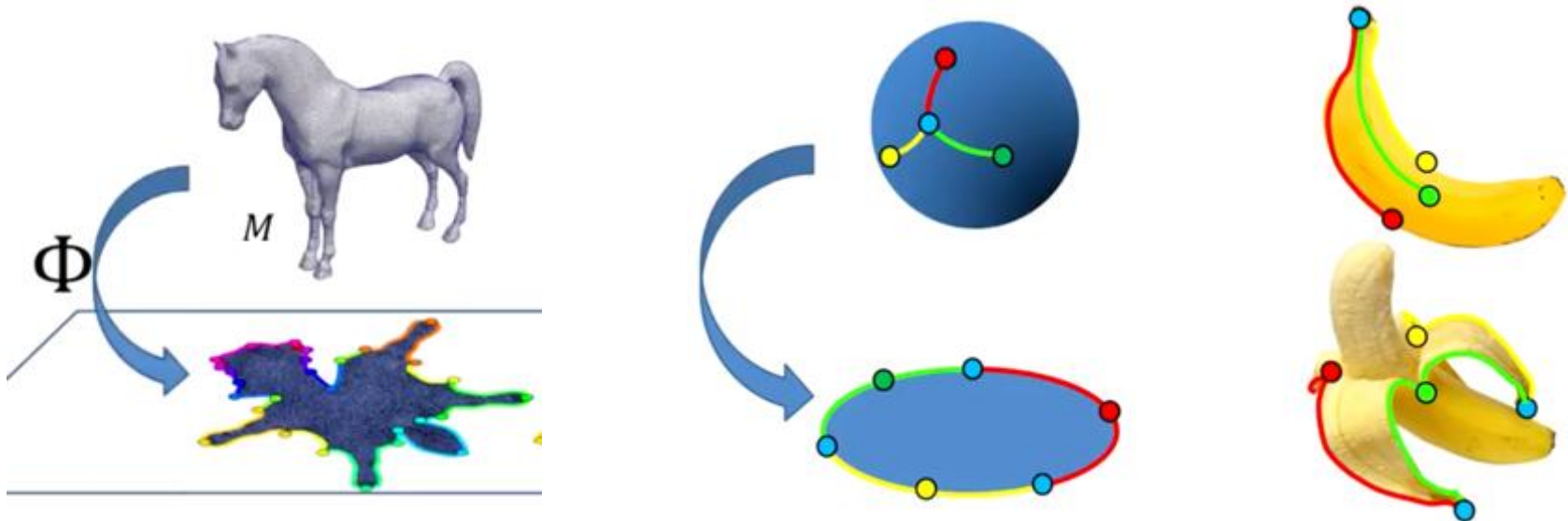
- ✓ Closed meshes need to be cut opened before mapped to a 2D plane.
- ✓ Better methods that optimize for cut/seam computation and parameterization distortion jointly.
  - ✓ OptCuts: Joint Optimization of Surface Cuts and Parameterization.
  - ✓ Autocuts: Simultaneous Distortion and Cut Optimization for UV Mapping.



# Parameterization of Closed Meshes

62 / 77

- ✓ Closed meshes need to be cut open before mapped to a 2D plane.
- ✓ See <https://youtu.be/Yi8i5iKVO7s> & [https://youtu.be/\\_W2ipCpCz8g](https://youtu.be/_W2ipCpCz8g) for cool demonstrations.



# Spherical Parameterization

63 / 77

- ✓ So far we dealt with planar parameterization:  $3D \rightarrow 2D$ .
  - ✓ Very useful for texture mapping.
- ✓ More natural parameterization domain is 3D for closed manifold genus-0 meshes, e.g., cow below.
- ✓ Sphere is the best choice since all such meshes are topologically equivalent (homeomorphic) to a sphere.



- ✓ A genus-1 mesh is homeomorphic to a torus.



# Spherical Parameterization

64 / 77

- ✓ So far we dealt with planar parameterization:  $3D \rightarrow 2D$ .
  - ✓ Very useful for texture mapping.
- ✓ Spherical parameterization allows for seamless and continuous parameterization of genus-0 models.
  - ✓ Cut/seam-based methods are discontinuous along cuts.
- ✓ Sphere is the best choice since all such meshes are topologically equivalent (homeomorphic) to a sphere, no cutting gluing needed.

