Data Visualization (CSCI 490/680)

Data & Isosurfacing

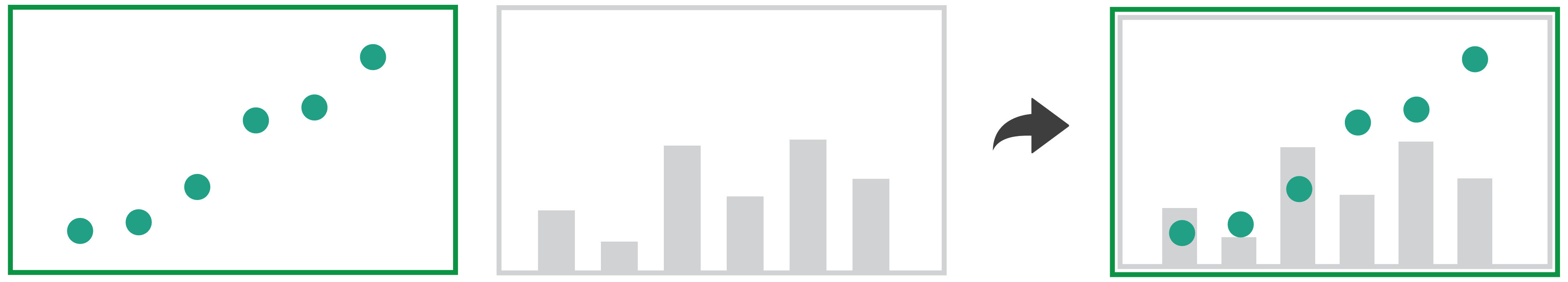
Dr. David Koop

Focus+Content Overview

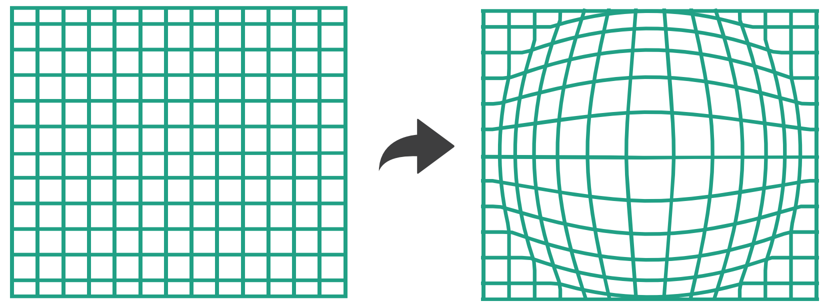
Embed

→ Elide Data

→ Superimpose Layer

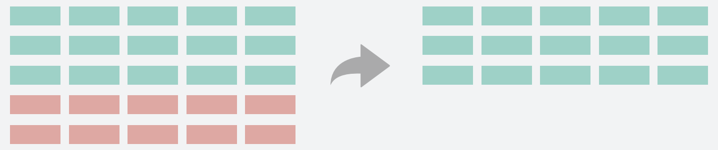


→ Distort Geometry

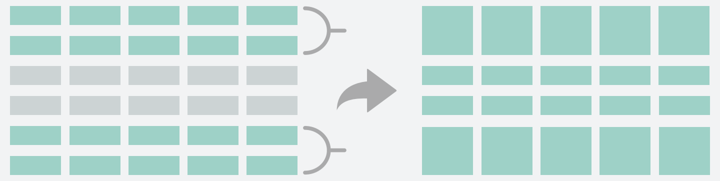


Reduce

Filter



Aggregate

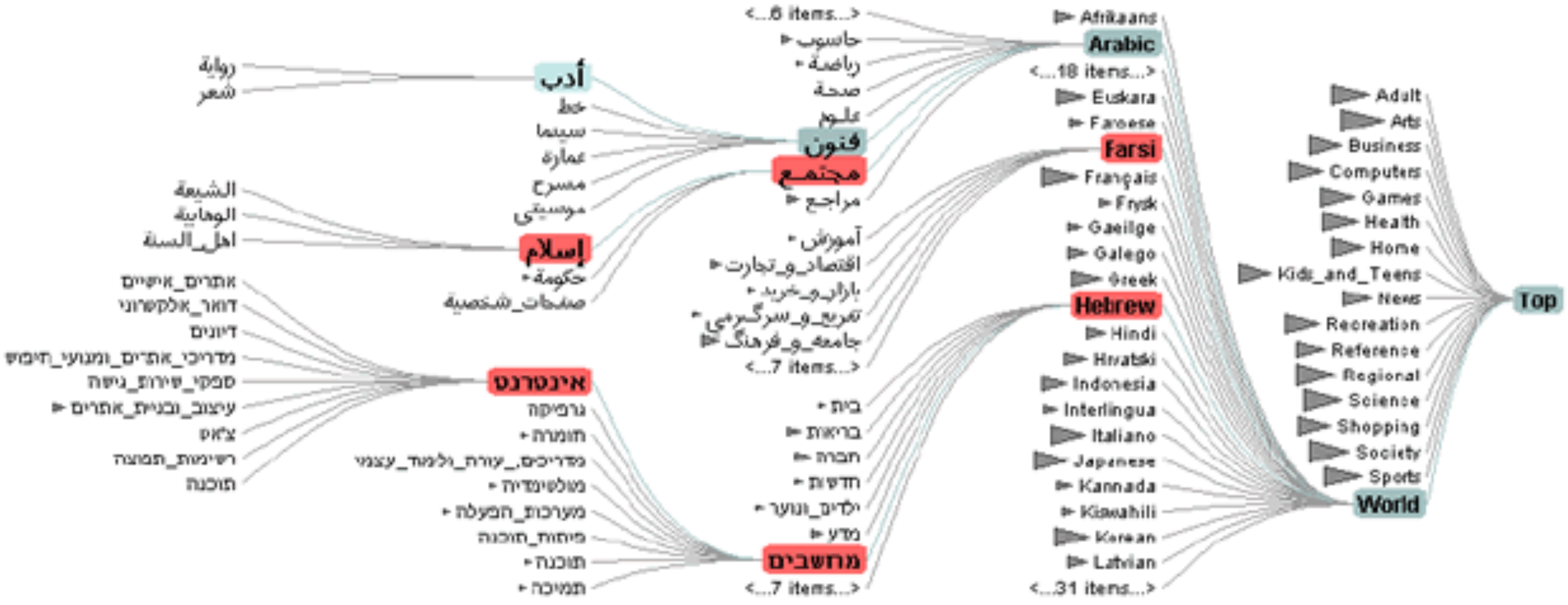


Embed

[Munzner (ill. Maguire), 2014]

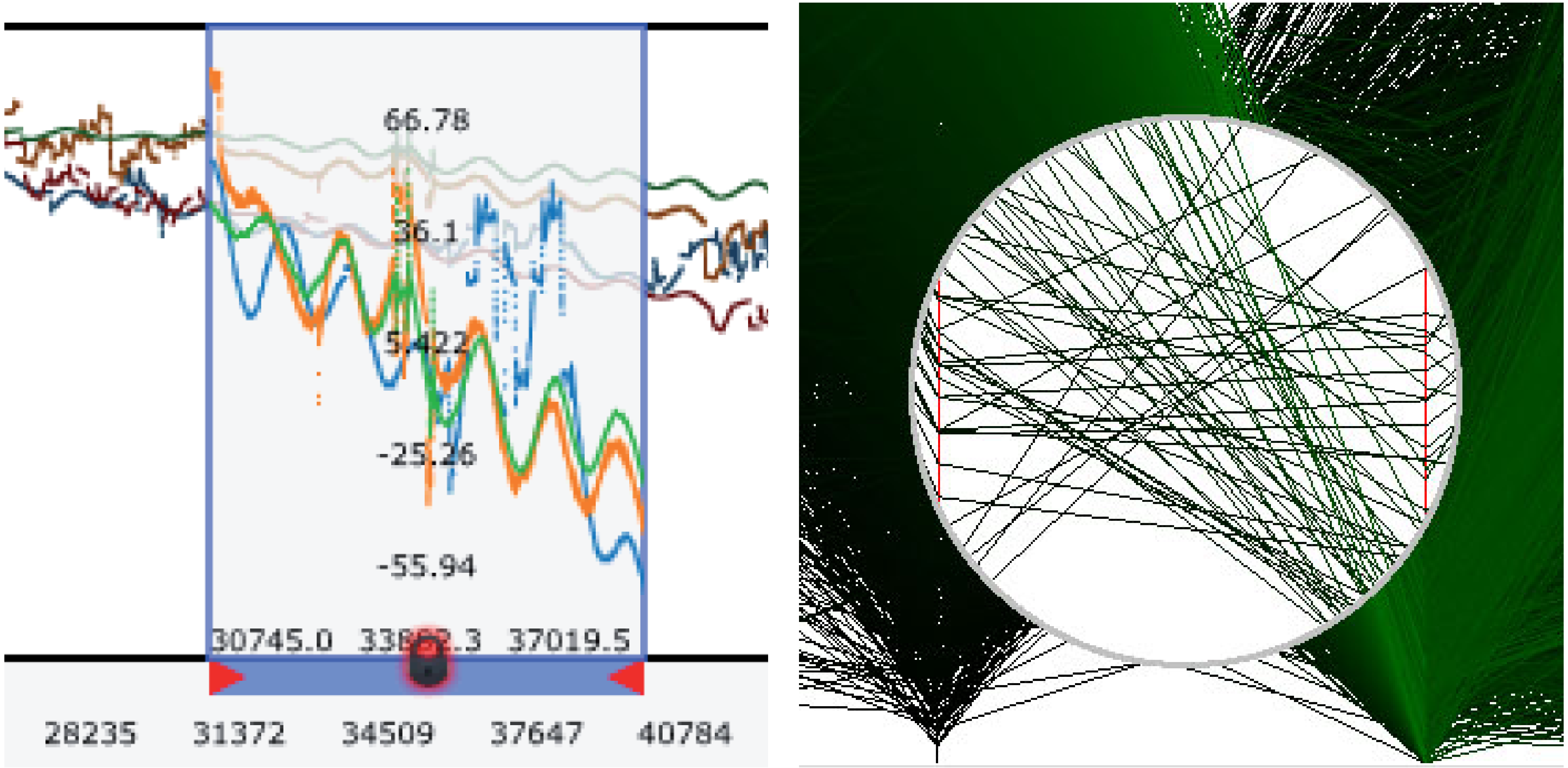
Elision & Degree of Interest Function

- DOI = I(x) - D(x,y)
  - I: interest function
  - D: distance (semantic or spatial)
  - x: location of item
  - y: current focus point
  - Interactive: y changes



