CSC 544
Data Visualization

Joshua Levine
josh@arizona.edu

Lecture 17
Isosurfaces

March 20, 2023

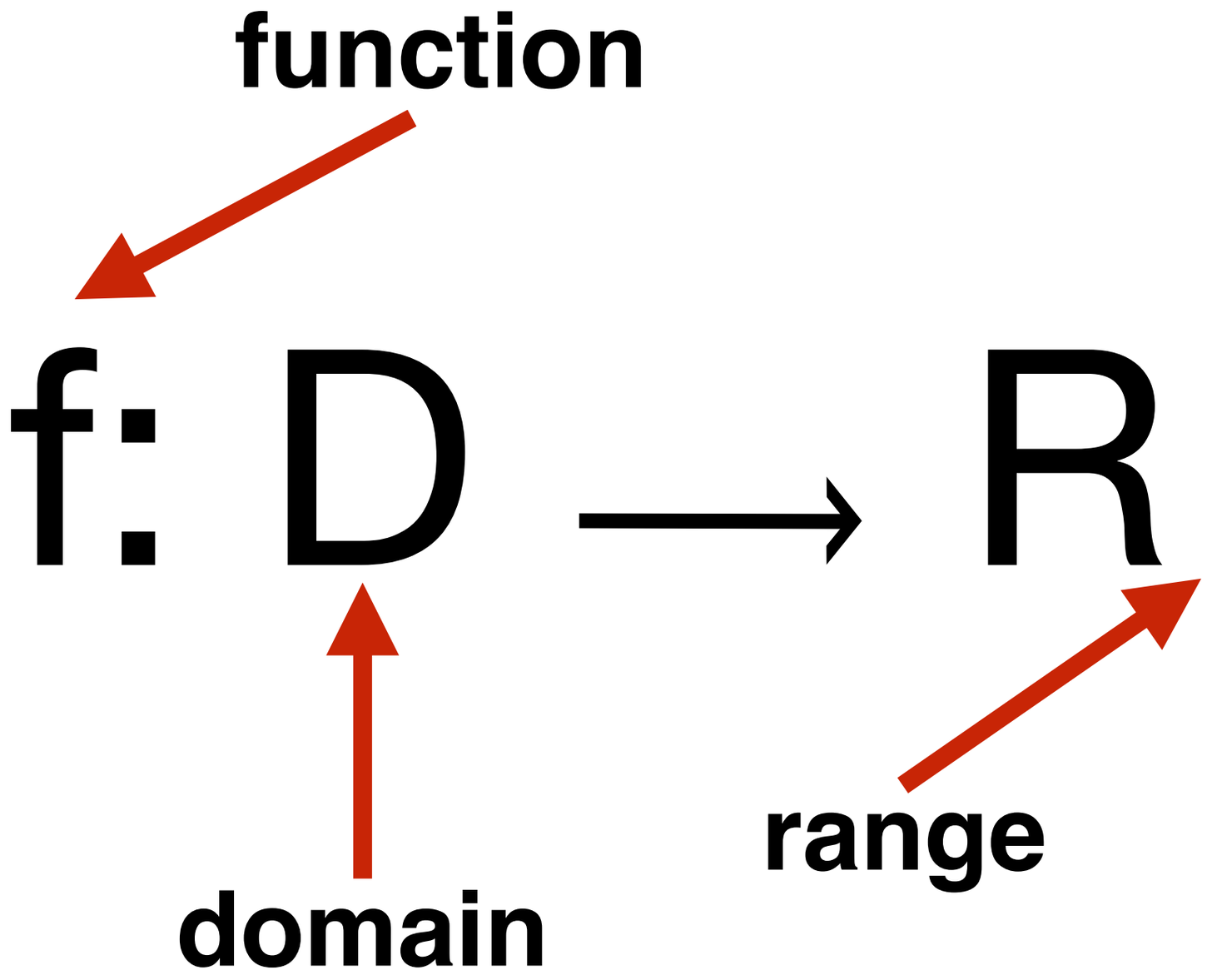
Today's Agenda

- Reminders:
  - A04 questions (due Mar. 27)? P02 (due Mar. 29)?

- Goals for today:
  - Discuss using isocontours to visualize scalar data in two and three dimensions

Visualizing Field Data

Recall: Field Data Models
Functions

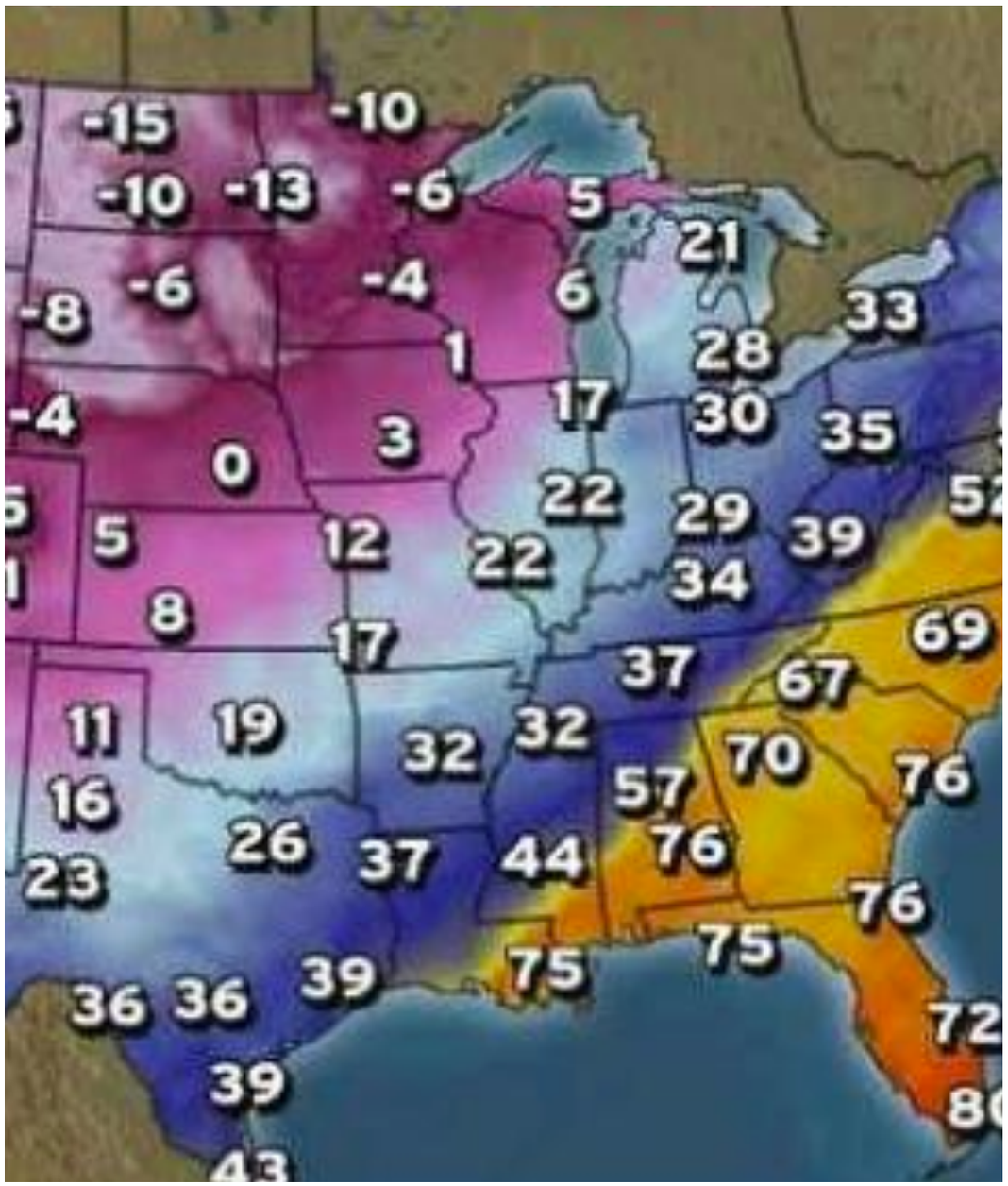


Recall: How Do We Represent Spatial Data?

- In the real world, there's infinitely many data points in a weather map.

- In a computer, we only have finite memory and finite time.

- How do we solve this problem? **<u>Interpolation</u>**

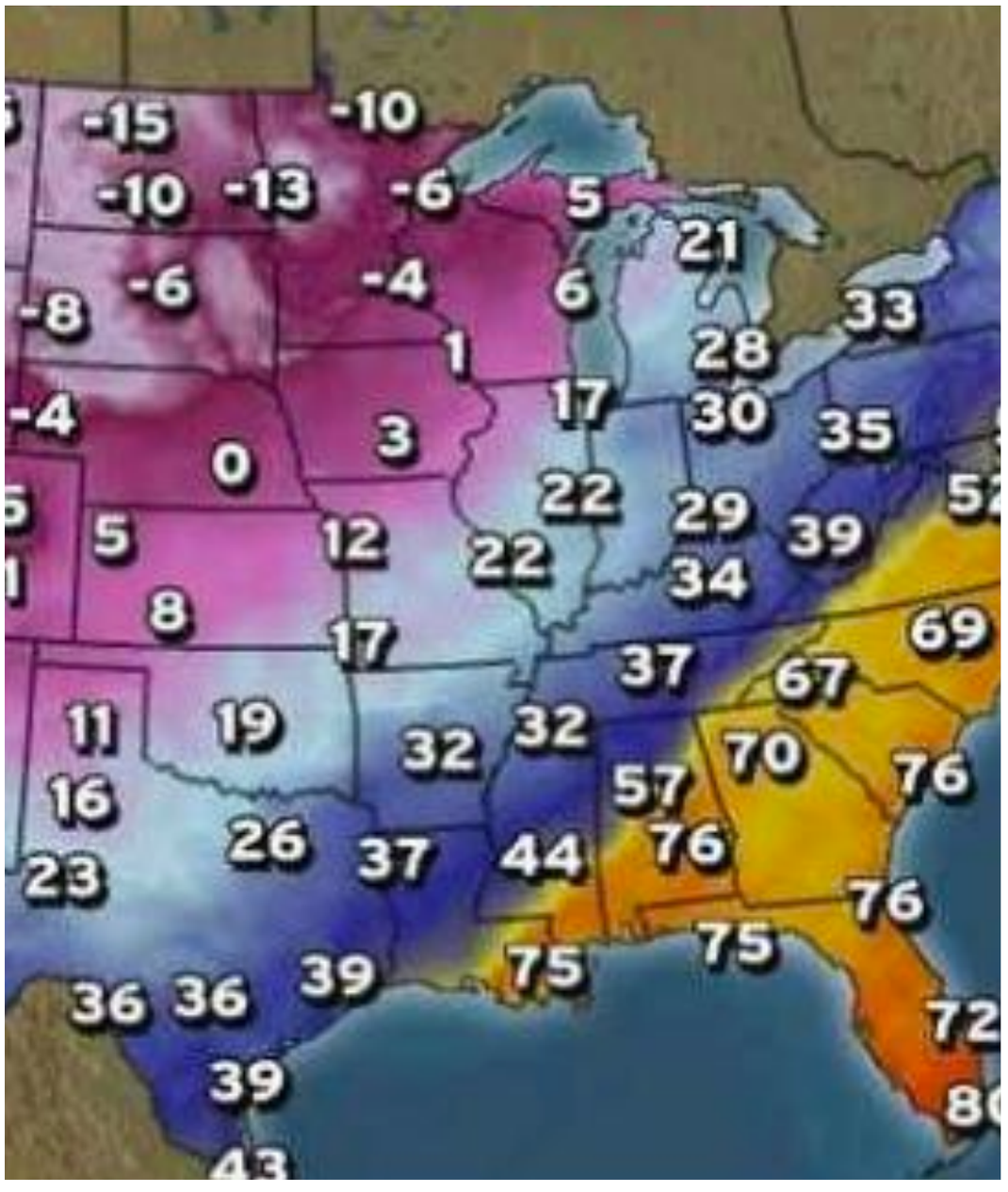


Techniques for 2D Scalar Field Visualization

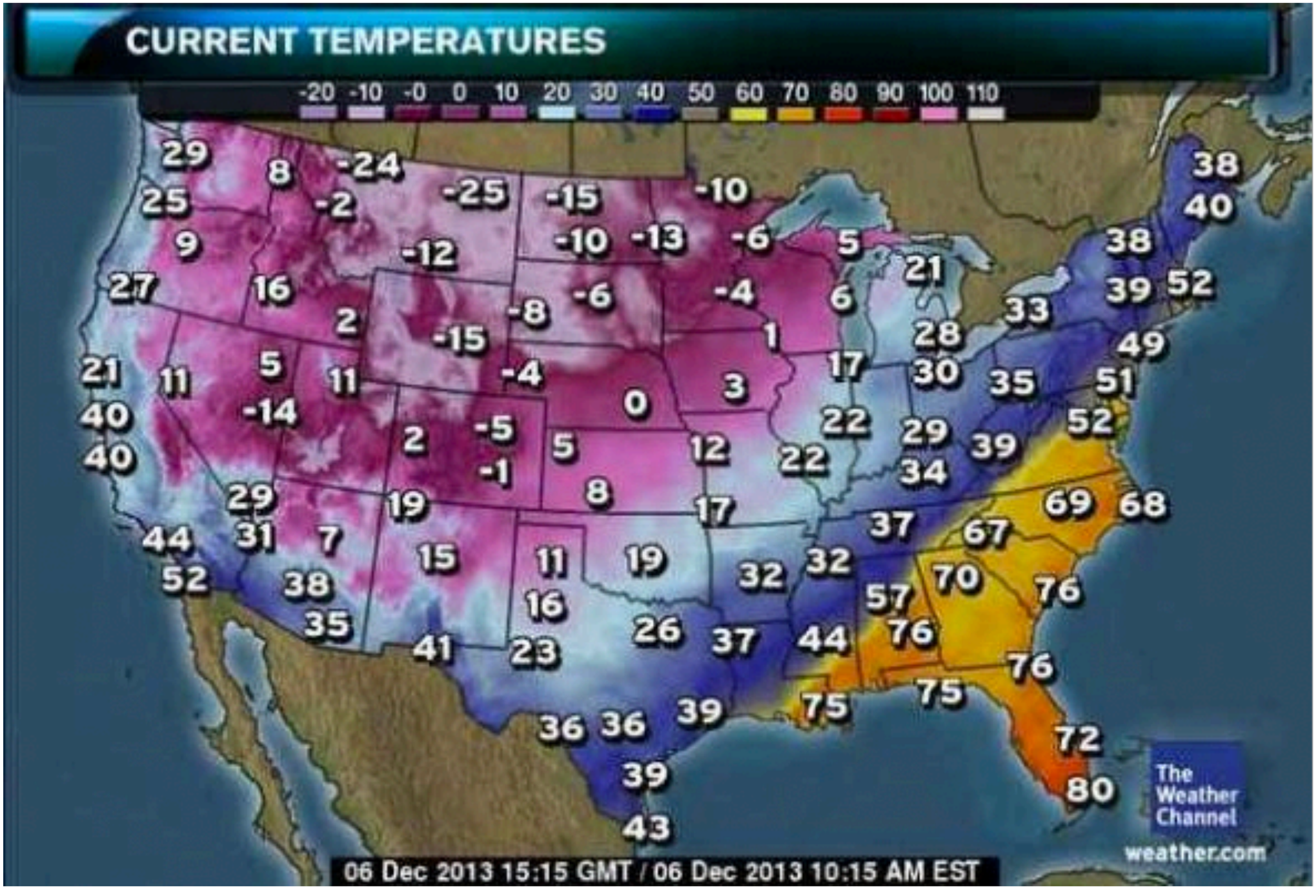
- Color-based
  - Colormapping / Transfer Function
- Geometry-based:
  - Height fields, surface plots
  - Contours

Colormapping

- “Default” strategy:
    - create a color scale using the **range** of the function as the **domain** of the scale
    - create a position scale to convert from the **domain** of the function to **positions** on the screen
- set the pixel color according to the scales



Colormapping Guidelines Still Apply!



More Examples

Penny Rheingans. Task-based Color Scale Design. SPIE 1999, pp. 35-43.

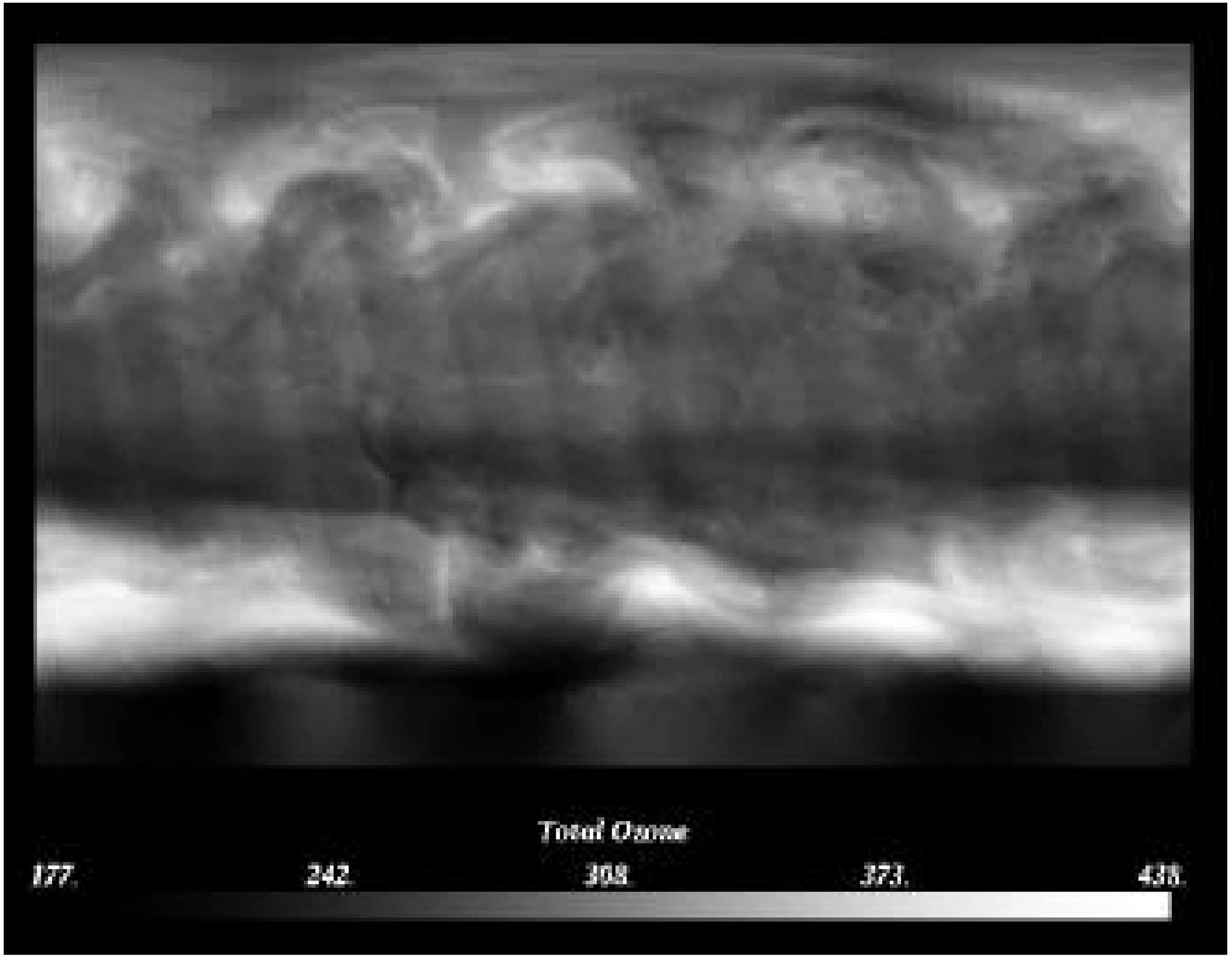


Figure 1. Grey scale.

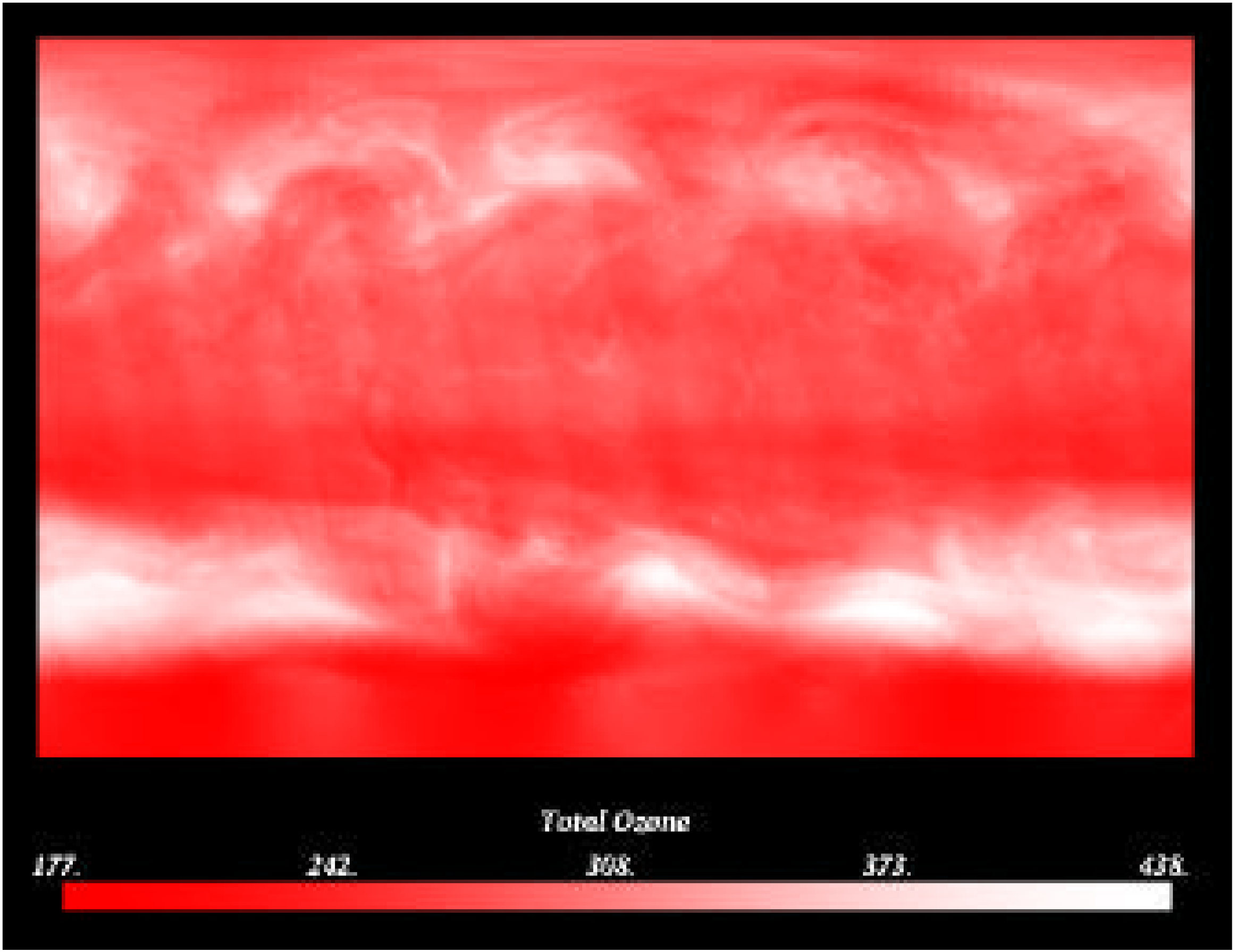


Figure 2. Saturation scale.

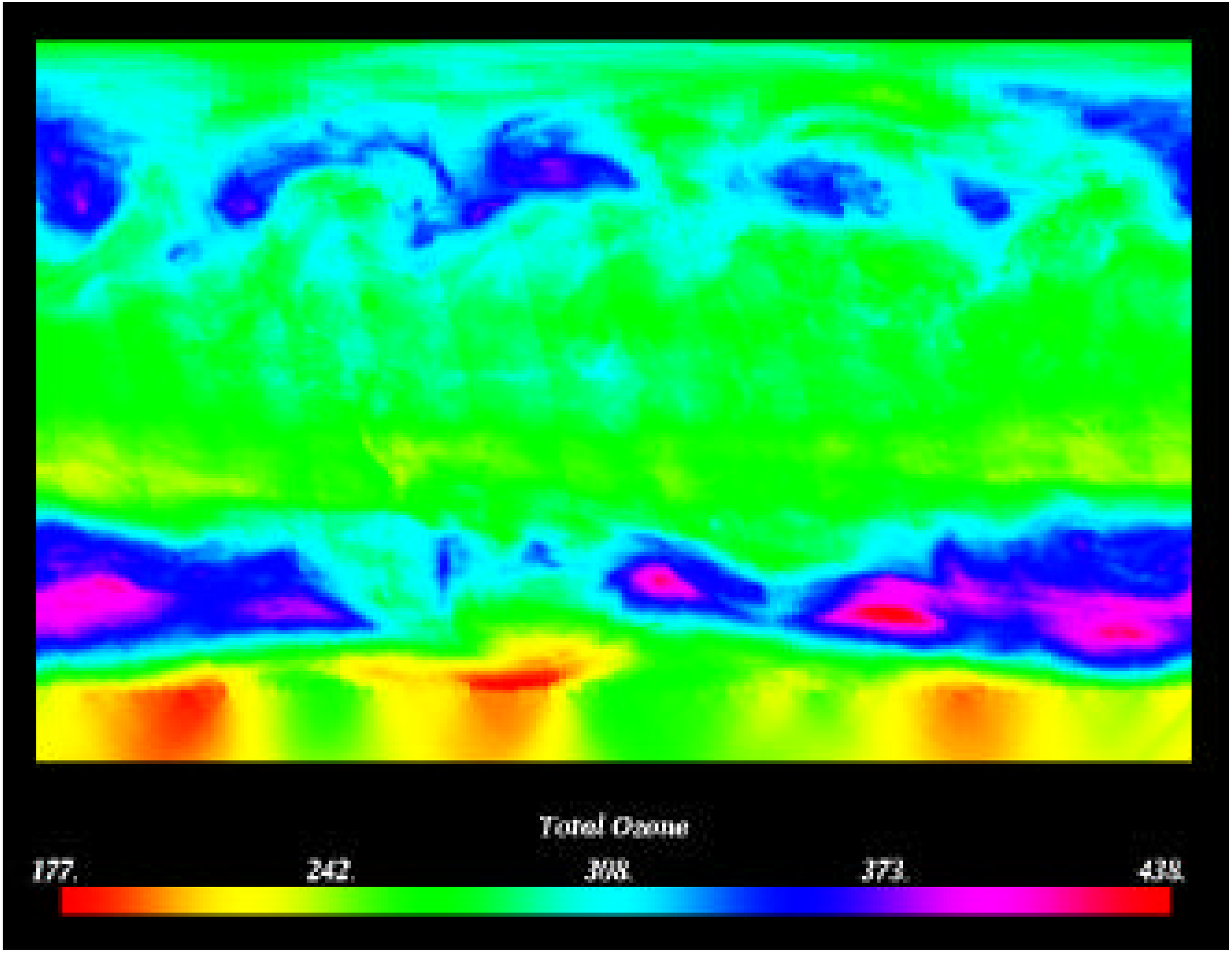


Figure 3. Spectrum scale.

