Digital Geometry
- Discrete Differential Geometry

Junjie Cao @ DLUT

Spring 2019

http://jjcao.github.io/DigitalGeometry/

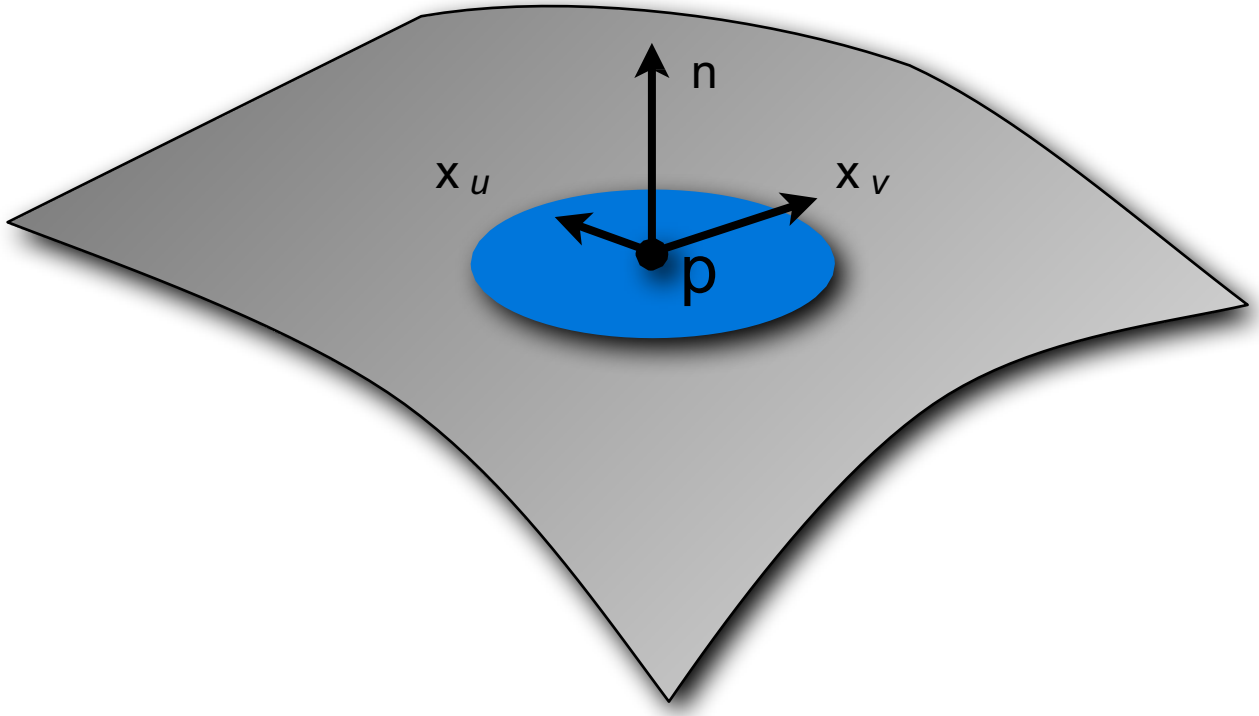
The purpose of computing is insight, not numbers

Normal

Normal Vectors

- Continuous surface

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



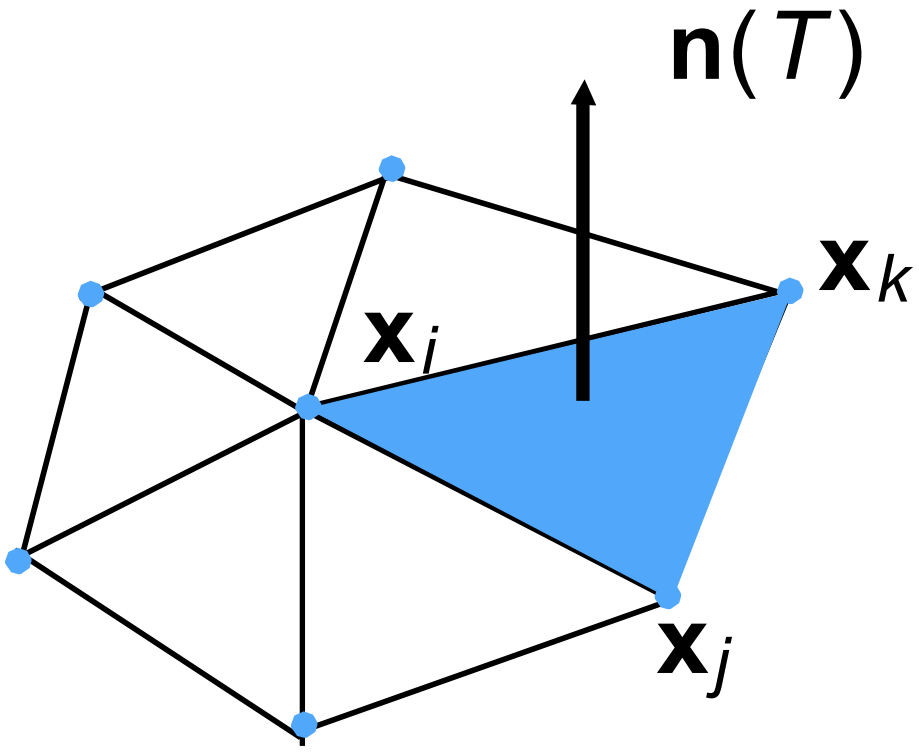
Normal vector

$$\mathbf{n} = \frac{\mathbf{x}_u \times \mathbf{x}_v}{\|\mathbf{x}_u \times \mathbf{x}_v\|}$$

Assume *regular* parameterization

$$\mathbf{x}_u \times \mathbf{x}_v \neq \mathbf{0}$$ normal exists

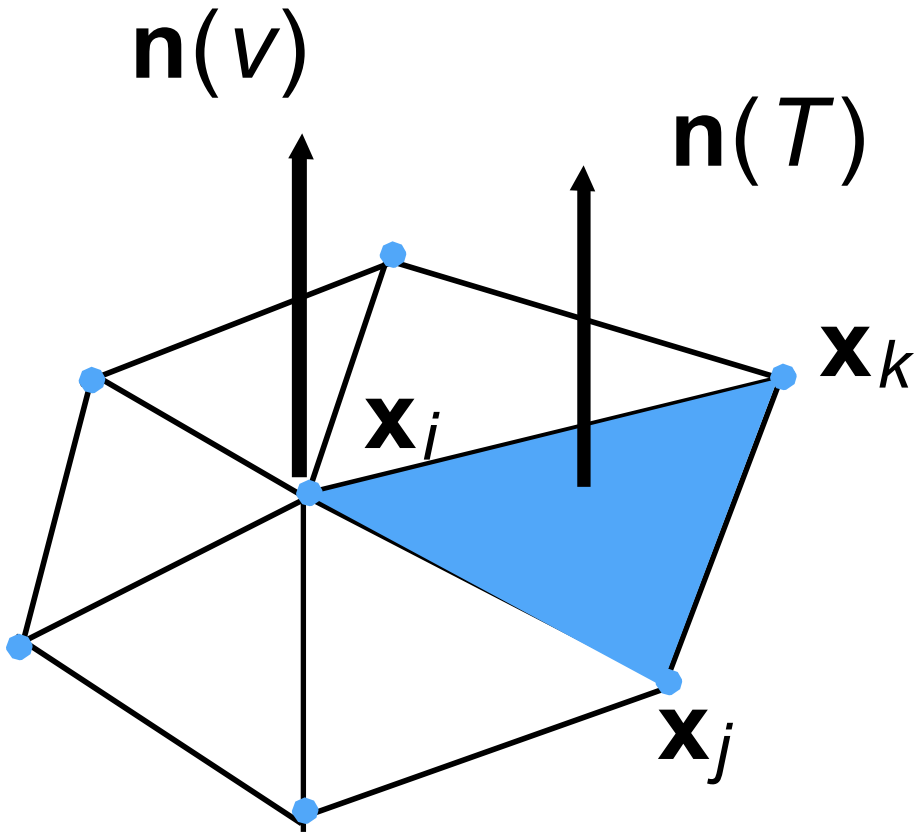
Discrete **Face Normal Vectors**



$$\mathbf{n}(T) = \frac{(\mathbf{x}_j - \mathbf{x}_i) \times (\mathbf{x}_k - \mathbf{x}_i)}{\|(\mathbf{x}_j - \mathbf{x}_i) \times (\mathbf{x}_k - \mathbf{x}_i)\|}$$

$$T = (\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_j)$$

Discrete **Vertex Normal Vectors**

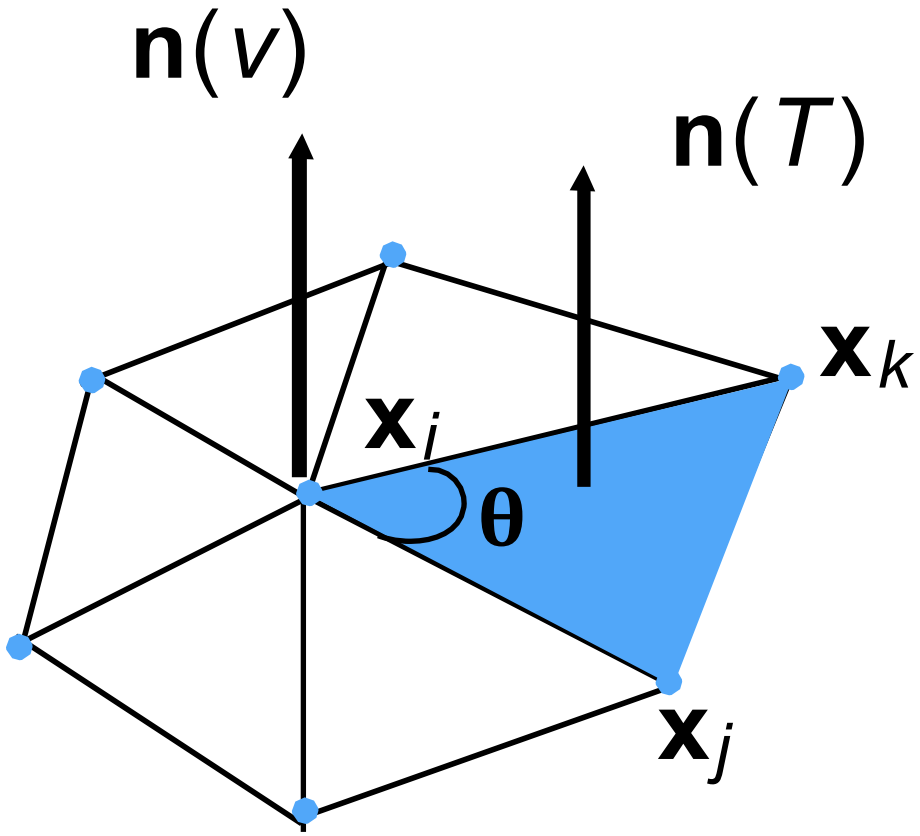


$$\mathbf{n}(T) = \frac{(\mathbf{x}_j - \mathbf{x}_i) \times (\mathbf{x}_k - \mathbf{x}_i)}{\|(\mathbf{x}_j - \mathbf{x}_i) \times (\mathbf{x}_k - \mathbf{x}_i)\|}$$

$$T = (\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_j)$$

$$\mathbf{n}(v) = \frac{\sum_{T \in \mathcal{N}_1(v)} \alpha_T \, \mathbf{n}(T)}{\left\| \sum_{T \in \mathcal{N}_1(v)} \alpha_T \, \mathbf{n}(T) \right\|}$$

Discrete **Vertex Normal Vectors**



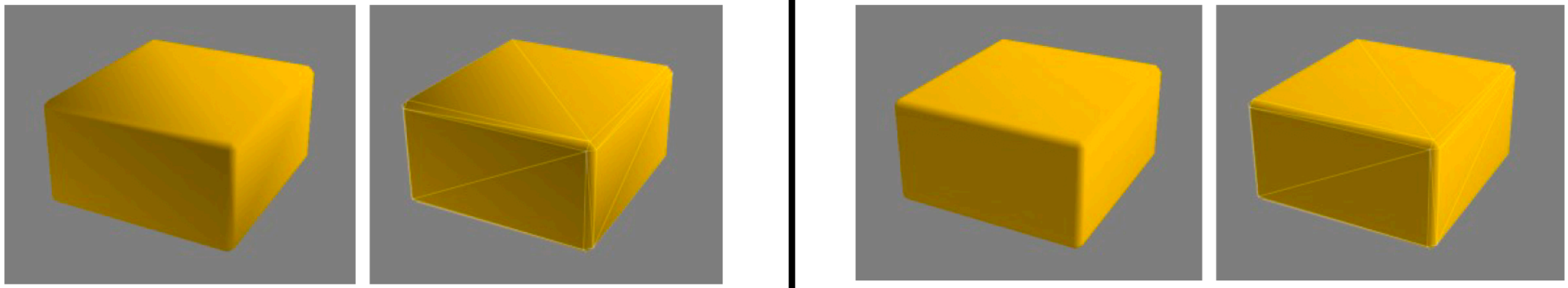
$$\mathbf{n}(T) = \frac{(\mathbf{x}_j - \mathbf{x}_i) \times (\mathbf{x}_k - \mathbf{x}_i)}{\|(\mathbf{x}_j - \mathbf{x}_i) \times (\mathbf{x}_k - \mathbf{x}_i)\|}$$

$$T = (\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_j)$$

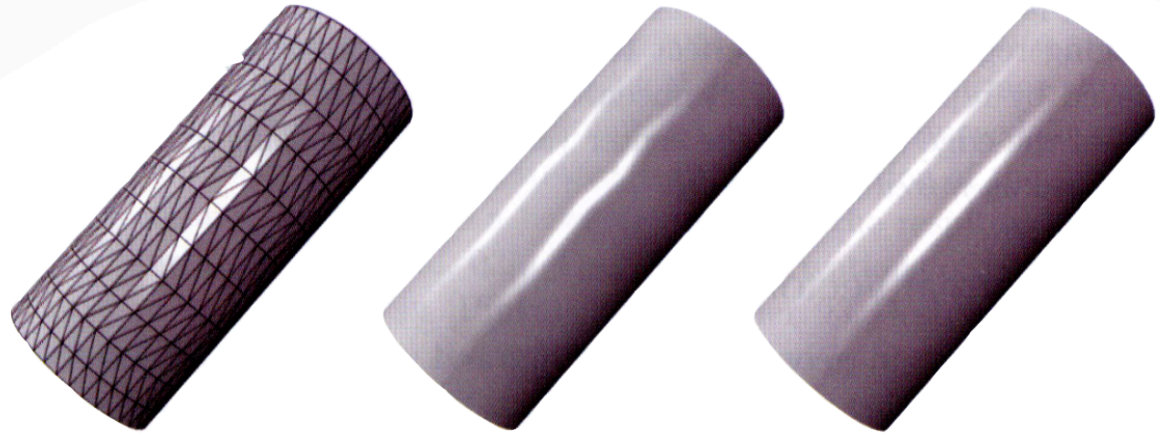
$$\mathbf{n}(v) = \frac{\sum_{T \in \mathcal{N}_1(v)} \alpha_T \, \mathbf{n}(T)}{\left\| \sum_{T \in \mathcal{N}_1(v)} \alpha_T \, \mathbf{n}(T) \right\|}$$

$$\alpha_T = 1 \qquad \alpha_T = |T| \qquad \alpha_T = \theta_T$$

Shading via various **Vertex Normal Vectors**



Without area-weighting



tessellated cylinder

$\alpha_T = 1$

$\alpha_T = |T|$

$\alpha_T = \theta_T$

A Comparison of Algorithms for Vertex-Normal Computation.“ The Visual Computer, 2005

Local Averaging

Local Averaging

- **Local Neighborhood $N(\mathbf{x})$ of a point**
  - often coincides with mesh vertex v_i
  - n-ring neighborhood $N_n(v_i)$ or local geodesic ball
- **Neighborhood size**
  - Large: smoothing is introduced, stable to noise
  - Small: fine scale variation, sensitive to noise

Local average operator

$$(\mathbf{W}f)_i = \sum_{(i,j)\in E} w_{ij} f_j$$

A particularly important class of local operators are local smoothings (also called filterings) that perform a local weighted sum around each vertex of the mesh. For this averaging to be consistent, we define a normalized operator W˜ whose set of weights sum to one

Definition 9 (Local averaging operator). *A local normalized averaging is* $\tilde{W} = (\tilde{w}_{ij})_{i,j\in V} \geqslant 0$ *where*

$$\forall (i,j) \in E, \quad \tilde{w}_{ij} = \frac{w_{ij}}{\sum_{(i,j)\in E} w_{ij}}.$$

It can be equivalently expressed in matrix form as

$$\tilde{W} = D^{-1}W \quad \textit{with} \quad D = \mathrm{diag}_i(d_i) \quad \textit{where} \quad d_i = \sum_{(i,j)\in E} w_{ij}.$$

- Combinatorial/uniform weights

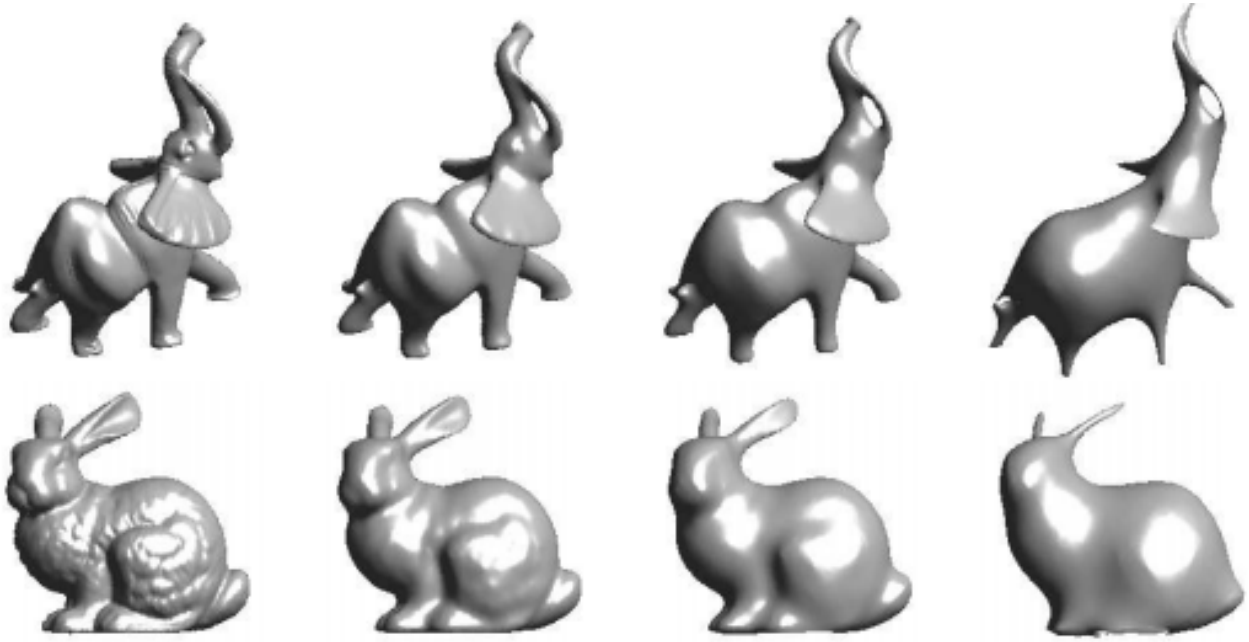
$$\forall (i,j) \in E, \quad w_{ij} = 1.$$

- Distance weights

$$\forall (i,j) \in E, \quad w_{ij} = \frac{1}{\|x_j - x_i\|^2}.$$

An **example** of such iterations applied to the three coordinates of mesh.

- One can use iteratively a smoothing in order to further filter a function on a mesh. The resulting vectors Wf, W^2f,. .. , W^kf are increasingly smoothed version of f.
- The sharp features of the mesh tend to disappear during iterations.

