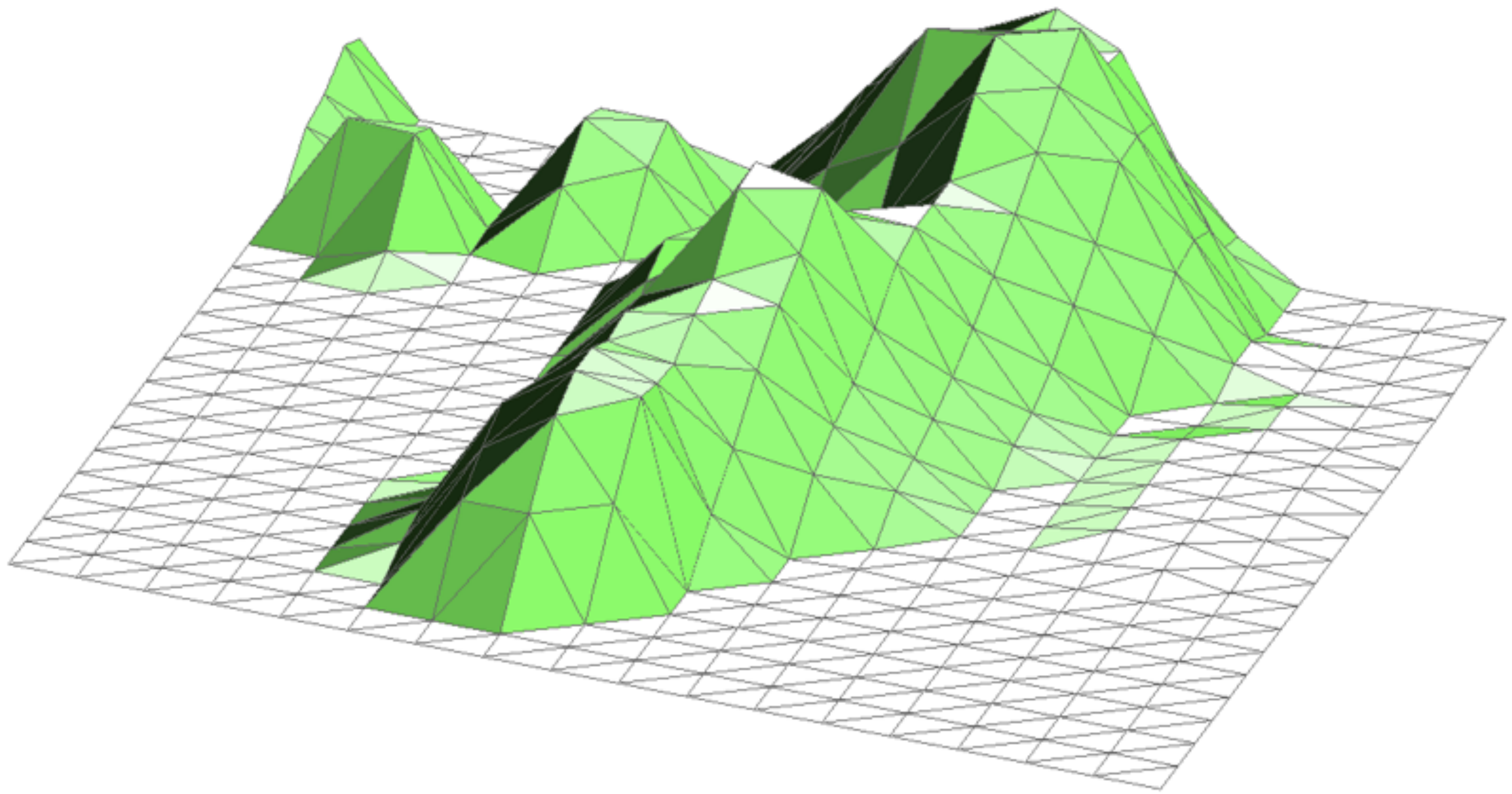
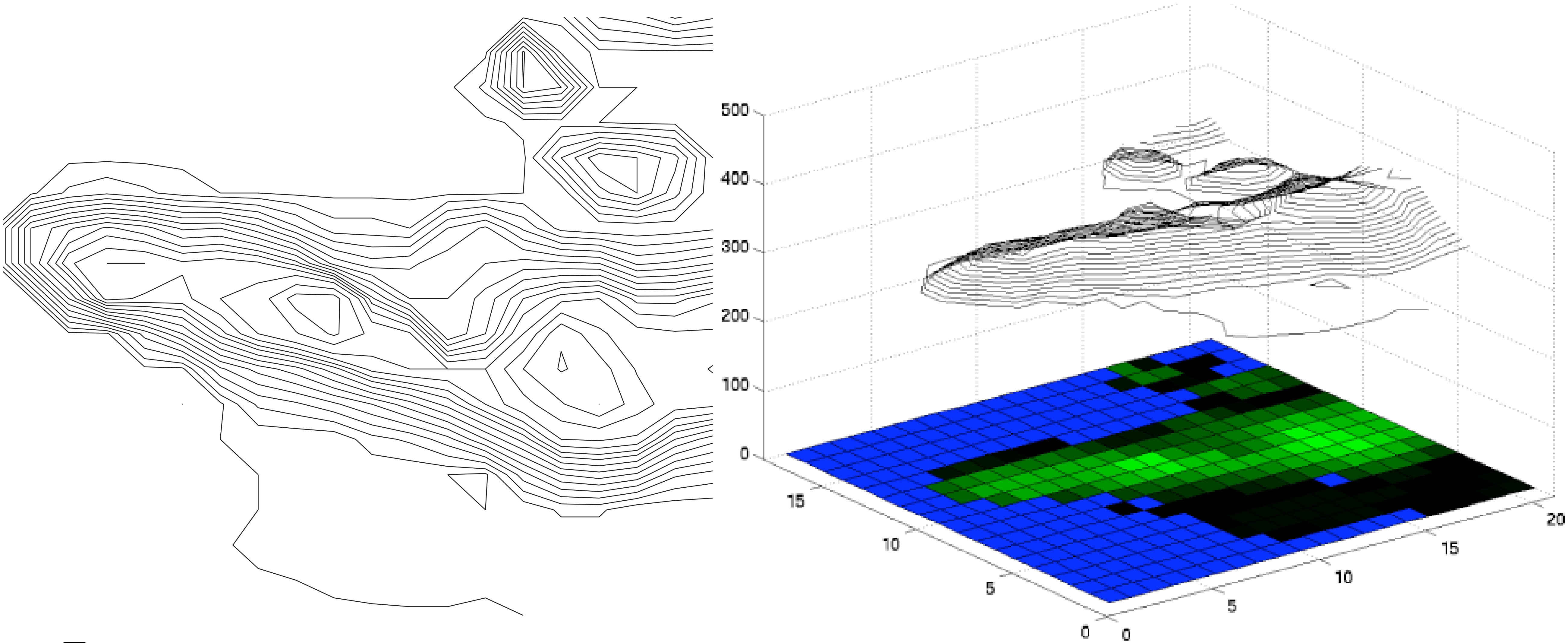
19: Isosurfaces & Distance Fields