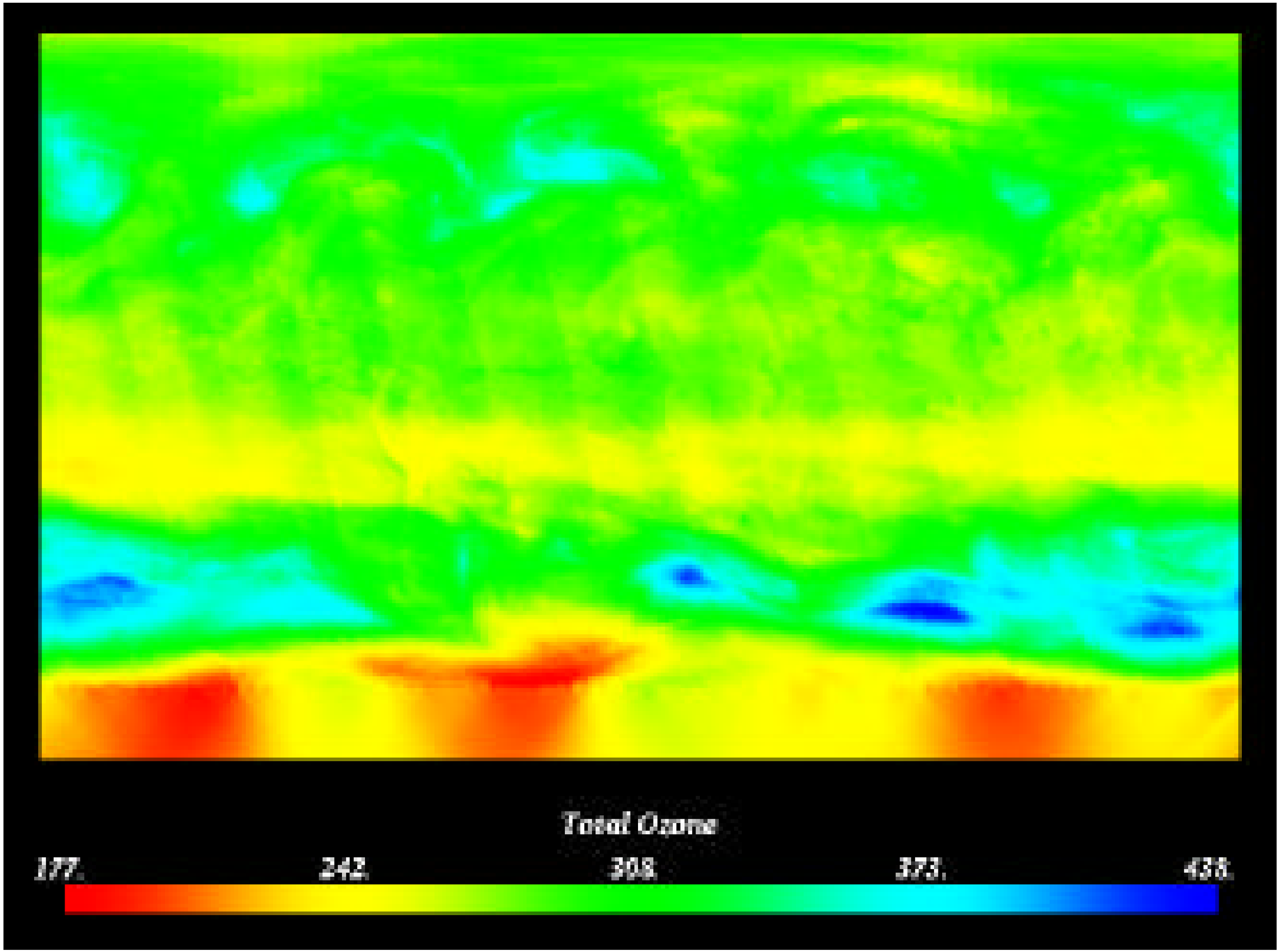


Figure 4. Limited spectrum scale.

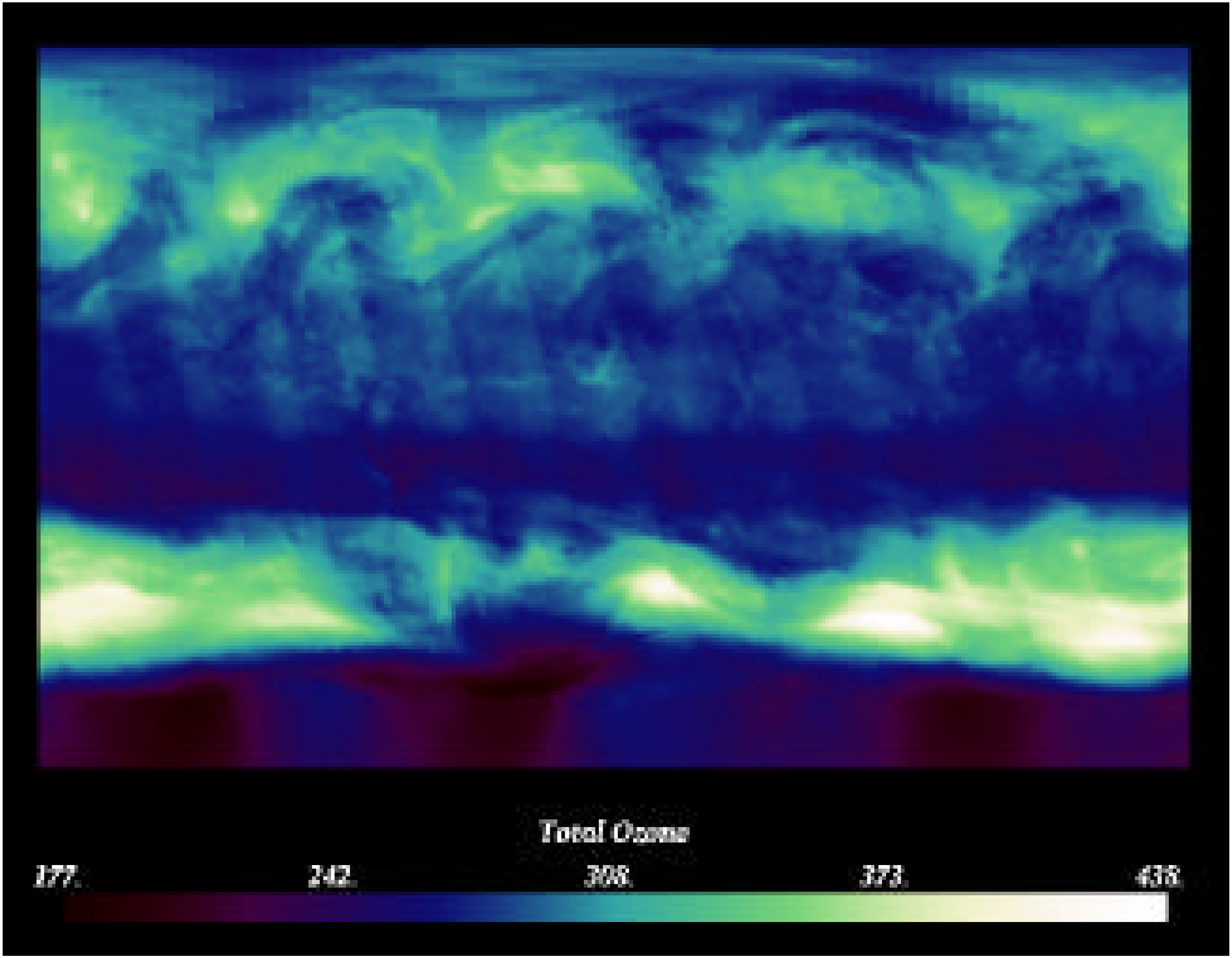


Figure 5. Redundant hue/lightness scale.

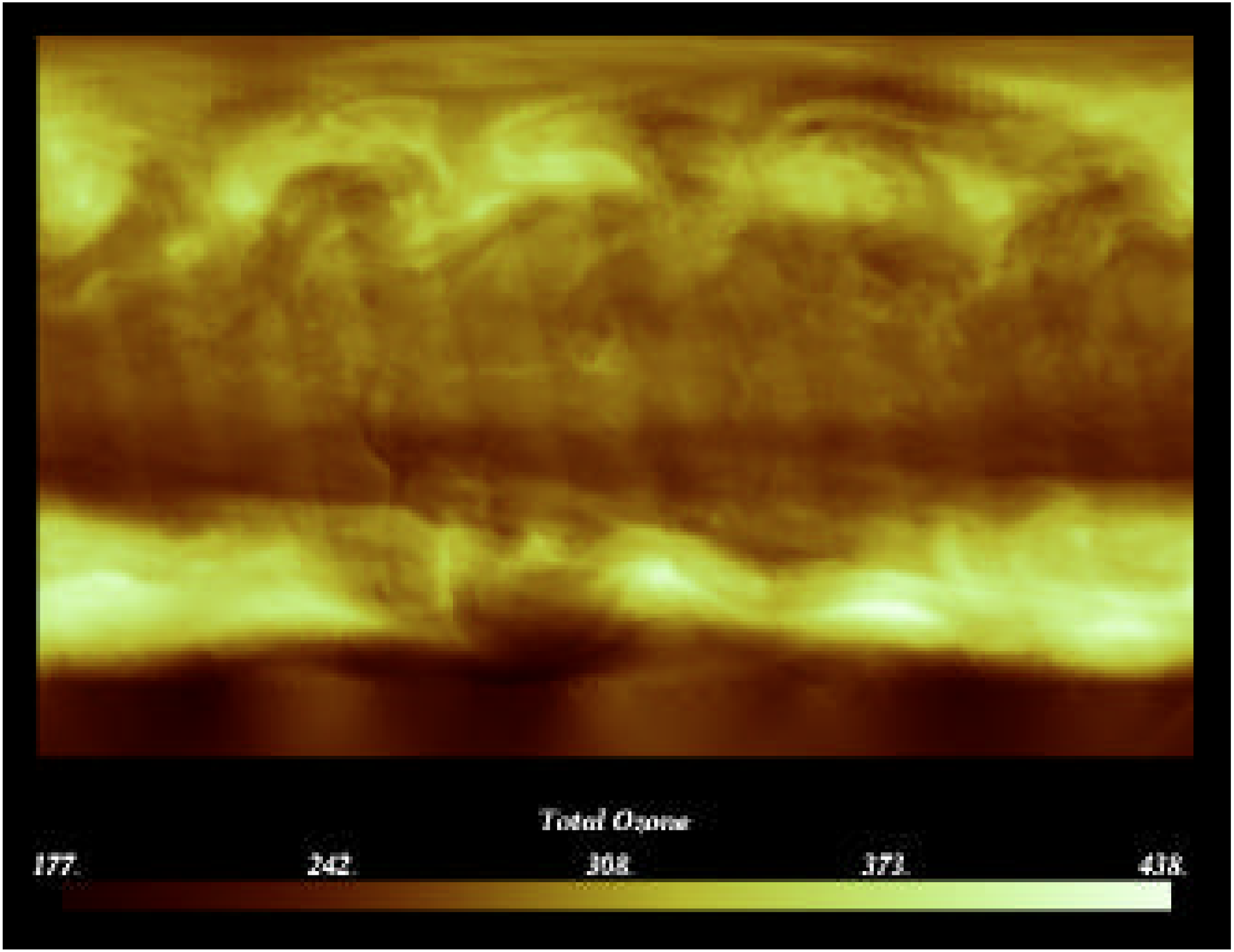


Figure 6. Heated-object scale.

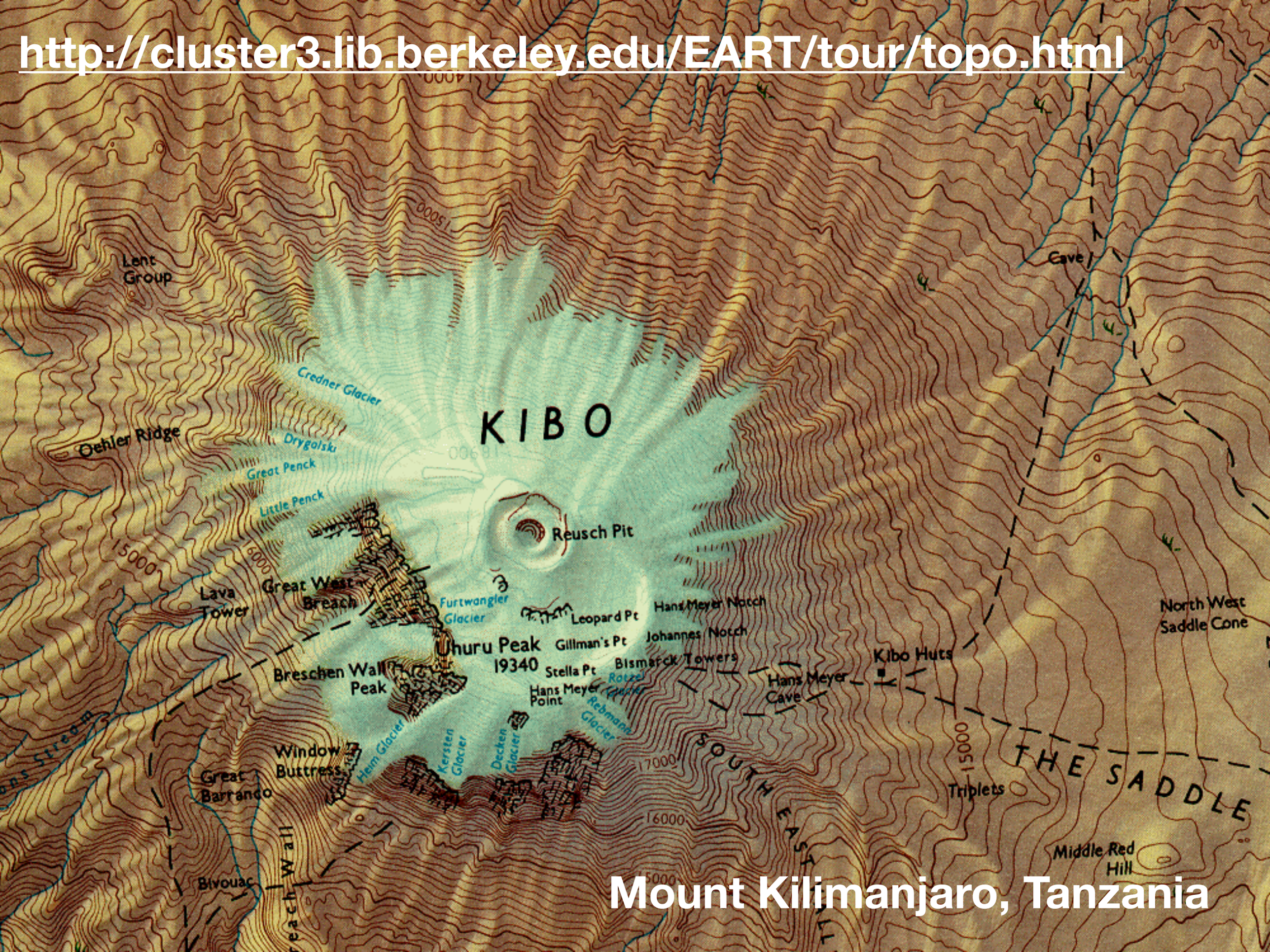
Height Fields

- We use height in 1D plots, let's use it in 2D plots
- Direct intuition of the topography
- Let the geometry convey the data

Contour Lines

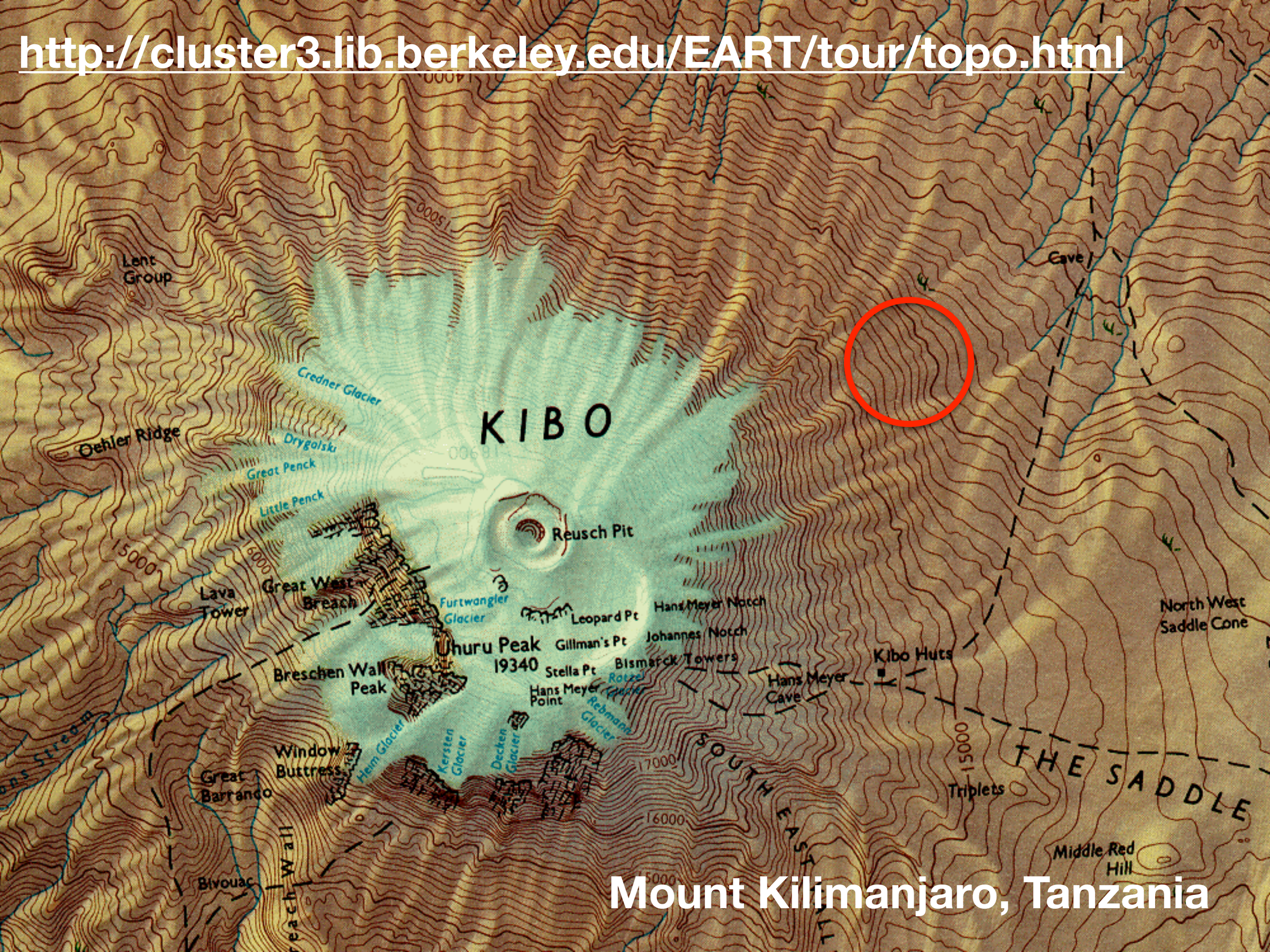
- Draw lines of constant value.
- These bound regions of contiguous hues
  - Loops or lines through end of the dataset
- Usually best to use multiple contours

http://cluster3.lib.berkeley.edu/EART/tour/topo.html



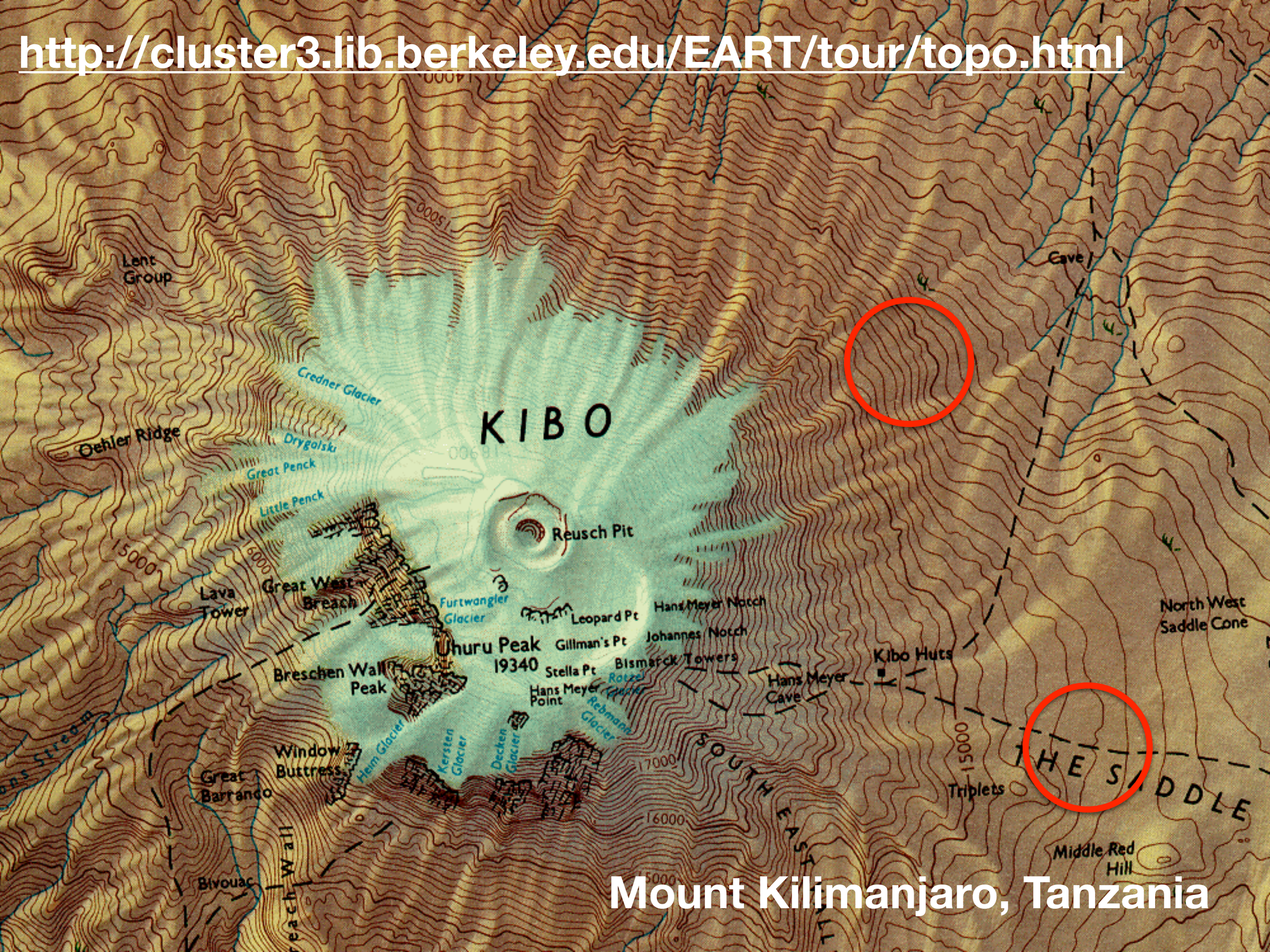
Mount Kilimanjaro, Tanzania

http://cluster3.lib.berkeley.edu/EART/tour/topo.html



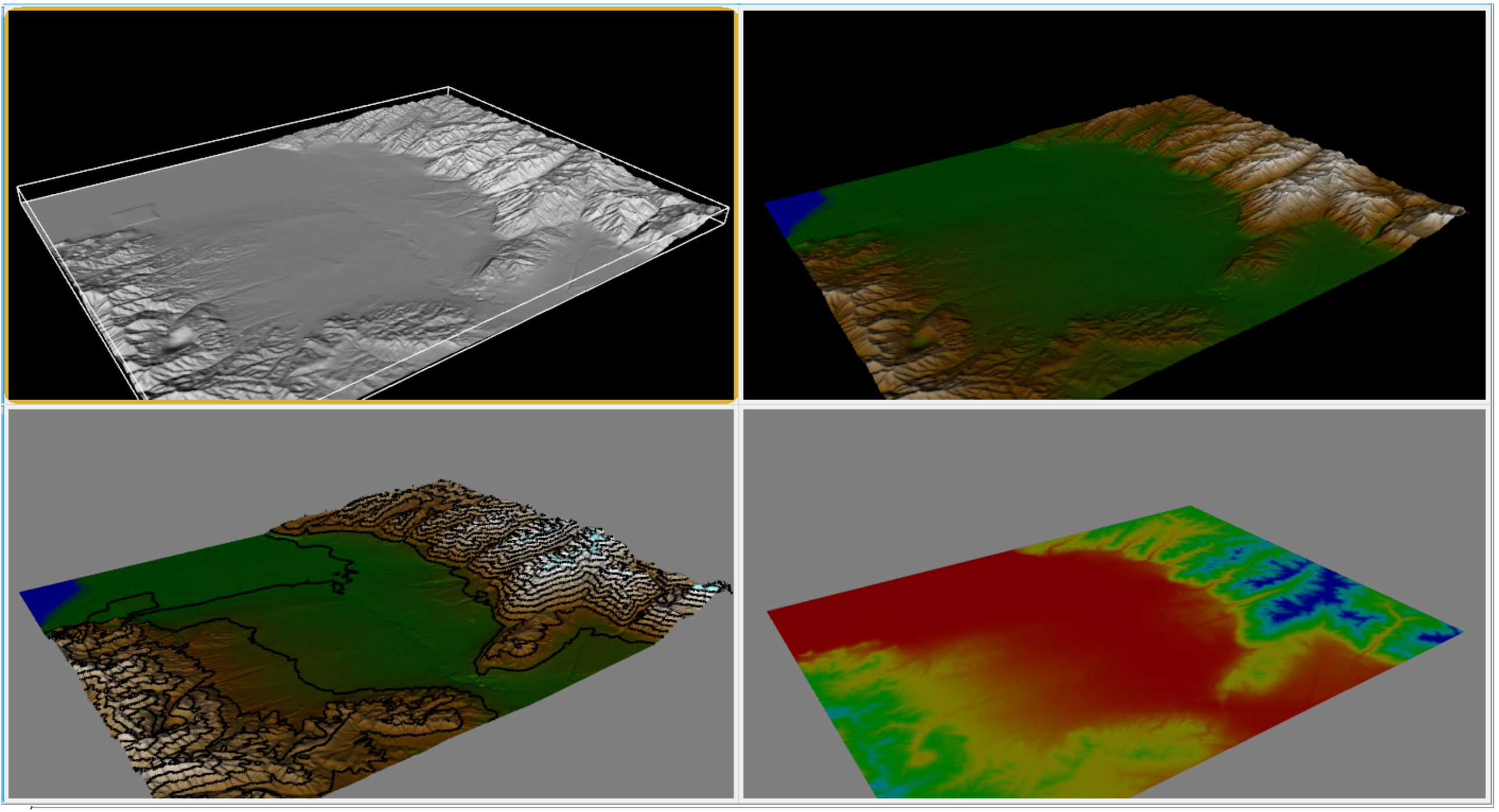
Mount Kilimanjaro, Tanzania

http://cluster3.lib.berkeley.edu/EART/tour/topo.html



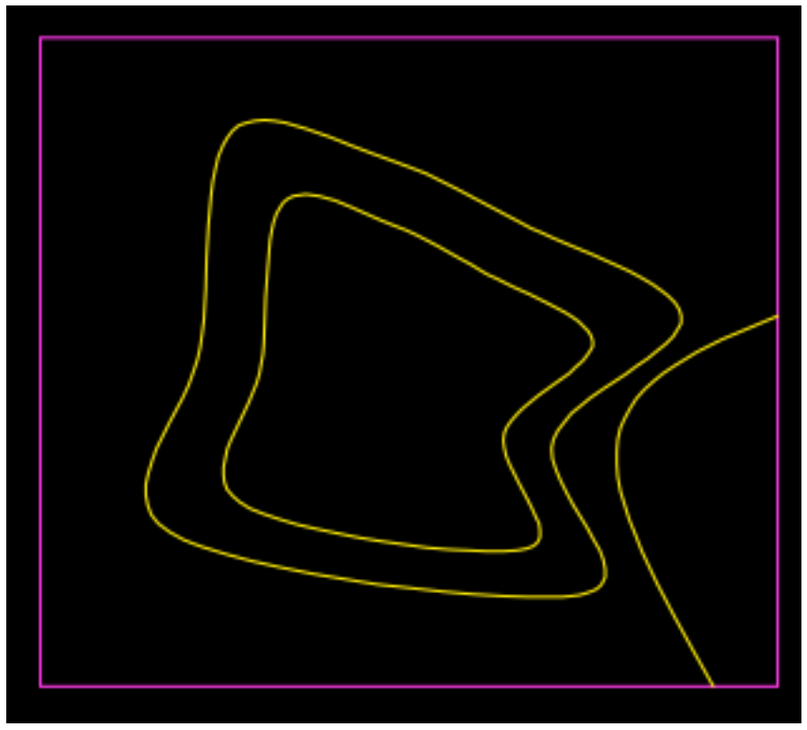
Mount Kilimanjaro, Tanzania

Compare



Computing Contours in 2D

Properties



- **Preimage** of a single scalar value v:
  - { (x,y) | (x,y) = $f^{-1}(v)$ }
  - Concept generalizes to any dimension

