Scientific Visualization
Part 1: Geometric Data

John Keyser

Scientific Visualization

- Key distinguishing feature is that the source data is scientific data
  - Can be measured or from simulation
  - But, this is still pretty general
  - Typically distinguished from information visualization
- Most commonly: the data has a strong spatial component
  - Possibly on a 2D surface
  - Often in 3D

Typical Scientific Data formats

- Data located at 3D spatial points
  - Data can be multidimensional
- Examples:
  - Weather data (various) sampled at various locations
    - On ground
    - Through the atmosphere
  - Fluid motion measured in a wind tunnel or a wind tunnel simulation
  - MRI/CT scan: value recorded across 3D domain

Dividing Scientific Visualization

- We will discuss SciVis in two “groups”
- Volumes and Geometry
  - Challenges revolve around the locations at which the samples are taken
- Vector and Multivariate Data
  - Challenges revolve around how to view and interpret multiple values at each point
- Note: many tasks will face both of these
- We’ll first discuss Volumes and Geometry

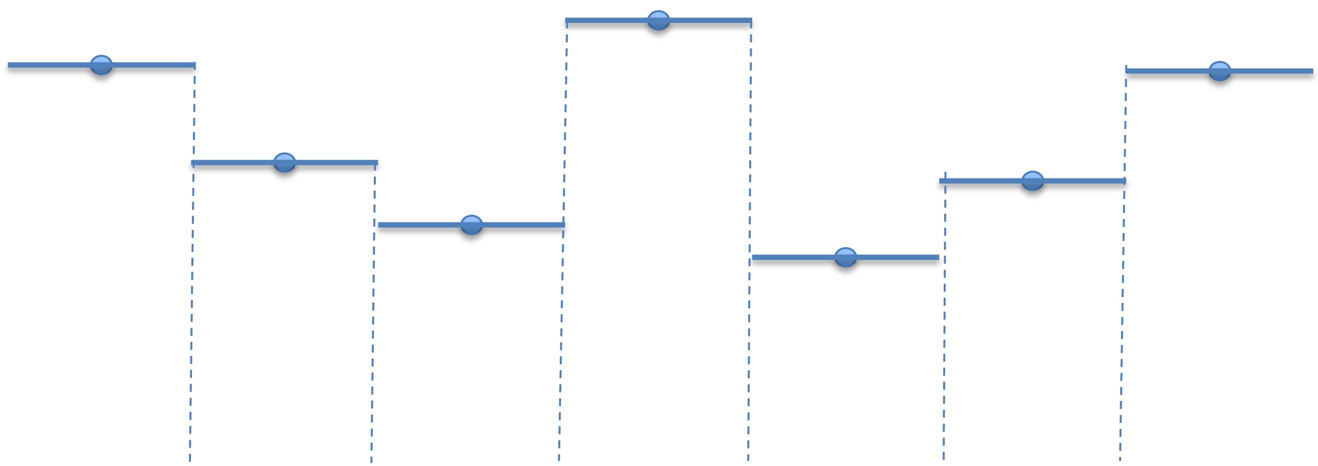
Geometric Data

- Assume a single value is sampled across a plane (or 2D surface)
- Data is known at discrete points, but needs to be known at other points
  - How can we do this?

- We'll start with 2D data, later discuss 3D

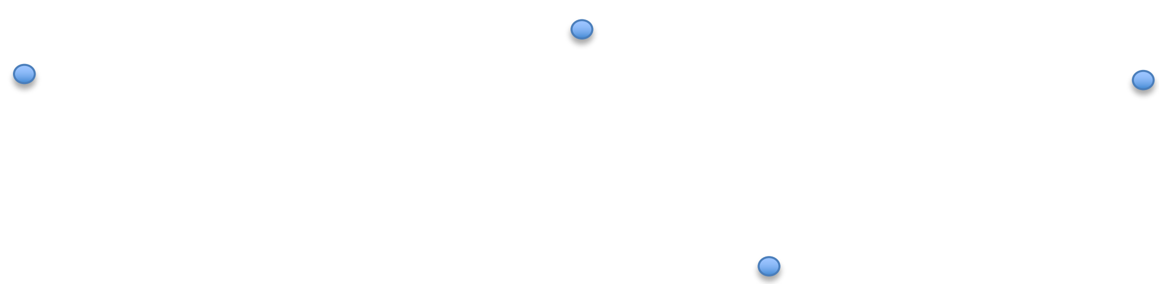
Nearest Neighbor Values

- Idea: the value at a point is the value of the nearest sample
- 1D:



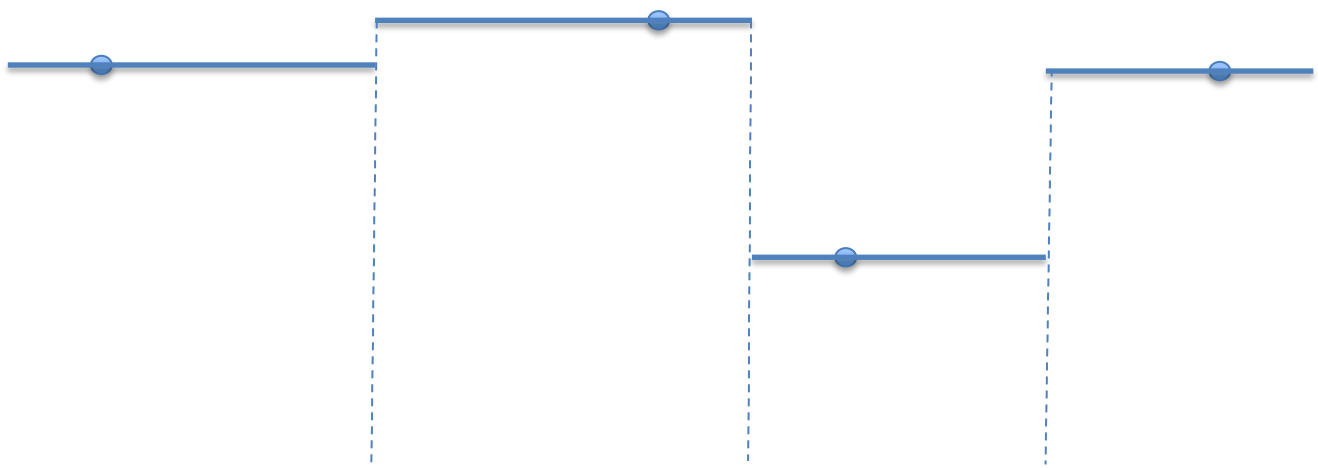
Nearest Neighbor Values

- Idea: the value at a point is the value of the nearest sample
- 1D with uneven samples



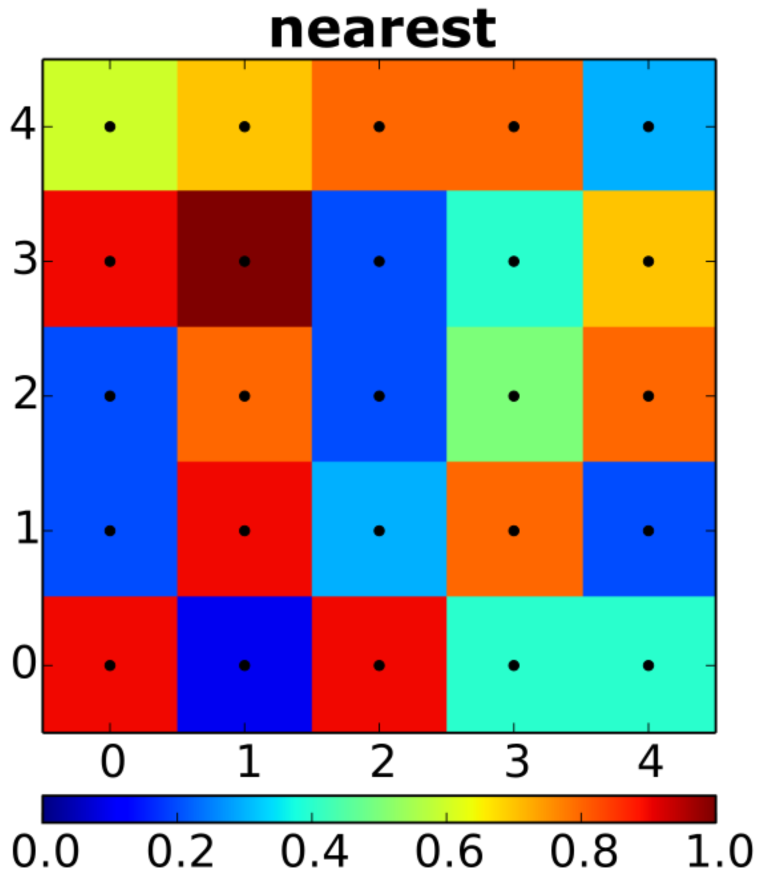
Nearest Neighbor Values

- Idea: the value at a point is the value of the nearest sample
- 1D with uneven samples



2D/3D Nearest Neighbors

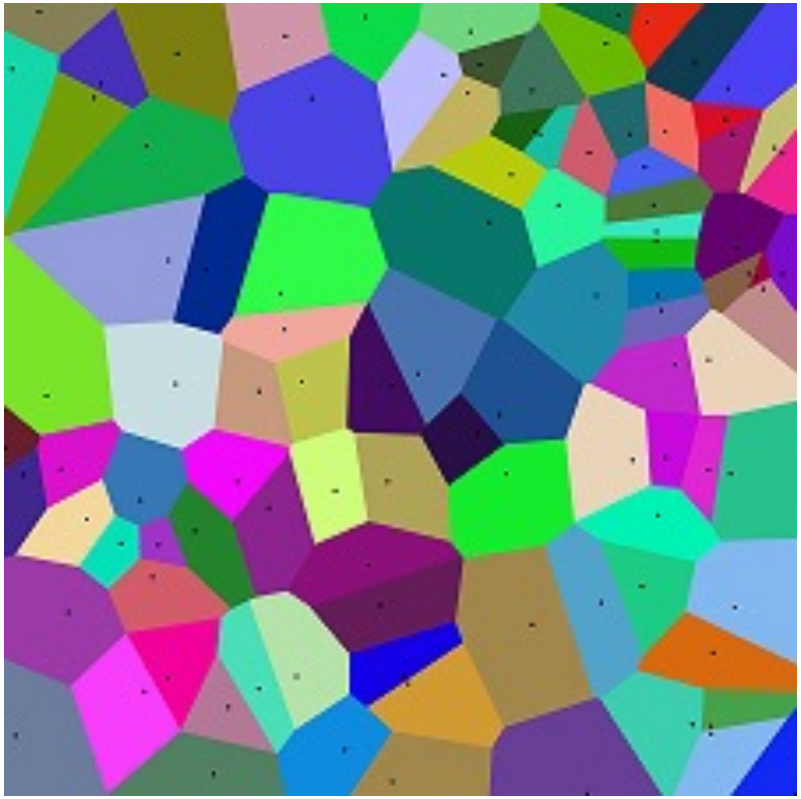
- If on a grid, this is similar to pixels in an image



