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."

\[ \alpha + \beta = \pi \]

Geometry principles works differently on curved spaces...

Euclidean, Elliptic, Hyperbolic

Euclid
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

\[\hat{\theta}_j = \theta_j \cdot \frac{2\pi}{\sum_i \theta_i}\]

\[
\Rightarrow \sum_j \hat{\theta}_j = \sum_j \theta_j \cdot \frac{2\pi}{\sum_i \theta_i} = \frac{2\pi}{\sum_i \theta_i} \sum_j \theta_j = 2\pi
\]

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 \( 2\pi \):

\[
\Omega_i = 2\pi - \sum \theta_i
\]
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 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

\[ N_i = \frac{\sum_i w_{ijk} N_{ijk}}{\left\| \sum_i w_{ijk} N_{ijk} \right\|} \]

Variances:
- Uniform (or Equally) Weighted \( w_{ijk} = 1 \)
**Vertex Normal**

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

\[ N_i = \frac{\sum_i w_{ijk} N_{ijk}}{\left| \sum_i w_{ijk} N_{ijk} \right|} \]

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

\[ N_i = \frac{\sum w_{ijk} N_{ijk}}{\left\| \sum w_{ijk} N_{ijk} \right\|} \]

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

$$\kappa_n = \frac{df(X) \cdot dN}{||df(X)||^2}$$

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

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

$\kappa_n = \frac{df(X) \cdot dN}{||df(X)||^2}$
**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

- \( dN = \kappa df(X) \)
Principal Curvature: Examples

\[ \kappa_{\text{min}} = 0, \kappa_{\text{max}} = \frac{1}{r} \]

\[ \kappa_{\text{min}} = \kappa_{\text{max}} = \frac{1}{r} \]

"Umbilic Points"
Principtal Curvature: Visualized

$\kappa_{\text{min}}$

$\kappa_{\text{max}}$
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 $\Rightarrow$ 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

\[ K = \Omega_i = 2\pi - \sum \theta_i \]
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):

\[ \int_{A_i} \Delta f \, dA = \int_{\partial A_i} \nabla f \cdot N \, ds \]

(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
\[ C_i = \text{barycenter} \]

Voronoi Cell
\[ C_i = \text{circumcenter} \]
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 $\frac{1}{2}$ 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]
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 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

![BMesh Diagram](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. [link](http://paulbourke.net/dataformats/)

Further Readings (Discrete Differential Geometry)


Pre-survey: Background

What's your major?
29 responses

- Medieninformatik: 10 (34.5%)
- Informatik: 12 (41.4%)
- Mensch-Computer Interaktion: -11 (37.9%)
- Mathematik: 0 (0%)
- Physik: 0 (0%)
- Kunst und Multimedia: 0 (0%)
- Bioinformatik: 0 (0%)
- Fachwechsel from Medieninformatik to In...: -1 (3.4%)

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

- C: -11 (37.9%)
- C++: -15 (51.7%)
- Python: 15 (51.7%)
- JavaScript: 19 (65.5%)
- Go: -3 (10.3%)
- Java, Kotlin: -2 (6.9%)
- Java: -2 (6.9%)
- Swift, Java: -1 (3.4%)
- Java, C#: -1 (3.4%)
- Swift: -1 (3.4%)
- C#, Java, Swift: -1 (3.4%)
- C#:

I enrolled this course for my...: 29 responses

- Bachelor program: 6 (20.7%)
- Master program: -22 (75.9%)
- Personal interests: 7 (24.1%)

I learned these before taking this course:
29 responses

- Computer Graphics (e.g. https://www.med...): 22 (75.9%)
- Machine learning (e.g. https://www.dbis...): -7 (24.1%)
- Calculus: -17 (56.6%)
- Linear algebra: -26 (89.7%)
- Probability and statistics: -14 (48.3%)
- Data structure and algorithms: -24 (82.8%)
Pre-survey: Expectations

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.

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 would 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 knowledge 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-taught 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 some of 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.

Well, I’m interested in the 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 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.