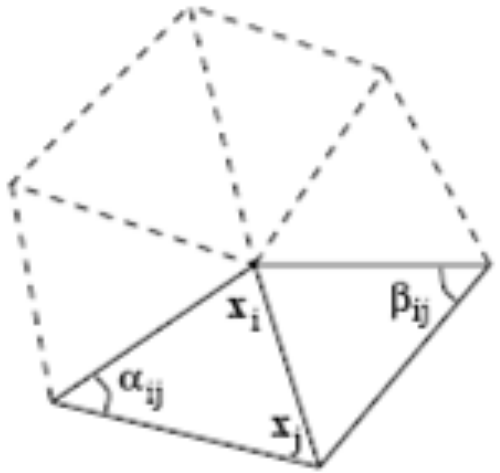


Normalization and some famous weights

$$(\mathbf{W}f)_i = \sum_{(i,j)\in E} w_{ij} f_j$$

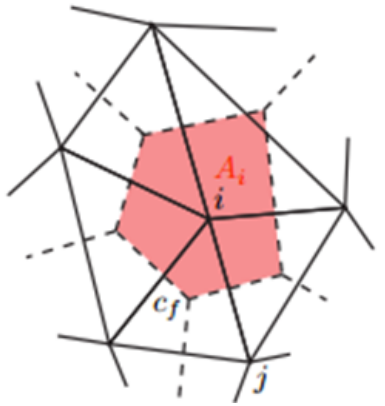
Cotangent weights

$$w_{ij} = \cot \alpha_{ij} + \cot \beta_{ij} .$$



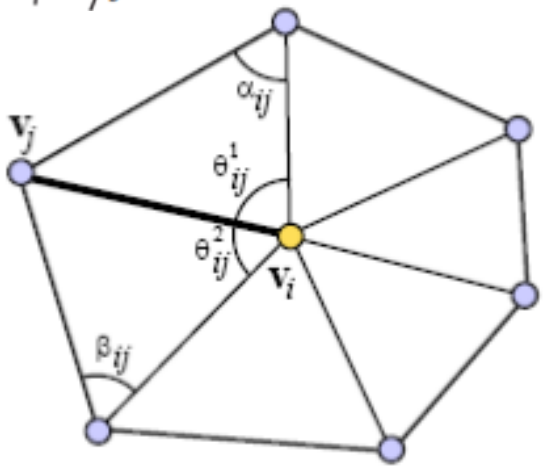
Mean curvature weights

$$w_{ij} = \frac{1}{4A(v_i)} (\cot \alpha_{ij} + \cot \beta_{ij}) .$$



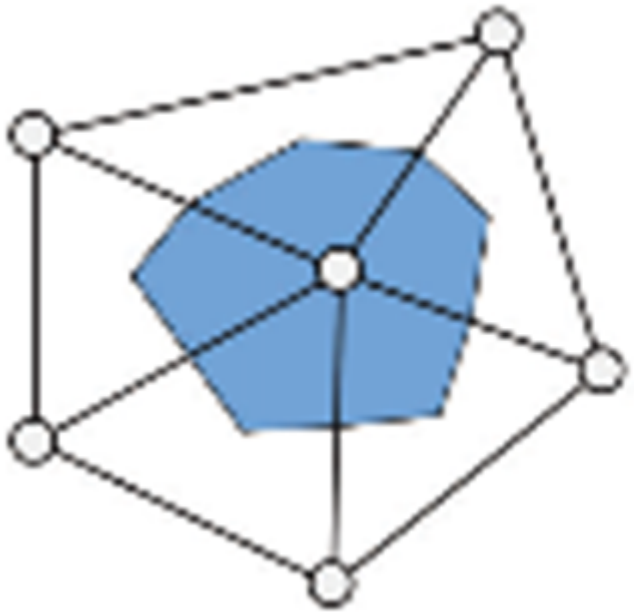
Mean value weights

$$w_{ij} = \frac{\tan(\theta_{ij}^1 / 2) + \tan(\theta_{ij}^2 / 2)}{\|v_i - v_j\|} .$$



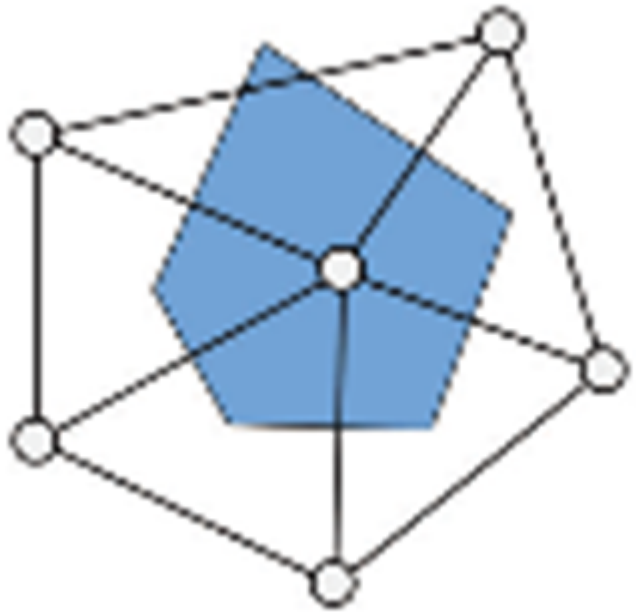
What is the area of the Voronoi region (in red) of the vertex?

Local Averaging: 1-Ring



Barycentric cell

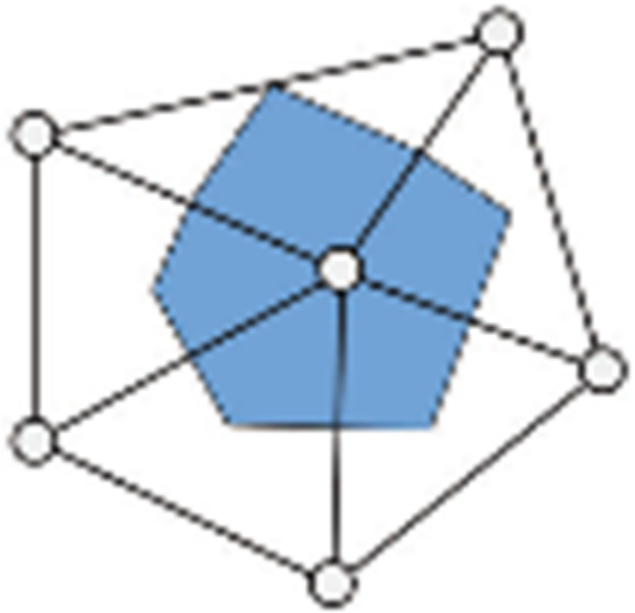
(barycenters/edgemidpoints)



Voronoi cell

(circumcenters)

tight error bound



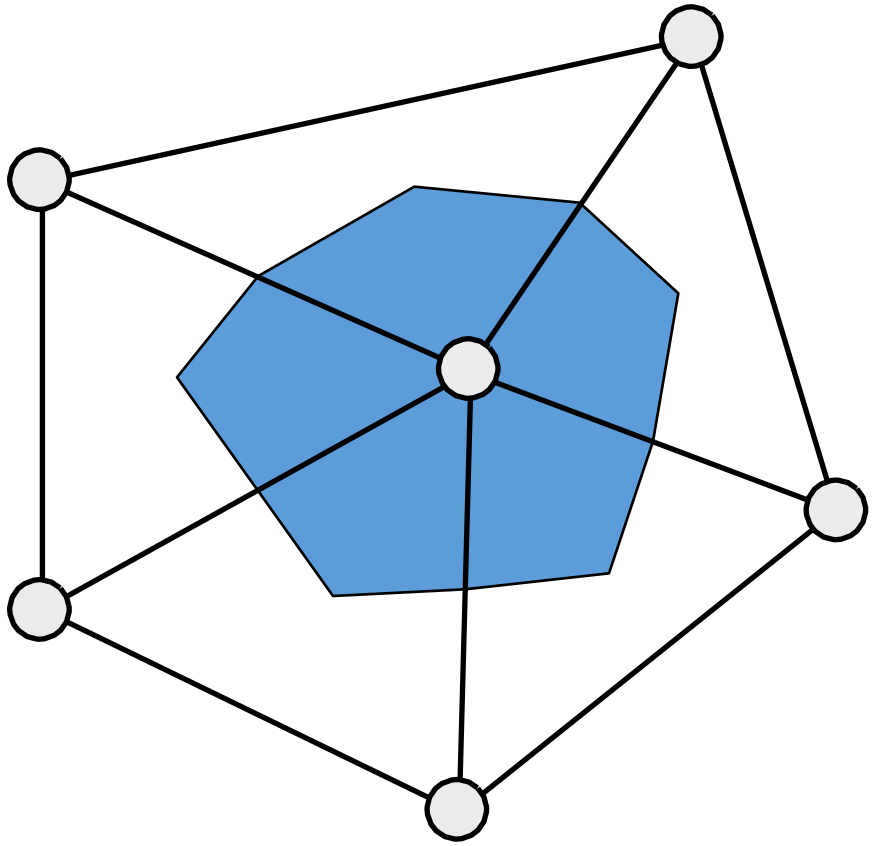
Mixed Voronoi cell

(circumcenters/midpoint)

better approximation

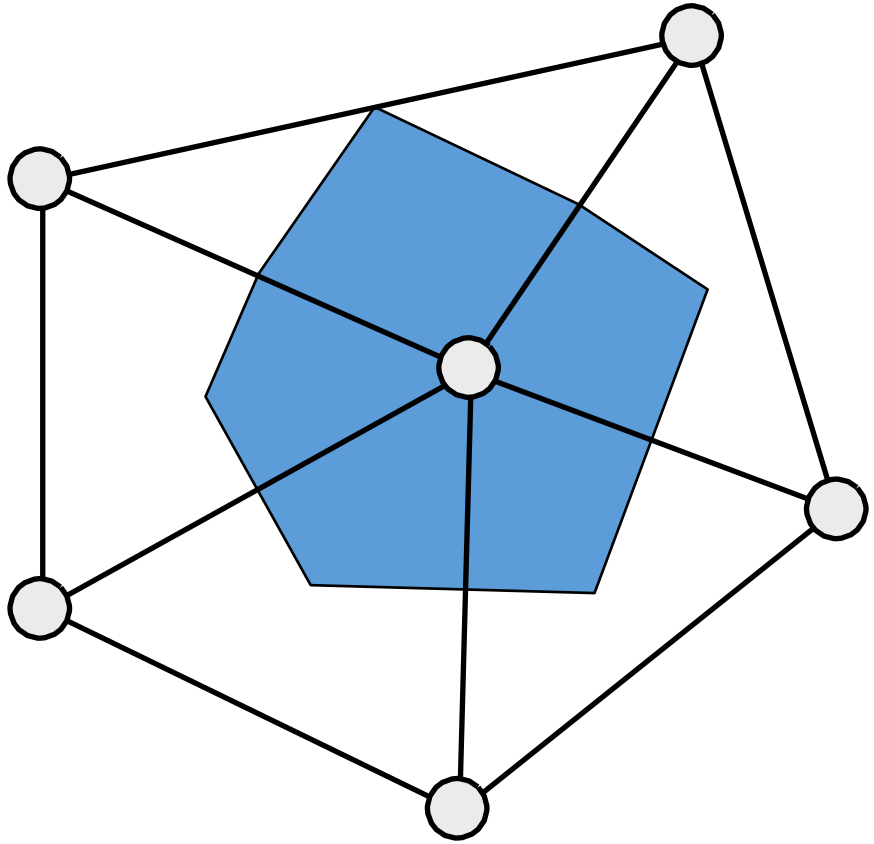
Barycentric Cells

- **Connect edge midpoints and triangle barycenters**
  - Simple to compute
  - Area is 1/3 o triangle areas



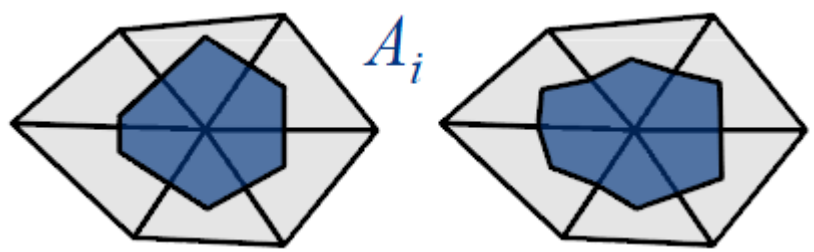
Mixed Cells

- **Connect edge midpoints and**
  - Circumcenters for non-obtuse triangles
  - Midpoint of opposite edge for obtuse triangles
  - Better approximation, more complex to compute…



Mixed Regions

$\mathcal{A}_{\text{Mixed}} = 0$
For each triangle T from the 1-ring neighborhood of $\mathbf{x}$
    If T is non-obtuse, `// Voronoi safe`
        `// Add Voronoi formula (see Section 3.3)`
        $\mathcal{A}_{\text{Mixed}} +=$ Voronoi region of $\mathbf{x}$ in T
    Else `// Voronoi inappropriate`
        `// Add either area(`T`)/4 or area(`T`)/2`
        If the angle of T at $\mathbf{x}$ is obtuse
            $\mathcal{A}_{\text{Mixed}} += \text{area}(T)/2$
        Else
            $\mathcal{A}_{\text{Mixed}} += \text{area}(T)/4$

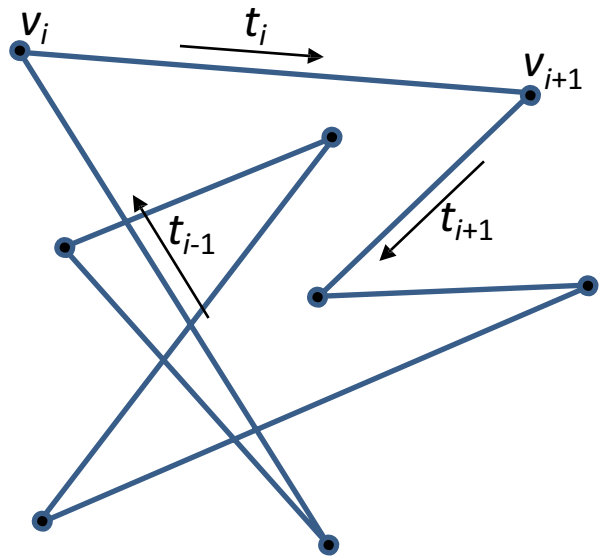


Discrete Curves

Q: Where and how should we define curvature?

A: Since the tangent only changes at vertices we should define the curvature as a vertex value.

We should define the value of the curvature as
the change in the tangent as
we move through the vertex.

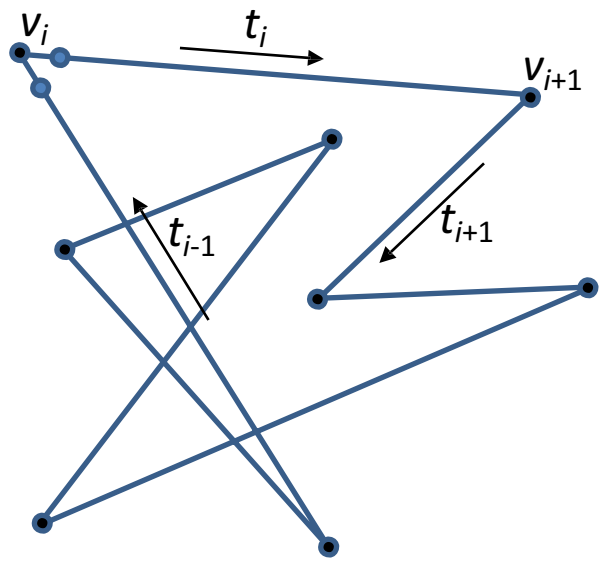


Discrete Curves

Note that we cannot define the curvature as the result of a limiting process:

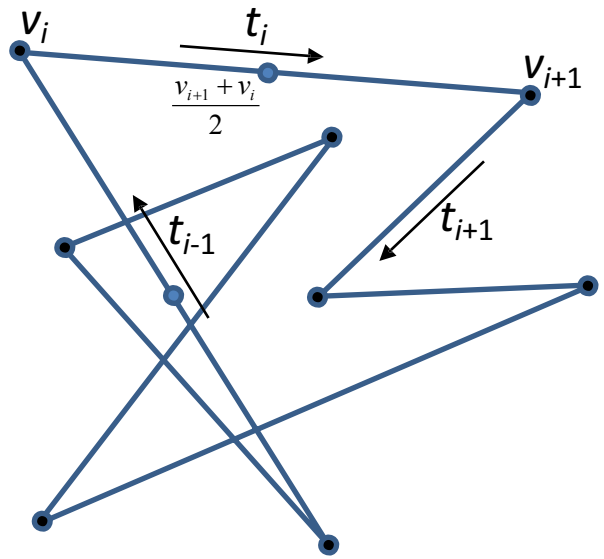
$$\kappa_i = \lim_{\Delta t \to 0} \frac{\left| t_i - t_{i-1} \right|}{\left\| \left[(1-\Delta t) v_i + \Delta t v_{i+1} \right] - \left[(1-\Delta t) v_i + \Delta t v_{i-1} \right] \right\|}$$

$$= \lim_{\Delta t \to 0} \frac{\left| t_i - t_{i-1} \right|}{\Delta t \left| v_{i+1} - v_{i-1} \right|}$$



Discrete Curves

However, we can estimate it using the finite- differences using the edge centers $(v_i+v_{i+1})/2$.

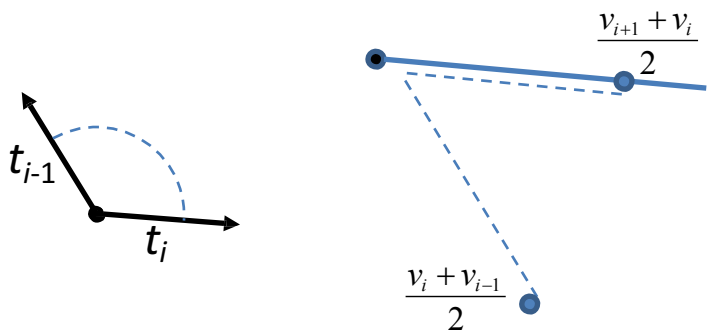


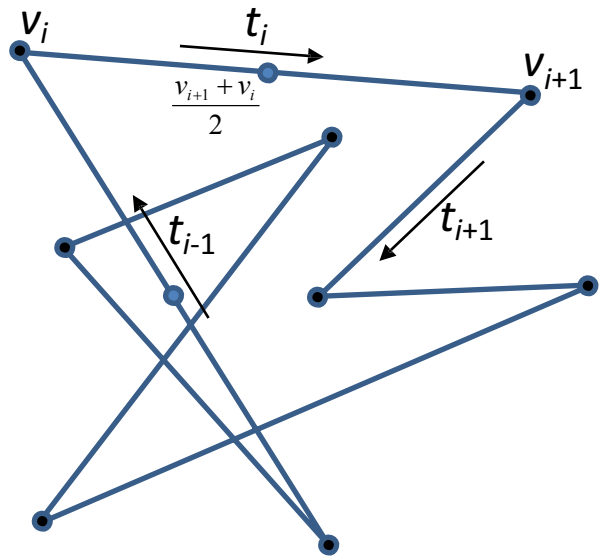
Discrete Curves

However, we can estimate it using the finite- differences using the edge centers $(v_i+v_{i+1})/2$.

Specifically, we define the **curvature** in terms of change in tangents angle divided by the arc-length between **edge centers**:

$$\kappa_i = \frac{\angle t_i t_{i+1}}{\left|(v_{i+1}-v_i)/2\right| + \left|(v_i - v_{i-1})/2\right|}$$

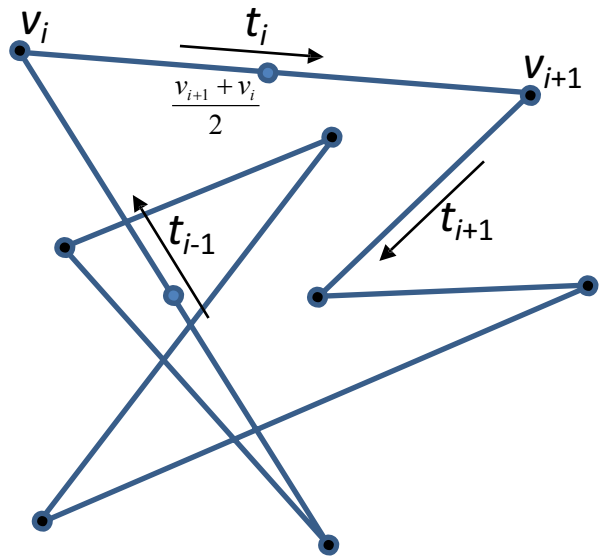




Discrete Curves

- Since we are only storing the curvature at the vertices, we want the value to correspond to the “total curvature associated to the vertex”:

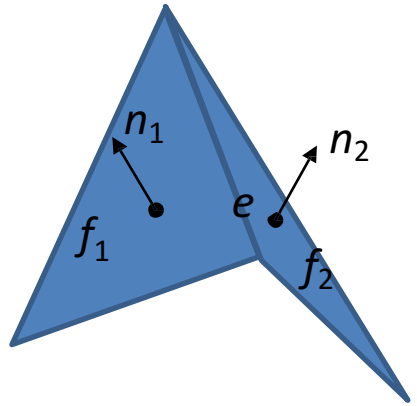
$$\kappa(v_i) = \frac{|v_{i+1} - v_i| + |v_i - v_{i-1}|}{2} \qquad \kappa_t = \angle\, t_i t_{i+1}$$



Discrete Surfaces - normal curvatures

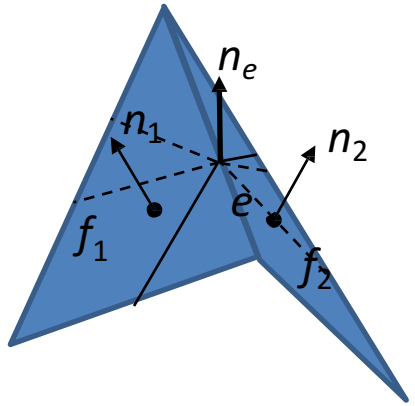
Q: Where and how should we compute the normal curvatures?

A: Since the normals only changes at edges we should compute the normal curvature at the edges.



Discrete Surfaces - normal curvatures

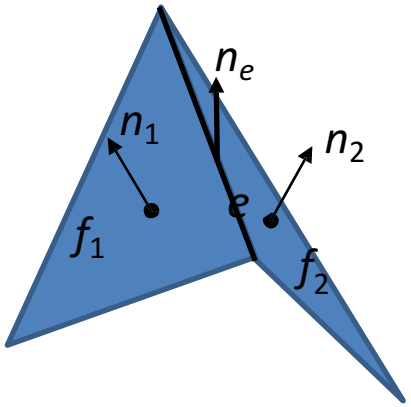
Picking a (good) normal at the edge and looking at the set of planes passing through the normal, we get a family of curves.



Discrete Surfaces - principal curvature

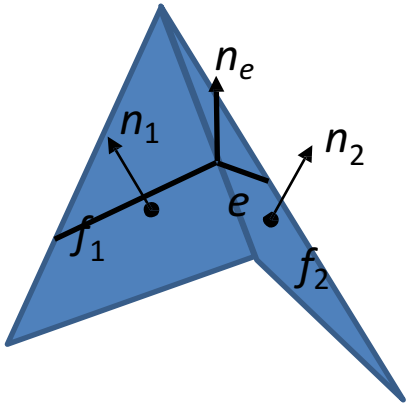
Picking a (good) normal at the edge and looking at the set of planes passing through the normal, we get a family of curves.

