Discrete Differential Geometry and It's Applications

Dissertation submitted in partial fulfilment

for the Award of

M.Sc **in** MATHEMATICS AND COMPUTING

by

AJEET KUMAR

under supervision of

PROF. BANKTESHWAR TIWARI

DST-CENTRE FOR INTERDISPLINARY MATHEMATICAL SCIENCES

INSTITUTE OF SCIENCE (BANARAS HINDU UNIVERSITY)

VARANASI – 221 005

ROLL NUMBER
23419MAC003

YEAR OF SUBMISSION
2025

CERTIFICATE

It is certified that the work contained in the dissertation titled **Discrete Differential Geometry and It's Applications** by **Ajeet Kumar** has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

It is further certified that the student has fulfilled all the requirements of Comprehensive Examination, Candidacy for the award of **M.Sc**.

Supervisor
Prof. Bankteshwar Tiwari
DST-Centre for Interdisplinary Mathematical Sciences
Institute of Science (Banaras Hindu University)
Varanasi – 221 005

External Examiner

DECLARATION BY THE CANDIDATE

I, **Ajeet Kumar**, certify that the work embodied in this dessertation is my own bonafide work and has been carried out under the supervision of **Prof. Bankteshwar Tiwari** from **January 2024** to **June 2025**, at the **DST-Centre for Interdisplinary Mathematical Sciences**, **Institute of Science(Banaras Hindu University), Varanasi**.

I further declare that this disseration has not been submitted, either in part or in full, for the award of any other degree/diploma. All sources of information and data have been duly acknowledged and cited wherever applicable. I affirm that I have not copied or included any material from other works such as journals, books, reports or websites without proper citation, nor have I presented such material as my own.

Date: July 12, 2025
Place: Varanasi, India

Enrollment Number : 471127
Signature of the Student

CERTIFICATE BY THE SUPERVISOR

It is certified that the above statement made by the student is correct to the best of my/our knowledge.

Supervisor
Prof. Bankteshwar Tiwari
DST-Centre for Interdisplinary Mathematical Sciences
Institute of Science (Banaras Hindu University)
Varanasi – 221 005

COPYRIGHT TRANSFER CERTIFICATE

Title of the Dissertation: Discrete Differential Geometry and It's Applications

Name of the Student: Ajeet Kumar

COPYRIGHT TRANSFER

The undersigned hereby assigns to the Institute of Science(Banaras Hindu University), Varanasi all rights under copyright that may exist in and for the above dissertation submitted for the award of the M.Sc in Mathematics and Computing.

Date: July 12, 2025
Place: Varanasi, India

Enrollment Number : 471127
Signature of the Student

Note: However, the author may reproduce or authorize others to reproduce material extracted verbatim from the dissertation or derivative of the dissertation for author's personal use provided that the source and the Institute's copyright notice are indicated.

ACKNOWLEDGEMENT

I would like to dedicate this dissertation to my loving parents . . .

Preface

Discrete differential geometry (DDG) represents a paradigm shift in computational geometry, emphasizing structure-preserving discretization of smooth geometric objects rather than mere equation approximations. Unlike traditional numerical methods that discretize differential or integral equations, DDG prioritizes the geometric essence by transforming curves into graphs, surfaces into triangle meshes, and embedding these discrete structures into space as piecewise-linear manifolds. This approach ensures that fundamental geometric properties, from curvature to topology, are faithfully preserved in computable frameworks.

This dissertation **Discrete Differential Geometry and Its Applications** explores discretization of smooth differential geometric objects such as curves, surfaces etc. to their discrete analog discrete curves, discrete surfaces and their intrinsic properties like curvature and topology. Central to this work is **Discrete Gauss-Bonnet Theorem** which bridges combinatorial surface and smooth geometry and **Discrete Laplacian-Beltrami Operator** a cornerstone for many applications in shape modeling, geometric processing and physical simulation in discrete surfaces.

My journey into the field of differential geometry began during undergraduate studies in **Applied Mathematics**, where I encountered the challenges related to the translation of smooth geometry of curves and surfaces into practically implementable algorithms. I explored further and got introduced with DDG's discrete Laplacian-Beltrami operator which played a pivotal role in geometric processing and analysis of the geometry of shapes.

I extend profound gratitude to my supervisor **Prof. Bankteshwar Tiwari** for their men-

torship in my master dissertation, who tested my learning from theoretical to practical results. A special acknowledgement goes to the open-source computational geometry cum discrete differential geometry community, whose library in python and C++ enables practical implementation faster.

Note :*The practical code, triangle mesh data and results are available at github repository link :* https://github.com/ajeetkbhardwaj/ddg-lab-for-master-thesis

Ajeet Kumar
Varanasi, June 2025

Contents

1 Introduction **1**

1.1 Motivation . . . 1

1.2 Geometry Acquisition Pipeline . . . 2

1.3 Pre-requisites . . . 4

1.3.1 Differential Geometry . . . 4

1.3.2 Combinatorial Surfaces . . . 6

1.3.3 Discrete Exterior Calculus . . . 8

1.4 An overview . . . 9

2 Discrete Curves **10**

2.1 Polygonal/Discrete curve . . . 10

2.2 Discrete Differential, Tangent, Normal . . . 11

2.3 Regular Discrete Curve . . . 13

2.4 Discrete curvature . . . 14

2.4.1 Turning Angle . . . 15

2.4.2 Length Variation . . . 16

2.4.3 Osculating Circle . . . 17

2.5 Fundamental Theorem of Discrete Curves . . . 18

3 Discrete Surfaces **21**

3.1 Discrete 2D Manifolds . . . 22

3.2 Topology of Discrete Manifolds . . . 22

3.3 Regularity of Discrete Surface 23
3.4 Differential, Tangent and Normal of Discrete Surface 24
3.5 Geometric Measurement on Triangle Mesh 29
3.6 Discrete Curvature 30
3.6.1 Discrete Gauss-Bonnet Theorem 30
3.6.2 Laplacian Operator 37

4 Applications in Geometric Processing **39**
4.1 Smoothing 39
4.1.1 Discrete Curve Smoothing 39
4.1.2 Discrete Surface Smoothing 43
4.2 Simulation of Heat Equation on Mesh 50
4.3 Geodesic Computation on Discrete Surface 52
4.3.1 Approximate Geodesic Computation 52
4.3.2 Heat Method for Geodesic Computation 53

5 Conclusion **56**
5.1 Summary of Contributions 56
5.2 Future Research Direction 57

List of Figures

1.1 Left : Principle of laser triangulation, Right : LiDAR scanner https://en.wikipedia.org/wiki/Lidar . . . 3

1.2 Half edge data structure . . . 7

1.3 OFF or OBJ file format . . . 7

2.1 Discrete curve [5] . . . 11

2.2 Discrete differential [5] . . . 12

2.3 Discrete tangent[5] . . . 12

2.4 Discrete normal, where $\Im$ is the 90^o rotation operator[5] . . . 13

2.5 Discrete regular curve [5] . . . 14

2.6 Fundamental theorem for plane curves . . . 20

2.7 Fundamental theorem for space curves . . . 20

3.1 Left : Manifold mesh, Right : Non-manifold mesh [5] . . . 22

3.2 Left : Discrete differential, Right : Discrete normal [5] . . . 24

3.3 Discrete Gauss map [5] . . . 31

3.4 Left : Angle deflect and discrete gaussian curvature, Right : discrete geodesic curvature [5] . . . 32

3.5 Gaussian-curvature . . . 33

3.6 Mean-curvature . . . 35

3.7 Min-curvature . . . 36

3.8 Max-curvature . . . 36

4.1 Finite difference approximation of laplacian of curve 40
4.2 Smoothing of plane curve 41
4.3 Smoothing of the space curve(helix) 42
4.4 Laplacian smoothing by explicit method 44
4.5 Laplacian smoothing by implcit method 44
4.6 Eigenfunction of laplacian 46
4.7 Energy optimization method for surface smoothing 49
4.8 Simulation of heat equation for f(x) = sin(x) 51
4.9 Simulation for $f(x) = \delta(x)$ 51
4.10 Simulation of heat equation on bunny mesh 52
4.11 Front : Fast marching algorithm 54
4.12 Back: Dijkstra's shortest path algorithm 54
4.13 Front : Geodesic 54
4.14 Back: Geodesic 54

Chapter 1

Introduction

In today's world, most of the manufactured objects and digital movie scenes from mechanical parts to artistic creations are first designed on computers via specialized softwares. Where, discrete differential geometry(DDG) provides mathematical backbone(foundation) from shape modelling to process them. We know that many industries like gaming and entertainment, textile fashion, transportation and logistics, automotive defense, healthcare and life sciences, construction and infrastructure have transformed due to 3D shape modeling and processing and now they mostly rely on the digital 3D shape creation therefore, more accurate and efficient geometric tools and algorithms become critical.

1.1 Motivation

What is the goal of discrete differential geometry ? The goal of DDG is to learn how to design, program and analyse algorithms that enables interactive 3D shape modelling and digital geometric processing, which bridges that gap between theoretical and practical for working with shapes. DDG provides a foundation to understand the theory and applications of the 2D and 3D shape processing and use algorithms for modelling and manipulating to accurately describe, analyse and transform real-life or imagine objects into digital shapes on a computer with their 2D/3D geometric structures. Thus, ultimate DDG enables us to build robust, efficient and expressive tools for applications ranging

from computer graphics and animation to scientific computing, simulations and digital fabrication.

What are the industry, where we use discrete differential geometry ? DDG has a wide range of applications across both research and industry, due to its ability to model, analyse and manipulate complex geometric structures efficiently. There are few key area where DDG are used as follows

1. Medicine and Prosthetics
2. Product Design and Prototyping
3. Architecture
4. Cultural Heritage
5. Digital Human and Avatars
6. Geographical Systems and Urban Planning
7. Manufacturing and Fabrication
8. Apparel and Fashion.

Thus, we can say that DDG plays a very important role in modern computational design where geometry determines the visual appearance and physical behaviour of products and structures.

1.2 Geometry Acquisition Pipeline

Geometry acquisition is the study of using techniques and technology to capture 3D shape and surface properties of real-world objects and environment. Today we require geometric data in autonomous driving, construction of industrial facilities and 3D reconstruction of the human body organism etc. We use the discrete differential geometry to process these geometric data for various applications like in autonomous cars, perceive their surroundings for obstacle detection, industrial facilities maintenance, planning and virtual tours

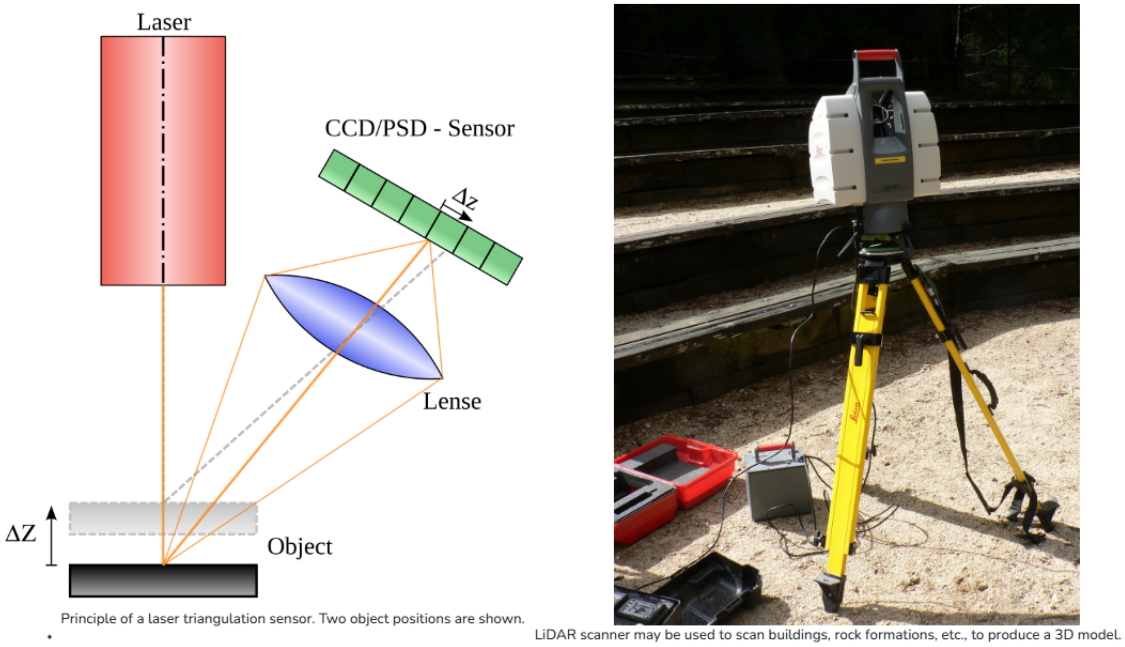
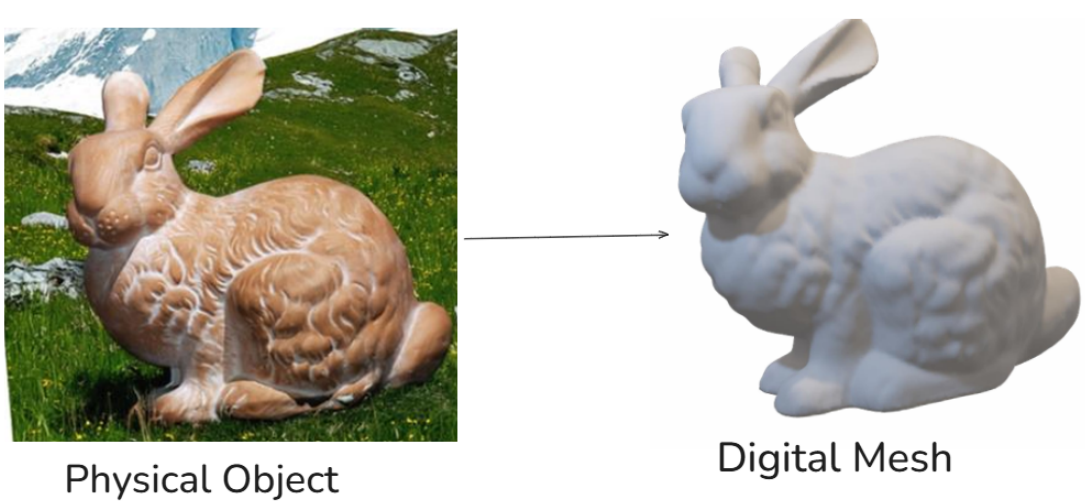


Figure 1.1: Left : Principle of laser triangulation, Right : LiDAR scanner https://en.wikipedia.org/wiki/Lidar

and healthcare for diagnostics, surgical planning and creating patient specific models etc.



So, we can say that geometry acquisition pipeline is a process such that a physical object is digitized through following steps

1. **Scanning** : A 3D scanner will capture shape and surface geometry of the physical object from multiple view points and we collect a set of range images, each image is a distance to object surface. So, geometry acquisition pipeline is a process such

that a physical object is digitized through following steps

2. **Registration** : We obtained scanned images from different angles and range images in their own local coordinate systems. Thus, the Registration aim is to bring all of these range image's local coordinates under one common coordinate system.

3. **Stitching/Reconstruction** : After registration of scanned images, we integrate these aligned scans into a single,coherent 3D mesh connecting the points from the registered range image to create a continuous surface representation of the 3D physical object to 3D digital mesh object.

4. **Post Processing** : We refine the generated digital 3D mesh by performing operation such as

   topological filtering, correct any issue related to the connectivity in the mesh such as hole, non-manifold edge etc.

   Geometric-filtering, smooth out noise in surface geometry of digital 3D mesh.

   Remeshing, create a more regular and structured mesh.

   Compression , reduction of final size of 3D model which preserves geometric details. and many more.

