

From Point Clouds to tessellated surfaces

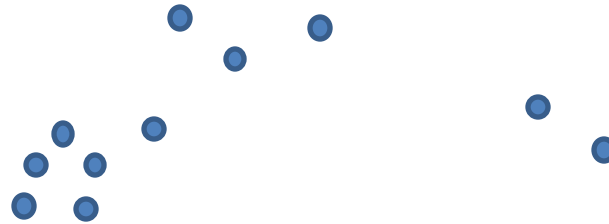


Paolo Cignoni,
Istituto di Scienza e Tecnologie dell'Informazione, Consiglio
Nazionale delle Ricerche



Problem Statement

Given a Point cloud $P = \{p_0, \dots, p_n\}, p_i \in \mathbb{R}^3$, find the mesh M that it *represents*



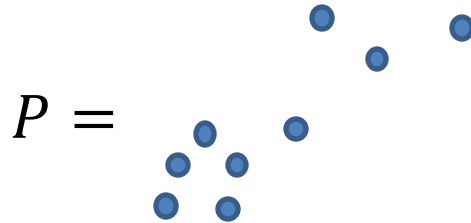
- Q1: It is a very ill posed problem, what does *represents* means?
- Q2: why do we care about this problem?

Motivations

- A1: Ideally, we want to find the surface which sampling produced the input problem
 - A2: Every device or methods produces a discrete puntual sampling of the surface
 - **Laser scanning**
 - **Image based techniques**
 - **Computerized Axial Tomography / simulation data**
- ... So that is what we are dealing with

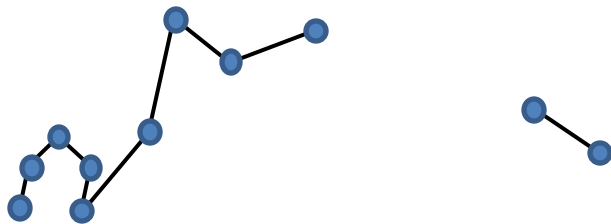


Explicit and Implicit Methods



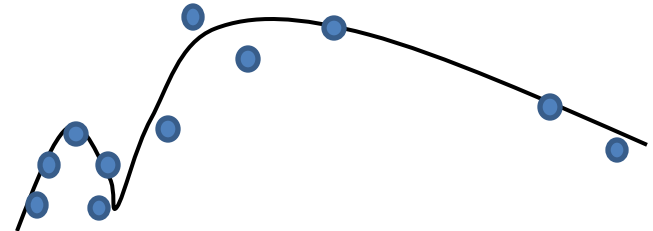
Explicit methods

Build a tessellation over the point cloud. The points map to vertices of the mesh



Implicit Methods

1. Define the surface implicitly, as the zeroes of a function $f_P: \mathbb{R}^3 \rightarrow \mathbb{R}$
2. Tessellate $\{f_P(x)=0\}$



Explicit and Implicit Methods

Explicit methods

Build a triangulation over the point cloud. The points map to vertices of the mesh

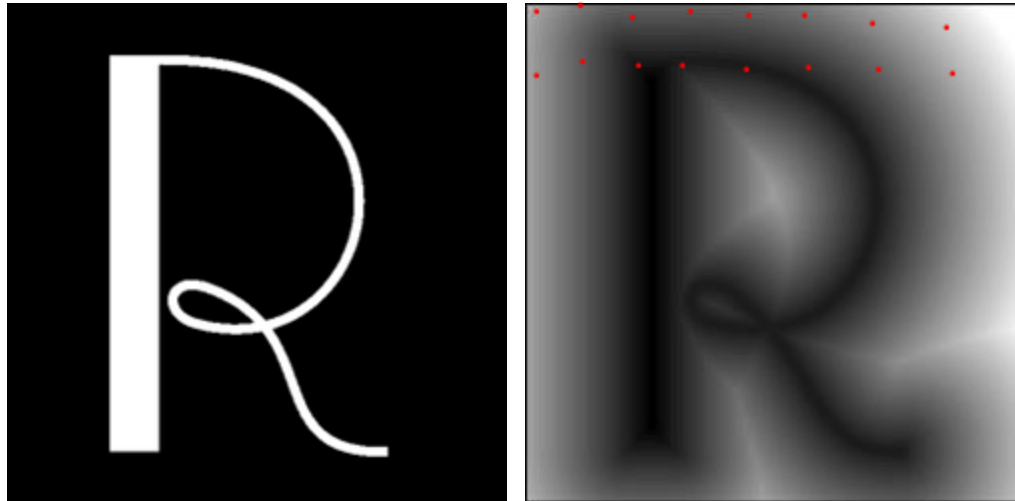
- less robust to noise
- require a dense and even sampling
- Generally easier to implement

Implicit Methods

1. Define the surface implicitly, as the zeroes of a function $f_P: \mathbb{R}^3 \rightarrow \mathbb{R}$
 2. Tessellate $\{f_P(x) = 0\}$
- more robust to noise
 - more resilient to noise and uneven sampling

Volumetric methods

- **define a distance field** from the surface



- return the **isosurface** for 0

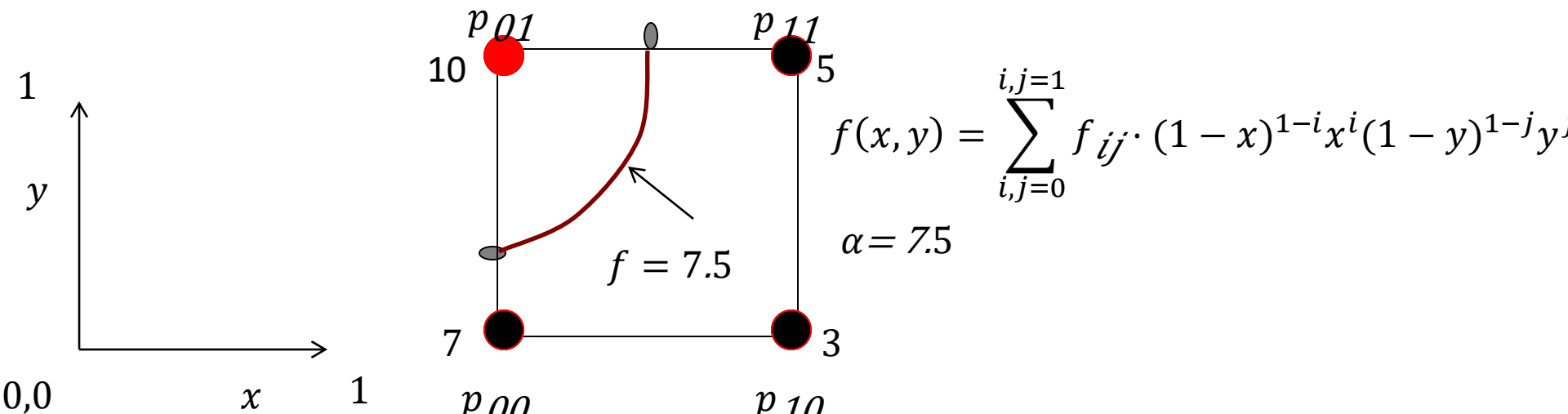
Marching Cubes: isosurfaces from volume data [Lorensen87] :

Input:

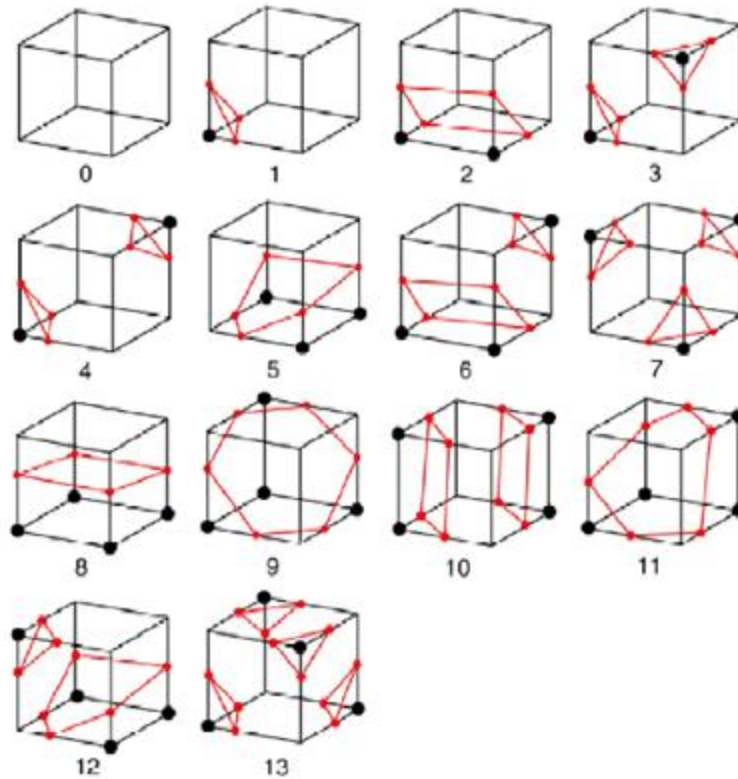
- a regular 3D grid where each node is associated with a scalar value f (i.e. a scalar field)
- a scalar value α

Output: a surface with scalar value α and non null gradient (the isosurface)

The value at p is obtained by trilinear interpolation of the values of the vertices of the grid cell contained in

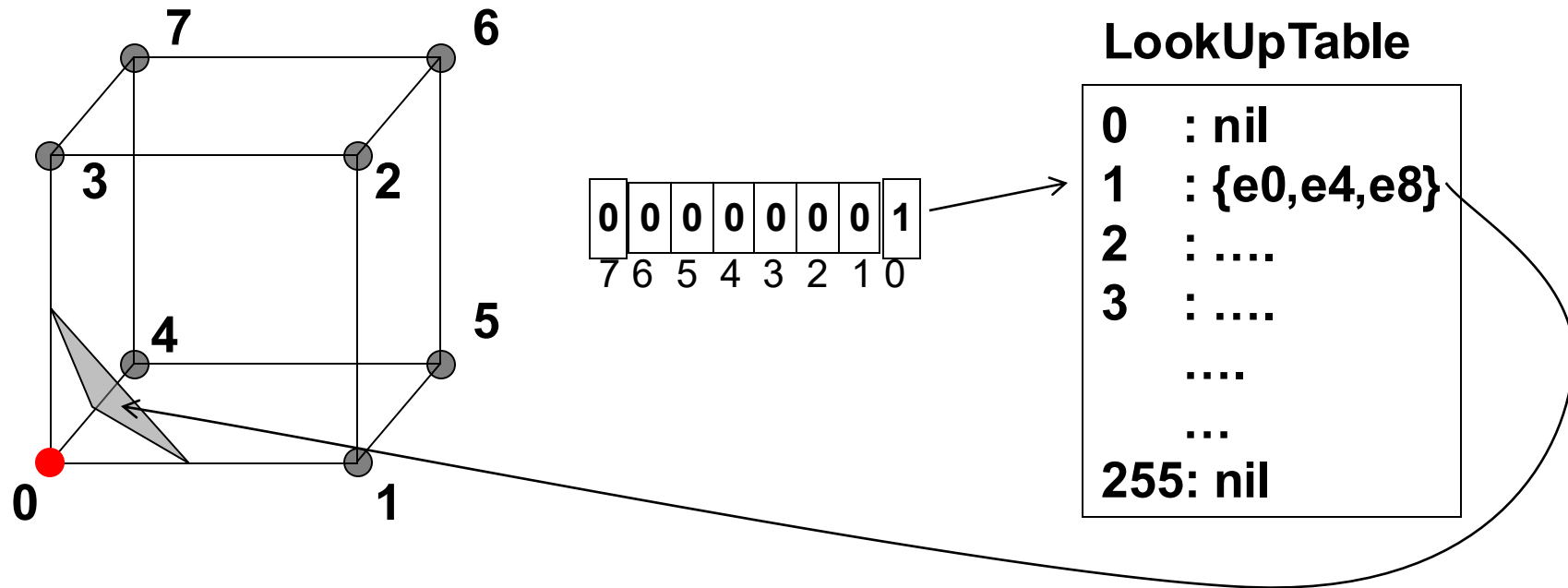


Marching Cube: configurations



- All configurations: $2^8=256$, but only 14 considering rotations, mirroring and complement

Marching Cube: LookUp Table



For each combination of field value respect to the threshold, store the corresponding triangulation.

Marching Cubes: pros/issues

- Pros:

- Quite easy to implement
- Fast and not memory consuming
- Very robust

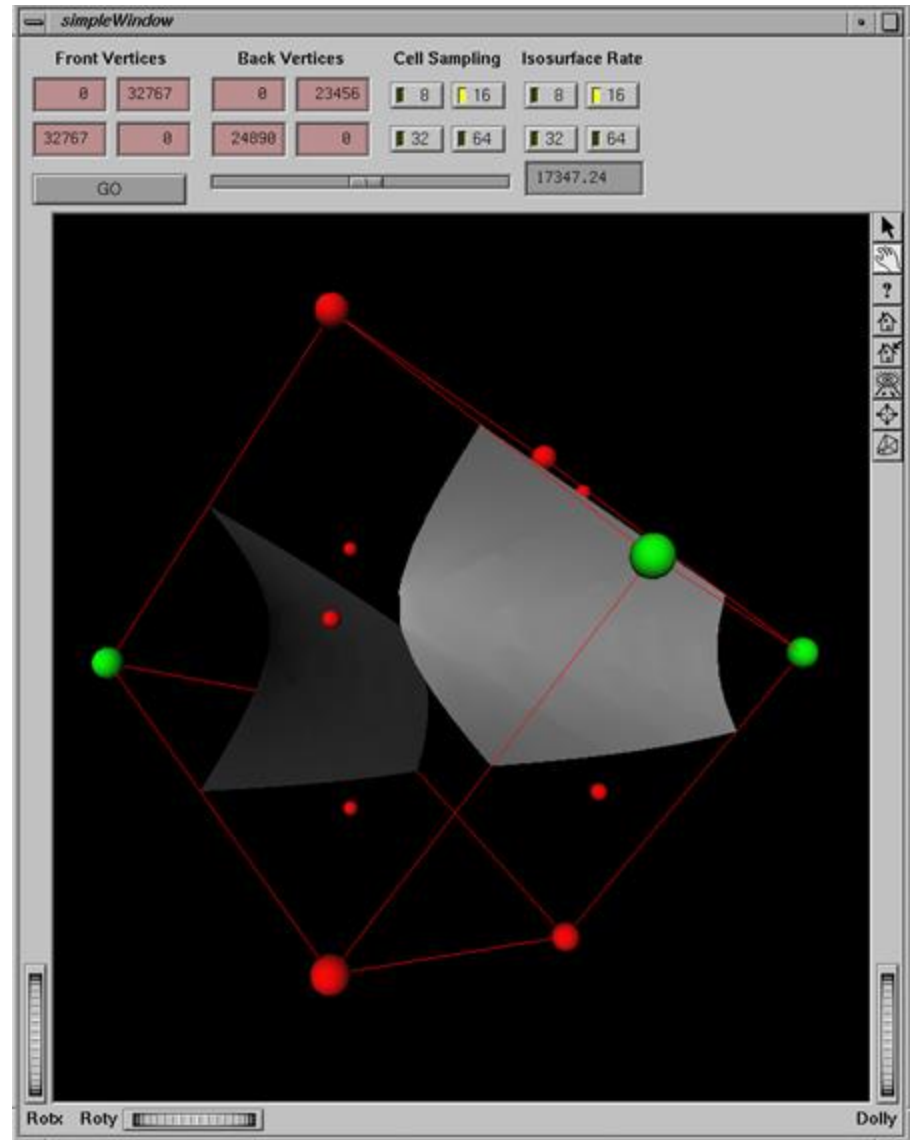
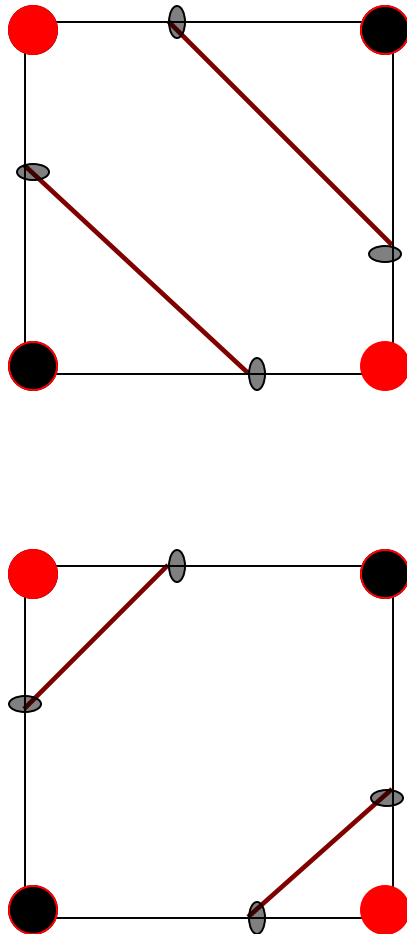
..then why from '87 zillions papers where published ?

Issues:

- **Consistency**. Guarantee a C0 and manifold result: ambiguous cases
- **Correctness**: return a good approximation of the “real” surface
- **Mesh complexity**: the number of triangles does not depend on the shape of the isosurface
- **Mesh quality**: arbitrarily ugly triangles

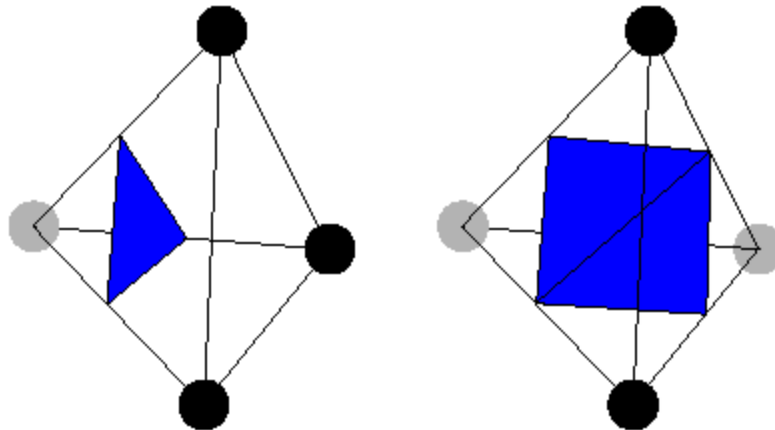
Marching Cubes: ambiguous cases

?



