Geometry Processing

2 Discrete Differential Geometry

Ludwig-Maximilians-Universität München

Announcements

- This course can also be recognized as **Praktikum (P5)** for Masters

Announcements

- This course can also be recognized as **Praktikum (P5)** for Masters
- Blender support:
  - Decide if you need PC for using Blender during the semester
  - Send an email titled by "[GP] PC room request" with your name and matriculation number to me (changkun.ou@ifi.lmu.de) using your campus address (**@campus.lmu.de**) before 18.11.2020.
  - You will receive the credentials when the room is ready for you, then
  - You ***must*** make an appointment via email to **labinfo@medien.ifi.lmu.de** before your visit
  - **Room address: Frauenlobstr. 7a Room 352**
  - **Bookable slots: 10:00-13:00 & 13:00-16:00**

Announcements

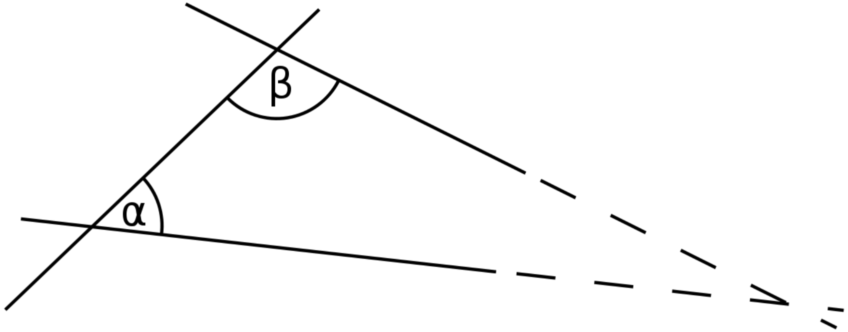
- This course can also be recognized as **Praktikum (P5)** for Masters
- Blender support:
    - Decide if you need PC for using Blender during the semester
    - Send an email titled by "[GP] PC room request" with your name and matriculation number to me (changkun.ou@ifi.lmu.de) using your campus address (**@campus.lmu.de**) before 18.11.2020.
    - You will receive the credentials when the room is ready for you, then
    - You ***must*** make an appointment via email to **labinfo@medien.ifi.lmu.de** before your visit
    - **Room address: Frauenlobstr. 7a Room 352**
    - **Bookable slots: 10:00-13:00 & 13:00-16:00**
- Open your camera so that we know each others more :)

Session 2: Discrete Differential Geometry

- Motivation
- Discrete Geometric Quantifies
  - Normals
  - Curvature
  - Laplace-Beltrami
- Summary
- Discussion: Homework 1
  - .OBJ Mesh Loader
  - Blender Python APIs
  - Blender BMesh Structure

Euclidean v.s. Non Euclidean: Parallel Postulate

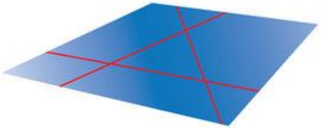
"In a plane, given a line and a point not on it, at most one line parallel to the given line can be drawn through the point."



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

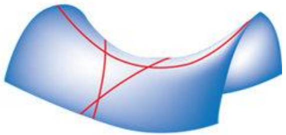
Euclid

Geometry principles works differently on curved spaces...



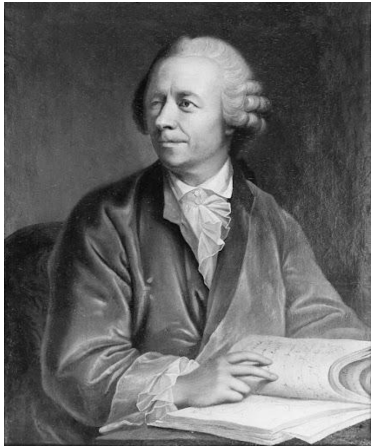
Euclidean

Elliptic



Hyperbolic

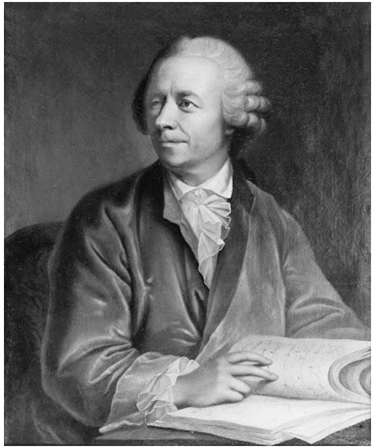
Differential Geometry



Leonhard
Euler

Carl
Gauss

Differential Geometry



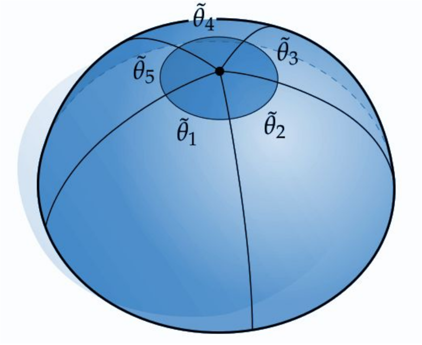
Leonhard
Euler

Carl
Gauss

Bernhard
Riemann

Example: Smooth Settings

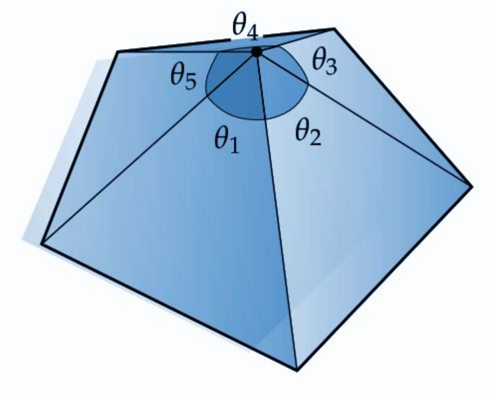
In smooth settings, the sum of the tip angle is always 2π



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

Example: Discrete Settings

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

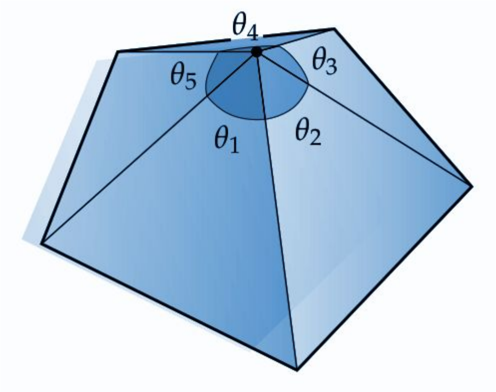


$$\sum \theta_i < 2\pi$$

(why?)

Example: Discrete Settings

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



$$\cancel{\sum \theta_i < 2\pi}$$

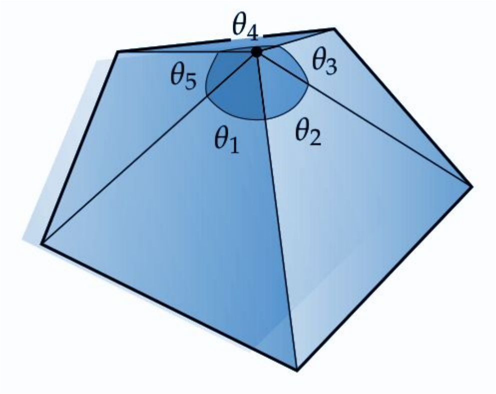
Redefine $\hat{\theta}_j = \theta_j \frac{2\pi}{\sum_i \theta_i}$

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

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

Example: Discrete Settings

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



$$\cancel{\sum \theta_i < 2\pi}$$

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

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

Caution:

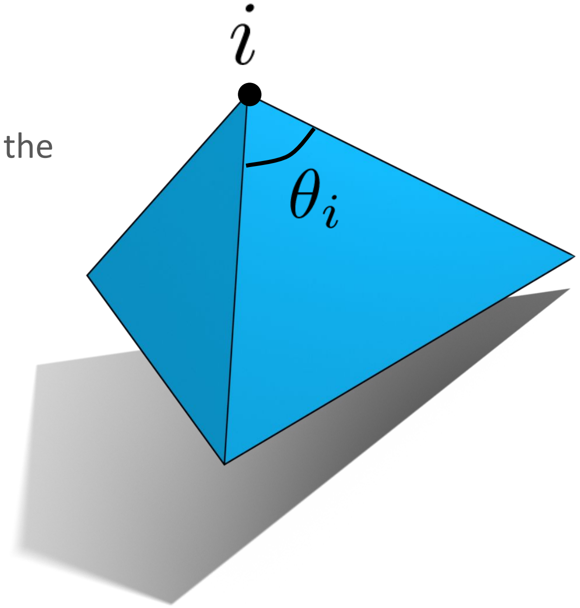
We are assuming the mesh is representing a smooth surface, what if it is intended to represent a "hard" surface?