1.3 Pre-requisites

1.3.1 Differential Geometry

Differential geometry is a field of geometry that studies smooth-varying shapes called smooth manifolds such as smooth curves and surfaces are 1-dim and 2-dim smooth manifolds resp.

Definition 1.3.1: Parameterized Curves

Let $I \in R$ be an open interval then $\gamma : I \to R^n$ be a map from the interval to the euclidean space that is said to be a parameterized curve.

In particular, $\gamma : I \rightarrow R^2$ defined as $\gamma(t)$ =(x(t), y(t)) $\forall t \in I$ is a plane curve and $\gamma : I \rightarrow R^3$ defined as $\gamma(t)$ =(x(t), y(t), z(t)) $\forall t \in I$ is a space curve.

Definition 1.3.2: Smooth Curve

A parameterized curve $\gamma : I \rightarrow R^n$ is said to be regular(smooth) iff $d\gamma(t) \neq 0, \forall t \in I$ i.e velocity of the curve never vanishes at any point.

Definition 1.3.3: Parameterized Surface

Let $U \subseteq R^2$ be a subset and a diffeomorphism map $f : U \rightarrow R^n$ which takes region in 2-dim to the euclidean space such that image $f(U) \subseteq$ is called as parameterized surface.

Definition 1.3.4: Embeddings

A map f is an **embedding** if it is a homeomorphism onto its image, i.e., a continuous bijection with a continuous inverse.

Definition 1.3.5: Differential

Let $f : U \rightarrow R^n$ be a parameterized surface then **differential** $df_p : T_pU \rightarrow T_{f(p)}\mathbb{R}^n$ at a point $p \in U$ tells how tangent vectors in domain U are mapped(stretched out) into euclidean space $\mathbb{R}^n$[11].

Let a tangent vector $X \in T_pU$ then $df_p(X) = J_fX \in R^n$ where J_f is the jacobian of size $n \times 2$. Hence, we can say that df differential pushes forward, tangent vectors in planner region to the euclidean space.If we choose

$$f(u, v) = (x(u, v), y(u, v), z(u, v))$$

be a surface then

$$df = \begin{bmatrix} \frac{\partial f}{\partial u} & \frac{\partial f}{\partial v} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \\ \frac{\partial z}{\partial u} & \frac{\partial z}{\partial v} \end{bmatrix} = J_fX$$

J_f is the Jacobian matrix.

Definition 1.3.6: Immersion

A map $f : U \to R^n$ is an **immersion** if its differential $df_p : T_pU \to T_{f(p)}\mathbb{R}^n$ is injective at every point(non-degenerate) i.e for $X \in T_pU, df_p(X) = 0 \iff X_p = 0 \; \forall p \in U$

Immersions are locally embeddings but may have self-intersections globally. Thus, we can say immersion(regularity) of surfaces is necessary to define quantities like tangent, normal etc on surfaces.

1.3.2 Combinatorial Surfaces

Definition 1.3.7: Simplicial Surface

It is a simplicial 2-manifold which is a triangle mesh having regularity properties like topological connectivity & geometry such vertex coordinate describes simplicial immersions.

Definition 1.3.8: Abstract Simplicial Surface

It is a simplicial 2-complex, having

- Highest degree simplicies are triangles
- Every edge, glue two triangles together and on boundary, only one.

Definition 1.3.9: Simplicial map

A simplicial map is a map between simplicial complexes that sends simplices to simplices linearly based on their vertices $\sigma : K \to K', \quad$ where each simplex $\sigma \in K$ maps to a simplex $k(\sigma) \in K'$.

How to give shape to an Abstract Simplicial Surface ? To embed an abstract simplicial surface into $\mathbb{R}^3$, we assign coordinates to each vertex. This process is described as a 0-form on the simplicial complex. We define a map $f : V \to \mathbb{R}^3$ such that for each vertex $v_i \in V$, we have $f(v_i) = f_i \in \mathbb{R}^3$. Now, this also map edges using linear interpolation called 1-form such that $f : E \to \mathbb{R}^3, \quad$ for each $(v_i, v_j) \in E. \implies f((v_i, v_j)) = tf_i + (1-t)f_j, \quad$ where $t \in [0, 1]$. and further extend this to triangular faces using barycentric coordinates which is called 2-form $f : F \to \mathbb{R}^3, \quad$ for each triangle $\triangle = (v_0, v_1, v_2) \in$

$F, \implies f(\triangle) = t_0 f_0 + t_1 f_1 + t_2 f_2, \quad$ where $t_0 + t_1 + t_2 = 1$ and $t_i \geq 0$.

Half-edge data structure The *half-edge data structure* is a powerful way to store the polygonal meshes in the memory. It enables efficient traversal and manipulation of mesh connectivity, which is essential for algorithms that require access to neighboring elements. Every *half-edge* in the mesh is split into two oppositely oriented half edges. Each half-edge

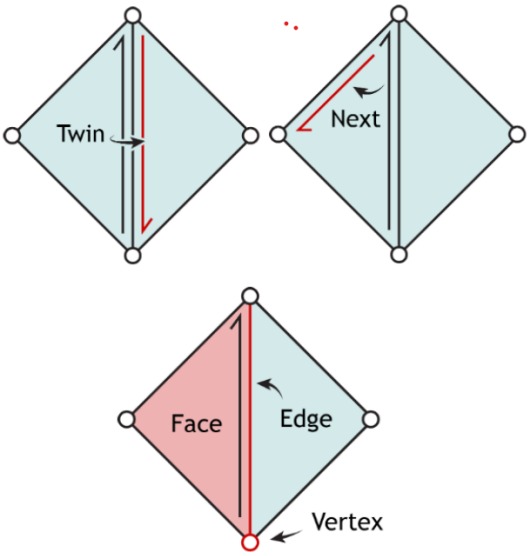


Figure 1.2: Half edge data structure

Vertices			
v0	x0	y0	z0
v1	x1	x1	z1
v2	x2	y2	z2
v3	x3	y3	z3
v4	x4	y4	z4
v5	x5	y5	z5
v6	x6	y6	z6
...	...	...	...

Triangles			
t0	v0	v1	v2
t1	v0	v1	v3
t2	v2	v4	v3
t3	v5	v2	v6
...	...	...	...

Figure 1.3: OFF or OBJ file format

stores pointers to its twin (the other half of the same edge, but in the opposite direction), its next half-edge in the current face (circulating counterclockwise), the vertex it points from, the edge it belongs to, and the face it borders. Each of these elements stores a pointer to one of its incident half-edges, enabling traversal from any mesh element to its neighbors.

So, our basic data structure for triangle mesh consists of 1. A list of vertex coordinates in $\mathbb{R}^3$ 2. A list of triangles, each defined by 3 ordered vertex indices $i, j, k \in \mathbb{N}$. The order of the vertices define the direction of the normal.

The ‘OFF‘ format, or Object File Format, stores this information in a ‘.off‘ file, which we will use for a triangle mesh.

1.3.3 Discrete Exterior Calculus

An exterior calculus is the generalization of the calculus concepts like gradient, curl and divergence on smooth manifolds using idea of differential forms and operators like functions, vector fields, area forms, volume forms and exterior derivative d, Hodge star $*$, wedge product resp. Vector calculus is the coordinate system-based language, used to describe the geometric quantities on smooth manifolds while exterior calculus is the coordinate-free language for the same. We will try to study the discretized concepts of exterior calculus on the discrete surfaces(triangular meshes) called discrete exterior calculus.

Discrete differential forms

A *discrete differential form* on a triangulated mesh $M = (V, E, F)$ is a function that assigns real values to k simplices (vertices, edges, or faces) as follows - 0-forms: Assign a value to each vertex V representing scalar fields on the mesh. - 1-forms: Assign a value to each edge E, representing integrated quantities like vector fields along edges. - 2-forms: Assign a value to each face F, representing area-related quantities, such as flux through faces.

Table 1.1: Smooth vs. Discrete Forms (with Hodge Star)

Degree	Smooth Form	Discrete Form	Mesh Element	Hodge Star (maps to)
0	Scalar field f	Value at vertices	Vertices (V)	Dual face (area)
1	Line integral ω	Value on edges	Edges (E)	Dual edge (length ratio)
2	Surface integral η	Value on faces	Faces (F)	Dual vertex (scalar)

Definition 1.3.10: Discrete exterior derivative

The *discrete exterior derivative* $\mathbf{d}_k$ maps discrete k forms to discrete $k+1$-forms. It is defined as the adjoint of the boundary operator, generalizing the notion of differentiation to the mesh setting:

$$\langle \mathbf{c}, \mathbf{d}_k \alpha \rangle = \langle \partial_{k+1} \mathbf{c}, \alpha \rangle$$

where α is a k-form.

The value of the derivative of α on a $(k+1)$-chain $\mathbf{c}$ equals the value of α on the boundary

of **c**. This construction ensures that key theorems like Stokes' theorem hold in the discrete setting.

Definition 1.3.11: Discrete Hodge Star

The *discrete Hodge star* is an operator that maps a k-form to an $(n-k)$-form on the dual mesh, incorporating the geometric information of the mesh.

For triangle meshes, the discrete Hodge star is typically defined using ratios of volumes (lengths, areas) between primal and dual mesh elements as follows

1. For a 0-form at a vertex, the Hodge star yields a value on the dual face (typically an area).
2. For a 1-form on an edge, it yields a value on the dual edge, often involving a length ratio.
3. For a 2-form on a face, it yields a value at the dual vertex(scalar)

1.4 An overview

In the next chapter we will explore the discrete curves in plane and space, discuss geometric quantities like discrete differential and discrete curvature and at the end fundamental theorem of discrete curves.

Then, move to the discrete surfaces, will cover discrete surface and it's regularity condition, discrete Gauss-Bonnet theorem, area weights and cotangent formula for computation of laplacian beltrami operator on triangle mesh.

The last will explore different applications on the laplacian beltrami operator from smoothening, physical simulation and geodesic computation on discrete surfaces.

Chapter 2

Discrete Curves

Discrete curve is the representation of a curve that has finitely many degrees of freedom like a polygonal curve, a spline curve, a subdivision curve, etc. Each of them has only a finite number of control points.

A smooth curve has geometric quantities which define the curve itself. Can we define these quantities for discrete curves ? such as the derivative of a curve at any point can be defined as a discrete differential and define different ways to compute the geometric invariant quantities for a discrete curve such as curvature and torsion based on the requirements.

In the end, try to give a discrete analogous to the fundamental theorem of plane and space curve and then based on these theorems design the algorithm to recover the discrete curve itself, given the curvature, torsion and initial vector.

2.1 Polygonal/Discrete curve

Definition 2.1.1: Discrete Plane Curve

A discrete plane curve, or more precisely a polygonal curve, is a map $\gamma : I \to R^2$ defined as $\gamma(s_i) = \gamma_i \in R^2 \; \forall s_i \in I$ where $I = \{s_0, s_1, \ldots, s_n\}$ such that every pair of point should have non-zero linear interpolation(line-segment) $|\gamma_{i+1} - \gamma_i| \neq 0$ and no 180^o folding back of $\gamma_{i+1} - \gamma_i \; \forall i$

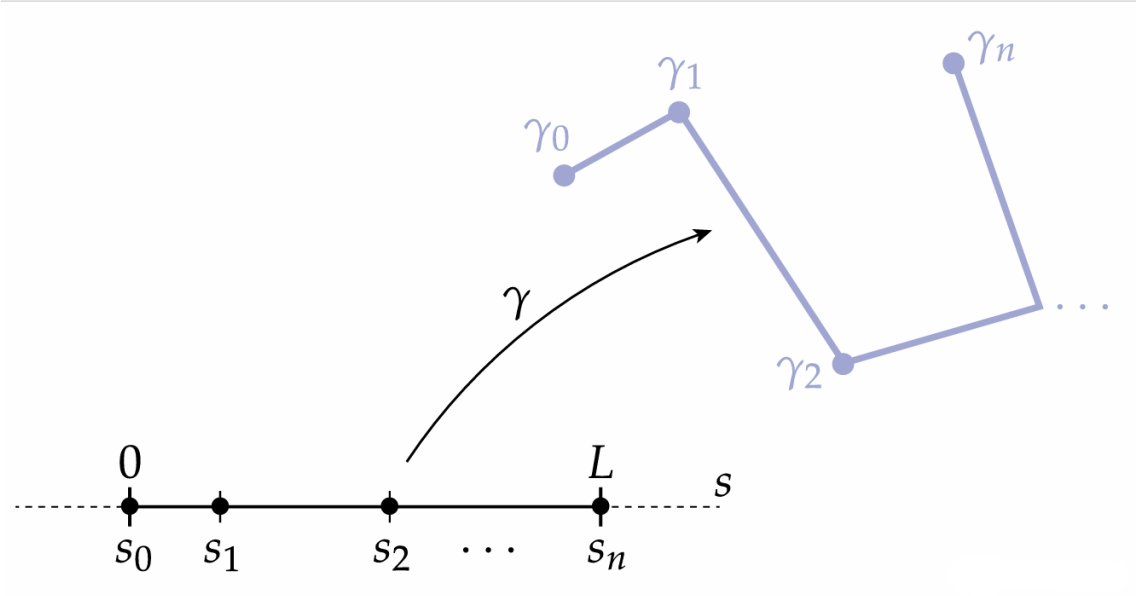


Figure 2.1: Discrete curve [5]

So, we can say that it is a simplicial 1-complex i.e. a connected graph $G = (V, E)$ having a set of vertices and edges such that $\gamma : V \to R^2$, it's values give the location of vertices in the plane and $\gamma : E \to R^2$ is the piecewise linear interpolation between any two ordered pair of vertices.