$f: \mathbb{R}^2 \rightarrow \mathbb{R}$: Height Field



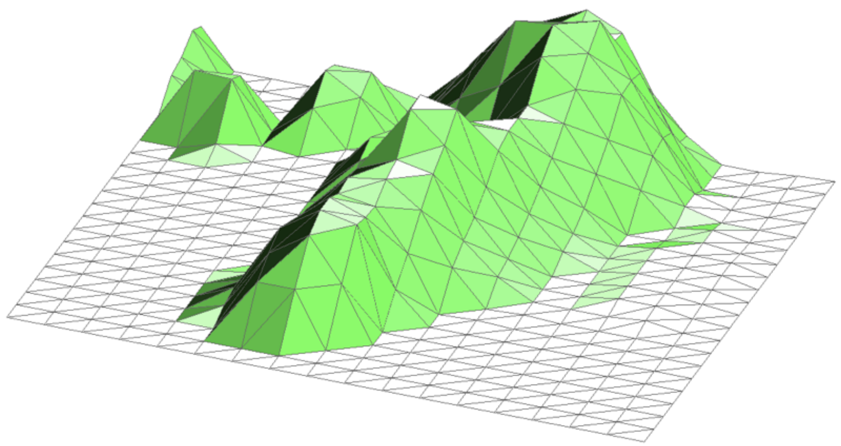
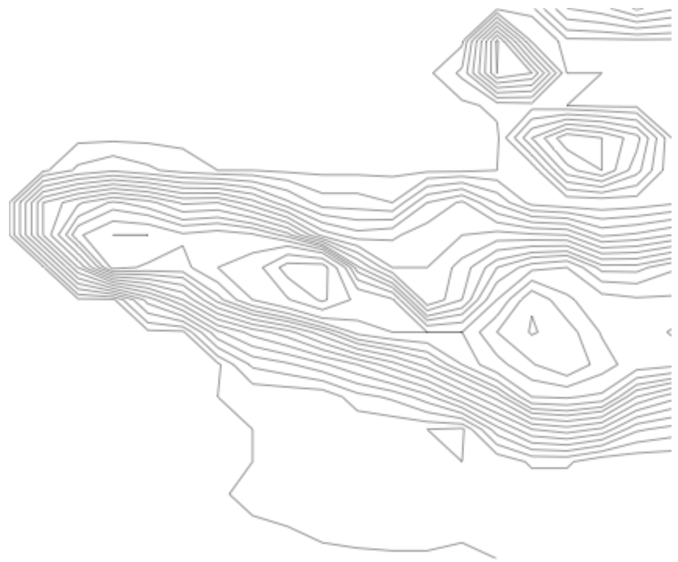
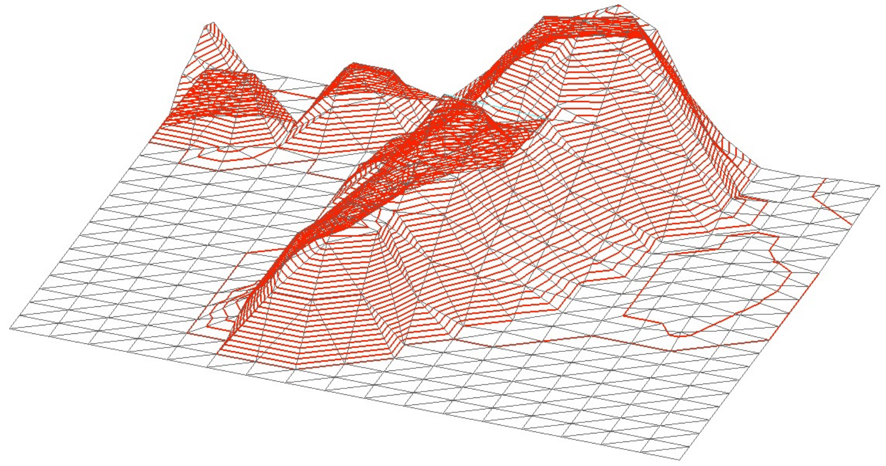
- For every point in the plane, define a *height*
- The graph is then a *terrain*
  - i.e. a surface in a 3D embedding space

Contours



- We can define curves using contours
- Let an artist design the landscape
- Pick a threshold, generate a curve

Manifold Dimension

- f is a 2-manifold embedded in 3-D
- It's contours are 1-manifolds
  - embedded in 2-D
  - or in 3-D
- What happens if we add a dimension?