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 2π:

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

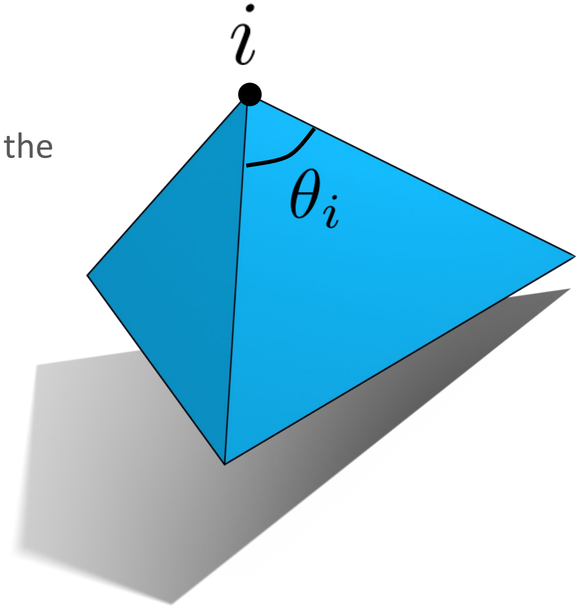


Angle Defect

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 2π:

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



Discrete ***Gauss-Bonnet Theorem***:

For a simplicial surface, the total angle defect is

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

(Euler Characteristic)

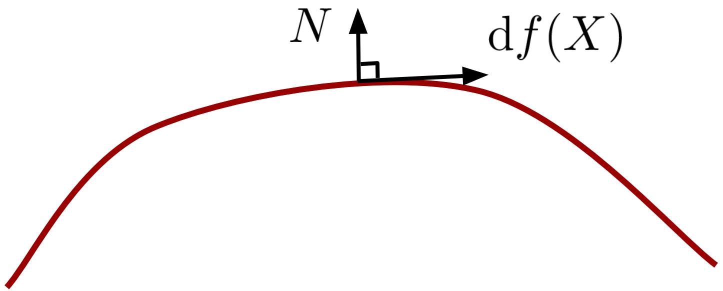
E.g. Given a convex polyhedra, the total angle defect is 4π (Try to verify this in homework)

Session 2: Discrete Differential Geometry

- Motivation
- Discrete Geometric Quantifies
  - Normals
  - Curvature
  - Laplace-Beltrami
- Summary
- Discussion: Homework 1
  - .OBJ Mesh Loader
  - Blender Python APIs
  - Blender BMesh Structure

Normals

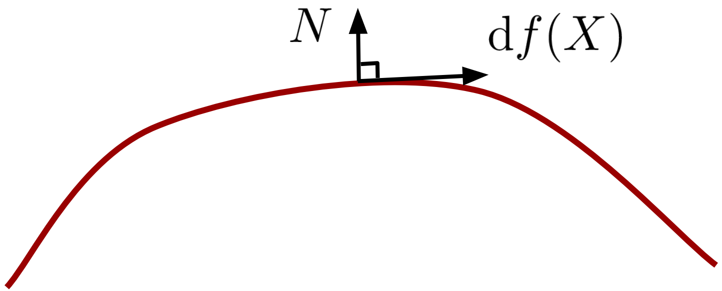
On a curve: *A vector is **normal** to a surface if it is orthogonal to all tangent vectors*

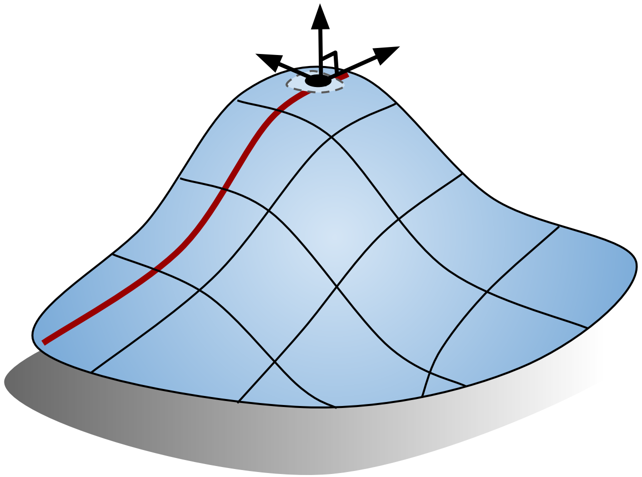


Normals

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*

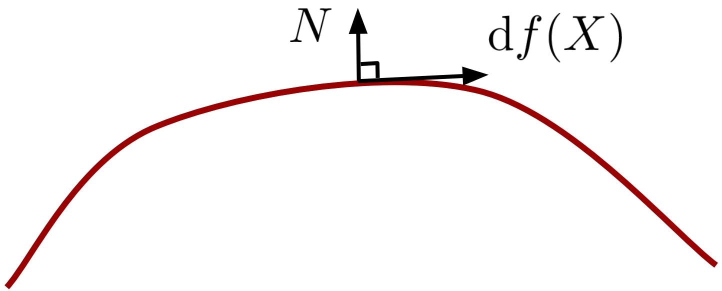


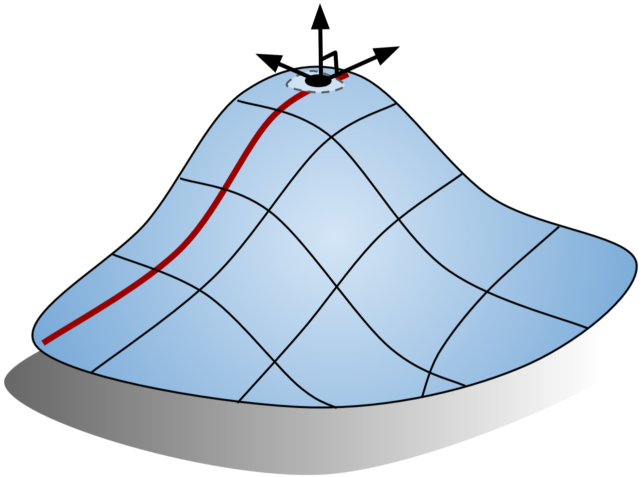


Normals

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

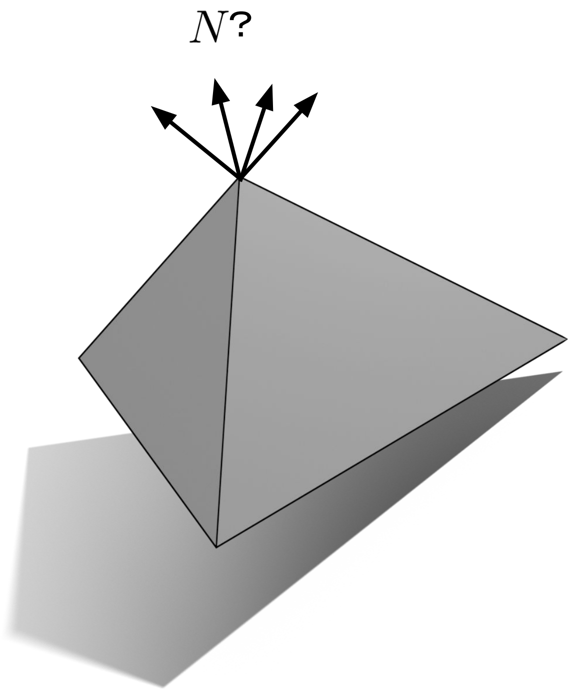
On a surface: *A normal is a unit vector along with the cross product of any given two tangent vectors*



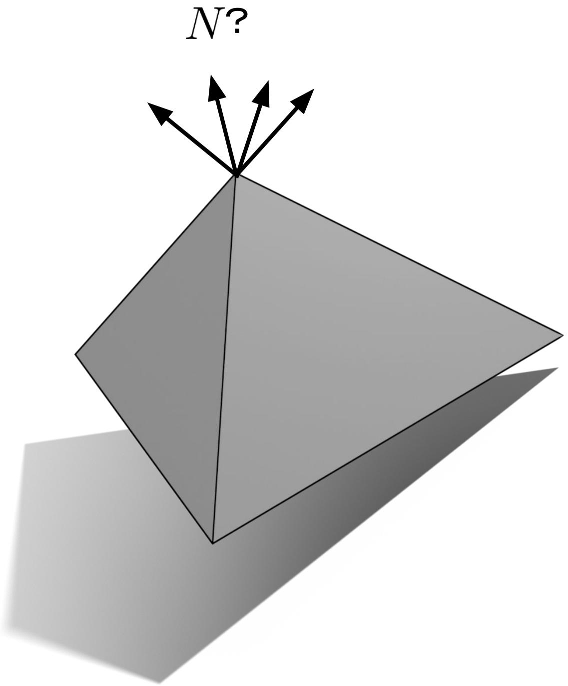


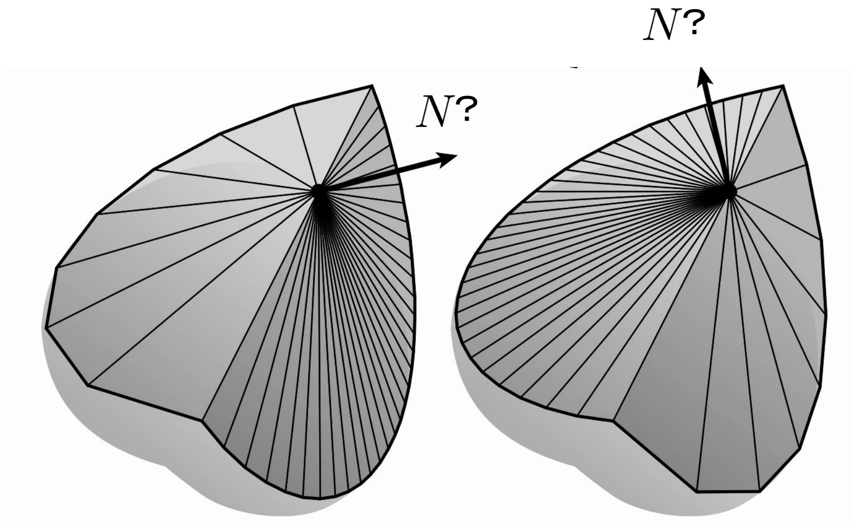
Q: How to discretize the definition on polygonal meshes?

