清華大學 Tsinghua University
交叉信息研究院 Institute for Interdisciplinary Information Sciences

Lecture 2: 3D Representations

Li Yi

2025.02.27

Homework 1 is coming!

- The first assignment is going to be released this week!

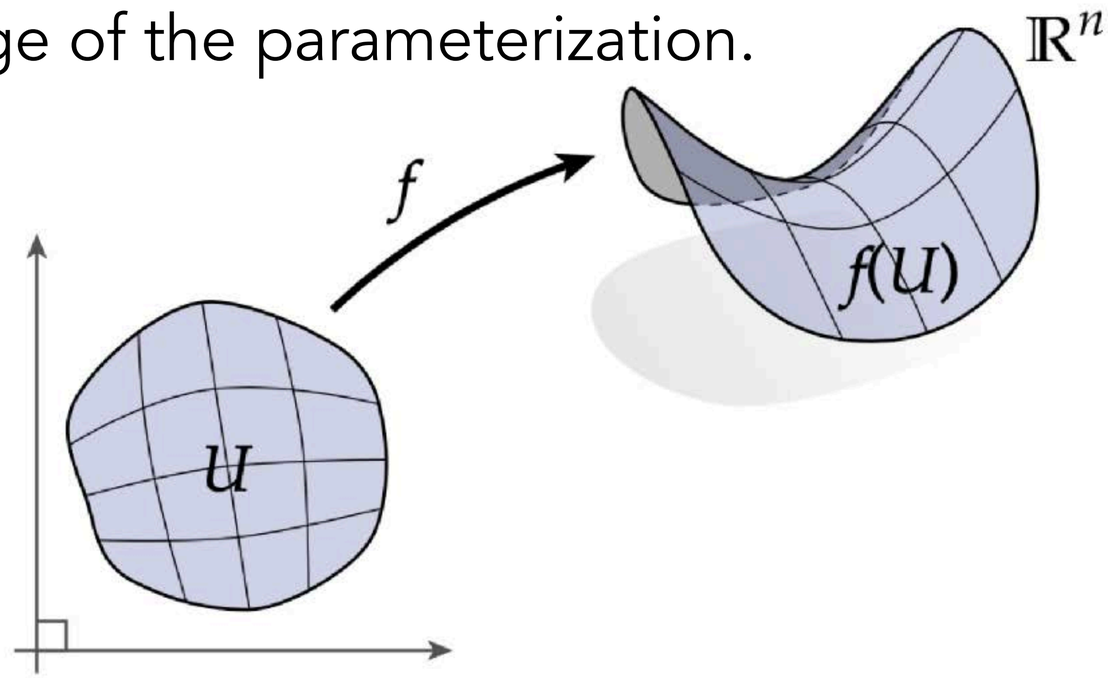
Recap

- We can use parametrized form to represent smooth curves and surfaces
- Such parametrized form allows us to define local properties such as tangent, normal and curvature
- We usually require ‘’smoothness’’ for easier analysis.

Previously: parametric surfaces

- A parameterized surface is a map f from a two-dimensional region U into $\mathbb{R}^n$
- $U \subset \mathbb{R}^2$ is the parameter domain
- The set of points $f(U)$ is called the image of the parameterization.

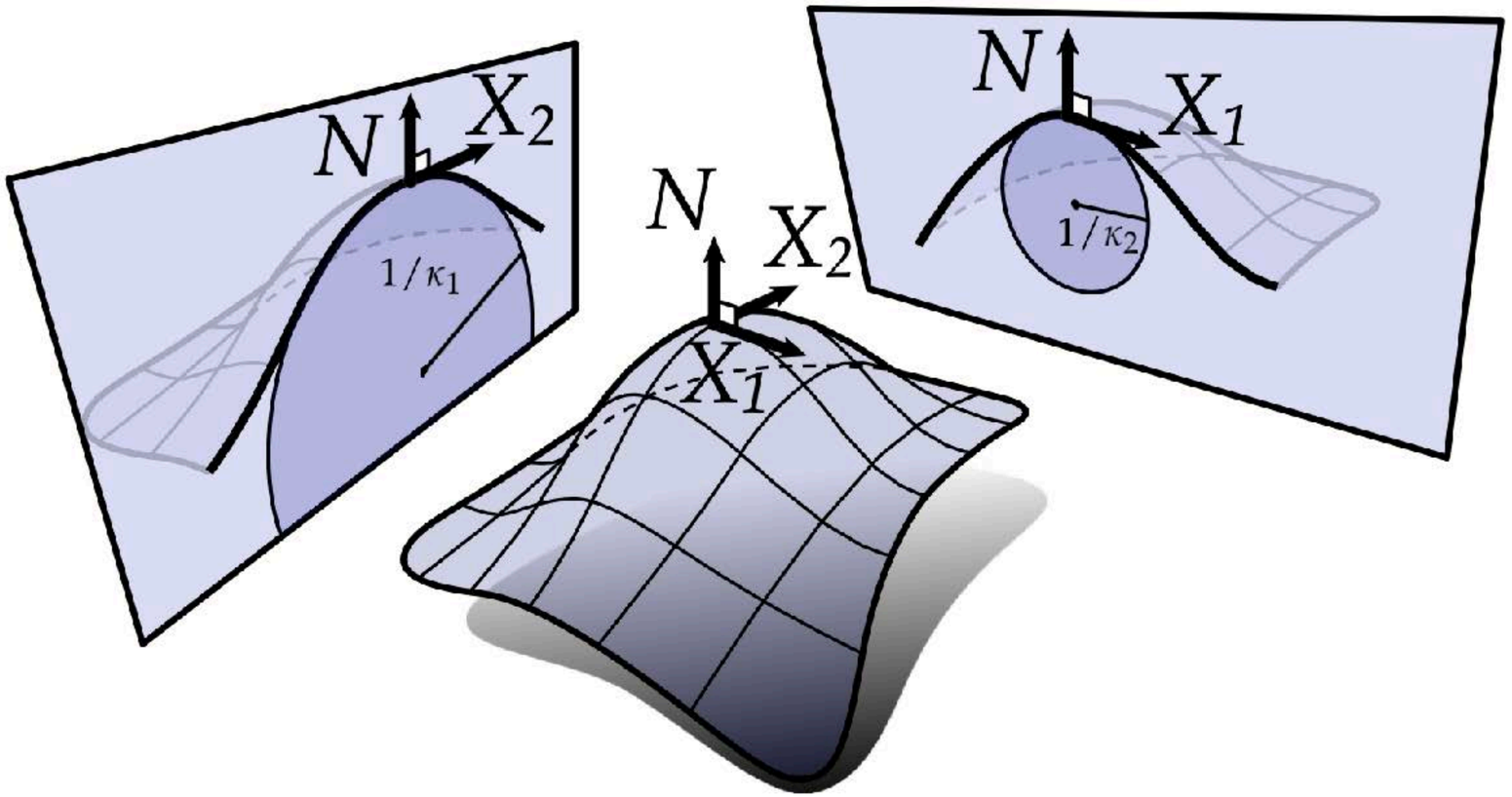
$$f : U \to \mathbb{R}^n$$



Previously: principal curvatures

Maximal curvature: $\kappa_1 = \kappa_{\max} = \max_{\varphi} \kappa_n(\varphi)$

Minimal curvature: $\kappa_2 = \kappa_{\min} = \min_{\varphi} \kappa_n(\varphi)$



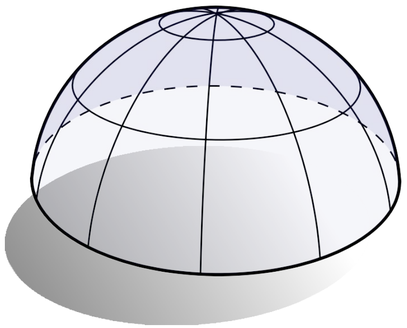
Slides credit: Hao Su

Previously: Gaussian and mean curvature

- Gaussian and mean curvature also fully describe local bending:

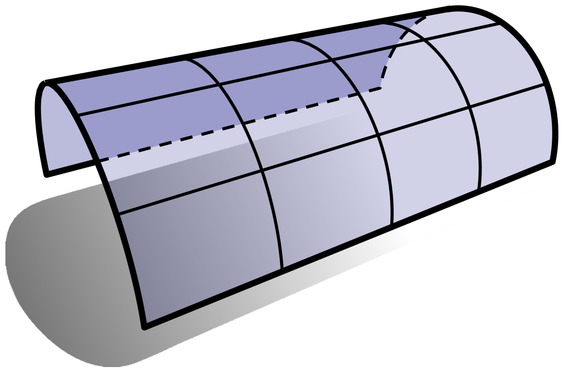
$$\text{Gaussian:} \quad K := \kappa_1 \kappa_2$$

$$\text{mean:} \quad H := \frac{1}{2}(\kappa_1 + \kappa_2)$$



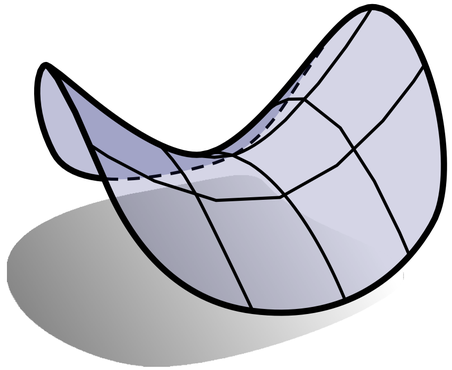
$K > 0$

$H \neq 0$



"developable" $K = 0$

$H \neq 0$



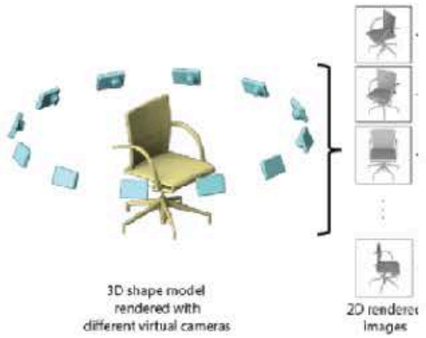
$K < 0$

"minimal" $H = 0$

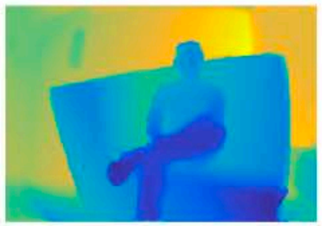
Slides credit: Hao Su

Today's focus: 3D representations

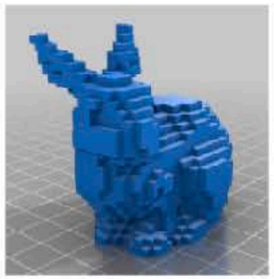
Rasterized form (regular grids)



Multi-view



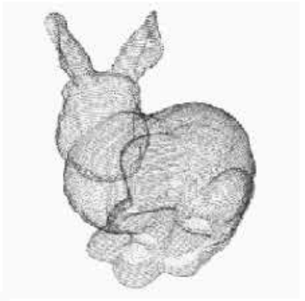
Depth Map



Volumetric

Geometric form (irregular)

Mesh



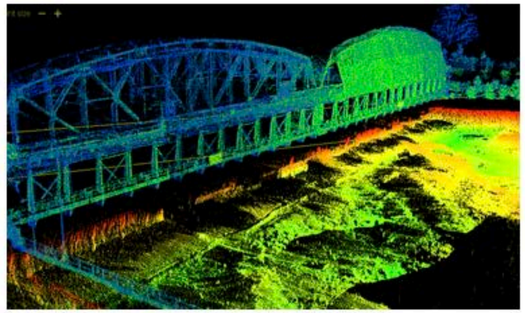
Point Cloud

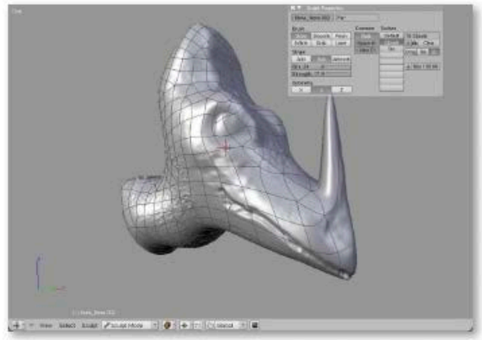
$$F(x) = 0$$

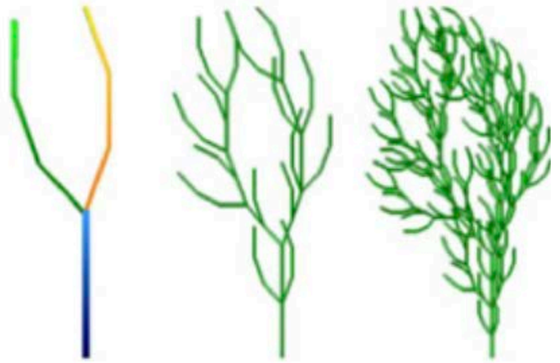
Implicit Shape

3D representations: origin-dependent

- Acquired real-world object
- Modeling “by hand”
- Procedural modeling
- …







3D representations: application-dependent

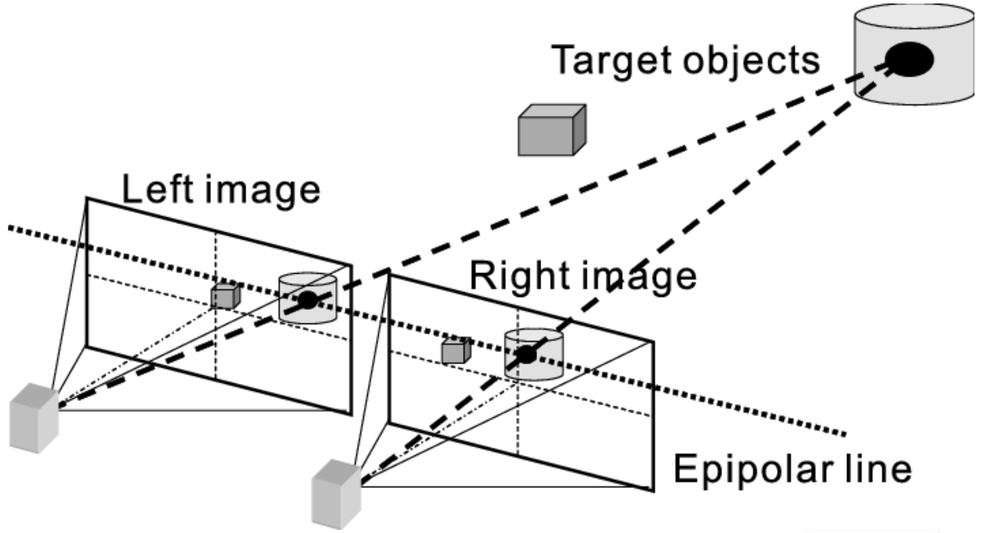
- Needs to be stored in the computer
- Creation of new shapes — input metaphors, interfaces…
- Operations — editing, simplification, smoothing, filtering, repairing…
- Rendering — rasterization, ray tracing…
- Animation

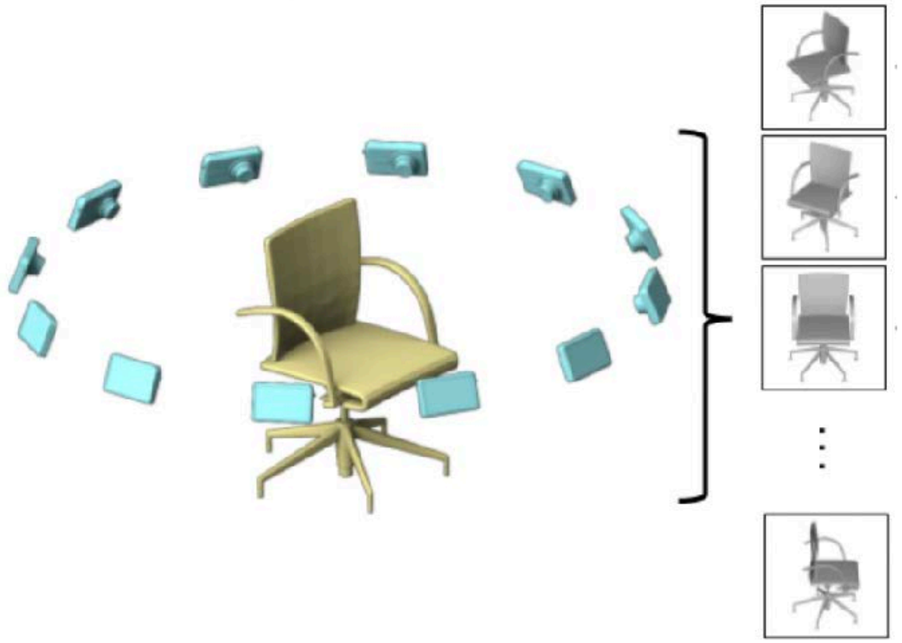
Outline

- *Rasterized 3D representations*
- Mesh
- Point cloud
- Implicit representations

Multiview 3D representation

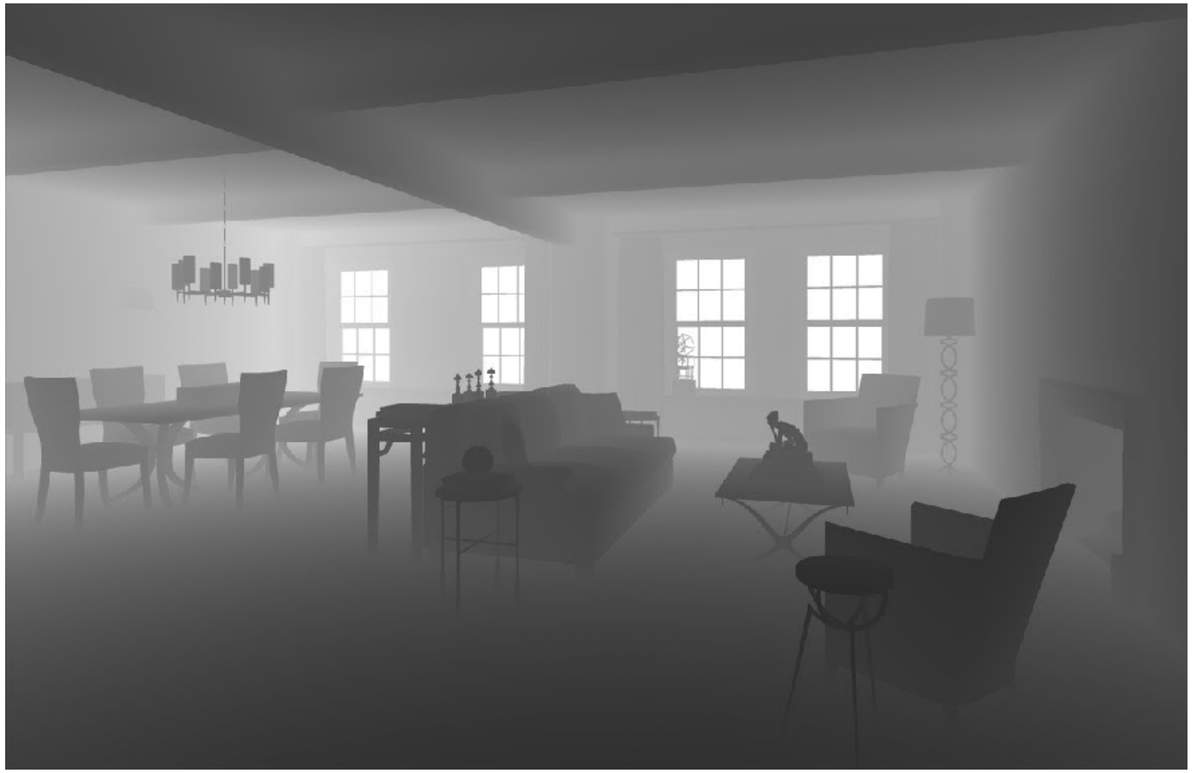
- No explicit 3D geometry
- Multiview reconstruction possible (non-trivial though)
- Viewpoint matters a lot
- How many views are sufficient?





