Embodied Vision

Structure from Motion and 3D Representations

Tsung-Wei Ke

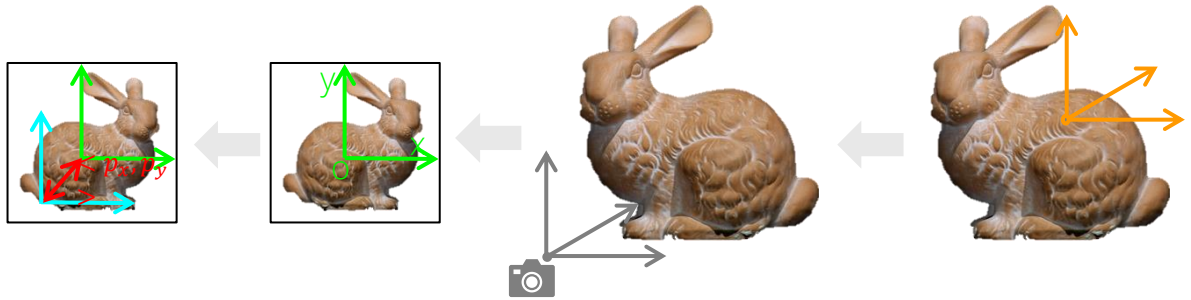
Spring 2026

Disclaimer

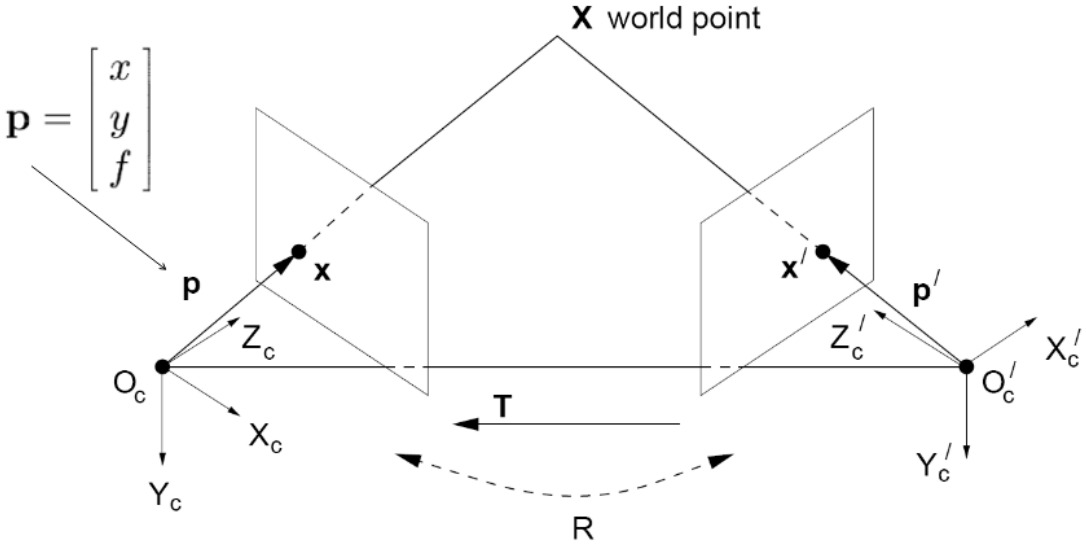
- This lecture borrows contents heavily from
  - Learning for 3D Vision by Shubam Tulsiani at CMU
  - Neural Representations and Generative Models for 3D Geometry by L. Guibas at Stanford
  - Computer Vision by Noah Snavely at Cornell
  - Intro to Computer Vision and Computational Photography by Alyosha Efros at UC Berkeley

Recap

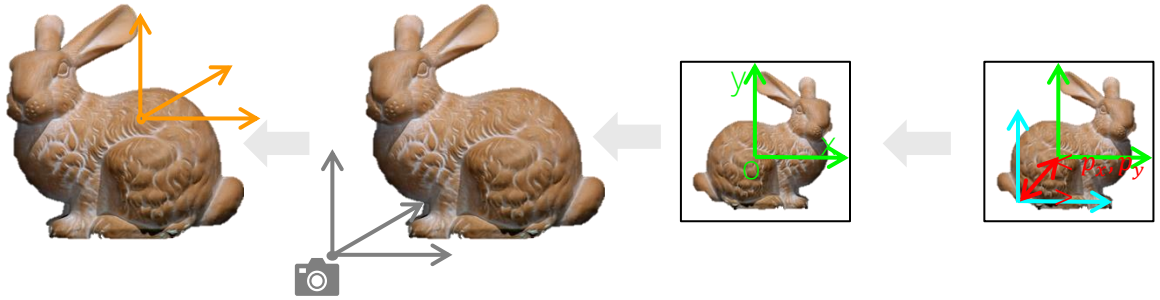
3D-to-2D perspective projection



$$\begin{bmatrix} x_{pix} \\ y_{pix} \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{R}_{\boldsymbol{w}}^{\boldsymbol{c}} & \boldsymbol{T}_{\boldsymbol{w}}^{\boldsymbol{c}} \\ \mathbf{0} & 1 \end{bmatrix} \quad \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

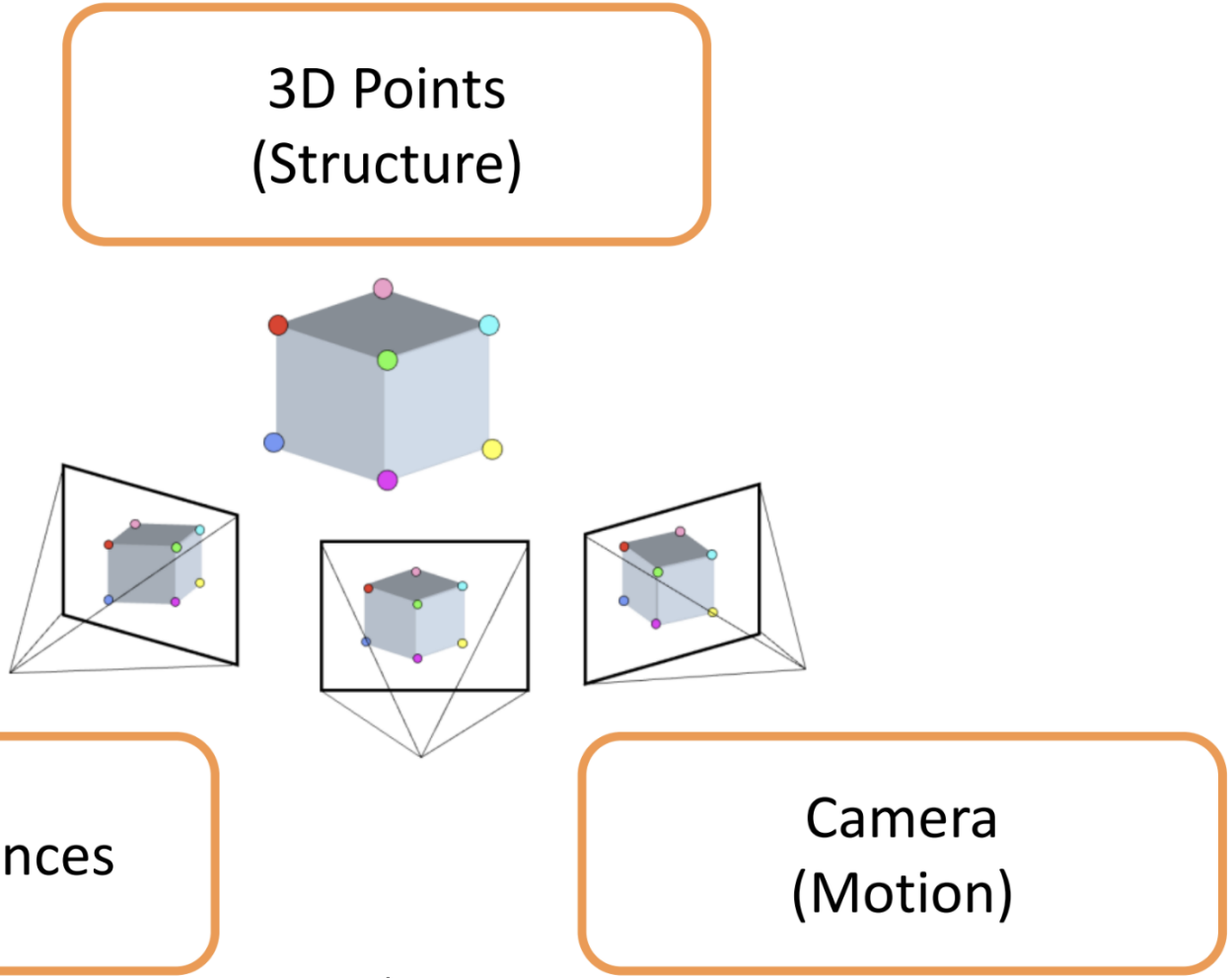


2D-to-3D reconstruction



$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\boldsymbol{w}}^{\boldsymbol{c}} & \boldsymbol{T}_{\boldsymbol{w}}^{\boldsymbol{c}} \\ \mathbf{0} & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_{pix} \\ y_{pix} \\ 1 \end{bmatrix}$$

To Reconstruct 3D Models, We Need to Know Correspondence and Camera Poses



How to Model the 3D World from Photo Collections?

Building Rome in a Day. Agarwal et al.

Estimate Camera Pose, Correspondence and 3D Models from Image Collection

From photo collections, we want to obtain:

- Camera pose: Where were the photo taken?
- 3D shape
- Pixel correspondence

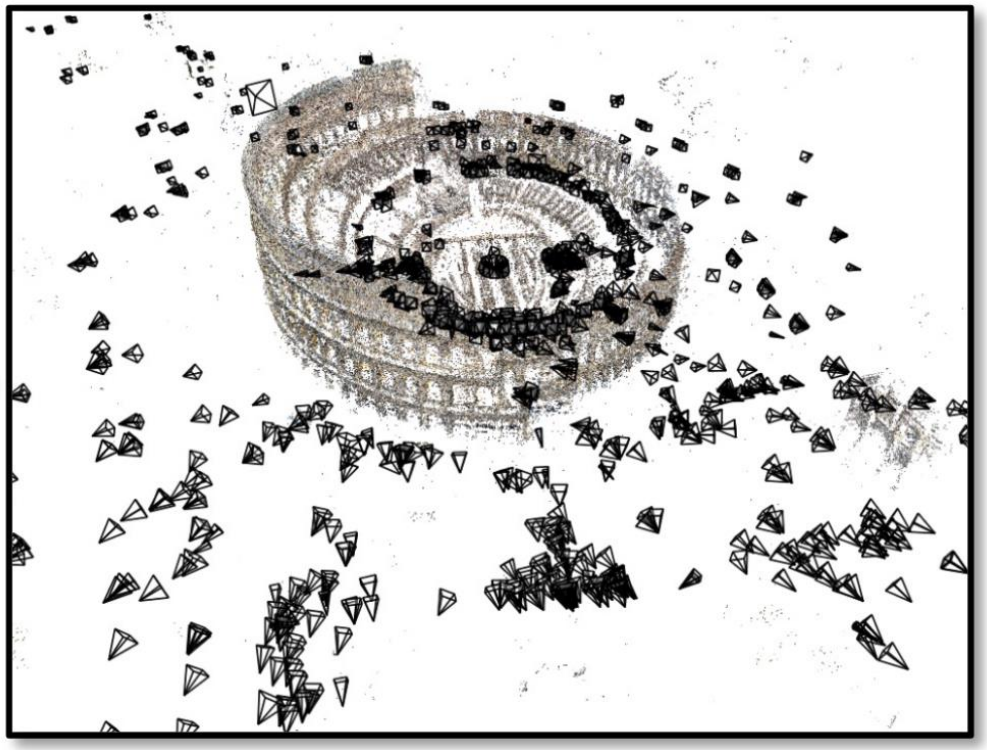
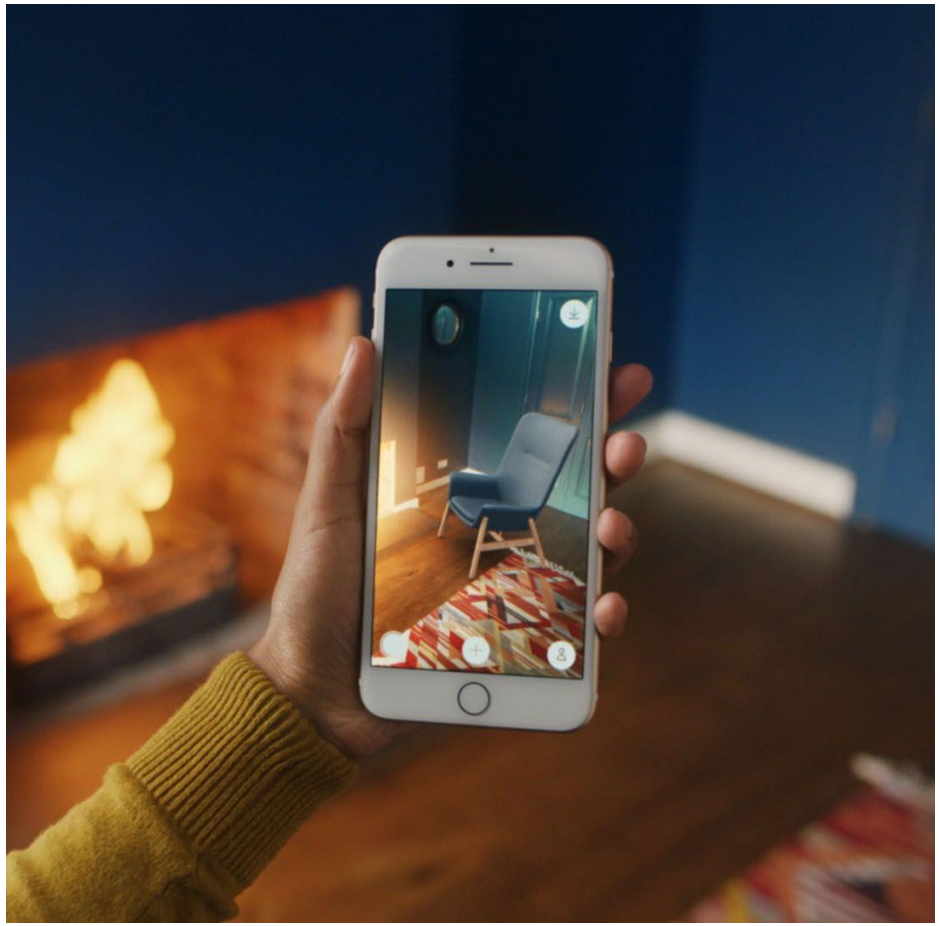


Image credit N. Snavely

How to Represent the 3D Models



https://www.ikea.com/global/en/newsroom/innovation/ikea-launches-ikea-place-a-new-app-that-allows-people-to-virtually-place-furniture-in-their-home-170912/

Image credit Davide Scaramuzza

Content

- Geometry Reconstruction
  - Structure from Motion
  - 3D Foundation Model
- 3D Representations
  - Depth Map
  - Point Cloud
  - Mesh
  - Parametric Surface
  - Signed Distance Function
  - Voxels
  - Others
- Conversion across Representations

Content

- Geometry Reconstruction
  - ➢Structure from Motion
  - ➢3D Foundation Model
- 3D Representations
  - ➢Depth Map
  - ➢Point Cloud
  - ➢Mesh
  - ➢Parametric Surface
  - ➢Signed Distance Function
  - ➢Voxels
  - ➢Others
- Conversion across Representations

Structure from Motion

From photo collections, we want to obtain:

- Camera pose: Where were the photo taken?
- 3D shape
- Pixel correspondence

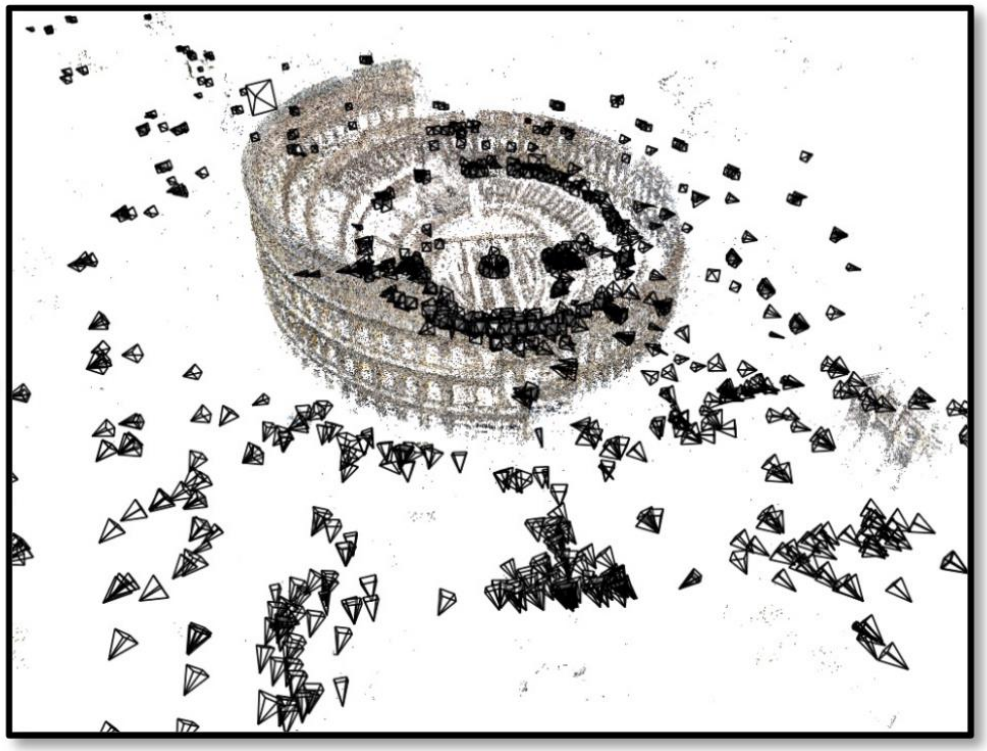
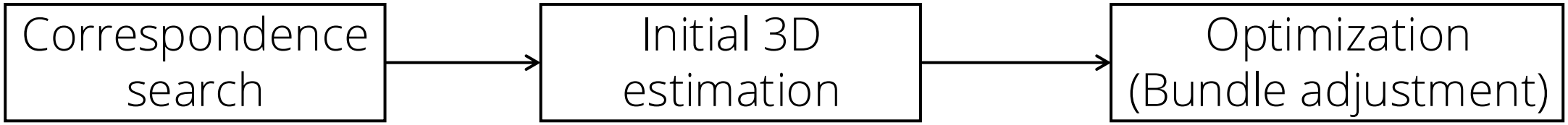


Image credit N. Snavely

Solve Simpler Problems First

Problem: 3D reconstruction without correspondence and camera motions



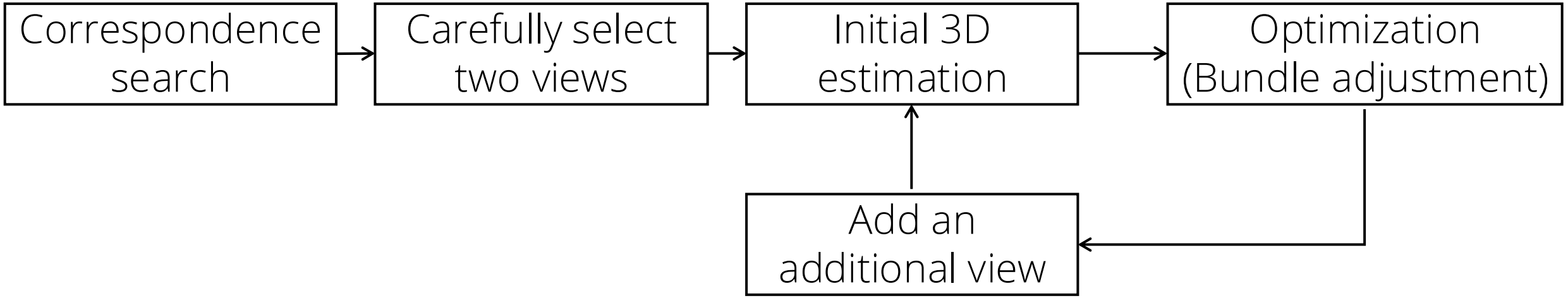
- Find the pair of overlapping
- Identify pixel correspondence

- Estimate the camera pose
- Reconstruct the 3D model with triangulation

- Refine the estimation of camera pose
- Refine the reconstruction of the 3D model

Solve Simpler Problems First

Problem: 3D reconstruction from a collection of images



Solutions:

- Initialize from the easiest/most-confident image pair
- Incrementally add hard/less-confident images

Correspondence Search: Feature Detection

Detect features using SIFT [Lowe, IJCV 2004]

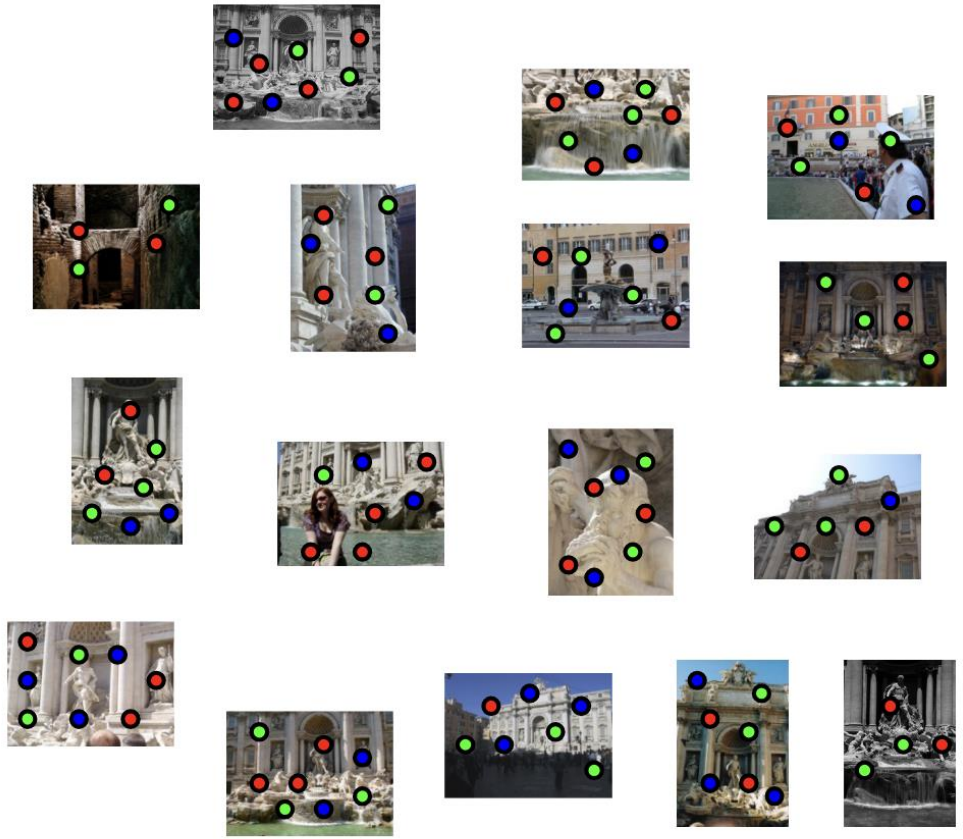
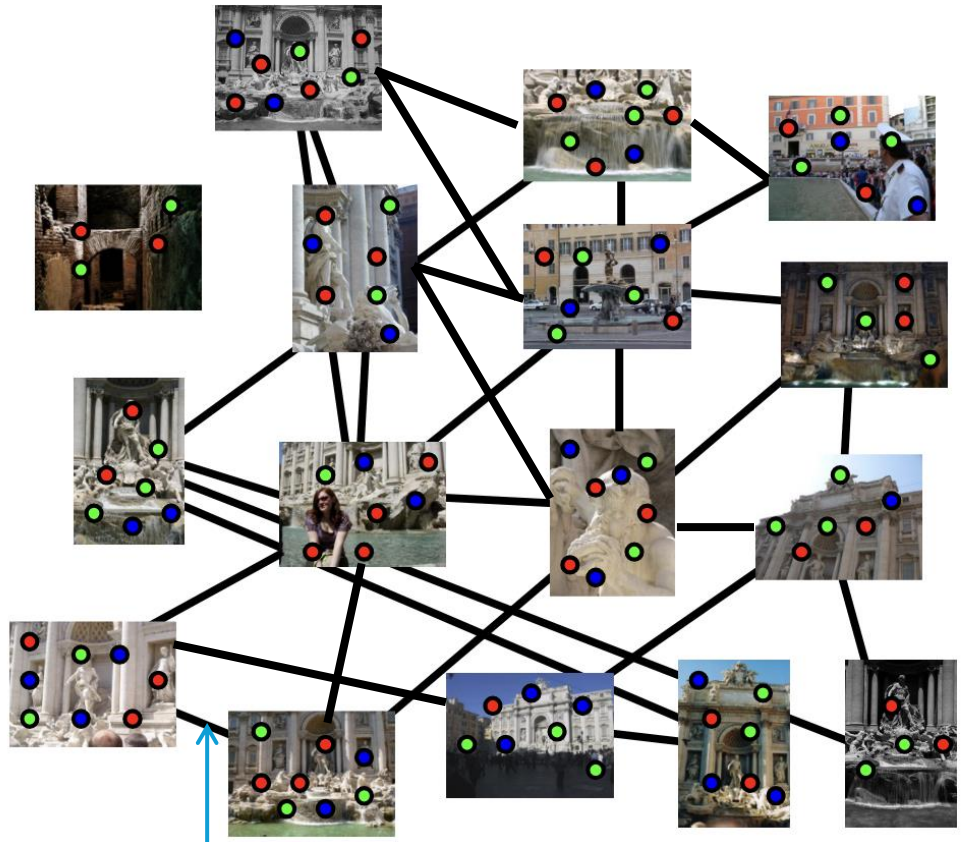


Image credit N. Snavely

Correspondence Search: Feature Detection

Match features between each pair of images

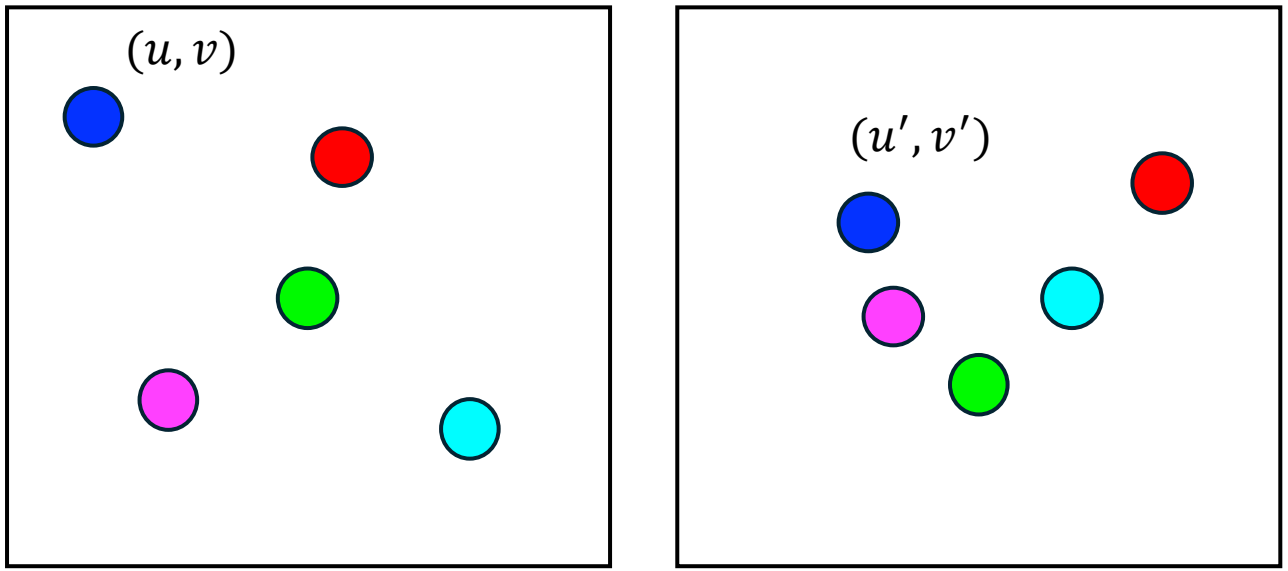


Connect a pair of image if they are overlapped (contain a certain number of corresponded pixels)

Image credit N. Snavely

Correspondence Search: Geometric Verification

Estimate fundamental matrix between each pair and re-evaluate the pixel correspondence



$$\begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0$$

Fundamental matrix

$\boldsymbol{F}$

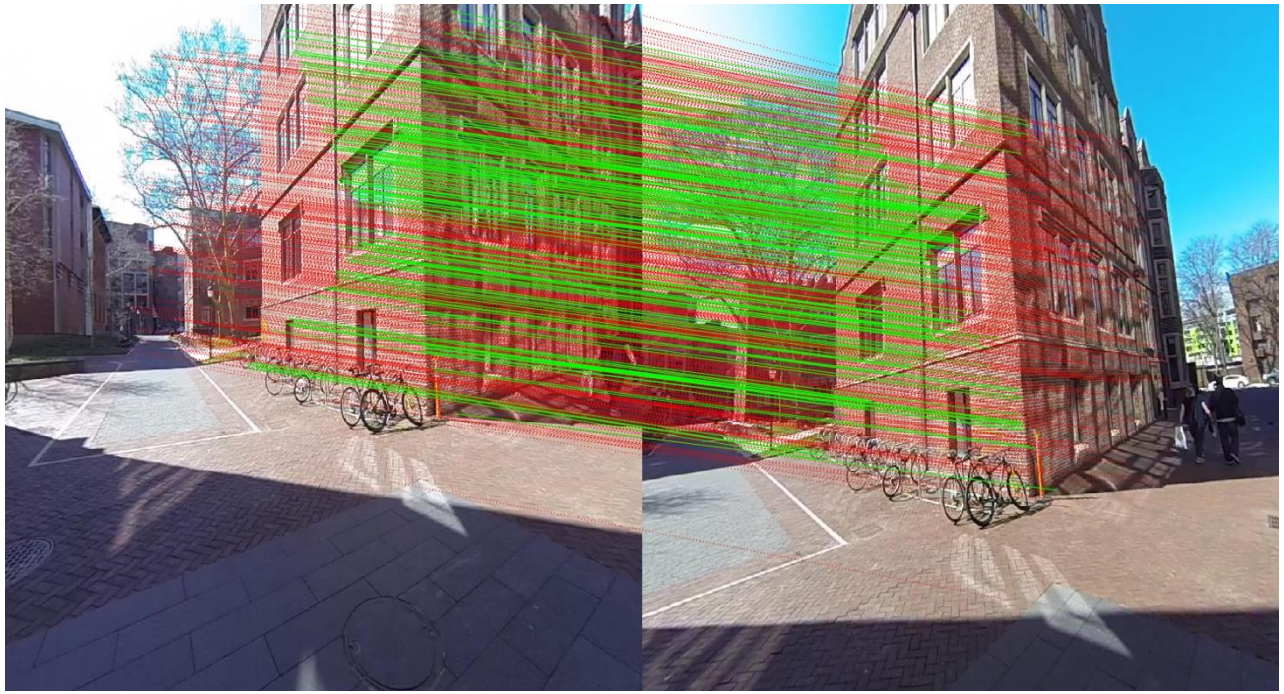
Initial Pixel Correspondence May be too Noisy… Remove Outliers with RANSAC

Algorithm 1 RANSAC

1: Select randomly the minimum number of points required to determine the model parameters.
2: Solve for the parameters of the model.
3: Determine how many points from the set of all points fit with a predefined tolerance ϵ.
4: If the fraction of the number of inliers over the total number points in the set exceeds a predefined threshold τ, re-estimate the model parameters using all the identified inliers and terminate.
5: Otherwise, repeat steps 1 through 4 (maximum of N times).

Initial Pixel Correspondence May be too Noisy… Remove Outliers with RANSAC

n=0;
for $i = 1{:}M$ **do**
  // Choose 8 correspondences, $\hat{x}_1$ and $\hat{x}_2$ randomly
  F = EstimateFundmentalMatrix($\hat{x}_1$, $\hat{x}_2$);
  $\mathcal{S} = \emptyset$;
  for $j = 1{:}N$ **do**
    if $| x_{2j}^T F x_{1j} | < \epsilon$ **then**
      $\mathcal{S} = \mathcal{S} \cup \{j\}$
    end
  end
  if $n < | \mathcal{S} |$ **then**
    $n = | \mathcal{S} |$;
    $\mathcal{S}_{in} = \mathcal{S}$
  end
end



Correspondence Search: Geometric Verification

Estimate fundamental matrix between each pair and re-evaluate the pixel correspondence

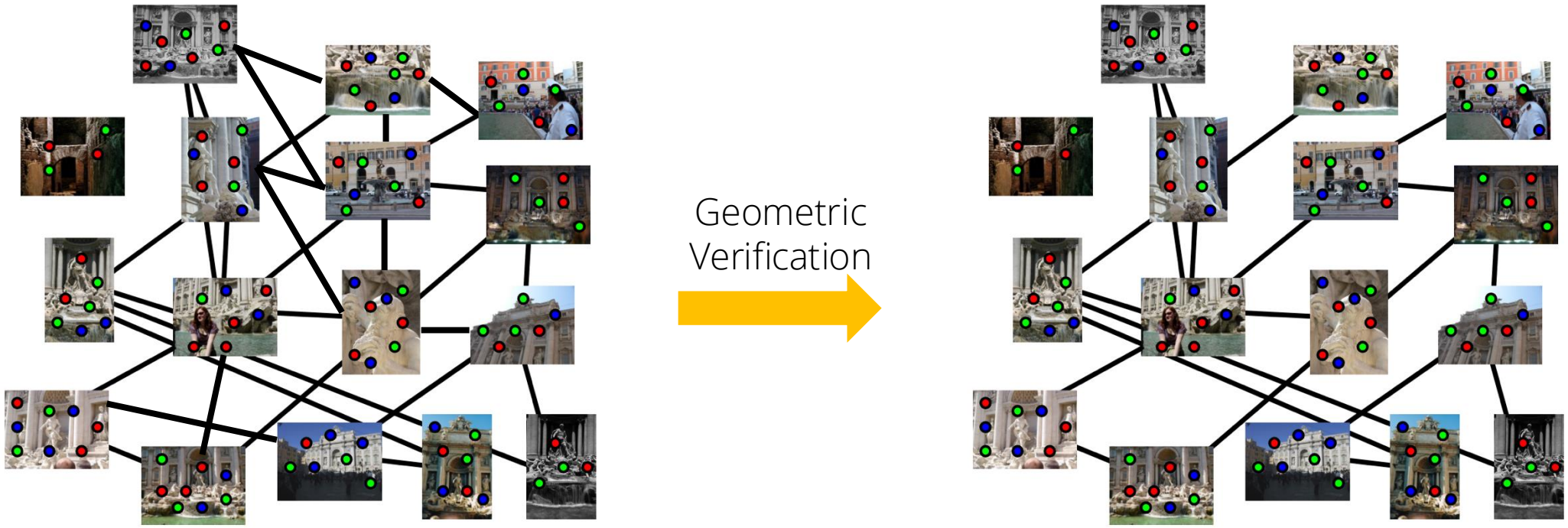
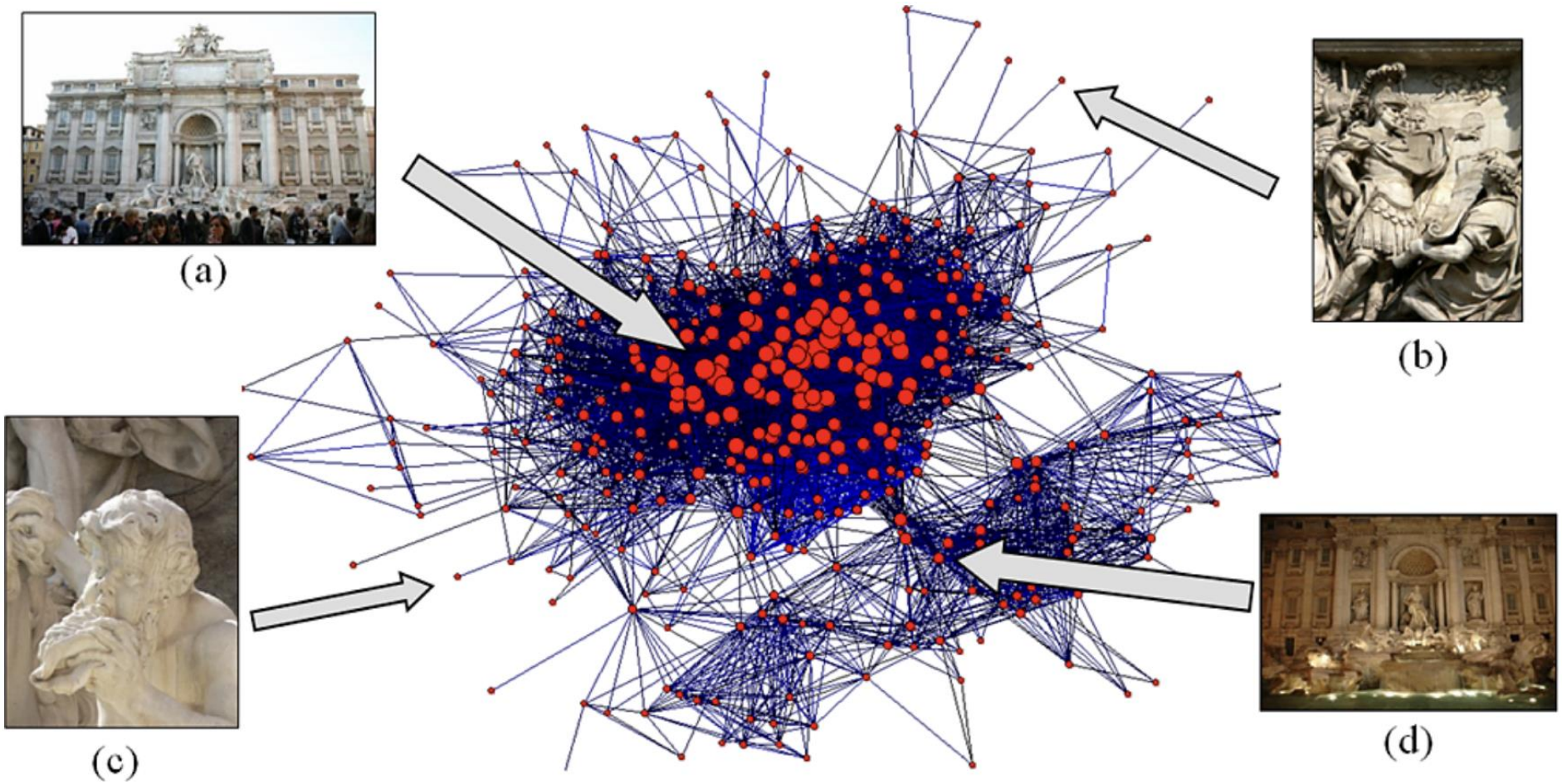


Image credit N. Snavely

Image Connectivity Graph



Modeling the World from Internet Photo Collections. Snavely et al. IJCV 2008.

Solve Simpler Problems First

Correspondence search → **Initial 3D estimation** → Optimization (Bundle adjustment)

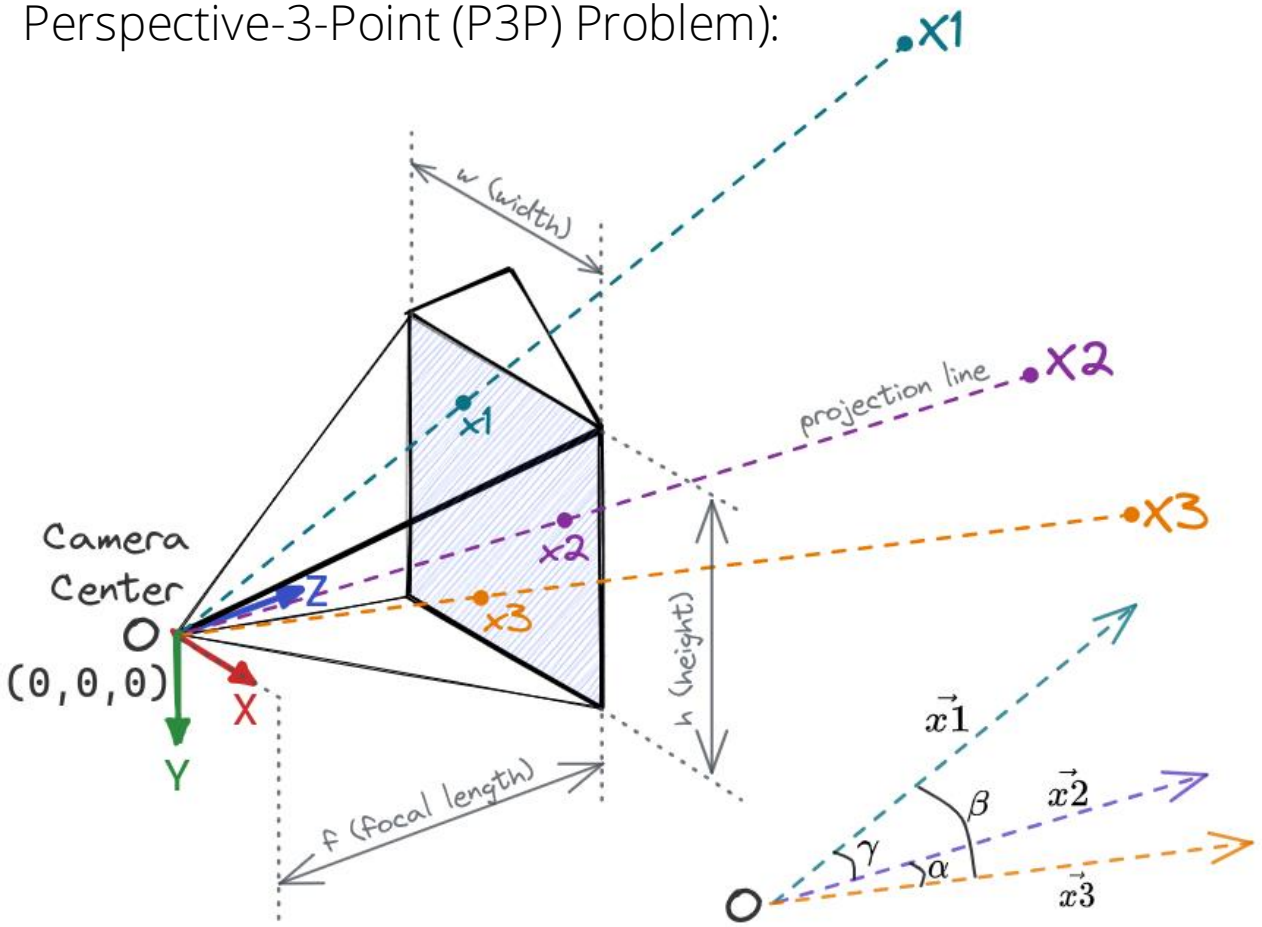
- Find the pair of overlapping
- Identify pixel correspondence

- Estimate the camera pose
- Reconstruct the 3D model with triangulation

- Refine the estimation of camera pose
- Refine the reconstruction of the 3D model

Initial 3D Estimation: Pose Estimation with Perspective-n-Point (PnP) Solver

Perspective-3-Point (P3P) Problem):



PnP solver derives the camera pose from n-point correspondence.

- The solution derived by SVD is sub-optimal
- The solution derived by Fundamental matrix estimation has ambiguity in “scale”

For more details, check:

- “Multiple view geometry in computer vision” by Richard Hartley and Andrew Zisserman
- Lecture by C. Stachniss (https://www.ipb.uni-bonn.de/html/teaching/photo12-2021/2021-pho1-23-p3p.pptx.pdf)
- Blog post by Jingnan Shi (https://jingnanshi.com/blog/pnp_minimal.html)
- Lecture by S. LaValle (https://www.youtube.com/watch?v=0JGC5hZYCVE)

https://medium.com/@rashik.shrestha/perspective-n-point-pnp-f2c7dd4ef1ed

Solve Hard Problems Last

Correspondence search → Initial 3D estimation → Optimization (Bundle adjustment)

- Find the pair of overlapping
- Identify pixel correspondence

- Estimate the camera pose
- Reconstruct the 3D model with triangulation

- Refine the estimation of camera pose
- Refine the reconstruction of the 3D model

Bundle Adjustment: Joint Estimation of 3D Model and Camera Pose

- Minimize sum of squared reprojection errors:

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \cdot \left\| \mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j) - \begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix} \right\|^2$$

w_{ij}: *indicator variable*: is point i visible in image j ?