Discrete Normals



Discrete Normals

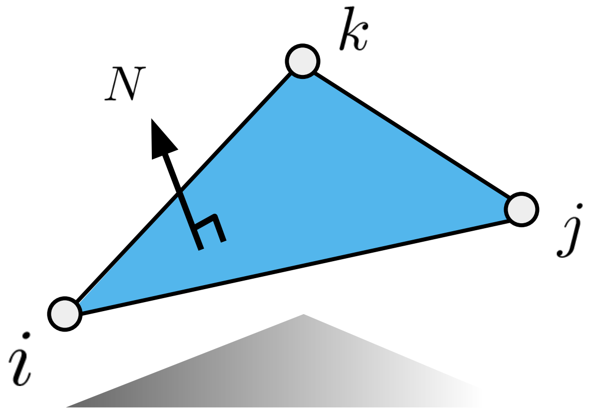




Face Normal

Face normals are well-defined:

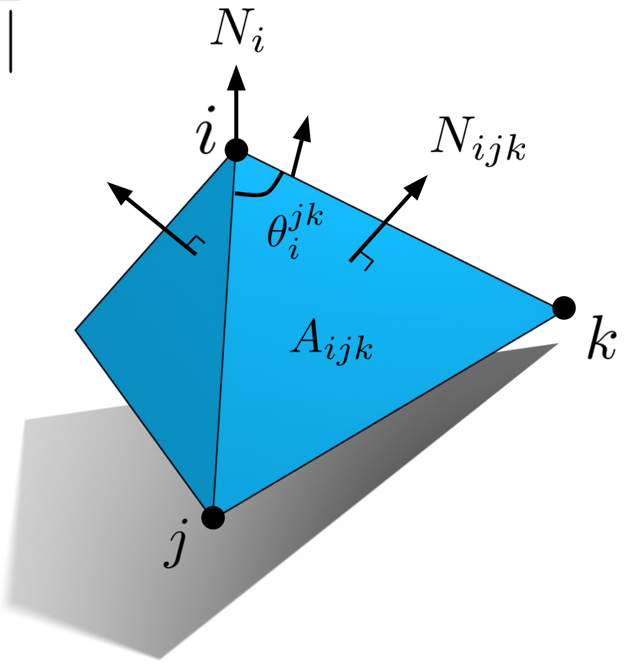
$$N = \frac{(f_j - f_i) \times (f_k - f_i)}{||(f_j - f_i) \times (f_k - f_i)||}$$



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 $w_{ijk} = 1$

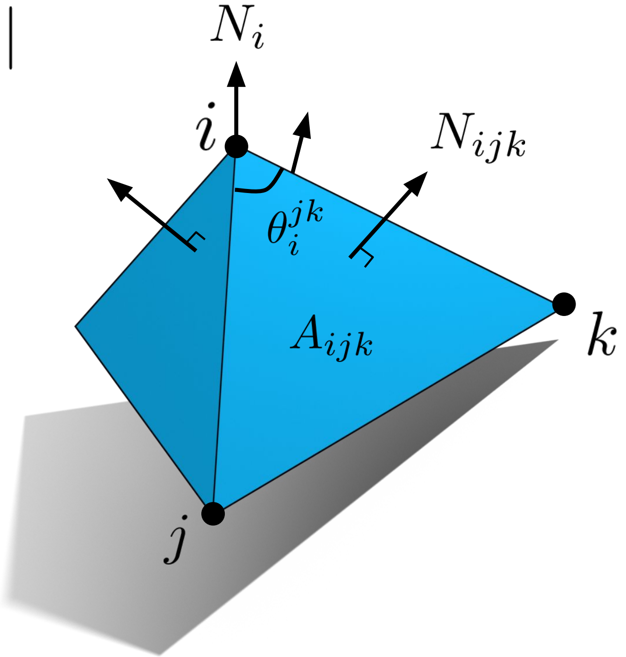
Vertex Normal

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:

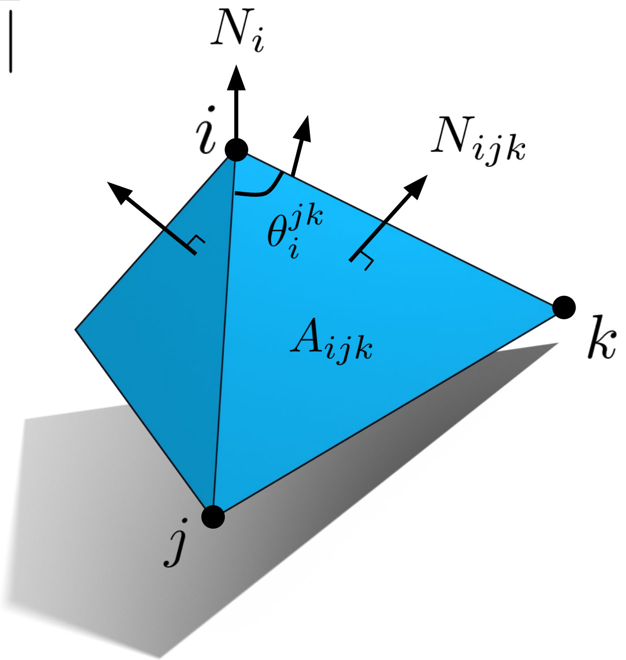
- Uniform (or Equally) Weighted $w_{ijk} = 1$
- Area Weighted $w_{ijk} = A_{ijk}$



Vertex Normal

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

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



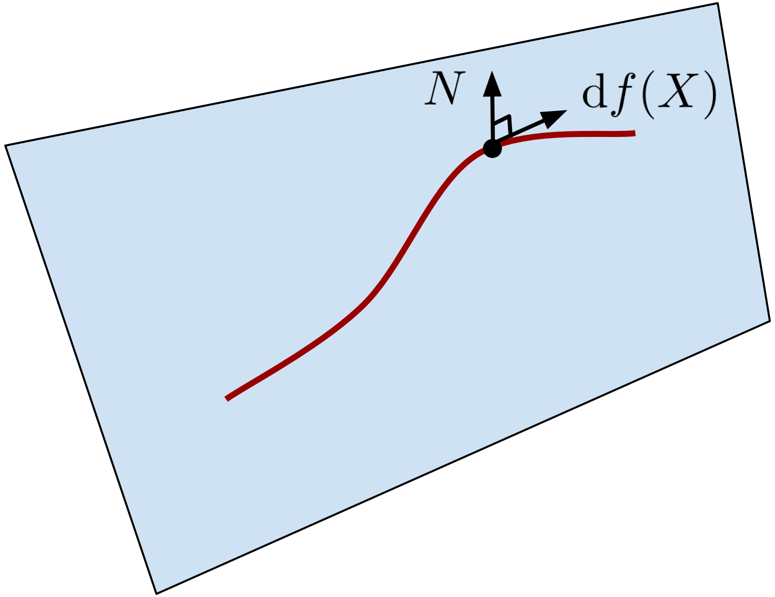
Variances:

- 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

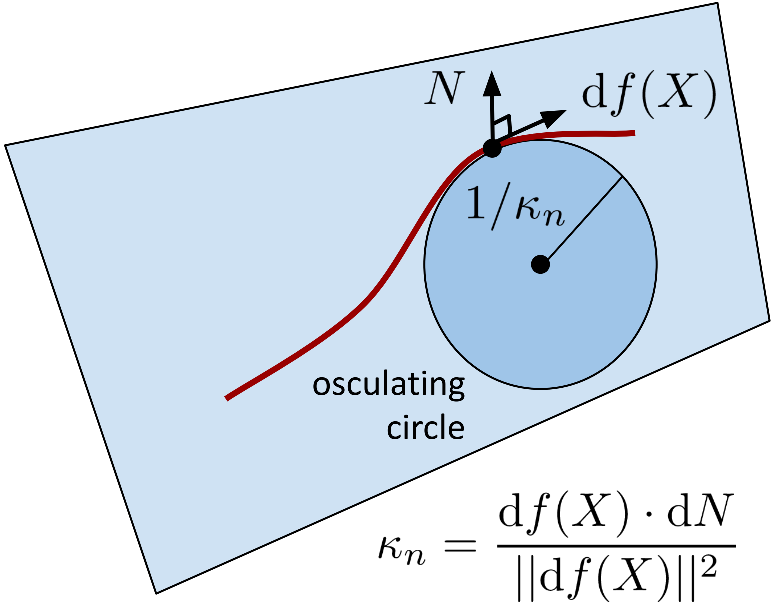
Curvature

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



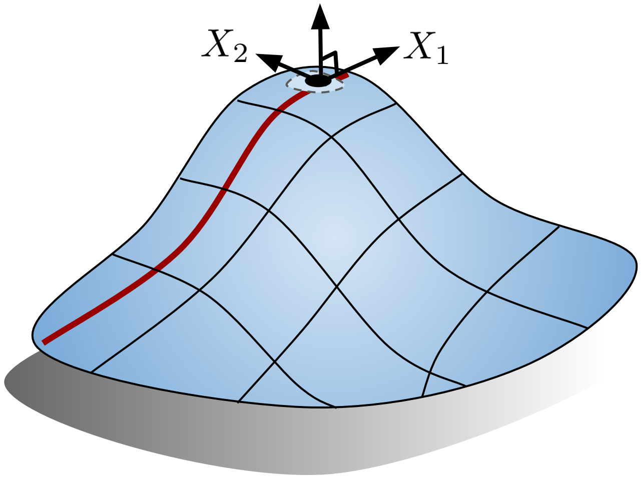
Normal Curvature κ_n

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



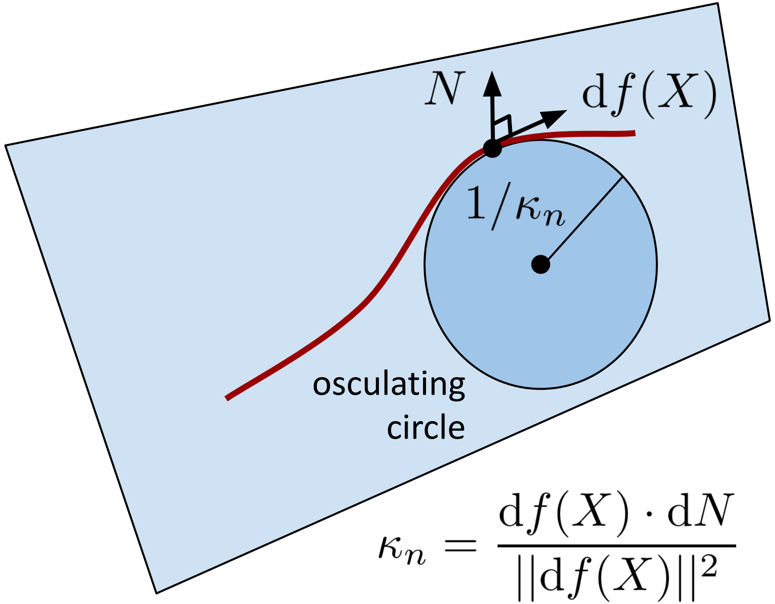
$$\kappa_n = \frac{\mathrm{d}f(X) \cdot \mathrm{d}N}{||\mathrm{d}f(X)||^2}$$

$$N = \frac{X_1 \times X_2}{||X_1 \times X_2||}$$



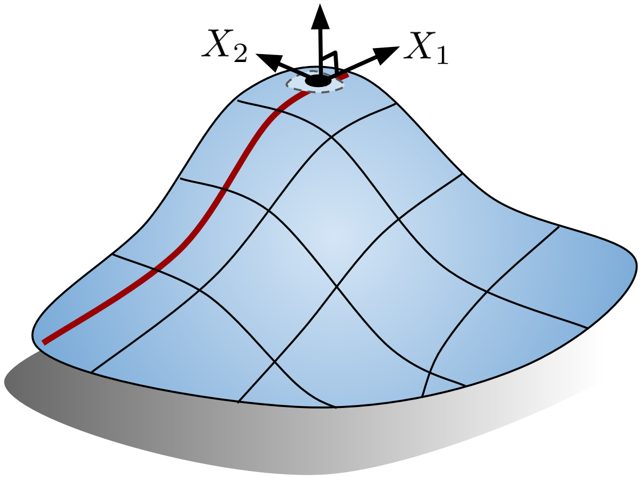
Normal Curvature κ_n

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



$$\kappa_n = \frac{\mathrm{d}f(X) \cdot \mathrm{d}N}{||\mathrm{d}f(X)||^2}$$

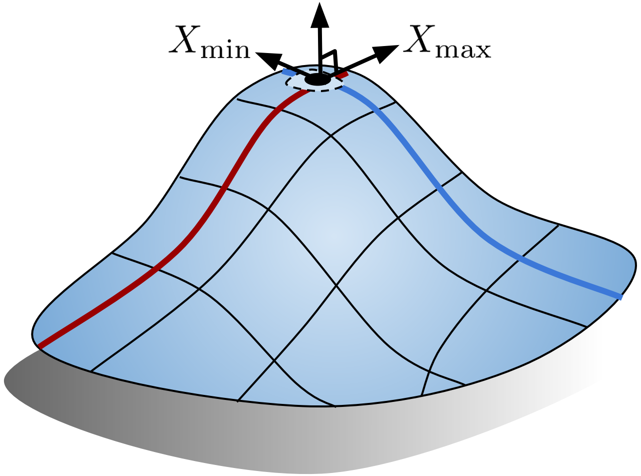
$$N = \frac{X_1 \times X_2}{||X_1 \times X_2||}$$



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

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

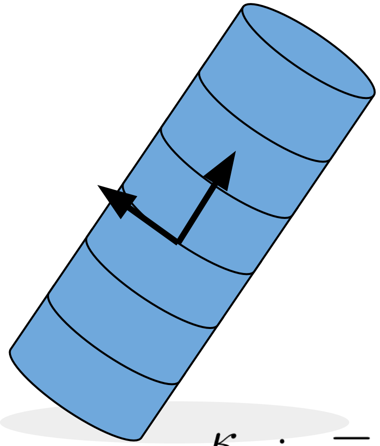
- **Principal directions:** Axes that describe the direction along which the normal changes the most/least
- **Principal curvatures:** along all directions, the two principal directions where normal curvature has minimum and maximum value respectively



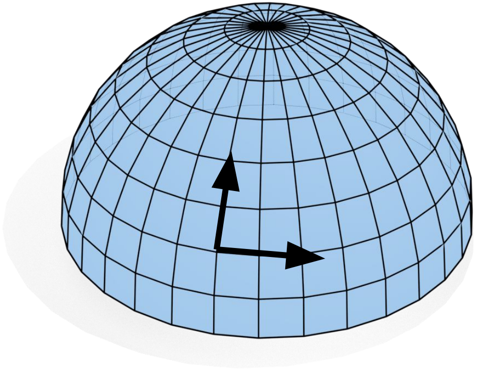
Some facts:

- (Euler's Theorem) principal directions are orthogonal
- $\mathrm{d}N = \kappa \mathrm{d}f(X)$

Principal Curvature: Examples



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



"Umbilic Points"

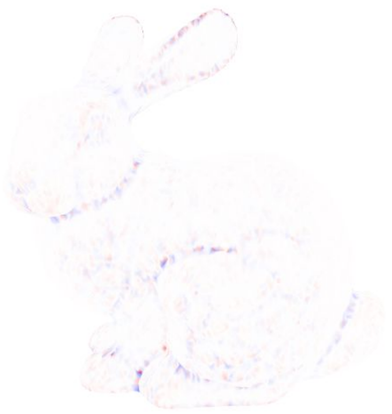
$$\kappa_{\min} = \kappa_{\max} = 1/r$$

Principal 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?

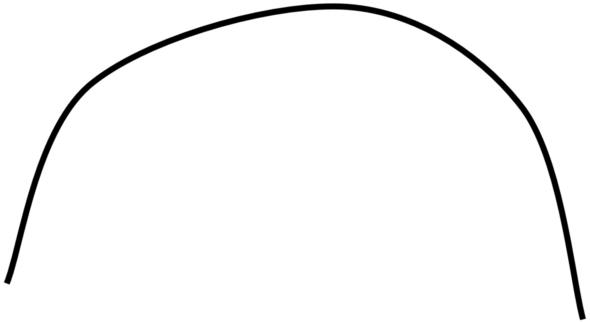
How do we actually compute curvatures in a discrete world?

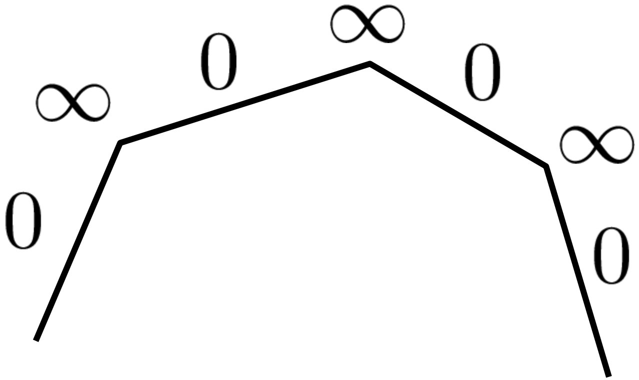
Discrete Curvature

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

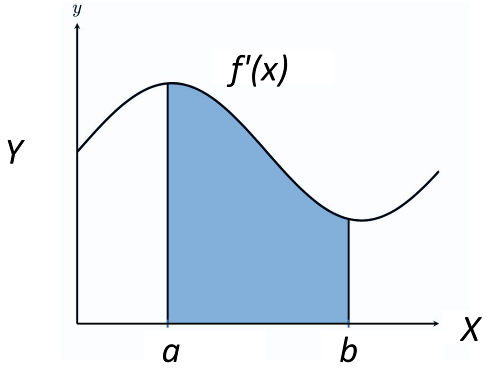
In discrete settings: No change along each edge ⇒ zero curvature?