Implicit (Blobby) Surfaces

- Define a function $f: \mathbb{R}^3 \rightarrow \mathbb{R}$
  - Often a sum of Gaussian distributions
  - It's a 3-manifold embedded in 4-D
  - It's contours are 2-manifolds in 3-/4-D
- Choose a threshold h
- Extract the level set / contours
  - They are *guaranteed* to be manifold

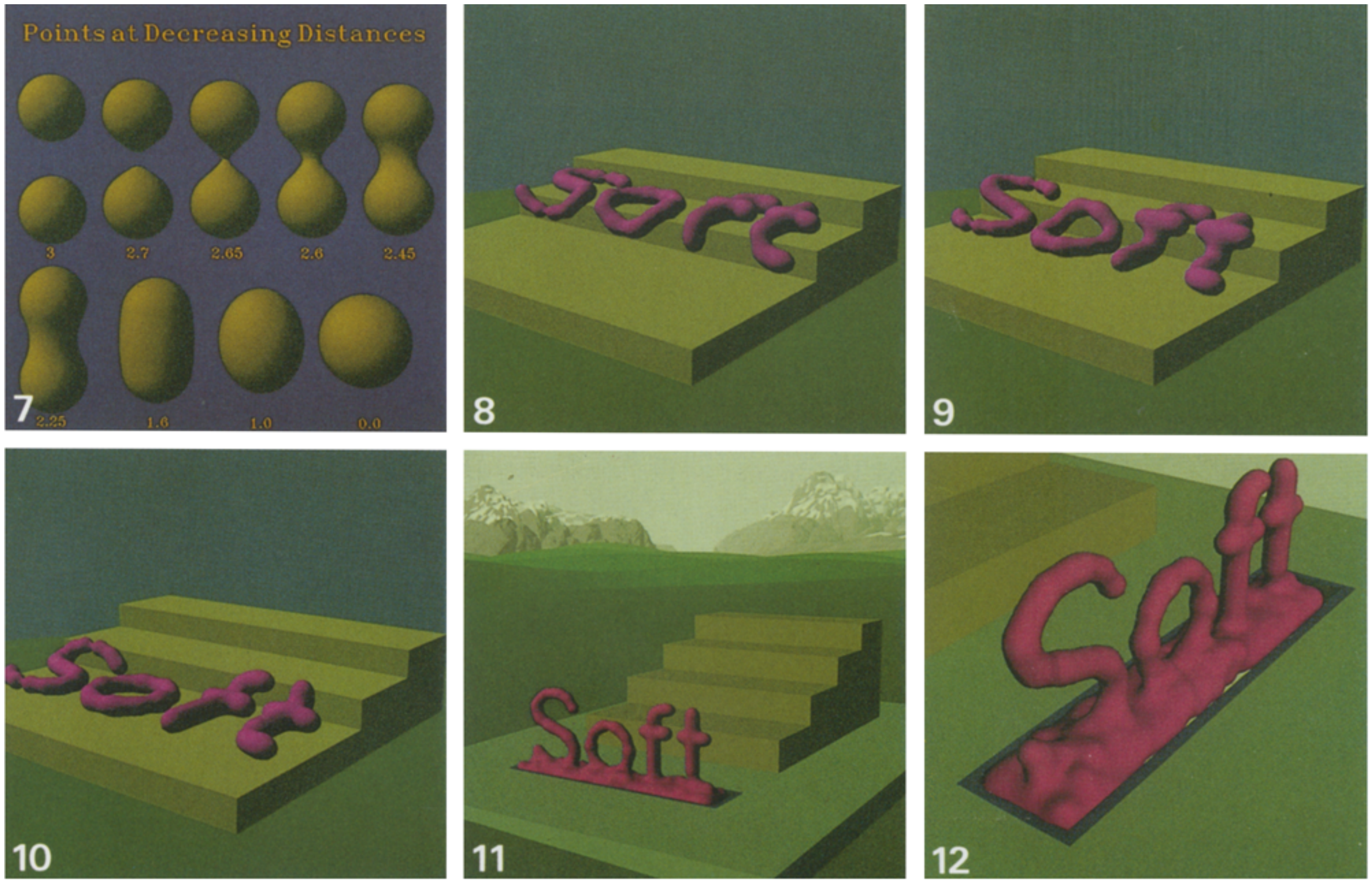
Isosurfaces

- Contours in 3D
- Based on the *inverse image* of and *isovalue*

$$f^{-1}(h) = \{(x, y, z) : f(x, y, z) = h\}$$

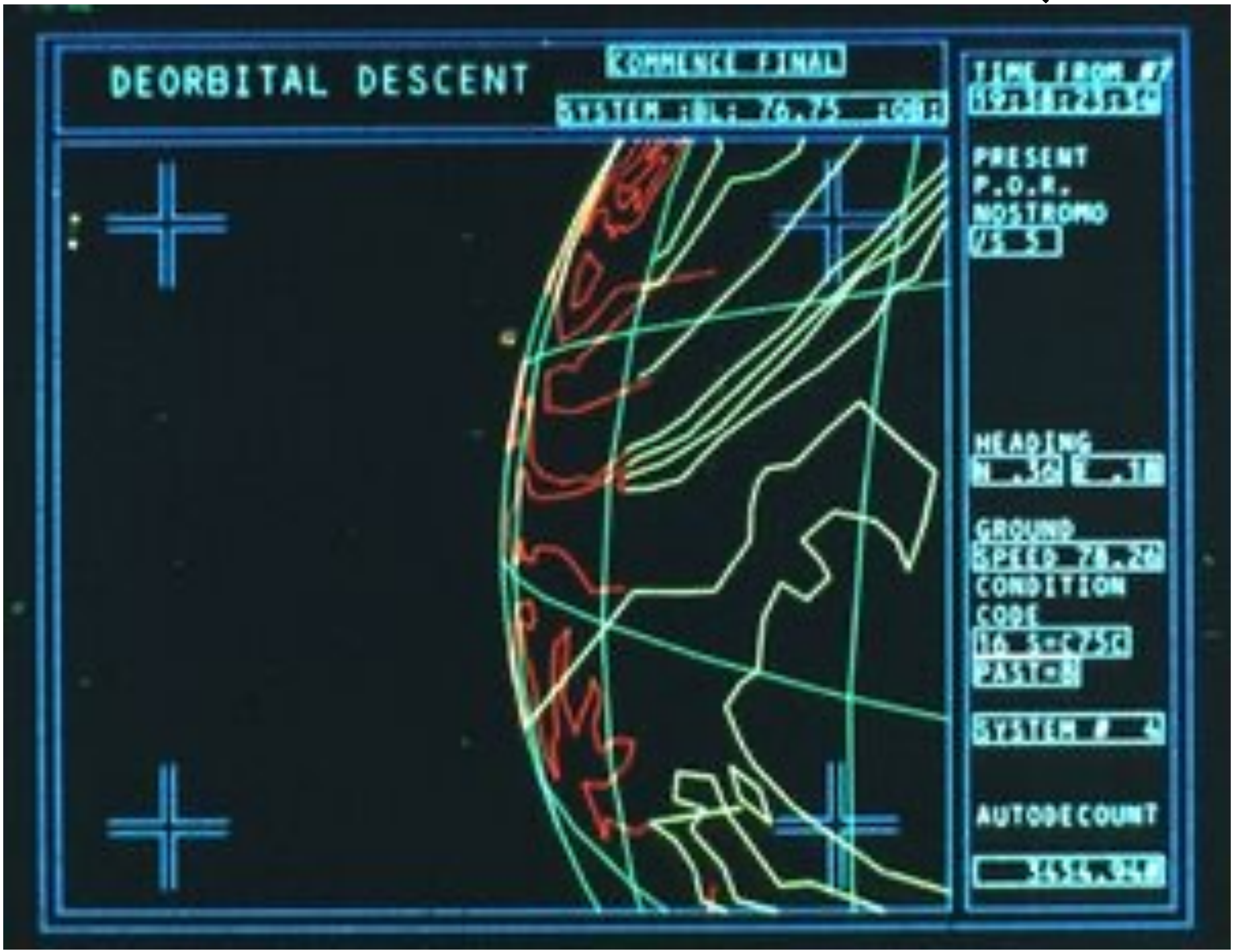
- 2-manifolds embedded in 3 or 4 space
- We can usually only show one isosurface
  - because they occlude each other