- **Closed** except at boundaries

- **Nested**: isocontours of different isovalues do not cross
  - Can consider the zero-set case (generalizes)
  - f(x,y) = v ⇔ f(x,y) - v = 0

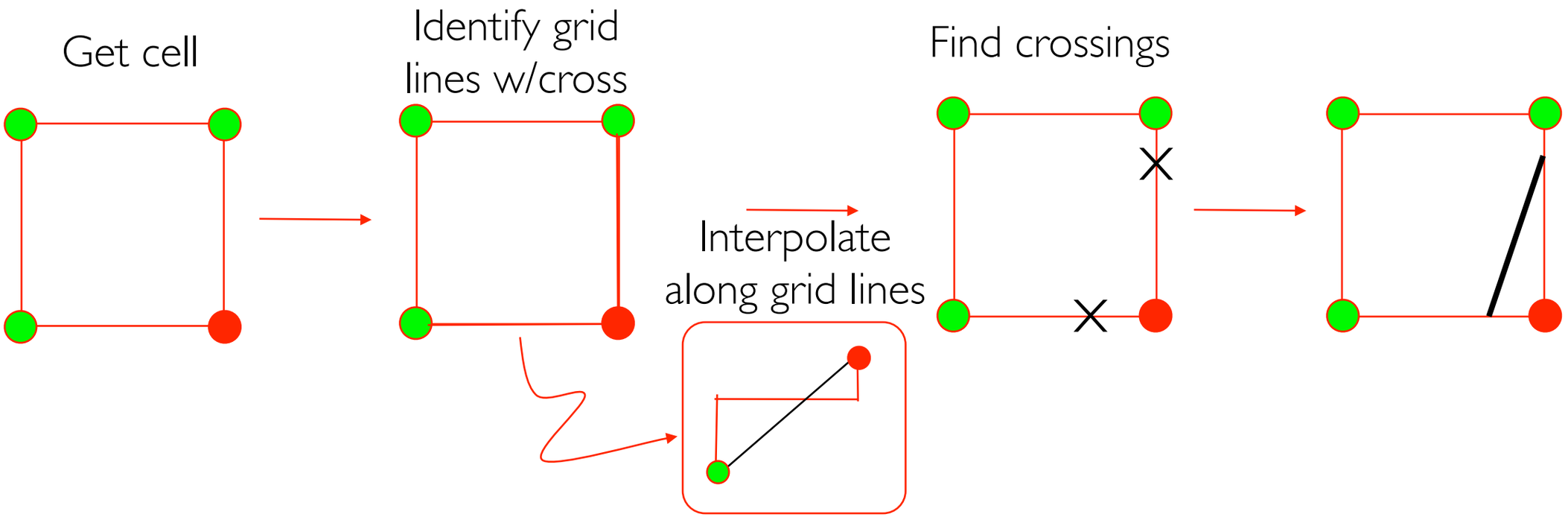
- Normals given by gradient vector of f()

Approach to Contouring in 2D

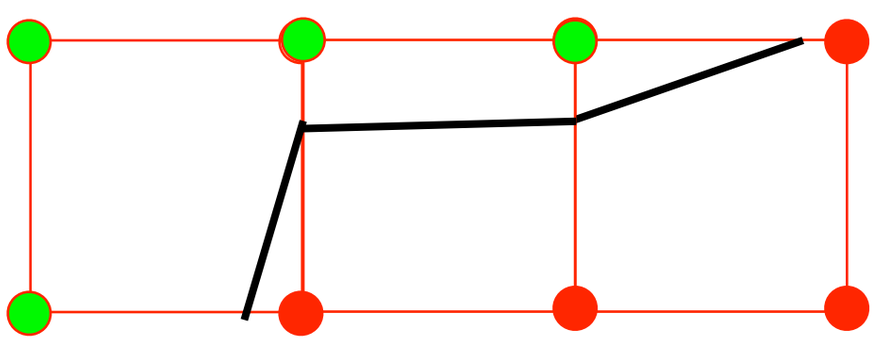
- **Idea**: Assign geometric primitives to individual cells
    - Will use line segments
- **Method**: Consider the “sign” of the values at vertices relative to if they are above or below the isovalue
    - Intersections MUST occur on **edges with sign change**
- Determine exact position of intersection by interpolate along grid edges

Approach to Contouring in 2D

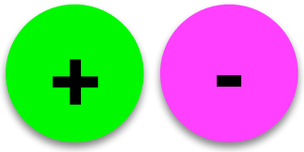
- Contour must cross every grid line connecting two grid points of opposite sign

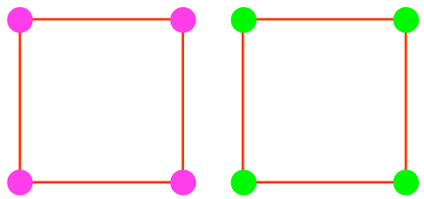
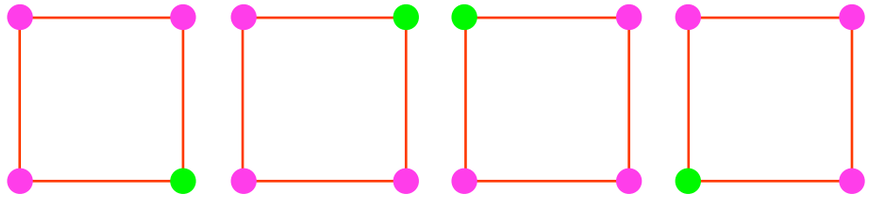
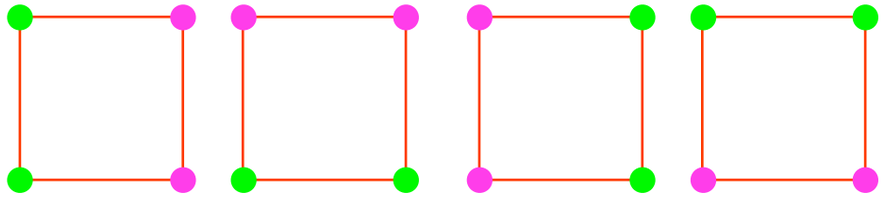
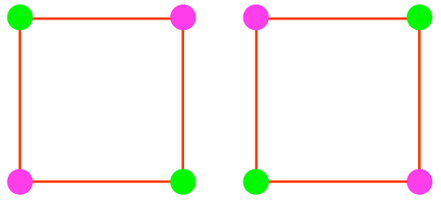


Primitives naturally chain together

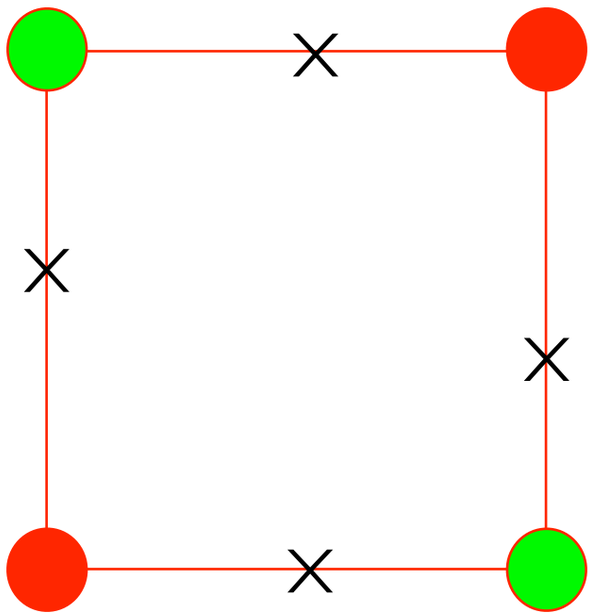


Cases

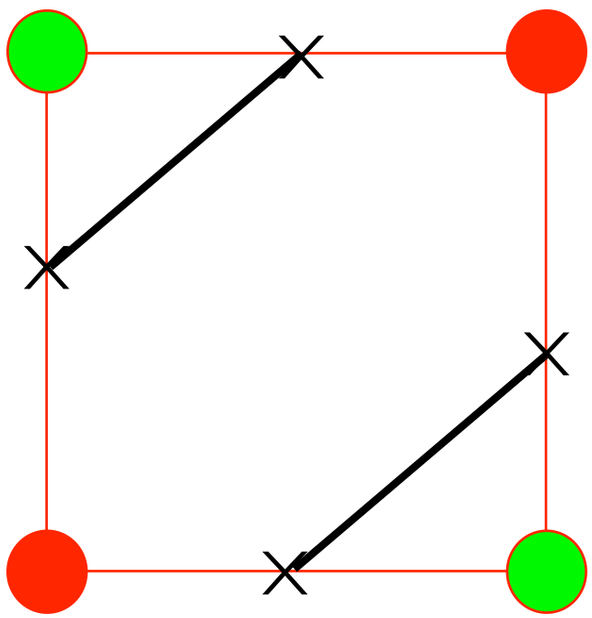


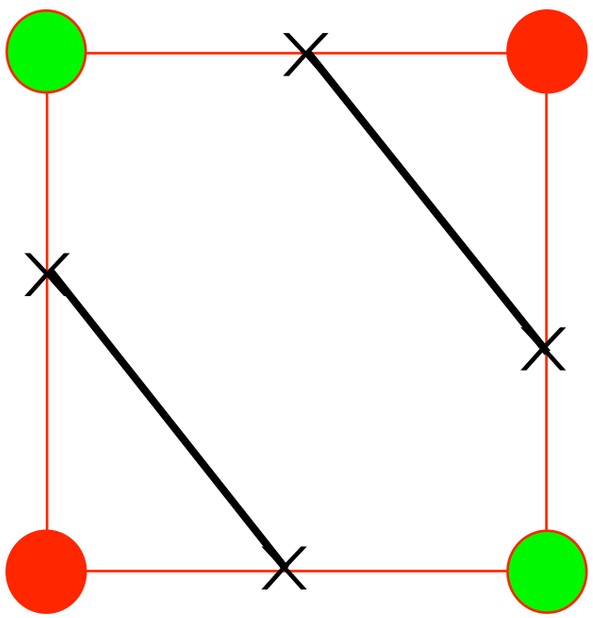
Case	Polarity	Rotation	Total	
No Crossings	x2		2	
Singlet	x2	x4	8	 (x2 for polarity)
Double adjacent	x2	x2 (4)	4	
Double Opposite	x2	x1 (2)	2	
			$16 = 2^4$	

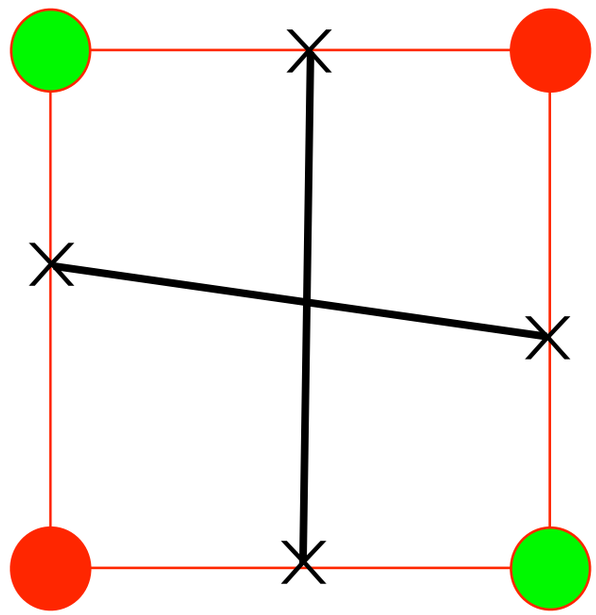
Ambiguities

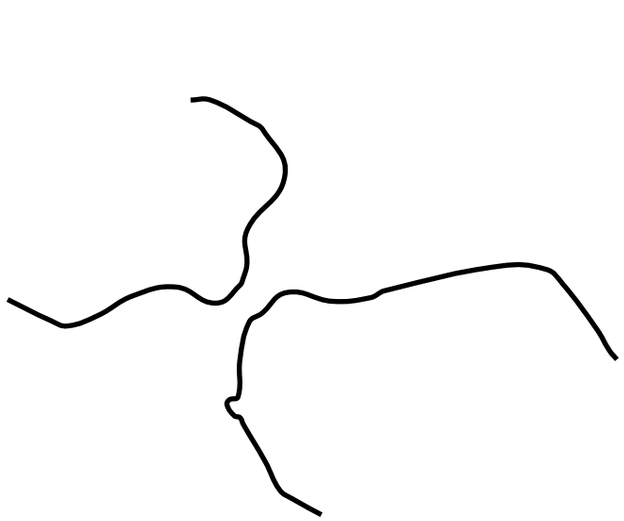


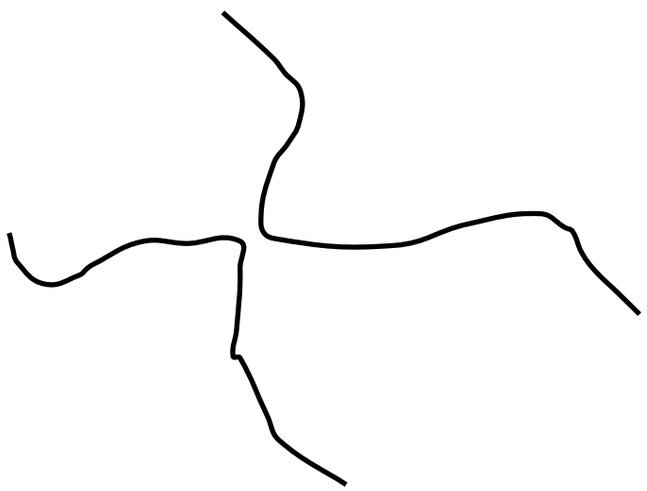
Ambiguities

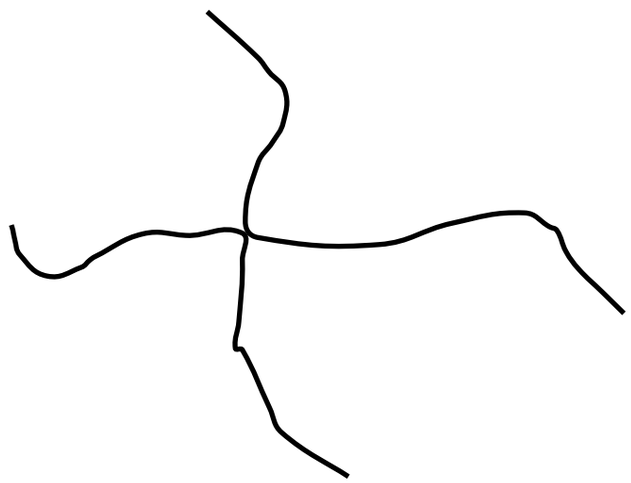












The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes

Gregory M. Nielson Bernd Hamann

Computer Science
Arizona State University
Tempe, AZ 85287-5406

Abstract

A method for computing isovalue or contour surfaces of a trivariate function is discussed. The input data are values of the trivariate function, F_{ijk}, at the cuberille grid points (x_i, y_j, z_k) and the output is a collection of triangles representing the surface consisting of all points where $F(x, y, z)$ is a constant value. The method described here is a modification that is intended to correct a problem with a previous method.

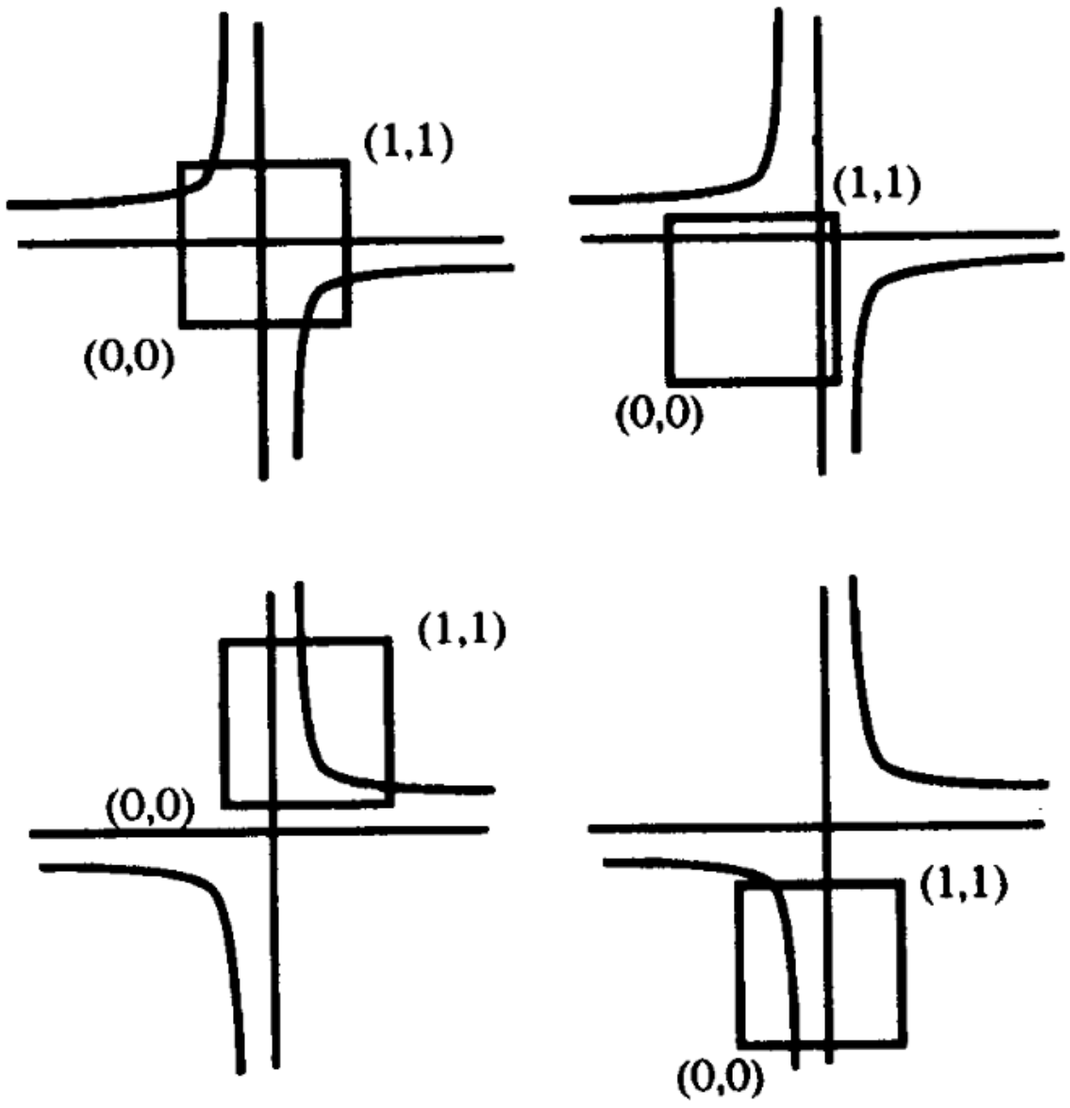
1.0 Introduction

The purpose of this paper is to describe a method for computing contour or isovalue surfaces of a trivariate function F(x, y, z). It is assumed that the function is continuous and that samples over a cuberille grid (see Figure 1) are available. These values are denoted by $F_{ijk} = F(x_i, y_j, z_k)$; i = 1, ... , Nx, j = 1, ... , Ny, k = 1, ... , Nz. The problem is to compute the isovalue or contour surface

$$S_\alpha = \{ (x, y , z) : F(x, y, z) = \alpha\}.$$

marked indicates $F_{ijk} > \alpha$. While there are $2^8 = 256$ possible configurations, there are only 15 shown in Figure 2. This is because some configurations are equivalent with respect to certain operations. First off, the number can be reduced to 128 by assuming two configurations are equivalent if marked grid points and unmarked grid points are switched. This means that we only have to consider cases where there are four or fewer marked grid points. Further reduction to the 15 cases shown is possible by equivalence due to rotations.

Ambiguities Occur when the Bilinear Interpolant Has Both Hyperbolic Arcs in the Grid Cell



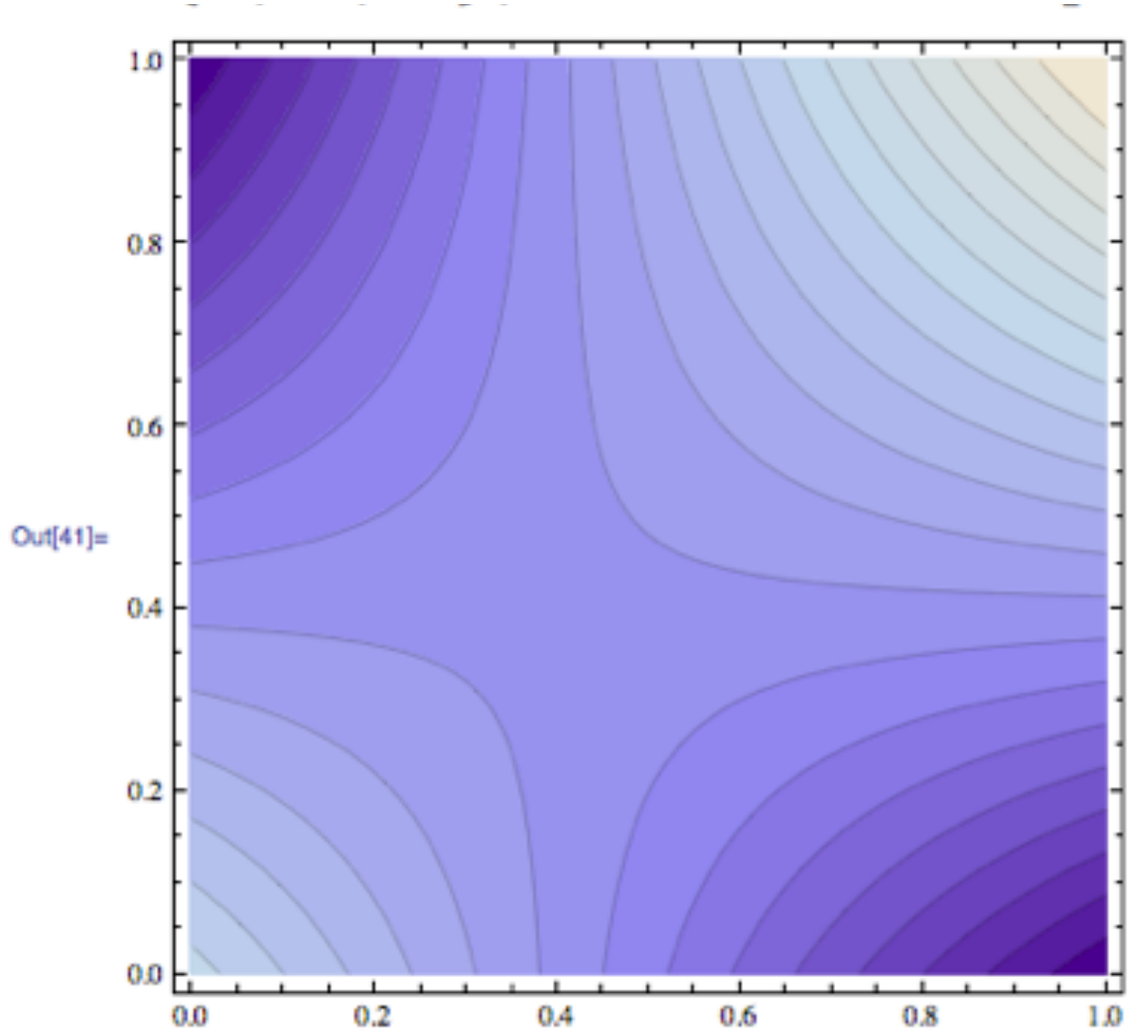
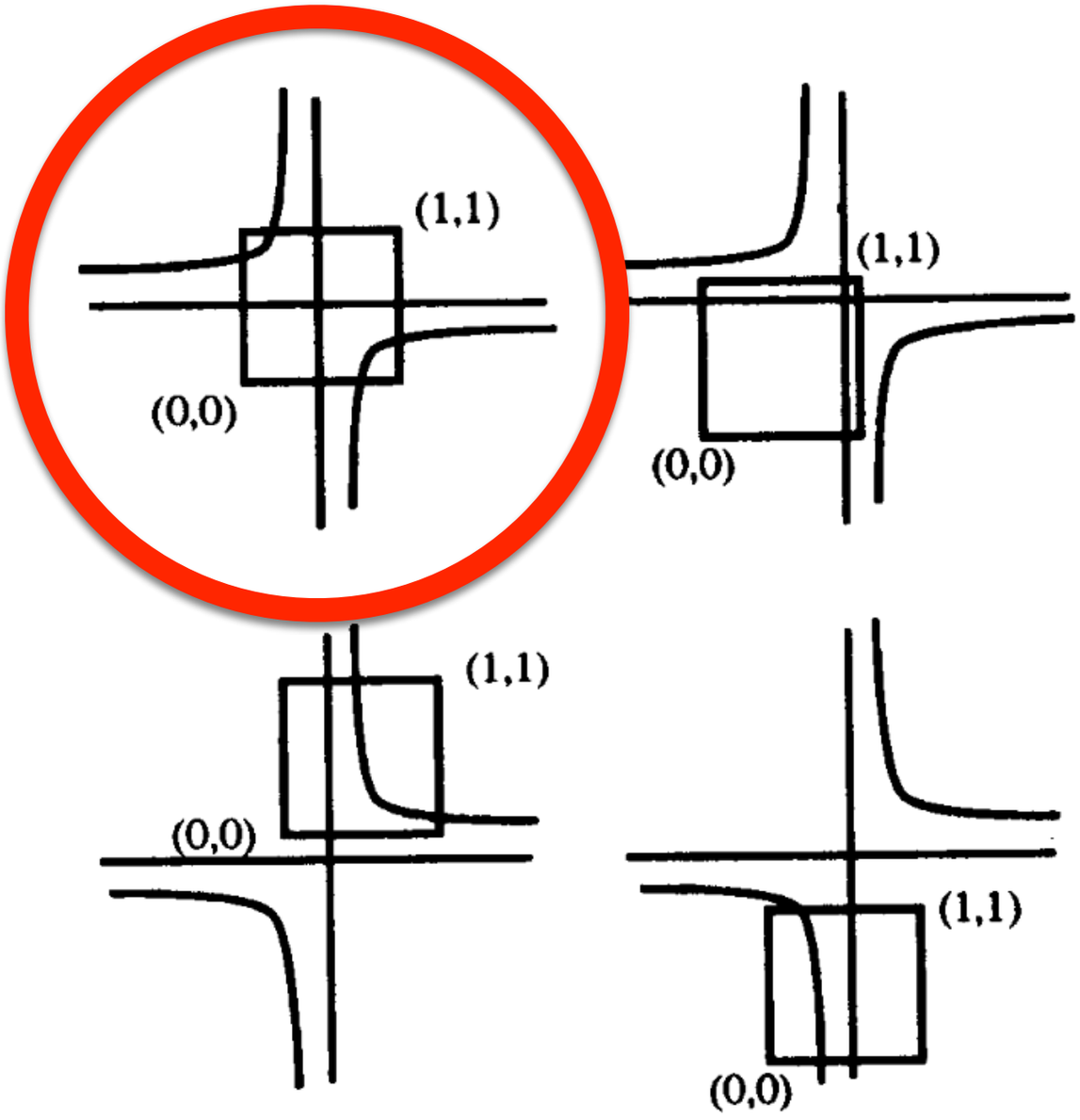