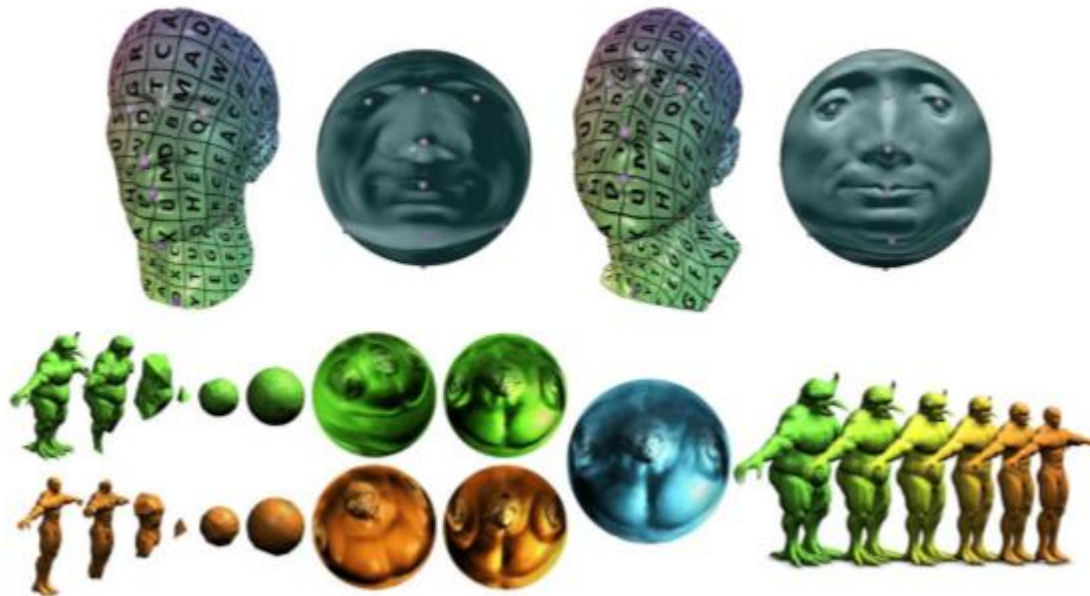




# Spherical Parameterization

65 / 77

- ✓ Main motivation: establish correspondences on spheres (easier) and then use them to obtain vertex paths for a morphing application.

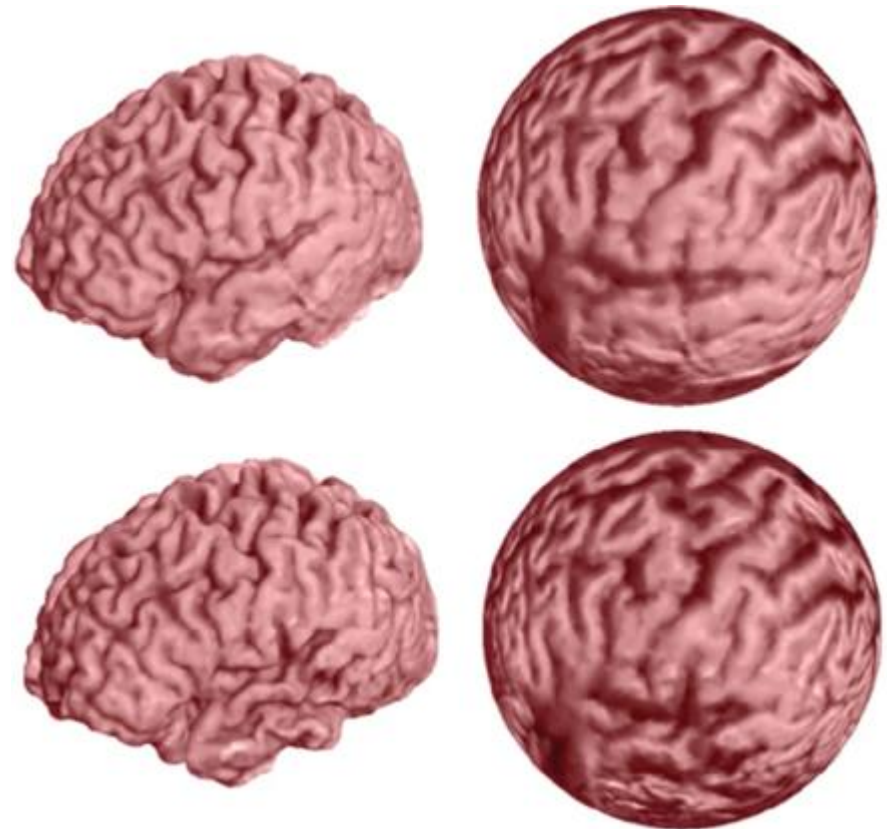
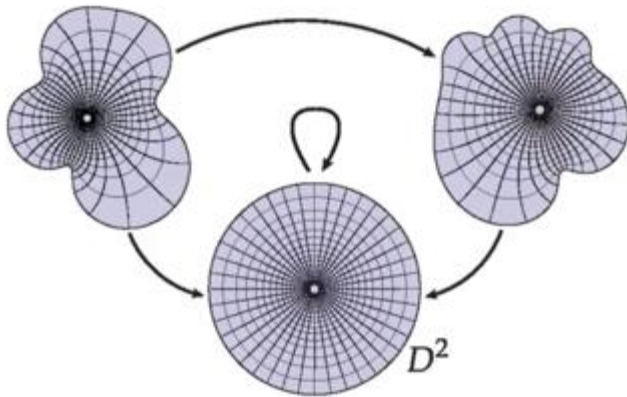