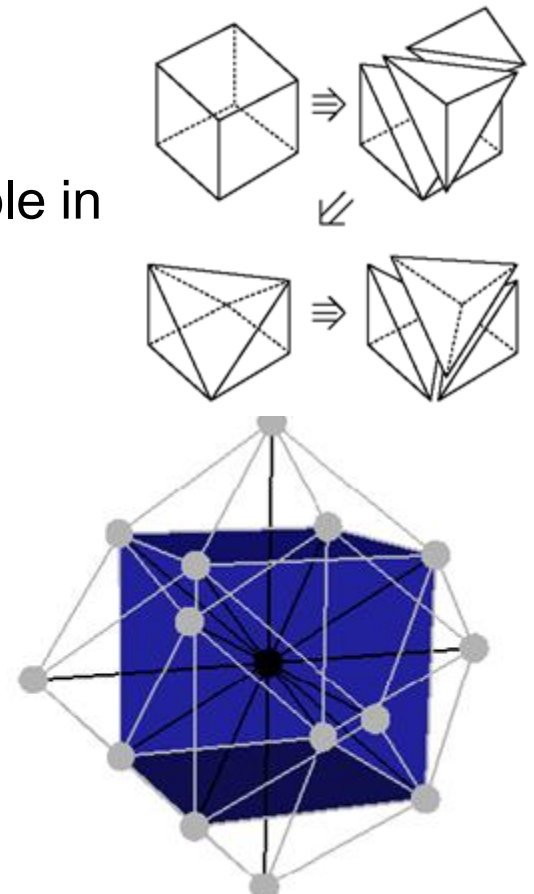
Marching Tetrahedra

- Tetrahedral cells (instead of cubical)
- Only 3 configurations ($2^4 = 16$ permutation of grid values reduce to 3 cases)
- No ambiguities (linear field on a value is planar)
- Boundary cases are easy to be managed too
 - Cases where you get a vertex with exact threshold value
- but it may be “less” correct
 - If you start from a cubic grid the tetrahedral decomposition is a biasing choice



Marching Tetrahedra

- Original approach [Treece99]: cubic cells are partitioned in 5 (o 6) tetrahedra.
 - Subdivision determines topology
- Body centered cubic lattice: one more sample in the cubic cell
 - Unique subdivision
 - Equal tetrahedral
 - Better surface (better triangles)



Resolving ambiguities

- The value of the scalar function inside each cell is interpolated by the (known) value of its 8 corners

$$T(x,y,z) = axyz + bxy + cyz + dxz + ex + fy + gz + h$$

$$a = v_1 + v_3 + v_4 + v_6 - v_0 - v_7 - v_5 - v_2$$

$$b = v_0 + v_2 - v_1 - v_3$$

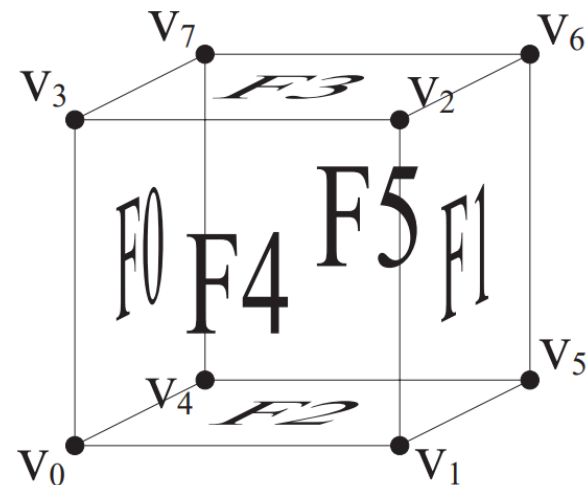
$$c = v_0 + v_7 - v_4 - v_3$$

$$d = v_0 + v_5 - v_1 - v_4$$

$$e = v_1 - v_0$$

$$f = v_3 - v_0$$

$$g = v_4$$



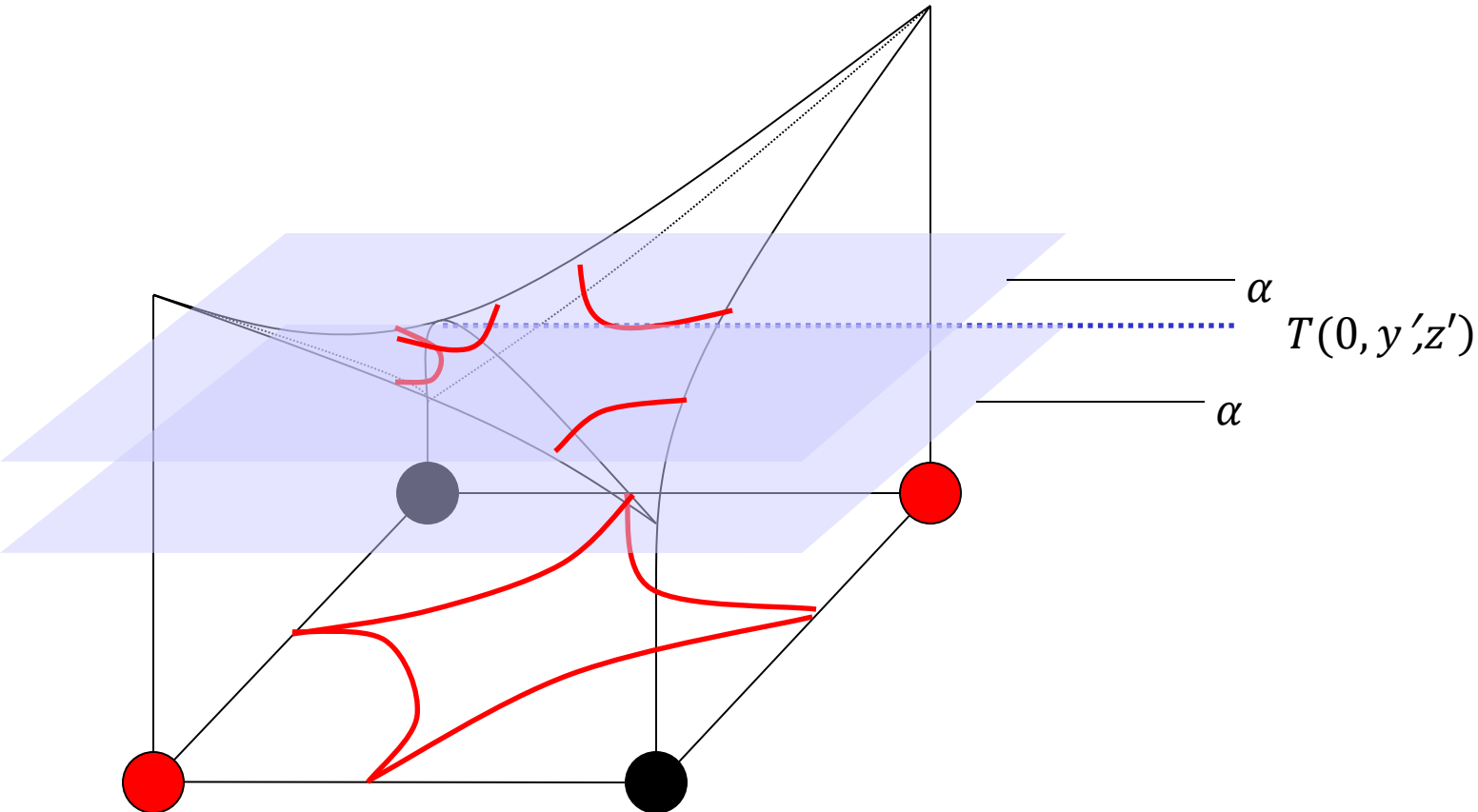
Saddle points

Field value on a cell's face

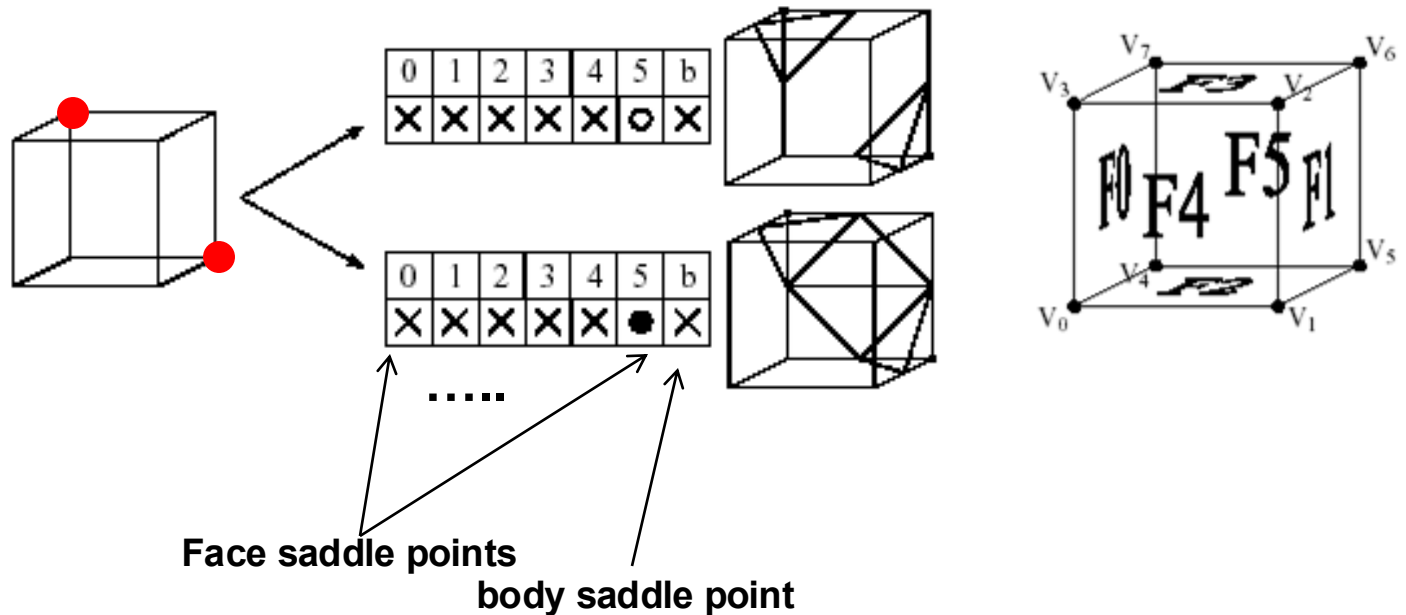
$$T(0, y, z) = cyz + fy + gz + h$$

$$\frac{\partial T(0, y', z')}{\partial y} = cz' + f = 0 \Rightarrow z' = -\frac{f}{c}$$

$$\frac{\partial T(0, y', z')}{\partial z} = cy' + g = 0 \Rightarrow y' = -\frac{g}{c}$$



ELUT: Exhaustive LUT [Cignoni00]

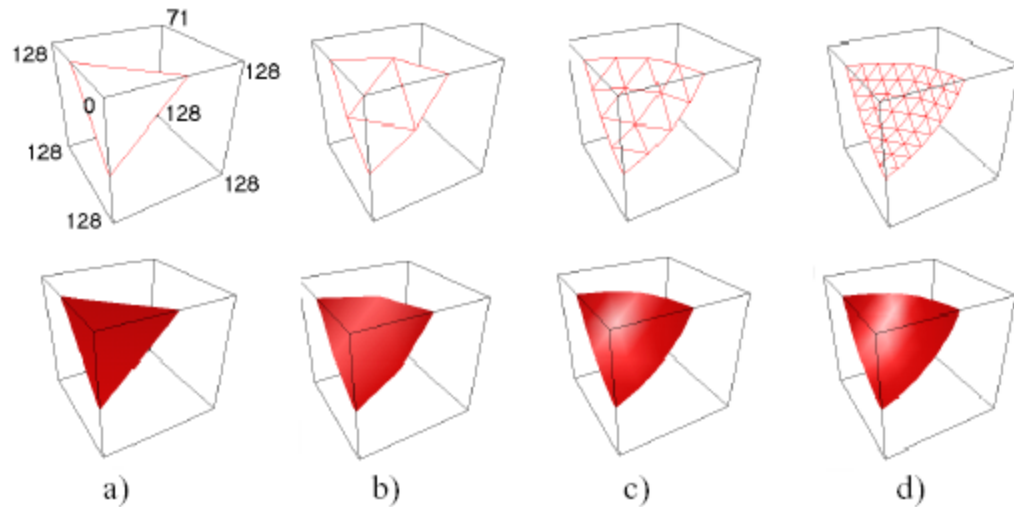


ELUT:

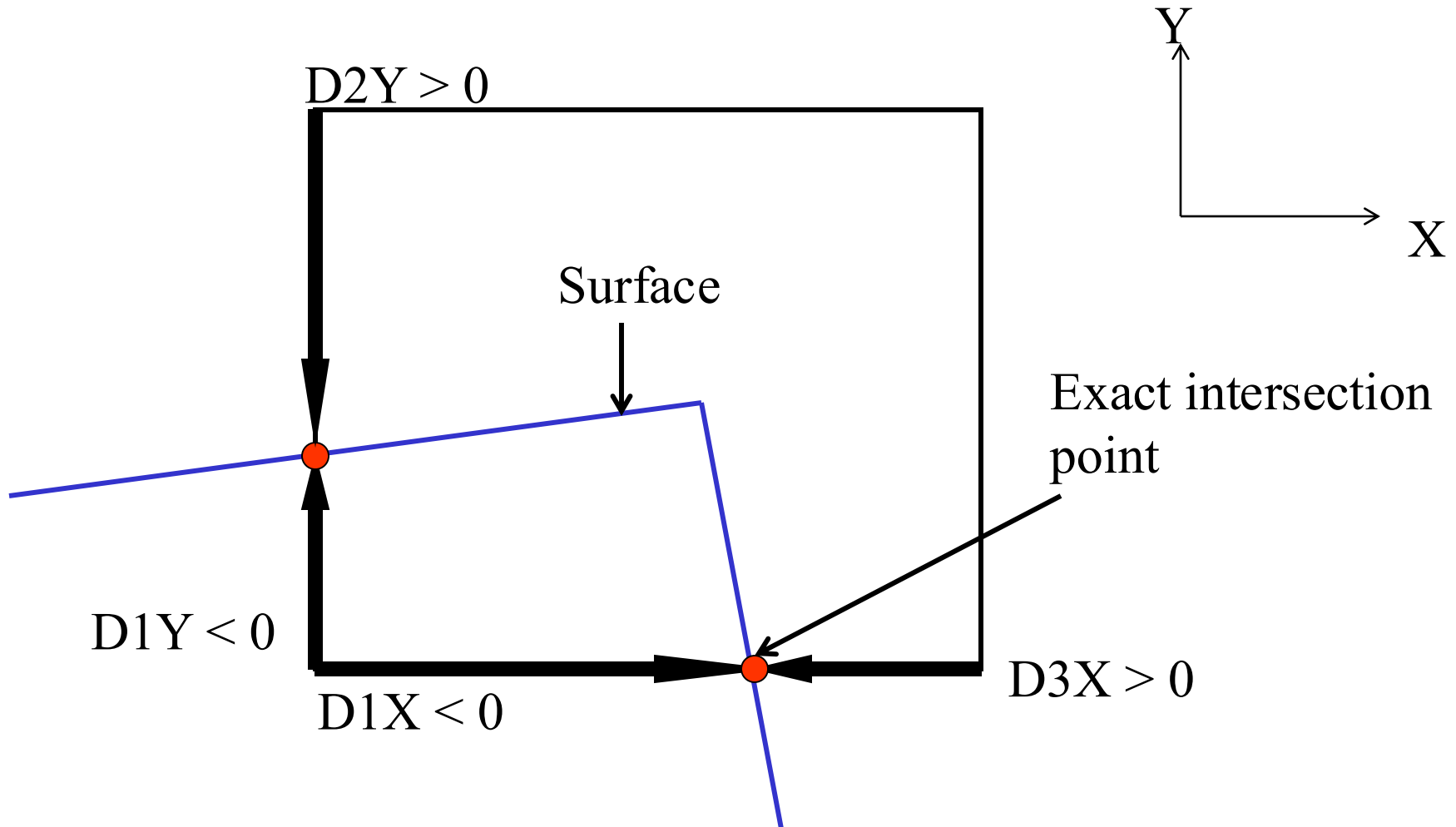
For each ambiguous configuration determines the coherent internal triangulation looking at the saddle points

Adaptive triangulation

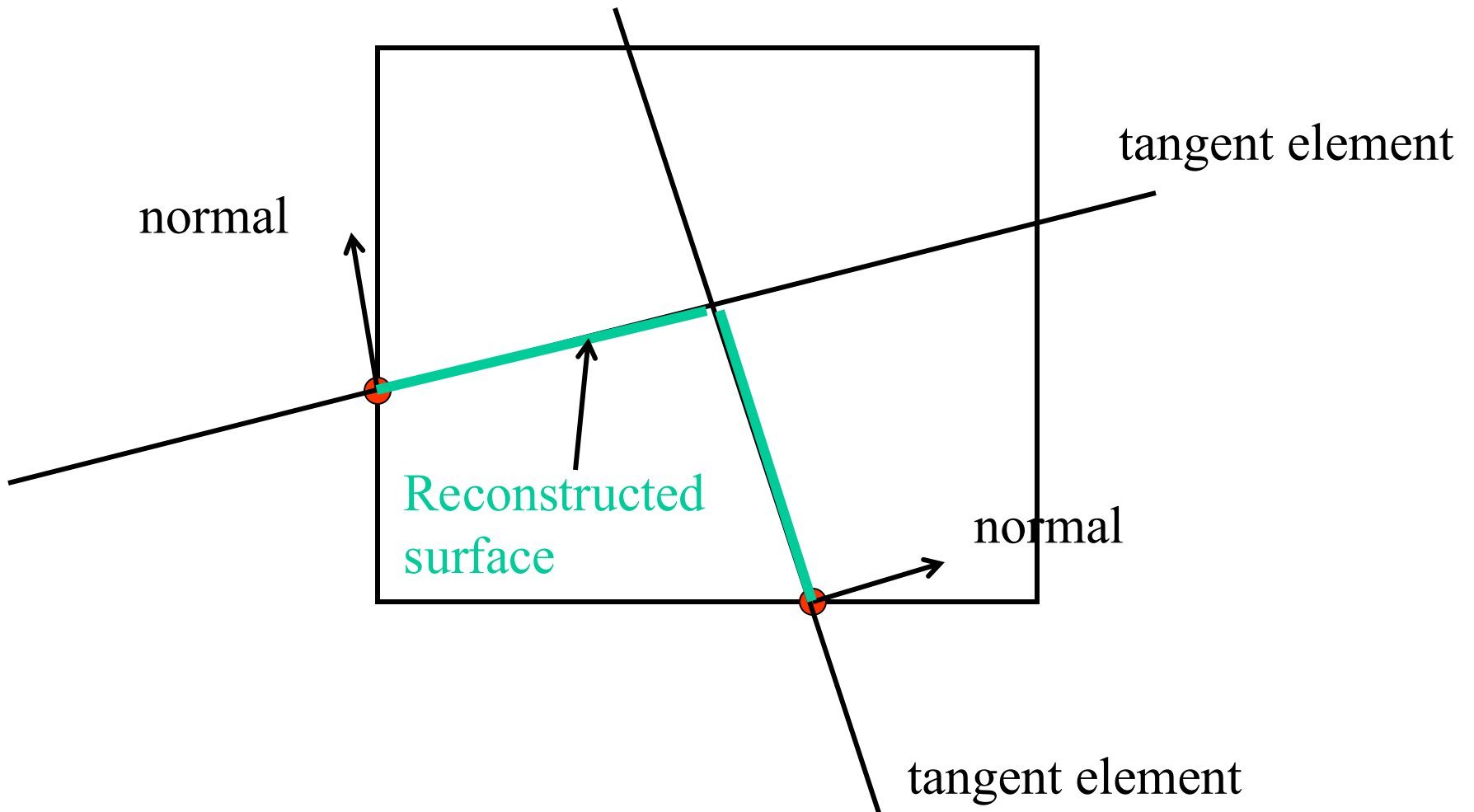
- Refine for better approximation (re-evaluate scalar field)



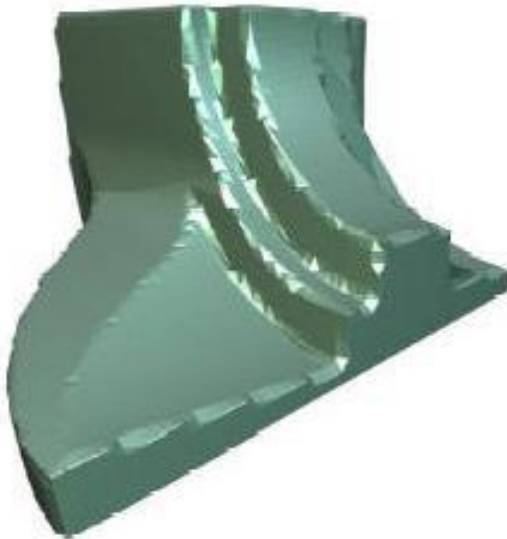
Extended MC [Kobbelt01]