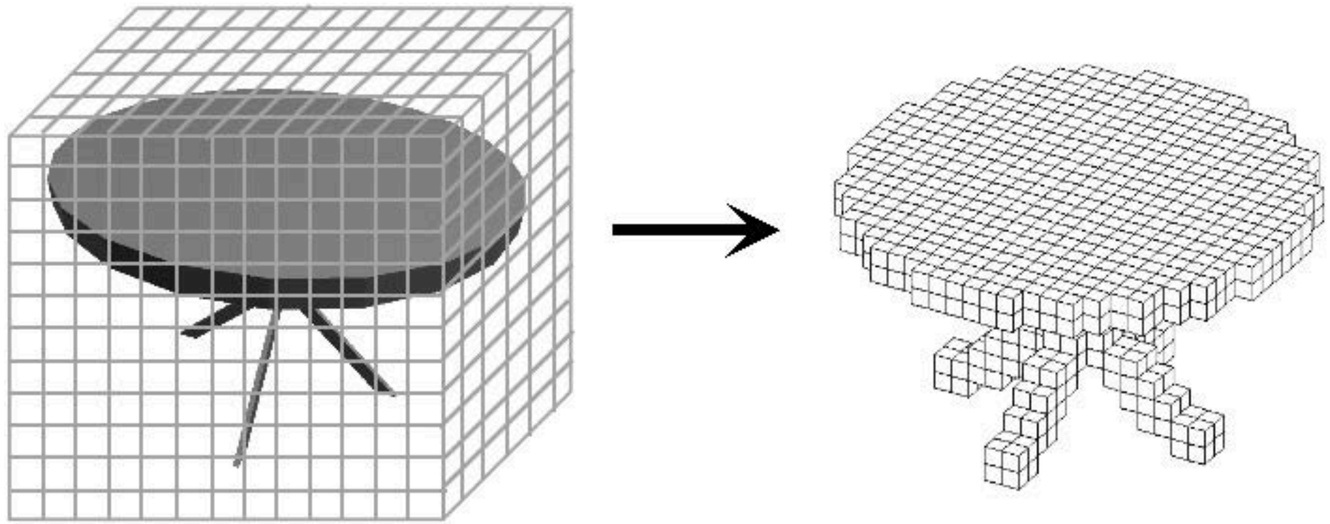
Depth map

- Incomplete 3D
- Need camera information for true 3D
- Resolution issue
- Object properties matter



Volumetric representation

- Storage and computation cost
- Obvious artifacts (w. low resolution)
- Sparse representation could be a remedy

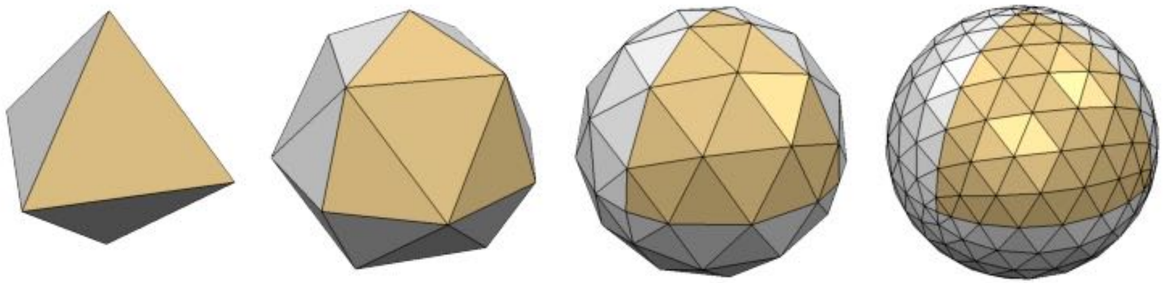


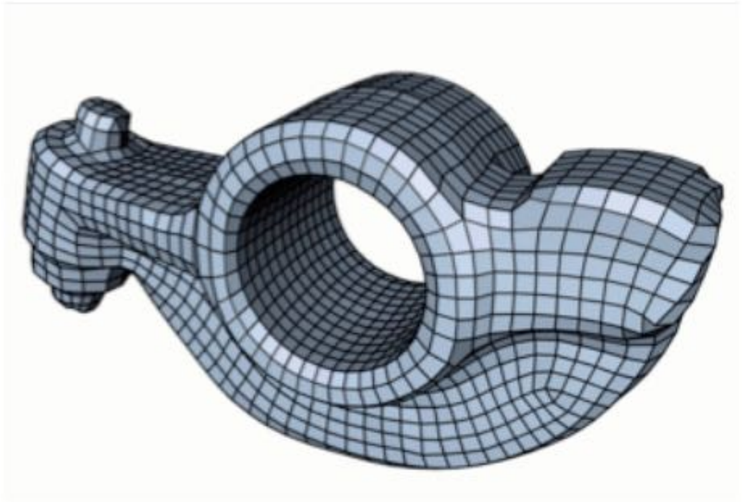
Outline

- Rasterized 3D representations
- ***Mesh***
  - ***Representation***
  - Storage
  - Curvature computation
- Point cloud
- Implicit representations

Polygonal Meshes

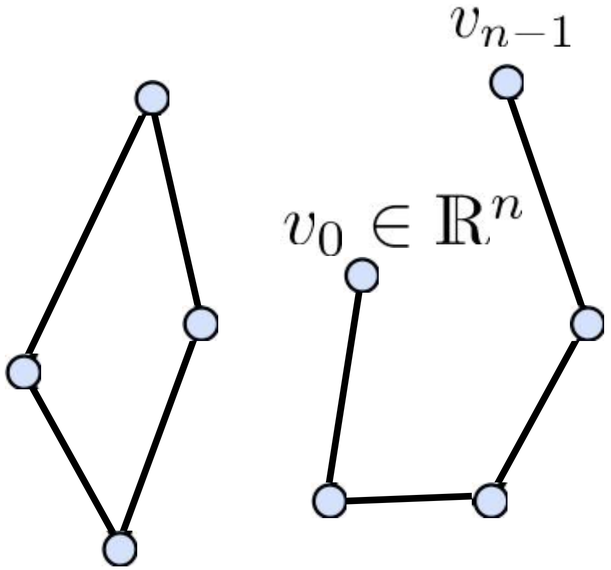
- Piece-wise linear surface representation





Slides credit: Hao Su

Polygon

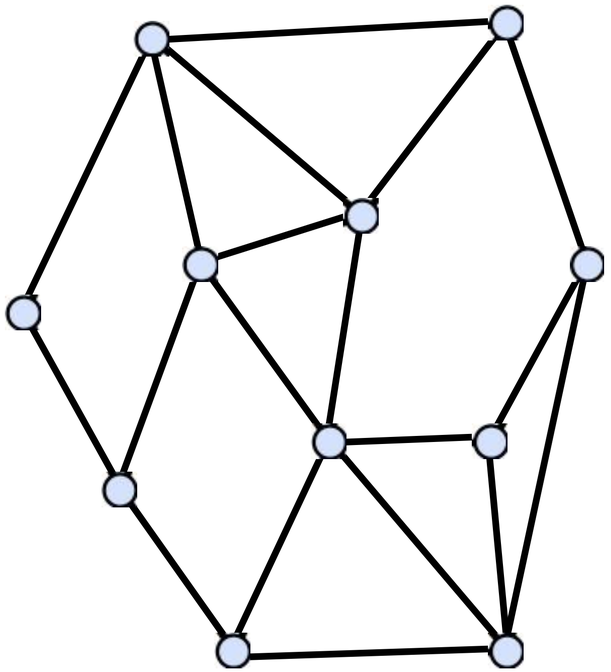


- Vertices: $v_0, v_1, \ldots, v_{n-1}$
- Edges: $\{(v_0, v_1), \ldots, (v_{n-2}, v_{n-1})\}$

- Closed: $v_0 = v_{n-1}$
- Planar: all vertices on a plane
- Simple: not self-intersecting

Slides credit: Jiajun Wu

Polygonal Mesh



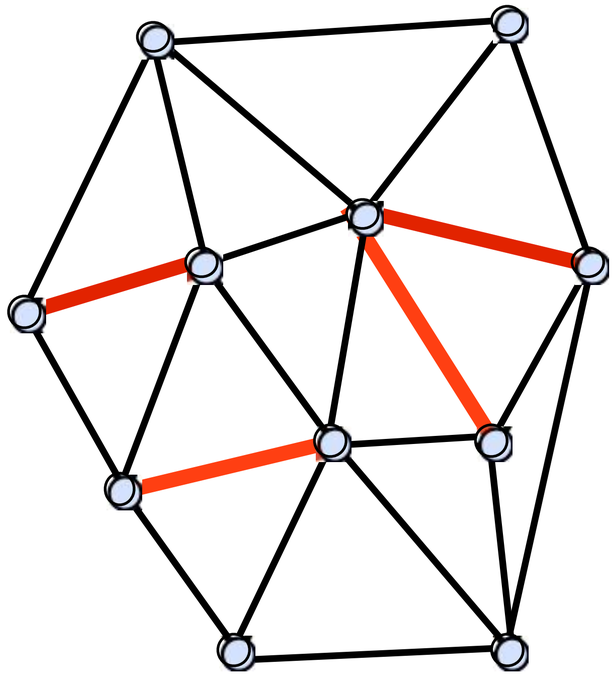
- A finite set M of closed, simple polygons Q_i is a polygonal mesh
- The intersection of two polygons in M is either empty, a vertex, or an edge (for triangle meshes)

$$M = \langle V, E, F \rangle$$

vertices edges faces

Slides credit: Jiajun Wu

Triangulation

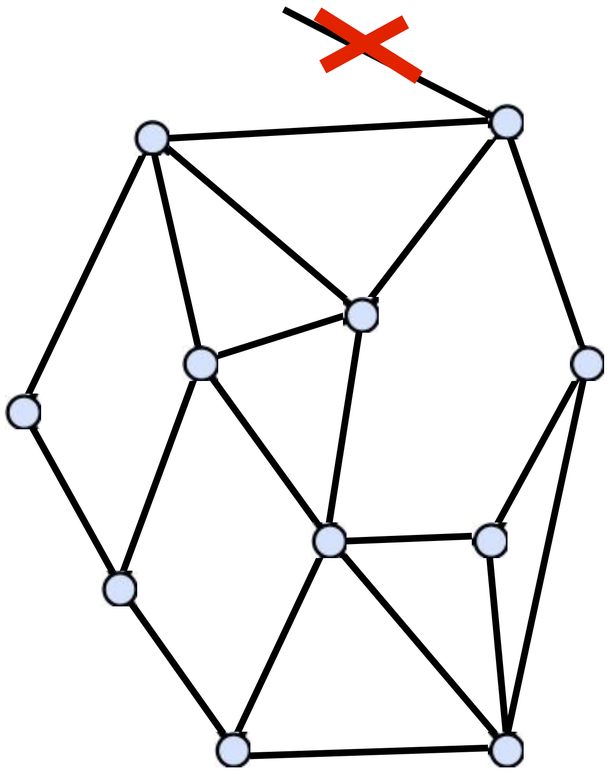


- Polygonal mesh where every face is a triangle

- Simplifies data structures
- Simplifies rendering
- Simplifies algorithms
- Each face planar and convex
- Any polygon can be triangulated

Slides credit: Jiajun Wu

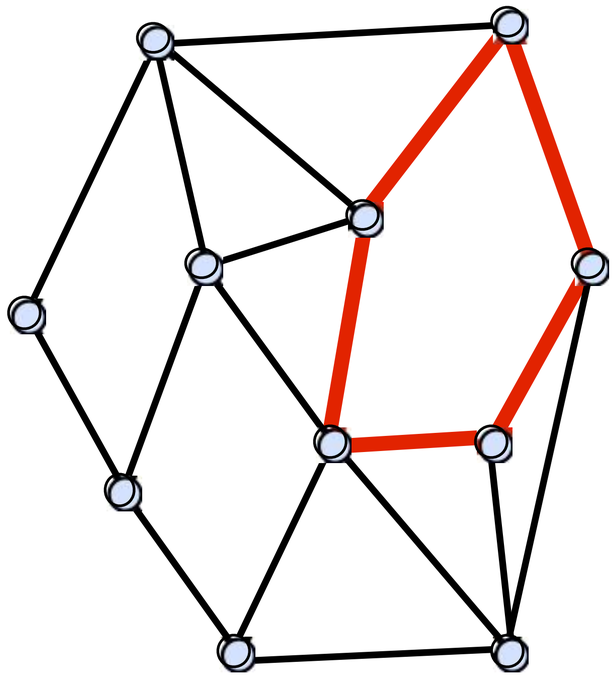
Polygonal Mesh



- A finite set M of closed, simple polygons Q_i is a polygonal mesh
- The intersection of two polygons in M is either empty, a vertex, or an edge
- Every edge belongs to at least one polygon

Slides credit: Jiajun Wu

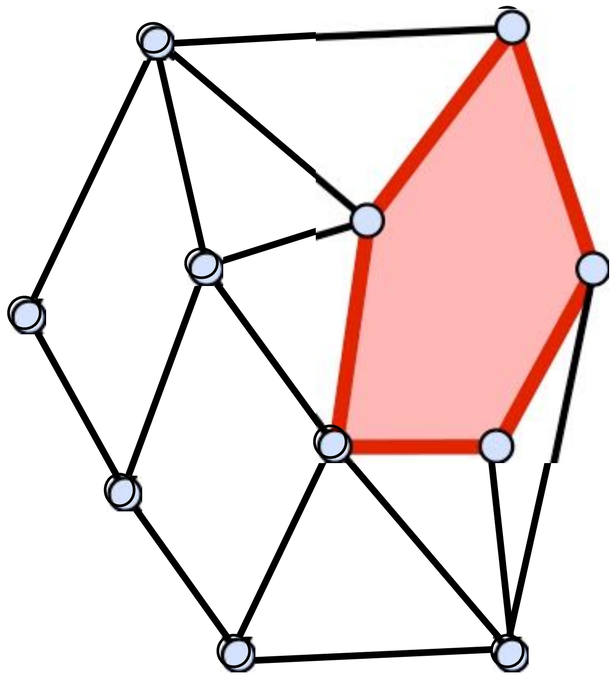
Polygonal Mesh



- A finite set M of closed, simple polygons Q_i is a polygonal mesh
- The intersection of two polygons in M is either empty, a vertex, or an edge
- Every edge belongs to at least one polygon
- Each Q_i defines a **face** of the polygonal mesh

Slides credit: Jiajun Wu

Polygonal Mesh

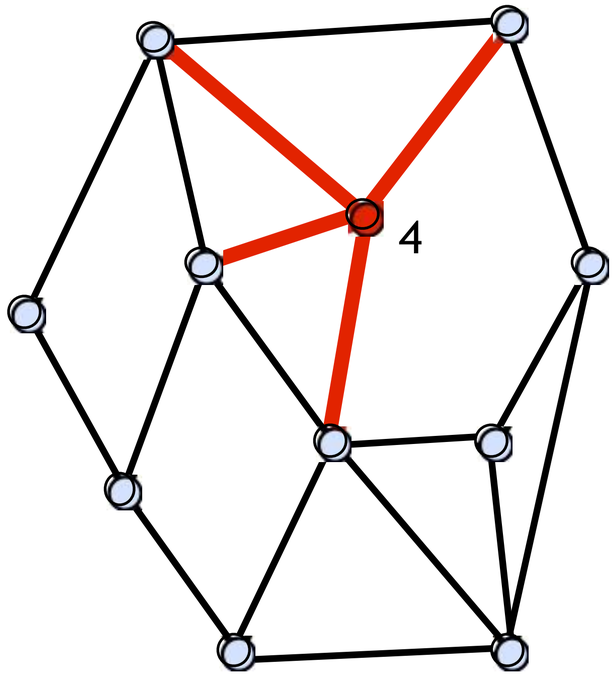


- A finite set M of closed, simple polygons Q_i is a polygonal mesh
- The intersection of two polygons in M is either empty, a vertex, or an edge
- Every edge belongs to at least one polygon
- Each Q_i defines a face of the polygonal mesh

Slides credit: Jiajun Wu

Polygonal Mesh

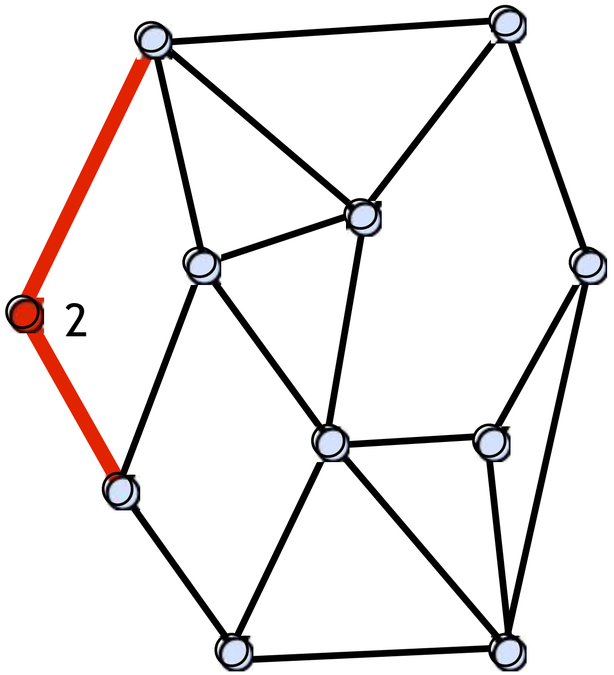
- Vertex **degree** or **valence** = number of incident edges



Slides credit: Jiajun Wu

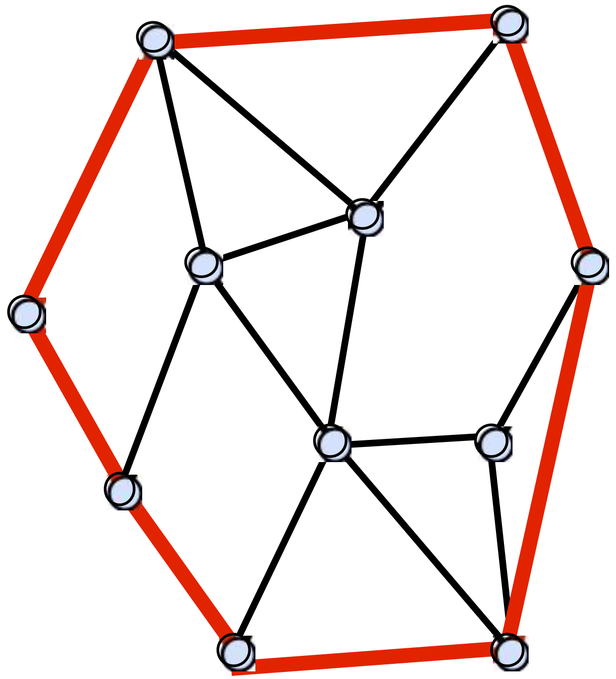
Polygonal Mesh

- Vertex degree or valence = number of incident edges

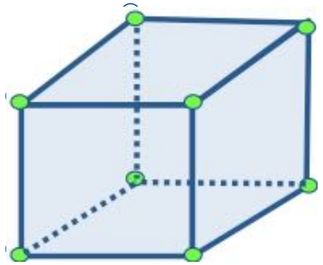


Slides credit: Jiajun Wu

Polygonal Mesh



- Boundary: the set of all edges that belong to only one polygon
  - Either empty or forms closed loops
  - If empty, then the polygonal mesh is closed



