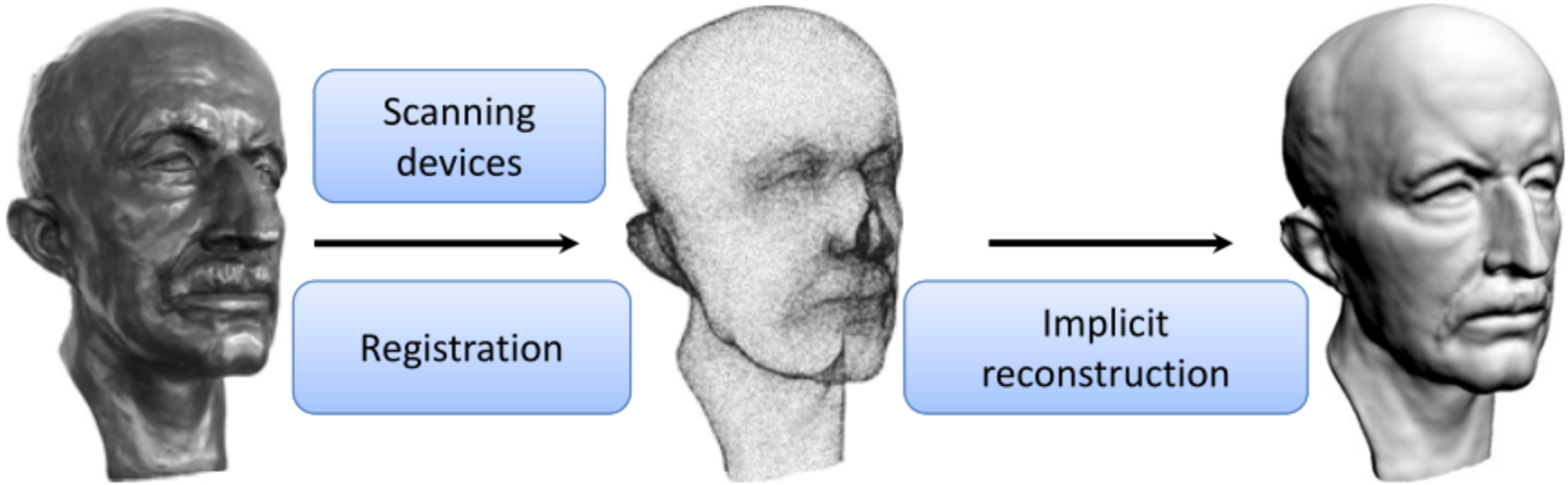
Surface Reconstruction: Part II

Surface Reconstruction

Today:



physical model

acquired point cloud

reconstructed model

Input Data

- Set of range scans
  - Each scan is a regular quad- or tri-mesh
  - Normal vectors are well defined
  - Scans registered in common coordinate system

- Set of irregular sample points
  - Typically without normal vectors
  - More general

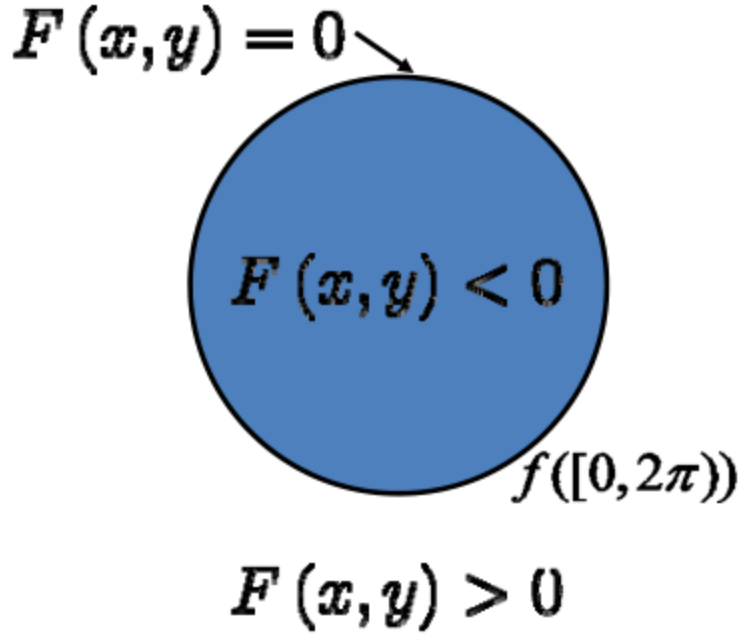
Shape Representations

- Explicit representation
  - Image of parametrization

$$f(t) = (x(t), y(t)) = (r\cos(t), r\sin(t))$$

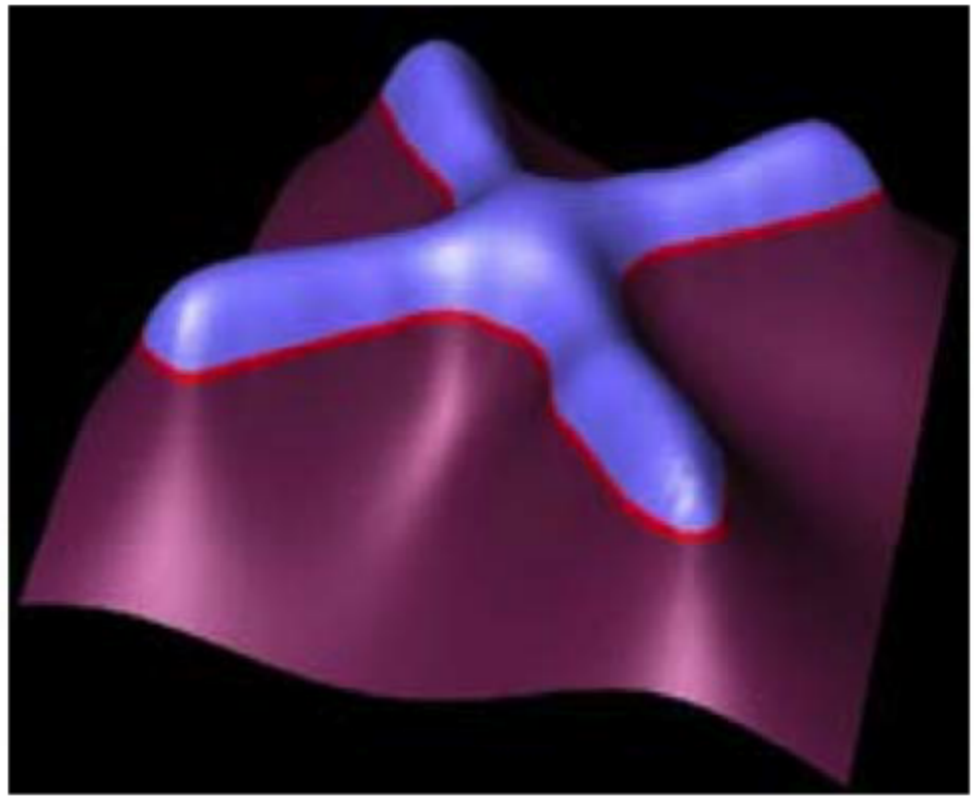
- Implicit representation
  - Zero set of distance function

$$F(x,y) = \sqrt{x^2 + y^2} - r$$



Implicit Representations

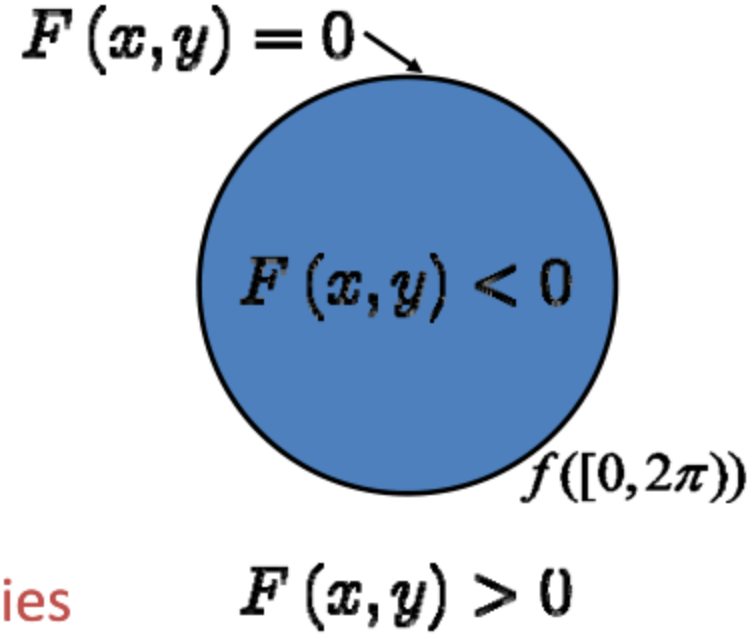
- Level set of 2D function defines 1D curve



$$F(x, y) = 0$$

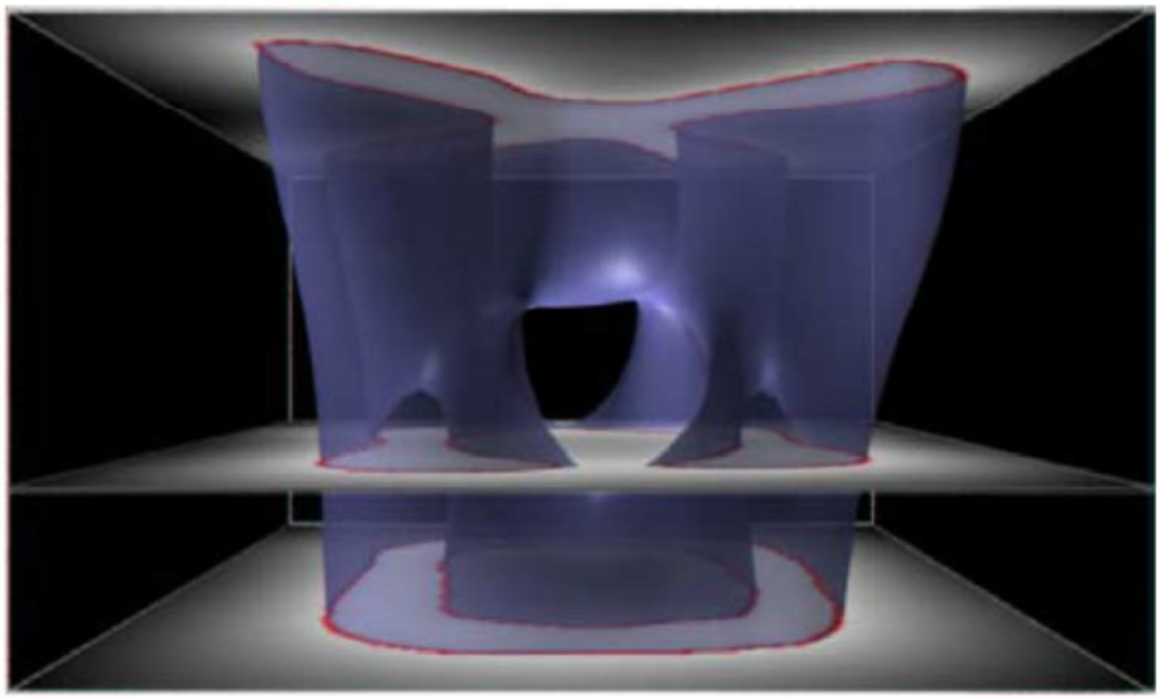
Explicit / Implicit

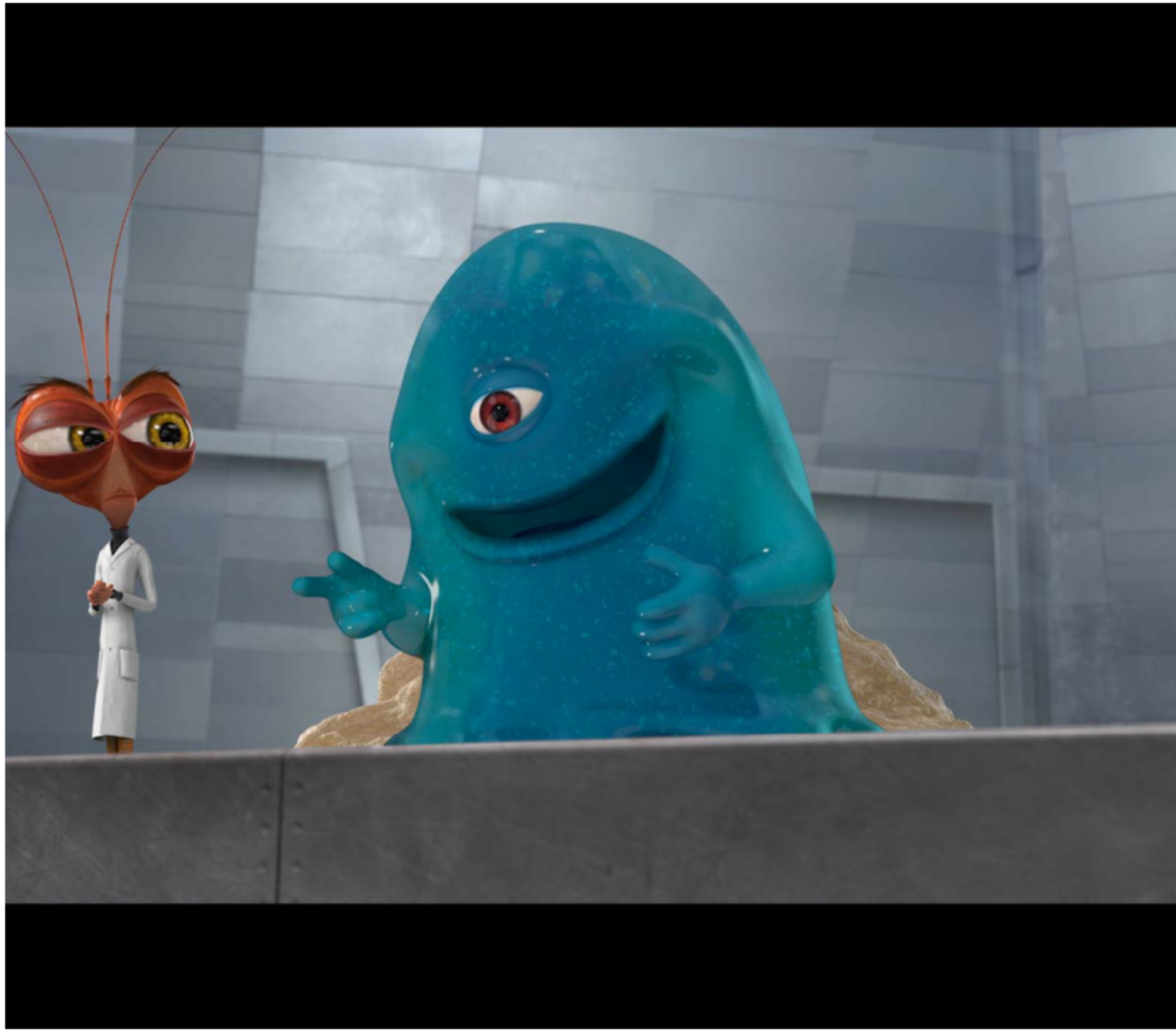
- Explicit representation
  - Image of parameterization
  - Easy to find points on shape
  - Can defer problems to param domain
- Implicit representation
  - Zero set of distance function
  - Easy in/out/distance test
  - Easy to handle different topologies



Implicit Representations

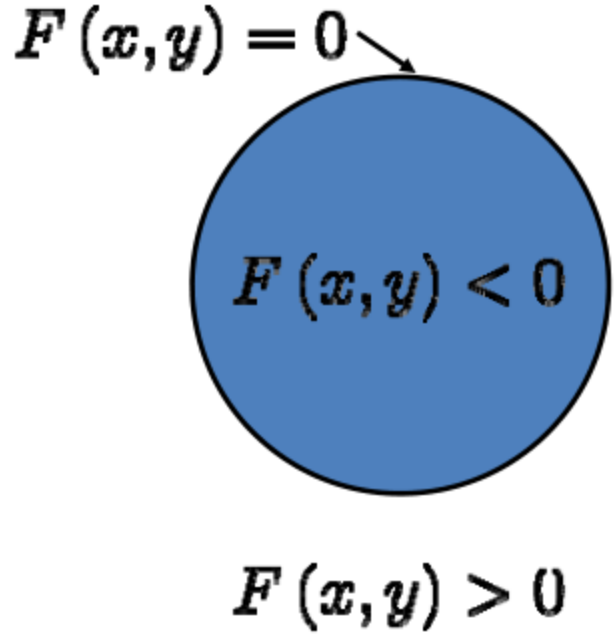
- Easy to handle different topologies





Implicit Representations

- General implicit function:
  - Interior: $F(x,y,z) < 0$
  - Exterior: $F(x,y,z) > 0$
  - Surface: $F(x,y,z) = 0$
- Special case
  - Signed distance function (SDF)

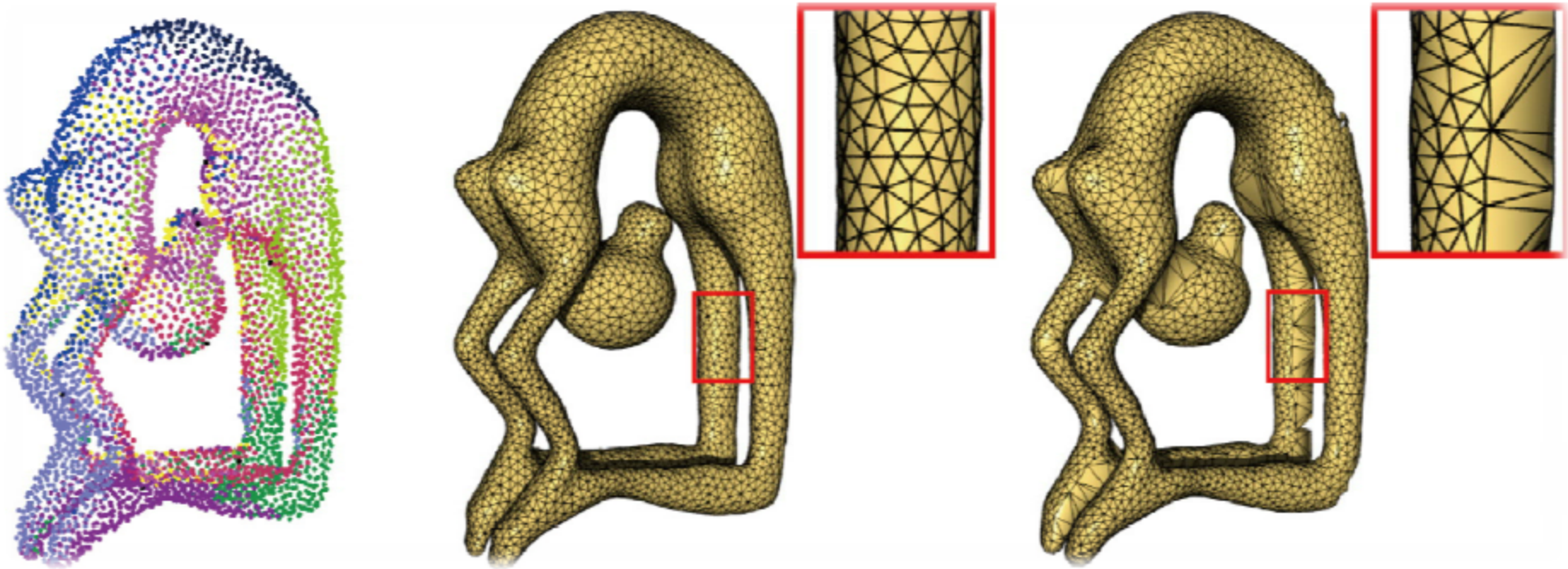


Explicit Reconstruction Methods

- Connect sample points by triangles
- Exact interpolation of sample points
- Bad for noisy or misaligned data
- Can lead to holes or non-manifold situations

Implicit Reconstruction Methods

- Estimate signed distance function (SDF)
- Extract zero iso-surface
- Approximation of input points
- Watertight manifold results by construction
  - Can have spurious components

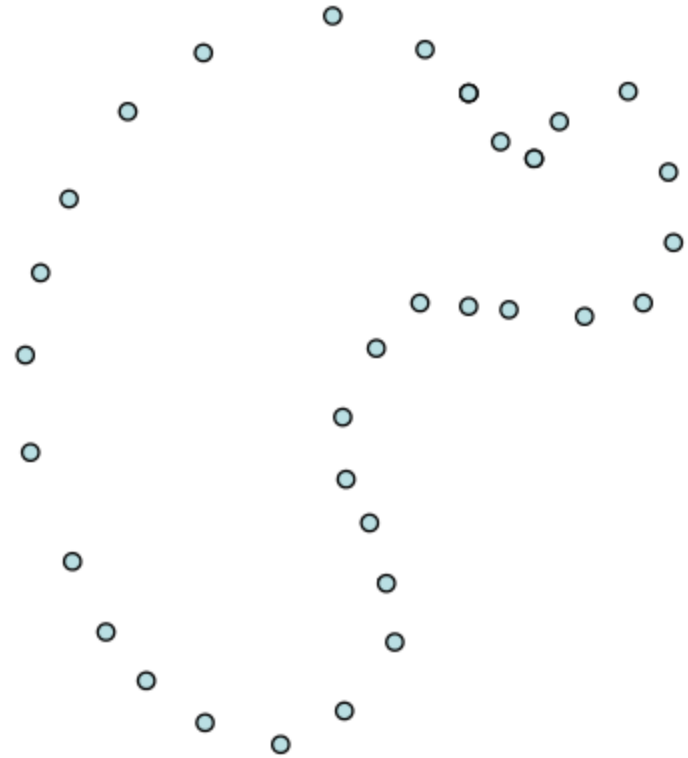


Input

Implicit

Explicit

Implicit Function Approach

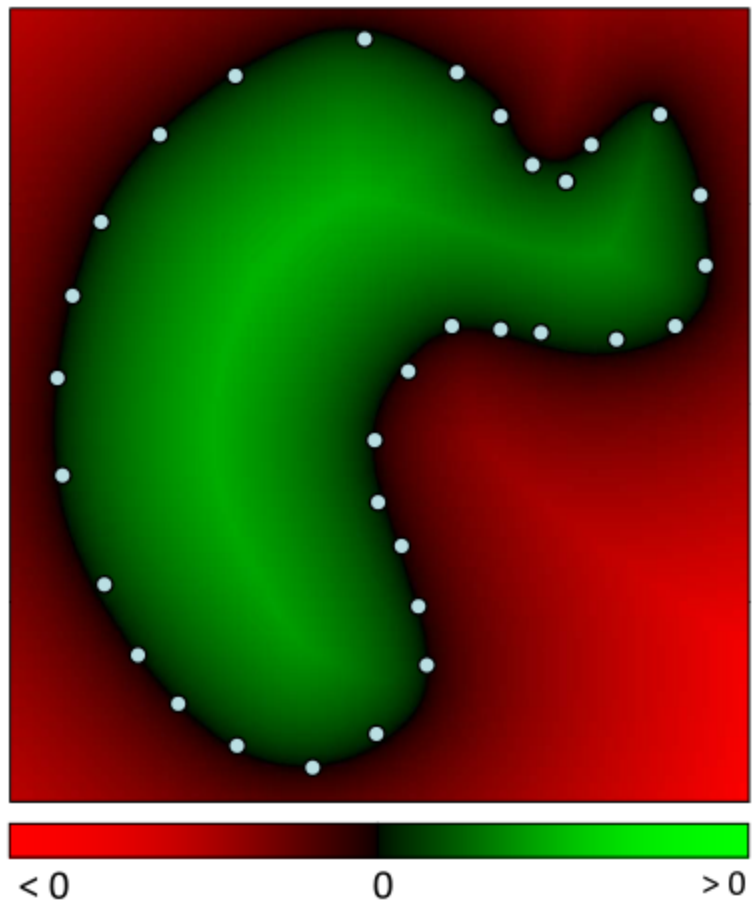


Implicit Function Approach

- Define a function

$$f : R^3 \rightarrow R$$

with value < 0 outside the shape and > 0 inside



Implicit Function Approach

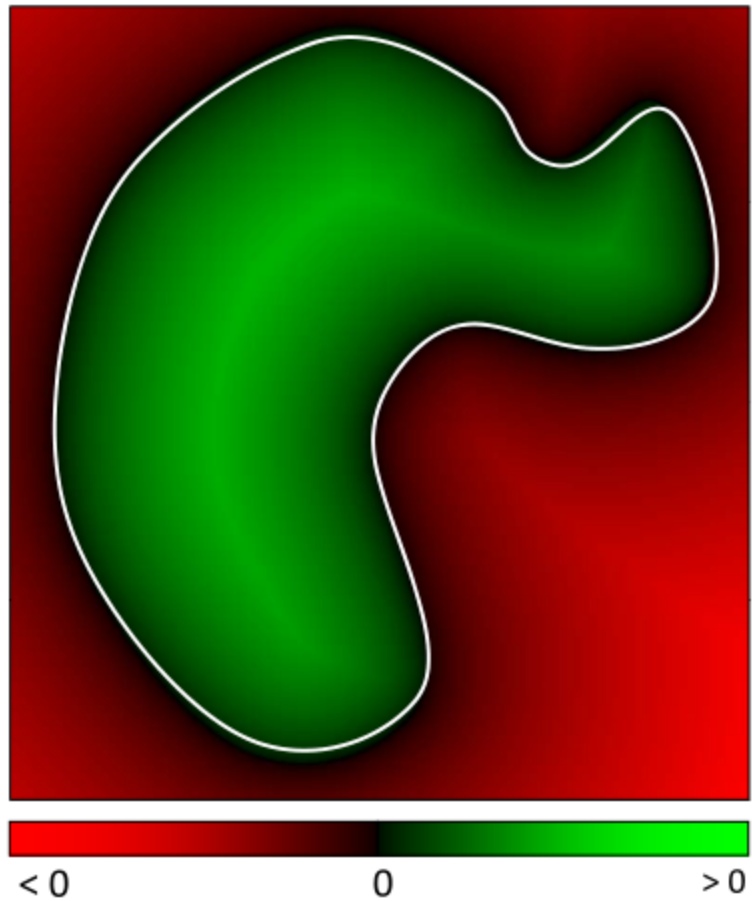
- Define a function

$$f : R^3 \rightarrow R$$

with value < 0 outside the shape and > 0 inside

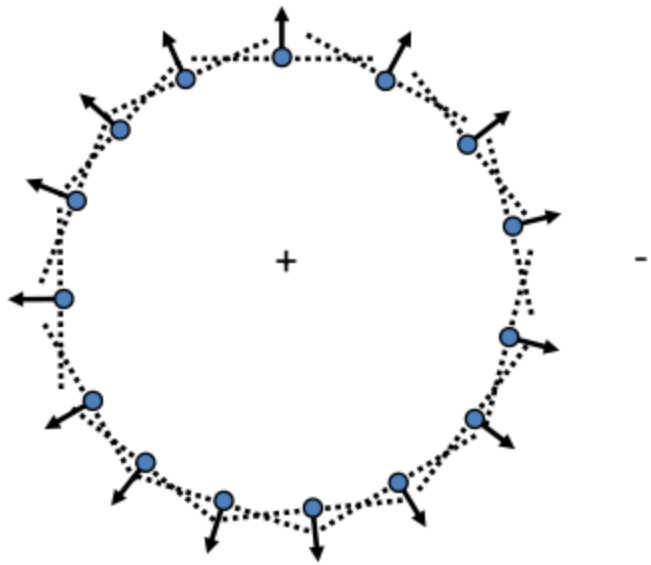
- Extract the zero-set

$$\{x : f(x) = 0\}$$



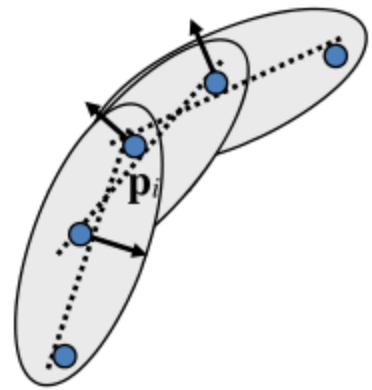
Signed Distance Function

- Construct SDF from point samples
  - Distance to points is not enough
  - Need inside/outside information
  - Reconstruct normal vectors first



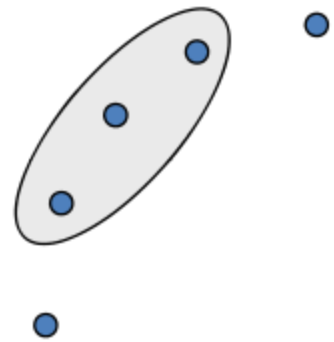
Normal Estimation

- To find normal $\mathbf{n}_i$ for each sample point $\mathbf{p}_i$
  1. Examine local neighborhood for each point
     - Set of k nearest neighbors
  2. Compute best approximating tangent plane
     - Covariance analysis
  3. Determine normal orientation
     - MST propagation



Normal Estimation

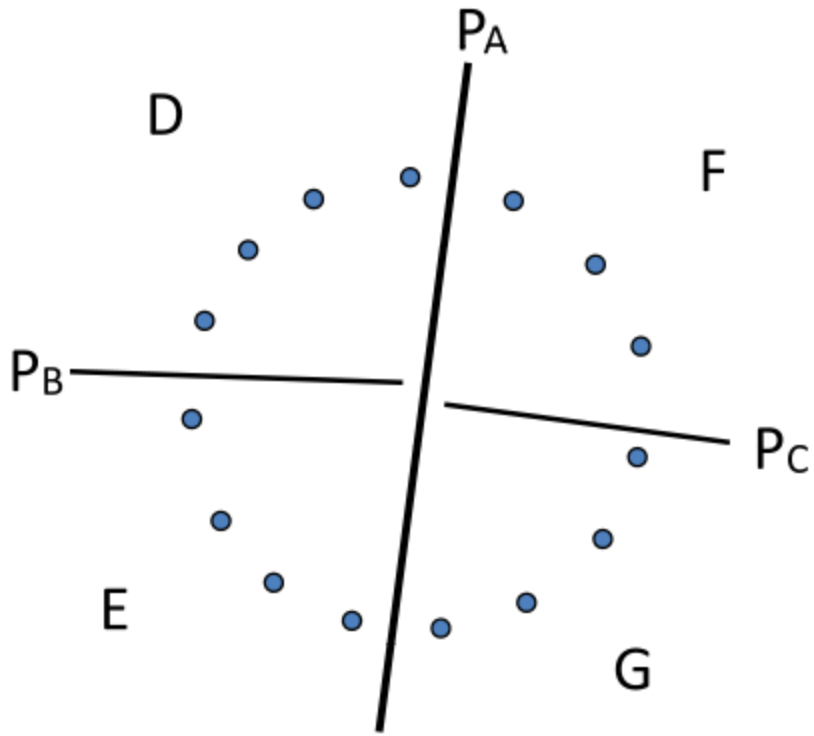
- To find normal $\mathbf{n}_i$ for each sample point $\mathbf{p}_i$
  1. Examine local neighborhood for each point
     - Set of k nearest neighbors
  2. Compute best approximating tangent plane
     - Covariance analysis
  3. Determine normal orientation
     - MST propagation

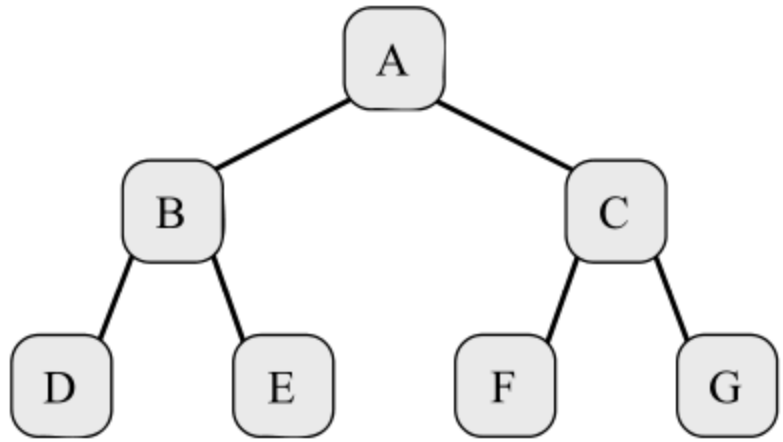


Local Neighborhood

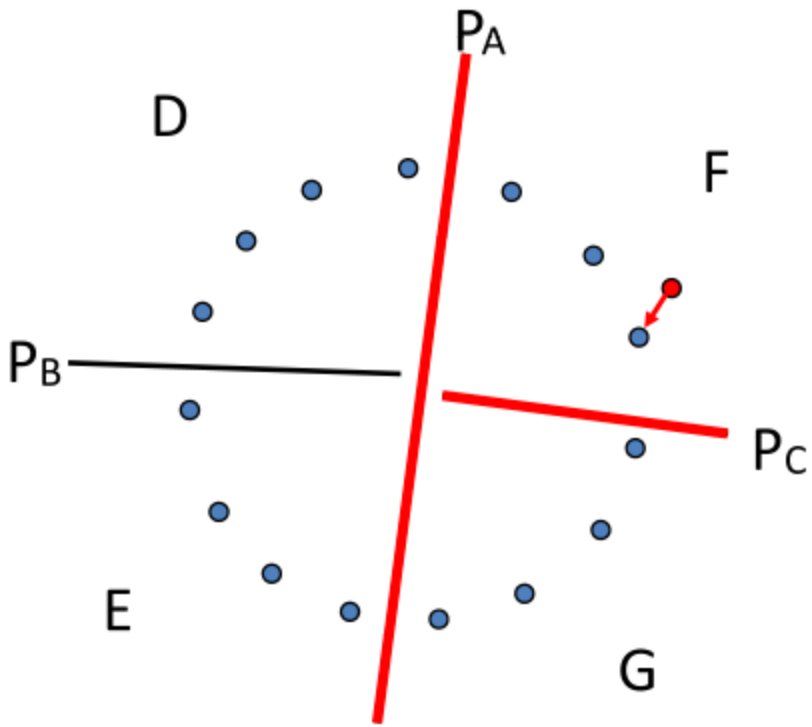
- Find k nearest neighbors (kNN) of a point
  - Brute force: $O(n)$ complexity
- Use BSP tree
  - Binary space partitioning tree
  - Recursively partition 3D space by planes
  - Tree should be balanced, put plane at median
  - $log(n)$ tree levels, complexity $O(log\ n)$

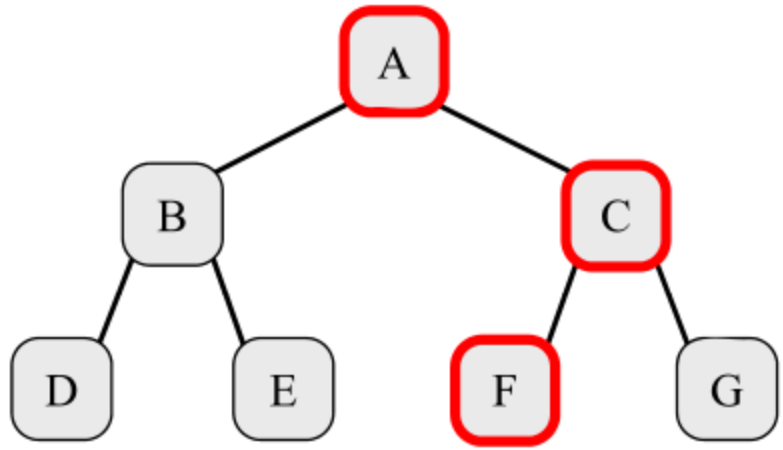
BSP Closest Points



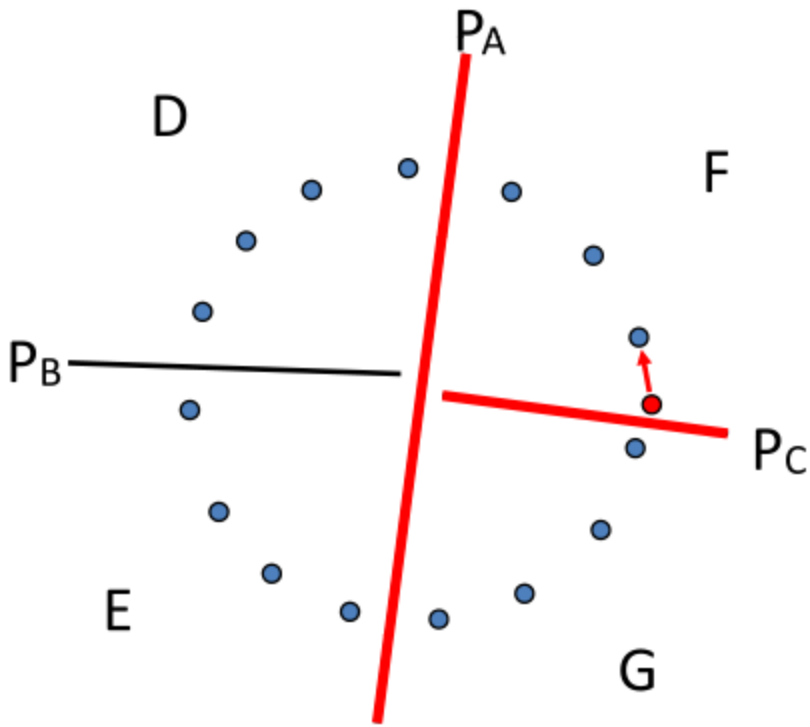


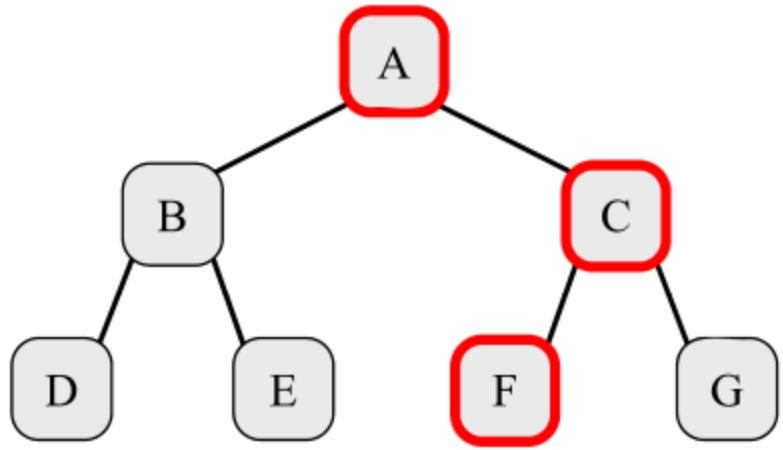
BSP Closest Points



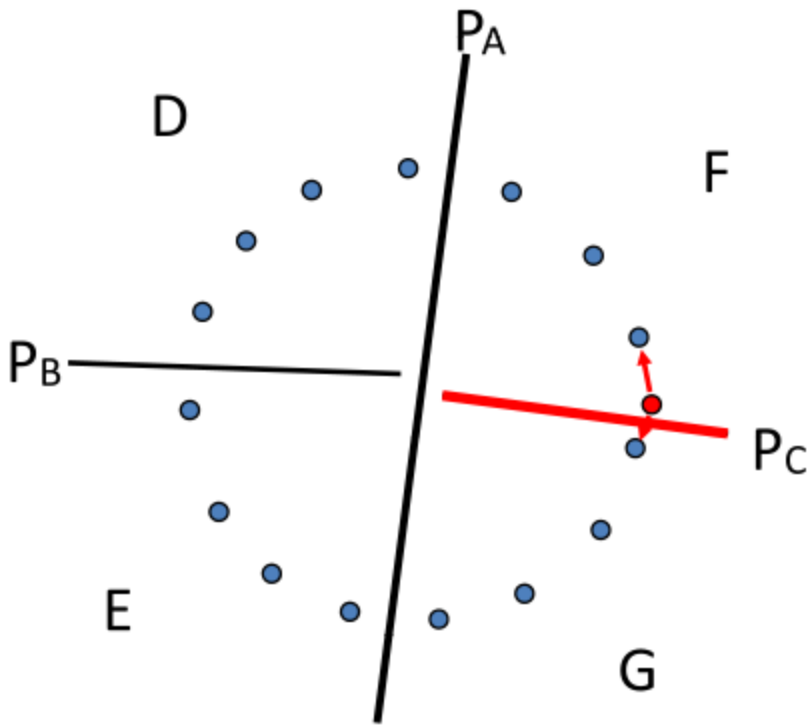


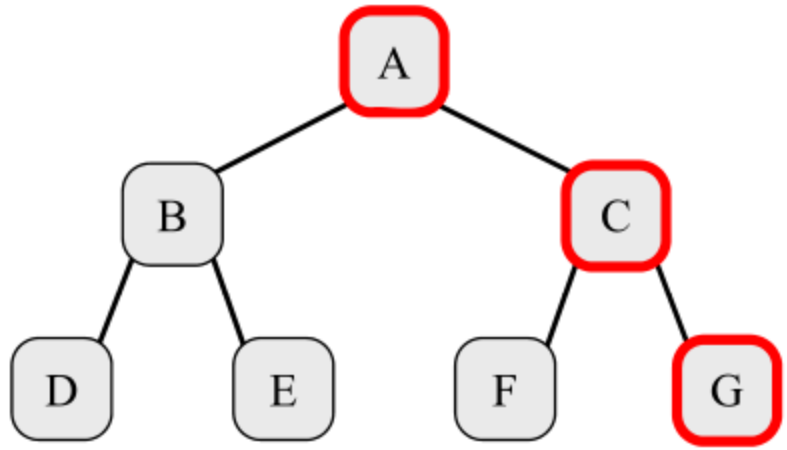
BSP Closest Points



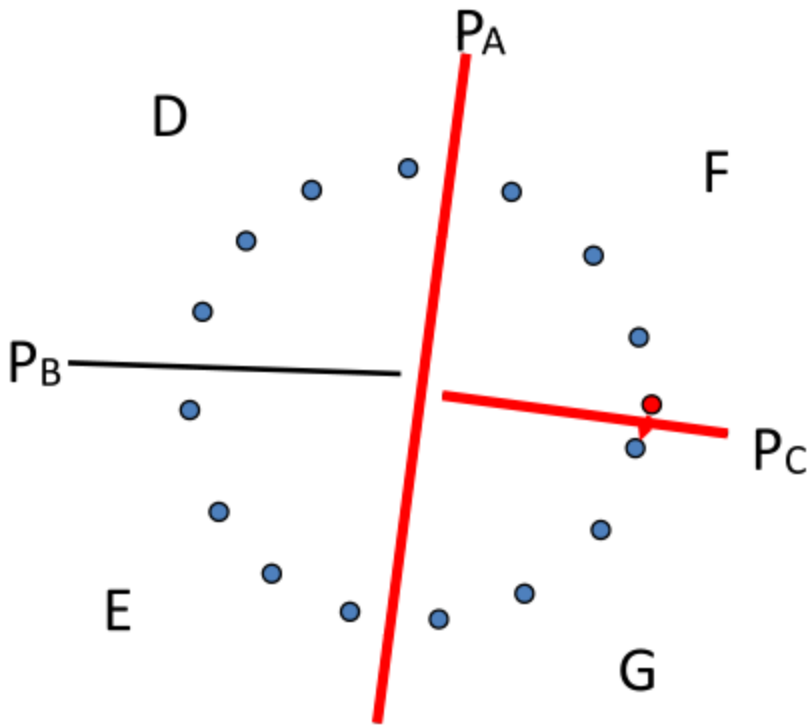


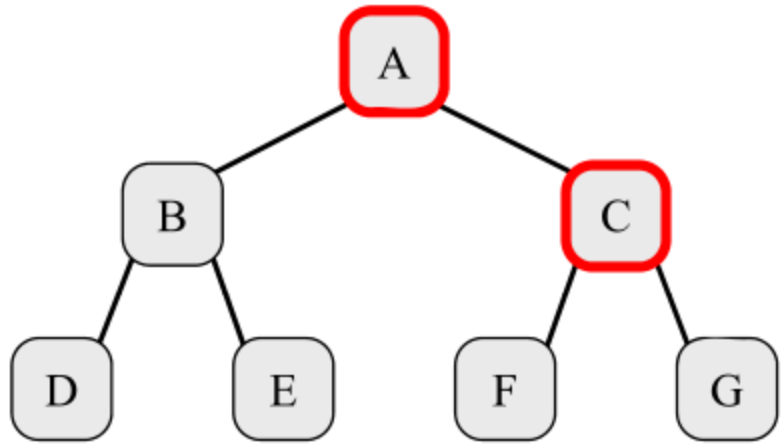
BSP Closest Points





BSP Closest Points

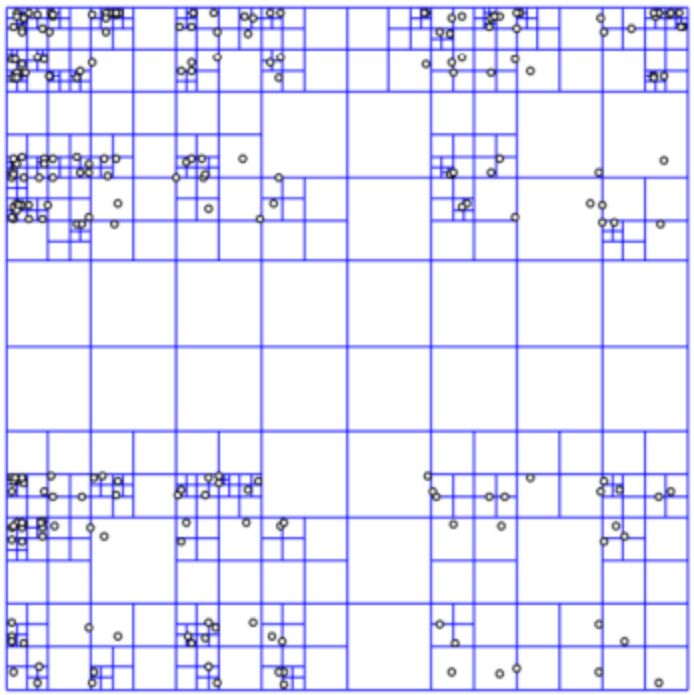




BSP Closest Points

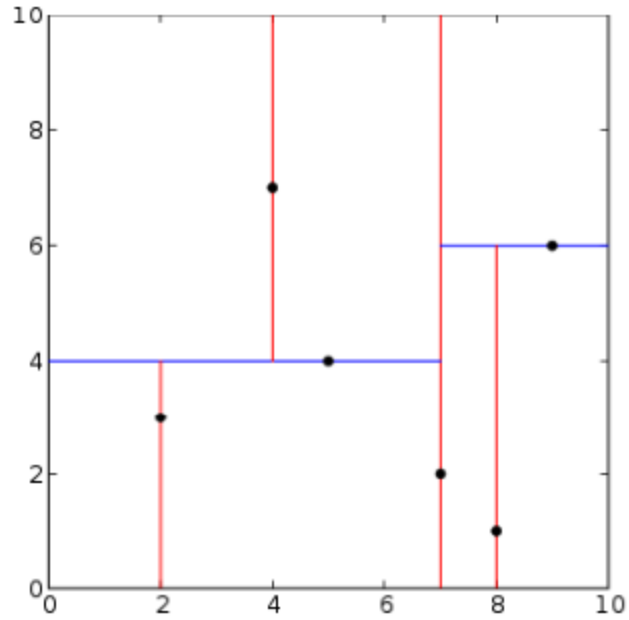
```
Node::dist(Point x, Scalar& dmin)
{
 if (leaf_node())
   for each sample point p[i]
     dmin = min(dmin, dist(x, p[i]));

 else
 {
   d = dist_to_plane(x);
   if (d < 0)
   {
     left_child->dist(x, dmin);
     if (|d| < dmin) right_child->dist(x, dmin);
   }
   else
   {
     right_child->dist(x, dmin);
     if (|d| < dmin) left_child->dist(x, dmin);
   }
 }
}
```

More Trees



Quad-tree (oct-tree)

Cells are squares (cubes)

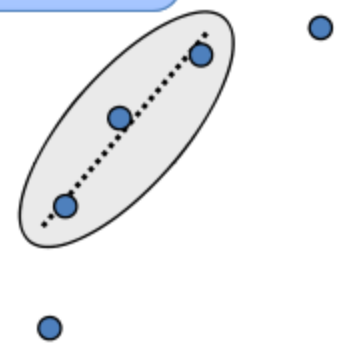


Kd-tree

Cells are axis-aligned boxes

Normal Estimation

- Find normal $\mathbf{n}_i$ for each sample point $\mathbf{p}_i$
  1. Examine local neighborhood for each point
     - Set of k nearest neighbors
  2. Compute best approximating tangent plane
     - Covariance analysis
  3. Determine normal orientation
     - MST propagation



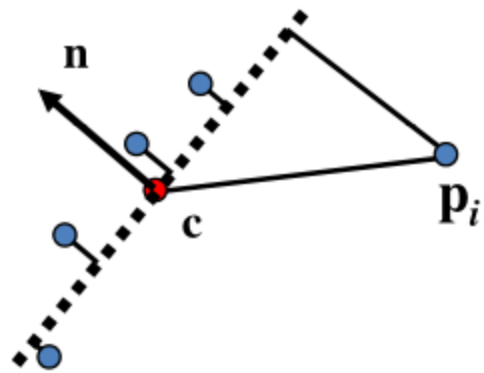
Principal Component Analysis

- Fit a plane with center $\mathbf{c}$ and normal $\mathbf{n}$ to a set of points $\{\mathbf{p}_1, \ldots, \mathbf{p}_m\}$
- Minimize least squares error

$$\min_{\mathbf{c},\mathbf{n}} \sum_{j=1}^{m} \left(\mathbf{n}^T(\mathbf{p}_j - \mathbf{c})\right)^2$$

- Subject to non-linear constraint

$$\|\mathbf{n}\| = 1$$



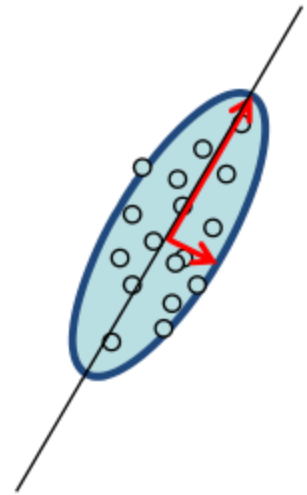
Principal Component Analysis

- Plane center is barycenter of points

$$\mathbf{c} = \bar{\mathbf{p}} =: \frac{1}{m}\sum_{j=1}^{m}\mathbf{p}_j$$

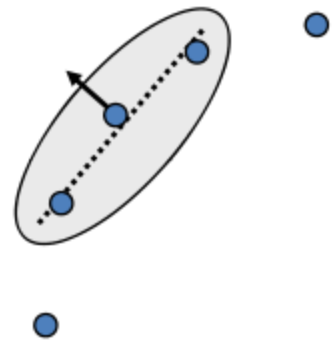
- Normal is eigenvector w.r.t. smallest eigenvalue of point covariance matrix

$$\sum_{j=1}^{m}(\mathbf{p}_j - \bar{\mathbf{p}})(\mathbf{p}_j - \bar{\mathbf{p}})^T \in \mathbb{R}^{3\times 3}$$

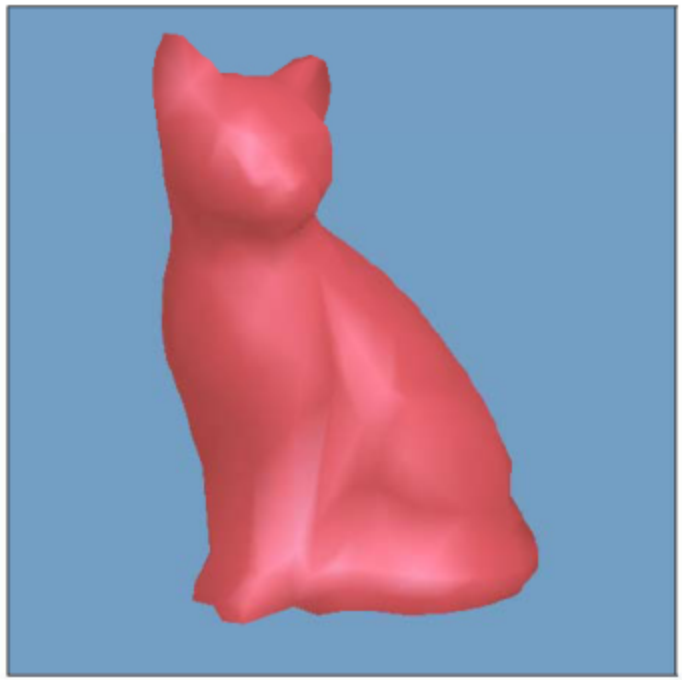


Normal Estimation

- Find normal $\mathbf{n}_i$ for each sample point $\mathbf{p}_i$
  1. Examine local neighborhood for each point
     - Set of k nearest neighbors
  2. Compute best approximating tangent plane
     - Covariance analysis
  3. Determine normal orientation
     - MST propagation



Naïve Orientation Propagation

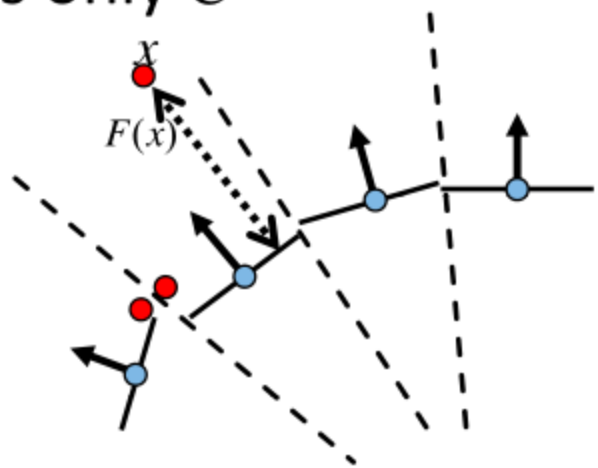


Normal Orientation

- Build graph connecting neighboring points
  - Edge (ij) exists if $\mathbf{p}_i \in \mathrm{kNN}(\mathbf{p}_j)$ or $\mathbf{p}_j \in \mathrm{kNN}(\mathbf{p}_i)$
- Propagate normal orientation through graph
  - For neighbors $\mathbf{p}_i, \mathbf{p}_j$: Flip $\mathbf{n}_j$ if $\mathbf{n}_i^{\mathrm{T}}\mathbf{n}_j < 0$
  - Fails at sharp edges/corners
- Propagate along "safe" paths (parallel normals)
  - Minimum spanning tree with angle-based edge weights $w_{ij} = 1 - |\mathbf{n}_i^{\mathrm{T}}\mathbf{n}_j|$

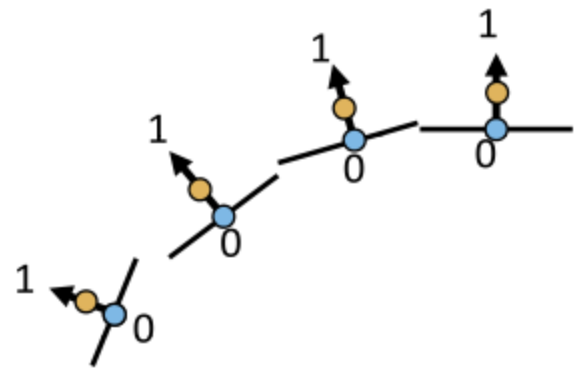
SDF from Point & Normals

- Signed distance from tangent planes
  - Points + normals determine local tangent planes
  - Use distance from closest center's tangent plane
  - Linear approximation in Voronoi cell
  - Simple and efficient, but SDF is only C^{-1}



Smooth SDF Approximation

- Scattered data interpolation problem
  - On-surface constraints $\text{dist}(\mathbf{p}_i) = 0$
  - Avoid trivial solution $\text{dist} \equiv 0$
  - Off-surface constraints $\text{dist}(\mathbf{p}_i + \boldsymbol{\varepsilon}\mathbf{n}_i) = \boldsymbol{\varepsilon}$



Smooth SDF Approximation

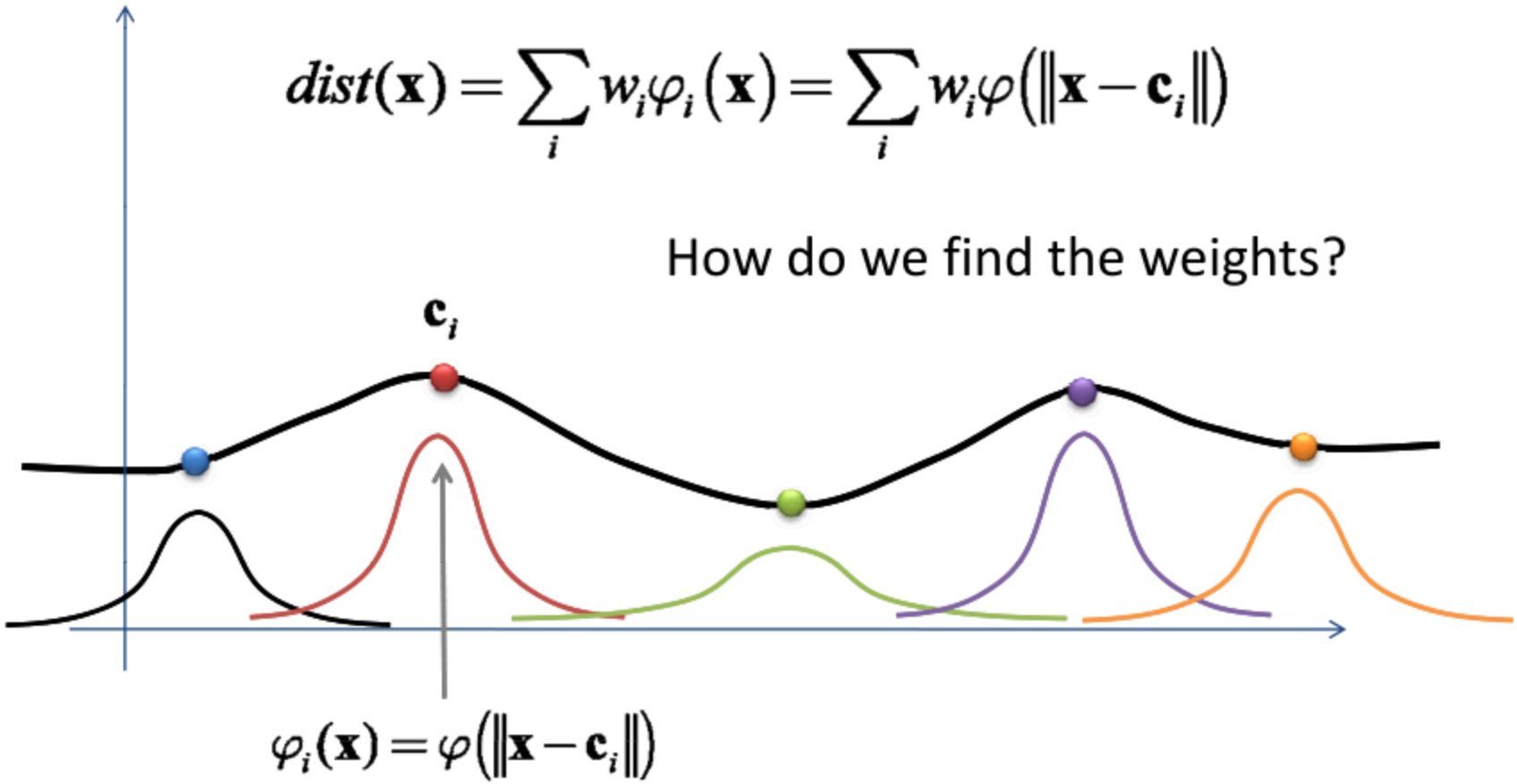
- Scattered data interpolation problem
  - On-surface constraints $\mathrm{dist}(\mathbf{p}_i) = 0$
  - Avoid trivial solution $\mathrm{dist} \equiv 0$
  - Off-surface constraints $\mathrm{dist}(\mathbf{p}_i + \varepsilon \mathbf{n}_i) = \varepsilon$
- Radial basis functions
  - Well suited for smooth interpolation
  - Sum of shifted, weighted *kernel* functions

$$dist(\mathbf{x}) = \sum_i w_i \varphi\left(\left\|\mathbf{x} - \mathbf{c}_i\right\|\right)$$

Radial Basis Functions Interpolation

$$dist(\mathbf{x}) = \sum_i w_i \varphi_i(\mathbf{x}) = \sum_i w_i \varphi(\|\mathbf{x} - \mathbf{c}_i\|)$$

How do we find the weights?



Radial Basis Functions

$$dist(\mathbf{p}_j) = \sum_i w_i \varphi_i(\mathbf{p}_j) = \sum_i w_i \varphi(\|\mathbf{p}_j - \mathbf{c}_i\|) = 0 \qquad 2n \text{ equations}$$

$$dist(\mathbf{p}_j + \varepsilon \mathbf{n}_j) = \sum_i w_i \varphi(\|(\mathbf{p}_j + \varepsilon \mathbf{n}_j) - \mathbf{c}_i\|) = \varepsilon \qquad 2n \text{ variables}$$

The on- and off-surface points are the centers $\mathbf{c}_i$

Radial Basis Functions

$$dist(\mathbf{p}_j) = \sum_i w_i \varphi_i(\mathbf{p}_j) = \sum_i w_i \varphi(\|\mathbf{p}_j - \mathbf{c}_i\|) = 0$$

$2n$ equations

$$dist(\mathbf{p}_j + \varepsilon \mathbf{n}_j) = \sum_i w_i \varphi(\|(\mathbf{p}_j + \varepsilon \mathbf{n}_j) - \mathbf{c}_i\|) = \varepsilon$$

$2n$ variables

The on- and off-surface points are the centers $\mathbf{c}_i$

$$\begin{bmatrix} \phi(||\mathbf{x}_1 - \mathbf{x}_1||) & \cdots & \phi(||\mathbf{x}_1 - (\mathbf{x}_n + \varepsilon \mathbf{n}_n)||) \\ \vdots & & \vdots \\ \phi(||(\mathbf{x}_n + \varepsilon \mathbf{n}_n) - \mathbf{x}_1||) & \cdots & \phi(||(\mathbf{x}_n + \varepsilon \mathbf{n}_n) - (\mathbf{x}_n + \varepsilon \mathbf{n}_n)||) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_n \\ \vdots \\ w_{2n} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \varepsilon \\ \vdots \\ \varepsilon \end{bmatrix}$$

$\mathbf{K}$ $\mathbf{w}$ $\mathbf{d}$

Radial Basis Functions

- Solve the system $\mathbf{Kw} = \mathbf{d}$ for $\mathbf{w}$
- Implicit function is defined as:

$$dist(\mathbf{x}) = \sum_{i=1}^{n} w_i \varphi\left(\left\|\mathbf{x} - \mathbf{p}_j\right\|\right) + \sum_{i=n+1}^{2n} w_i \varphi\left(\left\|\mathbf{x} - \left(\mathbf{p}_j + \varepsilon \mathbf{n}_j\right)\right\|\right)$$

RBF Basis Functions

- Tri-harmonic basis functions

$$\phi(r) = r^3$$

  - Globally supported
  - Leads to dense symmetric linear system
  - C^2 smoothness
  - Provably smooth
  - Works well for highly irregular sampling

Other Radial Basis Functions

- Polyharmonic spline

$$\phi(r) = r^k \log(r) \quad , k = 2,4,6,\ldots$$

$$\phi(r) = r^k \quad , k = 1,3,5,\ldots$$

- Multiquadratic

$$\phi(r) = \sqrt{r^2 + \beta^2}$$

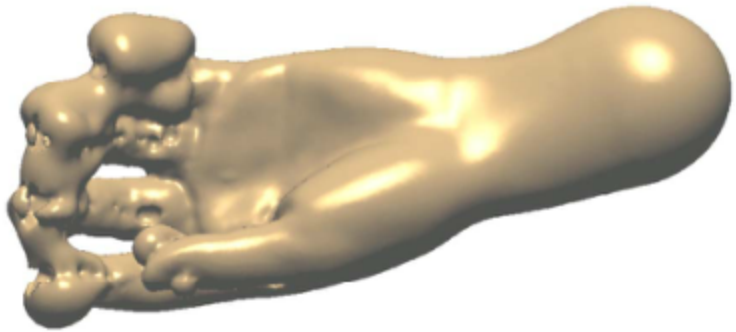
- Gaussian

$$\phi(r) = e^{-\beta r^2}$$

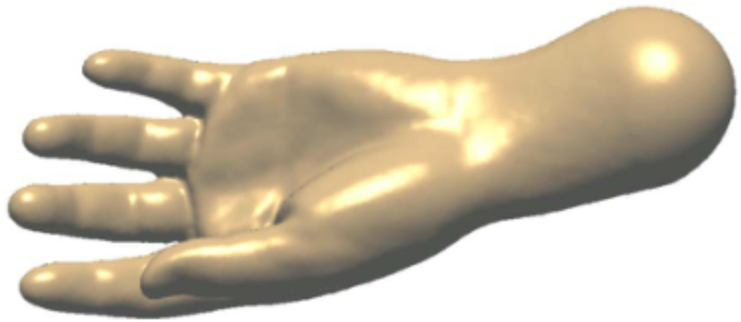
- B-Spline (compact support)

$$\phi(r) = \text{piecewise-poly}(r)$$

How Big is ε?

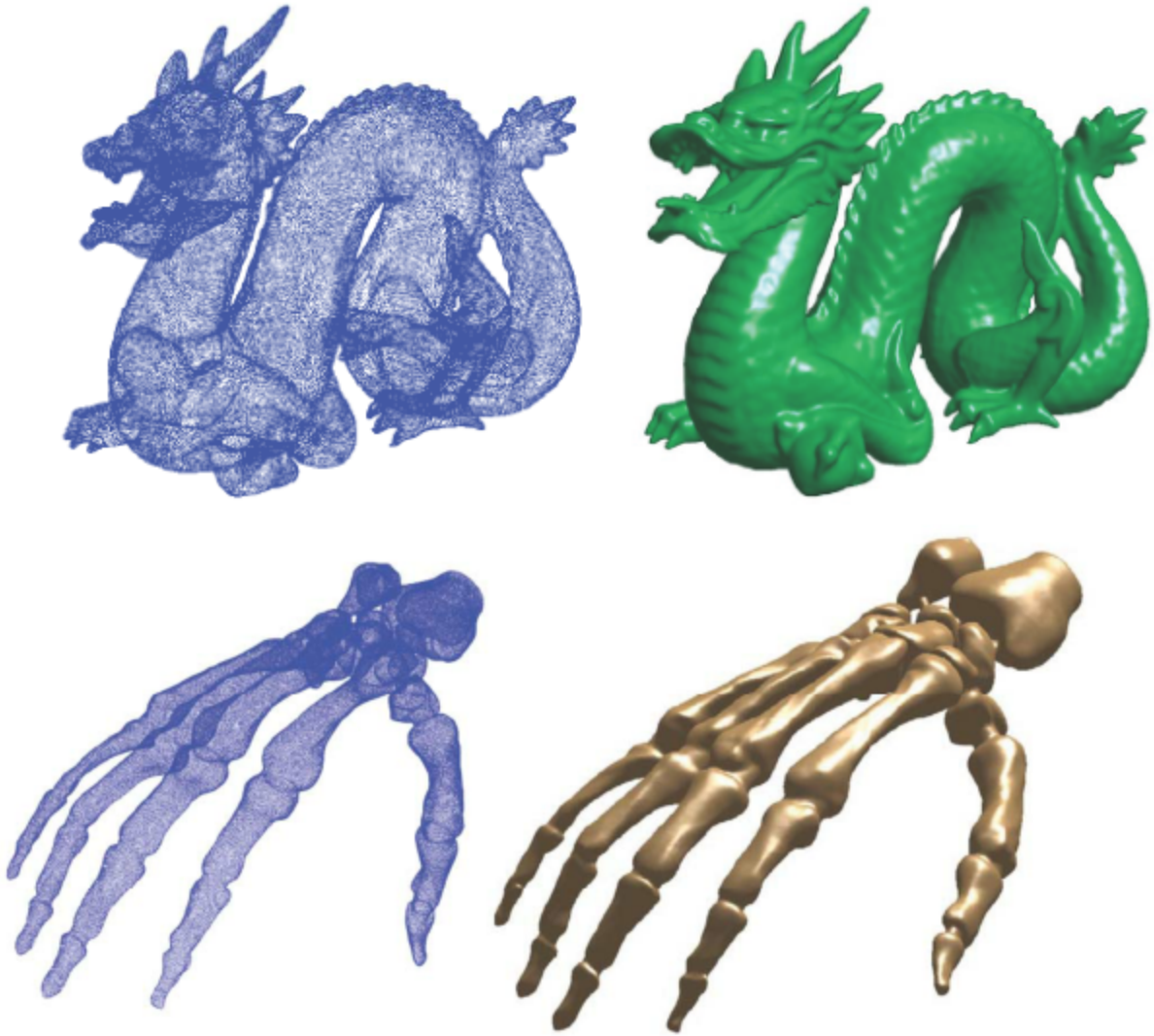


Without normal length validation



With normal length validation

Examples



Comparison

Distance
to plane

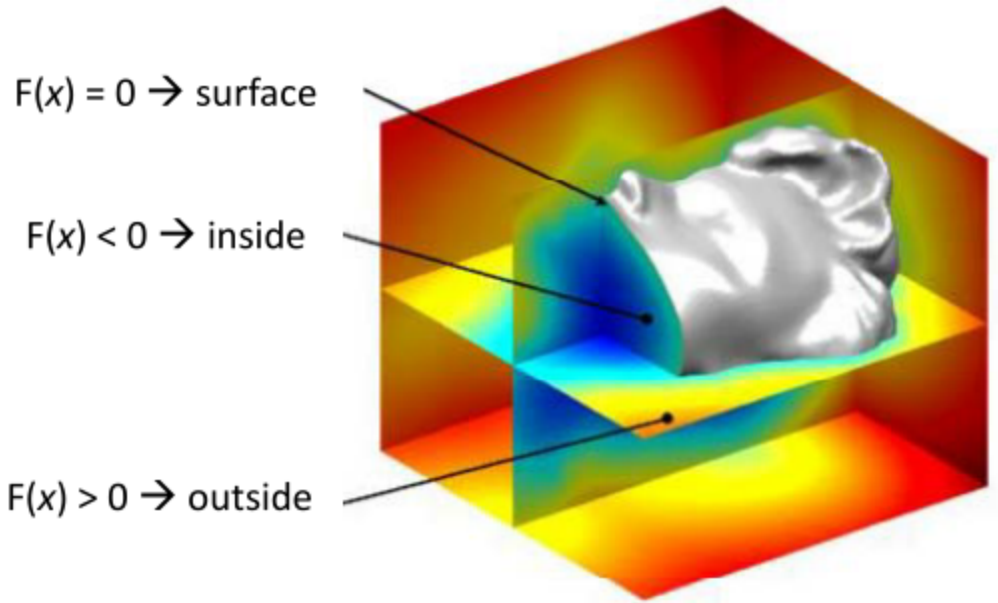
Compact RBF

Global RBF
Triharmonic

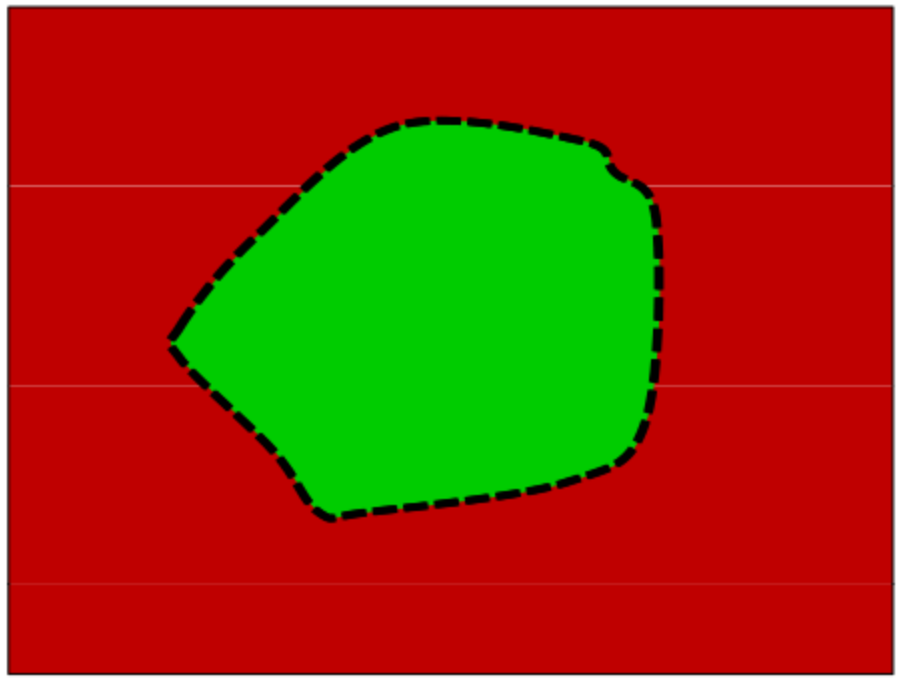
Complexity Issues

- Solve the linear system for RBF weights
  - Hard to solve for large number of samples
- Compactly supported RBFs
  - Sparse linear system
  - Efficient solvers
- Greedy RBF fitting
  - Start with a few RBFs only
  - Add more RBFs in region of large error

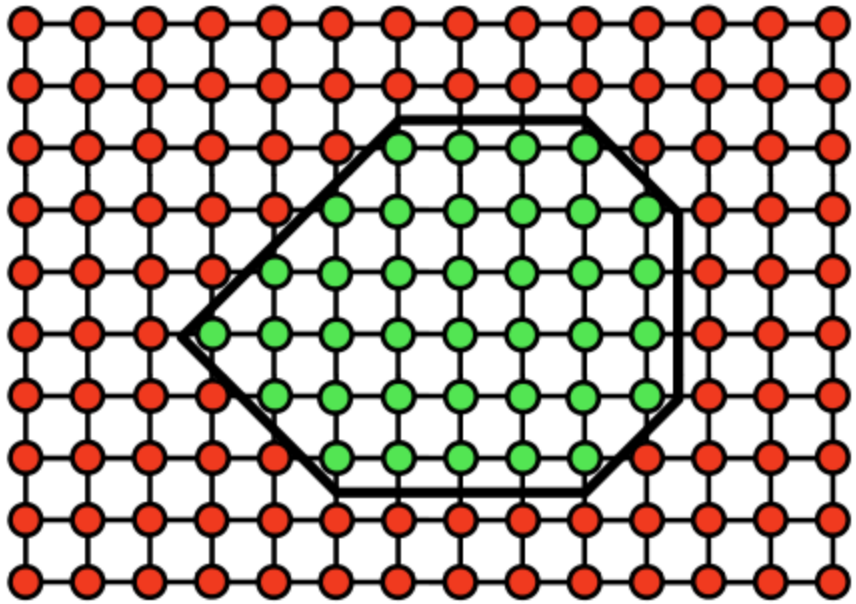
Extracting the Surface



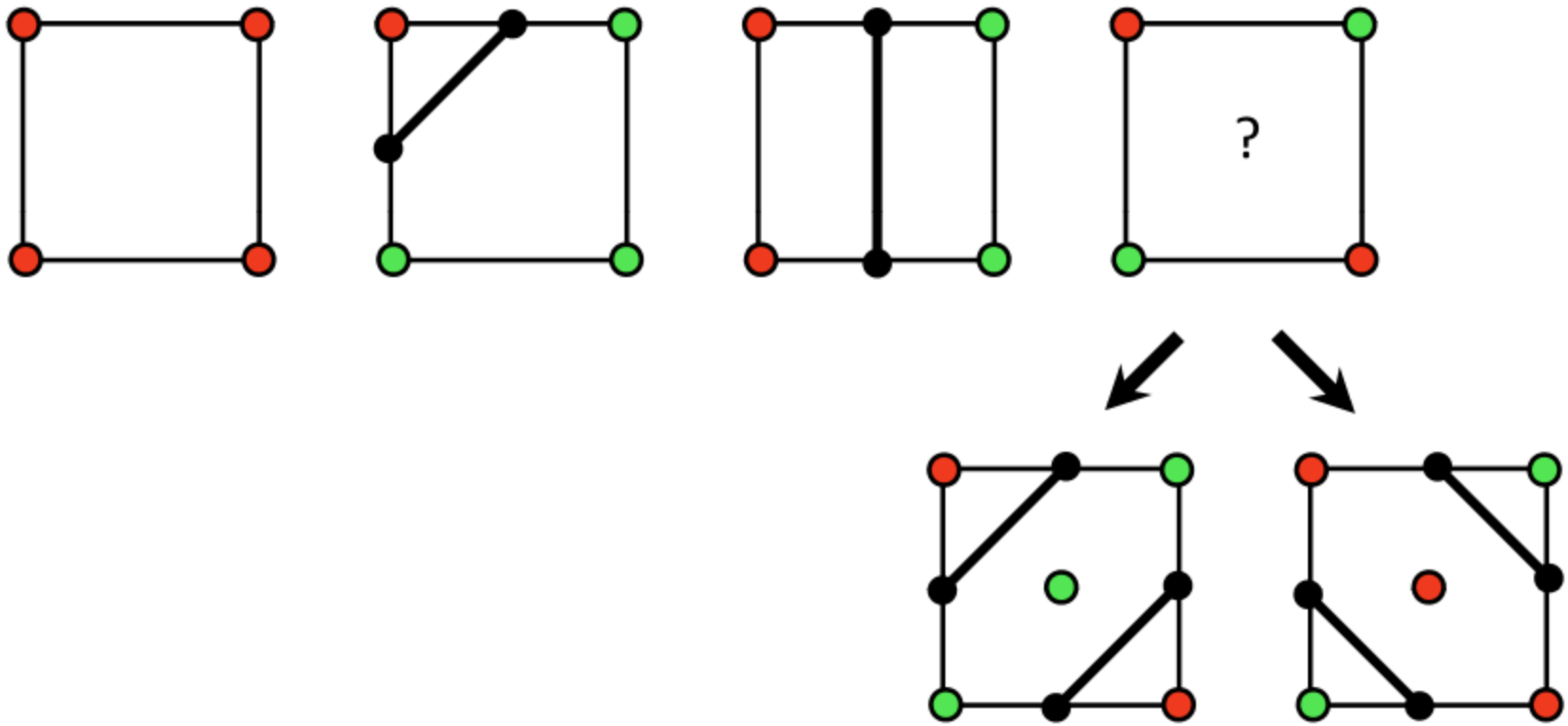
Sample the SDF



Sample the SDF



2D: Marching Squares

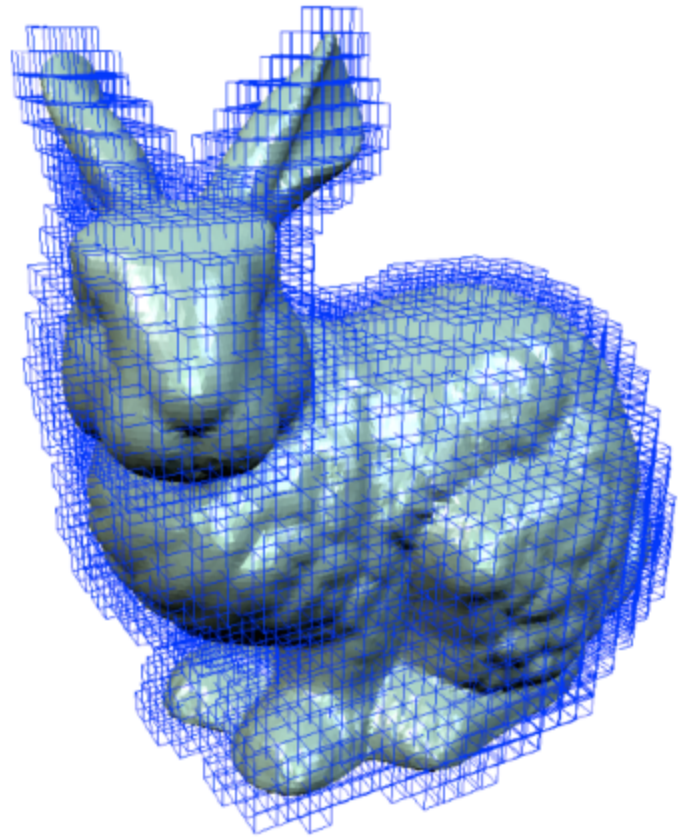


3D: Marching Cubes

- Classify grid nodes as inside/outside
- Classify cell: 2^8 configurations
- Linear interpolation along edges
- Look-up table for patch configuration
  - Disambiguation more complicated

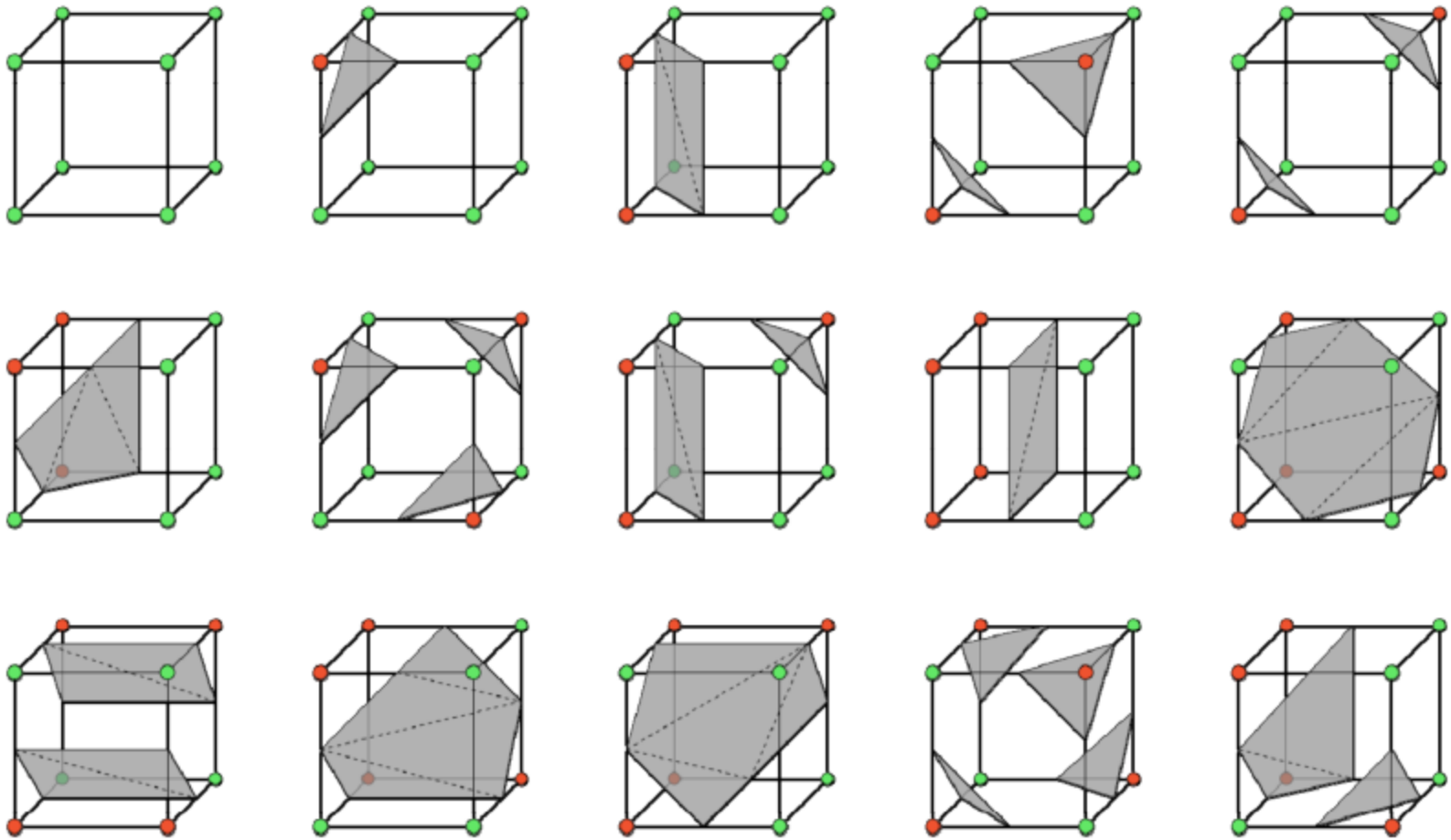
Marching Cubes

- Cell classification:
  - Inside
  - Outside
  - Intersecting

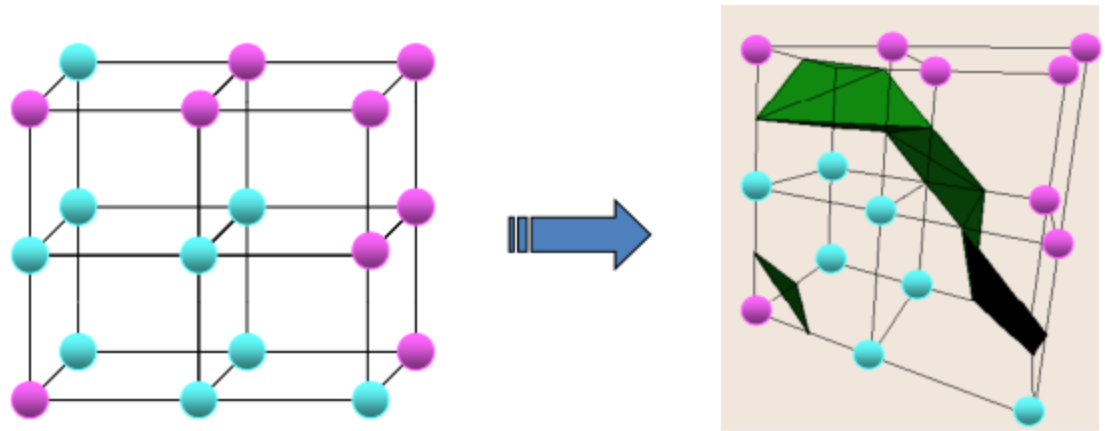


Marching Cubes

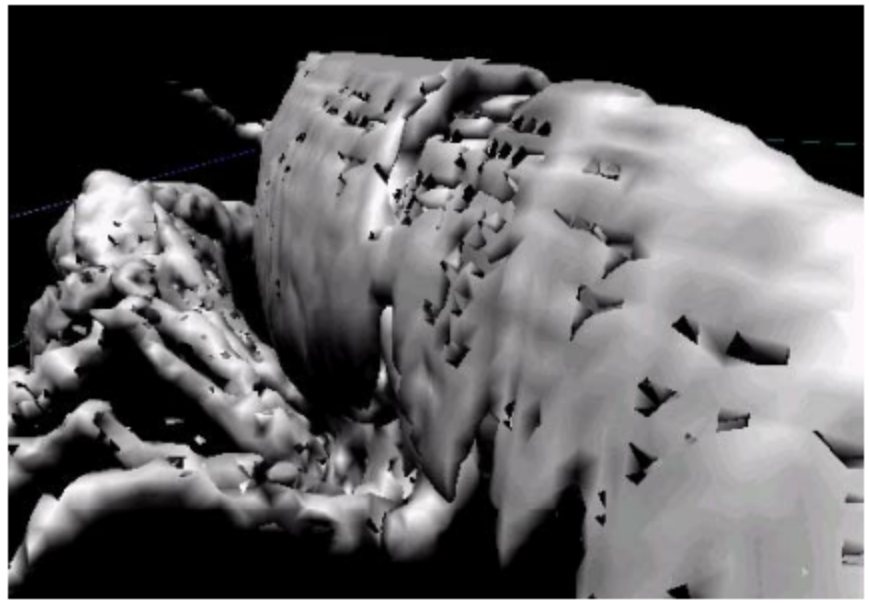
256 cases → 15 cases
Inversion
Rotation



Marching Cubes

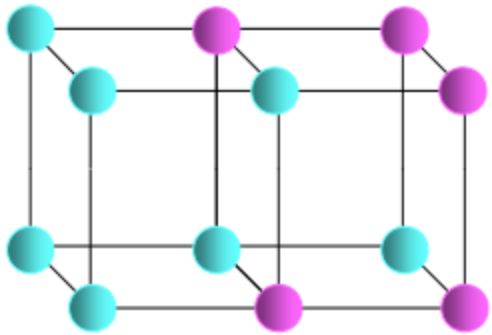


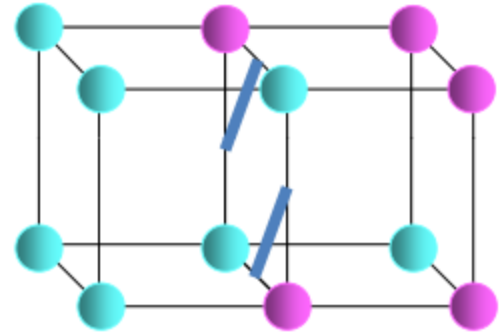
Marching Cubes Problems



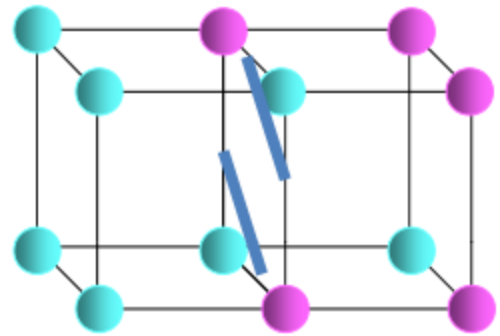
- Ambiguity
  - Holes
- Generates HUGE meshes
  - Millions of polygons

Ambiguity

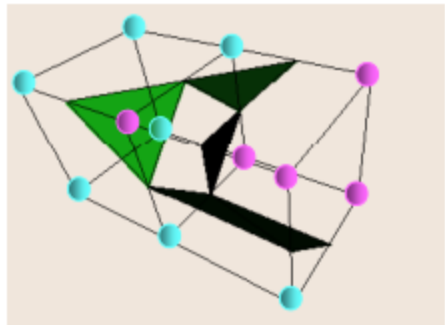




Separate pink



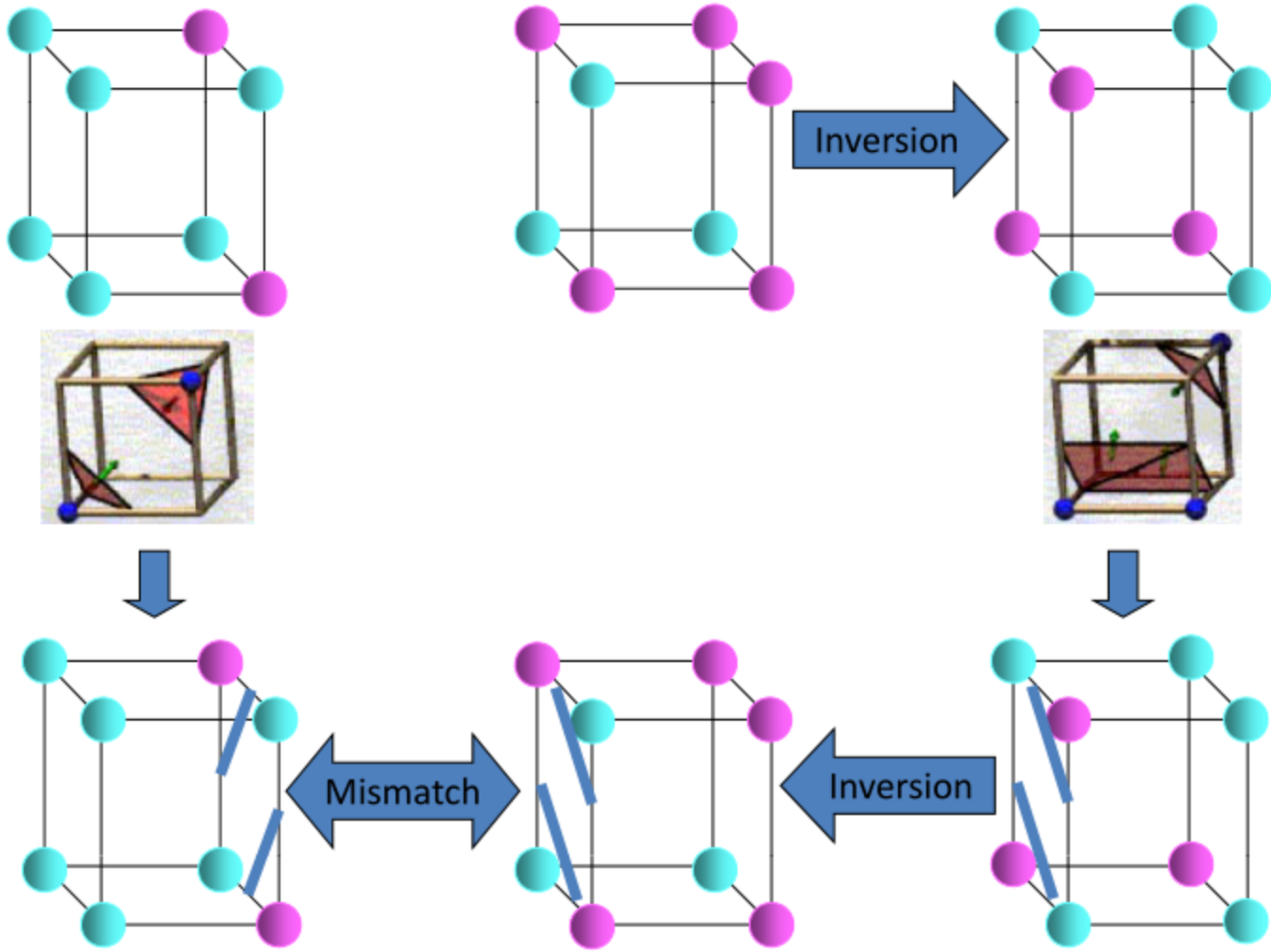
Separate blue



The Inversion Problem

- Reduction from 256 to 15 cases includes inversion

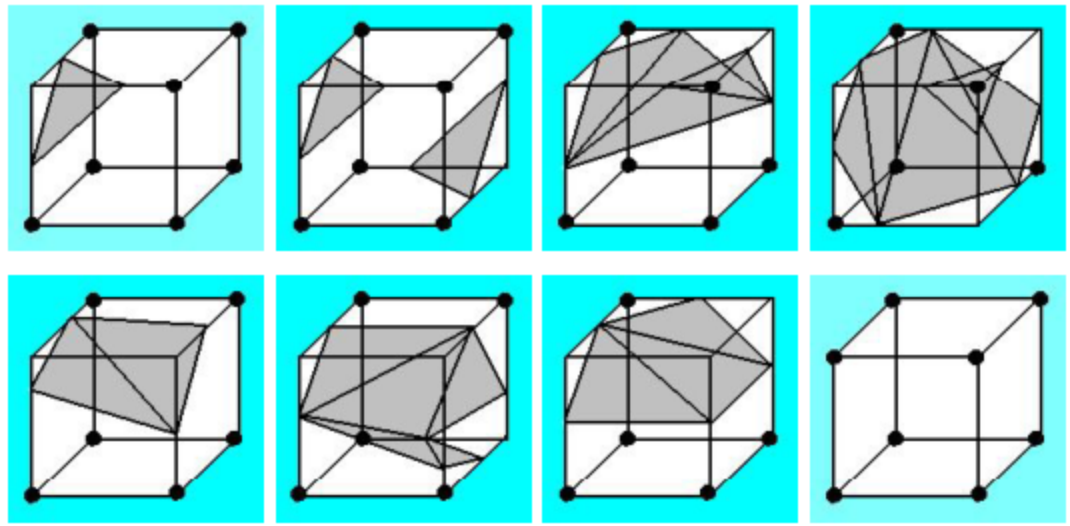
The Inversion Problem



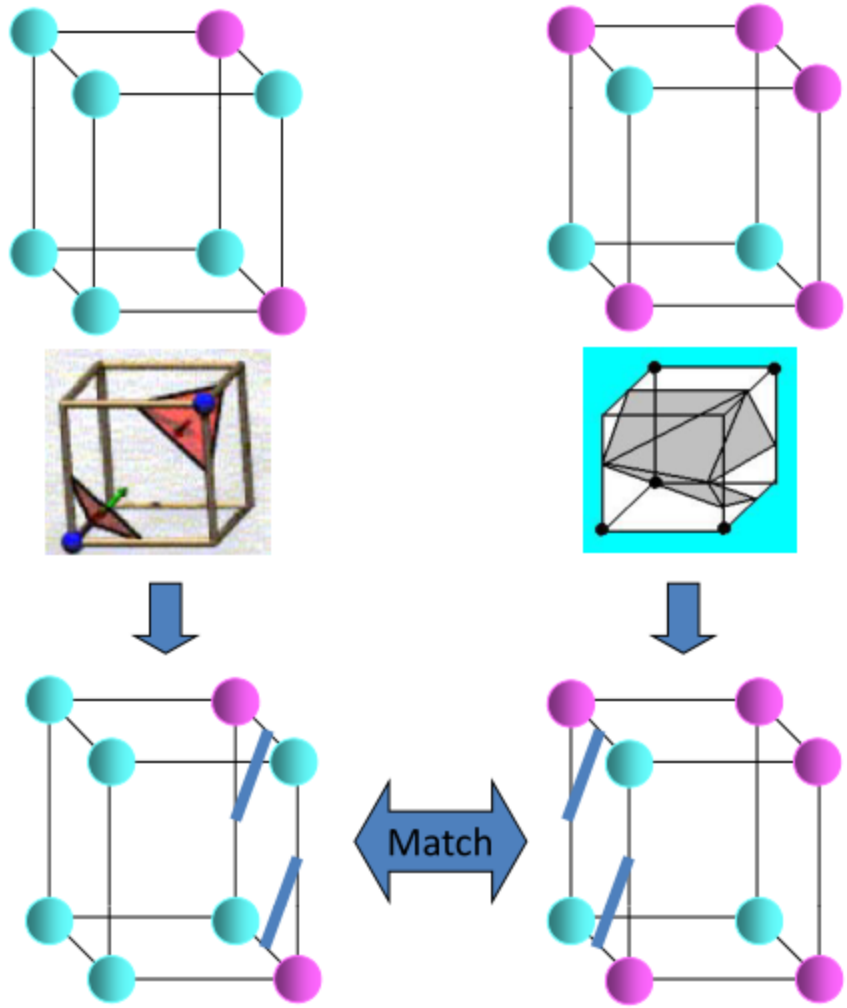
Ambiguity Solution

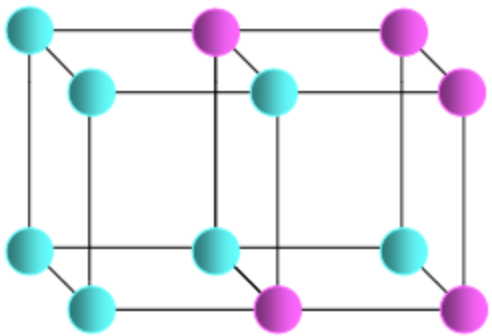
- 256 cases → 23 cases
  - Rotation only
  - Always separate same “color”
  - → Ambiguous faces triangulated consistently

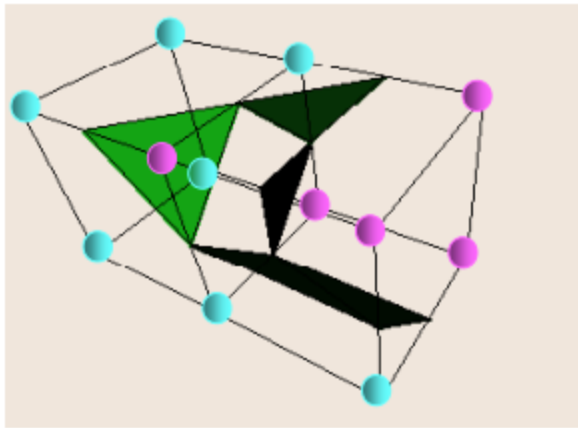
- 8 new cases



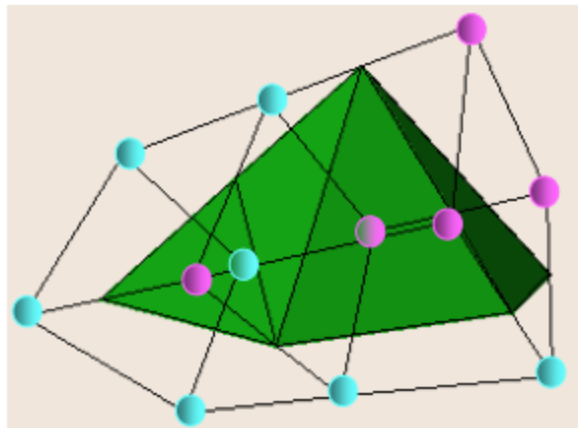
Without Inversion



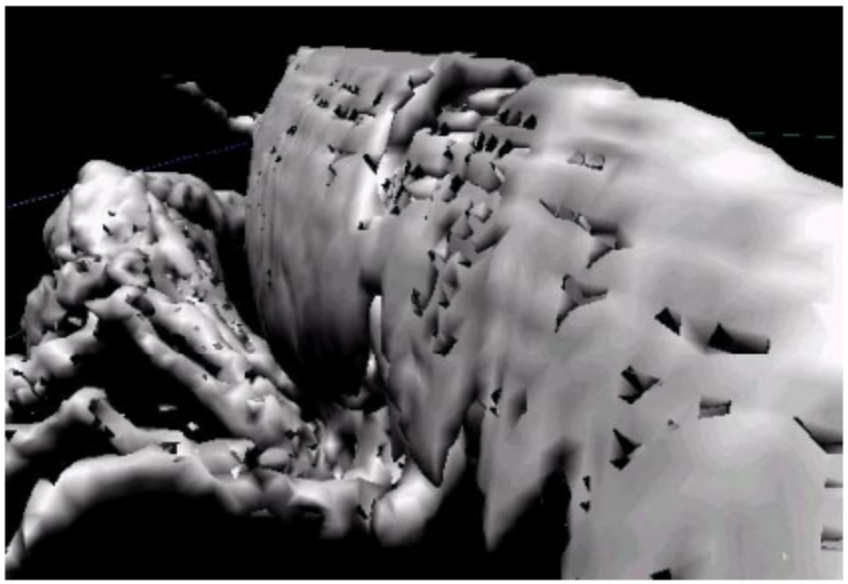




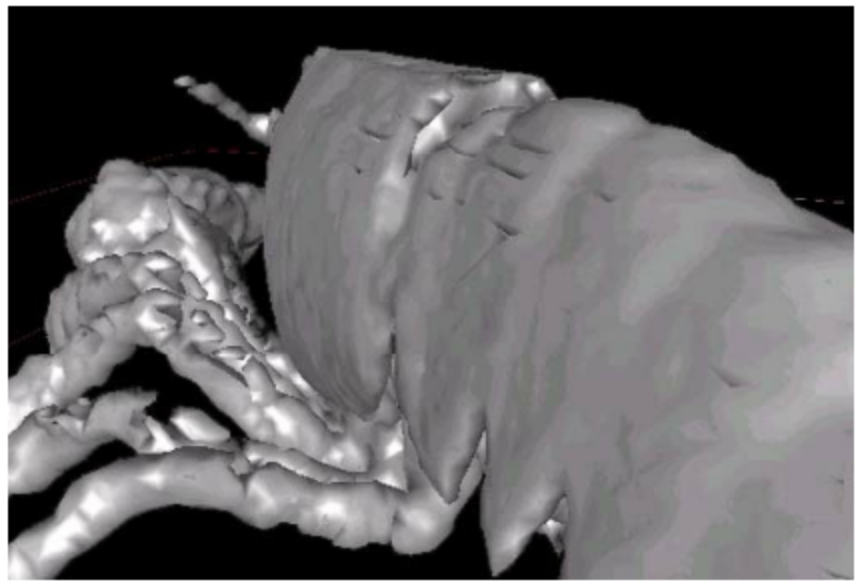
Ambiguity



No Ambiguity



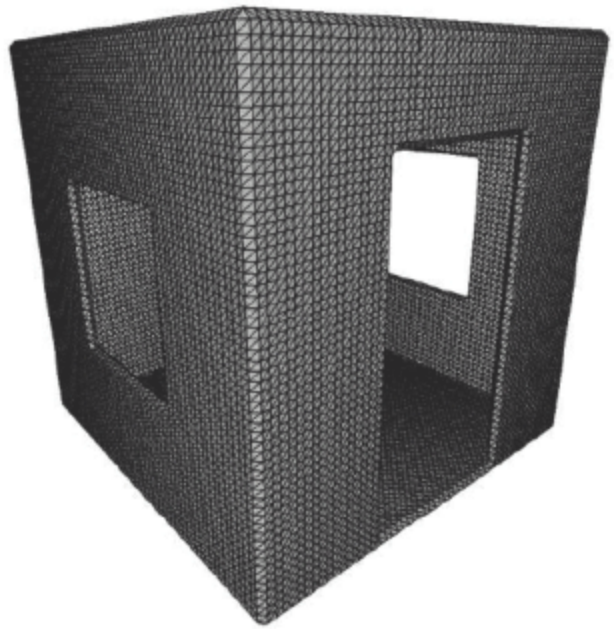
Ambiguity



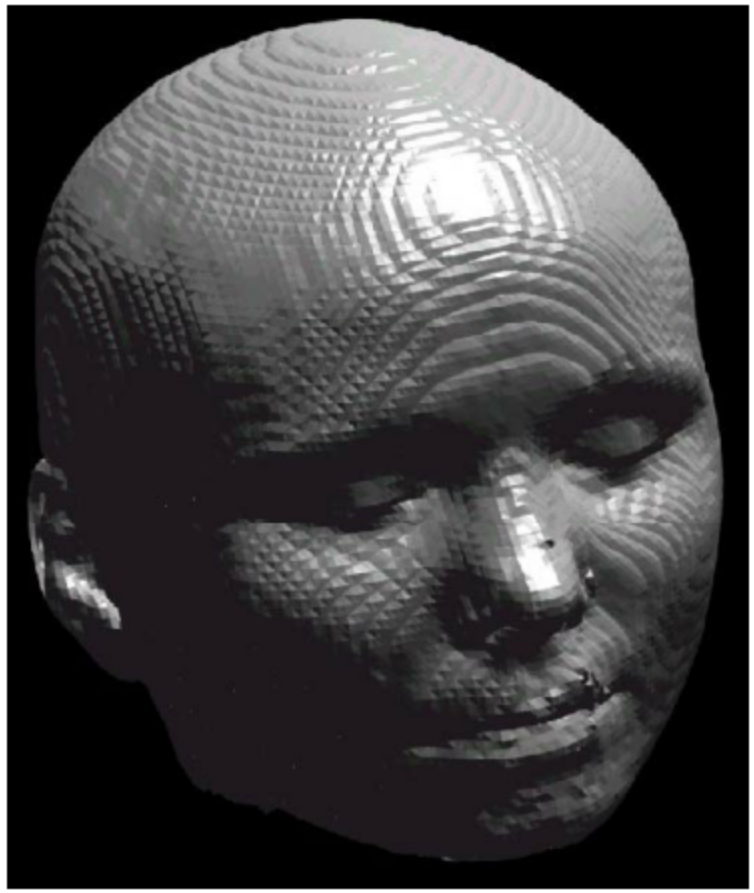
No Ambiguity

Marching Cubes Issues

- Grid not adaptive
- Many polygons required to represent small features



Images from: "Dual Marching Cubes: Primal Contouring of Dual Grids" by Schaeffer et al.



Beyond RBF

- Poisson reconstruction
  [Kazhdan 06]