COMPUTER VISION AND PHOTOGRAMMETRY

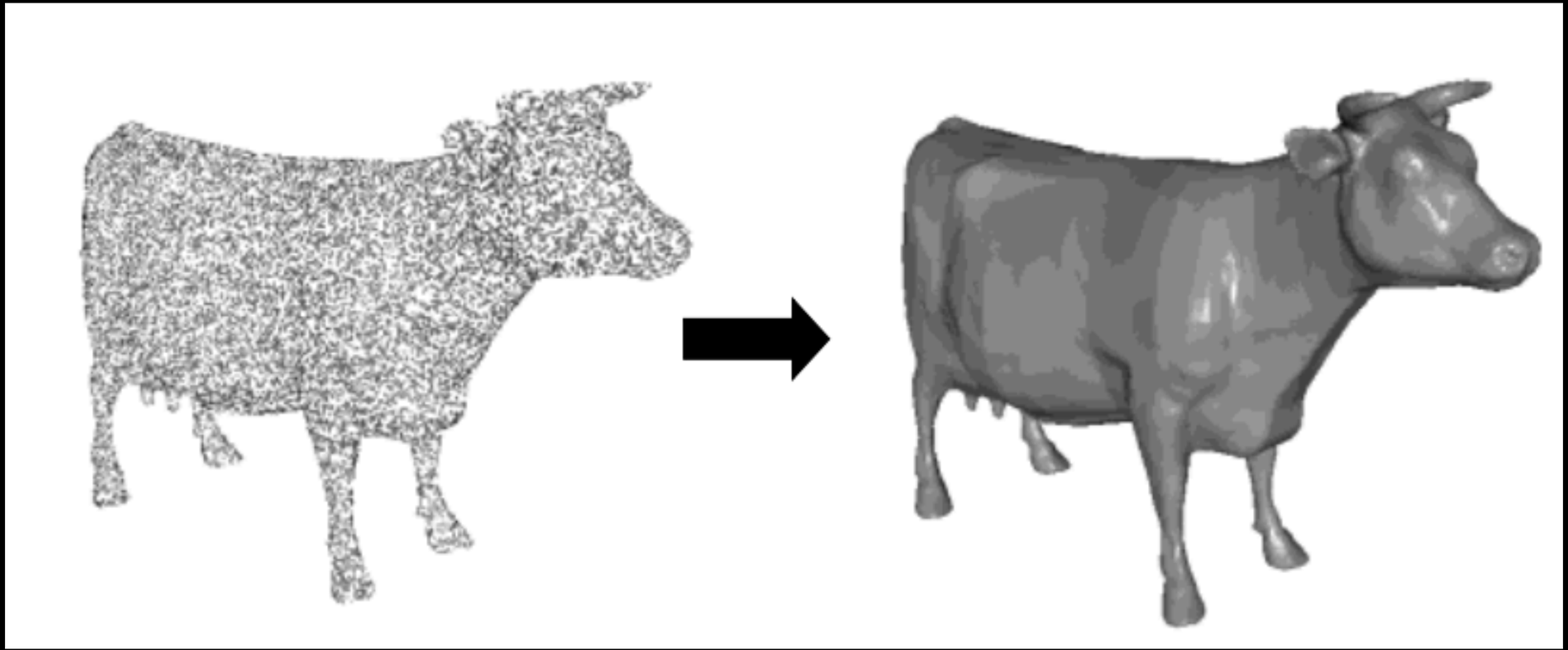
MESHING

CONTENT

- Intro
- Explicit
- Implicit
  - Normal estimation
  - Radial Basis Functions
  - Others
  - Poisson Reconstruction
  - Marching Cubes

PROBLEM FORMULATION AND OVERVIEW

FROM POINT CLOUDS TO MESHES



FROM POINT CLOUDS TO MESHES

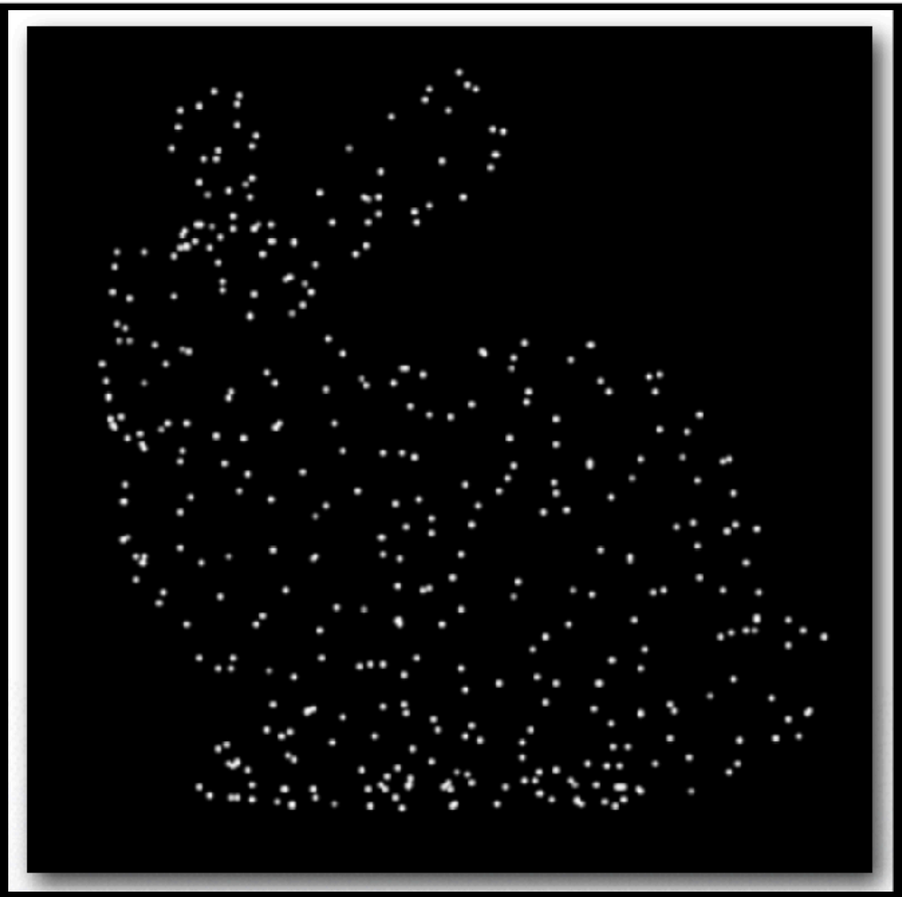
- This is a common problem for scan data
- Range imaging
- Multi-view stereo

INPUT CAN BE

- Organized or not (for us not)
- Oriented (normal information) or not (for us not)
- Non-uniform/sparse
- Noisy

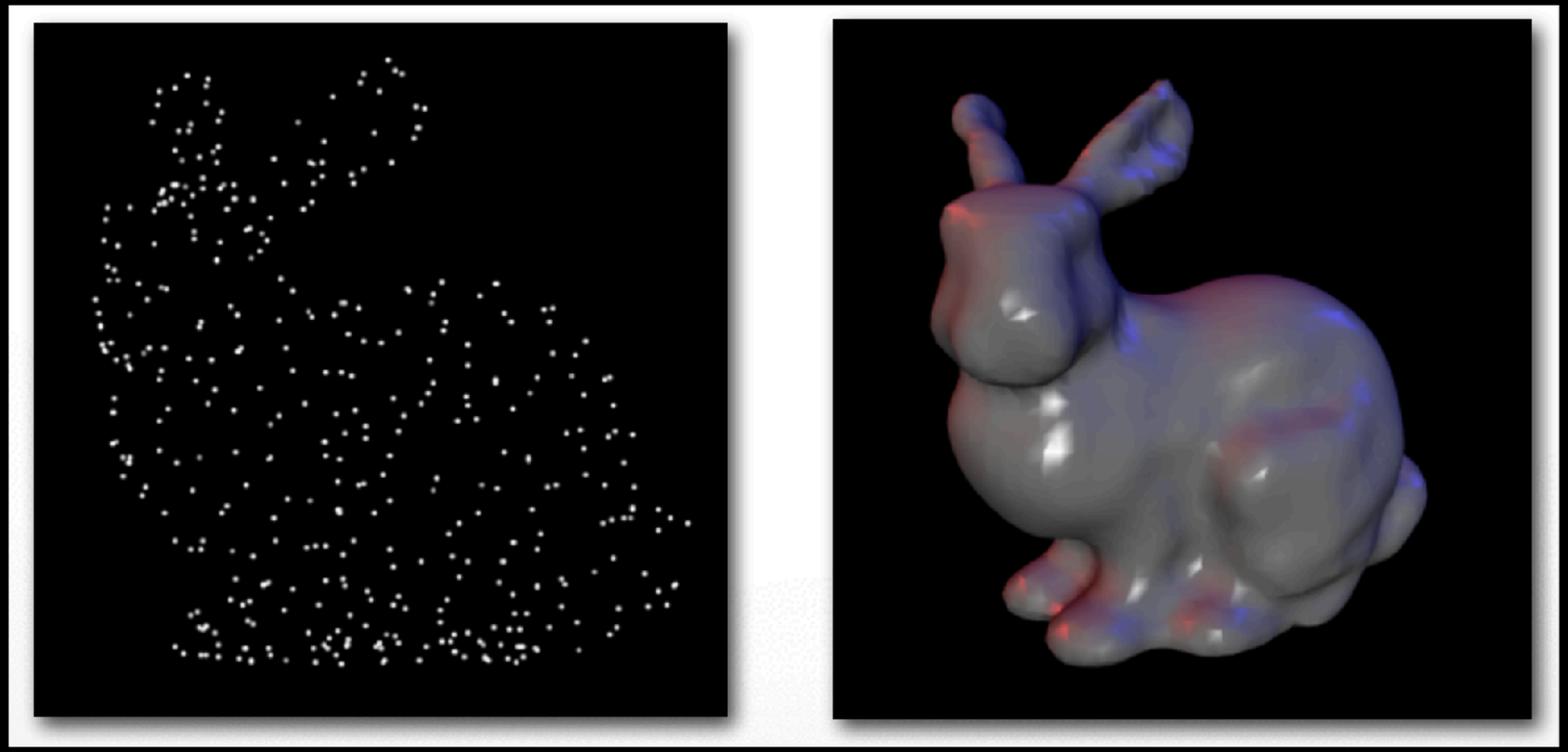
PROBLEM STATEMENT

Given a set of points $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ with $\mathbf{p}_i \in \mathbb{R}^3$



PROBLEM STATEMENT

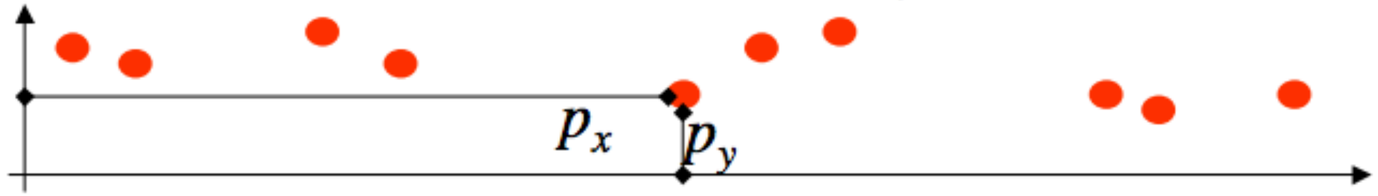
Find a manifold surface $\mathcal{S} \subset \mathbb{R}^3$ which approximates $\mathcal{P}$



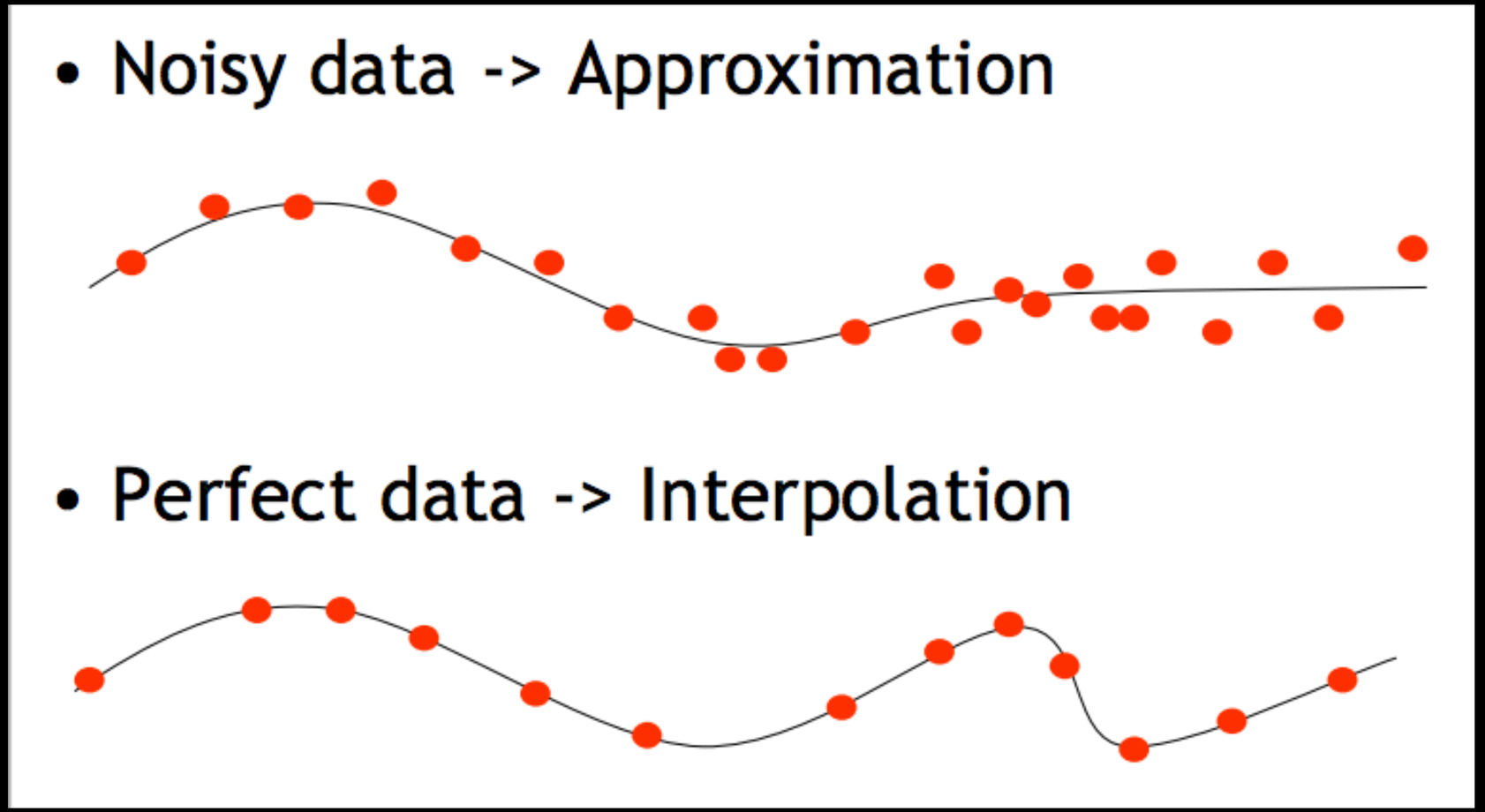
REGULAR / IRREGULAR

- Regular
  - Requires to store only values

- Irregular
  - Requires to store locations p_i

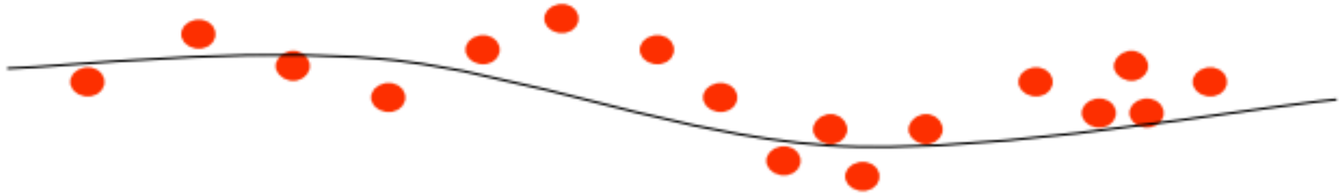


NOISY / PERFECT DATA



GLOBAL / LOCAL

- Global approximation



- Local approximation

- Locality comes at the expense of smoothness

TWO MAIN APPROACHES

- Explicit:
  - Local surface connectivity estimation
  - Point interpolation
- Implicit:
  - Signed distance function estimation
  - Mesh approximation

EXPLICIT RECONSTRUCTION

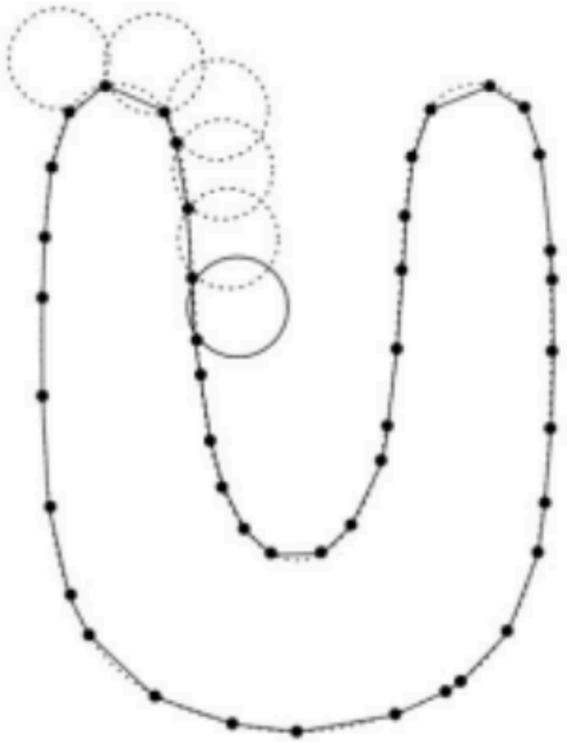
EXPLICIT RECONSTRUCTION

- Connect sample points by triangles
- Exact interpolation of sample points
- Bad for noisy or misaligned data
- Can lead to holes or non-manifold situations

BALL PIVOTING

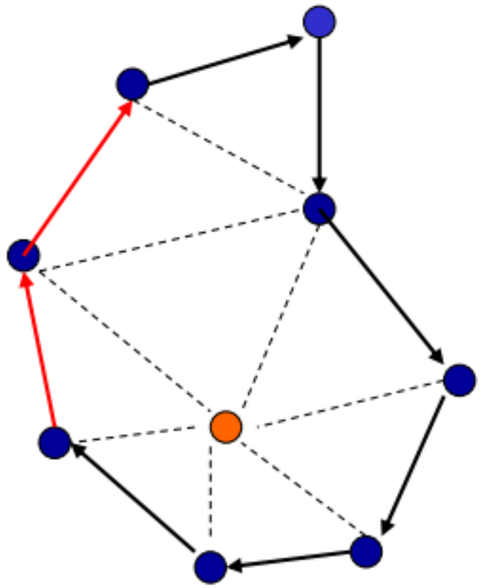
THE BALL PIVOTING ALGORITHM

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



THE BALL PIVOTING ALGORITHM

Boundary edge

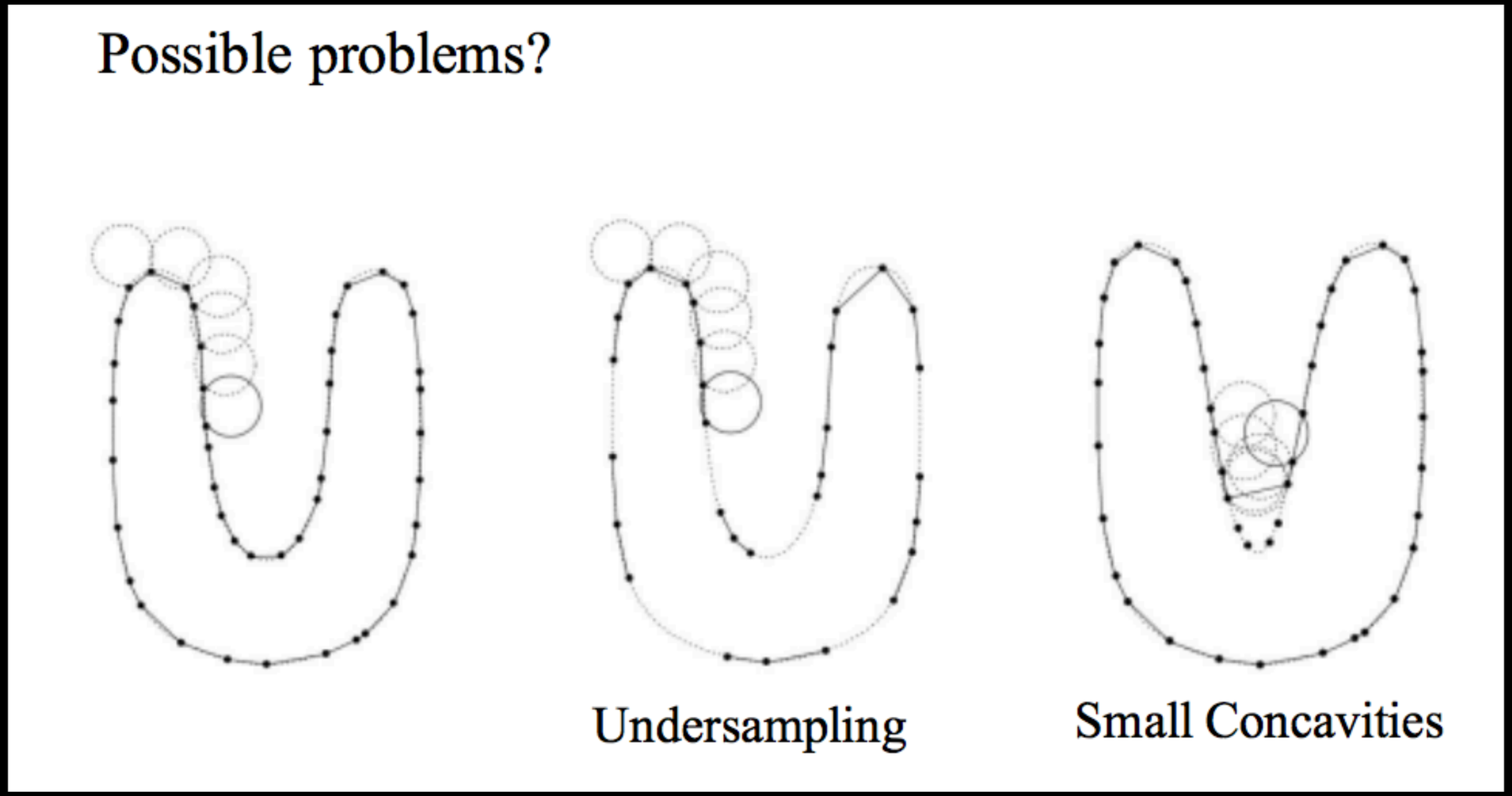


Ball pivoting around active edge

Active edge

- Point on front
- Internal point

THE BALL PIVOTING ALGORITHM



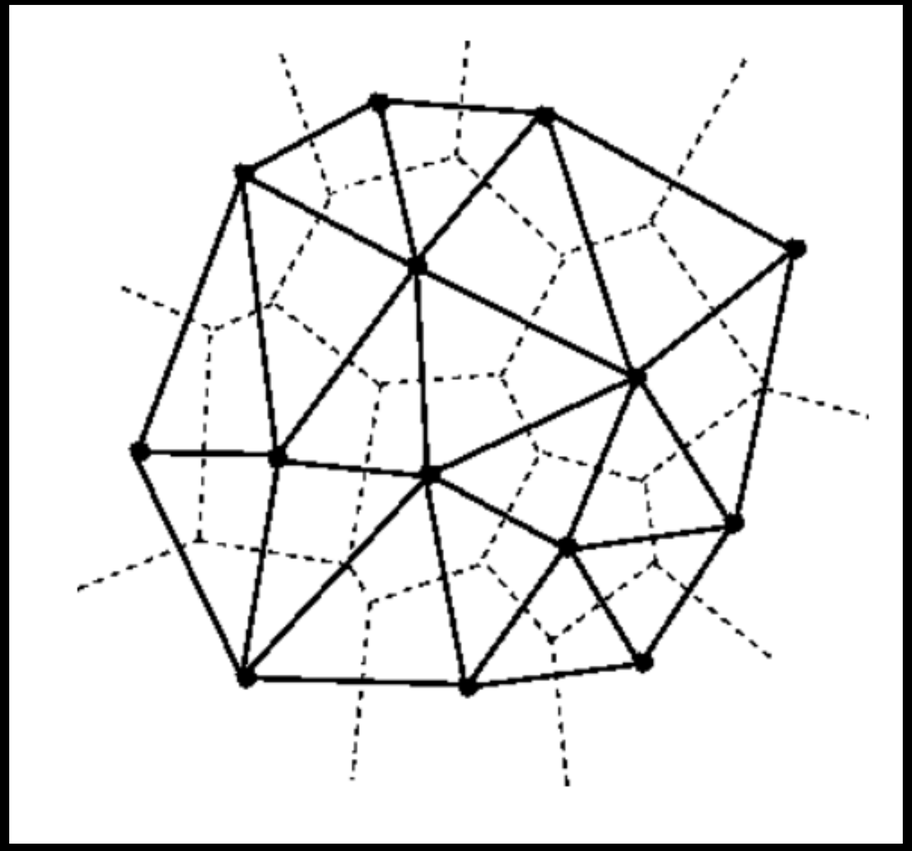
CRUST

CRUST

- Aims to find adjacent surface without a parameter specifying feature sizes

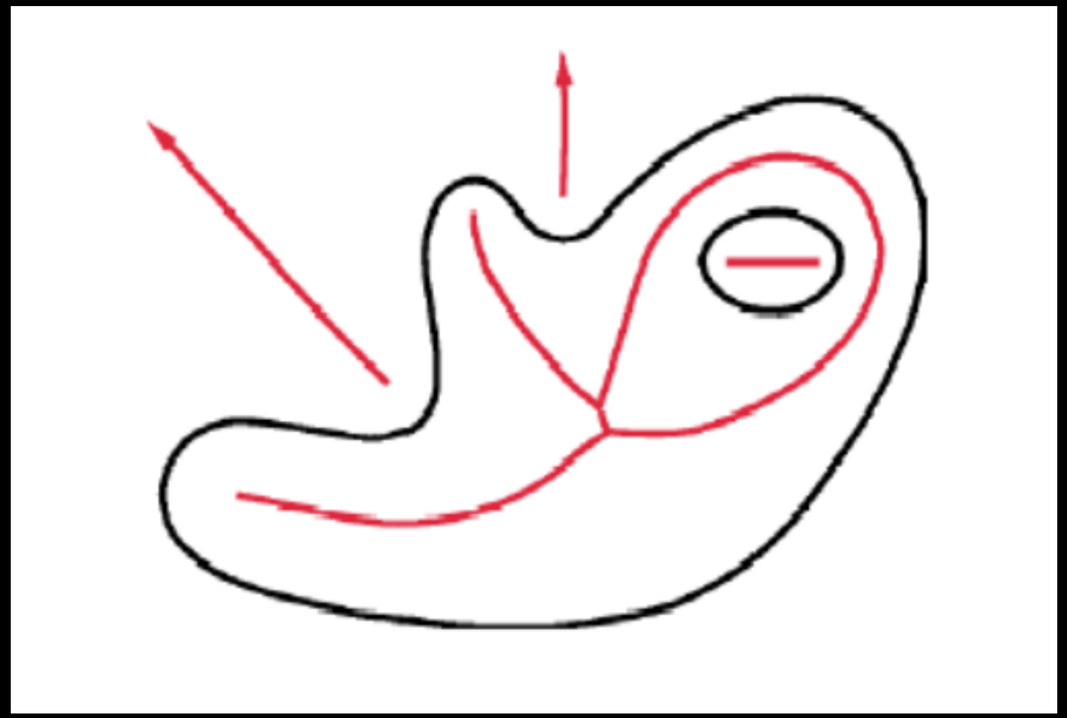
CRUST

- Delaunay Triangulation, Voronoi Diagram



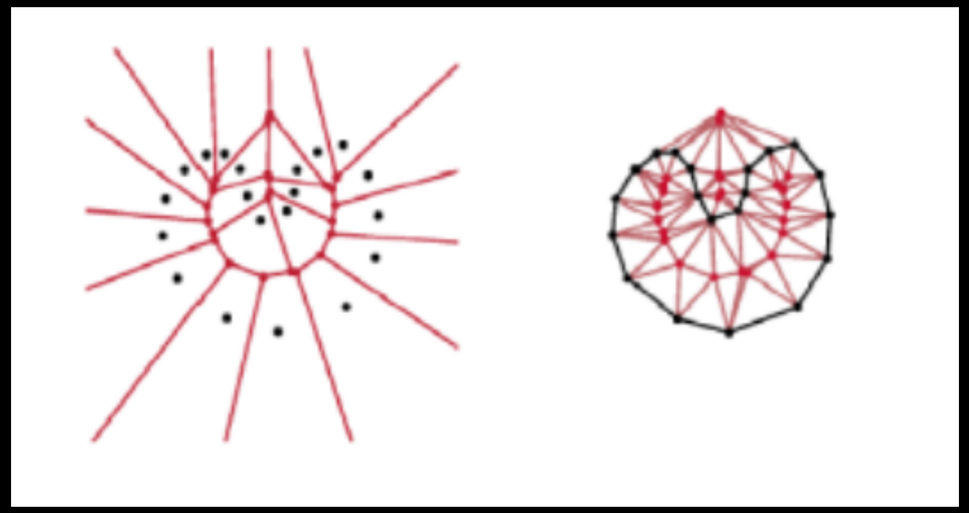
CRUST

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



CRUST

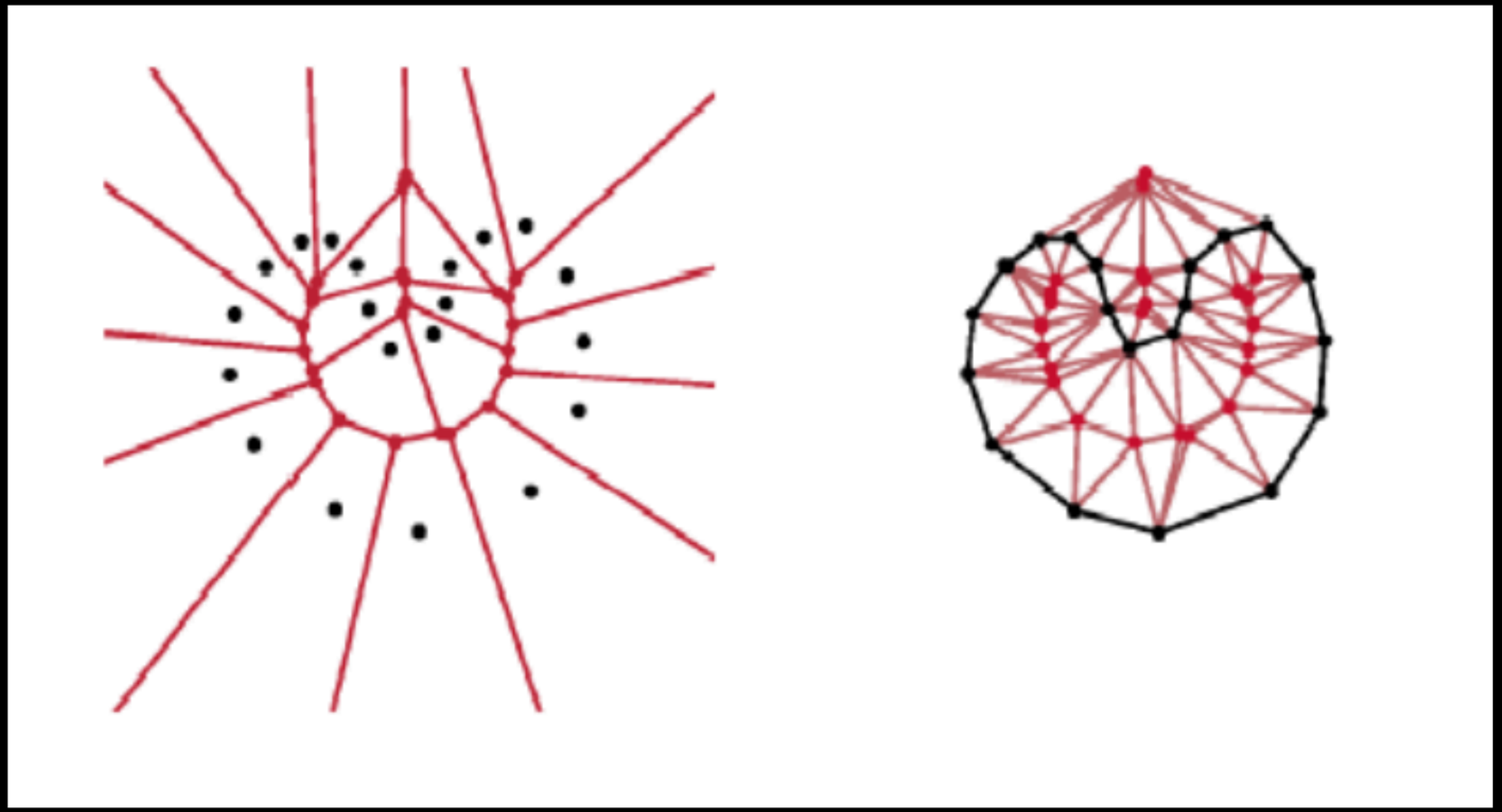
- The Voronoi Cells of a dense sampling are thin and long.
- The Medial Axis is the extension of Voronoi Diagram for continuous surfaces in the sense that the Voronoi Diagram of S Can be defined as the set of points with more than one closest point in S. (S = Sample Point Set)



CRUST IN 2D

- Input : P = Set of sample points in the plane
- Output: E = Set of edges connecting points in P
- The Algorithm
  - Compute the Voronoi vertices of P = V
  - Calculate the Delaunay of (P U V)
  - Pick the edges (p,q) where both p,q are in P