2.2 Discrete Differential, Tangent, Normal

Definition 2.2.1: Discrete Differential

Consider $\gamma : I \to R^2$ be a discrete plane curve, then the discrete differential will be equal to

$$(d\gamma)_{ij} = \gamma_j - \gamma_i$$

Remark 1 *Discrete differential is a finite difference between any pair of vertices that represents an edge vector.*

Definition 2.2.2: Length of a discrete curve

Let $\gamma : I \to R^2$ be a discrete plane curve then the length of γ over I will be equal to

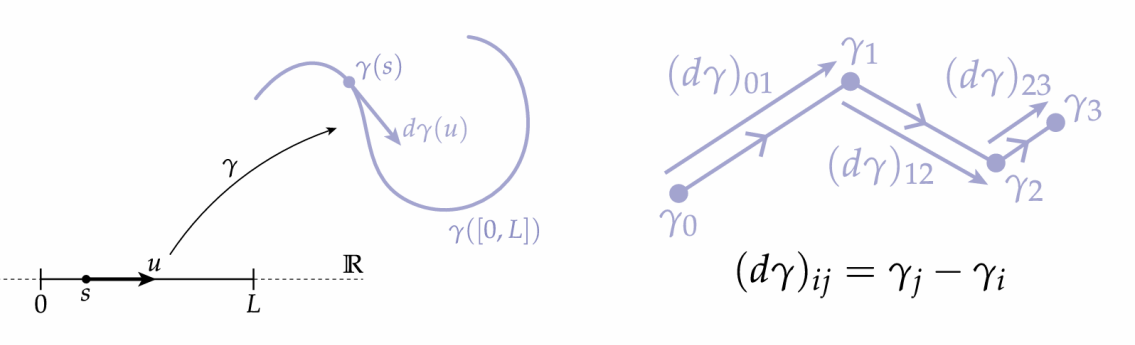


Figure 2.2: Discrete differential [5]

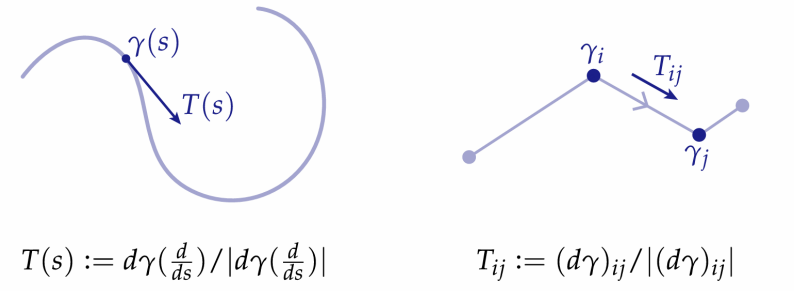


Figure 2.3: Discrete tangent[5]

$$L(\gamma) = \sum_{i=1}^{n} |\gamma_{i+1} - \gamma_i|$$

Remark 2 *Length of discrete plane curve is the sum of all edge lengths of the curve.*

Remark 3 *A discrete curve is said to be parameterised in arc length iff* $|(d\gamma)_{ij}| = 1 \; \forall i, j$

Definition 2.2.3: Discrete Tangent

A discrete tangent is a normalized discrete differential per edge, which means it is a unit-length edge vector. Thus, for a given discrete plane curve $\gamma : I \to R^2$, discrete tangent will be equal to

$$T_{ij} = \frac{(d\gamma)_{ij}}{|(d\gamma)_{ij}|} = \frac{\gamma_{i+1} - \gamma_i}{|\gamma_{i+1} - \gamma_i|} \; \forall \; i, j$$

Remark 4 *Discrete tangents are not defined at vertices, only defined at edges.*

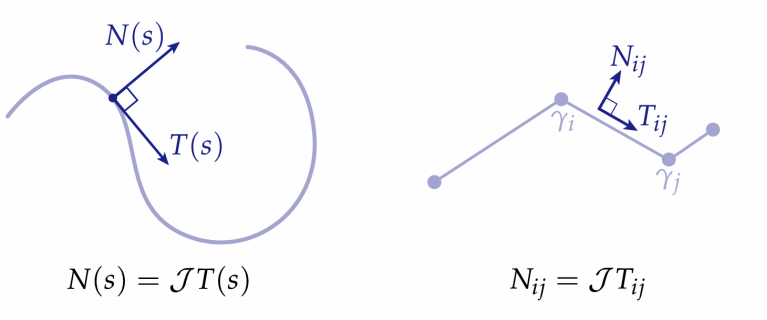


Figure 2.4: Discrete normal, where $\Im$ is the 90^o rotation operator[5]

Definition 2.2.4: Discrete Normal

Discrete normals are just the 90^o degree rotation of the discrete tangent vectors counterclockwise as $N_{ij} = R_{90^\circ} T_{ij}$, where $R_{90^\circ} = \begin{bmatrix} cos(90^\circ) & -sin(90^\circ) \\ sin(90^\circ) & cos(90^\circ) \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ be a rotation matrix that rotates the discrete tangent T_{ij} by 90°.

So, let $T_{ij} = \begin{bmatrix} T_x \\ T_y \end{bmatrix}$ then discrete normal will be $N_{ij} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} T_x \\ T_y \end{bmatrix} = \begin{bmatrix} -T_y \\ T_x \end{bmatrix}$

Remark 5 *The discrete normal is also a unit vector perpendicular to a discrete tangent vector.*

2.3 Regular Discrete Curve

Definition 2.3.1: Regular discrete curve

A discrete curve is said to be regular if

1. Non-zero differential (edge vector) $d\gamma_{ij} = \gamma_j - \gamma_i \neq 0$, preventing the curve from collapsing at any point.
2. No two consecutive edges are collinear and pointing in opposite directions, i.e., no zero turning angles $\frac{d\gamma_{ij}}{\|d\gamma_{ij}\|} \neq \pm \frac{d\gamma_{jk})}{\|d\gamma_{jk}\|}$ for adjacent edges e_{ij} and e_{jk}.

Remark : A discrete curve is regular if it is injective locally.

Why does regularity matter for the discrete curves ? We know that if a curve is smooth,

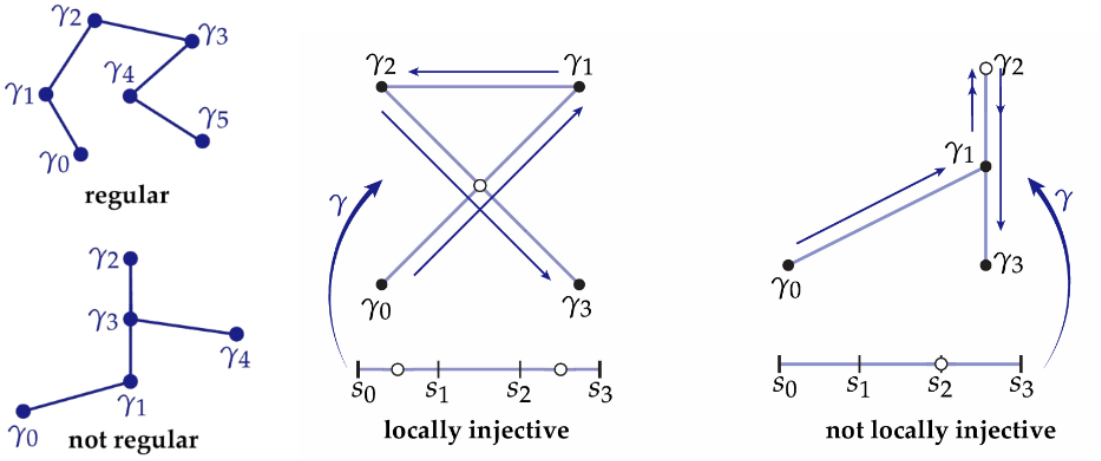


Figure 2.5: Discrete regular curve [5]

then we can easily compute the many geometric variants and invariant quantities which define the curve itself. Therefore, we need the regularity condition for the discrete curve to compute certain geometric quantities.

2.4 Discrete curvature

Definition 2.4.1: Discrete Curvature

It is the curvature of the discrete curve that measures the bending of the curve in plane as we move along the curve.

We know that tangents are well defined on a regular discrete curve over an edge vector(discrete differential); thus, discrete curvature measures how much the tangent vector has turned as we move along the curve.

There are many ways to define the discrete curvature based on the different properties of discrete curves, such as angle change, length variations, and osculating circle radius.

2.4.1 Turning Angle

Theorem 2.4.1: Total Turning Angle

For a closed polygonal curve, the sum of the turning angles is exactly

$$\sum_i \alpha_i = 2\pi \cdot n$$

where n is the turning number (typically 1 for a simple polygon with positive orientation). [4]

Proof

Consider a discrete curve having a sequence of points $\gamma_0, \gamma_1, \ldots, \gamma_n = \gamma_0$. The tangents are defined on the edges $\mathbf{t}_i = \frac{\gamma_{i+1}-\gamma_i}{\|\gamma_{i+1}-\gamma_i\|}$.

Then the turning angle α_i is the signed angle between two consecutive tangents $\mathbf{t}_{i-1}$ and $\mathbf{t}_i$, measured by $\alpha_i = \angle(\mathbf{t}_{i-1}, \mathbf{t}_i)$ and each turning angle α_i is the exterior angle at the vertex i, which quantifies how much the tangent *turns* at that point.

For a closed curve, the total turning of the tangent vector as it goes around the curve once must bring it back to its starting orientation. In the plane, this total angular change is exactly

$$\sum_i \alpha_i = 2\pi \cdot n$$

where n is the turning number.

For a simple polygon (non-self-intersecting), $n = 1$, so $\sum_i \alpha_i = 2\pi$

Theorem 2.4.2: The Gauss-Bonnet Theorem for Plane Curve

For closed, simple, positively oriented curve $\gamma : I \to R^2$, the total curvature of the curve will be

$$\int_\gamma \kappa \, ds = 2\pi \cdot n$$

where κ is the curvature of the discrete curve. [4]

In analogy to the smooth curve, the curvature of the discrete curve is defined using

$$(\kappa ds)_i := \alpha_i \quad \Rightarrow \quad \sum_i \kappa_i \, ds_i = \sum_i \alpha_i = 2\pi \cdot n$$

$$k_i ds_i = \alpha_i \Rightarrow k_i = \frac{\alpha_i}{\ell}$$

if $\ell = 1$ then $k_i = \alpha_i$

2.4.2 Length Variation

Theorem 2.4.3: Length Variation

Variation in the length can define the curvature of the discrete curves.

Proof

Let $\gamma_{i-1}, \gamma_i, \gamma_{i+1}$ be three points on the discrete curve γ and total length of the curve is

$$L = \sum_i \ell_{i,i+1} = \sum_i |\gamma_{i+1} - \gamma_i|$$

by small perturbation to the point γ_i as $\gamma_i = \gamma_i + \epsilon\dot{\gamma}_i$ where $\dot{\gamma}_i$ affects the two adjacent incoming and outgoing edge vector at point γ_i of length $\ell_{i-1,i}$&$\ell_{i,i+1}$ resp. Therefor change in the total length of curve L w.r.t to the point γ_i will be

$$\frac{\partial L}{\partial \gamma_i} = \frac{\partial \ell_{i-1,i}}{\partial \gamma_i} + \frac{\partial \ell_{i,i+1}}{\partial \gamma_i}$$

$$= \frac{\partial |\gamma_i - \gamma_{i-1}|}{\partial \gamma_i} + \frac{\partial |\gamma_{i+1} - \gamma_i|}{\partial \gamma_i}$$

$$= \frac{\gamma_i - \gamma_{i-1}}{|\gamma_i - \gamma_{i-1}|} + \frac{\gamma_i - \gamma_{i+1}}{|\gamma_i - \gamma_{i+1}|}$$

$$= \mathbf{t}_{i-1} - \mathbf{t}_i$$

We want the first variation of L, i.e., $\dot{L}$, under a perturbation $\gamma_i \mapsto \gamma_i + \varepsilon\dot{\gamma}_i$. So, a

discrete analogy to smooth variation.

$\dot{L} = \sum_i \left\langle \frac{\partial L}{\partial \gamma_i}, \dot{\gamma}_i \right\rangle = -\int_I \langle \kappa \mathbf{N}, \dot{\gamma} \rangle ds$

$\dot{L} = \sum_i \langle \mathbf{t}_{i-1} - \mathbf{t}_i, \dot{\gamma}_i \rangle = -\sum_i \langle \kappa_i \mathbf{N}_i, \dot{\gamma}_i \rangle (\Delta s)_i$

where $\kappa_i \mathbf{N}_i \approx \mathbf{t}_{i+1} - \mathbf{t}_i$ defines the discrete curvature, and $(\Delta s)_i$ is the discrete arc length element around vertex i.

2.4.3 Osculating Circle

Theorem 2.4.4: Osculating circle

Given a discrete curve $\gamma : I \to R^2$ and let any three consecutive point $\gamma_{i-1}, \gamma_i, \gamma_{i+1}$ on it then curvature κ_i at point γ_i will be

$$\kappa_i = \frac{2}{\ell} sin(\frac{\alpha_i}{2})$$

where α_i be the turning angle at point γ_i and l be the length of the differential(edge vector). [4]

Proof

Given $\gamma : I \to R^2$ be a discrete curve and let consider any three consecutive point $\gamma_{i-1}, \gamma_i, \gamma_{i+1}$ be on γ.

Let α_i be the turning angle at point γ_i, that is the angle between the vectors $\gamma_i - \gamma_{i-1}$ and $\gamma_{i+1} - \gamma_i$.

Assume that $\ell_- = \ell_+ = \ell$ where $\ell_- = |\gamma_i - \gamma_{i-1}|$ and $\ell_+ = |\gamma_{i+1} - \gamma_i|$ be the consecutive edge length of edge vectors.

Then there will be a unique circle passing through the three consecutive points $\gamma_{i-1}, \gamma_i, \gamma_{i+1}$ and d will be the chord of that circle between point γ_{i-1} and γ_{i+1}.

Thus, $\gamma_{i-1}, \gamma_i, \gamma_{i+1}$ forms a triangle and α_i be the angle at vertex γ_i.

Now, by law of sin on circle

$$\frac{1}{2R} = \frac{\sin(180° - \alpha_i)}{d} = \frac{\sin(\alpha_i)}{d} \quad \Rightarrow \quad R = \frac{d}{2\sin(\alpha_i)} \tag{2.1}$$

and by law of cosine $d^2 = \ell^2 + \ell^2 - 2\ell \cdot \ell \cdot \cos(180° - \alpha_i) = 2\ell^2(1 + \cos(\alpha_i))$

$\Rightarrow d^2 = 2\ell^2(1 + \cos(\alpha_i))$

by using trigonometric identity $\cos(\alpha) = 2\cos^2(\alpha/2) - 1$ we get

$$1 + \cos(\alpha_i) = 2\cos^2\left(\frac{\alpha_i}{2}\right) \quad \Rightarrow \quad d^2 = 4\ell^2\cos^2\left(\frac{\alpha_i}{2}\right) \quad \Rightarrow \quad d = 2\ell\cos\left(\frac{\alpha_i}{2}\right)$$

Putting value of d in the our first equation we will get

$R = \frac{d}{2\sin(\alpha_i)} = \frac{2\ell\cos(\alpha_i/2)}{2\sin(\alpha_i)} = \frac{\ell\cos(\alpha_i/2)}{\sin(\alpha_i)}$

by $\sin(\alpha) = 2\sin(\alpha/2)\cos(\alpha/2)$

$R = \frac{\ell\cos(\alpha_i/2)}{2\sin(\alpha_i/2)\cos(\alpha_i/2)} = \frac{\ell}{2\sin(\alpha_i/2)} \quad \Rightarrow \quad \kappa_i = \frac{1}{R} = \frac{2}{\ell}\sin\left(\frac{\alpha_i}{2}\right)$

2.5 Fundamental Theorem of Discrete Curves

Theorem 2.5.1: Fundamental Theorem of Discrete Plane Curves

A regular discrete plane curve is uniquely determined by its edge lengths and turning angles, up to a rigid motion. [**crane2018discrete**]

Algorithm 1 Plane Curve Reconstruction

Require: Initial point, Edge lengths and Curvatures : $\gamma_0 \in \mathbb{R}^2$, $\ell = [\ell_0, \ldots, \ell_{n-1}]$ and $\kappa = [\kappa_0, \ldots, \kappa_{n-1}]$

Ensure: Plane curve $\gamma = [\gamma_0, \ldots, \gamma_n]$

1: $\gamma[0] \leftarrow \gamma_0$
2: $T \leftarrow (1, 0)$
3: **for** $i = 0$ **to** $N - 1$ **do**
4: $\quad \Delta\theta \leftarrow \kappa[i] \cdot \ell[i]$
5: $\quad T \leftarrow \mathcal{R}_{2D}(T, \Delta\theta)$
6: $\quad \gamma[i + 1] \leftarrow \gamma[i] + \ell[i] \cdot T$
7: **return** $\gamma[0 \ldots N]$
8: $\mathcal{R}_{2D}((x, y), \theta) = (x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta)$.

