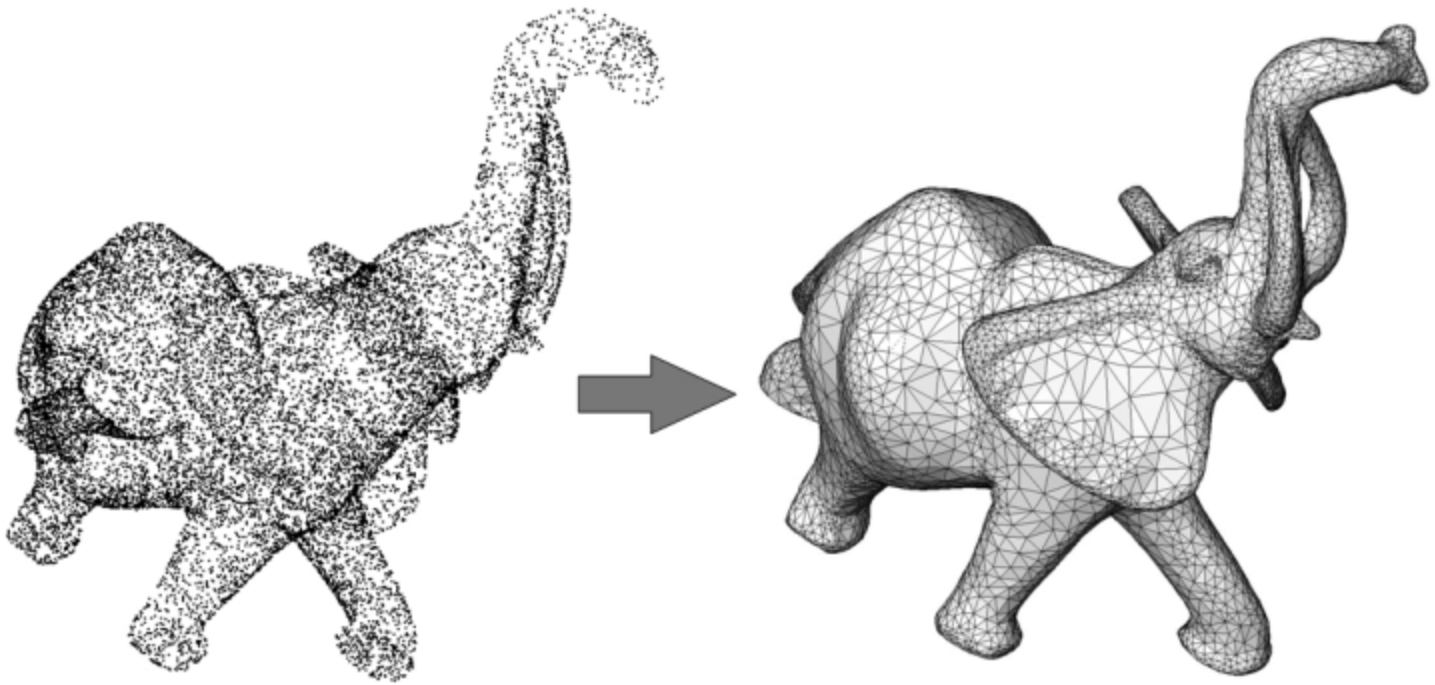
CMSC740
Advanced Computer Graphics

Matthias Zwicker
Fall 2025

Surface reconstruction

- Input: point cloud
- Output: watertight triangle mesh

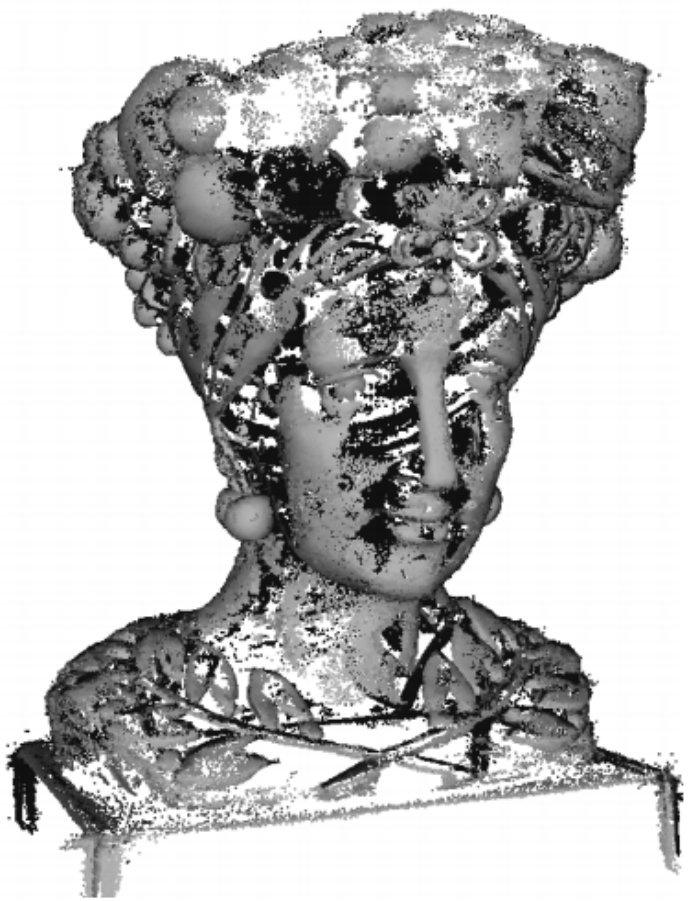


Challenges

Input data quality

- Noise
- Holes

How to assess output quality?



Input

Output

http://mesh.brown.edu/ssd/pdf/Calakli-pg2011.pdf

Quality measures

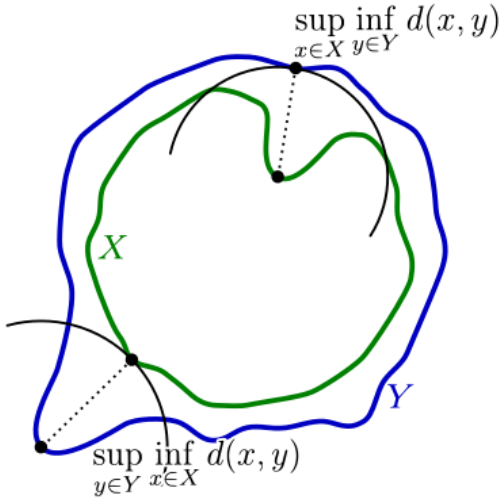
- Hausdorff distance: distance between two subsets of metric space
  - Surface: subset of all 3D points
  - “Greatest of all distances from point in one set to closest point in other set”
- Distance of point a to subset B

$$d(a, B) = \inf_{b \in B} d(a, b)$$

- Definition of Hausdorff distance

$$d_{\mathrm{H}}(X, Y) = \max \left\{ \sup_{x \in X} d(x, Y), \sup_{y \in Y} d(X, y) \right\}$$

- Efficient computation non-trivial
  https://dl.acm.org/doi/abs/10.3233/ICA-170544



Quality measures

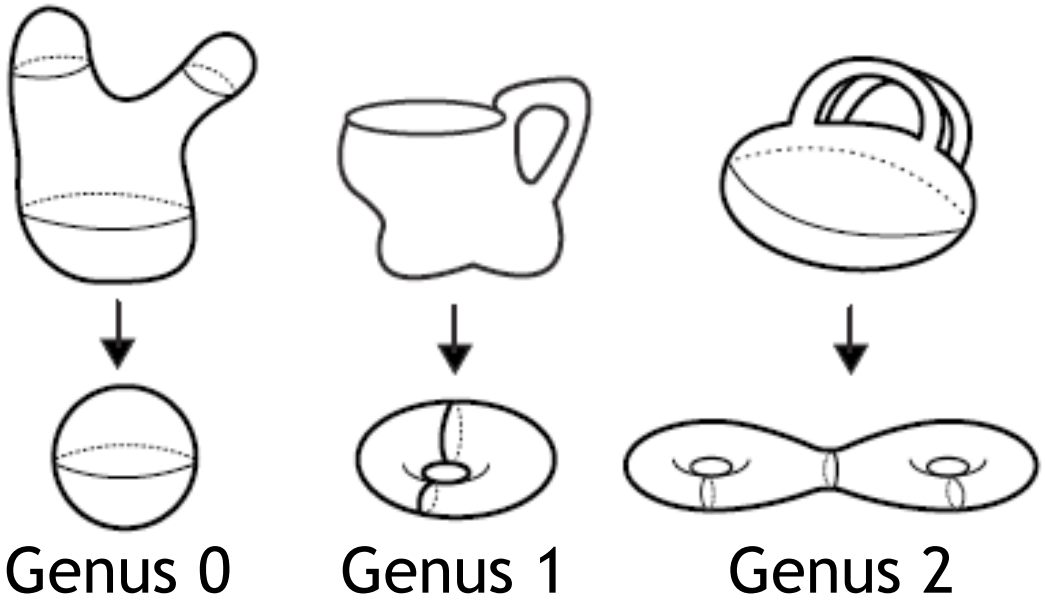
- Chamfer distance between discrete point sets

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

- In practice, random sampling of continuous surfaces to compute sum

Quality measures

- Preserve topology (number of holes, handles in closed surface)



Two examples today

1. Combinatorial
   - Connect input data points to mesh
2. Function fitting
   - Implicit function in 3D

- Many, many more variations from recent research
- «A Survey of Surface Reconstruction from Point Clouds”, Berger et al. 2016
  https://hal.inria.fr/hal-01348404v2/document

Two examples today

1. Combinatorial
   - Connect input data points to mesh
2. **Function fitting**
   - Implicit function in 3D

Gradients

- Implicit surface defined by function

  $f(\mathbf{x}) : \mathrm{R}^3 \text{ -> } \mathrm{R}$, $\mathbf{x} = (x,y,z)$, as set $\{\mathbf{x} \mid f(\mathbf{x})=C\}$

