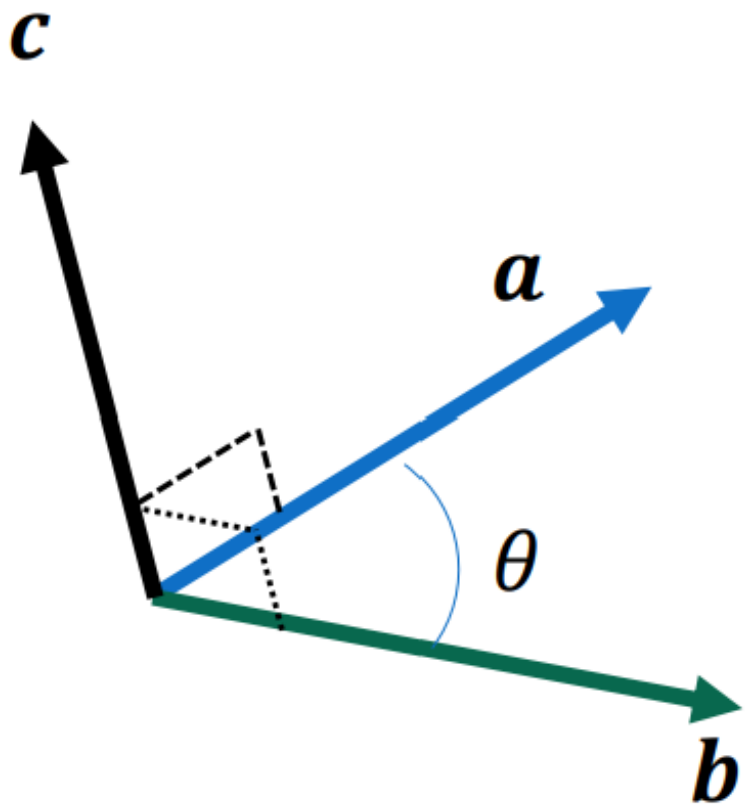
Geometry Processing

handouts

Basic operation: cross product

- $\boldsymbol{c} = \boldsymbol{a} \times \boldsymbol{b} = \begin{bmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{bmatrix}$
- $\boldsymbol{c} \cdot \boldsymbol{a} = \boldsymbol{c} \cdot \boldsymbol{b} = 0$

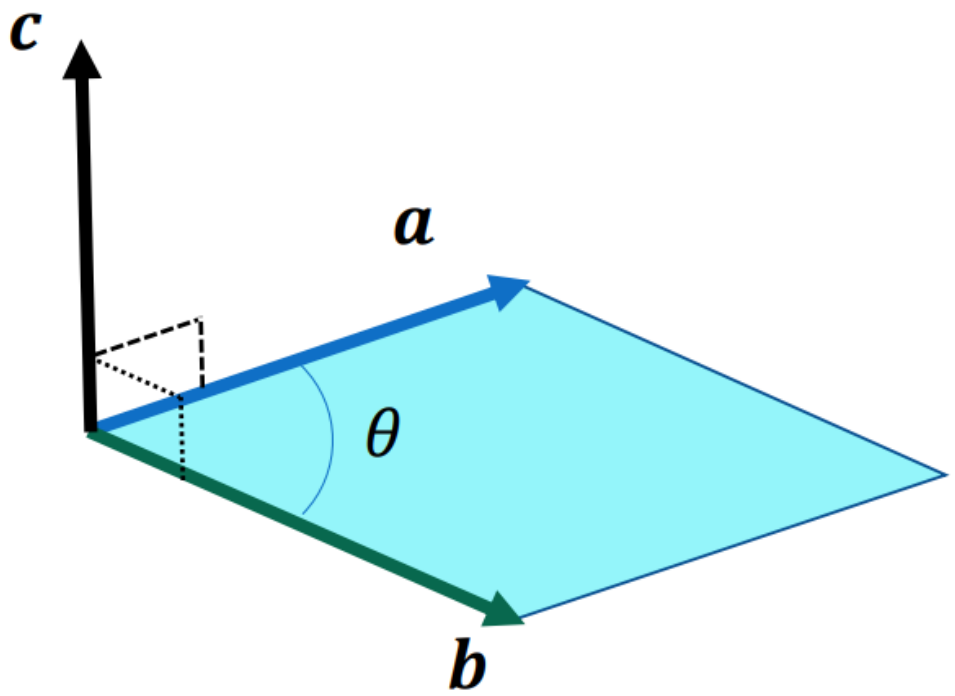
  $\boldsymbol{c}$ is perpendicular to both $\boldsymbol{a}$ and $\boldsymbol{b}$
- $\boldsymbol{a} \times \boldsymbol{b} = -\boldsymbol{b} \times \boldsymbol{a}$
- $\boldsymbol{a} \times (\boldsymbol{b} + \boldsymbol{d}) = \boldsymbol{a} \times \boldsymbol{b} + \boldsymbol{a} \times \boldsymbol{d}$



Basic operation: cross product

- $\boldsymbol{c} = \boldsymbol{a} \times \boldsymbol{b} = \begin{bmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{bmatrix}$
- $||\boldsymbol{c}|| = ||\boldsymbol{a}|| \cdot ||\boldsymbol{b}|| \cdot sin\theta$

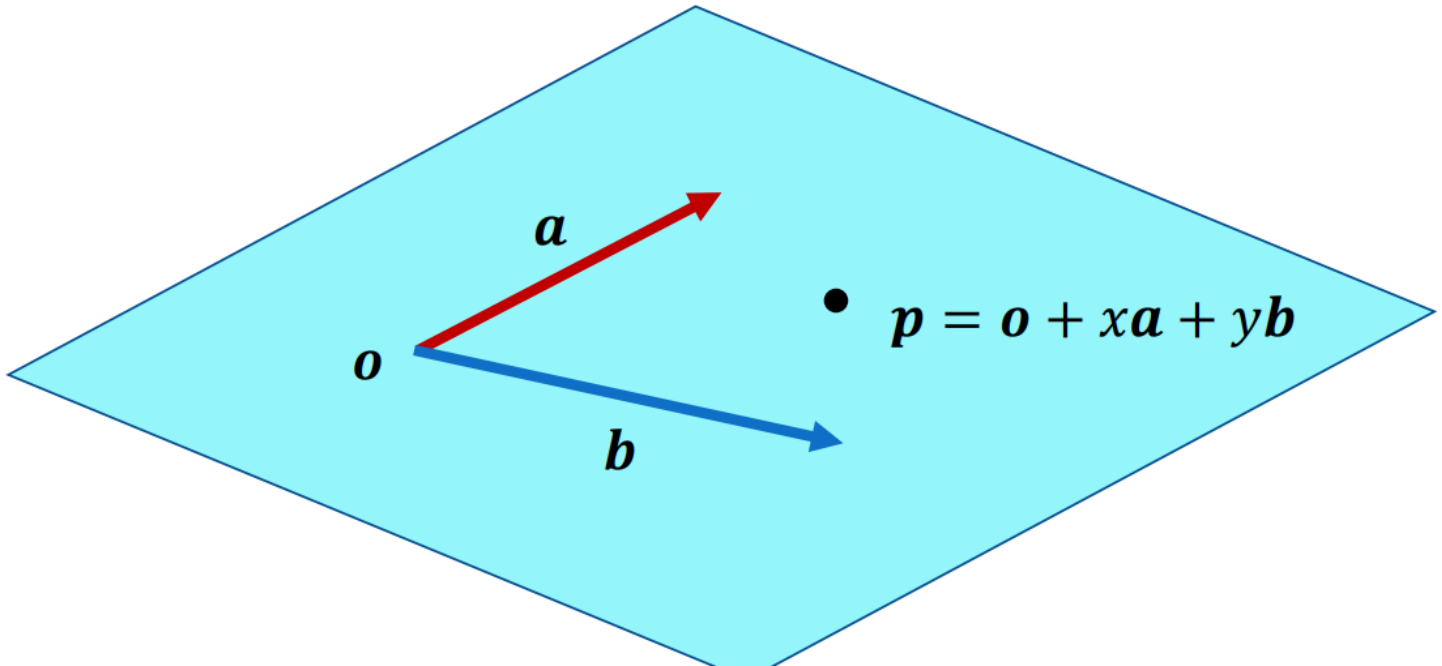
  Area of the parallelogram



Basic operation: plane equation

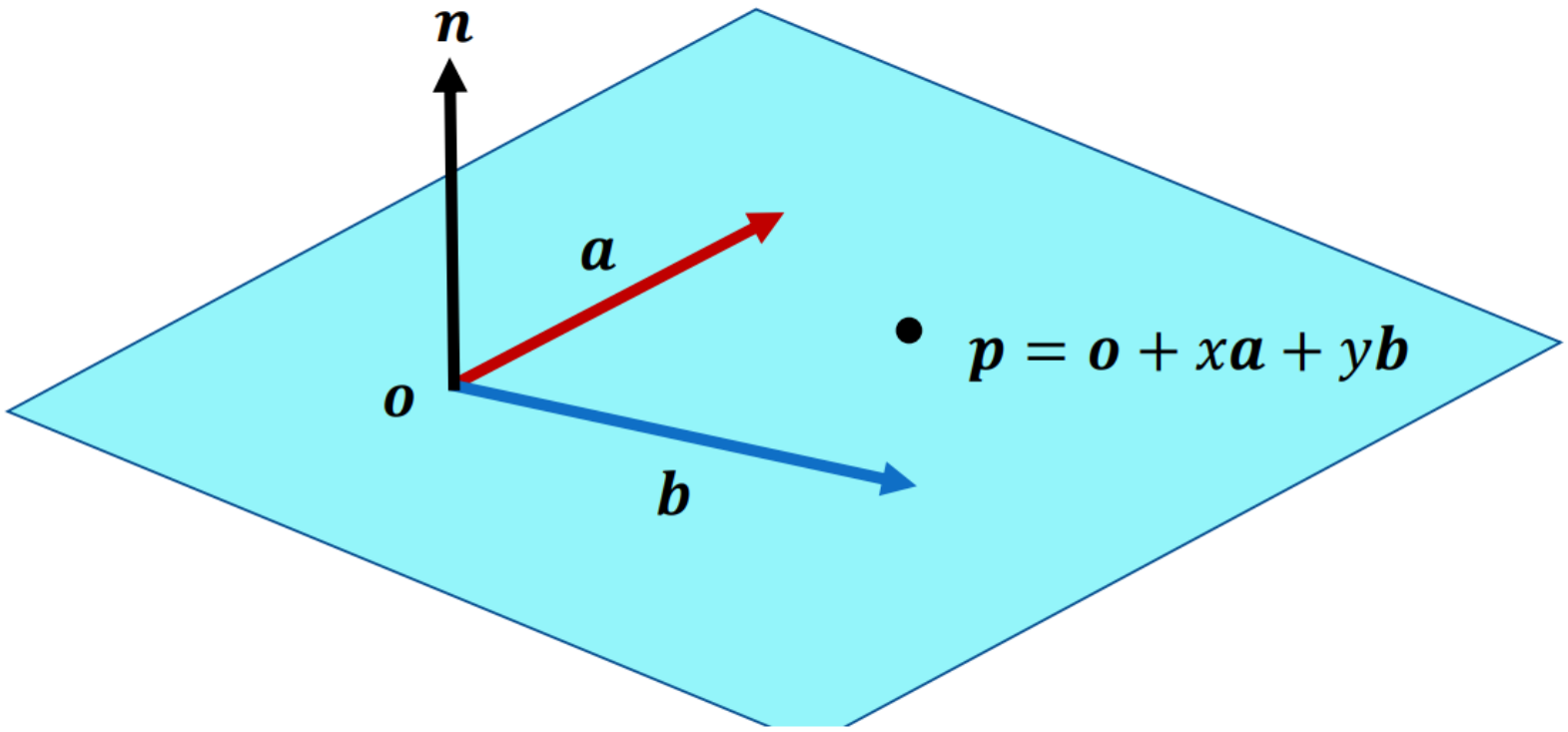
- Given point $\boldsymbol{o}$ and noncollinear vector $\boldsymbol{a}, \boldsymbol{b}$ from the plane
- Any point on the plane can be represent as:

$$\boldsymbol{p} = \boldsymbol{o} + x\boldsymbol{a} + y\boldsymbol{b}, \qquad x, y \in \mathbb{R}$$