- ✓ Once meshes are parameterized to spheres, use a few landmark correspondences to align the spheres properly. Then the full correspondence can be obtained by overlaying (closest points).

# Spherical Parameterization

66 / 77

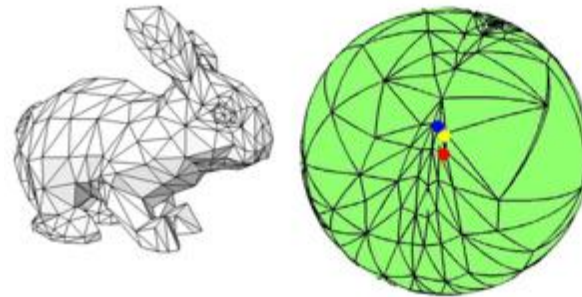
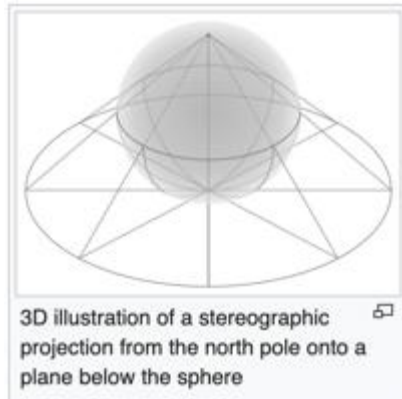
- ✓ Similar motivation: see/compare things more clearly, e.g., going to a canonical domain, such as a sphere, makes it easier to see features.



# Spherical Parameterization Method # 1

67 / 77

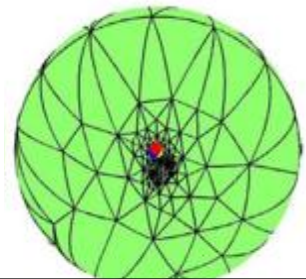
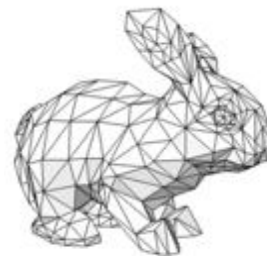
- ✓ Reduce the problem to planar case.
- ✓ Cut out 1 triangle to serve as a (convex) boundary.
- ✓ Parameterize the resulting open mesh to that unit triangle.
  - ✓ Any planar parameterization will do, e.g., Slide 29.
- ✓ Use the inverse stereographic projection to map the plane to the sphere.