MC



Extended MC



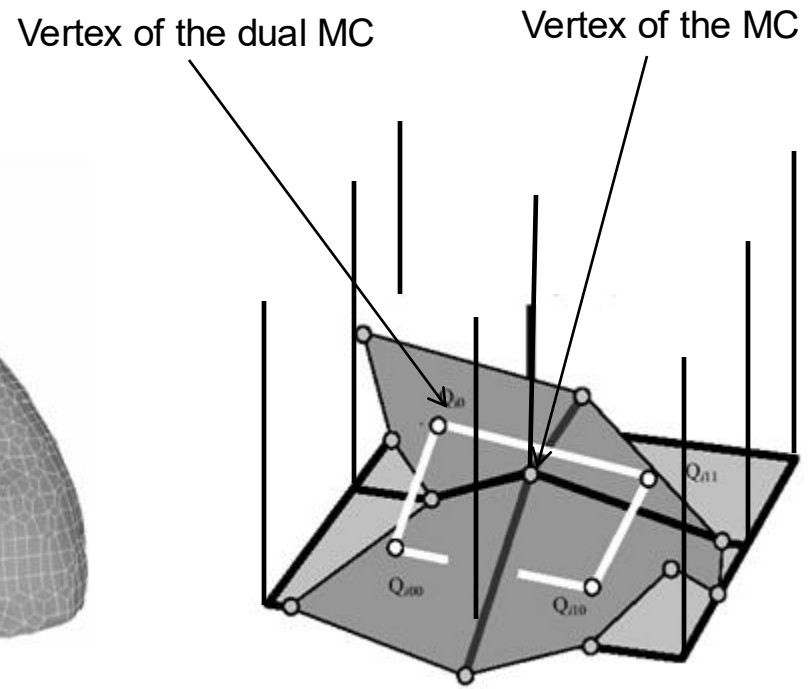
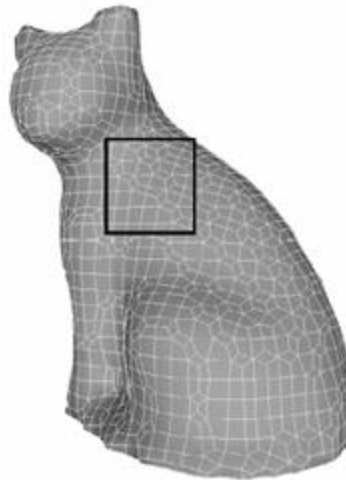
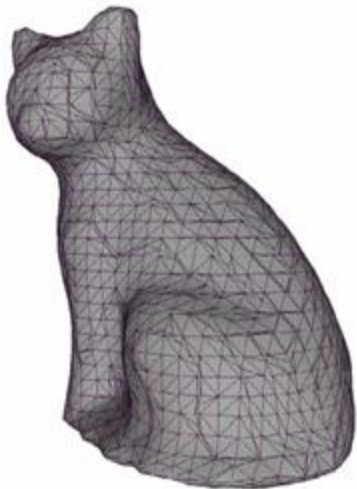
Marching Cubes



Extended Marching Cubes

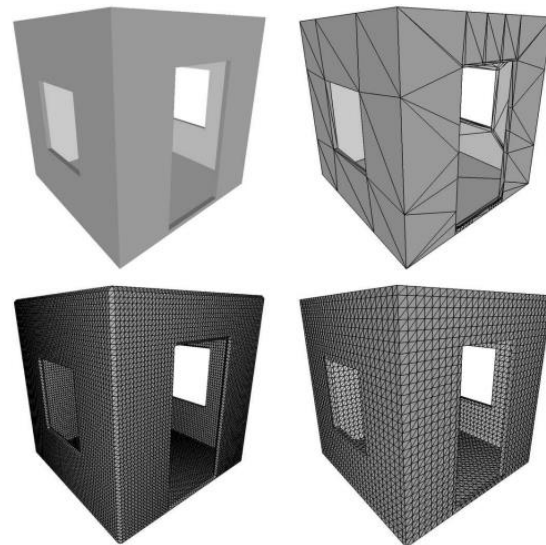
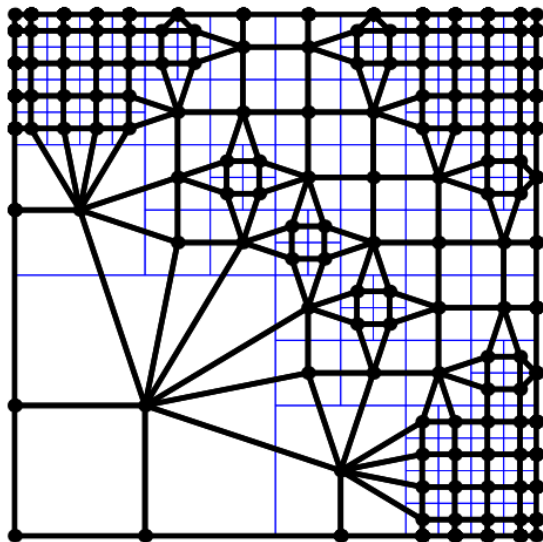
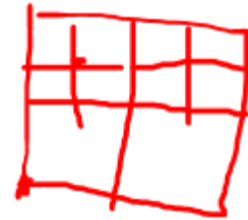
Dual Marching Cubes [Nielson04]

- one vertex for each patch generated by MC
- One quad for each intersected edge (the 4 vertices associated to the patches of the cells sharing the edge)
- Tends to improve triangles quality



Dual Marching Cubes: Primal Contouring of Dual Grids [Shaeffer04]

- Partition the space with an Octree
- Build the dual grid
- Run MC on the dual grid (consider non hexahedral cells as HC with collapsed edges)



From point cloud to a scalar field...

Problem: given a set of points $\{x_0, \dots, x_n\}$, define

$$f(x) = \varphi(\{x_0, \dots, x_n\})$$

$$S = \{x \mid f(x) = \alpha\}$$

so that S interpolates/approximates the point cloud

Normals are often either assumed or computed from the point cloud

Point Cloud Normals (1/2)

- Normals are important to define the surface



- Most of methods for building a surface from point cloud compute the normal on the points

Point Cloud Normals (1/2)

- Normals are important to define the surface



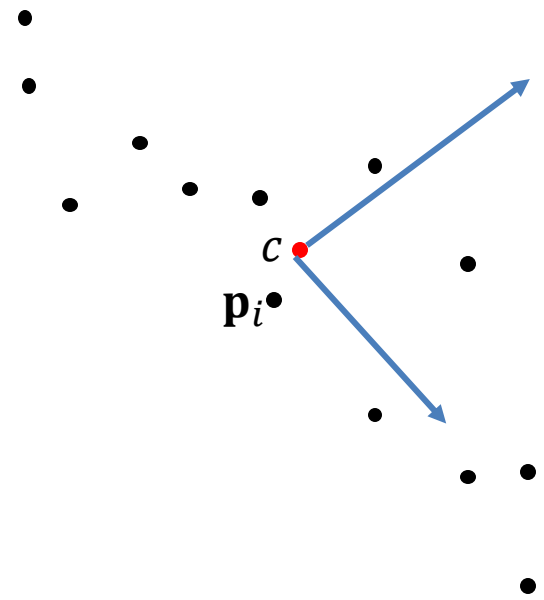
- Most of methods for building a surface from point cloud compute the normal on the points

Point Cloud Normals (2/2)

- Use Principal Component Analysis (PCA)

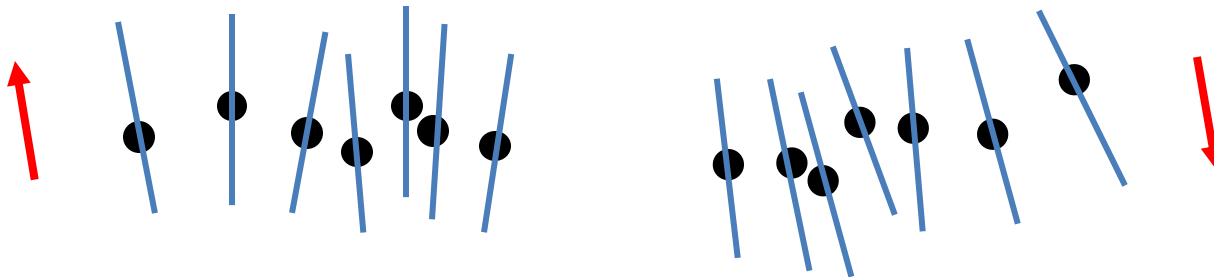
$$\mathbf{q}_i = \mathbf{p}_i - \mathbf{c}$$
$$\mathbf{C}_{ov} = \sum_i \mathbf{q}_i \mathbf{q}_i^T = \begin{bmatrix} \sum_i q_{ix}^2 & \sum_i q_{ix} q_{iy} & \sum_i q_{ix} q_{iz} \\ \sum_i q_{iy} q_{ix} & \sum_i q_{iy}^2 & \sum_i q_{iy} q_{iz} \\ \sum_i q_{iz} q_{ix} & \sum_i q_{iz} q_{iy} & \sum_i q_{iz}^2 \end{bmatrix}$$

- $\mathbf{C}_{ov}$ is symmetric $\rightarrow$ real eigenvalues and orthogonal eigenvectors
- take the eigenvector corresponding to the smallest eigenvalue as normal direction
 - Check that the smallest eigenvalue is unique
 - Check that the other two are similar



Point Cloud Normals (3/2)

- PCA is good for finding for each point a *direction* e.g. a line
- You have to choose a consistent alignment for groups of points
- Usually, heuristics based on sign propagation
 - Many difficult cases can arise for unconnected sets

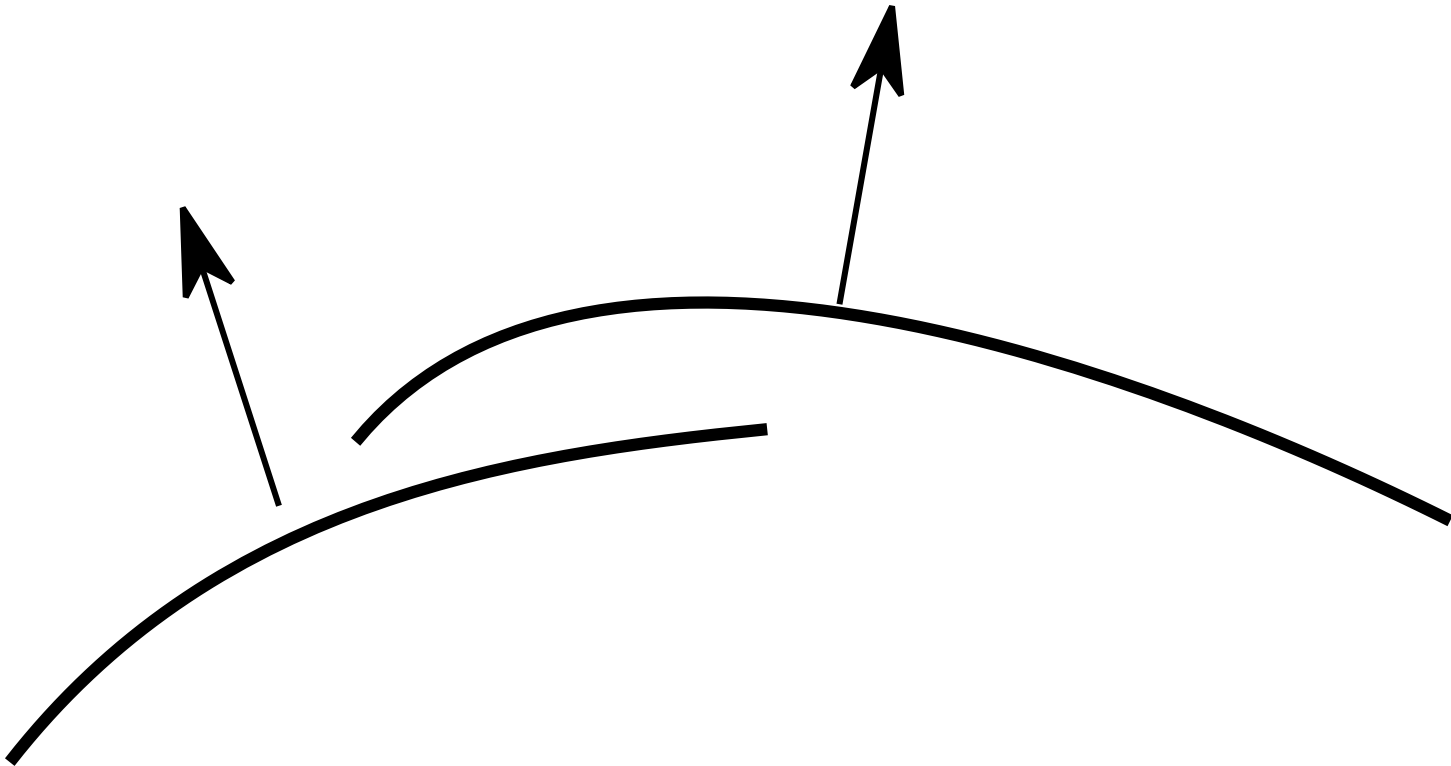


VCG Reconstruction/[Curless96]

- Suppose we do have aligned range maps
- We want to get a nice ISOSurface
 1. Compute signed distance field from each range map
 2. Average them
 3. Extract the isosurface

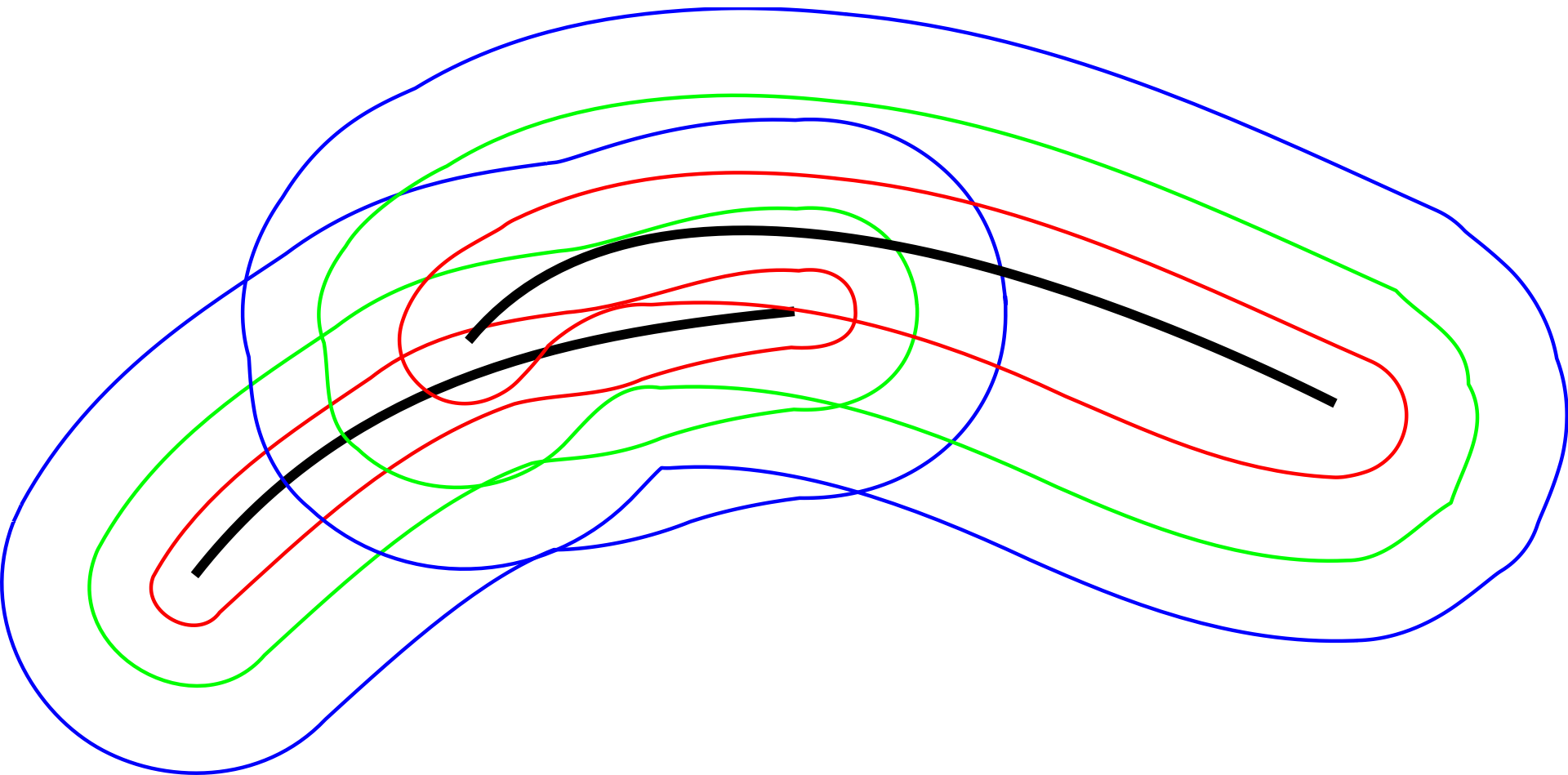
VCG Reconstruction/[Curless96]

▣ Surfaces with Normals



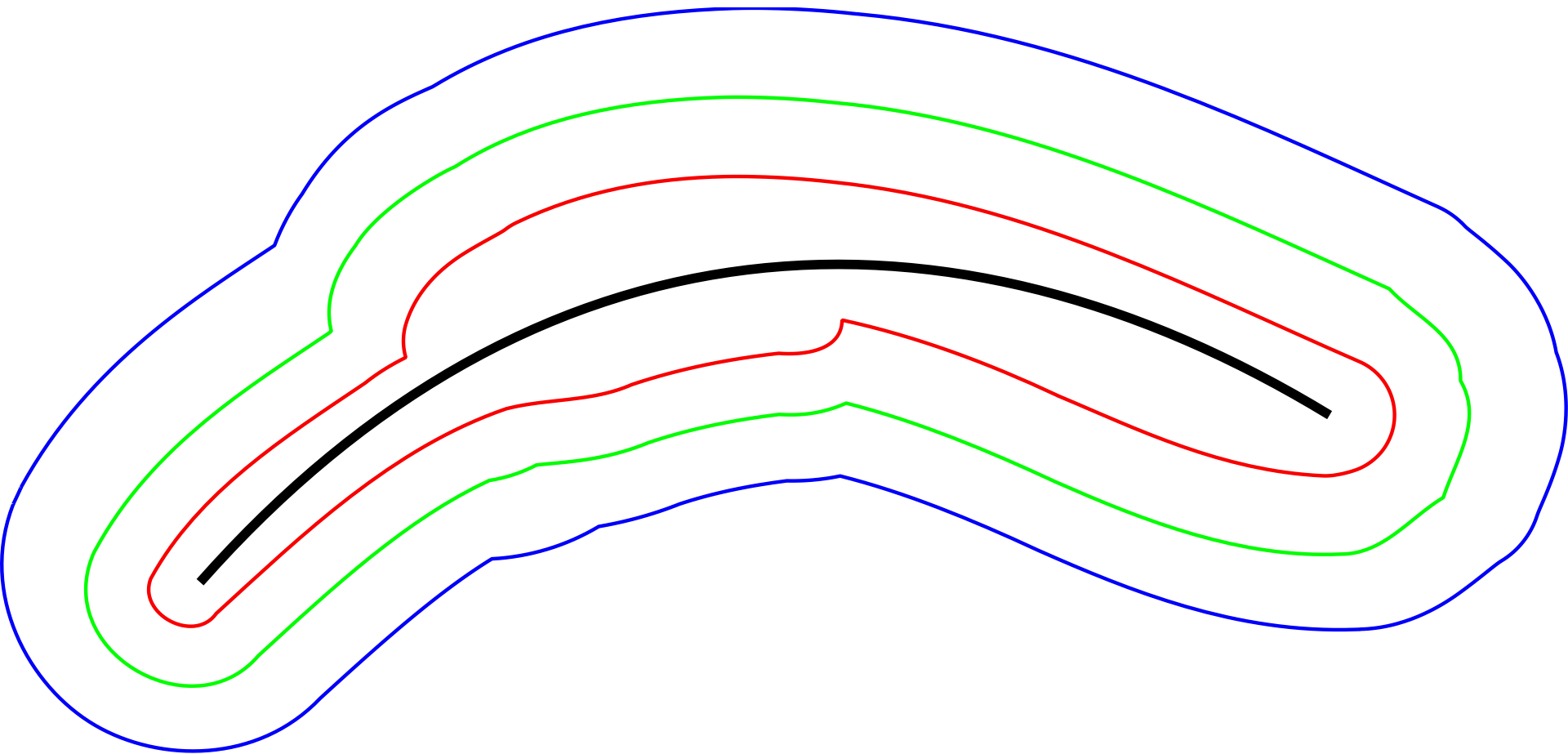
VCG Reconstruction/[Curless96]

- Compute Distance Fields (signed)



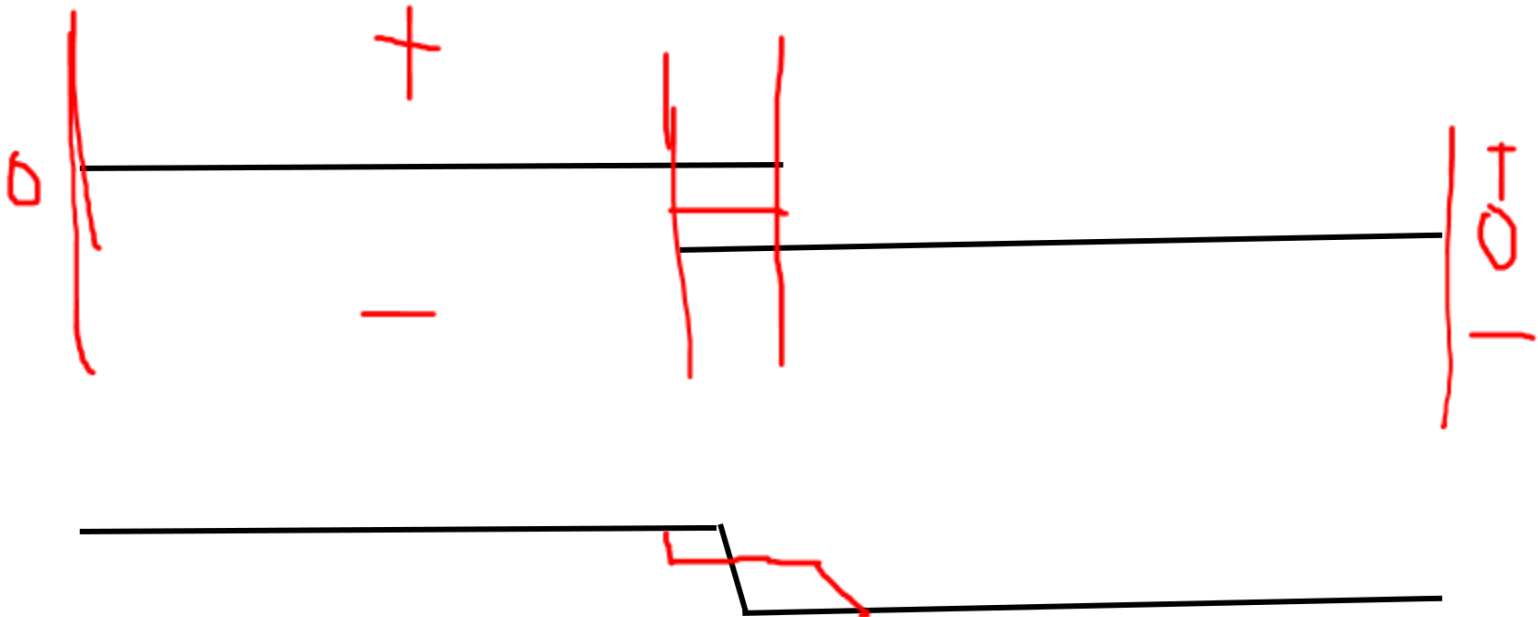
VCG Reconstruction/[Curless96]

■ Average Distance Fields!



