

# Geometric Modeling

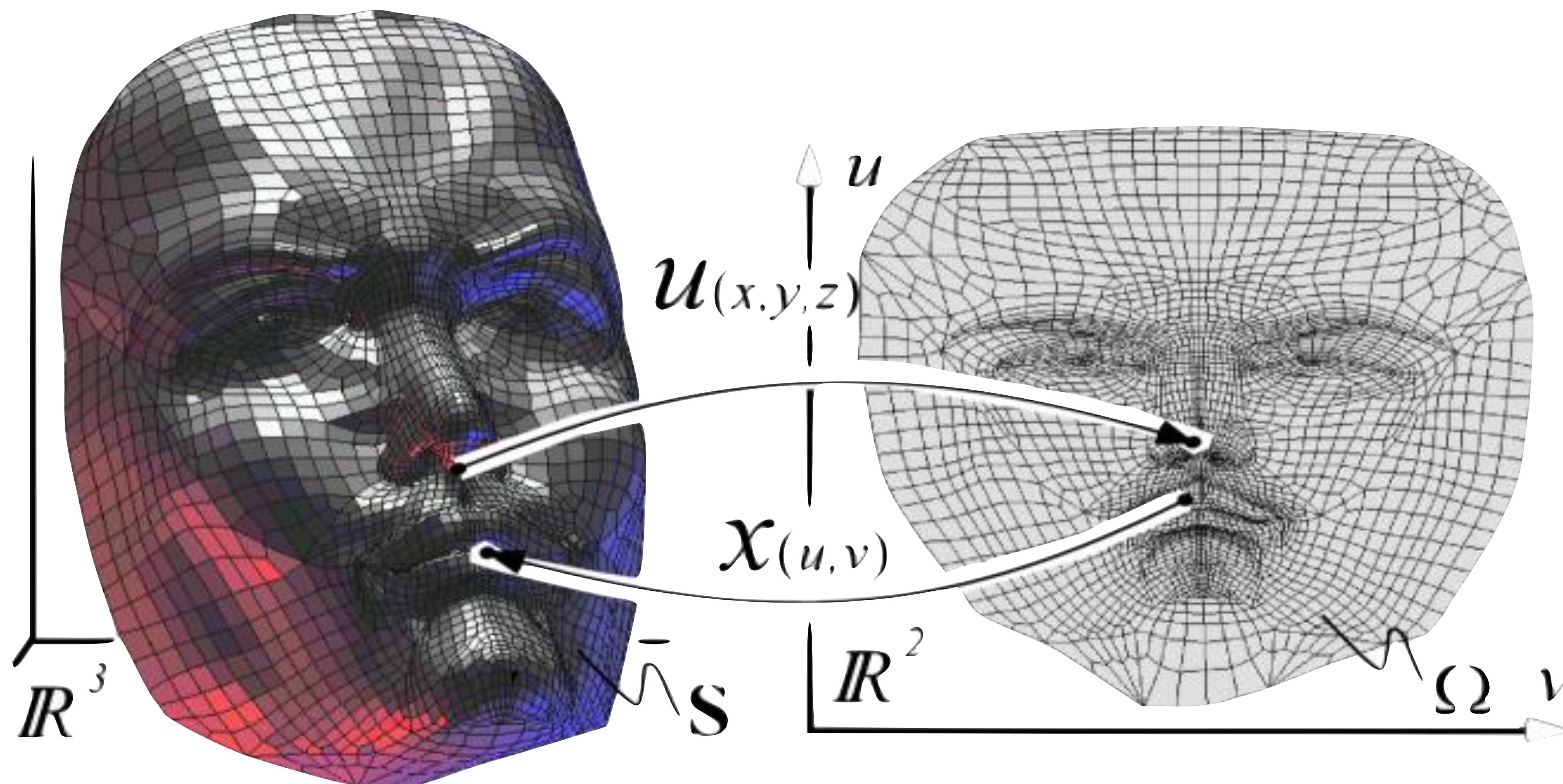
## Assignment 4: Mesh Parametrization

knowledgements: Olga Diamanti, Julian Panetta  
CSC 486B/586B - Geometric Modeling - Teseo Schneider

