Observe, Visualize, and Analyze

(curves, surfaces, normal, curvature)

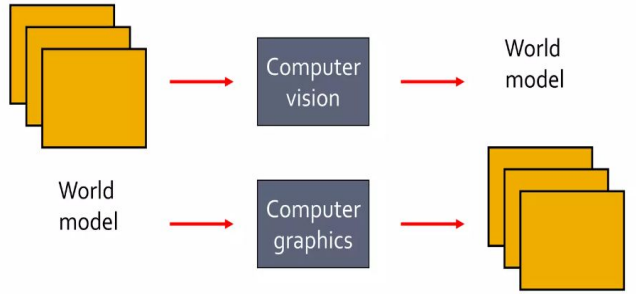
Date: 8 / 27 /2023

Speaker: Seungho Jang

Contents

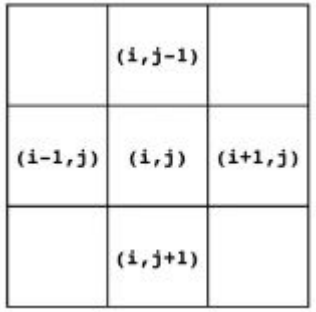
Preliminary Talks

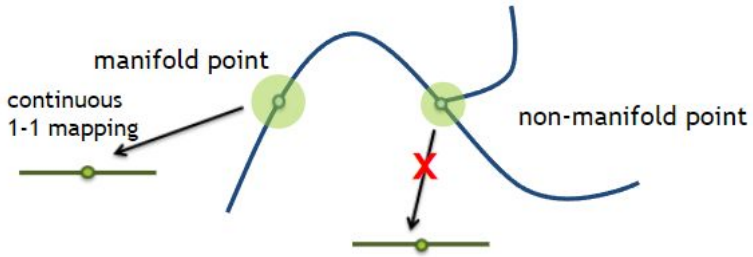
- **Common**
  - **Computer Graphics vs Computer Vision (Inverse Relationship)**
  - **Data Structure Motives** (Connectivity) → **How to represent the surface?**
  - **Geometry Analysis Tools**
    - **Derivatives, Curvature, and etc…)**
    - **Homogeneous Coordinate (Next…)**

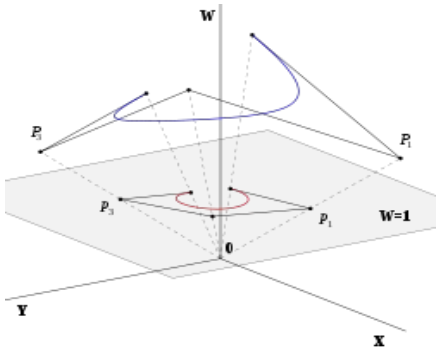


Why did we choose a square grid?

- simplicity / efficiency (Easy to index)
- always have four neighbors
- Generality
  - encode basically any image
- **Cons ?**
  - an **anisotropic sampling** image (vertical / horizontal direction)





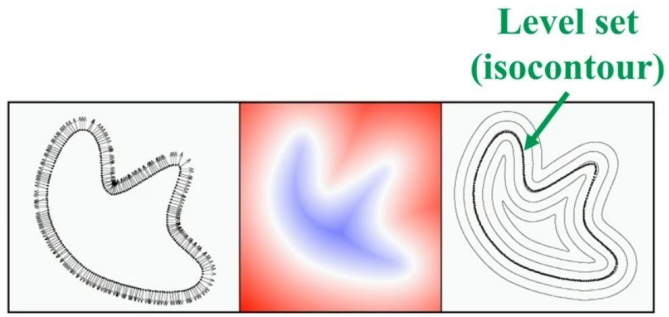


https://www.slideshare.net/KartikKalpande/computer-vision-64112850

How to Represent Shapes ?

- **Parametric Geometry → Mathematical Models**
- **Implicit Models → Distance Functions**
  - **Algebraic Surfaces**
  - **Level Set Methods**
  - **Fractals**
  - **Constructive Solid Geometry**
  - **…**
- **Explicit Models (Representing the Surfaces …)**
  - **Point Cloud**
  - **Polygon Meshes**
  - **Triangle Meshes**
  - **Subdivision, NURBS**
  - **…**

Alternative: Implicit Models

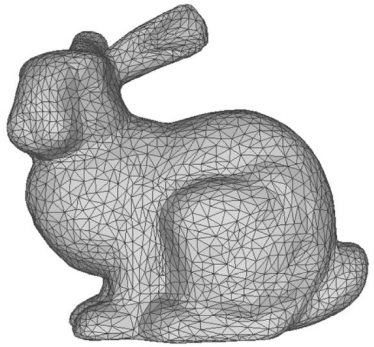


$$\{(x, y) \text{ such that } f(x, y) = 0\}$$

Point cloud

Voxel

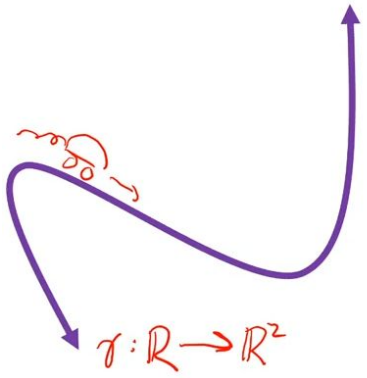
Polygon mesh



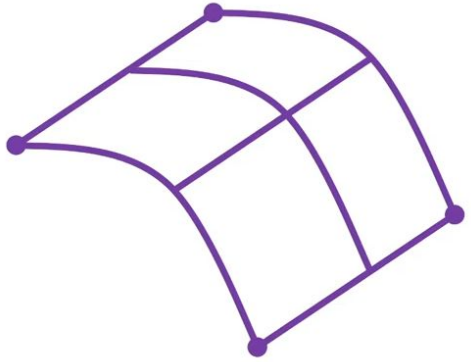
Parametrized Geometry (Curves & Surfaces)

- **DEFINITION: A parametrized differentiable curve is a differentiable map a : I → R3 of an open interval I = (a, b) of the real line R into R3.**

Parametric Geometry

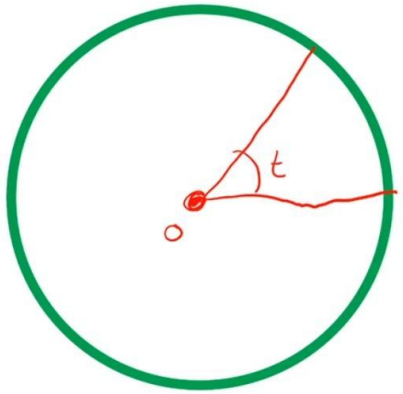


$\vec{\gamma}(t) = (\gamma_x(t), \gamma_y(t))$



$\vec{\beta}(u, v) = (\beta_x(u, v), \beta_y(u, v), \beta_z(u, v))$

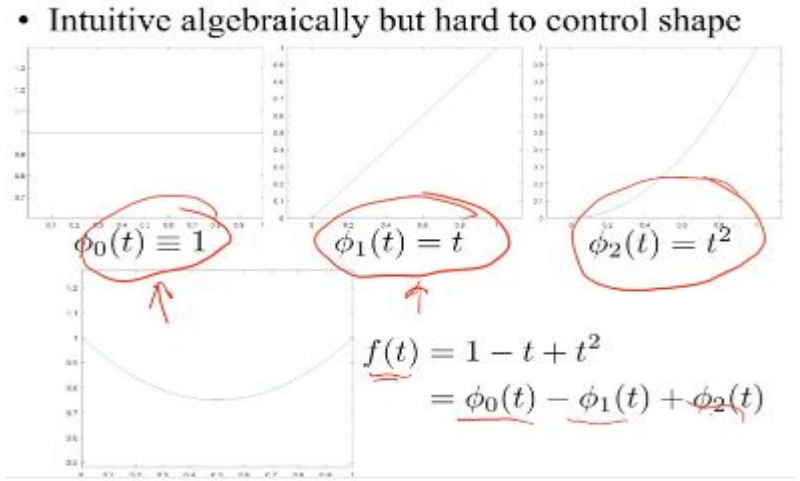
Parameterized Circle

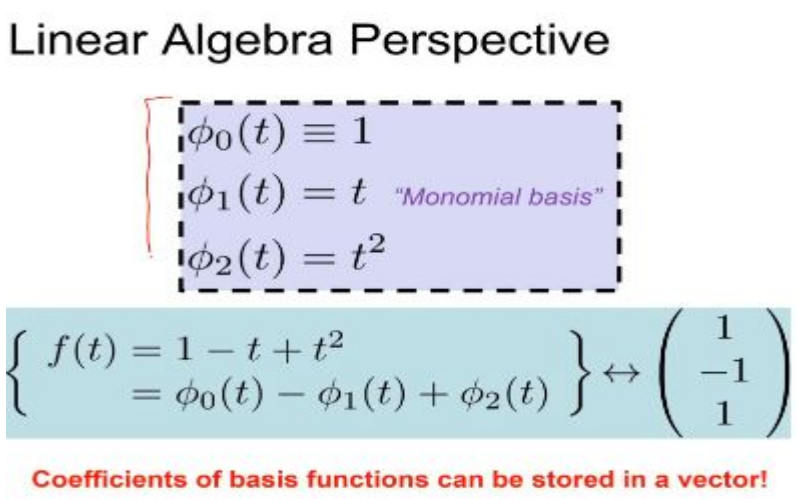


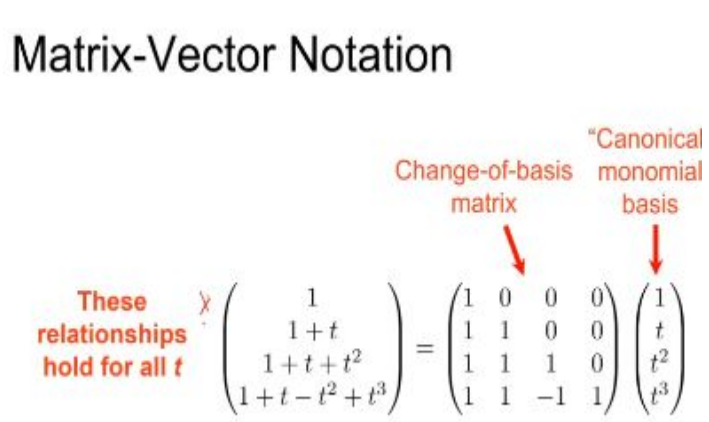
$$\vec{\gamma}(t) = (\cos t, \sin t)$$

Parameterized Curve (Monomial Basis)

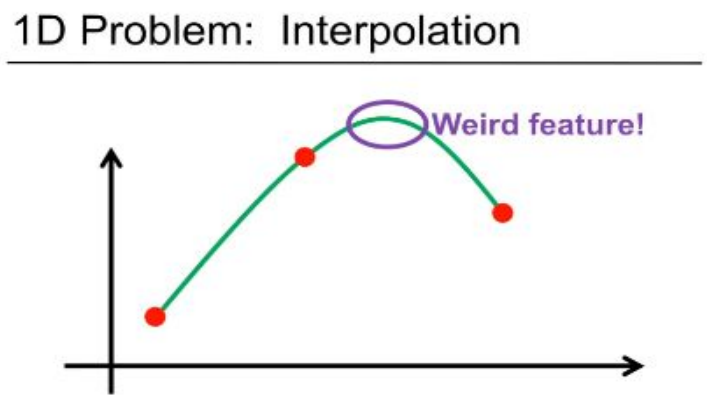
- **Monominal Basis**







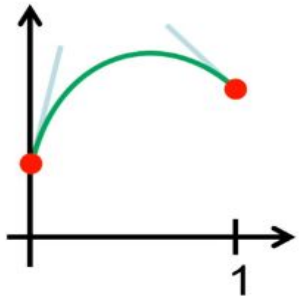
- **Monominal Basis Problem**



Parameterized Curve (Hermite Basis)

- **Hermite Basis**

Cubic Hermite Interpolation:
Monomial Basis



$$P(t) = at^3 + bt^2 + ct + d$$
$$P'(t) = 3at^2 + 2bt + c$$

$$P(0) = d = h_0$$
$$P(1) = a + b + c + d = h_1$$
$$P'(0) = c = h_2$$
$$P'(1) = 3a + 2b + c = h_3$$