Figure 6. Contours of bilinear interpolant

Ambiguities Occur when the Bilinear Interpolant Has Both Hyperbolic Arcs in the Grid Cell



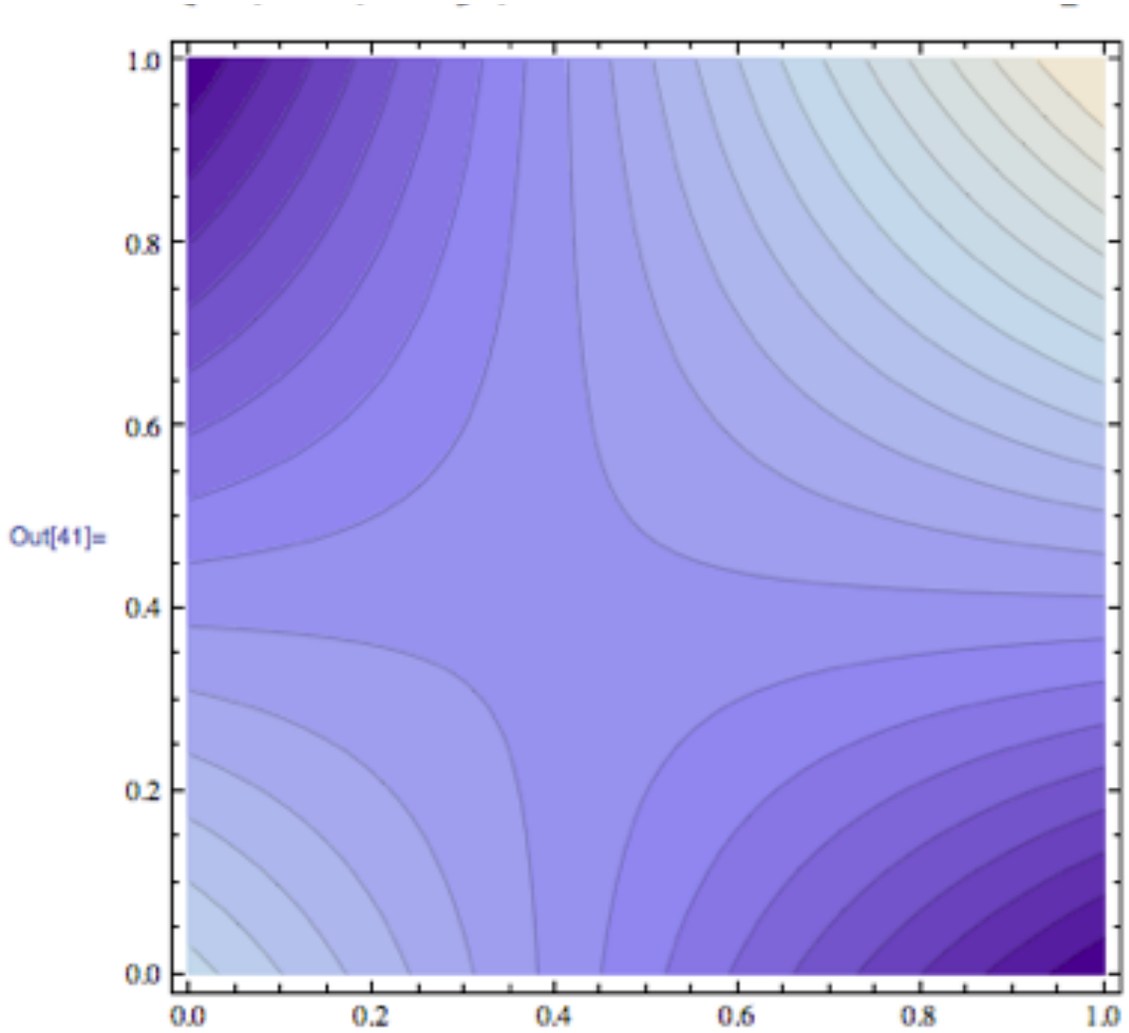
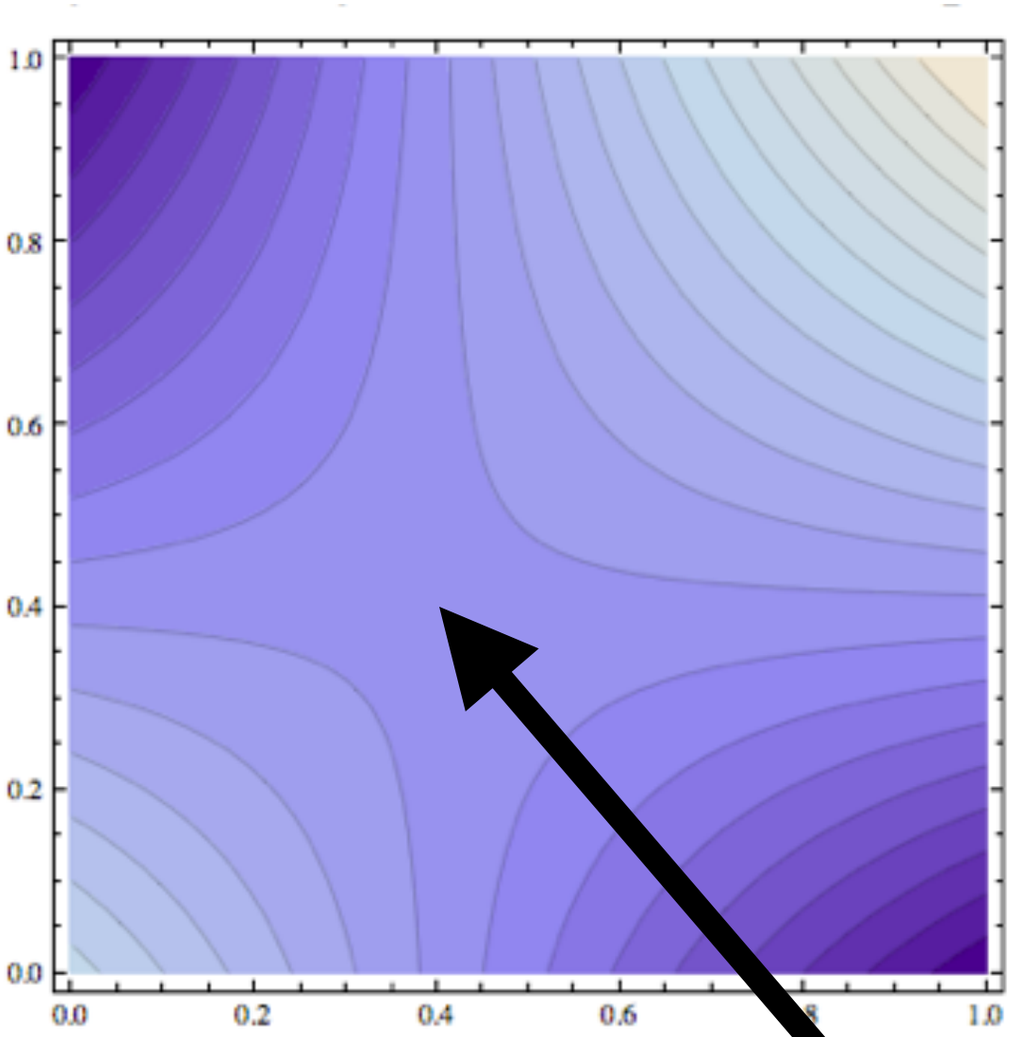


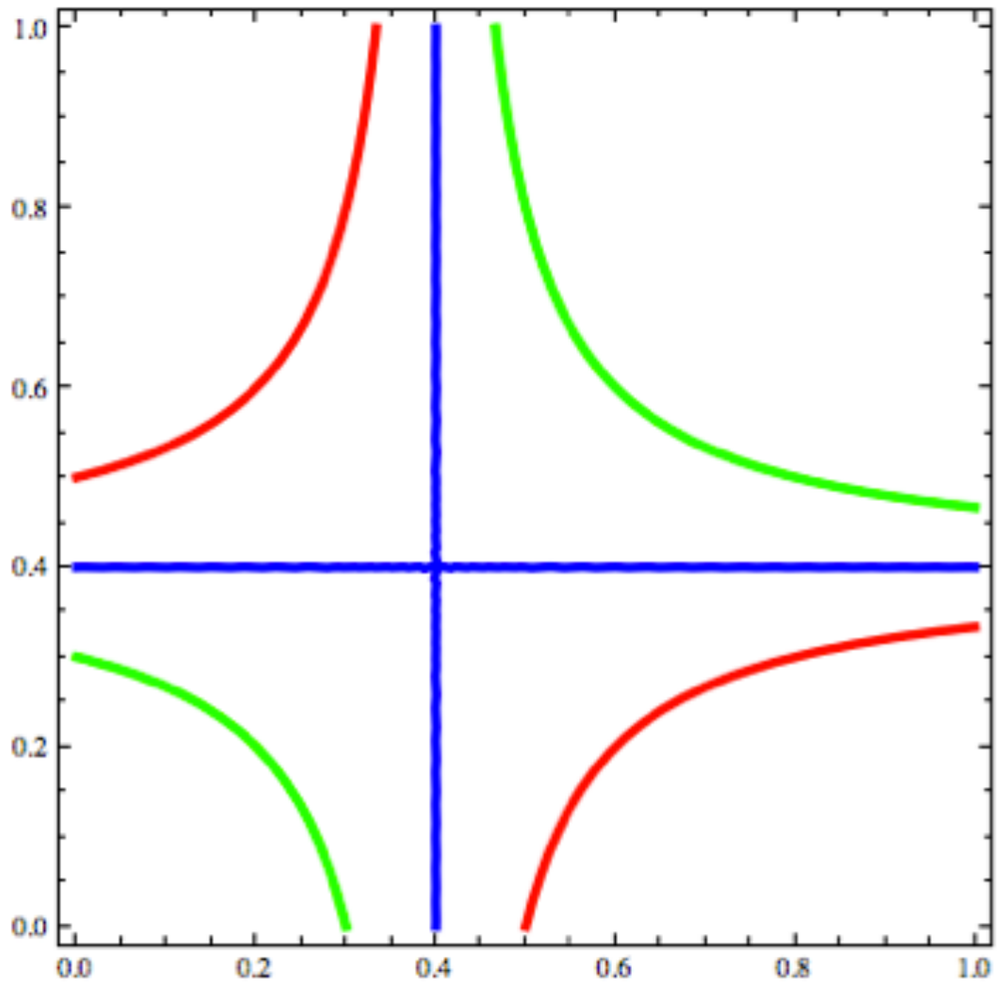
Figure 6. Contours of bilinear interpolant

By Comparing Isovalue With the Value At Intersection of Asymptotes, Can Determine Which Pair of Contours Should be Connected

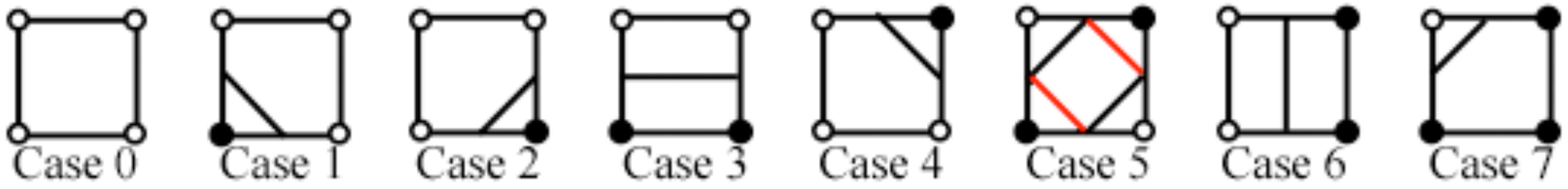
Out[41]=

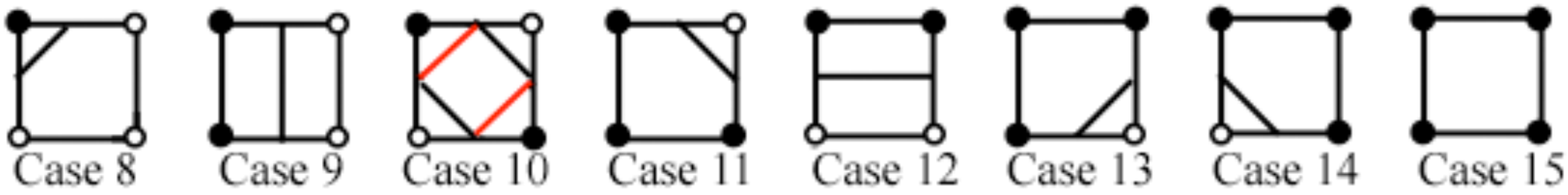


Out[40]=

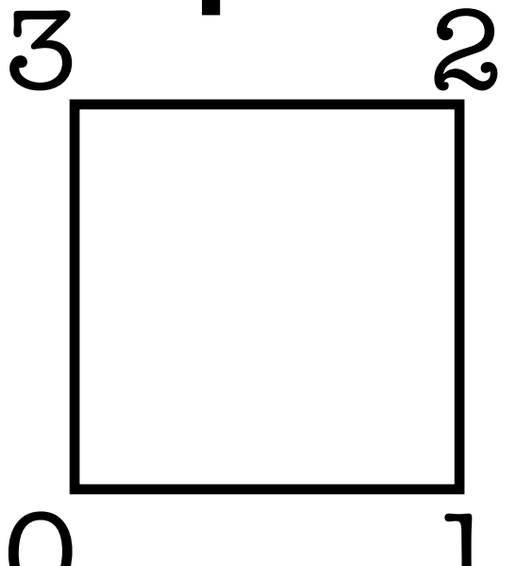


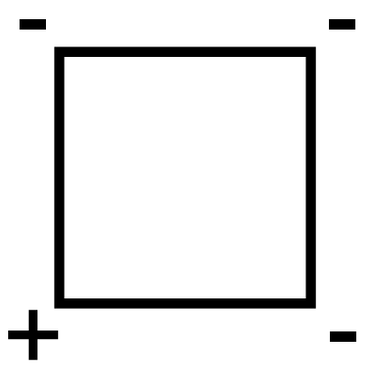
A Case Table Can Be Used To Implement The Algorithm



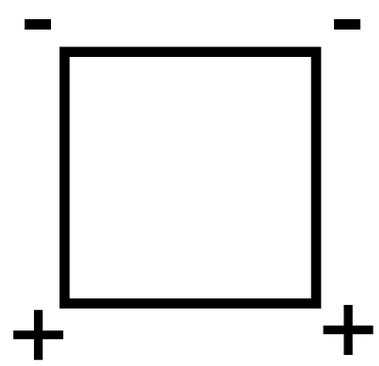


Accelerating Marching Squares With The Case Table

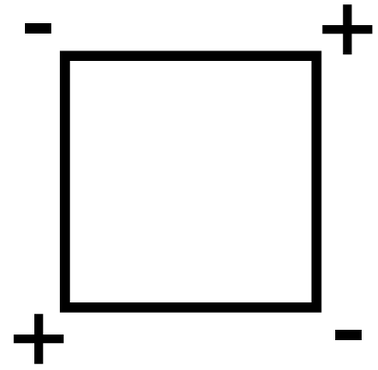




1000



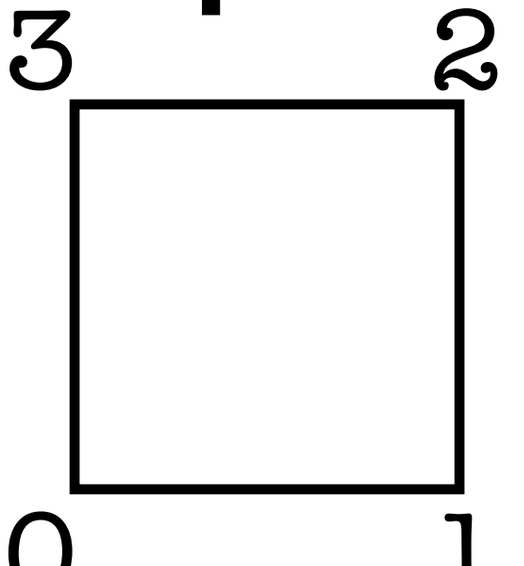
1100

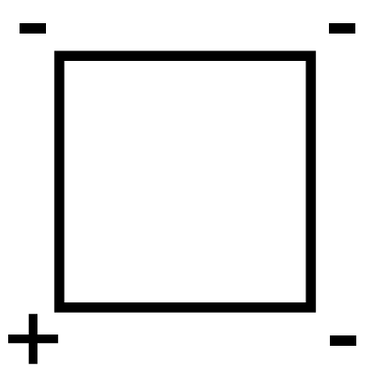


1010

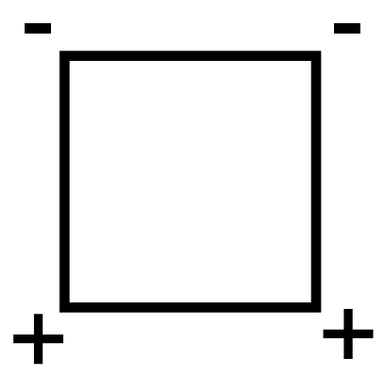
$\overline{0}$ $\overline{1}$ $\overline{2}$ $\overline{3}$

Accelerating Marching Squares With The Case Table

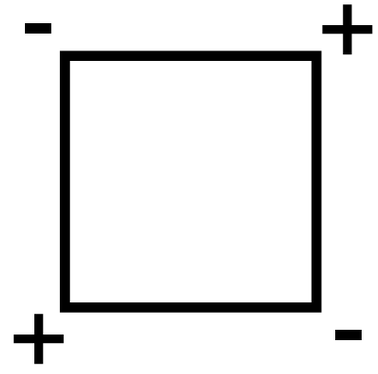




1000



1100



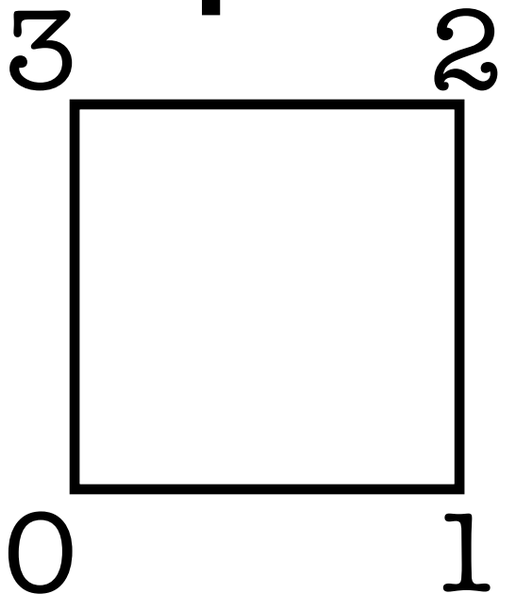
1010

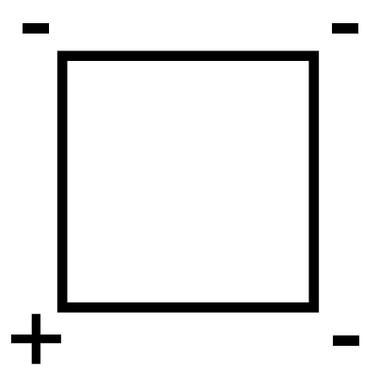
$\overline{0}\ \overline{1}\ \overline{2}\ \overline{3}$

0000	0100	1000	0000
0100	1110	1001	0000
0110	1011	0001	0000
0010	0001	0000	0000

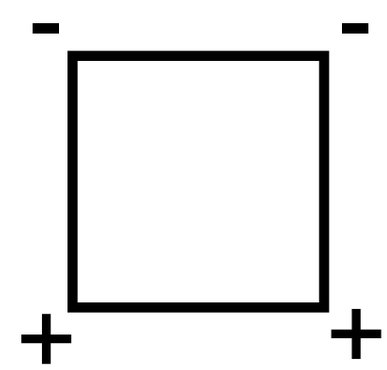
Binary	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Squares With The Case Table

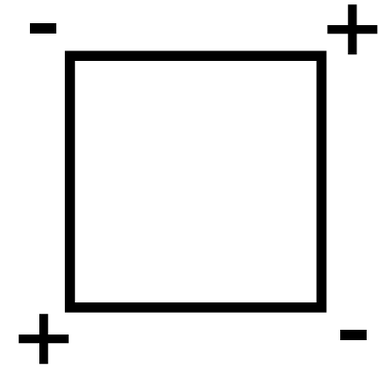




1000

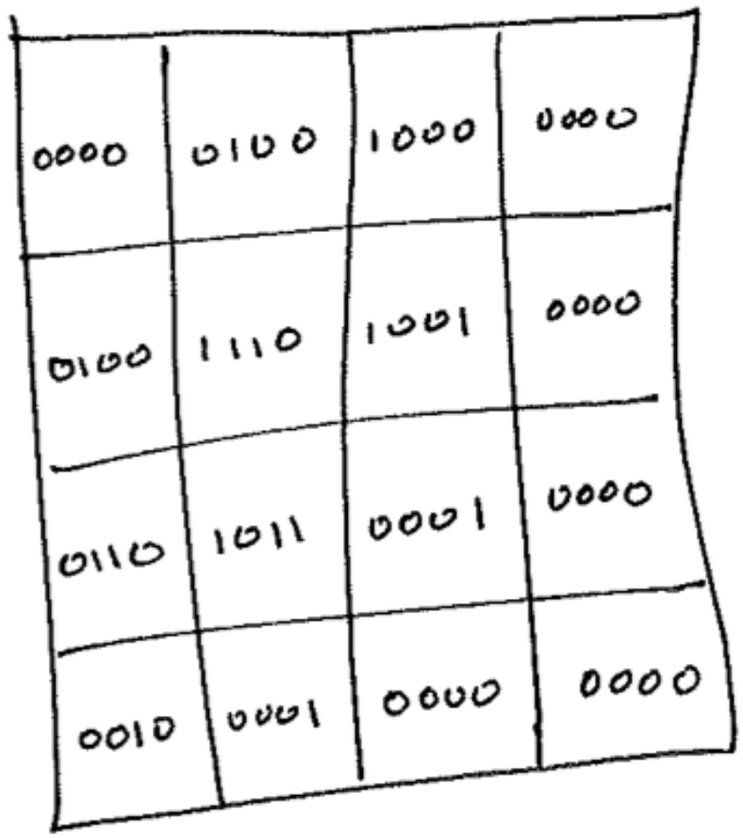


1100



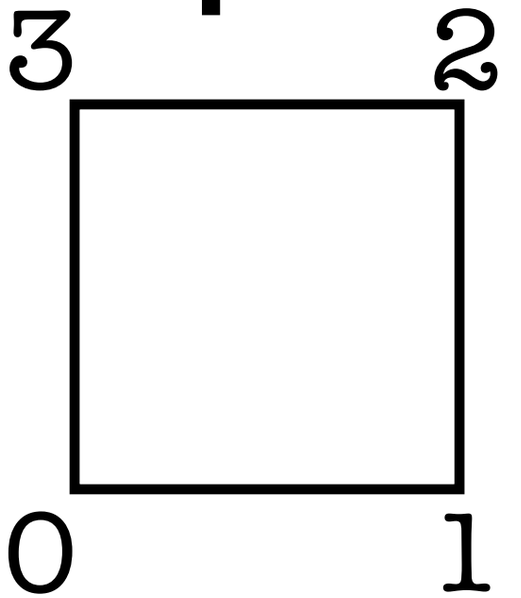
1010

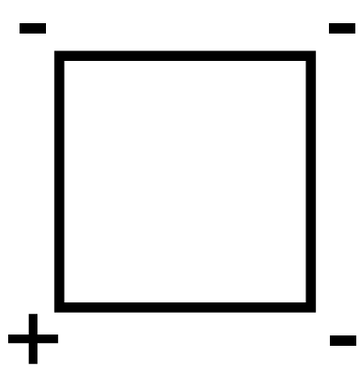
$\overline{0}\ \overline{1}\ \overline{2}\ \overline{3}$



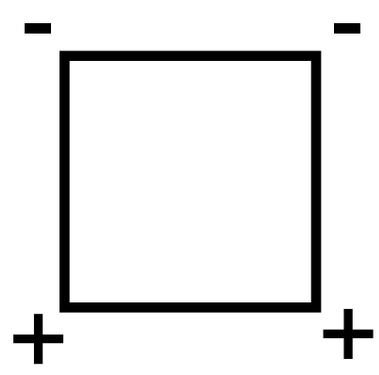
Binary	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Squares With The Case Table

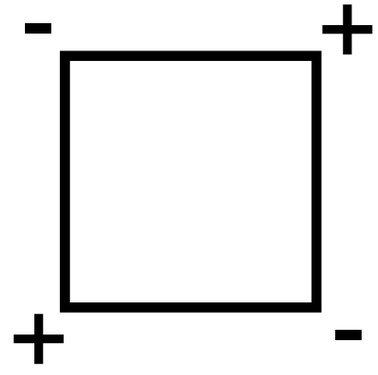




1000



1100



1010

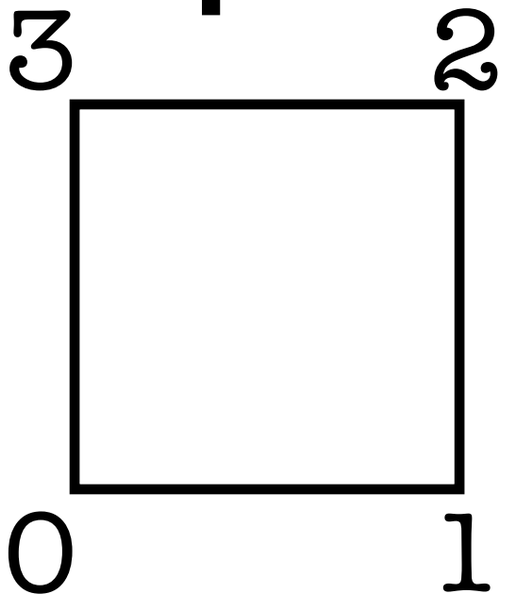
$\overline{0}\ \overline{1}\ \overline{2}\ \overline{3}$

