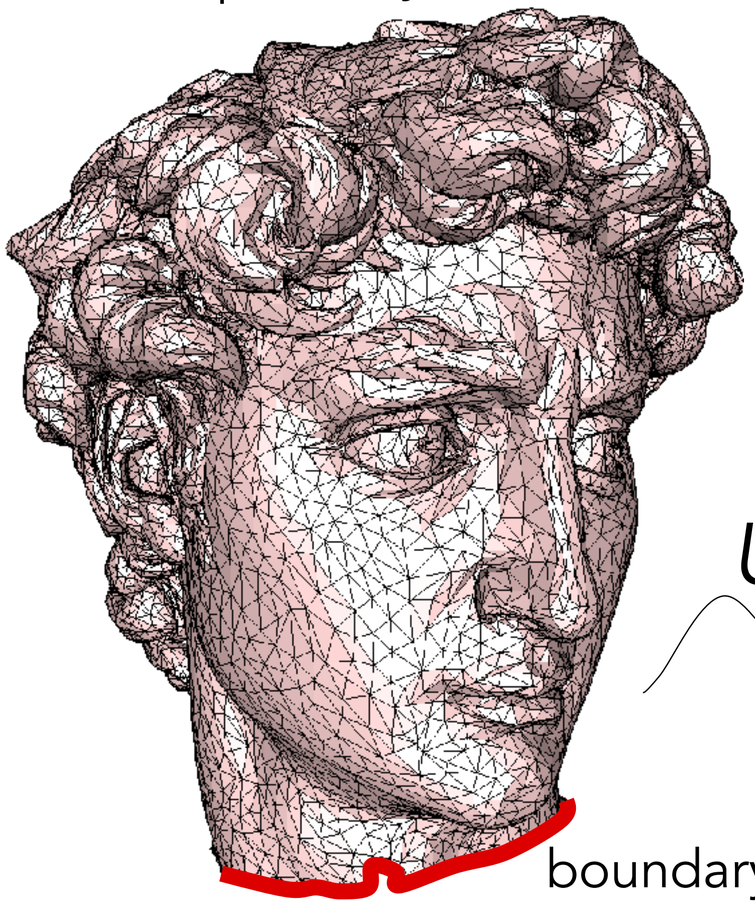
# Single Patch Parametrization

*Acknowledgements: Olga Sorkine-Hornung*

# Surface Parameterization

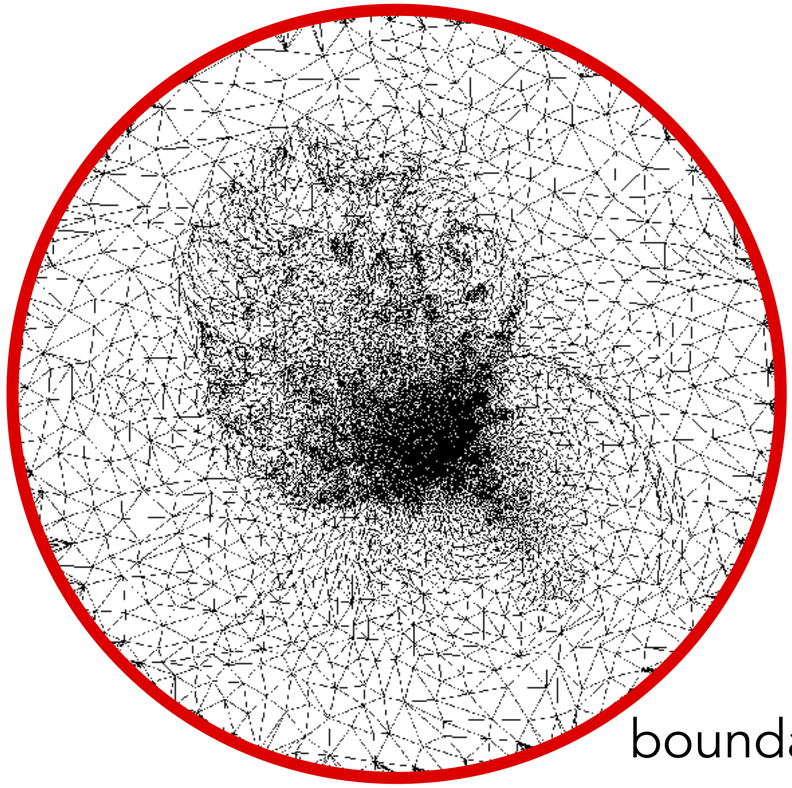
3D space (*x*,*y*,*z*)



boundary

*U*

2D parameter domain (*u*,*v*)



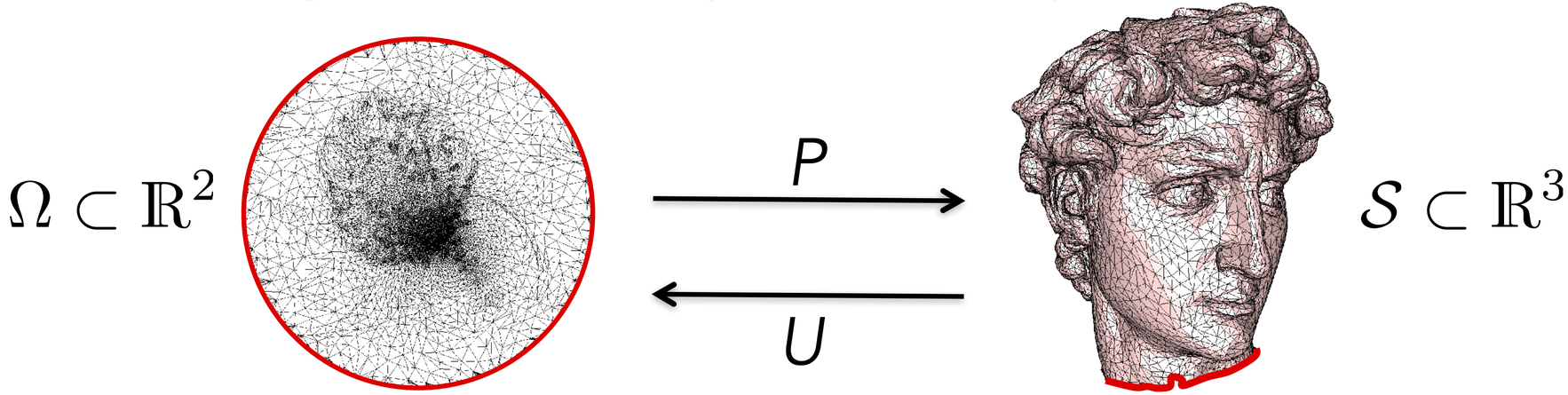
boundary

# Parameterization – Definition

- Mapping *P* between a 2D domain Ω and the mesh *S* embedded in 3D (the inverse = flattening)
- Each mesh vertex has a corresponding 2D position:

$$U(\mathbf{v}_i) = (u_i, v_i)$$

- Inside each triangle, the mapping is affine (barycentric coordinates)

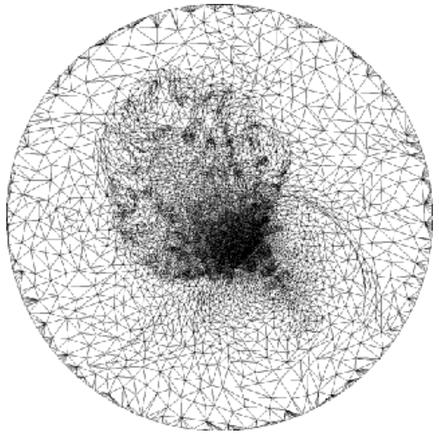


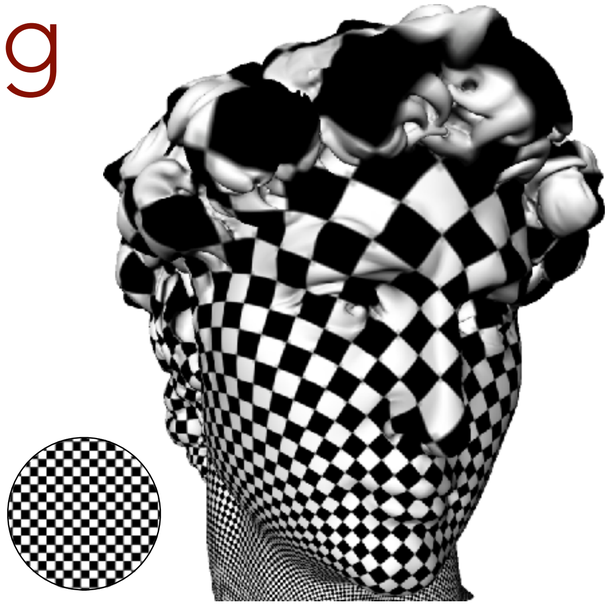
# Why?

# Why?

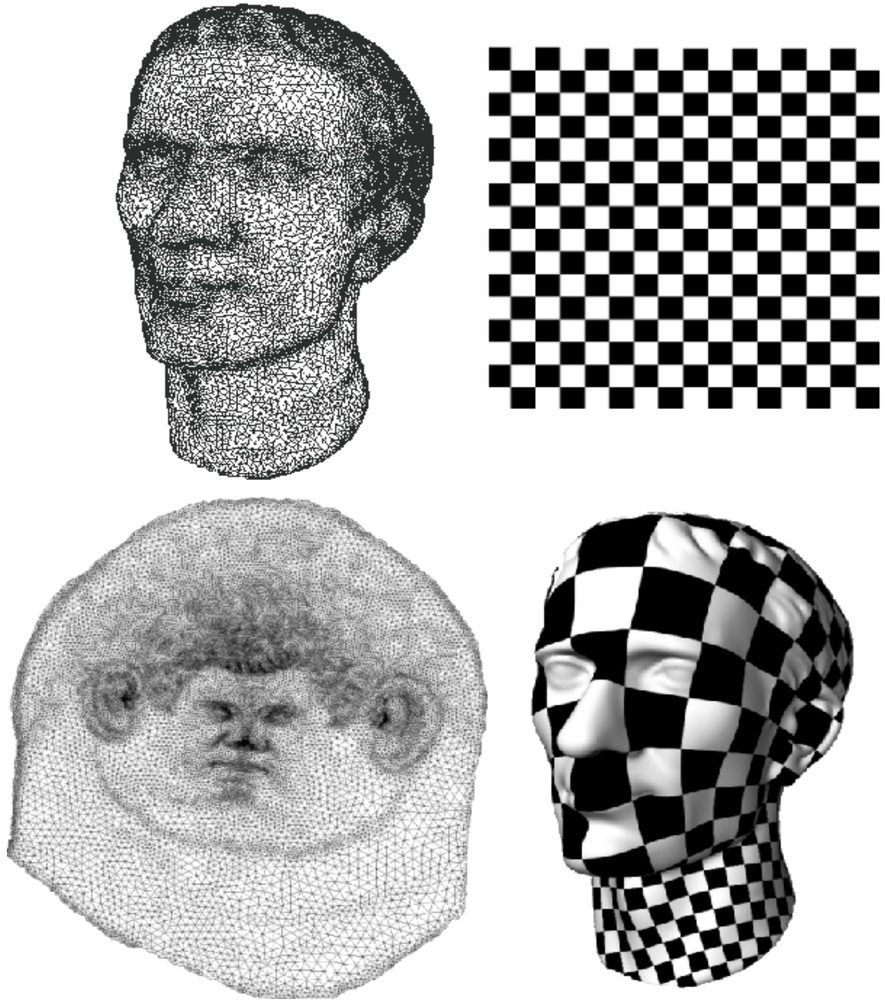
- Allows us to do processing in 2D and then map the results onto the 3D surface
- It is easier to operate in the 2D domain (but you have to create a parametrization first)

# Texture Mapping





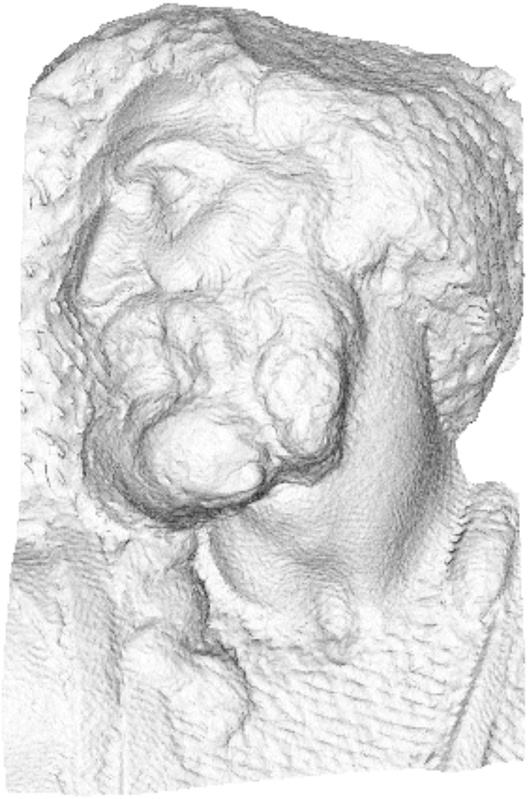
# Texture Mapping



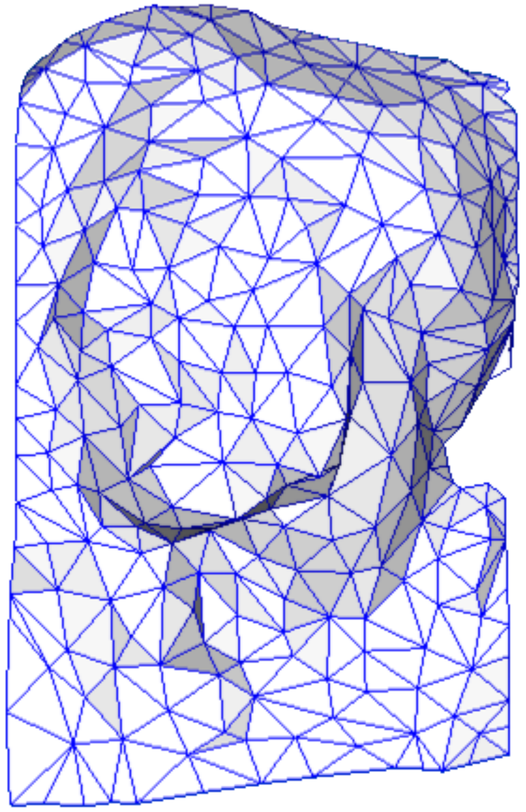
# Texture Mapping

Image from Vallet and Levy, techreport INRIA

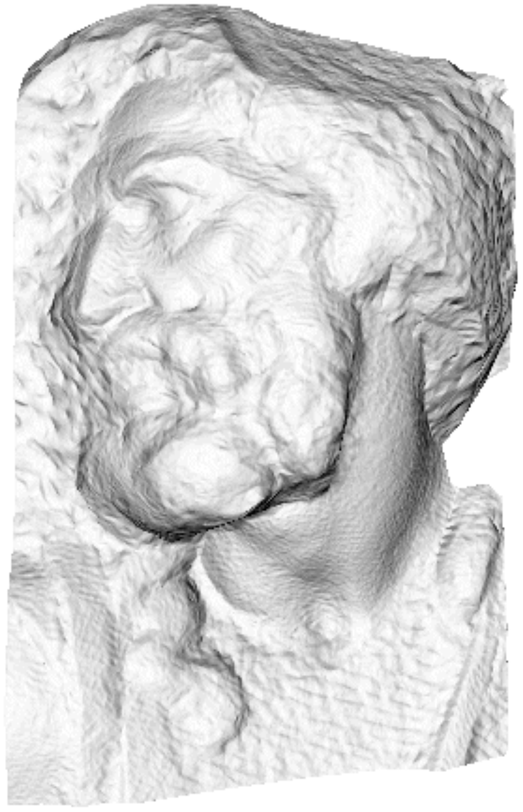
# Normal/Bump Mapping



original mesh
4M triangles

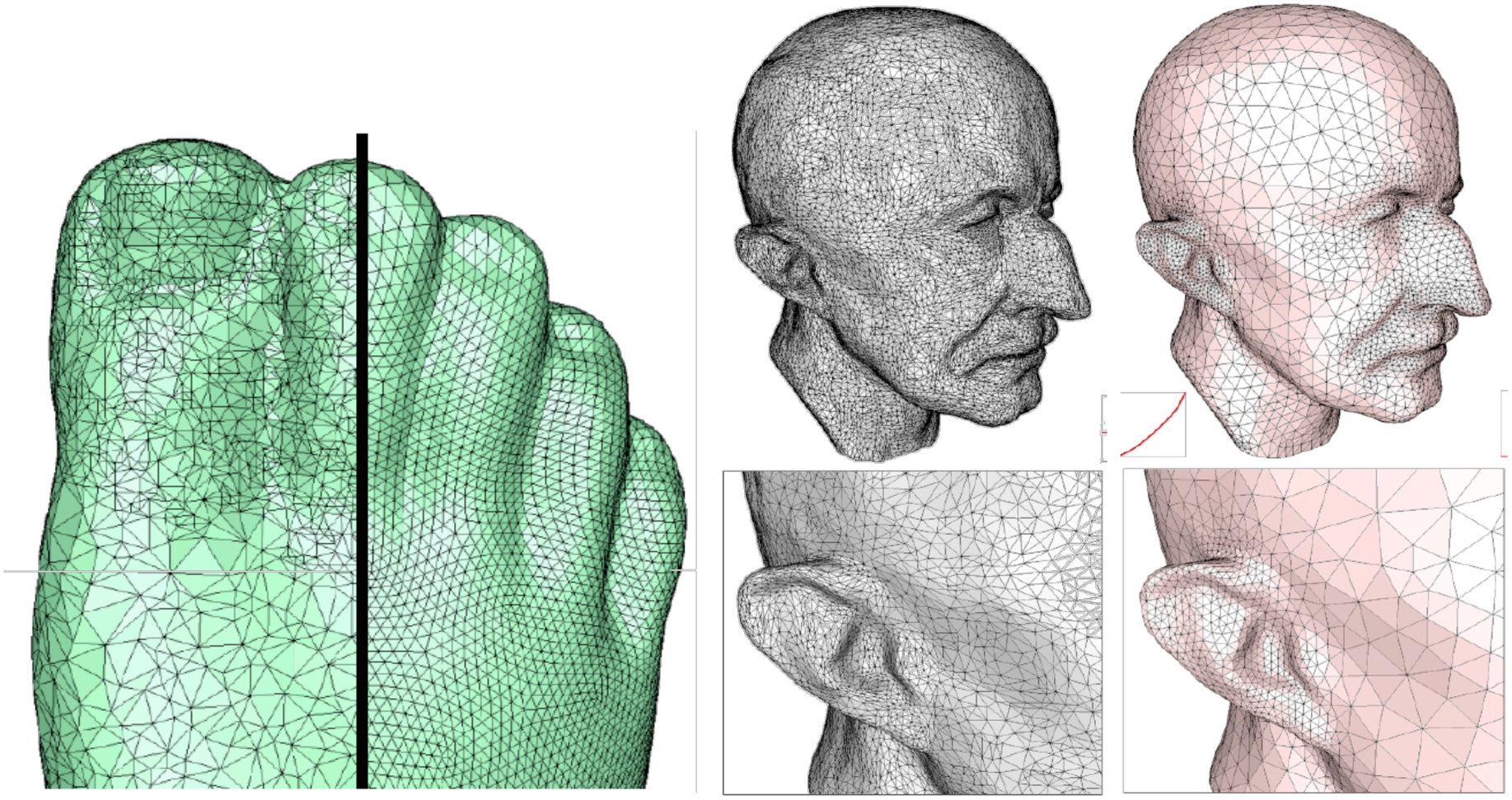


simplified mesh
500 triangles



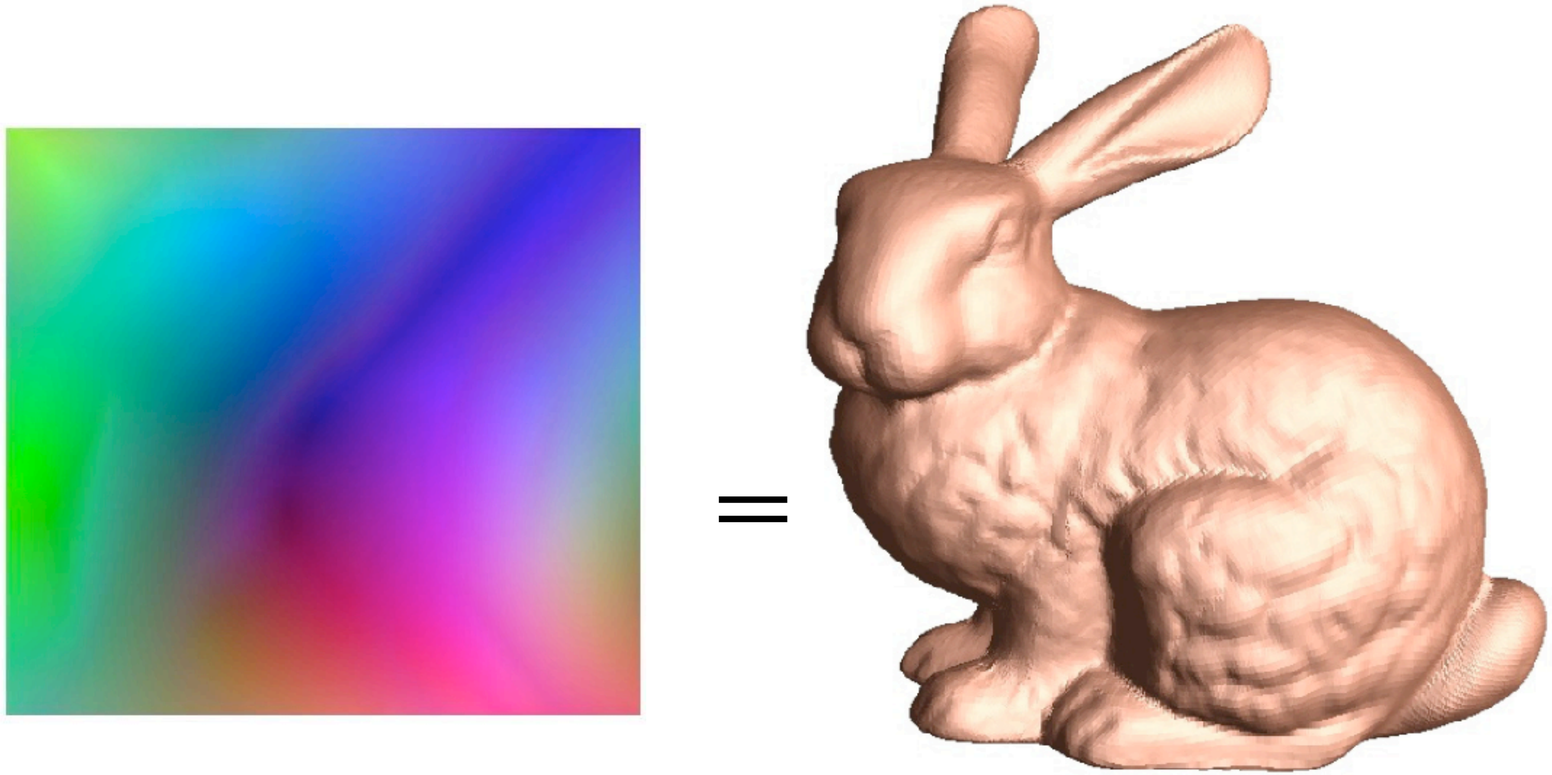
simplified mesh
and normal mapping
500 triangles

