Geometric Modeling

Bing-Yu Chen
National Taiwan University
The University of Tokyo

Surface Parameterization

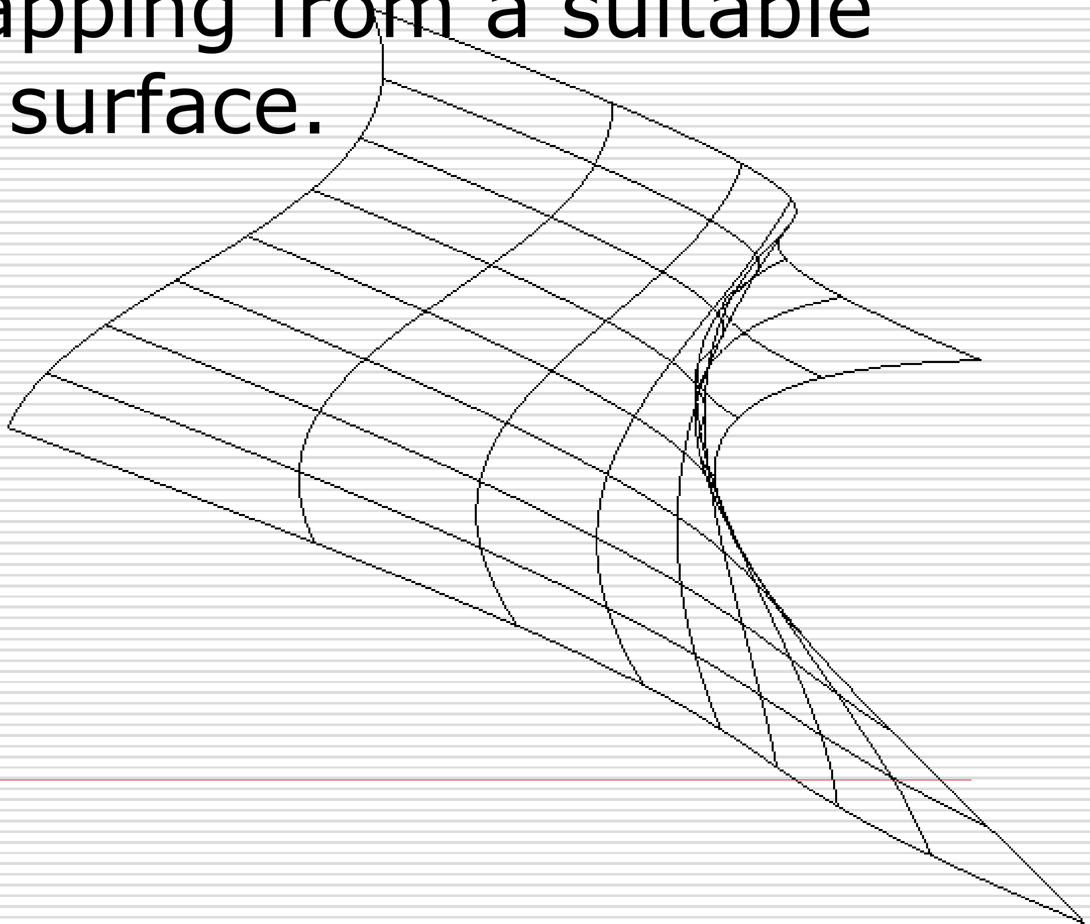
- Introduction
- Applications
- What is Parameterization?
- Parameterization Methods

Problem

- 1-1 mapping from domain to surface
- Original application:
  - Texture mapping
    - Images have a natural parameterization
  - Goal: map onto surfaces
- Geometry processing
  - Approximation
  - Remeshing
  - Data fitting
- Input: Piecewise (PL) triangular meshes

Introduction

- A parameterization of a surface is a one-to-one mapping from a suitable domain to the surface.



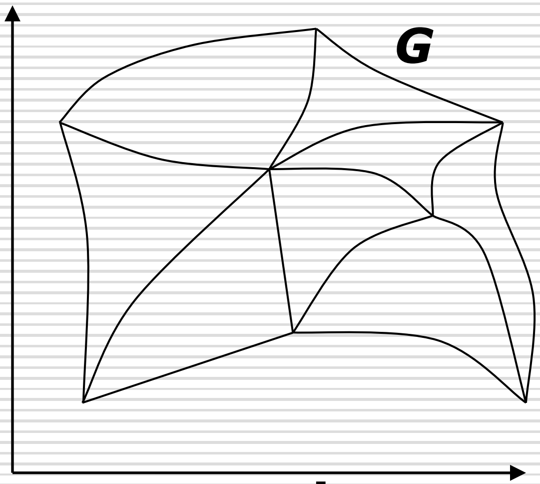
Introduction

- In general, the parameter domain itself will be a surface.
- Constructing a parameterization means mapping a surface into another.
- Usually the surfaces are either represented by or approximated by triangular meshes and the mappings are ***piecewise linear***.

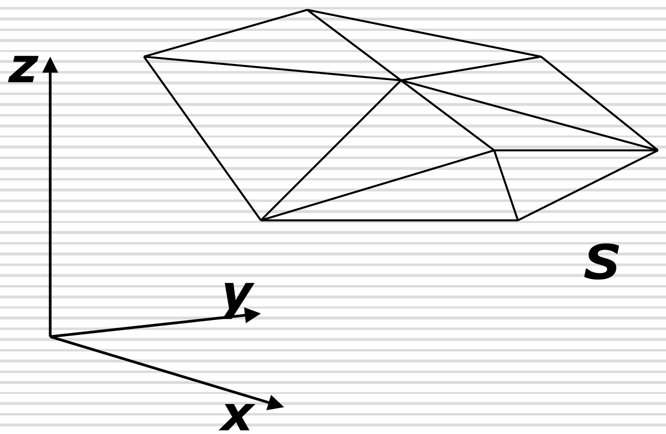
Applications

- Scattered data fitting.
- Reparameterization of Spline surfaces
- Texture-mapping
- Mesh compression
- Surface approximations & remeshing

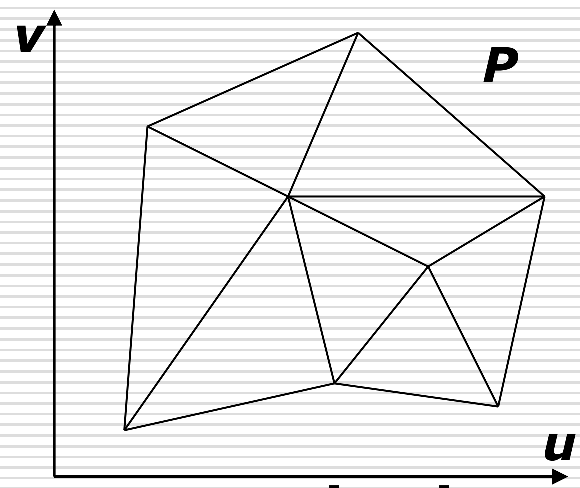
What is Parameterization?



graph



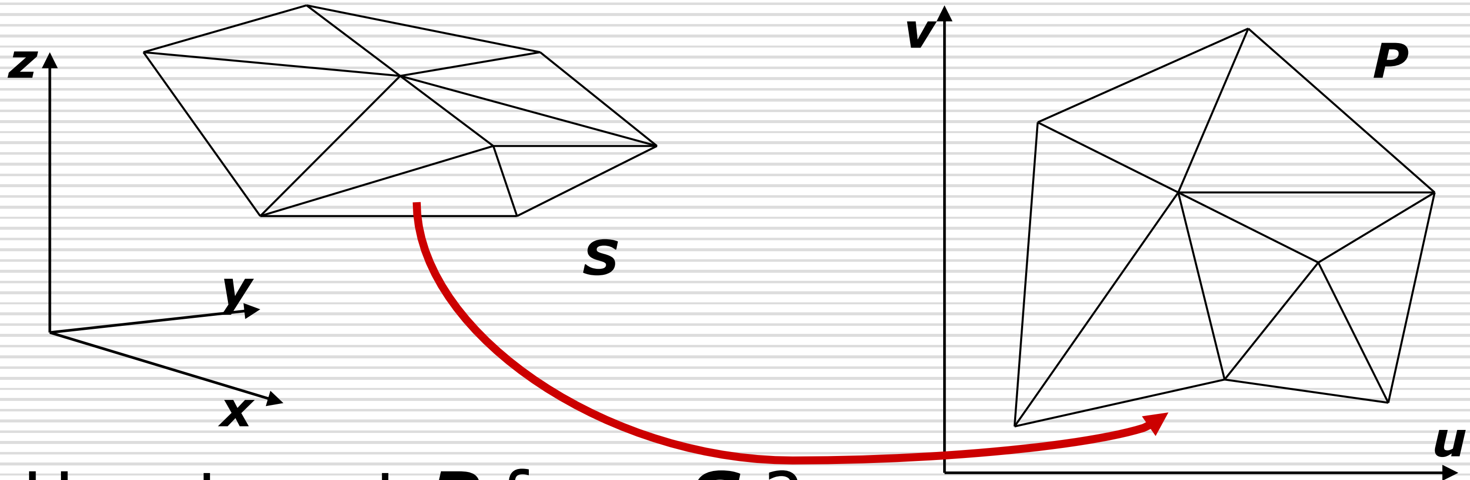
surface triangulation (mesh)