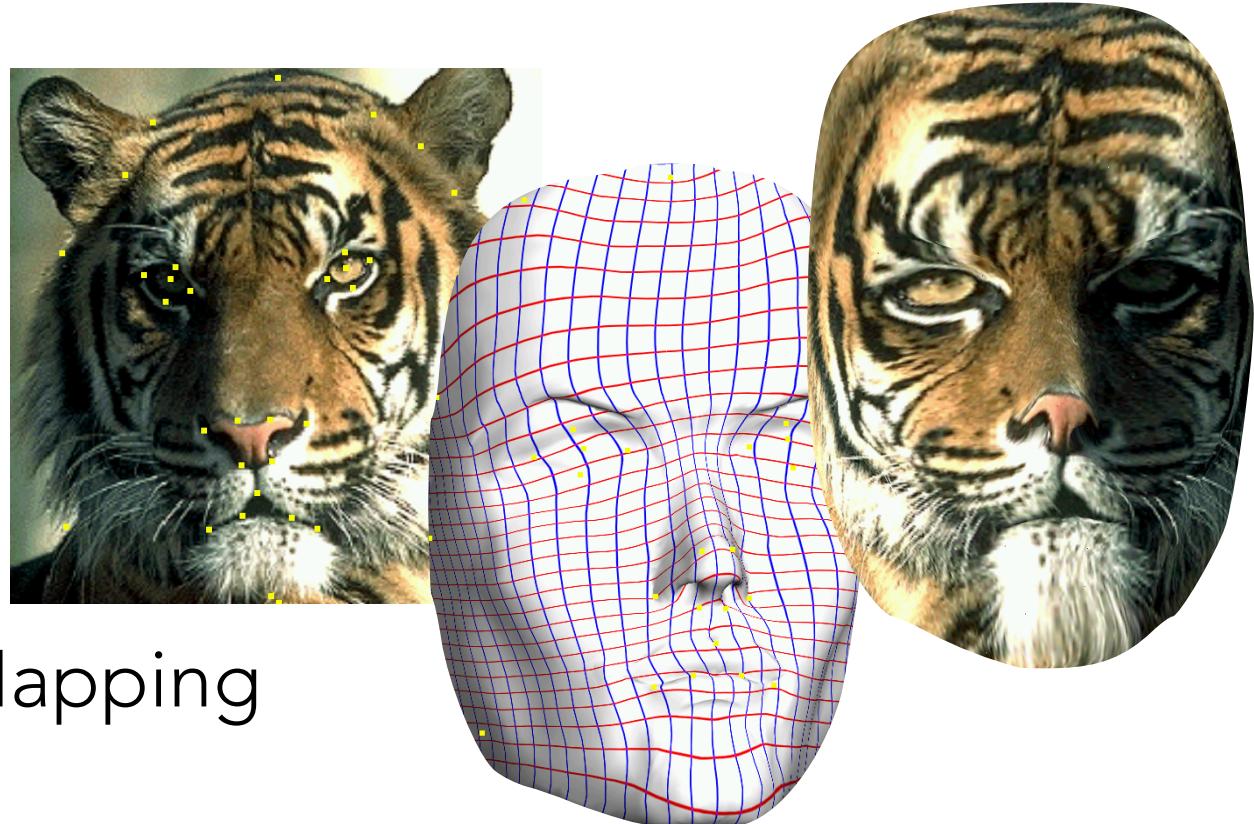


Computer Science

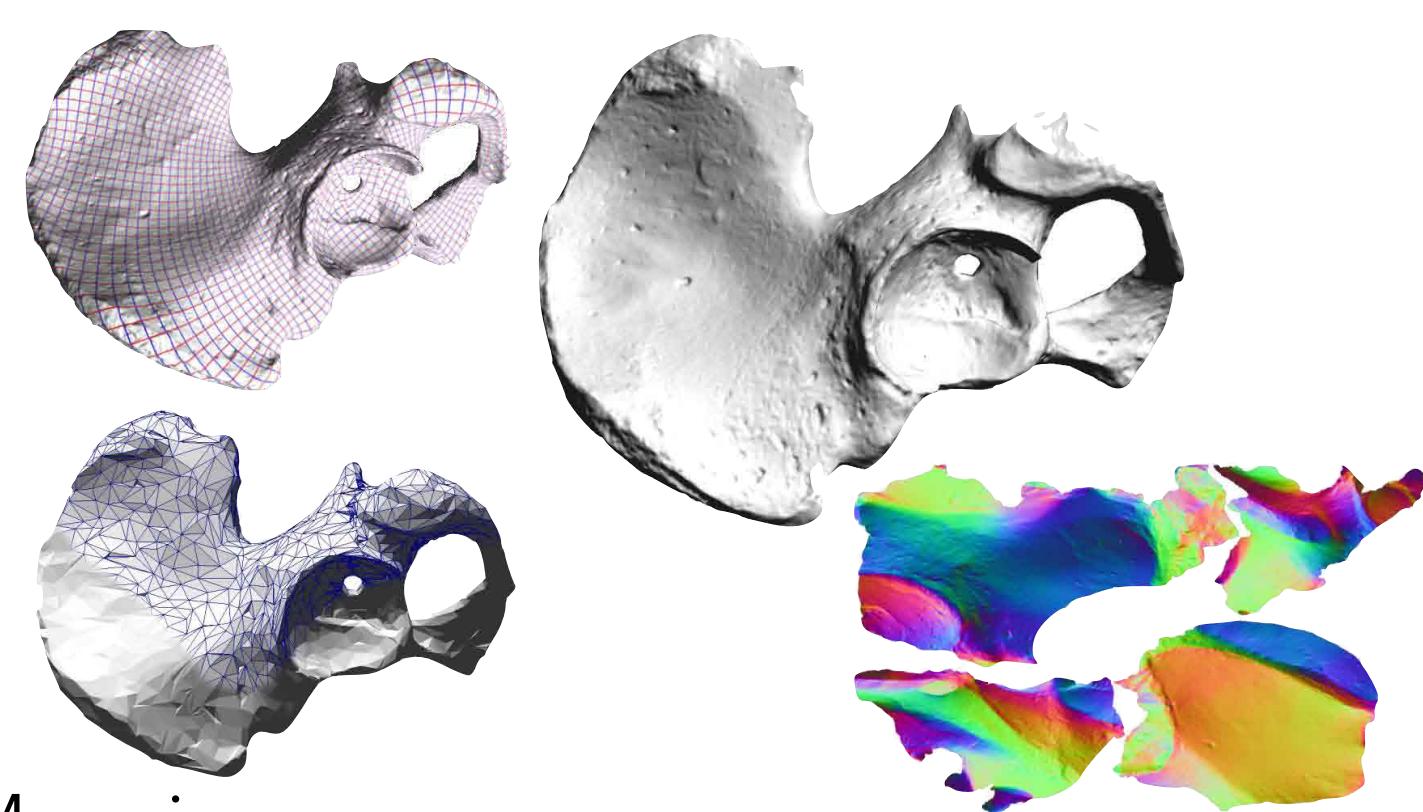
# Mesh Parametrization



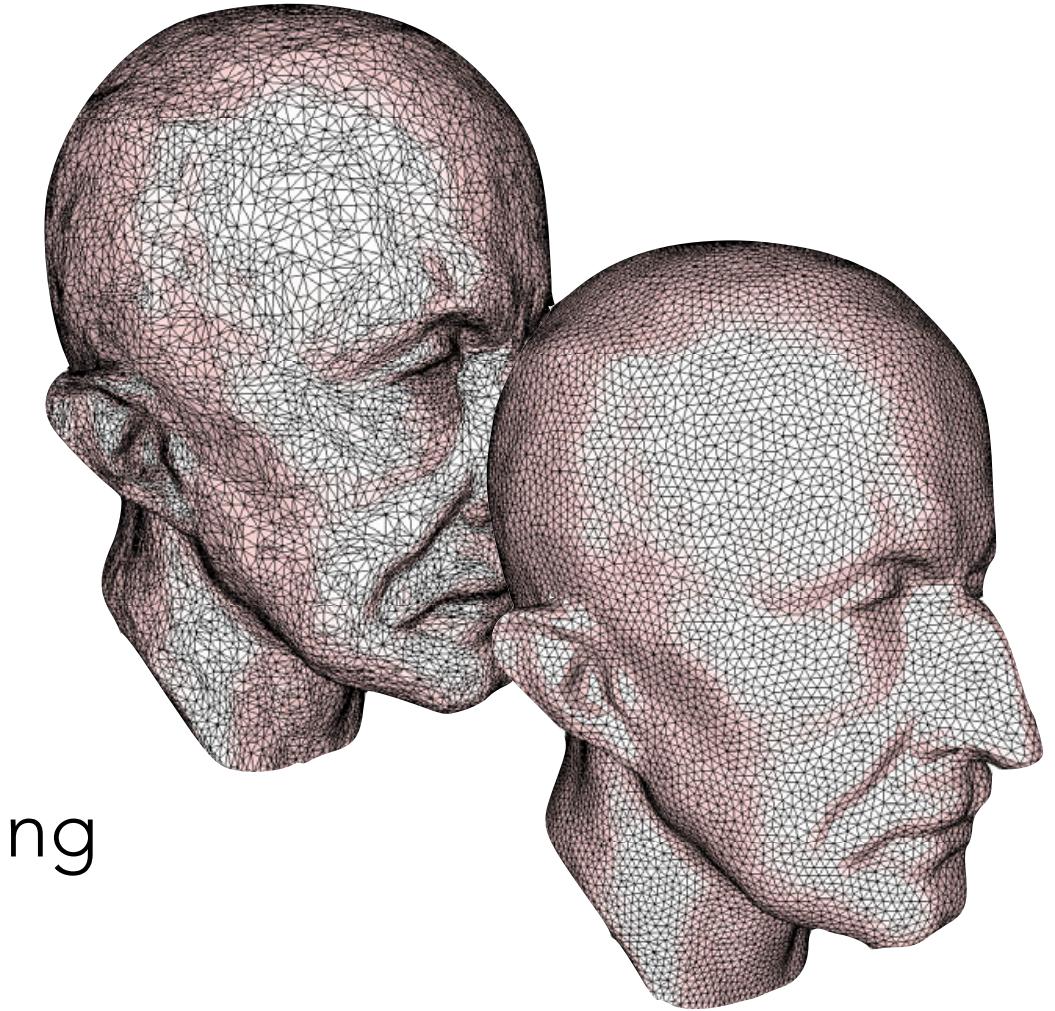
# Parametrization Applications



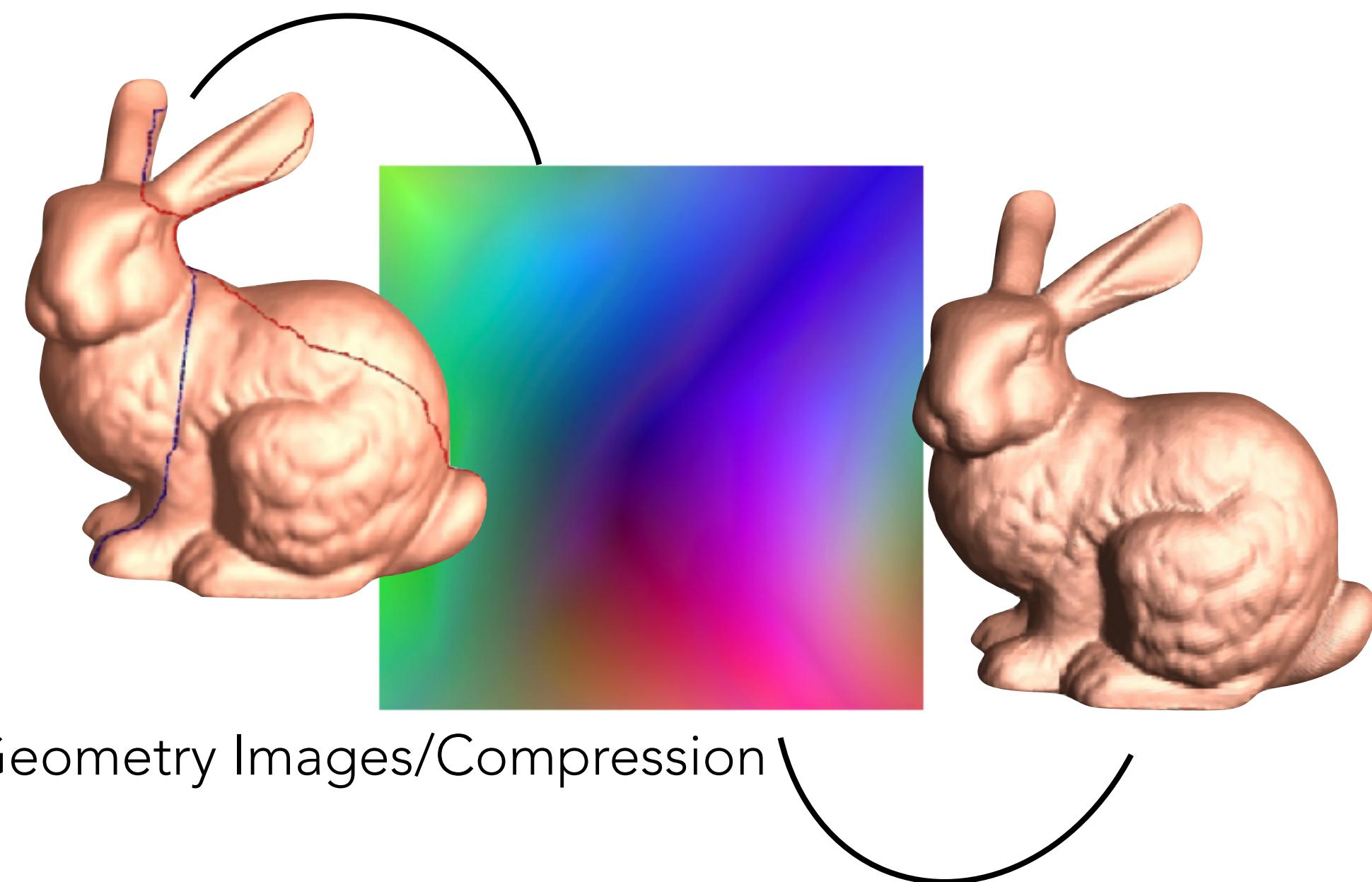
Texture Mapping



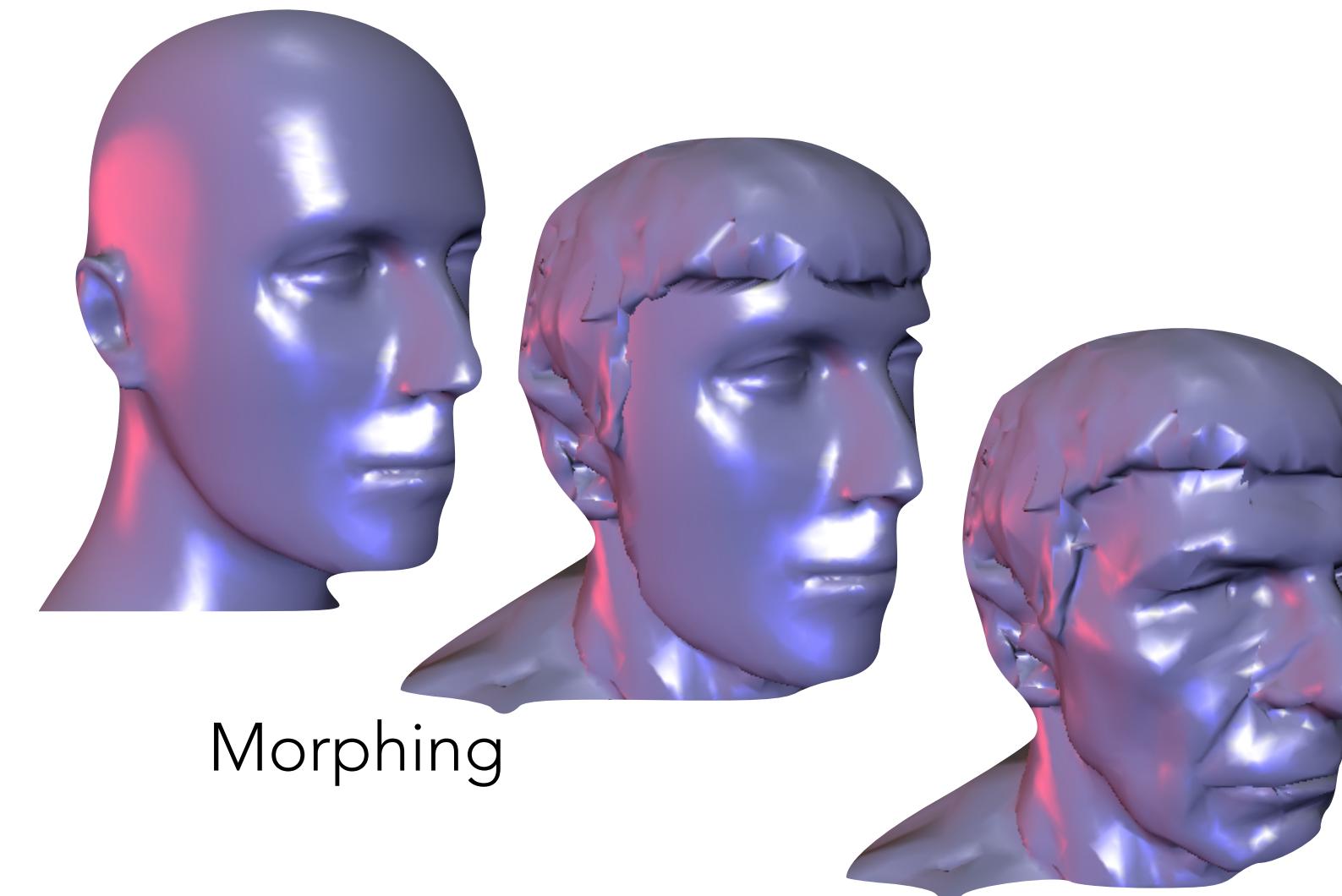
Normal/Bump Mapping



Remeshing



Geometry Images/Compression

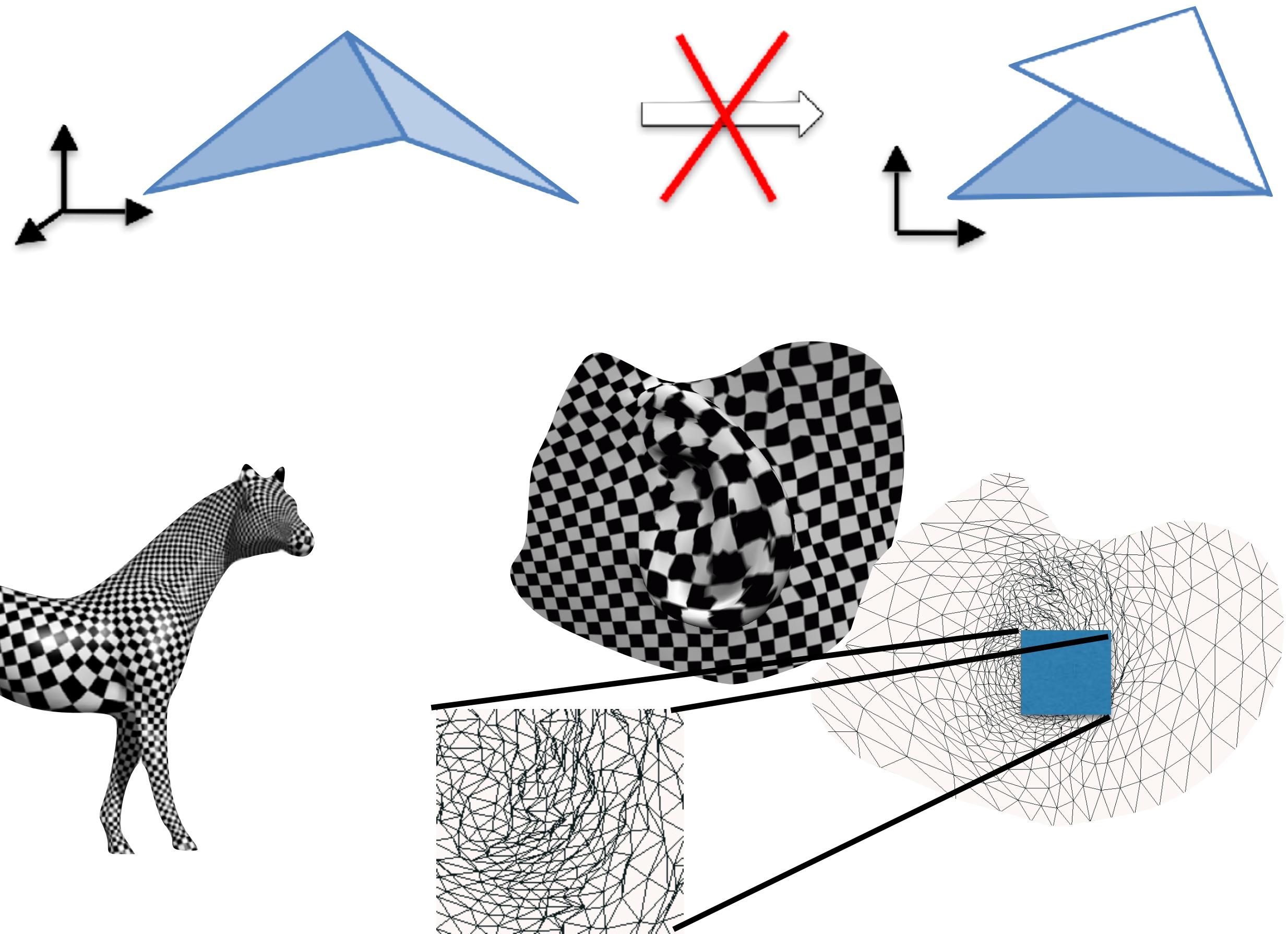


Morphing

- Detail Transfer
- Mesh Completion
- Editing
- Surface Fitting
- ...

# Desirable Properties

- Bijective (invertible):  
    avoid flips/overlaps
- Minimal distortion
  - Area
  - Angle
- Efficiently computable

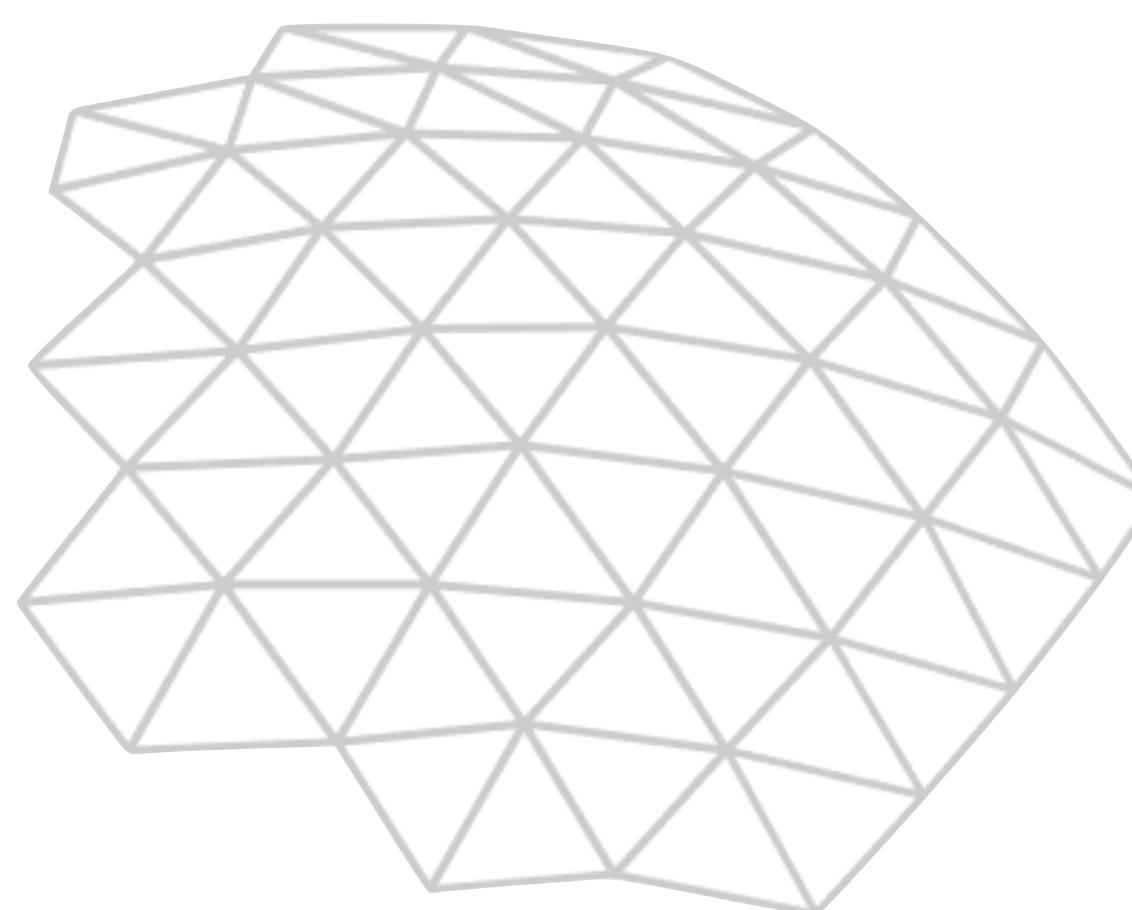


# Parametrization Approaches

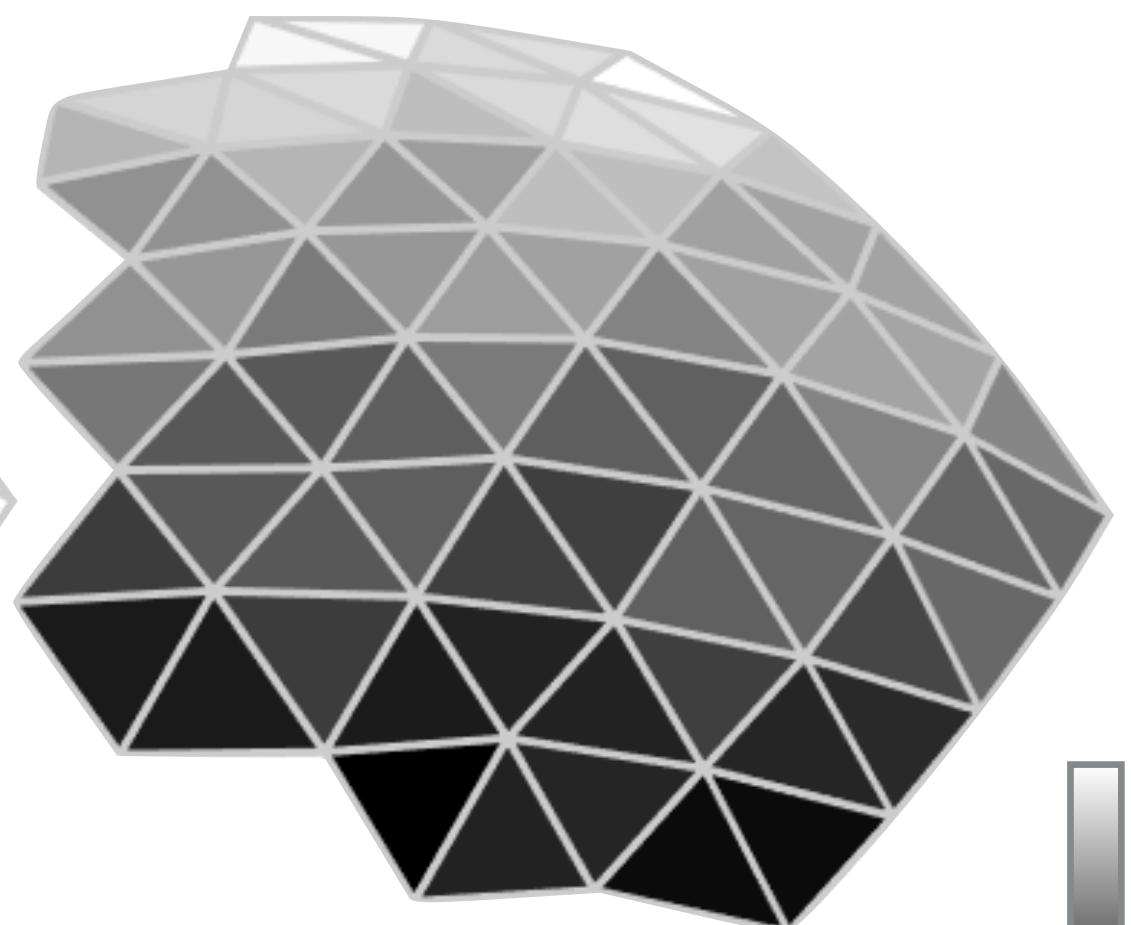
- This assignment will consider 3 approaches
  - Least-Squares Conformal (LSCM)
  - Harmonic
  - Field-Guided
- LSCM, Harmonic implemented in libigl

# Formulation: Design Scalar Fields

- Compute two scalar fields ( $u, v$ ) giving the plane coordinates of each mesh point



