Shape Modeling and Geometry Processing
Assignment 4 – **Mesh Parameterization**

INTERACTIVE GEOMETRY LAB

11. April 2025

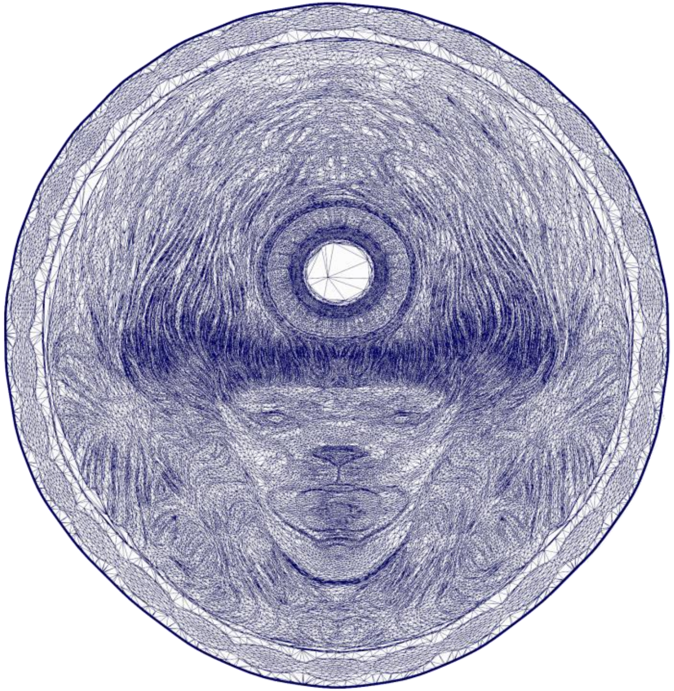
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

What is mesh parameterization

3D space (x, y, z)

$$S(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}$$

2D parameter domain (u, v)

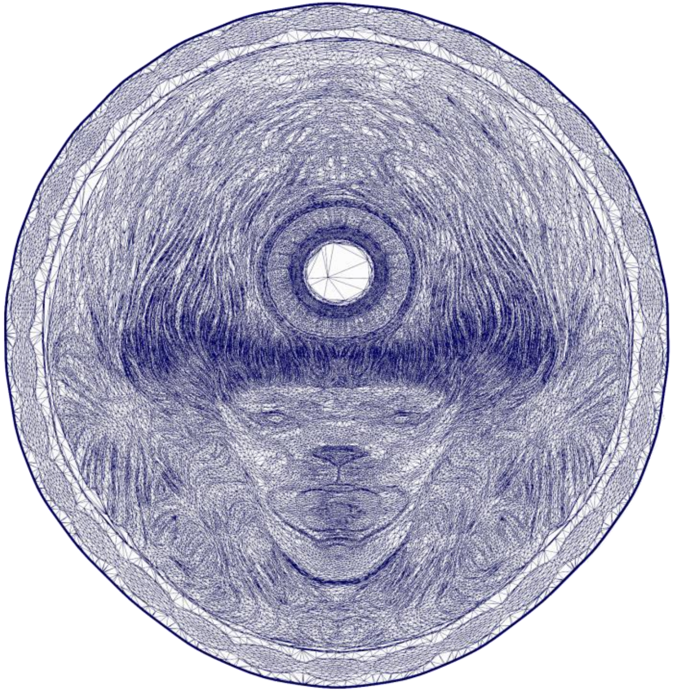


What is mesh parameterization

3D space (x, y, z)

$$f(x_i, y_i, z_i) = (u_i, v_i)$$

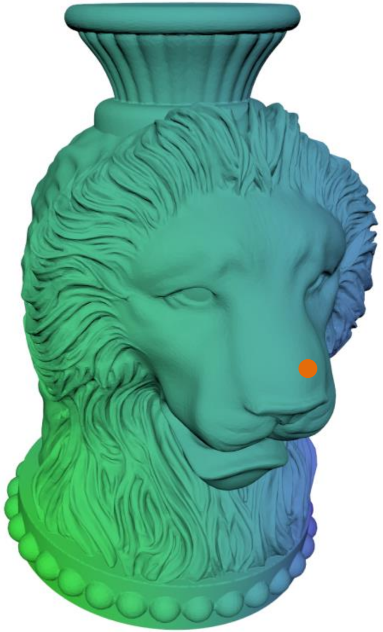
2D parameter domain (u, v)



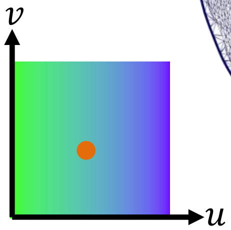
What is mesh parameterization

What is mesh parameterization

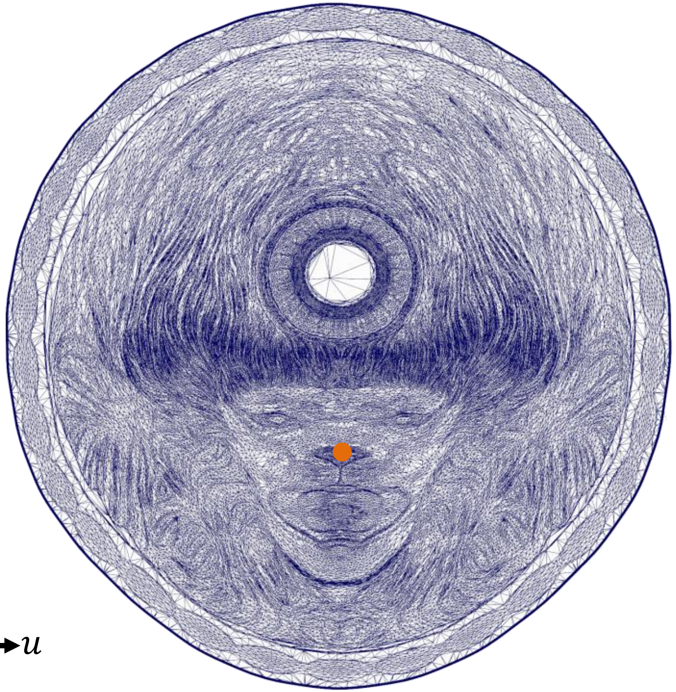
3D space (x, y, z)



$f(x_i, y_i, z_i) = (u_i, v_i)$



2D parameter domain (u, v)

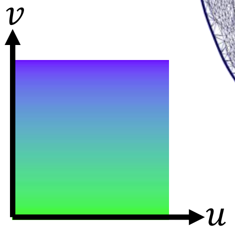


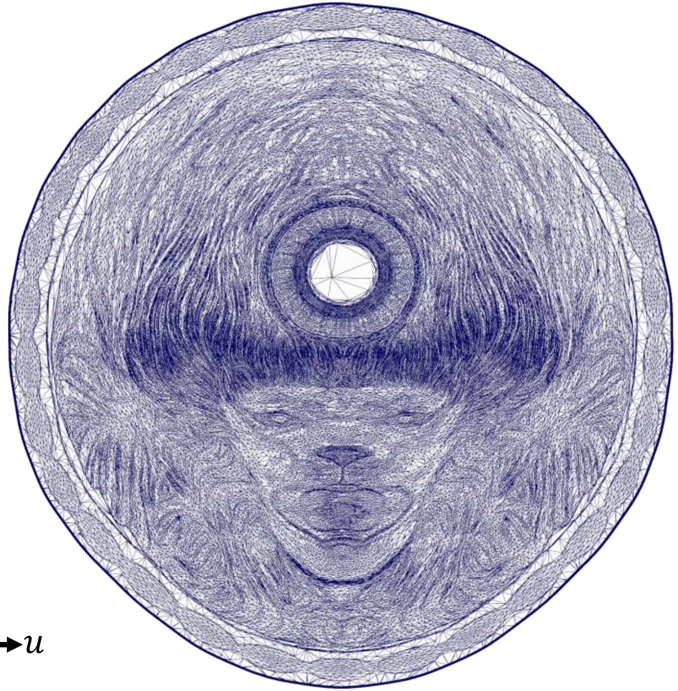
What is mesh parameterization

3D space (x, y, z)