Implicit (Blobby) Surfaces



- Wyvill, McPheeters & Wyvill, 1986

Bonus Points: Alien (1979)



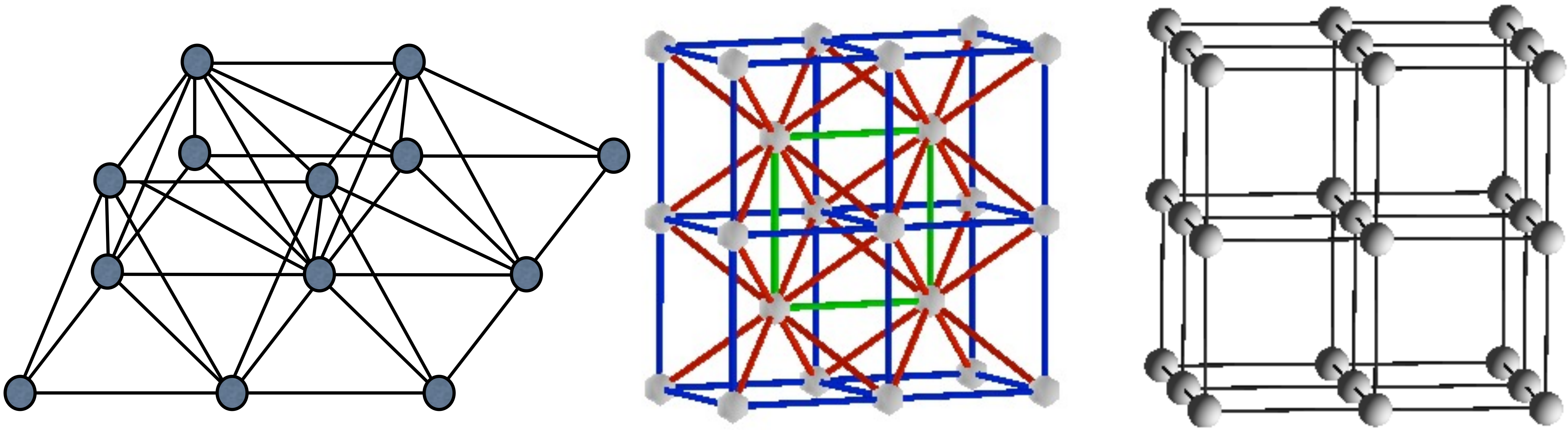
Implicit Surfaces

- We can use many types of functions:
  - CT/MRI scans
  - Numerical simulations
  - Mathematical models – eg. sum of Gaussians
  - Distance fields
  - *any* property we can compute in 3-D
- In practice, we need to start with a 3-D mesh

Meshes in 3D

- In 2-D, we use triangles and squares
- In 3-D, we use tetrahedra and cubes
  - also pyramids, octahedra & triangular prisms
- Interpolation follows the same rules as in 2-D
  - geometric interpolation (meshes)
  - kernel filters (sampling theory)