Matrix Representation

$$P(0) = d = h_0$$
$$P(1) = a + b + c + d = h_1$$
$$P'(0) = c = h_2$$
$$P'(1) = 3a + 2b + c = h_3$$

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{pmatrix}$$

Matrix Representation

Invert matrix

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{pmatrix}$$

$$\begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$$

Two Bases, One Curve

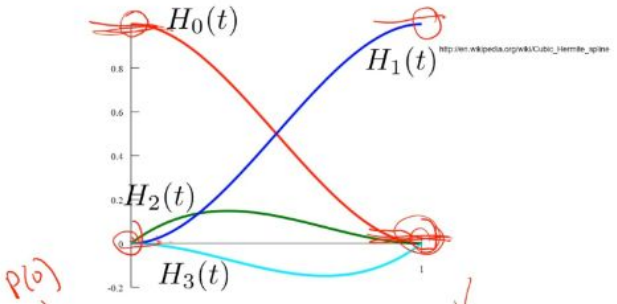
$$P(t) = at^3 + bt^2 + ct + d$$ Monomials

$$= (2h_0 - 2h_1 + h_2 + h_3)t^3 + (-3h_0 + 3h_1 - 2h_2 - h_3)t^2 + h_2 t + h_0$$ Previous slide

Cubic Hermite

$$= h_0(2t^3 - 3t^2 + 1) + h_1(-2t^3 + 3t^2) + h_2(t^3 - 2t^2 + t) + h_3(t^3 - t^2)$$

Hermite Basis



P(0)

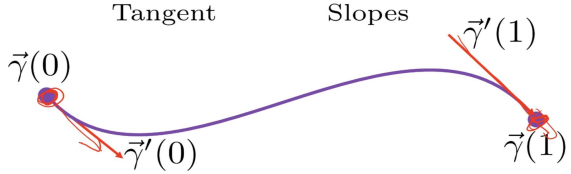
$$= h_0(2t^3 - 3t^2 + 1) + h_1(-2t^3 + 3t^2) + h_2(t^3 - 2t^2 + t) + h_3(t^3 - t^2)$$

$$P(t) = P(0)H_0(t) + P(1)H_1(t) + P'(0)H_2(t) + P'(1)H_3(t)$$

Hermite Curves

$$\vec{\gamma}(t) = (\gamma_0(t), \gamma_1(t))$$

$$\underbrace{\vec{\gamma}'(t)}_{\text{Tangent}} = \underbrace{(\gamma_0'(t), \gamma_1'(t))}_{\text{Slopes}}$$



Parameterized Curve (Cubic Blossom & Cubic Control Points)

- **4 Control Points Cubic Polynomial Model (Cubic Blossom)**

Detour: Cubic Blossom

$f(t)$ cubic polynomial $\mapsto F(t_1, t_2, t_3)$

- **Symmetric**

$$F(t_1, t_2, t_3) = F(t_1, t_3, t_2) = F(t_2, t_1, t_3) = \cdots$$

- **Affine**

$$F(\alpha u + (1-\alpha)v, t_2, t_3) = \alpha F(u, t_2, t_3) + (1-\alpha)F(v, t_2, t_3)$$

- **Diagonal**

$$f(t) = F(t, t, t)$$

Ramshaw, Lyle (1987). "Blossoming: A Connect-the-Dots Approach to Splines." Digital Systems Research Center.

Blossom of a Cubic Curve

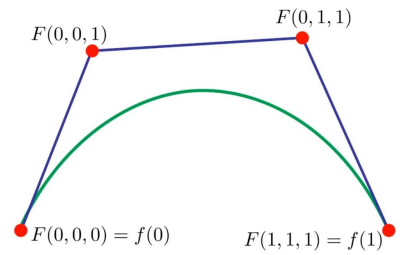
- Blossom each coordinate function separately

$$\vec{F}(t_1, t_2, t_3)$$

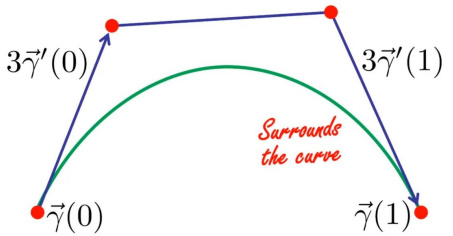
- (At least) two ways to obtain
  - Start with a cubic curve and compute its blossom
  - Specify four points

$$\vec{F}(0,0,0), \vec{F}(0,0,1), \vec{F}(0,1,1), \vec{F}(1,1,1)$$

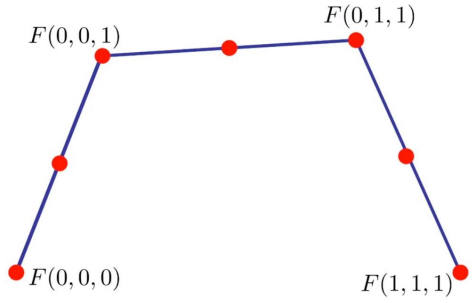
Control Polygon



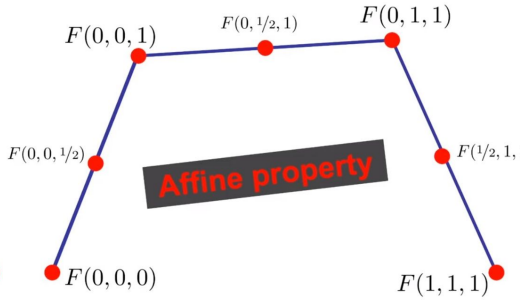
Simple Calculation



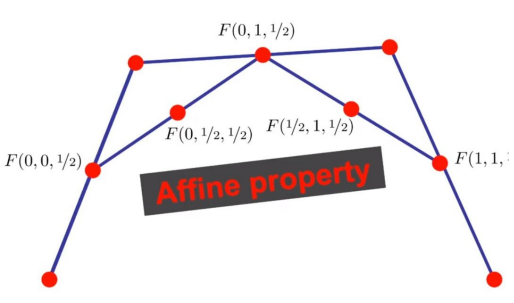
Cubic Control Polygon



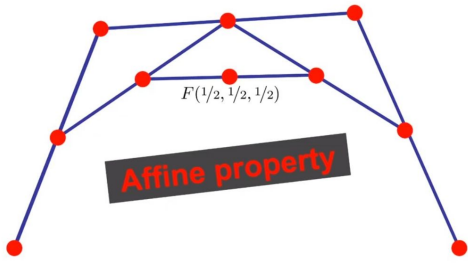
Cubic Control Polygon



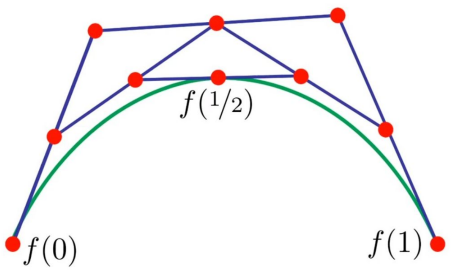
Cubic Control Polygon



Cubic Control Polygon



Cubic Control Polygon

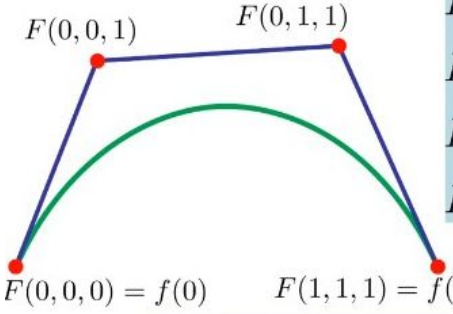


De Casteljau's Algorithm

Parameterized Curve (Bernstein Basis)

- **Alternative Cubic Basis (Bernstein Polynomial) → 4 Points**

Bernstein Polynomials



$$B_0(t) = (1-t)^3$$
$$B_1(t) = 3t(1-t)^2$$
$$B_2(t) = 3t^2(1-t)$$
$$B_3(t) = t^3$$

$$f(t) = F(0,0,0)B_0(t) + F(0,0,1)B_1(t) + F(0,1,1)B_2(t) + F(1,1,1)B_3(t)$$

Change of Basis

- How do we go from Bernstein basis to the canonical monomial basis $1, t, t^2, t^3$ and back?
  - With a matrix!

- $B_0(t)=(1-t)^3$
- $B_1(t)=3t(1-t)^2$
- $B_2(t)=3t^2(1-t)$
- $B_3(t)=t^3$

$$\begin{pmatrix} B_0(t) \\ B_1(t) \\ B_2(t) \\ B_3(t) \end{pmatrix} = \begin{pmatrix} \quad \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$$

How You Get the Matrix

Cubic Bernstein:

- $B_0(t)=(1-t)^3=1-3t+3t^2-t^3$
- $B_1(t)=3t(1-t)^2=3t-6t^2+3t^3$
- $B_2(t)=3t^2(1-t)=3t^2-3t^3$
- $B_3(t)=t^3$

Expand these out and collect powers of t. The coefficients are the entries in the matrix B!

$$\begin{pmatrix} B_0(t) \\ B_1(t) \\ B_2(t) \\ B_3(t) \end{pmatrix} = \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$$

Cubic Bézier in Matrix Notation

point on curve (2x1 vector)

$$P(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} =$$

Canonical monomial basis

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{pmatrix} \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$$

"Geometry matrix" of control points P_1, P_4

"Spline matrix"

General Spline Formulation

$\vec{\gamma}(t) =$ Geometry $G \cdot$ Spline basis $B \cdot$ Monomial basis $T(t)$

- **Geometry:** control points coordinates assembled into a matrix
- **Spline basis:** defines the type of spline
- **Monomial basis:** $(1, t, t^2, \ldots, t^n)$
- Advantages of general formulation
  - Compact expression
  - Easy to convert between types of splines

Spline (tangent & curvature vector)

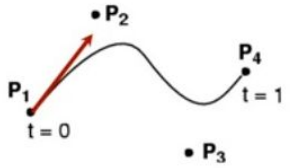
- **A piecewise polynomial with a high degree of smoothness where pieces meet. (glueing curves)**

Tangent Vector

- The tangent to the curve P(t) is the unit vector parallel to P'

$$T(t) := P'(t)/\|P'(t)\|$$

  - Defined so that ||T(t)|| = 1



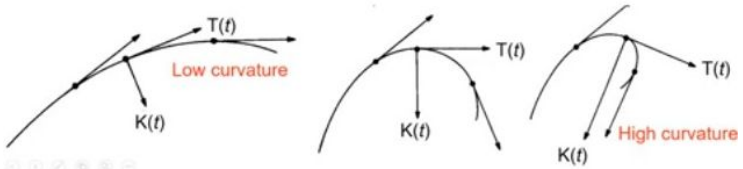
Regular curve: Curve for which P'(t) is always nonzero

Curvature Vector

- Derivative of unit tangent *(w.r.t. arc length)*

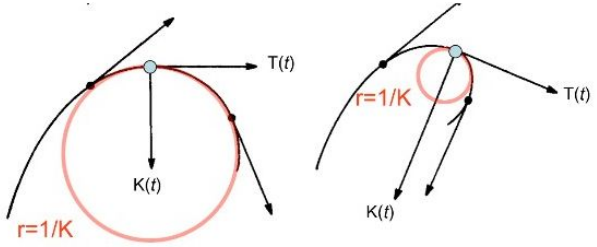
$$K(t) := \dot{T}(t) \propto T'(t)$$

  - Magnitude ||K(t)|| is constant for a circle
  - Zero for a straight line
- Always orthogonal to tangent, i.e. $K \cdot T \equiv 0$



Geometric Interpretation

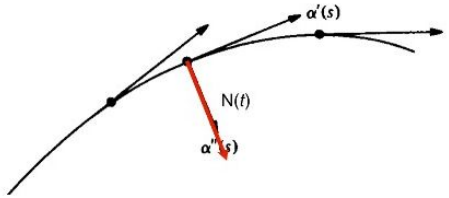
- 1/||K(t)|| is the radius of the circle that touches P(t) at t and has the same curvature as the curve



Curve Normal

- Normalized curvature vector:

$$N(t) := K(t)/\|K(t)\| = T'(t)/\|T'(t)\|$$



In order to connect the cubic curves, you only need to know the **unit tangent vector.**

Also, need to know little bit more information; **curvature vector (arc length)**

Orders of Continuity

- Continuous
- Tangents Align (Geometric Continuity)
- Same Velocity (Parametric Continuity)
- Tangent & Derivatives (Curvature Continuity)

Bezier Curves: Drawback

- Tangent requires constraints linking control points

More Cubic Curve Models

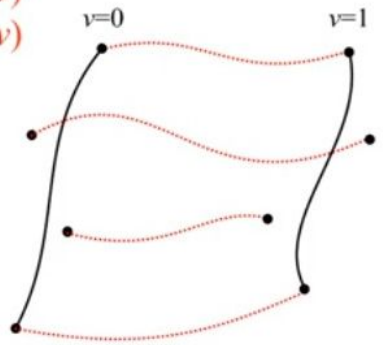
- B-Splines (More than 4 control points)

$$B_{B-Spline} = \frac{1}{6}\begin{pmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

From Parameterized Curves to Surfaces (Tessellation)

From Curves to Surfaces

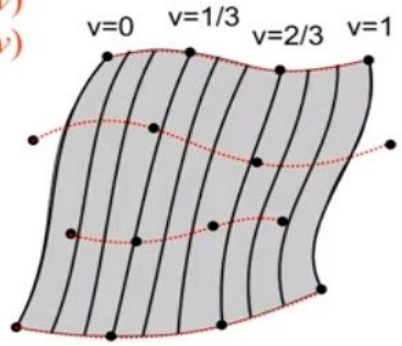
- $P(u, \underline{v}) = (1-u)^3 \quad P_1(v)$
  $+ \quad 3u(1-u)^2 \quad P_2(v)$
  $+ \quad 3u^2(1-u) \quad P_3(v)$
  $+ \quad u^3 \quad P_4(v)$
- Make P_i's move along curves!