[Heer and Card, 2004]

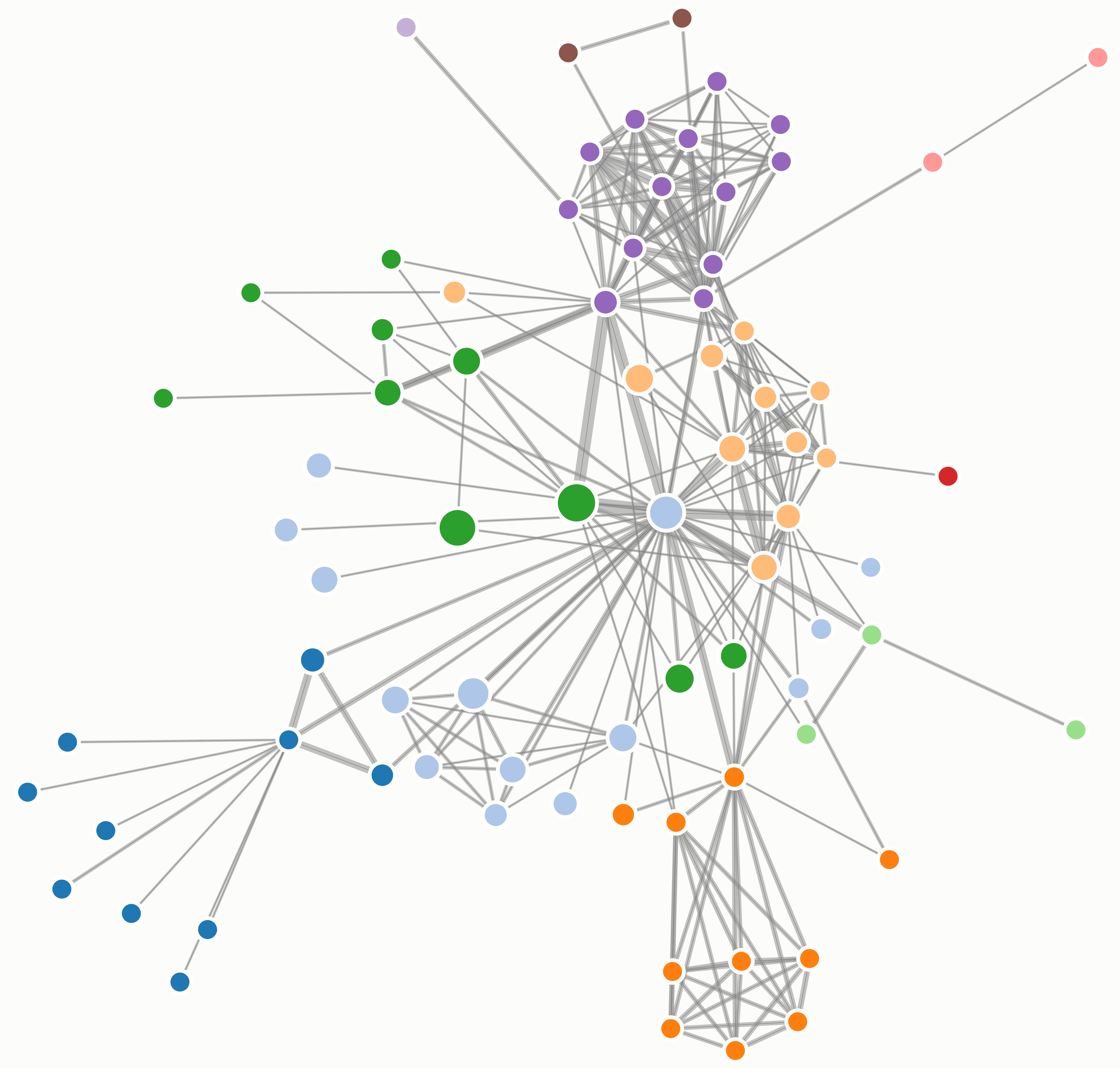
Superimposition with Interactive Lenses



(a) Alteration (b) Suppression

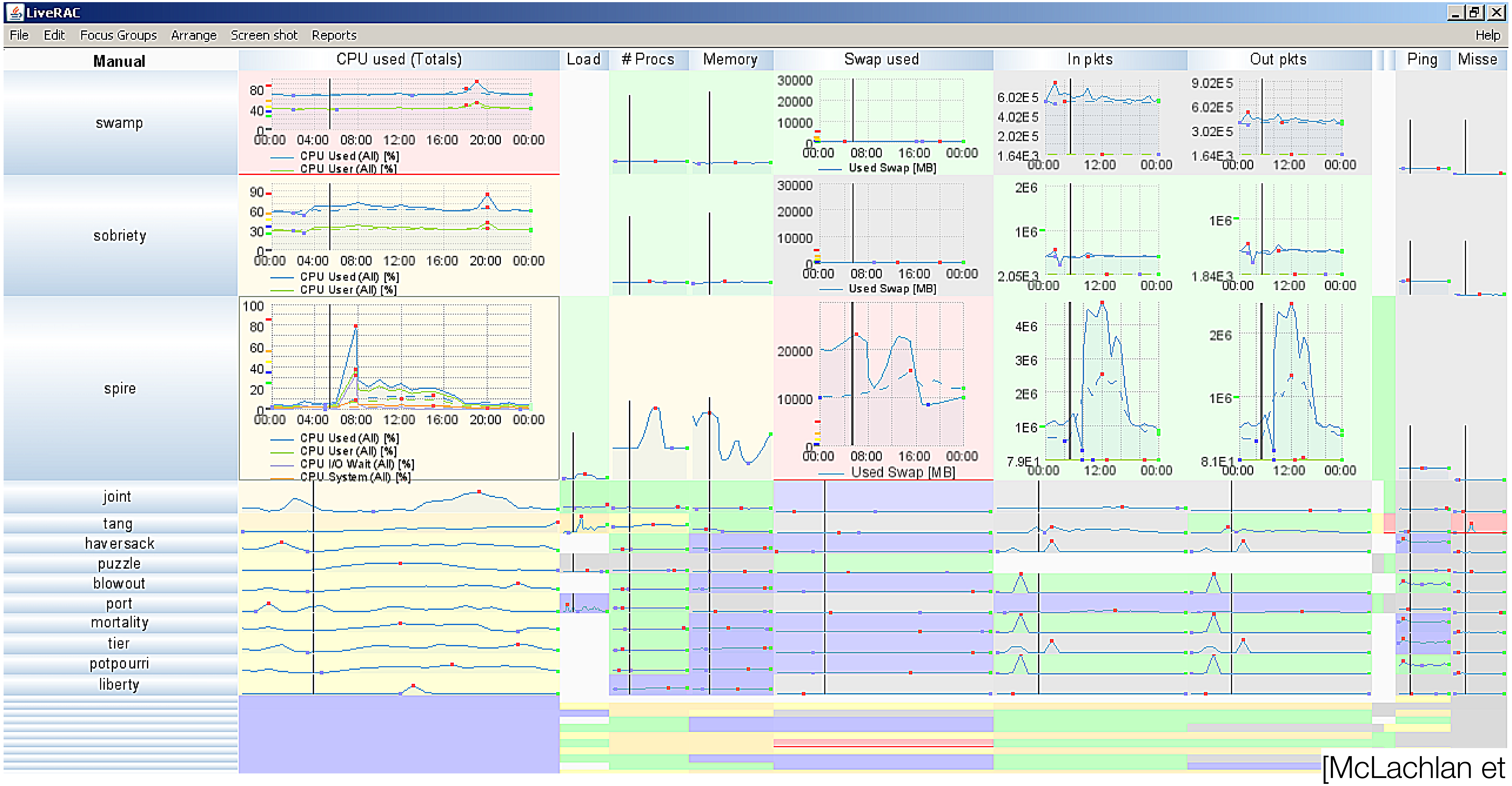
[ChronoLenses and Sampling Lens in Tominski et al., 2014]