From: https://jordhy97.github.io/Interpolation-Methods/

2D/3D Nearest Neighbors

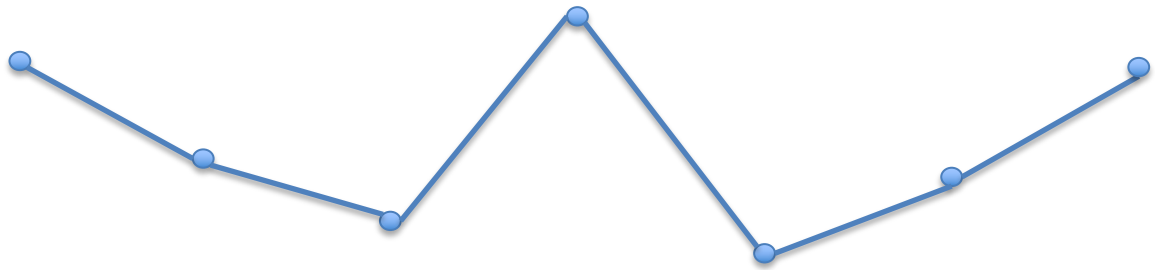
- If irregularly spaced, nearest neighbors are given by the Voronoi Diagram



From: https://jordhy97.github.io/Interpolation-Methods/

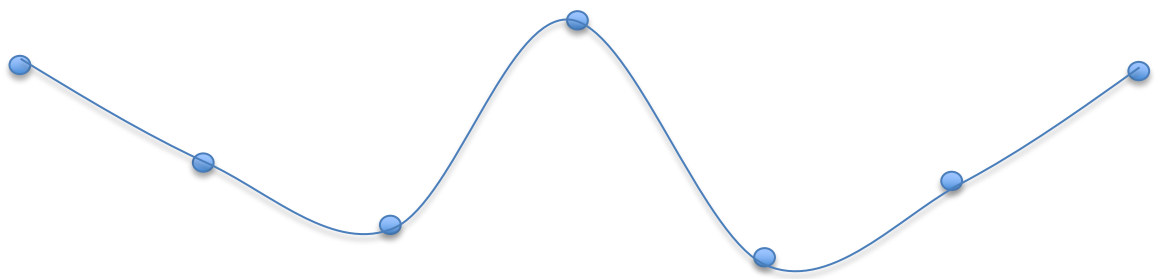
Interpolation

- Idea: Use a function to interpolate intermediate values
- 1D linear:



Interpolation

- Idea: Use a function to interpolate intermediate values
- 1D curve:

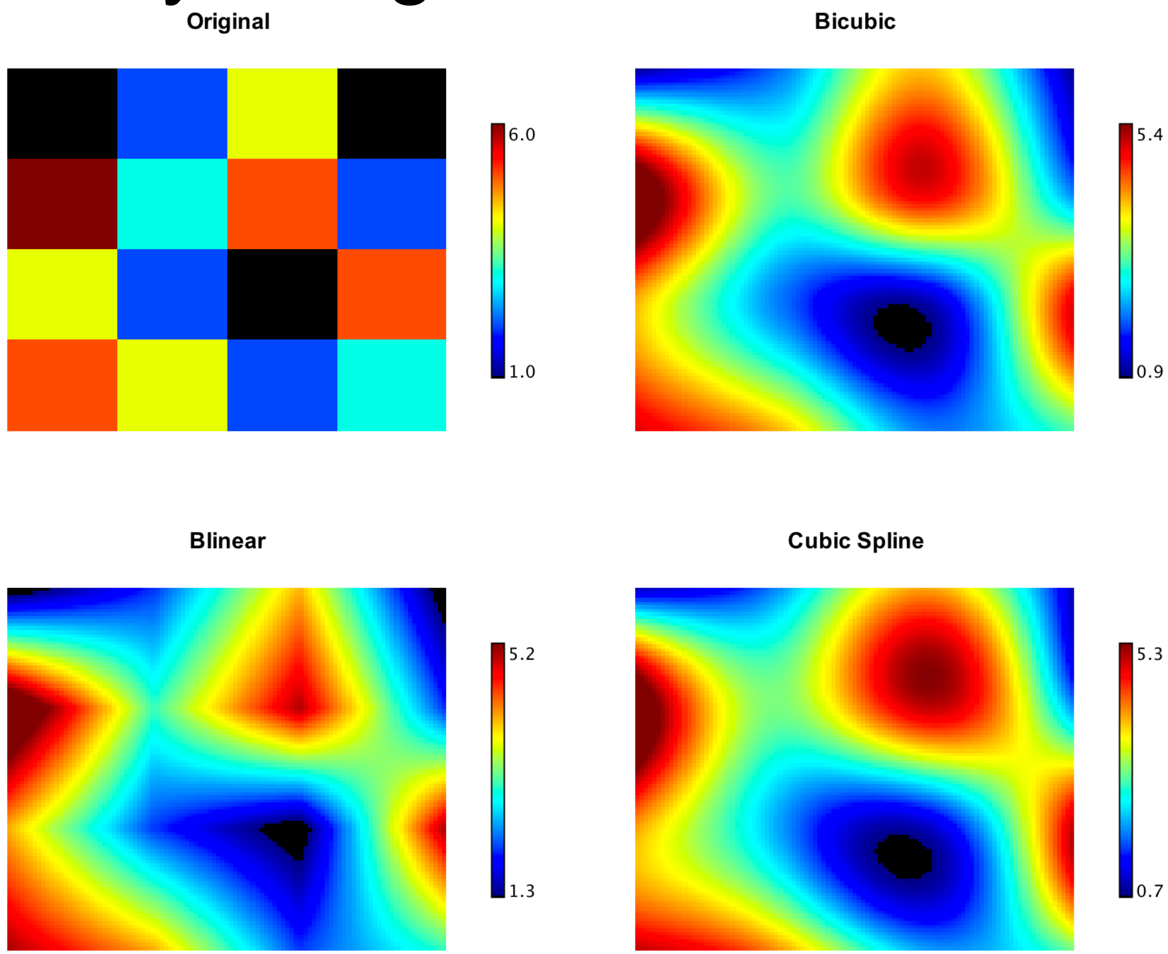


Interpolation

- There are many ways to interpolate values
  - Linear
  - Cubic
  - Higher order polynomials
  - Splines
- This is especially so for 2D/3D
- Irregular Data makes things tougher
- This is a topic all of its own, with many applications beyond just visualization