# Remeshing



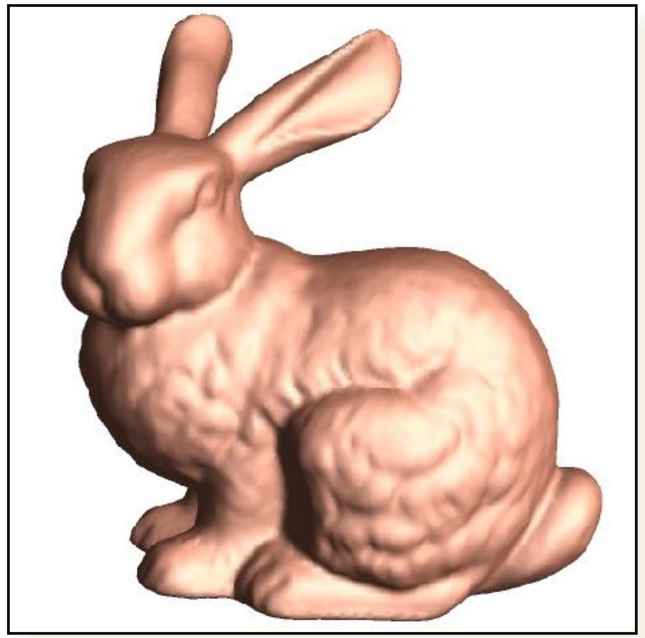
"Interactive Geometry Remeshing", Alliez et al., SIGGRAPH 2002

# Compression

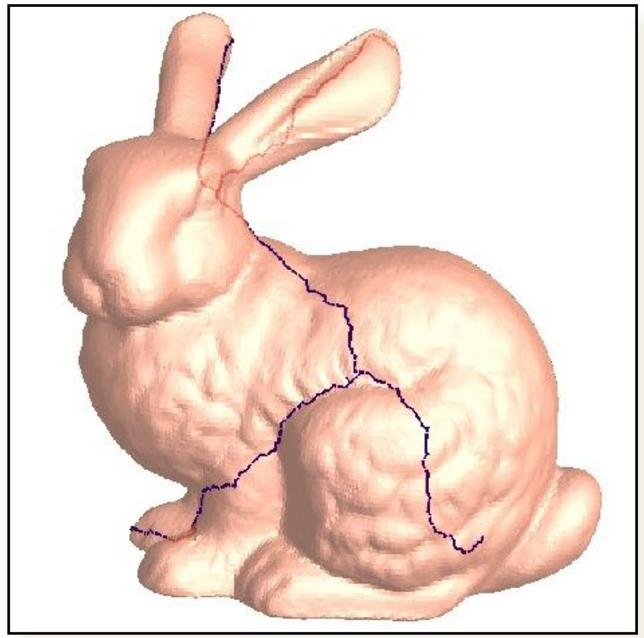


"Geometry images", Gu et al., SIGGRAPH 2002
http://research.microsoft.com/en-us/um/people/hoppe/proj/gim/