Slides credit: Jiajun Wu

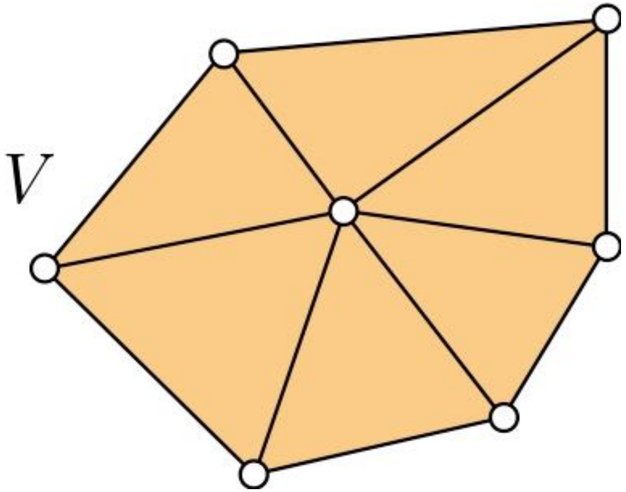
Triangle Mesh

- Connectivity: vertices, edges, triangles
- Geometry: vertex positions

$$V = \{v_1, \ldots, v_n\}$$

$$E = \{e_1, \ldots, e_k\}, \quad e_i \in V \times V$$

$$F = \{f_1, \ldots, f_m\}, \quad f_i \in V \times V \times V$$



Slides credit: Jiajun Wu

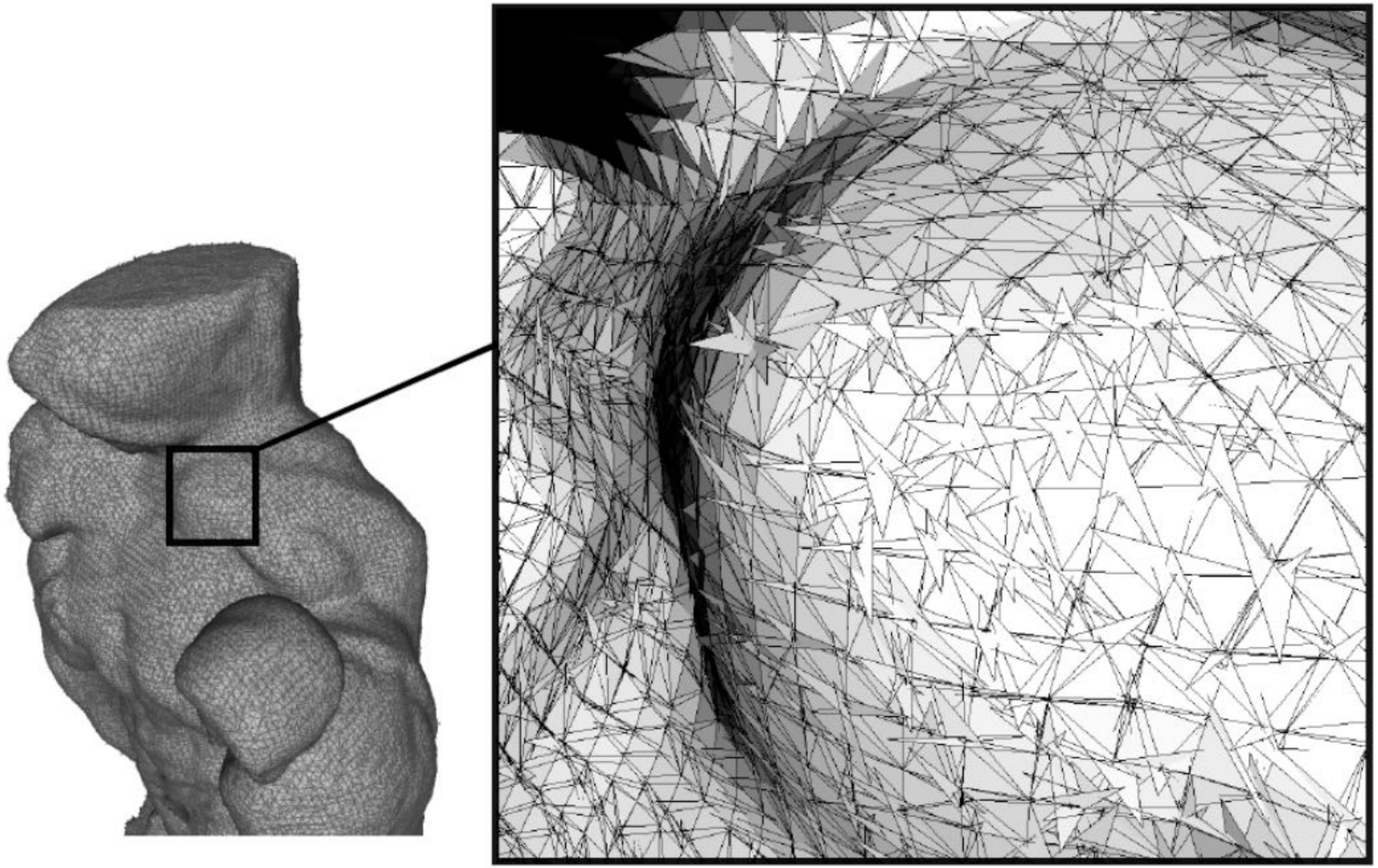
Triangle Mesh

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

Plus manifold conditions

Slides credit: Hao Su

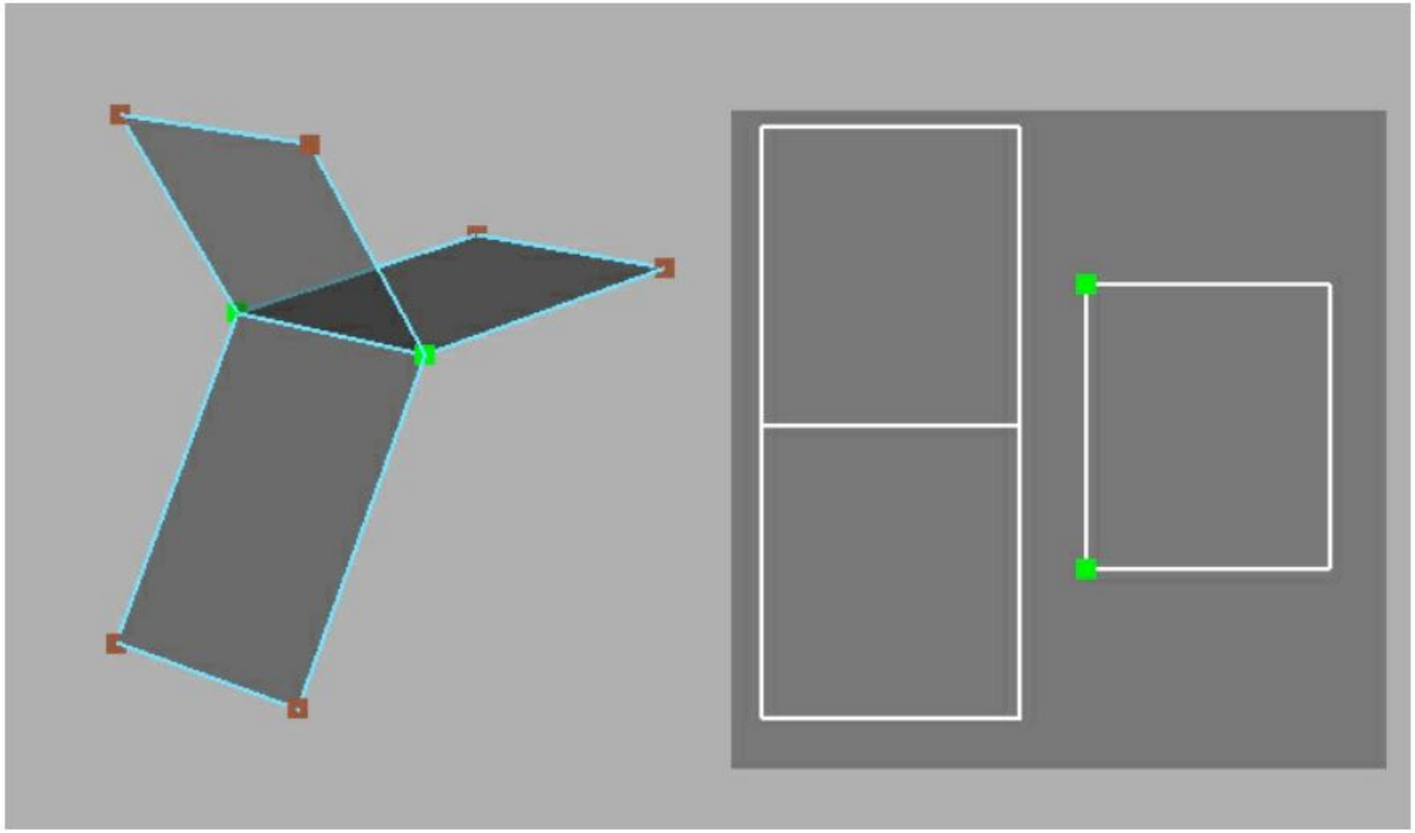
Bad Surfaces



http://igl.ethz.ch/projects/parameterization/rangemap-param/rangemap-param.pdf

Slides credit: Hao Su

Nonmanifold Edge



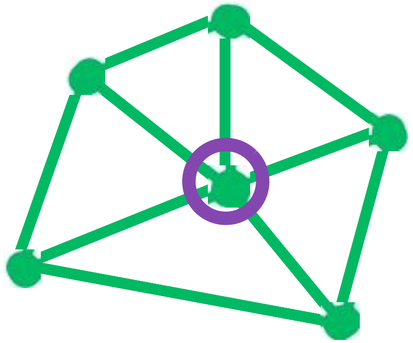
http://blog.mixamo.com/wp-content/uploads/2011/01/nonmanifold.jpg

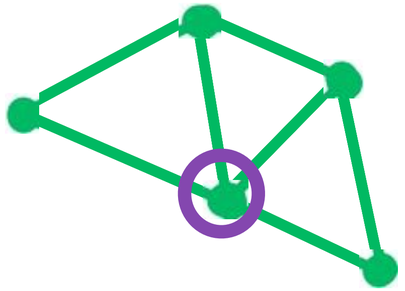
Slides credit: Hao Su

Manifold Mesh

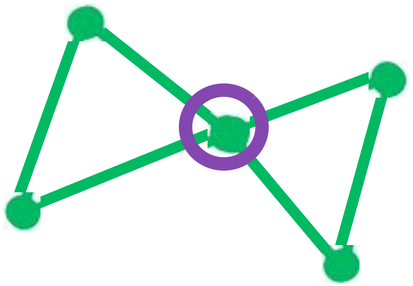
1. Each edge is incident to one or two faces

2. Faces incident to a vertex form a closed or open fan





This is not a fan:



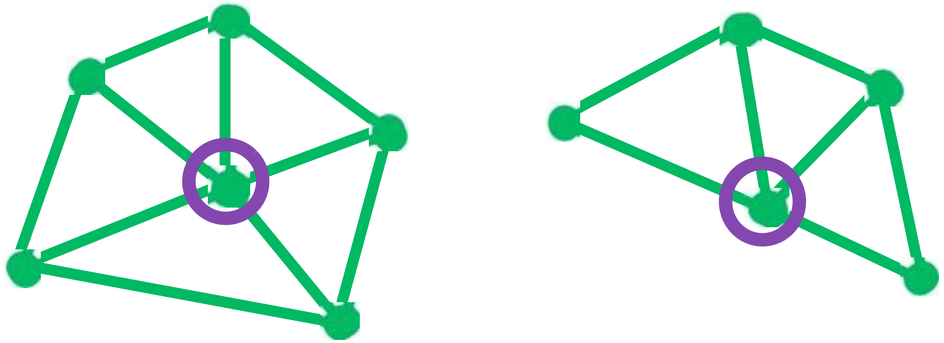
http://www.cs.mtu.edu/~shene/COURSES/cs3621/SLIDES/Mesh.pdf

Slides credit: Hao Su

Manifold Mesh

1. Each edge is incident to one or two faces

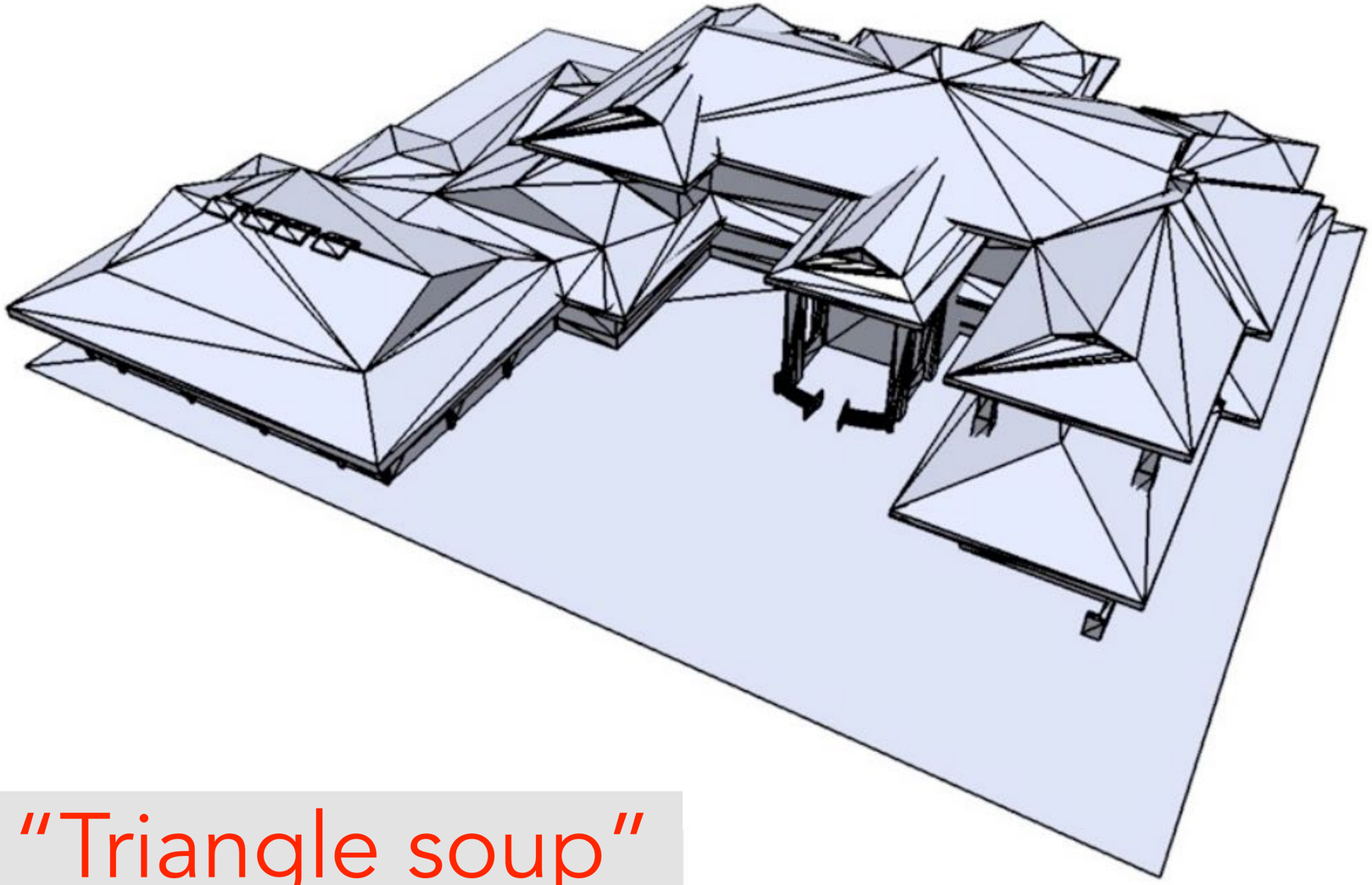
2. Faces incident to a vertex form a closed or open fan



Assume meshes are manifold (for now)

http://www.cs.mtu.edu/~shene/COURSES/cs3621/SLIDES/Mesh.pdf

Slides credit: Hao Su

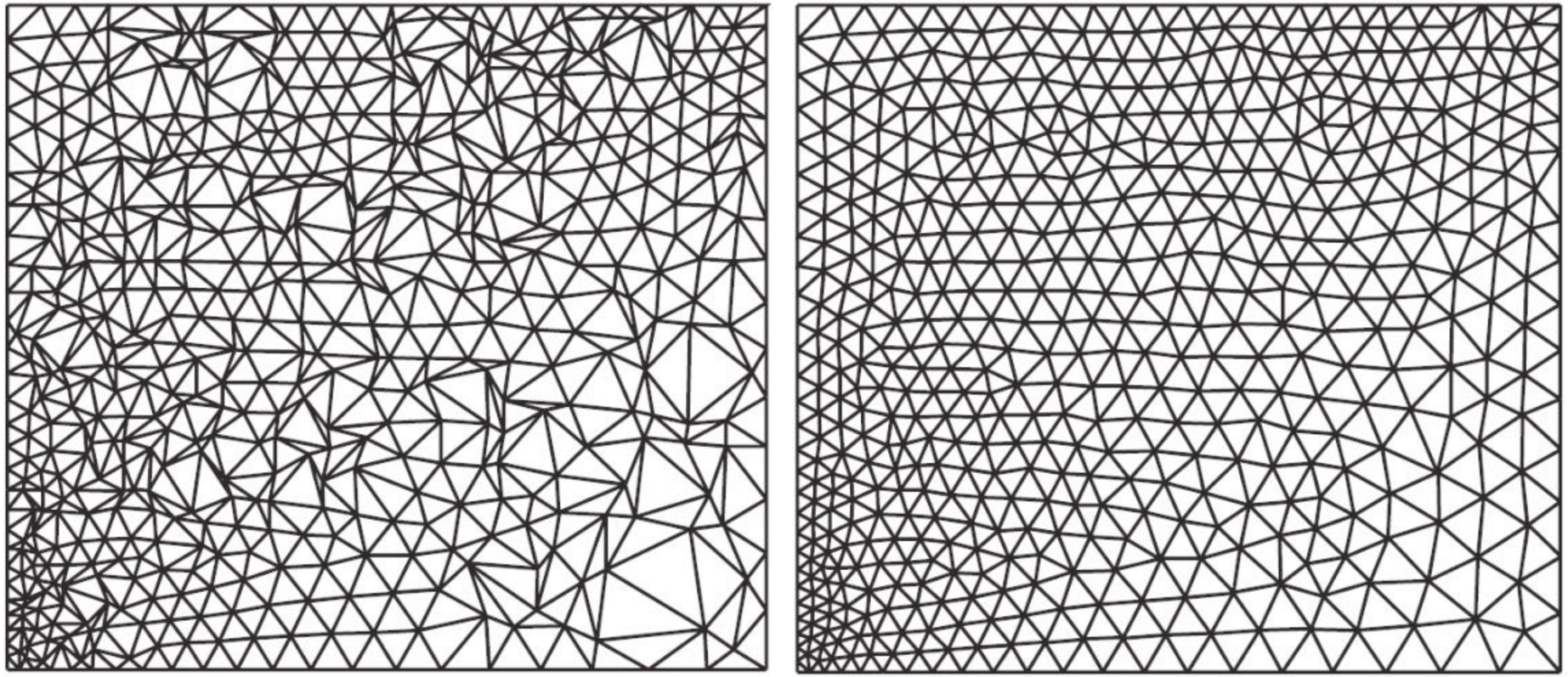


"Triangle soup"

http://www.pmp-book.org/download/slides/Representations.pdf

Slides credit: Hao Su

Bad Meshes

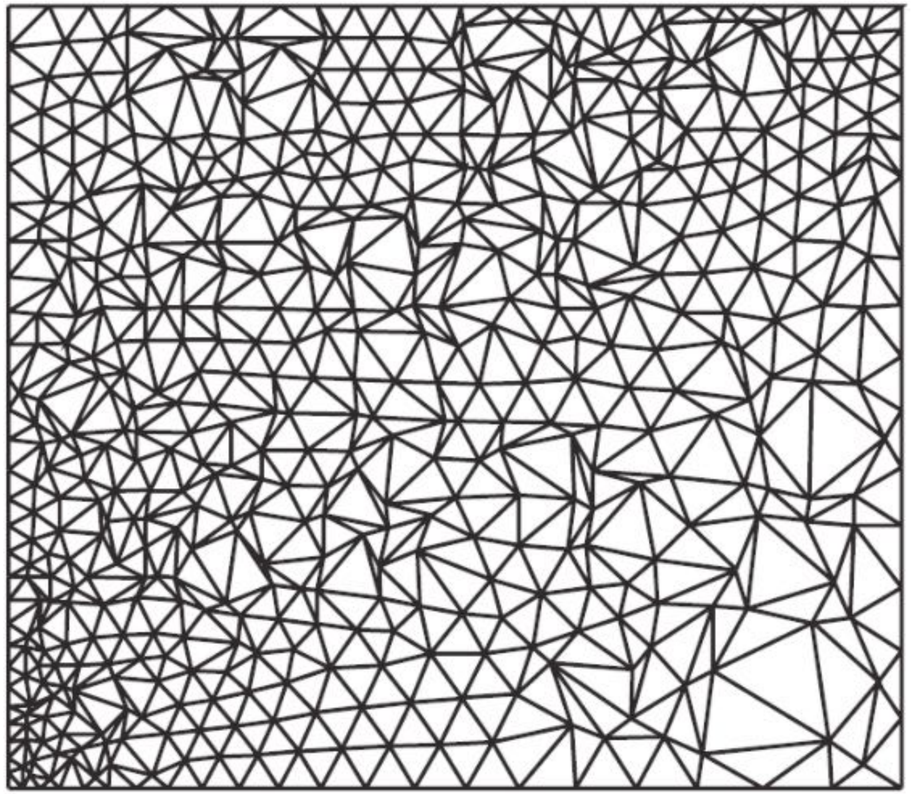


Nonuniform
areas and angles

http://www.sciencedirect.com/science/article/pii/S0168874X06000795

Slides credit: Hao Su

Why is Meshing an Issue?