- Gradients

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix}$$

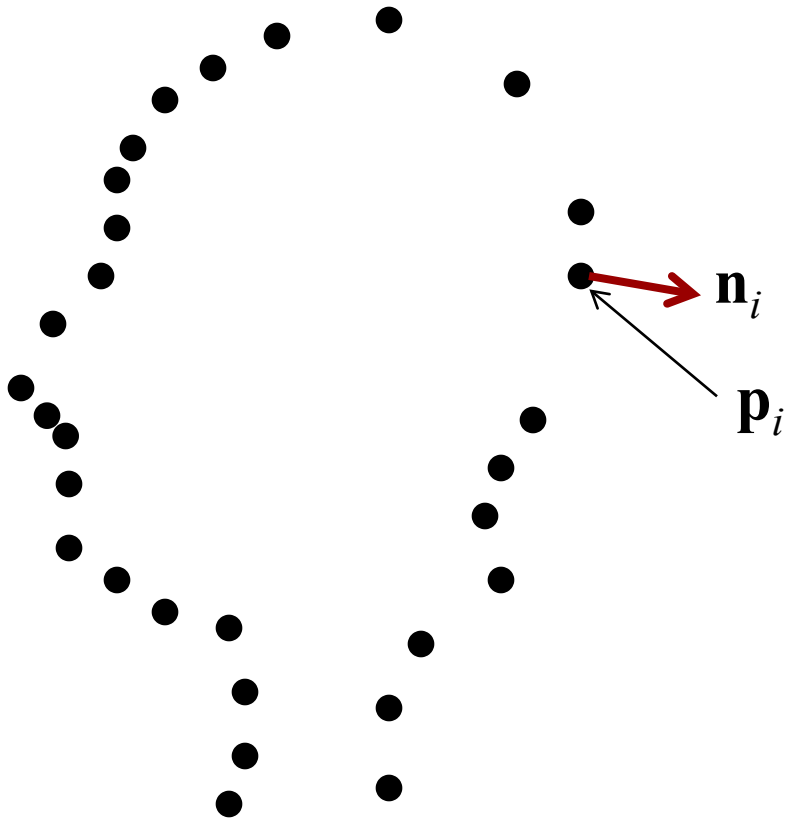
- Gradients are normals of isosurfaces

  $\{\mathbf{x} \mid f(\mathbf{x})=const\}$, in particular $\{\mathbf{x} \mid f(\mathbf{x})=0\}$

  for any value of $const$

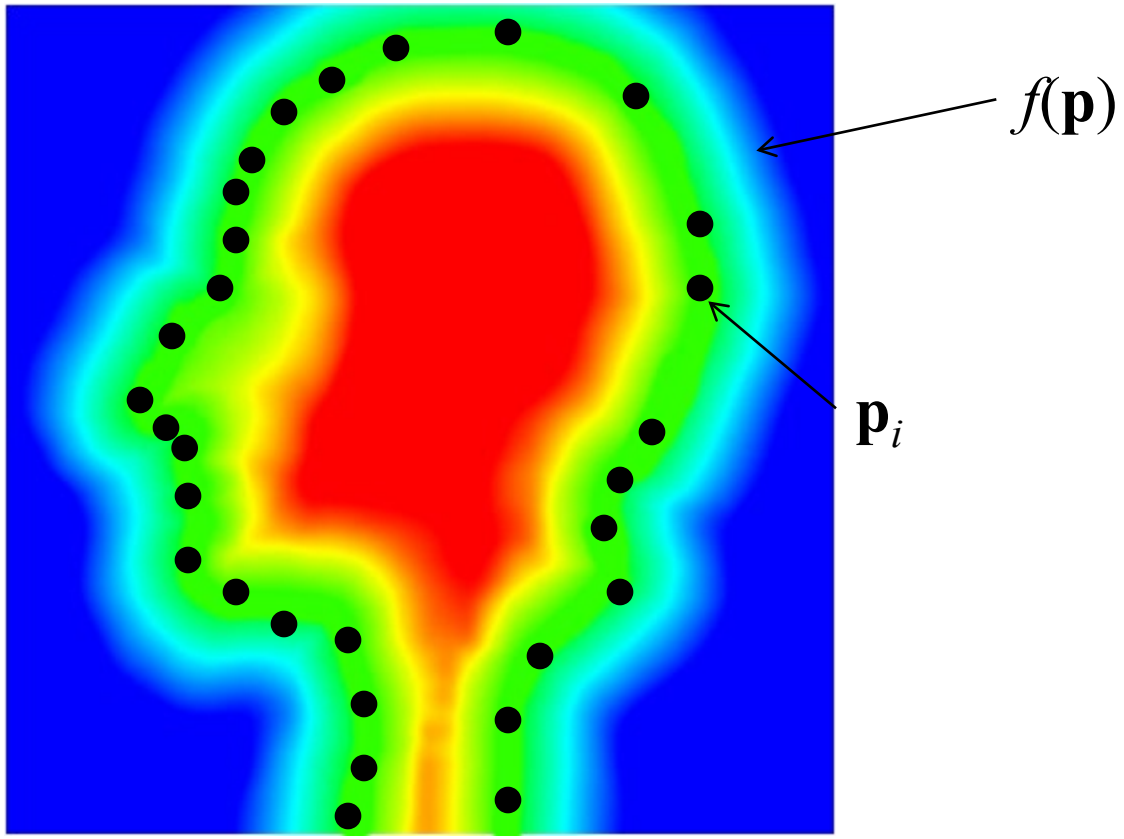
Overview

- Input: points $\mathbf{p}_i$, optional normals $\mathbf{n}_i$



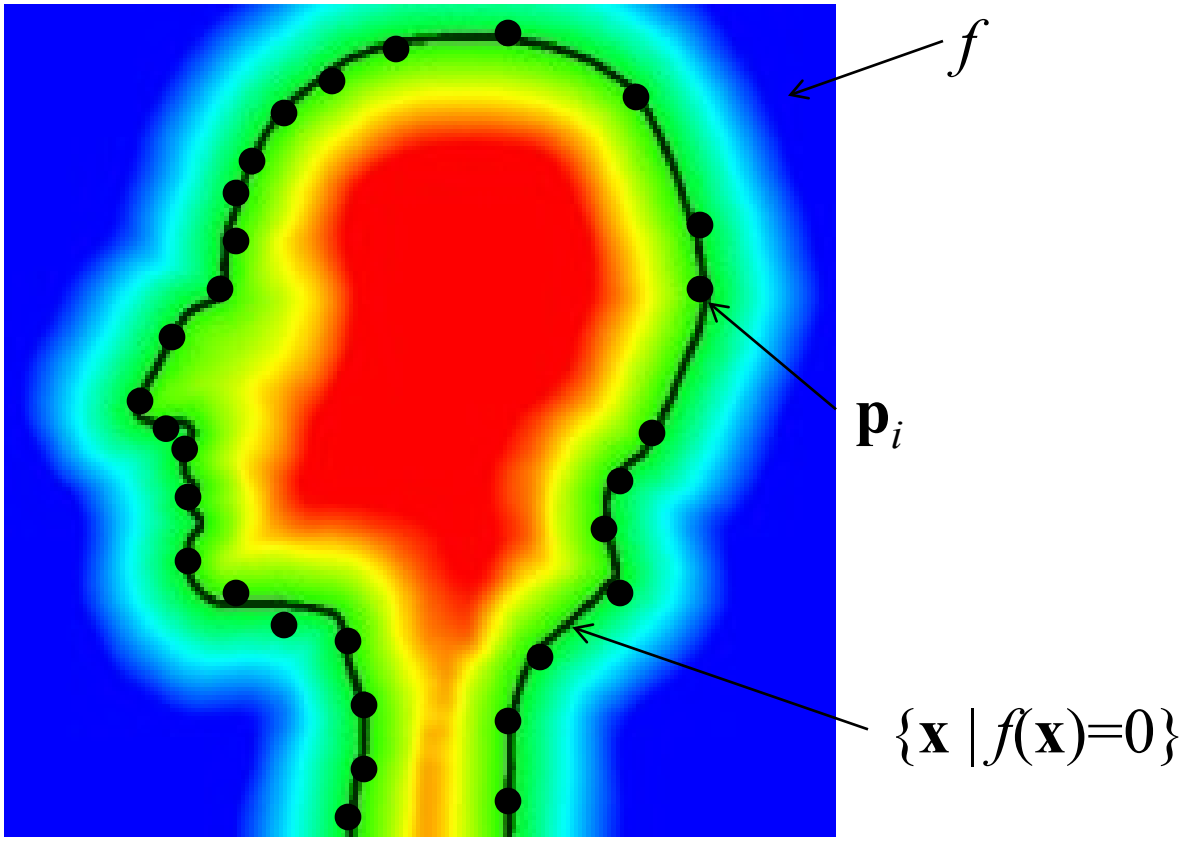
Overview

1. Find function f such that for all i: $f(\mathbf{p}_i)=0$, and additional contraints, such as $\|\nabla f(\mathbf{p})\|=1$ for all $\mathbf{p}$, or $\nabla f(\mathbf{p}_i)=\mathbf{n}_i$ for all i if normals $\mathbf{n}_i$ available



Overview

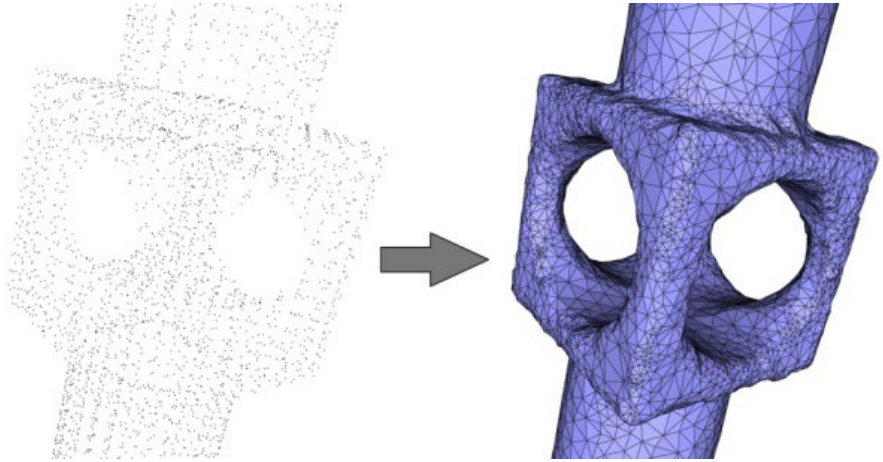
2. Triangulate isosurface $\{\mathbf{x} \mid f(\mathbf{x})=0\}$