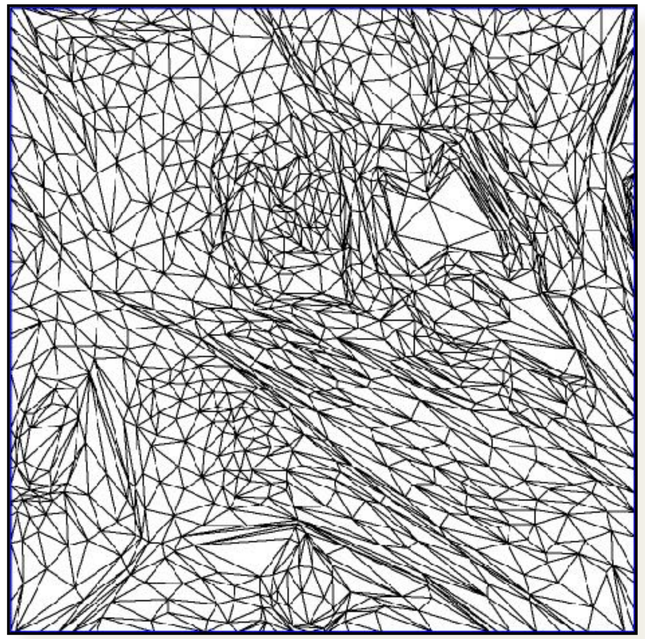
# Geometry Images



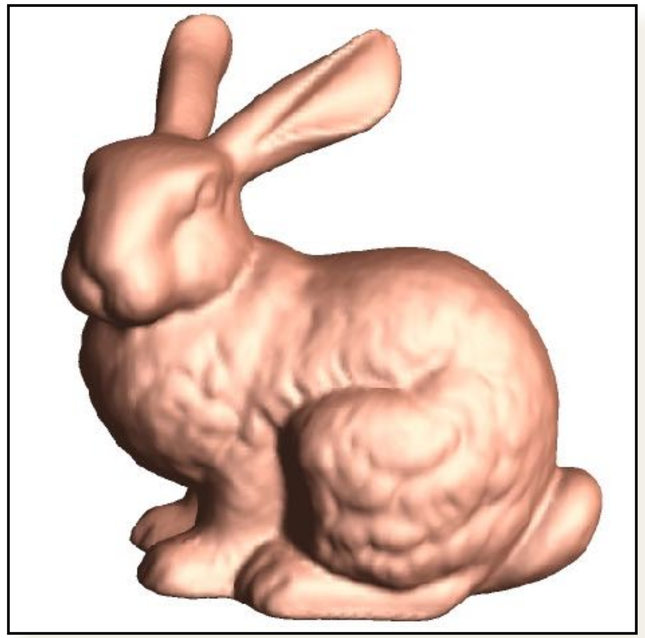
cut



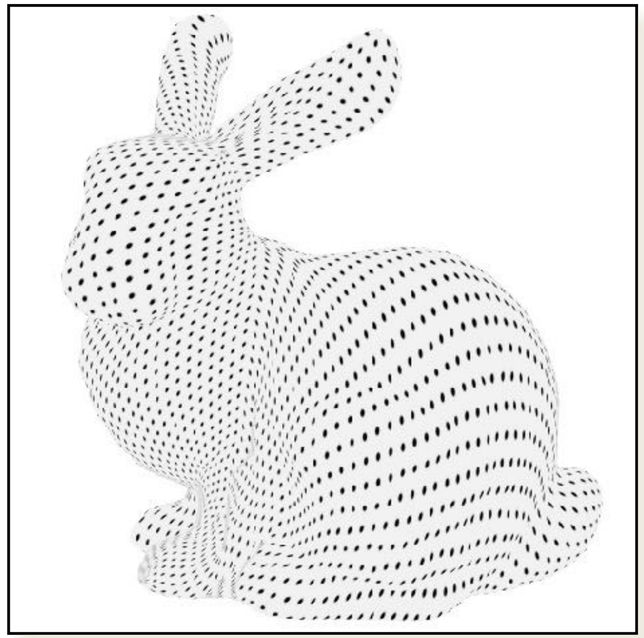
parametrize



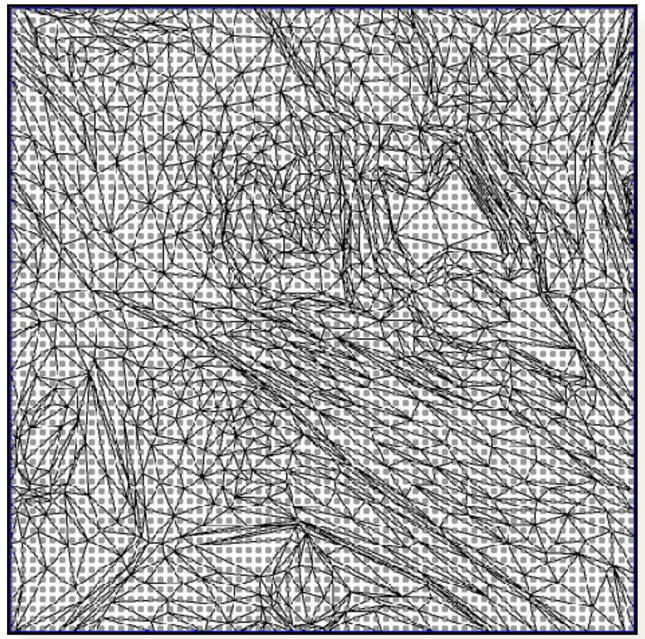
# Geometry Images



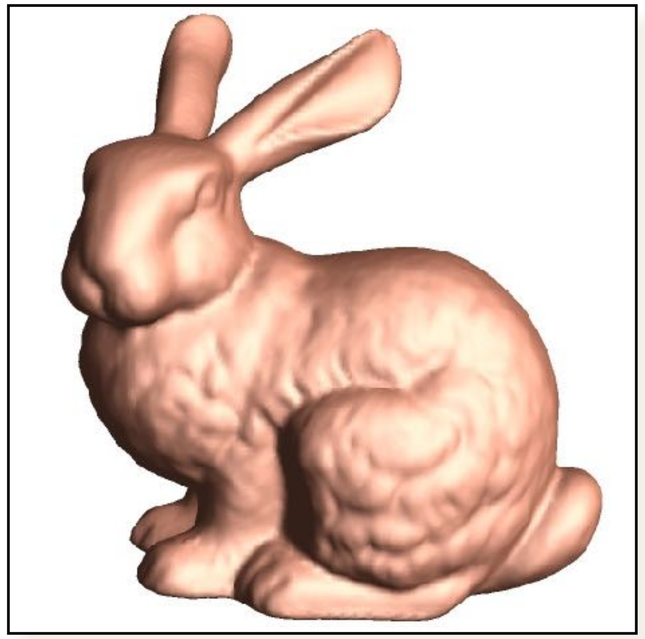
cut



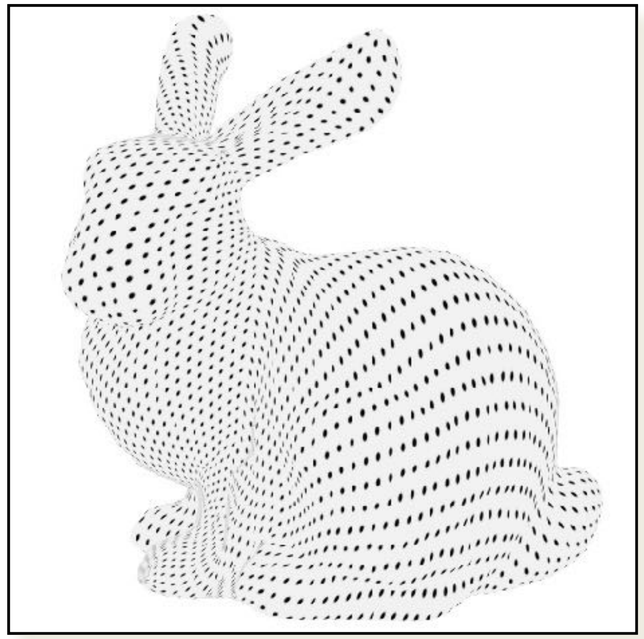
sample



# Geometry Images

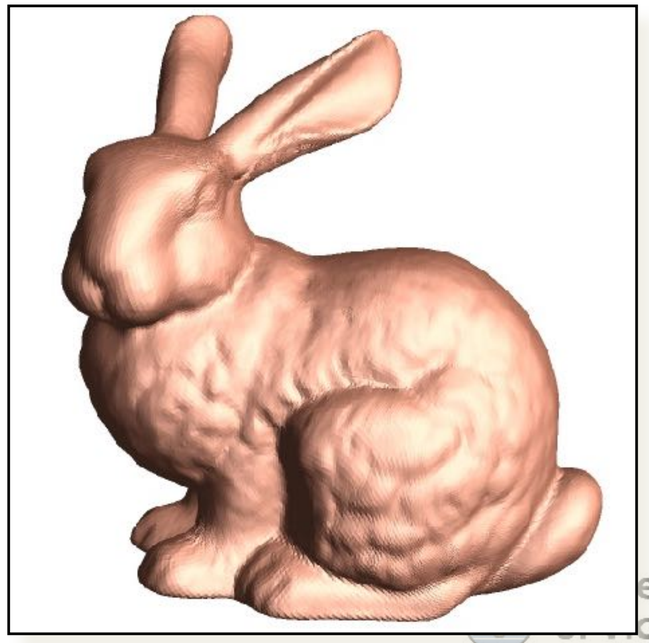


cut



store

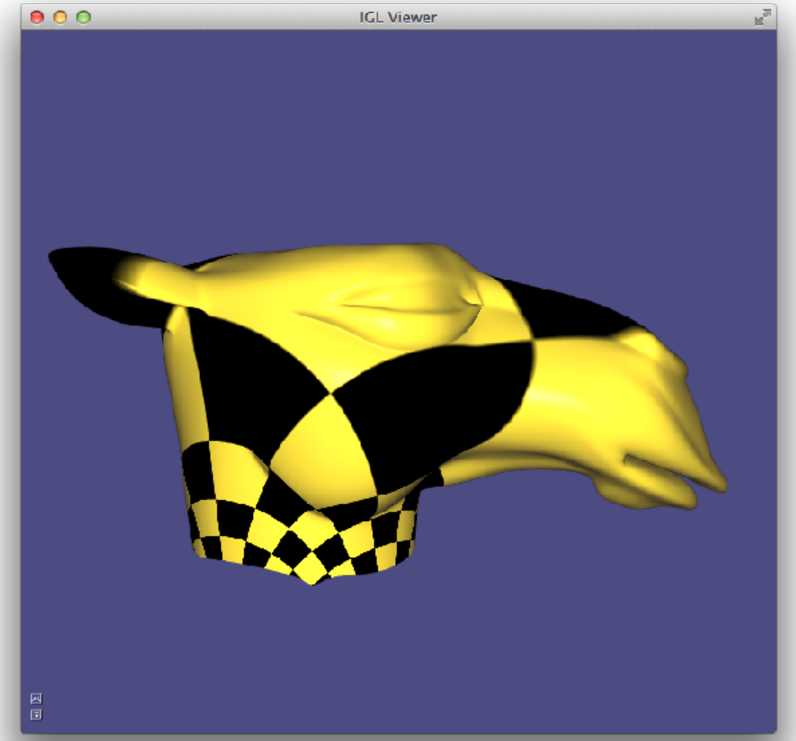
render

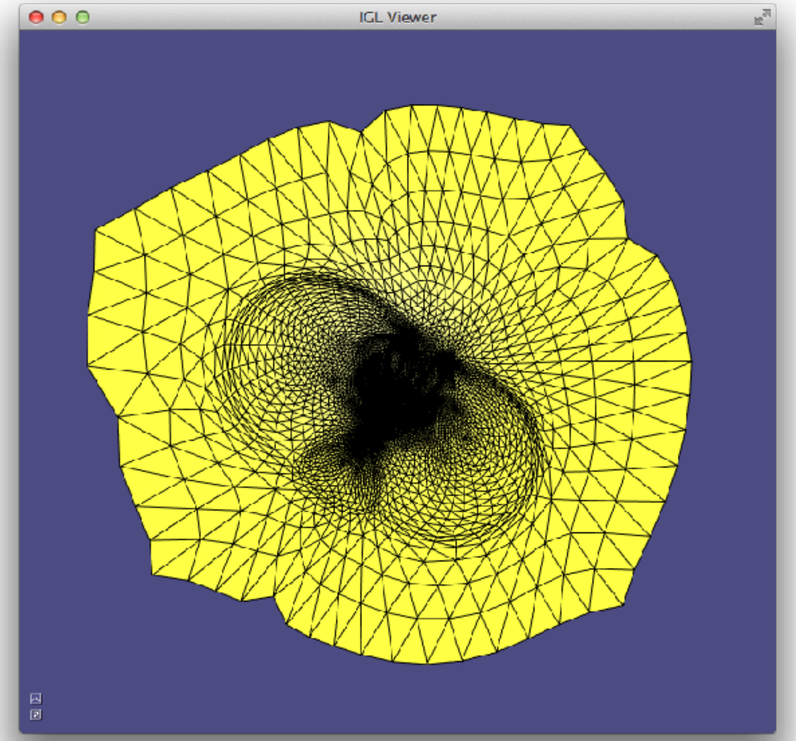


$[r,g,b] = [x,y,z]$

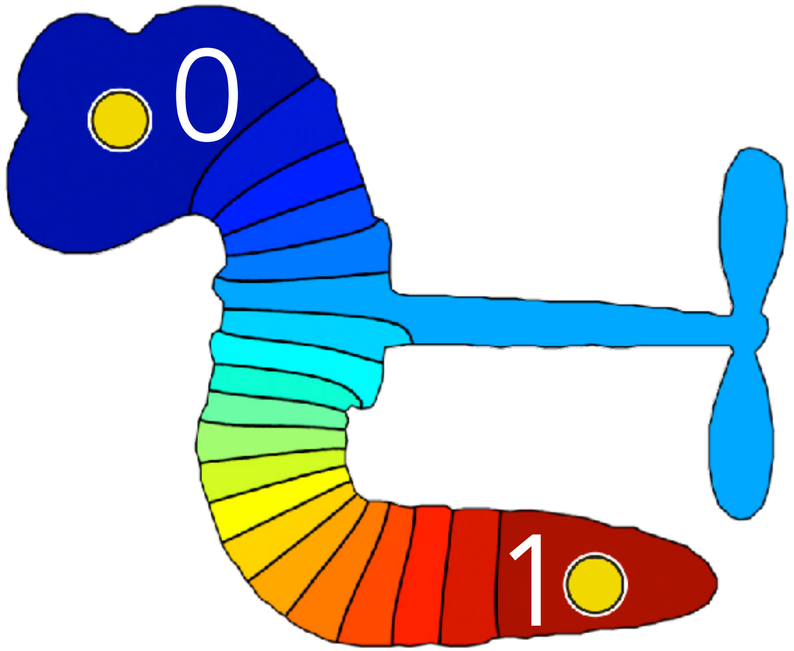
# Designing Scalar Functions

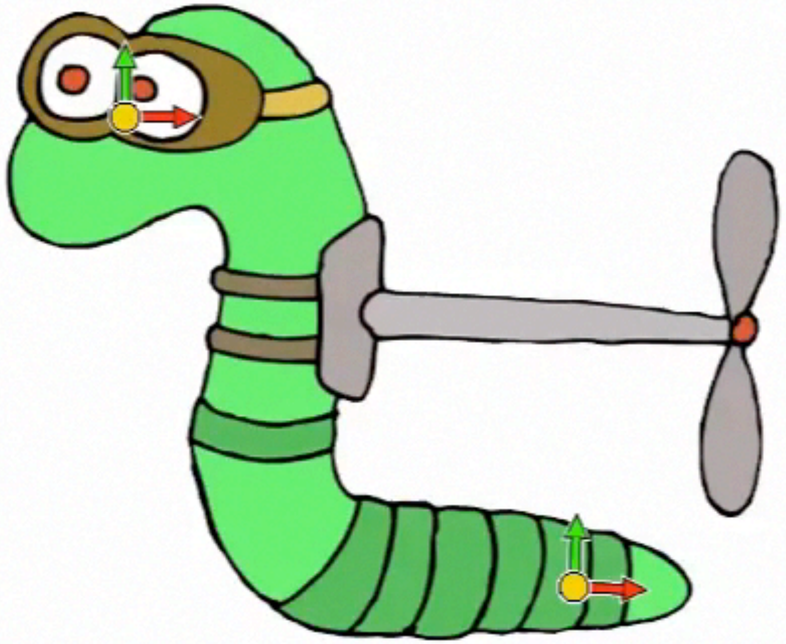
# Parametrization = 2 Scalar Functions



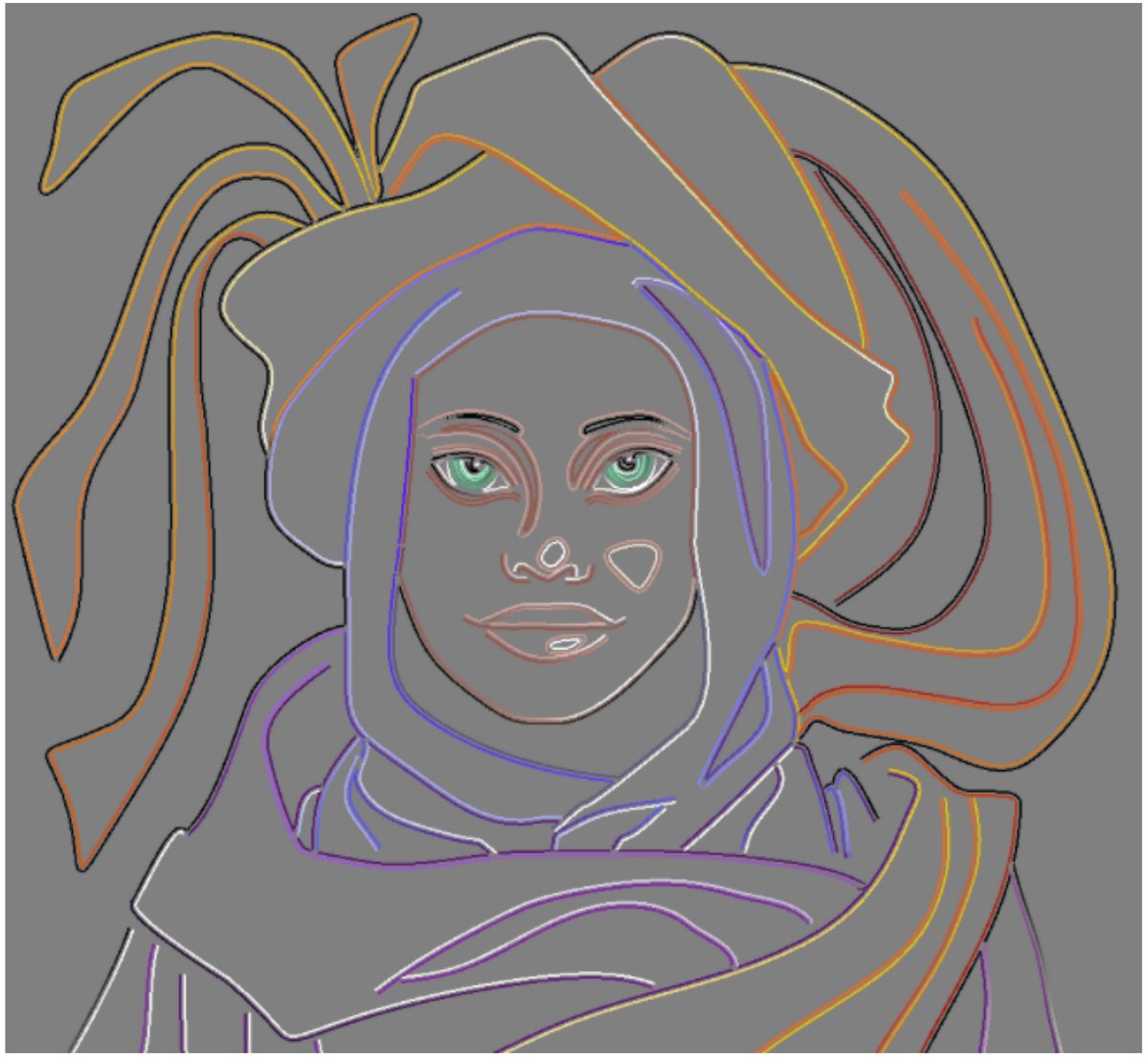
# Designing Scalar Functions



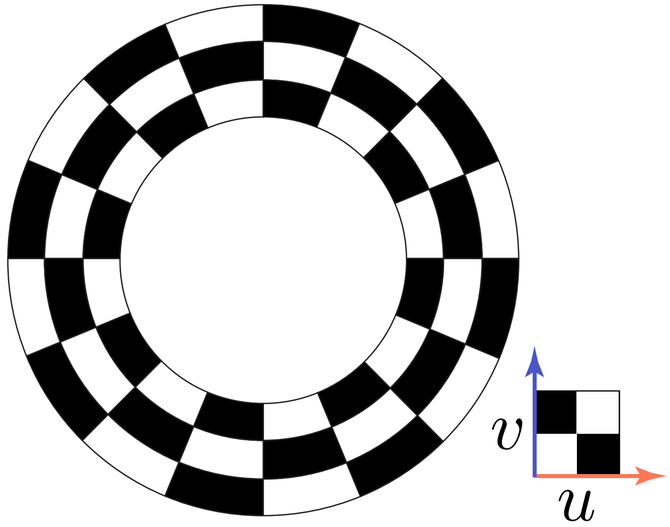
Jacobson et al. 2012

# Designing Scalar Functions

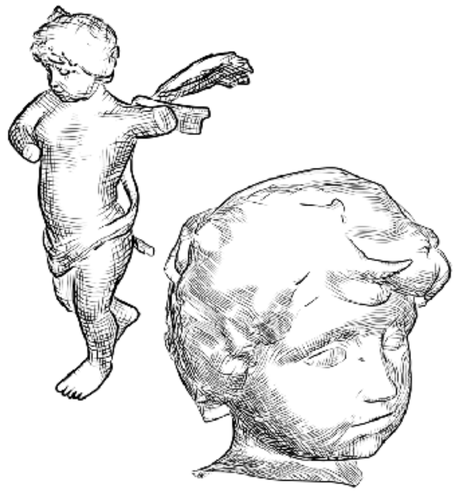


Orzan et al. 2008

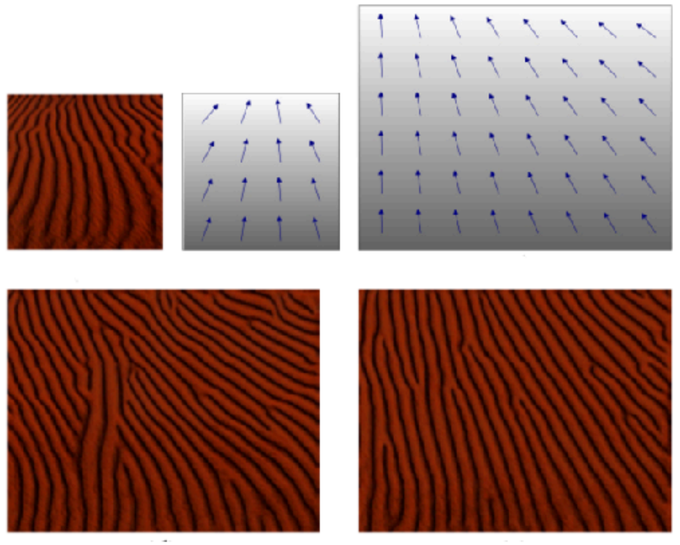
# Two Scalar Functions



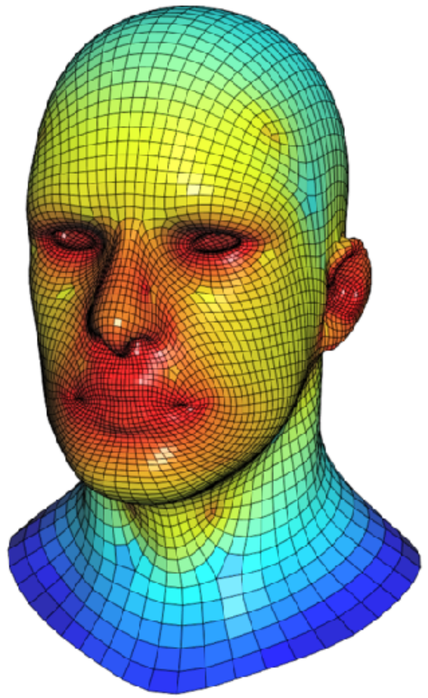
# Designing Gradients



Hatching

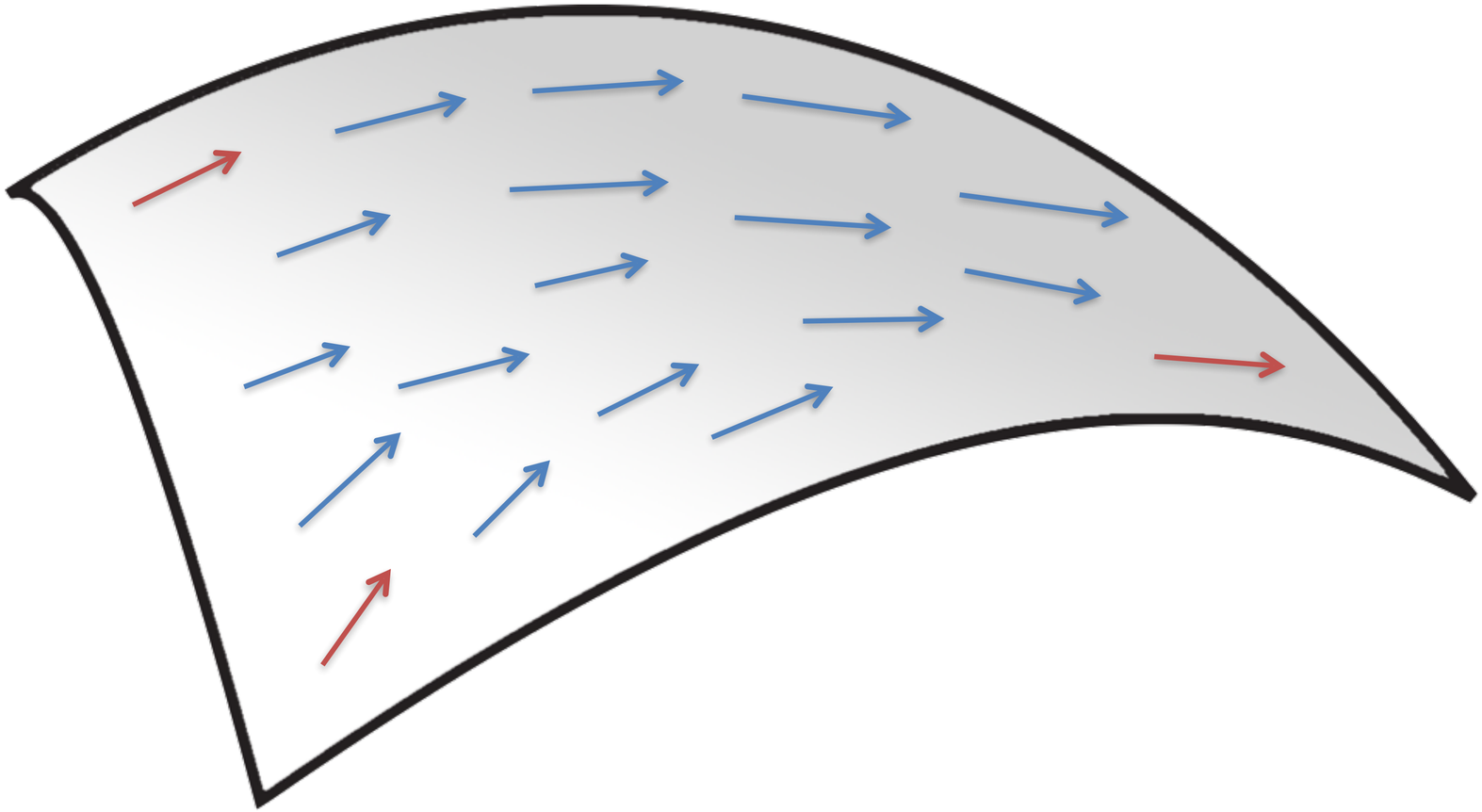


Texture Synthesis

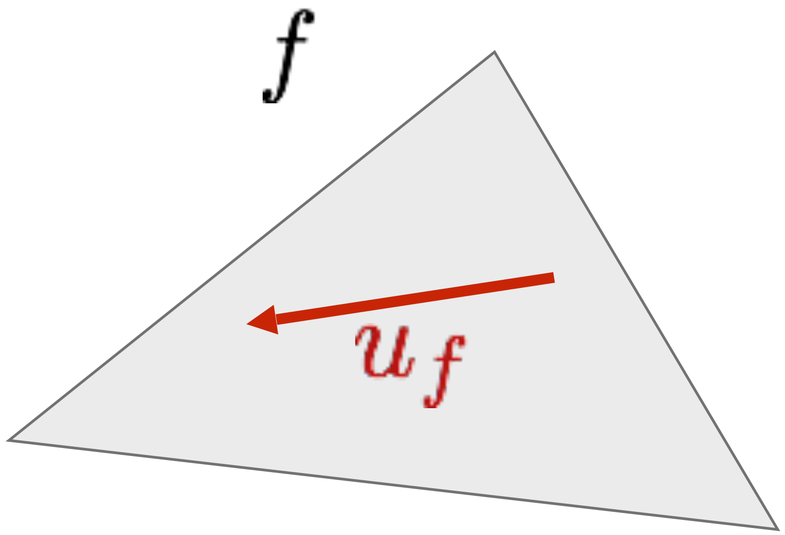


Meshing

# Intepolating Vectors



# Poisson Problem




Condition per triangle

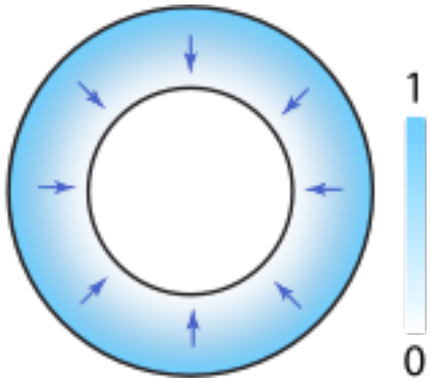
$$(\nabla s)_f = u_f$$

Error per triangle

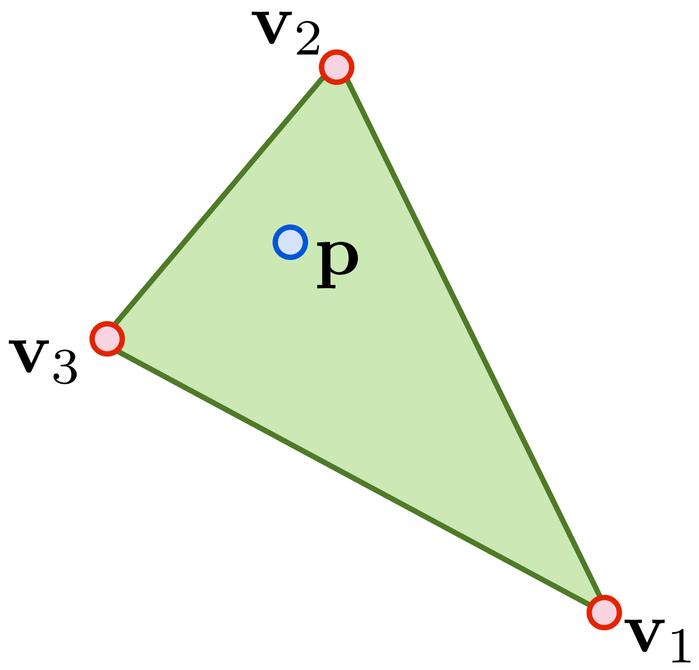
$$P_f = ||(\nabla s)_f - u_f||_2$$

Error over the entire mesh

$$P = \sum_{f \in \mathcal{F}} P_f$$

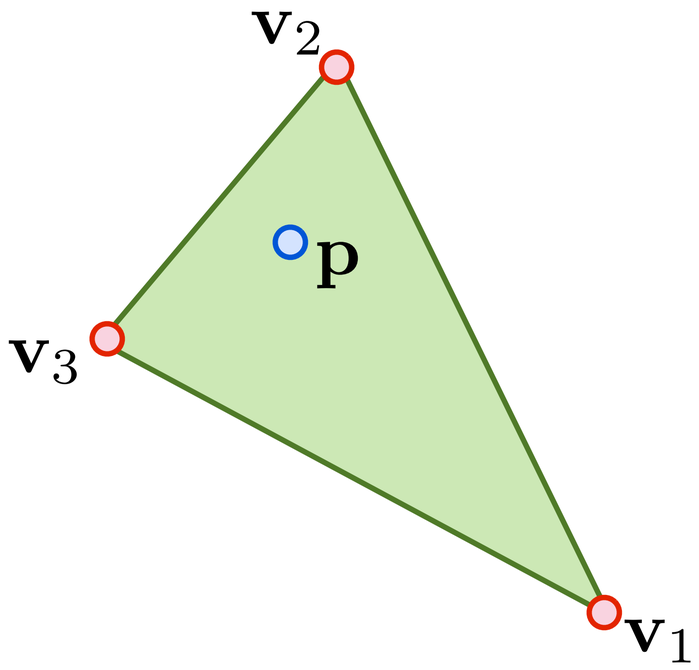


# Barycentric Coordinates




$$\mathbf{p} = w_1\mathbf{v}_1 + w_2\mathbf{v}_2 + w_3\mathbf{v}_3$$

# Barycentric Coordinates

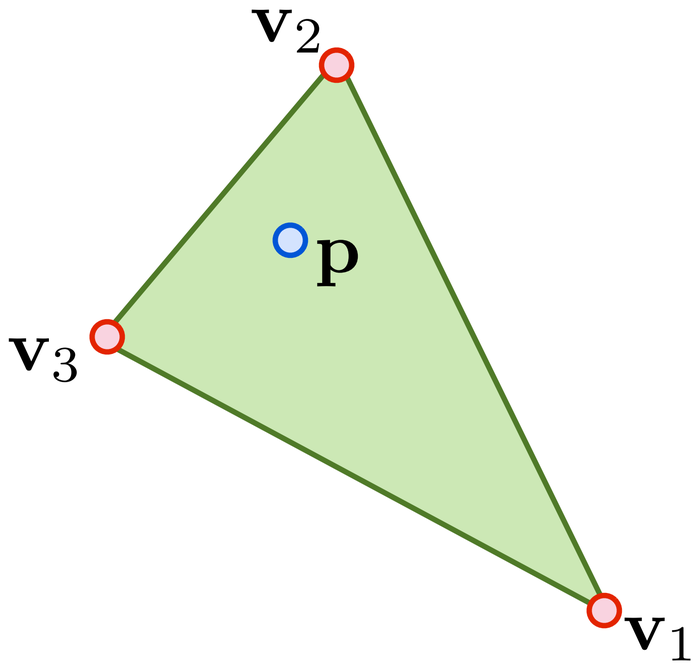



$$\mathbf{p} = w_1\mathbf{v}_1 + w_2\mathbf{v}_2 + w_3\mathbf{v}_3$$

Partition of unity: $w_1 + w_2 + w_3 = 1$

$$\mathbf{p} = w_1\mathbf{v}_1 + w_2\mathbf{v}_2 + (1 - w_1 - w_2)\mathbf{v}_3$$

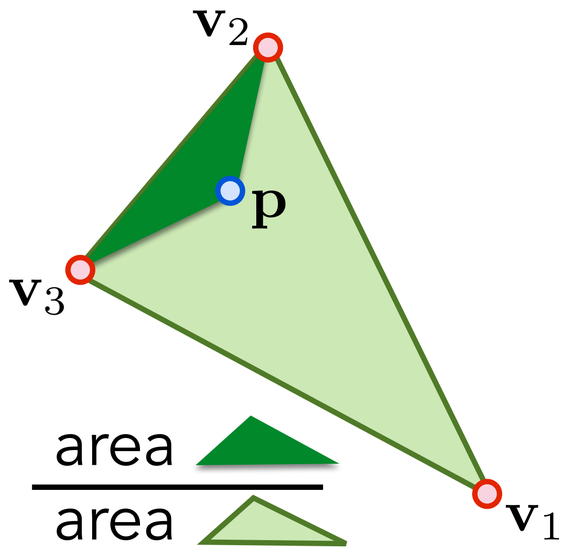
# Barycentric Coordinates



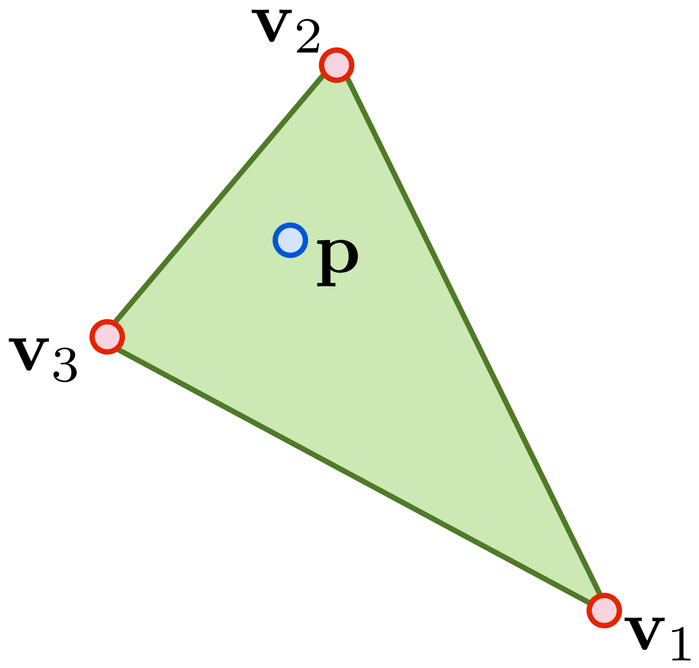

$$\mathbf{p} = w_1\mathbf{v}_1 + w_2\mathbf{v}_2 + w_3\mathbf{v}_3$$

Partition of unity: $w_1 + w_2 + w_3 = 1$

$$\mathbf{p} = w_1\mathbf{v}_1 + w_2\mathbf{v}_2 + (1 - w_1 - w_2)\mathbf{v}_3$$



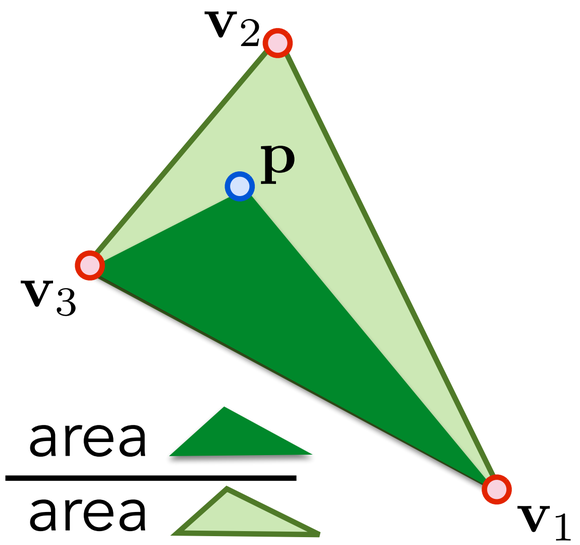
# Barycentric Coordinates




$$\mathbf{p} = w_1\mathbf{v}_1 + w_2\mathbf{v}_2 + w_3\mathbf{v}_3$$

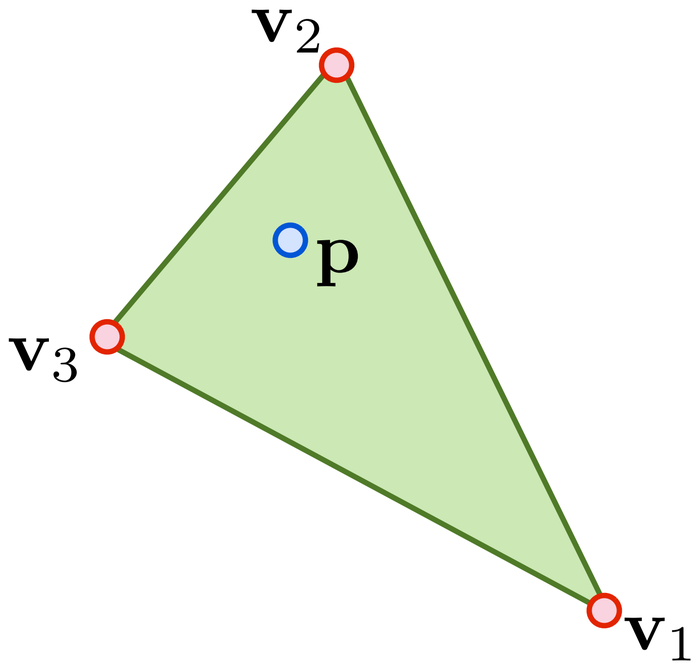
Partition of unity: $w_1 + w_2 + w_3 = 1$

