Data Visualization (CSCI 627/490)

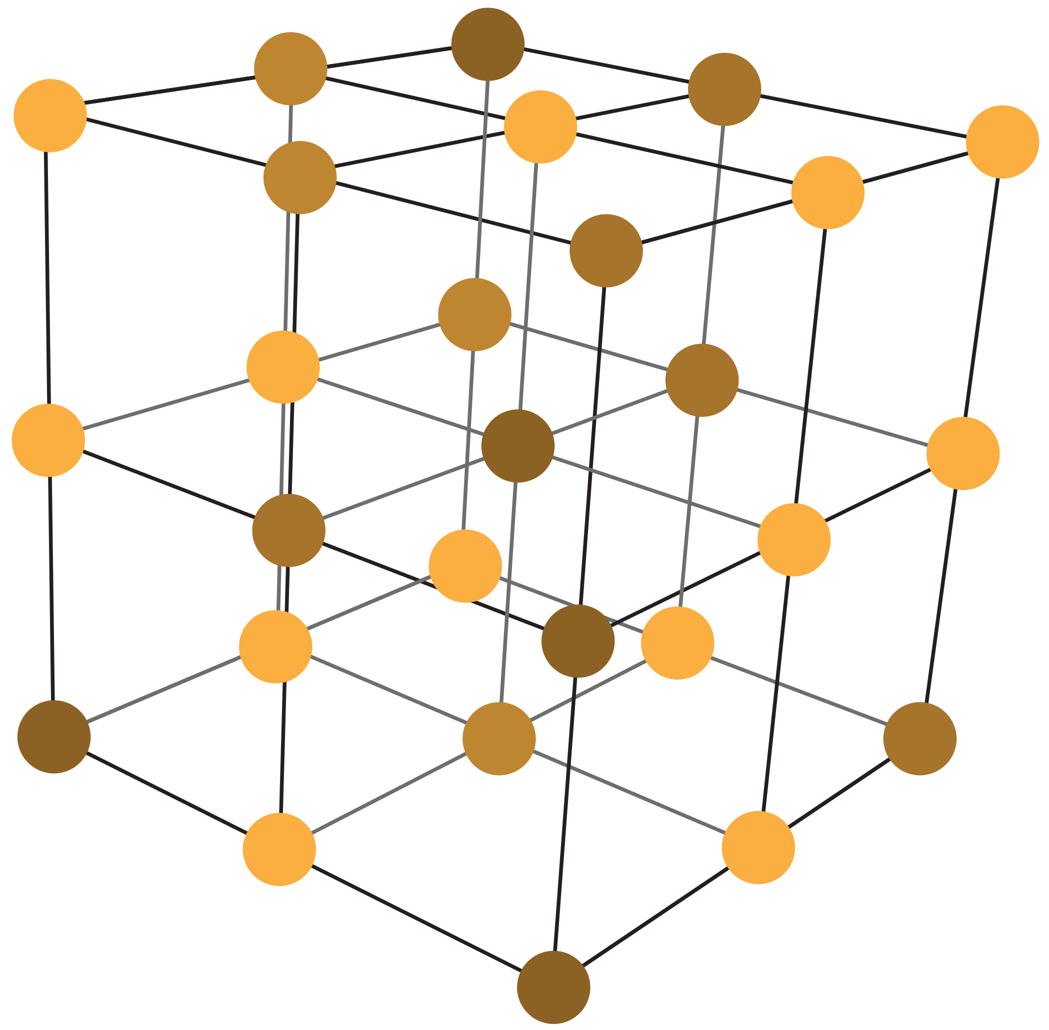
Isosurfacing & Volume Rendering

Dr. David Koop

Scivis and Infovis

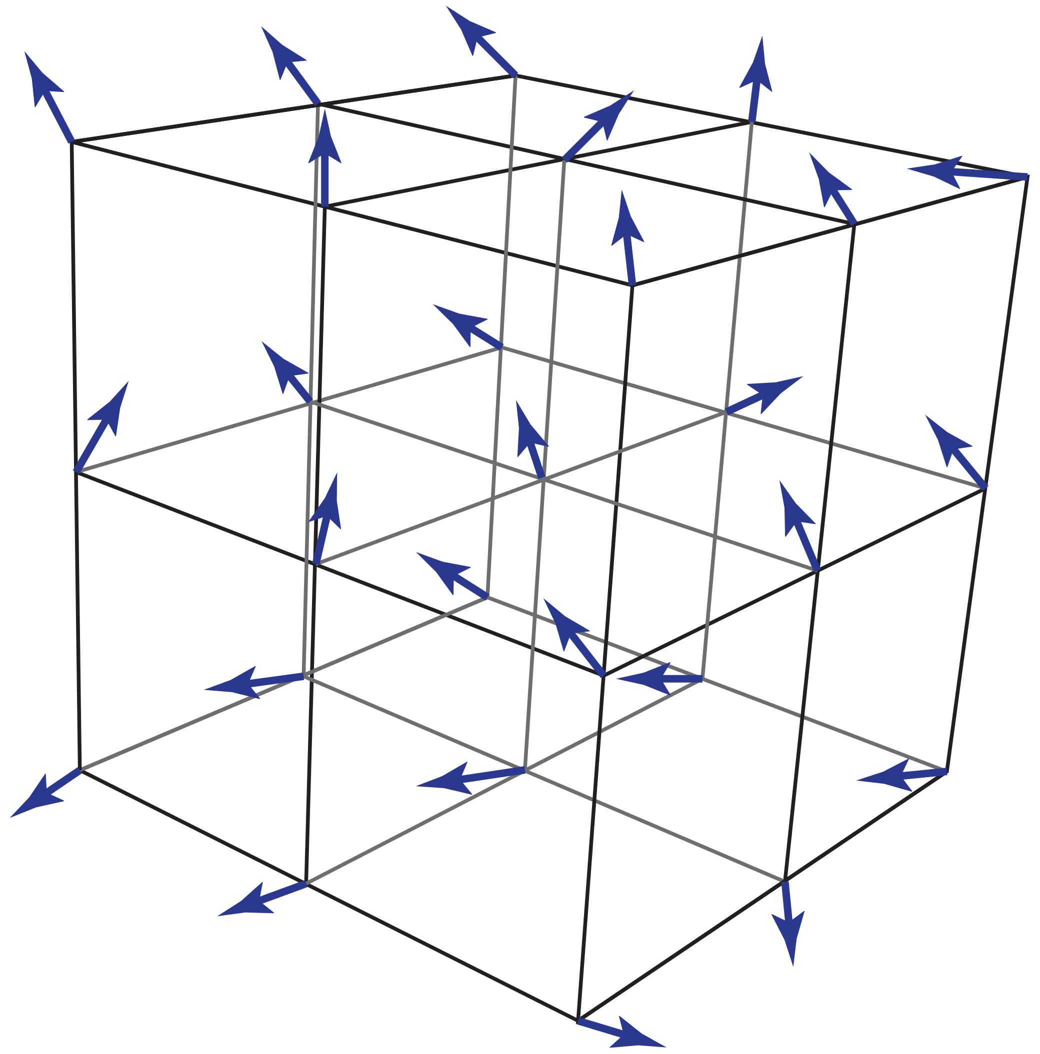
- Two subfields of visualization
- **Scivis** deals with data where the spatial position is given with data
  - Usually continuous data
  - Often displaying physical phenomena
  - Techniques like isosurfacing, volume rendering, vector field vis
- In **Infovis**, the data has no set spatial representation, designer chooses how to visually represent data

Fields in Visualization



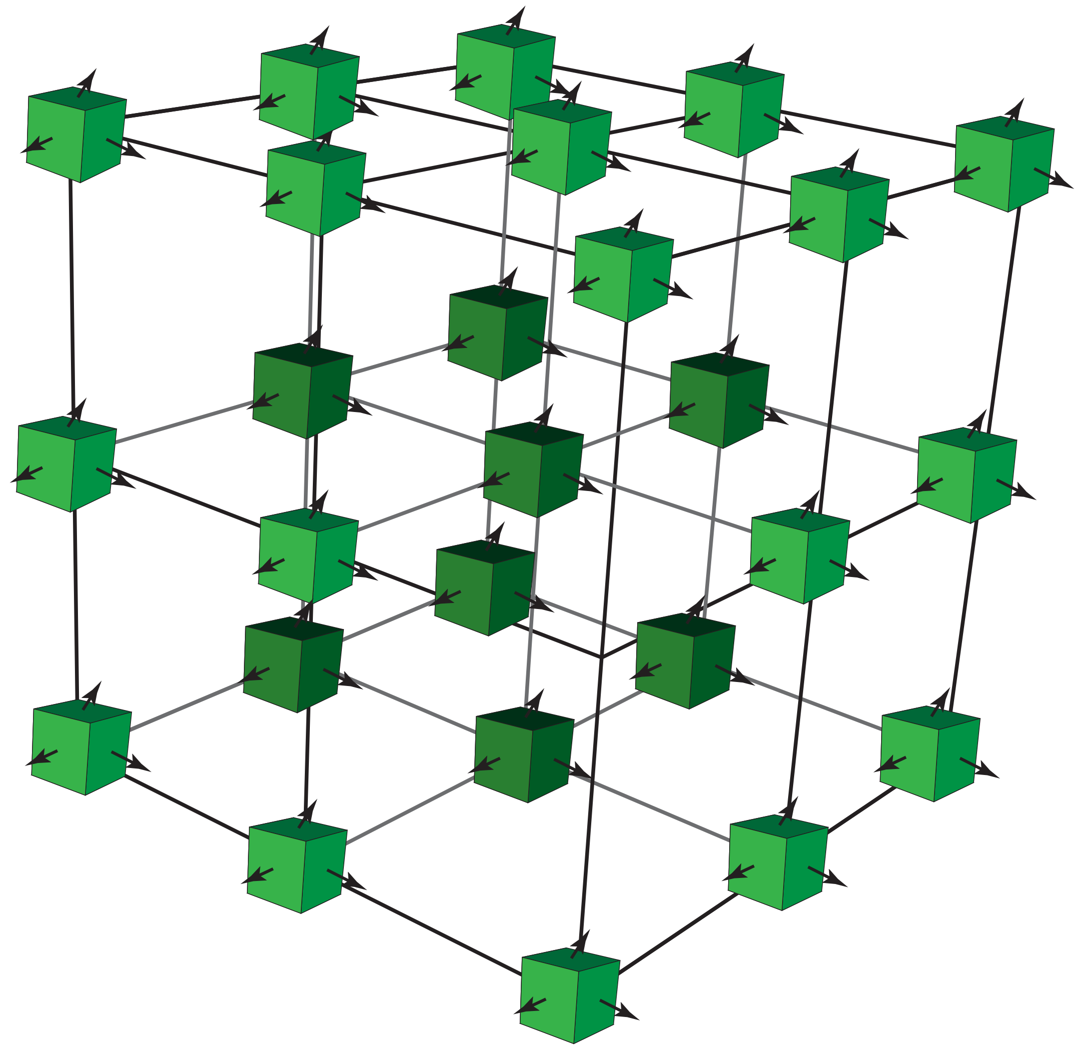
Scalar Fields

(Order-0 Tensor Fields)



Vector Fields

(Order-1 Tensor Fields)



Tensor Fields

(Order-2+)

Each point in space has an associated...

$$s_0$$

Scalar

$$\begin{bmatrix} v_0 \\ v_1 \\ v_2 \end{bmatrix}$$

Vector

$$\begin{bmatrix} \sigma_{00} & \sigma_{01} & \sigma_{02} \\ \sigma_{10} & \sigma_{11} & \sigma_{12} \\ \sigma_{20} & \sigma_{21} & \sigma_{22} \end{bmatrix}$$

Tensor

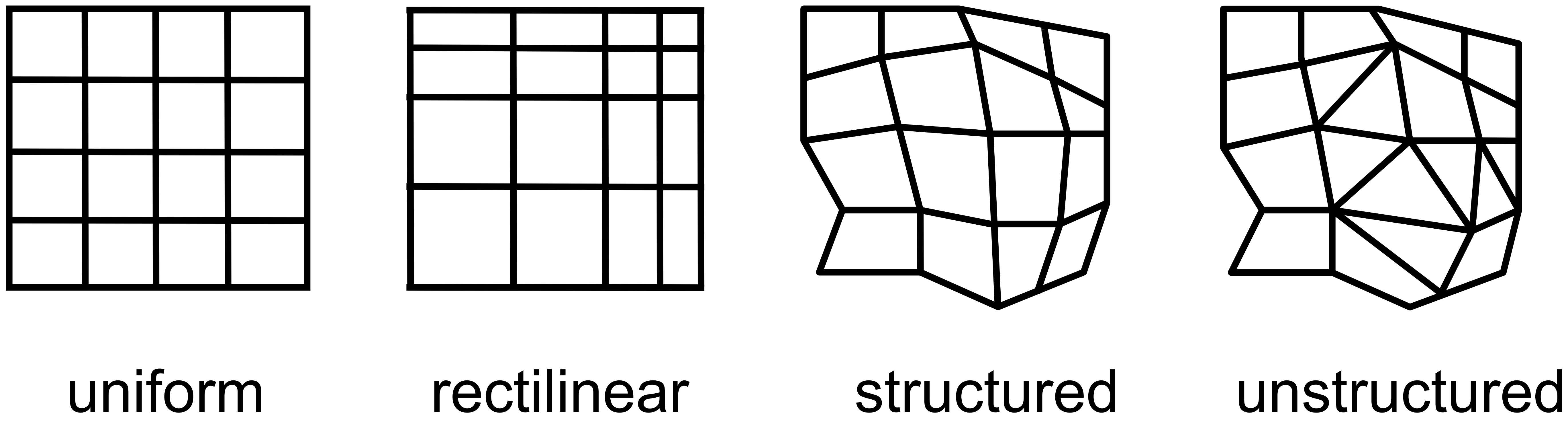
Grids

- Remember we have continuous data and want to sample it in order to understand the **entire** domain
- Possible schemes?