2D/3D Interpolation

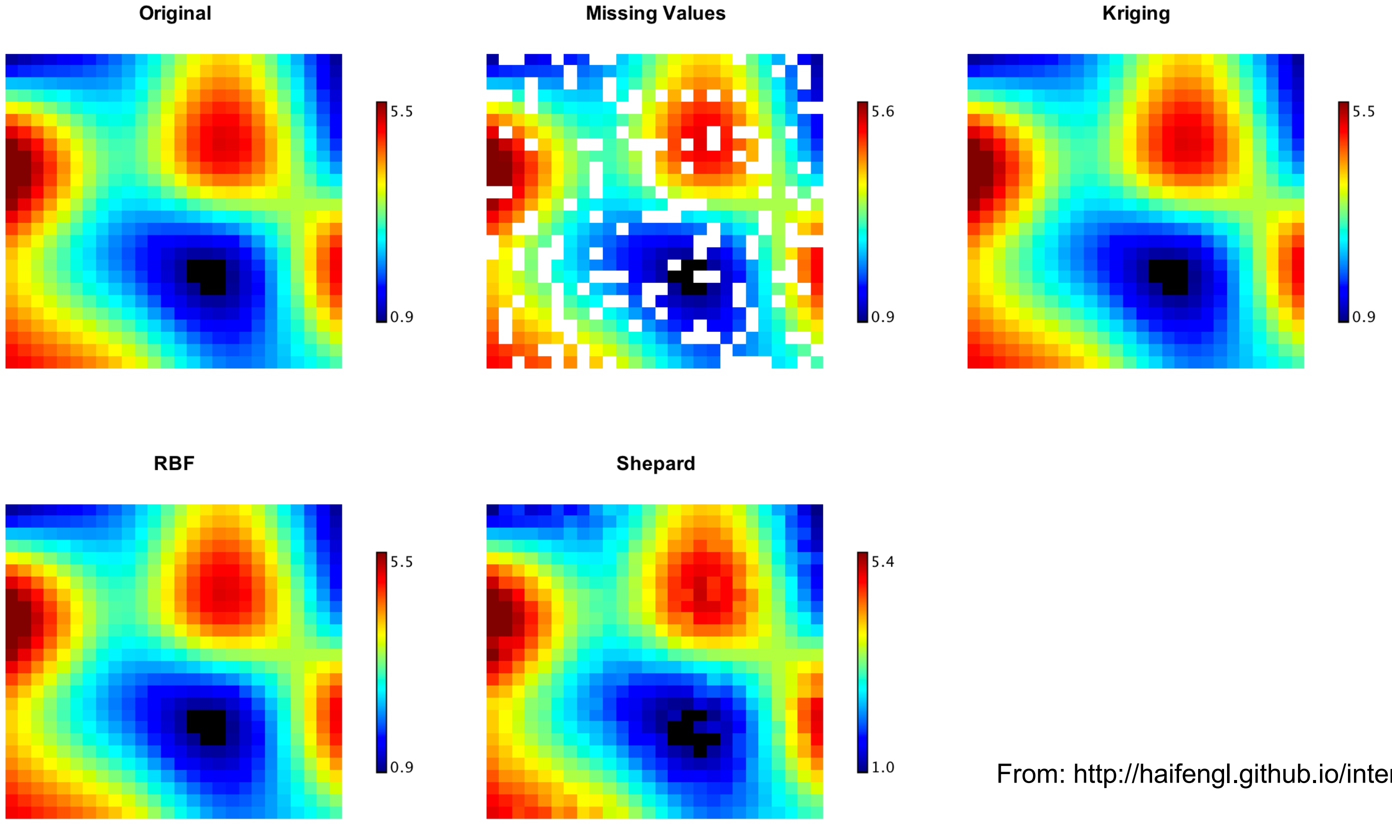
- For grid, relatively straightforward



From: http://haifengl.github.io/interpolation.html

Interpolating Irregular Spaced 2D/3D Data

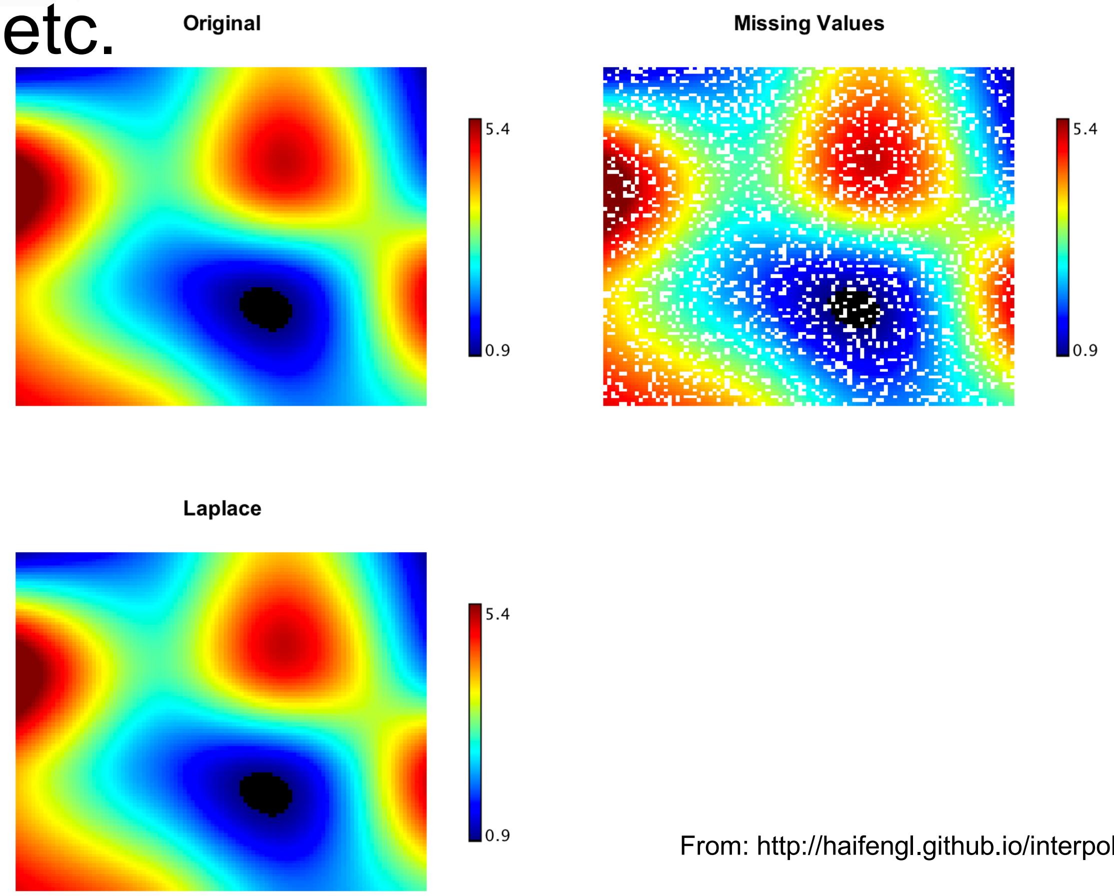
- Multiple methods: Kriging (like fitting a polynomial surface), Radial Basis Functions, Laplace Interpolation, etc.



From: http://haifengl.github.io/interpolation.html

Interpolating Irregular Spaced 2D/3D Data

- Multiple methods: Kriging (like fitting a polynomial surface), Radial Basis Functions, Laplace Interpolation, etc.

