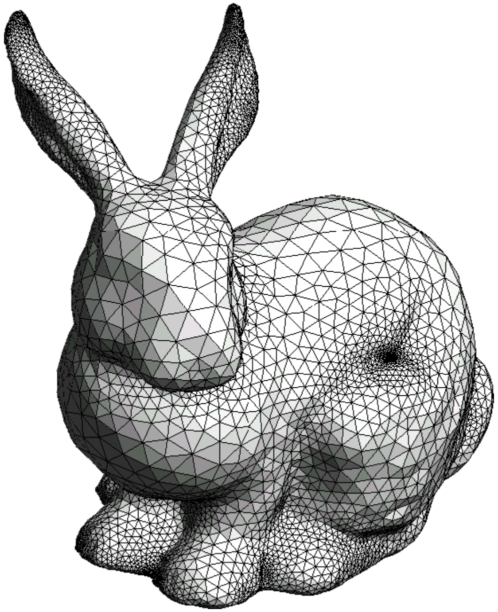


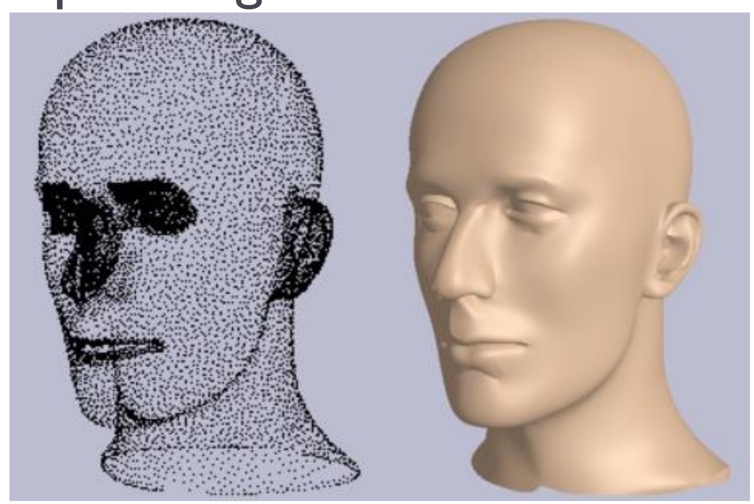
Geometric Modeling in Graphics

Part 10: Surface reconstruction

Martin Samuelčík
www.sccg.sk/~samuelcik
samuelcik@sccg.sk

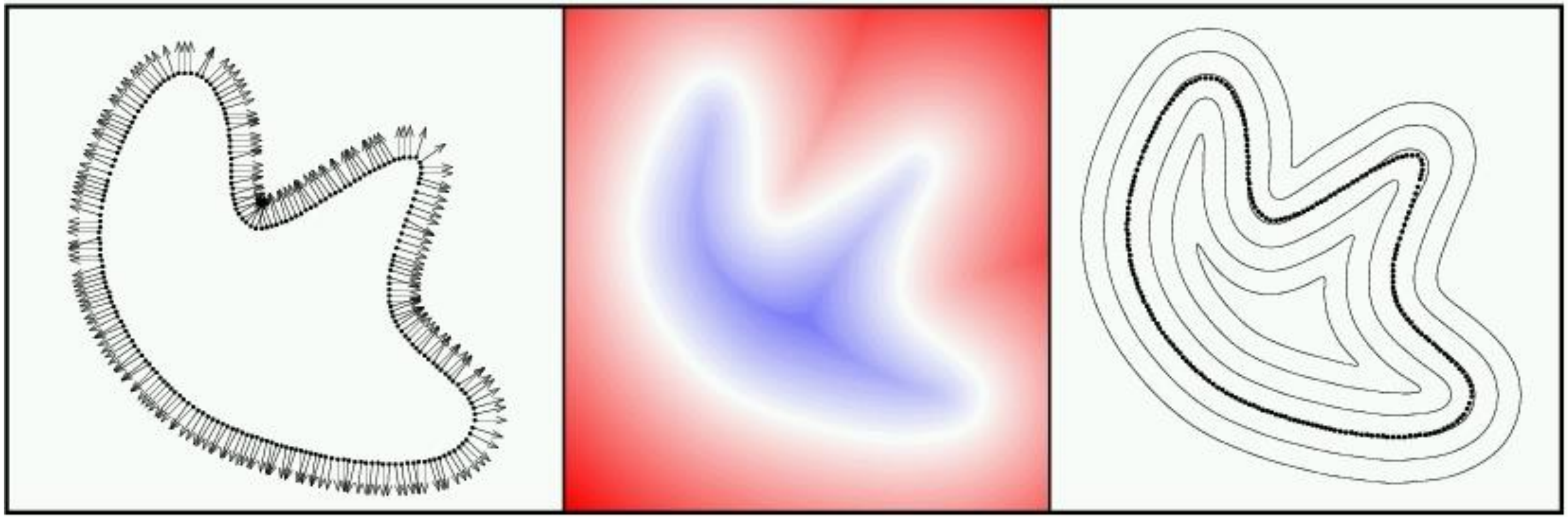
Curve, surface reconstruction

- Finding compact connected orientable 2-manifold surface possibly with boundary or closed, that is partially given by set of geometric elements
- Input elements: points, curves, part of surface
- Output: curve or surface
- Representation of reconstructed object
  - Zero level of implicit function sampled in grid
  - Parametric surface
  - Polygonal mesh