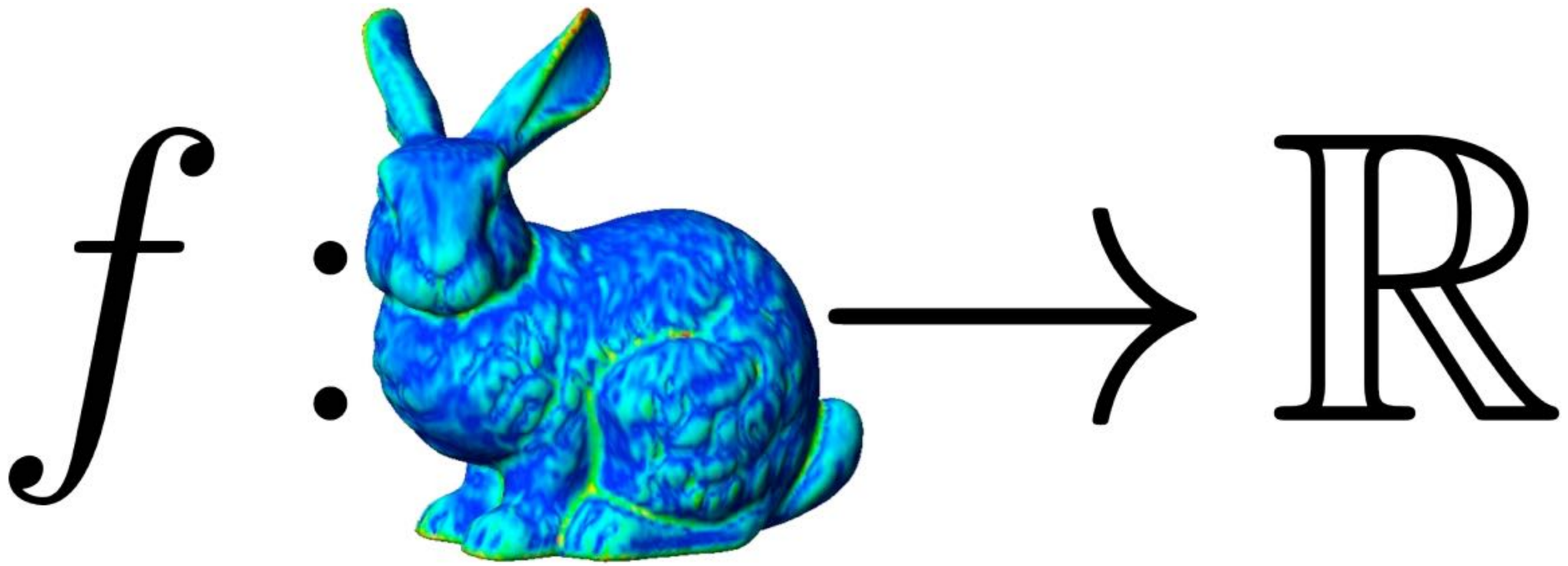


How do you interpret
one value per vertex?

http://www.sciencedirect.com/science/article/pii/S0168874X06000795

Slides credit: Hao Su

Assume Storing Scalar Functions on Surface



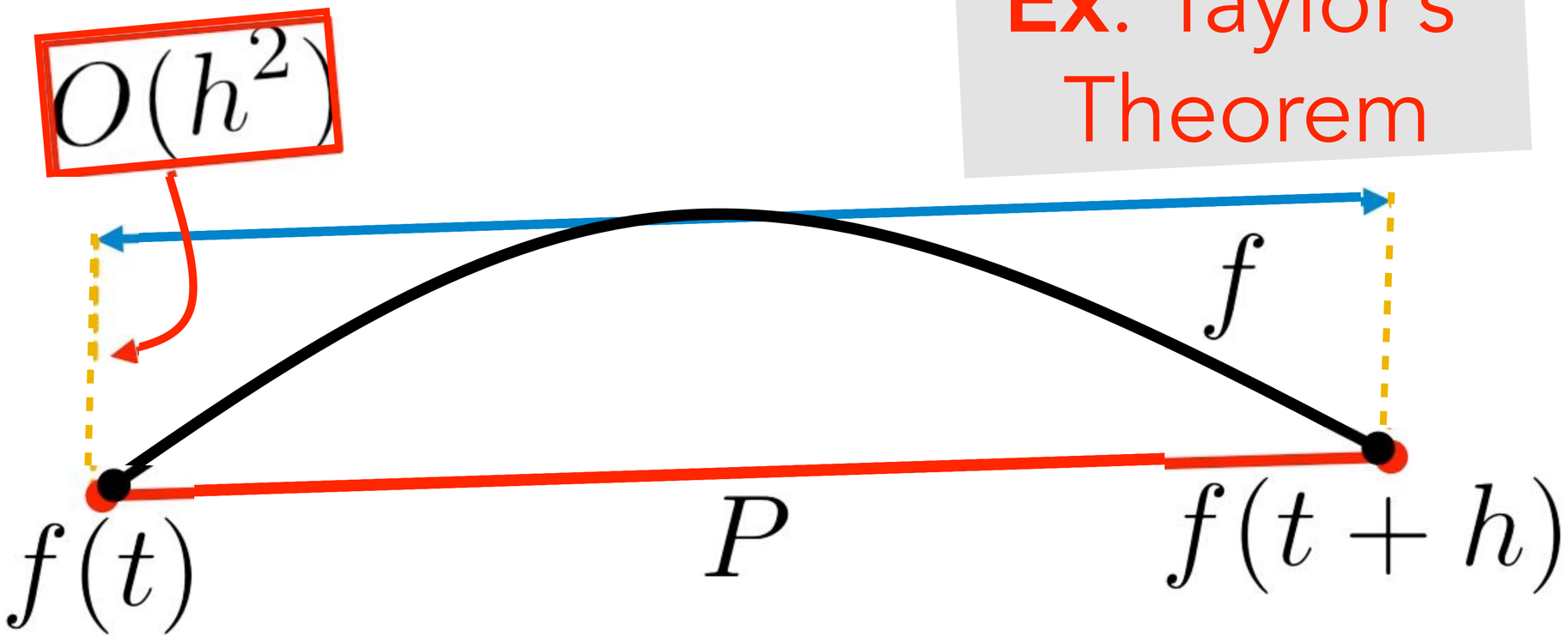
http://www.ieeta.pt/polymeco/Screenshots/PolyMeCo_OneView.jpg

Map points to real numbers

Slides credit: Hao Su

Approximation Properties

Ex: Taylor's Theorem

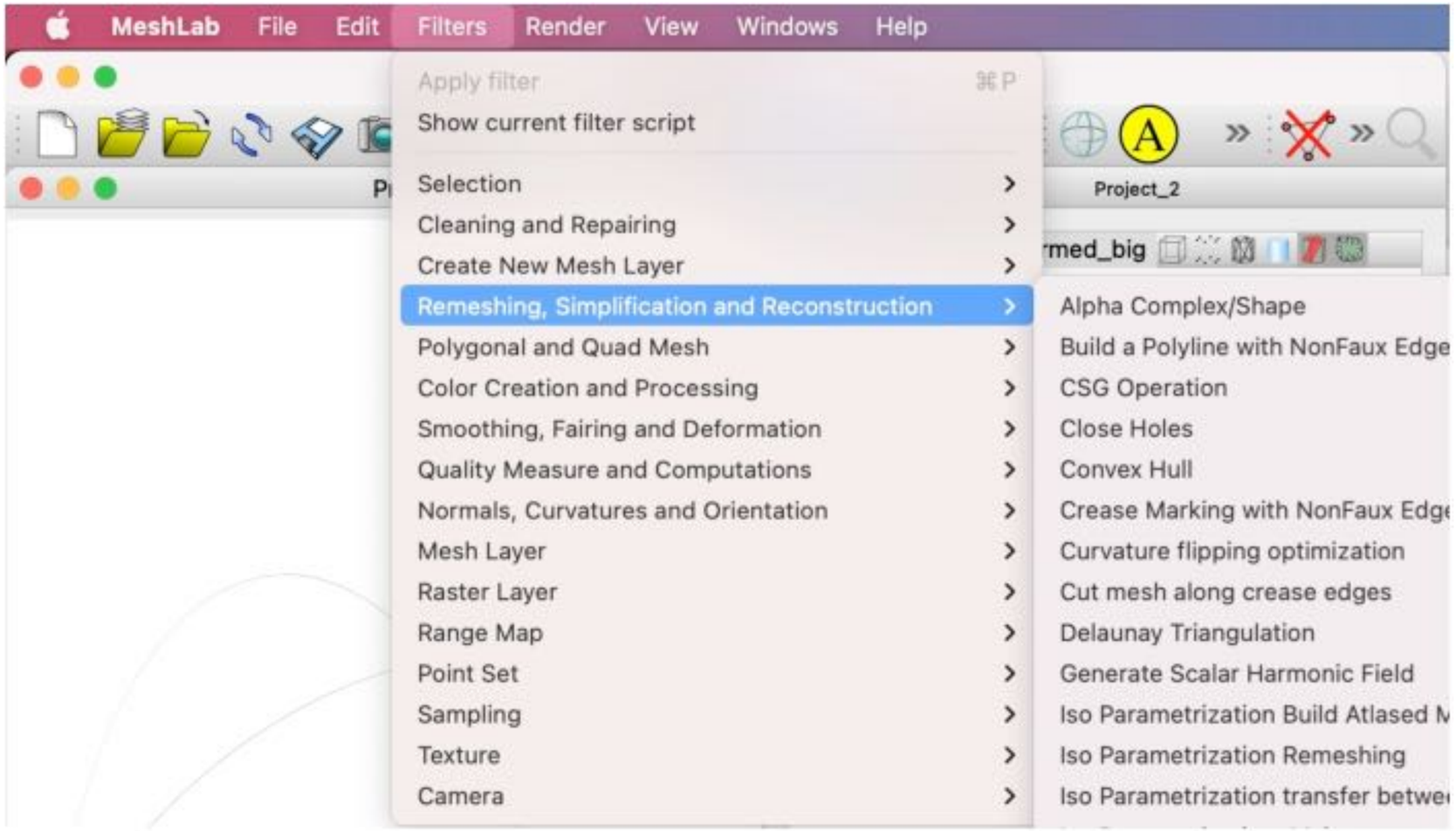


f: functions defined at vertices
(e.g., Gaussian curvature)

Slides credit: Hao Su

Techniques to Improve Mesh Quality

- Cleaning
- Repairing
- Remeshing
- …



Slides credit: Hao Su

Outline

- Rasterized 3D representations
- ***Mesh***
  - Representation
  - ***Storage***
  - Curvature computation
- Point cloud
- Implicit representations

Data Structures for Surfaces

What should be stored?

- Geometry: 3D coordinates
- Topology
- Attributes
  - Normal, color, texture coordinates
  - Per vertex, face, edge

Slides credit: Hao Su

Simple Data Structures: Triangle List

- STL format (used in CAD)
- Storage
  - Face: 3 positions
- No connectivity information

Triangles			
0	x0	y0	z0
1	x1	x1	z1
2	x2	y2	z2
3	x3	y3	z3
4	x4	y4	z4
5	x5	y5	z5
6	x6	y6	z6
…	…	…	…

Slides credit: Hao Su

Simple Data Structures: Indexed Face Set

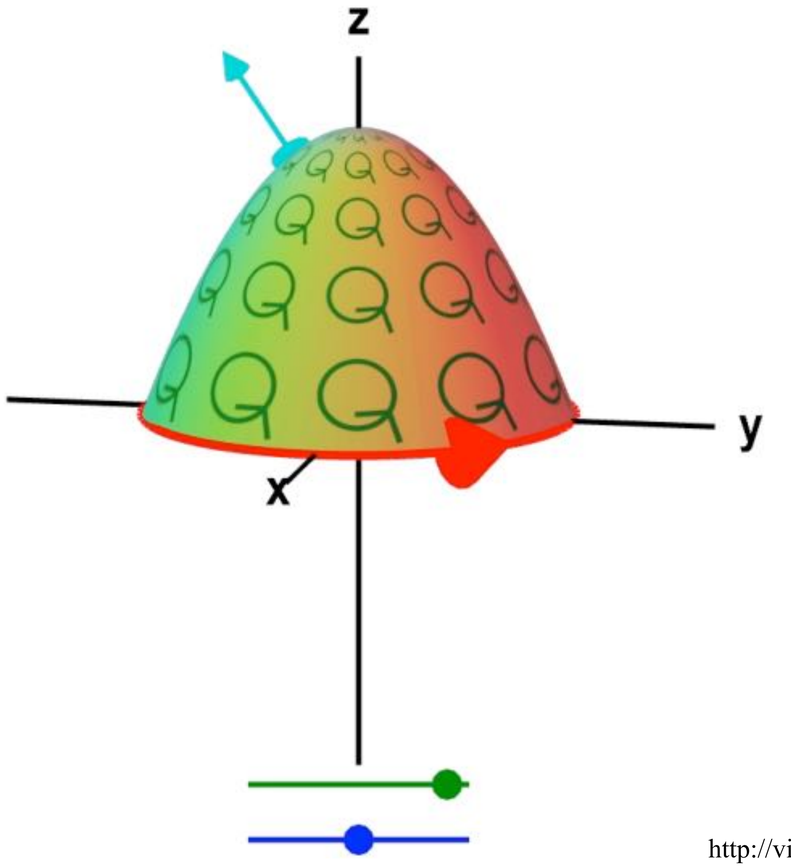
- Used in formats
  - OBJ, OFF, WR L
- Storage
  - Vertex: position
  - Face: vertex indices
  - Convention is to save vertices in counter-clockwise order for normal computation

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
…	…	…	…

Slides credit: Hao Su

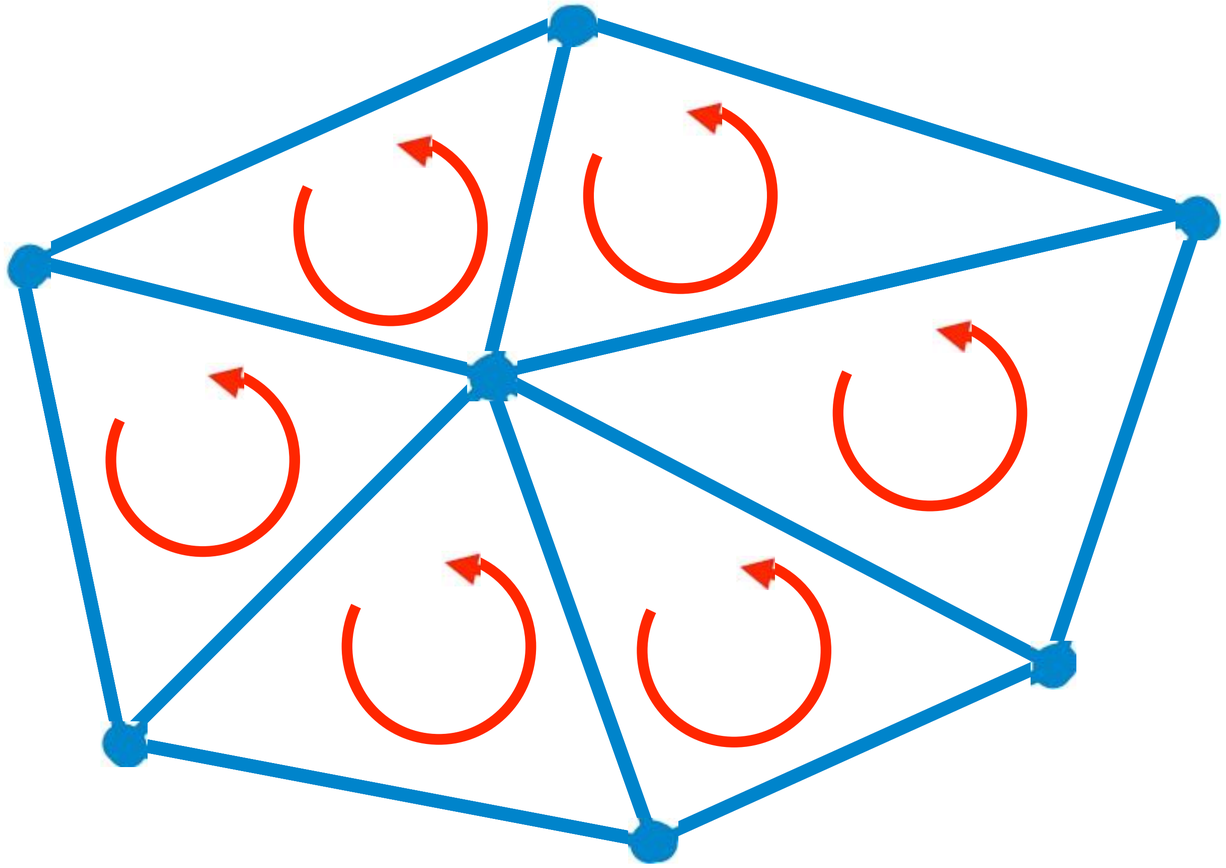
Right-Hand Rule



http://viz.aset.psu.edu/gho/sem_notes/3d_fundamentals/html/3d_coordinates.html
http://mathinsight.org/stokes_theorem_orientation

