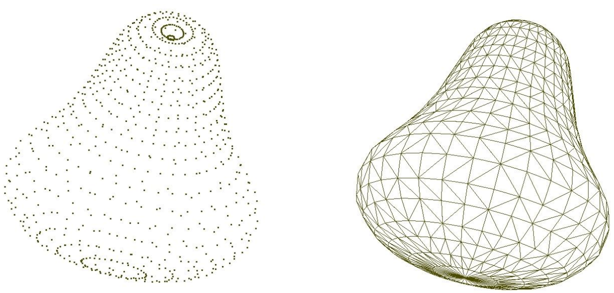
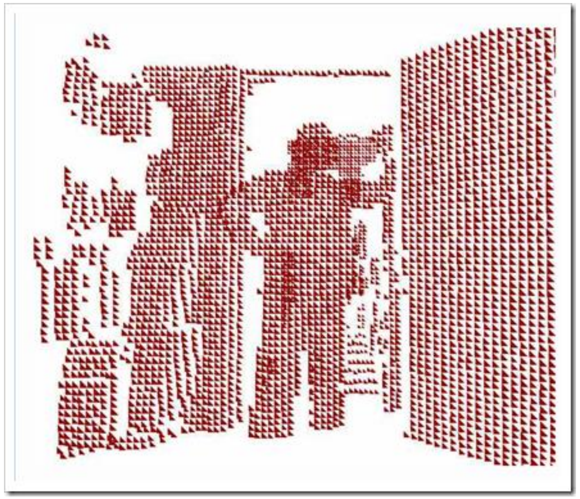
Surface Reconstruction From Point Sets

COS 526, Fall 2016

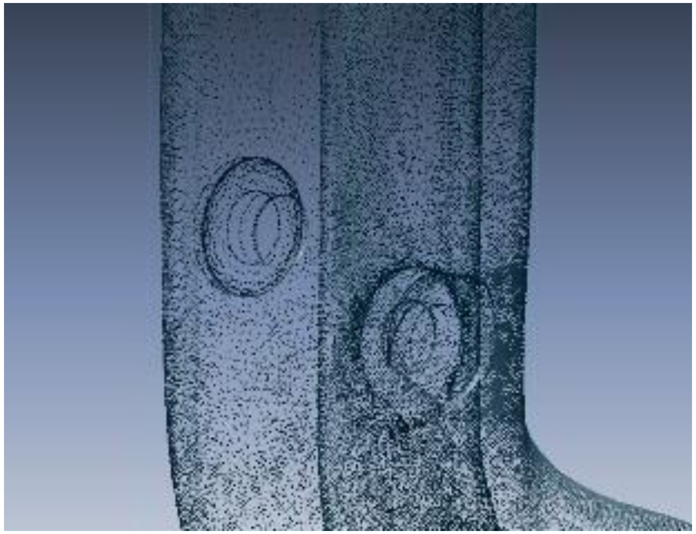


Slides from Misha Kazhdan, Fisher Yu, Szymon Rusinkiewicz, Ioannis Stamos, Hugues Hoppe, and Piyush Rai

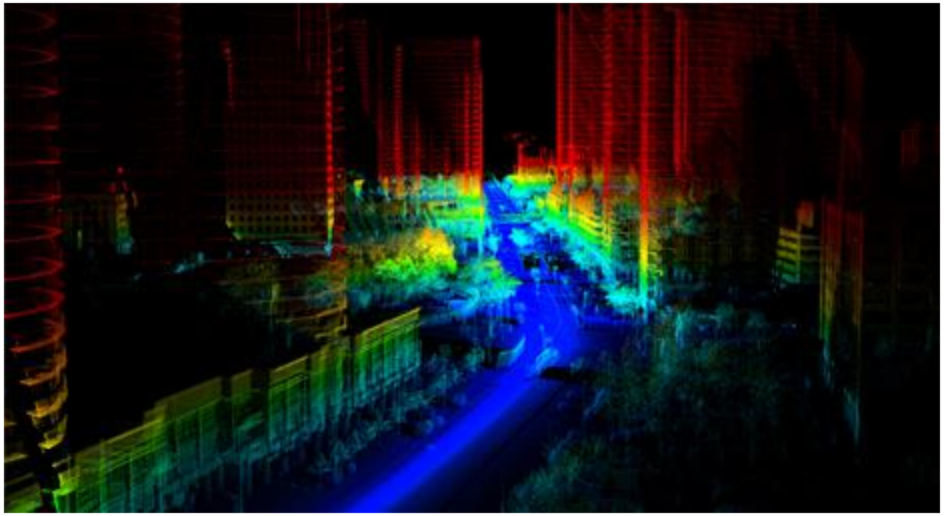
Point Sets



Greg Duncan



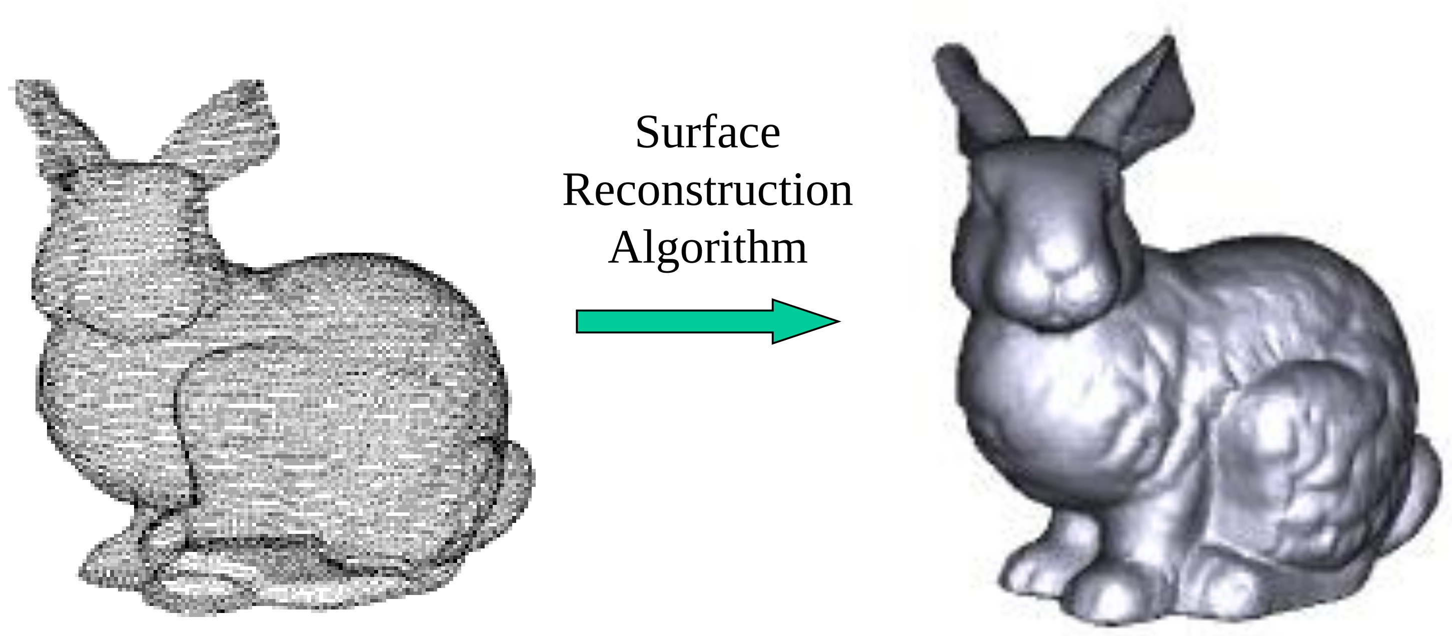
Absolute Geometries



OpTech

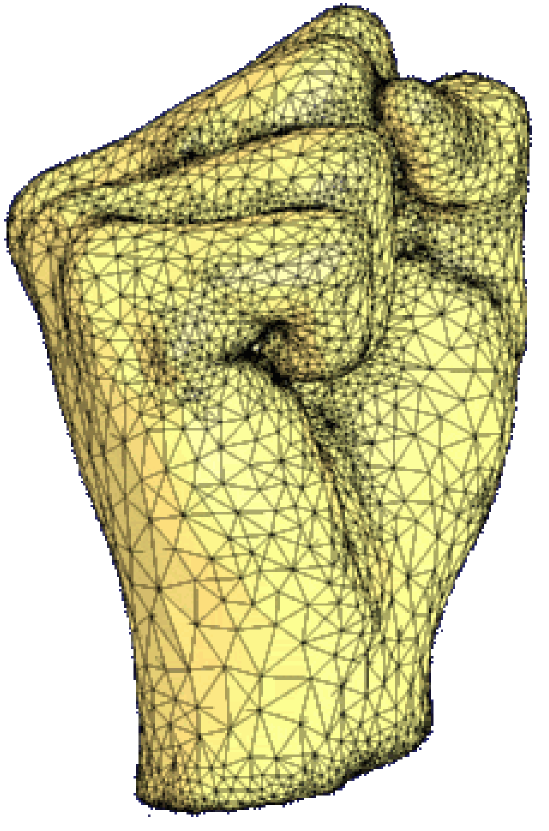
Problem Statement

- Given a set of sample points in three dimensions produce a surface that captures the “most reasonable shape” the points were sampled from.



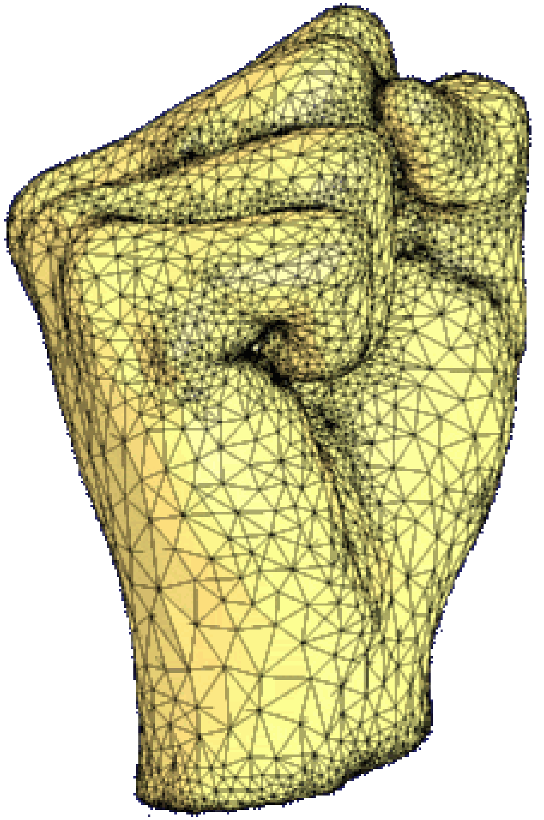
Applications

- Computer graphics,
- Medical imaging
- Cartography
- Compression
- Reverse engineering
- Urban modeling
- etc.

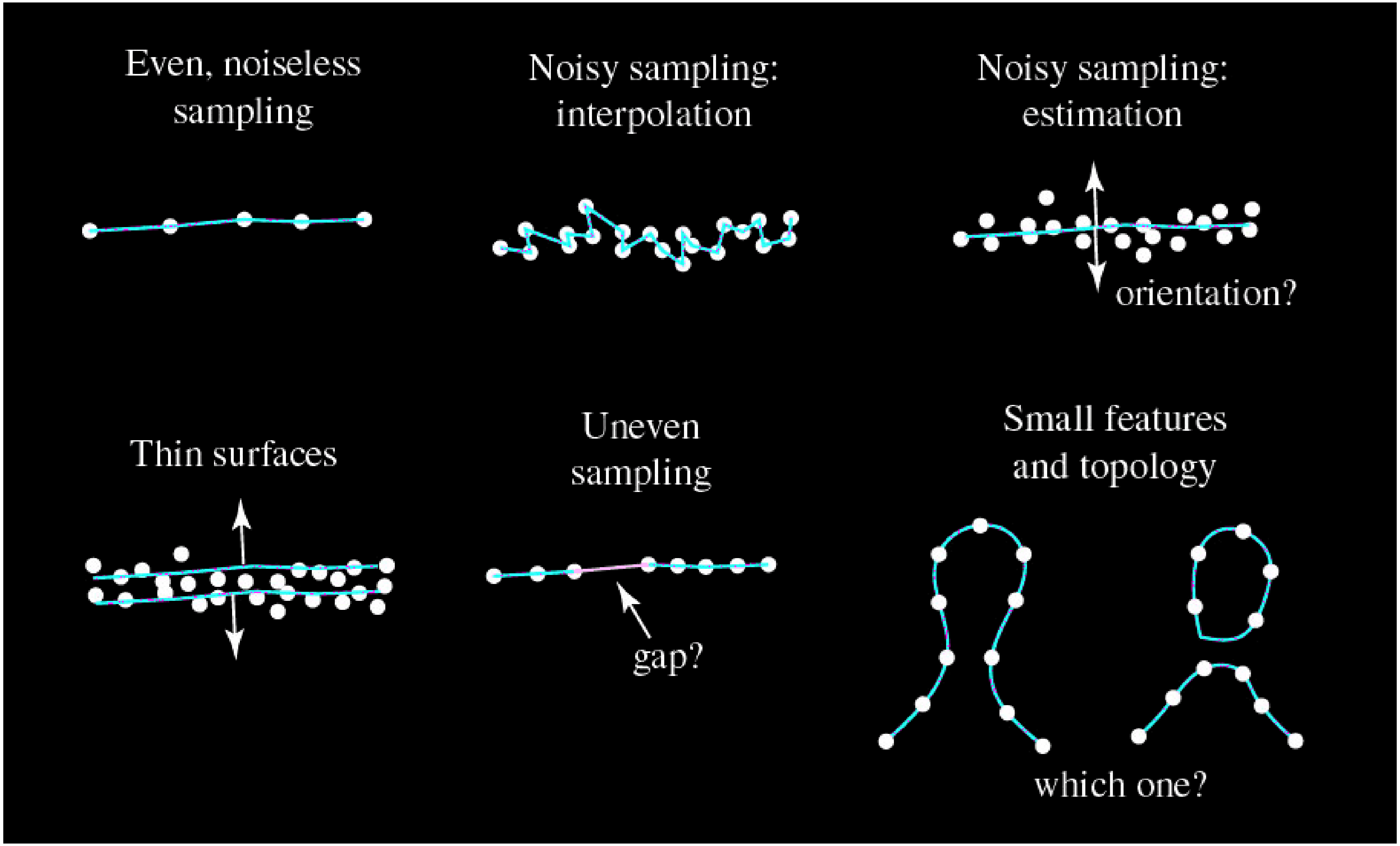


Desirable Properties

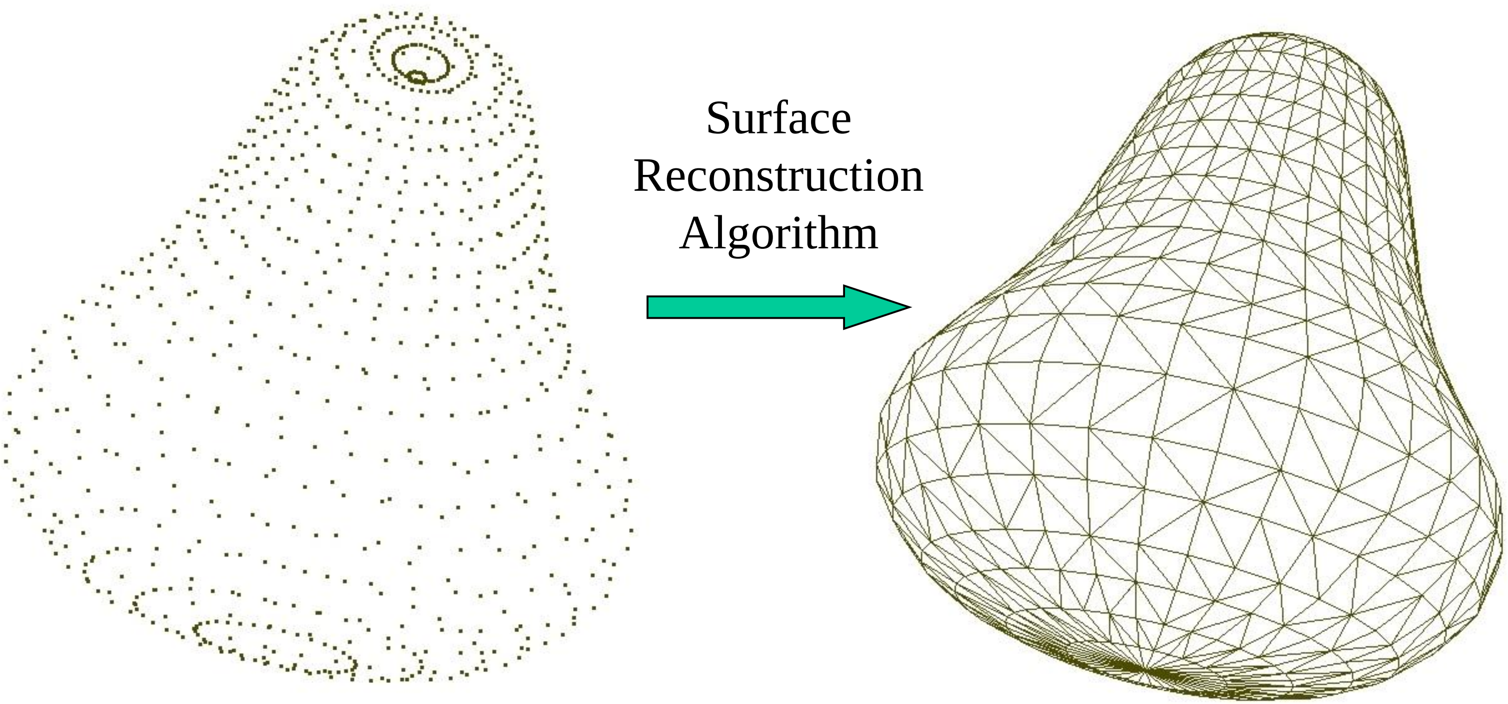
- Interpolate input points
- Handle arbitrary genus
- Generally smooth
- Retain sharp features
- Watertight surface



Challenges


Even, noiseless sampling
Noisy sampling: interpolation
Noisy sampling: estimation
orientation?
Thin surfaces
Uneven sampling
gap?
Small features and topology
which one?

Possible Approaches?