$f(x_i, y_i, z_i) = (u_i, v_i)$

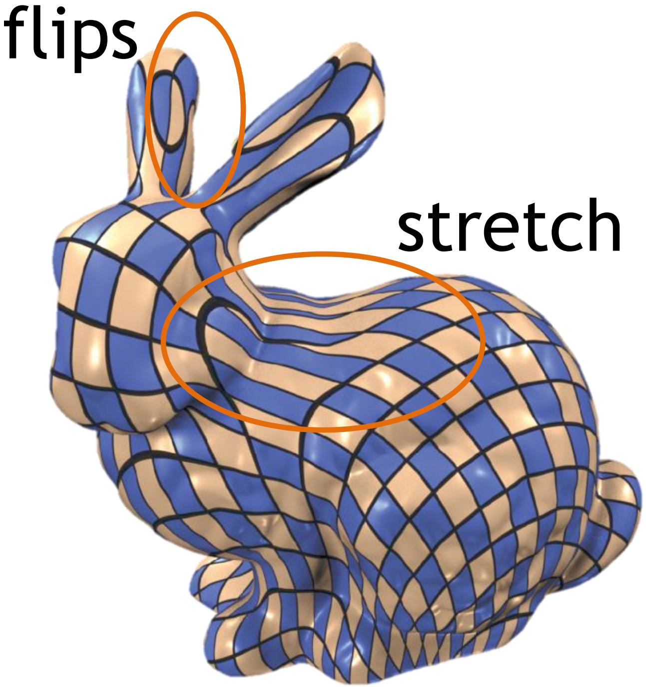
2D parameter domain (u, v)

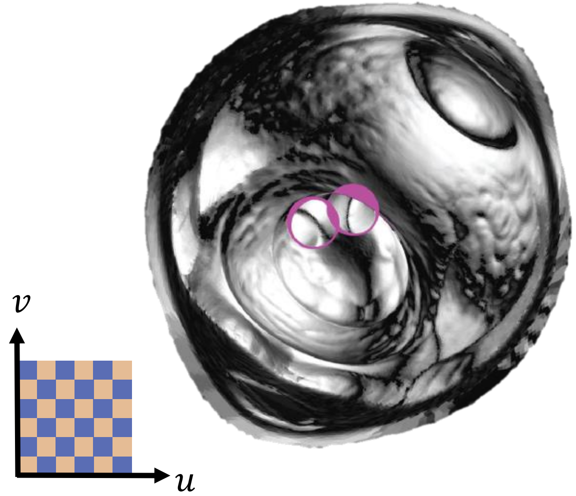




What is a good parameterization

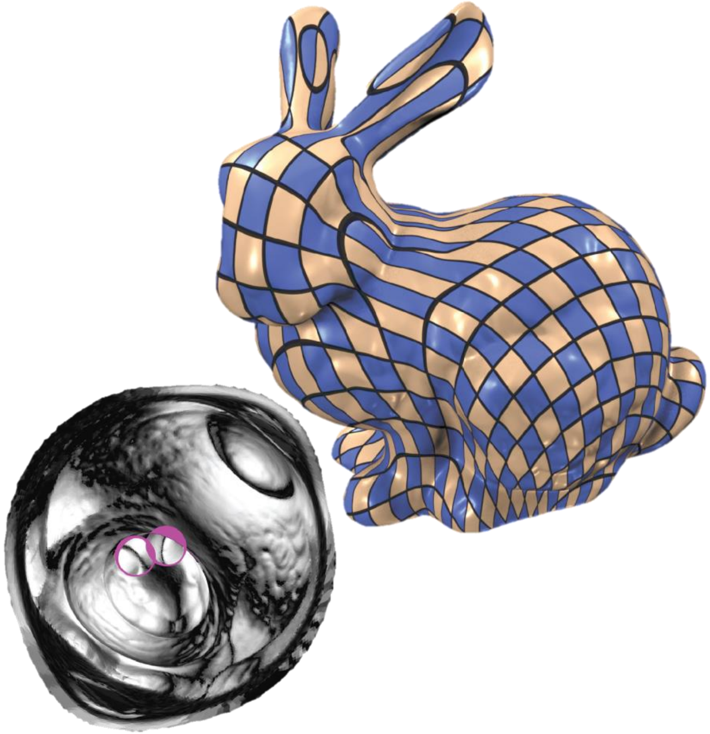
$f(x_i, y_i, z_i) = (u_i, v_i)$

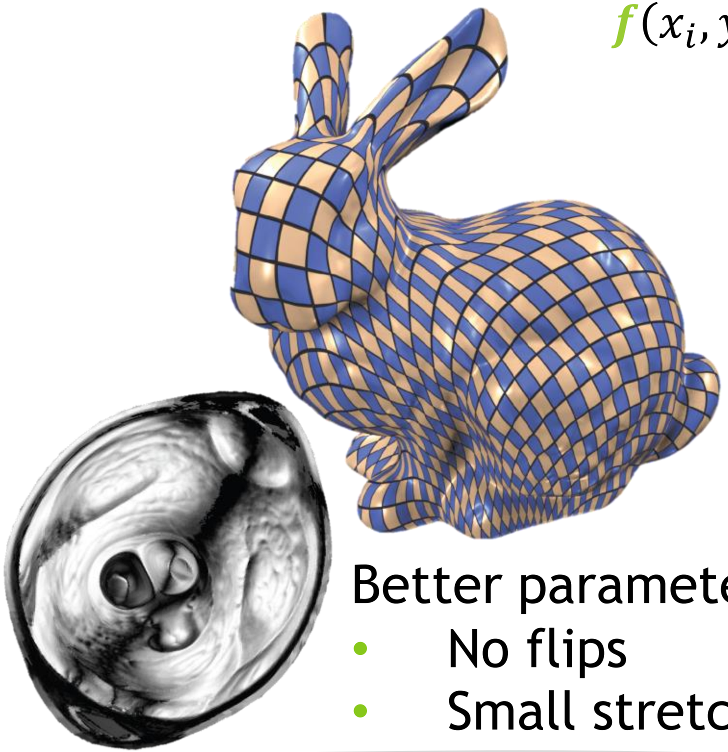




What is a good parameterization

$f(x_i, y_i, z_i) = (u_i, v_i)$





Better parameterization

- No flips
- Small stretch/distortion

How to find a parameterization

- Goal: find parameterization

$$f(x_i, y_i, z_i) = (u_i, v_i)$$

- i.e., for each vertex, find 2D coordinates
- Such that some distortion is minimized
- How to measure distortions?