Mesh Construction



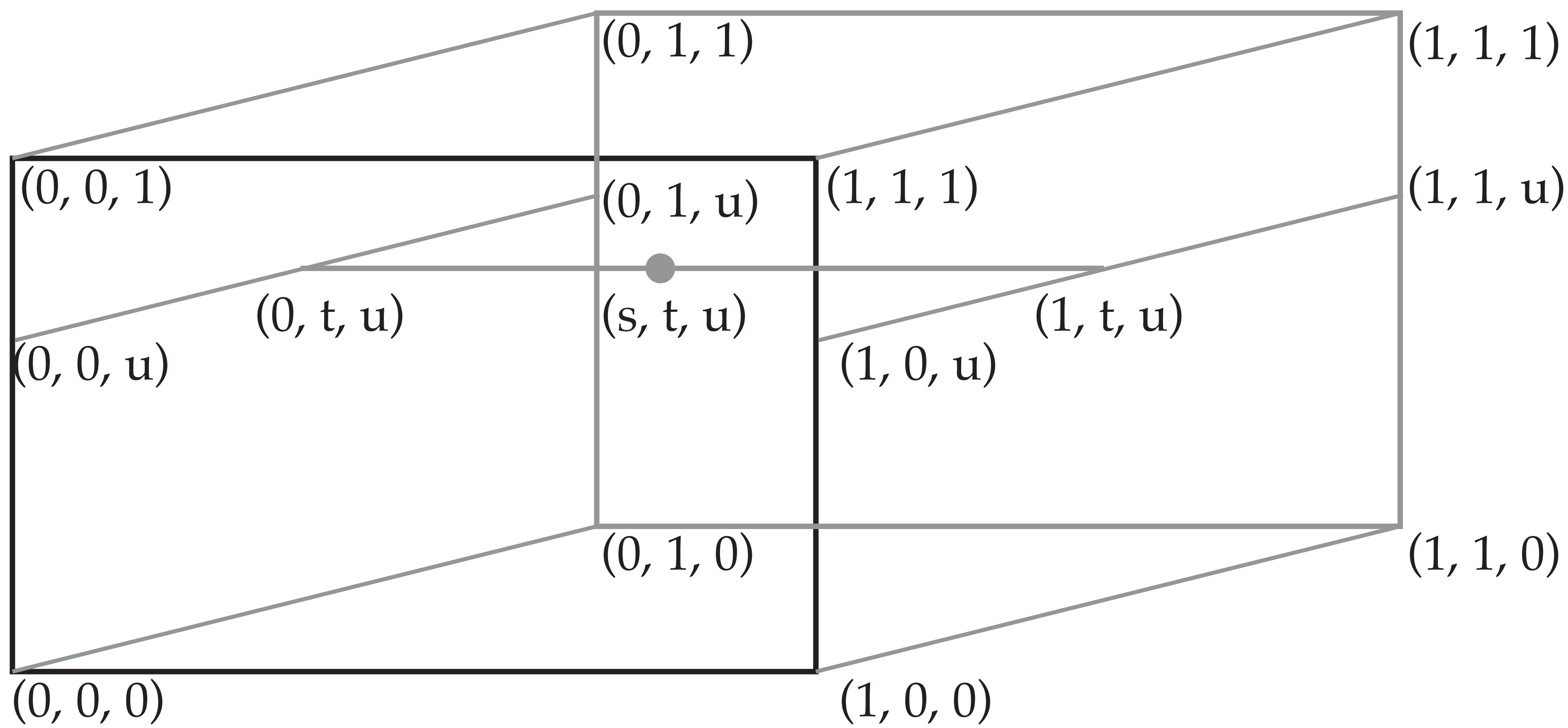
- Two common types of mesh:
  - Tetrahedral, either arbitrary or subdivided
  - Cubic – basically, a 3D array in memory

Barycentric Interpolation

- Linear interpolation based on distances
- Extend by using distance to faces
- Applied in each tetrahedron

$$f(p)=\alpha f(A)+\beta f(B)+\gamma f(C)+\delta f(D)$$
$$\alpha+\beta+\gamma+\delta=1$$
$$\alpha=d(p,b,c,d)=\begin{vmatrix}1 & p_x & p_y & p_z\\ 1 & b_x & b_y & b_z\\ 1 & c_x & c_y & c_z\\ 1 & d_x & d_y & d_z\end{vmatrix}$$

Trilinear Interpolation



- Repeat linear interpolation in 3 dimensions
- Used for cubic meshes

Trilinear Interpolation

$$f(x,y,z) = a\;xyz + b\;yz + c\;xz + d\;xy + ex + fy + gz + h$$

$$a = b_{111} - b_{110} - b_{101} - b_{011} + b_{100} + b_{010} + b_{001} + b_{000}$$

$$b = b_{011} - b_{010} - b_{001} + b_{000}$$

$$c = b_{101} - b_{100} - b_{001} + b_{000}$$

$$d = b_{110} - b_{100} - b_{010} + b_{000}$$

$$e = b_{100} - b_{000}$$

$$f = b_{010} - b_{000}$$

$$g = b_{001} - b_{000}$$

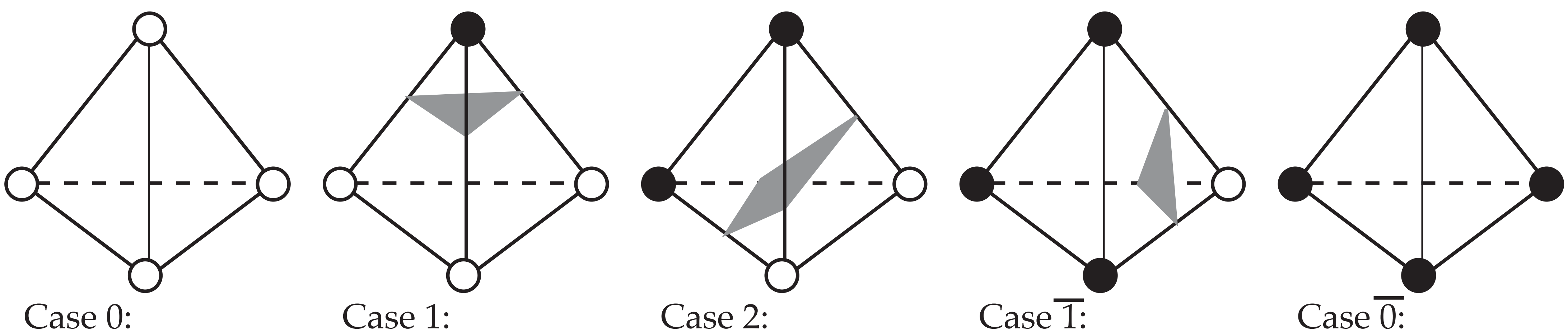
$$h = b_{000}$$

Marching Cells

- A class of algorithms that extract contours
- Given a mesh made of cells
  - triangles, squares, tetrahedra, cubes, &c.
- Iterate (march) through each cell
  - Extract the part of the mesh in the cell
- Produces triangle soup
- Sort out connectivity, &c. afterwards