Defining the curvature of the curve in terms of the angle between curve segments:

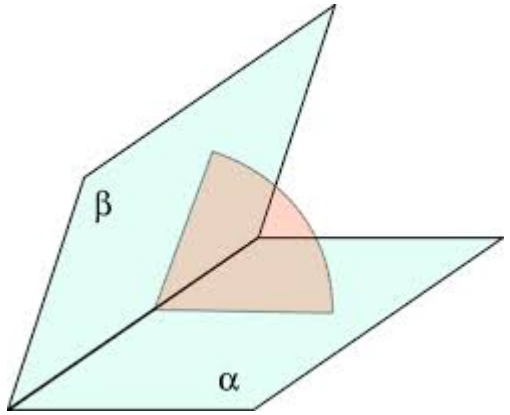


– The min curvature is 0,
with principal curvature direction along ***e***.

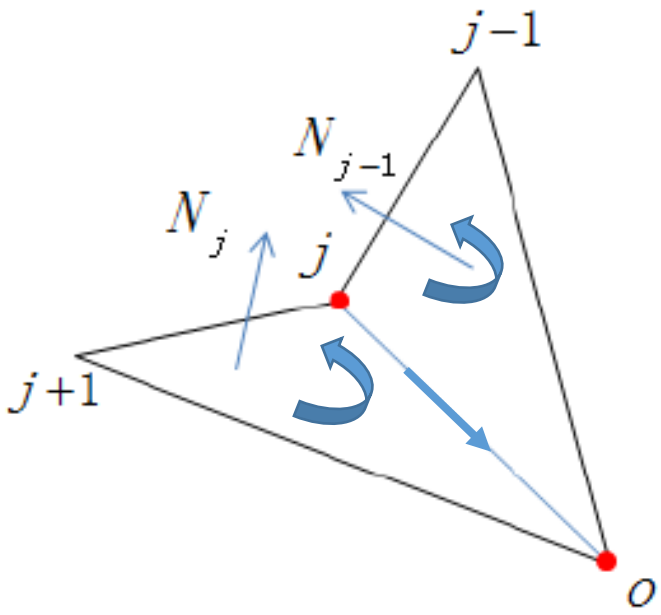
– The max curvature is equal to the **dihedral angle** ($\beta(e)=\angle n_1 n_2$), with principal curvature direction along $\boldsymbol{n_e x e}$, so **orthogonal** to e.

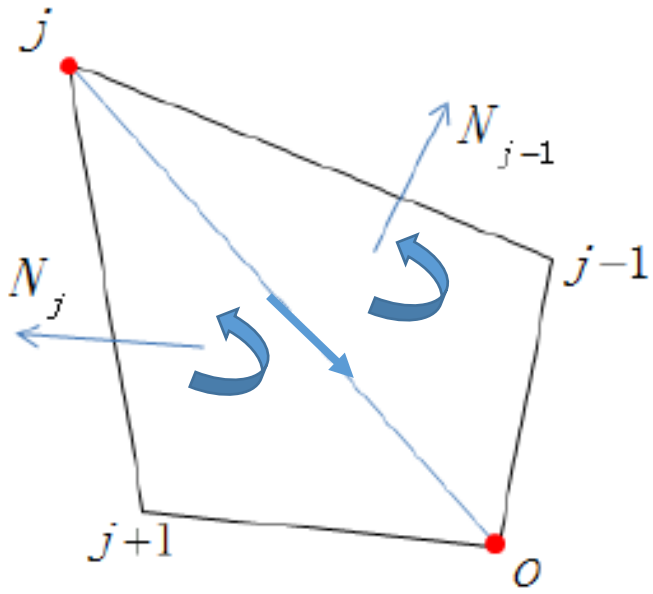


Dihedral-angle



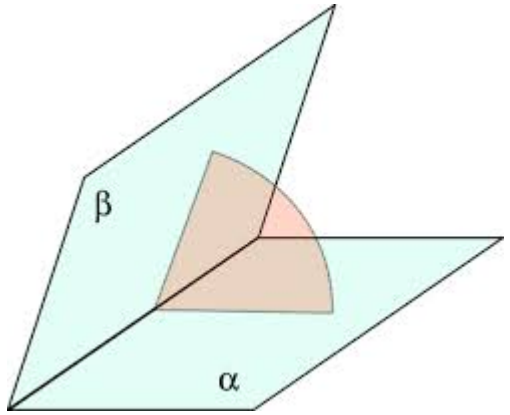
- $\varphi_{AB} = \text{acos}(n_A \cdot n_B) \in$ [0, PI] (see geom3d)
  - sharp features (e.g. dihedral angles <90 degrees)
- Signed dihedral-angle: $\text{dir} = (N_{j-1} \times N_j) \cdot e_{jo}$
  - dir>0, the edge is ridge;
  - dir<0, the edge is ravine





dihedral-angle

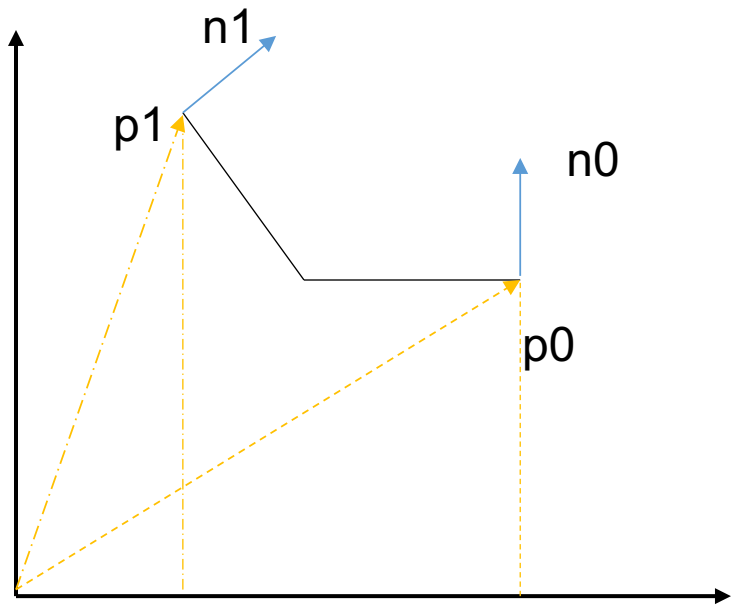
- $\varphi_{AB} = \text{acos}(n_A \cdot n_B) \in$ [0, PI] (see geom3d)
- sharp features (e.g. dihedral angles <90 degrees)



- Signed dihedral-angle
  - A surface may depart from planarity by a positive or a negative dihedral angle (convex or concave).
  - vcg::face::DihedralAngleRad (FaceType &f, const int i)
  - Compute the signed dihedral angle between the normals of two adjacent faces.
  - It simply use the projection of the opposite vertex onto the plane of the other one.

Signed dihedral-angle v.s. Convex & Concave

- vcg::face::DihedralAngleRad (FaceType &f, const int i)
- It simply use the projection of the opposite vertex onto the plane of the other one.
  - dist01 = n0*p0- n0*p1;
  - dist10 = n1*p1- n1*p0
  - // just to be sure use the sign of the largest in absolute value;
  - if(fabs(dist01) > fabs(dist10)) sign = dist01;
  - else sign=dist10;
- Positive for convex & negative for concave

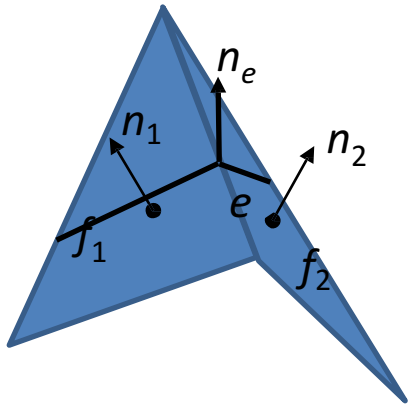


Curvature tensor @ edge

- Thus, the mean curvature at a point on the edge is equal to the dihedral angle between the faces: $H(p \in e) = 0.5 * \beta(e)$

- Moreover, since the principal curvatures at $p \in e$ are 0 and $\beta(e)$, with directions $e/|e|$ and $n_e x e/|n_e x e|$, we can use this to define a curvature tensor at $p \in e$:

$$C(p) = \beta(e) \frac{1}{\|n_e \times e\|^2} (n_e \times e)(n_e \times e)^t$$



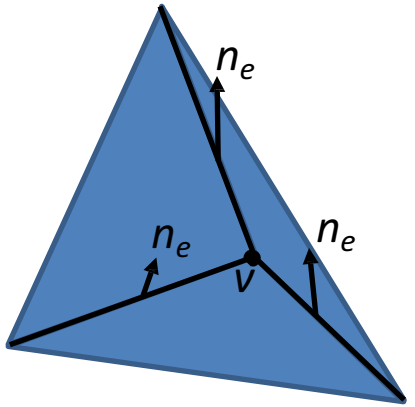
Recall that $C(x(p)) = \kappa_1 Jw_1 Jw_1^t + \kappa_2 Jw_2 Jw_2^t$

Curvature tensor @ vertex

$$C(p) = \beta(e) \quad \frac{1}{\|n_e \times e\|^2} (n_e \times e)(n_e \times e)^t$$

- Averaging the curvature tensor over the edges coming out of a vertex *v*, we get a curvature tensor associated with vertices:

$$C(v) = \sum_{v \in e} \frac{|e|}{2} \beta(e) \quad \frac{1}{\|n_e \times e\|^2} (n_e \times e)(n_e \times e)^t$$

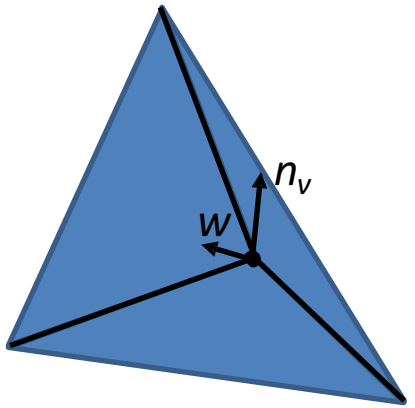


Discrete Surfaces – normal curvature

$$\mathbf{C}(v) = \sum_{v \in e} \frac{|e|}{2} \beta(e) \frac{1}{\|n_e \times e\|^2} (n_e \times e)(n_e \times e)^t$$

For any "tangent direction" *w* at *v*, we can now compute the normal curvature along *w* as:

$$\kappa_v(w) = \frac{w^t \mathbf{C}(v) w}{w^t w}$$



allowing us to compute principal curvatures values and directions at a vertex, then mean & Gaussian curvature

curvature of discrete curve=>normal curvature @ edge => principle curvature @ edge => curvature tensor @ edge => curvature tensor @ vertex => normal curvature @ vertex => principle curvature @ vertex => mean and Gaussian curvature @ vertex

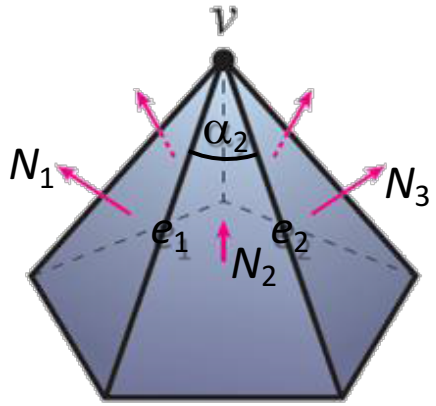
Gaussian curvature via angle deficits

- Note:

- This discretization of curvature information and does not have to conform with others.

- For example, the computed Gaussian curvature is not the same as the one defined using angle deficits.

$$\kappa(v) = 2\pi - \sum_{i=1}^{k} \alpha_i$$



Gauss-Bonnet Theorem

$$\int_M K\,dA + \int_{\partial M} k_g\,ds = 2\pi\chi(M)$$

Gaussian curvature

Geodesic curvature

2-2g

- The theorem states, somewhat surprisingly, if one deforms the surface *M*, its total Gaussian curvature will not change, while the curvatures at some points will.

Gauss-Bonnet Theorem

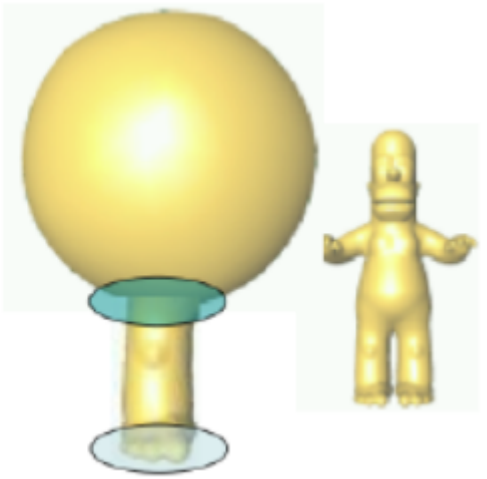
- For a closed surface M
  - Total Gauss Curvature: $\int_M KdA = 2\pi x(M)$, where V-E+F=2-2g=χ

- Sphere

$$\kappa_1 = \kappa_2 = 1/r$$

$$K = \kappa_1 \kappa_2 = 1/r^2$$

$$\int K = 4\pi r^2 \cdot \frac{1}{r^2} = 4\pi$$



when sphere is deformed, new **positive and negative curvature cancel out**

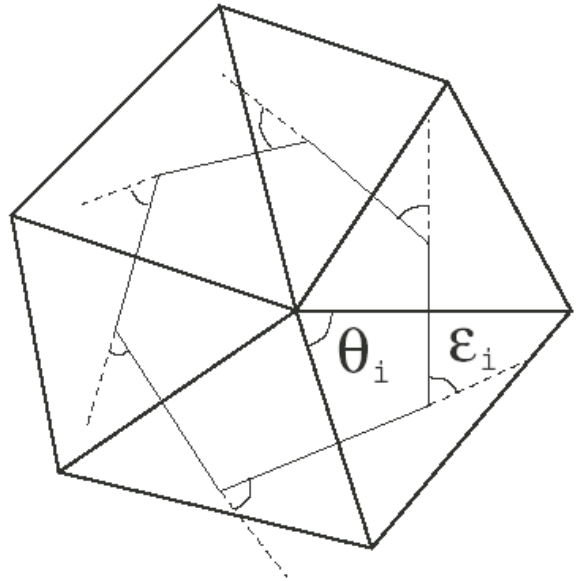
Gauss-Bonnet Theorem

- For a closed surface M
  - Total Gauss Curvature:

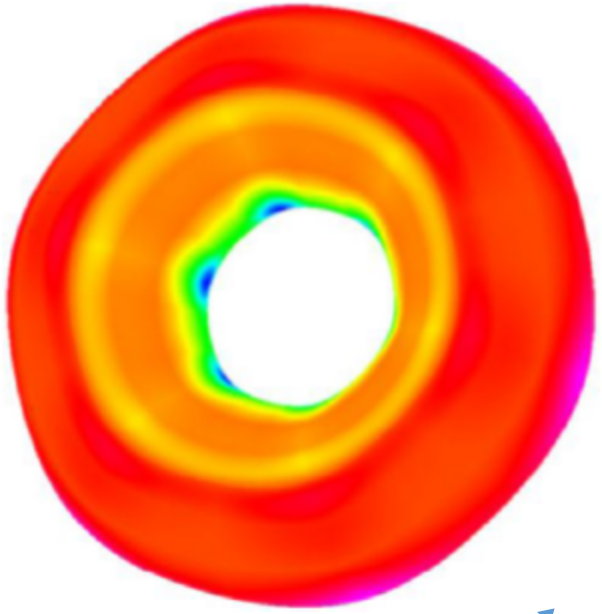
$$\int_M KdA = 2\pi x(M)$$, where V-E+F=2-2g=χ

- For a surface patch
  - For voronoi region: $\iint_{A_M} \kappa_G dA + \int_{\partial A_M} \kappa_g dl = 2\pi$

$$\iint_{A_M} \kappa_G dA = 2\pi - \sum_{j=1}^{\#f} \theta_j$$

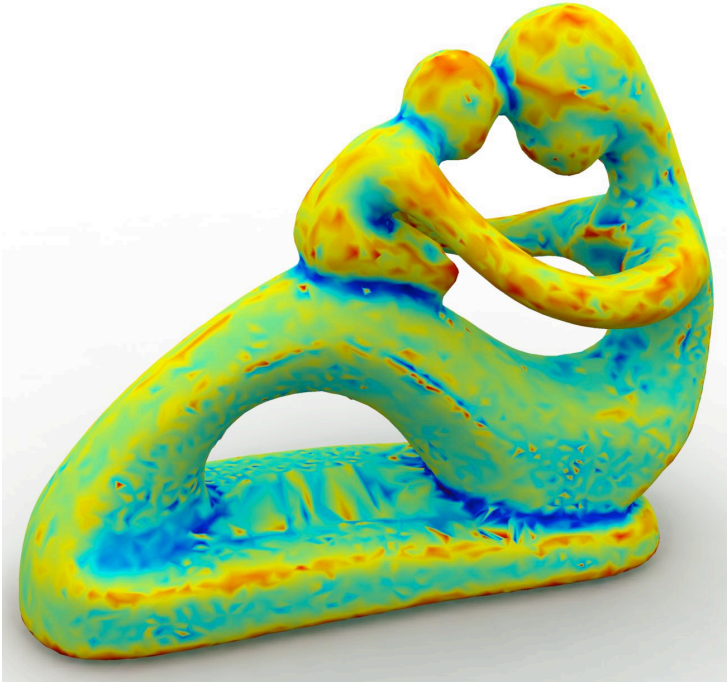


Curvature



K=k1k2

noisy



Discrete Curvature

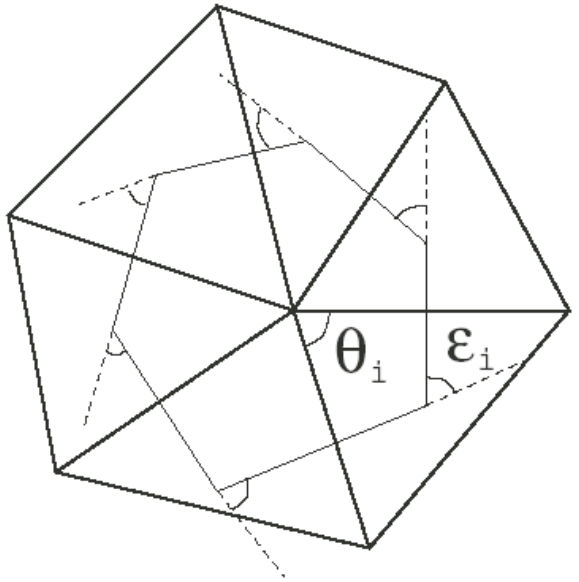
$$\Delta_S \mathbf{x} = -2H\mathbf{n}.$$

$$H(v_i) = \frac{1}{2}\|\Delta \mathbf{x}_i\|$$

Gauss-Bonnet theorem

$$\int_M K\, dA + \int_{\partial M} k_g\, ds = 2\pi\chi(M)$$

$$K(v_i) = \frac{1}{A_i}\left(2\pi - \sum_{v_j \in \mathcal{N}_1(v_i)} \theta_j\right)$$



$$k_H = \frac{k_1+k_2}{2},\ k_G = k_1 \cdot k_2$$

$$\kappa_{1,2}(v_i) = H(v_i) \pm \sqrt{H(v_i)^2 - K(v_i)}.$$

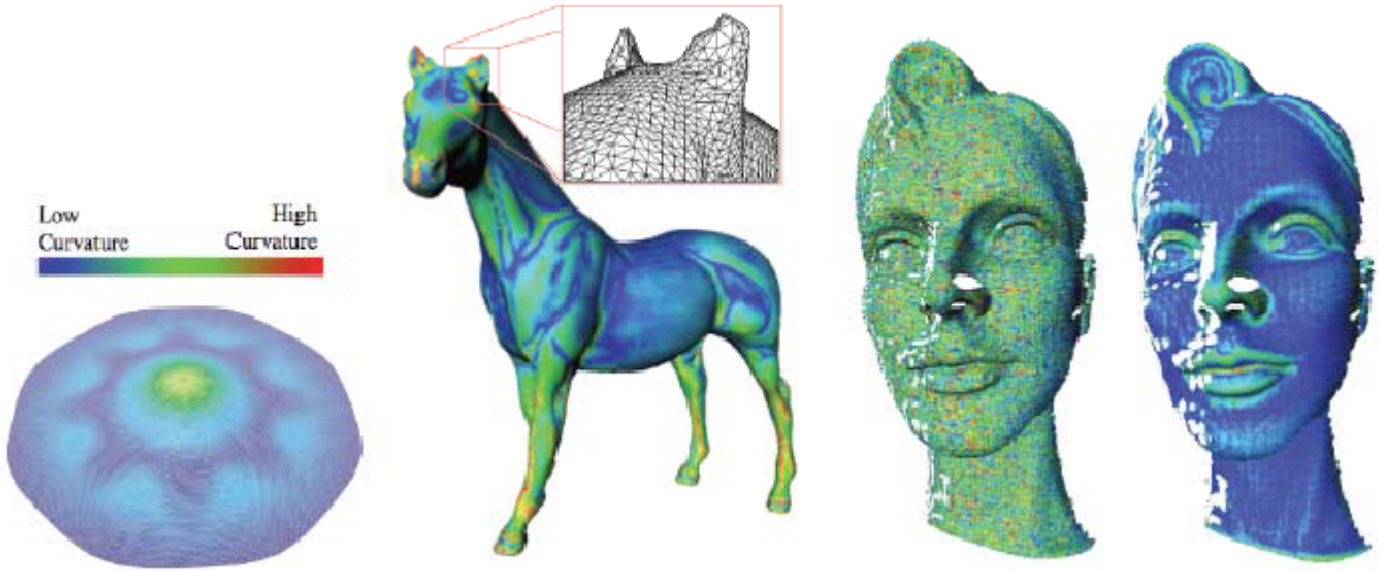
Curvature Computation

- Approaches:
  - Discrete differential geometry: Normal Cycle
  - Smooth differential geometry: Jet-fitting
- Implementation:
  - toolbox_mesh/compute_curvature
  - CGAL
  - MeshLab
  - 3d-workspace

Links and literature

- M. Meyer, M. Desbrun, P. Schroeder, A. Barr *Discrete Differential-Geometry Operators for Triangulated 2-Manifolds*, VisMath, 2002

- *[Hamann 93]* simple way to determine principal curvature and direction using least-squared paraboloid fitting
  - No easy way

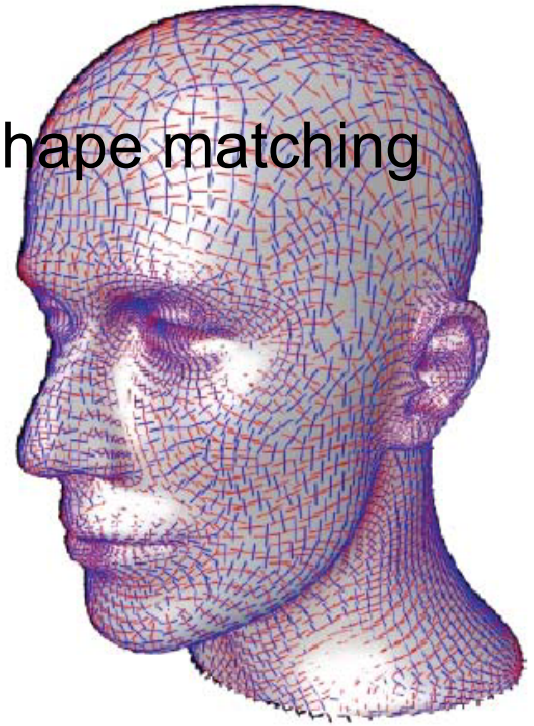


Links and literature

- P. Alliez, *Estimating Curvature Tensors on Triangle Meshes*, Source code!