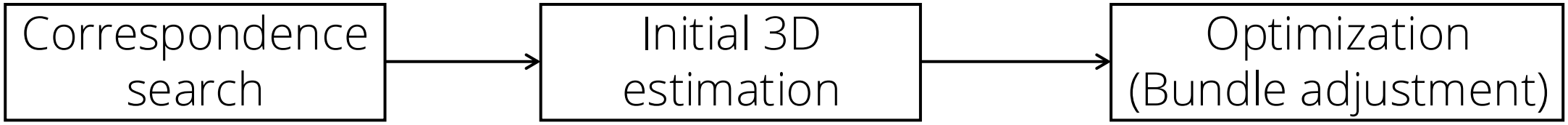
$\mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j)$: *predicted* image location

$\begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix}$: *observed* image location

- Minimizing this function is called *bundle adjustment*
  - Optimized using non-linear least squares, e.g. Levenberg-Marquardt algorithm

Slide credit N. Snavely

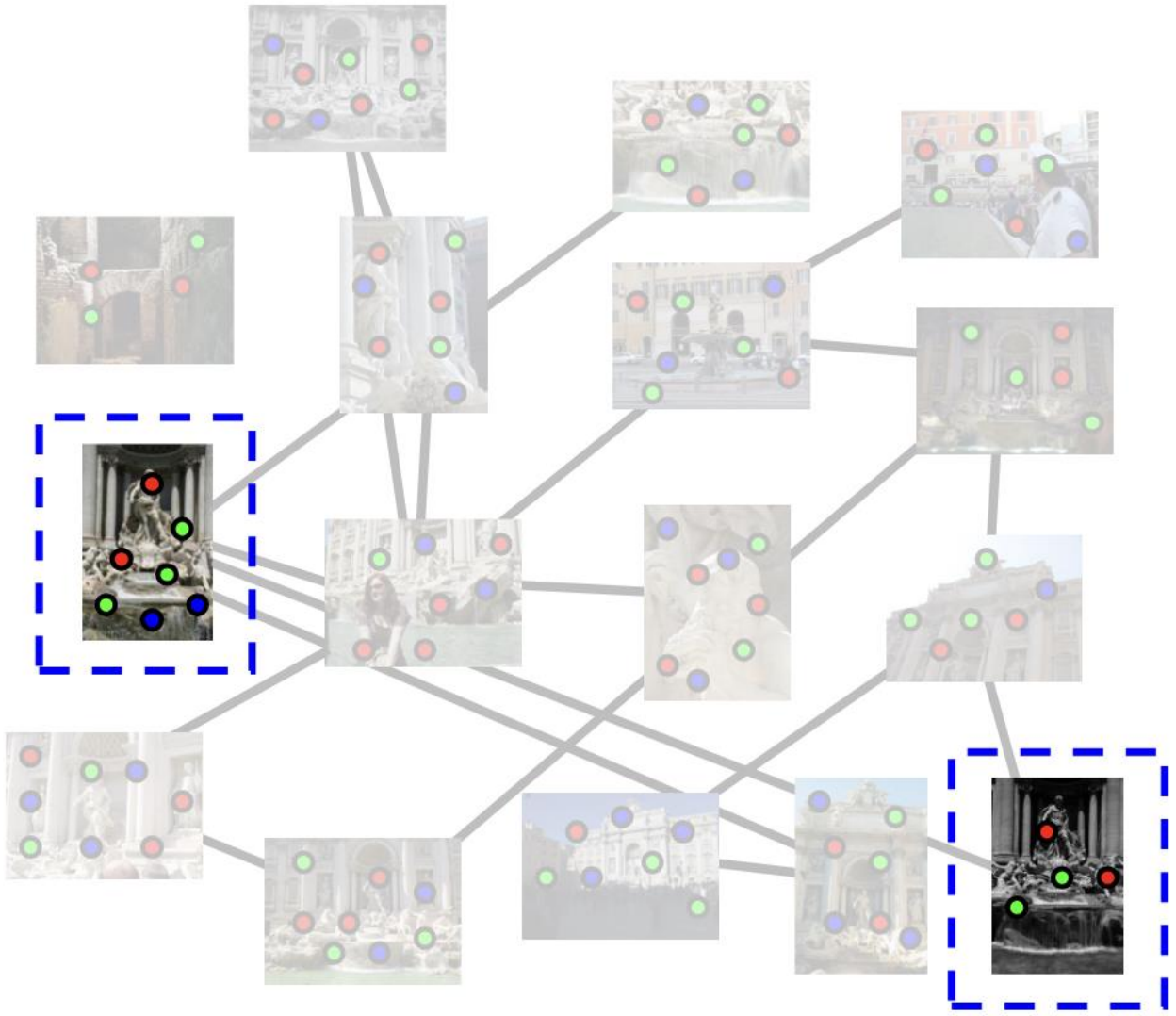
What Could be Wrong?



Problems:

- Difficult to initialize cameras all at once
- Bad initialization will lead to very-wrong results

Idea: Initialize from the easiest/most-confident image pair



More precise pose estimation and 3D reconstruction

Slide credit A. Efros

1. Picking the initial pair

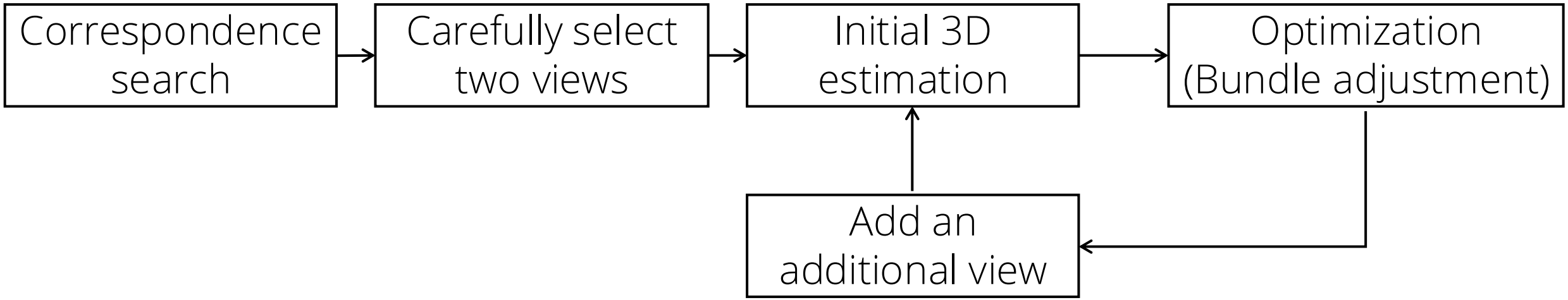
- We want a pair with many matches, but which has as large a baseline as possible


✔ lots of matches
✘ small baseline


✔ large baseline
✘ very few matches


✔ large baseline
✔ lots of matches

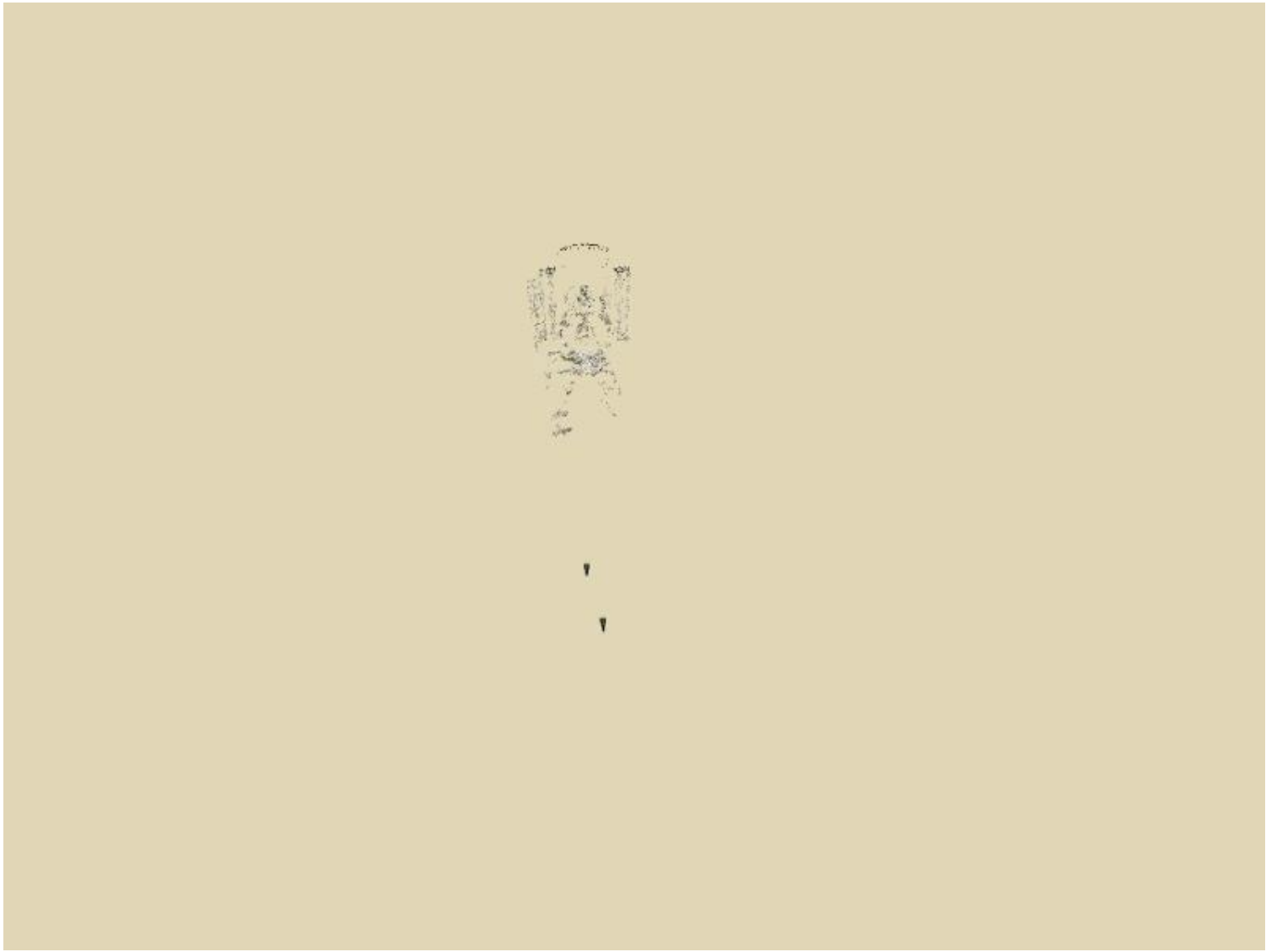
Incremental Structure from Motion



Solutions:

- Initialize from the easiest/most-confident image pair
- Incrementally add hard/less-confident images

Incremental Structure from Motion



Video credit N. Snavely

Incremental Structure from Motion

Video credit N. Snavely

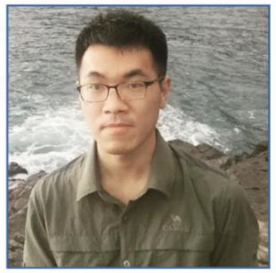
Content

- Geometry Reconstruction
  - ➢Structure from Motion
  - ➢3D Foundation Model
- 3D Representations
  - ➢Depth Map
  - ➢Point Cloud
  - ➢Mesh
  - ➢Parametric Surface
  - ➢Signed Distance Function
  - ➢Voxels
  - ➢Others
- Conversion across Representations

Do We Have Better Solution? Yes!

DUSt3R: Geometric 3D Vision Made Easy

CVPR 2024



Shuzhe Wang
Aalto University

Vincent Leroy
Naverlabs Europe

Yohann Cabon
Naverlabs Europe

Boris Chidlovskii
Naverlabs Europe

Jérome Revaud
Naverlabs Europe

Aalto University

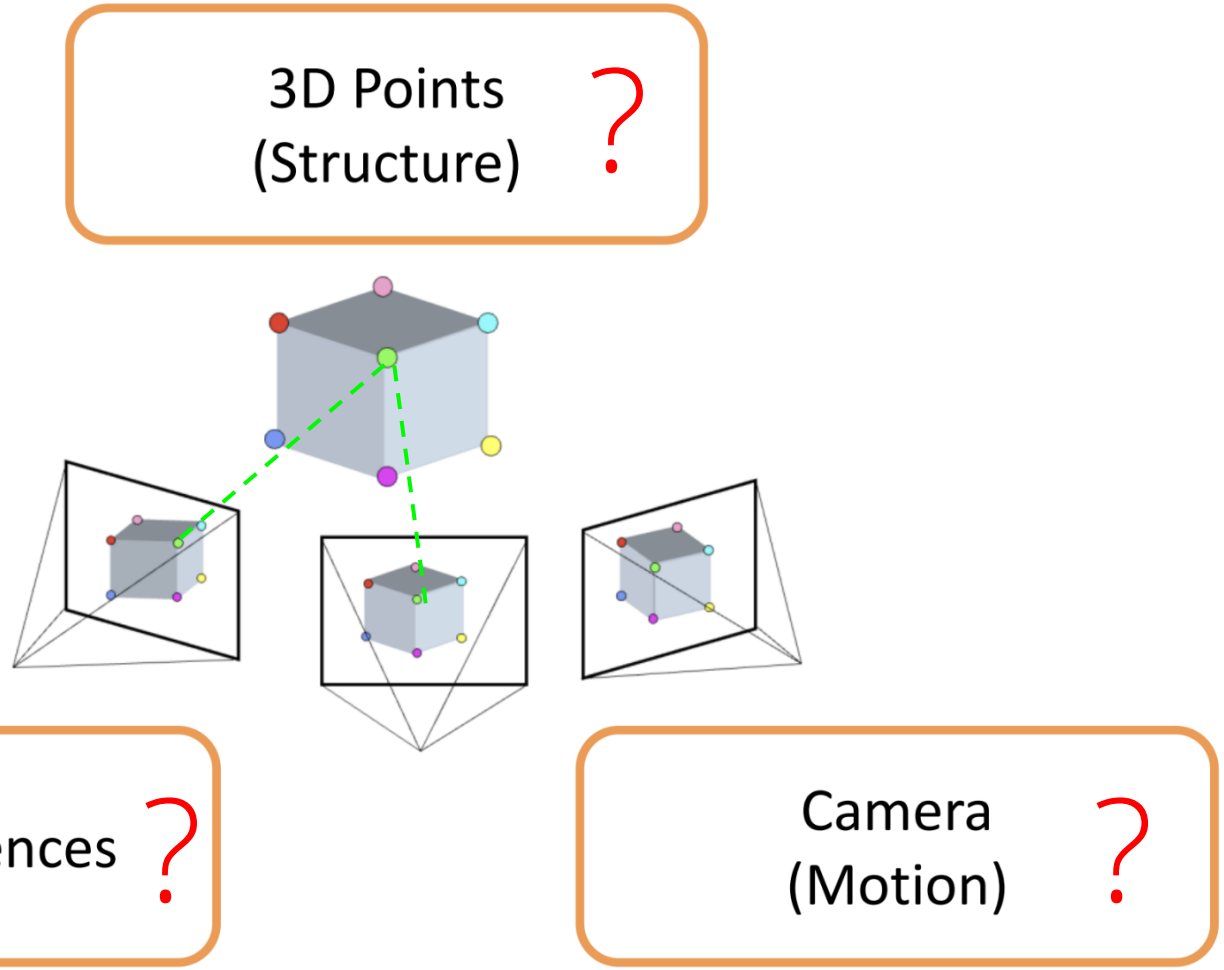
NAVER LABS Europe

Camera Pose, Correspondence, and 3D Shape are Unknown

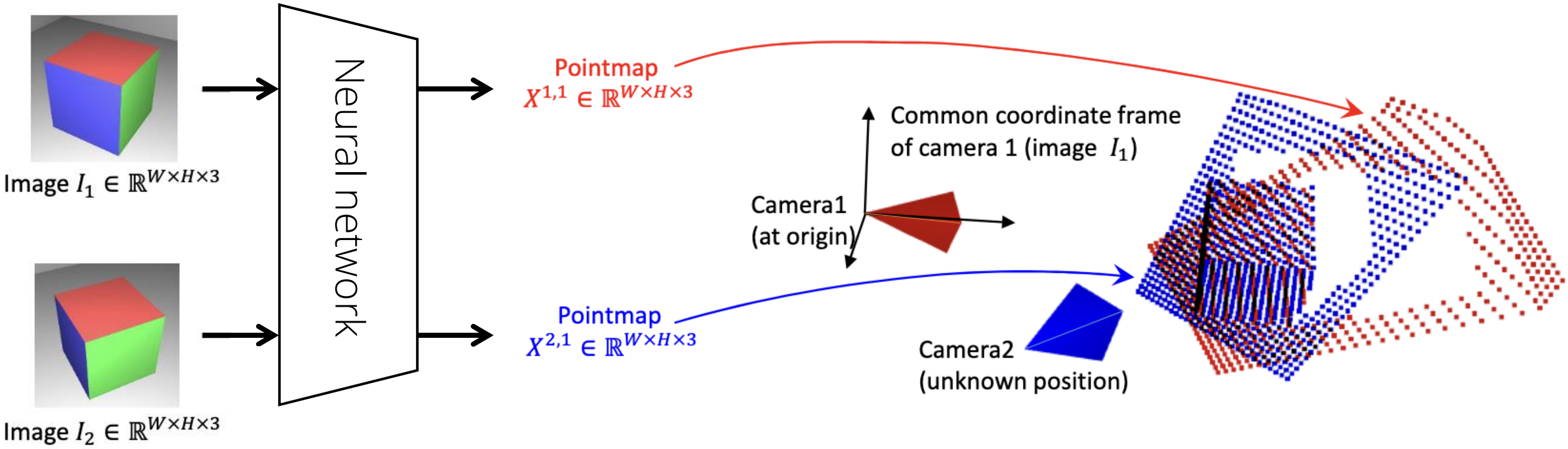
Solution: Learn to map each pixel to a 3D location among two views.

Correspondence emerges from the 3D location of every pixel!

Camera motion can be derived accordingly!

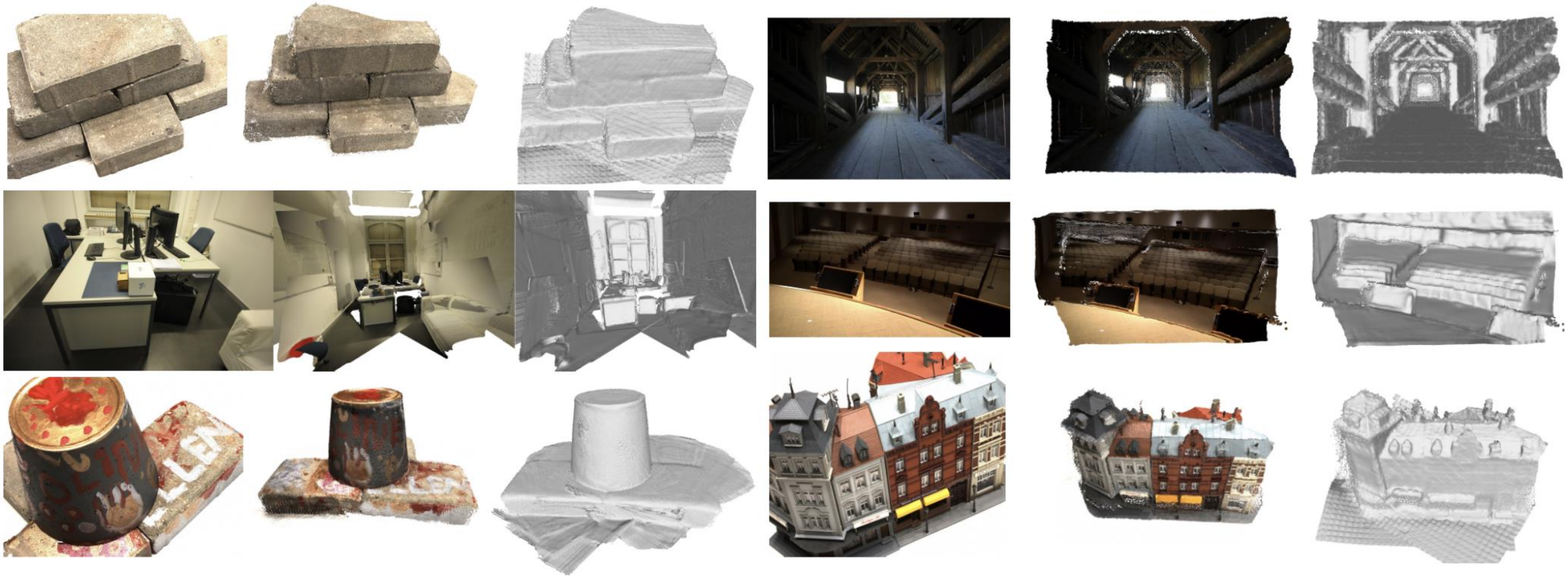


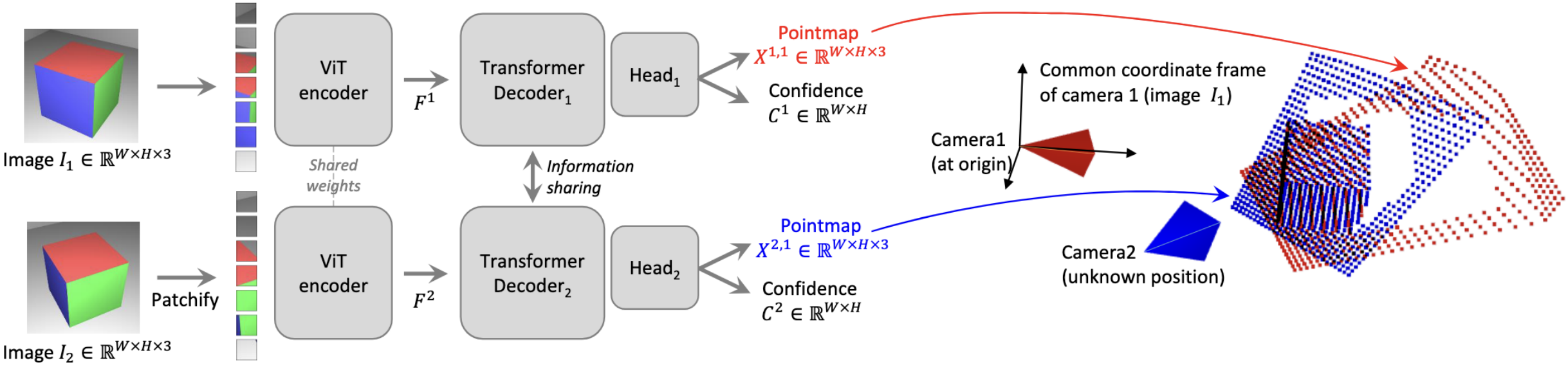
DUSt3R: Dense and Unconstrained Stereo 3D Reconstruction



Predict 3D Pointmap at the Coordinate Frame of View 1

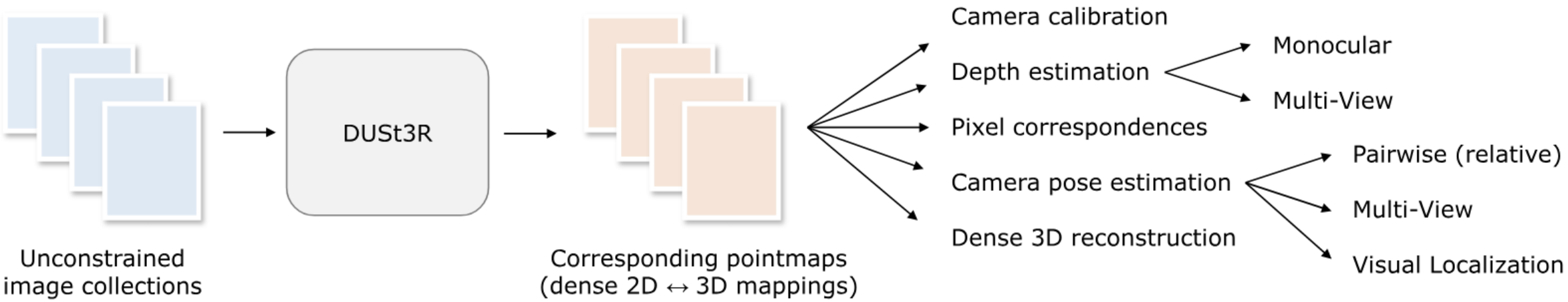
View 1 | View 2 | 3D model | View 1 | View 2 | 3D model

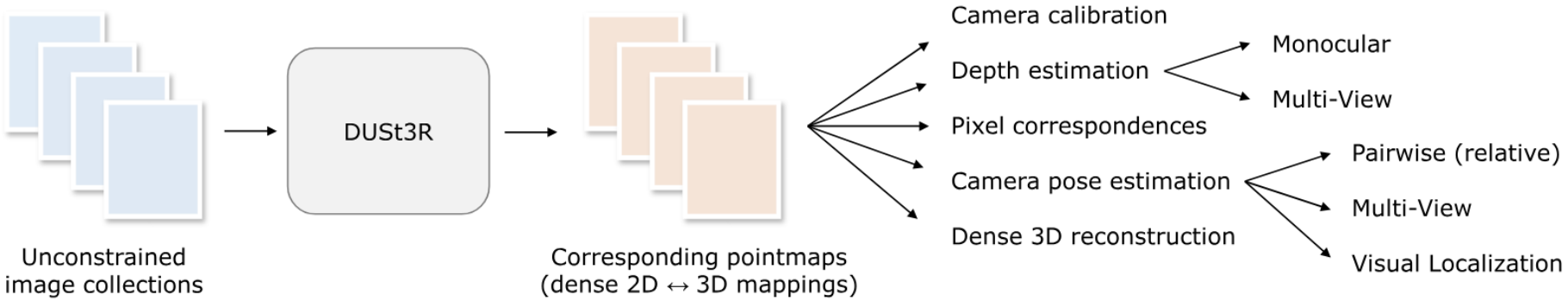




- Idea: Learn a neural network that predicts 3D pointmaps of two camera views
- Pixel correspondence emerges from the proximity of their 3D locations
- Camera motions can be derived accordingly
- Requirement:
  - ➢ A lots of 3D datasets with 3D pointmaps for image pairs
  - ➢ A strong visual encoder pre-trained with cross-view completion

A Model that Solves Structure from Motion

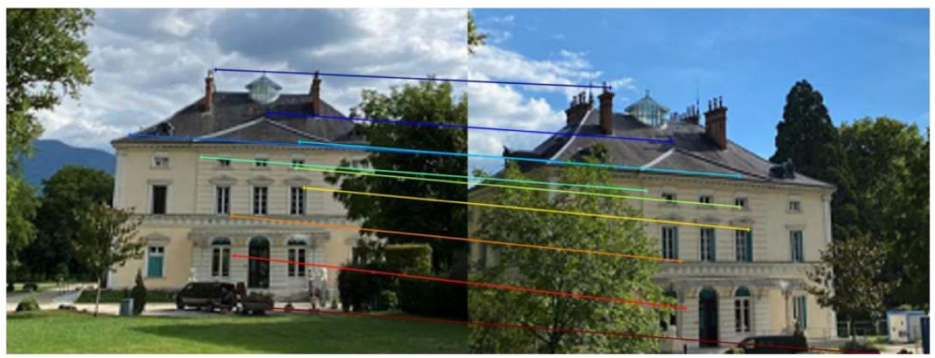




1. Point Matching

Achieved by mutual nearest neighbor (MNN) search in the 3D pointmap space.

$$\mathcal{M}_{1,2} = \{(i,j) \mid i = \mathrm{NN}_1^{1,2}(j) \text{ and } j = \mathrm{NN}_1^{2,1}(i)\}$$
$$\text{with } \mathrm{NN}_k^{n,m}(i) = \underset{j \in \{0,\dots,WH\}}{\arg\min} \left\| X_j^{n,k} - X_i^{m,k} \right\|.$$

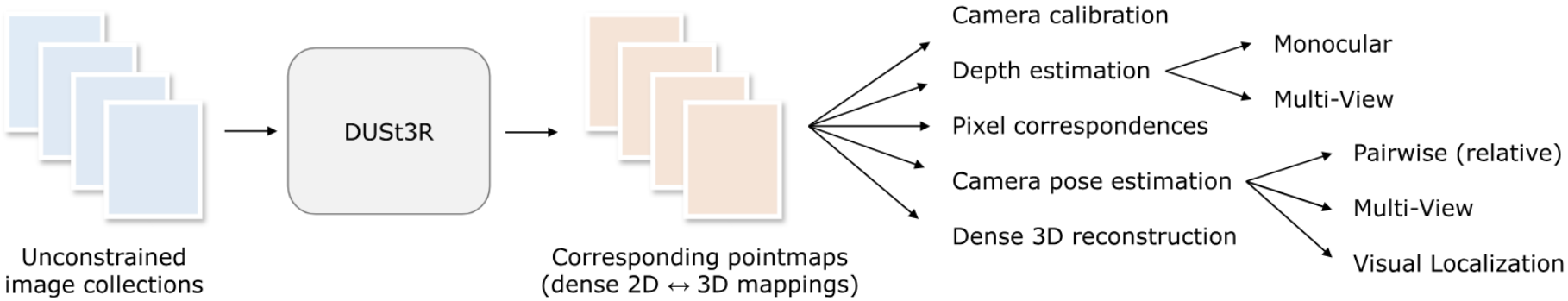


2. Recovering intrinsics

The pointmap is expressed in the first image coordinate frame (Extrinsic as identical matrix), and we assume that the principal point is approximately centered. We only need to estimate the focal lengths by minimize:

$$f_1^* = \underset{f_1}{\arg\min} \sum_{i=0}^{W} \sum_{j=0}^{H} C_{i,j}^{1,1} \left\| (i',j') - f_1 \frac{(X_{i,j,0}^{1,1}, X_{i,j,1}^{1,1})}{X_{i,j,2}^{1,1}} \right\|$$

Slide credit Wang et al.



3. Visual Localization

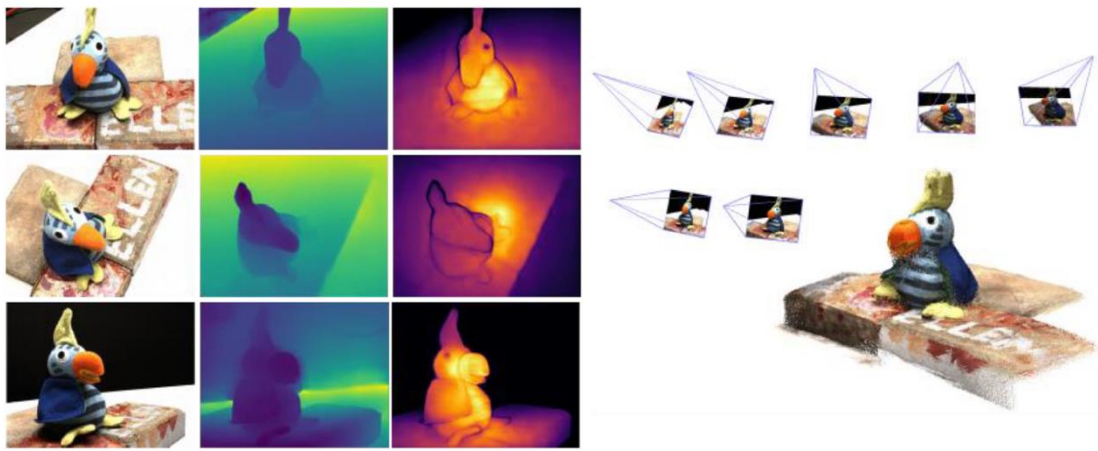
Given a **query image** and **retrieved database image**, the task can be achieved by :

(1) First build the pixel correspondences from point matching, which in turn yields 2D-3D correspondences. The camera pose is solved by the PnP-RANSAC with the estimated intrinsic.

(2) Estimate the relative pose by point matching, convert the pose to world coordinate by scaling (scale factor obtain from the predicted pointmap and ground truth pointmap of the database image)

$$R^*, t^* = \arg\min_{\sigma,R,t} \sum_i C_i^{1,1} C_i^{1,2} \left\| \sigma(RX_i^{1,1} + t) - X_i^{1,2} \right\|^2$$

4. Multi-view Pose Estimation



$$\chi^* = \arg\min_{\chi,P,\sigma} \sum_{e\in\mathcal{E}} \sum_{v\in e} \sum_{i=1}^{HW} C_i^{v,e} \left\| \chi_i^v - \sigma_e P_e X_i^{v,e} \right\|$$

a. Set up pairwise graph (like in SfM)
b. Joint optimization of 3D model, camera poses, and scale from the graph
c. Obtain all camera parameters from the 3D model

DUSt3R is Awesome. We can Extend it to Solve 4D Vision Problems

We'll cover 4D motion in this course!

MonST3R: A Simple Approach for Estimating Geometry in the Presence of Motion. Zhang et al.

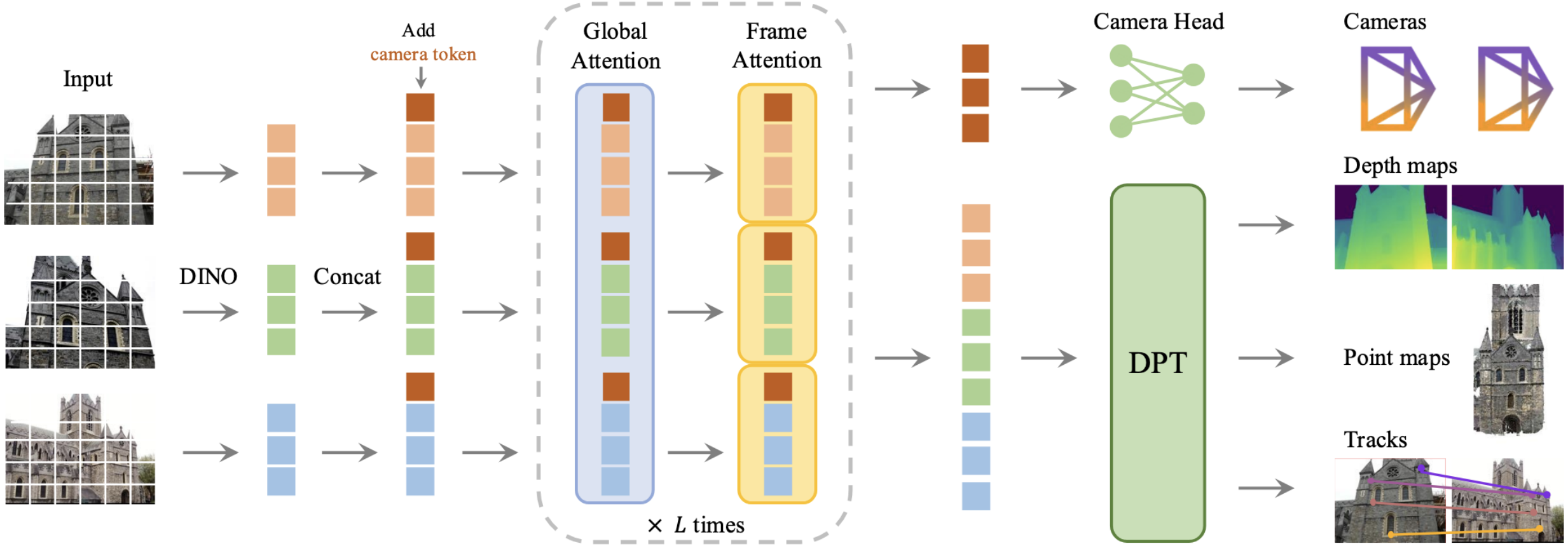
3D Foundation Model Has Made Great Progress

VGGT: Visual Geometry Grounded Transformer

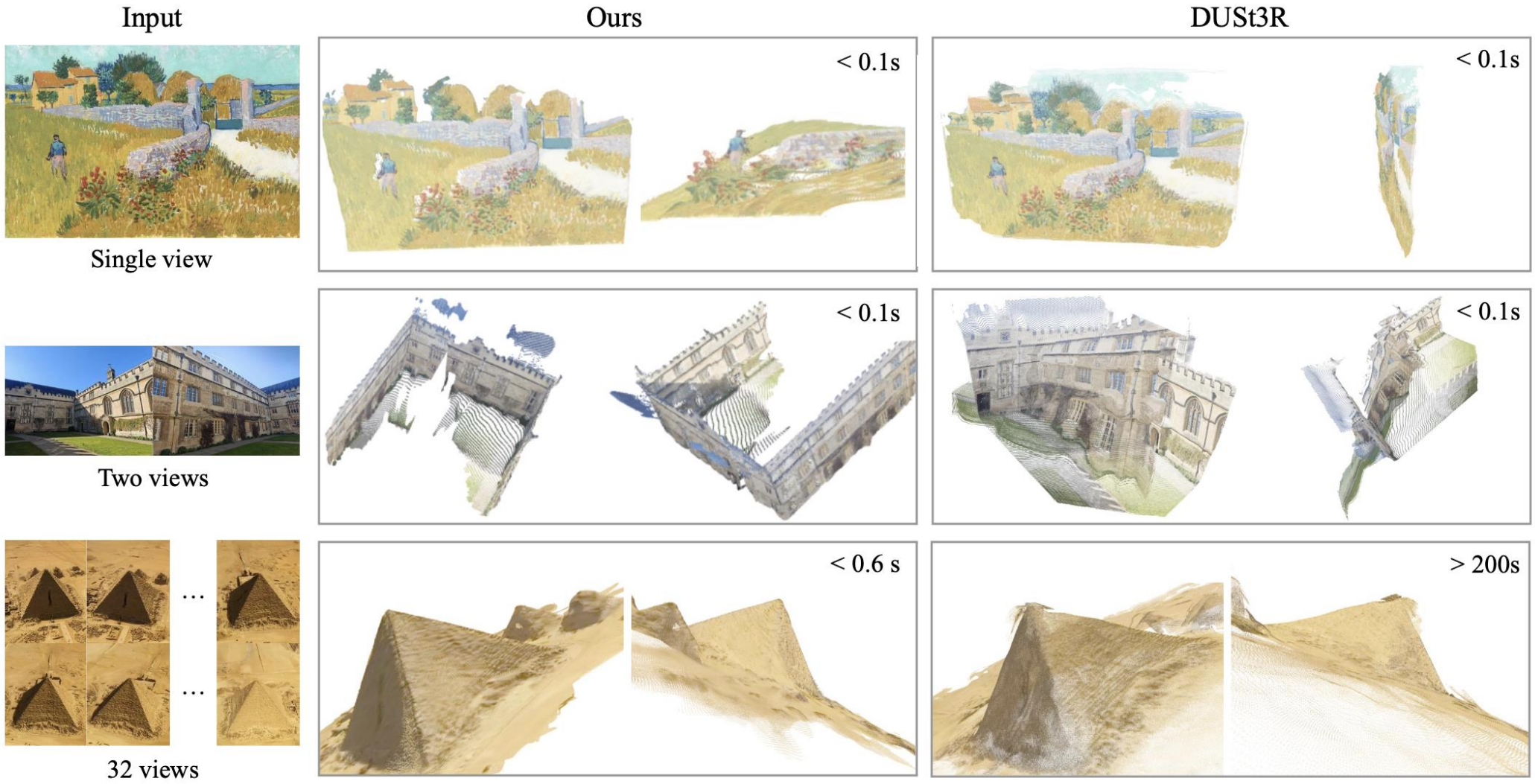
Jianyuan Wang[1,2] Minghao Chen[1,2] Nikita Karaev[1,2] Andrea Vedaldi[1,2]
Christian Rupprecht[1] David Novotny[2]

[1]Visual Geometry Group, University of Oxford [2]Meta AI

VGGT Goes Beyond Image Pairs



VGGT Outperforms DUSt3R



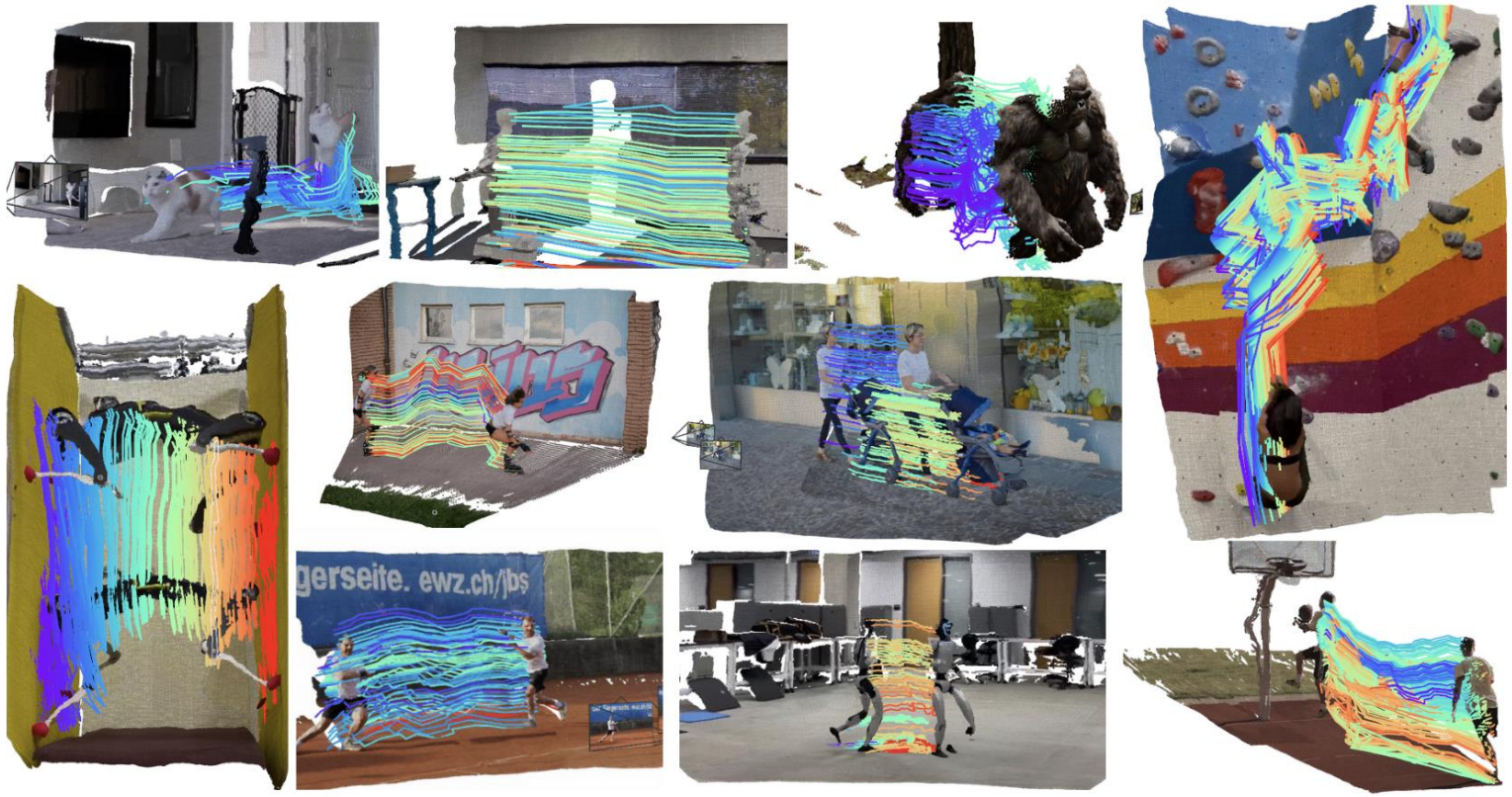
3D Foundation Model Has Made Great Progress

Any4D: Unified Feed-Forward Metric 4D Reconstruction

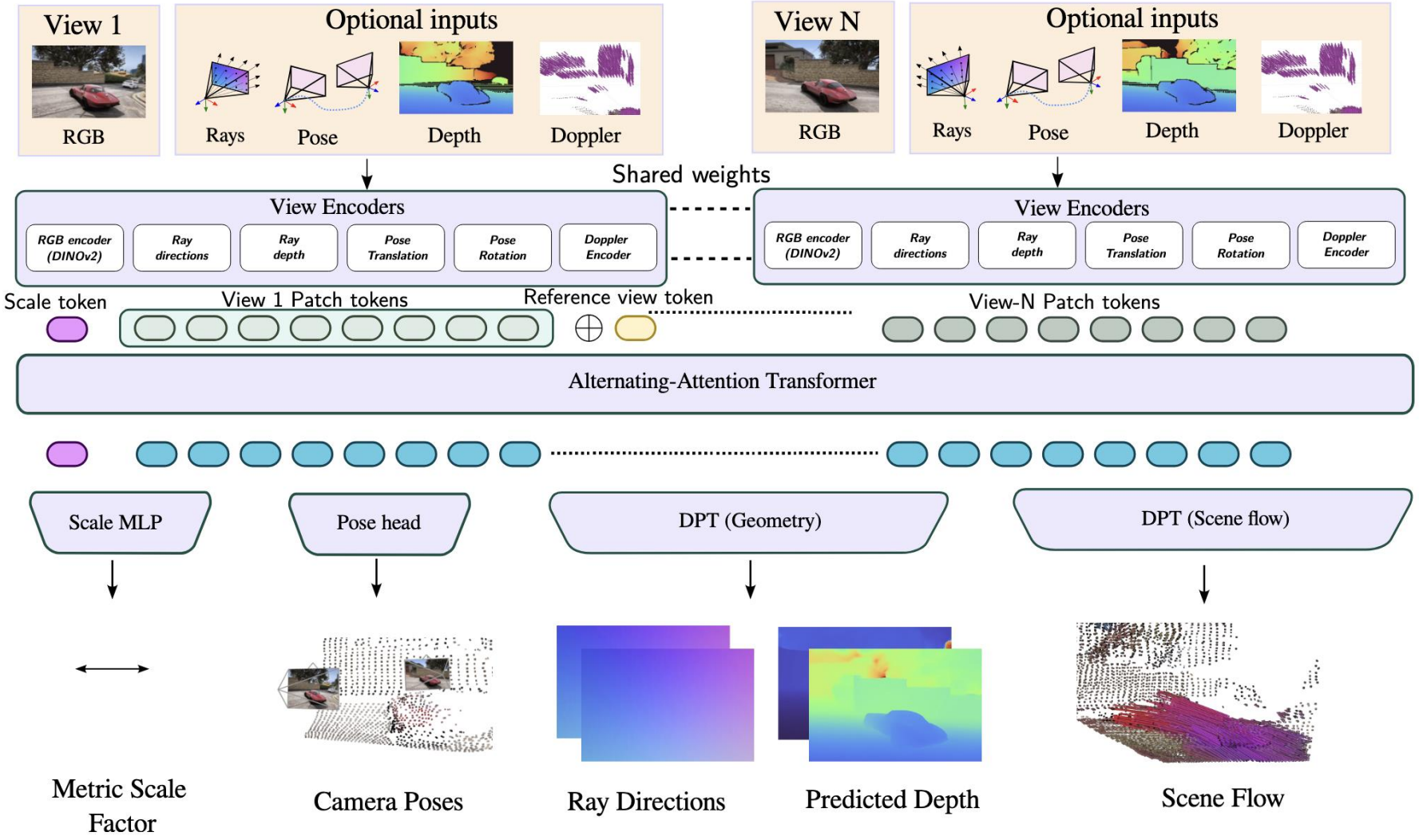
any-4d.github.io

Jay Karhade Nikhil Keetha Yuchen Zhang Tanisha Gupta
Akash Sharma Sebastian Scherer Deva Ramanan

Carnegie Mellon University



Any4D Goes Beyond 3D Reconstruction



Any4D Goes Beyond 3D Reconstruction

Content

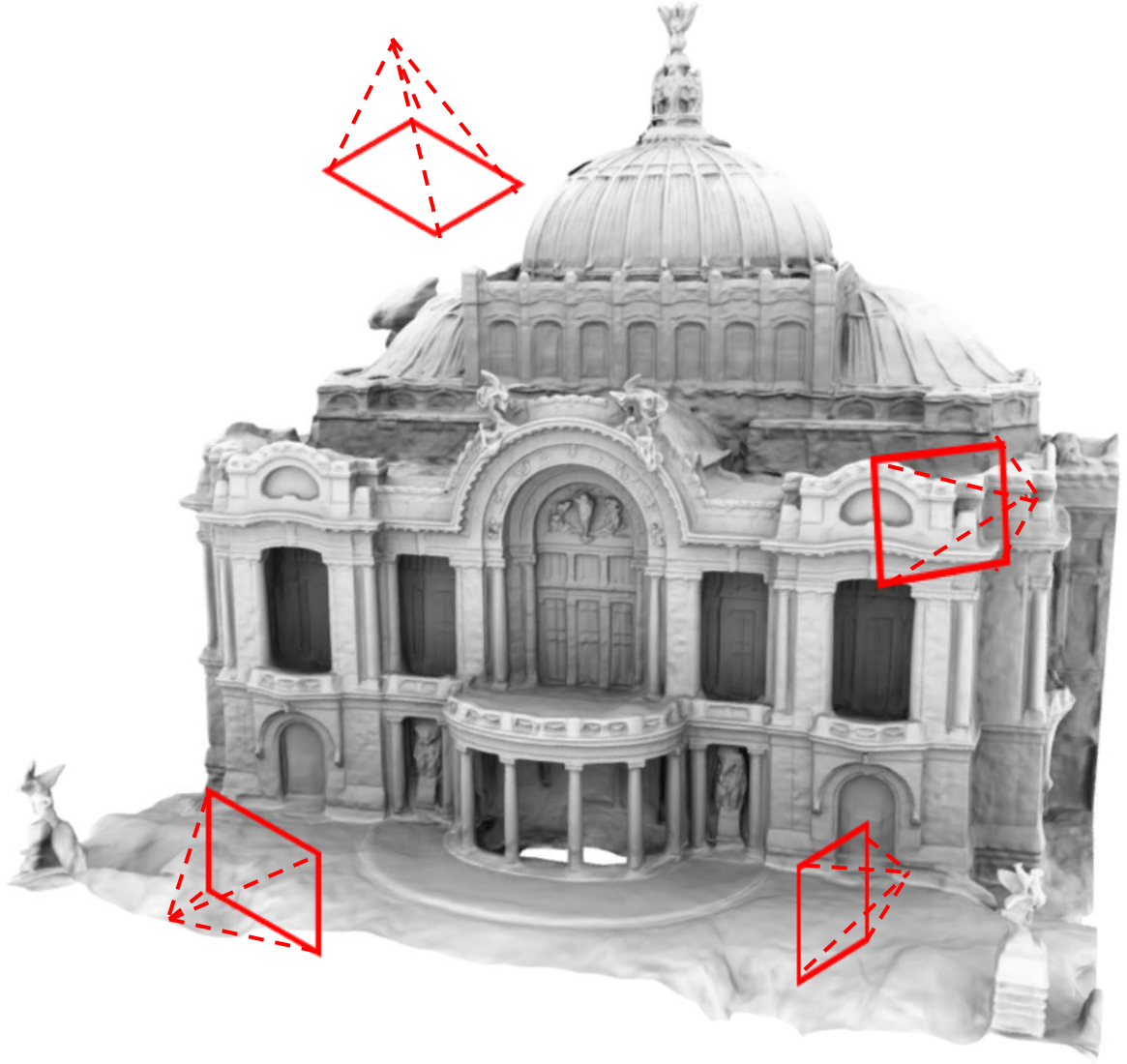
- Geometry Reconstruction
  - ➢Structure from Motion
  - ➢3D Foundation Model
- 3D Representations
  - ➢Depth Map
  - ➢Point Cloud
  - ➢Mesh
  - ➢Parametric Surface
  - ➢Signed Distance Function
  - ➢Voxels
  - ➢Others
- Conversion across Representations

Now We Can Reconstruct 3D from 2D Images
But How to Represent the 3D World Computationally?