Theorem 2.5.2: Fundamental Theorem of Discrete Space Curves

A discrete space curve is uniquely determined by its edge length, curvature(exterior angle at vertex) and torsion(Change in the normal i.e. angle between edges joining two planes.), upto rigid motion.[**crane2018discrete**]

Definition 2.5.1: Rotation Operator

Given $u = (u_x, u_y, u_z)$, the skew-symmetric matrix $\hat{u}$ is defined as: $\hat{u} = \begin{bmatrix} 0 & u_z & -u_y \\ -u_z & 0 & u_x \\ u_y & -u_x & 0 \end{bmatrix}$ The rotation operator is: $R(u, \theta) = \exp(\theta \hat{u})$ which rotates the vector by angle θ around axis u

Algorithm 2 Space Curve Reconstruction

Require: $\ell_{ij}, \kappa_i, \tau_{ij}, \gamma_0$
Ensure: Space curve $\gamma_0, \gamma_1, \ldots, \gamma_n$

Initialize T, N ▷ Orthonormal tangent/normal vectors
$\gamma[0] \leftarrow \gamma_0$
for $i = 1$ **to** n **do**
    $\gamma_i \leftarrow \gamma_{i-1} + \ell_{i-1,i} T$
    $T \leftarrow \mathcal{R}(N, \kappa_i) T$ ▷ Rotate tangent by curvature
    $N \leftarrow \mathcal{R}(T, \tau_{i,i+1}) N$ ▷ Twist normal by torsion
return $\{\gamma_i\}_{i=0}^{n}$
$\kappa_i = i/n, \tau_{ij} = (n-i)/n$

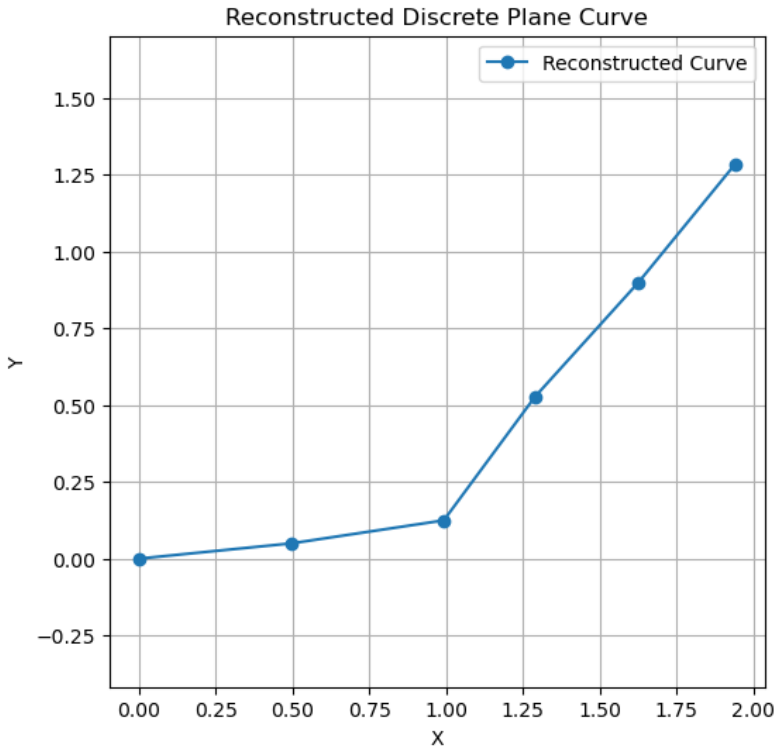


Figure 2.6: Fundamental theorem for plane curves

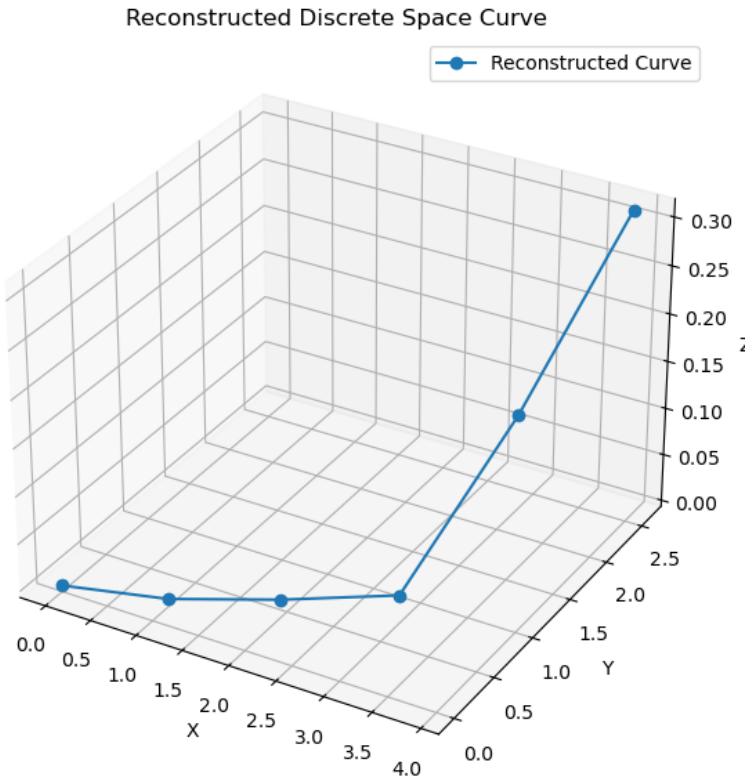


Figure 2.7: Fundamental theorem for space curves

Chapter 3

Discrete Surfaces

We have seen discrete curve which is a simplicial 1-complex realised in euclidean space, have geometric invariant quantities like curvature and torsion etc. used to state that given initial point and geometric invariant quantities, can retrace the curve upto rigid motion. Now, we will move towards discrete surfaces.

In smooth manifolds, we know that a surface is a 2-dim manifold realized in 3-dim euclidean space and if realization is nice enough, then it's an immersed surface or embedded surface in 3D euclidean space. Thus, a discrete surface is a simplicial 2-complex $M = (V, E, F)$ which is realized discrete 2D manifold(piecewise linear surface) M by function $f : M \to R^3$. So, we can say that a piecewise linear surface is a polygonal mesh with the property that it is locally looks like a 3D Euclidean space. What is polygonal mesh ? A polygonal mesh consists of a collection of vertices, edges and faces as V, E & F resp and each edge $e_{ij} \in E$ is a map from $v_i \to v_j$ where $v_i, v_j \in V$ and each face is a polygon descibed by a number n_f of distinct edges$f = \{e_{12}, \ldots, e_{n_f - 1 n_f}\} = \{(v_1, v_2), \ldots, (v_{n_f - 1}, v_{n_f})\}$ so that every vertex involved in these edges occure exactly twice. Therefore, a face is a cyclically ordered list of vertices $f = (v_1, \ldots, v_{n_f})$.

Given a polygonal mesh, when can we say it's a discrete manifold ?

A polygonal mesh $M = (V, E, F)$ is said to be a discrete manifold if each edge $e_{ij} \in E$ is incident to one or two faces and the faces incident to each vertex $v_i \in V$ form a closed/open

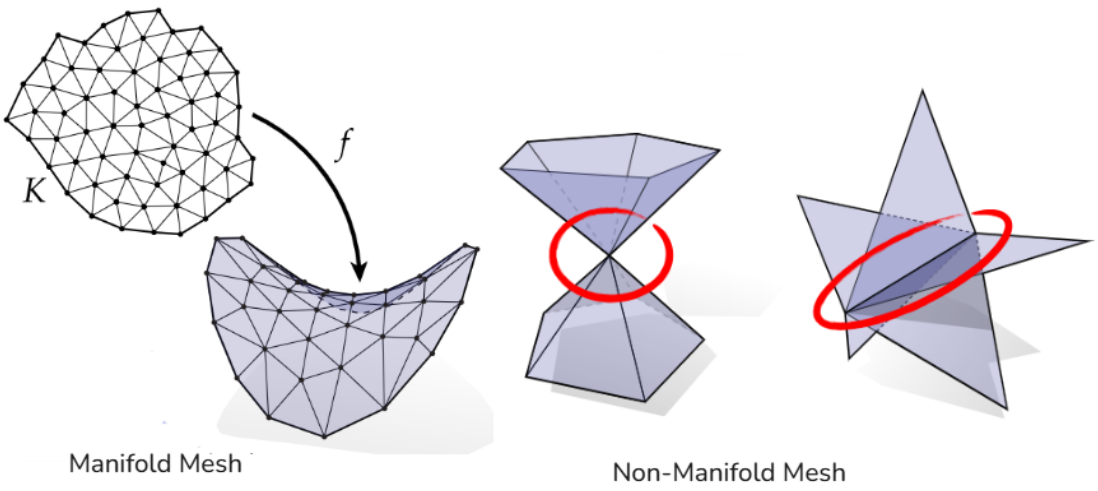


Figure 3.1: Left : Manifold mesh, Right : Non-manifold mesh [5]

cell.

3.1 Discrete 2D Manifolds

There are primarily two types of models for discrete surface

1. Simlicial 2-manifolds which naturally fits with exterior calculus

2. Nets, a piecewise integer lattice which is a natural fit with discrete integrable system.

But simplicial models of discrete surfaces are commonly used in many applications therefore they will be of our interest also for creating mesh manifolds.

Definition 3.1.1: Piecewise linear surface

A triangle mesh $M = (V, E, F)$ is realised as a surface in 3D by assigning a vertex position per vertex as a map $f : V \to R^3$. these vertex position are affinly interpolated to form straight edges and flat triangular faces. The resulting surface is called a piecewise linear surface. [4]

3.2 Topology of Discrete Manifolds

Topology of discrete manifolds discuss the connectivity of the faces, edges and vertices.

Definition 3.2.1: Euler polyhedral formula

Every polygonal disk and sphere with vertices, edges and faces as V, E & F resp. satisfies

$$|V| - |E| + |F| = 1 \tag{3.1}$$

and

$$|V| - |E| + |F| = 2 \tag{3.2}$$

resp.

Theorem 3.2.1: Euler Characteristics

The euler characteristics of any discrete manifold $M = (V, E, F)$ is defined by

$$\chi(M) := |V| - |E| + |F| \tag{3.3}$$

3.3 Regularity of Discrete Surface

Consider a triangle mesh M and a map $f : M \to \mathbb{R}^3$ is a **discrete immersion** if every **vertex star is embedded**, i.e., the union of faces around every vertex is mapped injectively into $\mathbb{R}^3$. This prevents local branching and ensures a well-defined tangent space.

Definition 3.3.1: Surface Immersion and Embedding

A piecewise linear surface is an immersion if

1. Each triangle has a non-zero area
2. Triangles incident to each vertex do not intersect each other.

A piecewise-linear surface is an embedding if it has no self-intersection. [5]

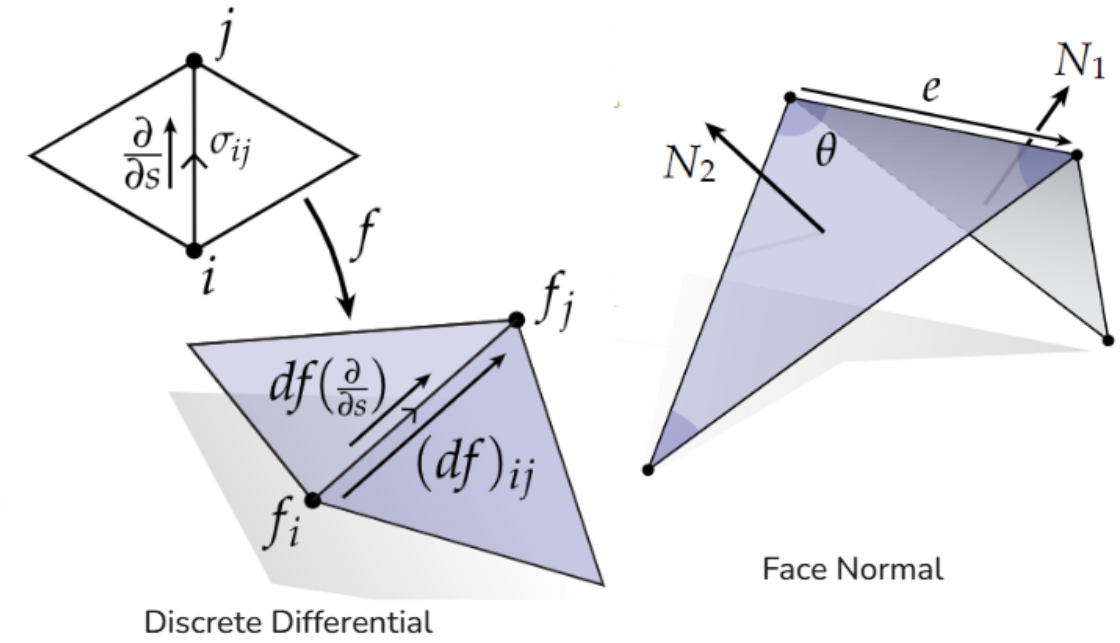


Figure 3.2: Left : Discrete differential, Right : Discrete normal [5]

3.4 Differential, Tangent and Normal of Discrete Surface

Definition 3.4.1: Discrete Differential

The discrete differential df is the exterior derivative of a 0-form f, and it assigns one value per oriented edge (i, j). Formally,

$$(df)_{ij} = \int_{(i,j)} df \overset{\text{Stokes}}{\Longrightarrow} \int_{\partial(i,j)} f = f_j - f_i.$$

Hence, the discrete differential is simply the edge vector from i to j, $(df)_{ij} = f_j - f_i$.
Remark -1 : df is a discrete 1-form and it is asymmetric with respect to orientation $(df)_{ij} = -(df)_{ji}$.

Definition 3.4.2: Discrete Tangent

The discrete tangent is a unit length edge vector such that for a triangular face with vertices i, j, k, we define two edge vectors using df: $(df)_{ij} = f_j - f_i, \quad (df)_{ik} = f_k - f_i$. These edge vectors span Tangent plane of the face $=$ span$\{(df)_{ij}, (df)_{ik}\}$. and also encode the local orientation of the surface.

Definition 3.4.3: Discrete Normal

The discrete face normal N_{ijk} of triangle (i, j, k) is computed using the cross product of edge vectors(discrete differentials)

$$N_{ijk} = \frac{(df)_{ij} \times (df)_{jk}}{\|(df)_{ij} \times (df)_{jk}\|}.$$

Where, $(df)_{ij} \times (df)_{jk}$ is a vector perpendicular to the face.

Remark-1 The normalization ensures that N_{ijk} is a unit normal and orientation is consistent using the right-hand rule (counter-clockwise vertex order).

Definition 3.4.4: Triangle Area and Total Area of Mesh

Let M be a triangle mesh and f_{ijk} be a face with edges f_{ij} & f_{ik} then face(triangle) is

$$A_i = \frac{1}{2}|f_{ij} \times f_{ik}|$$

and Total area of the mesh M is

$$\sum_{\forall\, f_{ijk} \in F} A(f_{ijk})$$

Algorithm 3 ComputeFaceAreas

procedure ComputeFaceAreas(V, F)
  Input: $V \in \mathbb{R}^{n\times 3}$ (vertices), $F \in \mathbb{N}^{m\times 3}$ (faces)
  Output: `face_areas` $\in \mathbb{R}^m$
  for each face $f = (i, j, k)$ in F **do**
    $\mathbf{v}_0 \leftarrow V[i],\ \mathbf{v}_1 \leftarrow V[j],\ \mathbf{v}_2 \leftarrow V[k]$
    $\mathbf{e}_1 \leftarrow \mathbf{v}_1 - \mathbf{v}_0, \quad \mathbf{e}_2 \leftarrow \mathbf{v}_2 - \mathbf{v}_0$
    $\mathbf{n} \leftarrow \mathbf{e}_1 \times \mathbf{e}_2$
    $A_f \leftarrow \frac{1}{2}\|\mathbf{n}\|$
    Store A_f in `face_areas`
  return `face_areas`

Definition 3.4.5: Interior, Bending and Dihedral Angle

Let M be a triangle mesh and for each face of f_{ijk}, vertex f_i then **the interior angle**

at f_i is

$$\theta^i_{jk} = cos^{-1}(\langle \hat{f}_{ij}, \hat{f}_{ik} \rangle)$$

. For each interior edge f_{ij} shared by two triangle faces f_{ijk} and f_{lkj} then **Bending angle**

$$\alpha_j k = \sin^{-1}\left(\langle \hat{f}_{jk}, N_{ijk} \times N_{\ell kj} \rangle\right)$$

. For edge f_{jk} shared by two faces f_{ijk} and f_{lkj} that have a normal face N_{ijk} and N_{lkj} then the dihedral angle **between two adjacent faces** will be

$$\phi_{jk} = \cos^{-1}\left(\langle N_{ijk}, N_{\ell kj} \rangle\right)$$

. [4]

Area around a vertex of mesh

We need to define the region around a vertex i.e. portion of area owned by a vertex i, there are many possible types of cells that define region but we only focus on the barycentric cells which define **barycentric area of cell is** area of each triangle incident to the vertex i divided equally among its three vertices.

$$A_i = \frac{1}{3} \sum_{f_{ijk}} \text{Area}(f_{ijk}) \tag{3.4}$$

We frequently use this equation such as to make the Laplacian scale-invariant, we normalize by an area A_i, representing the portion of the surface ”owned” by vertex i.

