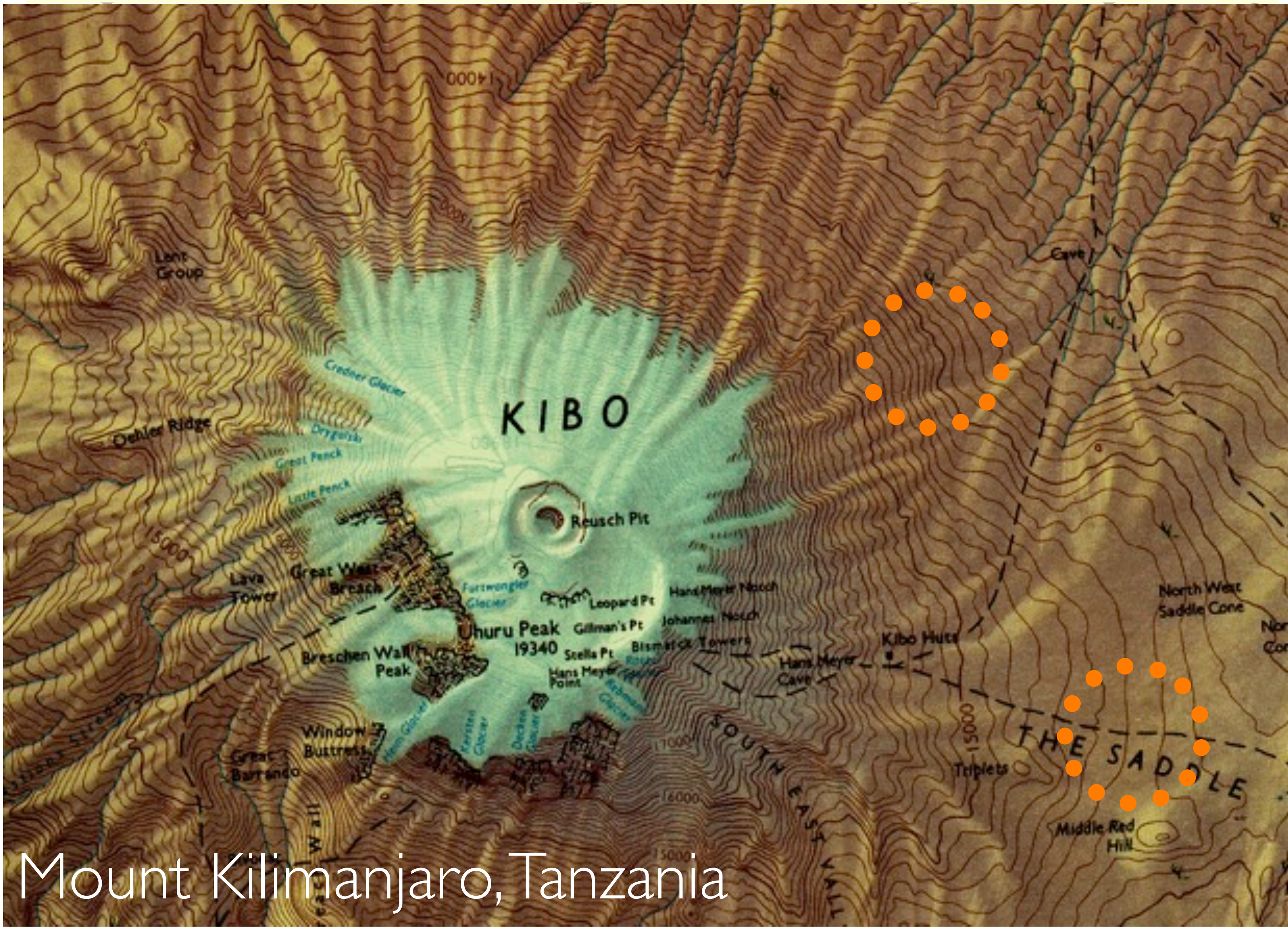
CS 530 - Visualization

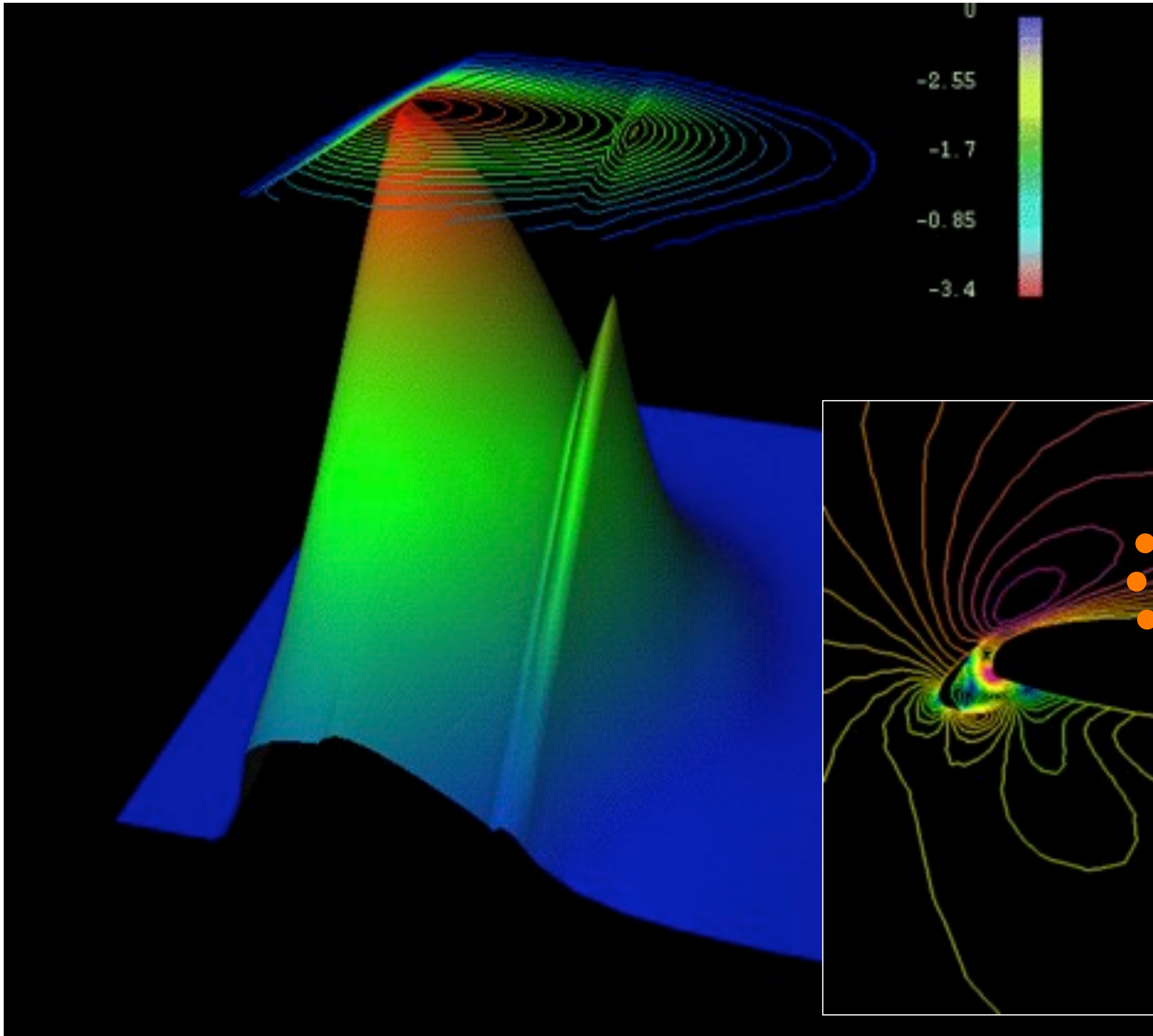
Isosurfacing

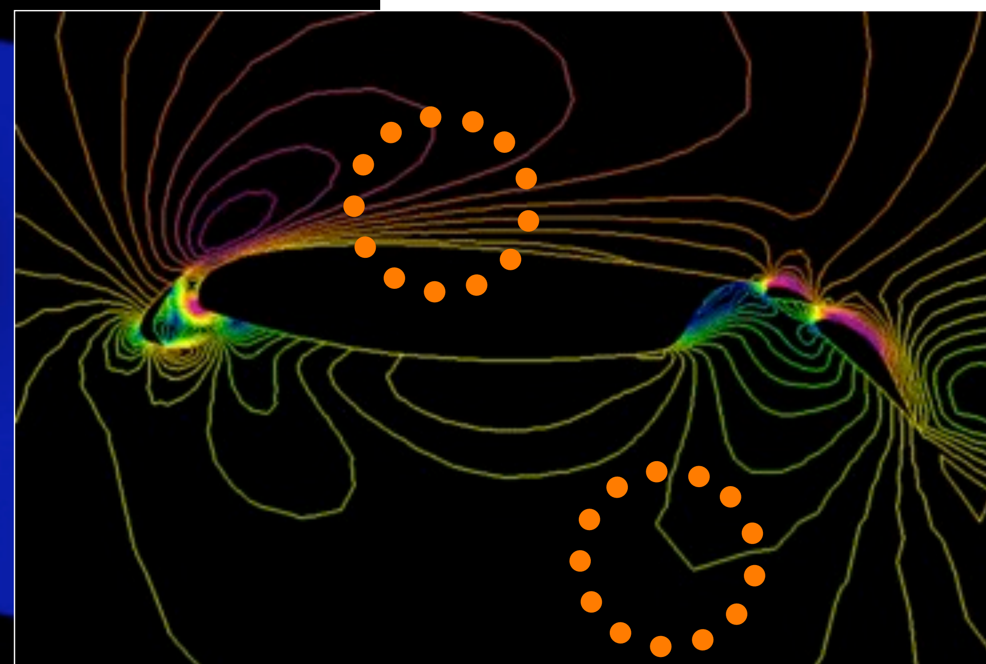
February, 6 2013

http://www.lib.berkeley.edu/EART/digital/topo.html

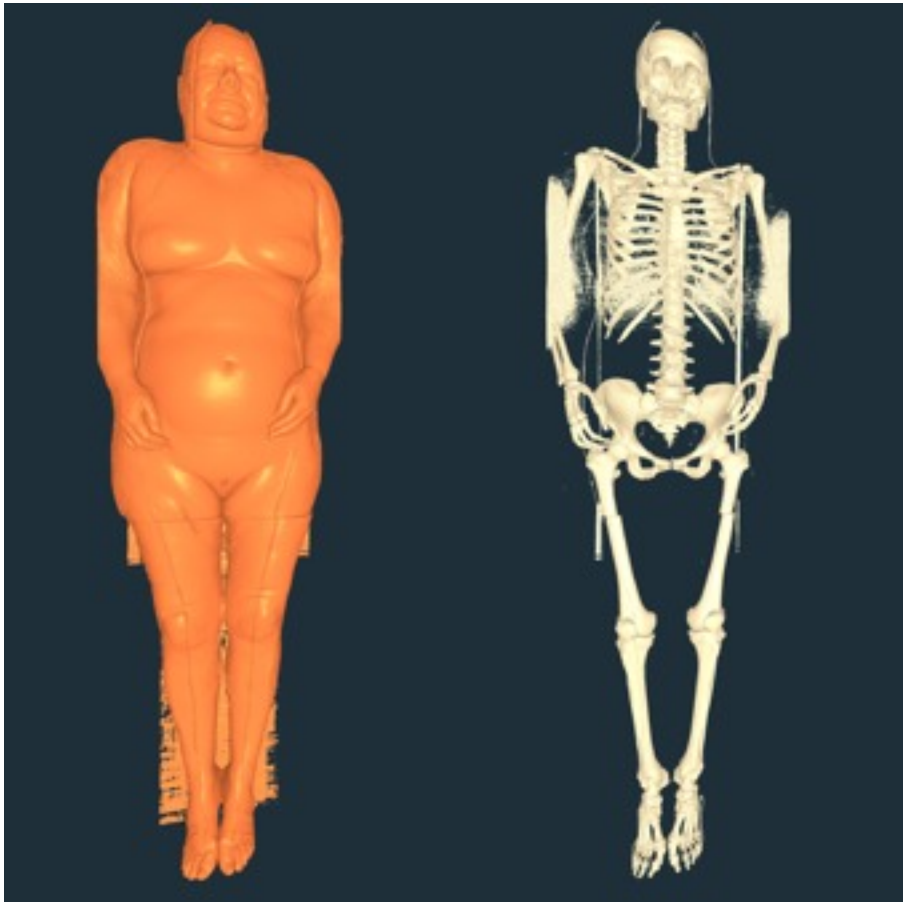


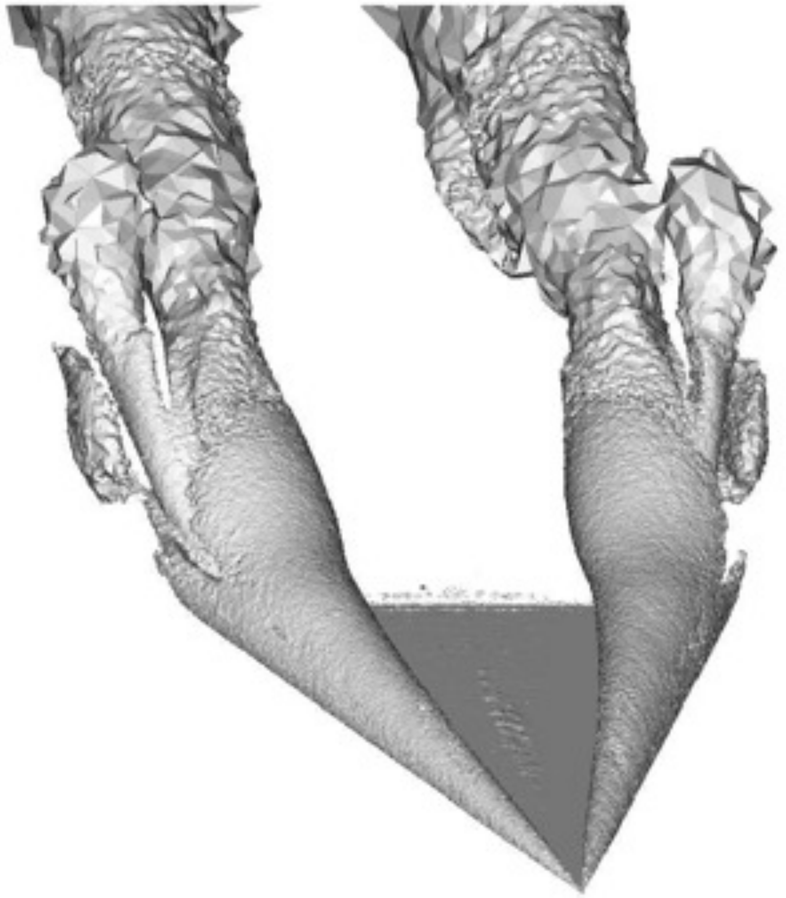
Other examples



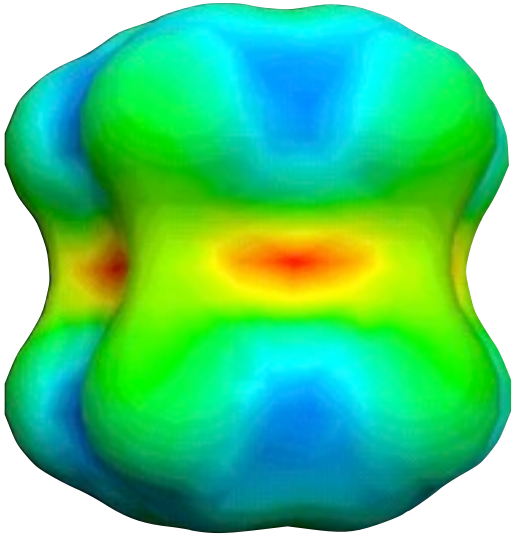


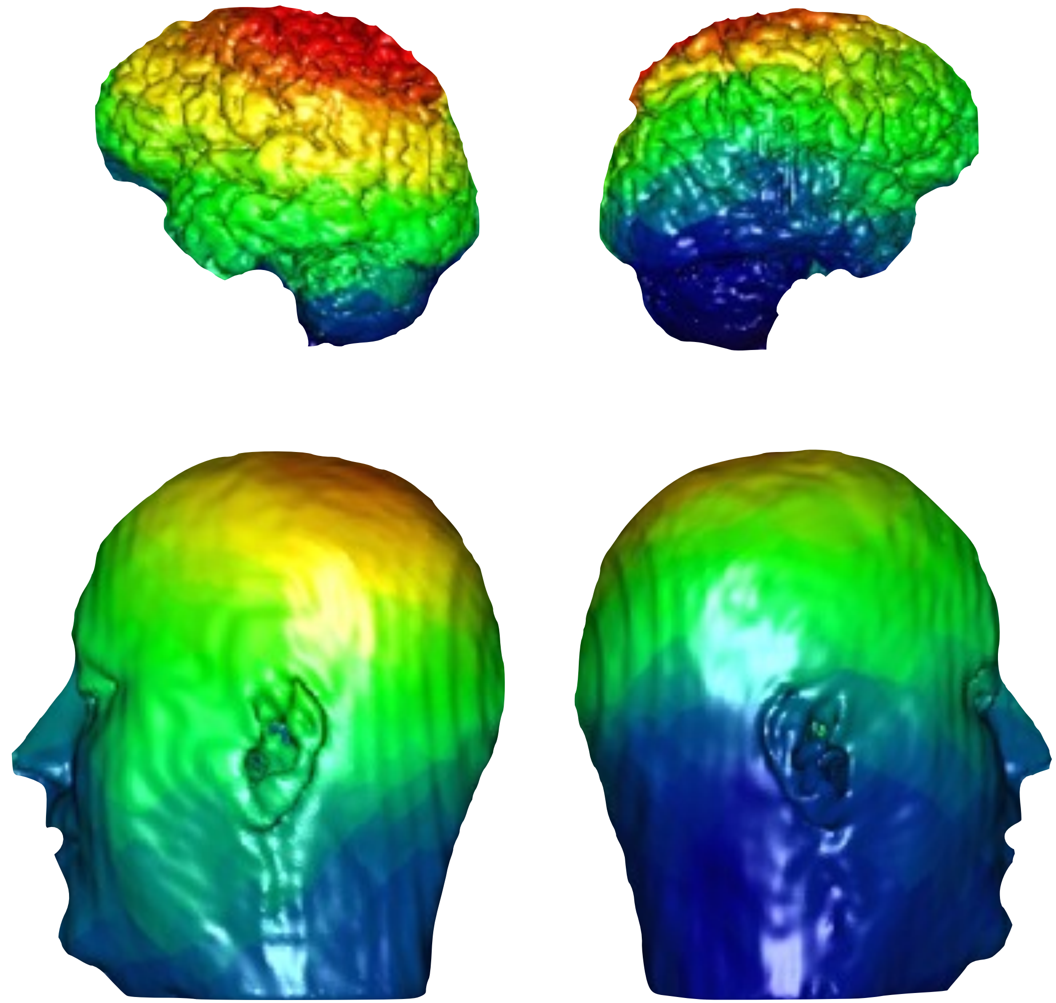
More examples

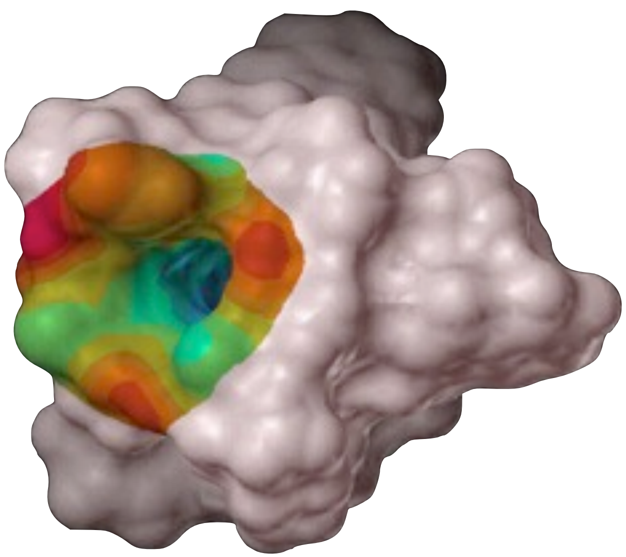




Colored Isosurfaces

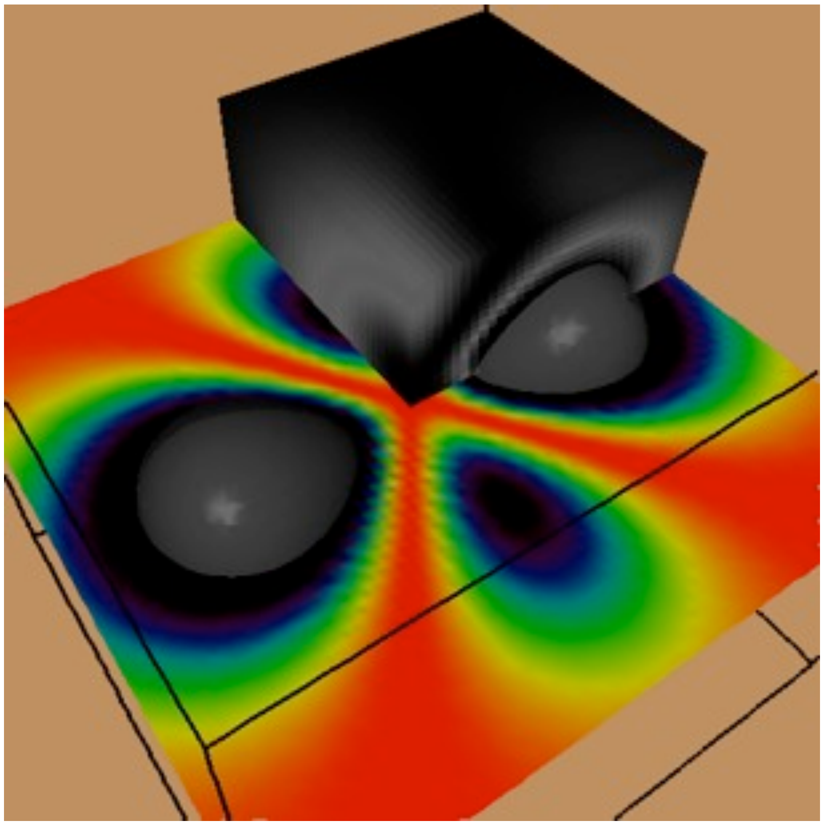


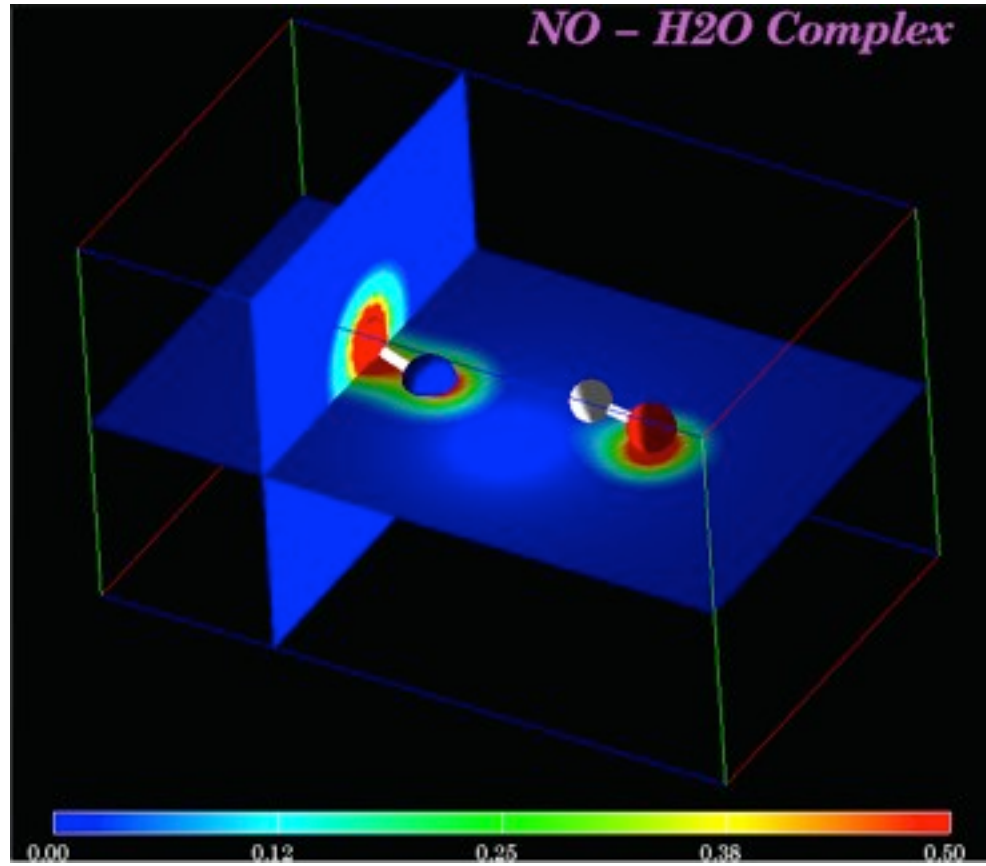




David Weinstein

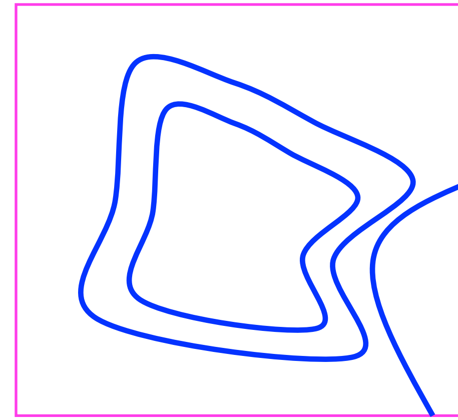
Slices still have their place





Colormapped slices

Properties of Isocontours



- Preimage of scalar value
    - Concept generalizes to any dimension
    - Manifolds of codimension 1
- Closed (except at boundaries)
- Nested–different values don't cross
    - Can consider the zero-set case (generalizes)
    - $F(x, y) = k \Leftrightarrow F(x, y) - k = 0$
- Normals given by gradient vector of F

Contours in 2D

- Assign geometric primitives to cells consisting of 2x2 grid points

Contours in 2D

- Assign geometric primitives to cells consisting of 2x2 grid points
  - Line segments

Contours in 2D

- Assign geometric primitives to cells consisting of 2x2 grid points
  - Line segments
- How do we know how to organize the primitives?