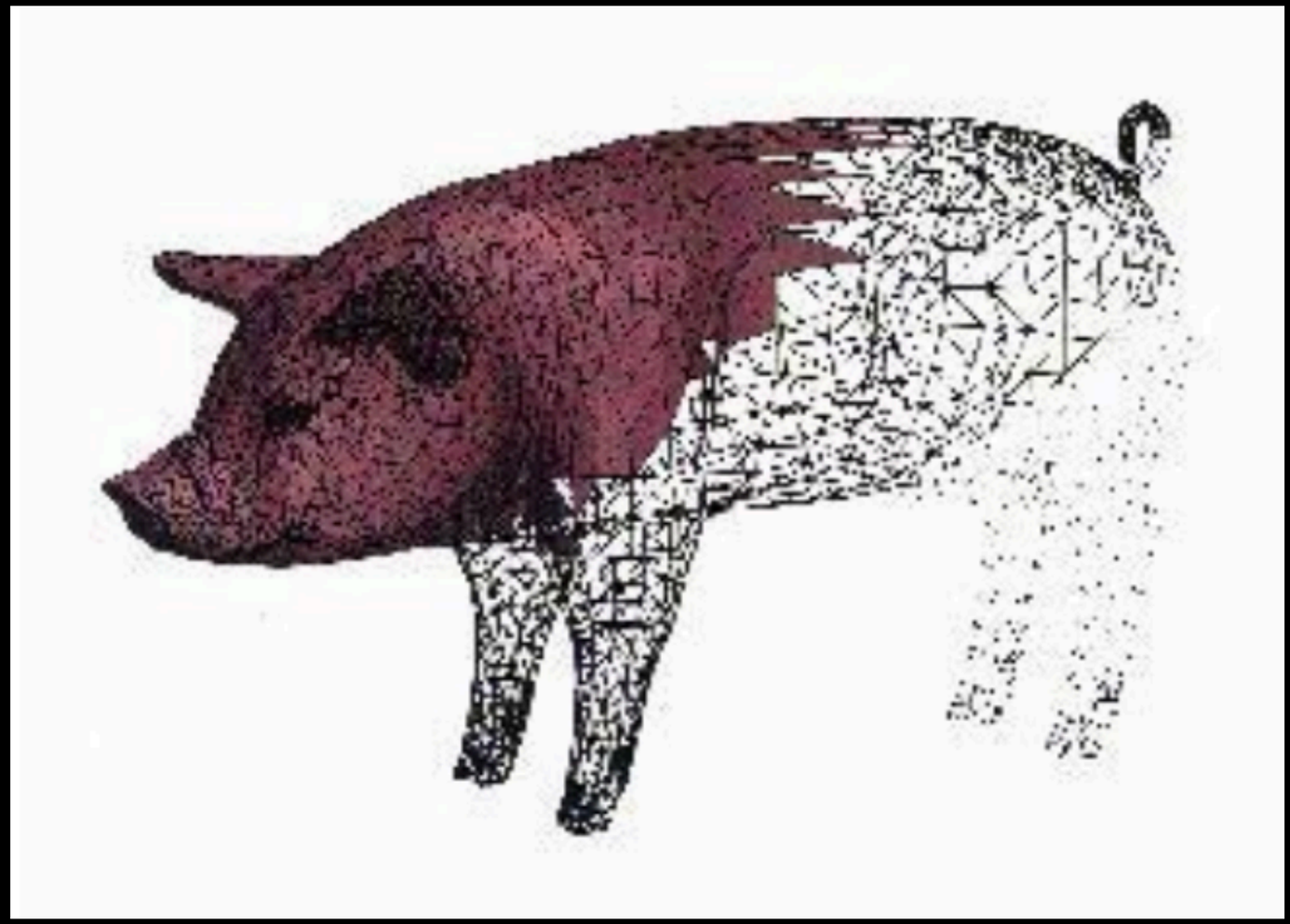
CRUST IN 2D: OUTPUT



CRUST IN 3D

- http://web.mit.edu/manoli/crust/www/sigcrust.pdf
- http://www.cgal.org/

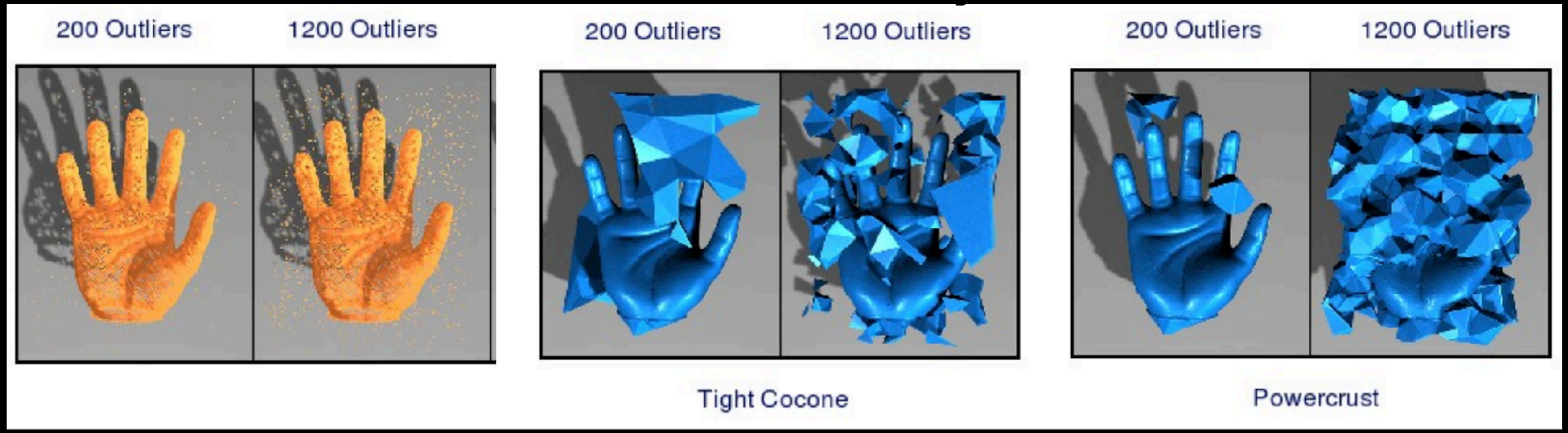
CRUST: SAMPLE OUTPUT



DELAUNAY TRIANGULATION

- More than 50% of existing methods are basedon Delaunay triangulation and Voronoi diagram
- Output mesh size ~ input point cloud size
- Known and uniform sampling => very accurate results
- Noise or outliers => usually fail

DELAUNAY TRIANGULATION

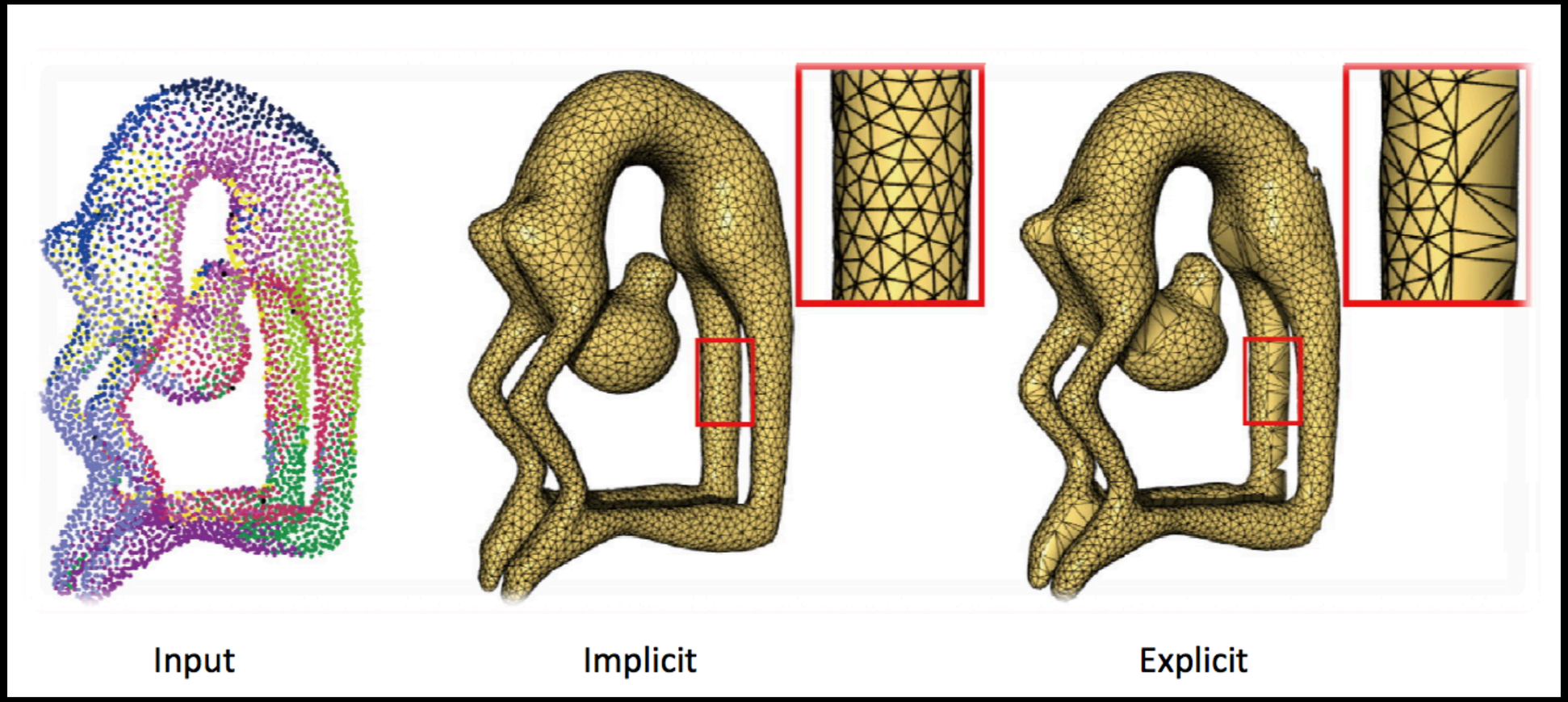


IMPLICIT RECONSTRUCTION

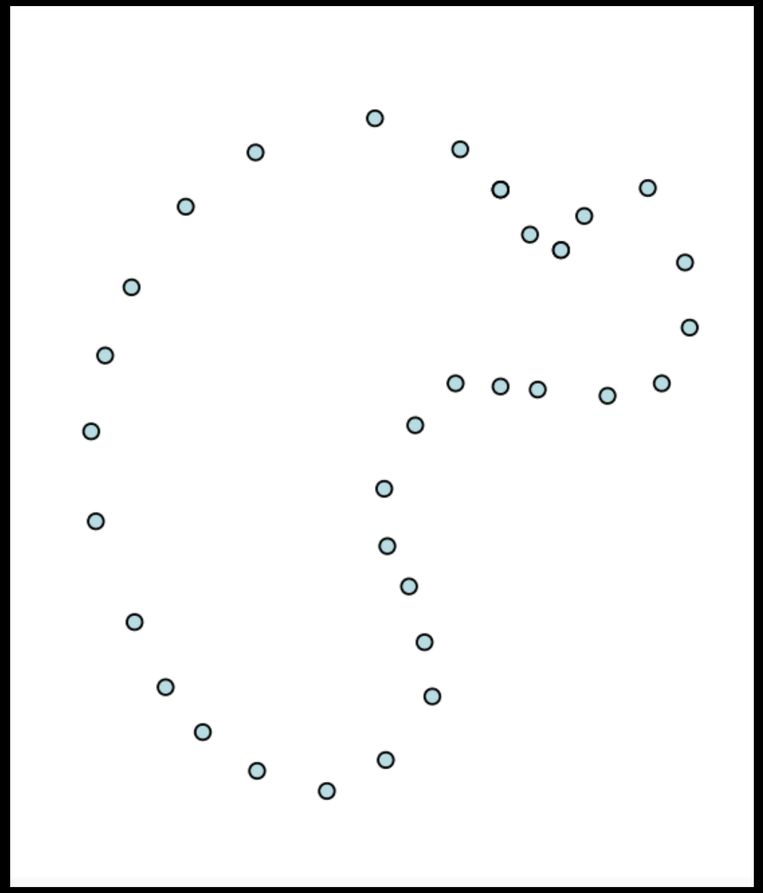
IMPLICIT SURFACE FITTING

- Idea:
  - 1.Define a smooth implicit surface that approximate the underlying real surface
  - 2.Project or generate points on this implicit surface
- Main issue: how to define the implicit surface ?
- Lots of possibilities: distance function, MLS, Radial Basis Functions (RBF), ...

IMPLICIT RECONSTRUCTION

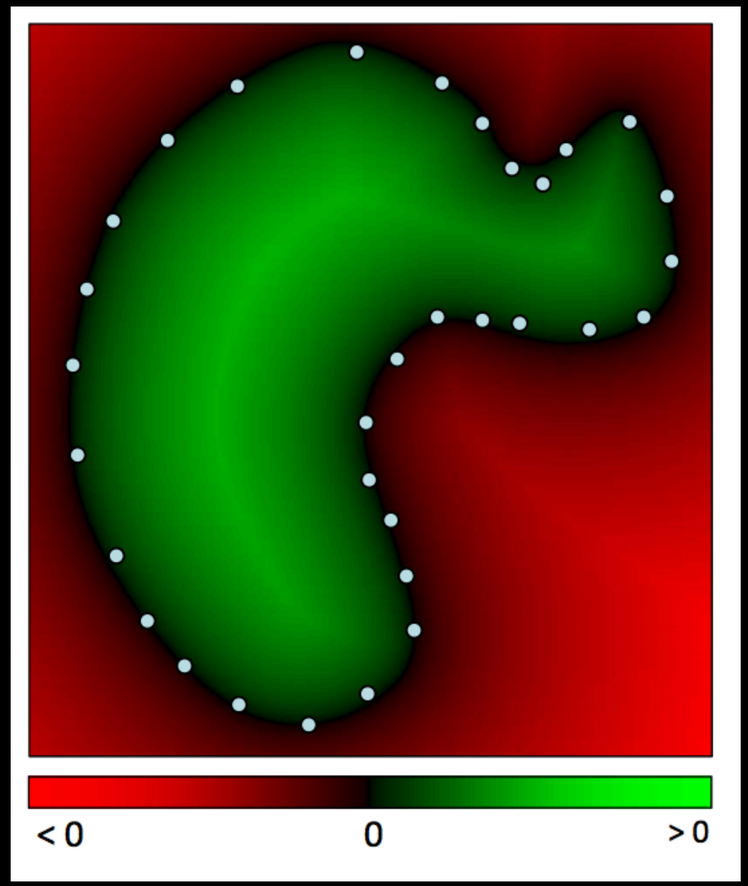


IMPLICIT RECONSTRUCTION



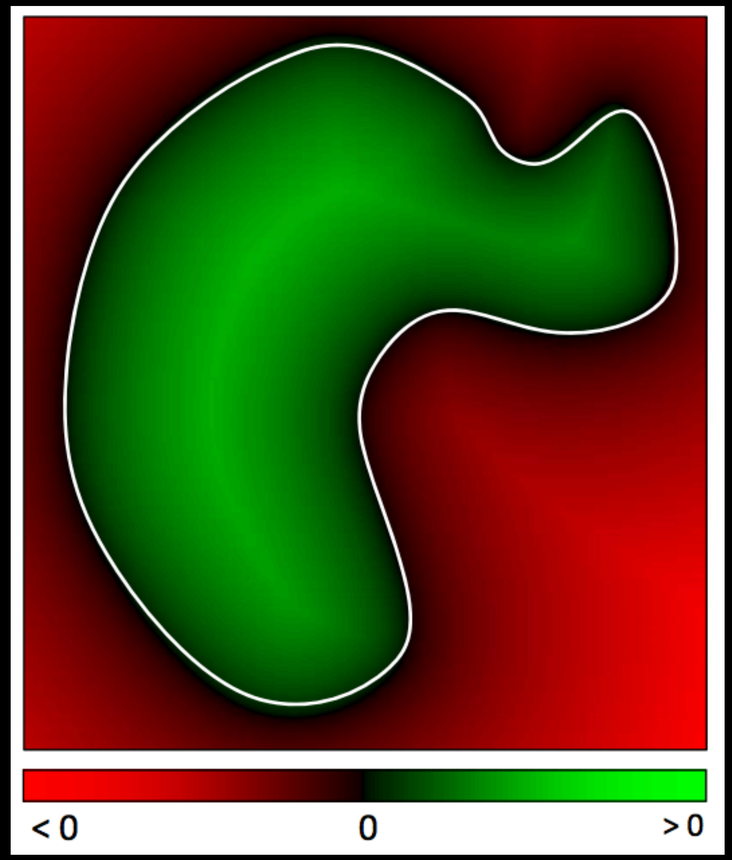
IMPLICIT RECONSTRUCTION

- Define a function
  - $f : R^3 \rightarrow R$
- with value < 0 outside the shape and > 0 inside



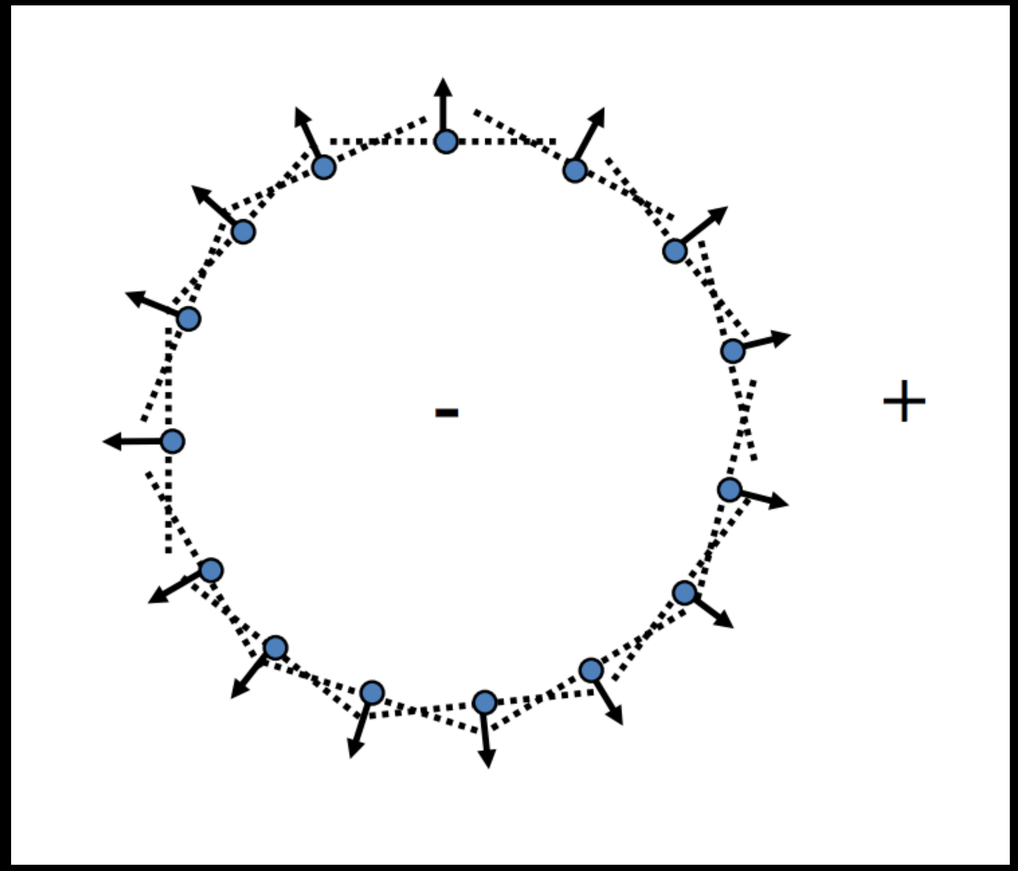
IMPLICIT RECONSTRUCTION

- Define a function
  - $f : R^3 \rightarrow R$
- with value < 0 outside the shape and > 0 inside
- Extract the 0 set
  - $\{ x : f(x) = 0 \}$



SDF FROM POINTS AND NORMALS

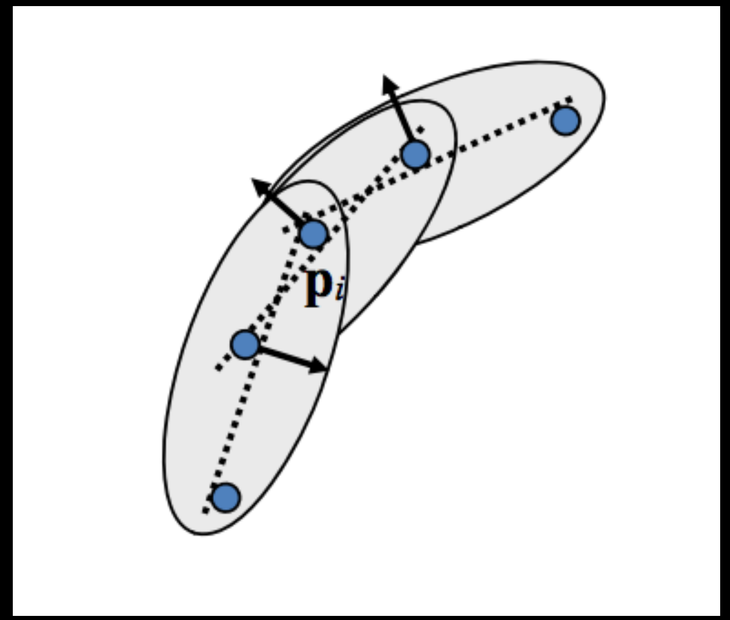
- Compute signed distance to the tangent plane of the closest point.
- Normals help to distinguish between inside and outside.
- How to get normals?



NORMAL ESTIMATION

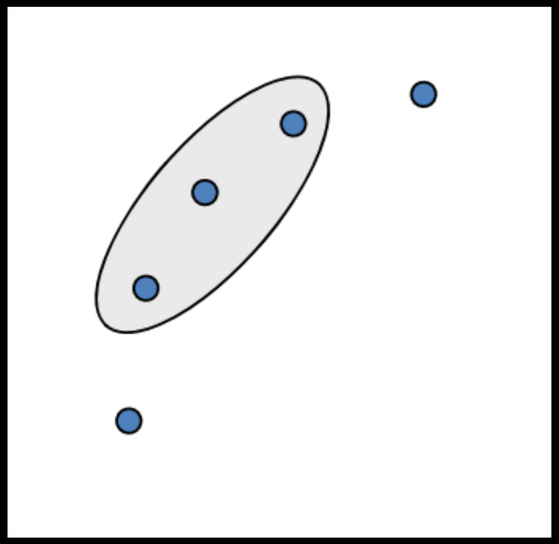
NORMAL ESTIMATION

- Find normal **n** for each sample point **p**
  - Examine local neighborhood for each point
    - Set of ***K*** nearest neighbors
  - Compute best approximating tangent plane
    - Covariance analysis
  - Determine normal orientation
    - Minimal Spanning Tree propagation



LOCAL NEIGHBORHOOD

- Find k nearest nearest neighbors neighbors (kNN) of a point
  - Brute force: O(n) complexity
- Use BSP tree
  - Binary space partitioning tree
  - Recursively partition 3D space by planes
  - Tree should be balanced, balanced, put plane at median
  - log(n) tree levels, complexity O(log n)



COMPUTE BEST APPROXIMATING TANGENT PLANE

- Plane center is barycenter of points

$$\mathbf{c} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{p}_i$$

- Normal is eigenvector w.r.t. smallest eigenvalue of covariance matrix

$$\mathbf{C} = \sum_{i=1}^{m}(\mathbf{p}_i - \mathbf{c})(\mathbf{p}_i - \mathbf{c})^T$$

COMPUTE BEST APPROXIMATING TANGENT PLANE

- Principal Component Analysis (PCA)
  - Extract points {qi} in neighborhood
  - Compute covariance matrix M
- Analyze eigenvalues and eigenvectors of M (via SVD)
- Surface normal is estimated by eigenvector (principal axis) associated with smallest eigenvalue

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

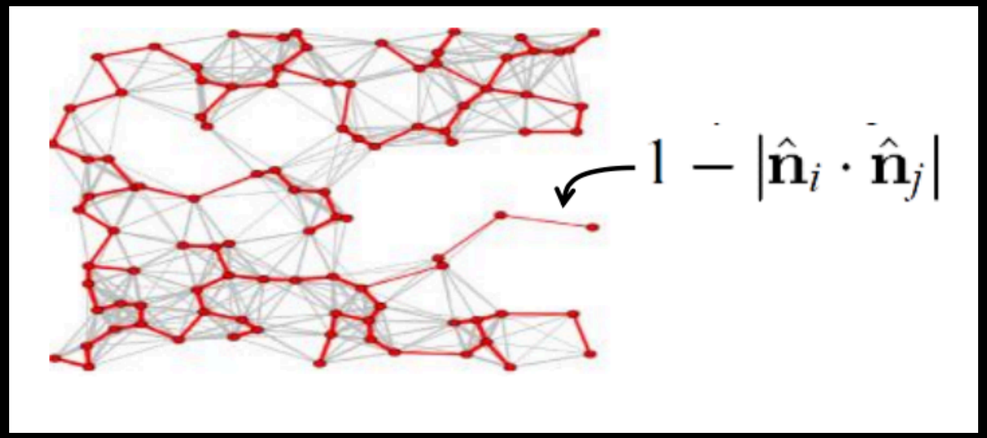
Covariance Matrix

$$\mathbf{M} = \mathbf{U}\mathbf{S}\mathbf{U}^t$$

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

Eigenvalues & Eigenvectors

NORMAL ORIENTATION



- Normals might not be consistent
- General Pointcloud
  - Traverse nearest neighbor graph flipping normals for consistency
  - Greedy propagation algorithm
    - (minimum spanning tree of normal similarity)
- In our case:
  - We can check that the normal of the point looks towards the camera where it’s visible

HOPPE

HOPPE ET AL. SIGGRAPH 1992

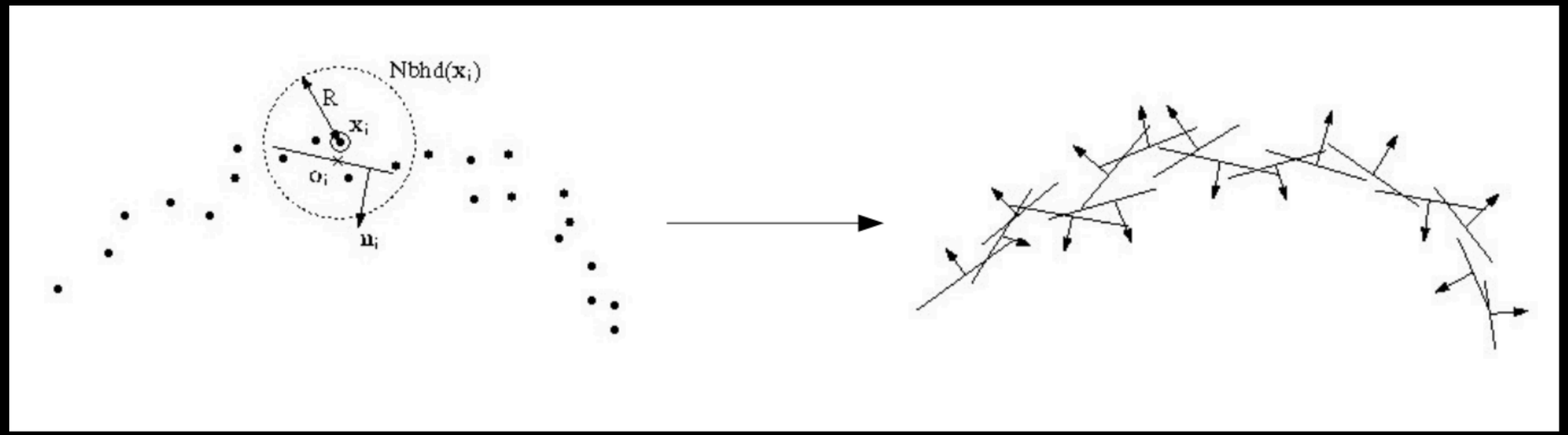
- “Surface reconstruction from unorganized points”
- Input:
  - No geometry (normals), topology or boundaries information: inferred from the point set
  - Sampling density and max error known
- Output: a meshed surface
  - Compact, connected, orientable 2-manifold
  - Not necessarily triangles

HOPPE ET AL. SIGGRAPH 1992

- 4 stages:
  - 1.Estimate tangent plane for each input point
    - Local linear approximation of the surface
  - 2.Establish consistent orientation for nearby planes
    - => consistent orientation for the whole surface
  - 3.Compute signed distances on a voxel grid
  - 4.Extract an isosurface
    - Distance function: f ~ signed Euclidean distance to the input unknown surface

STAGE 1: TANGENT PLANE

- Nearest neighbors approach
  - R = d+e, d = sampling density, e = max error
  - Approximating plane found by a least square approximation



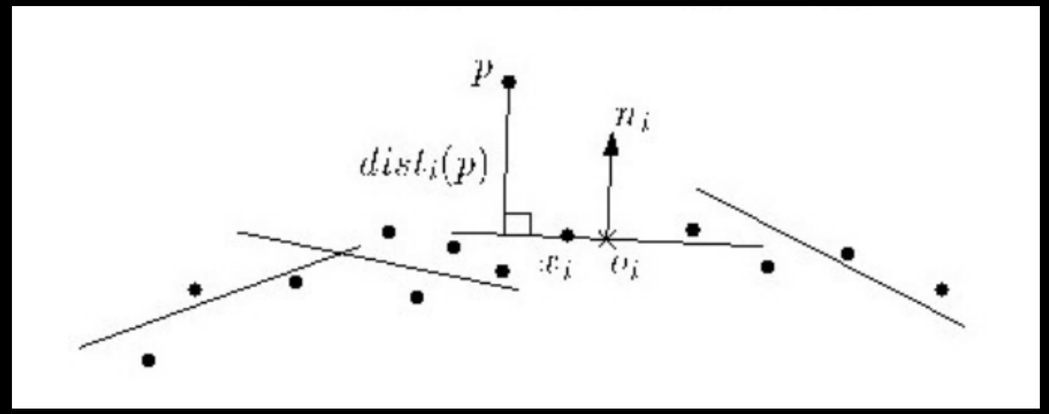
STAGE 2: ORIENTATION CONSISTENCY

- Graph optimization approach
  - One node/plane, one edge p1-p2 if centroids are close
  - Edge weight = scalar product of plane normals
  - Maximize total weight of the graph
  - NP-complete => a little more complicated than that

STAGE 3: SIGNED DISTANCES

- Signed distance from each point to the closest plane:

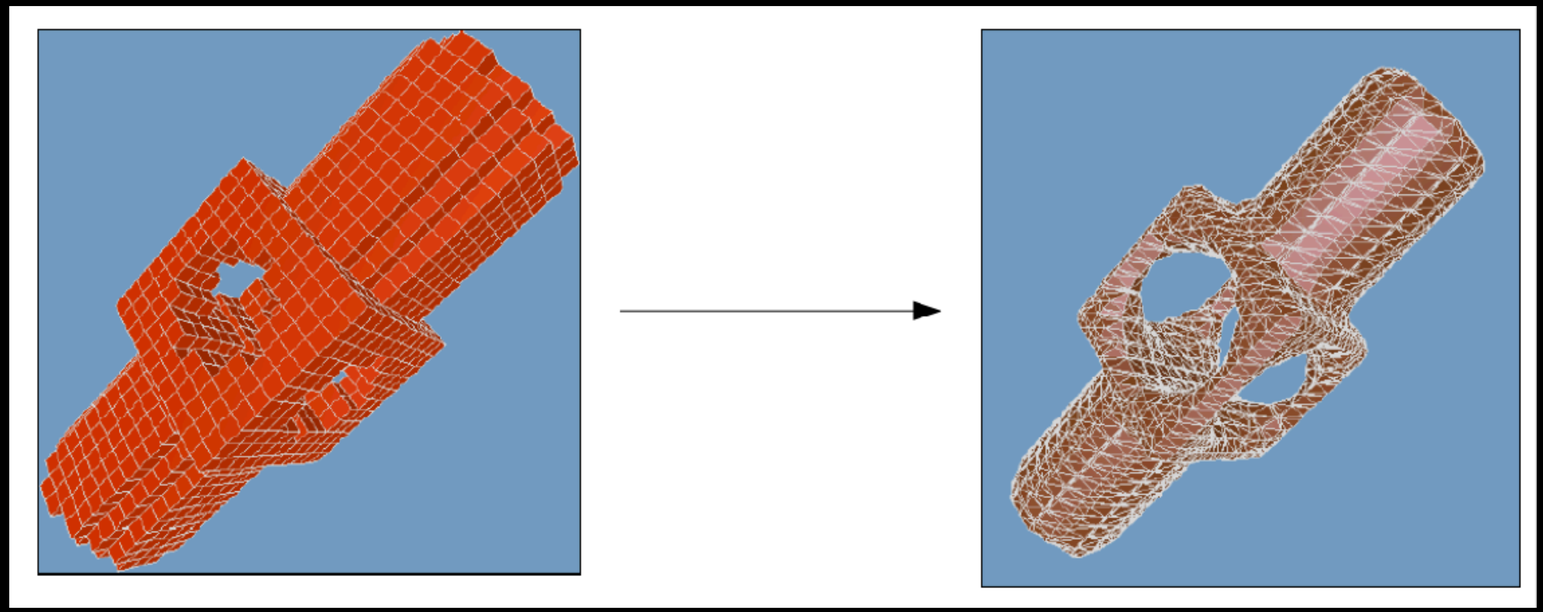
$$\mathrm{dist}_i(\mathbf{p}) = (\mathbf{p} - \mathbf{o_i}) \cdot \mathbf{n_i}$$



- Surfaces with boundaries: points too far away are assigned an “undefined” distance
- Distance sampled at the vertices of a voxel grid