The vertices are 0-form thus if we take the hodge star of it will give us an area mass matrix and the total area of the mesh remains the same as the sum of diagonal entries of the area

mass matrix is equal to the total face area.

$$\mathbf{A} = \begin{bmatrix} m_1 & & & \\ & m_2 & & \\ & & \ddots & \\ & & & m_{|V|} \end{bmatrix} \in R^{|V|\times|V|}$$

where, $m_i = A_i$

Algorithm 4 AreaMatrix

procedure AreaMatrix(V, F)
  Input: $V \in \mathbb{R}^{n\times 3},\ F \in \mathbb{N}^{m\times 3}$
  Output: $A \in \mathbb{R}^{n\times n}$ (diagonal mass matrix)
  face_areas $\leftarrow$ ComputeFaceAreas(V, F)
  vertex_areas $\leftarrow \mathbf{0}_n$
  for each face $f = (i, j, k)$ and area A_f **do**
    vertex_areas$[i]$ += $A_f/3$
    vertex_areas$[j]$ += $A_f/3$
    vertex_areas$[k]$ += $A_f/3$
  $A \leftarrow$ diag(vertex_areas)
  return A

Discrete vertex normal: The Discrete gaussian map for a discrete immersion is the triangle normal but it doesn't describe the vertex or edge normal. There are many approaches to describe it but none-of them are exact such as area weighted vertex normal and angle weighted vertex normal but there is a better approach is to start with smooth and then apply the principle of discretization.

In smooth, a vector area is the integral of the normal to the surface in area element over the region as

$$A = \int_\Omega N dA = \frac{1}{2}\int_\Omega df \wedge df = \frac{1}{2}\int_{\partial\Omega} f \wedge df$$

The idea of discretization is to integrate NdA over dual cell C_i to get our vertex normal. Let's consider the dual cell to be a barycentric cell then it's area will be

$$= \frac{1}{3}A = \frac{1}{3}\int_{C_i} N dA = \frac{1}{3}\times\frac{1}{2}\int_{C_i} df \wedge df = \frac{1}{6}\int_{\partial C_i} f \wedge df$$

$$= \frac{1}{6} \sum_{ij \in \partial C_i} \int_{e_{ij}} f \times df = \frac{1}{6} \sum_{ij \in \partial C_i} (\frac{f_i + f_j}{2}) \times (f_j - f_i)$$

$$= \frac{1}{6} \sum_{ij \in \partial C_i} (f_i \times f_j)$$

Cotangent Weights is the weight associated to the edges means

$$w_{e_{ij}} = \frac{\text{width of dual edge}}{\text{length of primal edge}} = \frac{|e^*_{ij}|}{|e_{ij}|} \tag{3.5}$$

Consider a triangle mesh $M = (V, E, F)$ by taking hodge star of M then it's dual will be $M^* = (V^*, E^*, F^*)$ and dual center c_k and c_l for the vertex v_k and v_l will be inside the triangle f_{ijk} and f_{ijl}, where $e_{ij} \in E$ and it's dual edge $e_{ij} \in E^*$ obtained by joining dual centers c_k and c_l and are orthogonal to each other.

Let angle corresponding to edge e_{ij} at the vertex v_k and v_l as α_{ij} and β_{ij} resp. Dual edge length

$$|e^*_{ij}| = v_k c_k + c_l v_l$$

$$\frac{|e_{ij}|}{2} cot(\alpha_{ij}) + \frac{|e_{ij}|}{2} cot(\beta_{ij})$$

$$= \frac{|e_{ij}|}{2} (cot(\alpha_{ij}) + cot(\beta_{ij}))$$

Then edge weight will become

$$w_{e_{ij}} = \frac{\frac{|e_{ij}|}{2}(cot(\alpha_{ij}) + cot(\beta_{ij}))}{|e_{ij}|} = \frac{1}{2}(cot(\alpha_{ij}) + cot(\beta_{ij})) \tag{3.6}$$

We knew that the edges represents the 1-form thus if we take the hodge star of them we

will get the edge weight matrix

$$W = \begin{bmatrix} w_1 & & & \\ & w_2 & & \\ & & \ddots & \\ & & & w_{|E|} \end{bmatrix} \in R^{|E|\times|E|}$$

3.5 Geometric Measurement on Triangle Mesh

We measure two different types of geometric quantities on triangle mesh

1. Intrinsic quantity depends solely on edge lengths
2. Extrinsic quantity depends on how the triangle mesh is embedded into 3D space.

Definition 3.5.1: Discrete metric(Polyhedral Metric)

A discrete metric $l : E \to R$ on an abstract triangulated surface $M = (V, E, F)$ is defined by

1. Assigning positive lengths to every edge.
2. Ensuring the triangle inequality holds for every face i.e for any triangle of edge length $\ell_{ij}, \ell_{jk}, \ell_{ki}$,

$$\ell_{ij} + \ell_{jk} > \ell_{ki}, \quad \ell_{jk} + \ell_{ki} > \ell_{ij}, \quad \ell_{ki} + \ell_{ij} > \ell_{jk}.$$

This special structure allows us for computations to depend only on intrinsic geometry, independent of 3D embedding(extrinsic geometry).

Therefore, we can say that triangle area and interior angles are intrinsic quantities because edge length uniquely determine a triangle shape in plane, Heron's formula and law of cosines $A_{ijk} = \frac{1}{4}\sqrt{(\ell_{ij} + \ell_{jk} + \ell_{ki})\,(-\ell_{ij} + \ell_{jk} + \ell_{ki})\,(\ell_{ij} - \ell_{jk} + \ell_{ki})\,(\ell_{ij} + \ell_{jk} - \ell_{ki})}$ and $\theta^i_{jk} = \cos^{-1}\left(\frac{\ell^2_{ij}+\ell^2_{ki}-\ell^2_{jk}}{2\ell_{ij}\ell_{ki}}\right)$

can compute area and define angle directly from edge length alone. Even though computation may involve edge vectors etc, the final result will only depend on the metric.

Example : If a curve $\gamma : I \to M$ travels within a single triangle, these measurements such as total length, curvature, turning angle, etc. are well defined because the triangle's shape is fixed by the edge lengths. When the curve crosses an edge shared by two neighboring triangles, both triangles can be realized in the plane as Euclidean triangles sharing that edge. Because the shared edge has the same length in both triangles, the two triangles fit in the plane, and the curve segment that spans them can be treated as a continuous plane curve.

Thus, the geometry of the triangulated surface with a discrete metric behaves like a flat, unfolded polyhedron except at vertices where the sum of angles around a vertex may differ from 2π.

3.6 Discrete Curvature

3.6.1 Discrete Gauss-Bonnet Theorem

Given a discrete immersion $f : M \to \mathbb{R}^3$, for each triangle(face) $f_{ijk} \in F$, the **face normal** N_{ijk} (also called the Gauss map value at that face) is defined as:

$$N_{ijk} = \frac{(df)_{ij} \times (df)_{jk}}{\|(df)_{ij} \times (df)_{jk}\|}$$

This is a unit vector orthogonal to the face f_{ijk}. Therefore, the **Discrete Gauss Map** assigns one unit normal vector per triangle (a dual 0-form).

Visualization: These face normals can be visualized as points on the unit sphere. Adjacent normals may be connected by geodesic arcs to represent local curvature change.

Note: The discrete Gauss map does *not* define normals at edges or vertices.

Definition 3.6.1: Angle Deflect

Consider a mesh manifold $M = (V, E, F)$ and a set of all triangle incident to the

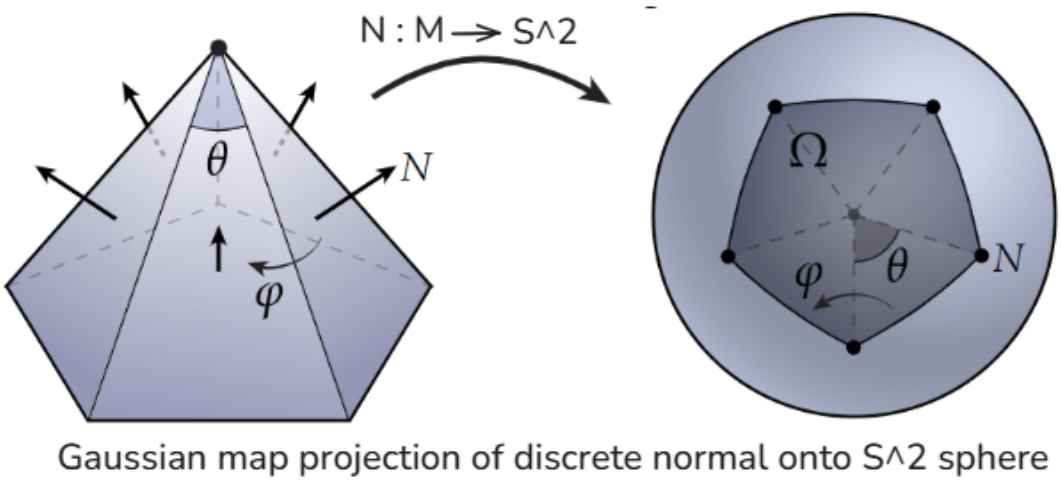


Figure 3.3: Discrete Gauss map [5]

vertex $v_i \in V$ then the angle deflected at the vertex i is

$$\Omega_i = 2\pi - \sum_{jk} \theta^i_{jk}$$

, where each triangle contributes an interior angle θ^i_{jk}.

Thus we can say that if $\Omega_i >=< 0$ then a positive, negative and zero curvature at that vertex.

Definition 3.6.2: Discrete Gaussian Curvature

For each $i^{th} \in V$ vertex, we define the discrete gaussian curvature as angle defect as follows

$$(KdA)_i = \Omega_i = 2\pi - \sum_{jk} \theta^i_{jk}$$

,

We know that gaussian curvature at the boundary of the surface vanishes but we can flatten the boundary vertices without stretching and can measure how straight the boundary is.

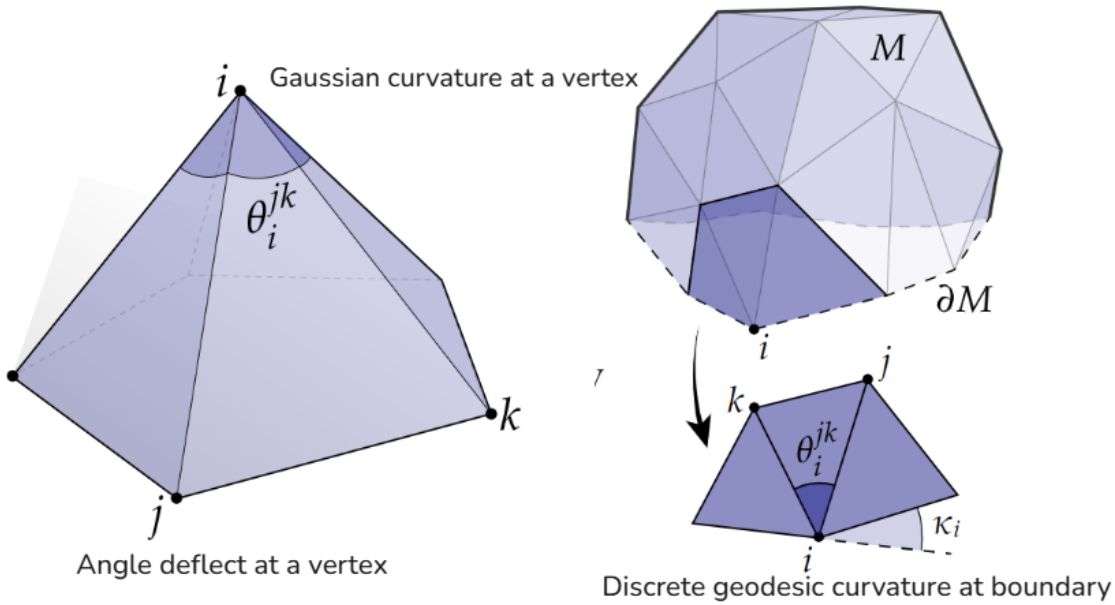


Figure 3.4: Left : Angle deflect and discrete gaussian curvature, Right : discrete geodesic curvature [5]

Definition 3.6.3: Discrete Geodesic Curvature

A measure of angle deflect around the boundary vertices such that

$$k_i = \pi - \sum_{f_{ijk}} \theta_i^{jk}$$

For a smooth surface with genus g, the total gauss curvature is

$$\int_M K dA = 2\pi\chi$$

where χ is the characteristic number. We want to use the angle deflection definition of gaussian curvature to discretise this theorem as follows.

Theorem 3.6.1: Discrete Gauss Bonnet Theorem

For the closed, discrete surface $M = (V, F, E)$ of genius g, the total angle deflected at the i^{th} vertex is

$$\sum_{i \in V} \Omega_i = 2\pi\chi(M)$$

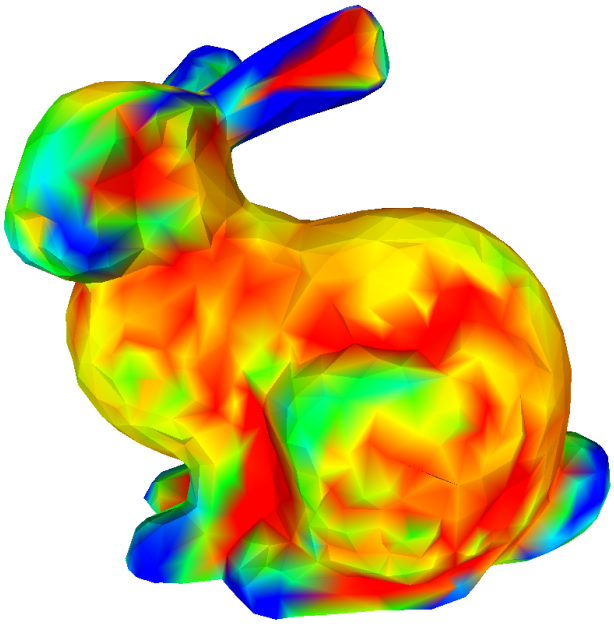


Figure 3.5: Gaussian-curvature

Proof

Let $M = (V, E, F)$ be a mesh manifold, having a set of vertices, edges and faces as V, E & F resp. We know that curvature is concentrated at vertices of mesh. Thus, curvature at each vertex is the angle defect Ω_i. The total curvature is equal to the sum of total angle deflect at each vertices of the mesh

$$\sum_{i \in V} \Omega_i = \sum_{i \in V} \left(2\pi - \sum_i \theta^i_{jk} \right)$$

Summing over all vertices and rearranging then

$$\sum_{i \in V} \Omega_i = 2\pi |V| - \sum_{i \in V} \sum_{j \neq k} \theta^i_{jk}$$

$$= 2\pi |V| - \sum_{\text{interior angle all triangle}} \theta$$

But the interior angle of the triangle is π. Thus, sum of all triangle's interior angle will be $\pi |F|$