Distortion



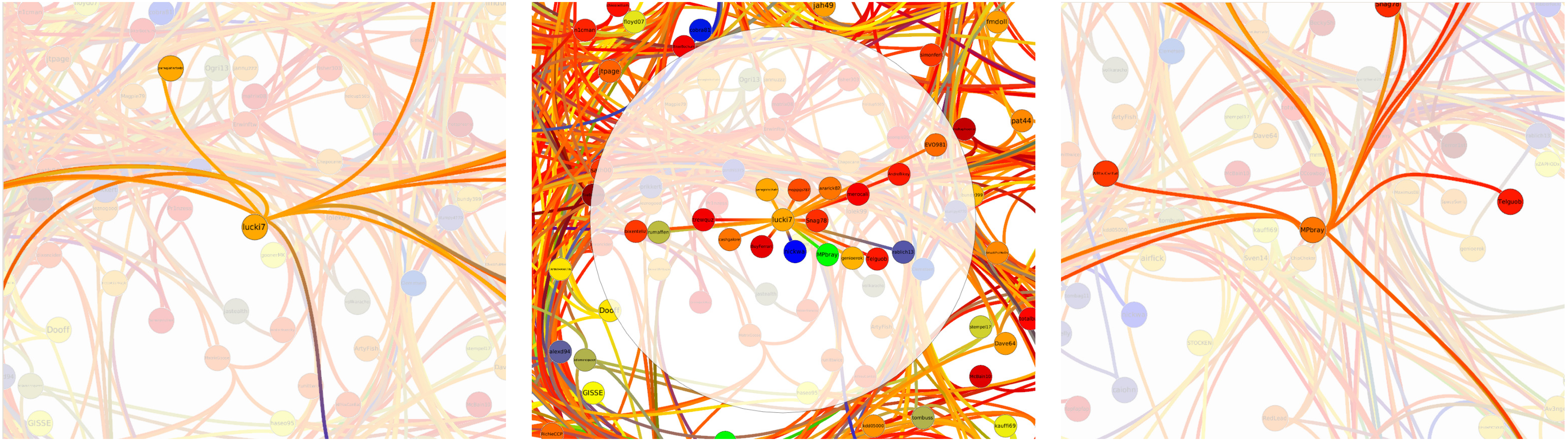
[M. Bostock]

Distortion: Stretch and Squish Navigation



[McLachlan et al., 2008]

Focus+Context in Network Exploration



(a) Bring (step 1) – Selecting a node fades out all graph elements but the node neighborhood.

(b) Bring (step 2) – Neighbor nodes are pulled close to the selected node.

(c) Go – After selecting a neighbor (the green node in Fig. 4(b)), a short animation brings the focus towards a new neighborhood.

[Lambert et al., 2010]

Distortion Concerns

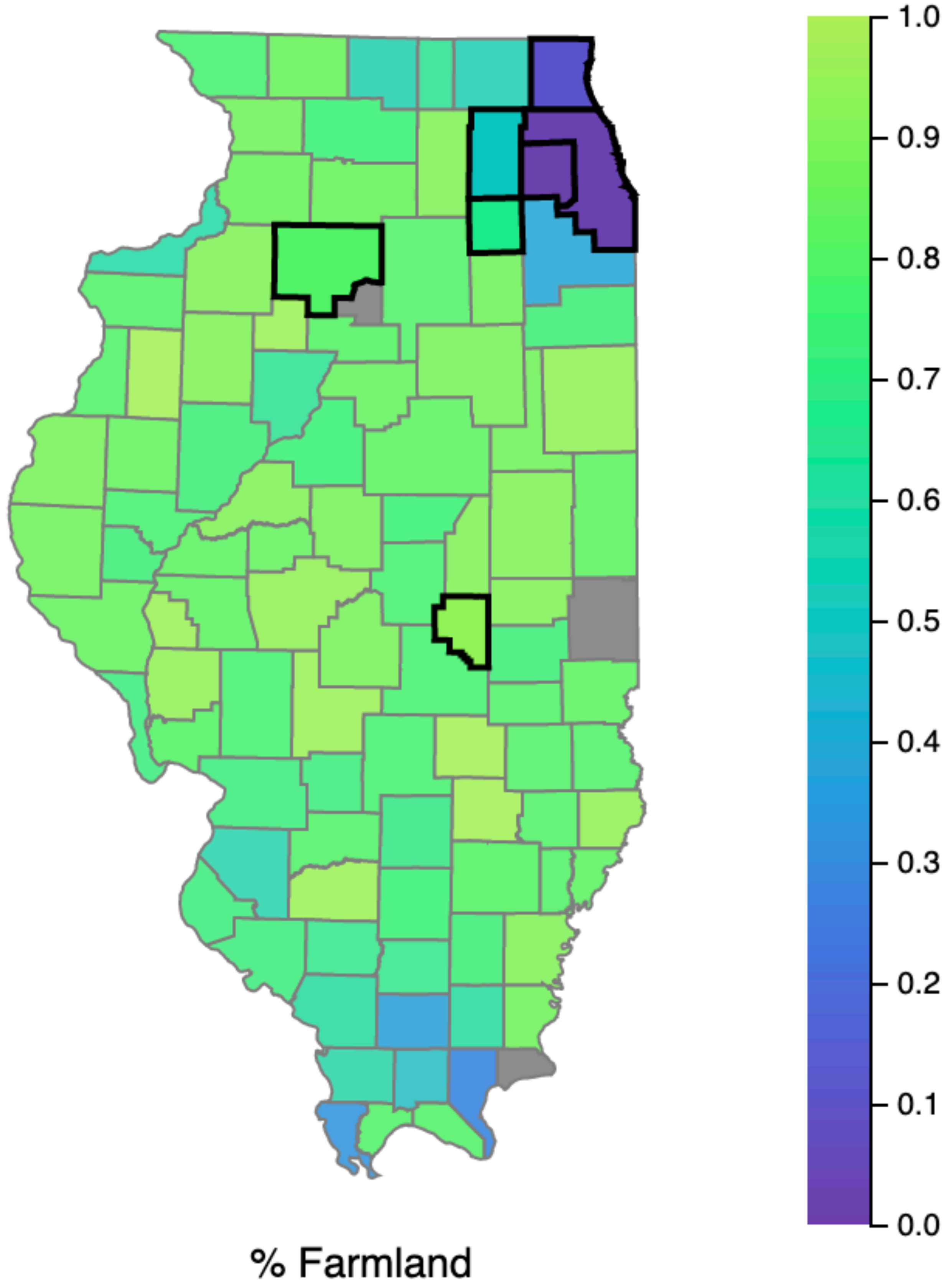
- Distance and length judgments are **harder**
  - Example: Mac OS X Dock with Magnification
  - Spatial position of items changes as the focus changes
- Node-link diagrams not an issue… why?
- Users have to be made aware of distortion
  - Back to scatterplot with distortion example
  - Lenses or shading give clues to users
- **Object constancy**: understanding when two views show the same object
  - What happens under distortion?
  - 3D Perspective is distortion… but we are well-trained for that
- Think about **what** is being shown (filtering) and method (fisheye)

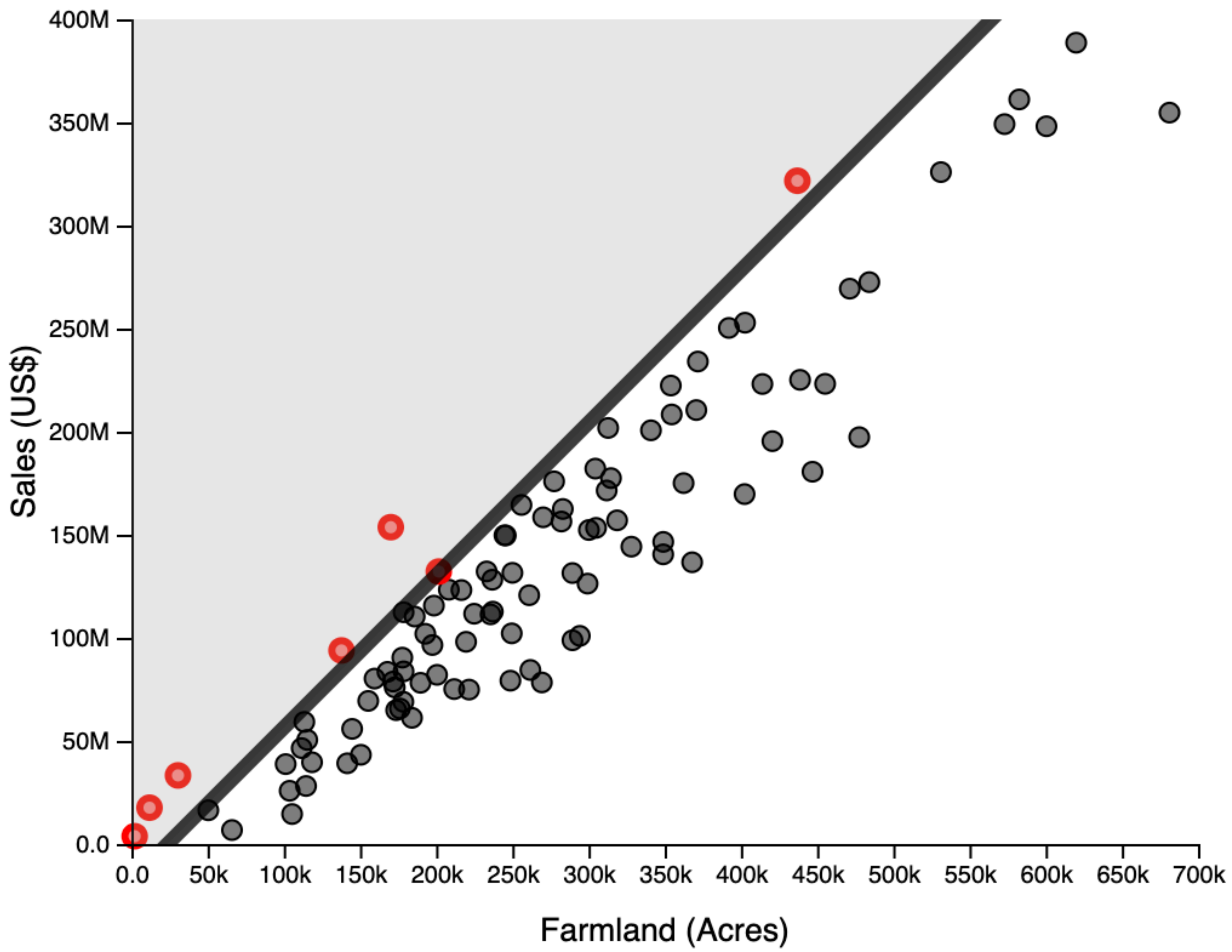
Designs Feedback

- Some good prototypes and focus on interactions
- Generally, would like to see more creativity
  - You can create scatterplots and choropleth maps using Tableau
  - https://xeno.graphics
  - https://www.informationisbeautifulawards.com/showcase
- Justify the use of widgets and/or tooltips
- Provide complete overviews, even if interactions will filter views or provide details
- Be careful with scrolling

Assignment 5

- Multiple Views and Interaction using Linked Highlighting
- Due November 22





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

Hosting data

- github.com
- gist.github.com
- figshare.com
- myjson.com
- Other services

Why JavaScript?

