

L18: Mesh Processing

Hao Su

Ack: Minghua Liu helped to prepare slides

Agenda

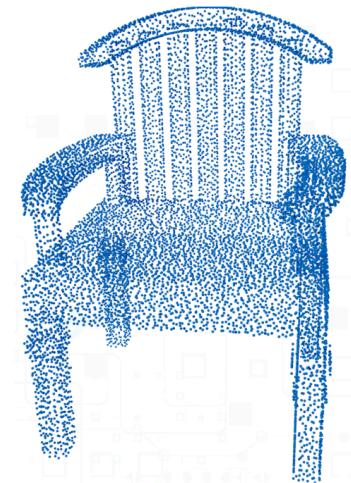
- Surface Reconstruction
- A Peek into Mesh Processing

Surface Reconstruction

- Explicit Algorithms
- Implicit Algorithms

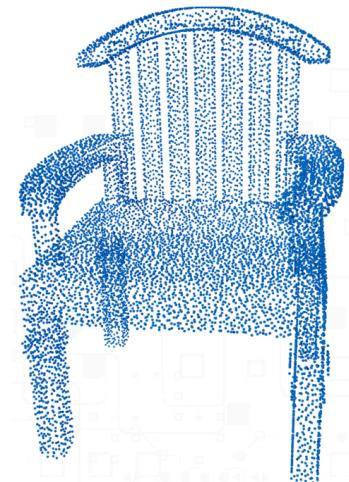
Surface Reconstruction Task

- Input: point cloud (with or without normals)
- Output: triangle mesh



Two Basic Families

- Explicit algorithms
 - Directly connect the input points with triangles, e.g.,
 - ball-pivoting algorithm
 - extrinsic-intrinsic ratio algorithm
- Implicit algorithms
 - Approximate the input points by implicit field functions $S = \{x : F(x) = 0\}$
 - Then extract iso-surfaces, e.g.,
 - poisson surface reconstruction
 - reconstruction with RBF

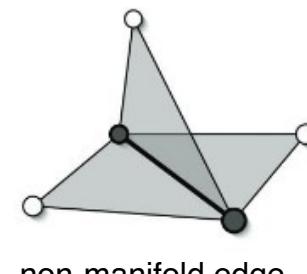
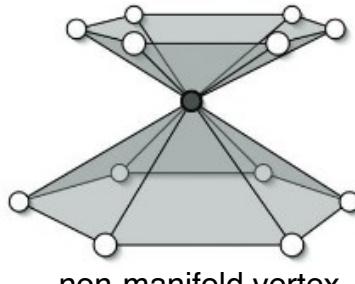
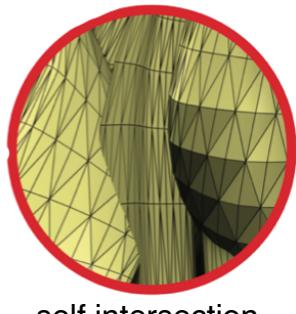


Some Desired Properties of the Algorithm

- Fast: The input point cloud may be large. We expect the computation to be fast.
- Robust: May recover the underlying surface structure even when the input point cloud is noisy
- Output mesh is desired to satisfy some geometric constraints

Geometric Constraint: Manifold

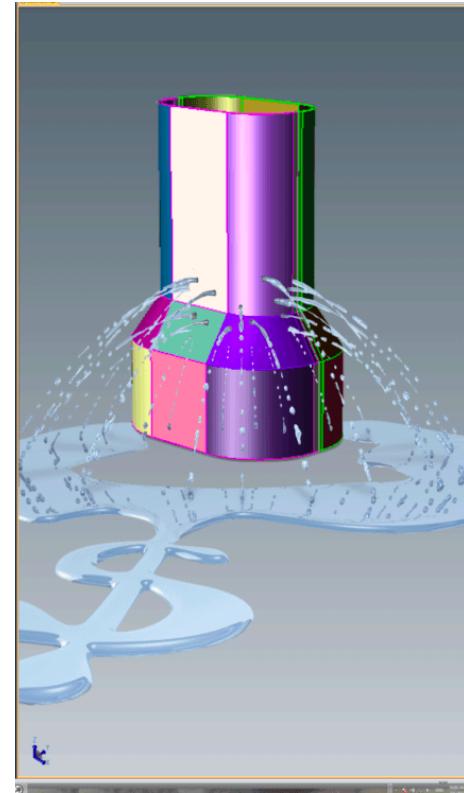
- A mesh is **manifold** if it does not contain:
 - self intersection
 - non-manifold edge (has more than 2 incident faces)
 - non-manifold vertex (one-ring neighborhood is not connected after removing the vertex)



- A useful property for many subsequent geometry processing pipelines
 - e.g., to add texture maps and ...

Geometric Constraint: Watertight

- A **manifold** mesh is **watertight** if each edge has **exactly** two incident faces, i.e., no boundary edges.
- Defines the interior, hence the volume of a solid object
- Required by many physical-simulation algorithms:
 - Estimate mass from density
 - Collision between objects
 - Force simulation
 - ...



<https://transmagic.com/wp-content/uploads/2016/05/watertight-solid-3d-cad-models-transmagic.png>

Surface Reconstruction

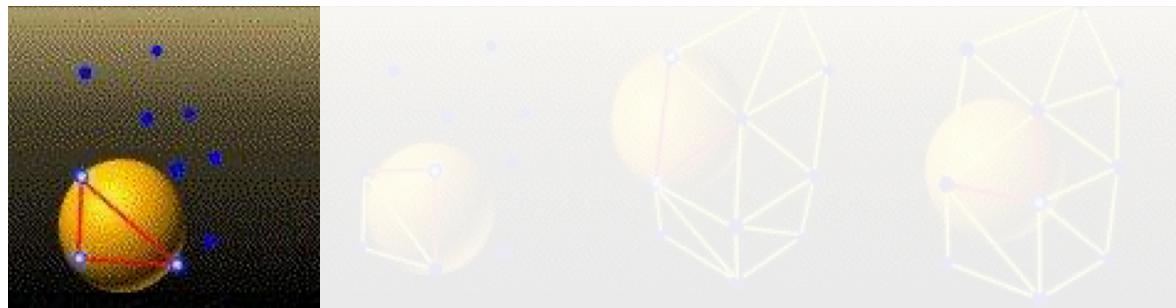
- Explicit Algorithms
 - Ball-Pivoting Algorithm
 - Extrinsic-Intrinsic Ratio Algorithm
- Implicit Algorithms

Ball-Pivoting Algorithm

- Input: a point cloud and a hyper-parameter ρ

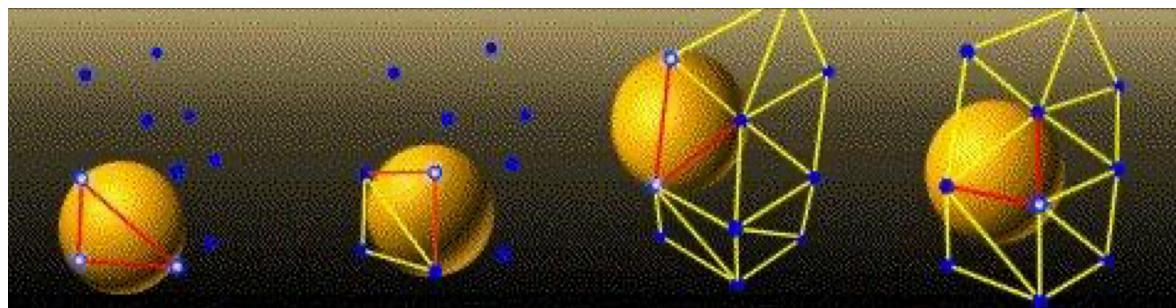
Ball-Pivoting Algorithm

- Input: a point cloud and a hyper-parameter ρ
- Assumption:
 - input points are dense enough that a ball of radius ρ cannot pass through the surface without touching the points.



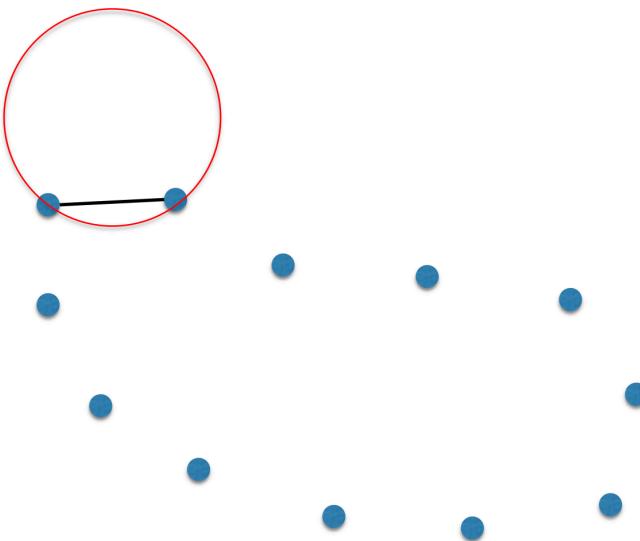
Ball-Pivoting Algorithm

- Input: a point cloud and a hyper-parameter ρ
- Assumption:
 - input points are dense enough that a ball of radius ρ cannot pass through the surface without touching the points.
- Principle for face formation:
 - three points form a triangle if a ball of radius ρ touches them without containing any other points.



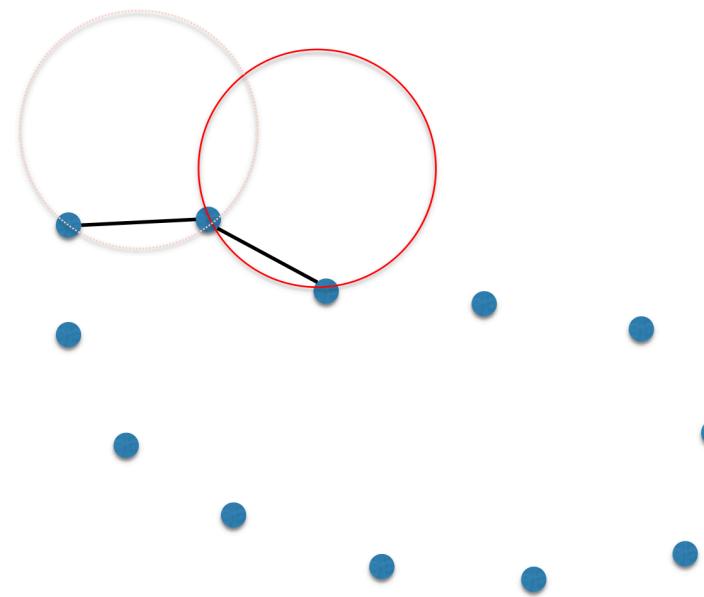
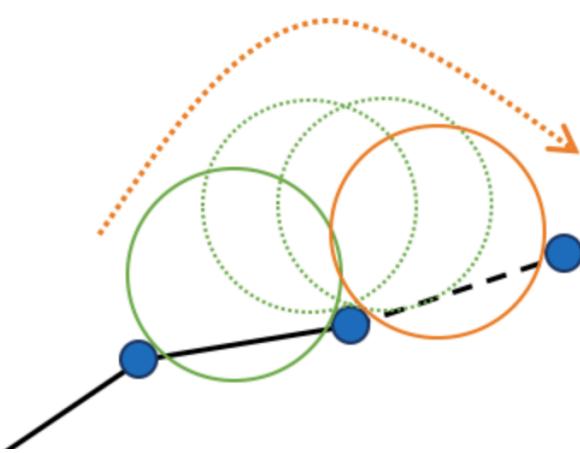
Ball-Pivoting Algorithm (2D)

- Starting with a corner point and a ρ -ball
- Verify potential edges (triangles) in the ρ -neighborhood by the previous principle