Contours in 2D

- Assign geometric primitives to cells consisting of 2x2 grid points
  - Line segments
- How do we know how to organize the primitives?
  - Signs of the values of corners of cells

Contours in 2D

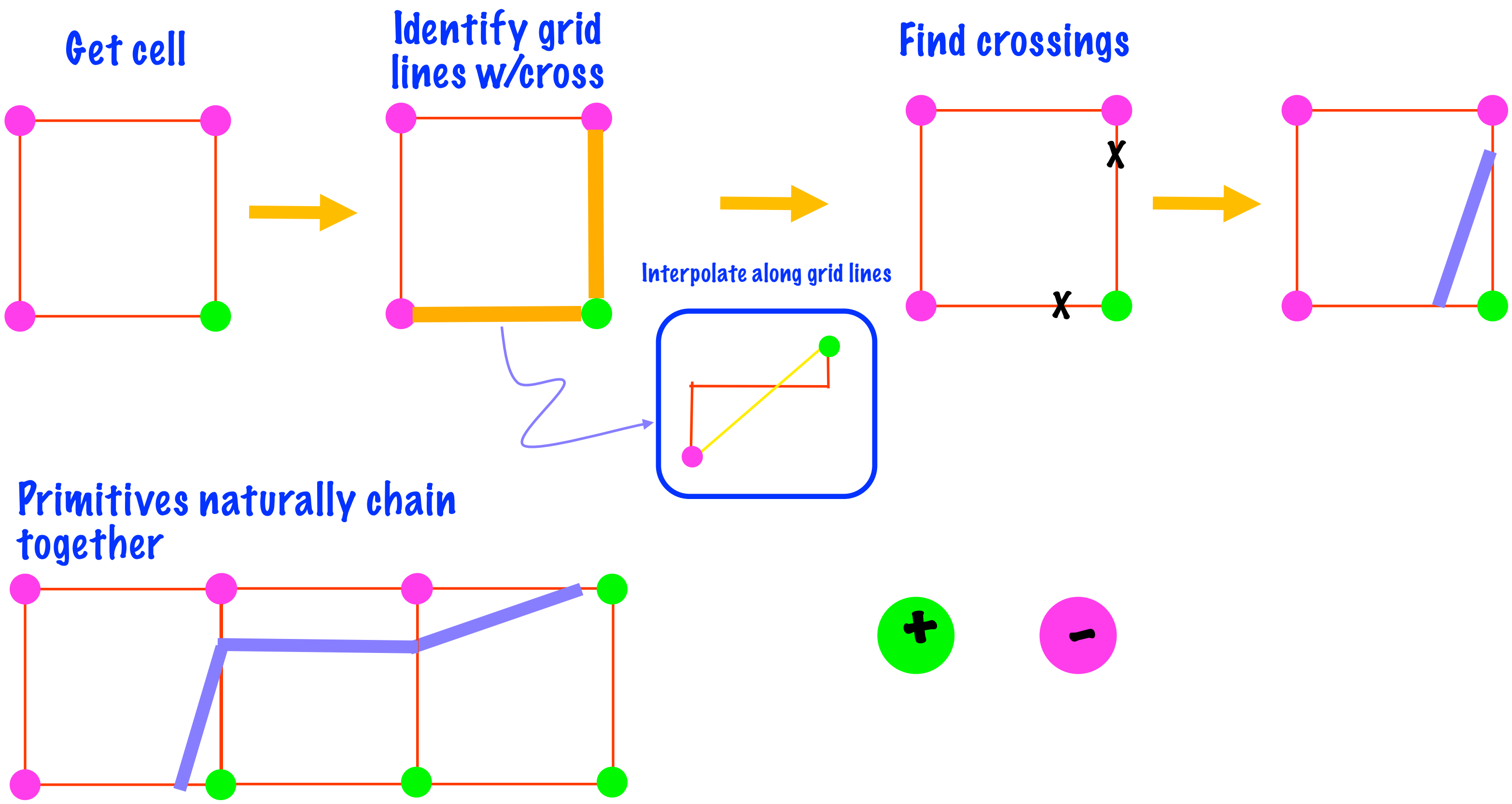
- Assign geometric primitives to cells consisting of 2x2 grid points
  - Line segments
- How do we know how to organize the primitives?
  - Signs of the values of corners of cells
- How do we know the position of the primitives?

Contours in 2D

- Assign geometric primitives to cells consisting of 2x2 grid points
  - Line segments
- How do we know how to organize the primitives?
  - Signs of the values of corners of cells
- How do we know the position of the primitives?
  - Interpolate along grid edges

Contours in 2D

- Idea: primitives must cross every grid line connecting two grid points of opposite sign



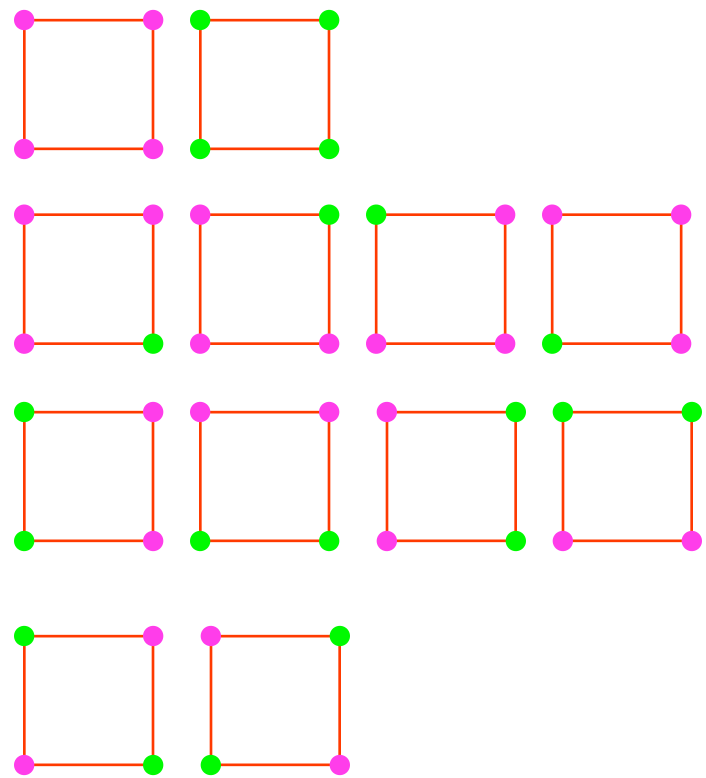
Questions

- How many grid lines with crossings can there be?
- What are the different configurations (adjacencies) of +/- grid points?

Cases

Case	Polarity	Rotation	Total
No Crossings	x2		2
Singlet	x2	x4	8
Double adjacent	x2	x2 (4)	4
Double Opposite	x2	x1 (2)	2

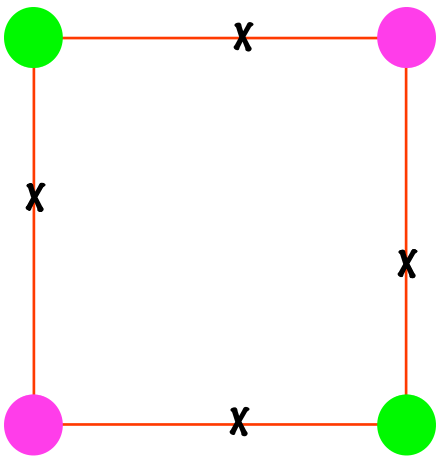
$16 = 2^4$



(x2 for polarity)

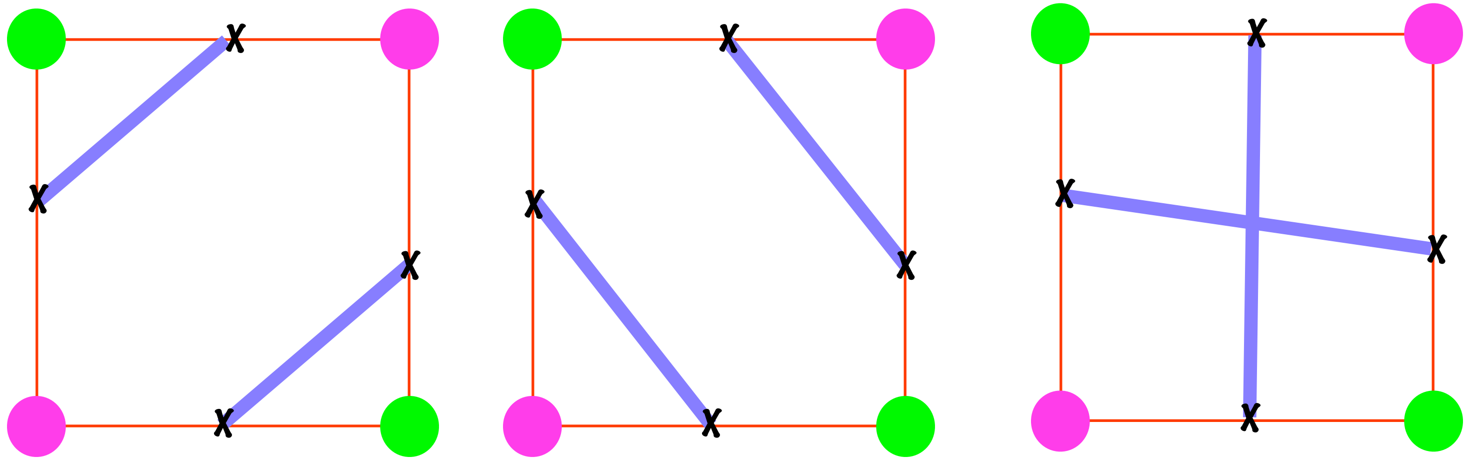
Ambiguities

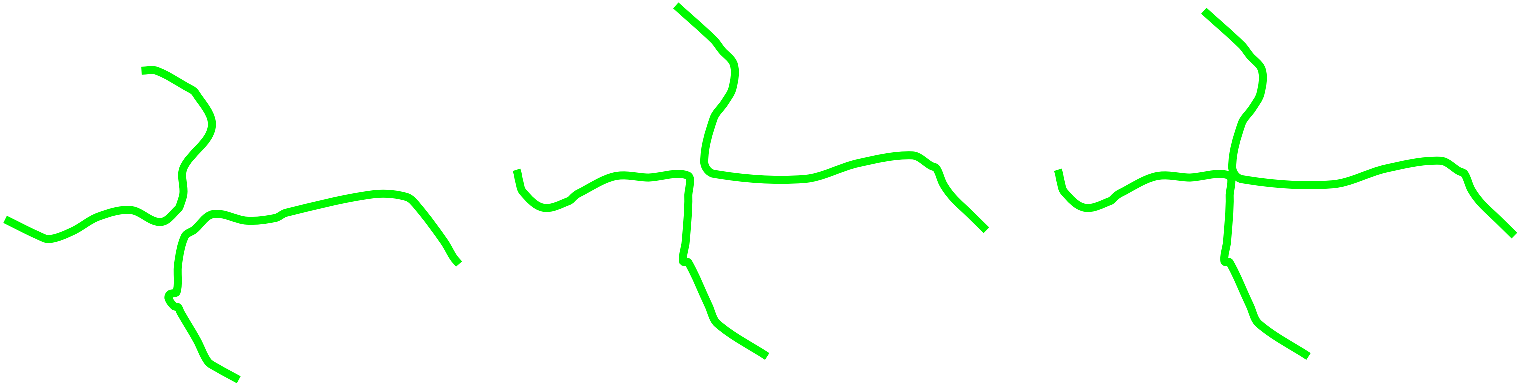
- How to form the lines?