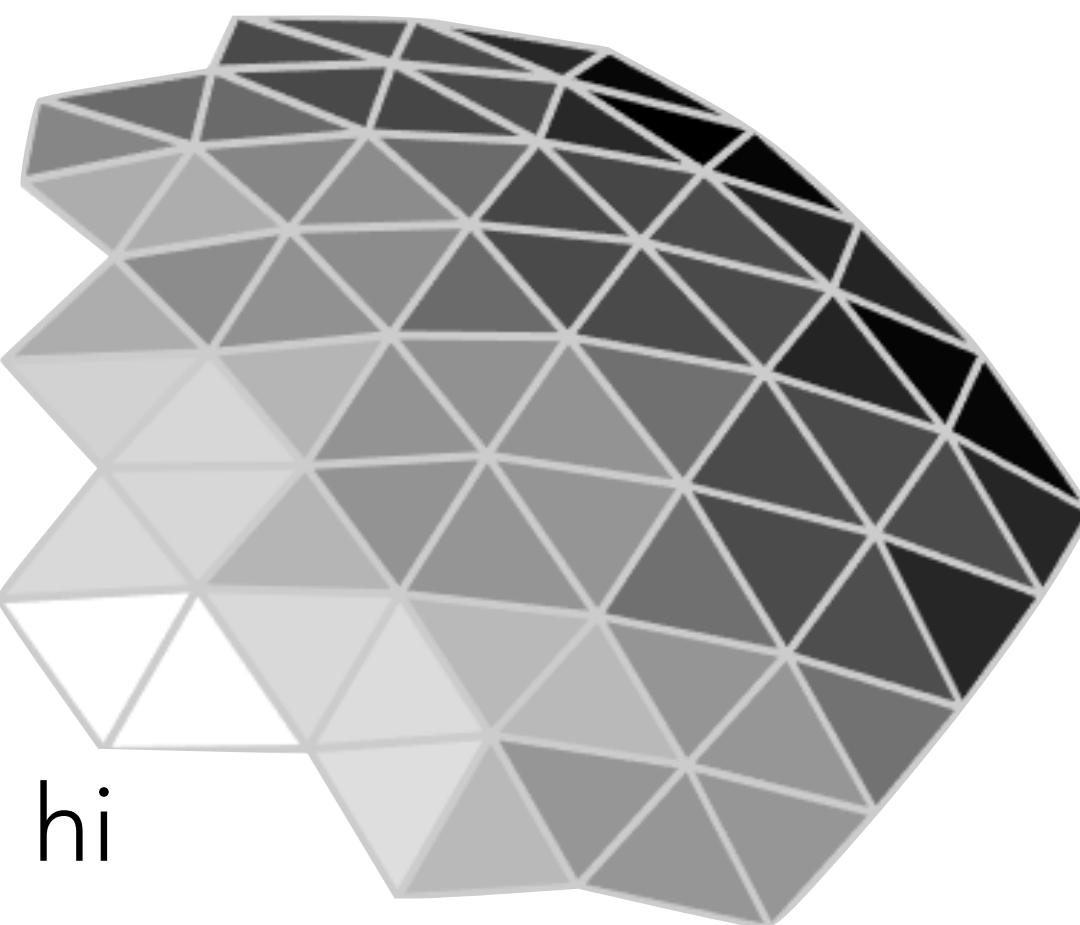
mesh



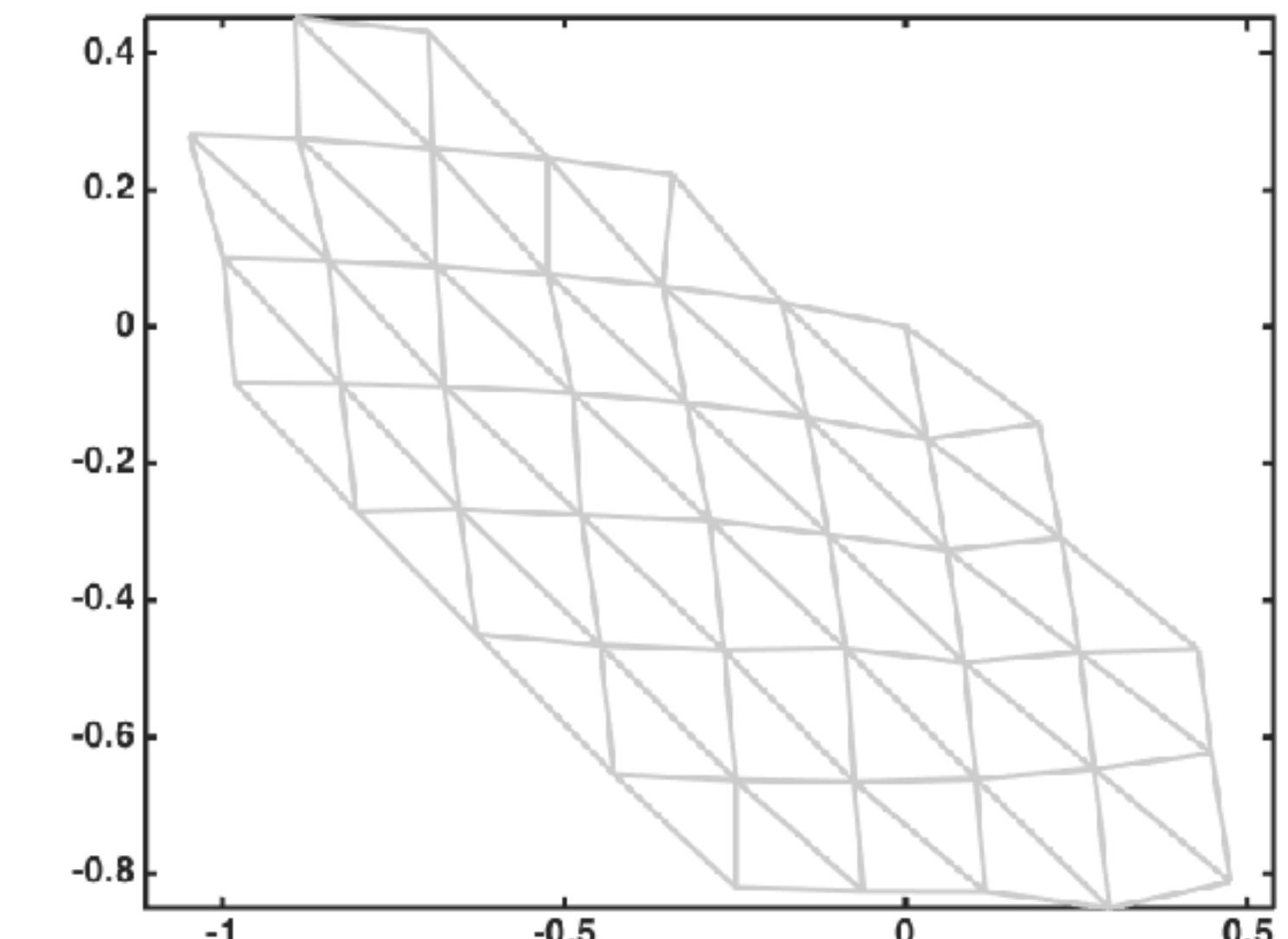
U-function



hi  
lo



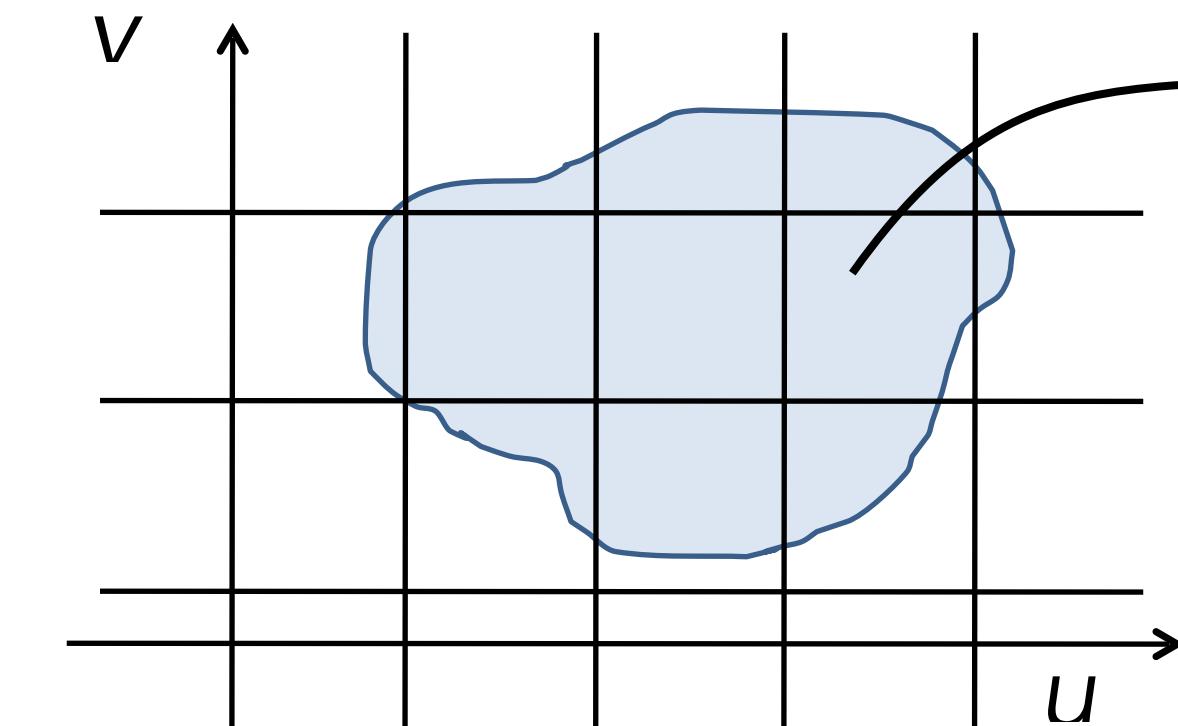
V-function



flattened mesh (UV-domain)

# Least-Squares Conformal

- **Conformal** maps have **zero angle distortion**  
(preserve 90° angles)



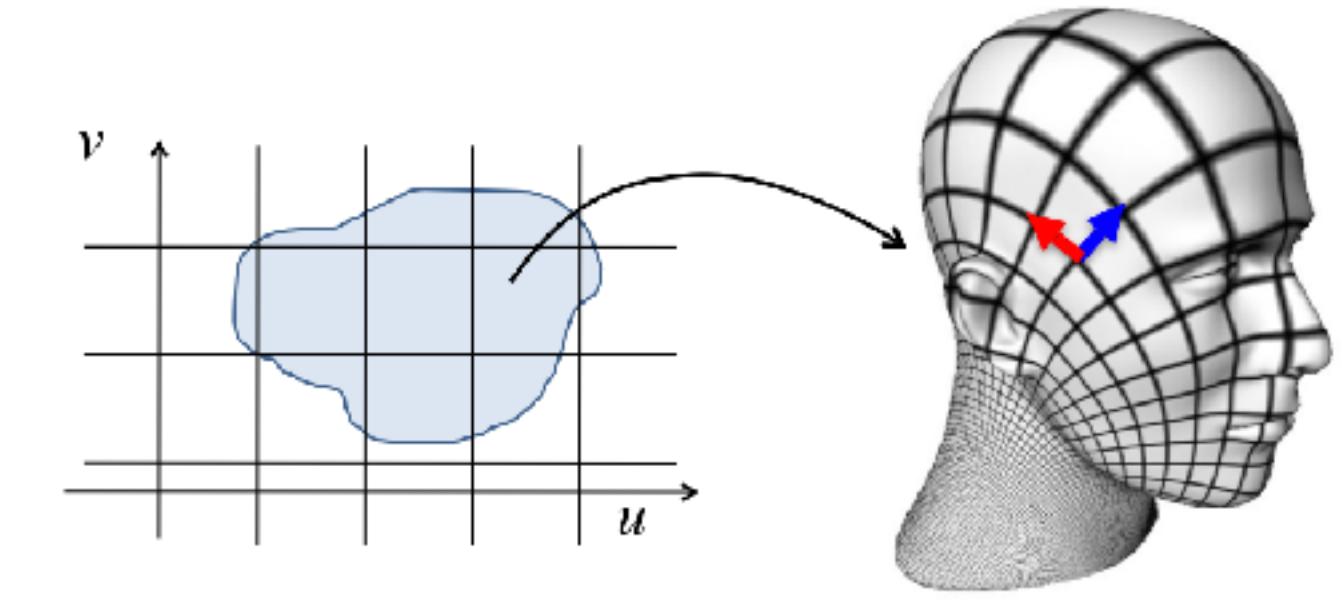
- Least-Squares Conformal: minimize angle distortion

$$\min_{u,v} \int_M \|\nabla u^\perp - \nabla v\|^2 dV$$

- Encourage perpendicular, equally scaled coordinate vectors

# Least-Squares Conformal

$$\begin{aligned}
 & \min_{u,v} \frac{1}{2} \int_M \|\nabla u^\perp - \nabla v\|^2 dV \\
 &= \min_{u,v} \frac{1}{2} \int_M (\hat{n} \times \nabla u - \nabla v) \cdot (\hat{n} \times \nabla u - \nabla v) dV \\
 &= \min_{u,v} \frac{1}{2} \int_M \|\hat{n} \times \nabla u\|^2 - 2(\hat{n} \times \nabla u) \cdot \nabla v + \|\nabla v\|^2 dV \\
 &= \min_{u,v} \frac{1}{2} \int_M \|\nabla u\|^2 + \|\nabla v\|^2 - 2(\hat{n} \times \nabla u) \cdot \nabla v dV \\
 &= \min_{u,v} \frac{1}{2} \int_M \|\nabla u\|^2 + \|\nabla v\|^2 - 2 \det \left( \begin{array}{c} \nabla u^T \\ \nabla v^T \\ \hat{n}^T \end{array} \right) dV \\
 &= \min_{u,v} \frac{1}{2} \int_M \|\nabla u\|^2 dV + \frac{1}{2} \int_M \|\nabla v\|^2 dV - A(u, v)
 \end{aligned}$$



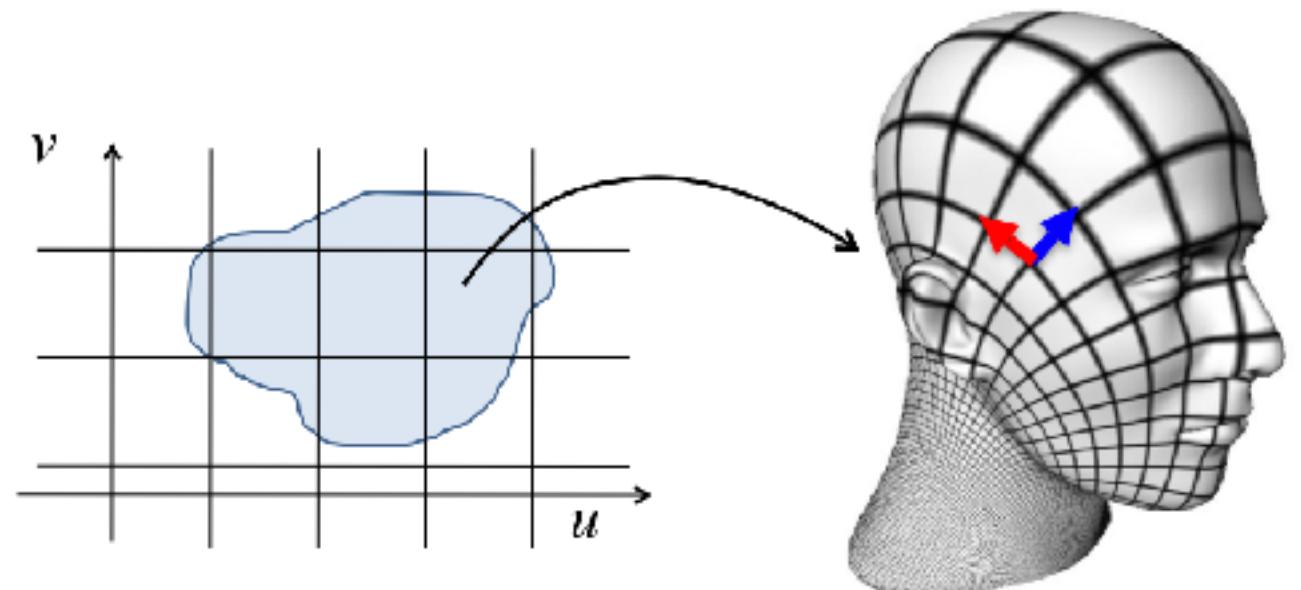
Jacobian Determinant  
(Area scaling at each pt)

Area in (u, v) plane!