From Curves to Surfaces

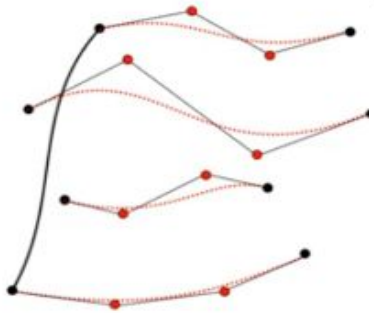
- $P(u, \underline{v}) = (1-u)^3 \quad P_1(v)$
  $+ \quad 3u(1-u)^2 \quad P_2(v)$
  $+ \quad 3u^2(1-u) \quad P_3(v)$
  $+ \quad u^3 \quad P_4(v)$
- Make P_i's move along curves!

A 2D surface patch!



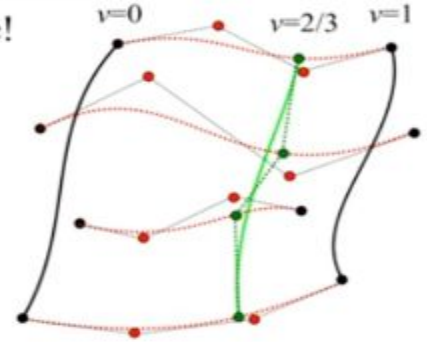
Tensor Product Bézier Patches

- In the previous, P_is were just some curves
- What if we make **them** Bézier curves?



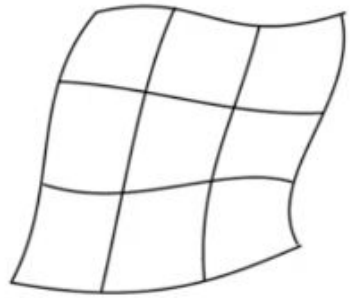
Tensor Product Bézier Patches

- In the previous, P_is were just some curves
- What if we make **them** Bézier curves?
- Each u=const. **and** v=const. curve is a Bézier curve!



Tensor Product Bézier Patches

A **bicubic Bézier surface**



From Parameterized Curves to Surfaces (Tessellation)

Bicubics, Tensor Product

$$P(u,v) = B_1(u)P_1(v) + B_2(u)P_2(v) + B_3(u)P_3(v) + B_4(u)P_4(v)$$

$$P_i(v) = B_1(v)P_{i,1} + B_2(v)P_{i,2} + B_3(v)P_{i,3} + B_4(v)P_{i,4}$$

16 control points $P_{i,j}$
16 2D basis functions $B_{i,j}$

$$P(u,v) = \sum_{i=1}^{4} B_i(u) \left[\sum_{j=1}^{4} P_{i,j} B_j(v) \right]$$

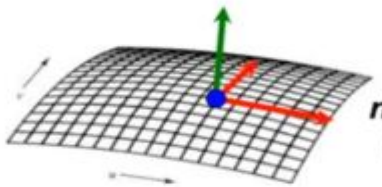
$$= \sum_{i=1}^{4} \sum_{j=1}^{3} P_{i,j} B_{i,j}(u,v)$$

$$B_{i,j}(u,v) = B_i(u) B_j(v)$$

Tangents and Normals for Patches

- $P(u,v)$ is a 3D point specified by u, v
- The partial derivatives $\partial P/\partial u$ and $\partial P/\partial v$ are tangent vectors
  - Normal is perpendicular to both:

$$n = \left(\frac{\partial P}{\partial u}\right) \times \left(\frac{\partial P}{\partial v}\right)$$



***n* is usually not unit, so must normalize!**

Matrix Notation for Patches

- Not required, but convenient!

$$P(u,v) = \sum_{i=1}^{4} B_i(u) \left[\sum_{j=1}^{4} P_{i,j} B_j(v) \right]$$

x coordinate of surface at (u,v)

$$P^x(u,v) =$$

Column vector of basis functions (v)

$$(B_1(u) \quad \cdots \quad B_4(u)) \begin{pmatrix} P^x_{1,1} & \cdots & P^x_{1,4} \\ \vdots & \ddots & \vdots \\ P^x_{4,1} & \cdots & P^x_{4,4} \end{pmatrix} \begin{pmatrix} B_1(v) \\ \vdots \\ B_4(v) \end{pmatrix}$$

Row vector of basis functions (u)

4x4 matrix of x coordinates of the control points

Hardcore: Matrix Notation for Patches

- Curves:

$$P(t) = G \cdot B \cdot T(t)$$

- Surfaces:

$$P^x(u,v) = T(u)^\top B^\top G^x B T(v)$$

A separate 4x4 geometry matrix for x, y, z

- $\boldsymbol{T}$ = power basis
  $\boldsymbol{B}$ = spline matrix
  $\boldsymbol{G}$ = geometry matrix

Advanced topic:
Combine into a *tensor*

Smooth Surfaces to Manifold

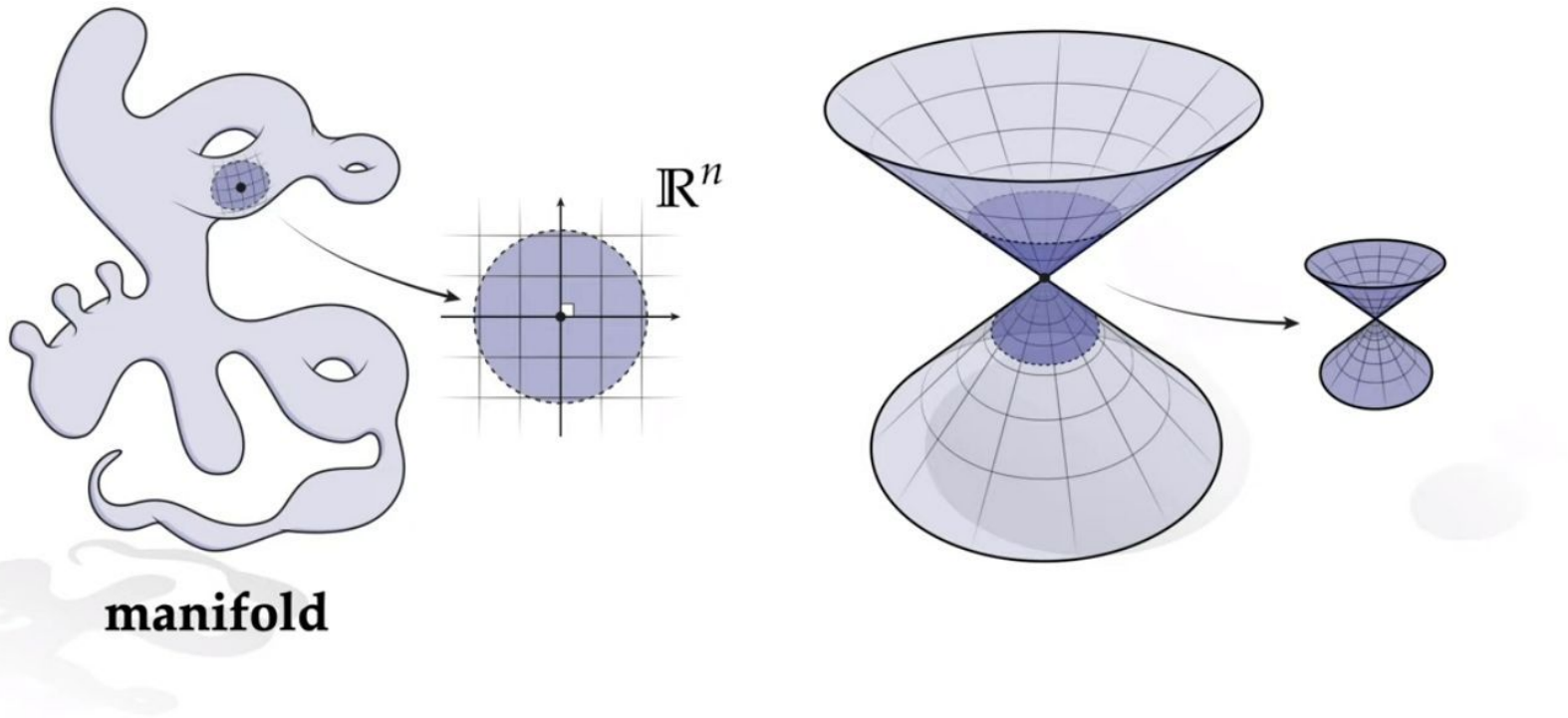
- **Intuitively, a surface is the boundary of “shell” of an object**
- **(Think about the candy shell, not the chocolate.)**
- **Surfaces are manifolds:**
  - If you zoom if far enough, can draw a regular coordinate grid
  - E.g., The Earth from space vs. from ground

Manifold

- Very rough idea: notion of “nice” space in geometry
- Can’t draw ordinary 2D grid at center, no matter how close we get (right image)

Manifold—First Glimpse

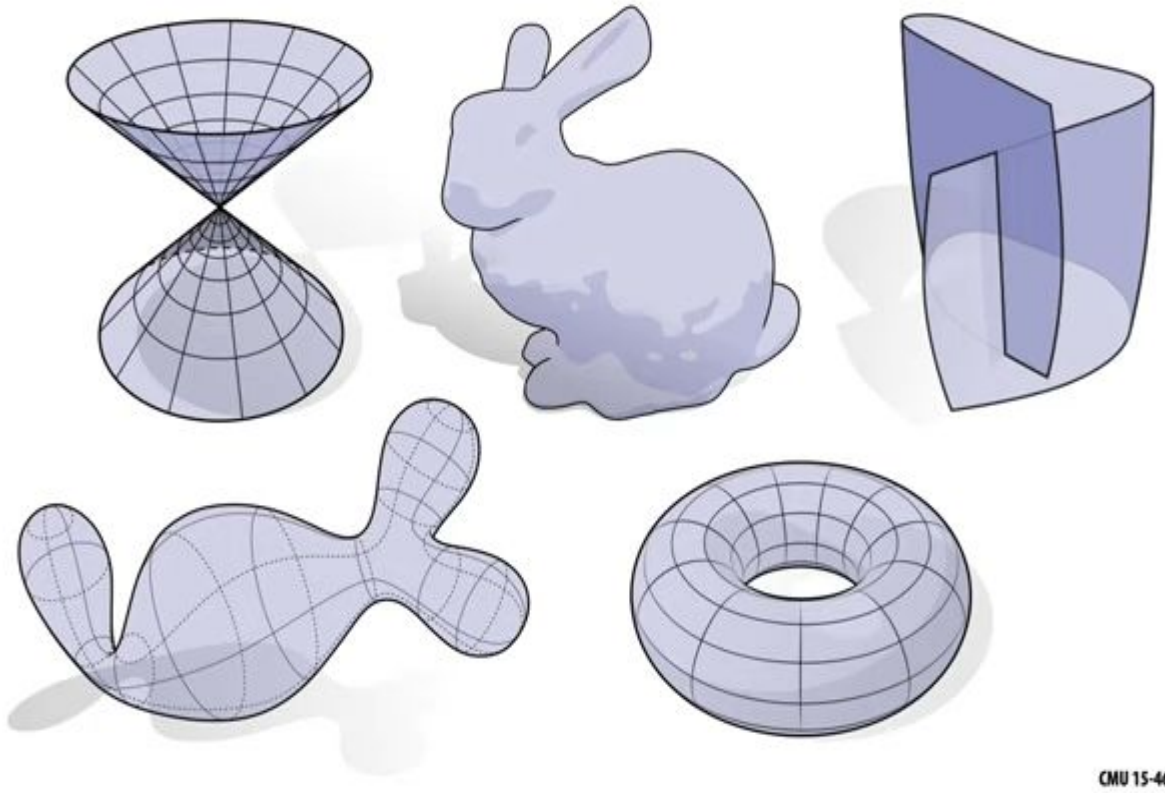
Key idea: manifold locally “looks like” $\mathbb{R}^n$