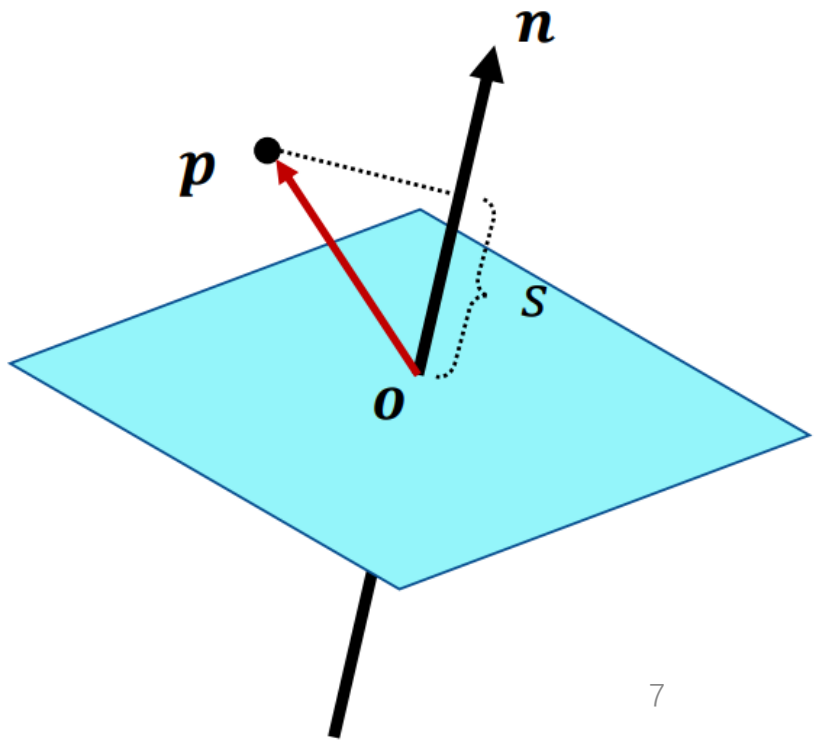
Basic operation: Surface Normal

- Surface normal $\boldsymbol{n} = \boldsymbol{a} \times \boldsymbol{b} / \, ||\boldsymbol{a} \times \boldsymbol{b}||$
- $\boldsymbol{n} \cdot (\boldsymbol{p} - \boldsymbol{o}) = 0, \ \forall \, \boldsymbol{p}$



Basic operation: Surface Normal

- Relation between the plane and a point in space
- $s = \boldsymbol{n} \cdot (\boldsymbol{p} - \boldsymbol{o}) = \begin{cases} s > 0, & \boldsymbol{p} \text{ above the plane} \\ s = 0, & \boldsymbol{p} \text{ on the plane} \\ s < 0, & \boldsymbol{p} \text{ beneath the plane} \end{cases}$



Basic operation: plane equation

- A plane can also be represented as:

$$Ax + By + Cz + D = 0$$

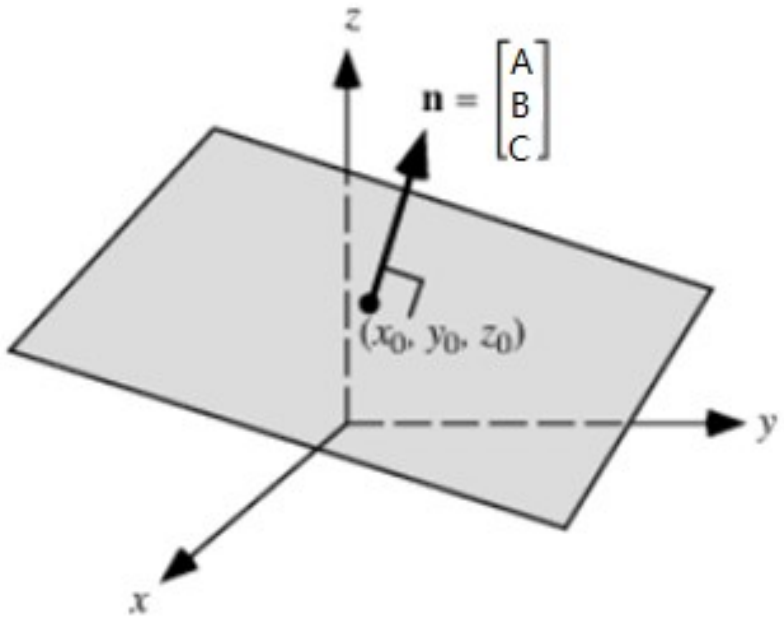
- (x, y, z) is arbitrary point on the plane, A, B, C, D are parameters used to describe features of the plane
- Given 3 noncollinear points from a plane, one can figure out the parameters A, B, C, D

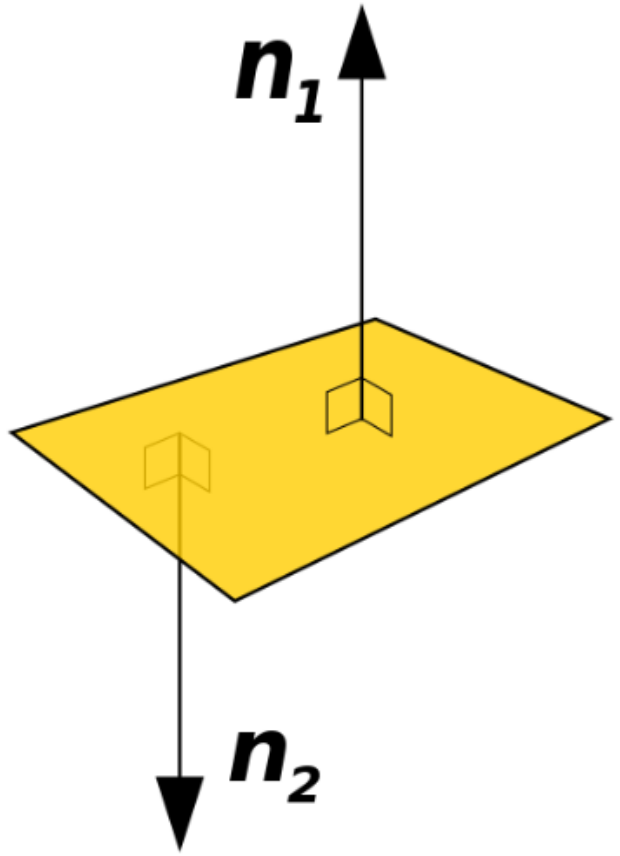
Basic operation: plane equation

- The solution of the equations can be written as:
- $A = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix}$, $B = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix}$
- $C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$, $\mathrm{D} = -\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$

Basic operation: plane equation

- Normal vector is used to define the direction of the plane
- For a plane $Ax + By + Cz + D = 0$
  - (A, B, C) is its normal vector





Basic operation: vertex-plane distance

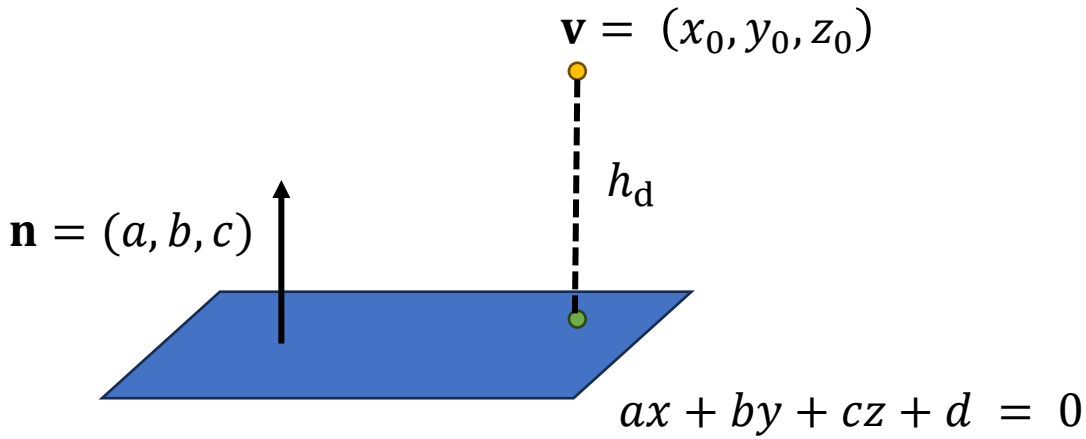
- Vertex： $\mathbf{v} = (x_0, y_0, z_0)$
- Plane： $ax + by + cz + d = 0$

- Vertex-plane distance h_d:

$$h_\mathrm{d}^2 = \tilde{\mathbf{v}}^\top \left(\frac{\tilde{\mathbf{n}}\tilde{\mathbf{n}}^\top}{\mathbf{n}\mathbf{n}^\top} \right) \tilde{\mathbf{v}}$$

where

$$\tilde{\mathbf{v}} = (x_0, y_0, z_0, 1);$$
$$\tilde{\mathbf{n}} = (a, b, c, d);$$
$$\mathbf{n} = (a, b, c)$$

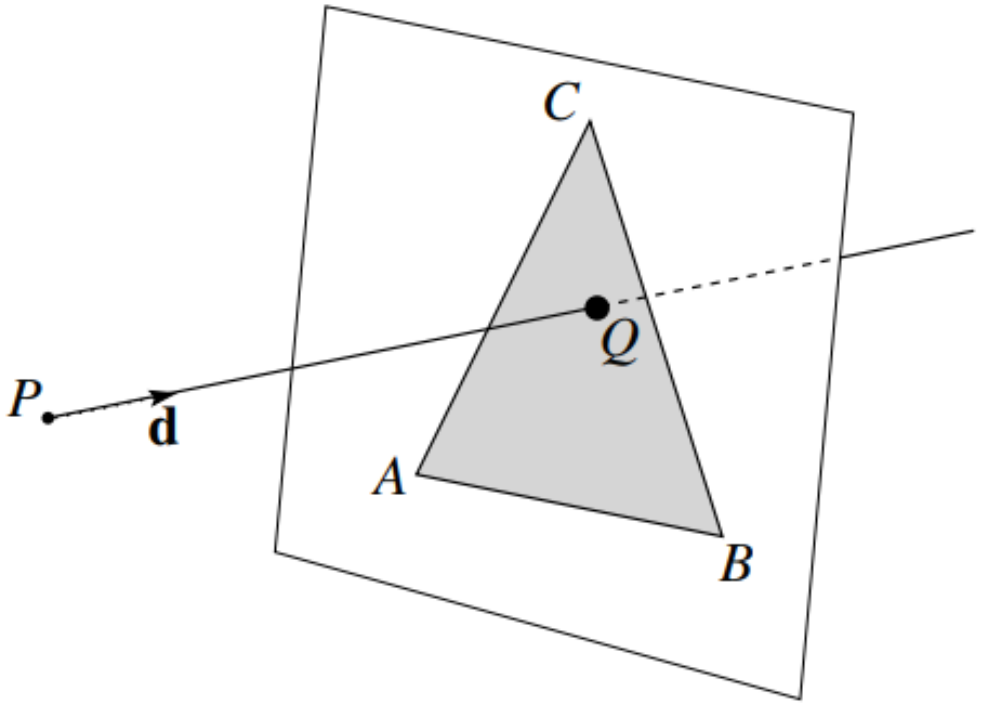


- Tips

$$h_\mathrm{d} = \frac{|ax_0 + by_0 + cz_0 + d|}{\sqrt{a^2 + b^2 + c^2}}$$

Basic operation: Intersection of line and triangle

- Calculate the position Q
  - Get the equation of plane ABC
  - Calculate the intersection of place ABC and line
- Check if Q is inside the triangle



End of handouts

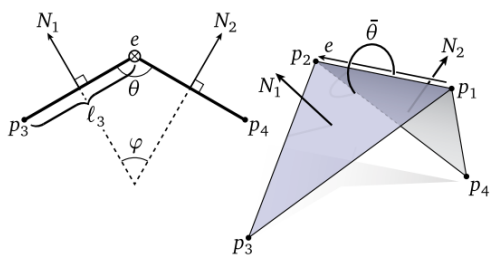
Outline

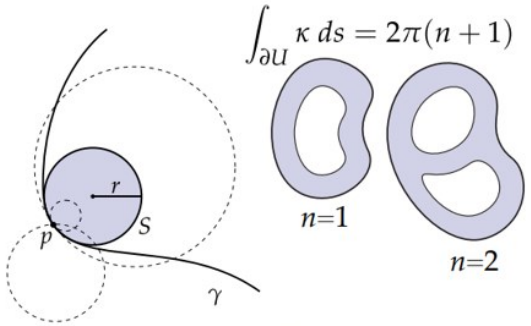
- Discrete differential geometry (this is fundamental to many of the following operations)
  - Local Averaging Region
  - Normal Vectors
  - Gradients
  - Laplace-Beltrami Operator
- Geometry Processing (we won’t introduce all of the these in class)
  - Mesh Parameterization (introduced earlier)
  - Mesh Simplification
  - Mesh Smoothing
  - Mesh Morphing
  - Mesh Stylization
  - Mesh Editing …

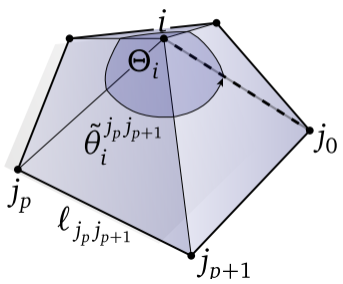
Discrete differential geometry

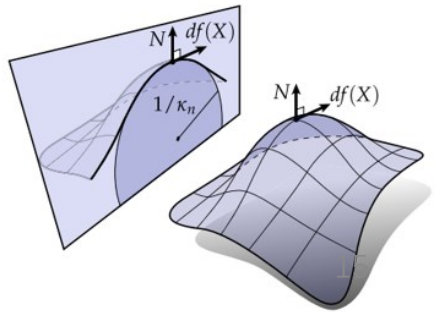
- Language for talking about local properties of shape
- Compute approximations of the differential properties of this underlying surface directly from the mesh data.
- Applications:
  - mesh filtering, parameterization, pose transfer, segmentation, reconstruction, re-meshing, compression, simulation, and interpolation via barycentric coordinates.