paramerization

- G(V,E,F)
- S(G,X), $X=\{x_i \in R^3\}$
- P(G,U), $U=\{u_i \in R^2\}$

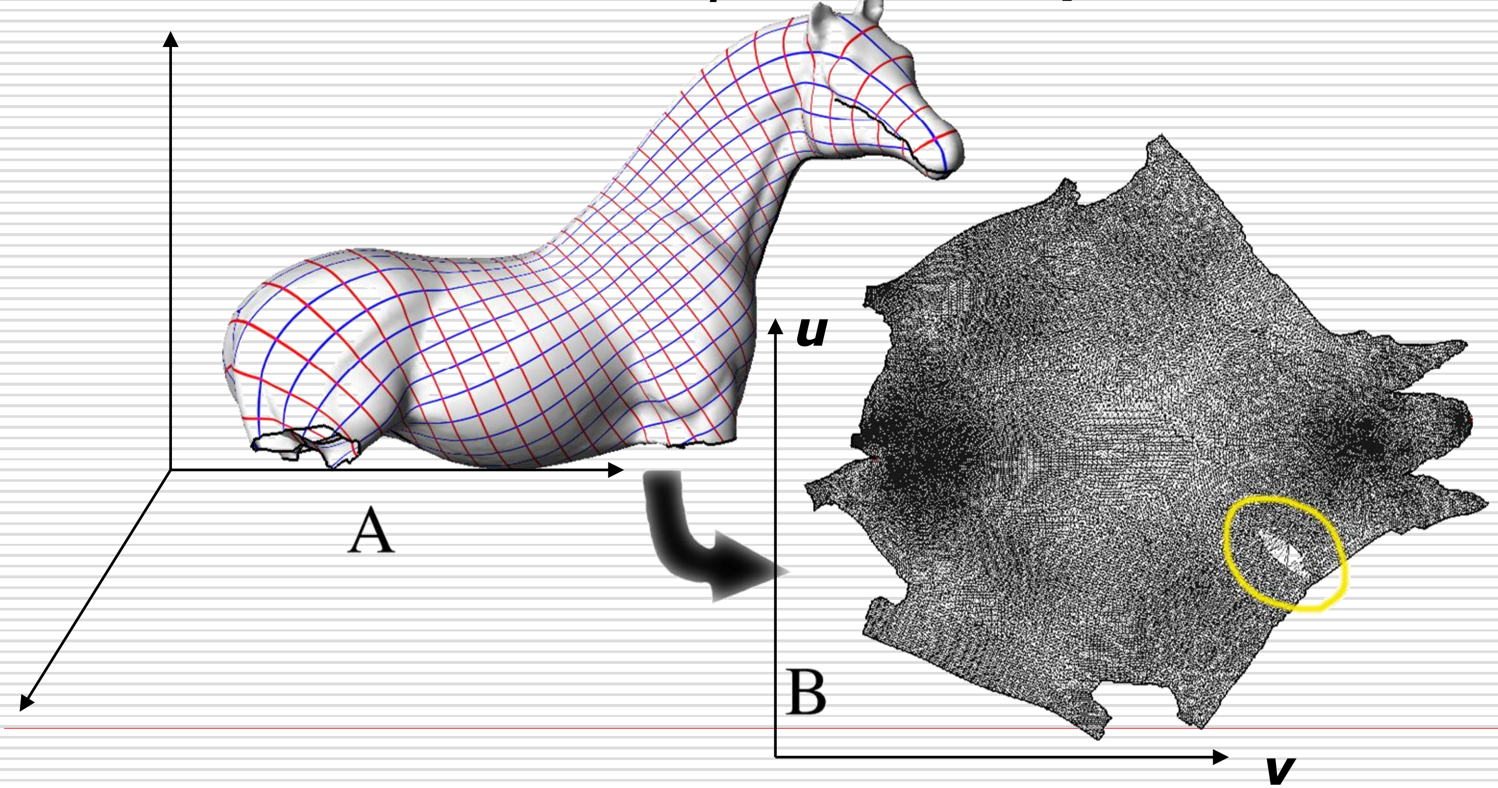
What is Parameterization?



- How to get ***P*** from ***S*** ?
  - for each vertex of ***S***, find its (u,v)
  - from (u,v) of ***P***, map image to ***S***
- A parameterization of a surface is a mapping ρ: (x,y,z)->(u,v) from 3D space to 2D space

Problem Definition

- Given a surface (mash) S in R^3 and a domain D find ρ: $D \leftrightarrow S$ (one-to-one)

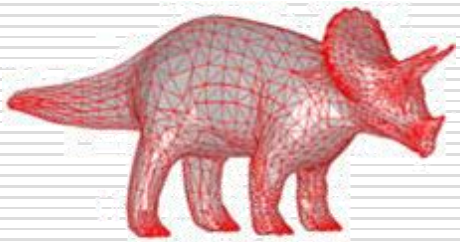


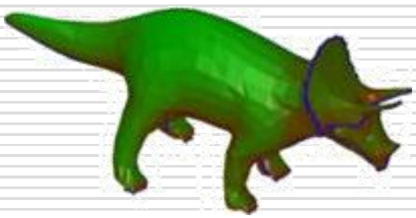
Recall: Applications

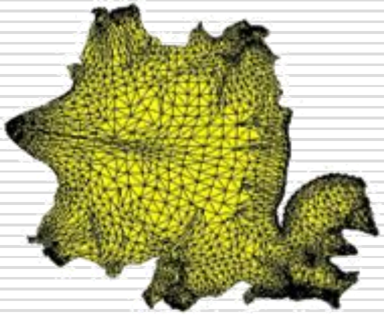
- Texture-mapping
  - ρ: (x,y,z) -> (u,v) from 3D to 2D
- Remeshing
  - ρ^{-1}: (u,v) -> (x,y,z) from 2D to 3D