$$\mathrm{d}N = \kappa \mathrm{d}f(X)$$



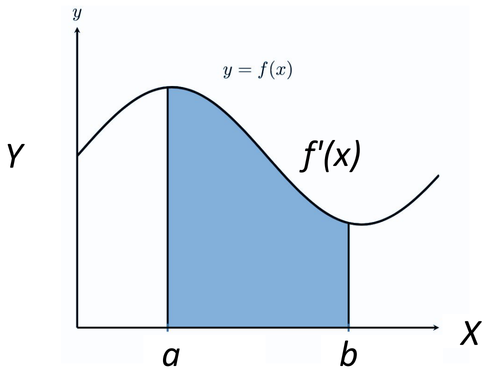


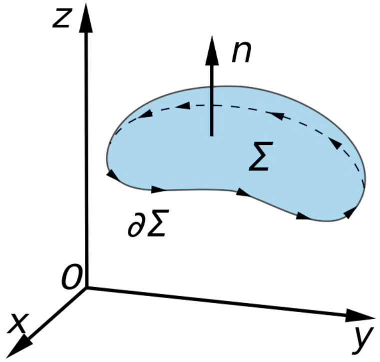
Revisit: Fundamental Theorem of Calculus



$$\int_a^b \mathrm{d}f = f(b) - f(a)$$

Revisit: Fundamental Theorem of Calculus and Stokes' Theorem





$$\int_a^b \mathrm{d}f = f(b) - f(a)$$

Insights from Stokes' Theorem

The change we see on the outside is

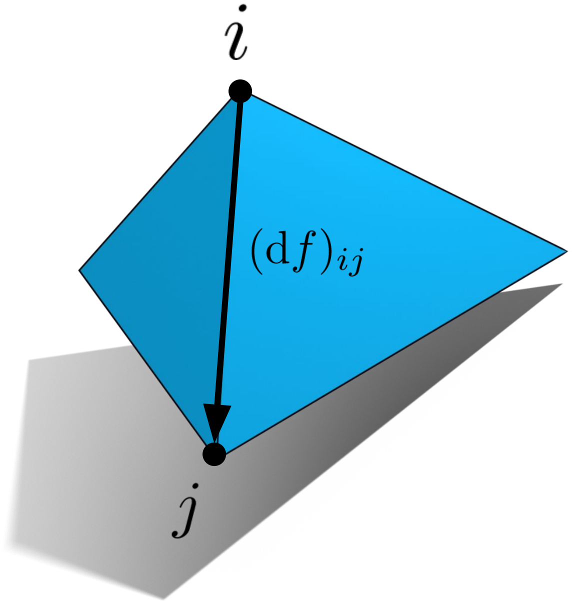
purely a function of the change within.

Example: Discrete *Differential*

Discrete differential *is just edge vectors in discrete settings*

$$(\mathrm{d}f)_{ij} = f_j - f_i$$

(Stokes' theorem)



Discrete *Principal Curvature*

$$K = \kappa_1 \kappa_2$$

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

$$\Rightarrow$$

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

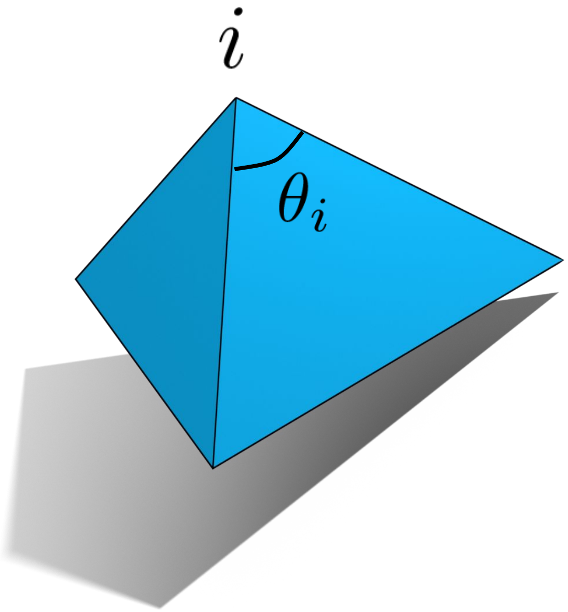
$$\kappa_2 = H + \sqrt{H^2 - K}$$

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

?

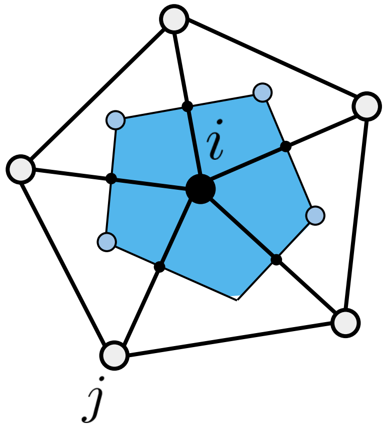
Laplacian

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(-Beltrami) 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

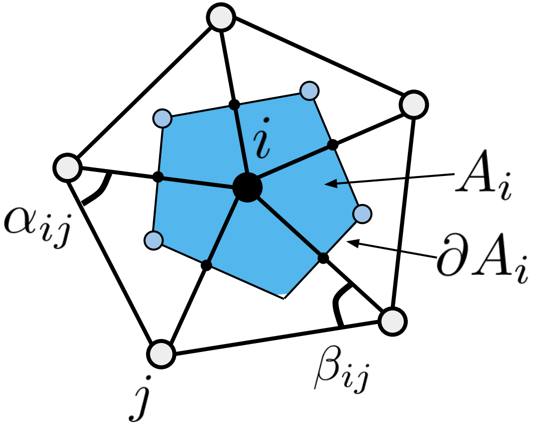
Basic idea: integrate the divergence of the gradient of a piecewise linear function over a *local averaging region* *(e.g. voronoi cell)*:

$$\int_{A_i} \Delta f \mathrm{d}A = \int_{\partial A_i} \nabla f \cdot N \mathrm{d}s$$

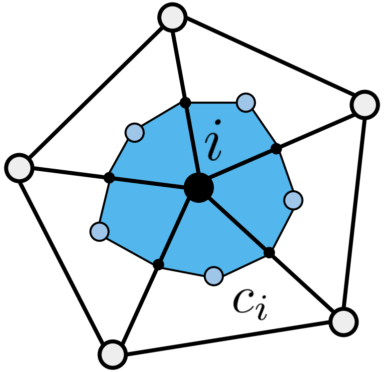
(Stokes' theorem)

One can prove:

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

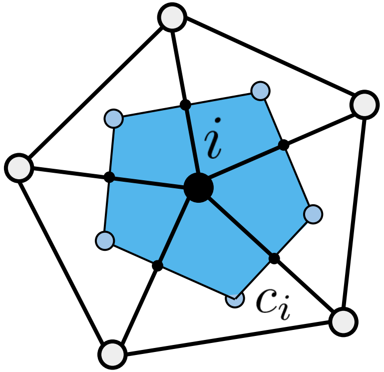


Local Averaging Region



Barycentric Cell

c_i = barycenter



Voronoi Cell

c_i = circumcenter

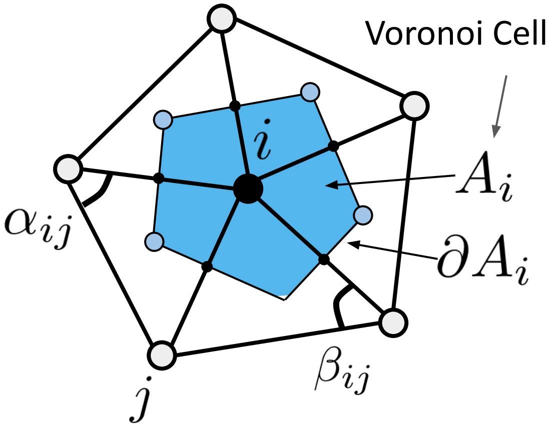
The *Laplace-Beltrami Operator*

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:

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

Implementation thinking: Is it necessary to keep the ½ factor?