$$\mathbf{p} = w_1\mathbf{v}_1 + w_2\mathbf{v}_2 + (1 - w_1 - w_2)\mathbf{v}_3$$




$$\frac{\text{area}}{\text{area}}$$

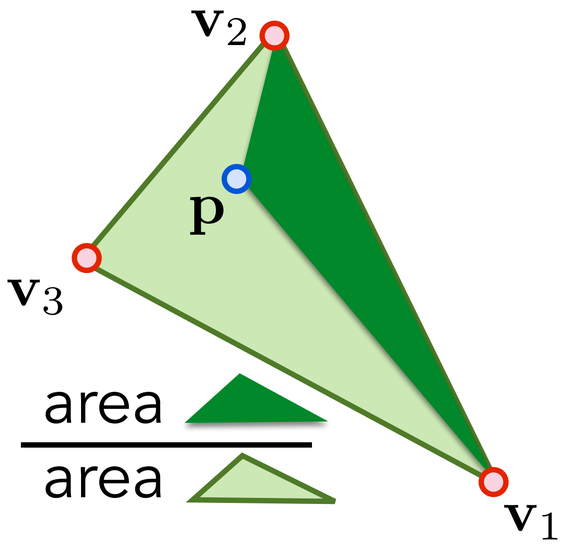
# Barycentric Coordinates




$$\mathbf{p} = w_1\mathbf{v}_1 + w_2\mathbf{v}_2 + w_3\mathbf{v}_3$$

Partition of unity: $w_1 + w_2 + w_3 = 1$

$$\mathbf{p} = w_1\mathbf{v}_1 + w_2\mathbf{v}_2 + (1 - w_1 - w_2)\mathbf{v}_3$$

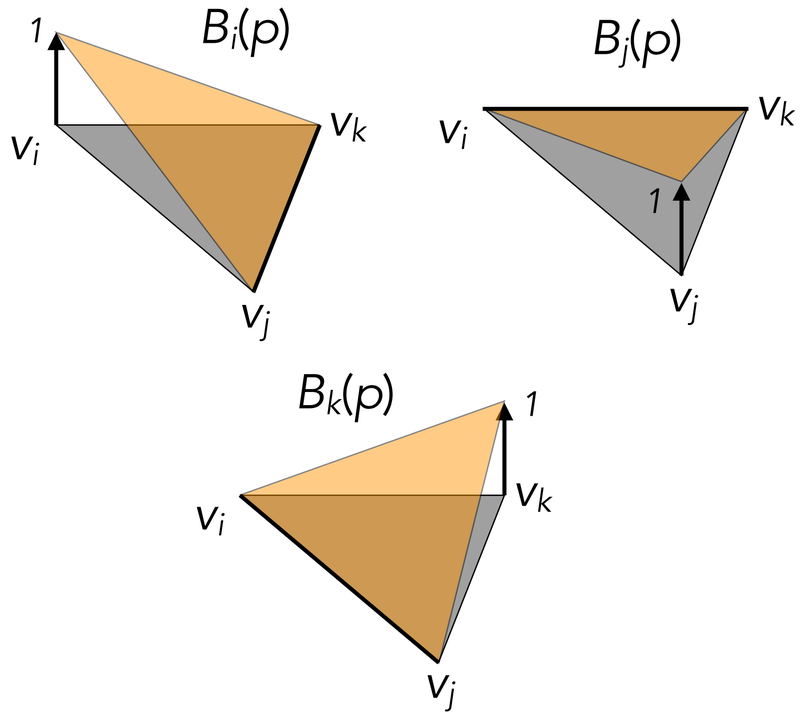


# Piecewise linear functions on meshes

Hat functions and PL interpolation

$$f(\mathbf{p}) = B_i(\mathbf{p})f_i + B_j(\mathbf{p})f_j + B_k(\mathbf{p})f_k$$

$$B_i(\mathbf{p}) + B_j(\mathbf{p}) + B_k(\mathbf{p}) = 1$$

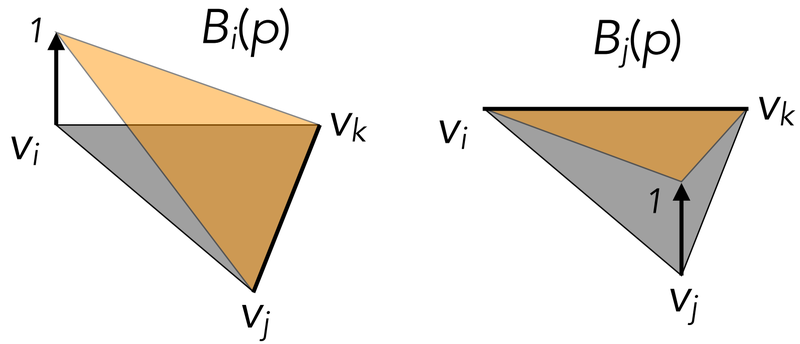


# Piecewise linear functions on meshes

Hat functions and PL interpolation

$$f(\mathbf{p}) = B_i(\mathbf{p})f_i + B_j(\mathbf{p})f_j + B_k(\mathbf{p})f_k$$

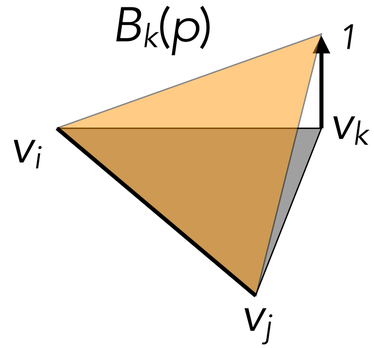
$$B_i(\mathbf{p}) + B_j(\mathbf{p}) + B_k(\mathbf{p}) = 1$$




Gradients

$$\nabla f(\mathbf{p}) = \nabla B_i(\mathbf{p})f_i + \nabla B_j(\mathbf{p})f_j + \nabla B_k(\mathbf{p})f_k$$

$$\nabla B_i(\mathbf{p}) + \nabla B_j(\mathbf{p}) + \nabla B_k(\mathbf{p}) = 0$$

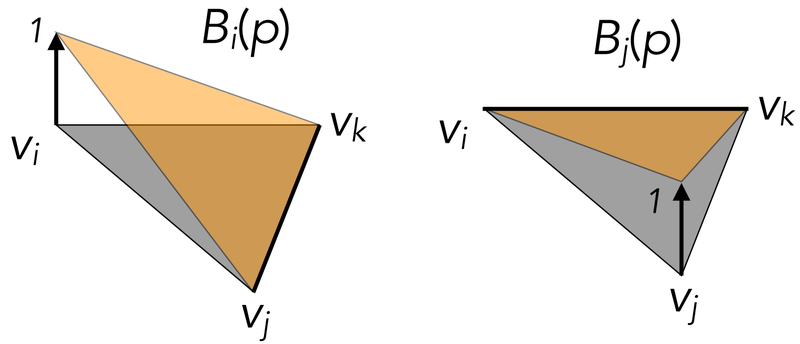


# Piecewise linear functions on meshes

Hat functions and PL interpolation

$$f(\mathbf{p}) = B_i(\mathbf{p})f_i + B_j(\mathbf{p})f_j + B_k(\mathbf{p})f_k$$

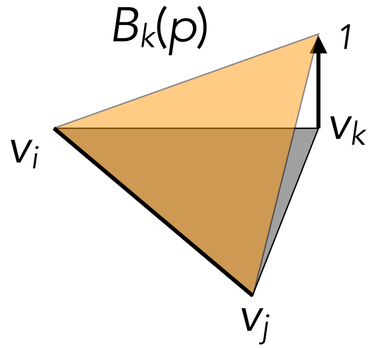
$$B_i(\mathbf{p}) + B_j(\mathbf{p}) + B_k(\mathbf{p}) = 1$$




Gradients

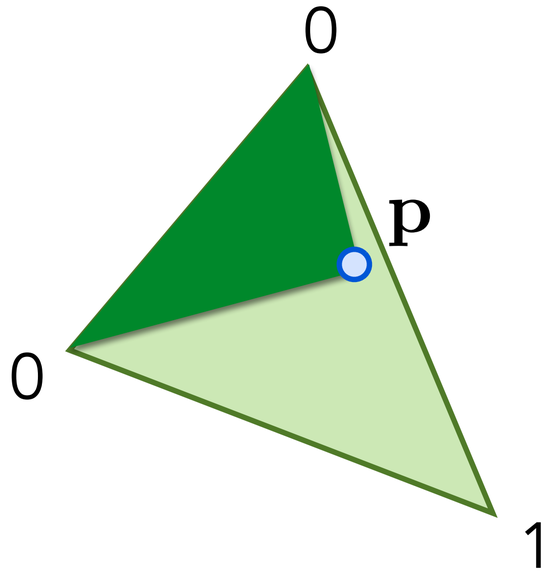
$$\nabla f(\mathbf{p}) = \nabla B_i(\mathbf{p})f_i + \nabla B_j(\mathbf{p})f_j + \nabla B_k(\mathbf{p})f_k$$

$$\nabla B_i(\mathbf{p}) + \nabla B_j(\mathbf{p}) + \nabla B_k(\mathbf{p}) = 0$$



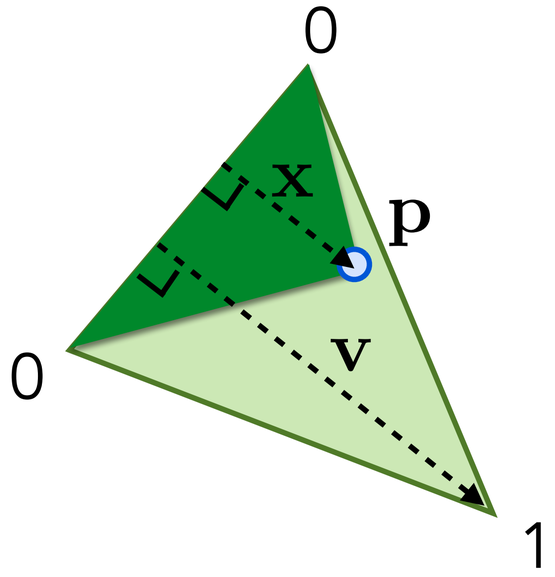

$$\nabla f(\mathbf{p}) = (f_j - f_i)\nabla B_j(\mathbf{p}) + (f_k - f_i)\nabla B_k(\mathbf{p})$$

# The hat function



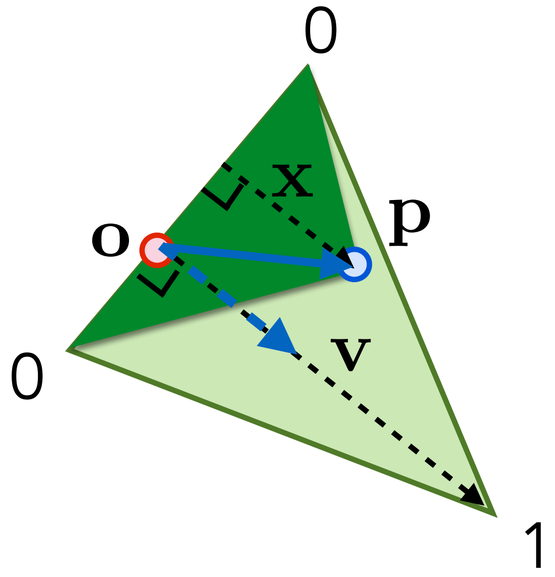

$$B(\mathbf{p}) = \frac{\text{area}}{\text{area}}$$

# The hat function



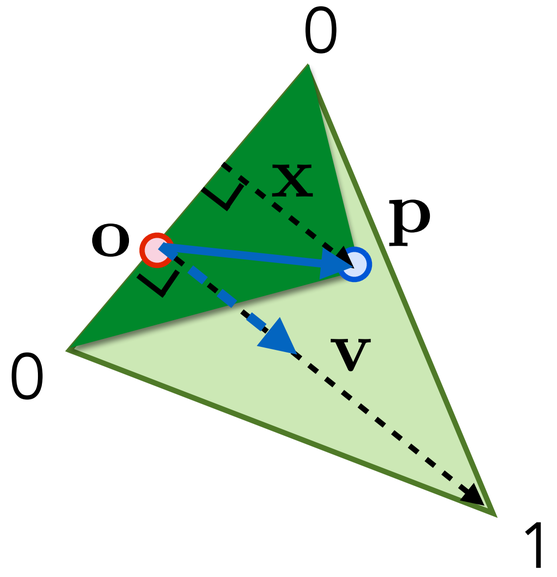

$$B(\mathbf{p}) = \frac{\text{area}}{\text{area}} = \frac{\|\mathbf{x}\|}{\|\mathbf{v}\|}$$

# The hat function



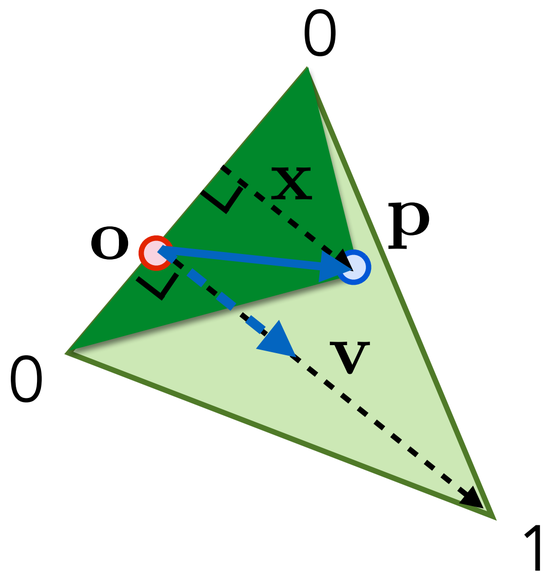

$$B(\mathbf{p}) = \frac{\text{area}}{\text{area}} = \frac{\|\mathbf{x}\|}{\|\mathbf{v}\|} = \frac{(\mathbf{p}-\mathbf{o}) \cdot \frac{\mathbf{v}}{\|\mathbf{v}\|}}{\|\mathbf{v}\|}$$

# The hat function



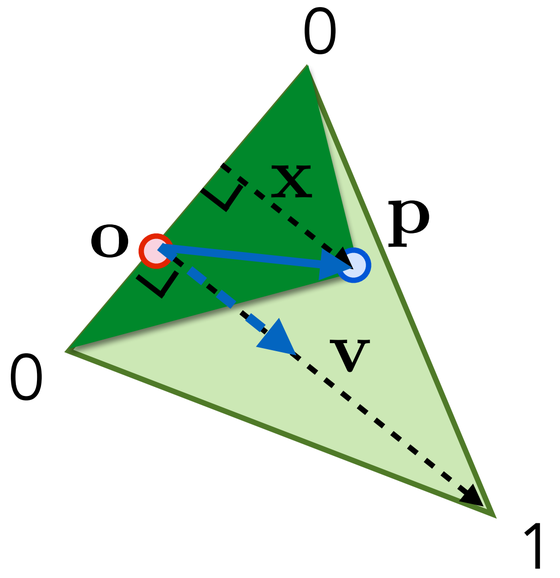

$$B(\mathbf{p}) = \frac{\text{area}}{\text{area}} = \frac{\|\mathbf{x}\|}{\|\mathbf{v}\|} = \frac{(\mathbf{p}-\mathbf{o})\cdot\frac{\mathbf{v}}{\|\mathbf{v}\|}}{\|\mathbf{v}\|} = \frac{(\mathbf{p}-\mathbf{o})\cdot\mathbf{v}}{\|\mathbf{v}\|\|\mathbf{v}\|}$$

# Gradient of the hat function




$$B(\mathbf{p}) = \frac{(\mathbf{p}-\mathbf{o})\cdot\mathbf{v}}{\|\mathbf{v}\|\|\mathbf{v}\|}$$

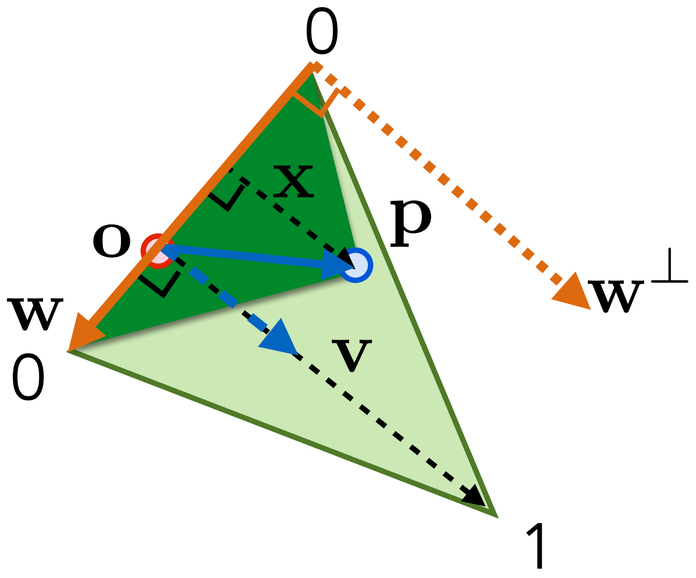
# Gradient of the hat function




$$\nabla B(\mathbf{p}) = \frac{\mathbf{v}}{\|\mathbf{v}\|\|\mathbf{v}\|}$$

$$B(\mathbf{p}) = \frac{(\mathbf{p}-\mathbf{o})\cdot\mathbf{v}}{\|\mathbf{v}\|\|\mathbf{v}\|}$$

# Gradient of the hat function

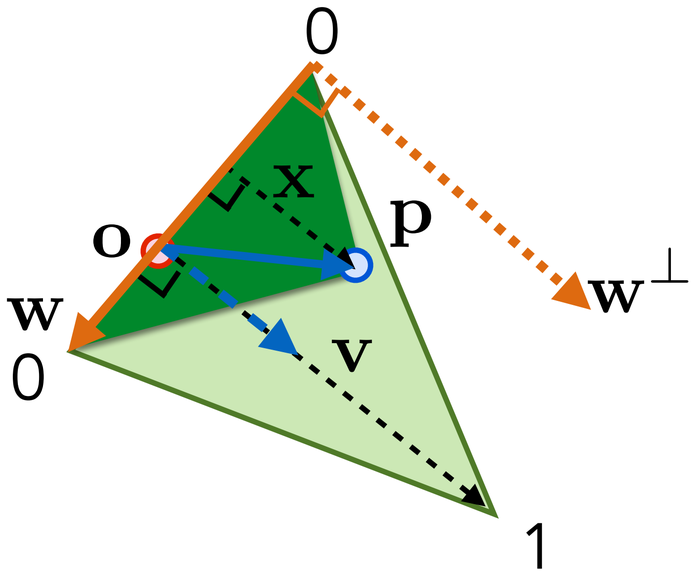



$$\nabla B(\mathbf{p}) = \frac{\mathbf{v}}{\|\mathbf{v}\|\|\mathbf{v}\|}$$

$$\frac{\mathbf{v}}{\|\mathbf{v}\|} = \frac{\mathbf{w}^\perp}{\|\mathbf{w}^\perp\|}$$

$$B(\mathbf{p}) = \frac{(\mathbf{p}-\mathbf{o})\cdot\mathbf{v}}{\|\mathbf{v}\|\|\mathbf{v}\|}$$

# Gradient of the hat function



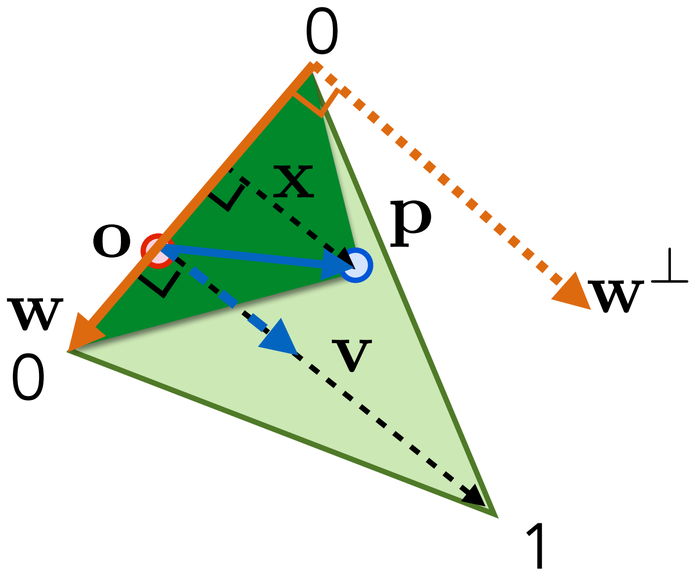

$$\nabla B(\mathbf{p}) = \frac{\mathbf{v}}{\|\mathbf{v}\|\|\mathbf{v}\|}$$

$$\nabla B(\mathbf{p}) = \frac{\mathbf{w}^\perp}{\|\mathbf{w}^\perp\|\|\mathbf{v}\|}$$

$$\frac{\mathbf{v}}{\|\mathbf{v}\|} = \frac{\mathbf{w}^\perp}{\|\mathbf{w}^\perp\|}$$

$$B(\mathbf{p}) = \frac{(\mathbf{p}-\mathbf{o})\cdot\mathbf{v}}{\|\mathbf{v}\|\|\mathbf{v}\|}$$

# Gradient of the hat function




$$\nabla B(\mathbf{p}) = \frac{\mathbf{v}}{\|\mathbf{v}\|\|\mathbf{v}\|}$$

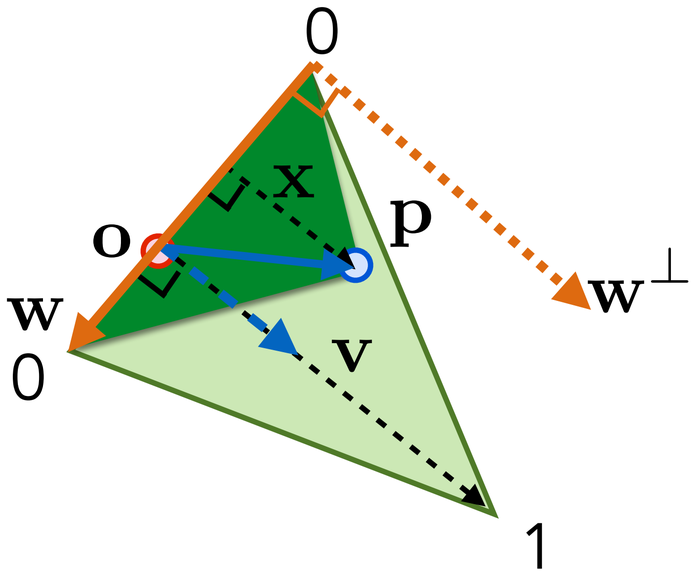
$$\nabla B(\mathbf{p}) = \frac{\mathbf{w}^\perp}{\|\mathbf{w}^\perp\|\|\mathbf{v}\|}$$

$$\frac{\mathbf{v}}{\|\mathbf{v}\|} = \frac{\mathbf{w}^\perp}{\|\mathbf{w}^\perp\|}$$

$$\|\mathbf{w}^\perp\| = \|\mathbf{w}\|$$

$$B(\mathbf{p}) = \frac{(\mathbf{p}-\mathbf{o})\cdot\mathbf{v}}{\|\mathbf{v}\|\|\mathbf{v}\|}$$