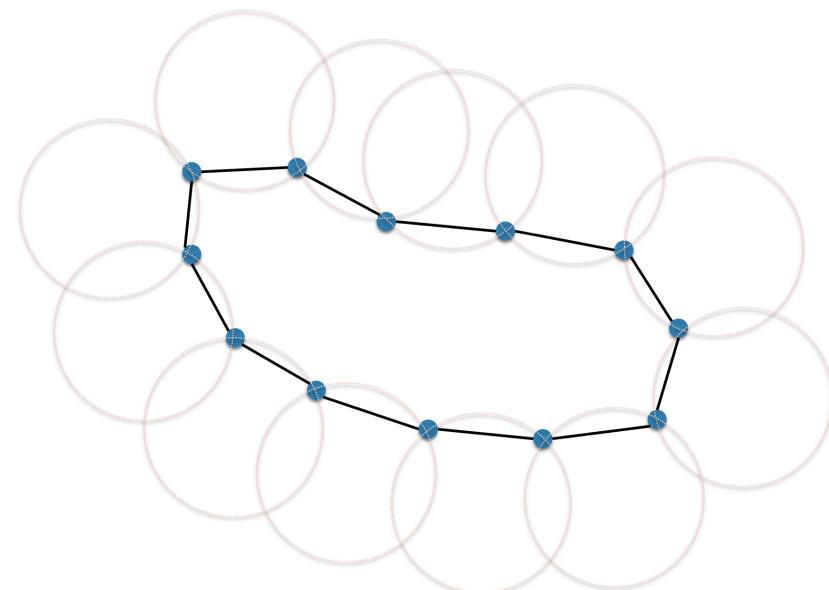
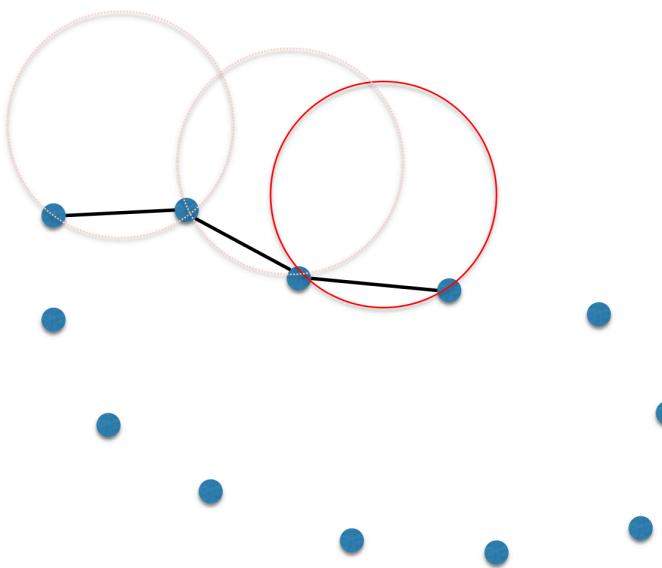
Ball-Pivoting Algorithm (2D)

- The ball pivots around an edge (triangles) until it touches another point, forming another triangle.



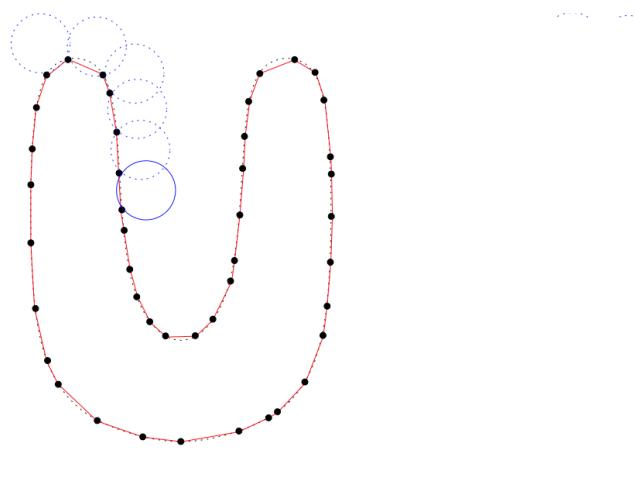
Ball-Pivoting Algorithm (2D)

- The process continues until all reachable edges have been tried
- Then starts from another seed triangle, until all points have been considered.



Radius ρ Matters

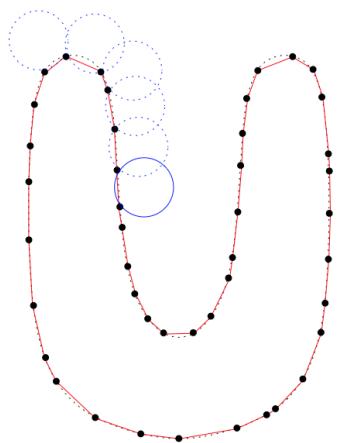
- Appropriate radius (a)



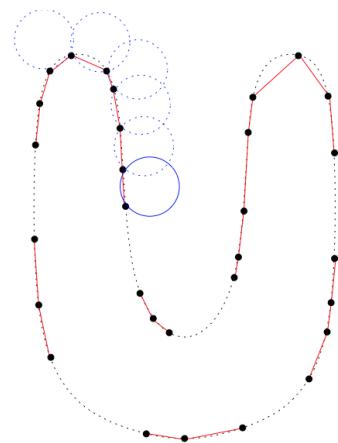
(a)

Radius ρ Matters

- Appropriate radius (a)
- Radius too small: some of the edges will not be created, leaving holes. (b)



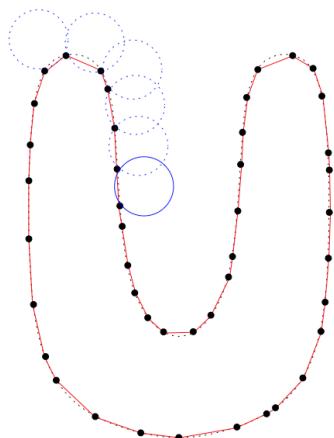
(a)



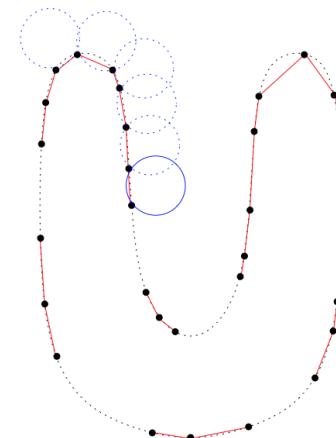
(b)

Radius ρ Matters

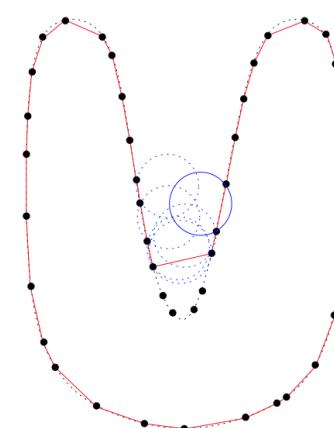
- Appropriate radius (a)
- Radius too small: some of the edges will not be created, leaving holes. (b)
- Large radius: some of the points will not be reached (when the curvature of the manifold is larger than $1/\rho$)
(c)



(a)



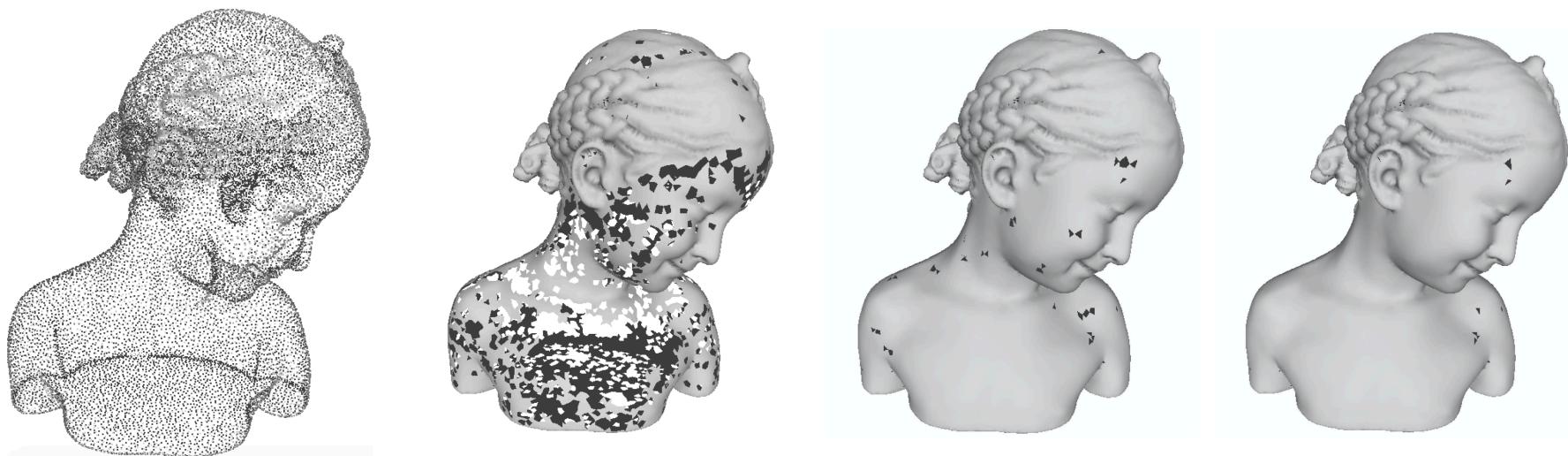
(b)



(c)

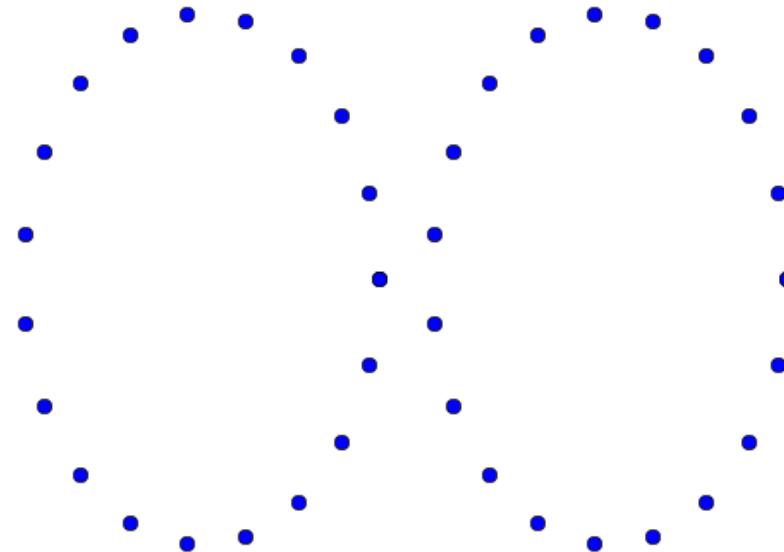
Iterative Approach

- Using multiple radius, iteratively connects the points.
- Small Radius capture high frequencies.
- Large Radius close holes.