- Geometry: the spatial positions of the data (points)

Grids

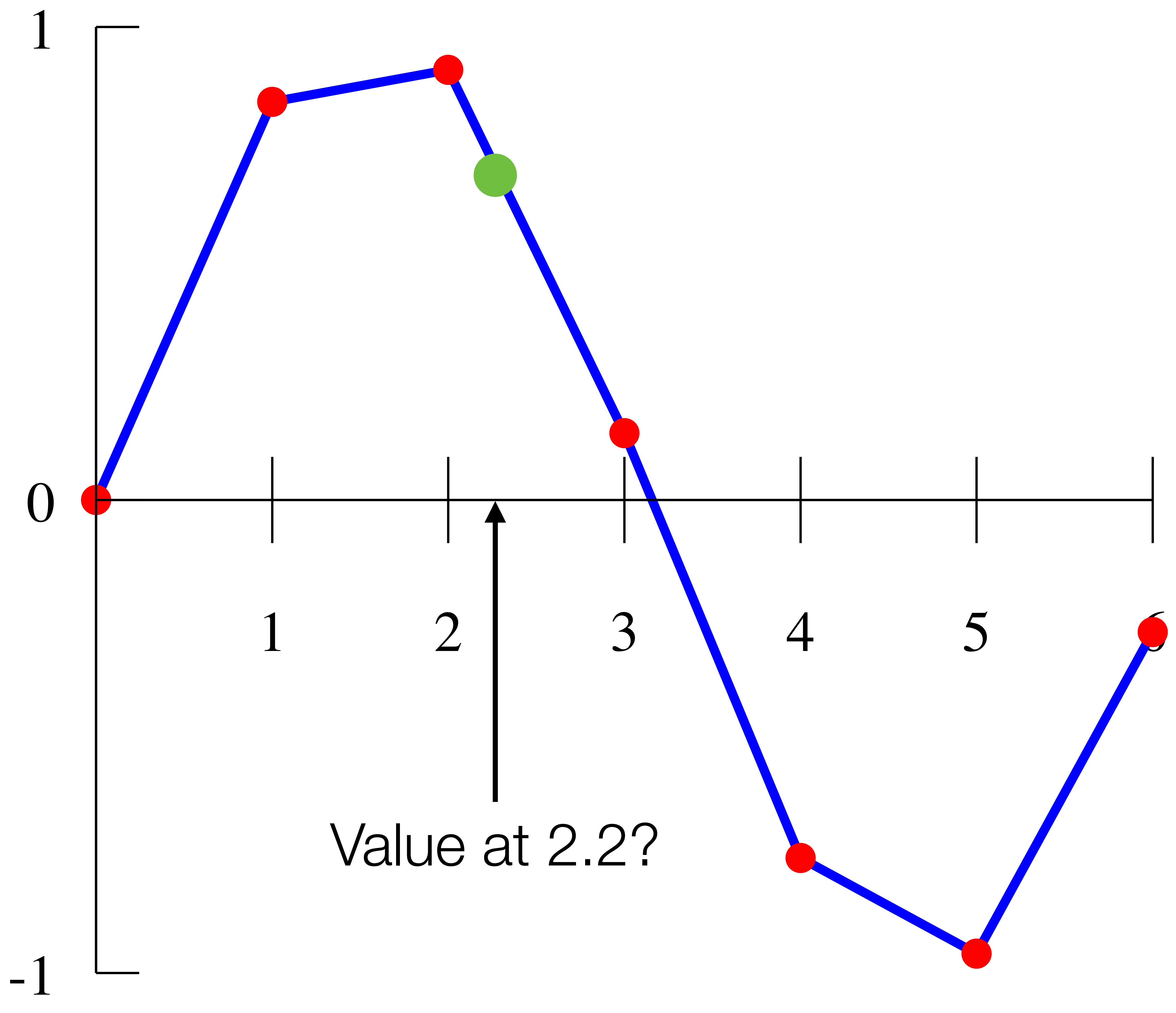
- Remember we have continuous data and want to sample it in order to understand the **entire** domain
- Possible schemes?



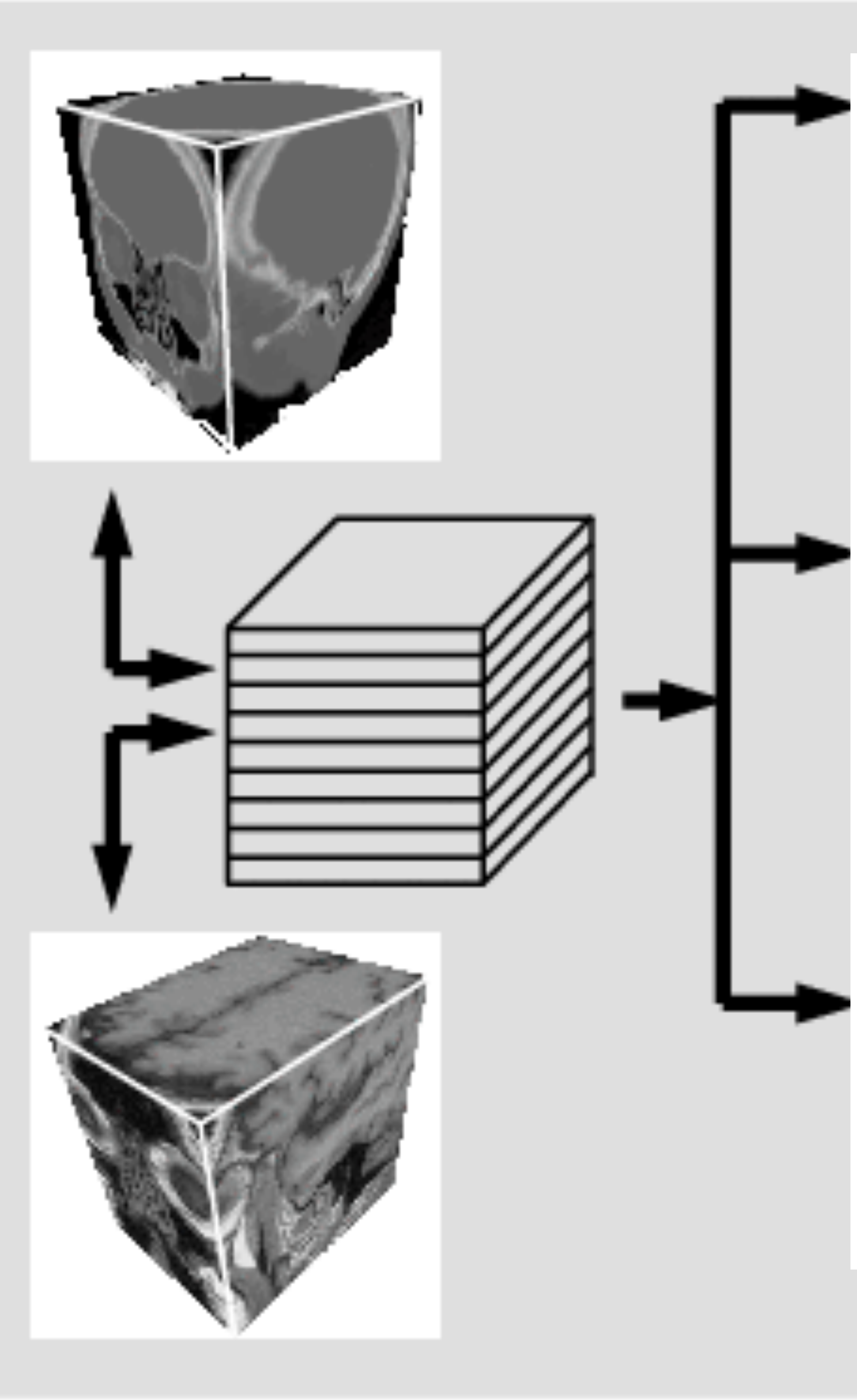
[© Weiskopf/Machiraju/Möller]

- Geometry: the spatial positions of the data (points)
- Topology: how the points are connected (cells)
- Type of grid determines how much data needs to be stored for both geometry and topology

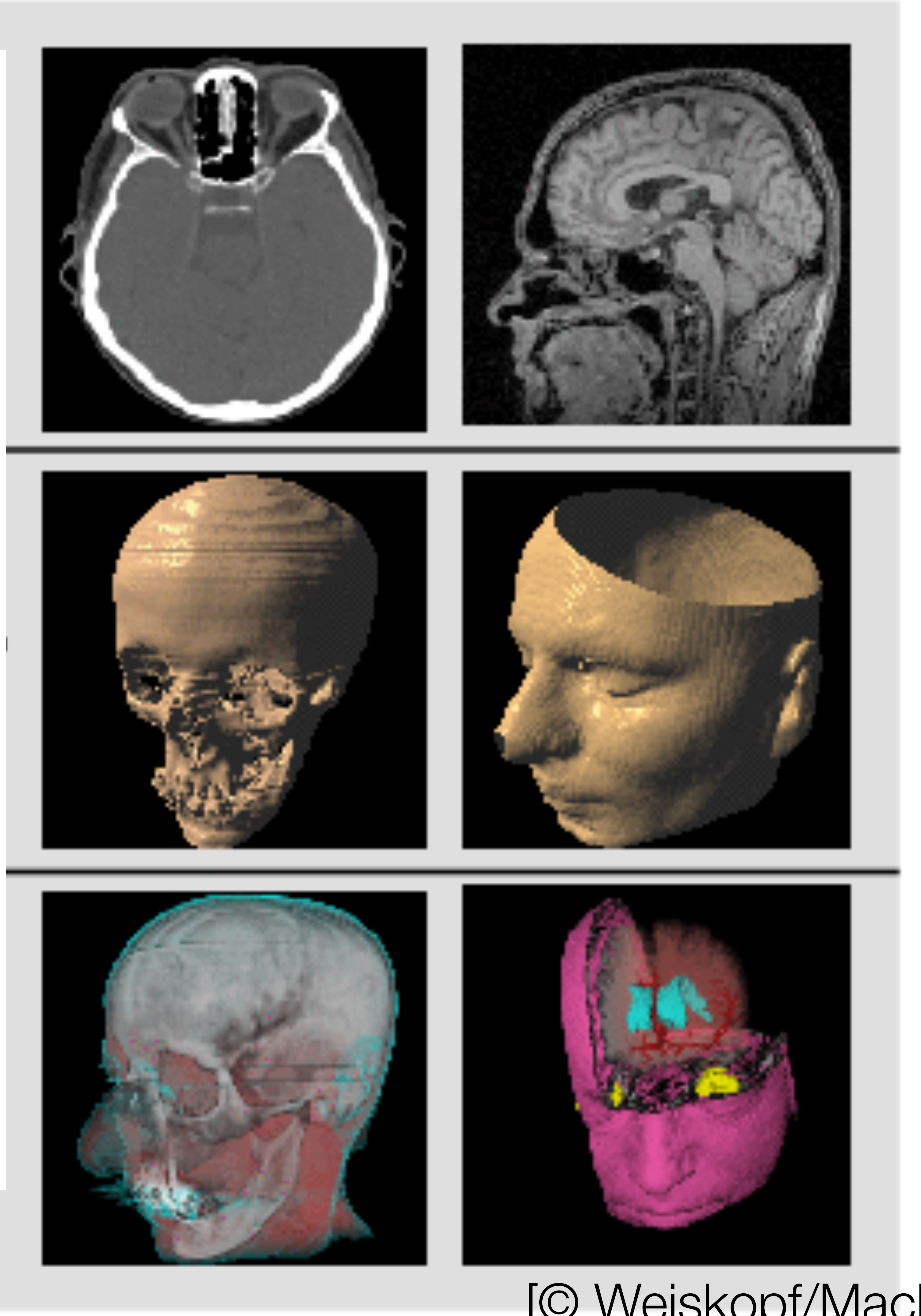
Linear Interpolation



Visualizing Volume (3D) Data



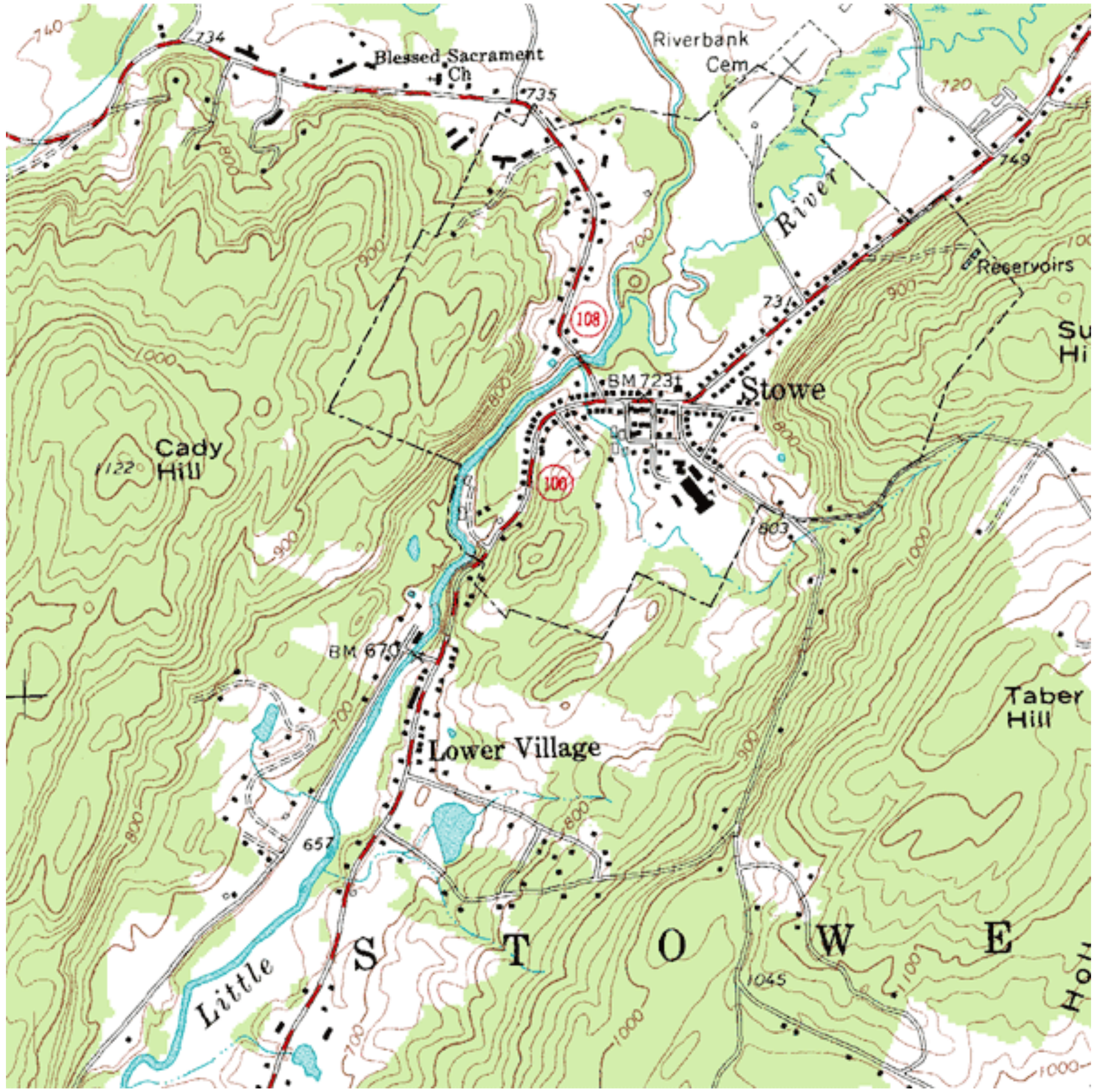
- 2D visualization slice images (or multi-planar reformating MPR)
- *Indirect* 3D visualization isosurfaces (or surface-shaded display SSD)
- *Direct* 3D visualization (direct volume rendering DVR)



[© Weiskopf/Machiraju/Möller]

Isolines (2D)