The Jacobian

- A mapping $f(x,y)\colon \mathrm{R}^2 \to \mathrm{R}^2$ is defined by two functions $u(x,y), v(x,y)$:

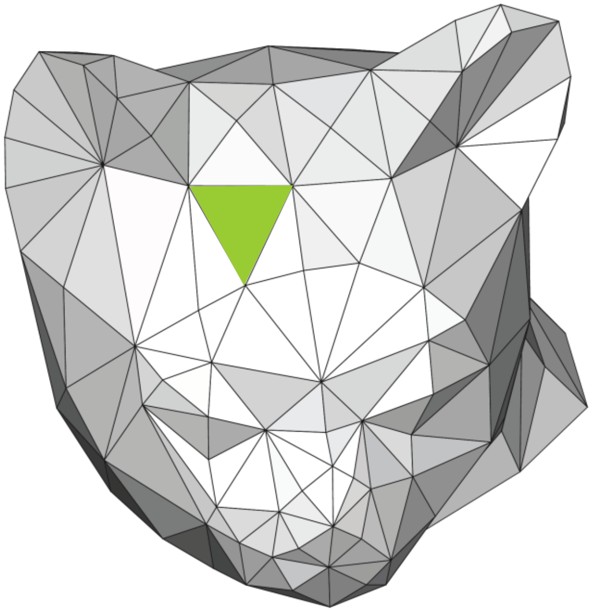
$$f(x,y) = \big(u(x,y), v(x,y)\big)$$

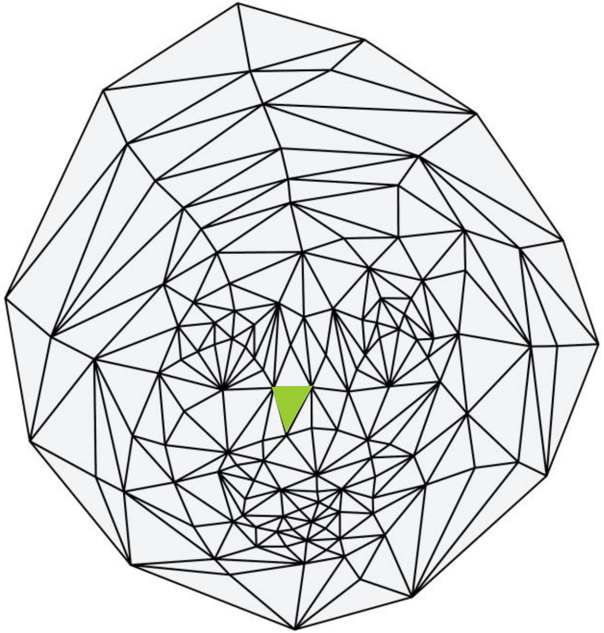
- The Jacobian J is defined by:

$$J = \begin{pmatrix} \nabla_x u & \nabla_y u \\ \nabla_x v & \nabla_y v \end{pmatrix} = \begin{pmatrix} \nabla u \\ \nabla v \end{pmatrix}$$

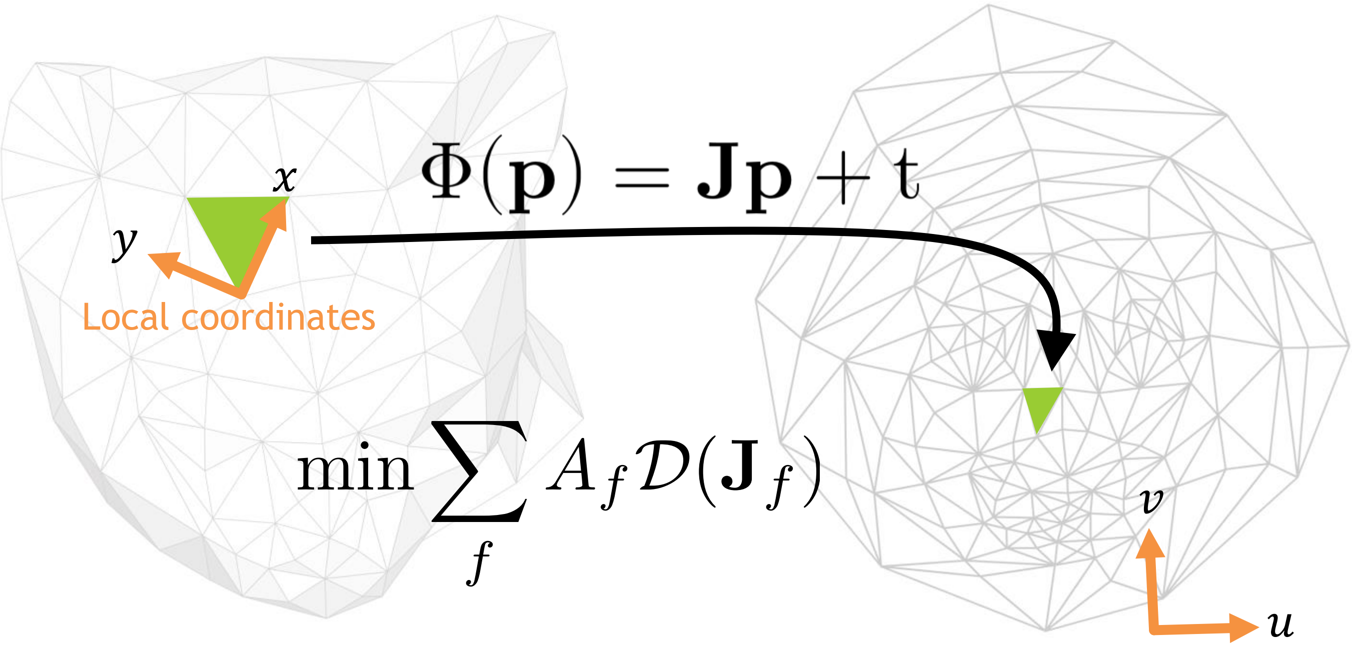
→ The Jacobian demonstrates the distortion