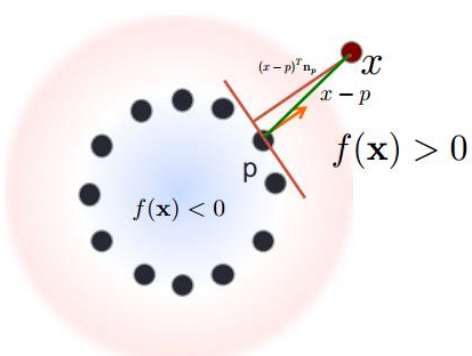
Implicit reconstruction

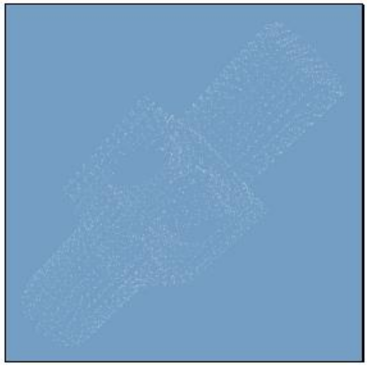
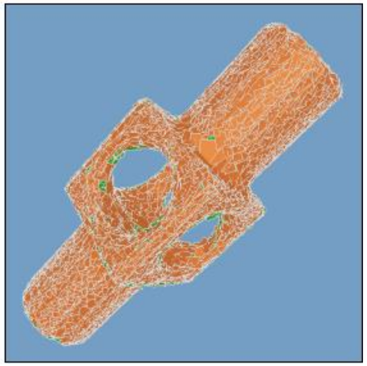
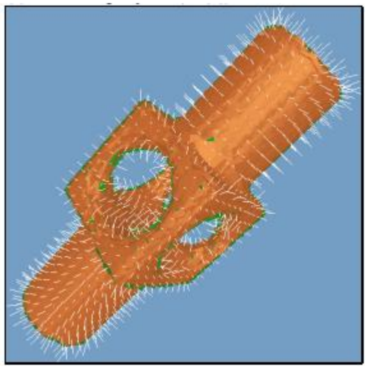
- Input: point cloud
- Conversion of unstructured to structured data
- Ill-posed (difficult) problem
- Output uniformly sampled implicit function



Implicit reconstruction

- Hoppe et al.
  - http://research.microsoft.com/en-us/um/people/hoppe/recon.pdf
- Needed properly oriented normals
- For sample point X, find its nearest point P in point cloud
- Compute $f(X)$ - signed distance of X and tangent plane in P
  - Tangent plane is given by P and normal in P
  - If projection of X on tangent plane is far away from points of point cloud, $f(X)$ is undefined and used for hole detection



Implicit reconstruction

- Weighted Least Squares
- Reconstruction and smoothing in one process
- For sample point $\mathbf{x}$, compute nearest point $\mathbf{a}(\mathbf{x})$ and normal $\mathbf{n}(\mathbf{x})$ on surface as weighted combination of near points of point cloud
- Computation of normal
  - Minimizing $\sum_{i=1}^{N} \left(\mathbf{n}(\mathbf{x}) \cdot (\mathbf{a}(\mathbf{x}) - \mathbf{p}_i)\right)^2 \theta(\|\mathbf{x} - \mathbf{p}_i\|)$
  - Eigenvector assigned to smallest eigenvalue of covariance matrix

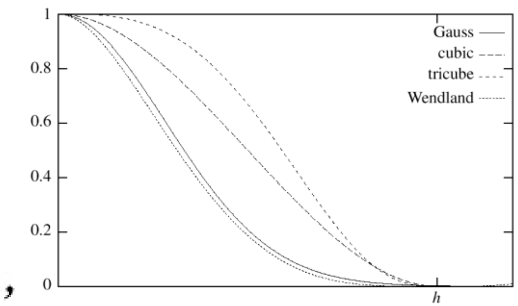
$$b_{ij} = \sum_{k=1}^{N} \theta(\|\mathbf{x} - \mathbf{p}_k\|)(p_{k_i} - a(\mathbf{x})_i)(p_{k_j} - a(\mathbf{x})_j).$$

- Computation of surface point $\mathbf{a}(\mathbf{x}) = \dfrac{\sum_{i=1}^{N} \theta(\|\mathbf{x} - \mathbf{p}_i\|)\mathbf{p}_i}{\sum_{i=1}^{N} \theta(\|\mathbf{x} - \mathbf{p}_i\|)}.$
- Computation of implicit function

$$f(\mathbf{x}) = \mathbf{n}(\mathbf{x}) \cdot (\mathbf{a}(\mathbf{x}) - \mathbf{x}),$$

