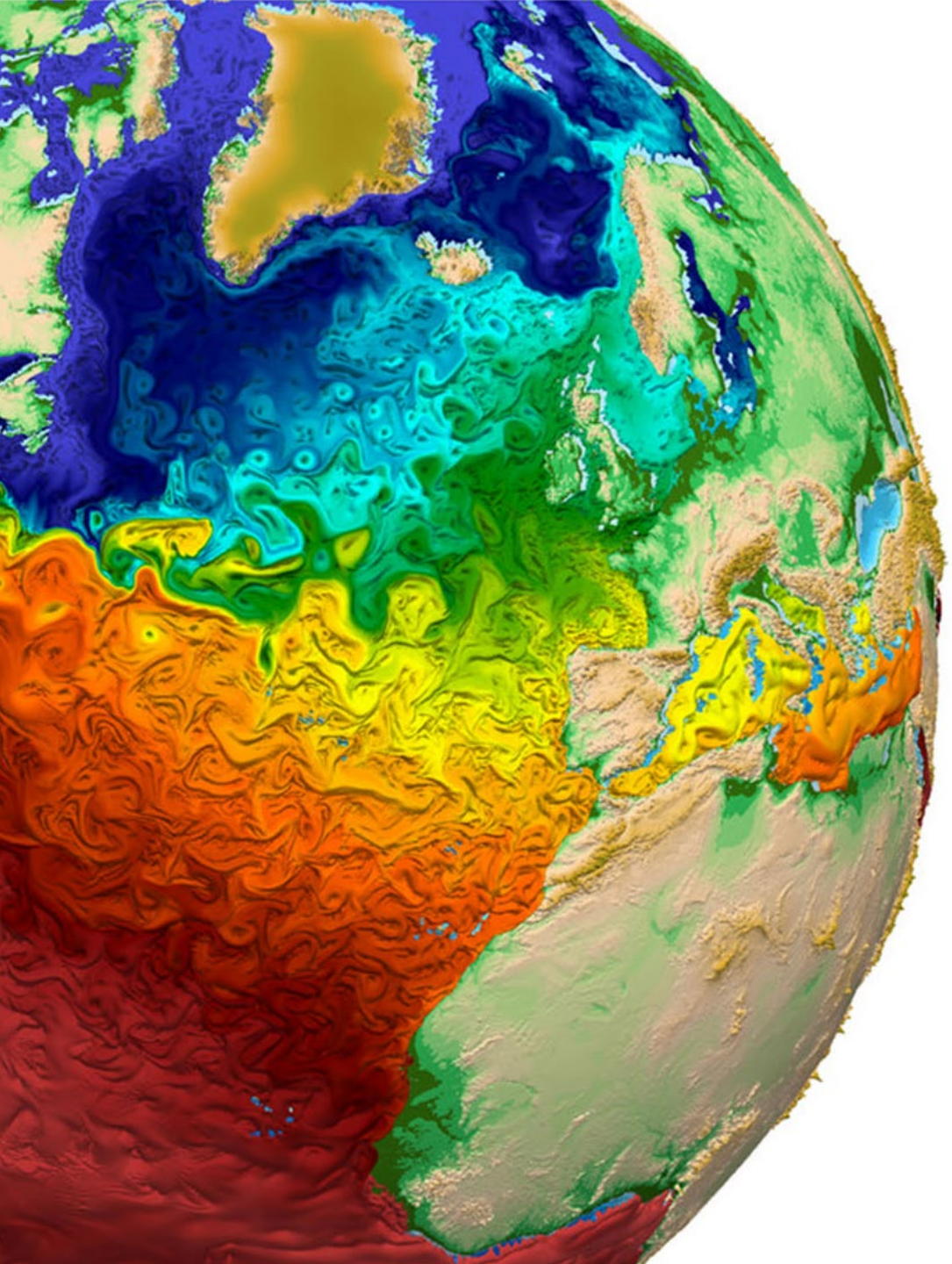


Contouring

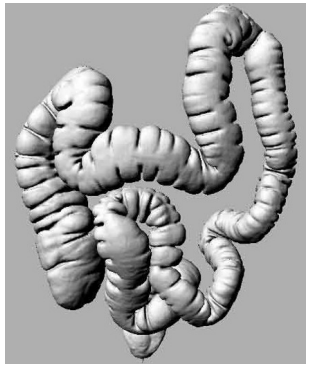
Marching Cubes

Scientific Visualization

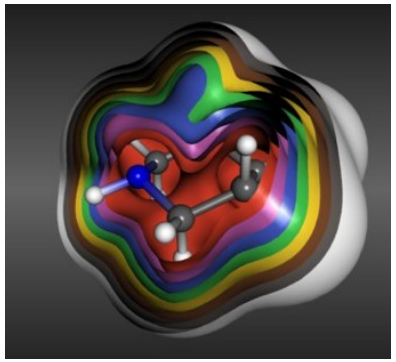
Professor Eric Shaffer

ILLINOIS

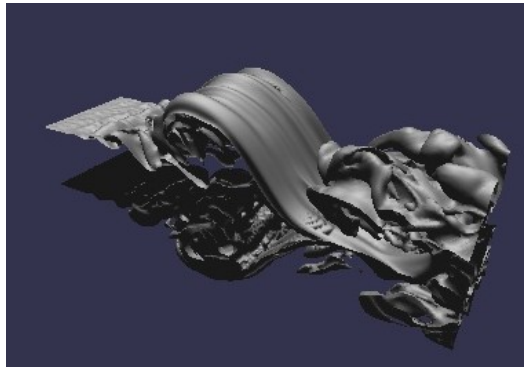
Contouring in 3D



colon (CT dataset)