# LSCM ==> Harmonic

- We showed LSCM decomposes into:

$$\frac{1}{2} \int_M \| \nabla u^\perp - \nabla v \|^2 dV = \frac{1}{2} \int_M \| \nabla u \|^2 dV + \frac{1}{2} \int_M \| \nabla v \|^2 dV - \underline{A(u, v)}$$



Smoothness of  $u$  and  $v$   
("Dirichlet Energy")

Area in  
plane

- If we fix the **flattened boundary** (area), LSCM is equivalent to designing **smooth scalar fields**



University  
of Victoria

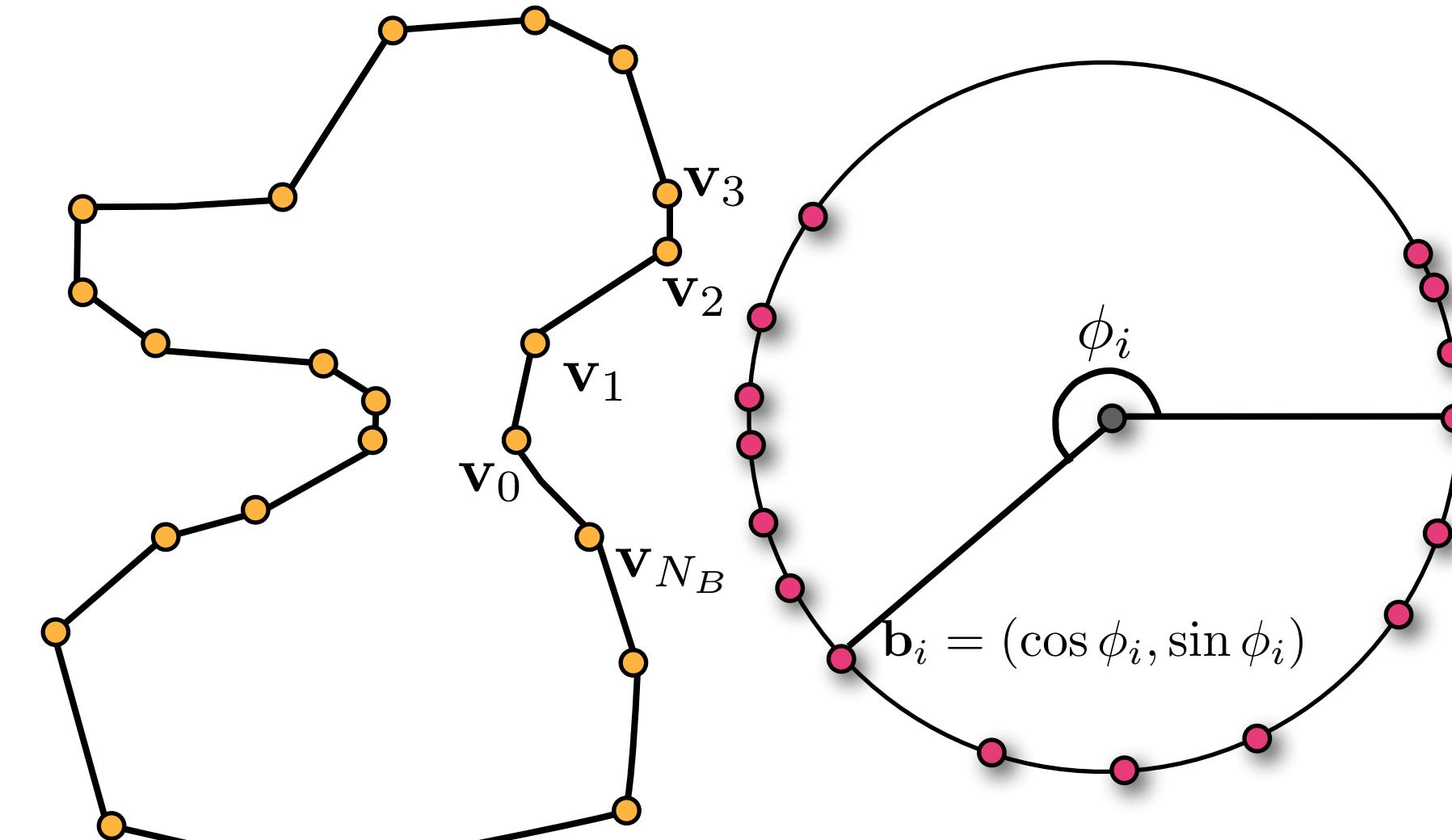
Computer Science

# Harmonic Parametrization

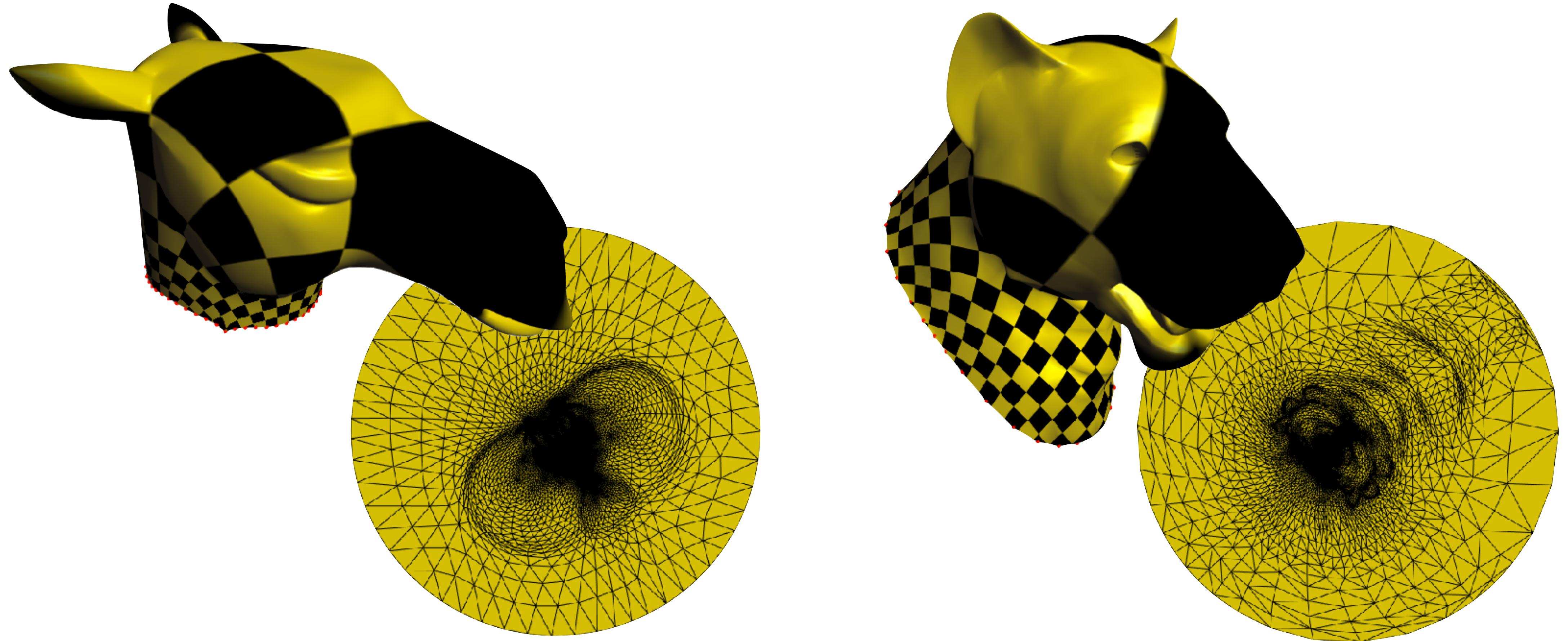
- Map mesh boundary to a convex polygon in plane
- Solve for smoothly interpolating  $(u, v)$ :

$$\min_{u=u_b \text{ on } \partial M} \frac{1}{2} \int_M \|\nabla u\|^2 dV \quad \Rightarrow \quad \begin{cases} \Delta u = 0 & \text{in } M \\ u = u_b & \text{on } \partial M \end{cases}$$

Just solve Laplace equation  
with fixed boundary values!



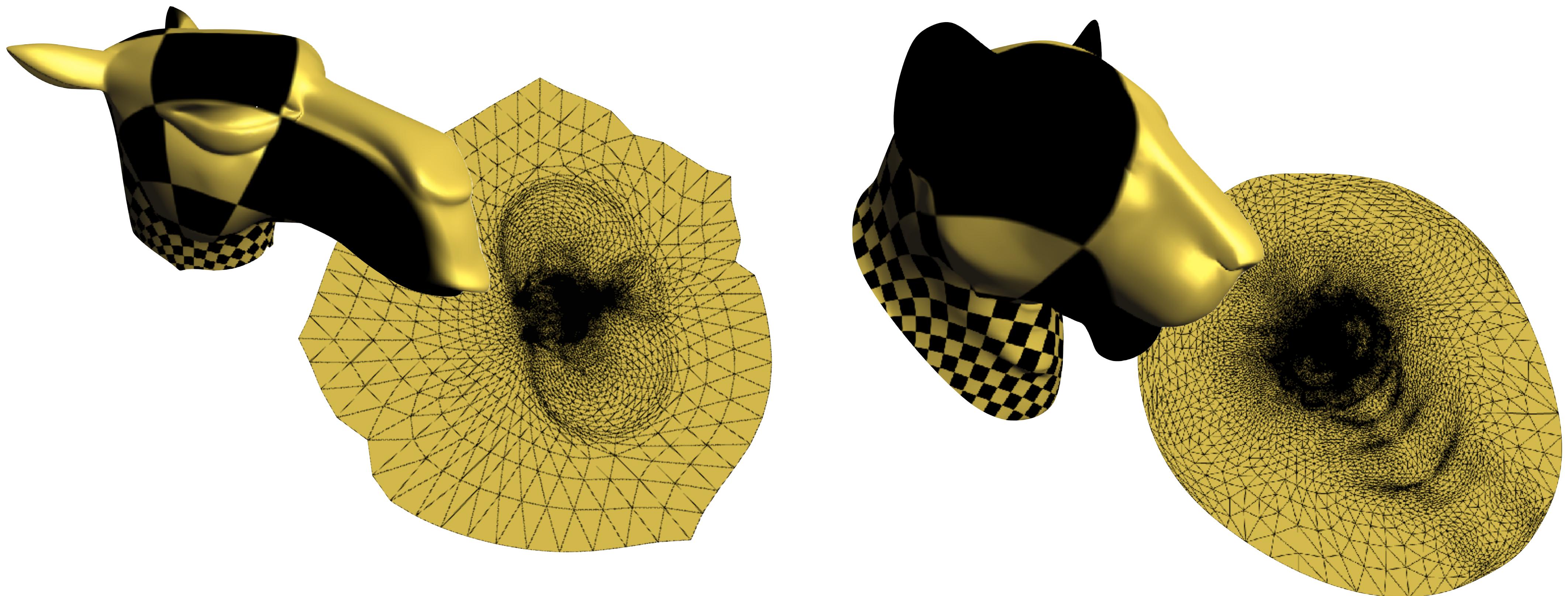
# Harmonic Parametrization



Already implemented in libigl!

See tutorial Chapter 4

# LSCM Parametrization

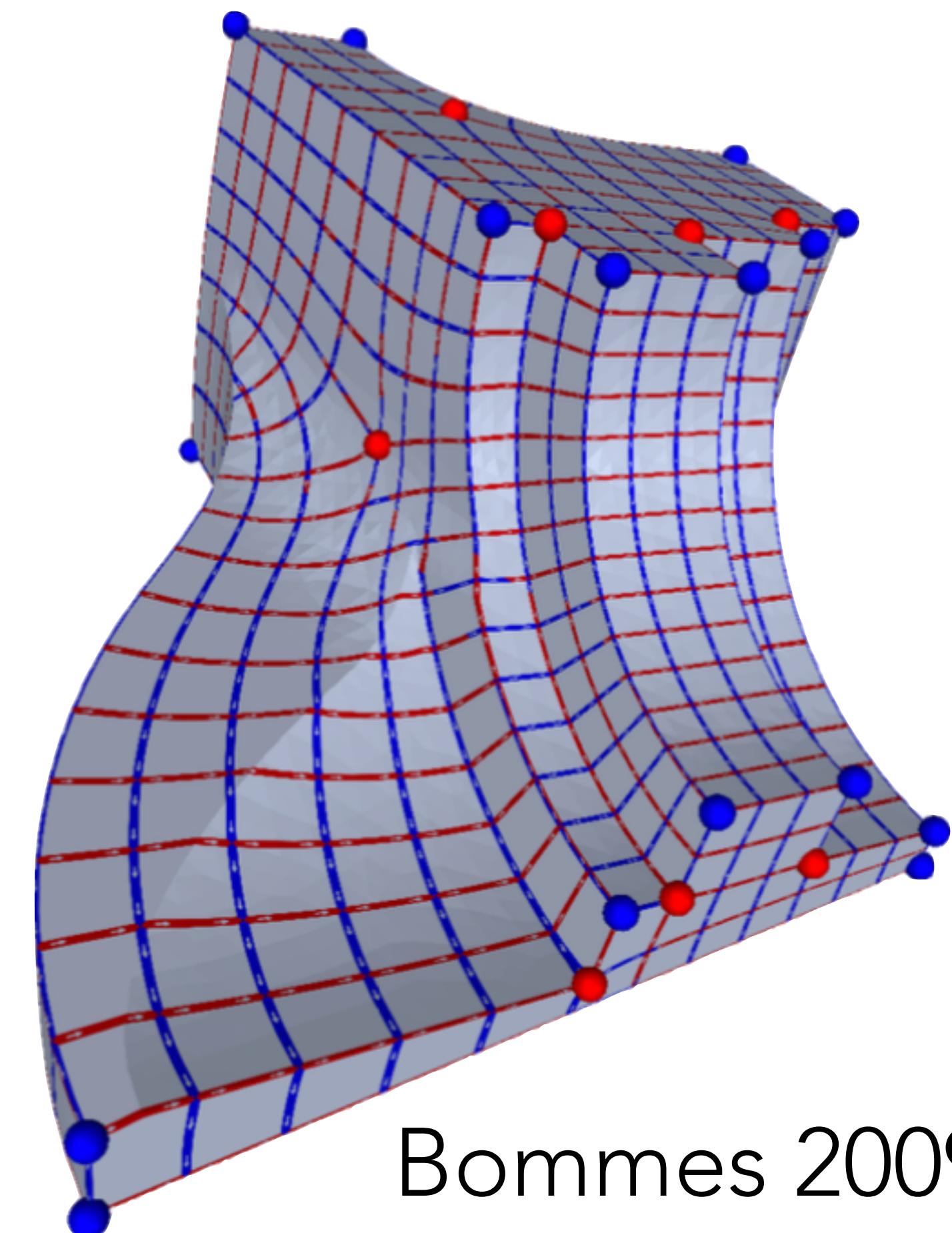


Already implemented in libigl!