Implicit function reconstruction

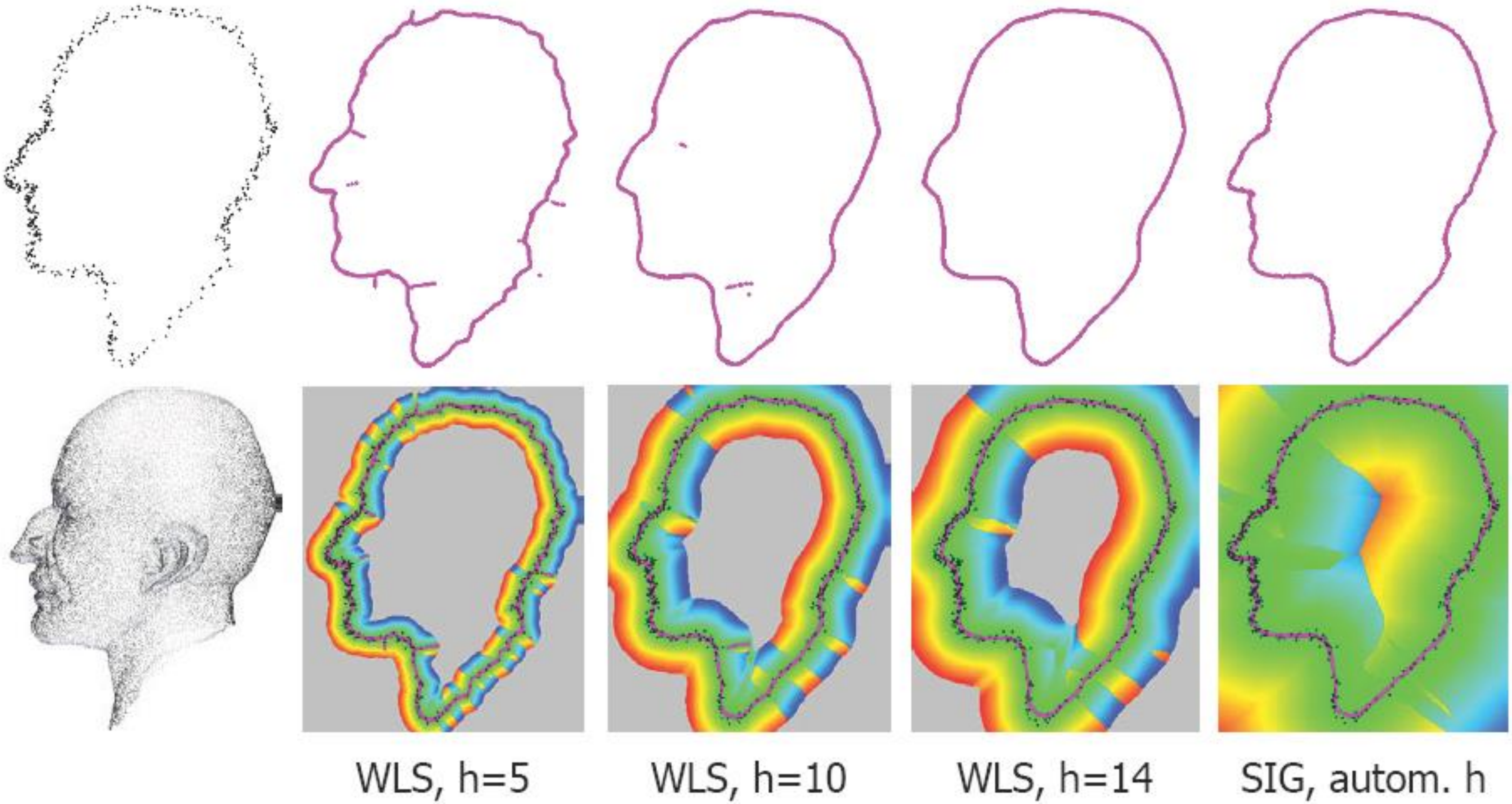
- Weighted, Moving Least Squares
- Weighted functions
  - Gaussian
    $$\theta(d) = e^{-d^2/h^2}, \quad d = \|\mathbf{x} - \mathbf{p}\|,$$
  - Cubic
    $$\theta(d) = 2\left(\frac{d}{h}\right)^3 - 3\left(\frac{d}{h}\right)^2 + 1,$$
  - Tricubic
    $$\theta(d) = 2\left(\frac{d}{h}\right)^3 - 3\left(\frac{d}{h}\right)^2 + 1,$$
  - Wendland
    $$\theta(d) = \left(1 - \frac{d}{h}\right)^4 \left(4\frac{d}{h} + 1\right),$$



- Adaptive computation of radius of influence h
- Reconstructed surface is isosurface for isovalue 0

Implicit function reconstruction

- Weighted, Moving Least Squares



Implicit function reconstruction

- Poisson reconstruction
  - http://research.microsoft.com/en-us/um/people/hoppe/poissonrecon.pdf
- Input: point cloud with oriented normals
- Computing indicator implicit function (1-insize, 0-outside)
- Normals at points should be as close as possible to gradients of indicator function at points
- Poisson problem: Laplacian of indicator function equals to divergence of normals vector field
- Global optimization
- Creates very smooth surfaces that robustly approximate noisy data

Implicit reconstruction

- Poisson reconstruction
- Constructing octree over input points
  - The depth of octree controls precision of reconstruction
- Computing indicator sample value for each node of octree
- Solving large linear system
  - Matrix size = number of octree nodes
  - Sparse and symmetric matrix
- Usage of smoothing functions
- Isovalue for surface extraction – average of indicator function values at points

Implicit reconstruction

- Poisson reconstruction

Tree Depth	Time	Peak Memory	# of Tris.
7	6	19	21,000
8	26	75	90,244
9	126	155	374,868
10	633	699	1,516,806

Table 1: *The running time (in seconds), the peak memory usage (in megabytes), and the number of triangles in the reconstructed model for the different depth reconstructions of the dragon model. A kernel depth of 6 was used for density estimation.*