Poisson surface reconstruction

- "Screened Poisson Surface Reconstruction", Kazhdan and Hoppe, ACM TOG 2013
- Standard approach often used in practice
- Implicit function is defined by 3D grid of function values, with trilinear interpolation
  - Implicit function represented as piecewise linear function https://en.wikipedia.org/wiki/Piecewise_linear_function
- Code, etc.

http://www.cs.jhu.edu/~misha/Code/PoissonRecon/Version8.0/
https://doc.cgal.org/latest/Poisson_surface_reconstruction_3/index.html



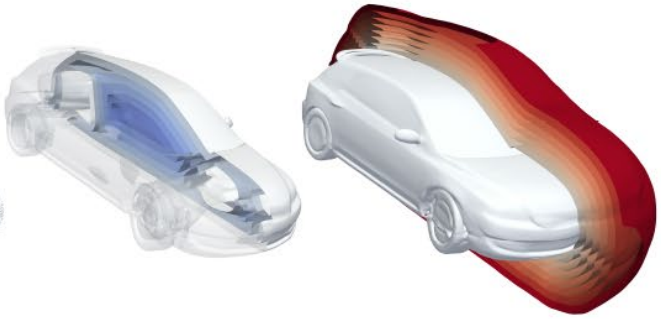
Distance function

- Implicit function with $\|\nabla f(\mathbf{p})\|=1$ for all $\mathbf{p}$
  - Signed distance function (SDF), if gradient is continuous
- Deep signed distance function: use neural network to represent $f(\mathbf{x})$
  - Instead of piecewise linear, or spline (piecewise polynomial) function

Deep SDF fitting

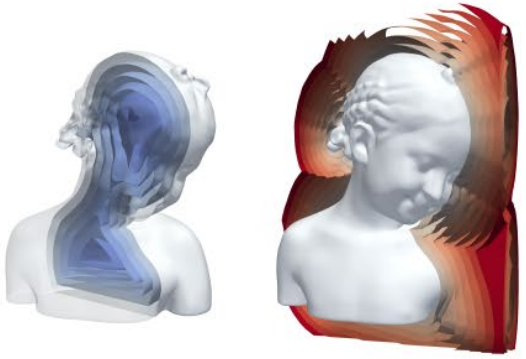
- "IGR: Implicit Geometric Regualrization for Learning Shapes" (ICML 2020), https://github.com/amosgropp/IGR
- Reconstruct continuous SDF represented by neural network from point cloud
  - Optimization ("training") for each point cloud, no data set reqired

Input



SDF as neural network

Input



SDF as neural network

Formulation

- Point cloud $\mathcal{X} = \{\boldsymbol{x}_i\}_{i \in I} \subset \mathbb{R}^3$
- Optional normals $\mathcal{N} = \{\boldsymbol{n}_i\}_{i \in I} \subset \mathbb{R}^3$
- Multilayer perceptron (MLP), parameters θ

  $f(\boldsymbol{x}; \theta)$: $\mathbb{R}^3 \times \mathbb{R}^m \to \mathbb{R}$

- Loss

  Ensure f is distance function

  $$\ell(\theta) = \ell_{\mathcal{X}}(\theta) + \lambda \mathbb{E}_{\boldsymbol{x}} \left(\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x}; \theta)\| - 1 \right)^2$$

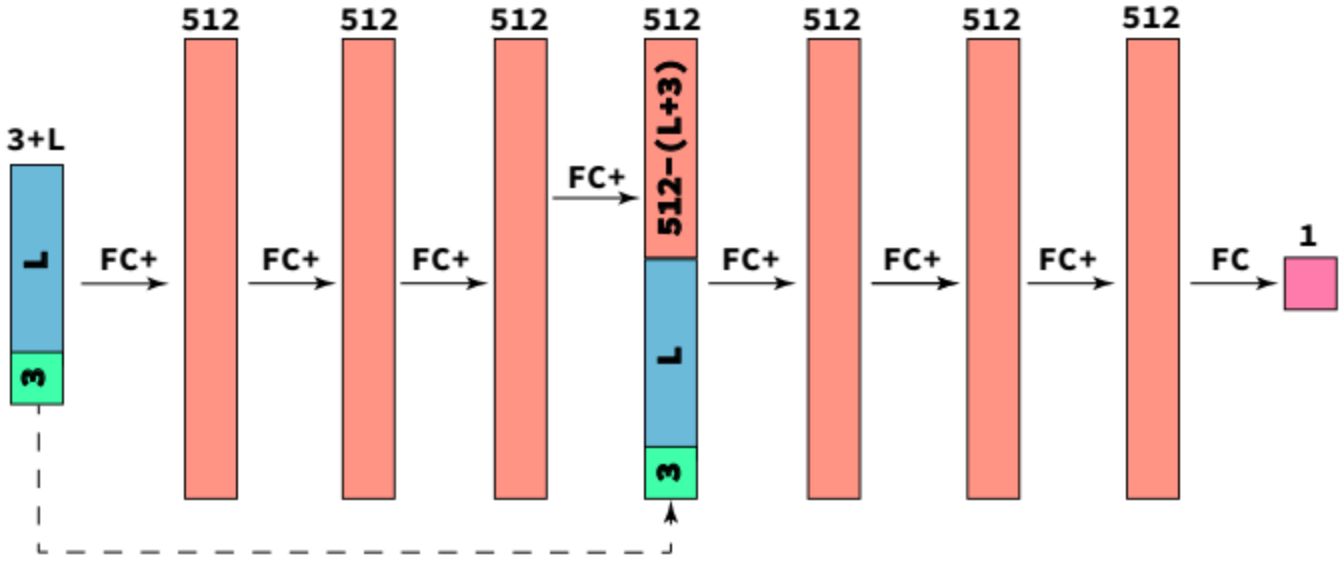
  $$\ell_{\mathcal{X}}(\theta) = \frac{1}{|I|} \sum_{i \in I} \left(|f(\boldsymbol{x}_i; \theta)| + \tau \|\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_i; \theta) - \boldsymbol{n}_i\| \right)$$

  Constrain function values at input points

  - If normals are available $\tau = 1$, else $\tau = 0$

Network architecture

- “Autodecoder”
- Input: Latent code L (learnable), 3D location
- Output: signed distance value (scalar)



FC: fully connected

Optimization

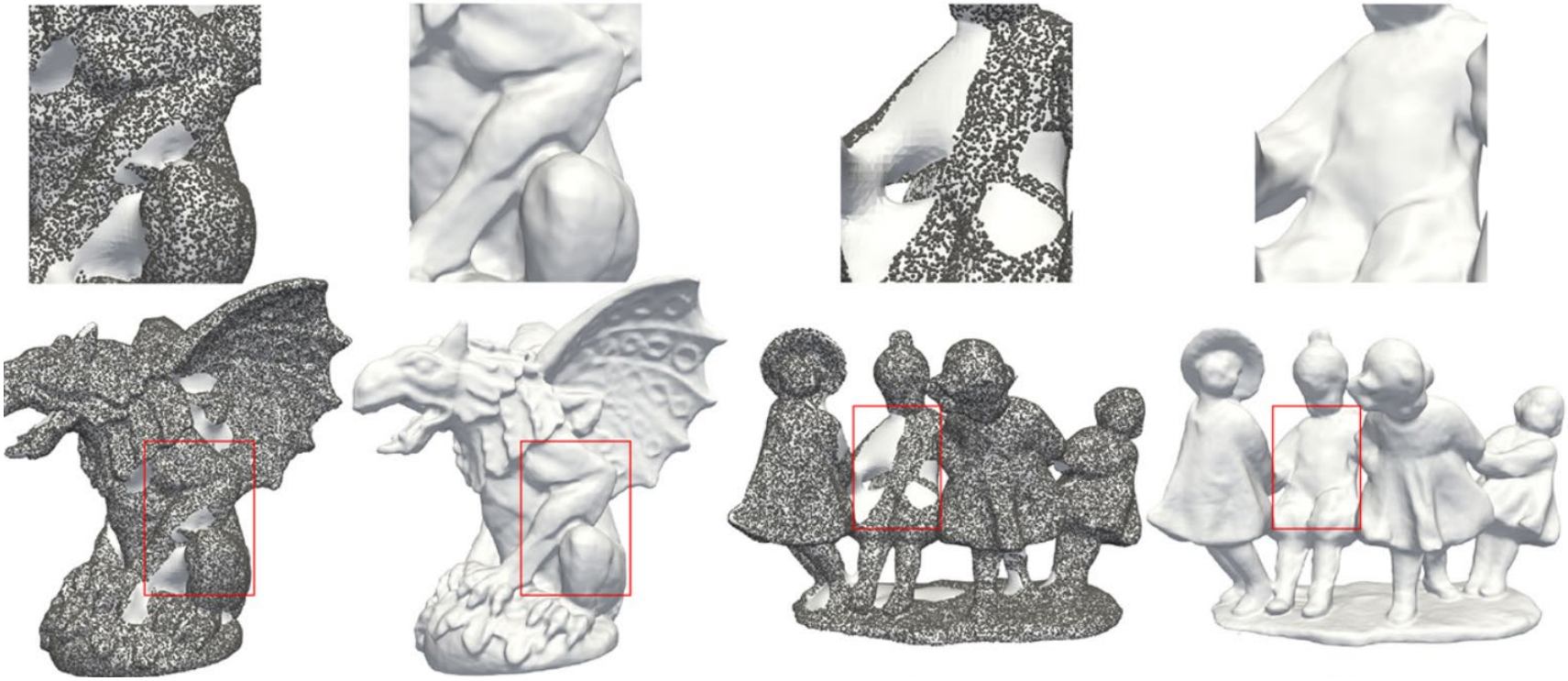
- Stochastic gradient descent with network parameters θ as variables
- Randomly sample 3D points $\boldsymbol{x}$ and data points $\boldsymbol{x}_i$

Why does it work?