Computing Discrete Curvatures

Mean curvature: via Laplace-Beltrami

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

Gaussian curvature: via angle defect

$$K = \Omega_i$$

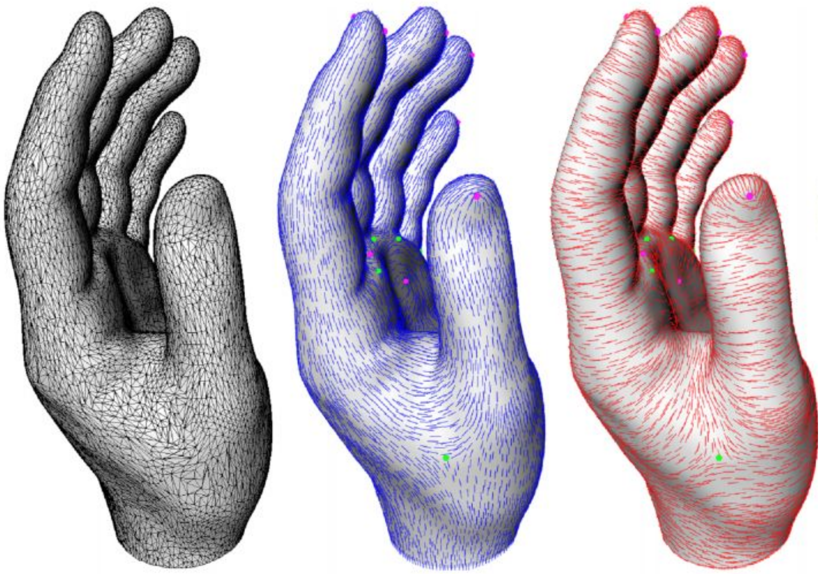
Principal curvature: via Gaussian and Mean

$$\kappa_1 = H - \sqrt{H^2 - K}$$
$$\kappa_2 = H + \sqrt{H^2 - K}$$

Application: Anisotropic Remeshing [Alliez et al. 2003]

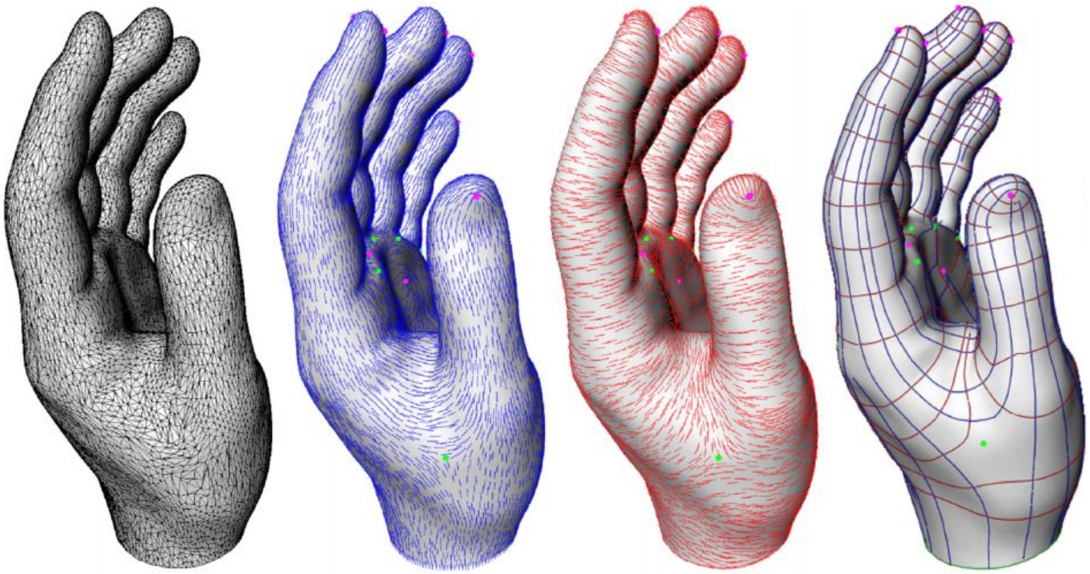
Input Mesh

Application: Anisotropic Remeshing [Alliez et al. 2003]



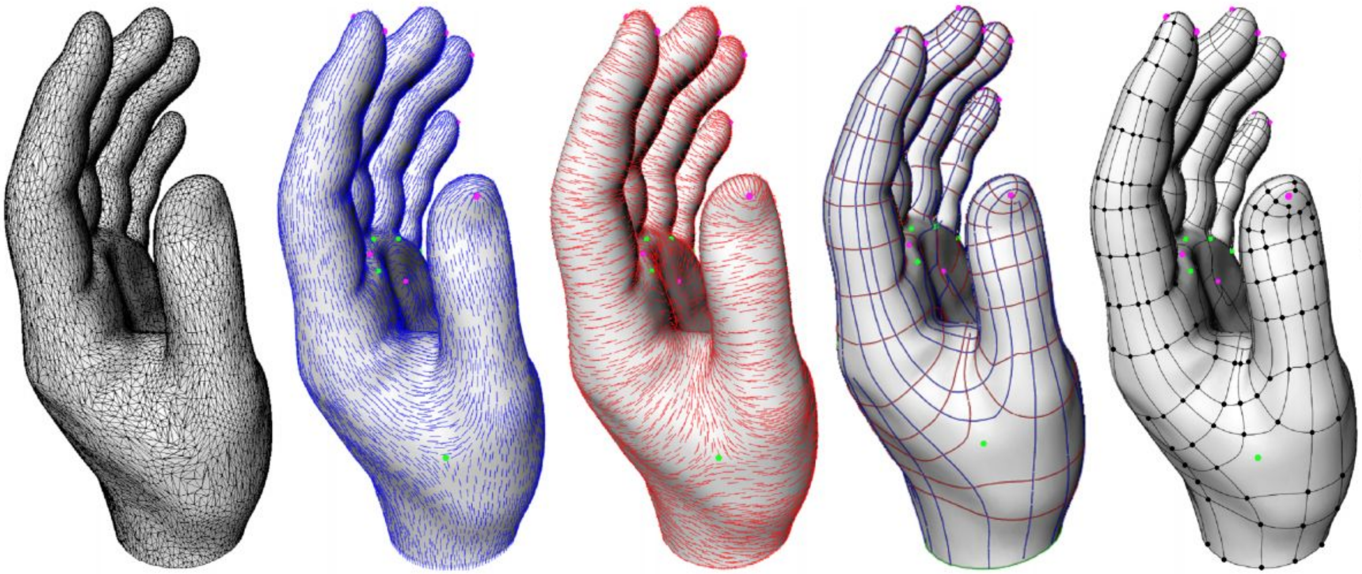
Input Mesh Kmin Kmax

Application: Anisotropic Remeshing [Alliez et al. 2003]



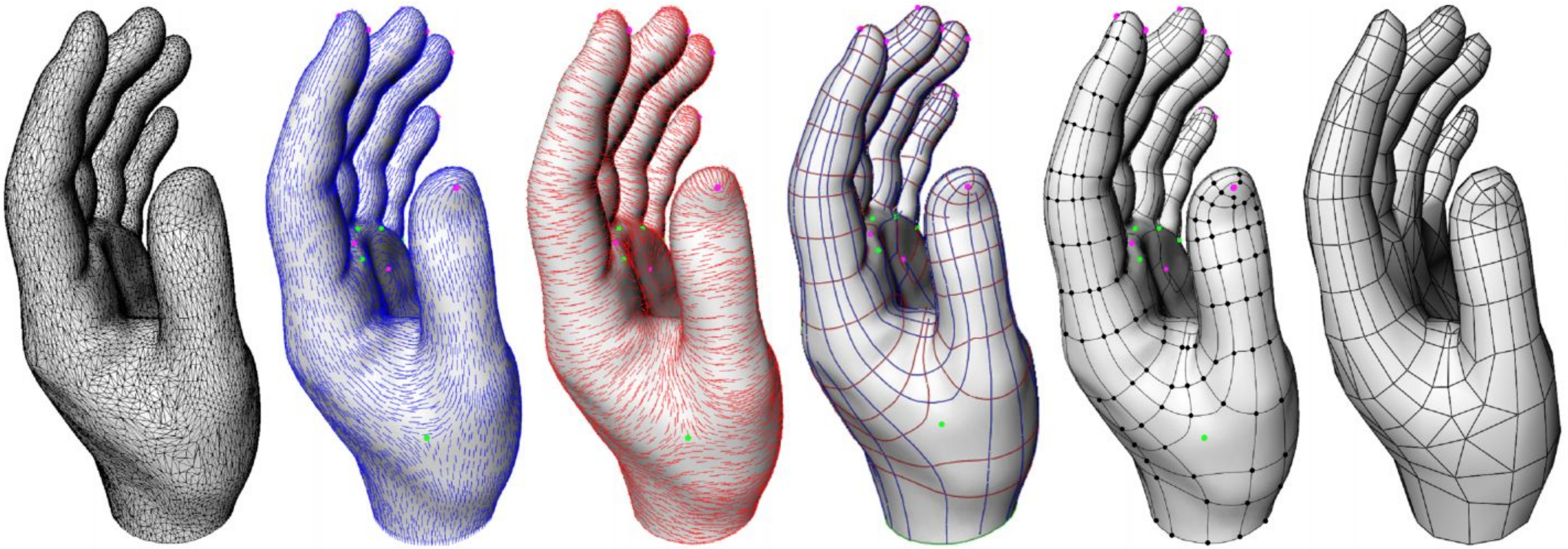
Input Mesh Kmin Kmax Sampling

Application: Anisotropic Remeshing [Alliez et al. 2003]