electron density in molecule

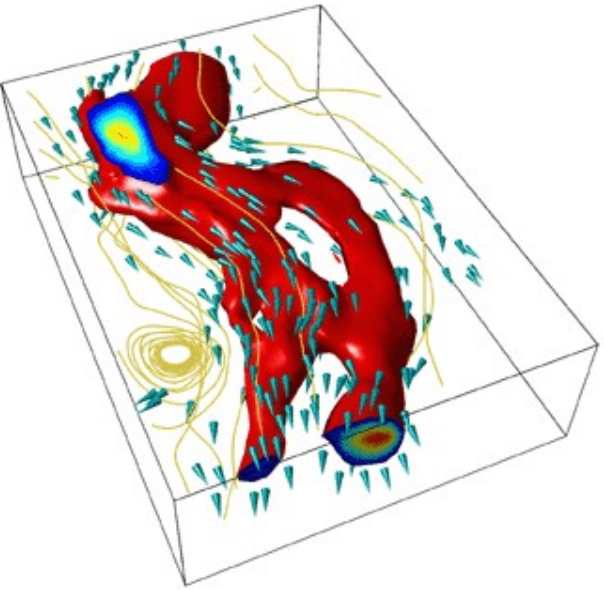


velocity in 3D fluid flow

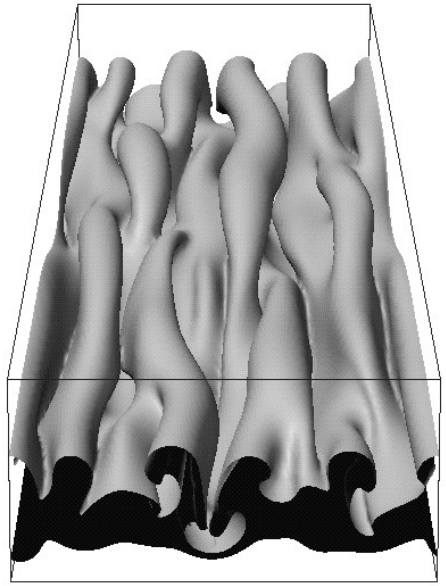
Scalar fields in 3D space are volumes

Contouring a volume → isosurface

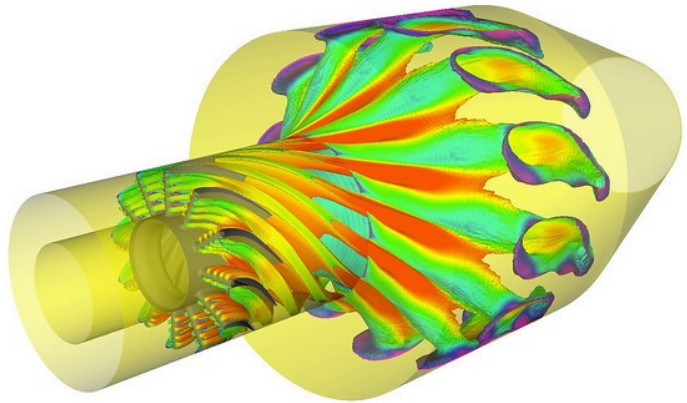
Many uses…
e.g. finding boundary between materials