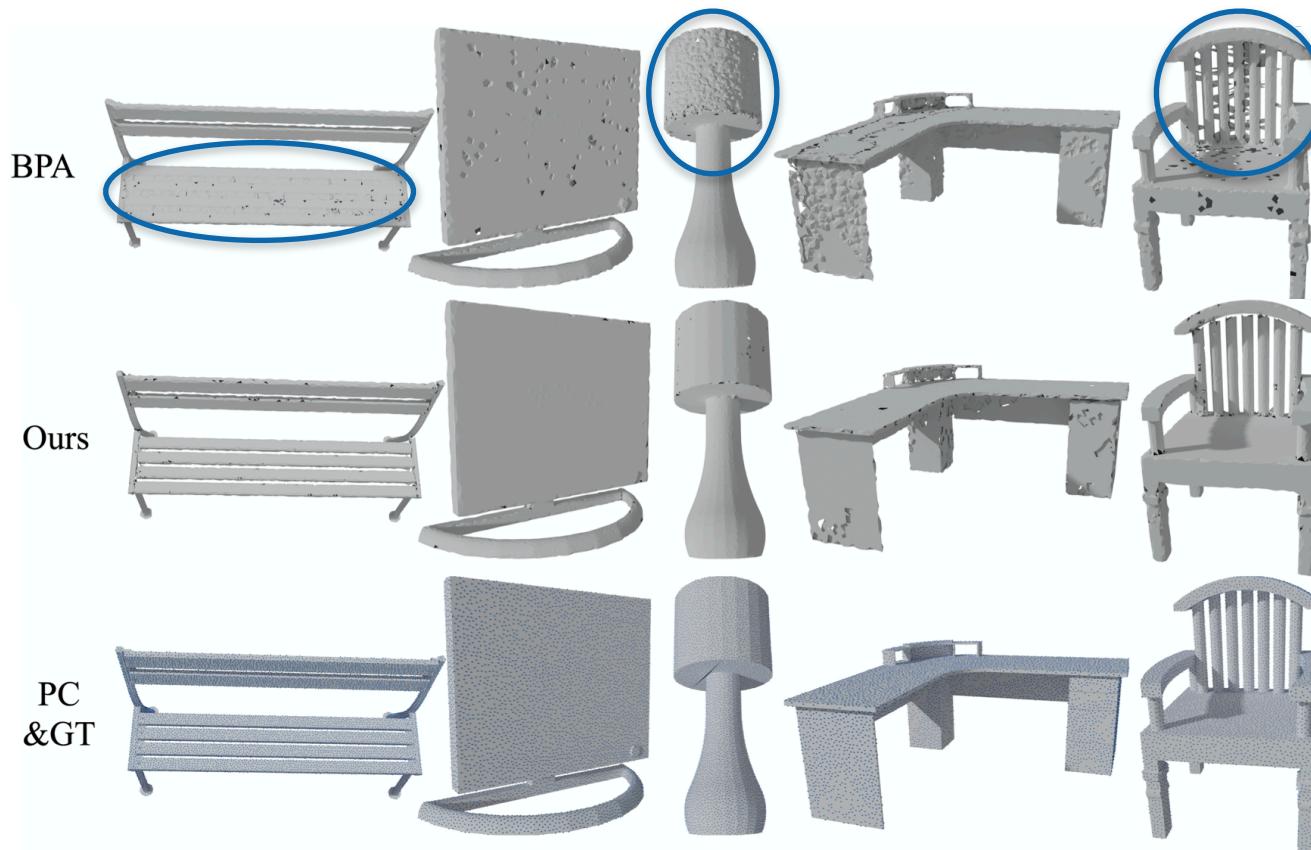
Ambiguous Structures

- Sometimes, defining a rule for structure estimation is hard
 - e.g., we tend to interpret the following point cloud as two disjoint ellipses; however, no ρ value allows us to separate them



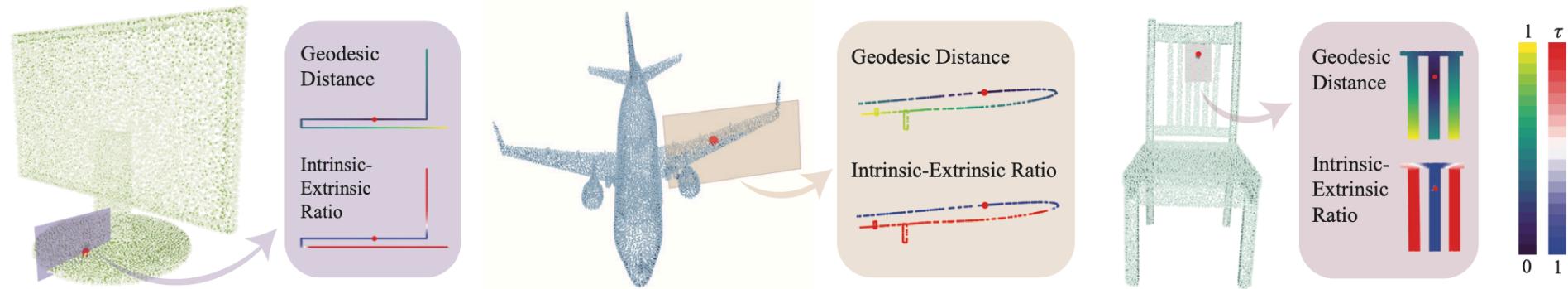
Ambiguous Structures

- Traditional Rule-based methods cannot handle ambiguous structures (e.g., thin structures & adjacent parts).

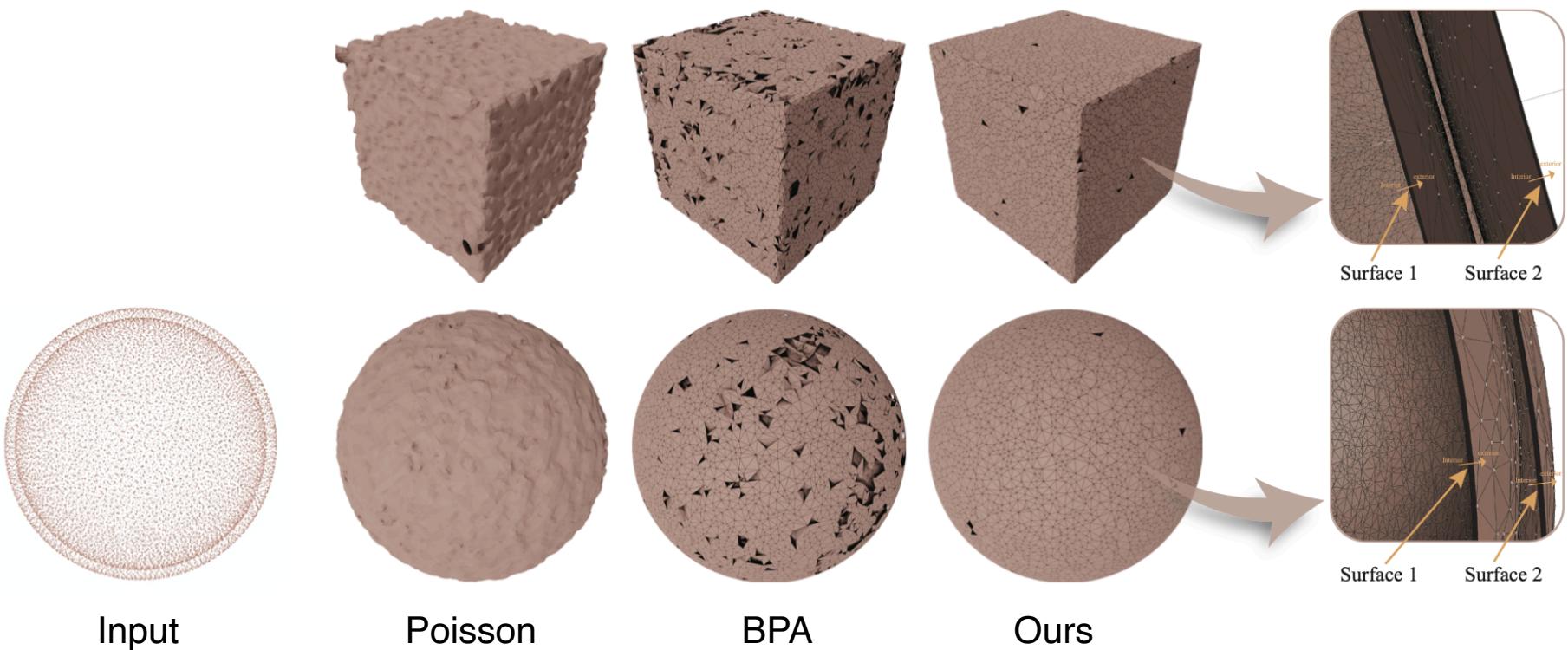


Review: Learning-Based Method

- Train a network to filter out incorrect connections.
- Utilize the Intrinsic-Extrinsic Ratio to guide the training.



Ambiguous Structures



Pros & Cons

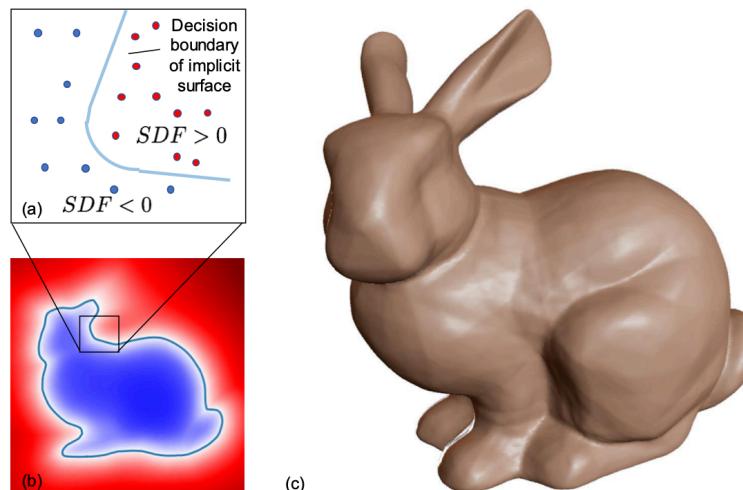
- Pros:
 - Linear complexity (fast)
 - No dependence on normals
- Cons:
 - Can lead to non-manifold situations, and no watertight guarantee
- Regarding robustness:
 - Learning can improve the robustness
 - However, current learning-based method would still not work when the sampling density is low

Surface Reconstruction

- Explicit Algorithms
- Implicit Algorithms
 - RBF implicit function estimation
 - Marching cube

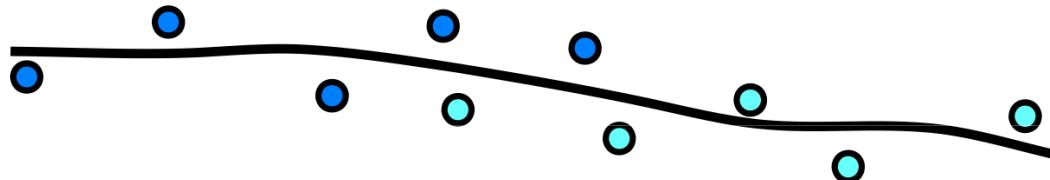
Implicit Field Function

- Interior: $F(x, y, z) < 0$
- Exterior: $F(x, y, z) > 0$
- Surface: $F(x, y, z) = 0$ (zero set, zero iso-surface)
- Example implementation:
 - SDF: $F(x, y, z) = \text{distance to the surface}$



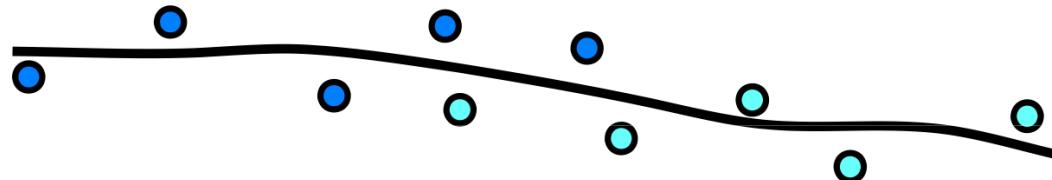
Implicit Meshing Algorithm

- Two basic steps:
 1. Estimate an implicit field function from data
 2. Extract the zero iso-surface



Implicit Meshing Algorithm

- Two basic steps:
 1. **Estimate an implicit field function from data**
 2. Extract the zero iso-surface