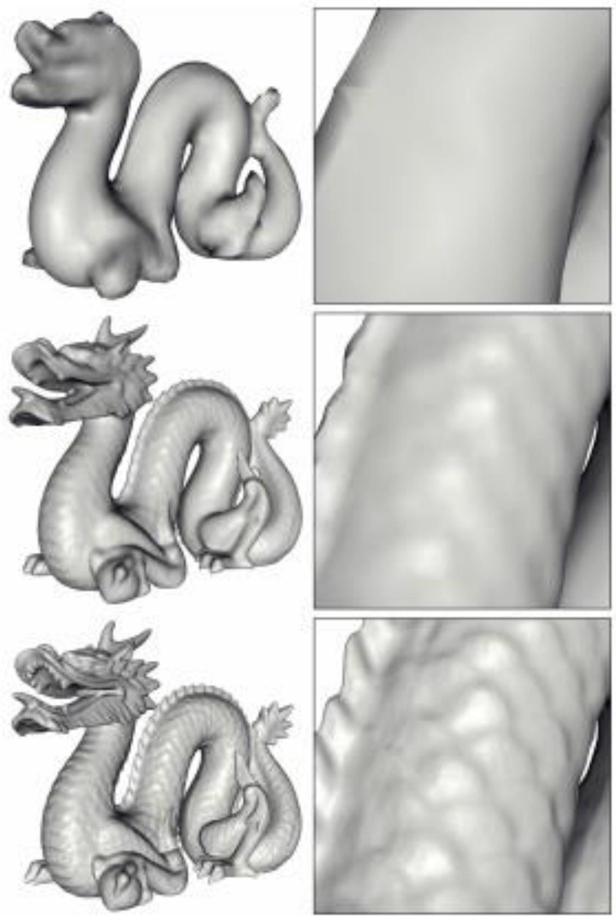


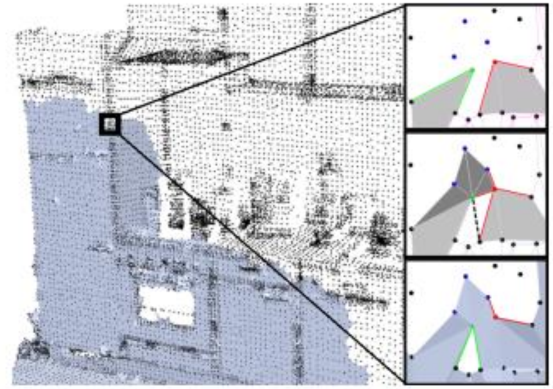
Figure 3: *Reconstructions of the dragon model at octree depths 6 (top), 8 (middle), and 10 (bottom).*

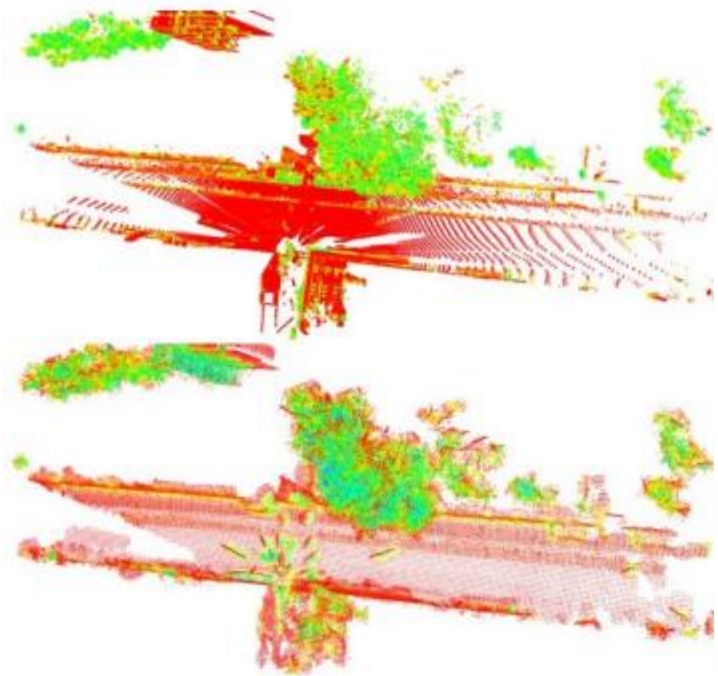
Advancing mesh reconstruction

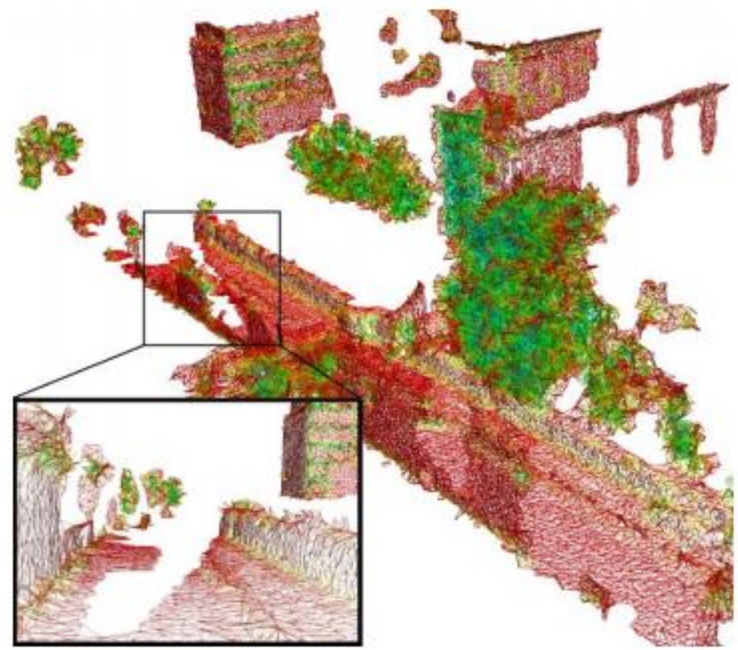
- Marton et al.
  - https://ias.informatik.tu-muenchen.de/_media/spezial/bib/marton09icra.pdf
- Greedy algorithm that directly creates triangle mesh
- Propagation of triangulation from starting point over all points of point cloud – advancing boundary fronts
- Computation of new triangles for points (fringe points) on a boundary of current triangulation
  - Compute normal for fringe point P using WLS
  - Find points near P and project them
  - Project triangles back and add them to triangulation
  - Do local pruning and smoothing of new triangle vertices
- Handle cases when two fronts meet

