

Data Visualization (CSCI 627/490)

Isosurfacing & Volume Rendering

Dr. David Koop

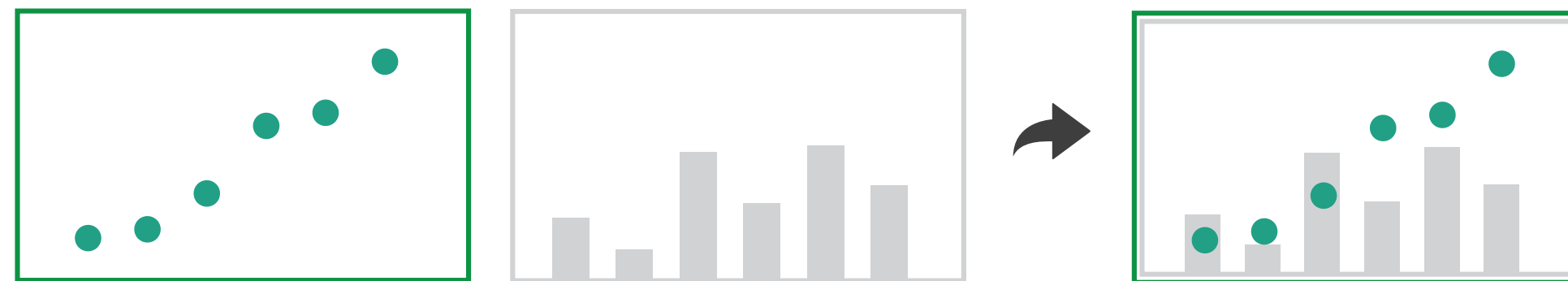
Focus+Context Overview

➔ Embed

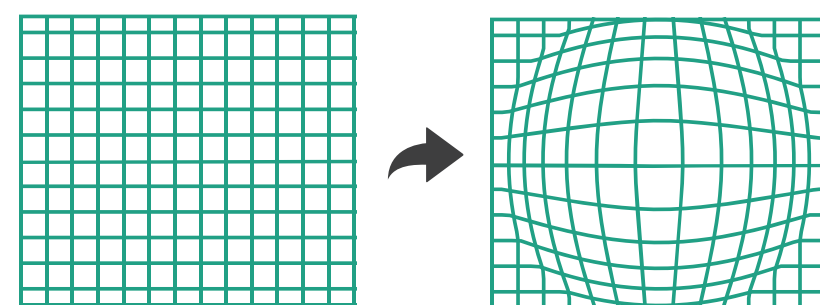
➔ Elide Data



➔ Superimpose Layer



➔ Distort Geometry

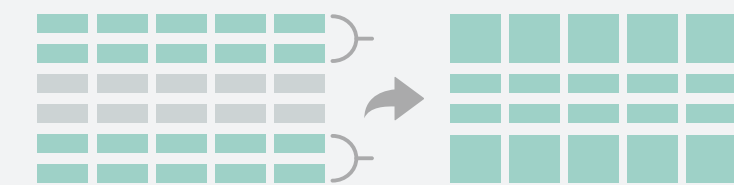


Reduce

➔ Filter



➔ Aggregate

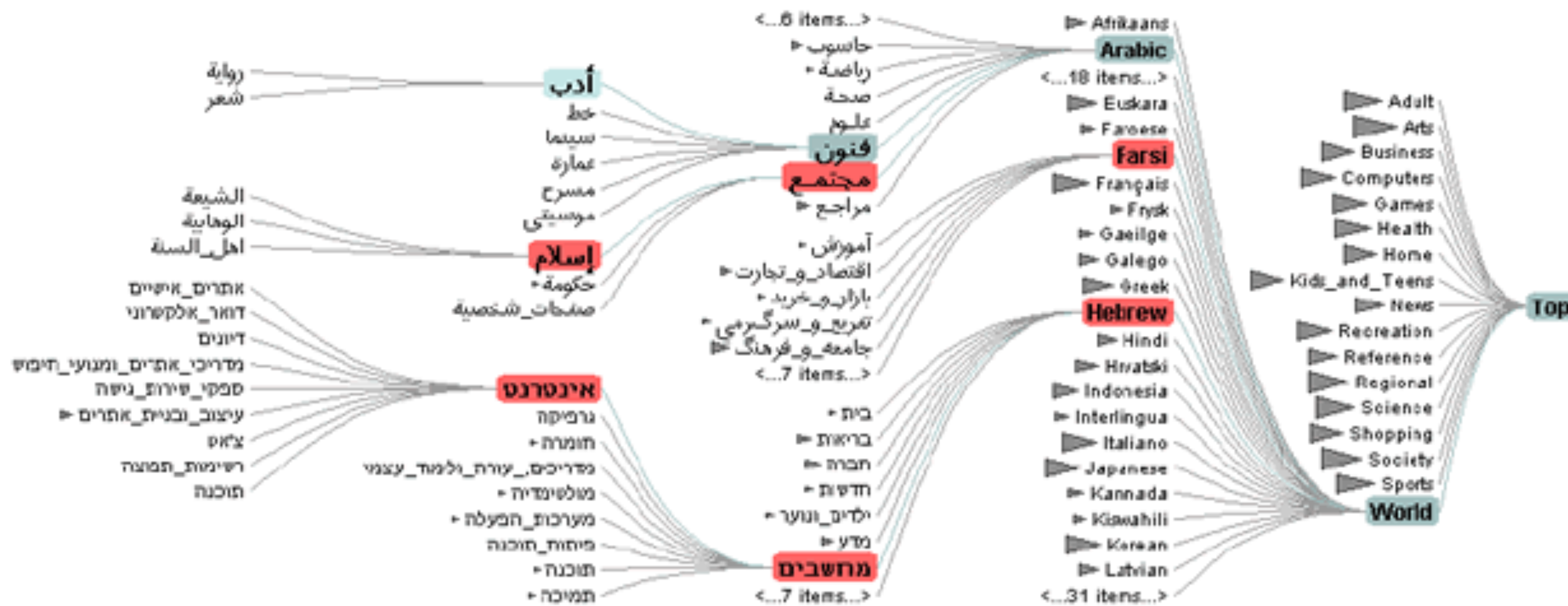


➔ Embed



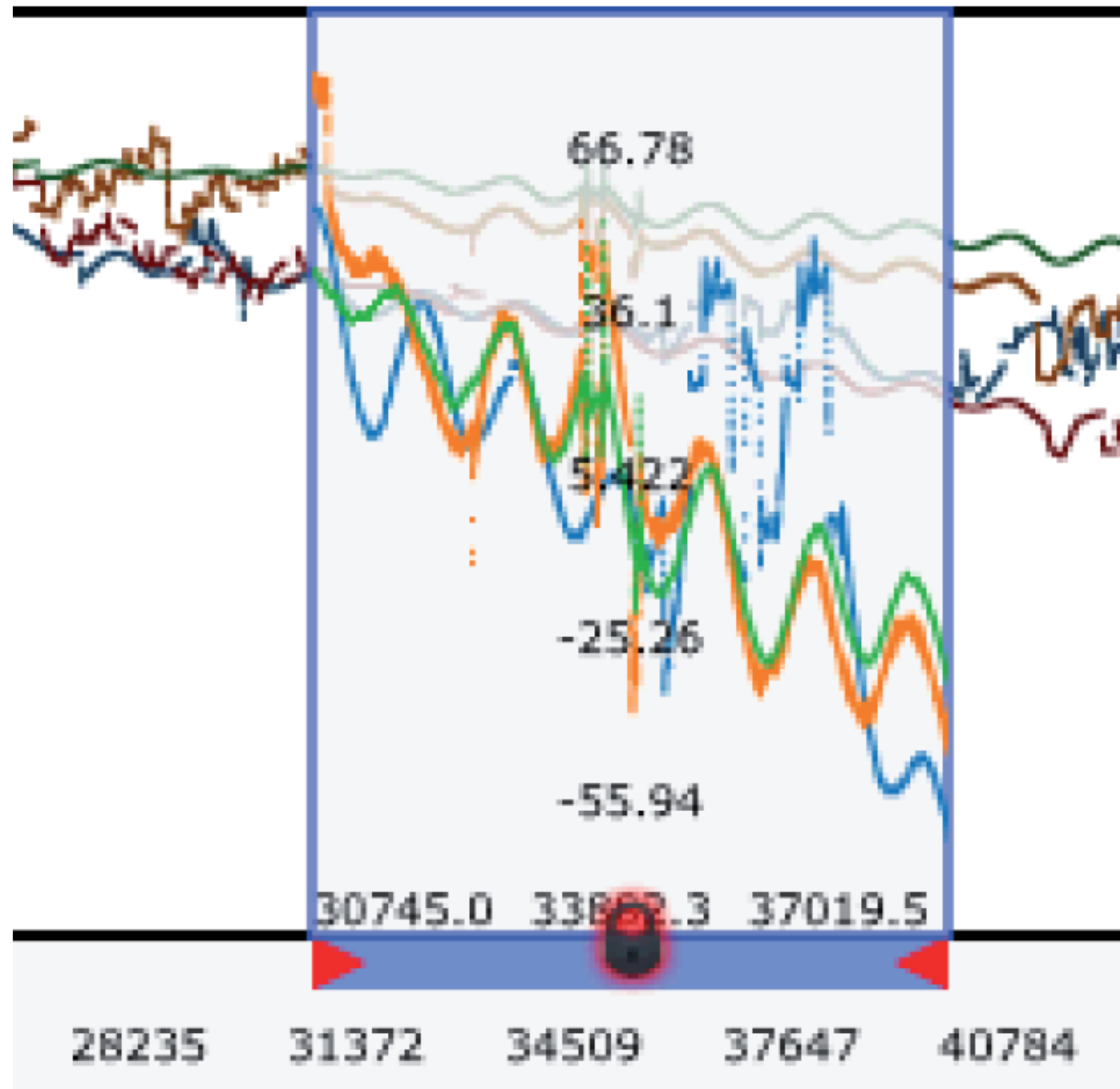
Elision: DOITrees

- Example: 600,000 node tree
 - Multiple foci (from search results or via user selection)
 - Distance computed topologically (levels, not geometric)

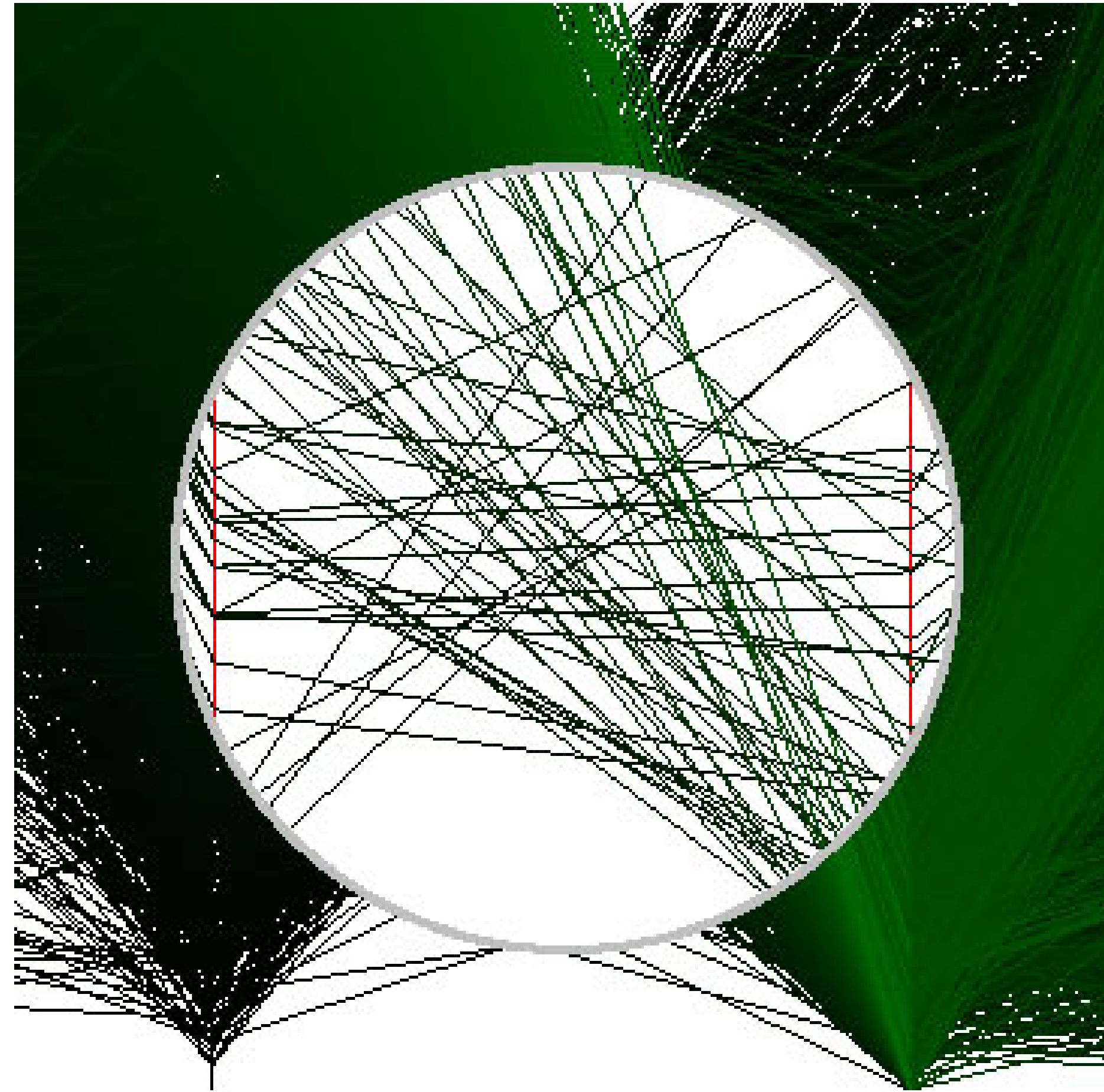


[Heer and Card, 2004]

Superimposition with Interactive Lenses



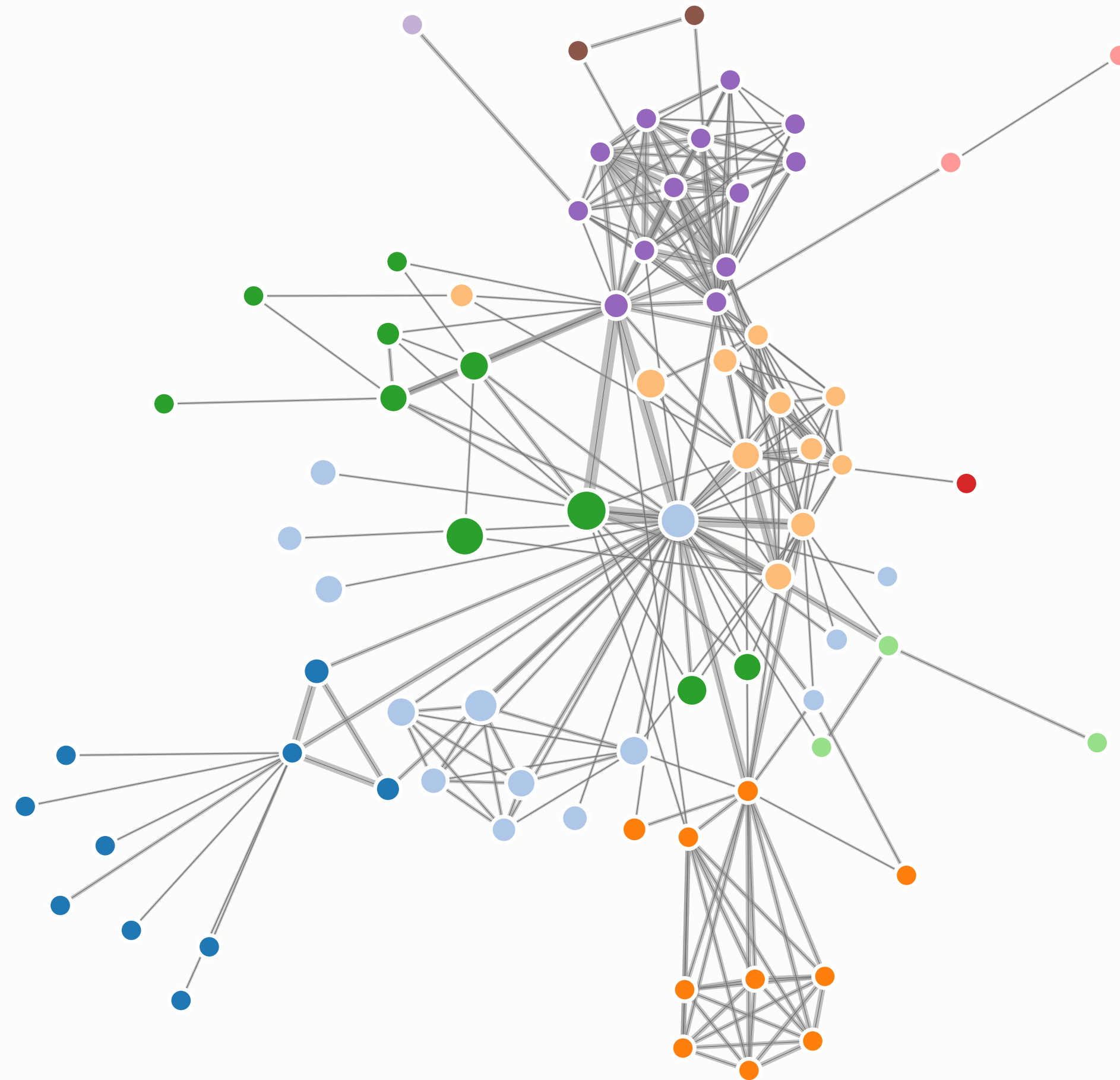
(a) Alteration



(b) Suppression

[ChronoLenses and Sampling Lens in Tominski et al., 2014]

Distortion



[M. Bostock]

Data Wrangling

- Problem 1: Visualizations need data
- Solution: The Web!
- Problem 2: Data has extra information I don't need
- Solution: Filter it
- Problem 3: Data is dirty
- Solution: Clean it up
- Problem 4: Data isn't in the same place
- Solution: Combine data from different sources
- Problem 5: Data isn't structured correctly
- Solution: Reorder, map, and nest it

JavaScript Data Wrangling Resources

- Latest version: <https://observablehq.com/@berkeleyvis/learn-js-data>
- My old version: <https://observablehq.com/@dakoop/learn-js-data>
- Based on <http://learnjsdata.com/>
- Good coverage of data wrangling using JavaScript

Grouping Data

- Take a flat structure and turn it into a (potentially nested) map
- Similar to a groupby in databases
- Data

```
var expenses = [{"name": "jim", "amount": 34, "date": "11/12/2015"},  
  {"name": "carl", "amount": 120.11, "date": "11/12/2015"},  
  {"name": "jim", "amount": 45, "date": "12/01/2015"},  
  {"name": "stacy", "amount": 12.00, "date": "01/04/2016"},  
  {"name": "stacy", "amount": 34.10, "date": "01/04/2016"},  
  {"name": "stacy", "amount": 44.80, "date": "01/05/2016"}  
];
```

- Grouping:

```
expensesByName = d3.group(expenses, d => d.name)
```

- Results:

```
Map(3) { "jim" => Array(2) [Object, Object]  
  "carl" => Array(1) [Object]  
  "stacy" => Array(3) [Object, Object, Object] }
```


Rollup Data

- Data

```
var expenses = [{ "name": "jim", "amount": 34, "date": "11/12/2015" },  
  { "name": "carl", "amount": 120.11, "date": "11/12/2015" },  
  { "name": "jim", "amount": 45, "date": "12/01/2015" },  
  { "name": "stacy", "amount": 12.00, "date": "01/04/2016" },  
  { "name": "stacy", "amount": 34.10, "date": "01/04/2016" },  
  { "name": "stacy", "amount": 44.80, "date": "01/05/2016" }  
];
```

- Using d3.rollup:

```
expensesAvgAmount = d3.rollup(  
  expenses,  
  v => d3.mean(v, d => d.amount), // aggregate by the mean of amount  
  d => d.name // group by name  
)
```

← the aggregation function
(difference from group)

- Result:

```
Map(3) {  
  "jim" => 39.5  
  "carl" => 120.11  
  "stacy" => 30.3  
}
```