- Using a “soft penalty” to ensure unit length gradients usually doesn’t guarantee unit length gradients in solution
- Solution is not unique; why does stochastic gradient descent on network parameters find a “reasonable” solution?
- Paper shows some analysis why a planar point cloud leads to SDF that exctly represents plane (and not “wiggly” surface)

Results

- State of the art accuracy



Previous SOTA | New approach | Previous SOTA | New approach

https://arxiv.org/pdf/2002.10099.pdf

Results

- DGP: previous state of the art (also neural network based)
- Chamfer distance d_C (one sided $d_{\vec{C}}$)
- Hausdorff distance d_H (one sided $d_{\vec{H}}$)

	Method	Ground Truth d_C	Ground Truth d_H	Scans $d_{\vec{C}}$	Scans $d_{\vec{H}}$
Anchor	DGP	0.33	8.82	**0.08**	2.79
	Ours	**0.22**	**4.71**	0.12	**1.32**
Daratech	DGP	**0.2**	**3.14**	**0.04**	1.89
	Ours	0.25	4.01	0.08	**1.59**
Dc	DGP	0.18	4.31	**0.04**	**2.53**
	Ours	**0.17**	**2.22**	0.09	2.61
Gargoyle	DGP	0.21	5.98	**0.062**	3.41
	Ours	**0.16**	**3.52**	0.064	**0.81**
Lord Quas	DGP	0.14	3.67	**0.04**	2.03
	Ours	**0.12**	**1.17**	0.07	**0.98**

https://arxiv.org/pdf/2002.10099.pdf

Neurall-Pull (ICML 2021) https://github.com/mabaorui/NeuralPull

- Neural network based implicit function fitting (similar as IGR)
- Different loss function d
  - Neural network $f(\boldsymbol{c}, \boldsymbol{q})$, trained over shape data set, learnable shape specific condition (latent code) $\boldsymbol{c}$
  - Query location $\boldsymbol{q}_i$, input point $\boldsymbol{t}_i$ closest to $\boldsymbol{t}_{i'}$

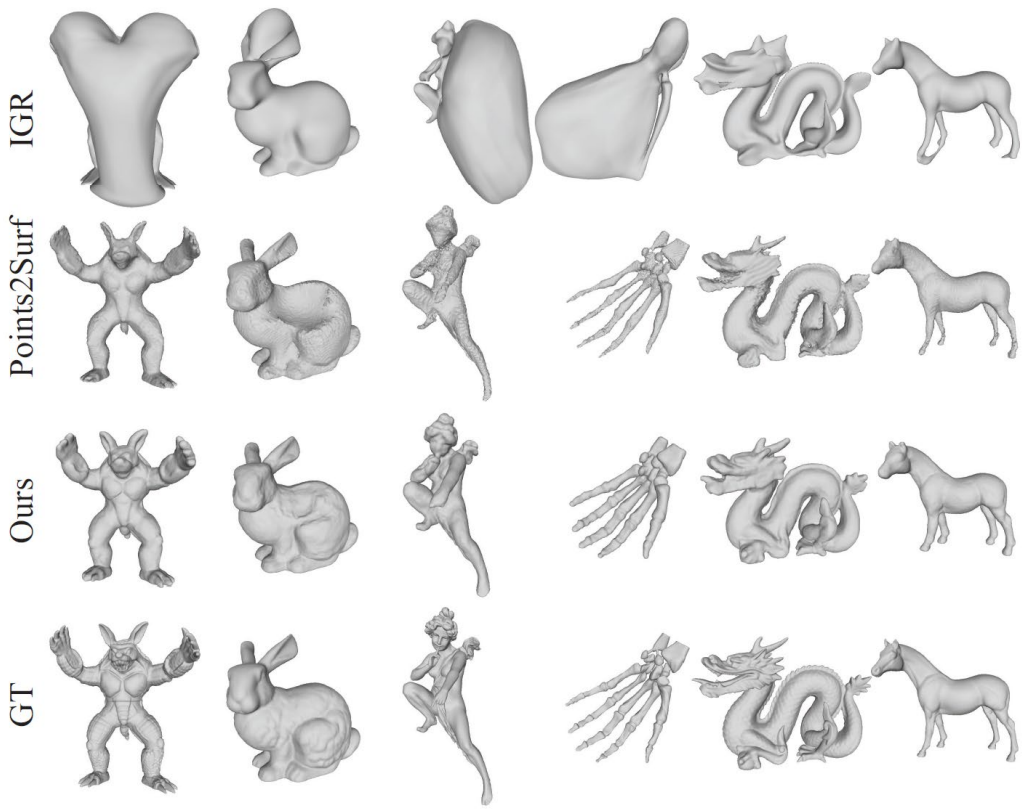
$$\boldsymbol{t}_i' = \boldsymbol{q}_i - \boldsymbol{f}(\boldsymbol{c}, \boldsymbol{q}_i) \times \nabla \boldsymbol{f}(\boldsymbol{c}, \boldsymbol{q}_i) / ||\nabla \boldsymbol{f}(\boldsymbol{c}, \boldsymbol{q}_i)||_2$$

$$d(\{\boldsymbol{t}_i'\}, \{\boldsymbol{t}_i\}) = \frac{1}{I} \sum_{i \in [1,I]} ||\boldsymbol{t}_i' - \boldsymbol{t}_i||_2^2$$

Pull query $\boldsymbol{q}_i$ towards surface using direction and distance given by neural network $\boldsymbol{f}$

Neurall-Pull

https://github.com/mabaorui/NeuralPull



Neural-pull

Table 1. Reconstruction comparison in terms of L2-CD (×100).

Dataset	DSDF	ATLAS	PSR	Points2Surf	IGR	Ours
ABC	8.41	4.69	2.49	1.80	0.51	**0.48**
FAMOUS	10.08	4.69	1.67	1.41	1.65	**0.22**
Mean	9.25	4.69	2.08	1.61	1.08	**0.35**

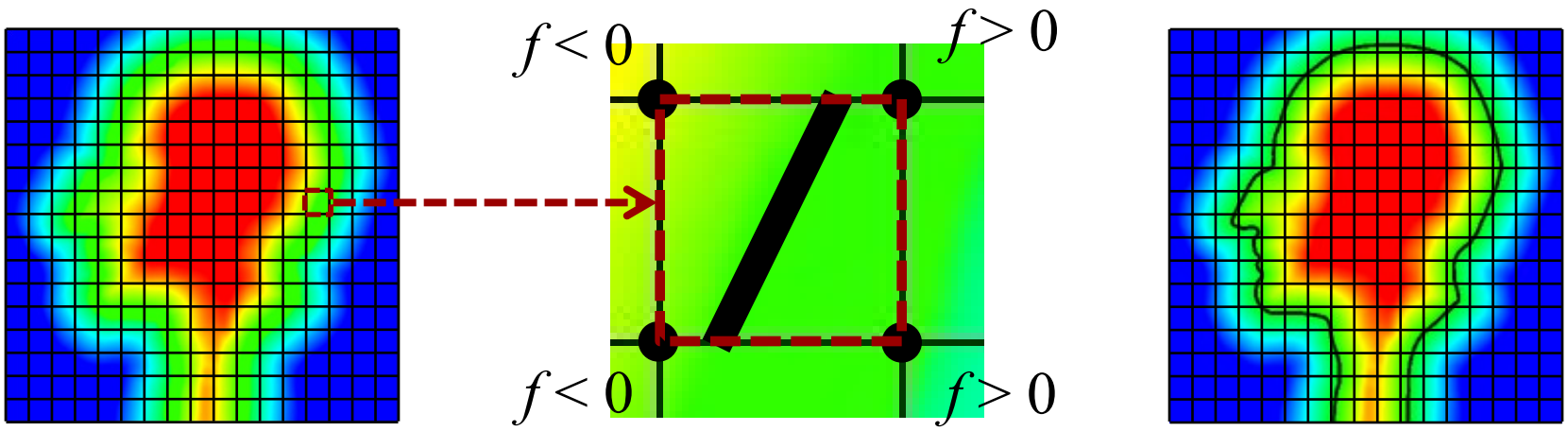
- CD: Chamfer distance
- DSDF, ATLAS, Points2Surf: neural network based
- PSR: Poisson surface reconstruction, piecewise polynomial implicit function

Triangulate isosurface

- Goal: construct triangle mesh that approximates implicite function
- Marching cubes approach
  http://en.wikipedia.org/wiki/Marching_cubes
  - Sample implicit function on 3D grid
  - Construct triangulation of each grid cell separately

Triangulate isosurface

- Given values of f at grid vertices
- Triangulate zero-isosurface $\{\mathbf{x} \mid f(\mathbf{x})=0\}$
- Idea: visit all cells, triangulate each cell with sign changes at its vertices separately