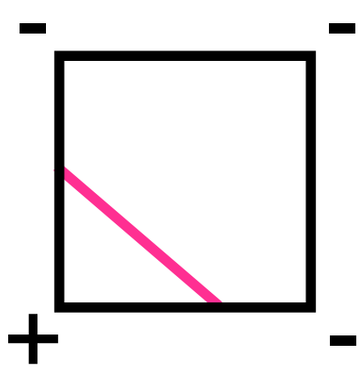
0000	0100	1000	0000
0100	1110	1001	0000
0110	1011	0001	0000
0010	0001	0000	0000

0	4	8	0
4	14	9	0
6	11	1	0
2	1	0	0

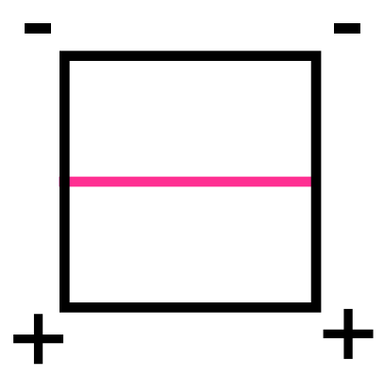
Binary	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Squares With The Case Table

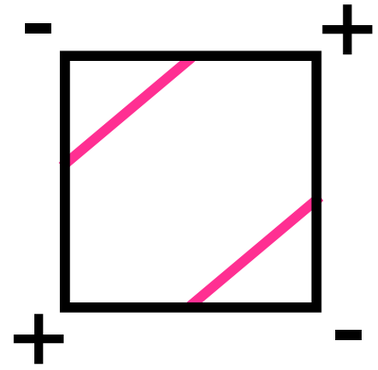




1000



1100



1010

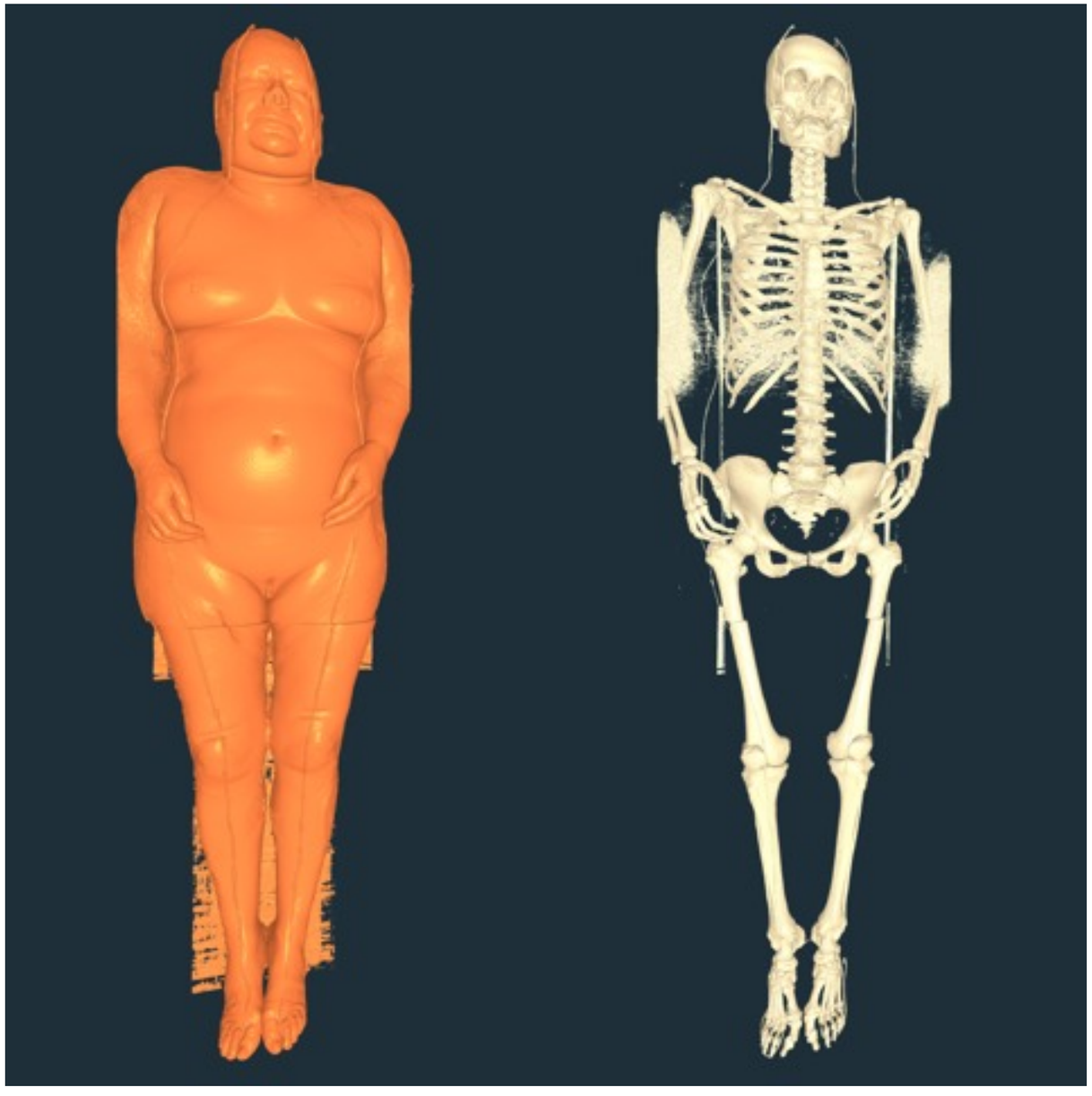
$\overline{0}\ \overline{1}\ \overline{2}\ \overline{3}$

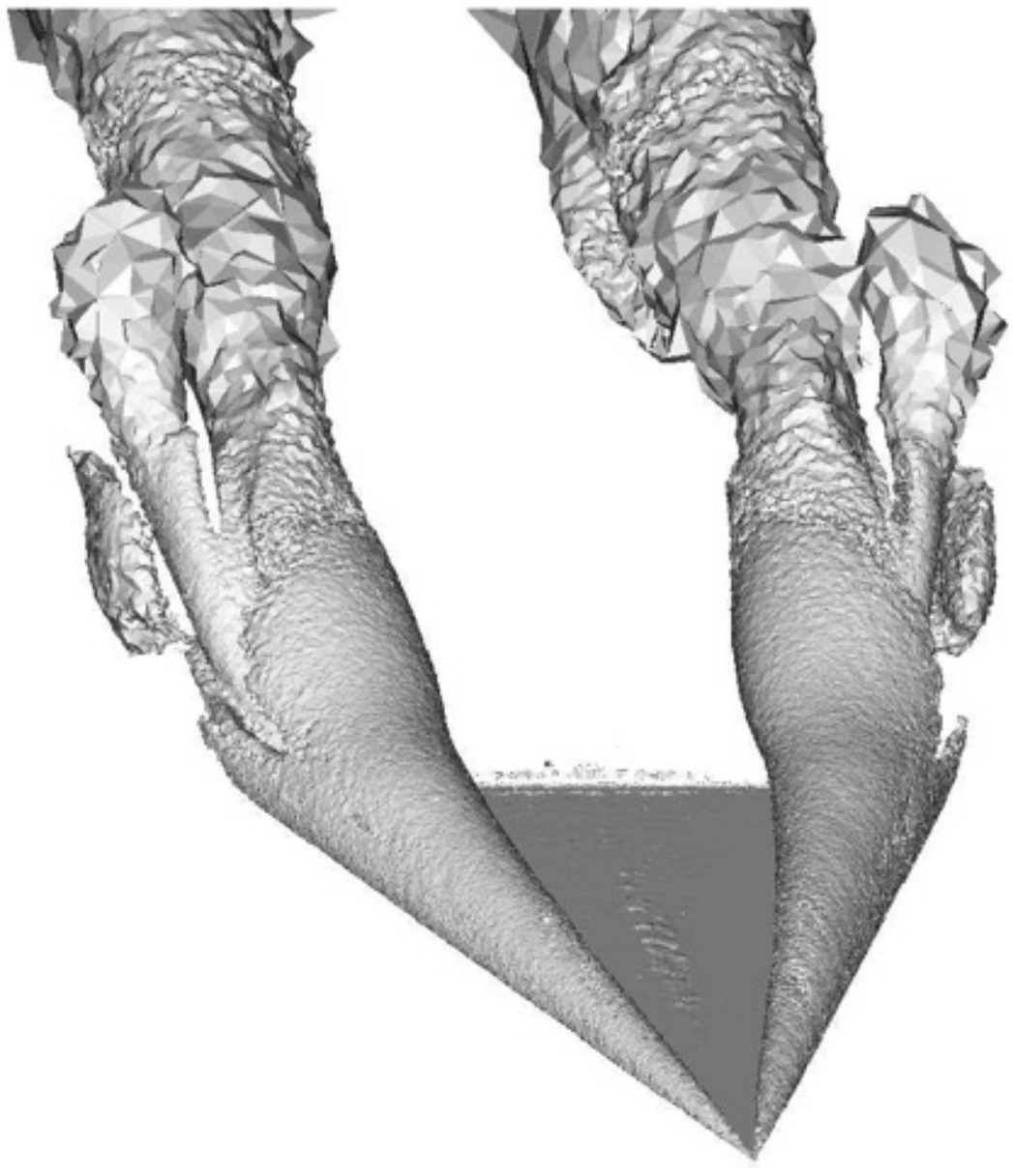
0000	0100	1000	0000
0100	1110	1001	0000
0110	1011	0001	0000
0010	0001	0000	0000

0	4	8	0
4	14	9	0
6	11	1	0
2	1	0	0

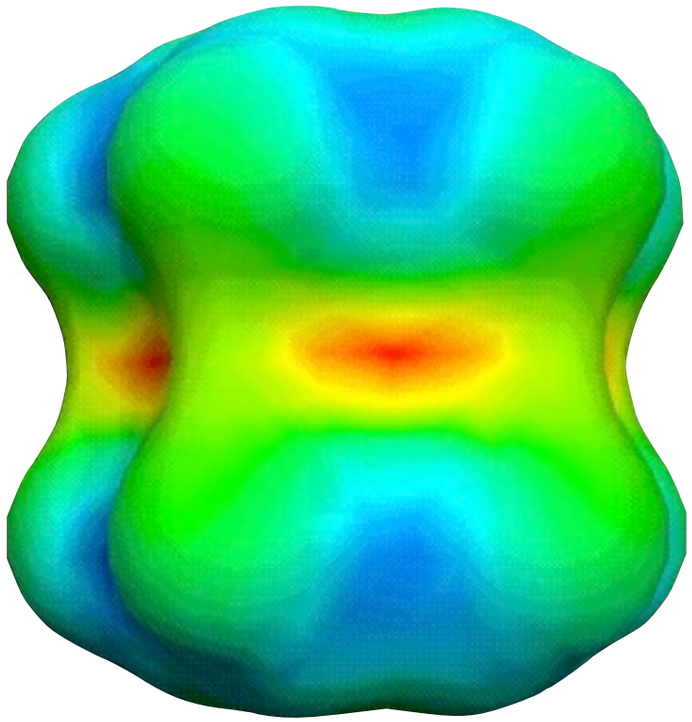
Isosurfaces of 3D Scalar Fields

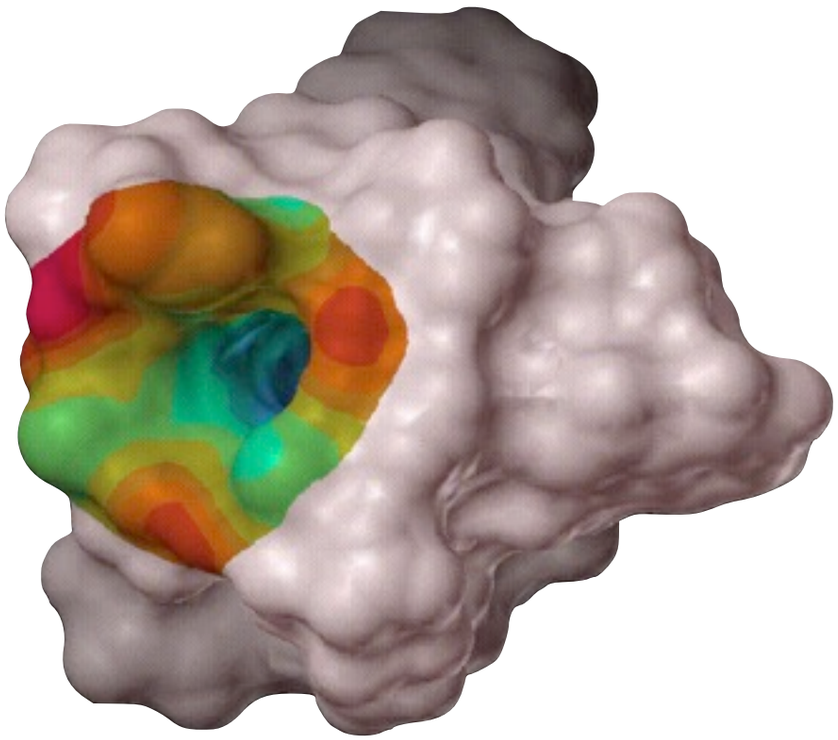
Other Examples

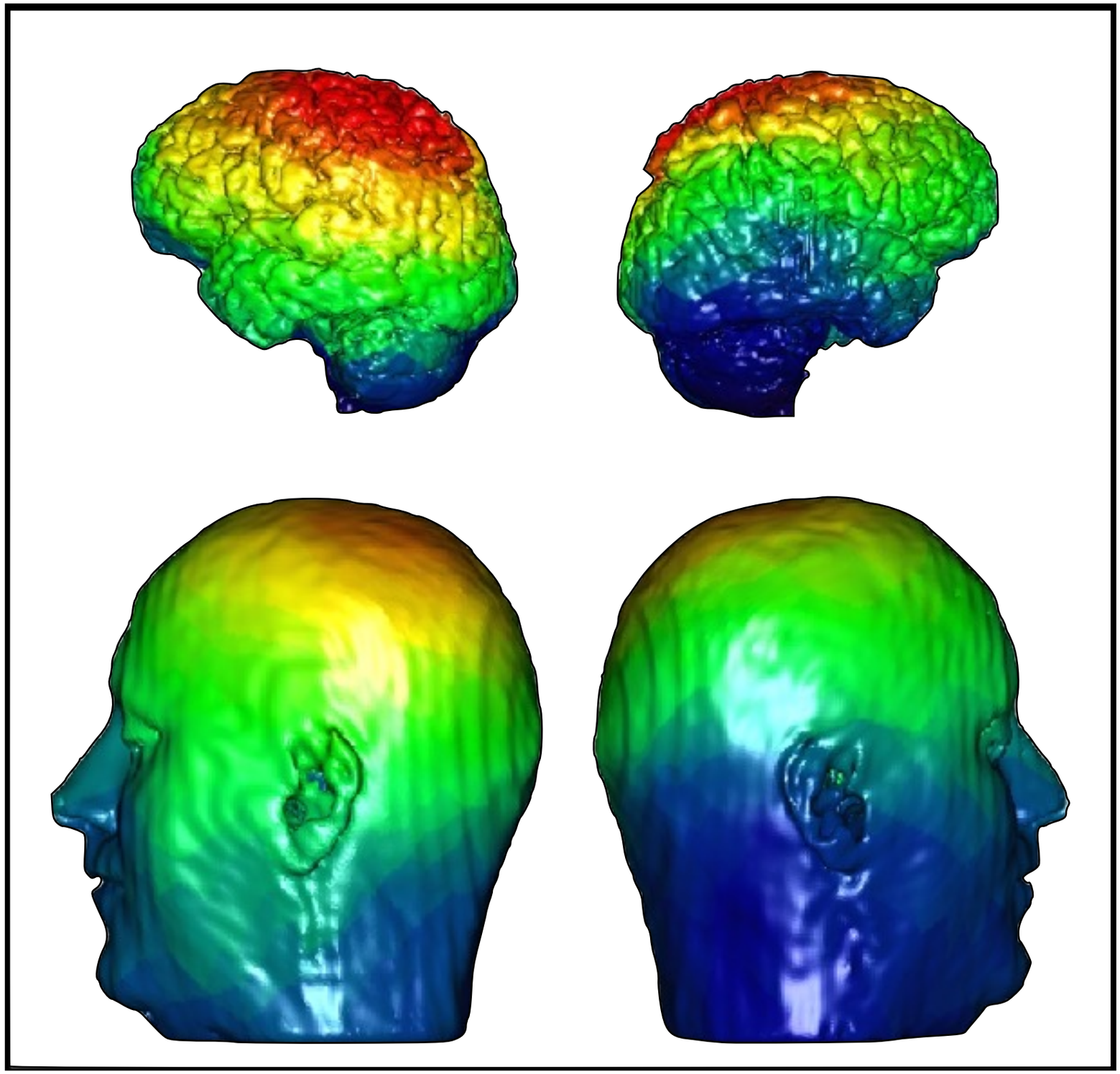




Colored Isosurfaces

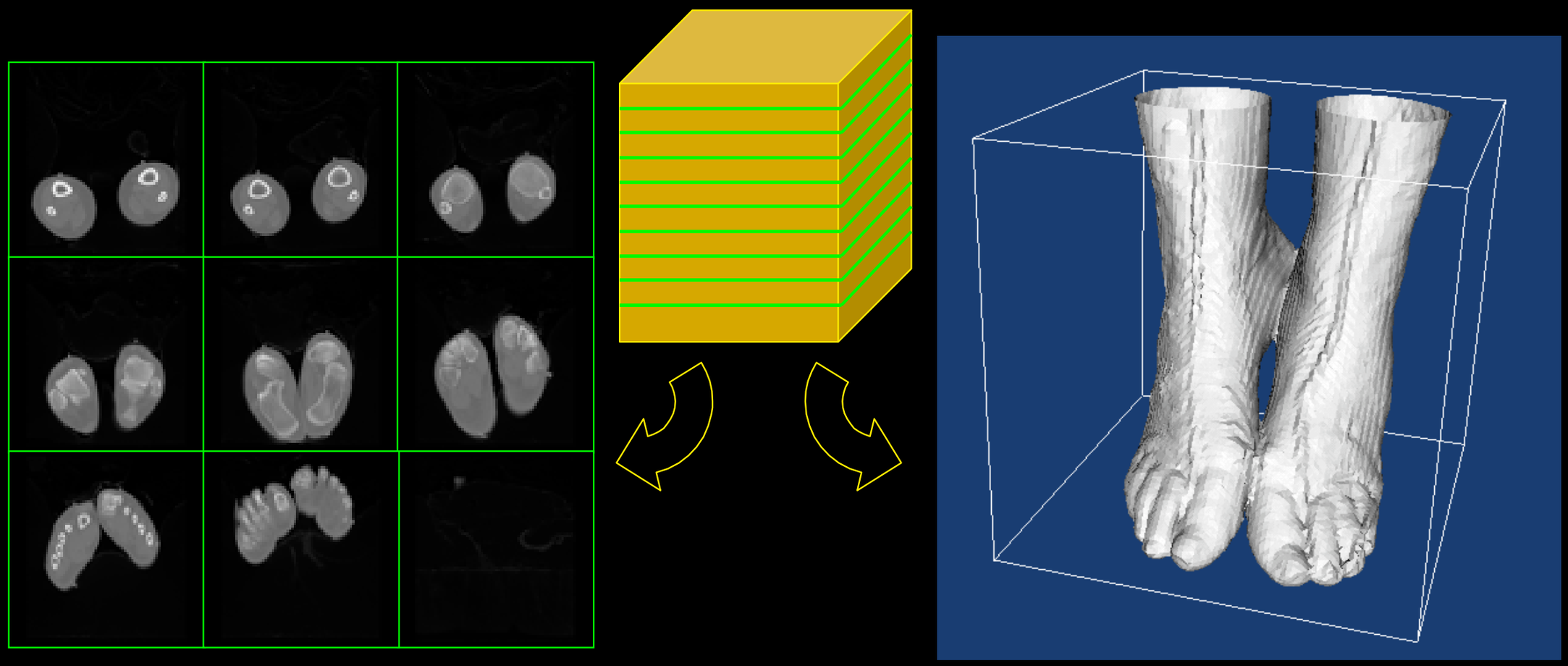






Isosurfacing

- You're given a big 3D block of numbers
- Make a picture
- Slicing shows data, but not its 3D shape
- Isosurfacing is one of the simplest ways



MARCHING CUBES: A HIGH RESOLUTION 3D SURFACE CONSTRUCTION ALGORITHM

William E. Lorensen
Harvey E. Cline

General Electric Company
Corporate Research and Development
Schenectady, New York 12301

18,716 Citations on Google Scholar

Related (earlier) paper: Wyvill et al. "Soft objects." Advanced Computer Graphics. Springer Japan, 1986. 113-128.

using linear interpolation. We find the gradient of the original data, normalize it, and use it as a basis for shading the models. The detail in images produced from the generated surface models is the result of maintaining the inter-slice connectivity, surface data, and gradient information present in the original 3D data. Results from computed tomography (CT), magnetic resonance (MR), and single-photon emission computed tomography (SPECT) illustrate the quality and functionality of *marching cubes*. We also discuss improvements that decrease processing time and add solid modeling capabilities.

CR Categories: 3.3, 3.5

Additional Keywords: computer graphics, medical imaging, surface reconstruction

Existing 3D algorithms lack detail and sometimes introduce artifacts. We present a new, high-resolution 3D surface construction algorithm that produces models with unprecedented detail. This new algorithm, called *marching cubes*, creates a polygonal representation of constant density surfaces from a 3D array of data. The resulting model can be displayed with conventional graphics-rendering algorithms implemented in software or hardware.

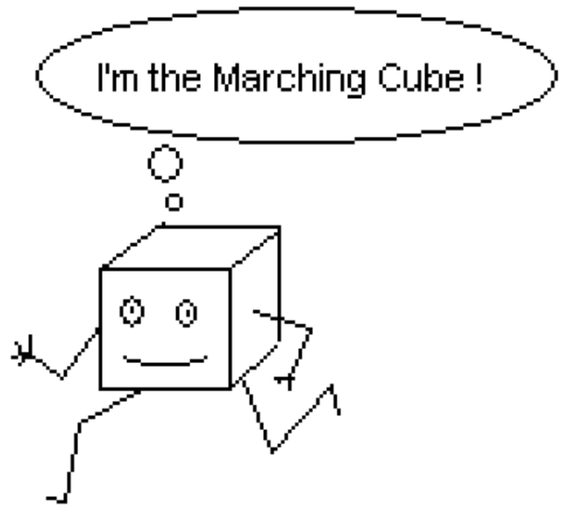
After describing the information flow for 3D medical applications, we describe related work and discuss the drawbacks of that work. Then we describe the algorithm as well as efficiency and functional enhancements, followed by case studies using three different medical imaging techniques to illustrate the new algorithm's capabilities.

Math for 3D Isosurfaces

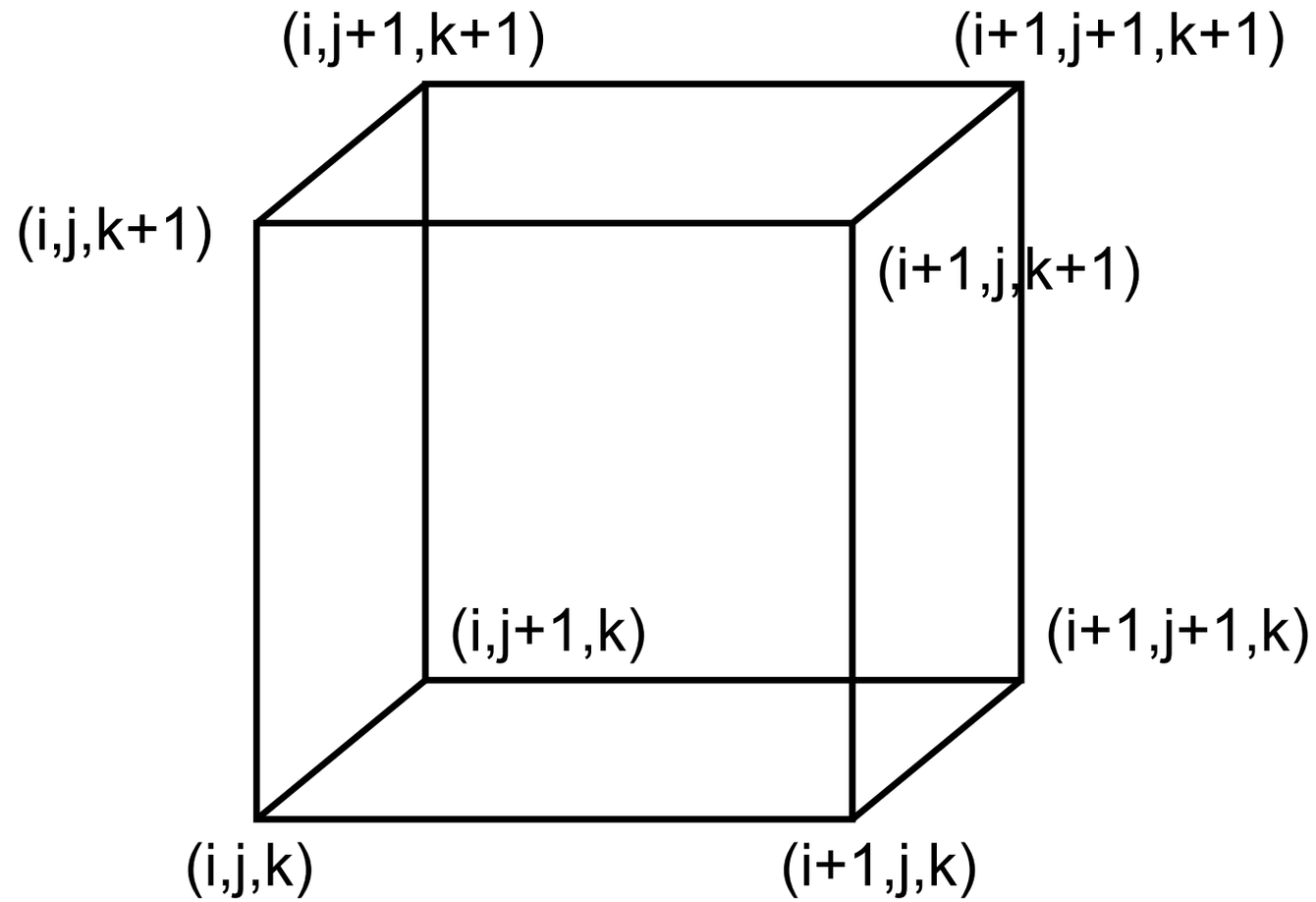
- **Input**: a 3D scalar field: f: R^3 -> R
- Isosurfaces are still the **preimage** of a single scalar value v:
  - $S_v = \{ (x,y,z) \mid (x,y,z) = f^{-1}(v) \}$
- Once we extract S_v, we can then visualize this set as a surface in 3D
- Unlike in 2D, frequently only look at one surface at a time

- The core MC algorithm
  - Cell consists of 4(8) pixel (voxel) values: (i+[01], j+[01], k+[01])