STAGE 4: ISOSURFACE EXTRACTION

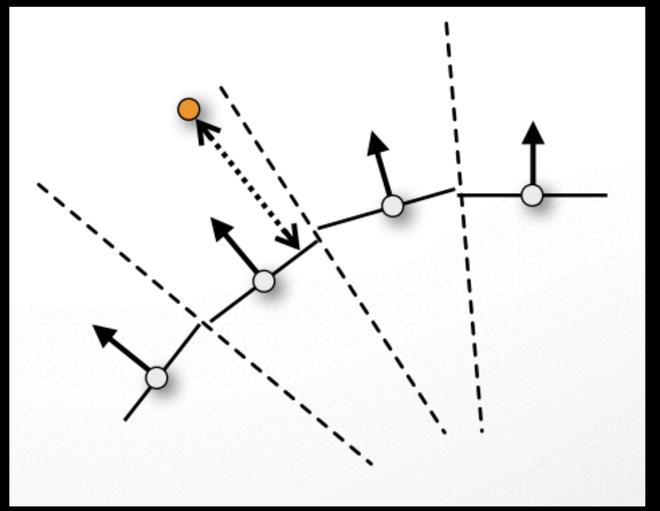
- Well-known Marching Cubes
  - Lorensen & Cline, SIGGRAPH 1987
- When “undefined”: no triangle



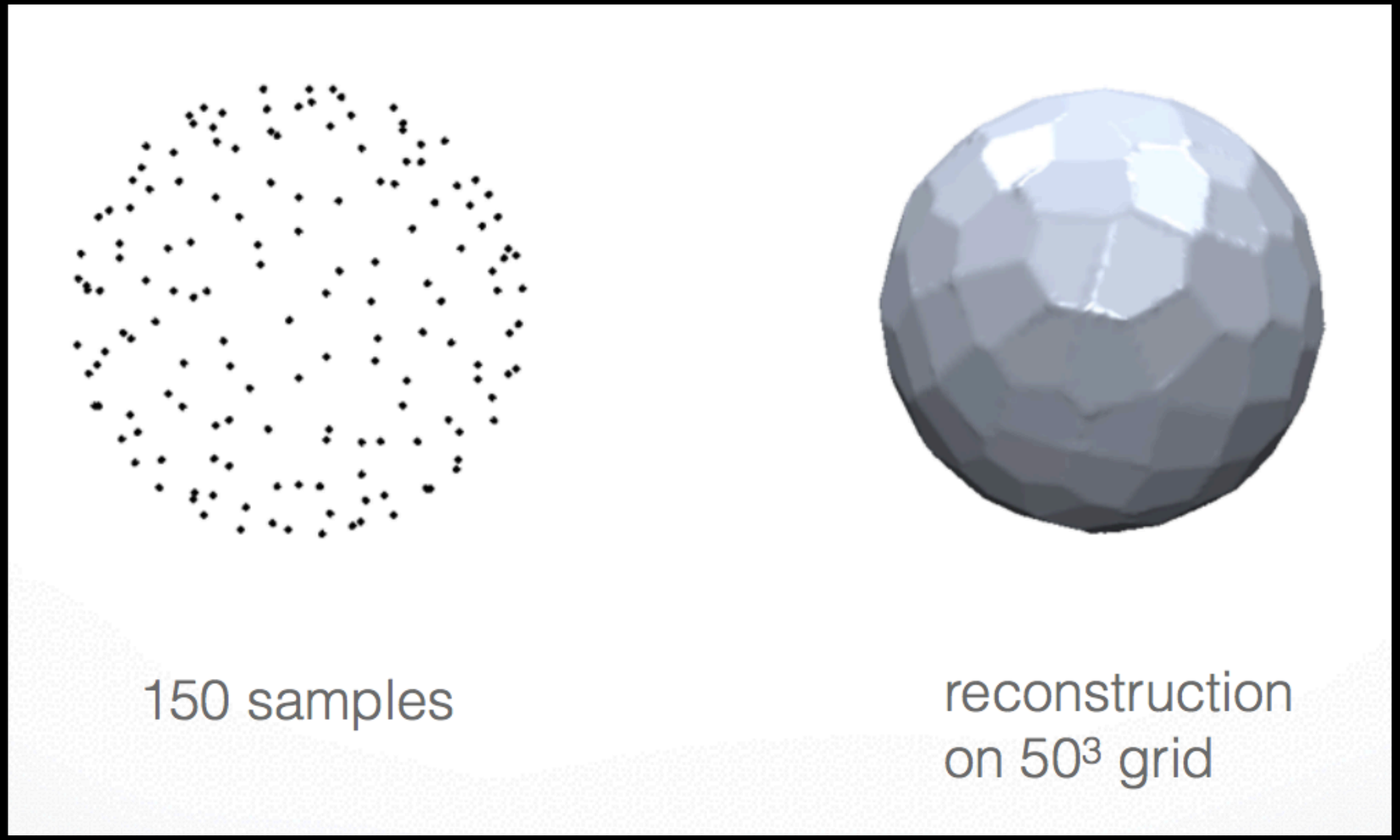
SIGNED DISTANCE FUNCTION

SIGNED DISTANCE FUNCTION

- Points + normals determine local tangent planes
- Use distance from closest point's tangent plane
- Linear approximation in Voronoi cell
- Simple and efficient, but…



SIGNED DISTANCE FUNCTION

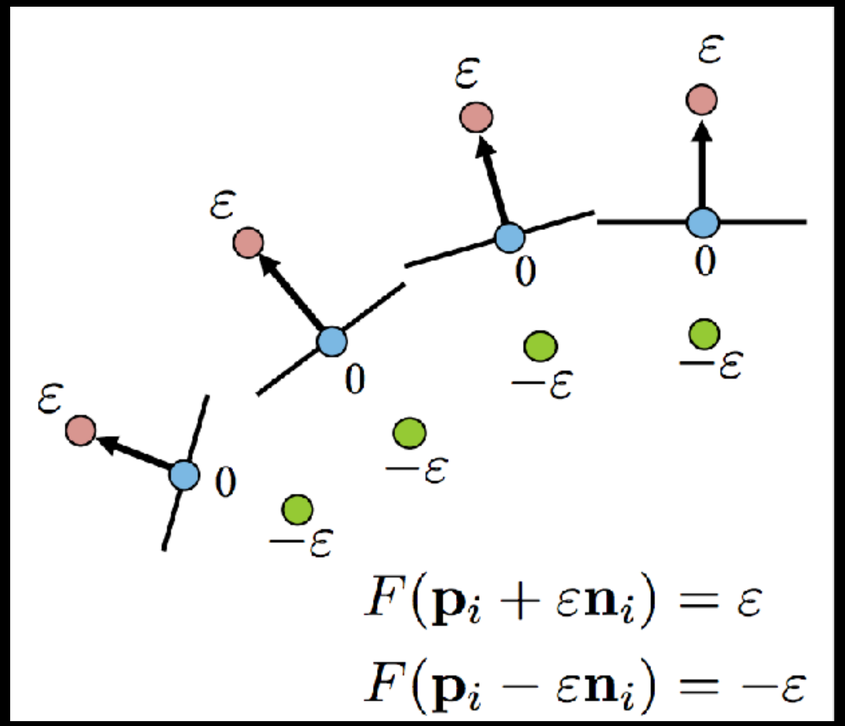


SMOOTH SDF

- Instead find a smooth formulation for F.
- Requiring:
- F(p) = 0 (interpolating)
- F is smooth.
- Avoid trivial solution F ≡ 0. (0 everywhere in space)

SMOOTH SDF

- Instead find a smooth formulation for F.
- Requiring:
- F(p) = 0 (interpolating)
- F is smooth.
- Avoid trivial solution F ≡ 0. (0 everywhere in space)



RADIAL BASIS FUNCTION INTERPOLATION

- RBF: Weighted sum of shifted, smooth kernels

$$F(x) = \sum_{i=0}^{N-1} w_i \varphi(\|x - c_i\|)$$

Scalar weights
Unknowns

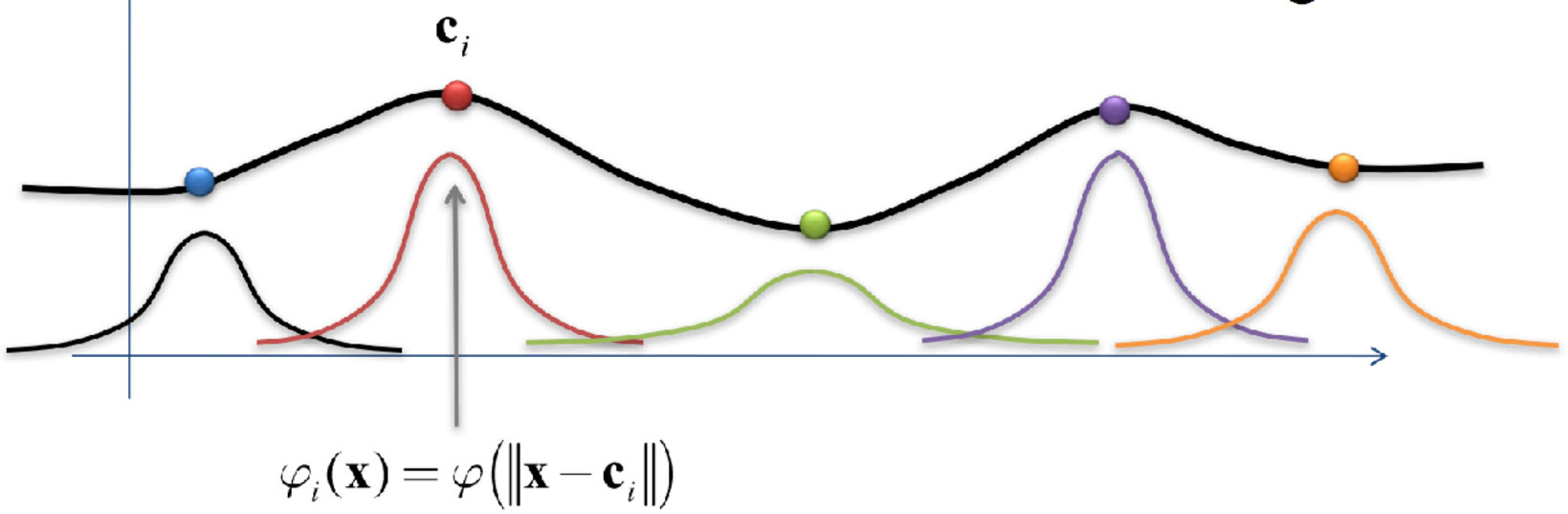
Smooth kernels
(basis functions)
centered at constrained
points.
For example:
$\varphi(r) = r^3$

$N = 3n$

RADIAL BASIS FUNCTION INTERPOLATION

$$F(x) = \sum_{i=0}^{N-1} w_i \varphi(\|x - c_i\|)$$

How do we find the weights?



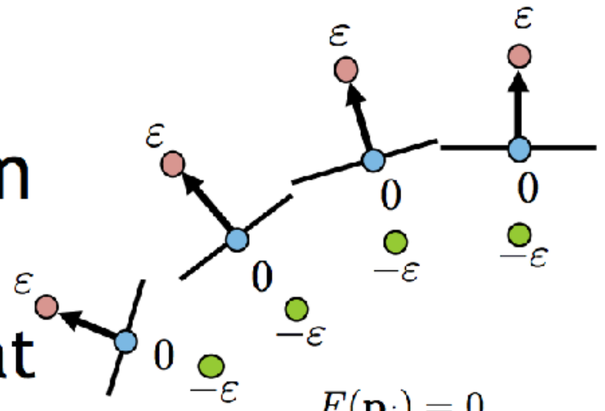
RADIAL BASIS FUNCTION INTERPOLATION

- Interpolate the constraints:

$$\{\mathbf{c}_{3i}, \mathbf{c}_{3i+1}, \mathbf{c}_{3i+2}\} = \{\mathbf{p}_i,\ \mathbf{p}_i + \varepsilon\mathbf{n}_i,\ \mathbf{p}_i - \varepsilon\mathbf{n}_i\}$$

$$\forall j = 0, \ldots, N-1, \quad \sum_{i=0}^{N-1} w_i\, \varphi(\|\mathbf{c}_j - \mathbf{c}_i\|) = d_j$$

- In words: we know the desired function value (sum of RBF) at each center.
  - We solve for the weights that do that!



$$F(\mathbf{p}_i) = 0$$
$$F(\mathbf{p}_i + \varepsilon\mathbf{n}_i) = \varepsilon$$
$$F(\mathbf{p}_i - \varepsilon\mathbf{n}_i) = -\varepsilon$$

RADIAL BASIS FUNCTION INTERPOLATION

- Interpolate the constraints:

$$\{\mathbf{c}_{3i}, \mathbf{c}_{3i+1}, \mathbf{c}_{3i+2}\} = \{\mathbf{p}_i,\ \mathbf{p}_i + \varepsilon\mathbf{n}_i,\ \mathbf{p}_i - \varepsilon\mathbf{n}_i\}$$

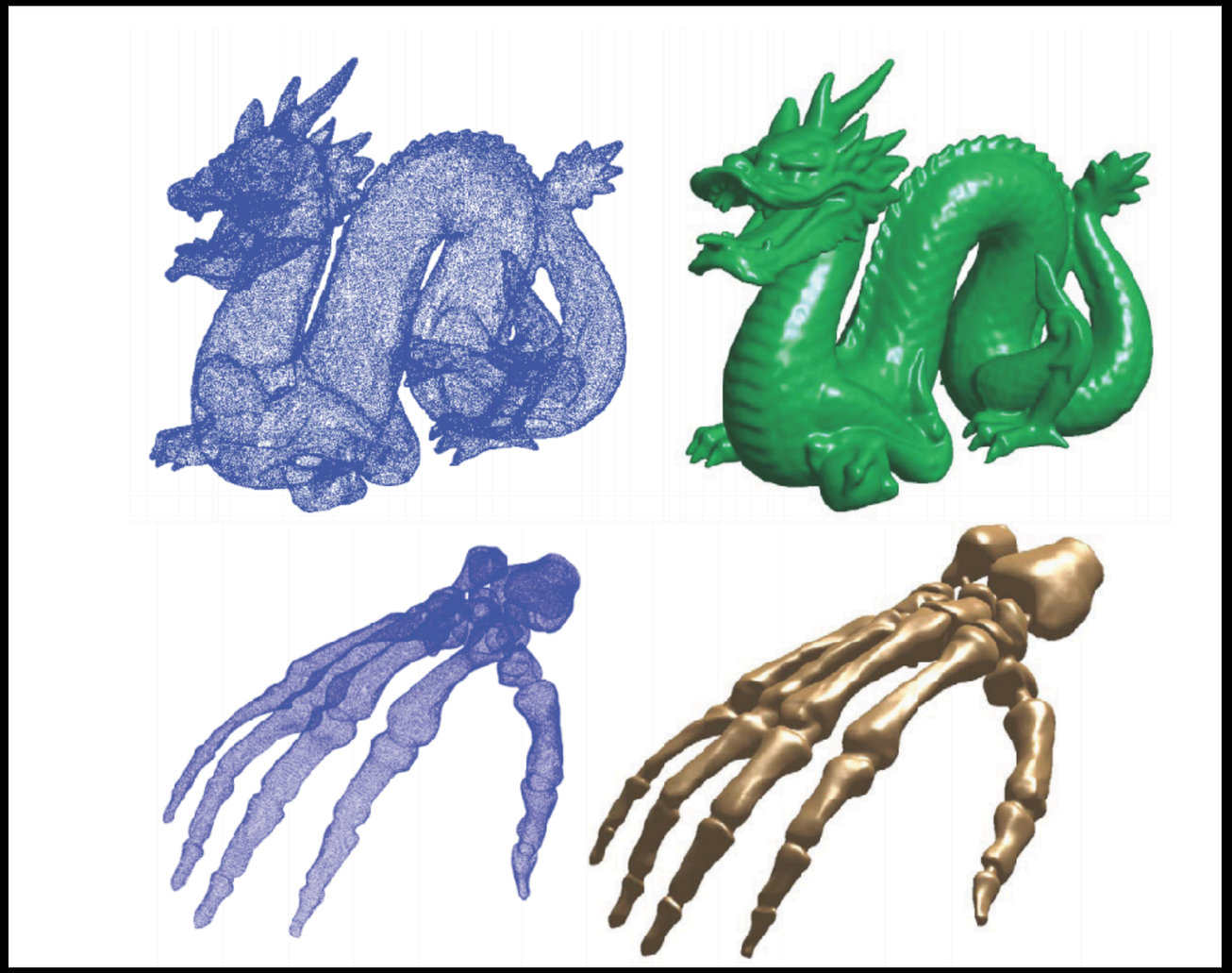
- Symmetric linear system to get the weights:

$$\begin{pmatrix} \varphi(\|\mathbf{c}_0 - \mathbf{c}_0\|) & \cdots & \varphi(\|\mathbf{c}_0 - \mathbf{c}_{N-1}\| \\ \vdots & \ddots & \vdots \\ \varphi(\|\mathbf{c}_{N-1} - \mathbf{c}_0\|) & \cdots & \varphi(\|\mathbf{c}_{N-1} - \mathbf{c}_{N-1}\|) \end{pmatrix} \begin{pmatrix} w_0 \\ \vdots \\ w_{N-1} \end{pmatrix} = \begin{pmatrix} d_0 \\ \vdots \\ d_{N-1} \end{pmatrix}$$

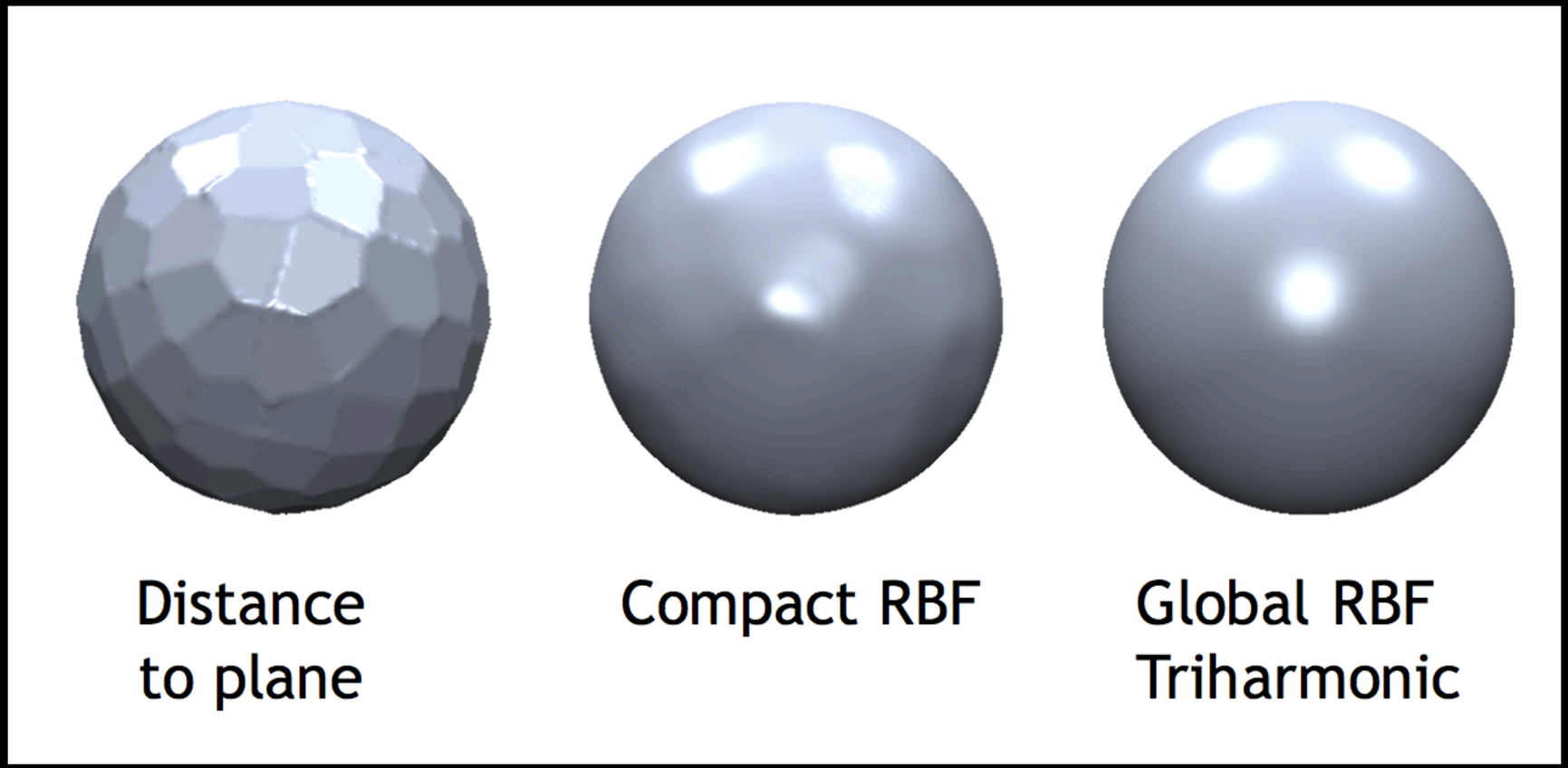
RBF KERNELS

- Triharmonic: $\varphi(r) = r^3$
  - Globally supported.
  - Leads to dense symmetric linear system.
  - C^2 smoothness.
  - Works well for highly irregular sampling.

RECONSTRUCTION RESULTS WITH RBF



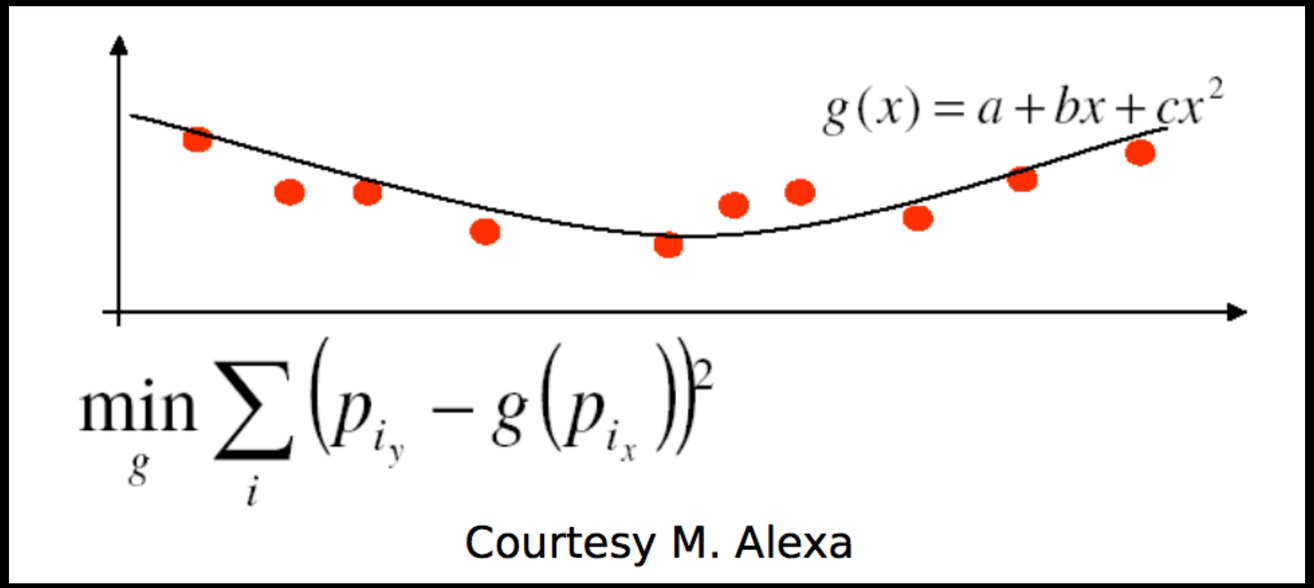
RBF KERNELS



MOVING LEAST SQUARES

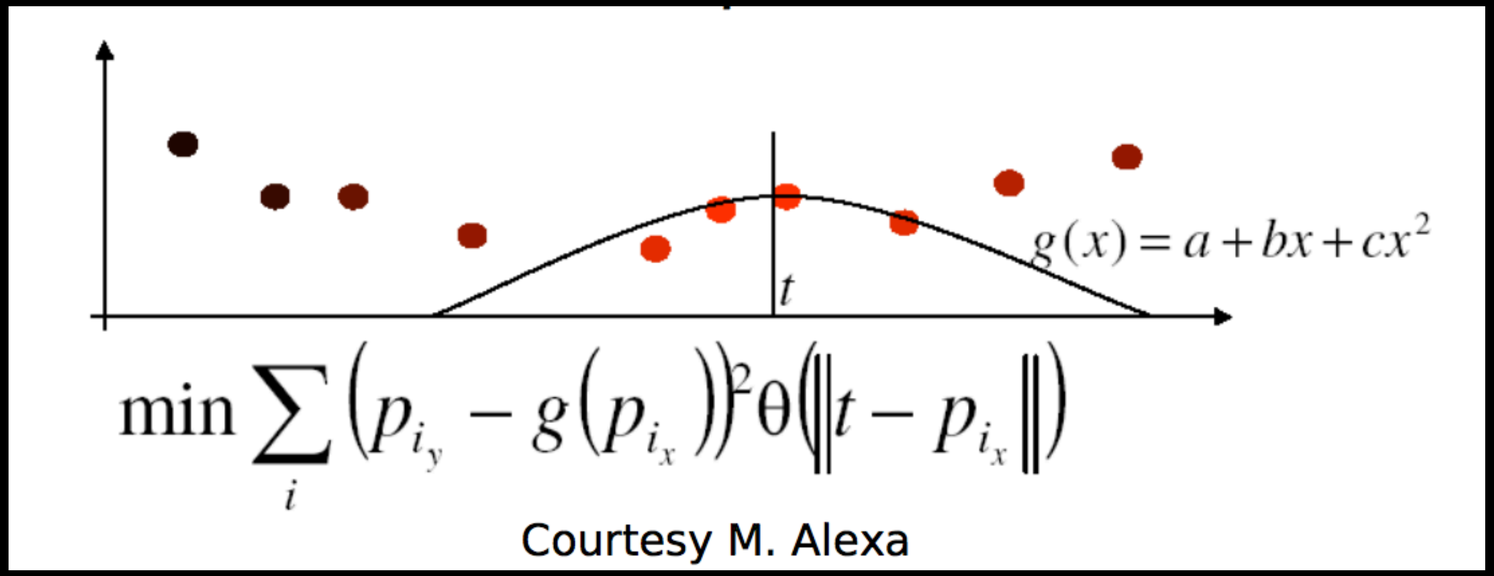
LEAST SQUARES

- Goal: fit a primitive (e.g. Polynomial function) to scattered data
- Idea: minimize square distance between the point's values and the primitive



MOVING LEAST SQUARES

- Compute a local LS approximation at t
- Weight points based on distance to t
  - Decrease when going far from t

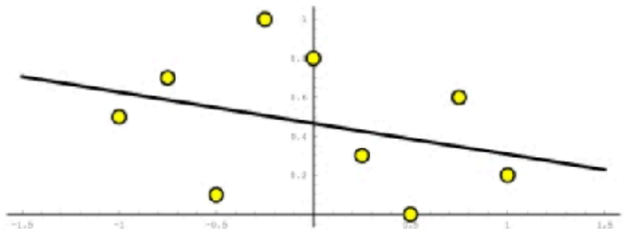


Courtesy M. Alexa

MOVING LEAST SQUARES

- Weighted LS where the weights depend on position

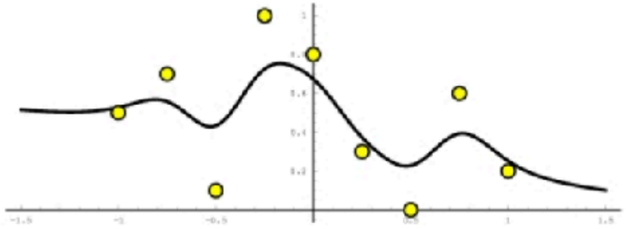
- Global least squares with linear basis



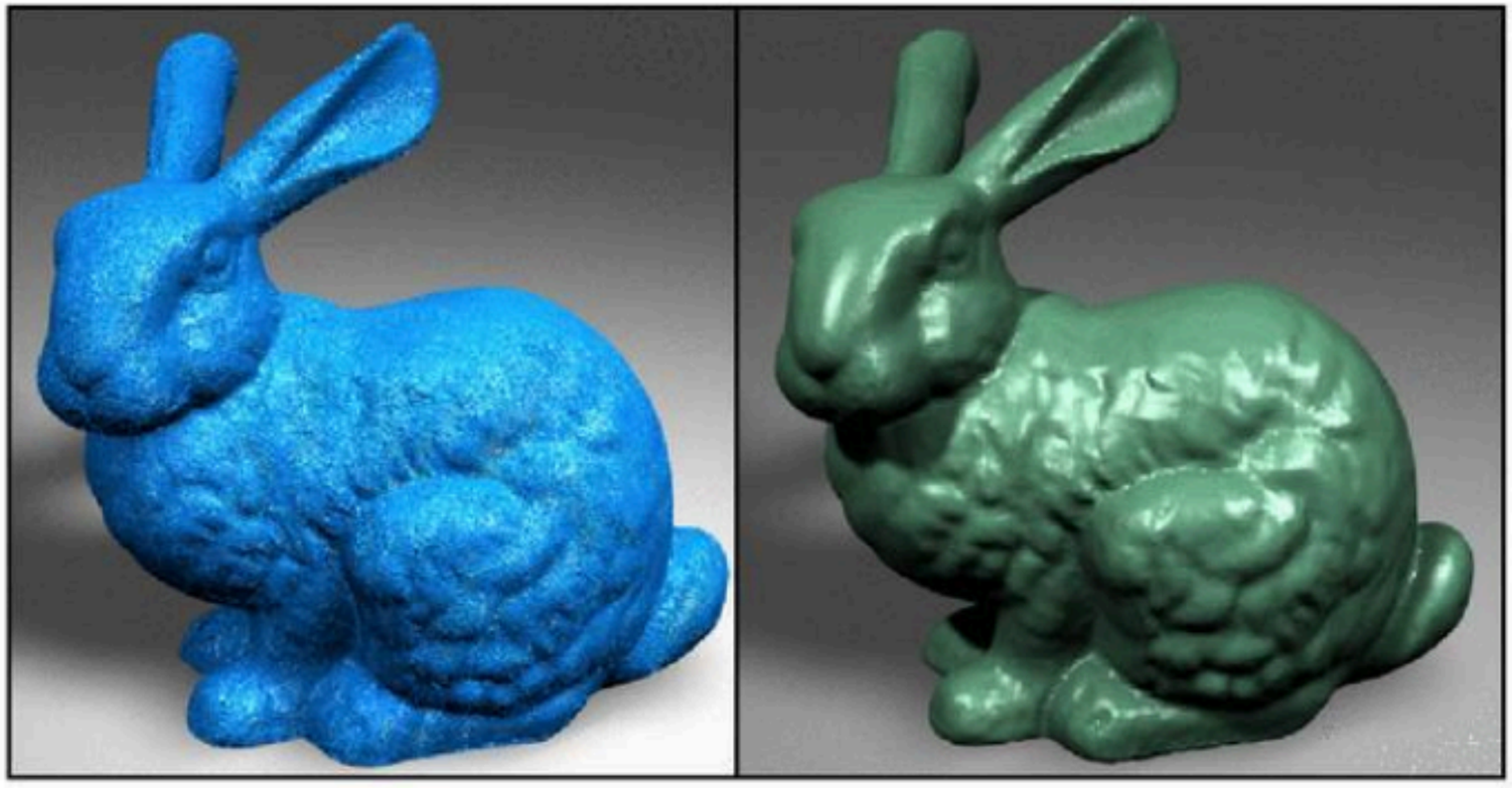
- MLS with (nearly) singular weight function

$$\theta(r) = \frac{1}{r^2 + \varepsilon^2}$$

- MLS with approximating weight function $\theta(r) = e^{-\frac{r^2}{h^2}}$



MOVING LEAST SQUARES



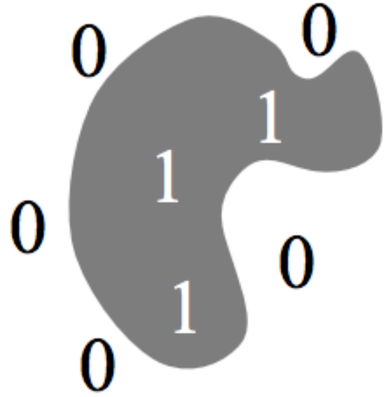
MLS

POISSON RECONSTRUCTION

THE INDICATOR FUNCTION

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

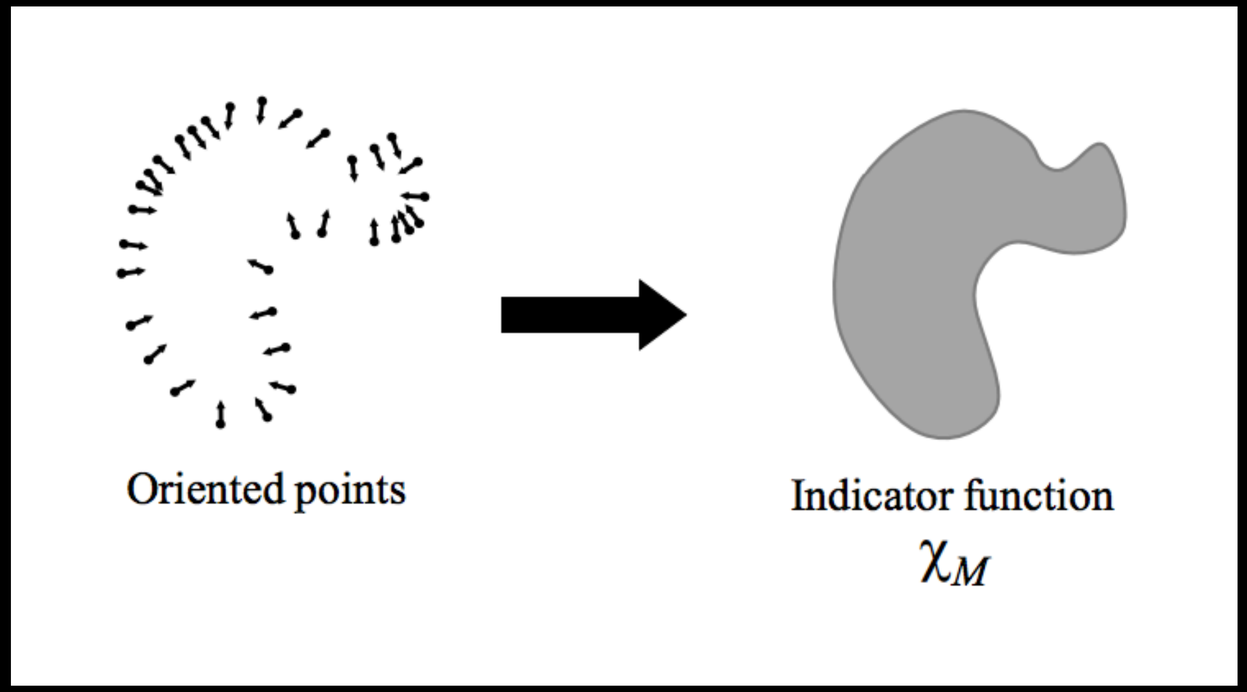
$$\chi_M(p) = \begin{cases} 1 & \text{if } p \in M \\ 0 & \text{if } p \notin M \end{cases}$$



Indicator function
χ_M

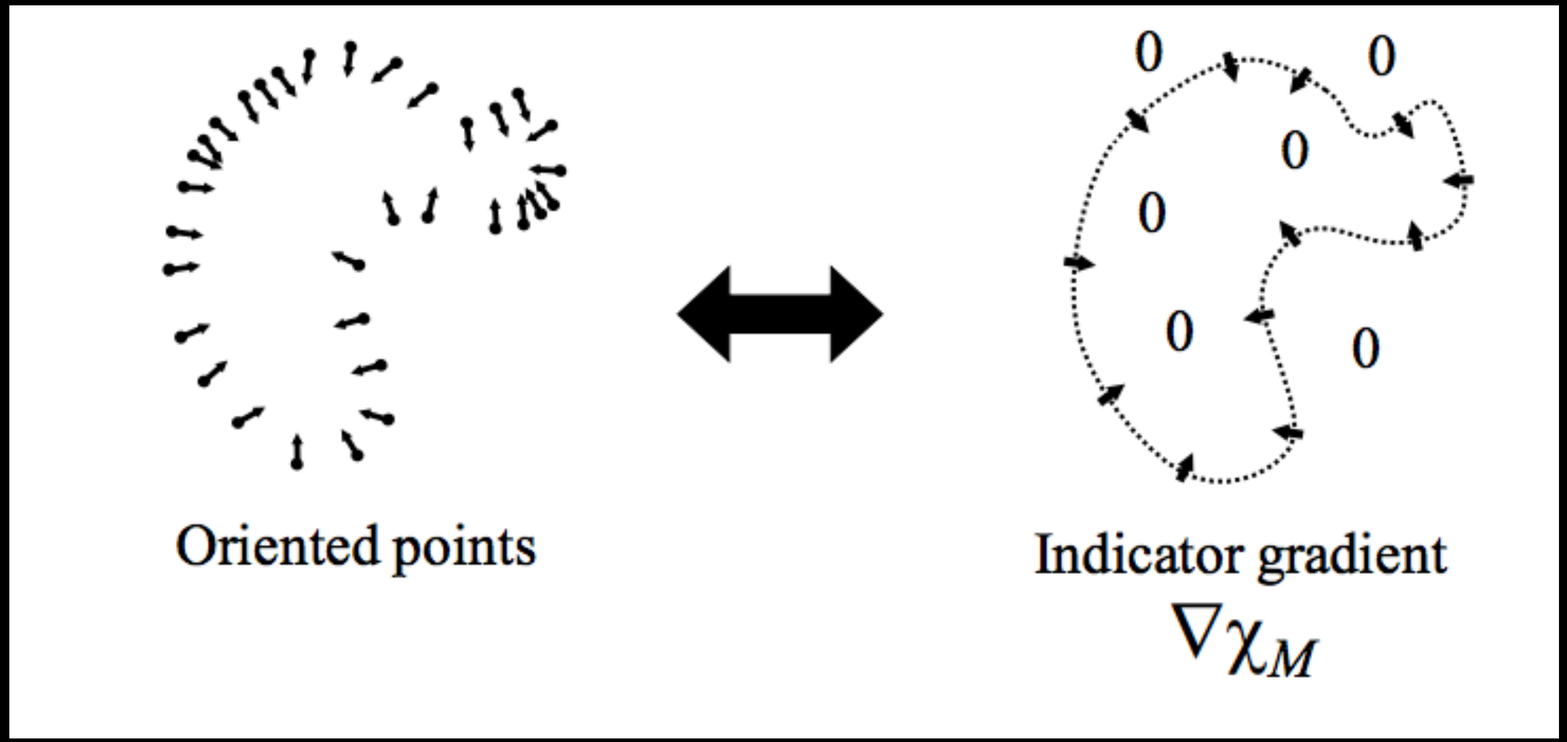
CHALLENGE

- How to construct the indicator function?