Possible Approaches

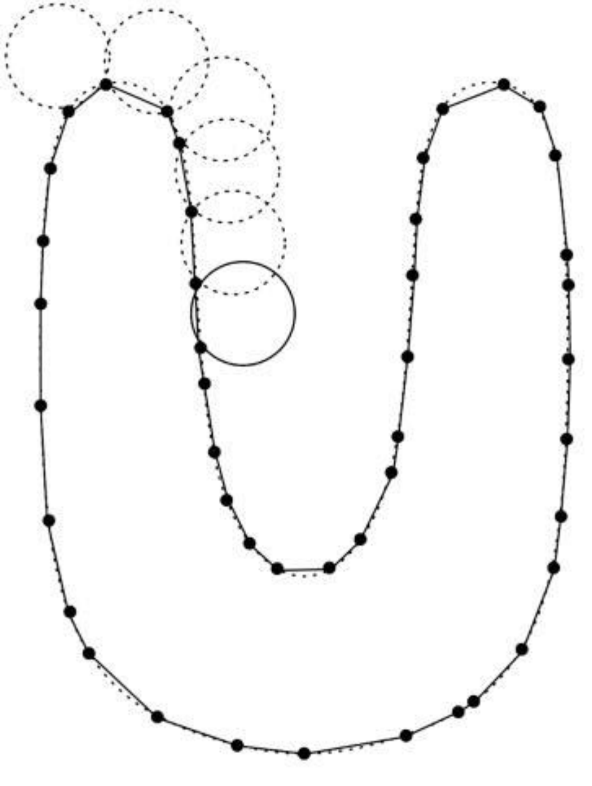
- Explicit Meshing
  - Ball pivoting algorithm
  - Crust
  - etc.
- Implicit Reconstruction
  - Hoppe's algorithm
  - Moving Least Squares (MLS)
  - Poisson surface reconstruction
  - etc.
- Surface fitting
  - Deformable templates
  - etc.

Possible Approaches

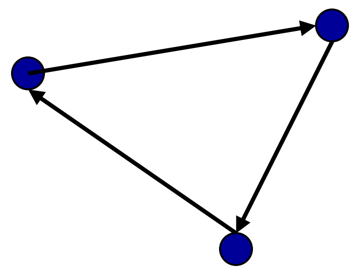
- Explicit Meshing
  - Ball pivoting algorithm
  - Crust
  - etc.
- Implicit Reconstruction
  - Hoppe's algorithm
  - Moving Least Squares (MLS)
  - Poisson surface reconstruction
  - etc.
- Surface fitting
  - Deformable templates
  - etc.

The Ball Pivoting Algorithm

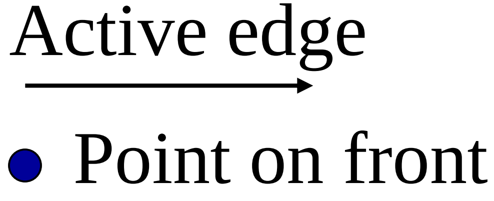
- Pick a ball radius, roll ball around surface, connect what it hits



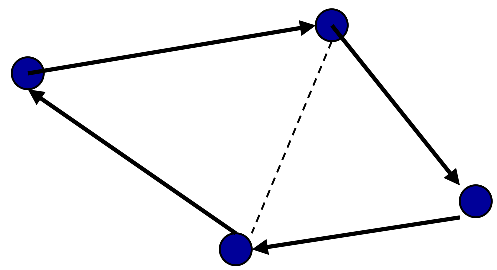
The Ball Pivoting Algorithm



Initial seed triangle



The Ball Pivoting Algorithm

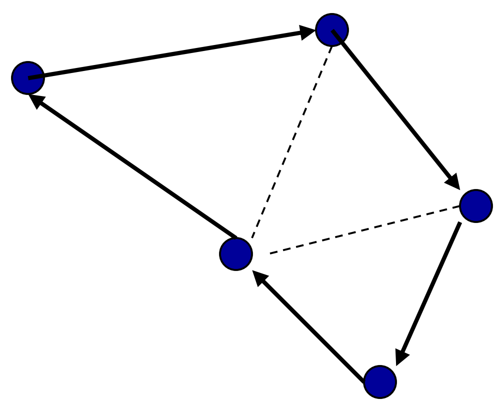


Ball pivoting around active edge

Active edge

Point on front

The Ball Pivoting Algorithm

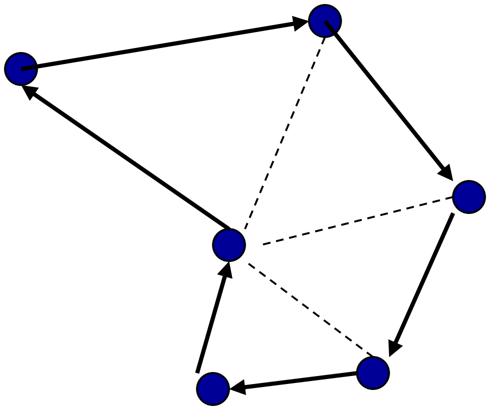


Ball pivoting around active edge

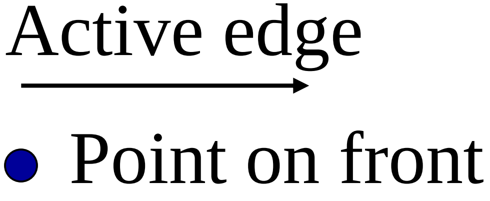
Active edge

Point on front

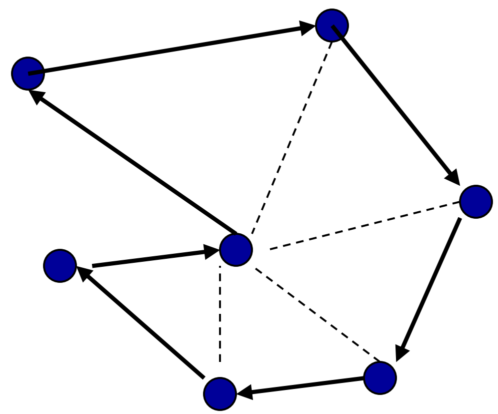
The Ball Pivoting Algorithm



Ball pivoting around active edge



The Ball Pivoting Algorithm

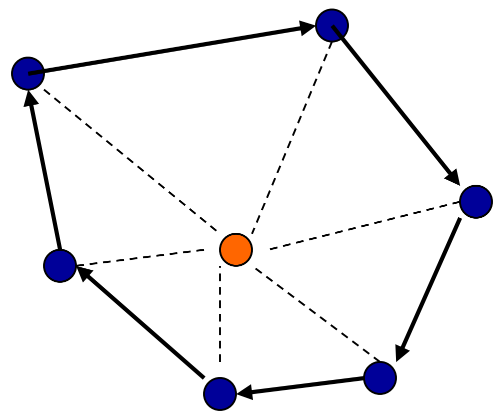


Ball pivoting around active edge

Active edge

Point on front

The Ball Pivoting Algorithm



Ball pivoting around active edge

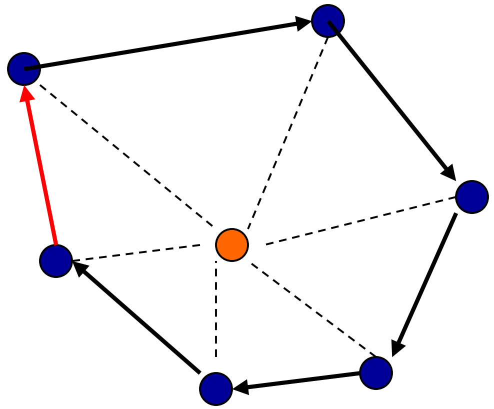
Active edge

Point on front

Internal point

The Ball Pivoting Algorithm

Boundary edge



Ball pivoting around active edge

No pivot found

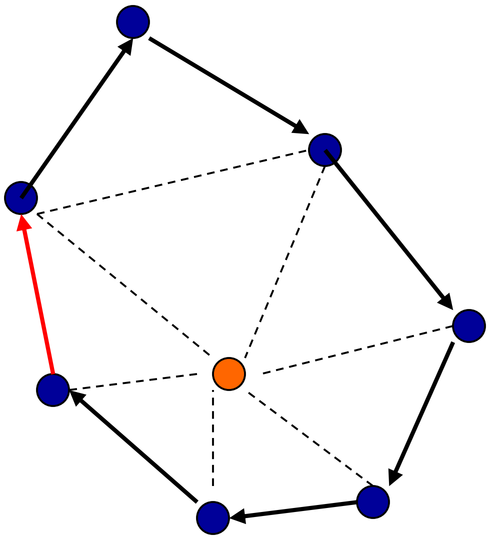
Active edge

Point on front

Internal point

The Ball Pivoting Algorithm

Boundary edge



Ball pivoting around active edge

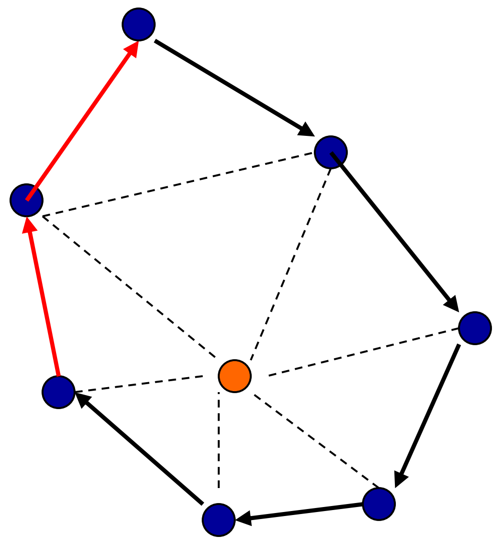
Active edge

Point on front

Internal point

The Ball Pivoting Algorithm

Boundary edge



Ball pivoting around active edge

Active edge

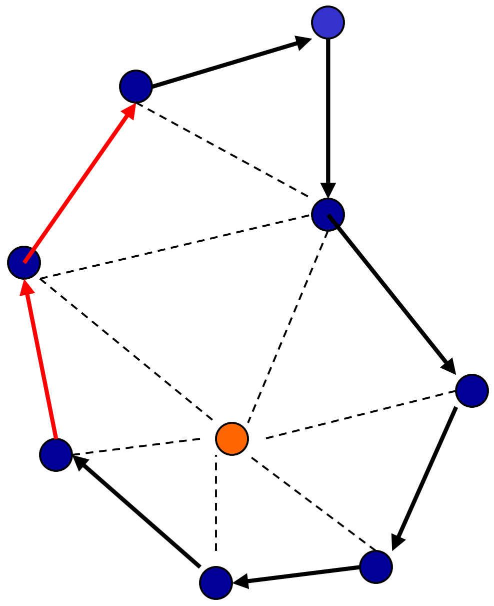
No pivot found

Point on front

Internal point

The Ball Pivoting Algorithm

Boundary edge



Ball pivoting around active edge

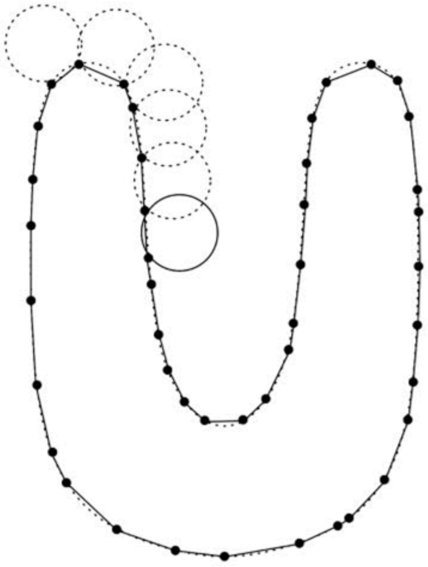
Active edge

Point on front

Internal point

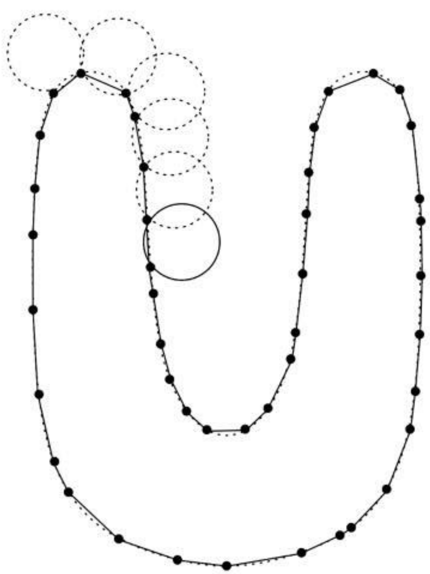
The Ball Pivoting Algorithm

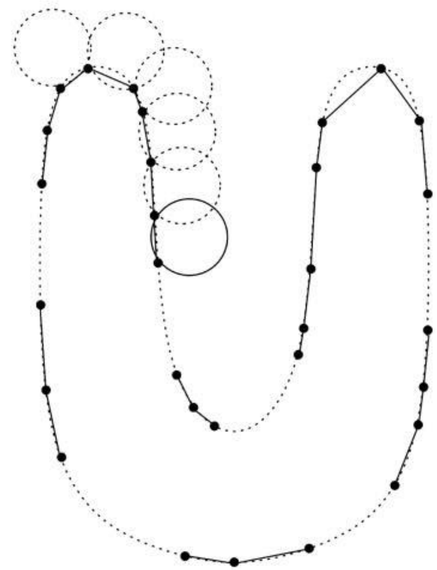
Possible problems?



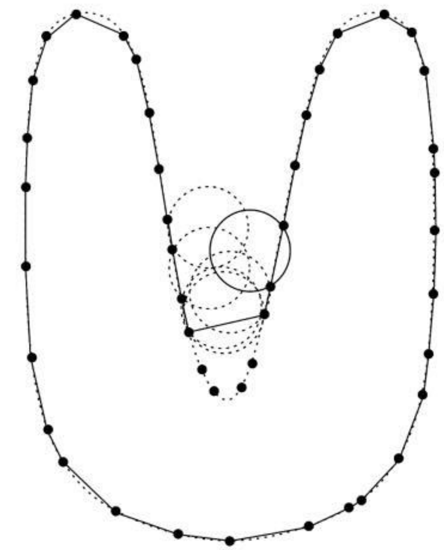
The Ball Pivoting Algorithm

Possible problems?





Undersampling

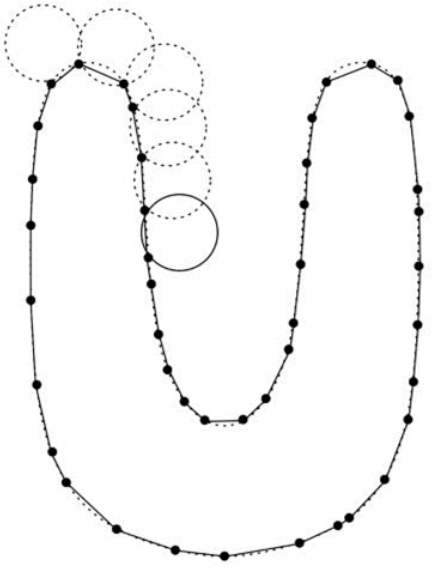


Small Concavities

The Ball Pivoting Algorithm

Possible problems?

Self-intersection? Watertight?



Possible Approaches

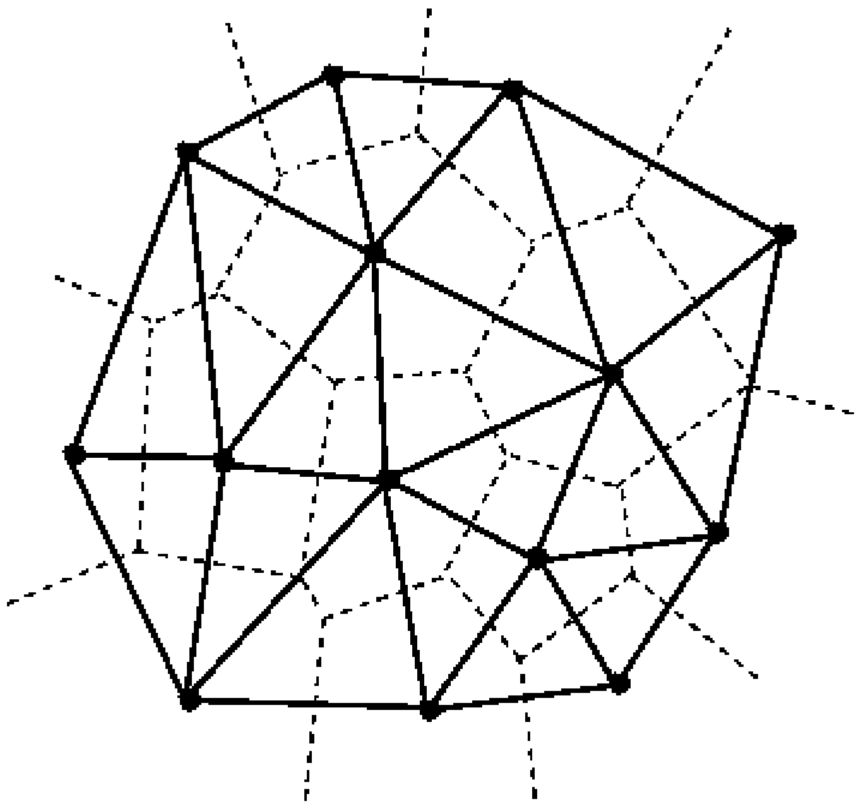
- Explicit Meshing
  - Ball pivoting algorithm
  - Crust
  - etc.
- Implicit Reconstruction
  - Hoppe's algorithm
  - Moving Least Squares (MLS)
  - Poisson surface reconstruction
  - etc.
- Surface fitting
  - Deformable templates
  - etc.

Crust

Aims to find adjacent surface without a parameter specifying feature sizes

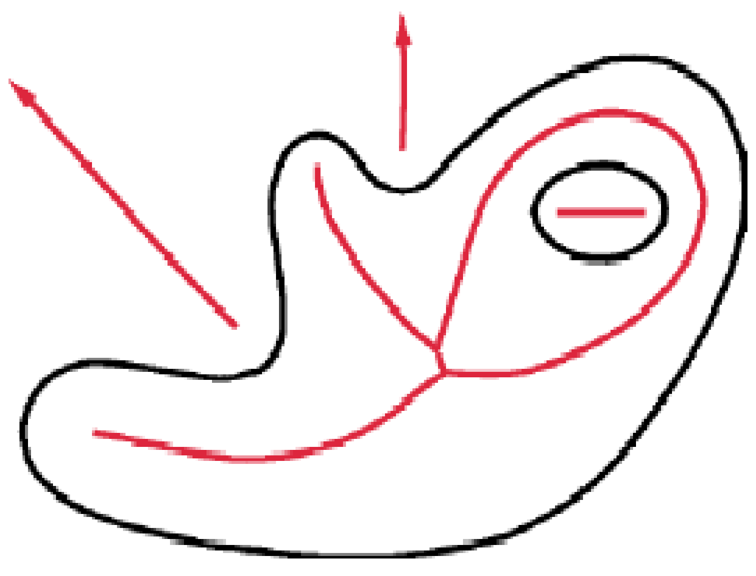
Definitions

Delaunay Triangulation, Voronoi Diagram



Definitions

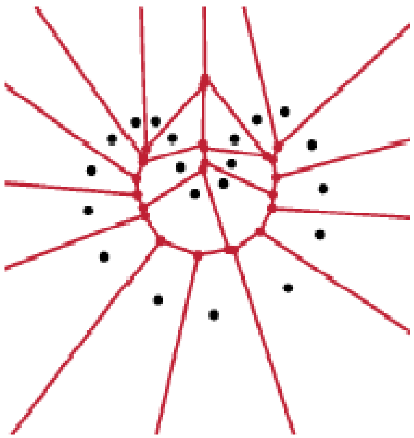
Medial Axis: of surface F is the closure of points that have more than one closest point in F.

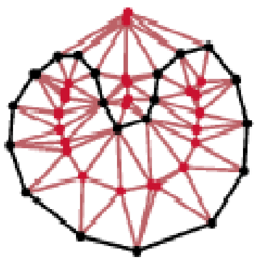


The Intuition behind Crust

The Voronoi Cells of a dense sampling are thin and long.

The Medial Axis is the extension of Voronoi Diagram for continuous surfaces in the sense that the Voronoi Diagram of S Can be defined as the set of points with more than one closest point in S. (S = Sample Point Set)





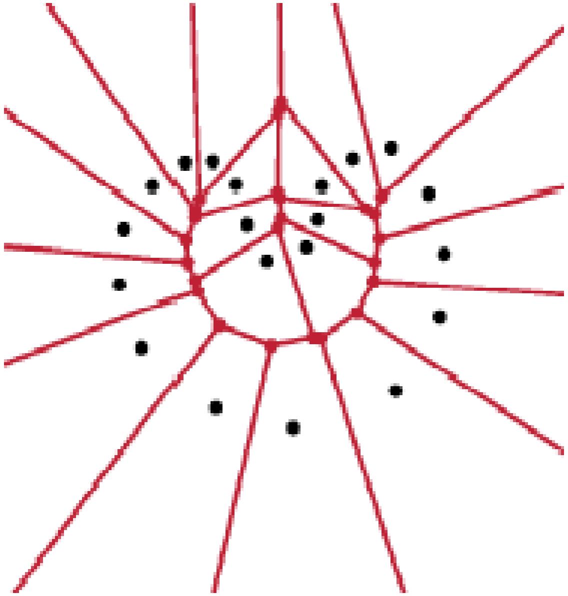
Crust in 2D

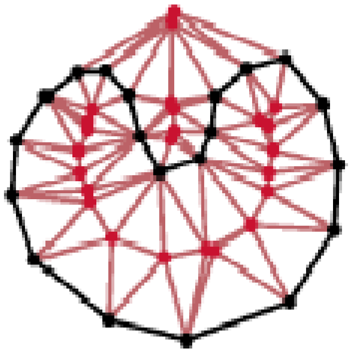
Input : P = Set of sample points in the plane
Output: E = Set of edges connecting points in P

The Algorithm

Compute the Voronoi vertices of P = V
Calculate the Delaunay of (P U V)
Pick the edges (p,q) where both p,q are in P

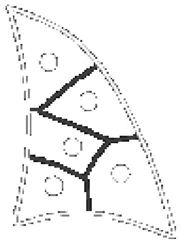
Sample Output

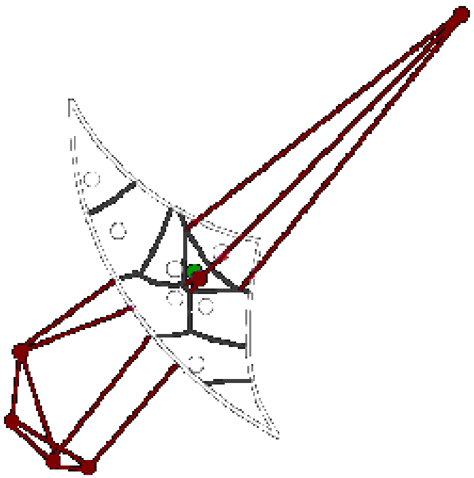


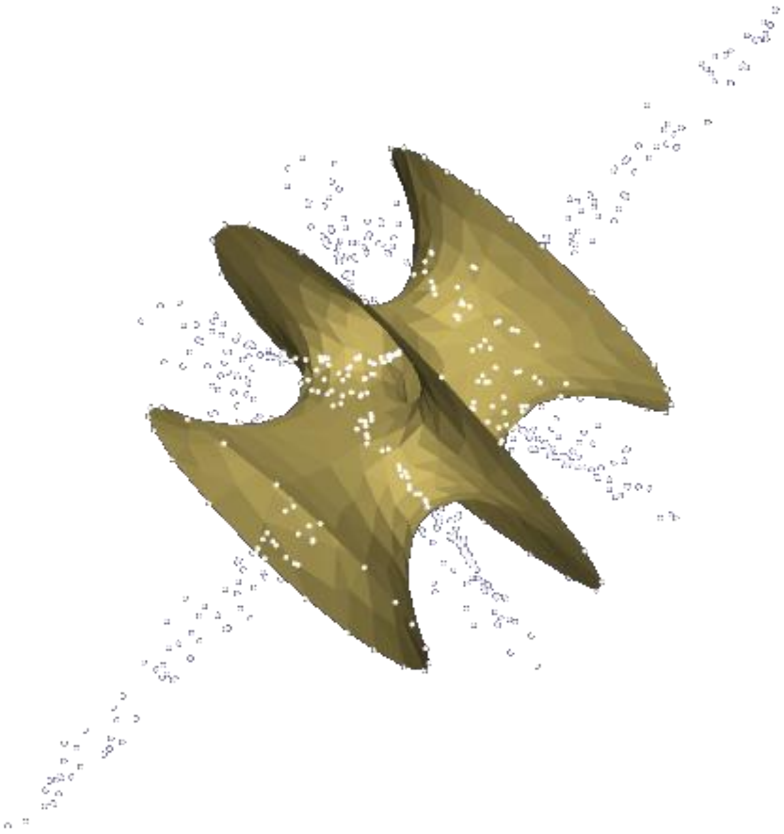


Crust in 3D

- Some Voronoi vertices lie neither near the surface nor near the medial axis
- Keep the “poles”



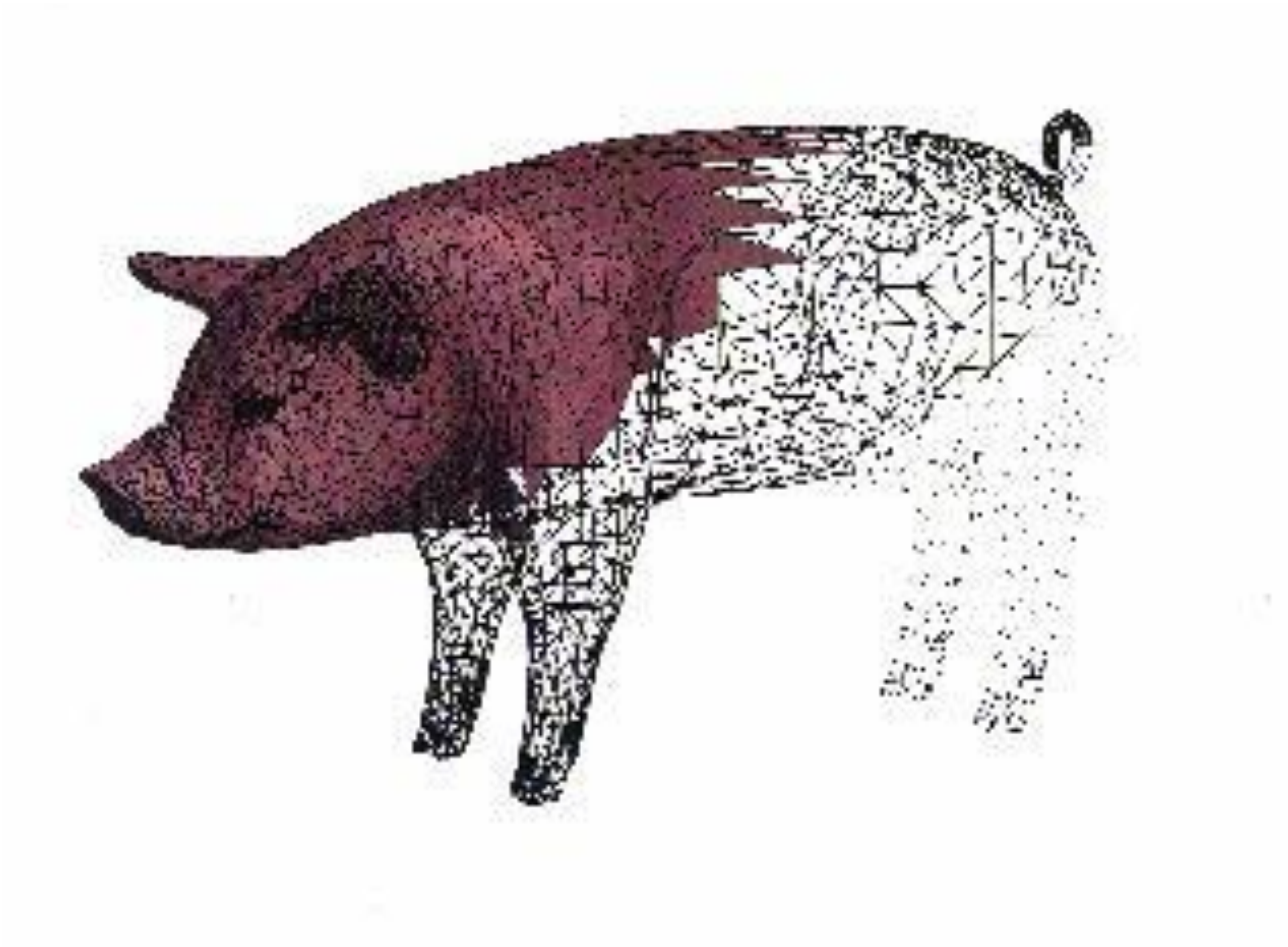




Crust in 3D

- Compute the 3D Voronoi diagram of the sample points.
- For each sample point s, pick the farthest vertex v of its Voronoi cell, and the farthest vertex v' such that angle vsv' exceeds 90 degrees.
- Compute the Voronoi diagram of the sample points and the "poles", the Voronoi vertices chosen in the second step.
- Add a triangle on each triple of sample points with neighboring cells in the second Voronoi diagram.

Sample Output

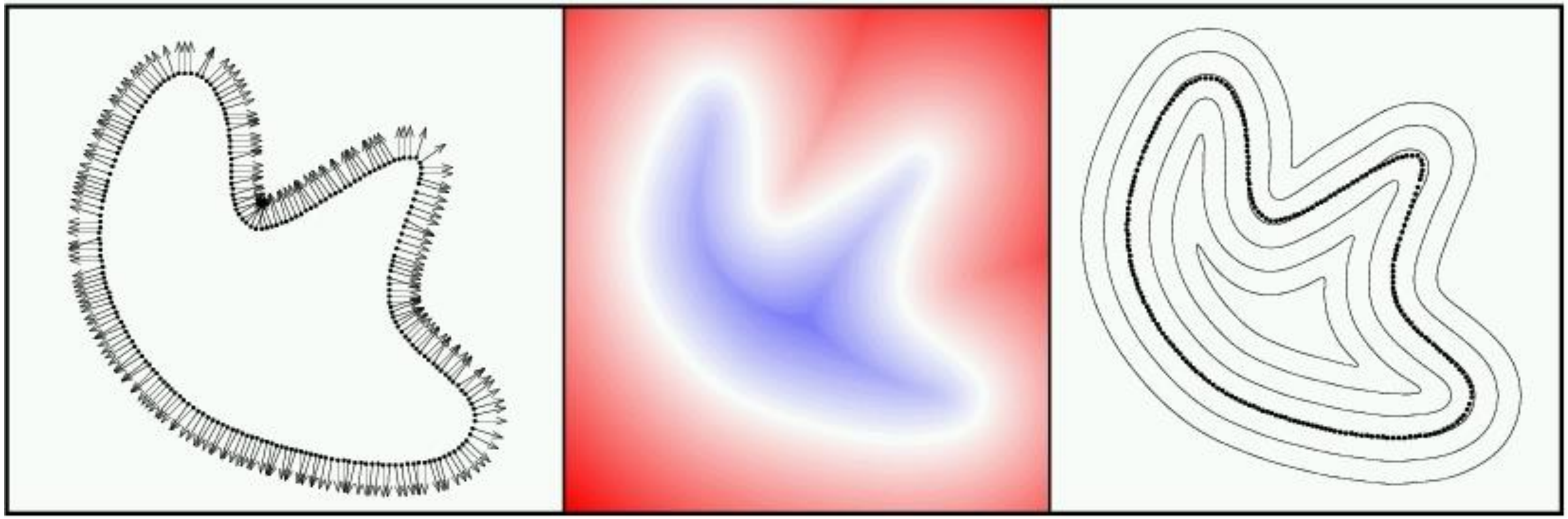


Possible Approaches

- Explicit Meshing
  - Ball pivoting algorithm
  - Crust
  - etc.
- Implicit Reconstruction
  - Hoppe’s algorithm
  - Moving Least Squares (MLS)
  - Poisson surface reconstruction
  - etc.
- Surface fitting
  - Deformable templates
  - etc.

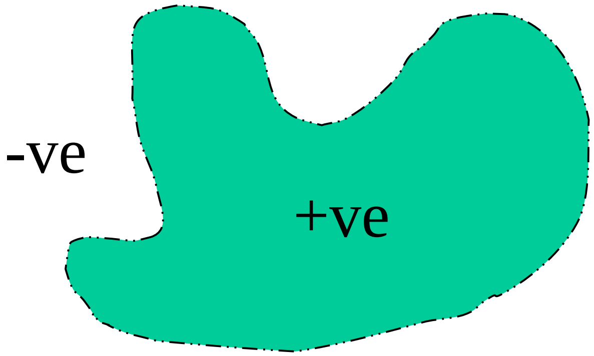
Implicit Reconstruction

- Main idea:
  - Compute an implicit function f(p) (negative outside, positive inside)
  - Extract surface where f(p)=0



Hoppe et al's Algorithm

1. Tangent Plane Estimation
2. Consistent tangent plane orientation
3. Signed distance function computation
4. Surface extraction



Tangent Plane Estimation

- Principal Component Analysis (PCA)
  - Extract points $\{q_i\}$ in neighborhood
  - Compute covariance matrix M
  - Analyze eigenvalues and eigenvectors of M (via SVD)

$$\mathbf{M} = \frac{1}{n}\sum_{i=1}^{n}\begin{bmatrix} q_i^x q_i^x & q_i^x q_i^y & q_i^x q_i^z \\ q_i^y q_i^x & q_i^y q_i^y & q_i^y q_i^z \\ q_i^z q_i^x & q_i^z q_i^y & q_i^z q_i^z \end{bmatrix}$$

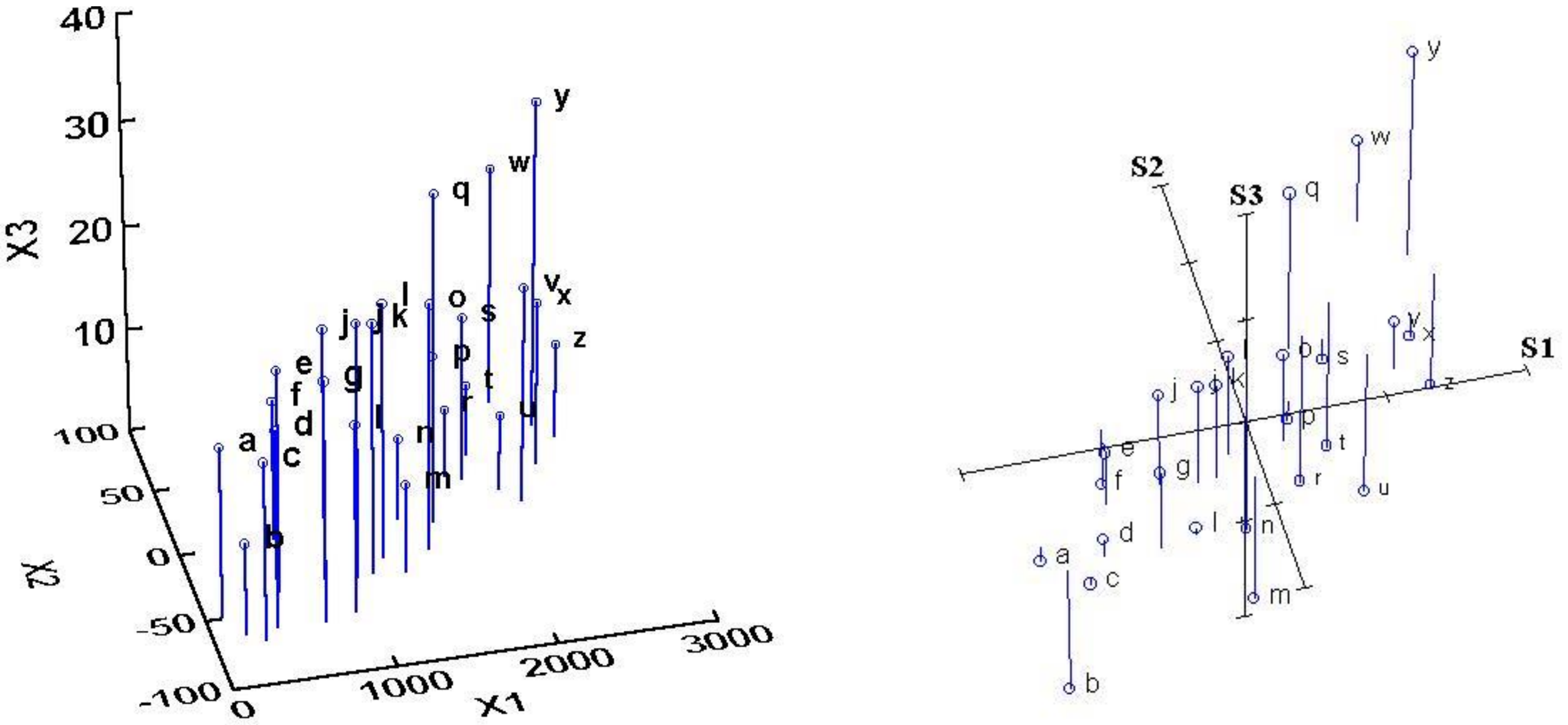
Covariance Matrix

$$\mathbf{M} = \mathbf{USU}^t$$

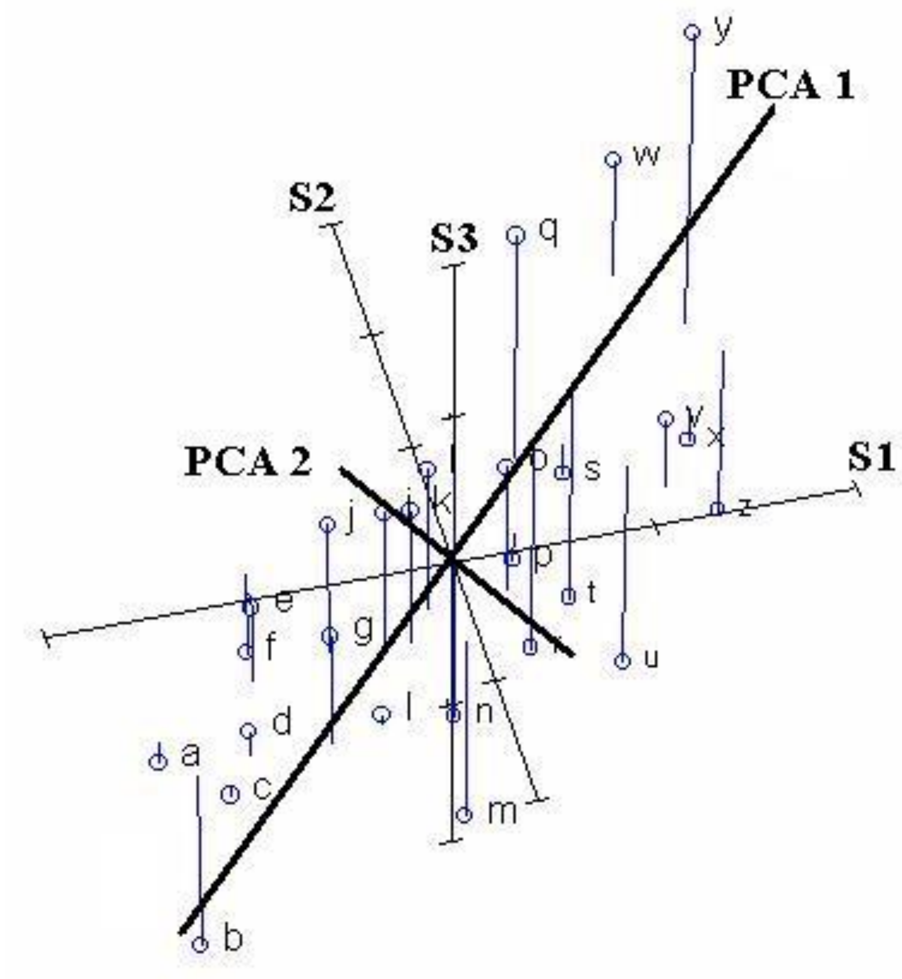
$$\mathbf{S} = \begin{bmatrix} \lambda_a & 0 & 0 \\ 0 & \lambda_b & 0 \\ 0 & 0 & \lambda_c \end{bmatrix} \quad \mathbf{U} = \begin{bmatrix} A_x & A_y & A_z \\ B_x & B_y & B_z \\ C_x & C_y & C_z \end{bmatrix}$$

Eigenvalues & Eigenvectors

Tangent Plane Estimation

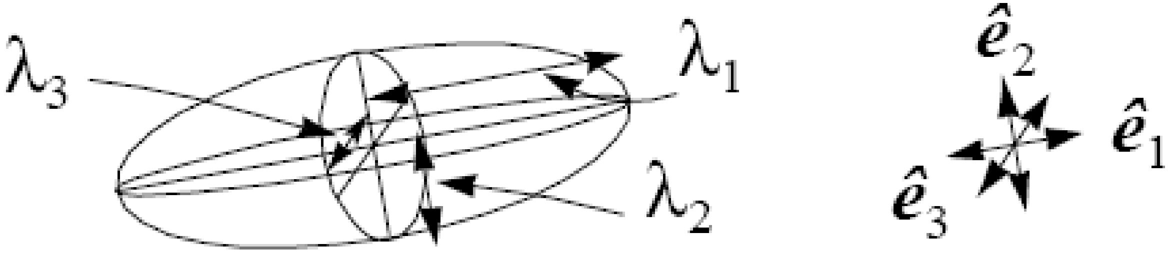


Tangent Plane Estimation



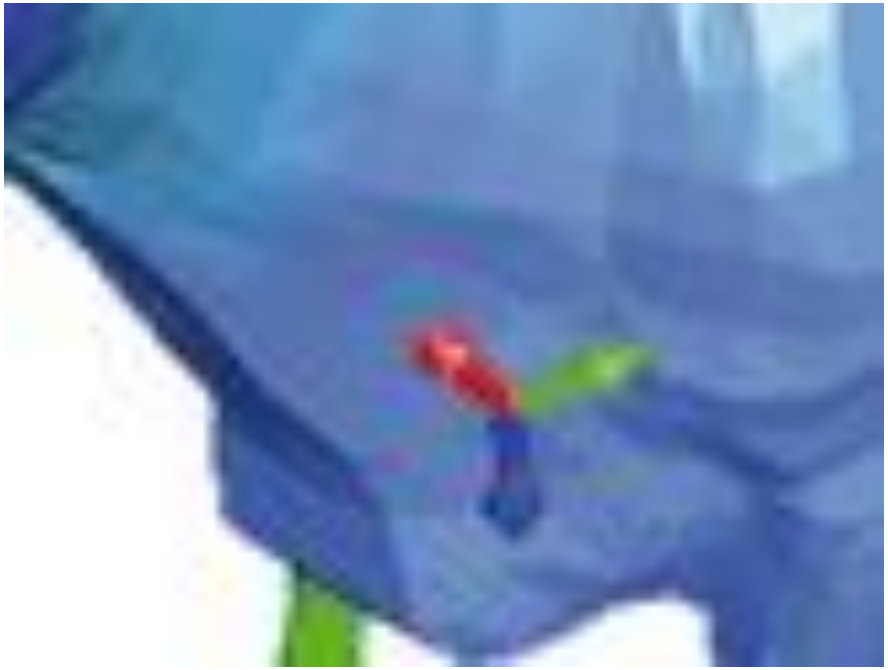
Tangent Plane Estimation

- Eigenvectors are "Principal Axes of Inertia"
- Eigenvalues are variances of the point distribution in those directions



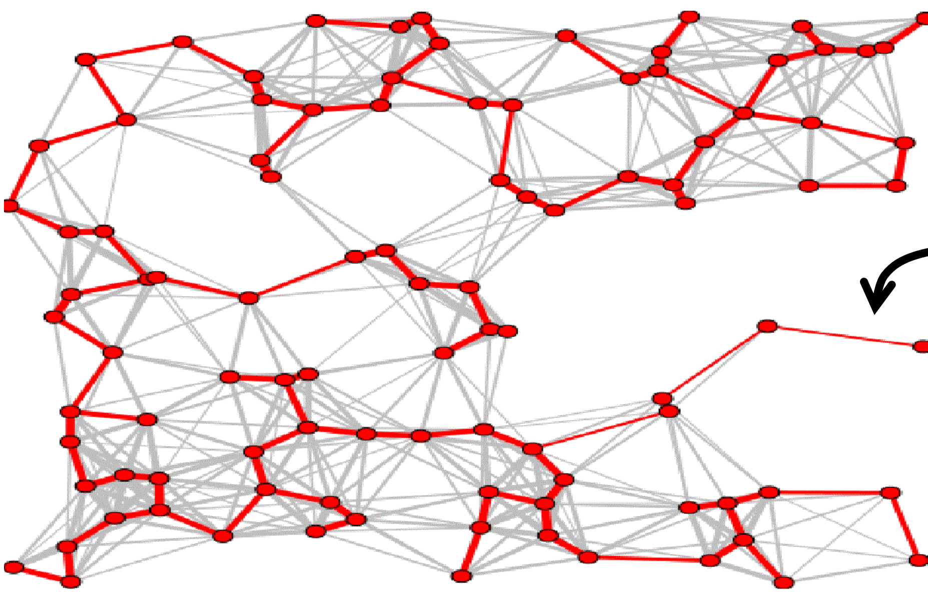
Tangent Plane Estimation

- Surface normal is estimated by eigenvector (principal axis) associated with smallest eigenvalue



Consistent Tangent Plane Orientation

- Traverse nearest neighbor graph flipping normals for consistency
  - Greedy propagation algorithm (minimum spanning tree of normal similarity)



$$1 - |\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j|$$

Signed Distance Function

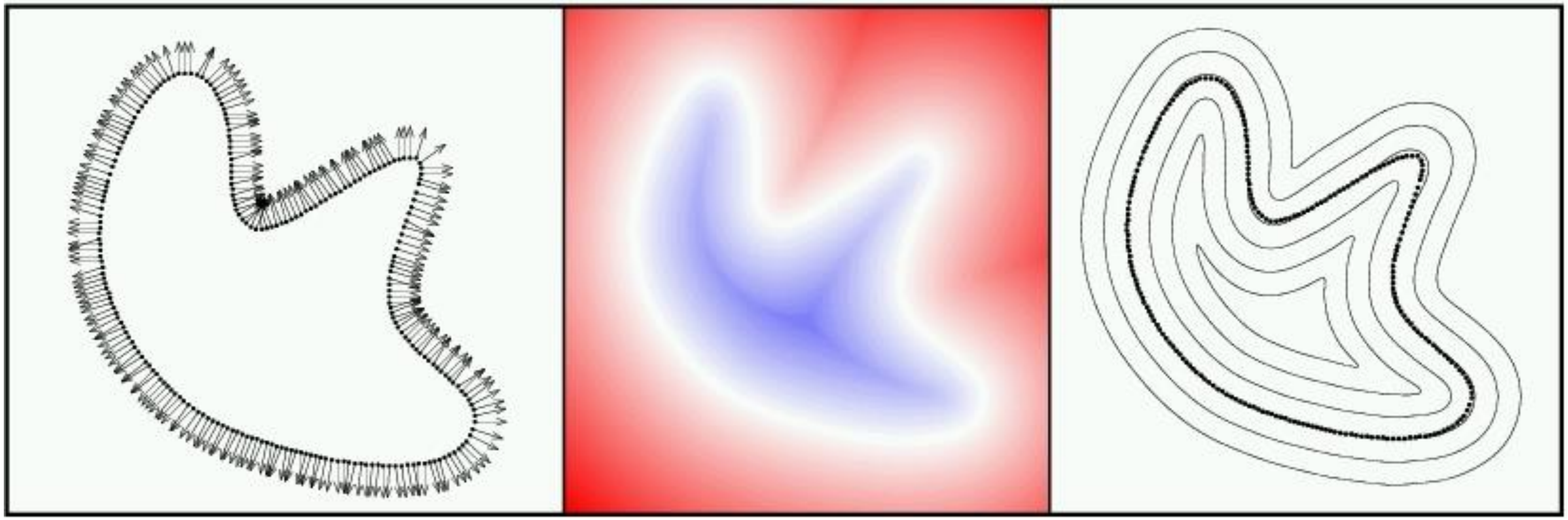
- f(p) is signed distance to tangent plane of closest point sample

$\{$ *Compute* $\mathbf{z}$ *as the projection of* $\mathbf{p}$ *onto* $Tp(\mathbf{x}_i)$ $\}$
$\mathbf{z} \leftarrow \mathbf{o}_i - ((\mathbf{p} - \mathbf{o}_i) \cdot \hat{\mathbf{n}}_i)\, \hat{\mathbf{n}}_i$

if $d(\mathbf{z}, X) < \rho + \delta$ **then**
$\quad f(\mathbf{p}) \leftarrow (\mathbf{p} - \mathbf{o}_i) \cdot \hat{\mathbf{n}}_i \qquad \{= \pm \|\mathbf{p} - \mathbf{z}\|\}$
else
$\quad f(\mathbf{p}) \leftarrow$ **undefined**
endif

Signed Distance Function

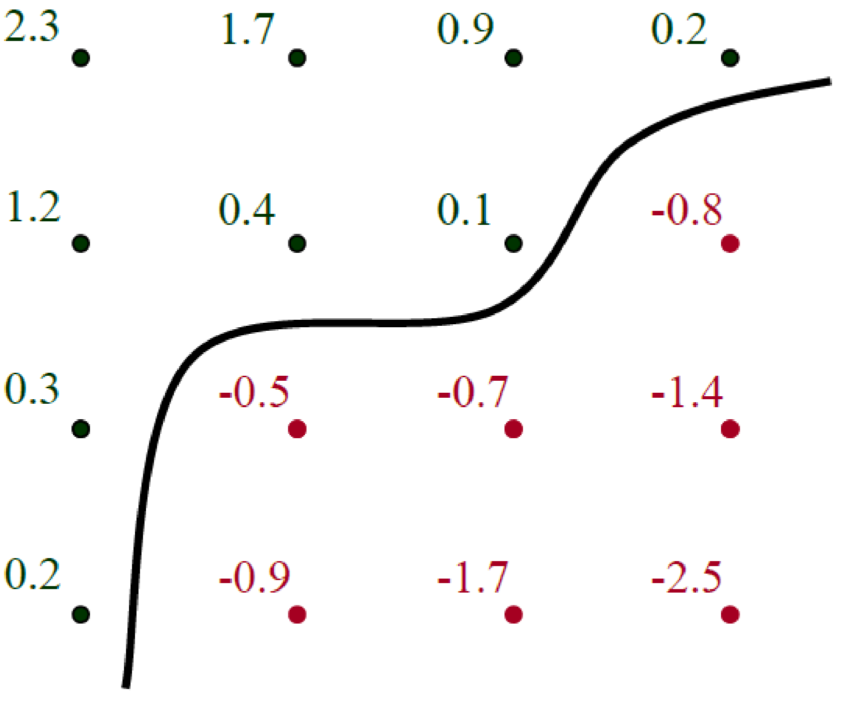
- f(p) is signed distance to tangent plane of closest point sample



Ravikrishna Bvs Kolluri

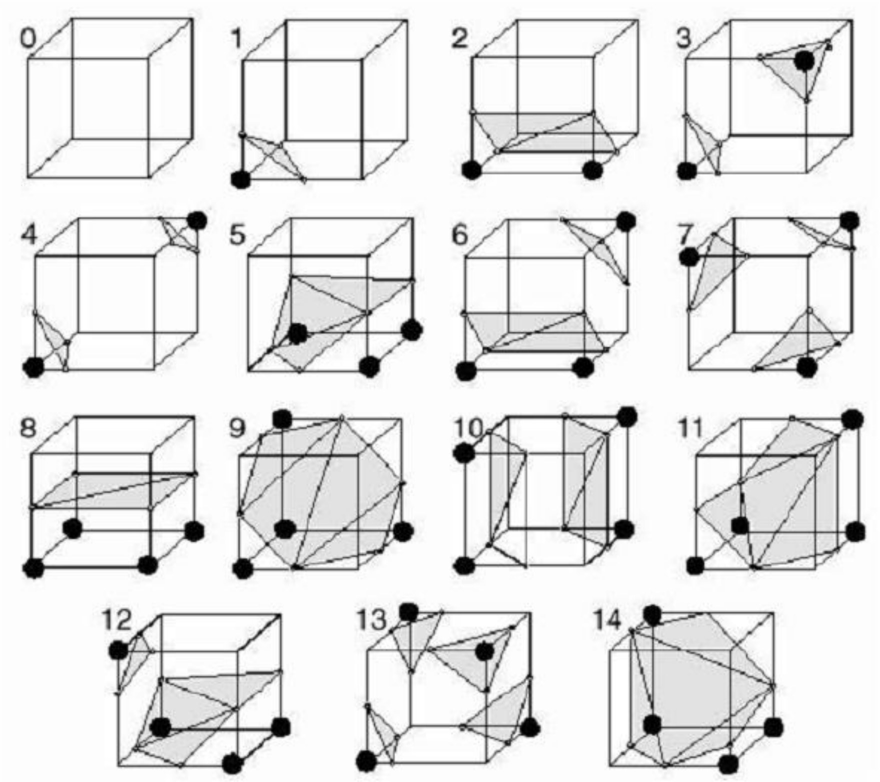
Surface Extraction

- Extract triangulated surface where f(p)=0
  - e.g., Marching Cubes

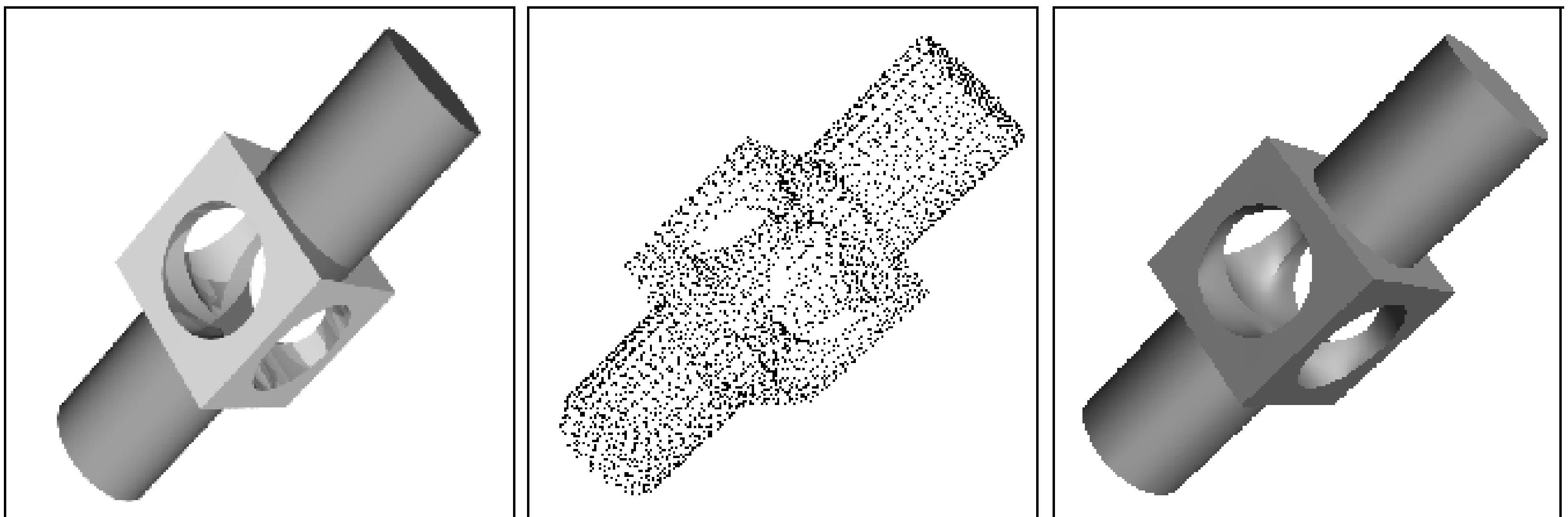


Surface Extraction

- Extract triangulated surface where f(p)=0
  - e.g., Marching Cubes

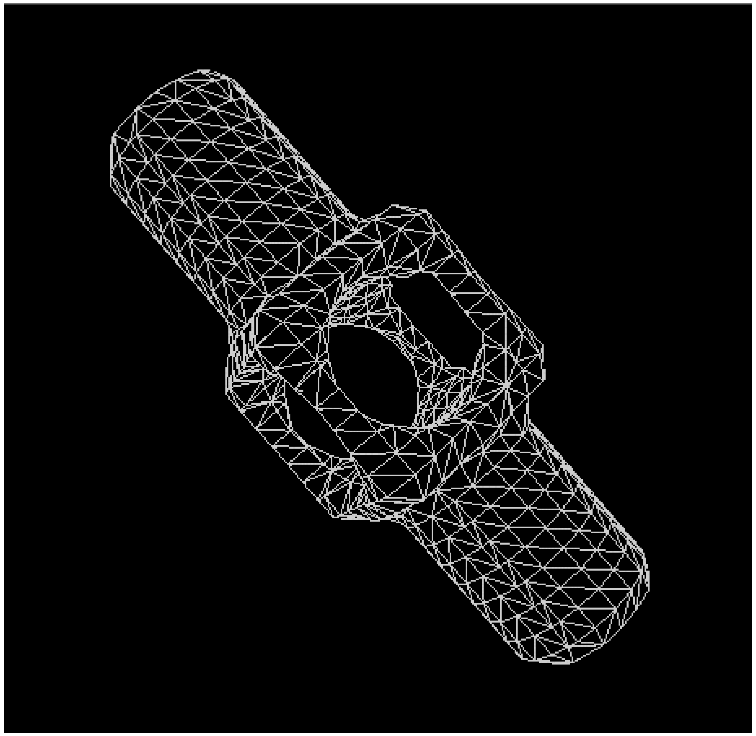


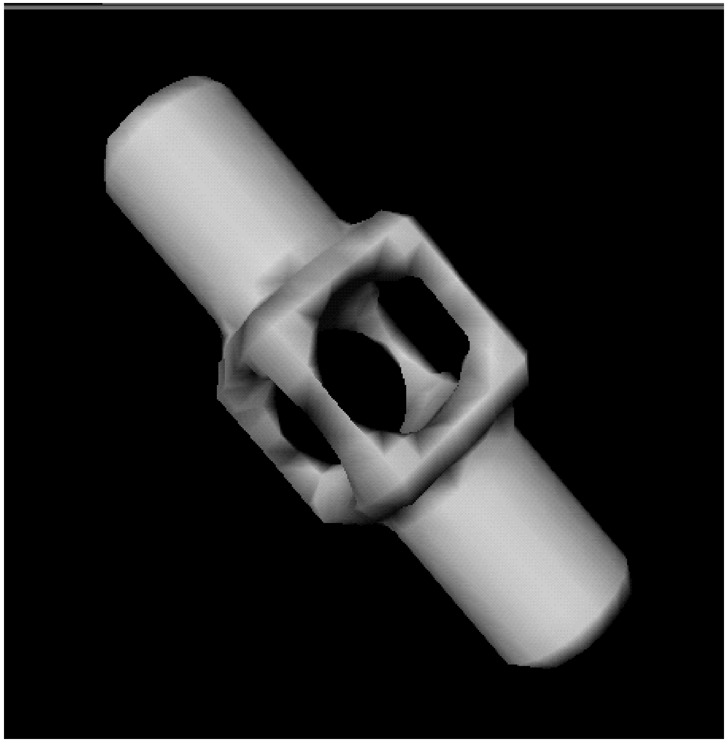
Sample Results



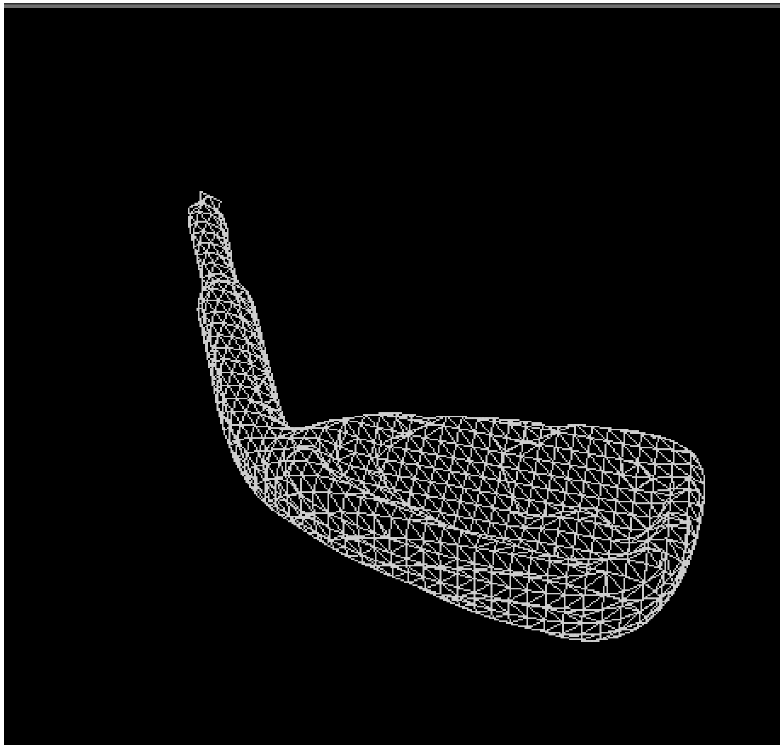
(a) Unknown surface U (b) Sampled points X (c) Reconstructed surface S

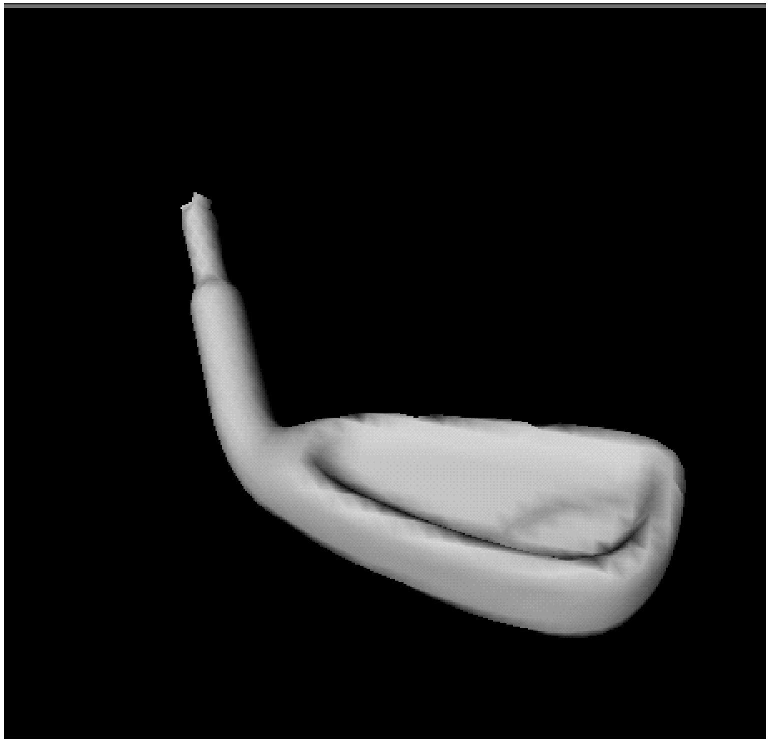
Sample Results



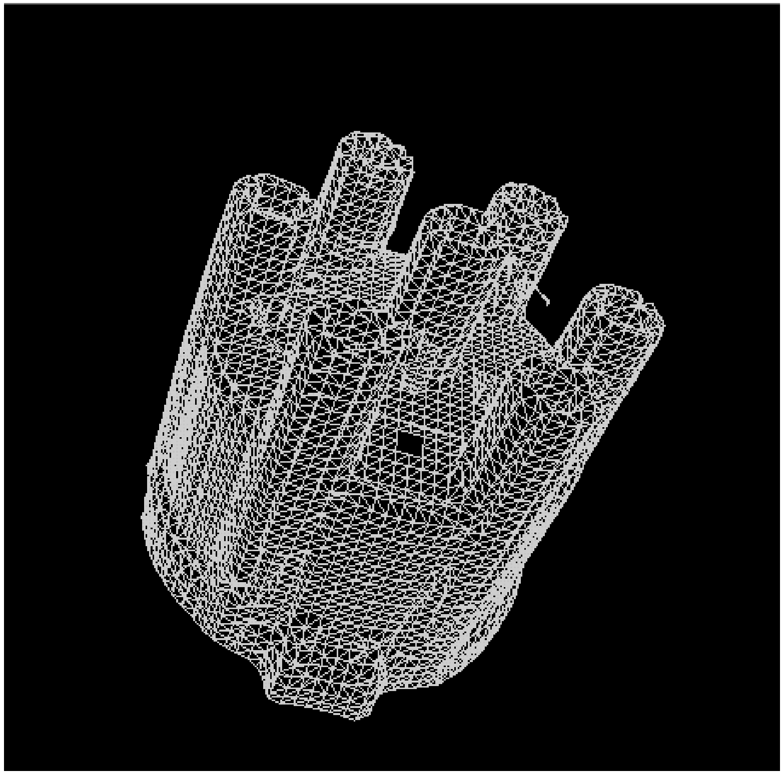


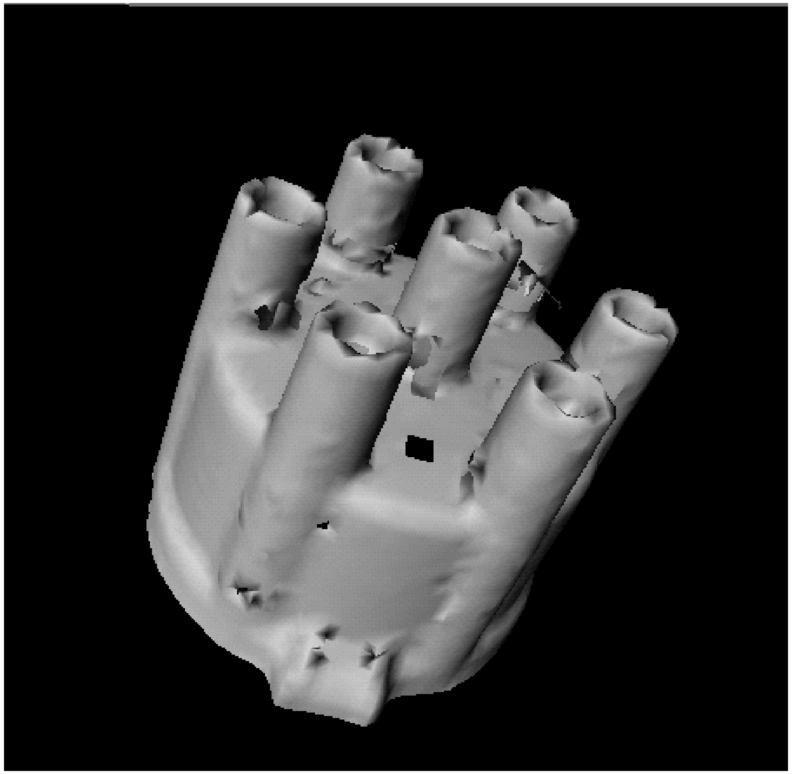
Sample Results





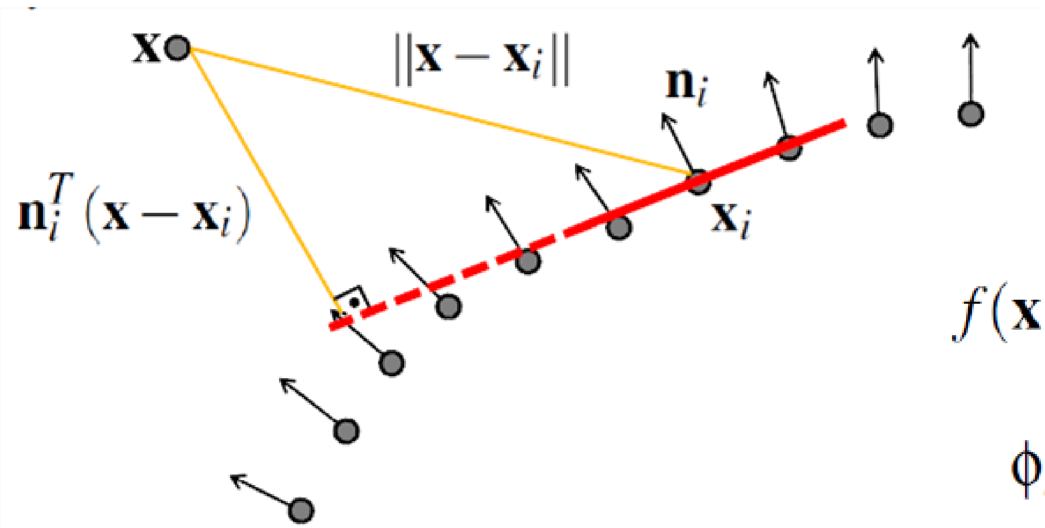
Sample Results





Moving Least Squares

- Similar, but different implicit function
  - Weighted contribution of nearby points

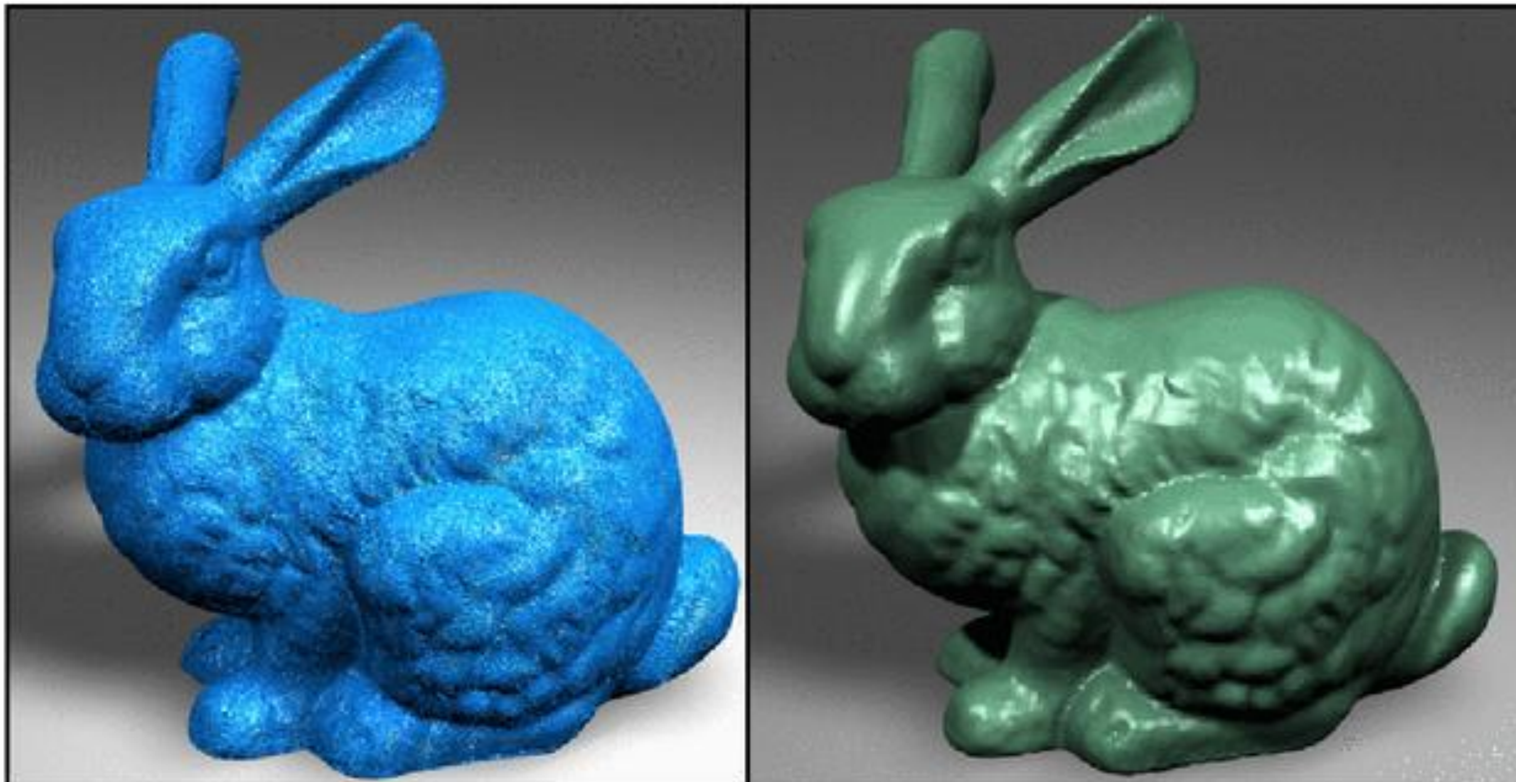


$$f(\mathbf{x}) = \frac{\sum \mathbf{n}_i^T(\mathbf{x}-\mathbf{x}_i)\phi_i(\mathbf{x})}{\sum \phi_i(\mathbf{x})}$$

$$\phi_i(\mathbf{x}) = \phi(\|\mathbf{x}-\mathbf{x}_i\|)$$

$$\phi(x) = e^{-\left(\frac{x}{\sigma_r}\right)^2}$$

Moving Least Squares



MLS

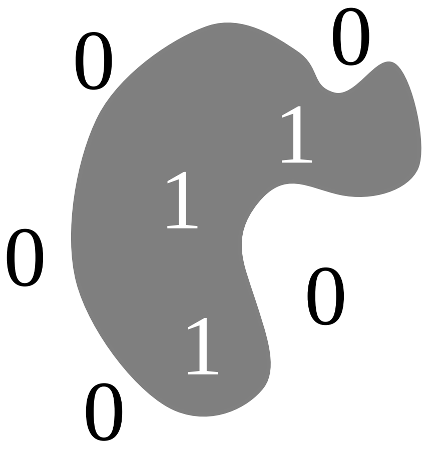
Possible Approaches

- Explicit Meshing
  - Ball pivoting algorithm
  - Crust
  - etc.
- Implicit Reconstruction
  - Hoppe's algorithm
  - Moving Least Squares (MLS)
  - Poisson surface reconstruction
  - etc.
- Surface fitting
  - Deformable templates
  - etc.

The Indicator Function

- We reconstruct the surface of the model by solving for the indicator function of the shape.

$$\chi_M(p) = \begin{cases} 1 & \text{if } p \in M \\ 0 & \text{if } p \notin M \end{cases}$$

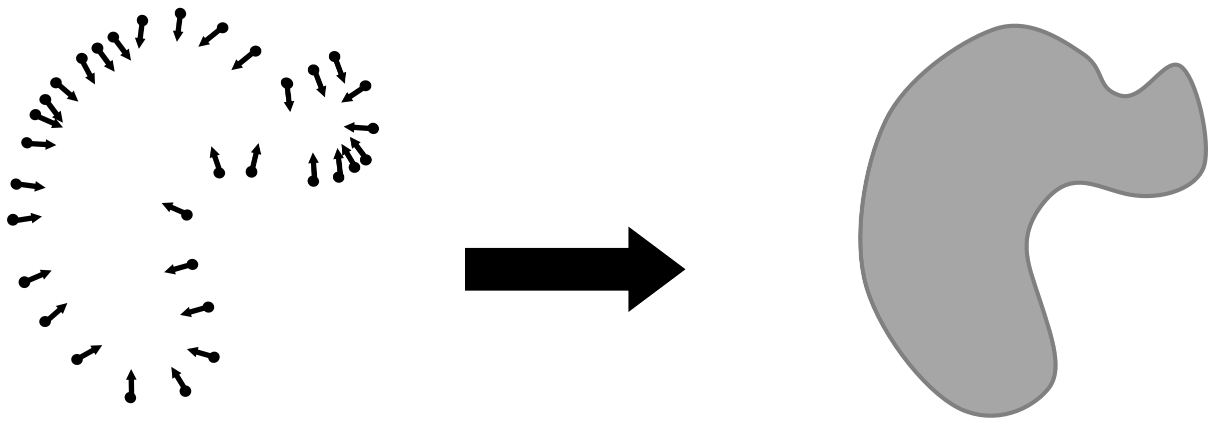


Indicator function

χ_M

Challenge

- How to construct the indicator function?



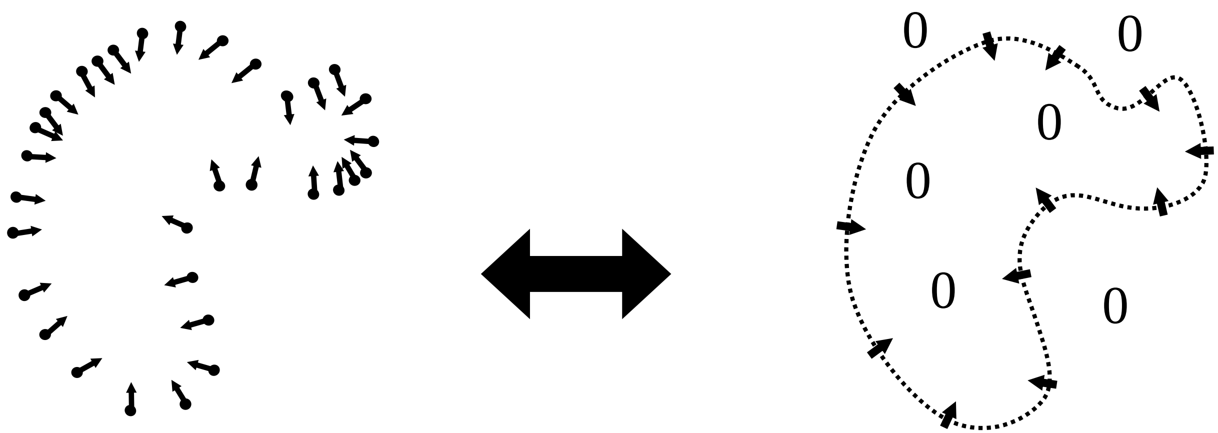
Oriented points

Indicator function

χ_M

Gradient Relationship

- There is a relationship between the normal field and gradient of indicator function



Oriented points

Indicator gradient
$\nabla \chi_M$

Integration

- Represent the points by a vector field $\vec{V}$
- Find the function χ whose gradient best approximates $\vec{V}$:

$$\min_{\chi} \left\| \nabla \chi - \vec{V} \right\|$$

Integration as a Poisson Problem

- Represent the points by a vector field $\vec{V}$
- Find the function χ whose gradient best approximates $\vec{V}$:

$$\min_{\chi} \left\| \nabla \chi - \vec{V} \right\|$$

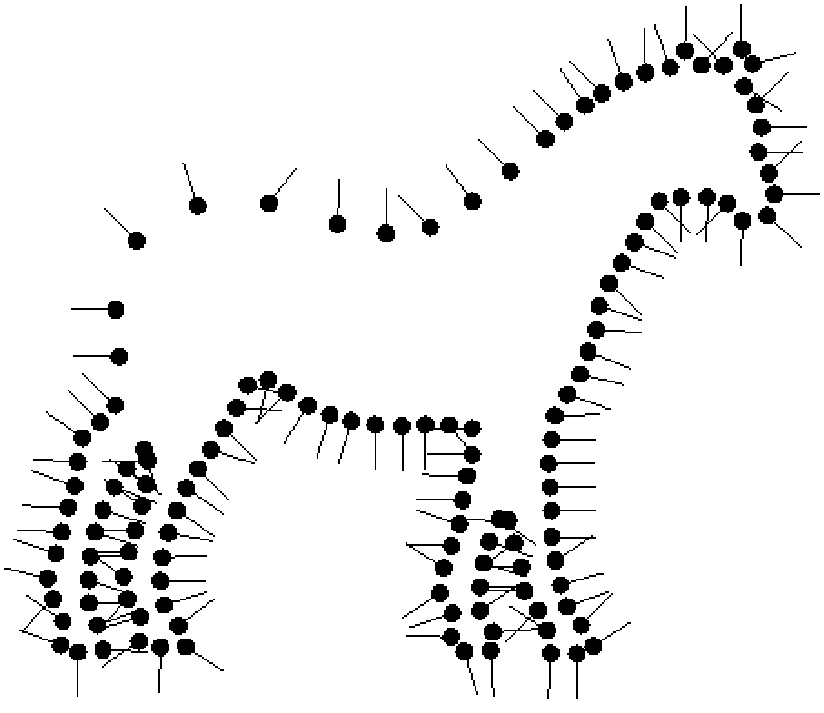
- Applying the divergence operator, we can transform this into a Poisson problem:

$$\nabla \cdot (\nabla \chi) = \nabla \cdot \vec{V} \quad \Leftrightarrow \quad \Delta \chi = \nabla \cdot \vec{V}$$

Implementation

Given the Points:

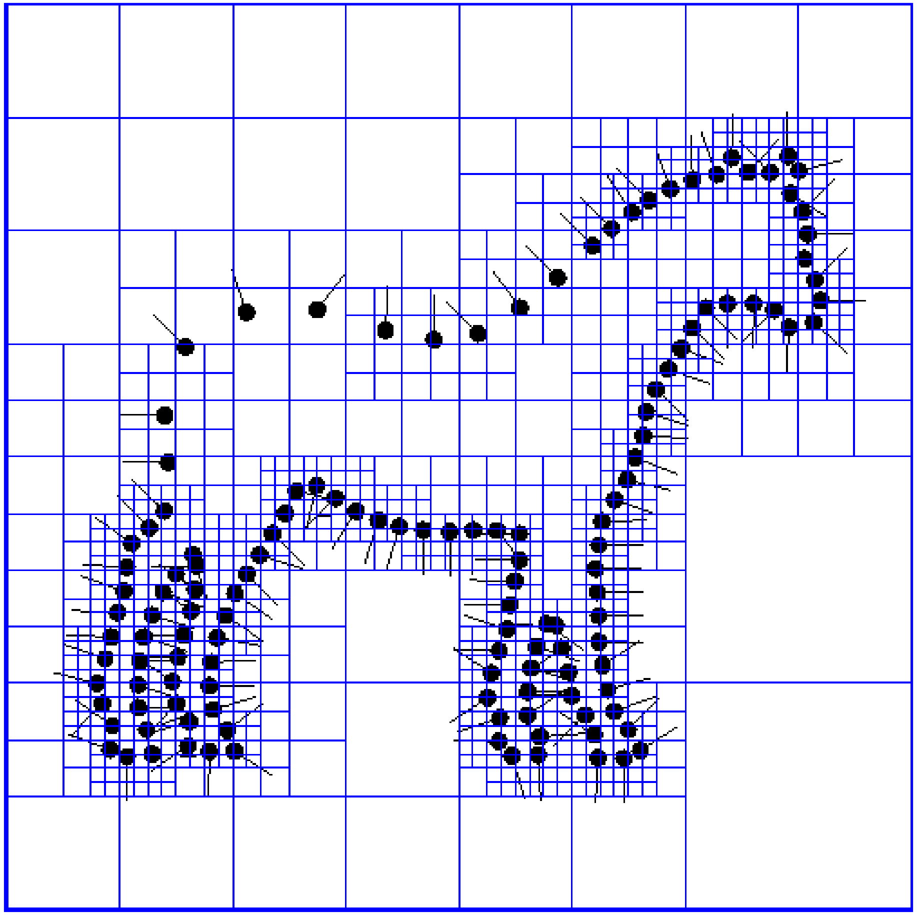
- Set octree
- Compute vector field
- Compute indicator function
- Extract iso-surface



Implementation: Adapted Octree

Given the Points:

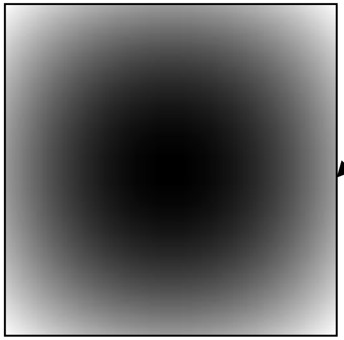
- Set octree
- Compute vector field
- Compute indicator function
- Extract iso-surface

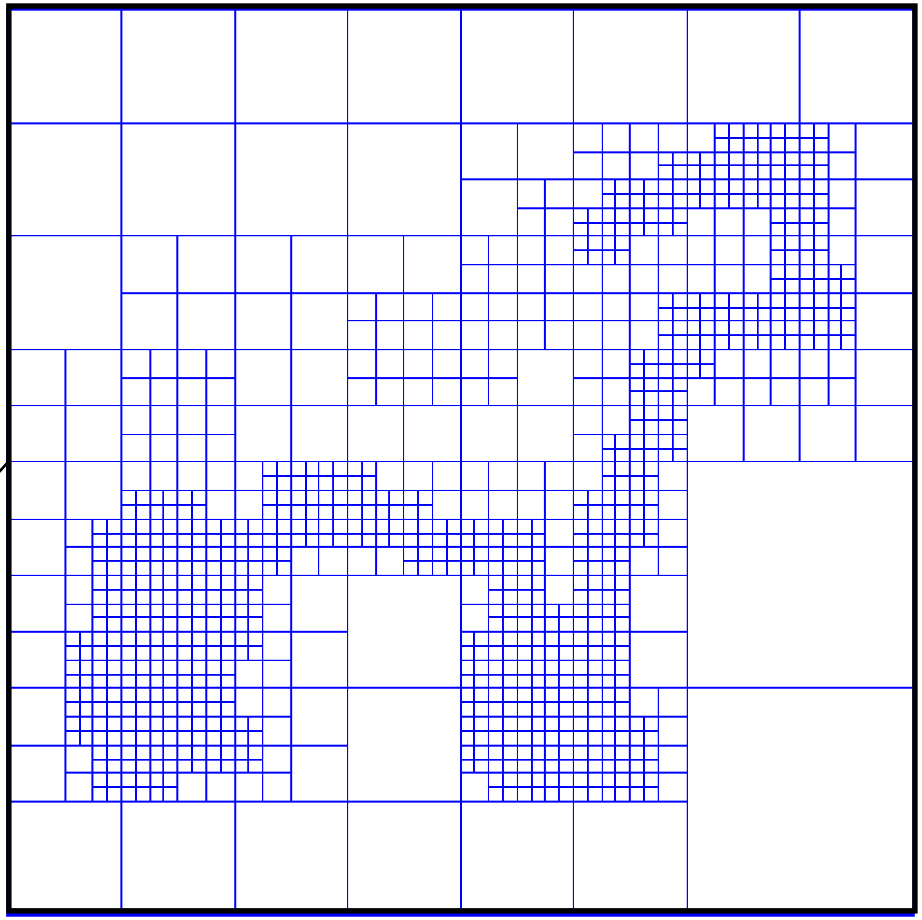


Implementation: Vector Field

Given the Points:

- Set octree
- Compute vector field
  - Define a function space
  - Splat the samples
- Compute indicator function
- Extract iso-surface

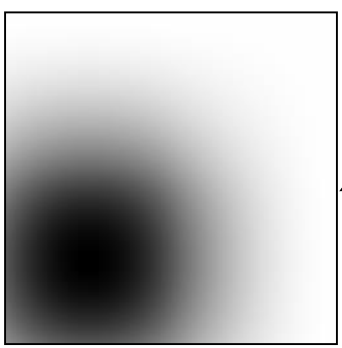


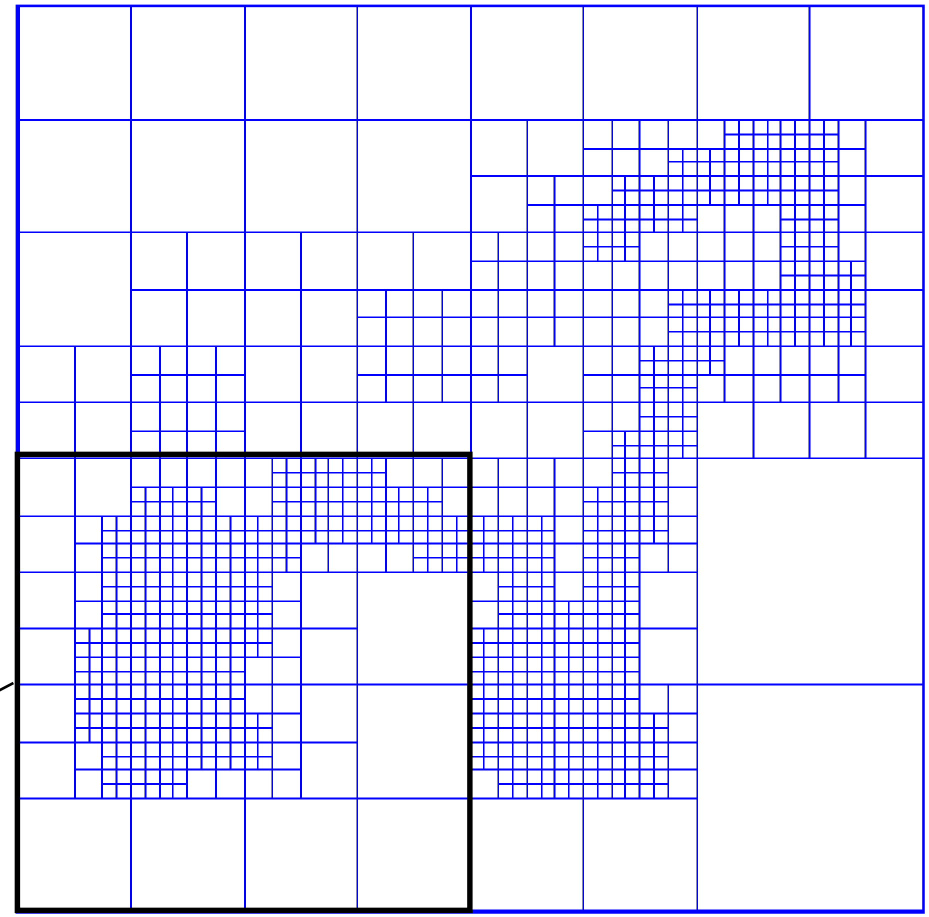


Implementation: Vector Field

Given the Points:

- Set octree
- Compute vector field
  - Define a function space
  - Splat the samples
- Compute indicator function
- Extract iso-surface

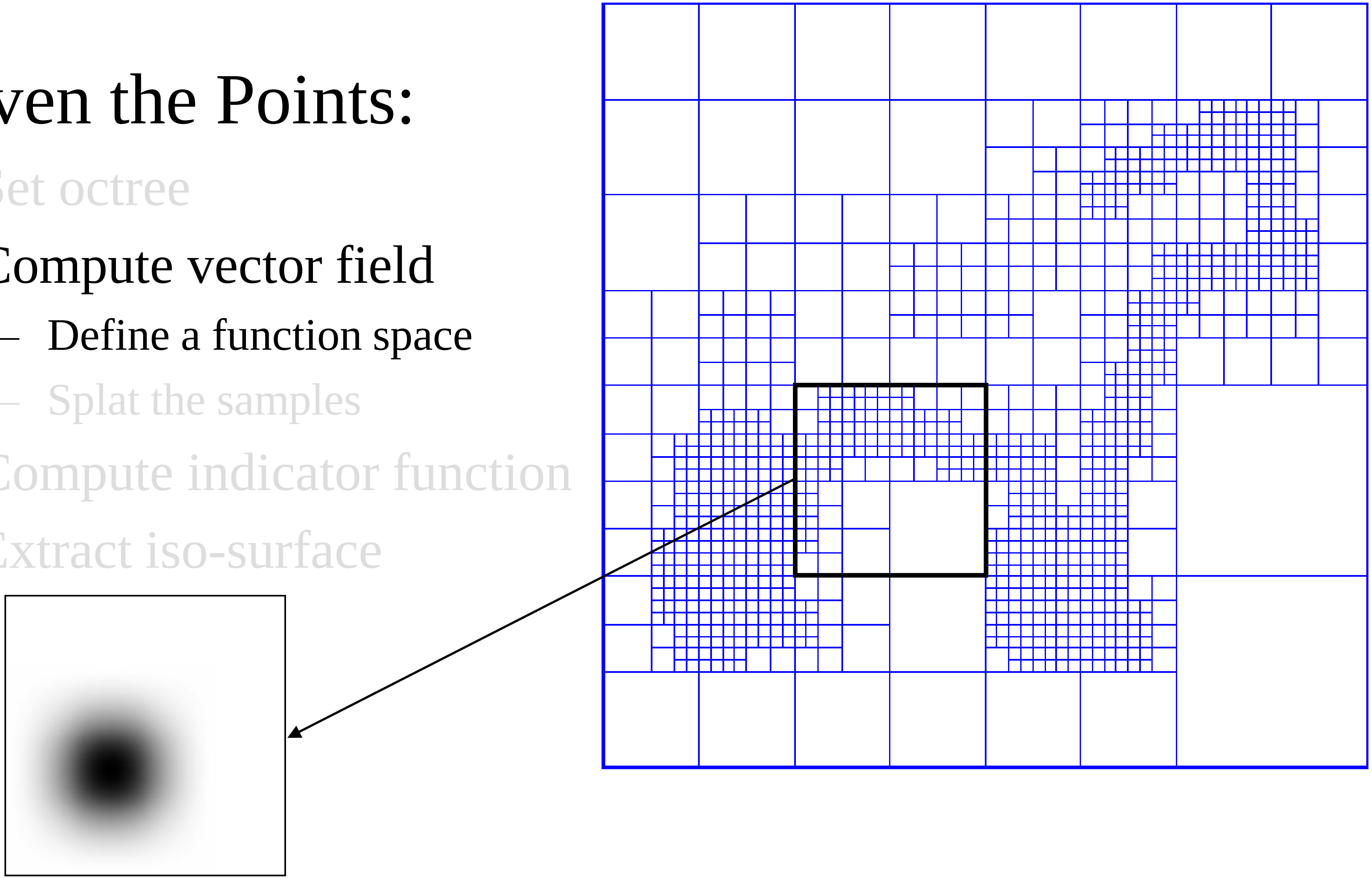




Implementation: Vector Field

Given the Points:

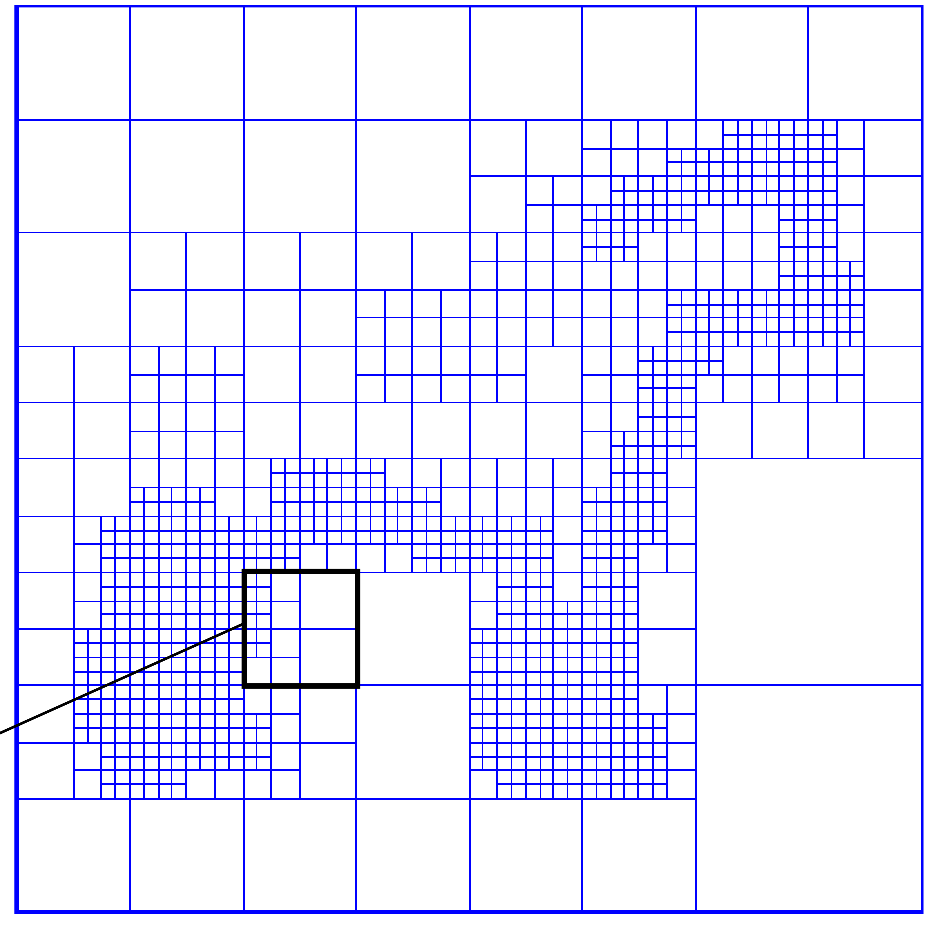
- Set octree
- Compute vector field
  - Define a function space
  - Splat the samples
- Compute indicator function
- Extract iso-surface

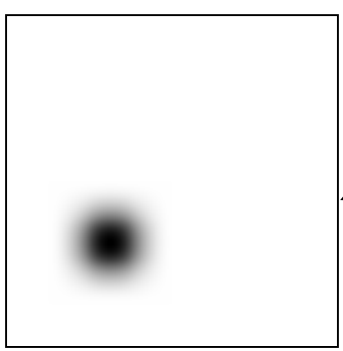


Implementation: Vector Field

Given the Points:

- Set octree
- Compute vector field
  - Define a function space
  - Splat the samples
- Compute indicator function
- Extract iso-surface

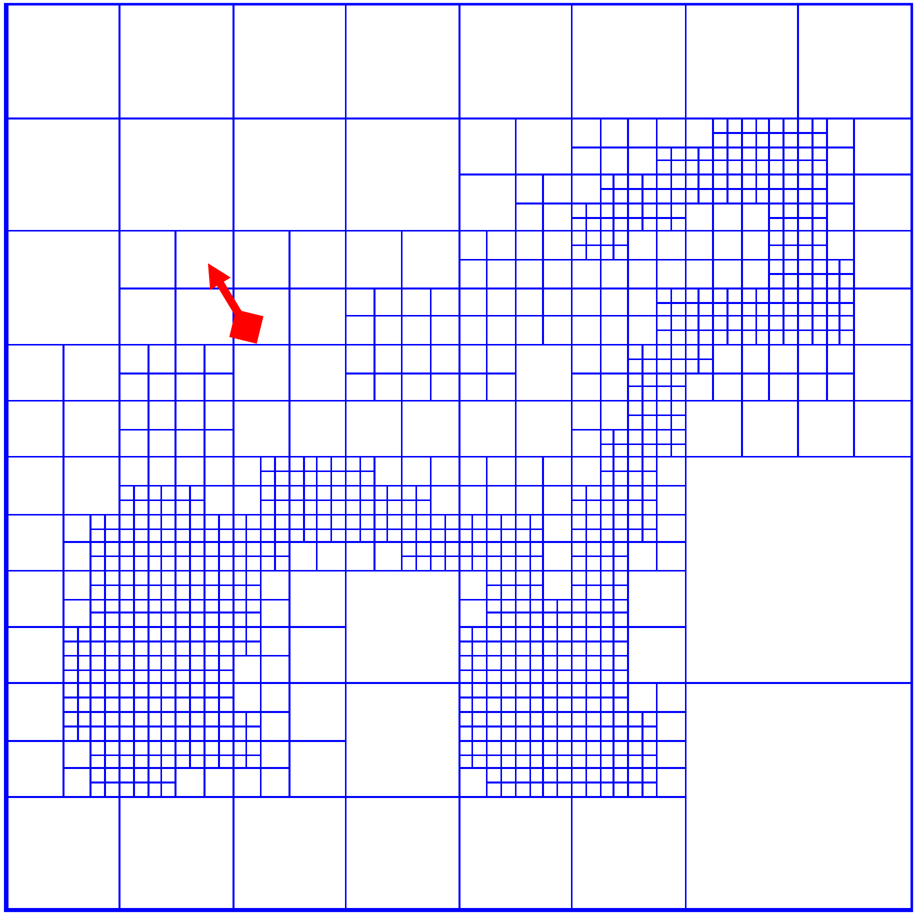




Implementation: Vector Field

Given the Points:

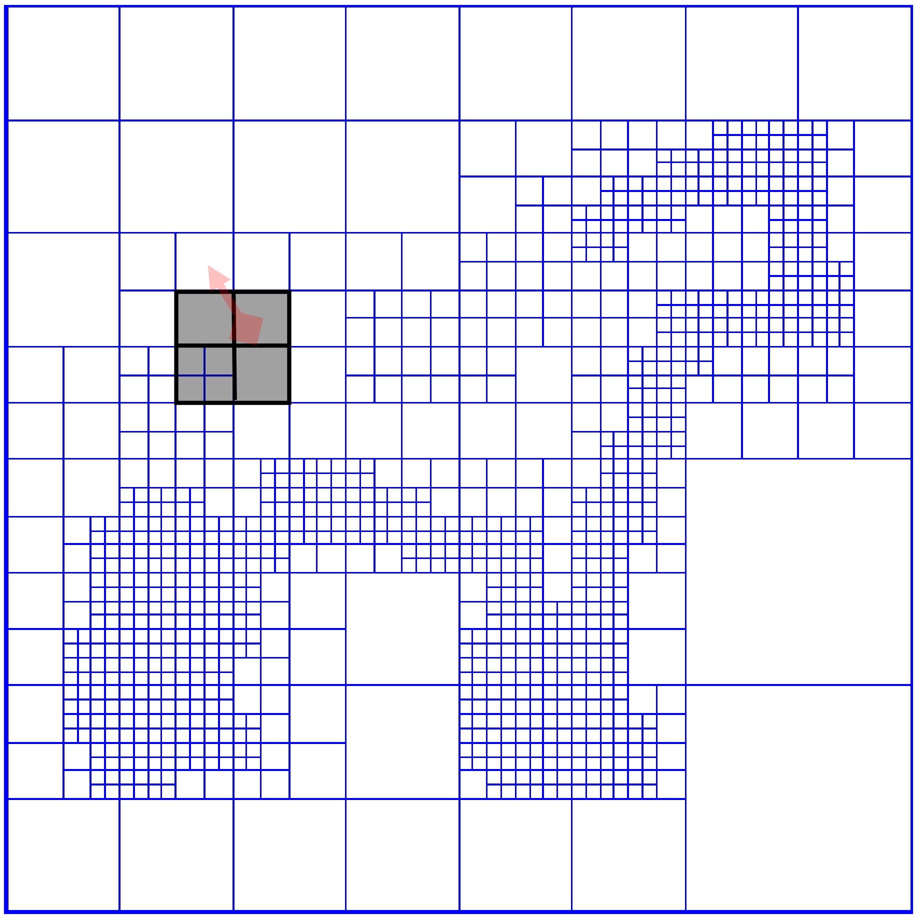
- Set octree
- Compute vector field
  - Define a function basis
  - Splat the samples
- Compute indicator function
- Extract iso-surface



Implementation: Vector Field

Given the Points:

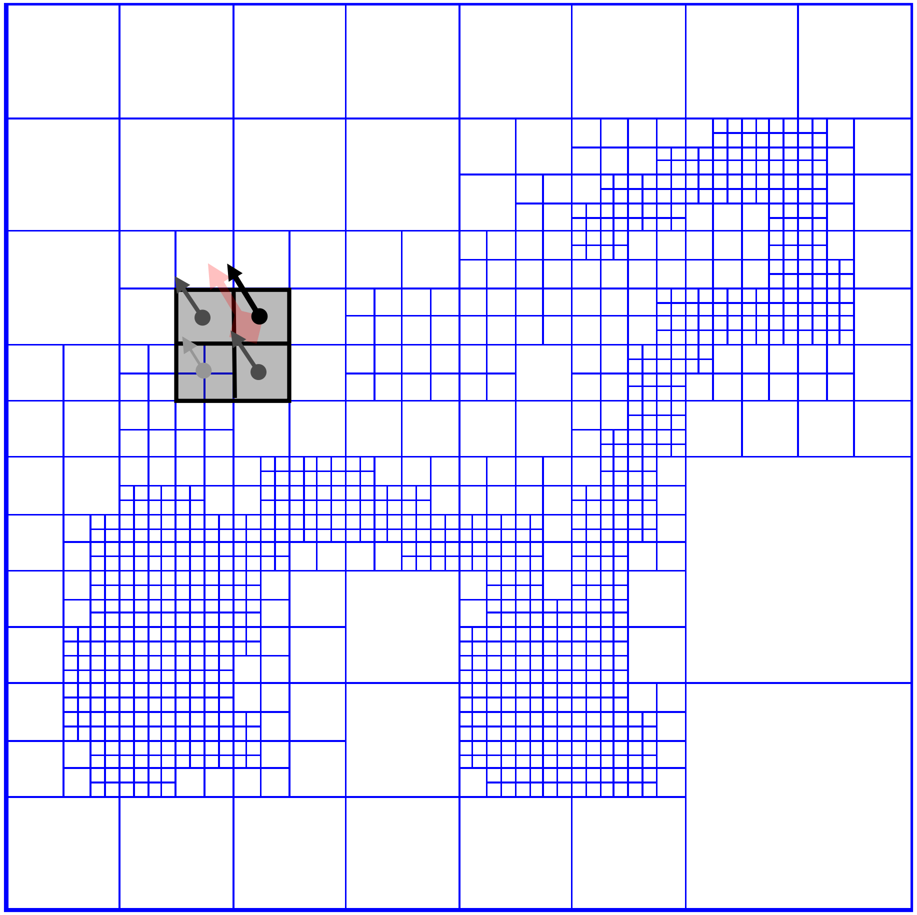
- Set octree
- Compute vector field
  - Define a function basis
  - Splat the samples
- Compute indicator function
- Extract iso-surface



Implementation: Vector Field

Given the Points:

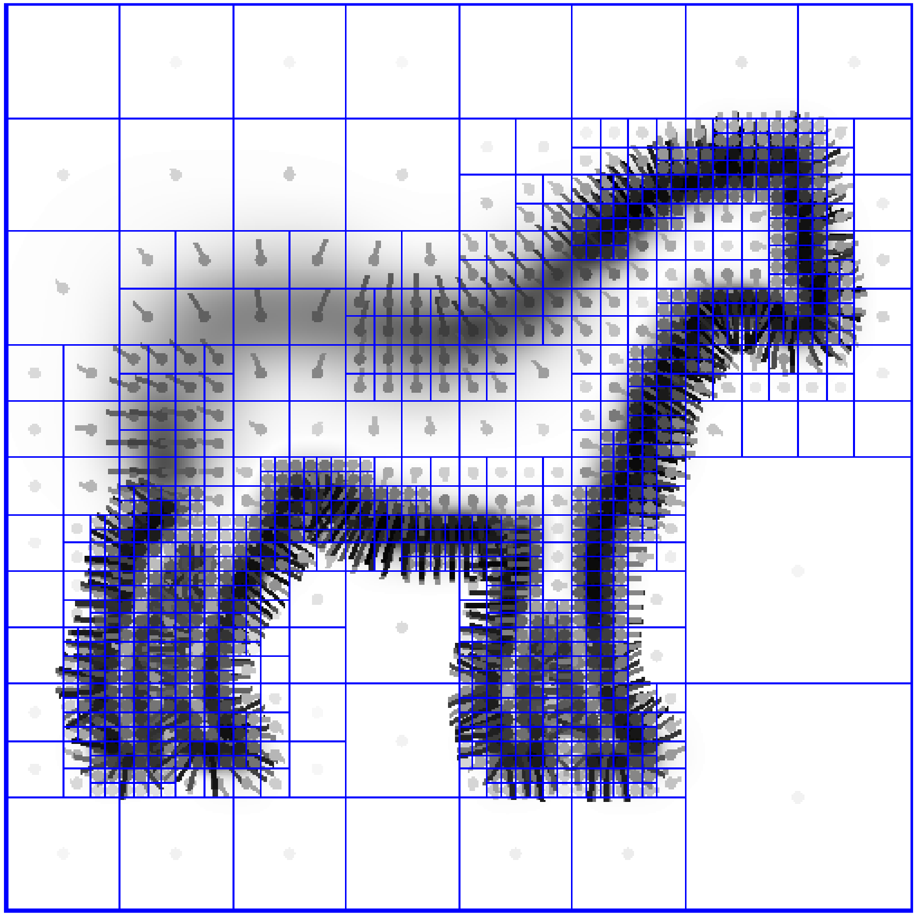
- Set octree
- Compute vector field
  - Define a function basis
  - Splat the samples
- Compute indicator function
- Extract iso-surface



Implementation: Vector Field

Given the Points:

- Set octree
- Compute vector field
  - Define a function space
  - Splat the samples
- Compute indicator function
- Extract iso-surface

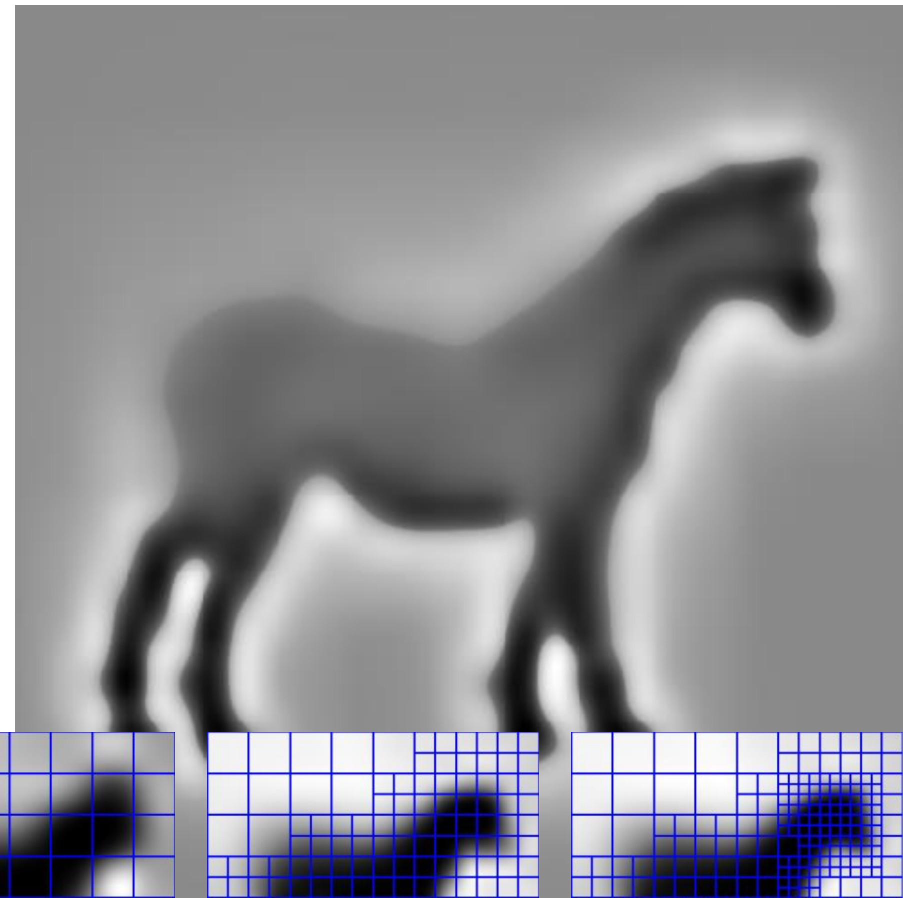


Implementation: Indicator Function

Given the Points:

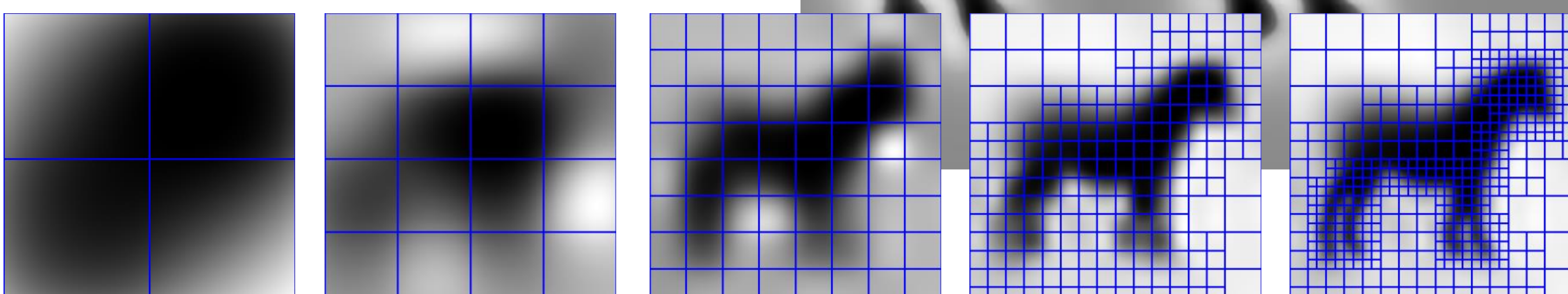
- Set octree
- Compute vector field
- Compute indicator function
  - Compute divergence
  - Solve Poisson equation
- Extract iso-surface

Implementation: Indicator Function

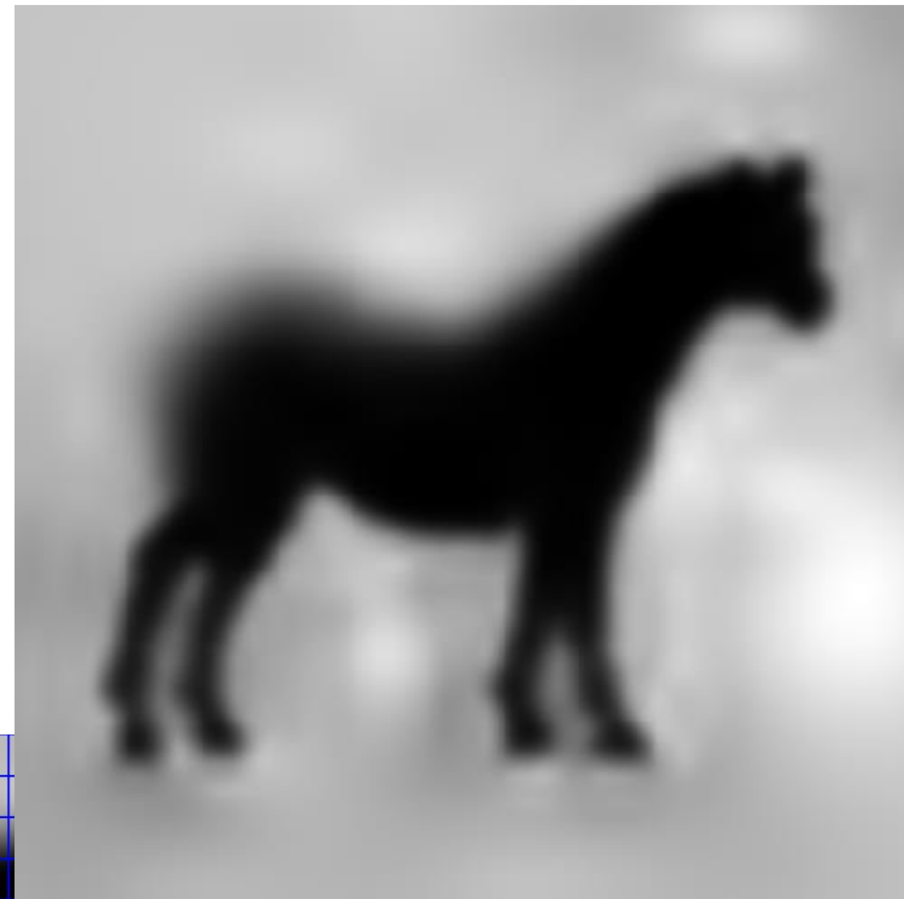


Given the Points:

- Set octree
- Compute vector field
- Compute indicator function
  - Compute divergence
  - Solve Poisson equation
- Extract iso-surface

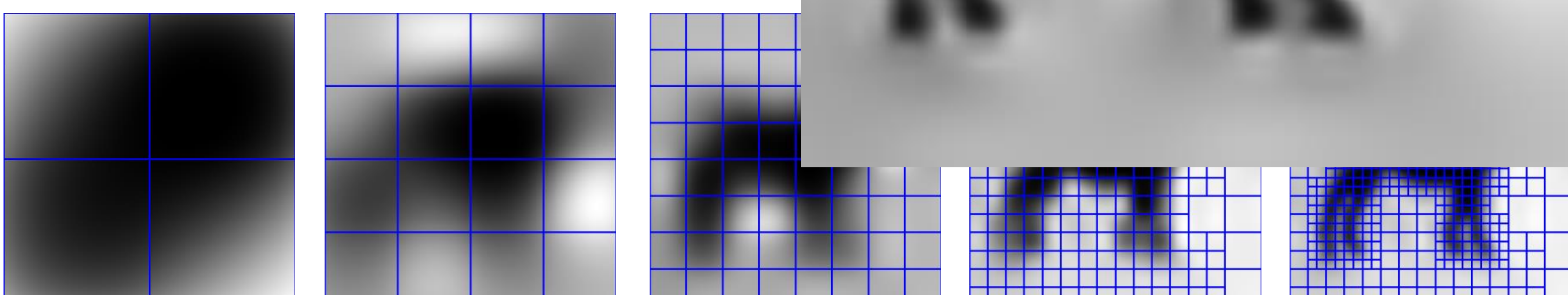


Implementation: Indicator Function



Given the Points:

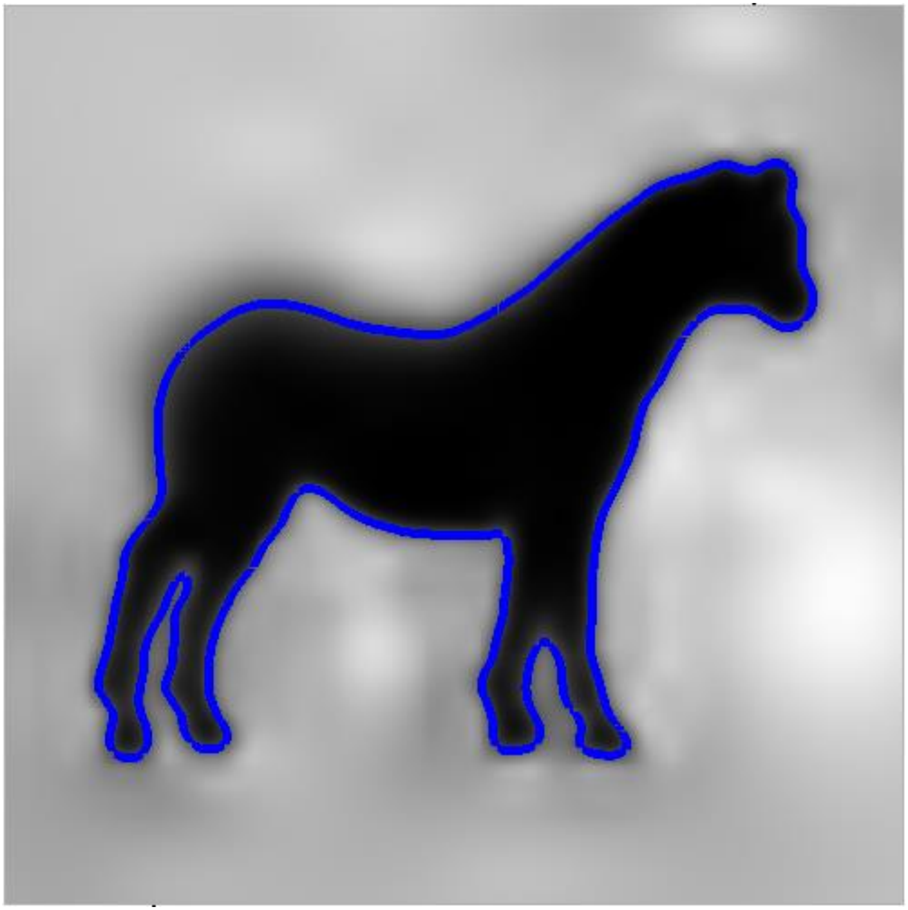
- Set octree
- Compute vector field
- Compute indicator function
  - Compute divergence
  - Solve Poisson equation
- Extract iso-surface



Implementation: Surface Extraction

Given the Points:

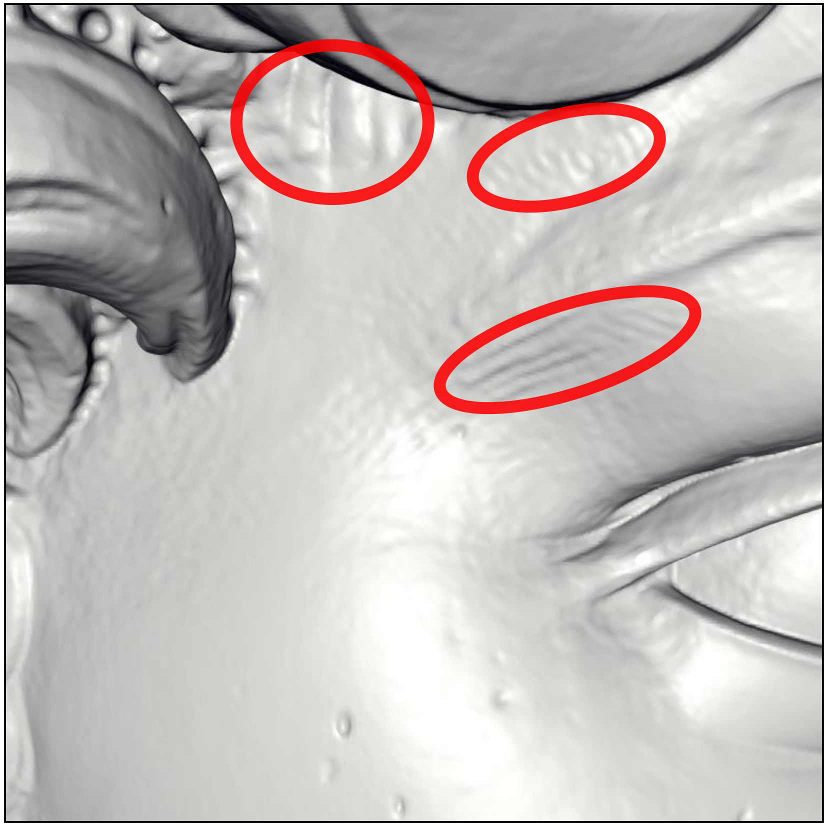
- Set octree
- Compute vector field
- Compute indicator function
- Extract iso-surface



Michelangelo's David

- 215 million data points from 1000 scans
- 22 million triangle reconstruction
- Maximum tree depth of 11
- Compute Time: 2.1 hours
- Peak Memory: 6600MB

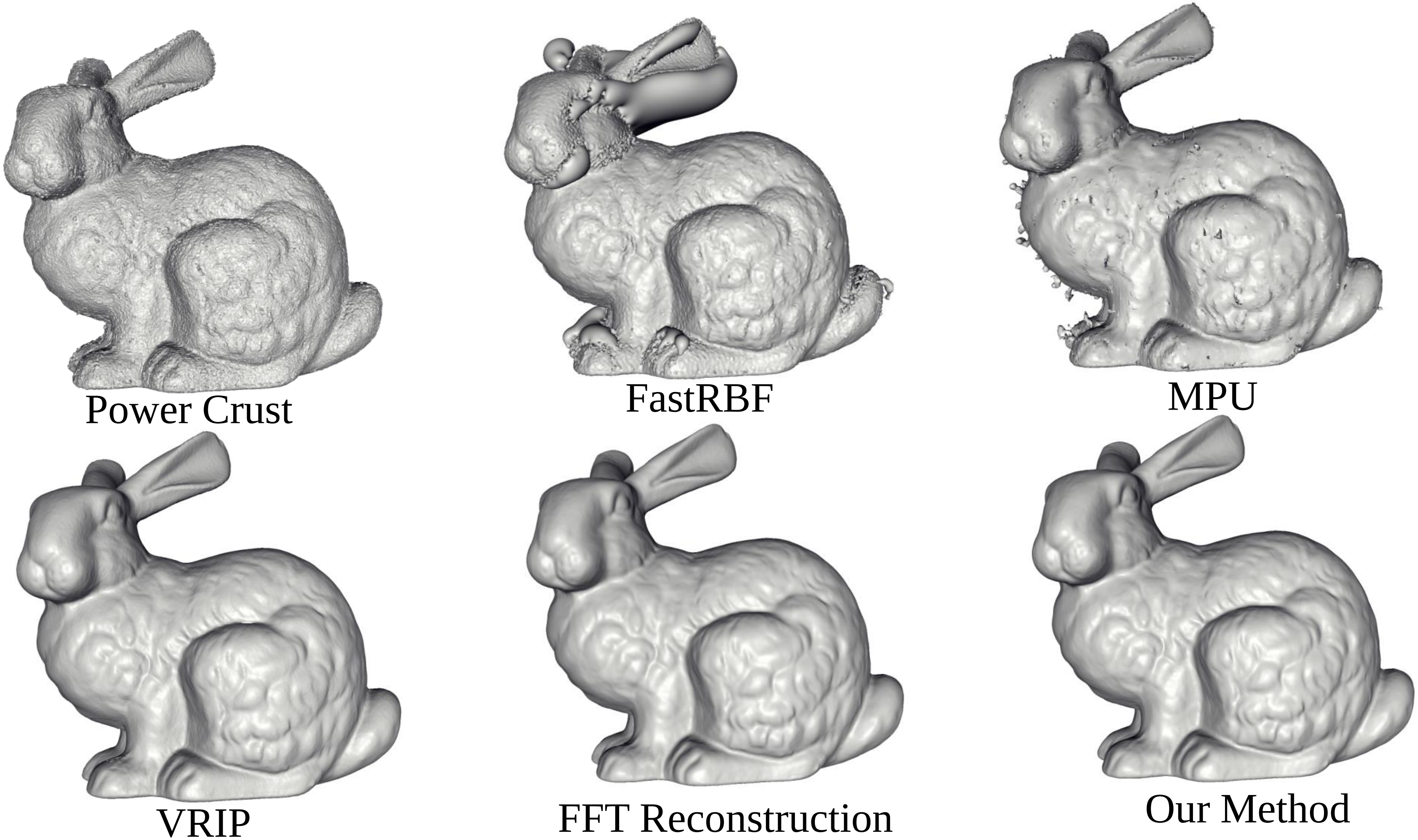
David – Chisel marks



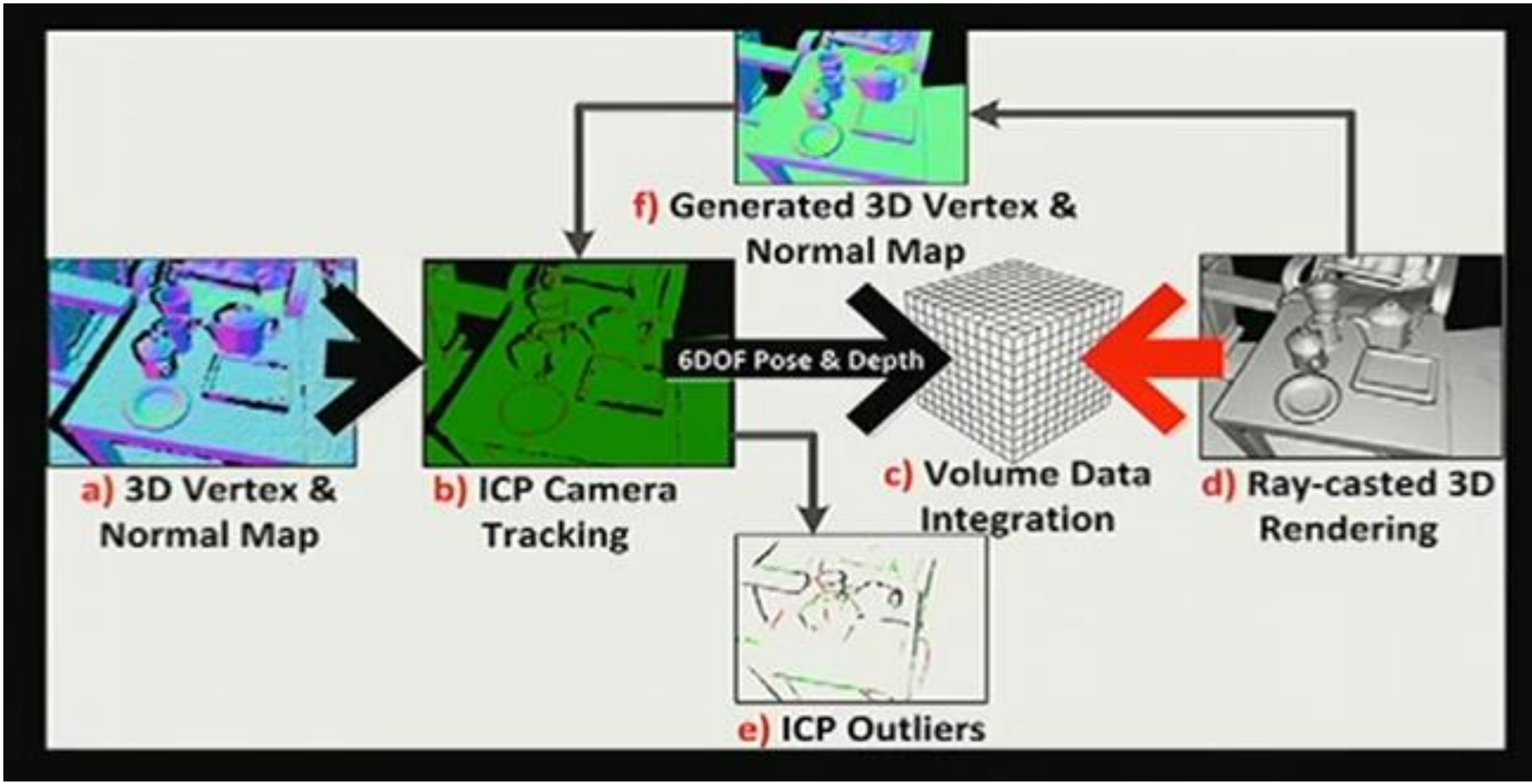
David – Drill Marks

David – Eye

Stanford Bunny



Kinect Fusion



Kinect Fusion

SIGGRAPH Talks 2011

KinectFusion:

Real-Time Dynamic 3D Surface Reconstruction and Interaction

Shahram Izadi 1, Richard Newcombe 2, David Kim 1,3, Otmar Hilliges 1, David Molyneaux 1,4, Pushmeet Kohli 1, Jamie Shotton 1, Steve Hodges 1, Dustin Freeman 5, Andrew Davison 2, Andrew Fitzgibbon 1

1 Microsoft Research Cambridge 2 Imperial College London
3 Newcastle University 4 Lancaster University
5 University of Toronto

Questions?