- *[Taubin 95]* introduced a complete derivation of surface properties approximating curvature tensors for polyhedral surface

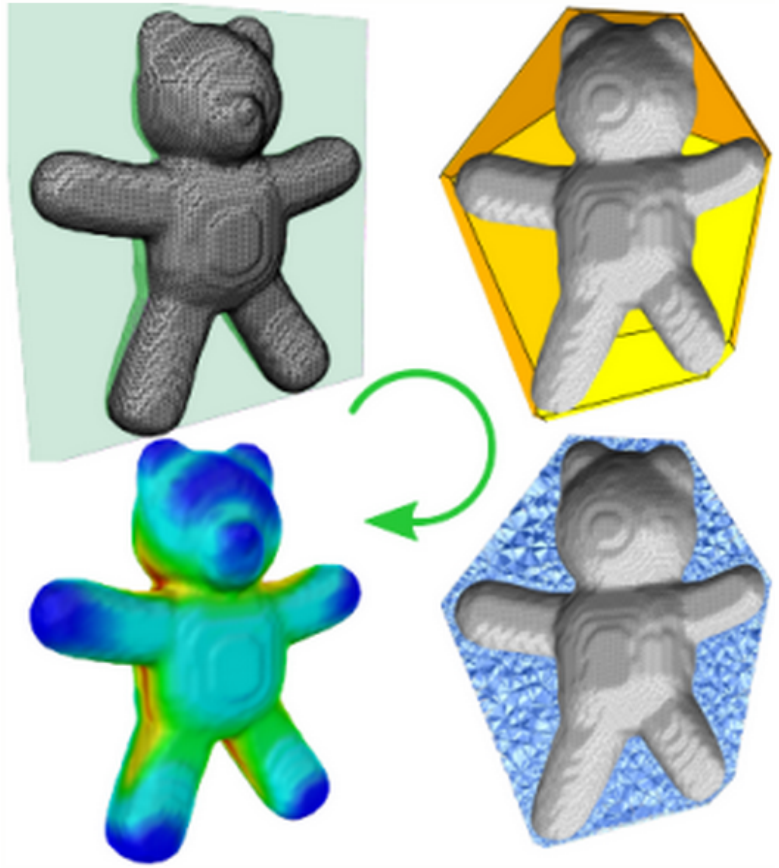
- Gaussian curvature: tog06_Salient geometric features for partial shape matching and similarity.



principal directions

Global concavity

- cvpr13_Efficient Computation of Shortest Path-Concavity for 3D Meshes, has source code



Discrete Surfaces

Note:

This discretization of curvature information and does not have to conform with others. Similarly, we can use the cotangent Laplacian and the fact that:

$$\Delta_S f = -2H\mathbf{n}$$

to define the mean curvature at a vertex, but this also won't agree with the mean-curvature defined by the curvature tensor.

How to compute it?

Scalar field & its gradient, Laplacian

Function on a mesh M={V,E,F}

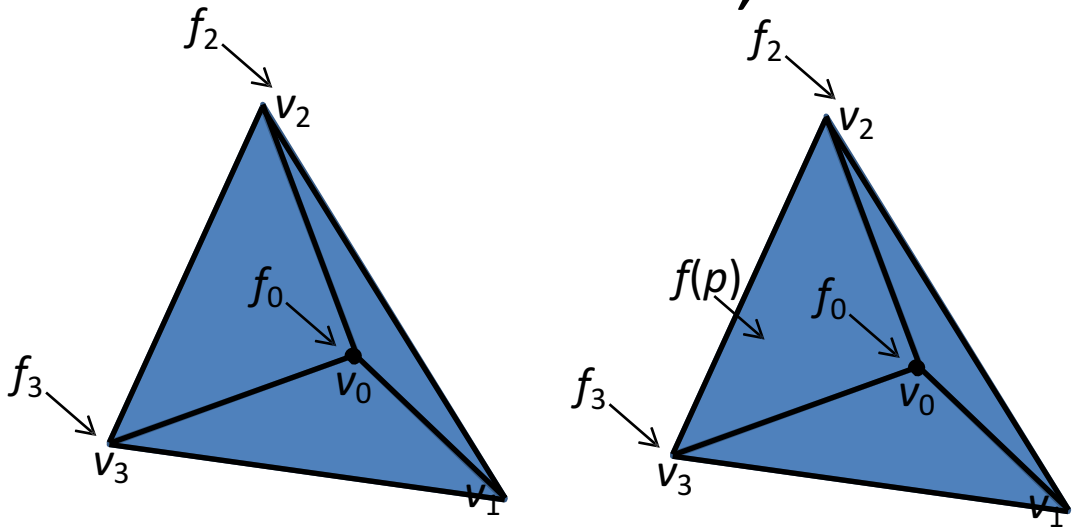
- A function is a discrete set of values or vectors defined at each vertex location. Then we have these two definitions,
- **Definition 2.1 (Scalar function on a mesh)**
  - *A scalar function **f** on a mesh with n vertices is a discrete set of values defined at each vertex, and can be viewed equivalently as an n-vector, that is*

$$f : \begin{cases} \mathcal{V} & \longrightarrow & \mathbb{R} \\ x_i & \longmapsto & f(x_i) \end{cases} \iff f : \begin{cases} V & \longrightarrow & \mathbb{R} \\ i & \longmapsto & f_i \end{cases} \iff f = (f_i)_{i \in V} \in \mathbb{R}^n.$$

- **Definition 2.2 (Vector function on a mesh)**
  - *A d-vector function on a mesh with n vertices is a discrete set of d-vectors defined at each vertex, and can be viewed equivalently as an n*d matrix, that is* $f = (f_i)_{i \in V} \in R^{n*d}$

Function on a mesh

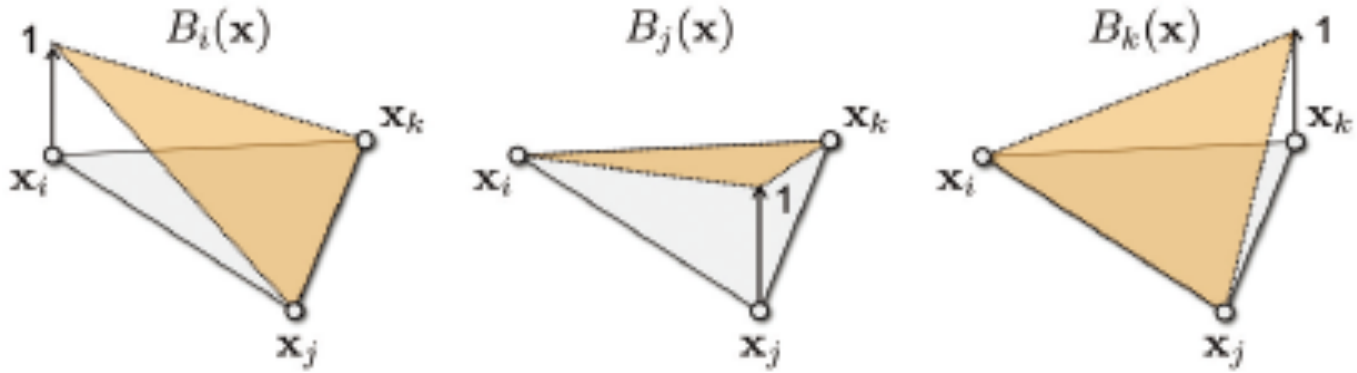
In considering functions on the a mesh, we will associate values with each vertex.

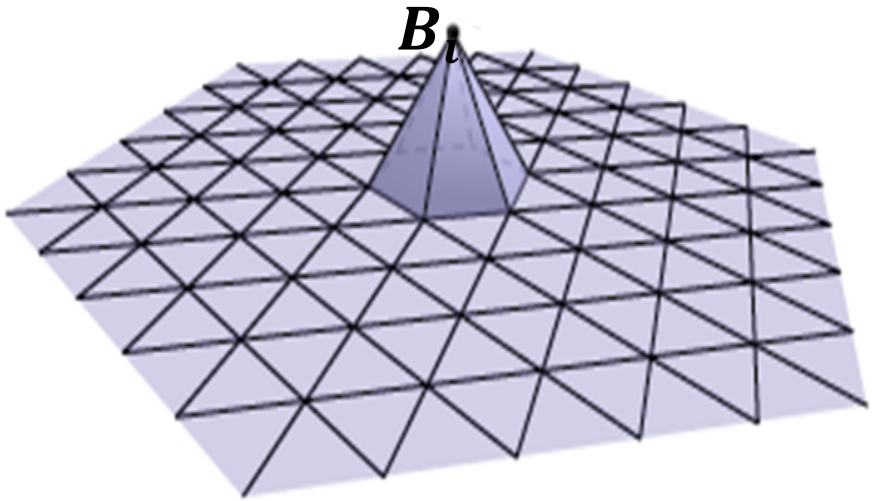


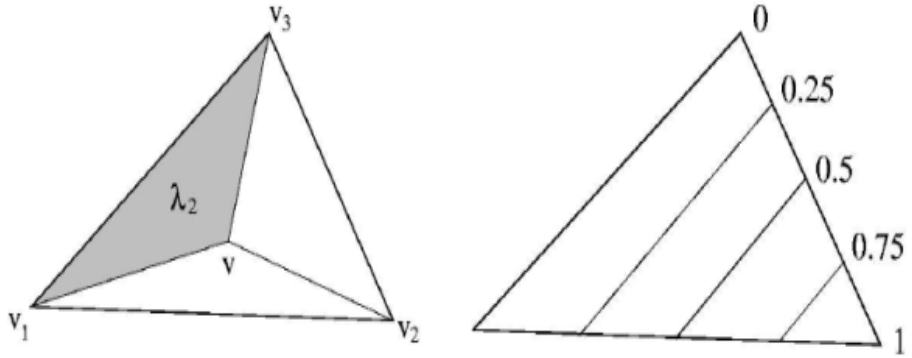
We then extend the functions to the interior of the triangles using barycentric interpolation:

$$f(p) = \sum f_i B_i(p)$$

The linear basis functions for barycentric interpolation on a triangle.

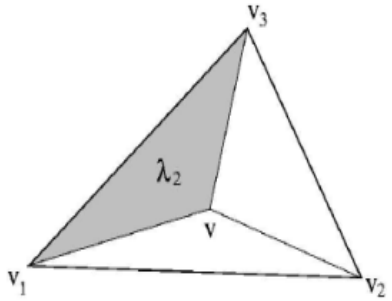


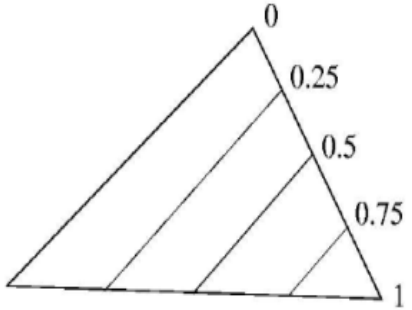




Triangle barycentric coordinates. Left: λ_2; Right: iso-curve of λ_2.

Triangle barycentric coordinates





- **Mobius [Mobius1827]** was the first to study ω_i and he defined ω_i as the barycentric coordinates of v.
- $\omega_1(v) = A(v, v_2, v_3) = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ x & x_2 & x_3 \\ y & y_2 & y_3 \end{vmatrix} = \frac{1}{2}[x(y_2 - y_3) + y(x_3 - x_2) + x_2y_3 - x_3y_2]$ is linear in v.

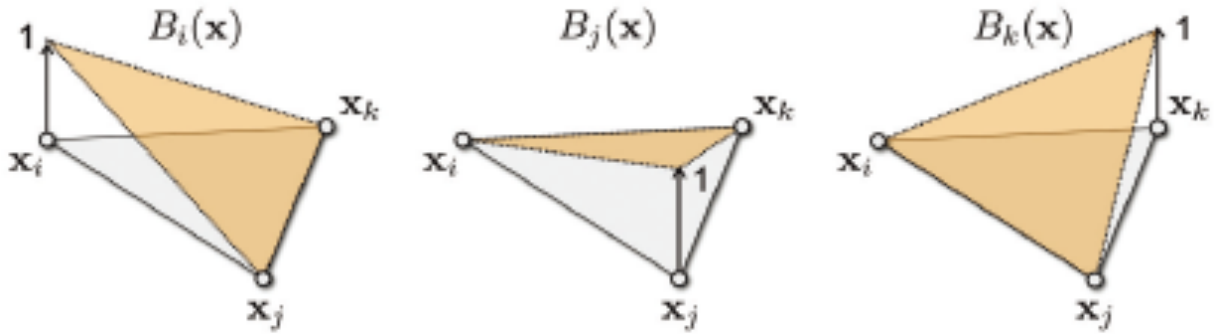
- add one other condition: $\sum_i \lambda_i = 1$.
- $\{\lambda_1 = \frac{A(v,v_2,v_3)}{A(v_1,v_2,v_3)}, \lambda_2 = \frac{A(v_1,v,v_3)}{A(v_1,v_2,v_3)}, \lambda_3 = \frac{A(v_1,v_2,v)}{A(v_1,v_2,v_3)}\}$ are called **Normalized barycentric coordinates** (NBC).

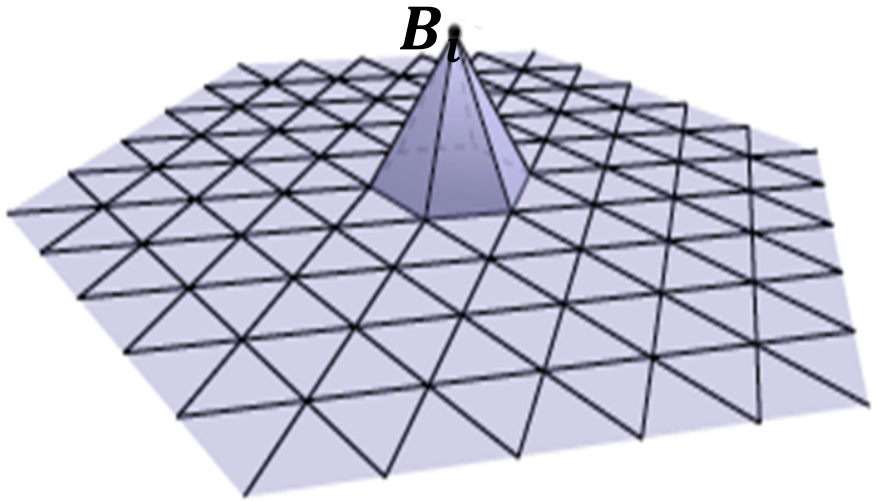
Function on mesh

- $f(\boldsymbol{u}) = \sum_i B_i(\boldsymbol{u}) f_i$,

$$f(v_i) = f(\mathbf{x}_i) = f(\mathbf{u}_i) = f_i$$

- Where **u**=(u,v) is the parameter pair corresponding to the surface point x in a 2D conformal parameterization induced by the triangle.





Approximating Integrals on a Mesh

Approximating Integrals on a Mesh

- In the continuous domain, **filtering** is defined through **integration** of functions over the mesh.

- In order to descretize integrals, one needs to define a partition of the mesh into small cells centered around a vertex or an edge.

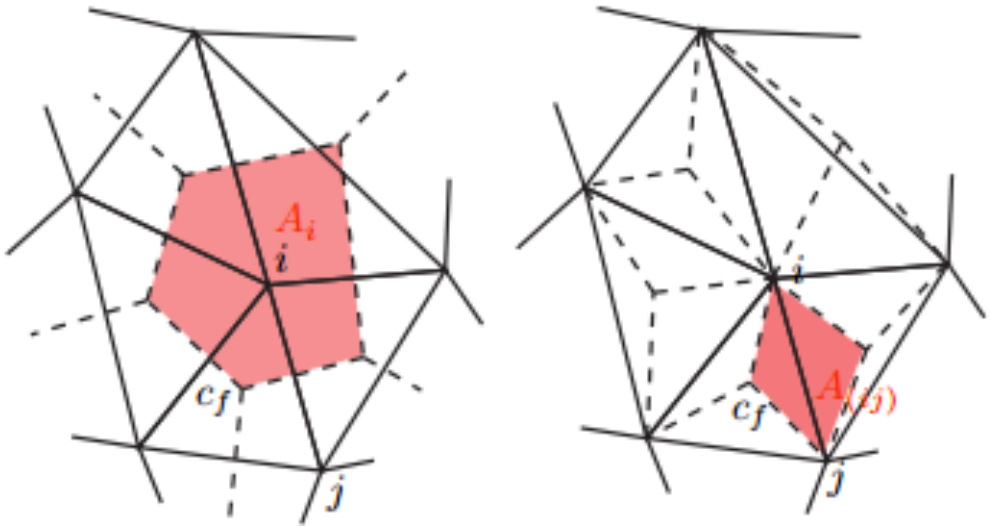
Approximating Integrals on a Mesh - Voronoi

Definition 10 (Vertices Voronoi). *The Voronoi diagram associated to the vertices is*

$$\forall i \in V, \quad E_i = \{x \in \mathcal{M} \setminus \forall j \neq i, \|x - x_i\| \leqslant \|x - x_j\|\}$$

Definition 11 (Edges Voronoi). *The Voronoi diagram associated to the edges is*

$$\forall e = (i,j) \in E, \quad E_e = \{x \in \mathcal{M} \setminus \forall e' \neq e, d(x,e) \leqslant d(x,e')\}$$



These Voronoi cells indeed form a partition of the mesh

$$\mathcal{M} = \bigcup_{i \in V} E_i = \bigcup_{e \in E} E_e.$$

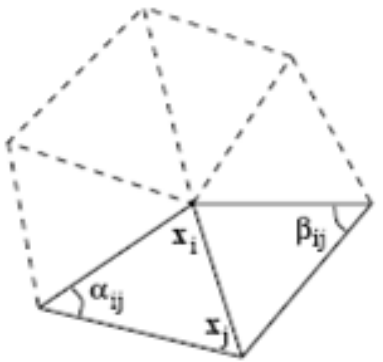
Meyer, 2003, **Course; *Discrete Differential-Geometry Operators for Triangulated 2-Manifolds***

Approximating Integrals on a Mesh

Theorem 1 (Voronoi area formulas). *For all* $e = (i, j) \in E$, $\forall\, i \in V$, *one has*

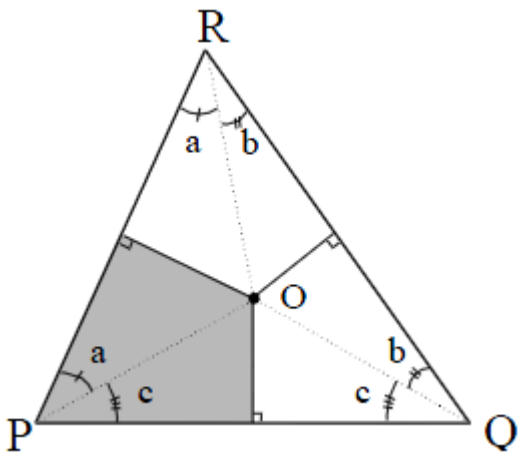
$$A_e = Area(E_e) = \frac{1}{2}\|x_i - x_j\|^2 \left(\cot(\alpha_{ij}) + \cot(\beta_{ij})\right)$$