Distortion Measure





Distortion Measure



Distortion Types

- Conformal - angle preserving

$$\mathcal{D}(J) = \|J + J^T - tr(J)I\|_F^2$$

- Isometric - length preserving

$$\mathcal{D}(J) = min_{R \in SO_2} \|J - R\|_F^2$$

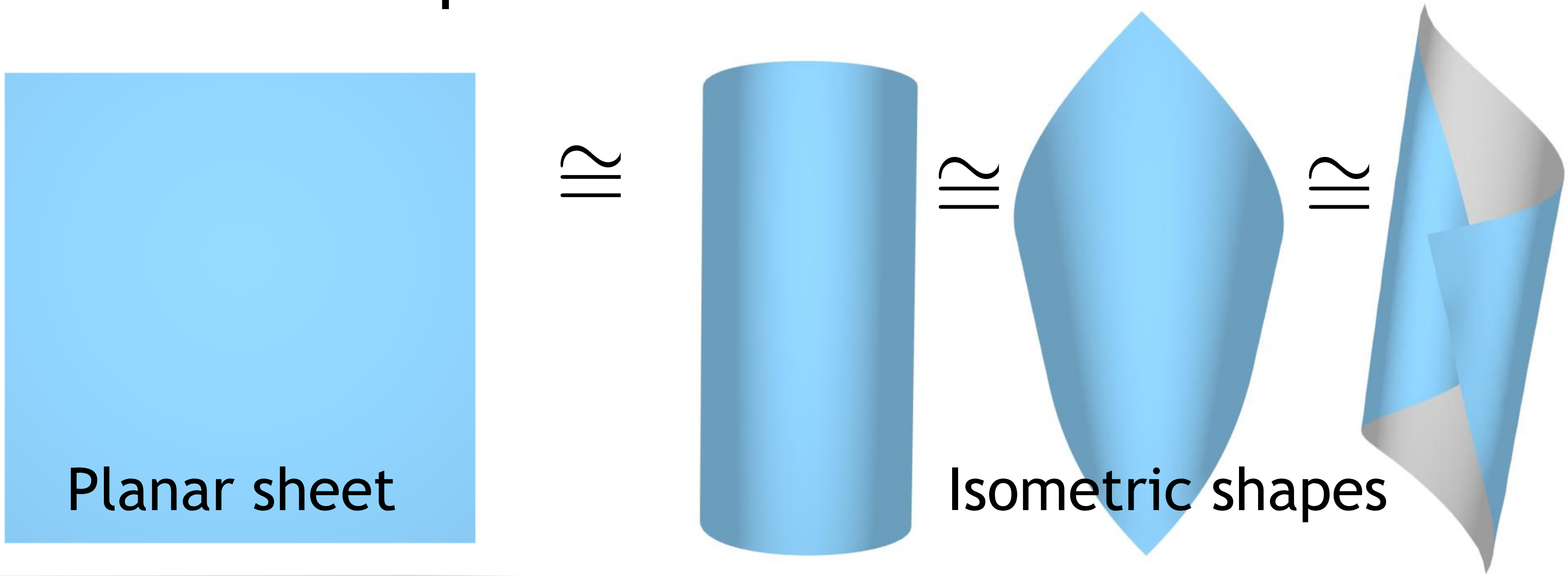
- Authalic - area preserving

$$\mathcal{D}(J) = (\det J \ - 1)^2$$

...and many more!

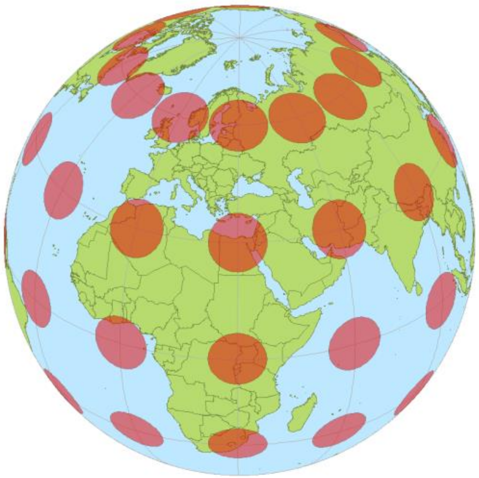
Disclaimer: Isometric Maps

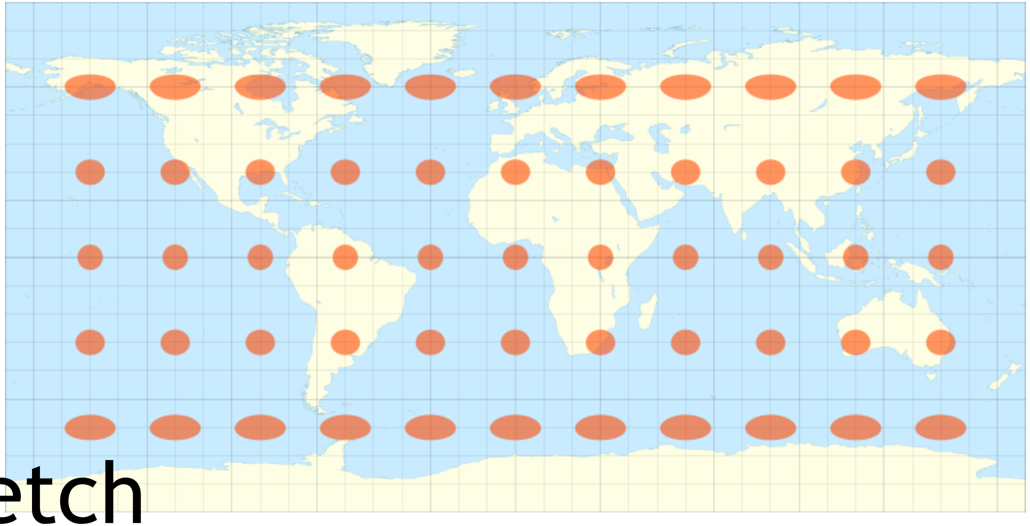
- Isometries preserves Gaussian curvature



Disclaimer: Isometric Maps

- No isometric map to the plane if non-zero Gaussian curvature !

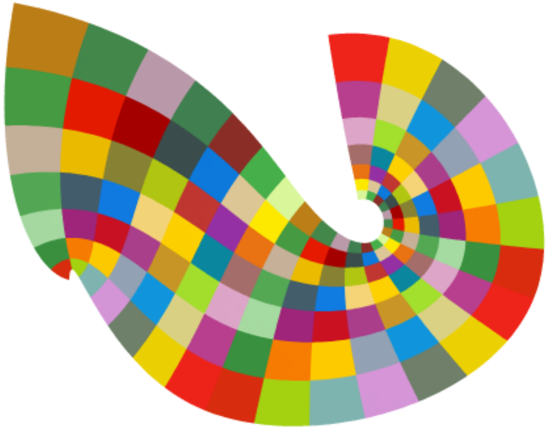




Stretch

Distortion Types

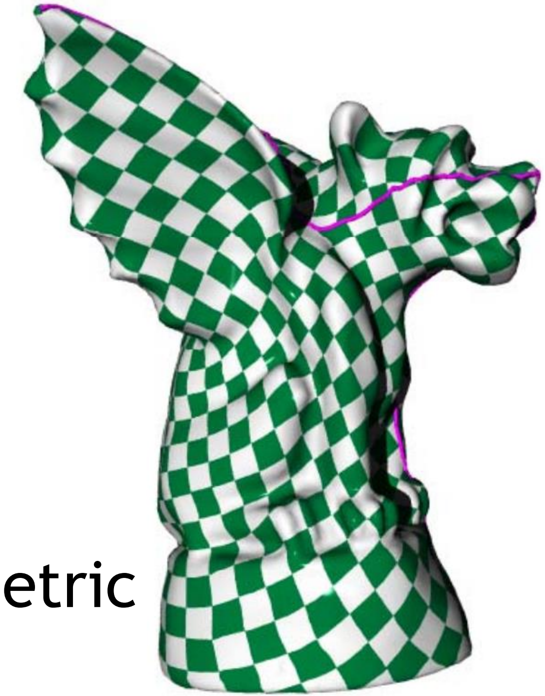
Conformal



Isometric

Distortion Types

Conformal



Isometric

Parameterization

- Goal: We want to solve for the parameterization u, v by minimizing energies defined on $J = \begin{pmatrix} \nabla_x u & \nabla_y u \\ \nabla_x v & \nabla_y v \end{pmatrix}$
- Solution: rewrite the gradient operator ∇ as a linear transformation i.e., $\nabla_x f = D_x f$, $\nabla_y f = D_y f$
- We then reformulate the problem as a least-square problem

$$\min_{u,v} \left\| \mathcal{A} \begin{pmatrix} u \\ v \end{pmatrix} - b \right\|^2$$

  - where $\mathcal{A}$ is constructed from the gradient operator (matrix) D_x, D_y

Example: Dirichlet Energy

Example: Dirichlet Energy

- Goal: $\min_{u,v} \sum_f A_f \|J_f\|_F^2$
- Assume we know the matrices D_x and D_y (defined per-face)

Check *libigl 204*!

$$J_f = \begin{pmatrix} D_x u & D_y u \\ D_x v & D_y v \end{pmatrix}$$

- $\sum_f A_f \|J_f\|_F^2 = \|A^{0.5}(D_x u, D_y u, D_x v, D_y v)\|_F^2$
- $A = diag(A_1, \cdots, A_{n_f})$