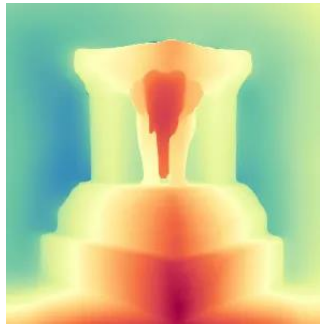




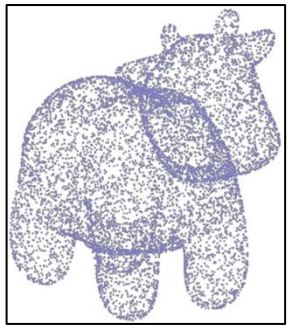


What is the Best Representation?
How to Convert Between Different Representations?

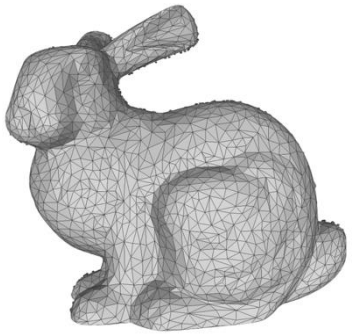
Depth map



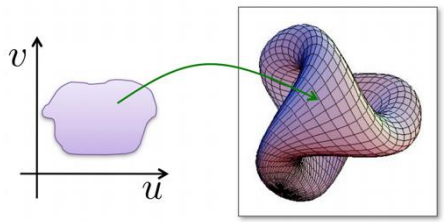
Pointcloud



Mesh

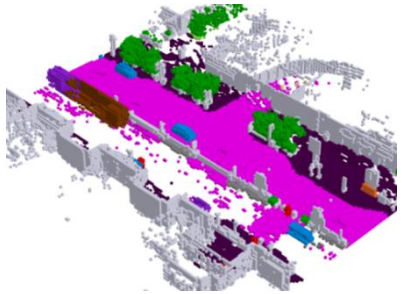


Parametric Surface

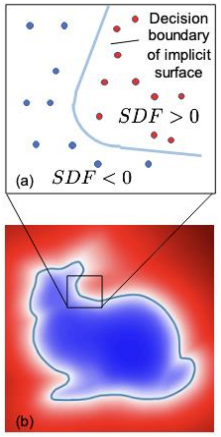


$$f(\mathbf{u}) = \mathbf{p} \in \mathbb{R}^3;\ \mathbf{u} \in \mathcal{M}$$

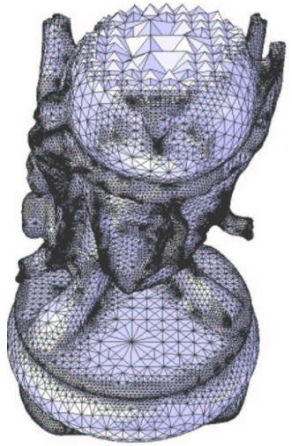
Voxels



Signed Distance Function



Tetrahedral Mesh

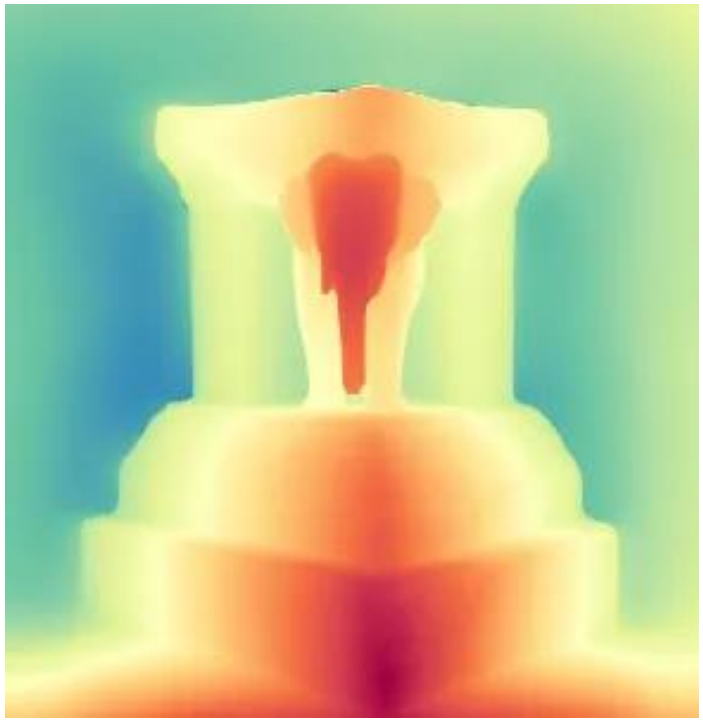


- What is the best representation?
  - Derivability from sensor data
  - Editability (modification, simplification, smoothing, filtering, repair …)
  - Rendering (rasterization, raytracing…)
  - Animation (physical modeling…)
  - Compatibility to machine learning (share structure across related shapes…)
- What are pros and cons of each type of representation?
- How to convert between different types of representations?

Content

- Geometry Reconstruction
  - ➢Structure from Motion
  - ➢3D Foundation Model
- 3D Representations
  - ➢Depth Map
  - ➢Point Cloud
  - ➢Mesh
  - ➢Parametric Surface
  - ➢Signed Distance Function
  - ➢Voxels
  - ➢Others
- Conversion across Representations

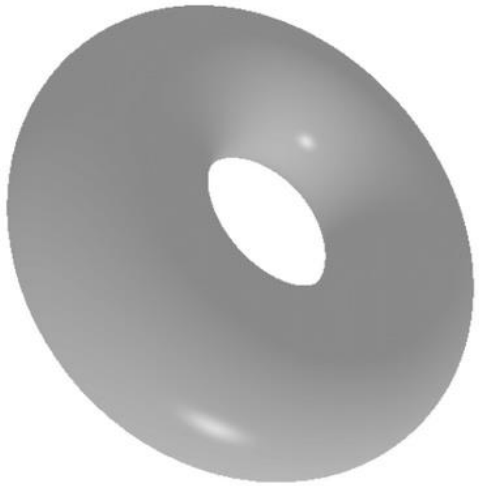
Depth Maps: A 2.5 D Representation



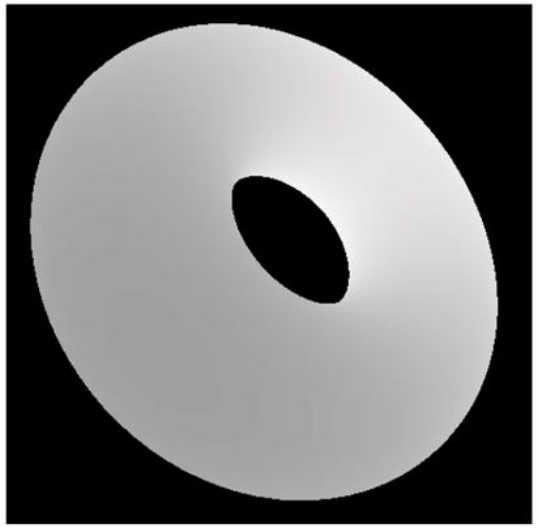
- Measure depth between each pixel and the camera center
- Depth map is ”2.5D” representations, associating distance with every image pixel

Image source Paul Bourke

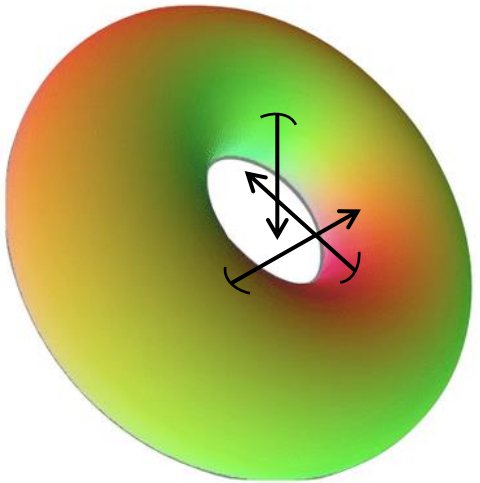
Depth Maps and Surface Normal Maps



Depth map



Surface normal map



Slide credit S. Tulsiani

From Depth Maps to Surface Normal Maps

- Let's parameterize a surface $\boldsymbol{S}$ in 3D space as:

$$\boldsymbol{r}(s,t) = (x(s,t), y(s,t), z(s,t))$$

- The surface normal $\boldsymbol{n}$ of $\boldsymbol{S}$ is:

$$\boldsymbol{n} = \frac{\partial \boldsymbol{r}}{\partial s} \times \frac{\partial \boldsymbol{r}}{\partial t}$$

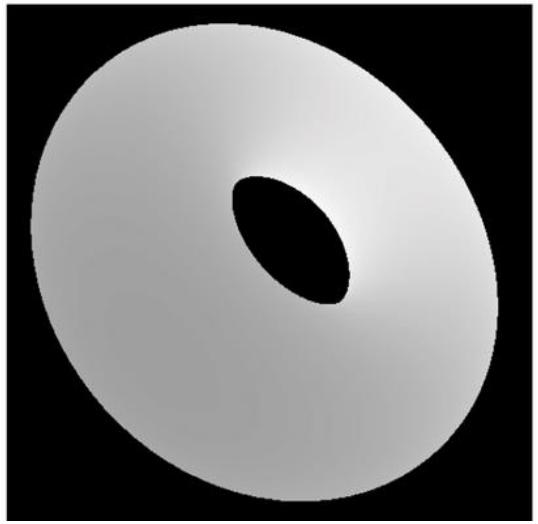
- For a depth map, the surface can be described as:

$$r(x,y) = (x, y, z(x,y))$$

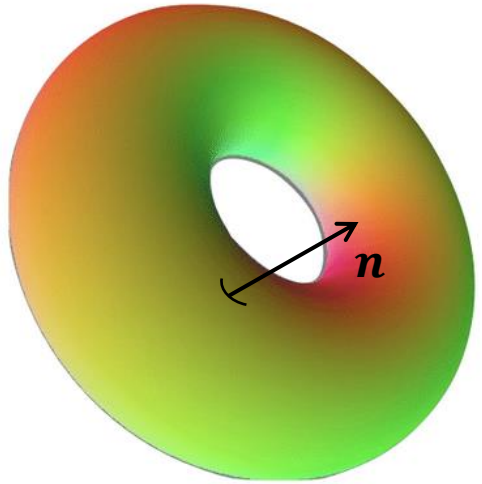
- The surface normal is:

$$\boldsymbol{n} = \frac{\partial \boldsymbol{r}}{\partial s} \times \frac{\partial \boldsymbol{r}}{\partial t} = \begin{bmatrix} 1 \\ 0 \\ \frac{\partial \boldsymbol{z}}{\partial x} \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ \frac{\partial \boldsymbol{z}}{\partial y} \end{bmatrix} = \begin{bmatrix} -\frac{\partial \boldsymbol{z}}{\partial x} \\ -\frac{\partial \boldsymbol{z}}{\partial y} \\ 1 \end{bmatrix}$$

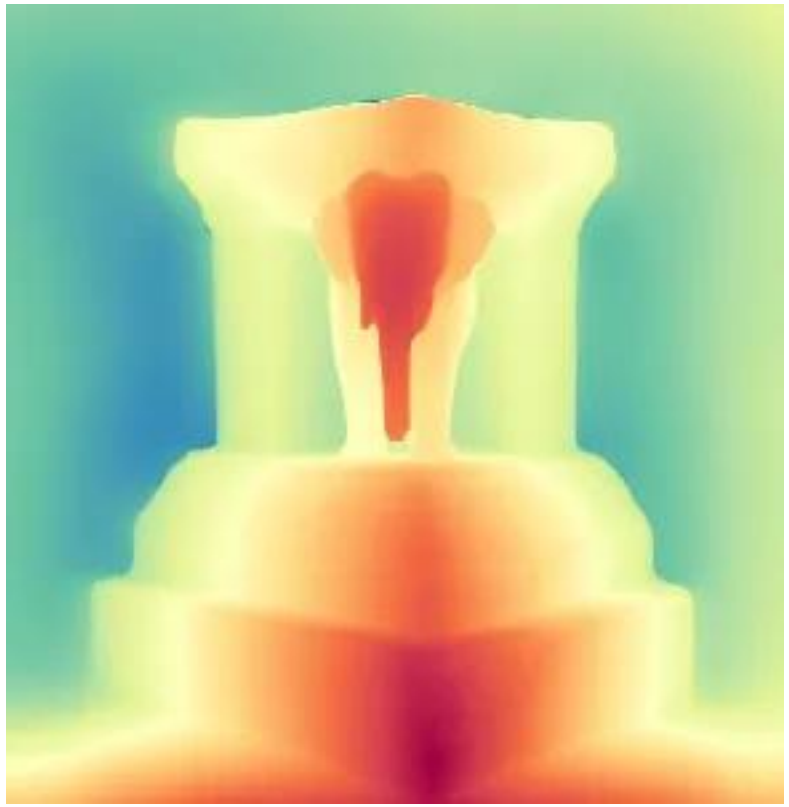
Depth map



Surface normal map



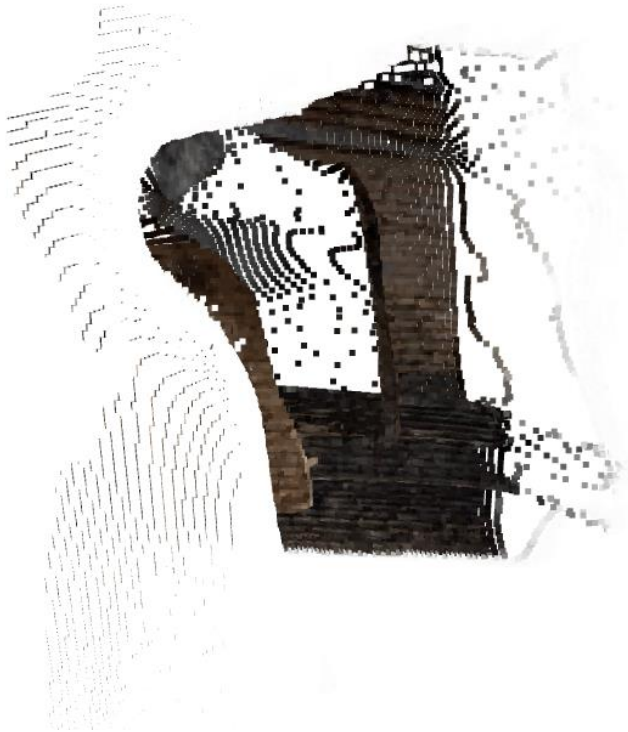
From Depth Maps to 3D Points



- Measure depth between each pixel and the camera center
- We can convert a set of pixels into a set of 3D points with the depth map if camera intrinsic is known $X = x\frac{Z}{f}$

Image source Paul Bourke

A Depth Map Captures Not the “Full” But “Visible” 3D Structure



A Depth Map Captures Not the “Full” But “Visible” 3D Structure

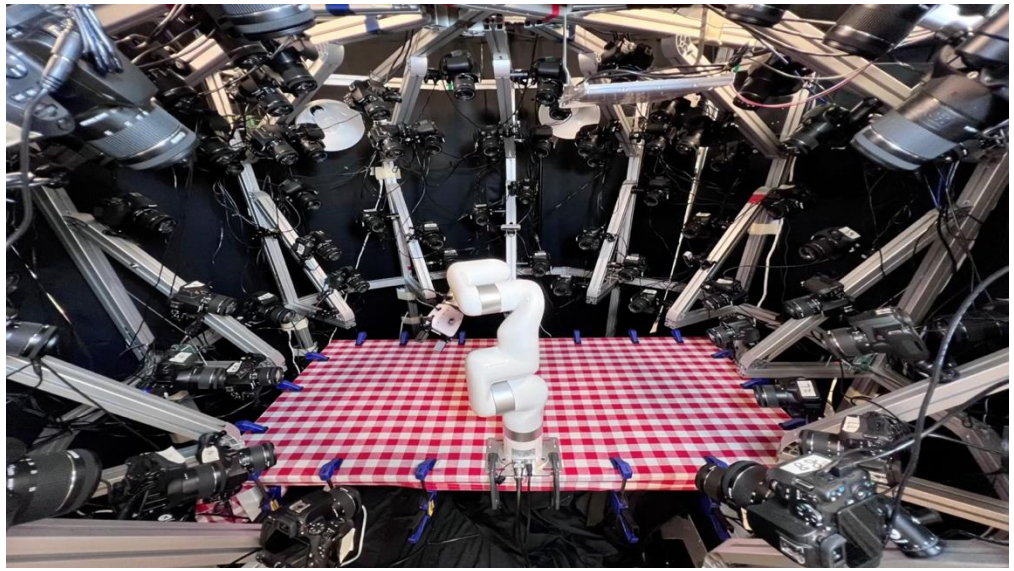


Image source: Robo360: A 3D Omnispective MultiMaterial Robotic Manipulation Dataset

Task. *Take something out of a drawer.*
(Robot Cfg. 4)

Video source: RH20T: A Comprehensive Robotic Dataset for Learning Diverse Skills in One-Shot

Content

- Geometry Reconstruction
  - Structure from Motion
  - 3D Foundation Model
- 3D Representations
  - Depth Map
  - Point Cloud
  - Mesh
  - Parametric Surface
  - Signed Distance Function
  - Voxels
  - Others
- Conversion across Representations

Going Beyond Image-based 2.5D Representations. Point Clouds: a Set of Unordered 3D Points

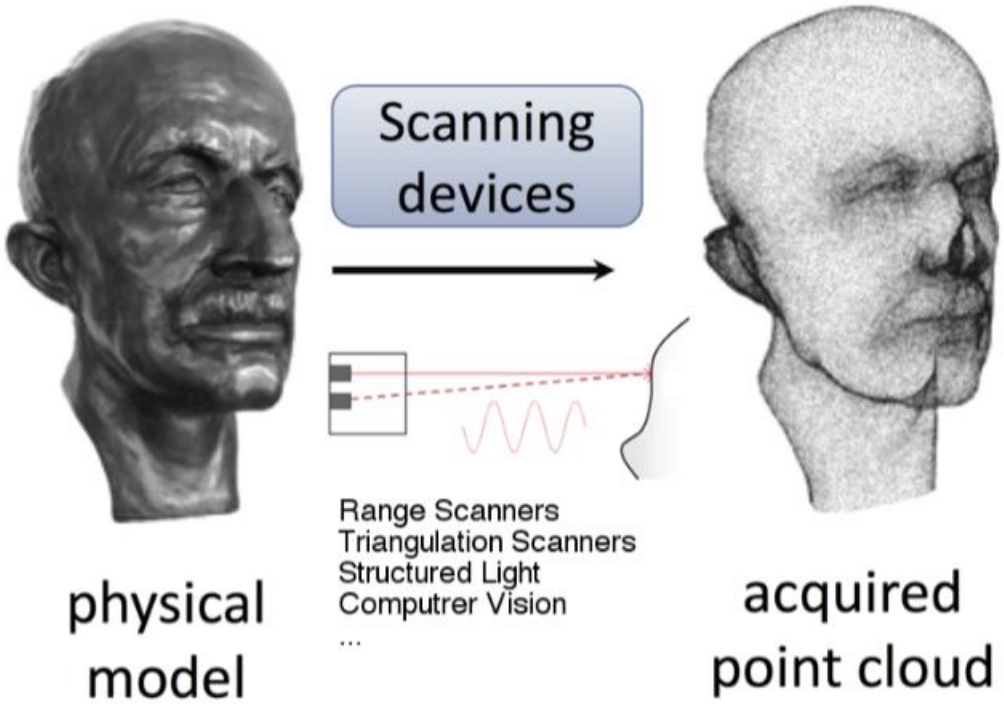
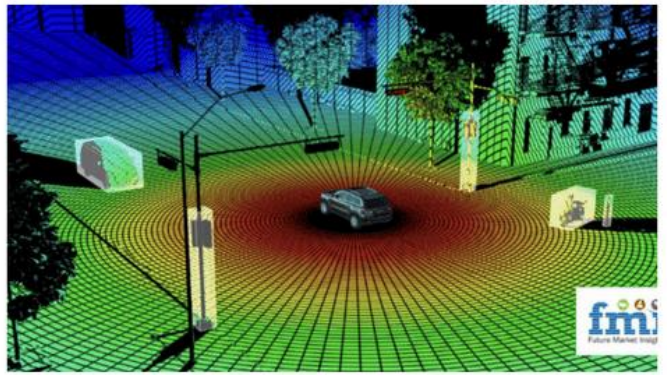


Image credit L. Guibas



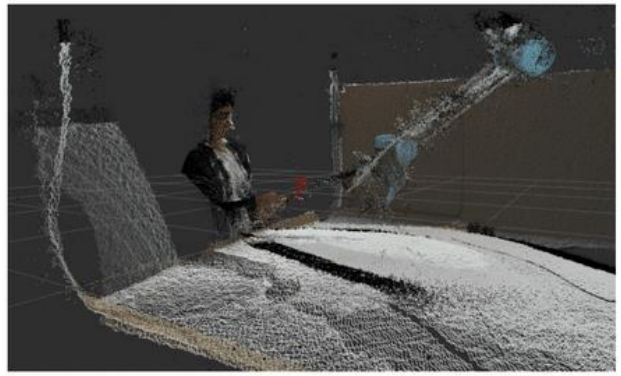
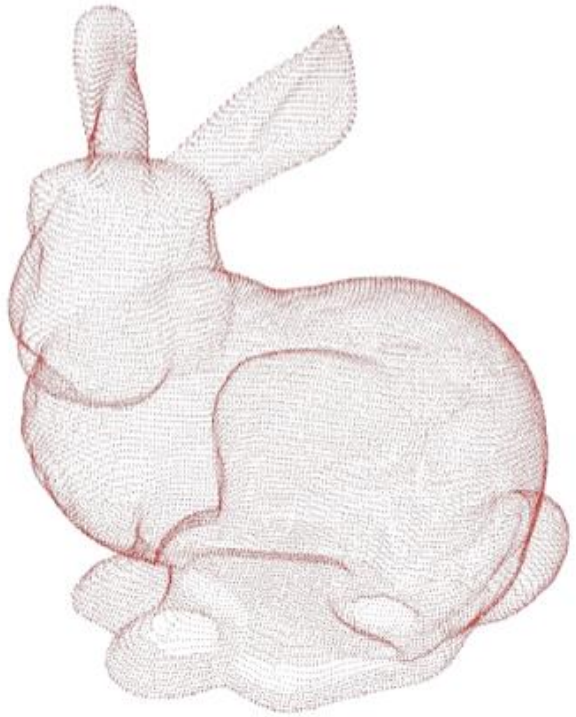
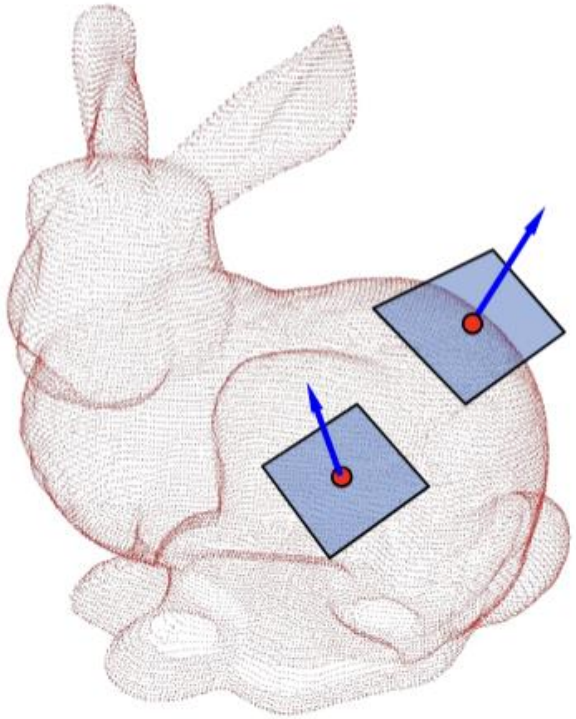


Image credit S. Tulsiani

Point Clouds: A Simple Surface Representations



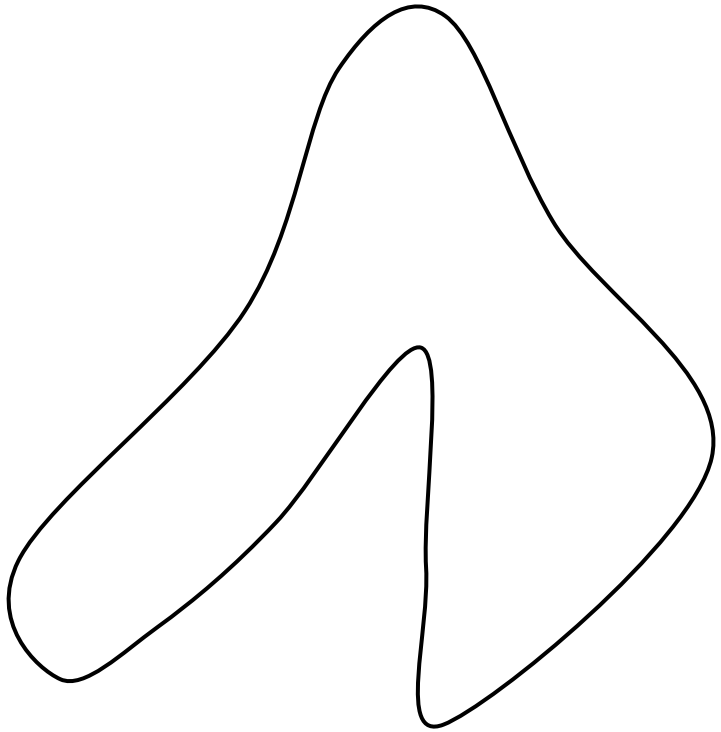


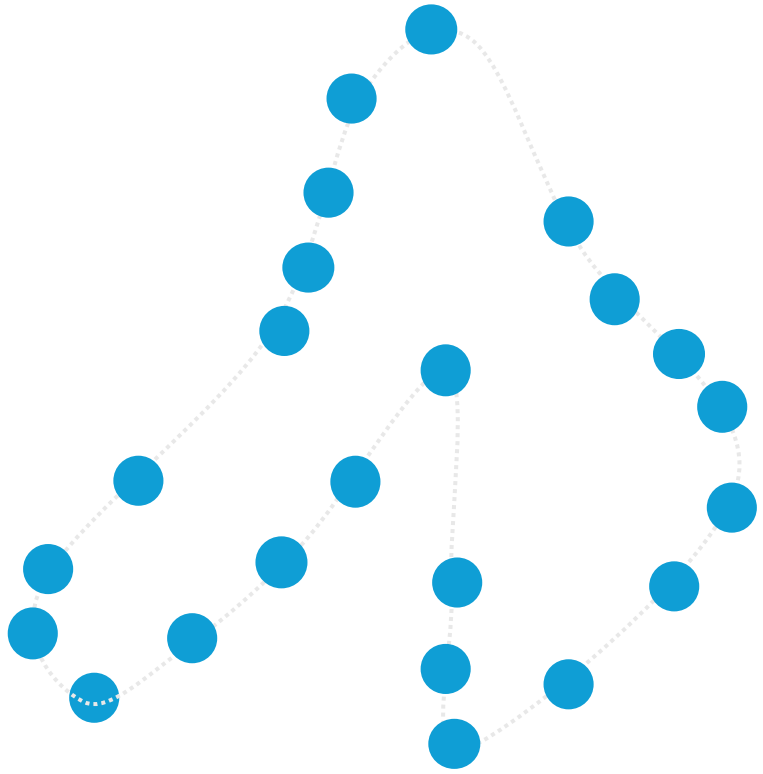
- Point cloud denotes a set of unordered 3D points $\{(x_i, y_i, z_i)|\forall i\}$ covering the object surface
- **Surfels** denotes a set of unordered 3D oriented points $\{(x_i, y_i, z_i, , \boldsymbol{n}_i)|\forall i\}$
- Point cloud is the simplest 3D representation
- A set of unordered points is invariant to permutation

Slide adapted from L. Guibas

Point Clouds Have Problems

- Points could be irregularly distributed among the object surface





Point Clouds Have Problems

- Points could be irregularly distributed among the object surface
  - ➢ Point clouds may have holes

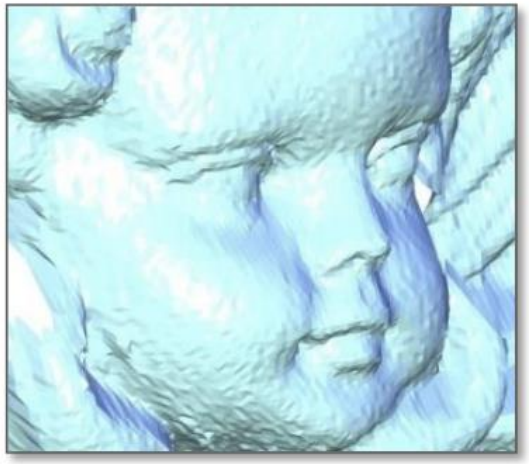
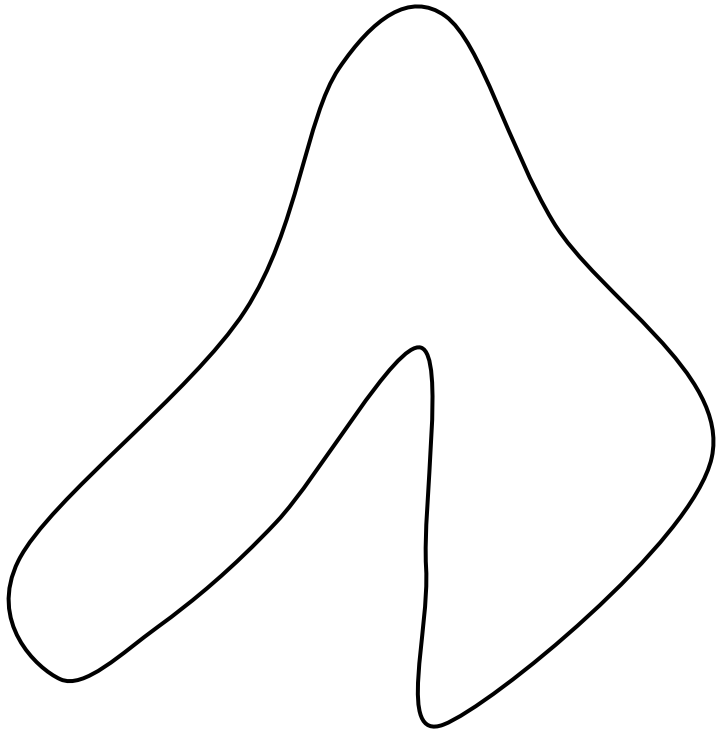
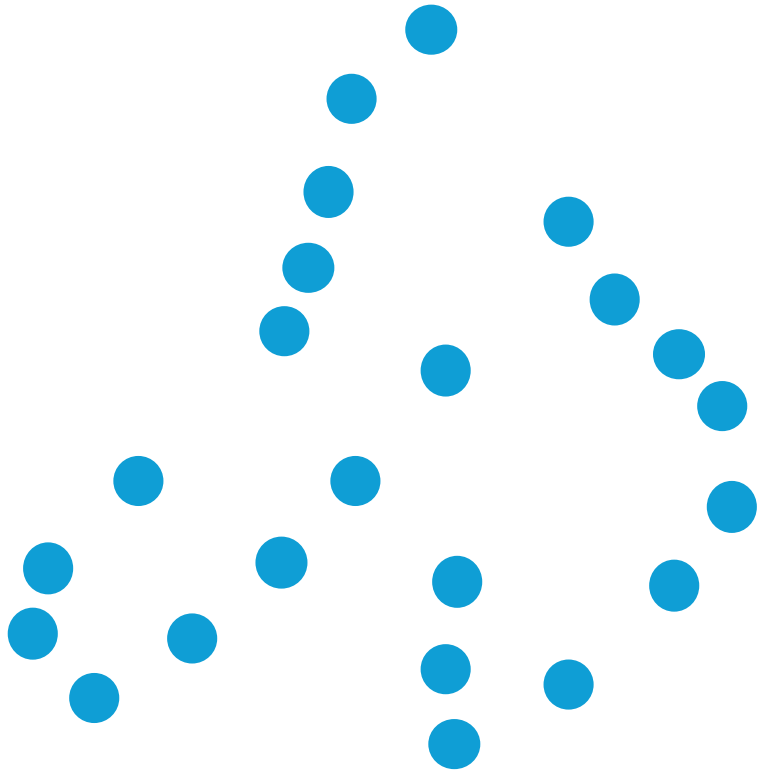


Image credit L. Guibas

Point Clouds Have Problems

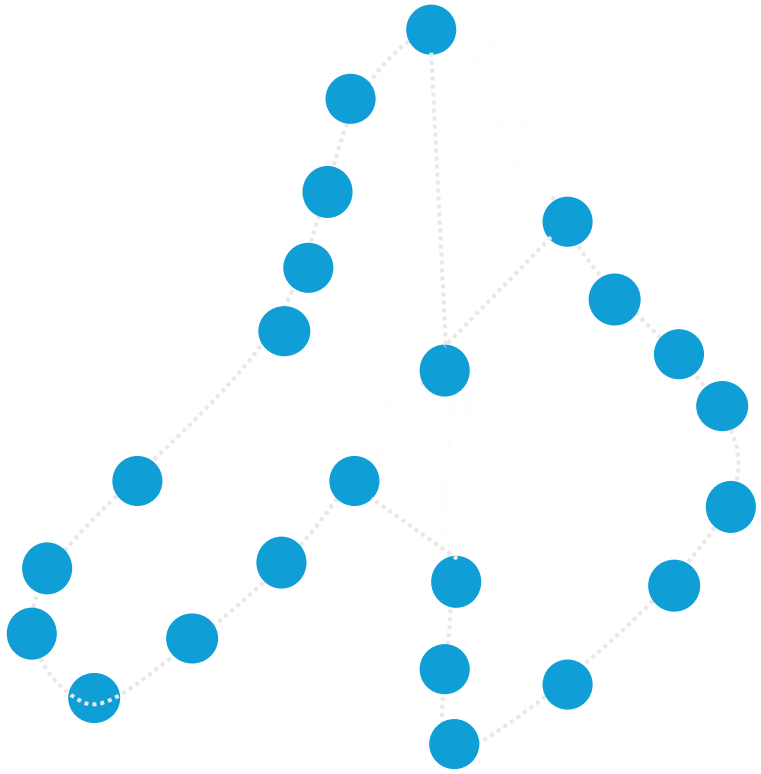
- Points could be irregularly distributed among the object surface
- Point Clouds have no connectivity

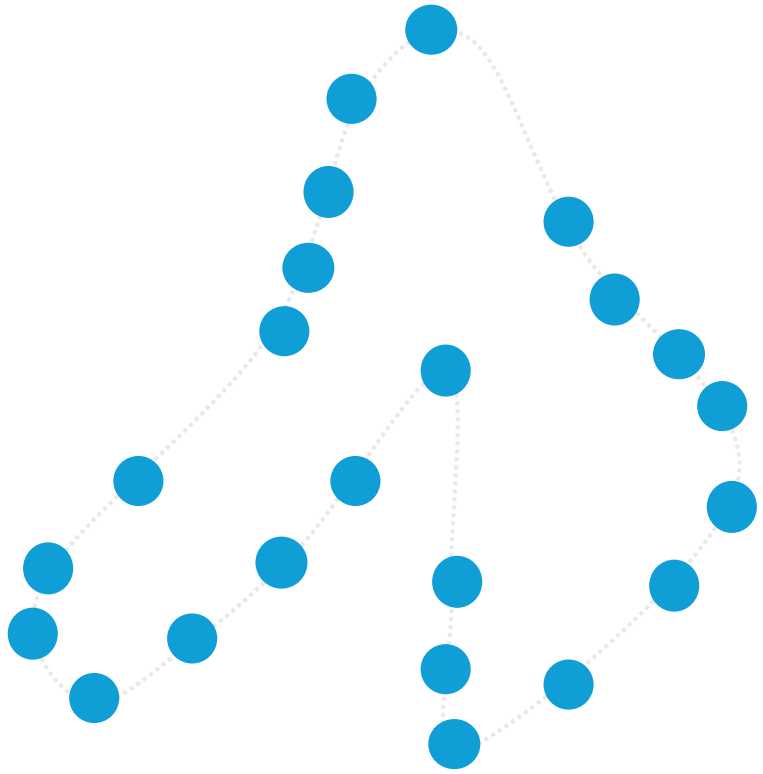




Point Clouds Have Problems

- Points could be irregularly distributed among the object surface
- Point Clouds have no connectivity
  - ➢ Reconstruction of object surface becomes ambiguous





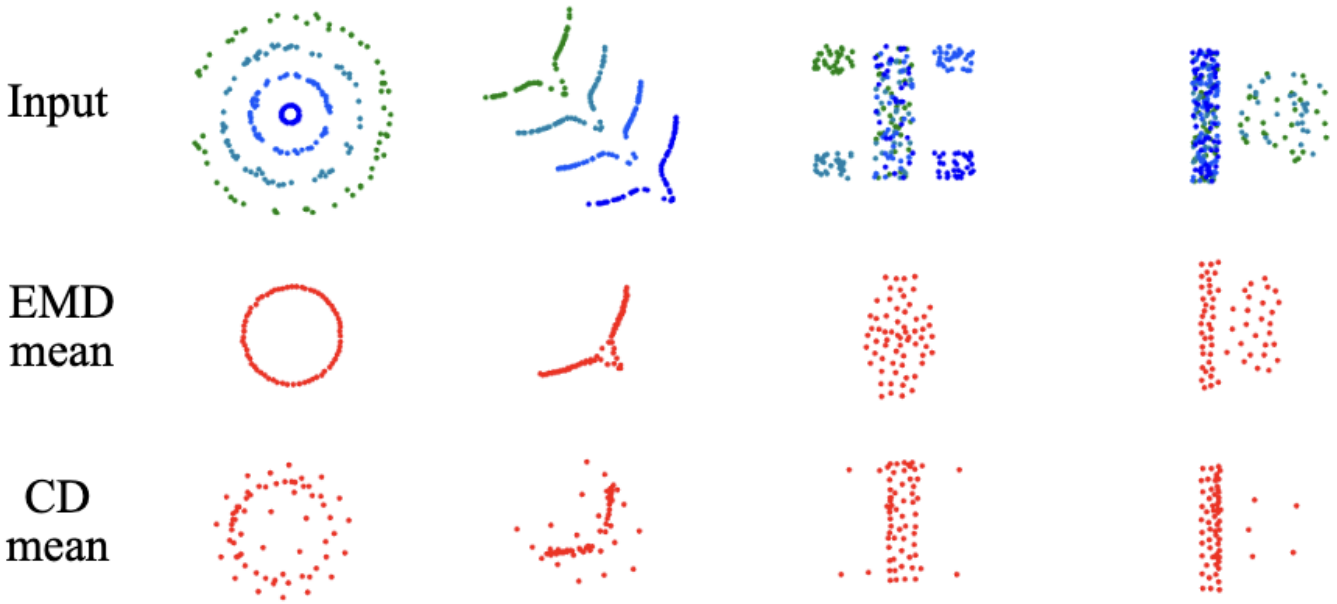
Metrics for Point Clouds

Chamfer distance

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2$$

Earth Mover's distance

$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2 \quad \text{where } \phi : S_1 \to S_2 \text{ is a bijection.}$$



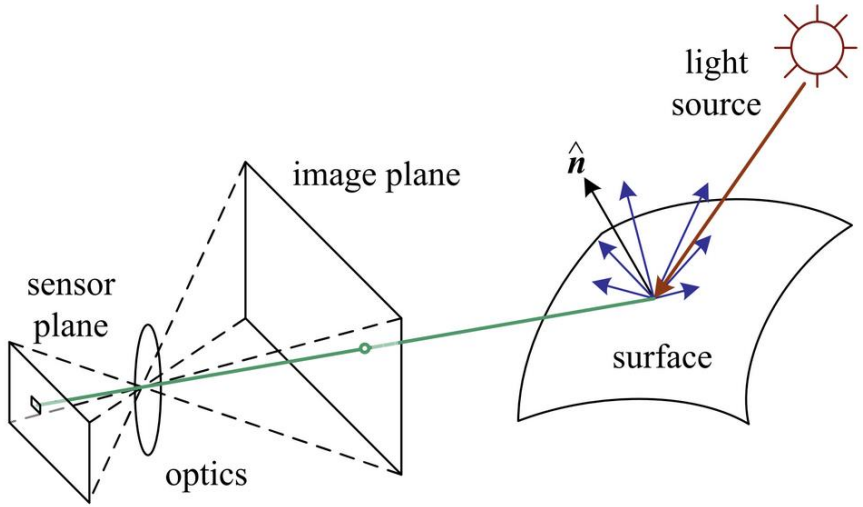
Given a shape distribution $\mathbb{S}$ (blue and green points) and distance function L, find the mean shape x (red points) that minimize:

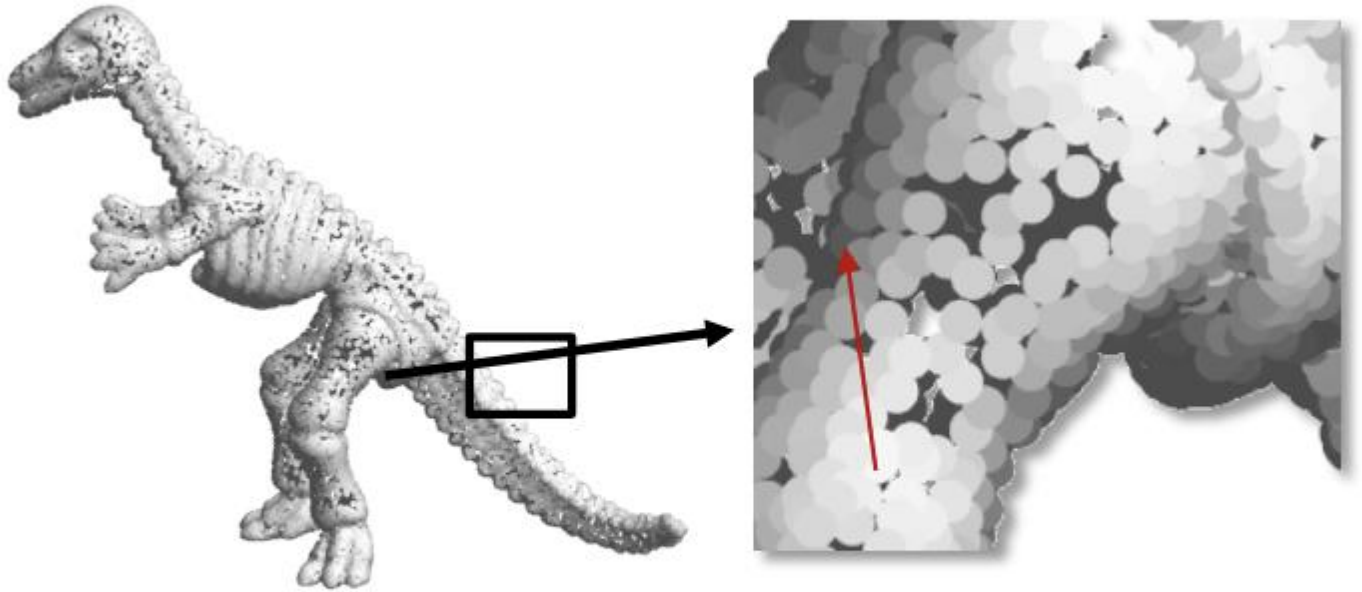
$$\mathrm{E}_{s \sim \mathbb{S}}[L(x, s)]$$

A Point Set Generation Network for 3D Object Reconstruction from a Single Image. Fan et al.