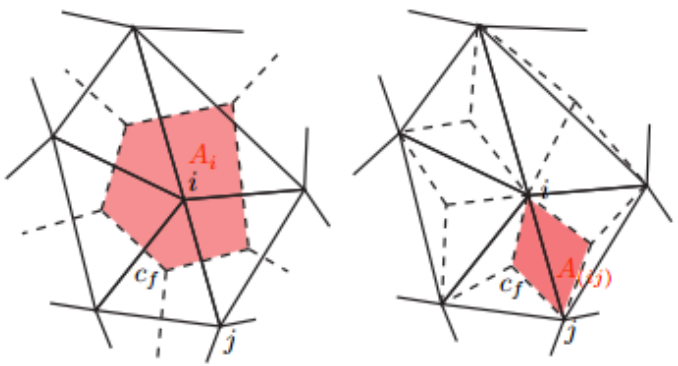
$$A_i = Area(E_i) = \frac{1}{2}\sum_{j \in N_i} A_{(ij)}.$$



With these areas, one can approximate integrals on vertices and edges using

$$\int_{\mathcal{M}} f(x)\mathrm{d}x \approx \sum_{i \in V} A_i\, f(x_i) \approx \sum_{e=(i,j) \in E} A_e\, f([x_i, x_j]).$$





Dirichlet's energy of a function ($\mathrm{f}: \mathrm{M} \to \mathrm{R}, \mathrm{M} \sqsubseteq R^n$) on a manifold:

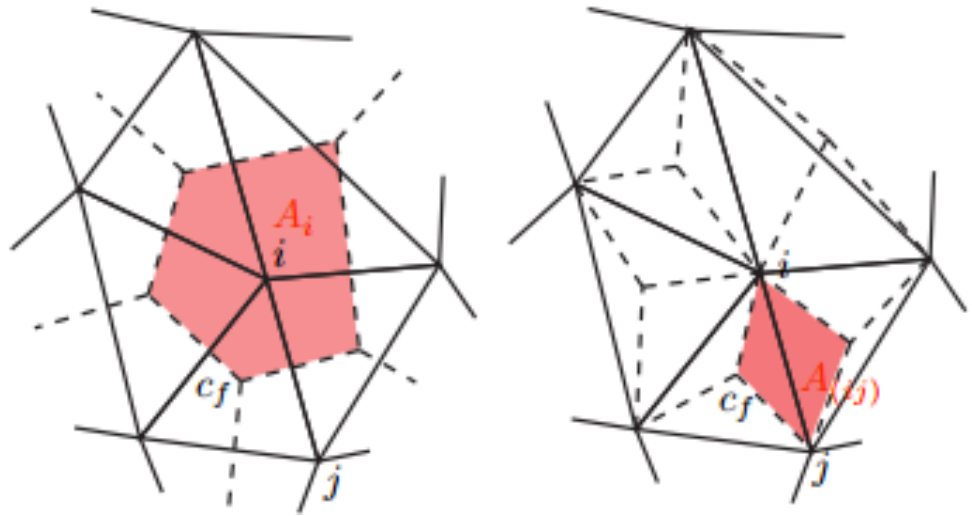
- **Dirichlet's energy** is a measure of how *variable* a function is.
- Solutions to $\Delta \mathrm{f} = 0$ are functions that make the Dirichlet energy functional stationary:

$$\mathrm{E_D(f)} = \frac{1}{2}\iint_{\mathrm{M}} |\nabla \mathrm{f}|^2 \mathrm{dudv}$$

- i.e. the Euler equation of the Dirichlet problem is a **Laplacian equation**: $\Delta \mathrm{f} = 0$

$$\int_{\mathcal{M}} f(x)\mathrm{d}x \approx \sum_{i\in V} A_i\, f(x_i) \approx \sum_{e=(i,j)\in E} A_e\, f([x_i, x_j]).$$

$$A_e = Area(E_e) = \frac{1}{2}\|x_i - x_j\|^2 \left(\cot(\alpha_{ij}) + \cot(\beta_{ij})\right)$$



$$\int_{\mathcal{M}} \|\nabla_x f\|^2 \mathrm{d}x \approx \sum_{(i,j)\in E} A_{(i,j)} \frac{|f(x_j) - f(x_i)|^2}{\|x_j - x_i\|^2}$$

$$= \sum_{(i,j)\subset E} w_{ij} |f(x_j) - f(x_i)|^2 \qquad \text{where} \qquad w_{ij} = \cot(\alpha_{ij}) + \cot(\beta_{ij}).$$

Approximate two vector fields

- M is a 2D manifold, and $\chi = (u(x,y,z), v(x,y,z))$ is an unknown map (vector function) defined on M. $G = \begin{pmatrix} g_1 \\ g_2 \end{pmatrix}$ is a known 2*2 **tensor field** defined on M, formed by two vector fields g_1 and g_2.
- If we want find a χ, using $\nabla\chi$ to approximate the tensor field G, we can get the following formula:
- $\min_{\chi} \int_M \|\nabla\chi - G\|^2 = \min_{(u,v)} \int_M \|\nabla u - g_1\|^2 + \|\nabla v - g_2\|^2 =$
  $\min_{(u,v)} \int_M F(u, v, \nabla u, \nabla v)$

Poisson Equation

- $\min_{\chi} \int_M \|\nabla\chi - G\|^2 = \min_{(u,v)} \int_M \|\nabla u - g_1\|^2 + \|\nabla v - g_2\|^2 =$
  $\min_{(u,v)} \int_M F(u, v, \nabla u, \nabla v)$

- The Euler equation of the above problem is:
- $\frac{\partial E(u,v)}{\partial u} = -\mathrm{div}(\nabla u - g_1) = 0, \frac{\partial E(u,v)}{\partial v} = -\mathrm{div}(\nabla v - g_1) = 0$
- i.e. a **Poisson Equation**
- $\begin{cases} \Delta u = \mathrm{div}(g_1) \\ \Delta v = \mathrm{div}(g_2) \end{cases}$ or $\Delta\chi = \mathrm{div}(G)$

Dirichlet energy

- When G is zero everywhere, we get the **Dirichlet energy**:
- $\mathrm{E_D}(\chi) = \frac{1}{2}\iint_{\mathrm{M}} \|\nabla\chi\|^2 \mathrm{ds}$
- The Euler equation of the Dirichlet problem is a **Laplacian equation**:
- $\Delta\chi = \mathrm{div}(\mathrm{G}) = 0$
- The **heat equation** is: $\mathrm{k}\Delta\chi = \frac{\partial\chi}{\partial \mathrm{t}}$

Variational: Euler equation

单元单标量函数	$$E(u) = \int_0^{`} F(x, u, u', u'')\, dx$$
	$$\frac{\partial F}{\partial u} - \frac{d}{dx}\left(\frac{\partial F}{\partial u'}\right) + \frac{d^2}{dx^2}\left(\frac{\partial F}{\partial u''}\right) = 0$$
多元单标量函数	$$E(u) = \iint_\Omega F(x, y, u, u_x, u_y, u_{xx}, u_{yy})\, dxdy$$
	$$\frac{\partial F}{\partial u} - \frac{d}{dx}\left(\frac{\partial F}{\partial u_x}\right) - \frac{d}{dy}\left(\frac{\partial F}{\partial u_y}\right) + \frac{d^2}{dx^2}\left(\frac{\partial F}{\partial u_{xx}}\right) + \frac{d^2}{dy^2}\left(\frac{\partial F}{\partial u_{yy}}\right) = 0$$ $$= \mathbf{F_u} - \mathbf{div}\left(\mathbf{F_{u_x}}, \mathbf{F_{u_y}}\right) + \Delta\left(\mathbf{F_{u_{xx}}}, \mathbf{F_{u_{yy}}}\right) = 0$$
多元多标量函数 **Multi multivariable (scalar) function**	$$E[u, v] = \int_\Omega F(x, y, u, u_x, u_y, v, v_x, v_y)dxdy$$
	$$\begin{cases} \mathbf{F_u} - \mathbf{div}\left(\mathbf{F_{u_x}}, \mathbf{F_{u_y}}\right) = 0 \\ \mathbf{F_v} - \mathbf{div}\left(\mathbf{F_{v_x}}, \mathbf{F_{v_y}}\right) = 0 \end{cases}$$

Operators on a mesh

Operators on a mesh M={V,E,F}

- **Definition 2.3 (Linear operator A)**
  - *A linear operator is defined as* $A = (a_{ij})_{i,j \in V} \in \mathbb{R}^{n \times n}\ (matrix).$
  - *And operate on a function as follow*

$$(Af)(x_i) = \sum_{j \in V} a_{ij} f(x_j) \iff \quad (Af)_i = \sum_{j \in V} a_{ij} f_j.$$

- **Definition2. 4 (Local operator)**
  - *A local operator* $W \in R^{n*n}$ *satisfies* $w_i = 0,\ if\ (i,j) \notin E$ *if , that is*

$$(\mathbf{W}f)_i = \sum_{(i,j) \in E} w_{ij} f_j$$

In most applications, we restrict our attention to local operators that can be conveniently stored as sparse matrices.

Gradient operator – for edges

$$\forall\,(i,j)\in E,\ i<j,\quad (Gf)_{(i,j)} \overset{\text{def.}}{=} \sqrt{w_{ij}}(f_j - f_i)\in\mathbb{R}.$$

$$w_{ij} = \|x_i - x_j\|^{-2},\quad (Gf)_{(i,j)} = \frac{f(x_j)-f(x_i)}{\|x_i - x_j\|}$$

which is exactly the finite difference discretization of a directional derivative.

Gradient operator – for vertices

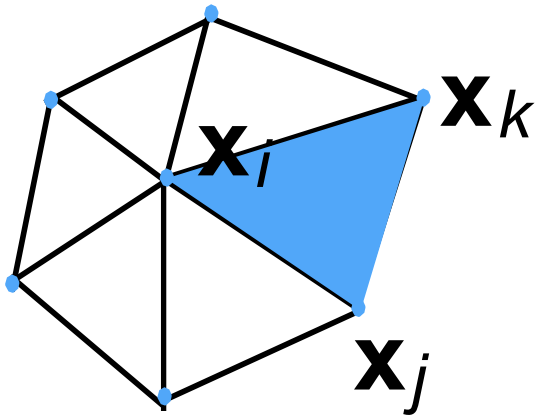
The gradient defined on a vertex as

$$\nabla_M^{(A)} f(p_i) = \frac{1}{\mathcal{A}(p_i)} \sum_{j \in N_1(i)} A_j \nabla_{T_j} f,$$

Where

$$\mathcal{A}(p_i) = \sum_{j \in N_1(i)} A_j .$$

Gradient operator – for faces

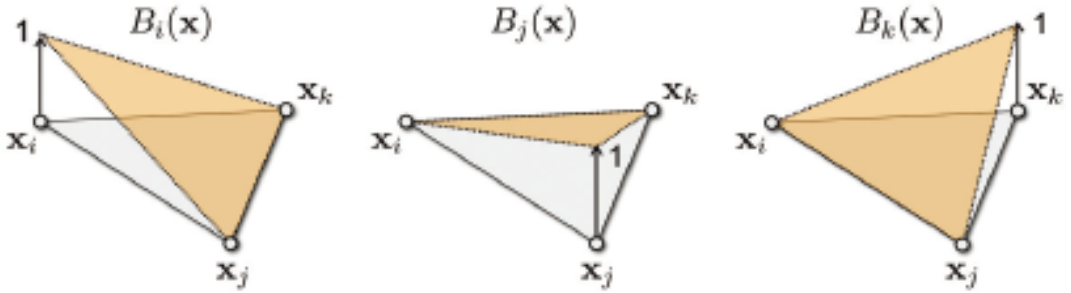


- $f(\boldsymbol{u}) = \sum_i B_i(\boldsymbol{u}) f_i$ **u** = (u, v)

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

gradient of linear function

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



partition of unity

$$B_i(\mathbf{u}) + B_j(\mathbf{u}) + B_k(\mathbf{u}) = 1 \quad \Rightarrow \quad \nabla B_i(\mathbf{u}) + \nabla B_j(\mathbf{u}) + \nabla B_k(\mathbf{u}) = 0.$$

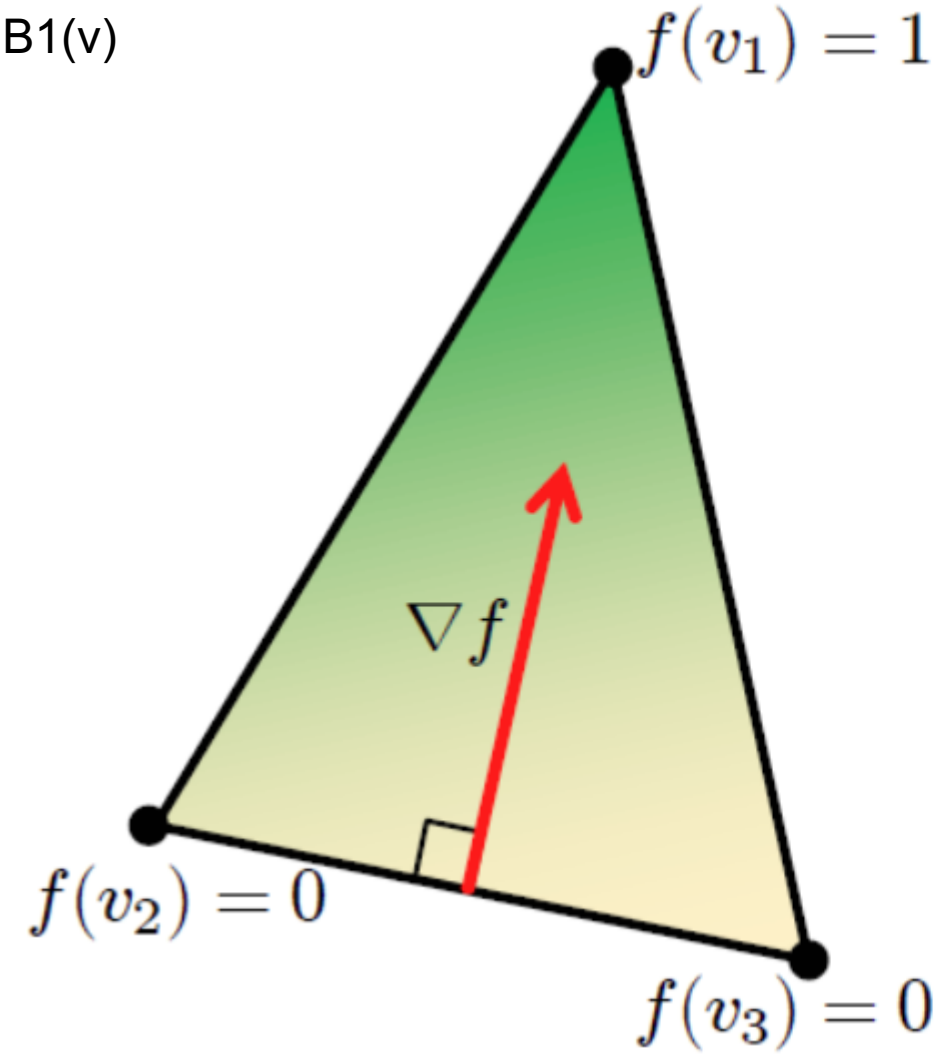
$$\nabla f(\mathbf{u}) = (f_j - f_i) \nabla B_j(\mathbf{u}) + (f_k - f_i) \nabla B_k(\mathbf{u})$$

$$\nabla B_i(\mathbf{u}) = \frac{(\mathbf{x}_k - \mathbf{x}_j)^{\perp}}{2 A_T},$$

linear basis functions for barycentric interpolation on a triangle

Direction of the Gradient

f (v):= B1(v)



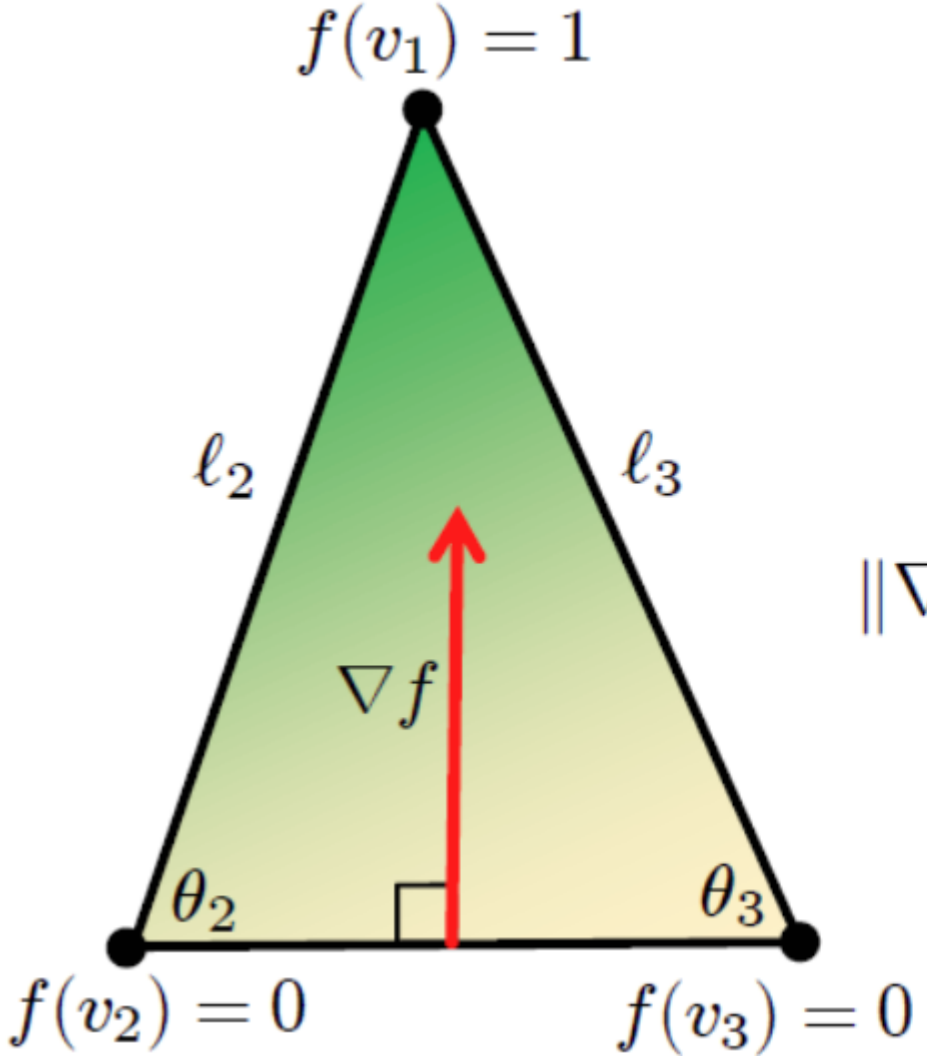
Linear along edges

$$\nabla f \cdot (v_1 - v_3) = 1$$

$$\nabla f \cdot (v_1 - v_2) = 1$$

$$\nabla f \cdot (v_2 - v_3) = 0$$

Magnitude of the gradient

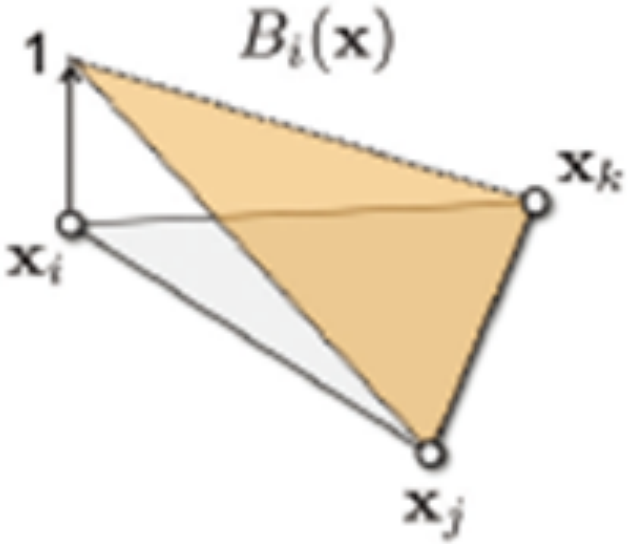


$$
\begin{aligned}
1 &= \nabla f \cdot (v_1 - v_3) \\
&= \|\nabla f\| \ell_3 \cos\left(\frac{\pi}{2} - \theta_3\right) \\
&= \|\nabla f\| \ell_3 \sin\theta_3 \\
\|\nabla f\| &= \frac{1}{\ell_3 \sin\theta_3} = \frac{1}{h}
\end{aligned}
$$

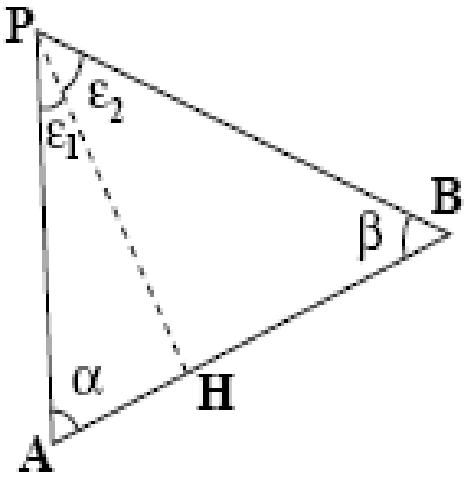
$$\nabla f = \frac{e_{23}^{\perp}}{2A}$$

Length of e_{23} cancels "base" in A

$$\nabla \boldsymbol{B_i}(\boldsymbol{u}) = \frac{(\mathrm{x}_k - \mathrm{x}_\mathrm{j})^{\perp}}{2A_T}$$