- Local Averaging Region
- Normal Vectors
- Gradients
- Laplace-Beltrami Operator





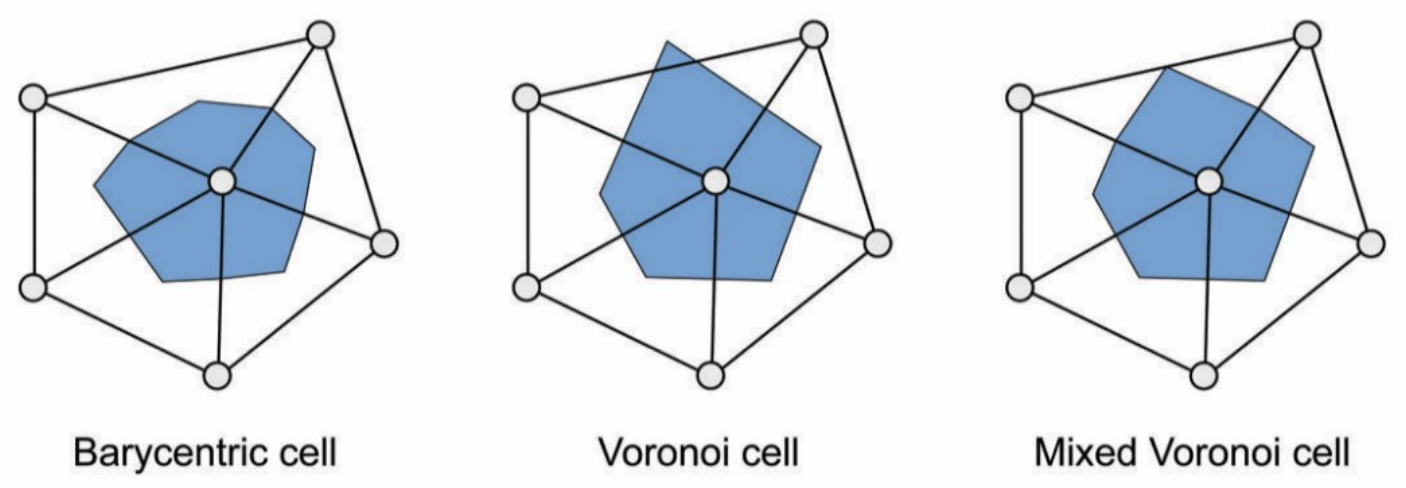




A nice tutorial: https://brickisland.net/DDGSpring2022/

Local Averaging Region

- General idea: spatial averages over a local neighborhood $\Omega(\boldsymbol{x})$ of a point $\boldsymbol{x}$.
- $\boldsymbol{x}$: one mesh vertex
- $\Omega(\boldsymbol{x})$: n-ring neighborhoods of mesh vertex or local geodesic balls.
- The size of the $\Omega(\boldsymbol{x})$: stability and accuracy
  - Large size: smooth
  - Small size: accurate for clean mesh data



Normal Vectors

- Normal vectors for individual triangles are well-defined.
- Vertex normal: spatial averages of normal vectors in a local one-ring neighborhood.

$$\mathbf{n}(v) = \frac{\sum_{T \in \Omega(v)} \alpha_T \mathbf{n}(T)}{\left\| \sum_{T \in \Omega(v)} \alpha_T \mathbf{n}(T) \right\|}$$

- 1. constant weights: $\alpha_T = 1$
- 2. triangle area: $\alpha_T = \text{area}(T)$
- 3. incident triangle angles: $\alpha_T = \theta(T)$

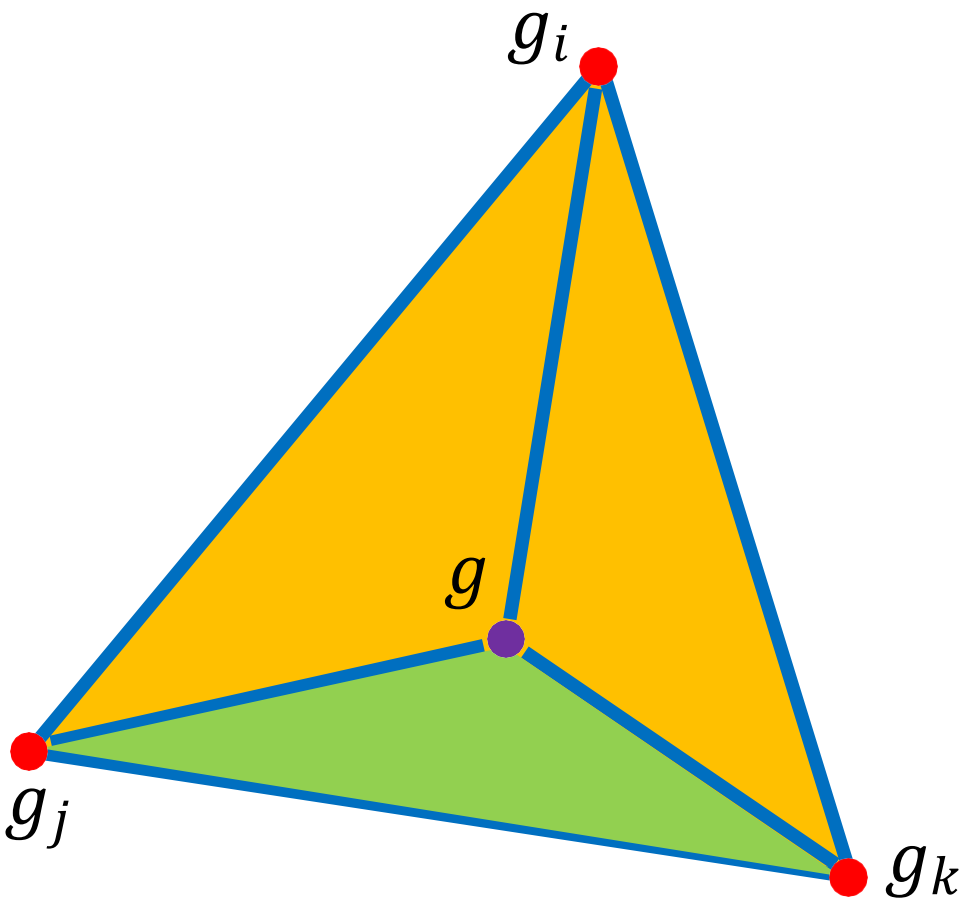
Gradients

First, let's introduce barycentric coordinates:

$$g = \alpha g_i + \beta g_j + \gamma g_k$$
$$\alpha + \beta + \gamma = 1,$$
$$\alpha, \beta, \gamma \geq 0$$

$$\alpha = \frac{s_i}{s_i + s_j + s_k}$$

s_i: area of the green triangle



Gradients

- Given the function value on vertices, compute the gradient on each triangle.
- A piecewise linear function

$$f(\mathrm{x}) = \alpha f_i + \beta f_j + \gamma f_k$$

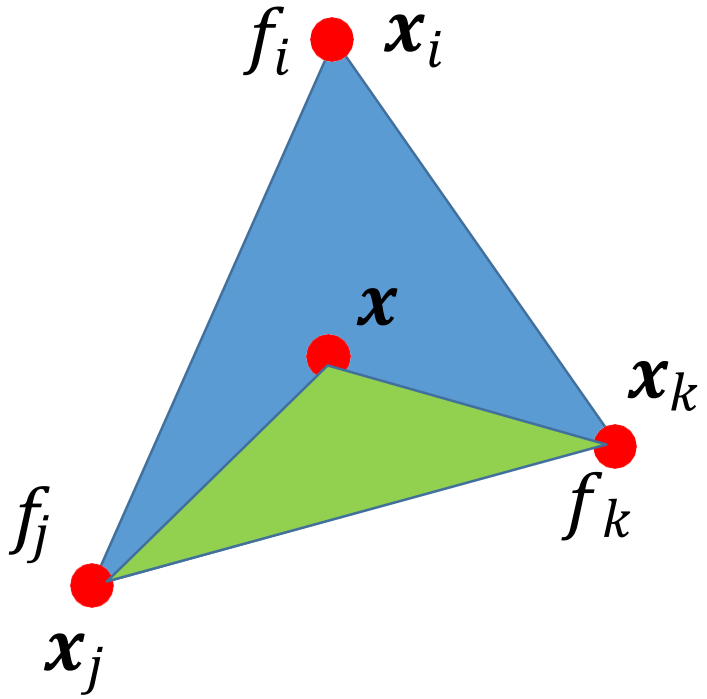
- Gradient

$$\nabla_{\boldsymbol{x}} f(\mathrm{x}) = f_i \nabla_{\boldsymbol{x}} \alpha + f_j \nabla_{\boldsymbol{x}} \beta + f_k \nabla_{\boldsymbol{x}} \gamma$$

- Because

$$\alpha = \frac{A_i}{A_T} = \frac{\left((\boldsymbol{x} - \boldsymbol{x}_j) \cdot \frac{(\boldsymbol{x}_k - \boldsymbol{x}_j)^{\perp}}{\|\boldsymbol{x}_k - \boldsymbol{x}_j\|_2} \right) \|\boldsymbol{x}_k - \boldsymbol{x}_j\|_2}{2A_T}$$

$$= (\boldsymbol{x} - \boldsymbol{x}_j) \cdot (\boldsymbol{x}_k - \boldsymbol{x}_j)^{\perp} / 2A_T$$



$\perp$: Rotate 90°

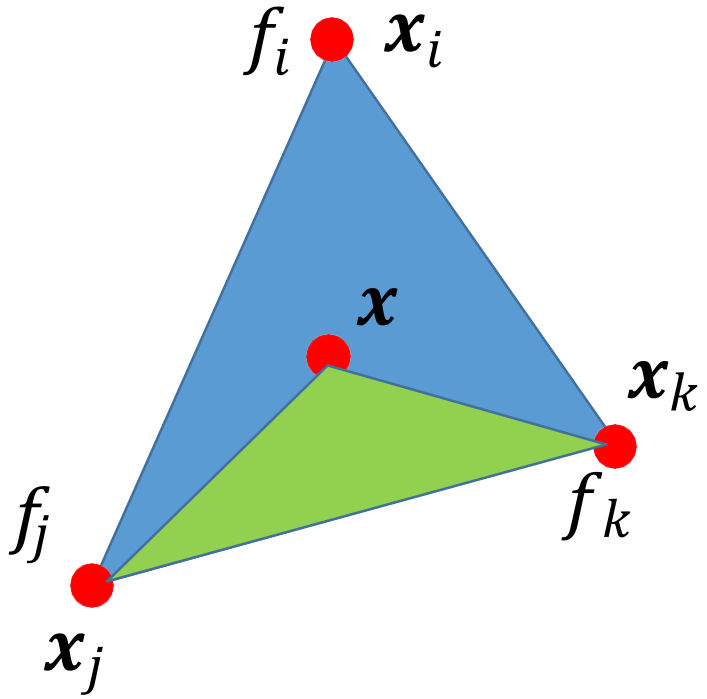
Gradients

- Then

$$\nabla_{\boldsymbol{x}}\alpha = \frac{(\boldsymbol{x}_k - \boldsymbol{x}_j)^{\perp}}{2A_T}$$

$$\nabla_{\boldsymbol{x}}\beta = \frac{(\boldsymbol{x}_i - \boldsymbol{x}_k)^{\perp}}{2A_T}$$

$$\nabla_{\boldsymbol{x}}\gamma = \frac{(\boldsymbol{x}_j - \boldsymbol{x}_i)^{\perp}}{2A_T}$$

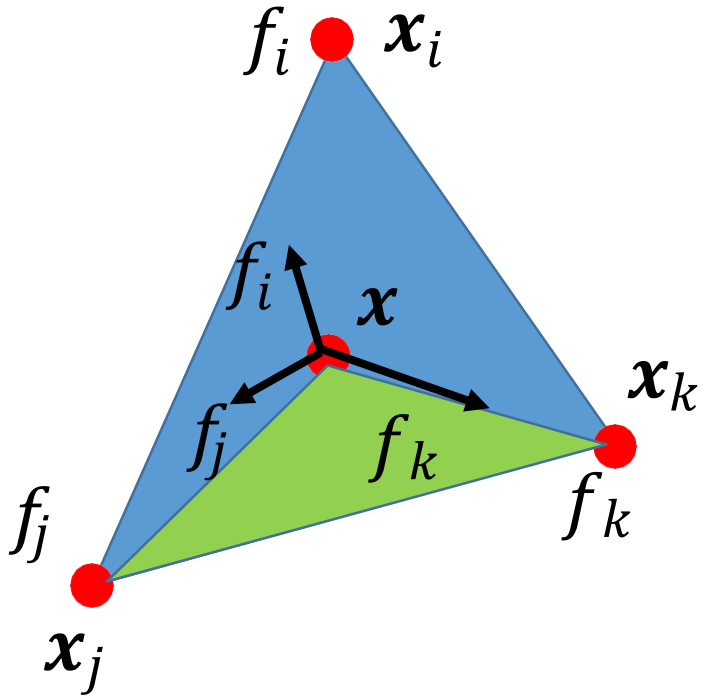


- Gradient:

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = f_i \frac{(\boldsymbol{x}_k - \boldsymbol{x}_j)^{\perp}}{2A_T} + f_j \frac{(\boldsymbol{x}_i - \boldsymbol{x}_k)^{\perp}}{2A_T} + f_k \frac{(\boldsymbol{x}_j - \boldsymbol{x}_i)^{\perp}}{2A_T}$$

Gradients

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = f_i \frac{(\boldsymbol{x}_k - \boldsymbol{x}_j)^{\perp}}{2A_T} + f_j \frac{(\boldsymbol{x}_i - \boldsymbol{x}_k)^{\perp}}{2A_T} + f_k \frac{(\boldsymbol{x}_j - \boldsymbol{x}_i)^{\perp}}{2A_T}$$



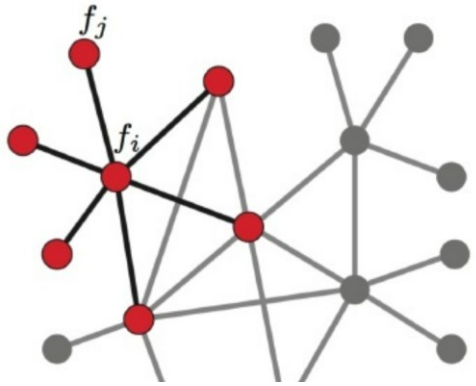
- Constant on each facet
- Different in different facets

Laplace-Beltrami Operator

- Constant gradient on facet → zero Laplace value on facet

- Gradient on the vertex： a simple version of discrete Laplace operator on vertex-based functions:

$$(\mathrm{L}f)_i = \sum_{j \in \Omega(i)} \omega_{ij} \, (f_j - f_i)$$



Uniform Laplacian

- $\omega_{ij} = 1$ or $\omega_{ij} = \frac{1}{N_i}$

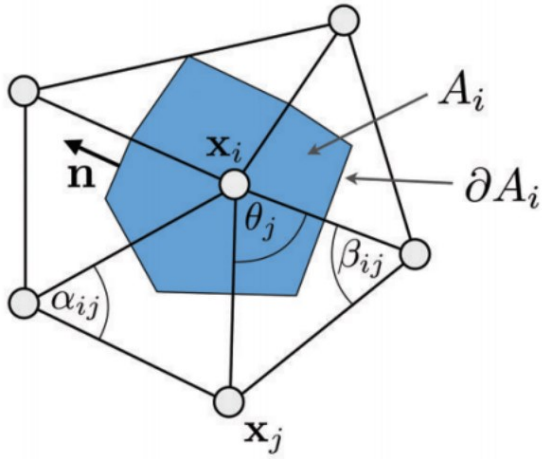
$$(\mathrm{L}f)_i = \sum_{j \in \Omega(i)} (f_j - f_i) \quad \text{or} \quad (\mathrm{L}f)_i = \frac{1}{N_i} \sum_{j \in \Omega(i)} (f_j - f_i) = \frac{\sum_{j \in \Omega} f_j}{N_i} - f_i$$

- The definition only depends on the connectivity of the mesh.
- The uniform Laplacian does not adapt at all to the spatial distribution of vertices.

Cotangent Formula

Discrete average of the Laplace-Beltrami operator of a function f at vertex v_i is given as

$$\Delta f(v_i) = \frac{1}{2A_i} \sum_{j \in \Omega(i)} (\cot\alpha_{ij} + \cot\beta_{ij})(f_j - f_i)$$



Most widely used discretization

Detailed proof in the following slides, read if interest.

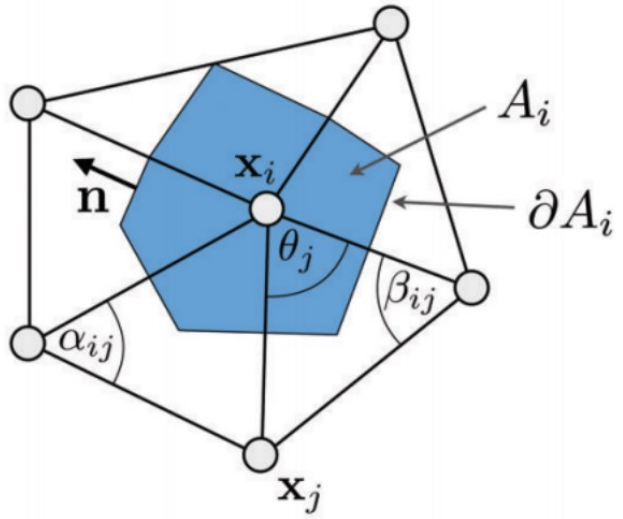
Cotangent Formula

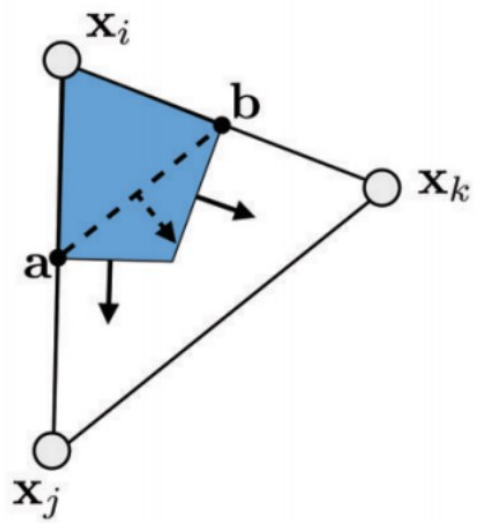
- Assume it constant on each vertex

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

根据格林公式，面积分转化为线积分

- A_i is the local averaging domain of vertex i
- ∂A_i is the boundary of A_i
- n is the outward pointing unit normal of the boundary
- f is the signal defined on the mesh

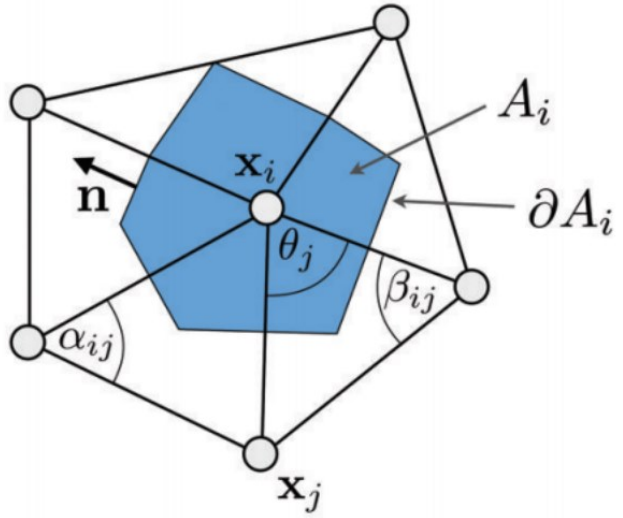




Cotangent Formula

We split this integral by considering the integration separately for each triangle T.

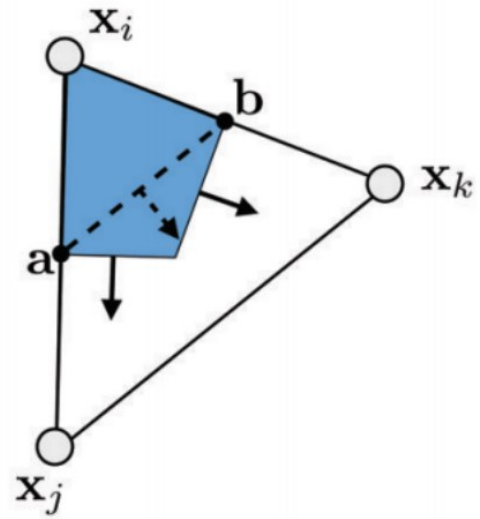
$$\int_{\partial A_i \cap T} \nabla f \cdot \boldsymbol{n} \boldsymbol{d} \boldsymbol{s} = \nabla f \cdot (\boldsymbol{a} - \boldsymbol{b})^{\perp} = \frac{1}{2} \nabla f \cdot (\boldsymbol{x}_j - \boldsymbol{x}_k)^{\perp}$$



∇f is constant within each triangle.

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

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



Cotangent Formula

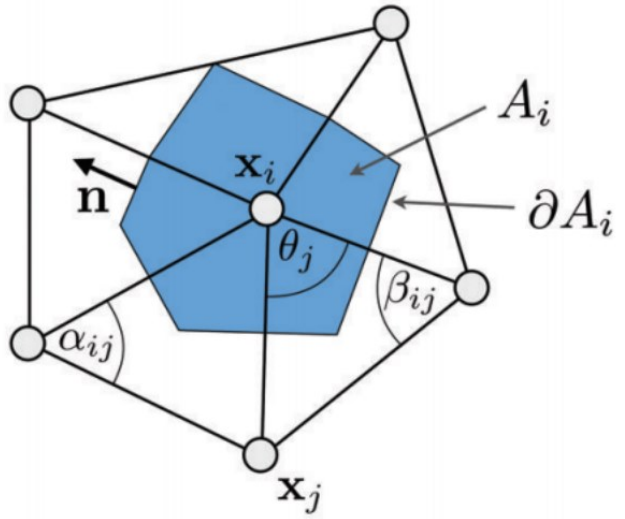
- Because:

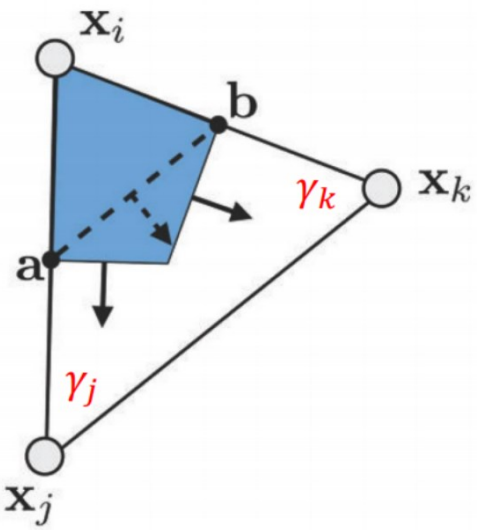
$$A_T = \frac{1}{2}\sin\gamma_j \|\boldsymbol{x}_j - \boldsymbol{x}_i\| \, \|\boldsymbol{x}_j - \boldsymbol{x}_k\|$$
$$= \frac{1}{2}\sin\gamma_k \|\boldsymbol{x}_i - \boldsymbol{x}_k\| \, \|\boldsymbol{x}_j - \boldsymbol{x}_k\|$$

and

$$\cos\gamma_j = \frac{(\boldsymbol{x}_j - \boldsymbol{x}_i)\cdot(\boldsymbol{x}_j - \boldsymbol{x}_k)}{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|\|\boldsymbol{x}_j - \boldsymbol{x}_k\|}$$

$$\cos\gamma_k = \frac{(\boldsymbol{x}_i - \boldsymbol{x}_k)\cdot(\boldsymbol{x}_j - \boldsymbol{x}_k)}{\|\boldsymbol{x}_i - \boldsymbol{x}_k\|\|\boldsymbol{x}_j - \boldsymbol{x}_k\|}$$

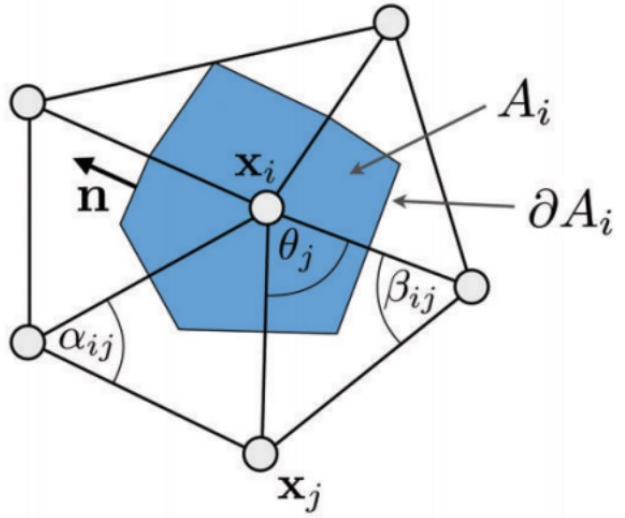




Cotangent Formula

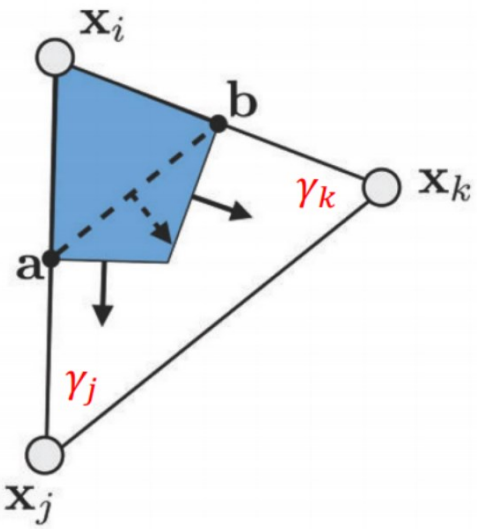
- and

$$(\boldsymbol{x}_i - \boldsymbol{x}_k)^{\perp} \cdot (\boldsymbol{x}_j - \boldsymbol{x}_k)^{\perp} = (\boldsymbol{x}_i - \boldsymbol{x}_k) \cdot (\boldsymbol{x}_j - \boldsymbol{x}_k)$$
$$(\boldsymbol{x}_j - \boldsymbol{x}_i)^{\perp} \cdot (\boldsymbol{x}_j - \boldsymbol{x}_k)^{\perp} = (\boldsymbol{x}_j - \boldsymbol{x}_i) \cdot (\boldsymbol{x}_j - \boldsymbol{x}_k)$$