# Spherical Parameterization Method # 2

68 / 77

- ✓ Reduce the problem to planar case.
- ✓ Cut the mesh into two pieces, each topologically equivalent to a disk.
- ✓ Parameterize the resulting open meshes to disks, e.g., Slide 29.
- ✓ Map each disk to a hemisphere by adding a nice z component to verts.
- ✓ The common boundary guarantees that the two hemispheres fit together at the equator.
- ✓ Since this boundary will presumably contain more than just three vertices, each of the two disk parameterizations will be less distorted than the one obtained by using a single triangle as the boundary (Method # 1), so the spherical result will also be less distorted.
- ✓ Disadvantage: Result strongly depends on the specific cut used to obtain the two disks.



# Spherical Parameterization Method # 3

69 / 77

- ✓ No going back and forth to the plane, better.
- ✓ For convex and star-shaped objects, there exists at least one interior point,  $p$ , from which all the vertices of the boundary are visible.
- ✓ Translate the object such that  $p$  coincides w/ the origin.
- ✓ Send a ray from  $p$  to each vertex.
- ✓ This ray moves the hit vertex in or out until it has unit length.



- ✓  $p$  can be algorithmically selected by first computing the intersection of the interior half spaces of all the planes of the faces of the model. The resulting volume is called the kernel of the 3D polyhedral model. If the original polyhedron is star-shaped, its kernel is a non-empty convex polyhedron. Averaging the vertices of the kernel yields  $p$ . For convex polyhedron, any interior point work as  $p$ .



# Spherical Parameterization Method # 4

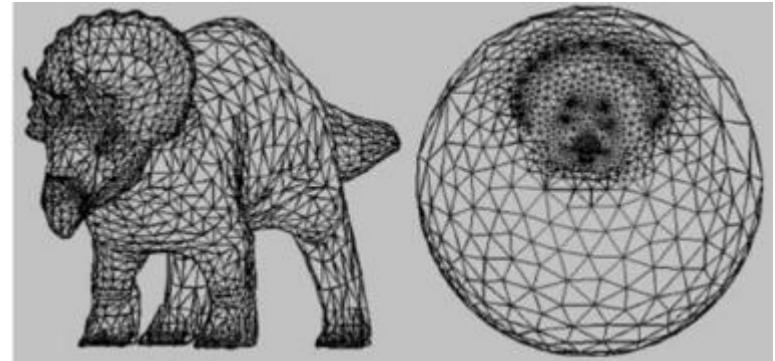
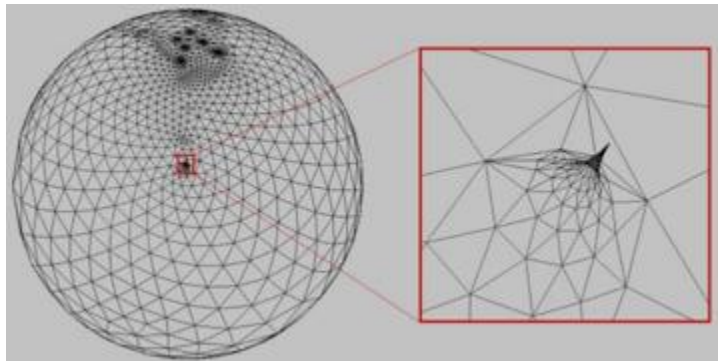
70 / 77

- ✓ No going back and forth to the plane, better.
- ✓ A direct parameterization for *any* polyhedron (not just convex ones and stars) is difficult to achieve.
- ✓ Less distorted and non-overlapping spherical triangles are desired.
  - ✓ Especially, non-overlapping 'cos it gives one-to-one bijective maps for upcoming applications, e.g., morphing.
- ✓ Fundamentals of Spherical Parameterization for 3D Meshes paper provides a natural non-linear extension of the linear parameterization method (Slide 29), good for a term project!

# Spherical Parameterization Method # 5

71 / 77

- ✓ No going back and forth to the plane, better.
- ✓ A direct parameterization for *any* polyhedron (not just convexes and stars) is difficult to achieve.
- ✓ Here is a heuristic.
- ✓ Send rays from the object centroid to each vertex.
- ✓ Pull them to unit sphere boundary just like Method # 3.
- ✓ If object is not star-shaped, there will be overlaps ☹. To alleviate the problem, move each vertex to the center of its 1-ring neighbors and project back to sphere, i.e., make length to the centroid 1. Repeat.