Ambiguities

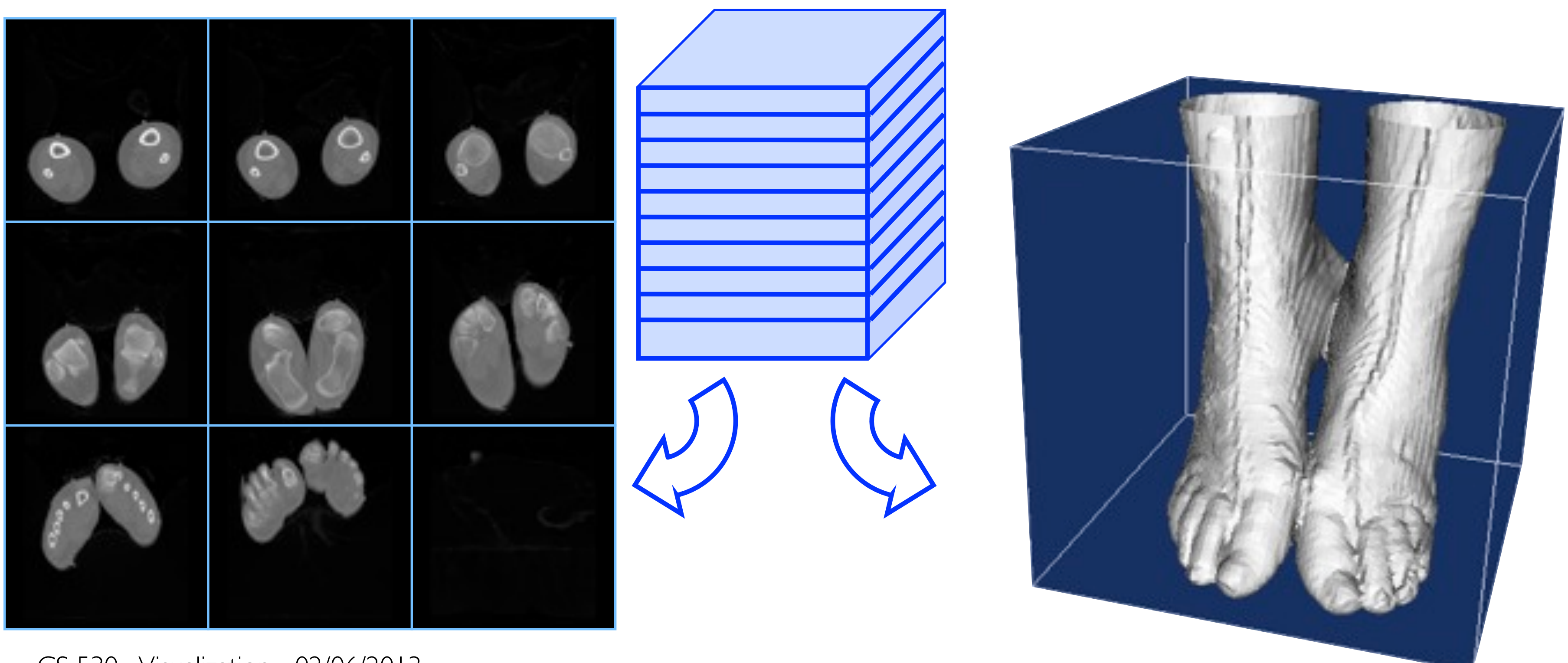
- Right or wrong?





Isosurfacing

- Have: a big 3D block of numbers ("scalars")
- Want: a picture
- Slicing shows data, but not its 3D shape
- Isosurfacing is one of the simplest ways



A little math

- Dataset: $v = f(x, y, z)$
- $f : \mathbb{R}^3 \rightarrow \mathbb{R}$
- Want to find $S_v = \{(x, y, z) | f(x, y, z) = v\}$
- All the locations where the value of f is v
- S_v: **isosurface** of f at v
  - In 2D: isocontours (some path)
  - In 3D: isosurface
- Why is this useful?

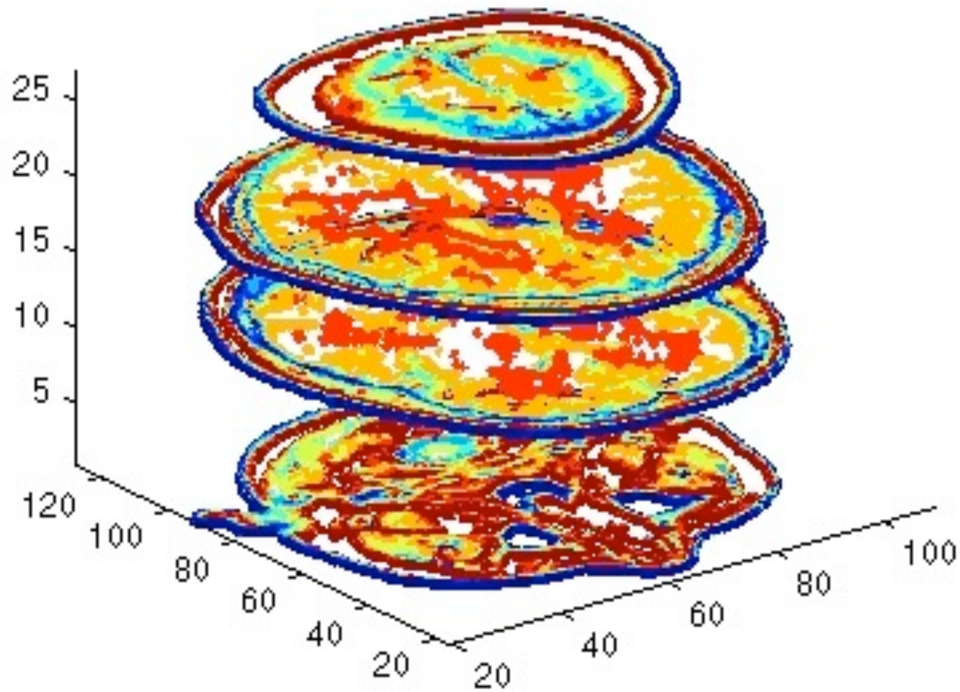
Surface Extraction (Isosurfacing)

Surface Extraction

- SLICING - Take a slice through the 3D volume (often orthogonal to one of the axes), reducing it to a 2D problem
  - Contour in 2D
  - Form polygons with adjacent polylines

Note analogous techniques in 2D visualization:
1D cross-sections, and contours (=isolines)

Isosurface from slices



Isosurface from slices

7

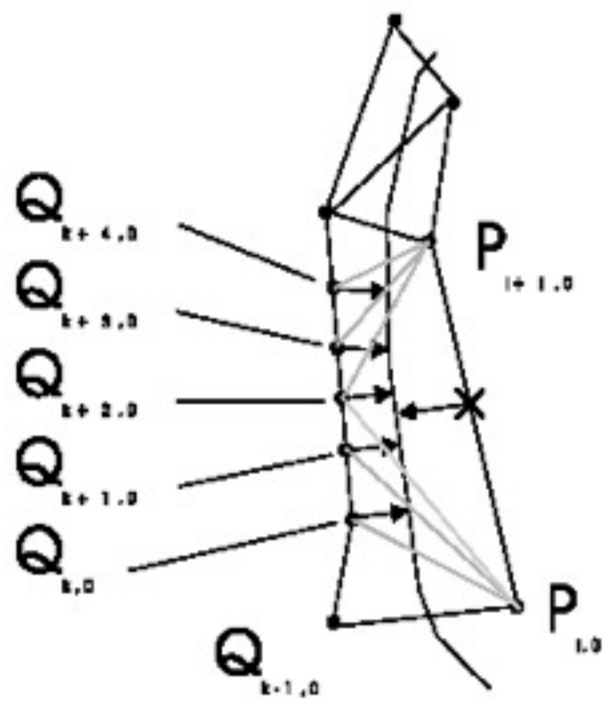
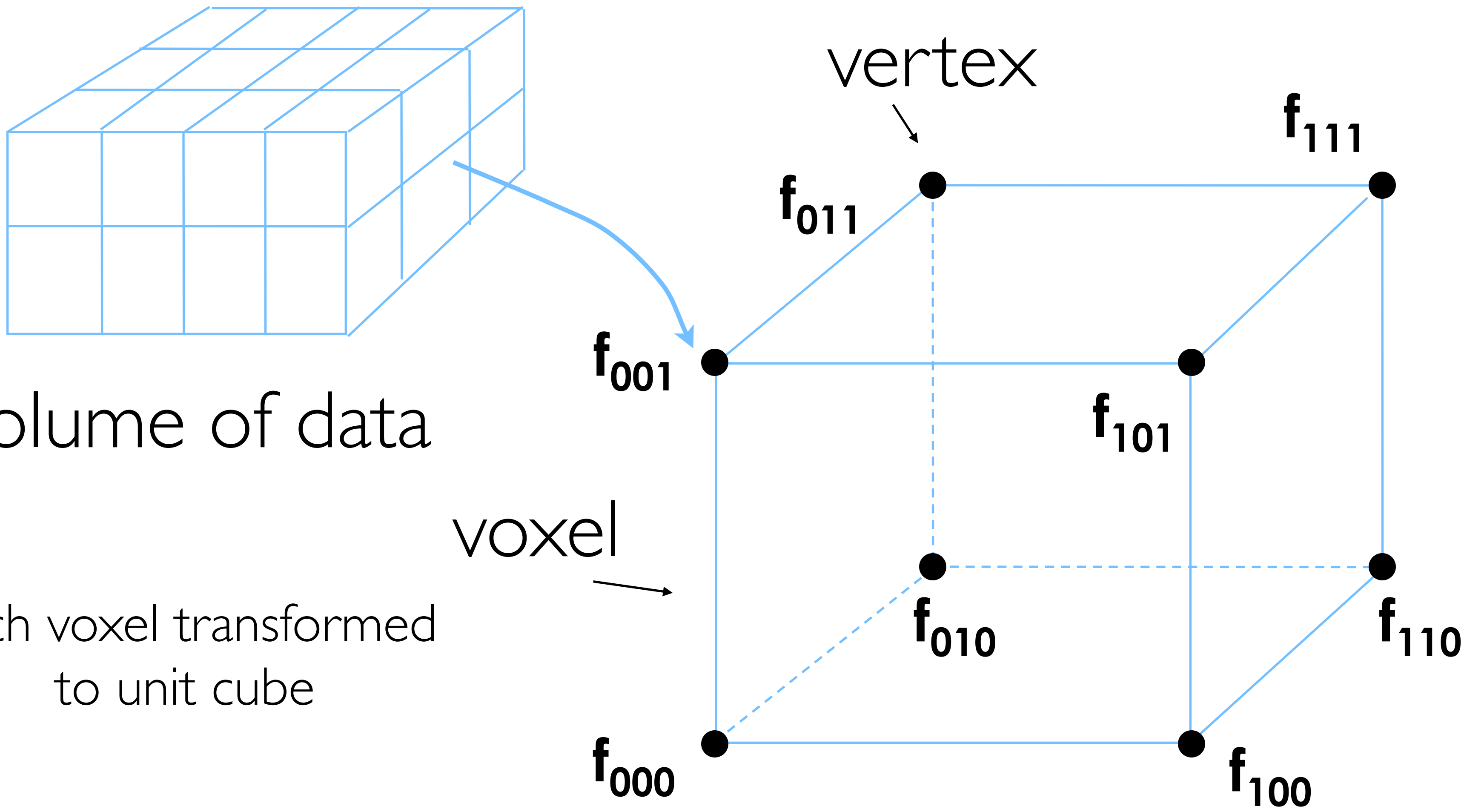


Fig. 6. Vertices $Q_{k,0}$ through $Q_{k+4,0}$ are connected to $P_{i,0}$ or $P_{i+1,0}$ resp. due to their correspondences on the medial axis.