Manifold with Polygon Mesh Connectivity

Examples—Manifold vs. Nonmanifold

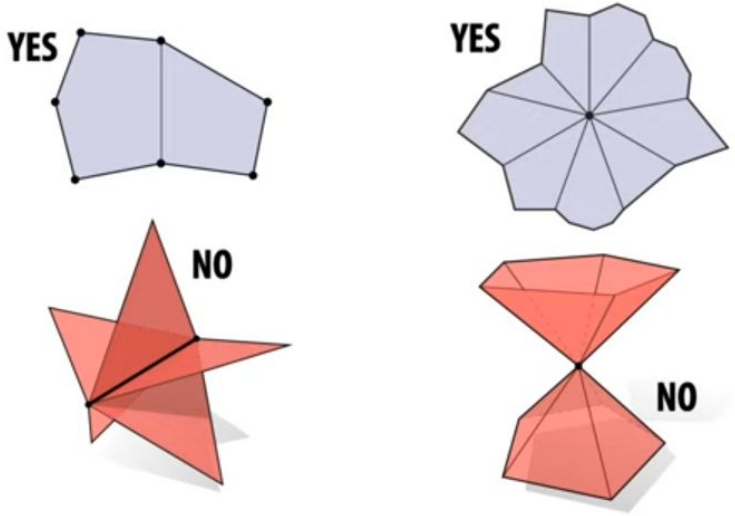
- Which of these shapes are manifold?

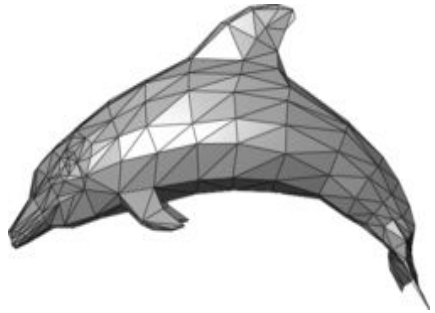


CMU 15-462/662

A Manifold Polygon mesh has fans, not fins.

1. Every Edge is contained in only two polygons (no “fins”)
2. The polygons containing each vertex make a single “fan”(loop)

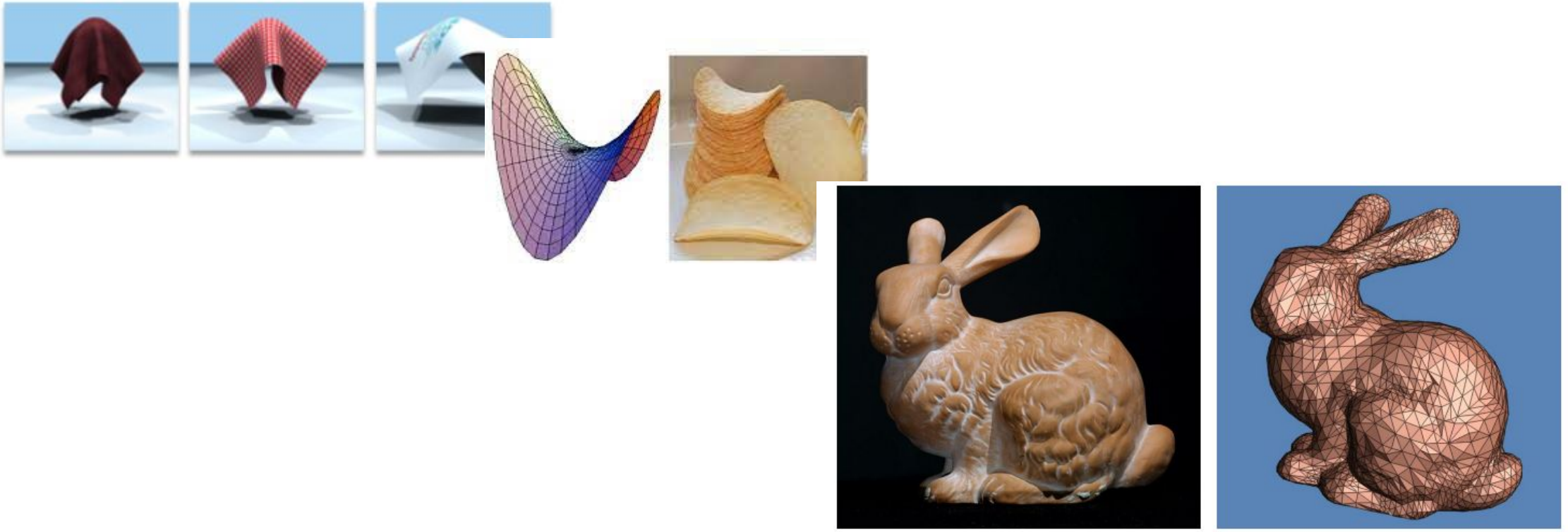




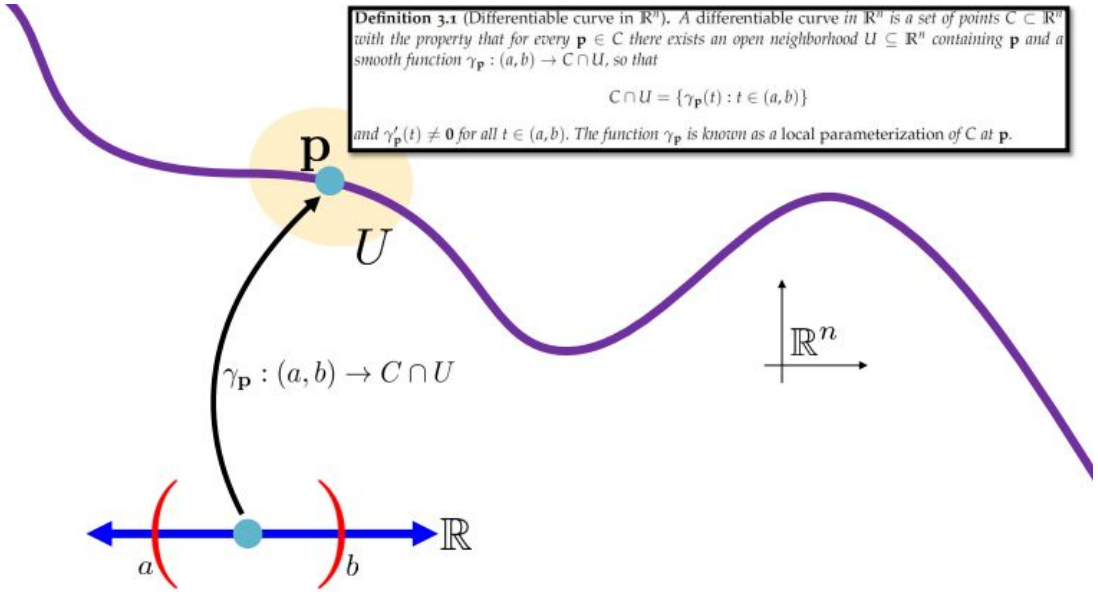
Polygon Mesh is a collection of **vertices, edges, and faces** that defines the shape of polyhedral object.

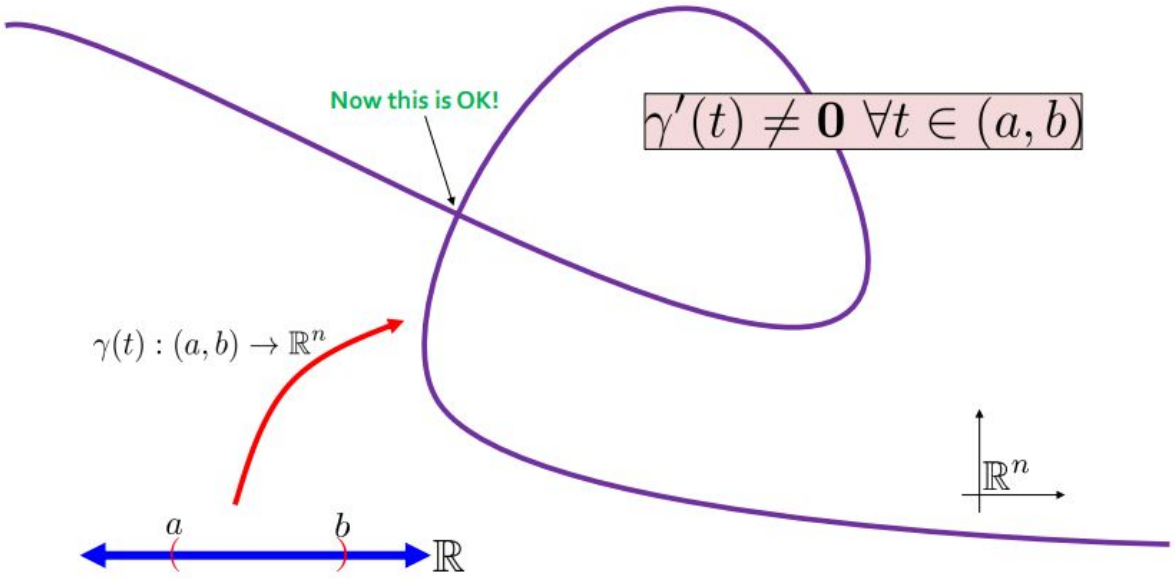
Differential Geometry (DG)

- **Differential Geometry: <u>Describe</u>** and **<u>Analyze</u>** geometric characteristics
  - how smooth?
  - how shapes deform ?
  - how curves, surfaces, and shapes change and bend in our world. (ex: how a round earth's surface becomes a flat map)

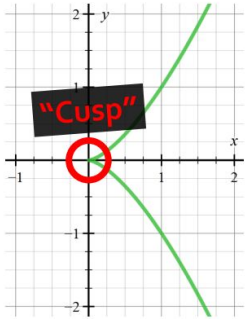


Curve is not a function, but set of points with certain properties





NO CUSP is allowed… in DG



$\gamma(t) = (t^2, t^3)$

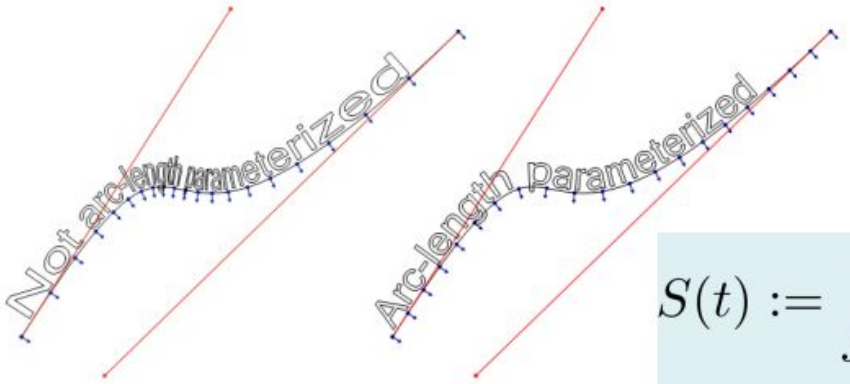
Reparameterization in terms of ArcLength (PBAL)

$$\tilde{\gamma}(t) := \gamma(\phi(t))$$

$$\int_a^b \|\gamma'(t)\|_2 \, dt$$

Effect on velocity and acceleration?