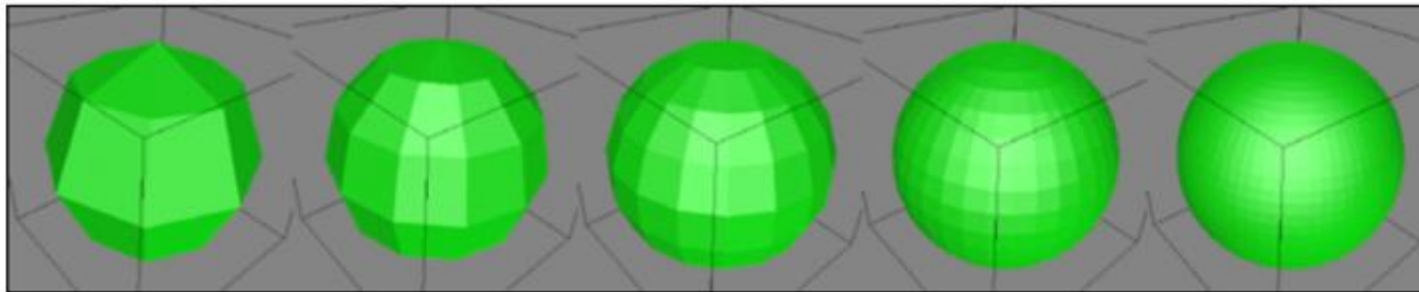
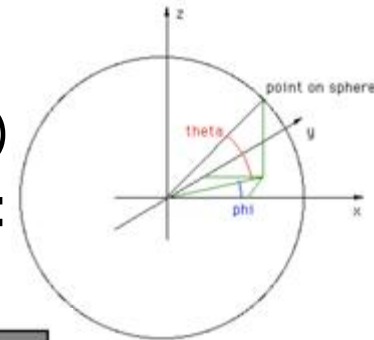
# Sphere Generation Method # 1

72 / 77

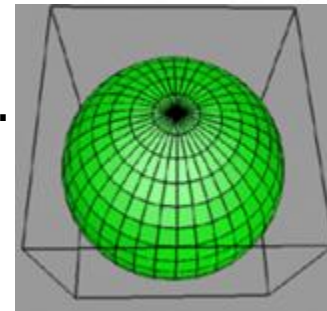
- ✓ Not about parameterization, but still related. How to generate sphere meshes from scratch?
- ✓ Go from polar to Cartesian coordinates.

$$x = \cos(\theta)\cos(\phi) \quad y = \cos(\theta)\sin(\phi) \quad z = \sin(\theta)$$

Step theta and phi in small angles to get a quad on sphere:  
(theta, phi), (theta+dt, phi), (theta+dt, phi+dp), (theta, phi+dp).



Progress from 45 degrees via 5 degree increments.  
Drawback: non-uniformity, smaller faces at the poles.

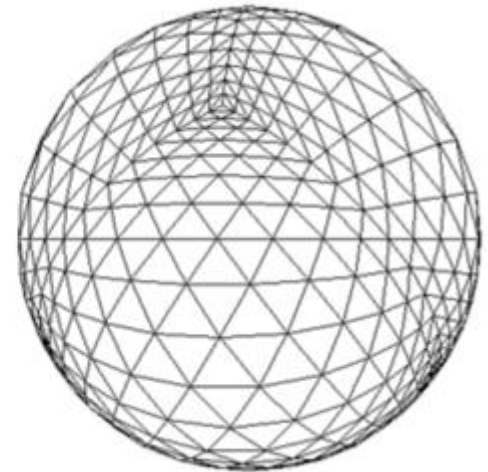
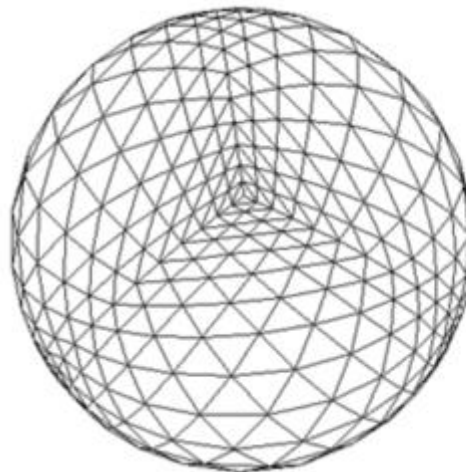
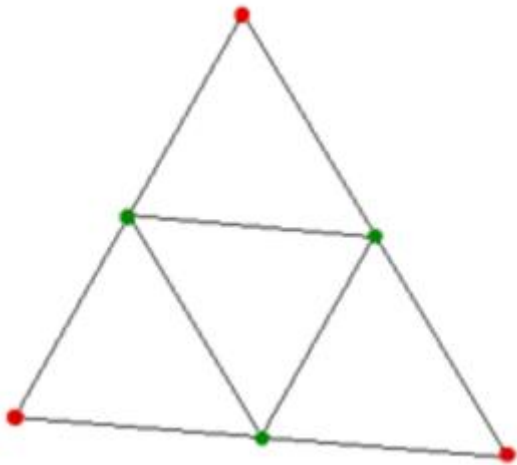