Notations



Volume of data

Each voxel transformed to unit cube

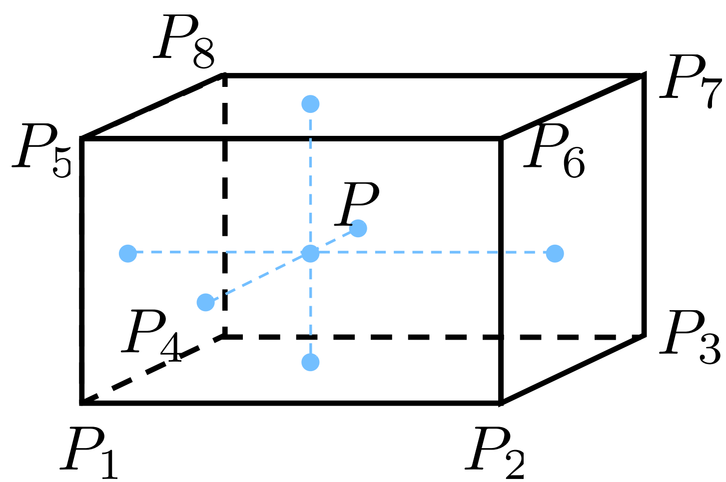
Trilinear Interpolation

- In a voxel
  - general formula

$$\phi(x,y,z) = axyz + bxy + cxz + dyz + ex + fy + gz + h$$

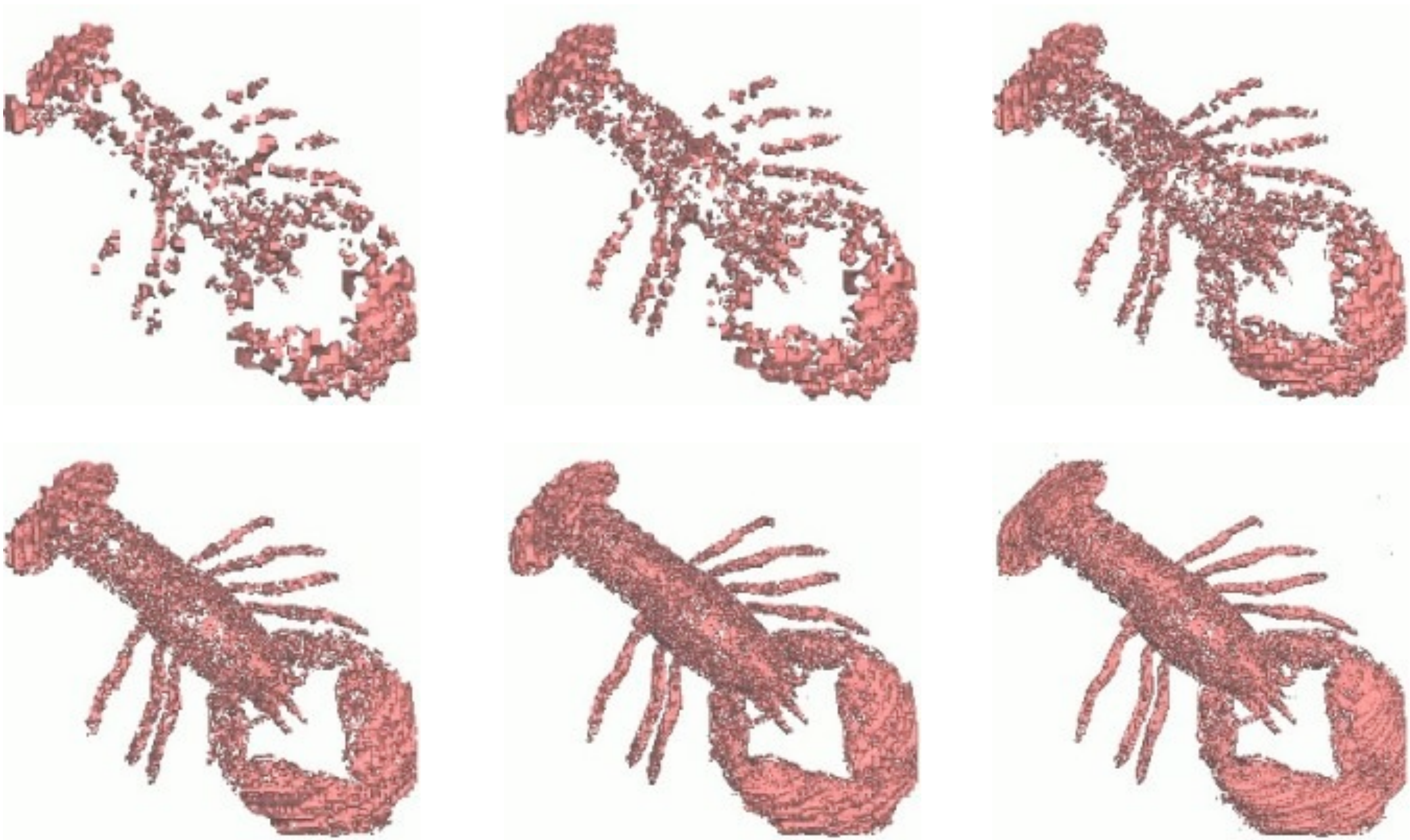
  - with local coordinates

$$\begin{aligned} P = P_1 & \\ & +u(P_2 - P_1) \\ & +v(P_4 - P_1) \\ & +w(P_5 - P_1) \\ & +uv(P_1 - P_2 + P_3 - P_4) \\ & +uw(P_1 - P_2 + P_6 - P_5) \\ & +vw(P_1 - P_4 + P_8 - P_5) \\ & +uvw(P_1 - P_2 + P_3 - P_4 + P_5 - P_6 + P_7 - P_8) \end{aligned}$$



Isosurfacing

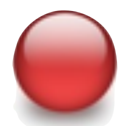
Lobster – Increasing the Threshold Level



From University of Bonn

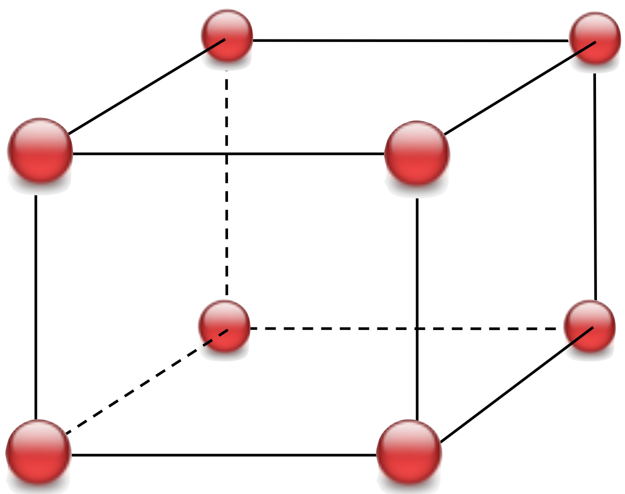
Isosurface Construction

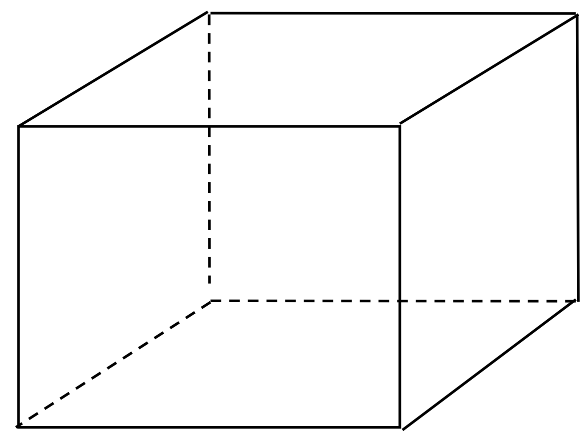
For simplicity, we shall work with zero level isosurface, and denote

positive vertices as 

There are **8** vertices, each can be positive or negative - so there are 2^8 = 256 different cases

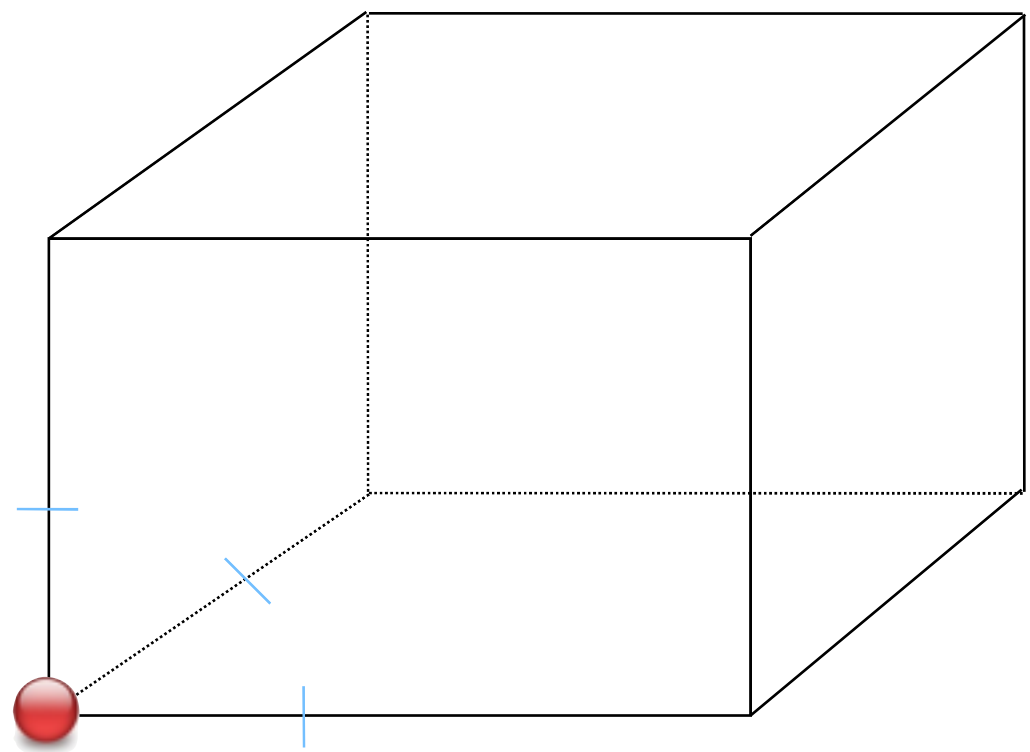
Straightforward Cases





There is no portion of the isosurface inside the cube!

Isosurface Construction One Positive Vertex - 1

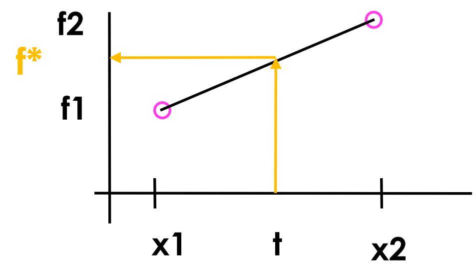


Intersections with edges found by inverse linear interpolation (as in contouring)

Note on Inverse Linear Interpolation

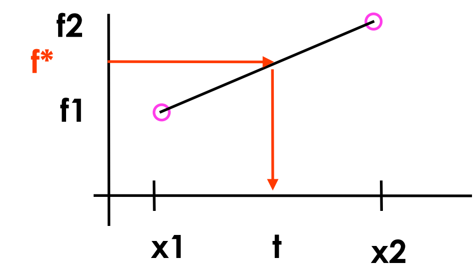
- The linear interpolation formula gives value of f at specified point t:

  $f(x^*) = f1 + t\ (\ f2 - f1\ )$

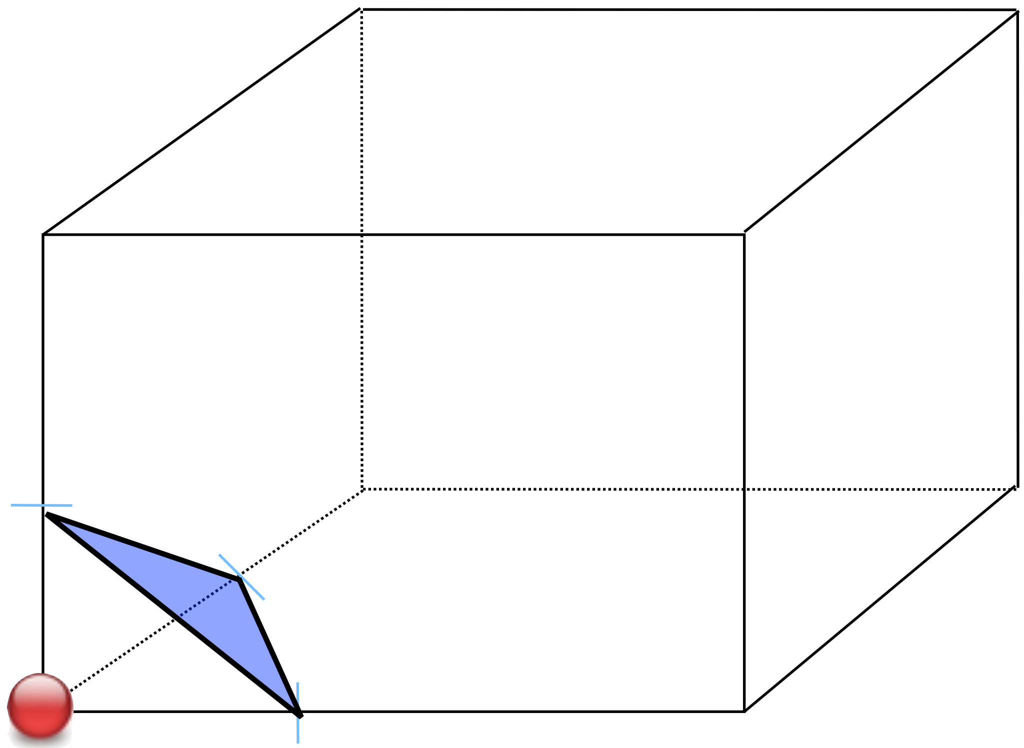


- Inverse linear interpolation gives value of t at which f takes a specified value f*

  $t = (f^* - f1)/(f2 - f1)$



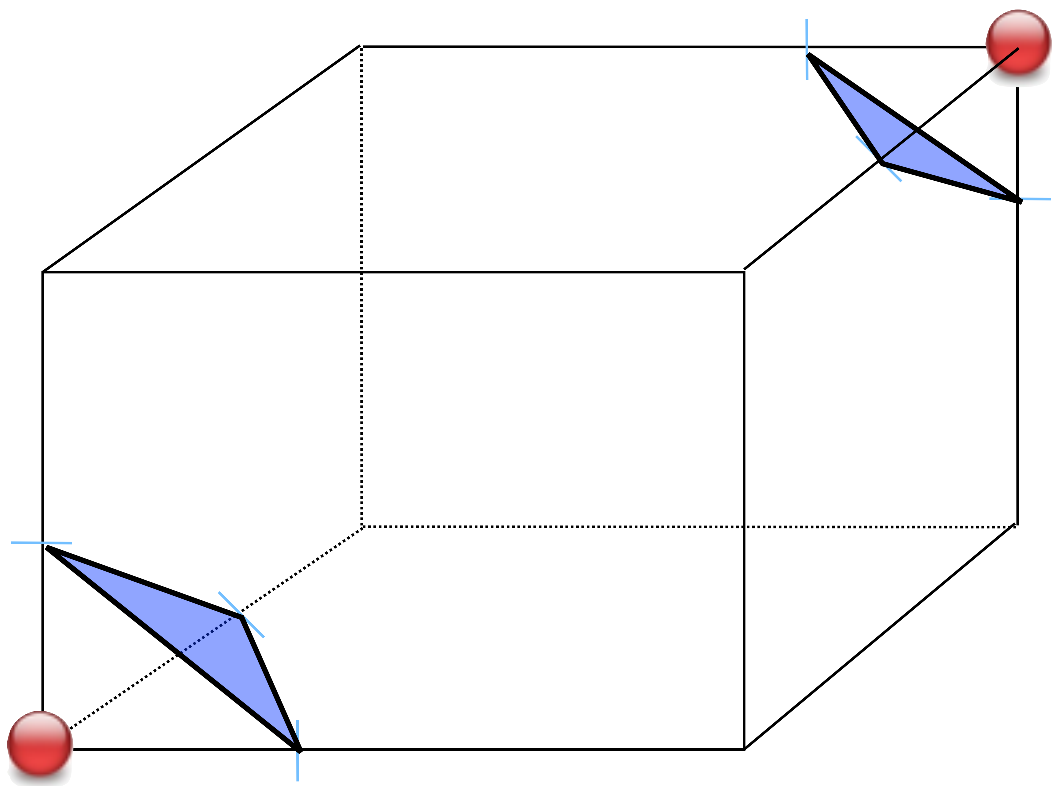
Isosurface Construction - One Positive Vertex - 2



Joining edge intersections across faces forms a triangle as part of the isosurface

Isosurface Construction

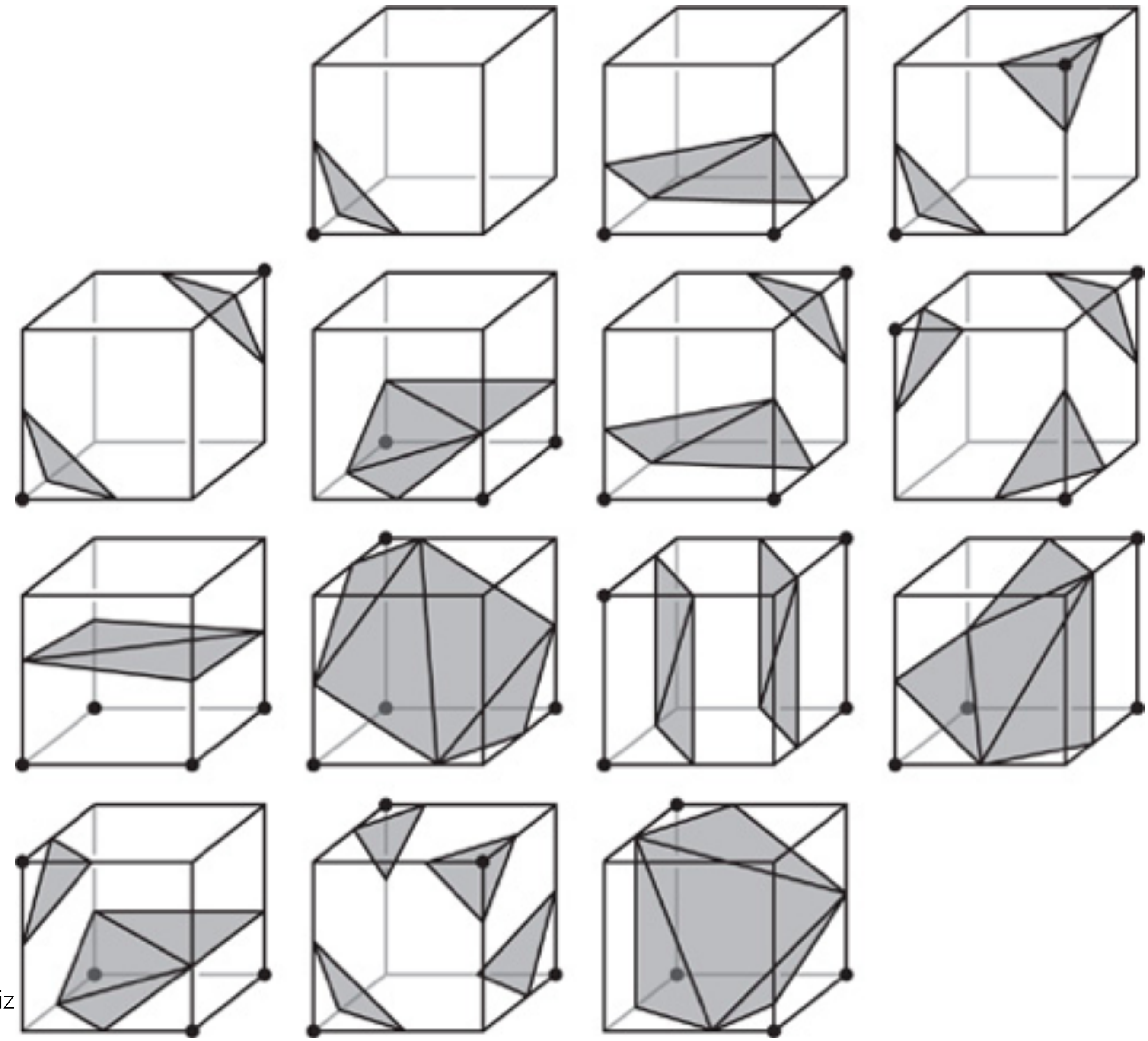
Positive Vertices at Opposite Corners



Isosurface Construction

- One can work through all 256 cases in this way - although it quickly becomes apparent that many cases are similar.
- For example:
  - 2 cases where all are positive, or all negative, give no isosurface
  - 16 cases where one vertex has opposite sign from all the rest
- In fact, there are only 15 topologically distinct configurations

Canonical Cases



Canonical Cases

- The 256 possible configurations can be grouped into these 15 canonical cases on the basis of complementarity (swapping positive and negative) and rotational symmetry.
- The advantage of doing this is for ease of implementation - we just need to code 15 cases not 256

Isosurface Construction

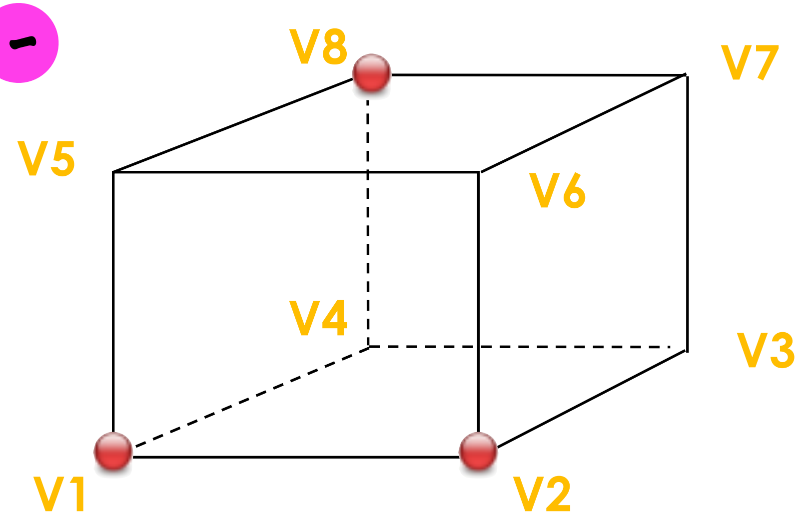
- In some configurations, just one triangle forms the isosurface
- In other configurations ...
  - ...there can be several triangles
  - …or a polygon with 4, 5 or 6 points which can be triangulated
- A software implementation will have separate code for each configuration

Marching Cubes Algorithm

- Step 1: Classify the eight vertices relative to the isosurface value

8-bit index ; 1 + ; 0 -

1	1	0	0	0	0	0	1



Code identifies edges intersected:
V1V4; V1V5; V2V3; V2V6; V5V8; V7V8; V4V8

Marching Cubes Algorithm

- Step 2: Look up table which identifies the canonical configuration
- For example:

00000000 Configuration 0
10000000 Configuration 1
01000000 Configuration 1
...
11000001 Configuration 6
...
11111111 Configuration 0

256 entries in table

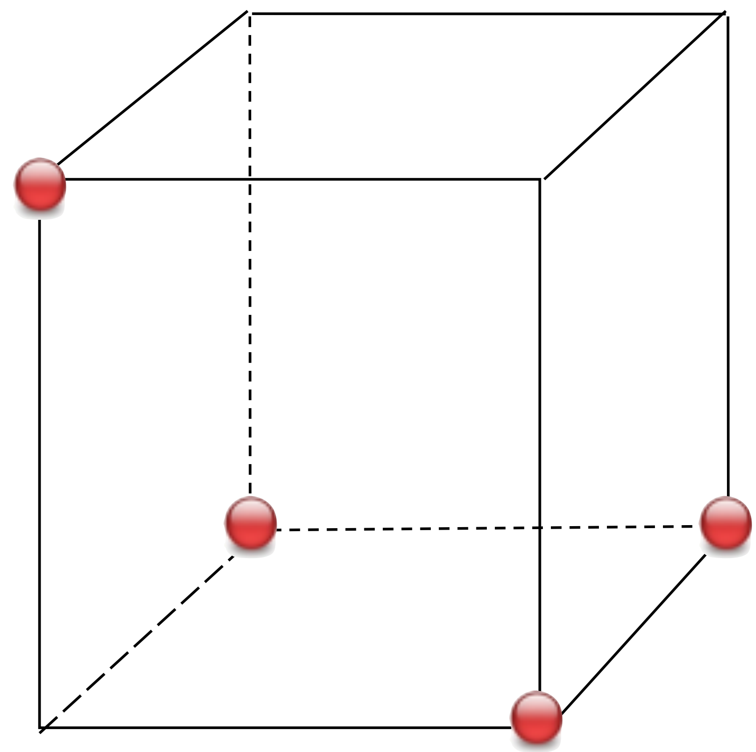
Marching Cubes Algorithm

- Step 3: Inverse linear interpolation along the identified edges will locate the intersection points
- Step 4: The canonical configuration will determine how the pieces of the isosurface are created (0, 1, 2, 3 or 4 triangles)
- Step 5: Pass triangles to renderer for display