velocity in 3D fluid flow



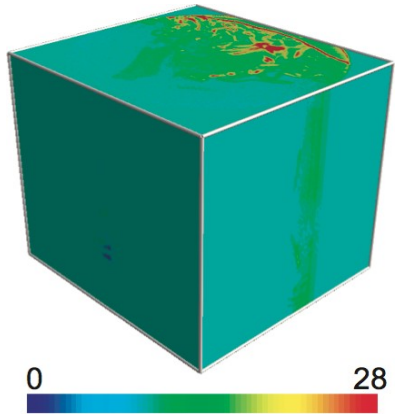
magnetic field in sunspots



fuel concentration, colored
by temperature in jet engine

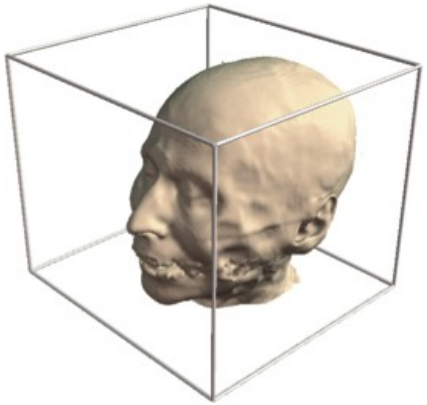
ILLINOIS

Isosurfaces

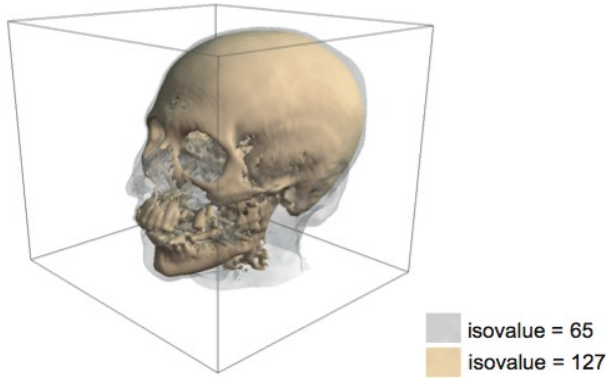


scalar CT volume (tissue density)

Isosurfaces



isosurface for scalar value corresponding to skin



isosurfaces for skin and bone

ILLINOIS

Marching Cubes

TITLE	CITED BY	YEAR
Marching cubes: A high resolution 3D surface construction algorithm WE Lorensen, HE Cline ACM siggraph computer graphics 21 (4), 163-169	16213	1987

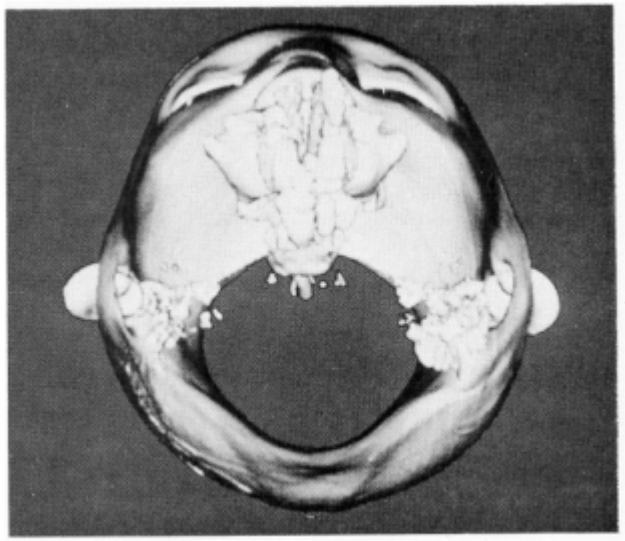


Figure 10. Soft Tissue, Top View.

Developed by William E. Lorensen and Harvey E. Cline at General Electric Medical.