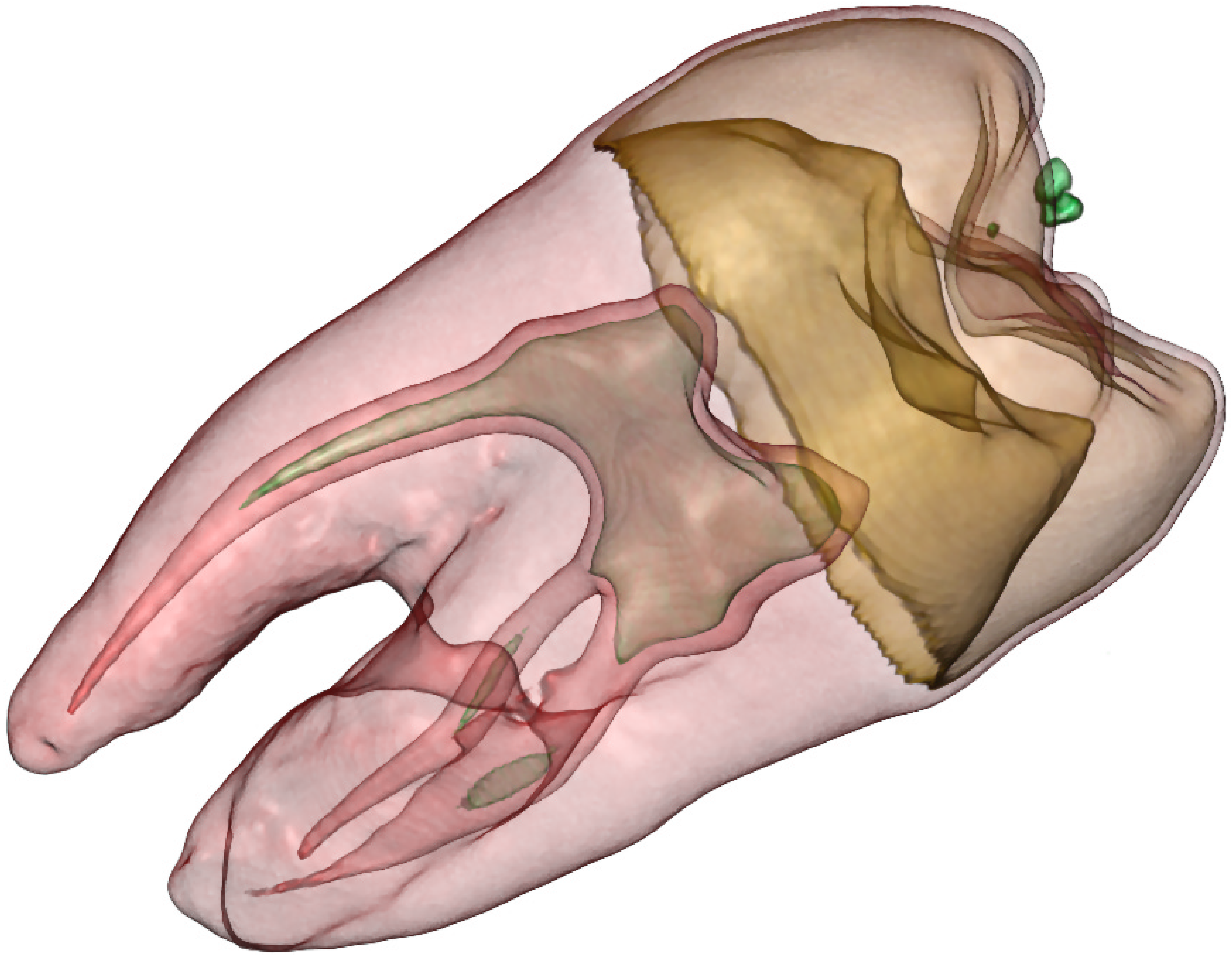
- Isoline: a line that has the same scalar value at all locations
- Example: Topographical Map



[USGS via Wikipedia]

Isosurfaces (3D)

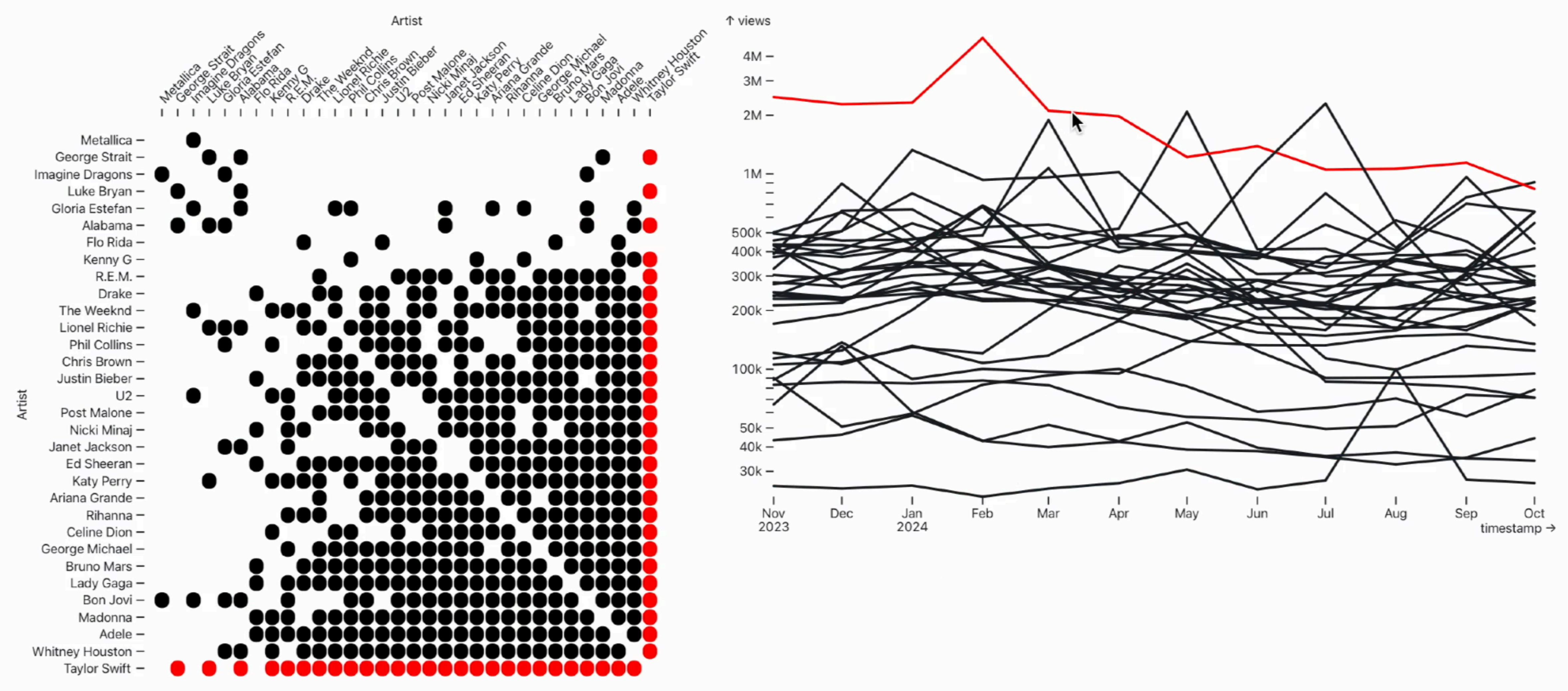
- Isosurface: a surface that has the same scalar value at all locations
- Often use multiple isosurfaces to show different levels



[J. Kniss, 2002]

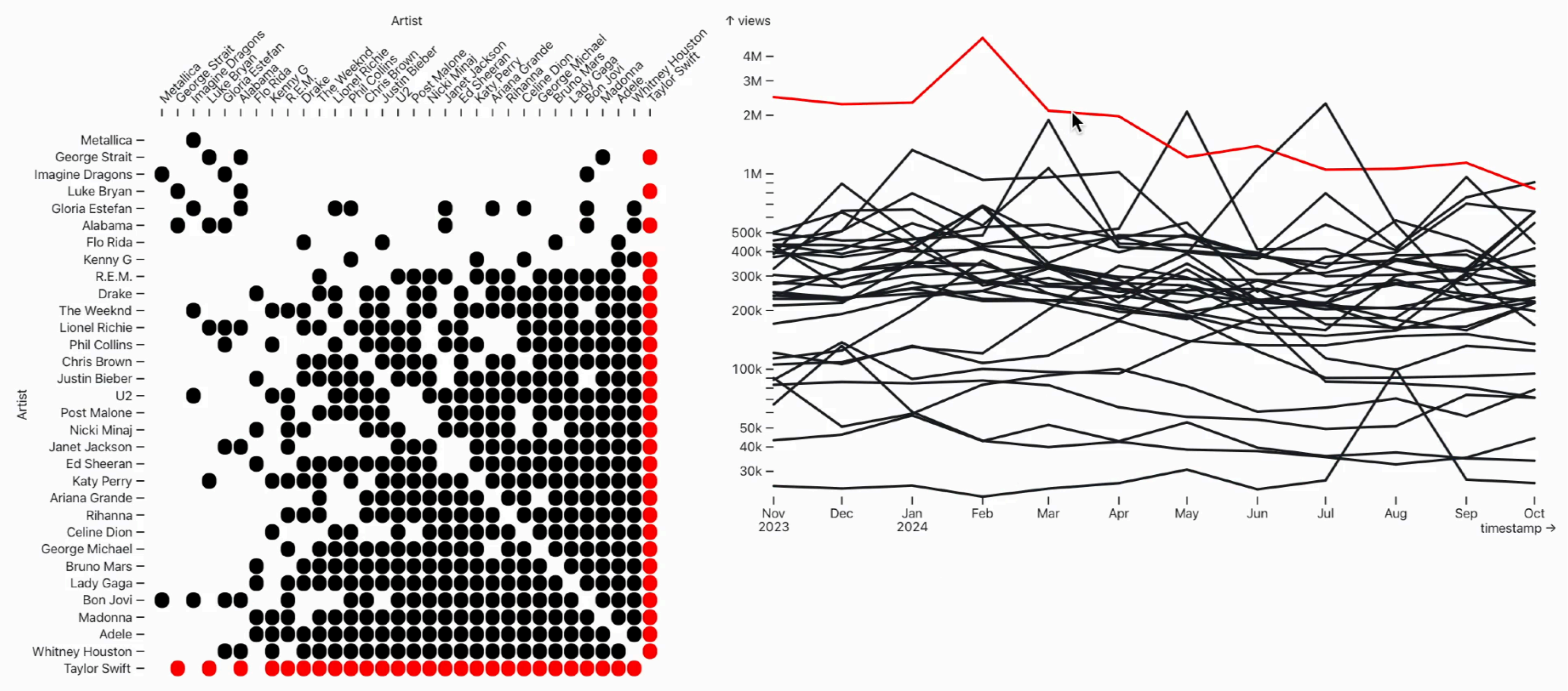
Assignment 5

- Adjacency Matrix
- Line Graph
- Linked Highlighting



Assignment 5

- Adjacency Matrix
- Line Graph
- Linked Highlighting



Project

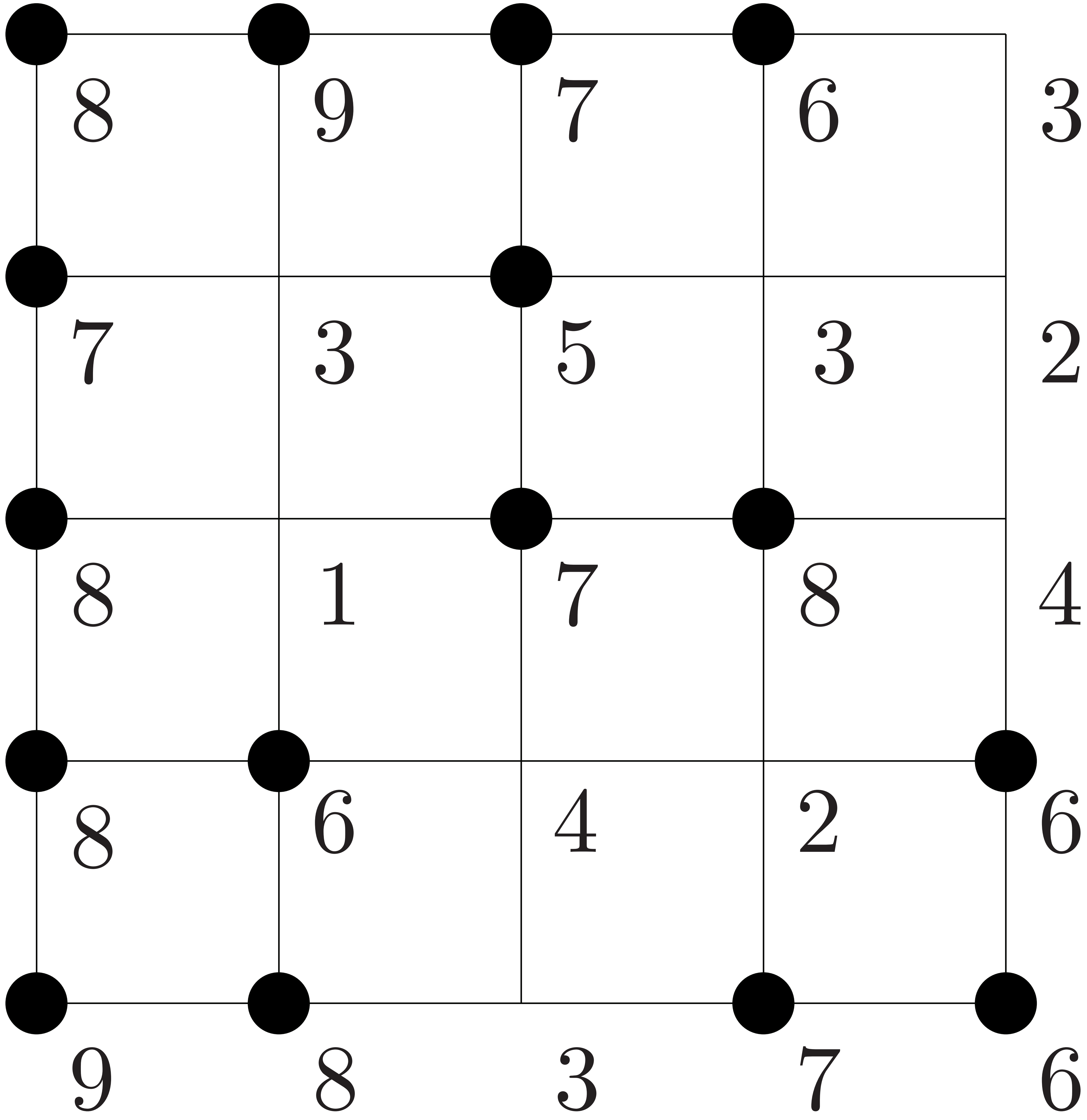
- Feedback from Designs soon
- Keep working on implementation
- Be creative
- Think about interaction
- Presentations on the last two days of class (Dec. 2 & Dec. 4)
  - Submit current visualization code (or a link) to Blackboard
  - Presentation preferences (Monday or Wednesday)
  - Upload full code to Blackboard beforehand in case of technical issues
- Can keep working on final project & report until end of semester

Generating Isolines (Isovalue = 5)

