Scalar Visualization

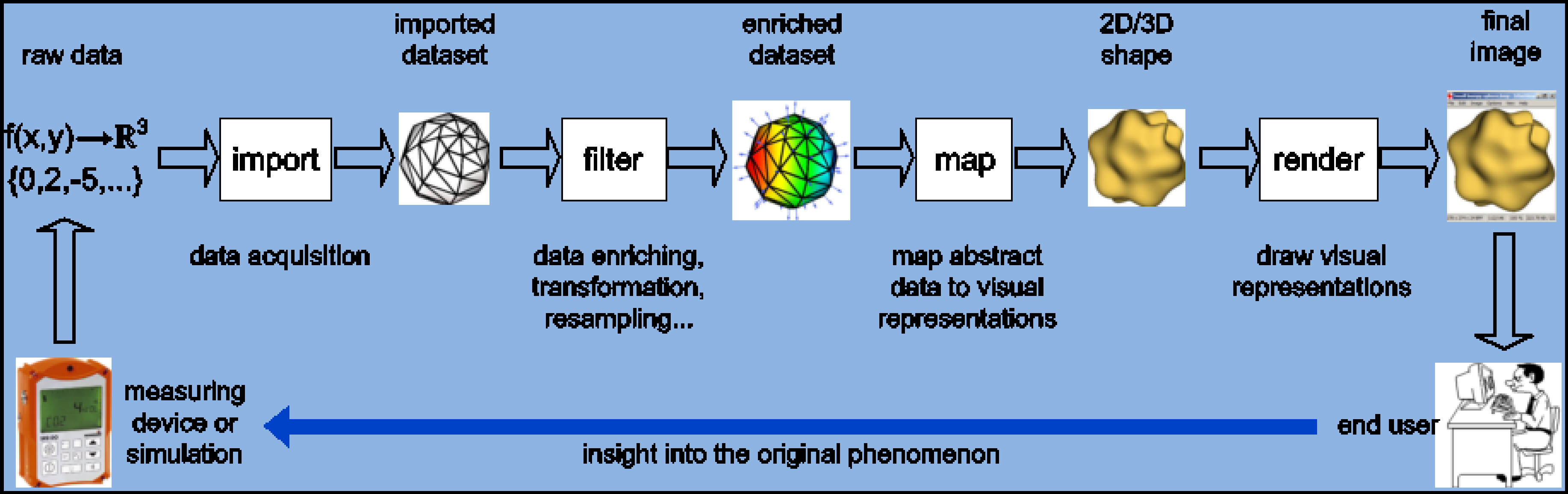
outline

Visualizing scalar data

Popular scalar visualization techniques

- Color mapping
- Contouring
- Height plots

Recap of Chap 4: Visualization Pipeline



1. Data Importing
2. Data Filtering
3. Data Mapping
4. Date Rendering

Scalar Function

$f: R \rightarrow R$

1-D, histogram

$f: R^2 \rightarrow R$

2-D, color mapping, contouring, height plot

$f: R^3 \rightarrow R$

3-D, isosurface, slicing, volume visualization

Visualizing scalar data

Popular scalar visualization techniques

- Color mapping
- Contouring
- Height plots

outline

Color Mapping

- Color mapping maps scalar data to colors. The scalar mapping is implemented by indexing into a color lookup table. Scalar values then serve as indices into this lookup table
- Color look-up table

  Associate a specific color with every scalar value

  The geometry of **Dv** is the same as **D**

$$C = \{c_i\}_{i=1,2,\dots N}$$

where

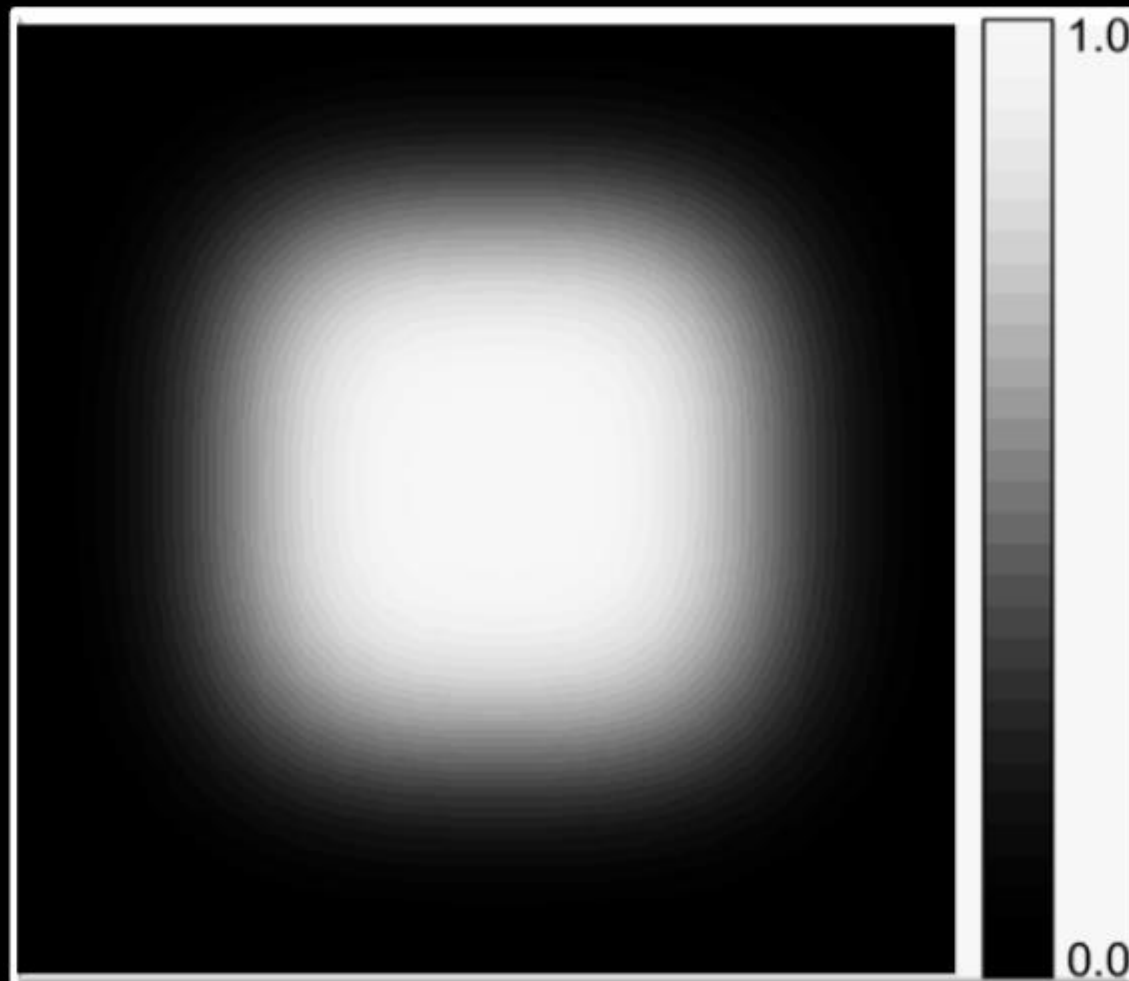
$$c_i = c\left(\frac{(N\text{-}i)f_{\min} + i f_{\max}}{N}\right)$$

Luminance Colormap

- Use grayscale to represent scalar value

$$f = e^{-10(x^4+y^4)}$$

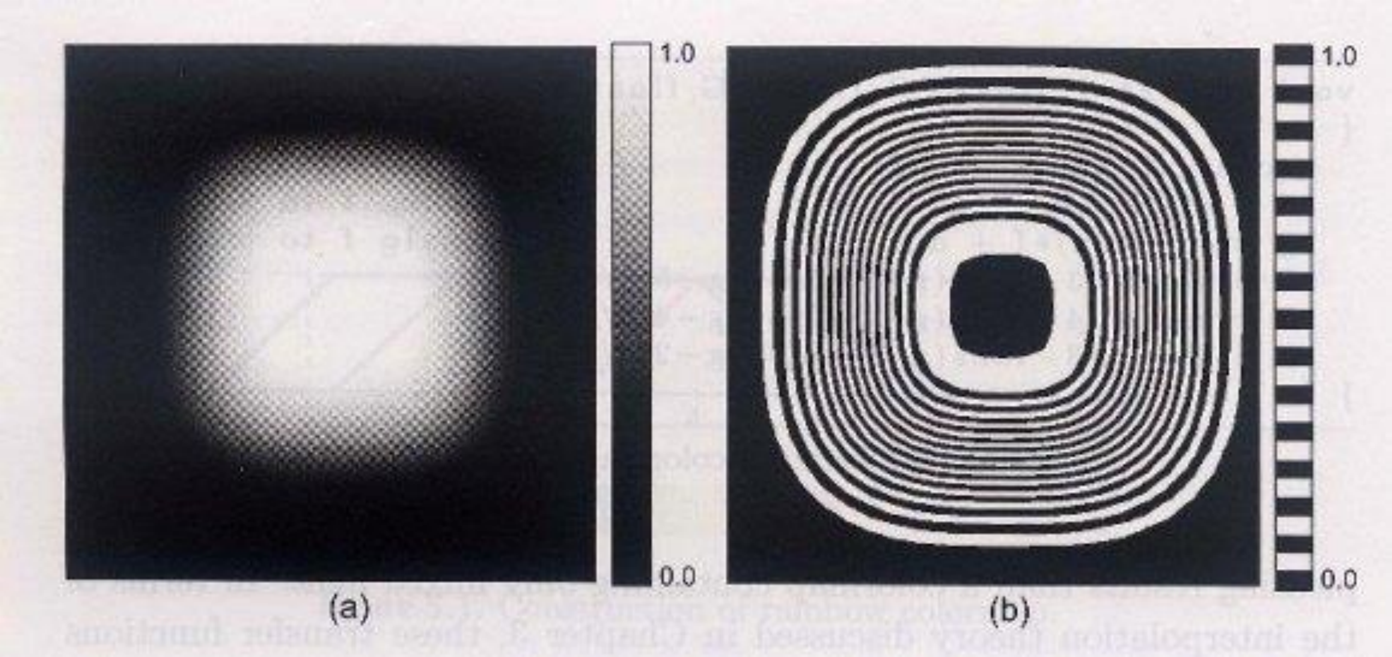
- Most scientific data (through measurement, observation, or simulation) are intrinsically grayscale, not color



Luminance Colormap

Legend

Luminance Colormap

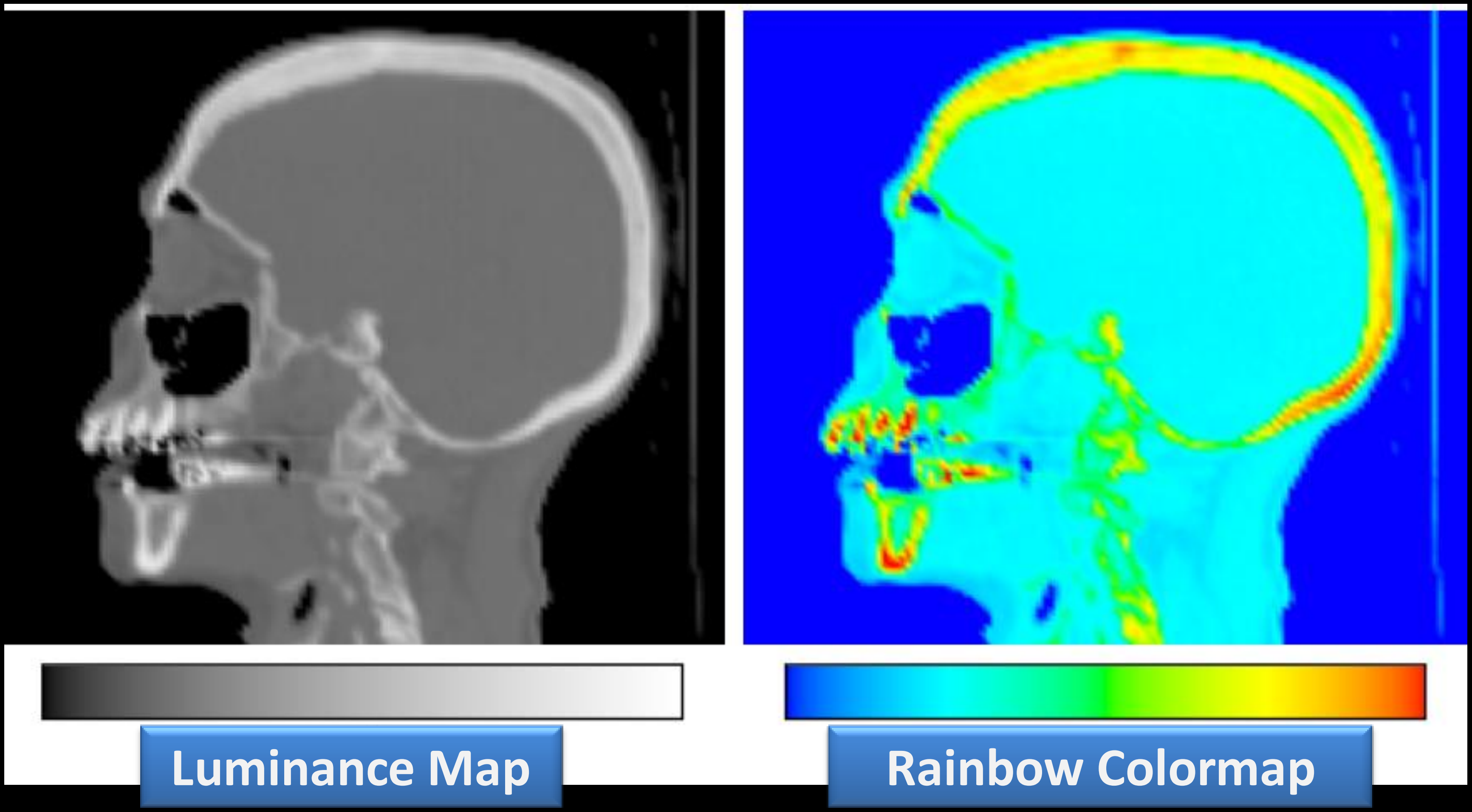


(a) a luminance *colormap* and (b) a zebra *colormap*

(b) The luminance colormap shows absolute values, whereas the zebra colormap emphasizes rapid value variations.