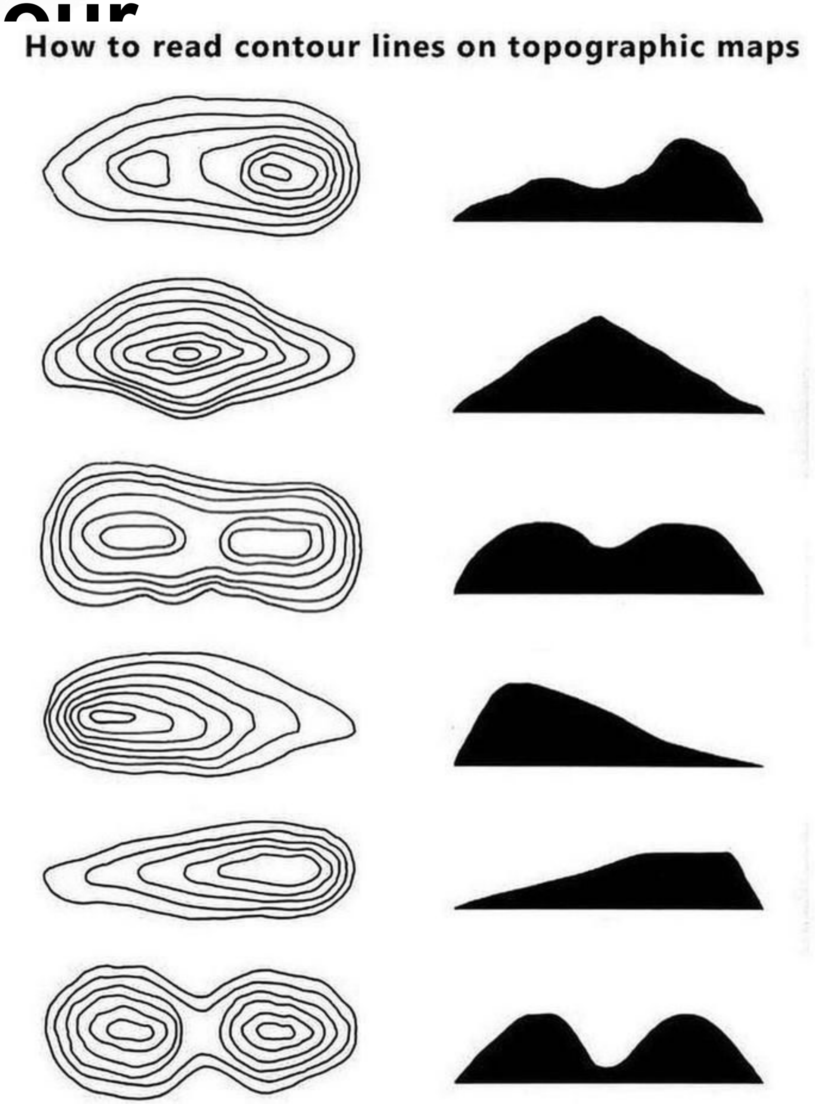


From: http://haifengl.github.io/interpolation.html

Contours

- A way of identifying **isovalues** (curves of constant value)
- For a given variable, contours will show the points assumed to be at a given constant value
- Can usually infer the “shape” from the contour
- Perpendicular to contours is the gradient (direction of maximum change)

Contour



- Commonly used for height in topographical maps, but can be used for any value on a 2D surface

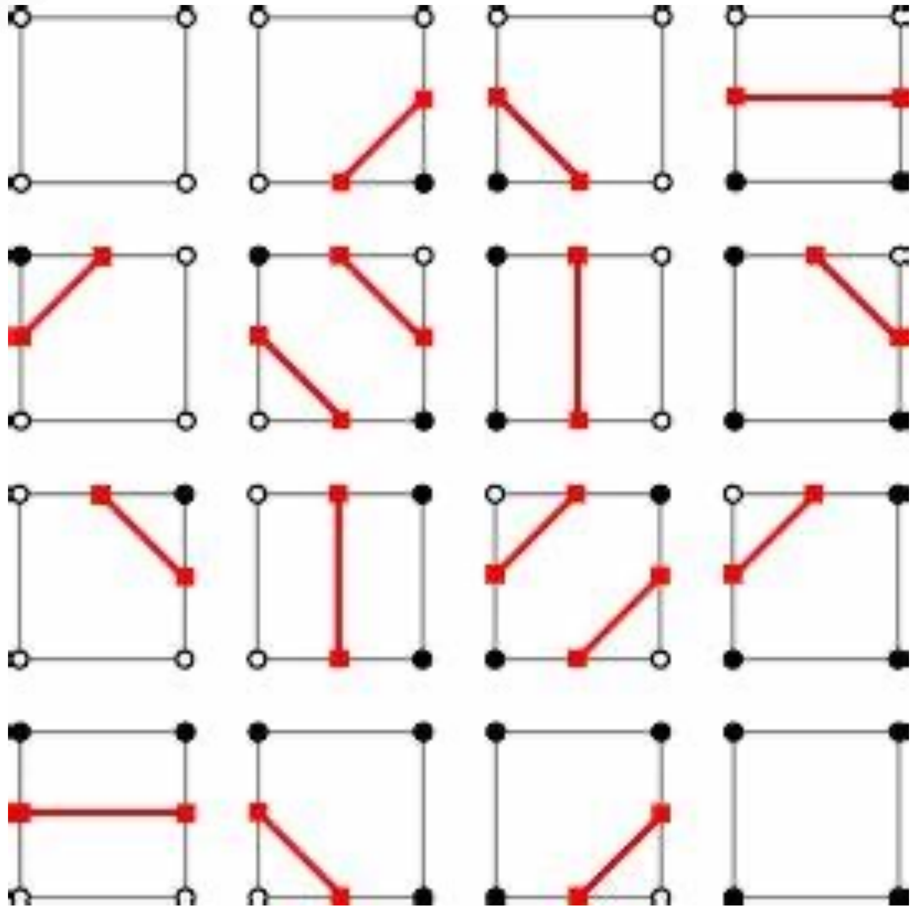
From: https://www.pinterest.com/pin/671669731912124660/

Contour Extraction

- Given data sample points, the contour itself can be interpolated between sample points
- For a grid, the contour locally can be drawn between sets of four points if some are on one side of contour and some on other
  - When connected together, this will form closed contours
- Smoothing is sometimes a post-process