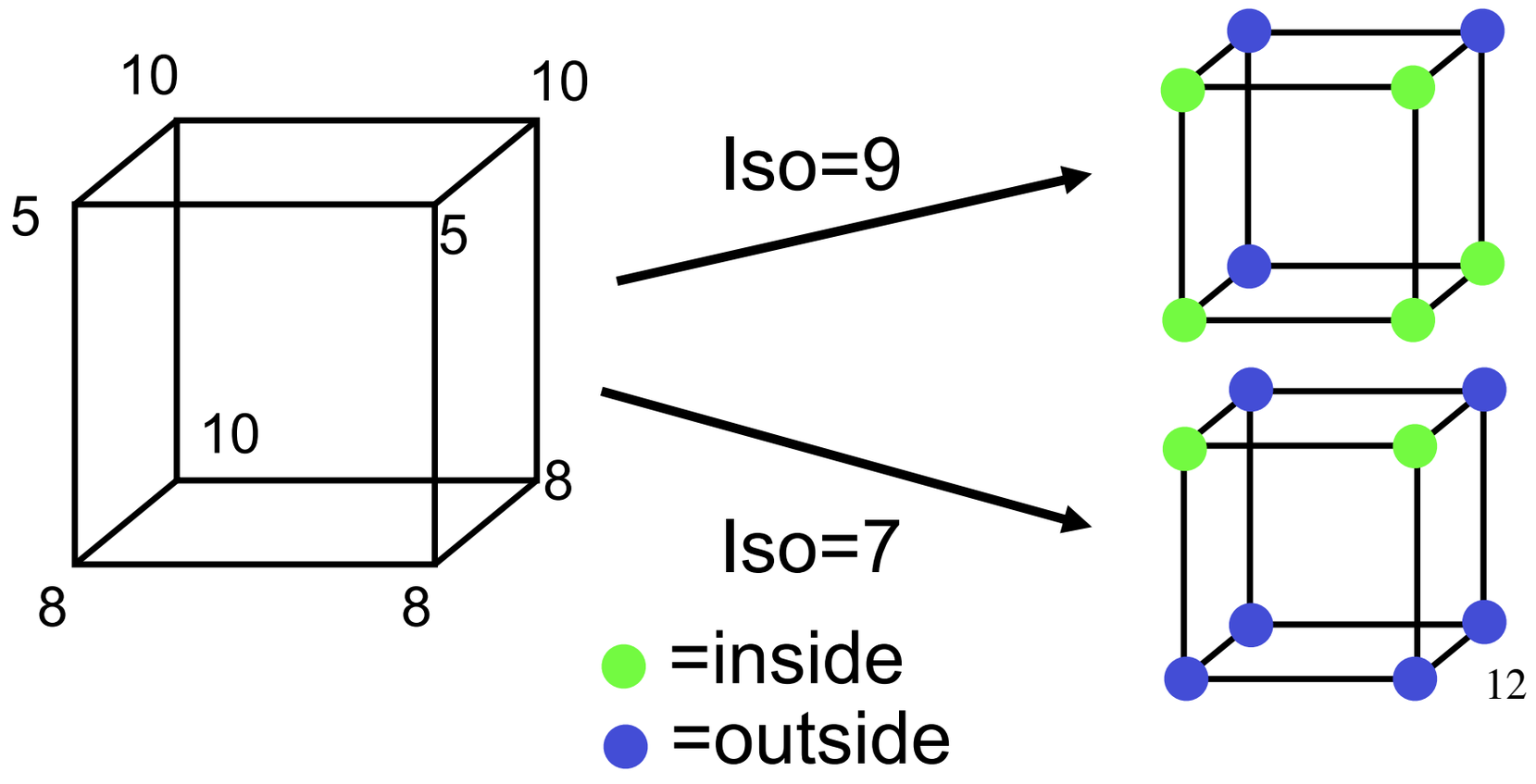
1. Consider a cell
2. Classify each vertex as inside or outside
3. Build an index
4. Get edge list from table[index]
5. Interpolate the edge location
6. Compute gradients
7. Consider ambiguous cases
8. Go to next cell



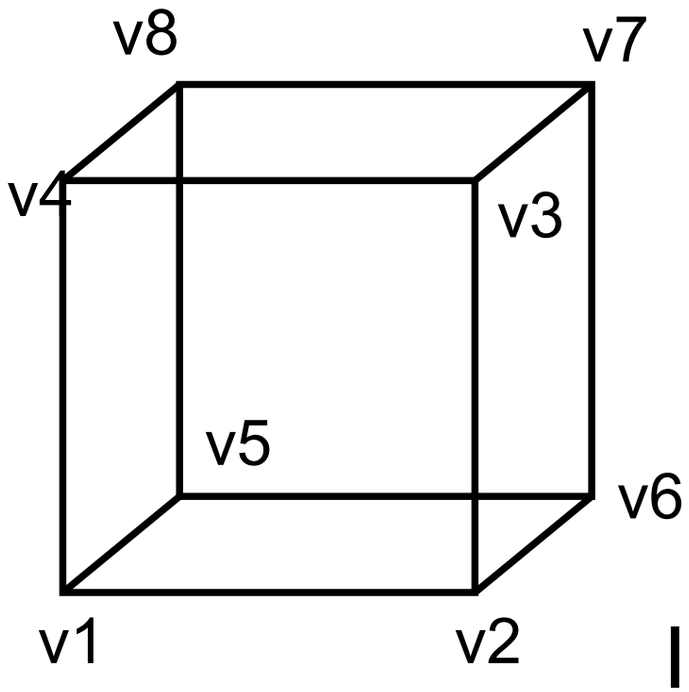
- Step 1: Consider a cell defined by eight data values



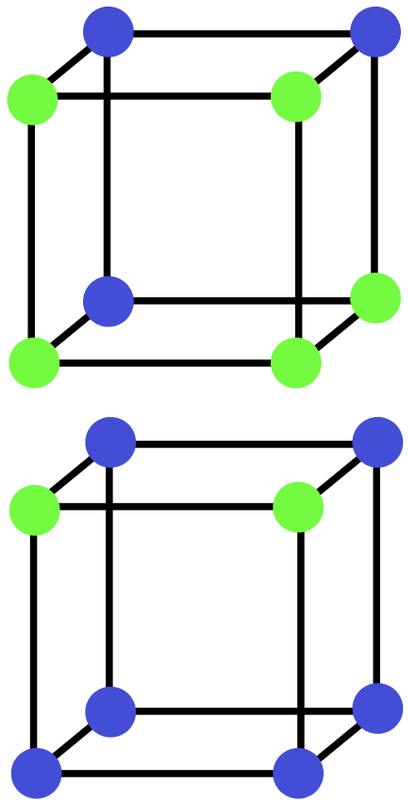
- Step 2: Classify each voxel according to whether it lies
  - Outside the surface (value > isosurface value)
  - Inside the surface (value <= isosurface value)



- Step 3: Use the binary labeling of each voxel to create an index



inside =1
outside=0



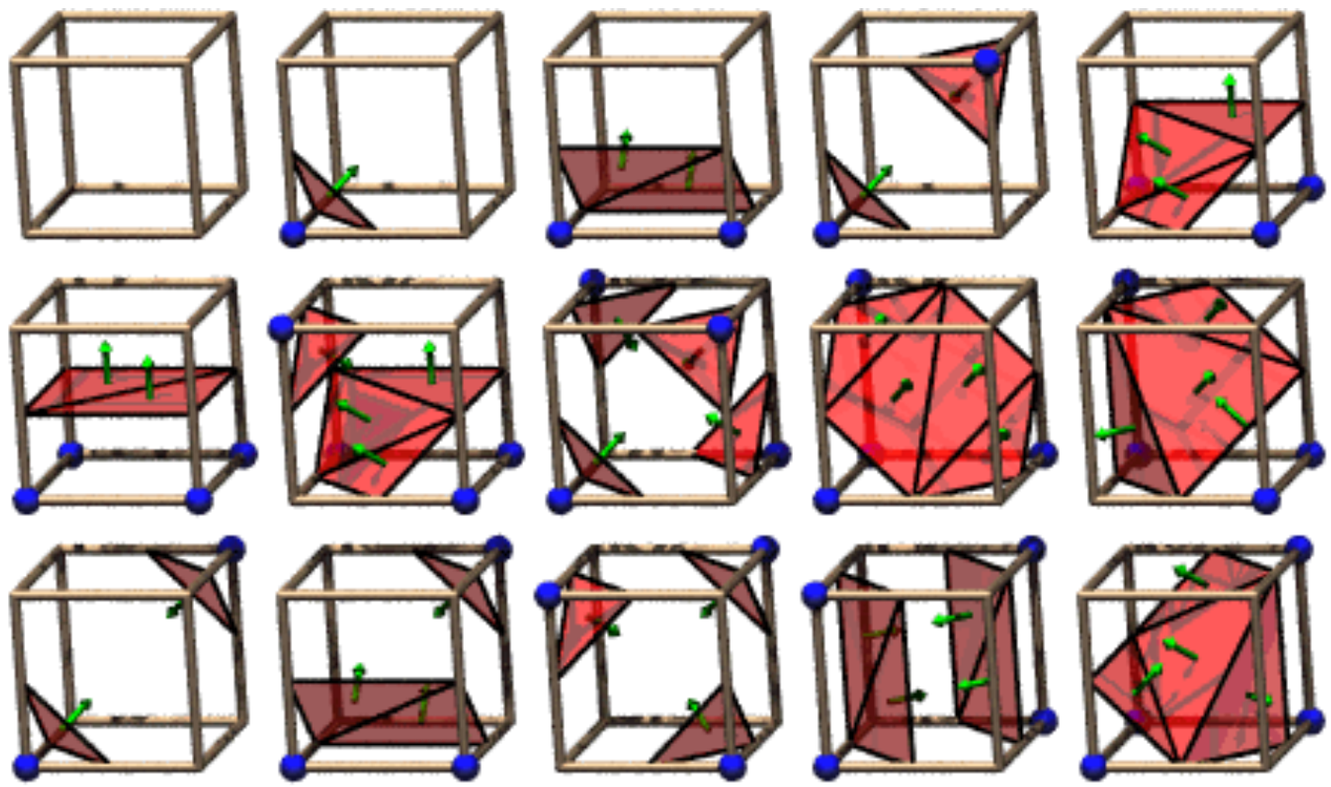
11110100

00110000

Index:

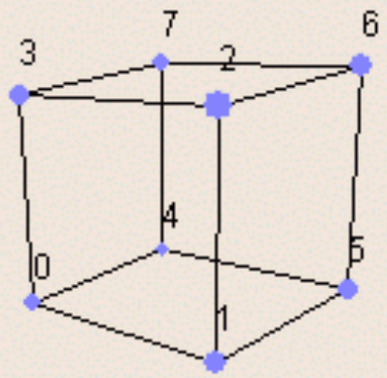
v1	v2	v3	v4	v5	v6	v7	v8

- Step 4: For a given index, access an array storing a list of edges
  - All 256 cases can be derived from 1+14=15 base cases due to symmetries