How to Render Images from Point Clouds

- Rendering: Project 3D models to 2D images
- Treat points as small flat disks instead of tiny spheres
- Rendering with shadows requires surface normals





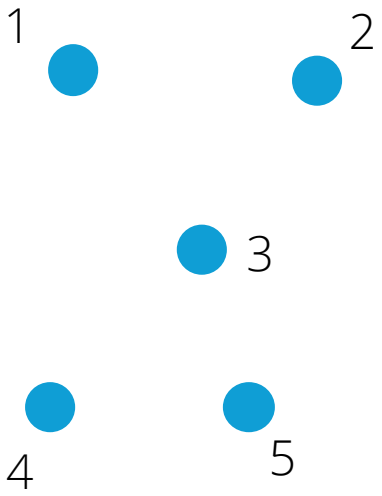
shading needs normals!

Slide adapted from S. Tulsiani

Content

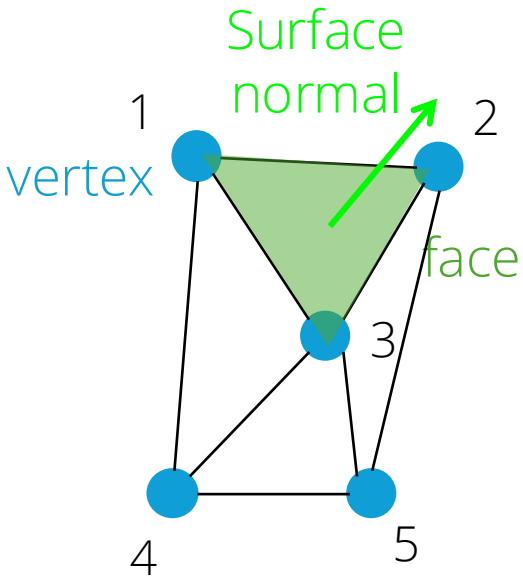
- Geometry Reconstruction
  - ➢Structure from Motion
  - ➢3D Foundation Model
- 3D Representations
  - ➢Depth Map
  - ➢Point Cloud
  - ➢Mesh
  - ➢Parametric Surface
  - ➢Signed Distance Function
  - ➢Voxels
  - ➢Others
- Conversion across Representations

Point-based vs. Graph-based Representations



Points
x_1, y_1, z_1
x_2, y_2, z_2
x_3, y_3, z_3
x_4, y_4, z_4
x_5, y_5, z_5

The order doesn't matter



Vertices	
v_1	x_1, y_1, z_1
v_2	x_2, y_2, z_2
v_3	x_3, y_3, z_3
v_4	x_4, y_4, z_4
v_5	x_5, y_5, z_5

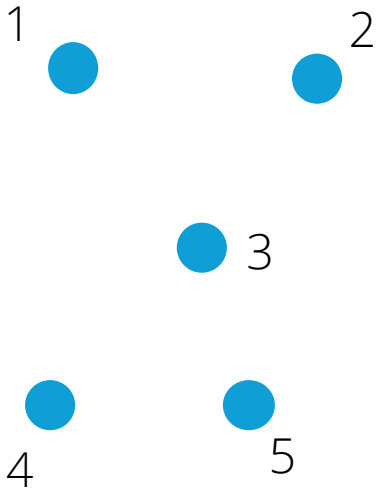
Coordinates of each vertex

Faces	
t_1	v_1, v_2, v_3
t_2	v_3, v_5, v_4
	...

Vertex indices of a face

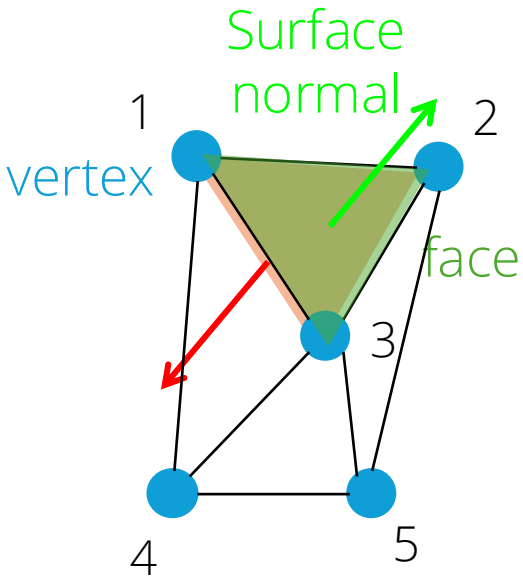
- The order of the vertex list doesn't matter
- The order of vertex indices in a face matters, which determine the direction of the surface normal

Point-based vs. Graph-based Representations



Points
x_1, y_1, z_1
x_2, y_2, z_2
x_3, y_3, z_3
x_4, y_4, z_4
x_5, y_5, z_5

The order doesn't matter



Vertices	
v_1	x_1, y_1, z_1
v_2	x_2, y_2, z_2
v_3	x_3, y_3, z_3
v_4	x_4, y_4, z_4
v_5	x_5, y_5, z_5

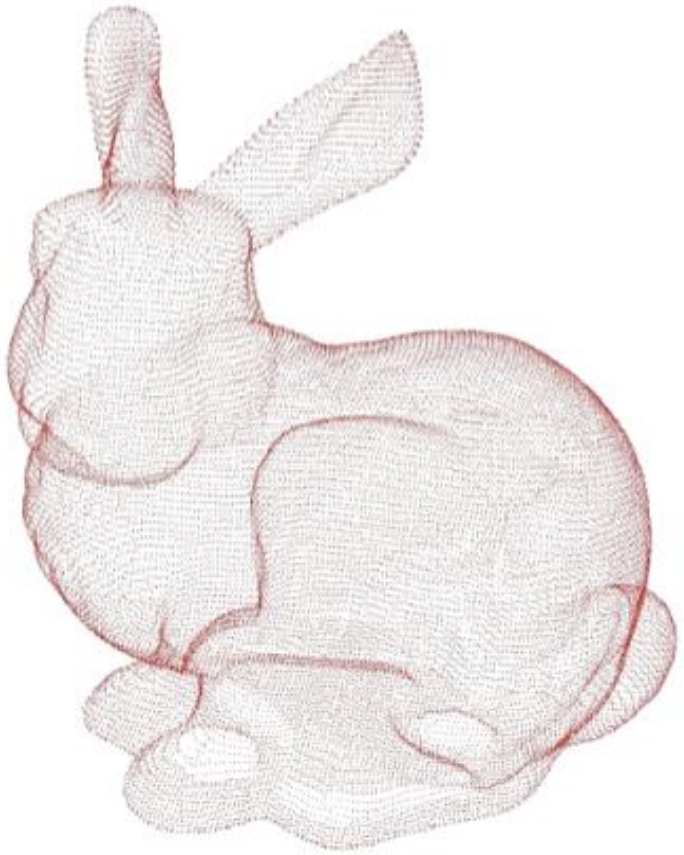
Coordinates of each vertex

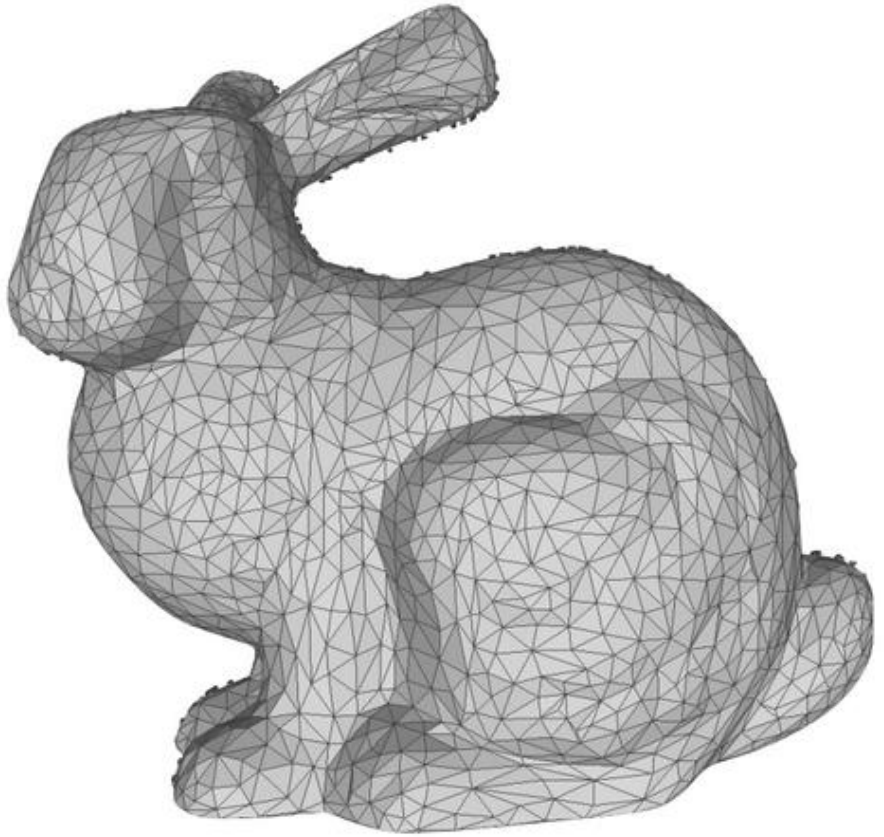
Faces	
t_1	v_1, v_2, v_3
t_1'	v_1, v_3, v_2
t_2	v_3, v_5, v_4
	...

Vertex indices of a face

- The order of the vertex list doesn't matter
- The order of vertex indices in a face matters, which determine the direction of the surface normal

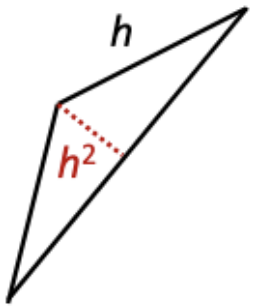
Point Clouds vs. (Triangular) Meshes

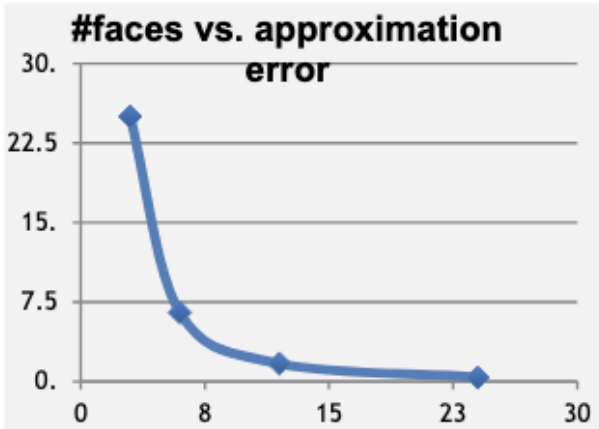


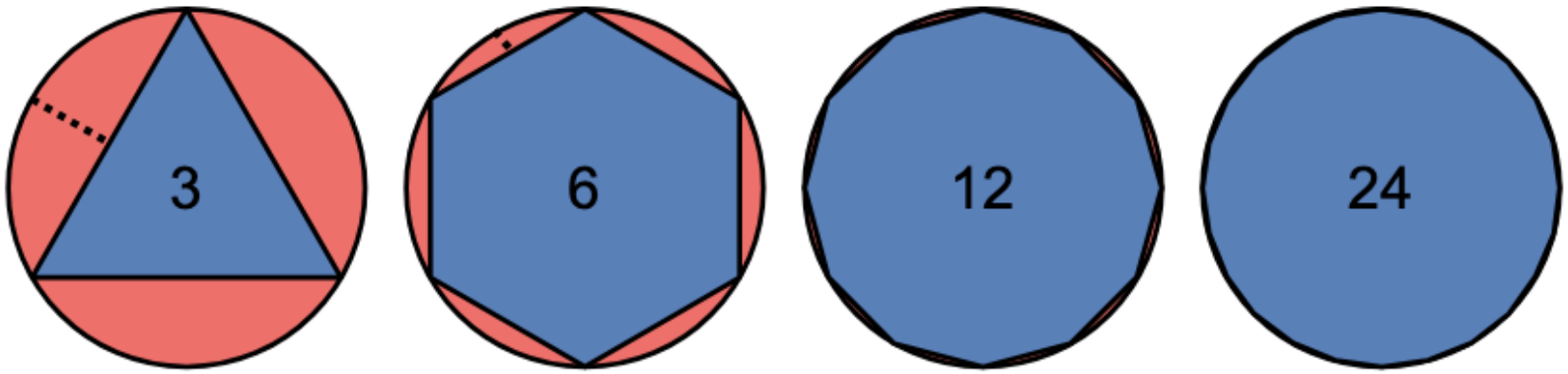


Meshes as Approximations of Smooth Surfaces

- Piecewise linear approximation
  - Error is $O(h^2)$ [$O(h)$ for points]

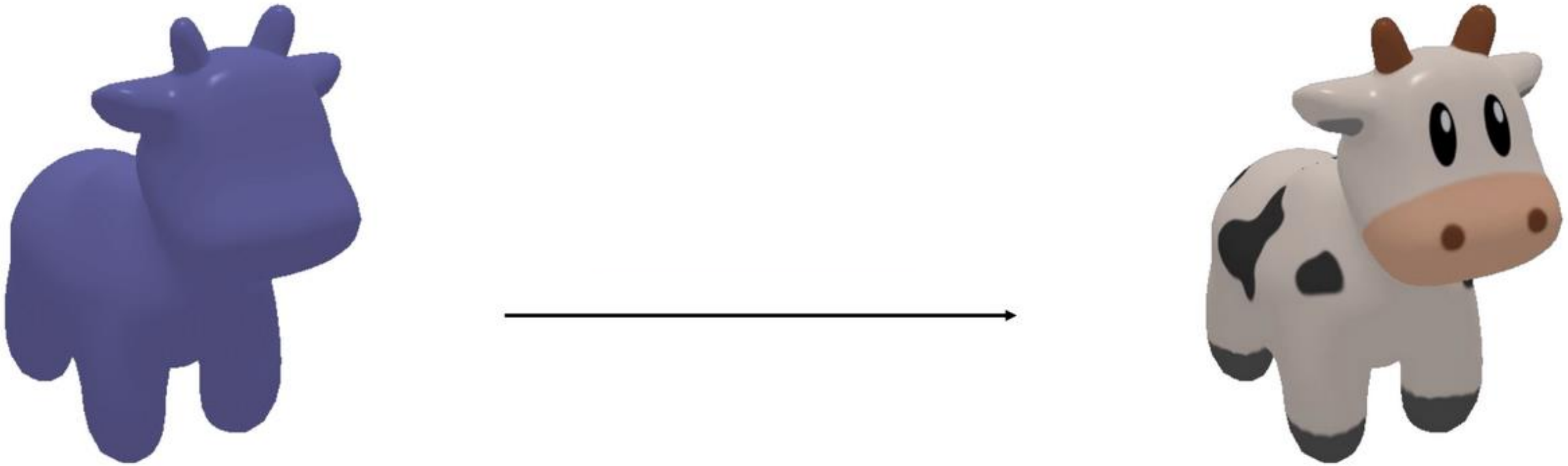




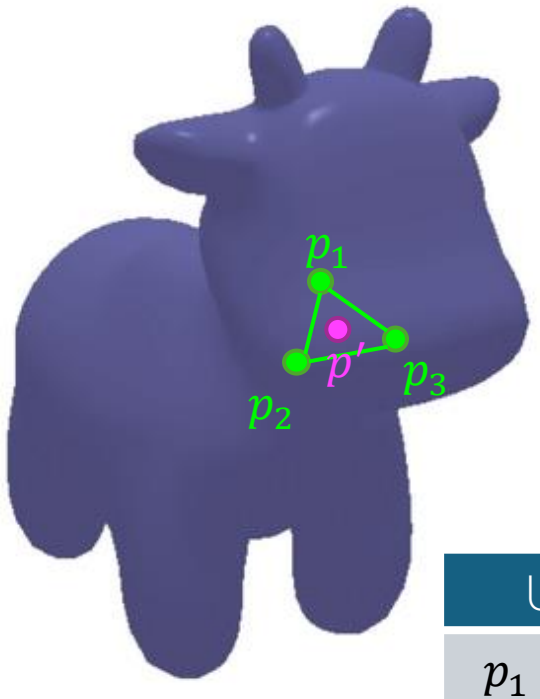


25% 6.5% 1.7% 0.4%

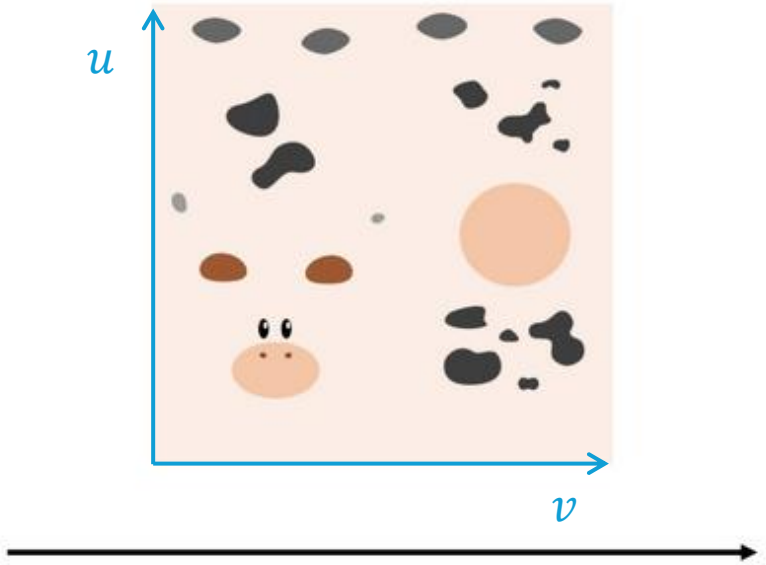
Texture Mapping of Meshes



Texture Mapping of Meshes



Texture map

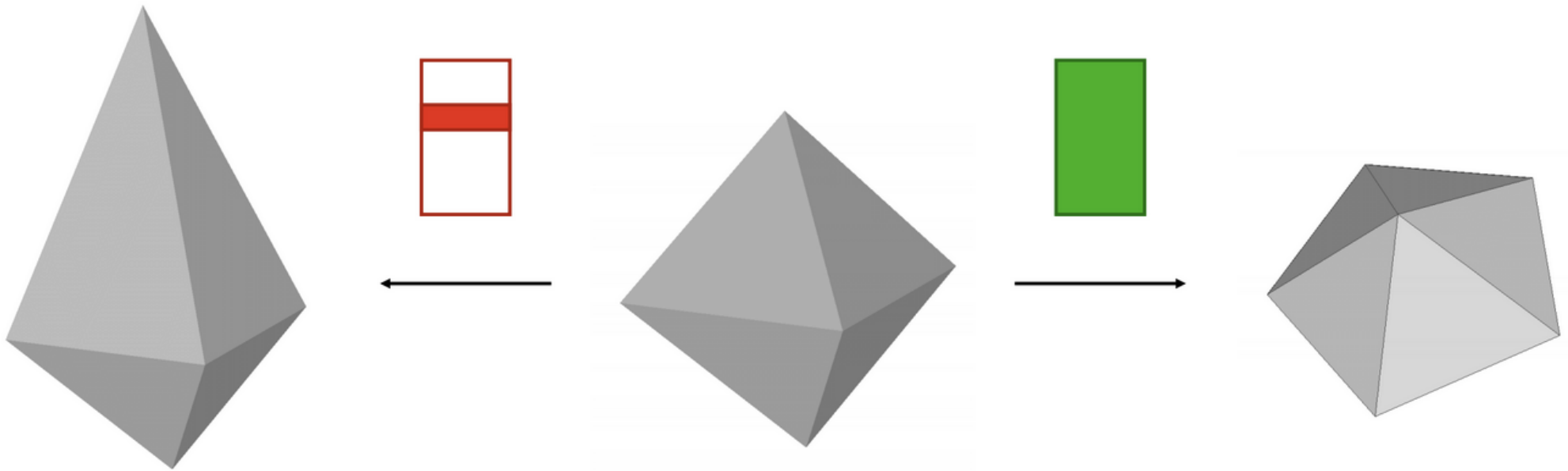


UV coordinates	
p_1	u_1, v_1
p_2	u_2, v_2
p_3	u_3, v_3

- Storage the u, v coordinate of the texture map for each vertex
- Obtain the texture of a point by interpolation

Slide credit S. Tulsiani

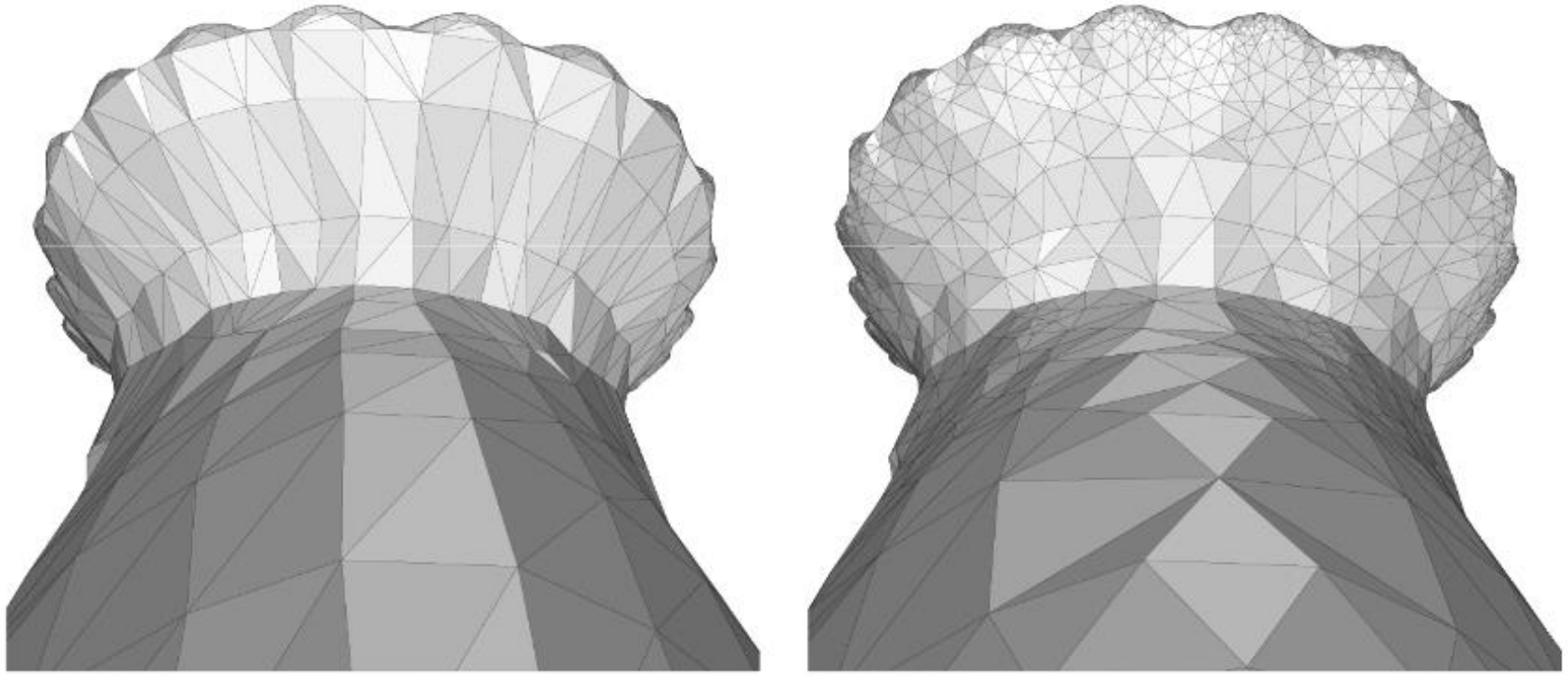
Modifying Meshes is a Big Research Problem



Modifying vertex positions can edit shape

But other changes are non-trivial

Example: Removing Skinny Triangles



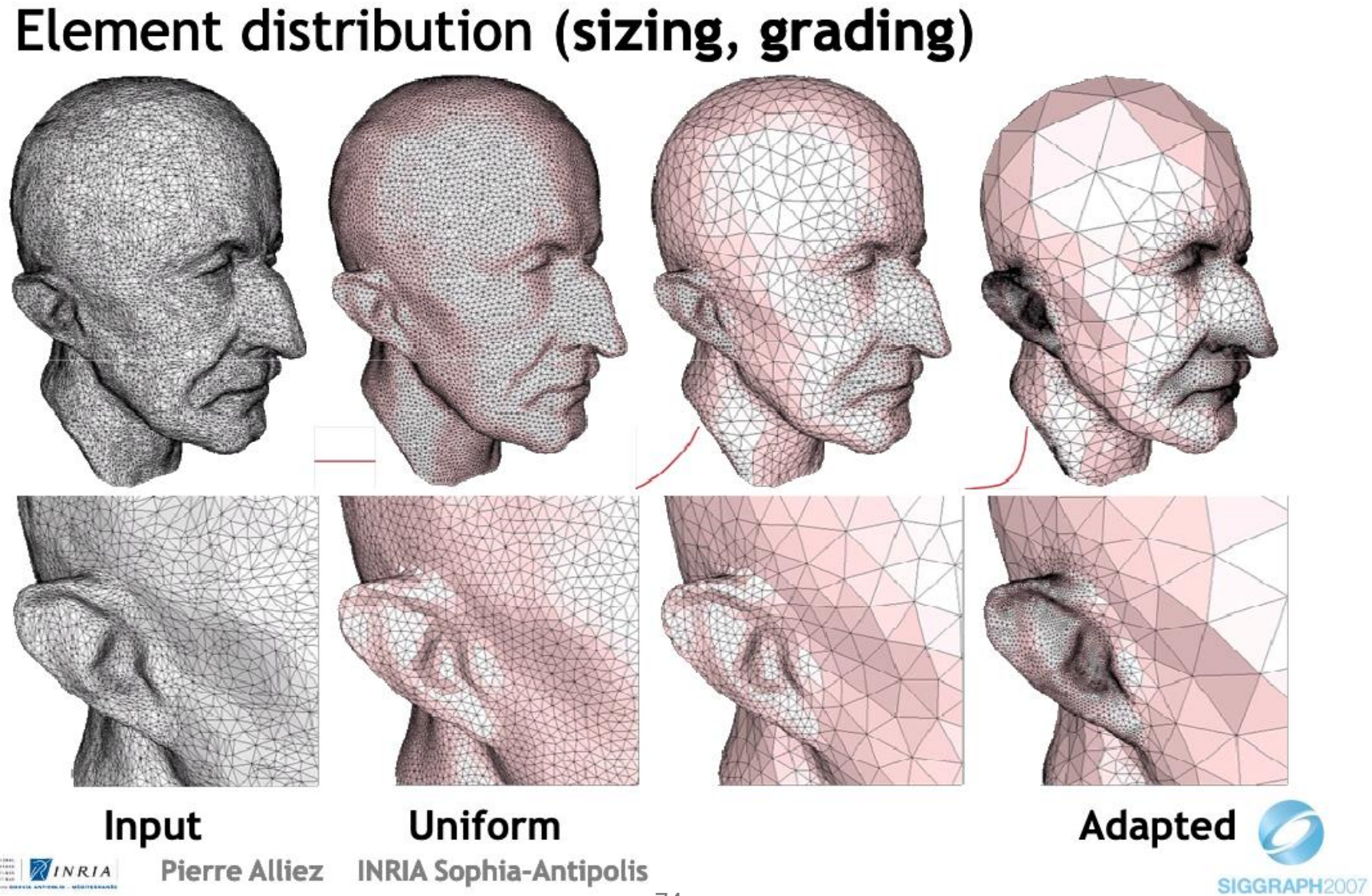
[Boissonnat-Oudot 03]

Pierre Alliez INRIA Sophia-Antipolis

SIGGRAPH2007

Slide credit P. Alliez

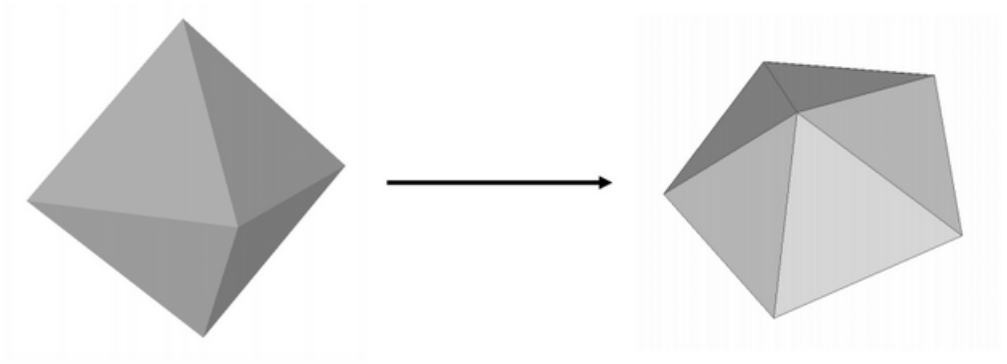
Example: Adaptively Refining Triangles



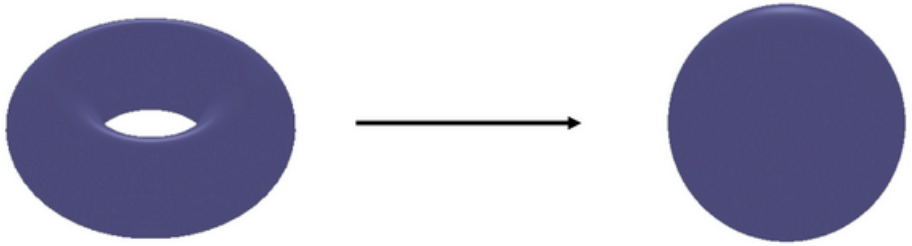
Slide credit P. Alliez

Modifying Meshes is a Big Research Problem

Non-trivial even though both have 8 vertices and 10 faces



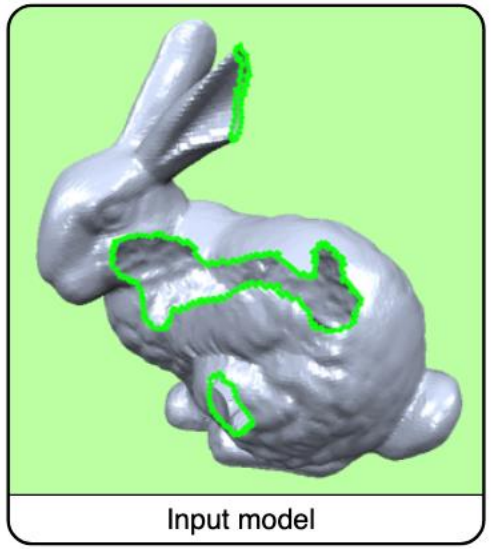
Even more challenging if topology changes

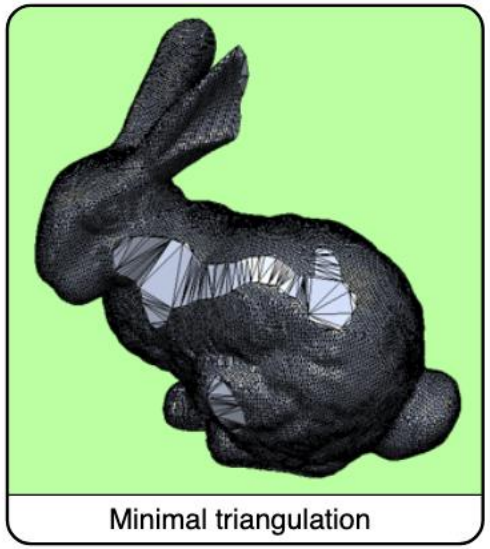


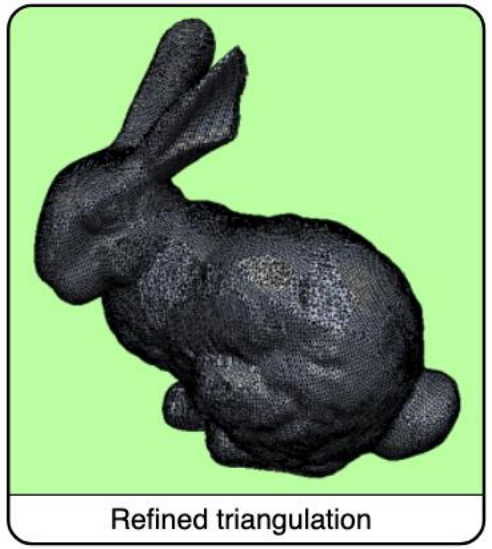
In general, designing methods that can predict **arbitrary** connectivity is challenging (e.g. image to mesh)

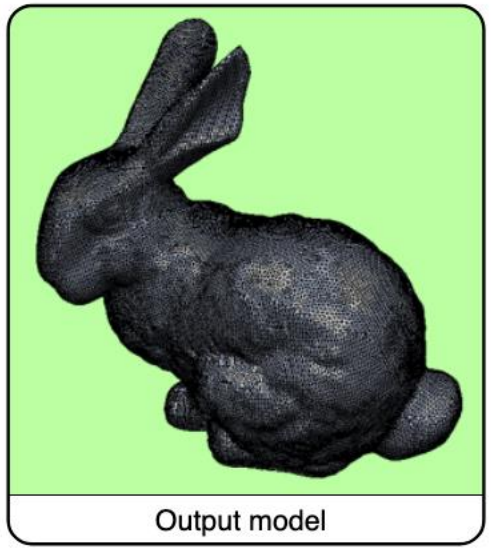
Slide credit S. Tulsiani

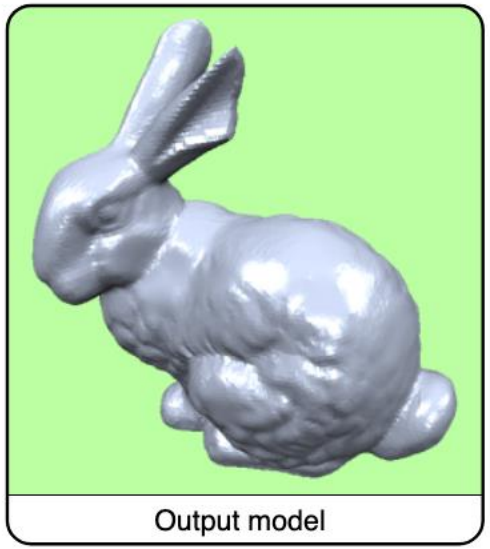
Example: Filling Holes in Meshes


Input model


Minimal triangulation

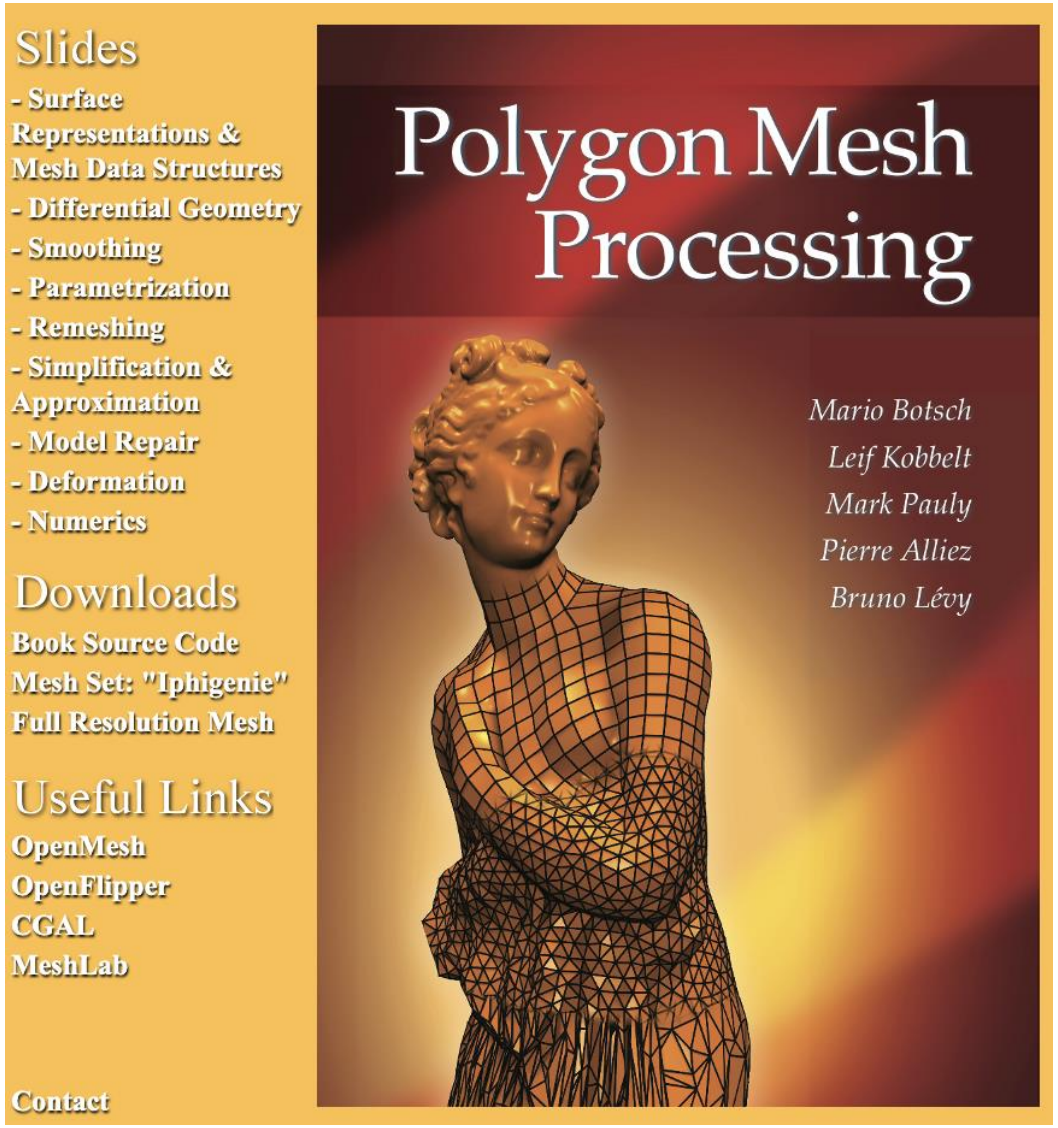

Refined triangulation


Output model


Output model

Slide credit L. Kobbelt

Mesh Processing is a Big Research Topic



Content

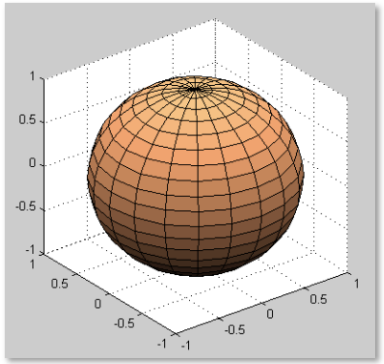
- Geometry Reconstruction
  - ➢Structure from Motion
  - ➢3D Foundation Model
- 3D Representations
  - ➢Depth Map
  - ➢Point Cloud
  - ➢Mesh
  - ➢Parametric Surface
  - ➢Signed Distance Function
  - ➢Voxels
  - ➢Others
- Conversion across Representations

Parametric Representations

- **Range of a function** $f : X \to Y, X \subseteq \mathbb{R}^m, Y \subseteq \mathbb{R}^n$