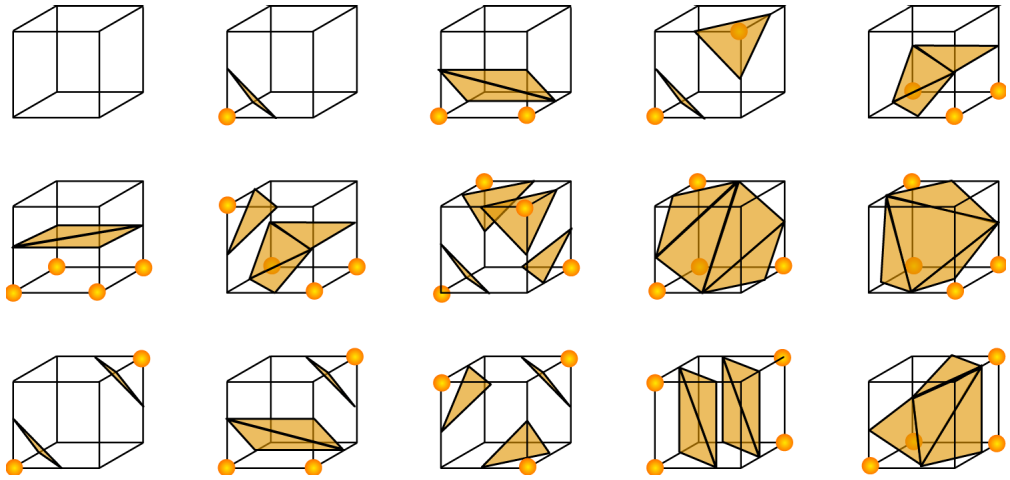


Triangulated cell

All cells triangulated

Isosurface extraction in 3D

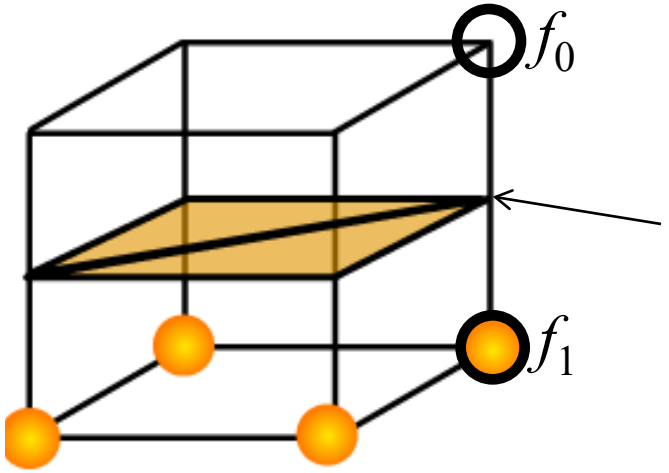
- Basic algorithm in 3D called marching cubes
  http://en.wikipedia.org/wiki/Marching_cubes
- $2^8 = 256$ cases of possible configurations for sign changes
- Can reduce to small number of cases using symmetries (rotation, reflection)



15 basic cases for sign change configurations in marching cubes

Marching cubes

- Encode each cell as 8-bit number given by signs at vertices
- Look-up triangulation in precomputed look-up table
- Position vertices along cube edges by linear interpolation



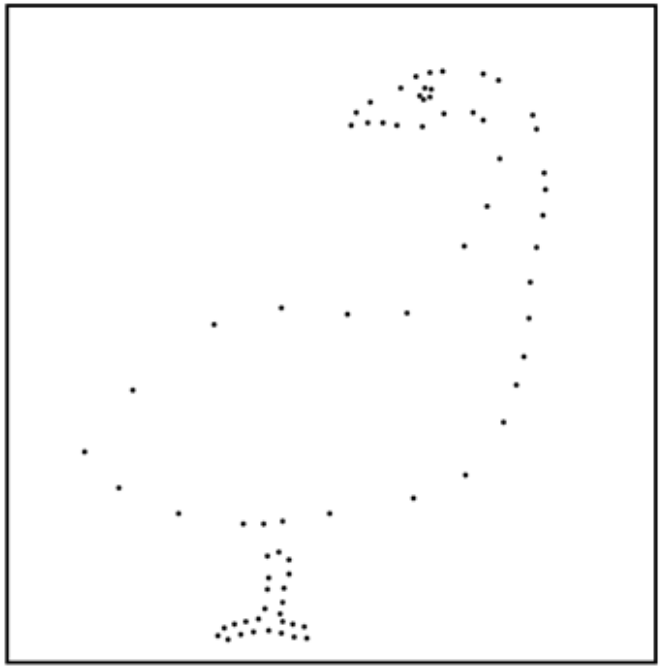
Position determined by linear interpolation of values f_0, f_1 at neighboring cube vertices

Two examples today

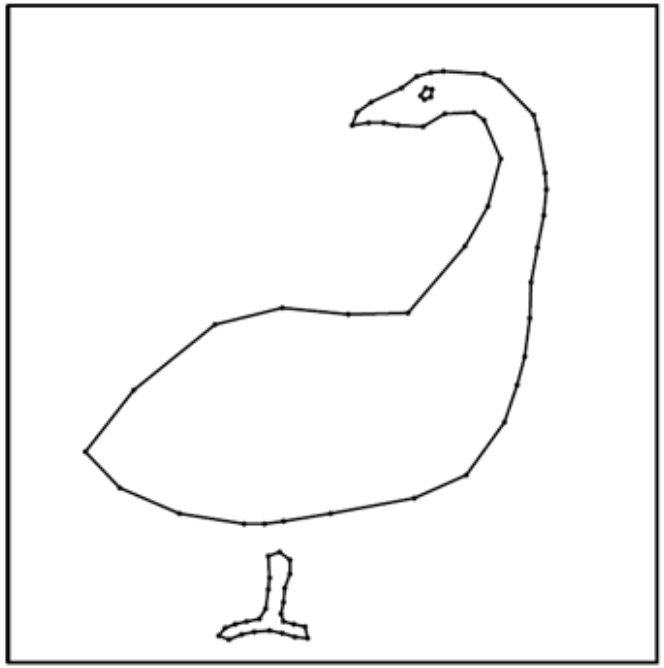
1. **Combinatorial**
   - Connect input data points to mesh
2. Function fitting
   - Implicit function in 3D

Combinatorial approaches

- Connect input points to form mesh
- Discrete number of possible connections, combinatorial problem
- Strategies?



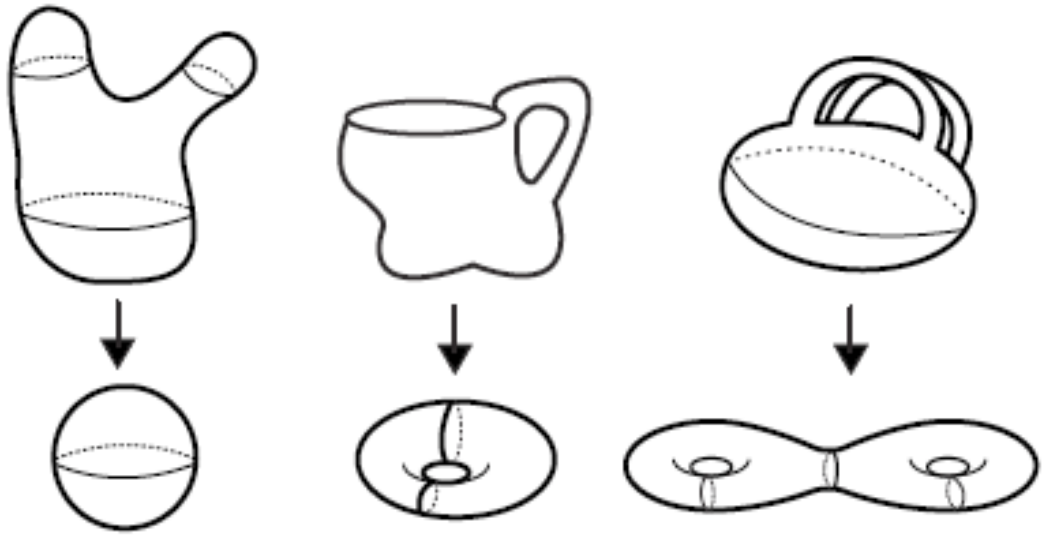
Input



Output

Goals

- Guarantee watertight mesh, without self intersections, gaps, cracks
- Guarantee same topology (genus) as (uknown) original shape
  http://en.wikipedia.org/wiki/Genus_(mathematics)
  - Same number of holes, handles



Topics

Background

- Voronoi diagram
- Delaunay triangulation
- Medial axis/surface
- Crust

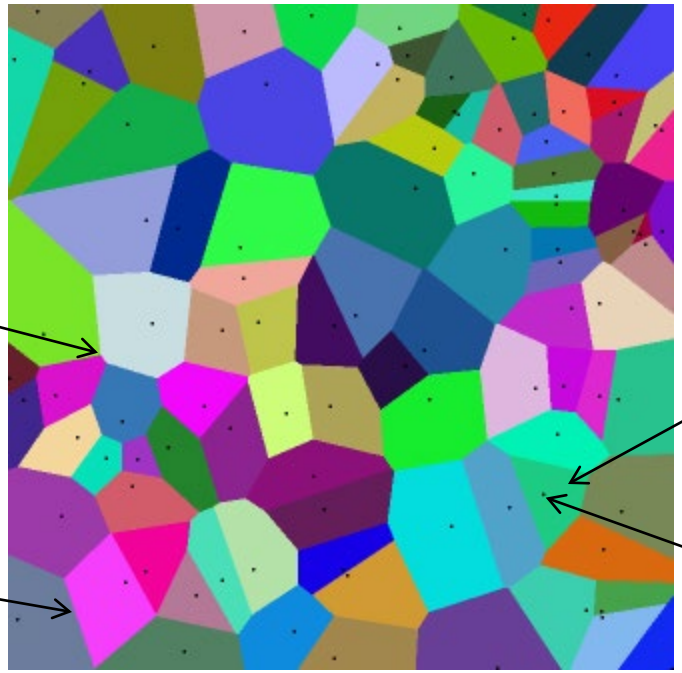
Reconstruction algorithms

- Curve reconstruction using crust
- Extension to surfaces in 3D

Voronoi diagram

http://en.wikipedia.org/wiki/Voronoi_diagram

- Partition of space into regions given by set of points
  - Partition: non overlapping, complete coverage
- Voronoi cell of each input point is area closest to that input point