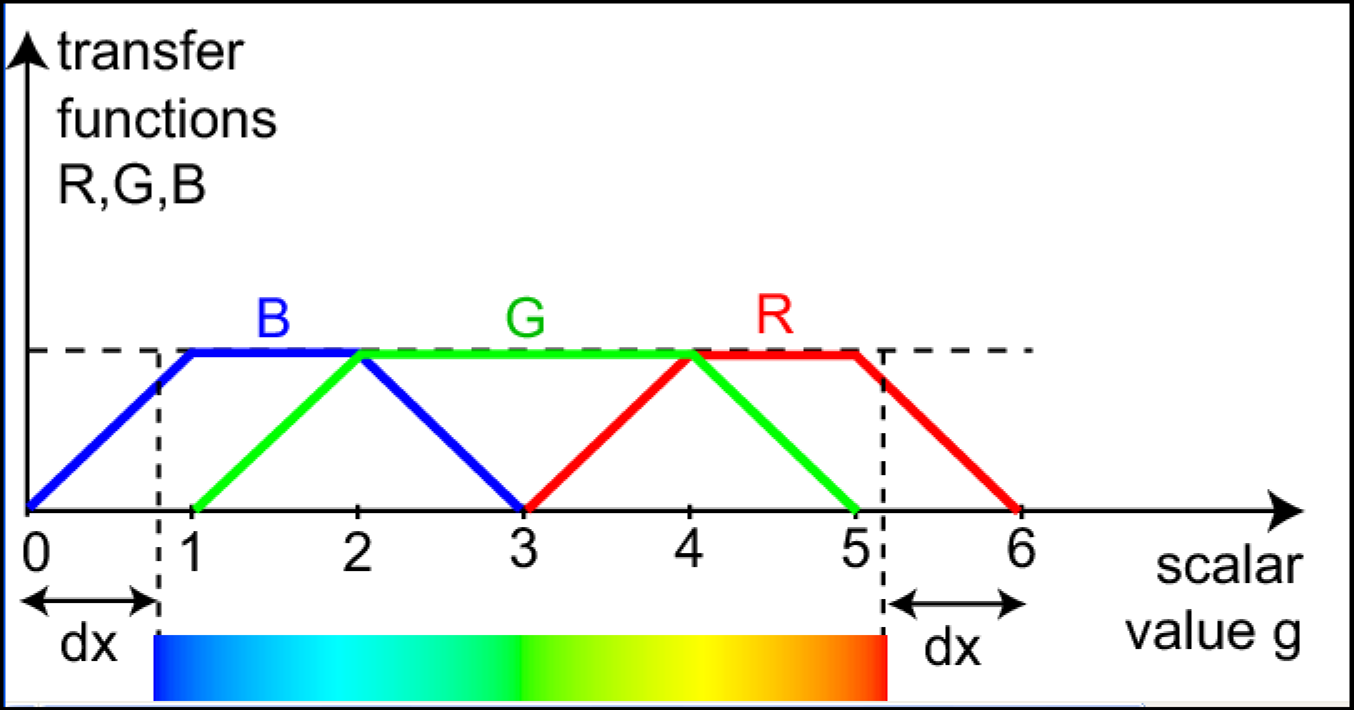
Rainbow Colormap

- Red: high value; Blue: low value
- Medical visualization with luminance and rainbow colormaps



Rainbow Colormap

- Construction



- $f < dx$: R=0, G=0, B=1
- $f = 2$: R=0, G=1, B=1
- $f = 3$: R=0, G=1, B=0
- $f = 4$: R=1, G=1, B=0
- $f > 6 - dx$: R=1, G=0, B=0

Colormap: Designing Issues

- Choose right color map for correct perception
  - Grayscale: good in most cases
  - Rainbow: e.g., temperature map
  - Rainbow + white: e.g., landscape
    - Blue: sea, lowest
    - Green: fields
    - Brown: mountains
    - White: mountain peaks, highest

Designing effective colormaps

- Attract user to certain value ranges or individual values
  - Colormap uses particularly salient colors
  - Colormap can be influenced by:
    - Application
    - Domain-specific convention &Traditions

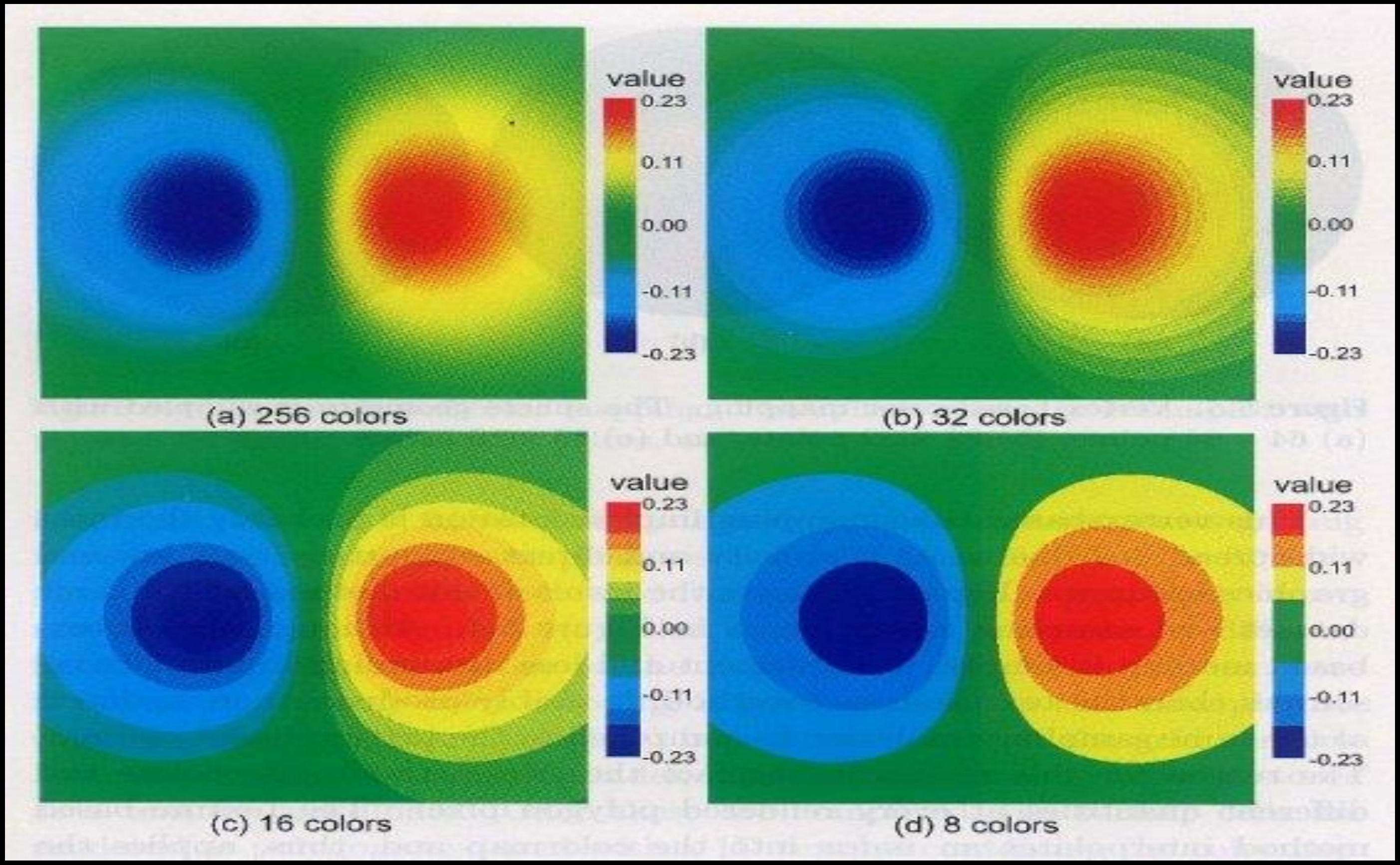
Designing effective colormaps

- Some application, emphasize the variation of the data
  - Colormap containing two or more alternating colors
- Many other colormap designs
  - Geographical application
  - Classical medical imaging

Designing effective colormaps

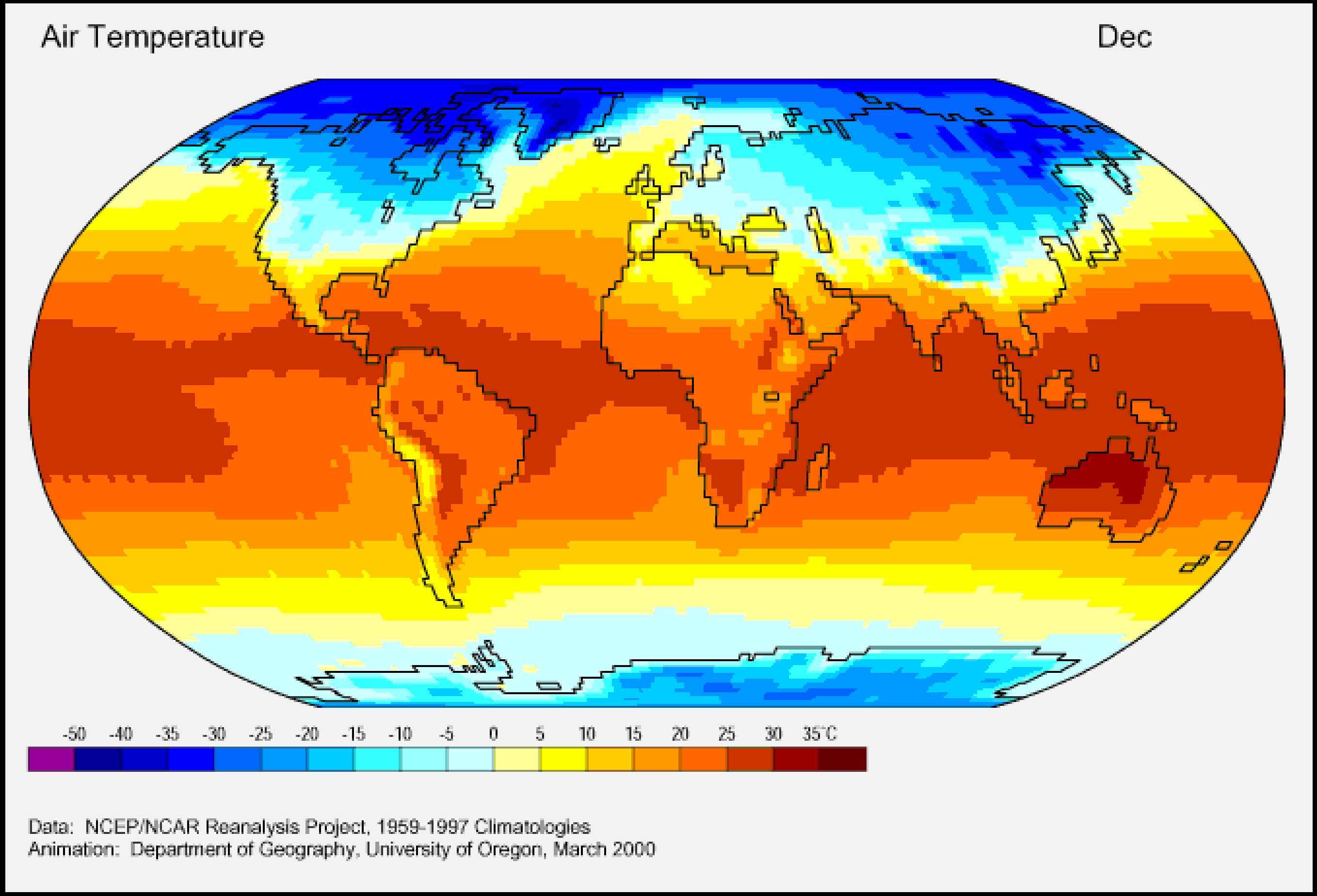
- The choice of the number of Color N
  - A small N: color banding effect, artifact
  - Typical scalar visualization applications use 64 to 256 different colors
- Other factors for Colormap
  - Geometric factors
  - User group
  - The medium used to present the visualization