- Sphere in 3D

$$s : \mathbb{R}^2 \to \mathbb{R}^3$$



$$s(u, v) = r\left(\cos(u)\cos(v), \sin(u)\cos(v), \sin(v)\right)$$

$$(u, v) \in [0, 2\pi) \times [-\pi/2, \pi/2]$$

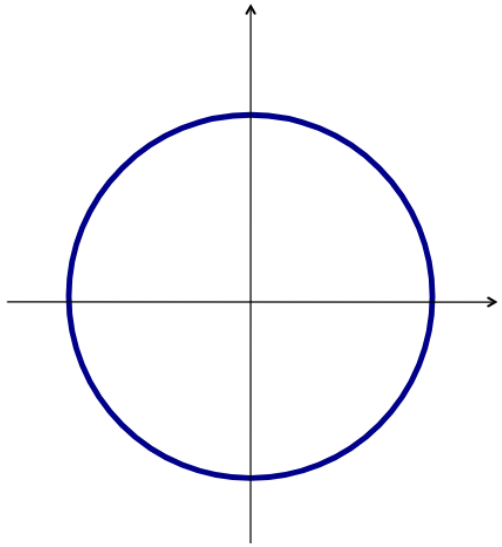
- Example: Explicit curve/circle in 2D

$$\mathbf{p} : \mathbb{R} \to \mathbb{R}^2$$

$$t \mapsto \mathbf{p}(t) = (x(t), y(t))$$

$$\mathbf{p}(t) = r\left(\cos(t), \sin(t)\right)$$

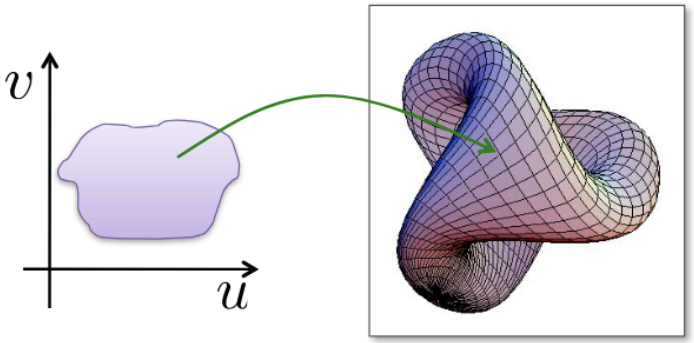
$$t \in [0, 2\pi)$$



Parametric Representations

- **Range of a function** $f : X \to Y, X \subseteq \mathbb{R}^m, Y \subseteq \mathbb{R}^n$

- **Surface in 3D:** $m = 2, n = 3$



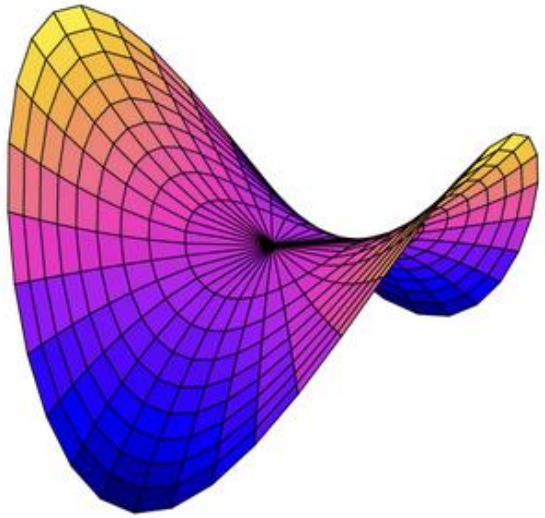
$$s(u, v) = (x(u, v), y(u, v), z(u, v))$$

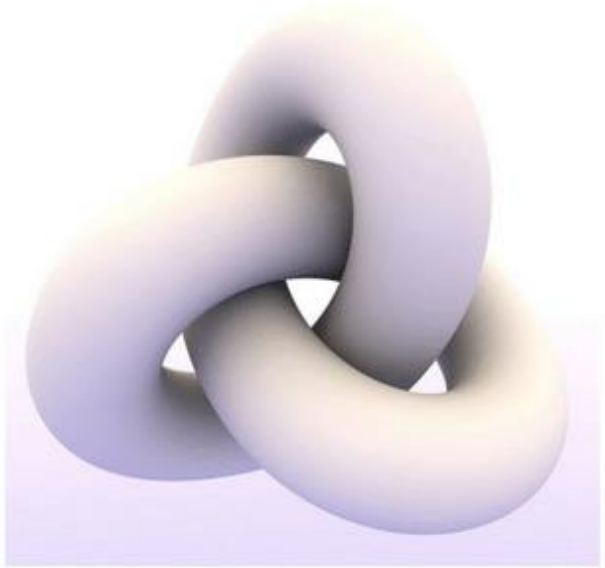
With parametric surfaces:

- Maps a low-dimensional manifold to a high-dimensional shape
- Disentangles discretization (vertices / faces) from shape representations
- Preserve the connectivity / topology of the manifold

Slide credit H. Su and S. Tulsiani

Parametric Surfaces





$$f((u,v)) = (u, v, v^2 - u^2)$$

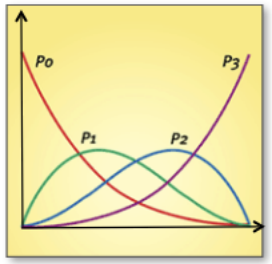
```
def trefoil(u, v):
    x = r * np.sin(3 * u) / (2 + np.cos(v))
    y = r * (np.sin(u) + 2 * np.sin(2 * u)) / (2 + np.cos(v + np.pi * 2 / 3))
    z = r / 2 * (np.cos(u) - 2 * np.cos(2 * u)) * (2 + np.cos(v)) * (2 + np.cos(v + np.pi * 2 / 3)) / 4
```

Slide credit S. Tulsiani

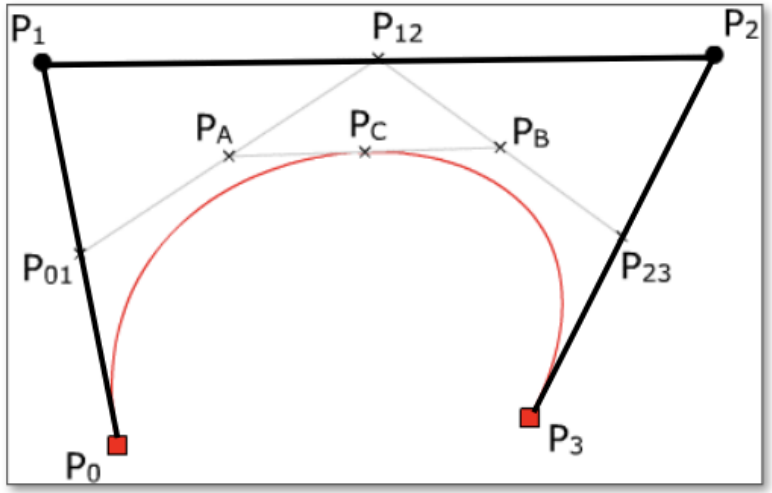
Bézier Curves

- **Bezier curves, splines**

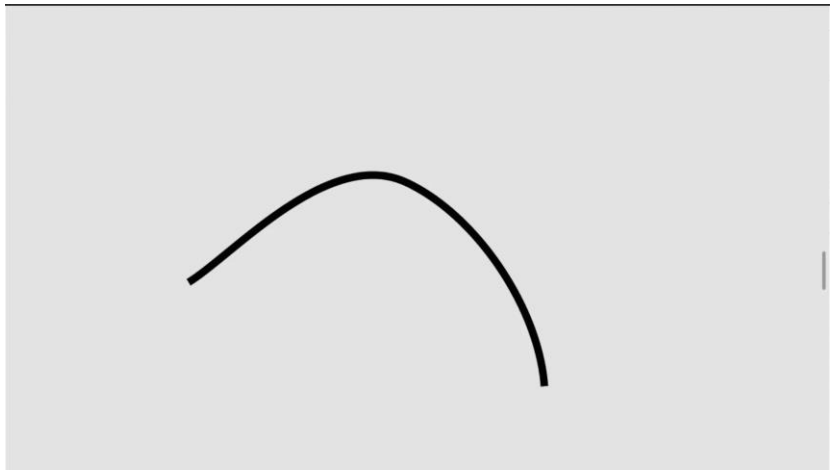
$$s(t) = \sum_{i=0}^{n} \mathbf{p}_i B_i^n(t) \qquad B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$



Basis functions



Curve and control polygon

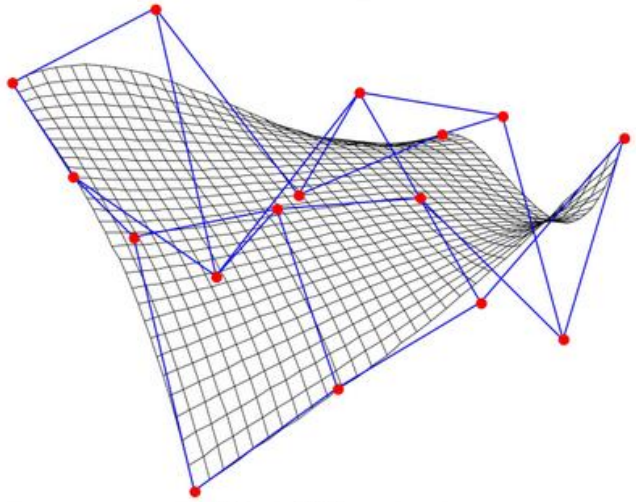


Video source: https://mmrndev.medium.com/understanding-b%C3%A9zier-curves-f6eaa0fa6c7d

Slide credit H. Su

Bézier Surfaces

Bezier Surface

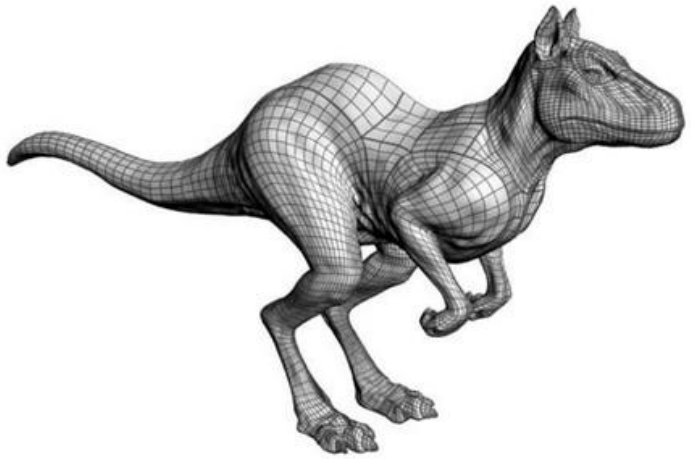


$$f((u,v)) = \sum_{i}^{N} \sum_{j}^{M} B_i^N(u)\ B_j^M(v)\ k_{i,j}$$

$$B_i^N(u) = \binom{N}{i} u^i (1-u)^{N-i}$$

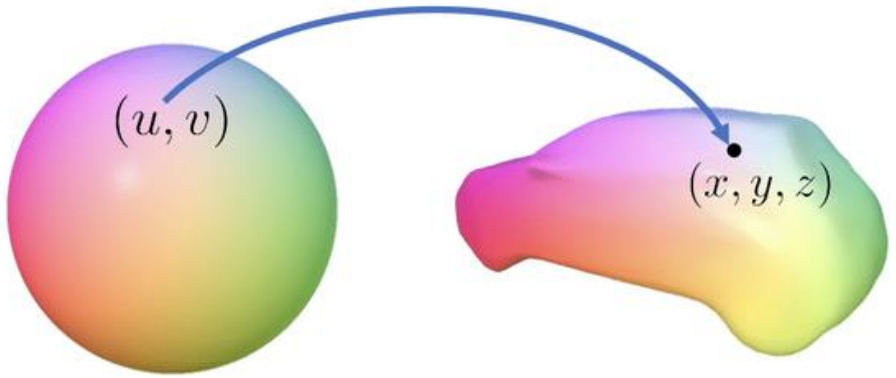
Image credits: Wikipedia

M*N control points can determine a smooth surface in 3D



Commonly used for designing meshes

Parametric Surfaces



$$f(\mathbf{u}) = \mathbf{p} \in \mathbb{R}^3;\ \mathbf{u} \in \mathcal{M}$$

- A continuous function f over a 2D manifold $\mathcal{M}$ defines a surface
- Pros:
  - Easy to sample points on surface
  - Can do computation in the domain $\mathcal{M}$
- Cons:
  - Difficult to reason about global structure (e.g., is a query point q outside the shape?)
  - Difficult to analytically render (ray-surface intersection is hard for arbitrary f)
  - Difficult to ensemble different shapes

Slide credit S. Tulsiani

Representations of 3D Models

Depth map

Image source Paul Bourke

Pointcloud

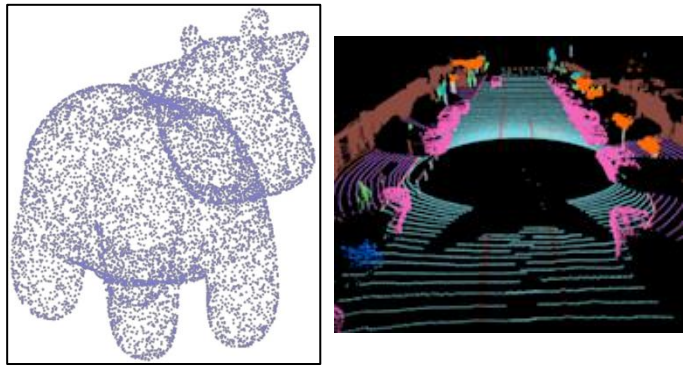


Image source S. Tulsiani Image source Waymo

Mesh

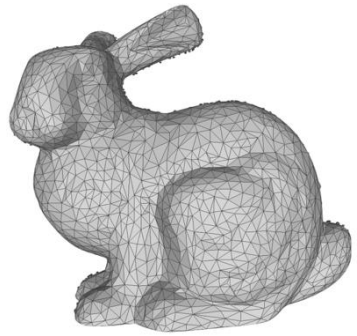


Image source S. Tulsiani

Parametric Surface

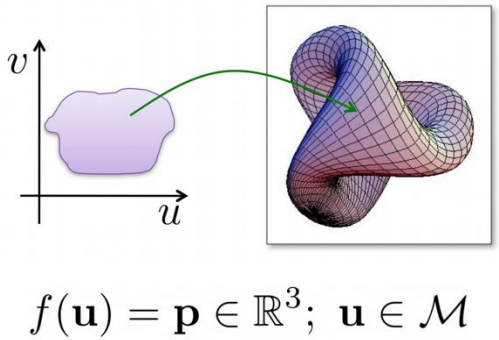


Image source S. Hao

- Depth maps, point clouds, meshes and parametric surfaces are surface representations, which describe the location of the surface, not if a point is inside / outside the shape
- Volume representations instead model the occupancy of the 3D structure, which enables inference of physical properties (e.g. mass, density, …) and facilitate physical modeling

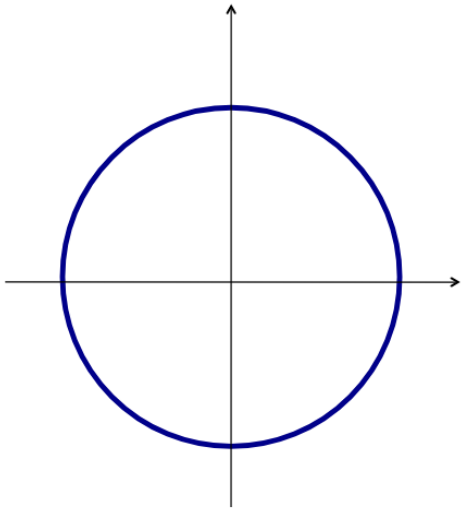
Content

- Geometry Reconstruction
  - Structure from Motion
  - 3D Foundation Model
- 3D Representations
  - Depth Map
  - Point Cloud
  - Mesh
  - Parametric Surface
  - Signed Distance Function
  - Voxels
  - Others
- Conversion across Representations

Explicit vs. Implicit Surfaces

$$\mathbf{p} : \mathbb{R} \to \mathbb{R}^2$$
$$t \mapsto \mathbf{p}(t) = (x(t), y(t))$$

$$\mathbf{p}(t) = r\,(\cos(t), \sin(t))$$
$$t \in [0, 2\pi)$$



- Define a function f
- Define the surface as:

$$\{p \mid f(p) = 0\}$$

- For example:

$$\{(x, y) | x^2 + y^2 - r^2 = 0\}$$

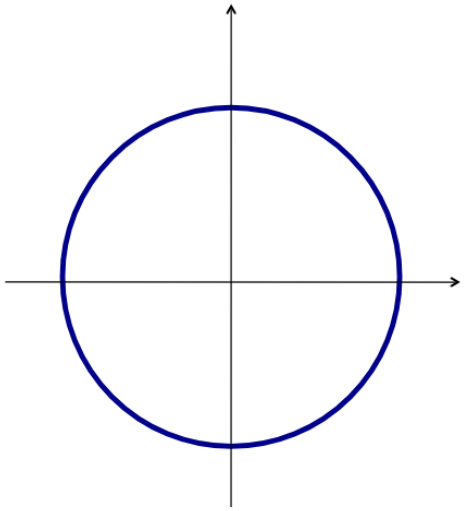
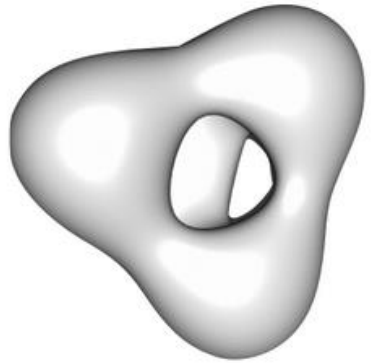


Image credit H. Su

Implicit Surfaces

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

$$f(x,y,z) = 4R^2(x^2 + y^2)$$
$$-(x^2 + y^2 + z^2 + R^2 - a^2)^2$$

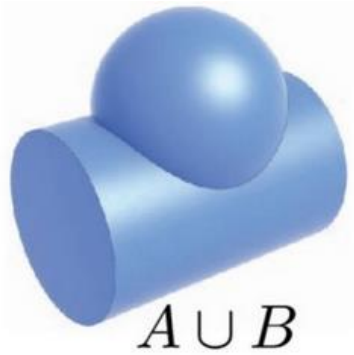


$$f(x,y,z) = x^4 + y^4 + z^4 + 2x^2y^2 + 2x^2z^2$$
$$+2y^2z^2 + 8xyz - 10x^2 - 10y^2 - 10z^2 + 20$$

Simple functions can offer concise representations for complex shapes

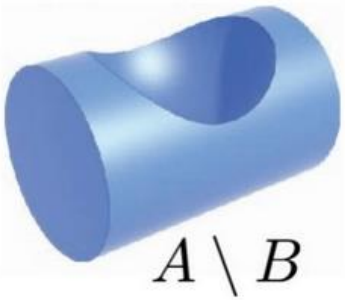
Slide credit S. Tulsiani

Assembling Shapes via Implicit Surfaces



$$\cup_i f_i(\mathbf{p}) = \min_i f_i(\mathbf{p})$$

$$\cap_i f_i(\mathbf{p}) = \max_i f_i(\mathbf{p})$$



$$(f - g)(\mathbf{p}) = \max(f(\mathbf{p}), -g(\mathbf{p}))$$

Boolean operations are easy

(max and min depend on convention — is +ve outside or inside? Assumed outside here.)

Image credits: Keenan Crane

Extend Implicit Surfaces to Volume Representations

- Define a function f
- Define the surface as:

$$\{p \mid f(p) = 0\}$$

- For example:

$$\{(x, y) | x^2 + y^2 - r^2 = 0\}$$
$$\{(x, y) | x^2 + y^2 - r^2 = 4\}$$
$$\{(x, y) | x^2 + y^2 - r^2 = -1\}$$
$$\{(x, y) | x^2 + y^2 - r^2 = -4\}$$

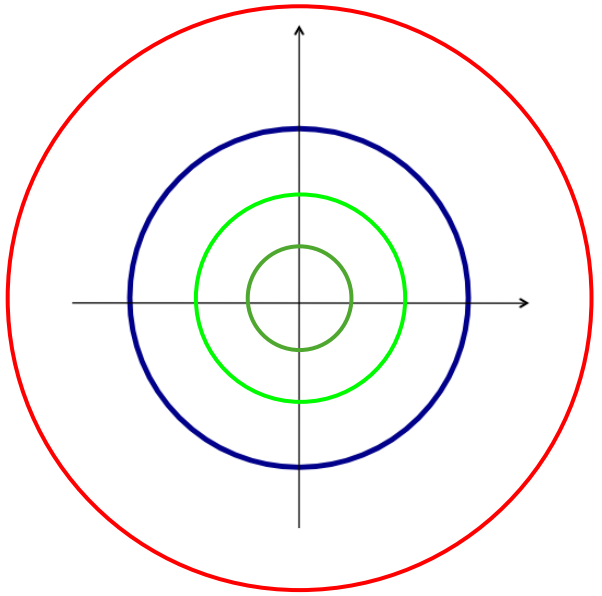


Image credit H. Su

Extend Implicit Surfaces to Volume Representations

- Define a function f
- Define the surface as:

$$\{p \mid f(p) = 0\}$$

- For example:

$$\{(x, y) | x^2 + y^2 - r^2 = 0\}$$
$$\{(x, y) | x^2 + y^2 - r^2 = 4\}$$
$$\{(x, y) | x^2 + y^2 - r^2 = -1\}$$
$$\{(x, y) | x^2 + y^2 - r^2 = -4\}$$

- For $f(p) = d$, we know can infer the distance of point p to the surface is d. The sign of d denotes if the point is inside / outside the shape.

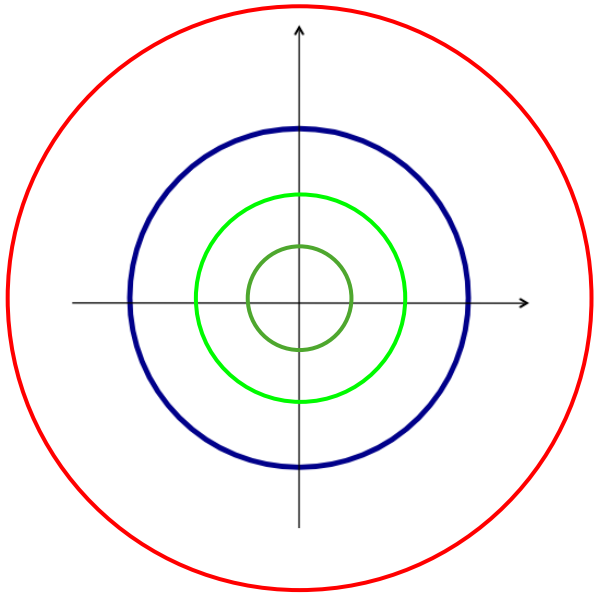
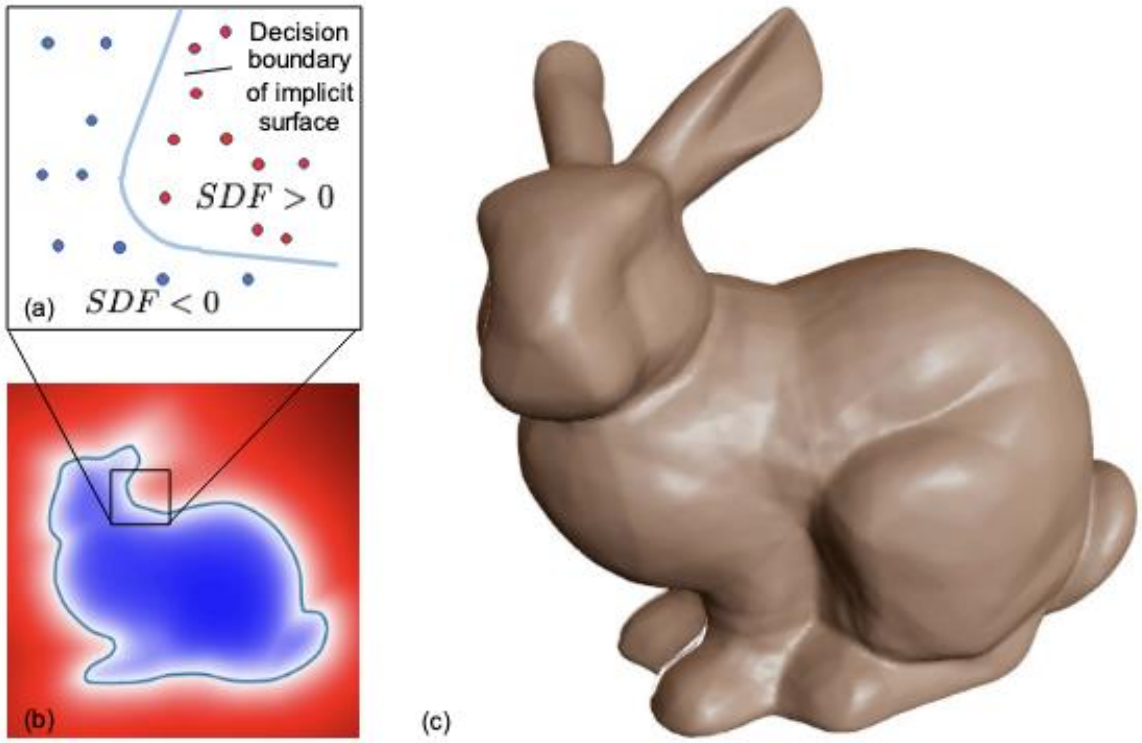
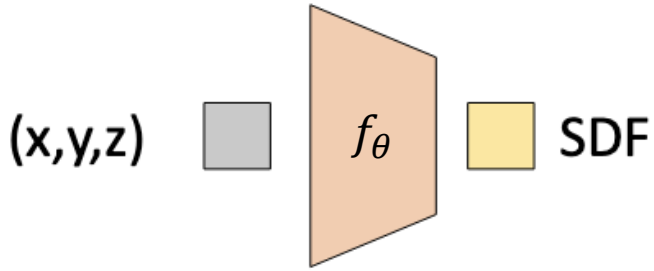


Image credit H. Su

Deep Signed Distance Function

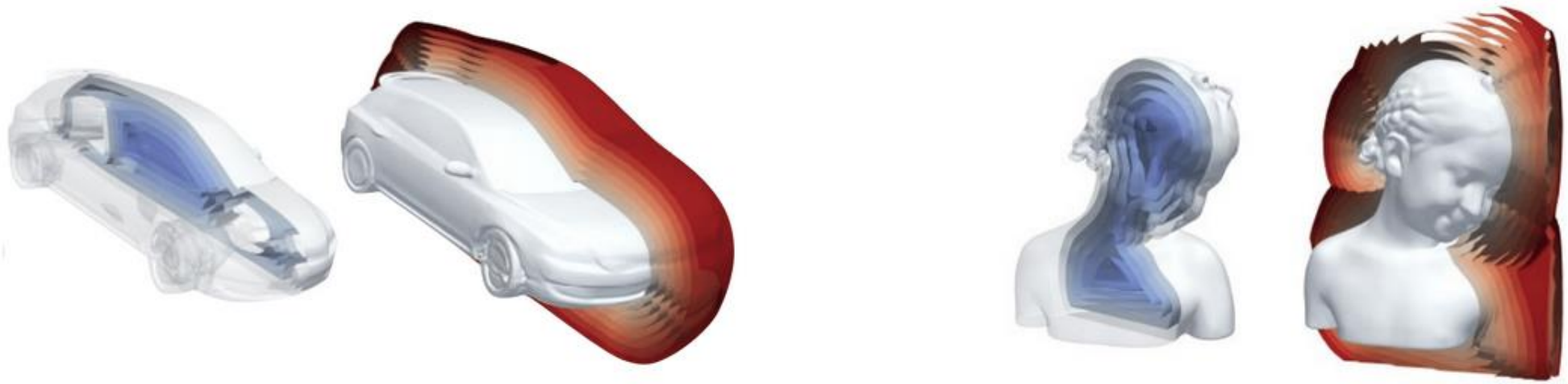


- Parameterize a function f_θ as a neural network with parameters θ
- The surface is: $\{p \mid f_\theta(p) = 0\}$
- Learn the neural work to predict the signed distance given any point p



DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. Park et al.

Learning Signed Distance Functions to Capture Complex Shapes

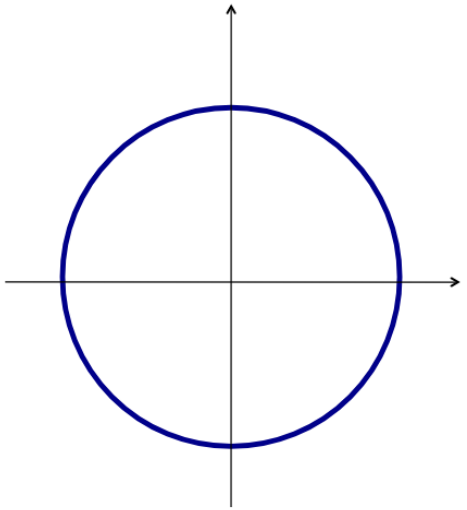


$$\{\mathbf{p} \mid f_\theta(\mathbf{p}) = 0\}$$

Neural network with parameters θ

Image credits: IGR: Implicit Geometric Regularization for Learning Shapes.Gropp et. al

Explicit Surface vs. Implicit Surface vs. Density Fields

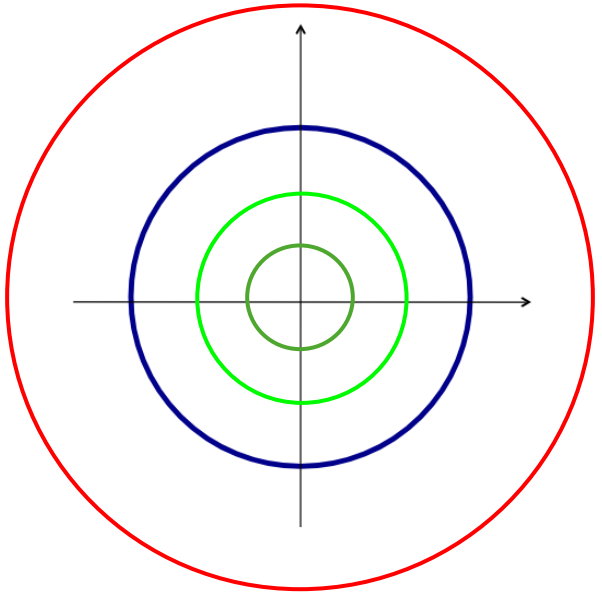


$\mathbf{p} : \mathbb{R} \to \mathbb{R}^2$

$t \mapsto \mathbf{p}(t) = (x(t), y(t))$

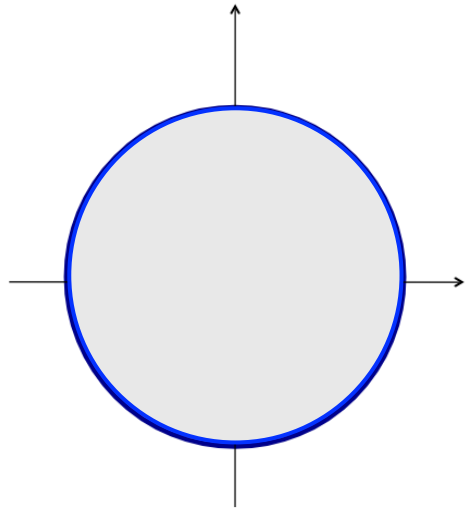
$\mathbf{p}(t) = r\,(\cos(t), \sin(t))$

$t \in [0, 2\pi)$



- Define a function f
- Define the surface as:

$$\{p \mid f(p) = 0\}$$



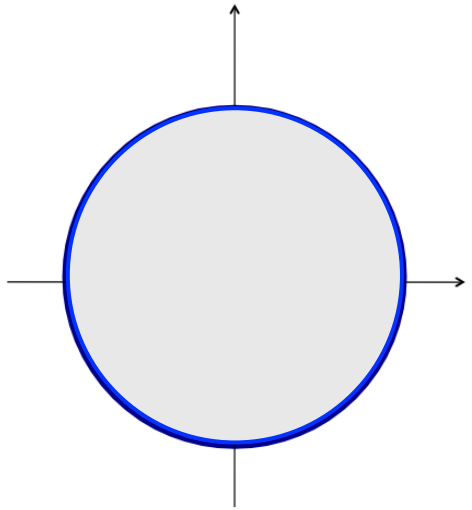
- Define a function f
- The set of internal points:

$$\{p \mid f(p) = 1\}$$

- The set of external points:

$$\{p \mid f(p) = 0\}$$

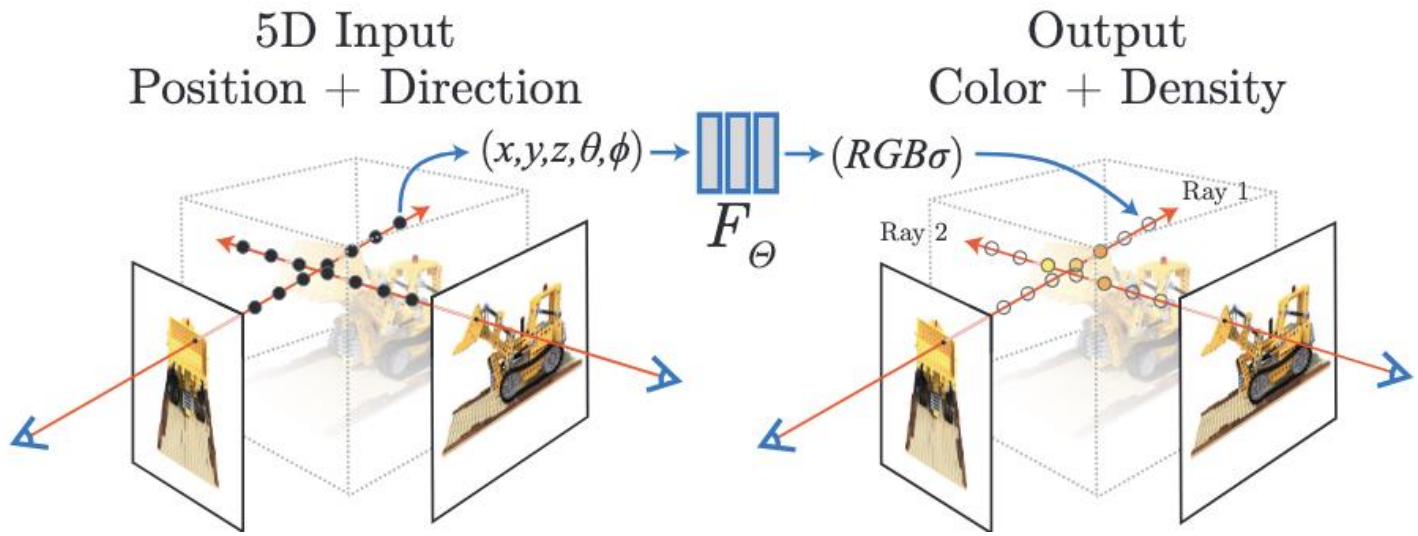
Parameterizing Fields with Neural Networks



- Parametrize a function f_θ by a neural network with parameters θ
- Predict the occupancy of any point p :

$$f_\theta(p) = \begin{cases} 1, & \text{p is inside the shape} \\ 0, & \text{p is outside the shape} \end{cases}$$

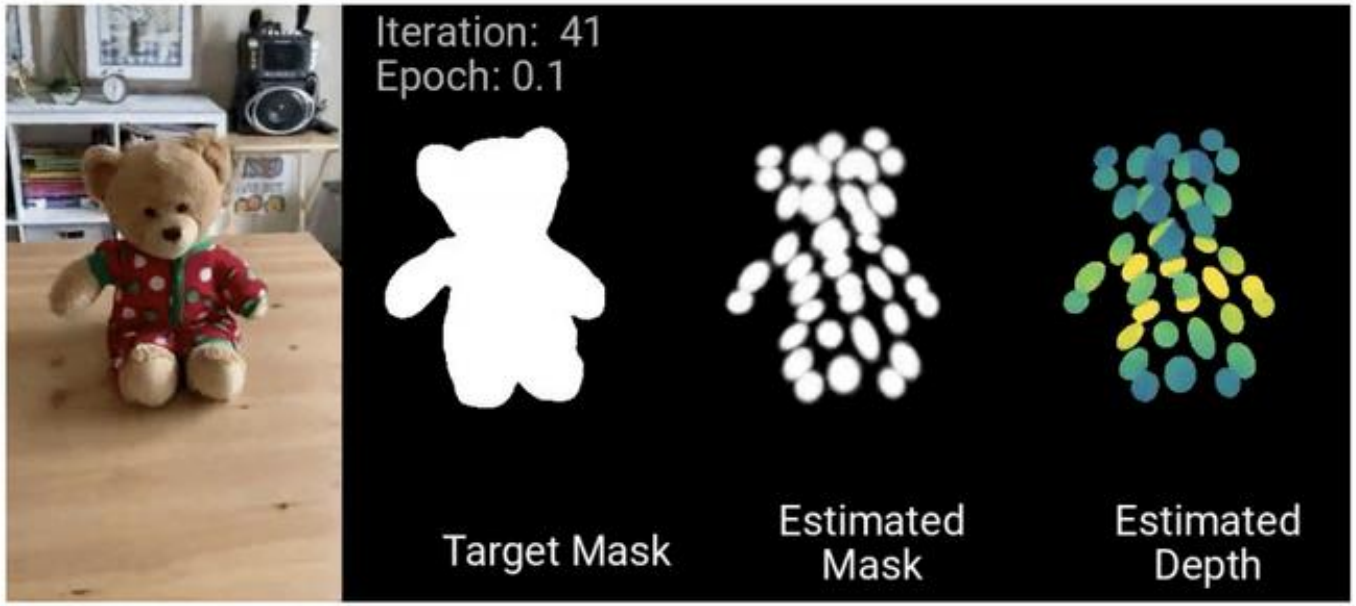
Neural Radiance Field



We'll cover in the next lecture

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. Mildenhall et al.

Parameterizing Fields with Primitive Shapes



We'll cover in the next lecture

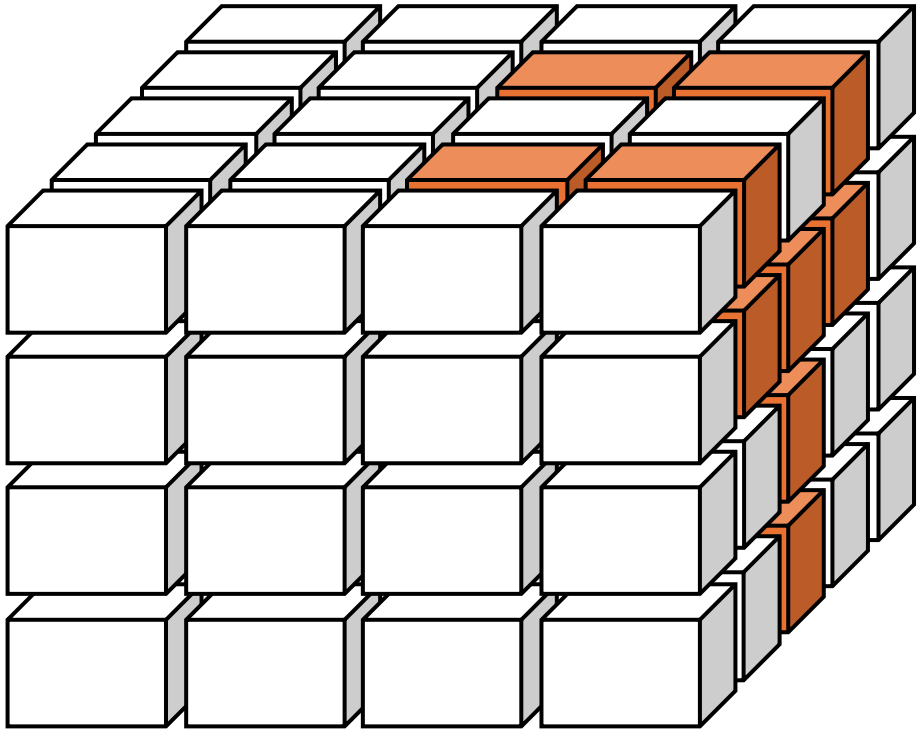
Slide credit S. Tulsiani

Image credits: Lef: Local Deep Implicit Functions for 3D Shape, Genova et. al. Right: Fuzzy Metaballs, Keselman & Hebert

Content

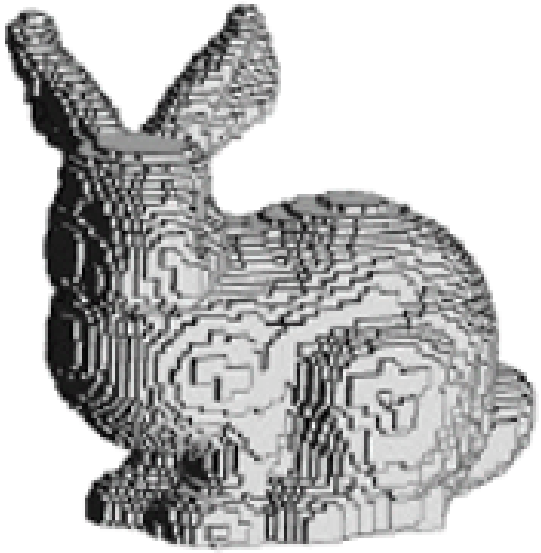
- Geometry Reconstruction
  - Structure from Motion
  - 3D Foundation Model
- 3D Representations
  - Depth Map
  - Point Cloud
  - Mesh
  - Parametric Surface
  - Signed Distance Function
  - Voxels
  - Others
- Conversion across Representations

Parameterizing Fields with Voxel Grids

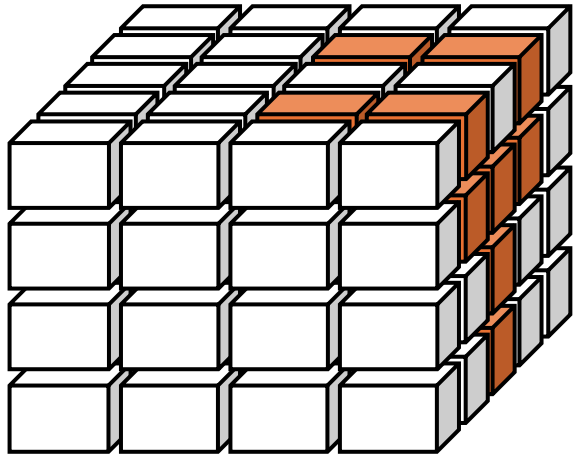


- Given a discrete voxel map $V \in \mathbb{R}^{H \times W \times D}$
- Predict the occupancy of any point p :

$$V(p) = \begin{cases} 1, & \text{p is inside the shape} \\ 0, & \text{p is outside the shape} \end{cases}$$



Pros and Cons of Voxel Grids



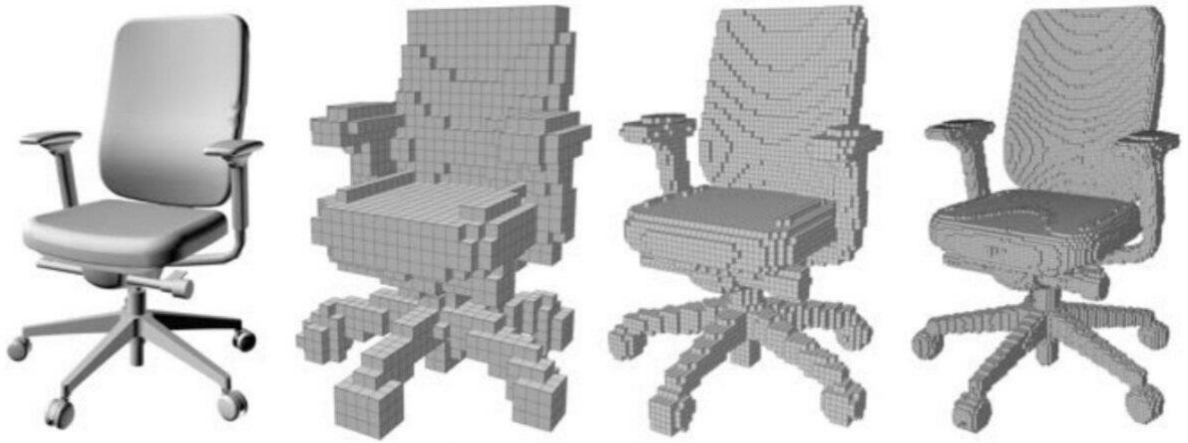
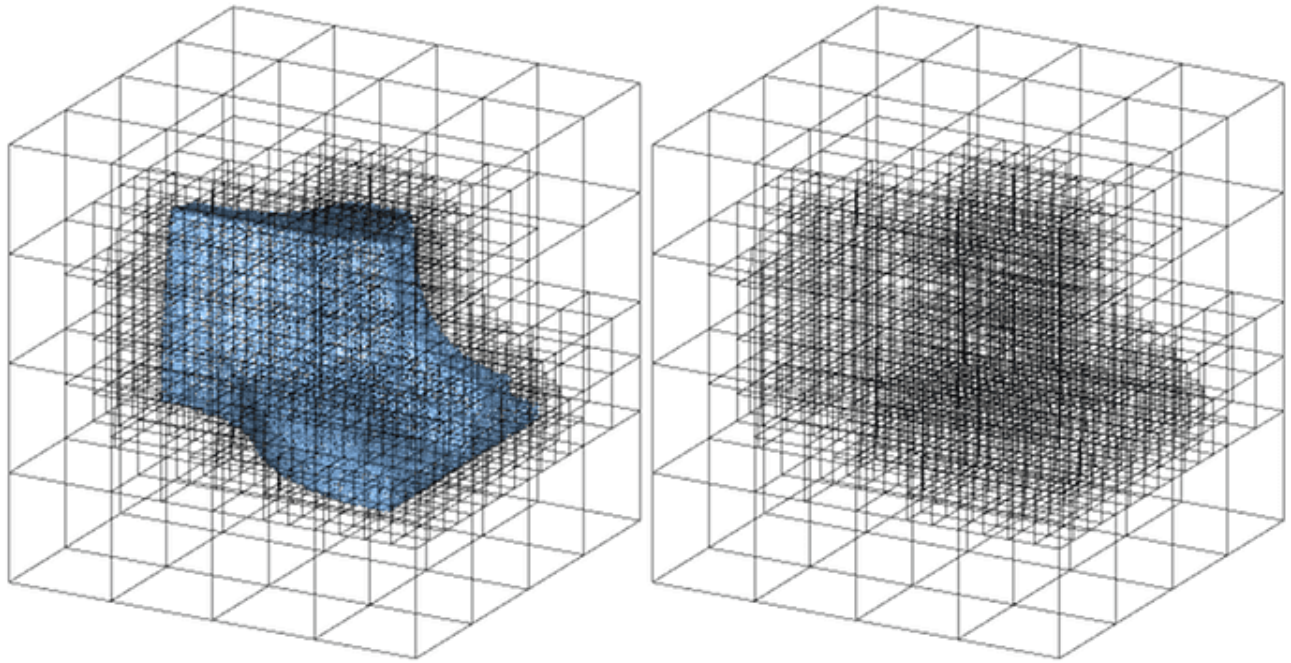


Image source https://www.antoinetlc.com/blog-summary/3d-data-representations

- Pros:
  - Fast sampling
  - Convenient processing as it exposes locality. Can easily run convolutions, nearest-neighbor lookups, etc.
- Cons:
  - Requires huge memory as dimension and resolution increase
  - Requires high resolution to capture finer details of the shape

Slide adapted from V. Sitzmann

Octrees: Multi-Resolution Voxel Grids



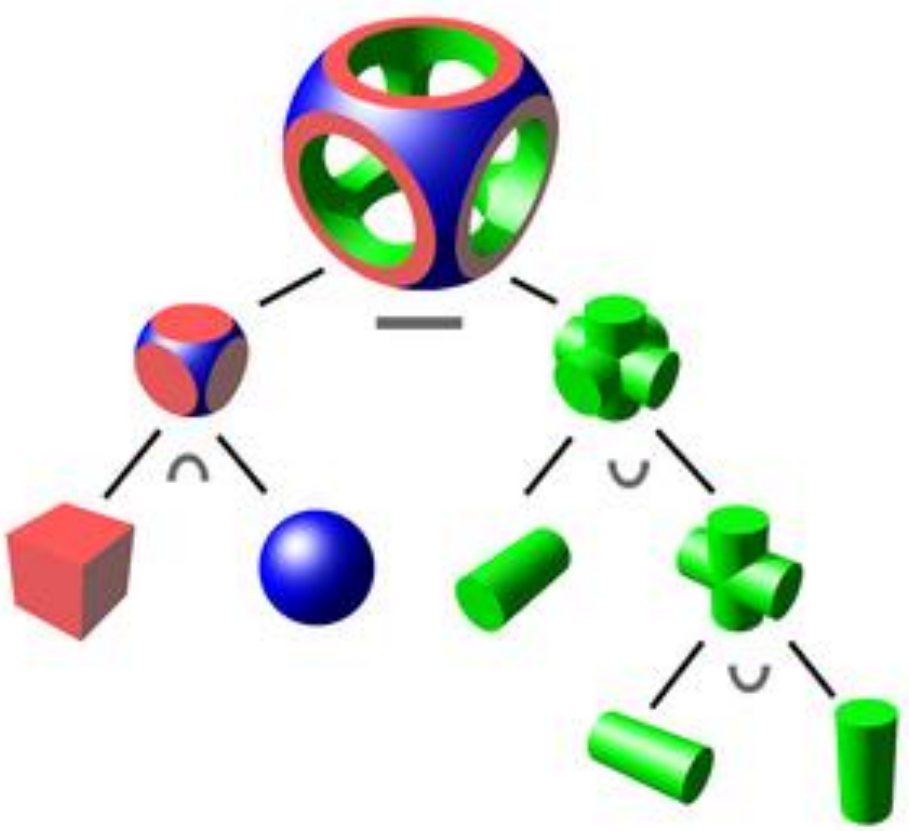
- Divide volume into 8 voxels. If any box has more than N points in it, divide it into 8 voxels
- Pros:
  - Saves memory (less resolution in empty parts)
- Cons:
  - Sub-division is not differentiable: can't easily build neural network that outputs Octree
  - To sample point, have to traverse multi-resolution hierarchy.