GRADIENT RELATIONSHIP

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



INTEGRATION

- Represent the points by a vector field *V*
- Find the function *X* whose gradient best approximates :

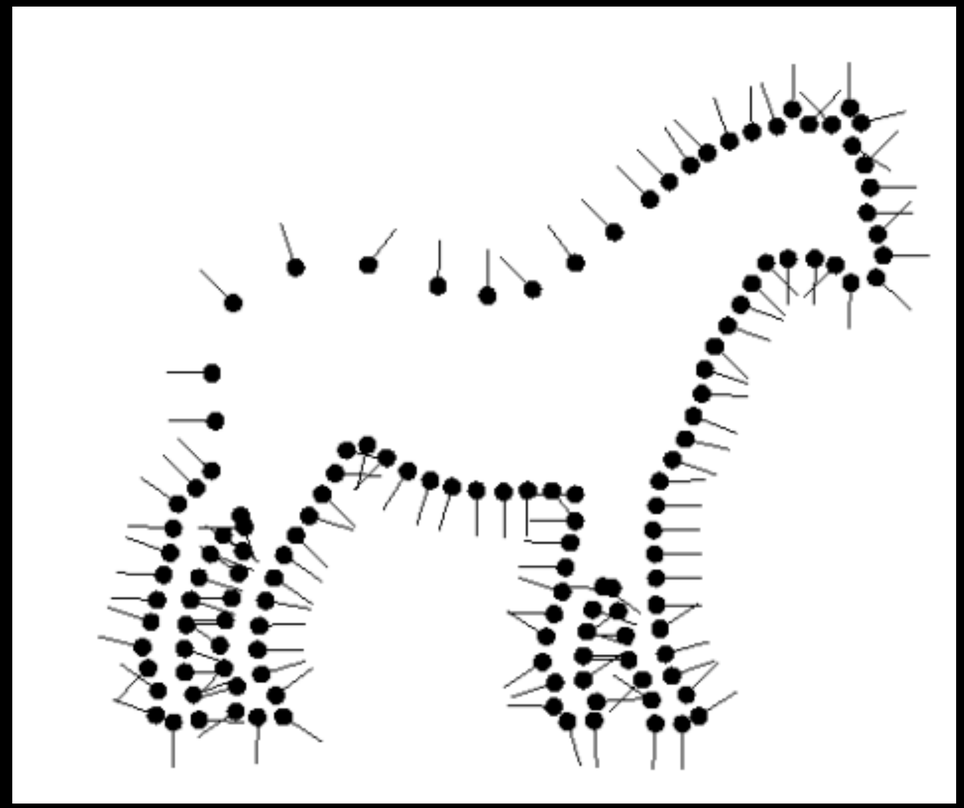
$$\min_{\chi} \left\| \nabla\chi - \vec{V} \right\|$$

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

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

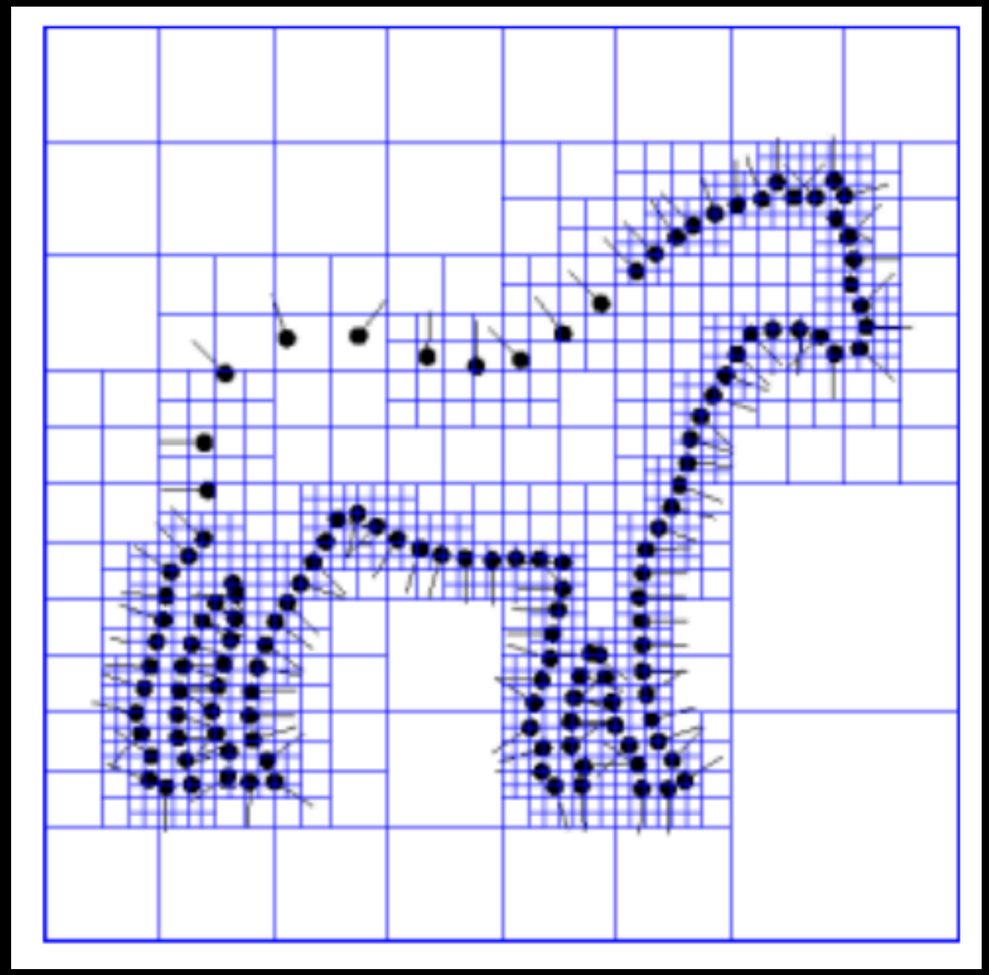
INTEGRATION

- Given the Points:
  - Set octree
  - Compute vector field
  - Compute indicator function
  - Extract iso-surface



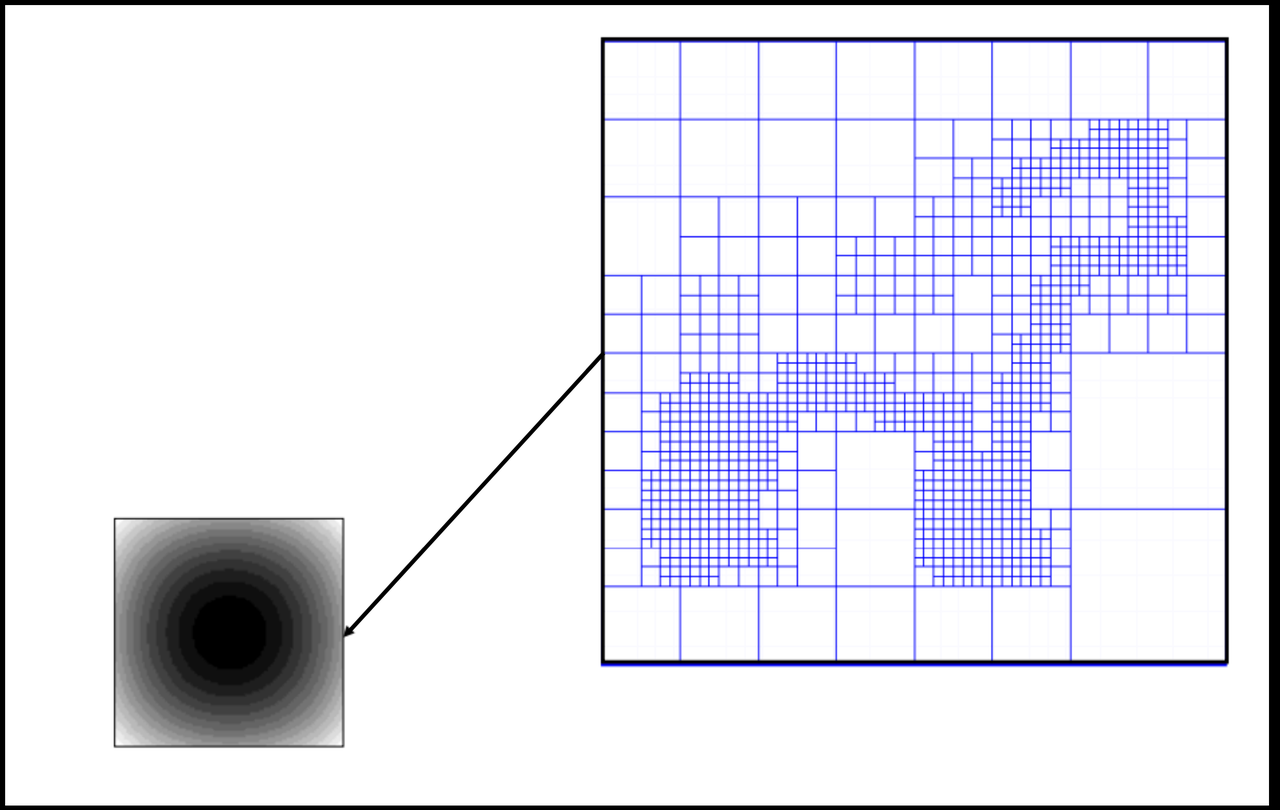
INTEGRATION

- Given the Points:
  - Set octree



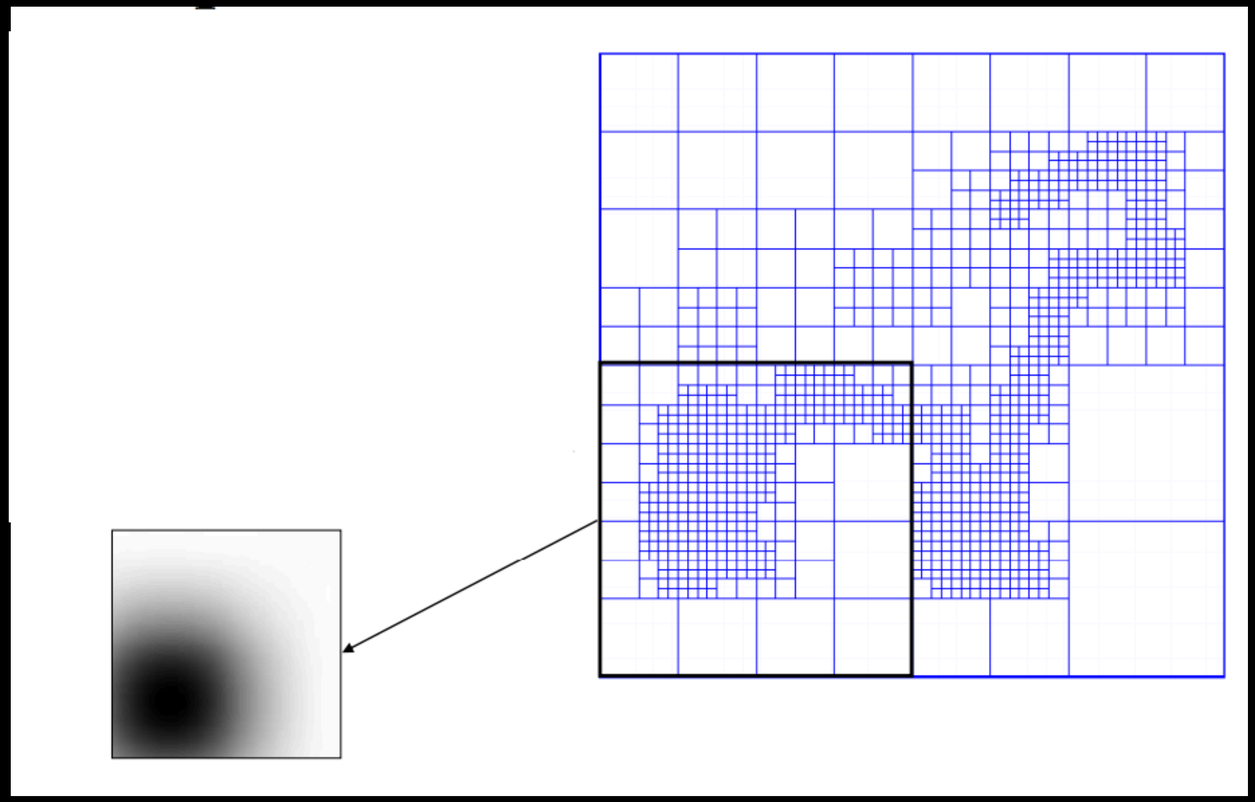
INTEGRATION

- Given the Points:
  - Set octree
  - Compute vector field
    - define function space



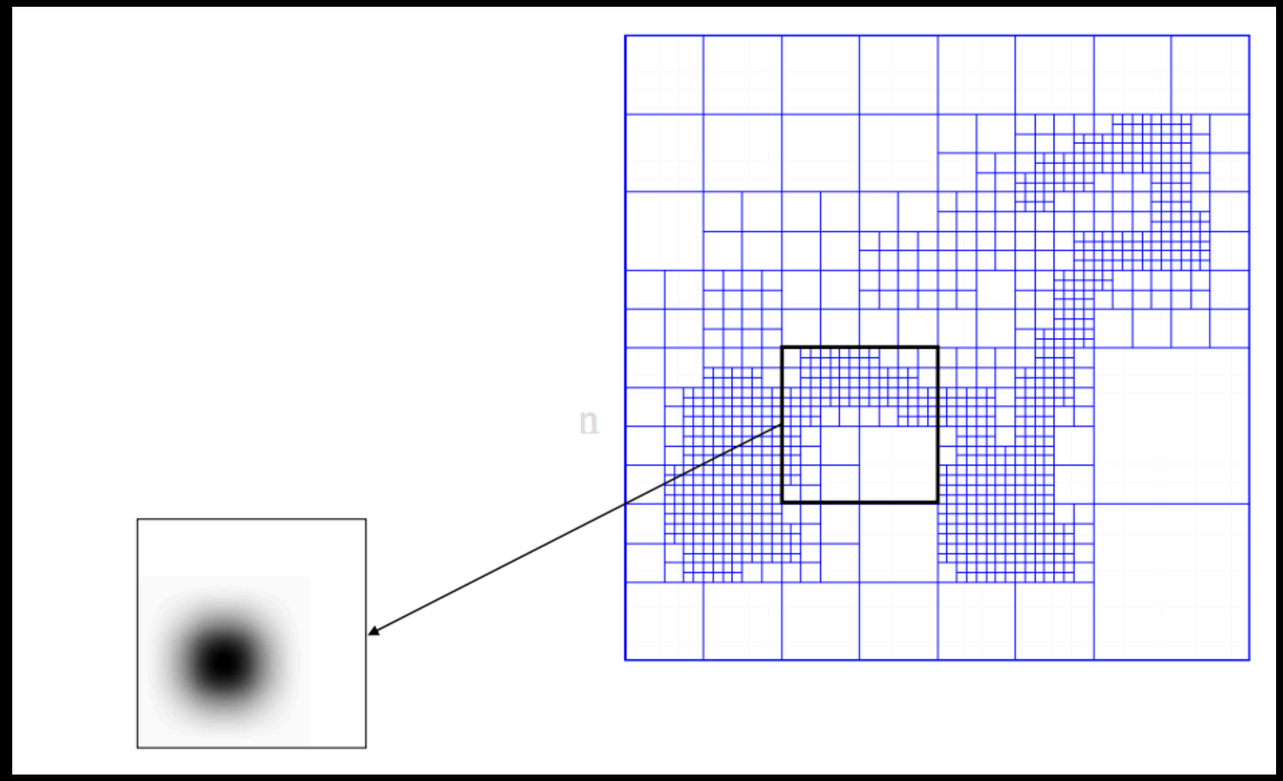
INTEGRATION

- Given the Points:
  - Set octree
  - Compute vector field
    - define function space



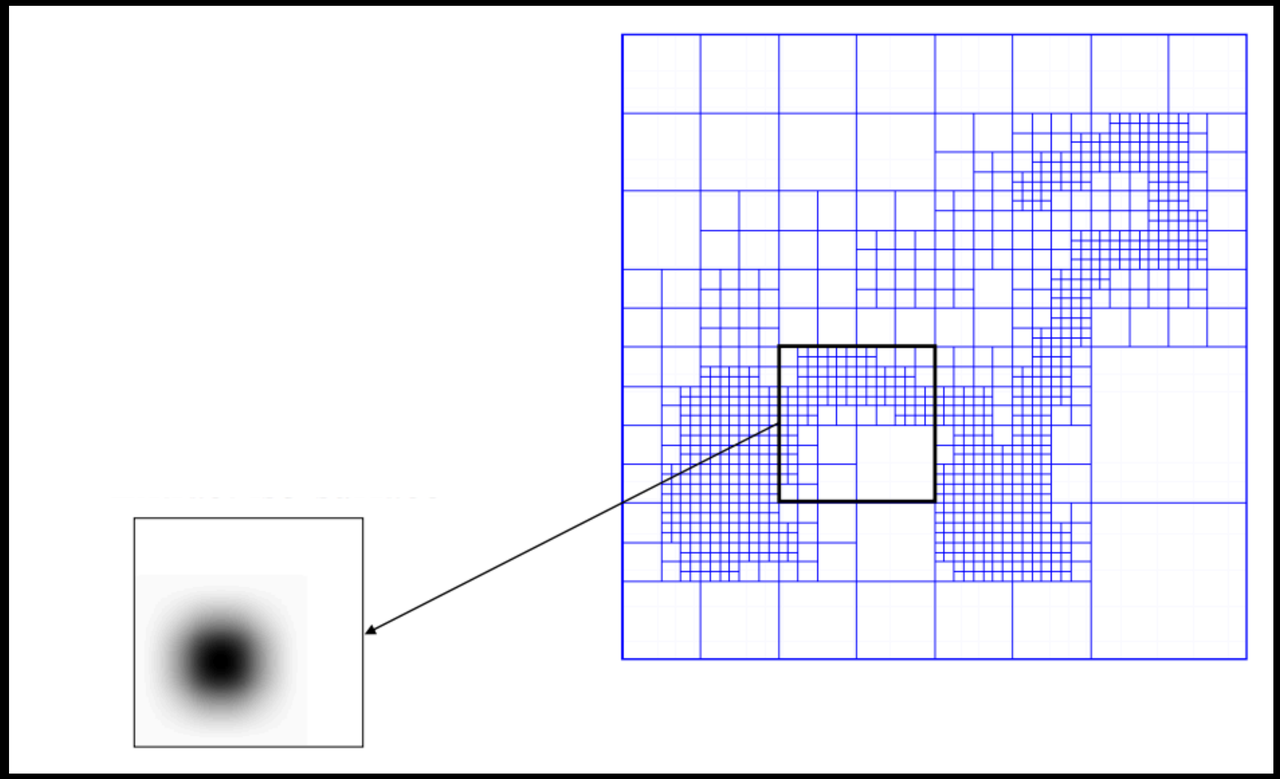
INTEGRATION

- Given the Points:
  - Set octree
  - Compute vector field
    - define function space



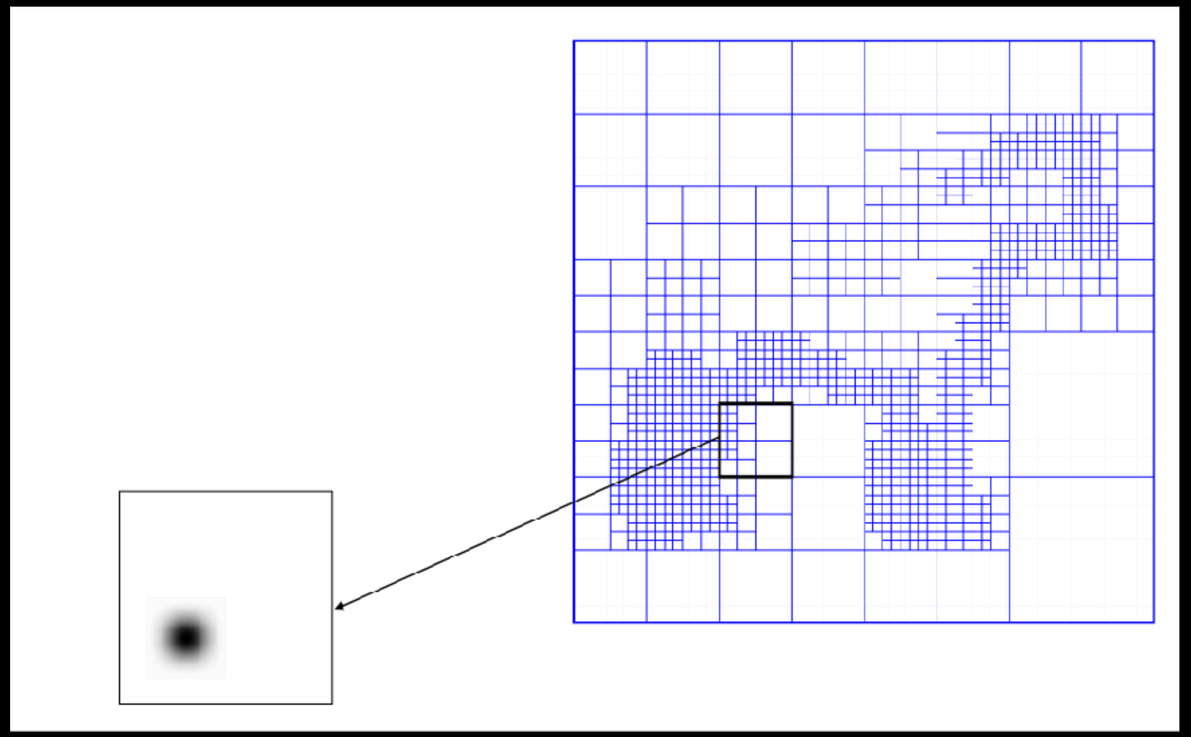
INTEGRATION

- Given the Points:
  - Set octree
  - Compute vector field
    - define function space



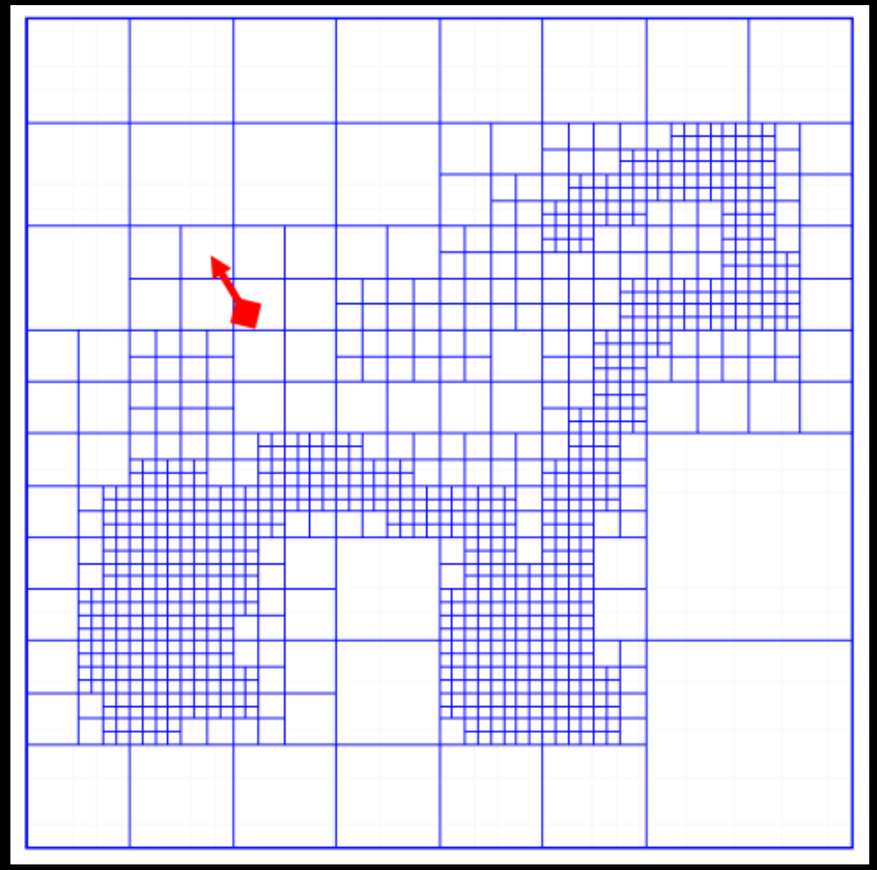
INTEGRATION

- Given the Points:
  - Set octree
  - Compute vector field
    - define function space



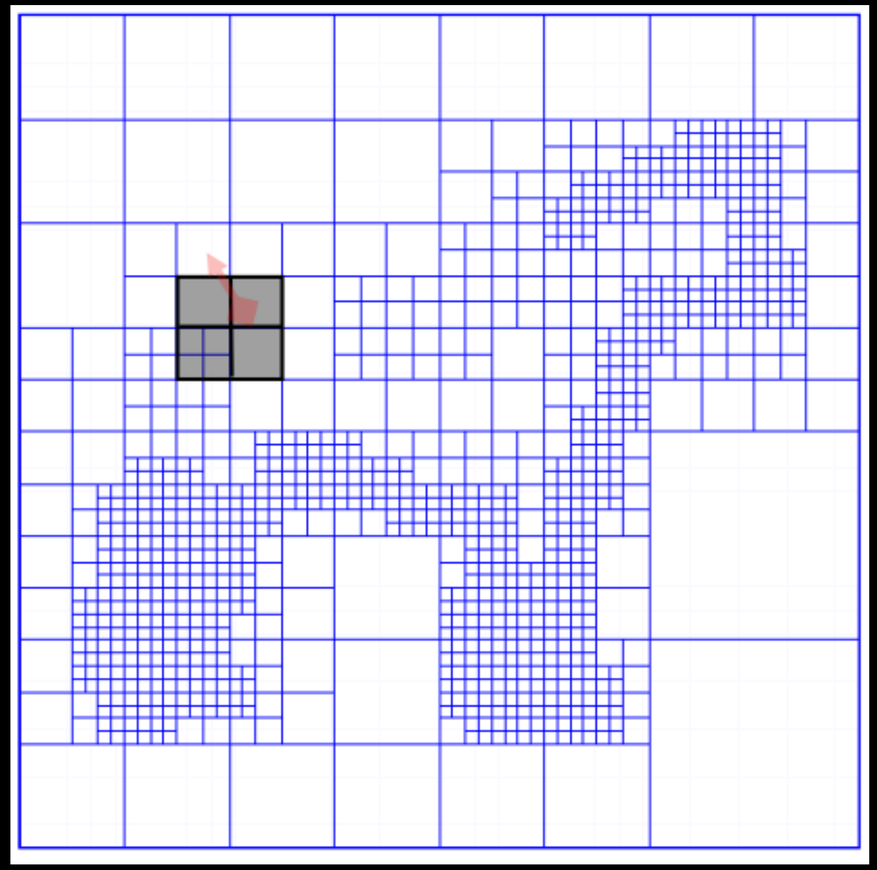
INTEGRATION

- Given the Points:
  - Set octree
  - Compute vector field
    - define function space
    - splat the samples



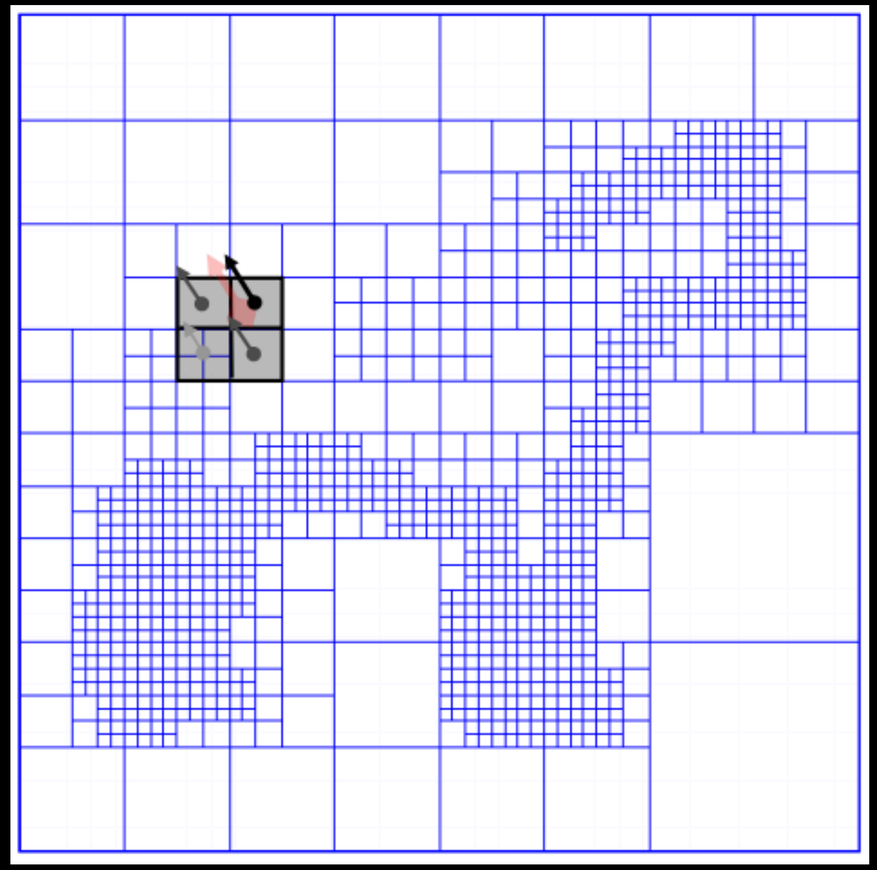
INTEGRATION

- Given the Points:
  - Set octree
  - Compute vector field
    - define function space
    - splat the samples