Designing effective colormaps



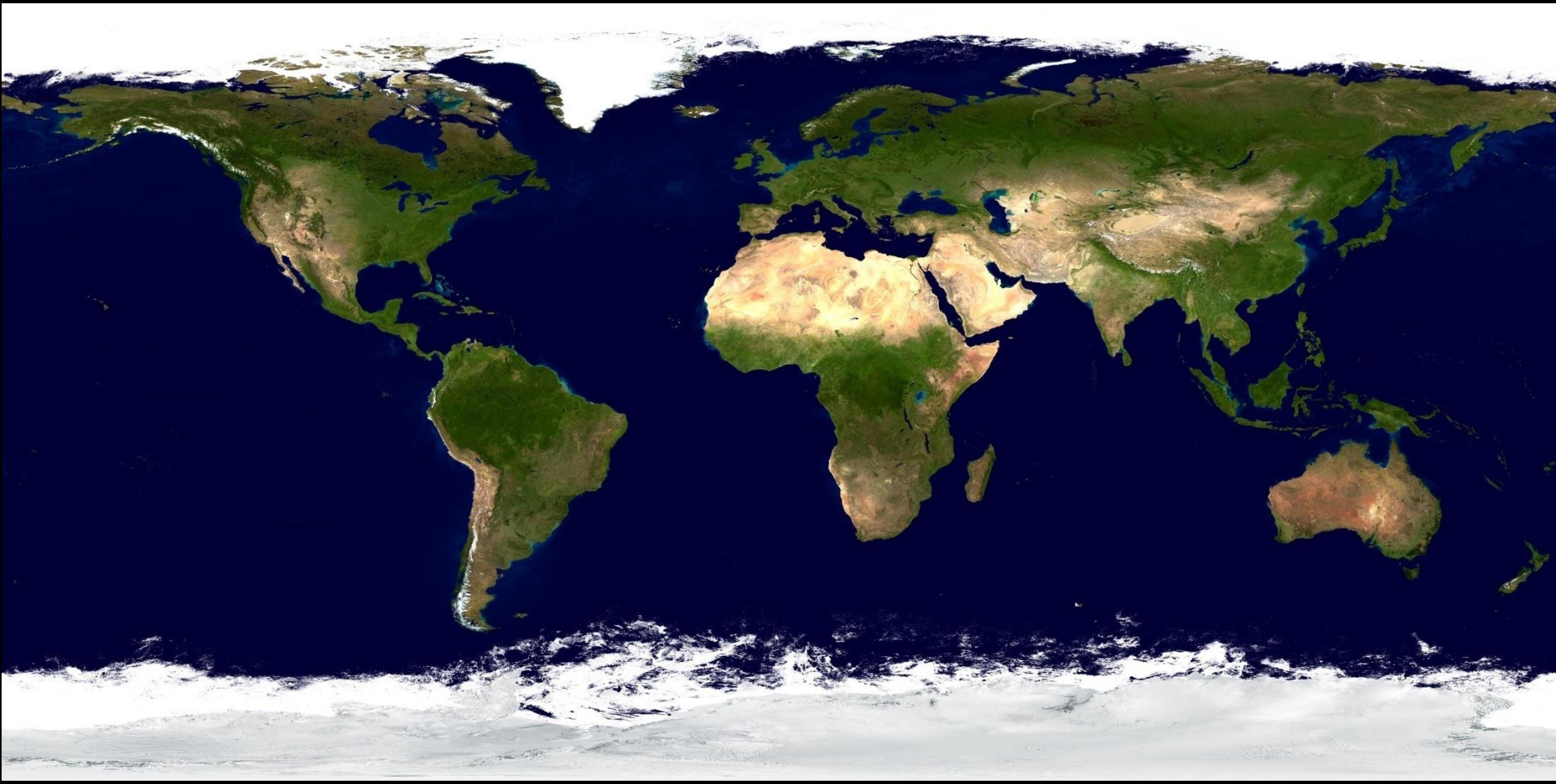
Color banding caused by a small number of colors in a look-up table.

Exp: Rainbow Colormap



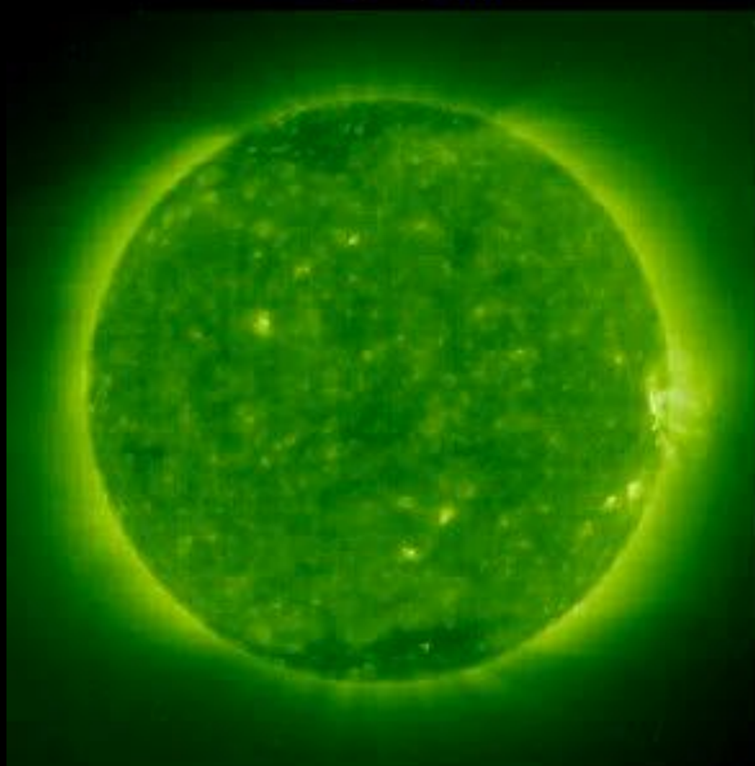
Global air temperature by month

Exp: Earth map

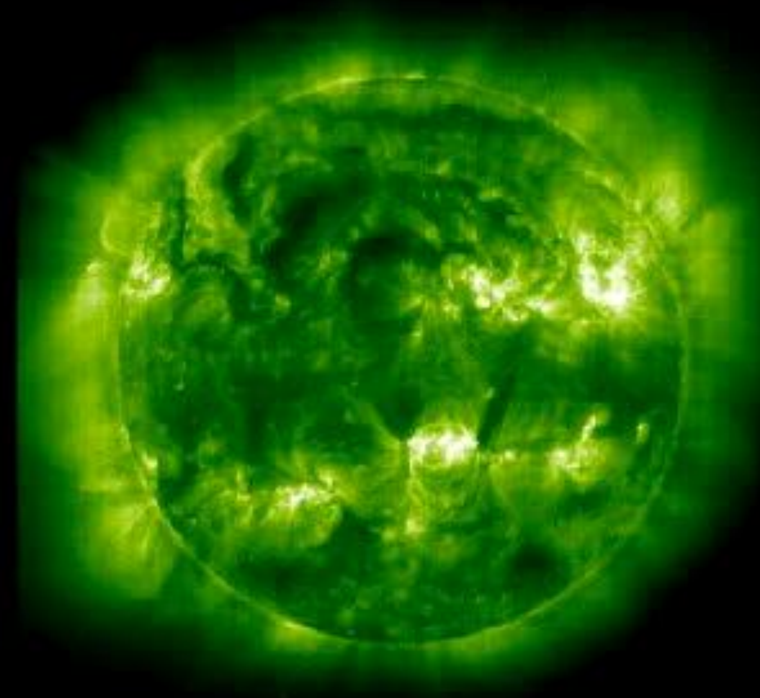


Exp: Sun in green-white colormap

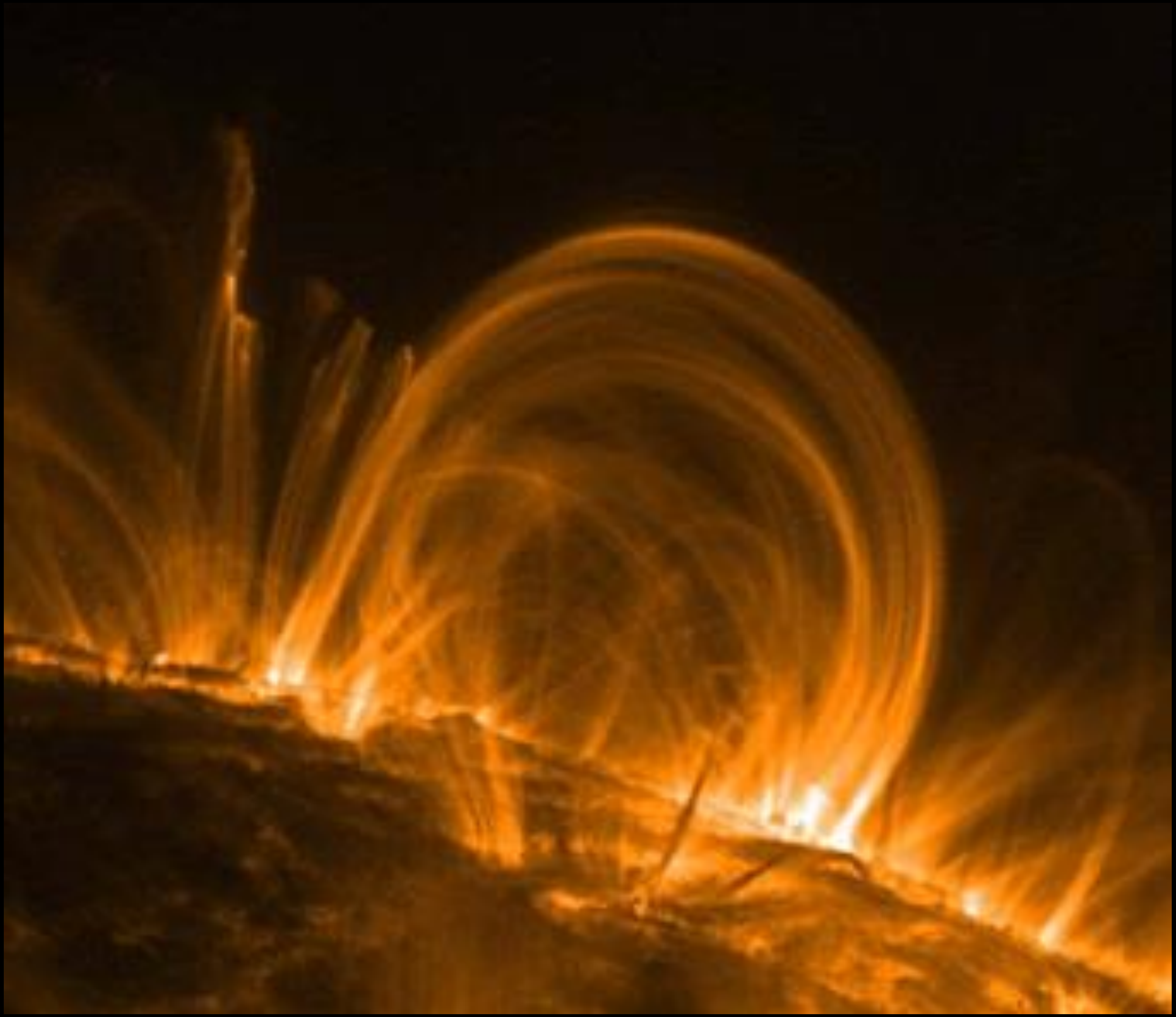
EIT 195 Å
Dec. 1996



EIT 195 Å
June 1999



Exp: Coronal loop



Summary

- Color mapping : generate color values from scalar values by
    - Colormap
    - Color transfer function
- Design issues for effective colormaps:
    - Knowledge of the application domain conventions
    - Typical data distribution
    - Visualization goals
    - General perception theory
    - Intended output devices
    - User preference