So total angle defect

$$\sum_{i \in V} \Omega_i = 2\pi|V| - \pi|F|$$

$$= 2\pi(|V| - \frac{|F|}{2})$$

$$= 2\pi(|V| + |F| - \frac{3}{2}|F|)$$

Every face is a triangle and each triangle has 3 edges and 3 vertices but each edge is shared by 2 face for a closed triangle mesh thus we have identity $3|F| = 2|E| = |F| = \frac{3}{2}|E|$ means each edge shared by 2 triangles thus

$$\sum_i \Omega_i = 2\pi(|V| - |E| + |F|)$$

using Euler characteristic for triangulated surface$\chi = |V| - |E| + |F|$

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

Definition 3.6.4: Mean Curvature

how that the mean curvature normal vector at vertex i is

$$H_i \mathbf{n}_i = \frac{1}{2} \sum_{j \in N(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(\mathbf{f}_j - \mathbf{f}_i)$$

We know that the smooth mean curvature of the surface is the integration of $HNdA$ over the region Ω as

$$\int_\Omega HNdA = \int_\Omega df \wedge dN = \int_\Omega dN \wedge df = \int_\Omega N \wedge df$$

,

Now, let's apply the discretization principle by integration of $HNdA$ over the barycentric cell C_i Then

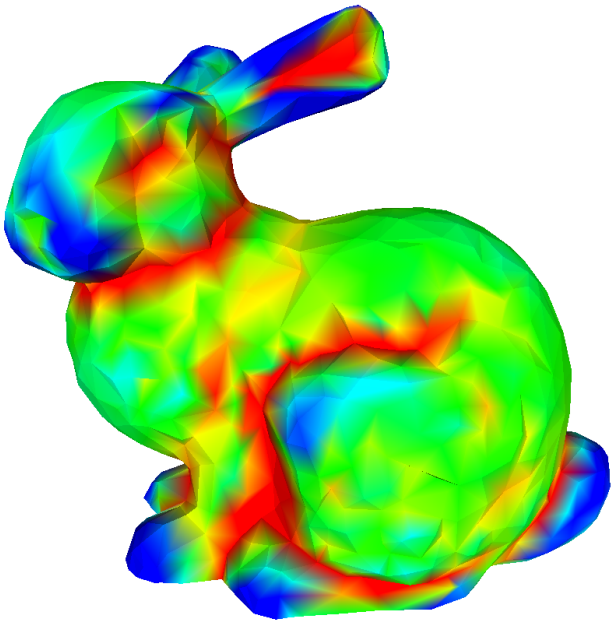


Figure 3.6: Mean-curvature

$$\int_{\Omega} N \wedge df = \sum_{j} \int_{e^*_{ij}} N \wedge df$$

, where for all $e_{ij} \in E$ there exist a dual edge e^*_{ij} such that

$$(HN)_i = \sum_{j \in N(i)} w_{ij}(f_j - f_i)$$

where $w_{ij} = \cot \alpha_{ij} + \cot \beta_{ij}$ and A_i is the Voronoi area or 1/3 of the surrounding triangle areas.

The cotangent weights come from discretizing the divergence theorem over the region around vertex i, approximating the Laplacian. Therefore

$$H_i N_i = \frac{1}{2} \sum_{j \in N(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(f_j - f_i)$$

Definition 3.6.5: Principal Curvature

For a given $f : M \to R^3$ discrete surface then the discrete principal curvature are

$$\kappa_1 = \frac{H_i}{A_i} - \sqrt{\left(\frac{H_i}{A_i}\right)^2 - \frac{K_i}{A_i}} \quad , \quad \kappa_2 = \frac{H_i}{A_i} + \sqrt{\left(\frac{H_i}{A_i}\right)^2 - \frac{K_i}{A_i}}$$

. Where H_i discrete mean curvature at vertex i, K_i discrete Gaussian curvature at vertex i and A_i area of the vertex region

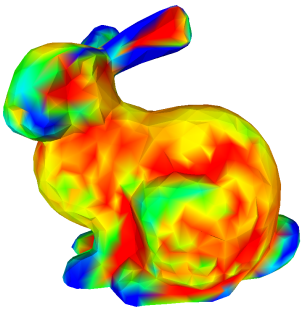


Figure 3.7: Min-curvature

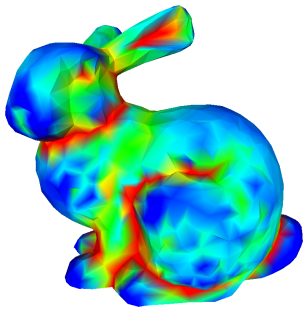


Figure 3.8: Max-curvature

For a given $f : M \to R^3$ discrete surface, we have a task about the Gaussian, mean, and geodesic curvature but not about discrete principal curvature. So, smooth mean and Gaussian curvature are

1. Mean curvature: $H = \frac{\kappa_1 + \kappa_2}{2}$
2. Gaussian curvature: $K = \kappa_1 \cdot \kappa_2$

where, κ_1, κ_2. by solving this quadratic equation, we will get the principal curvatures

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

In the discrete setting, we mimic the smooth relationships using discrete approximations but the problem is that discrete gaussian is the quantity of vertices and discrete mean curvature is the quantity of edges. How to use quantities defined on different places for computation of the principal curvature, the one idea could be vertex mean curvature means integrating the edge mean curvature over the neighbors of the vertex i.

Discrete Gaussian Curvature at vertex i.

$$H_i = \frac{1}{4}\sum_{e_{ij}} l_{ij}\phi_{ij}$$

, where ϕ_{ij} is the dihedral angle between two faces shared by common edgee_{ij} and l_{ij} is the edge length.

We know that H_i is the total mean curvature over the area of dual cell A_i thus mean curvature at the vertex will be $\frac{H_i}{A_i}$ similarly geodesic curvature at vertices will be $\frac{K_i}{A_i}$.

Now we add these discrete quantities which are scalar value in the smooth formula Then the discrete principal curvatures become

$$\kappa_1 = \frac{H_i}{A_i} - \sqrt{\left(\frac{H_i}{A_i}\right)^2 - \frac{K_i}{A_i}} \tag{3.7}$$

and

$$\kappa_2 = \frac{H_i}{A_i} + \sqrt{\left(\frac{H_i}{A_i}\right)^2 - \frac{K_i}{A_i}} \tag{3.8}$$

This matches the exact smooth derivation.

3.6.2 Laplacian Operator

Laplacian Operators exist in a variety of applied mathematics problems involving geometry of curved surfaces and physical models of complex systems and we needed a computational way to solve these models on the actual curved surfaces. Therefore, I wanted to discretize the Laplacian operator such that the physical model can be solved onto a discrete surface(triangular mesh).

There are two different ways of discretization of the laplacian operator

1. **Finite Element Method(FEM)**
2. **Discrete Exterior Calculus(DEC)**

FEM and DEC both lead to the same mathematical formula(cotangent) but through a dif-

ferent approach. Here I have used the DEC approach and mean curvature relationship with the laplacian operator on curved surfaces.

Definition 3.6.6: Discrete Laplacian Operator

Consider $f : M = (V, E, F) \rightarrow R^3$ be discrete surface then the mean curvature of the surface is $\Delta f_i = 2(HN)_i$ at vertex i of a triangle mesh then laplacian operator will be

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

, where f_i is the position vector of vertex $i \in V$, $j \in N(i)$ are star(i) neighboring vertices of i and α_{ij}, β_{ij} are the two angles opposite the edge f_{ij}.

Note: We have divide our cotangent formula with the area of C_i cell to get the average value over the cell

Chapter 4

Applications in Geometric Processing

4.1 Smoothing

4.1.1 Discrete Curve Smoothing

A discrete curve is polygonal, an approximate representation of the smooth curve that has noise. For plane and space discrete curves, we use uniform laplace flow which is the second derivative of the curve that mimics curvature along the curve by removing noise / high frequency terms.

Let $\gamma : I \rightarrow R^d$ be a discrete curve defined as $\gamma(i) = \gamma_i \in R^d \ \forall \ i \in I$. Uniform laplacian measures how much γ_i deviates from being between its neighbors. Thus, the heat diffusion equation

$$\frac{\partial \gamma}{\partial t} = \Delta \gamma$$

It can be discretize using finite difference approximation as

$$\frac{\gamma_{i+1} - \gamma_i}{\delta t} = \frac{1}{2}(\gamma_{i-1} - \gamma_i) + \frac{1}{2}(\gamma_{i+1} - \gamma_i) = \frac{1}{2}(\gamma_{i-1} - 2\gamma_i + \gamma_{i+1}) = \Delta \gamma_i$$

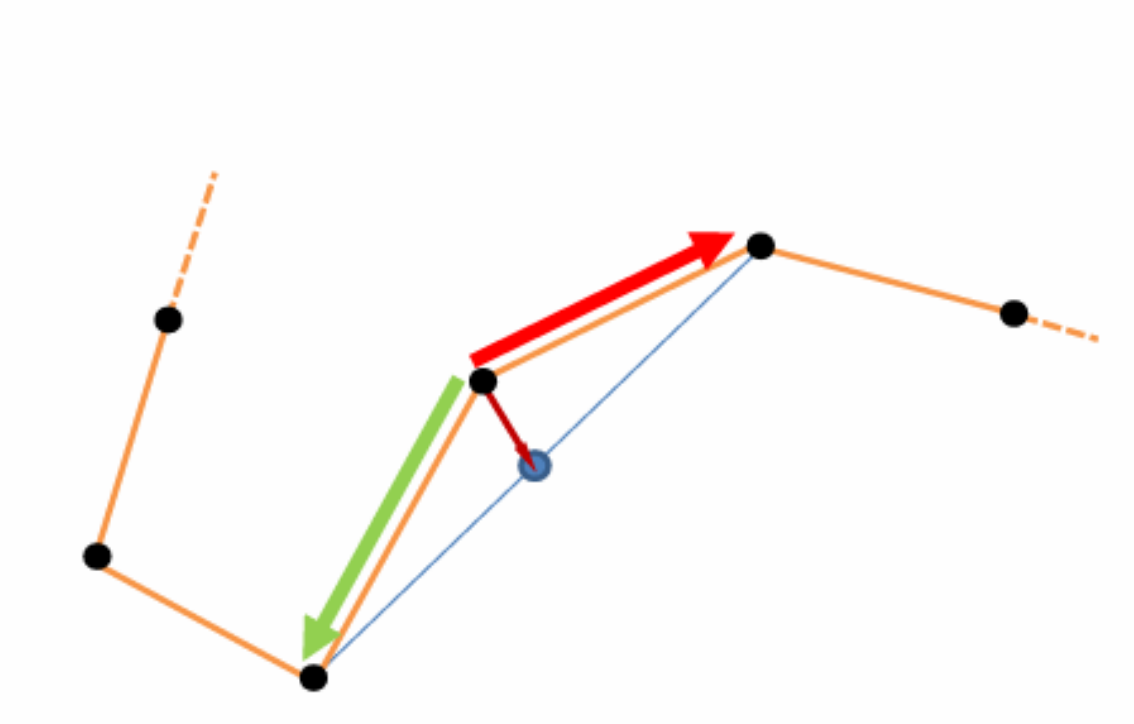


Figure 4.1: Finite difference approximation of laplacian of curve

So we can write in explicit form as

$$\gamma_{i+1} = \gamma_i + \delta t \frac{d^2\gamma_i}{ds^2} = \gamma_i + \delta t \Delta \gamma_i$$

The vector matrix formulation of the above equation is

$$\gamma_{\textbf{new}} = \gamma + \delta t \mathbf{L} \gamma$$

where

$$\mathbf{L} = \frac{1}{2}\begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & 1 & -2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & 1 \\ 0 & 0 & \cdots & 1 & -2 \end{bmatrix} \in R^{n \times n}$$

, a small time step δt and $\gamma \in R^{n \times 2}$

Algorithm 5 Laplacian Smoothing of Discrete Plane and Space Curve

Require: $\boldsymbol{\gamma} \in \mathbb{R}^{n\times 2}$ *or* $\mathbb{R}^{n\times 3}$ ▷ Discrete points of the plane and space curve
Require: $\delta t > 0$ ▷ Time step size
Require: $T \in \mathbb{N}$ ▷ Number of smoothing iterations
Ensure: $\boldsymbol{\gamma}$ ▷ Smoothed curve points

function Laplacian Smoothing($\boldsymbol{\gamma}, \delta t, T$)
  $n \leftarrow$ number of curve points
  Initialize $L \in \mathbb{R}^{n\times n}$ as a zero matrix ▷ Construct 1D Laplacian matrix
  for $i = 0$ to $n-1$ **do**
    $L[i,i] \leftarrow -2$
    if $i > 0$ **then**
      $L[i,i-1] \leftarrow 1$
    if $i < n-1$ **then**
      $L[i,i+1] \leftarrow 1$
  $L \leftarrow \frac{1}{2} \cdot L$ ▷ Iteratively apply Laplacian smoothing
  for $t = 1$ to T **do**
    $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma} + \delta t \cdot L \cdot \boldsymbol{\gamma}$
  return $\boldsymbol{\gamma}$