So

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



Cotangent Formula

$$\int_{A_i} \Delta f \, dA = \frac{1}{2} \sum_{j \in \Omega(i)} (\cot\alpha_{ij} + \cot\beta_{ij})(f_j - f_i)$$

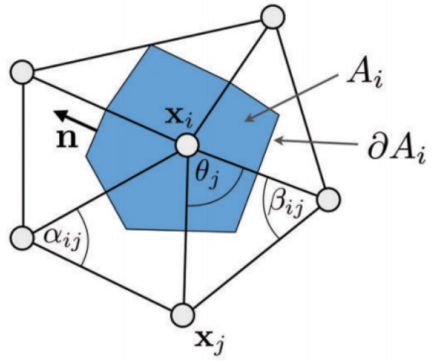
Discrete average of the Laplace-Beltrami operator of a function f at vertex v_i is given as

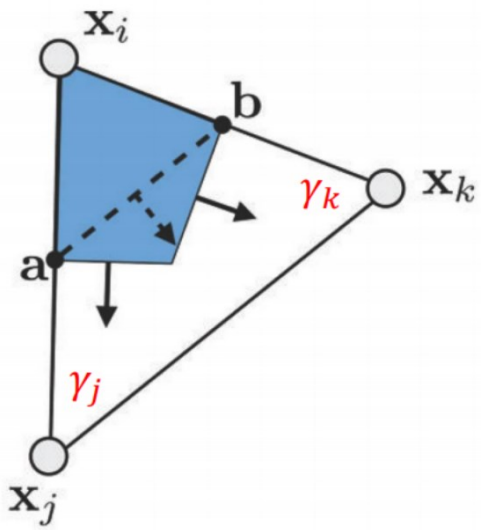
Discrete average of the Laplace-Beltrami operator of a function f at vertex v_i is given as

$$\Delta f(v_i) = \frac{1}{2A_i} \sum_{j \in \Omega(i)} (\cot\alpha_{ij} + \cot\beta_{ij})(f_j - f_i)$$

Most widely used discretization

Most widely used discretization



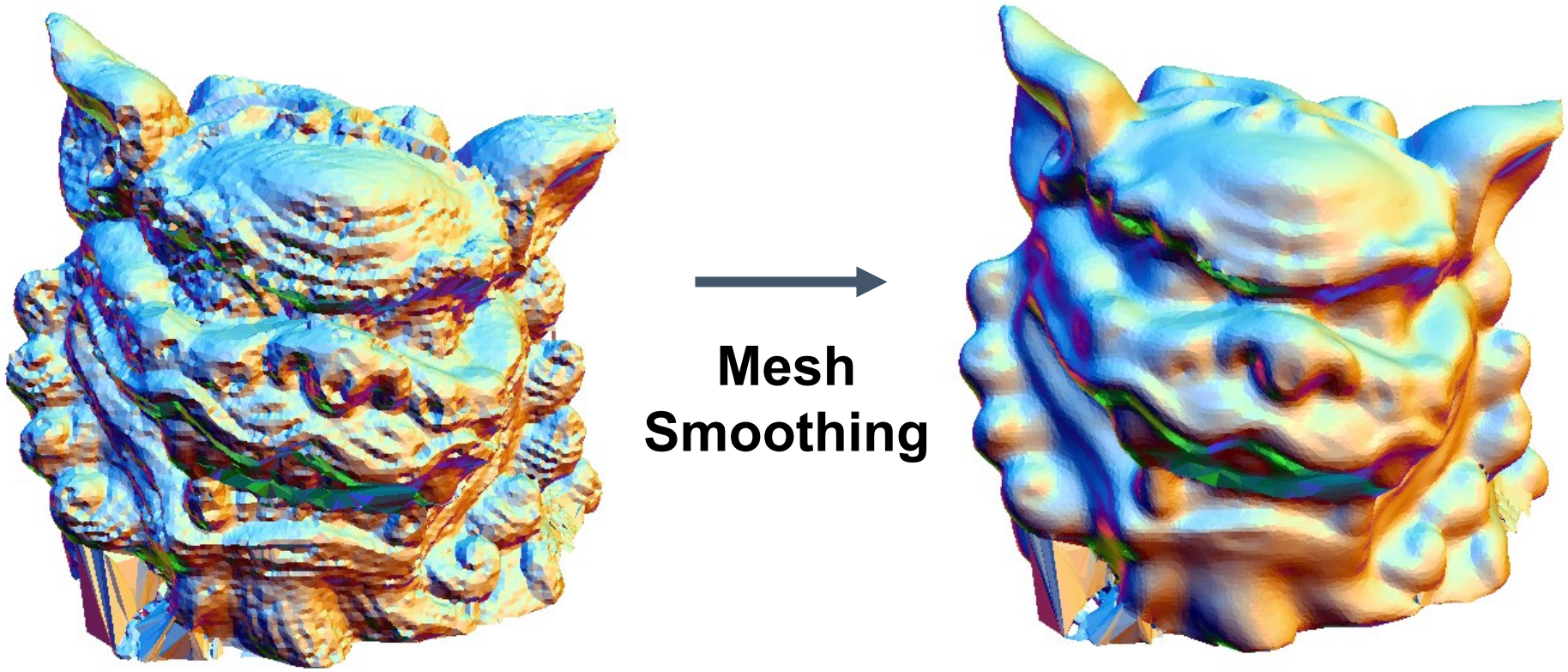


Gradient and Laplacian

Representation	Gradient	Laplacian
Triangle Mesh	$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = f_i \frac{(\boldsymbol{x}_k - \boldsymbol{x}_j)^{\perp}}{2A_T} + f_j \frac{(\boldsymbol{x}_i - \boldsymbol{x}_k)^{\perp}}{2A_T} + f_k \frac{(\boldsymbol{x}_j - \boldsymbol{x}_i)^{\perp}}{2A_T}$ 描述网格顶点间的标量场变化方向	$\Delta f(v_i) = \frac{1}{2A_i} \sum_{j \in \Omega(i)} (\cot\alpha_{ij} + \cot\beta_{ij})(f_j - f_i)$ $\Delta f(v_i) = \frac{\sum_{j \in \Omega} f_j}{N_i} - f_i$ 曲面上标量场的curvature
Cubic Bézier Curve	$\frac{dB(0)}{dt} = 3(P_1 - P_0)$ 曲线在起点的切线方向和变化速率	$\Delta B(0) = 6(P_0 - 2P_1 + P_2)$ 曲线在起点附近的curvature
2D Image	$\nabla_x f = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * f$ 检测图像边缘，表示亮度变化的方向和幅度	$\Delta f = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} * f$ 检测高频信息或增强图像中的细节

Mesh Smoothing

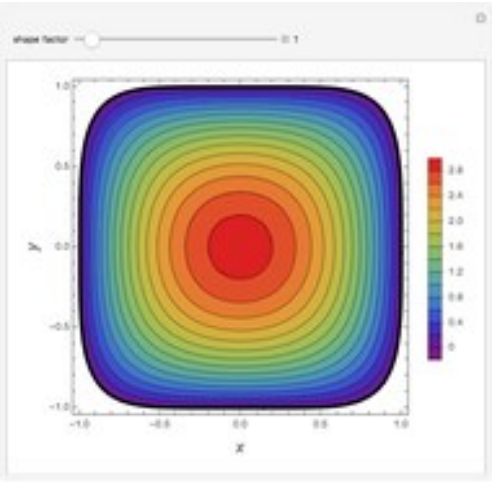
- Meshes obtained from real world are usually noisy…



Laplacian smoothing

- Diffusion flow: a mathematically well-understood model for the time-dependent process of smoothing a given signal $f(\boldsymbol{x},t)$.
  - Heat diffusion, Brownian motion
- Diffusion equation:

$$\frac{\partial f(\boldsymbol{x},t)}{\partial t} = \lambda \Delta f(\boldsymbol{x},t)$$



- Smooth an arbitrary function f on a manifold surface by using Laplace-Beltrami Operator

Spatial discretization

- Sample values at the mesh vertices $\boldsymbol{f}(t) = (f(v_1, t), \ldots, f(v_n, t))^T$

- The evolution of function value of each vertex:

$$\frac{\partial f(v_i, t)}{\partial t} = \lambda \Delta f(\boldsymbol{v}_i, t)$$

- Matrix form:

$$\frac{\partial \boldsymbol{f}(t)}{\partial t} = \lambda \cdot L\boldsymbol{f}(t)$$

Spatial & time discretization

- Uniform sampling: (t, t+h, t+2h……)
- Explicit Euler integration:

$$\boldsymbol{f}(t+h) = \boldsymbol{f}(t) + h\frac{\partial \boldsymbol{f}(t)}{\partial t} = \boldsymbol{f}(t) + h\lambda \cdot L\boldsymbol{f}(t)$$

- Arbitrary function -> vertex positions $\boldsymbol{f} \implies (\boldsymbol{x_1}, \ldots, \boldsymbol{x_n})^{\boldsymbol{T}}$

- Laplacian smoothing

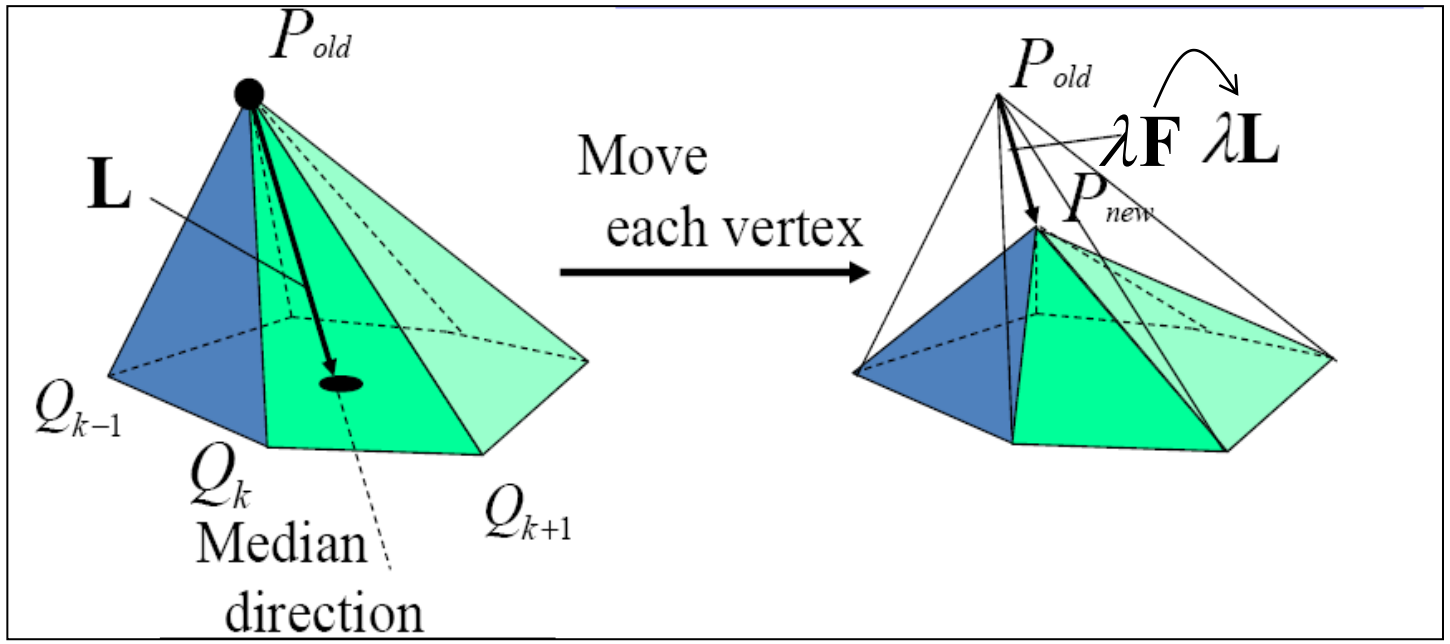
$$\boldsymbol{x}_i \longleftarrow \boldsymbol{x}_i + h\lambda \cdot \Delta \boldsymbol{x}_i$$