# Gradient of the hat function




$$B(\mathbf{p}) = \frac{(\mathbf{p}-\mathbf{o})\cdot\mathbf{v}}{\|\mathbf{v}\|\|\mathbf{v}\|}$$

$$\nabla B(\mathbf{p}) = \frac{\mathbf{v}}{\|\mathbf{v}\|\|\mathbf{v}\|}$$

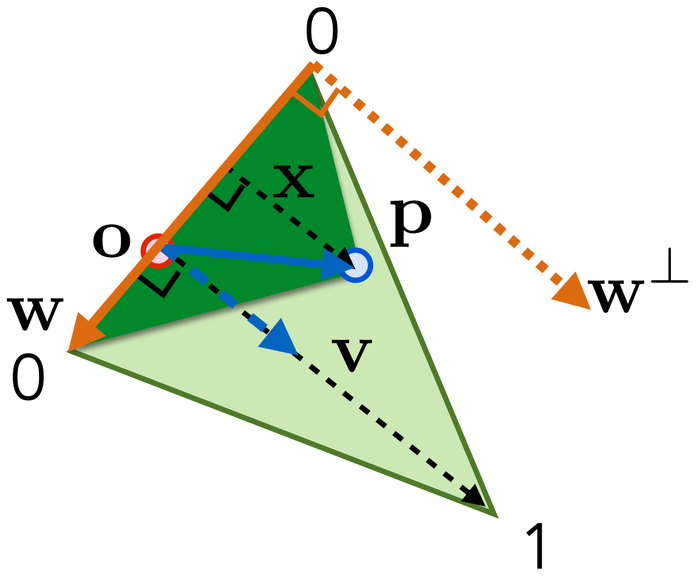
$$\nabla B(\mathbf{p}) = \frac{\mathbf{w}^\perp}{\|\mathbf{w}^\perp\|\|\mathbf{v}\|}$$

$$\nabla B(\mathbf{p}) = \frac{\mathbf{w}^\perp}{\|\mathbf{w}\|\|\mathbf{v}\|}$$

$$\frac{\mathbf{v}}{\|\mathbf{v}\|} = \frac{\mathbf{w}^\perp}{\|\mathbf{w}^\perp\|}$$

$$\|\mathbf{w}^\perp\| = \|\mathbf{w}\|$$

# Gradient of the hat function




$$B(\mathbf{p}) = \frac{(\mathbf{p}-\mathbf{o})\cdot\mathbf{v}}{\|\mathbf{v}\|\|\mathbf{v}\|}$$

$$\nabla B(\mathbf{p}) = \frac{\mathbf{v}}{\|\mathbf{v}\|\|\mathbf{v}\|}$$

$$\nabla B(\mathbf{p}) = \frac{\mathbf{w}^\perp}{\|\mathbf{w}^\perp\|\|\mathbf{v}\|}$$

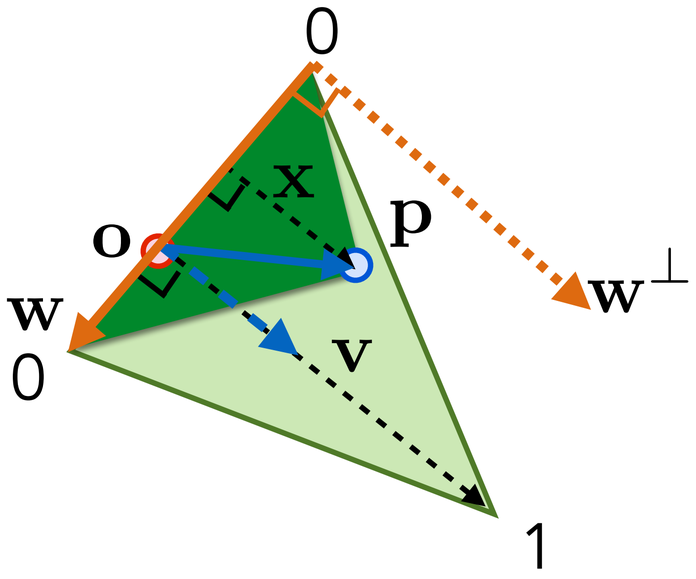
$$\nabla B(\mathbf{p}) = \frac{\mathbf{w}^\perp}{\|\mathbf{w}\|\|\mathbf{v}\|}$$

$$\frac{\mathbf{v}}{\|\mathbf{v}\|} = \frac{\mathbf{w}^\perp}{\|\mathbf{w}^\perp\|}$$

$$\|\mathbf{w}^\perp\| = \|\mathbf{w}\|$$

$$A = \frac{\|\mathbf{v}\|\|\mathbf{w}\|}{2}$$

# Gradient of the hat function




$$B(\mathbf{p}) = \frac{(\mathbf{p}-\mathbf{o})\cdot\mathbf{v}}{\|\mathbf{v}\|\|\mathbf{v}\|}$$

$$\nabla B(\mathbf{p}) = \frac{\mathbf{v}}{\|\mathbf{v}\|\|\mathbf{v}\|}$$

$$\nabla B(\mathbf{p}) = \frac{\mathbf{w}^\perp}{\|\mathbf{w}^\perp\|\|\mathbf{v}\|}$$

$$\nabla B(\mathbf{p}) = \frac{\mathbf{w}^\perp}{\|\mathbf{w}\|\|\mathbf{v}\|}$$

$$\nabla B(\mathbf{p}) = \frac{\mathbf{w}^\perp}{2A}$$

$$\frac{\mathbf{v}}{\|\mathbf{v}\|} = \frac{\mathbf{w}^\perp}{\|\mathbf{w}^\perp\|}$$

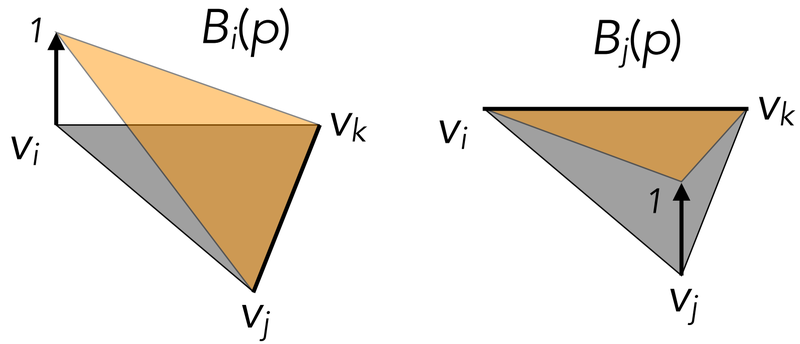
$$\|\mathbf{w}^\perp\| = \|\mathbf{w}\|$$

$$A = \frac{\|\mathbf{v}\|\|\mathbf{w}\|}{2}$$

# Piecewise linear functions on meshes

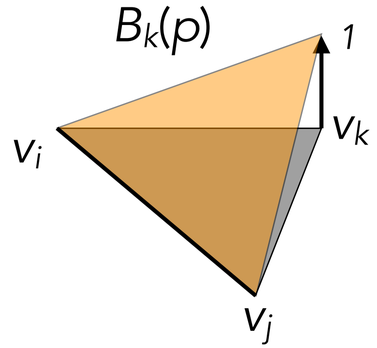
Hat functions and PL interpolation

$$f(\mathbf{p}) = B_i(\mathbf{p})f_i + B_j(\mathbf{p})f_j + B_k(\mathbf{p})f_k$$




Gradients

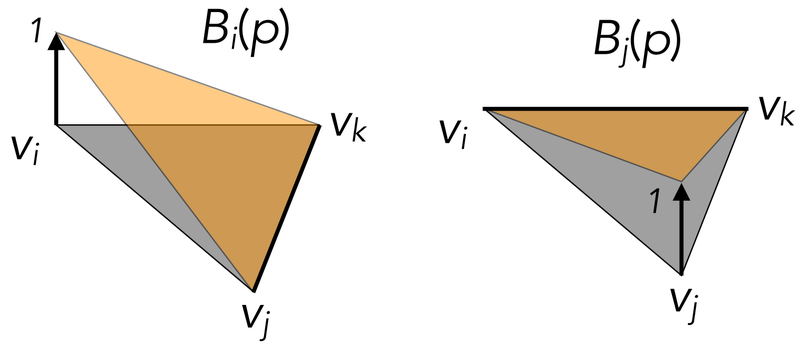
$$\nabla f(\mathbf{p}) = (f_j - f_i)\nabla B_j(\mathbf{p}) + (f_k - f_i)\nabla B_k(\mathbf{p})$$



# Piecewise linear functions on meshes

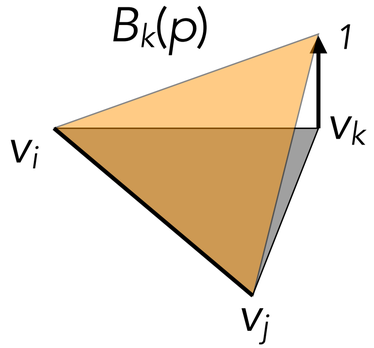
Hat functions and PL interpolation

$$f(\mathbf{p}) = B_i(\mathbf{p})f_i + B_j(\mathbf{p})f_j + B_k(\mathbf{p})f_k$$




Gradients

$$\nabla f(\mathbf{p}) = (f_j - f_i)\nabla B_j(\mathbf{p}) + (f_k - f_i)\nabla B_k(\mathbf{p})$$




$$\nabla f(\mathbf{p}) = (f_j - f_i)\frac{(\mathbf{v_i} - \mathbf{v_k})^\perp}{2A} + (f_k - f_i)\frac{(\mathbf{v_j} - \mathbf{v_i})^\perp}{2A}$$

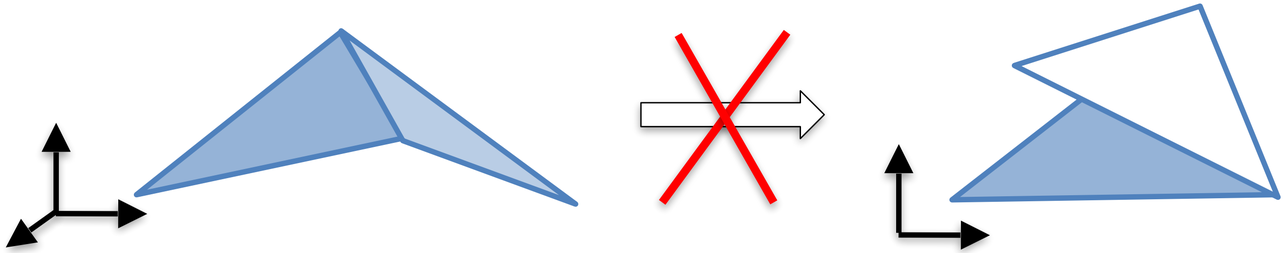
# Properties

# Desiderata?

- What are “good” parameterizations?
- How do we define “good”?

# Bijectivity

- Locally bijective (1-1 and onto): No triangles fold over.



- Globally bijective: locally bijective + no “distant” areas overlap

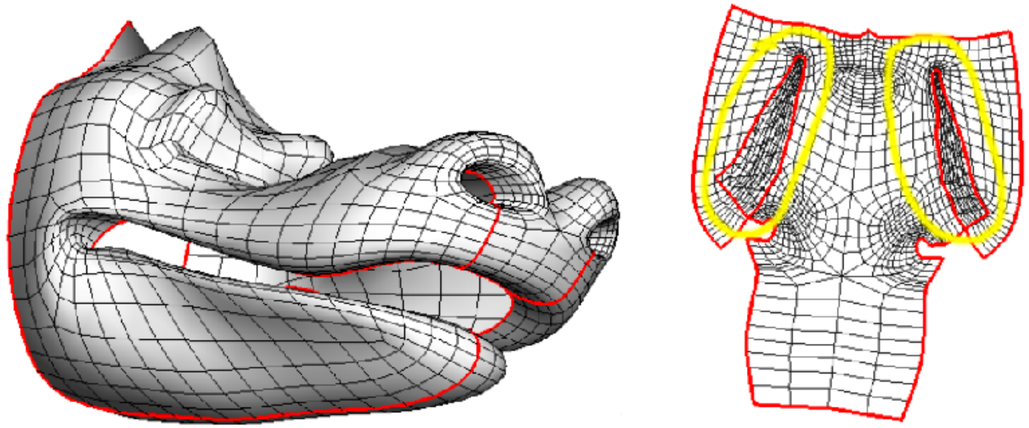
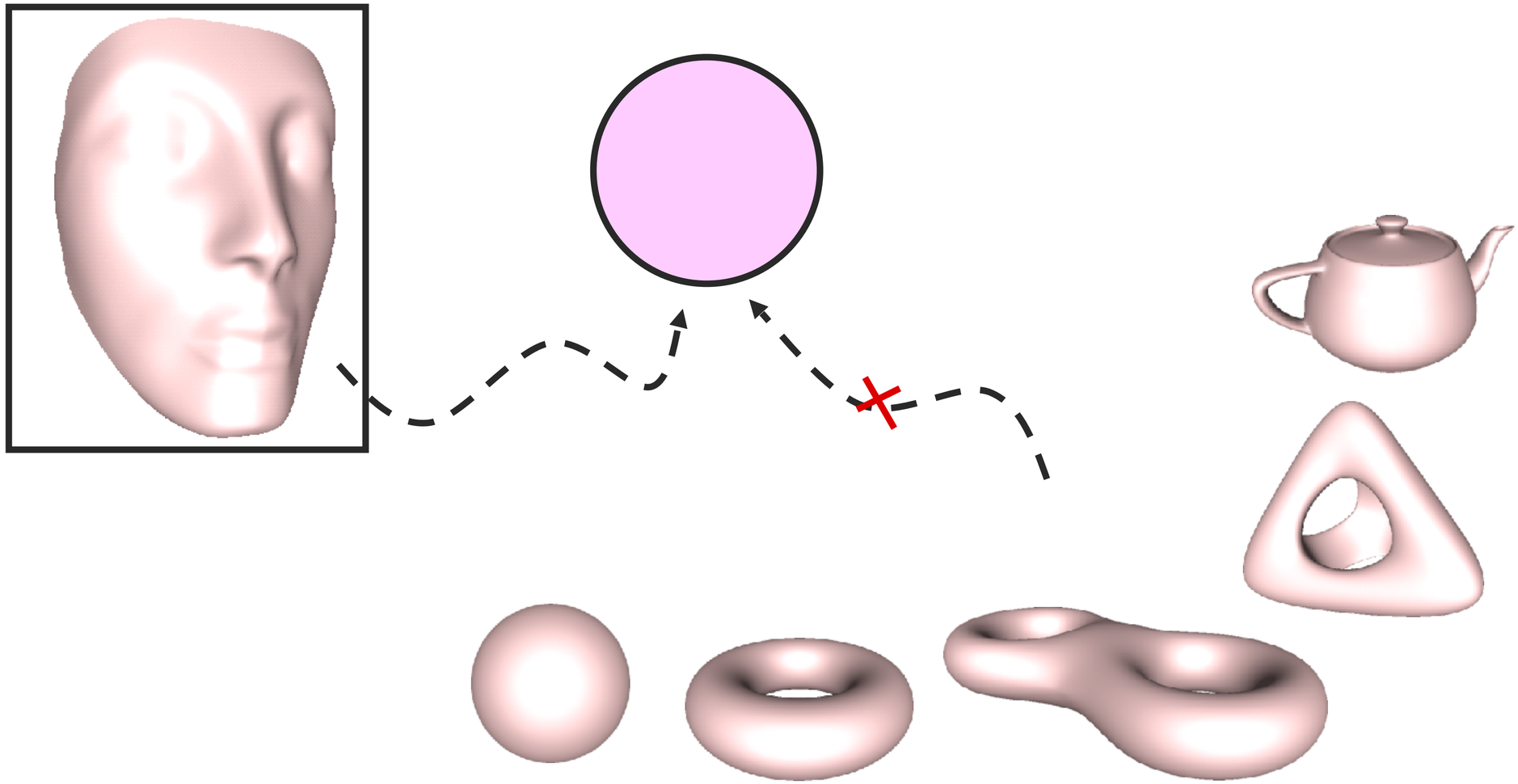
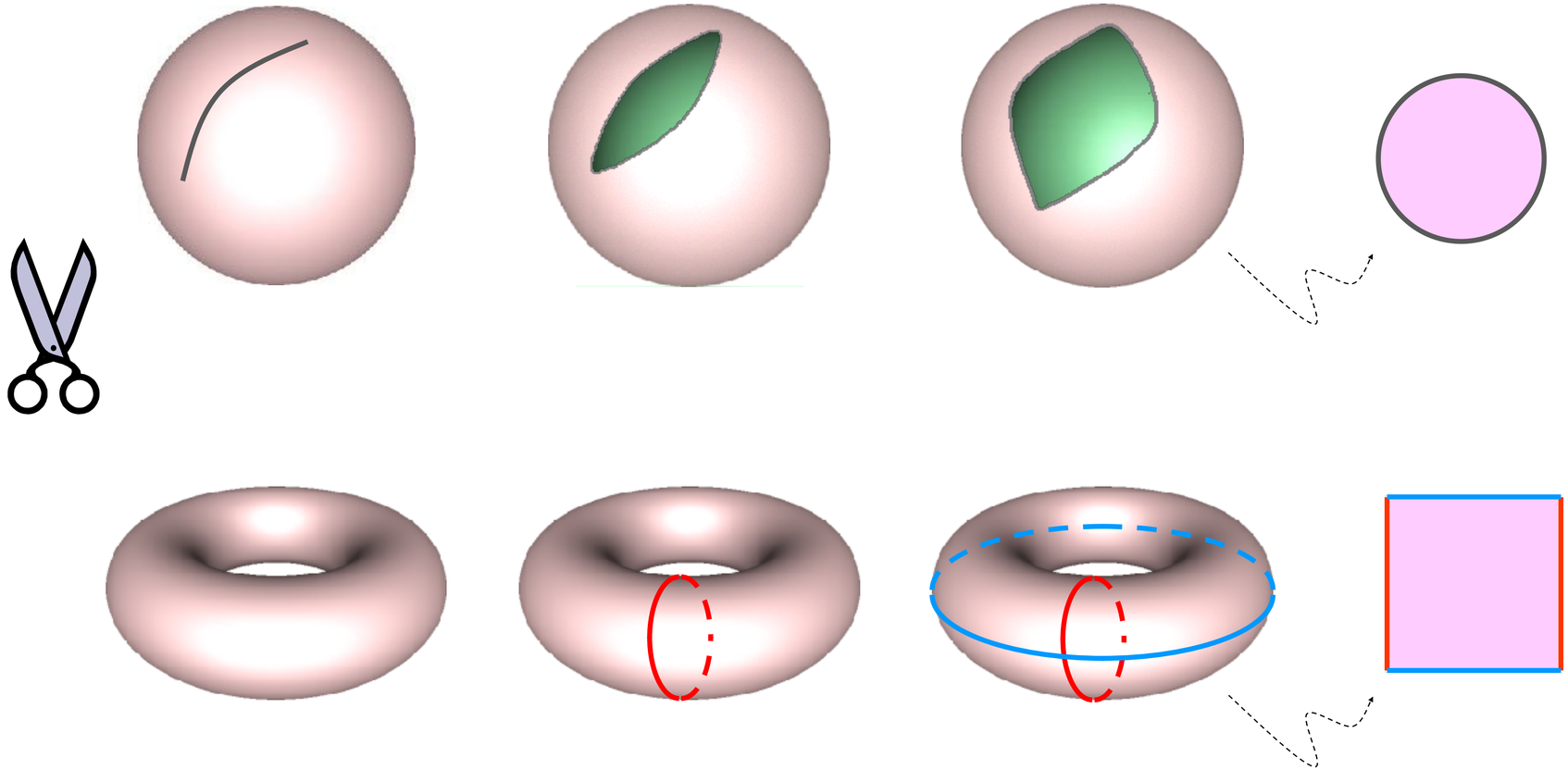