8	9	7	6	3
7	3	5	3	2
8	1	7	8	4
8	6	4	2	6
9	8	3	7	6

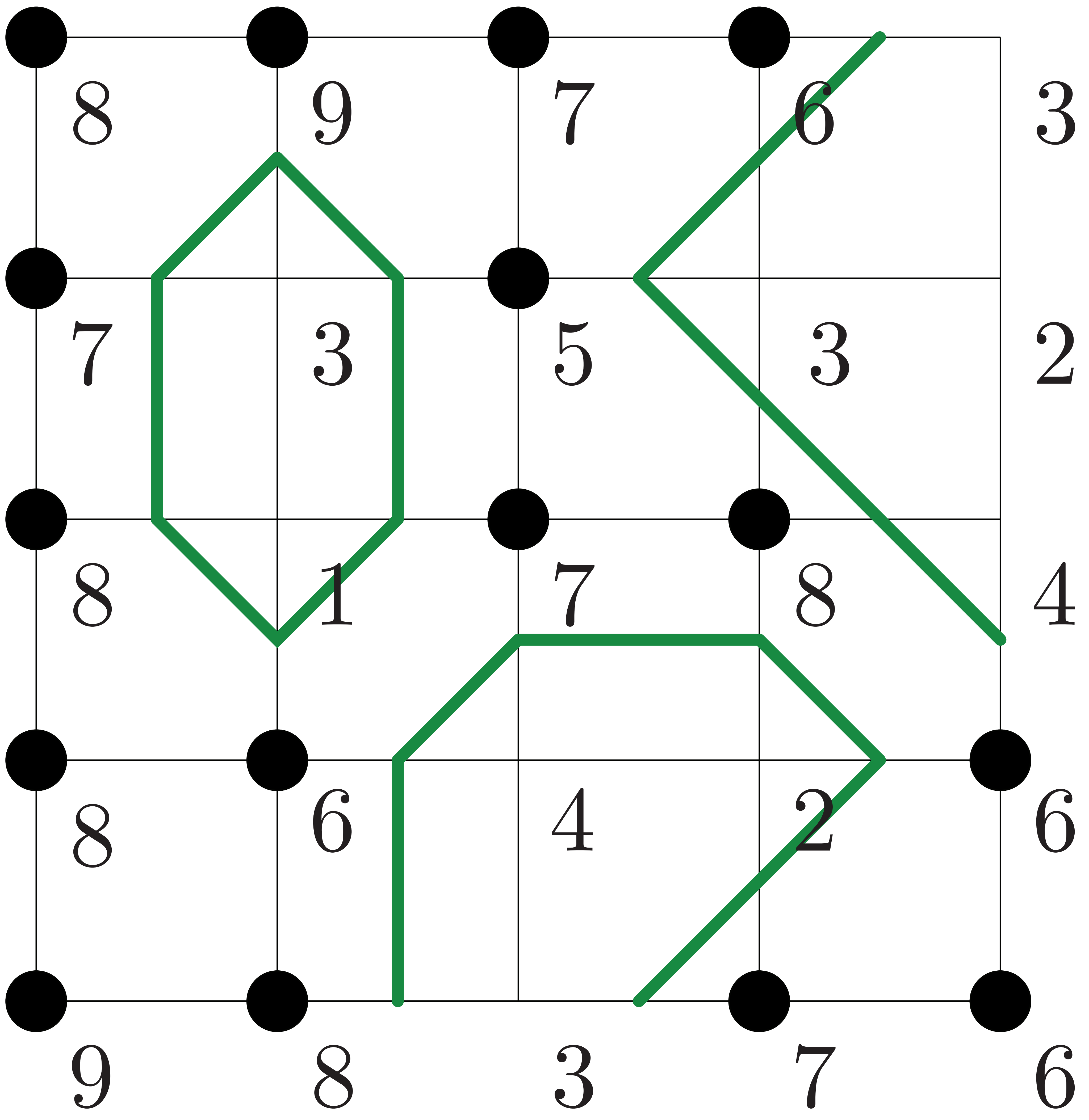
[R. Wenger, 2013]

Generating Isolines



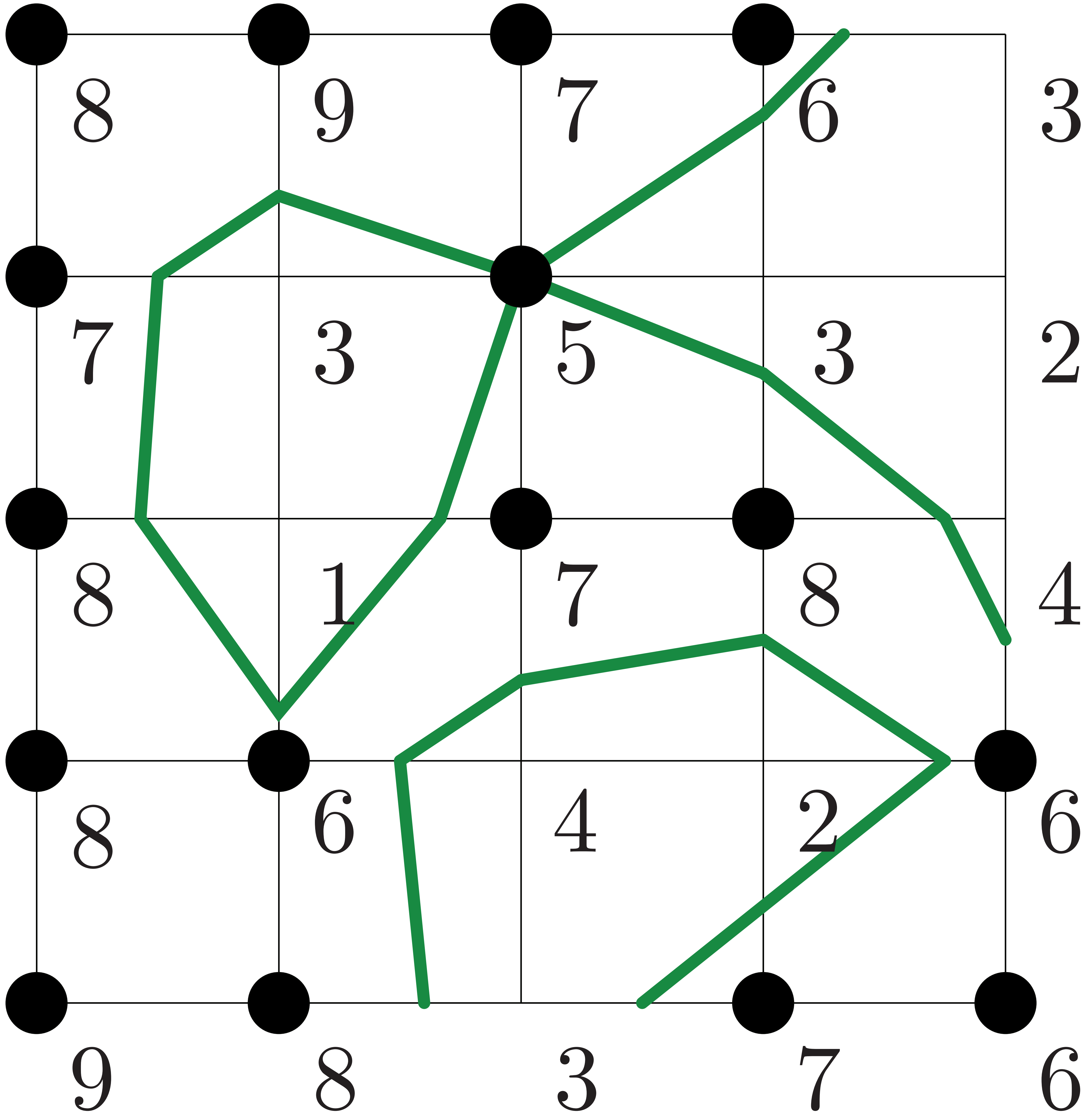
[R. Wenger, 2013]

Generating Isolines



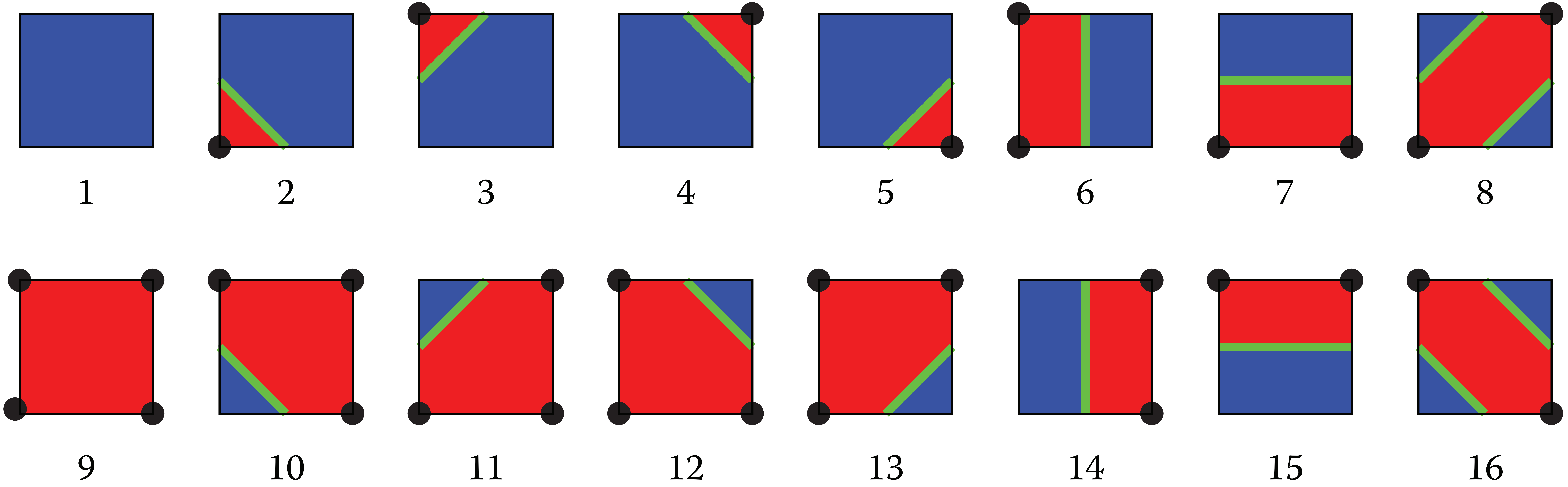
[R. Wenger, 2013]

Generating Isolines



[R. Wenger, 2013]

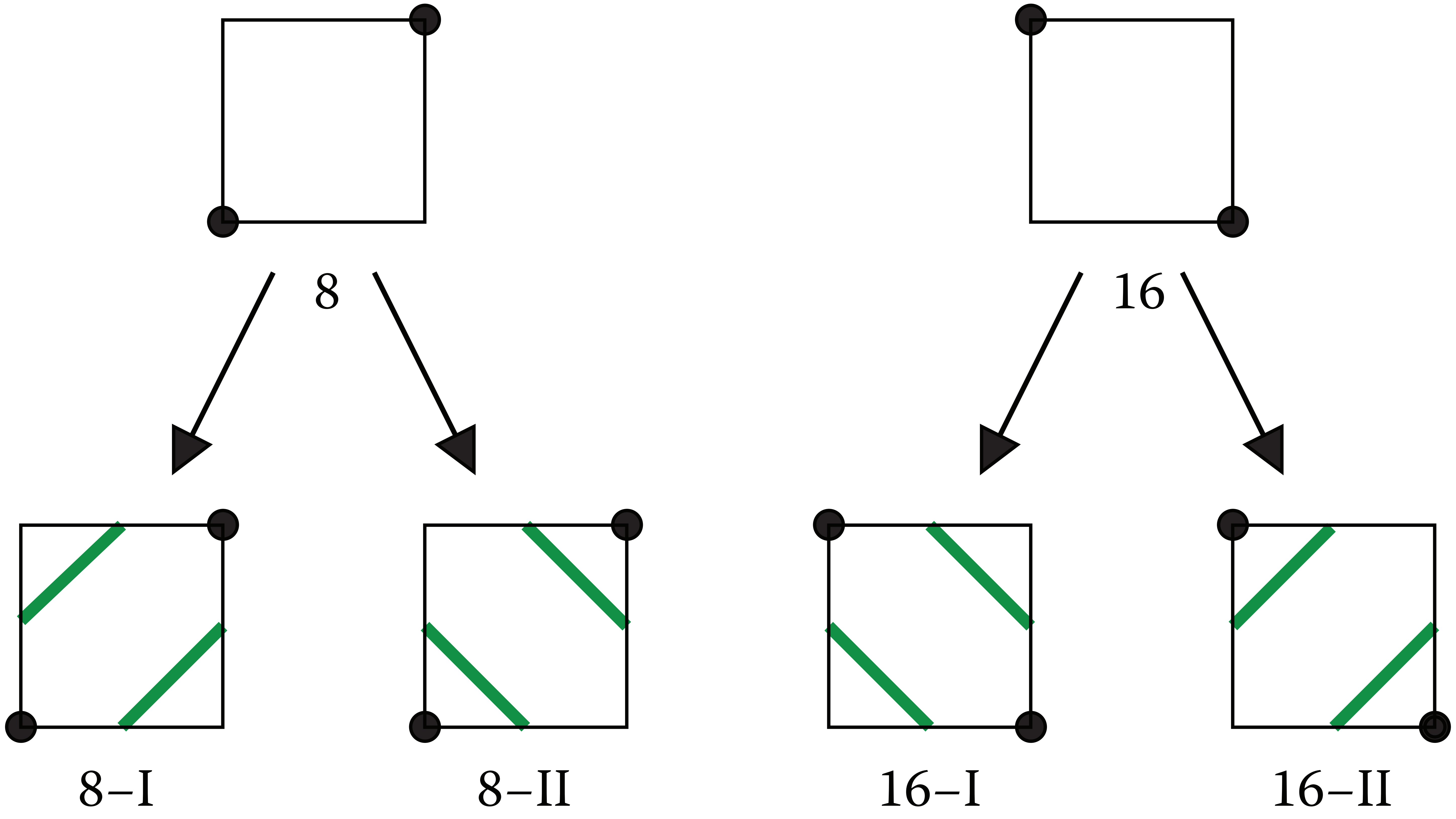
Marching Squares



[R. Wenger, 2013]

Ambiguous Configurations

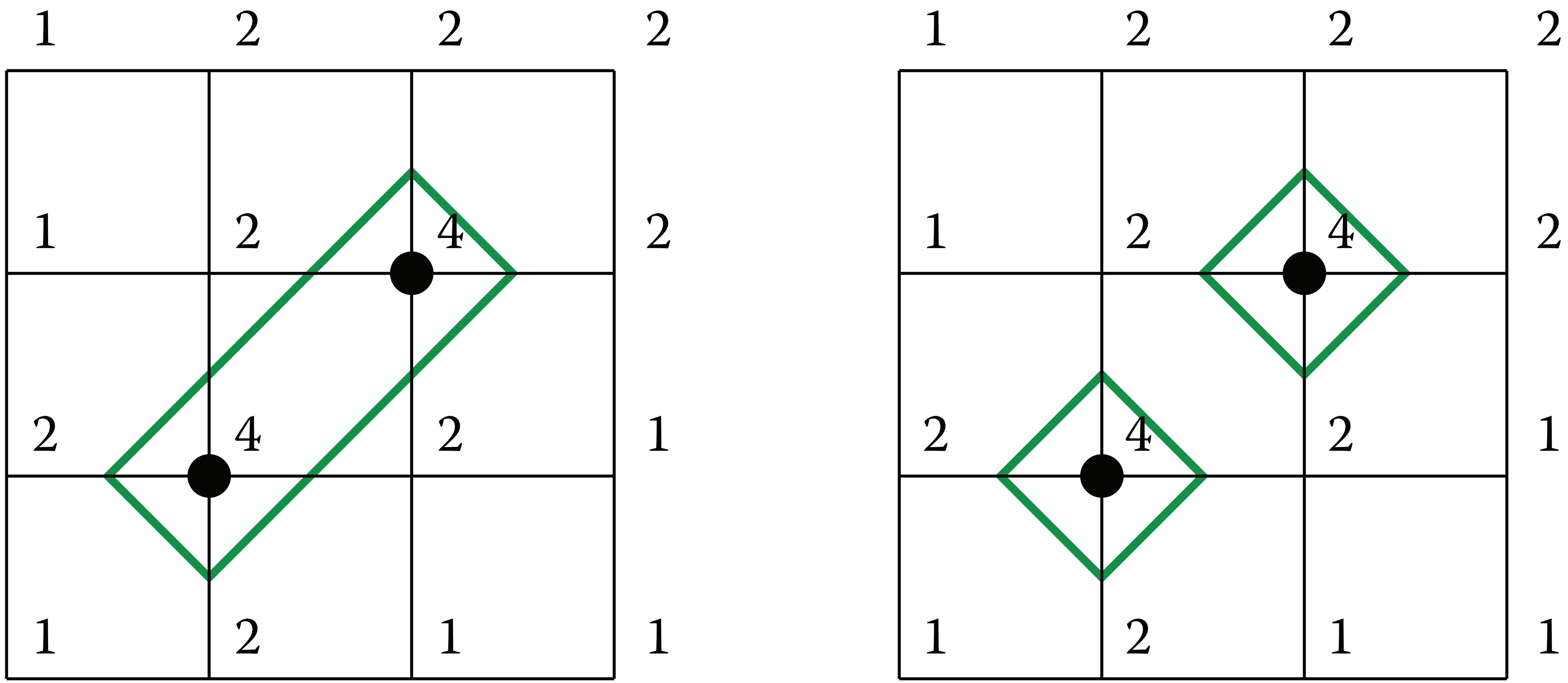
- There are some cases for which we cannot tell which way to draw the isolines…