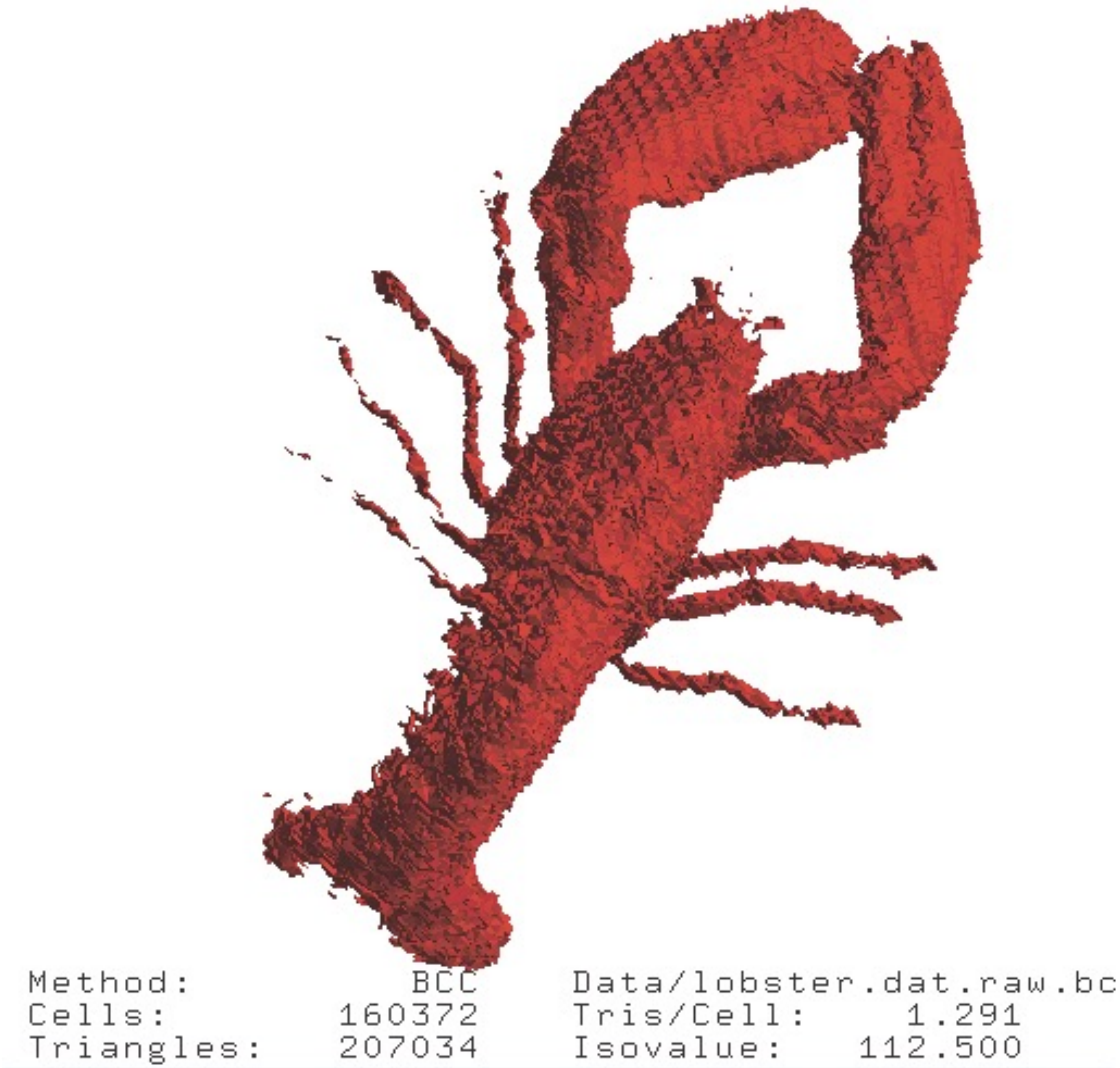
Marching Tetrahedra

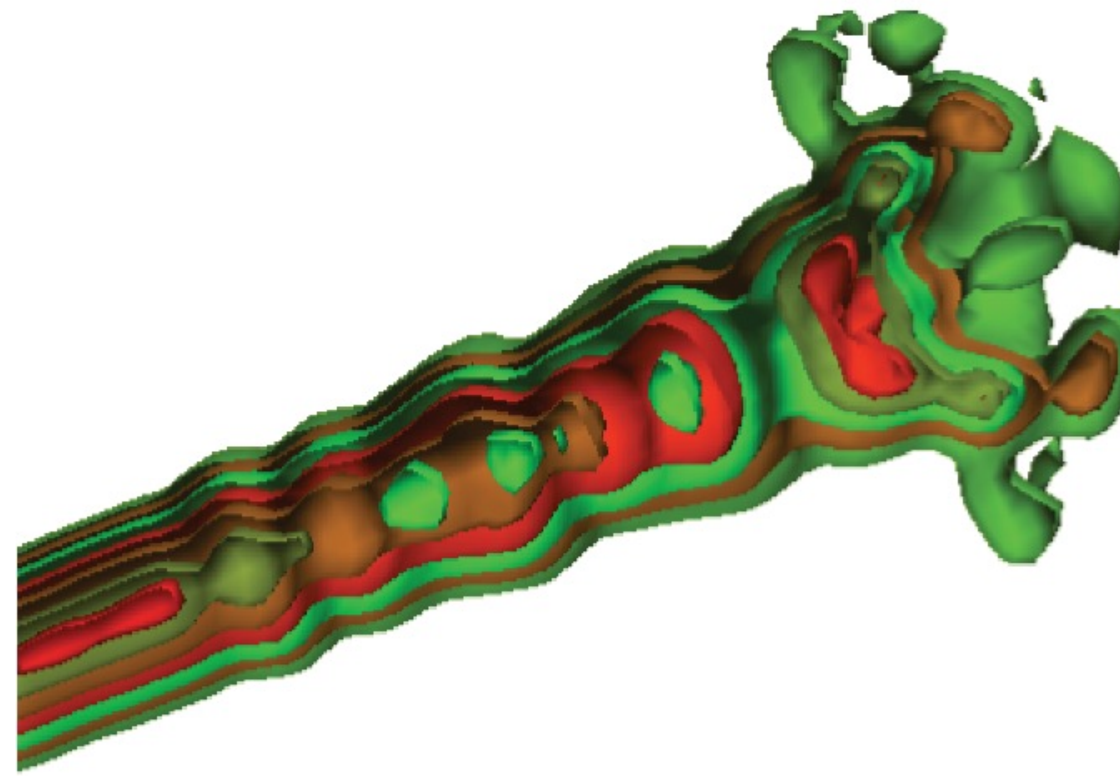
- Classify vertices black / white
- Interpolate points along edges
- Connect with simplices - in 3-D, triangles



- Any surface generated is flat (even quad in 2)

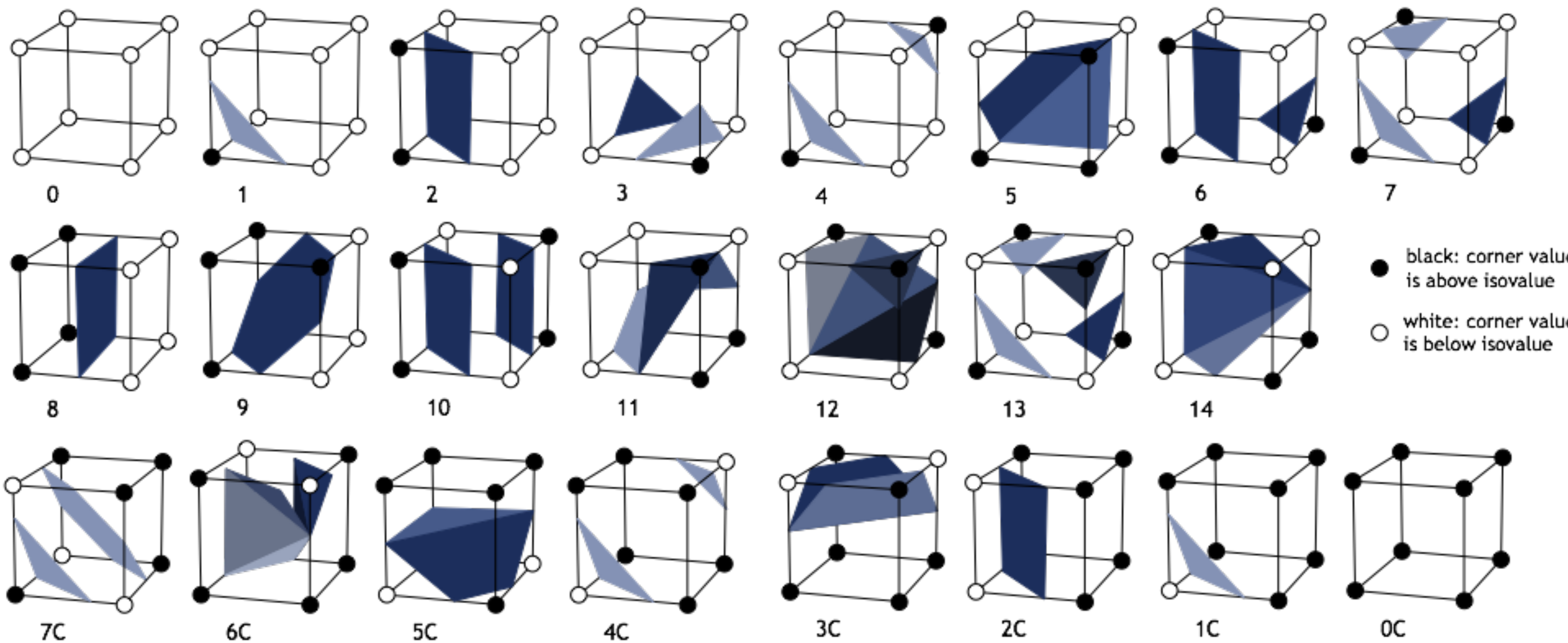
Result





UNIVERSITY OF LEEDS

Marching Cubes



An Example

Trilinear Tessellation

- Lorenson & Cline, 1987 (Marching Cubes)
- Wyvill, McPheeters & Wyvill, 1987 (Soft Objects)
- Dürst, 1988 (cracks in Marching Cubes)
- Wilhelms & van Gelder, 1990, (simplicial subdivision)
- Matveyev, 1994, Montani, Scateni & Scopigno, 1994 (modified MC tables)
- Nielson & Hamann, 1991 (Asymptotic Decider - bilinear saddles on faces)
- Natarajan, 1994 (Extension of Asymptotic Decider - trilinear body saddle)
- Cignoni, Ganovelli, Montani, & Scopigno, 2000 (Natarajan's solution)
- Pascucci (& others?), 2002 (also 2 body saddles)
- Nielson, 2003 (full set of cases)
- Lopes & Brodlie, 2003 (optimal placement of vertices)
- Carr (& Max) 2007/8 (alternate proof)