Slide credit V. Sitzmann

Content

- Geometry Reconstruction
  - Structure from Motion
  - 3D Foundation Model
- 3D Representations
  - Depth Map
  - Point Cloud
  - Mesh
  - Parametric Surface
  - Signed Distance Function
  - Voxels
  - Others
- Conversion across Representations

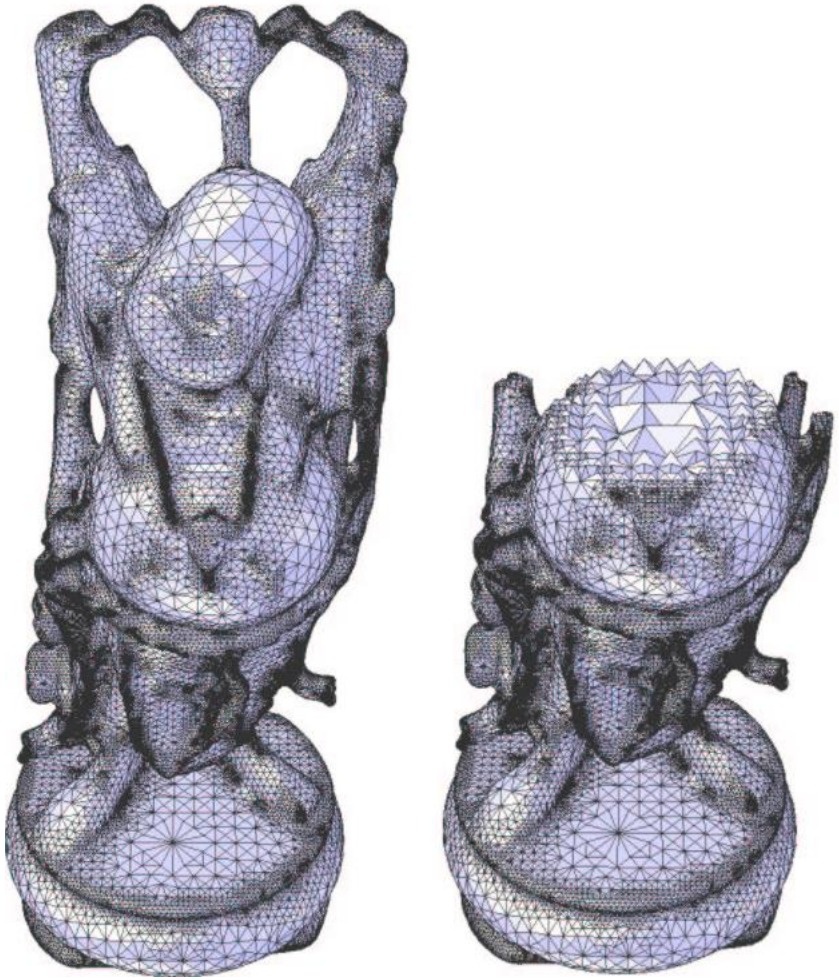
Constructive Solid Geometry

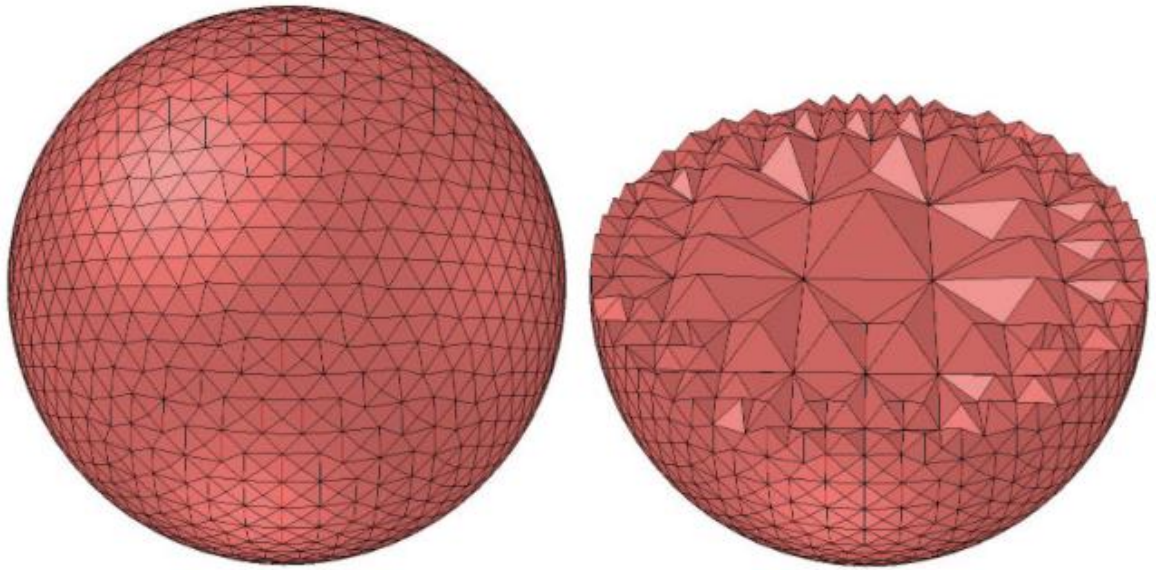


- Boolean ops over geometric primitives (spheres, boxes, cylinders, cones, …)
- Often non-unique

Slide credit L. Guibas

Tetrahedral Meshes





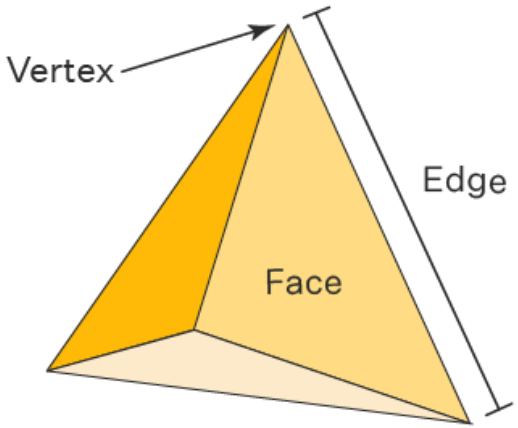
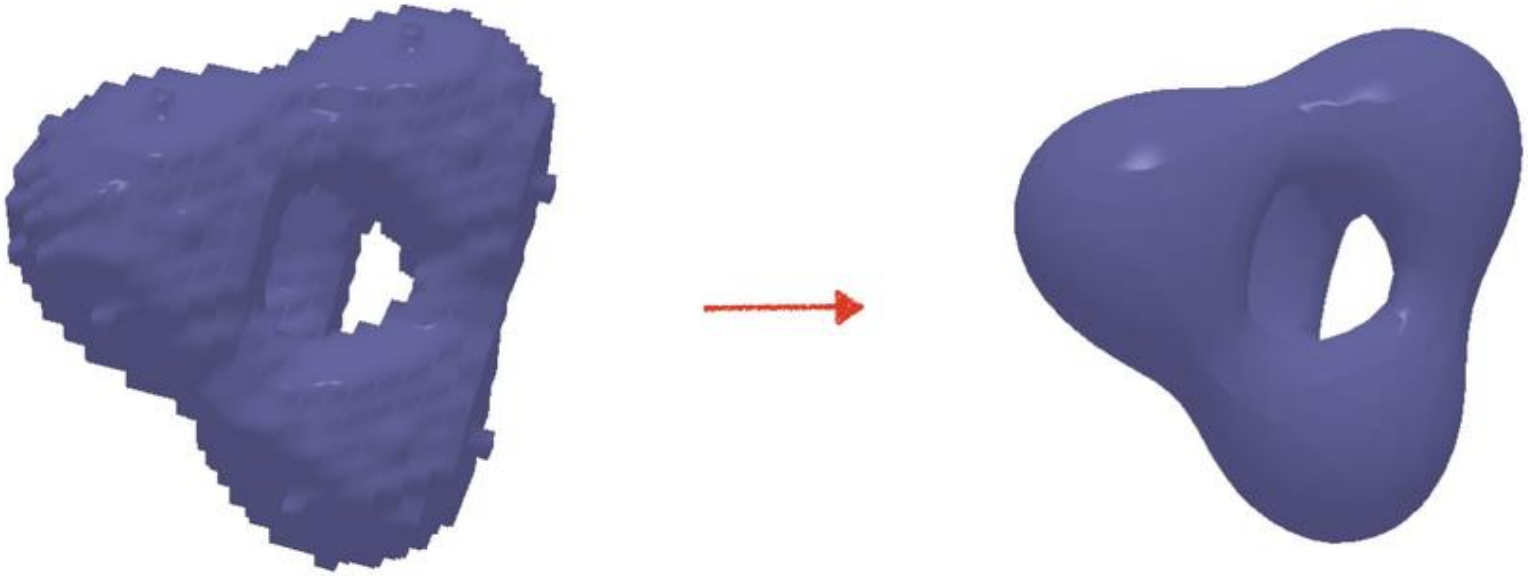


Image source https://www.cuemath.com/geometry/tetrahedron/

Content

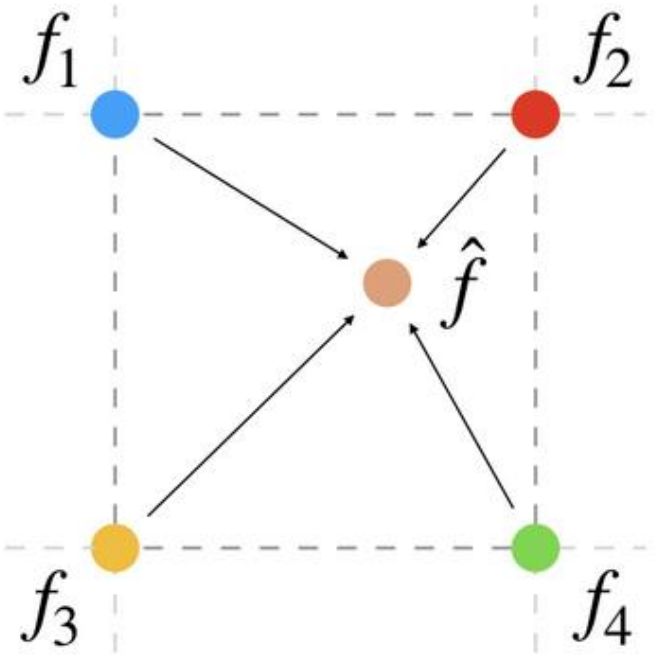
- Geometry Reconstruction
  - Structure from Motion
  - 3D Foundation Model
- 3D Representations
  - Depth Map
  - Point Cloud
  - Mesh
  - Parametric Surface
  - Signed Distance Function
  - Voxels
  - Others
- Conversion across Representations

Conversion Across Representations: From Discrete to Continuous



Given values in a discrete (regular) grid, can we predict **f(p)** for arbitrary points?

Interpolation: Query Discrete Map with Continuous Values



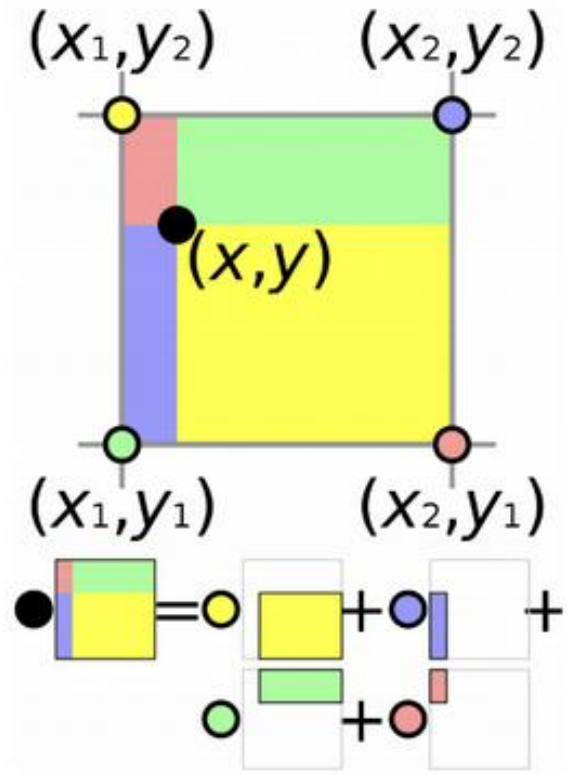
Interpolation in a 2D voxel grid.

- Only stores values at vertex locations, i.e. corners
- Values at intermediate coordinates are defined via interpolation w/ some kernel k

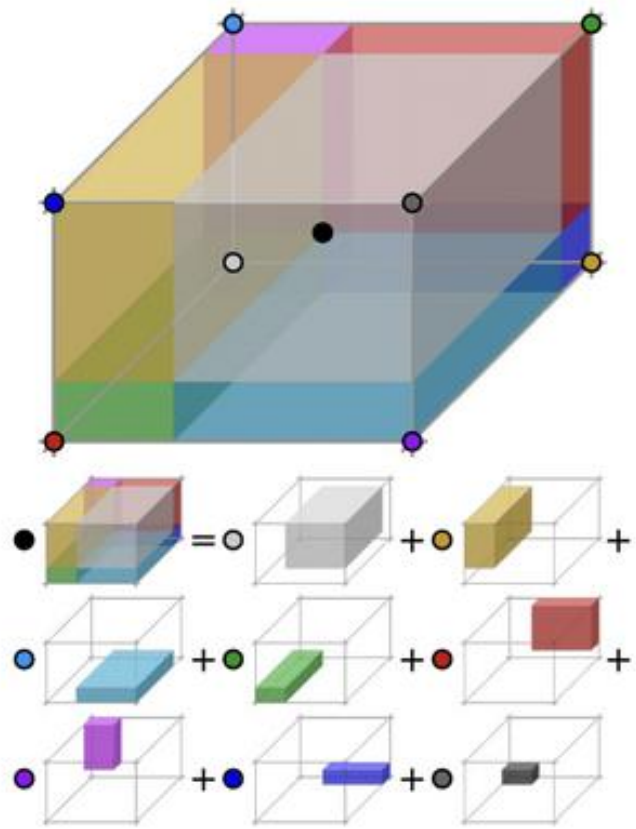
$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{N} f_i k(\mathbf{x}, \mathbf{x}_i)$$

Slide credit V. Sitzmann

Bilinear vs. Trilinear Interpolation



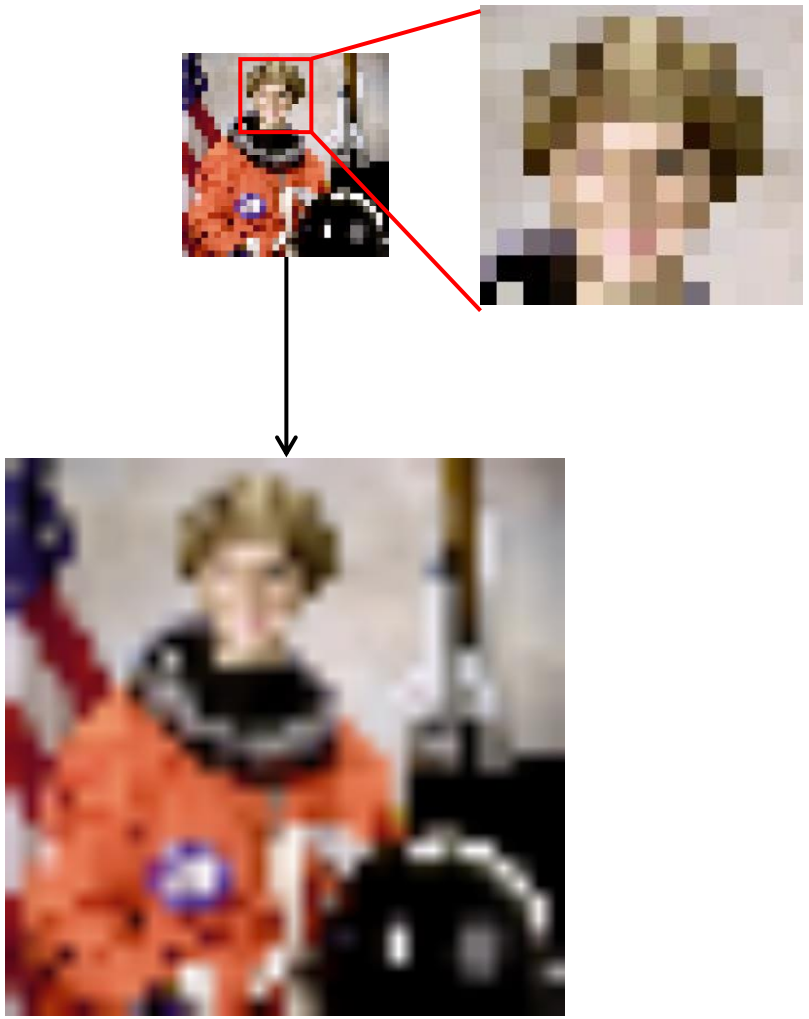
Bilinear Interpolation



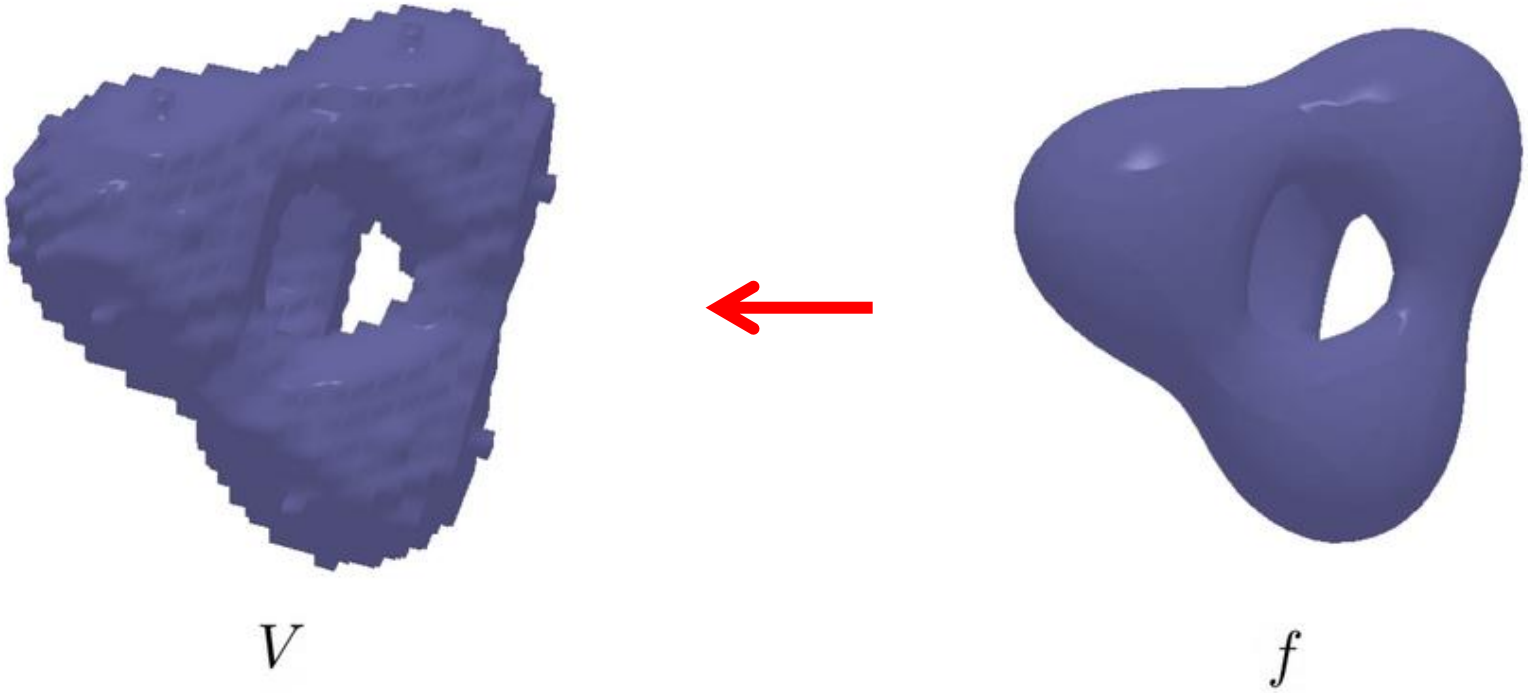
Trilinear Interpolation

Slide credit S. Tulsiani

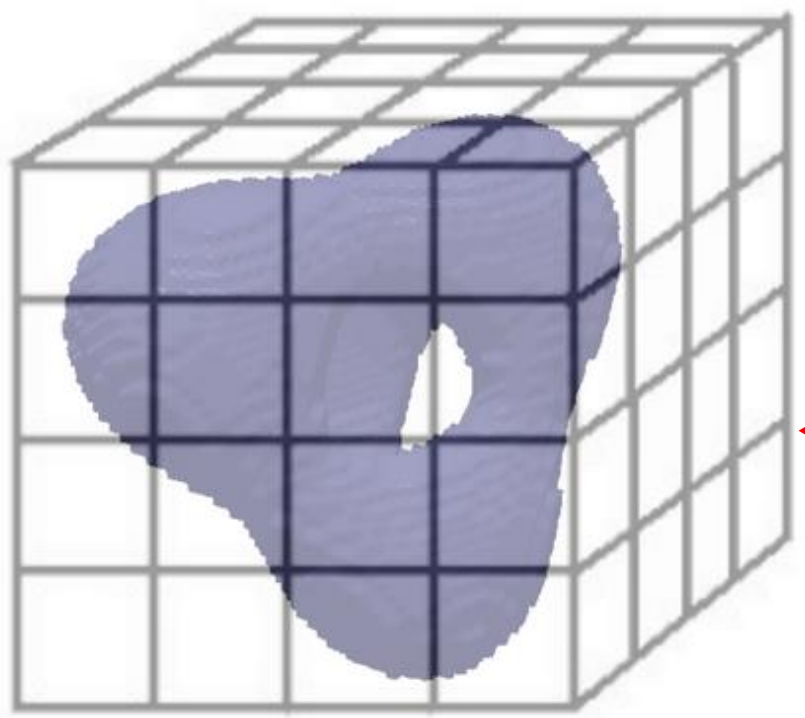
Accuracy Improves with Grid Resolution

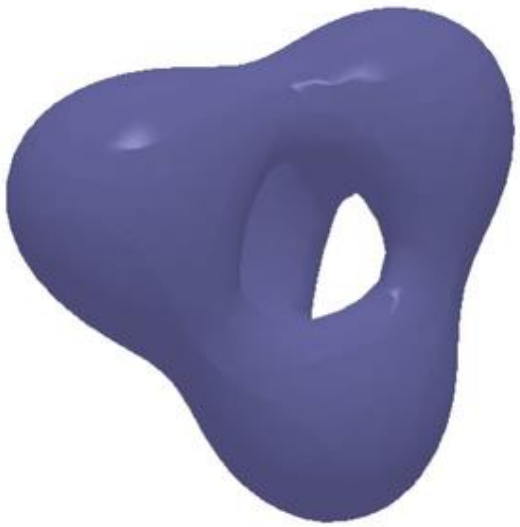


Conversion Across Representations: From Continuous to Discrete



Conversion Across Representations: From Continuous to Discrete





V f $\{\mathbf{p} \mid f(\mathbf{p}) = 0\}$

$V[\mathbf{p}] \in [0, 1]$

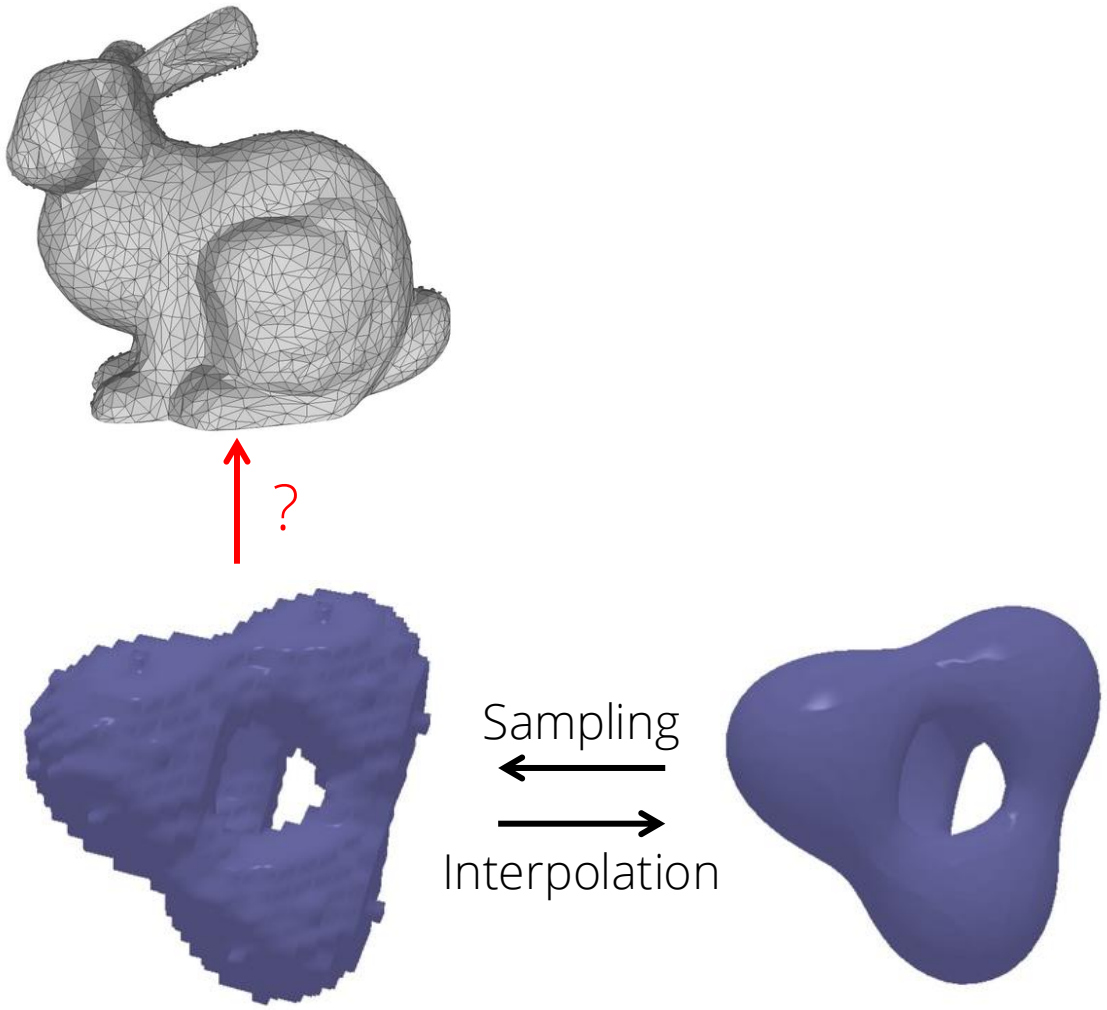
$\mathbf{p} \in \text{grid } \mathcal{G}$

Sample and evaluate points in the grid

$$V[\mathbf{p}] = sgn(f(\mathbf{p})) \quad \text{or} \quad V[\mathbf{p}] = sigmoid(f(\mathbf{p}))$$

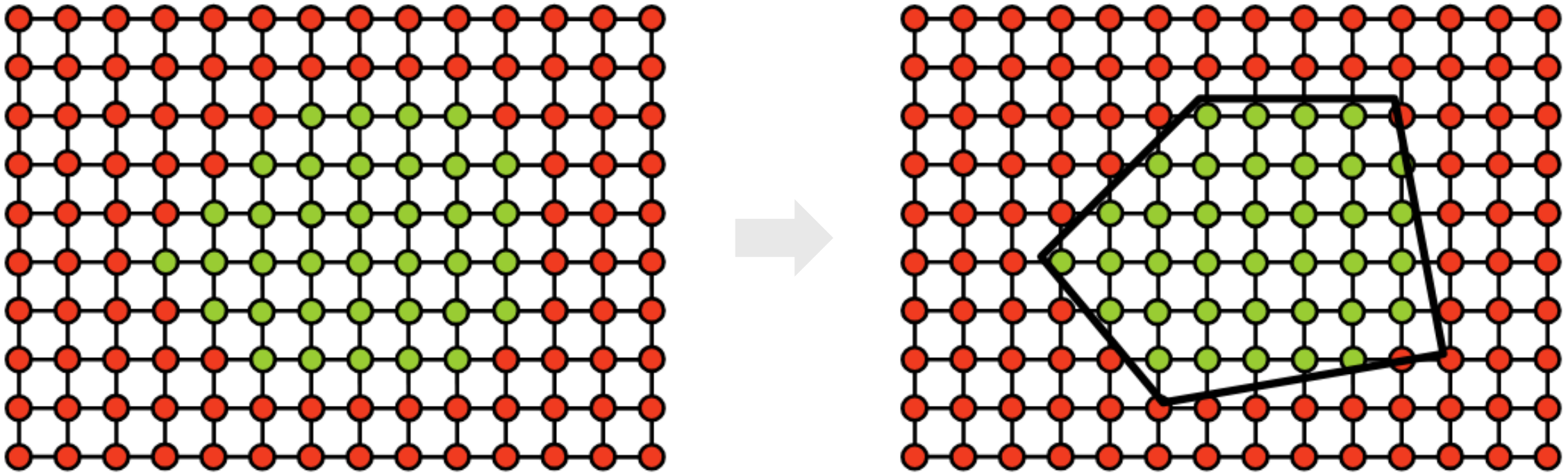
Slide credit S. Tulsiani

Conversion Across Representations



Slide credit S. Tulsiani

Conversion Across Representations: From Voxel to Meshes



- Find the boundary from a discrete map

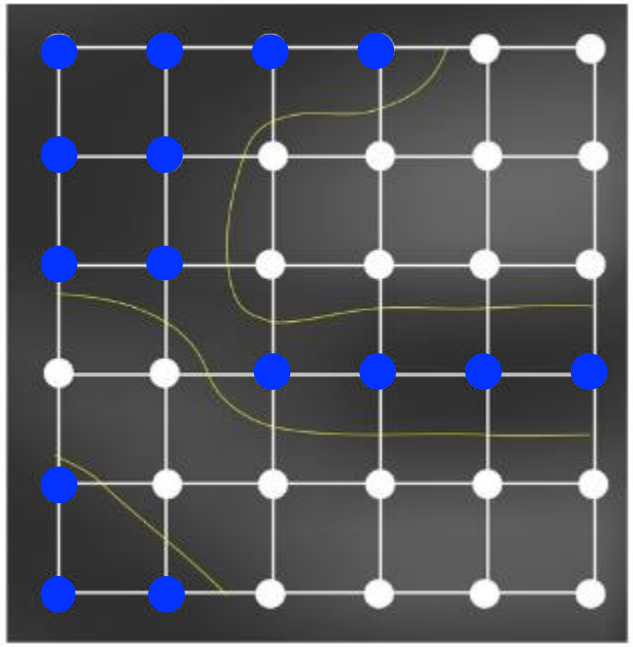
Image credit L. Guibas

Marching Squares in 2D

- Given a function $f(p)$:

$$f(p) = \begin{cases} 1, & \text{inside} \\ 0, & \text{outside} \end{cases}$$

- Steps:
  - Evaluate $f(p)$ on a grid
  - Classify grid points
  - Classify grid edges



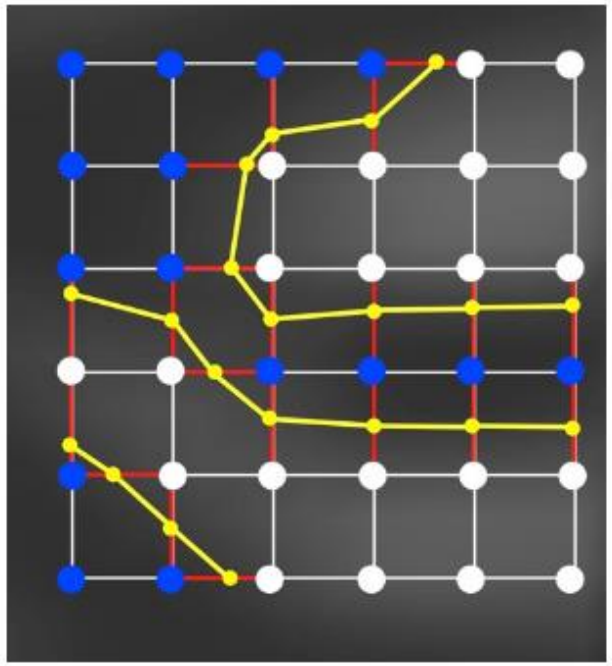
Slide credit L. Guibas

Marching Squares in 2D

- Given a function $f(p)$:

$$f(p) = \begin{cases} 1, & \text{inside} \\ 0, & \text{outside} \end{cases}$$

- Steps:
  - Evaluate $f(p)$ on a grid
  - Classify grid points
  - Classify grid edges
  - Compute intersections
  - Connect intersections



Slide credit L. Guibas

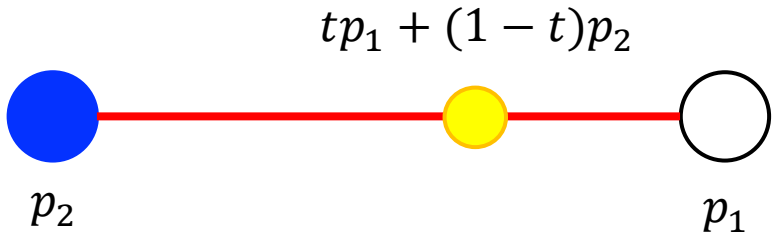
Compute Intersections for Marching Squares

- Edges with a sign switch contain intersections

$$f(p_1) < 0, f(p_2) > 0$$
$$\Rightarrow f(tp_1 + (1-t)p_2) = 0, \exists t \in [0,1]$$

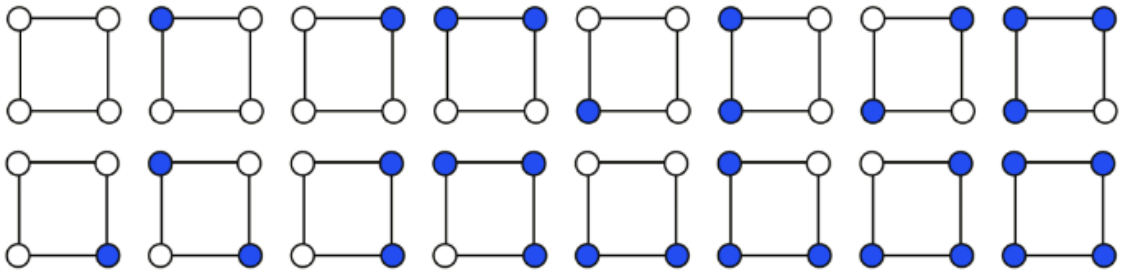
- The simplest way to compute t: assume f is linear between p_1 and p_2:

$$t = \frac{f(p_1)}{f(p_2) - f(p_1)}$$



Slide credit L. Guibas

Connect Intersections for Marching Squares



16 cases

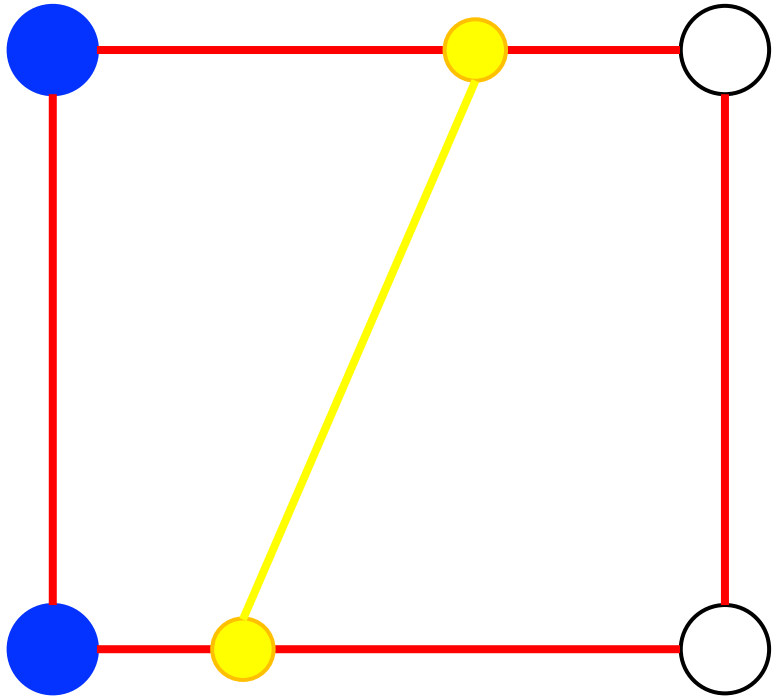
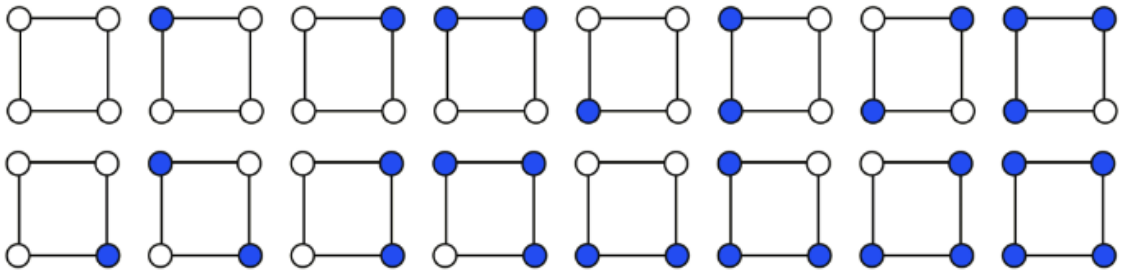
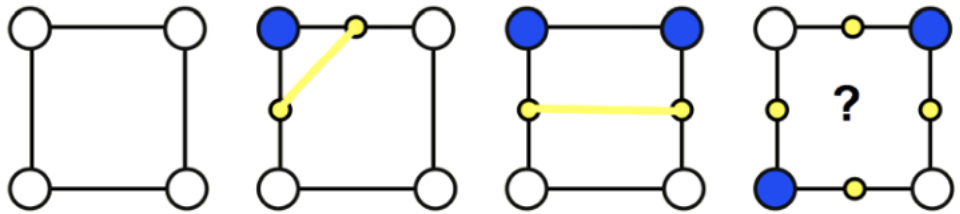


Image credit L. Guibas

Connect Intersections for Marching Squares



16 cases



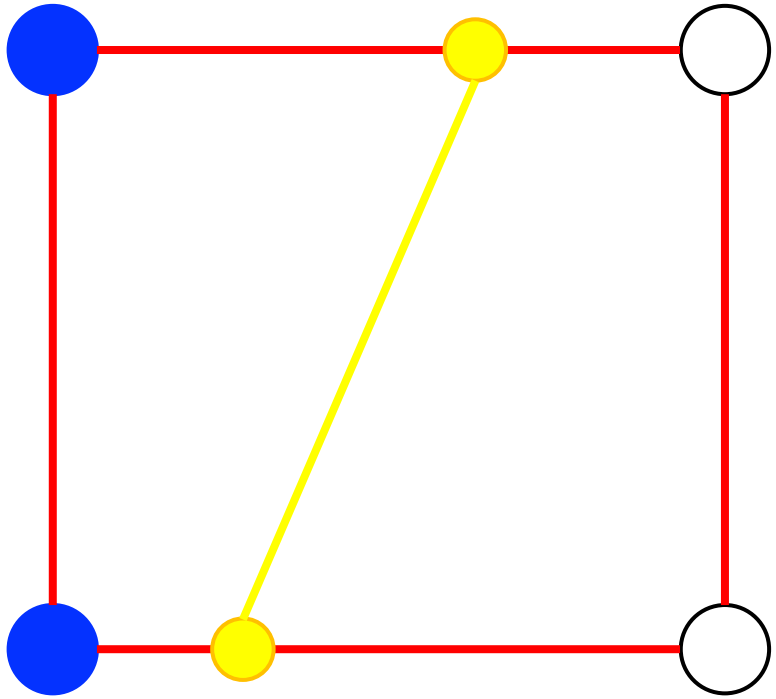
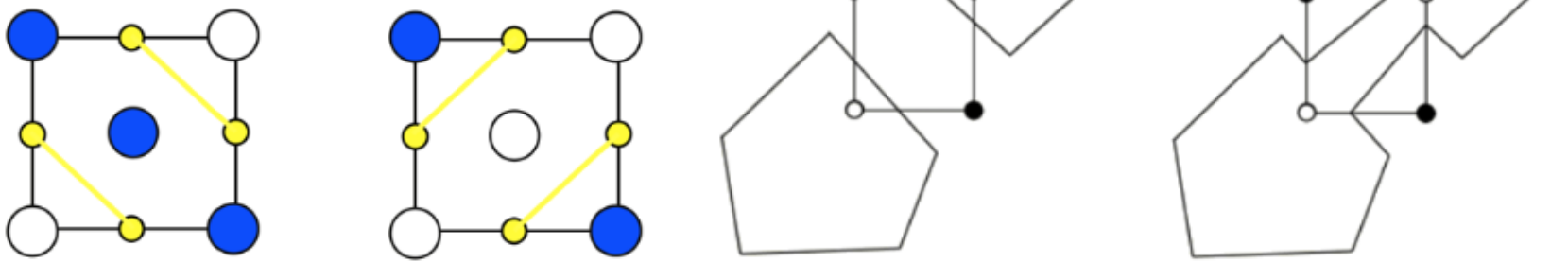


Image credit L. Guibas

Marching Squares in 2D

Connecting the intersections:

Ambiguous cases:



Two options:

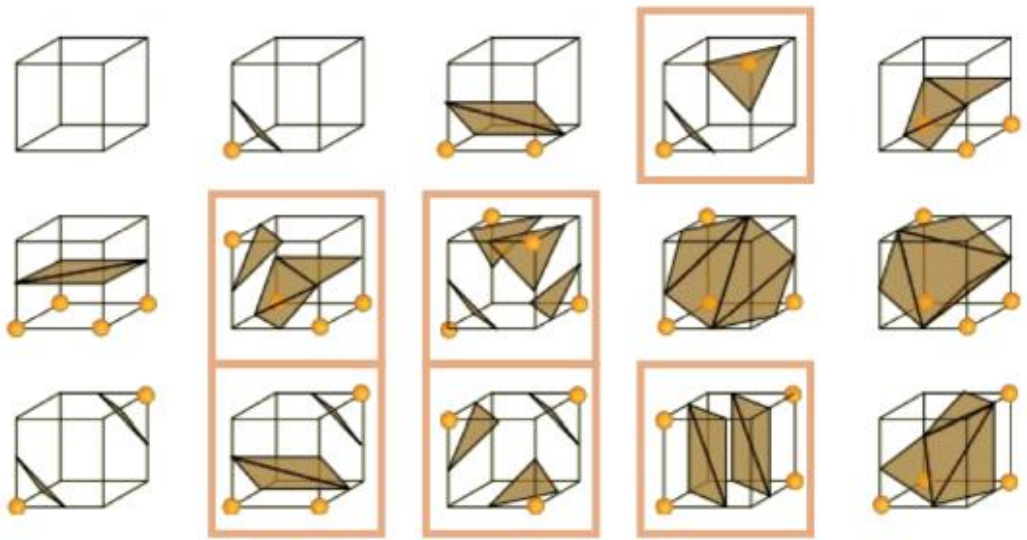
1) Can resolve ambiguity by subsampling inside the cell.
2) If subsampling is impossible, pick one of the two possibilities.