Example: Dirichlet Energy

$$\sum_f A_f \|J_f\|_F^2 = \|A^{0.5}(D_x u, D_y u, D_x v, D_y v)\|_F^2$$

$$= \left\| \begin{pmatrix} A^{0.5} D_x & 0 \\ A^{0.5} D_y & 0 \\ 0 & A^{0.5} D_x \\ 0 & A^{0.5} D_y \end{pmatrix} \binom{u}{v} \right\|^2 = \left\| \mathcal{A} \binom{u}{v} \right\|^2$$

Example: Dirichlet Energy

$$\min_{u,v} \left\| \mathcal{A} \begin{pmatrix} u \\ v \end{pmatrix} \right\|^2 \rightarrow \text{solve } \mathcal{A}^T \mathcal{A} \begin{pmatrix} u \\ v \end{pmatrix} = 0$$

$$\mathcal{A}^T \mathcal{A} = \begin{pmatrix} D_x^T A D_x + D_y^T A D_y & 0 \\ 0 & D_x^T A D_x + D_y^T A D_y \end{pmatrix}$$

Example: Dirichlet Energy

$$\mathcal{A}^T\mathcal{A} = \begin{pmatrix} D_x^T A D_x + D_y^T A D_y & 0 \\ 0 & D_x^T A D_x + D_y^T A D_y \end{pmatrix}$$

$$= \begin{pmatrix} L & 0 \\ 0 & L \end{pmatrix}$$

Boundary Conditions

Boundary Conditions

- In the assignment you are asked to fix the boundary of the parameterization to a disc, or to only a few fixed vertices. These constraints can be specified by a sparse linear system

$$C\begin{pmatrix} u \\ v \end{pmatrix} = d$$

Boundary Conditions

$$\min_{u,v} \left\| \mathcal{A} \begin{pmatrix} u \\ v \end{pmatrix} - \mathrm{b} \right\|^2$$

$$\text{s.t. } \mathcal{C} \begin{pmatrix} u \\ v \end{pmatrix} = d$$

Boundary Conditions

$$\min_{u,v} \left\| \mathcal{A} \begin{pmatrix} u \\ v \end{pmatrix} - \mathrm{b} \right\|^2$$

$$\text{s.t. } \mathcal{C} \begin{pmatrix} u \\ v \end{pmatrix} = d$$

Lagrange Multiplier Method

$$\min_{u,v,\lambda} \left\| \mathcal{A} \begin{pmatrix} u \\ v \end{pmatrix} - \mathrm{b} \right\|^2 + \lambda^T \left(\mathcal{C} \begin{pmatrix} u \\ v \end{pmatrix} - d \right)$$

Boundary Conditions

$$\min_{u,v,\lambda} \left\| \mathcal{A} \begin{pmatrix} u \\ v \end{pmatrix} - \mathrm{b} \right\|^2 + \lambda^T \left(\mathcal{C} \begin{pmatrix} u \\ v \end{pmatrix} - d \right)$$

Compute gradient w.r.t u, v, λ, and set to zero

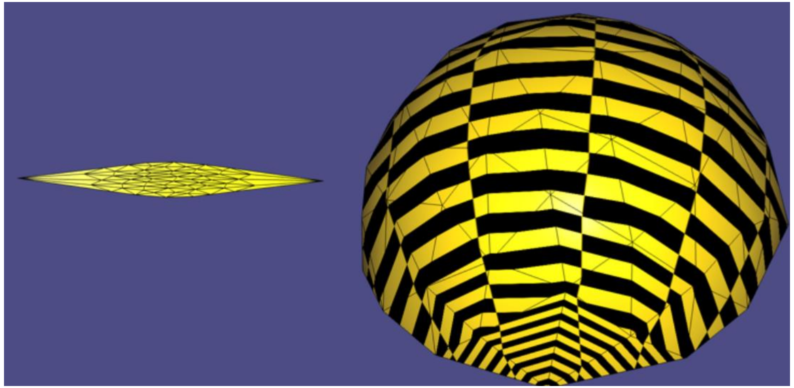
$$\begin{pmatrix} \mathcal{A}^T \mathcal{A} & \mathcal{C}^T \\ \mathcal{C} & 0 \end{pmatrix} \begin{pmatrix} u \\ v \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathcal{A}^T b \\ d \end{pmatrix}$$

Boundary conditions

- Which points to select?
- For LSCM – far away points
  - More uniform distortion distribution
  - Less chance for flips

Boundary conditions

- For ARAP
  - Points for which a reasonable guess can be made
  - Hint: remember that ARAP tries to preserve lengths on the mesh
  - Example for a bad guess:



Boundary conditions

- Does that mean fixing two points might be too strict?
- Task 1.2: you should think about how many DoF you really need to fix for LSCM and ARAP
- It can help to think about when the matrix gets full rank and/or how many DoF you need in each case to fix translation, rotation as well as scale

Distortion Types

- Conformal - angle preserving

$$\mathcal{D}(J) = \|J + J^T - tr(J)I\|_F^2$$

- Isometric - length preserving

$$\mathcal{D}(J) = min_{R \in SO_2} \|J - R\|_F^2$$

- Authalic - area preserving (not in assignment)

$$\mathcal{D}(J) = (\det J\ - 1)^2$$

LSCM Parameterization

- $\mathcal{D}(J) = \|J + J^T - tr(J)I\|_F^2$
- Recall the lecture notes, it is equivalent to $\mathcal{D}(J) = (J_{11} - J_{22})^2 + (J_{12} + J_{21})^2$ where
$$J = \begin{pmatrix} D_x u & D_y u \\ D_x v & D_y v \end{pmatrix}$$
- … now you can write out the least-square system

ARAP Parameterization

- $\mathcal{D}(J) = min_{R \in SO_2} \|J - R\|_F^2$
- R is the closest rotation matrix to J
- Non-linear relationship via SVD:

$$J = U\Sigma V^T \rightarrow R_J = U \begin{pmatrix} 1 & 0 \\ 0 & \det(UV^T) \end{pmatrix} V^T$$

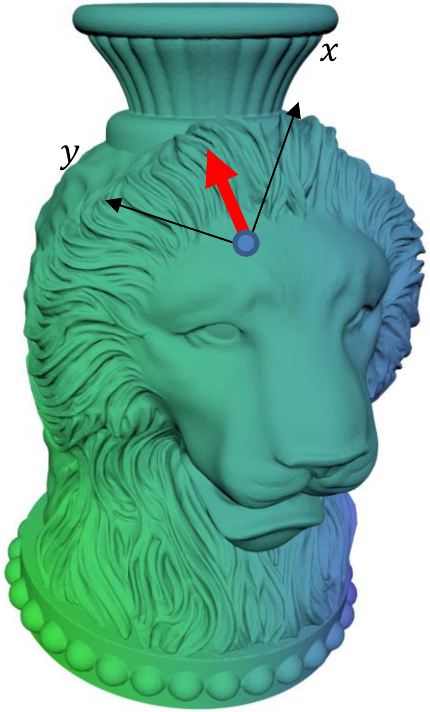
ARAP Parameterization

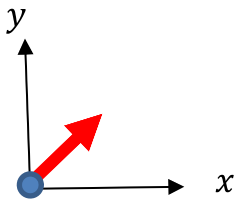
1. Initialize (e.g., using LSCM)
2. Compute Jacobians and closest rotations
3. Minimize $\mathcal{D}(J) = \|J - R\|_F^2$ via linear system with rotations found in (2) fixed
4. If not converged, go to 2

you don’t need to implement this yourselves, we provide the code in the template ☺

Gradient on Mesh

Gradients on surfaces





Like Euclidean gradient **arrow pointing in steepest direction**

Arrows are **tangent** to the surface

described in *local frames*

How to compute gradient on mesh?