[R. Wenger, 2013]

Ambiguous Configurations

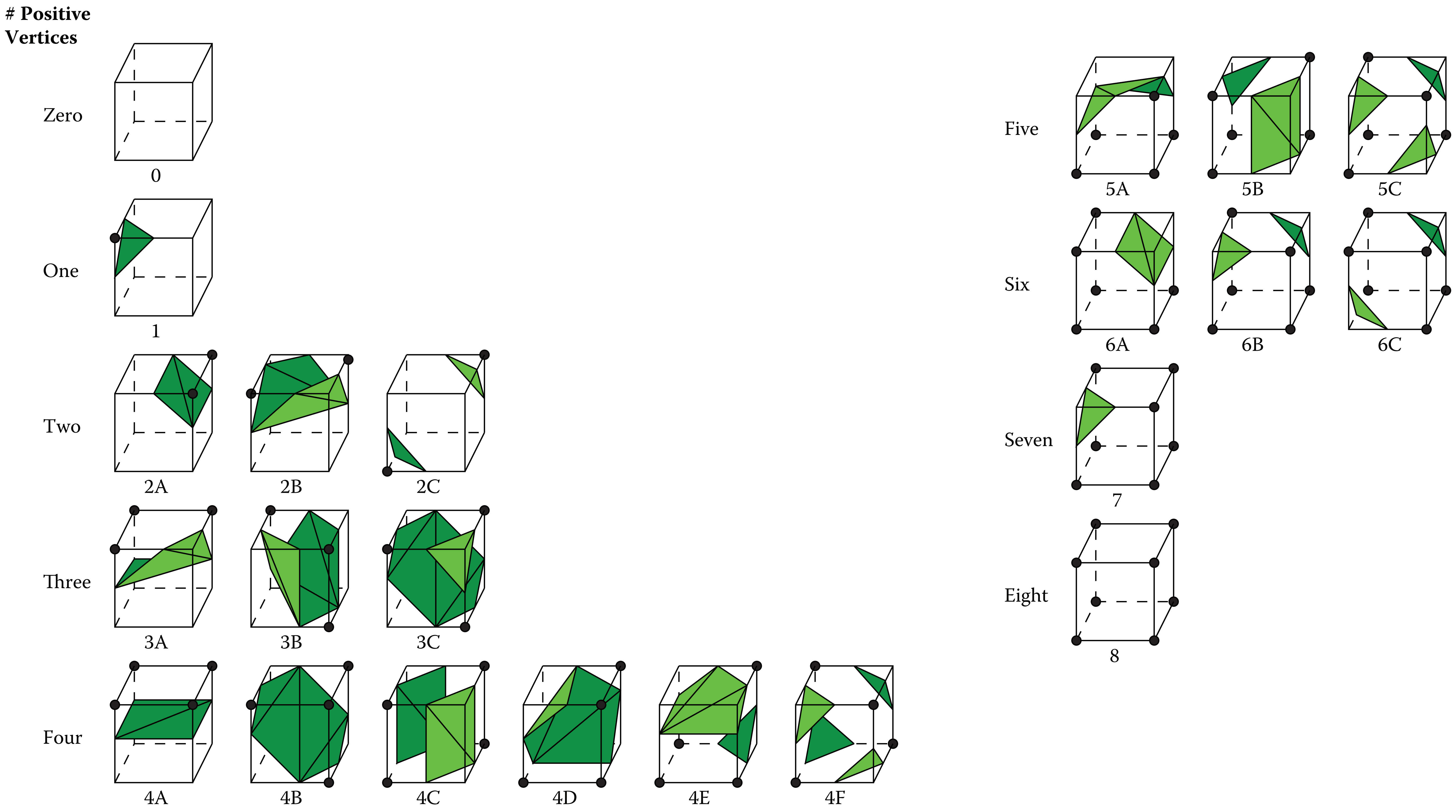
- Either works for marching squares, this isn't the case for 3D



[R. Wenger, 2013]

3D: Marching Cubes

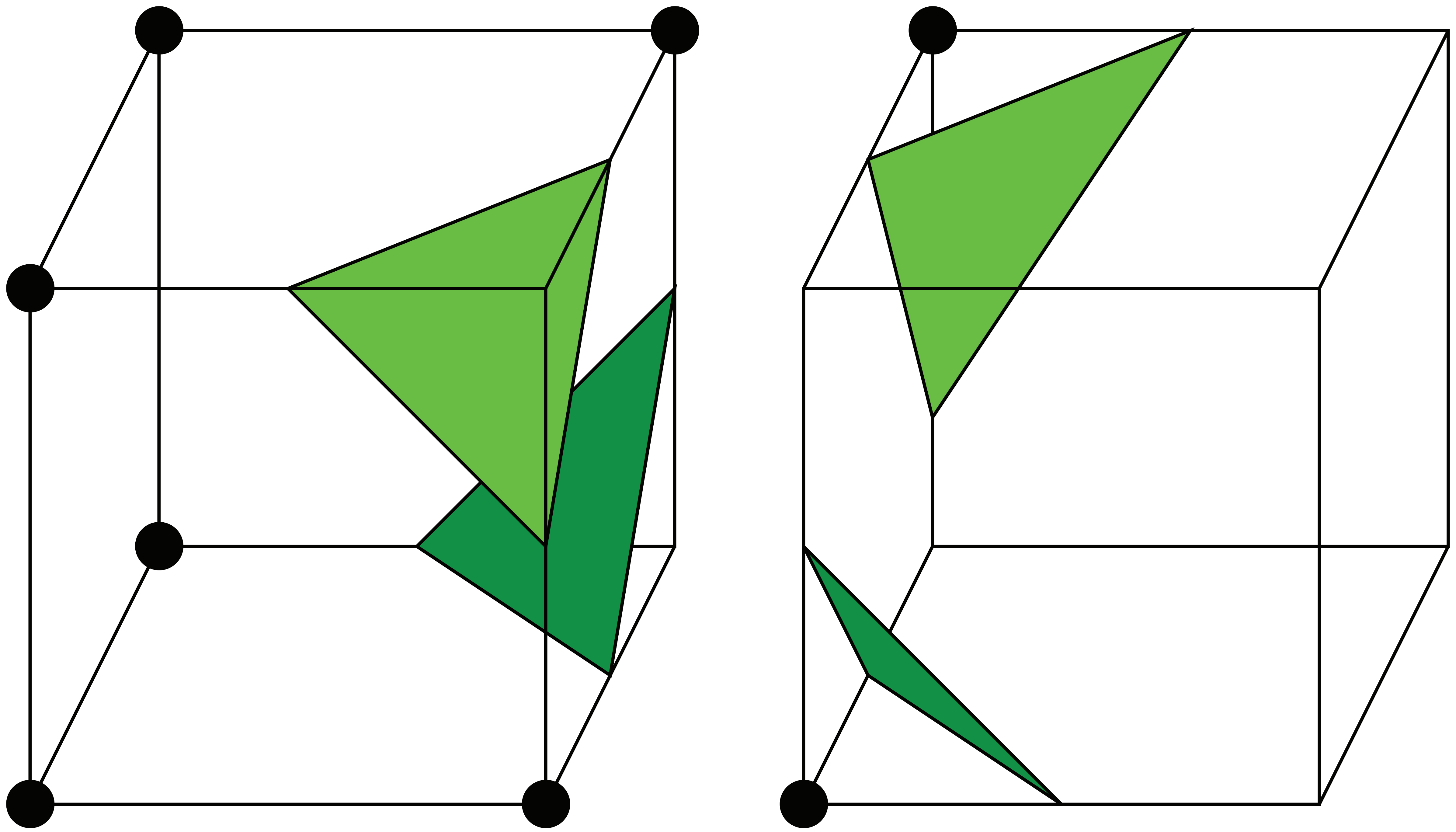
- Same idea, more cases [Lorensen and Cline, 1987]



[R. Wenger, 2013]

Incompatible Choices

- If we have ambiguous cases where we choose differently for each cell, the surfaces will not match up correctly—there are holes
- Fix with the **asymptotic decider** [Nielson and Hamann,1991]



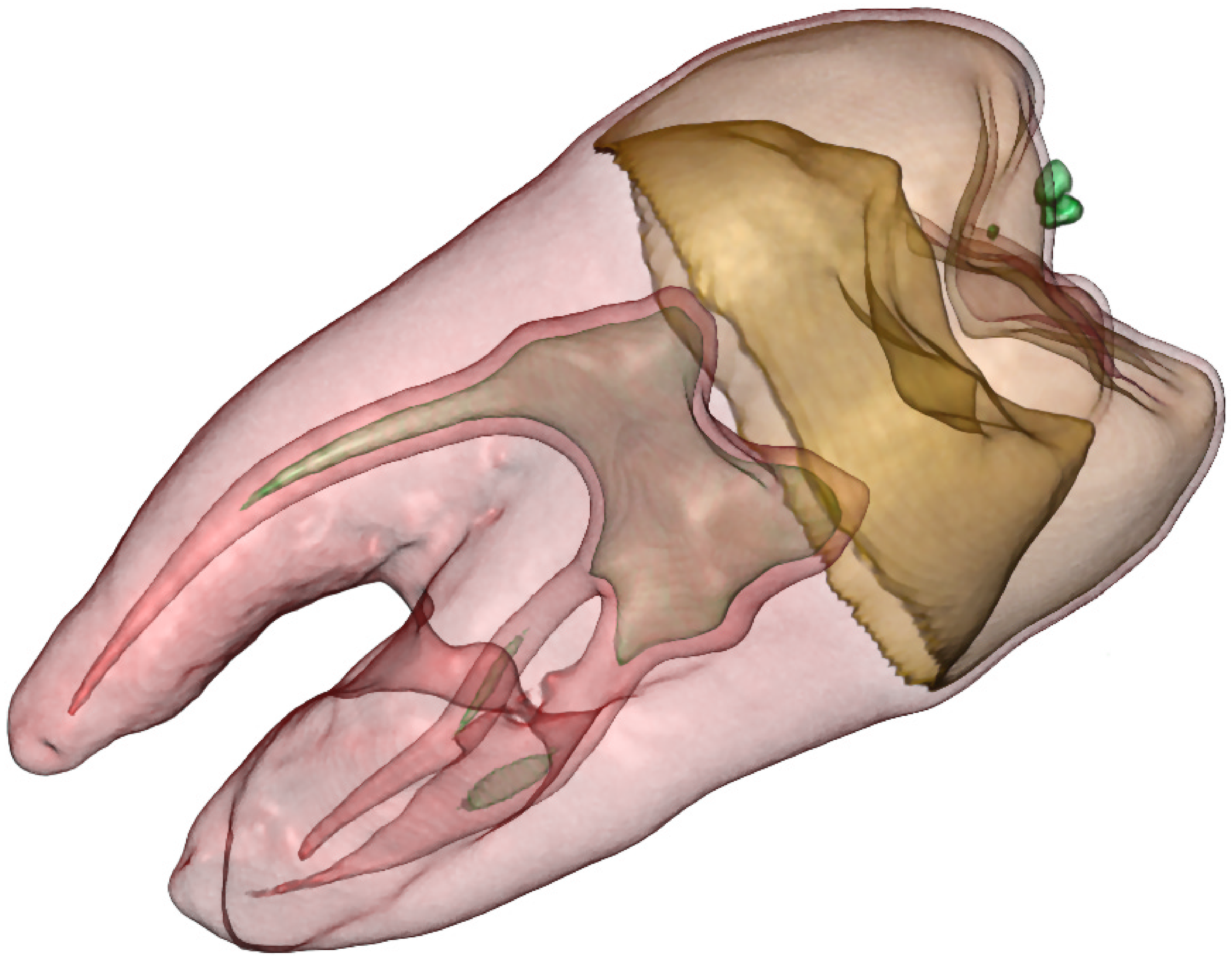
[R. Wenger, 2013]

Marching Cubes Algorithm

- For each cell:
  - Classify each vertex as inside or outside (>=, <) — 0 or 1
  - Take the eight vertex classifications as a bit string
  - Use the bit string as a lookup into a table to get edges
  - Interpolate to get actual edge locations
  - Compute gradients
  - Resolve ambiguities
- Render a bunch of triangles: easy for graphics cards

Multiple Isosurfaces

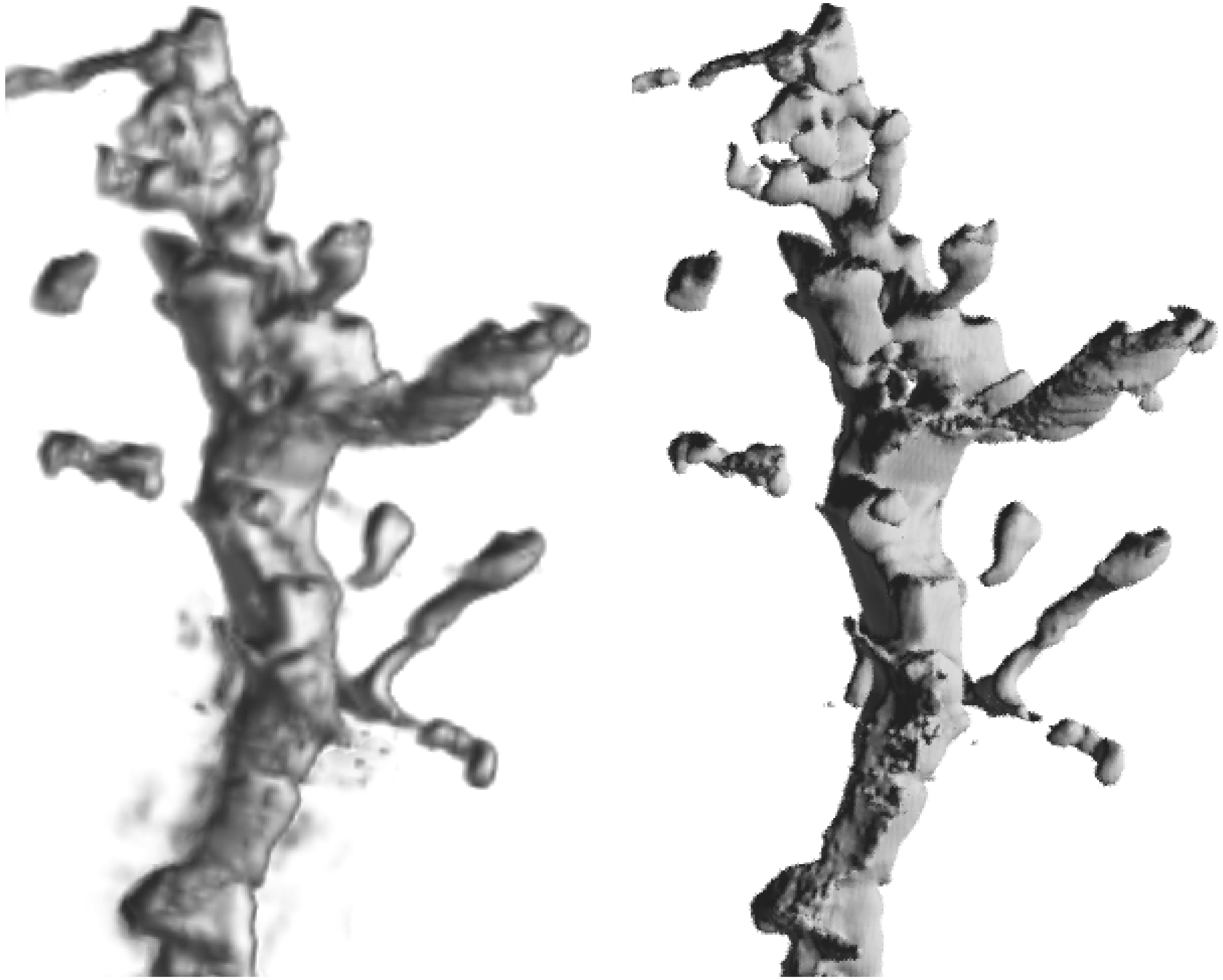
- Topographical maps have multiple isolines to show elevation trends
- Problem in 3D? **Occlusion**
- Solution? Transparent surfaces
- Issues:
  - Think about color in order to make each surface visible
  - Compositing: how do colors "add up" with multiple surfaces
  - How to determine good isovalues?