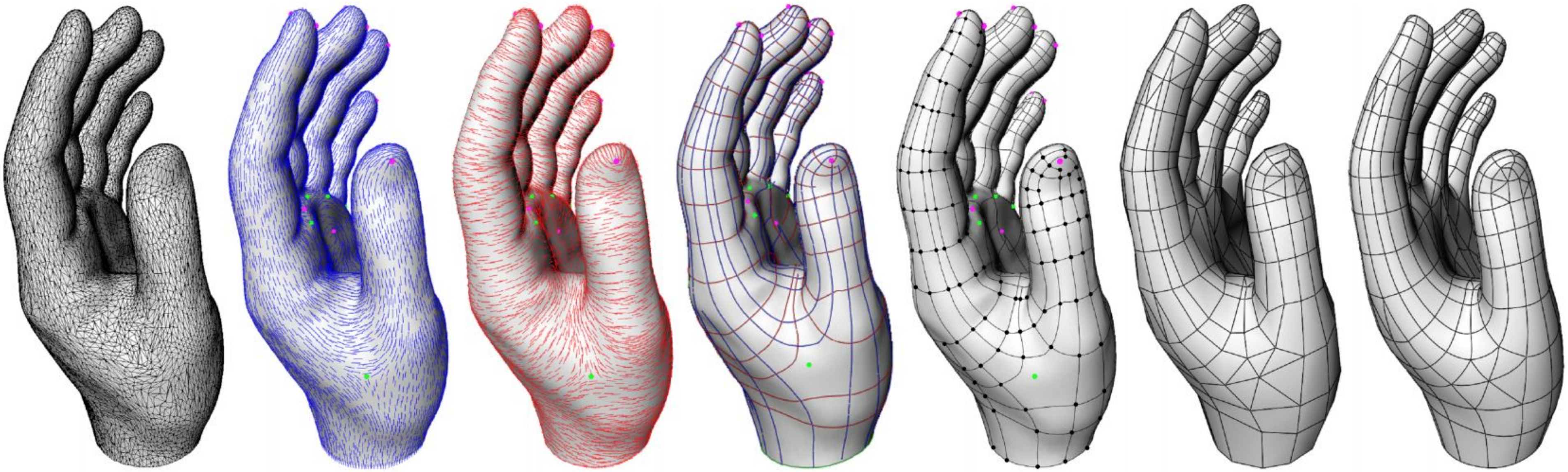
Input Mesh Kmin Kmax Sampling Meshing

Application: Anisotropic Remeshing [Alliez et al. 2003]



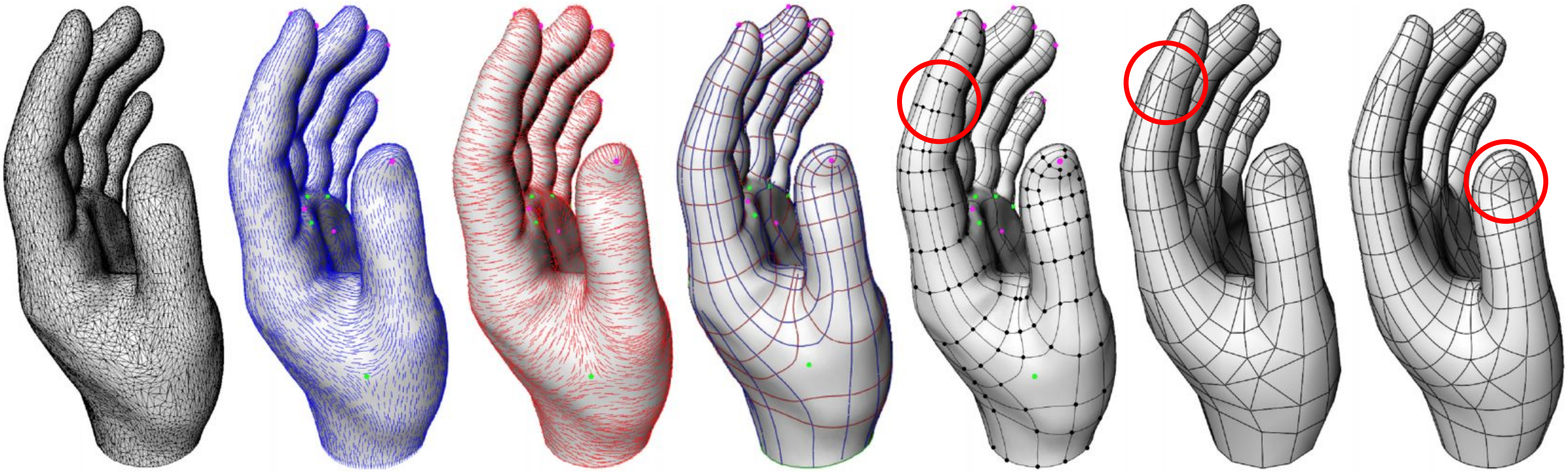
Input Mesh Kmin Kmax Sampling Meshing Output Mesh

Application: Anisotropic Remeshing [Alliez et al. 2003]



Input Mesh | Kmin | Kmax | Sampling | Meshing | Output Mesh | *Smoothing*

Application: Anisotropic Remeshing [Alliez et al. 2003]



Input Mesh | Kmin | Kmax | Sampling | Meshing | Output Mesh | *Smoothing*

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

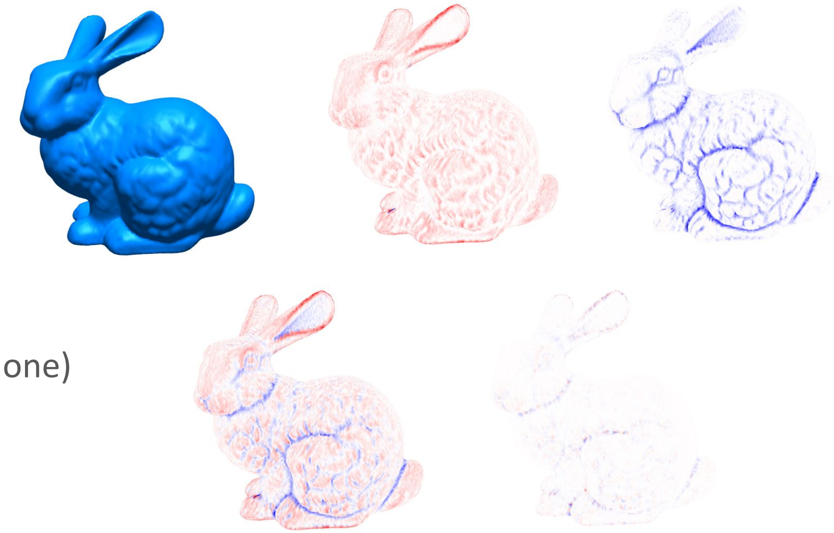
Session 2: Discrete Differential Geometry

- Motivation
- Discrete Geometric Quantifies
    - Normals
    - Curvature
    - Laplace-Beltrami
- Summary
- Discussion: Homework 1
    - .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 you should apply a certain definition in a context

Homework 2: Visualizing Normal and Curvature



Visualize the following two geometric quantities:

1. Compute and visualize the three different weighted normals
   - Equal weighted
   - Area weighted
   - Angle weighted
2. Compute and visualize the three different curvatures (actually just one)
   - Principal curvature
   - Mean curvature
   - Gaussian curvature

More details: https://github.com/mimuc/gp/blob/ws2021/homeworks/2-dgg

Discussion panel: https://github.com/mimuc/gp/issues/2

Submission Instructions: https://github.com/mimuc/gp/tree/master/homeworks#submission-instruction

Thanks! What are your questions?

Next session: Smoothing

Break

We will return at 16:15

Session 2: Discrete Differential Geometry

- Motivation
- Discrete Geometric Quantifies
  - Normals
  - Curvature
  - Laplace-Beltrami
- Summary
- Discussion: Homework 1
  - .OBJ Mesh Loader
  - Blender Python APIs
  - Blender BMesh Structure

Implementing A Naive .OBJ Mesh Loader

Questions

- What information must be loaded for a minimum implementation?
- How data structures are declared?
- How's the performance?
- What would change if we further do it in halfedge representation?

.OBJ File Format Specification

The most important fields for the homework:

`v  x y z w`	**Vertex position**
`vn i j k`	**Vertex normal**
`vt u v`	**Vertex texture coordinates**
`f  v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3`	**Relevant indices**