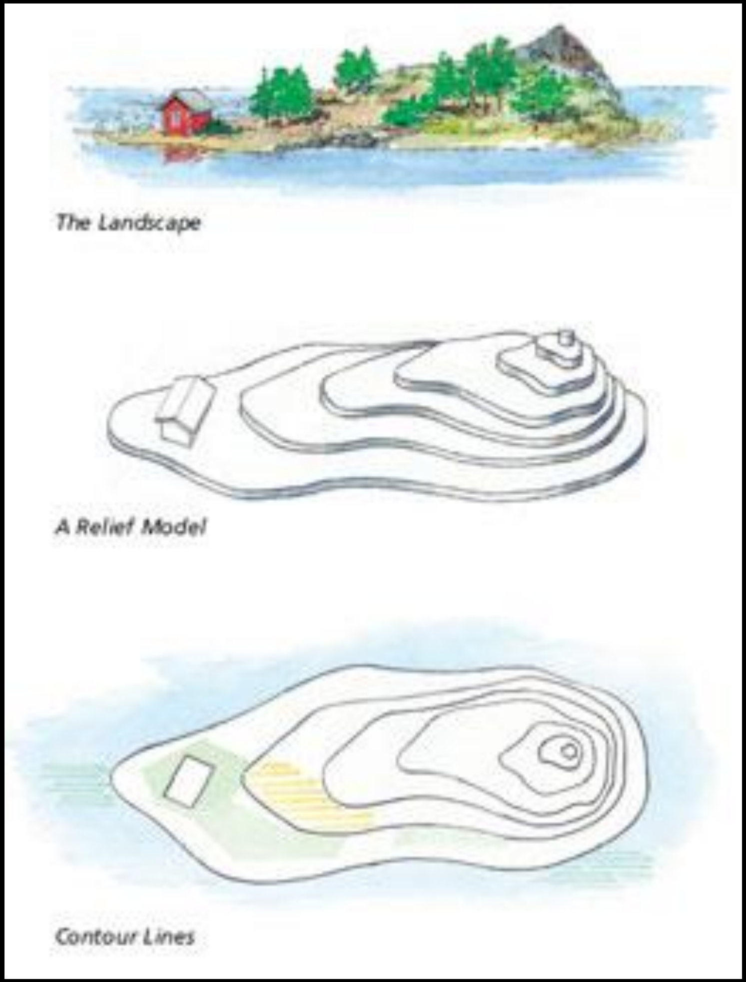
outline

Visualizing scalar data

Popular scalar visualization techniques

- Color mapping
- Contouring
- Height plots

Contouring



Cartograph

Contouring

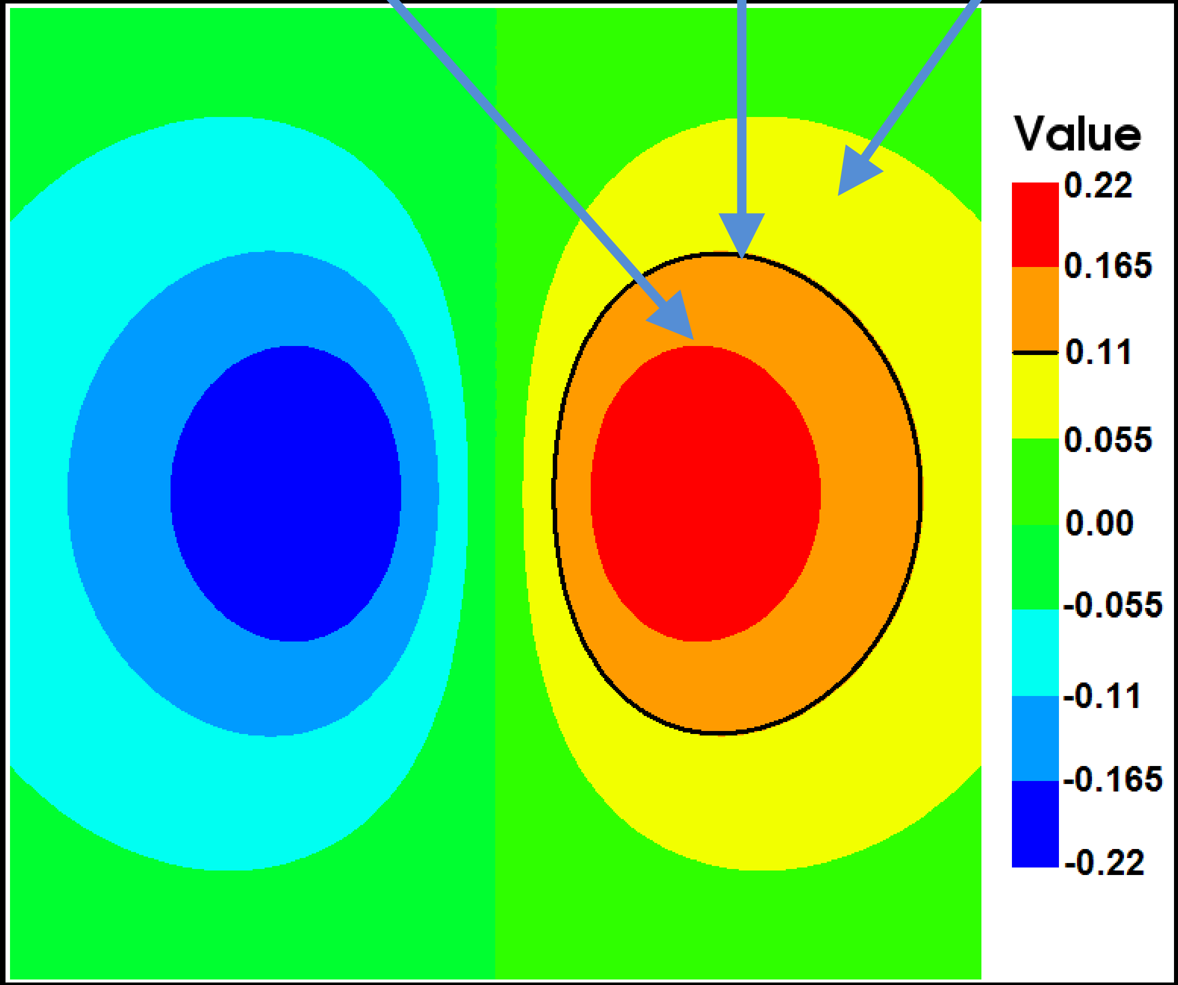
- A contour line C is defined as all points p in a dataset D that have the same scalar value, or isovalue s(p)=x

$$C(x) = \{p \in D \mid s(p) = x\}$$

- For 2D dataset, a contour line is called an isoline

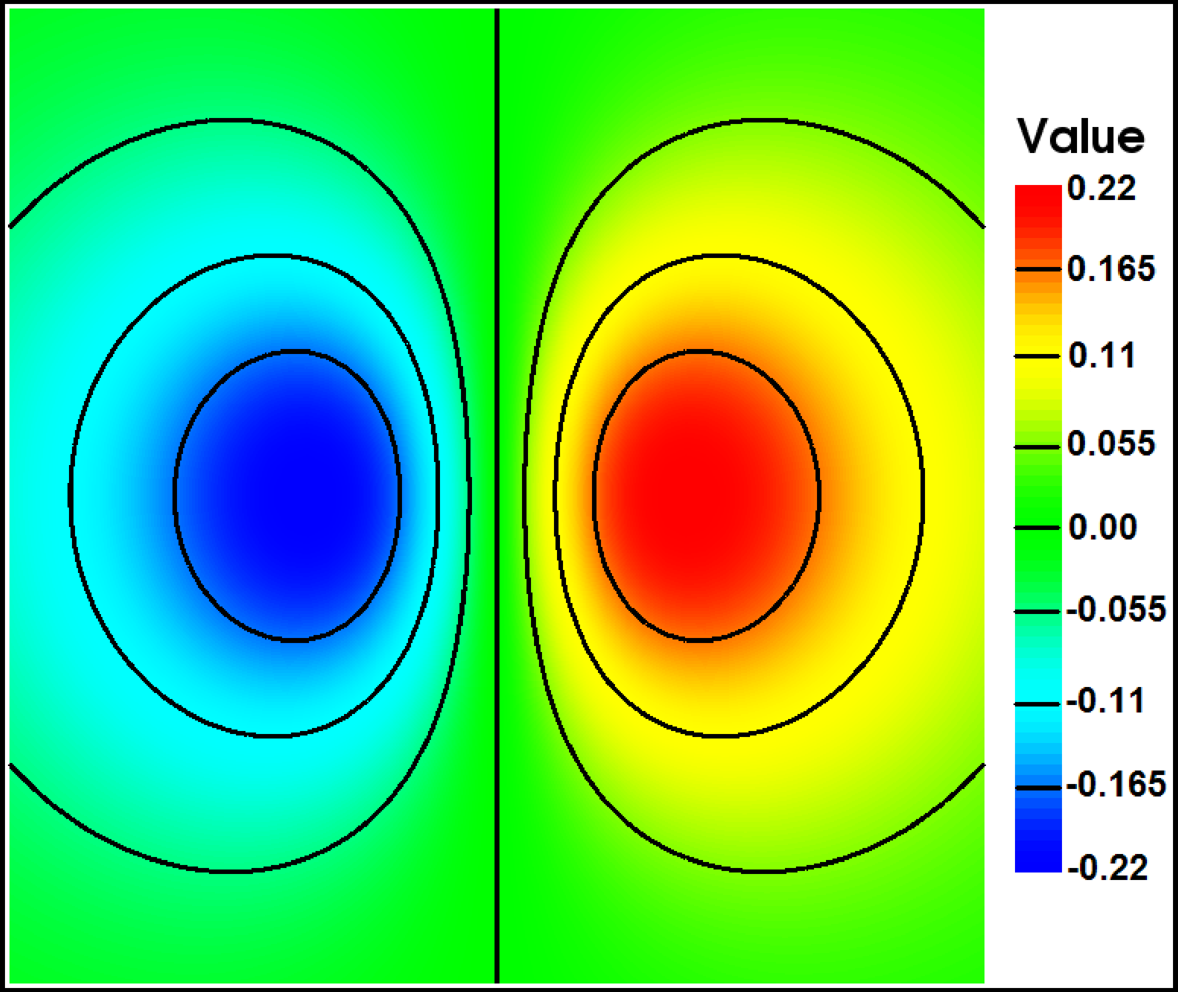
- For 3-D dataset, a contour is a 2-D surface, called isosurface

Contouring



Contouring and Color Banding

Contouring



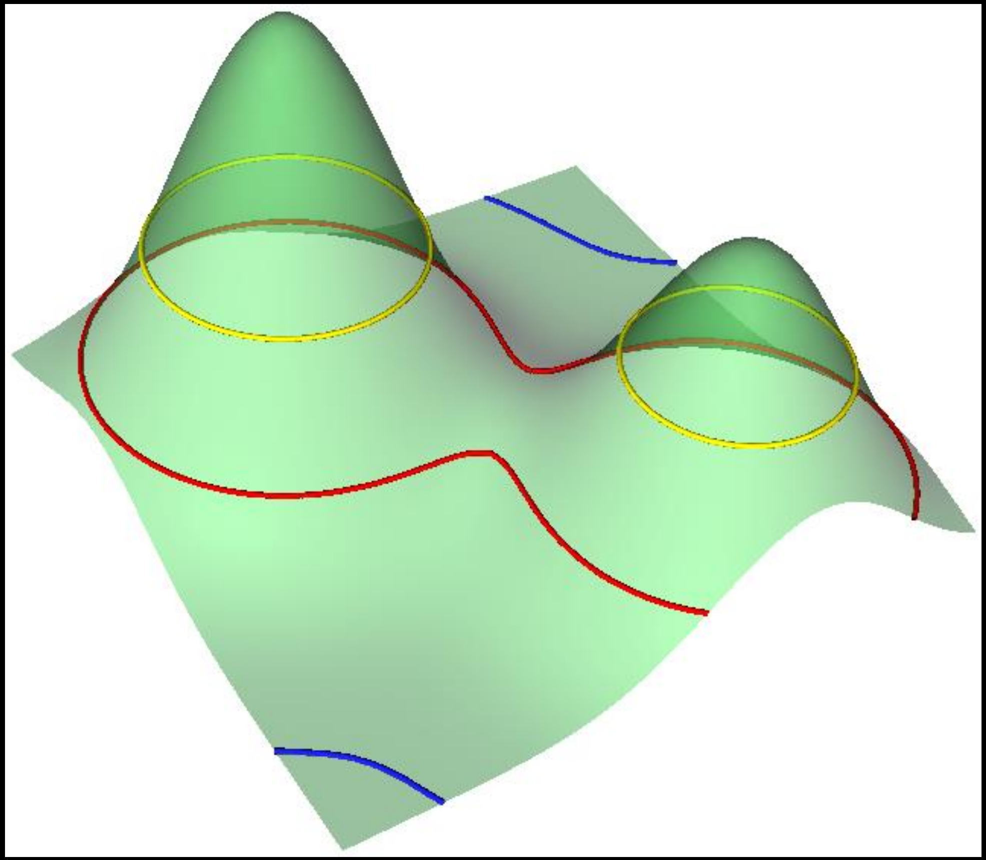
7 contour lines

Contouring and Colormapping:

Show (1) the smooth variation and (2) the specific values

Properties of Contours

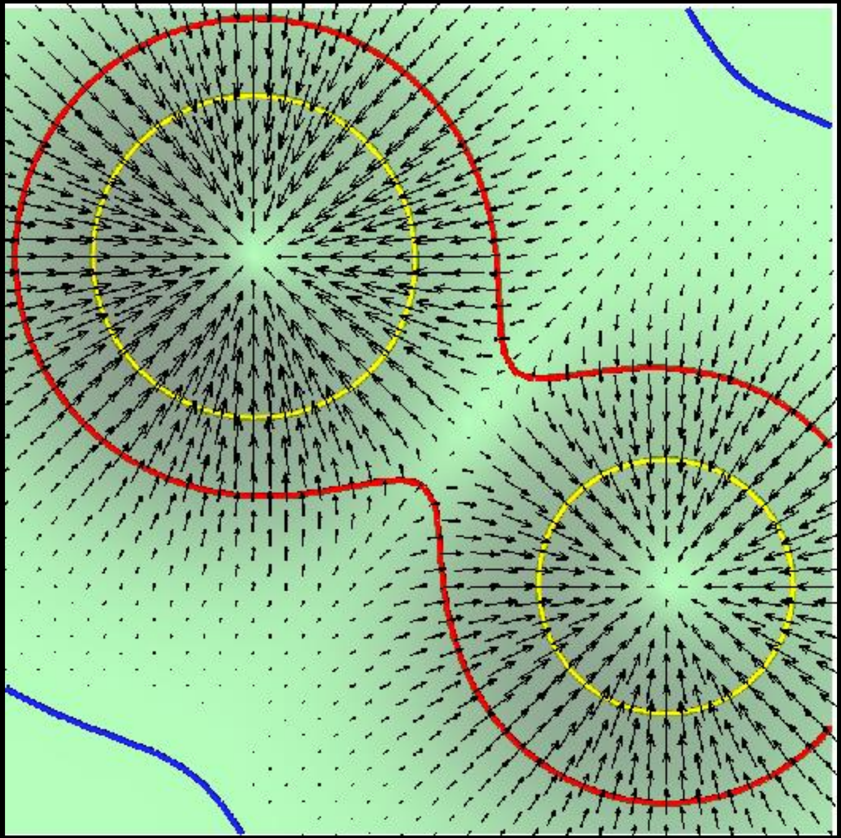
- Indicating specific values of interest
- In the height-plot, a contour line corresponds with the interaction of the graph with a horizontal plane of s value