Designed to efficiently visualize data from CT and MRI devices

Famously patented algorithm (application in 1985) with patent now expired

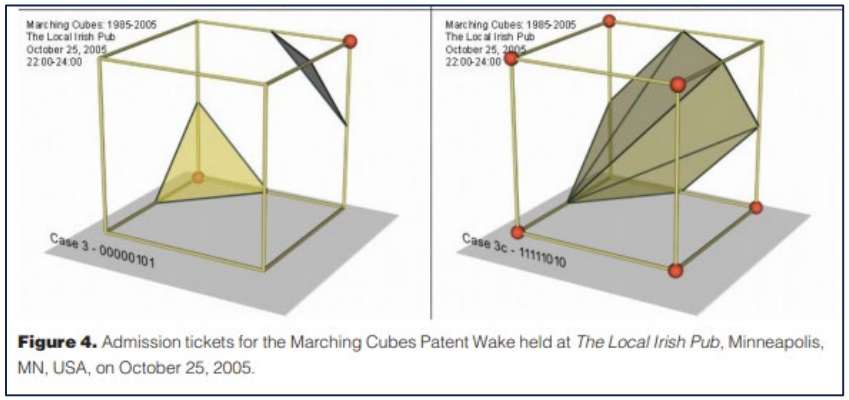


Figure 4. Admission tickets for the Marching Cubes Patent Wake held at *The Local Irish Pub*, Minneapolis, MN, USA, on October 25, 2005.

Home / Magazines / IEEE Computer Graphics and Applications / 2020.02

Remembering Bill Lorensen: The Man, the Myth, and Marching Cubes

March-April 2020, pp. 112-118, vol. 40

DOI Bookmark: 10.1109/MCG.2020.2971168

Bill Lorensen died on December 12, 2019.