- Python and R have great support for this sort of processing
- Data comes from the Web, want to put visualizations on the Web
- Sometimes unnecessary to download, process, and upload!
- More tools are helping JavaScript become a better language

JavaScript Data Wrangling Resources

- https://observablehq.com/@dakoop/learn-js-data
- Based on http://learnjsdata.com/
- Good coverage of data wrangling using JavaScript

Comma Separated Values (CSV)

- File structure:

```
cities.csv:

city,state,population,land area
seattle,WA,652405,83.9
new york,NY,8405837,302.6
boston,MA,645966,48.3
kansas city,MO,467007,315.0
```

- Loading using D3:

```
d3.csv("/data/cities.csv").then(function(data) {
  console.log(data[0]);
});
```

- Result:

```
=> {city: "seattle", state: "WA", population: 652405, land area: 83.9}
```

- Values are strings! Convert to numbers via the unary `+` operator:
  - `d.population => "652405"`
  - `+d.population => 652405`

[http://learnjsdata.com]

Tab Separated Values (TSV)

- File structure:

```
animals.tsv:

name	type	avg_weight
tiger	mammal	260
hippo	mammal	3400
komodo dragon	reptile	150
```

- Loading using D3:

```
d3.tsv("/data/animals.tsv").then(function(data) {
  console.log(data[0]);
});
```

- Result:

```
=> {name: "tiger", type: "mammal", avg_weight: "260"}
```

- Can also have other delimiters (e.g. '|', ';')

[http://learnjsdata.com]

JavaScript Object Notation (JSON)

- File Structure:

```
employees.json:
[
 {"name":"Andy Hunt",
  "title":"Big Boss",
  "age": 68,
  "bonus": true
 },
 {"name":"Charles Mack",
  "title":"Jr Dev",
  "age":24,
  "bonus": false
 }
]
```

- Loading using D3:

```
d3.json("/data/employees.json".then(function(data) {
  console.log(data[0]);
});
```

- Result:

```
=> {name: "Andy Hunt", title: "Big Boss", age: 68, bonus: true}
```

[http://learnjsdata.com]

Loading Multiple Files

- Use Promise.all to load multiple files and then process them all

```
Promise.all([d3.csv("/data/cities.csv"),
             d3.tsv("/data/animals.tsv")])
  .then(analyze);

function analyze(data) {
  cities = data[0]; animals = data[1];

  console.log(cities[0]);
  console.log(animals[0]);
}
=> {city: "seattle", state: "WA", population: "652405", land area: "83.9"}
{name: "tiger", type: "mammal", avg_weight: "260"}
```

[http://learnjsdata.com]

Combining Data

- Suppose given products and brands
- Brands have an id and products have a brand_id that matches a brand
- Want to join these two datasets together
  - Product.brand_id => Brand.id
- Use a nested forEach/filter
- Use a native join command

[http://learnjsdata.com]

Summarizing Data

- d3 has min, max, and extent functions of the form
  - 1st argument: dataset
  - 2nd argument: accessor function
- Example:

```
var landExtent = d3.extent(data, function(d) { return d.land_area; });
console.log(landExtent);
=> [48.3, 315]
```

- Summary statistics, e.g. mean, median, deviation → same format
- Median Example:

```
var landMed = d3.median(data, function(d) { return d.land_area; });
console.log(landMed);
=> 193.25
```

[http://learnjsdata.com]

Nesting Data

- Take a flat structure and turn it into something nested
- Often similar to a groupby in databases
- `key` indicate groupings
- `rollup` indicates how the groups are processed/aggregated
- Last function specifies the data and how the output should look
  - `entries: [{key: <key>, value: <value>}]`
  - `object: {<key>: <value>, …}`
  - `map: {<key>: <value>, …}` but as a `d3.map` (safer than object, but uses get/set instead of square brackets (`[]`)

[http://learnjsdata.com]

Nesting Example

- 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.nest:

```
var expensesAvgAmount = d3.nest()
  .key(function(d) { return d.name; })
  .rollup(function(v) { return d3.mean(v, function(d) { return d.amount; }); })
  .entries(expenses);
console.log(JSON.stringify(expensesAvgAmount));
```

- Result:

```
=> [{"key":"jim","values":39.5},
    {"key":"carl","values":120.11},
    {"key":"stacy","values":30.3}]
```

[http://learnjsdata.com]

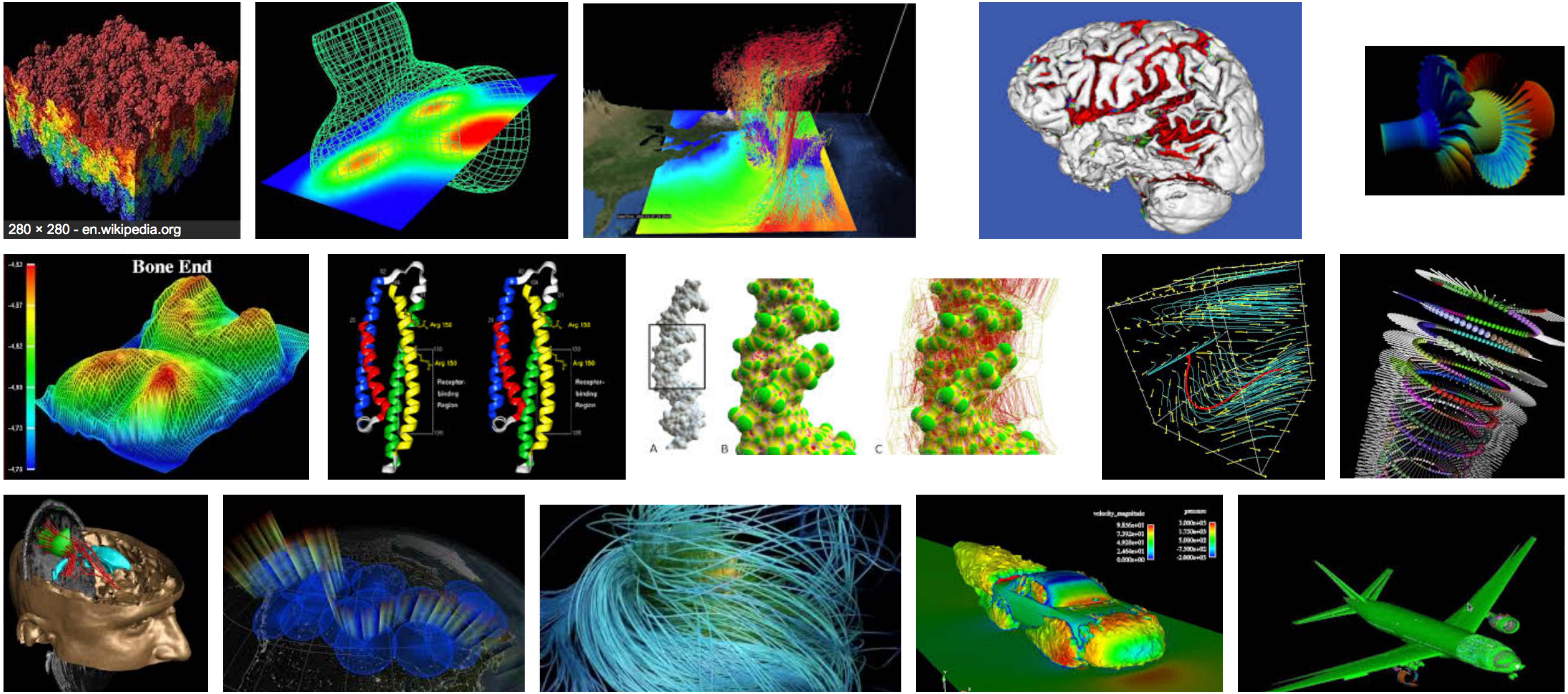
d3-array 2.0 Updates

- https://observablehq.com/@d3/d3-array-2-0
- Works with iterables
- group and rollup are separate now
- https://observablehq.com/@d3/d3-group