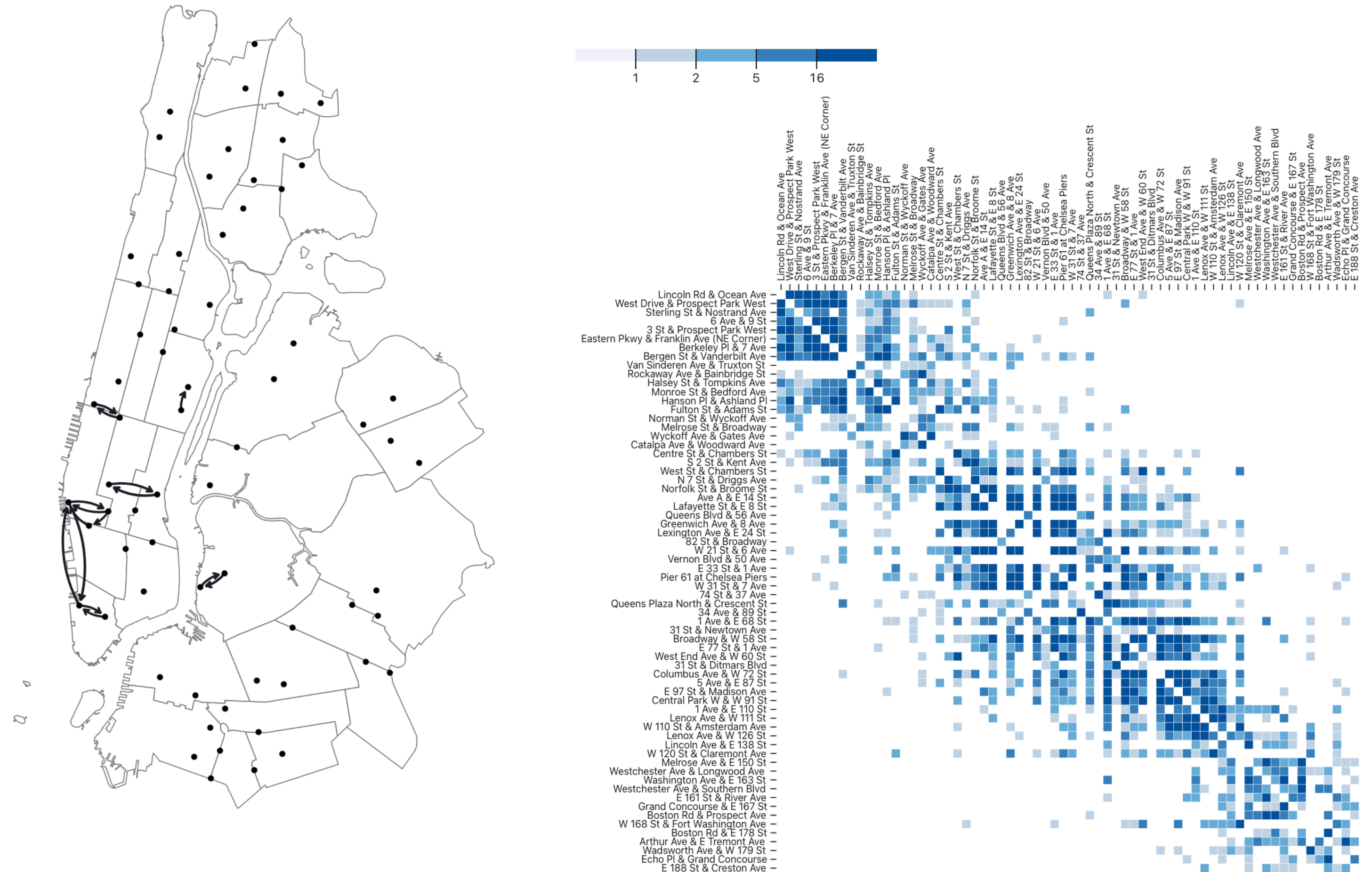
[<http://learnjsdata.com>]

arquero

- Library for query processing and transformation of array-backed data tables
- Similar to database and/or dataframe frameworks
- Integrates with Apache Arrow
- Documentation
- Illustrated Guide to Arquero's Verbs

Assignment 5

- Create Multiple Views
- Filtering
- Linked Highlighting
- Aggregation



Final Project

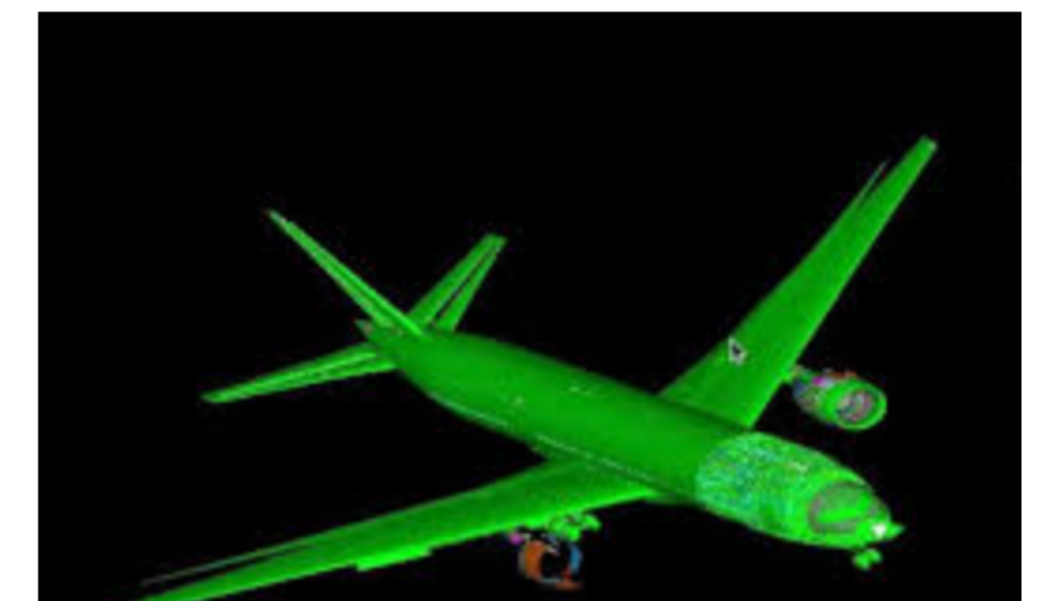
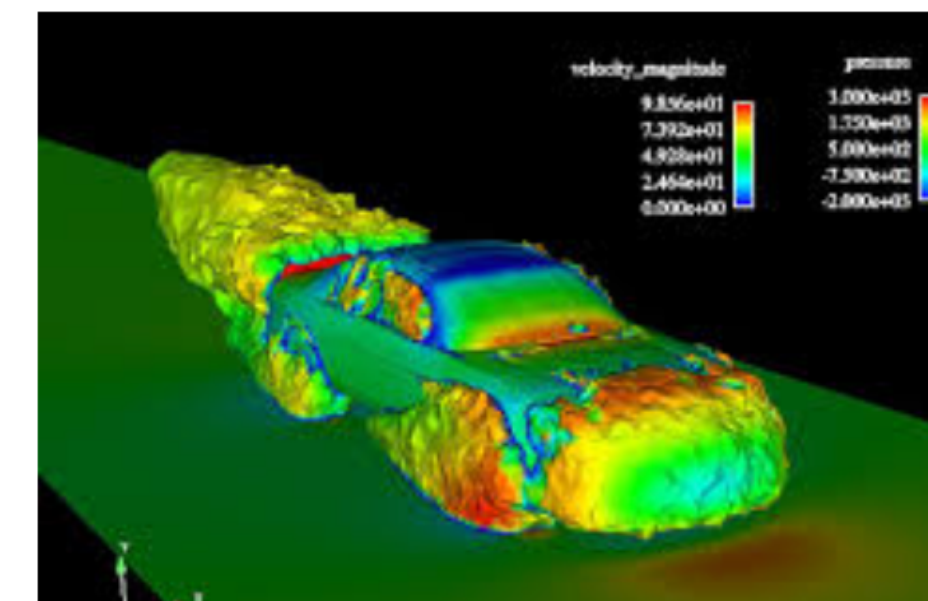
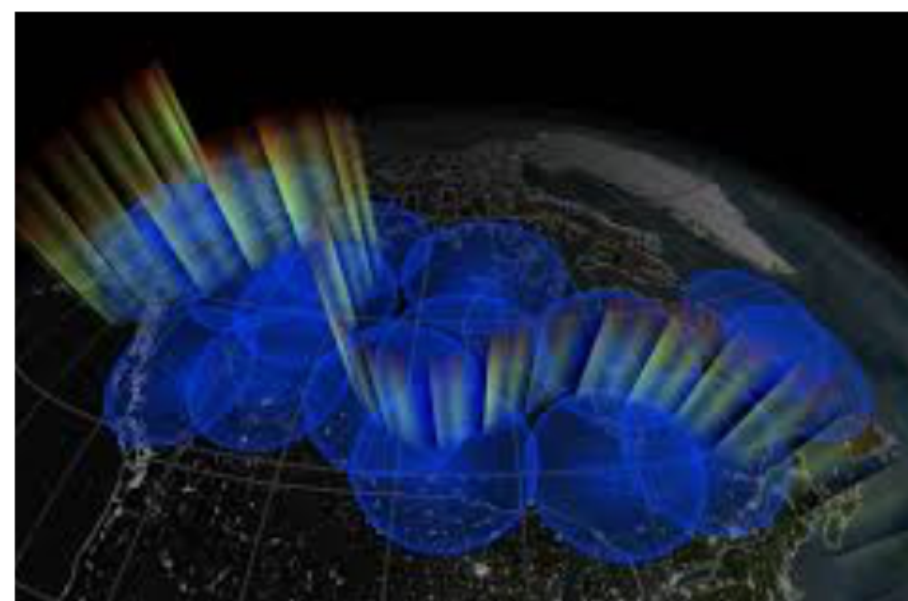
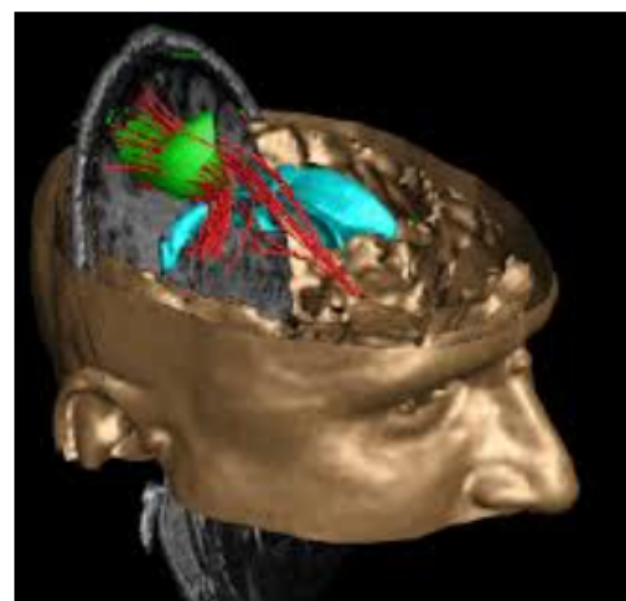
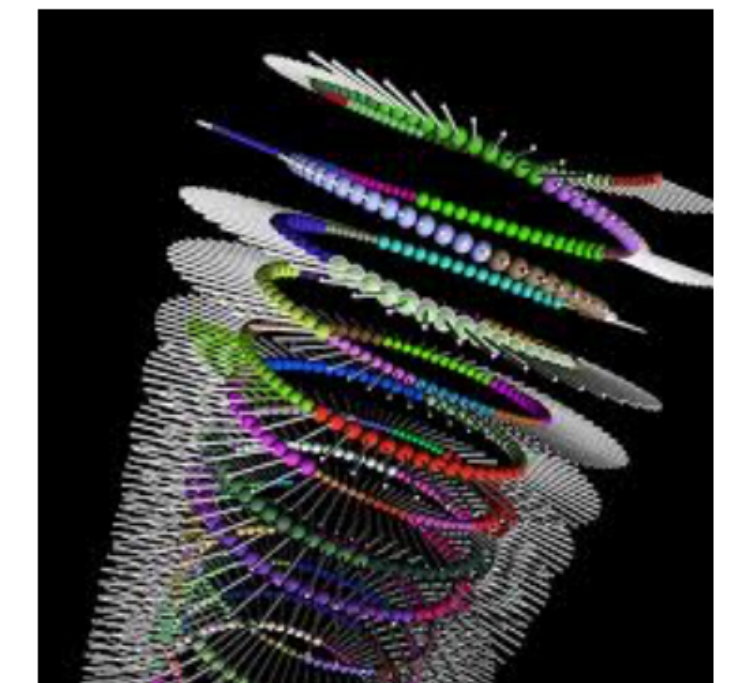
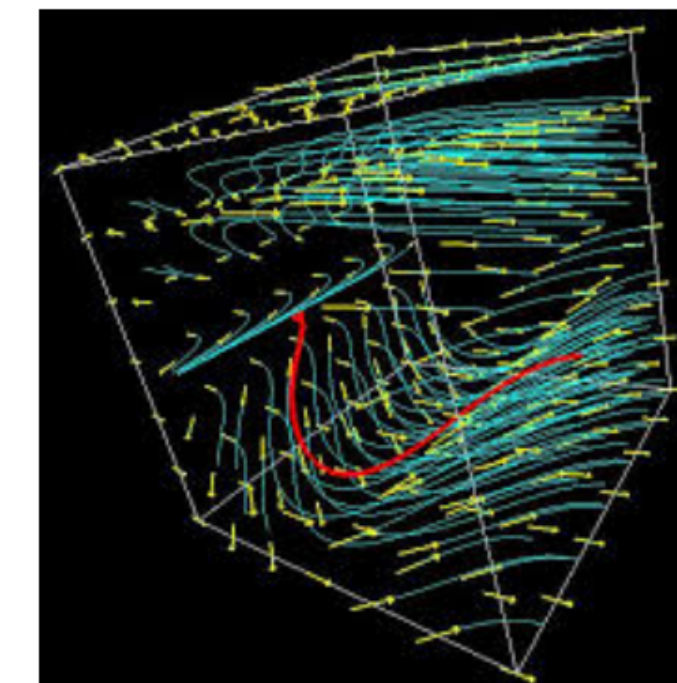
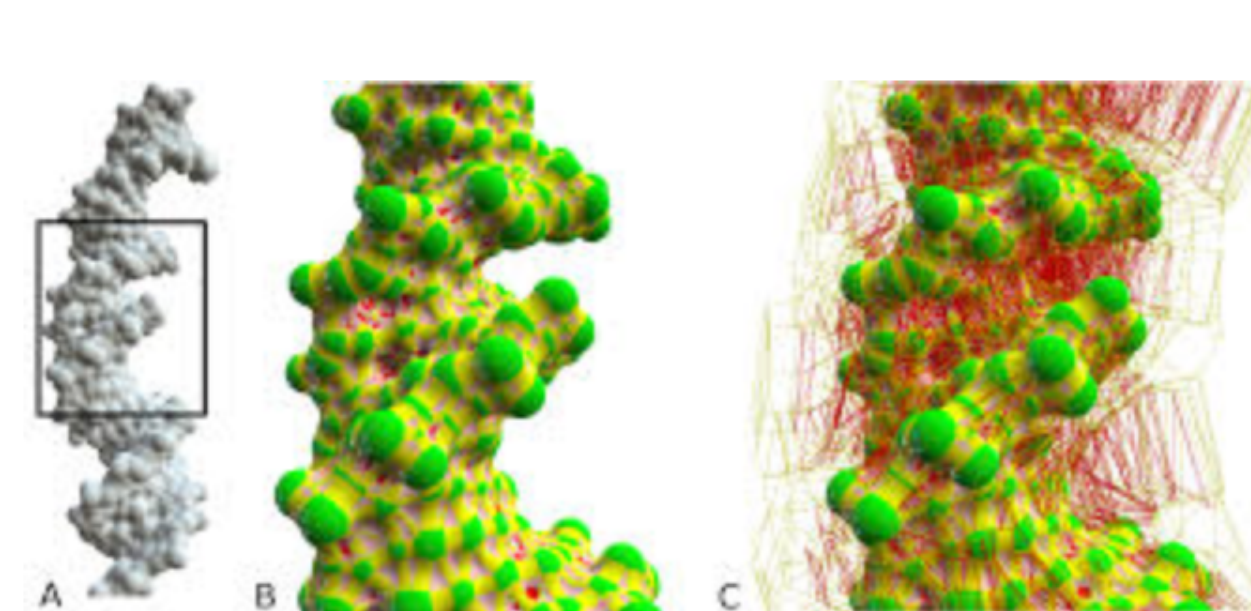
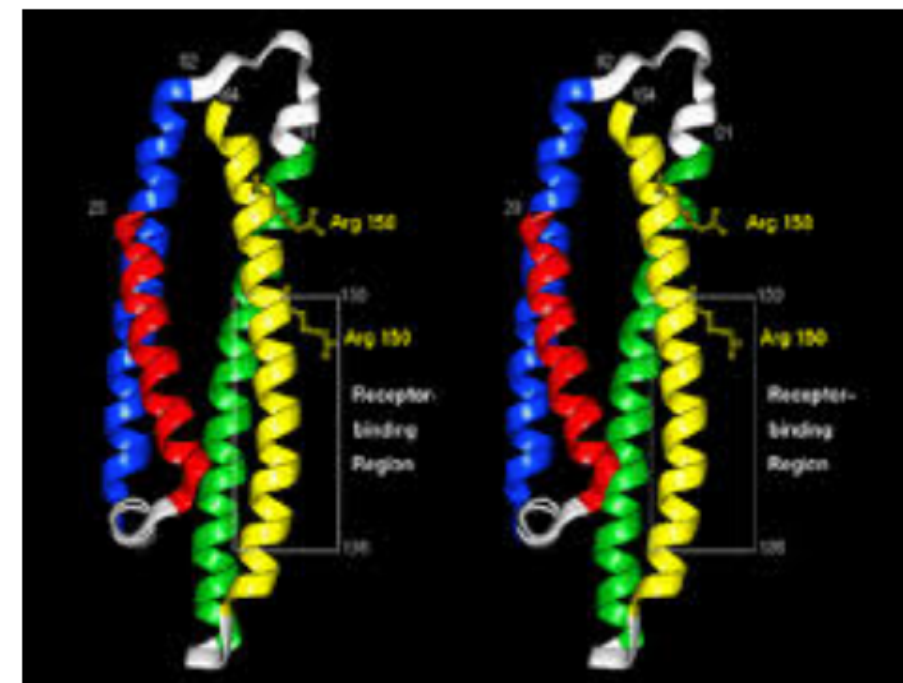
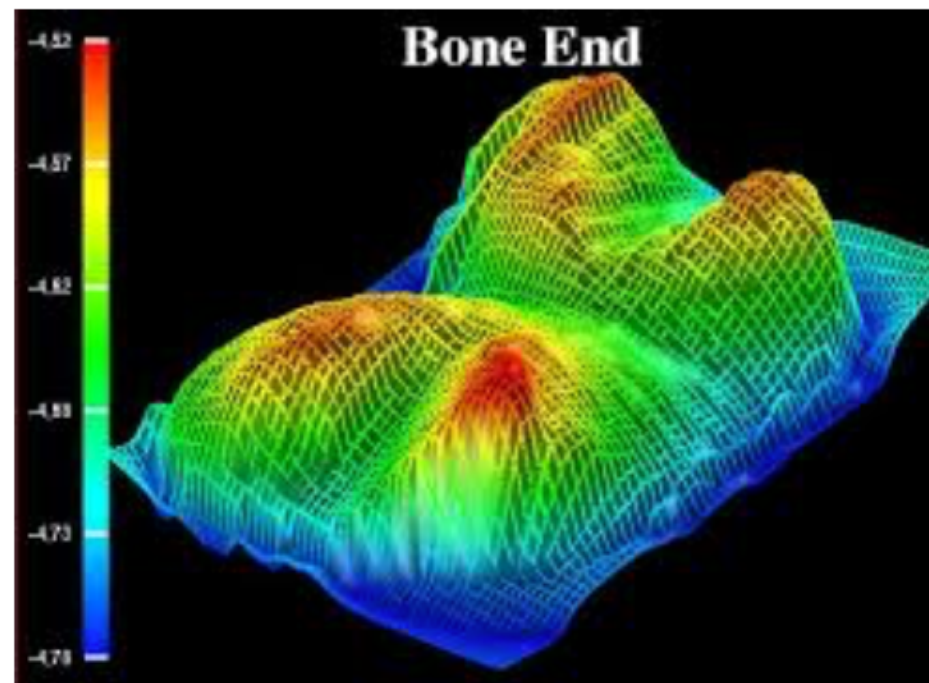
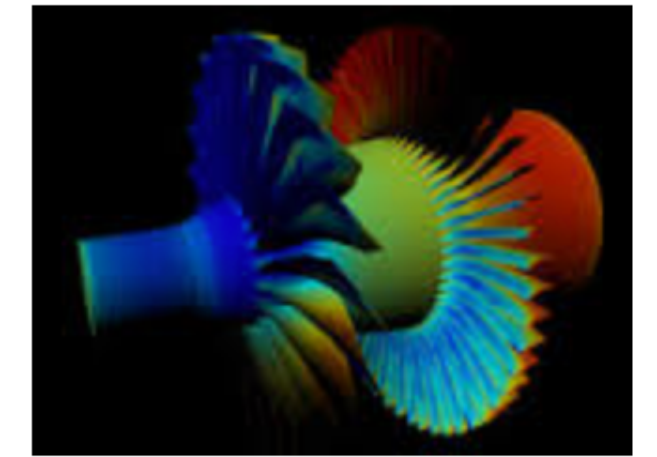
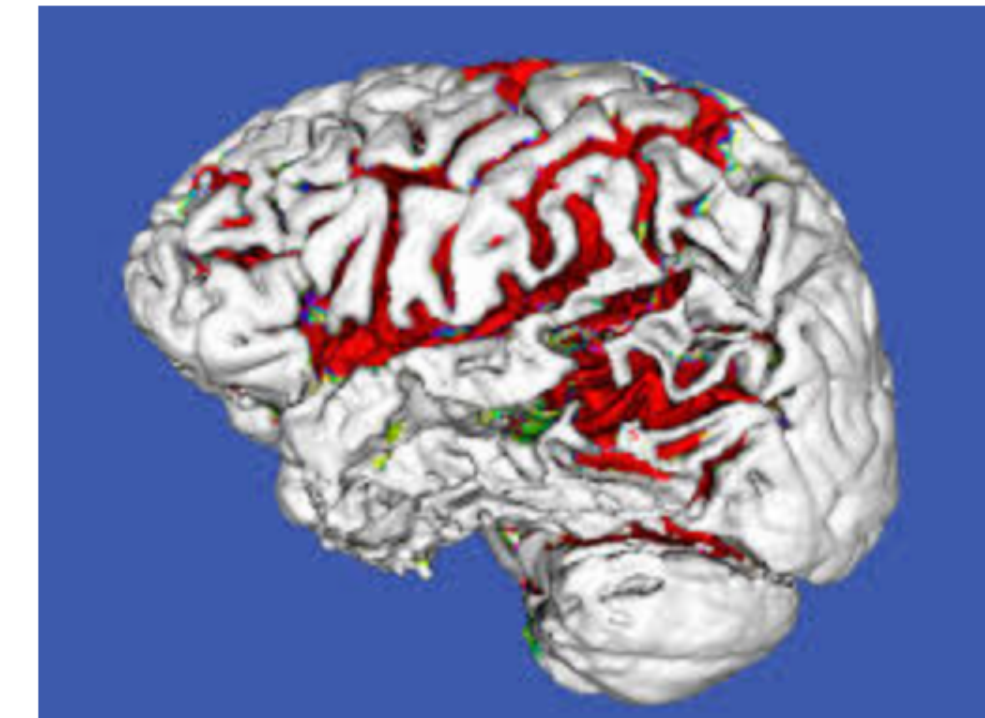
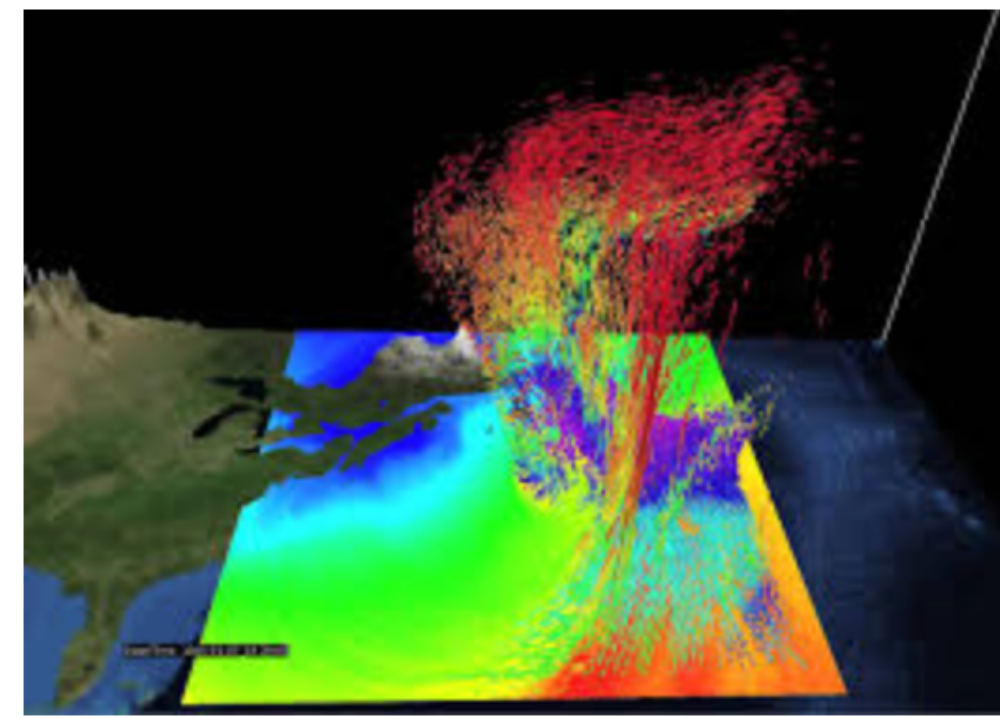
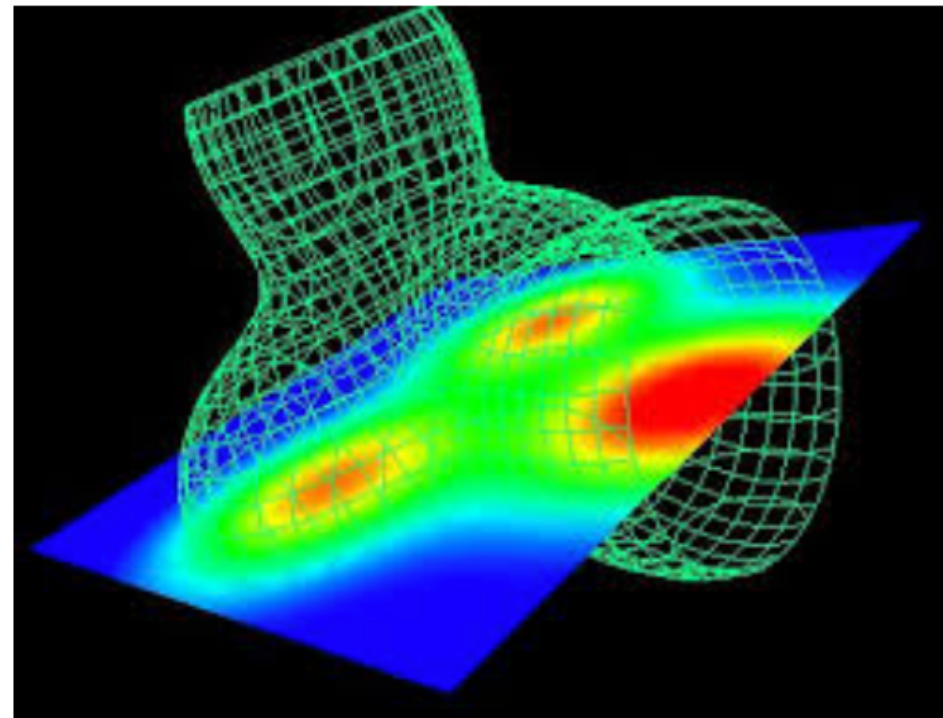
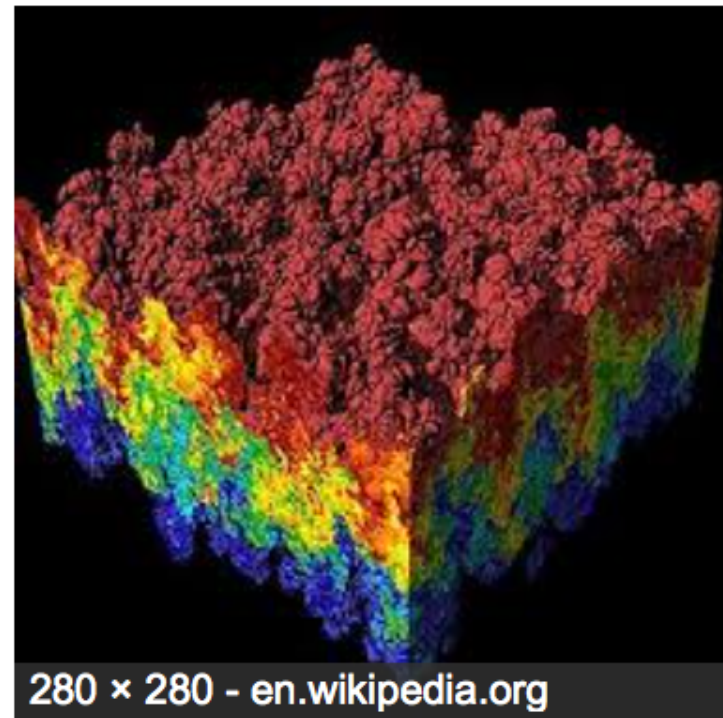
- Designs feedback soon
- Work on implementations
- Presentations will be last week of class
- Reports due at the end of the class