- Discrete Laplace-Beltrami using either the uniform or cotangent formula

Laplacian smoothing

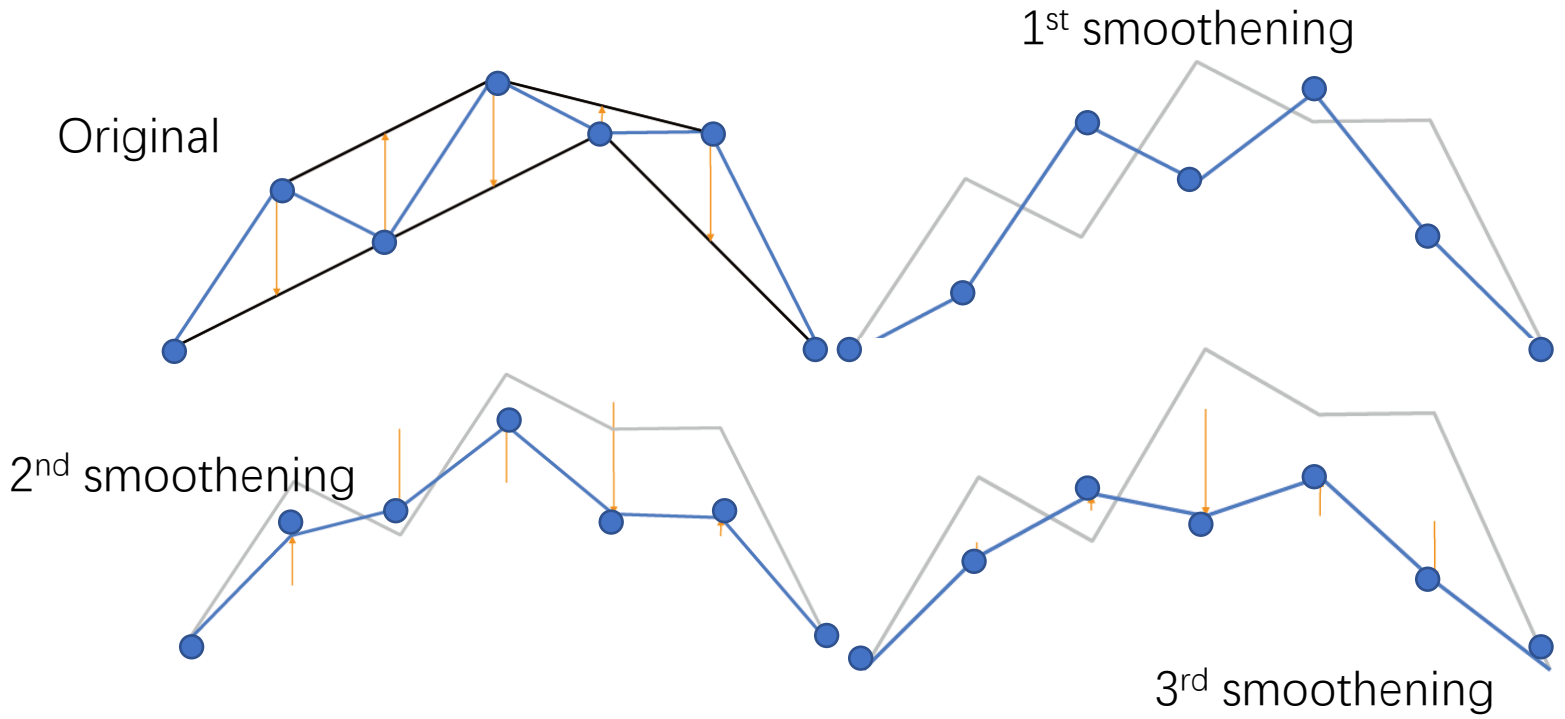
直接拉普拉斯平滑算法的原理是将每个顶点都移动到相邻顶点的平均位置，即采用所谓伞状算子。这里将拉普拉斯算子L离散化为顶点到邻域平均位置的平移

- Laplacian smoothing $\boldsymbol{x}_i \longleftarrow \boldsymbol{x}_i + h\lambda \cdot \Delta \boldsymbol{x}_i$



Laplacian smoothing

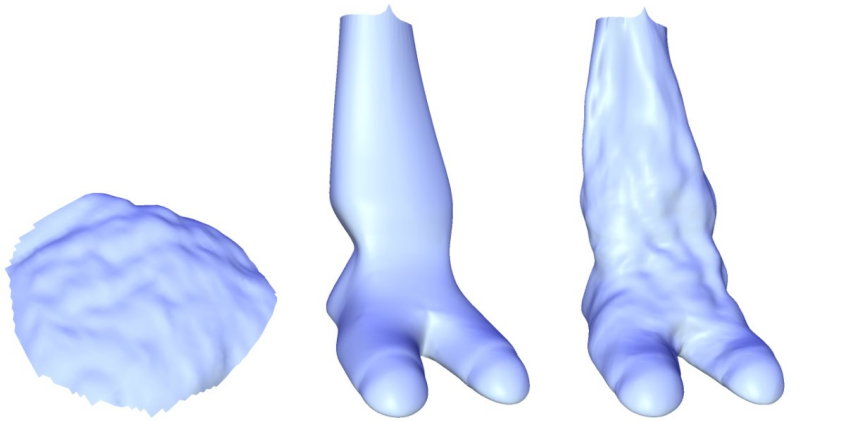
- Laplacian smoothing $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + h\lambda \cdot \Delta \boldsymbol{x}_i$



灰色是上一次的形状，蓝色是初始形状。理想情形下，蓝色的点要趋近于前后相邻两个顶点的中心位置。黄色箭头表示趋势。

Detail-Preserving Mesh Editing

- The spatial constraints will serve as modeling constraints
- Reconstruct the surface every time the modeling constraints are changed



Review: Poisson Image Editing

- How to “inject” some shapes or information to the region?
- Altering the optimization goal:

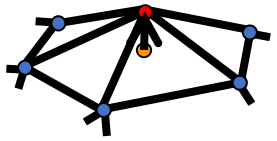
$$\underset{f}{\operatorname{argmin}} \iint_{\Omega} \|\nabla f - \mathbf{g}\|^2, \quad \text{s.t. } f|_{\partial\Omega} = f^*|_{\partial\Omega}\,.$$

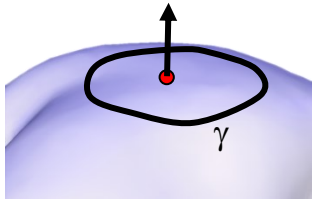
- g is the function that “guides” the completion and is usually the gradient of another picture.
- Laplace editing (Image completion) is the special case of Poisson editing where g=0 and the region has only one pure color.

Review: Poisson Image Editing

Detail-Preserving Mesh Editing

- The spatial constraints will serve as modeling constraints
- Reconstruct the surface every time the modeling constraints are changed





- Laplacian mesh editing
  - Calculate Laplacian $\Delta = L(V)$
  - Add model constrain $v_i^{'} = u_i,\ i\epsilon\ C$
  - Least squares method

$$\tilde{V}' = \arg\min_{V'} \left(\left\| L(V') - \Delta \right\|^2 + \sum_{i \in C} \left\| \mathbf{v}_i' - \mathbf{u}_i \right\|^2 \right)$$

Laplacian

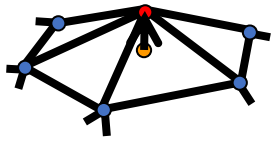
Constrain vertices

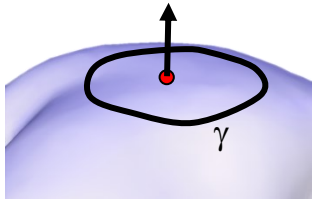
数学上，通过离散的Laplacian微分坐标表示不改变点的法向和曲率，因此，能够反映表面的细节。

在交互指定一些约束顶点的目标位置后，通过重建一个满足该约束顶点、同时保持局部拉普拉斯微分坐标的网格，便可以生成编辑后的网格。其本质上是求解最小二乘优化问题。

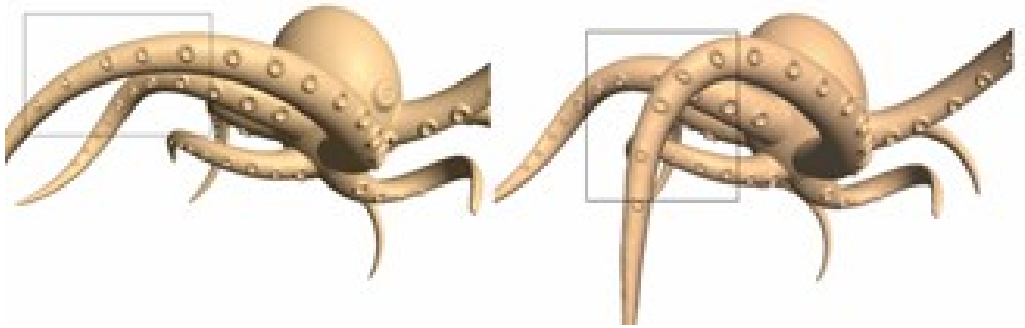
Detail-Preserving Mesh Editing

- The spatial constraints will serve as modeling constraints
- Reconstruct the surface every time the modeling constraints are changed





- Laplacian mesh editing
  - Calculate Laplacian $\Delta = L(V)$
  - Add model constrain $v_i^{'} = u_i,\ \text{i}\epsilon\ C$
  - Least squares method



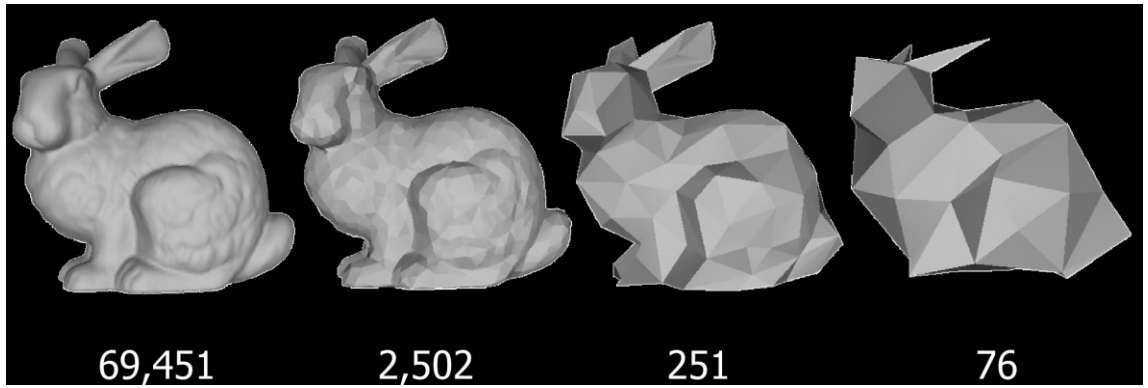
$$\tilde{V}' = \arg\min_{V'} \left(\left\| L(V') - \Delta \right\|^2 + \sum_{i \in C} \left\| \mathbf{v}_i' - \mathbf{u}_i \right\|^2 \right)$$

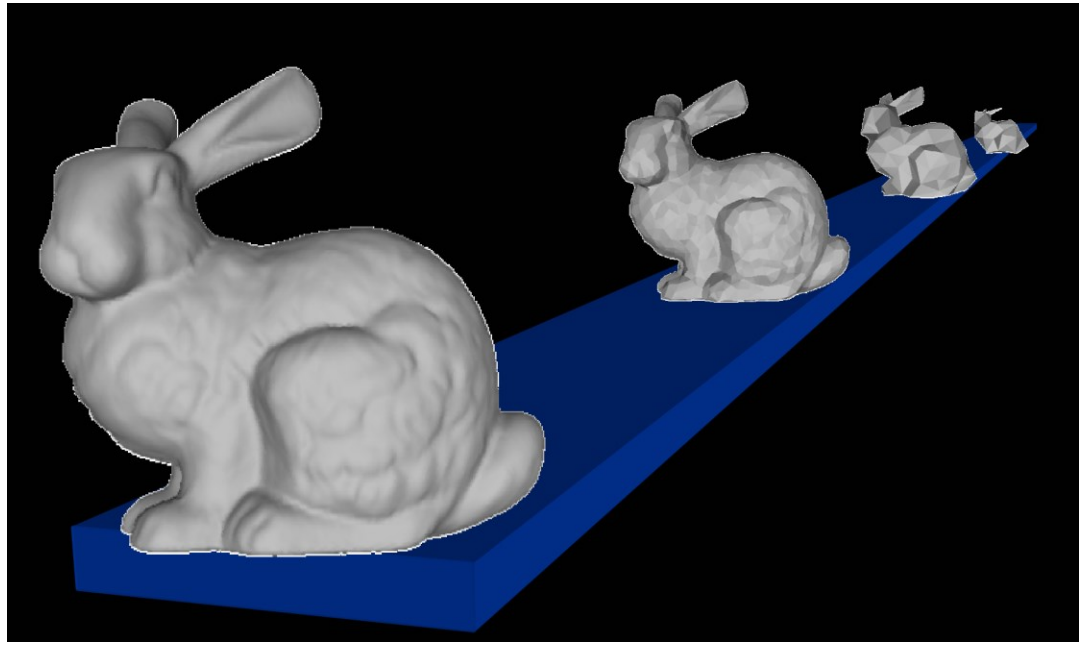
Laplacian

Constrain vertices

Level of Detail

- Allow objects to be represented with different numbers of polygons
- Use fewer polygons for distant objects
  - Less visual contribution
- Use more polygons for near objects
  - More visual contribution