Independent of parameter!



$$S(t) := \int_{t_0}^{t} \|\gamma'(t)\|_2 \, dt$$

$$t = \phi \circ S(t)$$

Inverse functions

$$\tilde{\gamma}(s) := \gamma \circ \phi(s)$$

PBAL(Parameterized by Arc Length)

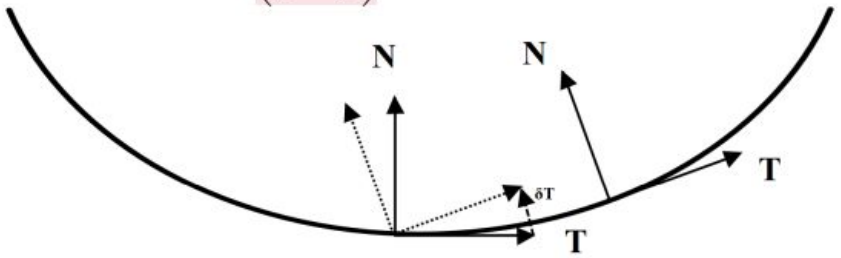
Composition Function

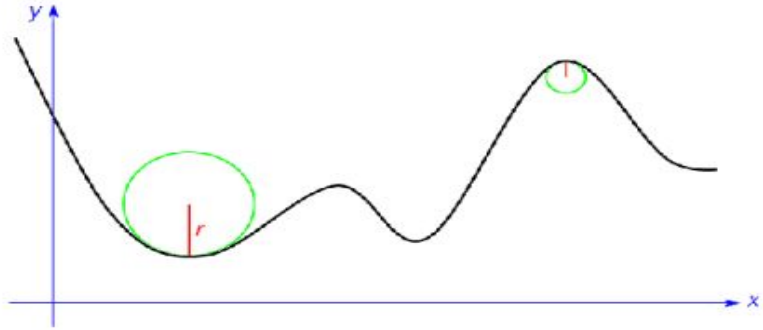
Constant-speed parameterization

Tangent, Curvature, and Normal in 2D Curve (Curvature is all you need)

$$\mathbf{T}(s) := \gamma'(s)$$
$$\implies \|\mathbf{T}(s)\|_2 \equiv 1$$
$$\mathbf{N}(s) := J\mathbf{T}(s) = \kappa(s)^{-1}\mathbf{T}'(s)$$

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

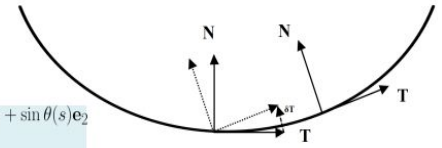




$$r(s) := \frac{1}{\kappa(s)}$$

$$\frac{d}{ds}\begin{pmatrix}\mathbf{T}(s)\\ \mathbf{N}(s)\end{pmatrix} := \begin{pmatrix} 0 & \kappa(s) \\ -\kappa(s) & 0 \end{pmatrix}\begin{pmatrix}\mathbf{T}(s)\\ \mathbf{N}(s)\end{pmatrix}$$

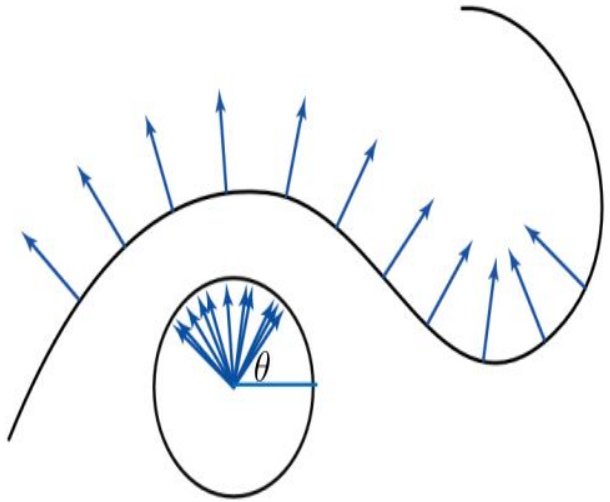
Signed curvature κ is rate of change of turning angle θ.



$\mathbf{T}(s) = \cos\theta(s)\mathbf{e}_1 + \sin\theta(s)\mathbf{e}_2$
$\kappa(s) := \theta'(s)$

https://en.wikipedia.org/wiki/Frenet%E2%80%93Serret_formulas

Use coordinates *from* the curve to express its shape!



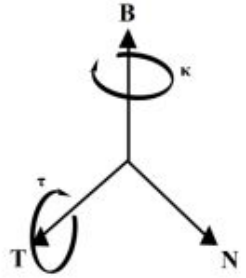
$$\mathbf{T}(s) := (\cos\theta(s), \sin\theta(s))$$
$$\implies \kappa(s) := \theta'(s)$$

Tangent, Curvature, and Normal, Binormal in 3D Curve

Frenet Frame: Curve in R^3 → Curvature and torsion distinguish a 3D curve up to rigid motion

$$\frac{d}{ds}\begin{pmatrix}\mathbf{T}(s)\\ \mathbf{N}(s)\\ \mathbf{B}(s)\end{pmatrix} = \begin{pmatrix}0 & \kappa(s) & 0\\ -\kappa(s) & 0 & \tau(s)\\ 0 & -\tau(s) & 0\end{pmatrix}\begin{pmatrix}\mathbf{T}(s)\\ \mathbf{N}(s)\\ \mathbf{B}(s)\end{pmatrix}$$

- **Binormal:** $\boldsymbol{T} \times \boldsymbol{N}$
- **Curvature:** In-plane motion
- **Torsion:** Out-of-plane motion



$$\gamma(s) : \mathbb{R} \to \mathbb{R}^n$$

$$\frac{d}{ds}\begin{pmatrix}\mathbf{e}_1(s)\\ \mathbf{e}_2(s)\\ \vdots\\ \mathbf{e}_n(s)\end{pmatrix} = \begin{pmatrix}0 & \chi_1(s) & & \\ -\chi_1(s) & \ddots & \ddots & \\ & \ddots & 0 & \chi_{n-1}(s)\\ & & -\chi_{n-1}(s) & 0\end{pmatrix}\begin{pmatrix}\mathbf{e}_1(s)\\ \mathbf{e}_2(s)\\ \vdots\\ \mathbf{e}_n(s)\end{pmatrix}$$

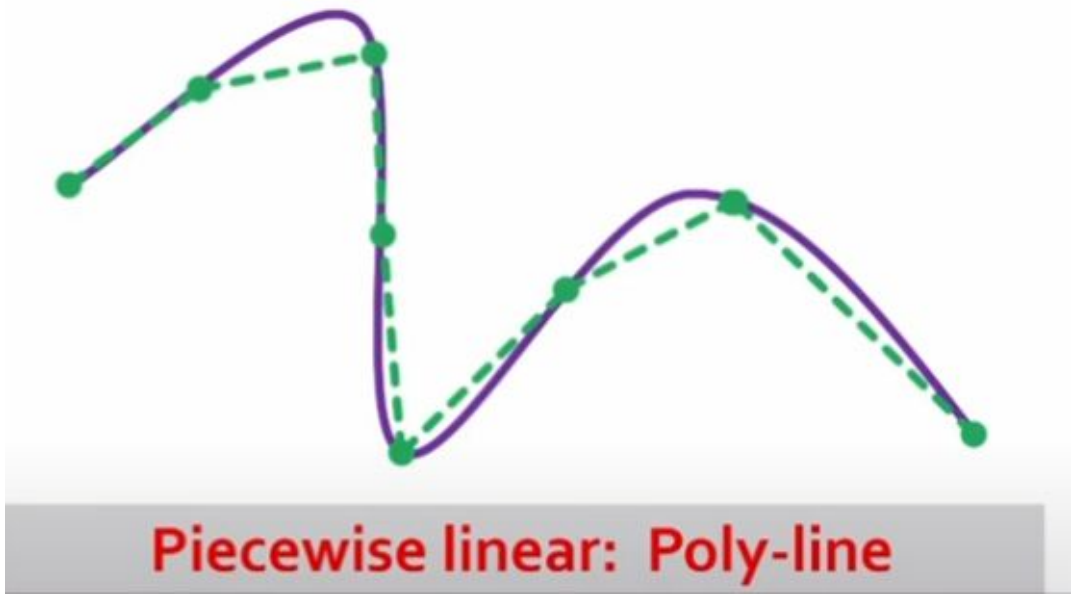
Discrete Curve Differential Geometry ⇔ Smooth Curve in Differential Geometry

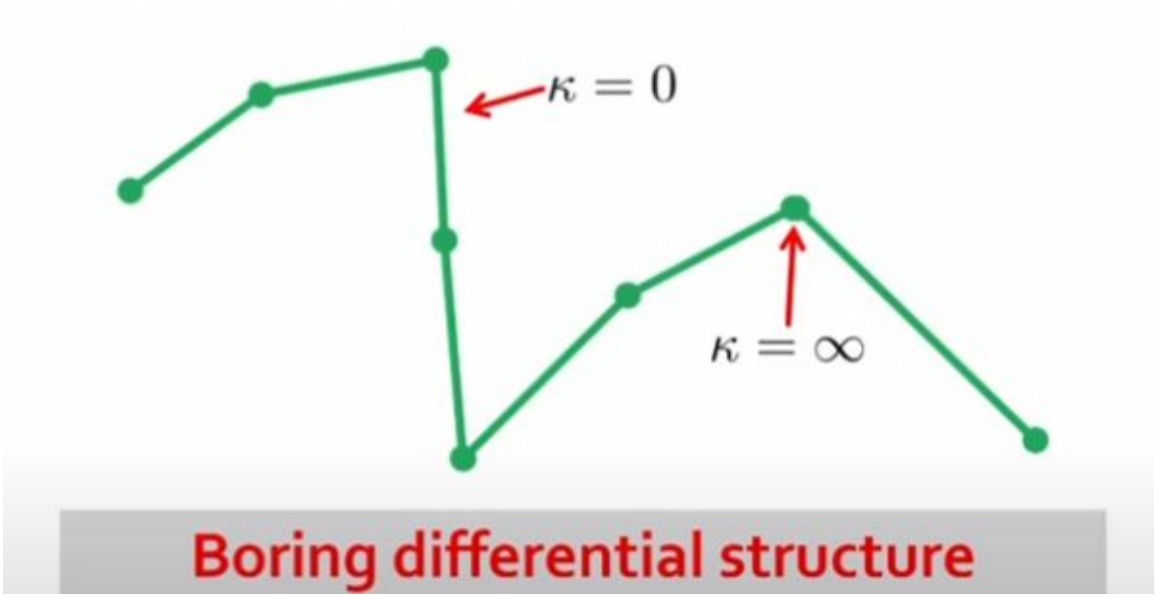
Input : Discretized Curve

Question : Can I approximate quantities like curvature and torsion in reasonable fashion ?