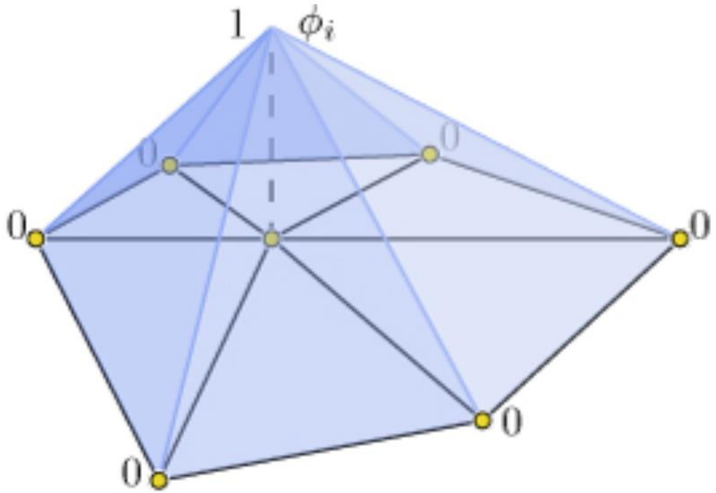
- A piecewise linear function defined on mesh $f\colon \mathcal{S} \rightarrow R$, where $f(v_i) = f_i$
  - Defined on each vertex
  - For a point inside a triangle: interpolation via Barycentric coordinate

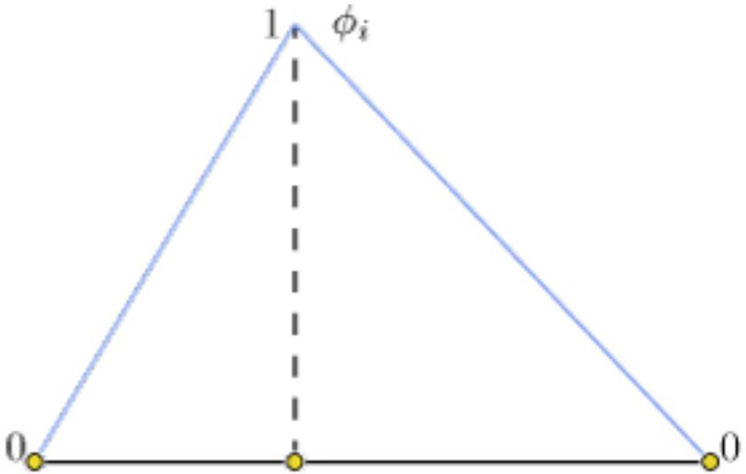
How to compute gradient on mesh?

- Rewrite $f(\boldsymbol{x}) = \sum_{i=1}^{n_v} \Phi_i(\boldsymbol{x}) f_i$
  - $\boldsymbol{x}$ is an arbitrary point on shape $\mathcal{S}$
  - $\Phi_i(\boldsymbol{x})$ is a hat function defined on vertex v_i:

$$\Phi_i(\boldsymbol{x}) = \begin{cases} 1, & \boldsymbol{x} = v_i \\ 0, & \boldsymbol{x} \neq v_i \end{cases}$$

Interpolation hat function





“Libigl Tutorial 204”

How to compute gradient on mesh?

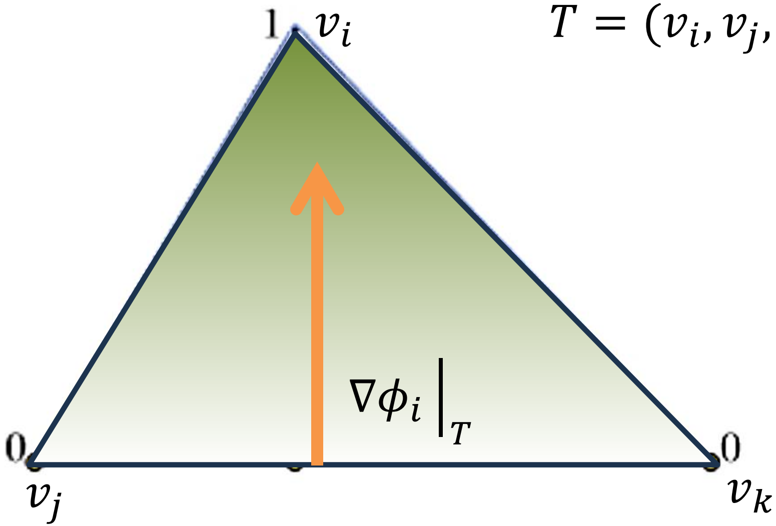
- Rewrite $f(\boldsymbol{x}) = \sum_{i=1}^{n_v} \phi_i(\boldsymbol{x}) f_i$
- Now, we can compute gradient of any f

$$\nabla f(\boldsymbol{x}) = \sum_{i=1}^{n_v} \nabla \phi_i(\boldsymbol{x}) f_i$$

- What is the gradient of a hat function?

Gradient of a hat function

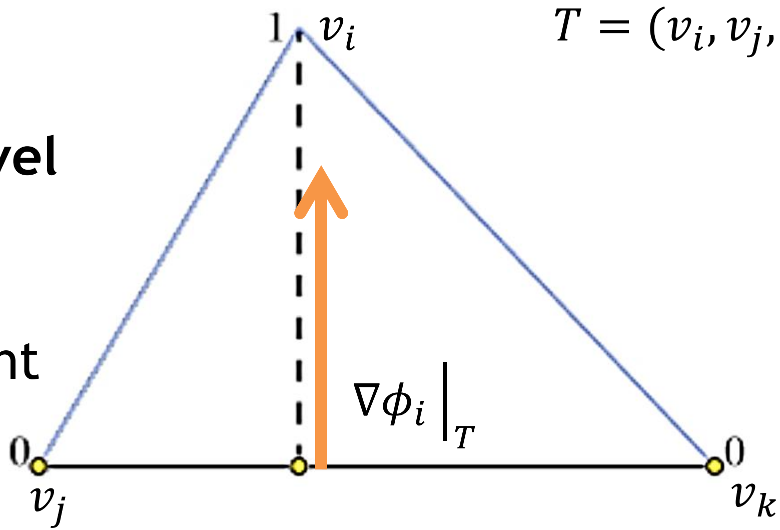
- ϕ_i is piecewise linear
- $\nabla\phi_i$ is constant in each triangle!
- Recall gradient is the steepest ascent direction.
- The steepest ascent direction of a triangle is its height direction



Gradient of a hat function

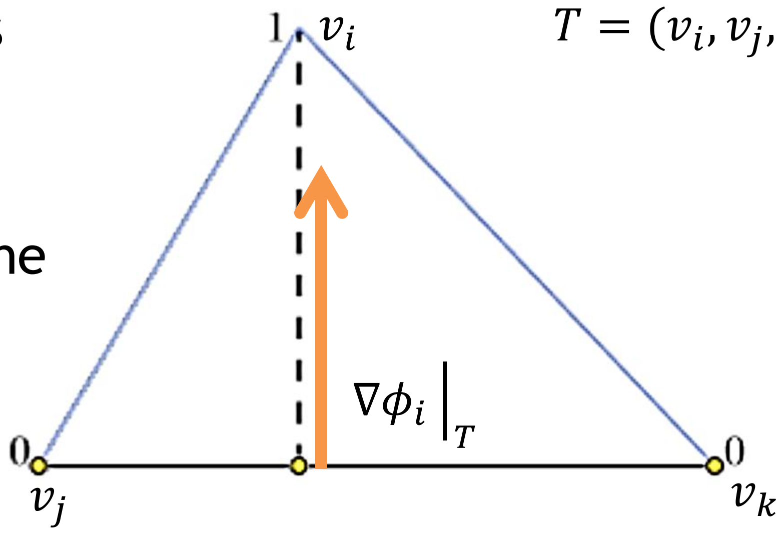
- Also recall in Assignment 2, you have proven that **gradient direction is parallel to the normal of the zero level set!**
- Here the zero level set is the edge (v_j, v_k) → it is in direction of the height as suspected before☺

