

# Geometry Processing

## 2 Discrete Differential Geometry

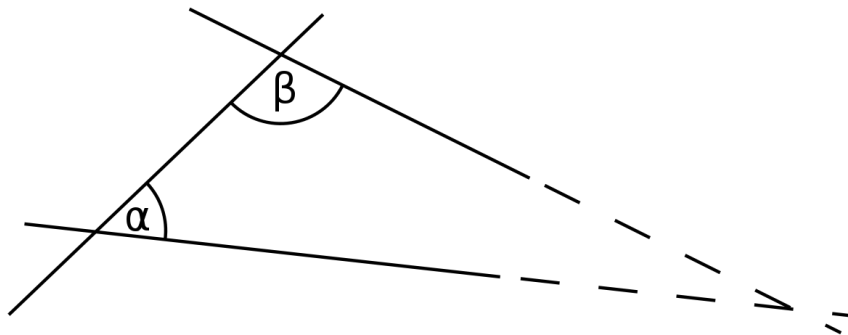
Ludwig-Maximilians-Universität München

# Session 2: Discrete Differential Geometry

- Motivation
- Discrete Geometric Quantities
  - Normals
  - Curvature
  - Laplace-Beltrami
- Summary
- Discussion
  - .OBJ Mesh Loader
  - Blender Python APIs
  - Blender BMesh Structure

# Euclidean v.s. Non Euclidean: Parallel Postulate

*"In a plane, given a line and a point not on it, at most one line parallel to the given line can be drawn through the point."*

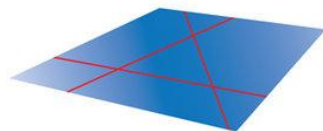


$$\alpha + \beta = \pi$$



Euclid

*Geometry principles works differently on curved spaces...*



Euclidean

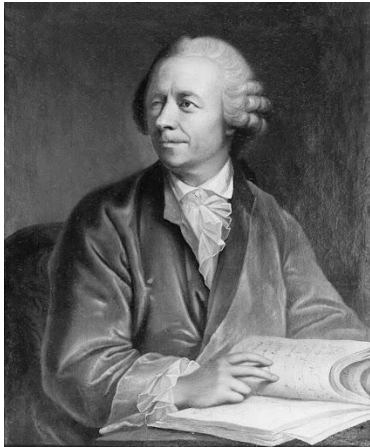


Elliptic



Hyperbolic

# Differential Geometry

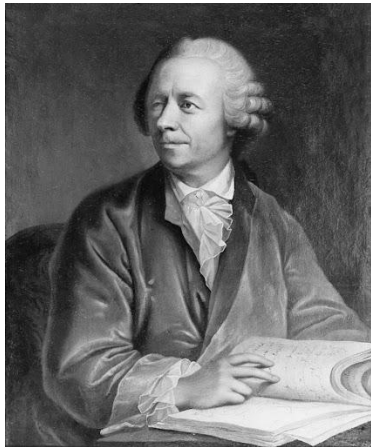


Leonhard  
Euler



Carl  
Gauss

# Differential Geometry



Leonhard  
Euler



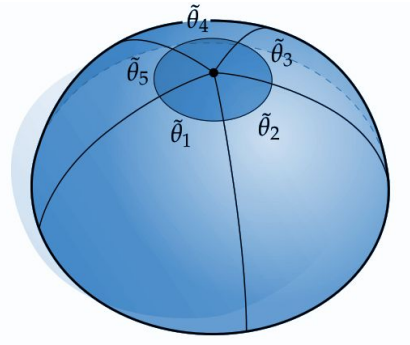
Carl  
Gauss



Bernhard  
Riemann

# Example: Smooth Settings

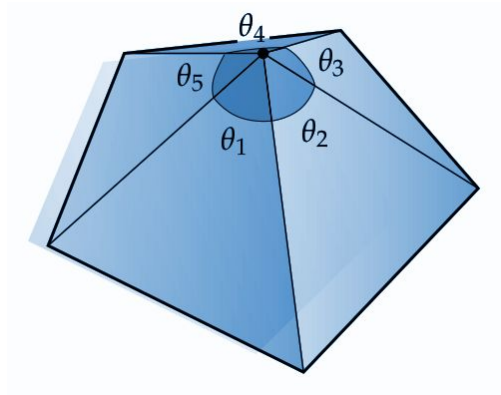
In smooth settings, the sum of the tip angle is always  $2\pi$



$$\sum \theta_i = 2\pi$$

# Example: Discrete Settings

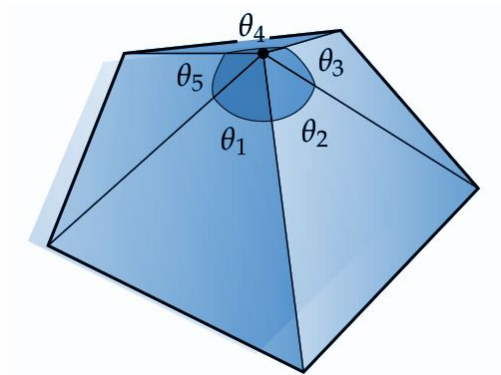
In discrete settings, the sum of the tip angle is not  $2\pi$  but approximately  $2\pi$  if we have infinite tessellated triangles



$$\sum \theta_i < 2\pi \quad (\text{why?})$$

# Example: Discrete Settings

In discrete settings, the sum of the tip angle is not  $2\pi$  but approximately  $2\pi$  if we have infinite tessellated triangles



$$\sum \theta_i < 2\pi$$

Redefine  $\hat{\theta}_j = \theta_j \frac{2\pi}{\sum_i \theta_i}$

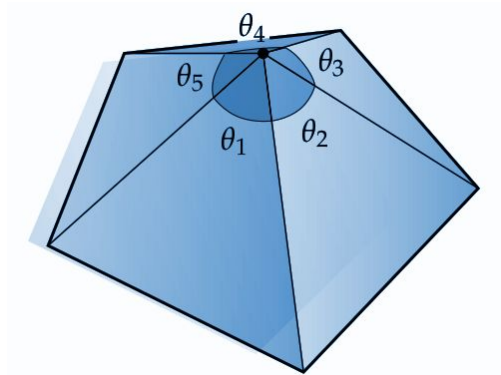
$$\Rightarrow \sum_j \hat{\theta}_j = \sum_j \theta_j \frac{2\pi}{\sum_i \theta_i} = \frac{2\pi}{\sum_i \theta_i} \sum_j \theta_j = 2\pi$$

By redefining the meaning of "angle", we preserved the geometric property that the sum of the tip angle is  $2\pi$



# Example: Discrete Settings

In discrete settings, the sum of the tip angle is not  $2\pi$  but approximately  $2\pi$  if we have infinite tessellated triangles



$$\sum \theta_i < 2\pi$$

Redefine  $\hat{\theta}_j = \theta_j \frac{2\pi}{\sum_i \theta_i}$

$$\Rightarrow \sum_j \hat{\theta}_j = \sum_j \theta_j \frac{2\pi}{\sum_i \theta_i} = \frac{2\pi}{\sum_i \theta_i} \sum_j \theta_j = 2\pi$$

Caution (Ambiguity of Interpretation):

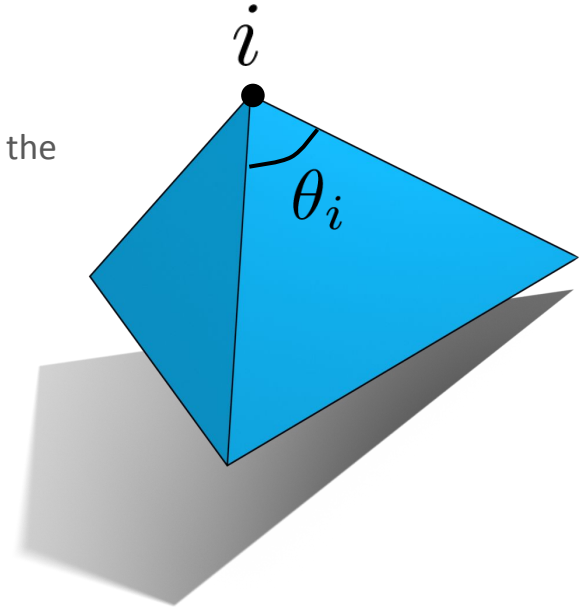
*We are assuming the mesh is representing a smooth surface, but what if it is intended to represent a "hard" surface?*