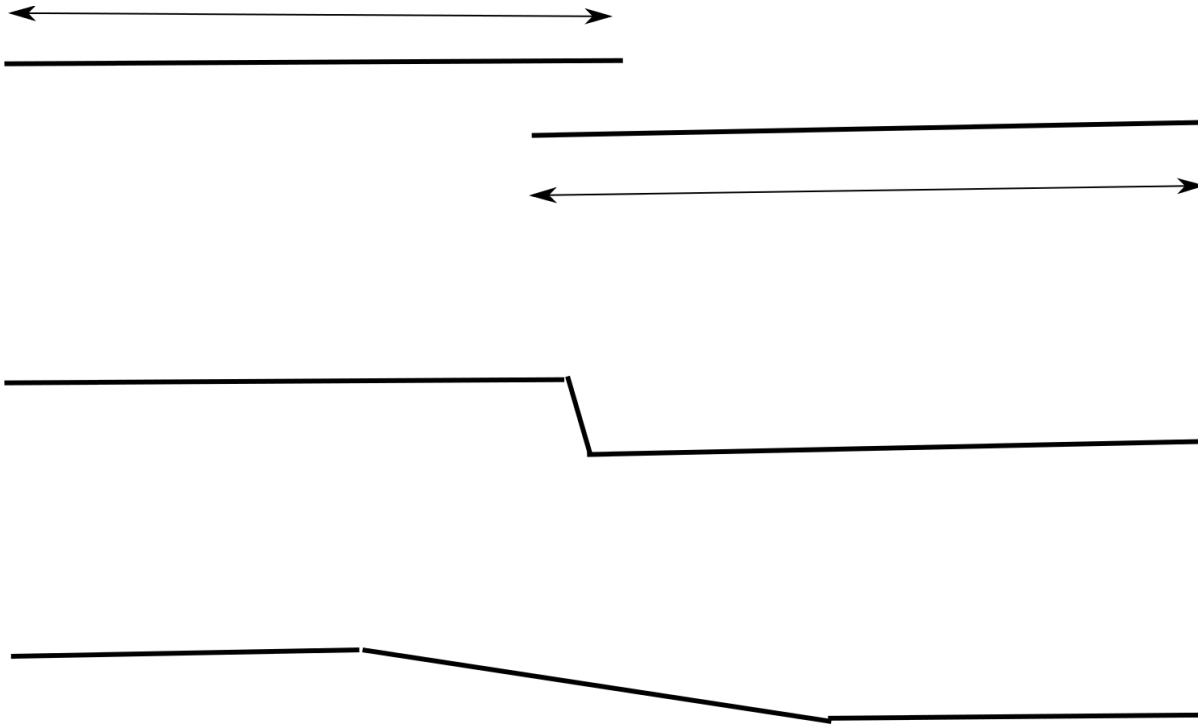
VCG Reconstruction: Issue

- This simple averaging can cause abrupt jumps



VCG Reconstruction (Use of geodesic)

- This simple averaging can cause abrupt jumps
- Solution: Weight the averaging by geodesic distance to border

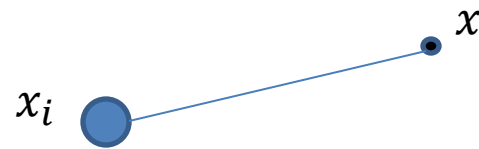


Metaballs [Blinnn92,Wyvill86]

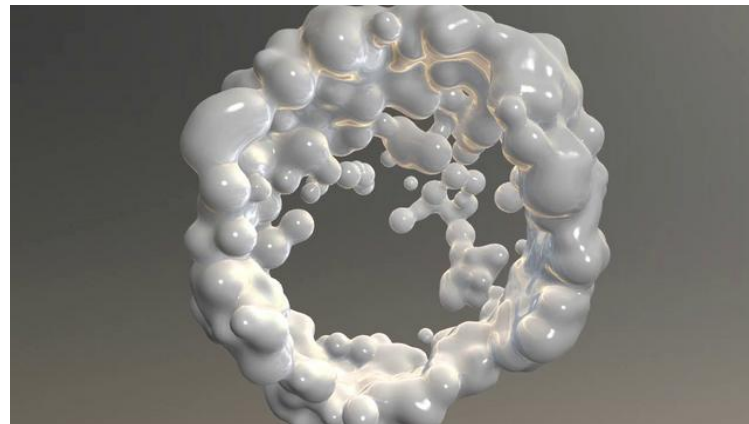
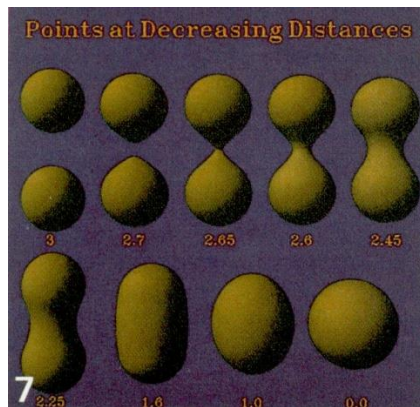
- f is the sum of function that have maximum in the points and decay with the distance*

$$f(x_i) = 1 \quad f(R) = 0$$

$$f'(x_i) = 0 \quad f'(R) = 0$$



$$f(x) = \sum_i \left(2 \frac{r^3}{R^3} - 3 \frac{r^2}{R^2} + 1 \right), r = \|x - x_i\|, R = \textit{support radius}$$



Radial Basis Functions (RBF)

Solutions that follow the general scheme:

$$f(x) = p(x) + \sum_i \omega_i \varphi(\|x - x_i\|)$$

$$f(x_i) = f_i$$

weights: $\omega_i \in \mathbb{R}$

RBF: $\varphi: \mathbb{R} \rightarrow \mathbb{R}$

p is a polynome

Radial Basis Functions (RBF)_[Carr01]

$$f(x) = p(x) + \sum_i \omega_i \varphi(\|x - x_i\|), \quad \begin{array}{l} \omega_i \in \mathbb{R} \\ \varphi: \mathbb{R} \rightarrow \mathbb{R} \\ p \text{ a polynome} \end{array}$$
$$\begin{bmatrix} A & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} \omega \\ c \end{bmatrix} = \begin{bmatrix} F \\ 0 \end{bmatrix}$$

$$F = [f(x_1), \dots, f(x_N)]^T$$

$$A_{ij} = \varphi(\|x_j - x_i\|)$$

p: basis for **all** polynomials of degree k

$$P_{ij} = p_j(x_i)$$

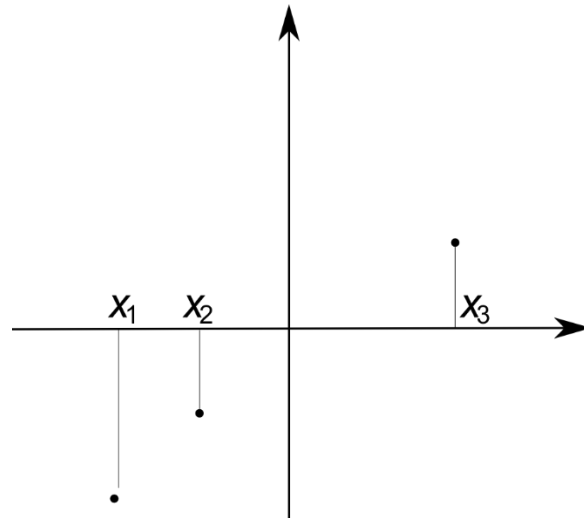
Examples of polynomial basis:

$$p = \{1, x, y, z\} \quad d=3, m=1$$

$$p = \{1, x, y, x^2, xy, y^2\} \quad d=2, m=2$$

$$p = \{1, x, x^2, x^3\} \quad d=1, m=3$$

Example



$$x_1 = -2 \quad f_1 = -2$$

$$x_2 = -1 \quad f_2 = -1$$

$$x_3 = 2 \quad f_3 = 1$$

Polynomial basis: $\{1, x\}$

$$P = \{1, x\}$$

$$\varphi(d) = d$$

$$\begin{bmatrix} \varphi(|x_1, x_1|) & \varphi(|x_1, x_2|) & \varphi(|x_1, x_3|) & p_1(x_1) & p_2(x_1) \\ \varphi(|x_2, x_1|) & \varphi(|x_2, x_2|) & \varphi(|x_2, x_3|) & p_1(x_2) & p_2(x_2) \\ \varphi(|x_3, x_1|) & \varphi(|x_3, x_2|) & \varphi(|x_3, x_3|) & p_1(x_3) & p_2(x_3) \\ p_1(x_1) & p_1(x_2) & p_1(x_3) & 0 & 0 \\ p_2(x_1) & p_2(x_2) & p_2(x_3) & 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ 0 \\ 0 \end{bmatrix}$$

Example

$$\begin{bmatrix} \varphi(|x_1, x_1|) & \varphi(|x_1, x_2|) & \varphi(|x_1, x_3|) & p_1(x_1) & p_2(x_1) \\ \varphi(|x_2, x_1|) & \varphi(|x_2, x_2|) & \varphi(|x_2, x_3|) & p_1(x_2) & p_2(x_2) \\ \varphi(|x_3, x_1|) & \varphi(|x_3, x_2|) & \varphi(|x_3, x_3|) & p_1(x_3) & p_2(x_3) \\ p_1(x_1) & p_1(x_2) & p_1(x_3) & 0 & 0 \\ p_2(x_1) & p_2(x_2) & p_2(x_3) & 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ 0 \\ 0 \end{bmatrix}$$

$\Rightarrow$

$$\begin{bmatrix} 0 & 1 & 4 & 1 & -2 \\ 1 & 0 & 3 & 1 & -1 \\ 4 & 3 & 0 & 1 & 2 \\ 1 & 1 & 1 & 0 & 0 \\ -2 & -1 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} -2 \\ -1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 0.125 \\ -0.166 \\ 0.0416 \\ -0.5 \\ 0.75 \end{bmatrix}$$

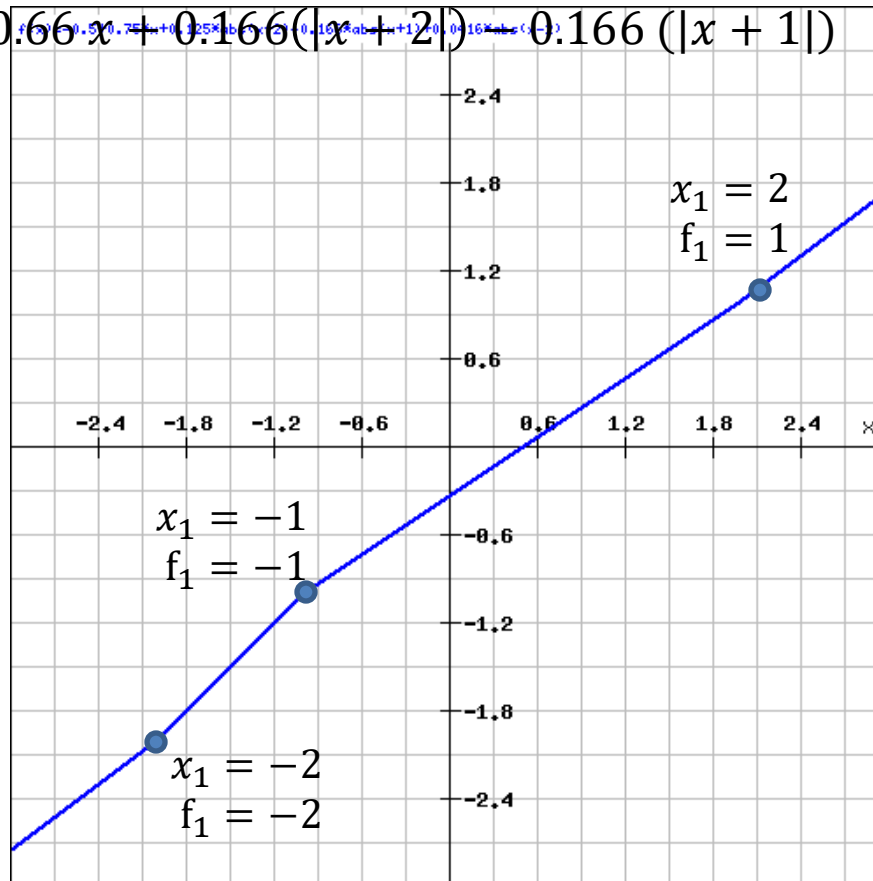
$$f(x) = -0.5 + 0.75x + 0.125|x+2| - 0.166|x+1| + 0.0416|x-2| =$$

$$= -0.334 + 0.66x + 0.166(|x+2|) - 0.166(|x+1|)$$

Example

$$f(x) = -0.5 + 0.75x + 0.125|x + 2| - 0.166|x + 1| + 0.0416|x - 2|$$

$$= -0.334 + 0.66x + 0.166(|x + 2|) - 0.166(|x + 1|)$$



Radial Basis Functions (RBF)

- Several possible choices for φ and p :
 - $\varphi(d) = d$, *linear polynomial*
 - $\varphi(d) = d^2$, *linear polynomial*
 - $\varphi(d) = d^3$, *linear/quadratic polynomial*
 - $\varphi(d) = d^2 \log(d)$, *linear/quadratic polynomial*
 -
- Issue 1: if functions have **unbounded** support, i.e. nonzero everywhere, the matrix will always be dense
 - Expensive to solve when n increase...
- Issue 2: the whole surface is influenced by each single input point

Bounded RBD [Morse01]

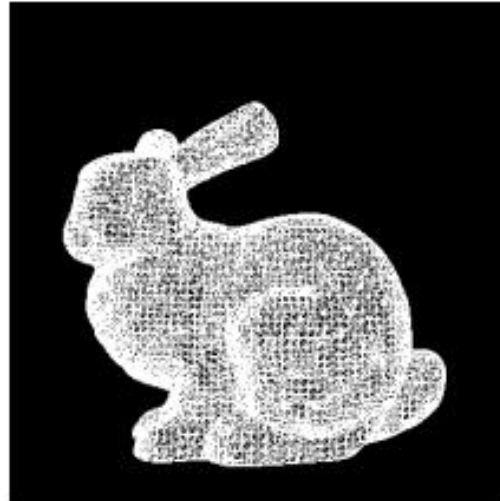
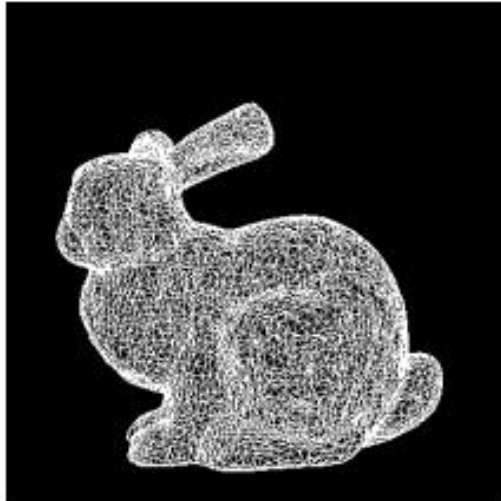
$$\varphi(d) = \begin{cases} (1 - d)^p P(d), & d < 1 \\ 0, & d \geq 1 \end{cases}$$

$P(d) = \text{polynome with degree } 6$

- The value of f is determined only locally (within the radius 1)
 - Use $\varphi(d/R)$ to adapt to the point cloud resolution
- The resulting matrix is **sparse**
- The *fitting* is local

Revised DDD

- The



(radius 1)

- The
- The



8000-point model



Interpolated to 41,864 points

Bounded RBF

$$\varphi(d) = \begin{cases} (1 - d)^p P(d), & d < 1 \\ 0, & d \geq 1 \end{cases}$$

$P(d) = \text{polynome with degree } 6$

- The value of f is determined only locally (withing the radius 1)
 - Use $\varphi(d/R)$ to adapt to the point cloud resolution
- The resulting matrix is **sparse**
- The *fitting* is local

More issues:

- Still hard to represent sharp features, anisotropic basis functions may be used [Dinh01]

Partition of Unity

$$f(x) = \sum_i \varphi_i(x) Q_i(x)$$

- $f(x)$ is defined globally as the weighted sum of local functions that describe (implicitly) the surface
- Each i corresponds to a region of $\mathbb{R}^3$ where the function is described by $f_i(x)$
- **The sum of the weights is 1 everywhere:**

$$\sum_i \varphi_i(x) = 1$$

- Which is obtained by normalization

$$\varphi_i(x) = \frac{\omega_i(x)}{\sum_i \omega_i(x)} \quad \{\omega_i(\mathbf{x})\} \text{ s. t. } \Omega \subset \bigcup_i \text{supp}(\omega_i)$$

Multilevel PoUI [Ohtake03]

- Starting from the bounding box of the point cloud, build an **octree**
- The rule for creating the children of a node is:
Can we define an implicit surface with the point corresponding to the cell as:

$$f(x) = \sum_i \varphi_i(x) Q_i(x)$$

- for $Q_i(x)$ in a set of predefined shape functions
- With and approximation error less than ε ?