Scivis and Infovis

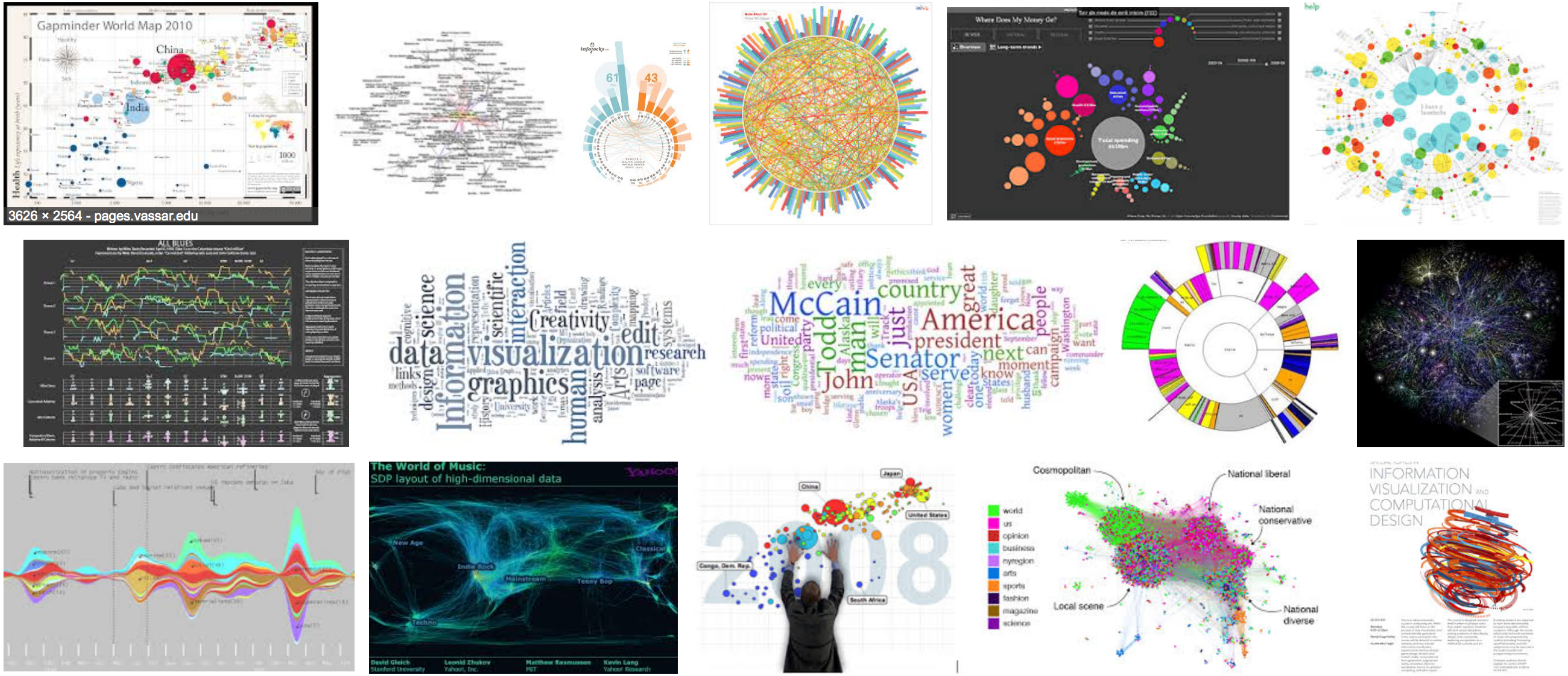
- Two subfields of visualization
- **Scivis** deals with data where the spatial position is given with data
  - Usually continuous data
  - Often displaying physical phenomena
  - Techniques like isosurfacing, volume rendering, vector field vis
- In **Infovis**, the data has no set spatial representation,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

SciVis



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

InfoVis



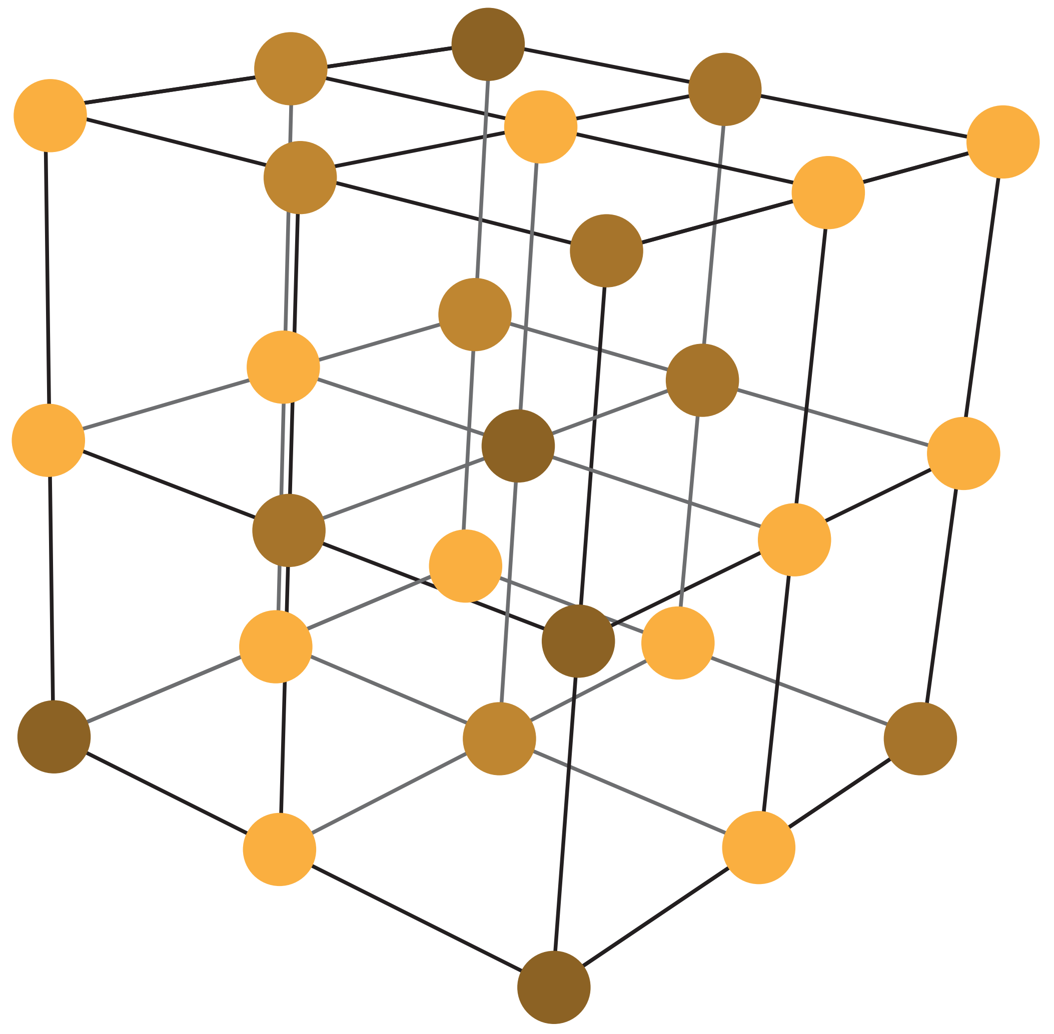
[Google Image Search for "information visualization", 2017]

Fields

Tables	Networks & Trees	Fields	Geometry	Clusters, Sets, Lists
Items	Items (nodes)	Grids	Items	Items
Attributes	Links	Positions	Positions	
	Attributes	Attributes		

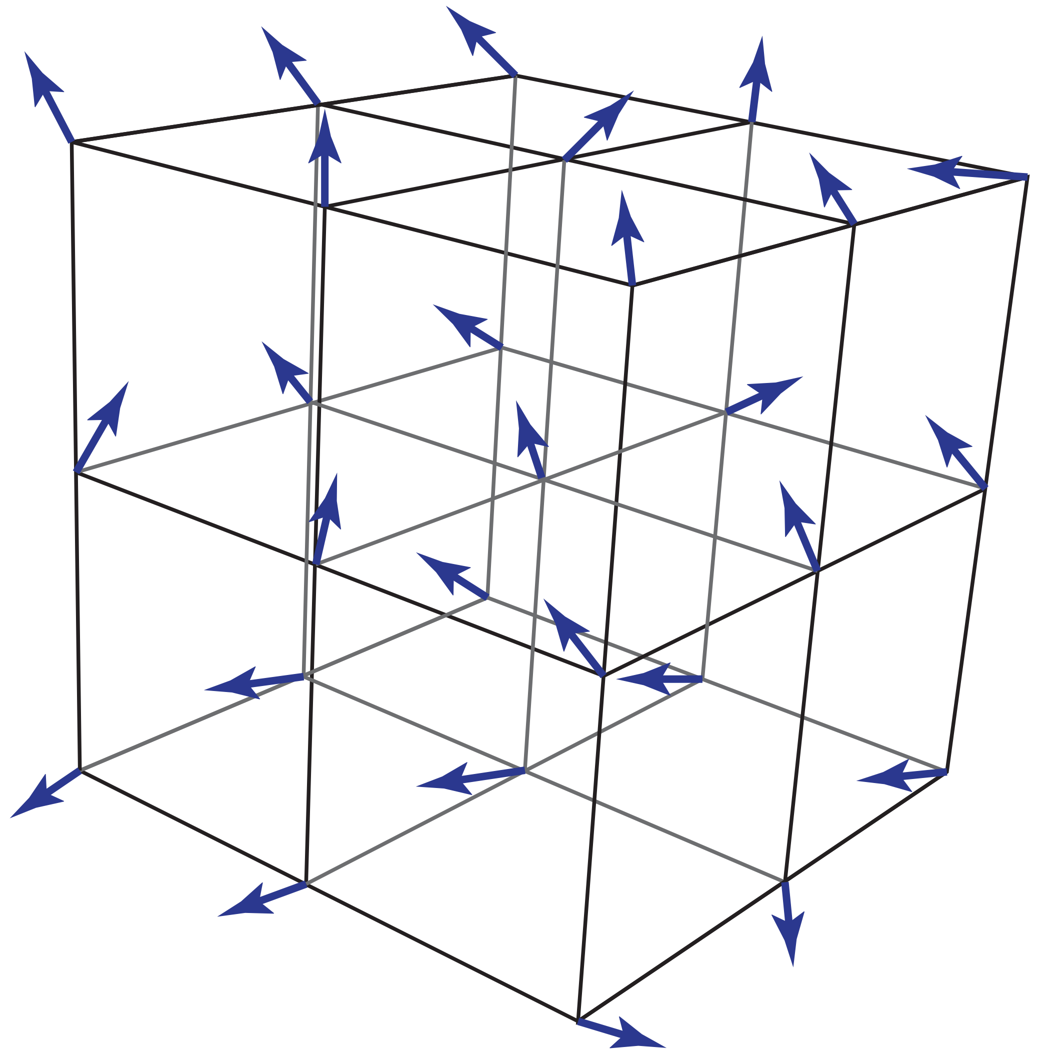
- Values come from a **continuous** domain, infinitely many values
- **Sampled** at certain positions to approximate the entire domain
- Positions are often aligned in **grids**
- Often measurements of natural or simulated phenomena
- Examples: temperature, wind speed, tissue density, pressure, speed,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