Algorithm marches from cube to cube between slices, and then from slice to slice to produce a smoothly triangulated surface

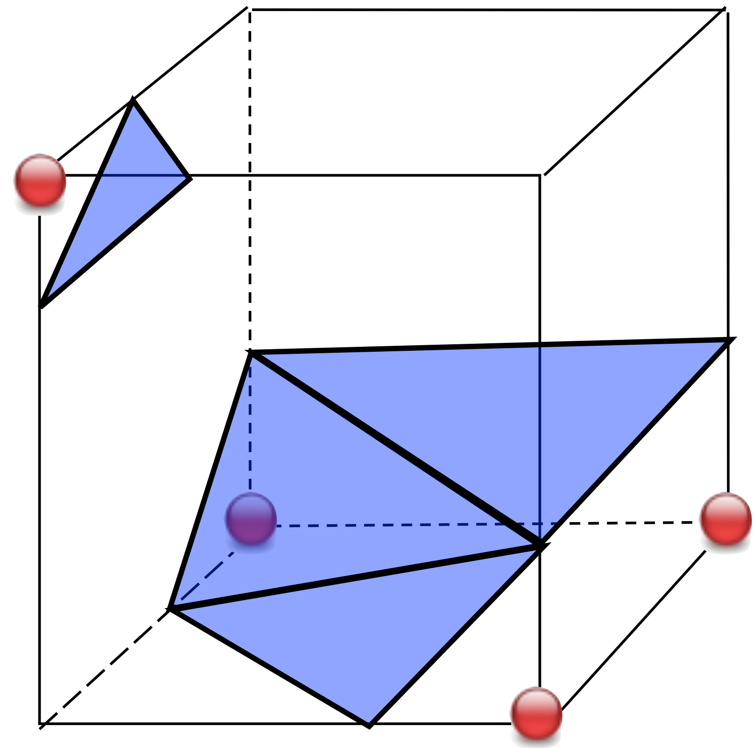
- Case 12 has three positive vertices on the bottom plane; and one positive vertex on the top plane, directly above the single negative on the bottom plane.

- Without looking at the answer….Try to work out the isosurface!



- Case 12 has three positive vertices on the bottom plane; and one positive vertex on the top plane, directly above the single negative on the bottom plane.

- Without looking at the answer….Try to work out the isosurface!

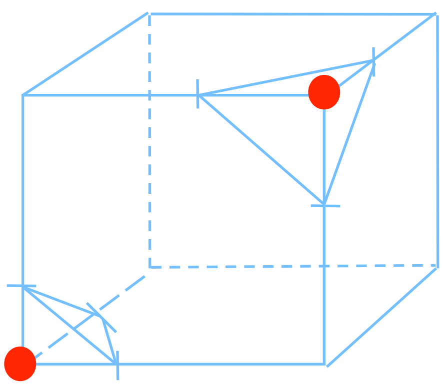


Isosurfacing by Marching Cubes Algorithm

- Advantages
  - isosurfaces good for extracting boundary layers
  - surface defined as triangles in 3D - well-known rendering techniques available for lighting, shading and viewing ... with hardware support
- Disadvantages
  - shows only a slice of data
  - ambiguities?

Ambiguities

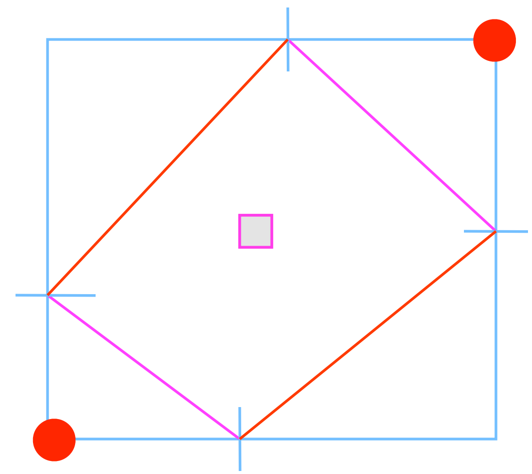
- Marching cubes suffers from exactly the same problems that we saw in contouring



Case 3: Triangles are chosen to slice off the positive vertices - but could they have been drawn another way?

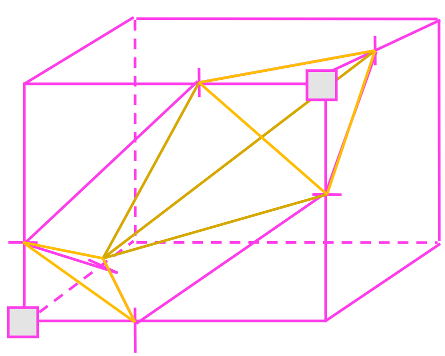
Ambiguities on Faces

- On the front face, we have exactly the same ambiguity problem we had with contouring
- We can determine which pair of intersections to connect by looking at value at saddle point



Ambiguities on Faces

- Trouble occurs because:
  - trilinear interpolant is only linear along the edges
  - on a face, it becomes a bilinear function ... and for correct topology we must join the correct pair of intersections



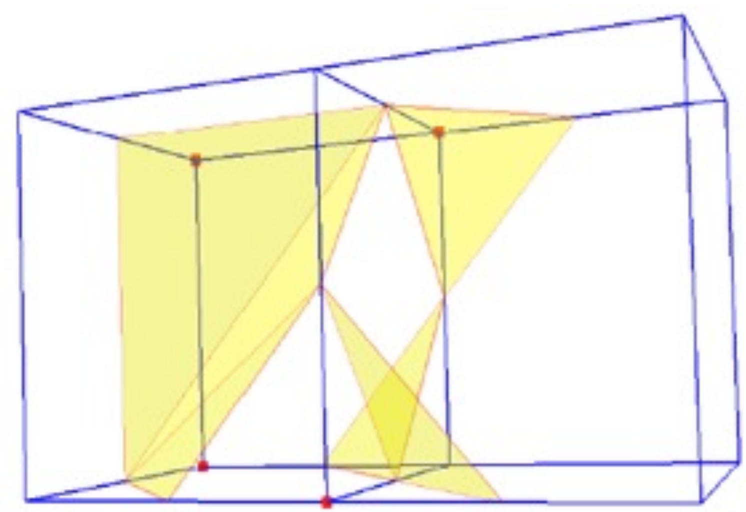
Case 3 has two triangle pieces cutting off corners!

.. but here is another interpretation!

6 configurations include ambiguous faces

Holes in Isosurfaces

- Because of the ambiguity, early implementations which did not allow for this could leave holes where cells join



Cases 12 and 3 in adjoining cells can cause holes

Resolving the Face Ambiguities

- Using the saddle point method to determine the correct behaviour on a face
    - generates sub-cases for each of the 6 ambiguous configurations
    - which sub-case is chosen depends on the value of the saddle-point on the face
    - note that some configurations have several ambiguous faces so many subcases arise - eg see config 13!
    - if we do not extend the 15 cases there is chance of holes appearing in surface

Trilinear Interpolant

- The trilinear function:

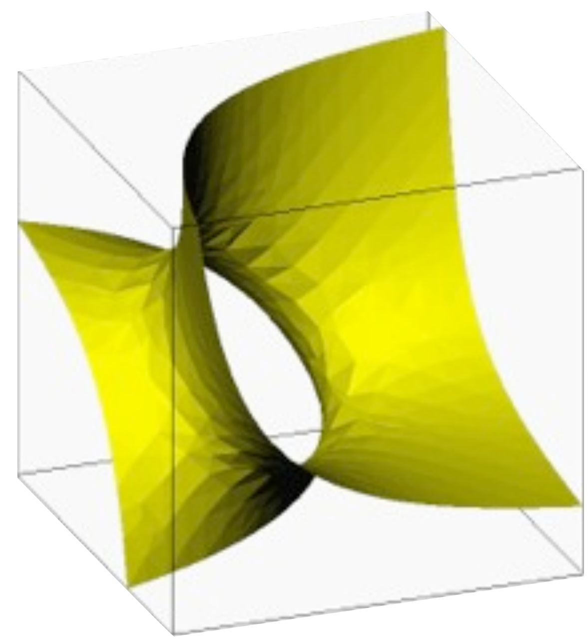
$$f(x,y,z) = f_{000}(1-x)(1-y)(1-z) + f_{100}x(1-y)(1-z) + f_{010}(1-x)y(1-z) + f_{001}(1-x)(1-y)z + f_{110}xy(1-z) + f_{101}x(1-y)z + f_{011}(1-x)yz + f_{111}xyz$$

is deceptively complex!

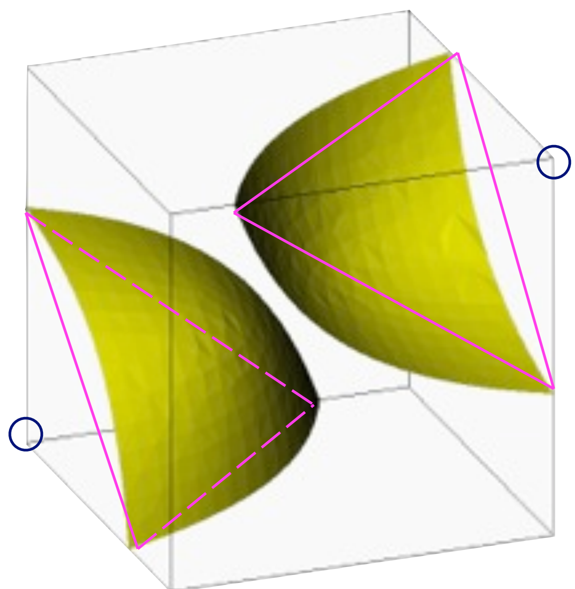
- For example, the isosurface of

$$f(x,y,z) = 0$$

is a cubic surface



Accurate Isosurface of Trilinear Interpolant



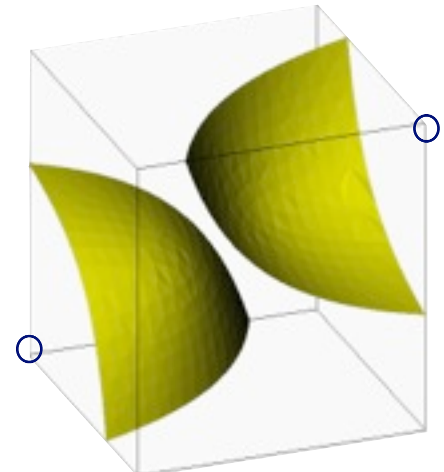
True isosurface of a trilinear interpolant is a curved surface

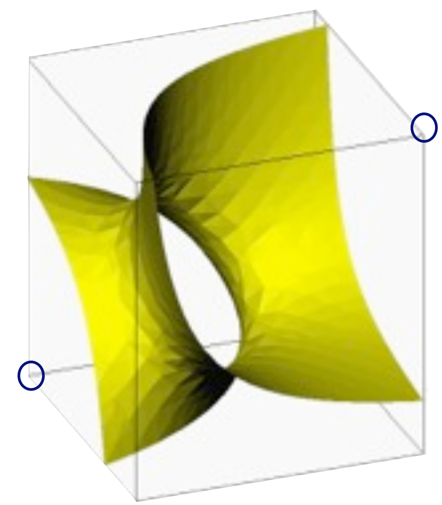
cf contouring where contours are hyperbola

We are approximating by the triangles shown

Interior Ambiguities

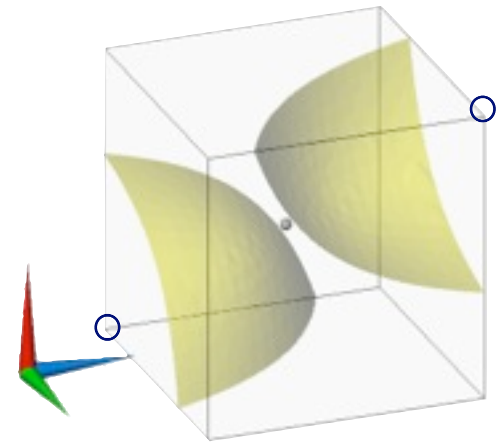
- In some cases there can also be ambiguities in the interior
- Consider case where opposite corners are positive
- Two possibilities: separated or tunnel