# Angle Defect

Important idea: Represent how flatten (or how curved) around a vertex  $i$

The **angle defect** at a vertex is the deviation of the sum of interior angles from the Euclidean angle sum of  $2\pi$ :

$$\Omega_i = 2\pi - \sum \theta_i$$



# Angle Defect

Basic idea: Represent how flatten (or how curved) around a vertex  $i$

The **angle defect** at a vertex is the deviation of the sum of interior angles from the Euclidean angle sum of  $2\pi$ :

$$\Omega_i = 2\pi - \sum \theta_i$$

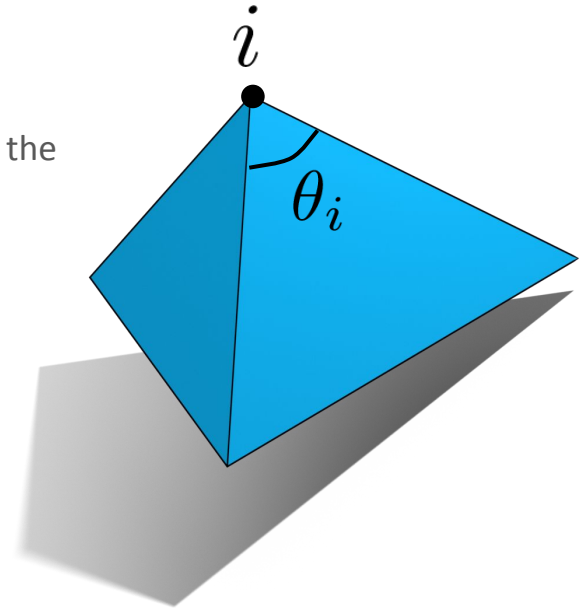
Discrete **Gauss-Bonnet Theorem**:

For a simplicial surface, the total angle defect is

$$\sum_i \Omega_i = 2\pi\chi$$

(Euler Characteristic)

E.g. Given a convex polyhedra, the total angle defect is  $4\pi$  (Try to verify this)

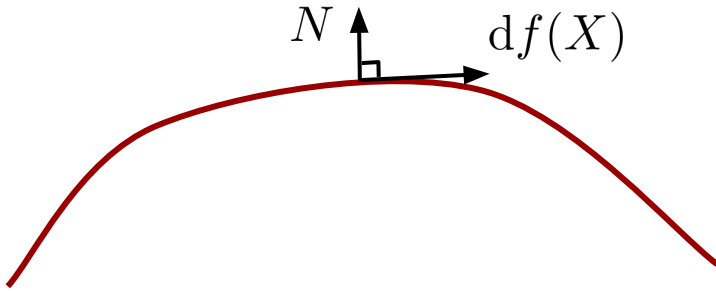


# Session 2: Discrete Differential Geometry

- Motivation
- Discrete Geometric Quantities
  - Normals
  - Curvature
  - Laplace-Beltrami
- Summary
- Discussion
  - .OBJ Mesh Loader
  - Blender Python APIs
  - Blender BMesh Structure

# Normals

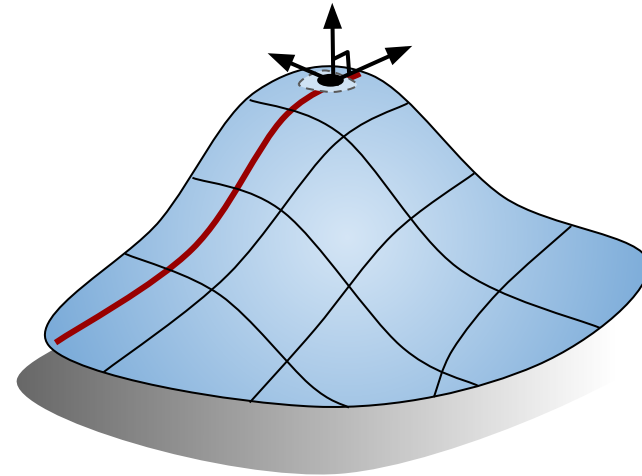
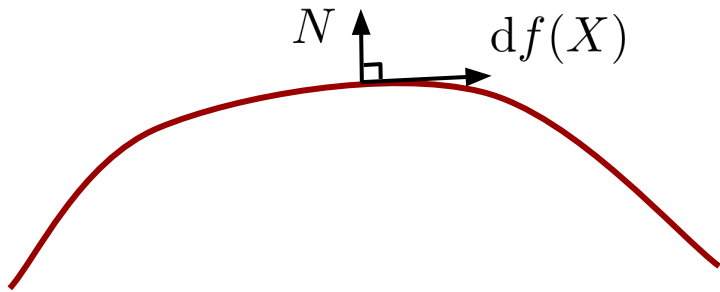
On a curve: A vector is **normal** to a surface if it is orthogonal to all tangent vectors



# Normals

On a curve: A vector is **normal** to a surface if it is orthogonal to all tangent vectors

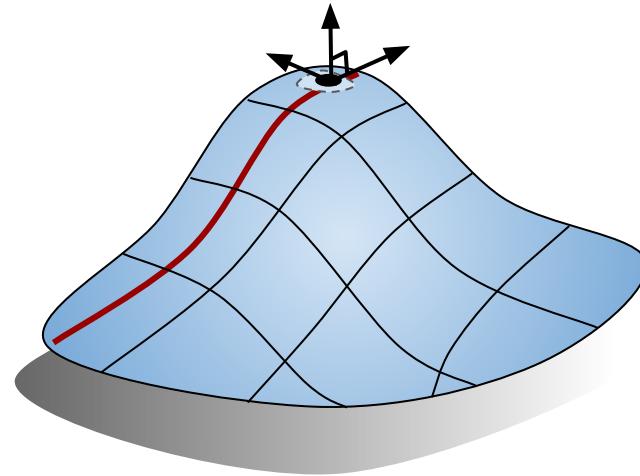
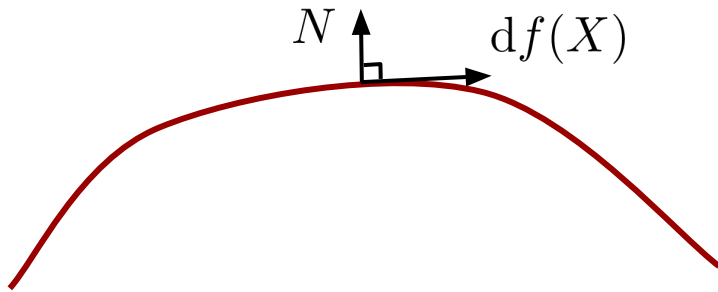
On a surface: A normal is a unit vector along with the cross product of any given two tangent vectors



# Normals

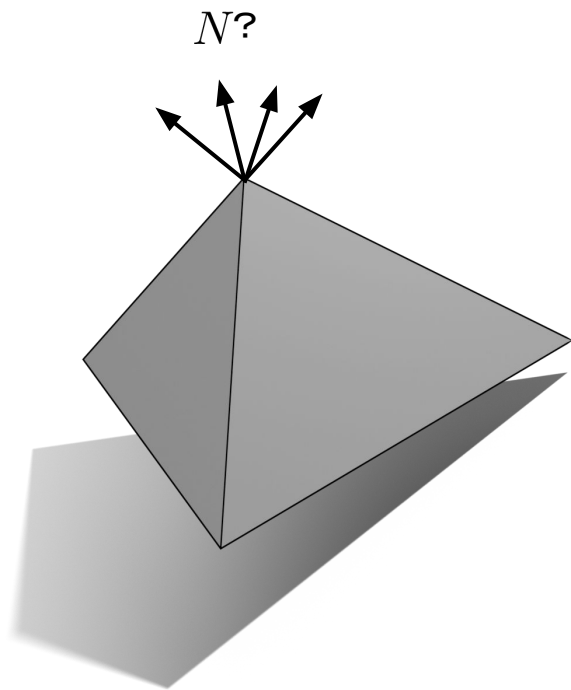
On a curve: A vector is **normal** to a surface if it is orthogonal to all tangent vectors