Motivation : What is the arc length of a cubic Bezier Curve ? → No formula(just approximation / not exact sln)

How : Solve this with piecewise linear: **Polyline**





Finite Difference Approach & Discrete Gauss Map

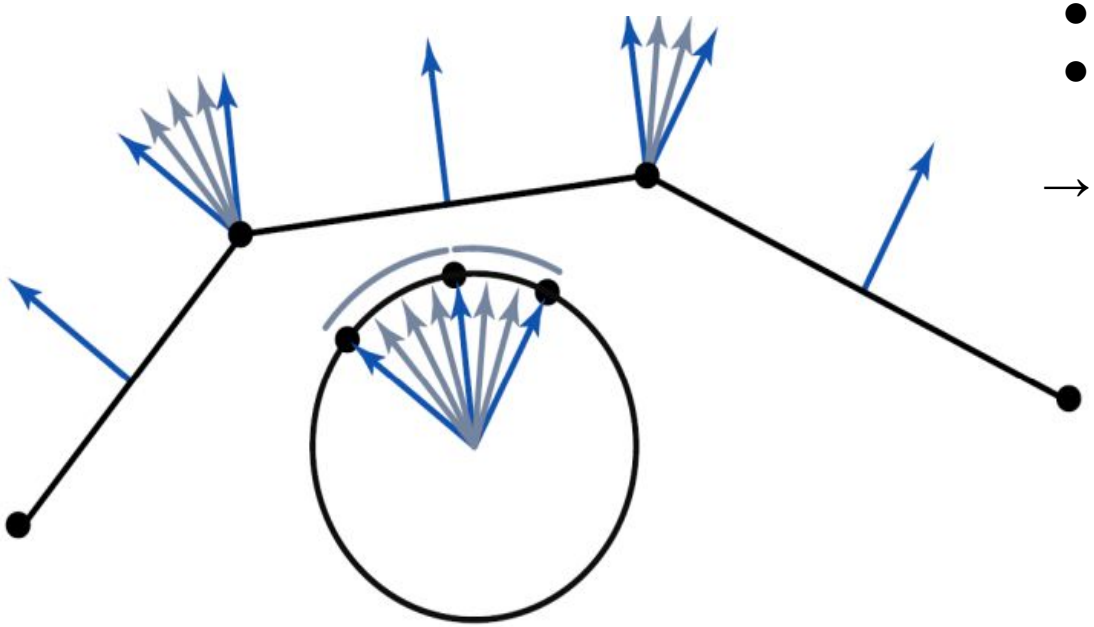
$$f'(x) \approx \frac{1}{h}[f(x+h) - f(x)]$$

h > 0

THEOREM: As $\Delta h \to 0$, [insert statement].

Two Key Considerations

1. Convergence to continuous theory
2. Discrete behavior

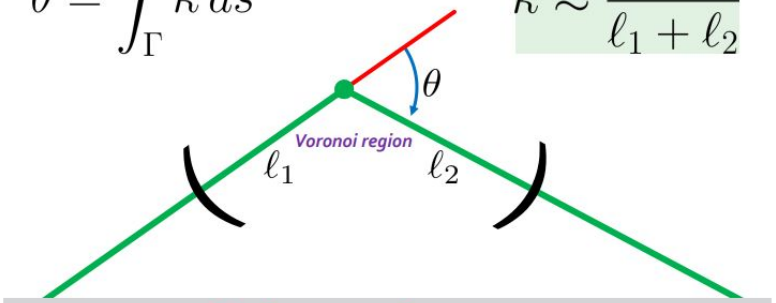


- Each Edges Become points
- Vertices becomes Arc

→ **Primal Dual Structure**

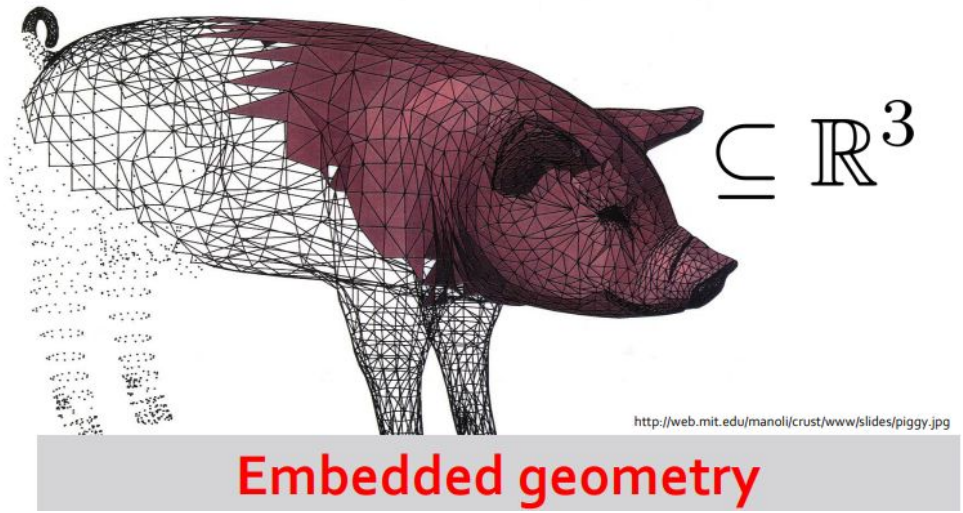
$$\theta = \int_\Gamma \kappa \, ds$$

$$\kappa \approx \frac{\theta}{\ell_1 + \ell_2}$$

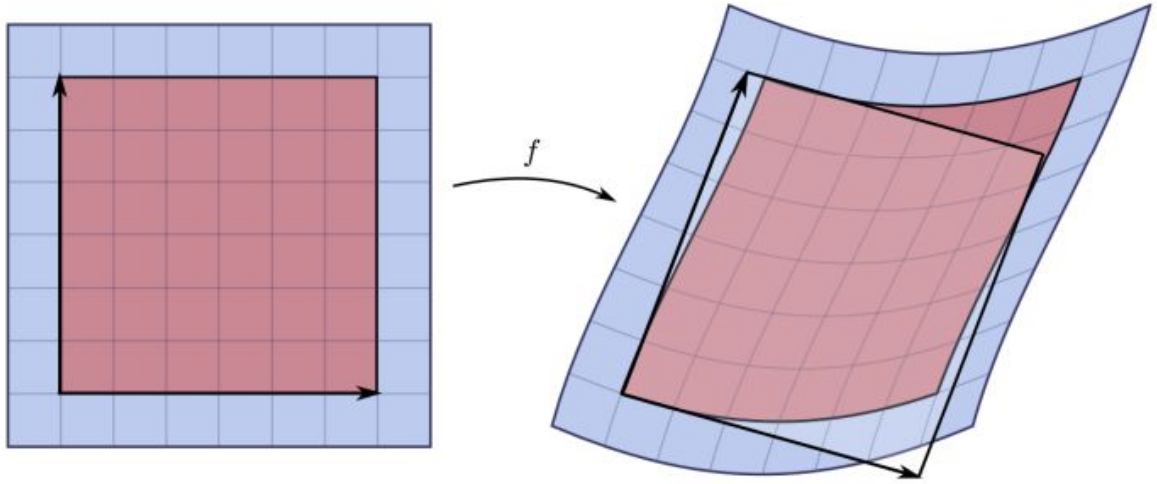


Total change in curvature

Surface (Embedded Surface)



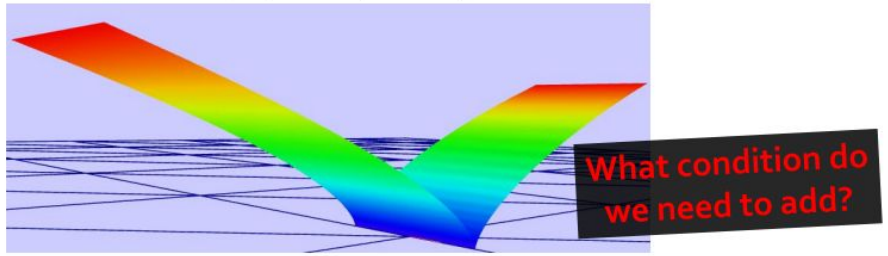
Parametric Surface



Something to Worry About

ex:) CUSP

$$f(u,v) = (u, u^2, \cos u)$$
$$f(u,v) = (0,0,0)$$
$$f(u,v) = (u, v^3, v^2)$$



$$f : \mathbb{R}^m \to \mathbb{R}^n$$

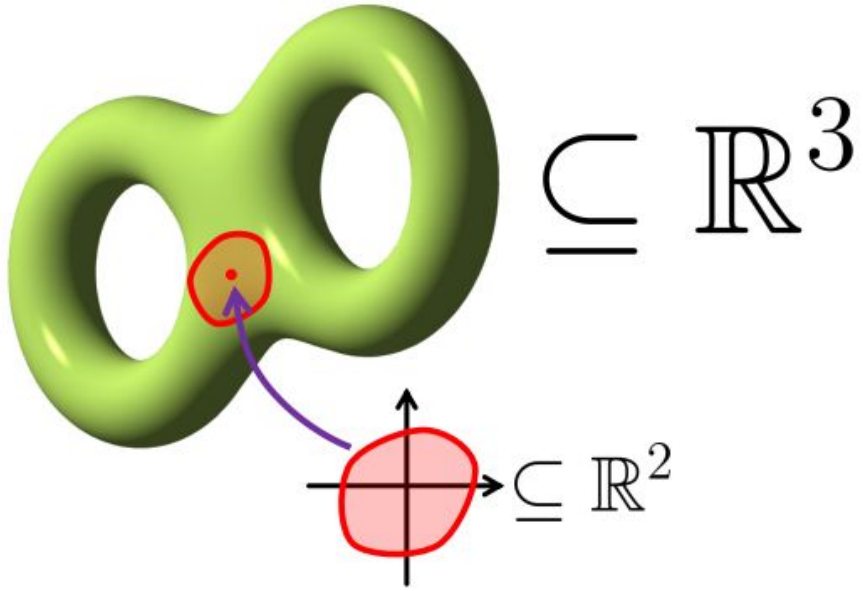
Jacobian matrix:

$$(Df)^i_j = \left(\frac{\partial f^i}{\partial x^j}\right)$$

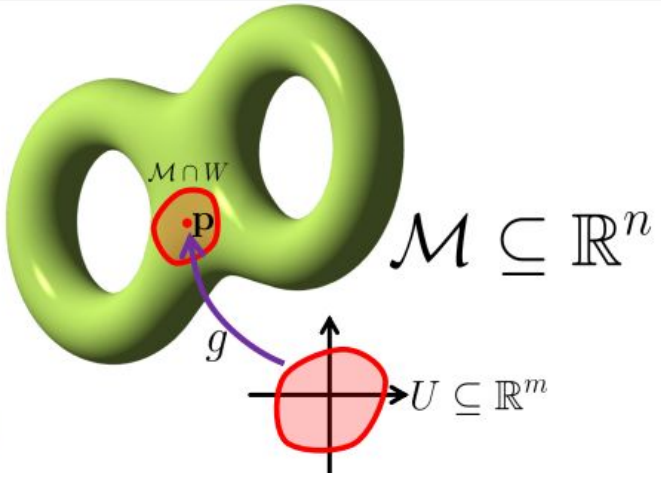
Matrix condition:

Df full rank

Surface is not a function, but set of points with certain properties



Definition (Submanifold of $\mathbb{R}^n$, with and without boundary). *A set $\mathcal{M} \subseteq \mathbb{R}^n$ is an m-dimensional submanifold of $\mathbb{R}^n$ if for each $\mathbf{p} \in \mathcal{M}$ there exist open sets $U \subseteq \mathbb{R}^m$, $W \subseteq \mathbb{R}^n$ and a function $g : U \cap \mathcal{H}_m \to \mathcal{M} \cap W$ such that $\mathbf{p} \in W$ and g is a one-to-one and smooth map whose Jacobian is rank-m and admitting a continuous inverse $g^{-1} : W \cap M \to U$.*