Slide credit L. Guibas

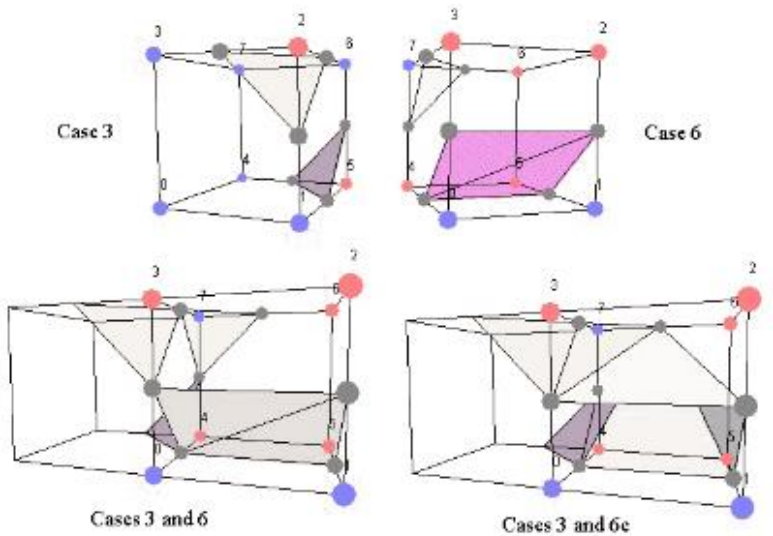
Marching Cubes in 3D

Same machinery: cells → **cubes** (voxels), lines → triangles

- 256 different cases - 15 after symmetries, 6 ambiguous cases
- More subsampling rules → 33 unique cases



the 15 cases



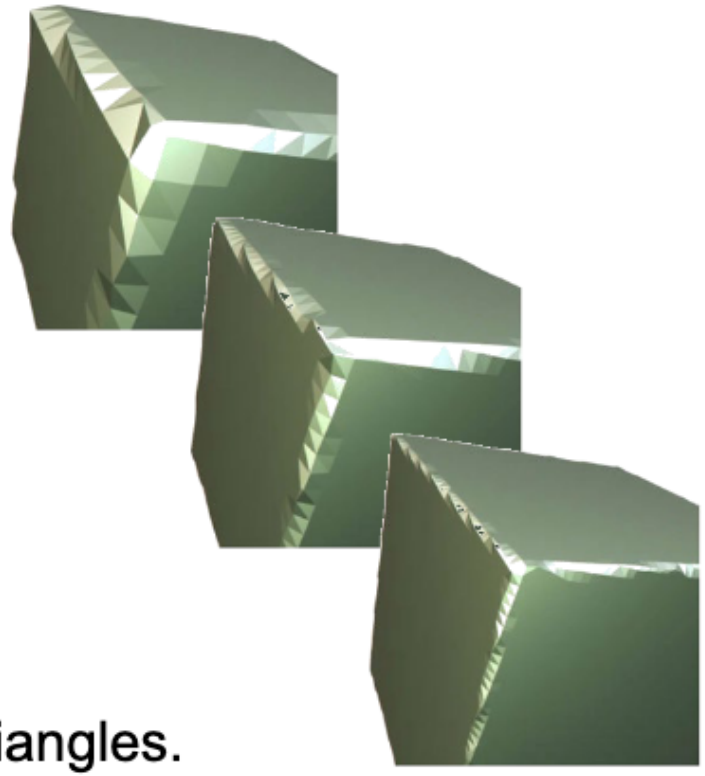
explore ambiguity to avoid holes!

Chernyaev, Marching Cubes 33,'95

Marching Cubes in 3D

Main Strengths:

- Very multi-purpose.
- Extremely fast and parallelizable.
- Relatively simple to implement.
- Virtually parameter-free

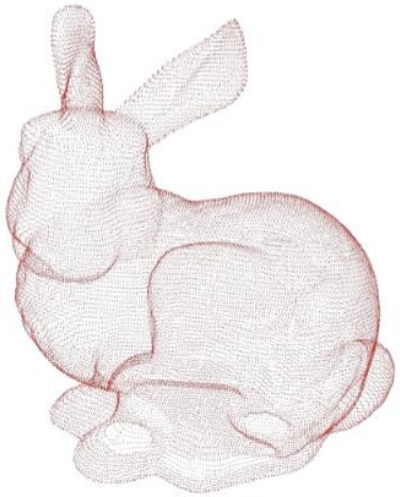


Main Weaknesses:

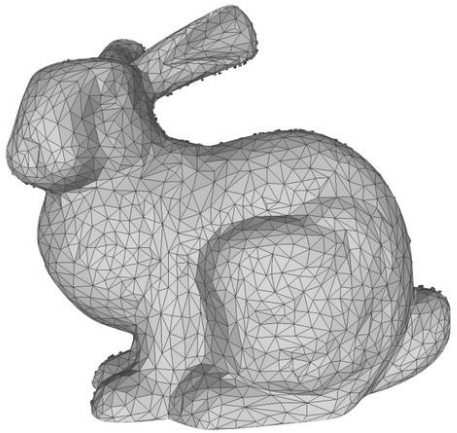
- Can create badly shaped (skinny) triangles.
- Many special cases (implemented as big lookup tables).
- No sharp features.

Slide credit L. Guibas

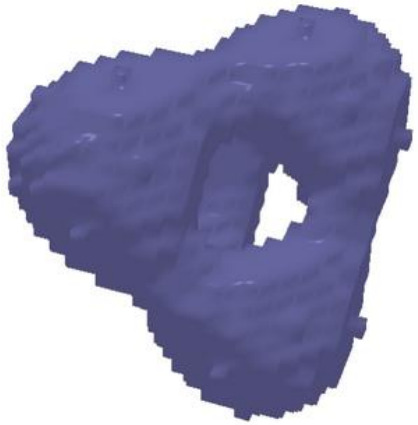
Conversion Across Representations



?

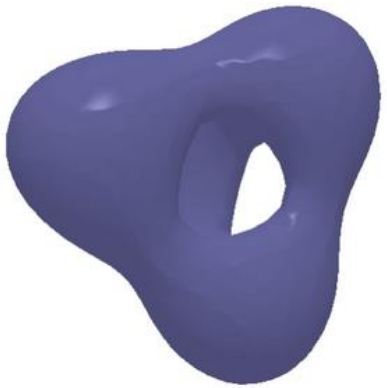


Marching Cubes

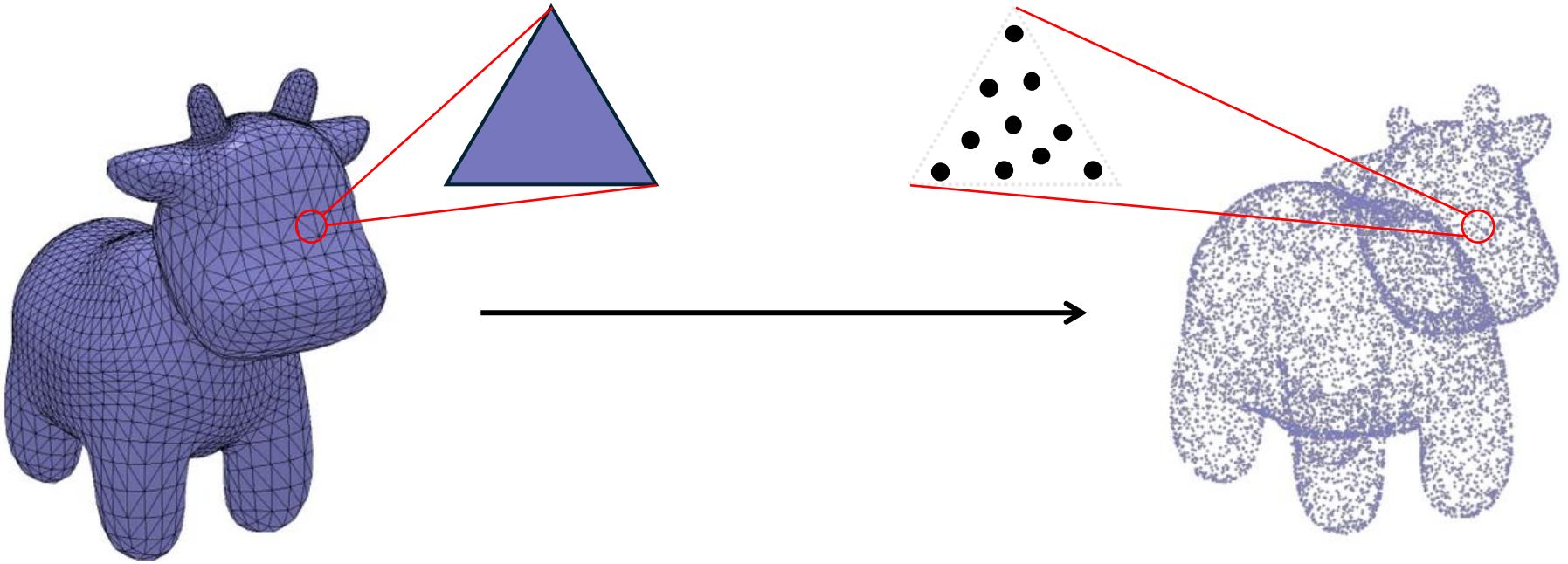


Sampling

Interpolation



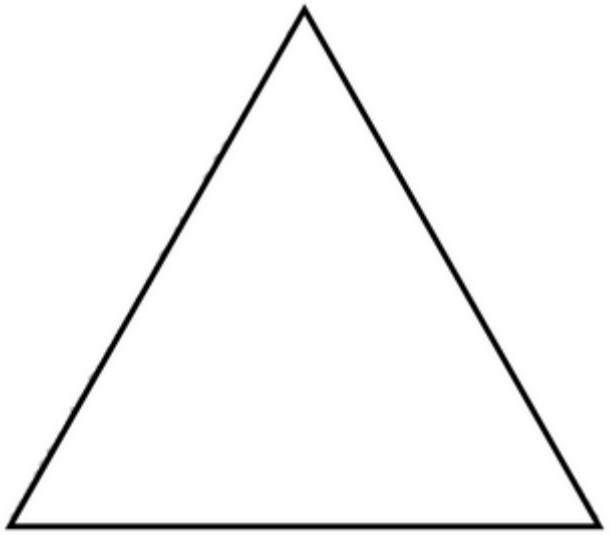
Conversion Across Representations: From Meshes to Point Clouds



Converting meshes to point clouds is simply sample points for a triangle

Image credit S. Tulsiani

Sampling Points from Triangles



- The simplest approach

$$\mathbf{v} = \alpha_1 \mathbf{p}_1 + \alpha_2 \mathbf{p}_2 + \alpha_3 \mathbf{p}_3$$

$$\sum_i \alpha_i = 1 \; ; \; \alpha_i > 0$$

Initial attempt:

$$\alpha_1 \sim [0, 1]$$

$$\alpha_2 \sim [0, 1 - \alpha_1]$$

$$\alpha_3 = 1 - \alpha_1 - \alpha_2$$

Slide credit S. Tulsiani

Sampling Points from Triangles

- The simplest approach

$$\mathbf{v} = \alpha_1 \mathbf{p}_1 + \alpha_2 \mathbf{p}_2 + \alpha_3 \mathbf{p}_3$$

$$\sum_i \alpha_i = 1 \; ; \; \alpha_i > 0$$

Initial attempt:

$$\alpha_1 \sim [0, 1]$$

$$\alpha_2 \sim [0, 1 - \alpha_1]$$

$$\alpha_3 = 1 - \alpha_1 - \alpha_2$$

$\alpha_3 < \alpha_2 < \alpha_1$: samples more points around p_1

Slide credit S. Tulsiani

Sampling Points from Triangles

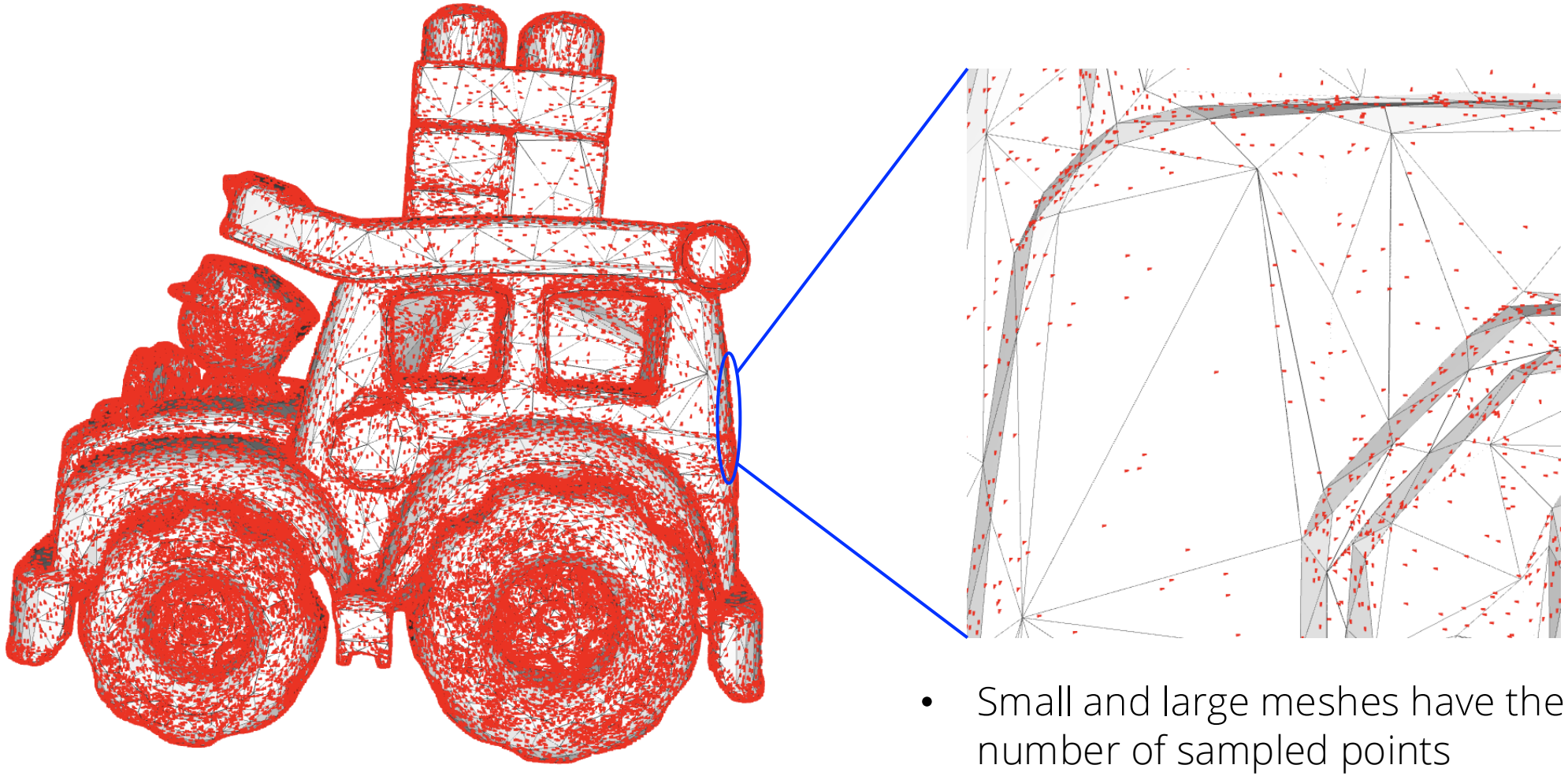
- A better approach

$$\mathbf{v} = \alpha_1 \mathbf{p}_1 + \alpha_2 \mathbf{p}_2 + \alpha_3 \mathbf{p}_3$$

$$\mathrm{u}, \mathrm{v} \sim [0, 1]$$
$$\alpha_1 = 1 - \sqrt{\mathrm{u}}$$
$$\alpha_2 = (1 - \mathrm{v})\sqrt{\mathrm{u}}$$
$$\alpha_3 = \mathrm{v}\sqrt{\mathrm{u}}$$

Slide credit S. Tulsiani

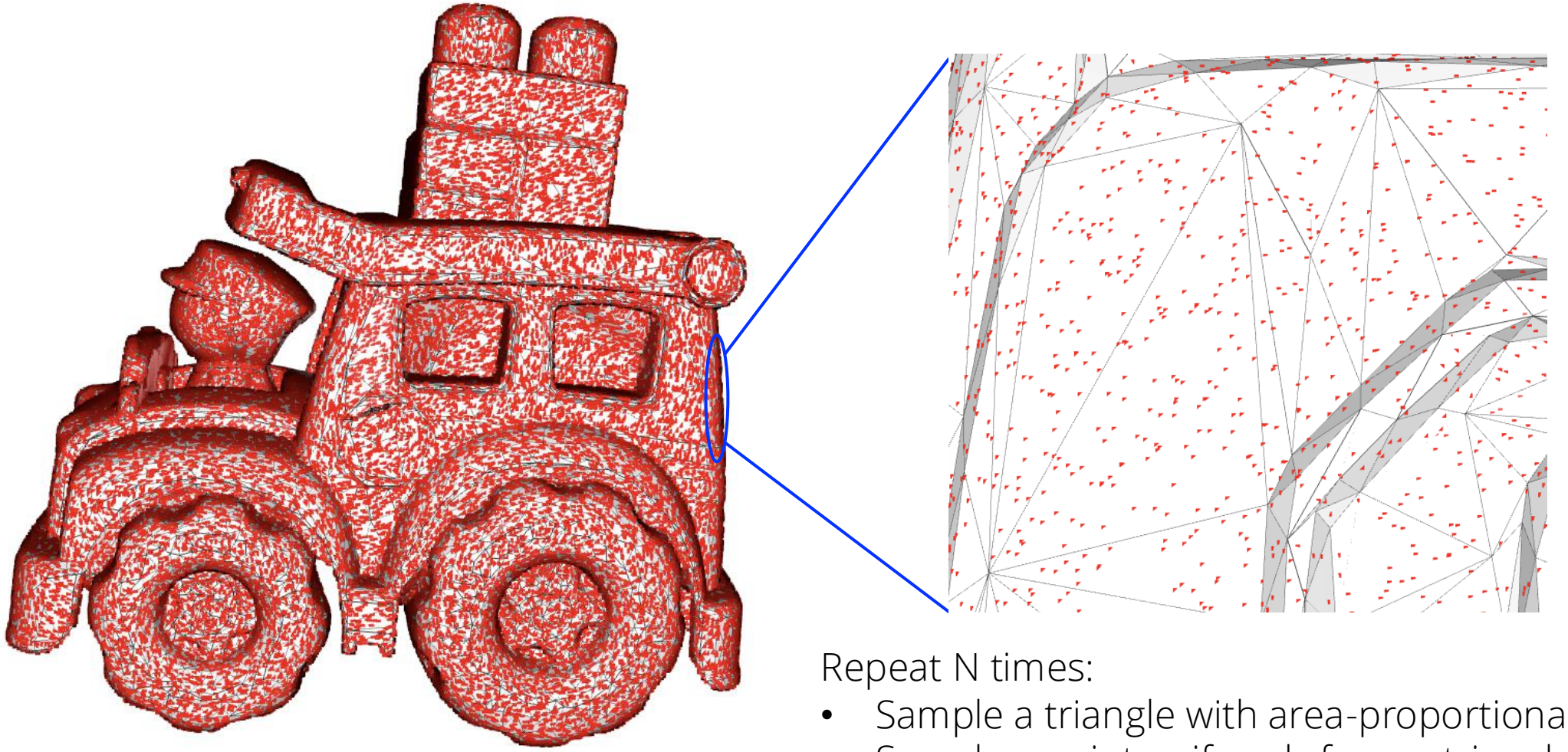
Uniform Point Sampling from Each Triangle



- Small and large meshes have the same number of sampled points
- Sample points are unevenly distributed

3D Asset: https://goo.gle/scanned-objects

Uniform Point Sampling from Meshes



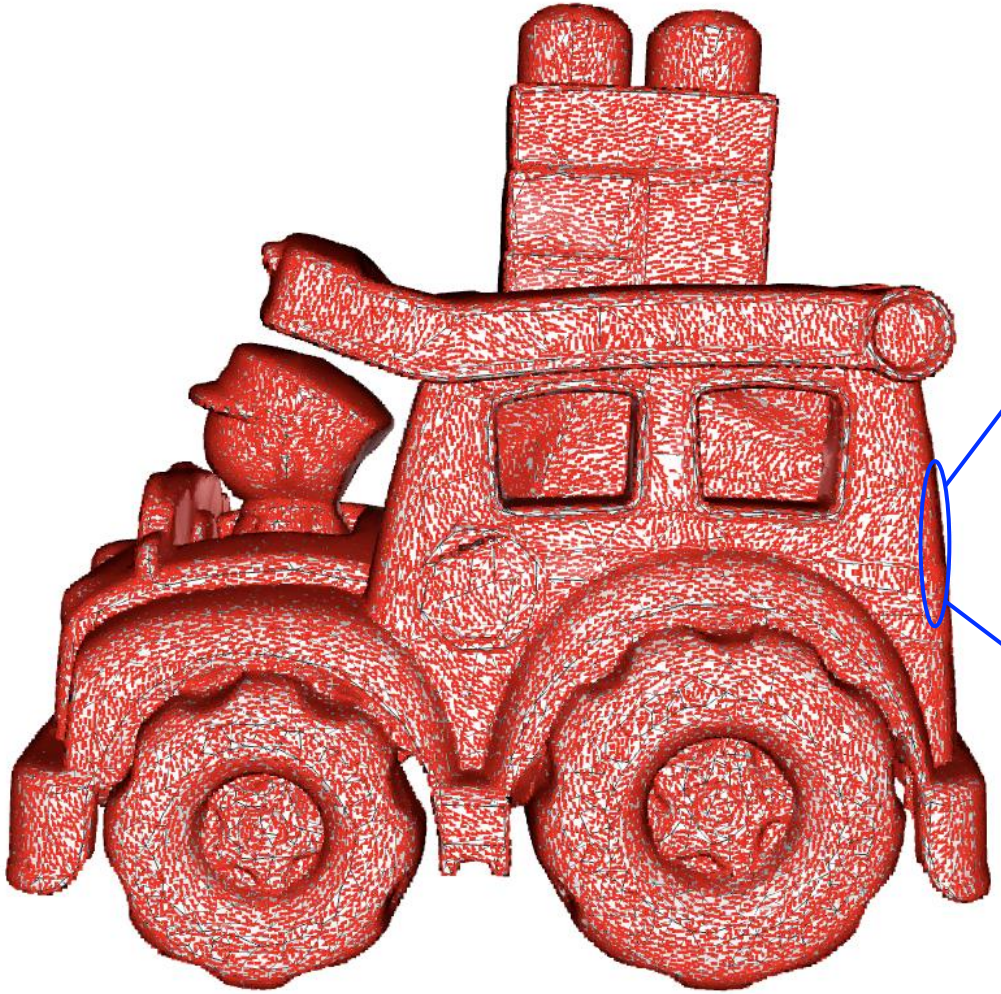
Repeat N times:

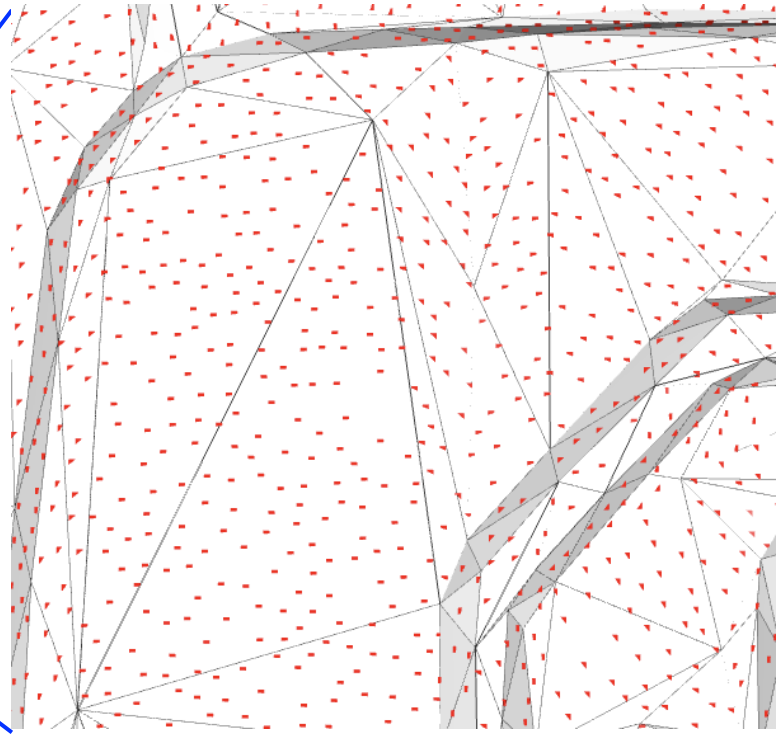
- Sample a triangle with area-proportional probability
- Sample a point uniformly from a triangle

Still sample points are unevenly distributed!

3D Asset: https://goo.gle/scanned-objects

Farthest Point Sampling from Triangles





Repeat N times:

- Select a point furthest form the current set

3D Asset: https://goo.gle/scanned-objects

Farthest Point Sampling

1. Create an initial sample point set *S*
   - Image corners + additional random point.
2. Find the point which is the farthest from all point in *S*

$$d(p,S) = \max_{q \in A}(d(q,S))$$

$$= \max_{q \in A}\left(\min_{0 \le i < N}(d(q,s_i))\right)$$

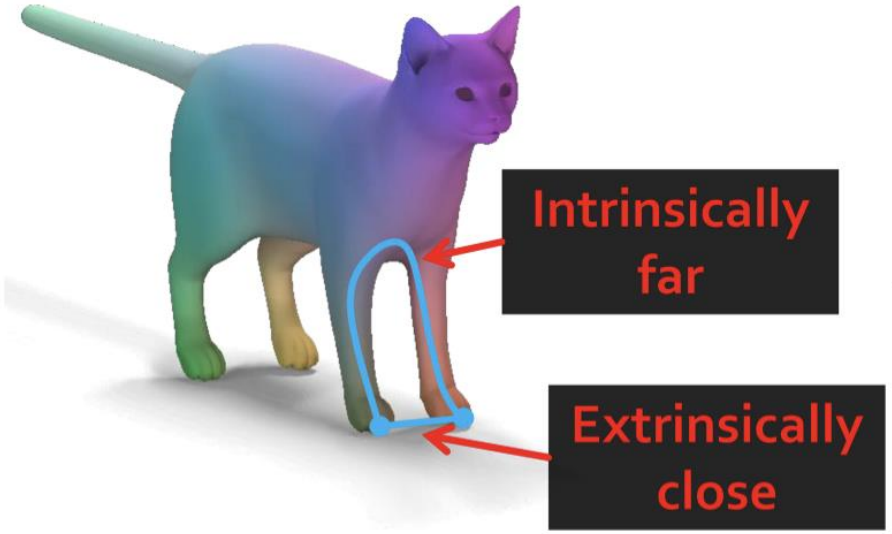
3. Insert the point to *S* and update the distances
4. While more points are needed, iterate

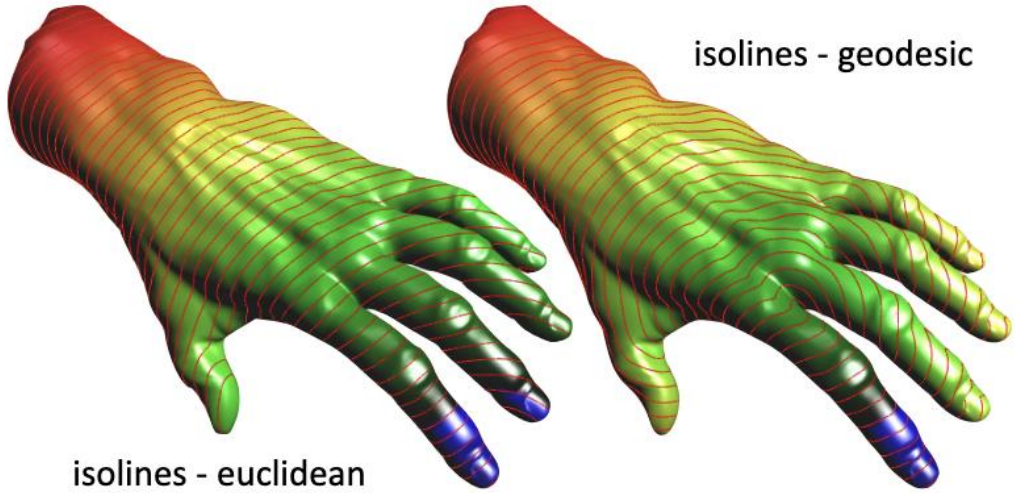
How to define distance on meshes?

Slide credit L. Guibas

Distance Metric on Meshes (Manifolds)

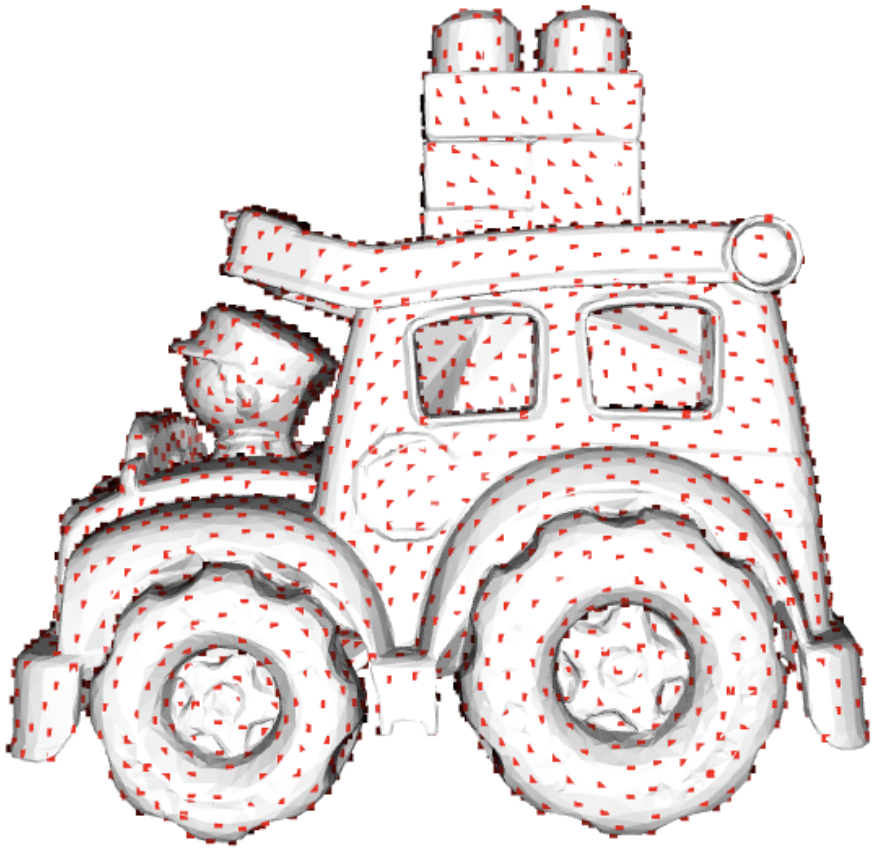
- Geodesics: Straightest and **locally shortest** curves

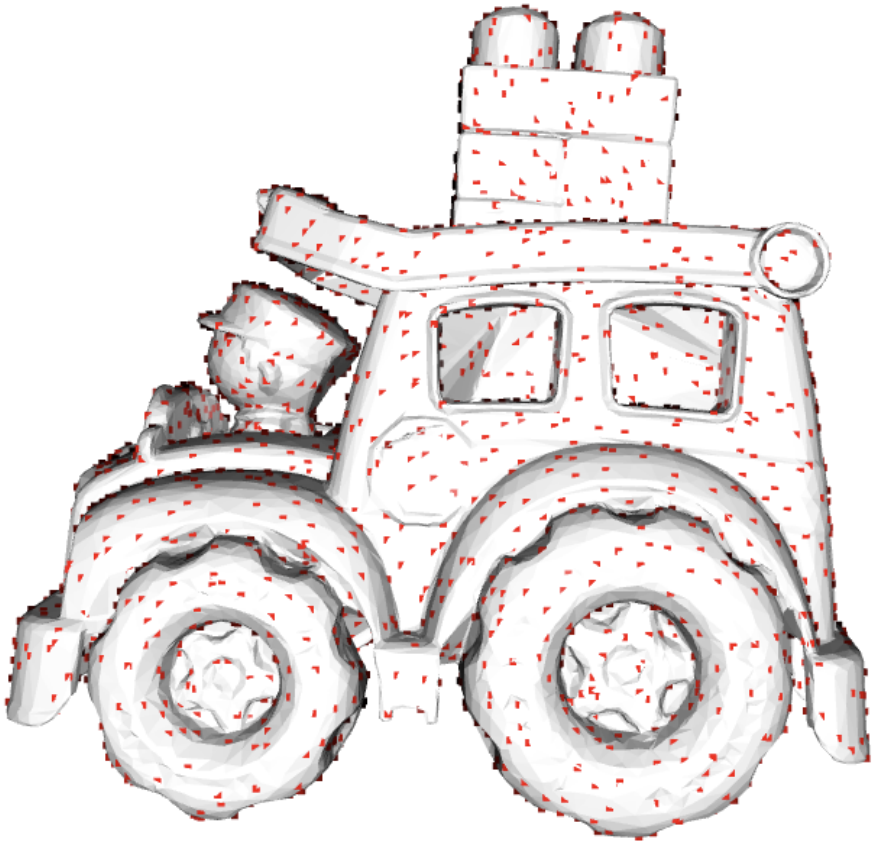




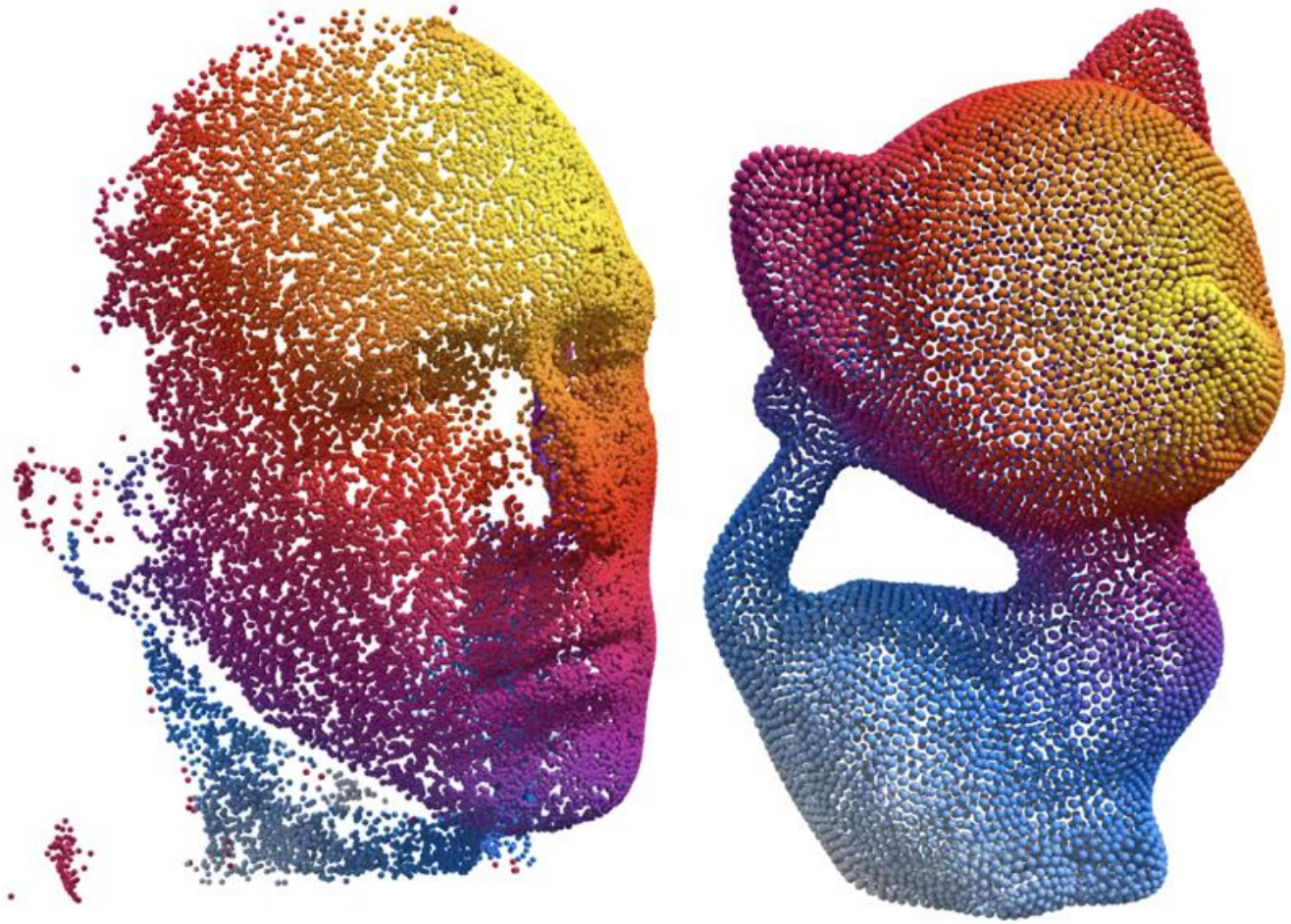
Slide credit L. Guibas

Farthest Point Sampling vs. Uniform Point Sampling





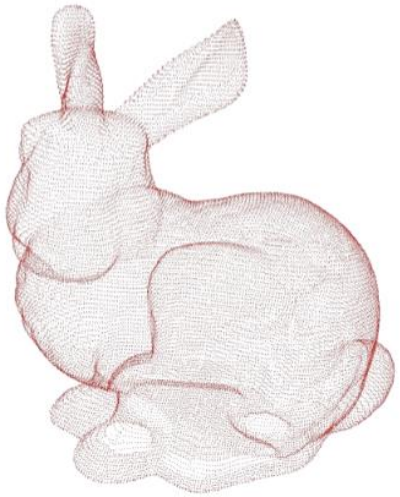
Farthest Point Sampling



Carne, Weischedel, and Wardetzky 2017,
The Heat Method for Distance Computation

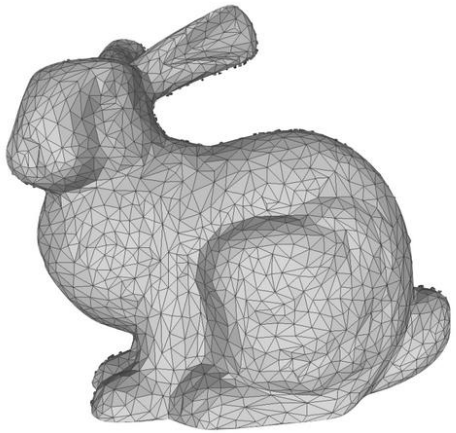
Slide credit L. Guibas

Conversion Across Representations

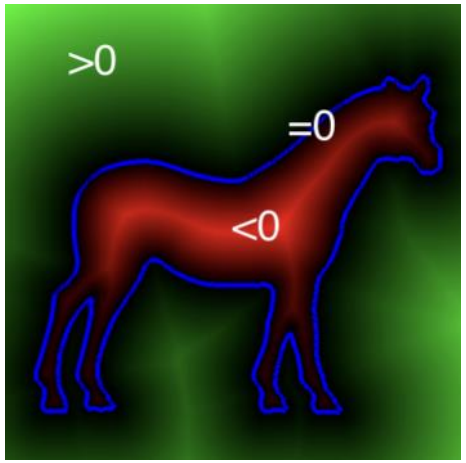


?

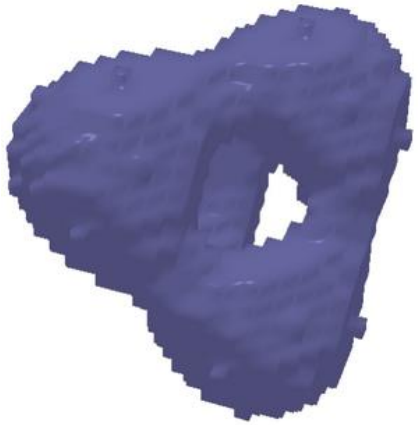
Sampling



?