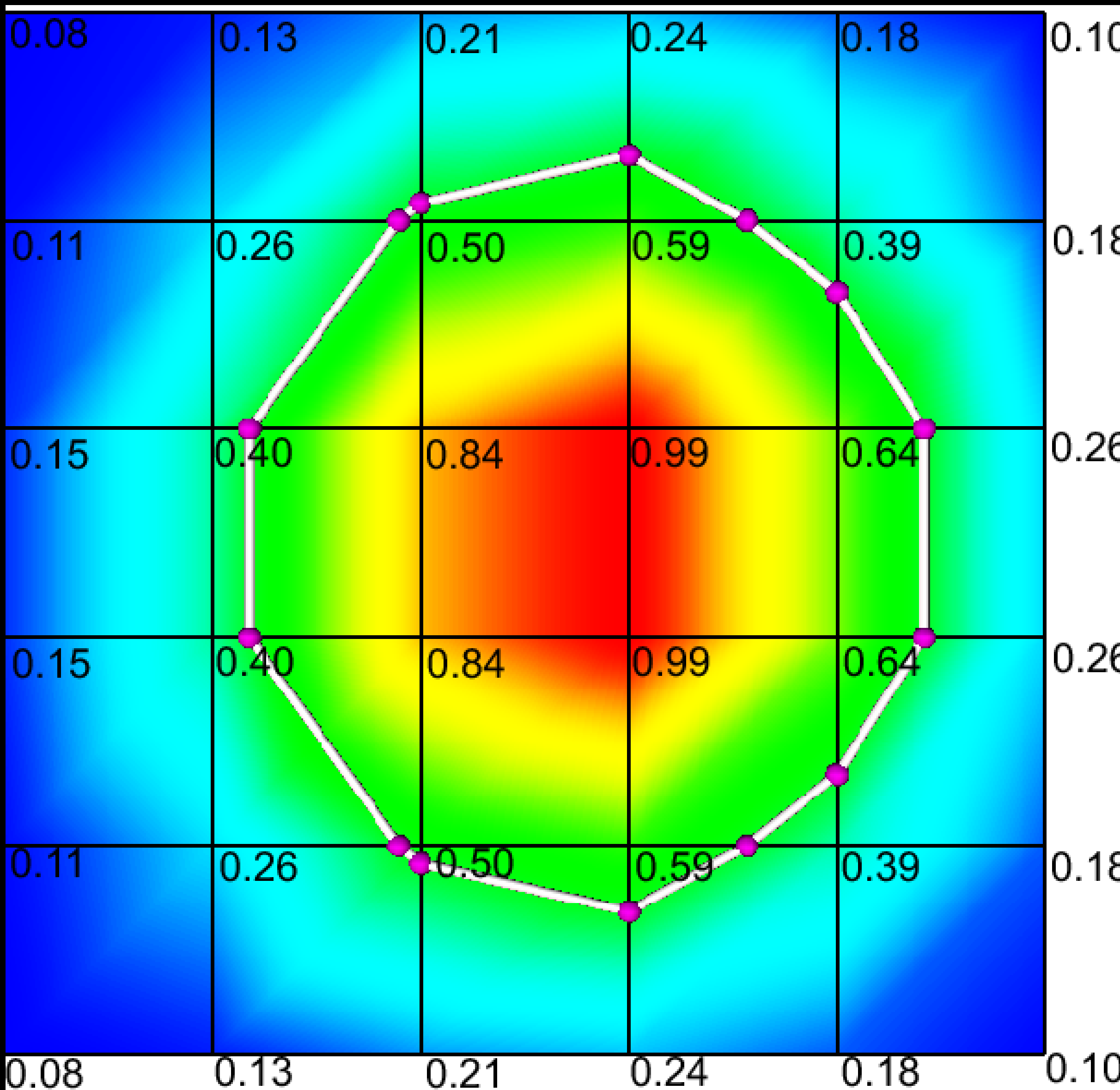
Properties of Contours

- The tangent to a contour line is the direction of the function's minimal (zero) variation
- The perpendicular to a contour line is the direction of the function's maximum variation: the **gradient**

Contour lines and Gradient vector



Constructing Contours



V=0.48

Finding line segments within cells

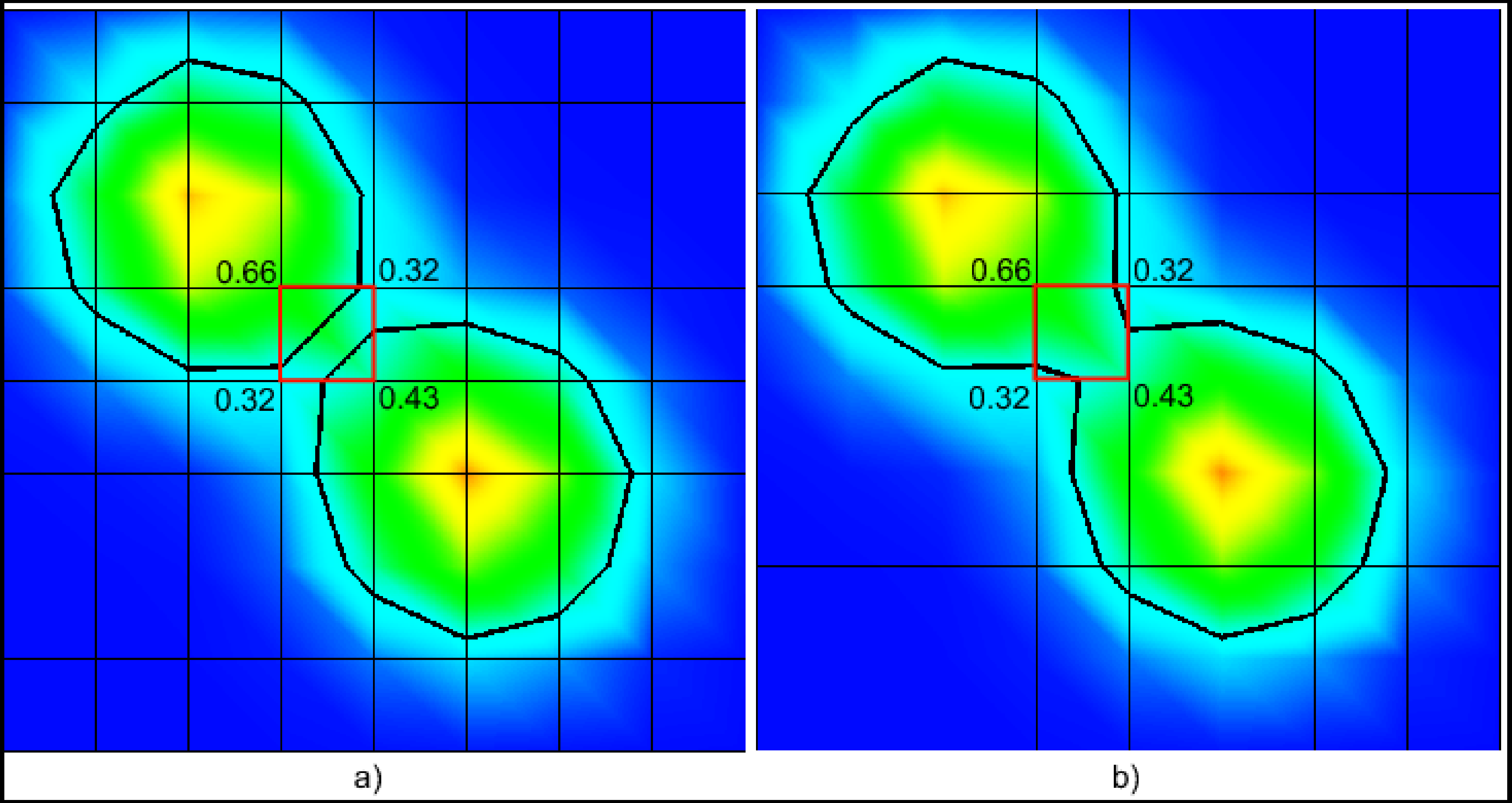
Constructing Contours

- For each cell, and then for each edge, test whether the isoline value v is between the attribute values of the two edge end points (v_i, v_j)
- If yes, the isoline intersects the edge at a point q, which uses linear interpolation

$$q = \frac{p_i(v_j - v) + p_j(v - v_i)}{v_j - v_i}$$

- For each cell, at least two points, and at most as many points as cell edges
- Use line segments to connect these edge-intersection points within a cell
- A contour line is a polyline.

Constructing Contours



V=0.37: 4 intersection points in a cell

-> Contour ambiguity

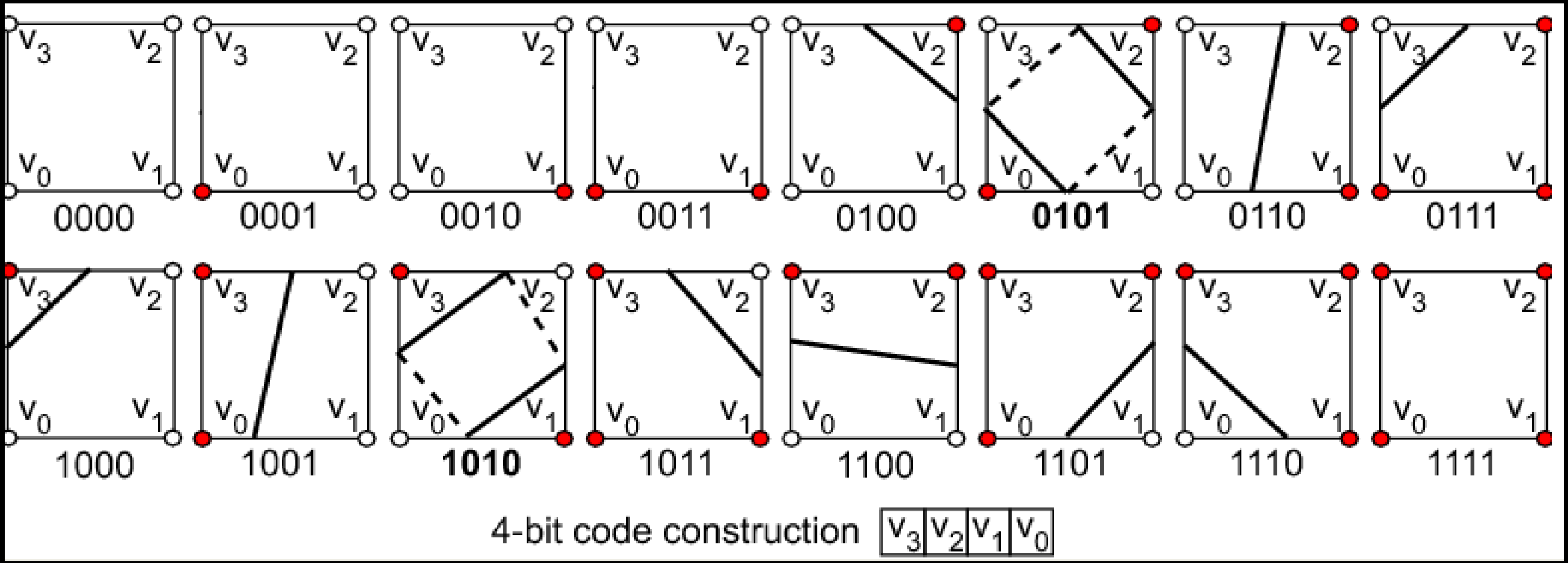
Contouring

- Contouring need
  - At least piecewise linear, C^1 dataset
  - The complexity of computing contours
- The most popular method
  - 2D: Marching Squares
  - 3D: Marching Cubes

Implementation: Marching Squares

- Determining the topological state of the current cell with respect to the isovalue v
  - Inside state (1): vertex attribute value is less than isovalue
  - Outside state (0): vertex attribute value is larger than isovalue
  - A quad cell: $(S_3S_2S_1S_0)$, $2^4=16$ possible states
    - (0001): first vertex inside, other vertices outside
- Use optimized code for the topological state to construct independent line segments for each cell
- Merge the coincident end points of line segments originating from neighboring grid cells that share an edge

Implementation: Marching Squares



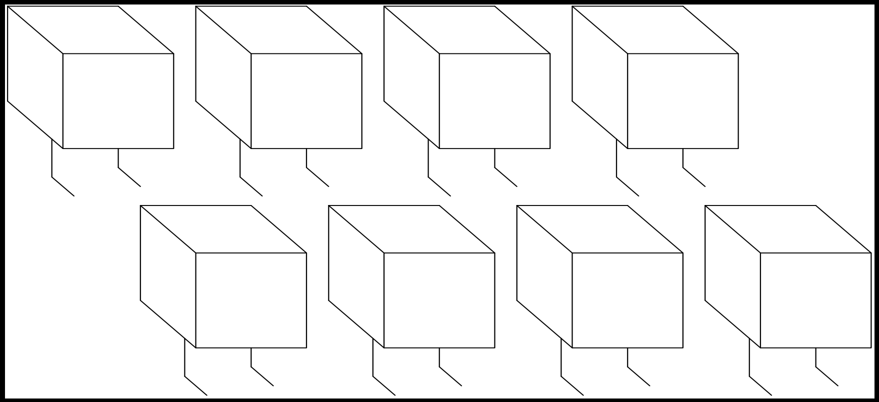
Topological State of a Quad Cell

Marching Cubes

- Similar to Marching Squares but 3D versus 2D
- $2^8 = 256$ different topological cases; reduced to only 15 by symmetry considerations
  - 16 topological states

- **Marching Cubes: A High Resolution 3D Surface Construction Algorithm**
  - *William E. Lorensen & Harvey E. Cline*
  - ACM SIGGRAPH 1987

What are Marching Cubes?