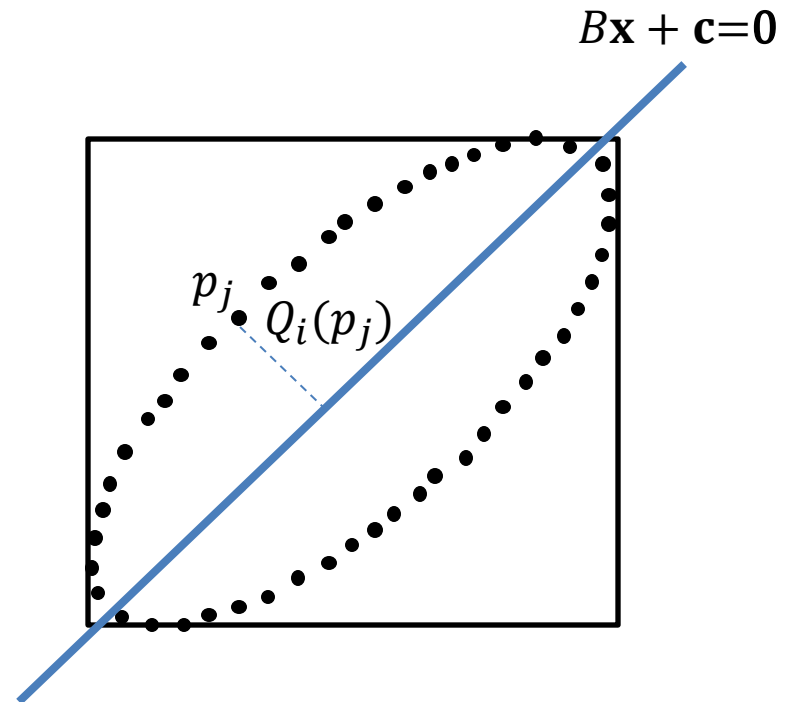
Multilevel PoUI [Ohtake03]

- (simplified) Example

$$f(x) = \sum_i \varphi_i(x) Q_i(x)$$

shape $Q_i(x) = B\mathbf{x} + \mathbf{c}$

approx $\varepsilon = \sum_j |Q_i(p_j)|$



Error is big, split

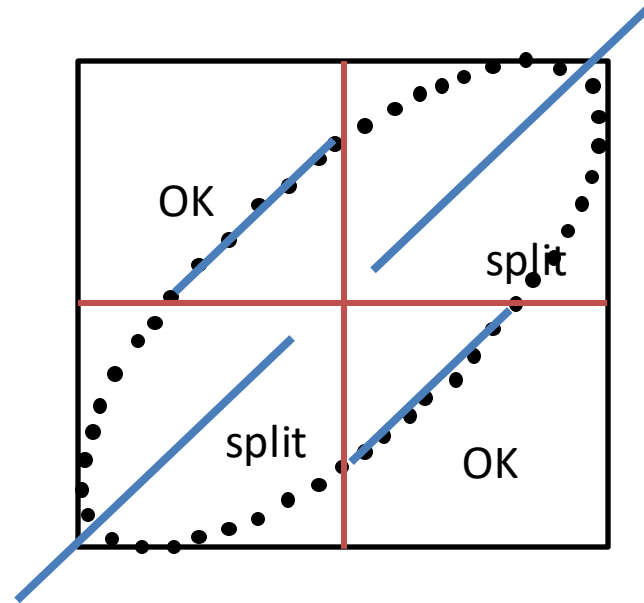
Multilevel PoUI [Ohtake03]

- (simplified) Example

$$f(x) = \sum_i \varphi_i(x) Q_i(x)$$

shape $Q_i(x) = B\mathbf{x} + \mathbf{c}$

approx $\varepsilon = \sum_j |Q_i(p_j)|$



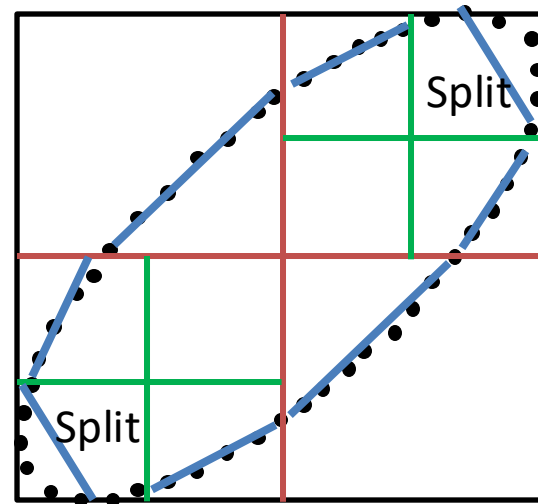
Multilevel PoUI [Ohtake03]

- (simplified) Example

$$f(x) = \sum_i \varphi_i(x) Q_i(x)$$

shape $Q_i(x) = B\mathbf{x} + \mathbf{c}$

approx $\varepsilon = \sum_j |Q_i(p_j)|$



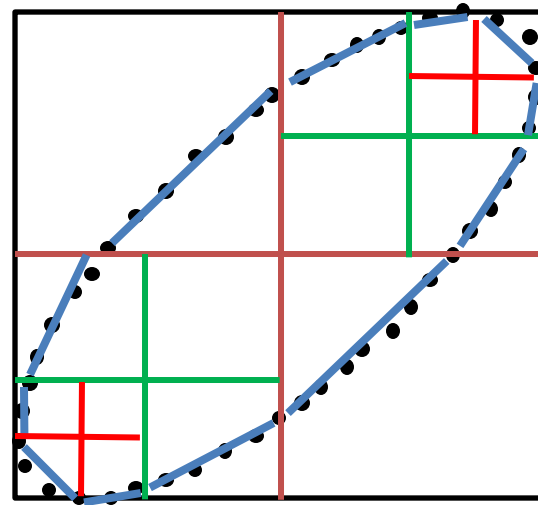
Multilevel PoUI [Ohtake03]

- (simplified) Example

$$f(x) = \sum_i \varphi_i(x) Q_i(x)$$

shape $Q_i(x) = B\mathbf{x} + \mathbf{c}$

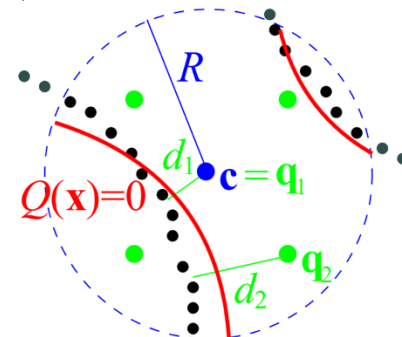
approx $\varepsilon = \sum_j |Q_i(p_j)|$



Multilevel PoUI

- Subdivide the domain with an **octree**
- Fit the points within each cell with a function $Q_i(x)$, either:
 - A quadric (for noisy and unbounded regions)

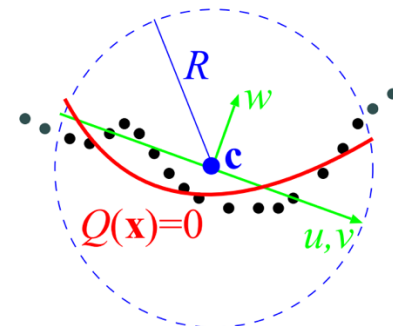
$$Q_i(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{x} + \mathbf{c}$$



- A bivariate (u,v) quadratic polynomial in a local coordinate system (for smooth patch)

$$Q_i(\mathbf{x}) = w - [u, v]^T \mathbf{A} \begin{bmatrix} u \\ v \end{bmatrix} + \mathbf{b}^T \begin{bmatrix} u \\ v \end{bmatrix} + \mathbf{c}$$

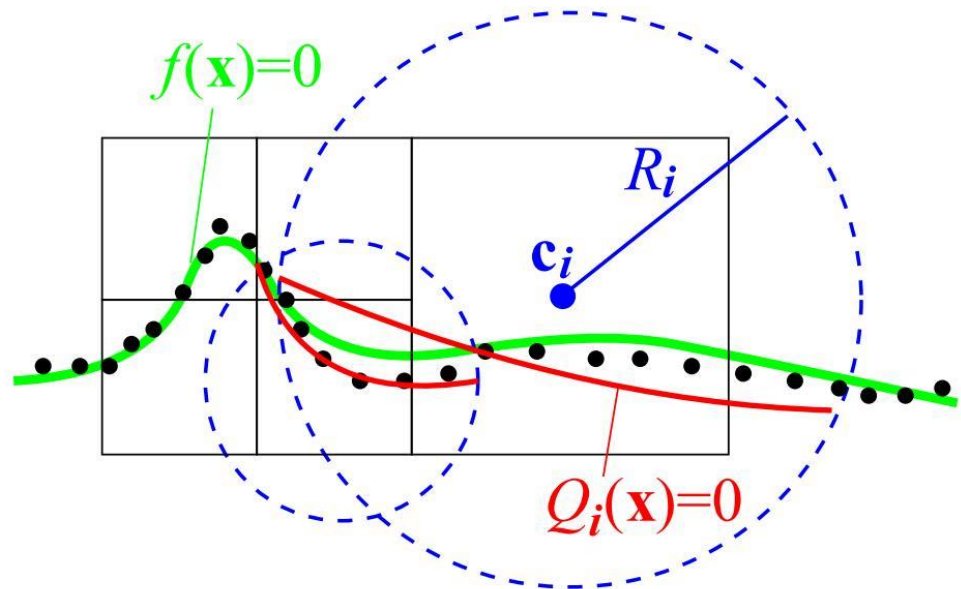
$[u, v, w]^T$ point expressed in a local frame



Multilevel PoUI

- Subdivide the domain with an **octree**
- Fit the points within each cell with a function $Q_i(x)$, either:
 - A quadric (for noisy and unbounded regions)
 - A bivariate (u,v) quadratic polynomial in a local coordinate system (for smooth patch)
 - A piecewise quadratic surface (for sharp features)
- Blending PU:

$$\omega_i(x) = b \left(\frac{3|x - c_i|}{2R_i} \right)$$
$$R_i = 0.75 * diag$$



Results



Distance field from range maps [Levoy]



MPU implicits

Moving Least Square Reconstruction

LS

Least square

$$\min_{f \in \Pi_m^d} \sum_i \|f(x_i) - f_i\| \quad \Pi_m^d : \text{polynomes degree } m \text{ in } d\text{-dimension}$$

WLS

Weighted
Least square

$$\min_{f, \bar{x} \in \Pi_m^d} \sum_i \theta(\|x_i - \bar{x}\|) \|f(x_i) - f_i\| \quad \bar{x}: \text{fixed point}$$

MLS

Moving
Least square

$$\min_{f_x \in \Pi_m^d} \sum_i \theta(\|x_i - x\|) \|f_x(x_i) - f_i\|$$

Moving Least Square Reconstruction

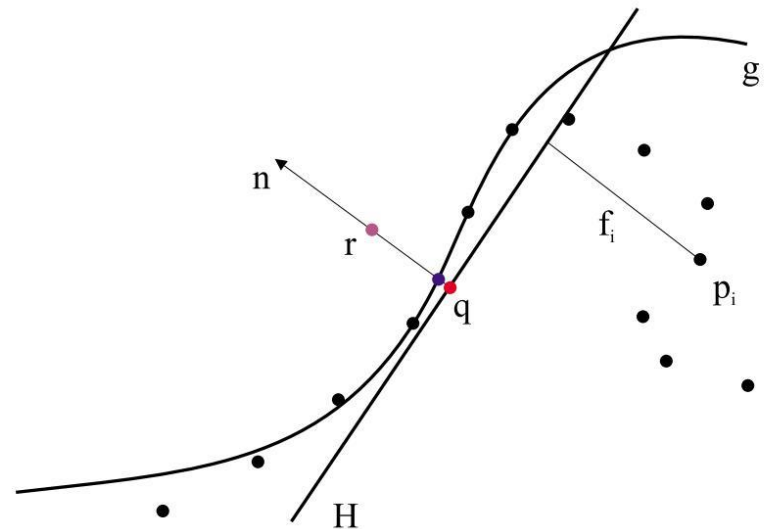
[Alexa01]

- Iterative approach: project the points near the surface onto the surface (??)

1.
$$\min_{n,t} \sum_{i=1}^N \langle n, p_i - r - tn \rangle^2 \theta(\|p_i - r - tn\|)$$

2.
$$\min_g \sum_{i=1}^N (g(x_i, y_i) - f)^2 \theta(\|p_i - q\|)$$

3. Move r to $q + g(0,0) n$



Moving Least Square Reconstruction

[Alexa01]

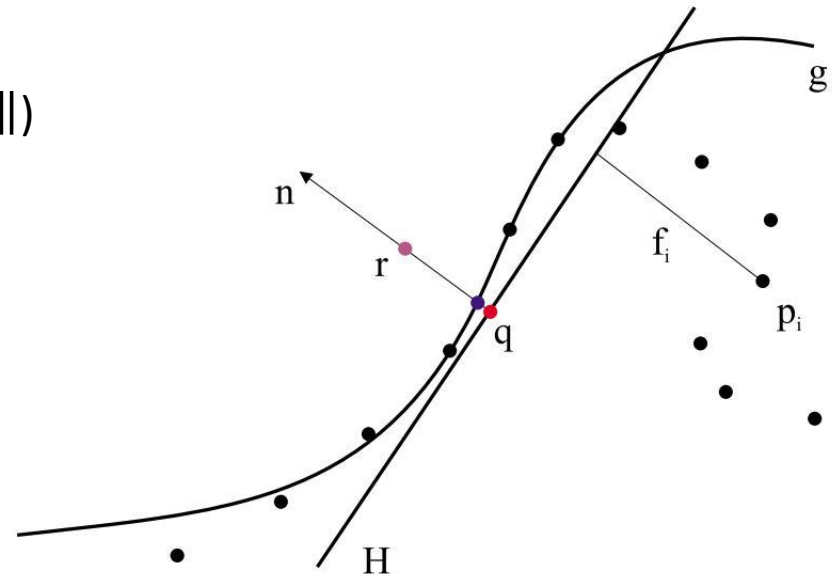
- Iterative approach: project the points near the surface onto the surface (??)

Squared distance between p_i and the plane n, t

1. $\min_{n,t} \sum_{i=1}^N \langle n, p_i - r - tn \rangle^2 \theta(\|p_i - r - tn\|)$ Non linear problem

$$2. \quad \min_g \sum_{i=1}^N (g(x_i, y_i) - f)^2 \theta(\|p_i - q\|)$$

3. Move r to $q + g(0,0) n$



Moving Least Square Reconstruction

[Alexa01]

- Iterative approach: project the points near the surface onto the surface

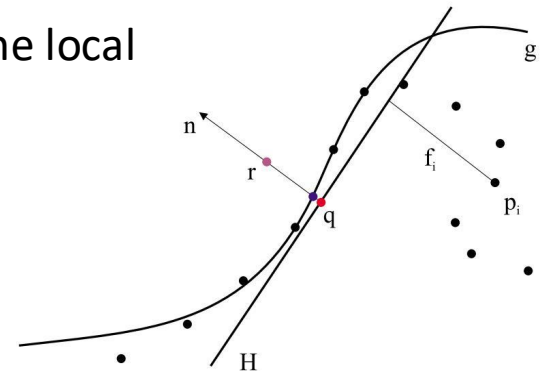
1.
$$\min_{n,t} \sum_{i=1}^N \langle n, p_i - r - tn \rangle^2 \theta(\|p_i - r - tn\|)$$

2.
$$\min_g \sum_{i=1}^N (g(x_i, y_i) - f_i)^2 \theta(\|p_i - q\|)$$
 Known from 1.

$g: \mathbb{R}^2 \Rightarrow \mathbb{R}$ approximates point set in the local reference system centered in q

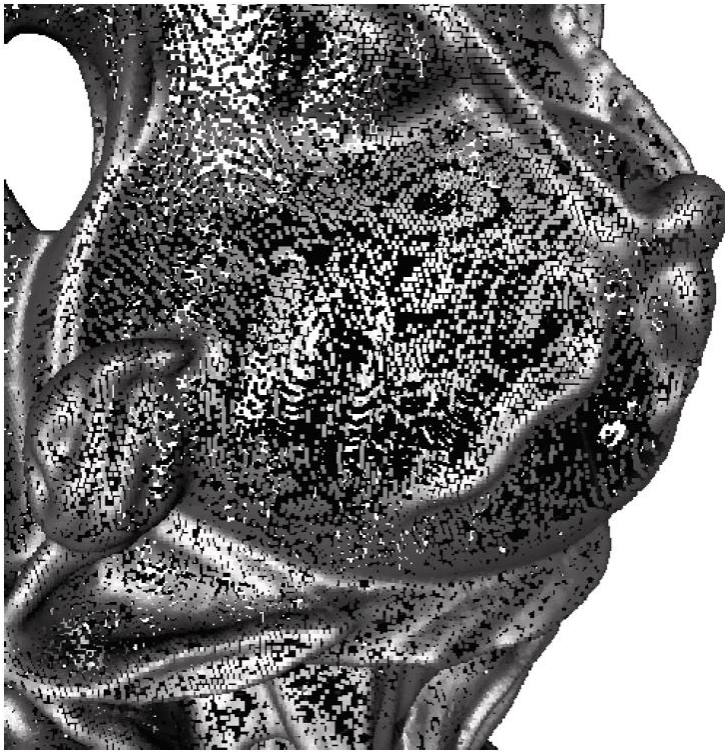
3. Move r to $q + g(0,0) n$

Non linear problem

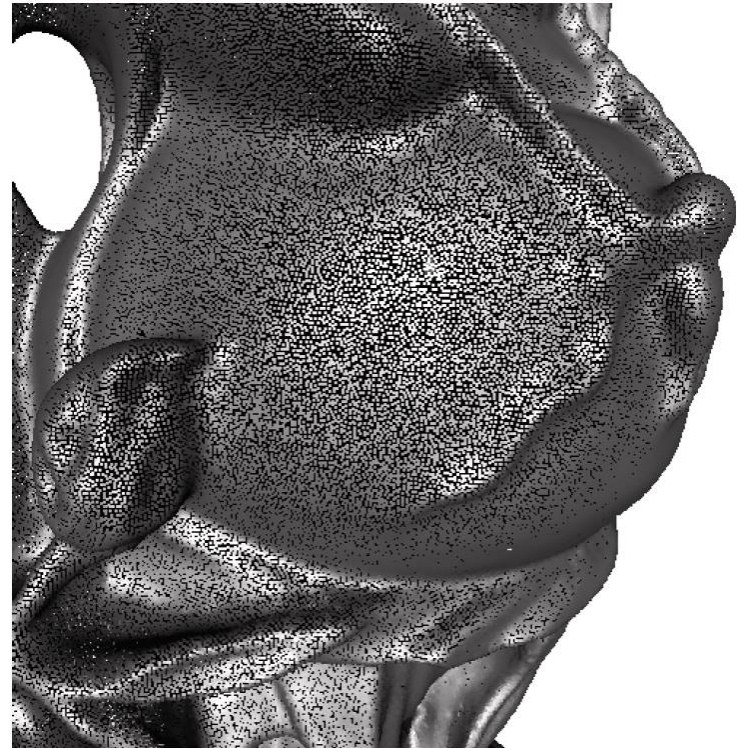


- Repeat 1-3 until stationary point (r projects on itself)

Moving Least Square Reconstruction



Irregular sampling as
acquired by a laser scanner



After MLS reconstruction

Moving Least Square Reconstruction

$$\theta(d) = e^{-\frac{d^2}{h^2}} \quad h \text{ is related to the spacing between samples}$$



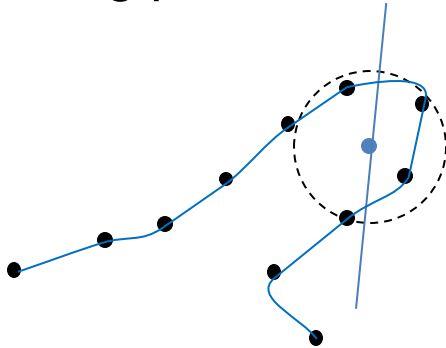
Smaller h



Larger h

Algebraic Point Set Surfaces [Guennebaud07]

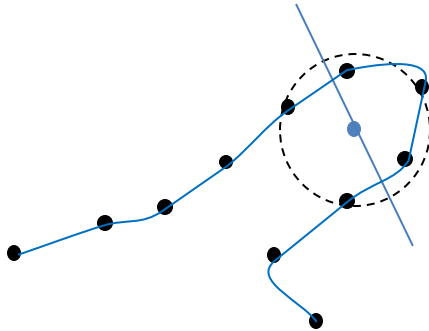
- Plane fitting problem with MLS:



- Nearby position lead to very different planes estimation
- Opposite sheets of surface considered as one

Algebraic Point Set Surfaces [Guennebaud07]

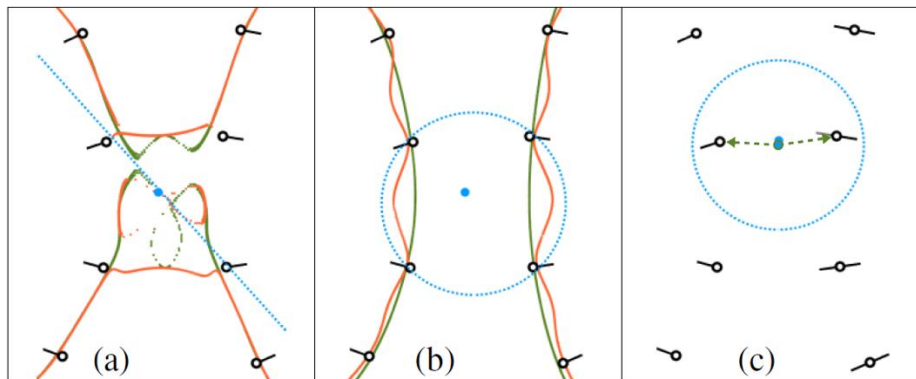
- Plane fitting problem with MLS:



- Nearby position lead to very different planes estimation
- Opposite sheets of surface considered as one

Algebraic Point Set Surfaces [Guennebaud07]

- **Main idea:** fit *spheres* instead of planes
 - Spheres to define normal at the points
 - Spheres to define the surface in the MLS iteration



Plane fitting

Sphere fitting

Algebraic Point Set Surfaces [Guennebaud07]

- Sphere fitting
 - Geometric fitting is unstable for planar configuration
 - Use an algebraic approach, define the surface of the sphere as the zeroes of the function $S_{\mathbf{u}}(\mathbf{x})$:

$$S_{\mathbf{u}}(\mathbf{x}) = [1, \mathbf{x}^T, \mathbf{x}^T \mathbf{x}] \mathbf{u}, \quad \mathbf{u} = [u_0, \dots, u_{d+1}]$$

$$S_{\mathbf{u}}(\mathbf{x}) = u_0 + u_1 x + u_2 y + u_3 z + u_4 (x^2 + y^2 + z^2)$$

$$\text{center} \quad \mathbf{c} = -\frac{1}{2u_4} [u_1, u_2, u_3]^T$$

$$\text{radius} \quad r = \sqrt{\mathbf{c}^T \mathbf{c} - u_0/u_4}$$

- $u_4 = 0 \rightarrow S_{\mathbf{u}}(\mathbf{x}) = 0$ defines a plane

Algebraic Point Set Surfaces [Guennebaud07]

$$\mathbf{W}(\mathbf{x}) = \begin{bmatrix} w_0(\mathbf{x}) \\ \vdots \\ w_{n-1}(\mathbf{x}) \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 1 & \mathbf{p}_0^T & \mathbf{p}_0^T \mathbf{p}_0 \\ \vdots & \vdots & \vdots \\ 1 & \mathbf{p}_{n-1}^T & \mathbf{p}_{n-1}^T \mathbf{p}_{n-1} \end{bmatrix}.$$

Algebraic
sphere fitting in
a neighborhood
of n points

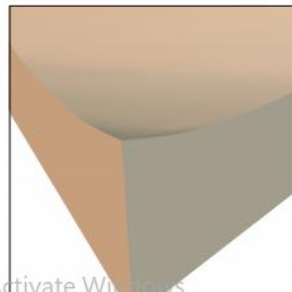
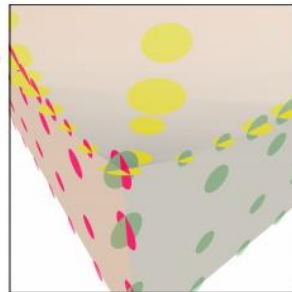
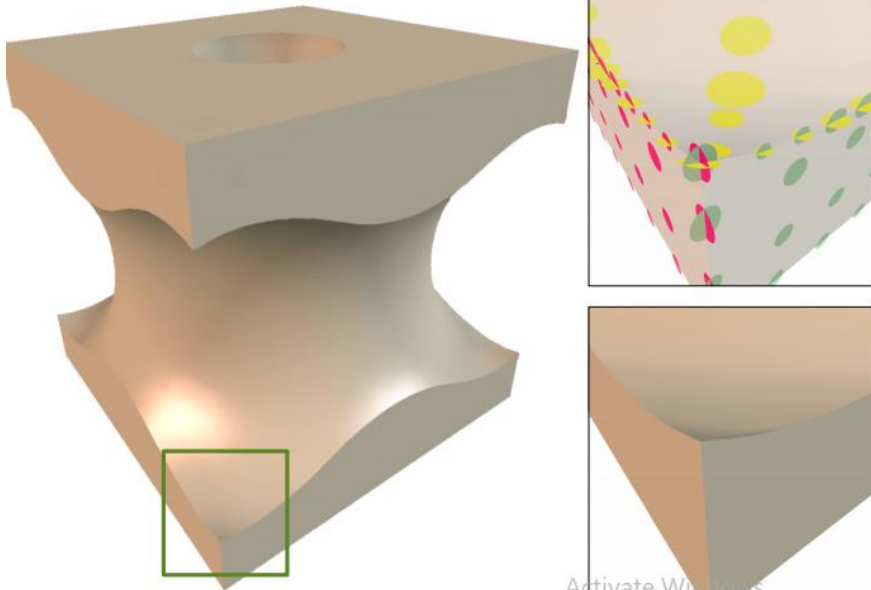
$$\mathbf{u}(\mathbf{x}) = \arg \min_{\mathbf{u}, \mathbf{u} \neq \mathbf{0}} \left\| \mathbf{W}^{\frac{1}{2}}(\mathbf{x}) \mathbf{D} \mathbf{u} \right\|^2$$

Weighting
scheme

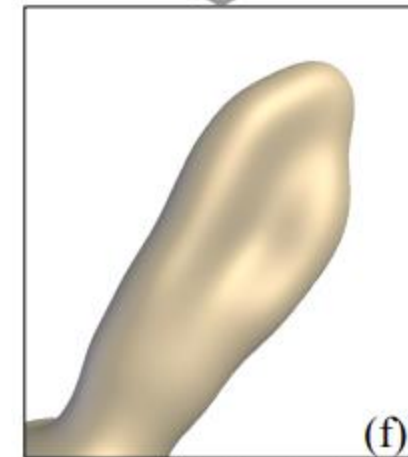
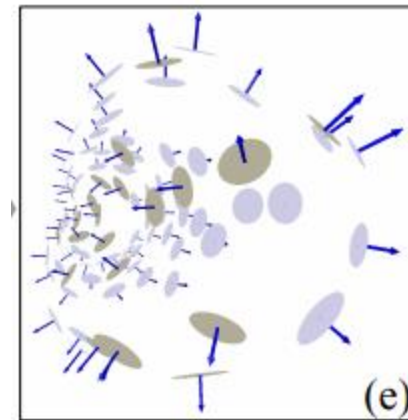
$$w_i(\mathbf{x}) = \phi \left(\frac{\|\mathbf{p}_i - \mathbf{x}\|}{h_i(\mathbf{x})} \right) \quad \leftarrow \text{Sampling radii}$$

$$\phi(x) = \begin{cases} (1 - x^2)^4 & \text{if } x < 1 \\ 0 & \text{otherwise.} \end{cases}$$

Algebraic Point Set Surfaces [Guennebaud07]



Activate Windows
Go to Settings to activate Windows.

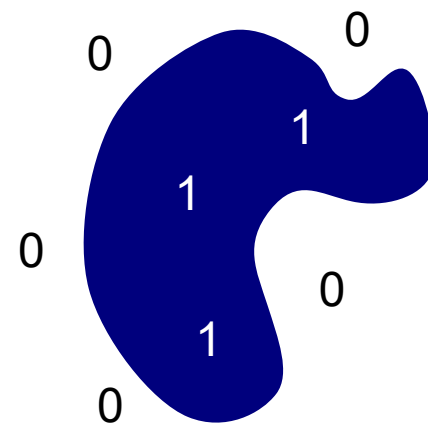


Activate Windows
Go to Settings to activate Windows.

Poisson surface reconstruction

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

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

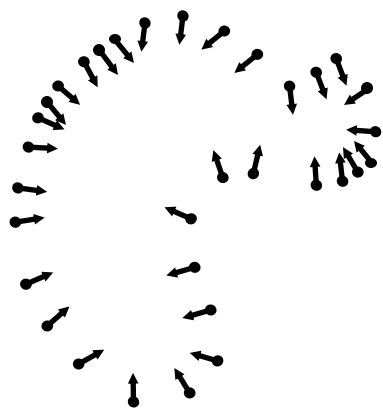


Indicator function

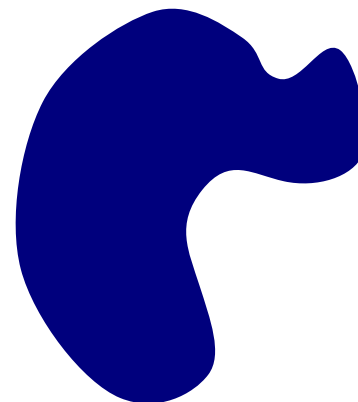
χ_M

Challenge

- How to construct the indicator function?



Oriented points

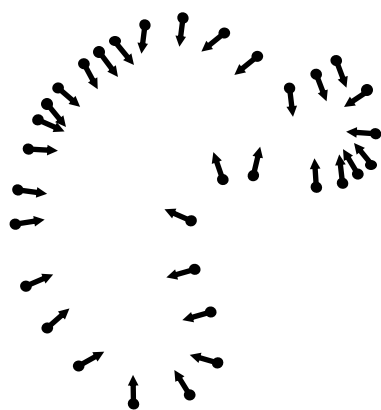


Indicator function

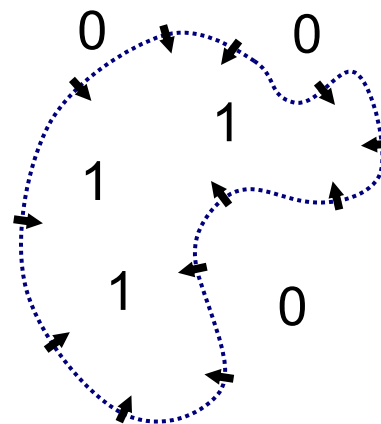
$$\chi_M$$

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 normals 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:

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

Vector field approximation from samples

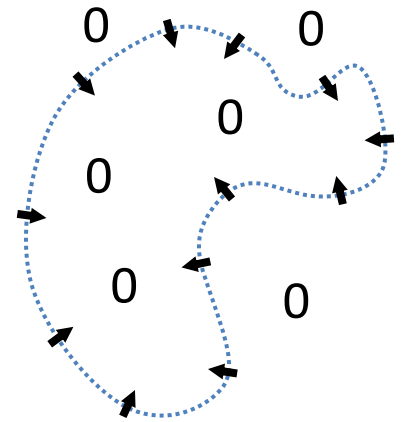
- Note: the indicator function is discontinuous, how can we compute its gradient?
- Smoothing Filter:

Lemma: [kazhdan06]

M manifold, $\vec{N}_{\partial M}(p)$ surface normal,
 $\tilde{F}$ smoothing filter:

$$\tilde{F}_p(q) = \tilde{F}(q - p)$$

$$\nabla(\chi_M * \tilde{F})(q_0) = \int_{\partial M} \tilde{F}_p(q_0) \vec{N}_{\partial M}(p) dp$$



Indicator gradient

$$\nabla \chi_M$$

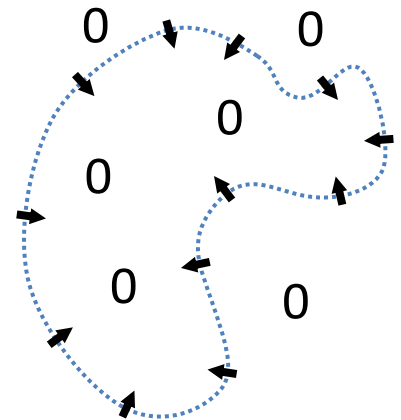
Vector field approximation from samples

- Note: the indicator function is discontinuous, how can we compute its gradient?
- Smoothing Filter:

$$\nabla(\chi_M * \tilde{F})(q_0) = \int_{\partial M} \tilde{F}_p(q_0) \vec{N}_{\partial M}(p) dp =$$

$$\sum_{s \in S} \int_{\mathcal{P}_s} \tilde{F}_p(q) \vec{N}_{\partial M}(p) dp \approx$$

$$\sum_{s \in S} |\mathcal{P}_s| \tilde{F}_{s,p}(q) s. \vec{N} dp \equiv \vec{V}$$



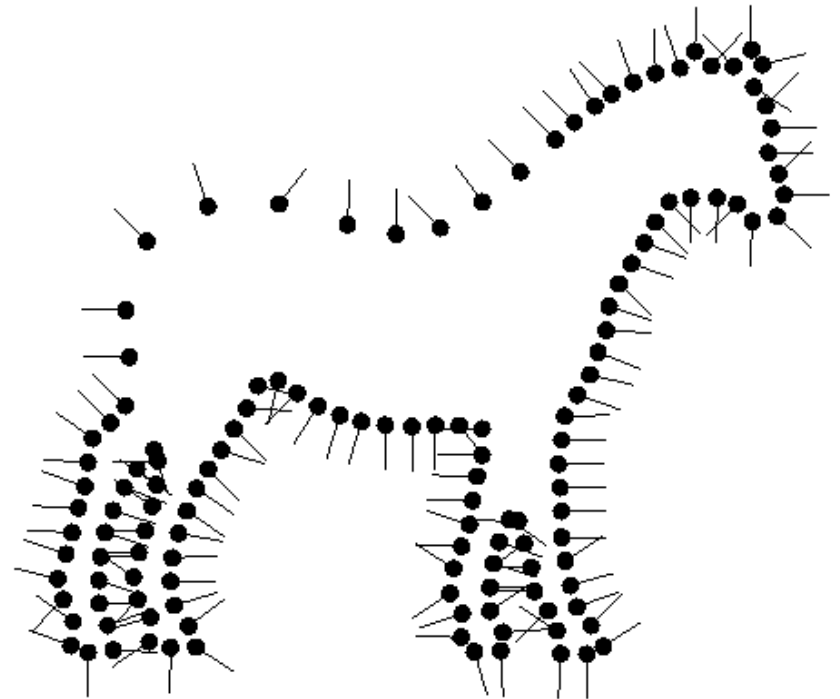
Indicator gradient

$\nabla \chi_M$

Implementation

Given the Points:

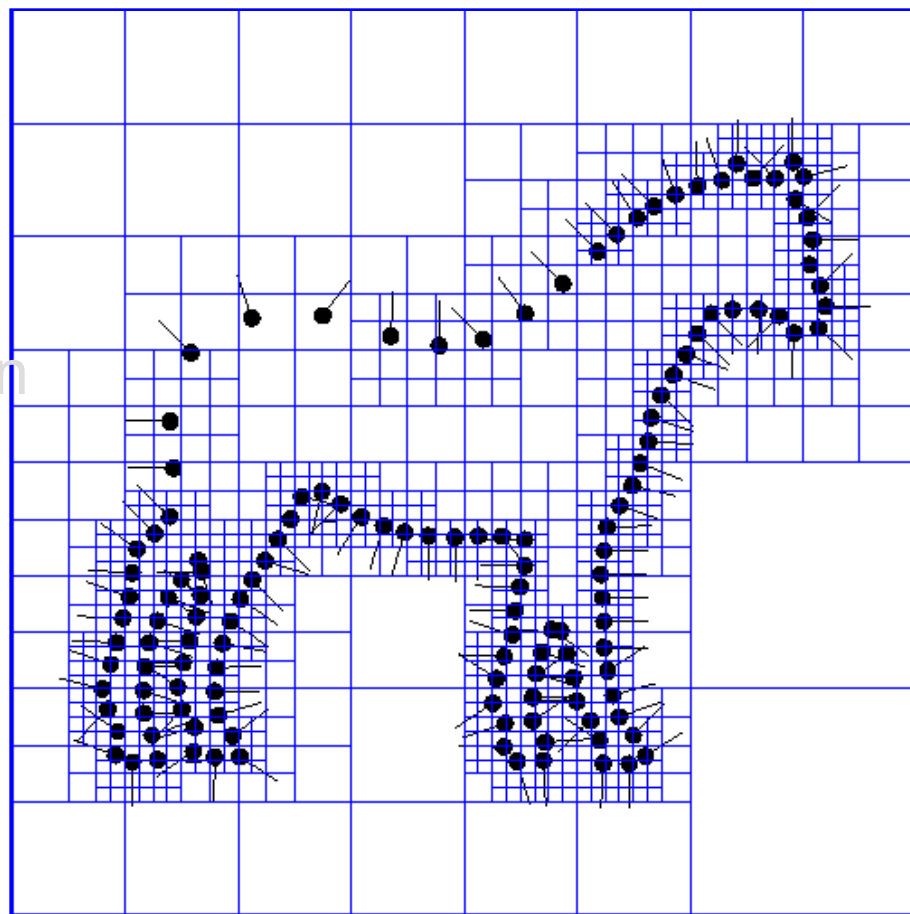
- Set octree
- Compute vector field
- Compute indicator function
- Extract iso-surface



Implementation: Adapted Octree

Given the Points:

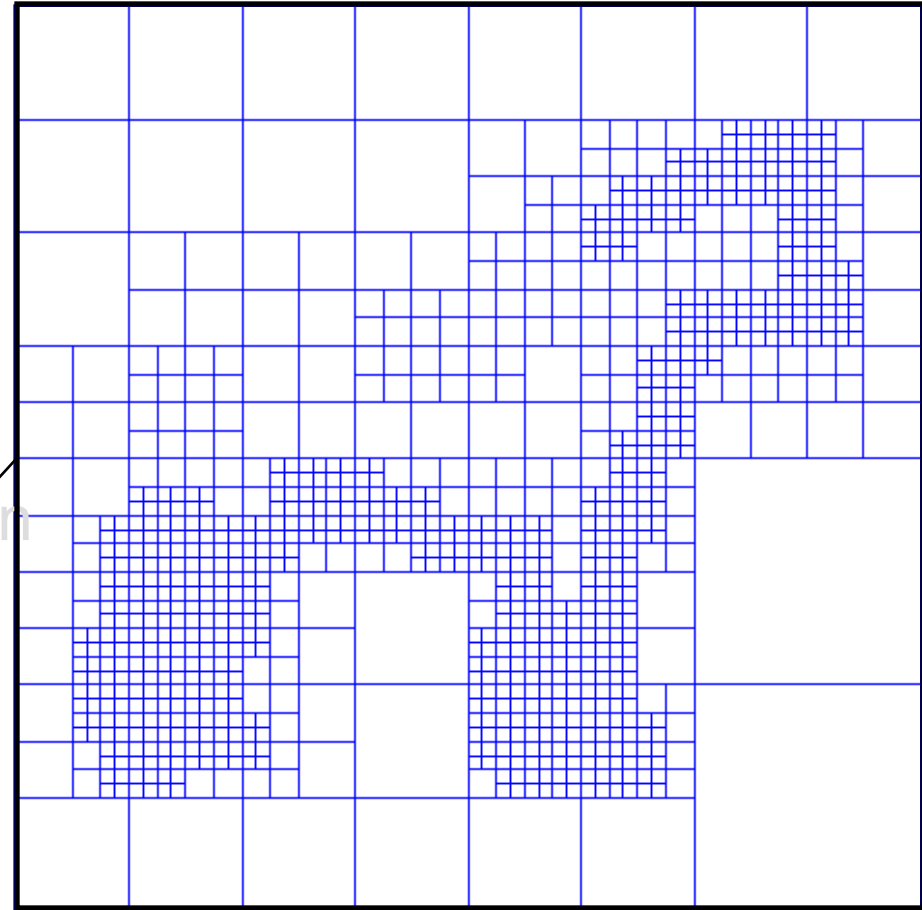
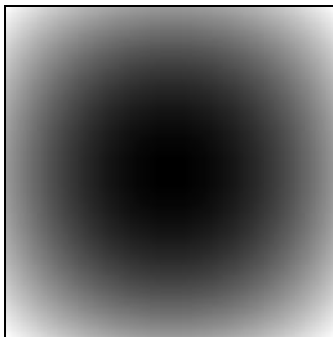
- Set octree
- Compute vector field
- Compute indicator function
- Extract iso-surface



Implementation: Vector Field

Given the Points:

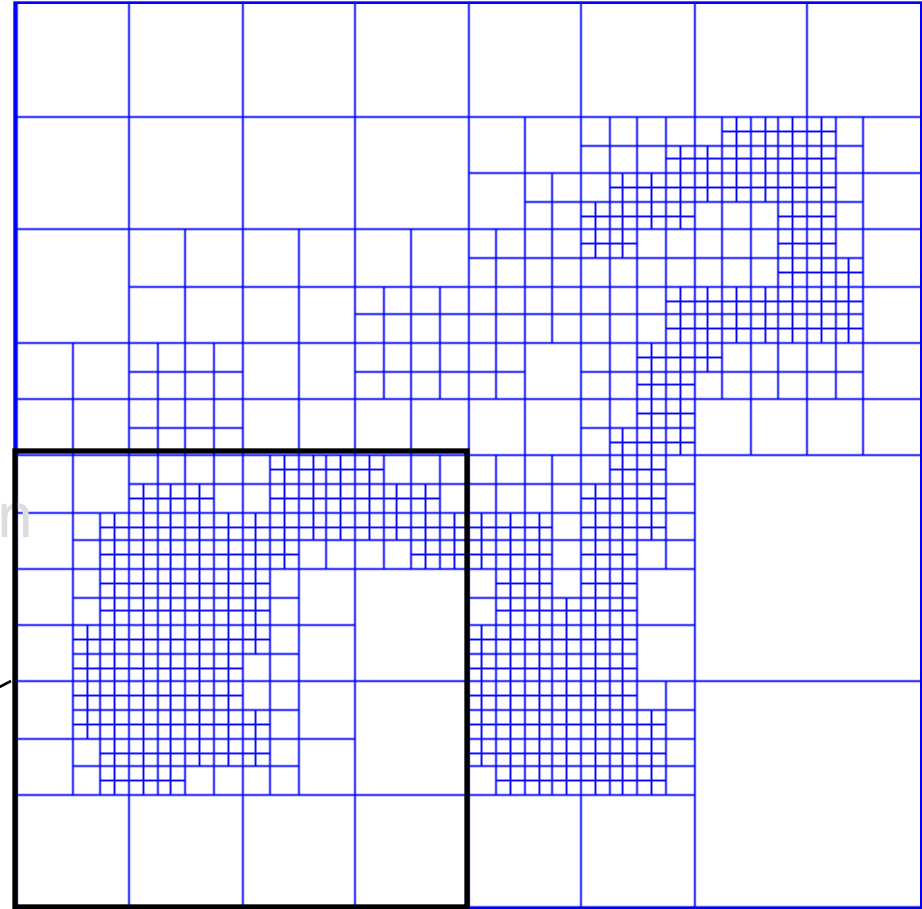
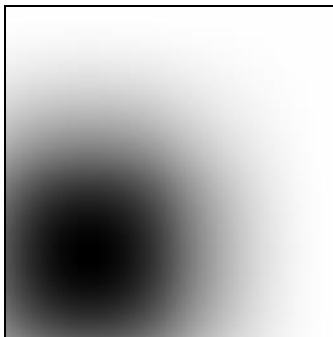
- Set octree
- Compute vector field
 - Define a function space
 - Splat the samples
- Compute indicator function
- Extract iso-surface



Implementation: Vector Field

Given the Points:

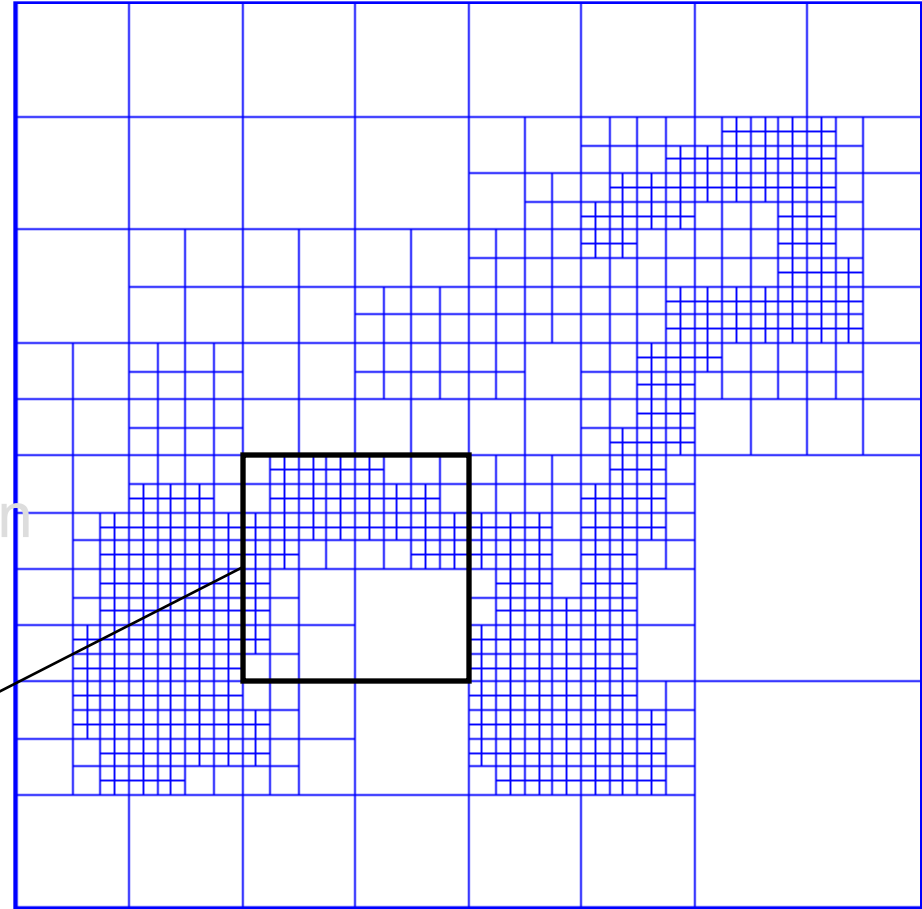
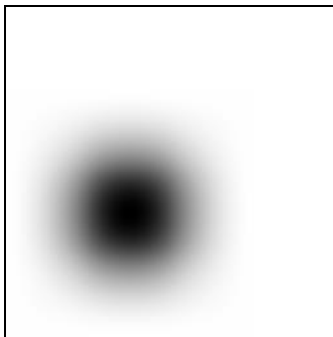
- Set octree
- Compute vector field
 - Define a function space
 - Splat the samples
- Compute indicator function
- Extract iso-surface



Implementation: Vector Field

Given the Points:

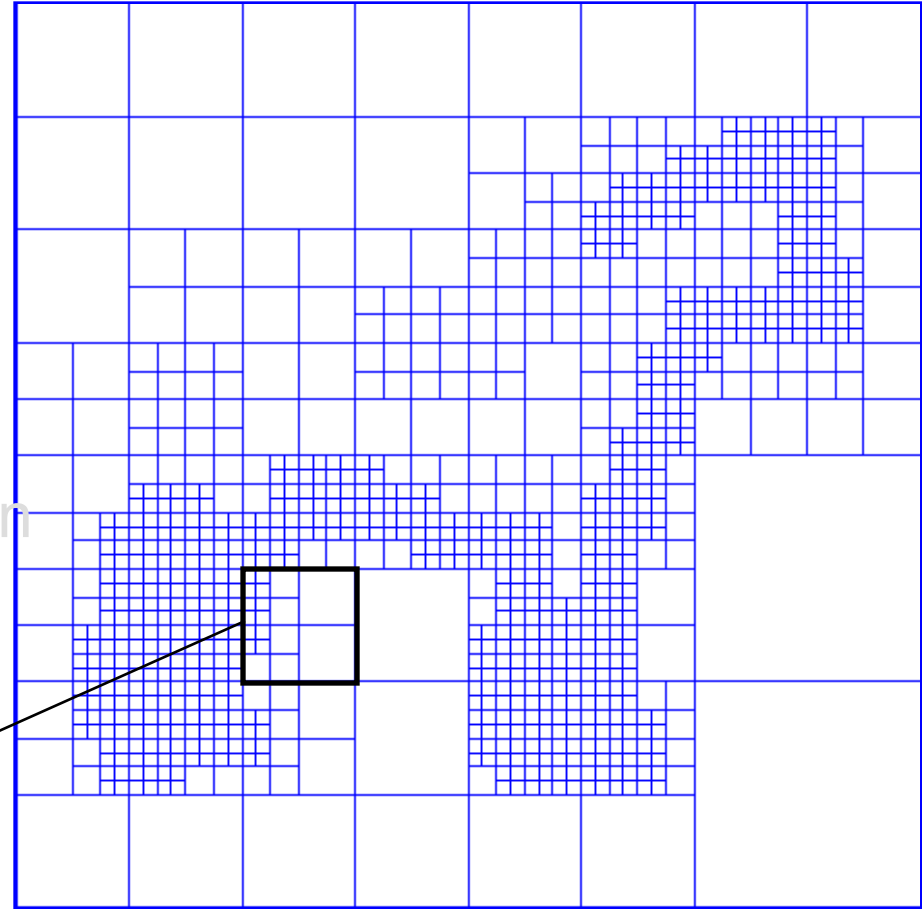
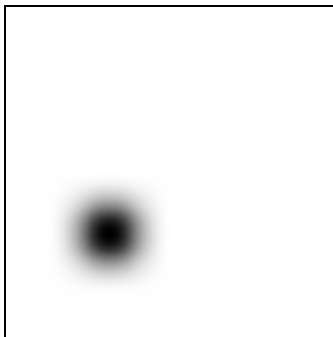
- Set octree
- Compute vector field
 - Define a function space
 - Splat the samples
- Compute indicator function
- Extract iso-surface



Implementation: Vector Field

Given the Points:

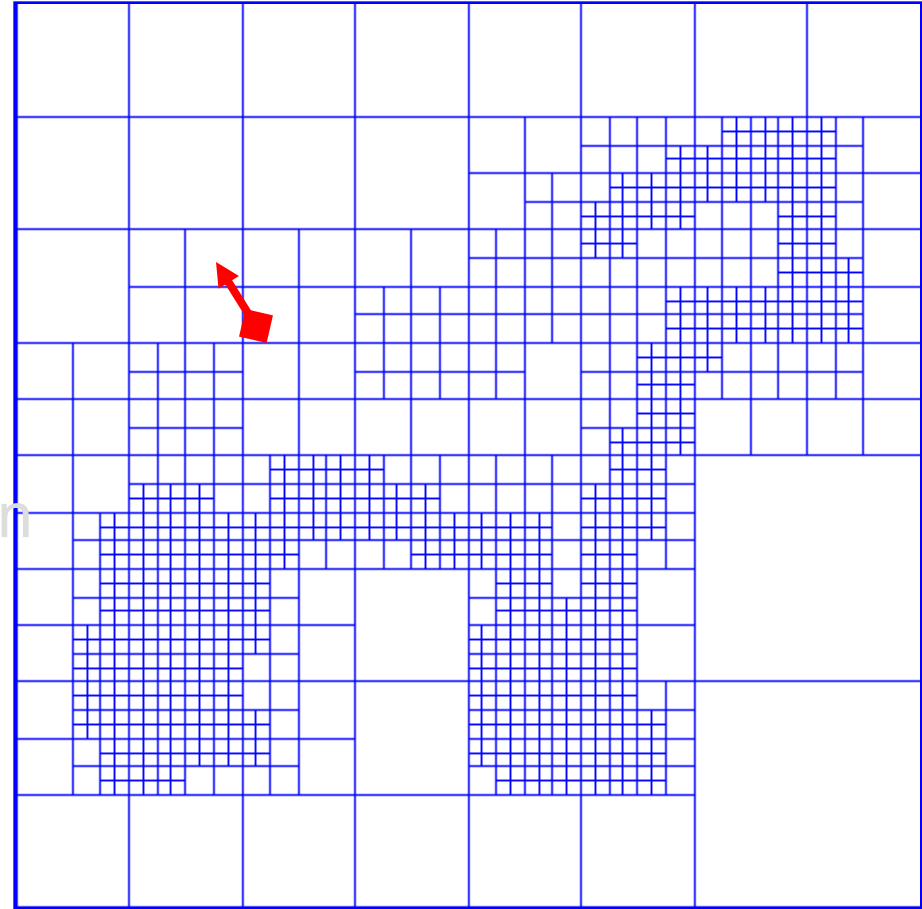
- Set octree
- Compute vector field
 - Define a function space
 - Splat the samples
- Compute indicator function
- Extract iso-surface



Implementation: Vector Field

Given the Points:

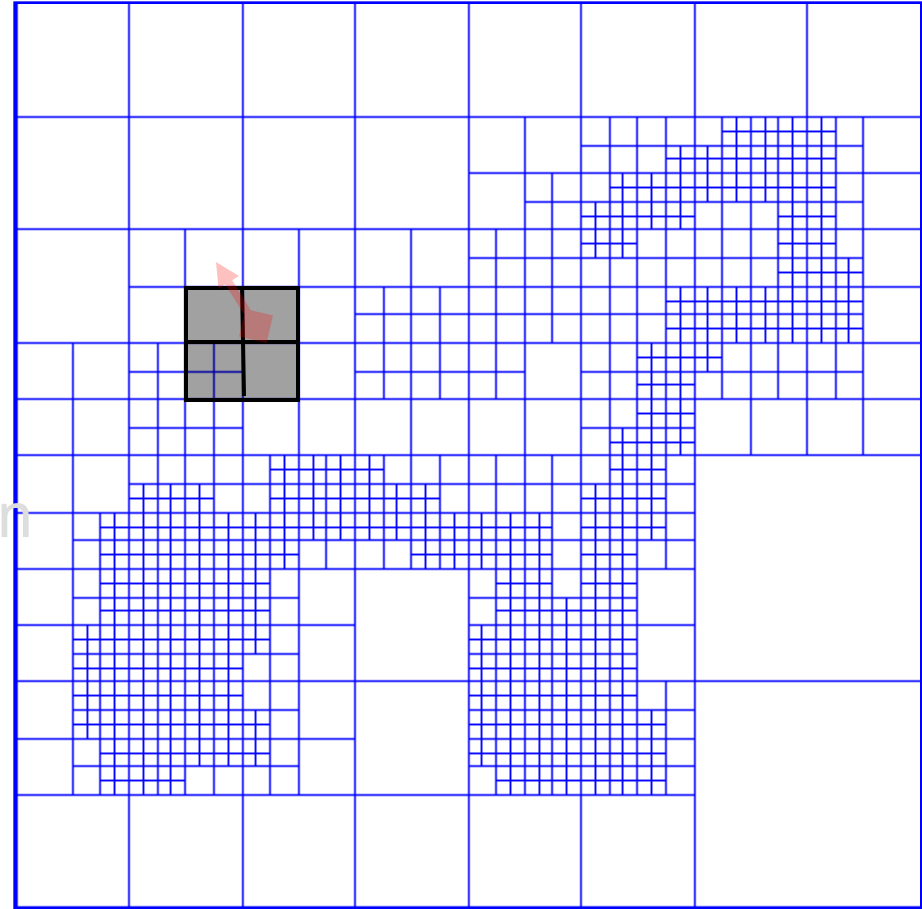
- Set octree
- Compute vector field
 - Define a function basis
 - Splat the samples
- Compute indicator function
- Extract iso-surface



Implementation: Vector Field

Given the Points:

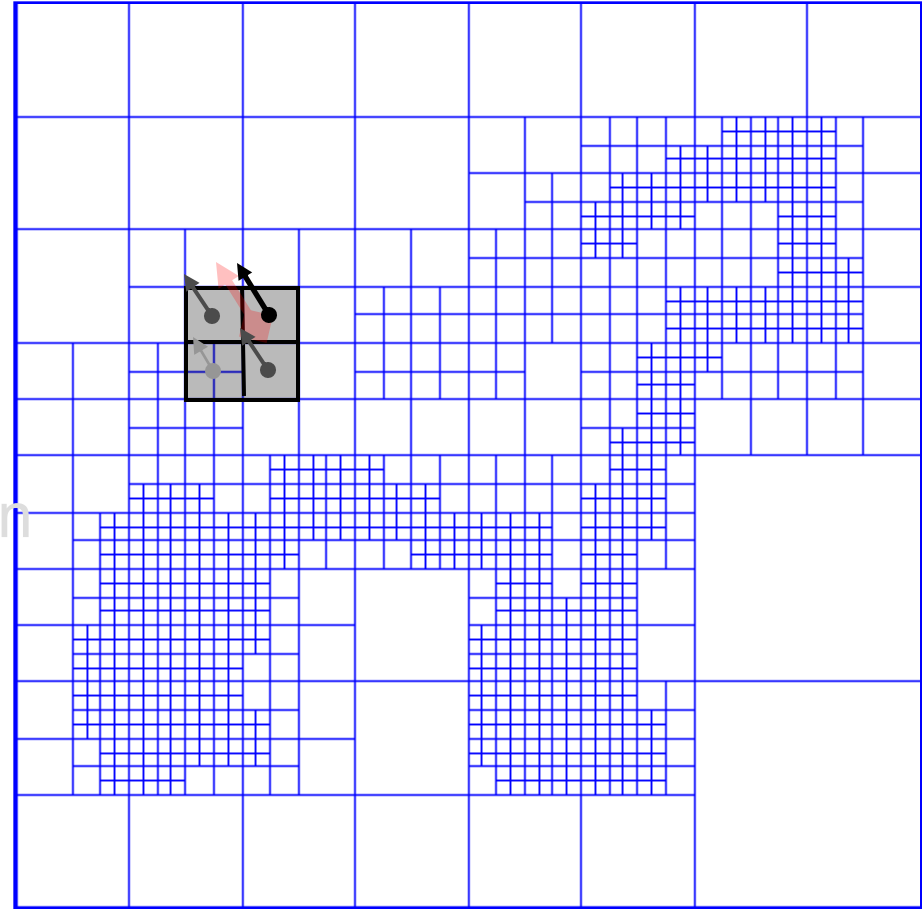
- Set octree
- Compute vector field
 - Define a function basis
 - Splat the samples
- Compute indicator function
- Extract iso-surface



Implementation: Vector Field

Given the Points:

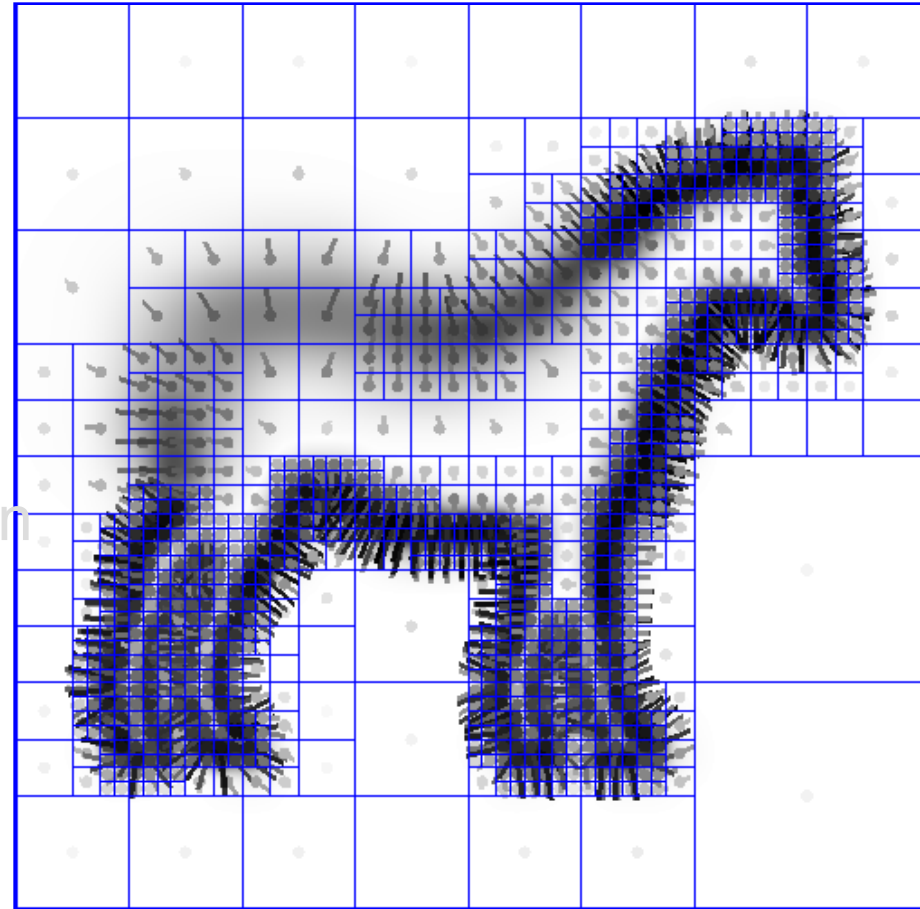
- Set octree
- Compute vector field
 - Define a function basis
 - Splat the samples
- Compute indicator function
- Extract iso-surface



Implementation: Vector Field

Given the Points:

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



Setting up the minimization problem

- So we have defined the vector field

$$\sum_{s \in S} |\mathcal{P}_s| \tilde{F}_{s,p}(q) s \cdot \vec{N} dp \equiv \vec{V}$$

- ...can't we just integrate it and get χ ?
 - No, no guarantees that $\vec{V}$ is curl free