[J. Kniss, 2002]

Volume Rendering

Volume Rendering vs. Isosurfacing



(a) Direct volume rendered (b) Isosurface rendered

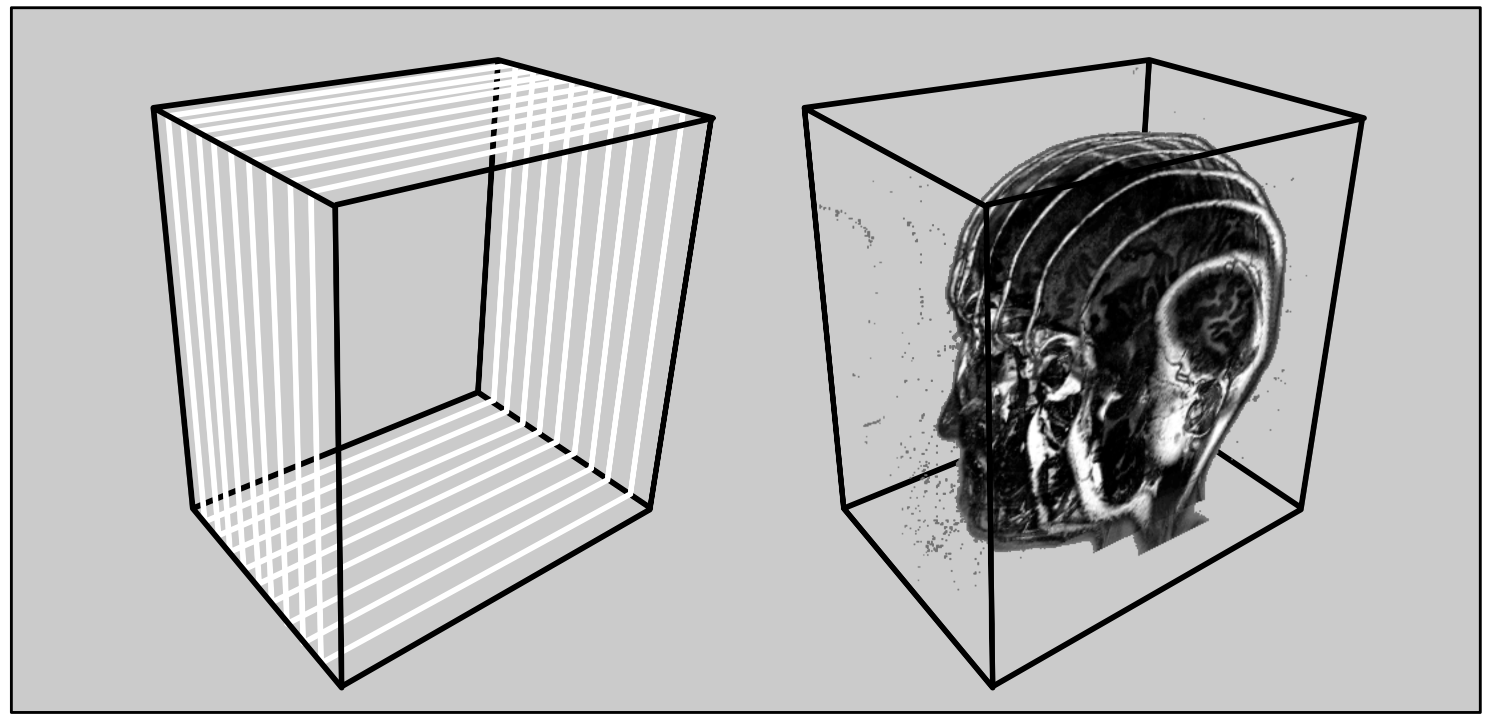
[Kindlmann, 1998]

(Direct) Volume Rendering

- Isosurfacing: compute a surface (triangles) and use standard computer graphics to render the triangles
- Volume rendering: compute the pixels shown directly from the volume information
- Why?
  - No need to figure out precise isosurface boundaries
  - Can work better for data with noise or uncertainty
  - Greater control over appearance based on values

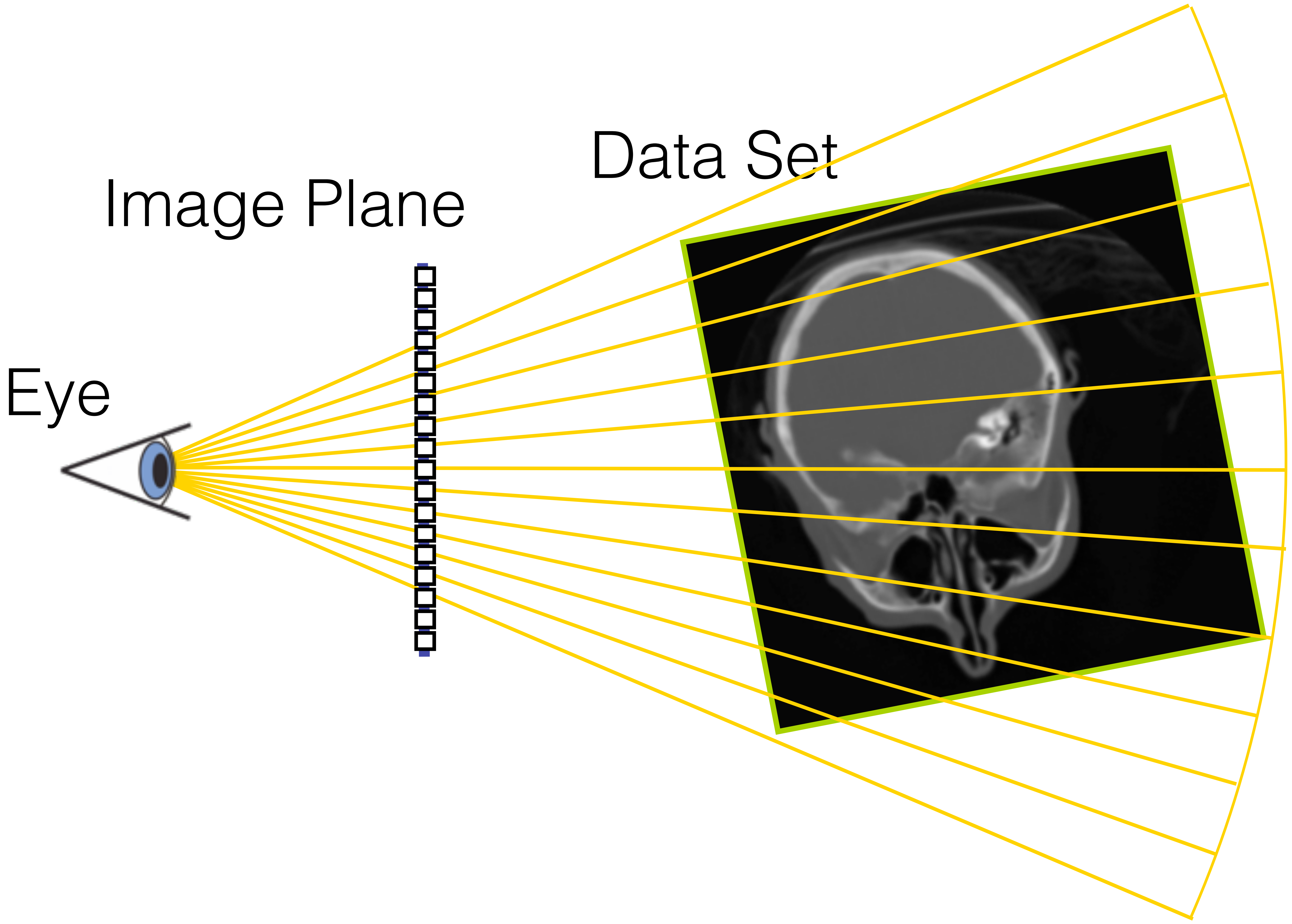
Types of Volume Rendering Algorithms

- Ray casting
  - Similar to ray tracing, but use rays from the viewer
- Splatting:
  - Object-order, voxels splat onto the image plane
- Shear Warp:
  - Object-space, slice-based, parallel viewing rays
- Texture-Based:
  - 2D Slices: stack of texture maps
  - 3D Textures



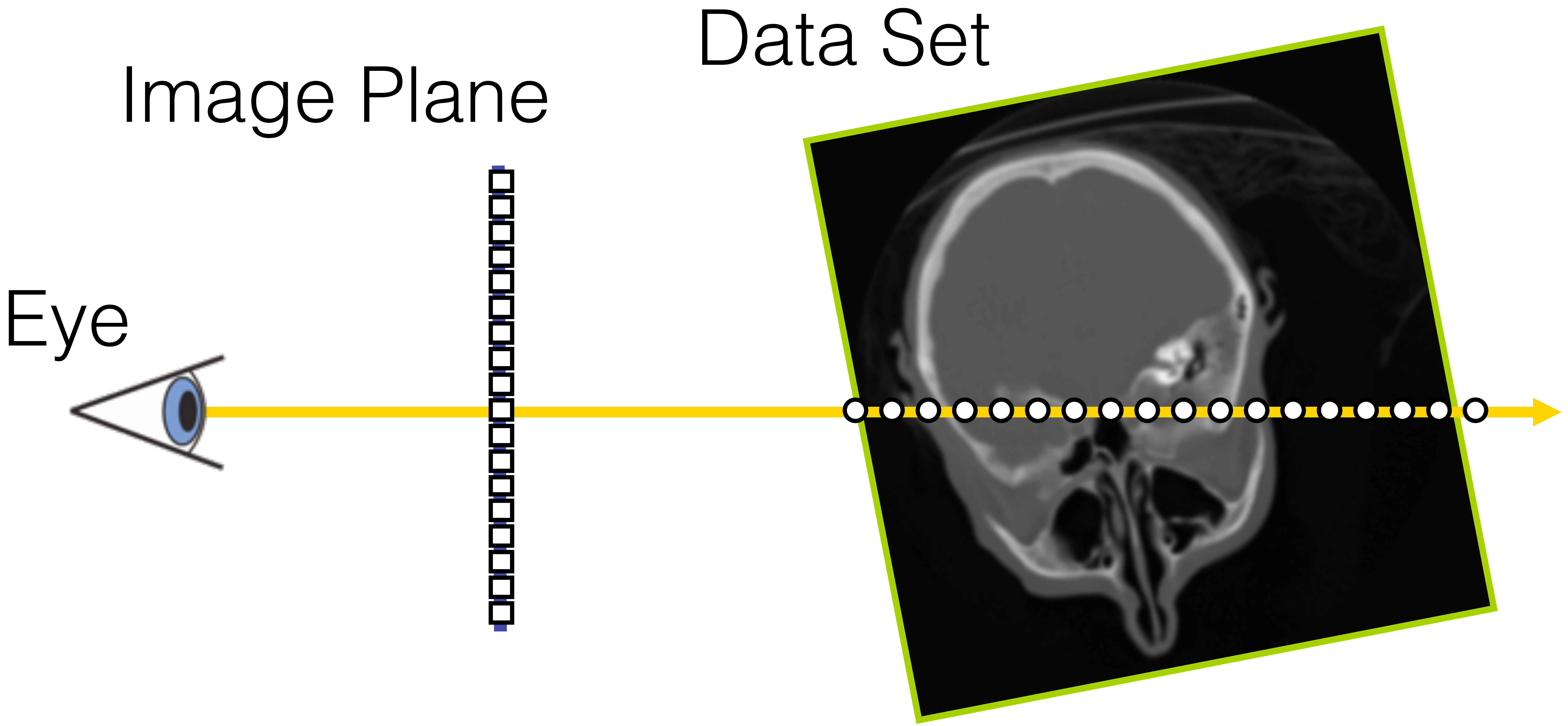
[via Möller]

Volume Ray Casting



[Levine]

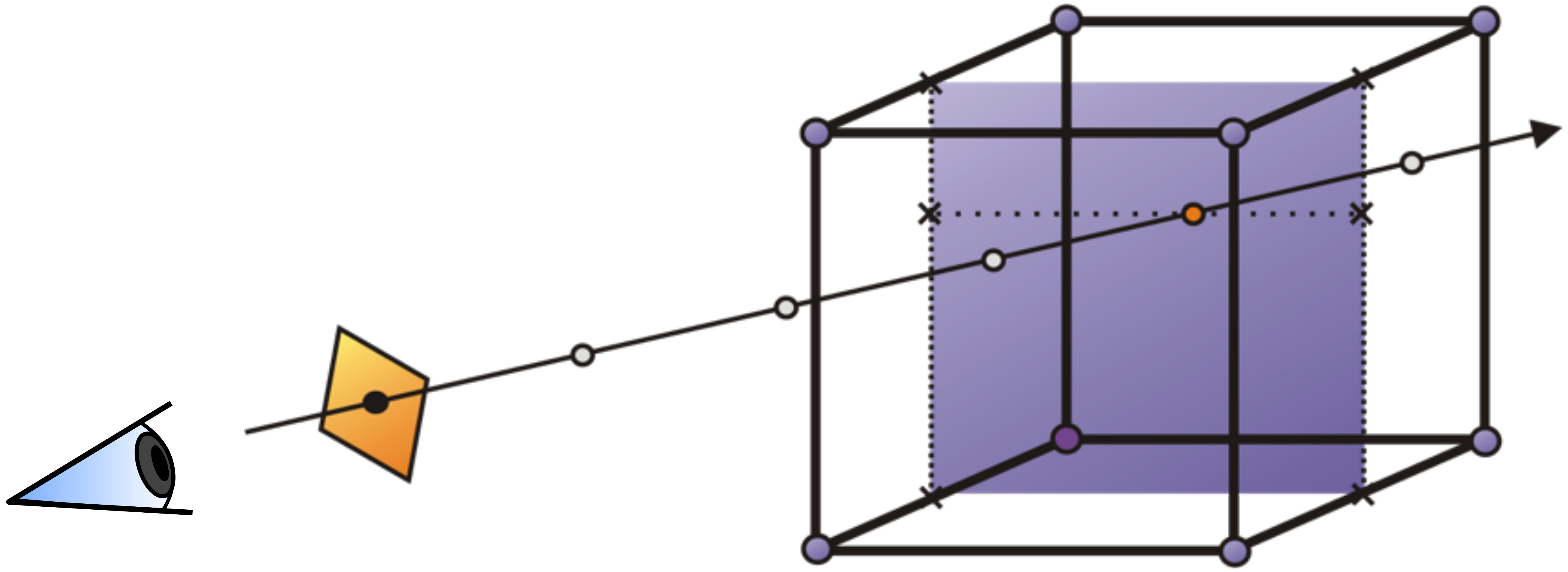
Volume Ray Casting



[Levine]

How?