Computing Normal

- Normal vector is based on gradient
  - always perpendicular to contours
  - describes steepest ascent
  - but outside is typically low-valued
  - use *negative* gradient as normal

$$\begin{aligned}\vec{n} &= -\nabla f(x) \\ &= \left(-\frac{\delta f}{\delta x}, -\frac{\delta f}{\delta y}, -\frac{\delta f}{\delta z}\right)\end{aligned}$$

Central Differencing:

- Approximation of gradient:

$$\vec{n}_{i,j,k} \approx \left(-\frac{f(x_{i+1}, y_j, z_k) - f(x_{i-1}, y_j, z_k)}{2}, -\frac{f(x_i, y_{j+1}, z_k) - f(x_i, y_{j-1}, z_k)}{2}, -\frac{f(x_i, y_j, z_{k+1}) - f(x_i, y_j, z_{k-1})}{\delta z}\right.$$

- Compute at vertices of cell
- Interpolate along edges

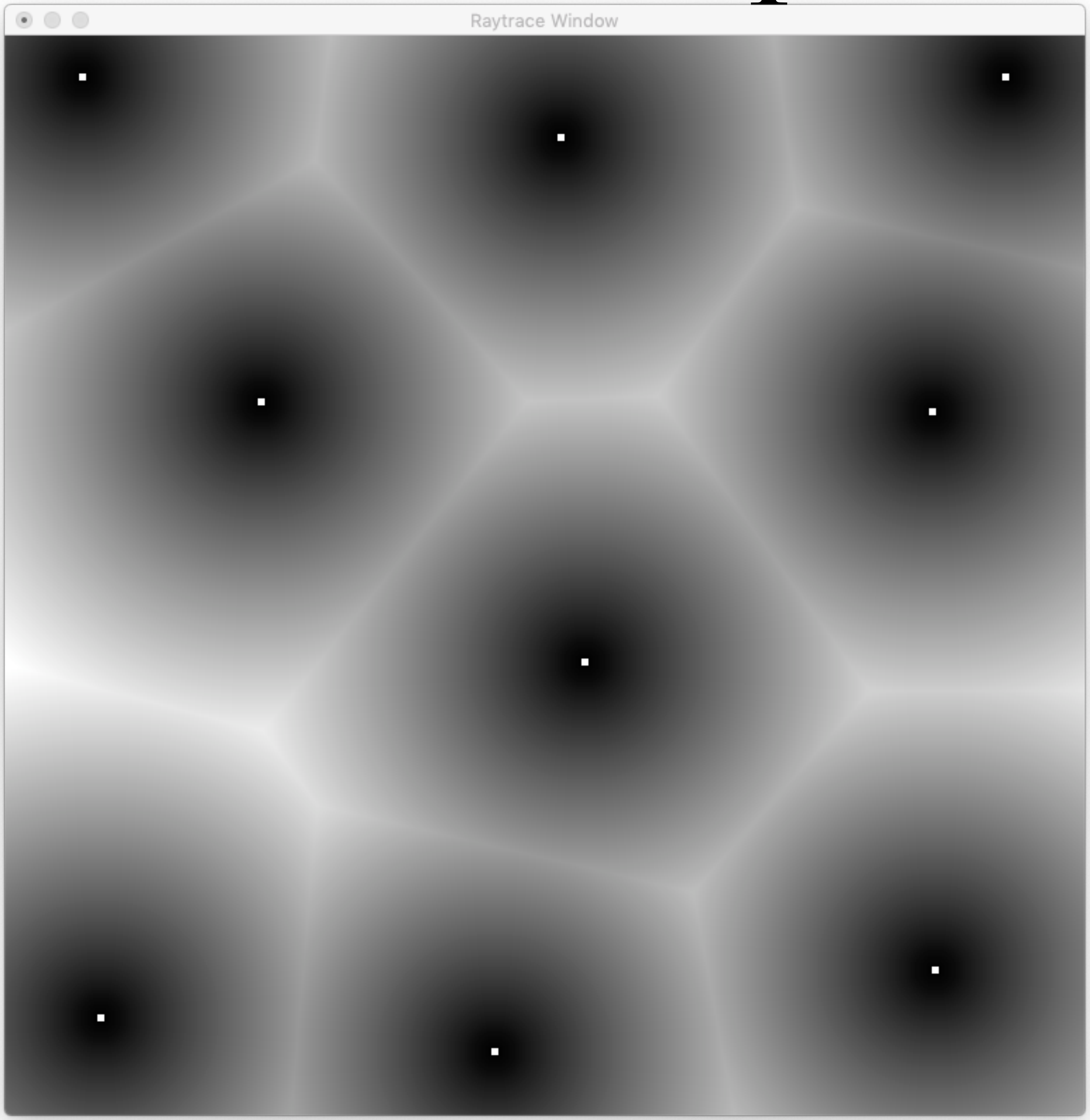
Distance Fields

- Given a set of points $\mathrm{P} = \{p \in \mathbb{R}^3\}$
- Define a distance function
  - $d(x, y) = \sqrt{(y - x)^2}$
- Use this to define a distance field:
  - $\delta(x) = \min_{p \in \mathrm{P}}(\mathrm{d}(\mathrm{x}, \mathrm{p}))$
- Which measures the distance from x to P
  - The closest distance)

Possibilities

- Explicit list of points
- Generic set (infinitely many)
  - Set of lines
  - Set of triangles
  - Arbitrary surface

An Example



Volumetric Mesh Repair

- Generate a distance field δ in 3D
  - From the primitives (triangles), &c.
- Choose a small distance value d
- Take the isosurface for value d
- Guaranteed manifold
  - But far too many triangles
  - And slow