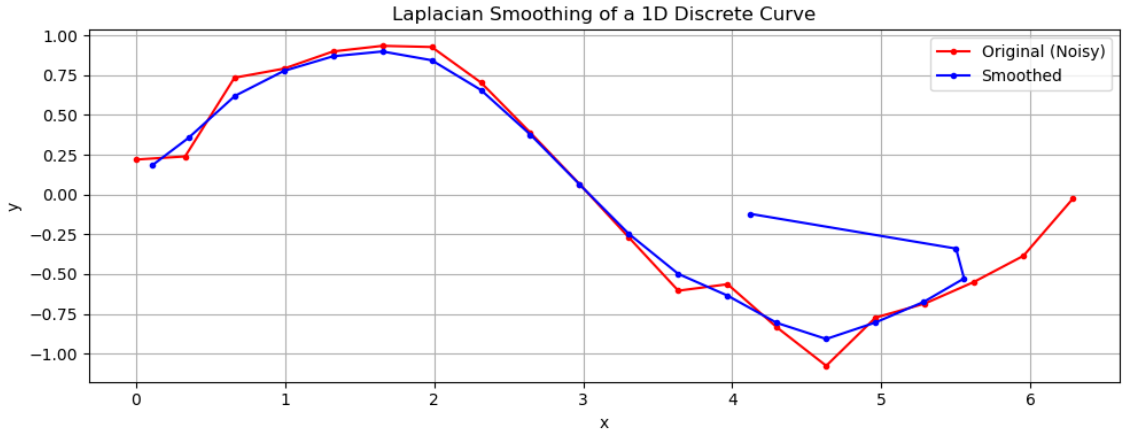


Figure 4.2: Smoothing of plane curve

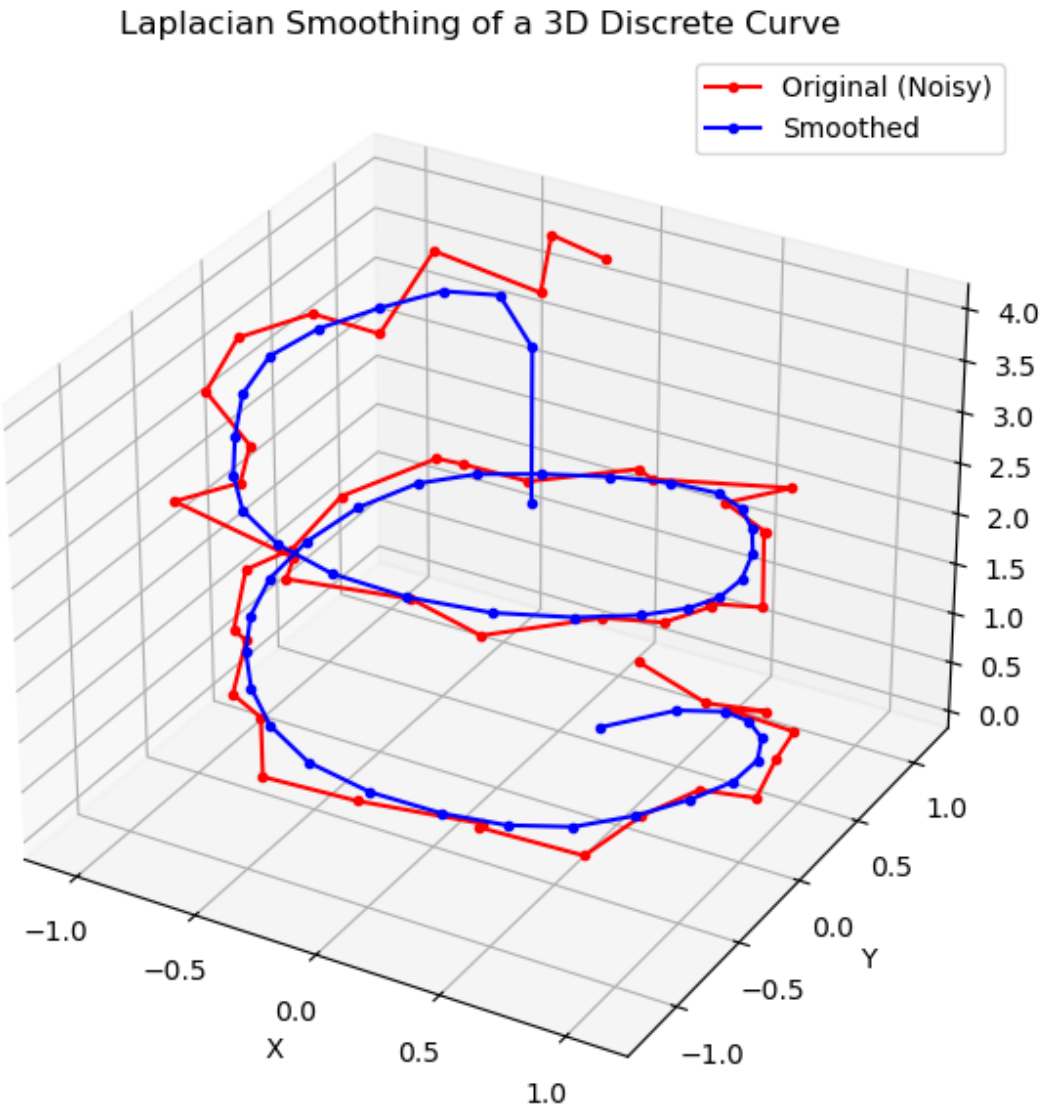


Figure 4.3: Smoothing of the space curve(helix)

4.1.2 Discrete Surface Smoothing

Discrete surface smoothing is concerned with design and computation of smooth functions $f : M \rightarrow R^n$ on a triangle mesh and these functions can be chosen to describe vertex position, texture coordinates or displacement etc. There are several techniques to perform the mesh smoothing can be classified as

We apply mathematical filters to the mesh geometry to remove the noise.

1. *Laplacian Smoothing* : We will use Laplace-Beltrami Operator to simulate heat diffusion on arbitrary mesh surfaces via averaging each vertex by its neighbours.

2. *Spectral Filtering* : Laplacian Beltrami Operator spectral decomposition provides information of spectrum of eigenfunctions and values of mesh. Filtering high frequency components and smoothing noise while maintaining mesh structure.

3. *Bilateral-Smoothing* : It is an edge-preserving filter that takes account of spatial closeness and geometry similarity to reduce noise while preserving important features like sharp edges.

Laplace Smoothing

Consider the heat equation

$$\frac{\partial u}{\partial t} = \Delta u \tag{4.1}$$

, with initial condition $u(x_0, t_0) = u_0$.

Now, our aim to find the function $u(x, t)$ where $u \in M$

We know that laplacian-beltrami operator L when applied on u can be written in discrete forms as

$$(Lu)_i = \frac{1}{2A_i} \sum_{j \in N(i)} (cot\alpha_{ij} + cot\beta_{ij})(u_j - u_i)$$

$$ALu_i = \frac{1}{2} \sum_{j \in N(i)} (cot\alpha_{ij} + cot\beta_{ij})(u_j - u_i)$$

Let take $A_i L = W$ and we know the $A_i = A$ area weight matrix. Therefore,

$$Wu_i = \frac{1}{2} \sum_{j \in N(i)} (cot\alpha_{ij} + cot\beta_{ij})(u_j - u_i)$$

and

$$AL = W \implies L = M^{-1}W \tag{4.2}$$

To solve the heat equation, we first discretize $u(x,t)$ using different schemes like Euler forward, Euler Backward etc.

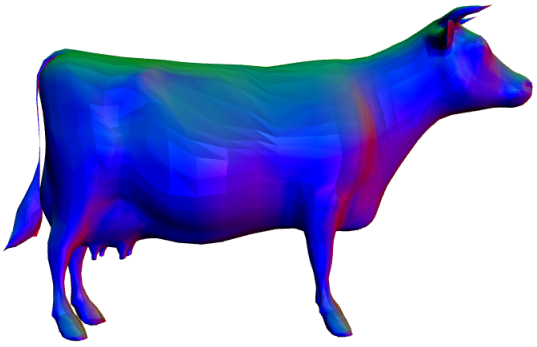


Figure 4.4: Laplacian smoothing by explicit method

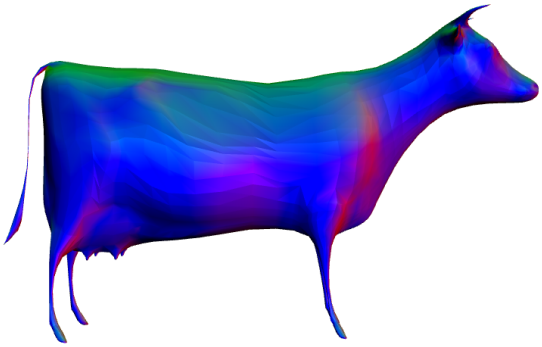


Figure 4.5: Laplacian smoothing by implcit method

Explicit Method by Euler-Forward Scheme :

$$\frac{u_t - u_0}{\delta t} = (Lu)_t$$

$$u_t = u_0 + \delta t L u_t$$

$$u_t = (I + \delta t L)u_0$$

let's put the value of L in equation we get

$$u_t = (I + \delta t A^{-1} W)u_0 \tag{4.3}$$

This equation can be solved using the Euler integration method.

Implicit Method by Euler-Backward Scheme : The heat equation can be written in discrete form using euler-backward scheme as

$$\frac{u_t - u_0}{\delta t} = (Lu)_t$$

$$u_t - u_0 = \delta t L u_t$$

$$(I - \delta t L)u_t = u_0$$

$$(I - \delta t A^{-1} W)u_t = u_0$$

$$(A - \delta t W)u_t = Au_0 \tag{4.4}$$

, where $A \in R^{n \times n}$ be mass matrix, $W \in R^{n \times n}$ be laplacian, $t > 0$ be diffusion time, $u_0 \in R^n$ be initial function and $u_t \in R^n$ be smoothed function.

Now the system of equations are symmetric and we can use the cholesky factorization to solve it.

Spectral Smoothing

Consider the discrete diffusion equation implicit form

$$(A + \delta t W)u_t = Au_0$$

and take the laplacian operator L and build a generalized eigenvalue problem

$$L\phi_j = \lambda_j A \phi_j$$

Now, let's collect the first k eigen-functions $\{\phi_1, \ldots, \phi_k\}$ into a matrix $\Phi \in \mathbb{R}^{n \times k}$, and their corresponding eigenvalues $\{\lambda_1, \ldots, \lambda_k\}$ into a diagonal matrix $\Lambda \in \mathbb{R}^{k \times k}$ such that

$$L\Phi = A\Phi\Lambda$$

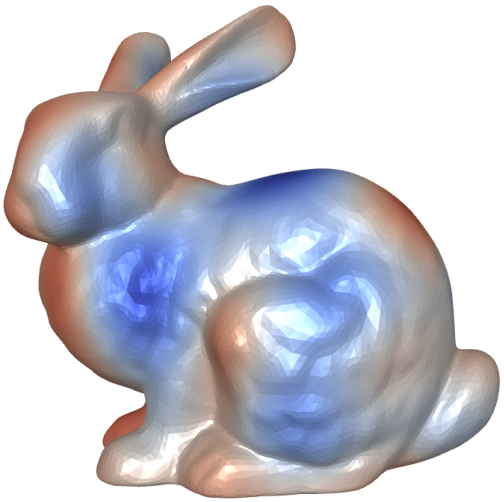


Figure 4.6: Eigenfunction of laplacian

$$\Phi^\top (A + \delta t W)\Phi = I_k + t\Lambda \tag{4.5}$$

How to reconstruct the function u Assume the solution u_t lies in the subspace spanned by the first k eigen-functions

$$u_t = \Phi \alpha_t$$

for some coefficients $\alpha_t \in \mathbb{R}^k$. Substitute into the diffusion equation

$$(A + \delta W)(\Phi \alpha_t) = A u_0$$

$$\Phi^\top (A + \delta W)\Phi(\alpha_t) = A u_0$$

from equation 4.5

$$I_k + t\Lambda \alpha_t = \Phi^\top M u_0$$

Let's $\beta = \Phi^\top M u_0 \in R^k$ which is projection of u_0 onto the eigenfunction basis such that

$$\alpha_t = (I_k + \delta t\Lambda)^{-1}\beta$$

Finally, reconstructed u is

$$u_t = \Phi\alpha_t = \Phi(I_k + t\Lambda)^{-1}\Phi^\top A u_0$$

where

$$(\alpha_t)_j = \frac{\beta_j}{1 + \lambda_j}$$

which requires only element wise multiplication.

Practically, The heat equation

$$\frac{\partial u_t}{\partial t} = \Delta u_t$$

. Consider $u_t = \Phi\alpha_t$ then $\Delta\Phi = -\Lambda\Phi$ Thus

$$\frac{\partial u_t}{\partial t} = \Delta u_t = -\Lambda\alpha_t$$

By solving this equation we will get

$$\alpha_t = \beta exp(-\Lambda t)$$

and element wise will be

$$(\alpha_t)_j = \beta_j exp(-\lambda_j t)$$

.

Algorithm 6 Spectral Smoothing of Function f on Mesh

Require: Function $f \in \mathbb{R}^n$, mass matrix $A \in \mathbb{R}^{n\times n}$, Laplacian $W \in \mathbb{R}^{n\times n}$, number of modes K, smoothing parameter t
Ensure: Smoothed function $f_t \in \mathbb{R}^n$

1: **Compute generalized eigen-decomposition:**

$$W\Phi = A\Phi\Lambda \quad \text{where } \Phi \in \mathbb{R}^{n\times K},\ \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_K)$$

2: **Project input function onto spectral basis:**

$$\beta = \Phi^\top A f \in \mathbb{R}^K$$

3: **Apply spectral filtering (diffusion decay):**

$$\alpha_t = e^{-t\Lambda}\beta \quad \text{where } (e^{-t\Lambda})_j = e^{-t\lambda_j}$$

4: **Reconstruct smoothed function:**

$$f_t = \Phi\alpha_t$$

5: **return** f_t

Optimization-based methods

We define an objective(energy) function that balances smoothness with original geometry.

L_0 *Smoothing* : The mesh model may consist of flat regions, then we perform mesh denoising by L_0 minimization, which maximizes flat regions and gradually removes noise while preserving sharp features.

Energy Optimization Consider an energy function $E : M \to R$ defined as

$$E(u) = E_{smooth}(u) + E_{error}(u)$$

, Where $E_{smooth}(u)$ smoothness energy of surface and $E_{error}(u)$ fidelity energy is the error to the original surface.

Smoothing of the surface can be modelled as an optimization problem via minimizing the energy function that balances the smoothness energy and keeping the smooth surface close to the original surface.

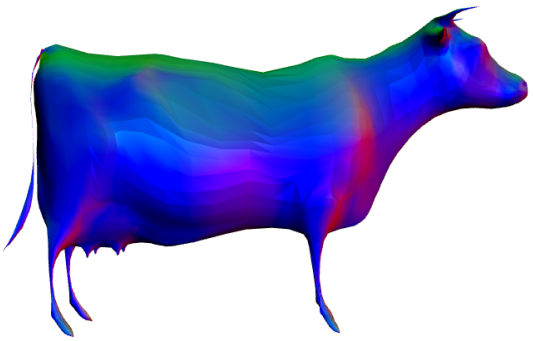


Figure 4.7: Energy optimization method for surface smoothing

$$\min_{u} E(\mathbf{u}) = E_{\text{smooth}}(\mathbf{u}) + E_{\text{error}}(\mathbf{u}) \tag{4.6}$$

Now, minimizing smooth energy of the surface can be seen as reducing the curvature variation using laplacian across the surface.

$$\Delta_M u = -2H \cdot n$$

. Thus the goal is to make $H = 0$ then the surface is called minimal(developable) surface or $H = constant$. Suppose that we are only minimizing the H then it can lead to $u = 0$ which collapses the geometry of the surface. Therefore, we will apply regularizer then energy functional will become

$$\min_{\tilde{u}} E(\tilde{\mathbf{u}}) = \min_{\tilde{u}} \int_{\mathcal{M}} \|\Delta_{\mathcal{M}} \tilde{\mathbf{u}}\|^2 \, dA + w\|\tilde{\mathbf{u}} - \mathbf{u}\|^2 \tag{4.7}$$

Where, $E_{\text{smooth}}(\tilde{\mathbf{u}}) = \|\Delta_{\mathcal{M}} \tilde{\mathbf{u}}\|^2$ encourages smoothness (small curvature$H$) and $E_{\text{error}}(\tilde{\mathbf{u}}) = \|\tilde{\mathbf{u}} - \mathbf{u}\|^2$ penalizes deviation from original surface and w is a tuning parameter that balances deviation from original surface and smoothing.

Let's discretize the energy functional using mesh vertices and area

$$\min_{\tilde{u}} E(\tilde{\mathbf{u}}) = \min_{\tilde{\mathbf{u}}} \sum_{i=1}^{n} A_i \left(\|L\tilde{\mathbf{u}}_i\|^2 + w\|\tilde{\mathbf{u}}_i - \mathbf{u}_i\|^2 \right)$$

where, L is the laplacian-beltrami operator, $\mathbf{u}_i \in \mathbb{R}^3$ vertex original position and $\tilde{\mathbf{u}}_i \in \mathbb{R}^3$

vertex optimized position. So, matrix representation of

$$\tilde{\mathbf{u}} = \begin{bmatrix} \tilde{u}_{1x} & \tilde{u}_{1y} & \tilde{u}_{1z} \\ \tilde{u}_{2x} & \tilde{u}_{2y} & \tilde{u}_{2z} \\ \vdots & \vdots & \vdots \\ \tilde{u}_{nx} & \tilde{u}_{ny} & \tilde{u}_{nz} \end{bmatrix} \in \mathbb{R}^{n\times 3}$$

Then energy functional will be

$$E(\tilde{\mathbf{u}}) = \sum_{u\in\{x,y,z\}} (L\tilde{\mathbf{u}})^T M(L\tilde{\mathbf{u}}) + w(\tilde{\mathbf{u}} - \mathbf{u})^T M(\tilde{\mathbf{u}} - \mathbf{u}) \tag{4.8}$$

where, $L \in \mathbb{R}^{n\times n}$: laplacian matrix, $M \in \mathbb{R}^{n\times n}$ is mass matrix and $\tilde{\mathbf{u}}, \mathbf{u} \in \mathbb{R}^n$ vertex coordinates.

$$\frac{\partial E}{\partial \tilde{\mathbf{u}}} = 2L^T ML\tilde{\mathbf{u}} + 2wM(\tilde{\mathbf{u}} - \mathbf{u}) = 0$$

$$ccccccc \tag{4.9}$$

Solving this systems to obtain the min value of u.