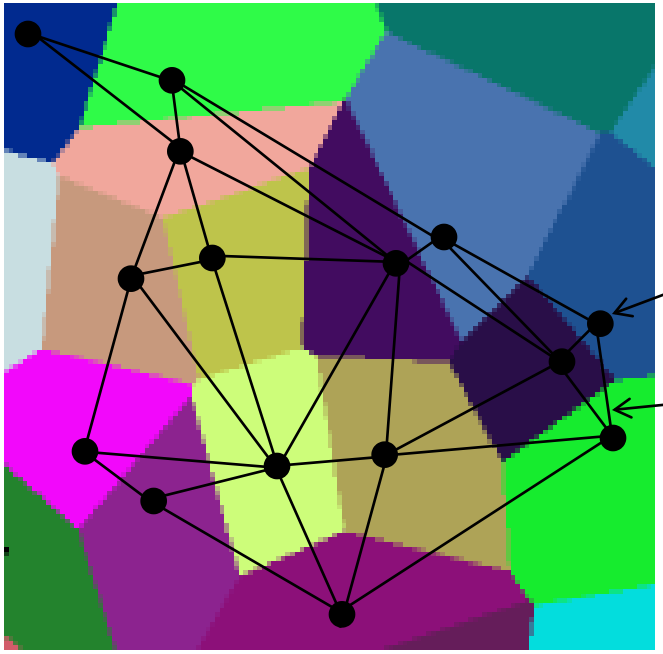


Voronoi diagram

Delaunay triangulation

http://en.wikipedia.org/wiki/Delaunay_triangulation

- Dual graph of Voronoi diagram
  http://en.wikipedia.org/wiki/Dual_graph
  - Delaunay vertex (input point) corresponding to each Voronoi cell
  - Delaunay edge joining two neighboring Voronoi cells

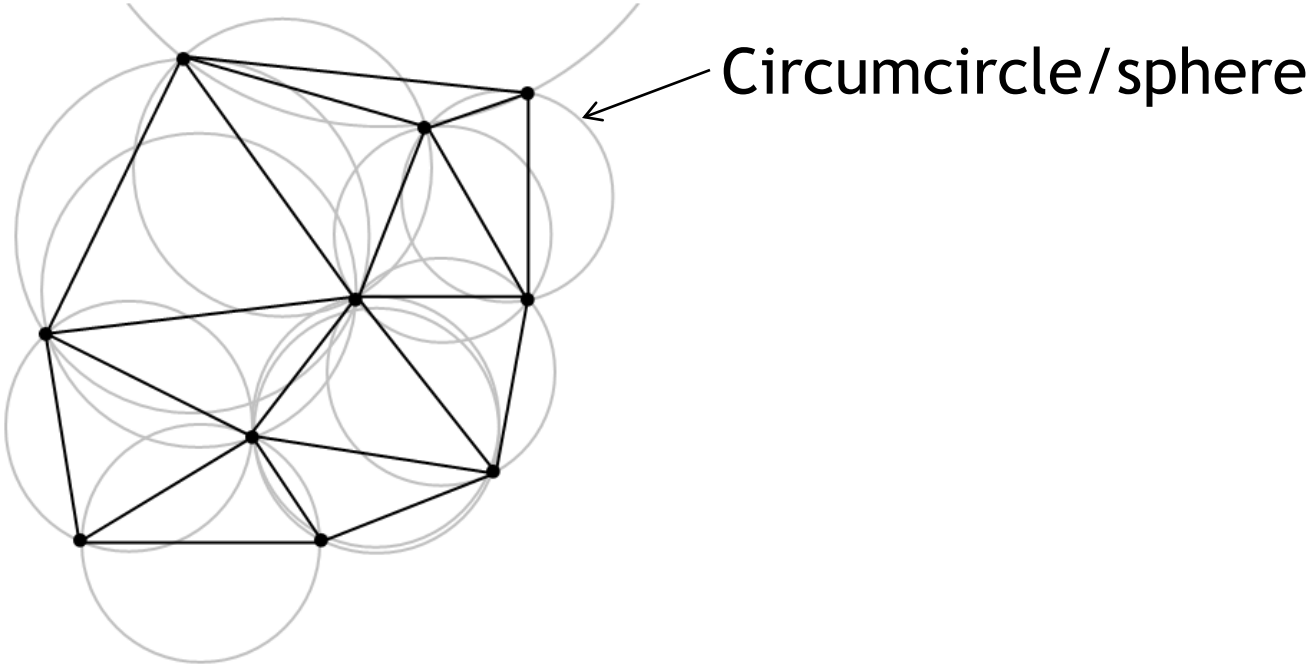


Delaunay triangulation

Delaunay triangulation

Properties

- No Delaunay vertex lies inside any circumsphere of any Delaunay triangle
- Maximizes minimum angle in all triangles

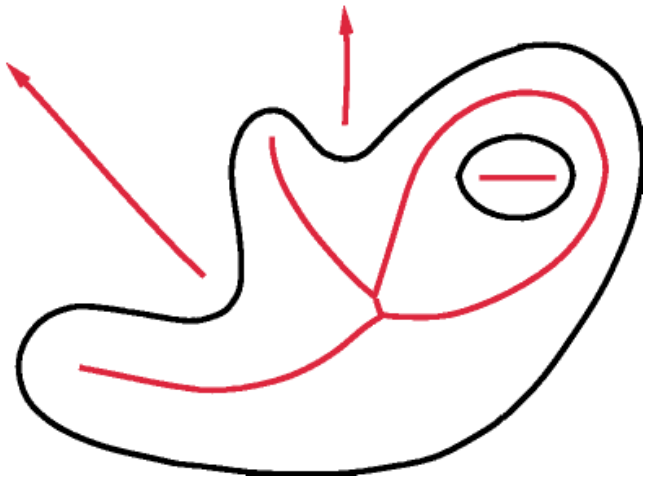


Algorithms for Delaunay triangulation

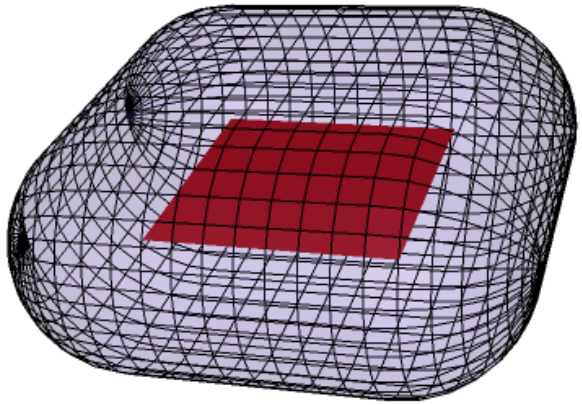
- Non-trivial, in particular in higher dimensions
- Separate research area, computational geometry
  https://en.wikipedia.org/wiki/Computational_geometry
- Standard implementations available
  http://www.qhull.org/

Medial axis/surface

- Set of points with more than one closest point on curve/surface
- Can interpret as extension of Voronoi diagram to continuous curves/surfaces
- For curves in 2D, Voronoi vertices (vertices of Voronoi diagram) of dense set of sample points along curve approximate medial axis
  - Unfortunately, doesn‘t hold in 3D



Medial axis



Medial surface

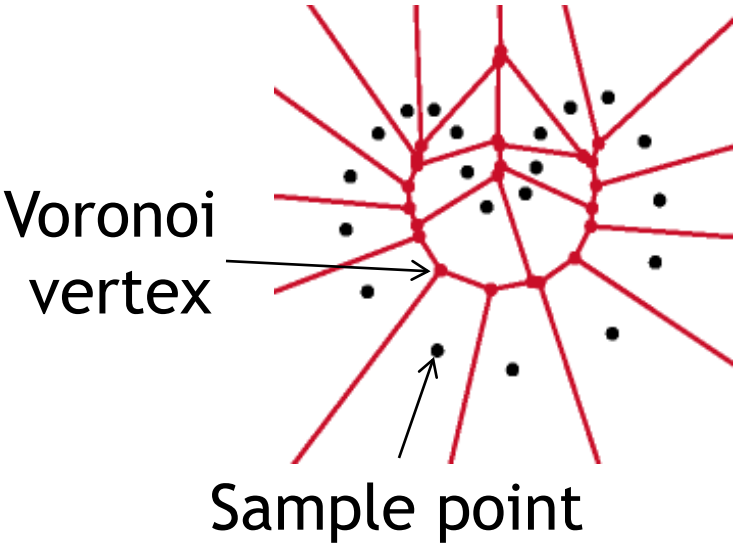
Crust

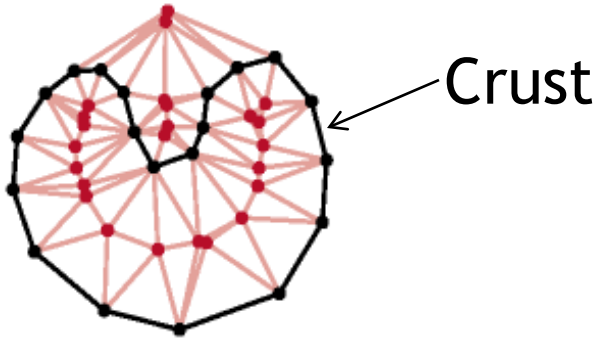
Definition of Crust

- A graph on the sample points
- Each pair of sample points potentially forms an edge
- An edge belongs to the crust if it has circumcircle empty of
  - other sample points
  - and all Voronoi vertices given by the sample points

Curve reconstruction using „crust“

1. Compute Voronoi diagram of sample points
2. Compute Delaunay triangulation of sample points **and** Voronoi vertices
3. Crust is all edges of this Delaunay triangulation that contain only sample points („Voronoi“ filtering)