Blender + Python

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

Key Concepts

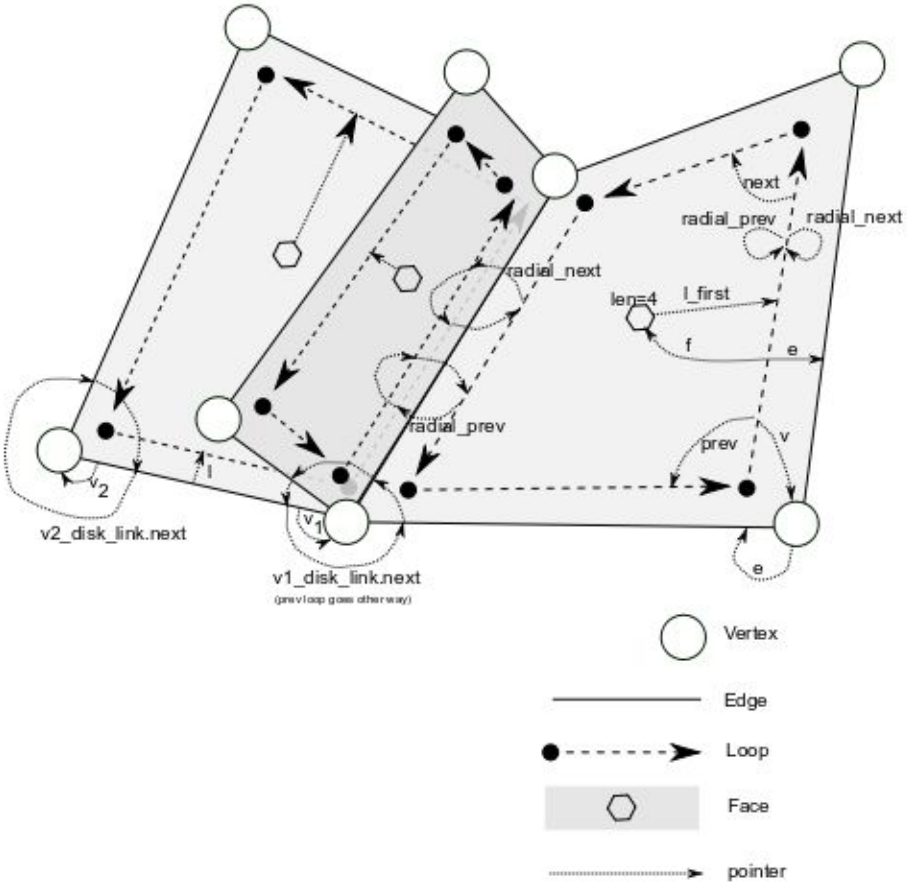
Types: bpy.types

Data: bpy.data

Operator: bpy.ops

Context: bpy.context

BMesh: A Non-Manifold Boundary Representation



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

Blender's Mesh Editing APIs

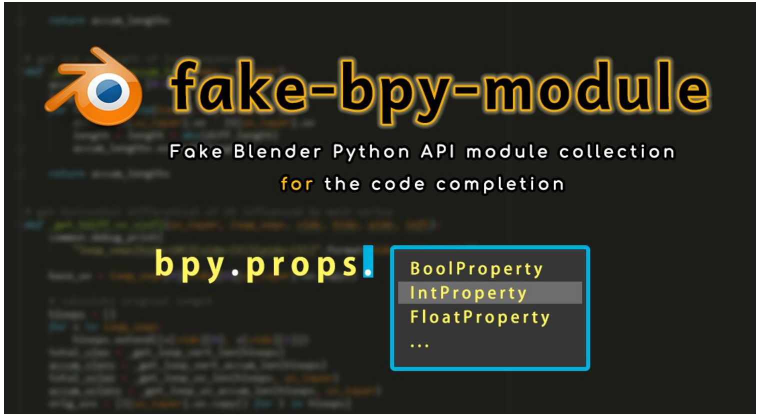
Low-level Operators

Mid-level Operators

Top-level Operators

Tip: Code Completion

```
pip install fake-bpy-module-2.90
```



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)

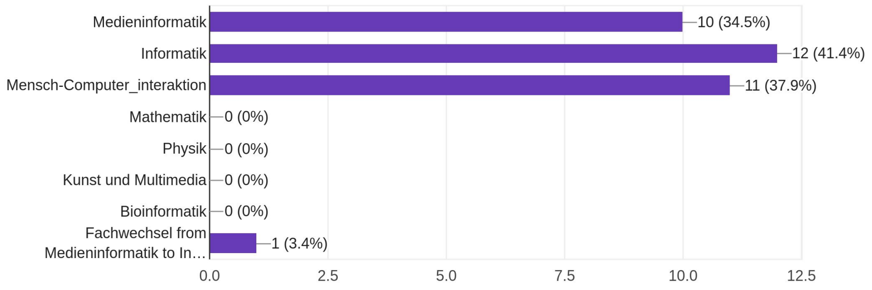
Jin S, Lewis RR, West D. A comparison of algorithms for vertex normal computation. The visual computer. 2005 Feb 1;21(1-2):71-82.

Wardetzky, Max, et al. Discrete Laplace operators: no free lunch. Symposium on Geometry processing. 2007.

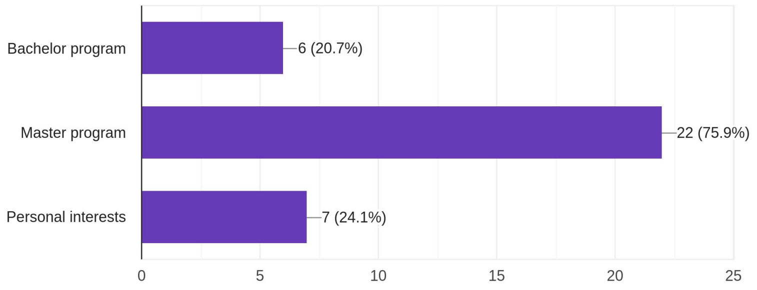
Keenan Crane. Discrete Differential Geometry: An Applied Introduction. 2020.

Pre-survey: Background

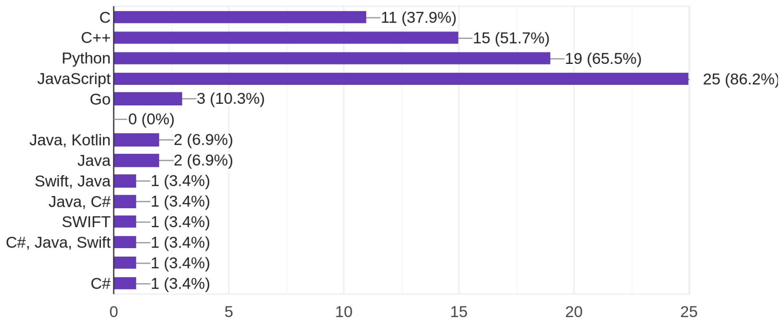
What's your major?
29 responses



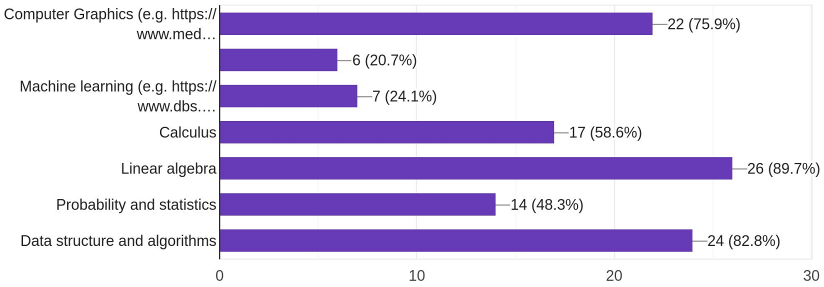
I enrolled this course for my...
29 responses



What programming languages you have already used before taking this course?
29 responses

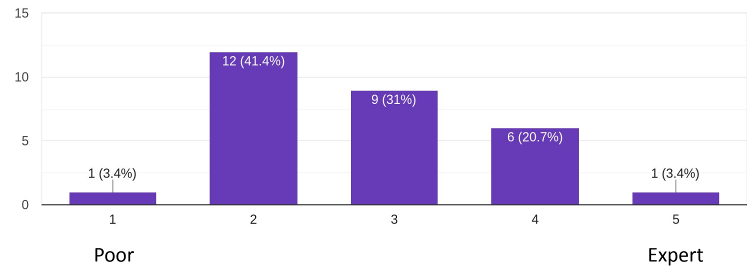


I learned these before taking this course:
29 responses



Pre-survey: Expectations

How do you think your knowledge level in computer graphics?
29 responses



In this course i expect a recap of important Computer Graphics Topics as well as an introduction to blender. After this i expect a deeper dive into more sophisticated 3D Modelling subjects.

lear more about computer graphics and data structures, how they can be handled more efficient and processed faster
I want to learn more about 3D Algorithms and how they are implemented in modern GPUs.

I expect to get more insight into the graphics design world and maybe come out of it with a more developed skill set which I'd like to build onto in my career.

I am interested in the difference of geometry, point clouds, meshes, nurbs models etc. and how they work

Deepen my knowlegde about CG techniques and algorithms.

Especially getting to know basic algorithms for mesh manipulation.

I would like to learn more about how 3D graphics generated

Programming skill

Deepen my experience with graphics and rendering in general; Possibly apply some self-tought knowledge and make the best out of it

I am quite interested in computer graphics and want to build some practical experience onto my theoretical knowledge. I used to 3d model (self taught) and I expect to fill some knowledge gaps and get a better understanding of the whole picture

I really enjoyed CG1 and I would love to extend on the concepts learned there and go into more detail

I'm very interested in Autonomous Systems with a focus on Robotics. I think learning more about computer graphics and geometric processing might be helpful for this.

This work is an example of where I see both areas profiting from each other and potential for new applications: https://github.com/autonomousvision/differentiable_volumetric_rendering

Personal interest

I just think it all sounds quite interesting and that it might be a fun lecture what with it being a practical course

I would like to learn more about geometry processing. I would also like to share my theoretical knowledge of computer graphics

Deepen my knowledge from cg1, experience in blender

I liked the computer graphics lecture and am interested in gaining more knowledge about this topic and am looking forward to the coding projects to improve my programming skills

learn more through doing

weil mich die Modellierung von 3d Modellen interessiert

I want to improve my 3D-Design skills

I would like to Refresh my theoretical knowledge from computer graphics by practical exercises, because I know I from myself i can learn more from doing specific tasks than read only the theory. And it is a long time ago that I attended computer graphics 1... Furthermore I would like to learn to work with Blender.

to learn an empirical example about the GC, to know how

Learning more about the technical side. Algorithms, etc

I am interested in the Geometry Processing which related to AI and Graphics. I hope i can learn something interesting during this practice and somehow improve my hand-on ability.

I want to obtain an extensive knowledge of computer graphics and improve my coding skills in that field

not too much