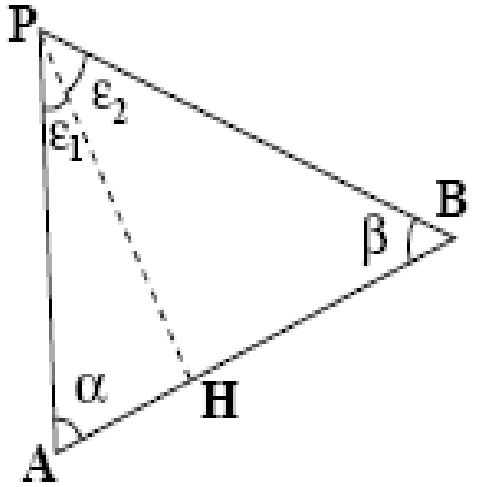


- $\boldsymbol{B_i}(\boldsymbol{u}) = \frac{\mathrm{A}(\mathrm{x},\mathrm{x}_j,\mathrm{x}_k)}{\mathrm{A}(\mathrm{x}_\mathrm{i},\mathrm{x}_j,\mathrm{x}_k)}$
- $\nabla \boldsymbol{B_i}(\boldsymbol{u}) = \frac{\nabla\mathrm{A}(\mathrm{x},\mathrm{x}_j,\mathrm{x}_k)}{\mathrm{A}(\mathrm{x}_\mathrm{i},\mathrm{x}_j,\mathrm{x}_k)}$
- $\nabla \mathrm{A}\left(\mathrm{x}, \mathrm{x}_j, \mathrm{x}_k\right) = \frac{(\mathrm{x}_k - \mathrm{x}_\mathrm{j})^{\perp}}{2}$
- $\mathrm{Area}\Delta\mathrm{PAB} = \mathrm{F(P)} = \frac{1}{2}|\mathrm{AB}||\mathrm{PH}|$
- $\nabla \mathrm{F} = \frac{1}{2}|\mathrm{AB}|\nabla|\mathrm{PH}| = \frac{1}{2}|\mathrm{AB}|\frac{\mathrm{HP}}{|\mathrm{PH}|} = \frac{1}{2}\mathrm{AB}^{90}$



$\nabla|\mathrm{PH}| = \frac{\mathrm{HP}}{|\mathrm{PH}|}$: Unit vector in HP direction

$$\nabla|\mathrm{PH}| = \frac{\mathrm{HP}}{|\mathrm{PH}|}$$

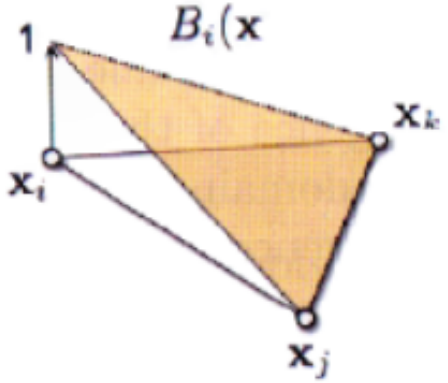


- Proof:
- $\mathrm{P(x,y)}$, $v_1 = \frac{\mathrm{AB}}{|\mathrm{AB}|} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$, $v_2 = {v_1}^{\perp} = \begin{pmatrix} -y_1 \\ x_1 \end{pmatrix}$,
- $\mathrm{H} = (v_1 \quad v_2)\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}(v_1 \quad v_2)^{\mathrm{T}}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} {x_1}^2 & x_1y_1 \\ x_1y_1 & {y_1}^2 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$
- $\mathrm{PH} = \mathrm{H} - \mathrm{P} = \begin{pmatrix} {x_1}^2 - 1 & x_1y_1 \\ x_1y_1 & {y_1}^2 - 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -{y_1}^2 & x_1y_1 \\ x_1y_1 & -{x_1}^2 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_1y_1y - {y_1}^2x \\ x_1y_1x - {x_1}^2y \end{pmatrix}$
- $|\mathrm{PH}| = \sqrt{(x_1y_1y - {y_1}^2x)^2 + (x_1y_1x - {x_1}^2y)^2}$
- $\frac{\partial|\mathrm{PH}|}{\partial x} = \frac{-(x_1y_1y - {y_1}^2x){y_1}^2 + (x_1y_1x - {x_1}^2y)x_1y_1}{\sqrt{(x_1y_1y - {y_1}^2x)^2 + (x_1y_1x - {x_1}^2y)^2}} = \frac{[(x_1y_1)^2 + (y_1y_1)^2]x - [x_1y_1(y_1)^2 + x_1y_1(x_1)^2]y}{|\mathrm{PH}|}$
- $= \frac{{y_1}^2x - x_1y_1y}{|\mathrm{PH}|} = \frac{\mathrm{HP}_x}{|\mathrm{PH}|}$
- $\nabla|\mathrm{PH}| = \frac{\mathrm{HP}}{|\mathrm{PH}|}$
- END.

Gradients

gradient of linear function $\nabla f(\mathbf{u}) = (f_j - f_i)\nabla B_j(\mathbf{u}) + (f_k - f_i)\nabla B_k(\mathbf{u})$

with appropriate normalization: $\nabla B_i(\mathbf{u}) = \dfrac{(\mathbf{x}_k - \mathbf{x}_j)^\perp}{2\,A_T}$



$$\nabla f(\mathbf{u}) = (f_j - f_i)\frac{(\mathbf{x}_i - \mathbf{x}_k)^\perp}{2A_T} + (f_k - f_i)\frac{(\mathbf{x}_j - \mathbf{x}_i)^\perp}{2A_T} \qquad f_i = f(\mathbf{x}_i)$$

discrete gradient of a piecewiese linear function within T

Laplacian Operators

Simple Curvature Discretization

gradient operator

mean curvature

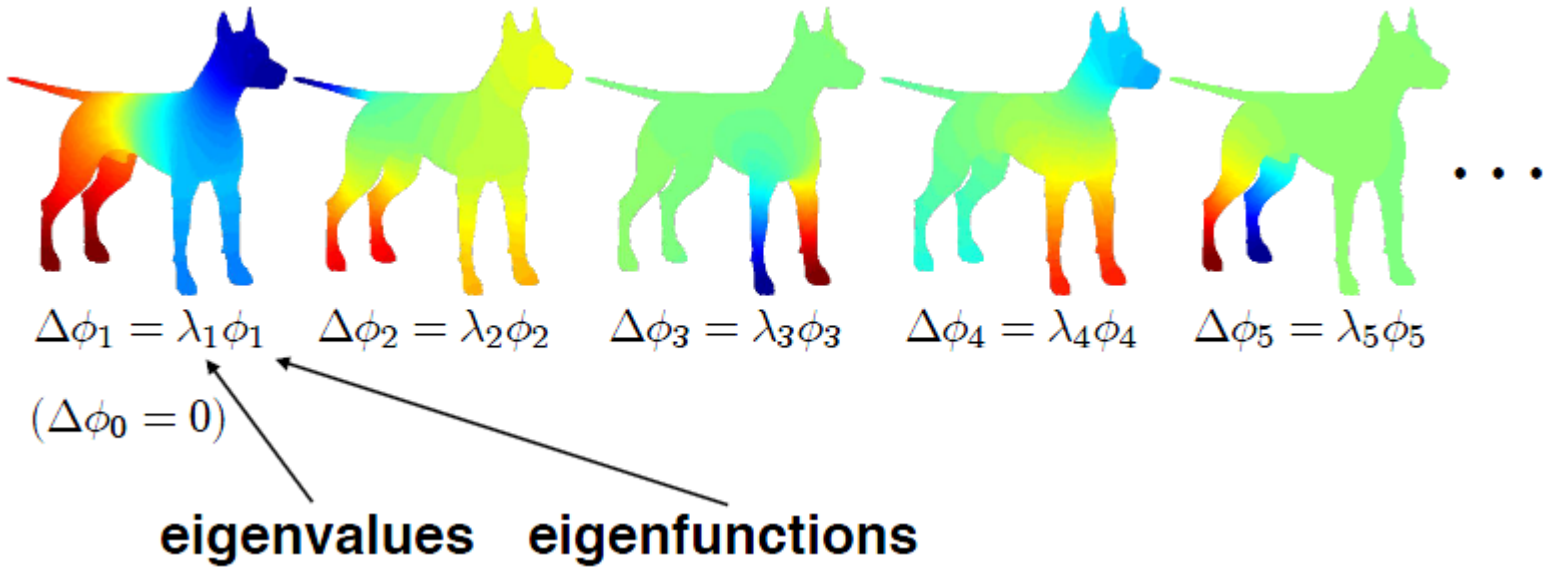
Laplace-Beltrami $\Delta_S x = \mathrm{div}_S \nabla_S x = -2Hn$

How to discretize?

Laplace-Beltrami Operator

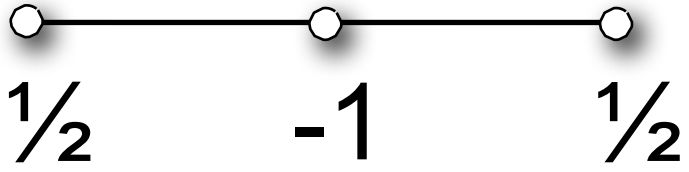
Functional basis on a surface

- Invariant to isometric deformations
- Has physics interpretation
  - Low-frequency to high-frequency

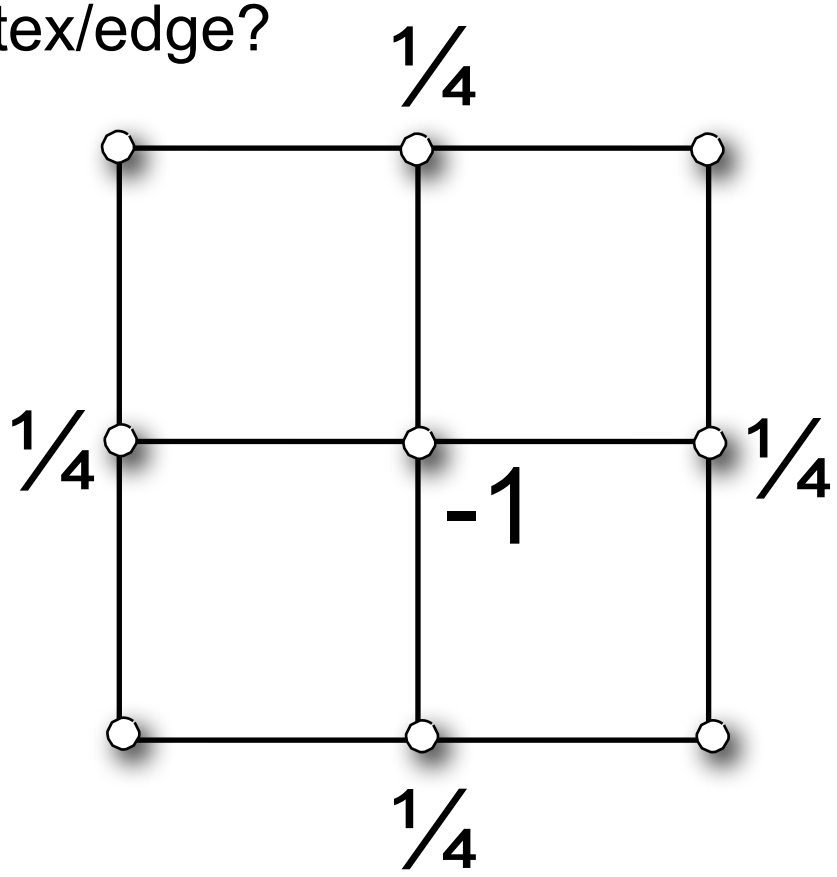


Laplace Operator on Meshes

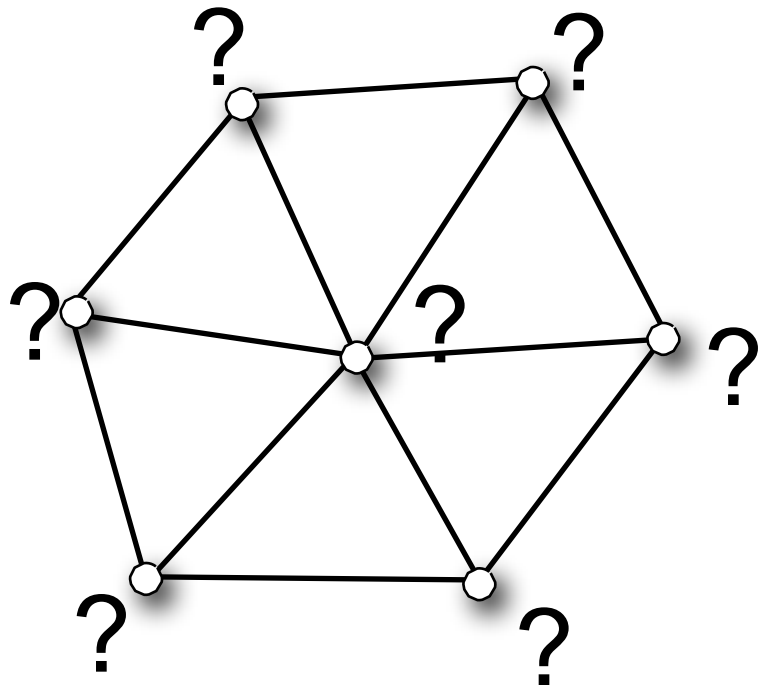
- **Extend finite differences to meshes?**
  - What weights per vertex/edge?



1D grid



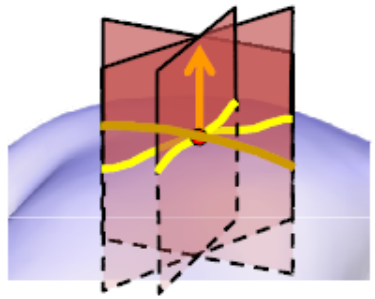
2D grid

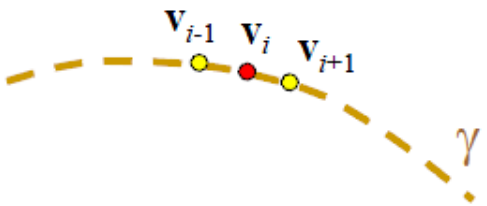


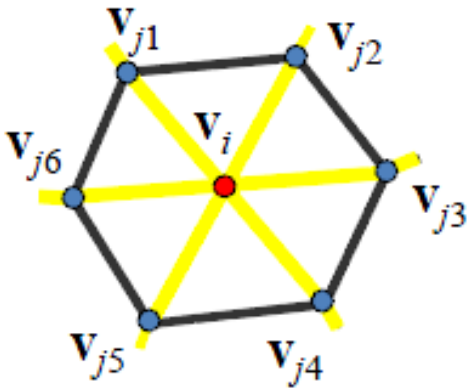
2D/3D grid

Uniform Laplace

- Intuition for uniform discretization







$$H = \frac{1}{2\pi}\int_0^{2\pi} \kappa(\varphi)\,\mathrm{d}\varphi$$

$$\kappa\mathbf{n} = \gamma''$$

$$\gamma'' \approx \frac{1}{t}\left((\mathbf{v}_i - \mathbf{v}_{i-1}) - (\mathbf{v}_{i+1} - \mathbf{v}_i)\right) = -\frac{1}{t}\left(\mathbf{v}_{i-1} + \mathbf{v}_{i+1} - 2\mathbf{v}_i\right)$$

$$\Delta_s \mathbf{x} = -2H\mathbf{n}$$

$$\begin{aligned} \mathbf{v}_{j1} + \mathbf{v}_{j4} - 2\mathbf{v}_i \quad &+ \\ \mathbf{v}_{j2} + \mathbf{v}_{j5} - 2\mathbf{v}_i \quad &+ \\ \mathbf{v}_{j3} + \mathbf{v}_{j6} - 2\mathbf{v}_i \quad &= \end{aligned}$$

$$6L(\mathbf{v}_i) = \sum_{k=1}^{6} \mathbf{v}_{jk} - 6\mathbf{v}_i \approx -6 \cdot 2H\mathbf{n}$$

Uniform Laplace

Uniform discretization

$$\Delta_{\text{uni}}\mathbf{x}_i := \frac{1}{|\mathcal{N}_1(v_i)|}\sum_{v_j \in \mathcal{N}_1(v_i)} (\mathbf{x}_j - \mathbf{x}_i) \approx -2H\mathbf{n}$$

Properties

- depends only on connectivity
- simple and efficient
- bad approximation for irregular triangulations
  - can give non-zero H for planar meshes
  - tangential drift for mesh smoothing

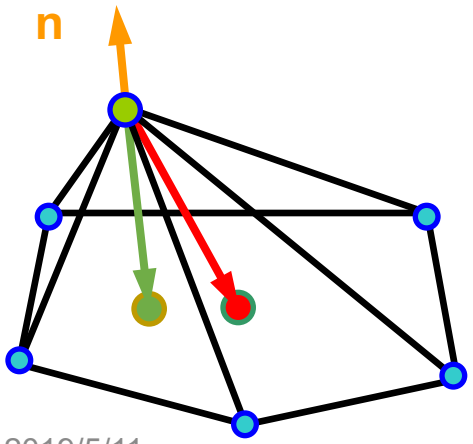
Discrete Laplacians

$$\delta_i = \mathbf{x}_i - \frac{1}{\sum_{(i,j)\in E} w_{ij}} \sum_{(i,j)\in E} w_{ij}\mathbf{x}_j$$

$\delta_{uniform}$: $w_{ij} = 1$

$\delta_{cotangent}$: $w_{ij} = \cot \alpha_{ij} + \cot \beta_{ij}$

$\Delta_{mean\ curvature}$: $w_{ij} = \frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2A(v_i)}$

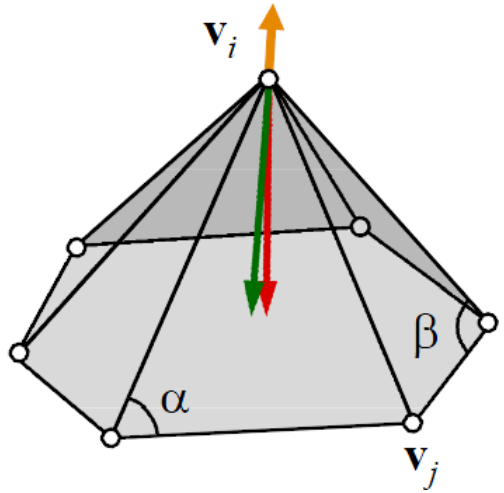


$$\Delta_S \mathbf{x} = -2H\mathbf{n}.$$

L v = δ

- For nearly equal edge lengths **Uniform** ≈ **Cotangent**
- While simple and efficient to compute, the resulting vector can be non-zero even for a planar configuration of vertices. However, in such a setting we would expect a zero Laplacian since the mean curvature over the entire mesh region is zero.

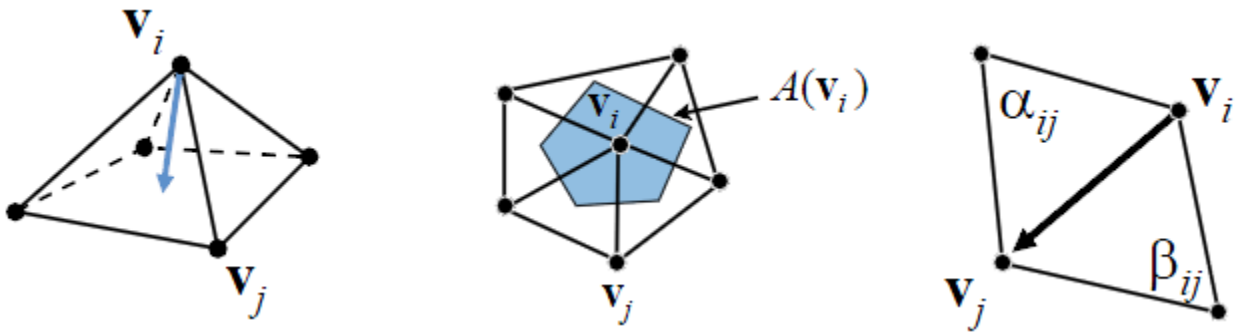
$$L_c(\mathbf{v}_i) = \frac{1}{2A(\mathbf{v}_i)} \sum_{\mathbf{v}_j \in N_1(\mathbf{v}_i)} (\cot \alpha_{ij} + \cot \beta_{ij})(\mathbf{v}_j - \mathbf{v}_i)$$



Discrete Laplacian by integration of Laplacian

- a mixed finite element/finite volume method [Meyer et al. 03]
- 2010_Polygon Mesh Processing

$$L_c(\mathbf{v}_i) = \frac{1}{2A(\mathbf{v}_i)} \sum_{\mathbf{v}_j \in N_1(\mathbf{v}_i)} (\cot \alpha_{ij} + \cot \beta_{ij})(\mathbf{v}_j - \mathbf{v}_i)$$

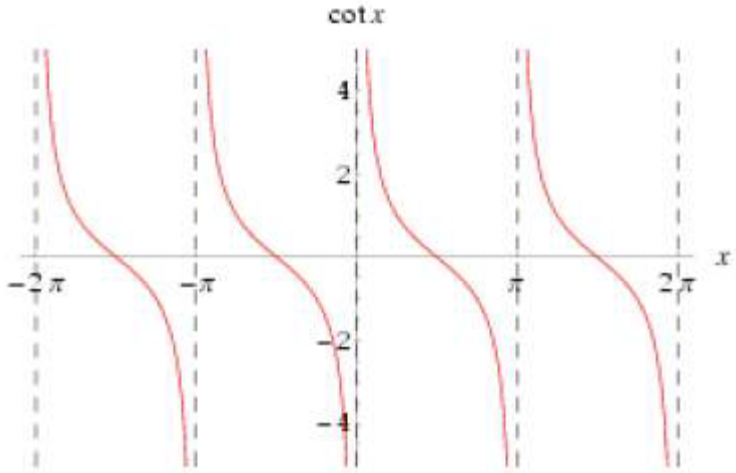


Discrete Laplace-Beltrami

- Cotangent formula

$$L_c(\mathbf{v}_i) = \frac{1}{2A(\mathbf{v}_i)} \sum_{\mathbf{v}_j \in N_1(\mathbf{v}_i)} (\cot \alpha_{ij} + \cot \beta_{ij})(\mathbf{v}_j - \mathbf{v}_i)$$