On a surface: A normal is a unit vector along with the cross product of any given two tangent vectors



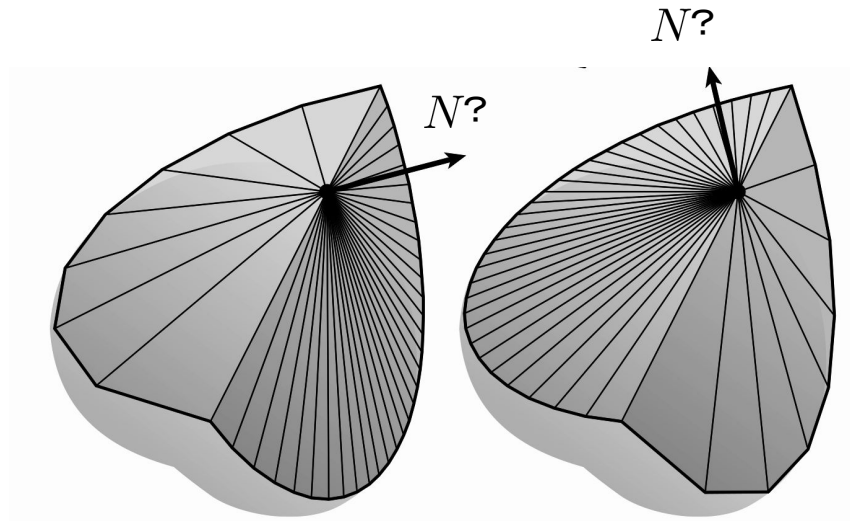
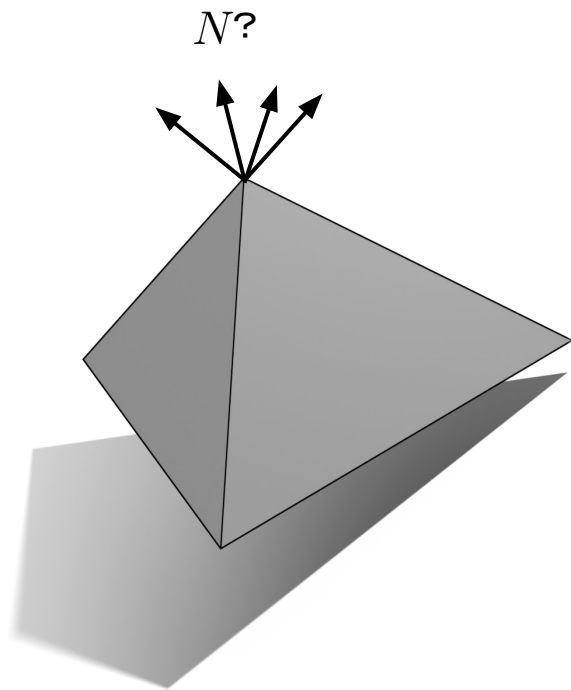
Q: How to discretize the definition on polygonal meshes?

# Discrete Normals





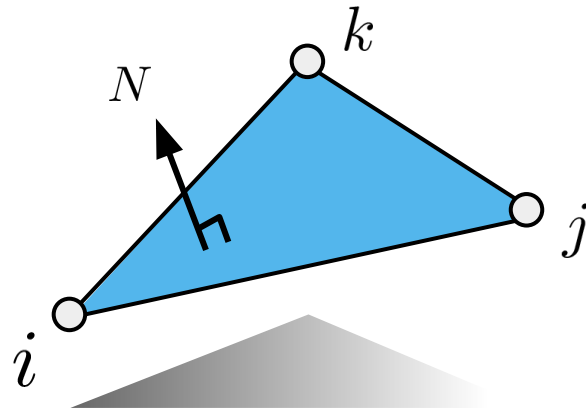
# Discrete Normals



# Face Normal

Face normals are well-defined:

$$N = \frac{(f_j - f_i) \times (f_k - f_i)}{\|(f_j - f_i) \times (f_k - f_i)\|}$$



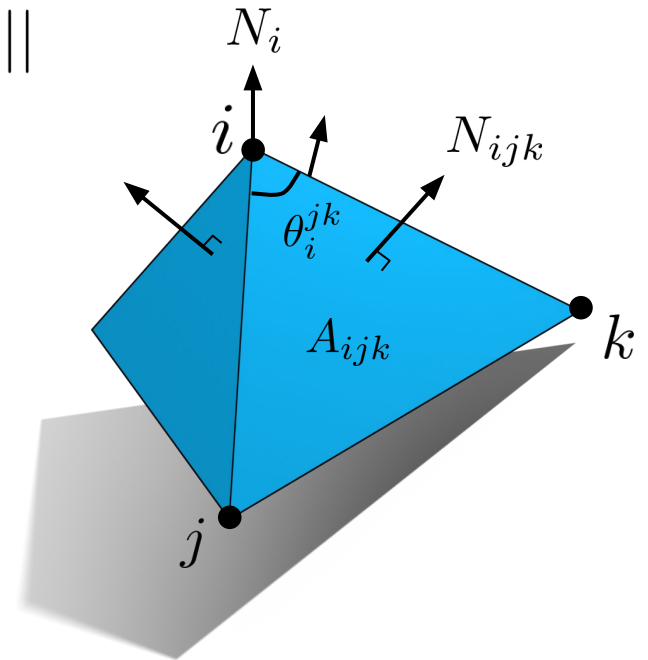
# Vertex Normal

Basic idea: weighted average of the normal vectors of incident faces

$$N_i = \frac{\sum_i w_{ijk} N_{ijk}}{\|\sum_i w_{ijk} N_{ijk}\|}$$

Variances:

- Uniform (or Equally) Weighted  $w_{ijk} = 1$



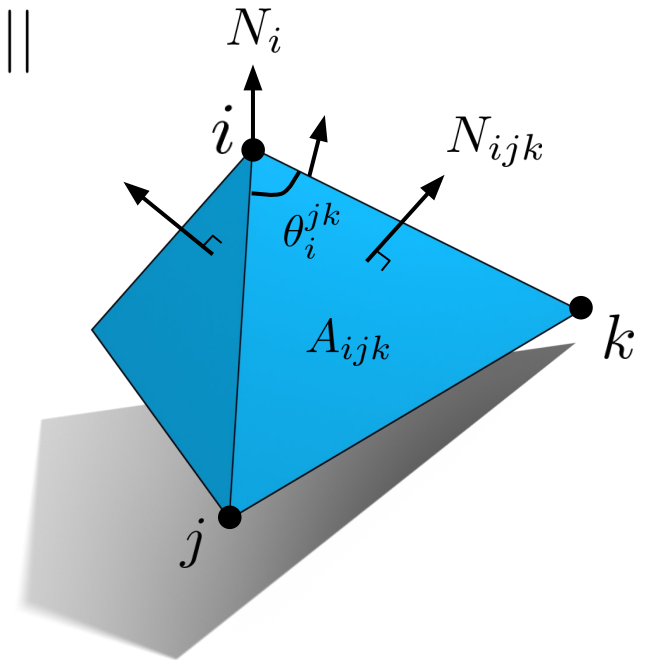
# Vertex Normal

Basic idea: weighted average of the normal vectors of incident faces

$$N_i = \frac{\sum_i w_{ijk} N_{ijk}}{\|\sum_i w_{ijk} N_{ijk}\|}$$

Variances:

- Uniform (or Equally) Weighted  $w_{ijk} = 1$
- Area Weighted  $w_{ijk} = A_{ijk}$