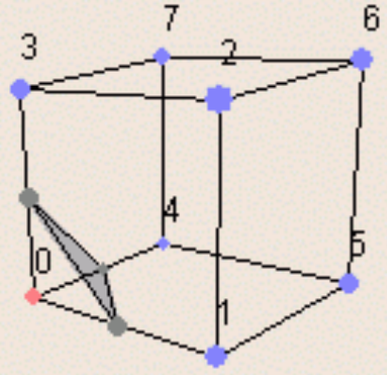


The 15 Cube Combinations

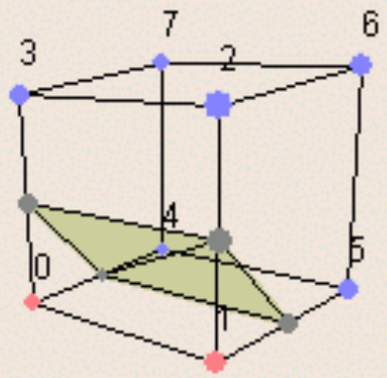
Case Table



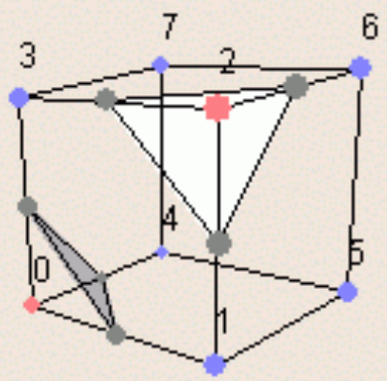
Case 0



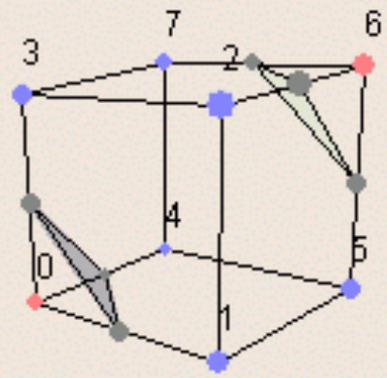
Case 1



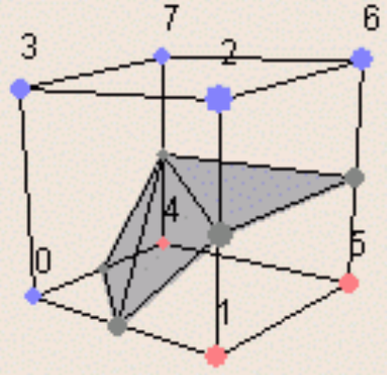
Case 2



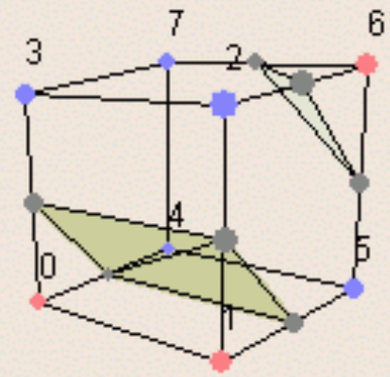
Case 3



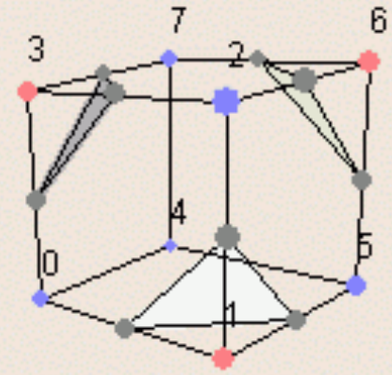
Case 4



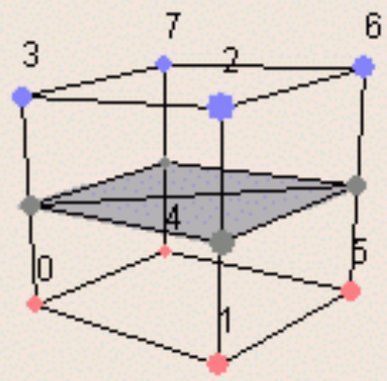
Case 5



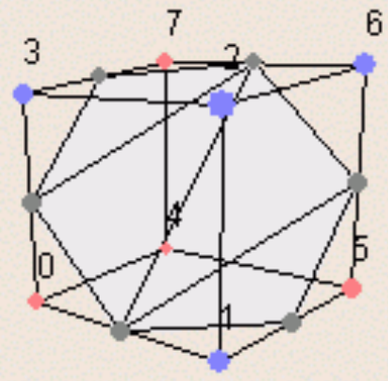
Case 6



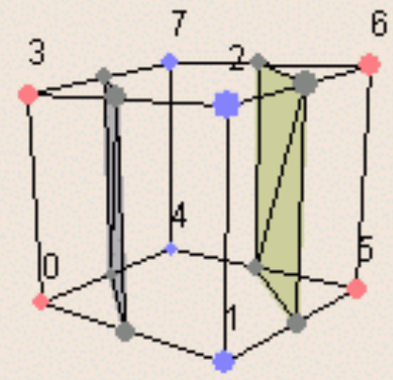
Case 7



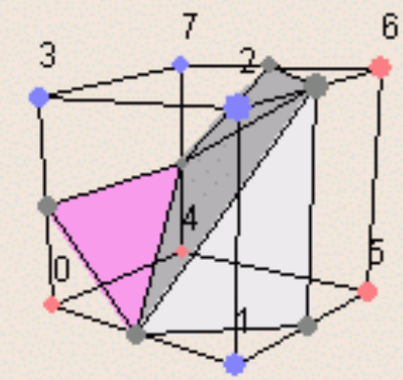
Case 8



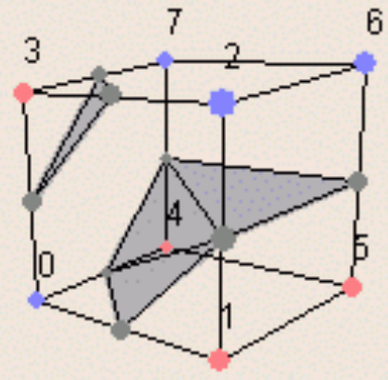
Case 9



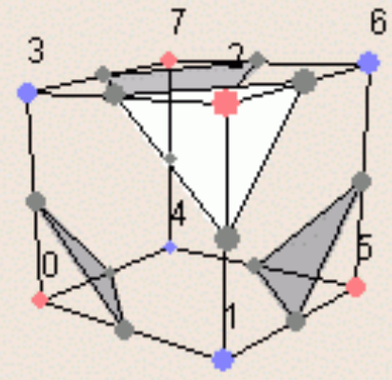
Case 10



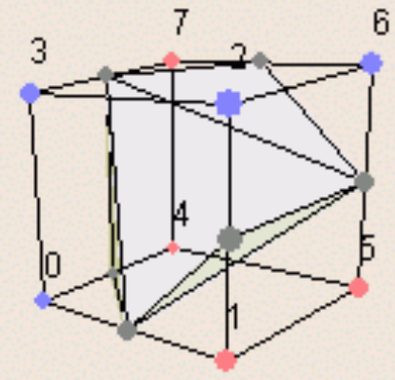
Case 11



Case 12



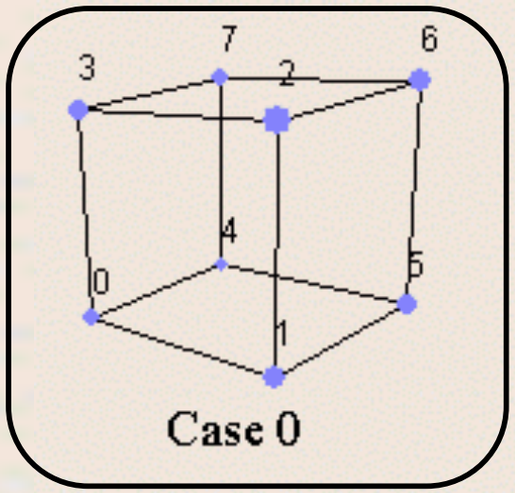
Case 13



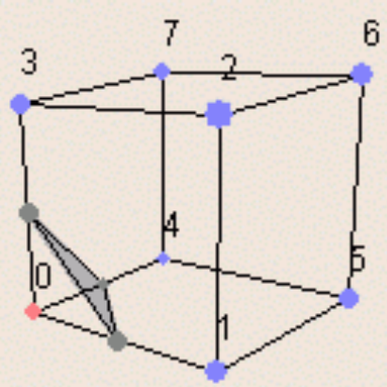
Case 14

- 8 Above
- 0 Below

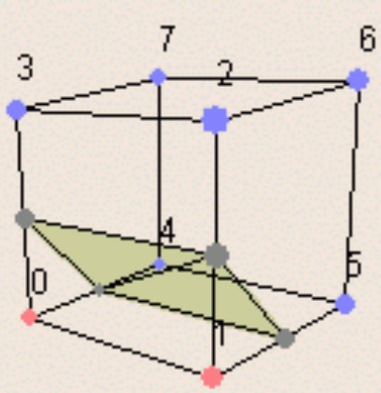
1 case



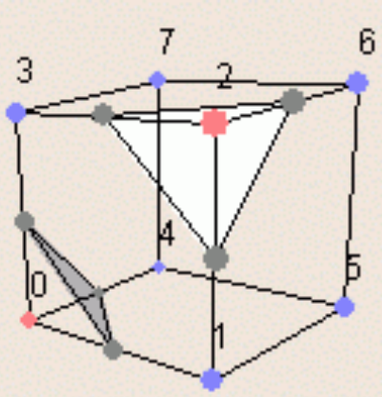
Case 0



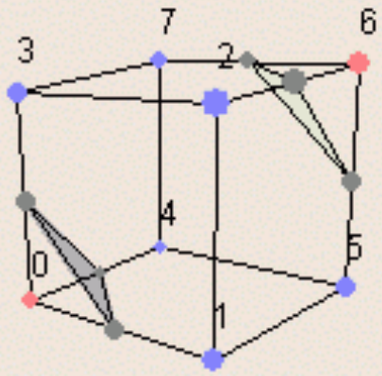
Case 1



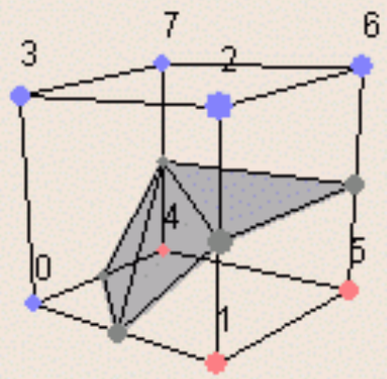
Case 2



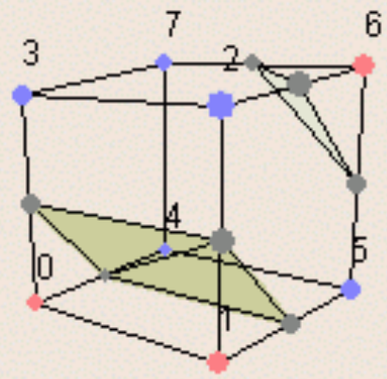
Case 3



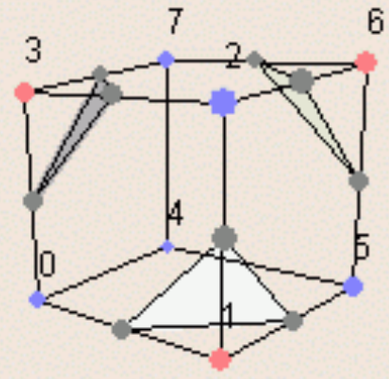
Case 4



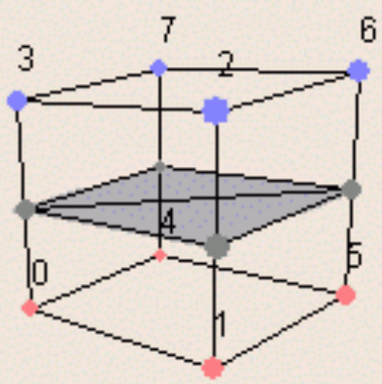
Case 5



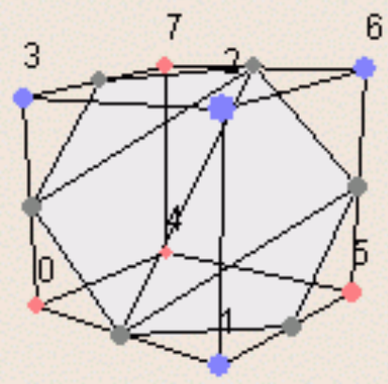
Case 6



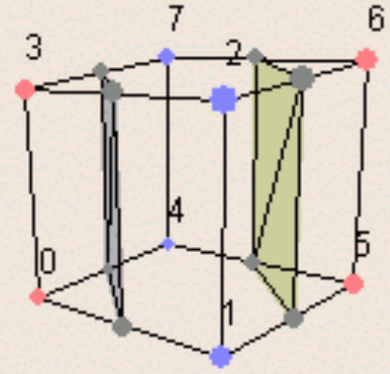
Case 7



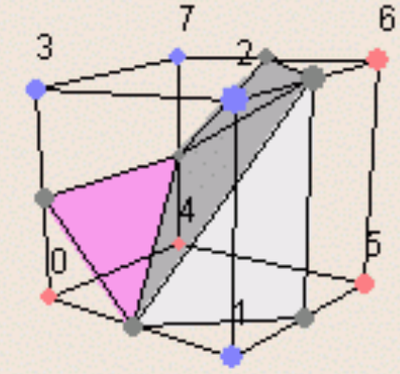
Case 8



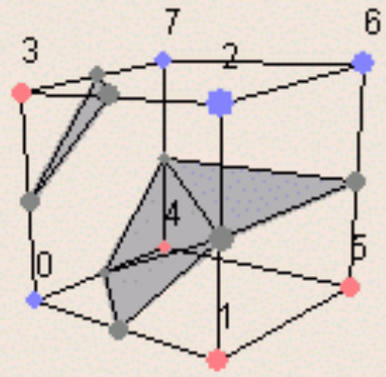
Case 9



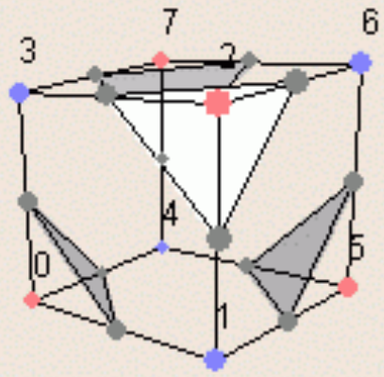
Case 10



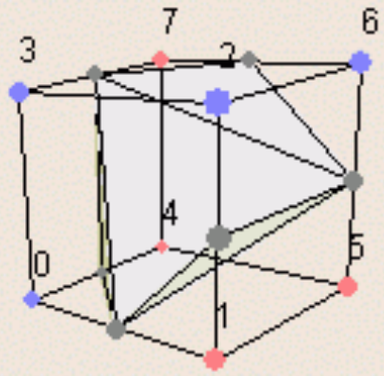
Case 11



Case 12



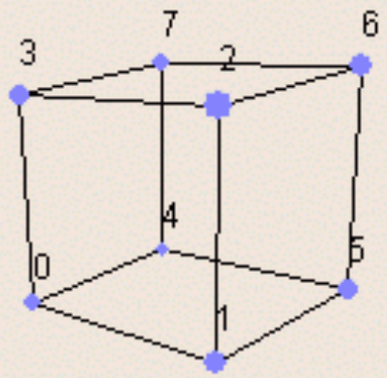
Case 13



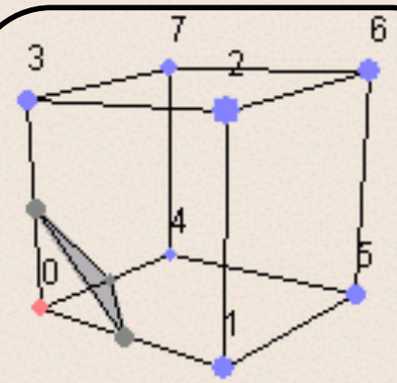
Case 14

- 7 Above
- 1 Below

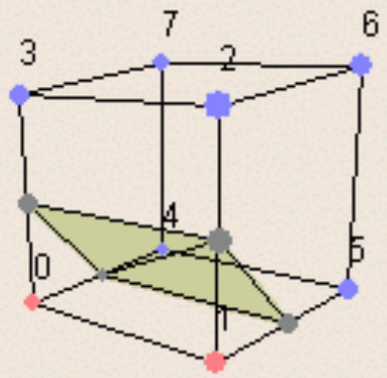
1 case



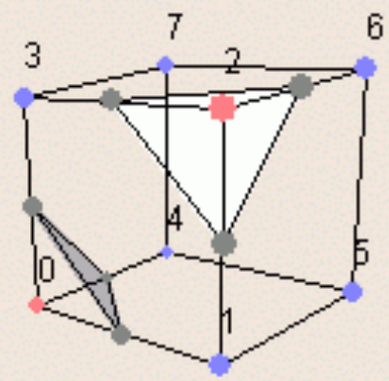
Case 0



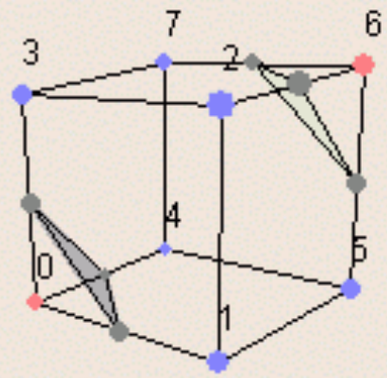
Case 1



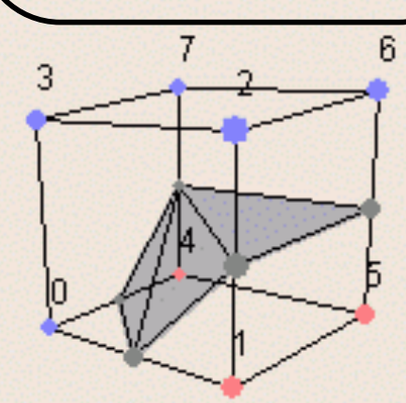
Case 2



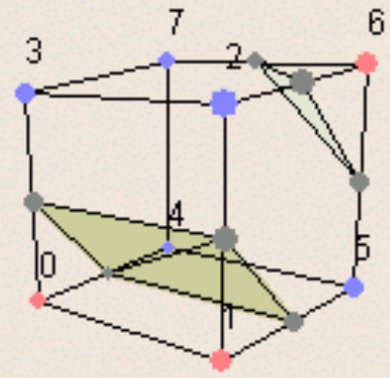
Case 3



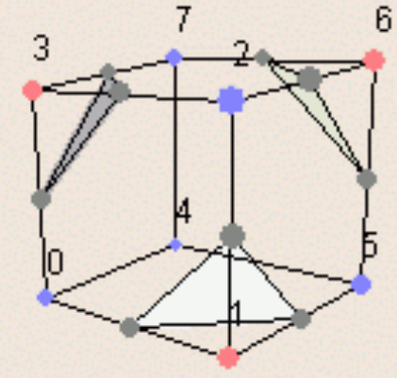
Case 4



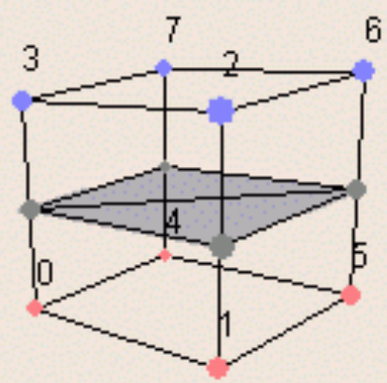
Case 5



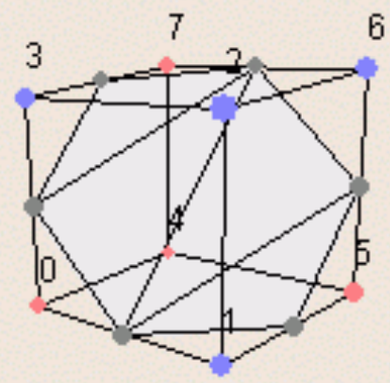
Case 6



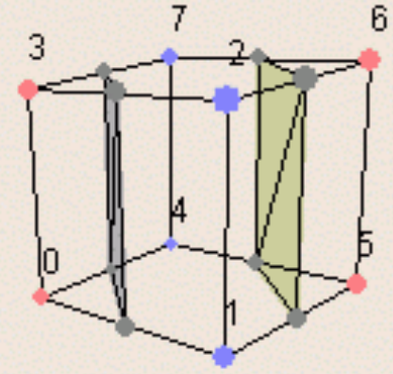
Case 7



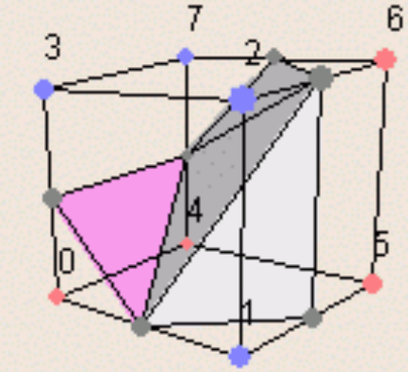
Case 8



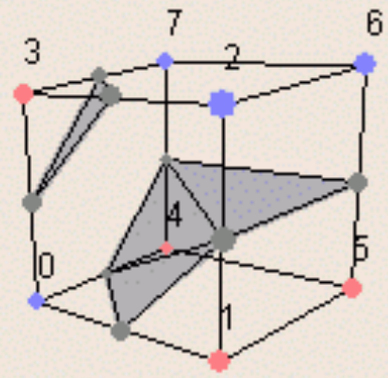
Case 9



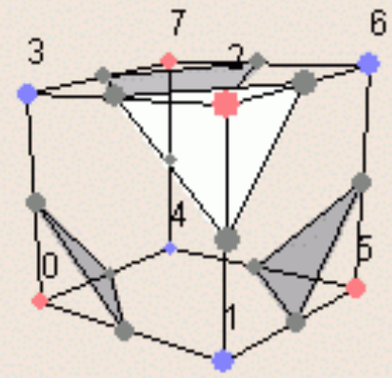
Case 10



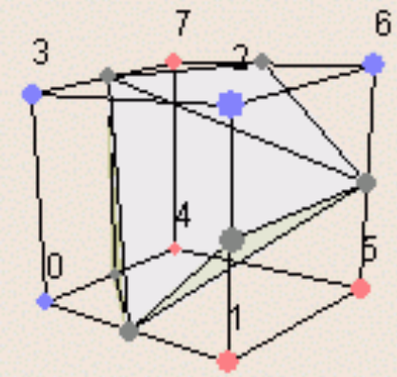
Case 11



Case 12



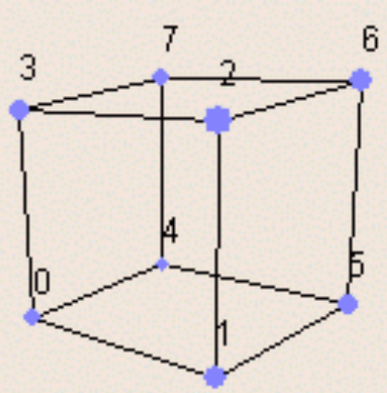
Case 13



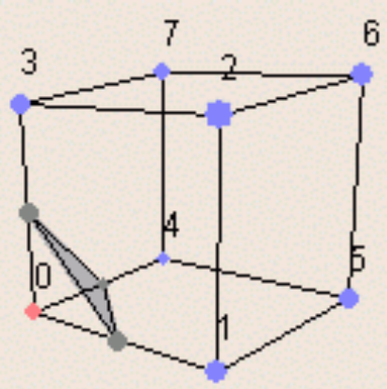
Case 14

- 6 Above
- 2 Below

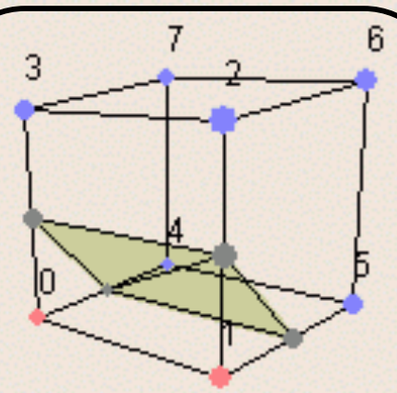
3 cases



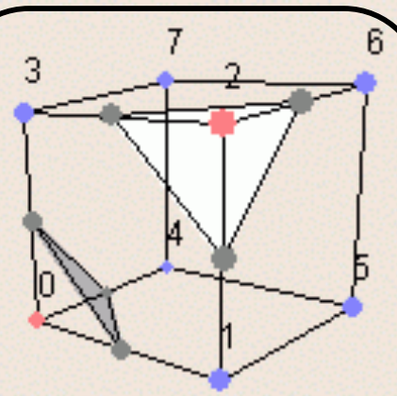
Case 0



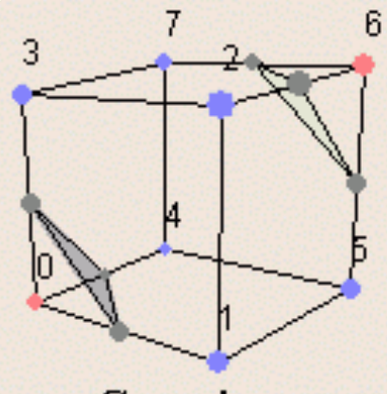
Case 1



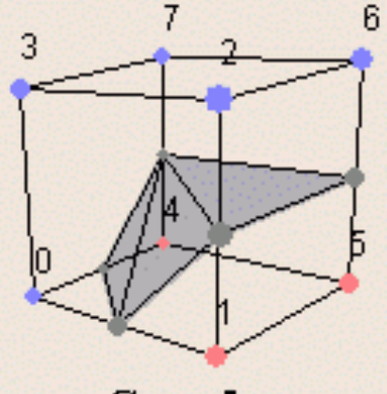
Case 2



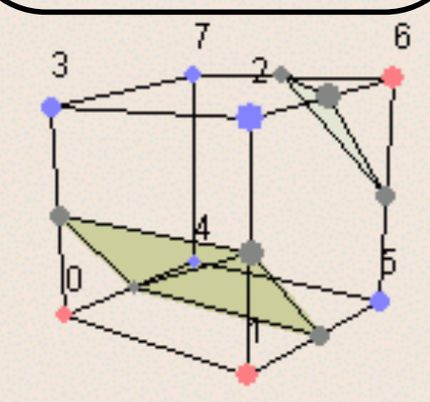
Case 3



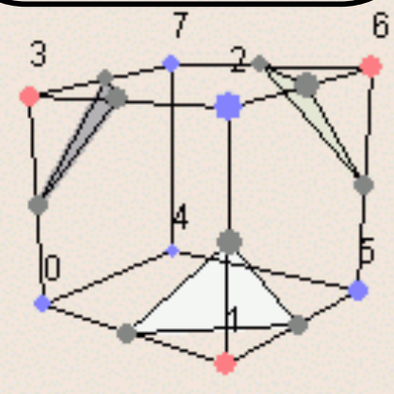
Case 4



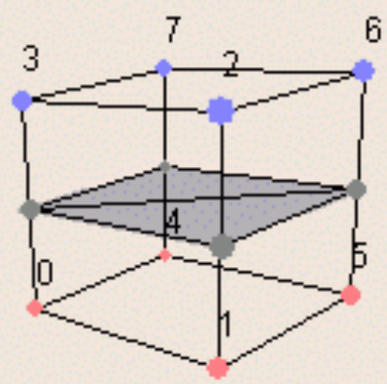
Case 5



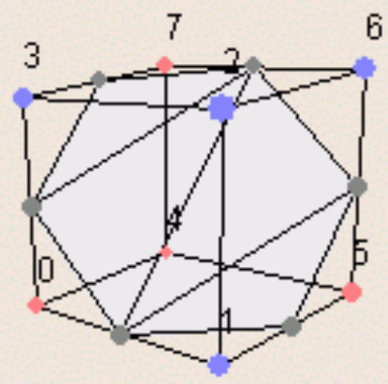
Case 6



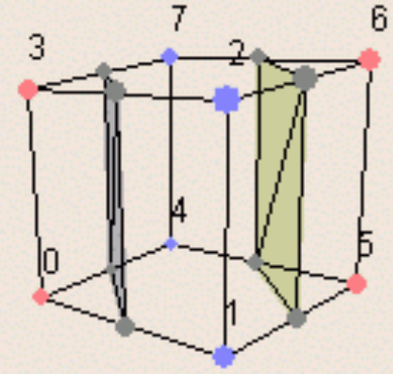
Case 7



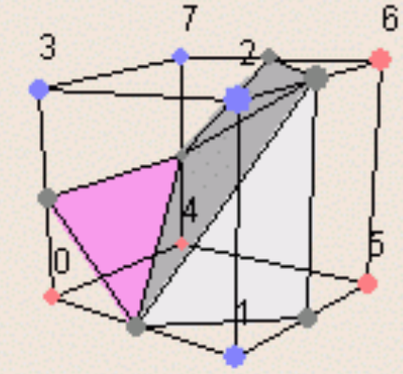
Case 8



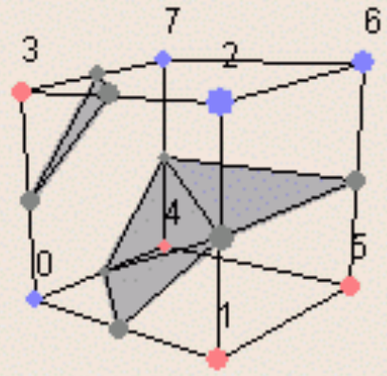
Case 9



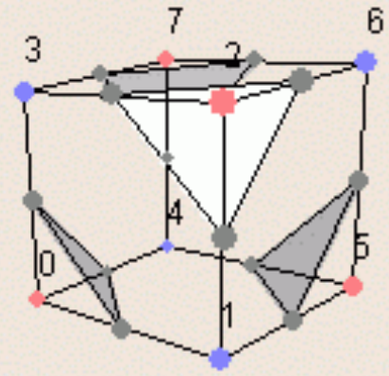
Case 10



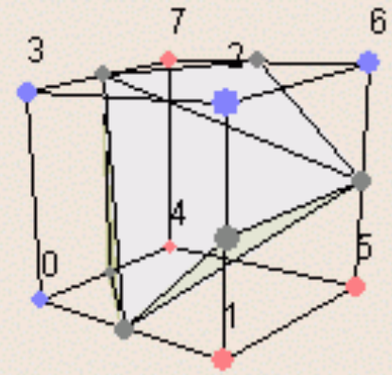
Case 11



Case 12



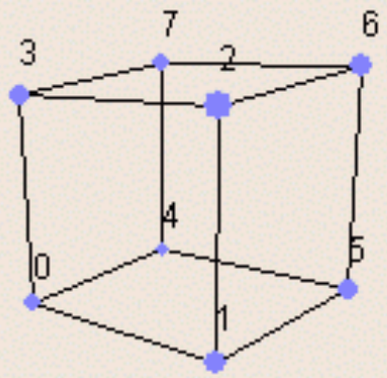
Case 13



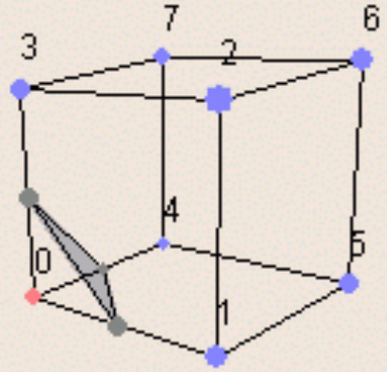
Case 14

- 5 Above
- 3 Below

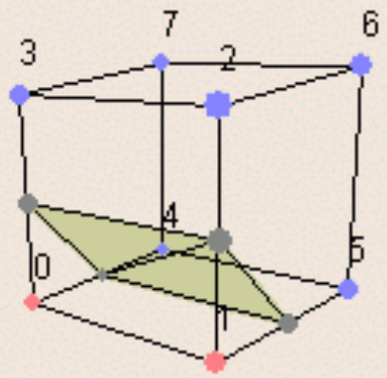
3 cases



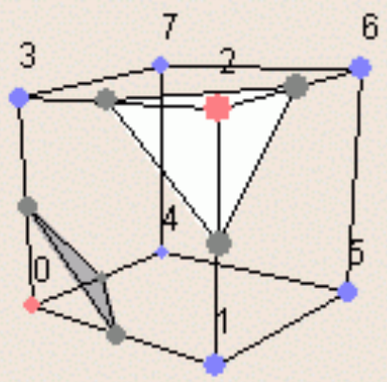
Case 0



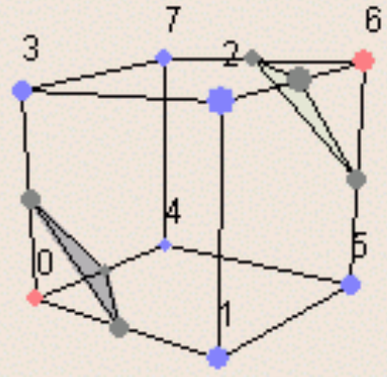
Case 1



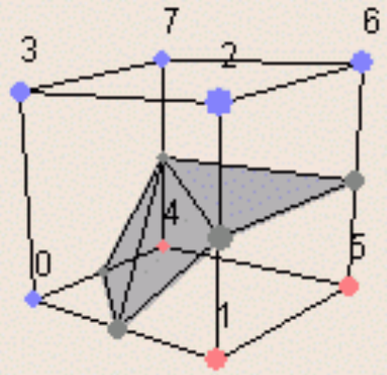
Case 2



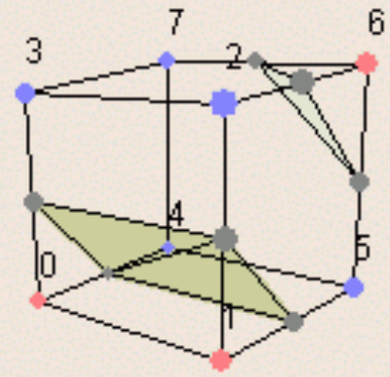
Case 3



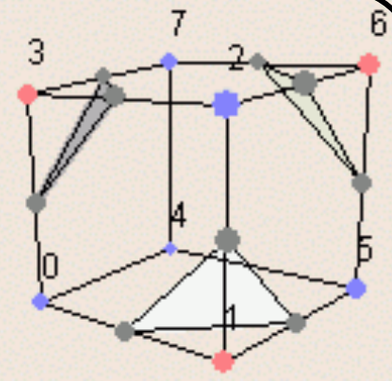
Case 4



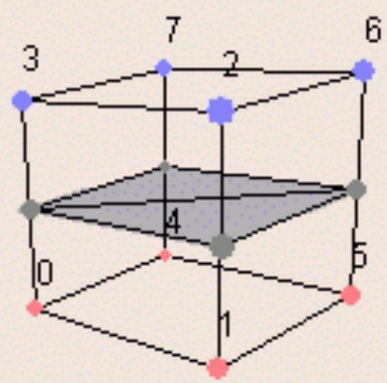
Case 5



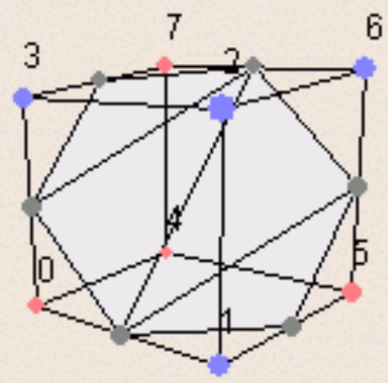
Case 6



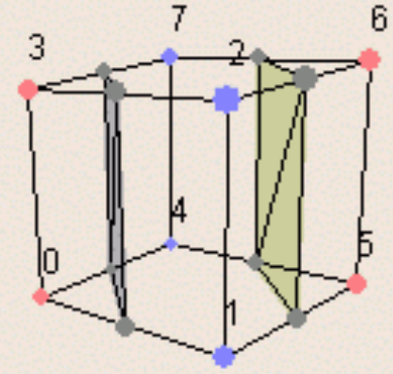
Case 7



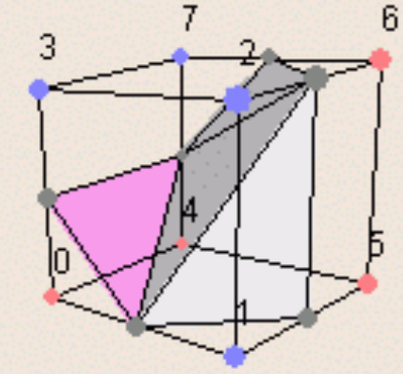
Case 8



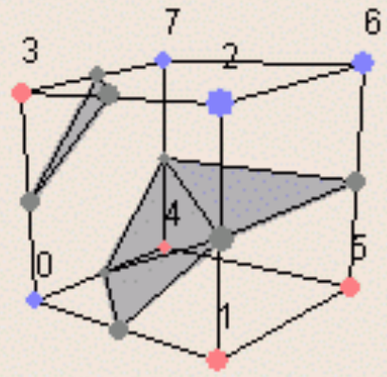
Case 9



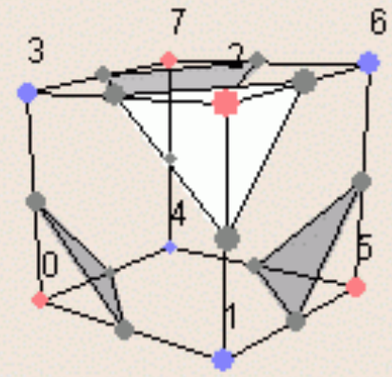
Case 10



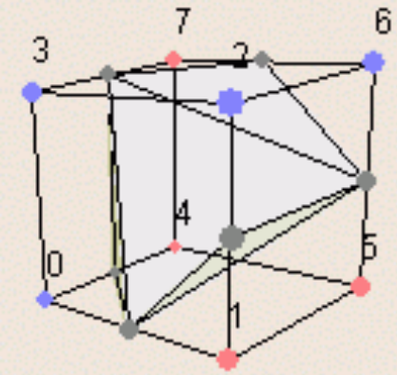
Case 11



Case 12



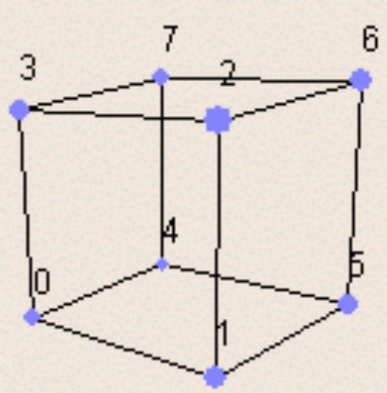
Case 13



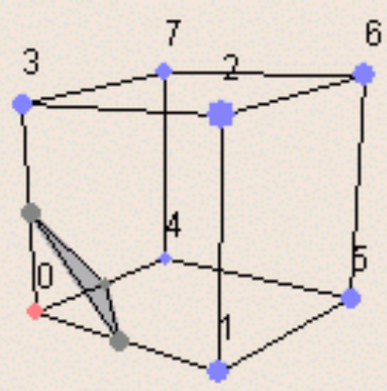
Case 14

- 4 Above
- 4 Below

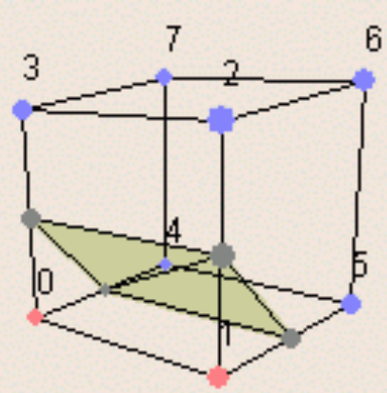
7 cases



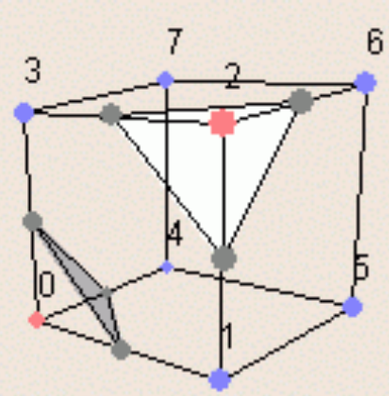
Case 0



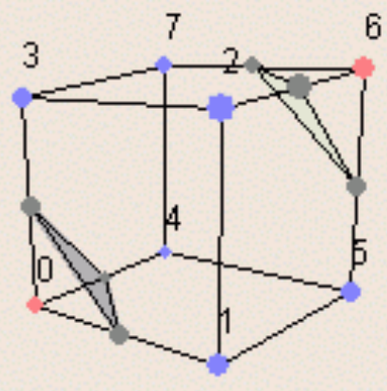
Case 1



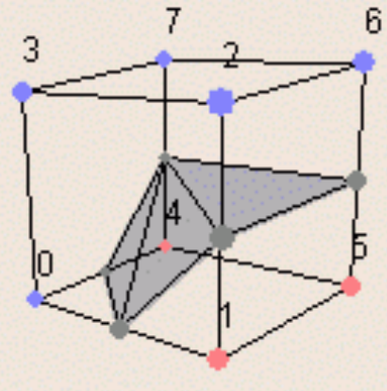
Case 2



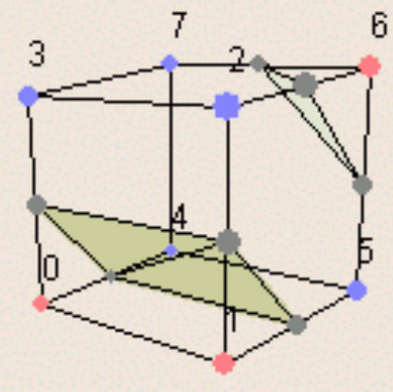
Case 3



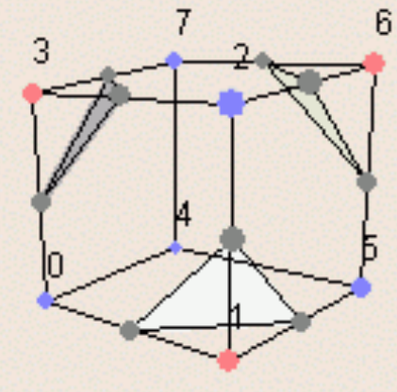
Case 4



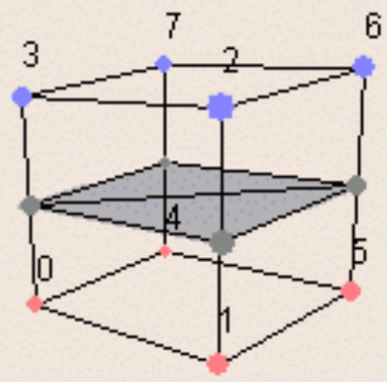
Case 5



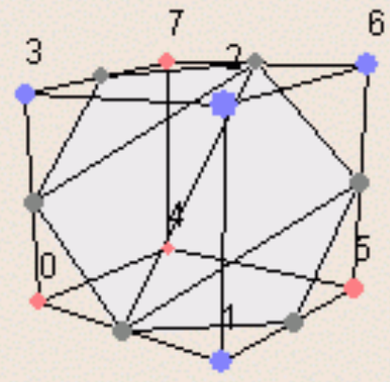
Case 6



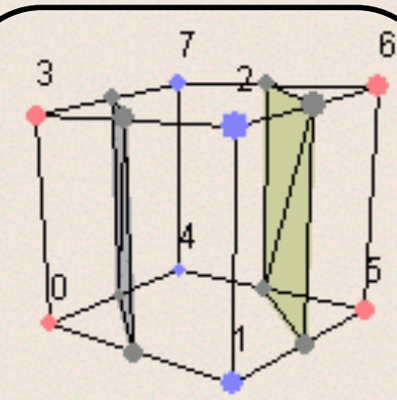
Case 7



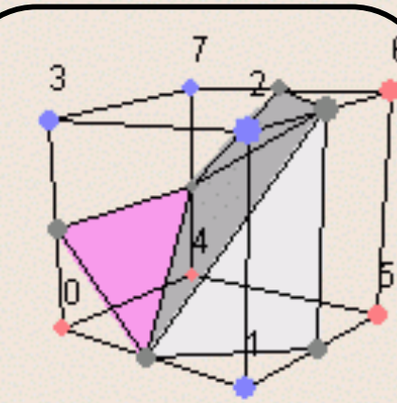
Case 8



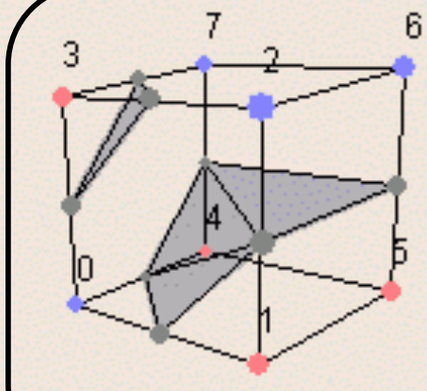
Case 9



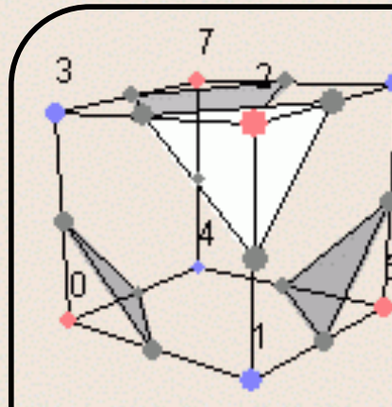
Case 10



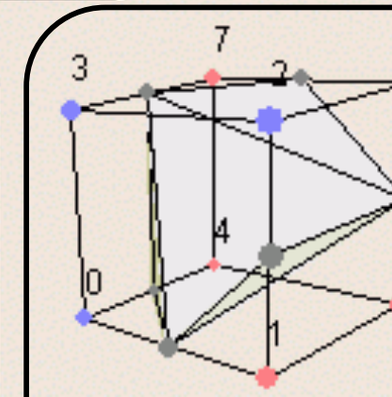
Case 11



Case 12



Case 13



Case 14

- Step 4 *cont.*: Get edge list from table
  - Example for

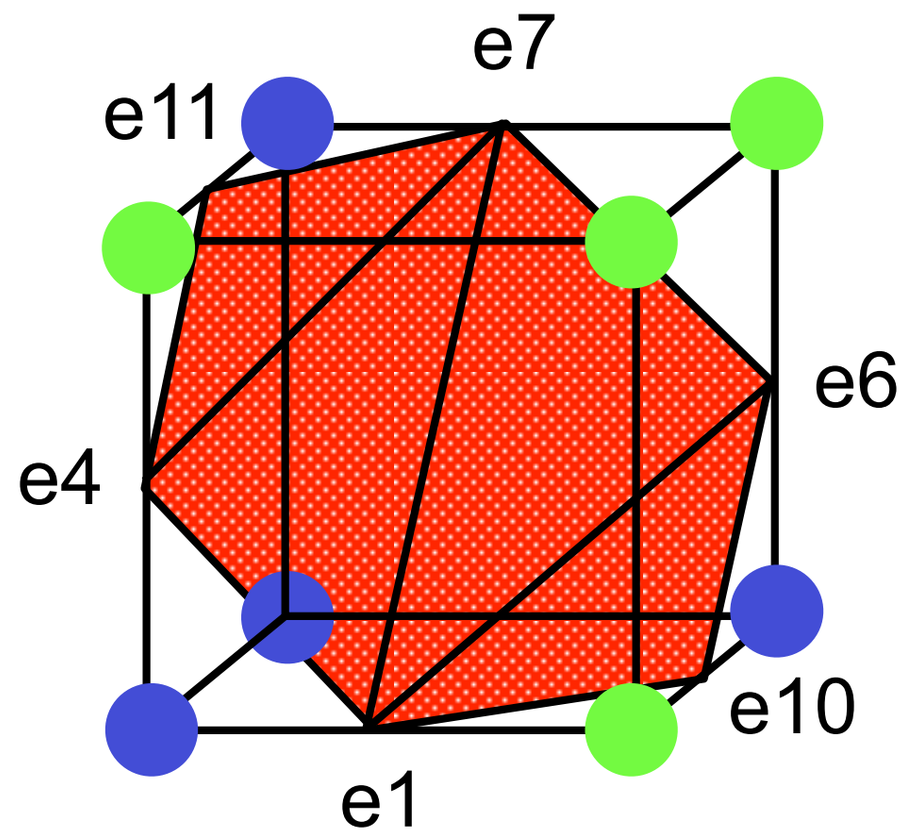
Index = 10110001

triangle 1 = e4,e7,e11

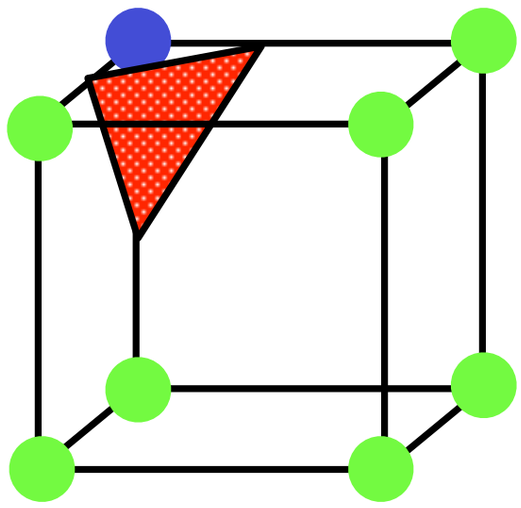
triangle 2 = e1, e7, e4

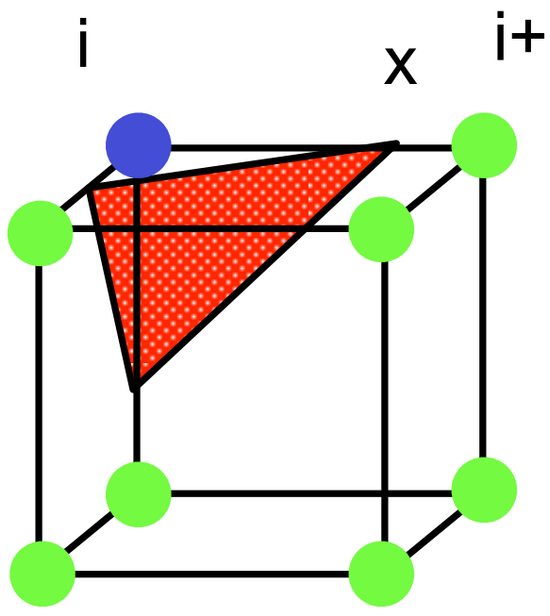
triangle 3 = e1, e6, e7

triangle 4 = e1, e10, e6



- Step 5: For each triangle edge, find the vertex location along the edge using linear interpolation of the voxel values



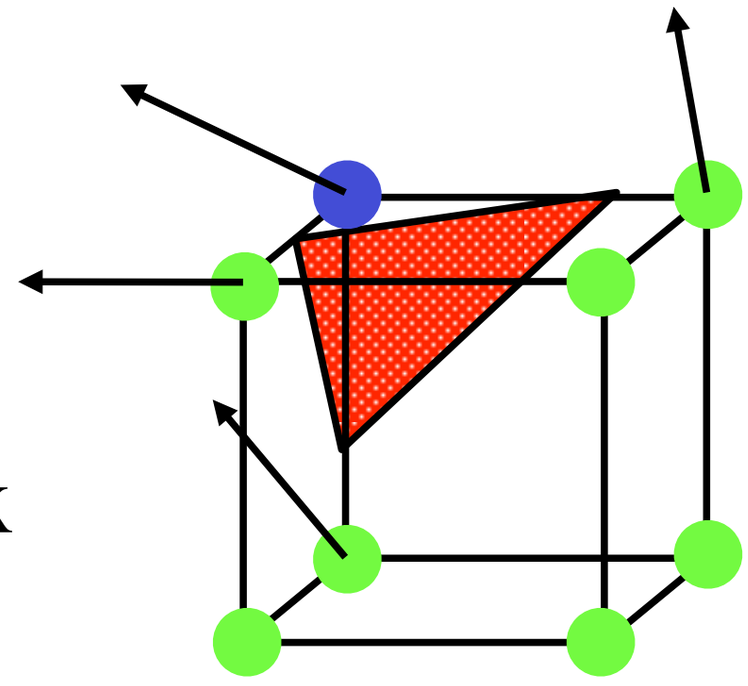


T=5

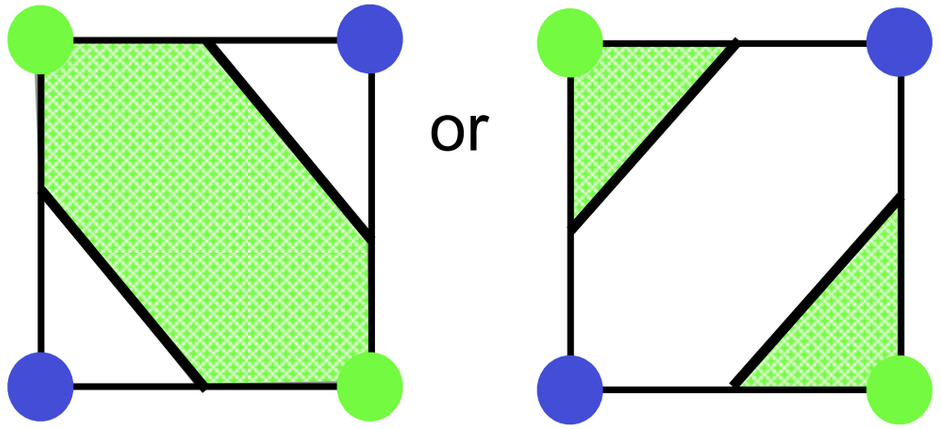
T=8

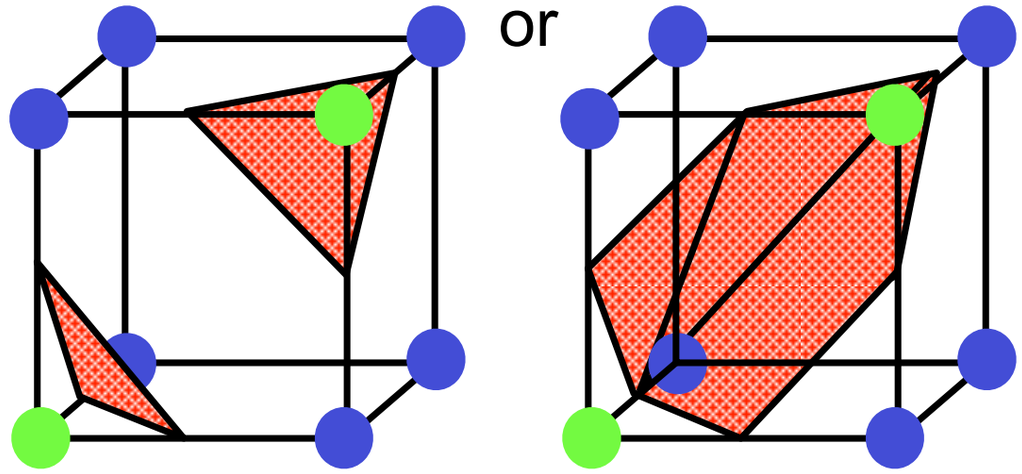
$$x = i + \left(\frac{T - v[i]}{v[i+1] - v[i]} \right)$$

- Step 6: Calculate the normal at each cube vertex (central differences)
  - $G_x = V_{x+1,y,z} - V_{x-1,y,z}$
    $G_y = V_{x,y+1,z} - V_{x,y-1,z}$
    $G_z = V_{x,y,z+1} - V_{x,y,z-1}$
  - Use linear interpolation to compute the polygon vertex normal (of the isosurface)

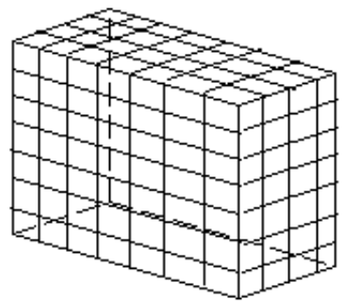


- Step 7: Consider ambiguous cases