Advancing mesh reconstruction

- Marton et al.
- Implemented in PCL

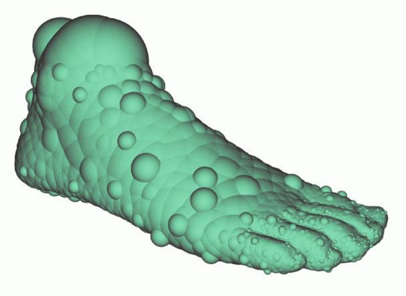




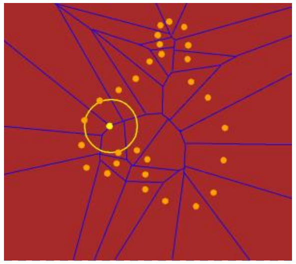


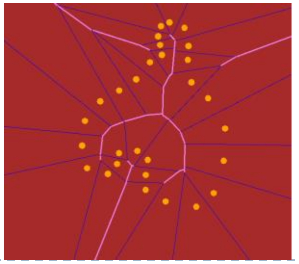
Power crust reconstruction

- Power crust algorithm
  - http://web.cs.ucdavis.edu/~amenta/pubs/sm.pdf
- Representing solid as MAT(medial axis transformation)
  - Union of balls contained in the interior
  - Centers of balls – medial axis



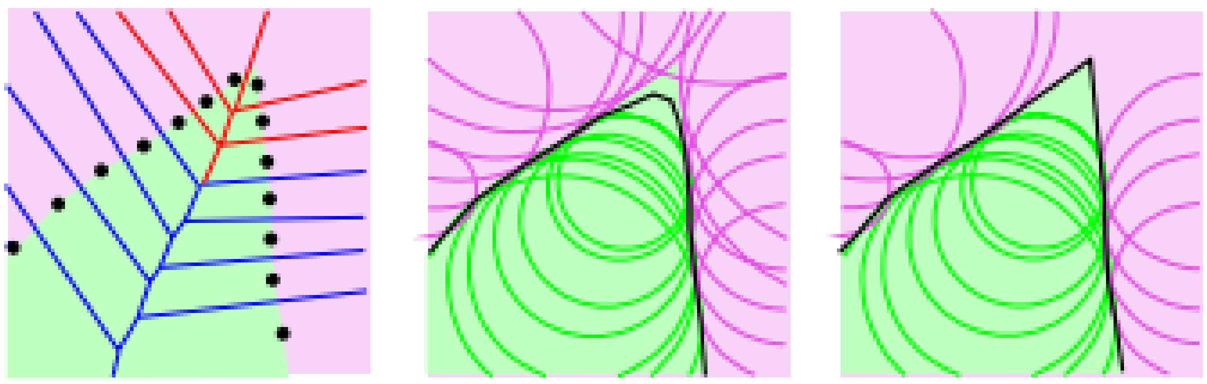
- Approximating MAT from point cloud using Voronoi diagram
  - Using subset of Voronoi vertices called poles – farthest vertices in Voronoi cell