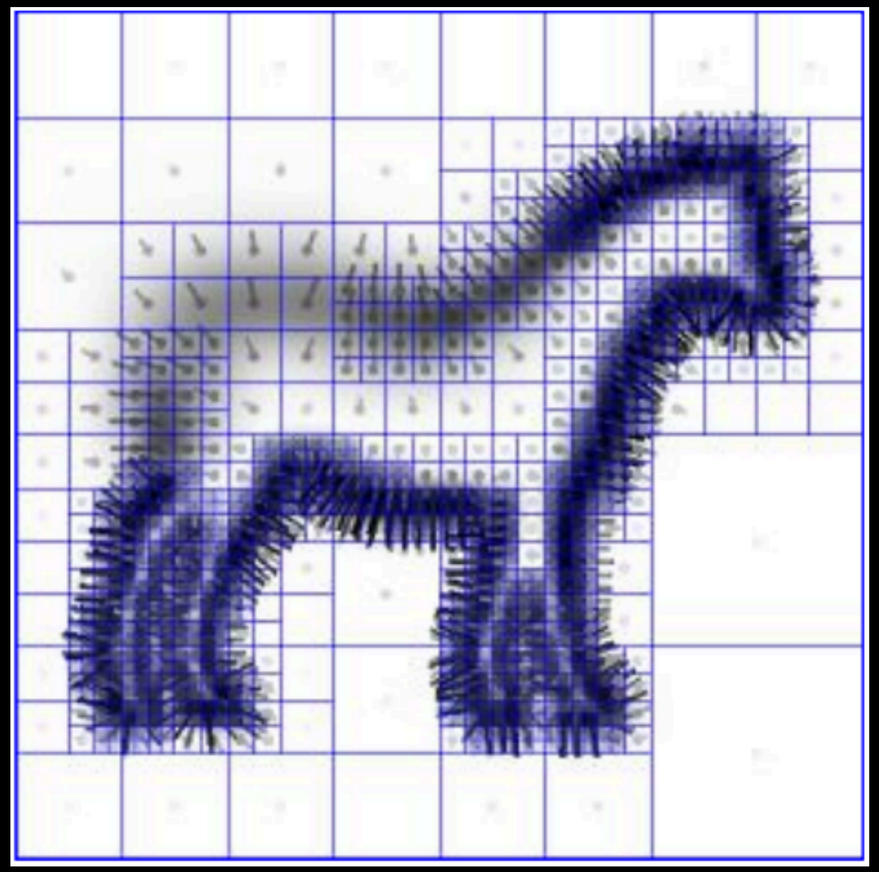
INTEGRATION

- Given the Points:
  - Set octree
  - Compute vector field
    - define function space
    - splat the samples



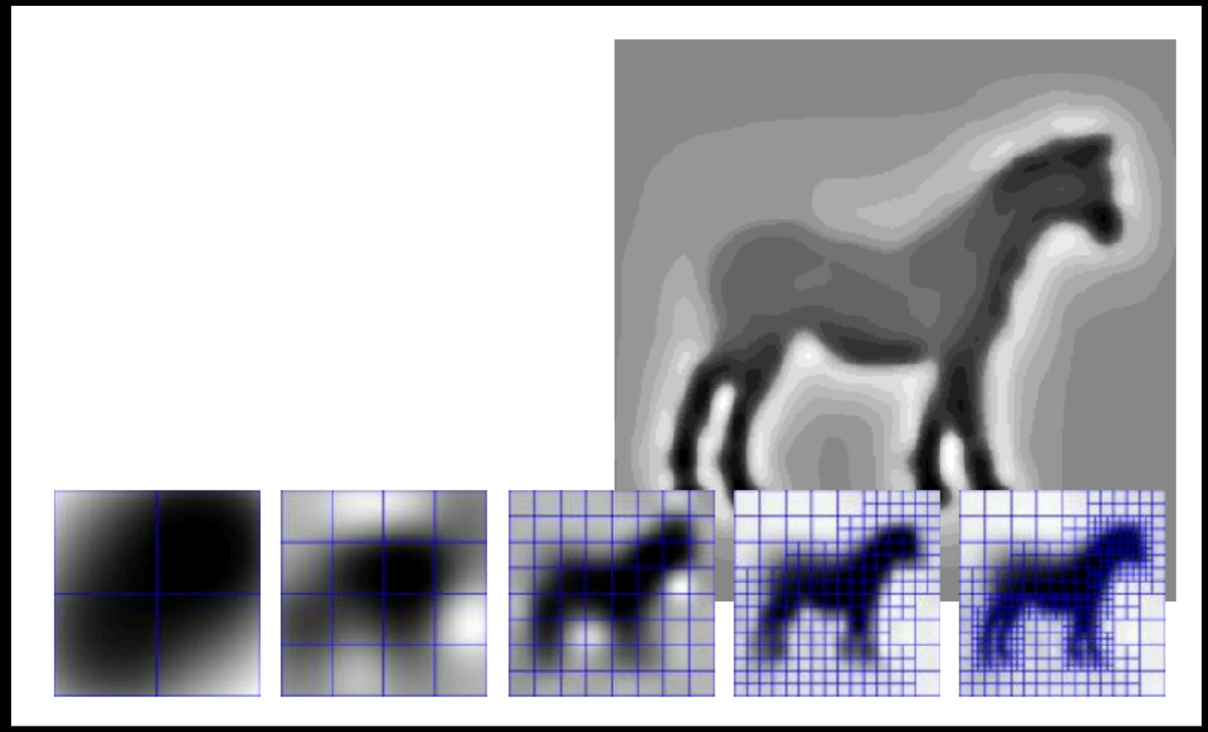
INTEGRATION

- Given the Points:
  - Set octree
  - Compute vector field
    - define function space
    - splat the samples



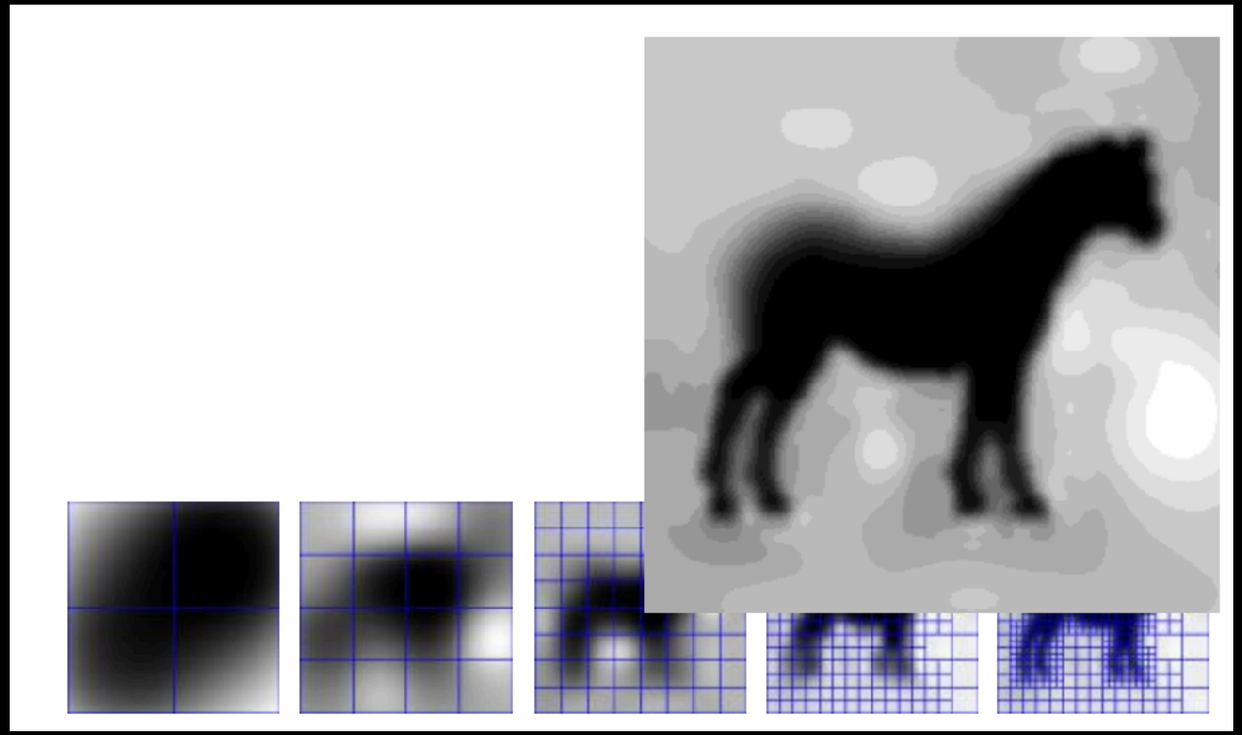
INTEGRATION

- Given the Points:
  - Set octree
  - Compute vector field
  - Compute indicator function
    - Solve Poisson Equation



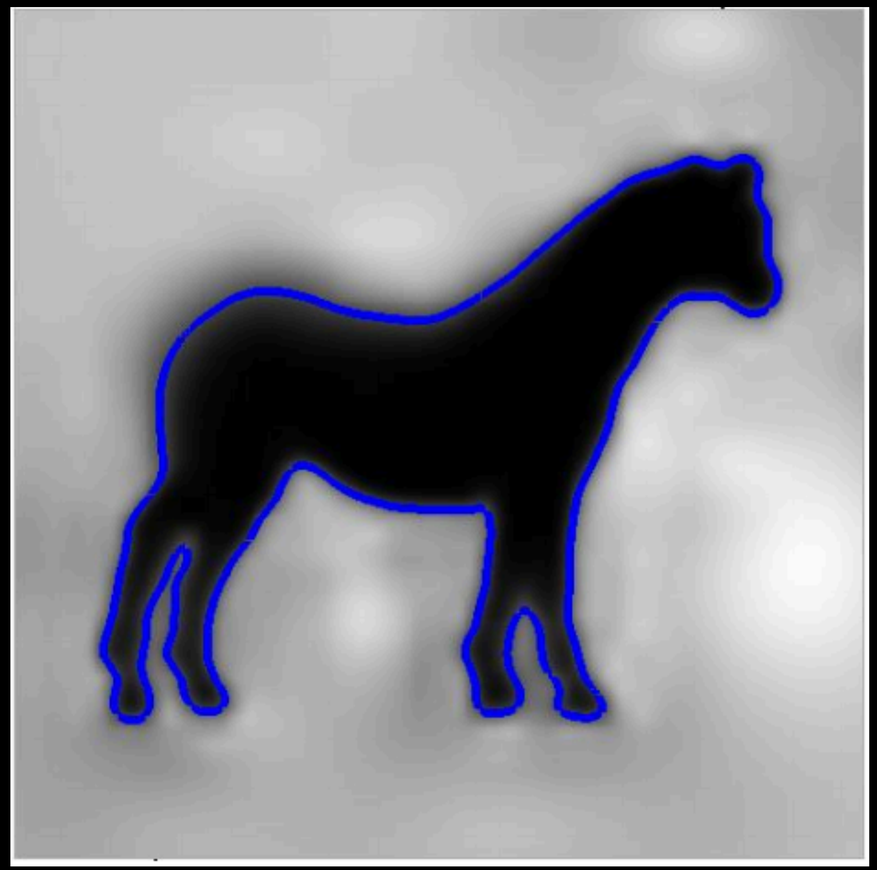
INTEGRATION

- Given the Points:
  - Set octree
  - Compute vector field
  - Compute indicator function
    - Compute divergence
    - Solve Poisson Equation



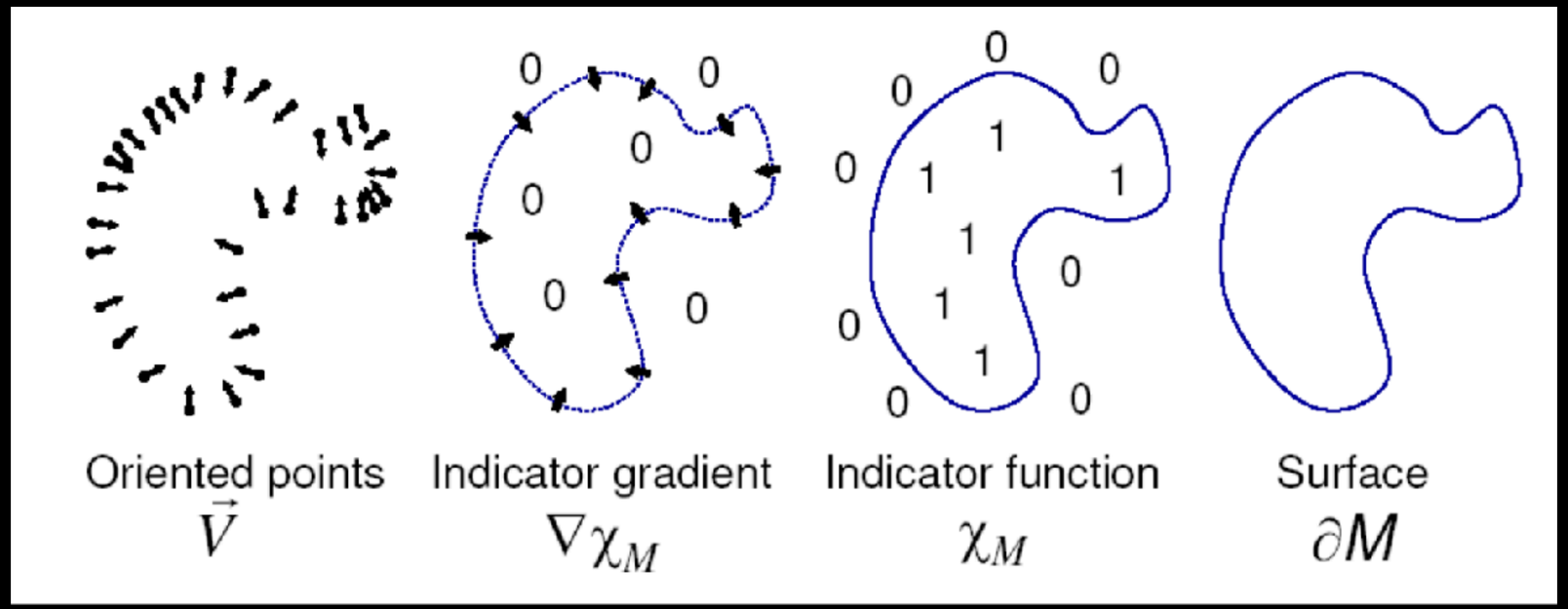
INTEGRATION

- Given the Points:
  - Set octree
  - Compute vector field
  - Compute indicator function
  - Extract iso-surface



PROBLEM MODELING

- Input: points with associated normals
- • Indicator function: 1 inside, 0 outside
- ➢ => gradient = 0 everywhere except near surface



POISSON RECONSTRUCTION

- Problem: find the indicator function starting
- from the gradient
- - $\min_{\chi} \|\vec{\nabla}\chi - \vec{V}\|$
- – V gradient field defined by the points
- Transforms to a Poisson equation:

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

POISSON RECONSTRUCTION

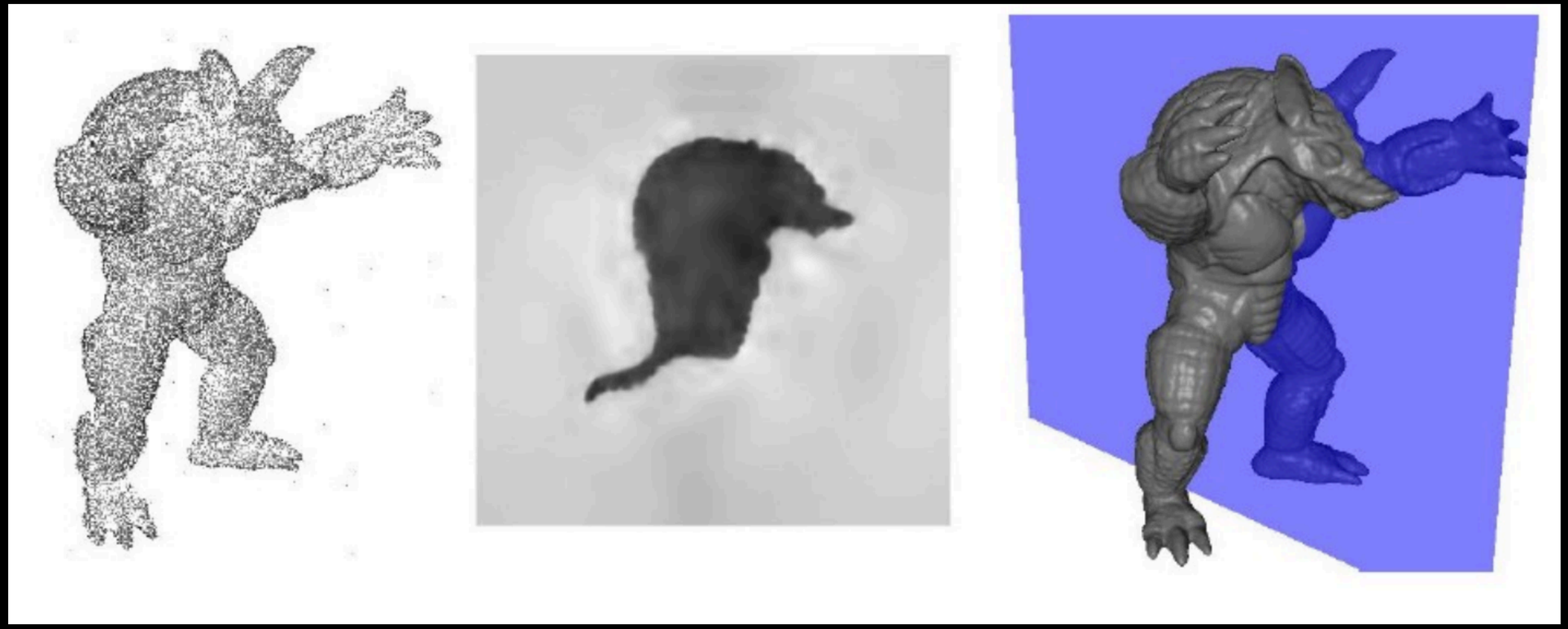
- Basis functions with local support (≠ RBF)
- ➢ => sparse system, fast to solve
- Implicit function constrained everywhere, not
- only near input points
- Good result even for noisy data
- Main drawback: consistent normal orientation

KAZHDAN ET AL. SGP 2006

- “Poisson surface reconstruction”
- Discretization of space: not a uniform grid (Hoppe et al.), but an adaptive octree (Ohtake et al.)
- Time and memory complexity for a given octree depth = O(n)
- Octree depth += 1 => time and memory complexity + number of output triangles ~ multiplied by 4