Fields in Visualization



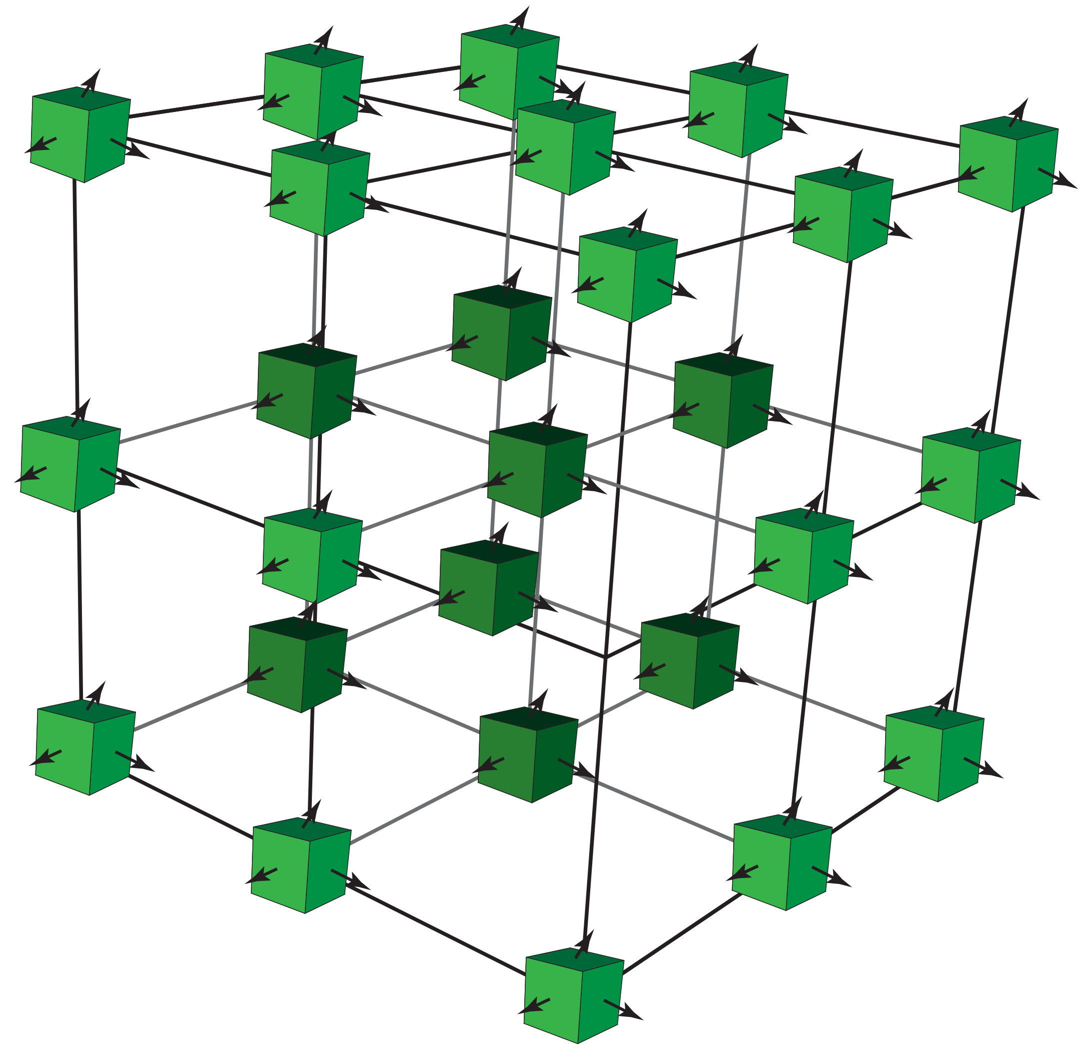
Scalar Fields

(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

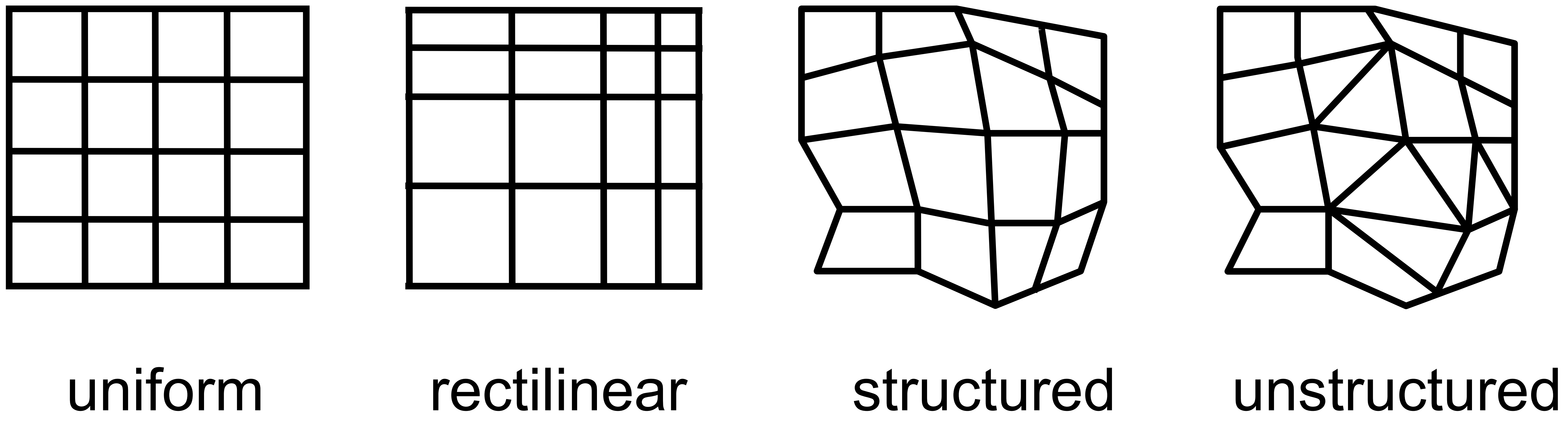
Grids

- 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

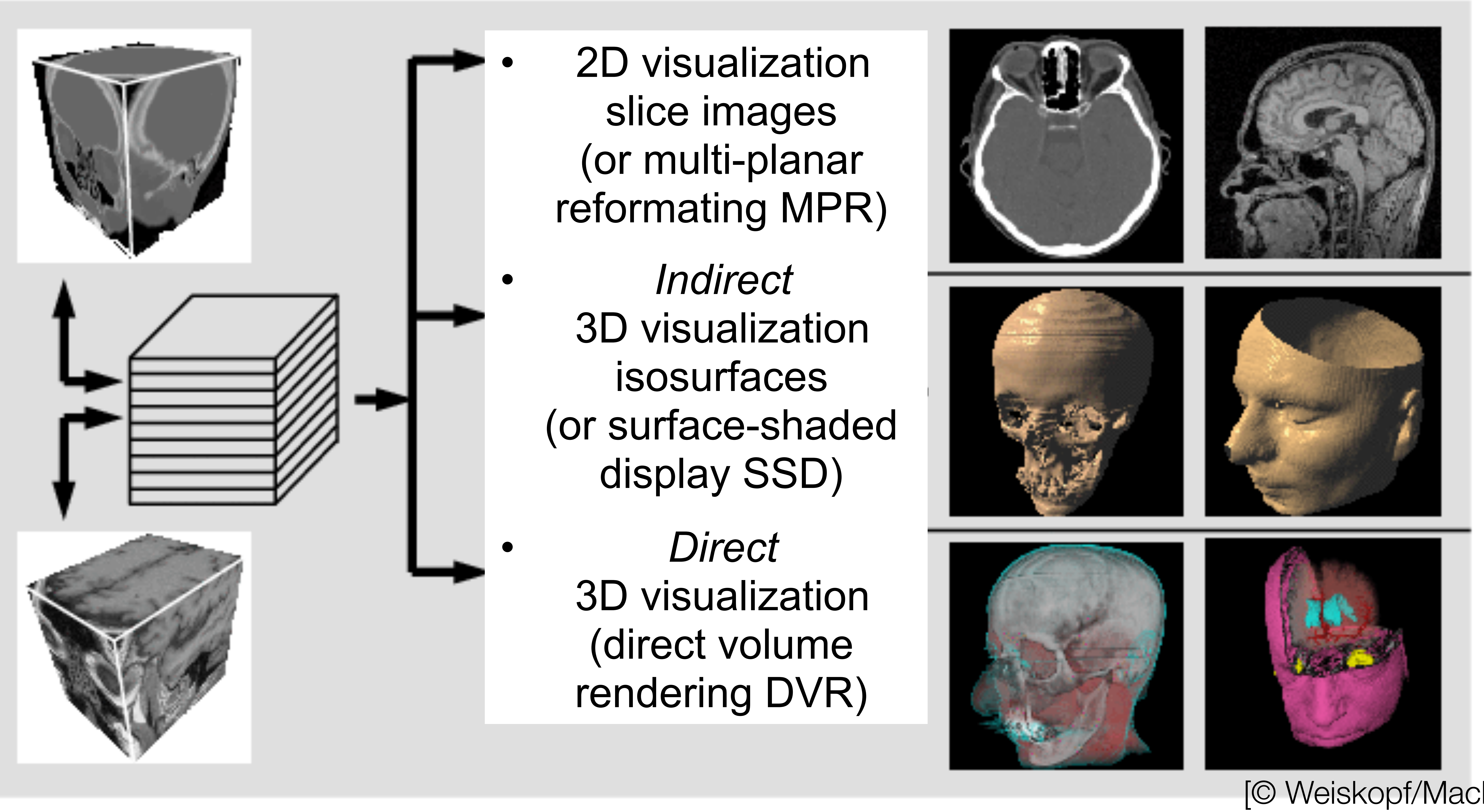
- Remember we have continuous data and want to sample it in order to understand the **entire** domain
- Possible schemes?