$T = (v_i, v_j, v_k)$



Gradient of a hat function

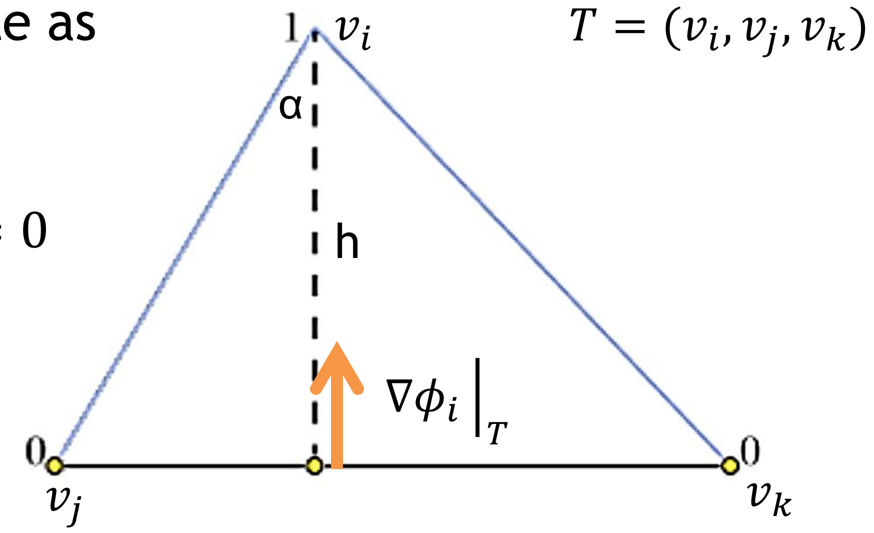
- We now know the gradient direction is orthogonal to the edge jk:
$$\left(v_j - v_k\right)^T \nabla\phi_i = 0$$
- How to compute the scale/length of the gradient?



"Libigl Tutorial 204"

Gradient of a hat function

- ϕ_i is piecewise linear, thus can rewrite as
  $$\phi_i(x) = \phi_i(v_i) + \nabla\phi_i(v_i)^T(x - v_i)$$
- Evaluate it at vertex v_j
  $$\phi_i(v_j) = \boxed{1} + \nabla\phi_i^T(v_j - v_i) = 0$$
- i.e., $\nabla\phi_i^T(v_i - v_j) = 1$
- $|\nabla\phi_i^T||vi - vj|\cos\alpha = 1$
- $|\nabla\phi_i^T||vi - vj|\frac{|h|}{|vi-vj|} = 1$
- $|\nabla\phi_i^T||h| = 1 \rightarrow |\nabla\phi_i^T| = \frac{1}{|h|}$

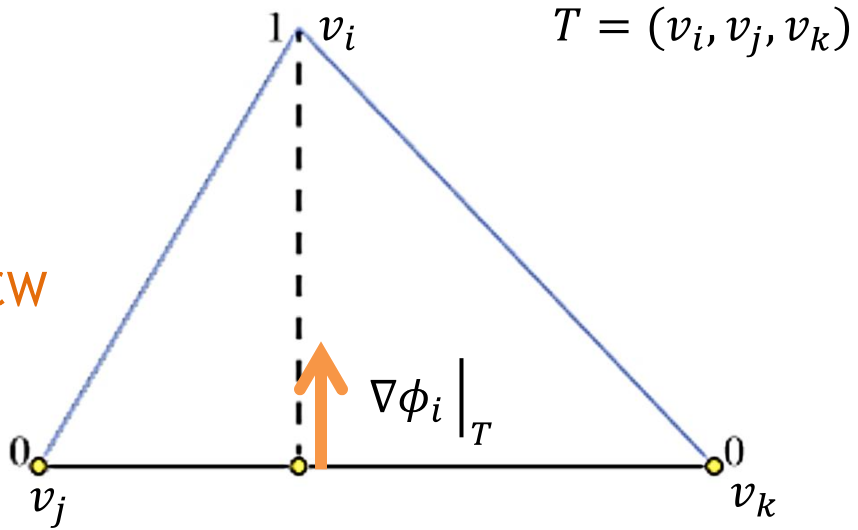


Gradient of a hat function

- Bringing together scale and direction:
- $\nabla \phi_i = \frac{1}{|h|} \frac{e_{jk}^{\perp}}{|e_{jk}^{\perp}|}$

$e_{jk}^{\perp} = n_T \times (v_k - v_j)$, rotate the edge 90° CCW

- $\nabla \phi_i = \frac{e_{jk}^{\perp}}{2A_T}$



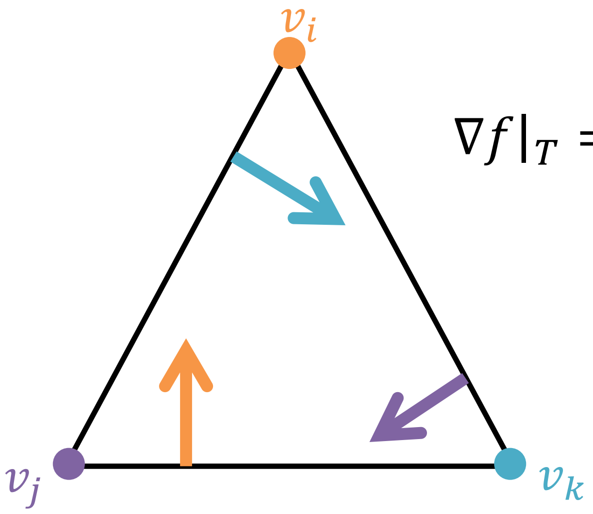
How to compute gradient on mesh?

- Rewrite $f(\boldsymbol{x}) = \sum_{i=1}^{n_v} \phi_i(\boldsymbol{x}) f_i$
- Now, we can compute gradient of any f

$$\nabla f(\boldsymbol{x}) = \sum_{i=1}^{n_v} \nabla \phi_i(\boldsymbol{x}) f_i$$

$$\boxed{\nabla \phi_i \Big|_{T=(v_i, v_j, v_k)} = \frac{e_{jk}^{\perp}}{2A_T}}$$

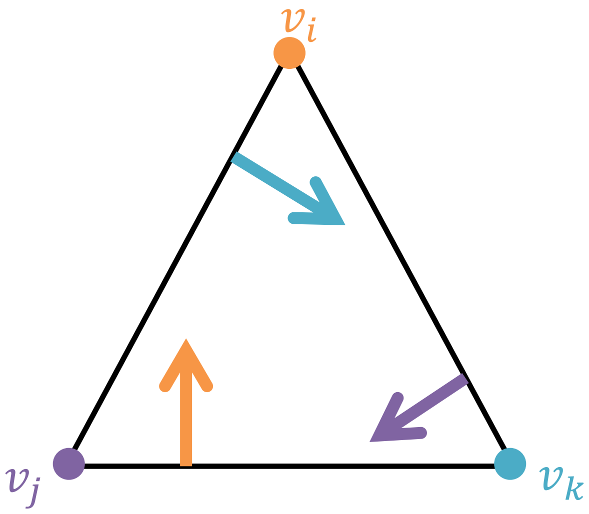
How to compute gradient on mesh?