4.2 Simulation of Heat Equation on Mesh

One of the most common applications of the Laplace-Beltrami operator in geometry processing is simulating heat diffusion over arbitrary surfaces. The continuous heat equation is given by

$$\frac{\partial u}{\partial t} = \Delta u,$$

where u is a function defined on the surface, and Δ is the Laplace-Beltrami operator.

Given an initial heat distribution u_0, the heat distribution at a later time t, denoted u_t, can be computed by discretizing the equation. Using a single backward Euler time step, we arrive at

$$(I - t\Delta)\, u_t = u_0,$$

where I is the identity operator and t is the time step.

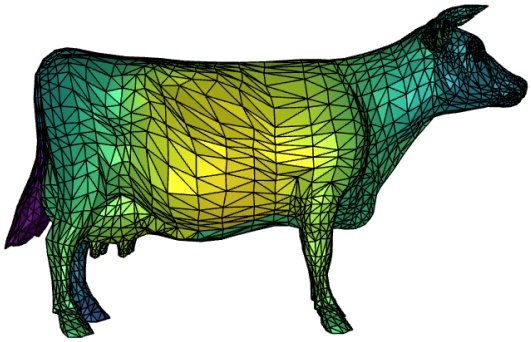


Figure 4.8: Simulation of heat equation for f(x) = sin(x)

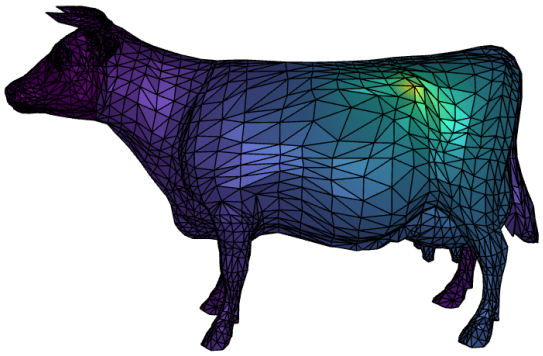


Figure 4.9: Simulation for $f(x) = \delta(x)$

In discrete differential geometry, the Laplacian matrix L used in computations is typically positive semidefinite, while the mathematical Laplacian Δ is negative semidefinite. In practice, we use $L \approx -\Delta$. To account for this, we substitute Δ with $-L$ in the equation above. Additionally, we multiply both sides by the mass matrix A (which encodes area weights for the mesh vertices), resulting in

$$(A + tW)\, u_t = Au_0,$$

where W is the stiffness matrix (the discrete Laplacian), and A is the diagonal mass matrix.

This equation forms a sparse linear system because both A and W are sparse matrices—most entries are zero, reflecting the local connectivity of the mesh. Although the system has as many equations as there are vertices (which can be very large), the sparsity allows for highly efficient solution methods that are much faster than those for dense systems.

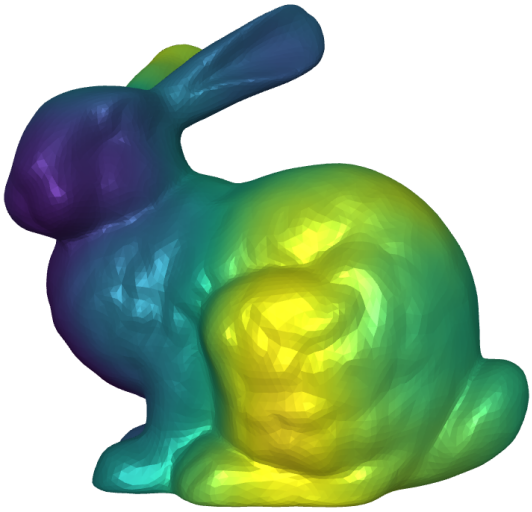


Figure 4.10: Simulation of heat equation on bunny mesh

Importantly, solving this system does not require explicitly computing the inverse of $(A + tW)$; instead, modern numerical solvers can efficiently find u_t directly.

4.3 Geodesic Computation on Discrete Surface

We know that the geodesic on the surface is the shortest path between any two points on it. For example AIR India flights wanted to travel from New Delhi to Canada then what will be the shortest path required to travel from departure to reach the destination.

4.3.1 Approximate Geodesic Computation

Triangle meshes are composed of vertices, edges, and faces. When computing approximate geodesics, these meshes are commonly abstracted as graphs, where vertices become graph nodes and edges connect nodes and are weighted by their euclidean lengths. Thus, This graph representation allows us to use classical shortest-path algorithms, such as Dijkstra's algorithm, to compute geodesic distances.

Dijkstra's algorithm is used to find the shortest path from a source vertex to all other vertices along edges of the graph and it's faster and easy to implement, because it restricts the paths to mesh edges therefore only gives an approximation of true geodesic distance.

The true geodesic may go through the cut across the triangle interior, not only edges. So, to

improve the approximation we needed to consider the distance propagation within triangle faces which is called a fast marching algorithm.

4.3.2 Heat Method for Geodesic Computation

The geodesic path computation across the manifold mesh, uses the diffusion of the dirac delta δ_x function.

Given a point$x$, we start from a dirac function δ_x centered on x, then heat equation(implicit form)

$$(M - \delta t L_c)u_{i+1} = M\delta_x \tag{4.10}$$

It diffuses alot heat to the point close to the x and the amount of heat decreases as we move away from the point x. So, we need to evaluate the vector field X using formula

$$X = -\frac{\nabla u_t}{||\nabla u_t||}$$

which is opposite the normalized direction of gradient of diffusion(going geodesically towards x.

Now, solving the poisson equation

$$\Delta\phi = -\nabla \cdot X = \nabla \cdot \frac{\nabla u_t}{||\nabla u_t||} = \nabla^2 u_t$$

. We obtain our geodesic paths ϕ and x is the starting point therefore we put a constraint distance to x is zero.

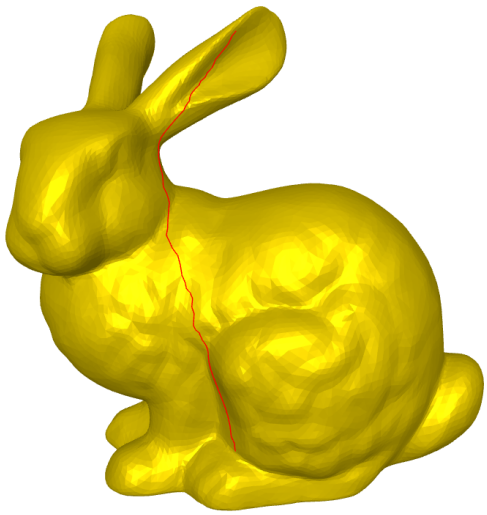


Figure 4.11: Front : Fast marching algorithm

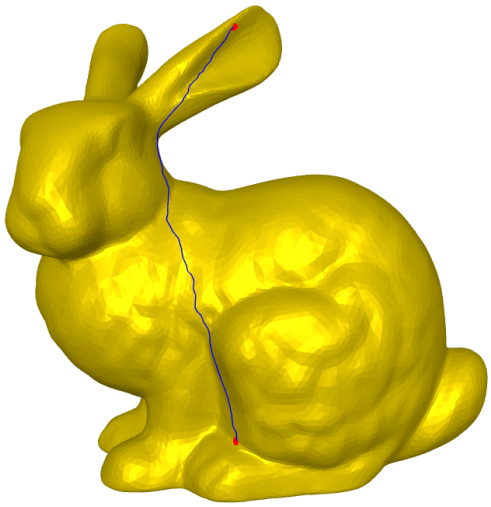


Figure 4.12: Back: Dijkstra's shortest path algorithm

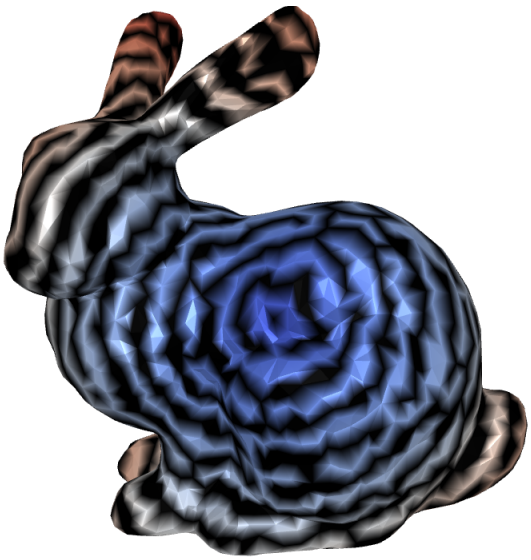


Figure 4.13: Front : Geodesic

Figure 4.14: Back: Geodesic

Algorithm 7 Geodesic Distance via Heat Method

Require: Mesh with Laplace matrix $W \in \mathbb{R}^{n \times n}$, mass matrix $A \in \mathbb{R}^{n \times n}$, source point index x, time step $t = h^2$

Ensure: Approximate geodesic distance $\varphi \in \mathbb{R}^n$ from source point x

1: Initialize Dirac delta: $\delta_x \in \mathbb{R}^n$, where $(\delta_x)_i = \begin{cases} 1 & \text{if } i = x \\ 0 & \text{otherwise} \end{cases}$

2: **Solve heat diffusion:**

$$(A + tW)u_t = A\delta_x$$

3: **Compute normalized gradient field:**

$$X = -\frac{\nabla u_t}{\|\nabla u_t\|}$$

4: **Solve Poisson equation:**

$$W\varphi = A(\nabla \cdot X)$$

5: **Normalize distance:**

$$\varphi \leftarrow \varphi - \varphi(x)$$

6: **return** φ

Chapter 5

Conclusion

This dissertation has presented a comprehensive study of the *discrete differential geometry(DDG)* and its applications in shape modeling and geometric processing. The main objective was to develop the discrete analogues of the smooth differential geometric concepts and creation of a mathematical-cum-computational framework for analysis and manipulation of the curves and surfaces through discretization. Rather than discretizing the equations of smooth differential geometry, the focus was on discretizing the geometry itself. Graphs were used to represent discrete plane and space curves, while triangle meshes served as discrete surfaces, embedded in space to form piecewise linear surfaces.

5.1 Summary of Contributions

The dissertation begins with discrete curves, introducing key geometric quantities such as tangent, normal, curvature, and torsion in their discrete formulations. These definitions enabled the discretization the *fundamental theorem of the curve in plane and space* and facilitated the design algorithm for curve reconstruction that preserves the intrinsic structure of discrete curves.

After discrete curves work progressed to the discrete surfaces, where defined triangle mesh were discrete surface and embedding of discrete surface into space as piecewise linear surface. The topological and geometric properties of these surfaces were analysed with

detailed discussion of discrete differential, tangent space, normal and immersion etc.

The discrete curvatures including geodesic, gaussian, mean and principle were computed and the concept of the diflect angle, to compute *gaussian curvature*. The discrete Gauss-Bonnet theorem was stated and proven, which highlights the deep connections between discrete and smooth geometry.

Particular attention was given to *Laplacian-Beltrami Operator*, which is a cornerstone of modern discrete geometric processing. Two primary discretization approaches were discussed for it.

1. **Finite Element Method(FEM)** : Widely used in scientific and engineering applications for solving partial differential equations.
2. **Discrete Exterior Calculus(DEC)** : specifically used for most of our discretization tasks.

The final part of the dissertation focused on practical application of discrete differential geometry based on the idea of *Laplacian beltrami operator*.

1. **Smoothing** of discrete curves and discrete surface
2. **Simulation** of heat equation on mesh
3. **Geodesics** on the discrete surface

5.2 Future Research Direction

Discrete differential geometry remains an active and rapidly evolving field of research. It presents numerous open challenges to both theoretical and applied research and its real life applications span a wide range of industries, from manufacturing to entertainment.

1. **Development of new discretization methods** : They better preserve the geometric quantities and topological properties of smooth geometric objects such as invariance under transformation remains a central and open problem.[2] The different discretizations can preserve different properties and understanding these trade offs is the key for reliable

geometric computation.

2. **Mathematical Analysis of Discrete Models** : There are fundamental questions regarding the rigorous analysis of convergence rates and stability of discrete models as they approximate their smooth counterparts. This includes studying how discrete curvatures, geometric flows, and invariants behave under mesh refinement, which is essential for ensuring the reliability of discrete geometric algorithms.[2]

3. **Discrete Conformal and Holomorphic Mappings** : The applications of discrete conformal mappings and discrete holomorphic functions—especially for triangle and quadrilateral meshes—remains a vibrant area. Progress here has direct implications for mesh parameterization, texture mapping, and computer graphics.[6]

4. **Data-Driven Geometric Processing** : Integrating geometric processing with deep learning models operating on graphs, meshes, and point clouds enables the extraction and learning of complex geometric features beyond the reach of traditional methods. Research in this direction can lead to advances in shape classification, correspondence, parameterization, and generative modeling, pushing the boundaries of what is possible in geometric data analysis.

The dissertation has contributed to the theoretical foundation of discrete differential geometry and practical applications like surface smoothing, scientific simulation and geodesic computation in geometric processing. As computational geometry continues to intersect with the modern data driven methods then discrete differential geometry undoubtedly will play a pivotal role in shaping the future of geometric modeling, analysis and simulation.

References

[1] Alexander I Bobenko and Boris A Springborn. "A discrete Laplace–Beltrami operator for simplicial surfaces". In: *Discrete & Computational Geometry* 38.4 (2007), pp. 740–756.

[2] Alexander I. Bobenko, ed. *Advances in Discrete Differential Geometry*. 1st ed. Springer Berlin, Heidelberg, 2016. isbn: 978-3-662-50446-8. doi: 10.1007/978-3-662-50447-5. url: https://doi.org/10.1007/978-3-662-50447-5.

[3] Alexander I. Bobenko et al., eds. *Discrete Differential Geometry*. Vol. 38. Oberwolfach Seminars. Basel: Birkhäuser Basel, 2008. isbn: 978-3-7643-8620-7. doi: 10.1007/978-3-7643-8621-4. url: https://doi.org/10.1007/978-3-7643-8621-4.

[4] Albert Chern. *Discrete Differential Geometry*. Lecture Notes, Computer Science and Engineering, University of California San Diego. Last update: November 19, 2020. 2020. url: https://cseweb.ucsd.edu/~alchern/teaching/DDG.pdf.

[5] Keenan Crane. *Discrete Differential Geometry : Lecture Slides*. https://brickisland.net/ddg-web/. Accessed: 2025-05-25. 2020.

[6] Keenan Crane. *Discrete Differential Geometry: An Applied Introduction*. Lecture Notes / Draft Textbook, Carnegie Mellon University; Last updated: January 29, 2025. 2025. url: https://www.cs.cmu.edu/~kmcrane/Projects/DDG/paper.pdf (visited on 05/20/2025).

[7] Keenan Crane and Max Wardetzky. "A glimpse into discrete differential geometry". In: *Notices of the American Mathematical Society* 64.10 (2017).

[8] Alec Jacobson et al. *libigl/libigl-python-bindings: libigl python bindings*. GitHub repository. Accessed: 2025-05-20. 2024. url: https://github.com/libigl/libigl-python-bindings.

[9] Alec Jacobson et al. "libigl: A simple C++ geometry processing library". In: *Google Scholar* (2013). url: https://github.com/libigl/libigl.

[10] Mark Meyer et al. "Discrete differential-geometry operators for triangulated 2-manifolds". In: *Visualization and mathematics III*. Springer. 2003, pp. 35–57.

[11] Ulrich Pinkall and Oliver Gross. *Differential Geometry: From Elastic Curves to Willmore Surfaces*. Springer Nature, 2024.