image from “Least Squares Conformal Maps”, Lévy et al., SIGGRAPH 2002

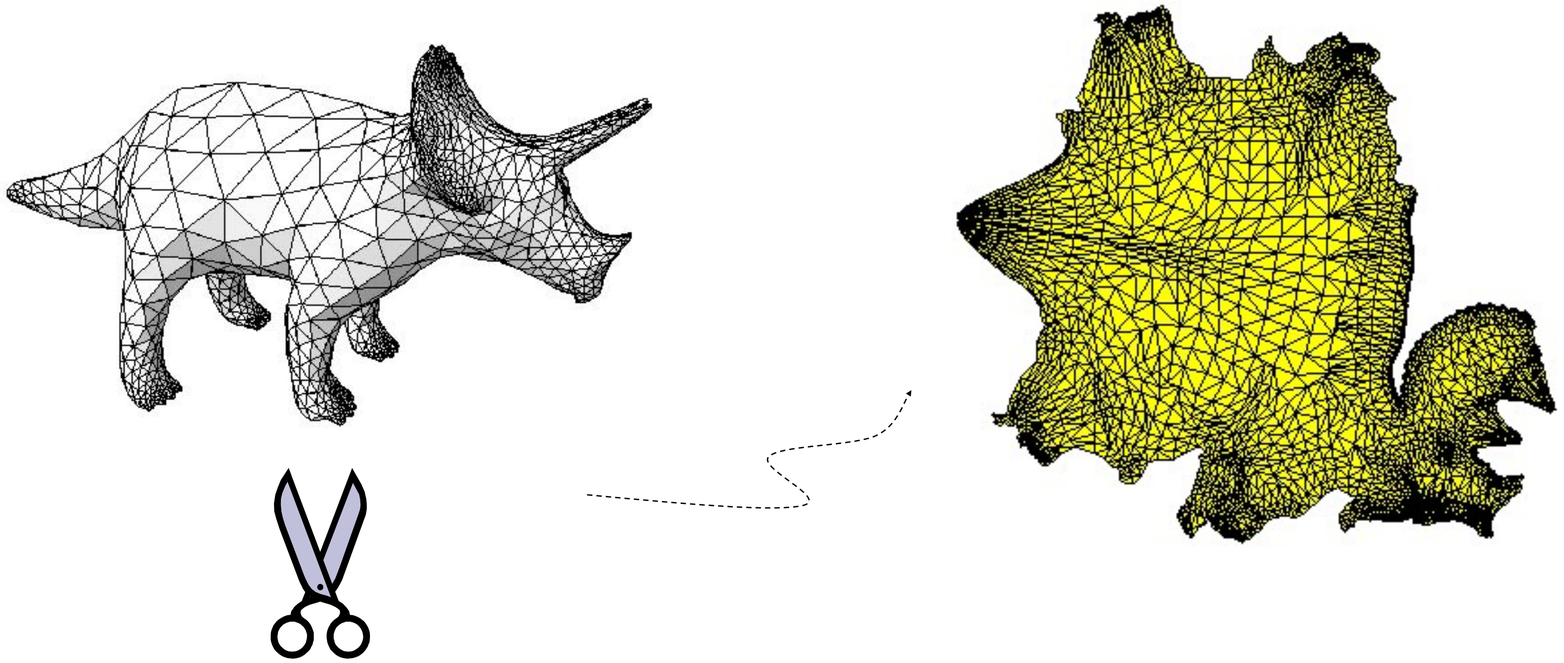
# Bijectivity: Non-Disk Domains



# Topological Cutting

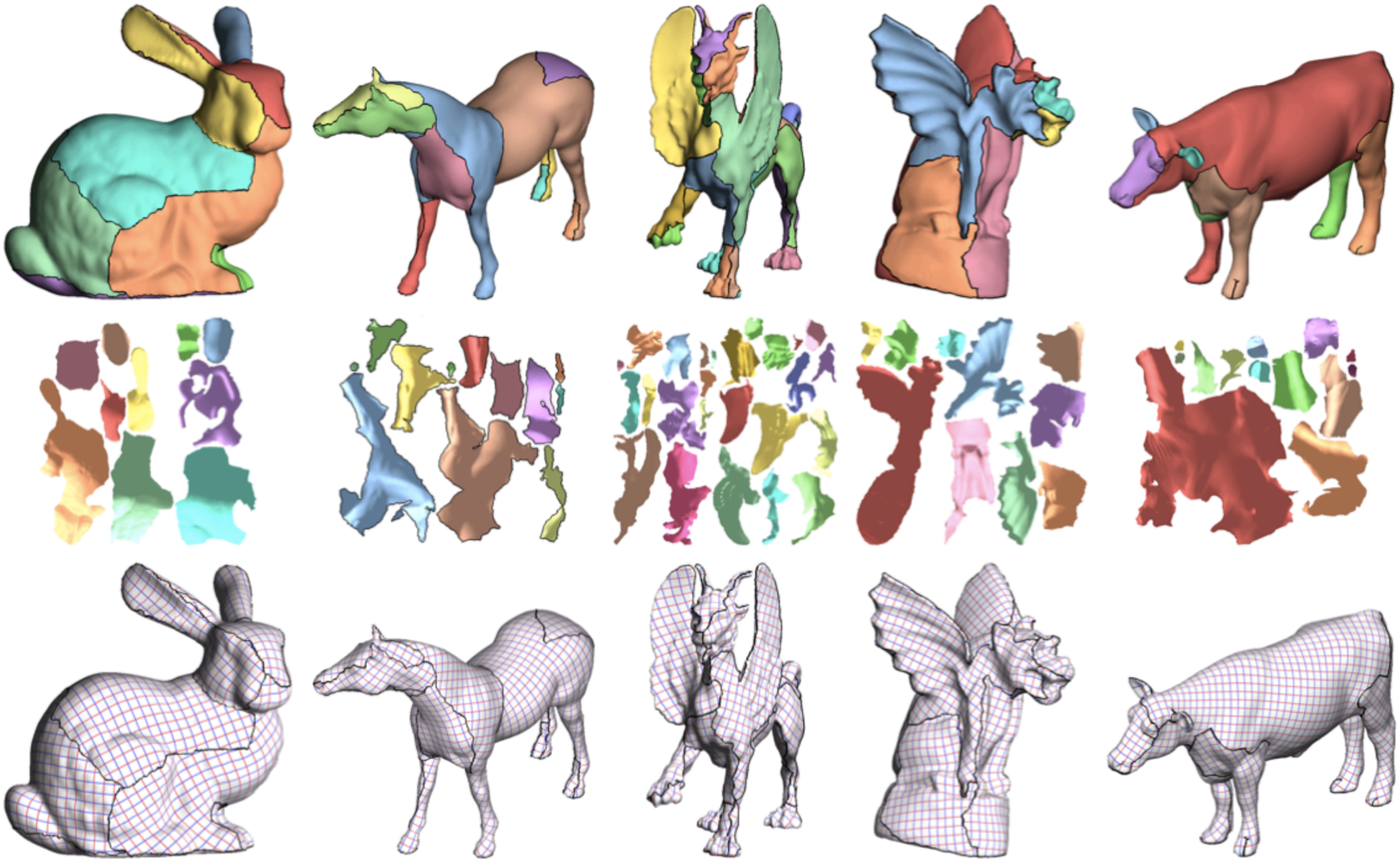


# Topological Cutting



A. Sheffer, J. Hart:
**Seamster: Inconspicuous Low-Distortion Texture Seam Layout**, IEEE Vis 2002
http://www.cs.ubc.ca/~sheffa/papers/VIS02.pdf

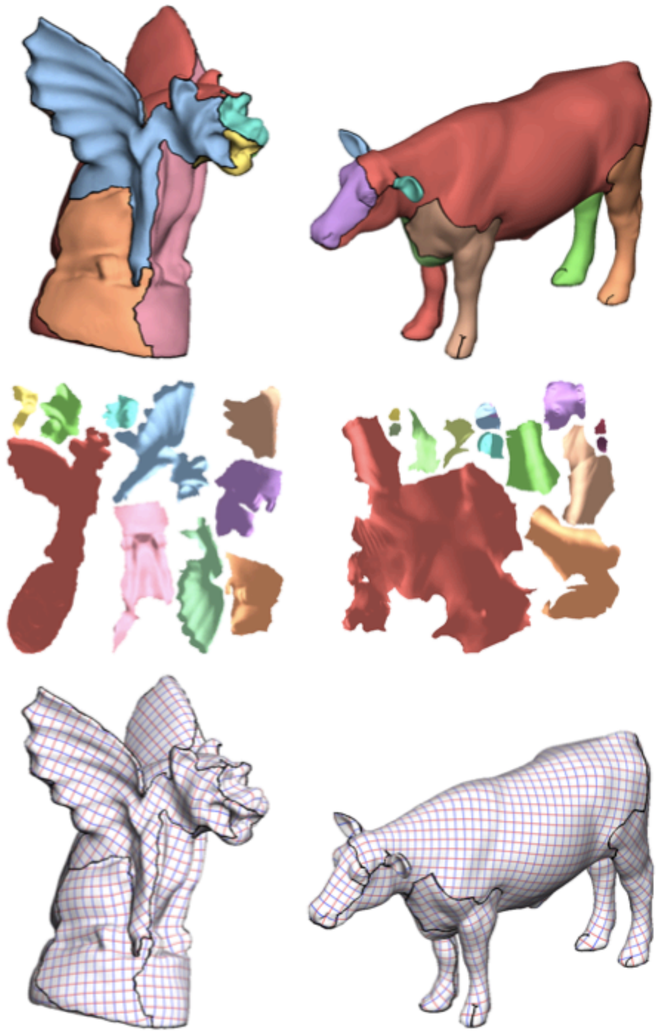
# Segmentation



# Segmentation

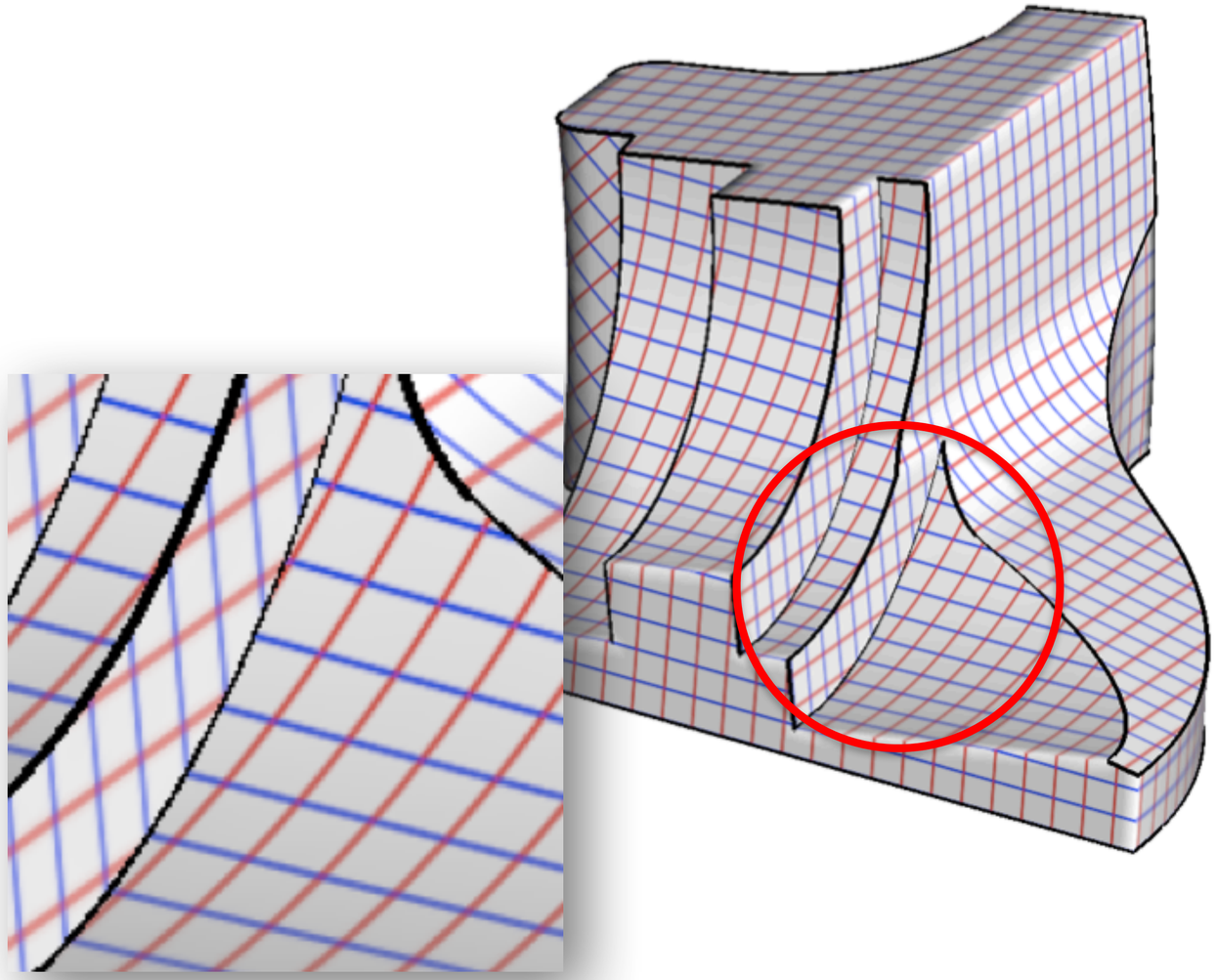
- D-Charts: Quasi-Developable Mesh Segmentation,
  D. Julius,V. Kraevoy, A. Sheffer,
  EUROGRAPHICS 2005

- Find patches that align to mesh features and are close to being developable surfaces



# Good Cuts/Segmentations?

- Hide seams
- Small number/length of seams
- In quad meshing, we want the parameterization derivatives to be continuous across seams!

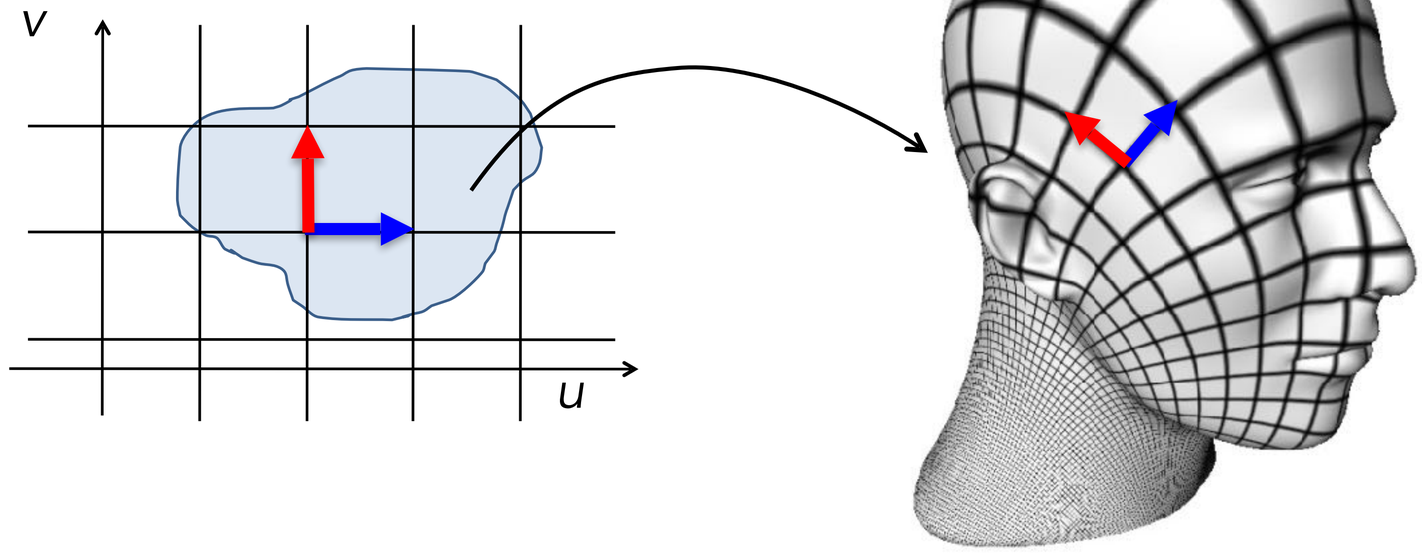


# How to Measure Distortion?

# Measures of Local Distortion

$$\mathbf{p}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}, \ (u,v) \in \mathbb{R}^2$$

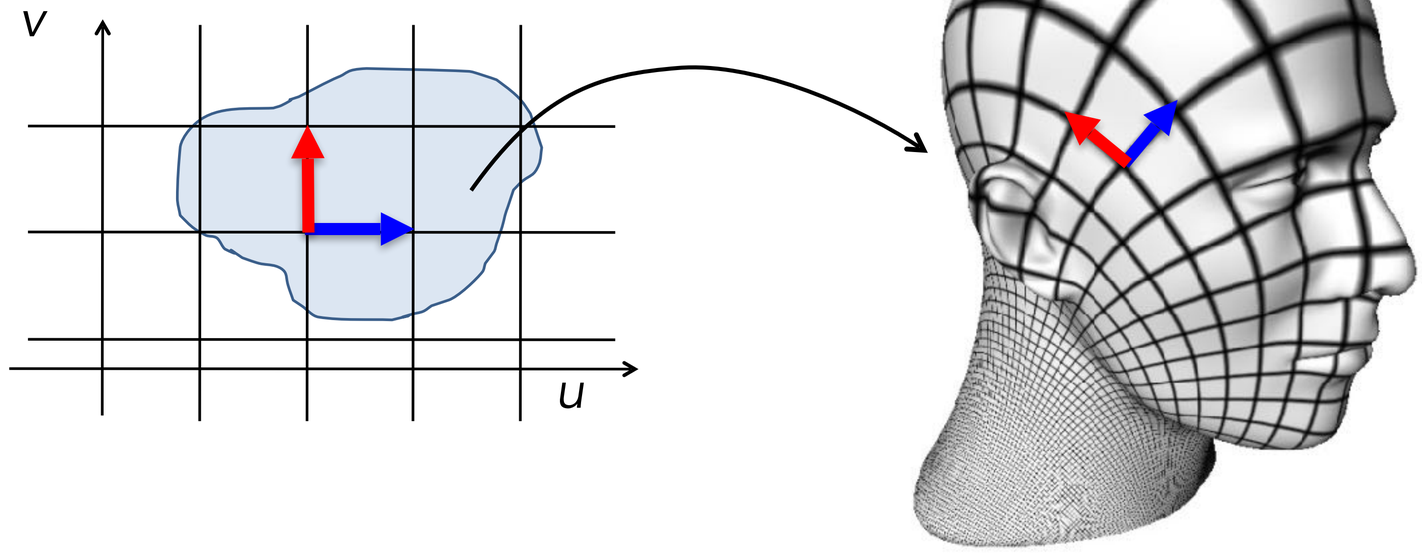
What happens to tangent vectors?



# Measures of Local Distortion

$$\mathbf{p}_u = \frac{\partial \mathbf{p}(u,v)}{\partial u}, \quad \mathbf{p}_v = \frac{\partial \mathbf{p}(u,v)}{\partial v}$$

What happens to tangent vectors?

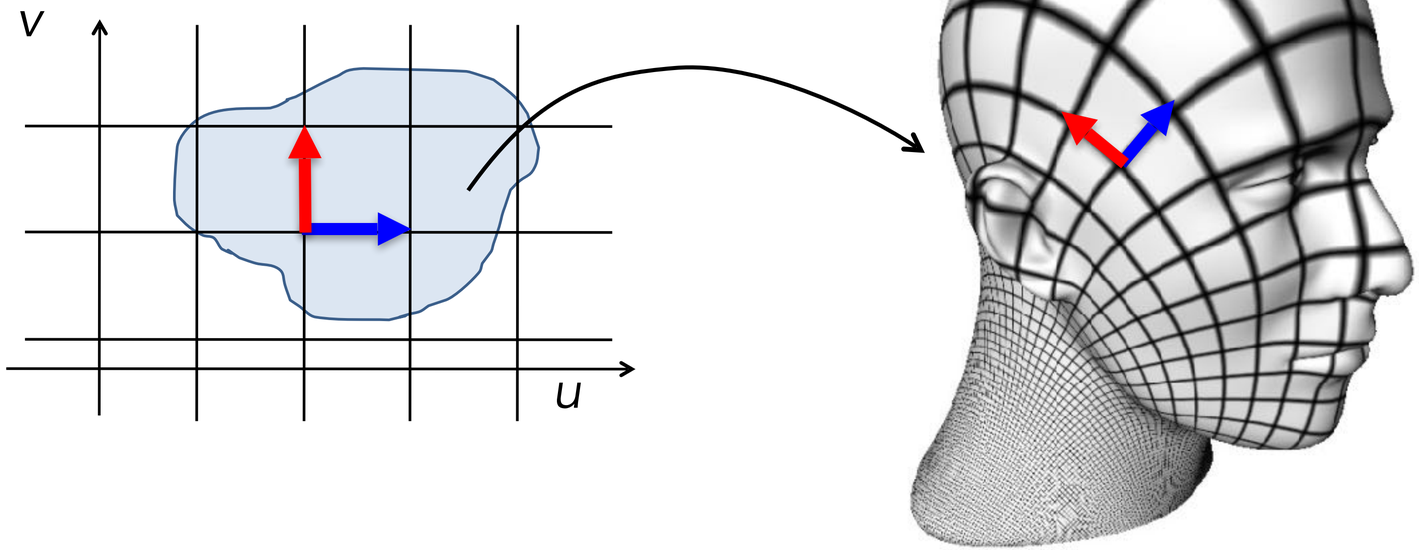


# Measures of Local Distortion

- How do lengths and angles of tangents change?
  - First fundamental form!

$$\mathbf{I} = \begin{pmatrix} E & F \\ F & G \end{pmatrix} = \begin{pmatrix} \mathbf{p}_u^T \mathbf{p}_u & \mathbf{p}_u^T \mathbf{p}_v \\ \mathbf{p}_u^T \mathbf{p}_v & \mathbf{p}_v^T \mathbf{p}_v \end{pmatrix}$$

What happens to tangent vectors?