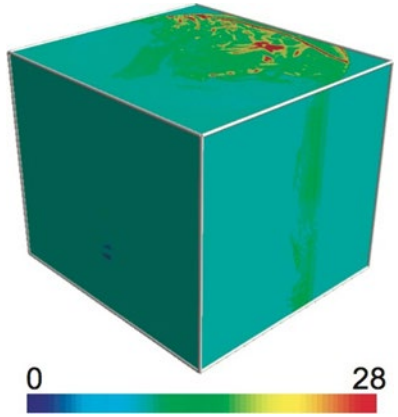
ILLINOIS

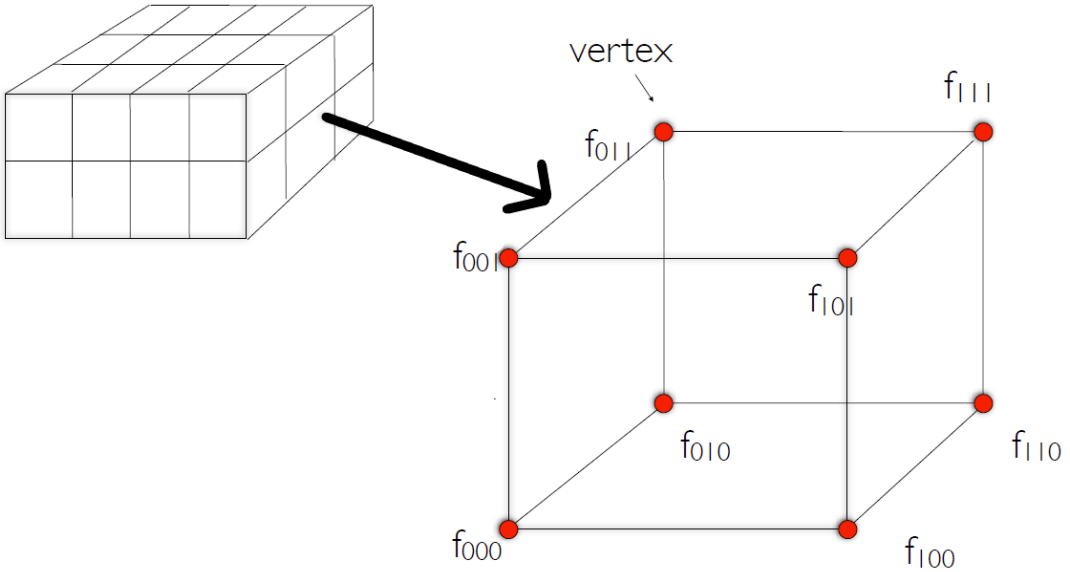
Marching Cubes

Scalar volume: $f : D \subset \mathbb{R}^3 \to \mathbb{R}$

$$(x, y, z) \mapsto f(x, y, z)$$

Want to find $S_v = \{(x, y, z) | f(x, y, z) = v\}$



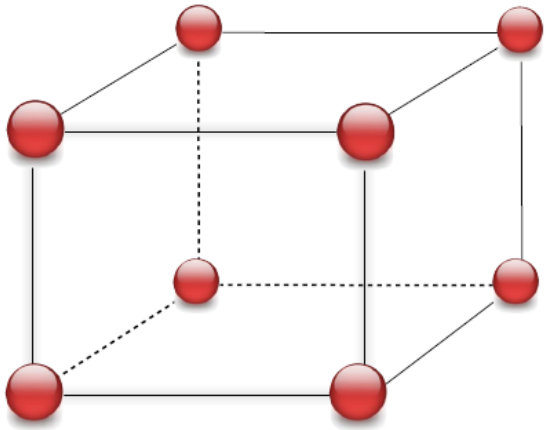


- We seek to construct a polygonal approximation to S_v
- Function is sampled at vertices of a regular cuboid grid
- Generate isosurface cell-by-cell
- Polygons generated across the cells

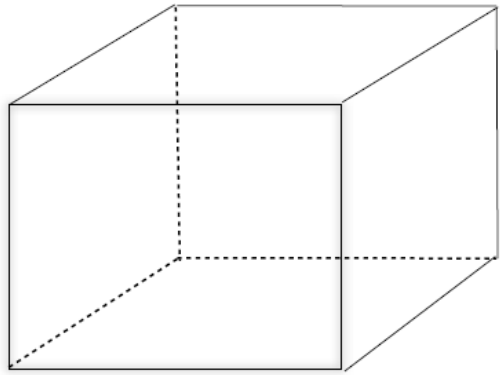
ILLINOIS

Labeling Cubes

Label each vertex as greater than or less than the isovalue
For example:



All inside $f > f_0$

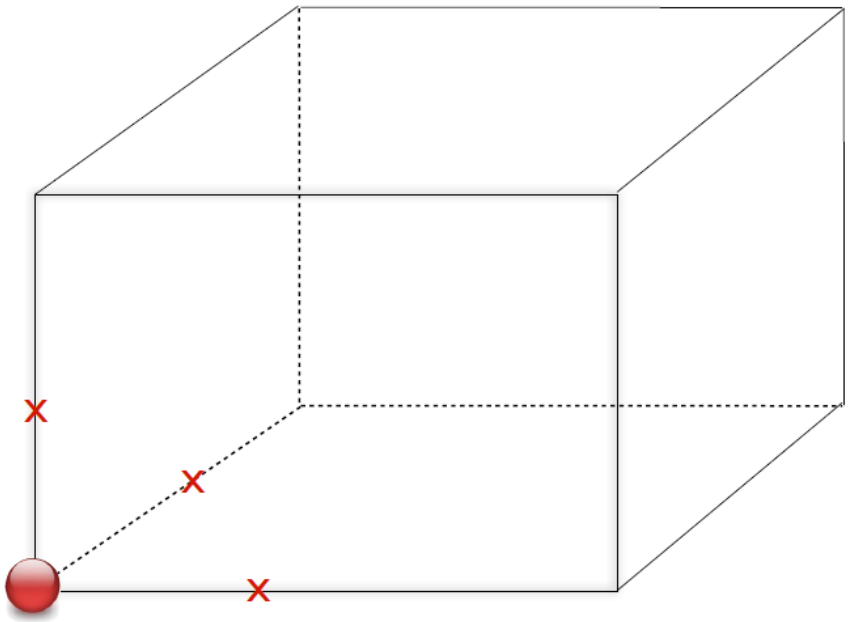


All outside $f \leq f_0$

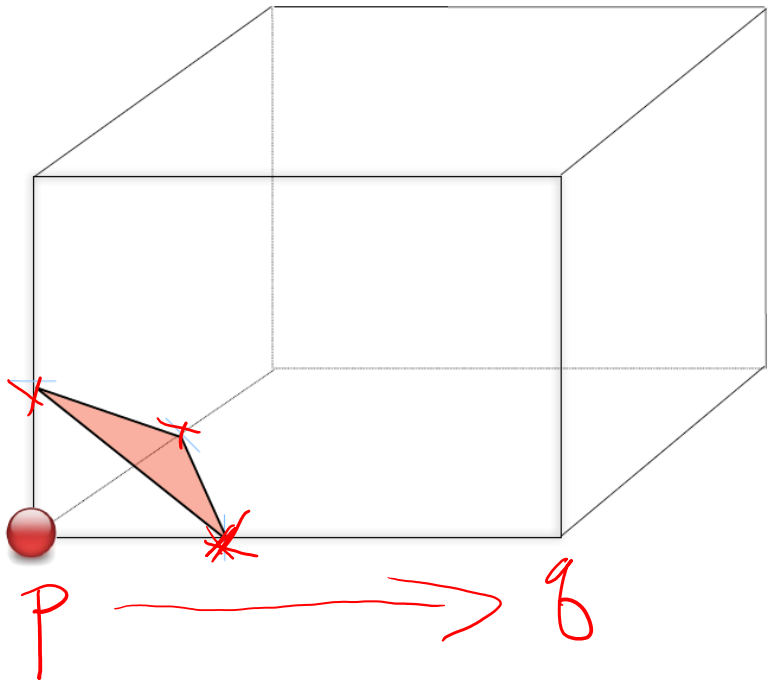
Bipolar Edges

Edges with two differently classified endpoints are bipolar

The isosurface will cut the edge



Generating a Polygon

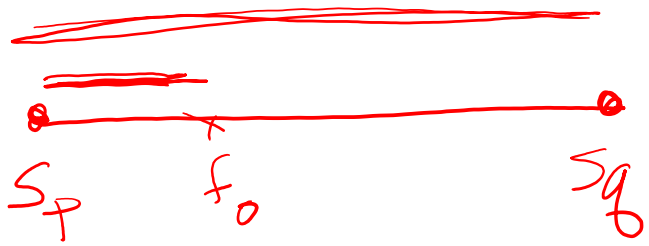


Use linear interpolation to place the polygon vertices on the edges

Grid edge $[p, q]$ with $f(p) = s_p$ and $f(q) = s_q$ and isovalue f_0

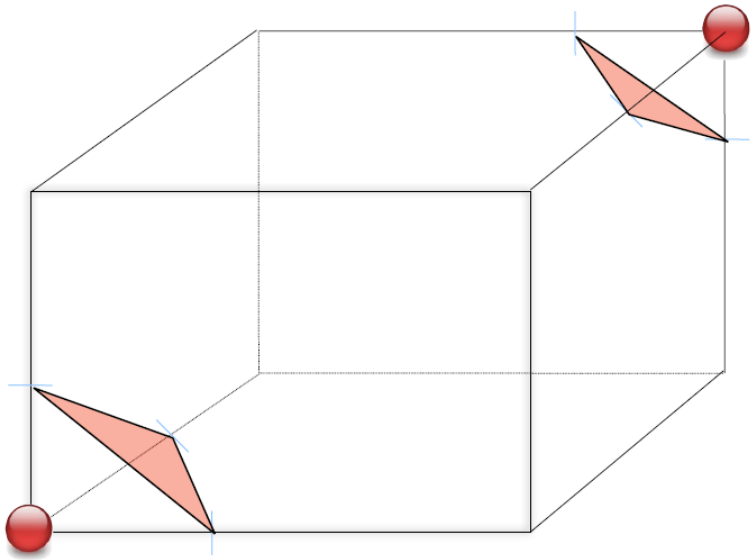
Place vertex at $(1-t)p + tq$ where $t = \frac{(f_0 - s_p)}{(s_q - s_p)}$