Scientific Visualization

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

SciVis



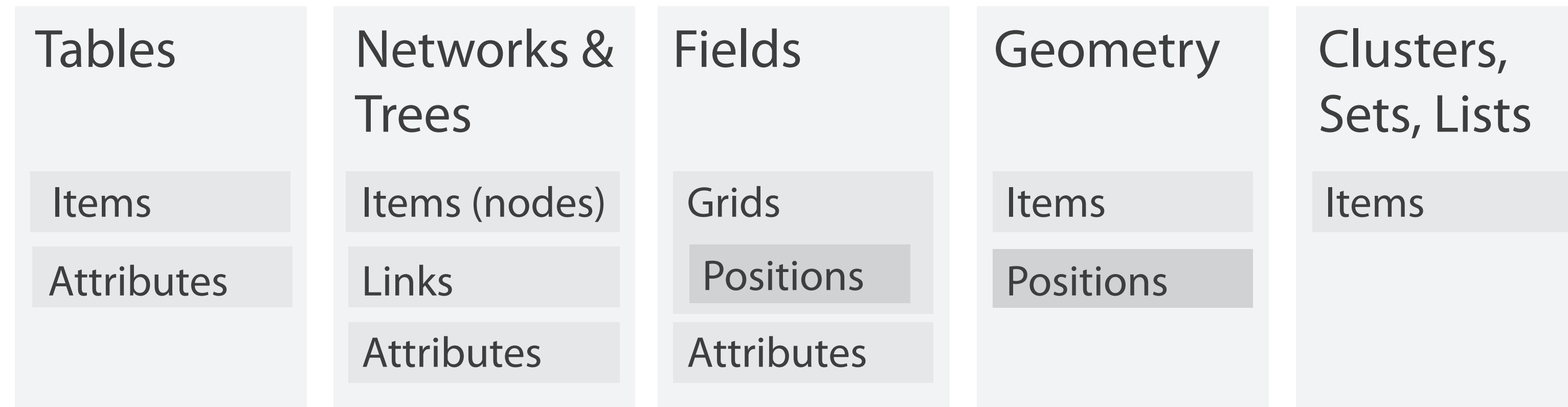
[Google Image Search for "scientific visualization", 2017]

InfoVis



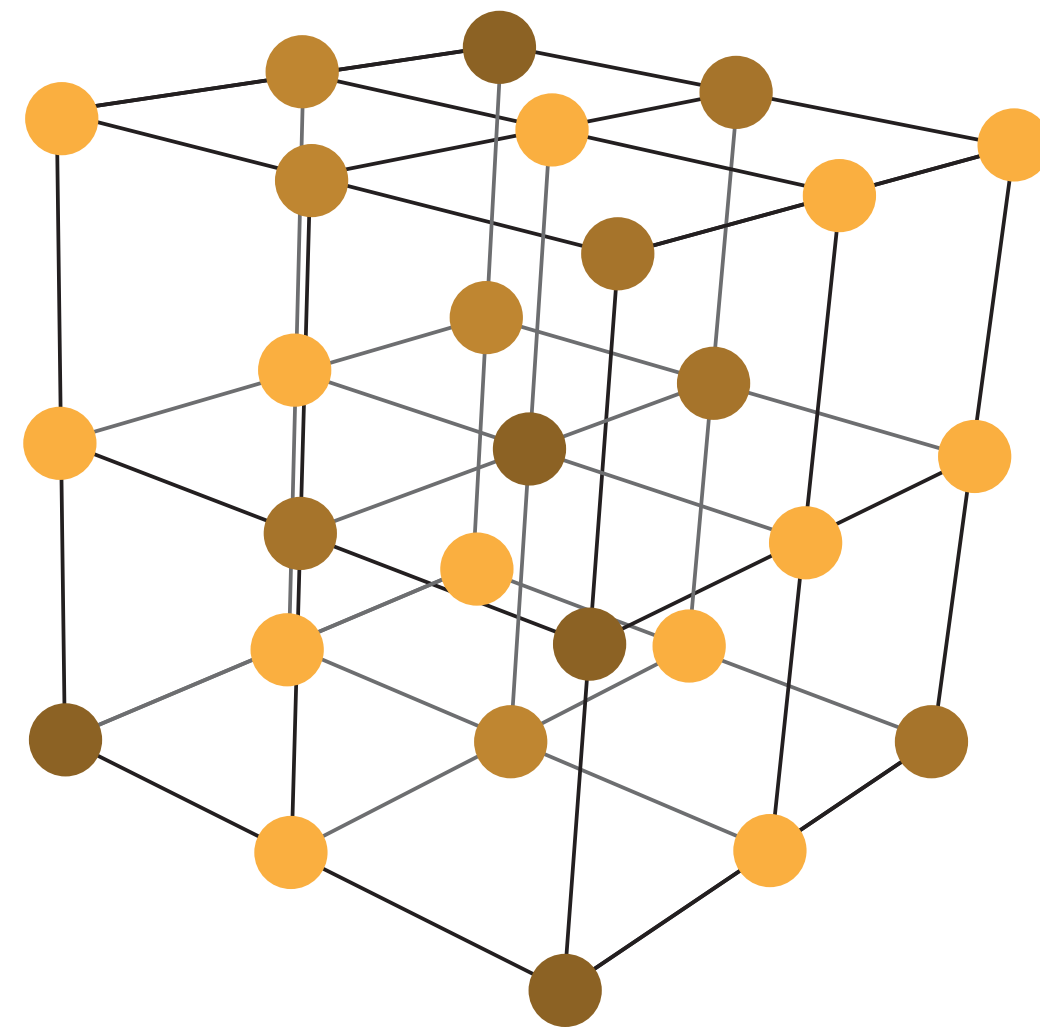
[Google Image Search for "information visualization", 2017]

Fields



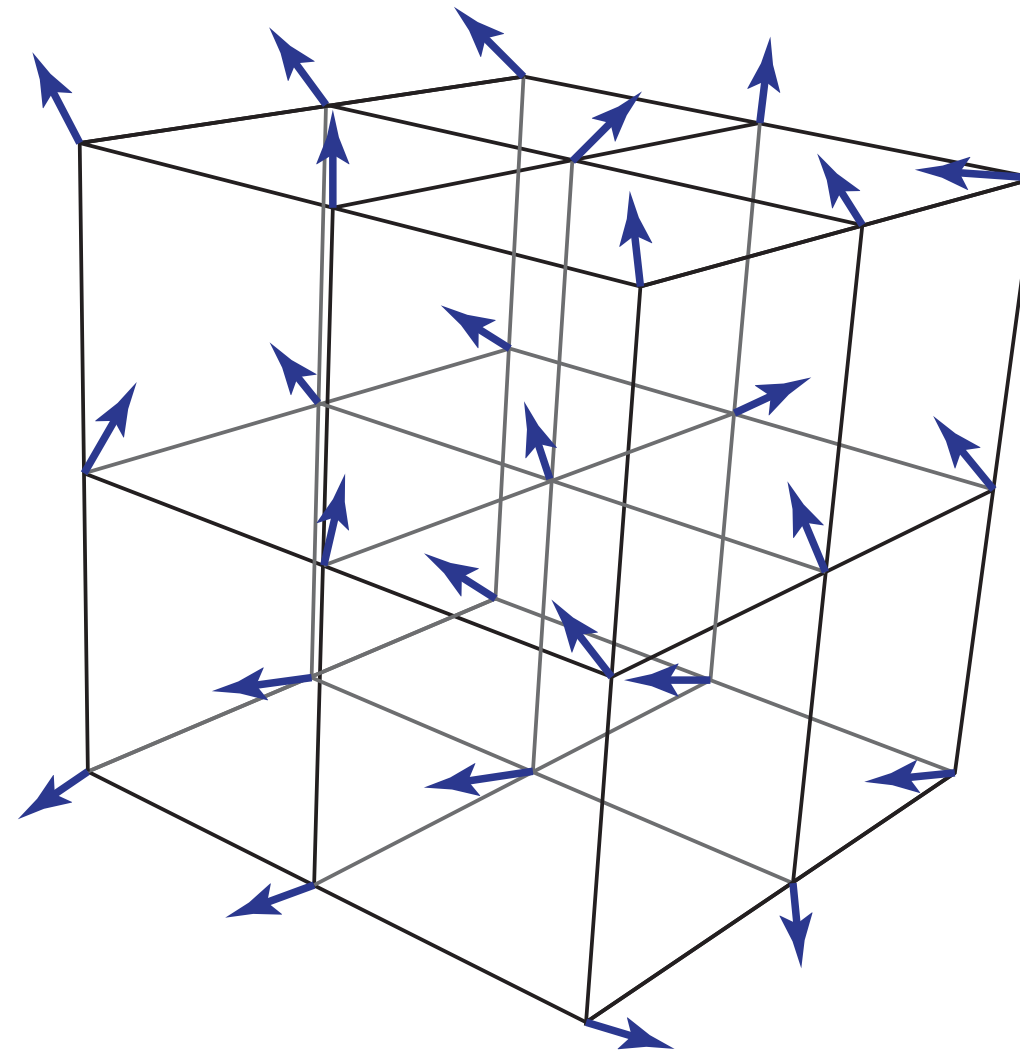
- Values come from a **continuous** domain, infinitely many values
- **Sampled** at certain positions to approximate the entire domain
- Positions are often aligned in **grids**
- Often measurements of natural or simulated phenomena
- Examples: temperature, wind speed, tissue density, pressure, speed, electrical conductance

Fields in Visualization



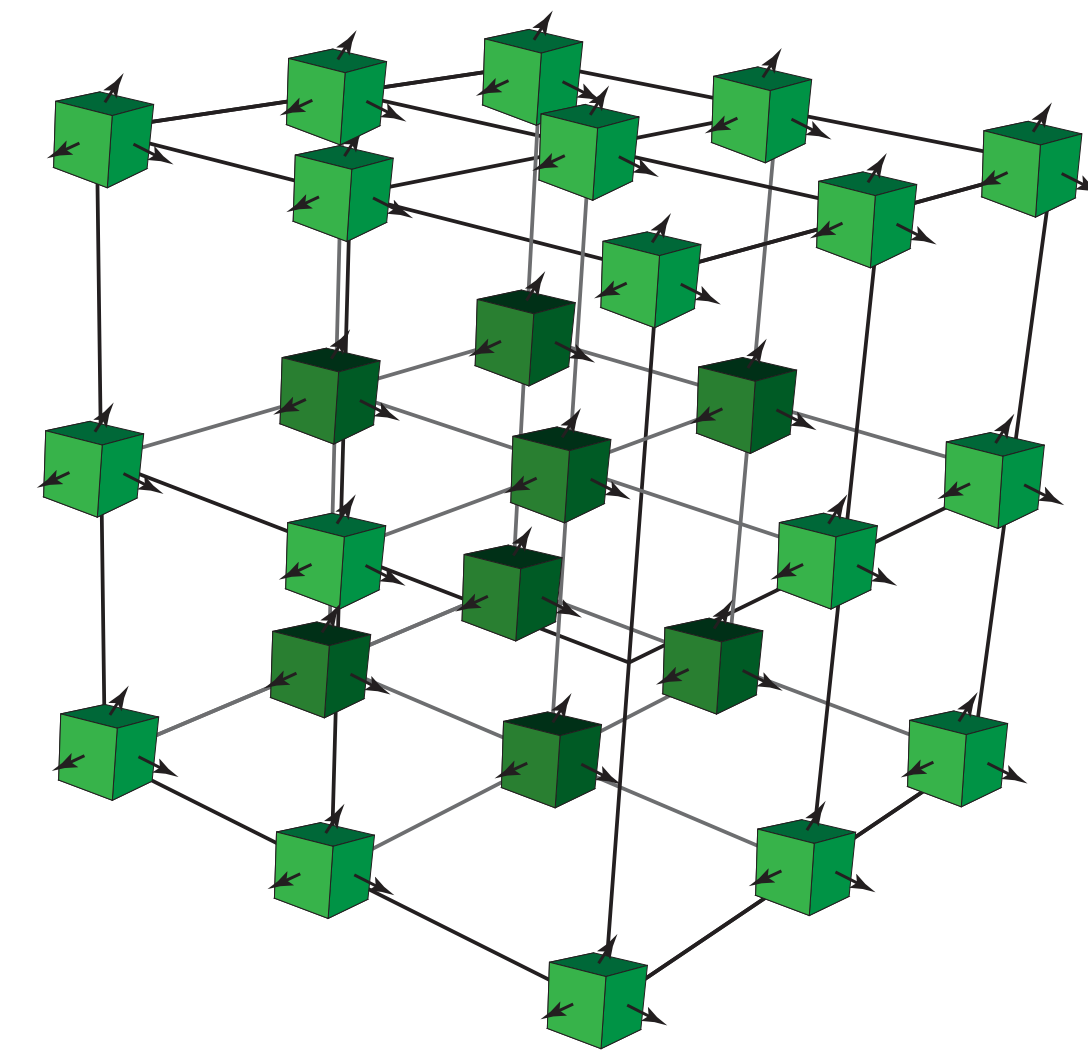
Scalar Fields

(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

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

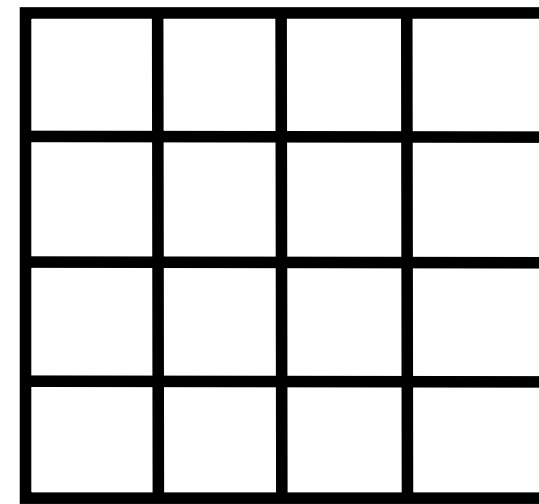
Tensor

Grids

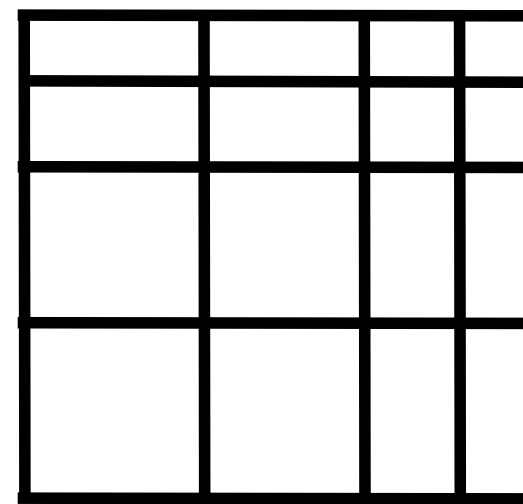
- Remember we have continuous data and want to sample it in order to understand the **entire** domain
- Possible schemes?
- Geometry: the spatial positions of the data (points)

Grids

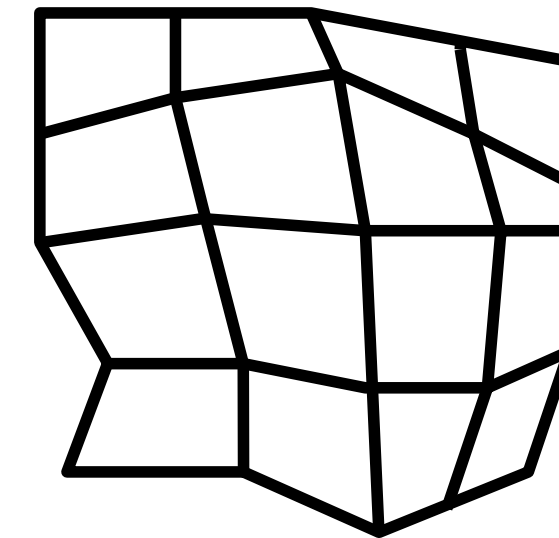
- Remember we have continuous data and want to sample it in order to understand the **entire** domain
- Possible schemes?