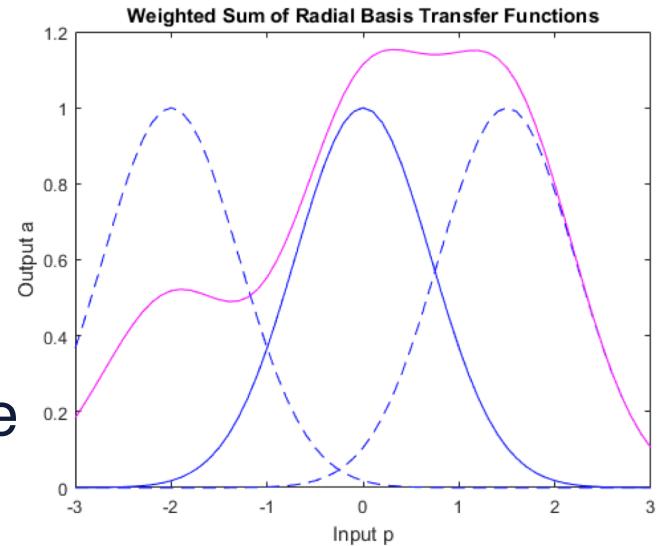
Radial Basis Functions

- Radial basis functions (RBF) $\phi_c(\mathbf{x})$: function value depends only on the distance from a center point c
- Use a weighted sum of radial basis functions to approximate the shape:

$$\phi_c(\mathbf{x}) = \phi(\|\mathbf{x} - \mathbf{c}\|)$$

$$f(\mathbf{x}) = \sum_{i=1}^n \omega_i \phi\left(\|\mathbf{x} - \mathbf{x}_i\|\right) + p(x)$$

where p is a polynomial of low degree



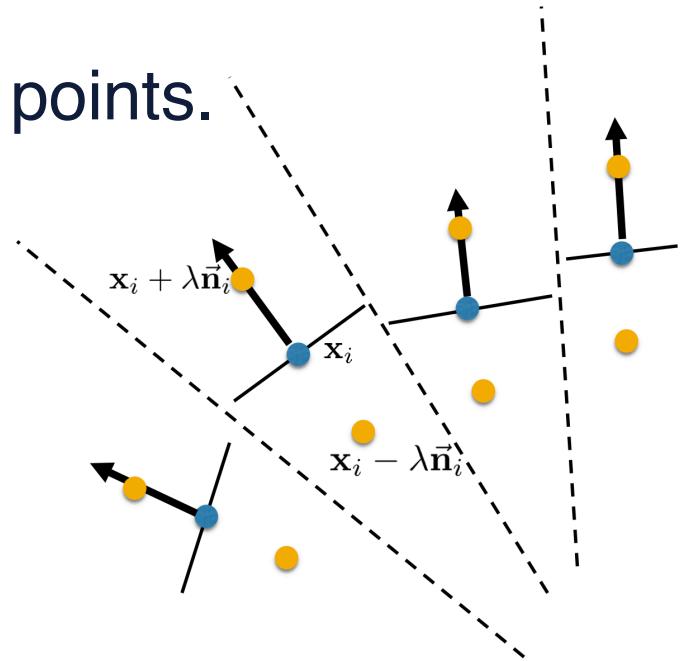
Constraints: Avoiding Non-Trivial Solutions

- Only force input points \mathbf{x}_i to have zero value
 $f(\mathbf{x}_i) = 0$ is not enough, it may get the trivial solution
 $f(\mathbf{x}) \equiv 0$
- Use normal to add off-surface points.

$$f(\mathbf{x}_i) = 0$$

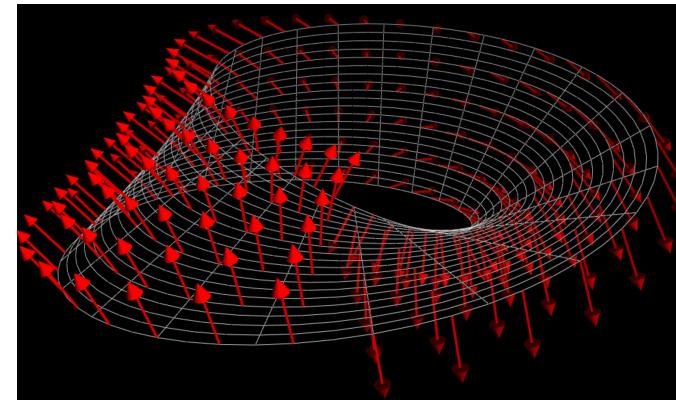
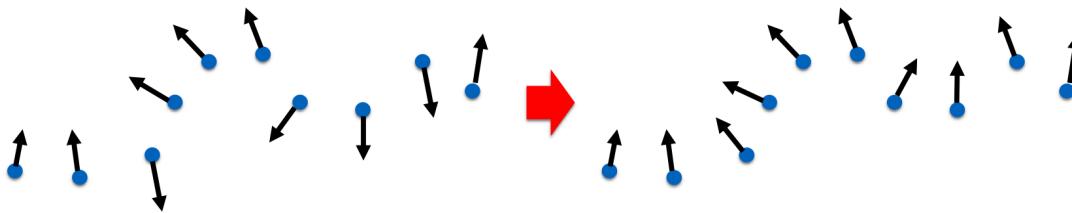
$$f(\mathbf{x}_i + \lambda \vec{\mathbf{n}}_i) = \lambda$$

$$f(\mathbf{x}_i - \lambda \vec{\mathbf{n}}_i) = -\lambda$$



Consistent Normals are Required

- We just assumed consistent normals
- Normal is typically required to build watertight meshes
- However, obtaining consistent normal orientation is non-trivial (discussions deferred to later).



Estimate Parameters

- Variables:
 - $n + l$ variables on ω_i (RBF coef.) and c_i (polynomial coef.)
- Solve a linear system of $3n + l$ equations
 - $3n$: from the point, inside, and outside
 - l : additional constraints to guarantee the smoothness and integrability of f

$$\begin{pmatrix} A & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = B \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}$$

$$A_{i,j} = \phi(|x_i - x_j|), \quad i, j = 1, \dots, N,$$
$$P_{i,j} = p_j(x_i), \quad i = 1, \dots, N, \quad j = 1, \dots, \ell.$$

Implementation Details

- Triharmonic basis functions: $\phi(r) = r^3$
 - Need its extrapolation ability
 - Should **not** use RBF with compact or local support (e.g., Gaussian density)
- Polynomial: third-order is practically good

Implementation Details

- Do not need to use all the input data points as RBF centers
 - Use a greedy algorithm to select a subset of points
- Noisy data
 - Exact interpolation?
 - Treat the linear equation as solving a linear square problem and add a smoothness term

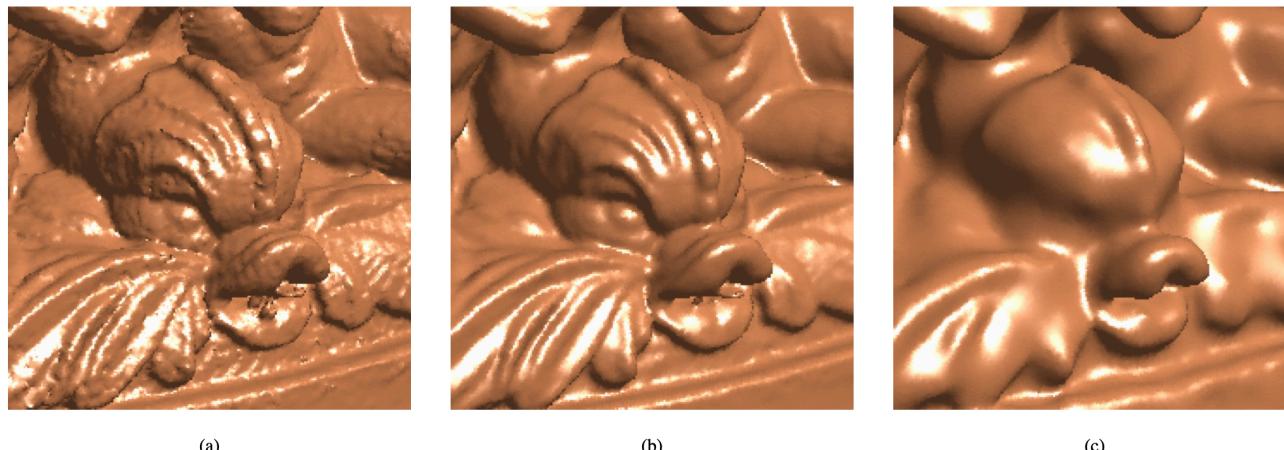
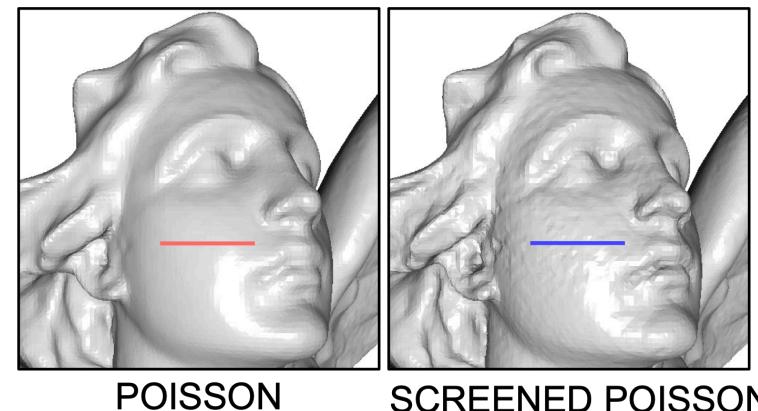


Figure 9: (a) Exact fit, (b) medium amount of smoothing applied (the RBF approximates at data points), (c) increased smoothing.

More than RBF

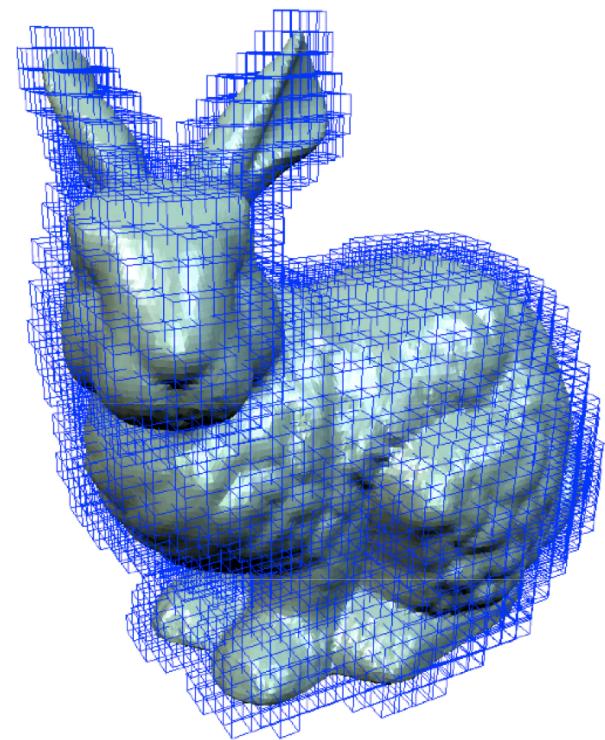
- Kazhdan M, Bolitho M, Hoppe H. “**Poisson surface reconstruction.**” ESGP, 2006.
 - Robust to noise, adapt to the sampling density
 - Over-smoothing
- Kazhdan M, Hoppe H. “**Screened poisson surface reconstruction.**” ToG, 2013.
 - Sharper reconstruction, faster
 - But it assumes clean data



Implicit Meshing Algorithm

- Two basic steps:
 1. Estimate an implicit field function from data
 2. **Extract the zero iso-surface**

Input: a signed distance field
(Implicitly assumed knowing the
inside/outside of the shape, often
needs to be estimated with
normal information)



Classical Solution: 2D Marching Square

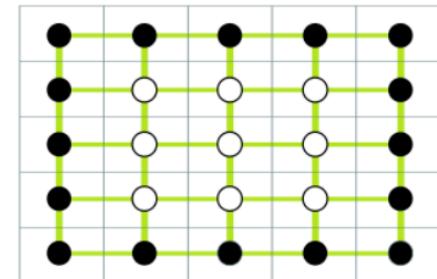
1	1	1	1	1
1	2	3	2	1
1	3	3	3	1
1	2	3	2	1
1	1	1	1	1

Threshold
with iso-value

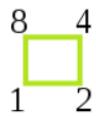


0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

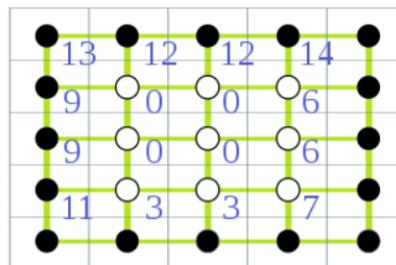
Binary image
to cells



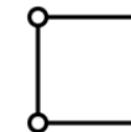
Classical Solution: 2D Marching Square



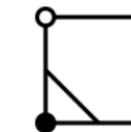
Give every cell a number based on which corners are true/false



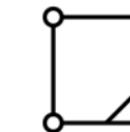
Look-up table contour lines



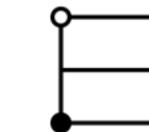
Case 0



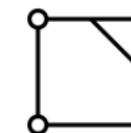
Case 1



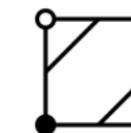
Case 2



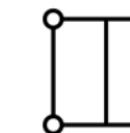
Case 3



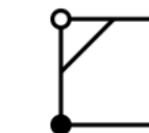
Case 4



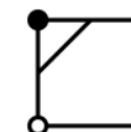
Case 5



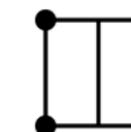
Case 6



Case 7



Case 8



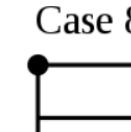
Case 9



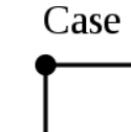
Case 10



Case 11



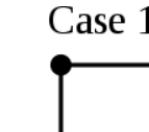
Case 12



Case 13

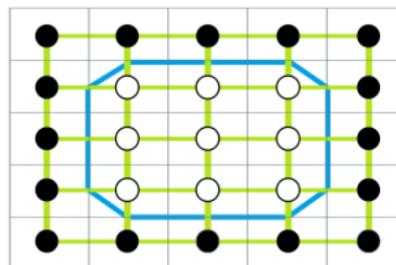


Case 14



Case 15

Look up the contour lines in the database and put them in the cells



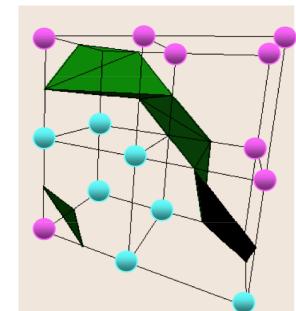
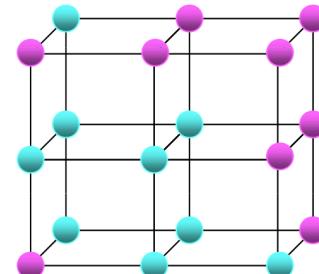
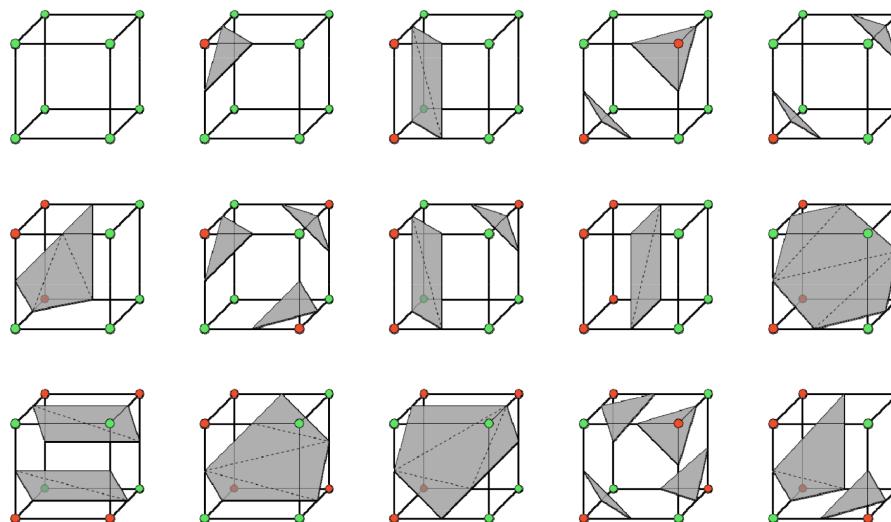
Look at the original values and use linear interpolation to determine a more accurate position of all the line end-points



1	1	1	1	1
1	2	3	2	1
1	3	3	3	1
1	2	3	2	1
1	1	1	1	1

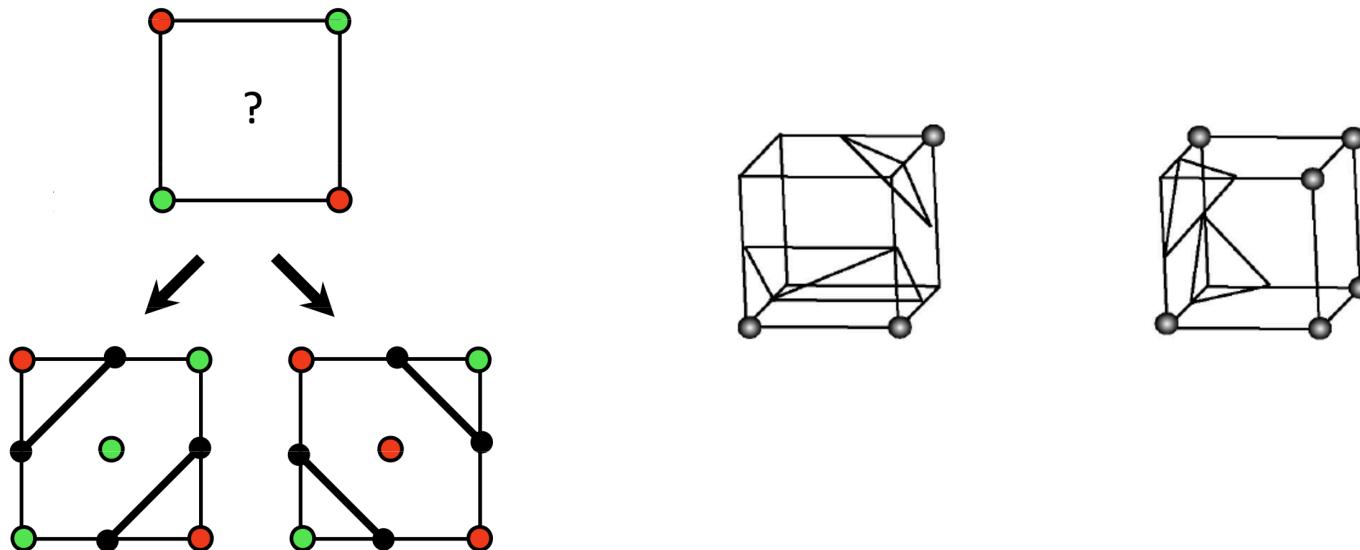
Classical Solution: 3D Marching Square

- $2^8 = 256$ cases
- The first published version exploits rotation and inversion, and only considers 15 unique cases:



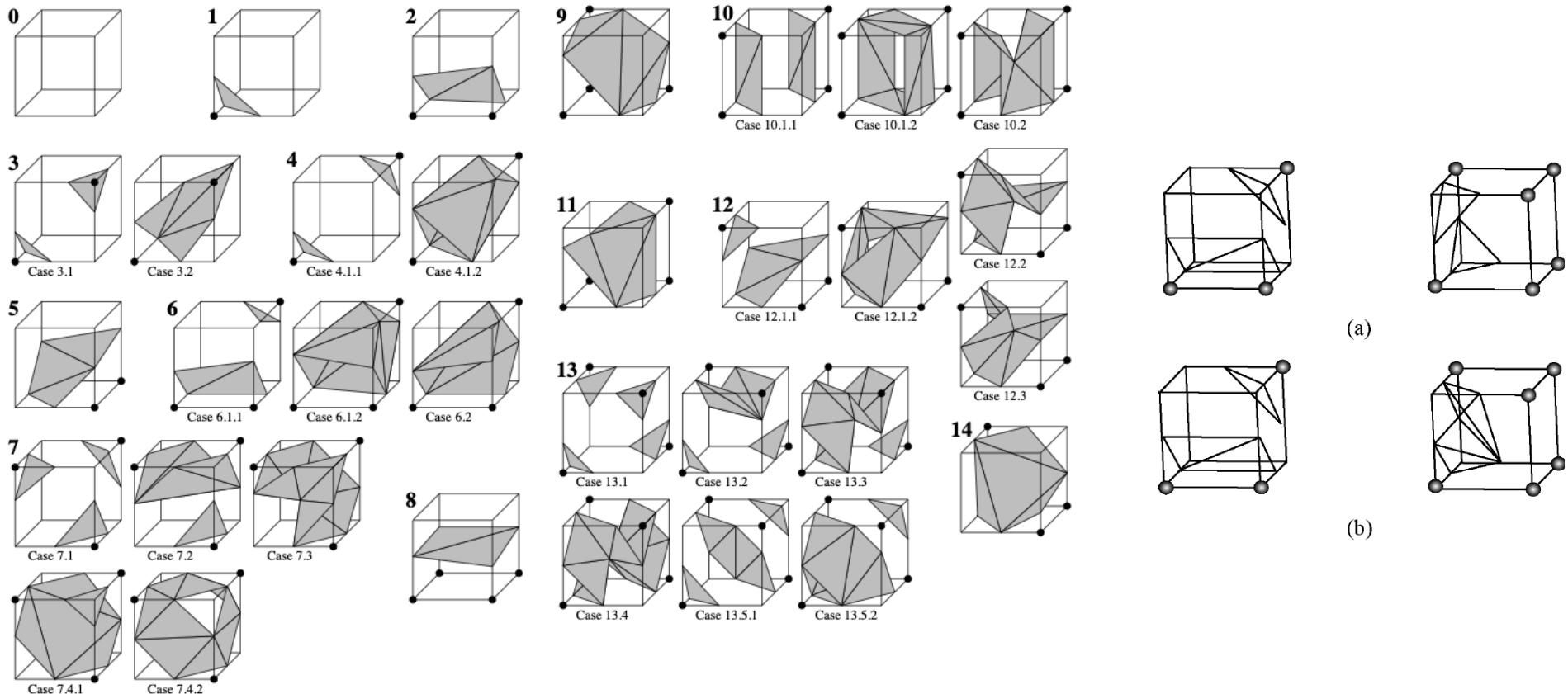
Ambiguity

- Ambiguity leads to holes:



Solution to Ambiguity

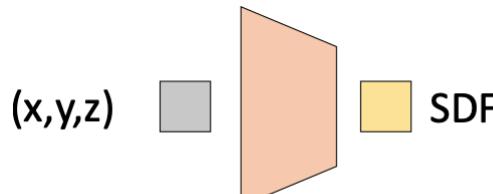
- Considering more cases in the look-up table by watching larger context:



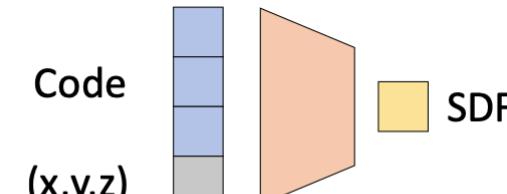
Comparison

	Explicit meshing (e.g., ball-pivoting)	Implicit meshing (e.g., RBF, Poisson)
Sensitive to normals	No	Yes
Watertight manifold	No	Yes, in most cases
Complexity	Linear	<ul style="list-style-type: none">• Large-scale equations to estimate implicit function• Marching cubes• Dense voxelization

Use Neural Network to Approximate Implicit Field Function

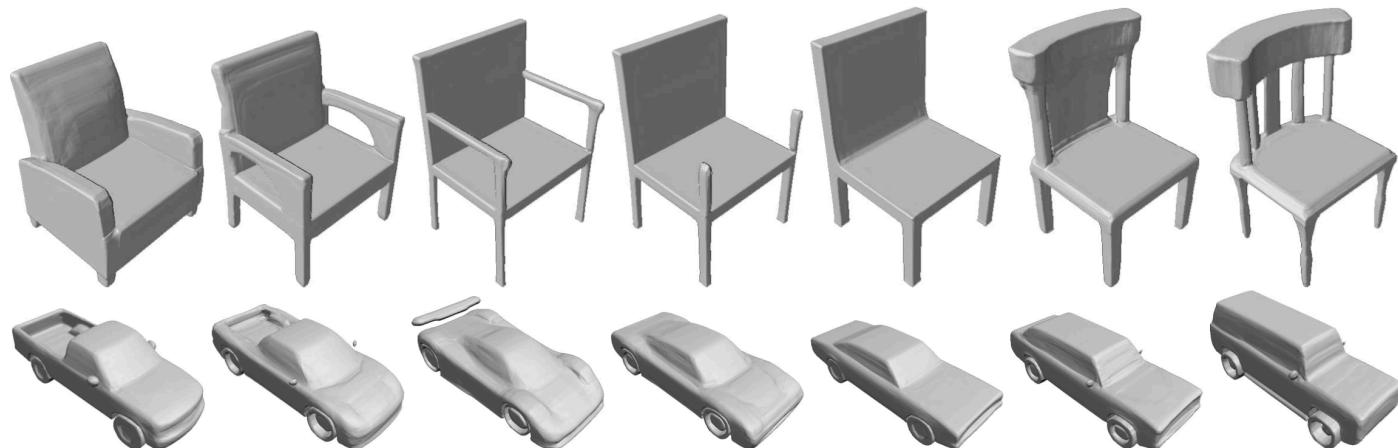


(a) Single Shape DeepSDF



(b) Coded Shape DeepSDF

- (a) use the network to overfit a single shape
- (b) use a latent code to represent a shape, so that the network can be used for multiple shapes



Consistent Orientation?

- Require the ground-truth *signed* implicit functions during training (e.g., signed distance, occupancy)
- However, 3D raw data, such as point cloud or triangle soup, are not necessarily consistently oriented
 - e.g., getting normal orientation for ShapeNet data is not easy

Recent Work: Sign Agnostic Learning of Shapes from Raw Data

- Unsigned distance is easy to obtain
 - Distance to the point cloud & triangle soup
- Learn ***signed*** distance from ***unsigned distance*** groundtruth
 - Require a special loss function

Sign Agnostic Learning

$$\text{loss}(\theta) = \mathbb{E}_{x \sim D_\chi} \tau(f(x; \theta), h_\chi(x))$$

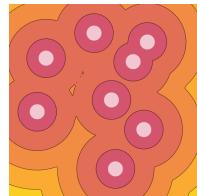
- $\chi \subset \mathbb{R}^3$: input raw data (e.g., a point cloud or a triangle soup)
- $f(x; \theta) : \mathbb{R}^3 \times \mathbb{R}^m \rightarrow \mathbb{R}$: learned signed function
- D_χ : distribution of the training samples defined by χ
- $h_\chi(x)$: some *unsigned* distance measure to χ
- $\tau : \mathbb{R} \times \mathbb{R}_+ \rightarrow \mathbb{R}$: a similarity function

Sign Agnostic Learning

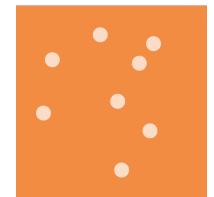
$$\text{loss}(\theta) = \mathbb{E}_{x \sim D_\chi} \tau(f(x; \theta), h_\chi(x))$$

- $h_\chi(x)$: some *unsigned* distance measure to χ

$$h_2(z) = \min_{x \in \mathcal{X}} \|z - x\|_2$$



$$h_0(z) = \begin{cases} 0 & z \in \mathcal{X} \\ 1 & z \notin \mathcal{X} \end{cases}$$



- $\tau : \mathbb{R} \times \mathbb{R}_+ \rightarrow \mathbb{R}$: a similarity function

$$\tau_\ell(a, b) = |a - b|^\ell$$

Two Local Minima

$$\text{loss}(\theta) = \mathbb{E}_{x \sim D_\chi} \tau(f(x; \theta), h_\chi(x))$$

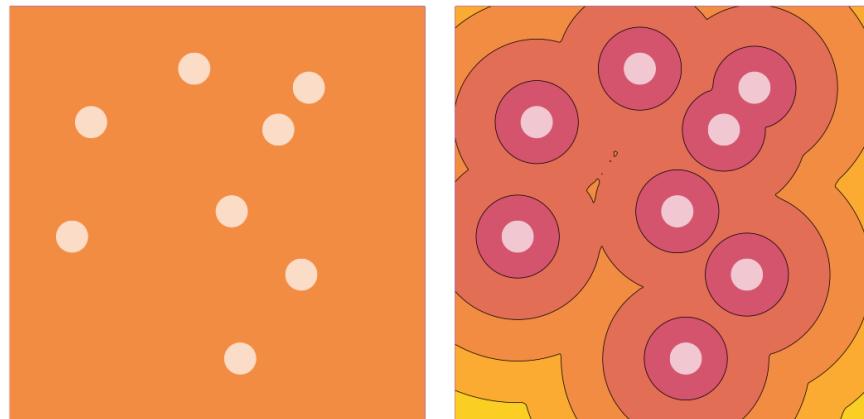
- f is an ***unsigned*** function that resembles $h_\chi(x)$
- f is a ***signed*** function and $|f|$ resembles $h_\chi(x)$
- We prefer the second case to use marching cube

Two Local Minima

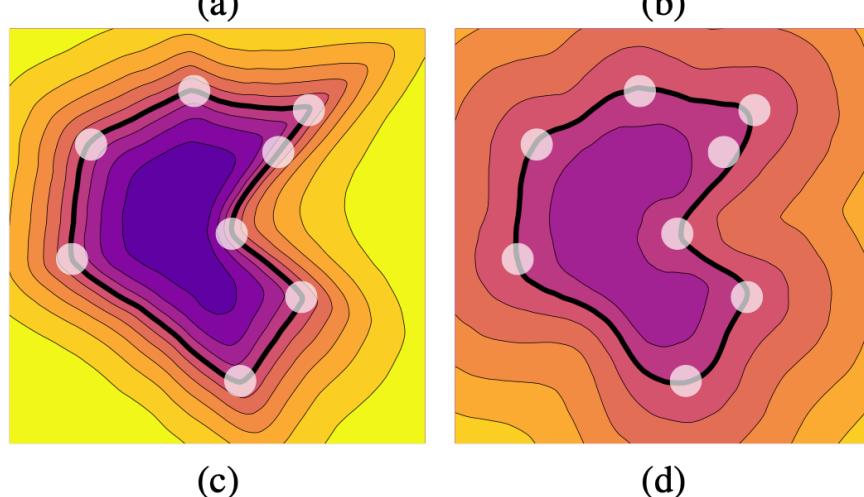
- Pick a special weight initialization θ^0 so that $f(x; \theta^0) \approx \varphi(\|x\| - r)$, where $\varphi(\|x\| - r)$ is the signed distance function to an r -radius sphere
 - $f > 0$ if $\|x\| > r$
 - $f < 0$ if $\|x\| < r$
- Under such an initialization, f is not easy to converge to the unsigned local minima.

2D Results

Unsigned function
as supervision



Learned signed
function



Surface Reconstruction Results from Raw Data

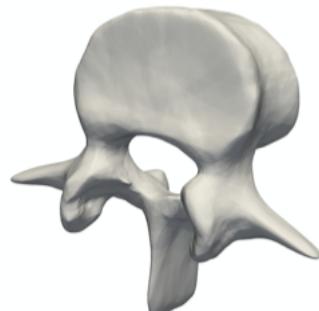
Raw Point Cloud



Ball-Pivoting Algorithm



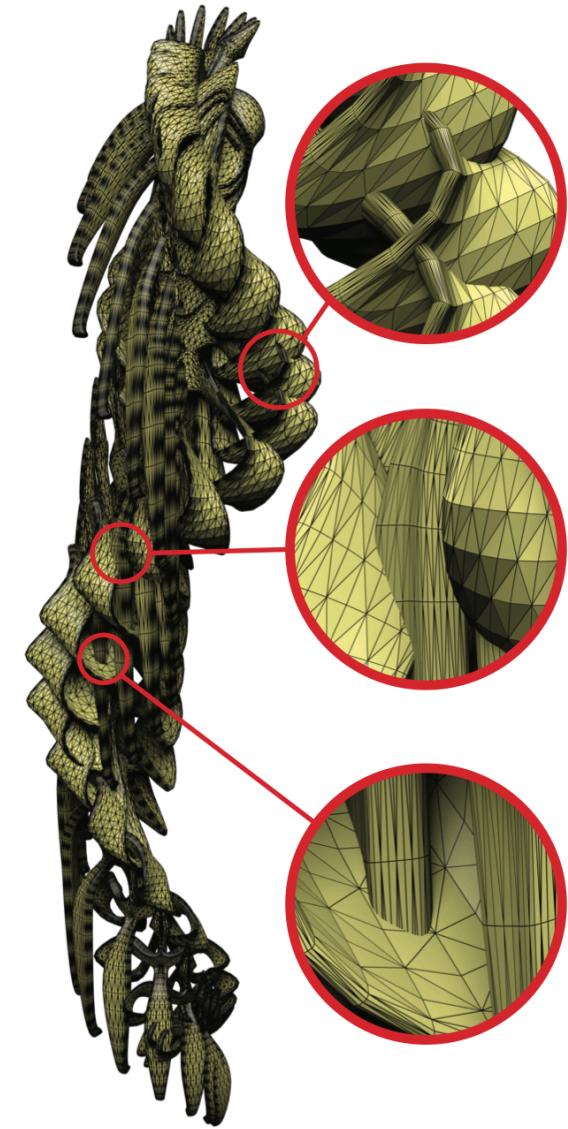
SAL



A Peek into Mesh Processing

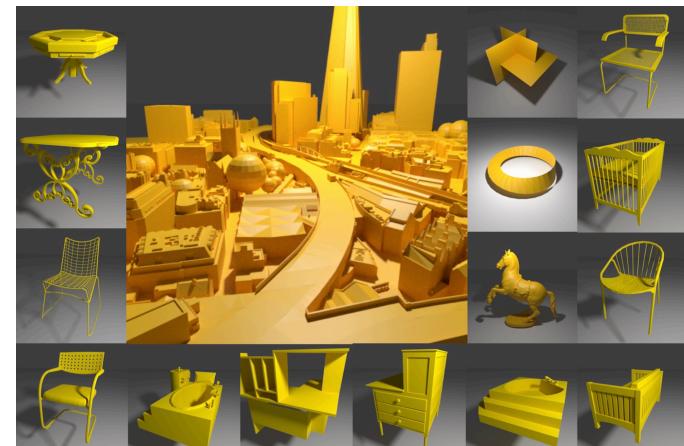
Watertight Manifold Surface Generation

- Many downstream tasks require the input mesh to be a watertight manifold.
- However, meshes designed by artists or reconstructed by some algorithms are often non-manifold:
 - open boundaries
 - self-intersections
 - incorrect connectivity
 - ambiguous face orientation
 - double surfaces



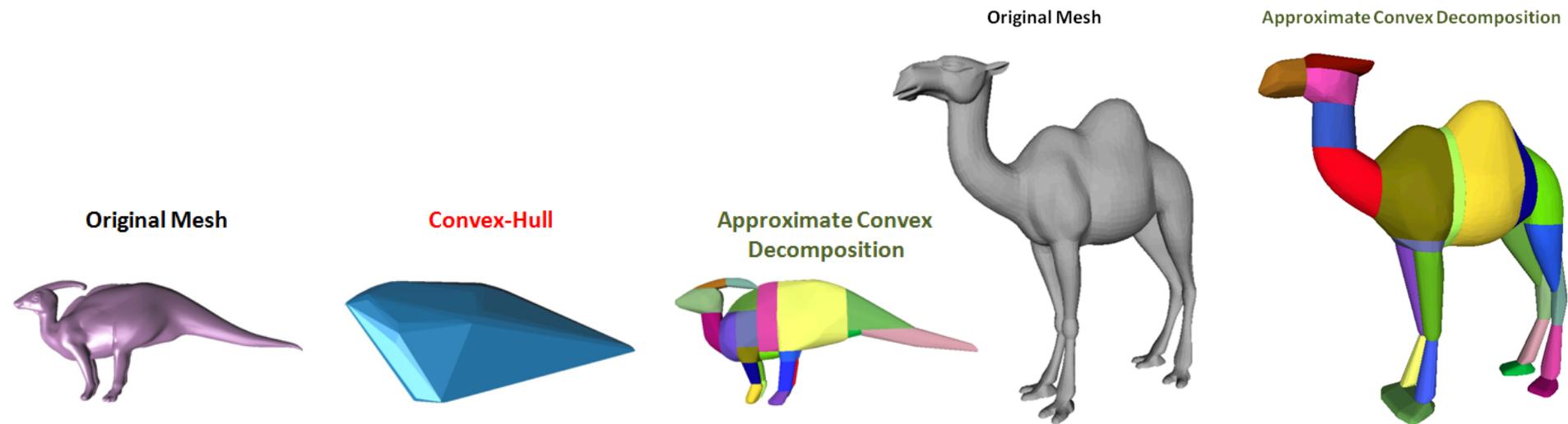
Watertight Manifold Surface Generation

- Huang J, Su H, Guibas L. “**Robust watertight manifold surface generation method for shapenet models.**”
 - Small artifacts, all surface have volume
- Huang J, Zhou Y, Guibas L. “**ManifoldPlus: A Robust and Scalable Watertight Manifold Surface Generation Method for Triangle Soups.**”
 - Less artifacts, may contain zero-volume structures
- Basic steps:
 - Voxelize the surface
 - Extracts exterior faces between occupied voxels and empty voxels
 - Projection-based optimization



Convex Decomposition

- In some applications (e.g., collision detection), convex shapes is an input requirement.
- Directly using convex-hull provides poor approximation for concave surfaces.
- Exact convex decomposition is NP-hard and may produce a high number of primitives.



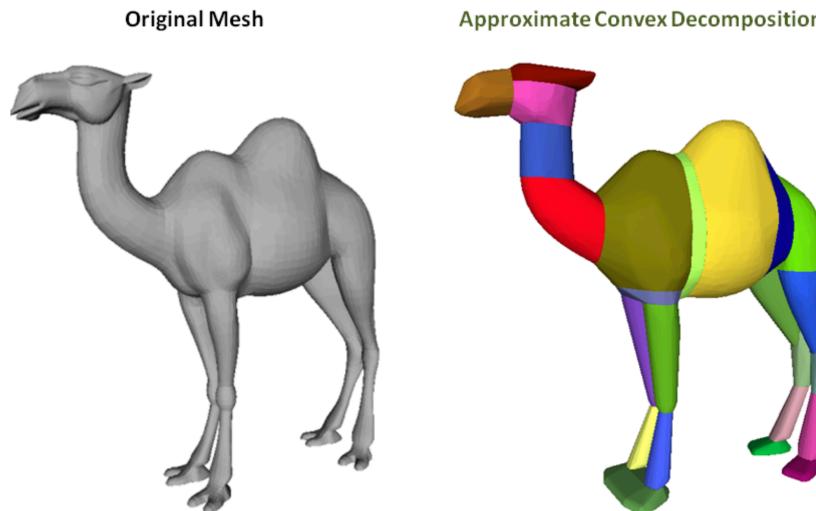
Approximate Convex Decomposition

- Goal: partition the mesh into a minimal number of clusters while ensuring that each cluster has a concavity metric lower than a pre-defined threshold

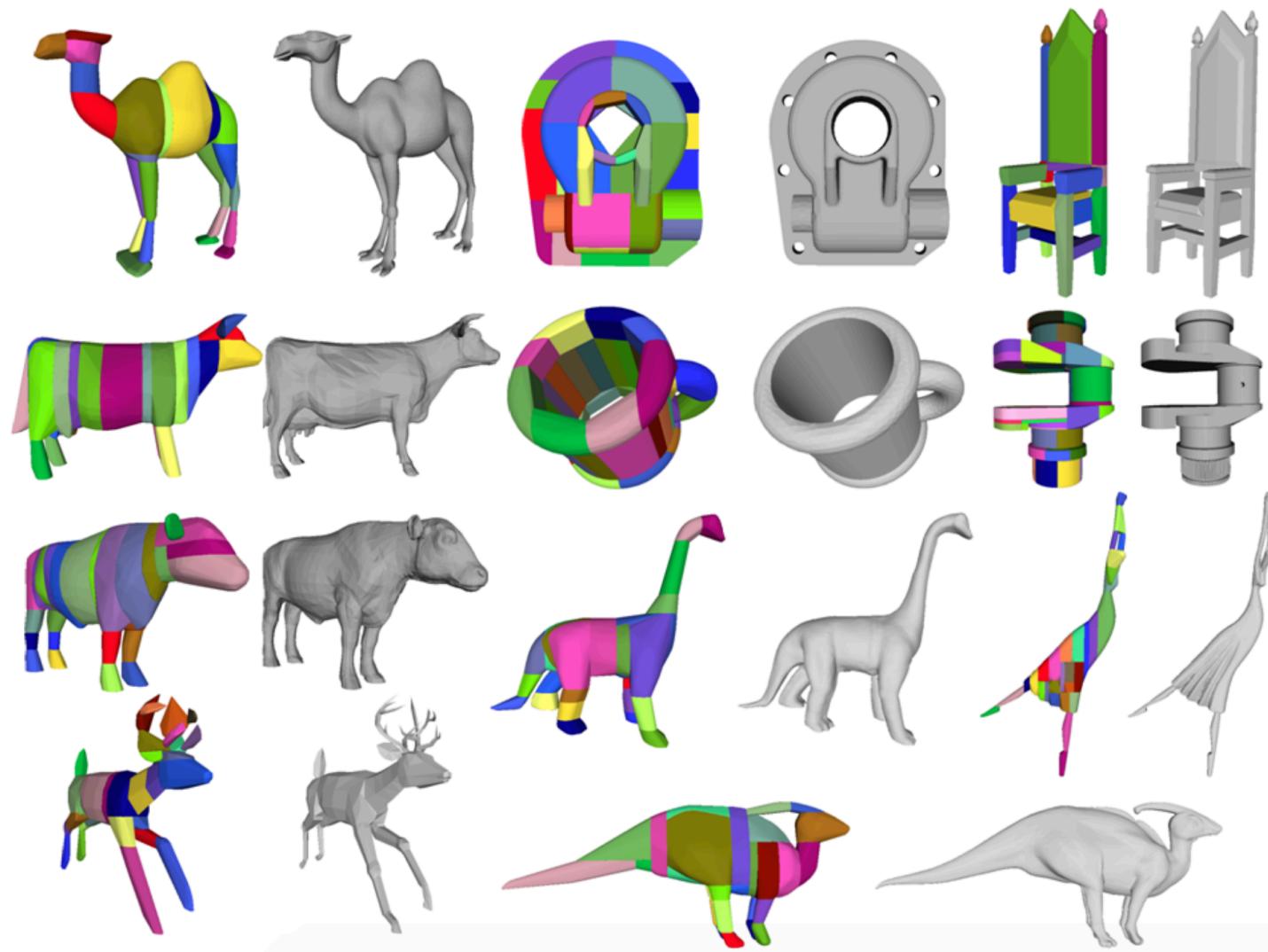
V-HACD

- First voxelize the shape
- Then use a heuristic algorithm (e.g., cut by axis-aligned plane) to segment the shape into parts
- The segmentation optimizes a volume-based concavity measure:

$$\frac{V(ConvexHull(Sub_1)) + V(ConvexHull(Sub_2)) - V(original)}{V(original)}$$

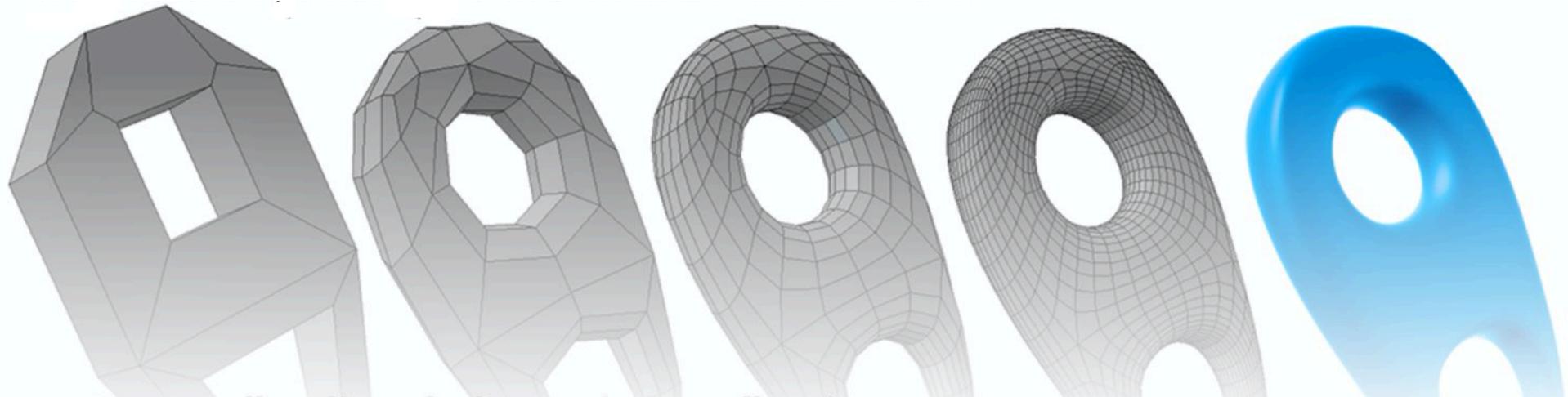
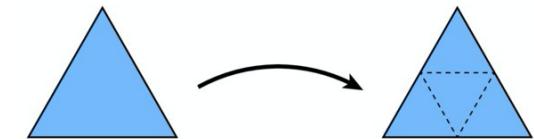


V-HACD Results



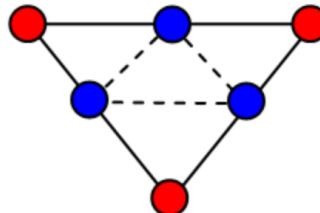
Mesh Upsampling via Subdivision

- Coarse mesh: memory efficient
- Fine-grained mesh: smooth, details
- Subdivision: coarse mesh -> fine-grained mesh
- Recursive upsampling:
 - Divide mesh element (one triangle becomes three)
 - Calculate new positions of the mesh vertices

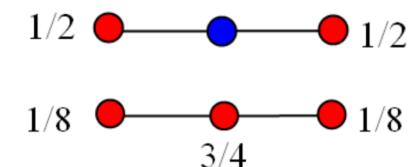
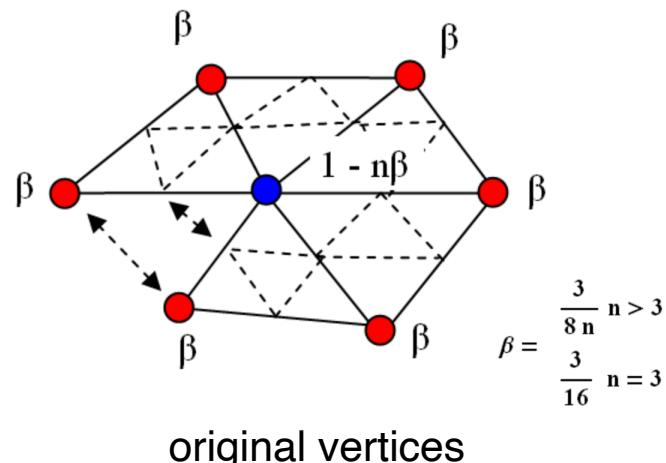
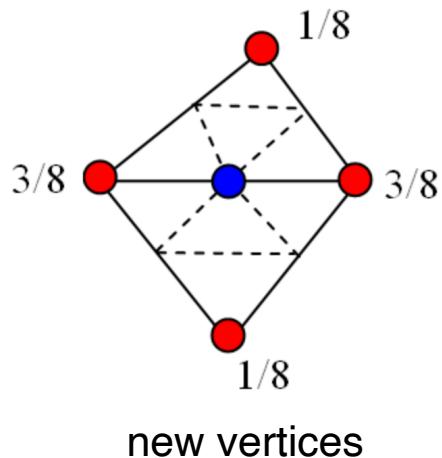


Loop Subdivision

- Divide mesh element (one triangle becomes four)



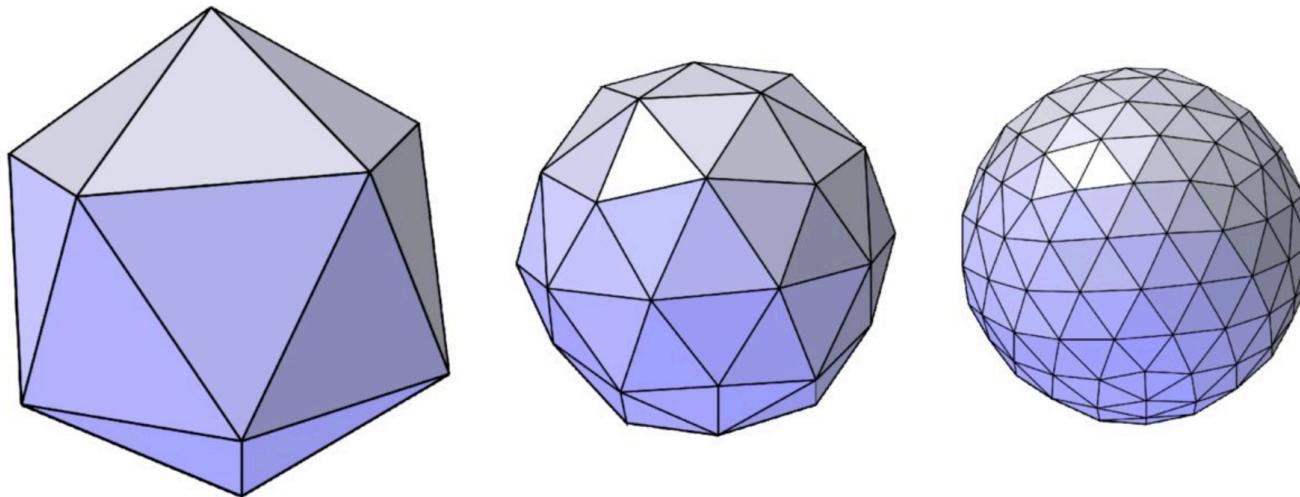
- Calculate new positions of the mesh vertices (linear combinations of original positions)



boundary cases

Loop Subdivision

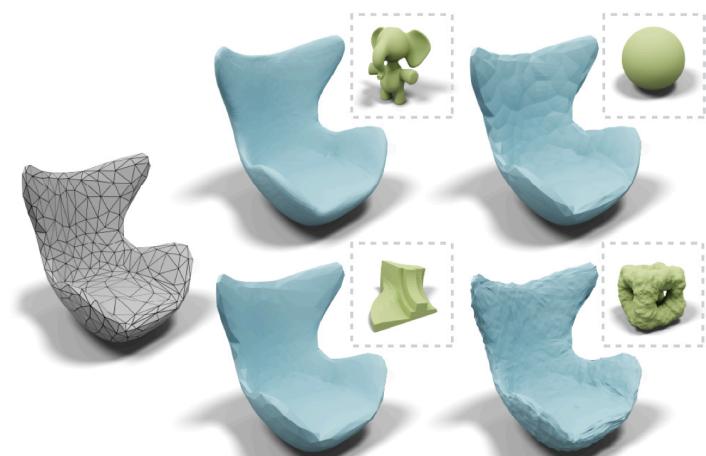
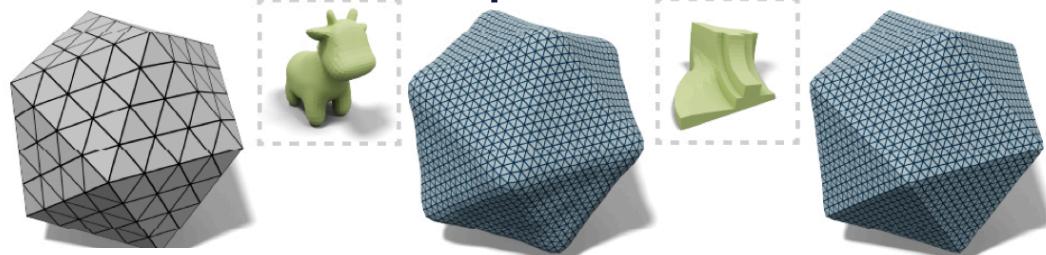
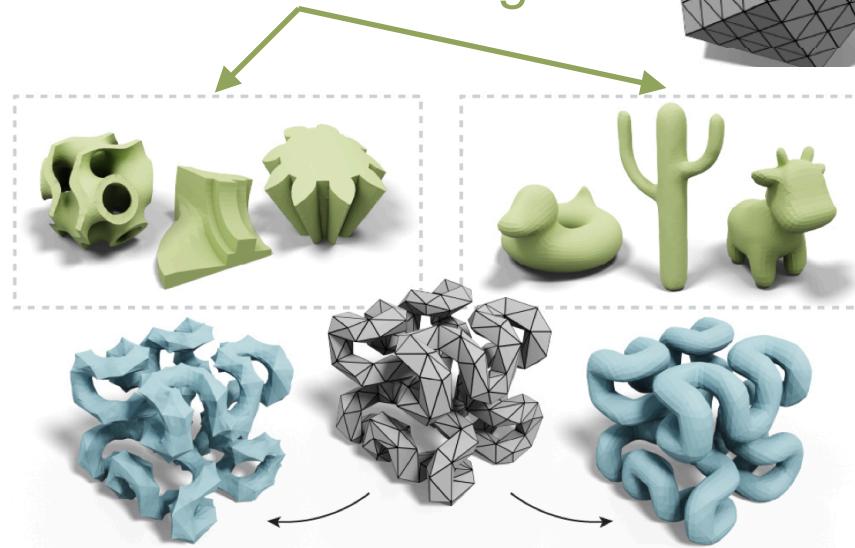
- Common subdivision rule
- “C2” smoothness of its limit shape
- Approximating, not interpolating



Neural Subdivision

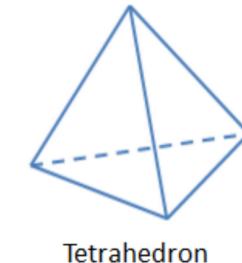
- Using network to predict vertex positions conditioned on the local geometry enables us to learn complex non-linear subdivision scheme (more than smoothing).
- Data-driven: training on different shapes can learn different styles

Two modalities of training data

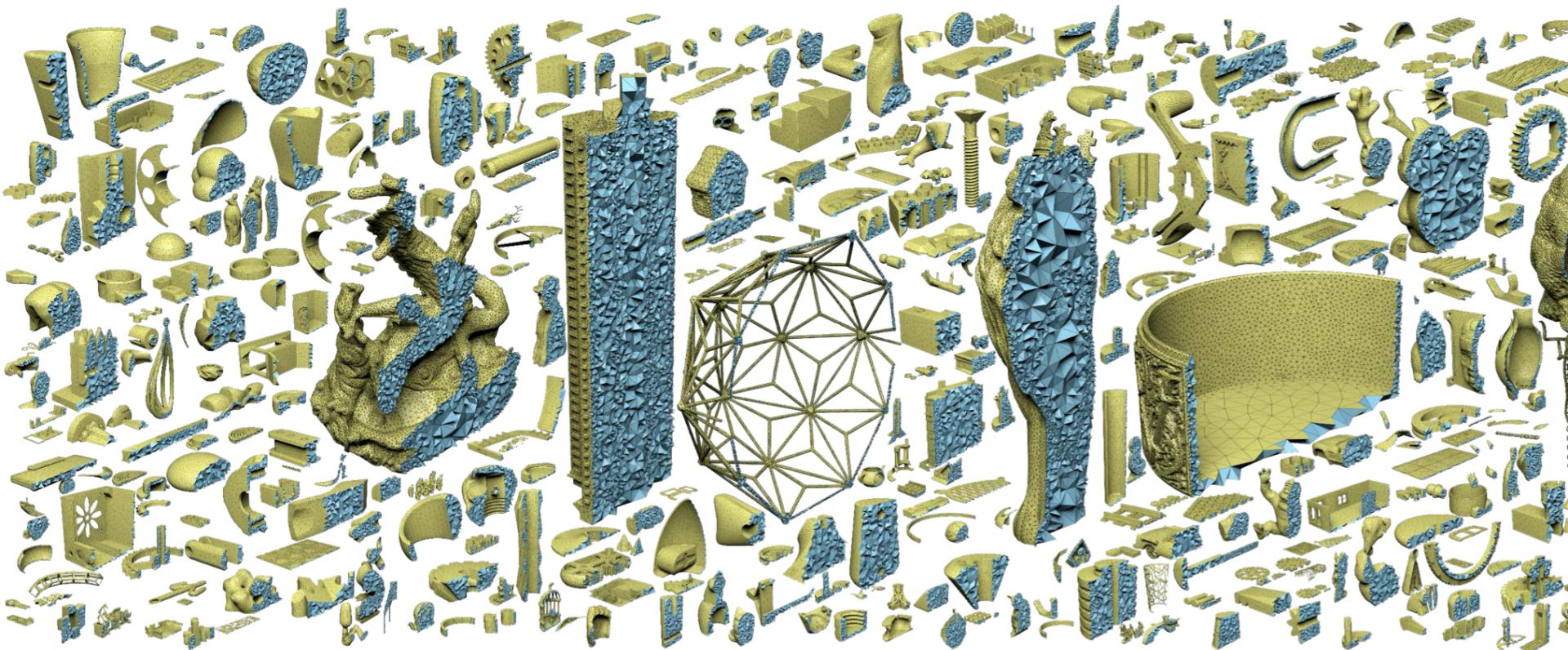


Tetrahedral Meshing

- Triangle mesh -> tetrahedral mesh
- Surface (triangle) -> Volume (tetrahedron)
- Finite element method



Tetrahedron



Tetrahedral Meshing

- Hu Y, Zhou Q, Gao X, Jacobson A, Zorin D, Panozzo D. “**Tetrahedral meshing in the wild**”. Siggraph 2018
- Hu Y, Schneider T, Wang B, Zorin D, Panozzo D. “**Fast tetrahedral meshing in the wild.**” Siggraph 2020