[© 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

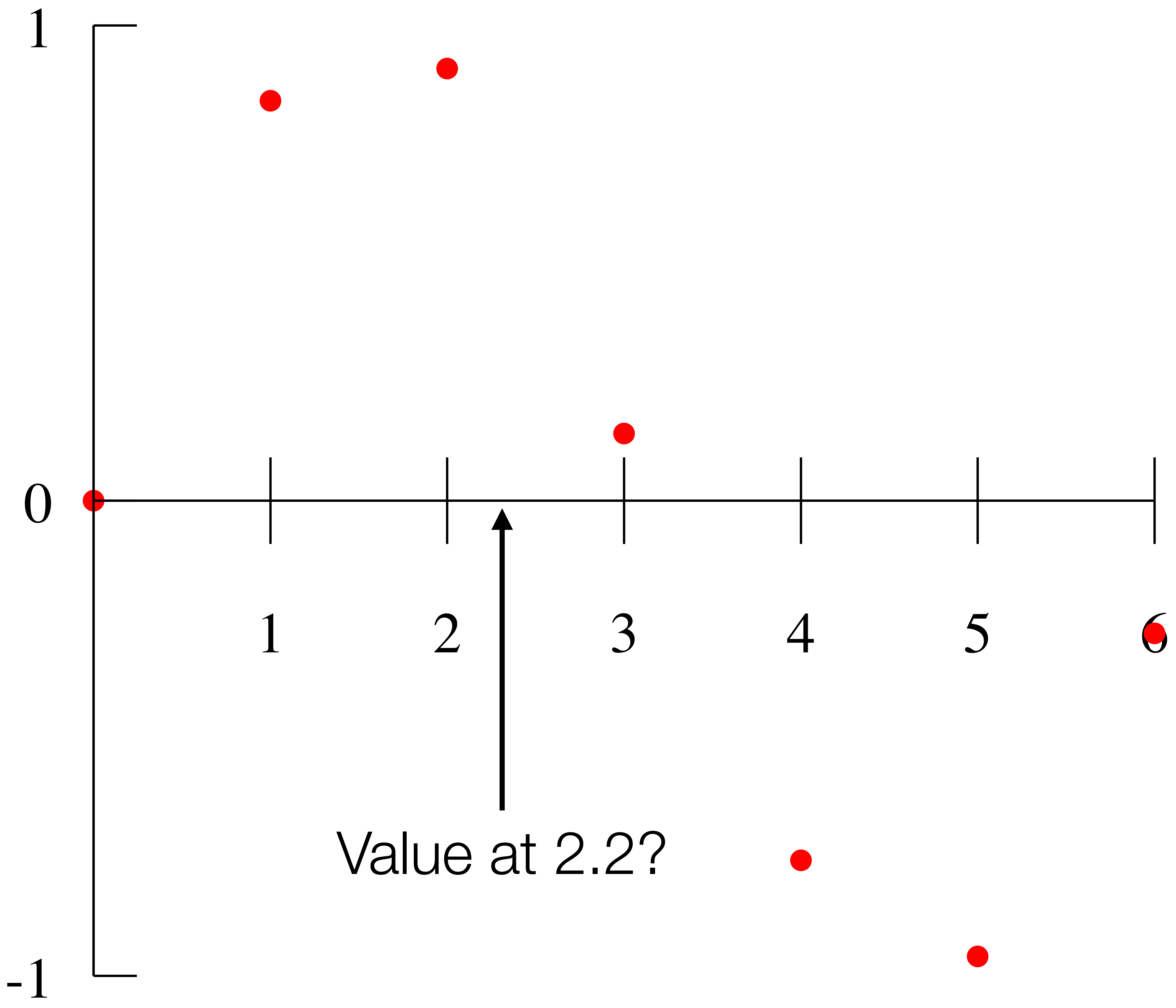


[© Weiskopf/Machiraju/Möller]