Marching Cubes is an algorithm which "creates triangle models of constant density surfaces from 3D scalar data."

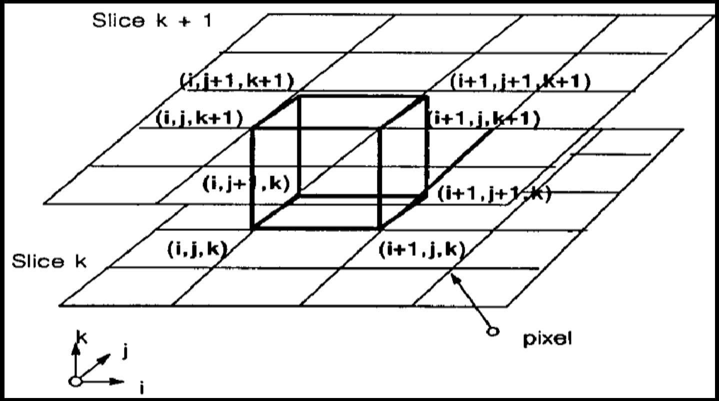
Marching Cubes Explained

- High resolution surface construction algorithm.
- Extracts surfaces from adjacent pairs of data slices using cubes.
- Cubes “march” through the pair of slices until the entire surface of both slices has been examined.

Marching Cubes Overview

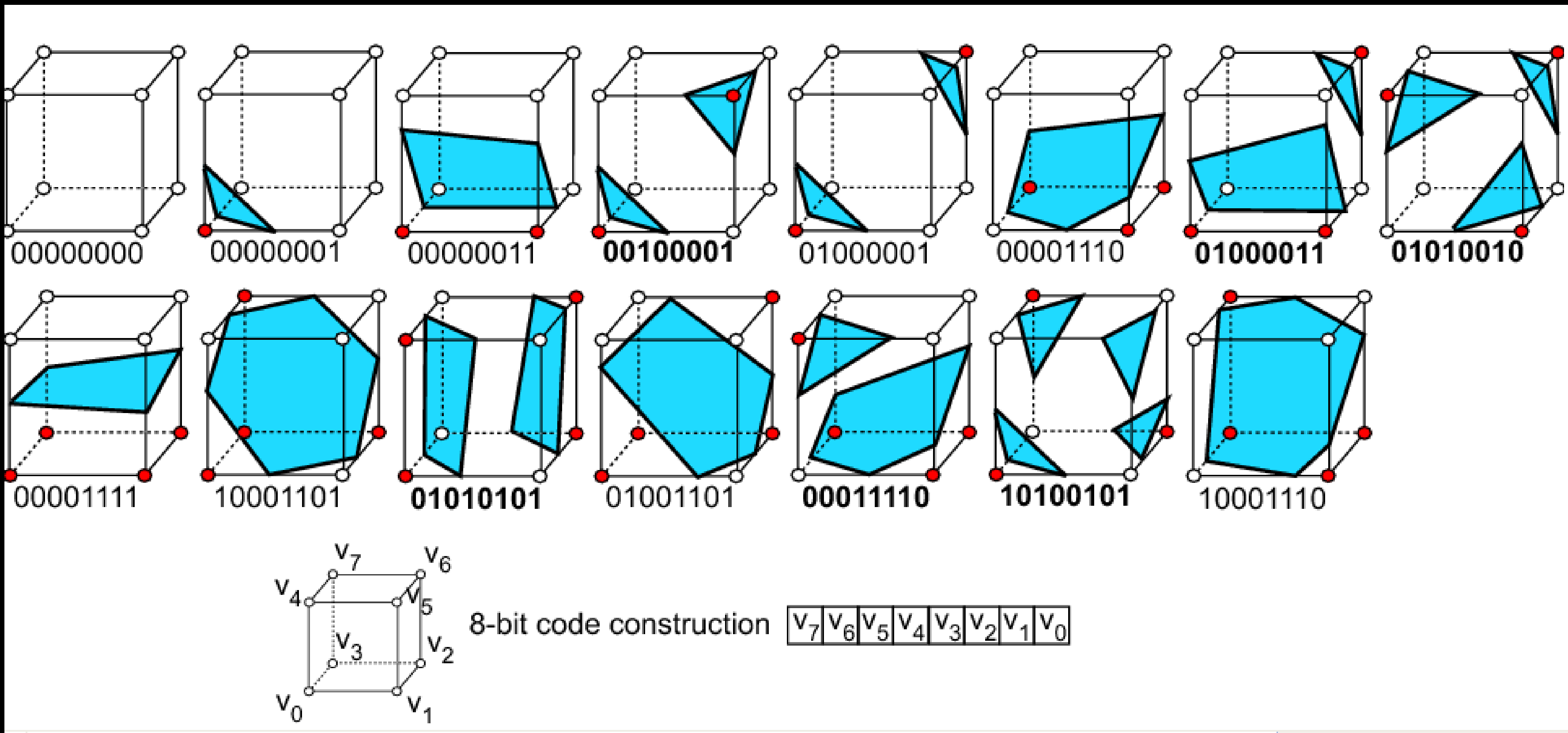
- Load slices.
- Create a cube from pixels on adjacent slices.
- Find vertices on the surfaces.
- Determine the intersection edges.
- Interpolate the edge intersections.
- Calculate vertex normals.
- Output triangles and normals.

How Are Cubes Constructed



- Uses identical squares of four pixels connected between adjacent slices.
- Each cube vertex is examined to see if it lies on or off of the surface.

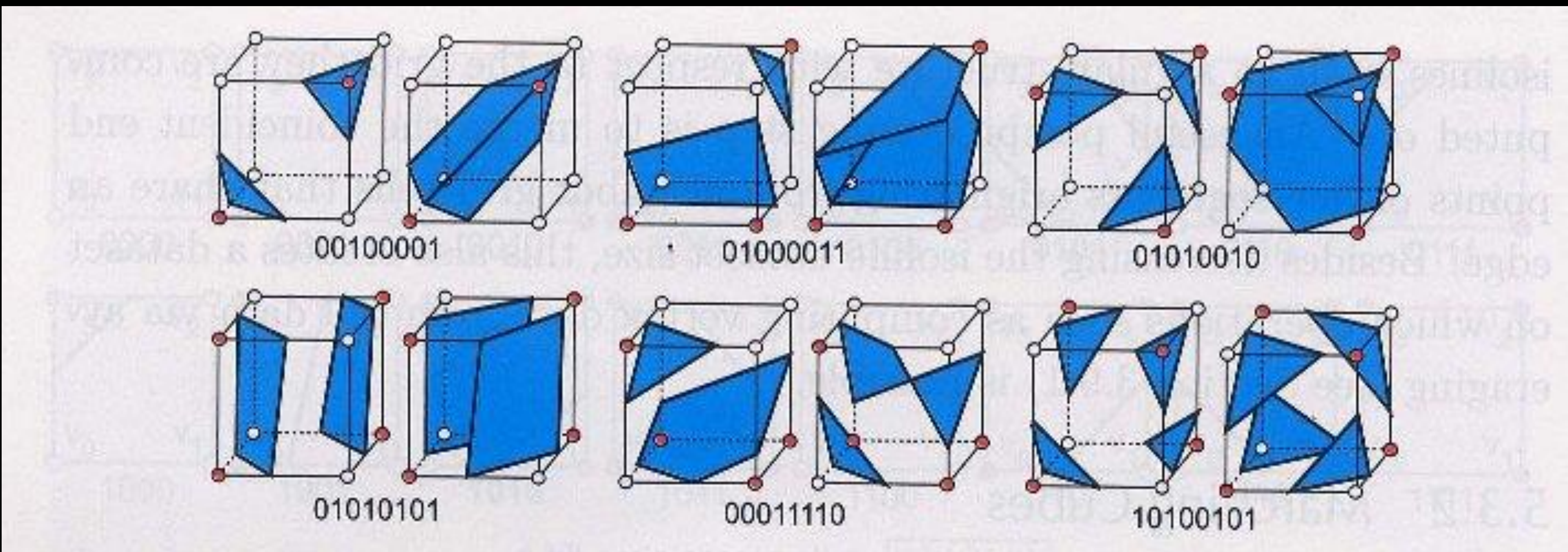
Implementation: Marching Cube



Topological State of a hex Cell

Marching cube generates a set of polygons for each contoured cell: triangle, quad, pentagon, and hexagon

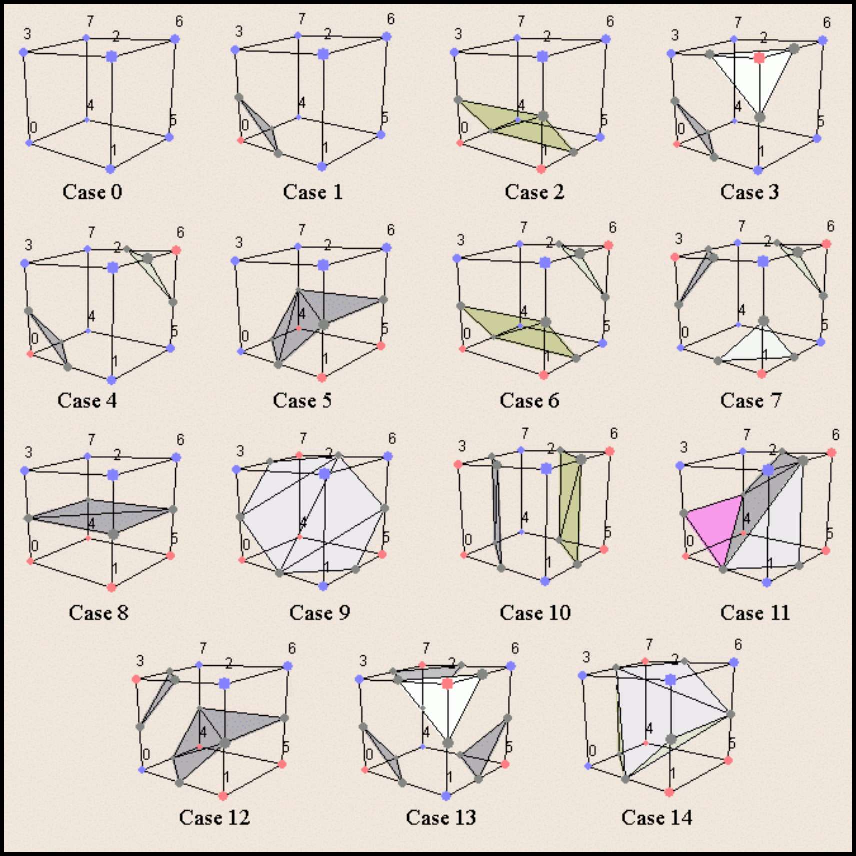
Marching Cubes -- Ambiguity



Ambiguous cases for marching cubes. Each case has two contouring variants.

How Are The Cubes Used

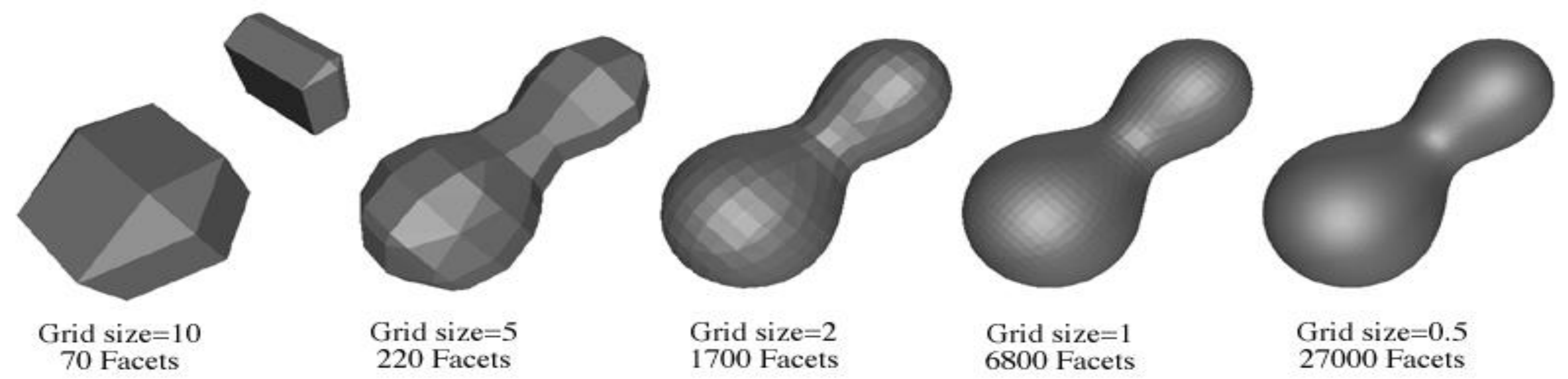
- Pixels on the slice surfaces determine 3D surfaces.
- 256 surface permutations, but only 14 unique patterns.
- A normal is calculated for each triangle vertex for rendering.



Triangle Creation

- Determine triangles contained by a cube.
- Determine which cube edges are intersected.
- Interpolate intersection point using pixel density.
- Calculate unit normals for each triangle vertex using the gradient vector.