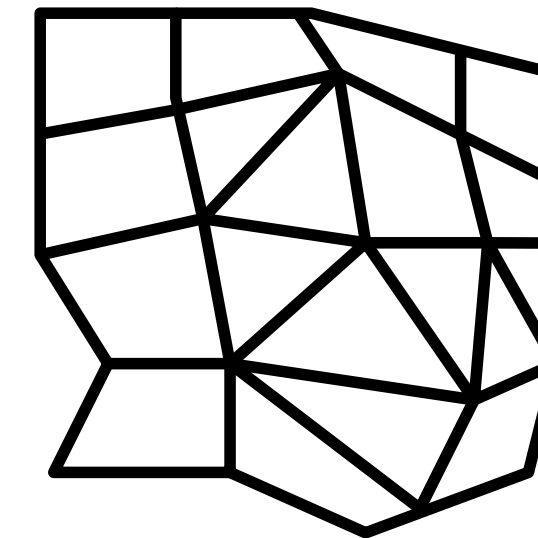
uniform



rectilinear



structured

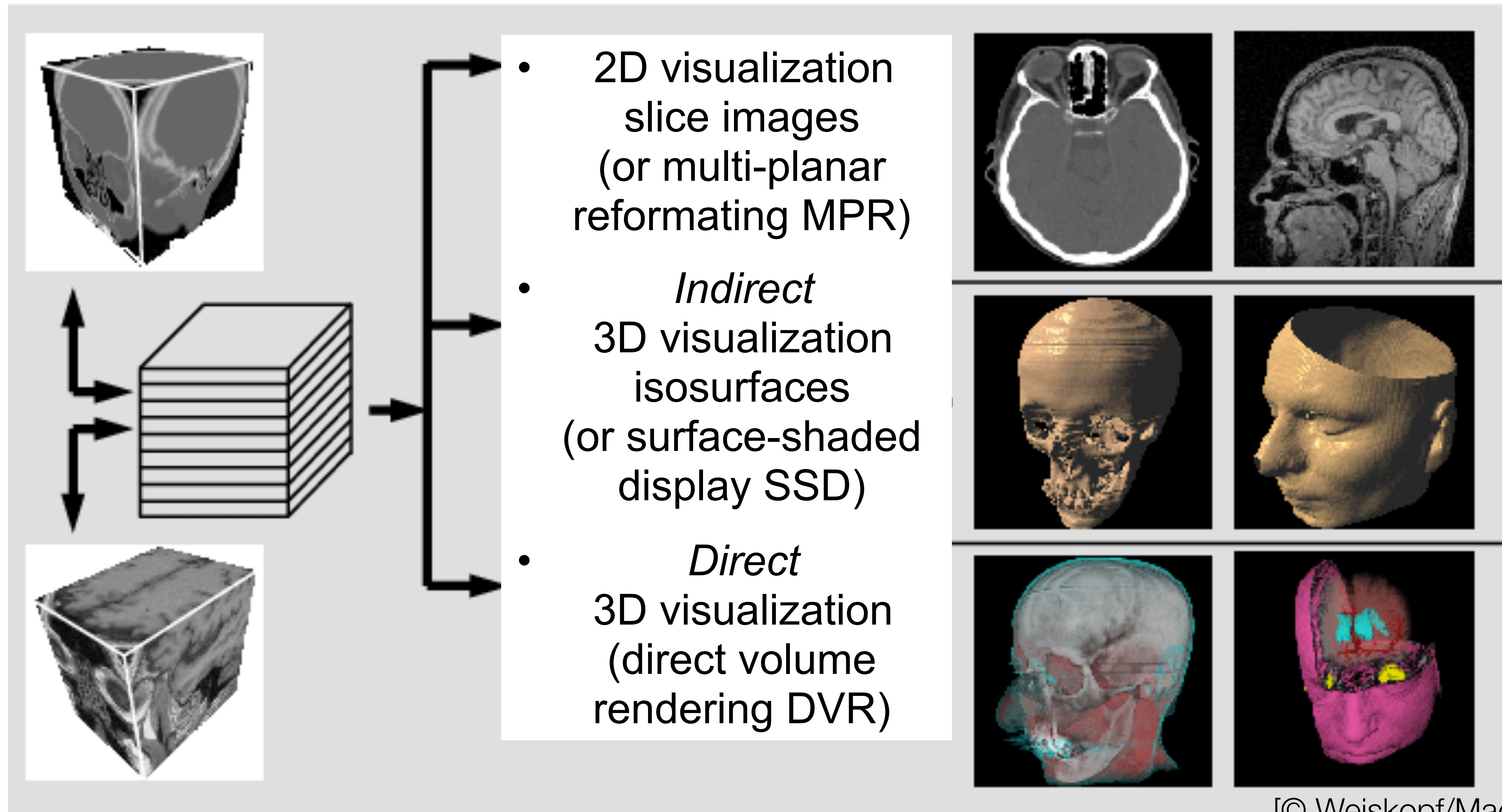


unstructured

[© 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

Visualizing Volume (3D) Data

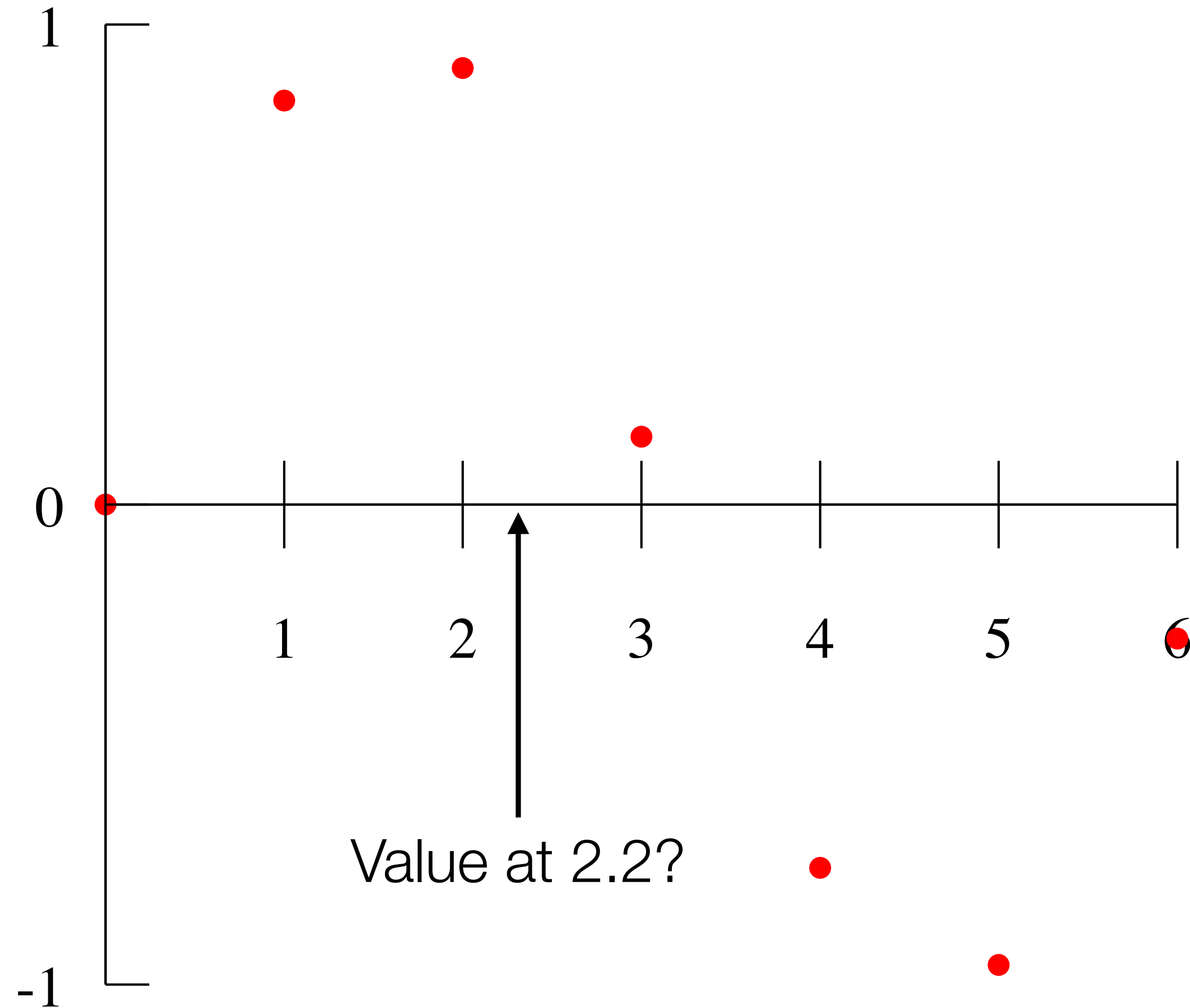


[© Weiskopf/Machiraju/Möller]

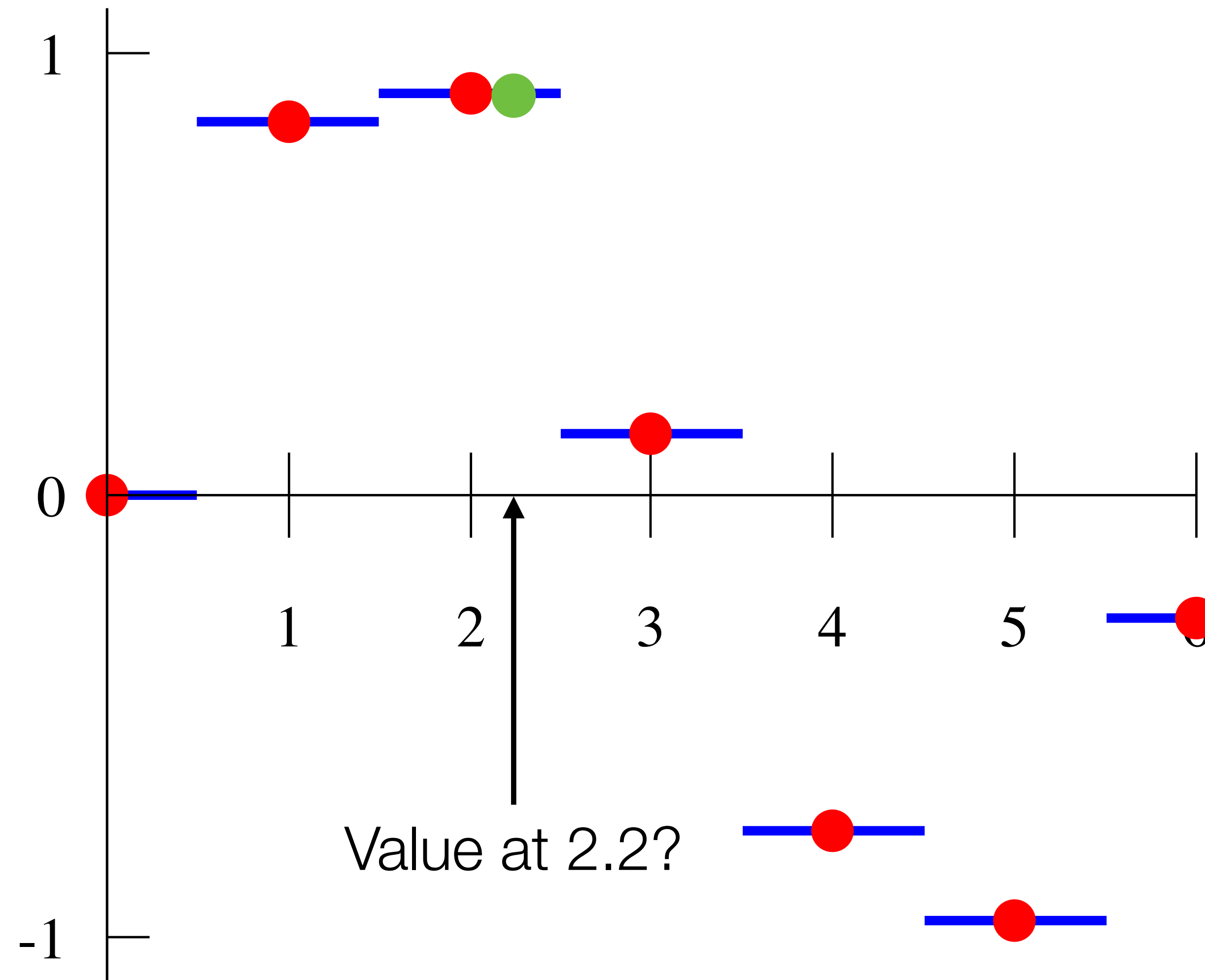
Data

- In this lecture, we will be considering **scalar** data: a single value at each point
- Our data is always discrete, what is the value of a point not exactly on our grid?
- Need a method to determine what these values are...

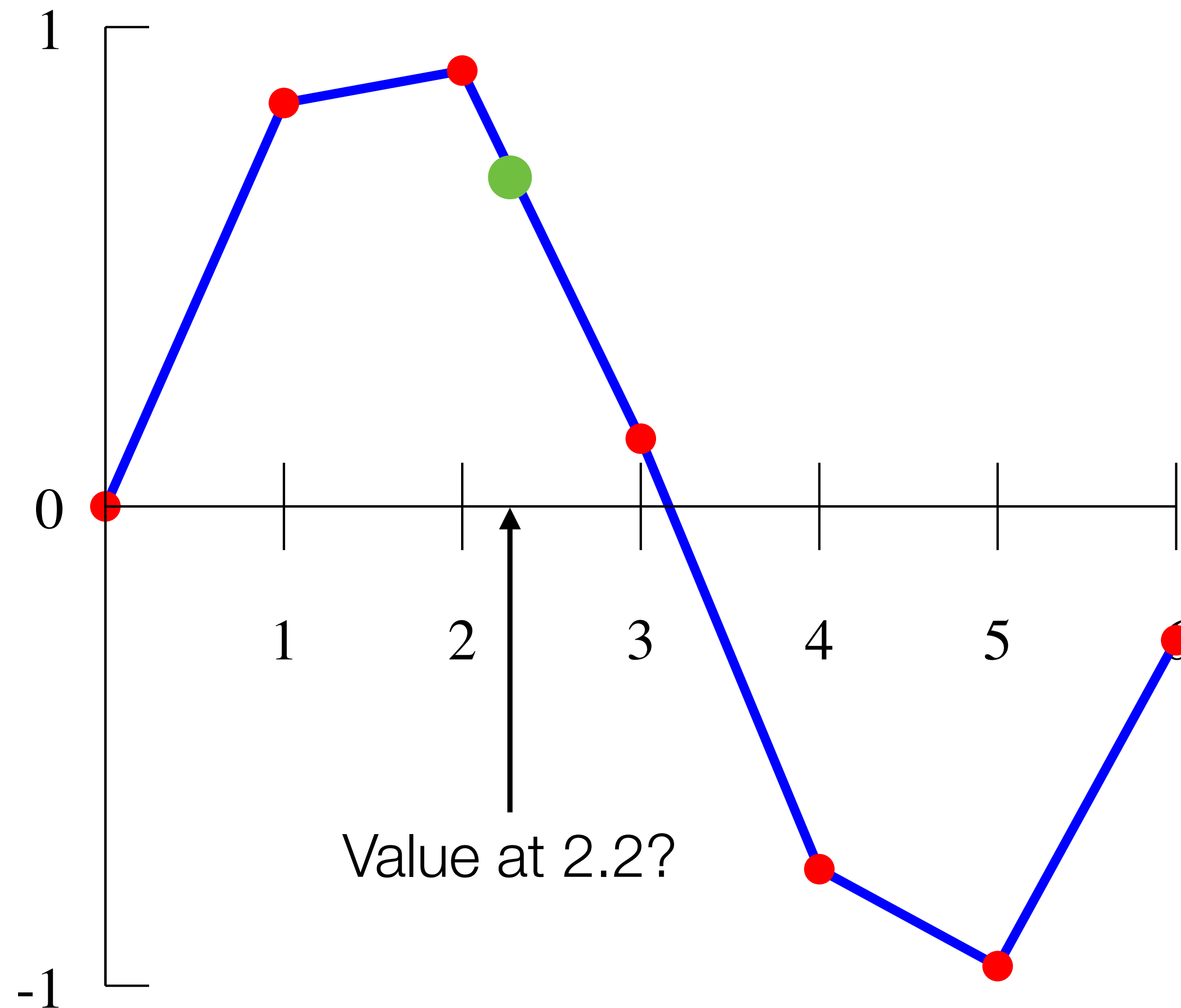
Interpolation



Nearest Neighbor Interpolation



Linear Interpolation



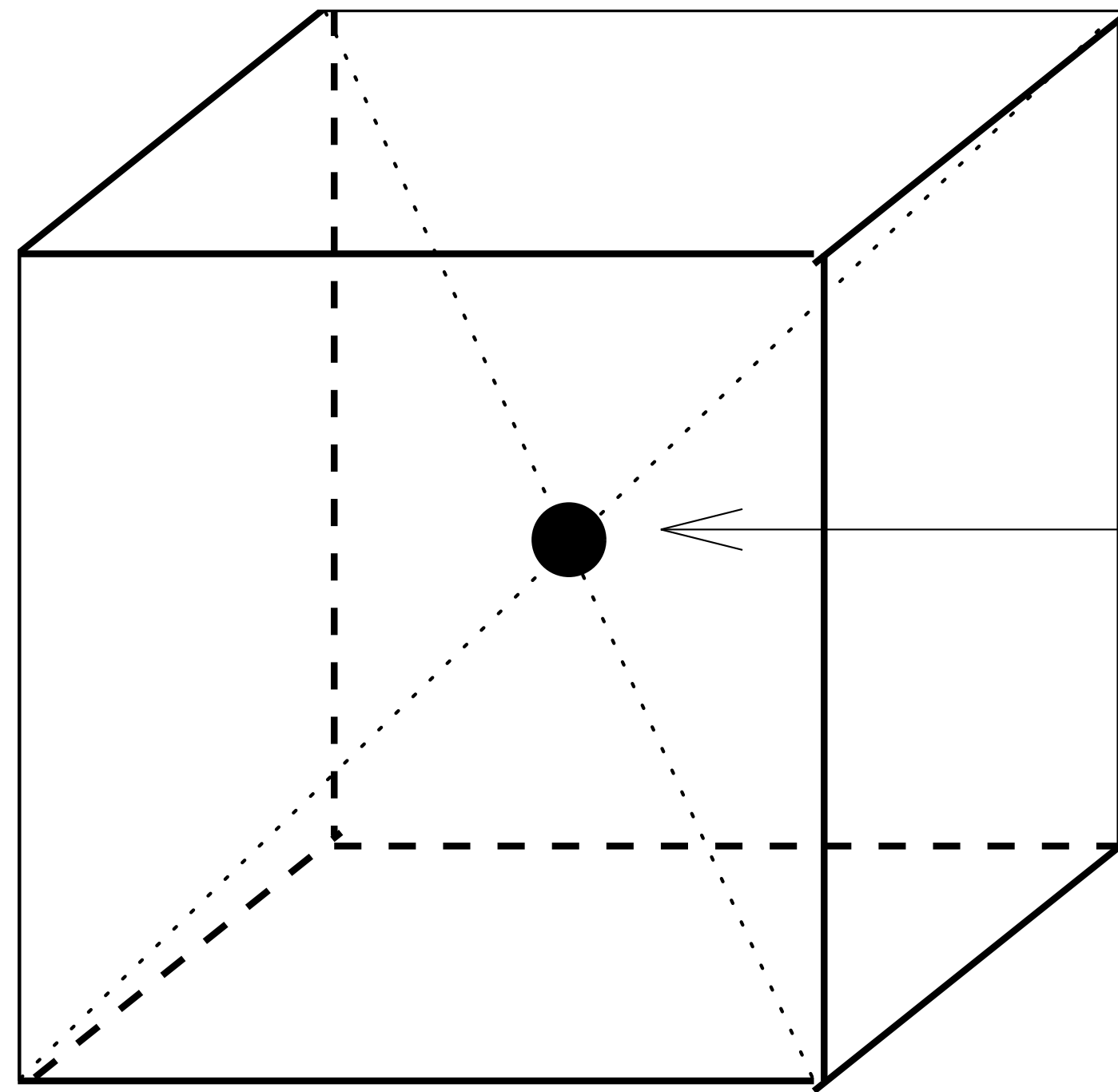
Interpolation

- Other schemes:
 - polynomial interpolation
 - splines
 - more...