# Sphere Generation Method # 2

73 / 77

- ✓ Not about parameterization, but still related. How to generate sphere meshes from scratch?
- ✓ Refine surfaces

Start with a tetrahedron. Split each face to 4 by bisecting edges. After splitting all faces, move each vertex to the boundary of the sphere by simply normalizing the vector and scaling by the desired radius.



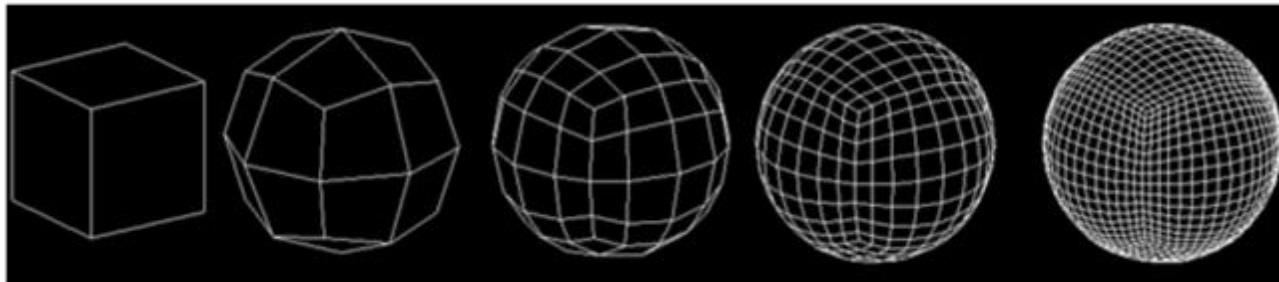
# Sphere Generation Method # 2

74 / 77

- ✓ Not about parameterization, but still related. How to generate sphere meshes from scratch?

- ✓ Refine surfaces

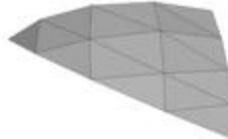
Start with a different polyhedron and repeat.



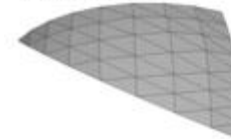
Iteration 1: 4 triangles



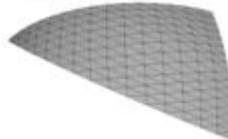
Iteration 2: 16 triangles



Iteration 3: 64 triangles



Iteration 4: 256 triangles



Iteration 5: 1024 triangles



Iteration 6: 4096 triangles

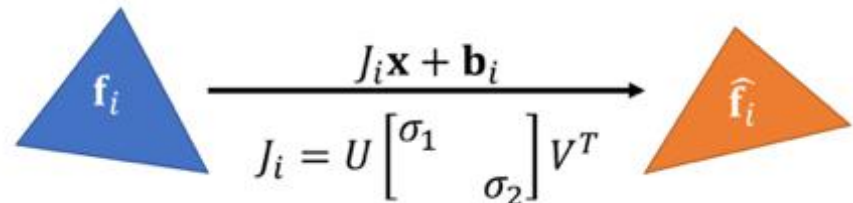




# Parameterization Distortion

75 / 77

- ✓ Should be minimized for a good parameterization.
  - ✓ Given a cut/seam, do disk parameterization, measure distortion, repeat w/ another\* cut. Select result w/ min distortion (or Slide61).
- ✓ Discussed length distortion (MDS, Slide 46) and seen angular/conformal distortion (Disk Param. Method # 2 & 3, Slides 30-31). Let's formalize:



Conformal distortion (angle preserving) [Hormann et al., 2000]

$$d_i^{\text{conf}} = \frac{1}{2} \left( \frac{\sigma_1}{\sigma_2} + \frac{\sigma_2}{\sigma_1} \right) = \frac{1}{2} \frac{\|J_i\|^2}{\det J_i}$$

Areal distortion (area preserving) [Fu et al., 2015]

$$d_i^{\text{area}} = \frac{1}{2} (\det J_i + (\det J_i)^{-1})$$

Isometric distortion (isometry preserving) [Fu et al., 2015]

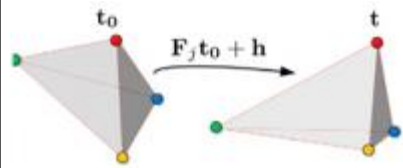
$$d_i^{\text{iso}} = \alpha d_i^{\text{conf}} + (1 - \alpha) d_i^{\text{area}}$$

\* Add most distorted vertex to new cut: Greedy Cut Construction for Parameterizations, 2020.

# Parameterization Distortion

76 / 77

- ✓ Should be minimized for a good parameterization.
- ✓ Discussed length distortion (MDS, Slide 46) and seen angular/conformal distortion (Disk Param. Method # 2 & 3, Slides 30-31). Formalize:
- ✓  $J_i$  is the deformation gradient of the  $i^{\text{th}}$  triangle. I call it  $F$  below & find it for a tetrahedron. Triangle case similar; shown as 2D case at bottom.



Deformation gradient  $F$ : Transformation (e.g.  $3 \times 3$  matrix in 3D) that maps rest-pose (or any other initial pose) edges to their deformed pose

$$x = F \cdot \tilde{x} + t$$

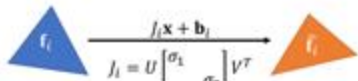
$$\begin{aligned} x_1 &= F \tilde{x}_1 + t \\ x_2 &= F \tilde{x}_2 + t \\ x_3 &= F \tilde{x}_3 + t \\ x_4 &= F \tilde{x}_4 + t \end{aligned} \Rightarrow \begin{array}{l} \text{Subtract} \\ x_4 \text{ from} \\ \text{all to} \\ \text{cancel } t \end{array}$$

$$\Rightarrow \begin{aligned} x_1 - x_4 &= F(\tilde{x}_1 - \tilde{x}_4) \\ x_2 - x_4 &= F(\tilde{x}_2 - \tilde{x}_4) \\ x_3 - x_4 &= F(\tilde{x}_3 - \tilde{x}_4) \end{aligned} \Rightarrow \begin{pmatrix} x_1 - x_4 & x_2 - x_4 & x_3 - x_4 \end{pmatrix} = F \cdot \begin{pmatrix} \tilde{x}_1 - \tilde{x}_4 & \tilde{x}_2 - \tilde{x}_4 & \tilde{x}_3 - \tilde{x}_4 \end{pmatrix}$$

$$\Rightarrow F = D_s \cdot D_m^{-1} \quad \Rightarrow \text{precompute}$$

2D case  $\Rightarrow \begin{pmatrix} x_1 - x_3 & x_2 - x_3 \end{pmatrix} = F \cdot \begin{pmatrix} \tilde{x}_1 - \tilde{x}_3 & \tilde{x}_2 - \tilde{x}_3 \end{pmatrix}$

$$\Rightarrow F = D_s \cdot D_m^{-1}$$



# Potential Project Topics

77 / 77

- ✓ Implement an MST-based cut/seam computation paper (Slides 58-59).
- ✓ Implement a non-linear method from: Mesh Parameterization Methods and Their Applications.
- ✓ Implement an untangling method: Untangling of 2D meshes in ALE simulations.
- ✓ Implement kernel computation for a star-shaped polyhedron (Slide 69).