Grid Resolution

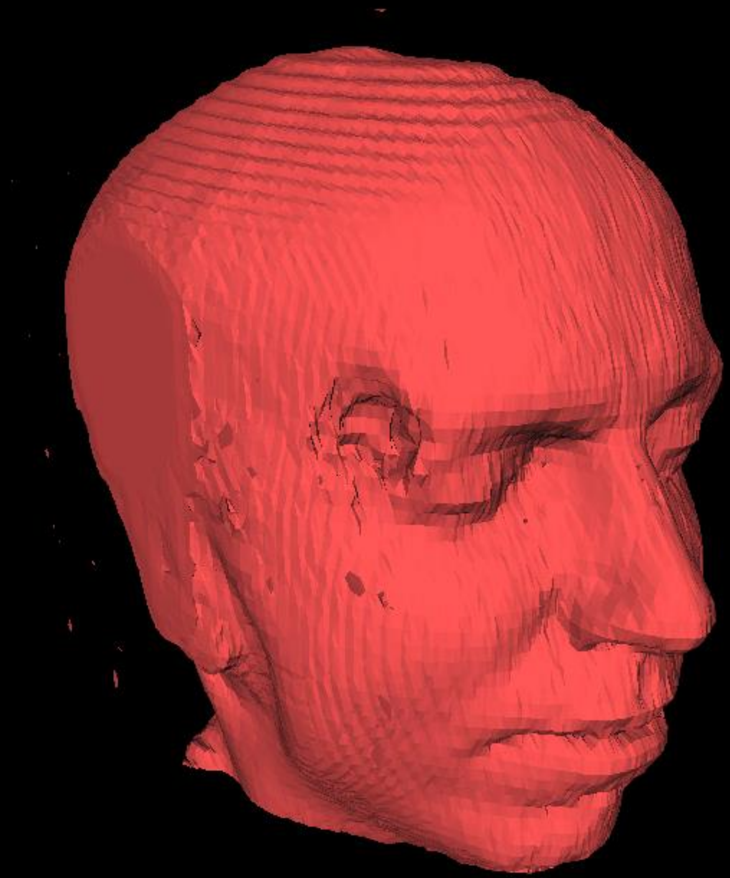


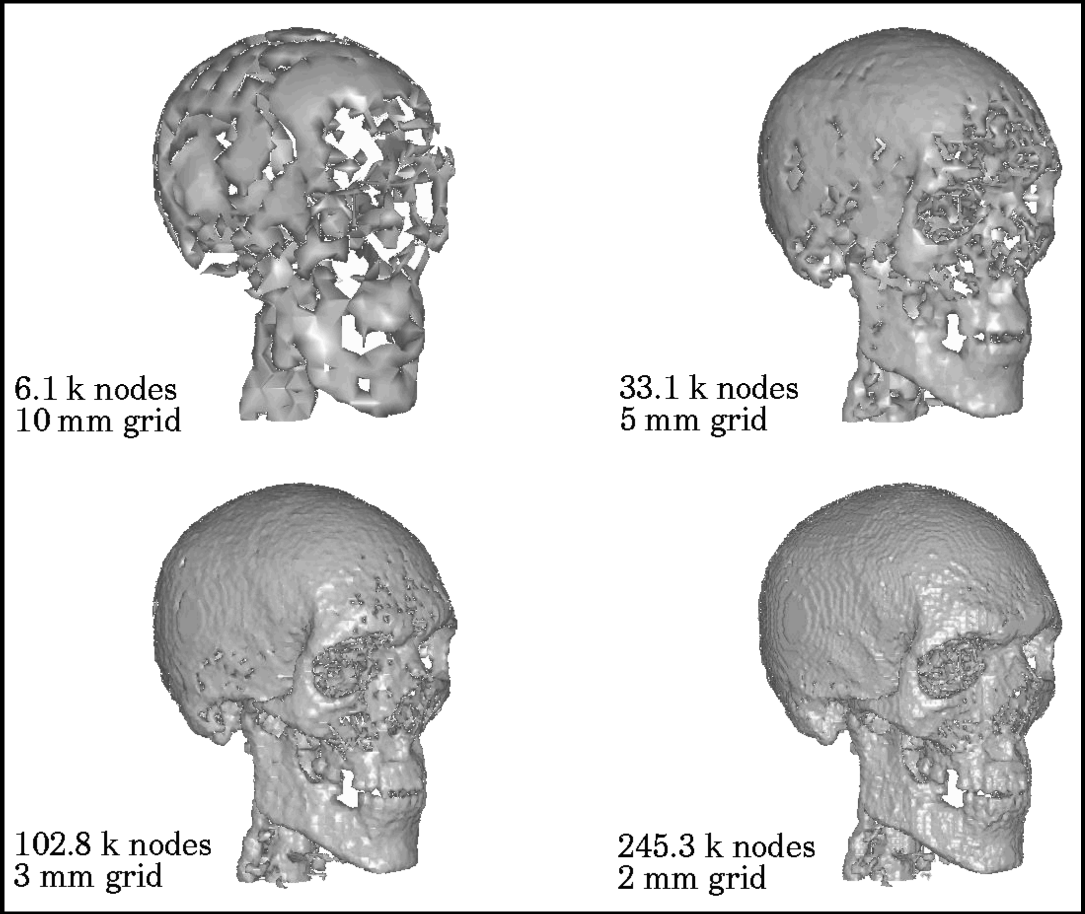
- Variations can increase/decrease surface density.

Improvements over Other Methods

- Utilizes pixel, line and slice coherency to minimize the number of calculations.
- Can provide solid modeling.
- Can use conventional rendering techniques and hardware.
- No user interaction is necessary.
- Enables selective displays.
- Can be used with other density values.

Examples

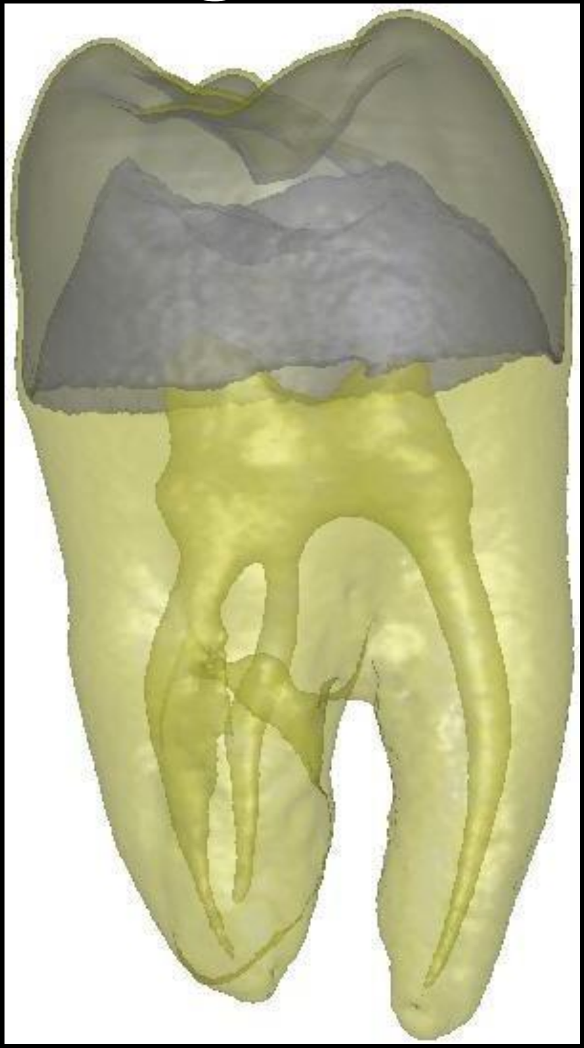



6.1 k nodes
10 mm grid
33.1 k nodes
5 mm grid
102.8 k nodes
3 mm grid
245.3 k nodes
2 mm grid

Marching Cubes

- General rule: most isosurface details that are under or around the size of the resolution of the iso-surfaced dataset can be
  - either actual data or artifact
  - should be interpreted with great care
- Can draw more than a single iso-surface of the same dataset in one visualization

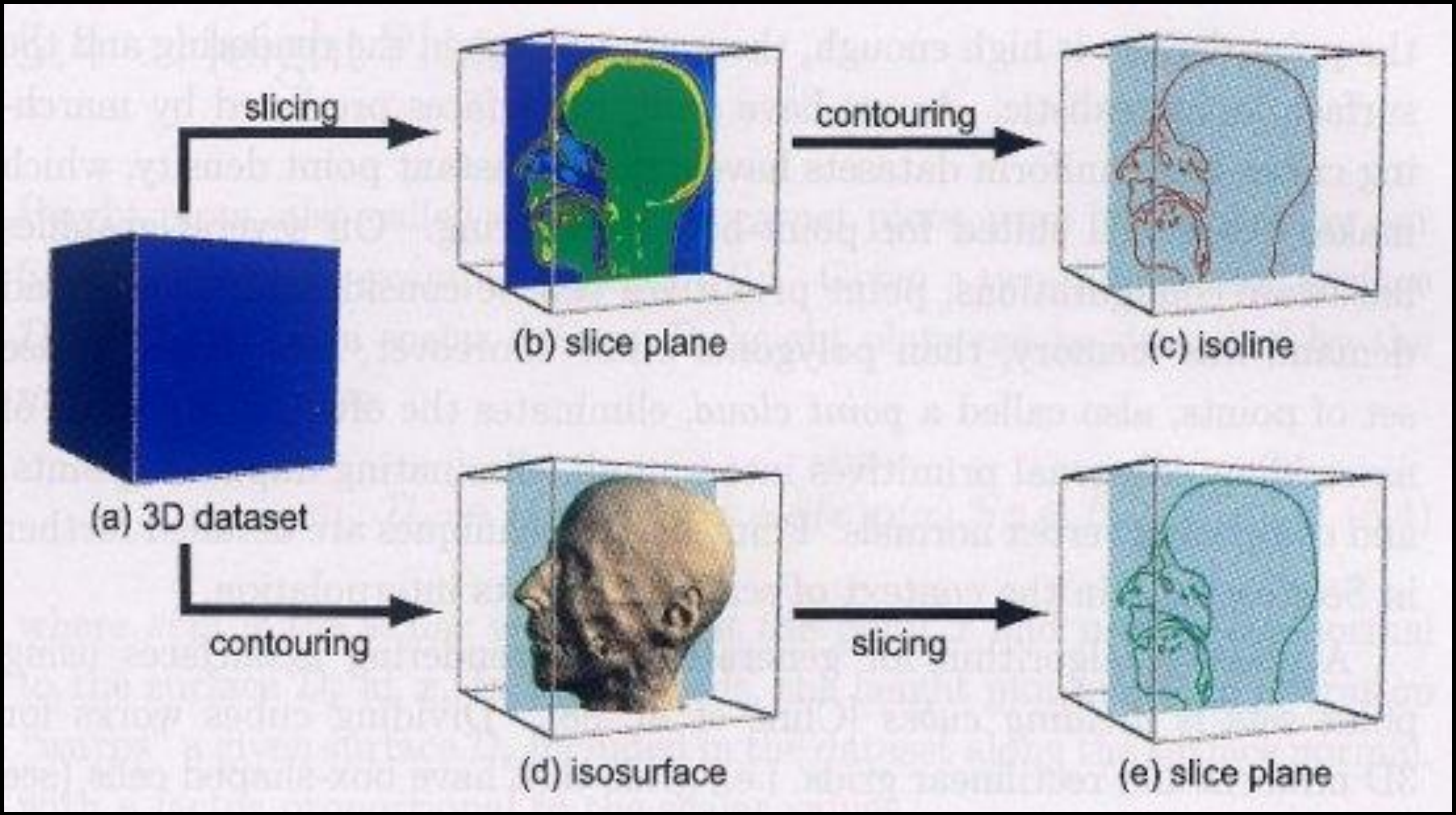
Marching Cubes



Nested isosurfaces of a tooth scan dataset

Marching Cubes

Isosurfaces and isolines are strongly related



Isosurfaces, isolines, and slicing

Summary

- Marching Cubes provides a simple algorithm to translate a series of 2D scans into 3D objects
- Marching Squares and Marching Cubes have many variations to address:
  - Generalcity in terms of input dataset type
  - Speed of execution
  - Quality of obtained contours
- Isosurface can also be generated and rendered using point-based techniques
  - 3D surface can be rendered using large numbers of (shaded) point primitives
  - Point primitive can be considerably faster and demand less memory than polygonal ones on some graphics hardware
  - Point cloud

outline

Visualizing scalar data

Popular scalar visualization techniques

- Color mapping
- Contouring
- Height plots

Height Plots

- The height plot operation is to "warp" the data domain surface along the surface normal, with a factor proportional to the scalar value

$$
\begin{aligned}
& m : D_s \rightarrow D_h, \\
& m(x) = x + s(x)\vec{n}(x), \\
& \forall x \in D_s
\end{aligned}
$$