# Measures of Local Distortion

- How do lengths and angles of tangents change?
  - First fundamental form!

$$\mathbf{I} = \begin{pmatrix} E & F \\ F & G \end{pmatrix} = \begin{pmatrix} \mathbf{p}_u^T \mathbf{p}_u & \mathbf{p}_u^T \mathbf{p}_v \\ \mathbf{p}_u^T \mathbf{p}_v & \mathbf{p}_v^T \mathbf{p}_v \end{pmatrix}$$

Angle change

Length change

  - Area distortion: area element:

$$dA = \sqrt{EG - F^2}\, dudv$$

# Measures of Local Distortion

- How do lengths and angles of tangents change?
  - First fundamental form!

$$\mathbf{I} = \begin{pmatrix} E & F \\ F & G \end{pmatrix} = \begin{pmatrix} \mathbf{p}_u^T \mathbf{p}_u & \mathbf{p}_u^T \mathbf{p}_v \\ \mathbf{p}_u^T \mathbf{p}_v & \mathbf{p}_v^T \mathbf{p}_v \end{pmatrix}$$

  - The eigenvalues of $\mathbf{I}$ tell us the maximal/minimal stretching of a tangent vector

# Distortion on Triangle Meshes?

- Triangle in 3D is mapped to triangle in 2D
- Unique affine mapping


$$P : \mathbb{R}^2 \rightarrow \mathbb{R}^3, \quad P(\mathbf{u}) = A\mathbf{u} + \mathbf{c}$$

$$[P_u \; P_v] = A$$ ← Jacobian

# Distortion on Triangle Meshes?

- SVD of the Jacobian reveals directions of extreme stretching

$$A = U \begin{pmatrix} \Gamma & \\ & \sigma \end{pmatrix} V^T$$


$$P : \mathbb{R}^2 \to \mathbb{R}^3, \quad P(\mathbf{u}) = A\mathbf{u} + \mathbf{c}$$

$$[P_u \; P_v] = A \Longrightarrow \mathbf{I} = A^T A$$

# Distortion on Triangle Meshes?

- Possible distortion measures:

$$A = U \begin{pmatrix} \Gamma & \\ & \sigma \end{pmatrix} V^T$$

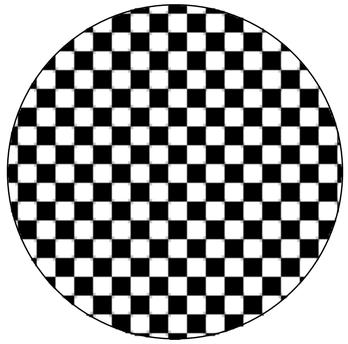
$$E(T) = \sqrt{\Gamma^2 + \sigma^2} \qquad E(T) = \max\left\{\Gamma, \frac{1}{\sigma}\right\}$$


$$P : \mathbb{R}^2 \to \mathbb{R}^3, \quad P(\mathbf{u}) = A\mathbf{u} + \mathbf{c}$$

$$[P_u \; P_v] = A$$

# How To Compute Parameterizations?

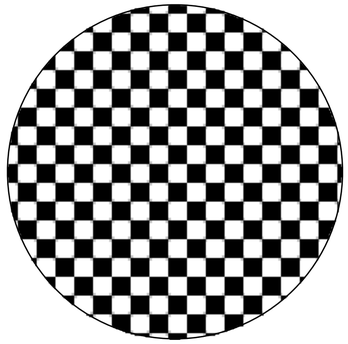
# Distortion Minimization



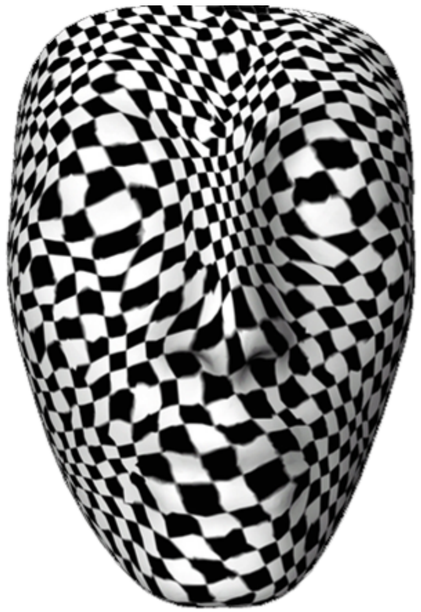
Texture map

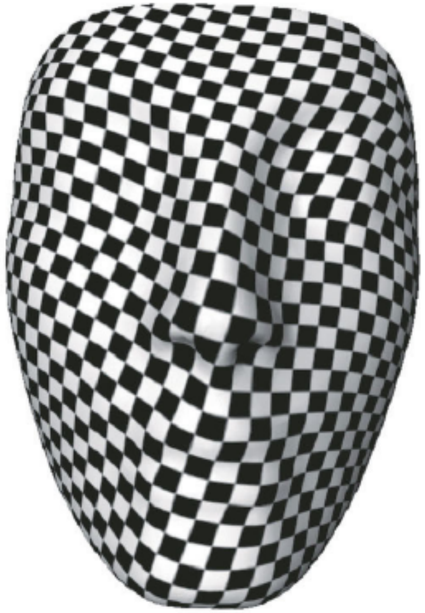
$$\underset{(u_1,v_1),\ldots,(u_n,v_n)}{\operatorname{argmin}} \sum_T E(T)$$

# Distortion Minimization

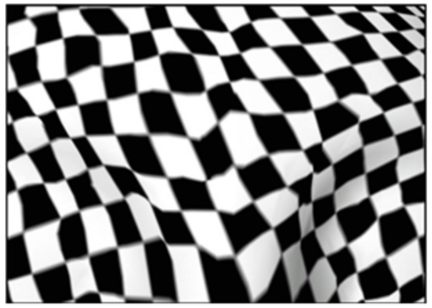


Texture map

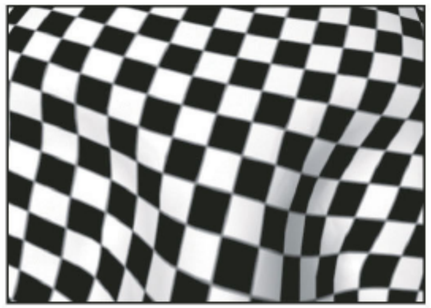




Kent et al ‘92



Floater 97

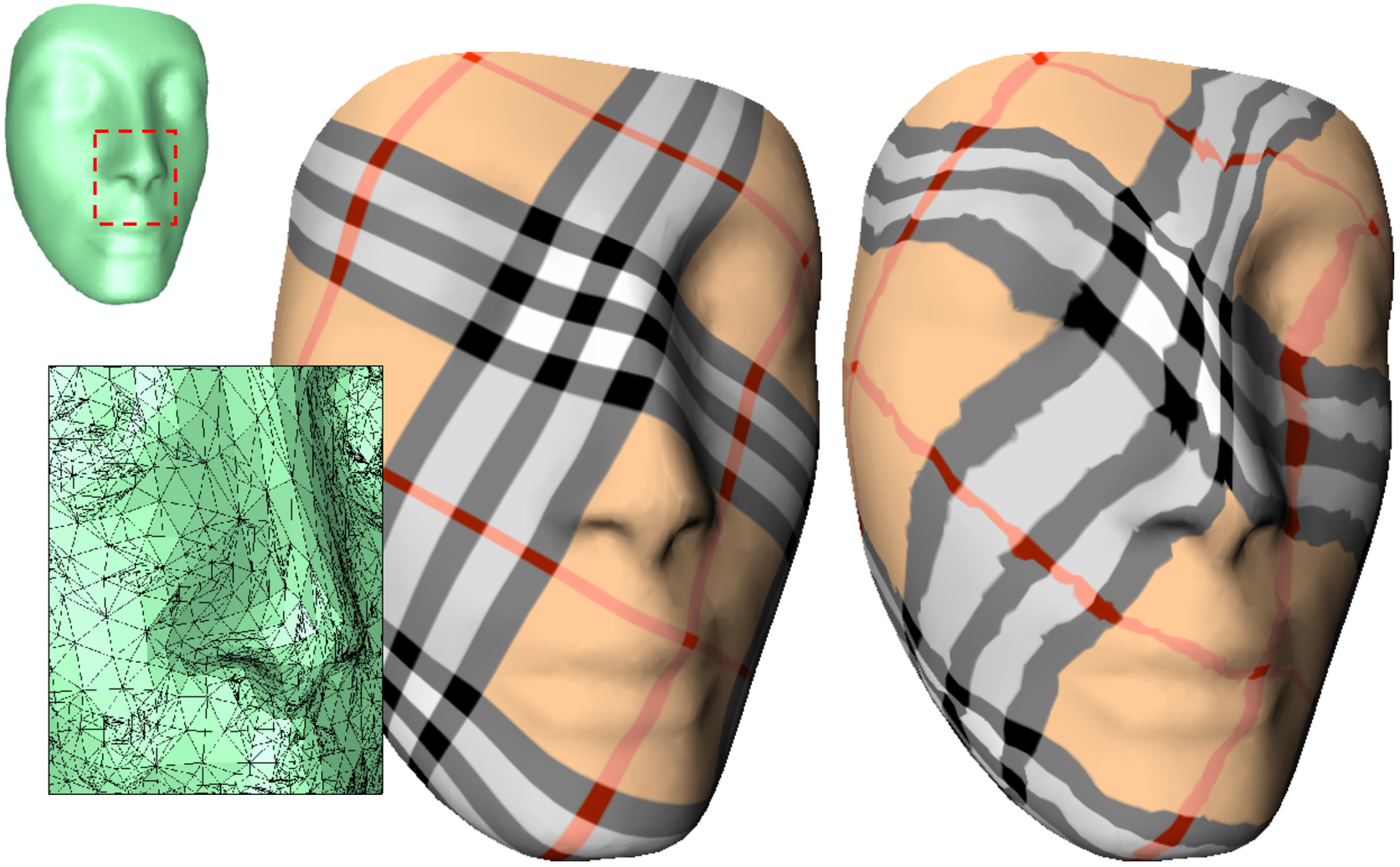


Sander et al ‘01

# Literature for previous slide

- Kent et al. ‘92, “Shape transformation for polyhedral objects”, Computer Graphics Vol. 26(2), 47-54
  http://ijcc.org/ojs/index.php/ijcc/article/viewFile/87/78
- Floater ’97, “Parametrization and smooth approximation of surface triangulations”, Computer Aided Geometric Design Vol. 14 (1997), 231-250.
  http://www.mn.uio.no/math/english/people/aca/michaelf/papers/param.pdf
- Sander et al. ‘01, “Texture mapping progressing meshes”, SIGGRAPH 2001.
  http://research.microsoft.com/en-us/um/people/hoppe/proj/tmpm/

# Mesh Dependence

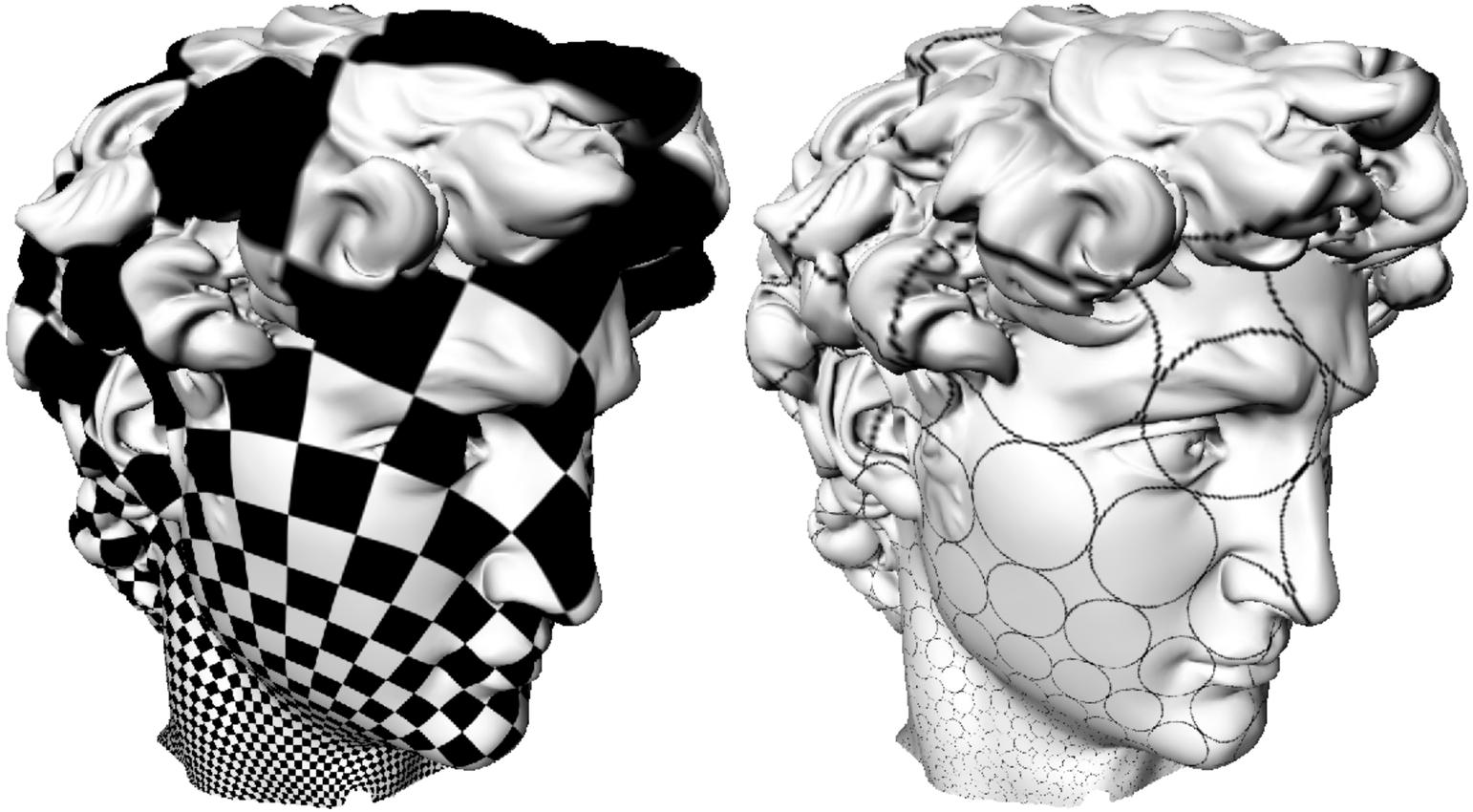


Alg. 1 – mesh-independent

Alg. 2 – … less mesh-independent

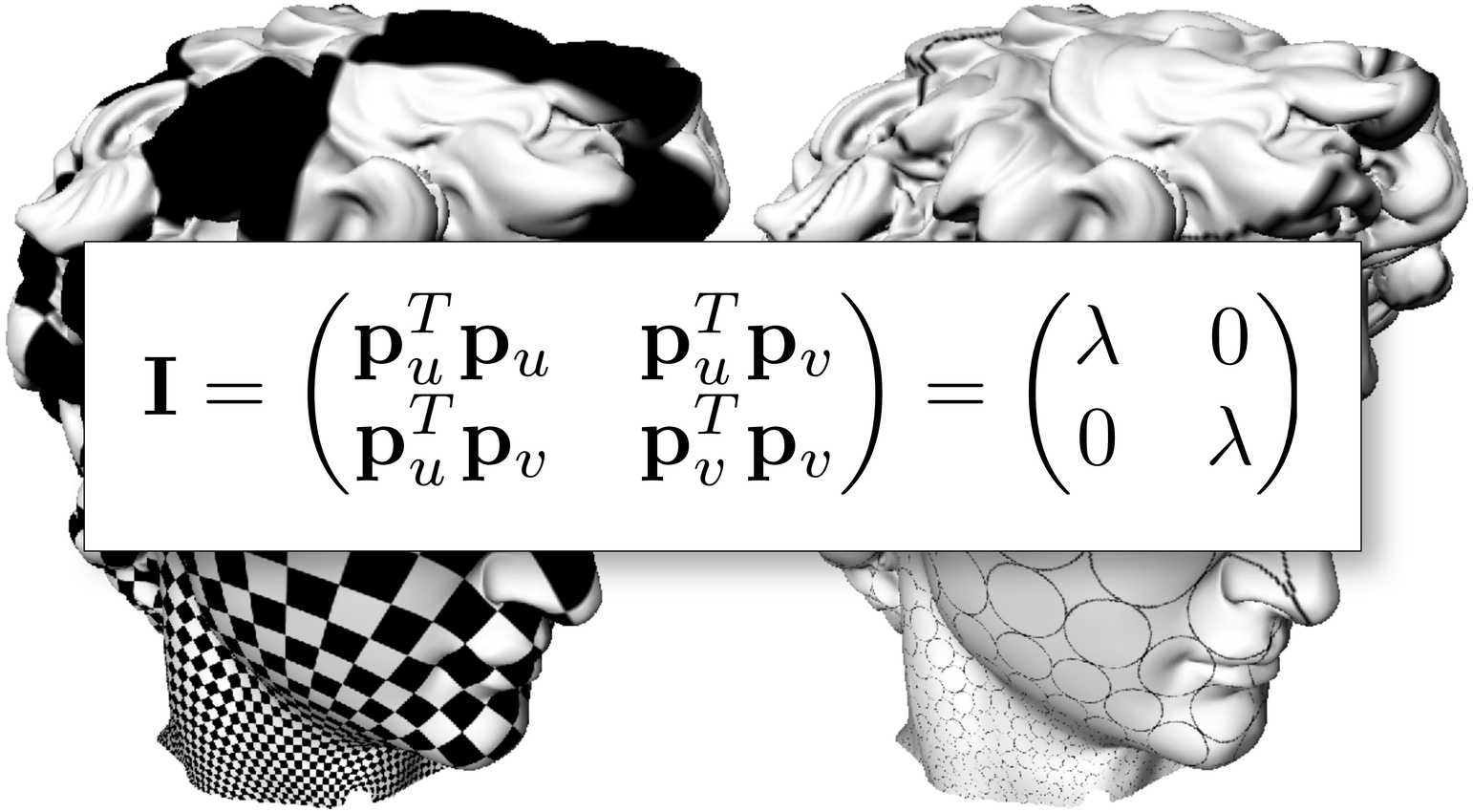
# Conformal Parameterization

- Angle preservation; circles are mapped to circles



# Conformal Parameterization

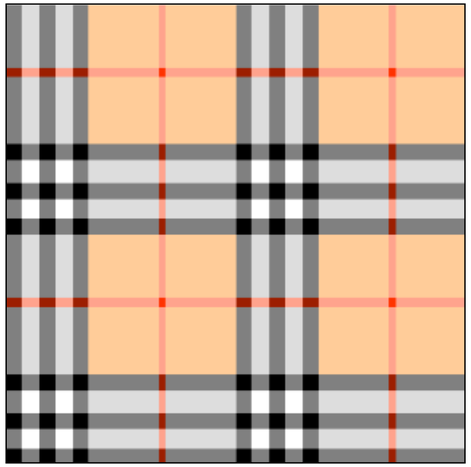
- Angle preservation; circles are mapped to circles

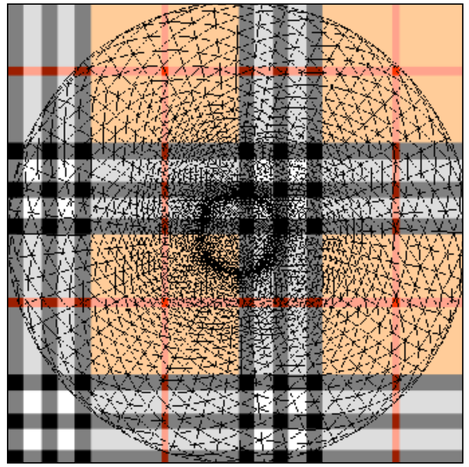


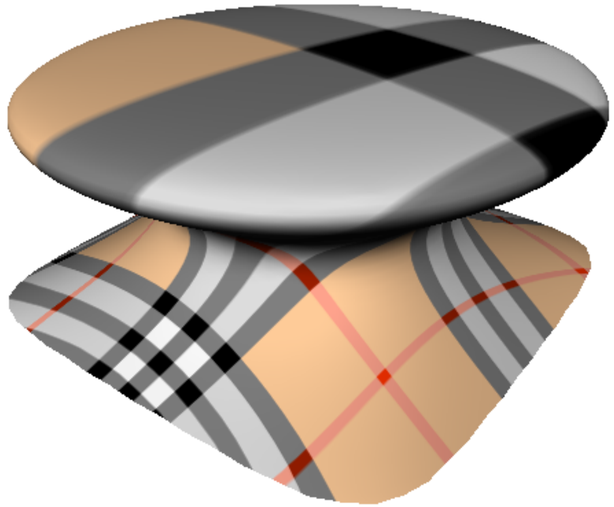
$$\mathbf{I} = \begin{pmatrix} \mathbf{p}_u^T \mathbf{p}_u & \mathbf{p}_u^T \mathbf{p}_v \\ \mathbf{p}_u^T \mathbf{p}_v & \mathbf{p}_v^T \mathbf{p}_v \end{pmatrix} = \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix}$$

# Area Distortion vs. Angle Distortion

- Is it possible to preserve both angles and areas at the same time?

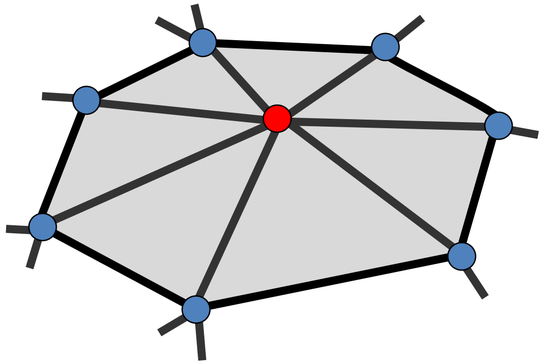






# Harmonic Mapping – Idea

- Want to flatten the mesh → no curvature → Laplace operator gives zero.