  - Ambiguous cases: 3, 6, 7, 10, 12, 13
  - Adjacent vertices: different states
  - Diagonal vertices: same state
  - Resolution: choose one case (the right one!)

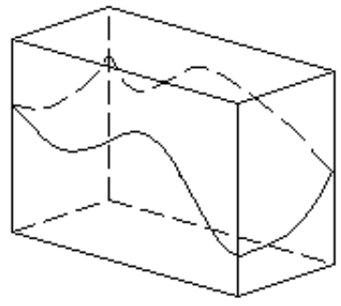




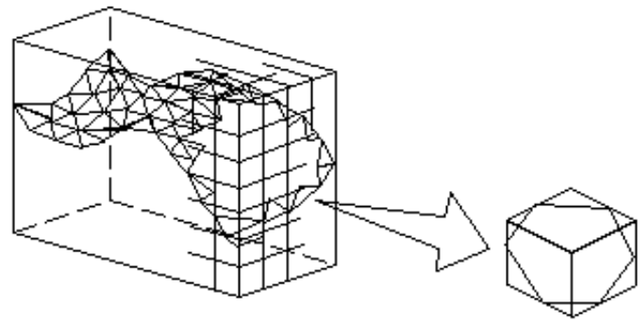
- Summary
  - 256 Cases
  - Reduce to 15 cases by symmetry
  - Ambiguity in cases
    3, 6, 7, 10, 12, 13
  - Causes holes if arbitrary choices are made
- Up to 5 triangles per cube
- Several isosurfaces
  - Run MC several times
  - Semi-transparency requires spatial sorting



(a) Volume data



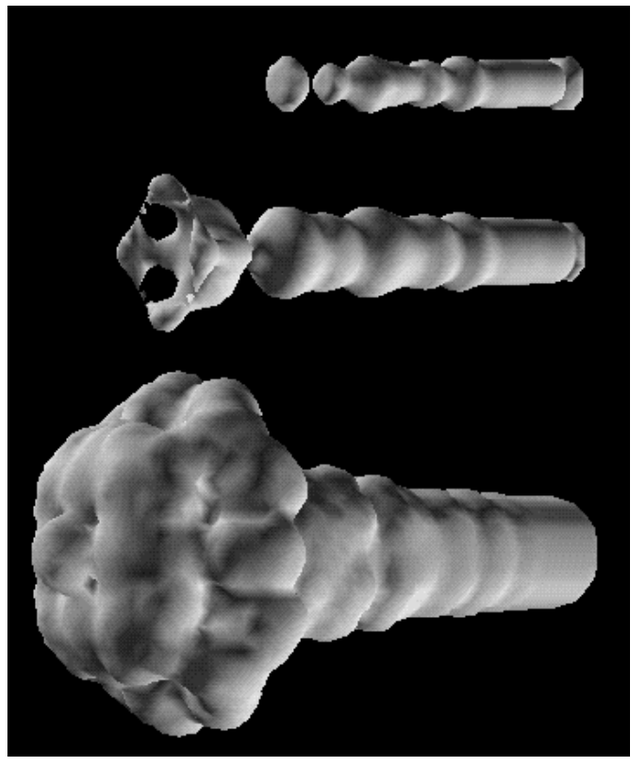
(b) Isosurface
S = f(x,y,z)



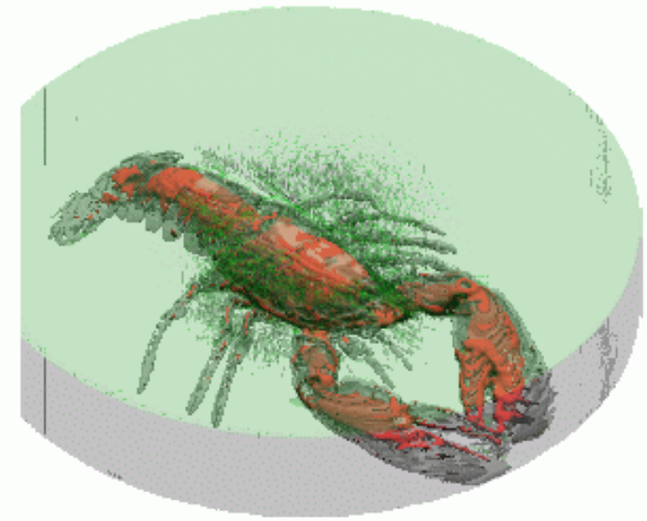
(c) Polygonal Apploximation

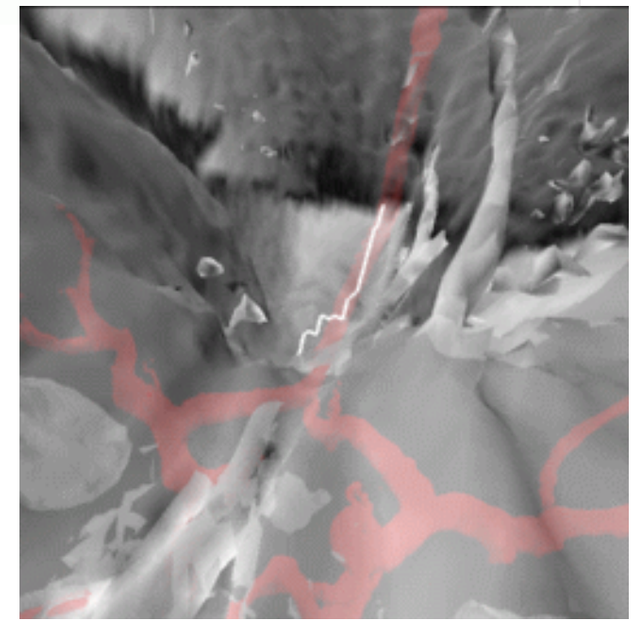
- Examples

1 Isosurface



3 Isosurfaces





2 Isosurfaces

Challenges

- Ambiguities
- Requires examining every voxel
  - Can be fixed? How?
- Isovalue selection: what is a good isovalue?

Improving the Robustness and Accuracy of the Marching Cubes Algorithm for Isosurfacing

Adriano Lopes and Ken Brodlie

Abstract—This paper proposes a modification of the Marching Cubes algorithm for isosurfacing, with the intent of improving the representation of the surface in the interior of each grid cell. Our objective is to create a representation which correctly models the topology of the trilinear interpolant within the cell and which is robust under perturbations of the data and threshold value. To achieve this, we identify a small number of key points in the cell interior that are critical to the surface definition. This allows us to efficiently represent the different topologies that can occur, including the possibility of "tunnels." The representation is robust in the sense that the surface is visually continuous as the data and threshold change in value. Each interior point lies on the isosurface. Finally, a major feature of our new approach is the systematic method of triangulating the polygon in the cell interior.

Index Terms—Isosurface, marching cubes, robustness, accuracy, trilinear interpolation.

1 Introduction

The accurate representation of an isosurface has proved to be one of the most interesting challenges in scientific visualization. Since the early work on the Marching Cubes

3D rectilinear mesh and the required isosurface level. The Marching Cubes algorithm generates the set of isosurfaces for that value by inspecting each cell in turn and constructing the corresponding portion, or portions, of surface. Thus, the problem can be reduced to the construction of the isosurface within a cube and, without loss of generality, we may take this to be the unit cube with corner at origin and we may take the isosurface threshold value as zero.

Since the data is only given at the vertices of the cube, we must model the data in the interior, and we shall take as our model the trilinear interpolant, $F(x, y, z)$. F can be written in terms of the vertex values F_{ijk}, $i, j, k = 0, 1$ as:

$$\begin{aligned} F(x,y,z) = {} & F_{000}\,(1-x)\,(1-y)\,(1-z) + \\ & F_{001}\,(1-x)\,(1-y)\,z + F_{010}\,(1-x)\,y\,(1-z) + \\ & F_{011}\,(1-x)\,y\,z + F_{100}\,x\,(1-y)\,(1-z) + \\ & F_{101}\,x\,(1-y)\,z + F_{110}\,x\,y\,(1-z) + F_{111}\,x\,y\,z\,. \end{aligned} \tag{1}$$

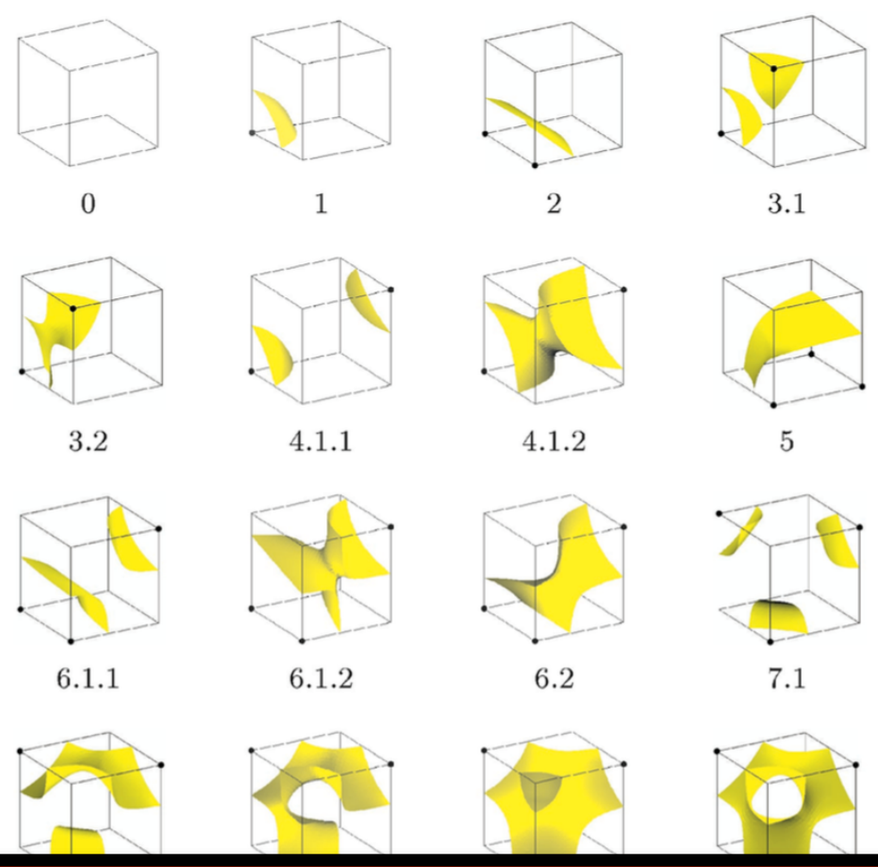
An alternative representation of (1), which we shall use in this paper, is

accurately and efficiently.

The original Marching Cubes algorithm identifies 256 configurations for the cube, depending on whether each of the eight vertices is positive or negative. Any edge with endpoints of opposite sign is intersected by the isosurface and inverse linear interpolation yields an estimate of the intersection point. The set of intersection points can be triangulated to yield an approximation to the isosurface within the cube. In fact, Lorensen and Cline showed that, with rotational symmetry and complementarity (switching positive and negative vertices), the 256 configurations can be reduced to a set of 15 cases. A further case can be removed by reflectional symmetry, leaving a canonical set of 14 cases. These are shown in Fig. 1.

Since its original conception, the Marching Cubes algorithm has been the subject of much further research to improve its quality of surface representation and its performance on large data sets. It is the quality issue with which we are concerned in this paper: the quality of representing the trilinear interpolating surface. The key contributions on which we build have been by Nielson and Hamann [2], Natarajan [3], Chernyaev [4], and Cignoni et al. [5].

Nielson and Hamann [2] point out that there is an ambiguity in the face of a cube when all four edges of the face are intersected. The triangulation has to choose which pairs of intersections to connect. An inconsistent strategy can lead to "holes" when surfaces in adjacent cells use



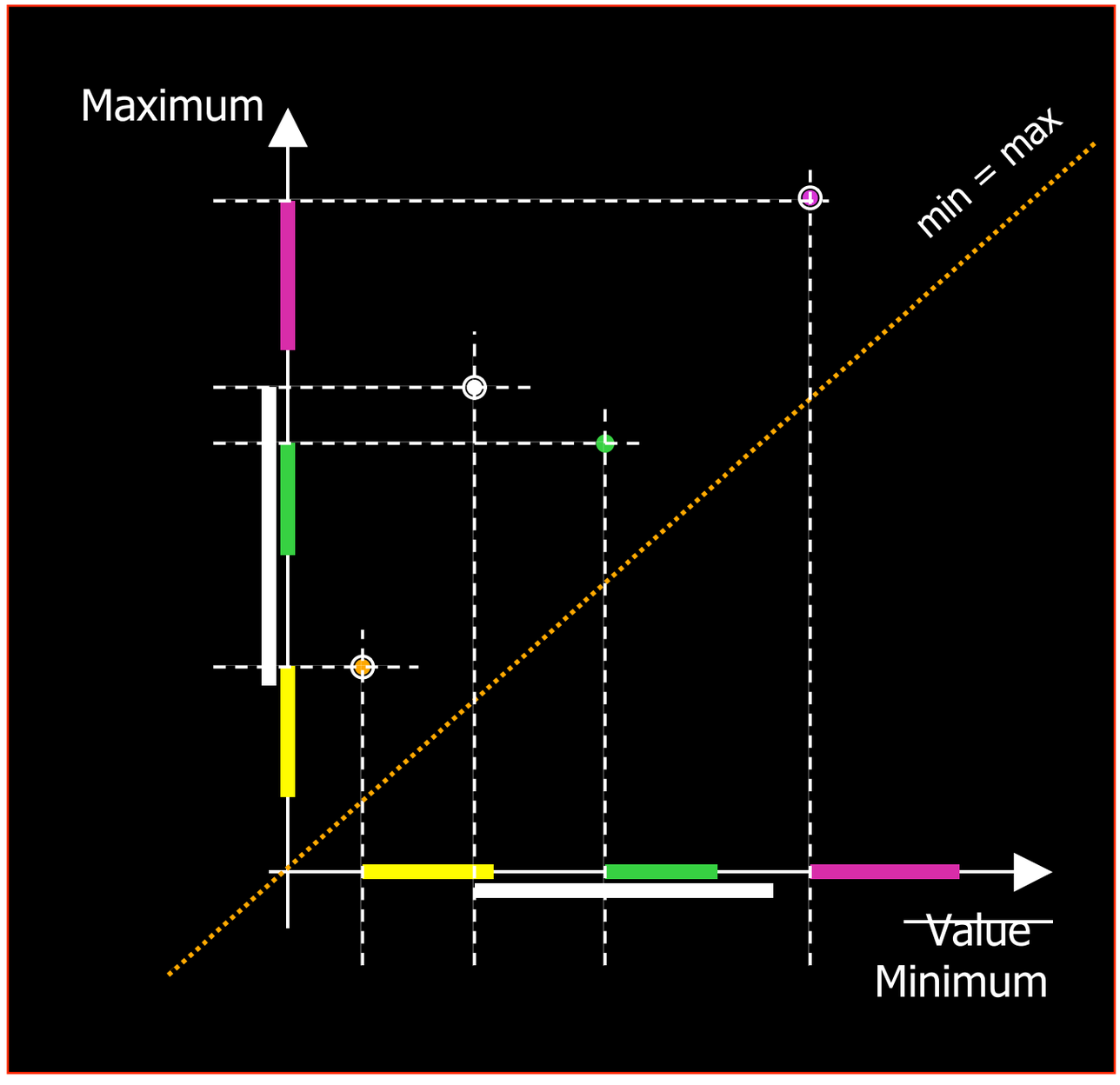
- *A. Lopes was with the Department of Mathematics of FST/CIS, University of Coimbra, Portugal. He is currently with the Department of Informatics, New University of Lisbon, Portugal. E-mail: alopes@di.fct.unl.pt.*
- *K. Brodlie is with the School of Computing, University of Leeds, United Kingdom. E-mail: kwb@comp.leeds.ac.uk.*

Manuscript received 7 Jan. 2000; revised 19 Jan. 2001; accepted 28 Aug. 2001. For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 111190.

Challenge: Examining Fewer Voxels

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



Lec18 Reading

- None required, but see optional reading for some additional background

Reminder
Assignment 04

Assigned: Monday, March 13
Due: Monday, March 27, 4:59:59 pm

Reminder
Project Milestone 02

Assigned: Wednesday, February 22
Due: Wednesday, March 29, 4:59:59 pm