Contour Extraction on 2D grid

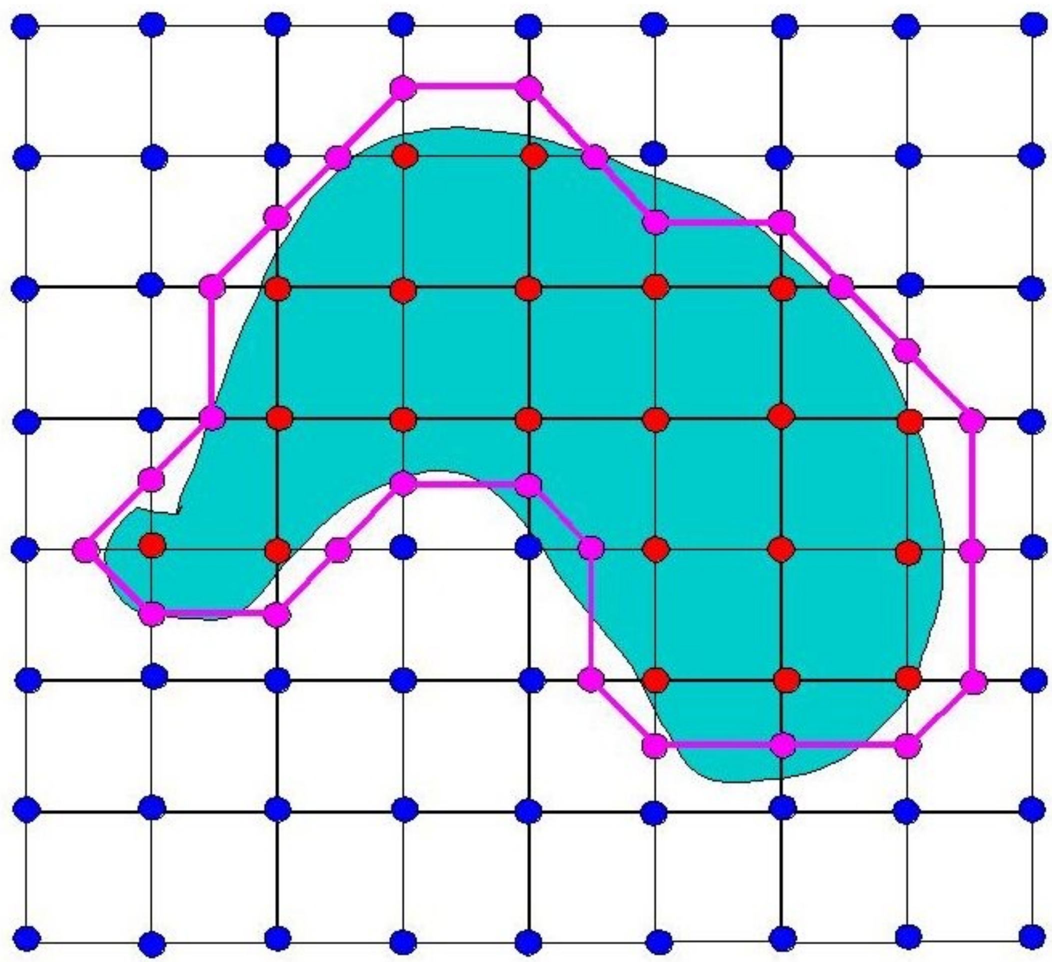
- Draw a contour based on pattern of which corners of a box are above vs. below the isovalue
- 2D version of Marching Cubes



From: https://www.boristhebrave.com/2018/04/15/marching-cubes-tutorial/

Contour Extraction on 2D grid

- Connecting the individual elements forms a closed contour



https://www.cs.carleton.edu/cs_comps/0405/shape/marching_cubes.html

Isosurfaces

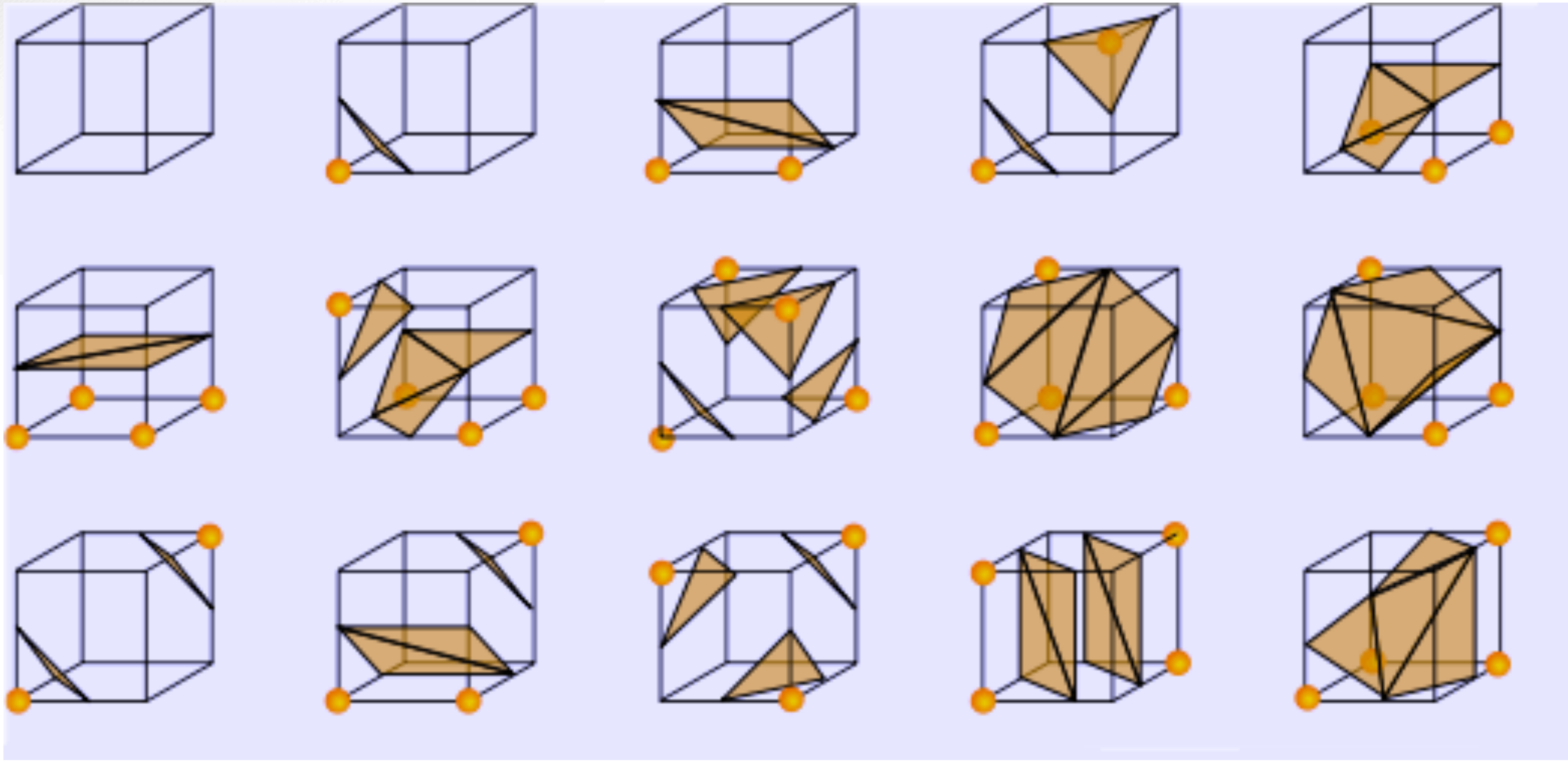
- The 3D extension to contours
- Describes the surface of constant value
- For many applications, this is also key to segmentation
  - Identifies one region that is of a different value than the surroundings