Dimensions of Data

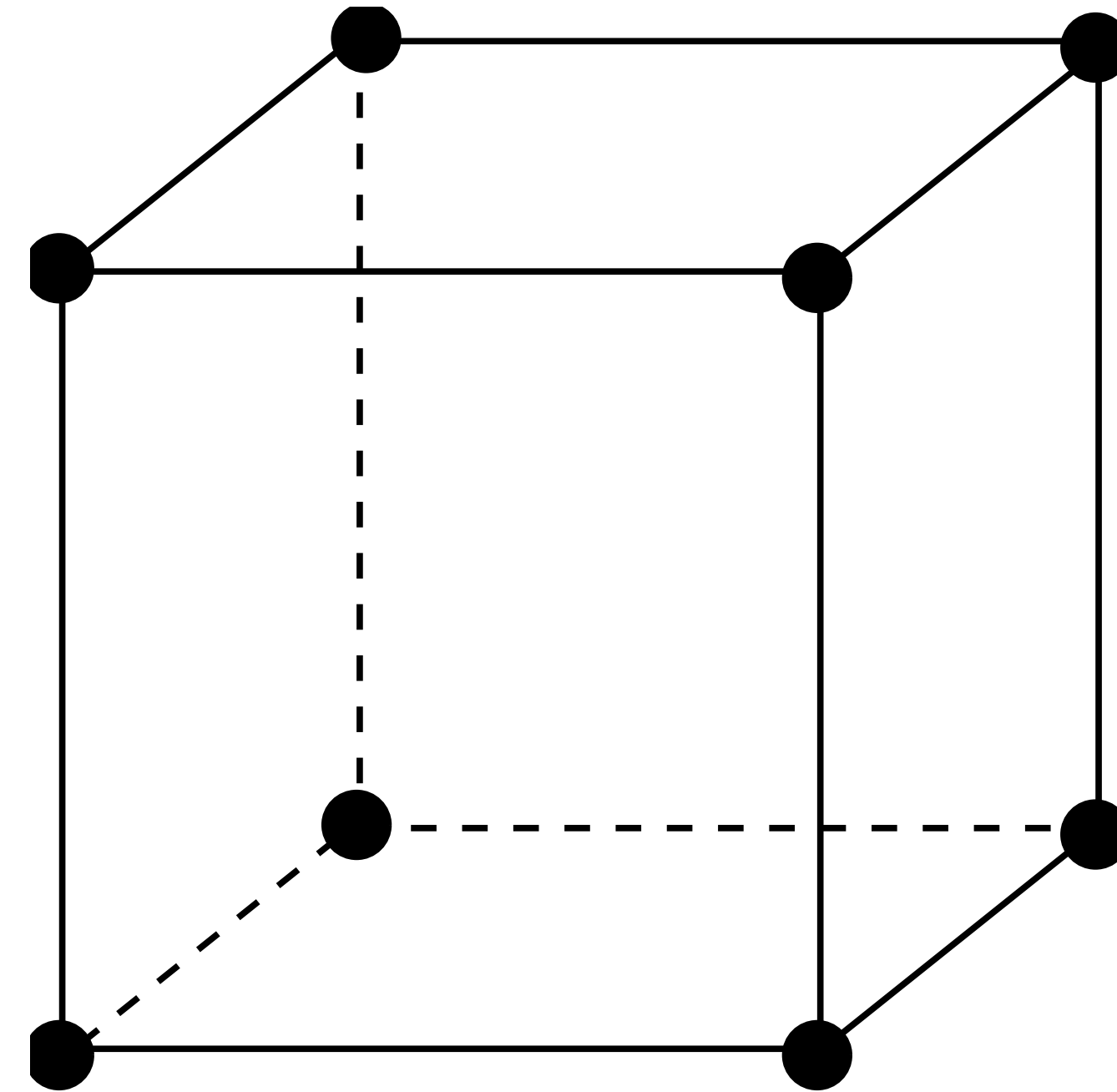
- 1-Dimension: data along a line
 - Example: temperature along my drive from Massachusetts to Illinois
- 2-Dimensional: data on a plane
 - Example: temperature on the surface of a pond
- 3-Dimensional: data in our normal world (data in a **volume**)
 - Example: temperature at every point in the room
- Complexity increases as we add dimensions
- Visualization complexity also increases
- Often, want to be able to see phenomena as we see them in real life settings

3D: Voxels and Cells



VOXEL

gridpoint

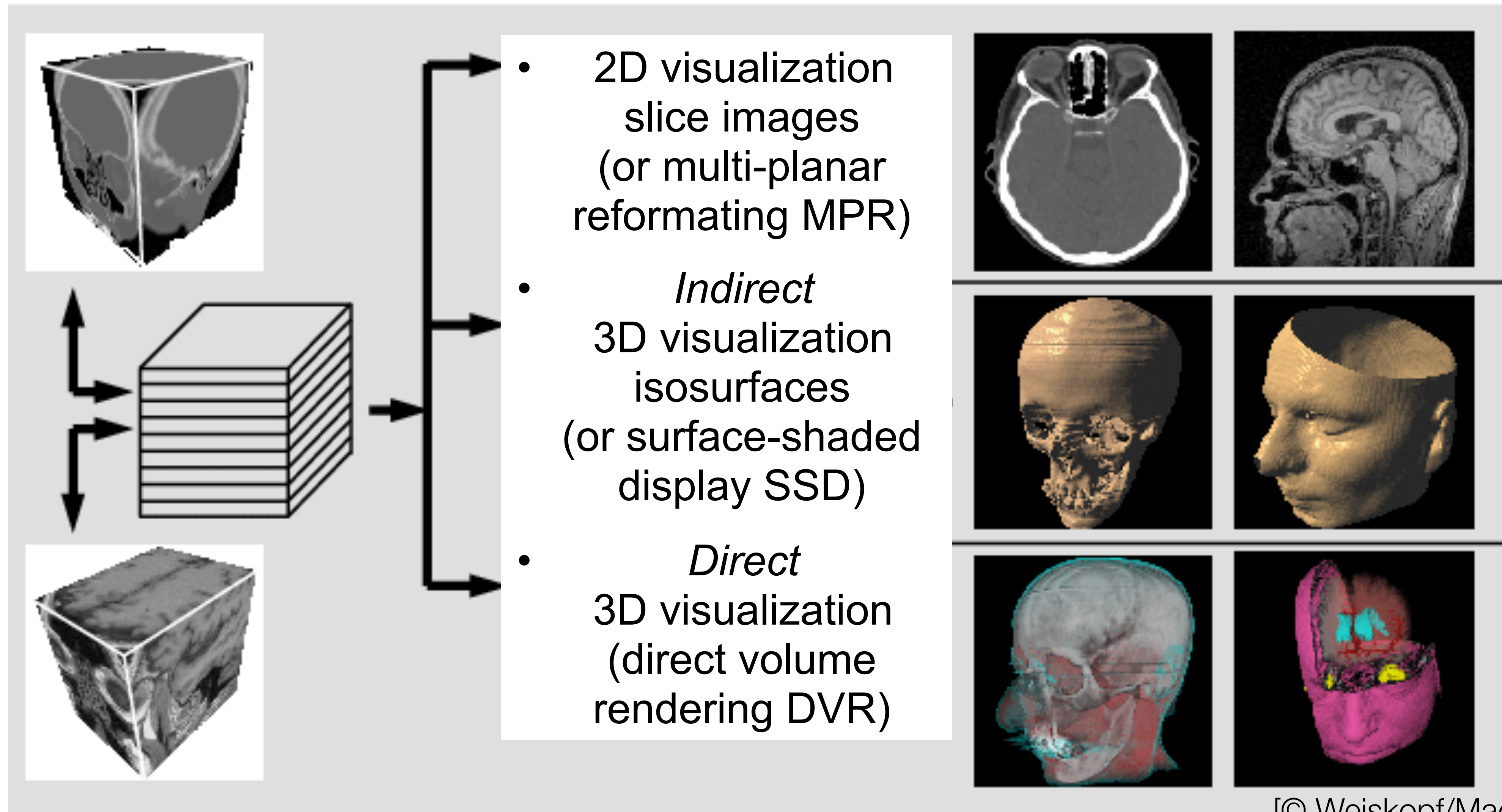


gridpoint

CELL

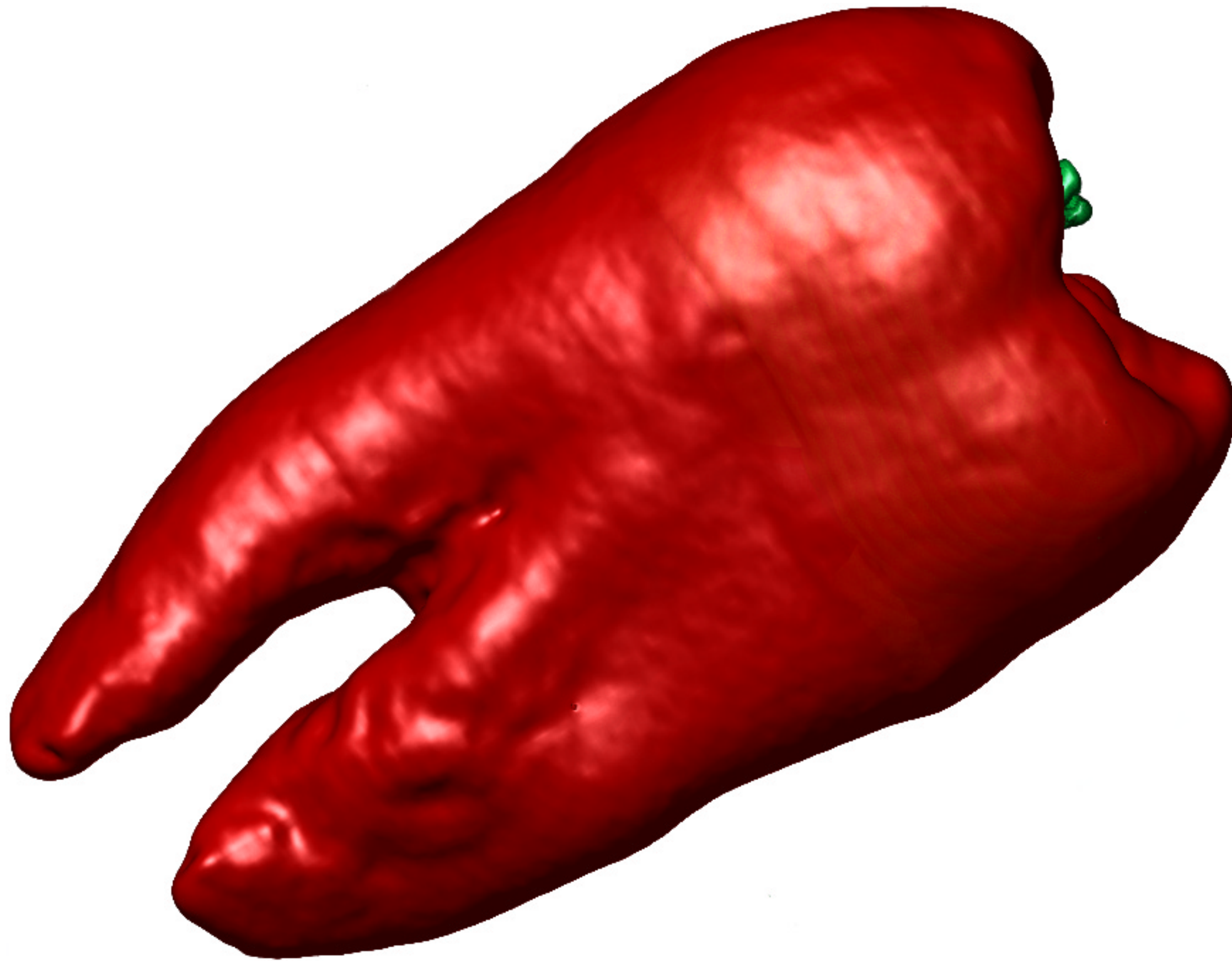
[from <http://www.cs.rug.nl/~michael/FANTOM/FANTOM1a.pdf>]

Visualizing Volume (3D) Data

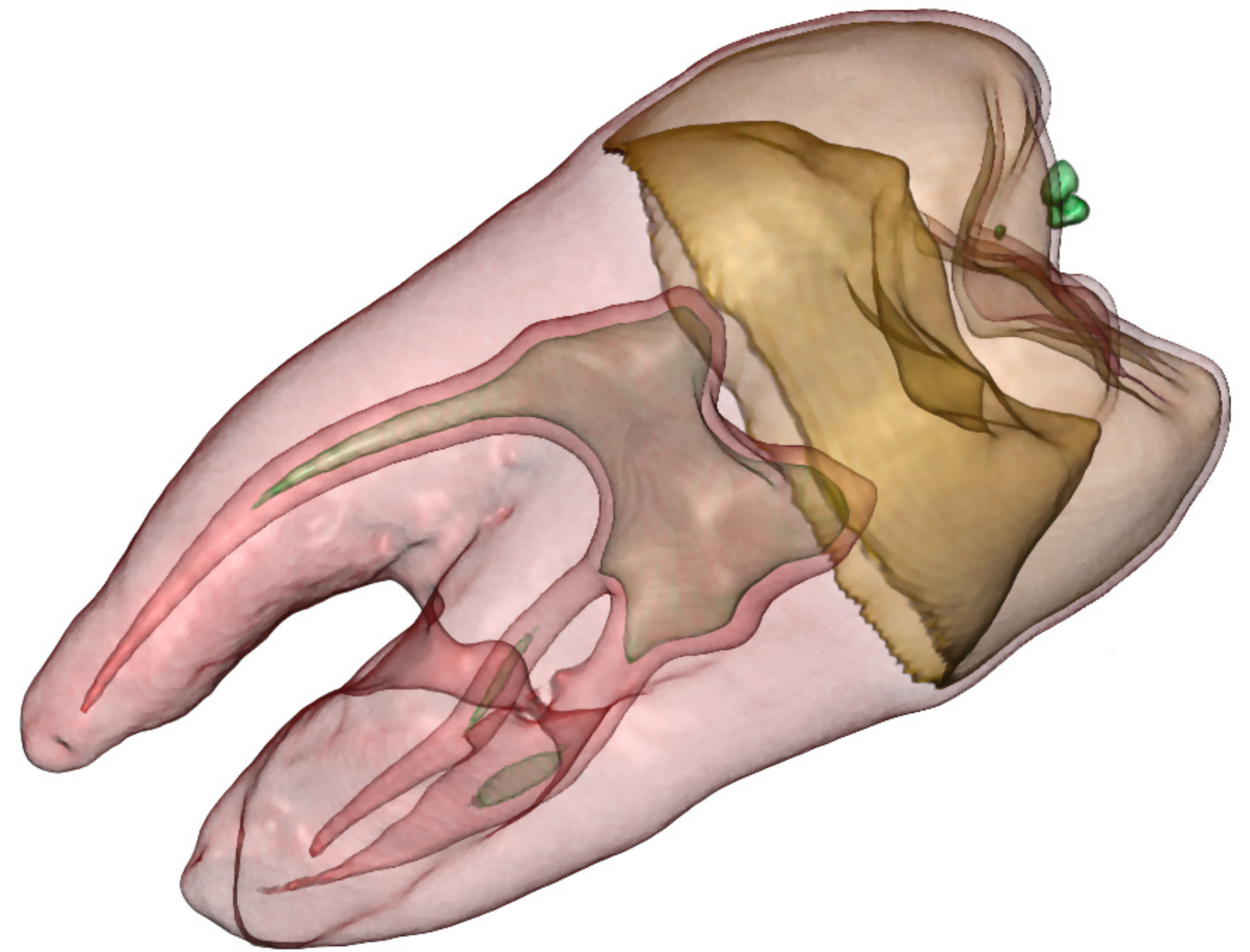


[© Weiskopf/Machiraju/Möller]

Visualizing Volume (3D) Data



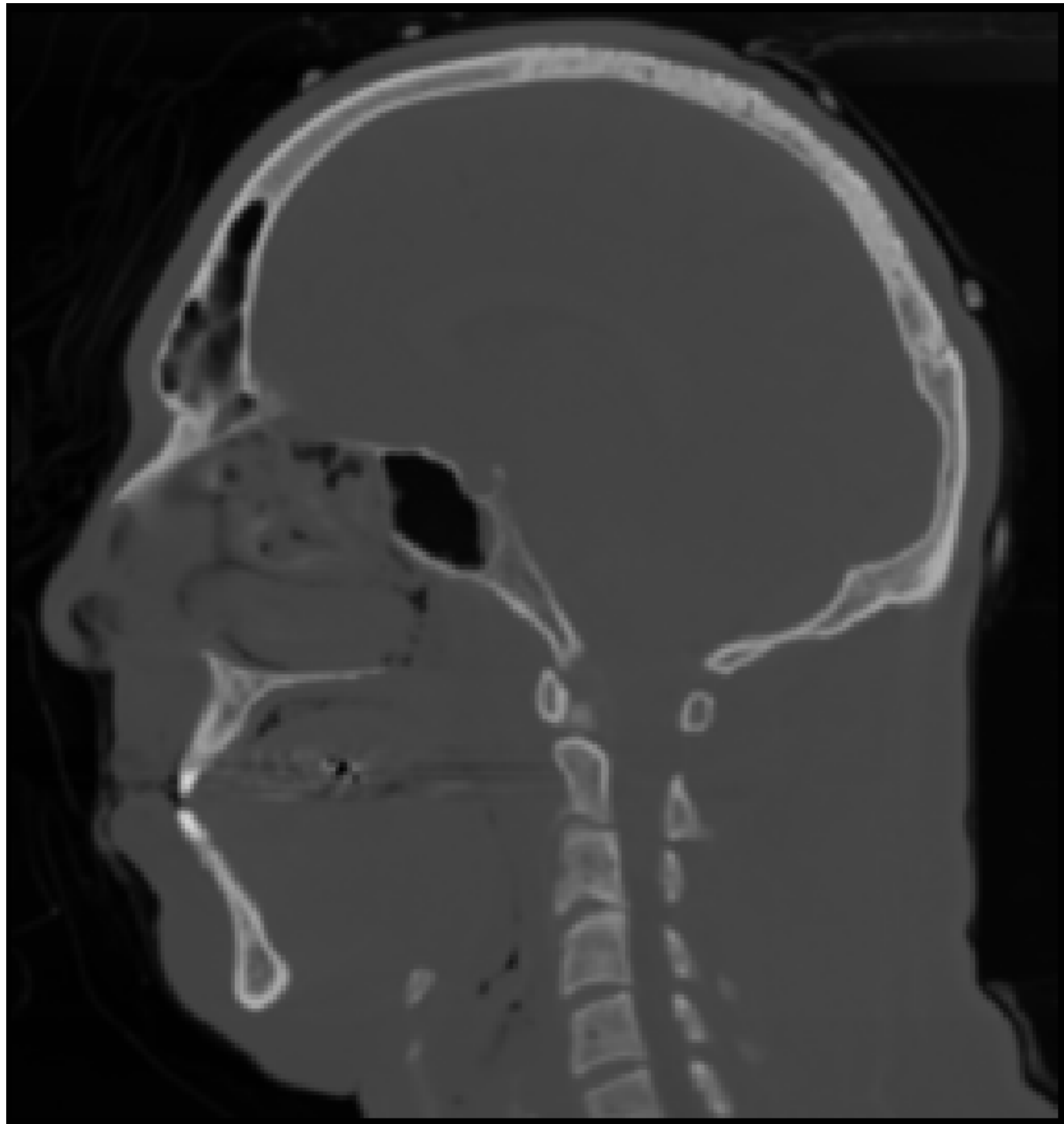
(a) An isosurfaced tooth.



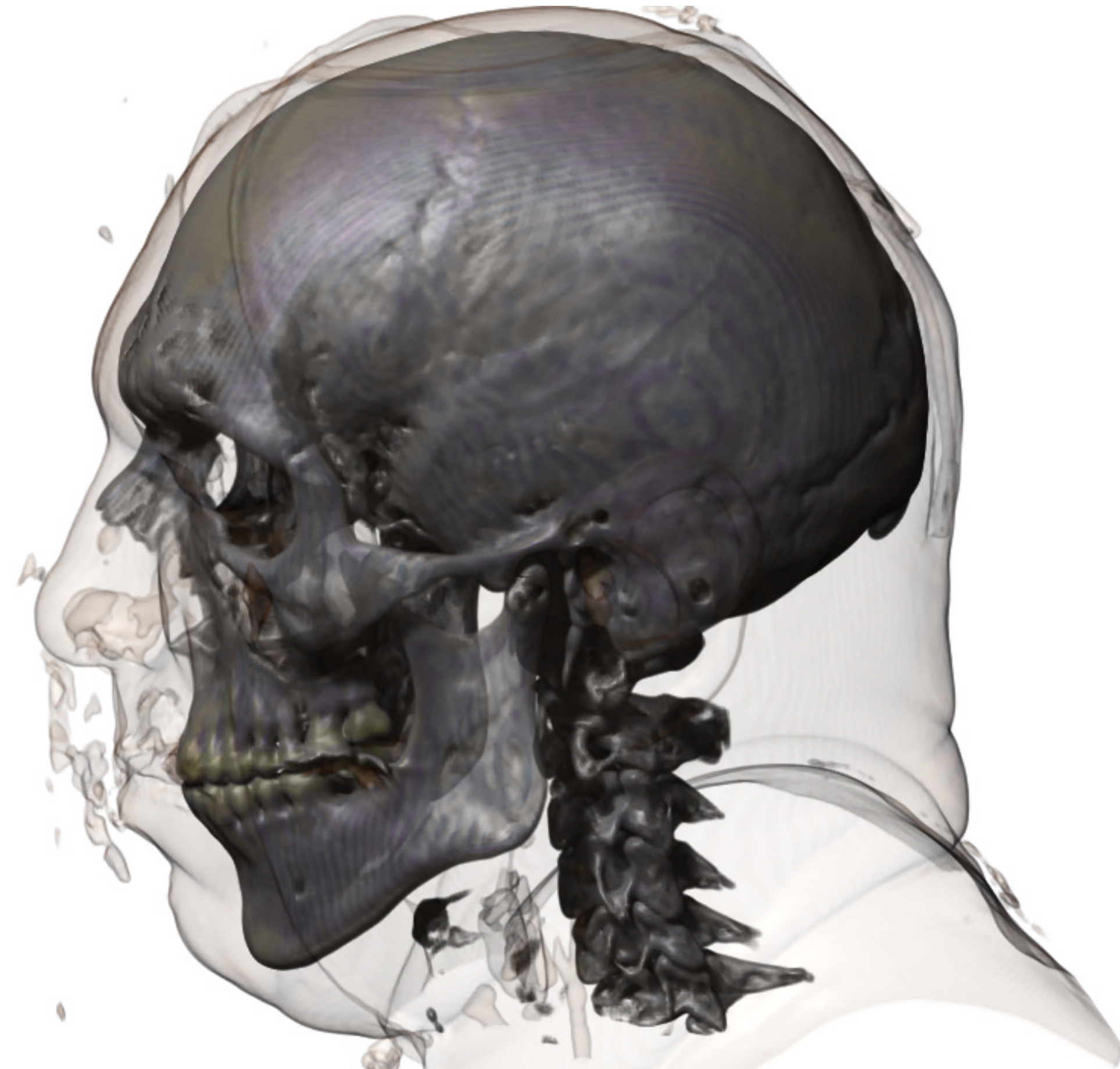
(b) Multiple isosurfaces.