See tutorial Chapter 4

# Field-Guided Parametrization

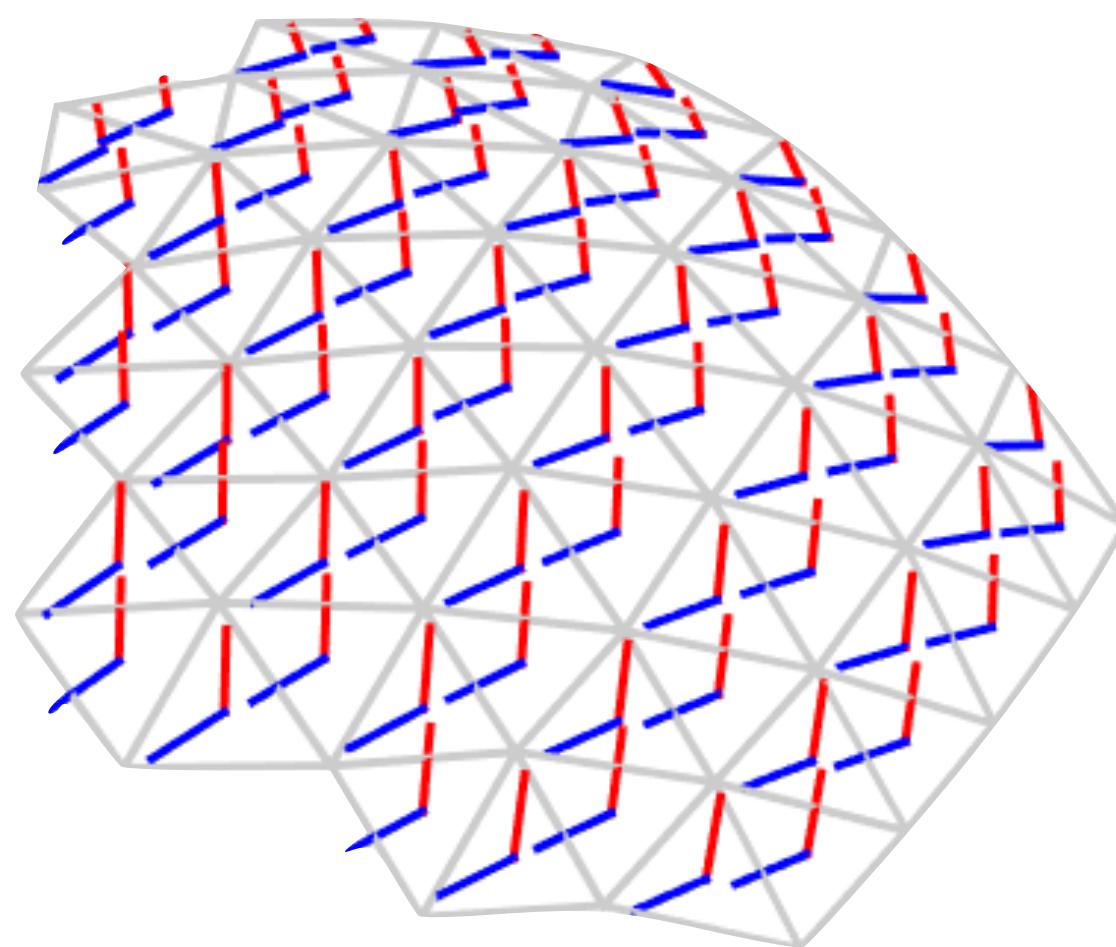
- Often we want coordinate directions to **align** with mesh features.
- We can do this by  
**specifying vector fields** and  
**fitting (u, v) gradients**  
(coordinate directions) to them.
- Orthonormal vectors =>  
minimize **angle and area distortion**



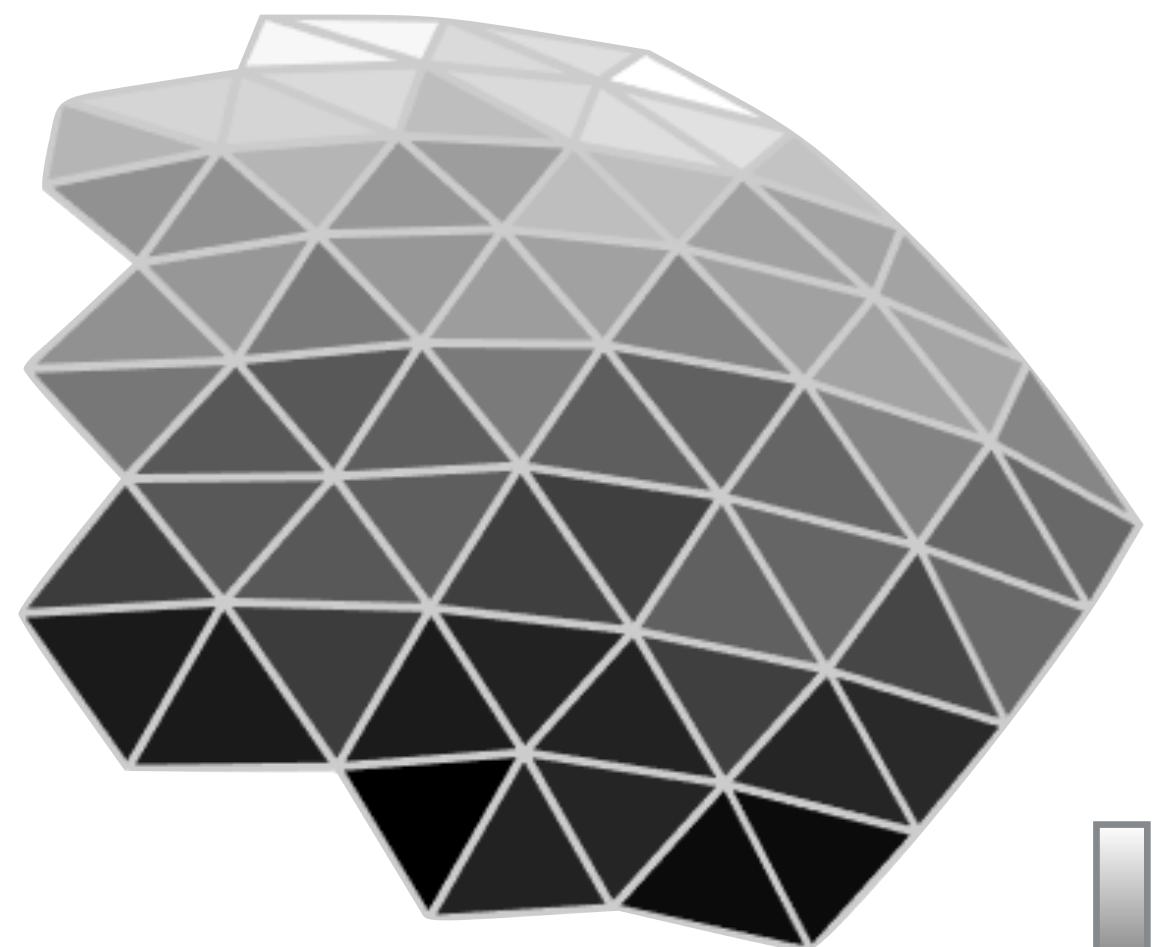
Bommes 2009

# Field-Guided Parametrization

- We'll consider piecewise linear (per-vertex) scalar fields, so gradients and vector fields are constant per triangle:

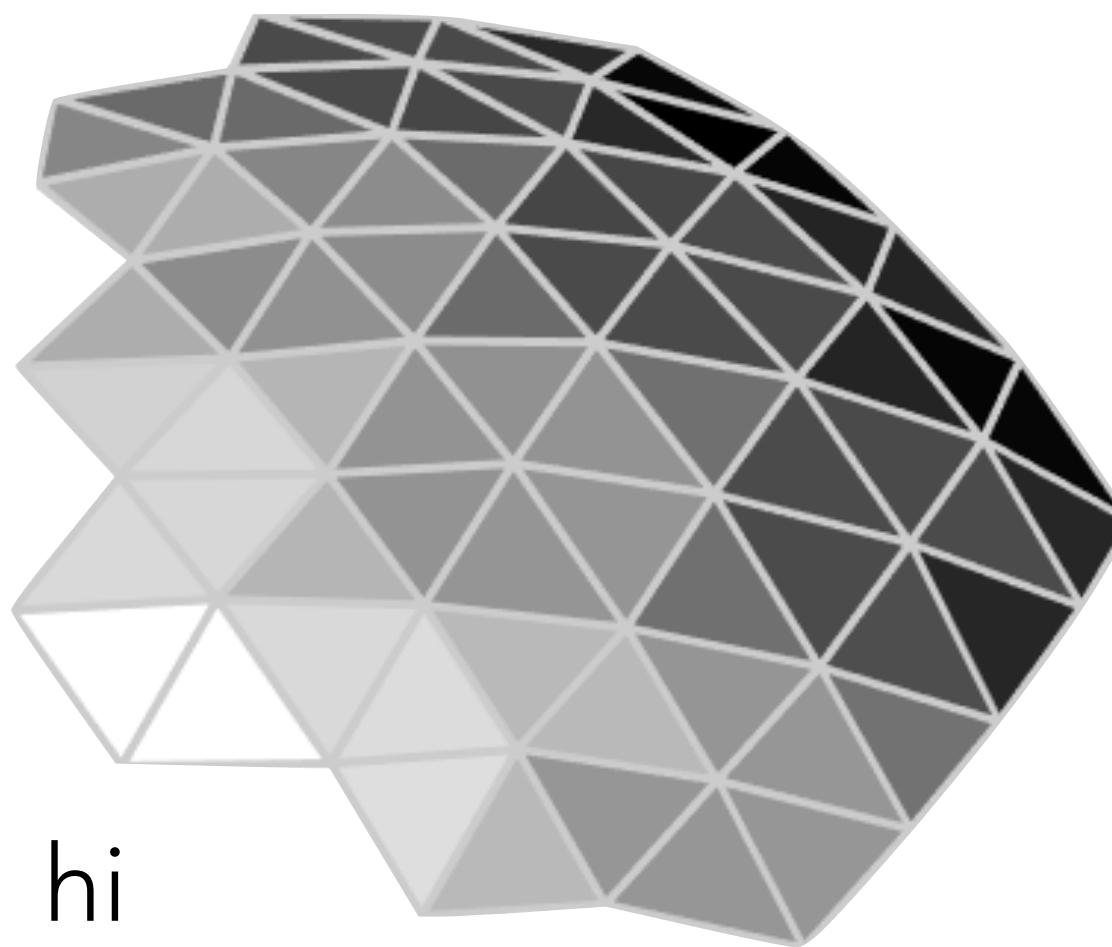


mesh

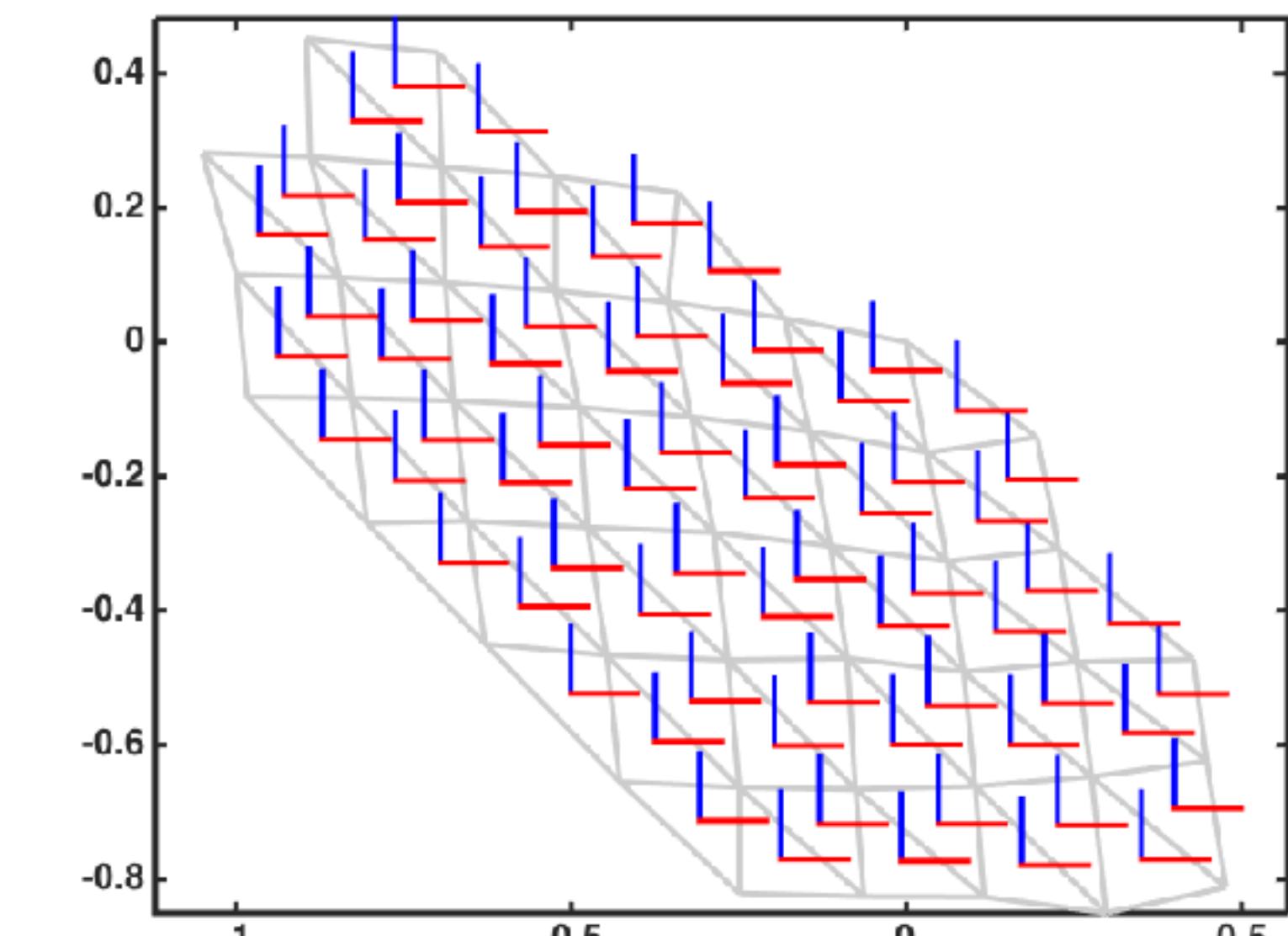


U-function

hi  
lo



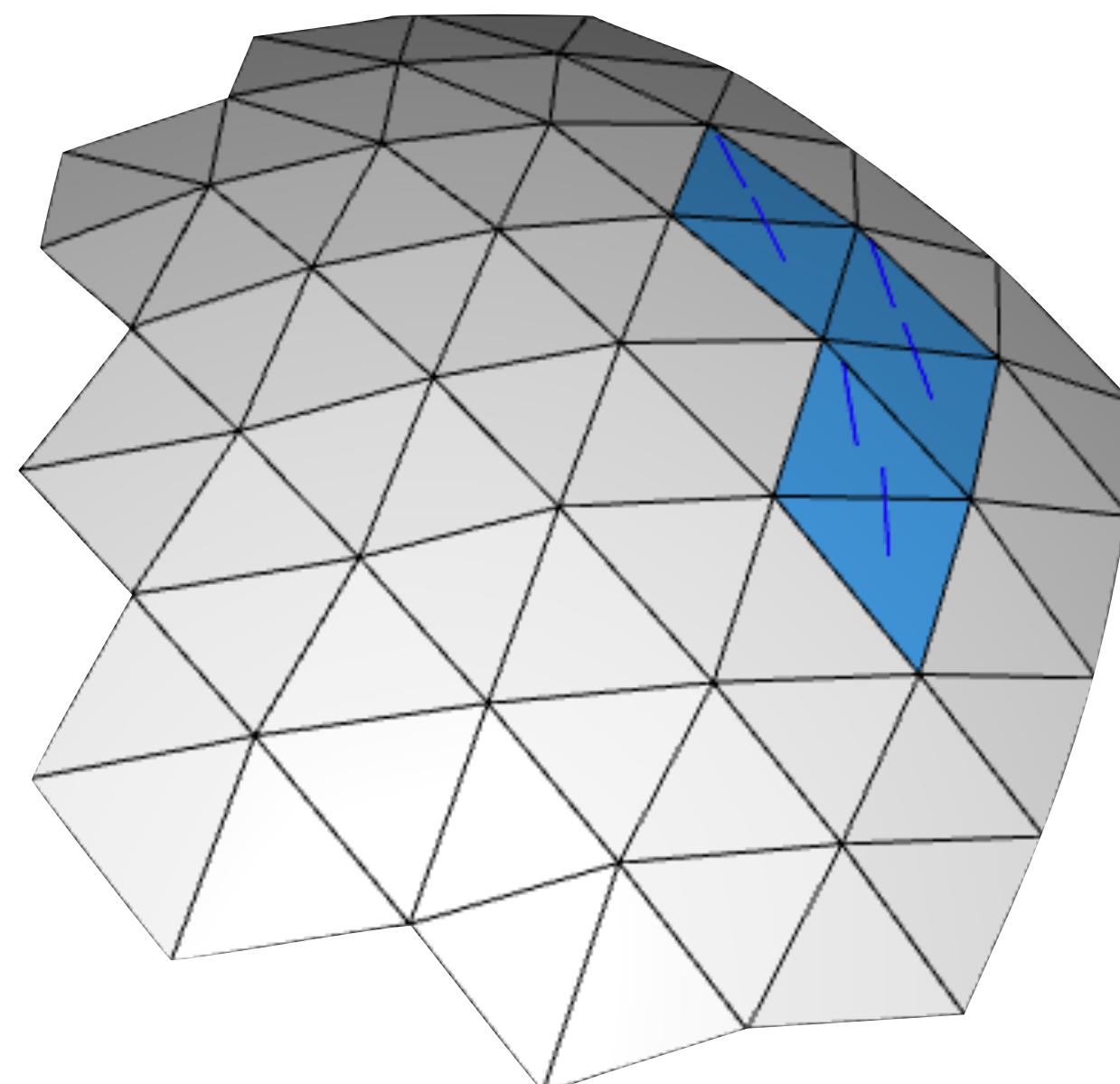
V-function



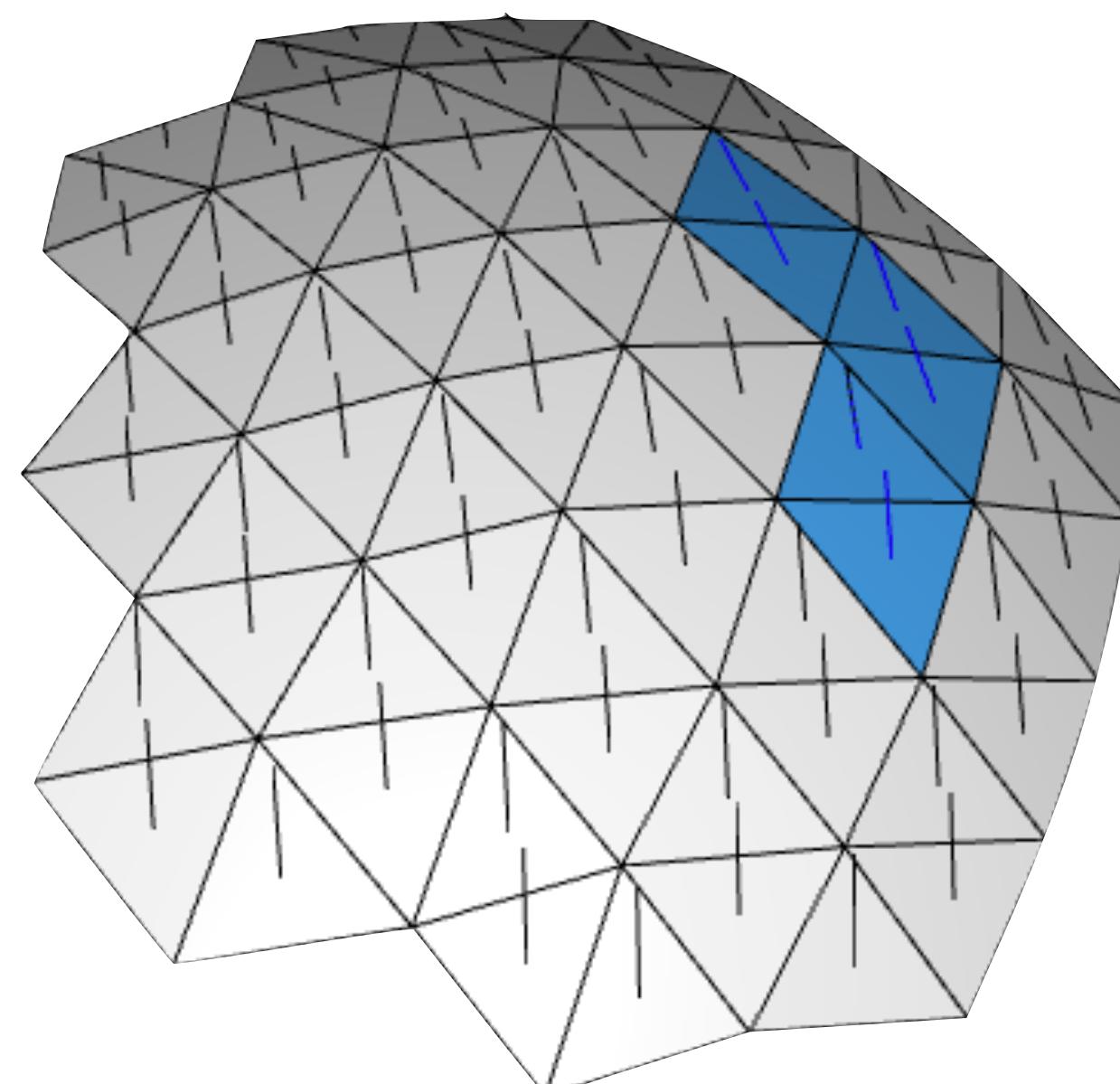
flattened mesh (UV-domain)

# Vector-Guided Scalar Fields

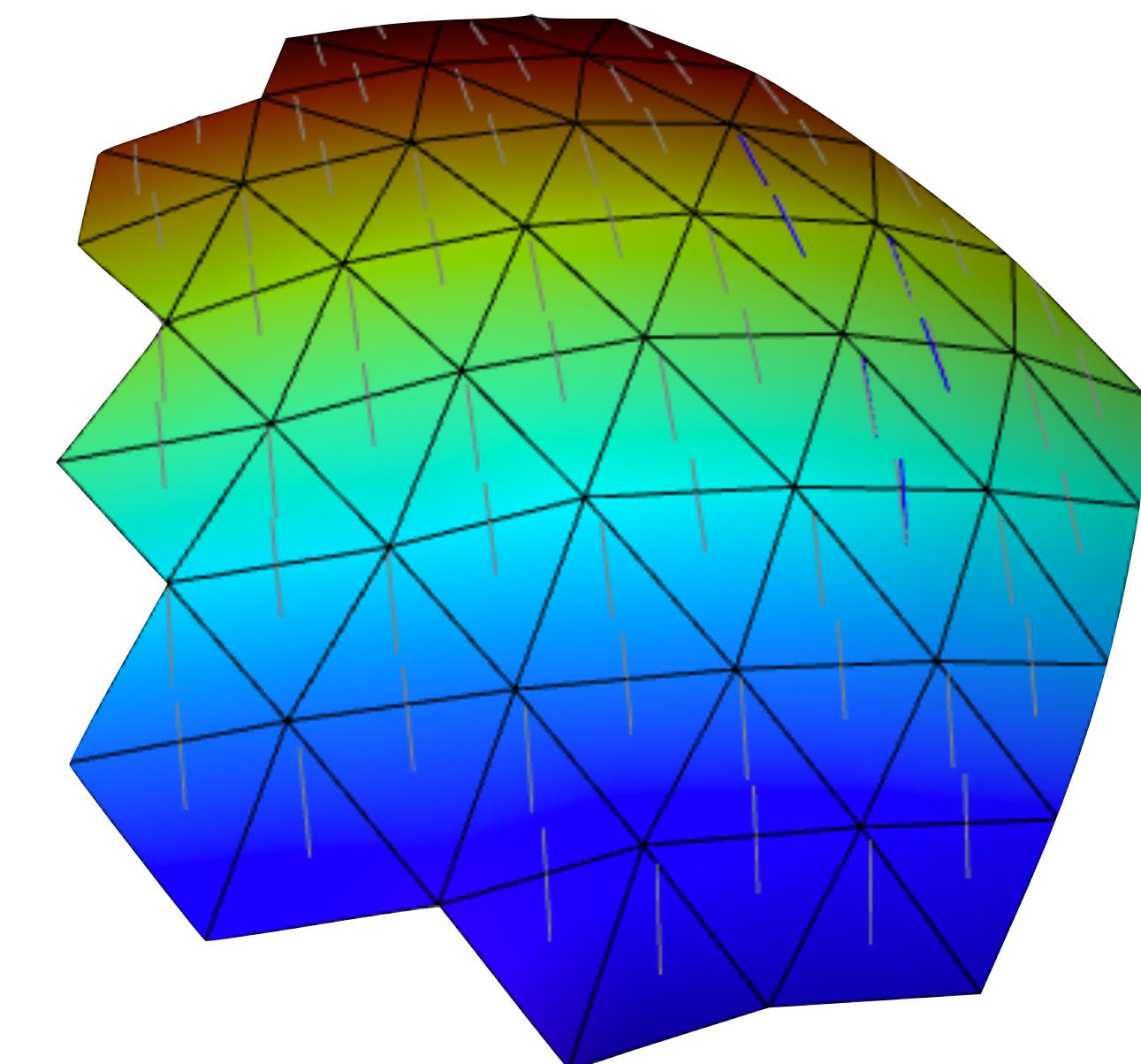
- Main HW4 task: design smooth vector fields and compute a scalar field aligned to them.



vector constraints

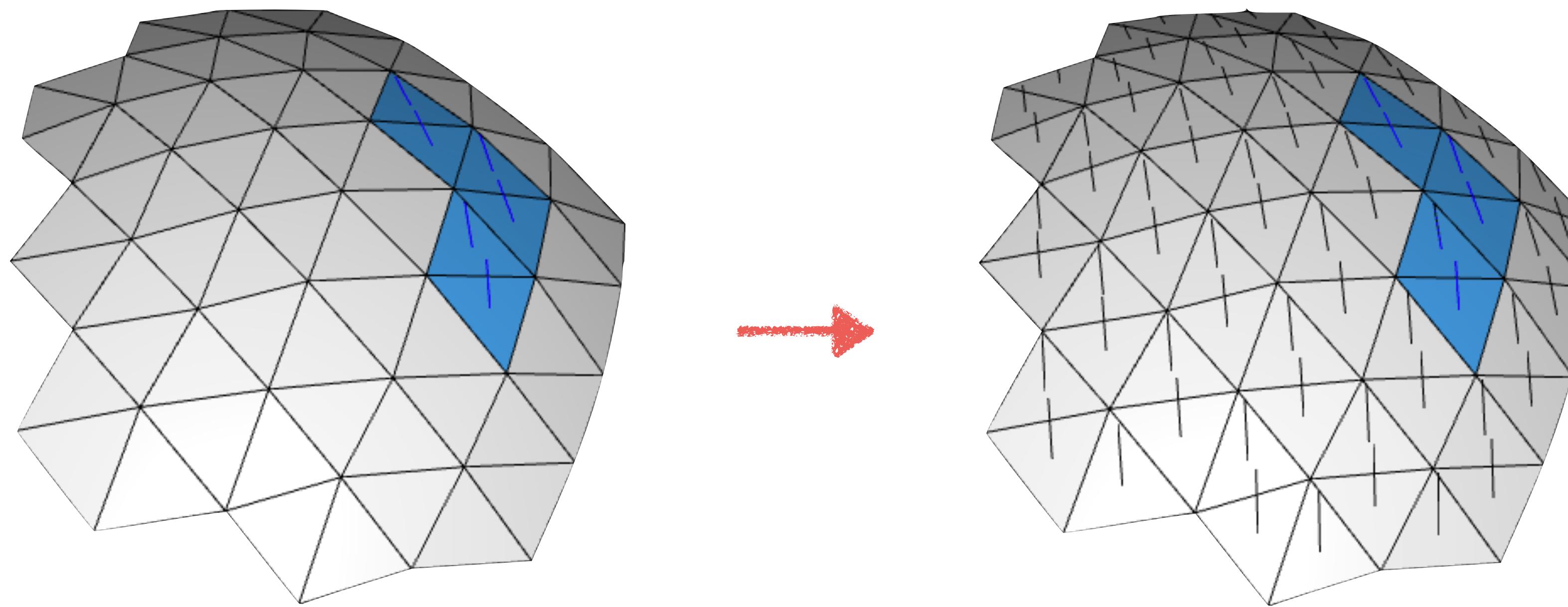


smooth interpolated  
field



reconstructed scalar function  
and gradient

# Step 1: Vector Field Design

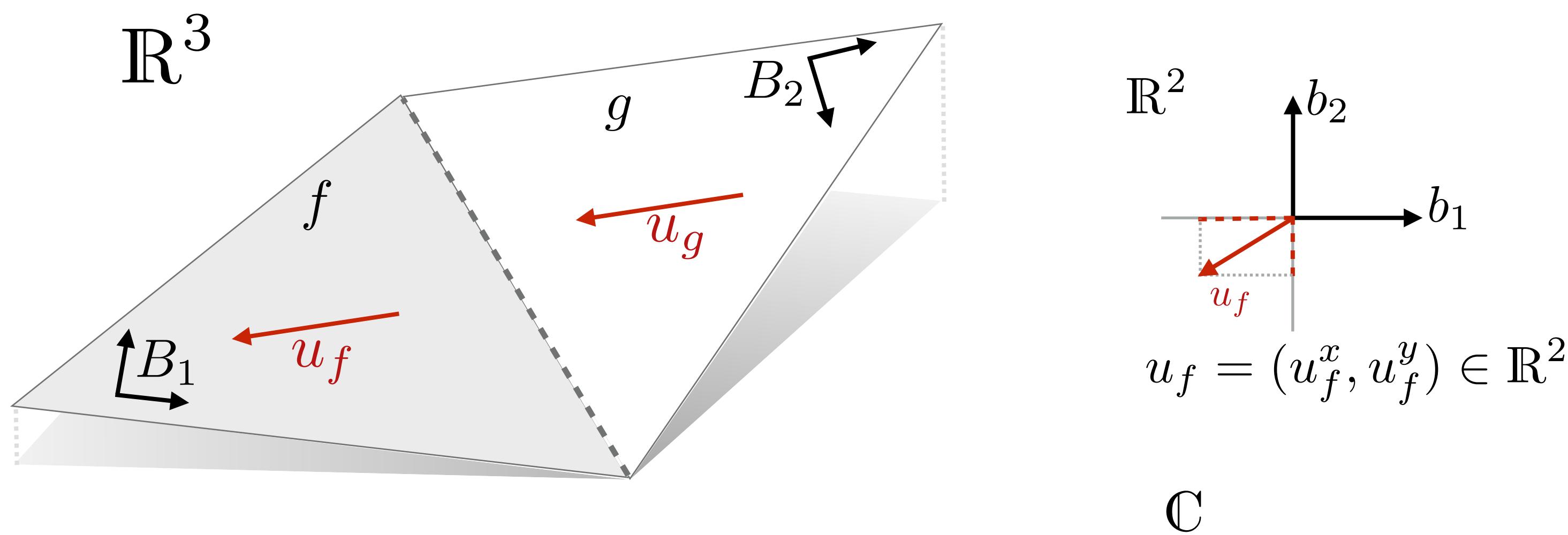


Paint constraint vectors  
on some faces

Smoothly interpolated  
field

# Vector Field Representation

- Tangent vector per triangle:  
represent in an arbitrary local basis for each triangle:

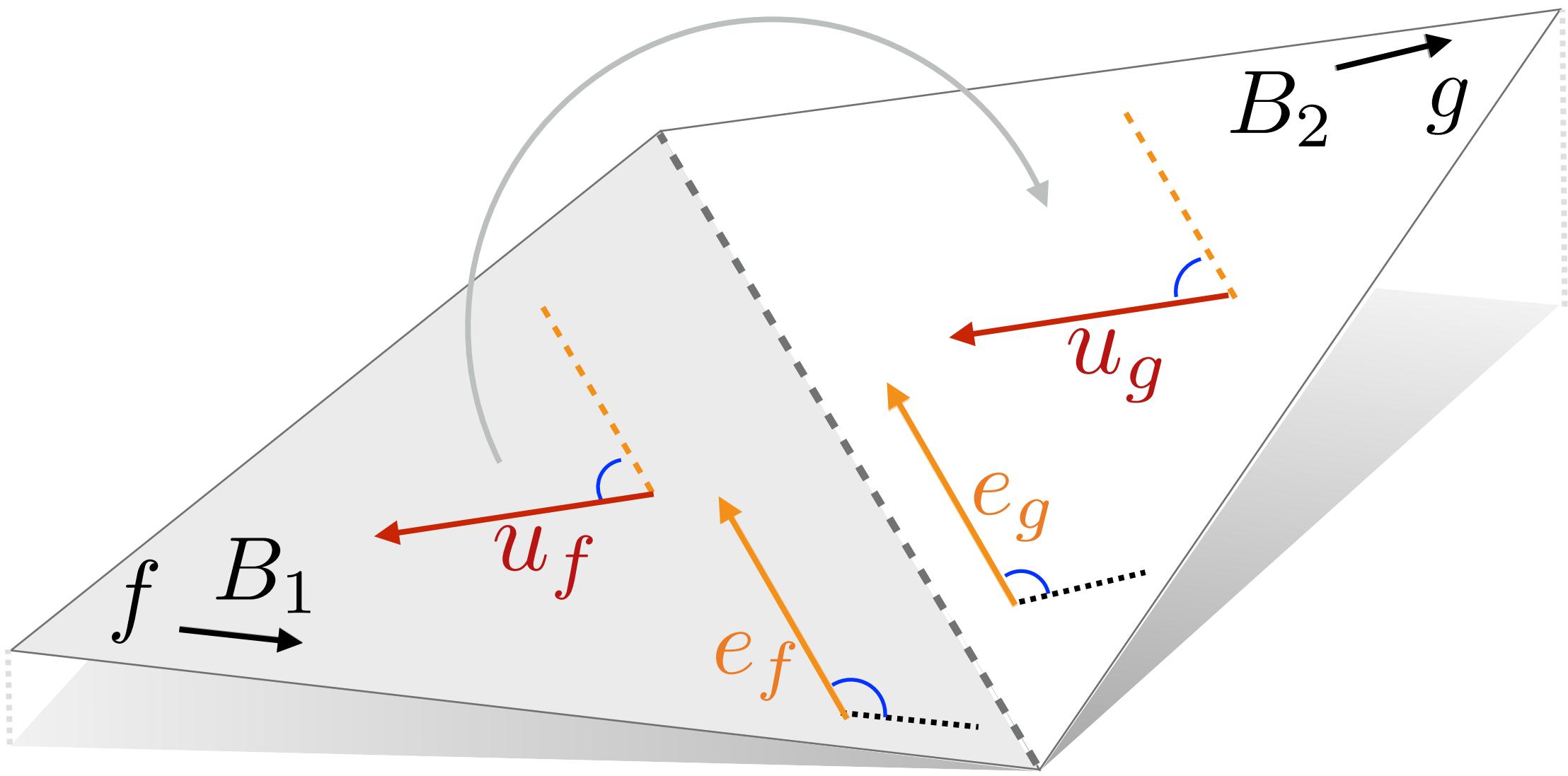


- (Optional) notation simplification:  
Vector = complex number  
(Tangent plane = complex plane)

A diagram of the complex plane  $\mathbb{C}$  with a horizontal "Real Axis". A red vector  $u_f$  is shown originating from the origin, representing a complex number  $u_f = u_f^x + iu_f^y \in \mathbb{C}$ .