# Vertex Normal

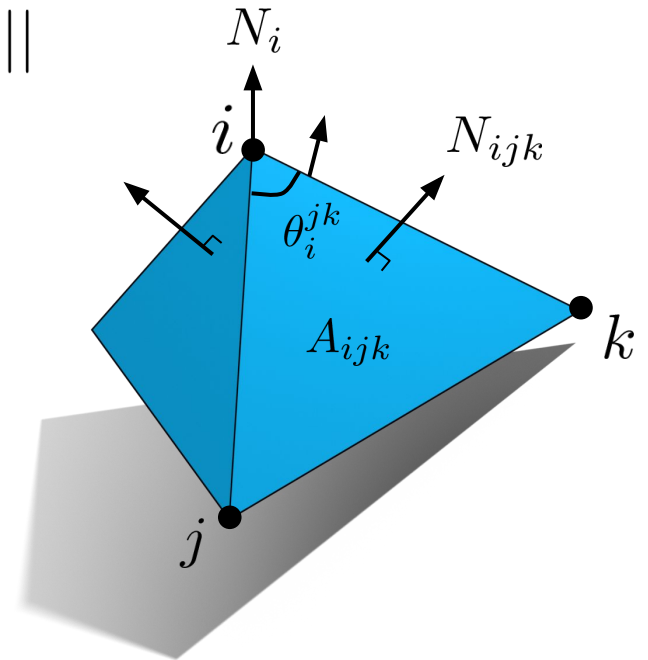
Basic idea: weighted average of the normal vectors of incident faces

$$N_i = \frac{\sum_i w_{ijk} N_{ijk}}{\|\sum_i w_{ijk} N_{ijk}\|}$$

Variances:

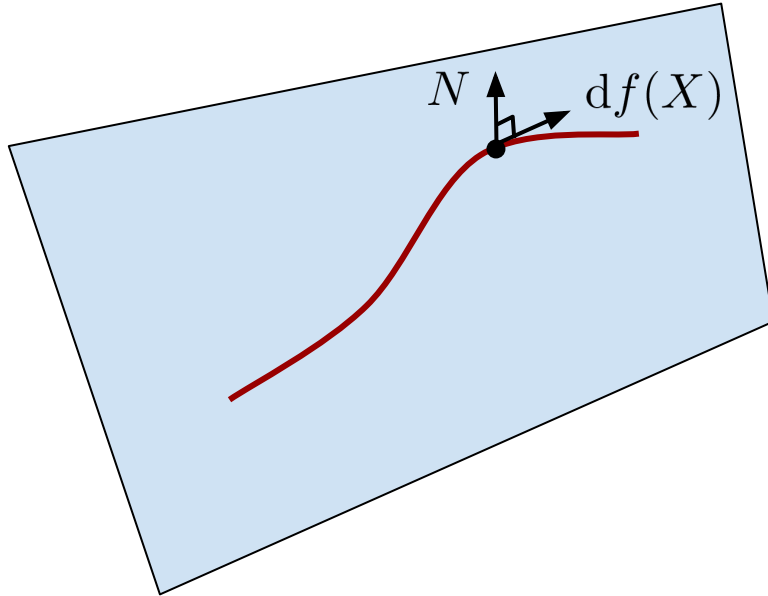
- Uniform (or Equally) Weighted  $w_{ijk} = 1$
- **Area Weighted**  $w_{ijk} = A_{ijk}$
- **Angle Weighted**  $w_{ijk} = \theta_i^{jk}$
- ...

*Caution:* face normal v.s. vertex normal v.s. normal interpolation



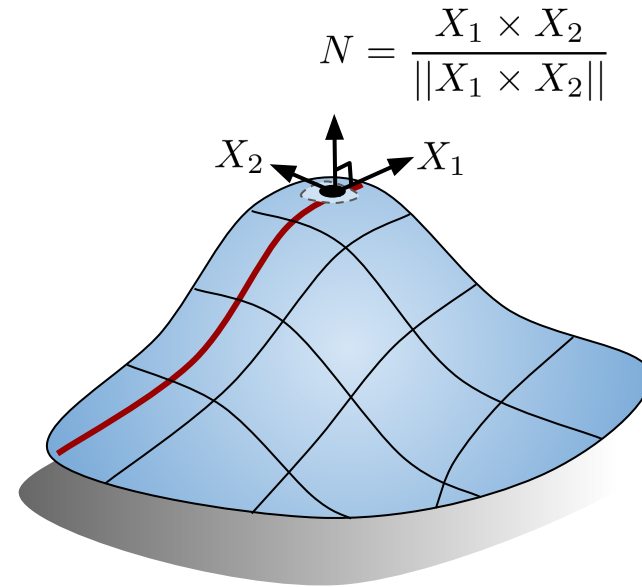
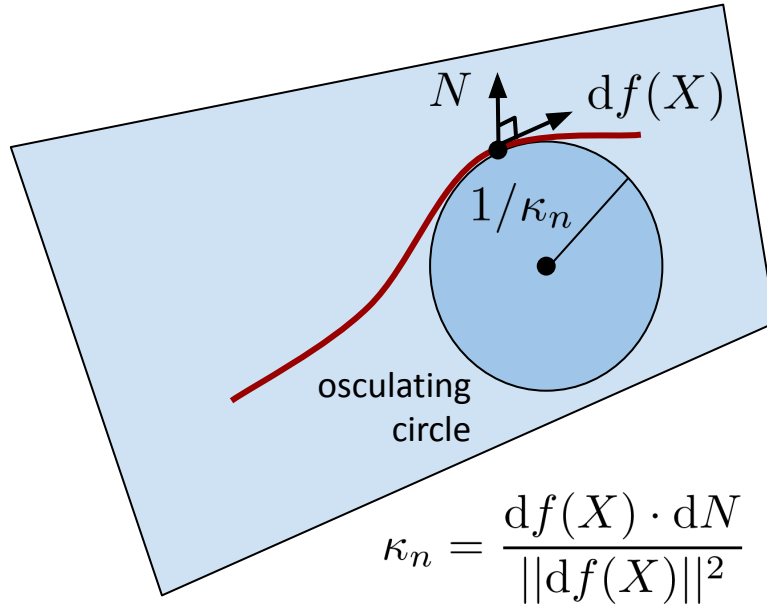
# Curvature

Intuitively, curvature describes "how much a curve bends", or the rate of change in the tangent, or *second derivative*



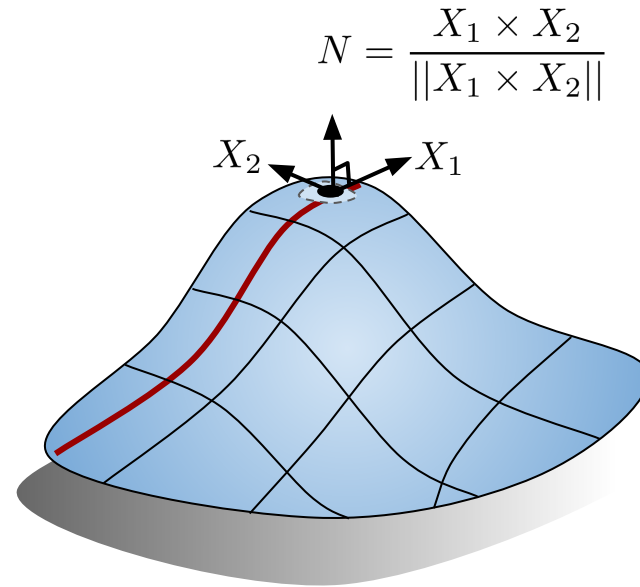
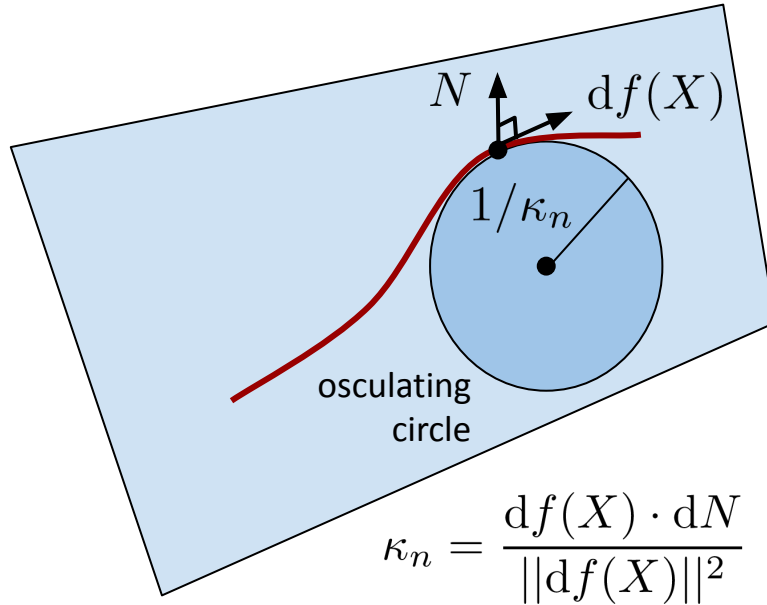
# Normal Curvature $\kappa_n$