- Problems
  - Potentially negative weights
  - Depends on geometry



Discrete Laplacians – Cotangent formula

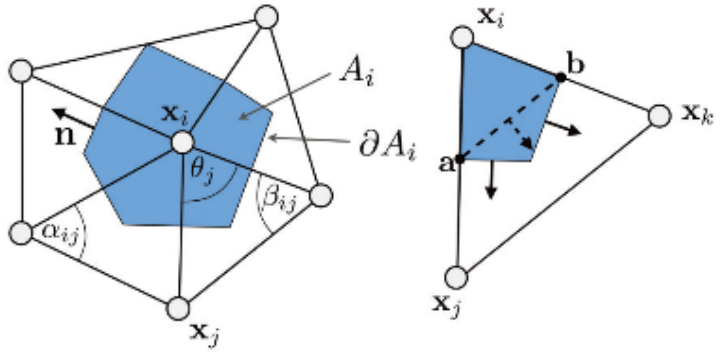
Gauss' theorem (divergence theorem) for a vector-valued function F

$$\int_{A_t} \operatorname{div} \mathbf{F}(\mathbf{u})\, dA = \int_{\partial A_t} \mathbf{F}(\mathbf{u}) \cdot \mathbf{n}(\mathbf{u})\, ds.$$

$$\int_{A_i} \Delta f(\mathbf{u})\, dA = \int_{A_i} \operatorname{div} \nabla f(\mathbf{u})\, dA = \int_{\partial A_i} \nabla f(\mathbf{u}) \cdot \mathbf{n}(\mathbf{u})\, ds.$$

$$\int_{\partial A_i \cap T} \nabla f(\mathbf{u}) \cdot \mathbf{n}(\mathbf{u}) ds = \nabla f(\mathbf{u}) \cdot (\mathbf{a} - \mathbf{b})^{\perp}$$

$$= \frac{1}{2} \nabla f(\mathbf{u}) \cdot (\mathbf{x}_j - \mathbf{x}_k)^{\perp}$$



Gradient is constant in T &
Normal of point (x,y,z) of the surface g(x,y,z)=0 is the gradient of g.
Gradient field is Conservative vector field
Integral of a conservative field over a closed path is zero.

$$\nabla f(\mathbf{u}) = (f_j - f_i) \nabla B_j(\mathbf{u}) + (f_k - f_i) \nabla B_k(\mathbf{u}) \qquad \nabla B_i(\mathbf{u}) = \frac{(\mathbf{x}_k - \mathbf{x}_j)^{\perp}}{2 A_T},$$

- F is a vector field $\xrightarrow{\text{Path independence}}$ F is a Gradient field of a scalar function f$\Leftrightarrow$F is a conservative Field=> F is cur-free: $\oint_C \mathbf{F} \cdot d\mathbf{r} = \iint_R \nabla \times \mathbf{F} \cdot \mathbf{k} dA = 0$=> $\forall p \in R, \text{curl}(F) = 0$=>$\forall p \in R, \text{curl}(\nabla f) = 0$
- (f is called potential function of the vector field F)

Unifying the Integral Theorems

Green's Theorem and Its Generalization to Three Dimensions

Normal form of Green's Theorem: $\oint_C \mathbf{F} \cdot \mathbf{n}\, ds = \int\int_R \bigtriangledown \cdot \mathbf{F}\, dA$

Divergence Theorem: $\int\int_S \mathbf{F} \cdot \mathbf{n}\, d\sigma = \int\int\int_D \bigtriangledown \cdot \mathbf{F}\, dV$

Tangential form of Green's Theorem: $\oint_C \mathbf{F} \cdot d\,\mathbf{r} = \int\int_R \bigtriangledown \times \mathbf{F} \cdot \mathbf{k}\, dA$

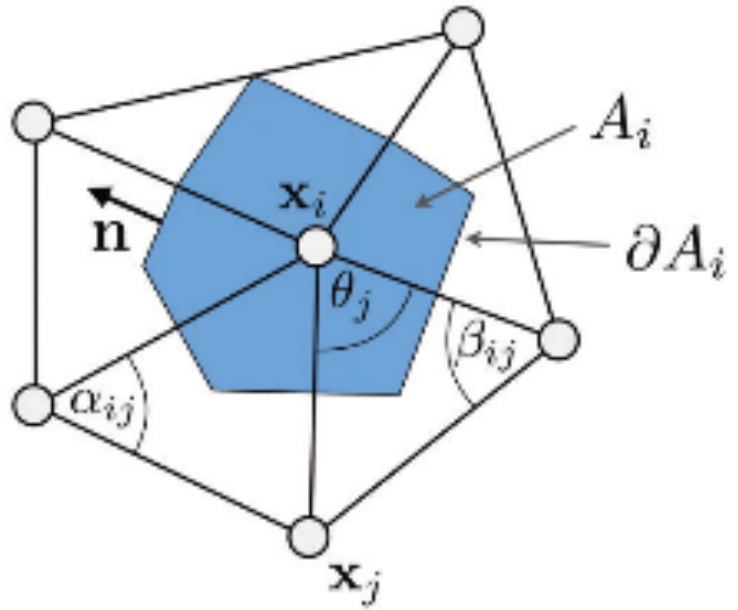
Stokes' Theorem: $\oint_C \mathbf{F} \cdot d\,\mathbf{r} = \int\int_S \bigtriangledown \times \mathbf{F} \cdot \mathbf{n}\, d\sigma$

$$\int_{\partial A_i \cap T} \nabla f(\mathbf{u}) \cdot \mathbf{n}(\mathbf{u}) \mathrm{d}s \;=\; \nabla f(\mathbf{u}) \cdot (\mathbf{a}-\mathbf{b})^{\perp} = \frac{1}{2} \nabla f(\mathbf{u}) \cdot (\mathbf{x}_j - \mathbf{x}_k)^{\perp}$$

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

$$\nabla B_i(\mathbf{u}) \;=\; \frac{(\mathbf{x}_k - \mathbf{x}_j)^{\perp}}{2\,A_T},$$

$$\int_{\partial A_i \cap T} \nabla f(\mathbf{u}) \cdot \mathbf{n}(\mathbf{u}) \mathrm{d}s \;=\; (f_j - f_i) \frac{(\mathbf{x}_i - \mathbf{x}_k)^{\perp} \cdot (\mathbf{x}_j - \mathbf{x}_k)^{\perp}}{4A_T} + (f_k - f_i) \frac{(\mathbf{x}_j - \mathbf{x}_i)^{\perp} \cdot (\mathbf{x}_j - \mathbf{x}_k)^{\perp}}{4A_T}.$$



Let γ_j, γ_k denote the inner triangle angles at vertices v_j, v_k, respectively. Since $A_T = \frac{1}{2}\sin\gamma_j \, \|\mathbf{x}_j - \mathbf{x}_i\| \, \|\mathbf{x}_j - \mathbf{x}_k\| = \frac{1}{2}\sin\gamma_k \, \|\mathbf{x}_i - \mathbf{x}_k\| \, \|\mathbf{x}_j - \mathbf{x}_k\|$, and $\cos\gamma_j = \frac{(\mathbf{x}_j - \mathbf{x}_i)\cdot(\mathbf{x}_j - \mathbf{x}_k)}{\|\mathbf{x}_j - \mathbf{x}_i\| \|\mathbf{x}_j - \mathbf{x}_k\|}$ and $\cos\gamma_k = \frac{(\mathbf{x}_i - \mathbf{x}_k)\cdot(\mathbf{x}_j - \mathbf{x}_k)}{\|\mathbf{x}_i - \mathbf{x}_k\| \|\mathbf{x}_j - \mathbf{x}_k\|}$, this expression simplifies to

$$\int_{\partial A_i \cap T} \nabla f(\mathbf{u}) \cdot \mathbf{n}(\mathbf{u}) \mathrm{d}s \;=\; \frac{1}{2}\left(\cot\gamma_k (f_j - f_i) + \cot\gamma_j (f_k - f_i)\right).$$

$$\int_{A_i} \Delta f(\mathbf{u}) \mathrm{d}A \;=\; \frac{1}{2} \sum_{v_j \in \mathcal{N}_1(v_i)} (\cot\alpha_{i,j} + \cot\beta_{i,j})(f_j - f_i),$$

$$\Delta f(v_i) \;:=\; \frac{1}{2A_i} \sum_{v_j \in \mathcal{N}_1(v_i)} \left(\cot\alpha_{i,j} + \cot\beta_{i,j}\right)(f_j - f_i).$$

Discrete Laplace-Beltrami

Cotangent discretization

$$\Delta_{\mathcal{S}} f(v) := \frac{1}{2A(v)} \sum_{v_i \in \mathcal{N}_1(v)} (\cot \alpha_i + \cot \beta_i)\,(f(v_i) - f(v))$$

Problems

- weights can become negative
- depends on triangulation

Still the most widely used discretization

Discretizing the Laplacian

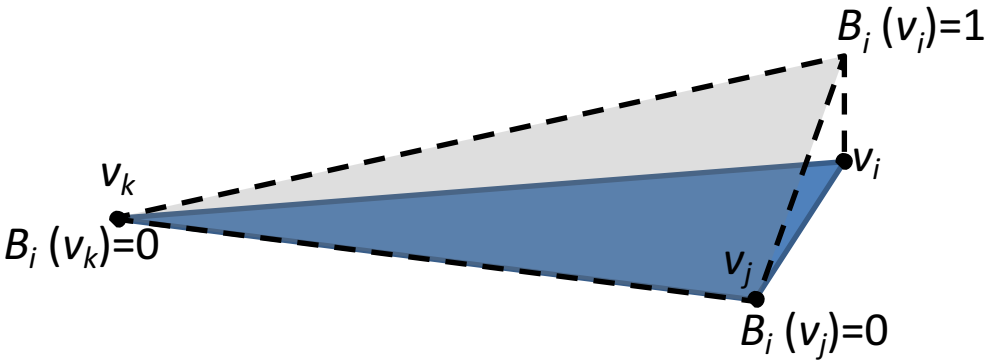
Since the Laplace-Beltrami operator is linear, to compute the Laplacian of f it suffices to be able to compute the Laplacians of $B_i(p)$:

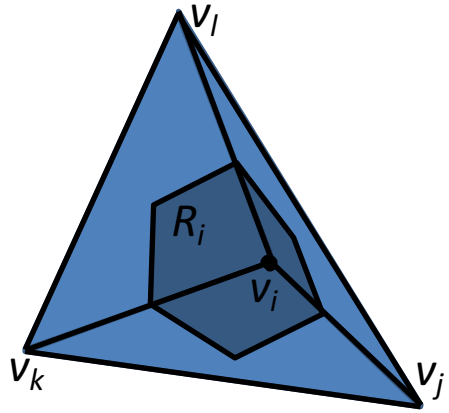
$$\Delta f(p) = \sum_i f_i \Delta B_i(p)$$

Discretizing the Laplacian on vertex

Since also want to represent the Laplacian of *f* just by prescribing vertex values, we will set the Laplacian of *f* at vertex *v* to be the average of the Laplacian in a neighborhood around *v*:

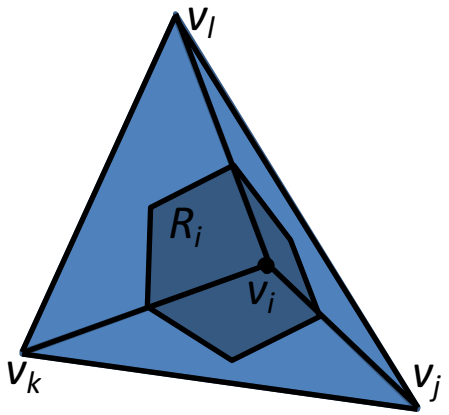
$$\left(\Delta B_j\right)_i = \frac{1}{|R_i|}\int_{R_i} \Delta B_j(r)\,dr = \int_{R_i} \operatorname{div}\left(\nabla B_j(r)\right)dr$$





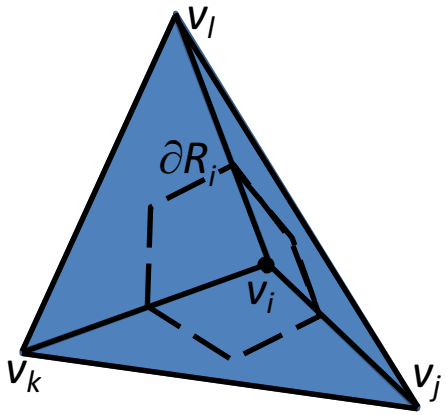
Discretizing the Laplacian on vertex

$$\left(\Delta B_j\right)_i = \frac{1}{|R_i|}\int_{R_i} \Delta B_j(r)\,dr = \int_{R_i} \operatorname{div}\left(\nabla B_j(r)\right)dr$$



But by the definition of the divergence, this is just the integral over the boundary:

$$\left(\Delta B_j\right)_i = \frac{1}{|R_i|}\int_{\partial R_i} \left\langle \nabla B_j(r), n_r \right\rangle dr$$

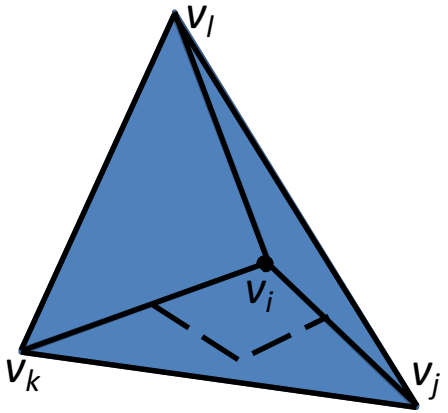


Discretizing the Laplacian

$$\left(\Delta B_j\right)_i = \frac{1}{|R_i|}\int_{\partial R_i}\left\langle \nabla B_j(r), n_r \right\rangle dr$$

Breaking up the R_i per triangle, we get:

$$\left(\Delta B_j\right)_i = \frac{1}{|R_i|}\sum_{v_i \in T}\int_{\partial R_i \cap T}\left\langle \nabla B_j(r), n_r \right\rangle dr$$



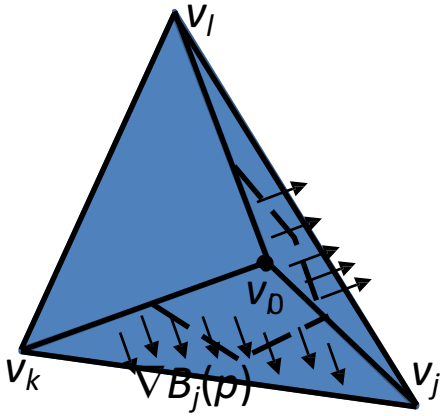
Discretizing the Laplacian

$$\left(\Delta B_j\right)_i = \frac{1}{|R_i|}\sum_{v_i \in T}\int_{\partial R_i \cap T}\left\langle \nabla B_j(r), n_r \right\rangle dr$$

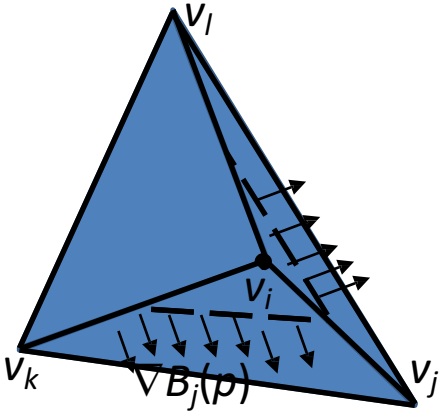
Computing the gradient of B_j we get:

$$\nabla B_j(p) = \frac{(v_i - v_k)^{\perp}}{2\text{Area}(T_{ijk})}$$

for all p in triangle T_{ijk}.



Note that since the gradient is constant over the interior of the triangle, the integral is path independent.



Discretizing the Laplacian

$$(\Delta B_j)_i = \frac{1}{|R_i|} \sum_{v_i \in T} \int_{\partial R_i \cap T} \langle \nabla B_j(r), n_r \rangle dr$$

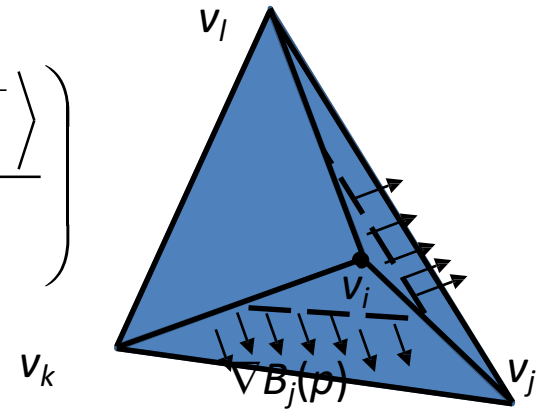
Computing the gradient of B_j we get:

$$\nabla B_j(p) = \frac{(v_i - v_k)^\perp}{2\text{Area}(T_{ijk})}$$

for all p in triangle T_{ijk}.

Computing the integral over the new boundary gives:

$$(\Delta B_j)_i = \frac{1}{|R_i|} \left(\frac{\langle (v_j - v_k)^\perp, (v_k - v_i)^\perp \rangle}{4\text{Area}(T_{ikj})} + \frac{\langle (v_l - v_j)^\perp, (v_i - v_l)^\perp \rangle}{4\text{Area}(T_{ijl})} \right)$$

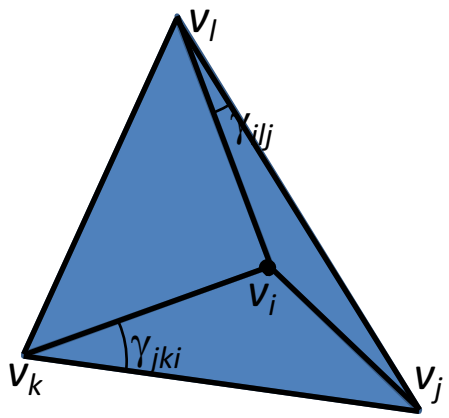


Discretizing the Laplacian

$$(\Delta B_j)_i = \frac{1}{|R_i|}\left(\frac{\langle (v_j - v_k)^{\perp}, (v_k - v_i)^{\perp}\rangle}{4\mathrm{Area}(T_{ikj})} + \frac{\langle (v_l - v_j)^{\perp}, (v_i - v_l)^{\perp}\rangle}{4\mathrm{Area}(T_{ijl})}\right)$$

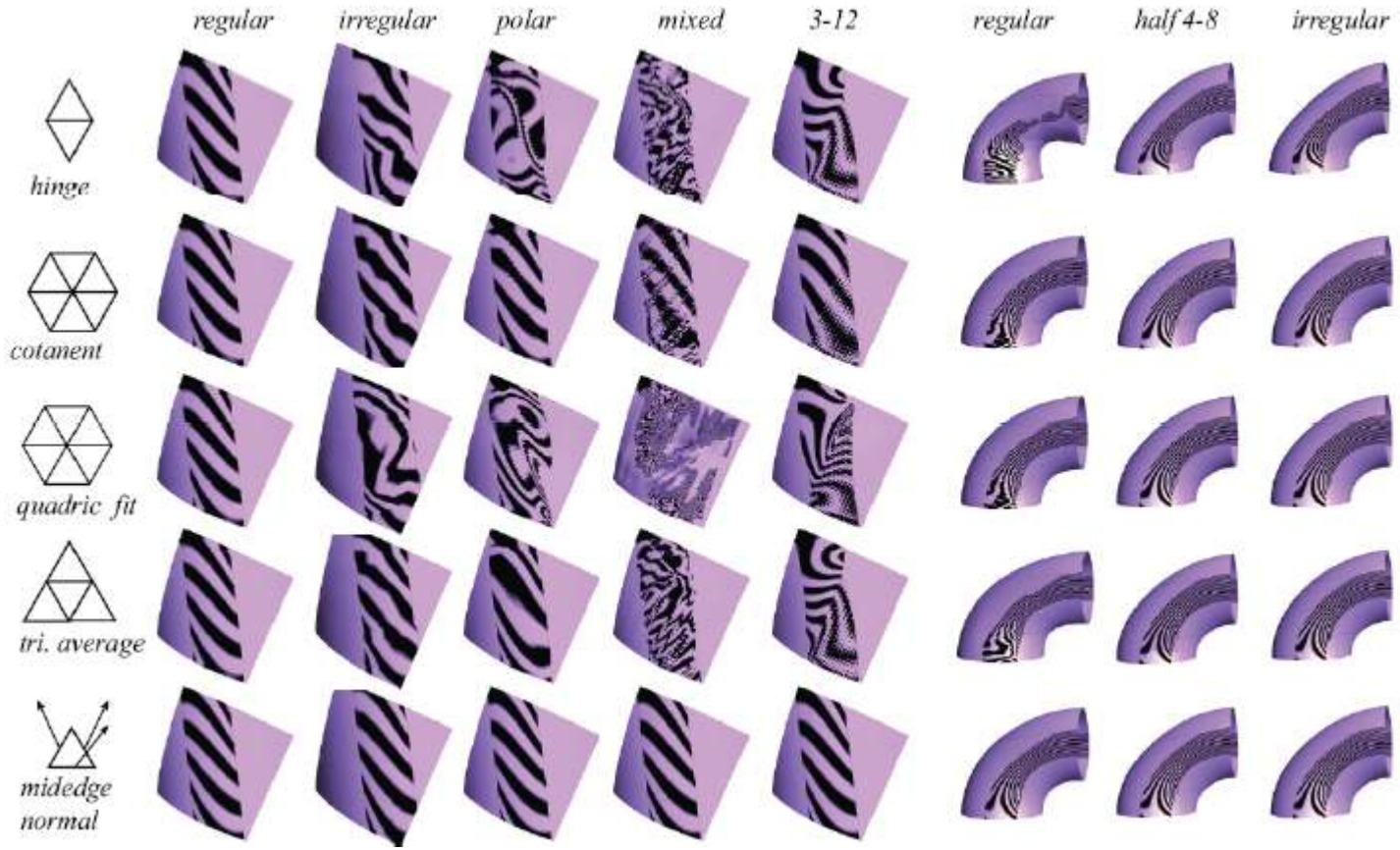
With a little bit of trigonometric manipulation, this gives:

$$(\Delta B_j)_i = \frac{1}{2|R_i|}(\cot\gamma_{jki} + \cot\gamma_{ilj})$$



Links and literature

- Grinspun et al.: *Computing discrete shape operators on general meshes, Eurographics 2006*