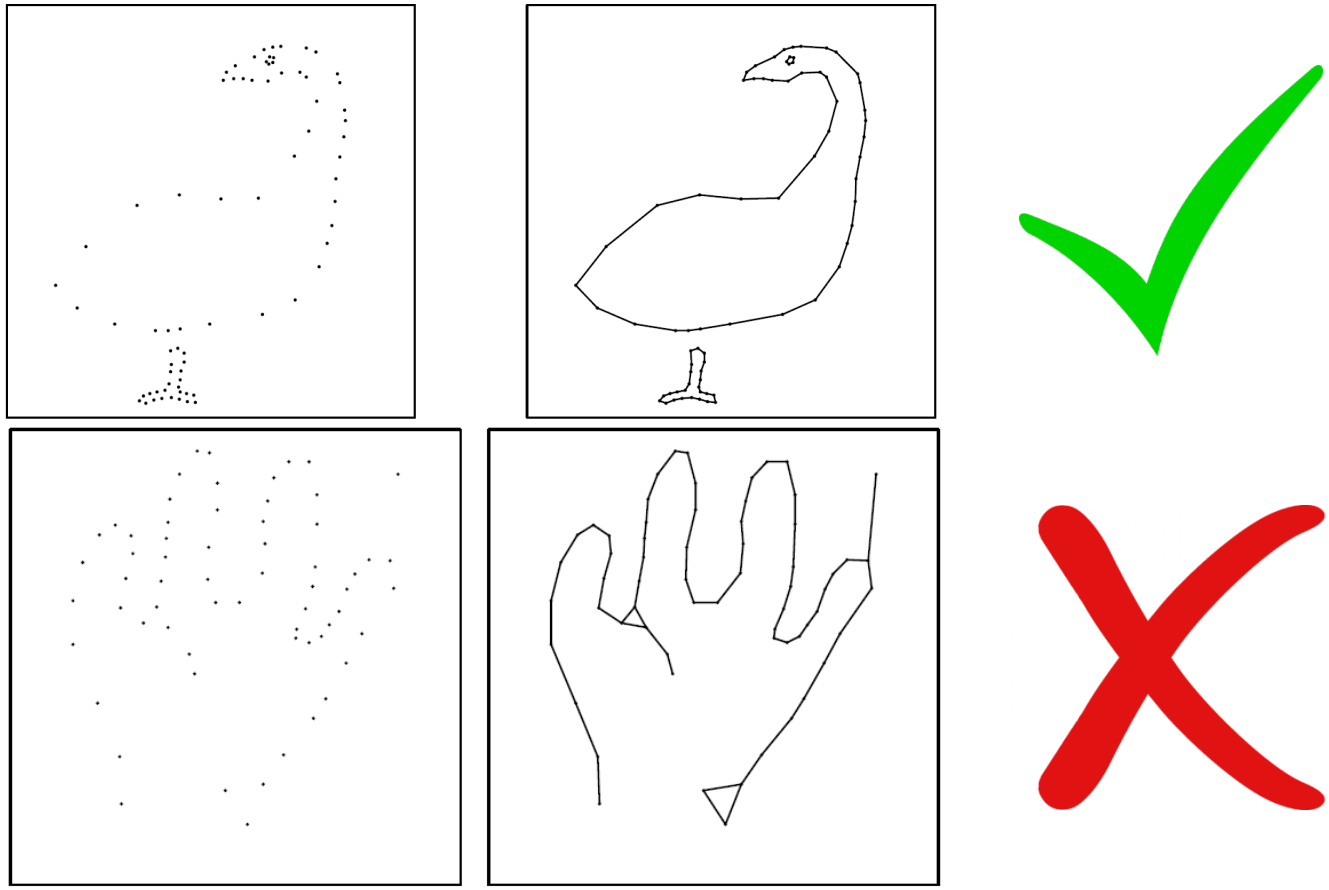




Delaunay triangulation of sample points and Voronoi vertices

Crust algorithm

- When does it work, when doesn‘t it?



- Intuitively, need „dense enough“ samples

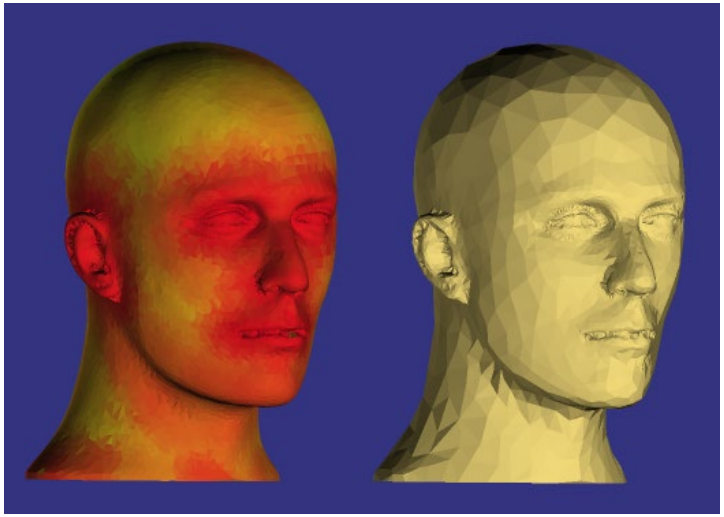
Sampling criterion

Feature size

- Distance from point on (continuous) surface to nearest point on medial axis

r-sample

- A set of points is r-sample of surface, if distance from any point on surface to nearest sample is at most r times feature size
- „If the medial axis is close to the surface, we need to sample more densely to satisfy the r-sampling criterion“
- Medial axis is close to surface if curvature is high, or in thin structures
- R-sampling criterion takes into account both curvature and proximity of surface parts („object thickness“)



Visualization of feature size

Sampling theorem

Theorem (Amenta et al.) http://www.cs.ucdavis.edu/~amenta/pubs/crust.pdf

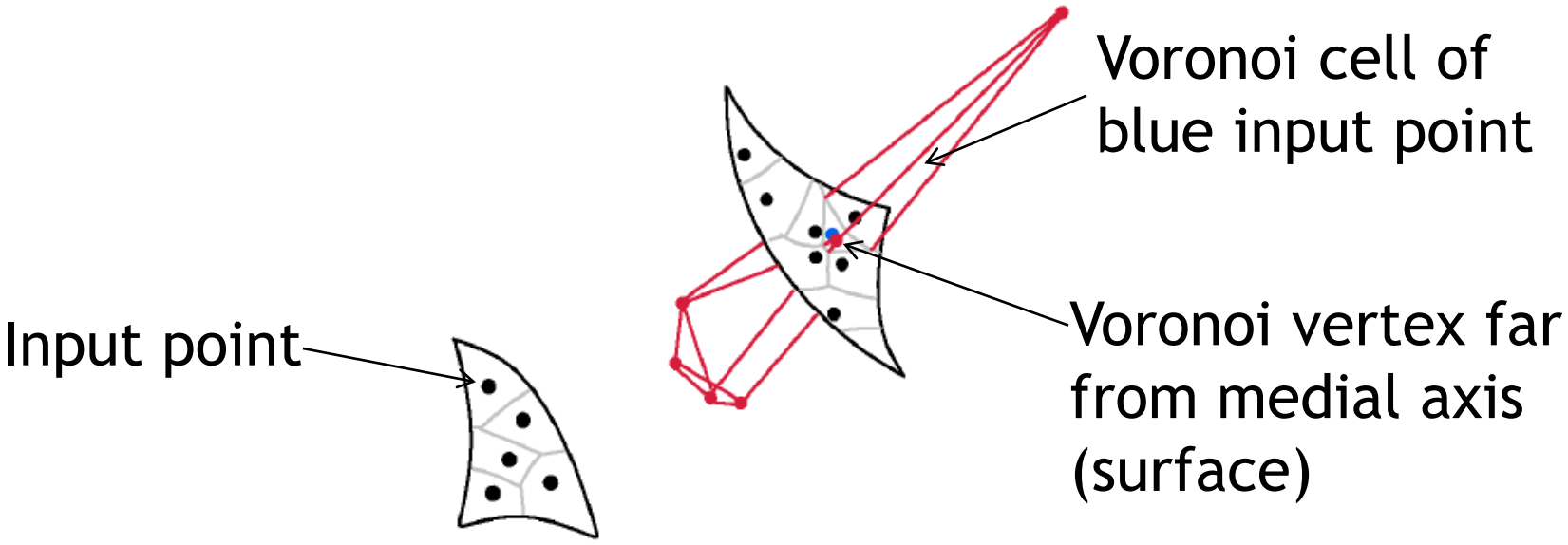
Crust of an r-sample from a smooth curve connects only adjacent points on the curve if $r <= 0.25$

„Curve reconstruction works correctly if distance from any point on original, continuous surface to nearest sample is at most $r=0.25$ times feature size“