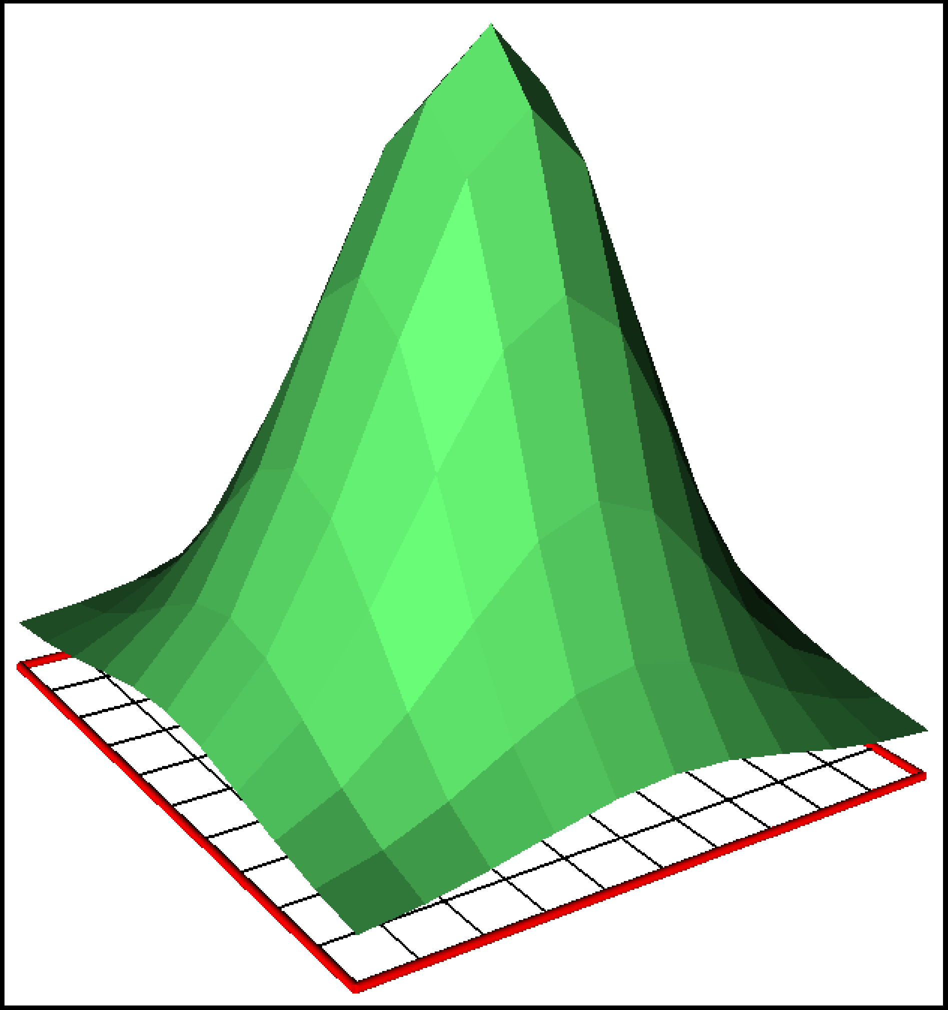
Height plots

- Height plots (elevation or carpet plots)

$$m : D_s \rightarrow D, m(x) = x + s(x)n(x), \forall x \in D_s$$

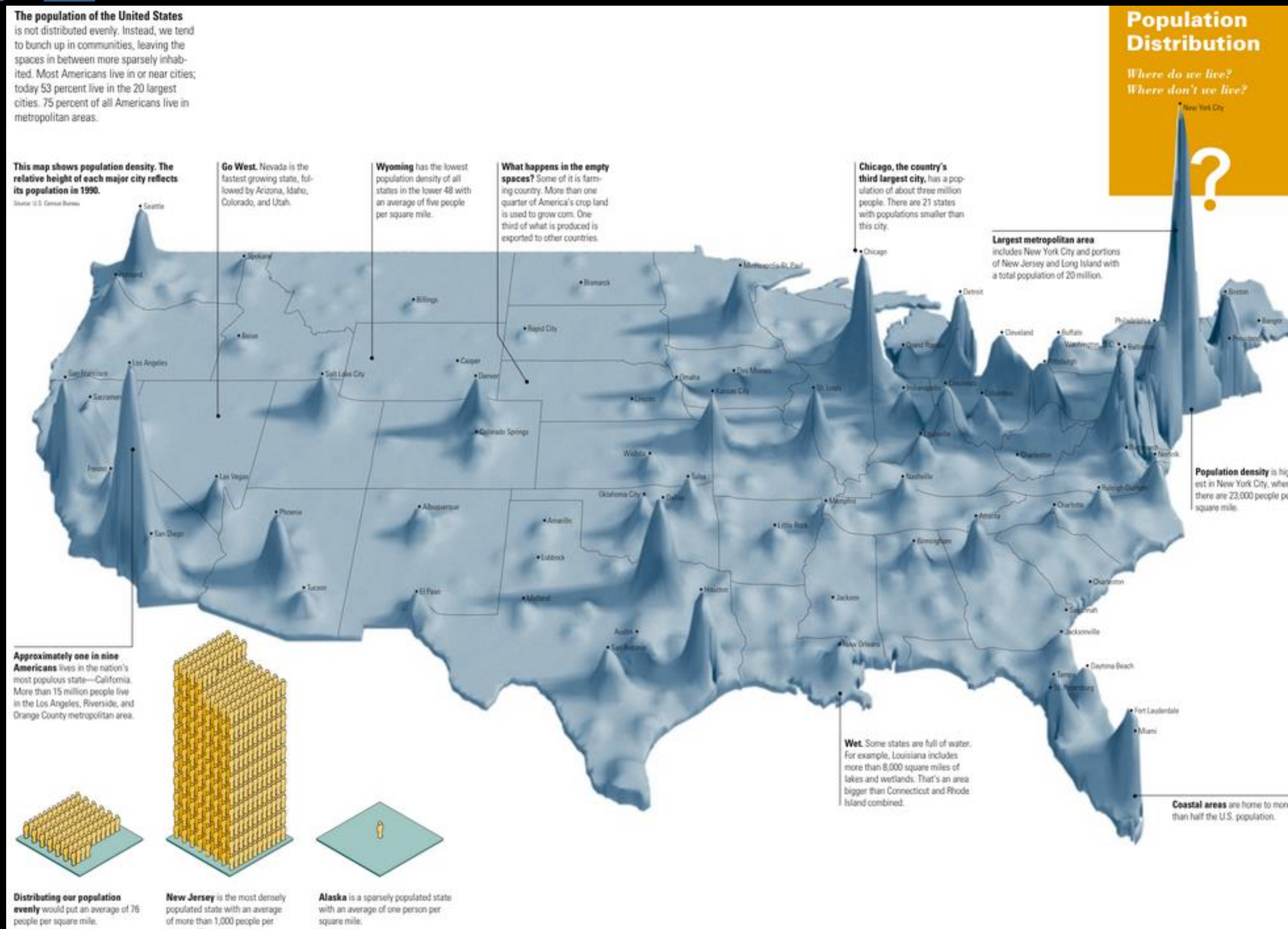
  - S(x) is the scalar value of D at the point x
  - n(x) is the normal to the surface Ds at x
- The height plot mapping operation “warp” a given surface Ds included in the dataset along the surface normal, with a factor proportional to the scalar values.
- Height plots are a particular case of displacement, or warped plots

Height Plots



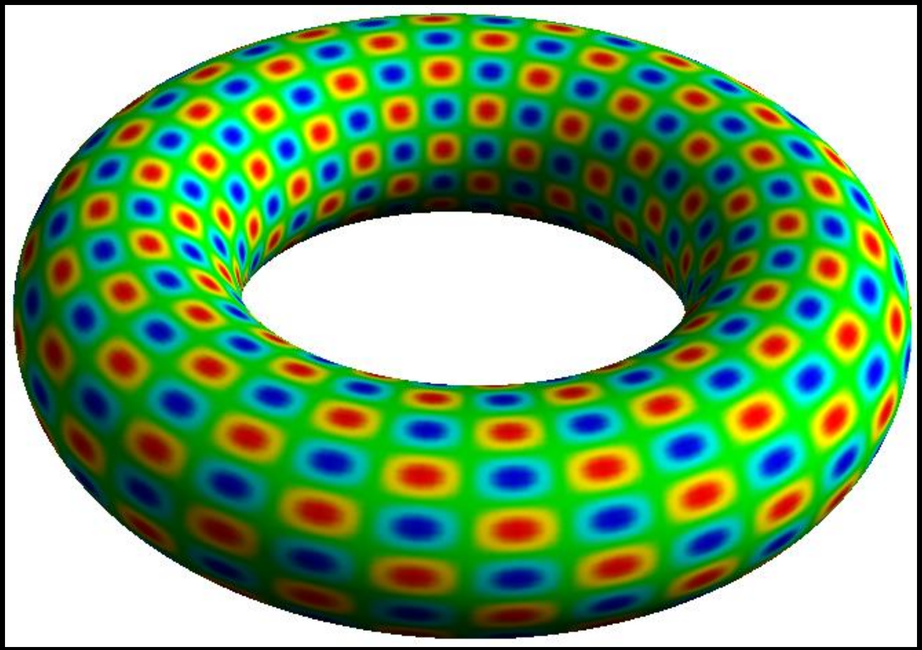
Height plot over a planar 2-D surface

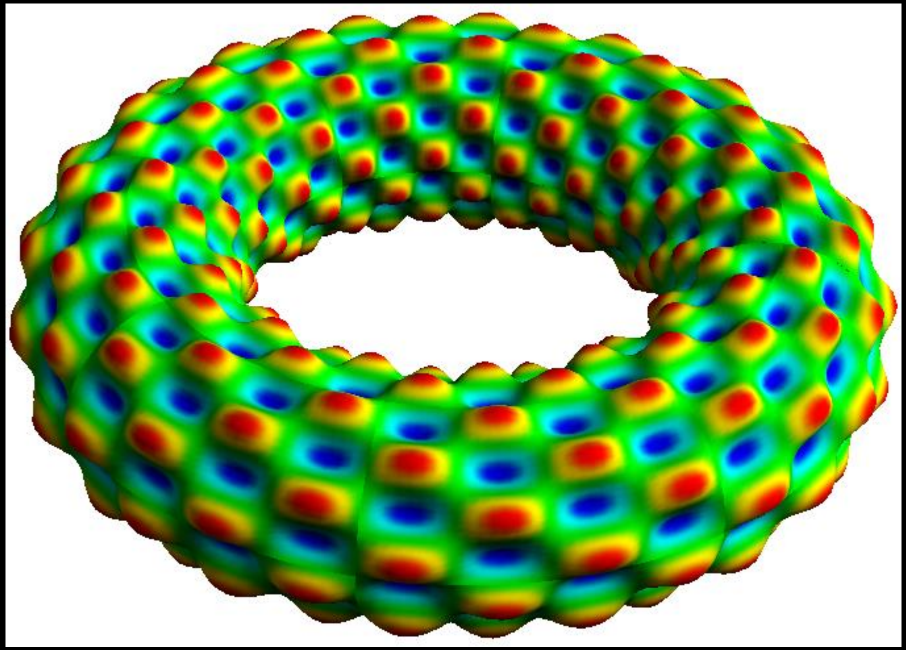
Height Plots



Population density of America

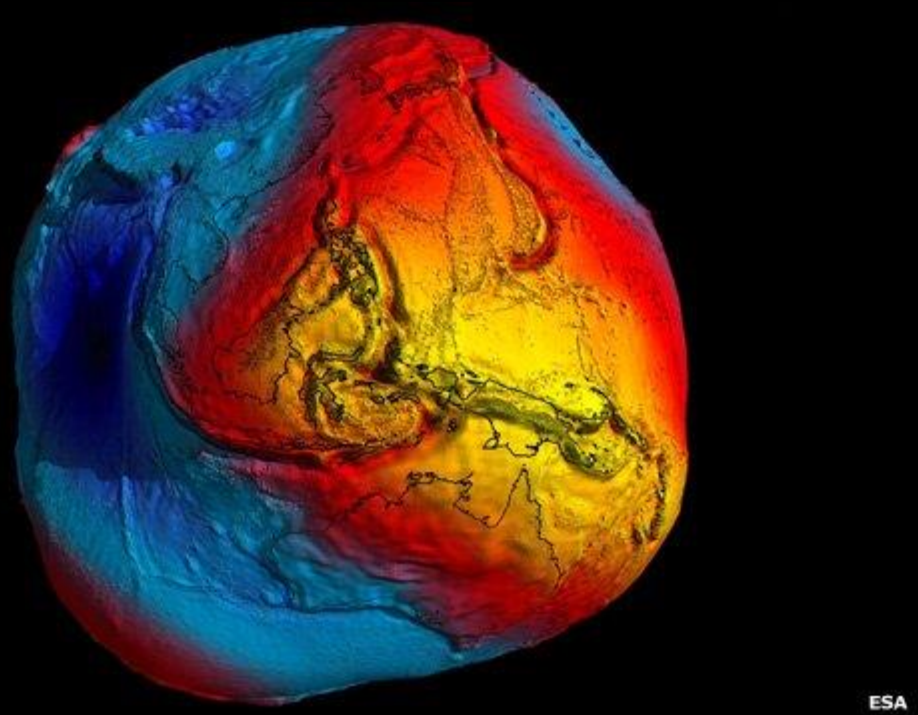
Height Plots





Height plot over a nonplanar 2-D surface

Height Plots



“Potato Earth” shows variations in planet's gravity

Summary

- Visualizing scalar data
  - Color mapping
    Assign a color as a function of the scalar value at each point of a given domain
  - Contouring
    Displaying all points with a given 2D or 3D domain that have a given scalar value
  - Height plots
    Deform the scalar dataset domain in a given direction as a function of the scalar data
- Advantage
  - Produce intuitive results
  - Easily understood by the vast majority of users
  - Simple to implement
- Disadvantage
  - A number of restrictions
  - One or two dimensional scalar dataset
  - We want to visualize the scalar values of ALL, not just a few of the data points of a 3D dataset