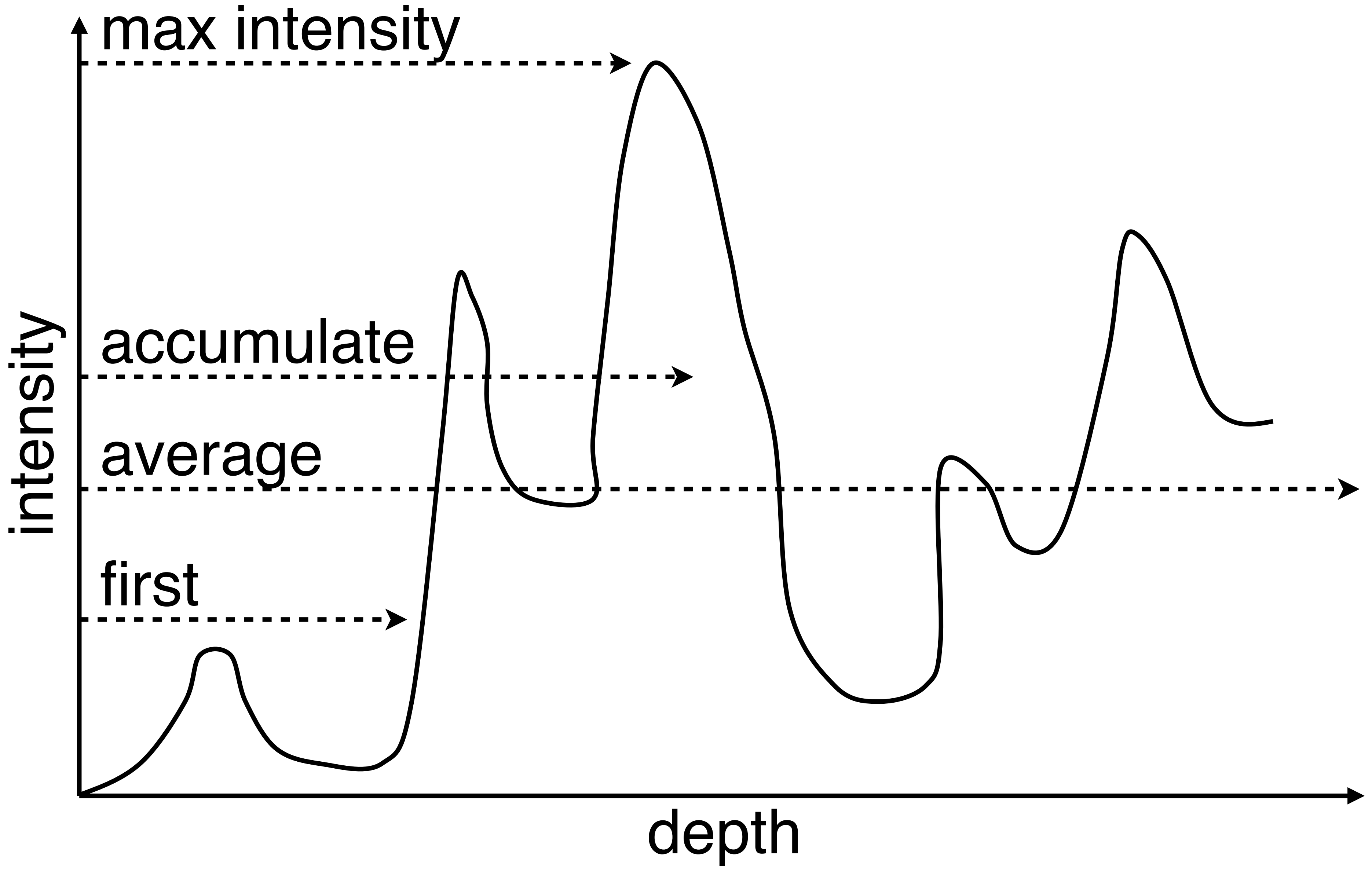
- Approximate volume rendering integral: light absorption & emission
- Sample at regular intervals along each ray
- Trilinear interpolation: linear interpolation along each axes (x,y,z)



- Not the only possibility, also "object order" techniques like splatting or texture-based and combinations like shear-warp

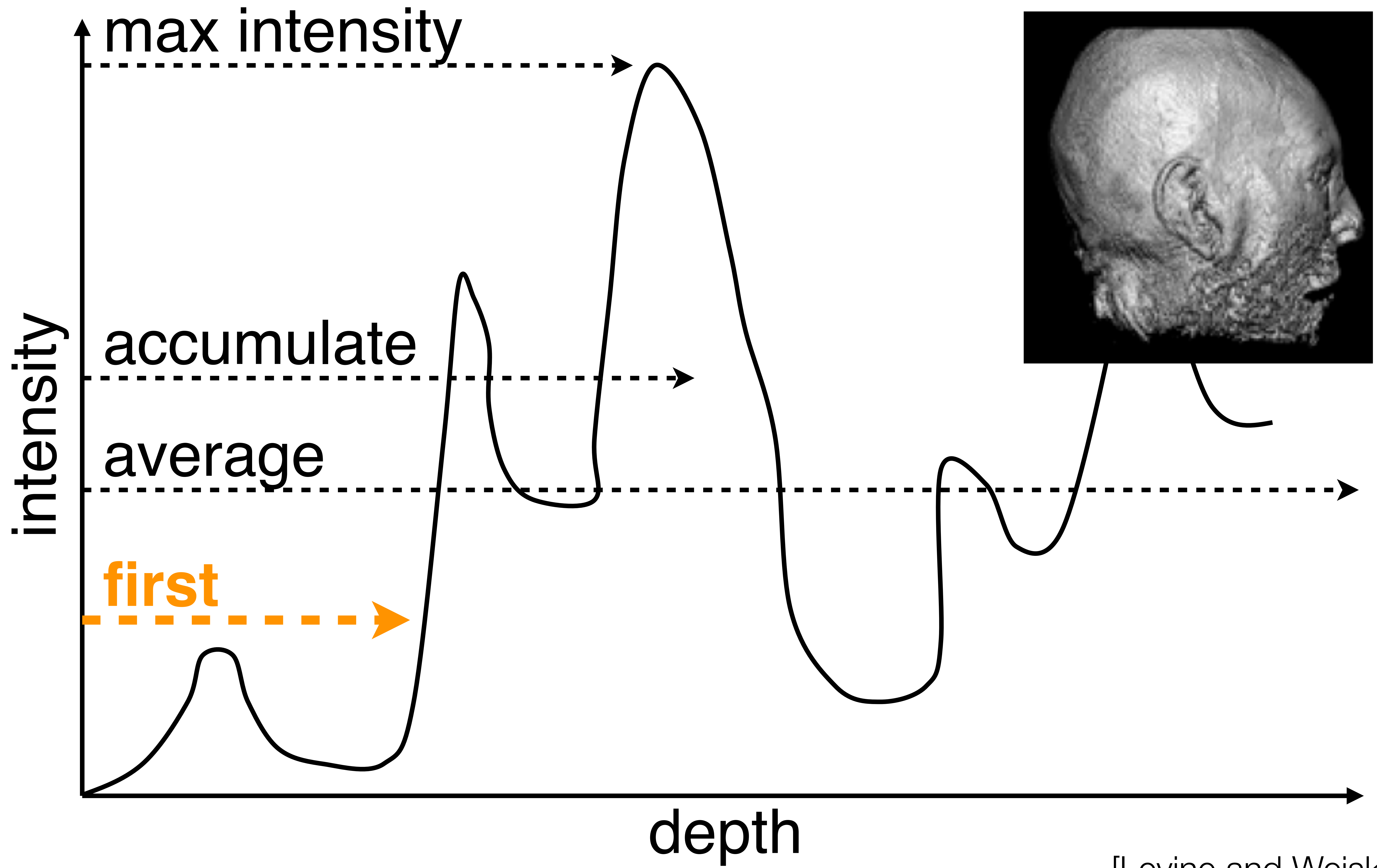
Compositing

- Need **one pixel** from all values along the ray
- Q: How do we "add up" all of those values along the ray?
- A: Compositing!
- Different types of compositing
  - First: like isosurfacing, first intersection at a certain intensity
  - Max intensity: choose highest val
  - Average: mean intensity (density, like x-rays)
  - Accumulate: each voxel has some contribution



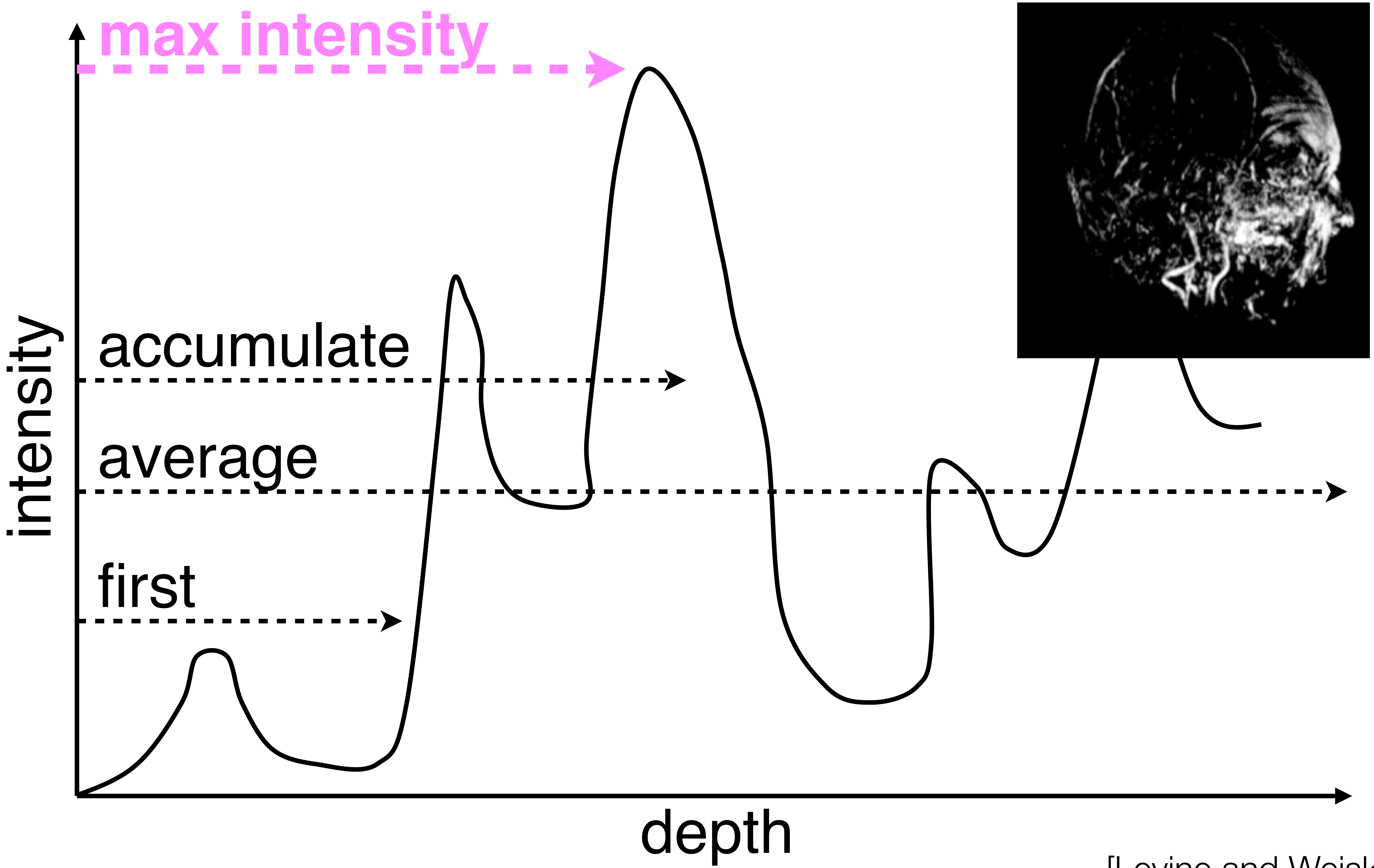
[Levine and Weiskopf/Machiraju/Möller]

Types of Compositing



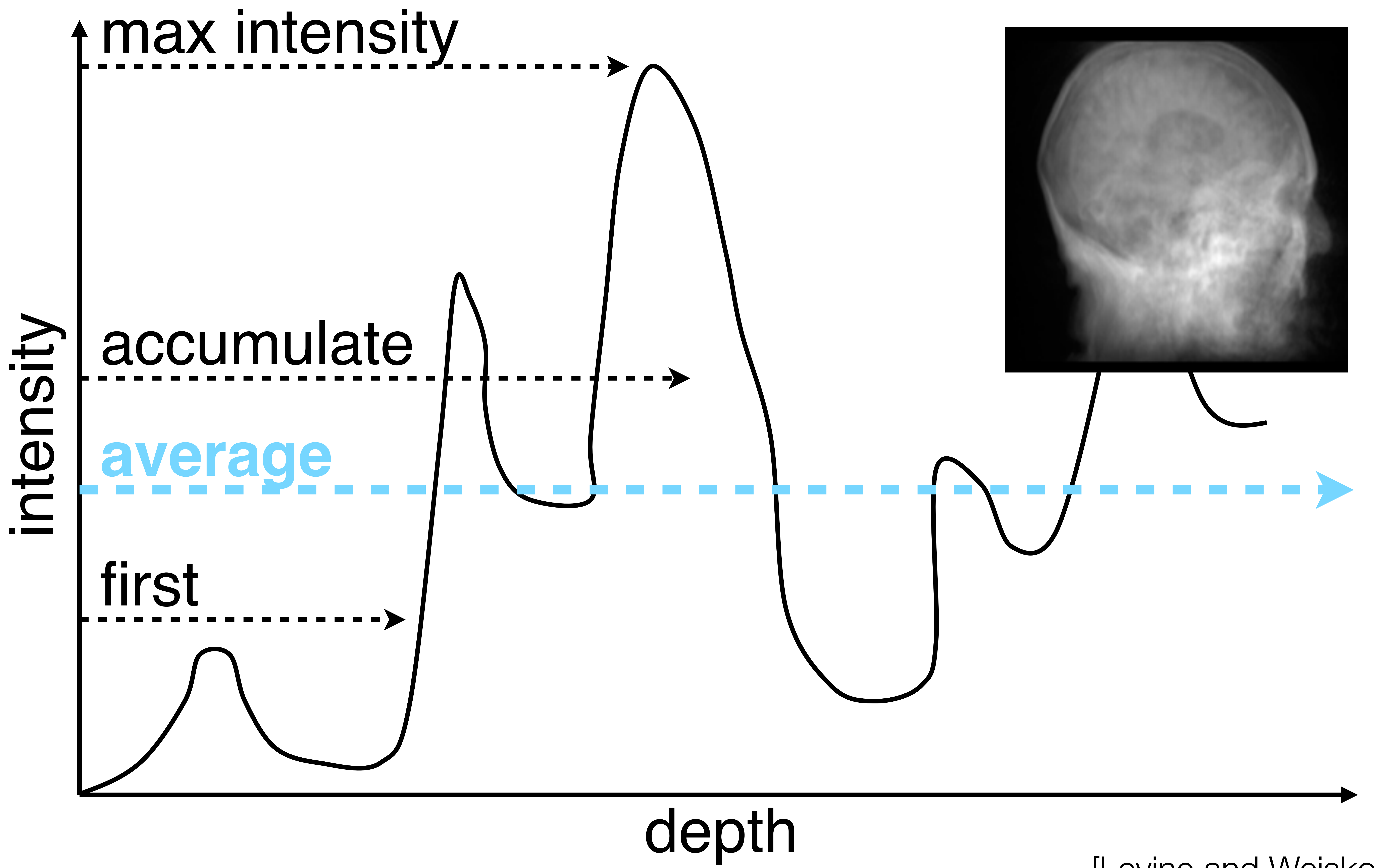
[Levine and Weiskopf/Machiraju/Möller]