Extension to 3D

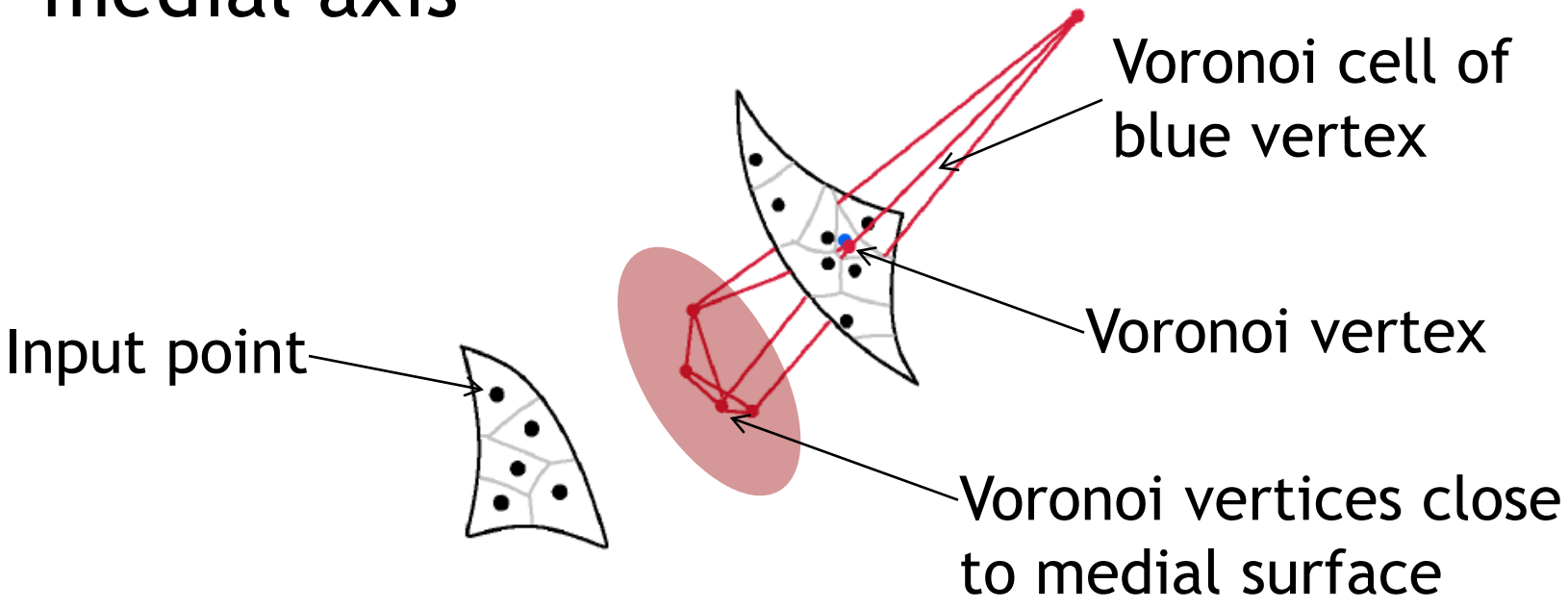
http://www.cs.ucdavis.edu/~amenta/pubs/crust.pdf

- In 3D, Voronoi vertices do not necessarily lie close to medial axis (which they do in 2D)
  - Even if surface is densely sampled
- Cannot use the exact same scheme as for curves



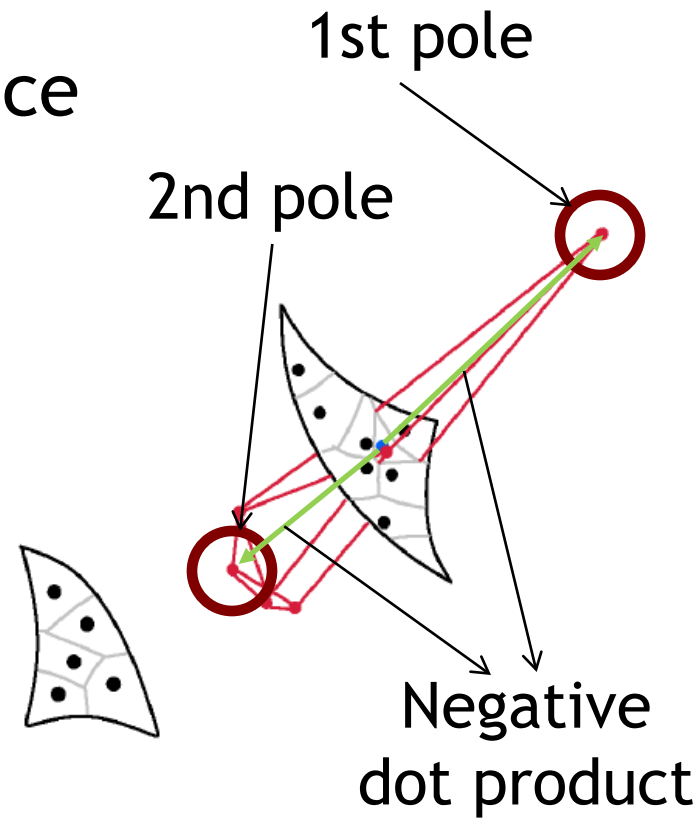
Observations

- Voronoi cell of blue sample is long and thin
- Roughly perpendicular to surface, extending out to medial surface
- Some Voronoi vertices do lie close to medial axis



Crust in 3D

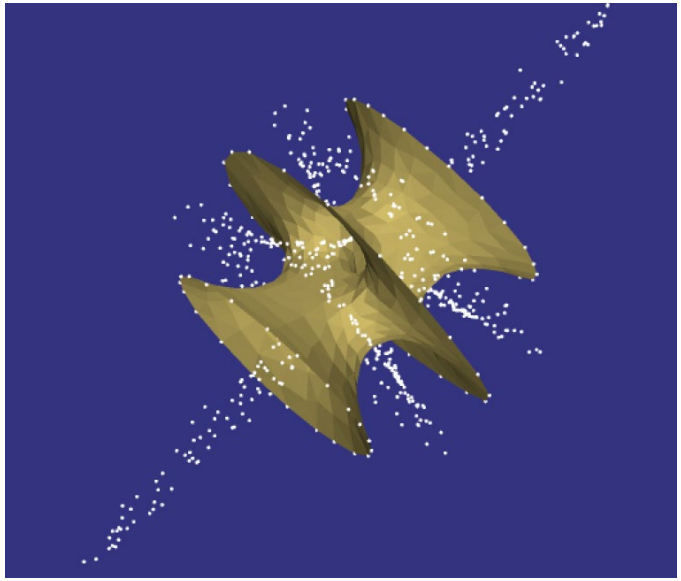
- Instead of using all Voronoi vertices in Voronoi filtering step, for each sample use only the two Voronoi vertices farthest from the sample, instead of all Voronoi vertices
- One on each side of the surface
- Called „poles“
- First pole: farthest vertex
- Second pole: farthest vertex that forms negative dot product with first pole



Crust in 3D

http://www.cs.ucdavis.edu/~amenta/pubs/crust.pdf

1. *Compute the Voronoi diagram of the sample points S*
2. *For each sample point s do:* Voronoi filtering
   (a) *If s does not lie on the convex hull, let p^+ be the farthest Voronoi vertex of V_s from s. Let n^+ be the vector $sp+$.*
   (b) *If s lies on the convex hull, let n^+ be the average of the outer normals of the adjacent triangles.*
   (c) *Let p^- be the Voronoi vertex of V_s with negative projection on n^+ that is farthest from s.*
3. *Let P be the set of all poles p^+ and p^-. Compute the Delaunay triangulation of $S \cup P$.*
4. *Keep only those triangles for which all three vertices are sample points in S.*



s: sample point
V_s: voronoi cell of s
p^+,p^-: poles
n^+,n^- : directions from s to poles

Crust in 3D

http://www.cs.ucdavis.edu/~amenta/pubs/crust.pdf

Theorem (Amenta et al.)

Given an r-sample of a surface with $r<0.06$. Then:

- The crust of the sample **contains** a set of triangles forming a mesh topologically equivalent to the surface
- Every point on the crust lies within distance $5r*d(p)$ of some point p on the surface, where $d(p)$ is the distance of p to the medial axis

Proof: see paper

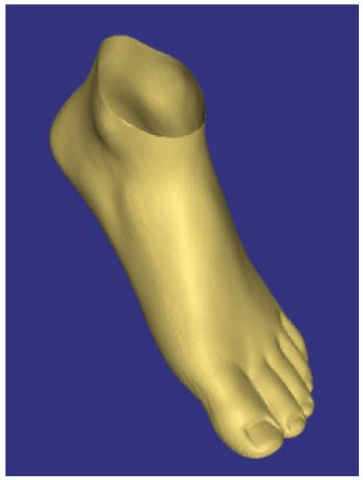
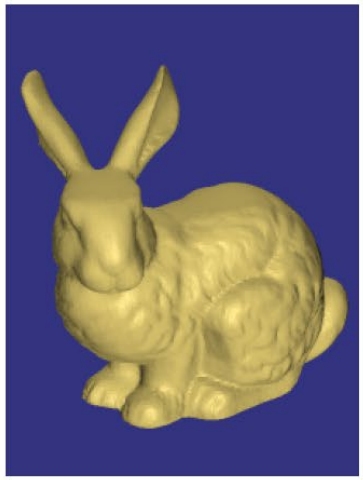
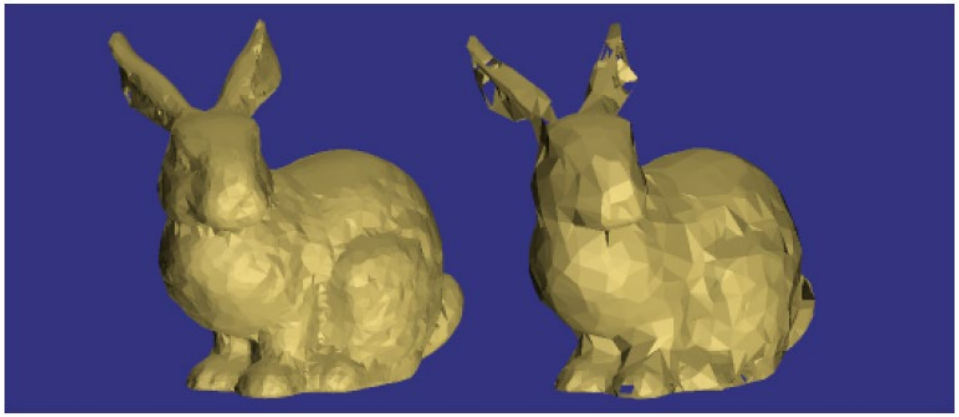
Normal estimation and filtering

- Crust contains many small sliver triangles that are
  - Perpendicular to surface
  - Form T-junctions
- Additional filtering (removal of triangles) required
  - Based on fact that vectors to poles are nearly perpendicular to surface
- Throw away triangles whose normal differs too much from estimated normals
  - Can prove that the remaining triangles still form a subset forming a mesh topologically equivalent to the surface

Examples

http://www.cs.ucdavis.edu/~amenta/pubs/crust.pdf



Summary

- Combinatorial approach using Voronoi diagram, Delaunay triangulation, Crust
- Advantages
  - Guarantees correct topology under certain sampling criterion (**feature size, r-sample**)
- Disadvantages
  - Interpolating input points undesireable if input points are noisy