TEXT

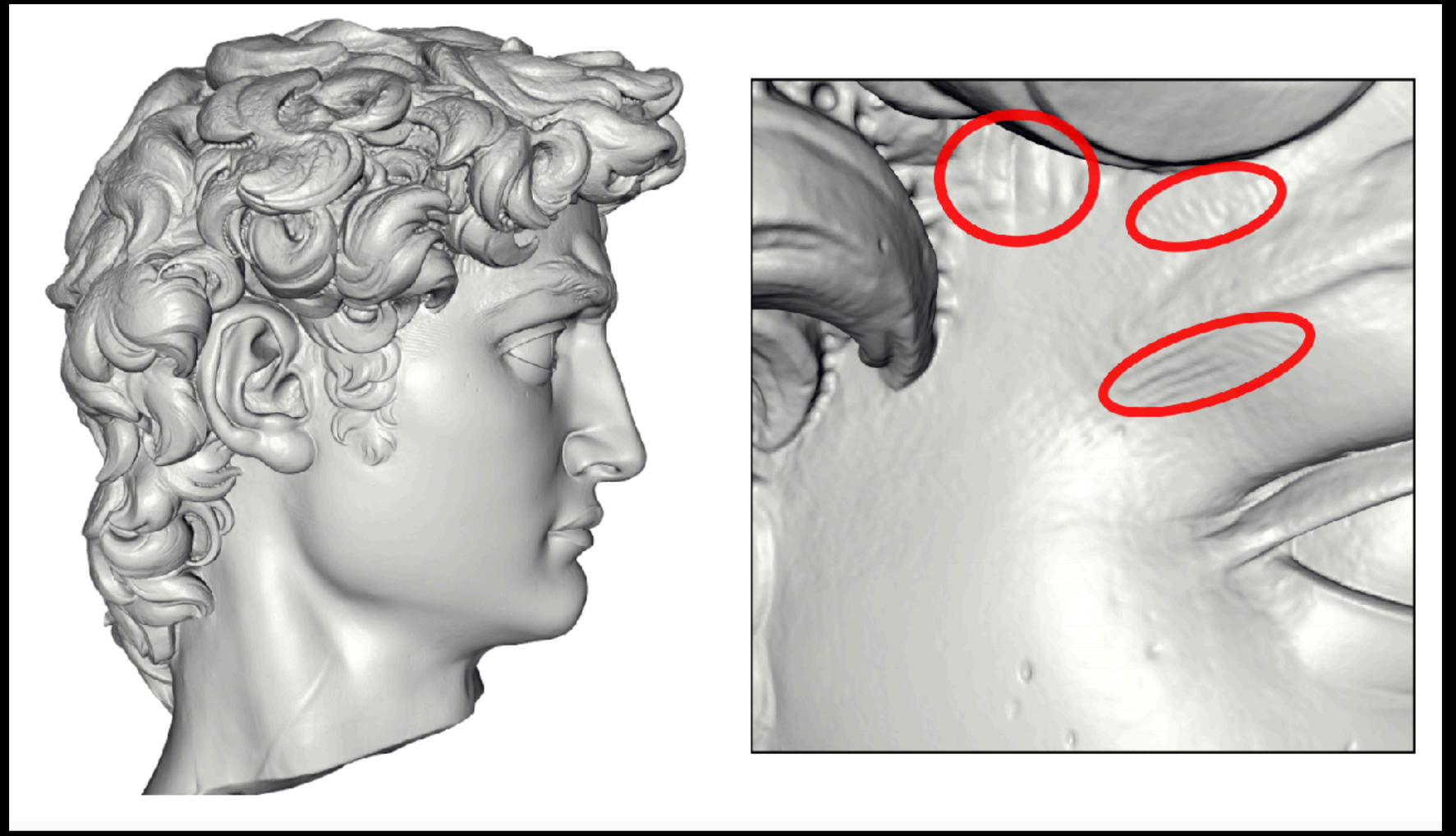
KAZHDAN ET AL. SGP 2006



MICHELANGELO'S DAVID

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

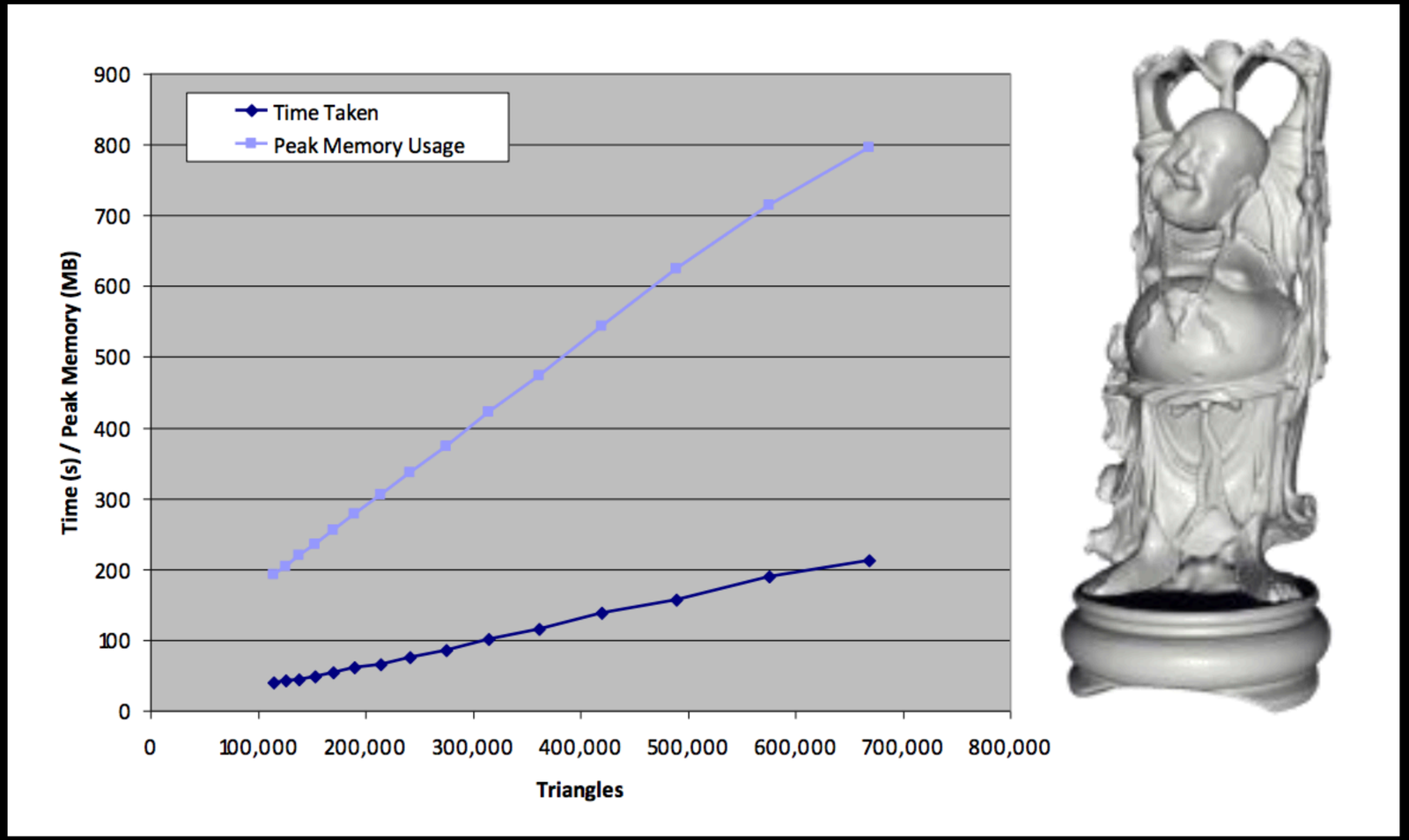
MICHELANGELO'S DAVID – CHISEL MARKS



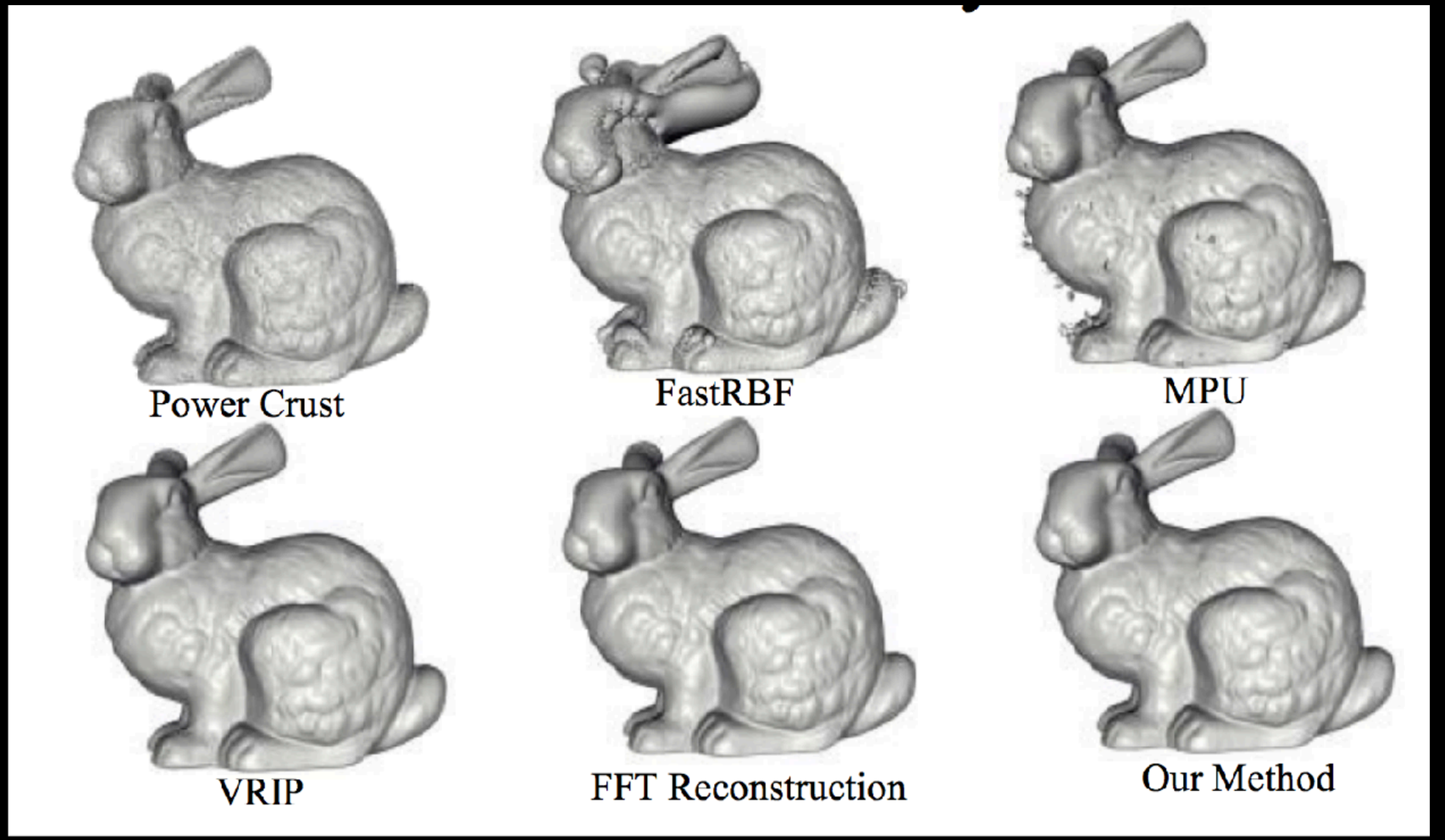
MICHELANGELO'S DAVID – DRILL MARKS

MICHELANGELO'S DAVID – EYE

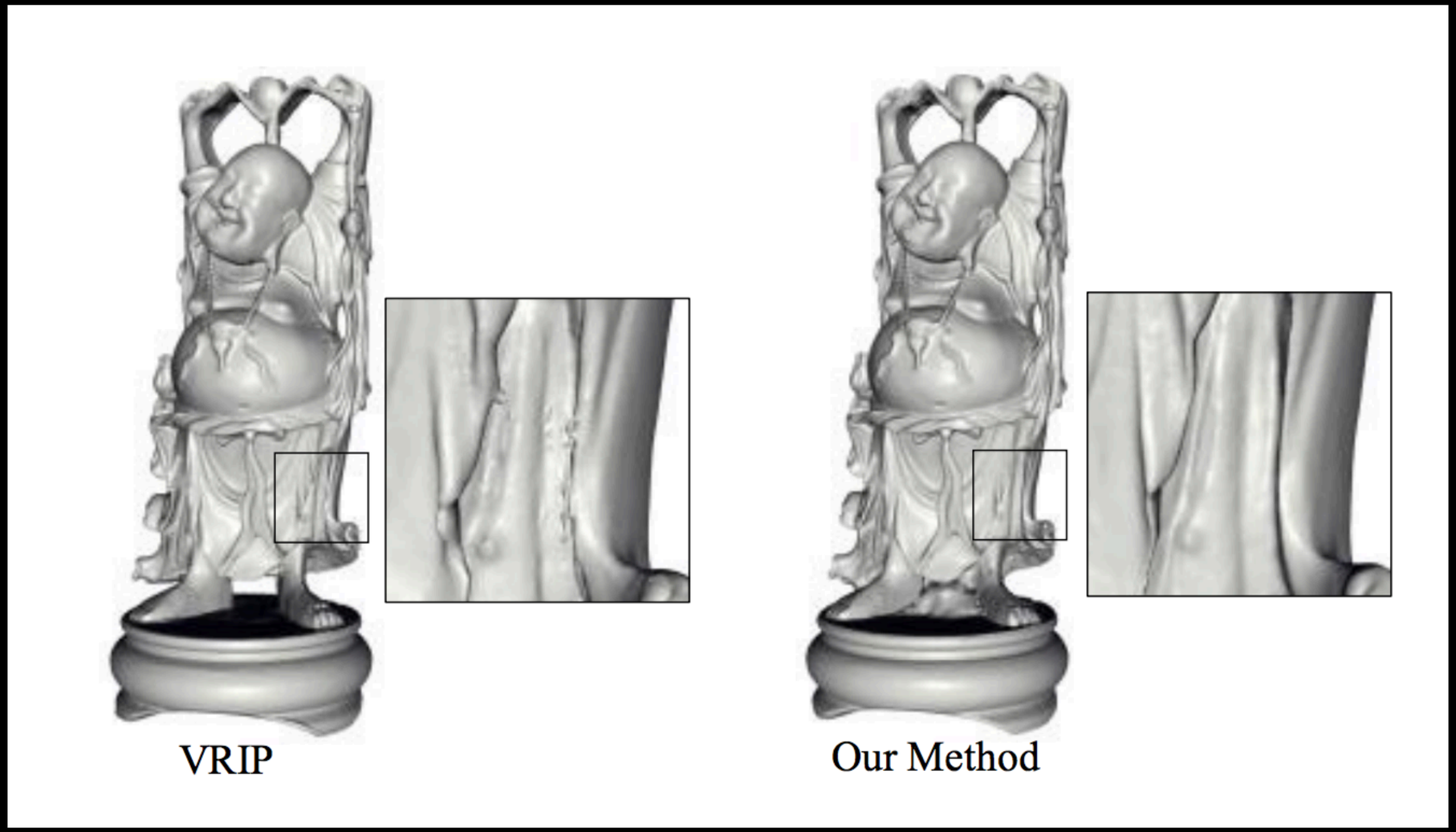
SCALABILITY – BUDDHA MODEL



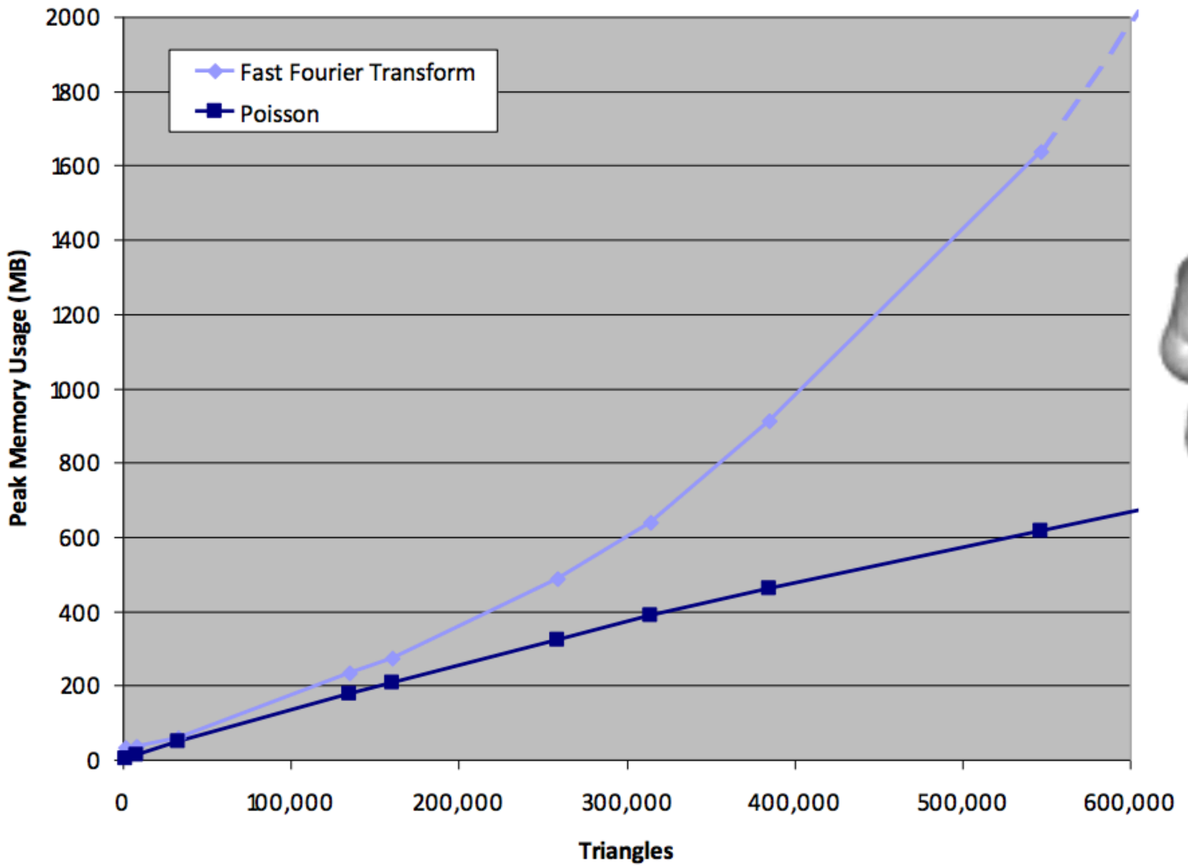
STANFORD BUNNY MODEL



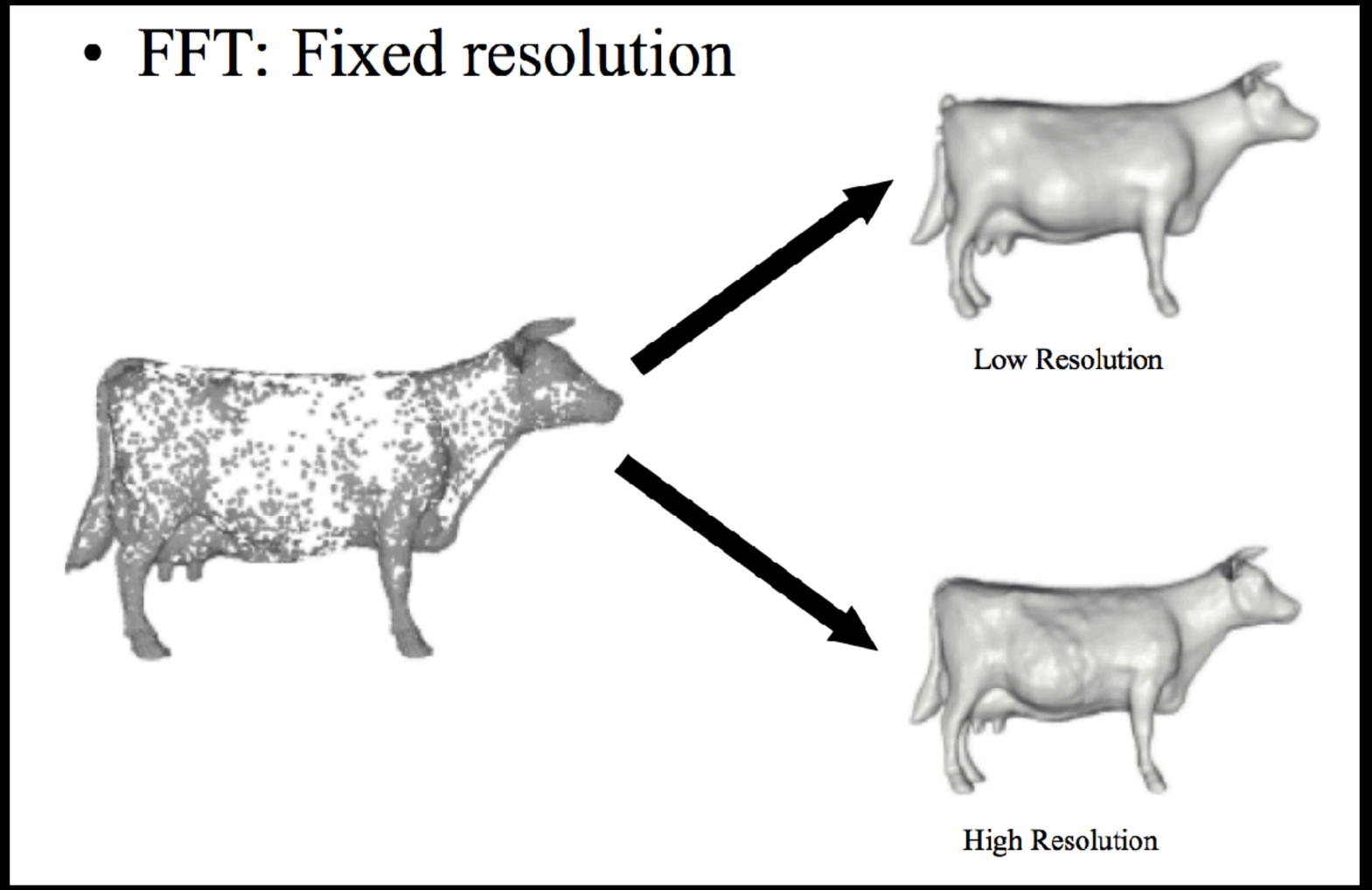
VRIP COMPARISON



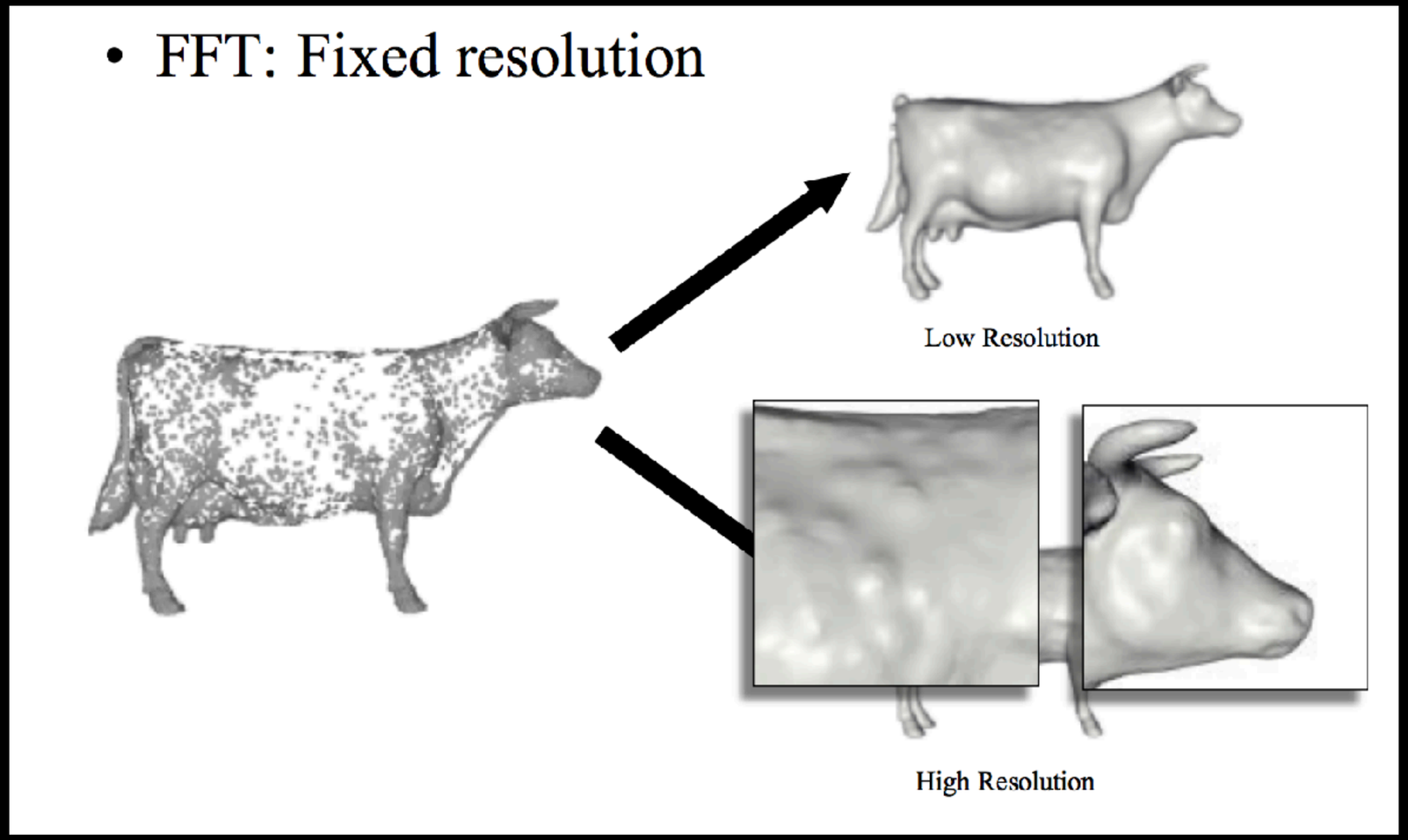
FFT RECONSTRUCTION COMPARISON



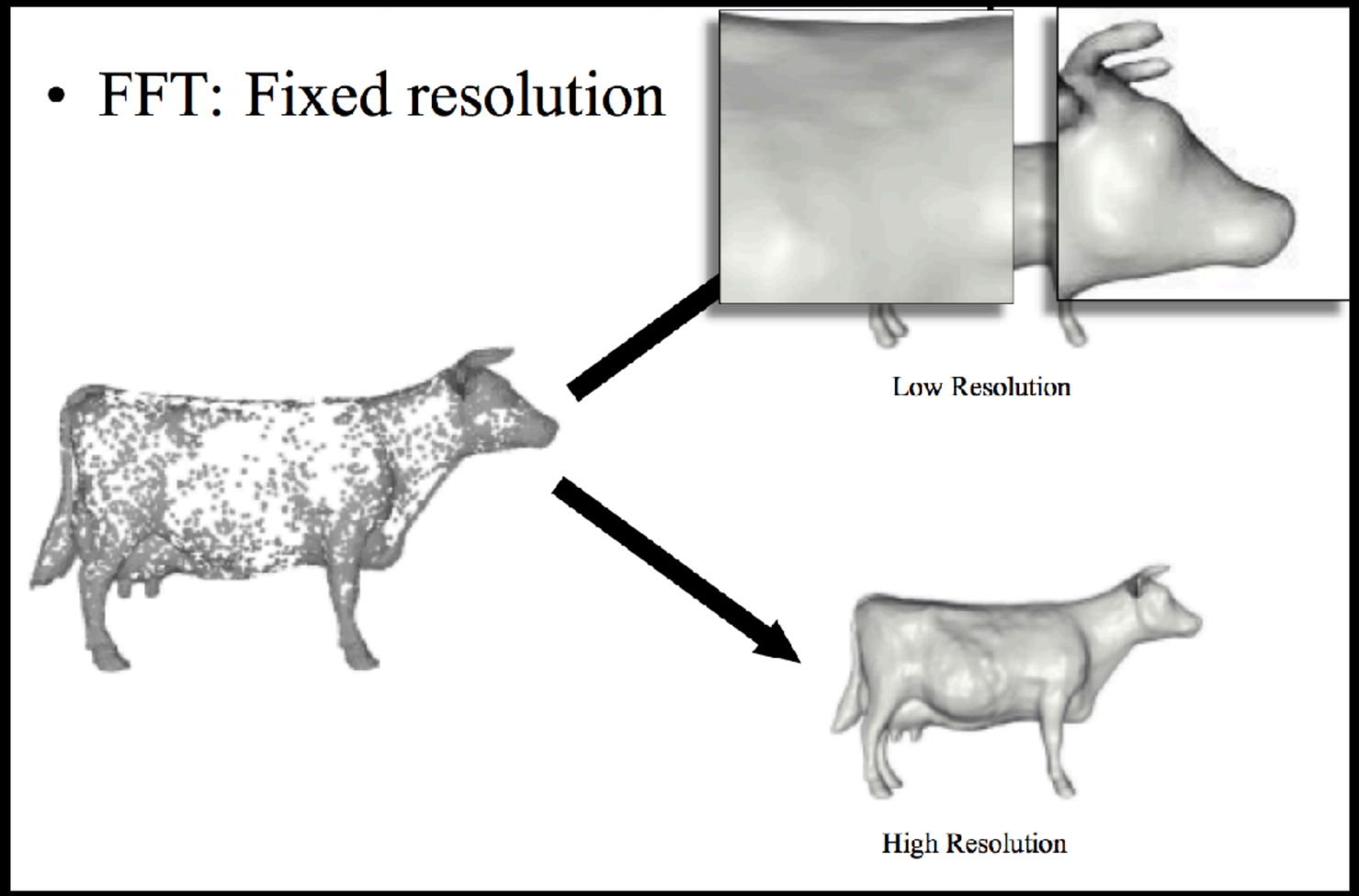
FFT RECONSTRUCTION COMPARISON



FFT RECONSTRUCTION COMPARISON

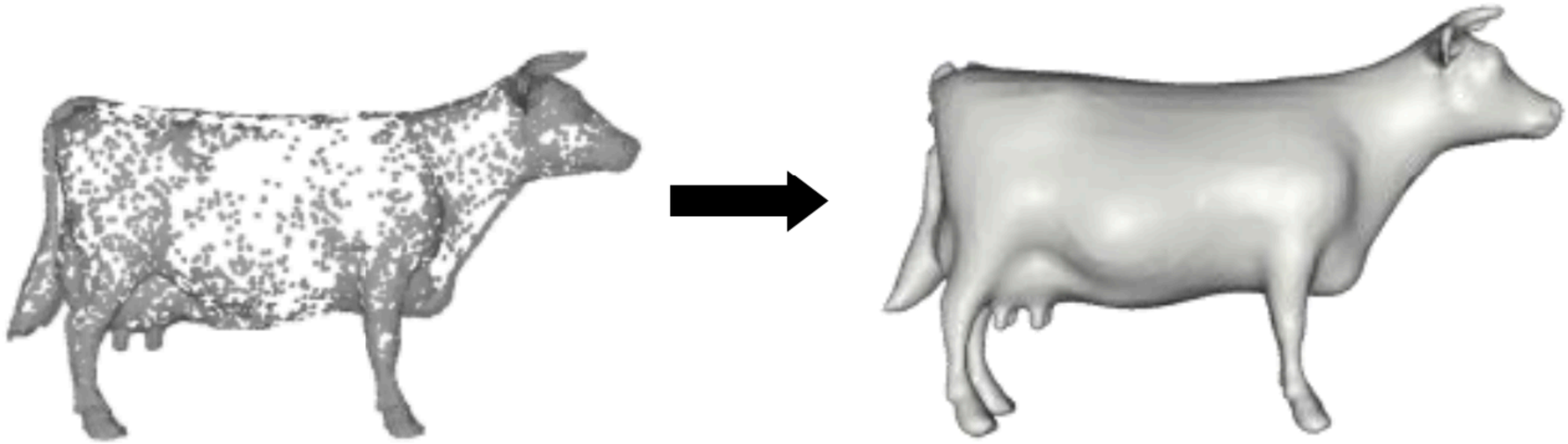


FFT RECONSTRUCTION COMPARISON



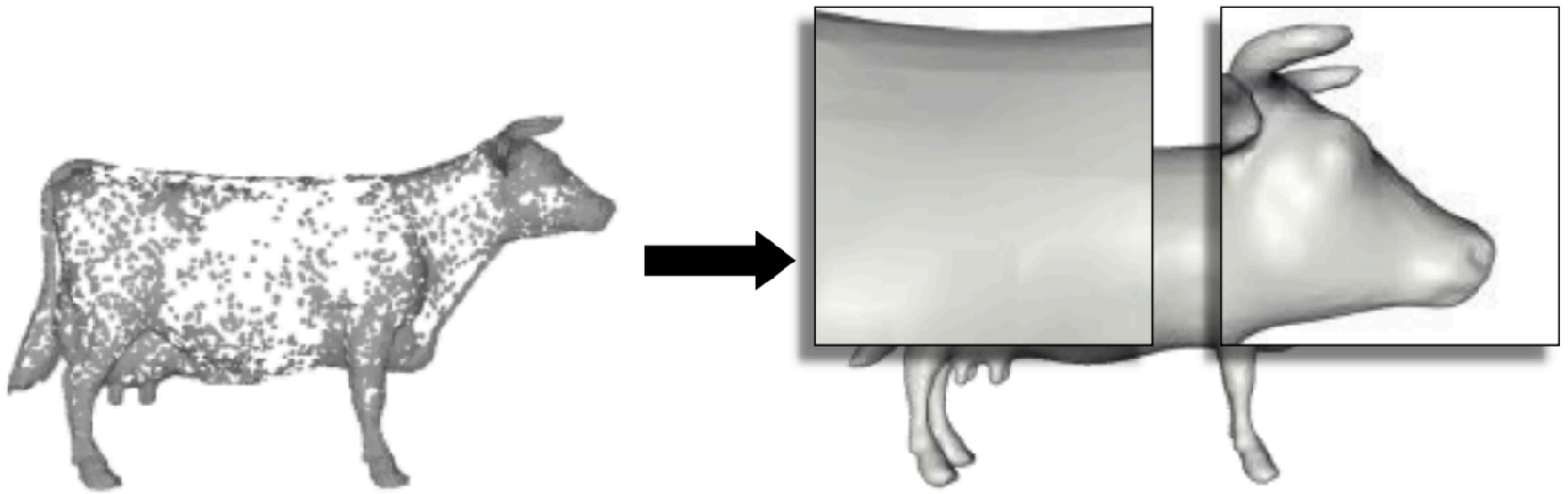
FFT RECONSTRUCTION COMPARISON

- Poisson: Adaptive resolution



FFT RECONSTRUCTION COMPARISON

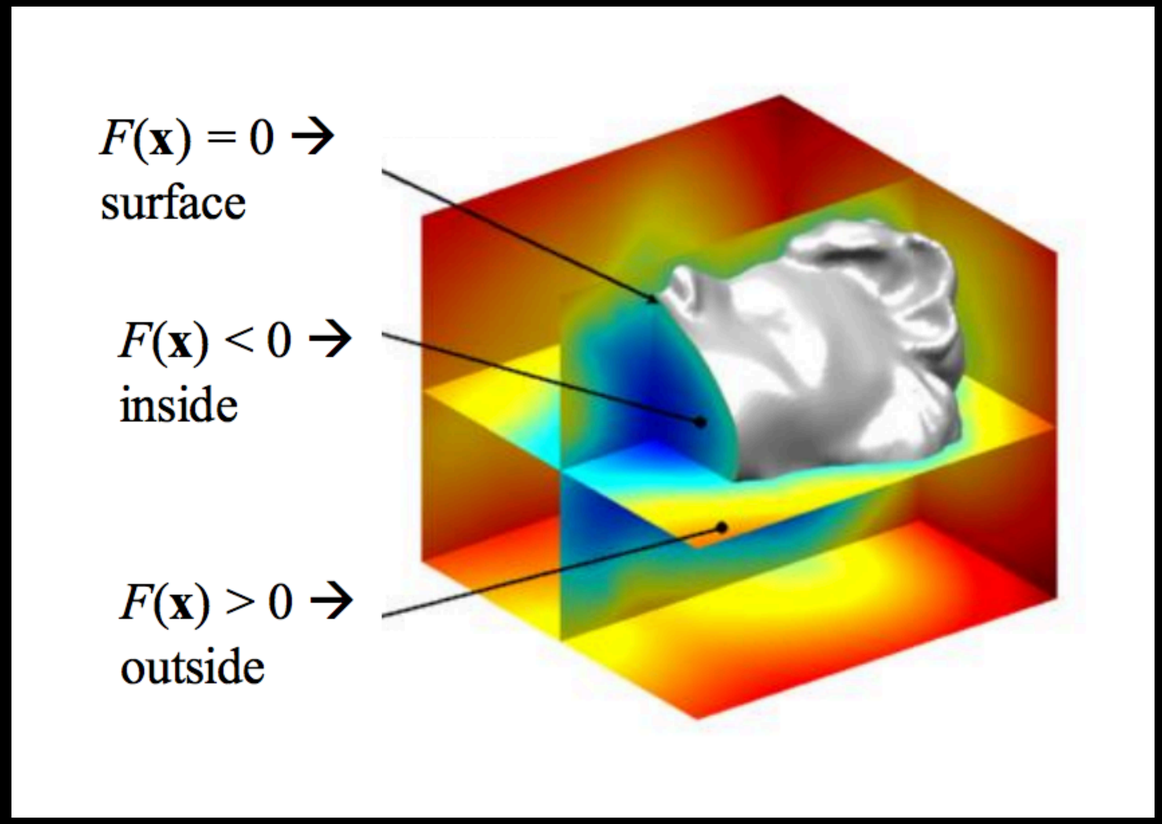
- Poisson: Adaptive resolution



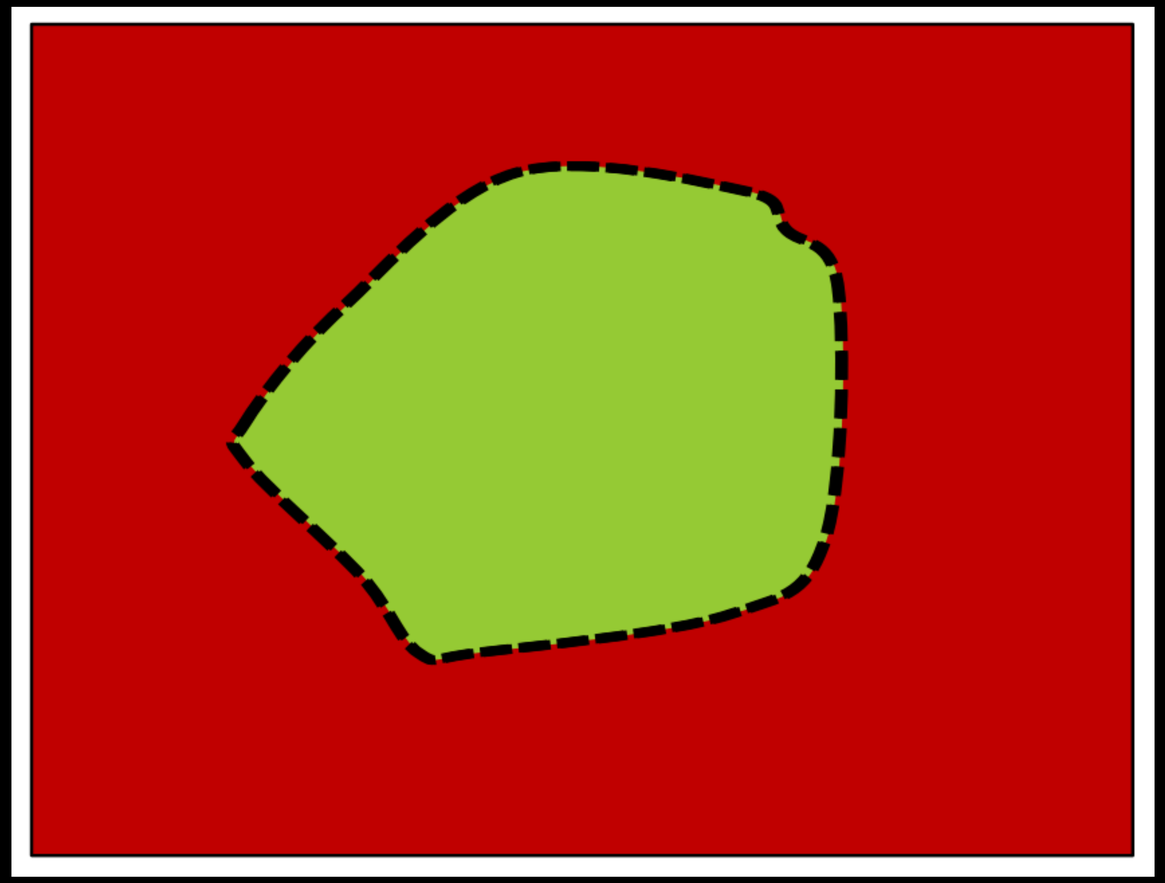
EXTRACTING THE SURFACE

EXTRACTING THE SURFACE

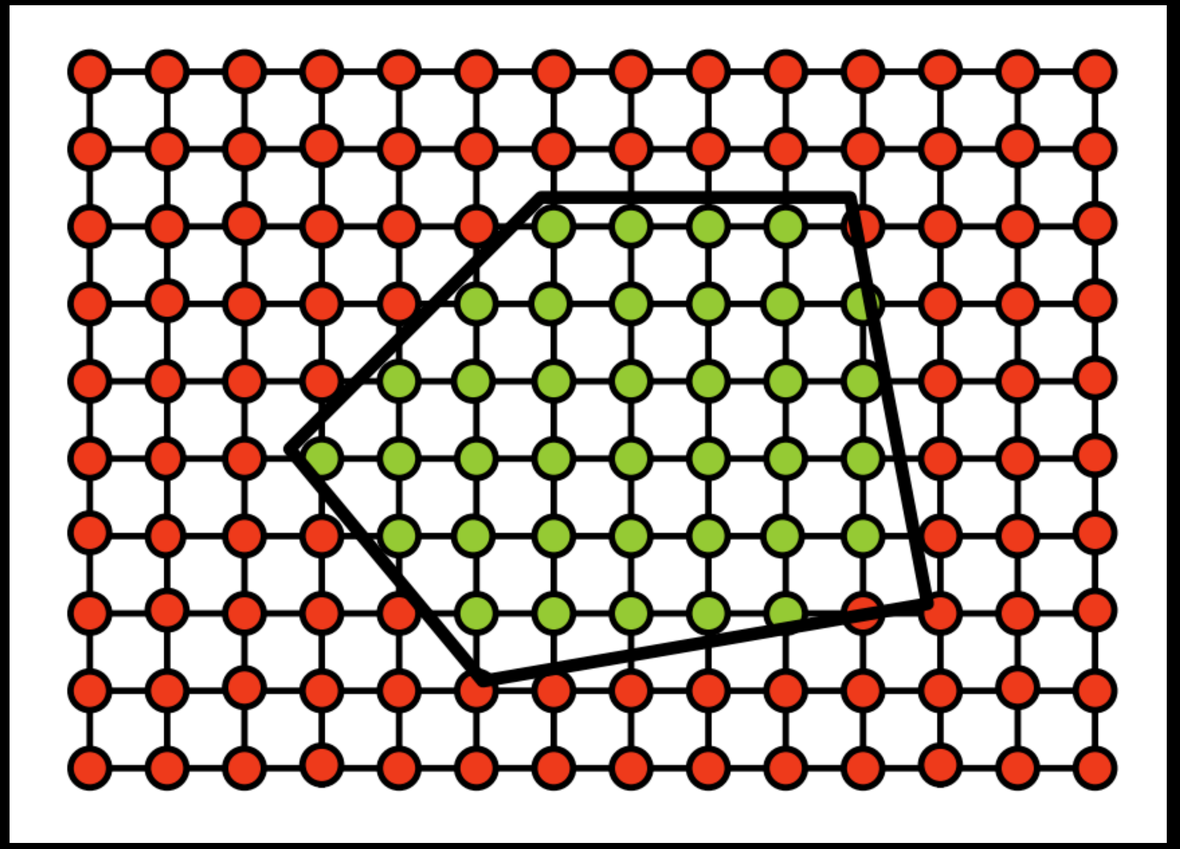
- How do we make a manifold mesh of the SDF?



EXTRACTING THE SURFACE

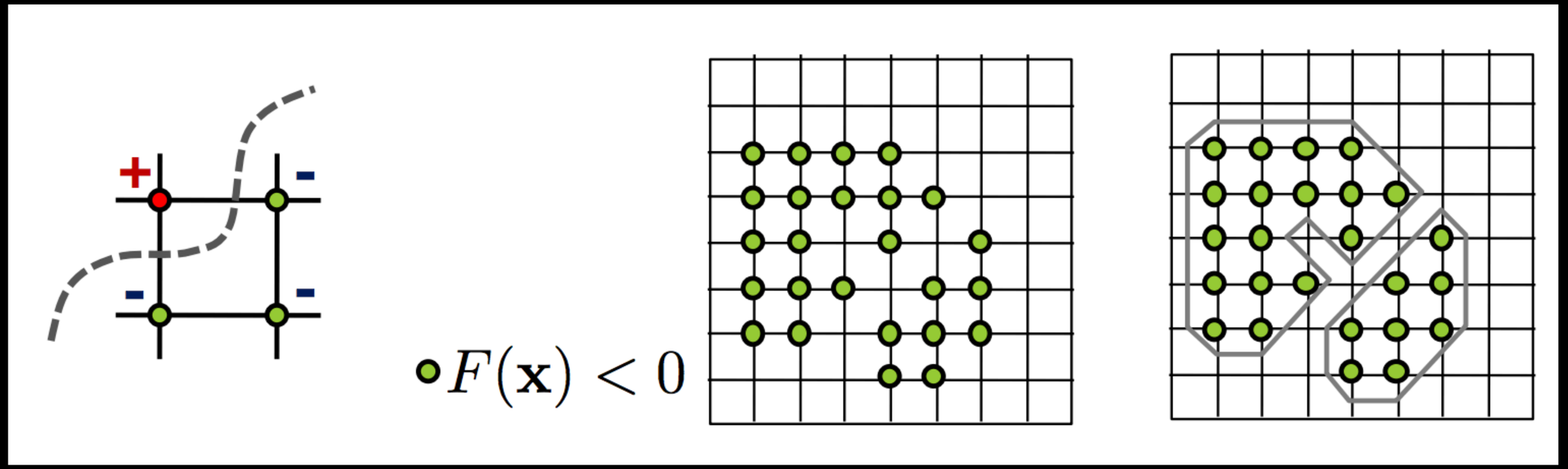


EXTRACTING THE SURFACE



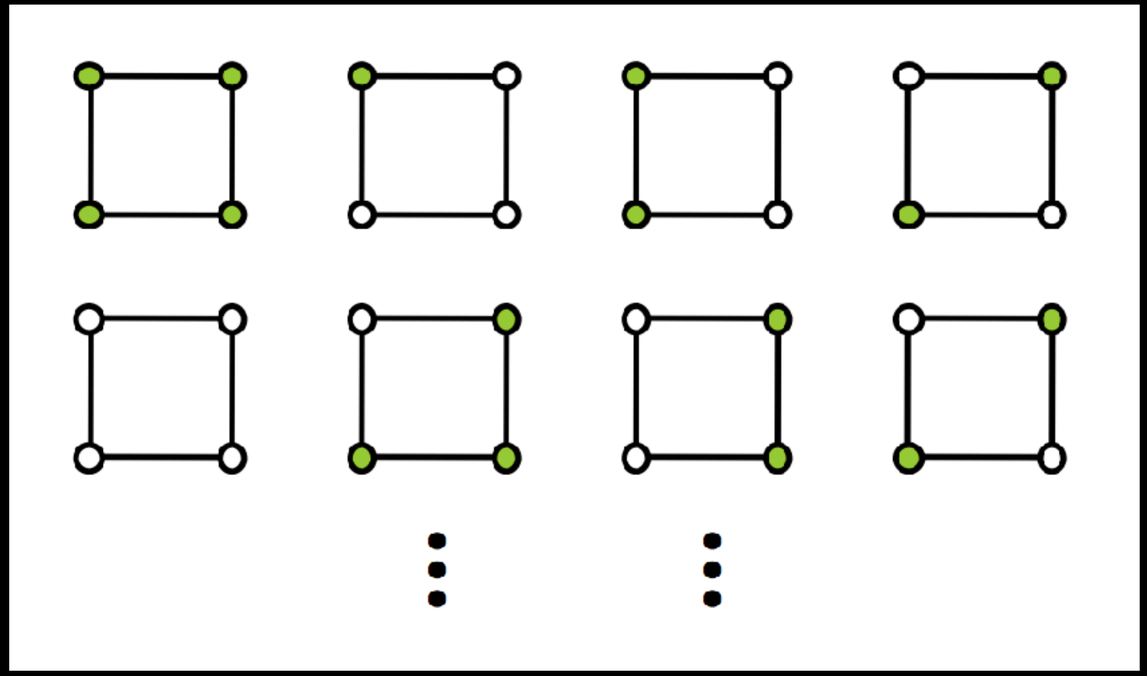
TESSELLATION

- Idea: sample and assume function is linear within the sampled grid.
- Zero set passes only where different signs exist in a grid cell.