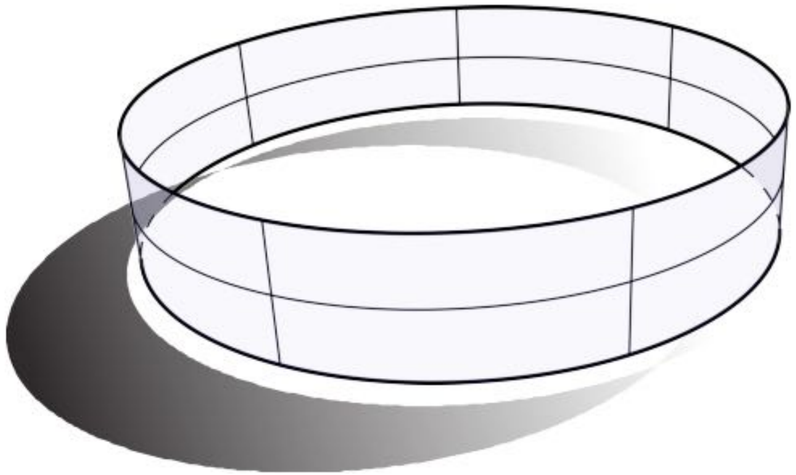
Slides credit: Hao Su

Normal Computation

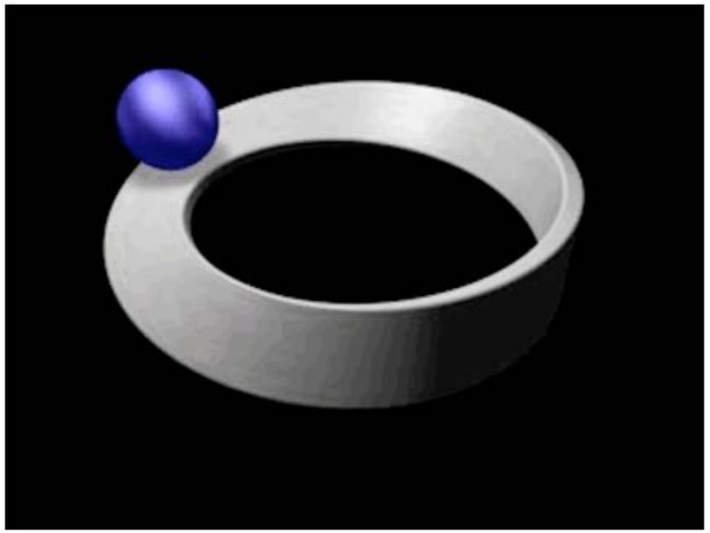


Slides credit: Hao Su

Orientability



orientable



non-orientable

http://blog.lightformshop.com/the-kepler-collection-nemo-math-inspired-wonder/mobius-strip-sphere-2/

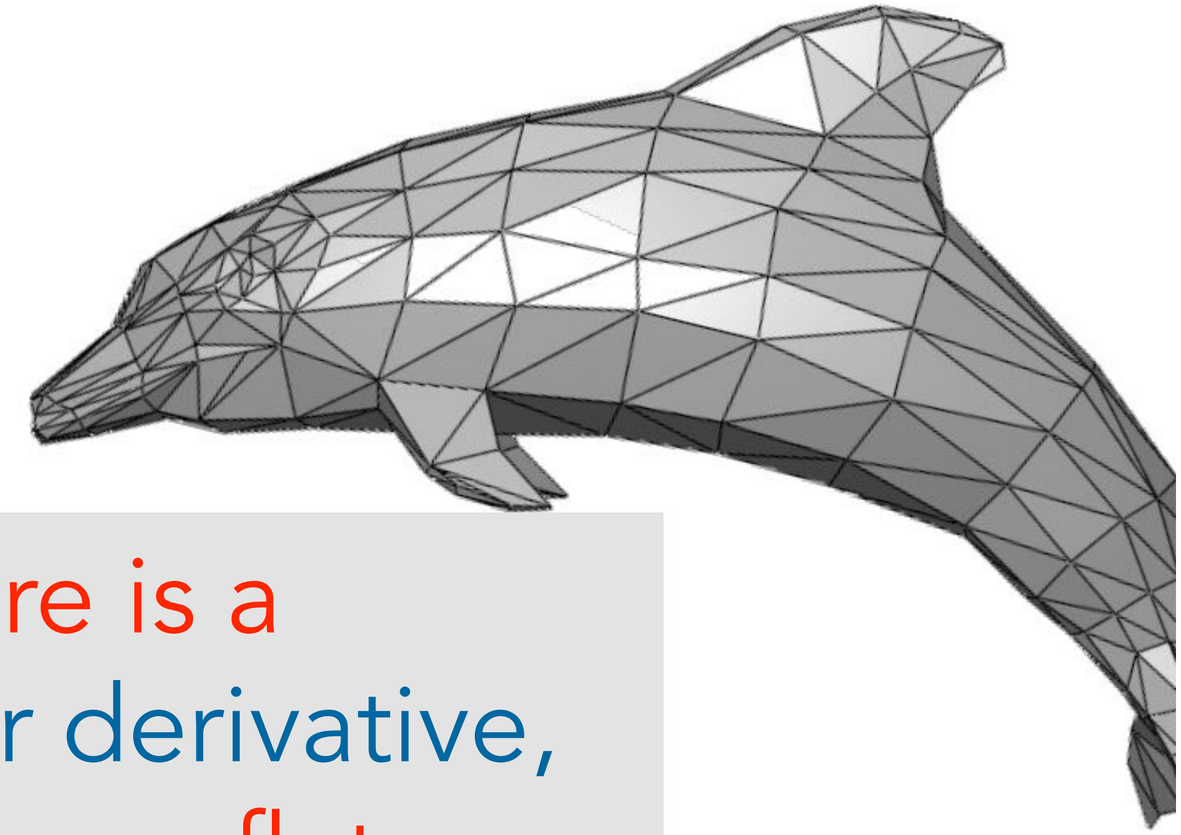
Summary of Polygonal Meshes

- Polygonal meshes are piece-wise linear approximation of smooth surfaces
- Good triangulation is important (manifold, equi-length)
- Vertices, edges, and faces are basic elements
- While real-data 3D are often point clouds, meshes are quite often used to visualize 3D and generate ground truth for machine learning algorithms

Outline

- Rasterized 3D representations
- ***Mesh***
  - Representation
  - Storage
  - ***Curvature computation***
- Point cloud
- Implicit representations

Challenge on Meshes



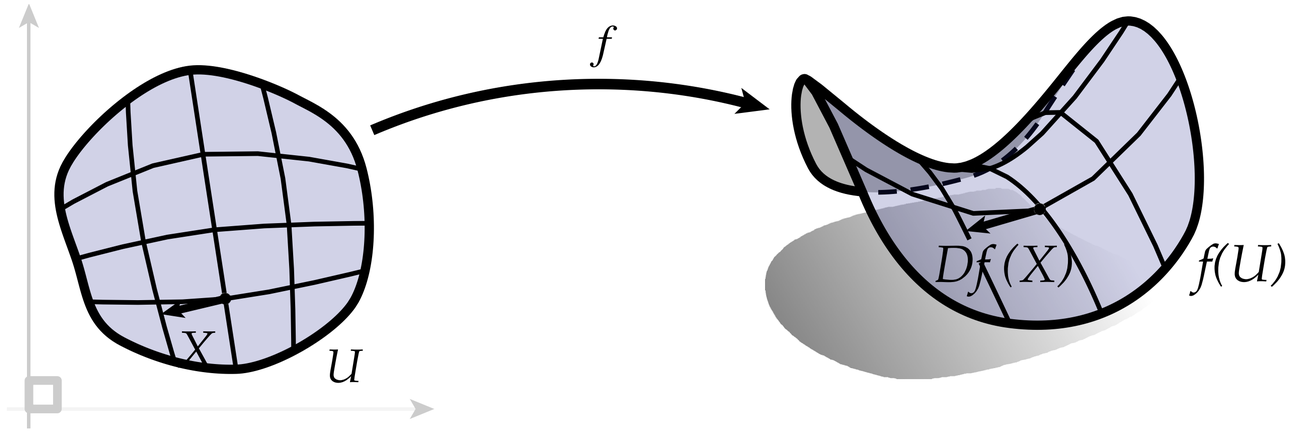
Curvature is a
second-order derivative,
but triangles are flat.

http://upload.wikimedia.org/wikipedia/commons/f/fb/Dolphin_triangle_mesh.png

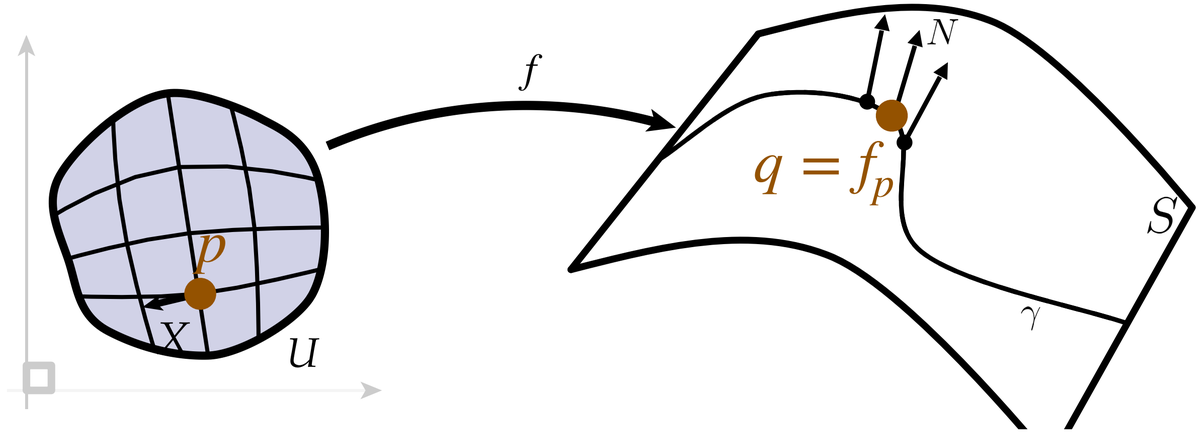
Slides credit: Hao Su

Differential Map in a Nutshell

Differential Map Df:



Differential of Normal DN:

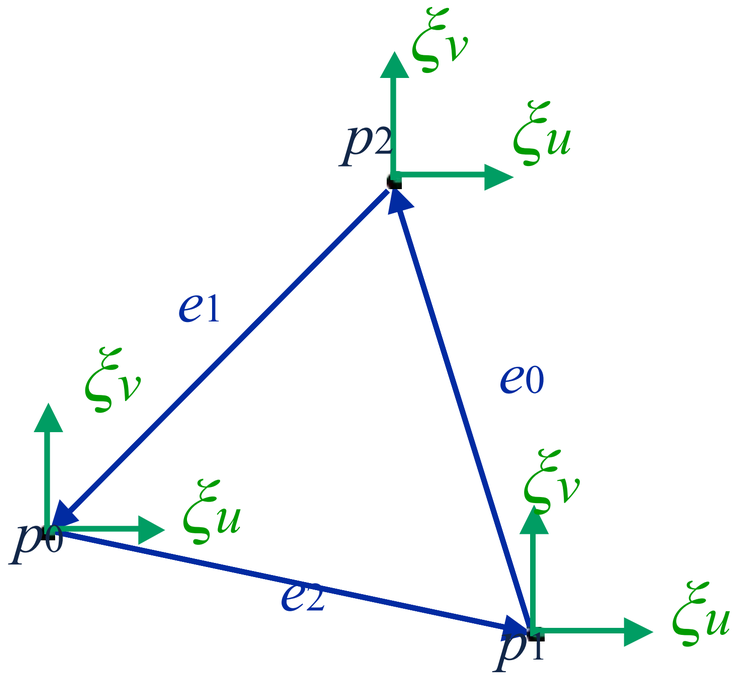


Shape Operator S:

$$\exists S \in \mathbb{R}^{2\times 2} \text{ such that } DN_p = Df_p S$$

If we have S , we can compute principal curvatures!

Rusinkiewicz's Method



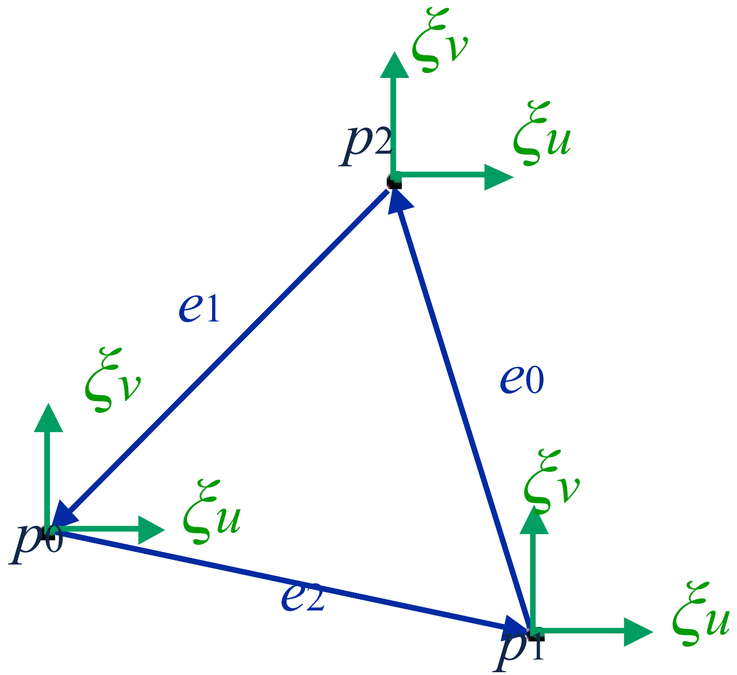
Assume a local $f: U \rightarrow \mathbb{R}^3$ at a small triangle

Assume that $\mathbf{T}\ p_i$'s are roughly parallel

Assume that $Df[\ uv\ ] = u\ \xi_u + v\ \xi_v$, i.e., $Df = [\ \xi_u\ ,\ \xi_v\ ]$
(U is also aligned with $\mathbf{T}\ p_i$'s)

Slides credit: Hao Su

Rusinkiewicz's Method



Shape operator S satisfies: $DN = Df \cdot S$.

$$\because Df = [\, \xi_u , \xi_v \,] , \ \therefore S = Df^T DN$$

How to estimate S ?

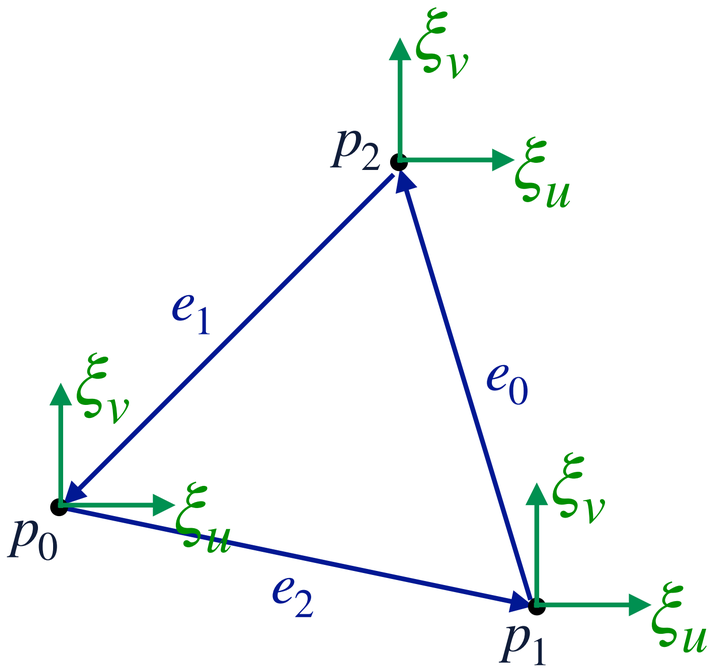
Slides credit: Hao Su

Rusinkiewicz's Method

$$Df^T \left(DN \begin{bmatrix} u \\ v \end{bmatrix} \right) \approx Df^T \Delta \vec{n} \implies S \begin{bmatrix} u \\ v \end{bmatrix} \approx Df^T \Delta \vec{n}$$

$$\because \; Df \begin{bmatrix} u \\ v \end{bmatrix} = Y \in \mathbf{T}(\mathbb{R}^3) \text{ and } Df = [\vec{\xi}_u, \vec{\xi}_v] \quad \therefore \begin{bmatrix} u \\ v \end{bmatrix} = Df^T Y$$

$$\therefore \; S[Df]^T Y \approx [Df]^T \Delta \vec{n}$$



Slides credit: Hao Su

Rusinkiewicz's Method

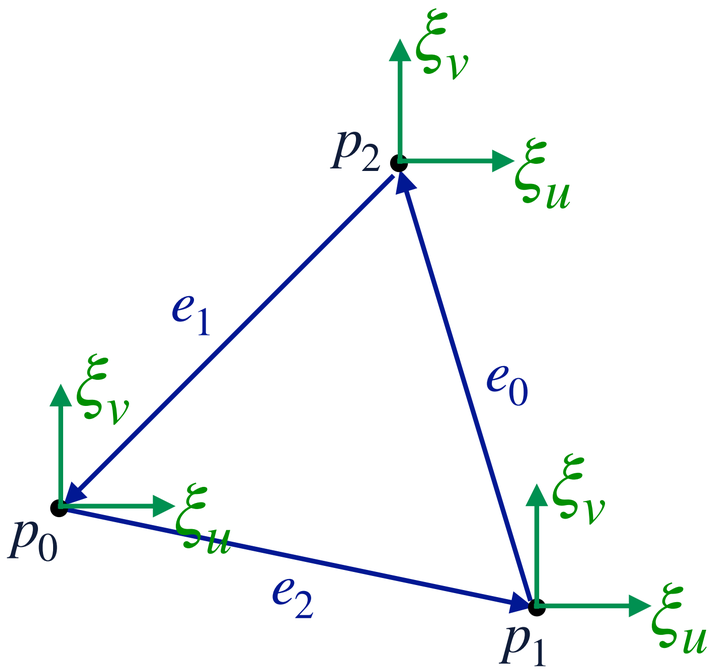
$$Df^T\left(DN\begin{bmatrix}u\\v\end{bmatrix}\right) \approx Df^T\Delta\vec{n} \implies S\begin{bmatrix}u\\v\end{bmatrix} \approx Df^T\Delta\vec{n}$$

$$\because\ Df\begin{bmatrix}u\\v\end{bmatrix} = Y \in \mathbf{T}(\mathbb{R}^3) \text{ and } Df = [\vec{\xi}_u, \vec{\xi}_v] \quad \therefore \begin{bmatrix}u\\v\end{bmatrix} = Df^T Y$$

$$\therefore\ S[Df]^T Y \approx [Df]^T\Delta\vec{n}$$

$$\begin{cases} S[Df]^T e_0 = Df^T(\vec{n}_2 - \vec{n}_1), \\ S[Df]^T e_1 = Df^T(\vec{n}_0 - \vec{n}_2), \\ S[Df]^T e_2 = Df^T(\vec{n}_1 - \vec{n}_0), \end{cases}$$

So we can solve $S \in \mathbb{R}^{2\times 2}$ by least square(6 equations and 4 unknowns)