[J. Kniss, 2002]

Visualizing Volume (3D) Data



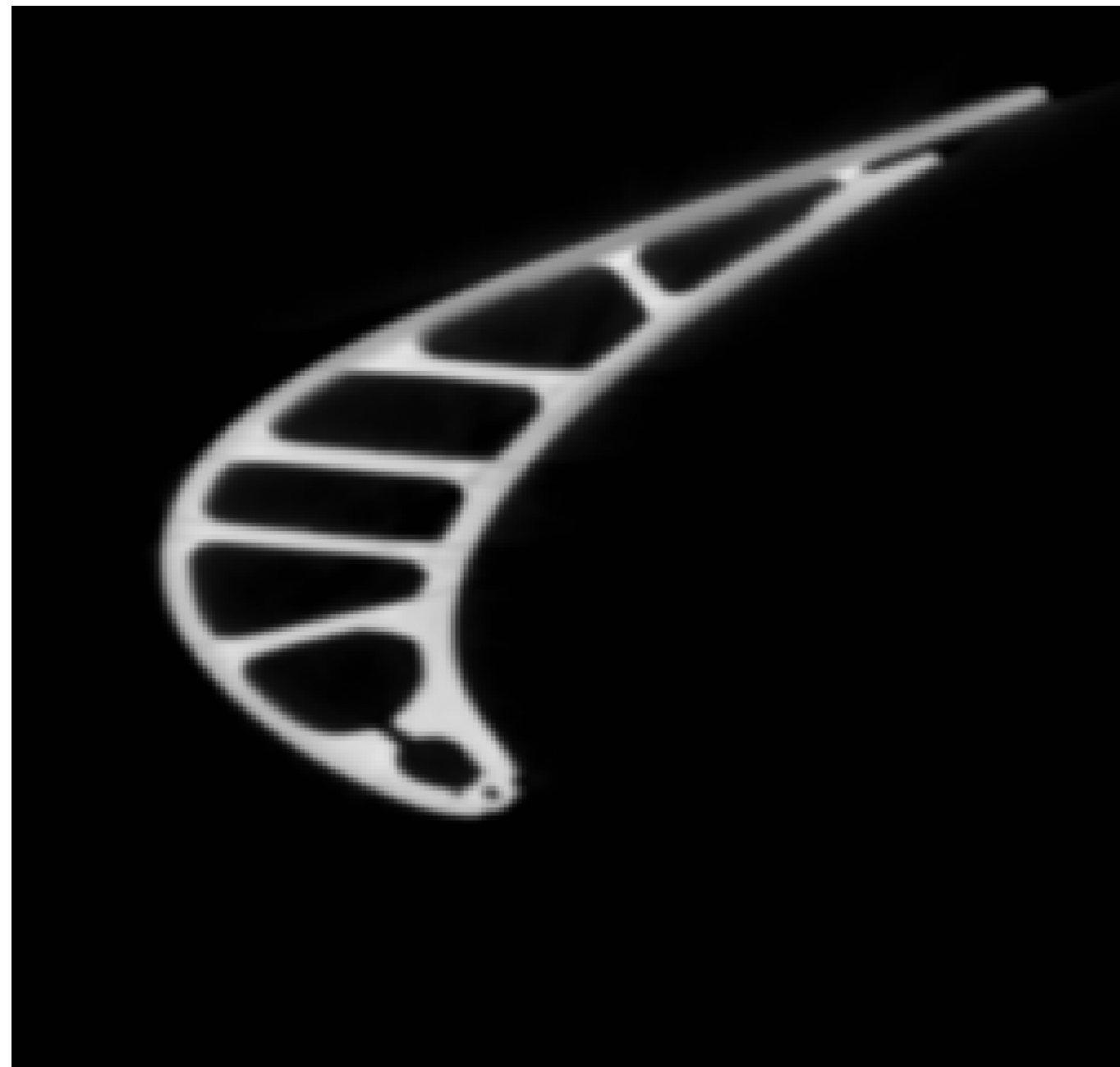
(a) 2D slice



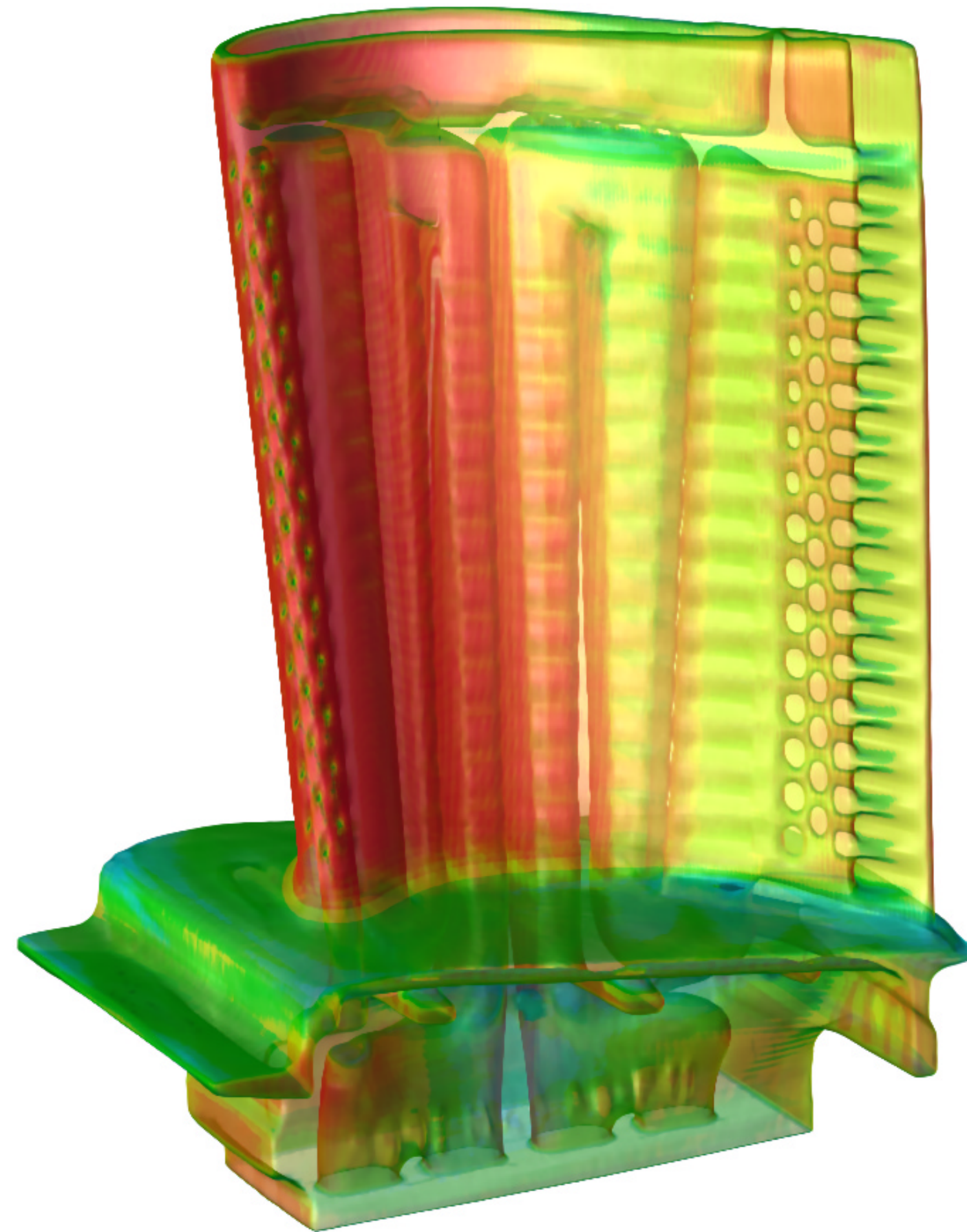
(b) Volume Rendering

[J. Kniss, 2002]

Visualizing Volume (3D) Data



(a) 2D slice



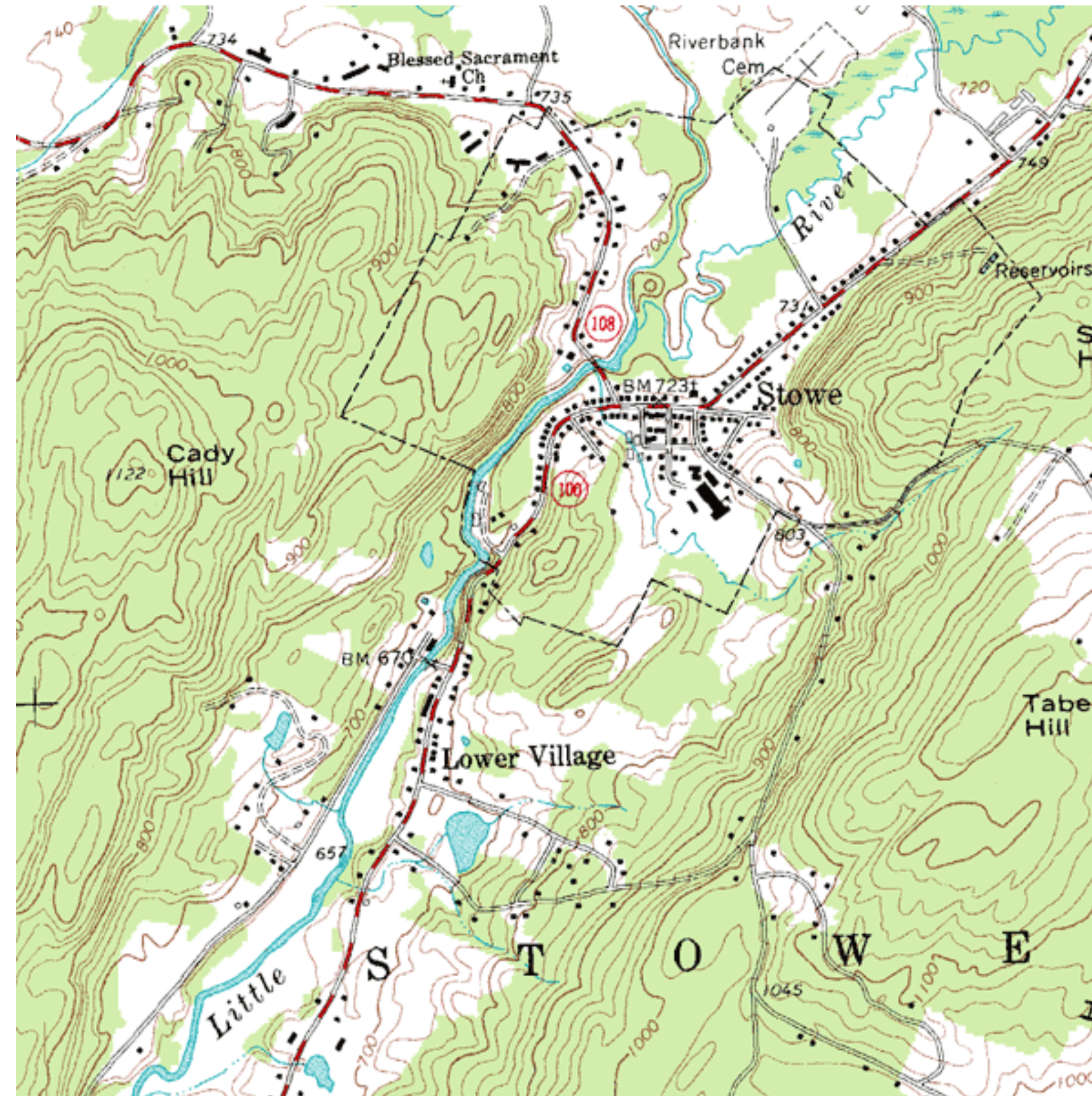
(b) Volume Rendering

[J. Kniss, 2002]

How have we encoded 3D scalar data before?
Hint: Think about elevation maps

Isolines (2D)

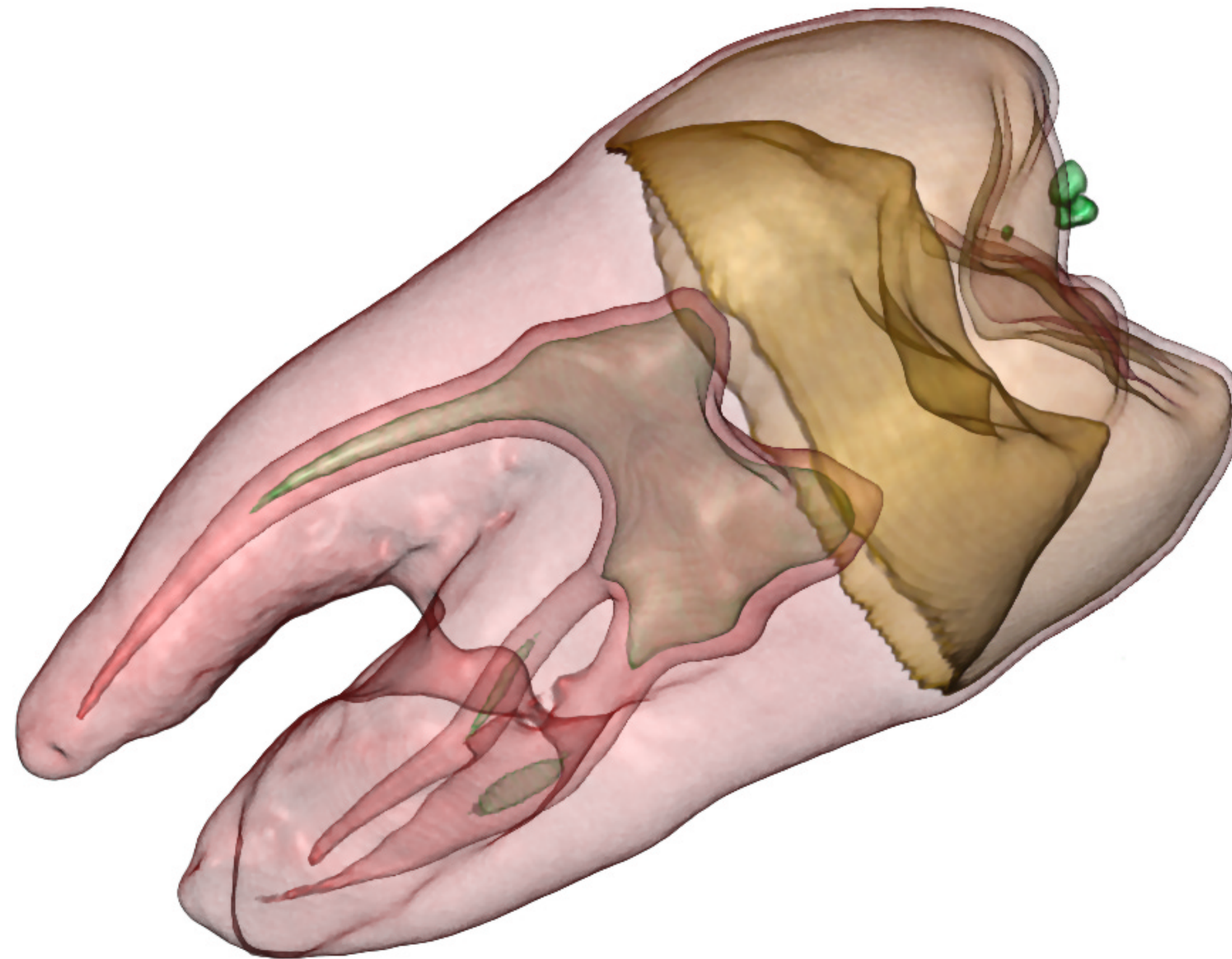
- Isoline: a line that has the same scalar value at all locations
- Example: Topographical Map



[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

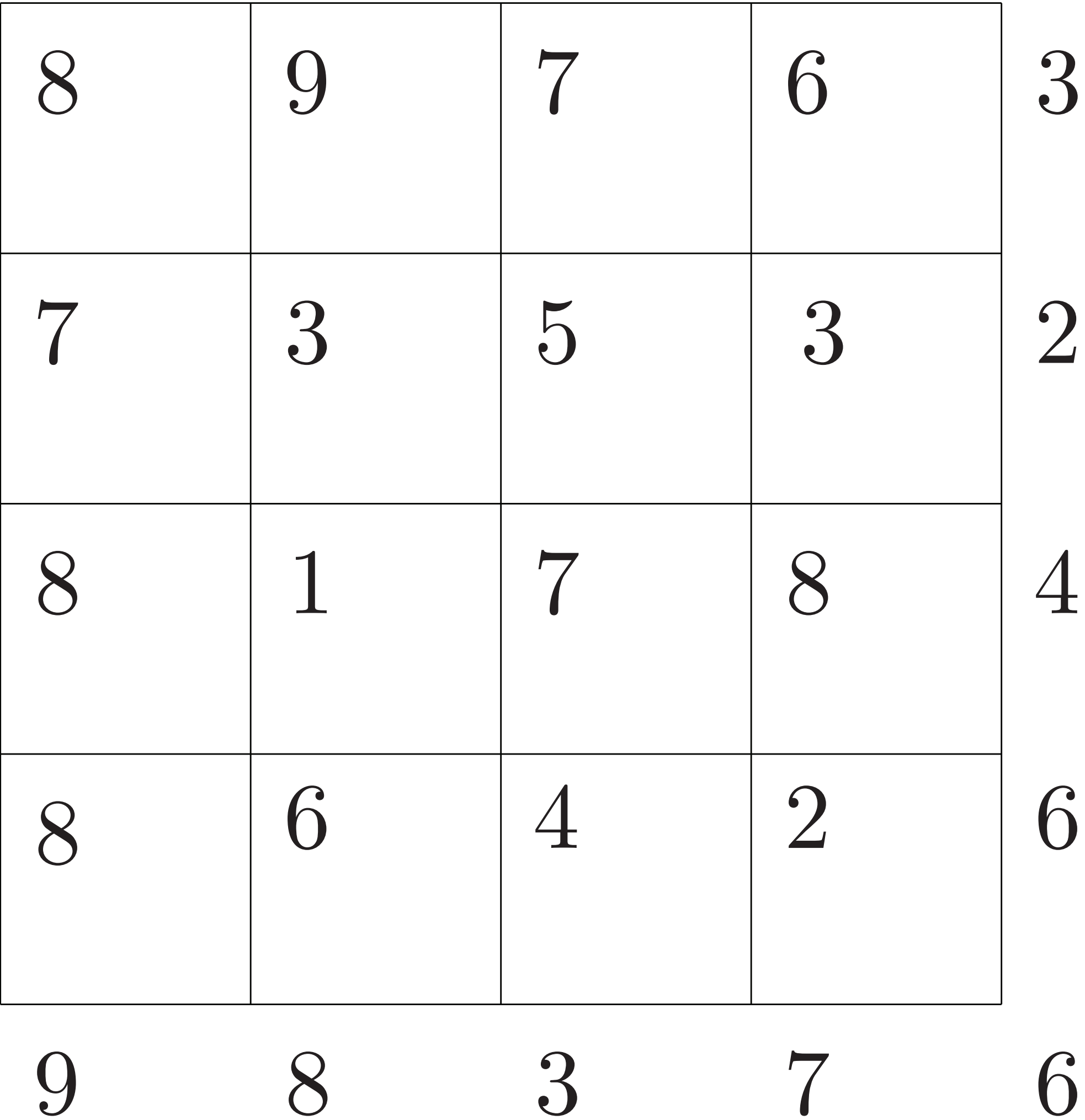


[J. Kniss, 2002]

How?

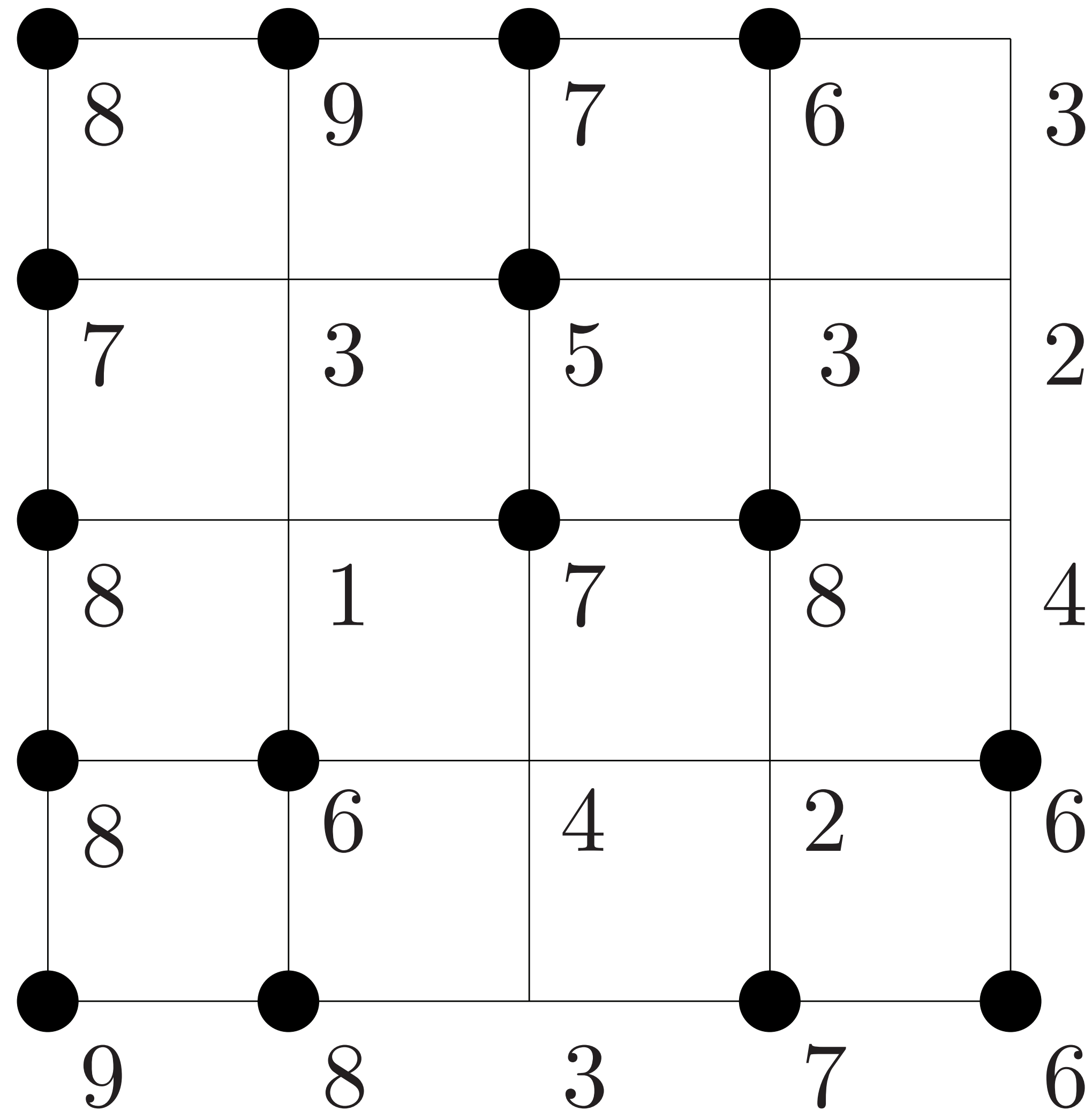
- Given an **isovalue**, we want to draw the isocontours corresponding to that value
- Remember we only have values defined at grid points
- How do we get isolines or isosurfaces from that data?
- Can we use the ideas from interpolation?