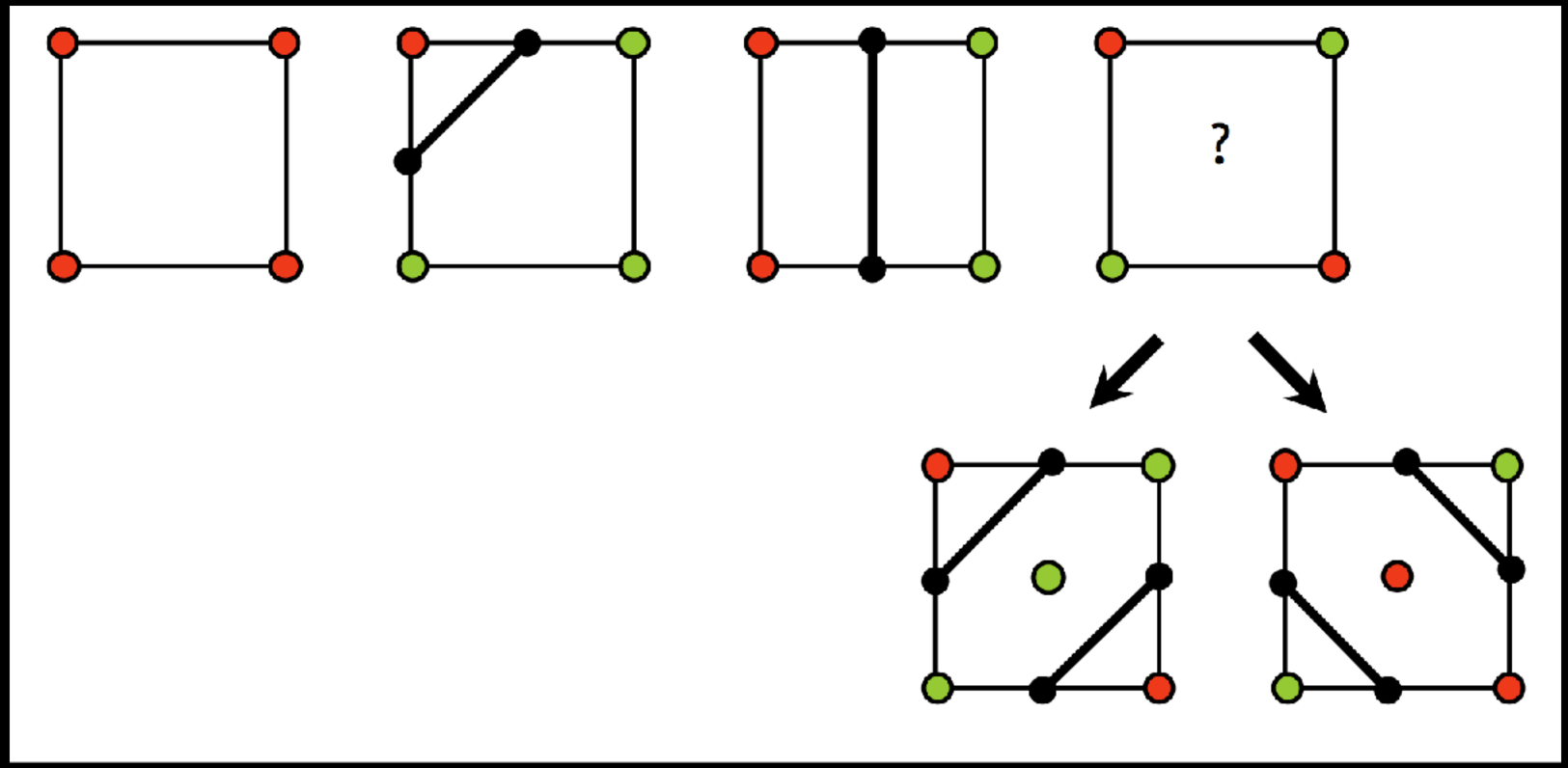
2D MARCHING SQUARES

- 16 different configurations in 2D.
- 4 equivalence classes (up to rotational and reflection symmetry + complement).



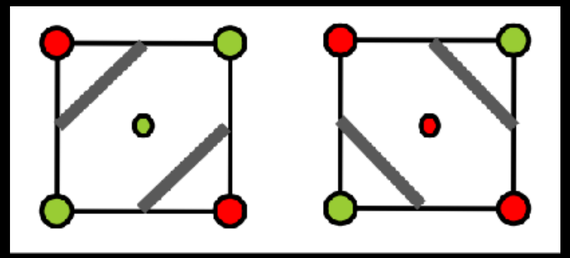
TESSELLATION IN 2D

- 4 equivalence classes (up to rotational and reflection symmetry + complement)

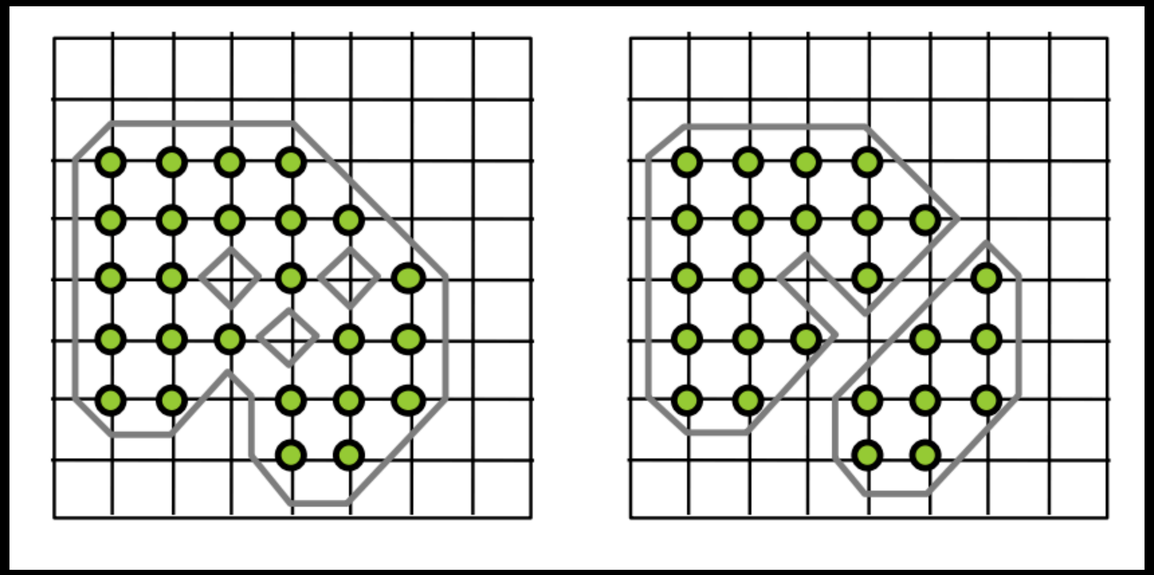


TESSELLATION IN 2D

- Case 4 is ambiguious:



- Choose arbitrarilty; but consistently.
- To avoid problems with the resulting mesh.

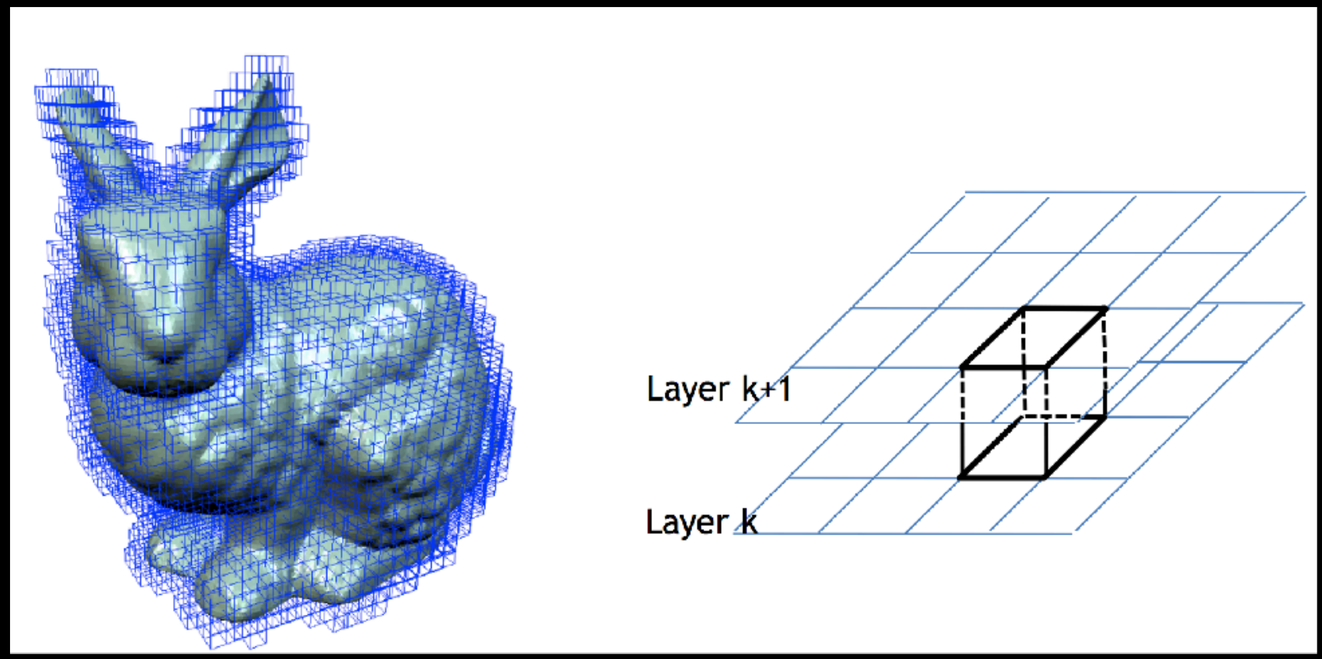


3D: MARCHING CUBES

- Classify grid nodes as inside/outside inside/outside
- Classify cell: 2^8 configurations
- Linear interpolation along edges
- Look−up table for patch configuration
  - Disambiguation more complicated

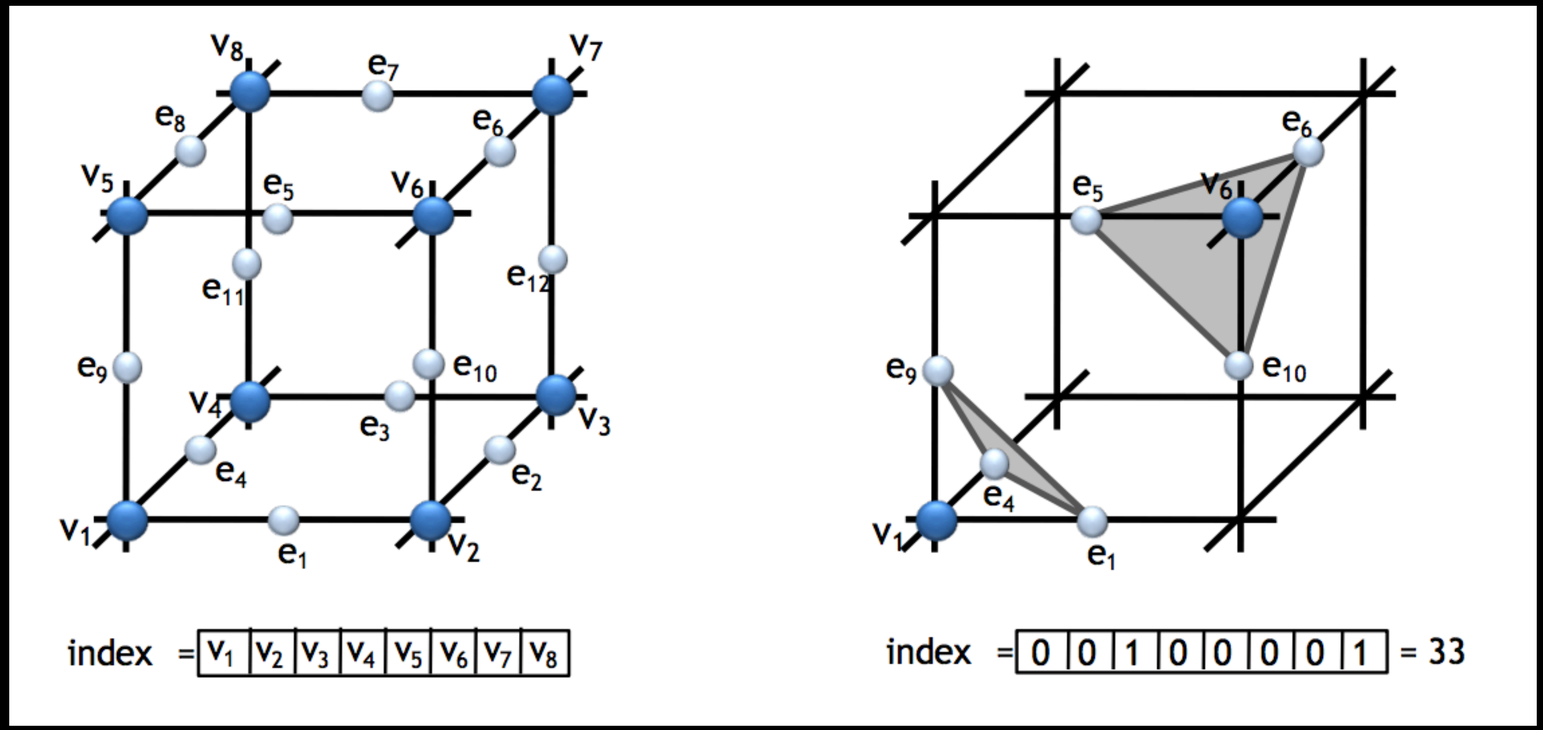
3D: MARCHING CUBES

- Cell classification:
  - Inside
  - Outside
  - Intersecting



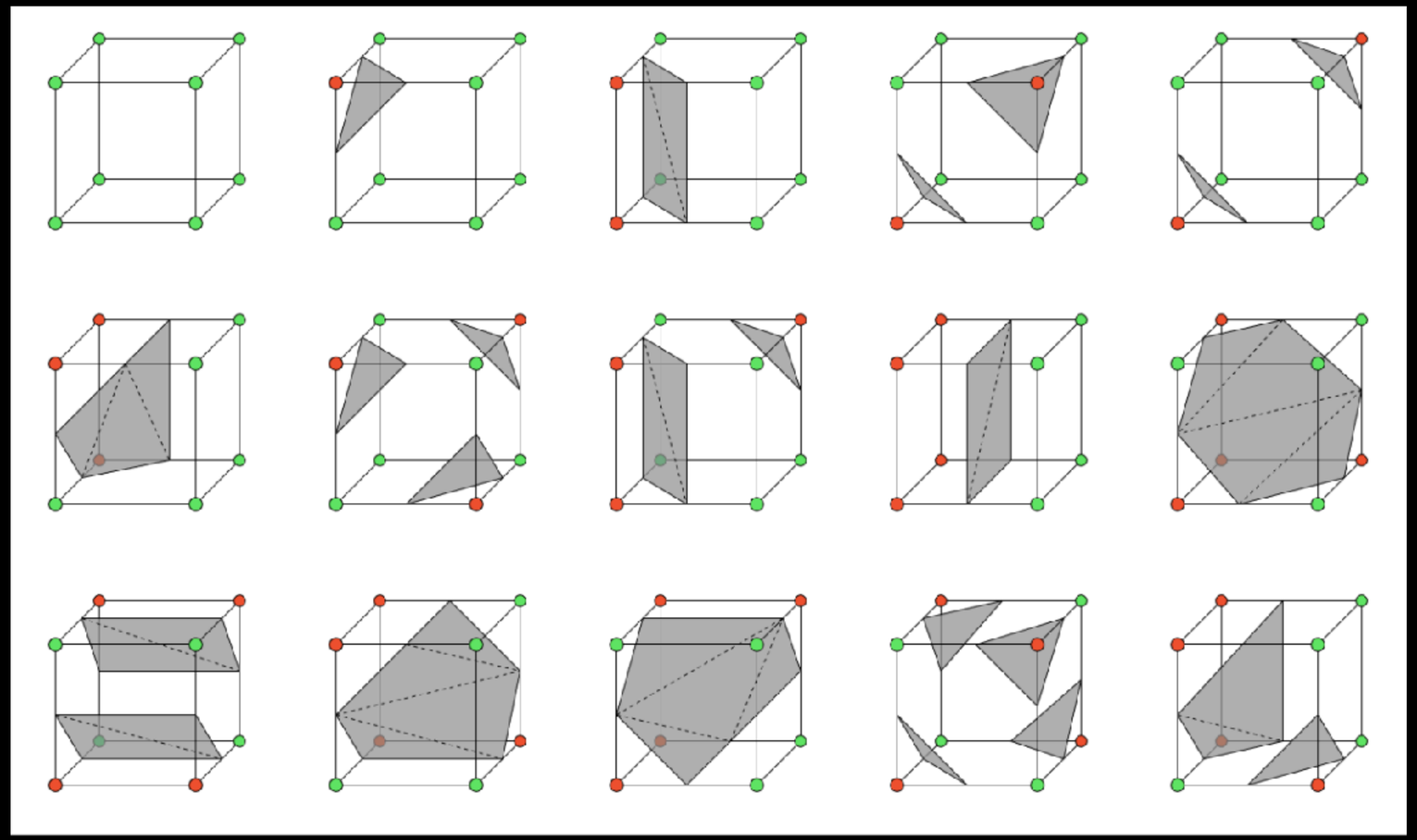
MARCHING CUBES

- Compute case index. We have $2^8 = 256$ cases (0/1 for each of the eight vertices) – can store as 8 bit (1 byte) index.



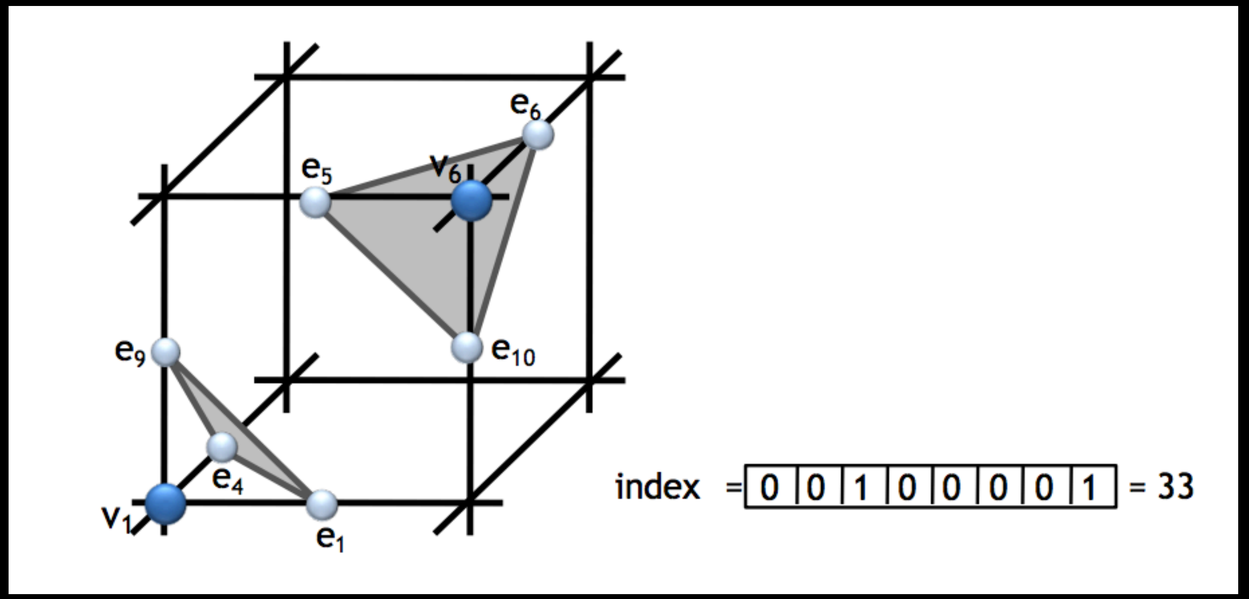
MARCHING CUBES

- Unique cases (by rotation, reflection and complement)

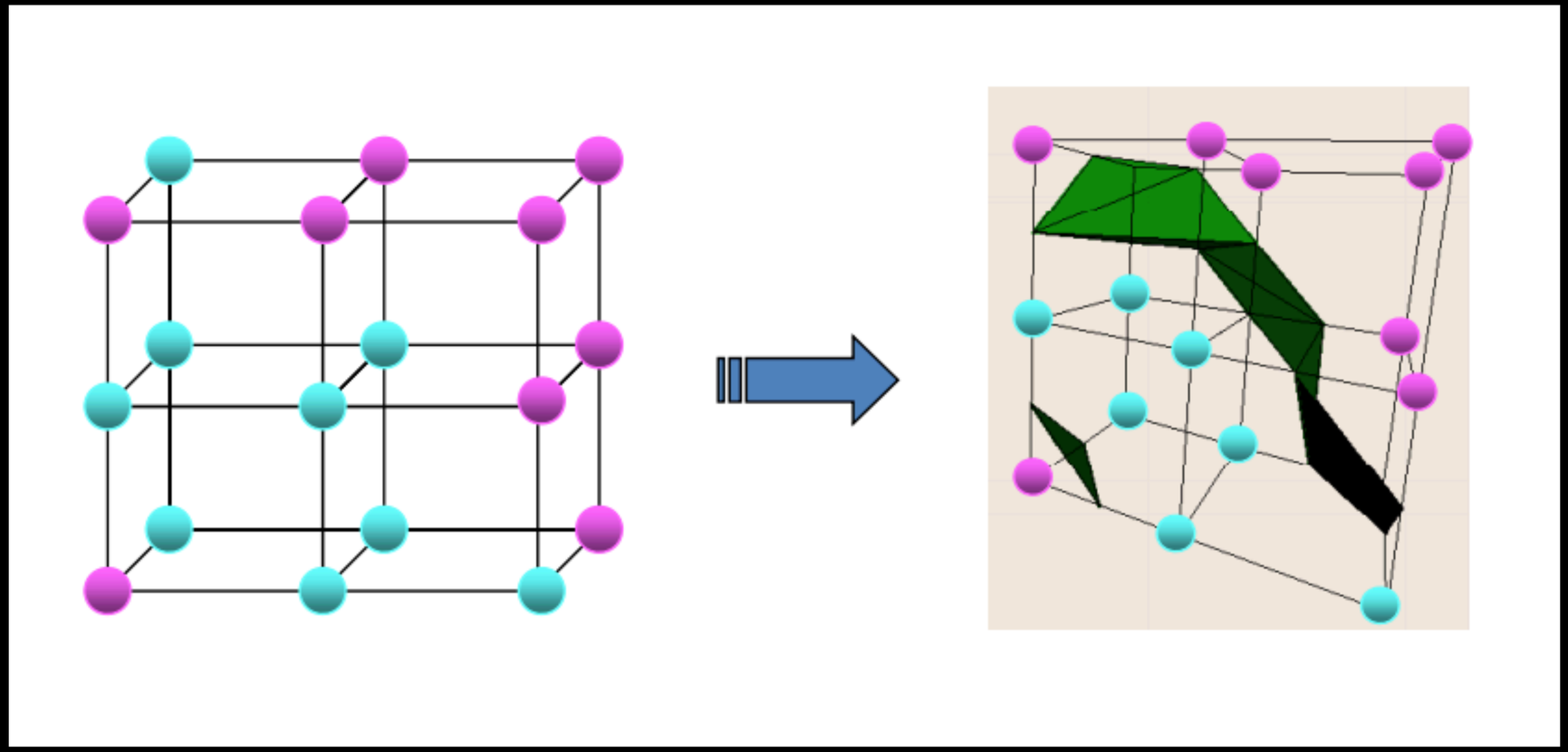


TESSELLATION

- Using the case index, retrieve the connectivity in the look-up table.
- Example: the entry for index 33 indicates:
  - Cut edges are e1; e4; e5; e6; e9 and e10 ;
  - Output triangles are (e1; e9; e4) and (e5; e10; e6).