Slides credit: Hao Su

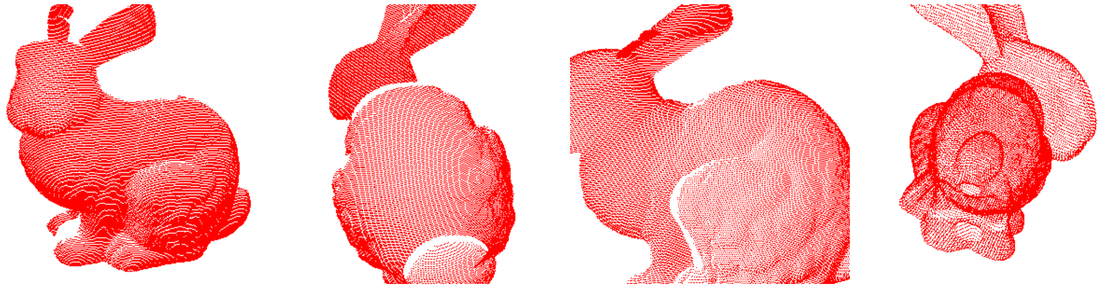
Summary of Mesh Curvature Estimation

- Rusinkiewicz's method is an effective approach for face curvature estimation
  - Szymon Rusinkiewicz, "Estimating Curvatures and Their Derivatives on Triangle Meshes", 3DPVT, 2004
- Good robustness to moderate amount of noise and free of degenerate configurations
- Can be used to compute curvatures for point cloud as well

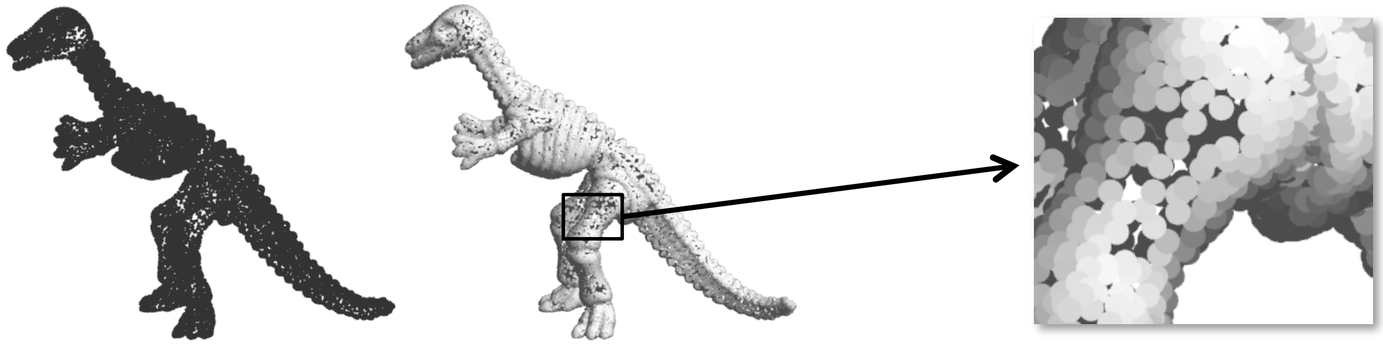
Outline

- Rasterized 3D representations
- Mesh
- ***Point cloud***
  - ***Representation***
  - Sampling points on surfaces
  - Normal computation
- Implicit representations

Point Cloud Representation



- Simplest representation: **only points**, no connectivity
- Collection of (x,y,z) coordinates, possibly with normal
- Points with orientation are called **surfels**

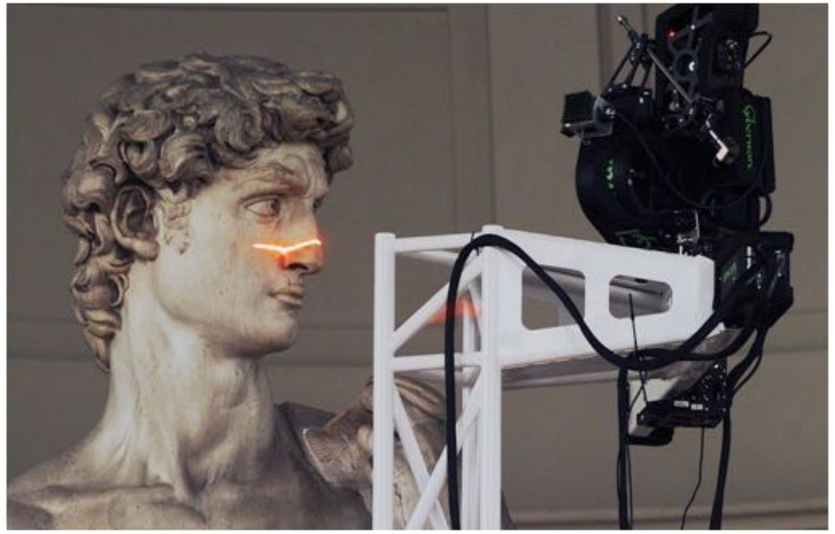


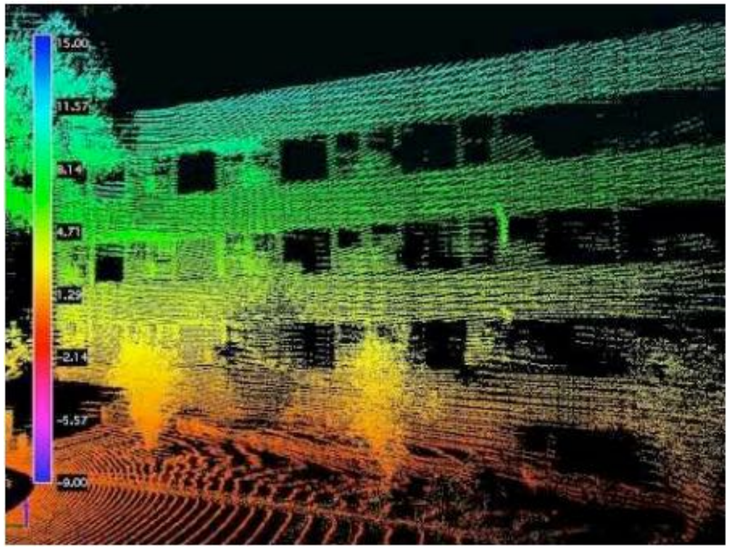
shading needs normals!

Slides credit: Jiajun Wu

Acquiring Point Clouds

- From the real world
  - 3D scanning
    - Data is “striped”
    - Need multiple views to compensate occlusion





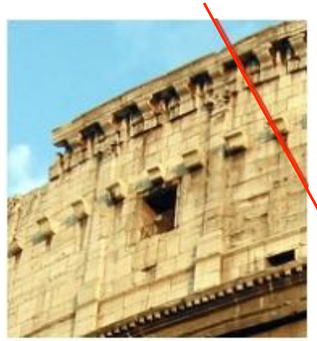
- Many techniques
  - Laser (LIDAR, e.g., StreetView)
  - Infrared (e.g., Kinect)
  - Stereo (e.g., Bundler)
- Many challenges: resolution, occlusion, noise, registration

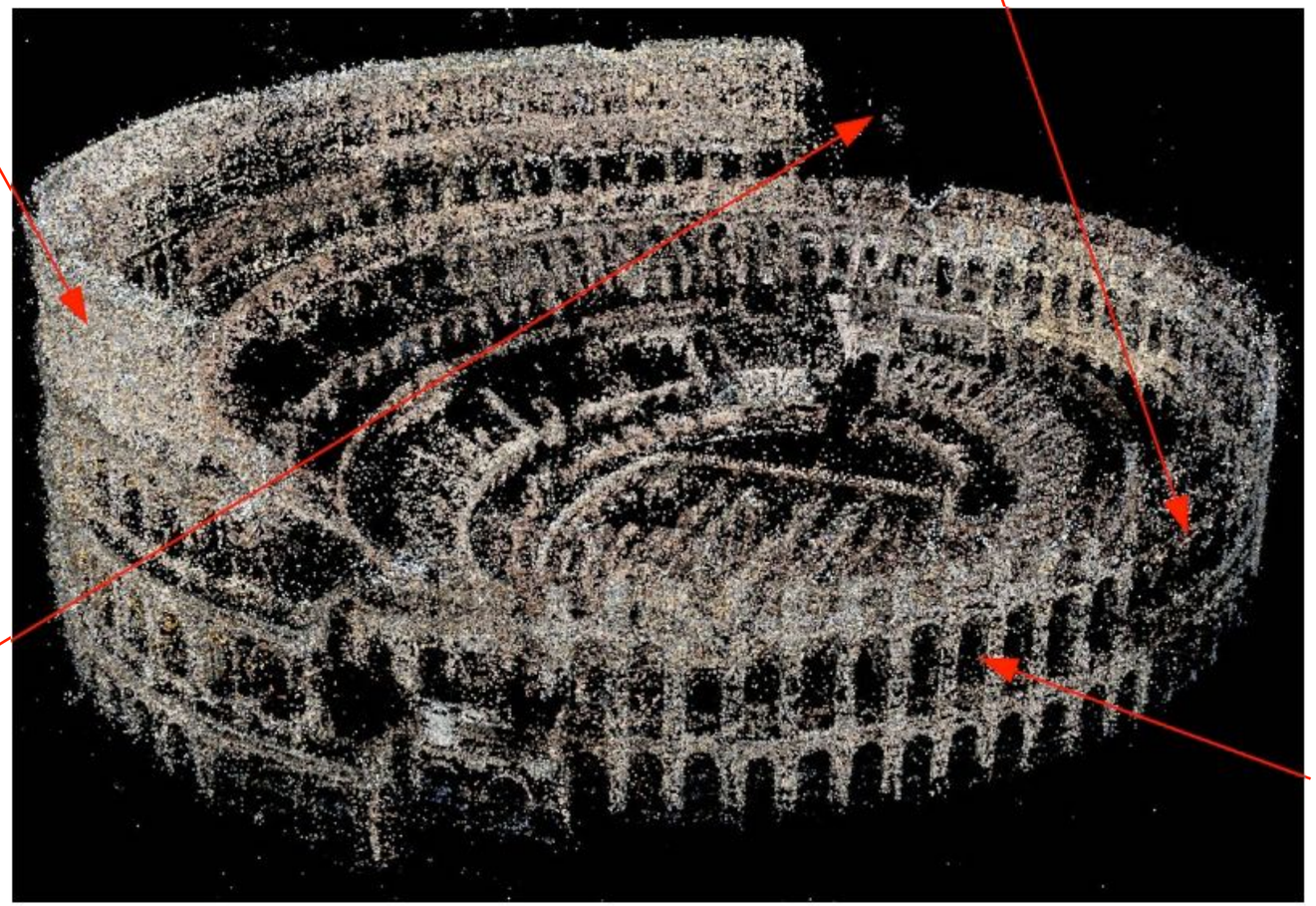
Slides credit: Hao Su

Acquisition Challenges

Noise → Poor detail reproduction

Low resolution further obscures detail





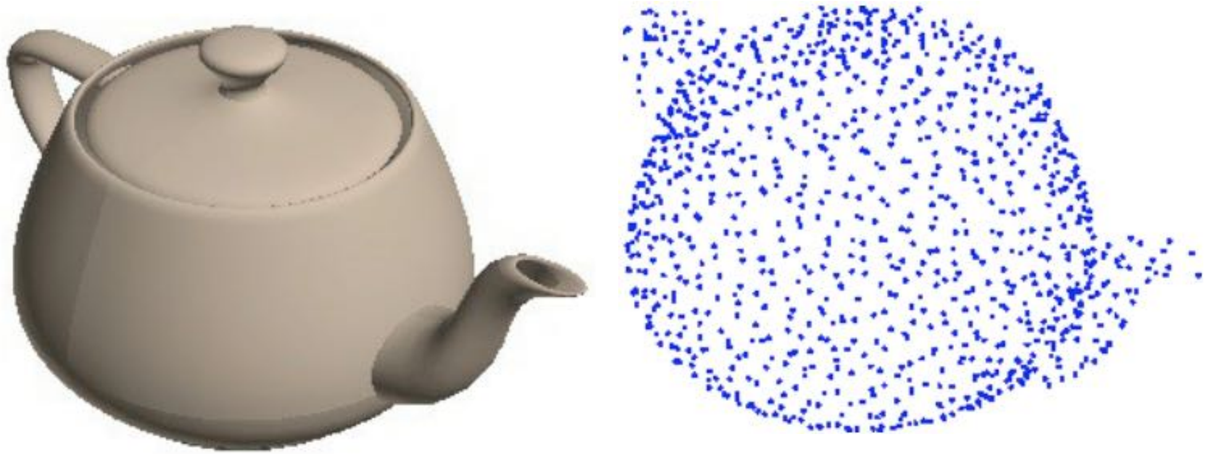
Some data was not properly registered with the rest

Occlusion → Interiors not captured

Slides credit: Hao Su

Acquiring Point Clouds

- From existing virtual shapes



- Why would we want to do this?

Slides credit: Hao Su

Light-weight Shape Representation

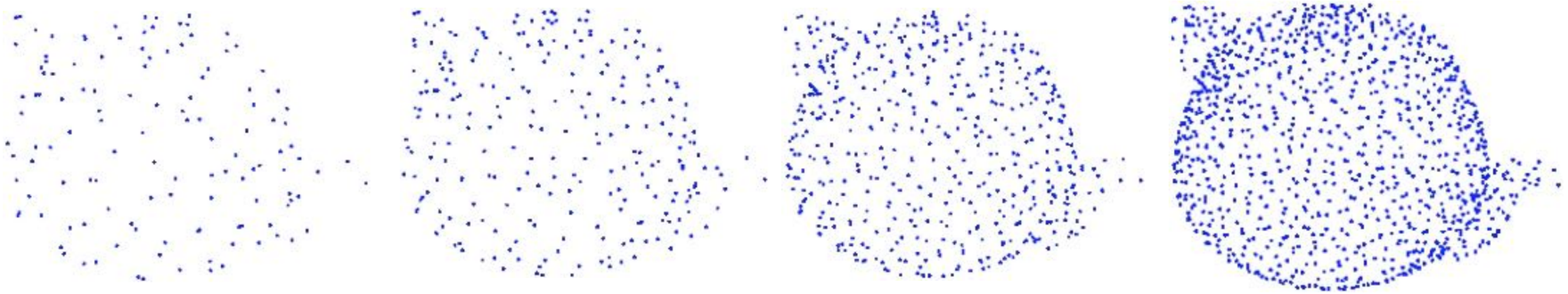
Point cloud:

· Simple to understand

· Compact to store

· Generally easy to build algorithms

Yet already carries rich information!



$N = 125$ $N = 250$ $N = 500$ $N = 1000$

Slides credit: Hao Su

Outline

- Rasterized 3D representations
- Mesh
- ***Point cloud***
  - Representation
  - ***Sampling points on surfaces***
  - Normal computation
- Implicit representations

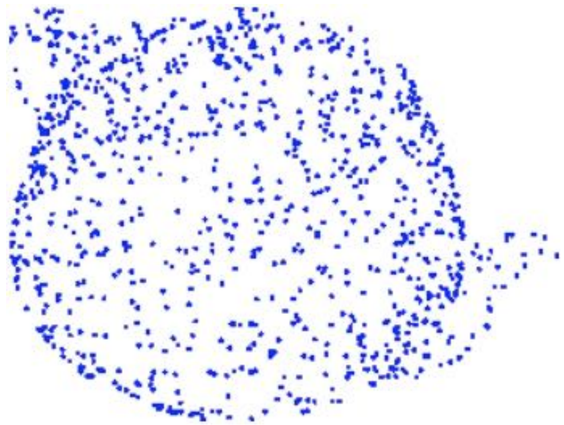
Application-based Sampling

- For storage or analysis purposes (e.g., shape retrieval, signature extraction),
  - the objective is often to preserve surface information as much as possible

- For learning data generation purposes (e.g., sim2real),
  - the objective is often to minimize virtual-real domain gap

Naive Strategy: Uniform Sampling

- Independent identically distributed (i.i.d.) samples by surface area:

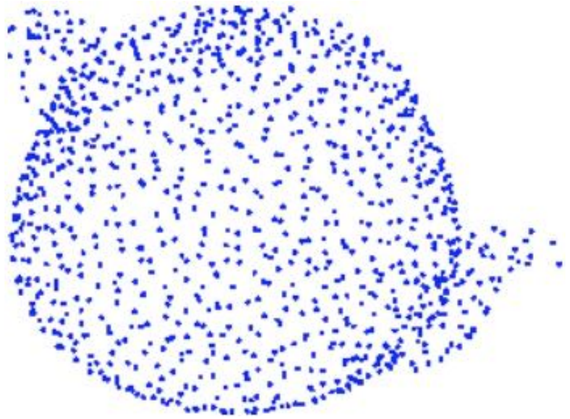


- Usually the easiest to implement
- Issue: Irregularly spaced sampling

Slides credit: Hao Su

Farthest Point Sampling

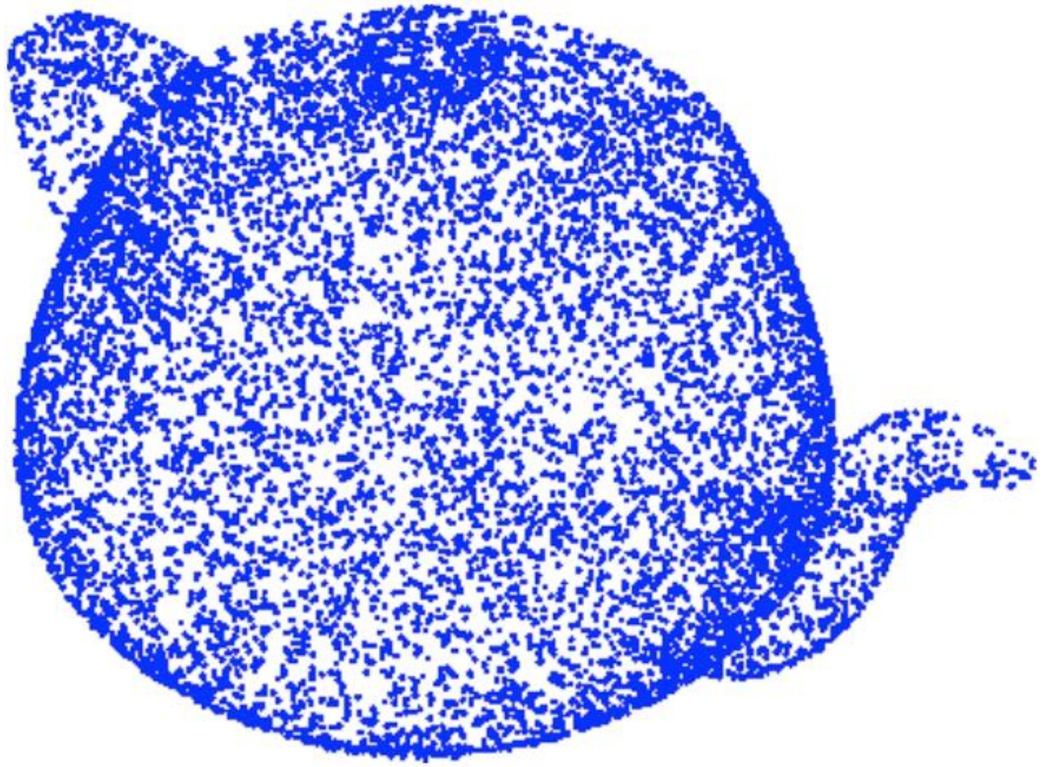
- Goal: Sampled points are far away from each other
- NP-hard problem
- What is a greedy approximation method?