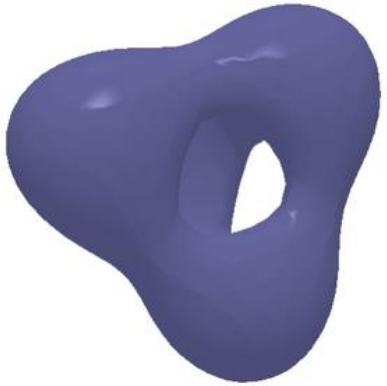


Marching Cubes

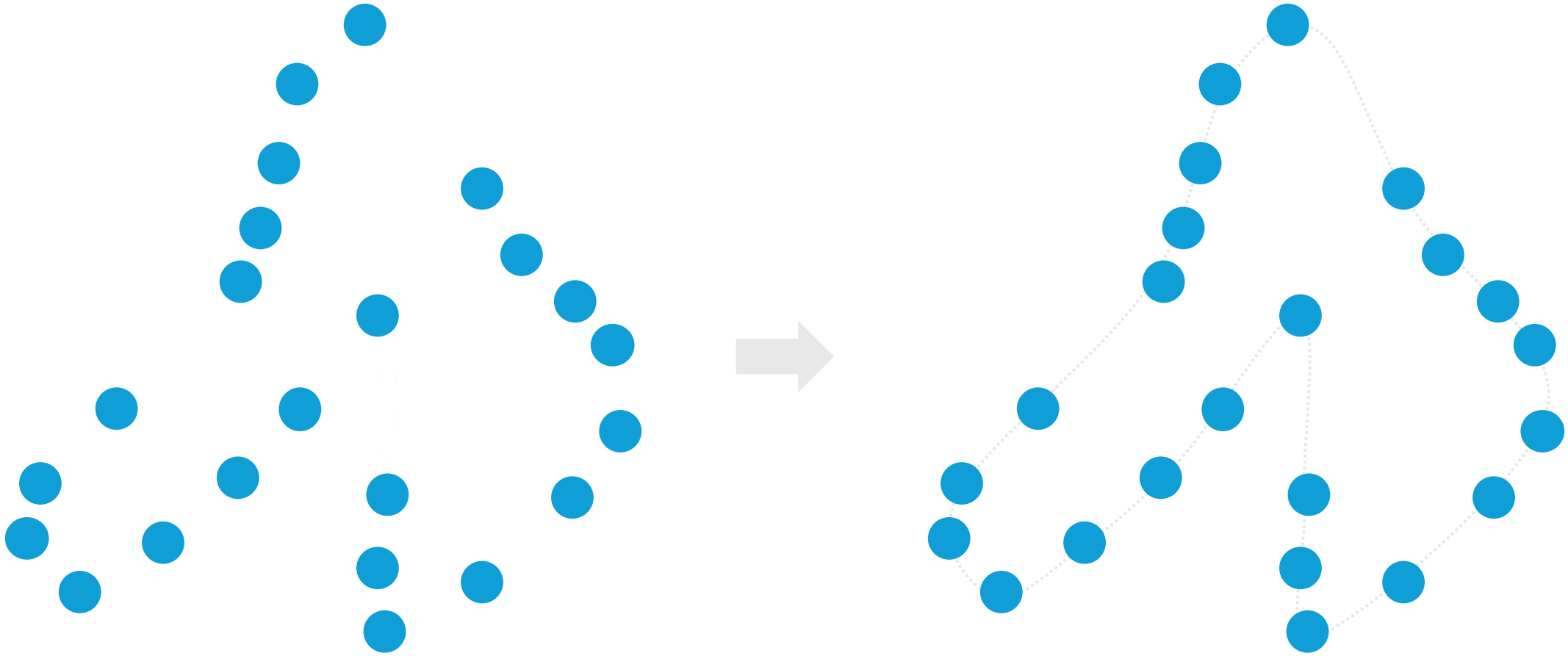


Sampling

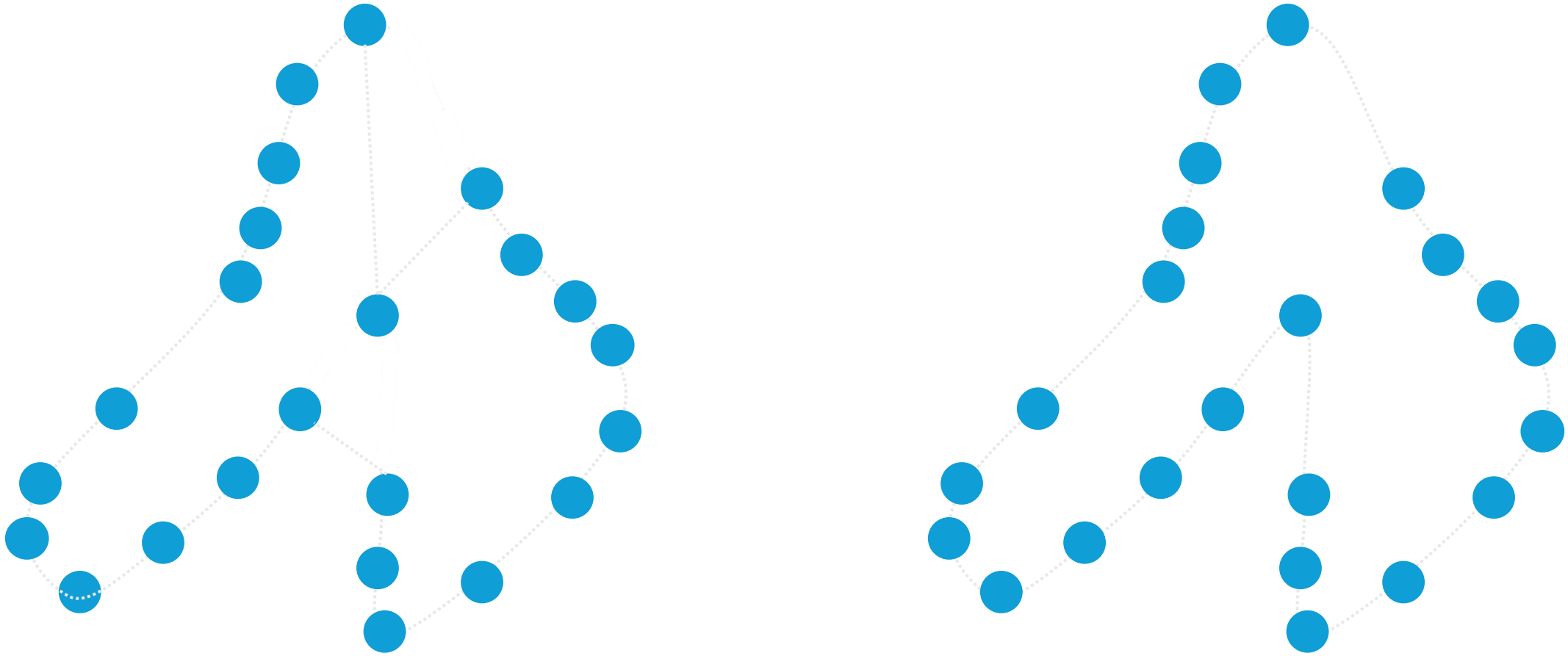
Interpolation



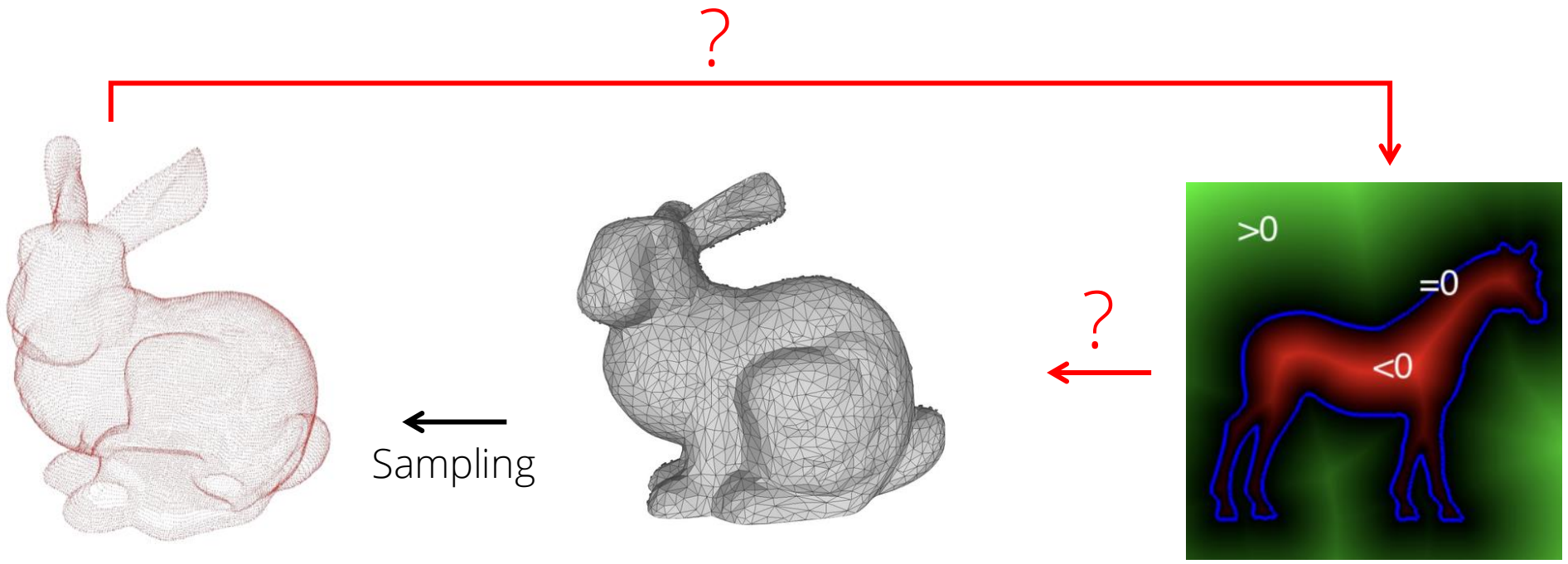
Conversion Across Representations: From Points to Meshes



Converting Unstructured Points to Meshes is Inherently an ill-posed Problem

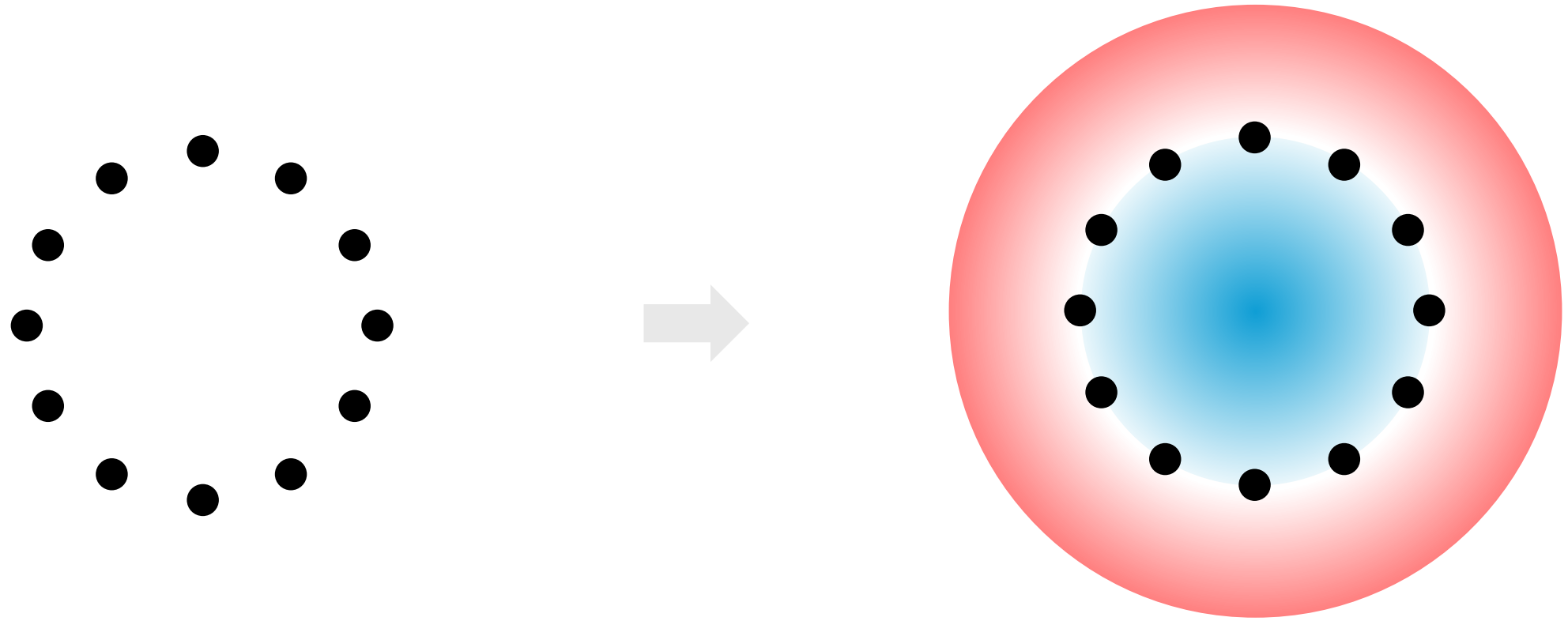


Idea? Converting Points to Implicit Surface Functions
Converting Implicit Surface Functions to Meshes

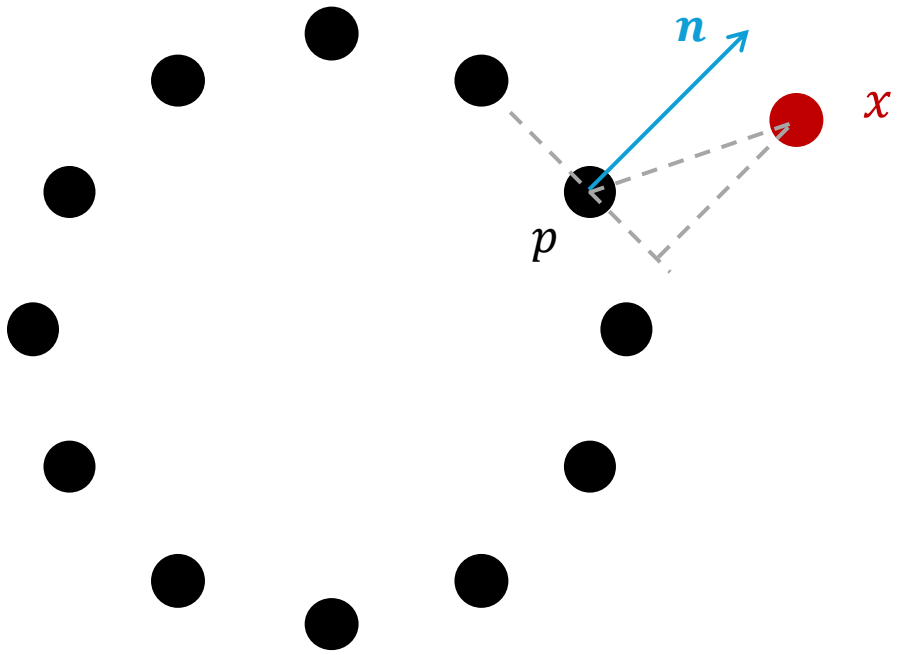


Slide credit S. Tulsiani

Conversion Across Representations: From Points to Implicit Surface Functions



Conversion Across Representations: From Points to Implicit Surface Functions



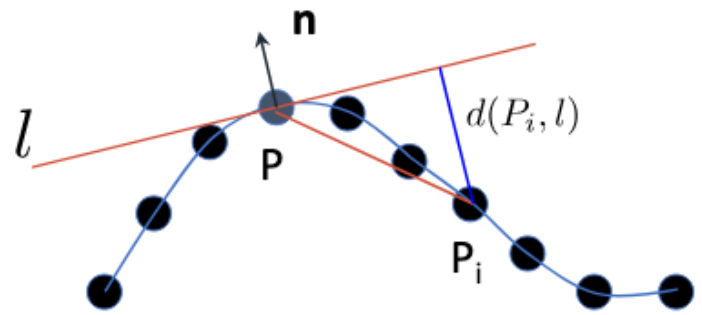
Simplest method:

- Given a point x in space, find nearest point p in the point cloud
- The signed distance $f(x) = (x - p)^T n$

How to obtain surface normal n?

Estimate Surface Normals from Points

Assume we have a clean sampling of the surface. OK, start with a curve.



Goal: find best approximation of the normal at P.

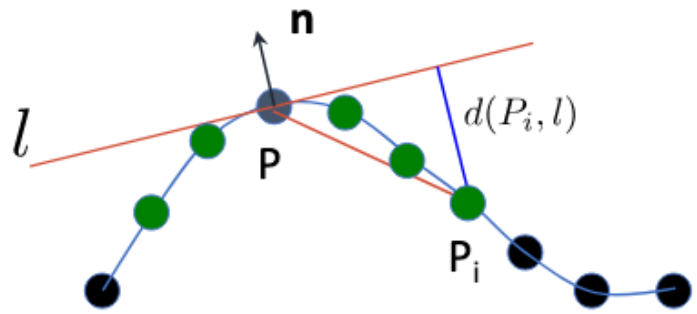
Method: Given line l through P with normal **n**, for another point p_i:

$$d(p_i, l)^2 = \frac{((p_i - P)^T \mathbf{n})^2}{\mathbf{n}^T \mathbf{n}} = ((p_i - P)^T \mathbf{n})^2 \text{ if } \|\mathbf{n}\| = 1$$

Slide credit L. Guibas

Estimate Surface Normals from Points

Assume we have a clean sampling of the surface. OK, start with a curve.



Goal: find best approximation of the normal at P.

Method: Find **n**, minimizing $\sum_{i=1}^{k} d(p_i, l)^2$ for a set of *k* points near P (e.g. *k* nearest neighbors of P).

$$\mathbf{n}_{\text{opt}} = \arg \min_{\|\mathbf{n}\|=1} \sum_{i=1}^{k} ((p_i - P)^T \mathbf{n})^2$$

Slide credit L. Guibas

Estimate Surface Normals from Points

$$\boldsymbol{n}_{opt} = \arg\min_{||\boldsymbol{n}||=1} \sum_{i=1}^{k} ((p_i - P)^{\mathsf{T}}\boldsymbol{n})^2$$

$$= \arg\min_{\boldsymbol{n}} \sum_{i=1}^{k} ((p_i - P)^{\mathsf{T}}\boldsymbol{n})^2 - \lambda \boldsymbol{n}^{\mathsf{T}}\boldsymbol{n}$$

$$\Rightarrow \frac{\partial}{\partial \boldsymbol{n}} \left(\sum_{i=1}^{k} ((p_i - P)^{\mathsf{T}}\boldsymbol{n})^2 \right) - \lambda \frac{\partial}{\partial \boldsymbol{n}} (\boldsymbol{n}^{\mathsf{T}}\boldsymbol{n}) = \mathbf{0}$$

$$\Rightarrow \sum_{i=1}^{k} 2(p_i - P)(p_i - P)^{\mathsf{T}}\boldsymbol{n} = 2\lambda \boldsymbol{n} \Rightarrow C\boldsymbol{n} = \lambda \boldsymbol{n}$$

C: the covariance matrix around P

$\boldsymbol{n}$: the eigen vector of C

Estimate Surface Normals from Points

$$\boldsymbol{n}_{opt} = \arg\min_{||\boldsymbol{n}||=1} \sum_{i=1}^{k} ((p_i - P)^{\mathsf{T}}\boldsymbol{n})^2 = \arg\min_{||\boldsymbol{n}||=1} \boldsymbol{n}^{\mathsf{T}} C \boldsymbol{n}$$

$$= \arg\min_{\boldsymbol{n}} \sum_{i=1}^{k} ((p_i - P)^{\mathsf{T}}\boldsymbol{n})^2 - \lambda \boldsymbol{n}^{\mathsf{T}}\boldsymbol{n}$$

$$\Rightarrow \frac{\partial}{\partial \boldsymbol{n}} \left(\sum_{i=1}^{k} ((p_i - P)^{\mathsf{T}}\boldsymbol{n})^2 \right) - \lambda \frac{\partial}{\partial \boldsymbol{n}} (\boldsymbol{n}^{\mathsf{T}}\boldsymbol{n}) = \mathbf{0}$$

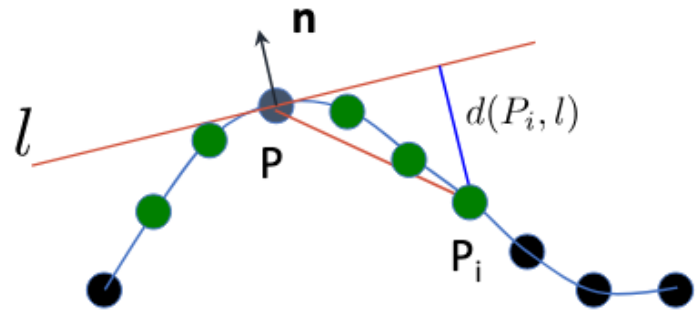
$$\Rightarrow \sum_{i=1}^{k} 2(p_i - P)(p_i - P)^{\mathsf{T}}\boldsymbol{n} = 2\lambda \boldsymbol{n} \Rightarrow C\boldsymbol{n} = \lambda \boldsymbol{n}$$

C: the covariance matrix around P

$\boldsymbol{n}$: the eigen vector corresponding to the smallest eigen value of C

Estimate Surface Normals from Points

Assume we have a clean sampling of the surface. OK, start with a curve.



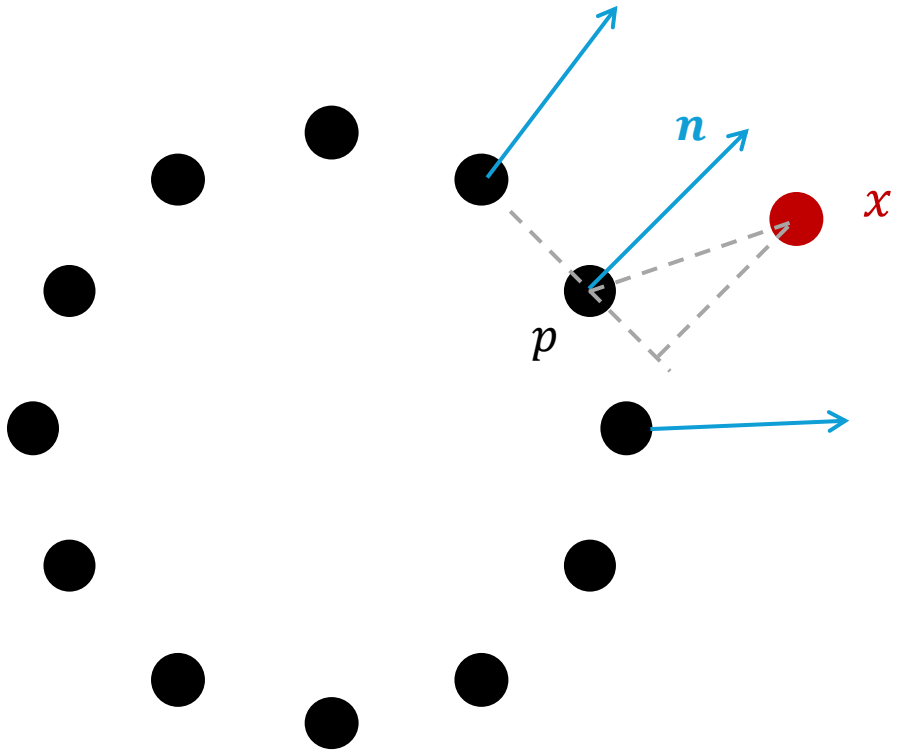
Using Lagrange multipliers:

$$\frac{\partial}{\partial \mathbf{n}} \left(\sum_{i=1}^{k} ((p_i - P)^T \mathbf{n})^2 \right) - \lambda \frac{\partial}{\partial \mathbf{n}} \left(\mathbf{n}^T \mathbf{n} \right) = 0$$

$$\sum_{i=1}^{k} 2(p_i - P)(p_i - P)^T \mathbf{n} = 2\lambda \mathbf{n}$$

Slide credit L. Guibas

Conversion Across Representations: From Points to Implicit Surface Functions



Simplest method:

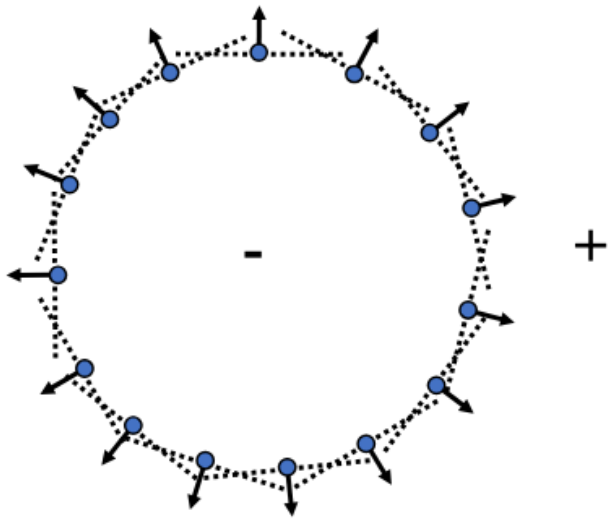
- Given a point x in space, find nearest point p in the point cloud
- The signed distance $f(x) = (x - p)^{\mathsf{T}} n$

  (obtain surface normals from local covariance matrix)
- Need consistently oriented normals. In general, difficult problem, but can try to locally connect points and fix orientations

Slide adapted from L. Guibas

Signed Distance Function from Points and Normals

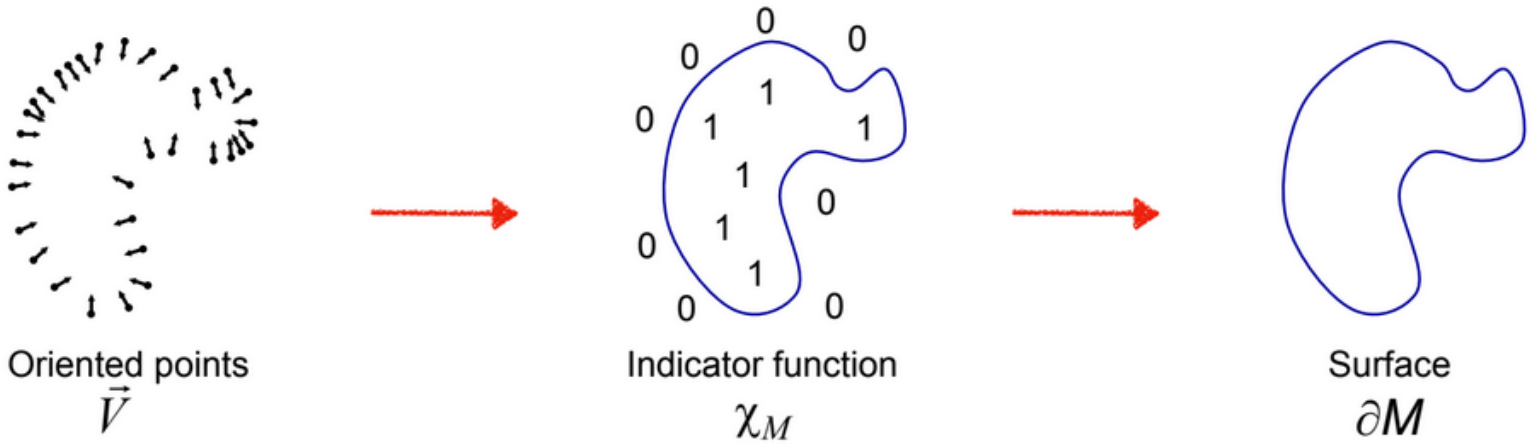
- Input: Points + Normals
- Normals help to distinguish between inside and outside
- Computed via locally fitting planes at the points (and consistently oriented)
- Previous method is very local and gives noisy results



"Surface reconstruction from unorganized points", Hoppe et al., ACM SIGGRAPH 1992
http://research.microsoft.com/en-us/um/people/hoppe/proj/recon/

Slide credit L. Guibas

Other Ideas? Poisson Reconstruction



Key Idea: Find a (smoothed) indicator function whose gradient corresponds to the surface normals

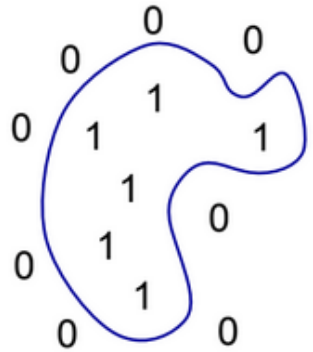
A surface can then be extracted via iso-surface of the indicator function

Slide credit S. Tulsiani

Poisson Reconstruction

Oriented points

$\vec{V}$



Indicator function

χ_M

$$\min_\chi \|\nabla\chi - \vec{V}\| \longrightarrow \Delta\chi \equiv \nabla\cdot\nabla\chi = \nabla\cdot\vec{V}$$

Efficient solvers exist (common problem in physics)

Standard tool in geometry processing

Slide credit S. Tulsiani

Poisson Reconstruction in Practice

215 million data points from 1000 scans

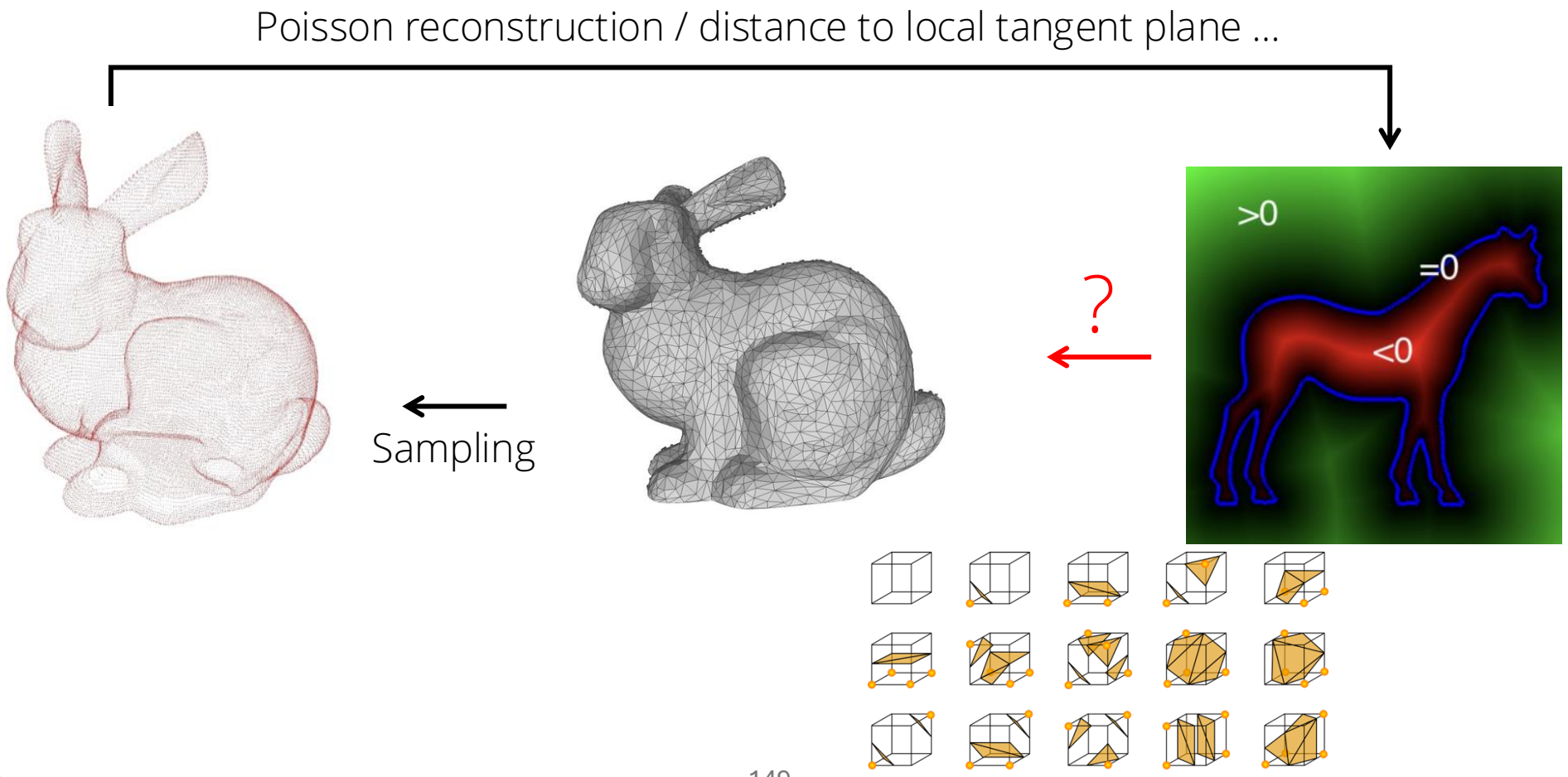
22 million triangles in reconstruction

Compute: 2.1hrs, 6.6GB memory

This is still an open research problem!

Slide credit S. Tulsiani

Idea? Converting Points to Implicit Surface Functions
Converting Implicit Surface Functions to Meshes



Conversion Across Representations: From Implicit Surface Function to Discrete Map

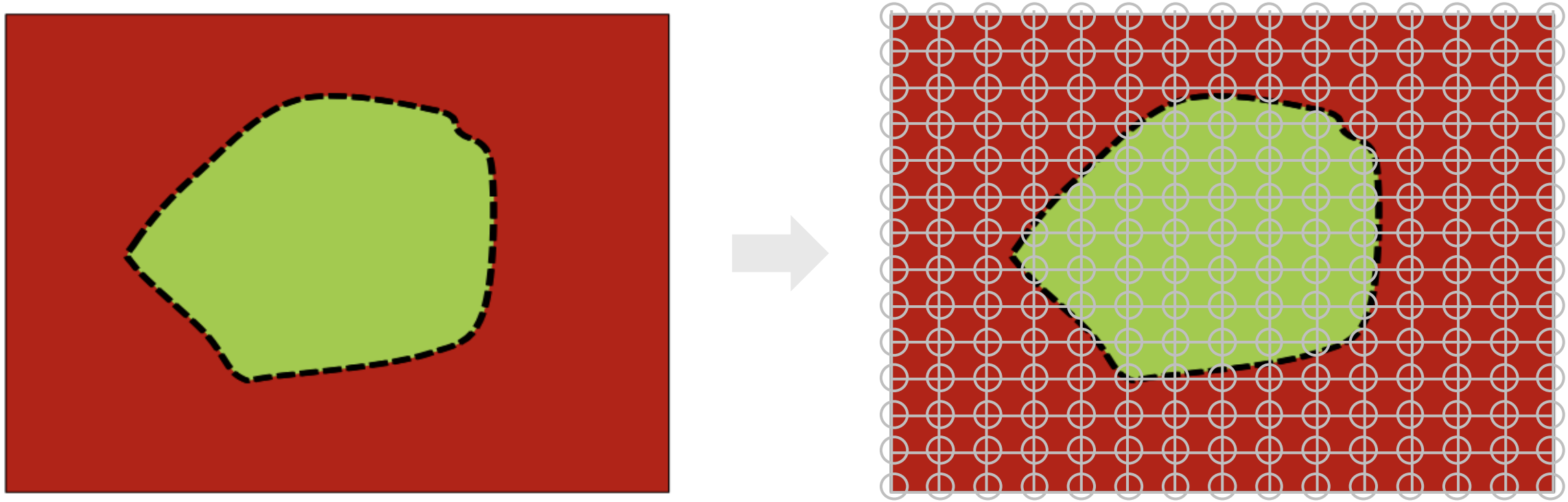
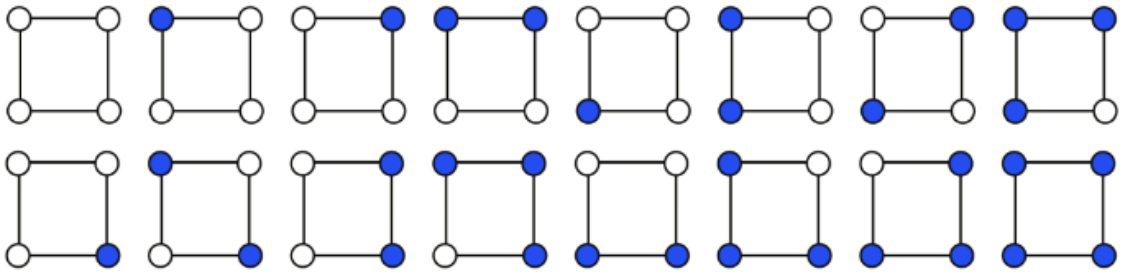
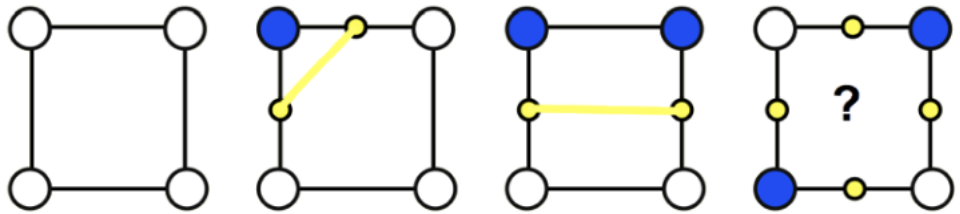


Image credit L. Guibas

We Can Apply the Marching Cubes Method to Convert Implicit Surface Function to Meshes



16 cases



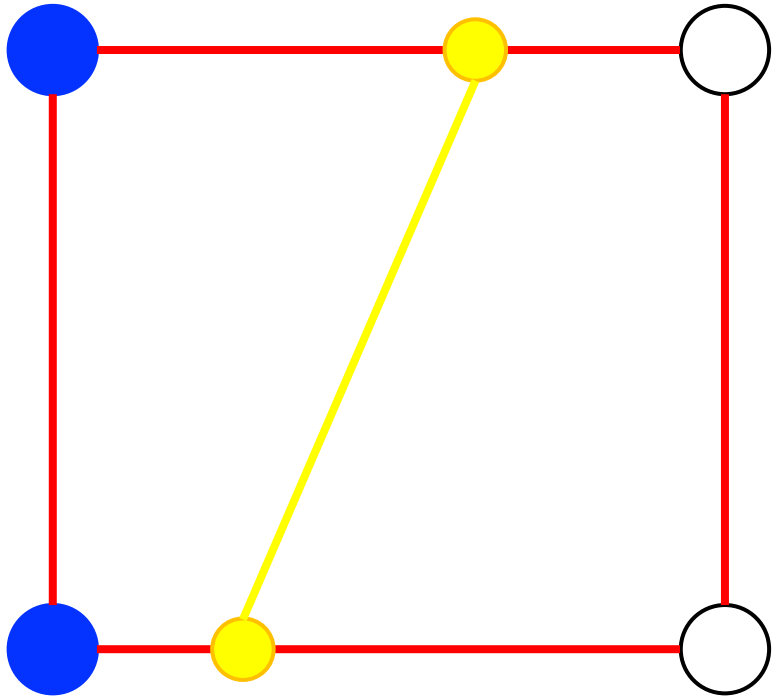
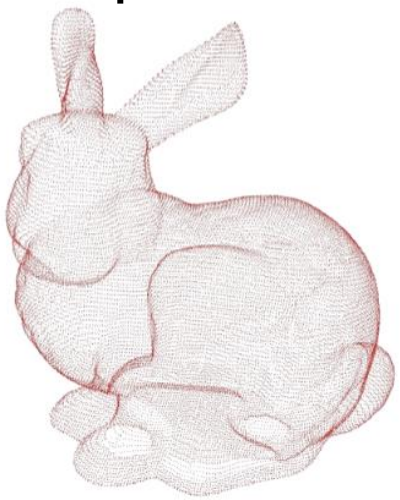
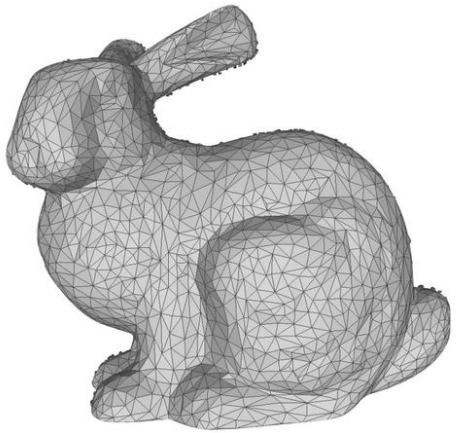


Image credit L. Guibas

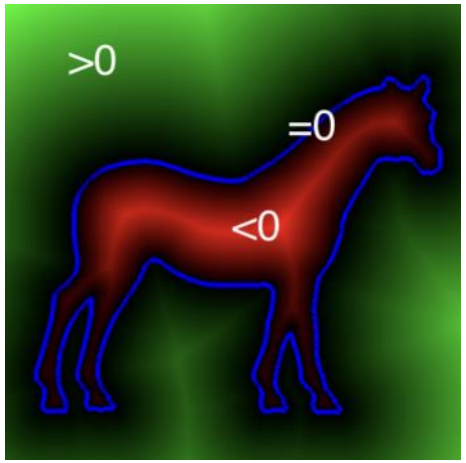
Poisson reconstruction / distance to local tangent plane …



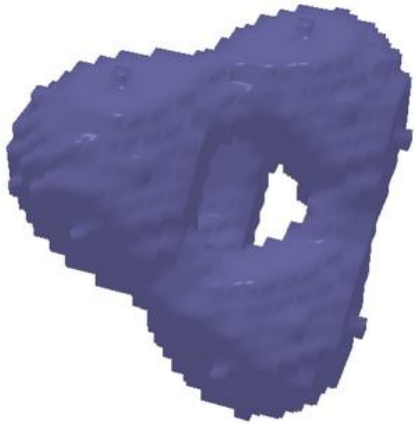
Sampling



Marching Cubes

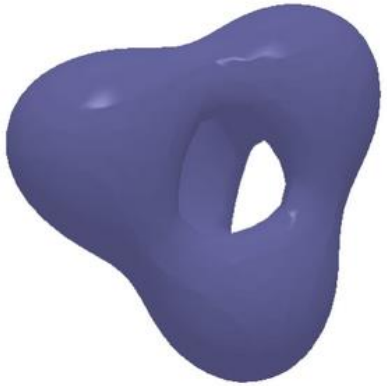


Marching Cubes



Sampling

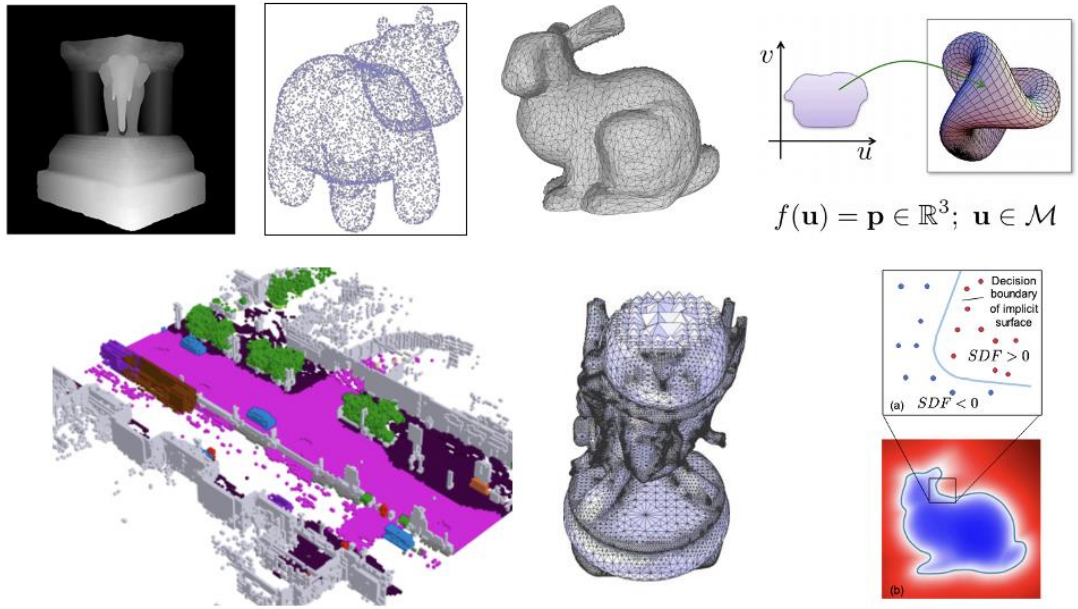
Interpolation



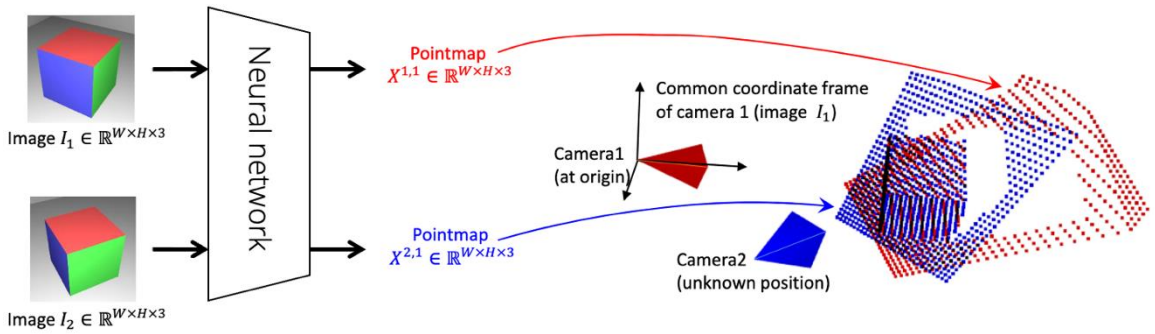
Summary

Structure from Motion

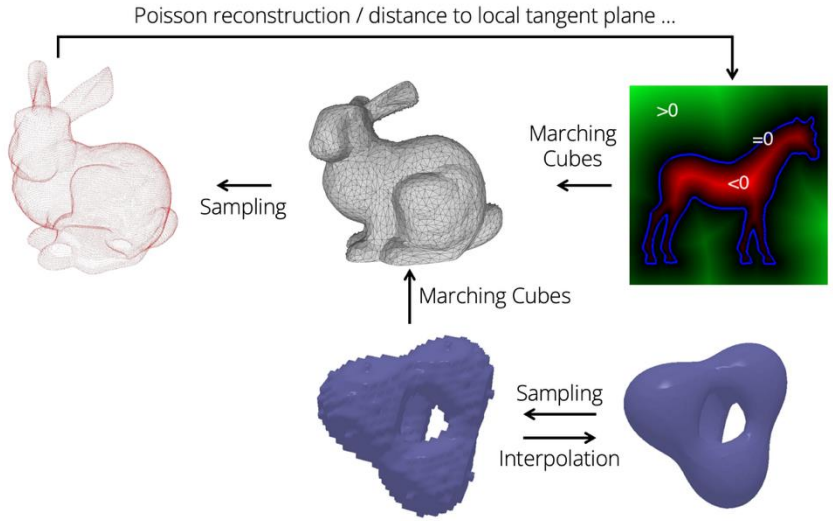
3D Representations



DUSt3R



Conversion between 3D Representations



What We Will Cover Next Week

- Rendering and 3D scene reconstruction