Generating Isolines (Isovalue = 5)



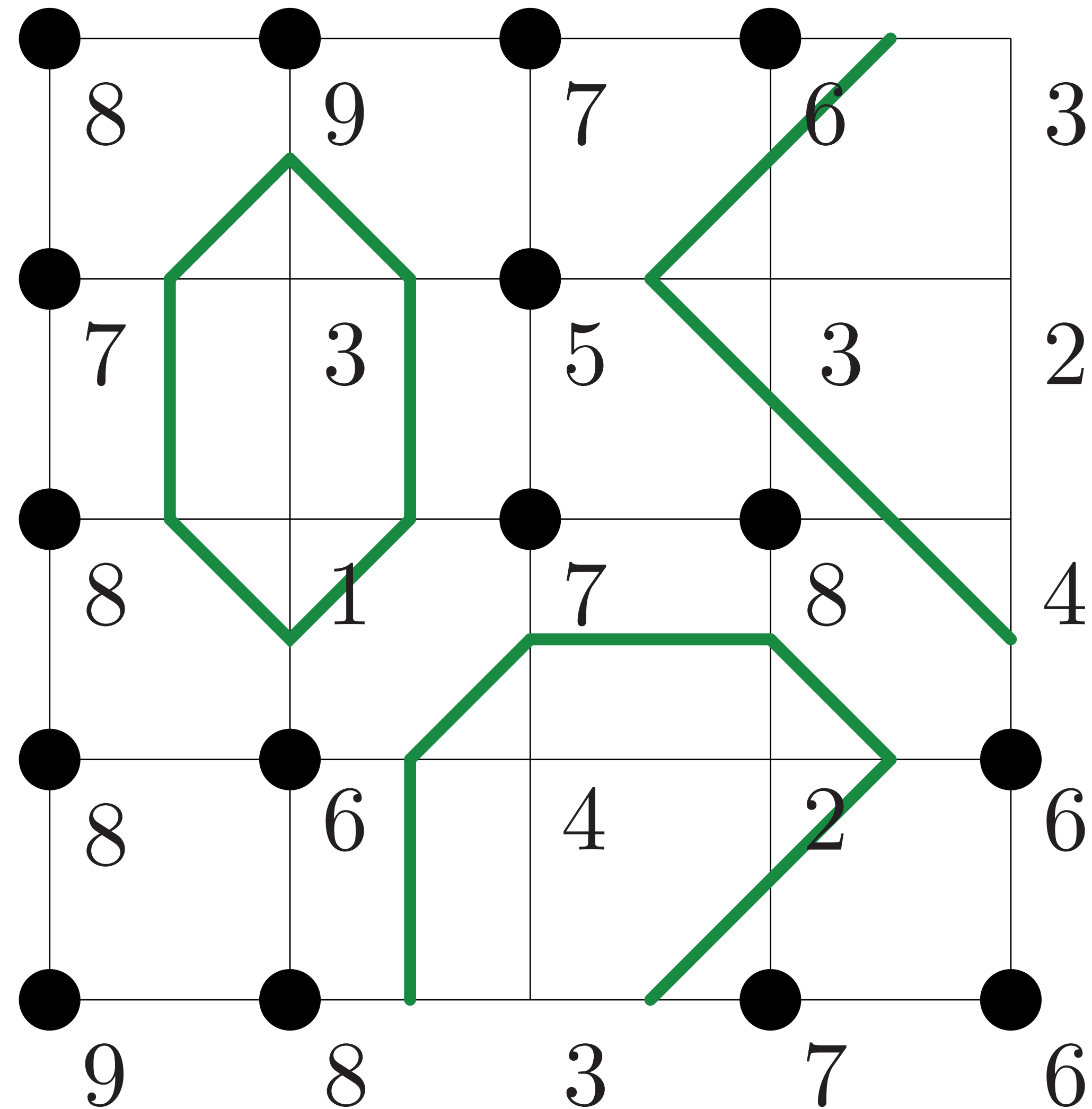
[R. Wenger, 2013]

Generating Isolines



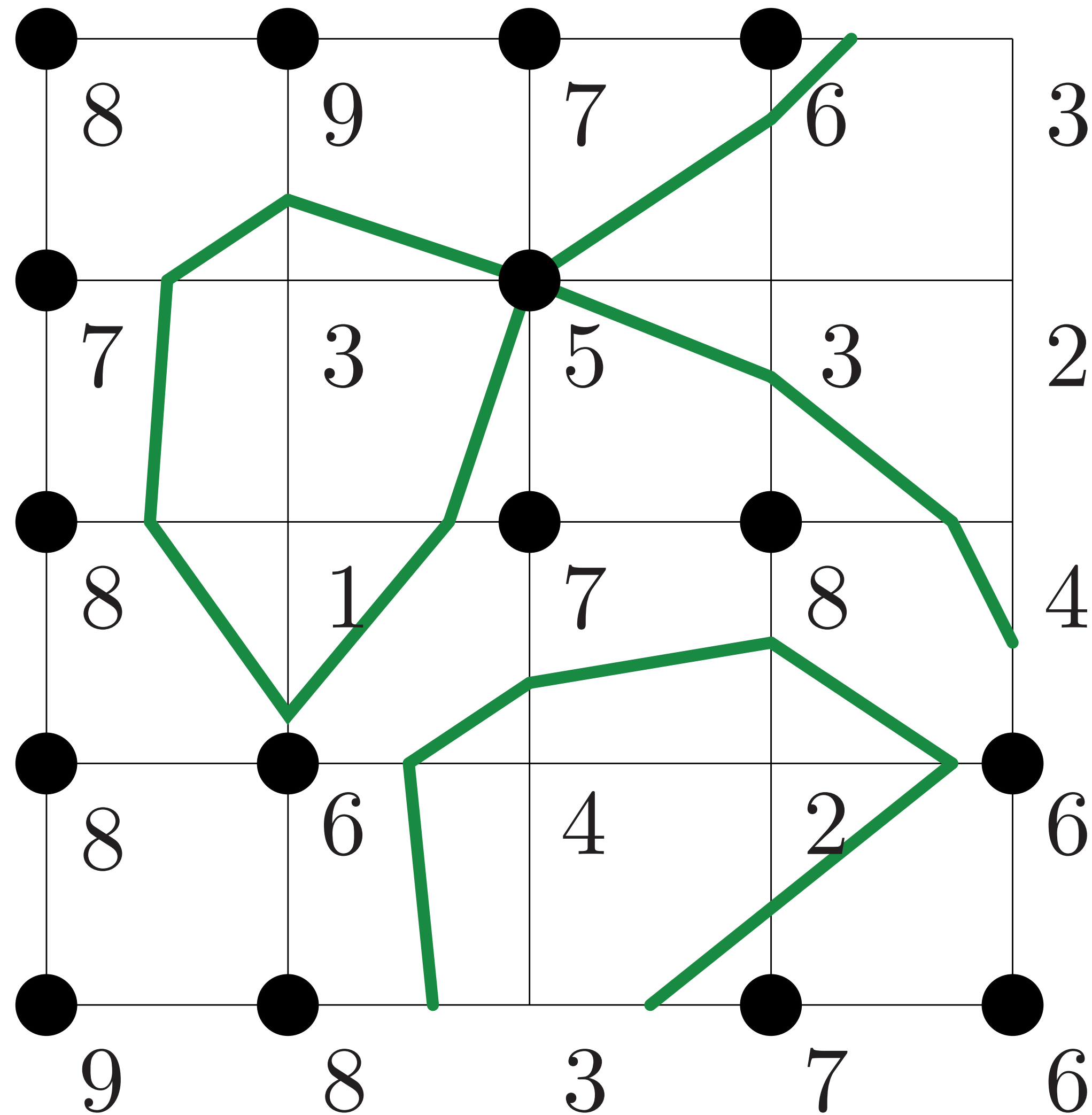
[R. Wenger, 2013]

Generating Isolines



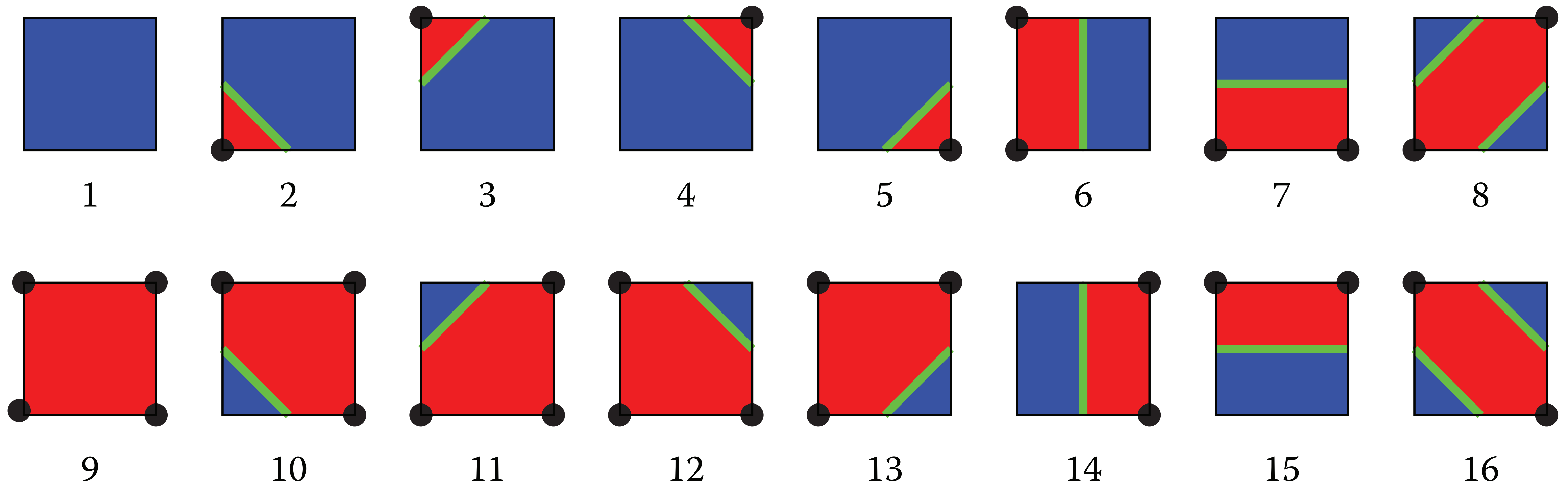
[R. Wenger, 2013]

Generating Isolines



[R. Wenger, 2013]

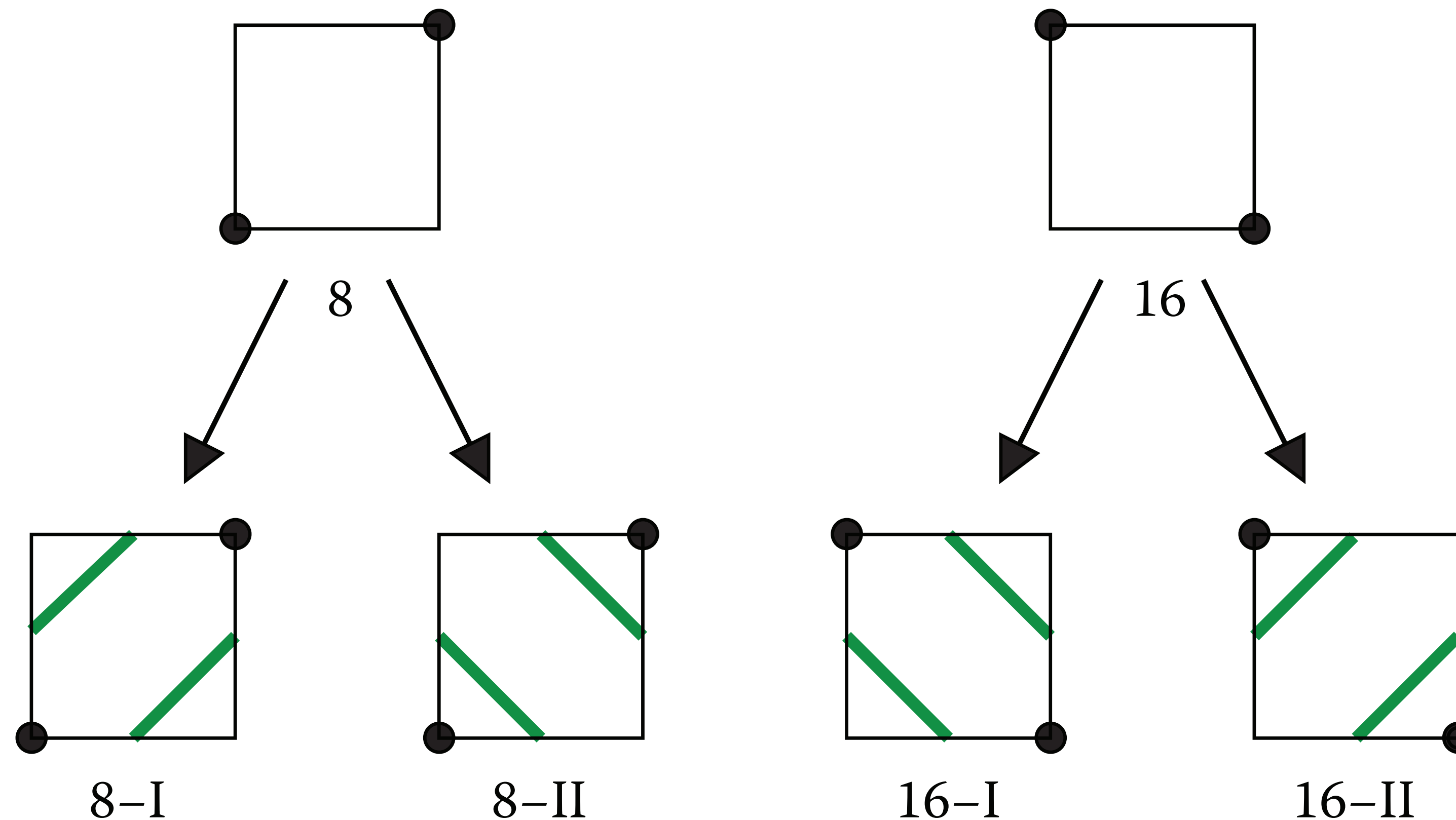
Marching Squares



[R. Wenger, 2013]

Ambiguous Configurations

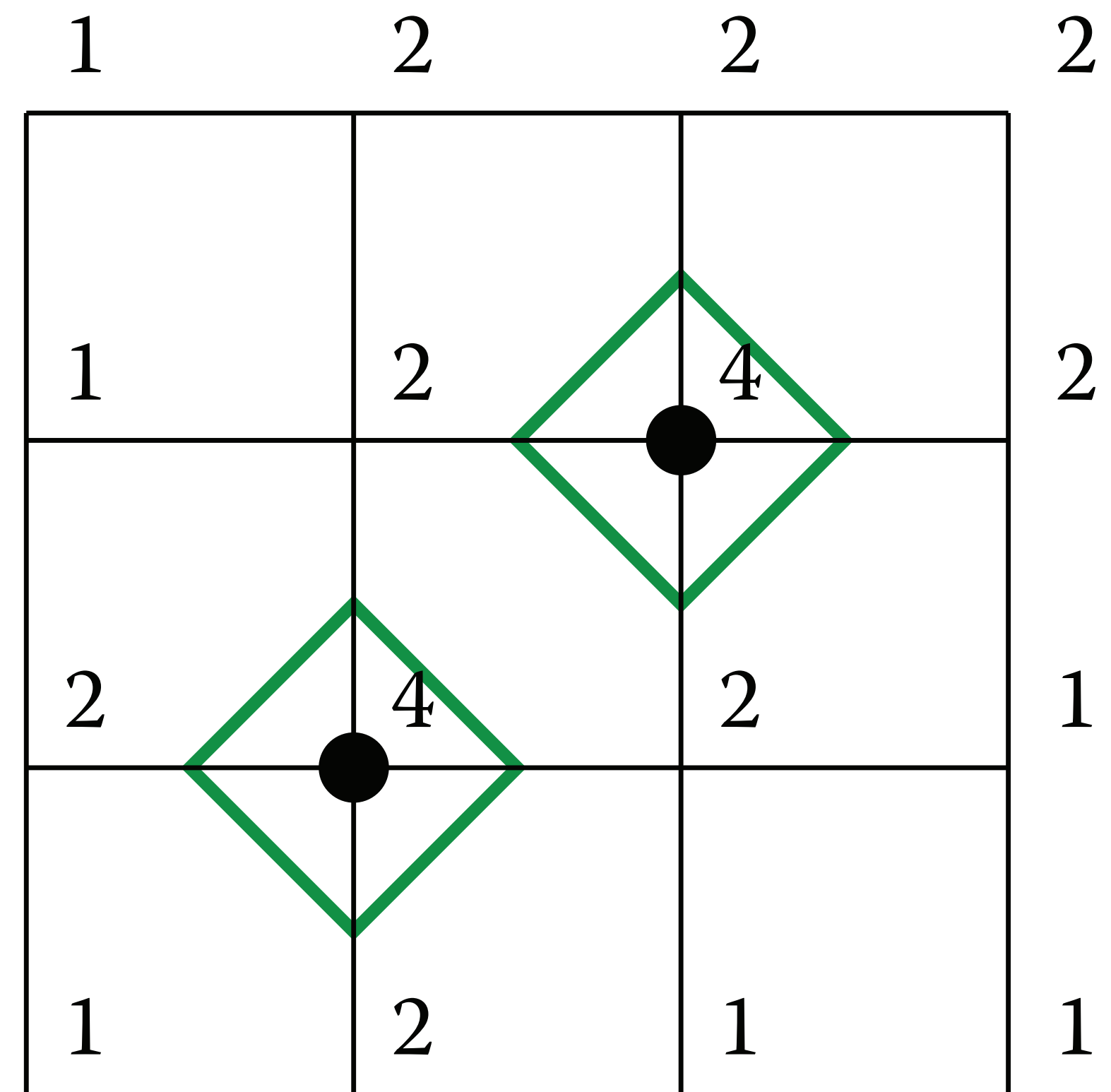
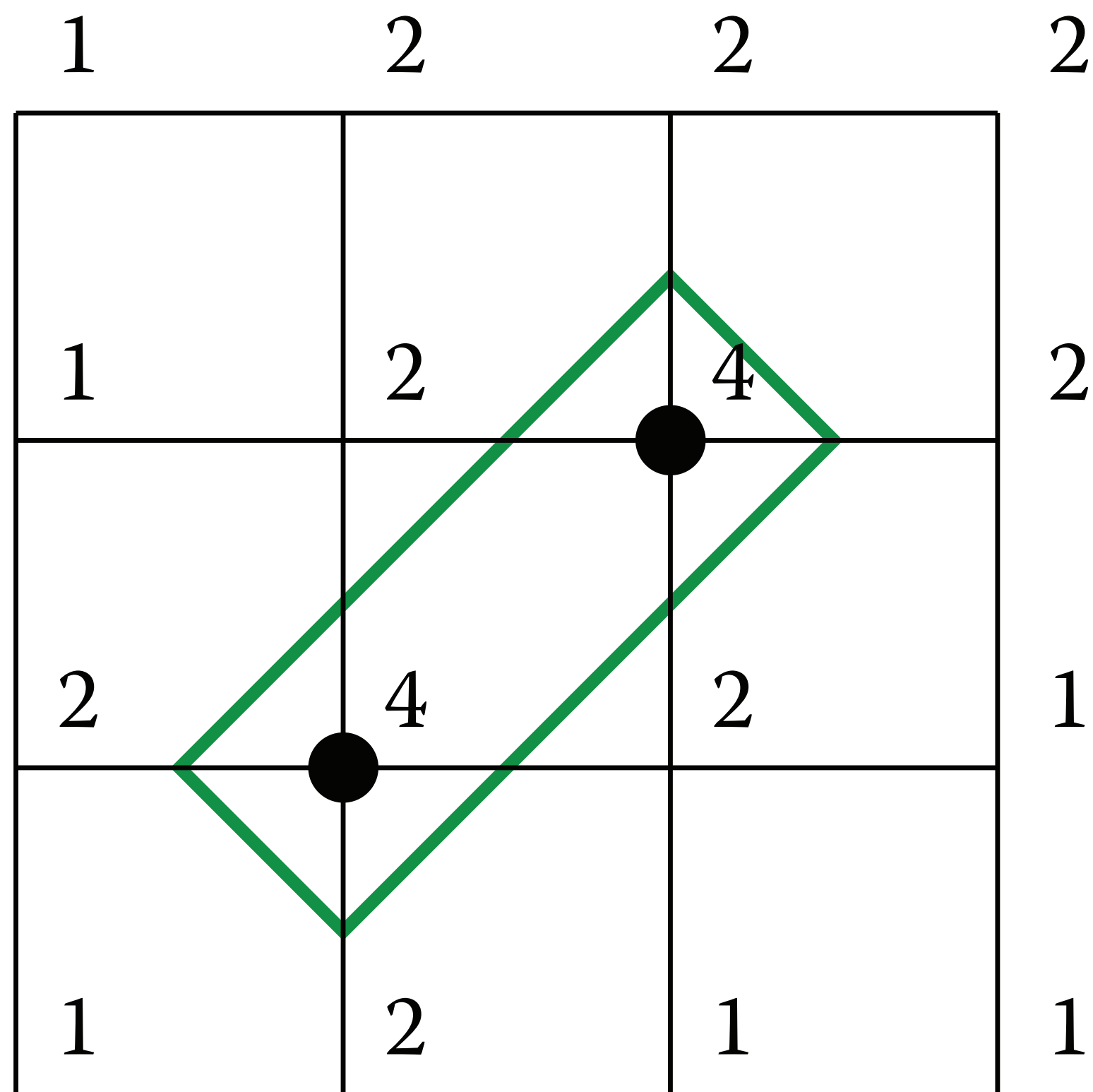
- There are some cases for which we cannot tell which way to draw the isolines...