Slides credit: Hao Su

Iterative Furthest Point Sampling

· Step 1: Over sample the shape by any fast method (e.g., uniformly sample N=10,000 i.i.d. samples)



Slides credit: Hao Su

Iterative Furthest Point Sampling

- Step 2: Iteratively select K points

U is the initial big set of points
$S = \{\}$

add a random point from U to S

for i=1 to K
    find a point $u \in U$ with the largest distance to S
    add u to S

Slides credit: Hao Su

Issues Relevant to Speed

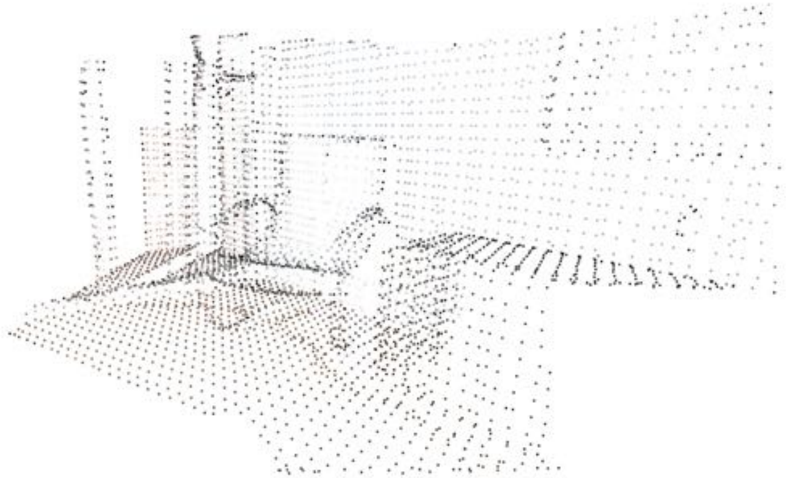
- Theoretically, naive implementation gives $\mathcal{O}(KN)$, but how to improve from $\mathcal{O}(KN)$ is an open question

- **Implementation can cause large speed difference**
  - As this is a serial algorithm in *K*, engineers optimize the efficiency in *N* (computing point-set distance)
  - CPU: Suggest using vectorization (e.g., numpy, scipy.spatial.distance.cdist)
  - GPU: By using shared memory, the complexity can be reduced to $\mathcal{O}(K(N/M + \log M))$, where *M* is the number of threads (*M=512* in practice for modern GPU).

Slides credit: Hao Su

Voxel Downsampling

- Uses a ***regular*** voxel grid to downsample, taking one point per grid
- Allows higher parallelization level
- Generates regularly spaced sampling (with noticeable artifacts)



Credit to: Open3D

Issues Relevant to Speed

- Need to map each point to a bin. Often implemented as adding elements into a hash table

- $\mathcal{O}(N)$ (assuming that the inserting into hash table takes O(1))

- On GPUs, parallelization reduces complexity of
  - Mapping each point to an integer value
  - Assign each value to an index so that the same value shares the same index
  - Aggregate indexes and form the output (called scattering in CUDA)

Slides credit: Hao Su

Application-based Sampling

- For storage or analysis purposes (e.g., shape retrieval, signature extraction),
  - the objective is often to preserve surface information as much as possible

- For learning data generation purposes (e.g., sim2real),
  - the objective is often to minimize virtual-real domain gap

Application-based Sampling

- For storage or analysis purposes (e.g., shape retrieval, signature extraction),
  - the objective is often to preserve surface information as much as possible

- For learning data generation purposes (e.g., sim2real),
  - the objective is often to minimize virtual-real domain gap

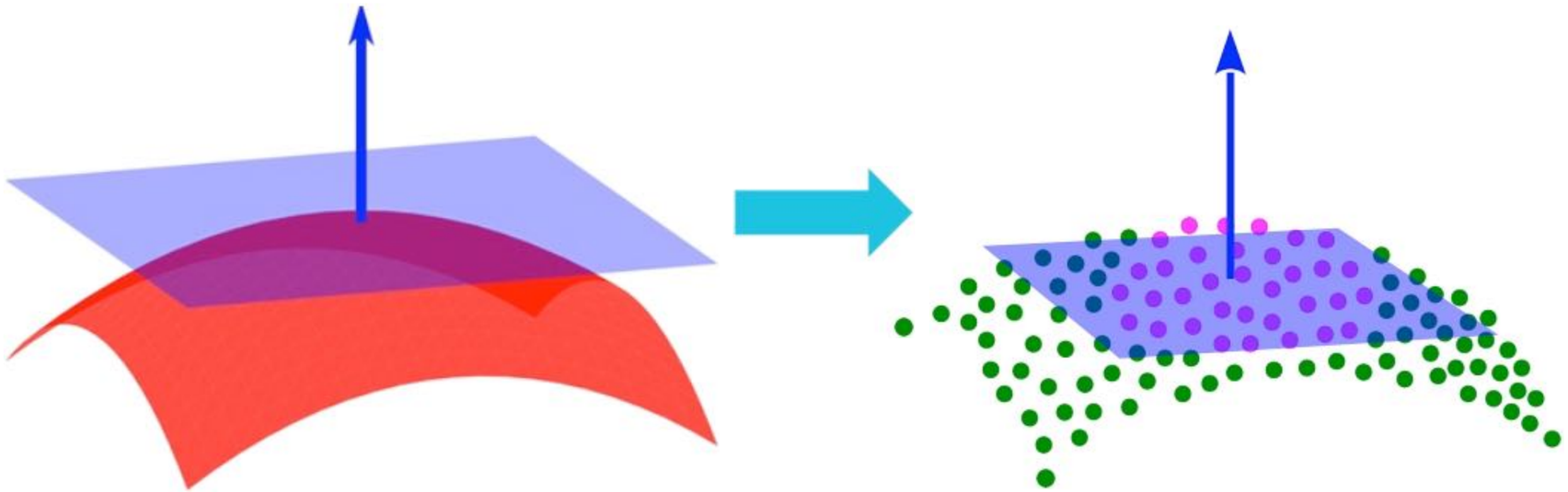
A good research topic
(e.g., GAN? Adversarial training? Differentiable sampling?)

Outline

- Rasterized 3D representations
- Mesh
- ***Point cloud***
  - Representation
  - Sampling points on surfaces
  - ***Normal computation***
- Implicit representations

Estimating Normals

- Plane-fitting: find the plane that best fits the neighborhood of a point of interest



Slides credit: Hao Su

Least-square Formulation

- Assume the plane equation is:

$$w^T(x - c) = 0 \quad \text{with} \quad \|w\| = 1$$

- Plane-fitting solves the least square problem:

$$\text{minimize} \quad \sum_i \|w^T(x_i - c)\|_2^2$$

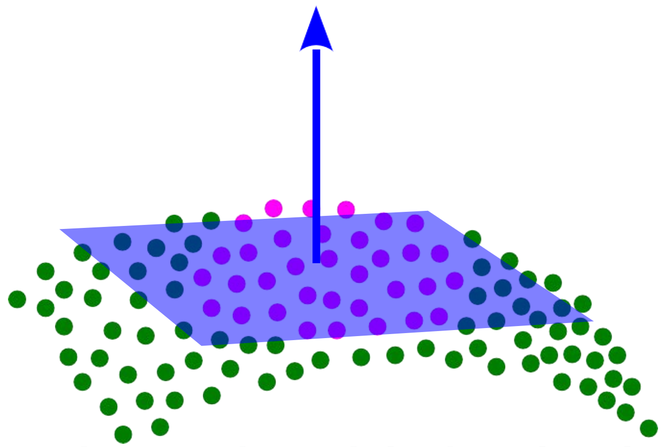
$$\text{subject to} \quad \|w\|^2 = 1$$

where $\{x_i\}$ is the neighborhood of a point x
that you query the normal

Slides credit: Hao Su

Least-square Formulation

- Doing Lagrangian multiplier and the solution is:
  - Let $M = \sum_i (x_i - \bar{x})(x_i - \bar{x})^T$ and $\bar{x} = \frac{1}{n}\sum_i x_i$,
  - w: the smallest eigenvector of M
  - $c = w^T \bar{x}$



- w also corresponds to the third principal component of M (yet another usage of PCA)
  - Where are the first and second principal components?

Slides credit: Hao Su

Summary of Normal Computation

- The normal of a point cloud can be computed through PCA over a local neighborhood
- Remark:
  - The choice of neighborhood size is important
  - When outlier points exist, RANSAC can improve quality

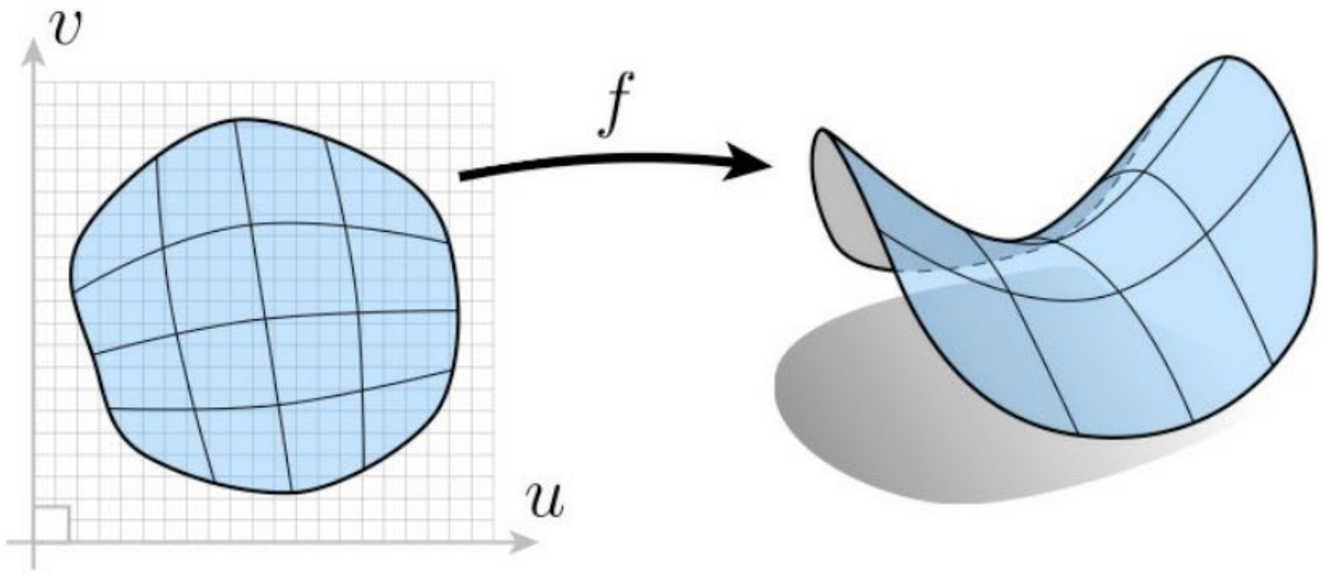
Outline

- Rasterized 3D representations
- Mesh
- Point cloud
- *Implicit representations*

Explicit Representation of Geometry

All points are given directly

Generally: $f : \mathbb{R}^2 \to \mathbb{R}^3; (u, v) \mapsto (x, y, z)$



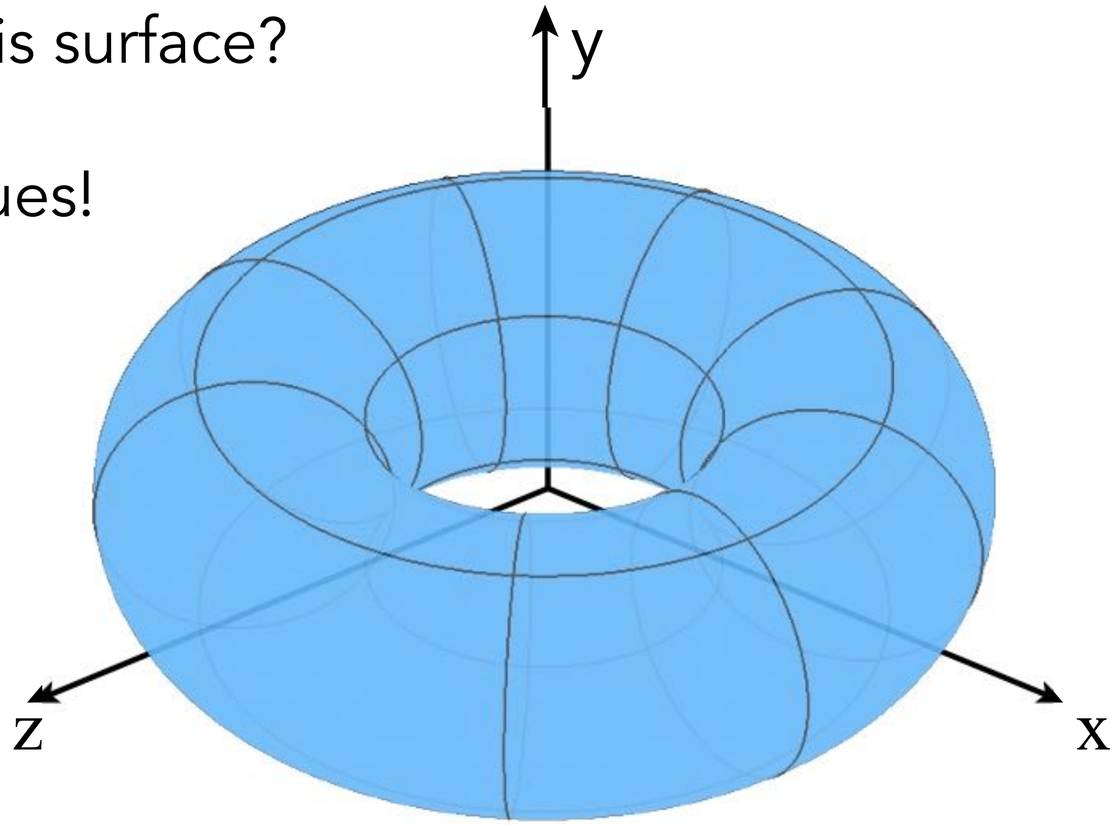
Slides credit: Ren Ng

Explicit Surface – Sampling Is Easy

$$f(u, v) = ((2 + \cos u)\cos v, (2 + \cos u)\sin v, \sin u)$$

What points lie on this surface?

Just plug in (u, v) values!

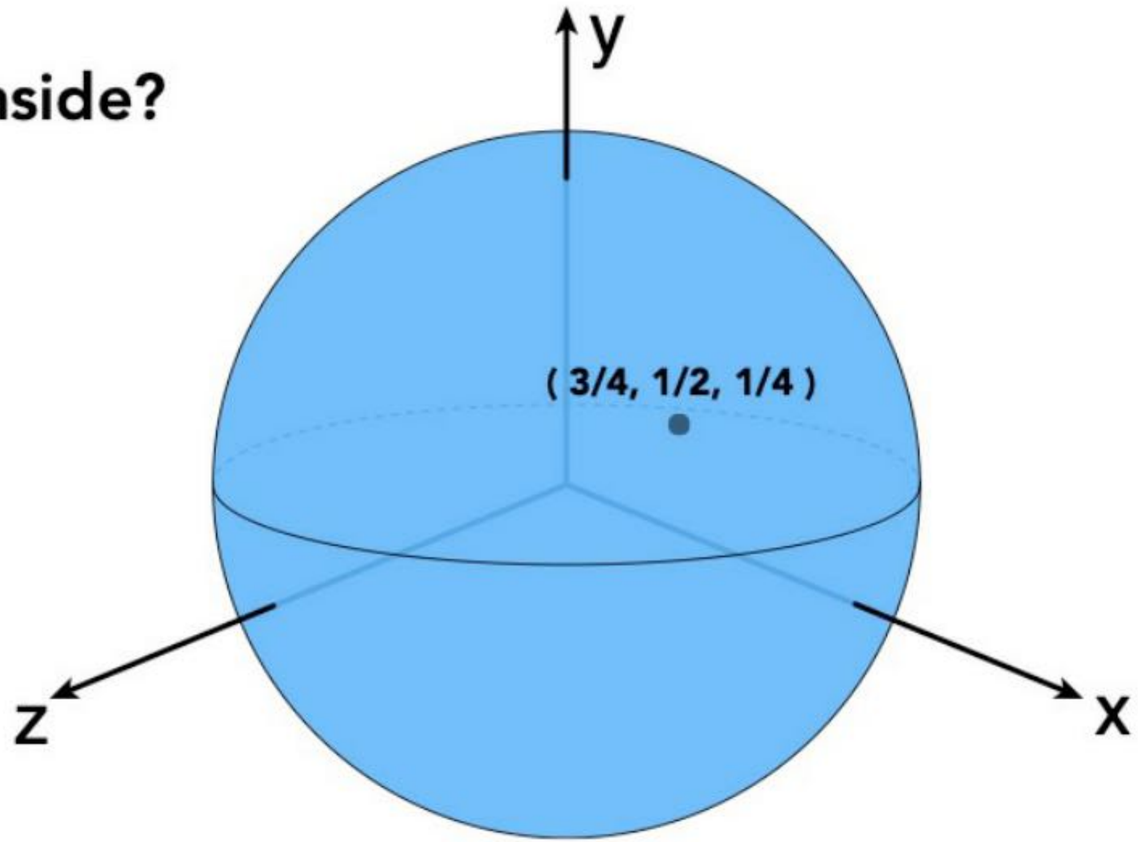


Explicit representations make some tasks easy

Explicit Surface - Inside/Outside Test Hard

$$f(u, v) = (\cos u \sin v, \sin u \sin v, \cos v)$$

Is (3/4, 1/2, 1/4) inside?