$t \in [0, 1]$



ILLINOIS

Generating a Polygon



Use linear interpolation to place the polygon vertices on the edges

Grid edge $[p, q]$ with $f(p) = s_p$ and $f(q) = s_q$ and isovalue f_0

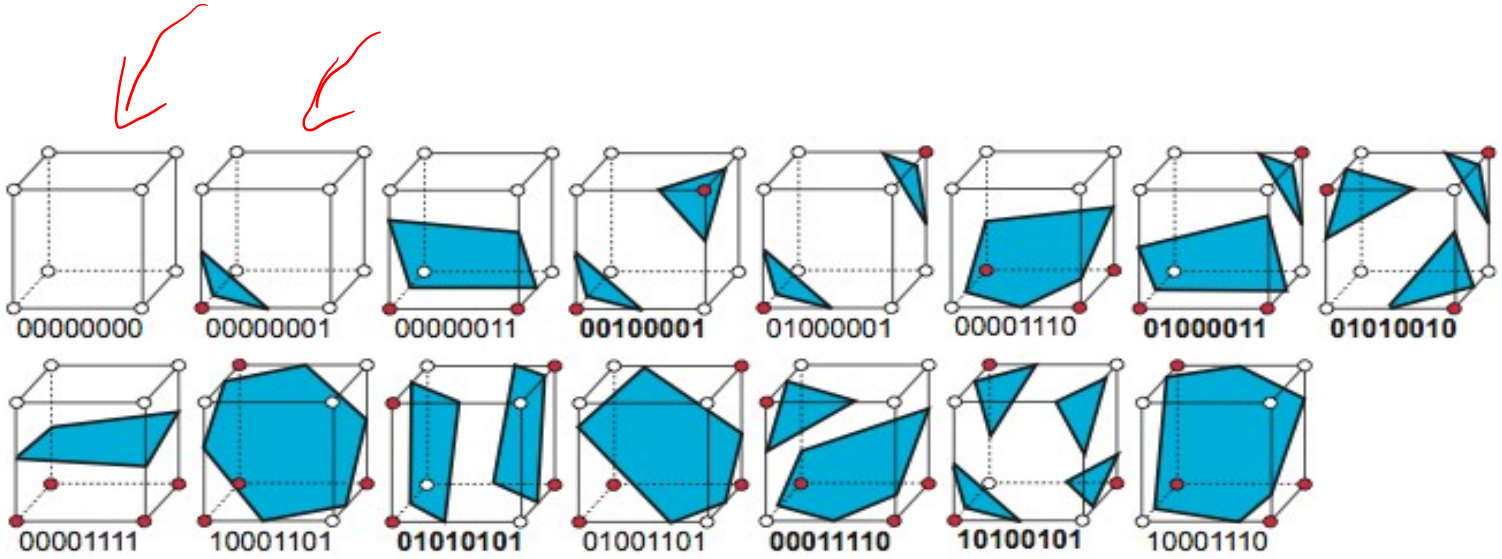
Place vertex at $(1 - t)p + q$ where $t = \frac{(t - s_p)}{(s_q - s_p)}$

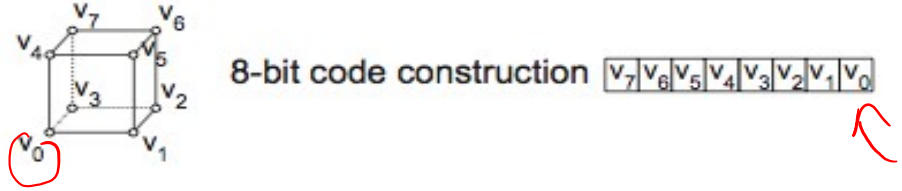
ILLINOIS

Marching Cubes: Cases

Encode inside/outside state of each vertex w.r.t. contour in an 8-bit code

256 cases (2^8=256)





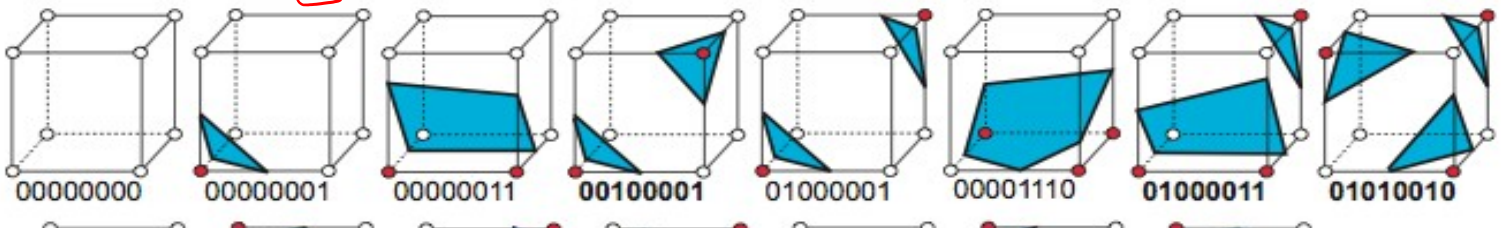
8-bit code construction $v_7 v_6 v_5 v_4 v_3 v_2 v_1 v_0$

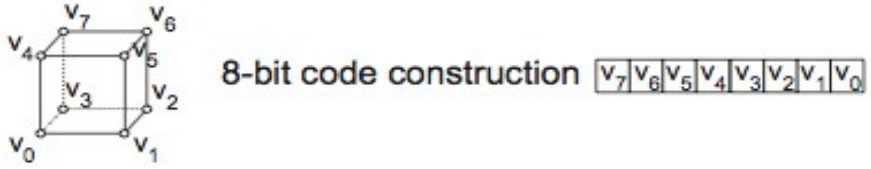
Marching Cubes: Cases

The 256 possible configurations can be grouped into these 15 cases

- complementarity (swapping positive and negative)
- rotational symmetry

Need to code 15 cases not 256 !