[R. Wenger, 2013]

Ambiguous Configurations

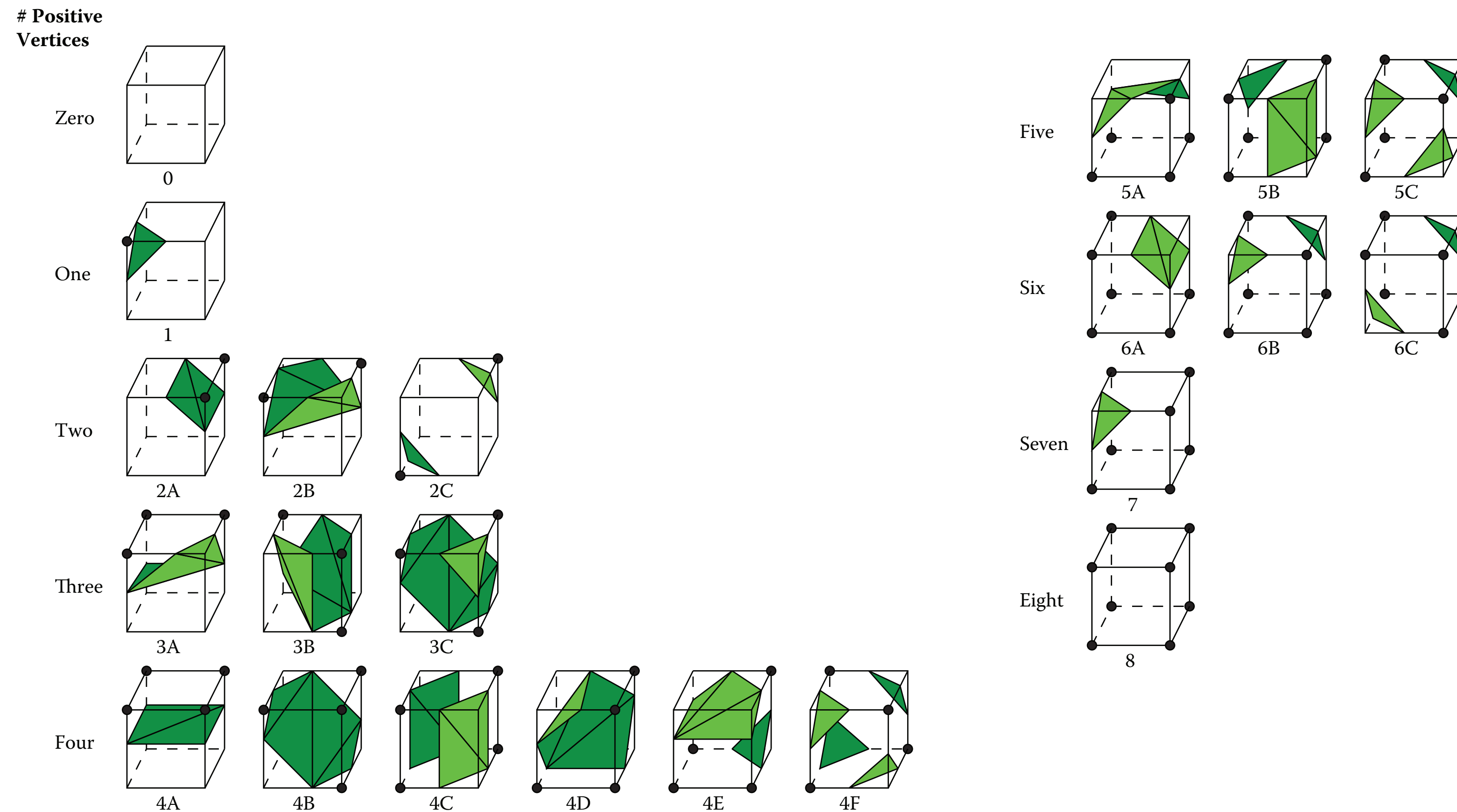
- Either works for marching squares, this isn't the case for 3D



[R. Wenger, 2013]

3D: Marching Cubes

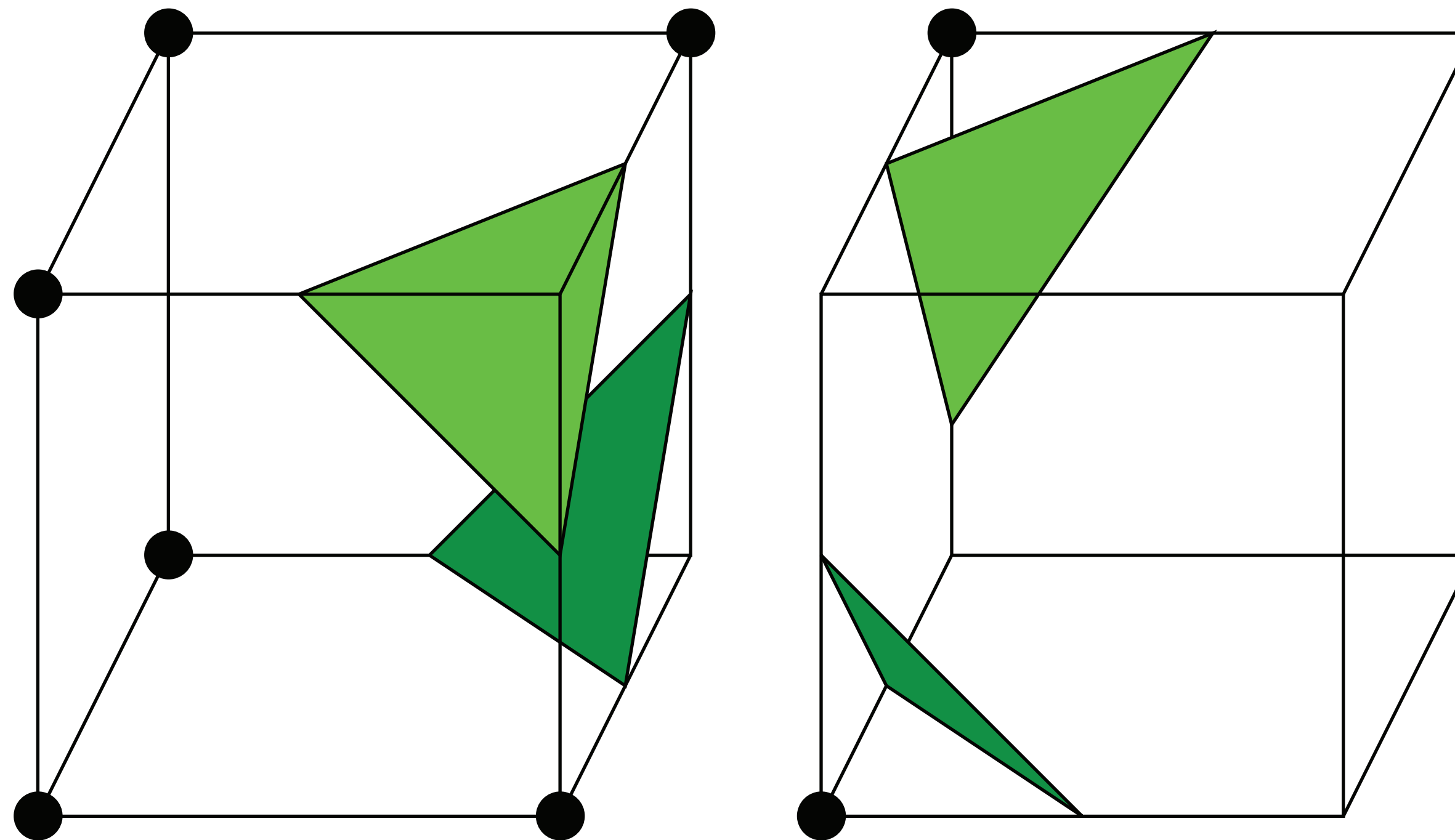
- Same idea, more cases [Lorensen and Cline, 1987]



[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]



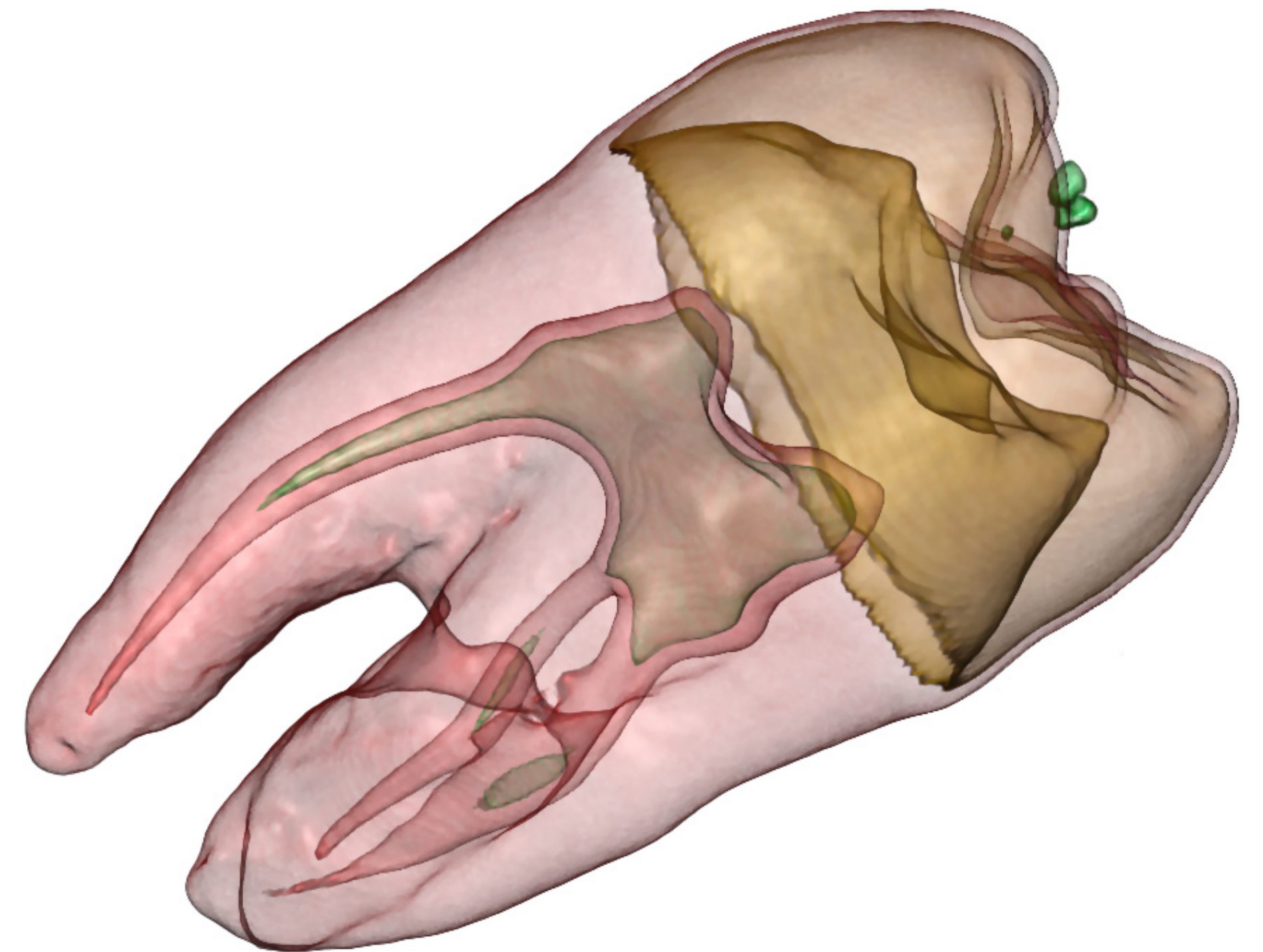
[R. Wenger, 2013]

Marching Cubes Algorithm

- For each cell:
 - Classify each vertex as inside or outside (\geq , $<$) — 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

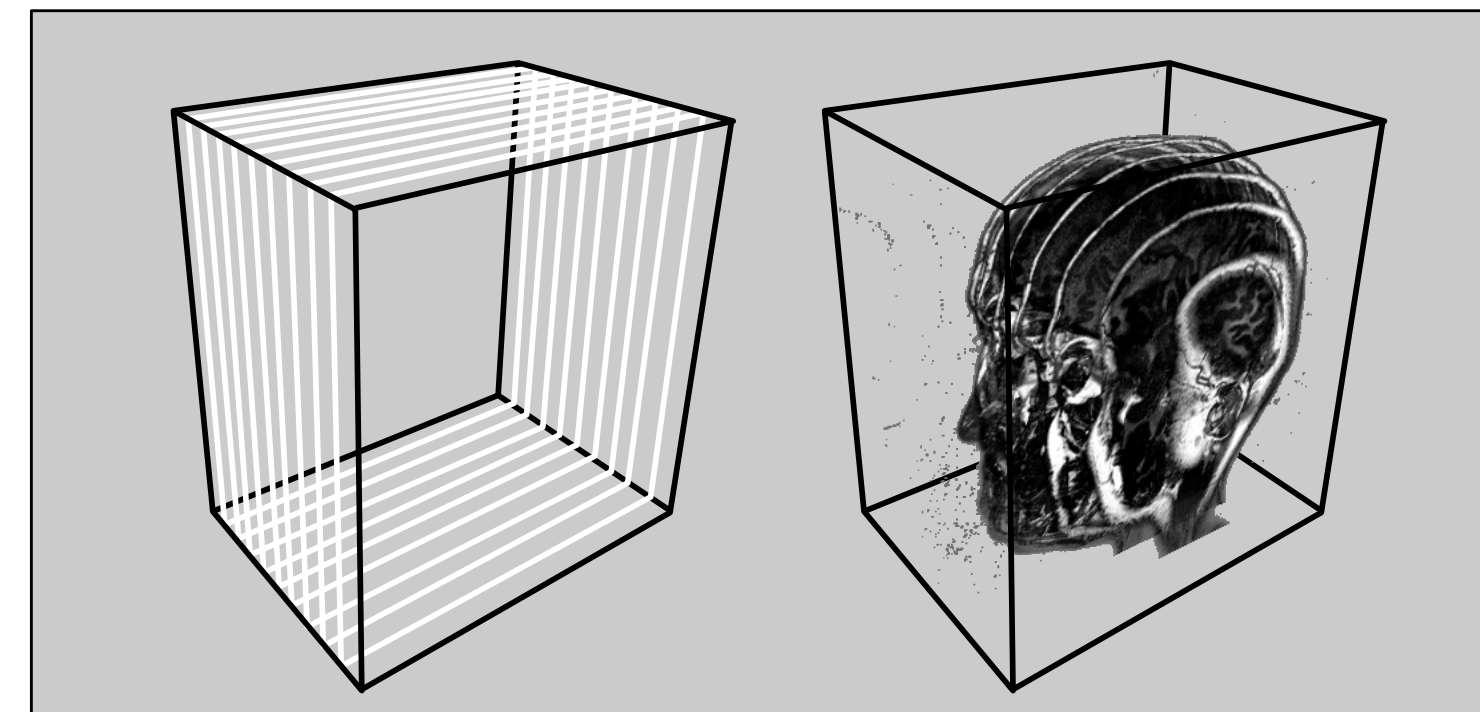
[Kindlmann, 1998]

(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

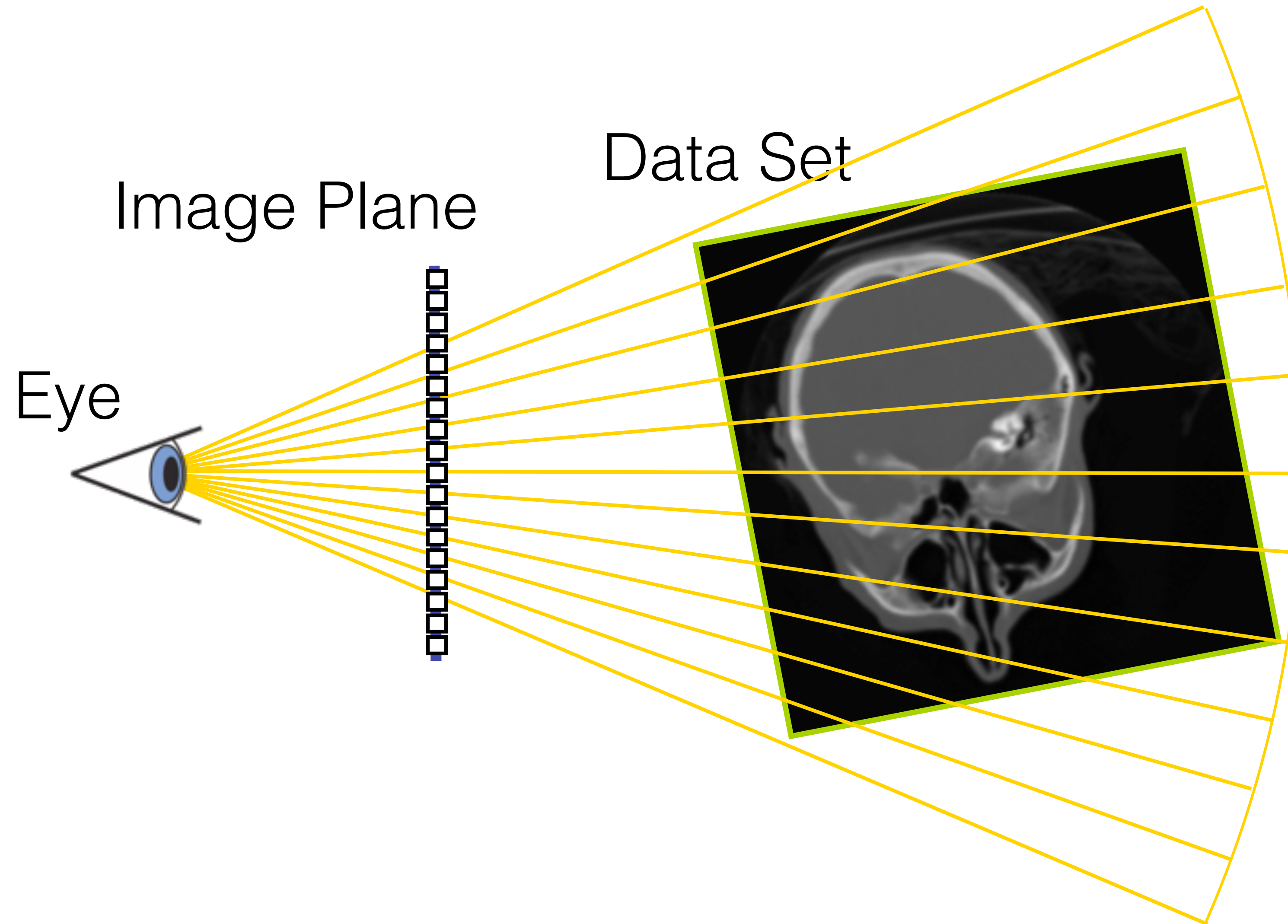
Types of Volume Rendering Algorithms

- 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



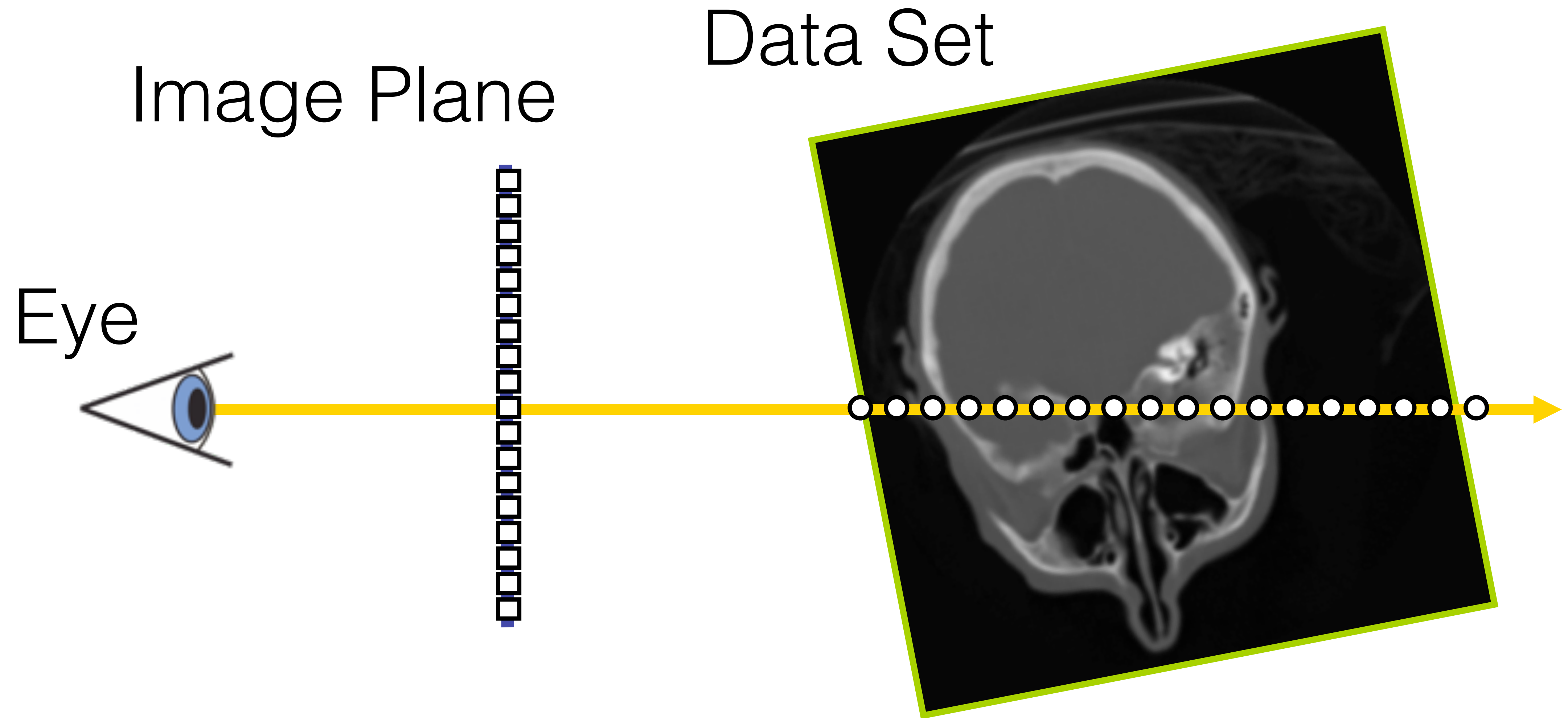
[via Möller]

Volume Ray Casting



[Levine]

Volume Ray Casting



[Levine]