The rate at which normal is bending along a given tangent direction



# Normal Curvature $\kappa_n$

The rate at which normal is bending along a given tangent direction

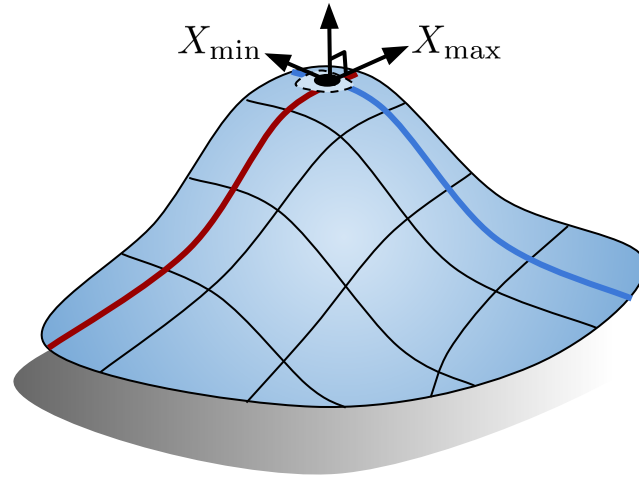


**Q:** which direction does the surface bend the most?



# Principal Curvature $\kappa_{\min}, \kappa_{\max}$

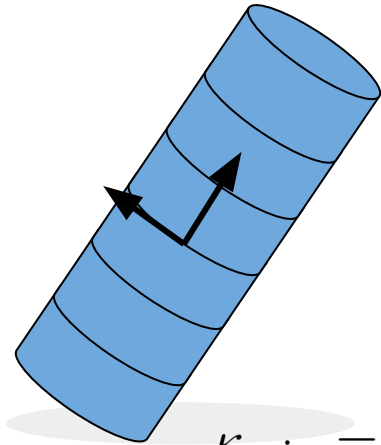
- **Principal directions:** Axes that describe the direction along which the normal changes the most/least
- **Principal curvatures:** along all directions, the two principal directions where normal curvature has minimum and maximum value respectively



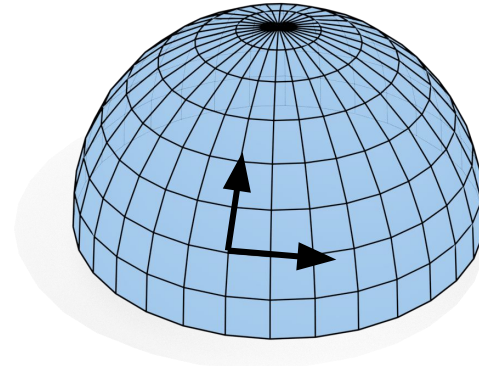
Some facts:

- (Euler's Theorem) principal directions are orthogonal
- $dN = \kappa df(X)$

# Principal Curvature: Examples



$$\kappa_{\min} = 0, \kappa_{\max} = 1/r$$



*"Umbilic Points"*

$$\kappa_{\min} = \kappa_{\max} = 1/r$$

# Principial Curvature: Visualized



$\kappa_{\min}$



$\kappa_{\max}$



# Gaussian and Mean Curvature



$$K = \kappa_1 \kappa_2$$



$$H = \frac{\kappa_1 + \kappa_2}{2}$$

Q: Why Gaussian/Mean curvature is interesting to us?

# Discussion Break: Smooth Curvature

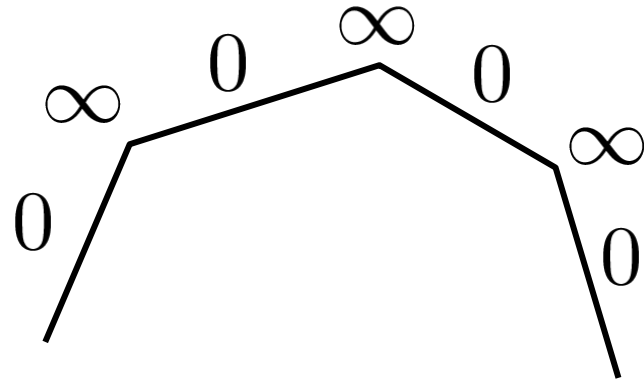
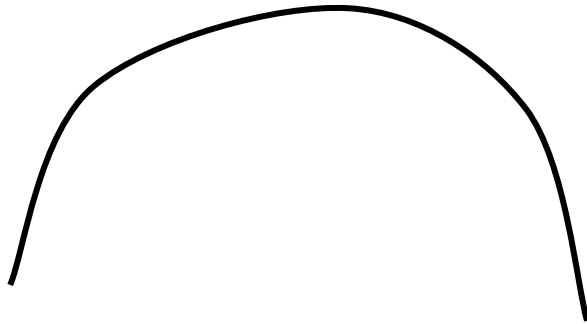
# How do we actually compute curvatures in a discrete world?

# Discrete Curvature

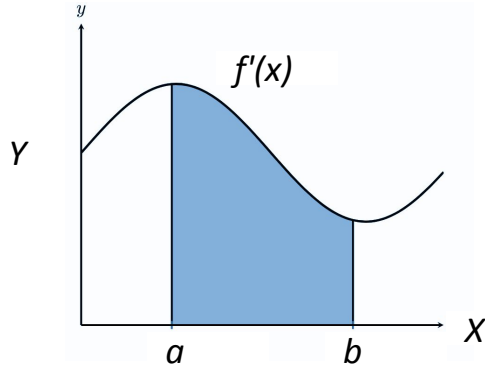
Curvature is the change in normal direction as we travel along the curve

In discrete settings: No change along each edge  $\Rightarrow$  zero curvature?

$$dN = \kappa df(X)$$



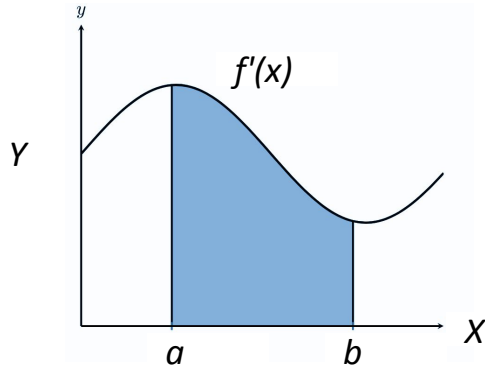
# Revisit: Fundamental Theorem of Calculus



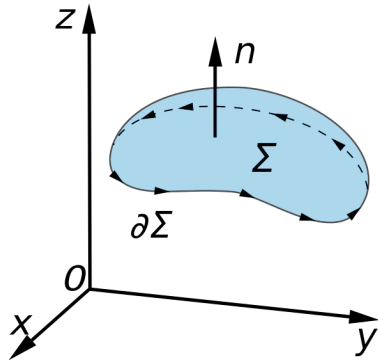
$$\int_a^b df = f(b) - f(a)$$



# Revisit: Fundamental Theorem of Calculus and Stokes' Theorem



$$\int_a^b df = f(b) - f(a)$$



Insights from Stokes' Theorem:

The change we see on the outside is purely a function of the change within.