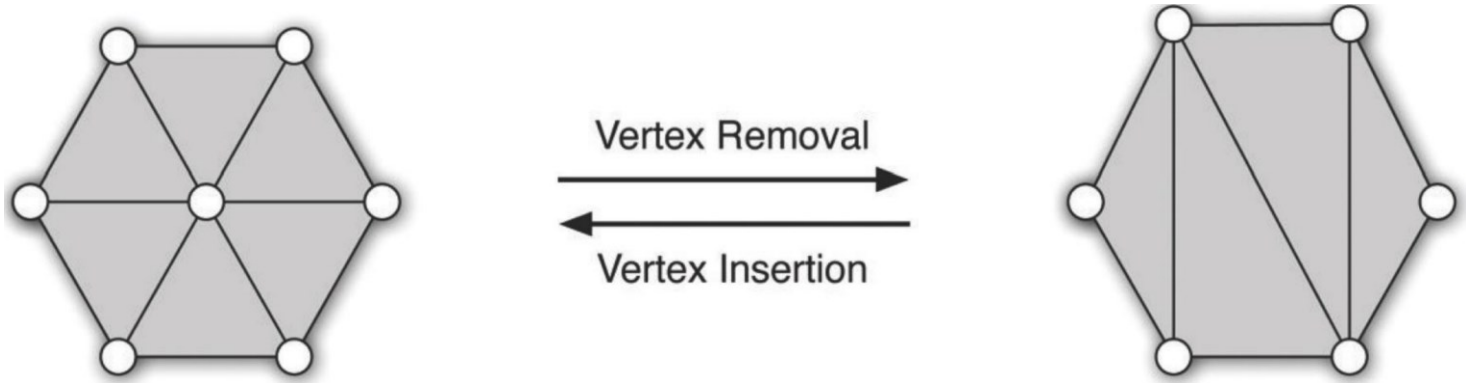
Mesh Simplification

- Transform a given polygonal mesh into another mesh with **fewer** faces, edges, and vertices
- The simplification can be done either statically (as a preprocessing), or dynamically on the fly

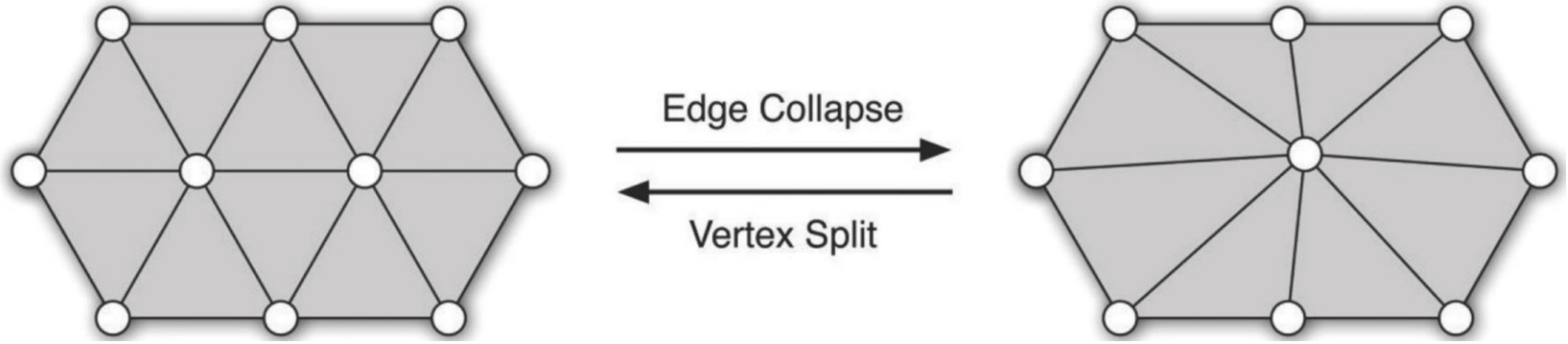
Mesh Simplification

- Delete unnecessary vertices, edges and triangles

- Vertex removal

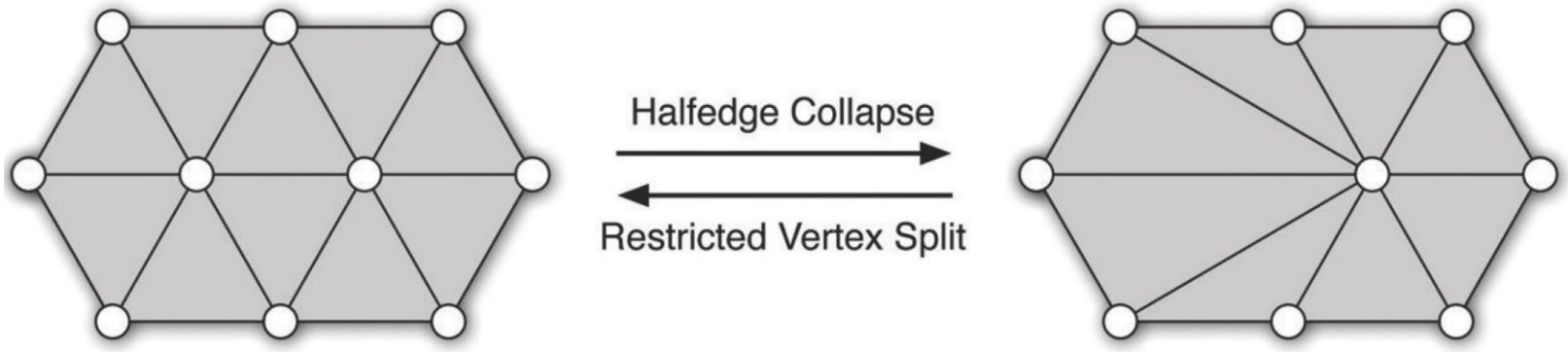


- Edge collapse

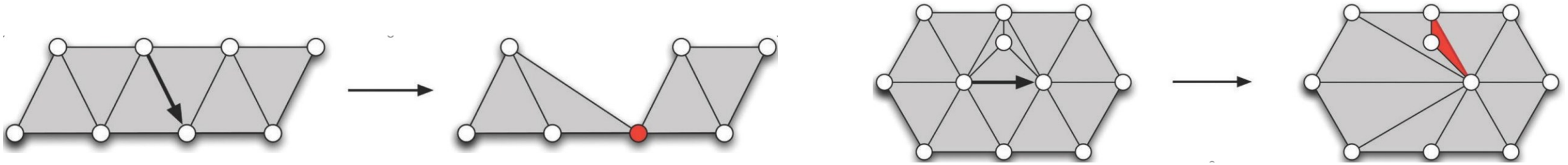


Mesh Simplification

- Half-edge collapse

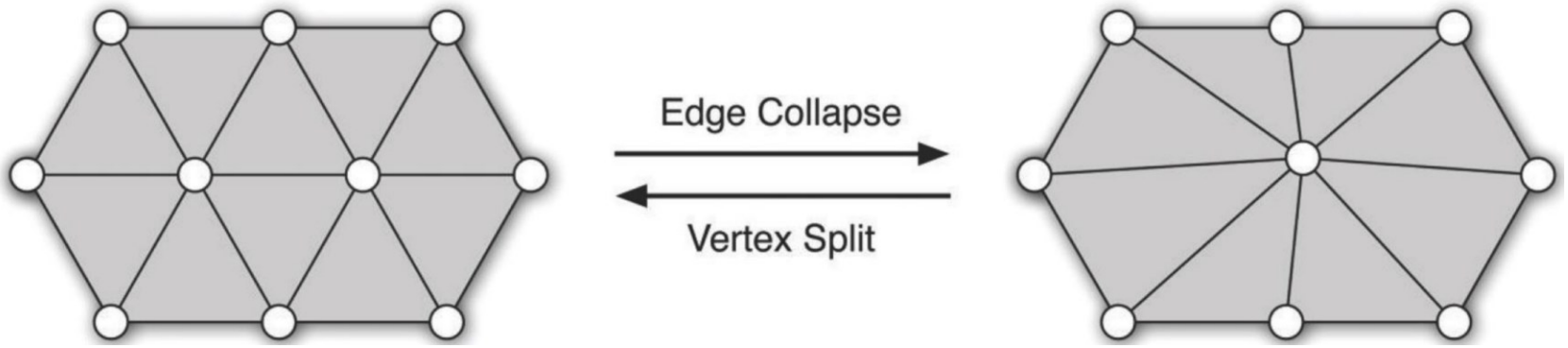


- Topologically illegal (half-)edge collapses



Collapsing An Edge [Garland & Heckbert 1997]

- How much does it cost to collapse an edge?
- Basic idea: compute edge midpoint, measure quadric error



- Better idea: choose a point that minimizes quadric error
- More details: Garland & Heckbert 1997.

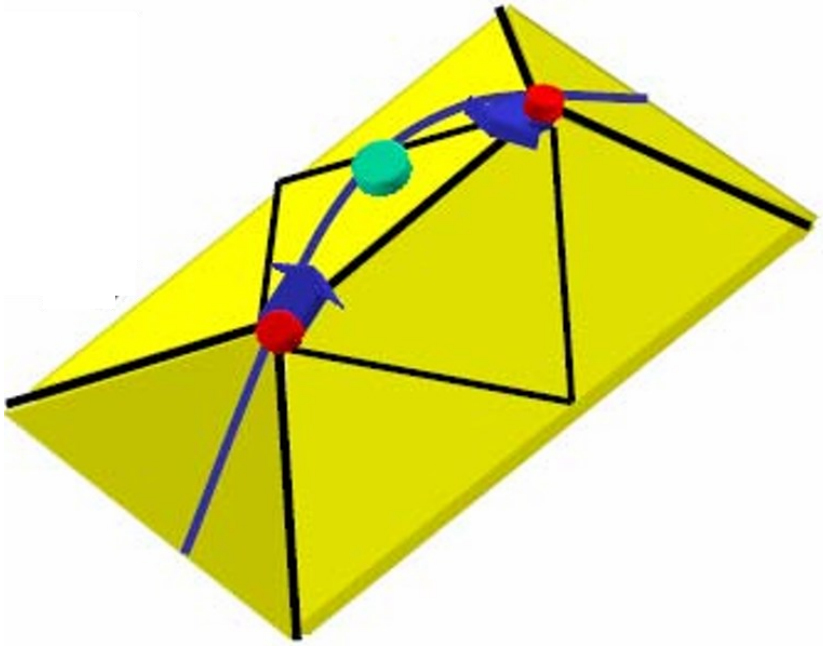
Quadric Error Metrics [Garland & Heckbert 1997]

- Not a good idea to perform local averaging of vertices
- How much geometric error is introduced by the Edge collapse?
- Quadric error: new vertex should minimize its sum of square distance (L2 distance) to previously related triangle planes!