The Taste of Mathematics

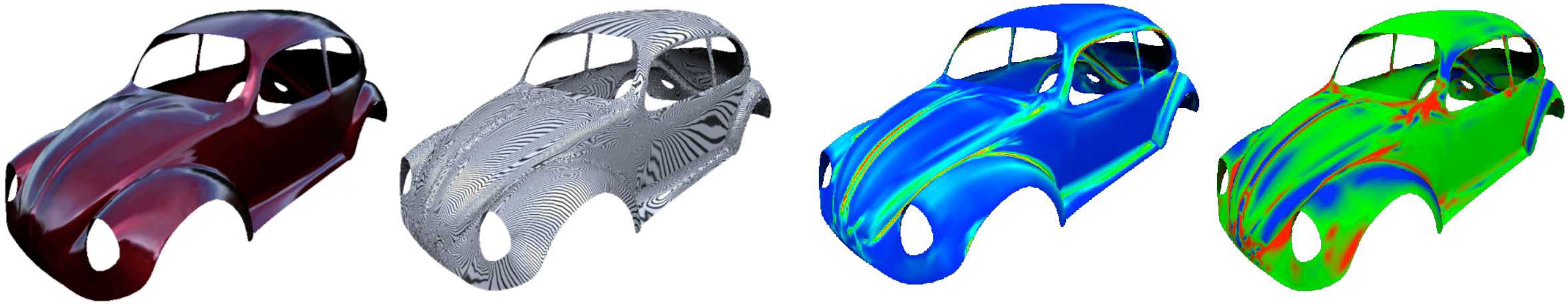
- Discrete Exterior Calculus_03_thesis
- Guoliang Xu, Convergent Discrete Laplace-Beltrami Operators over Triangular Surfaces

Mesh Quality Measures

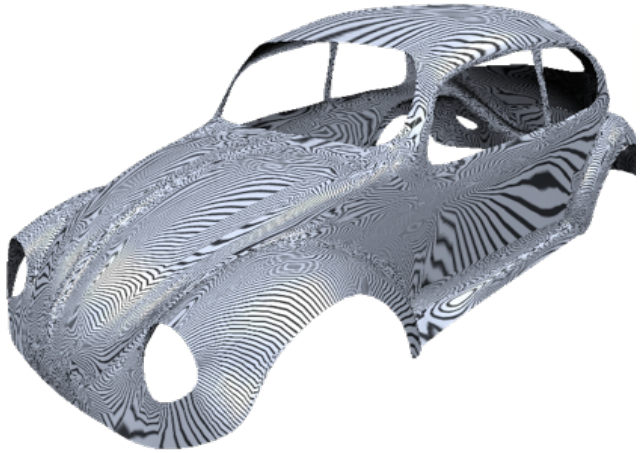
Mesh Quality

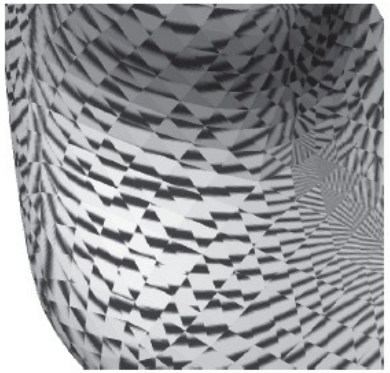
Visual inspection of “sensitive” attributes

- Specular shading
- Reflection lines
- Curvature
  - Mean curvature, Gauss curvature



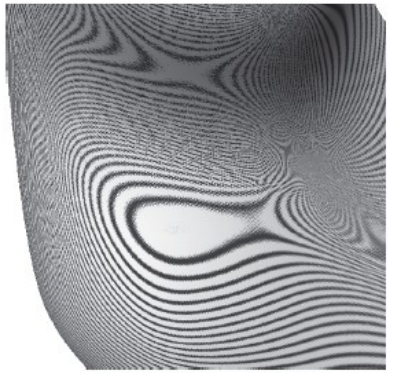
Reflection lines as an inspection tool



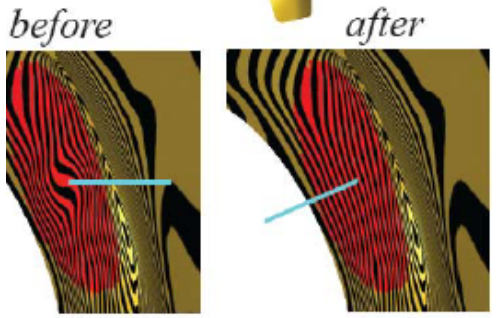


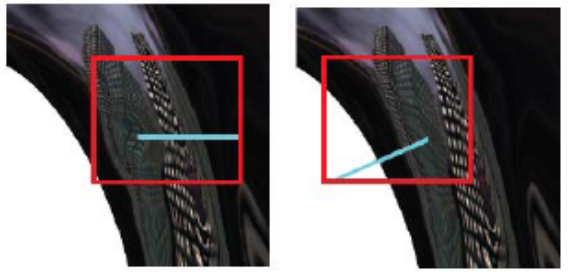
C^0

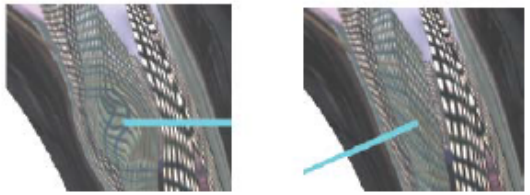
C^1



C^2





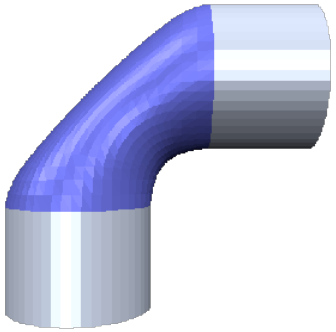


Shape optimization using reflection lines
E. Tosun, Y. I. Gingold, J. Reisman, D. Zorin
Symposium on Geometry Processing 2007

Mesh Quality Criteria

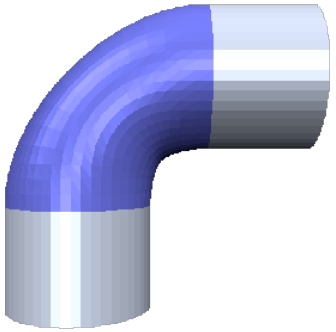
- Smoothness
  - continuous differentiability of a surface (Ck)

- Fairness
  - aesthetic measure of “well-shapedness”
  - principle of simplest shape
  - fairness measures from physical models



$$\int_{\mathcal{S}} \kappa_1^2 + \kappa_2^2 \, dA$$

strain energy



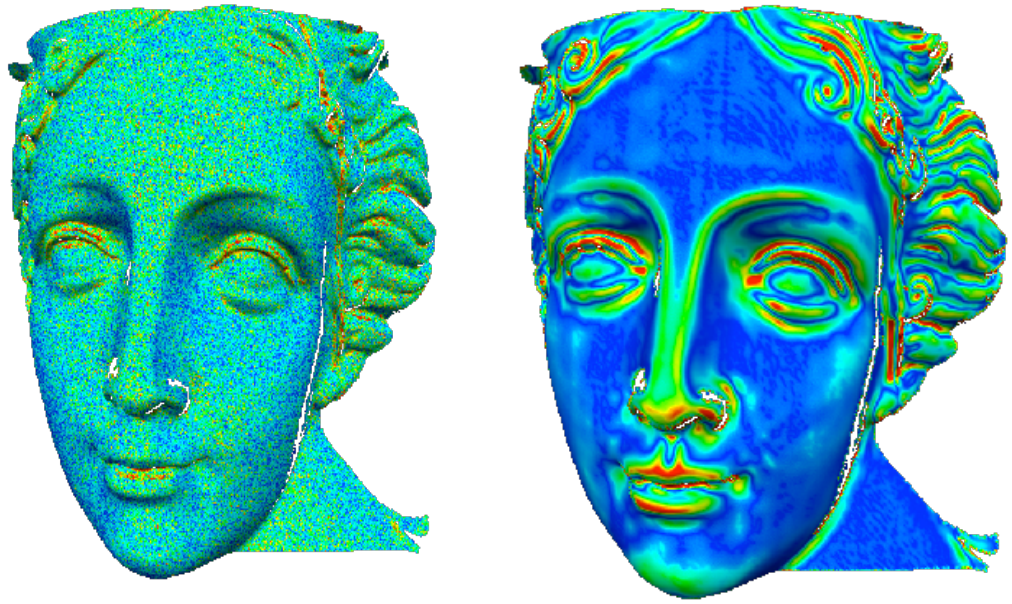
$$\int_{\mathcal{S}} \left(\frac{\partial \kappa_1}{\partial \mathbf{t}_1} \right)^2 + \left(\frac{\partial \kappa_2}{\partial \mathbf{t}_2} \right)^2 dA$$

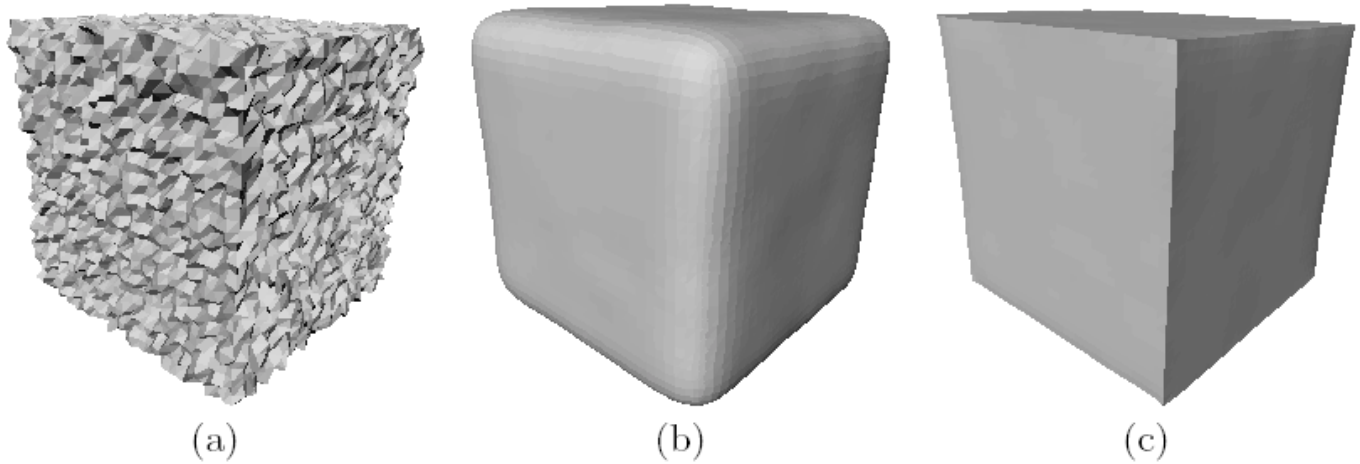
variation of curvature

Mesh Quality Criteria

Smoothness

- Low geometric noise



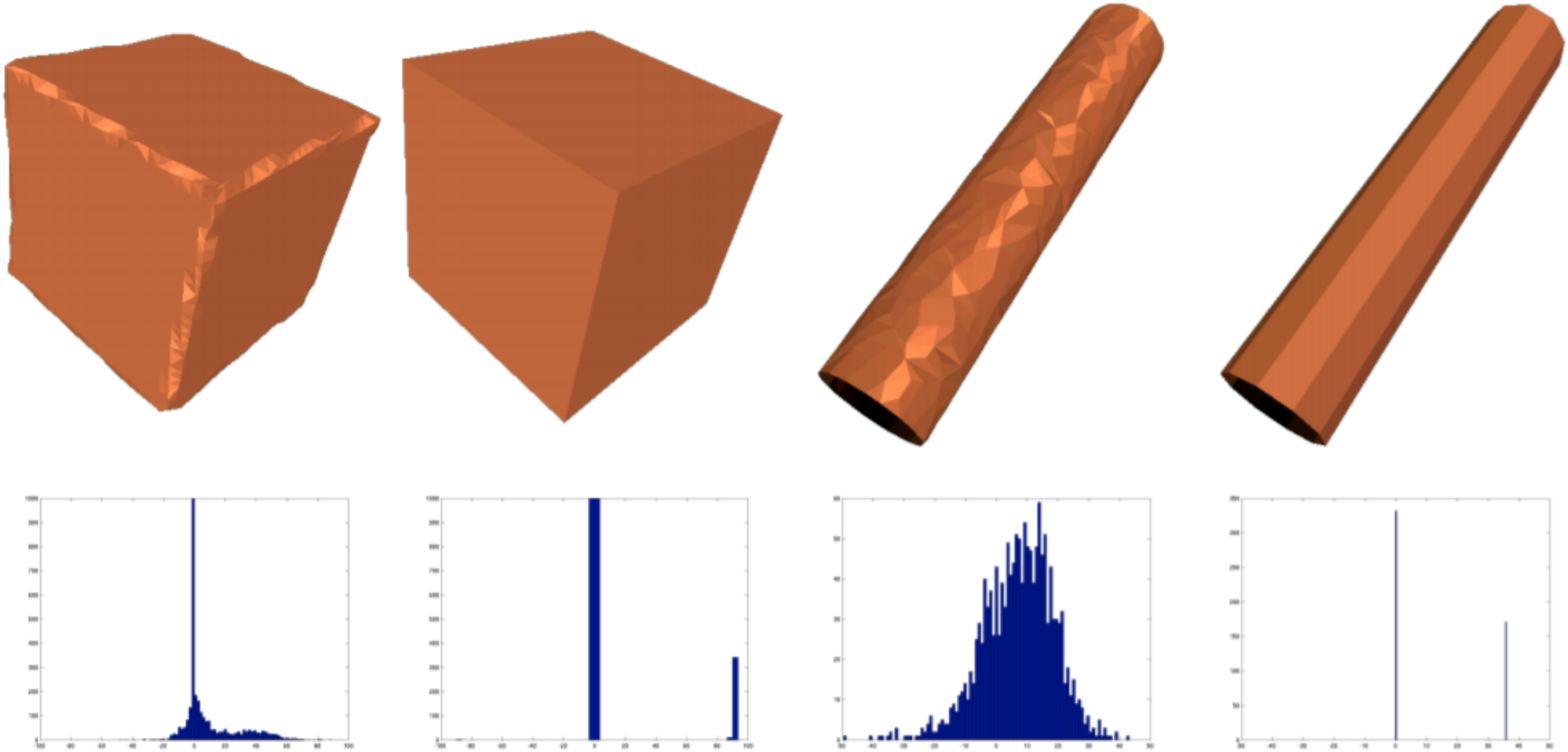


(a) 3% noise added along the normal

(b) Isotropic smoothing

(c) Anisotropic smoothing

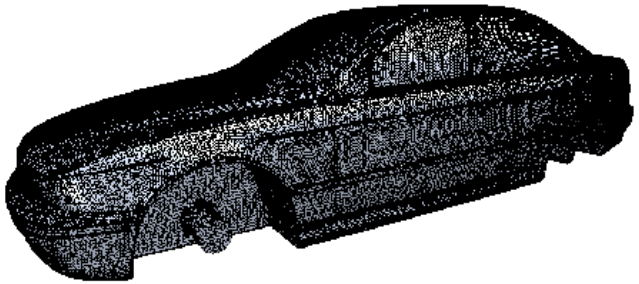
Normal Noise Analysis

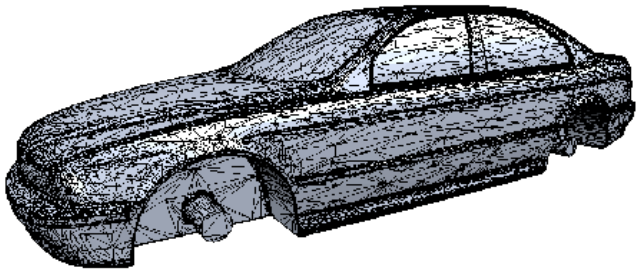


Mesh Quality Criteria

Smoothness

- Low geometric noise



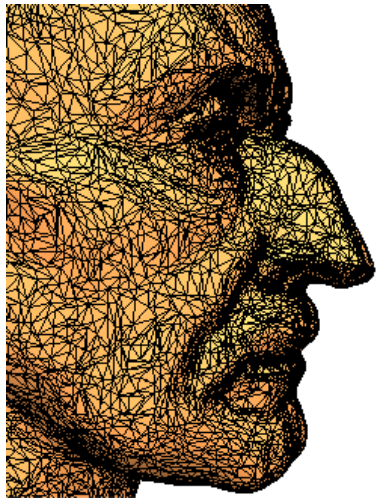


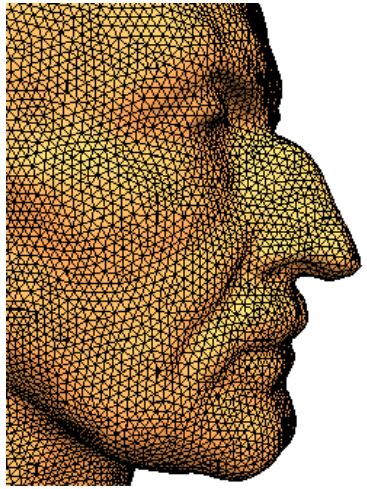
Fairness

- Simplest shape

Adaptive tesselation

- Low complexity



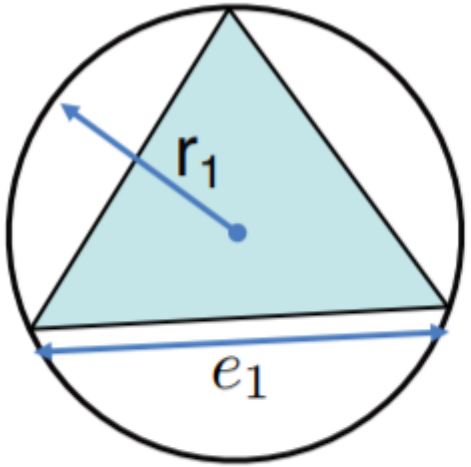


Triangle shape

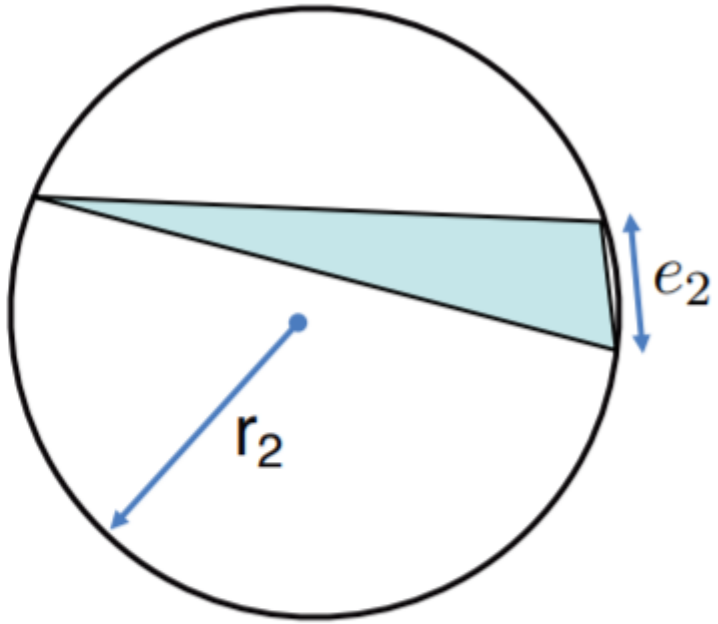
- Numerical Robustness

Triangle Shape Analysis

- Circum radius / shortest edge



$$\frac{r_1}{e_1} < \frac{r_2}{e_2}$$



- Needles and caps

Needle

Cap

Summary

Invariants as overarching theme

- shape does not depend on Euclidean motions (no stretch)
  - **metric** & **curvatures**
- smooth continuous notions to discrete notions
  - generally only as **averages**
- different ways to derive same equations
  - DEC: discrete exterior calculus, FEM, abstract measure theory.

Summary

- A systematic study of convergence conditions for discrete geometry properties is given in [Hildebrandt et al. 06].
- An alternative approach to estimating local surface properties uses a local higher-order reconstruction of the surface, followed by analytic evaluation of the desired properties on the reconstructed surface patch.
- [Wardetzky et al. 07]. They show that the discrete operators cannot simultaneously satisfy all of the identified properties of symmetry, locality, linear precision, and positivity.
  - For example, the cotangent formula of Equation (3.11) satisfies the first three properties, but not the fourth, since edge weights can assume negative values. The choice of discretization thus depends on the specific application.

Literature

- Book: Chapter 3
- Taubin: A signal processing approach to fair surface design, SIGGRAPH 1996
- Desbrun et al. : Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow, SIGGRAPH 1999
- Meyer et al.: Discrete Differential-Geometry Operators for Triangulated 2-Manifolds, VisMath 2002
- Wardetzky et al.: Discrete Laplace Operators: No free lunch, SGP 2007

[1] Gabriel, 2007, **Course**; ***Numerical Mesh Processing***.
[2] 2010_Polygon Mesh Processing

Reading

- **Computing discrete shape operators on general meshes_eg06**

Thanks