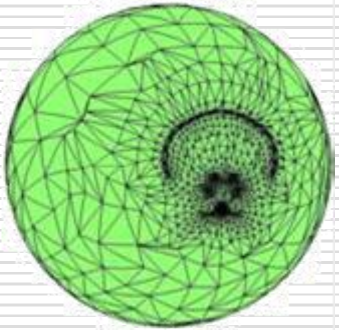
Typical Domains

sub-domain of R^2
- genus-0 + **boundary**



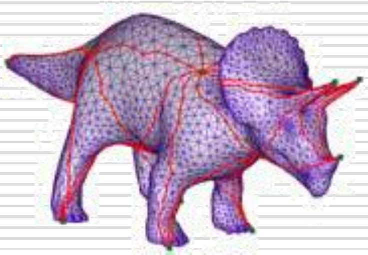


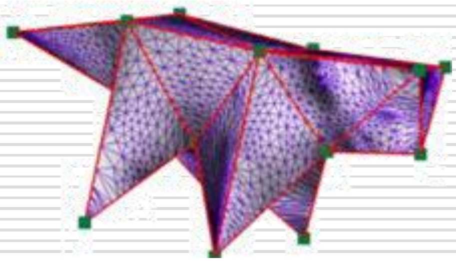




sphere
- closed genus-0

base mesh
- all (closed) models

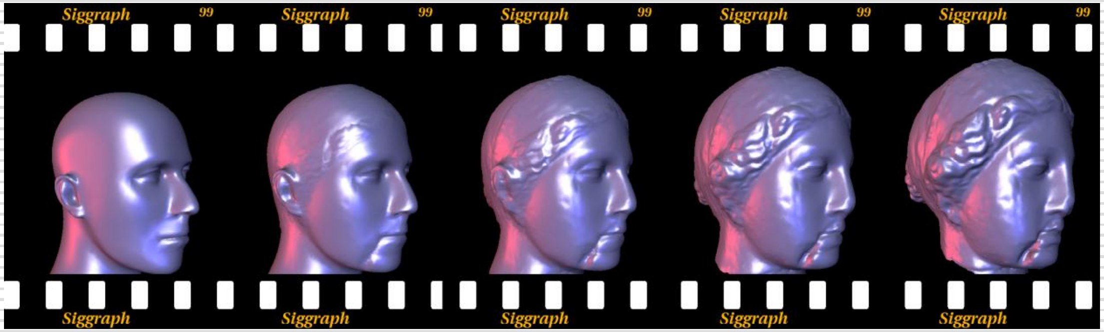




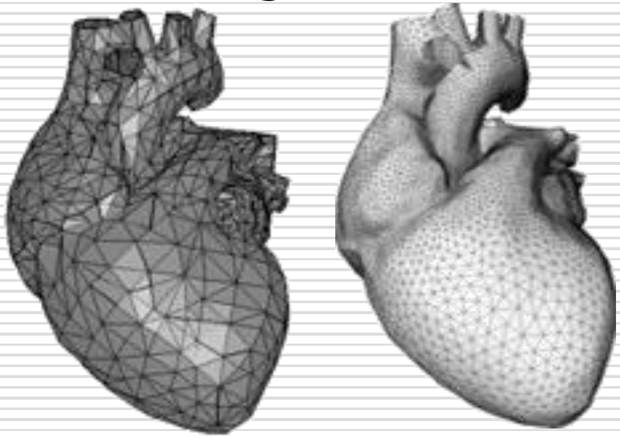
Applications

texture mapping

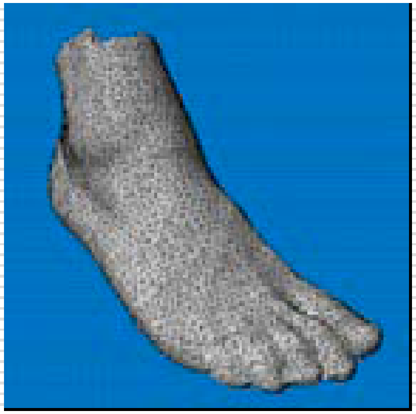
morphing



remeshing

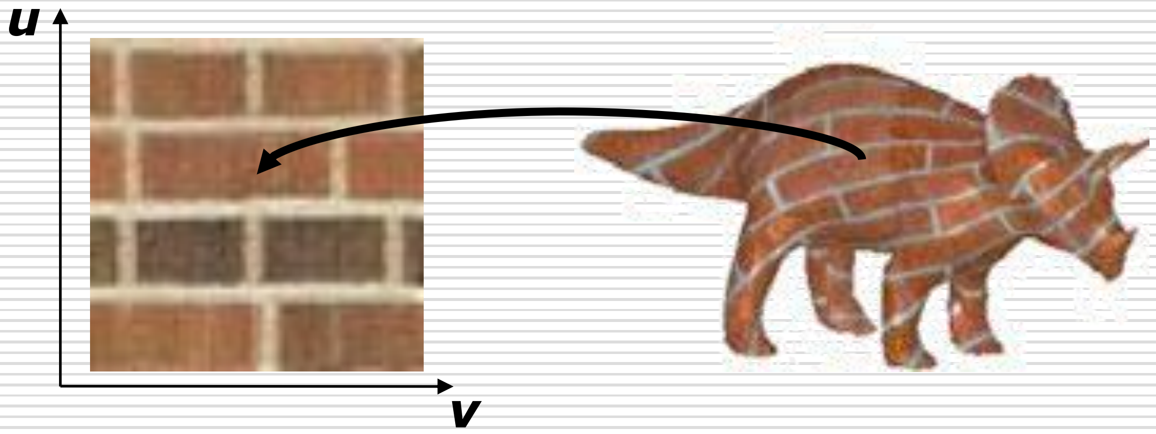


reconstruction



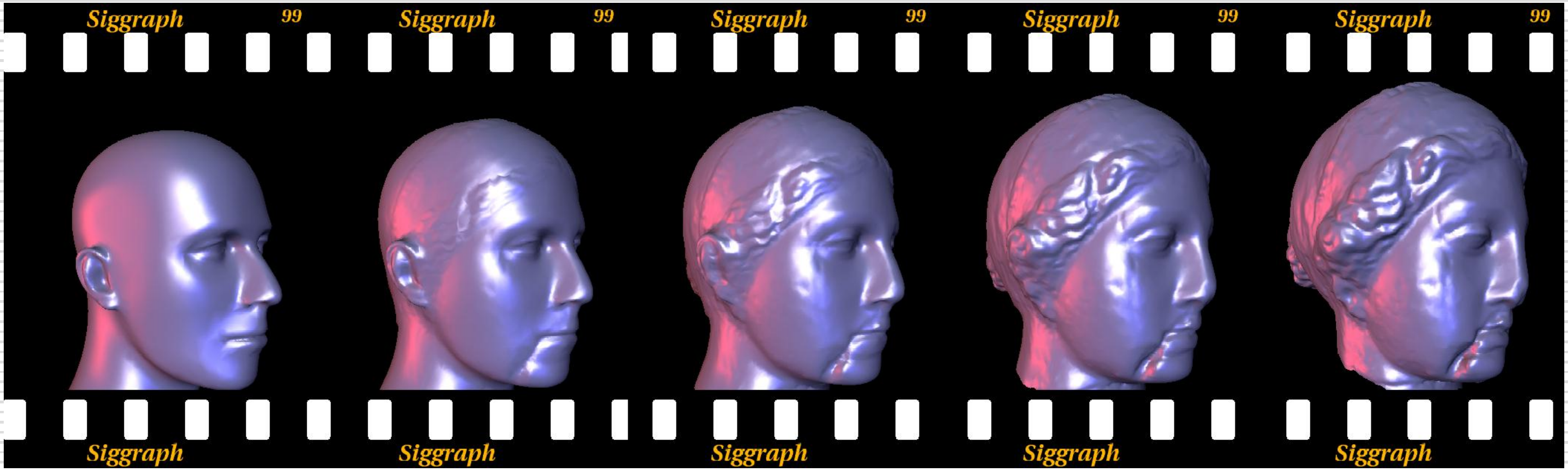
Texture Mapping

- Real life objects not uniform in terms of color
- **Texturing** - define color for each point on object surface
- Map 2D texture to model surface:
  - Have texture pattern defined over (u,v) domain (Image)
  - Assign (u,v) coordinates to each point on object surface

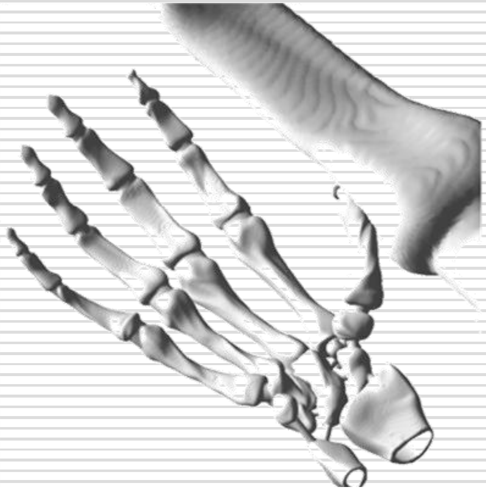


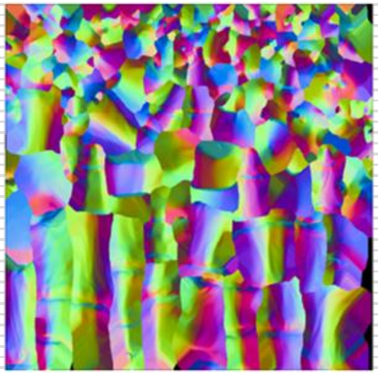
Morphing

- Morphing requires one-to-one correspondence between the surfaces of the two models

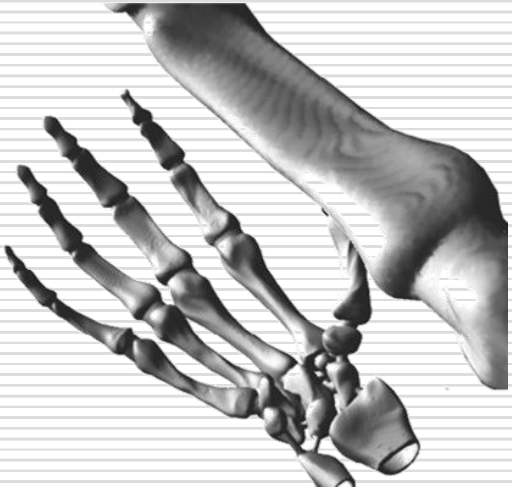


Normal / Bump Mapping

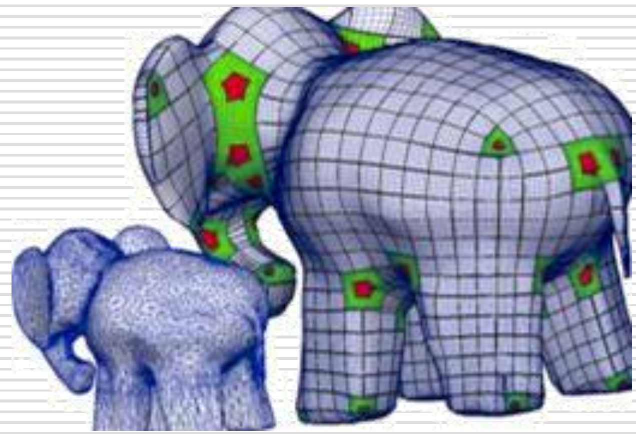

650K faces

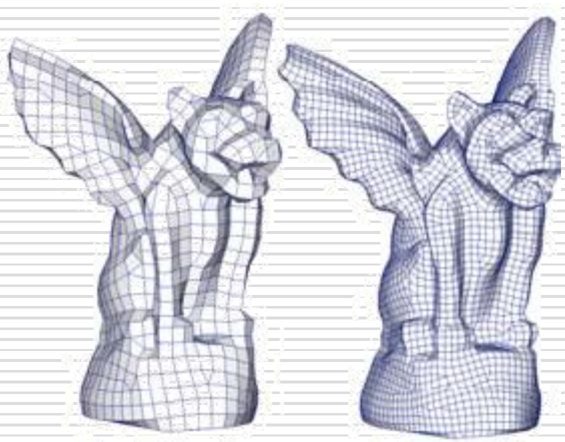

normal map

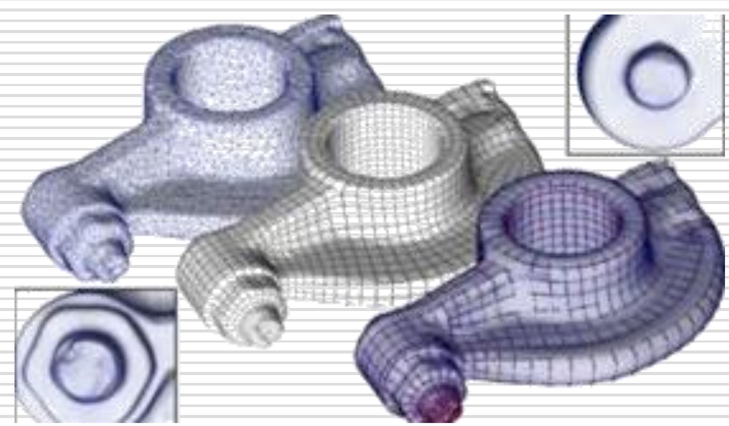

29K faces


normal mapped simplified mesh

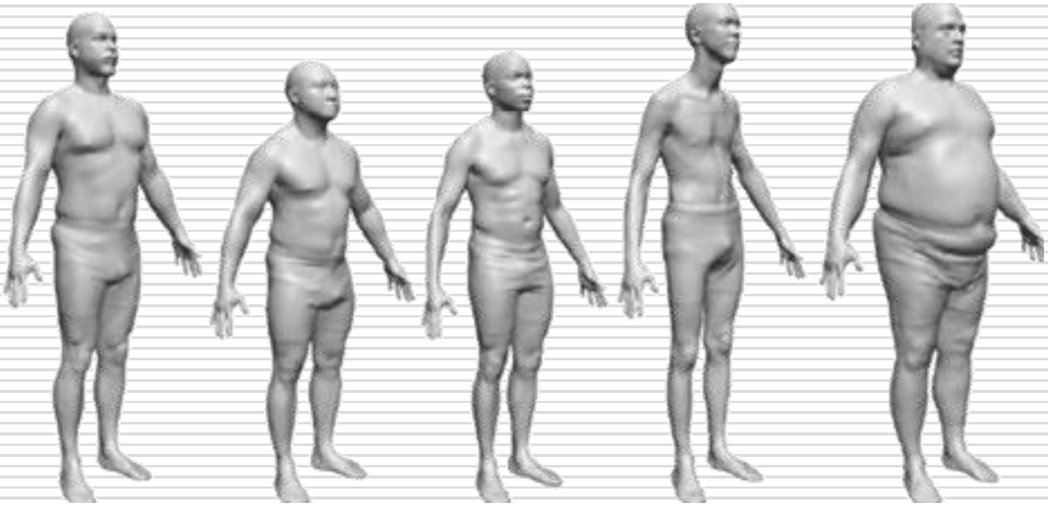
Remeshing & Surface Fitting

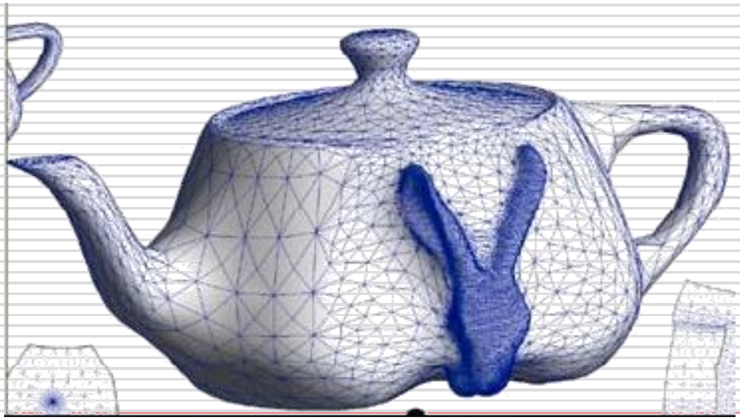


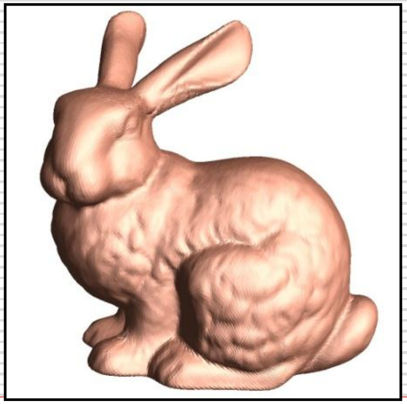


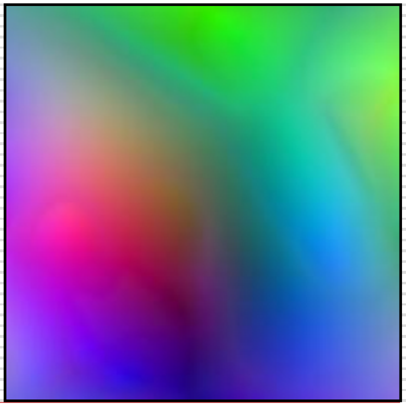


More Applications



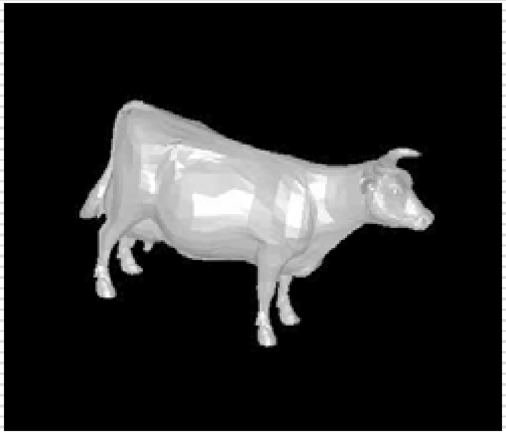


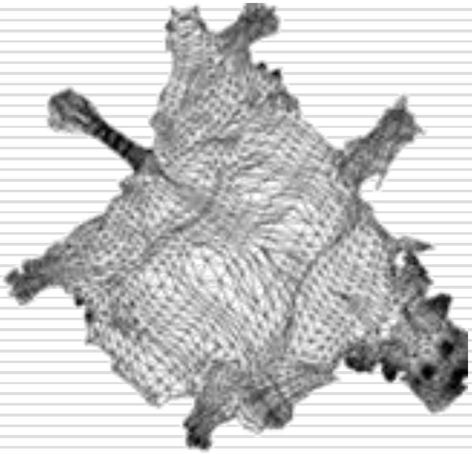




Mesh Parameterization (2D)

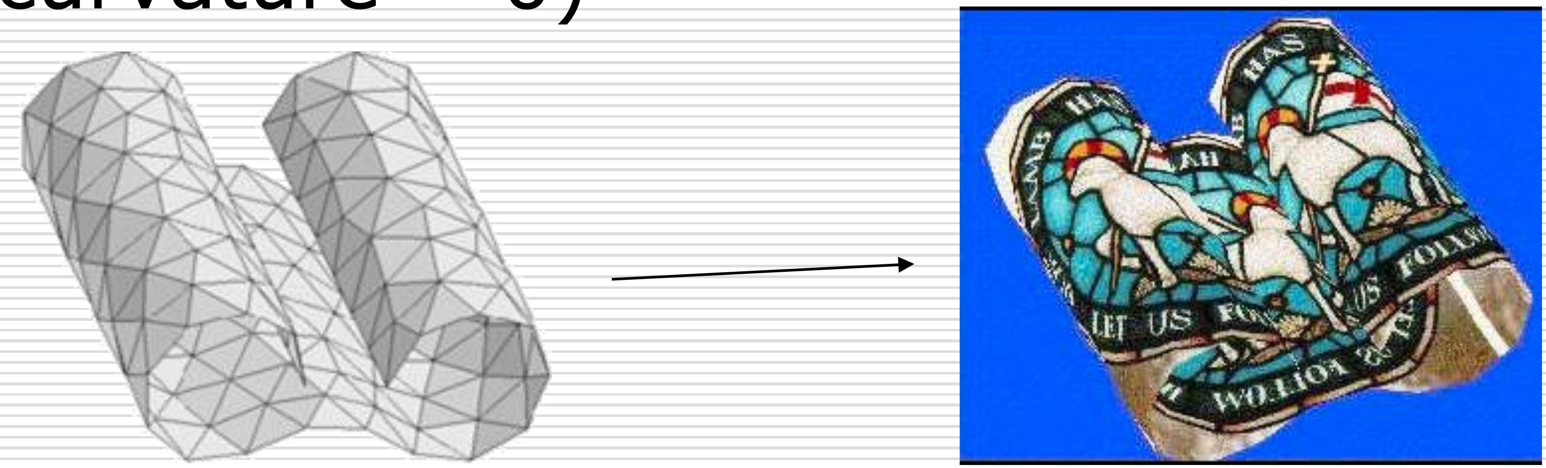
- Problem definition:
  - Input:
    - triangulated surface mesh in 3D
  - Output:
    - valid 2D mesh with same connectivity & minimal metric distortion
      - Mapping defined by vertex correspondence + barycentric coordinates
  - Validity – no inverted (overlapping) triangles





Parameterization in 2D

- Can do only for genus 0 surfaces with boundary
- Metrics preserved fully only for developable surfaces (Gaussian curvature = 0)



Distortion

- Function of second fundamental form

$$J = \begin{pmatrix} a & b \\ b & c \end{pmatrix} = \begin{pmatrix} \left(\dfrac{\partial f}{\partial u}\right)^2 & \dfrac{\partial f}{\partial u}\dfrac{\partial f}{\partial v} \\ \dfrac{\partial f}{\partial u}\dfrac{\partial f}{\partial v} & \left(\dfrac{\partial f}{\partial v}\right)^2 \end{pmatrix}$$

- Isometry (0-distortion) if J is identity matrix
- Components measure
  - Shear (angles)
  - Stretch (lengths)
  - Conformality: angles + equal stretch

Mapping Properties

- Validity: no folded over triangles

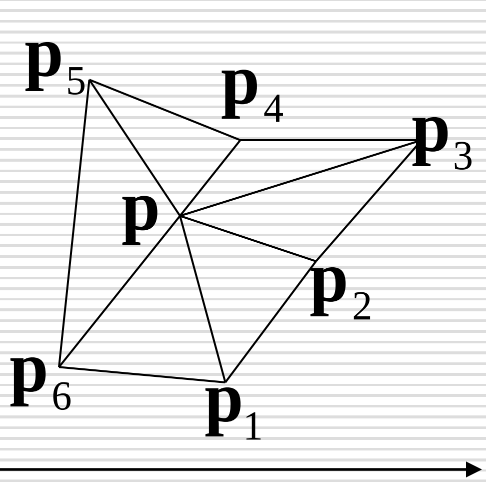
- Distortion – preserve (as mush as possible) lengths, angles, and areas
  - isometric (length-preserving) mappings
  - conformal (angle-preserving) mappings
  - equiareal (area-preserving) mappings

- (Theorem)
  isometric $\Leftrightarrow$ conformal + equiareal

Barycentric Combination

- $\mathbf{p} = \sum \lambda_i \mathbf{p}_i,$

  $i = 1...n, \sum \lambda_i = 1$



i.e., a combination of its neighbors

Barycentric Combination

Choose $\mathbf{u}_{n+1},\ldots,\mathbf{u}_N$ to be the vertices of any *K*-sided convex polygon in an anticlockwise sequence.

For each $i \in \{1,\ldots,n\}$, choose any set of real numbers $\lambda_{i,j}$ for $j = 1,\ldots,N$ such that

$$\lambda_{i,j} = \begin{cases} > 0 & (i,j) \in E \\ 0 & (i,j) \notin E \end{cases} \qquad \sum_{j=1}^{N} \lambda_{i,j} = 1$$

and define $\mathbf{u}_1,\ldots,\mathbf{u}_n$ to be the solution of the linear system of equations.

$$\mathbf{u}_i = \sum_{j=1}^{N} \lambda_{i,j} \mathbf{u}_j$$

Barycentric Combination

$$\mathbf{u}_i = \sum_{j=1}^{N} \lambda_{i,j} \mathbf{u}_j \Rightarrow \mathbf{u}_i - \sum_{j=1}^{n} \lambda_{i,j} \mathbf{u}_j = \sum_{j=n+1}^{N} \lambda_{i,j} \mathbf{u}_j$$

By considering the two components u_i and v_i of $\mathbf{u}_i$ separately this is equivalent to the matrix equation

$$\mathbf{A}\mathbf{u} = \mathbf{b}$$

where the matrix $\mathbf{A}$ is $n \times n$ having elements

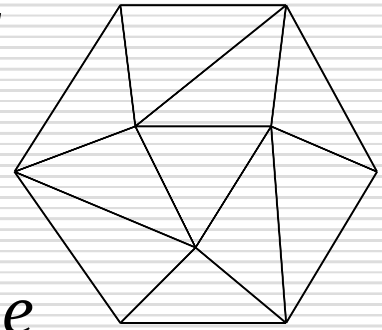
$$a_{i,j} = \begin{cases} 1 & i = j \\ -\lambda_{i,j} & i \neq j \end{cases}$$

2D Barycentric Embeddings

- Fix 2D boundary to convex polygon
- Define embedding as a solution of

$$\mathbf{W}x = b_x \qquad \mathbf{W}y = b_y$$

$$w_{ij} = \begin{cases} < 0 & (i,j) \in E \\ -\sum_{j \neq i} w_{ij} & (i,i) \\ 0 & otherwise \end{cases}$$



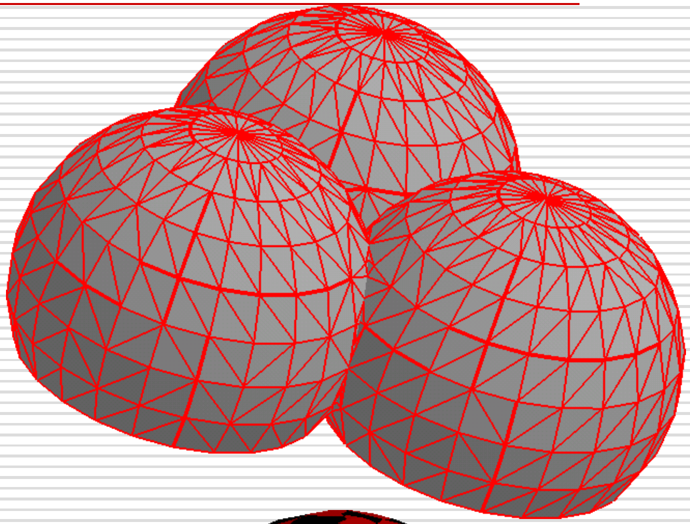
  - $\mathbf{W}$ is symmetric: $w_{ij} = w_{ji}$
  - weights w_{ij} control parameterization shape

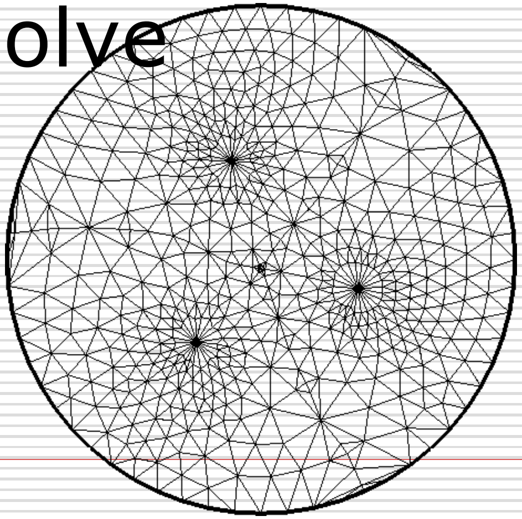
Weights – Uniform (Tutte)

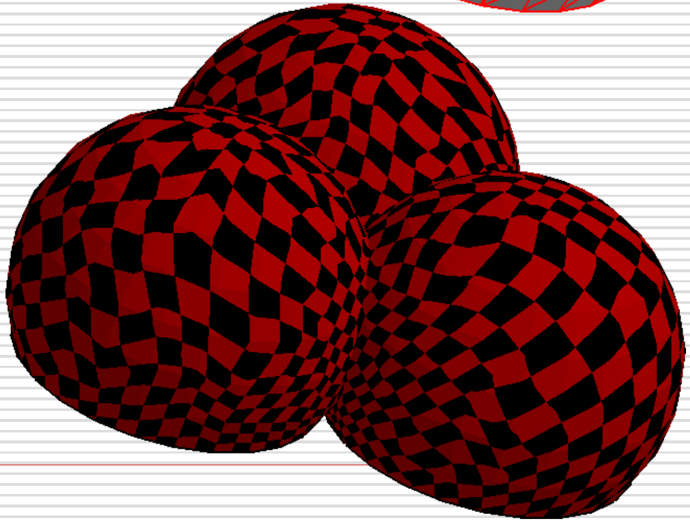
- Set

$$w_{ij} = 1 \quad \left(\lambda_{i,j} = \frac{1}{d_i} \right)$$

- No shape information
  - equilateral triangles
- Fastest to solve



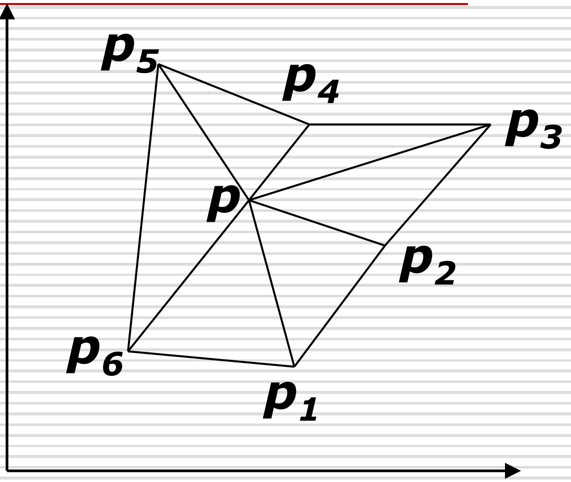




Weights – Shape Preserving (Floater)

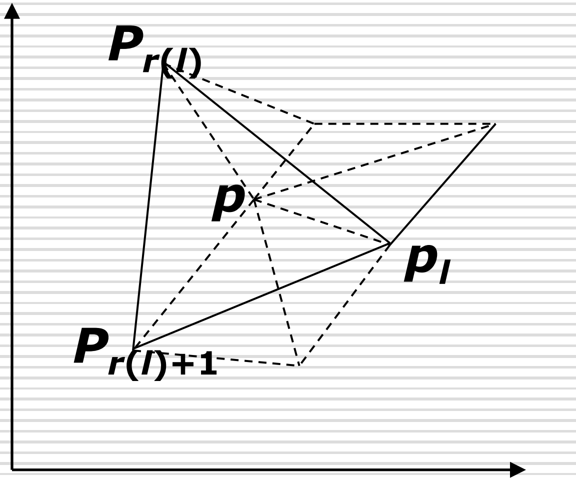
For each $l \in \{1,\ldots,d_i\}$, $\overrightarrow{\mathbf{p}_l\mathbf{p}}$ will intersect $\overline{\mathbf{p}_{r(l)}\mathbf{p}_{r(l)+1}}$, then we can have

$$\mathbf{p} = \delta_1\mathbf{p}_l + \delta_2\mathbf{p}_{r(l)} + \delta_3\mathbf{p}_{r(l)+1} \quad \delta_1 + \delta_2 + \delta_3 = 1$$



Define $\mu_{k,l}$, for $k = 1,\ldots,d_i$, by $\mu_{l,l} = \delta_1$, $\mu_{r(l),l} = \delta_2$, $\mu_{r(l)+1,l} = \delta_3$, and $\mu_{k,l} = 0$ otherwise. Then for each l we now find

$$\mathbf{p} = \sum_{k=1}^{d_i} \mu_{k,l}\mathbf{p}_k \qquad \sum_{k=1}^{d_i} \mu_{k,l} = 1$$



Finally define $\lambda_{i,j} = \dfrac{1}{d_i}\sum_{l=1}^{d_i} \mu_{j,l}$

Weights – Harmonic Mapping

- Quasi – conformal : minimize angular distortion
- Locally, preserving angles preserves distances (ratios)
- For fixed boundary have unique *harmonic* map which minimizes conformal energy
  - Rieman theorem: any C^1 continuous surface in R^3 can be mapped conformally to fixed domain in R^2
  - Nearly true for meshes

Weights – Harmonic Mapping

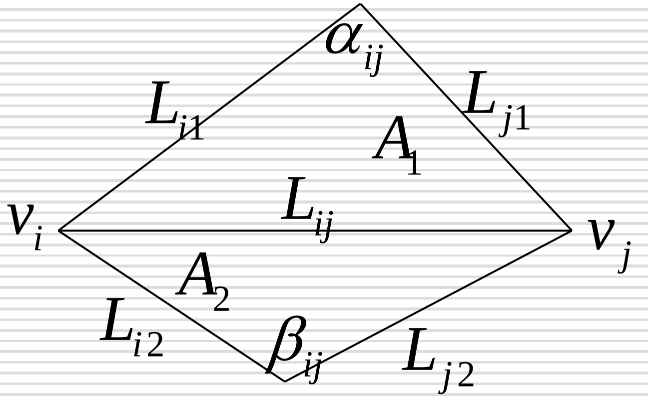
- Approximate harmonic map for fixed boundary
- Represent as configuration of springs on mesh edges

$$E(v) = \frac{1}{2} \sum_{(i,j)} w_{ij} \left\| v_i - v_j \right\|^2$$

- Spring coefficients

$$w_{ij} = \frac{L_{i1}^2 + L_{j1}^2 - L_{ij}^2}{A_1} + \frac{L_{i2}^2 + L_{j2}^2 - L_{ij}^2}{A_2}$$

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



  - L_{ij} - edge length in 3D
  - A_{ijk} - triangle area in 3D
  - α_{ij} and β_{ij} - opposing angles in 3D

Weights – Conformal Mapping

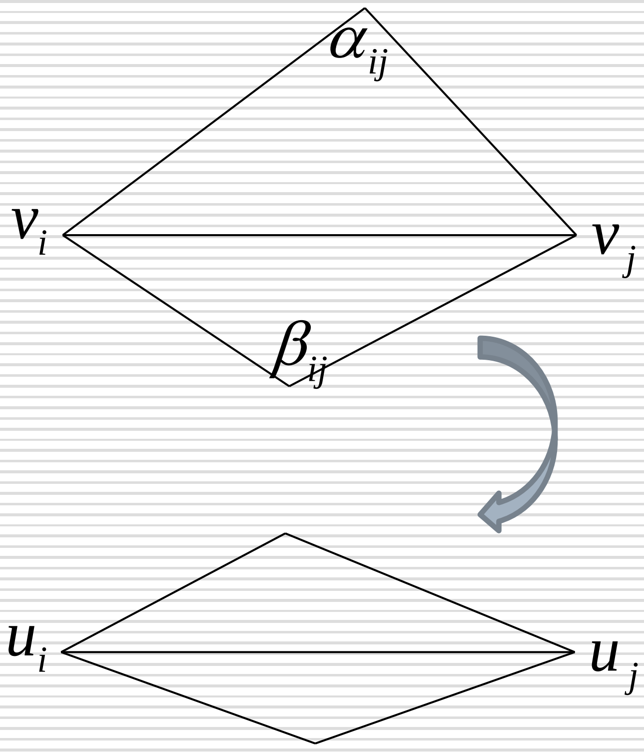
- Represent as configuration of springs on mesh edges

$$E(v) = \sum_{(i,j)} w_{ij} \left\| u_i - u_j \right\|^2$$

- Spring coefficients

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

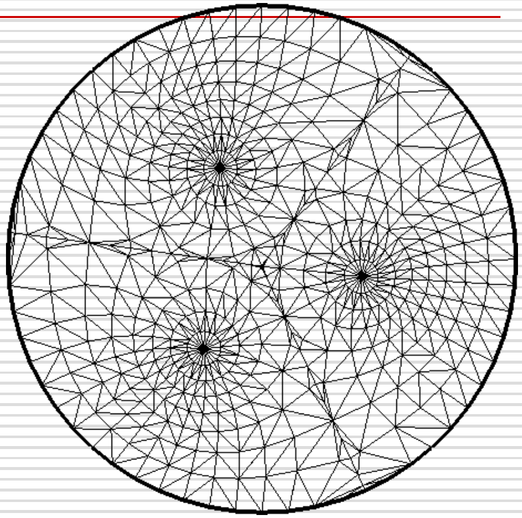
  - α_{ij} and β_{ij} - opposing angles in 3D

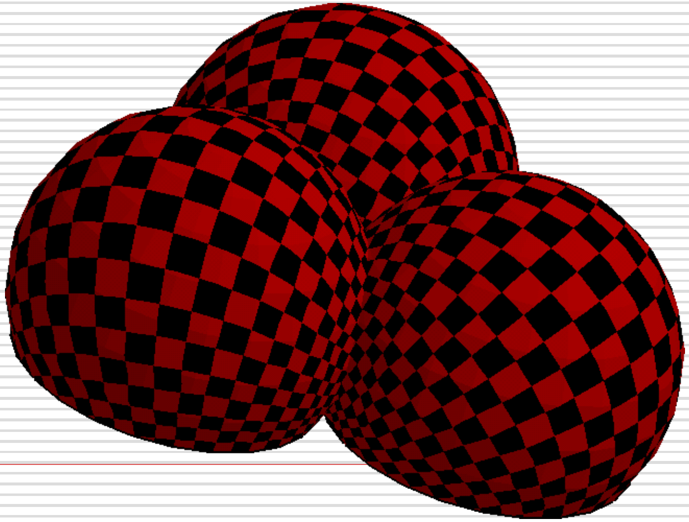


Barycentric Formulation

- $E(v)$ minimum reached when gradient equal 0

$$\frac{\partial E(v)}{\partial v_i} = \sum_j w_{ij}(v_i - v_j) = 0$$



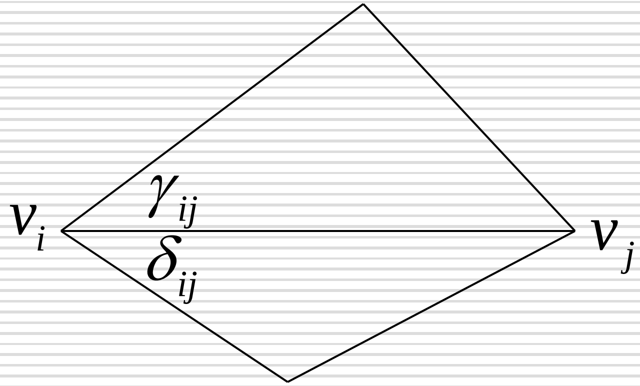


- Barycentric embedding formulation
- Can have negative weights – does not guarantee validity