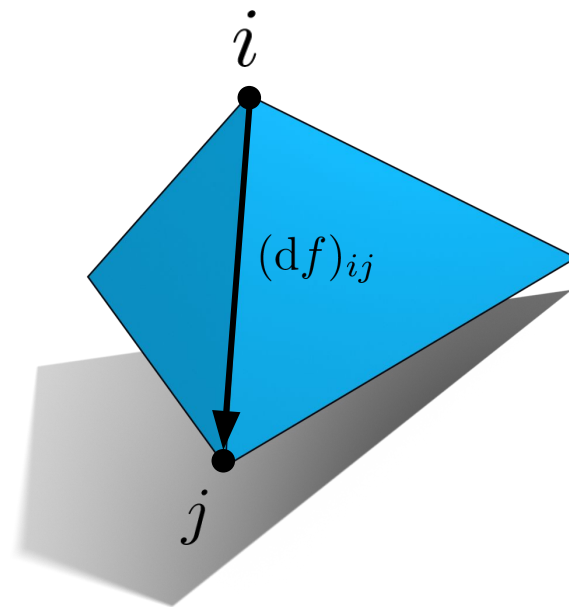
# Example: Discrete *Differential*

Discrete differential is just edge vectors in discrete settings

$$(df)_{ij} = f_j - f_i$$



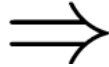
(Stokes' theorem)



# Discrete *Principal Curvature*

$$K = \kappa_1 \kappa_2$$

$$H = \frac{\kappa_1 + \kappa_2}{2}$$



$$\kappa_1 = \|H\| - \sqrt{H^2 - K}$$

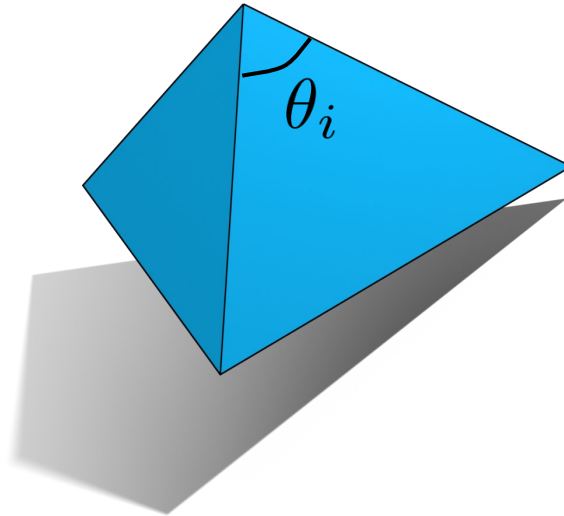
$$\kappa_2 = \|H\| + \sqrt{H^2 - K}$$

Then the question is: how to compute gaussian and mean curvature?

# Discrete *Gaussian Curvature*

We already know how to compute Gaussian curvature: the angle defect is a good approximation

$$K = \Omega_i = 2\pi - \sum_i \theta_i$$



# Discrete Mean Curvature

?

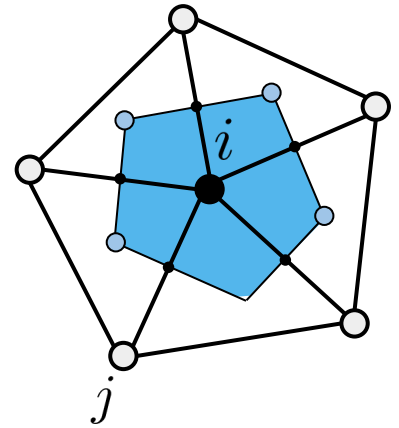
# Laplacian

Basic idea: Laplacian is (scalar) deviation from local average

$$\Delta f = \nabla \cdot \nabla f = \sum_i \frac{\partial^2 f}{\partial x_i^2}$$

(Discrete) Laplacian is the divergence of gradient

$$(\Delta f)_i = w_i \sum_{ij} w_{ij} (f_j - f_i)$$



# Cotangent Formula

A more accurate discretization of the Laplace-Beltrami operator

Basic idea: integrate the divergence of the gradient of a piecewise linear function over a *local averaging region*

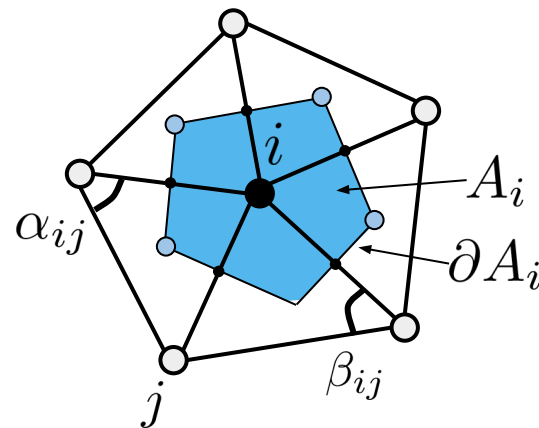
(e.g. *voronoi cell*):

$$\int_{A_i} \Delta f dA = \int_{\partial A_i} \nabla f \cdot N ds$$

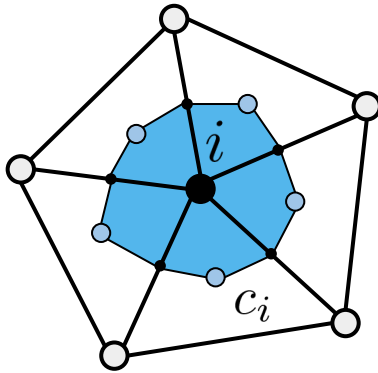
↑  
(Stokes' theorem)

One can prove:

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

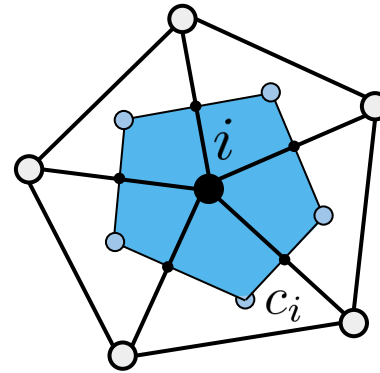


# Local Averaging Region



Barycentric Cell

$C_i$  = barycenter



Voronoi Cell

$C_i$  = circumcenter



# The *Laplace-Beltrami Operator*

The discrete version of the Laplace operator, of a function at a vertex  $i$  is given as

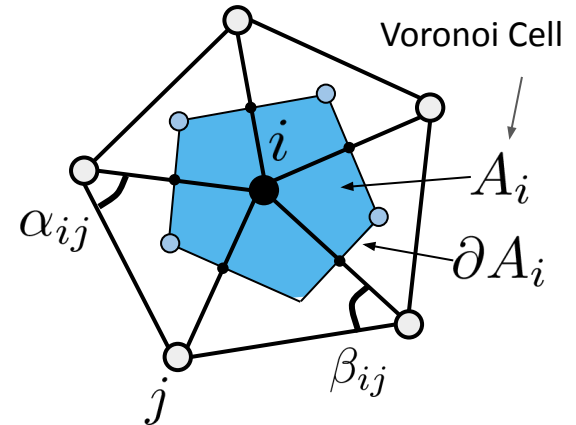
$$(\Delta f)_i = w_i \sum_{ij} w_{ij} (f_j - f_i)$$

This (cotan-version) is the most widely used discretization of the Laplace-Beltrami operator for geometry processing:

$$(\Delta f)_i = \frac{1}{2A_i} \sum_{ij} (\cot \alpha_{ij} + \cot \beta_{ij}) (f_j - f_i)$$

The mean curvature is tightly related to the cotan Laplace-Beltrami:

$$H = \frac{1}{2} \|(\Delta f)_i\|$$



# Discrete Mean Curvature

The Laplace-Beltrami operator is tightly related to the mean curvature:

$$\|\Delta f\| = 2\|H\|\|N\| \Rightarrow \|H\| = \frac{1}{2}\|(\Delta f)_i\|$$

Implementation thinking: Is it necessary to keep the  $\frac{1}{2}$  factor?