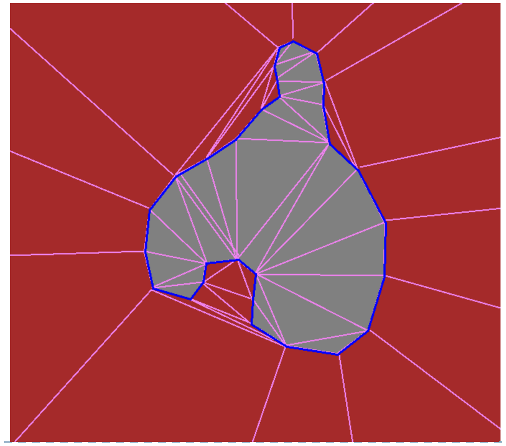


Power crust mesh reconstruction

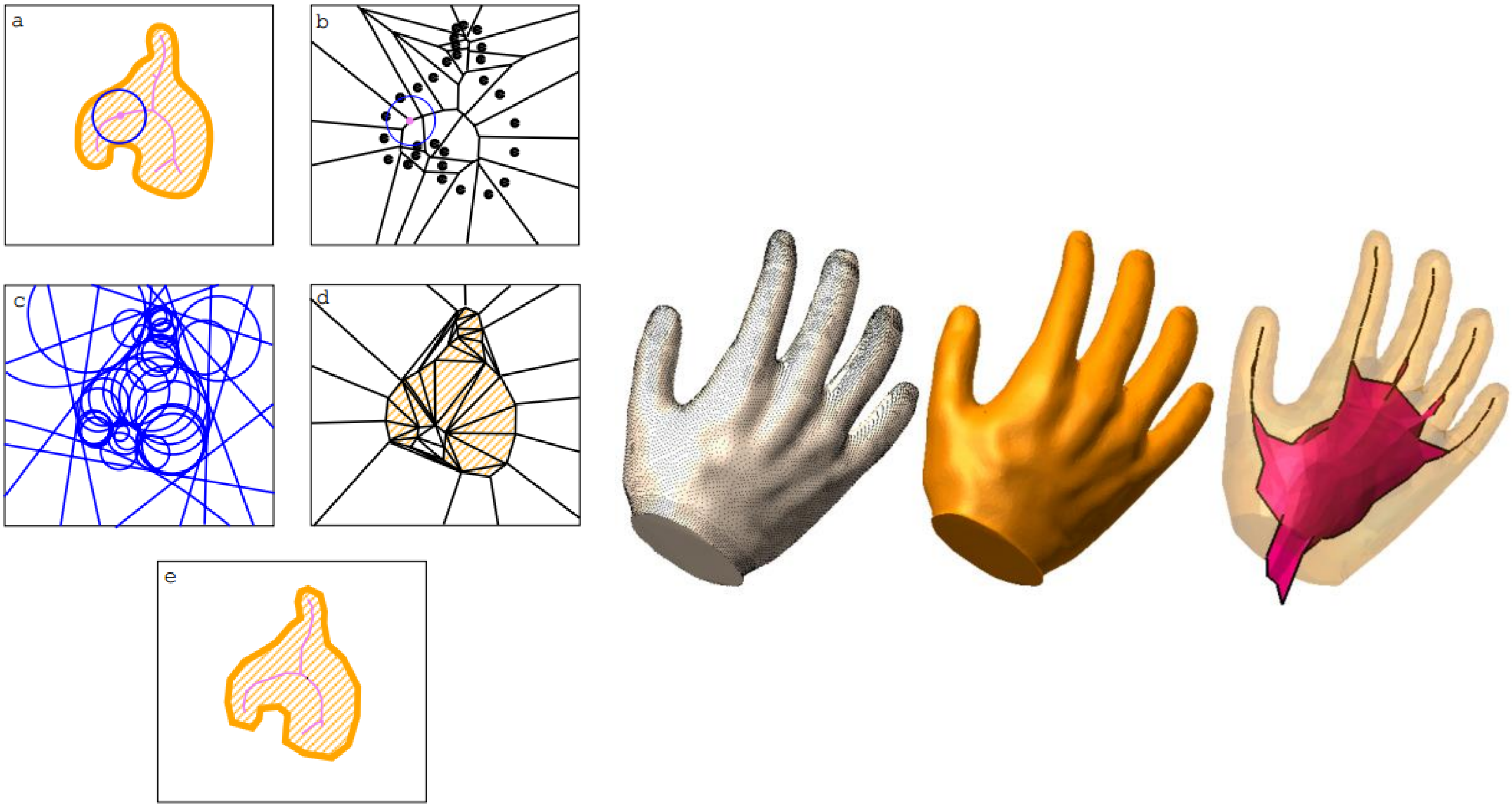
- Construct Voronoi ball for each pole such that it contains only points from neighbor Voronoi cells



- Generate triangles between interior and exterior Voronoi balls



Power crust mesh reconstruction



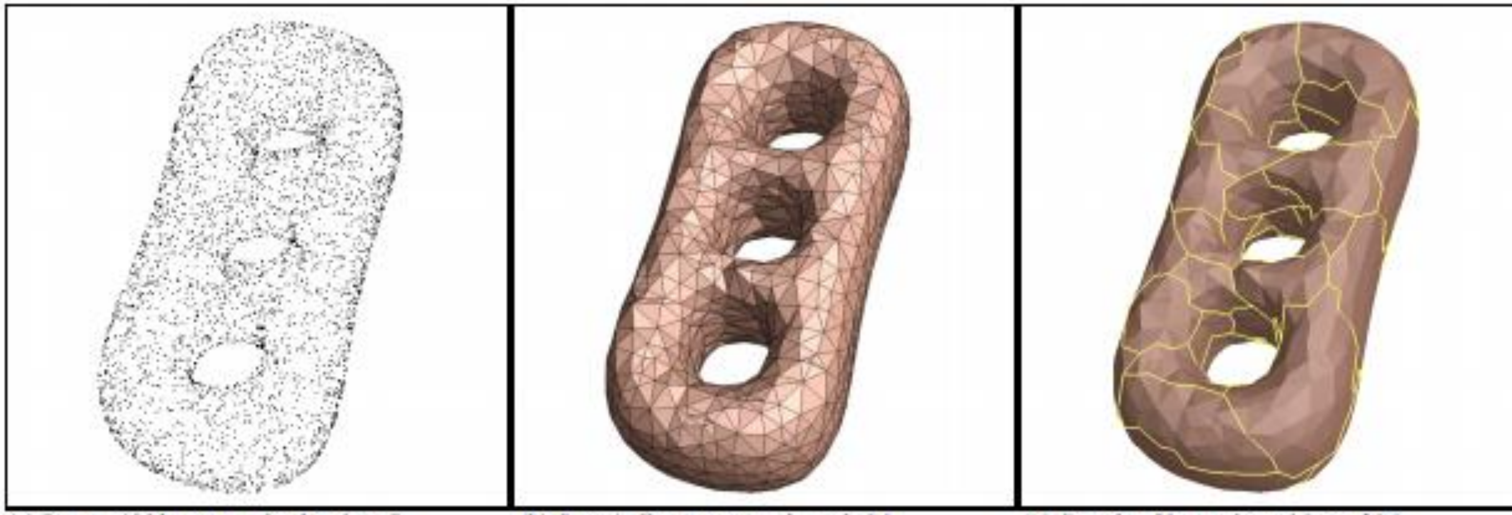
Parameteric reconstruction

- Hoppe et al.
  - http://research.microsoft.com/en-us/um/people/hoppe/bspline.pdf
- Reconstructing surface as B-spline patch network of arbitrary topology
- 1. Create dense approximating mesh M0 from input point cloud
- 2. Construct Voronoi partition of M0 forming triangular base complex
- 3. Refine triangular base complex to quadrilateral base complex
- 4. Compute parameterization of over each quad of quadrilateral base complex