Setting up the minimization problem

- Minimize $|\Delta\chi - \nabla \cdot \vec{V}|$ instead...
- ... More precisely minimize the difference of their projections on the basis of functions F

$$\sum_{o \in \mathcal{O}} \left\| \langle \Delta\tilde{\chi} - \nabla \cdot \vec{V}, F_o \rangle \right\|^2 = \sum_{o \in \mathcal{O}} \left\| \langle \Delta\tilde{\chi}, F_o \rangle - \langle \nabla \cdot \vec{V}, F_o \rangle \right\|^2$$


 All the nodes of
The octree


 The unknown

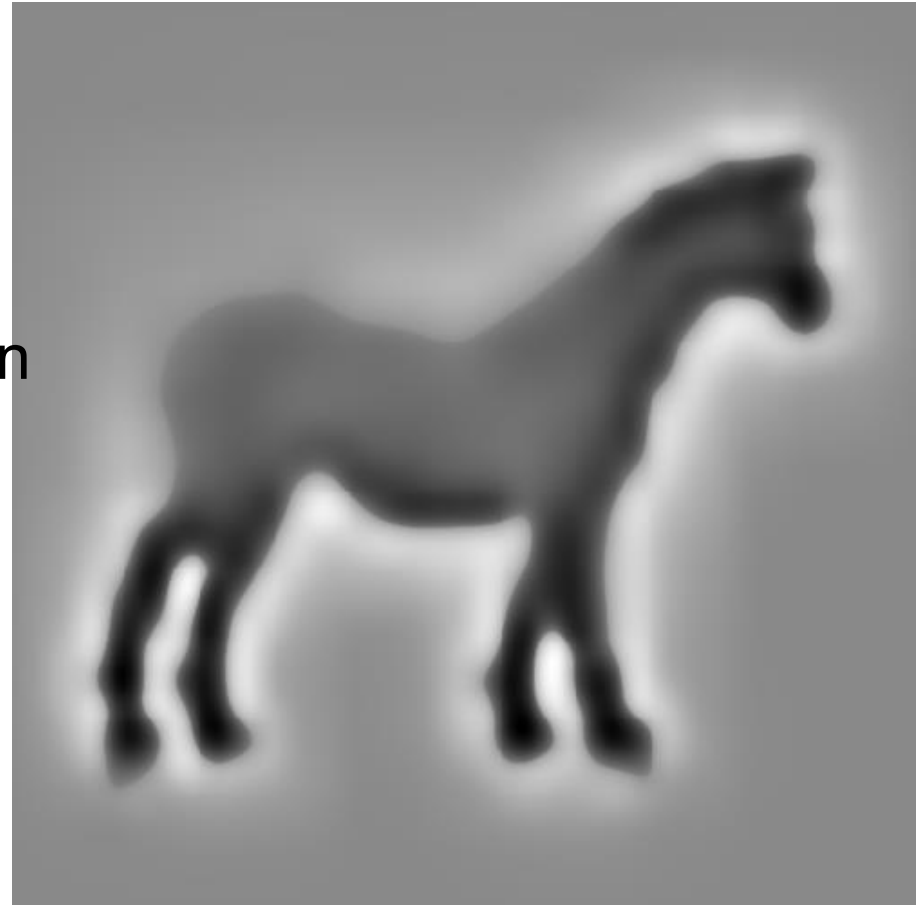
$$\min_{x \in \mathbb{R}^{|\mathcal{O}|}} \|Lx - v\|^2$$


 Coefficients producing χ

Implementation: Indicator Function

Given the Points:

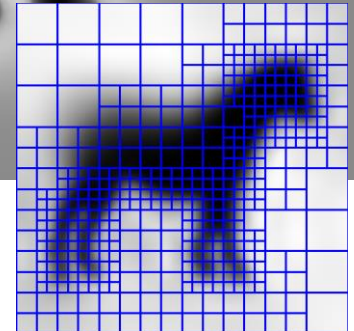
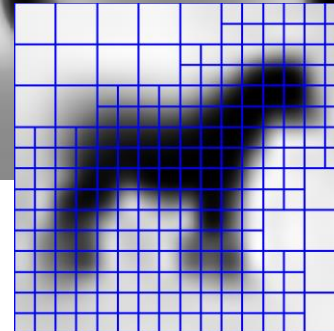
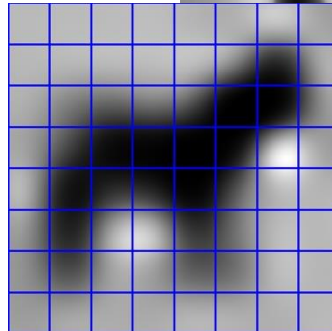
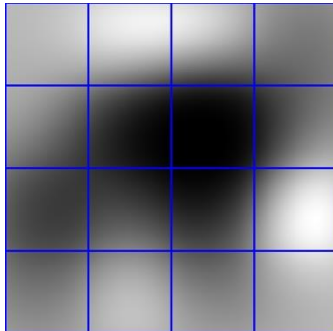
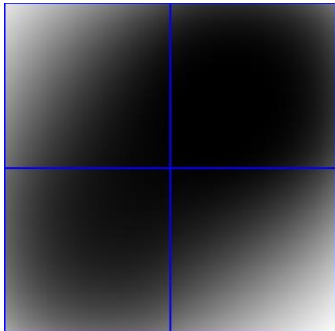
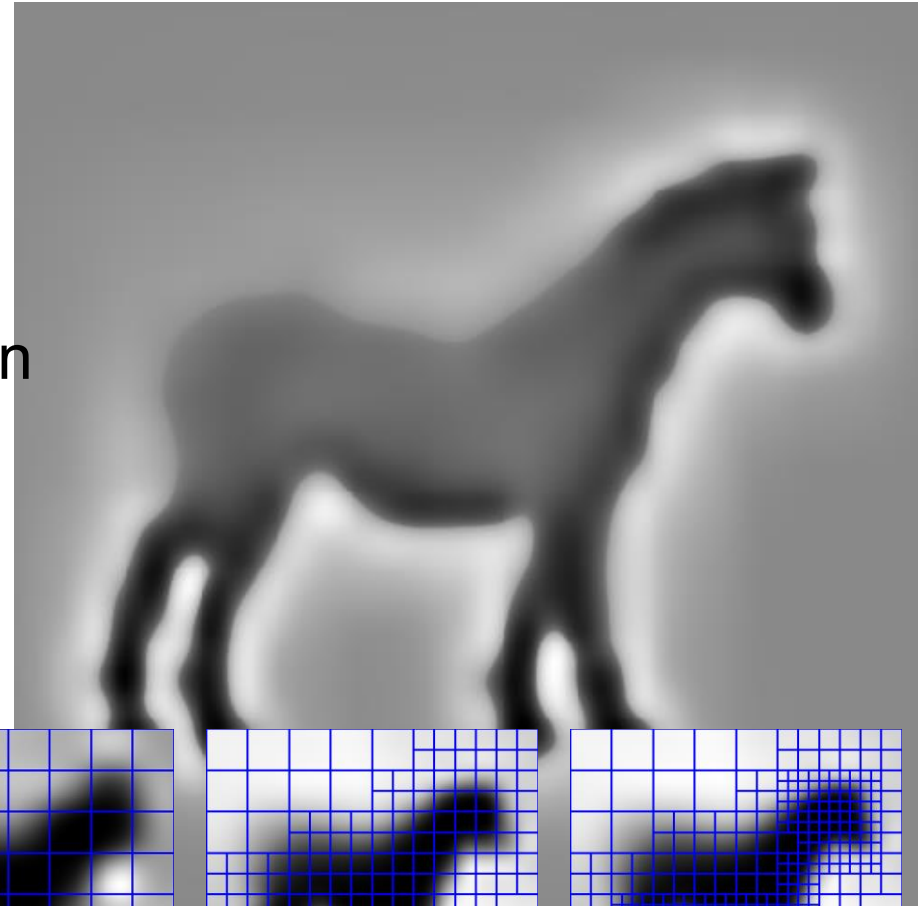
- Set octree
- Compute vector field
- **Compute indicator function**
 - Compute divergence
 - Solve Poisson equation
- Extract iso-surface



Implementation: Indicator Function

Given the Points:

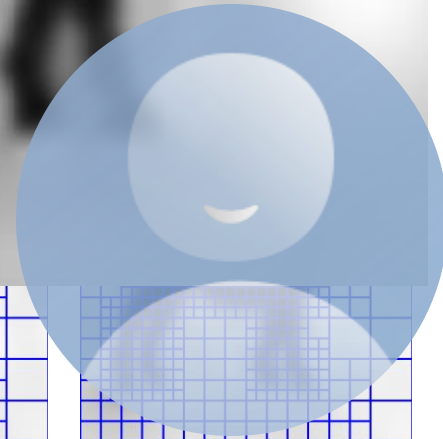
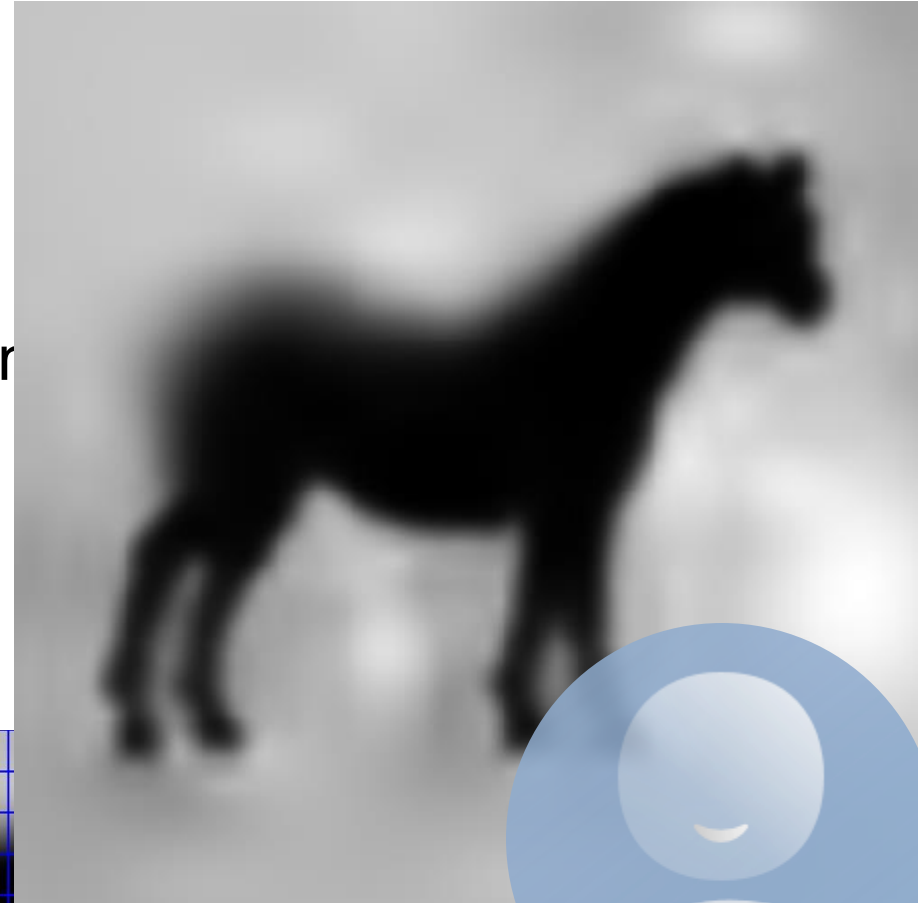
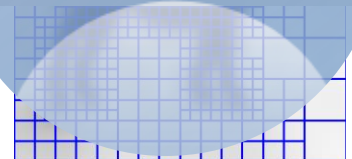
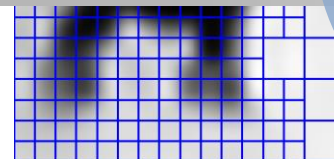
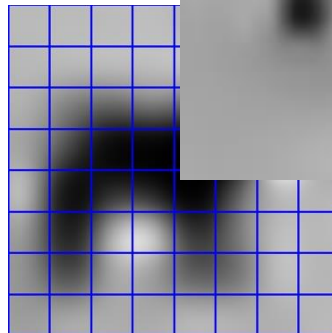
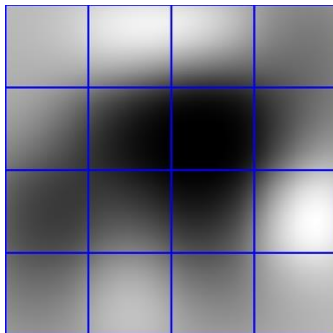
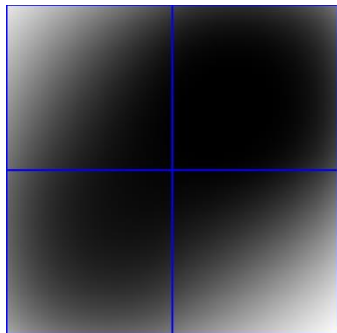
- Set octree
- Compute vector field
- Compute indicator function
 - Compute divergence
 - Solve Poisson equation
- Extract iso-surface



Implementation: Indicator Function

Given the Points:

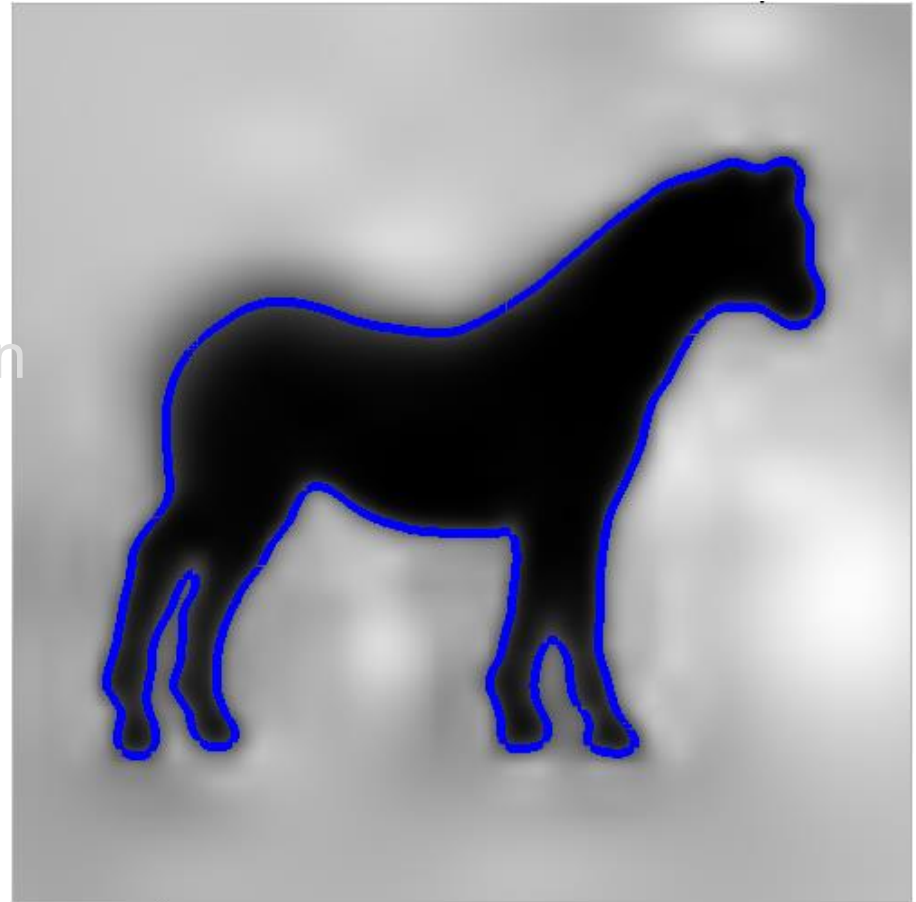
- Set octree
- Compute vector field
- Compute indicator function
 - Compute divergence
 - Solve Poisson equation
- Extract iso-surface



Implementation: Surface Extraction

Given the Points:

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



References

- [turk94] Turk, G. and Levoy, M. (1994). Zippered polygon meshes from range images. ACM Computer Graphics, 28:311-318.
- [Marras10] Controlled and adaptive mesh zippering S.Marras, F. Ganovelli, P. Cignoni, R. Scateni and R. Scopigno, GRAPP 2010
- [Bernardini99] The Ball-Pivoting Algorithm for Surface Reconstruction, Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, Gabriel Taubin IEEE Transactions on Visualization and Computer Graphics archive Volume 5 Issue 4, October 1999 Page 349-359
- [Treece99] G.M. Treece, R.W. Prager, A.H. Gee, Regularised marching tetrahedra: improved iso-surface extraction, Computers & Graphics, Volume 23, Issue 4, 1999
- [Lorensen87] William E. Lorensen, Harvey E. Cline: Marching Cubes: A high resolution 3D surface construction algorithm. In: Computer Graphics, Vol. 21, Nr. 4, July 1987
- [Cignoni00] Reconstruction of topologically correct and adaptive trilinear isosurfaces P Cignoni, F Ganovelli, C Montani, R Scopigno Computers and graphics 24 (3), 399-418
- [Edelsbrunner83] D. G. Kirkpatrick and R. Seidel. On the shape of a set of points in the plane. IEEE Trans. Inform. Theory IT-29 (1983), 551–559.

References

- [Ning93] Paul Ning and Jules Bloomenthal. An evaluation of implicit surface tilers. IEEE Computer Graphics and Applications, 13(6):33-41, 1993.
- [Kobbelt01] Feature sensitive surface extraction from volume data L.P. Kobbelt, M. Botsch, U. Schwaner, Hans-Peter Seidel SIGGRAPH '01 Proceedings of the 28th annual conference on Computer graphics and interactive techniques Pages 57-66
- [Schaefer04] Dual Marching Cubes: Primal Contouring of Dual Grids Scott Schaefer, Joe Warren PG '04: PROCEEDINGS OF THE COMPUTER GRAPHICS AND APPLICATIONS
- [Blinn92] Blinn, J. F. "A Generalization of Algebraic Surface Drawing". ACM Transactions on Graphics 1 (3): 235-256
- [Wyvill86] Data structure for soft objects, Geoff Wyvill, Craig McPheeters, Brian Wyvill The Visual Computer, Vol. 2, No. 4. (1 August 1986), pp. 227-234
- [Carr01] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In Proceedings of ACM SIGGRAPH 2001, pages 67-76, August 2001.
- [Boissonnat84]. Geometric structures for three-dimensional shape representation. ACM Trans. Graph., 3(4):266-286, 1984.

References

- [Morse01] B. S. Morse, T. S. Yoo, D. T. Chen, P. Rheingans, and K. R. Subramanian. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In SMI'01: Proceedings of the International Conference on Shape Modeling & Applications, pages 89-98. IEEE Computer Society, 2001.
- [Dinh01] H. Q. Dinh, G. Turk, and G. Slabaugh. Re-constructing surfaces using anisotropic basis functions. In International Conference on Computer Vision (ICCV) 2001, volume 2, pages 606-613, 2001.
- [Ohtake03] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. ACM Transactions on Graphics, 22(3):463-470, July 2003. Proceedings of SIGGRAPH 2003.
- [Alexa01] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, and C. T. Silva. Point set surfaces. IEEE Visualization 2001, pages 21-28, October 2001.
- [Kazhdan06] Poisson surface reconstruction, Michael Kazhdan, Matthew Bolitho, Hugues Hoppe. Symposium on Geometry Processing 2006, 61-70.
- [Fortune86]. A sweepline algorithm for Voronoi diagrams. Proceedings of the second annual symposium on Computational geometry. Yorktown Heights, New York, United States, pp.313-322. 1986
- M. I. Shamos and D. Hoey, *Closest-point problems*, Proc. 16th Annu. IEEE Sympos. Found. Comput. Sci. (1975), 151-162



[Amenta99]. The crust algorithm for 3D surface reconstruction. In Proceedings of the fifteenth annual symposium on Computational geometry (SCG '99). ACM, New York, NY, USA, 423-424. DOI: <https://doi.org/10.1145/304893.305002>

[Amenta01]. The power crust. In Proceedings of the sixth ACM symposium on Solid modeling and applications (SMA '01), David C. Anderson and Kunwoo Lee (Eds.). ACM, New York, NY, USA, 249-266. DOI=<http://dx.doi.org/10.1145/376957.376986>

[Curless96] Brian Curless and Marc Levoy. 1996. A volumetric method for building complex models from range images. In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH '96). ACM, New York, NY, USA, 303-312. DOI=<http://dx.doi.org/10.1145/237170.237269>