# Vector Field Smoothness

- Try to keep vectors constant
- Vectors lie in different tangent planes!  
Must transport to compare.
- Idea: unfold triangle pair along shared edge, compare vectors
- Equivalent: represent vectors in orthonormal basis determined by shared edge.  
(Rotate so shared edge is real axis)



Vectors in  
common basis:

$$\tilde{u}_f = u_f \bar{e}_f$$

$$\tilde{u}_g = u_g \bar{e}_g$$

$$|\tilde{u}_f - \tilde{u}_g|^2$$

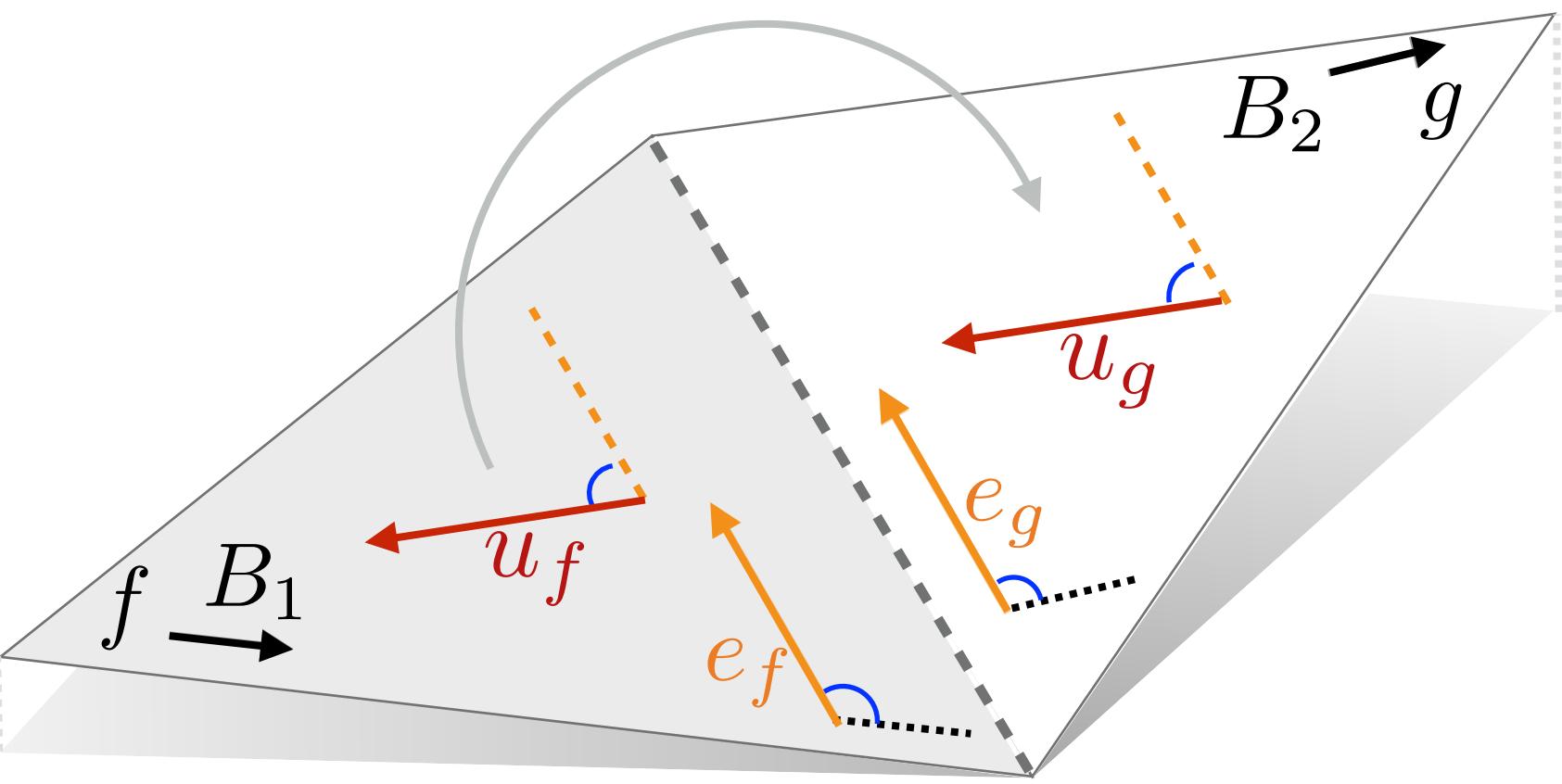
Since for unit  
complex  
numbers:  
 $\bar{e}_f^{-1} = \bar{e}_f$



University  
of Victoria

Computer Science

# Vector Field Smoothness



- Minimize non-smoothness across all edges:

$$\sum_{\text{edge } (f,g)} |u_f \overline{e_f} - u_g \overline{e_g}|^2 = \sum_{\text{edge } (f,g)} [\overline{u_f} \quad \overline{u_g}] Q_{fg} \begin{bmatrix} u_f \\ u_g \end{bmatrix}$$

- Can be re-written as one large complex quadratic form:

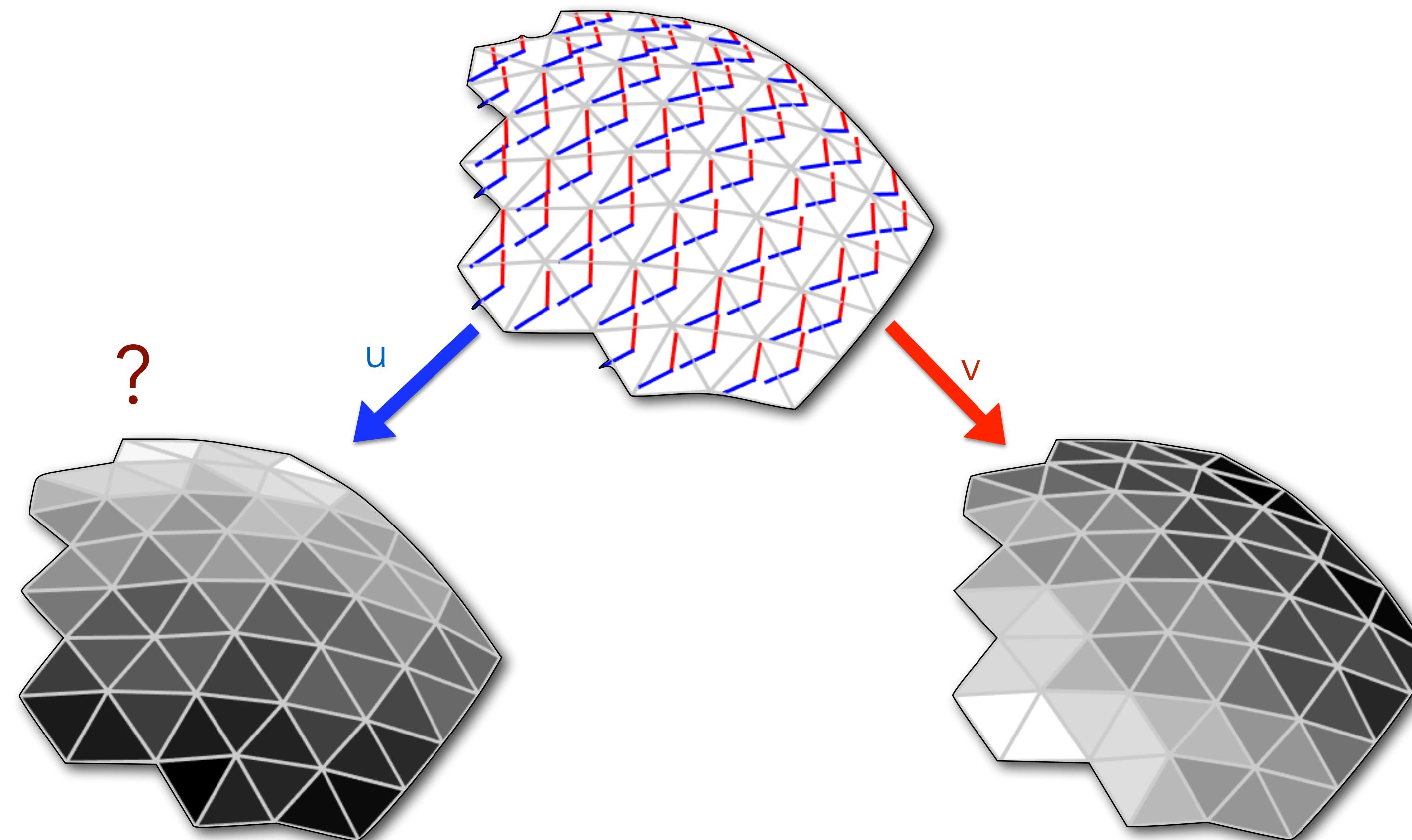
$$\min_{\mathbf{u}|_{cf}=\vec{c}} \mathbf{u}^* Q \mathbf{u} \iff \frac{\partial}{\partial \overline{\mathbf{u}}} \mathbf{u}^* Q \mathbf{u} = Q \mathbf{u} = 0$$

Complex equiv to  
gradient



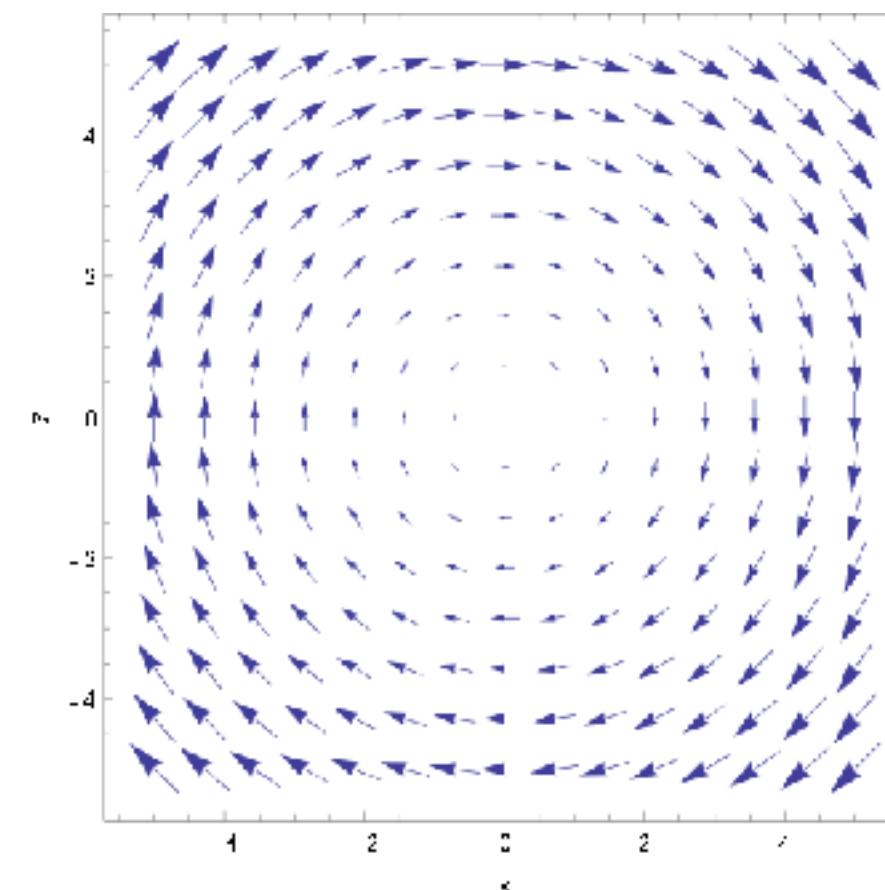
# Step 2: Scalar Field Design

- Find a scalar field whose gradients match the vector field:

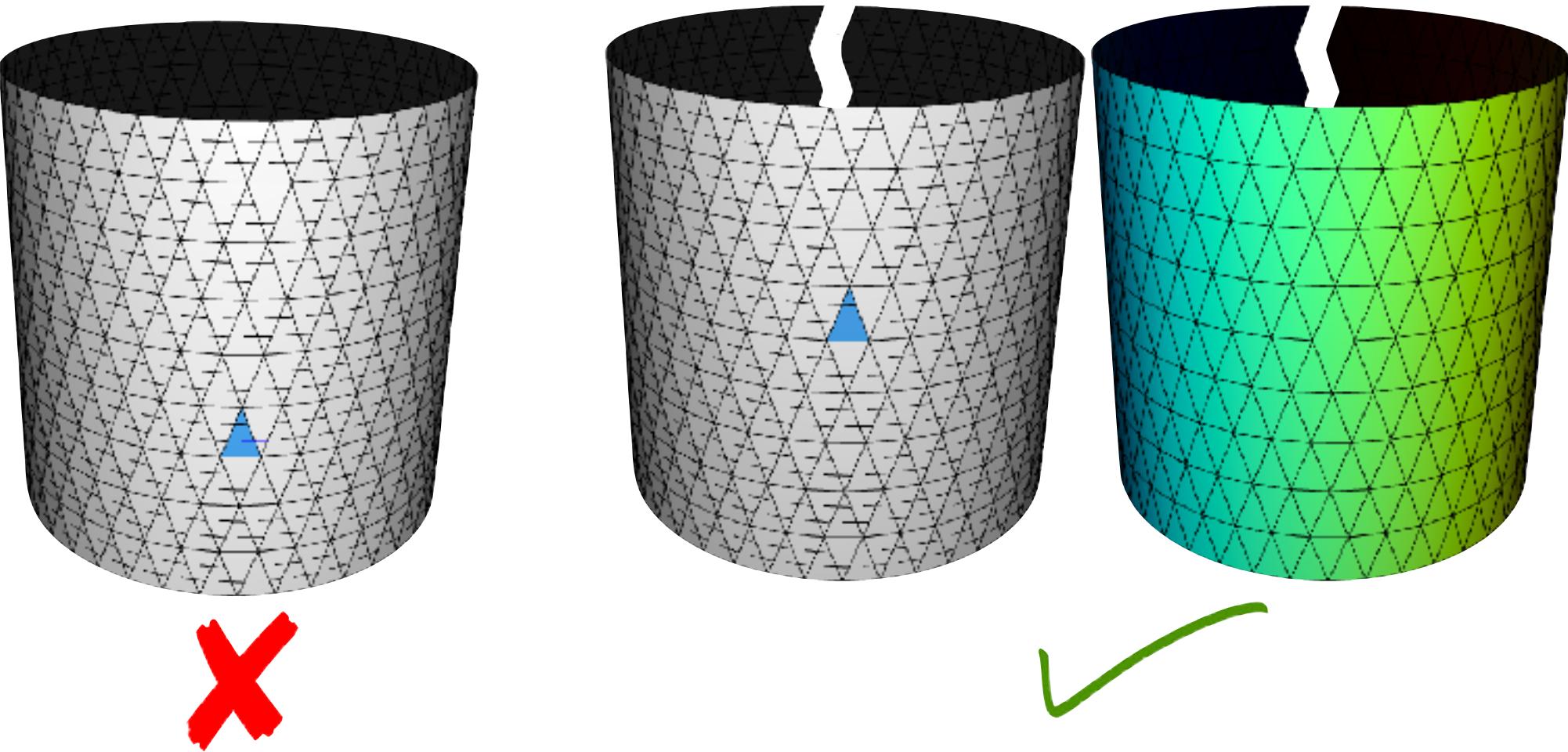


# Integrating the Vector Field

- Why not just assign zero to some vertex and “integrate” the vector field outward to get all other vertex values?
- Problem: most vector fields are not actually integrable (i.e. not the gradient of a scalar field)
- This can happen for local or global reasons:



Local: nonzero curl



Global: non-simply-connected domains

# Integrating the Vector Field

- The fix: integrate in the “least-squares sense”

$$\min \sum_{\text{face } t} w_t \|\mathbf{g}_t - \mathbf{u}_t\|^2 \quad \mathbf{g}_t = \nabla f|_t$$

- This turns out to be equivalent to the Poisson equation:

$$\nabla \cdot \nabla f = \nabla \cdot u \quad \text{in } M$$

$$n \cdot \nabla f = n \cdot u \quad \text{on } \partial M$$

- Can also be written as a quadratic form:

$$\frac{1}{2} \mathbf{f}^T K \mathbf{f} + \mathbf{f}^T \mathbf{b} + c$$

- Only determined up to constant: fix one vertex to zero!