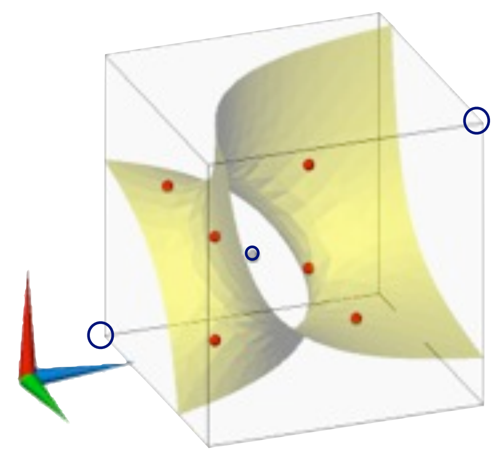




Resolving the Interior Ambiguity

- Decided by value at body saddle point ($f_x = f_y = f_z = 0$)
  - Negative: two separate shells
  - Positive: tunnel



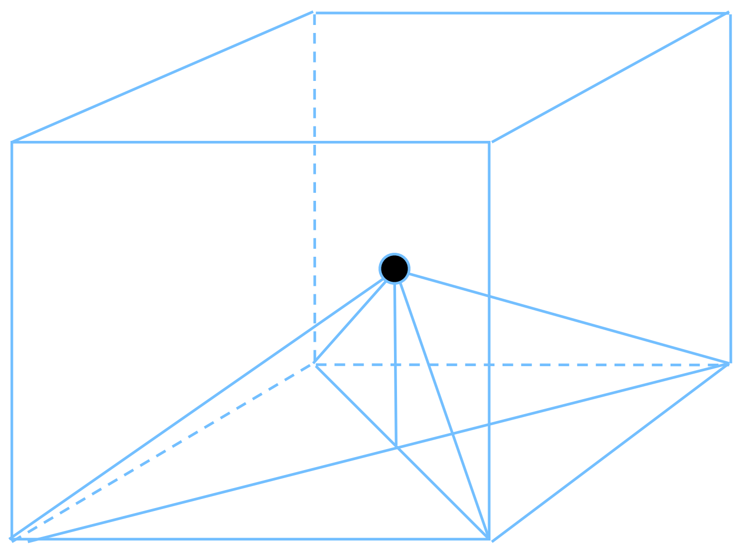


Marching Tetrahedra

- As in contouring, another solution is to divide into simpler shapes - here they are tetrahedra

24 tetrahedra in all

Value at centre = average of vertex values



Fit linear function in each tetrahedron:

$f(x,y,z) = a + bx + cy + dz$

Isosurface of linear function is triangle

Marching Tetrahedra

- A disadvantage of the ‘24’ marching tetrahedra is the large number of triangles which are created - slowing down the rendering time
- There are versions that just use 5 tetrahedra

Credits and References

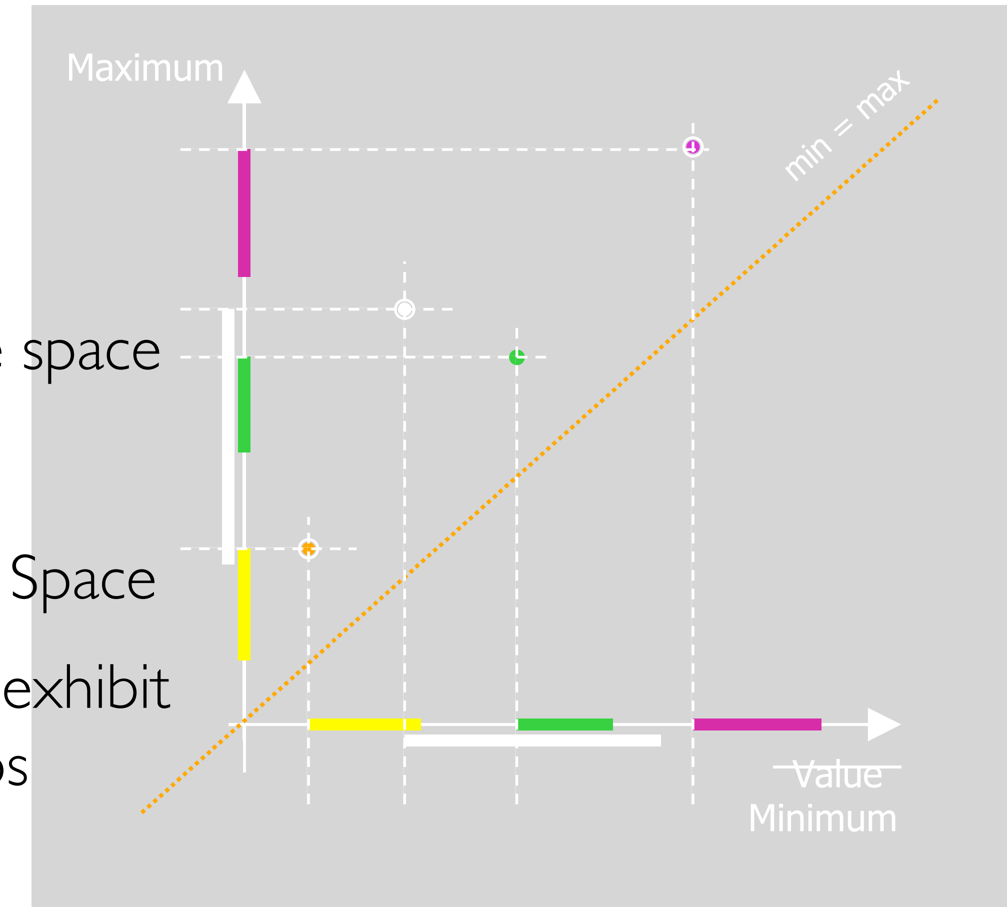
- Original marching cubes algorithm
  - Lorensen and Cline (1987)
- Face ambiguities
  - Nielson and Hamann (1992)
- Interior ambiguities
  - Chernyaev (1995)
- Accurate marching cubes
  - Lopes and Brodlie (2003)

The Span Space

The Span Space

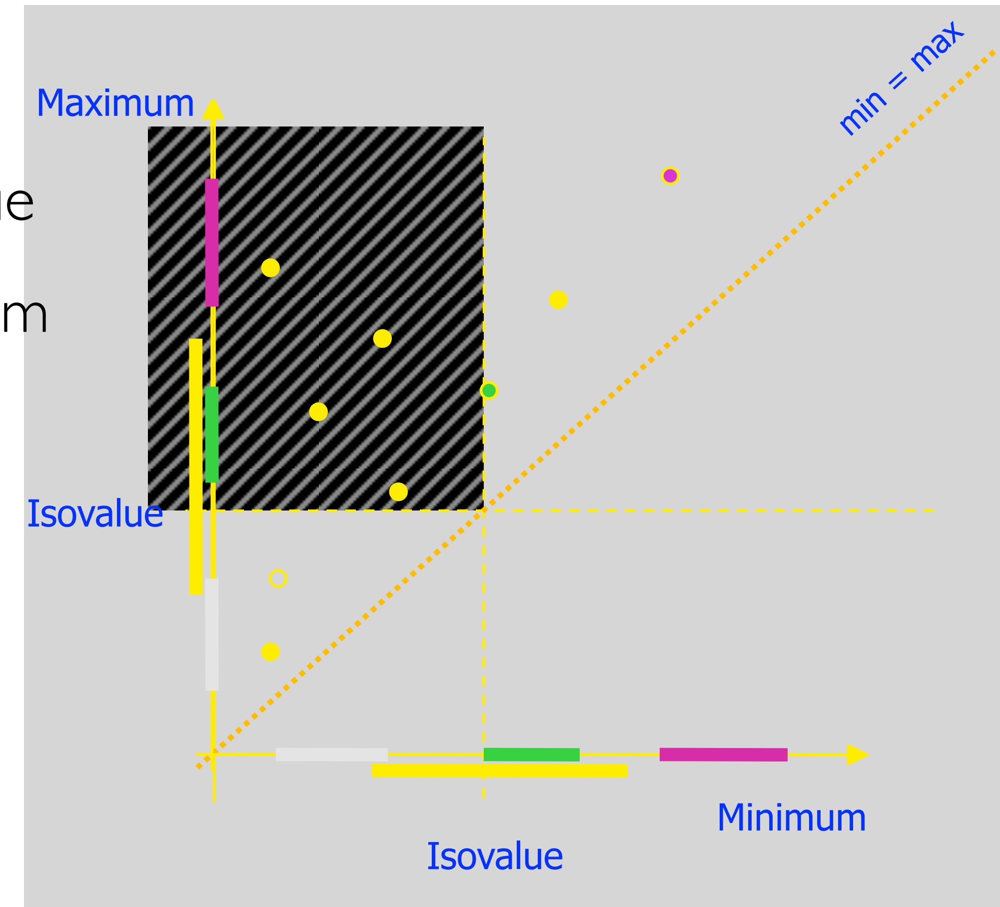
Livnat, Shen, Johnson 96

- Given:
  - Data cells in 8D
- Past (active list):
  - Intervals in a 1D Value space
- New:
  - Points in the 2D Span Space
  - Benefit: Points do not exhibit any spatial relationships



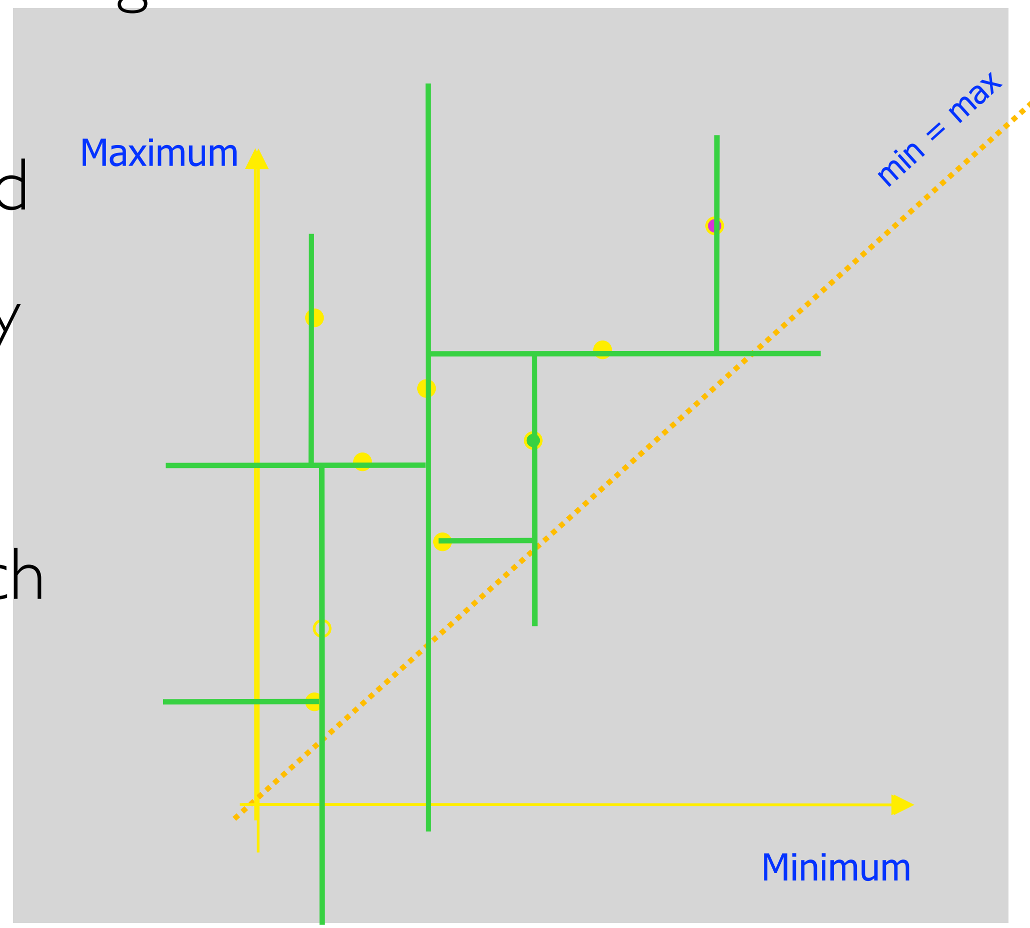
The Span Space

- Search
  - Find all the points
    - minimum < isovalue
    - isovalue < maximum
  - Semi-infinite area
    - Quadrant



The Span Space

- Search for rectangles using Kd-tree*
  - O(n log(n)) to build
  - Search Complexity
    - O(√ n+k)
- Recursively divide each axis along median



* more on this later