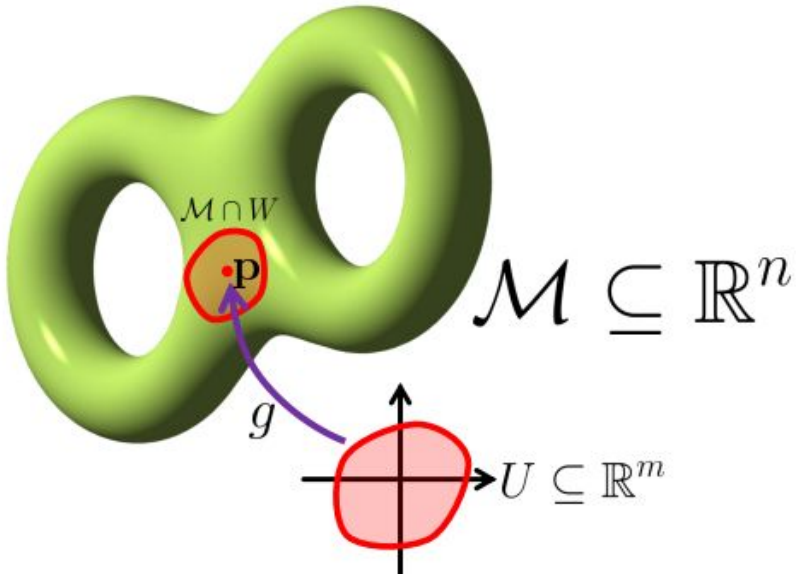


$$\mathcal{H}_m := \{\mathbf{x} \in \mathbb{R}^m : x^m \geq 0\}$$

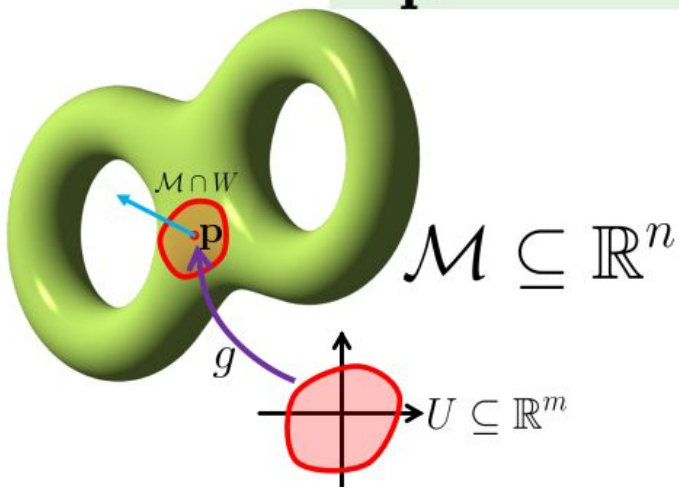
Surface is locally Planar

Tangent, Normal Space

$$T_{\mathbf{p}}\mathcal{M} = \gamma'(0), \text{ where } \gamma(0) = \mathbf{p}$$



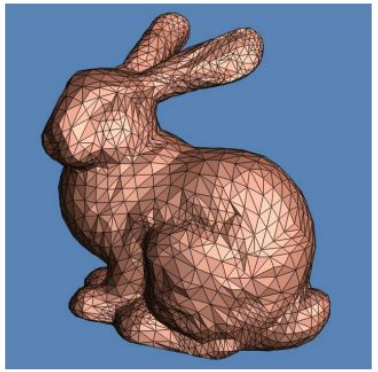
$$N_{\mathbf{p}}\mathcal{M} := (T_{\mathbf{p}}\mathcal{M})^{\perp}$$



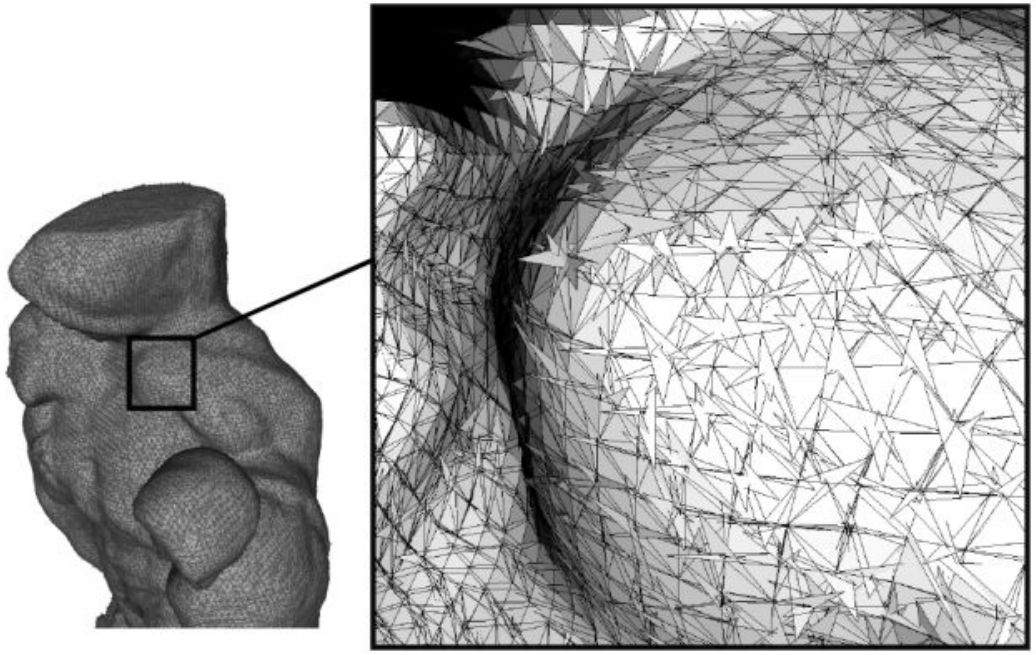
Triangle Mesh

$$V = (v_1, v_2, \ldots, v_n) \subset \mathbb{R}^3$$
$$E = (e_1, e_2, \ldots, e_k) \subseteq V \times V$$
$$F = (f_1, f_2, \ldots, f_m) \subseteq V \times V \times V$$

Plus manifold topological conditions

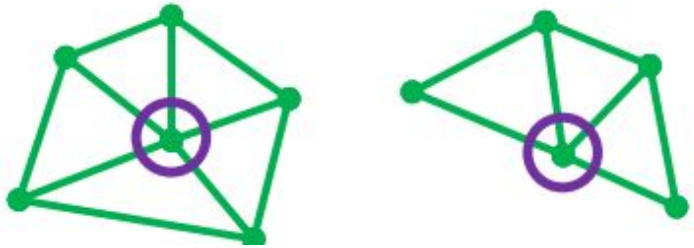


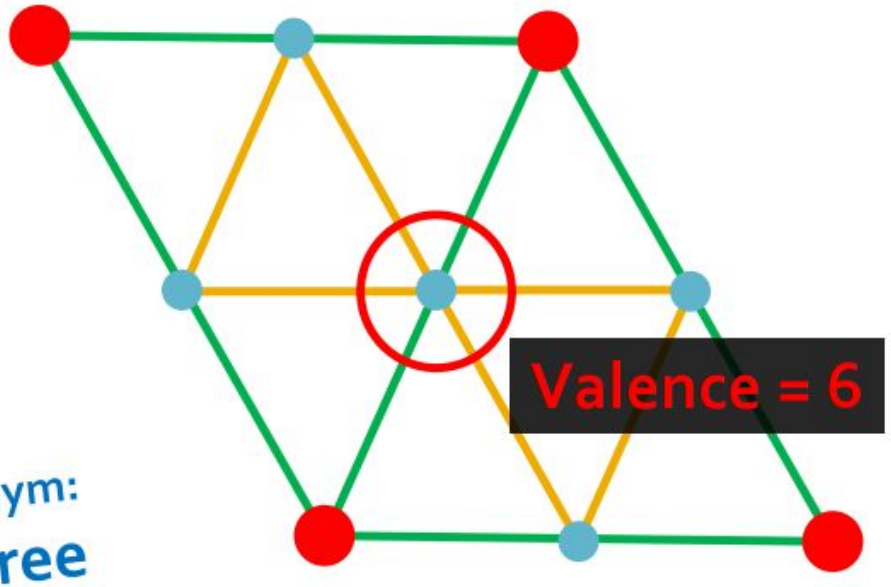
Is this legit **Discrete Surface ?**



Before Getting into Topology

1. Each edge is incident to one or two faces
2. Faces incident to a vertex form a closed or open fan





Synonym:
Degree

Topology: Euler Theorem for Meshes

$$V - E + F := \chi$$

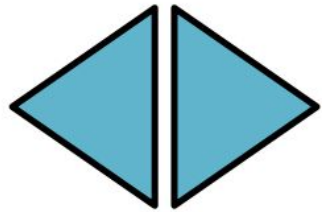
$$\chi = 2 - 2g$$

$g = 0$

$g = 1$

$g = 2$

"Each edge is adjacent to two faces. Each face has three edges."



$$2E = 3F$$

$$E \approx 3V$$

$$F \approx 2V$$

$$\text{average valence} \approx 6$$

Average Valance = 2 * #E / V

= 6v / v = 6

Data Structure: Smoothing Example

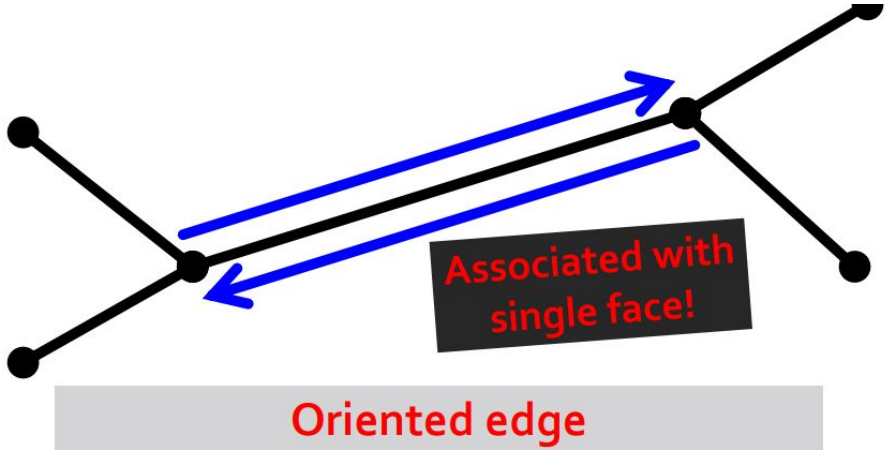
```
for i = 1 to n
    for each vertex v
        v = .5 * + .5 * (average of neighbors);
```

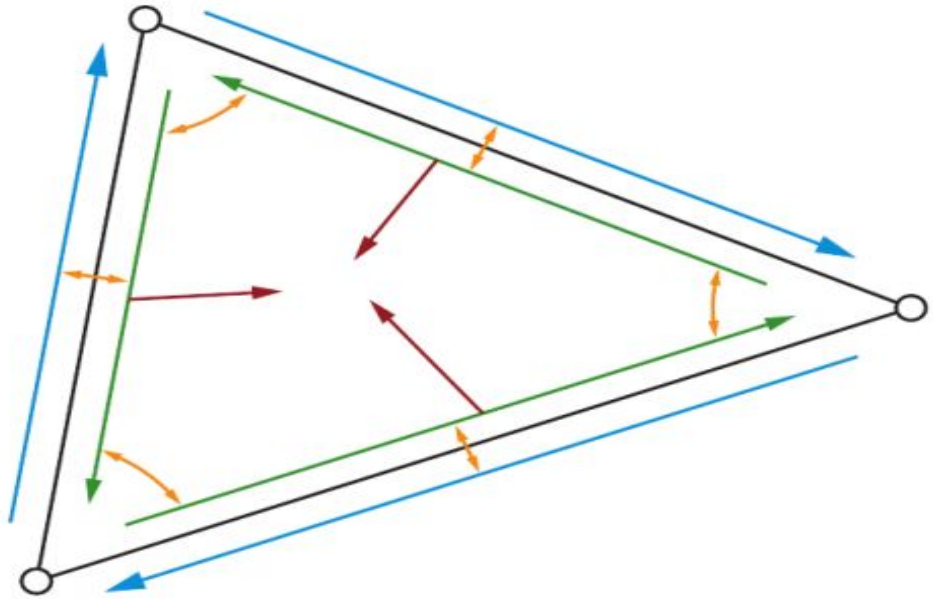
Then how would I store mesh ?