Square distance to a single plane: $h_{\mathrm{d}}^2 = \tilde{\mathbf{v}}^{\mathsf{T}} \left(\frac{\tilde{\mathbf{n}}\tilde{\mathbf{n}}^{\mathsf{T}}}{\mathbf{n}\mathbf{n}^{\mathsf{T}}} \right) \tilde{\mathbf{v}}$

Quadric error: sum of square distances:

$$\mathrm{Error}(\mathbf{v}) = \tilde{\mathbf{v}}^{\mathsf{T}} \sum \mathrm{K}_p \tilde{\mathbf{v}} = \tilde{\mathbf{v}}^{\mathsf{T}} \mathrm{Q}\, \tilde{\mathbf{v}}$$



Mesh Simplification with QEM [Garland & Heckbert 97]

Algorithm Outline

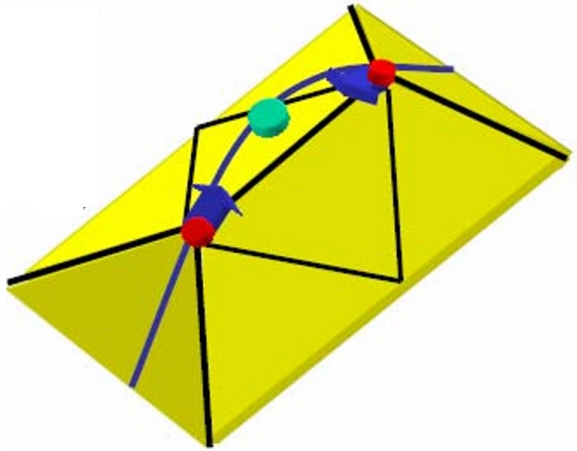
Initialization

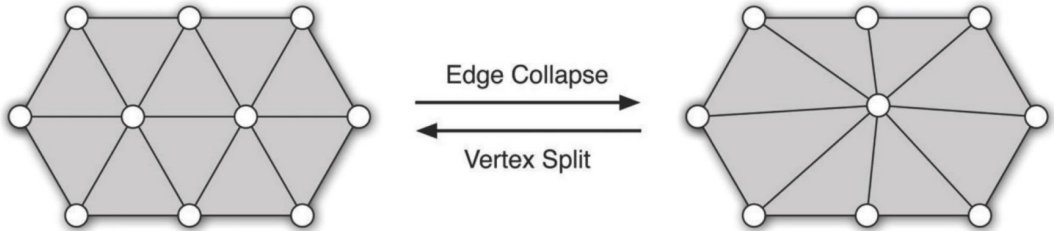
- Compute quadric **Q** for each vertex
- Select set of valid vertex pairs (edges + non-edges)
- Compute minimal cost candidate for each pair

Iteration

- Select lowest cost pair (v_1,v_2)
- Contract (v_1,v_2) — **Q** for new vertex is $\mathbf{Q}_1+\mathbf{Q}_2$
- Update all pairs involving v_1 & v_2

$$\text{Error}(\mathbf{v}) = \tilde{\mathbf{v}}^{\top}\sum \mathrm{K}_p\tilde{\mathbf{v}} = \tilde{\mathbf{v}}^{\top}\mathrm{Q}\,\tilde{\mathbf{v}}$$

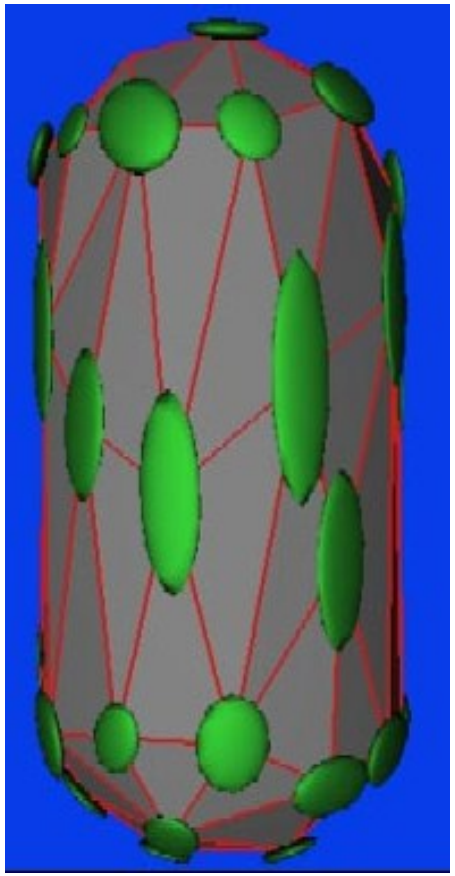


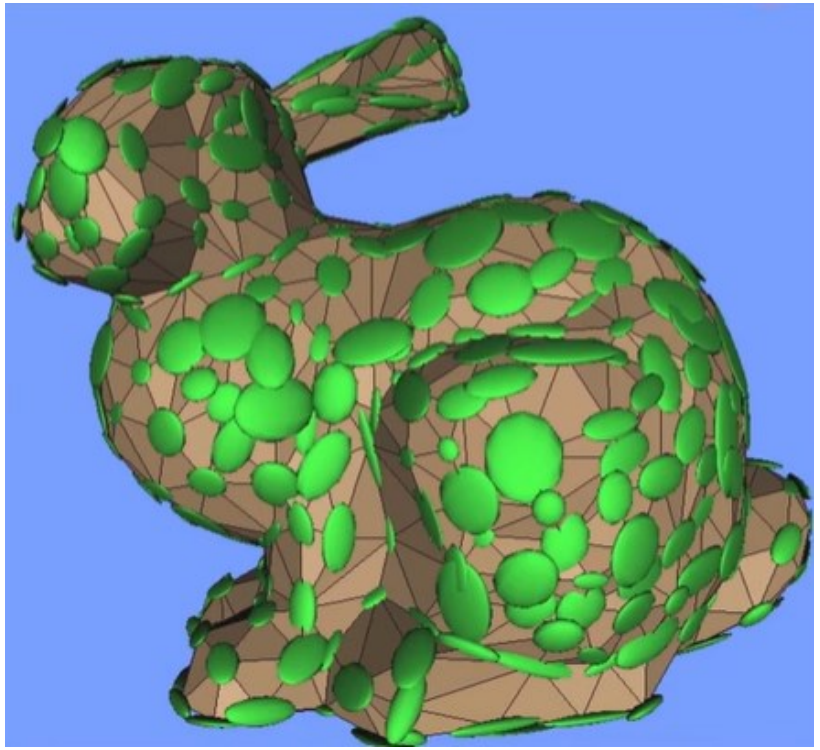


https://mgarland.org/files/papers/s97-quadric-talk.pdf

What does QEM[Garland & Heckbert 97] measure?

$$\text{Error}(\mathbf{v}) = \tilde{\mathbf{v}}^{\mathsf{T}} \mathrm{Q} \tilde{\mathbf{v}}$$





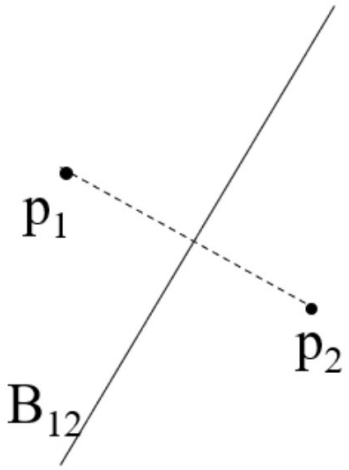
- Quadric isosurfaces
  - Are ellipsoids (maybe degenerate)
  - Centered around vertices
  - Characterize shape
  - Stretch in least-curved directions

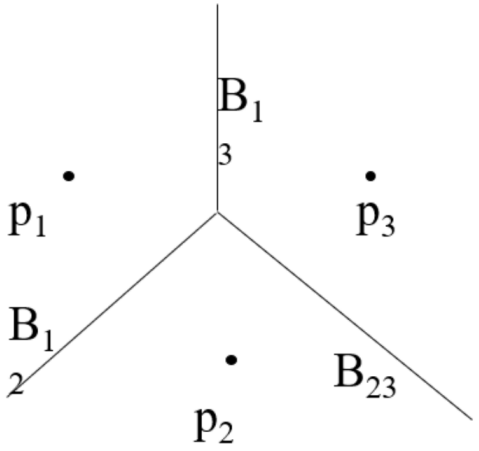
https://mgarland.org/files/papers/s97-quadric-talk.pdf

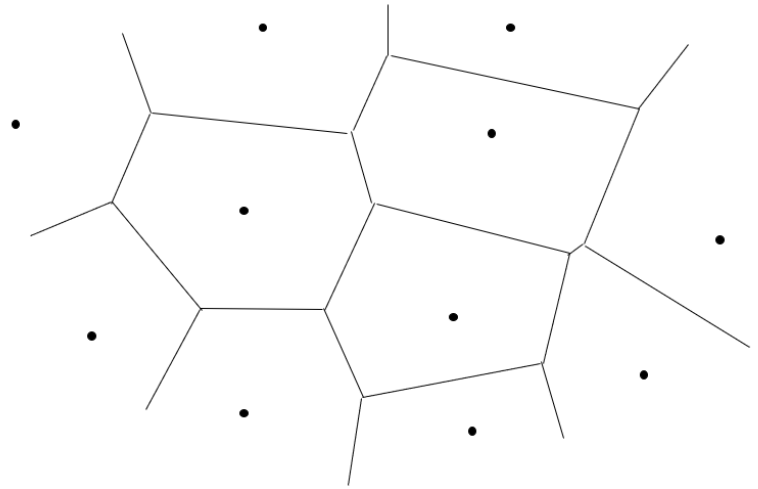
Voronoi diagram

- Voronoi diagram is a partition of a plane into regions close to each of a given set of objects.
- All those points assigned to p_i from the **Voronoi region** $V(p_i)$ consists of all the points at least as close to p_i as to any other site:

$$V(p) = \{x \mid d(p_i - x) \leq d(p_j - x), \forall j \neq i\}$$

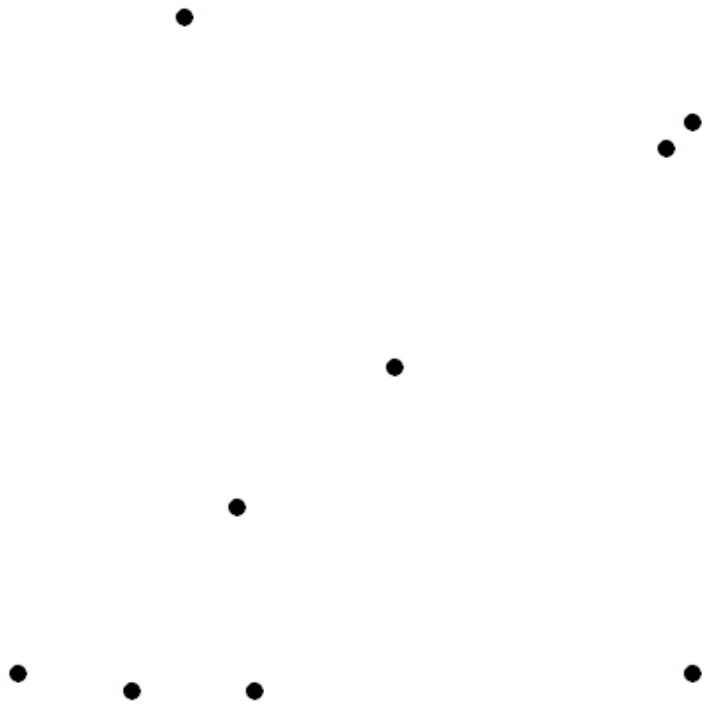






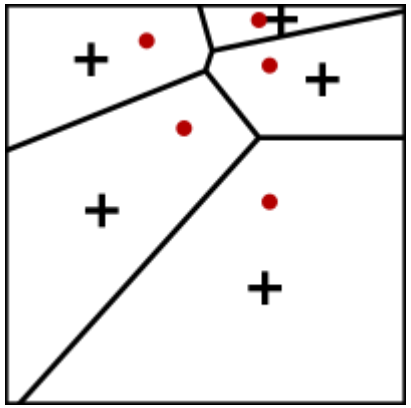
Voronoi diagram

- A Voronoi tessellation emerges by radial growth from seeds outward.

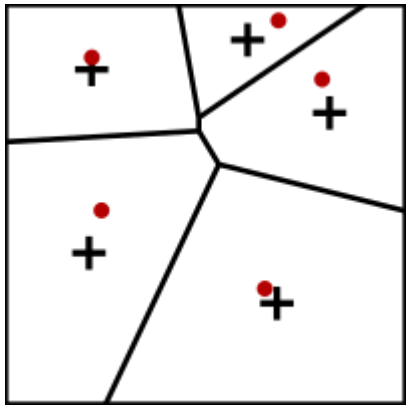


Remeshing

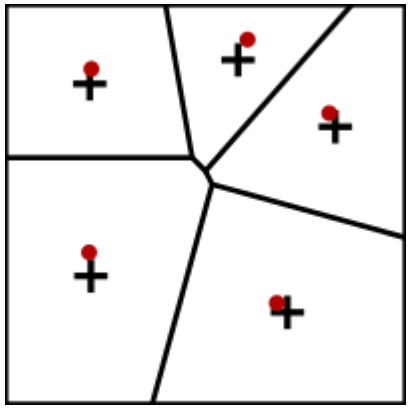
- Centroidal Voronoi Tessellation (CVT)
  - Lloyd's algorithm
  - The Voronoi diagram of the k sites is computed.
  - Each cell of the Voronoi diagram is integrated, and the centroid is computed.
  - Each site is then moved to the centroid of its Voronoi cell.



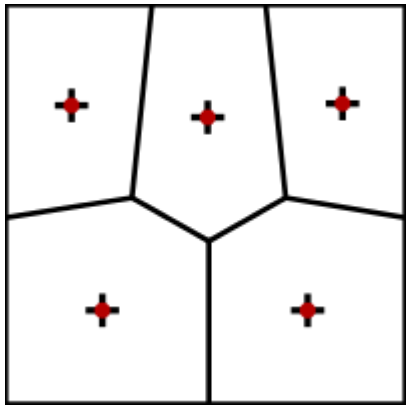
Iteration 1



Iteration 2



Iteration 3



Iteration 15

The Voronoi diagram of the current points at each iteration is shown. The "+" signs denote the centroids of the Voronoi cells.

Delaunay triangulation

- The geometric dual of the Voronoi diagram. There is a line segment connecting Pi and Pj in a Delaunay triangulation if and only if the Voronoi Diagram Regions of Pi and Pj share the same edge.
- Property: for each triangle of the triangulation, the circumcircle of that triangle is empty of all other sites.
- We will explain later in ‘reconstruction’