# Expressing as Quadratic Form

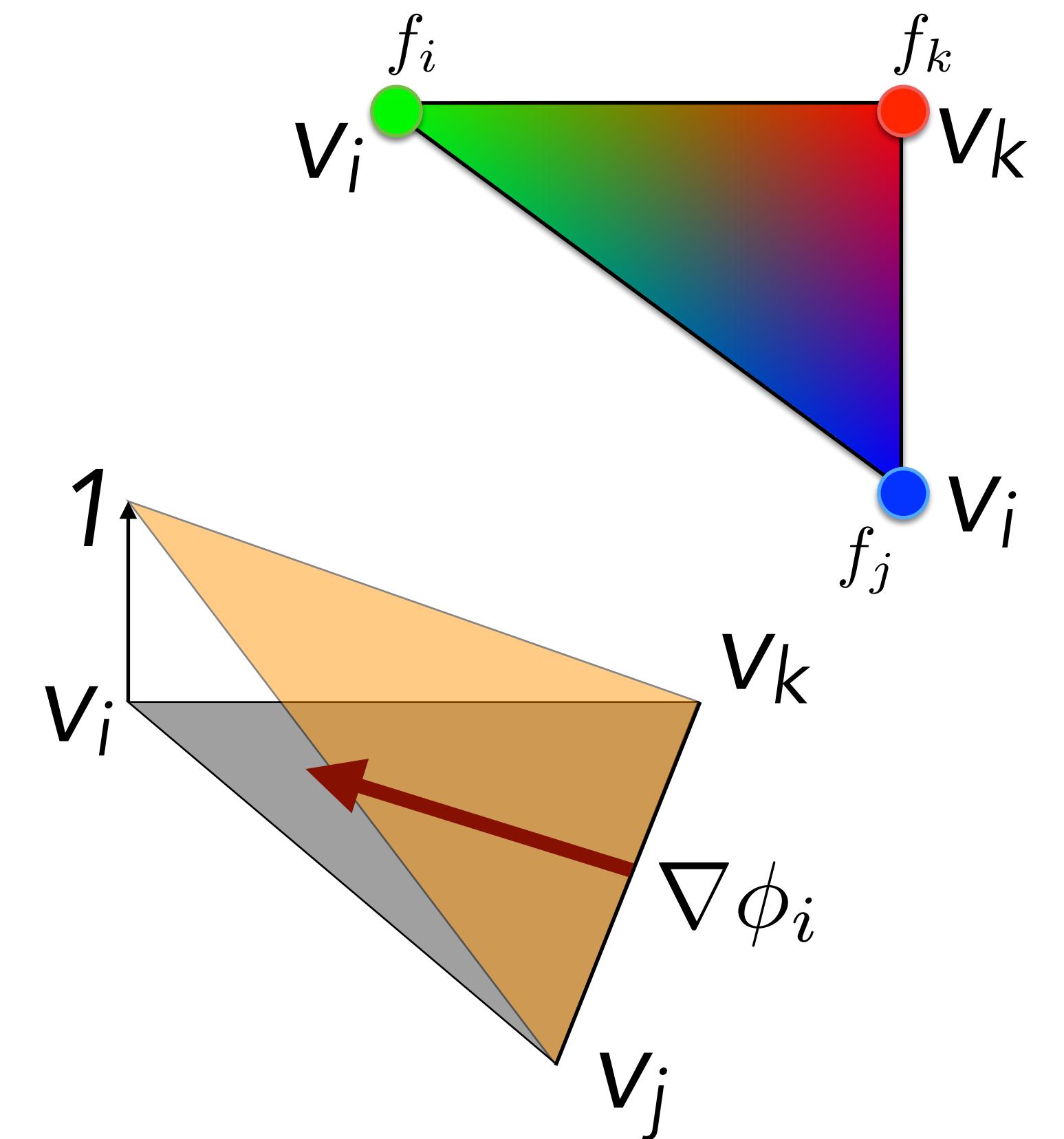
$$\min \sum_{\text{face } t} w_t \|\mathbf{g}_t - \mathbf{u}_t\|^2$$

$$\mathbf{g}_t = \nabla f|_t$$

- Recall, piecewise linear  $f$  can be written as a sum of “hat” basis functions:

$$f(x) = f_i \phi_i(x) + f_j \phi_j(x) + f_k \phi_k(x)$$

$$\nabla f(x) = f_i \nabla \phi_i(x) + f_j \nabla \phi_j(x) + f_k \nabla \phi_k(x)$$



# The “Gradient Matrix”

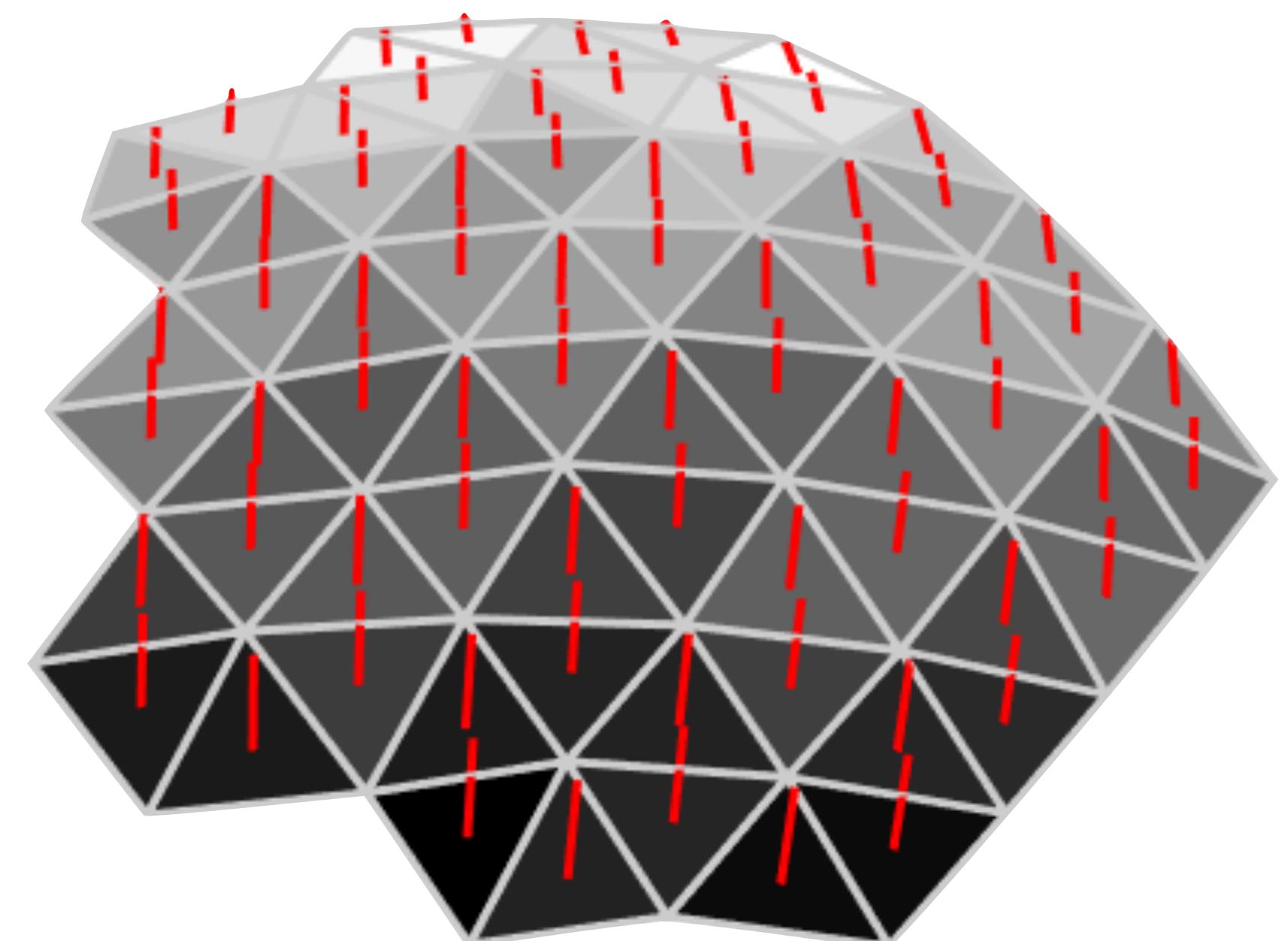
- So the gradient is just a linear combination of vertex scalar field values

$$\begin{bmatrix} \frac{\partial f}{\partial x} \Big|_{t_1} \\ \vdots \\ \frac{\partial f}{\partial x} \Big|_{t_{\#F}} \\ \frac{\partial f}{\partial y} \Big|_{t_1} \\ \vdots \\ \frac{\partial f}{\partial z} \Big|_{t_{\#F}} \end{bmatrix} = G \begin{bmatrix} f|_{v_1} \\ \vdots \\ f|_{v_{\#V}} \end{bmatrix}$$

$3\#F \times 1$        $\#V \times 1$

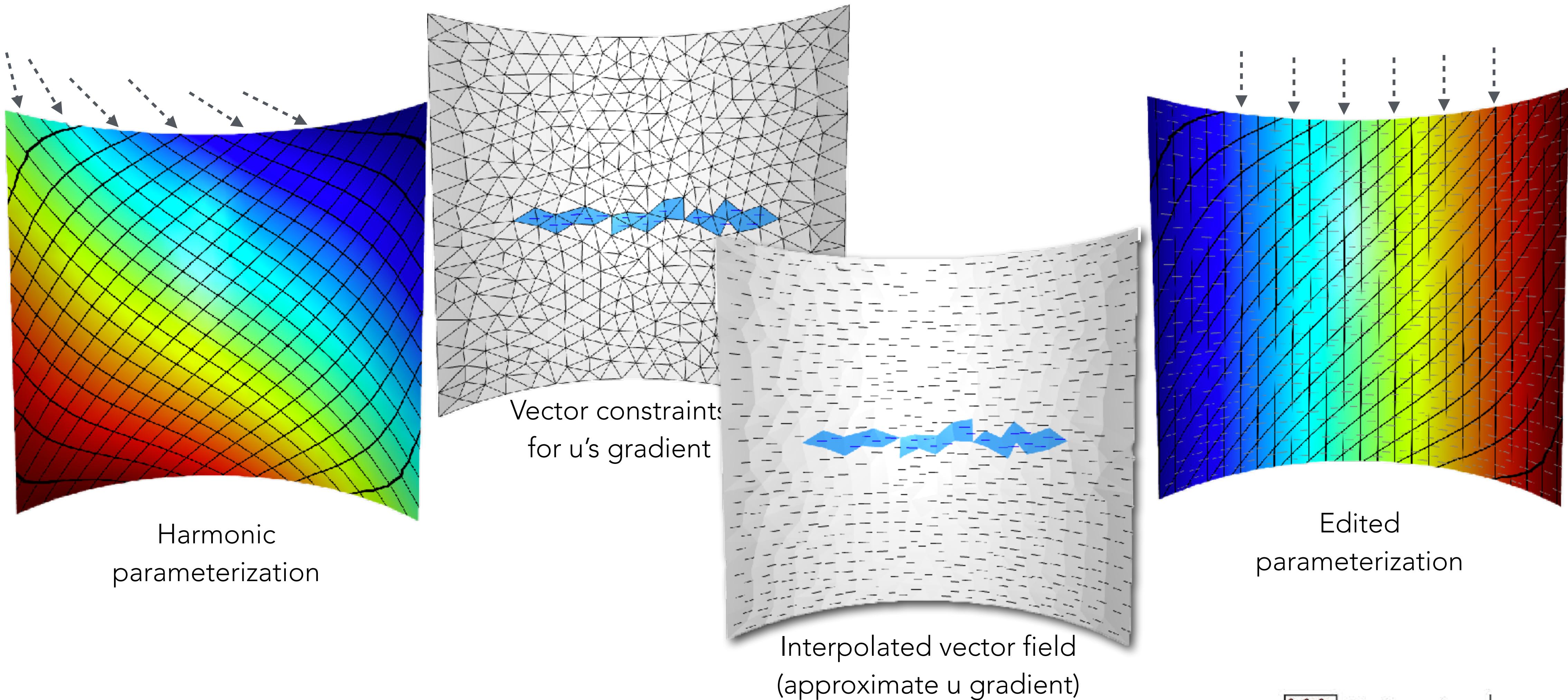
$G$   
 $3\#F \times \#V$

Maps per-vertex scalars  
to per-triangle gradient  
vectors



$G = \text{igl.grad}(v, f)$

# Application: Edit Parameterizations



# Equality Constraints

- Both steps involve equality constraints
  - Painted vectors on constrained faces
  - Scalar field value on single vertex fixed to zero
- Two approaches:
  - Lagrange multipliers (standard approach)
  - Variable elimination (much more efficient here)

# Variable Elimination via Row/Col Removal

- I'll derive this since it's less discussed than Lagrange multipliers
- Most efficient way to fix variables to constants:

$$\min_{\mathbf{x}} \frac{1}{2} \mathbf{x}^T A \mathbf{x} - \mathbf{b}^T \mathbf{x} + c \quad \text{s.t. } \mathbf{x}_c = \mathbf{d}$$
$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_f \\ \mathbf{x}_c \end{bmatrix}$$

Free Variables  
Constrained Variables

- For derivation, assume vars ordered like this;

Trick works for any ordering

Note that the matrix  $A$  is not the same  $A$  as in the code of the assignment.

# Row/Col Removal

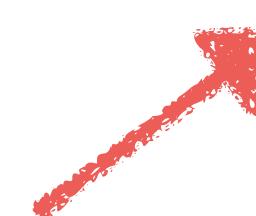
$$\begin{aligned}
 & \frac{1}{2} \mathbf{x}^T A \mathbf{x} - \mathbf{b}^T \mathbf{x} + c \\
 &= \frac{1}{2} \begin{bmatrix} \mathbf{x}_f^T & \mathbf{x}_c^T \end{bmatrix} \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \begin{bmatrix} \mathbf{x}_f \\ \mathbf{x}_c \end{bmatrix} - \begin{bmatrix} \mathbf{b}_f^T & \mathbf{b}_c^T \end{bmatrix} \begin{bmatrix} \mathbf{x}_f \\ \mathbf{x}_c \end{bmatrix} + c \\
 &= \frac{1}{2} (\mathbf{x}_f^T A_{ff} \mathbf{x}_f + 2\mathbf{x}_f^T A_{fc} \mathbf{x}_c + \mathbf{x}_c^T A_{cc} \mathbf{x}_c) - \mathbf{b}_f^T \mathbf{x}_f - \mathbf{b}_c^T \mathbf{x}_c + c
 \end{aligned}$$

$$\min_{\mathbf{x}_f} \implies 0 = A_{ff} \mathbf{x}_f + A_{fc} \mathbf{x}_c - \mathbf{b}_f$$

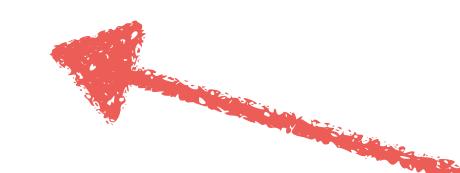
$$\tilde{A} \tilde{\mathbf{x}} = \tilde{\mathbf{b}}$$

$$\tilde{A} = A_{ff} \quad \tilde{\mathbf{x}} = \mathbf{x}_f \quad \tilde{\mathbf{b}} = \mathbf{b}_f - A_{fc} \mathbf{x}_c$$

Rows/cols corresponding to  
fixed variables removed!



Columns corresponding to fixed vars  
contribute to RHS



# Questions?