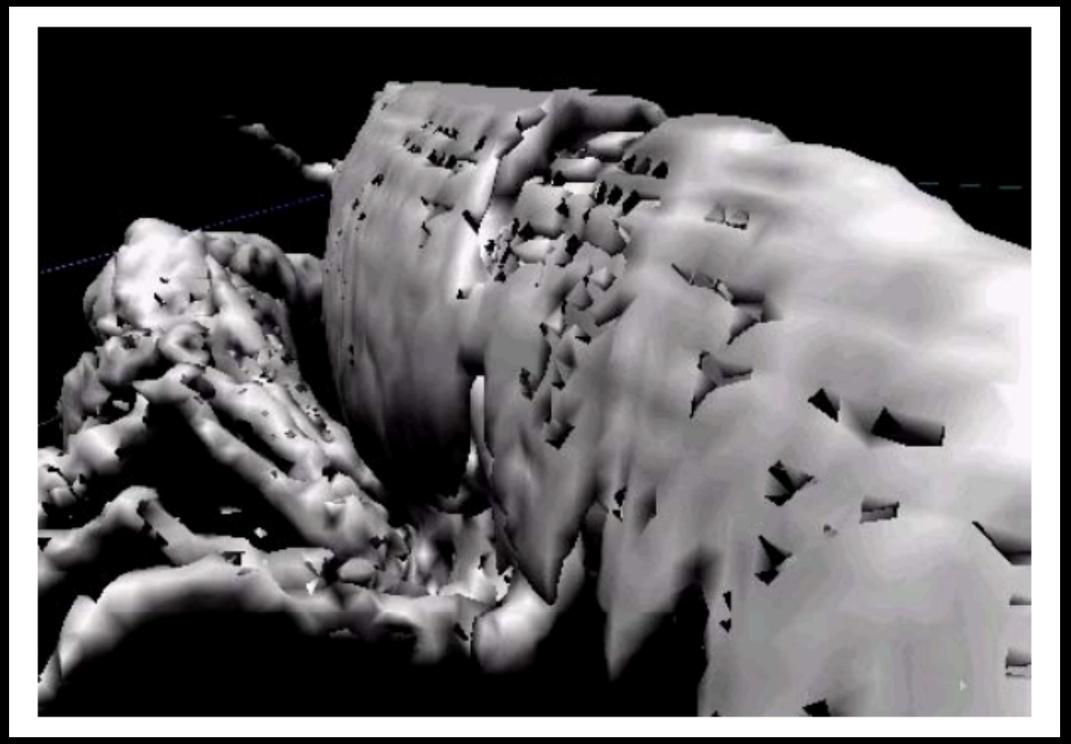


MARCHING CUBES

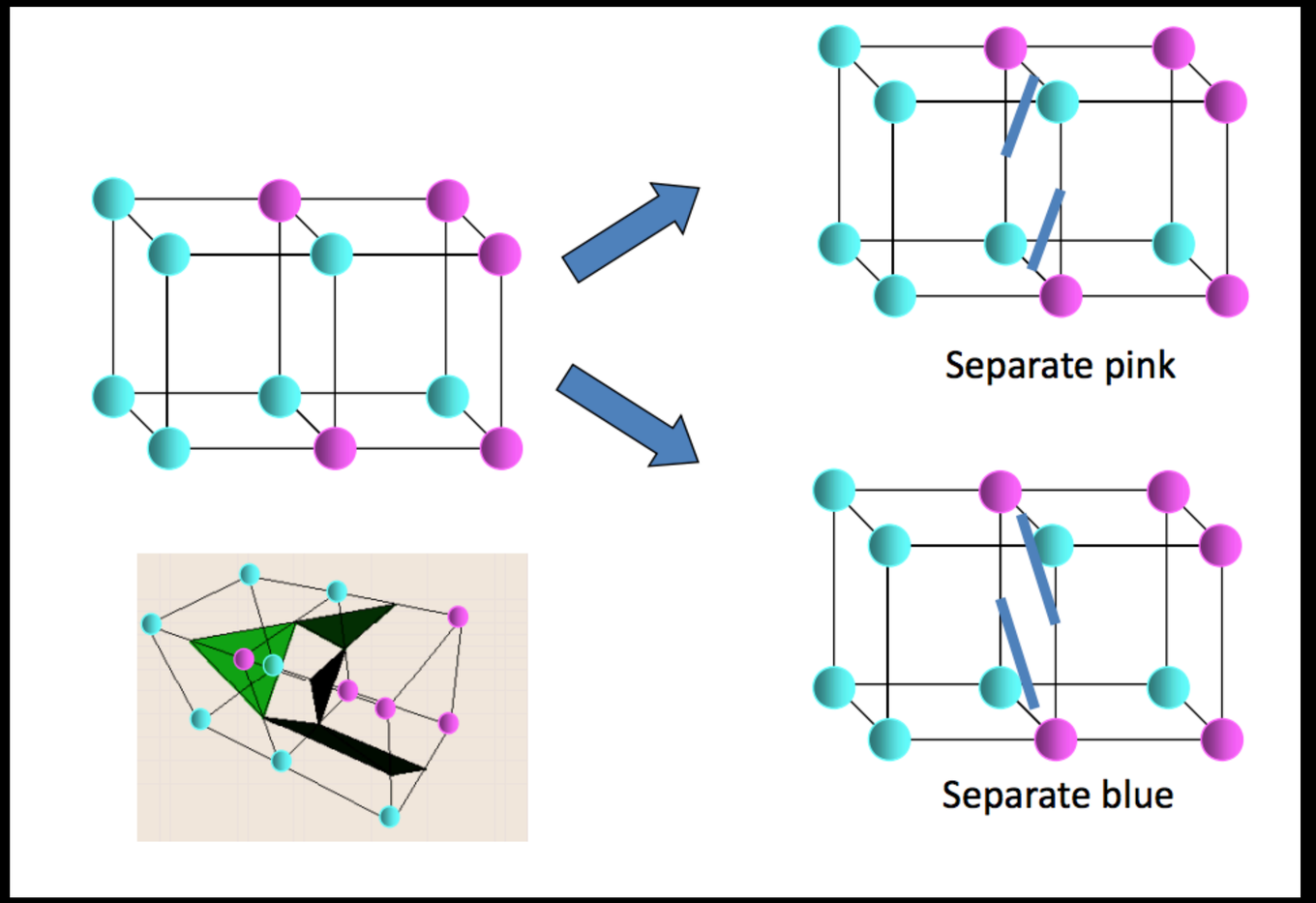


MARCHING CUBES PROBLEMS

- Ambiguity
  - Holes
- Generates HUGE meshes
  - Millions of polygons

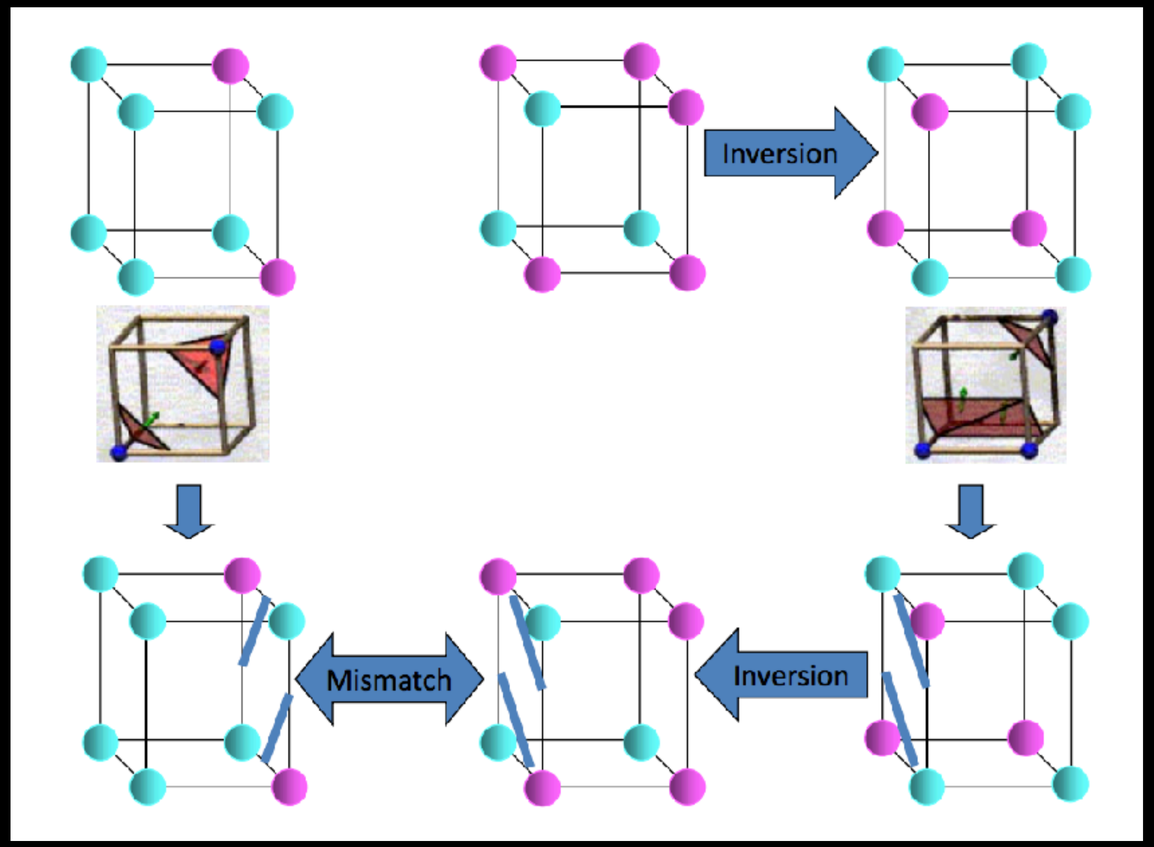


AMBIGUITY



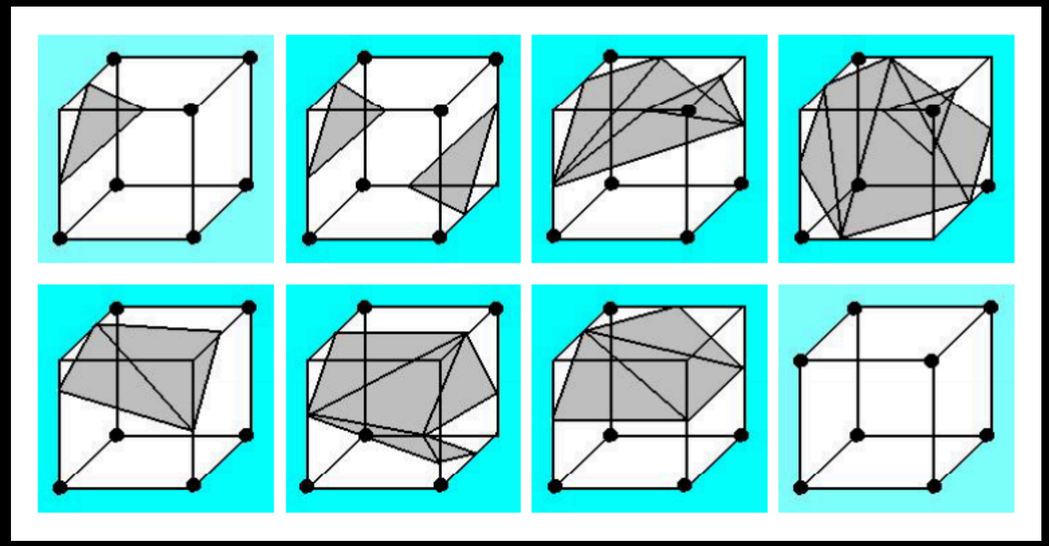
THE INVERSION PROBLEM

- Reduction from 256 to 15 cases includes inversion

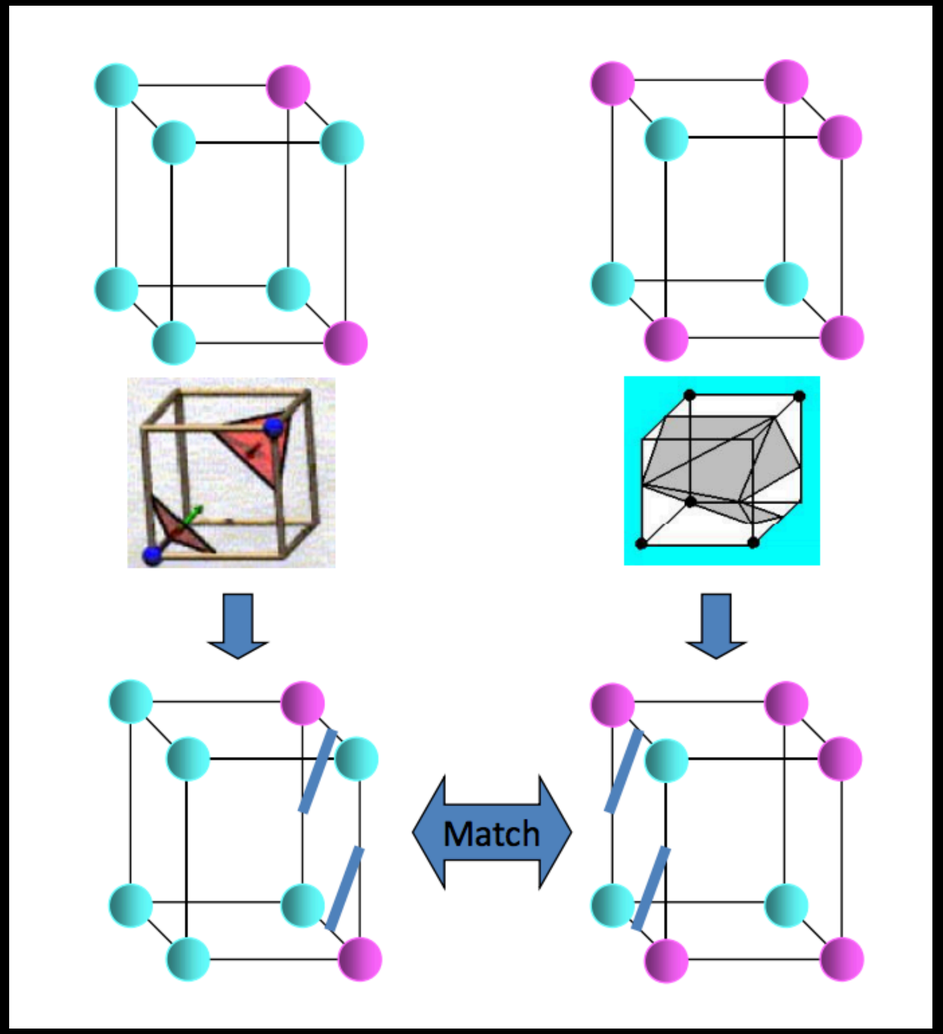


AMBIGUITY SOLUTION

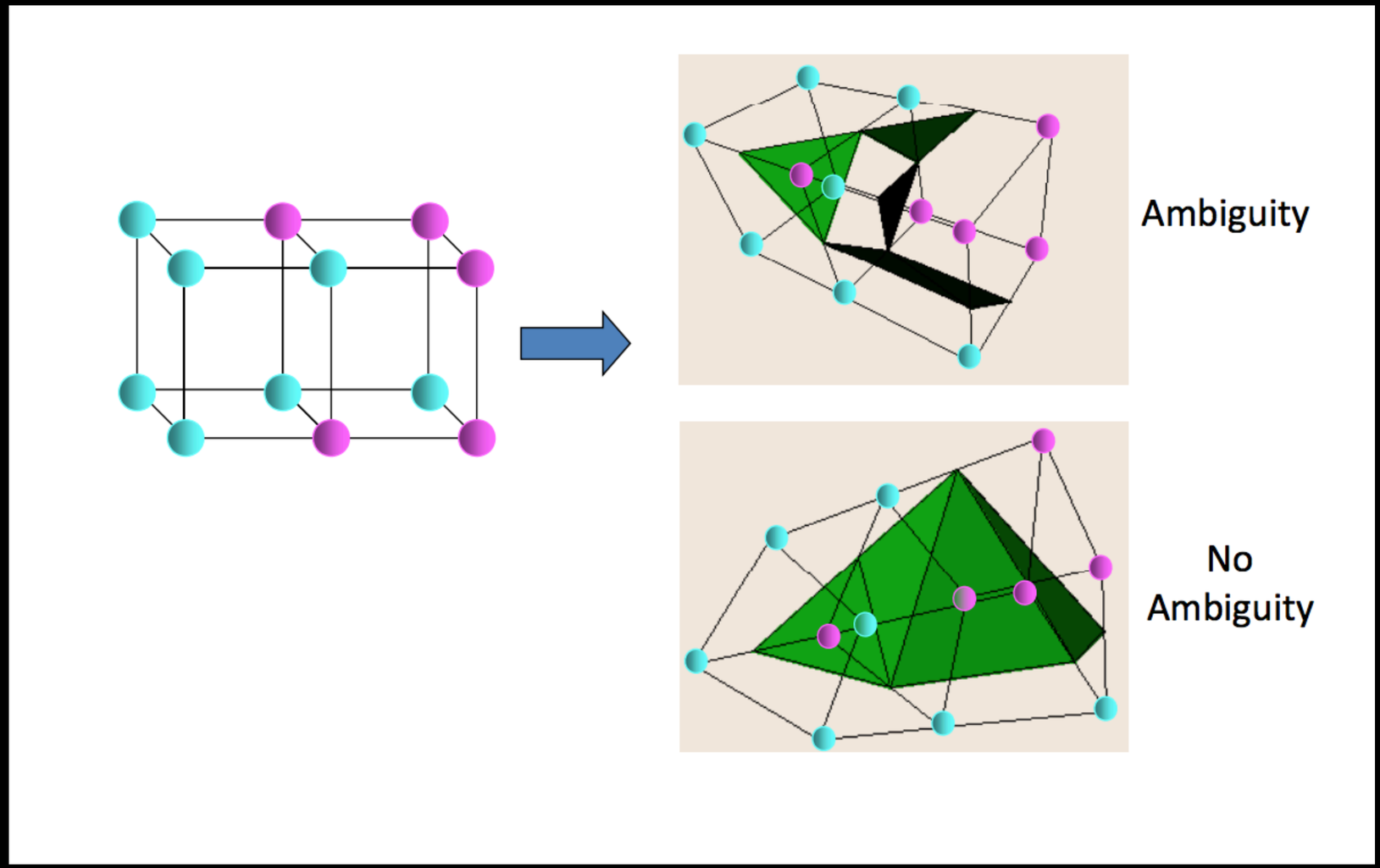
- 256 cases -> 23 cases
  - Rotation only
  - Always separate same “color”
  - Ambiguous faces triangulated consistently
- 8 new cases



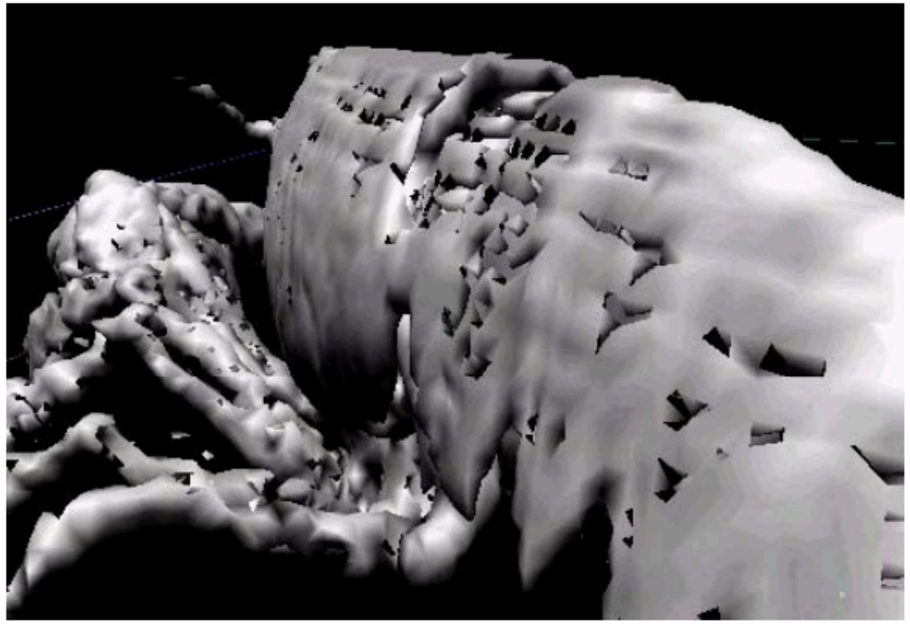
WITHOUT INVERSION



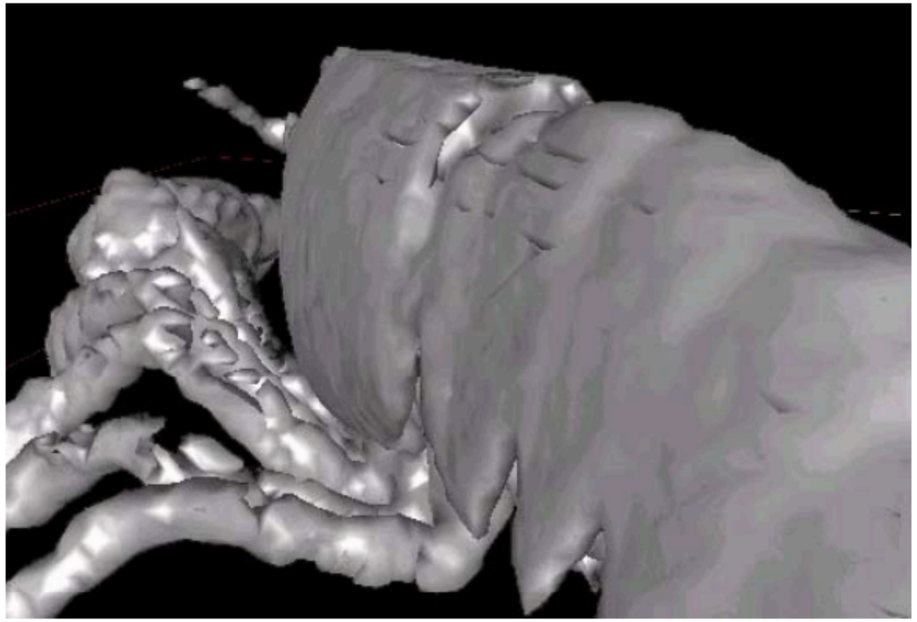
WITHOUT INVERSION



TEXT



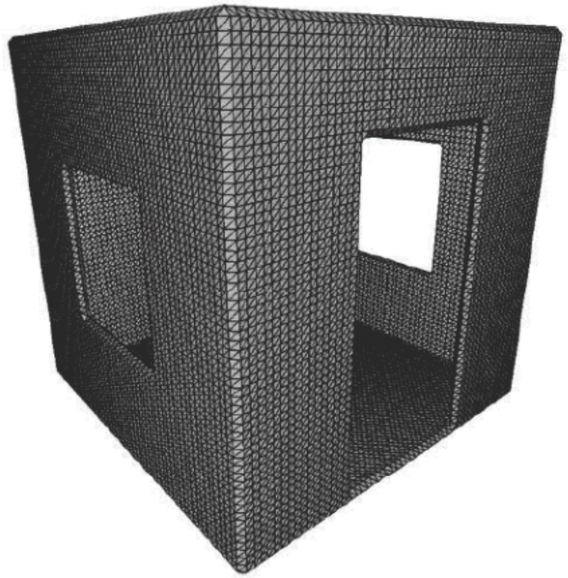
Ambiguity



No Ambiguity

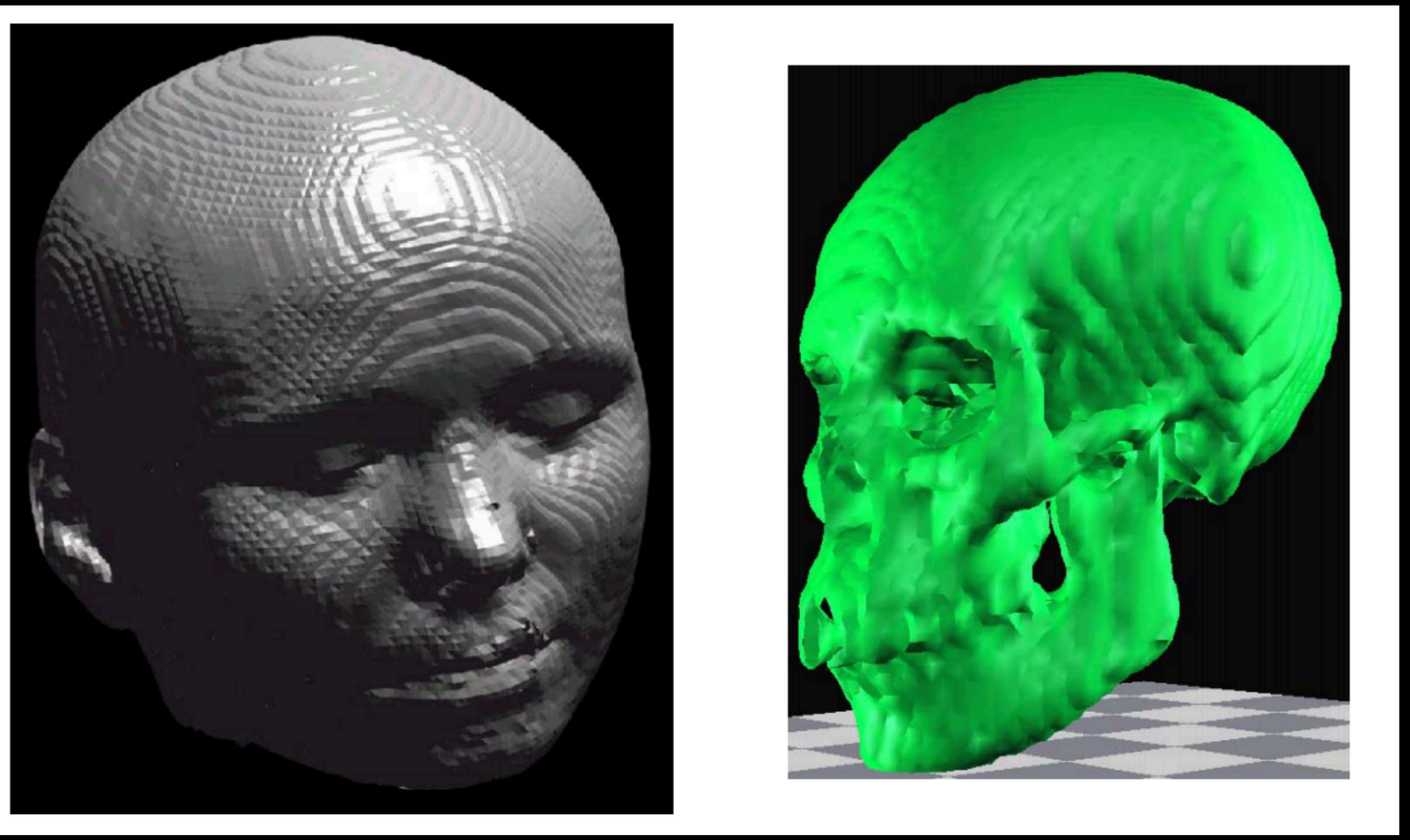
MARCHING CUBES ISSUES

- Grid not adaptive
- • Many polygons are used regardless of the geometric feature size.



Images from: “Dual Marching Cubes: Primal Contouring of Dual Grids” by Schaeffer et al.

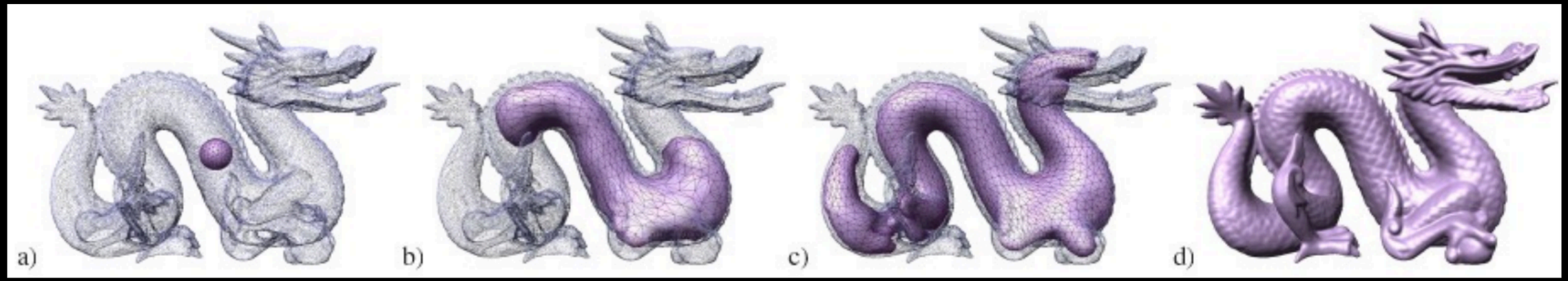
TEXT



DEFORMABLE MODELS

DEFORMABLE MODELS

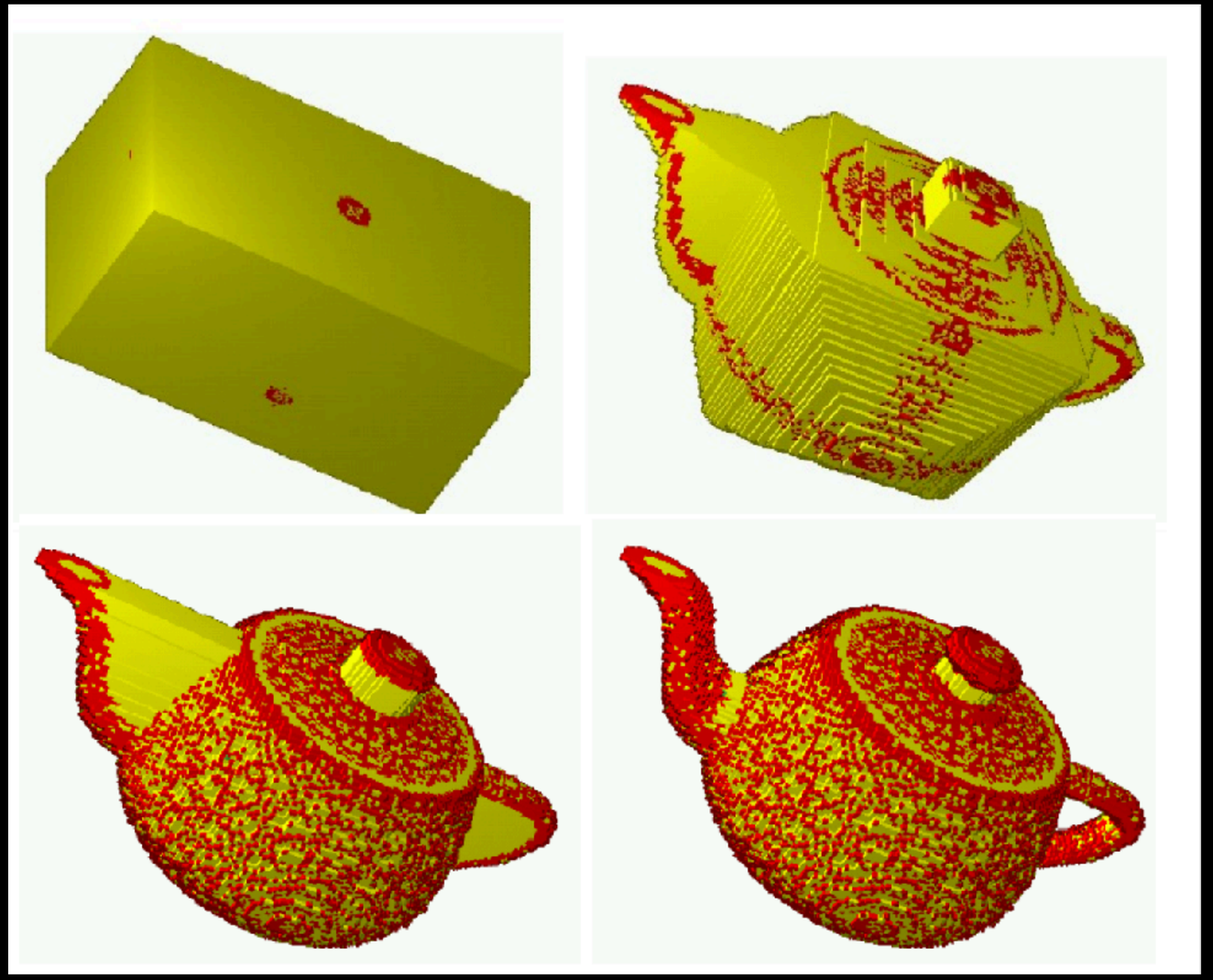
- Define a close surface that will deform to fit the input points
- Suppose the surface to be watertight
- Can be combined with previous approaches



ESTEVE ET AL. 2005

- Discretization into a regular grid
- Discrete membrane = close connected set of voxels
- At the beginning: boundary voxels of the grid
- Then shrunk until it contains input points
- Operations: contraction, undo contraction, freeze
- No use of normal information
- OK for non-uniform sampling

ESTEVE ET AL. 2005

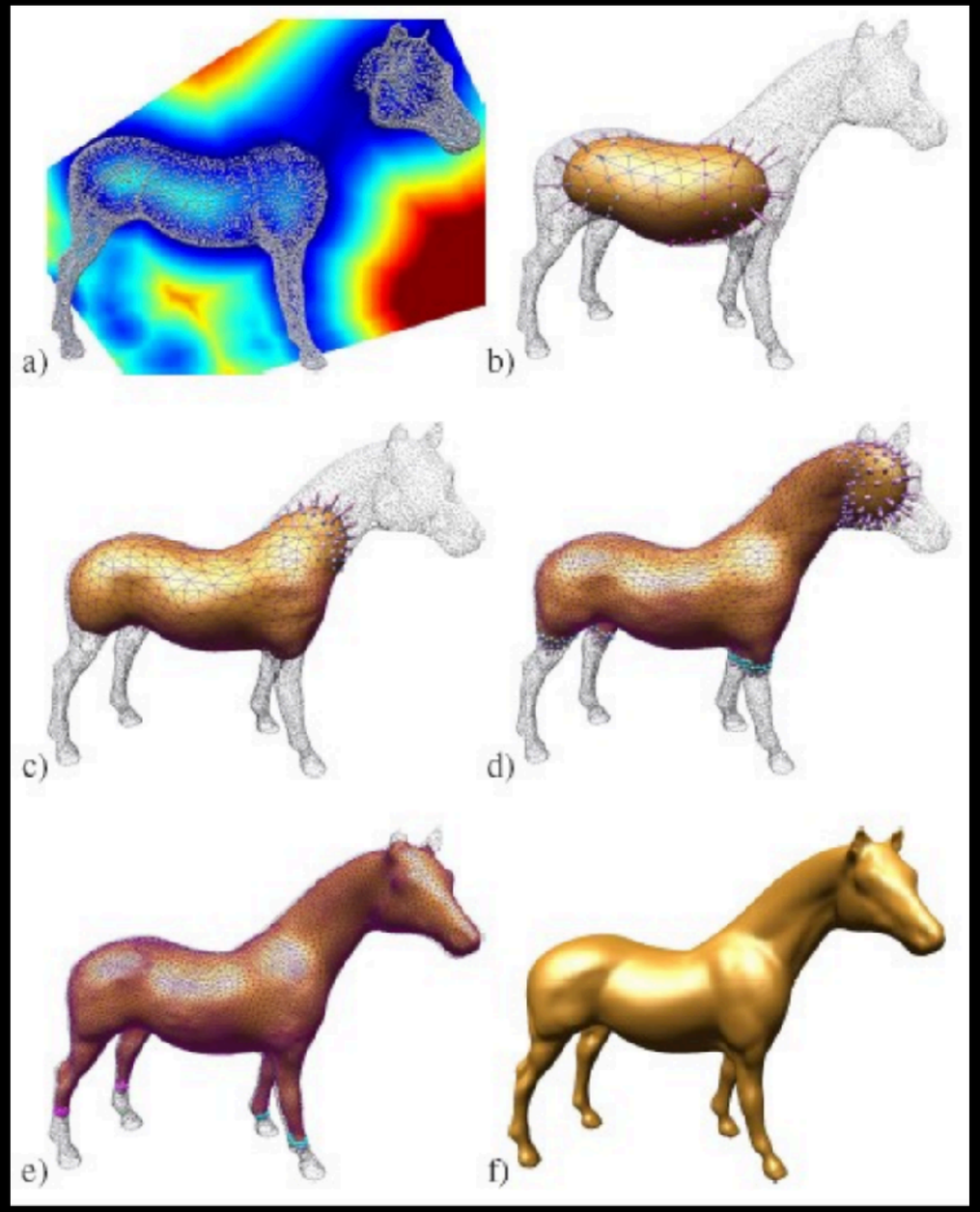


SHARF ET AL. 2006

- Start from a small sphere mesh inside the object
- Move its vertices in outward normal direction
- Using a volumetric distance map
- Adjust to local curvature and features (subdivision)
- Heuristics to handle topology changes

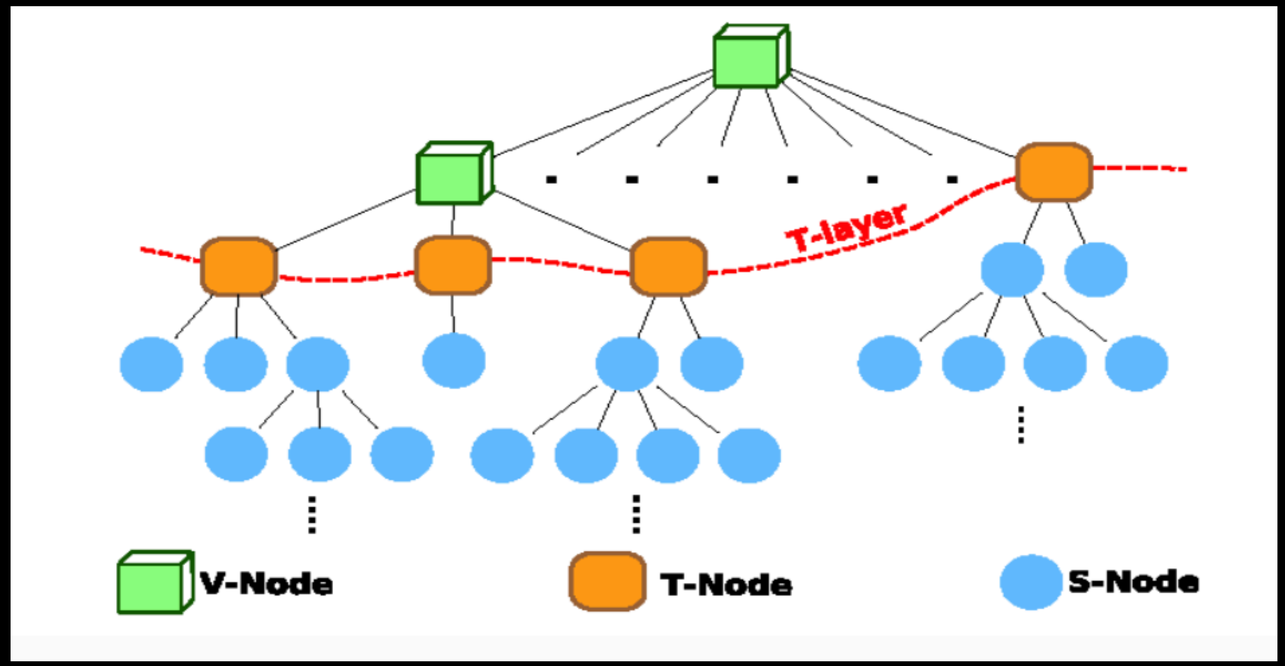
TEXT

SHARF ET AL. 2006



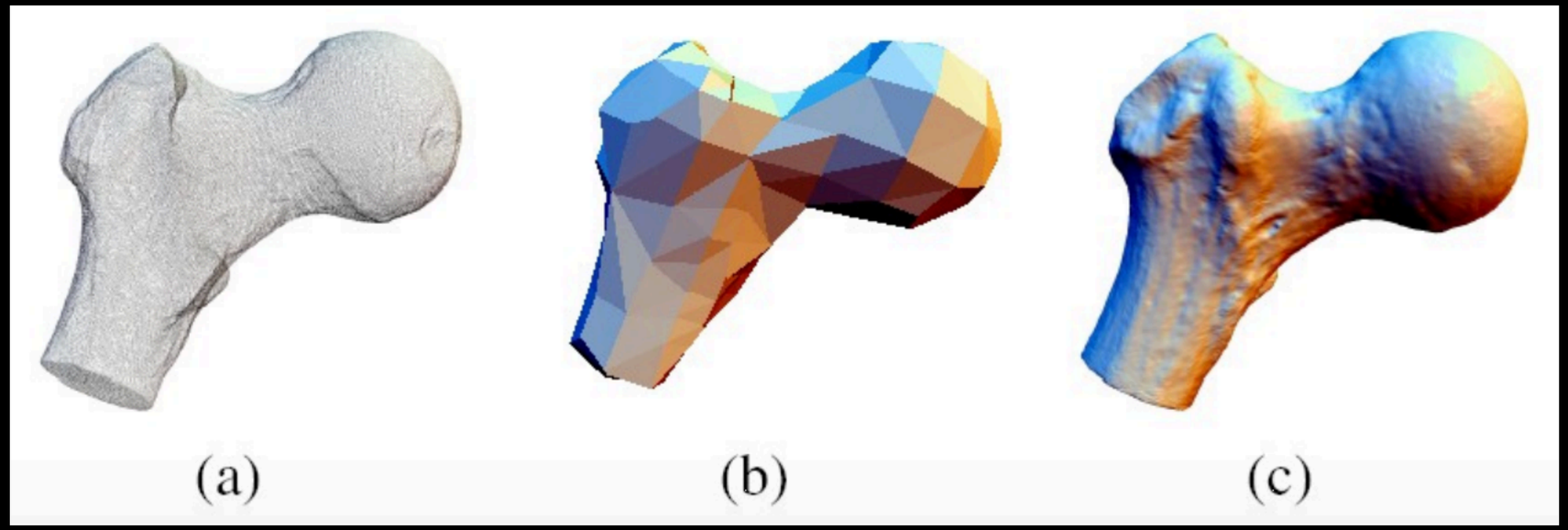
BOUBEKEUR ET AL. 2006

- Part of a more general paper presenting a new hierarchical space subdivision tool: VStrees
- ~ octree with surface leaves, forming a mesh



BOUBEKEUR ET AL. 2006

- Reconstruction process:
  1. VS-tree construction
  2. Coarse mesh constructed using the T-layer and MPU implicit reconstruction
  3. Several refinement tricks



SUMMARY

- Implicit and Explicit Meshing
  - Implicit is more adequate for us (noisy and irregular data)
- Need normals
  - Estimated with [Hoppe92], PCA on KNN
  - Check orientation with camera
- Moving Least Squares, Radial Basis Functions, Poisson.
- Marching Cubes to Extract the Surface from the Function

SOURCES

- https://www.cs.princeton.edu/courses/archive/fall12/cos526/lectures/11-reconstruction.pdf
- https://graphics.ethz.ch/publications/tutorials/points/powerpoint/Representation.Alexa.pdf
- http://www-evasion.imag.fr/people/Franck.Hetroy/Teaching/Geo3D/seance06-1.pdf
- http://www.hao-li.com/cs599-ss2015/slides/Lecture06.2.pdf
- http://www.cs.uu.nl/docs/vakken/ddm/2016-2017/Lecture%203%20-%20Reconstruction.pdf
- http://www.cs.technion.ac.il/~cs236329/lectures/poisson_reconstruction.pdf
- http://www.cs.uu.nl/docs/vakken/ga/slides9alt.pdf
- http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/04_Surface_Reconstruction.pdf

RESOURCES

- http://hhoppe.com/poissonrecon.pdf
- http://www.cs.jhu.edu/~misha/Code/PoissonRecon/Version9.01/
- http://hhoppe.com/proj/recon/
- http://www-lb.cs.umd.edu/class/spring2005/cmsc828v/papers/siggraph01.pdf