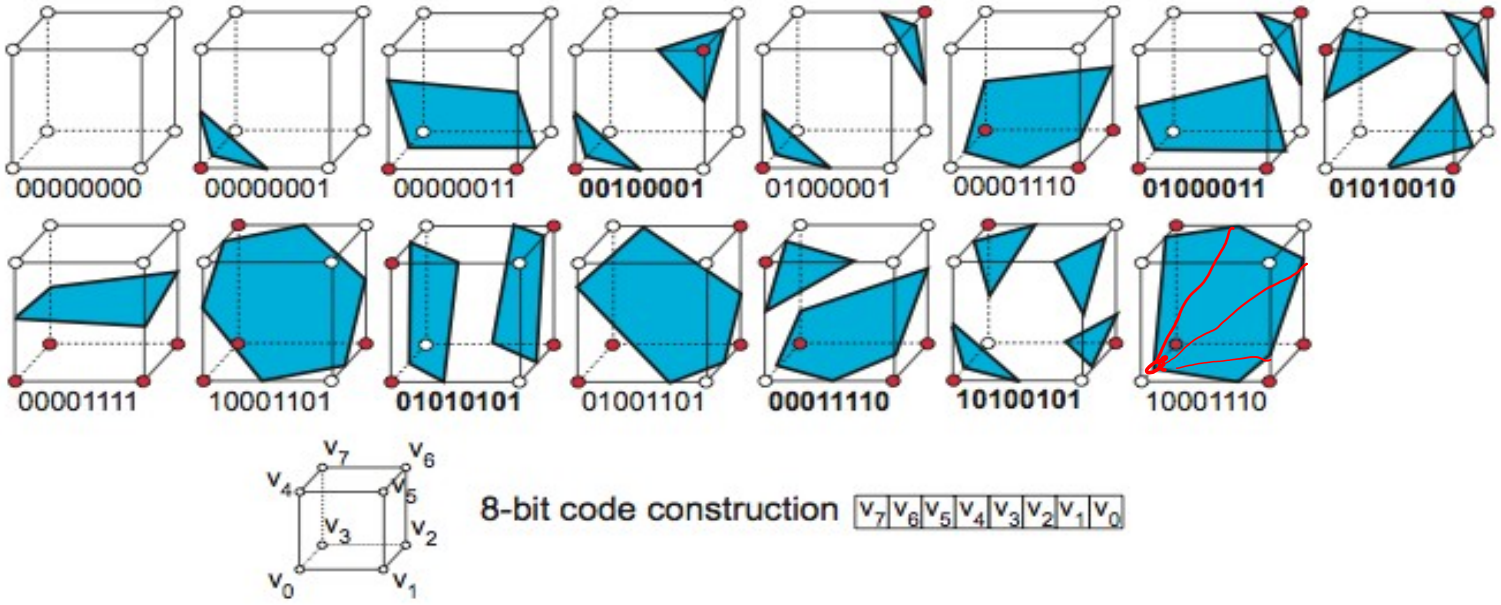


Marching Cubes: Cases

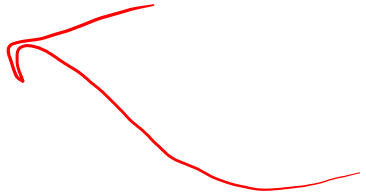
The 256 possible configurations can be grouped into these 15 cases

- complementarity (swapping positive and negative)
- rotational symmetry

Need to code 15 cases not 256 !



Non-triangle polygons can be triangulated



Marching Cubes Algorithm

Conceptually simple algorithm

1. Classify vertices of a cube and generate bitcode
2. Read isosurface lookup table using bitcode
3. Retrieve triangles
4. Compute vertex coordinates using linear interpolation
5. Store the triangles….

Process cubes in 2D sheets…marching by row and column within a sheet

Only need 2 sheets in memory at a time….

Tradeoffs

- Can be relatively memory efficient…scalable
  - only 2 sheets in memory at once
  - Write out triangles of a sheet to storage after neighbor sheet is done
- Parallelizable
- Relatively simple implementation

- Not adaptive…uses too many triangles
- Ringing artifacts
- Ambiguity