# Computing Discrete Curvatures

**Mean curvature:** via Laplace-Beltrami (length of Laplacian vector)

$$\|H\| = \frac{1}{2} \|(\Delta f)_i\|$$

**Gaussian curvature:** via angle defect

$$K = \Omega_i$$

**Principal curvature:** via Gaussian and Mean

$$\kappa_1 = \|H\| - \sqrt{H^2 - K}$$

$$\kappa_2 = \|H\| + \sqrt{H^2 - K}$$

# Application: Anisotropic Remeshing [Alliez et al. 2003]



Input Mesh

# Application: Anisotropic Remeshing [Alliez et al. 2003]

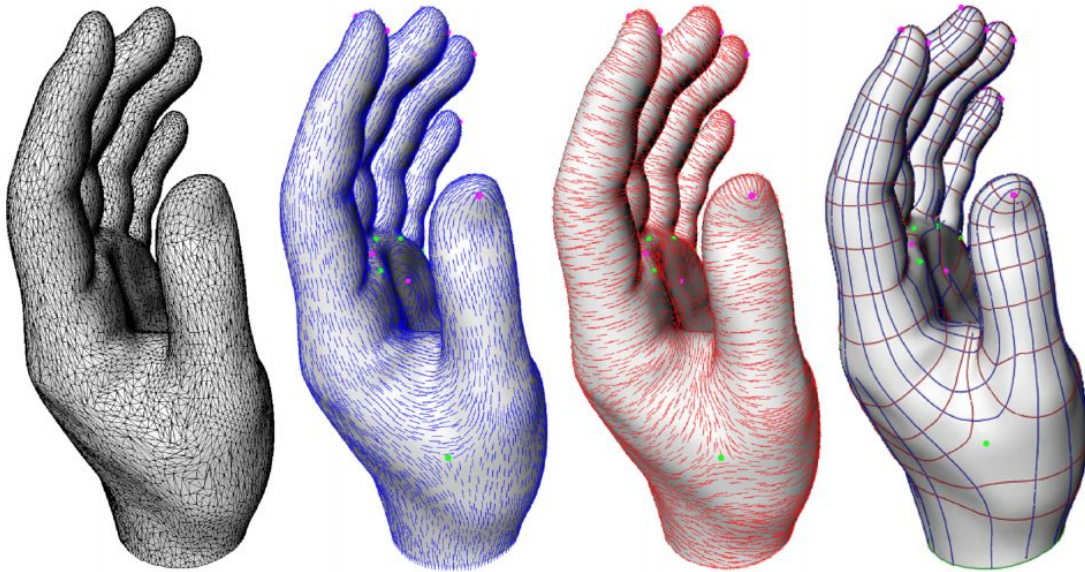


Input Mesh

Kmin

Kmax

# Application: Anisotropic Remeshing [Alliez et al. 2003]



Input Mesh

Kmin

Kmax

Sampling

# Application: Anisotropic Remeshing [Alliez et al. 2003]



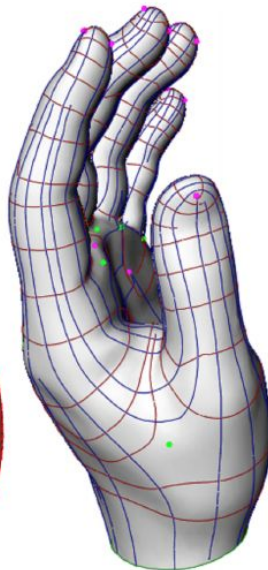
Input Mesh



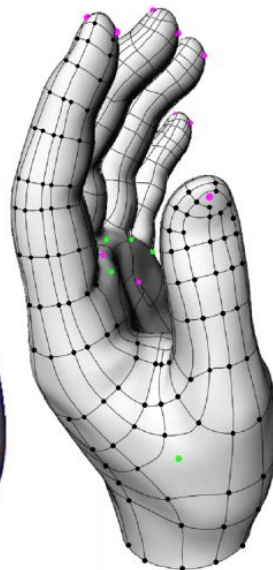
Kmin



Kmax



Sampling



Meshing

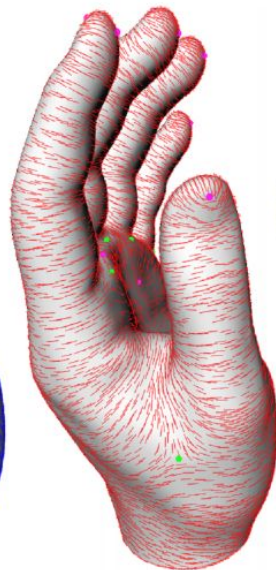
# Application: Anisotropic Remeshing [Alliez et al. 2003]