Some tasks are hard with explicit representations

Slides credit: Ren Ng

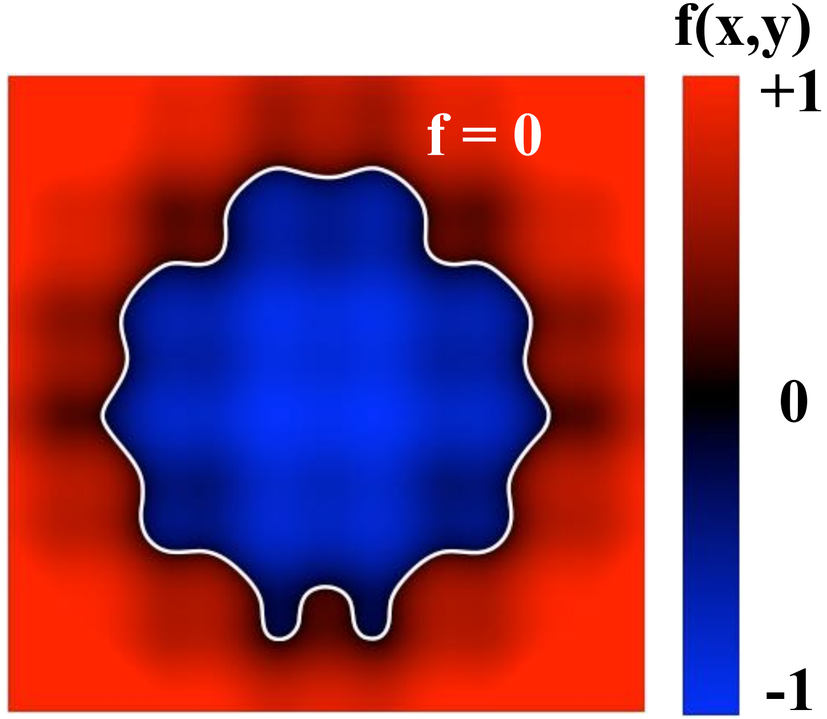
Implicit Representations of Geometry

Based on classifying points

- Points satisfy some specified relationship

E.g. sphere: all points in 3D, where $x^2 + y^2 + z^2 = 1$

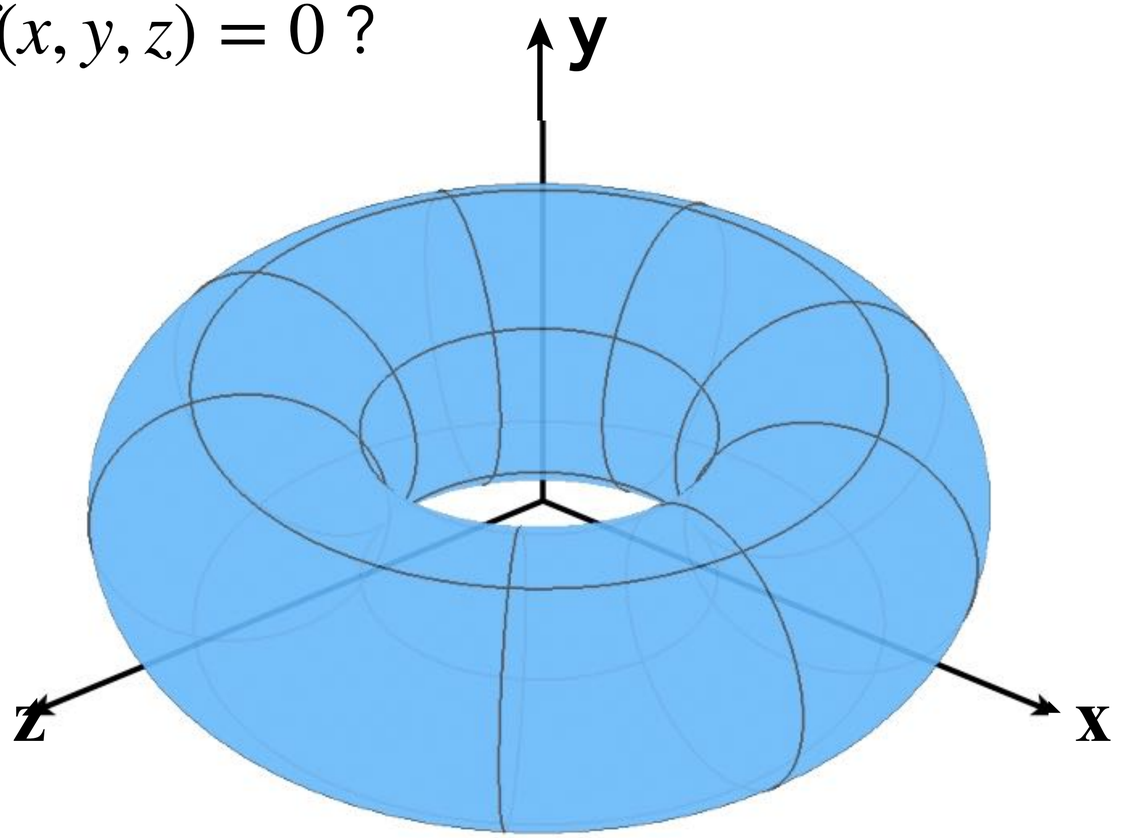
More generally, $f(x, y, z) = 0$



Implicit Surface – Sampling Can Be Hard

$$f(x, y, z) = (2 - \sqrt{x^2 + y^2})^2 + z^2 - 1$$

What points lie on $f(x, y, z) = 0$?



Some tasks are hard with implicit representations

Implicit Surface - Inside/Outside Tests Easy

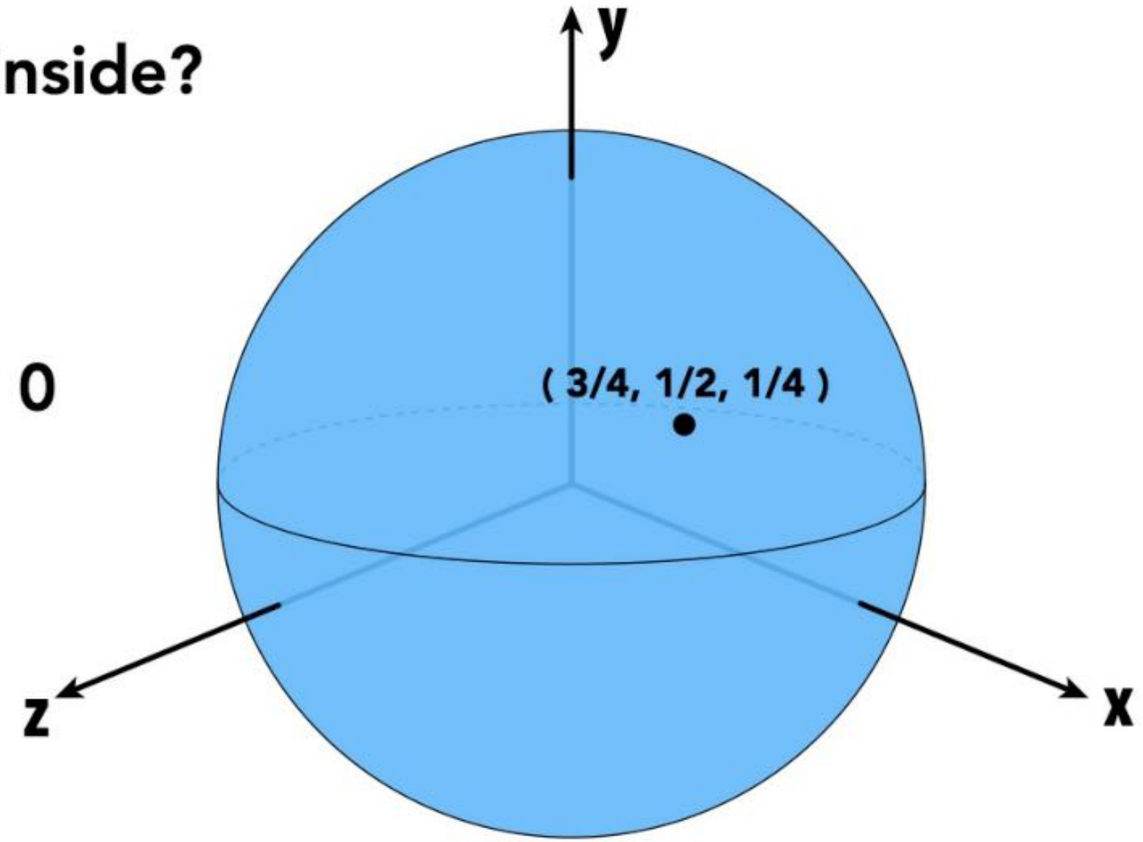
$$f(x, y, z) = x^2 + y^2 + z^2 - 1$$

Is (3/4, 1/2, 1/4) inside?

Just plug it in:

f(x,y,z) = –1/8 < 0

Yes, inside.



Implicit representations make some tasks easy

Slides credit: Ren Ng

Algebraic Surfaces (Implicit)

Surface is zero set of a polynomial in x, y, z

$$x^2+y^2+z^2=1$$

$$(R-\sqrt{x^2+y^2})^2+z^2=r^2$$

$$(x^2+\frac{9y^2}{4}+z^2-1)^3=$$

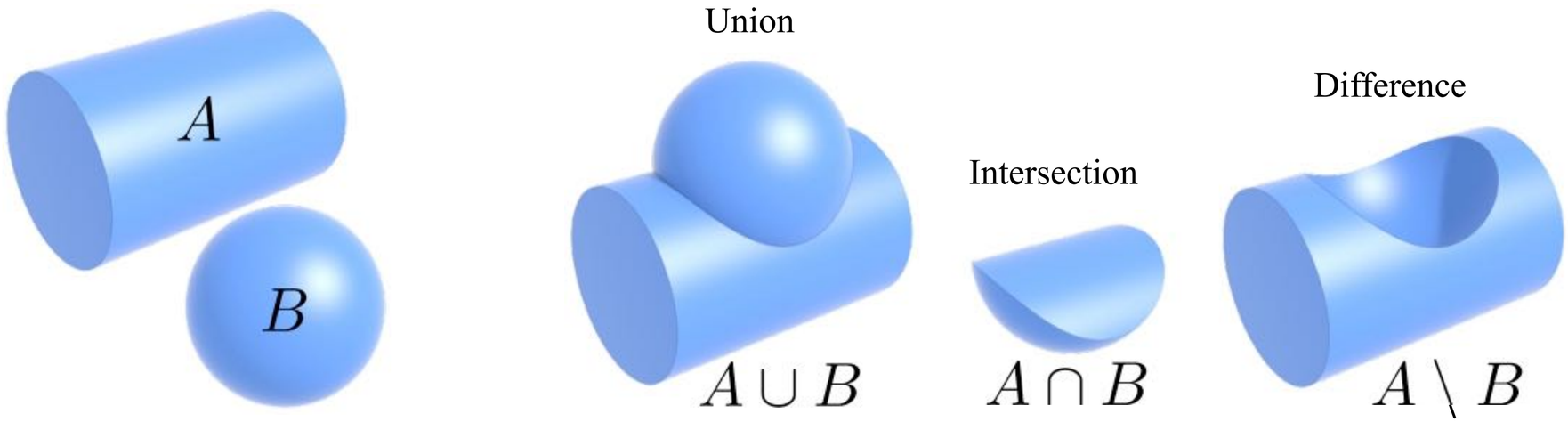
$$x^2z^3+\frac{9y^2z^3}{80}$$

More complex shapes?

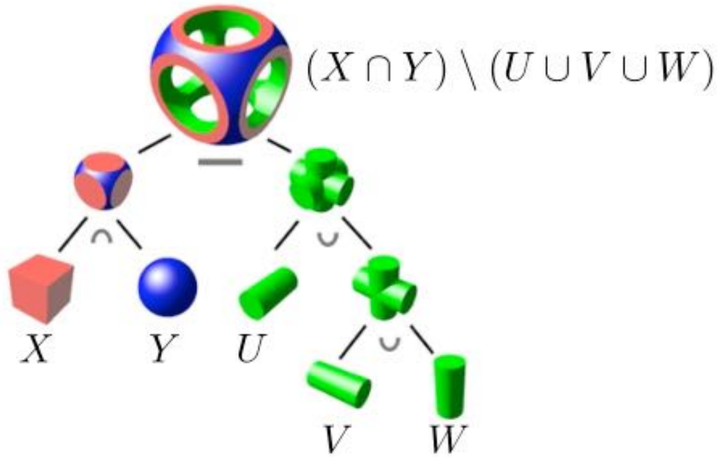
Slides credit: Jiajun Wu

Constructive Solid Geometry (Implicit)

Combine implicit geometry via Boolean operations



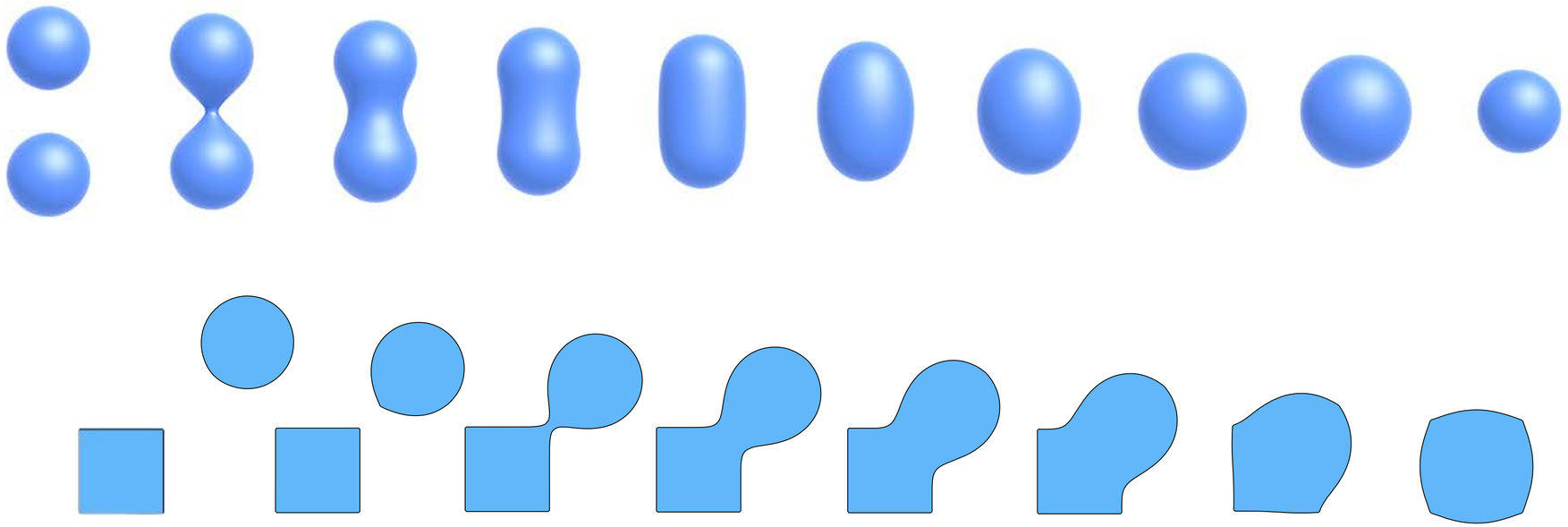
Boolean expressions :



Distance Functions

- Distance functions: giving minimum distance (could be signed distance) from anywhere to object

- Instead of booleans, gradually blend surfaces together using distance functions



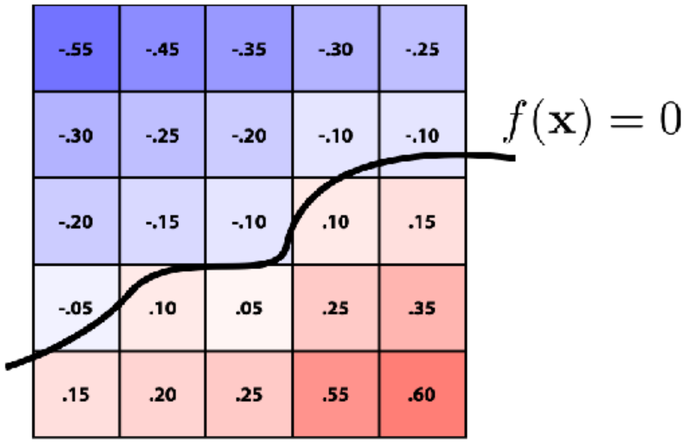
Scene of Pure Distance Functions (Not Easy!)

Slides credit: Jiajun Wu

Level Set Methods

Closed-form equations are hard to describe complex shapes

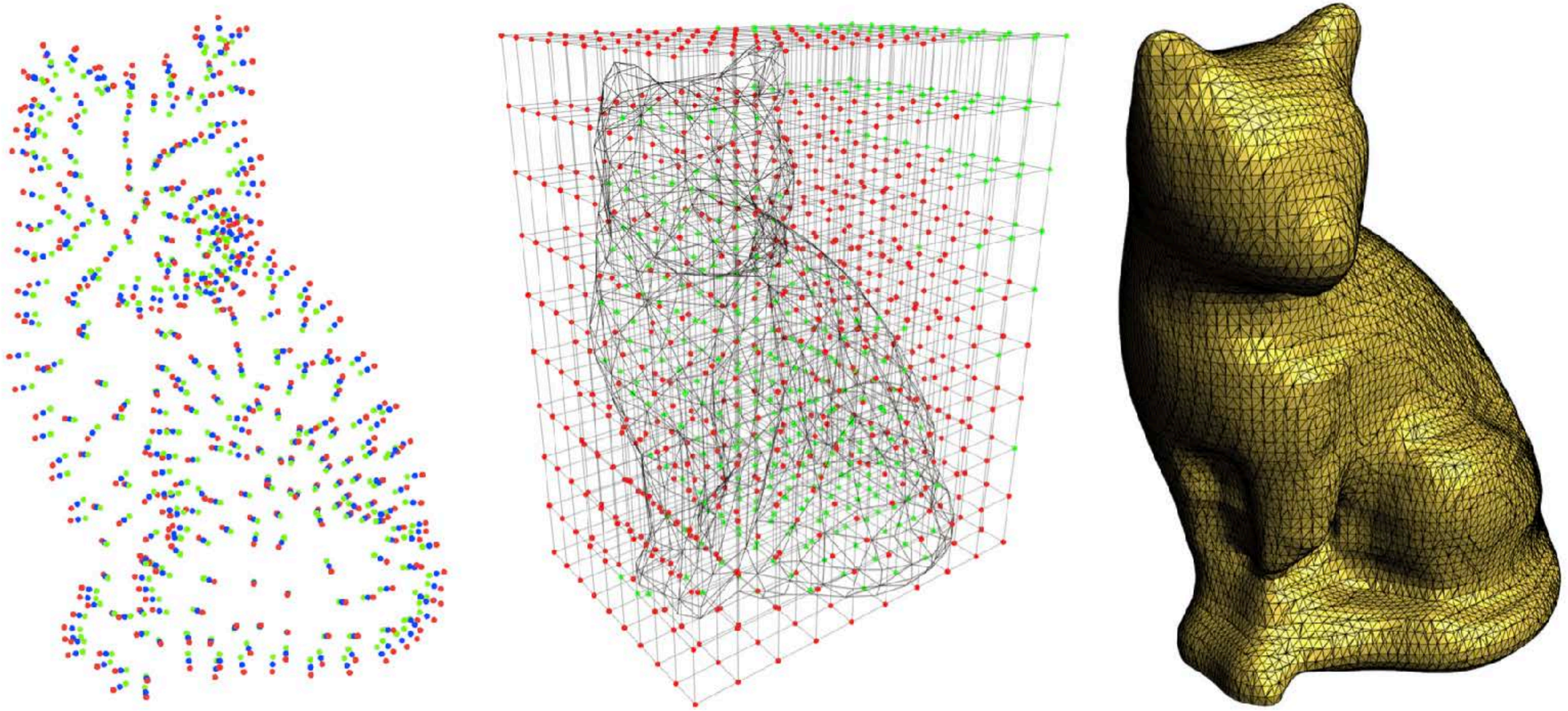
Alternative: store a grid of values approximating function



Surface is found where interpolated values equal zero

Provides much more explicit control over shape (like a texture)

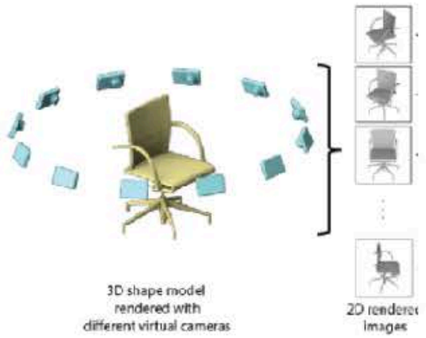
Conversion between different representations



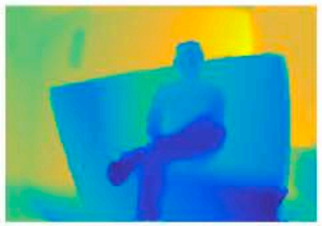
Slides credit: Hao Su

Summary

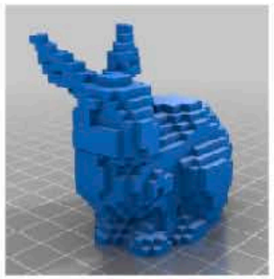
Rasterized form (regular grids)



Multi-view



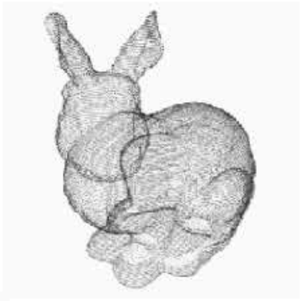
Depth Map



Volumetric

Geometric form (irregular)

Mesh



Point Cloud

$$F(x) = 0$$

Implicit Shape

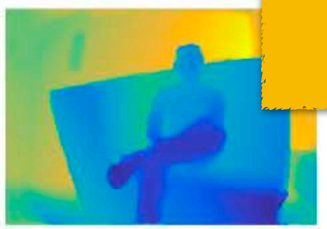
Summary

Rasterized form (regular grids)

Geometric form (irregular)

No "best" geometric representation

Choose your representation depending on the task!



Depth Map

Volumetric

Point Cloud

$$f(x) = 0$$

Implicit Shape

Next time: 3D transformations