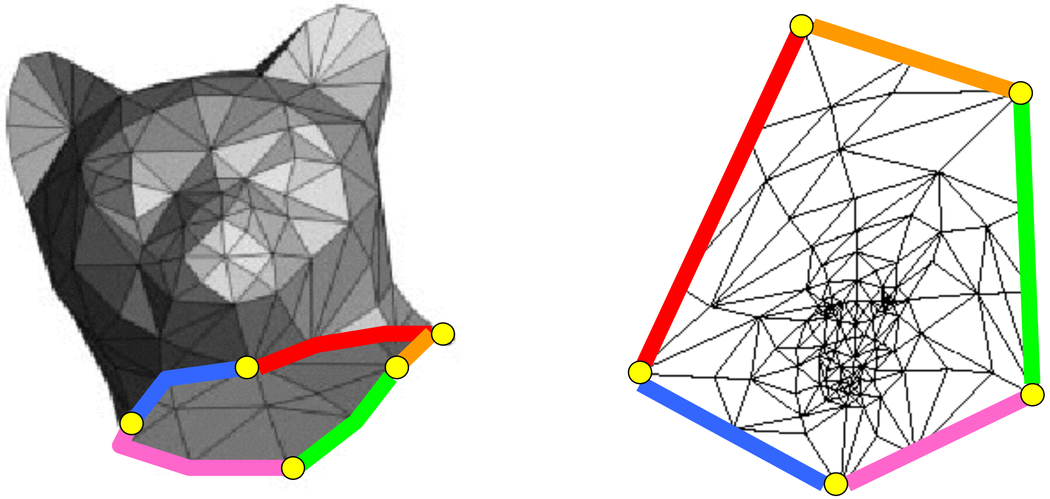
**u** = (*u*,*v*) domain

$$\Delta(\mathbf{u}) = 0$$

need boundary constraints
to prevent trivial solution;

# Convex Mapping (Tutte, Floater)

- Boundary vertices are fixed
- Convex weights



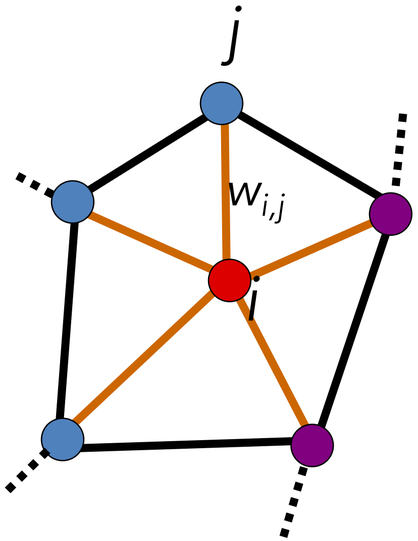
$\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ – inner vertices

$\mathbf{v}_{n+1}, \ldots, \mathbf{v}_N$ – boundary vertices

# Convex Mapping (Tutte, Floater)

- Boundary vertices are fixed
- Convex weights

$$\Delta(\mathbf{u}_i) = 0, \quad i = 1, \ldots, n$$




$$L(\mathbf{u}_i) = \frac{1}{W_i} \sum_{j \in \mathcal{N}(i)} w_{ij}(\mathbf{u}_j - \mathbf{u}_i) = 0, \quad i = 1, \ldots, n$$

$$w_{ij} > 0$$

# Convex Mapping (Tutte, Floater)

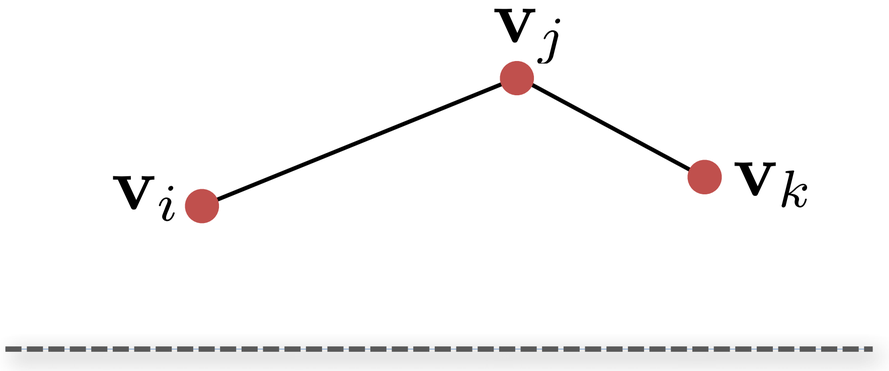
- Solve the linear system

$$L\mathbf{u} = 0 \qquad \mathbf{u} \in \mathbb{R}^{n \times 2}$$

- The values of the boundary vertices are known and thus substituted (transfer to right-hand side)

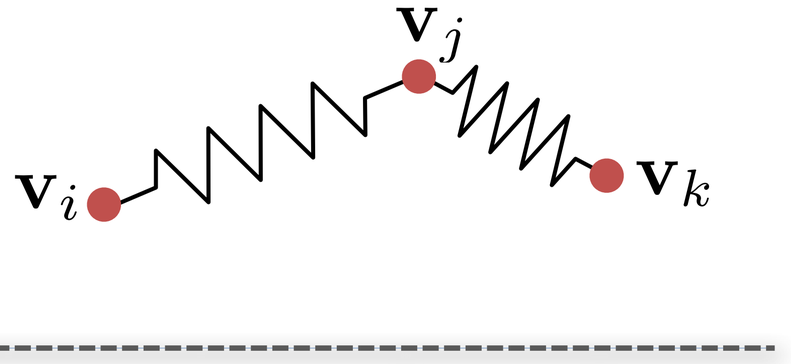
# Harmonic Mapping

- Inner mesh edges as springs
- Find minimum-energy state where all vertices lie in the 2D plane



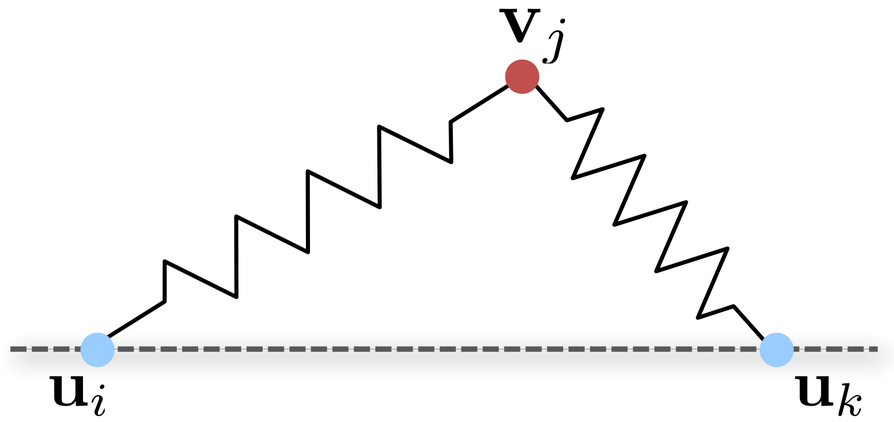
# Harmonic Mapping

- Inner mesh edges as springs
- Find minimum-energy state where all vertices lie in the 2D plane



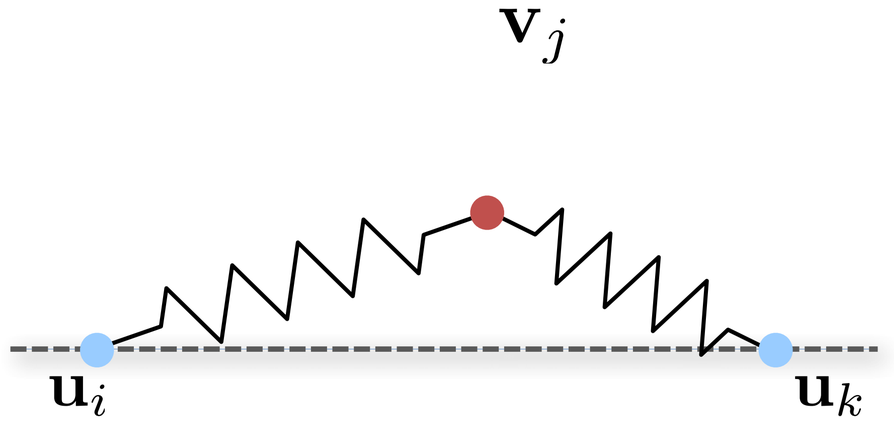
# Harmonic Mapping

- Inner mesh edges as springs
- Find minimum-energy state where all vertices lie in the 2D plane



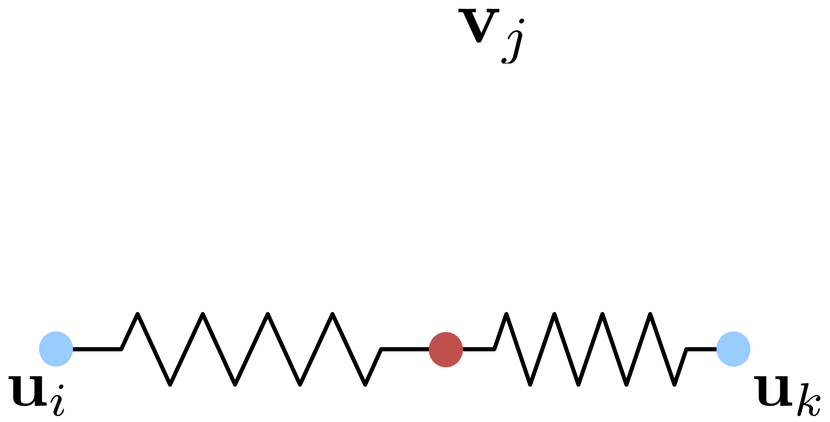
# Harmonic Mapping

- Inner mesh edges as springs
- Find minimum-energy state where all vertices lie in the 2D plane



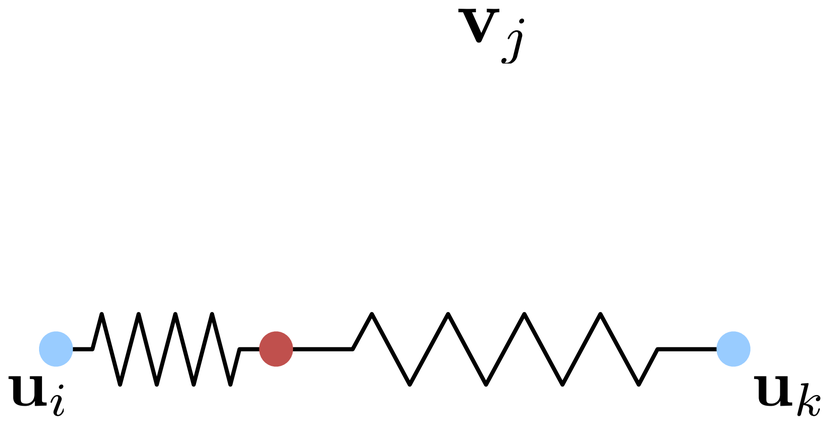
# Harmonic Mapping

- Inner mesh edges as springs
- Find minimum-energy state where all vertices lie in the 2D plane



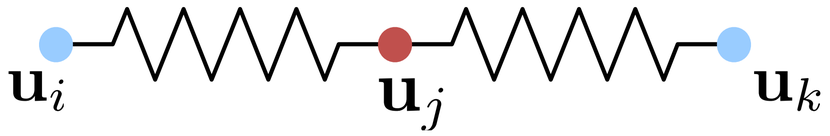
# Harmonic Mapping

- Inner mesh edges as springs
- Find minimum-energy state where all vertices lie in the 2D plane



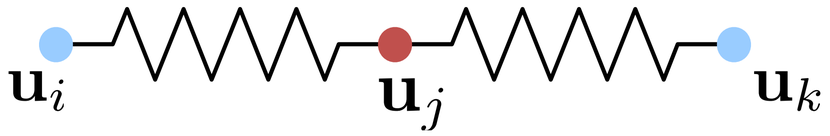
# Harmonic Mapping

- Inner mesh edges as springs
- Find minimum-energy state where all vertices lie in the 2D plane



# Harmonic Mapping

- Inner mesh edges as springs
- Find minimum-energy state where all vertices lie in the 2D plane
- Spring energy:

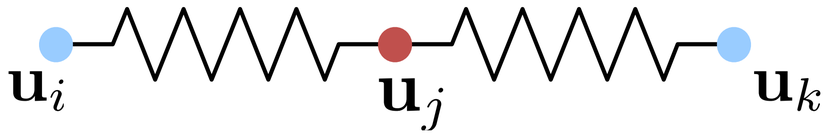



$$\frac{1}{2} k_{i,j} \|\mathbf{u}_i - \mathbf{u}_j\|^2$$

$$\mathbf{u}_i, \mathbf{u}_j \in \mathbb{R}^2$$

# Harmonic Mapping

- Inner mesh edges as springs
- Find minimum-energy state where all vertices lie in the 2D plane
- Total spring energy of the flattened mesh:




$$E(\mathbf{u}_1, \ldots, \mathbf{u}_n) = \sum_{(i,j)\in\mathcal{E}} \frac{1}{2} k_{i,j} \|\mathbf{u}_i - \mathbf{u}_j\|^2$$

# Demo

- https://libigl.github.io/libigl-python-bindings/tut-chapter4/

# Minimizing Spring Energy

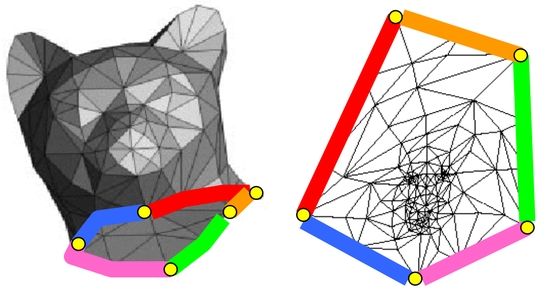
$$E(\mathbf{u}_1, \ldots, \mathbf{u}_n) = \sum_{(i,j)\in\mathcal{E}} \frac{1}{2} k_{i,j} \|\mathbf{u}_i - \mathbf{u}_j\|^2$$

$$\frac{\partial E(\mathbf{u}_1, \ldots, \mathbf{u}_n)}{\partial \mathbf{u}_i} = \sum_{j\in\mathcal{N}(i)} k_{i,j}(\mathbf{u}_i - \mathbf{u}_j) = 0$$

$$\sum_{j\in\mathcal{N}(i)\cap\mathcal{B}} k_{i,j}\mathbf{u}_i + \sum_{j\in\mathcal{N}(i)\setminus\mathcal{B}} k_{i,j}(\mathbf{u}_i - \mathbf{u}_j) = \sum_{j\in\mathcal{N}(i)\cap\mathcal{B}} k_{i,j}\mathbf{u}_j$$

unknown flat vertex positions

known fixed boundary positions



$\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ – inner vertices

$\mathbf{v}_{n+1}, \ldots, \mathbf{v}_N$ – boundary vertices

# Minimizing Spring Energy

- Sparse linear system of *n* equations to solve!

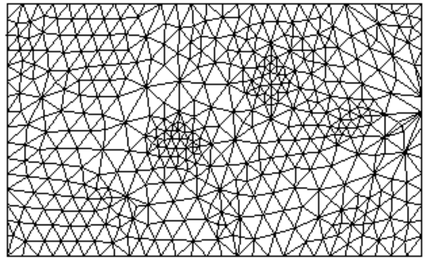
$$\sum_{j \in \mathcal{N}(i) \cap \mathcal{B}} k_{i,j}\mathbf{u}_i + \sum_{j \in \mathcal{N}(i) \setminus \mathcal{B}} k_{i,j}(\mathbf{u}_i - \mathbf{u}_j) = \sum_{j \in \mathcal{N}(i) \cap \mathcal{B}} k_{i,j}\mathbf{u}_j$$

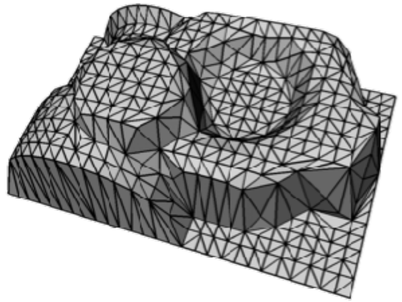
$$\begin{pmatrix} \sum_j k_{i,j} & * & \cdots & -k_{i,j} \\ * & \sum_j k_{i,j} & * & \vdots \\ \vdots & * & \ddots & * \\ -k_{j,i} & \cdots & * & \sum_j k_{i,j} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_n \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{u}}_1 \\ \bar{\mathbf{u}}_2 \\ \vdots \\ \bar{\mathbf{u}}_n \end{pmatrix}$$

# Choice of spring constants $k_{i,j}$

- Uniform

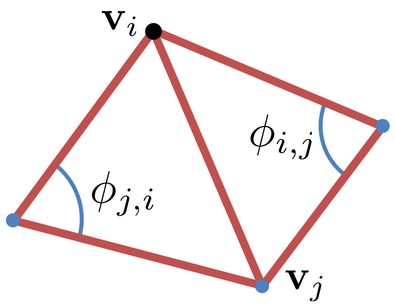
$$k_{i,j} = 1$$

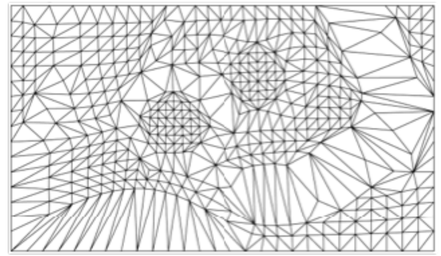




- Cotan

$$k_{i,j} = \cot \phi_{i,j} + \cot \phi_{j,i}$$



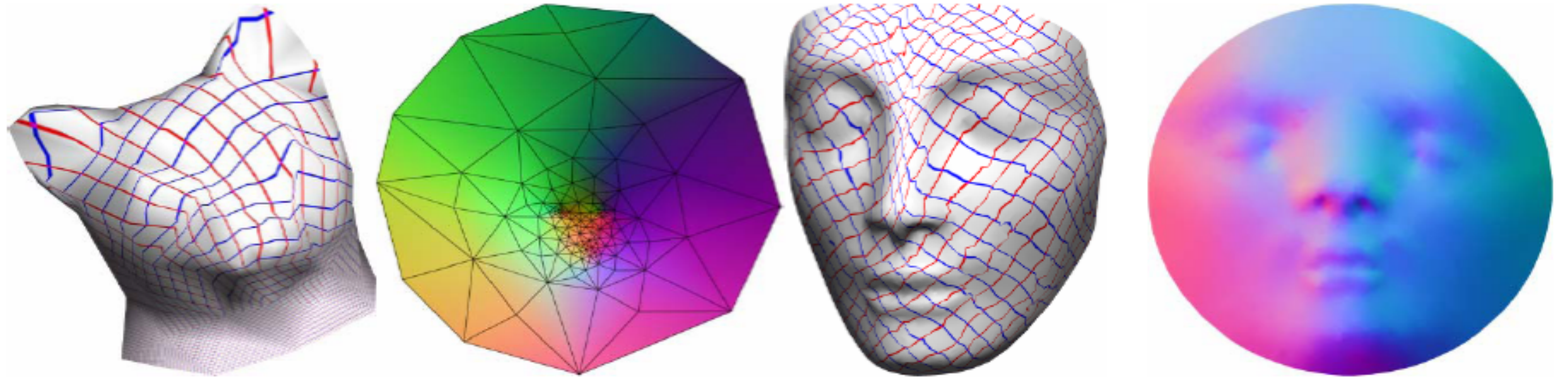
# Tutte's Theorem

- If the weights are **nonnegative**, and the boundary is fixed to a **convex** polygon, the parameterization is **bijective**
- Tutte'63 proved for uniform weights
- Floater'97 extended to arbitrary nonnegative weights


  - W.T. Tutte. "How to draw a graph". Proceedings of the London Mathematical Society, 13(3):743-768, 1963.
  - Michael S. Floater. 1997. Parametrization and smooth approximation of surface triangulations. Comput. Aided Geom. Des. 14, 3 (April 1997), 231–250. DOI:https://doi.org/10.1016/S0167-8396(96)00031-3

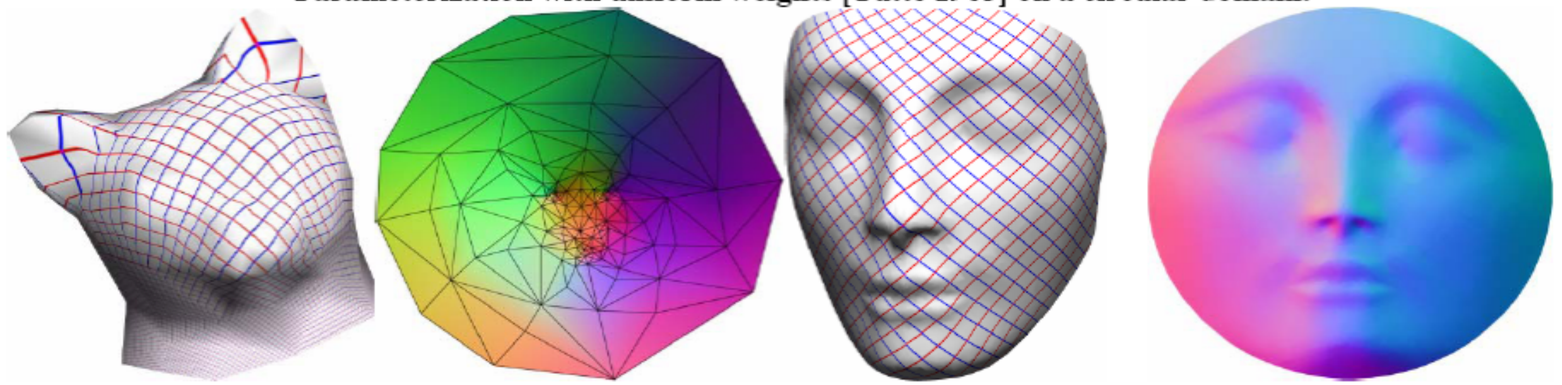
# Comparison of Weights

uniform weights



Parameterization with uniform weights [Tutte 1963] on a circular domain.

cotan weights



Parameterization with harmonic weights [Eck et al. 1995] on a circular domain.

Eck et al. 1995, "Multiresolution analysis of arbitrary meshes", SIGGRAPH 1995

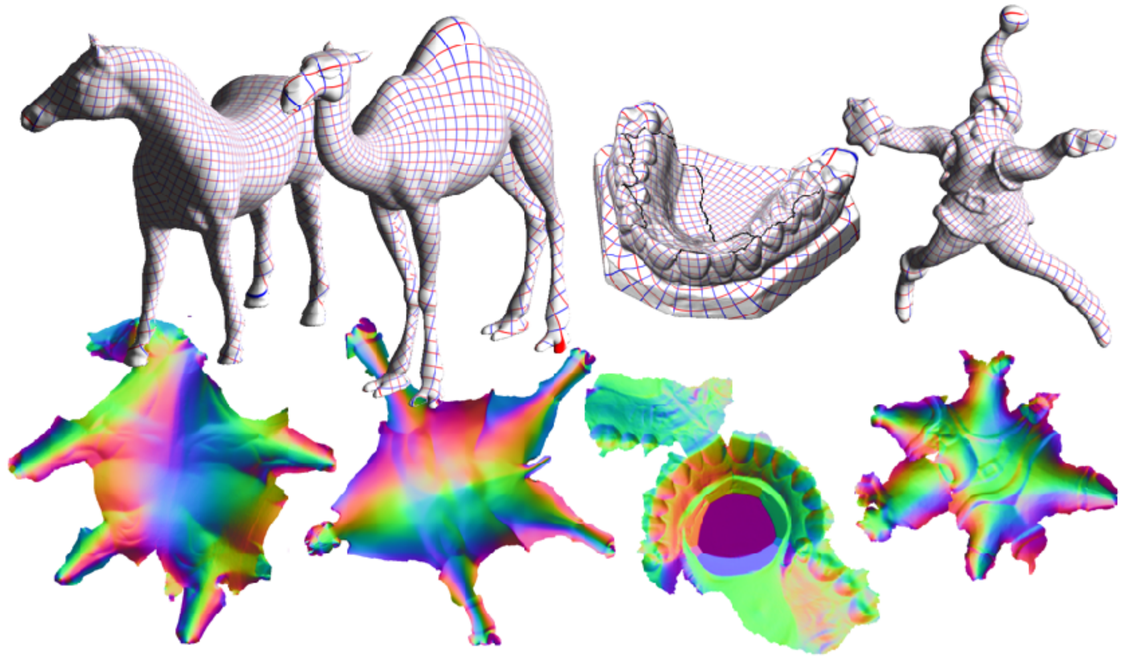
# Discussion

- The results of cotan-weights mapping are better than those of uniform convex mapping (local area and angles preservation).
- But: the mapping is not always legal (the cotan weights can be negative for badly-shaped triangles…)

- In any case: sparse system to solve, so robust and efficient numerical solvers exist

# Discussion

- Both mappings have the problem of fixed boundary – it constrains the minimization and causes distortion.
- More advanced methods do not require boundary conditions.



ABF++ method,
Sheffer et al. 2005
http://www.cs.ubc.ca/~sheffa/ABF++/abf.htm

# Summary

- Parametrization have many uses in computer graphics: they allow to use 2D algorithms and data on surfaces
- Designing a parametrization is equivalent to designing two scalar functions
- Scalar functions can be designed interpolating values directly, or interpolating vectors and solving a Poisson problem

# References

- Surface Parameterization: a Tutorial and Survey: http://graphics.stanford.edu/courses/cs468-05-fall/Papers/param-survey.pdf
- Polygon Mesh Processing: Chapter 5