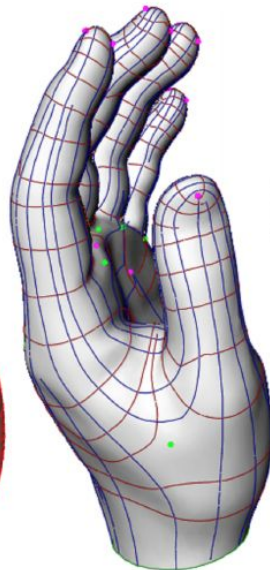
Input Mesh



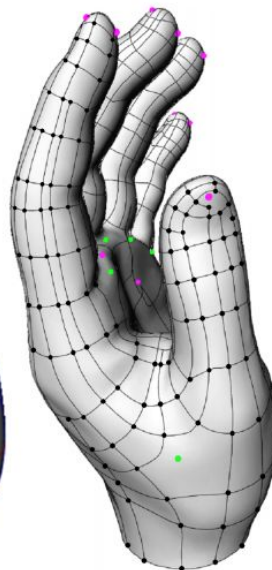
Kmin



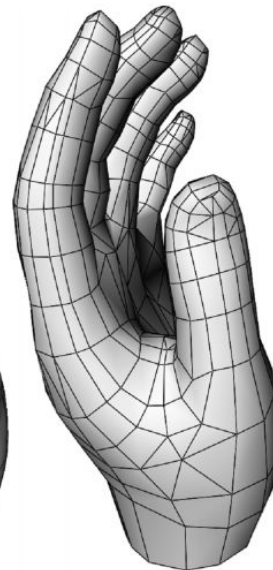
Kmax



Sampling



Meshing



Output Mesh



# Application: Anisotropic Remeshing [Alliez et al. 2003]



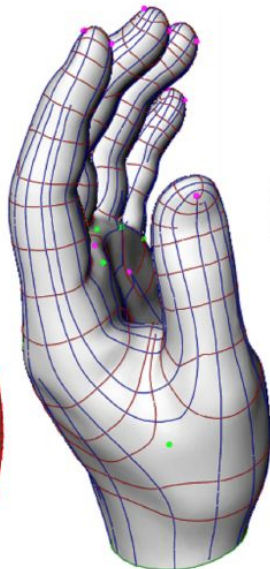
Input Mesh



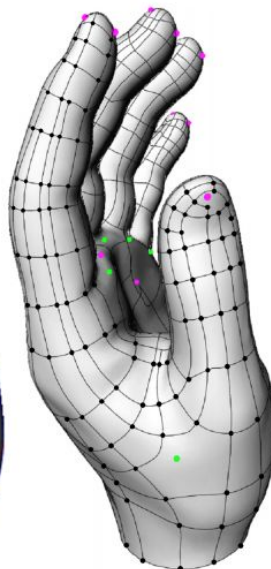
Kmin



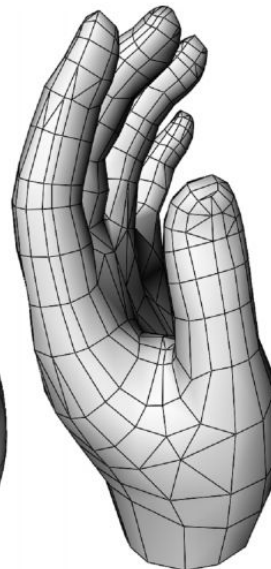
Kmax



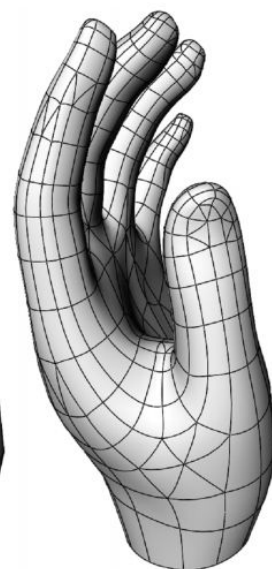
Sampling



Meshing

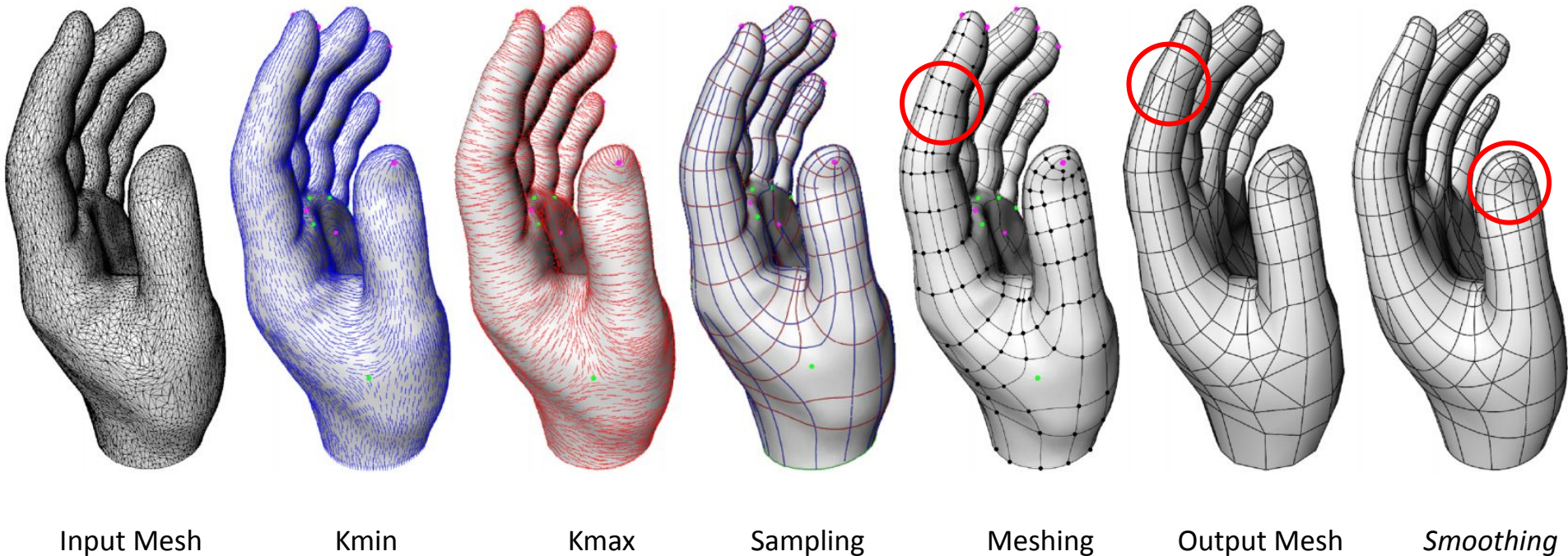


Output Mesh



*Smoothing*

# Application: Anisotropic Remeshing [Alliez et al. 2003]



\* This is not an novel idea, we will see more advances later in our remeshing session. 😊

# Session 2: Discrete Differential Geometry

- Motivation
- Discrete Geometric Quantities
  - Normals
  - Curvature
  - Laplace-Beltrami

## ● Summary

- Discussion
  - .OBJ Mesh Loader
  - Blender Python APIs
  - Blender BMesh Structure

# Summary

- Different discretized definition of a geometry quantity preserves different properties
- Curvature and the Laplacian are the core tools for geometry processing
- No free lunch (again): Compare discretized version to its smooth setting, not all properties can be preserved in discrete settings. Understand the landscape of possibilities of when we should apply a certain definition in a context

# Session 2: Discrete Differential Geometry

- Motivation
- Discrete Geometric Quantifies
  - Normals
  - Curvature
  - Laplace-Beltrami
- Summary
- Discussion
  - .OBJ Mesh Loader
  - Blender Python APIs
  - Blender BMesh Structure

# Implementing A Mesh Loader

## Questions

- What information must be loaded for a minimum implementation?
- How to define a data structure?
- How's the performance of loading?
- What was changed from constructing triangle soup to halfedge representation?

# Blender + Python

<https://docs.blender.org/api/current/>



# Key Concepts

Types: bpy.types

Data: bpy.data

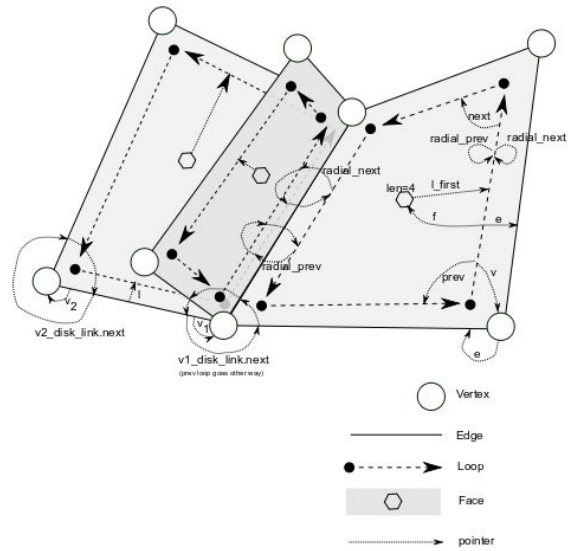
Operator: bpy.ops

Context: bpy.context

BMesh



# BMesh: A Non-Manifold Boundary Representation



<https://wiki.blender.org/w/images/0/06/Dev-BMesh-Structures.png>

# Blender's Mesh Editing APIs

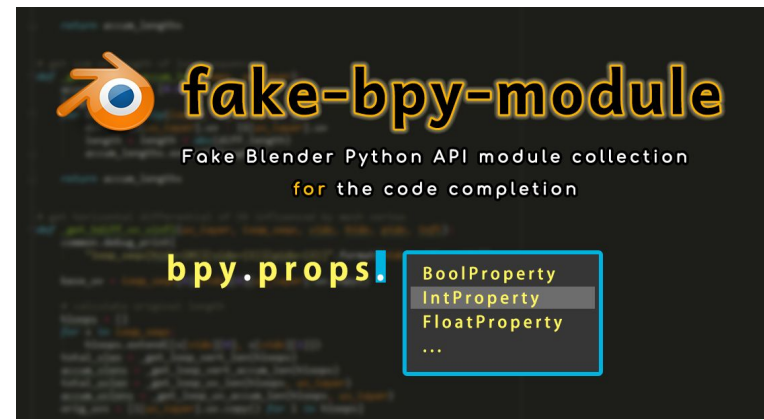
Low-level Operators

Mid-level Operators

Top-level Operators

# Code Completion

```
pip install fake-bpy-module-2.93
```



<https://github.com/nutti/fake-bpy-module>

# Further Readings (Mesh Structures)

Paul Bourke. Data Formats: 3D, Audio, Image. <http://paulbourke.net/dataformats/>

Weiler, K.J. : [The Radial Edge Structure: A Topological Representation for Non-Manifold Geometric Modeling](#). in Geometric Modeling for CAD Applications, Springer Verlag, May 1986.

# Further Readings (Discrete Differential Geometry)

Jin S, Lewis RR, West D. [A comparison of algorithms for vertex normal computation](#). The visual computer. 2005 Feb 1;21(1-2):71-82.

Wardetzky, Max, et al. [Discrete Laplace operators: no free lunch](#). Symposium on Geometry processing. 2007.

Keenan Crane. [Discrete Differential Geometry: An Applied Introduction](#). 2020.