Types of Compositing



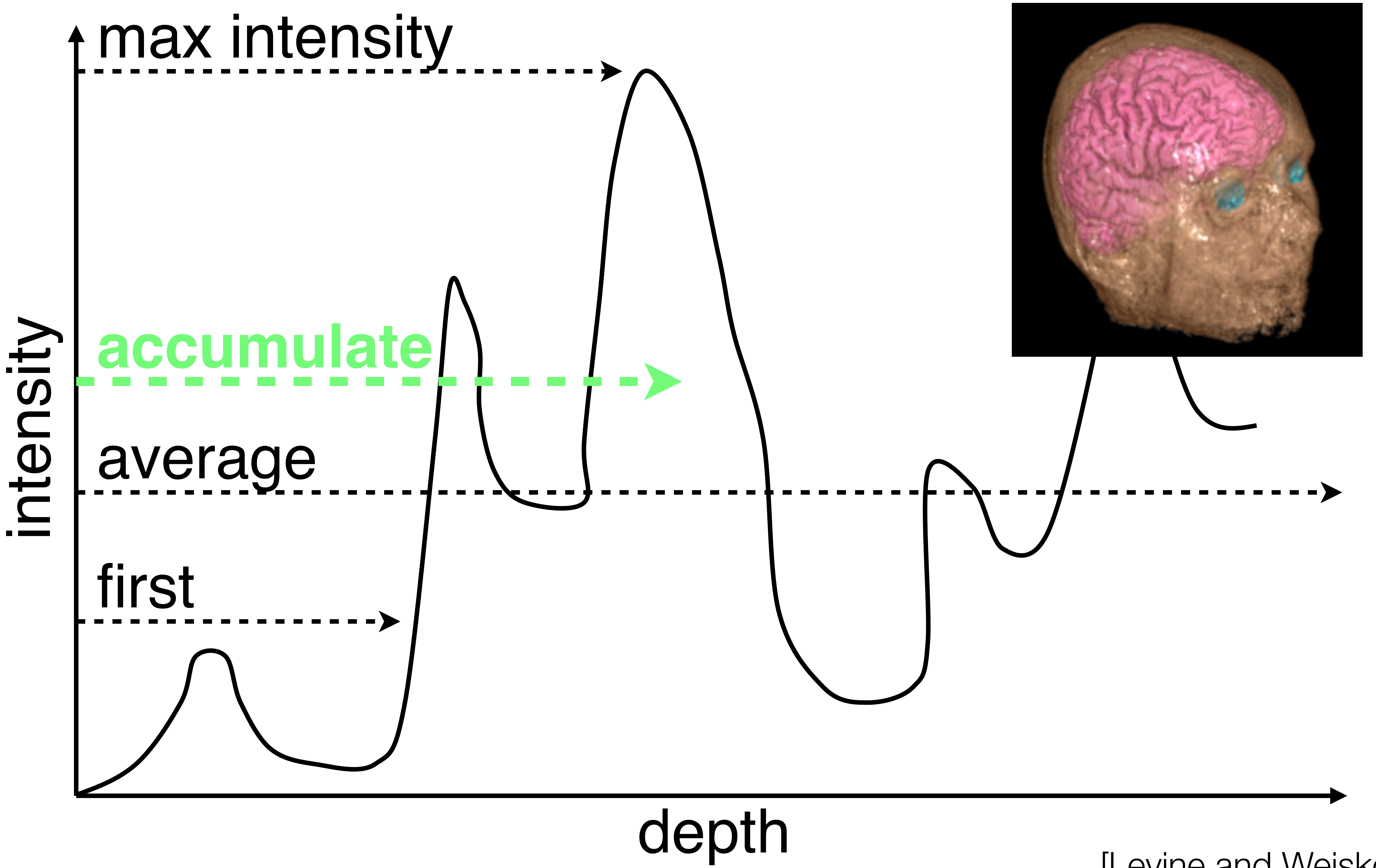
[Levine and Weiskopf/Machiraju/Möller]

Types of Compositing



[Levine and Weiskopf/Machiraju/Möller]

Types of Compositing

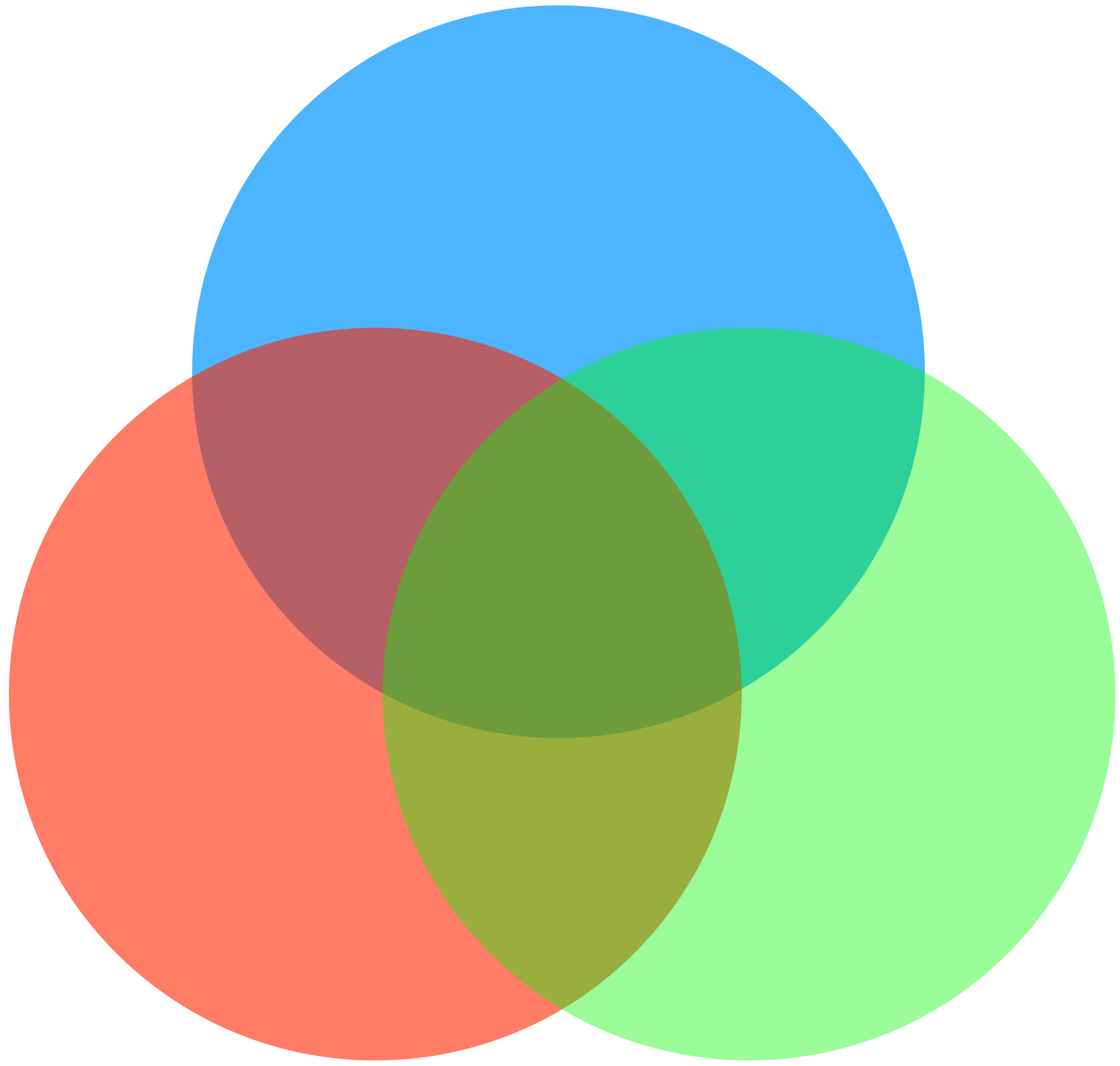


[Levine and Weiskopf/Machiraju/Möller]

Accumulation

- If we're not just calculating a single number (max, average) or a position (first), how do we determine the accumulation?
- Assume each value has an associated color (c) and opacity (α)
- Over operator (back-to-front):
  - $c = \alpha_f \cdot c_f + (1-\alpha_f) \cdot \alpha_b \cdot c_b$
  - $\alpha = \alpha_f + (1-\alpha_f) \cdot \alpha_b$
- Order is important!

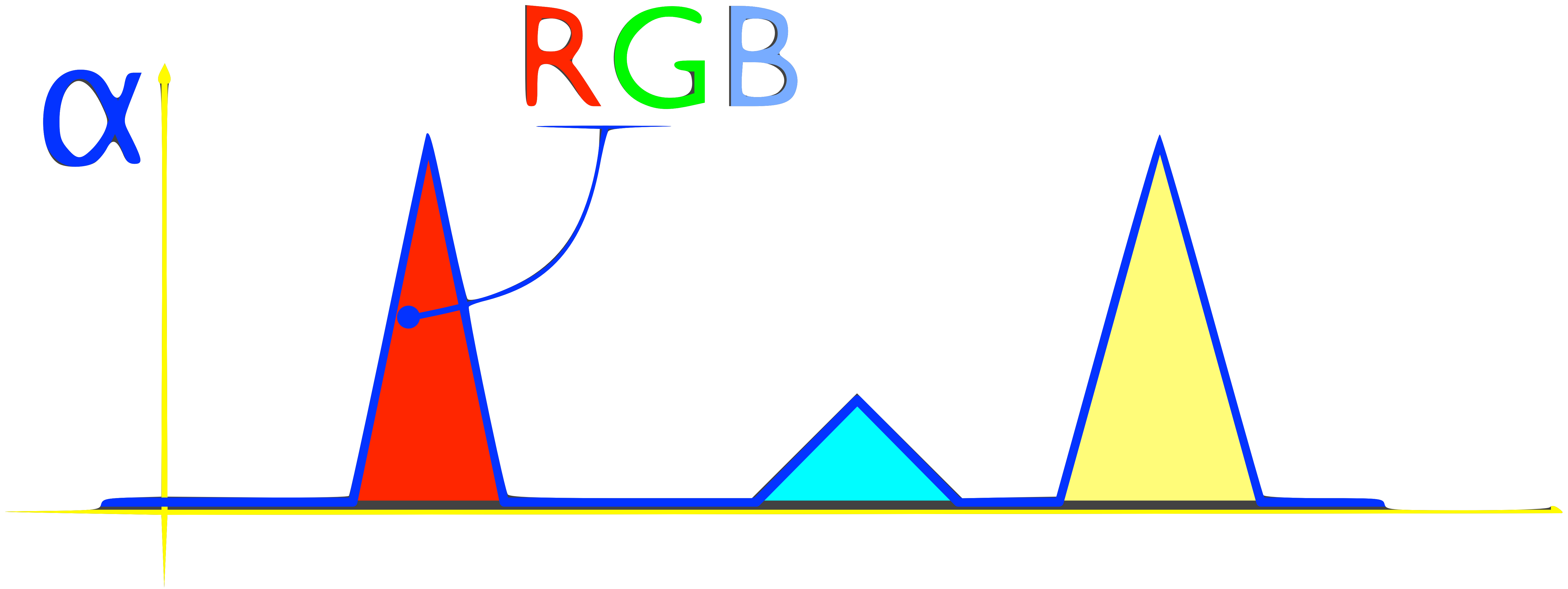
Blue Last



Blue First

Transfer Functions

- Where do the colors and opacities come from?
- Idea is that each voxel emits/absorbs light based on its scalar value
- …but users get to choose how that happens
- x-axis: color region definitions, y-axis: opacity



[Kindlmann]

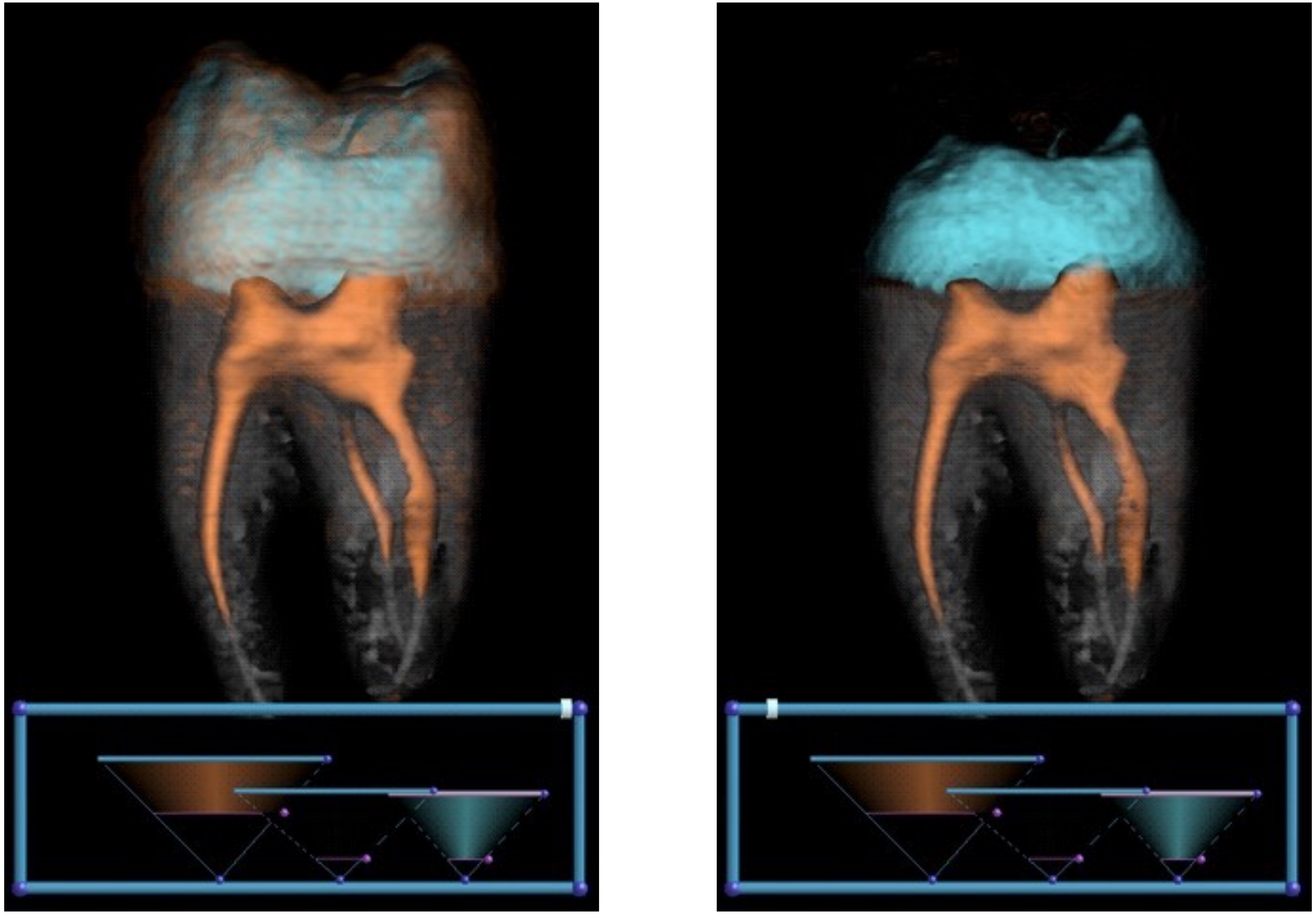
Transfer Function Design

- Transfer function **design** is non-trivial!
- Lots of tools to help visualization designers to create good transfer functions
- Histograms, more attributes than just value like gradient magnitude

Multidimensional Transfer Functions

[J. Kniss]

Multidimensional Transfer Functions



[J. Kniss]

ParaView Examples