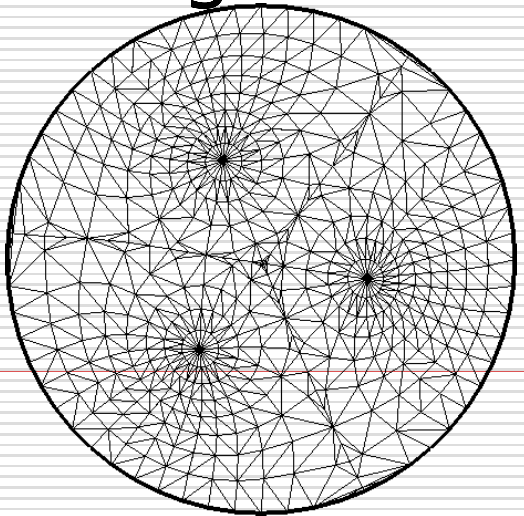
Weights – Mean Value

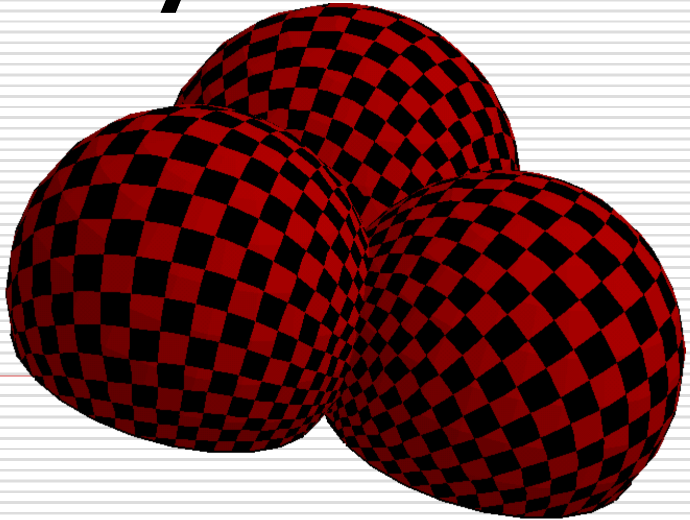
- Set

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

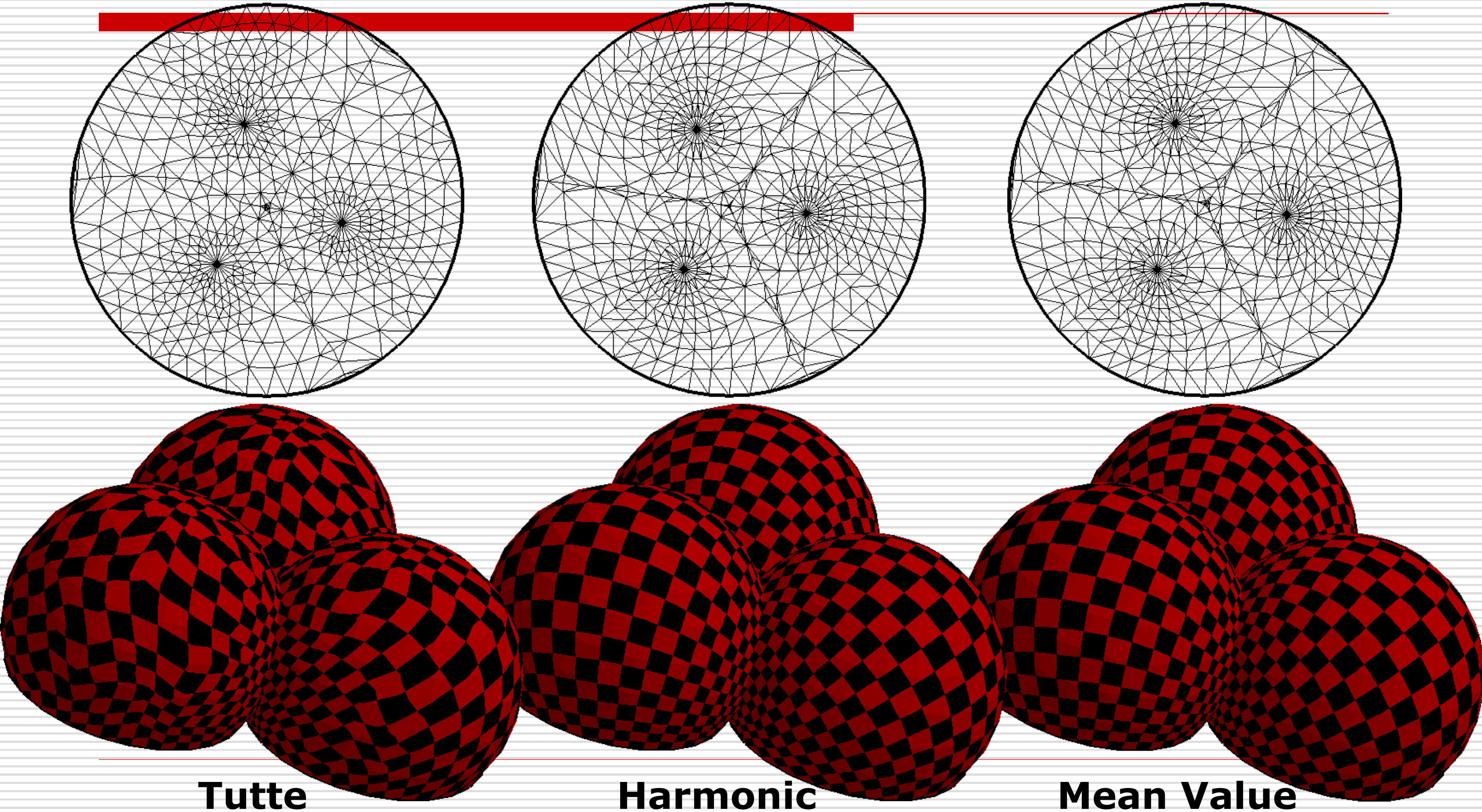


- Result visually identical to conformal
- No negative weights – **always** valid





Comparison



Tutte **Harmonic** **Mean Value**

Practical Implementation

- Boundary
  - Popular options: Square, circle, triangle
  - Application specific
    - Reconstruction – rectangle
    - Mapping to base mesh– triangle
    - Spreading points along boundary
    - Cord length
- Solve

$$\mathbf{W}x = b_x$$

$$\mathbf{W}y = b_y$$

  - Right hand side determined from boundary vertices

Practical Implementation

- Solving linear system expensive ($O(n^3)$)
- Use iterative solution:
  - Get initial guess for interior nodes
  - While conditions not met:
    - Set each interior node to weighted average of neighbors:

$$v_i = \frac{1}{\sum_{(i,j)} w_{ij}} \sum_{(i,j)} w_{ij} v_j$$

  - Stopping conditions:
    - Convergence: vertices do not move
    - Exceed maximal number of iterations
    - Parameterization is valid
- Solution exists & is reached thanks to matrix structure

Fixed vs. Free Boundary

- Fixed
  - Useful when boundary fixed a priory (e.g. mapping to base mesh)
  - Increase distortion
- Free
  - Typically less distortion

Local Unfolding

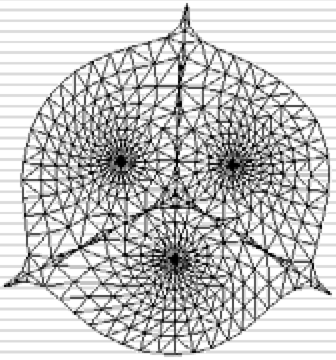
- While not all mesh flattened
  - Select seed triangle & map as is to 2D
  - Define front - boundary of unfolded patch
  - Assign cost to each vertex adjacent to boundary−amount of distortion caused by mapping it to 2D
  - Map best current vertex to 2D (if cost below threshold), add it to front & recompute adjacent costs

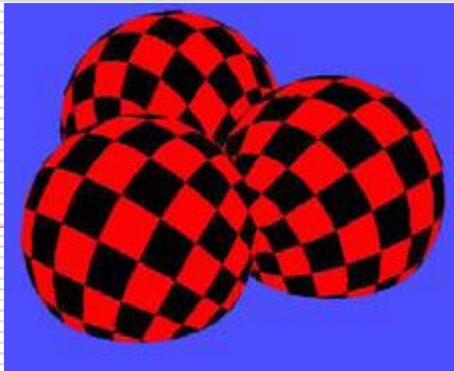
Local Unfolding

- Advantages
  - Bounded distortion
  - Simple
- Drawback
  - Generate long seams – parameterization/texture discontinuities

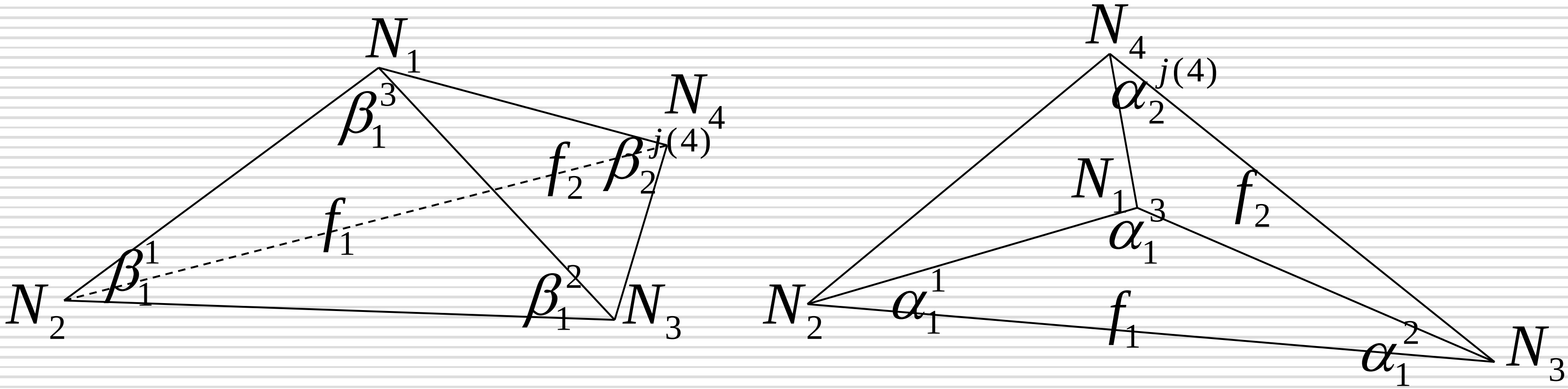
Angle Based Flattening (ABF)

- Fact:
  - Triangular 2D mesh is defined by its angles
- Define problem in angle space
- Angle based formulation:
  - Distortion as function of angles
  - Validity - set of angle constraints





Constrained Minimization



- Objective: minimize (relative) deviation of angles

$$F(\alpha) = \sum_{i,j} w_i^j (\alpha_i^j - \beta_i^j)^2$$

- Initial choice for weights:

$$w_i^j = \beta_i^{j-2}$$

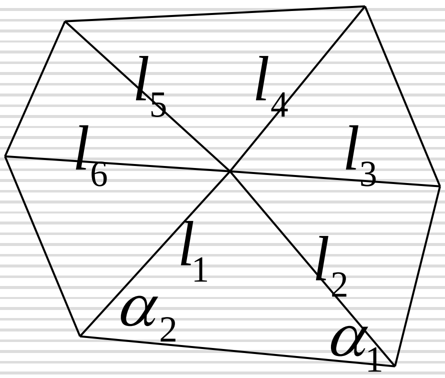
Constraints

$$g^1(\alpha) \equiv \alpha_i^j \geq \varepsilon$$

$$g^2(\alpha) \equiv \alpha_i^1 + \alpha_i^2 + \alpha_i^3 = \pi$$

$$g^3(\alpha) \equiv \sum_k \alpha_i^{j(k)} = 2\pi$$

$$g^4(\alpha) \equiv \prod_k \sin(\alpha_i^{j(k)-1}) - \prod_k \sin(\alpha_i^{j(k)+1}) = 0$$



$$\frac{l_1}{l_2} = \frac{\sin(\alpha_1)}{\sin(\alpha_2)}$$

$$\frac{l_1}{l_2} \cdots \frac{l_6}{l_1} = \frac{\sin(\alpha_1)}{\sin(\alpha_2)} \cdots \frac{\sin(\alpha_6)}{\sin(\alpha_1)}$$

Solution

- Use Lagrange Multipliers

$$F^*(\alpha,\mu) = F(\alpha) + \mu_1 g^2(\alpha) + \mu_2 g^3(\alpha) + \mu_3 g^4(\alpha)$$

- Solve the min-max problem (minimum on α , maximum on μ)
- Reached when all derivatives are zero
- Have non-linear system of equations
- Use Newton method to solve

ABF Summary

- Advantages
  - No fixed boundary – less distortion
  - No flipped triangles
  - Proven to converge to solution for any valid input
- Drawbacks
  - Expensive – solve non linear system
  - Linear sub-systems can’t be solved iteratively FAST
  - Can have boundary overlaps
  - Can’t handle multiple boundary loops