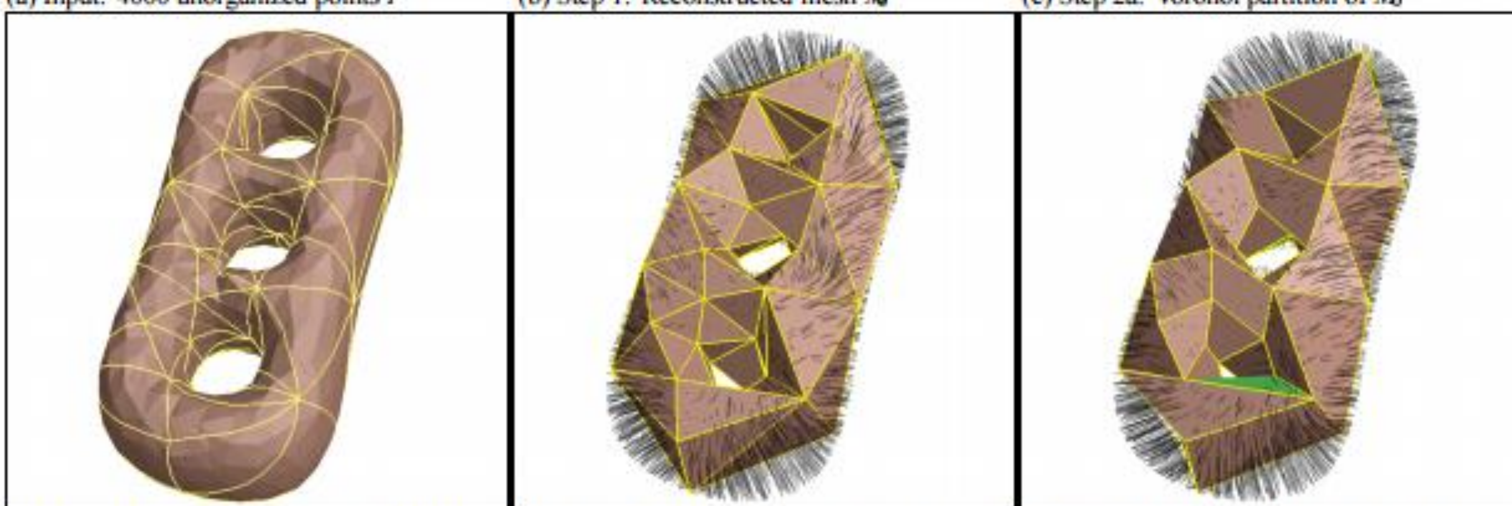
Parameteric reconstruction

- Hoppe et al.
- 5. Fit B-spline patch over each parametrized quad
  - Find points from point cloud that are parametrized by current quad and computing parameter values for each point
  - Iterative fitting that minimizes distance of points to B-spline patch
  - Adding fairness term for controlling patch wiggles making patch more planar
  - Ensuring G1 connectivity
- 6. Adaptive refinement
  - Quadrilateral base complex
  - Patch control points