$$\nabla f|_T = \sum_{l=1}^{n_v} \nabla\phi_l(\boldsymbol{x}) f_l = \nabla\phi_i(\boldsymbol{x}) f_i + \nabla\phi_j(\boldsymbol{x}) f_j + \nabla\phi_k(\boldsymbol{x}) f_k$$

$$= \frac{1}{2A_T}\left(f_i e_{jk}^{\perp} + f_j e_{ki}^{\perp} + f_k e_{ij}^{\perp}\right)$$

How to write gradient in matrix form

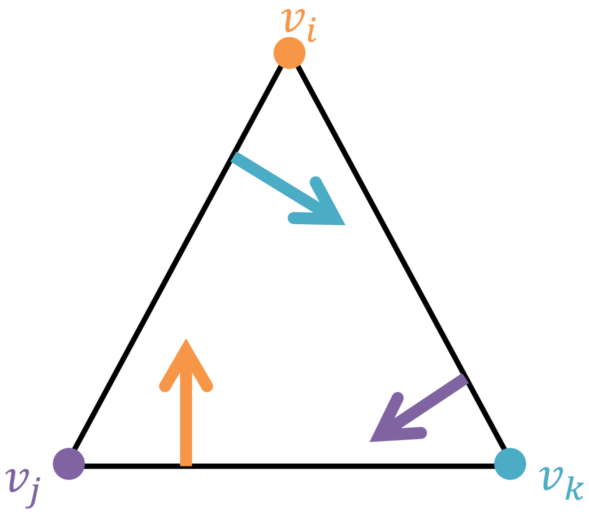


$$\nabla f \Big|_T = \frac{1}{2A_T}\left(f_i e_{jk}^{\perp} + f_j e_{ki}^{\perp} + f_k e_{ij}^{\perp}\right)$$

- f is defined on vertices, i.e., $f \in R^{n_v \times 1}$
- ∇f assigns a constant vector to each face, i.e., $\nabla f \in R^{n_f \times 3}$, i-th row represent the gradient in the i-th face
- We can flatten ∇f to a vector, i.e., $\nabla f \in R^{3n_f \times 1}$
- Gradient operator is a linear transformation G:

$$\nabla f = Gf, \text{ where } G \in R^{3n_f \times n_v}$$

How to write gradient in matrix form



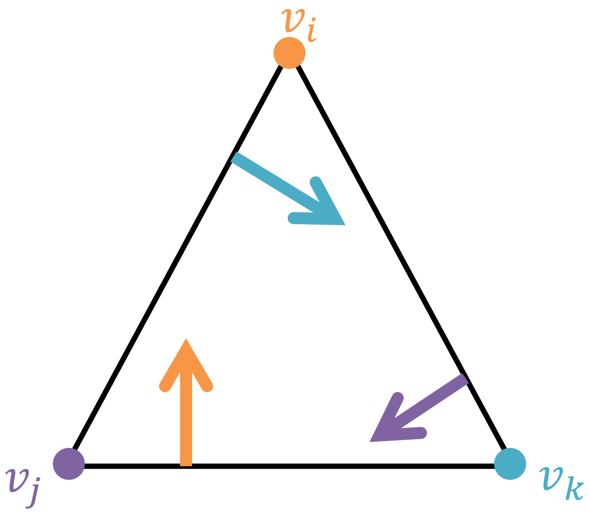
$$\nabla f \Big|_T = \frac{1}{2A_T}\left(f_i e_{jk}^{\perp} + f_j e_{ki}^{\perp} + f_k e_{ij}^{\perp}\right)$$

- $\nabla f = Gf$, where $G \in R^{3n_f \times n_v}$
- $G(t,i) = \frac{1}{2A_t}\left(e_{jk}^{\perp}\right)_1$
- $G(t + n_f, i) = \frac{1}{2A_t}\left(e_{jk}^{\perp}\right)_2$
- $G(t + 2n_f, i) = \frac{1}{2A_t}\left(e_{jk}^{\perp}\right)_3$

Flatten the three entries of the gradient vector

Similar for j, k

How to write gradient in matrix form



“igl::grad”

```
// create sparse gradient operator matrix
G.resize(dims*m,nv);
std::vector<Eigen::Triplet<typename DerivedV::Scalar> > Gijv;
Gijv.reserve(4*dims*m);
for(int f = 0;f<F.rows();f++)
{
  for(int d = 0;d<dims;d++)
  {
    Gijv.emplace_back(f+d*m,F(f,1), eperp13(f,d));
    Gijv.emplace_back(f+d*m,F(f,0),-eperp13(f,d));
    Gijv.emplace_back(f+d*m,F(f,2), eperp21(f,d));
    Gijv.emplace_back(f+d*m,F(f,0),-eperp21(f,d));
  }
}
G.setFromTriplets(Gijv.begin(), Gijv.end());
}
```

Why only used $e_{13}^{\perp}$ and $e_{21}^{\perp}$?

Hint: try to prove by definition $e_{13}^{\perp} + e_{32}^{\perp} + e_{21}^{\perp} = 0$

Gradient in local frame

"src/main.cpp"

```
static void computeSurfaceGradientMatrix(SparseMatrix<double> & D1, SparseMatrix<double> & D2)
{
  MatrixXd F1, F2, F3;
  SparseMatrix<double> DD, Dx, Dy, Dz;

  igl::local_basis(V, F, F1, F2, F3);
  igl::grad(V, F, DD);

  Dx = DD.topLeftCorner(F.rows(), V.rows());
  Dy = DD.block(F.rows(), 0, F.rows(), V.rows());
  Dz = DD.bottomRightCorner(F.rows(), V.rows());

  D1 = F1.col(0).asDiagonal()*Dx + F1.col(1).asDiagonal()*Dy + F1.col(2).asDiagonal()*Dz;
  D2 = F2.col(0).asDiagonal()*Dx + F2.col(1).asDiagonal()*Dy + F2.col(2).asDiagonal()*Dz;
}
```

$$G = DD = \begin{pmatrix} D_x \\ D_y \\ D_z \end{pmatrix} \in R^{3n_f \times n_v}$$

$F_1, F_2 \in R^{n_f \times 3}$ local frames defined per face

Gradient in local frame

"src/main.cpp"

```cpp
static void computeSurfaceGradientMatrix(SparseMatrix<double> & D1, SparseMatrix<double> & D2)
{
  MatrixXd F1, F2, F3;
  SparseMatrix<double> DD, Dx, Dy, Dz;

  igl::local_basis(V, F, F1, F2, F3);
  igl::grad(V, F, DD);

  Dx = DD.topLeftCorner(F.rows(), V.rows());
  Dy = DD.block(F.rows(), 0, F.rows(), V.rows());
  Dz = DD.bottomRightCorner(F.rows(), V.rows());

  D1 = F1.col(0).asDiagonal()*Dx + F1.col(1).asDiagonal()*Dy + F1.col(2).asDiagonal()*Dz;
  D2 = F2.col(0).asDiagonal()*Dx + F2.col(1).asDiagonal()*Dy + F2.col(2).asDiagonal()*Dz;
}
```

D_x, D_y, D_z gradient operator in R^3

$D_1, D_2 \in R^{n_f \times n_v}$ gradient operator along the two local axis

Thank You

INTERACTIVE GEOMETRY LAB

11. April 2025

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich