

IFT 6113

SURFACE RECONSTRUCTION

tiny.cc/ift6113

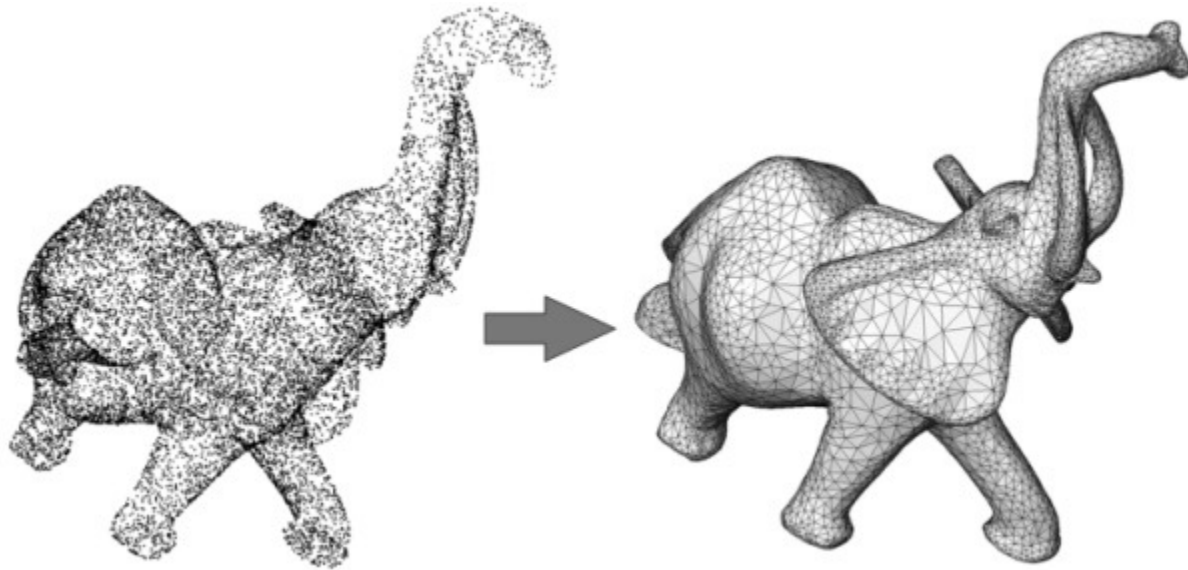
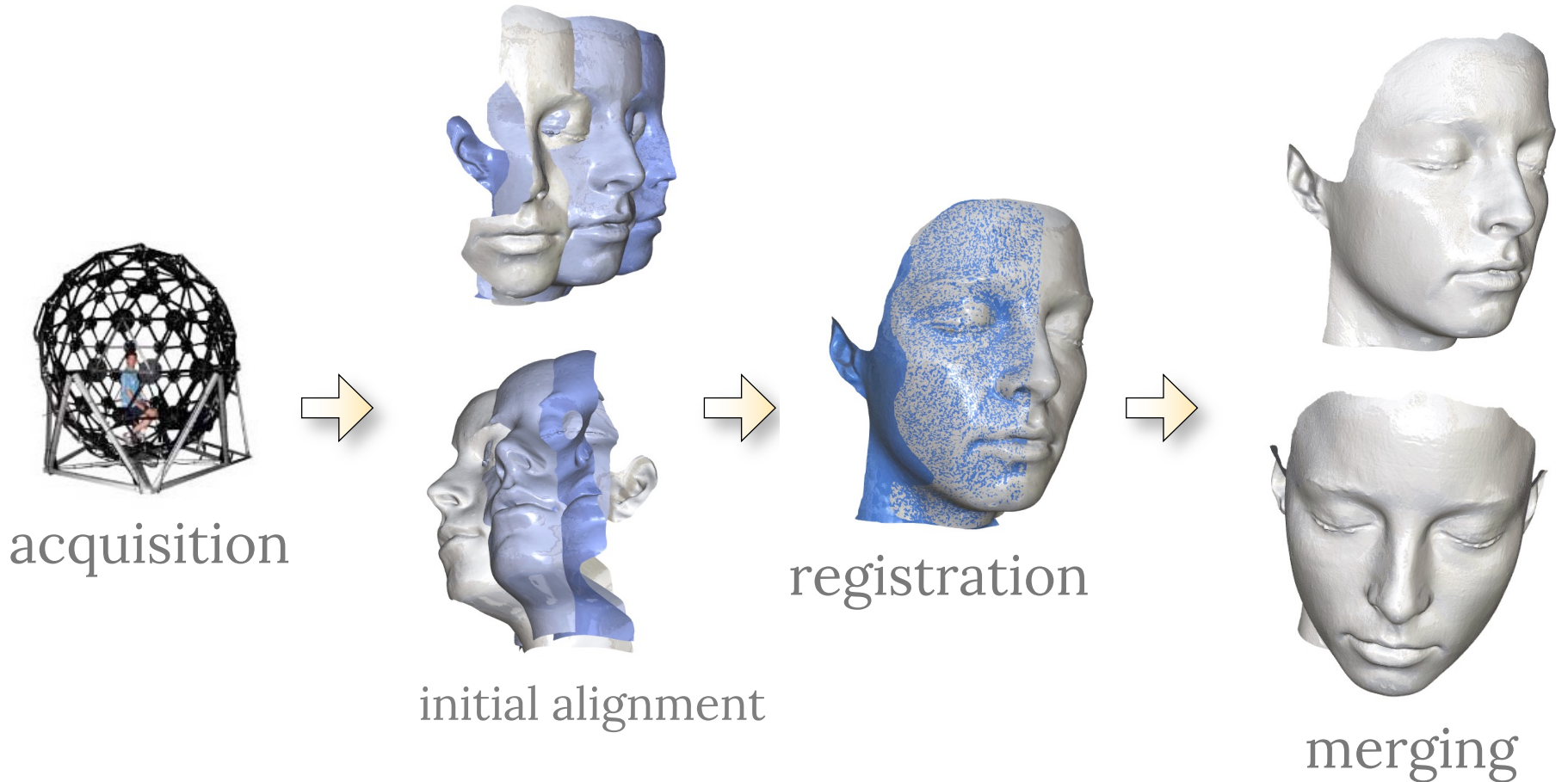


Image from https://doc.cgal.org/latest/Poisson_surface_reconstruction_3/index.html

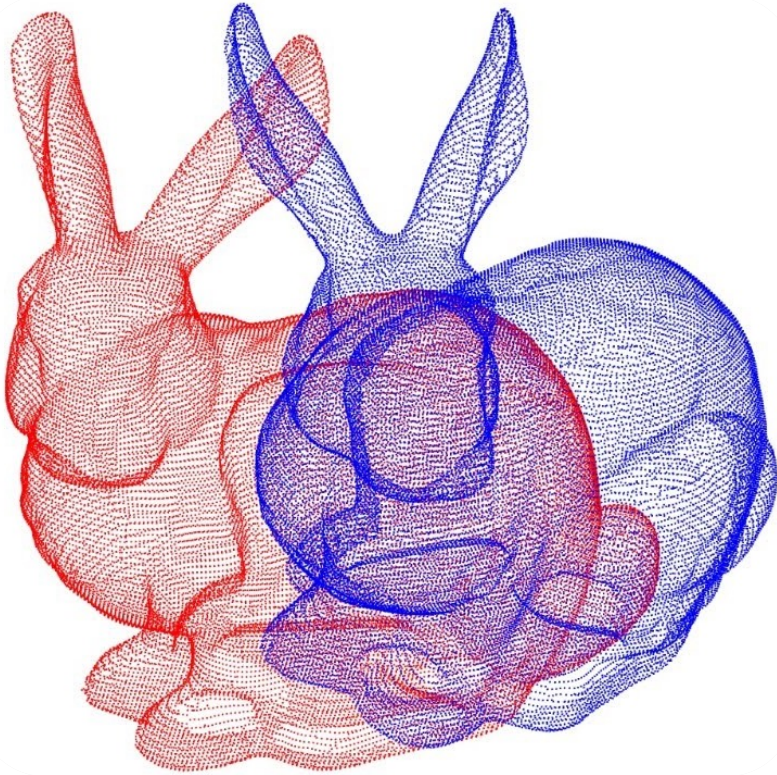
Mikhail Bessmeltsev

Some slides from Alla Sheffer,
Justin Solomon, and Hao Li

3D Reconstruction Pipeline

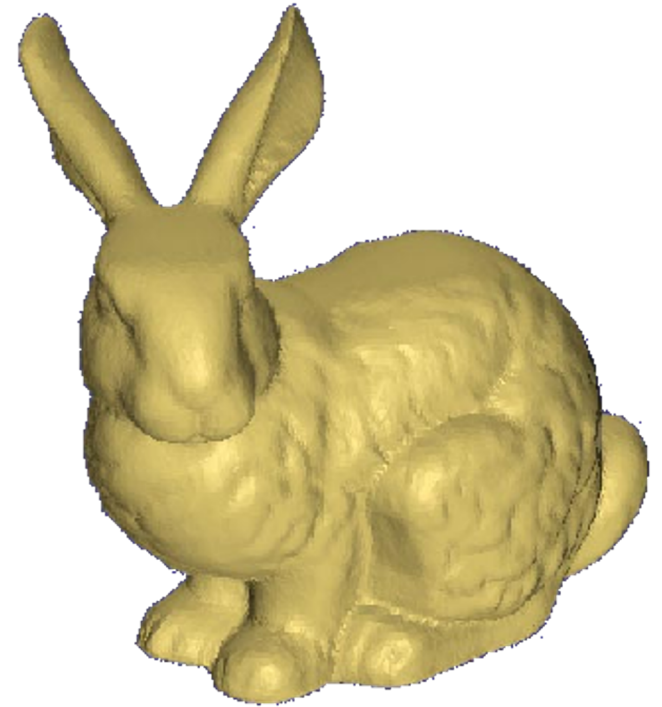


Two components



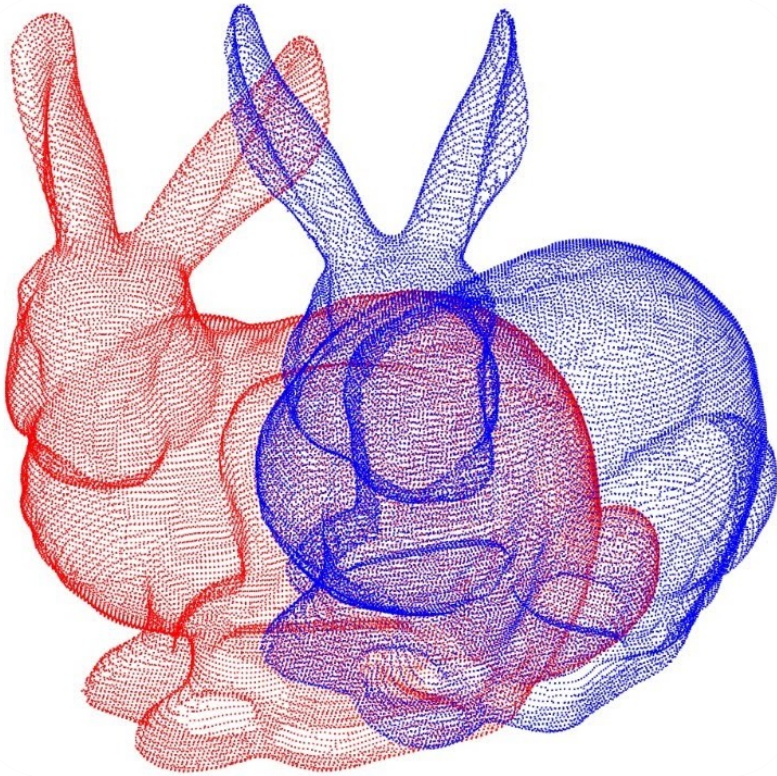
https://i.ytimg.com/vi/uzOCS_gdZuM/maxresdefault.jpg

Registration



Meshing

Two components



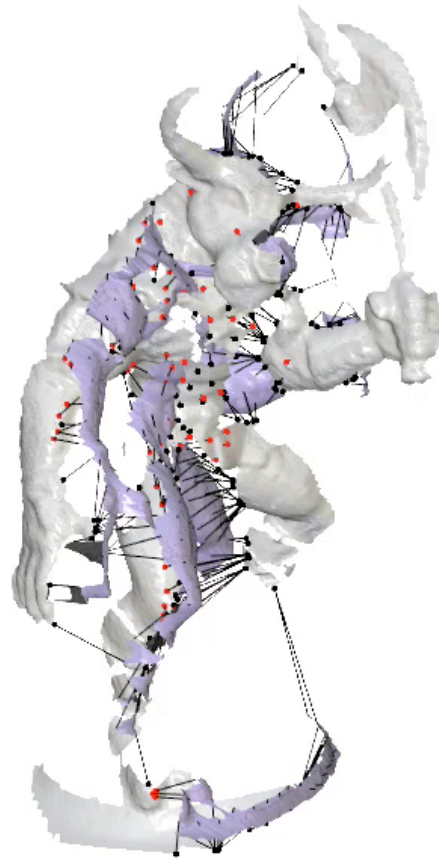
https://i.ytimg.com/vi/uzOCS_gdZuM/maxresdefault.jpg

Registration



Meshing

Registration Problem

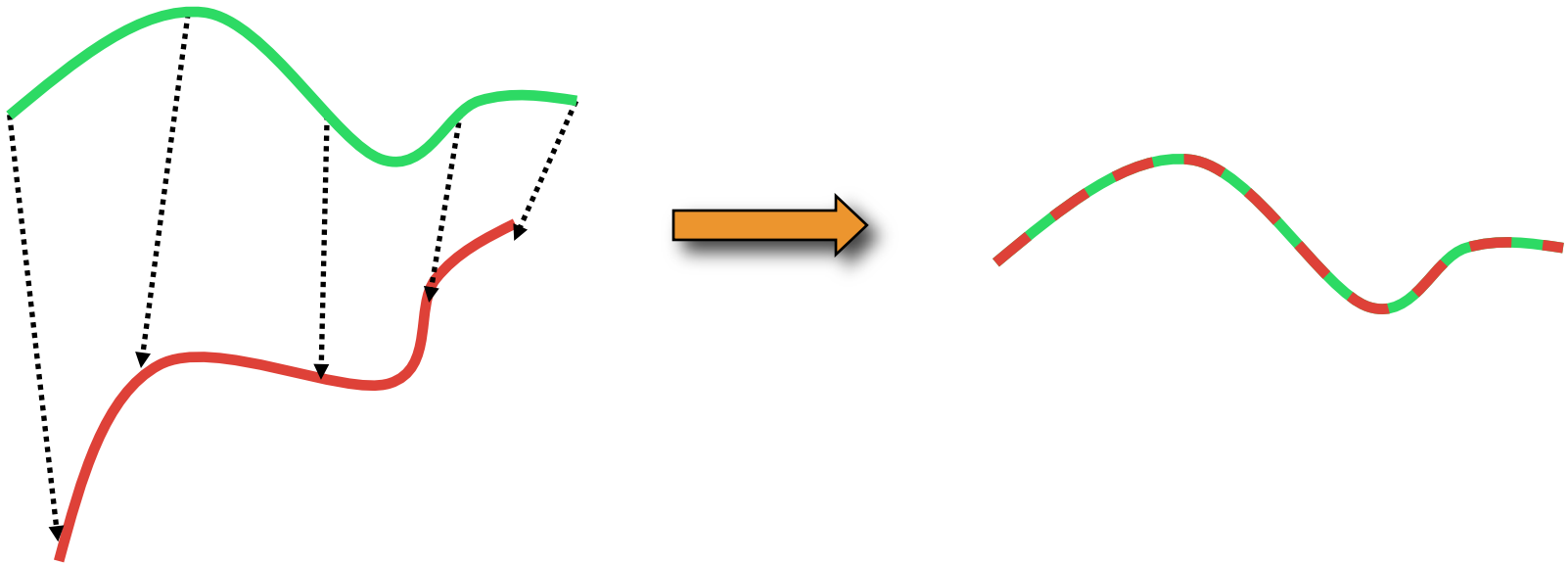


Align two overlapping objects

Rough Plan

- ***ICP algorithm***
A classic!
- ***ICP variants***

Starting Point



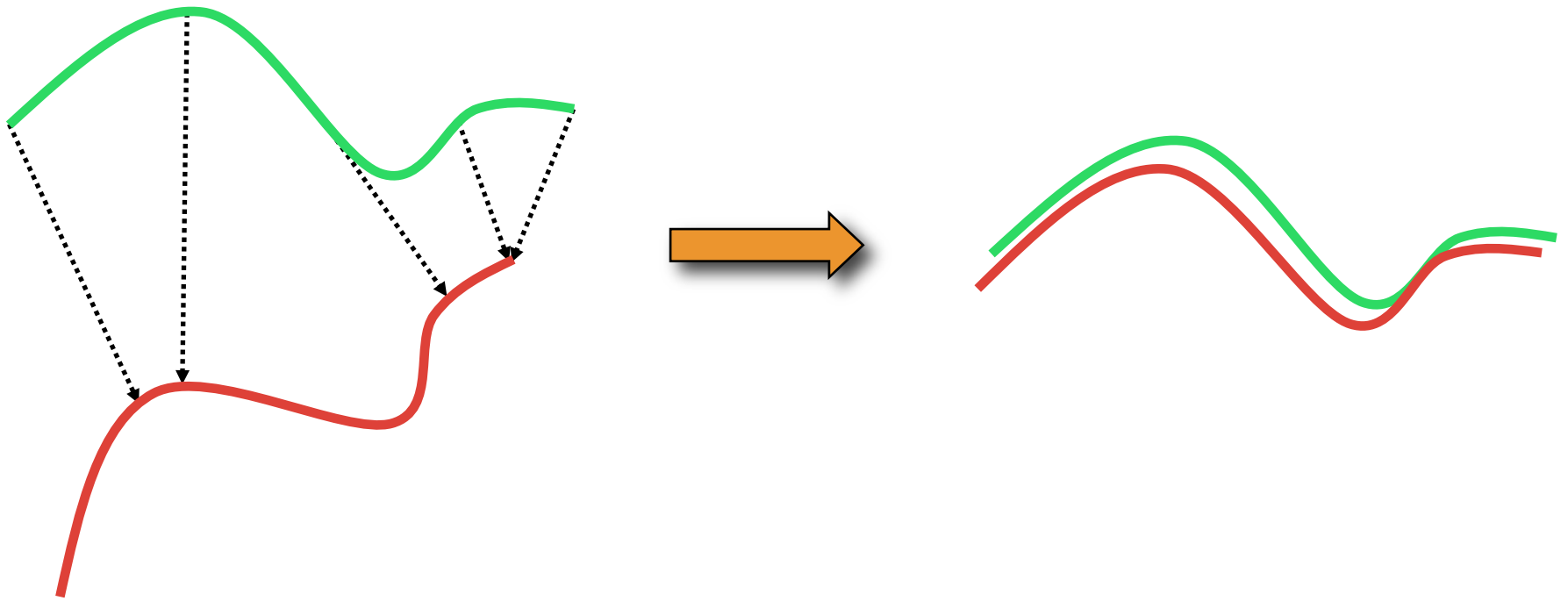
$$q_i = Rp_i + t$$

Can align given enough matches

***How many
correspondences
determine R and t ?***

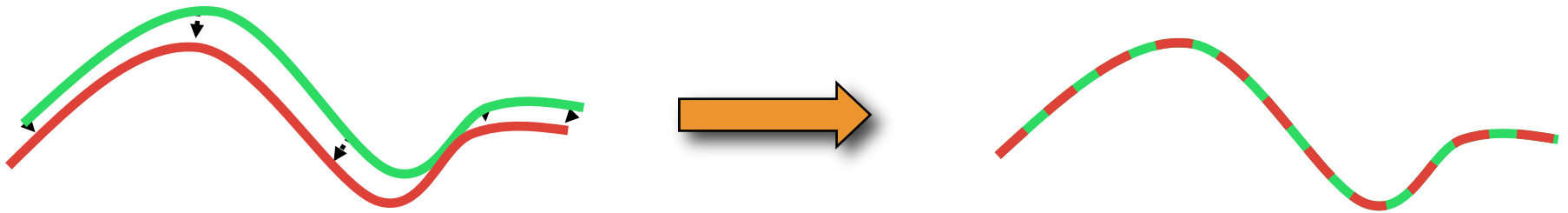
***How do you get
correspondences?***

Rough Approximation



Closest points correspond

Try a Second Time...



Iterative Closest Point (ICP)

- **Choose** e.g. 1000 random points
- **Match** each to closest point on other scan
- **Reject** pairs with distance $> k$ times median
- **Minimize**

$$E[R, t] := \sum_i \|Rp_i + t - q_i\|^2$$

- **Iterate**

“A method for registration of 3-D shapes.”
Besl and McKay, PAMI 1992.

On the Board

$$\min_{t \in \mathbb{R}^3, R^\top R = I} \sum_i \|Rp_i + t - q_i\|^2$$

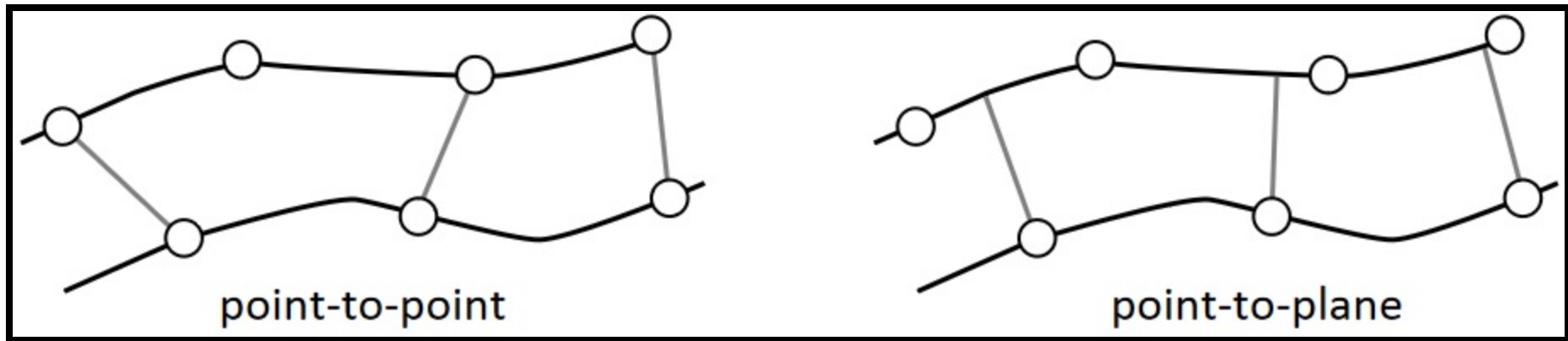
Closed-form formulas!

Many (!) Variants of ICP

- **Source points** from one or both meshes
- **Matching** to points in the other mesh
 - **Weighting** correspondences
 - **Rejecting outlier** point pairs
 - **Alternative error metrics**

Point-to-Plane Error Metric

Flat parts can slide along each other



$$E[R, t] := \sum_i ((Rp_i + t - q_i)^\top n_i)^2$$

$$\approx \sum_i [(p_i - q_i)^\top n_i + r^\top (p_i \times n_i) + t^\top n_i]^2 \text{ after linearizing}$$

where $r := (r_x, r_y, r_z)$

Least-squares!

“Object modelling by registration of multiple range images”

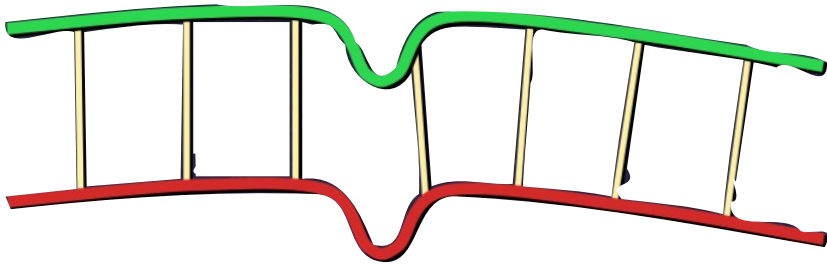
Chen and Medioni, Image and Vision Computing 10.3 (1992); image courtesy N. Mitra

Closest Compatible Point

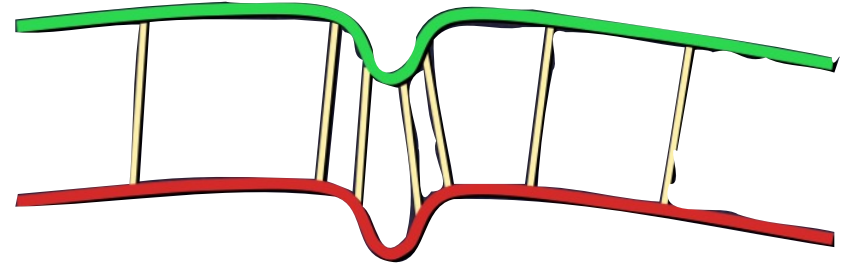
Can improve matching effectiveness by restricting match to **compatible points**

- Compatibility of colors [Godin et al. 94]
- Compatibility of normals [Pulli 99]
- Other possibilities:
curvatures, higher-order derivatives, and other local features

Choose Points to Improve Stability



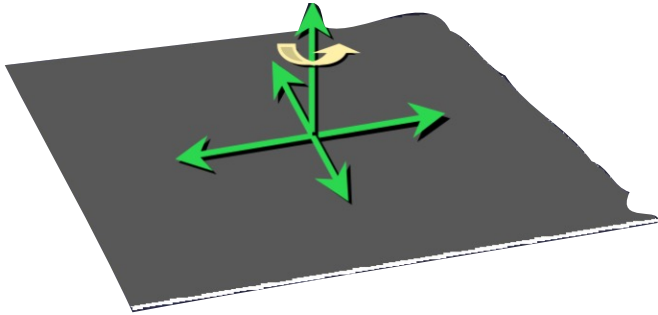
Uniform Sampling



Stable Sampling

Sample discriminative points

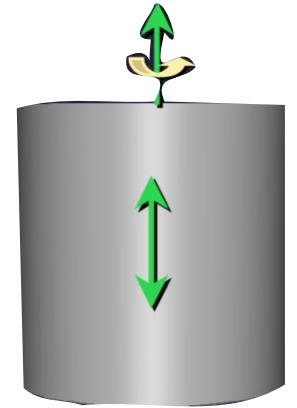
Local Covariance



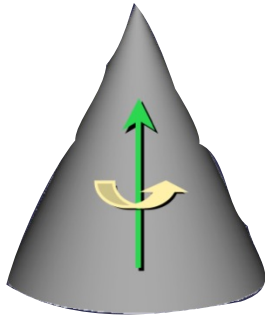
3 small eigenvalues
2 translation
1 rotation



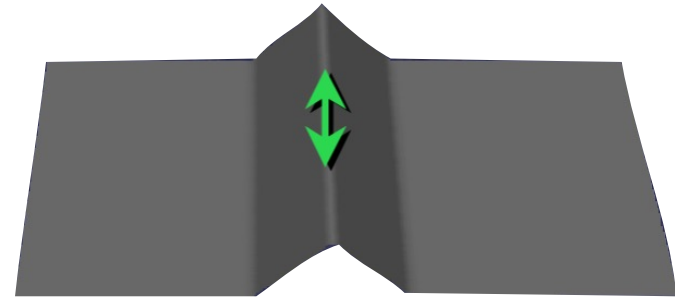
3 small eigenvalues
3 rotation



2 small eigenvalues
1 translation
1 rotation



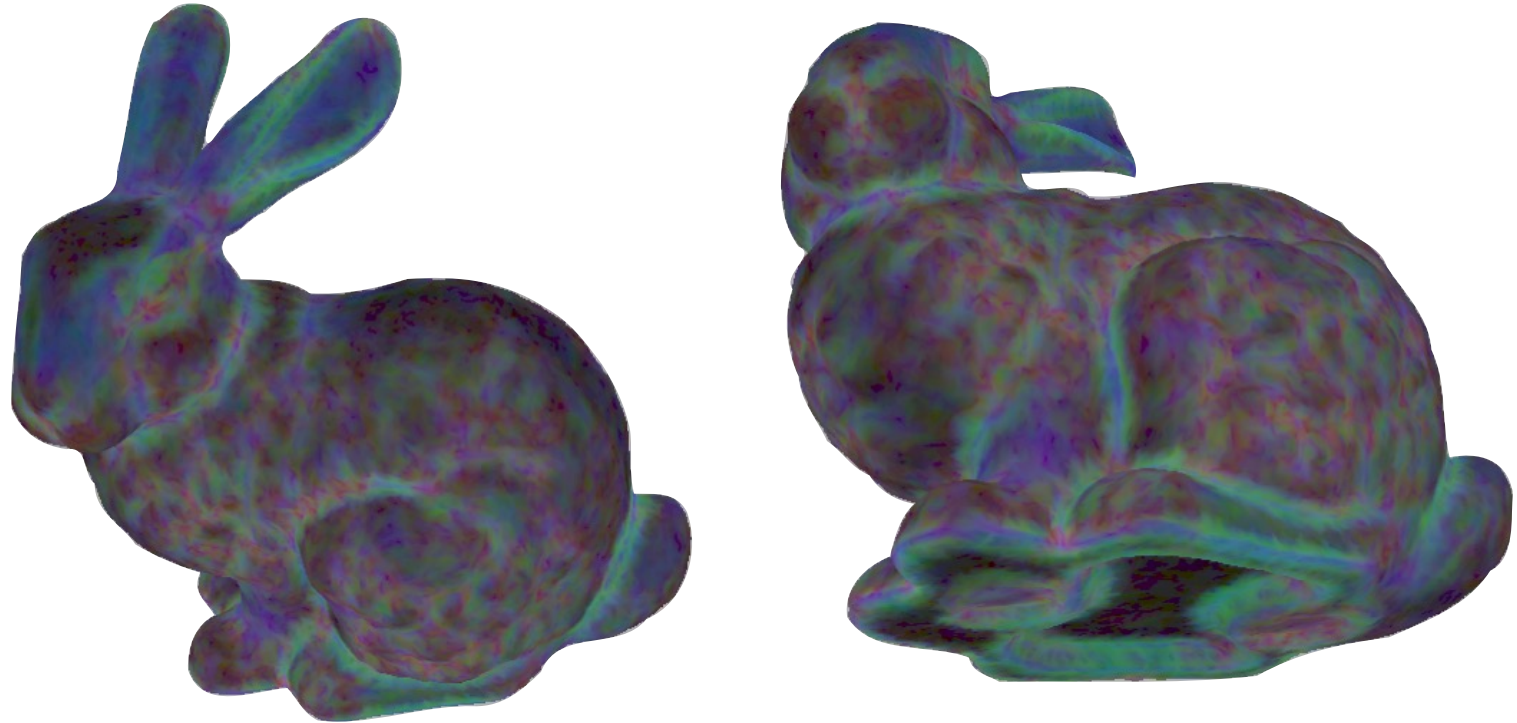
1 small eigenvalue
1 rotation



1 small eigenvalue
1 translation

[Gelfand et al. 2004]

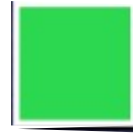
Stability Analysis



Key:



3 DOFs stable



5 DOFs stable

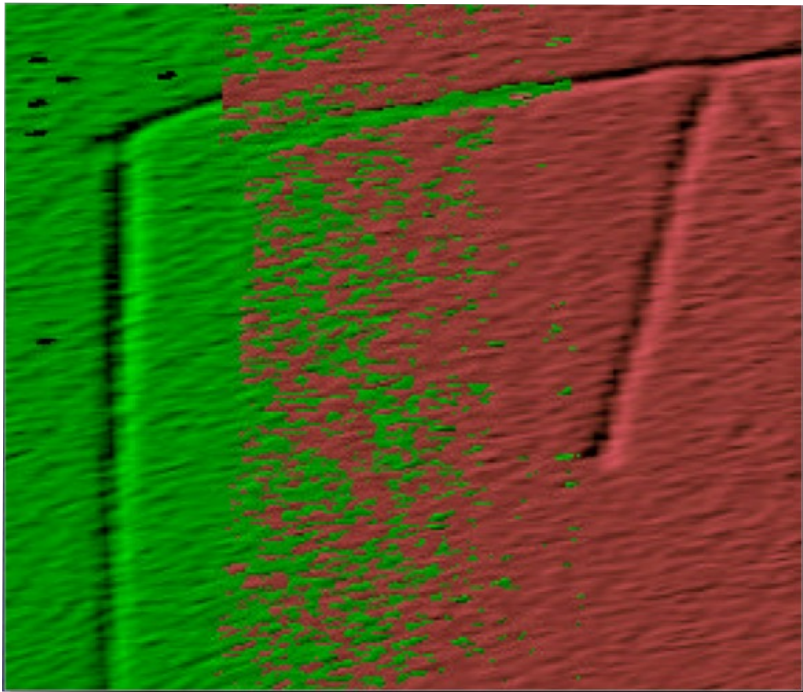


4 DOFs stable

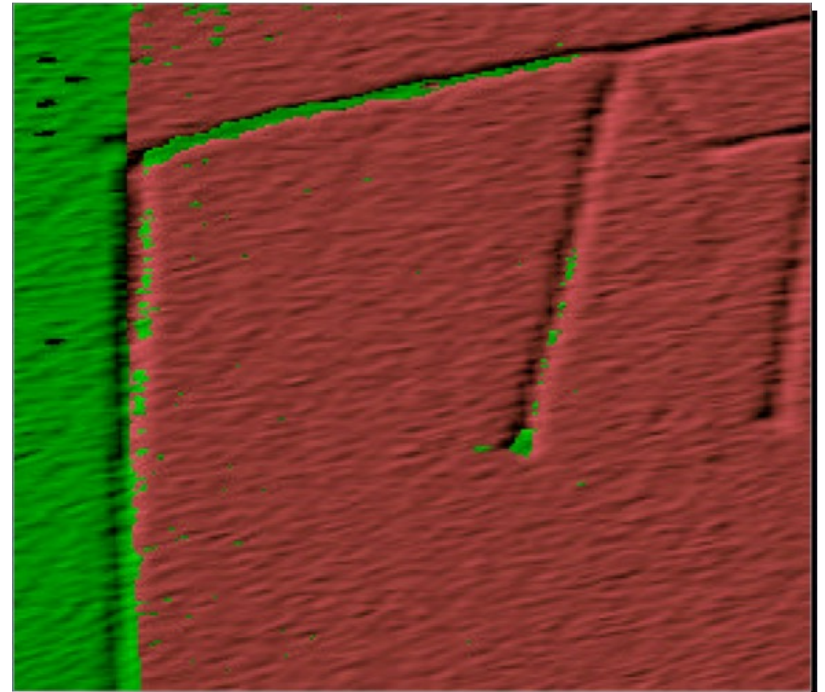


6 DOFs stable

Alternative: Uniform Normals

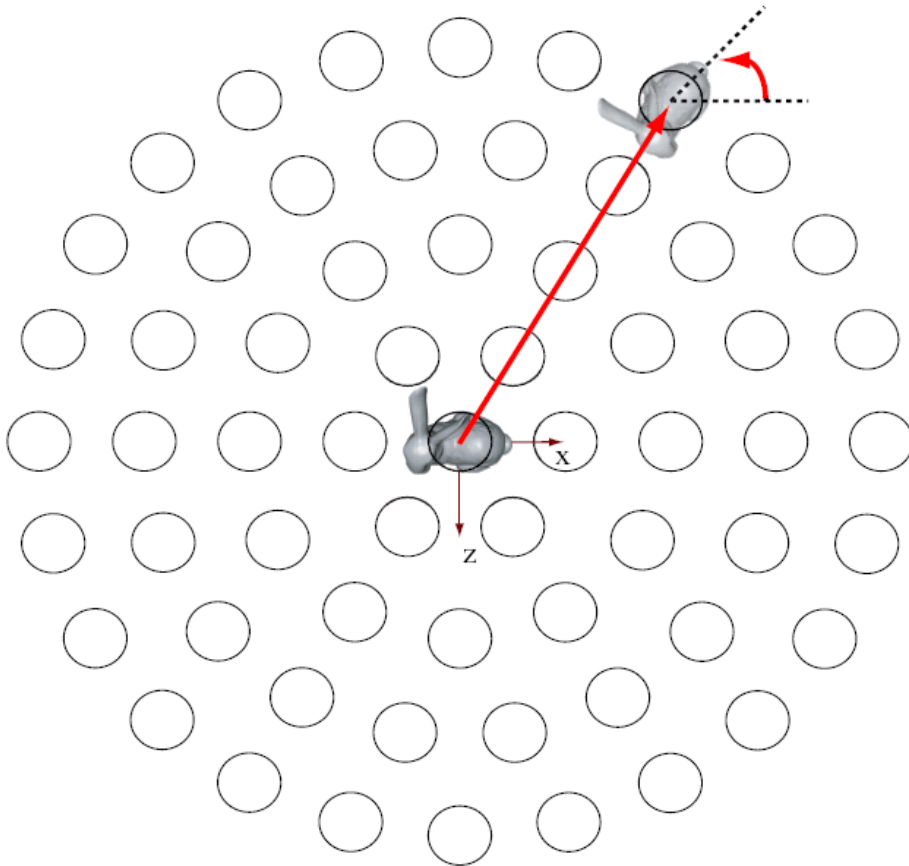


Random Sampling

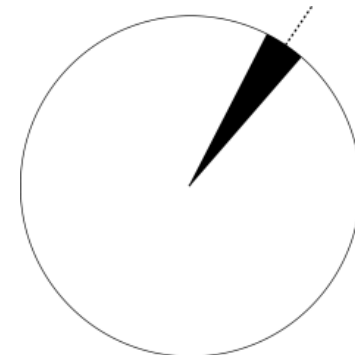


Normal-space Sampling

Convergence Funnel Visualization

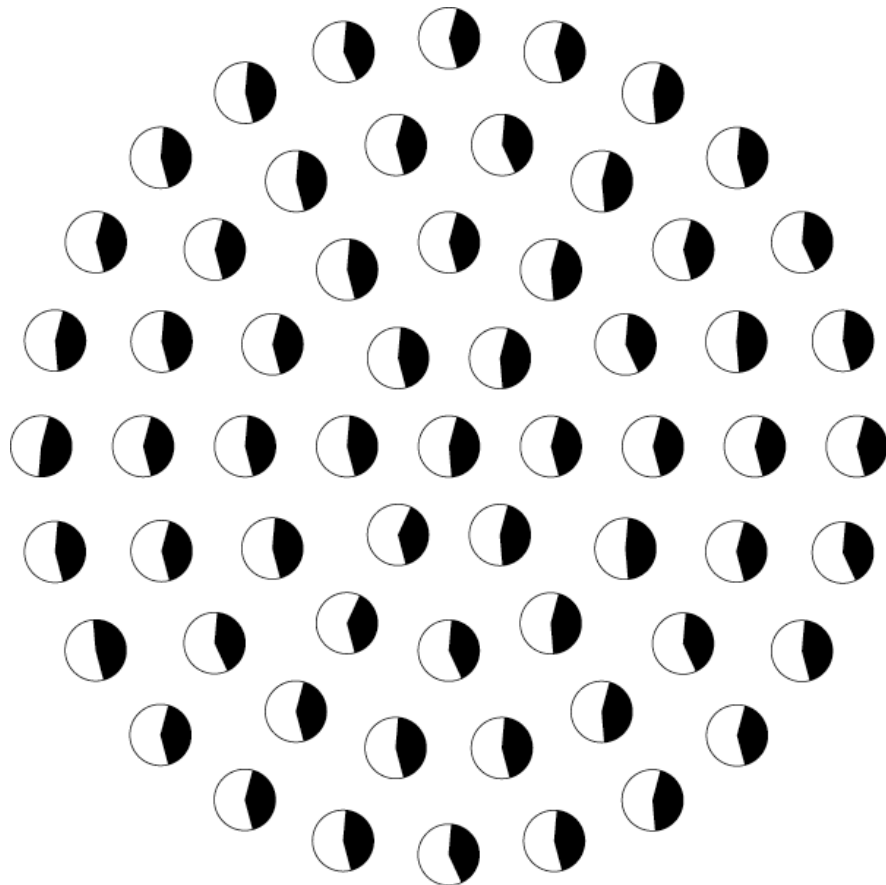


Translation in xz plane
Rotation about y

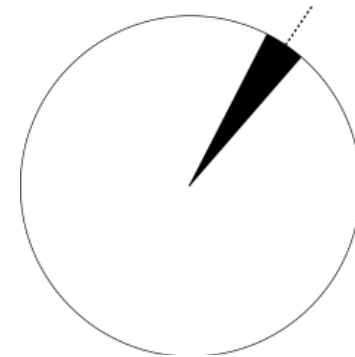


- Converges
- Does not converge

Distance Field Method

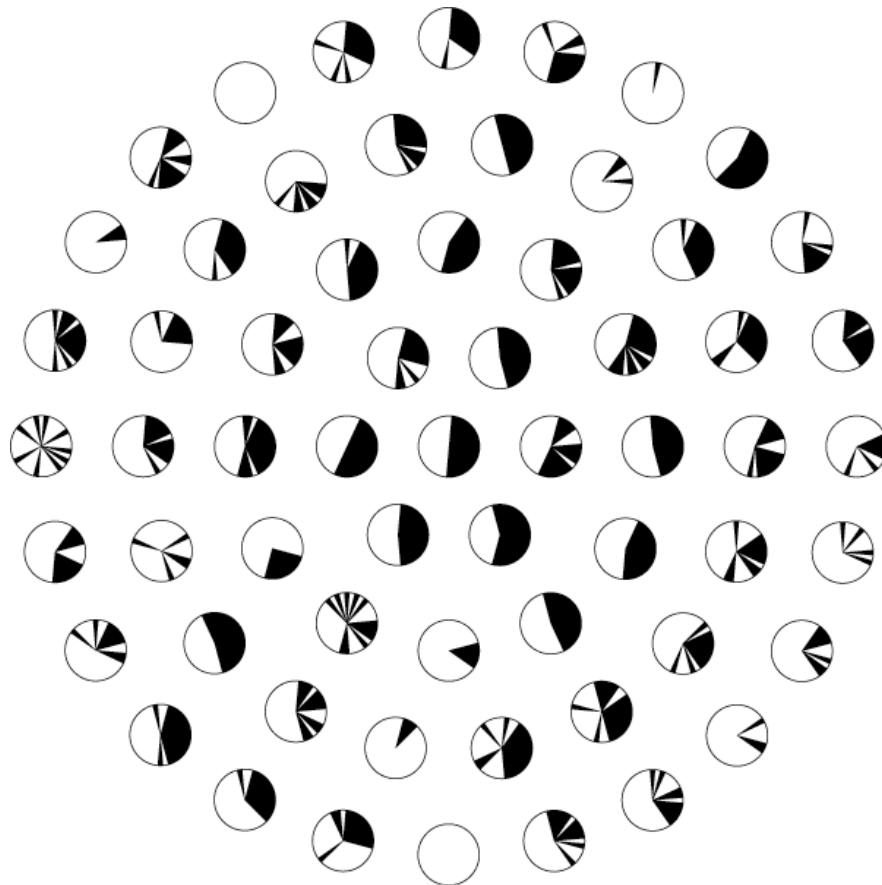


Translation in xz plane
Rotation about y

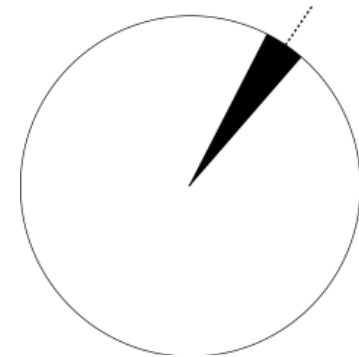


- Converges
- Does not converge

Point-to-Plane

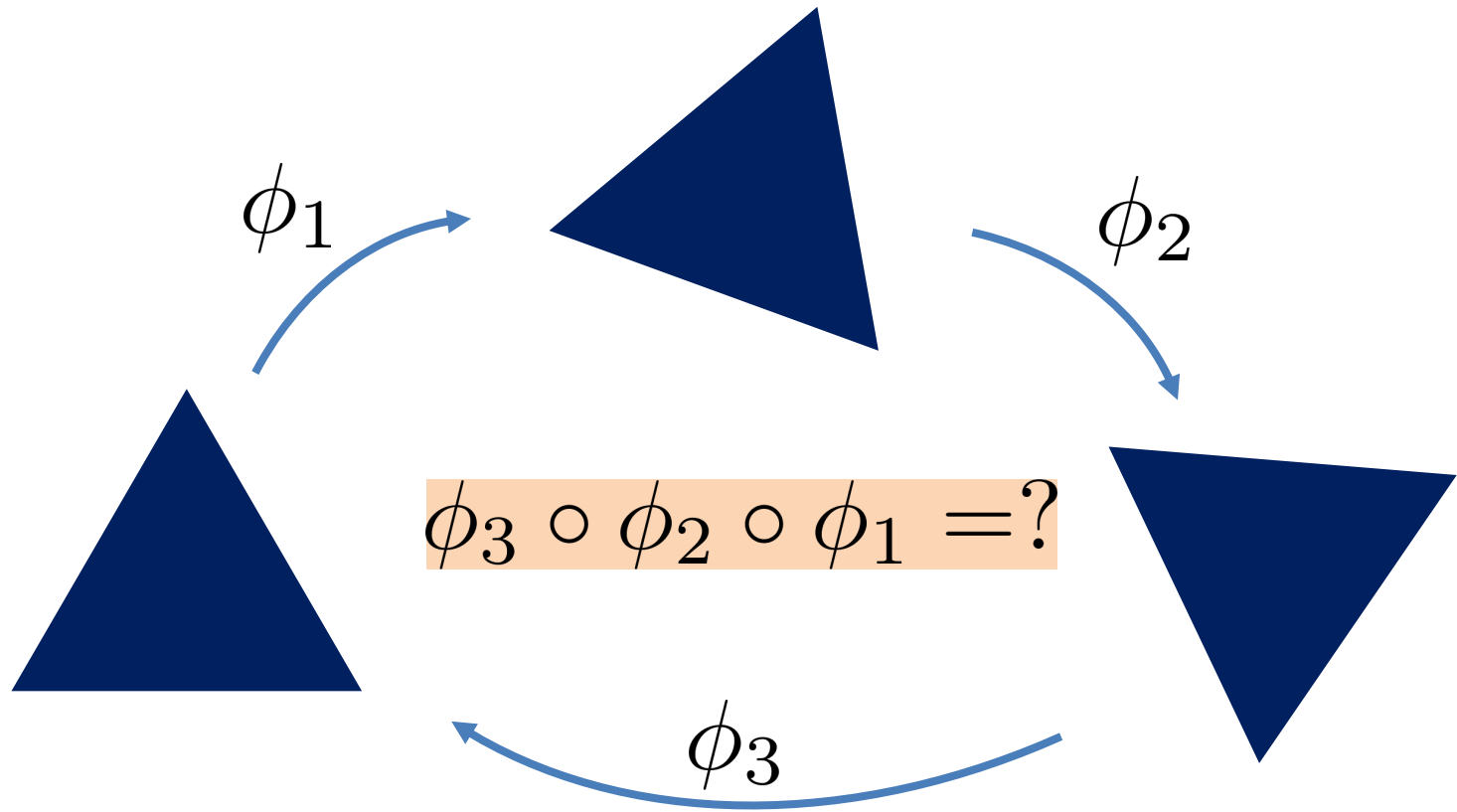


Translation in xz plane
Rotation about y



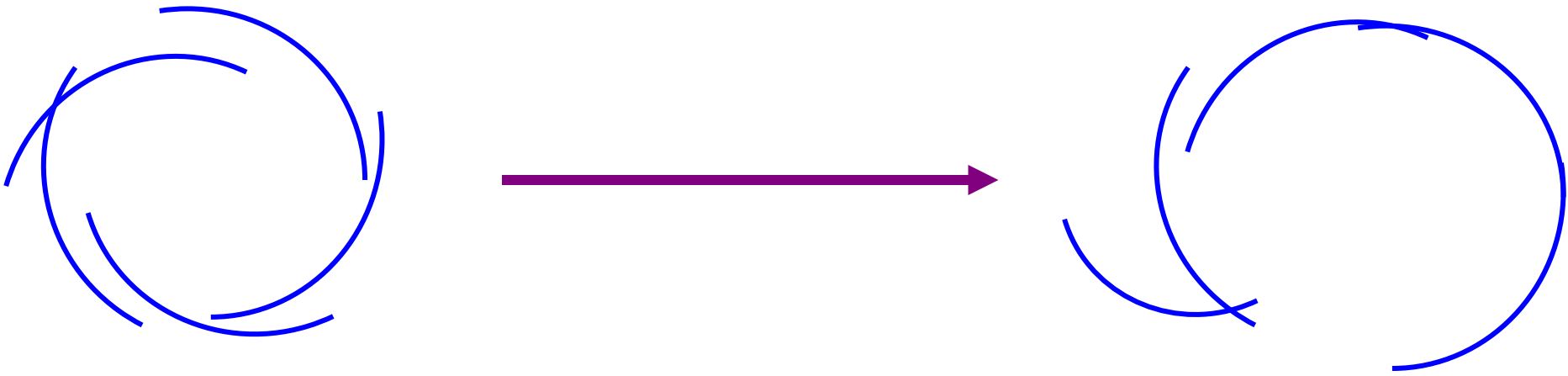
- Converges
- Does not converge

Issue: ICP Three Times



Usually have ≥ 2 scans

Improve Sequential Alignment?

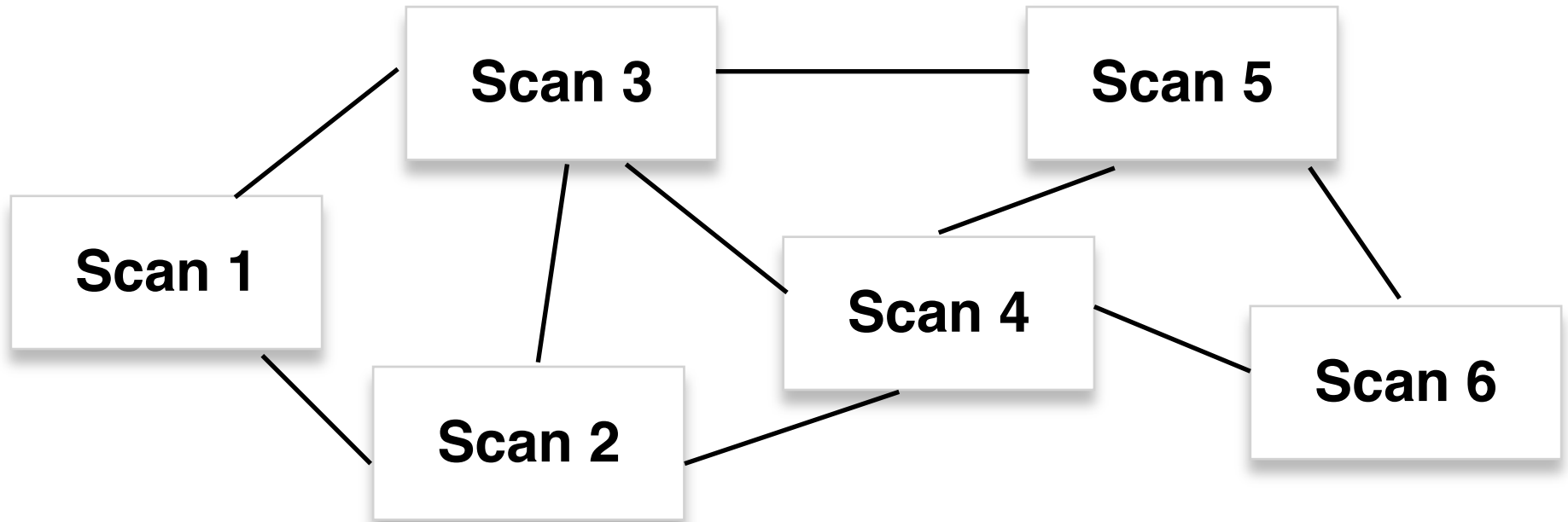


Prevent “drift”

Simple Methods

- **Align everything to *anchor* scan**
Which to choose? Dependence on anchor?
- **Align to *union* of previous scans**
Order dependence? Speed?
- ***Simultaneously* align everything using ICP**
Local optima? Computational expense?

Graph Approach



Align similar scans, then assemble

Lu and Milios

- ***Pairwise phase***

Compute pairwise ICP on graph

- ***Global alignment***

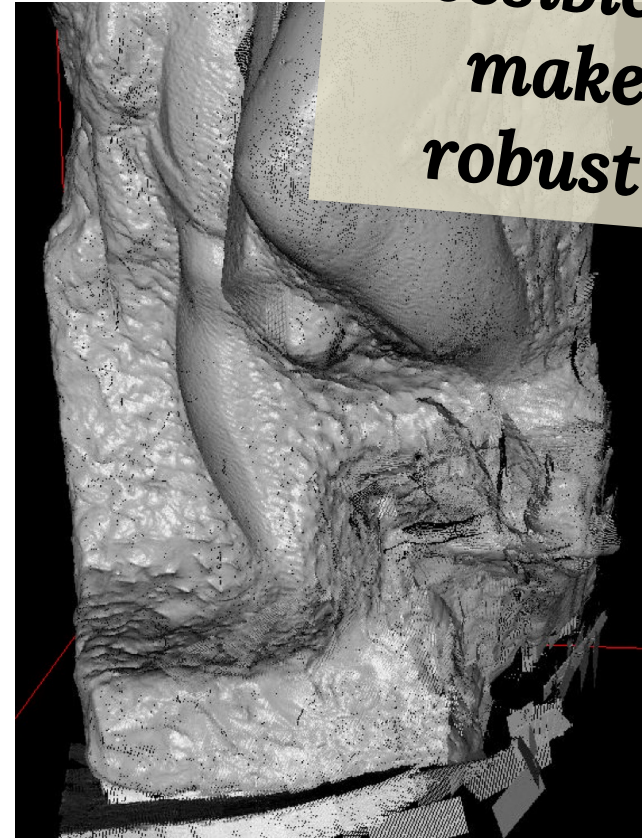
Least-squares rotation/translation

**Linearize for
global alignment**

Failed ICP in Global Registration



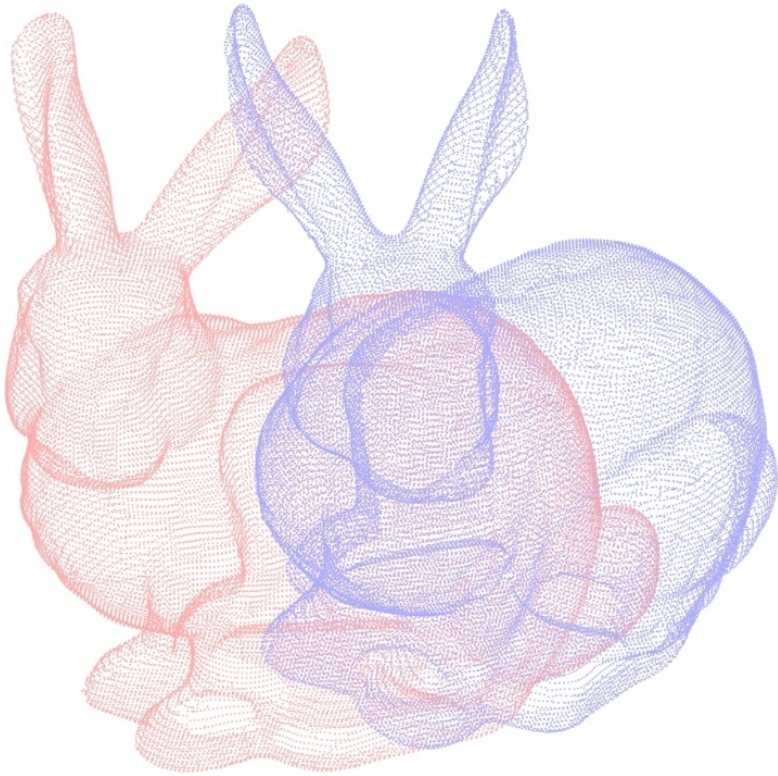
Correct global registration



Global registration including bad ICP

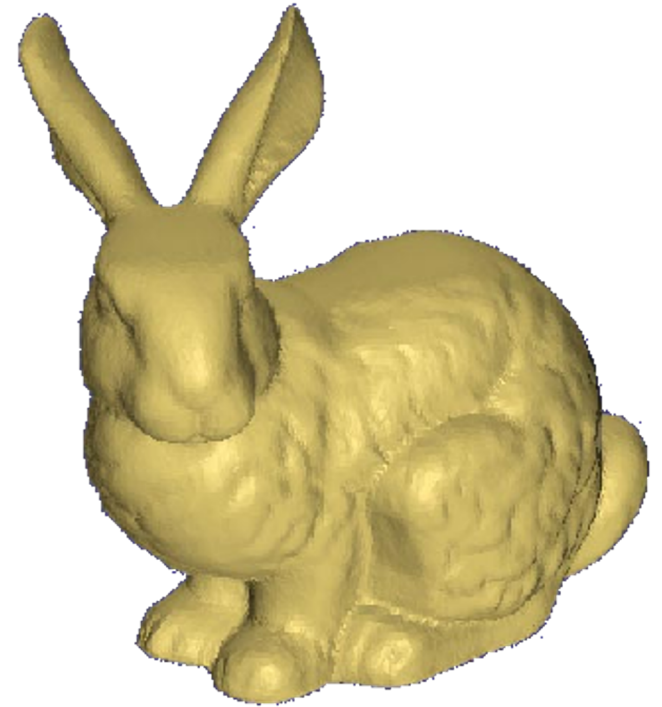
Possible to make robust?

Two components



https://i.ytimg.com/vi/uzOCS_gdZuM/maxresdefault.jpg

Registration



Meshing

Triangulating Point Clouds

Connect neighboring points into triangles



Point cloud data



Reconstructed surface



Triangulating Point Clouds

Connect neighboring points into triangles



Point cloud data



Reconstructed surface



Who are the neighbors?
⇔ What's the connectivity/topology

Methods

- Explicit, *or reconstruction circa 1998*
 - Zippering
 - Delaunay/Voronoi-based
- Implicit
 - Signed distance function
 - Poisson
- Data-driven

Methods

- Explicit, *or reconstruction circa 1998*
 - **Zippering**
 - Delaunay/Voronoi-based
- Implicit
 - Signed distance function
 - Poisson
- Data-driven

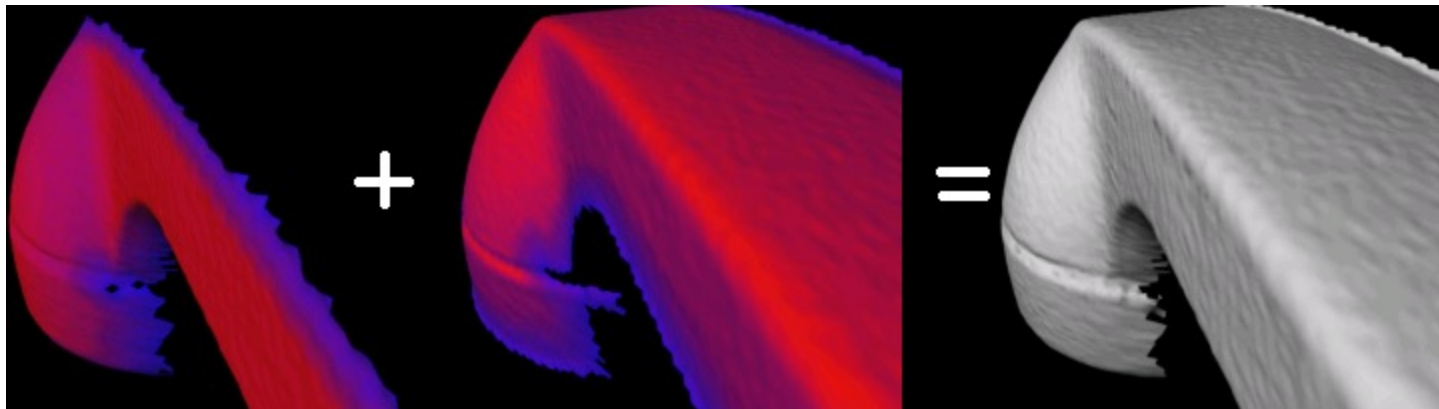
Basic Reconstruction: Zippering

Single scan \rightarrow mesh

- regular lattice of points in X and Y with changing depth (Z) = height map

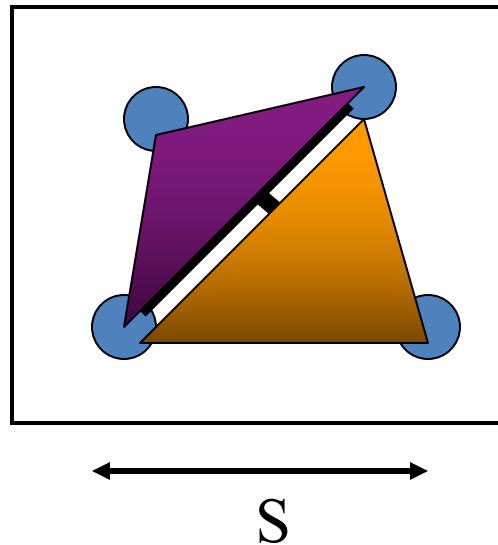
Register

Merge meshes



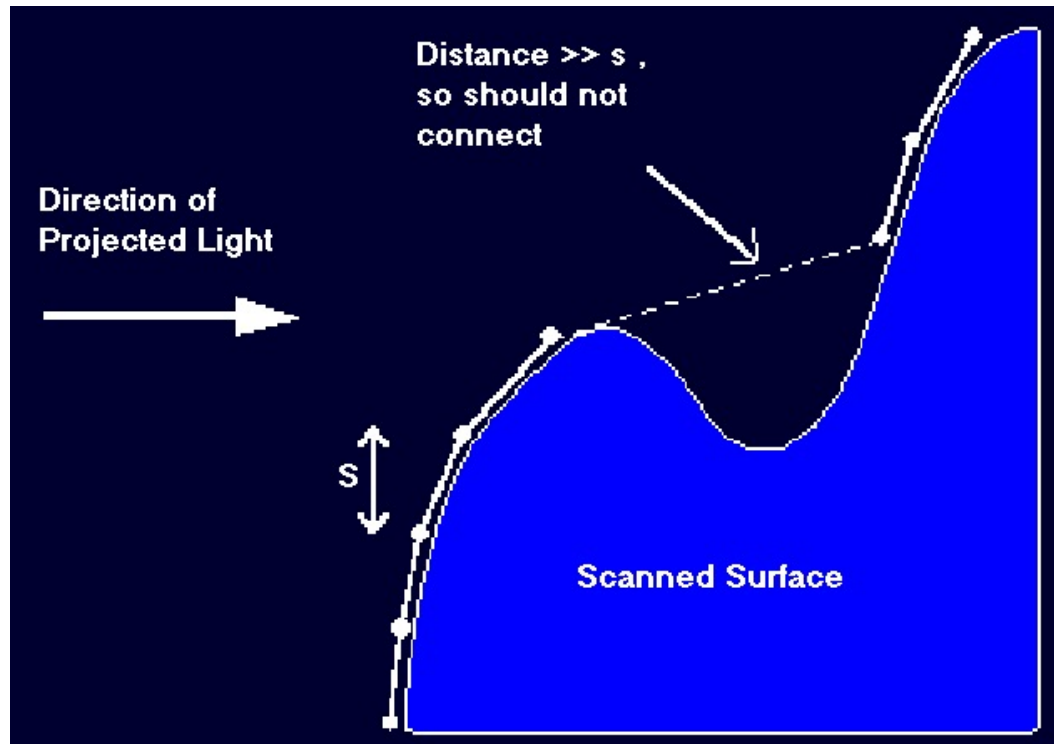
One scan \rightarrow mesh

- Find quadruples of lattice points
- Form triangles
 - Find shortest diagonal
 - Form two triangles (test depth)



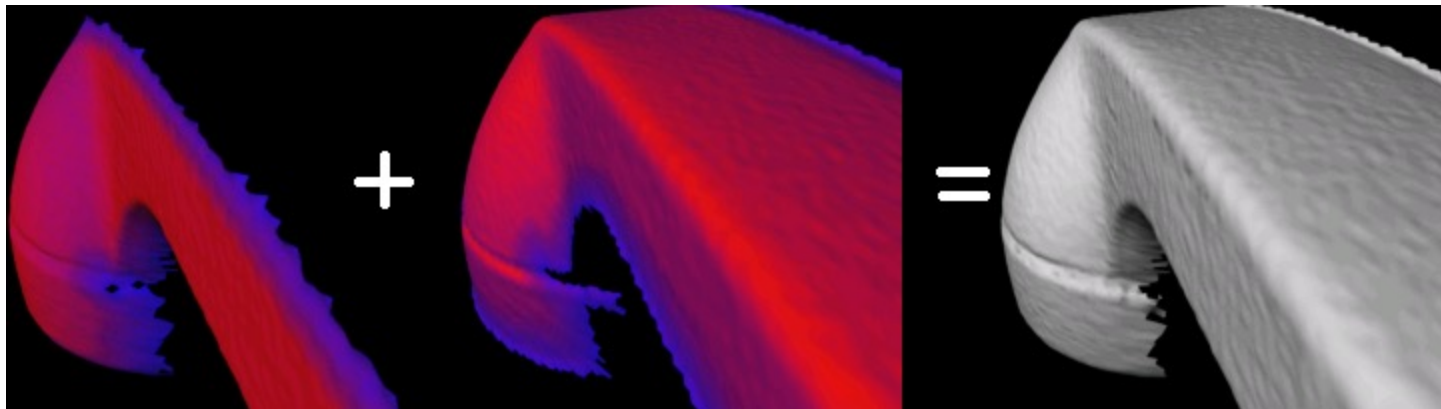
One scan \rightarrow mesh

Avoid connecting depth discontinuities



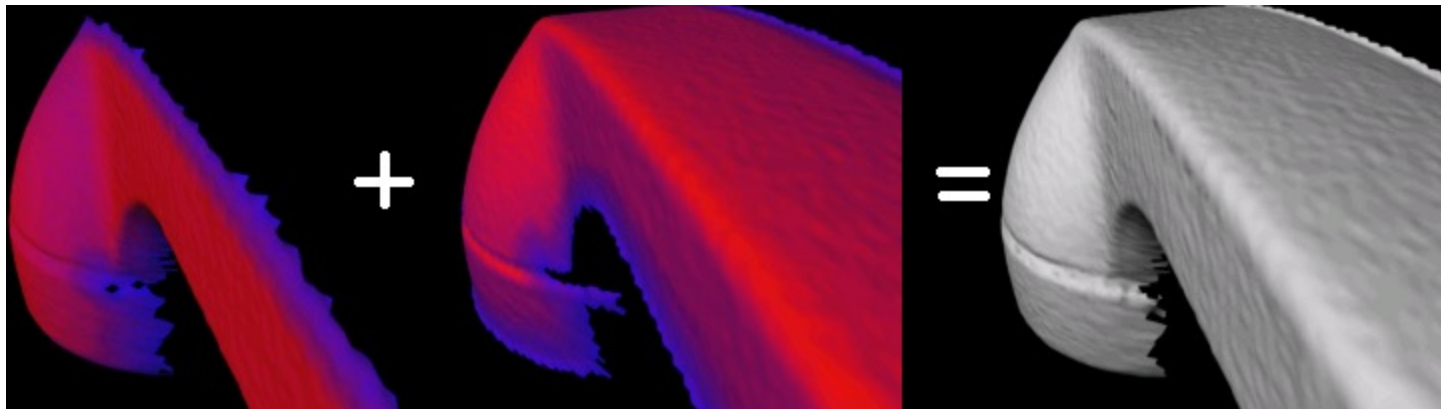
Basic Reconstruction: Zippering

- ✓ Single scan \rightarrow mesh
- Register
- Merge meshes



Basic Reconstruction: Zippering

- ✓ Single scan \rightarrow mesh
- ✓ Register
- Merge meshes

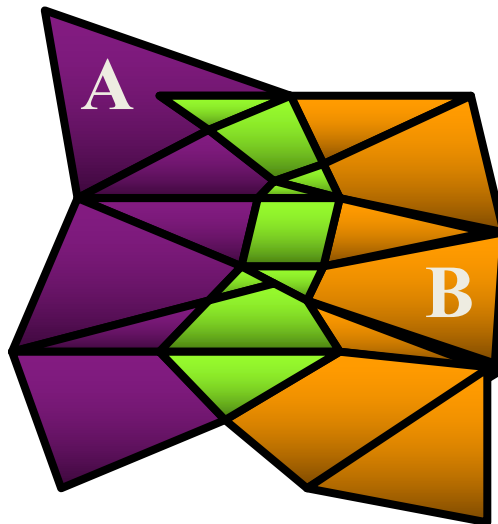


Merging

2 overlapping meshes

Zippering

- Remove overlapping portion of the mesh
 - Use for ***consensus geometry***
- Clip one mesh against another
- Remove triangles introduced during clipping



Post-processing



Zippering
results

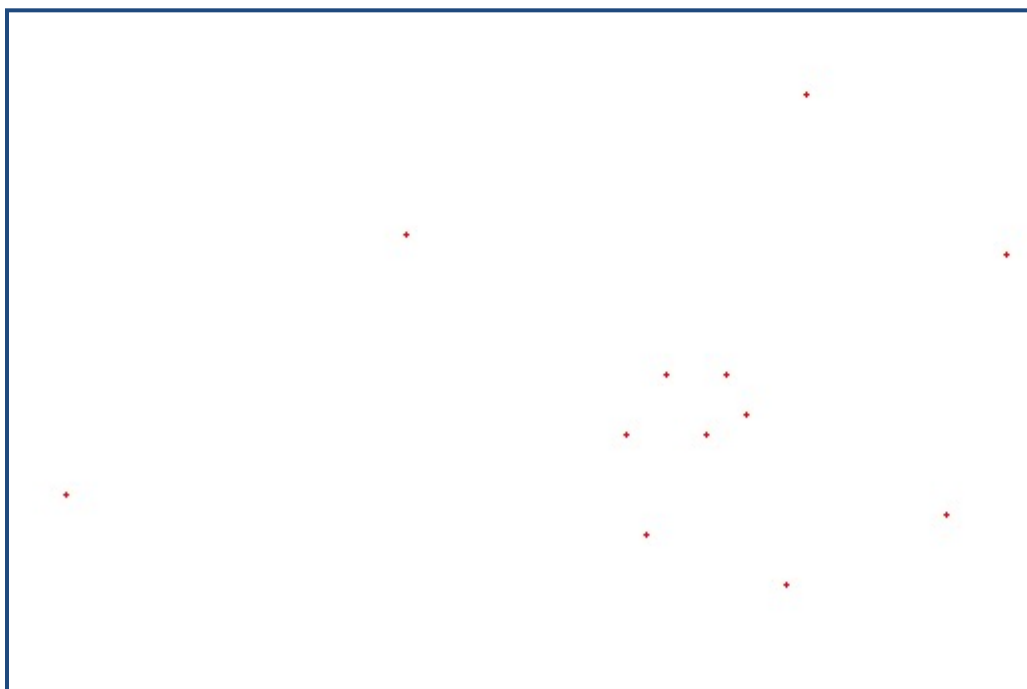
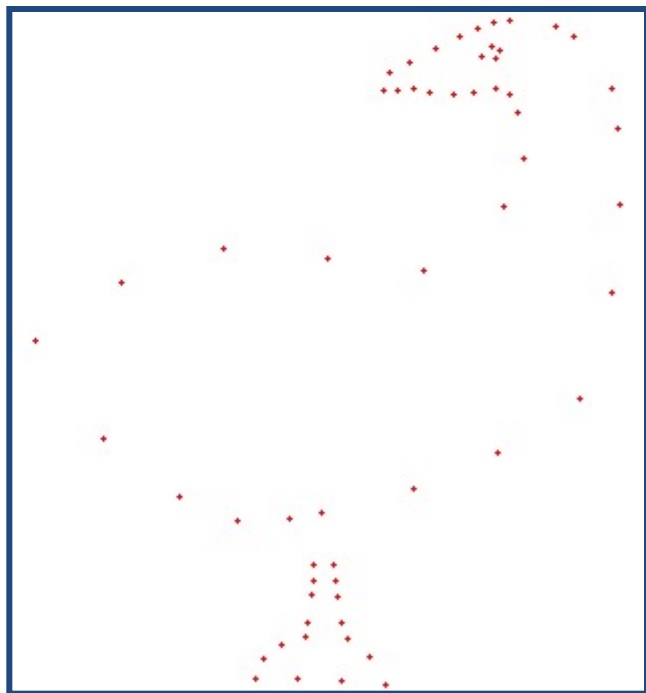
'Consensus
geometry'

*Move vertices to their average positions
over all scans*

Methods

- Explicit, *or reconstruction circa 1998*
 - Zippering
 - ***Delaunay/Voronoi-based***
- Implicit
 - Signed distance function
 - Poisson
- Data-driven

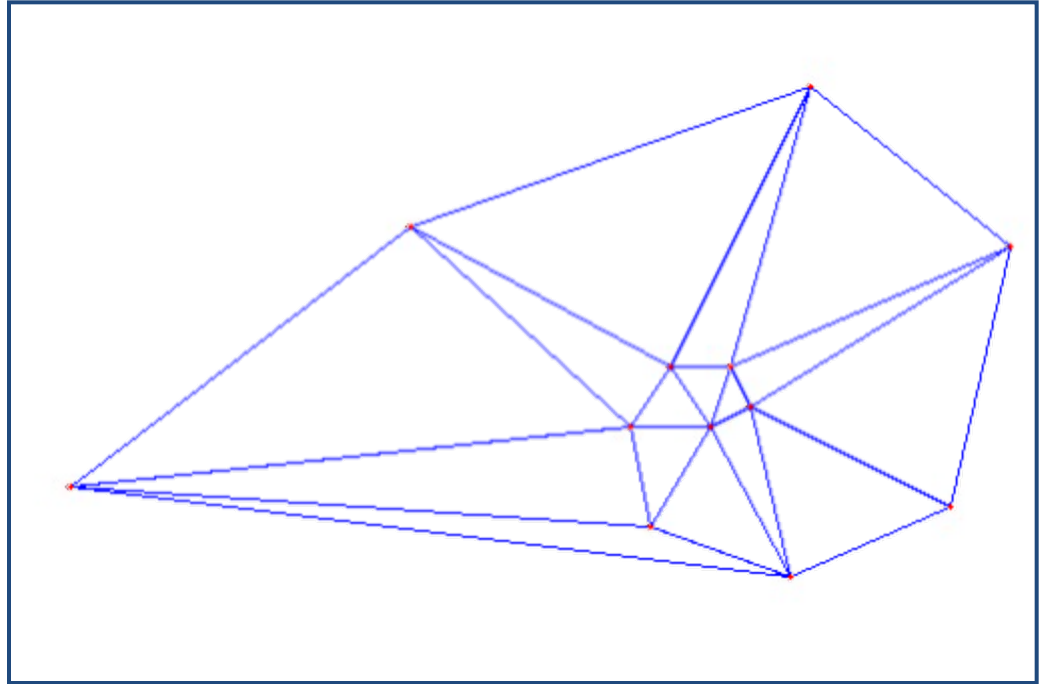
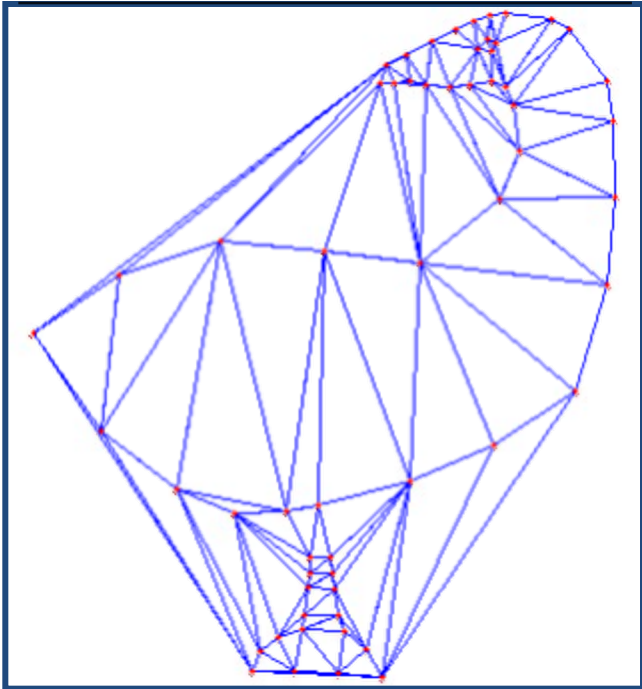
2D: connect the dots



Connectivity?

Edges should be far from other points

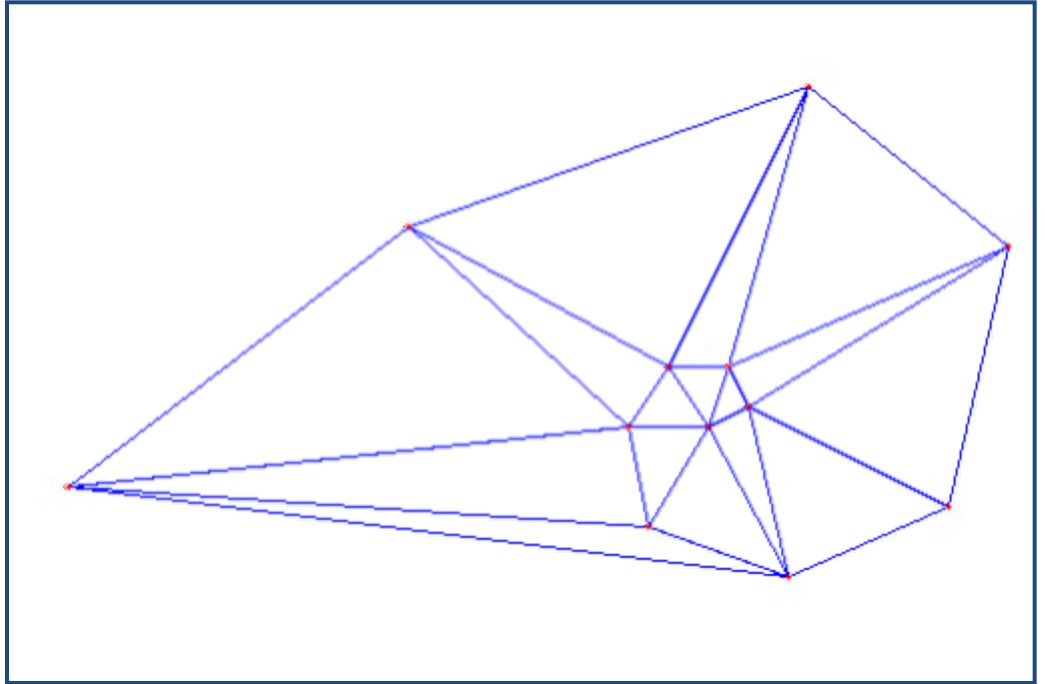
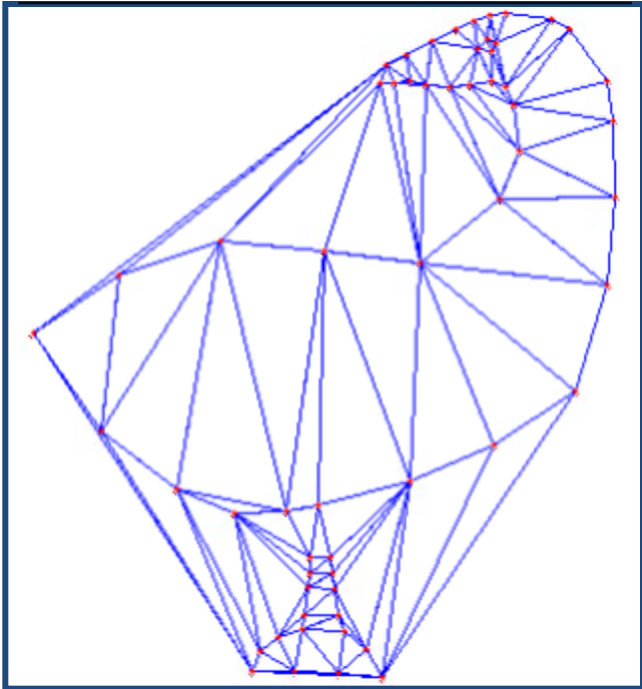
2D: connect the dots



Delaunay Triangulation

Edge e is Delaunay \Leftrightarrow some circumcircle of e contains no other sample points

2D: connect the dots

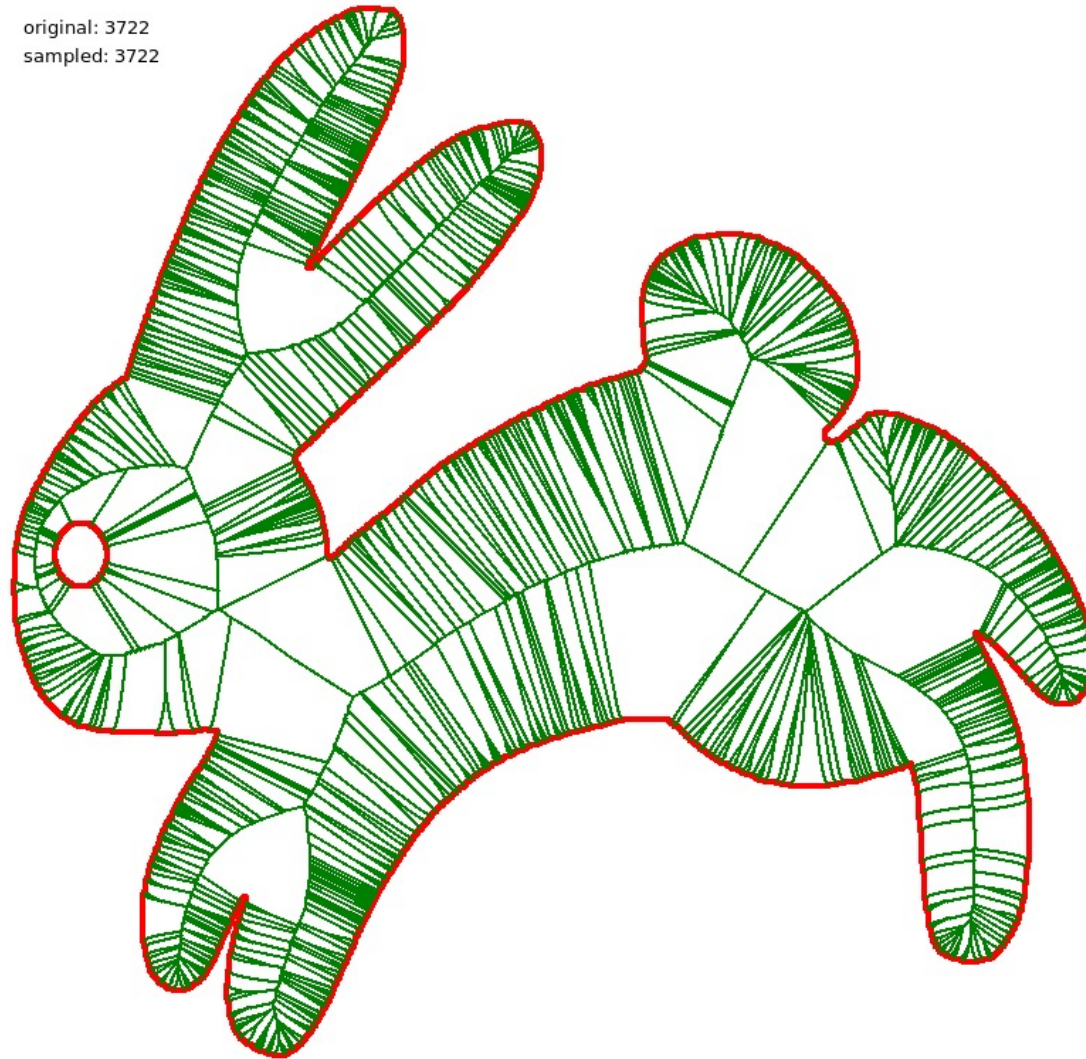


Which edges to pick?

Recall:

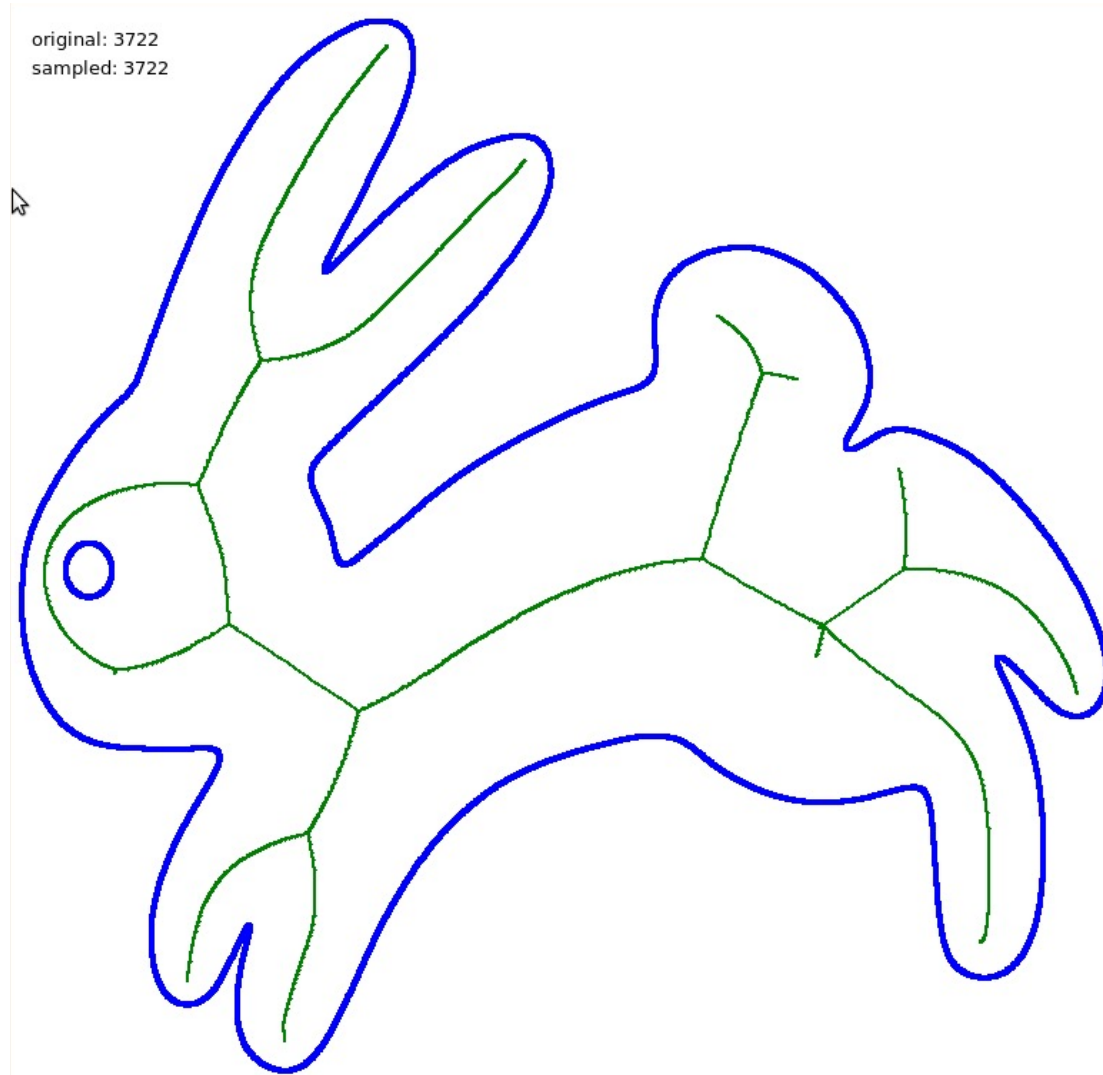
Medial axis vs Voronoi diagram

original: 3722
sampled: 3722

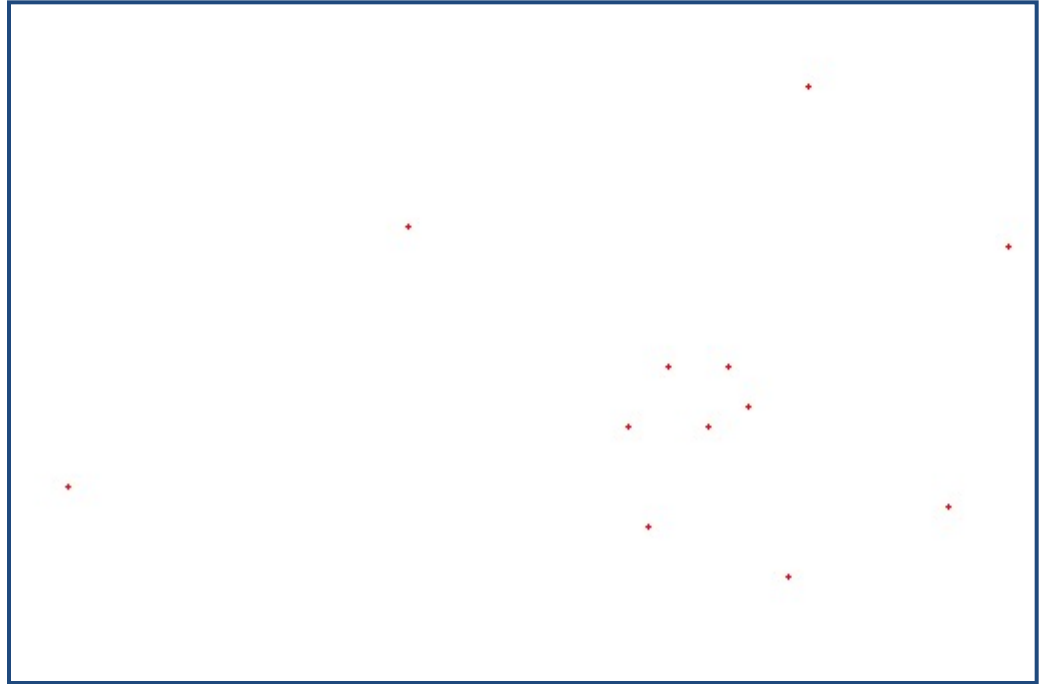
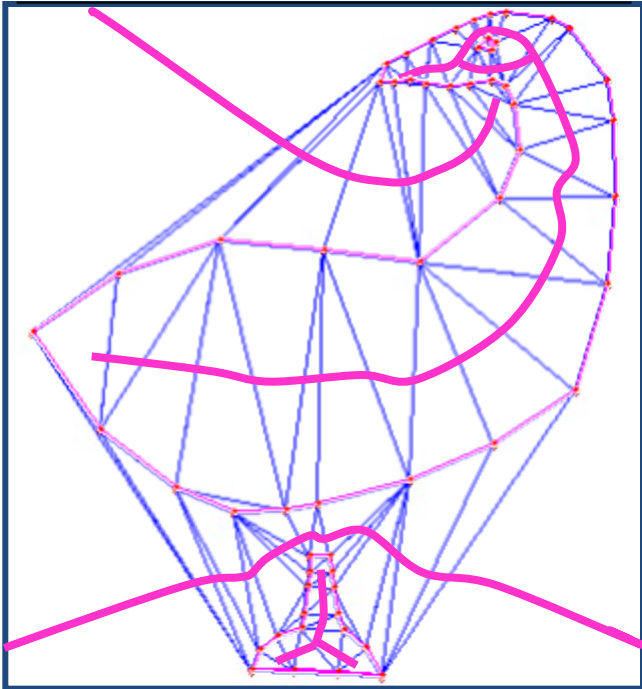


Recall:

Medial axis vs Voronoi diagram

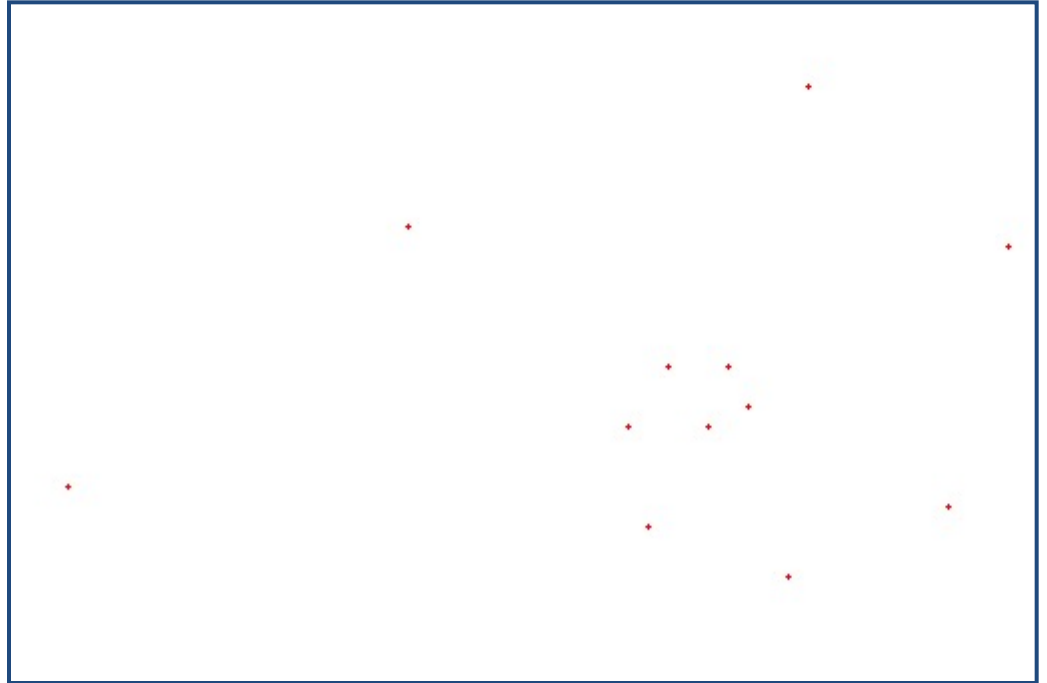
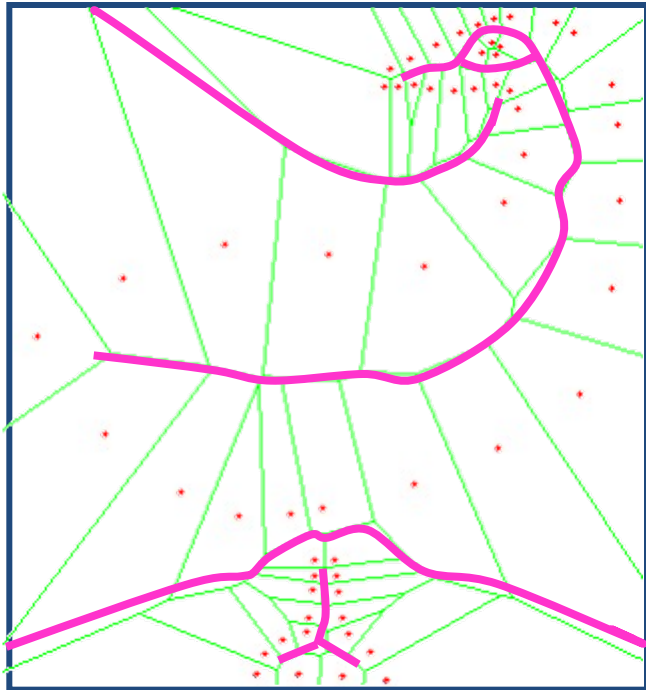


2D: connect the dots



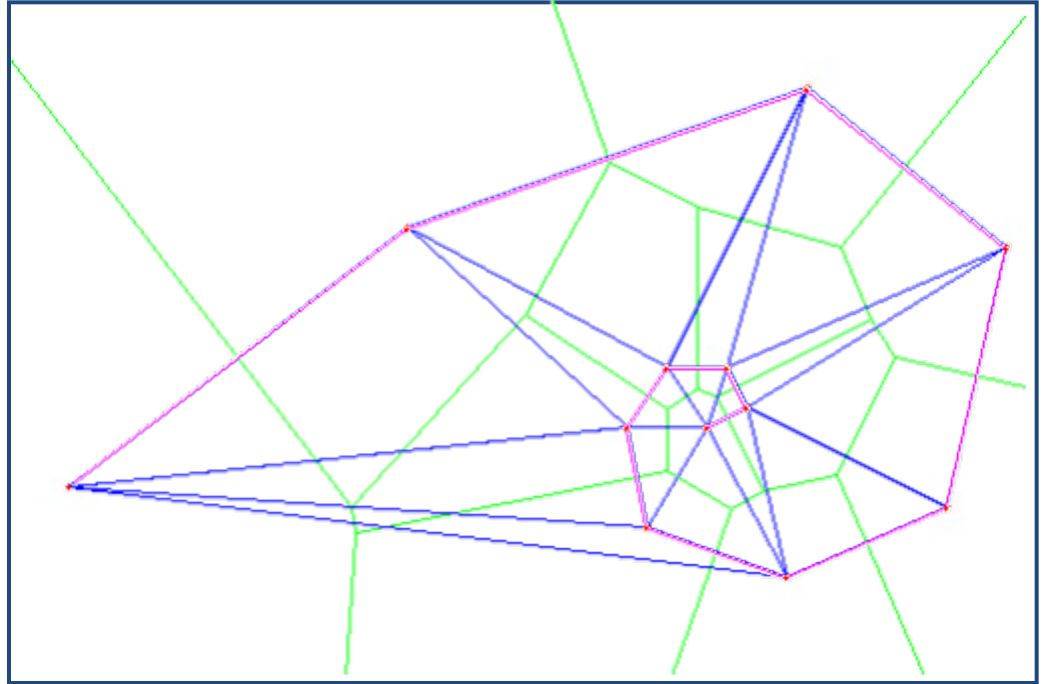
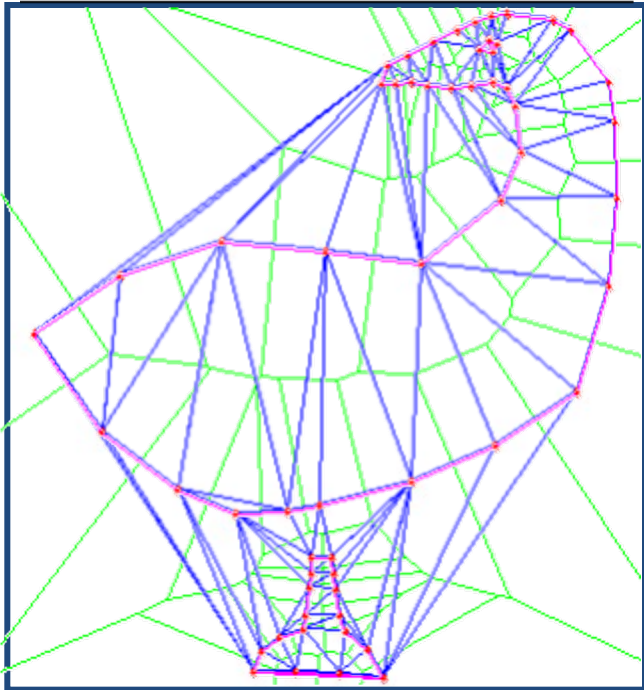
Edges should be “far” from Medial Axis

2D: connect the dots



Voronoi diagram approximates Medial Axis
if points are sampled densely enough

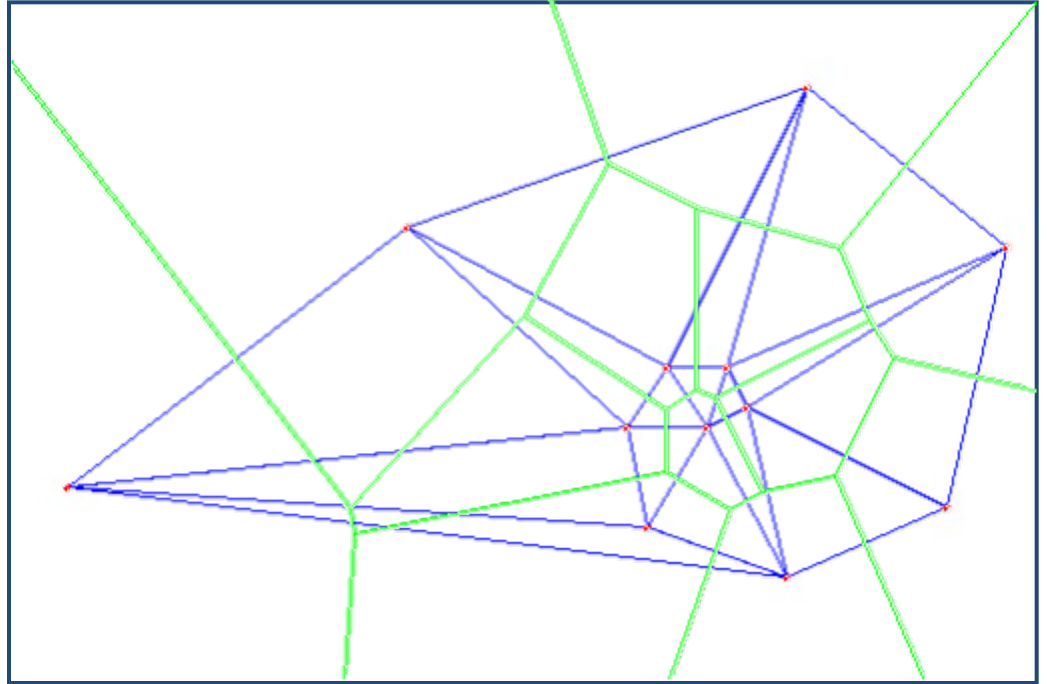
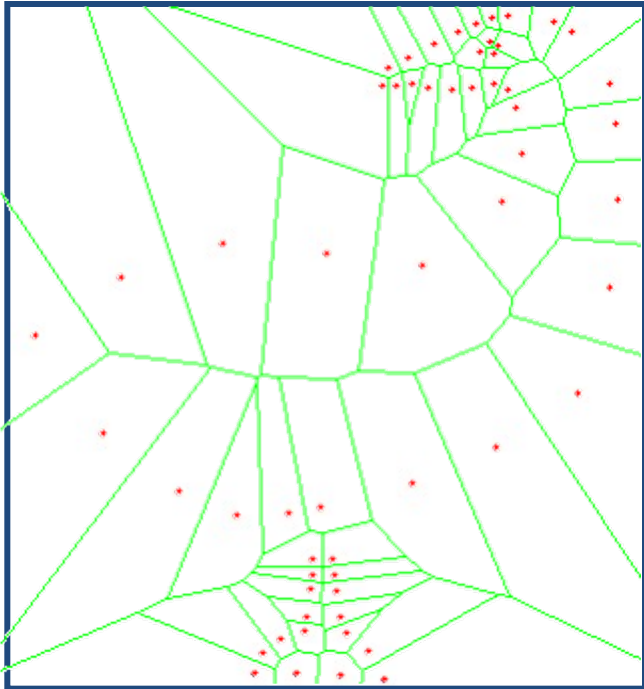
2D: connect the dots



Edge e in **crust** \Leftrightarrow

a circumcircle of e contains no other sample points or Voronoi vertices of S

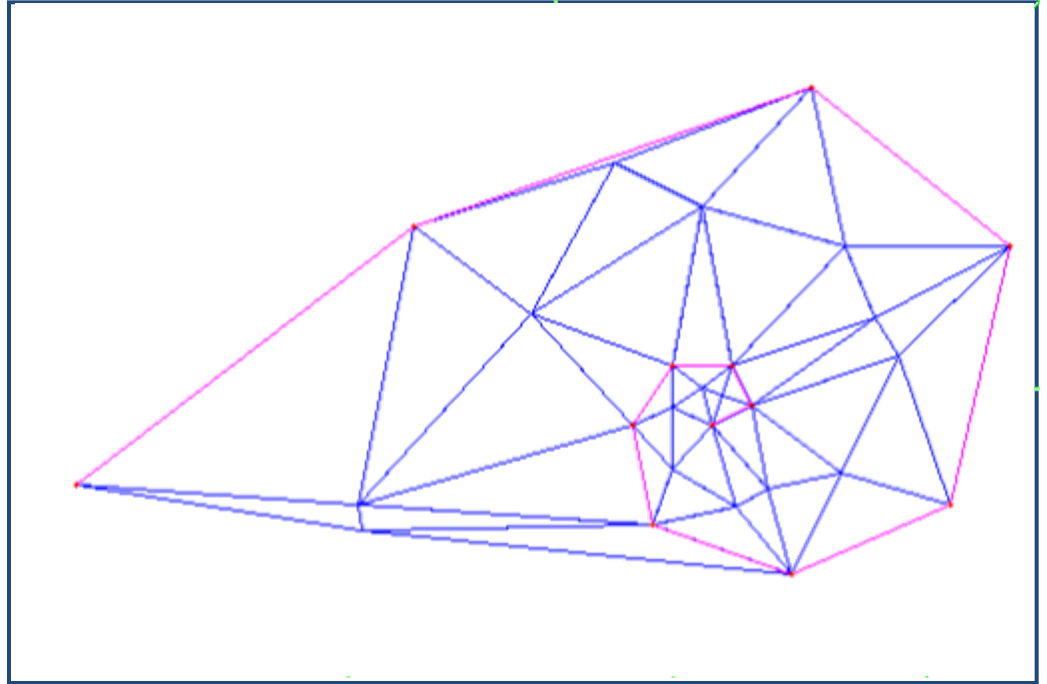
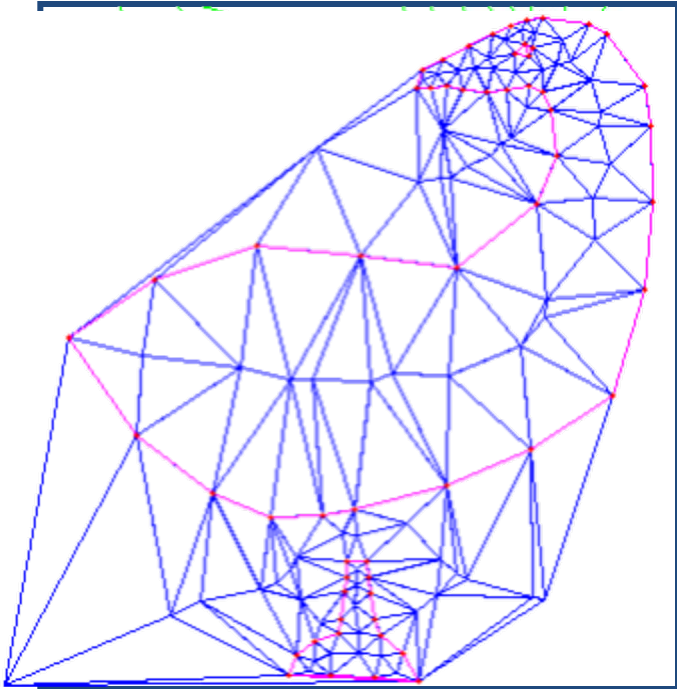
Crust: Algorithm



Compute Voronoi diagram of S

$V = \{\text{Voronoi vertices}\}$

Crust: Algorithm

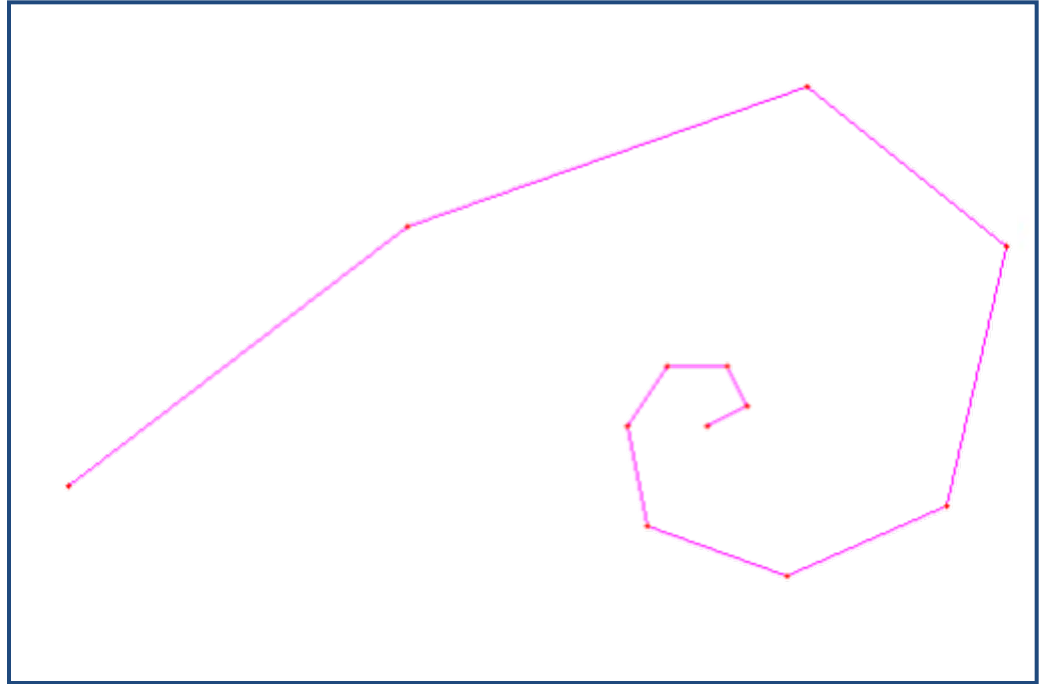
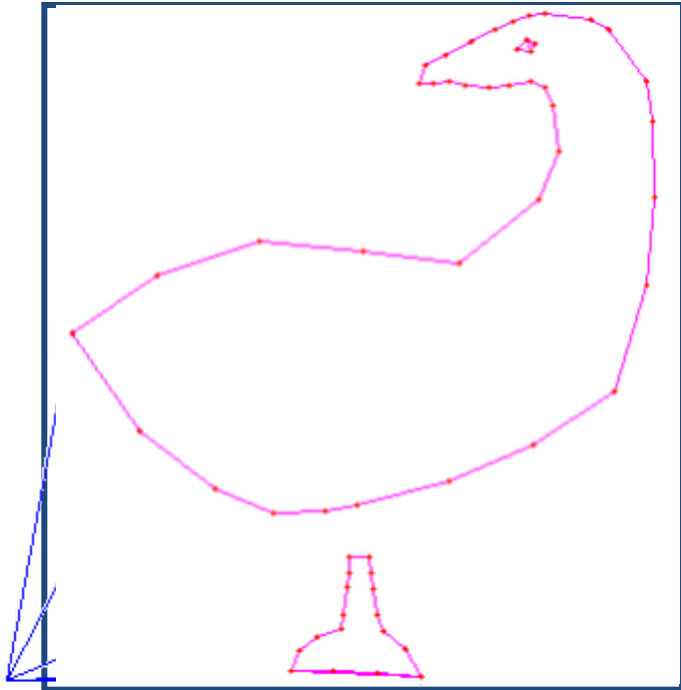


Compute Voronoi diagram of S

$$V = \{\text{Voronoi vertices}\}$$

Compute Delaunay Triangulation of $S \cup V$

Crust: Algorithm



Compute Voronoi diagram of S

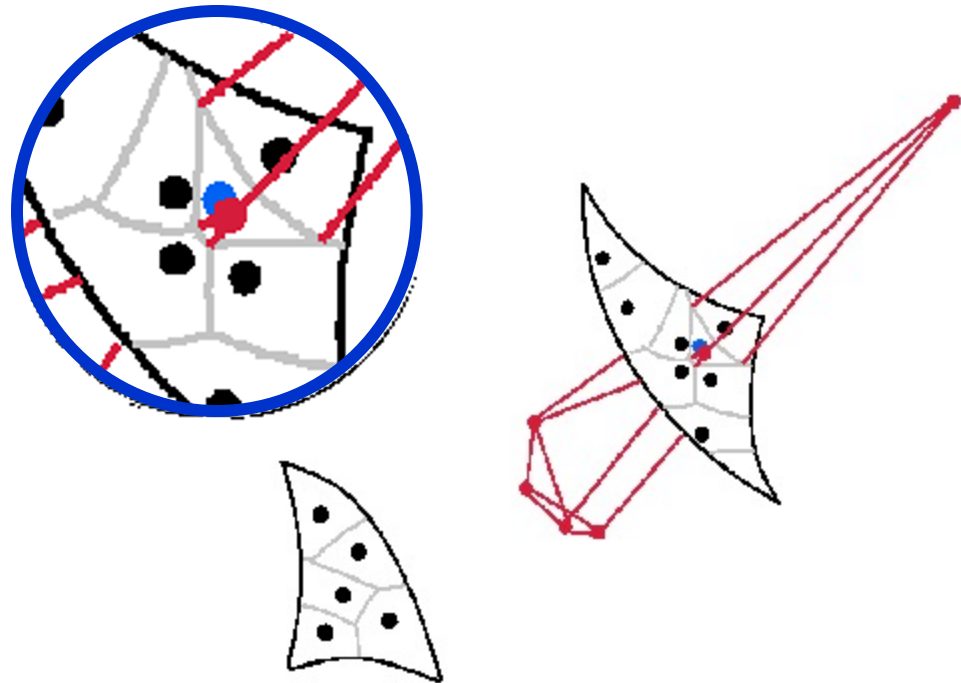
$$V = \{\text{Voronoi vertices}\}$$

Compute Delaunay Triangulation of $S \cup V$

Crust = all edges between points of S

3D Crust Algorithm

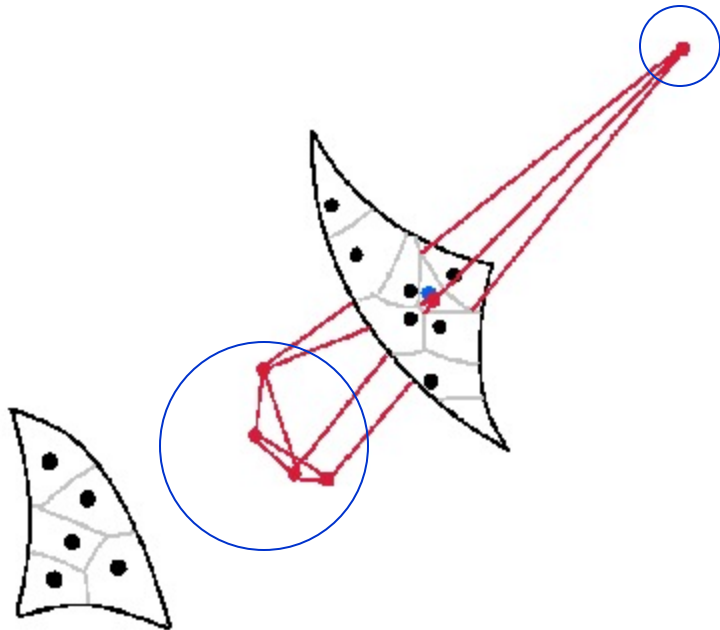
- Extend 2D approach
- Voronoi vertex is equidistant from 4 sample points
- BUT in 3D not all Voronoi vertices are near medial axis (regardless of sampling density)



3D Crust Algorithm

Some vertices of the Voronoi cell are near medial axis

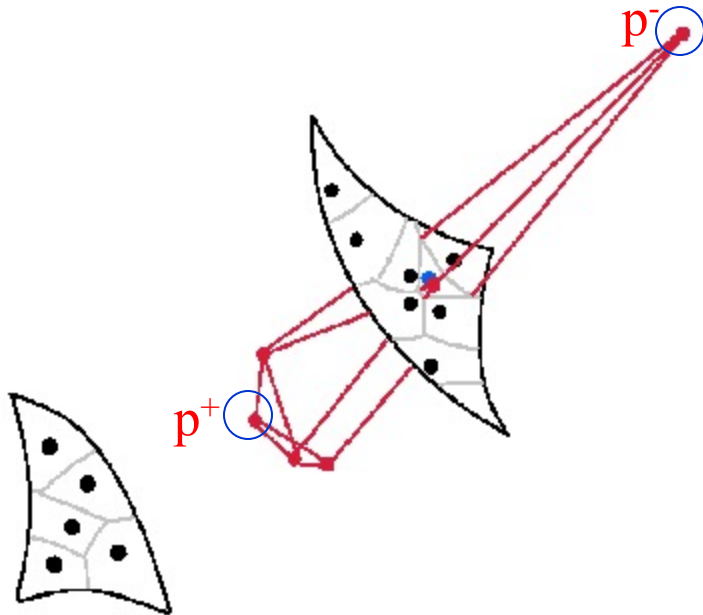
Intuitively – cell is closed not just from the sides but also from “top” & “bottom”



3D Crust Algorithm

Solution: use only two farthest vertices of V_s - one on each side of the surface

- Call vertices **poles** of s (p^+ , p^-)



3D Crust Algorithm

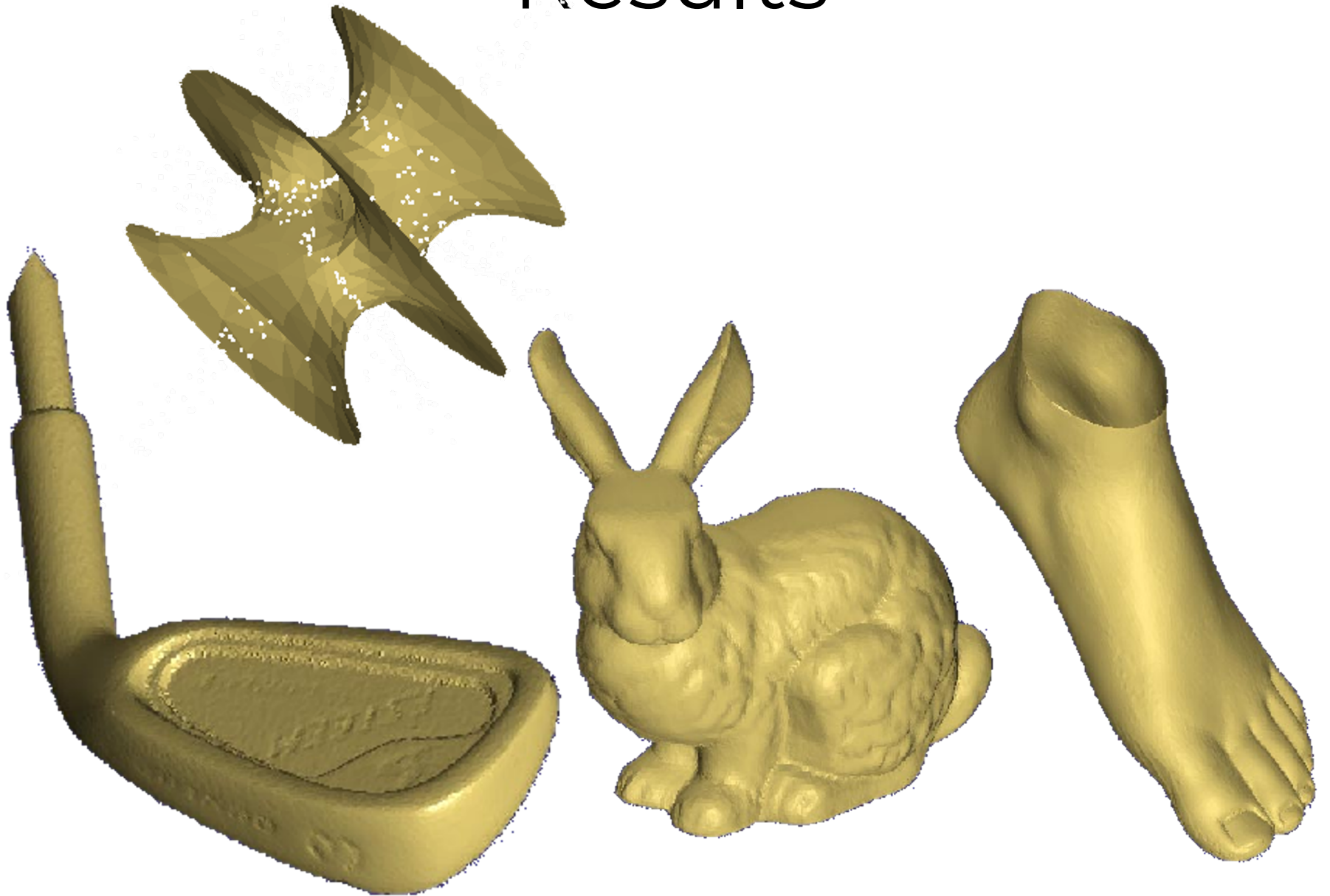
- Compute Voronoi diagram of S
- For each $s_i \in S$, compute

$$P = \{p_i^+, p_i^-\}$$

- Compute Delaunay triangulation T of $S \cup P$

Crust = all triangles in T with vertices in S

Results



Problems & Modifications

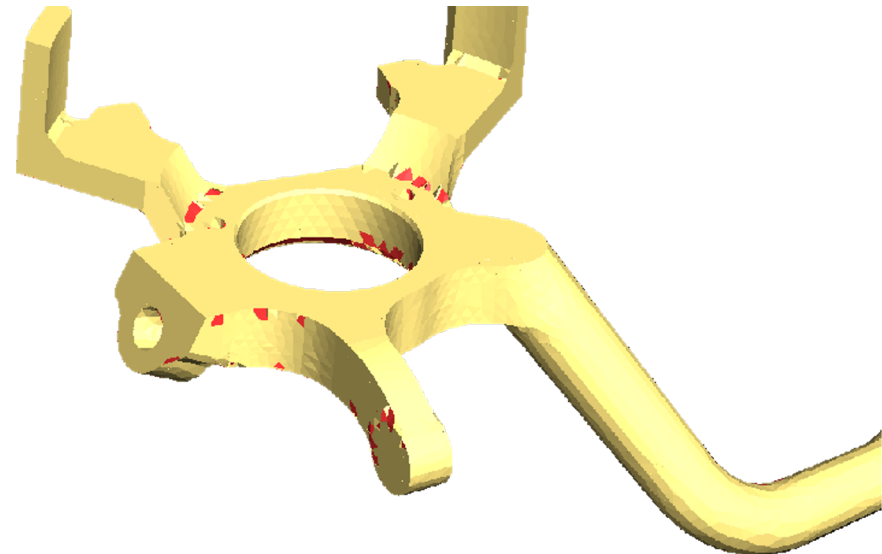
Correct in the absence of noise

Slow-*ish*

Need dense samples

Problems at sharp corners

Noise

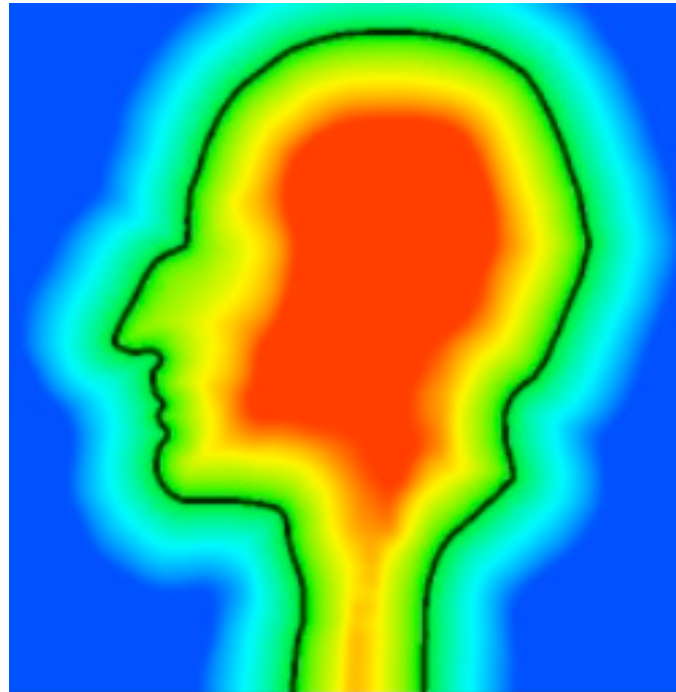


Methods

- Explicit, *or reconstruction circa 1998*
 - Zippering
 - Delaunay/Voronoi-based
- ***Implicit***
 - Signed distance function
 - Poisson
- Data-driven

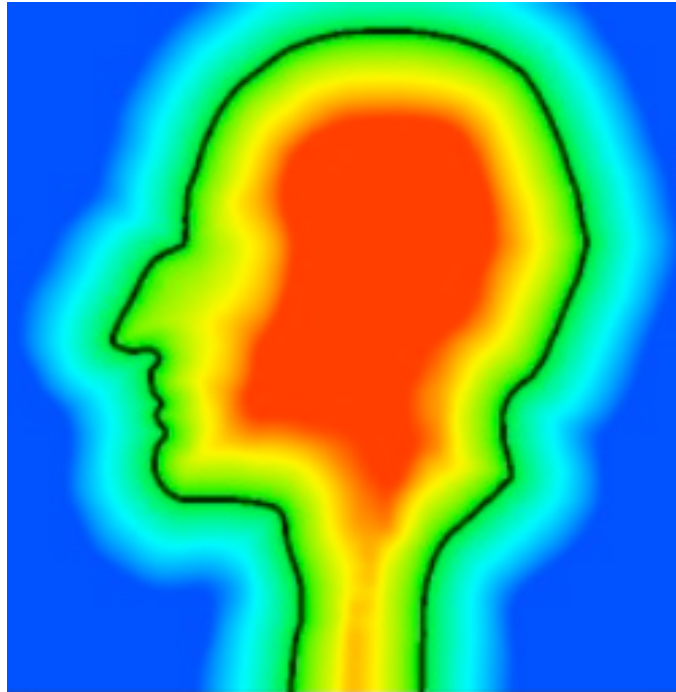
Implicit Reconstruction

1. Estimate signed distance function $d: \mathbb{R}^3 \rightarrow \mathbb{R}$
2. Extract an isosurface $d = 0$



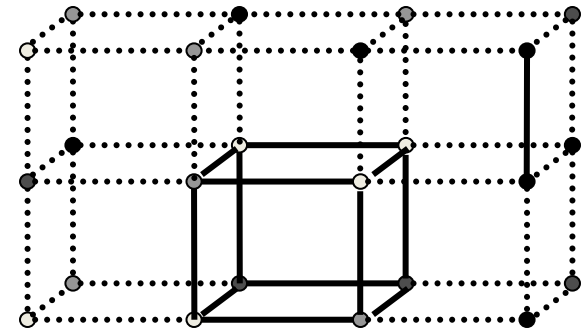
Implicit Reconstruction

1. Estimate signed distance function $d: \mathbb{R}^3 \rightarrow \mathbb{R}$
2. ***Extract an isosurface $d = 0$***



Marching Cubes

- Each voxel:
 - Has values at 8 corners
 - Has 256 possible configurations
 - 15 after counting symmetries and rotations
 - Either
 - Inside isosurface
 - Outside isosurface
 - **Intersects** isosurface
- Can extract triangulation independently per voxel



Marching Cubes

For each *intersecting* voxel contains triangles of the isosurface

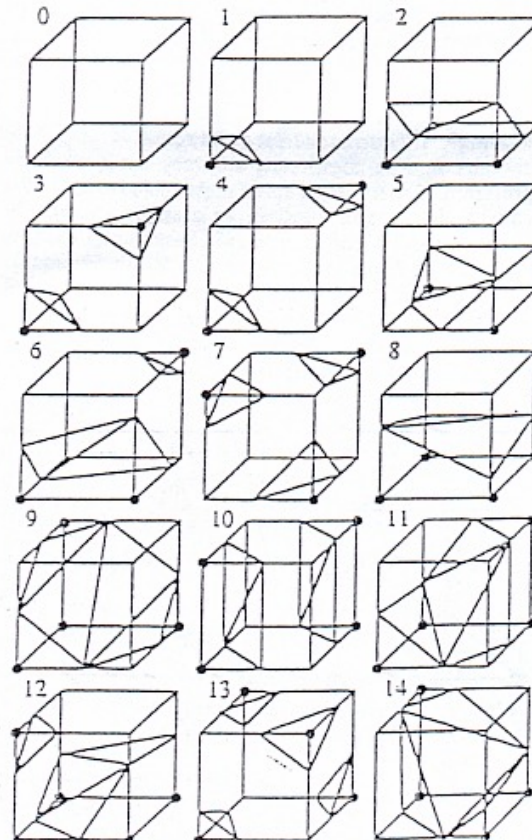
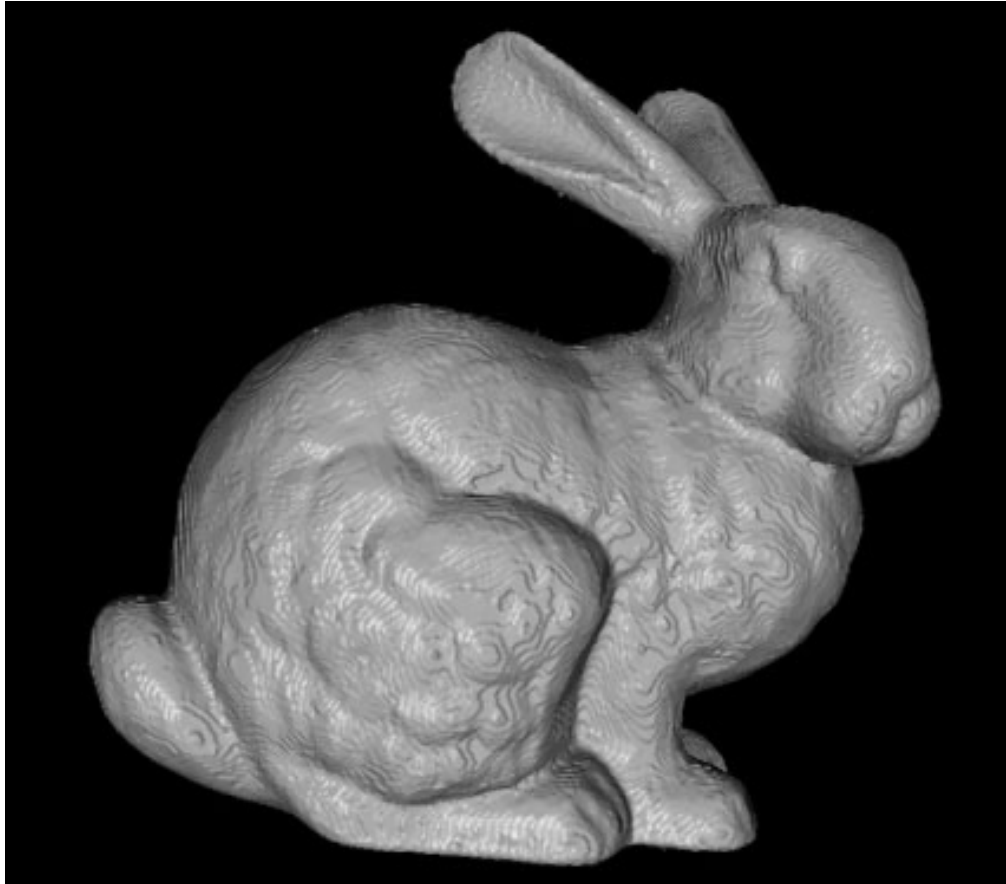


Figure 2. Configurations.

Configurations

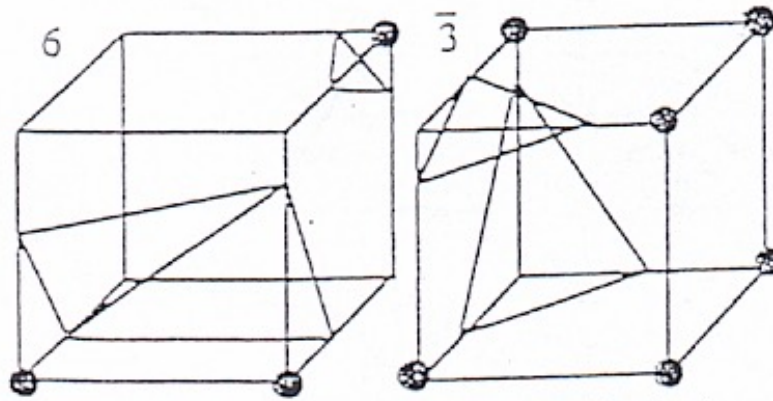
- For each configuration add 1-4 triangles to isosurface
- Isosurface vertices computed by:
 - Interpolation along edges (according to grid values)

Example



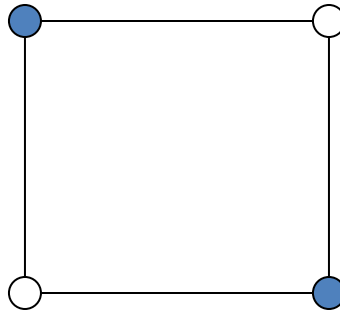
Problem

Can produce non-manifold results
and wrong genus

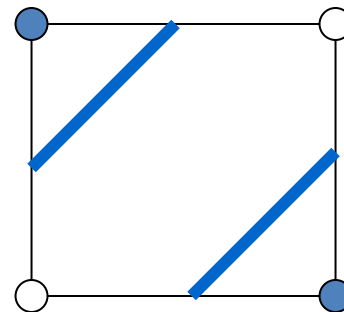
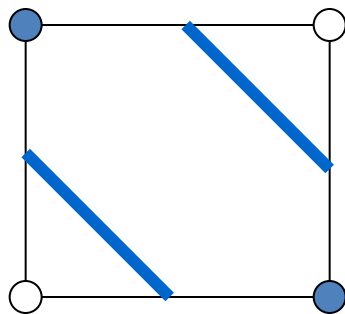


- What if those two are adjacent?
 - Each is ambiguous
- Consistency?

Ambiguous Faces



- Two locally valid interpretations



- Source of MC consistency problem

Solution

For those cases, store multiple triangulations

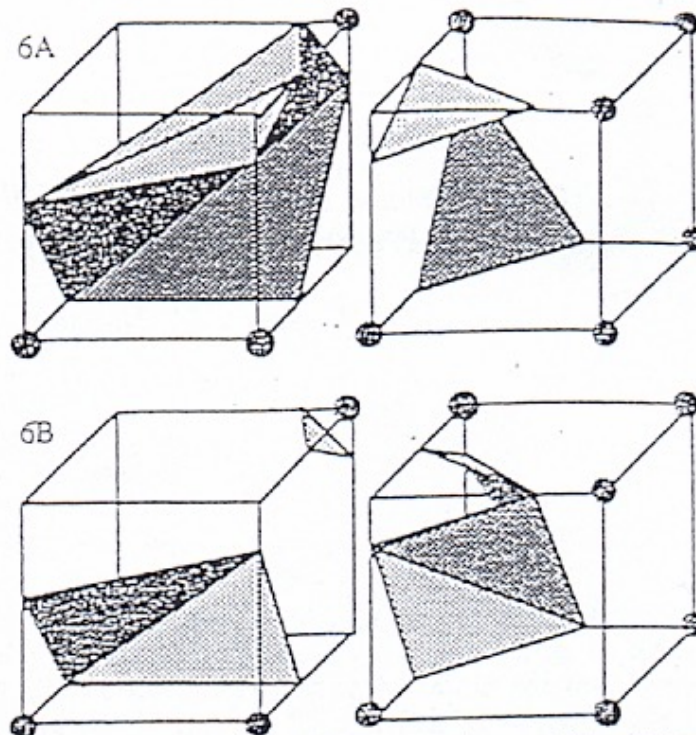


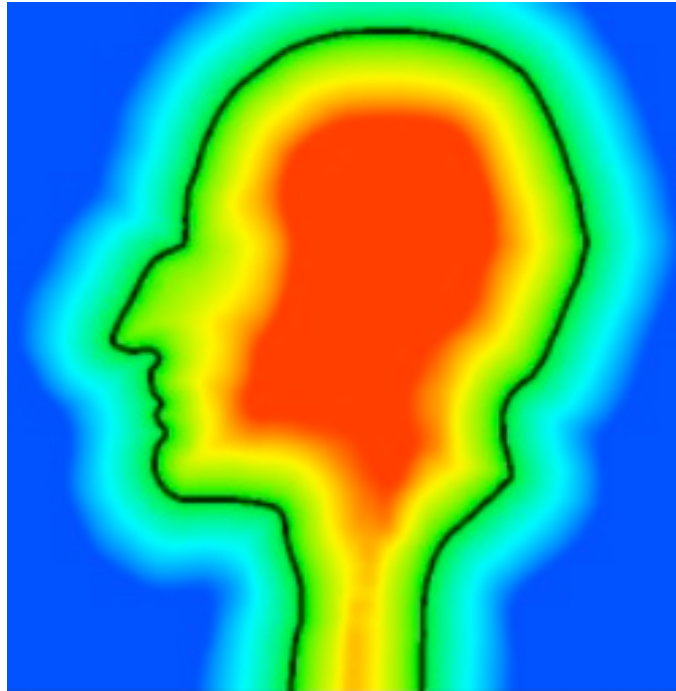
Figure 4. Two possible triangulations which yield a topologically correct isovalue surface.

2.0 Asymptotic Decider

Choose one that agrees with neighbor voxels

Implicit Reconstruction

1. **Estimate signed distance function $d: \mathbb{R}^3 \rightarrow \mathbb{R}$**
2. Extract an isosurface $d = 0$



Signed distance function

Distance to points is not enough

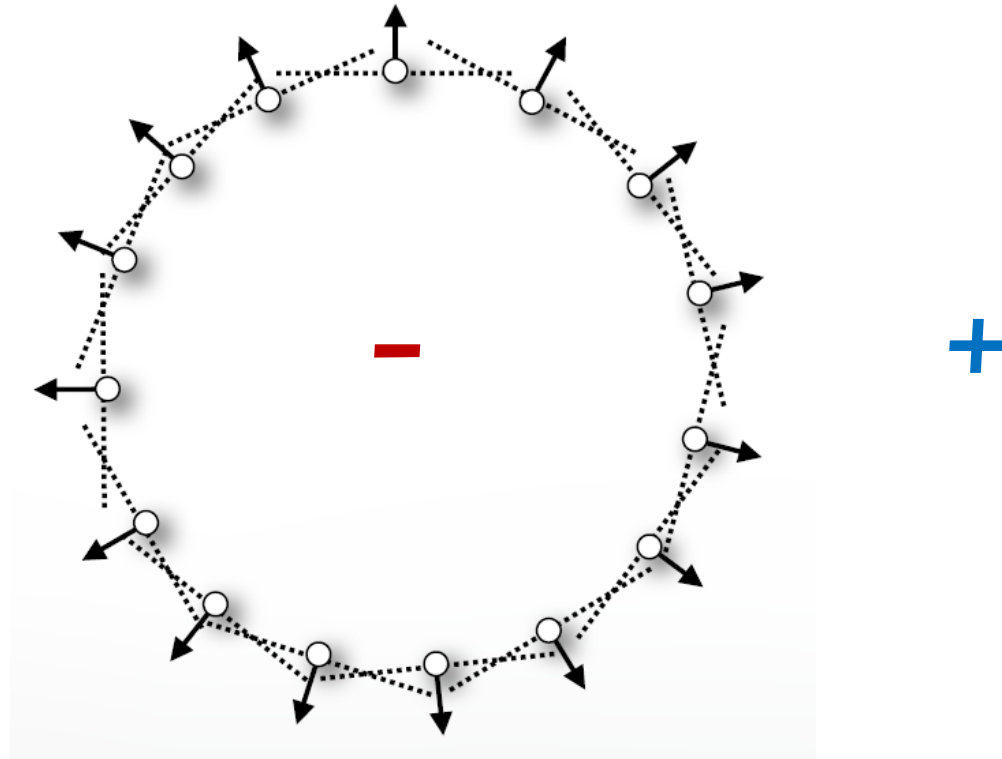
Need more structure



Signed distance function

How can we tell inside from outside?

Estimate normals.



Estimating normals

- Fit a plane into neighborhood of each point
 - Neighborhood = k nearest neighbors
- Determine **consistent** normal orientation



Estimating normals

- Fit a plane into neighborhood of each point
 - Neighborhood = k nearest neighbors
 - Use spatial decompositions (BSP-trees)
- Determine **consistent** normal orientation



Fitting plane

$$\min_{c \in \mathbb{R}^3, \|n\|=1} \sum_i (n^T (p_i - c))^2$$

On the board, time permitting

Estimating normals

- Fit a plane into neighborhood of each point
 - Neighborhood = k nearest neighbors
- Determine **consistent** normal orientation



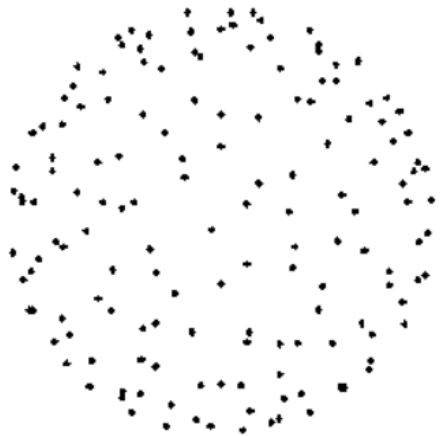
Estimating normals

- Fit a plane into neighborhood of each point
 - Neighborhood = k nearest neighbors
- Determine **consistent** normal orientation
 - Make sure $n_i \cdot n_j > 0$ for neighbors



Signed Distance Function

- Distance to tangent planes
 - [Hoppe et al. '92]



150 samples



reconstruction
on 50^3 grid

Signed Distance Function

- Smoother: RBF basis



Hoppe '92



Compact RBF
Wendland C^2



Global RBF
Triharmonic

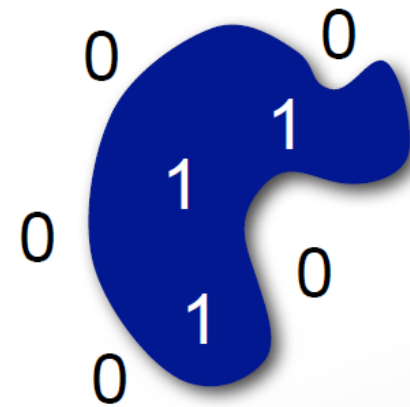
Signed Distance Function

- Poisson surface reconstruction
 - [Kazhdan et al. '06]

Signed Distance Function

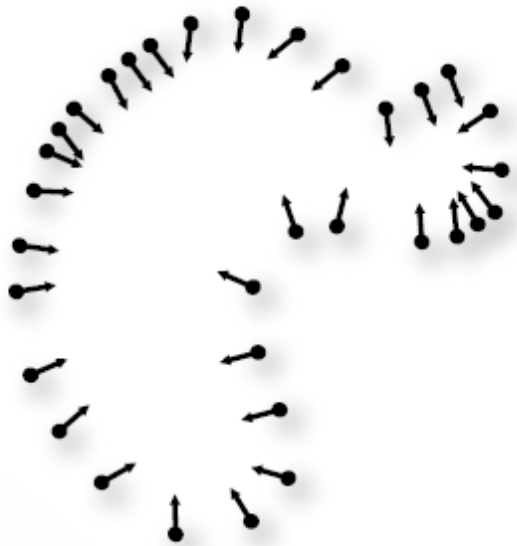
- Poisson surface reconstruction
 - Solve for indicator function

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

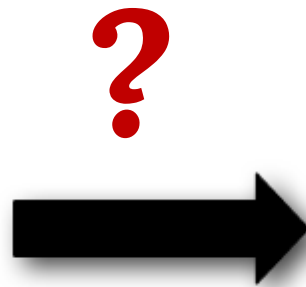


Indicator function

χ_M



Oriented points



Indicator function

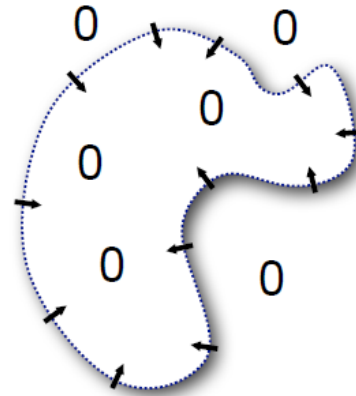
$$\chi_M$$

Idea

Oriented normals =
gradient of an indicator function?



Oriented points



Indicator gradient

$$\nabla \chi_M$$

Idea

Oriented normals \Rightarrow vector field \vec{V}

Find indicator function:

$$\min_{\chi} \|\vec{V} - \nabla\chi\|^2$$

Idea

Oriented normals \Rightarrow vector field \vec{V}

Find indicator function:

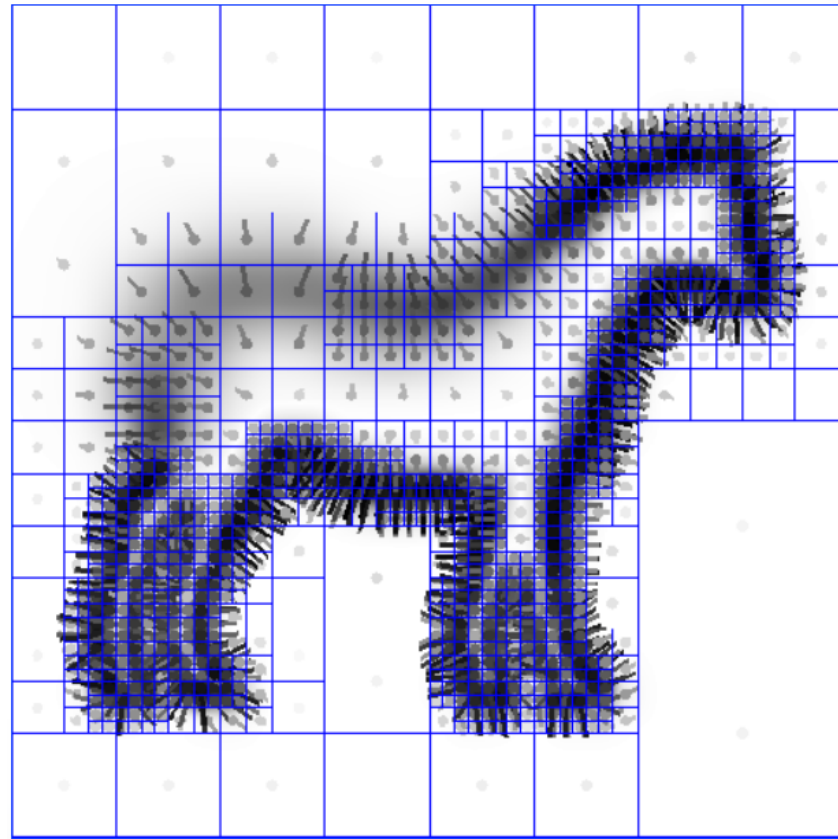
$$\min_{\chi} \|\vec{V} - \nabla\chi\|^2$$

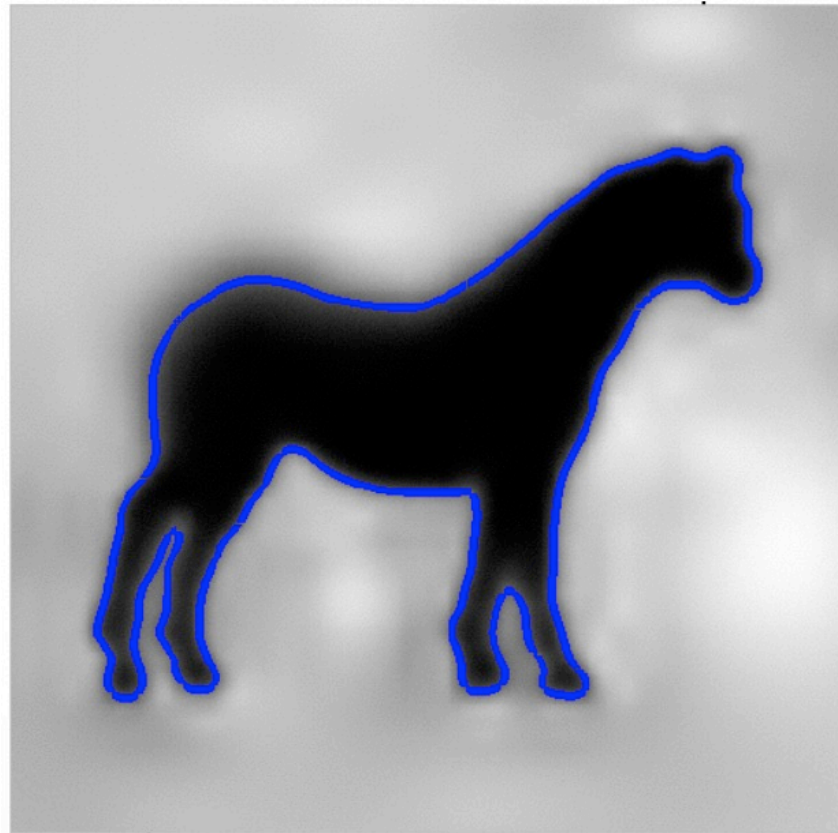
Differentiate,

Poisson
equation

$$\Delta\chi = \nabla \cdot \vec{V}$$

Process





Results