Parameteric reconstruction

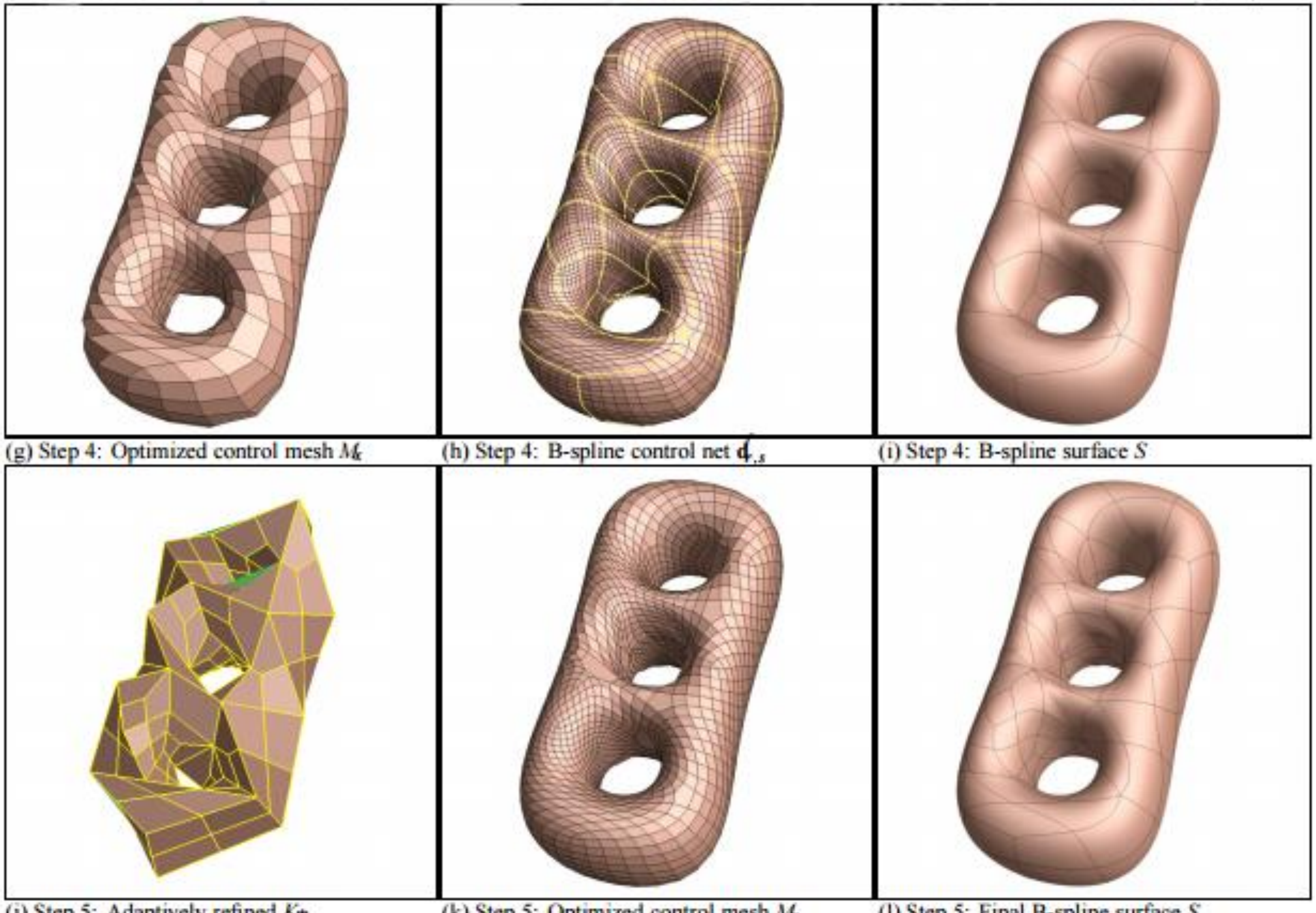


(a) Input: 4000 unorganized points P (b) Step 1: Reconstructed mesh M_0 (c) Step 2a: Voronoi partition of M_0



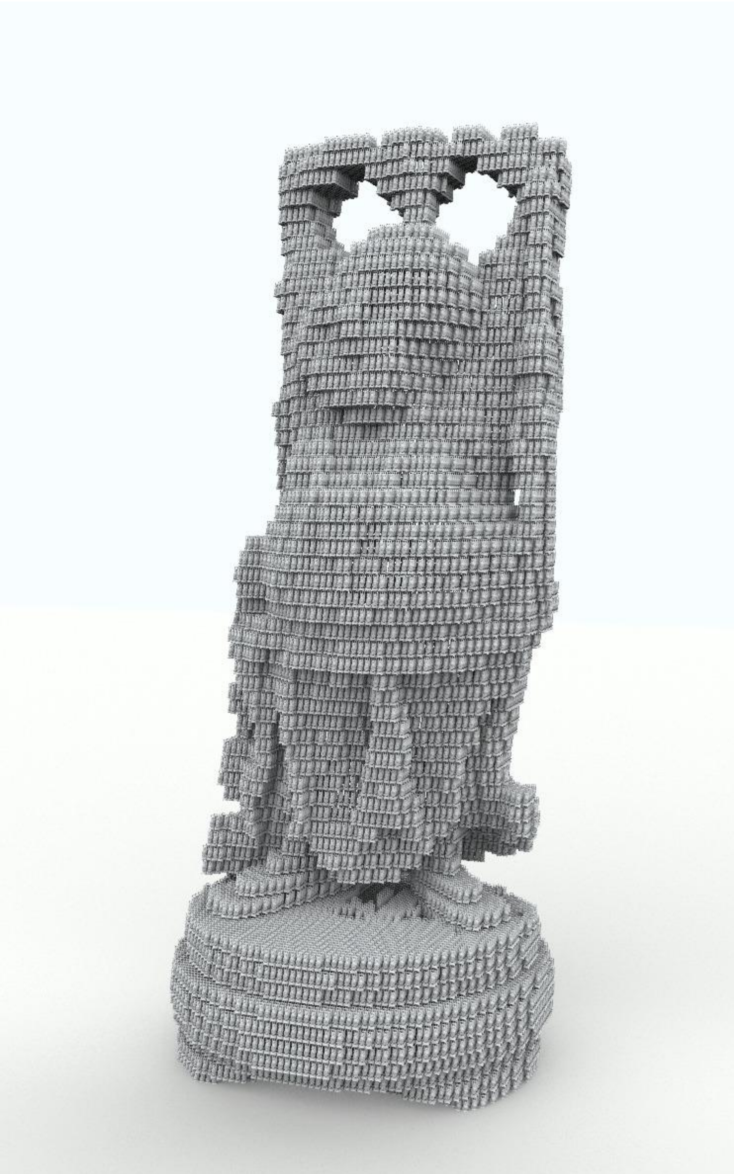
(d) Step 2b: Delaunay triangulation of M_0 (e) Step 2c: Triangular base complex $K_\triangle$ (f) Step 3: Quadrilateral base complex $K_\square$

Parameteric reconstruction



(g) Step 4: Optimized control mesh M_k (h) Step 4: B-spline control net $\mathbf{d}_{\cdot,s}$ (i) Step 4: B-spline surface S

(j) Step 5: Adaptively refined K_m (k) Step 5: Optimized control mesh M_k (l) Step 5: Final B-spline surface S



The End
for today