- **Incidence Matrix**
- **Half-Edges (***)**

- Neighboring vertices to a vertex
- Neighboring faces to an edge
- Edges adjacent to a face
- Edges adjacent to a vertex

Half-Edge Data Types



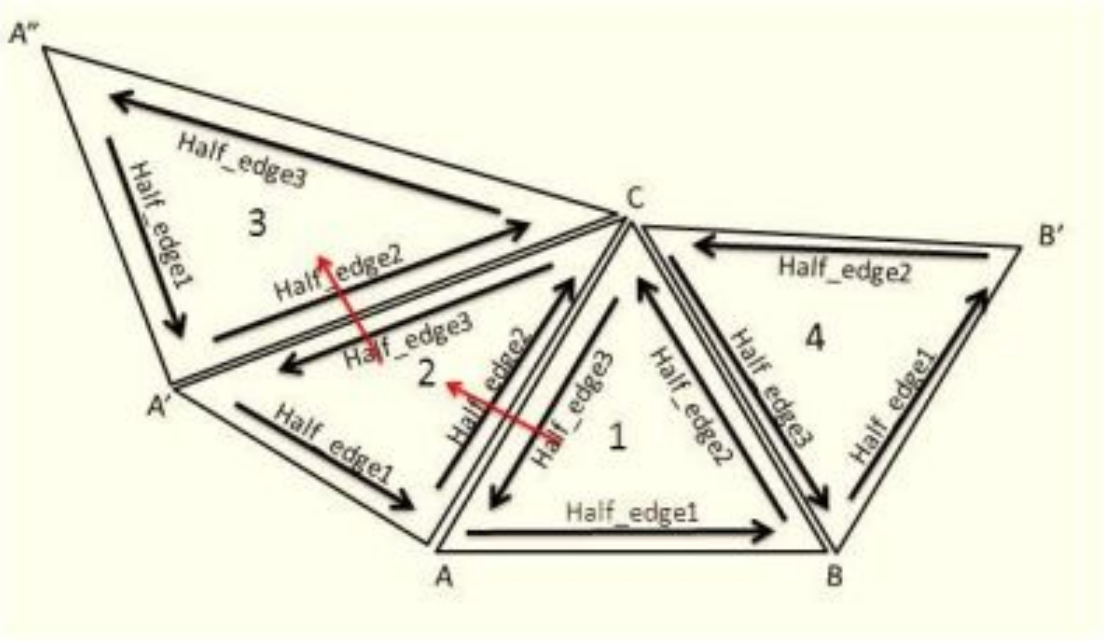


Vertex: stores arbitrary outgoing halfedge

Faces: stores arbitrary adjacent halfedge

Halfedge: stores Flipe, Next, Face, Vertex

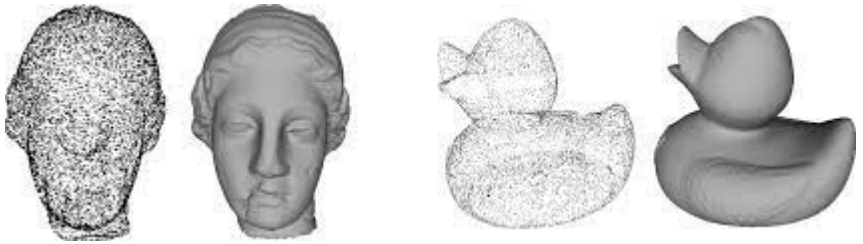
Iterating Over Vertex Neighborhoods



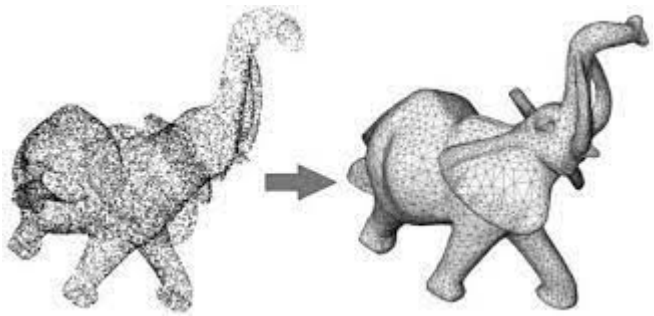

```
Iterate(v):
startEdge = v.out;
e = startEdge;
do
    process(e.flip.from)
    e = e.flip.next
while e != startEdge
```

Geometric Processing - Reconstruction

- **Reconstruction**
  - **Given Samples of geometry, reconstruct surface**
  - **What are “samples”? Many possibilities:**
    - **points, points & normals, points & normals & color**
    - **image pairs / sets (multi-view stereo)**

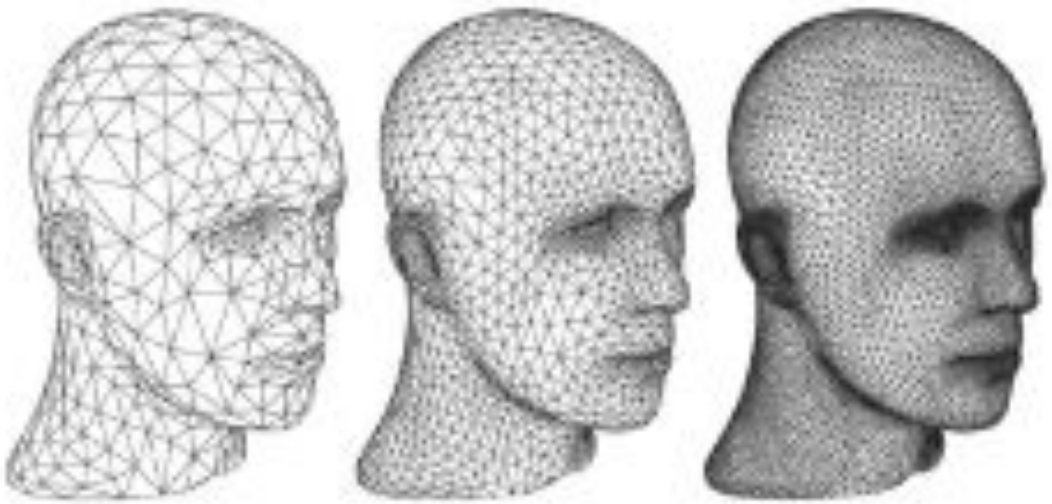


  - **How do you get a surface?**
    - **Silhouette-based (visual hull)**
    - **Voronoi-based**
    - **PDE-based (e.g Poisson Reconstruction)**
    - **Random Transform / Isosurfacing**



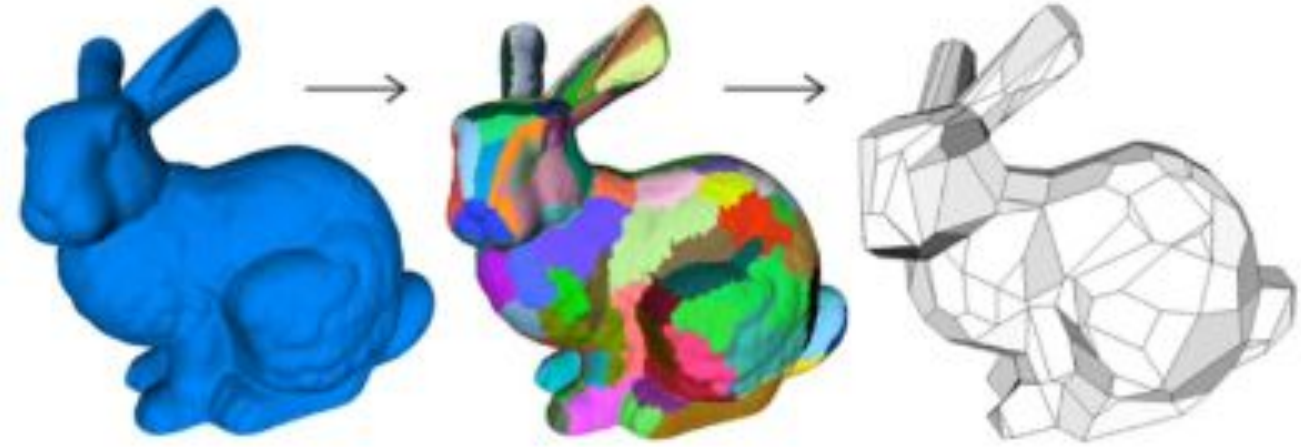
Geometric Processing - Upsampling

- **Increase Resolution via interpolation**
- **Images: e.g., bilinear, bicubic interpolation**
- **Polygon Meshes:**



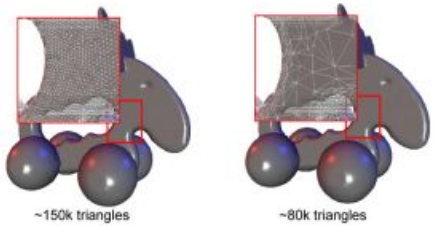
Geometric Processing - Downsampling

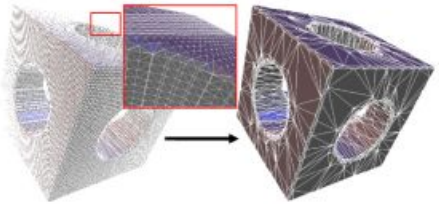
- **Decrease Resolution try to preserve shape / appearance**
- **Images: e.g., nearest-neighbor, bilinear, bicubic interpolation**
- **Point Clouds: subsampling**
- **Polygon Meshes:**
  - **Iterative decimation, variational shape approximation, …**



Geometric Processing - Resampling

- **Modify sample distribution to improve quality**
- **Images: not an issue! (pixels always stored on a regular grid)**
- **Meshes: shape of polygons is extremely important**
  - **Reducing number of faces while trying to keep overall shape, volume, and boundaries**
  - **Oversampled 3D Scan data**
  - **Filtering isosurface out of volume datasets**





Geometric Processing - Shape Analysis

- **Identify/understand important semantic features**
- **Images: computer vision, segmentation, face detection, …**
- **Polygon meshes:**
  - **Segmentation, correspondence, symmetry detection…**

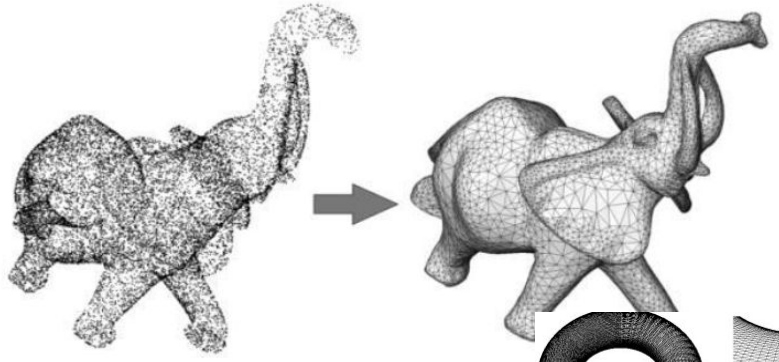
Extrinsic symmetry

Intrinsic symmetry

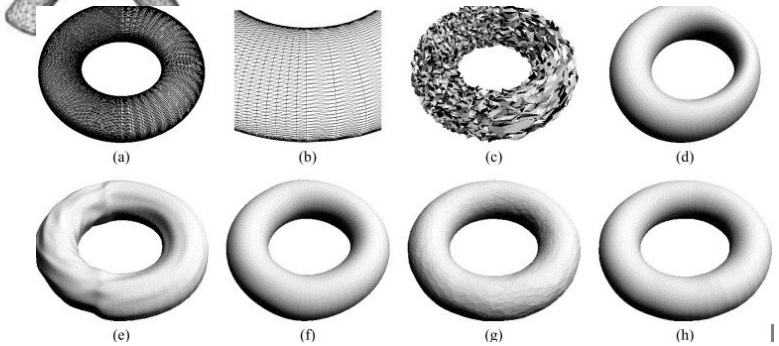
DG in Computer Vision

- **In Computer Vision…**
  - **Surface Registration :** DG is used in aligning and registering different views of the same object or scene. By analyzing the geometric properties of surface, computer vision system can align and merge multiple images or point clouds to create coherent 3D representation.
  - **Surface Reconstruction :** By estimating surface normals, curvatures, and other geometric properties, computer vision system can create accurate 3D models of objects
  - **Shape Regularization and Smoothing :** DG provides methods for smoothing and regularizing surfaces, which can be used to enhance the quality of reconstructed or digitized shape.
  - **…**

https://www5.informatik.uni-erlangen.de/Forschung/Publikationen/2011/Neumann11-RRM.pdf



CGAL: Poisson surface reconstruction from a point-cloud.



mesh regularization and smoothing

Further Reading

- **Flows and Vector Field**
- **Differential Operators**
- **Riemannian Geometry**
- **Geometric Mechanics and Lie Groups**
- **…**

[Ref] Shan Liu, Min Zhang, Pranav Kadam, C.-C. Jay Kuo - 3D Point Cloud Analysis_ Traditional, Deep Learning, and Explainable Machine Learning Methods-Springer (2021)

Resource

- Youtube
  - Introduction To Computer Graphics, Computer Graphics
  - Discrete Differential Geometry

- Book
  - Differential Geometry of Curves & Surfaces - Manfredo P. Do Carmo
  - Differential Geometry: Connections, Curvature, and Characteristic Classes

- Slide
  - Differential geometry I
  - Differential Geometry of Surfaces

- Paper
- Multiview Differential Geometry

Q & A

Thank You

🗂️3D Sensor Data Processing Curriculum