Extracting Isosurfaces: Marching Cubes

- The most well-cited graphics paper
- Works on grid data
  - For other data, often first interpolate to a grid
- For 8 vertices of a cube in grid, classify vertices as above or below isovalue.
  - There are 2^8 = 256 possible combinations
  - But, due to symmetry, there are really only 15 possible cases
- Surface is not as smooth as often desired, though, and sharp features are limited

Marching Cubes

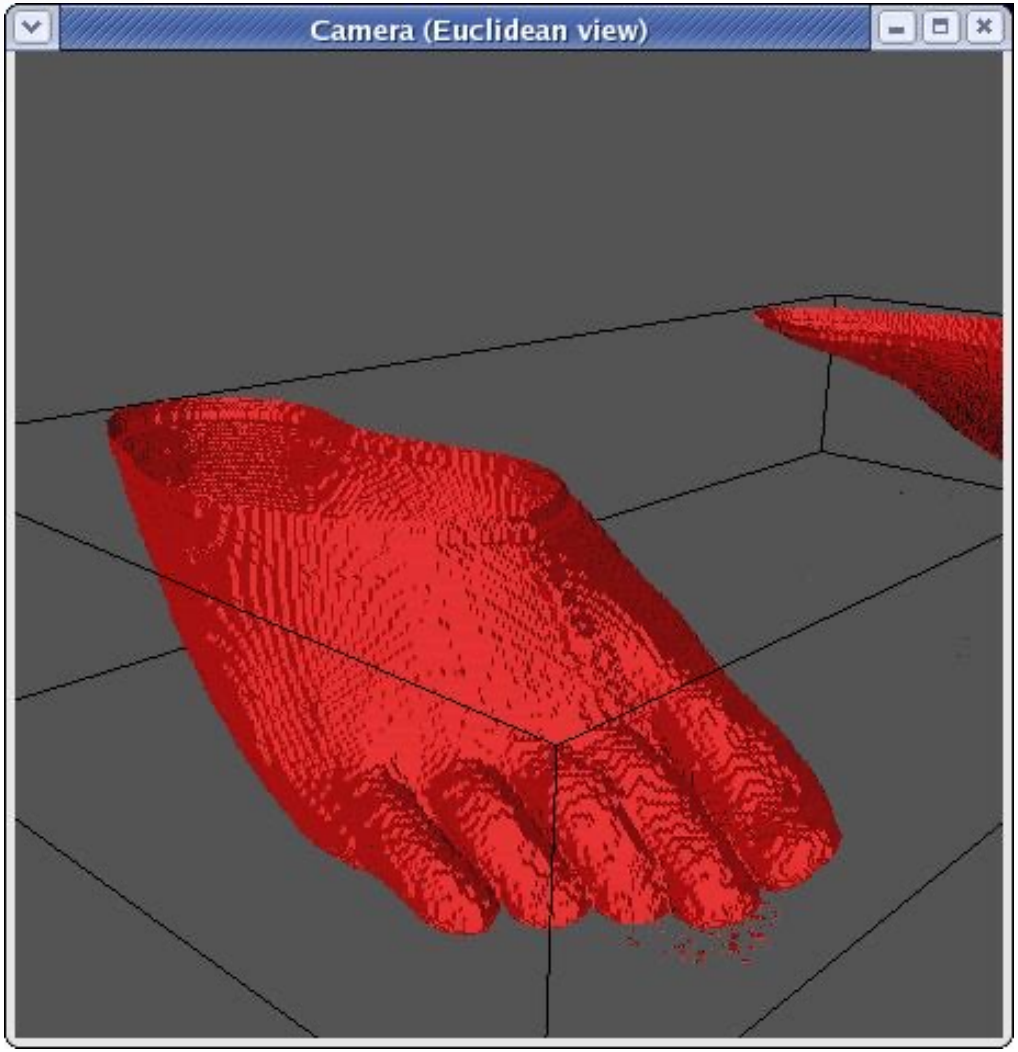
- The 15 Cases



From: https://www.scientificlib.com/en/ComputerGraphics/MarchingCubes.html/

Marching Cubes

- Connected elements of marching cubes will create the isosurface
- Some smoothing might be desirable



https://www.cs.carleton.edu/cs_comps/0405/shape/marching_cubes.html

Isosurface Extraction and Segmentation

- There are other methods of doing “better” isosurface extraction
  - But Marching Cubes is still the most common
- For segmentation, need to find a single isovalue that separates region from its surroundings
  - When this is not possible, other approaches besides isosurfaces are sometimes used
  - Note that noise can lead to noticeable artifacts on the isosurface
  - Segmentation is a challenge all of its own