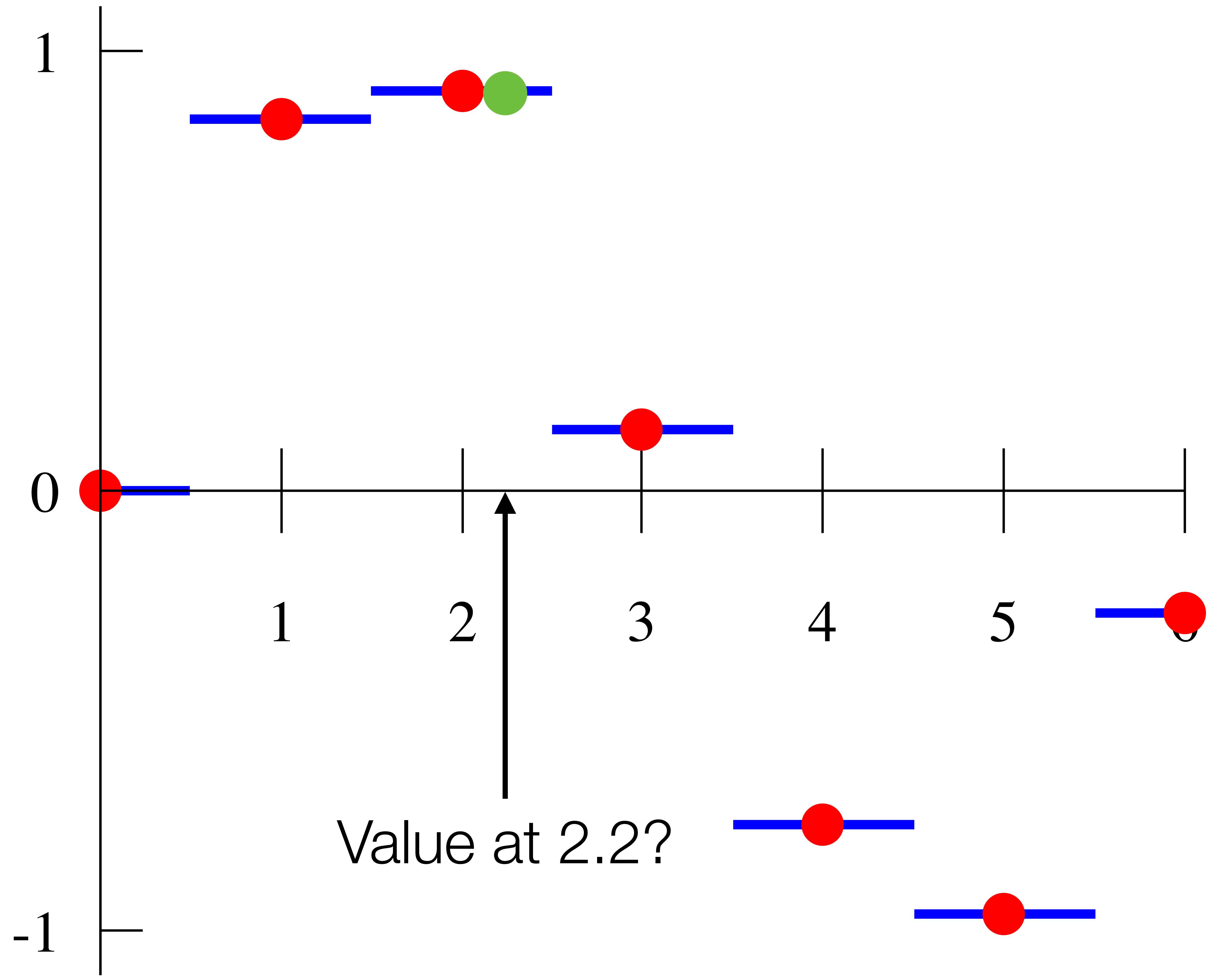
Data

- 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 schemes

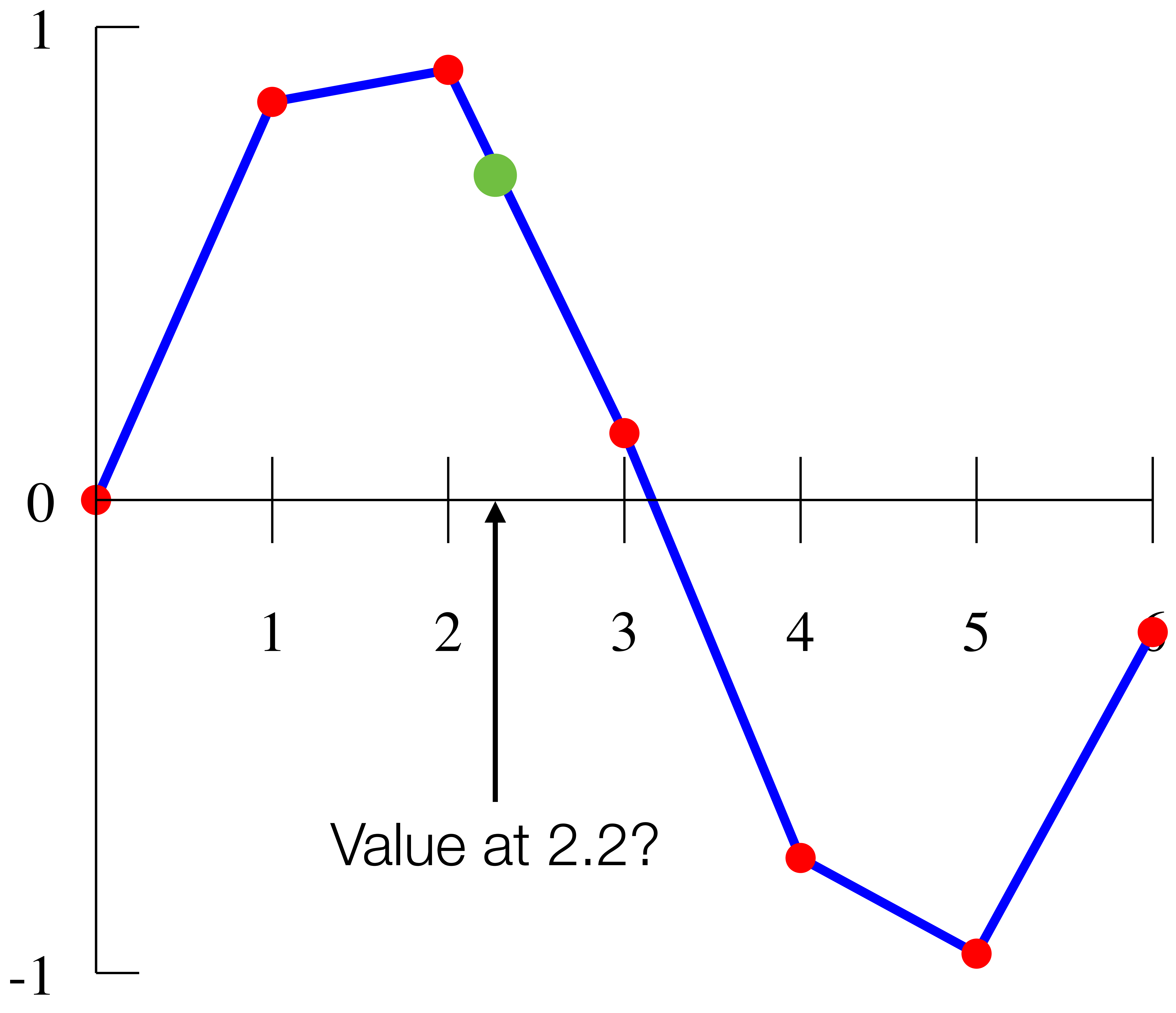
Interpolation



Nearest Neighbor Interpolation



Linear Interpolation



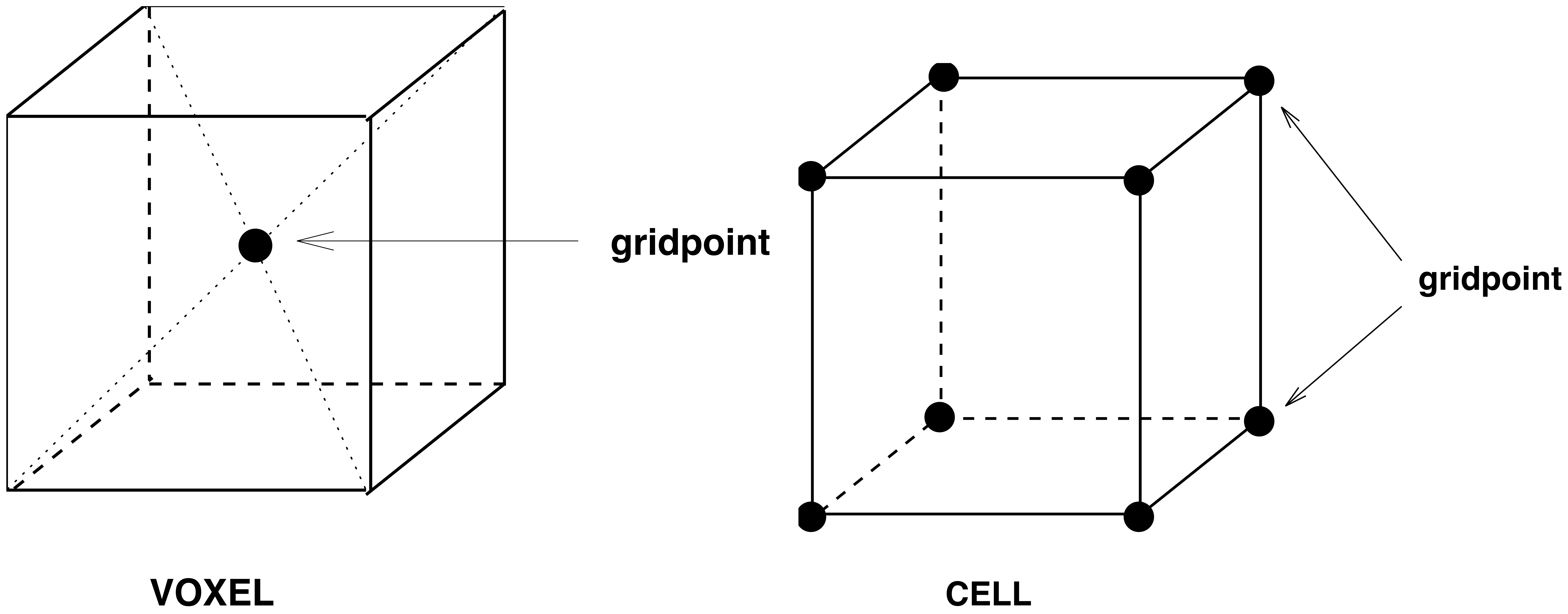
Interpolation

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

Dimensions of Data

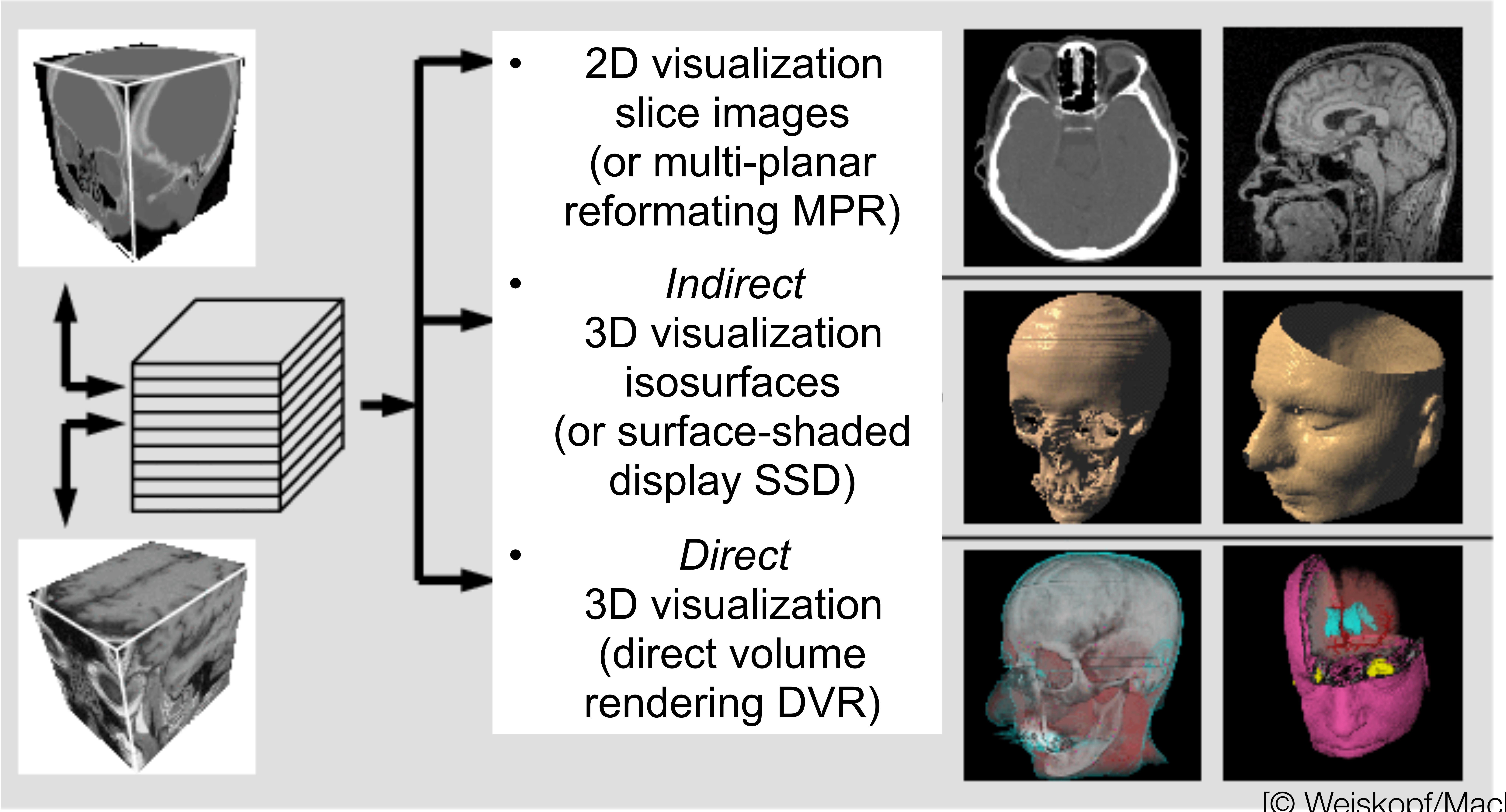
- 1-Dimension: data along a line
  - Example: temperature along my drive from Tucson to Dartmouth
- 2-Dimensional: data on a plane
  - Example: temperature on the surface of a pond
- 3-Dimensional: data in our normal world (data in a **volume**)
  - Example: temperature at every point in the room
- Complexity 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



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

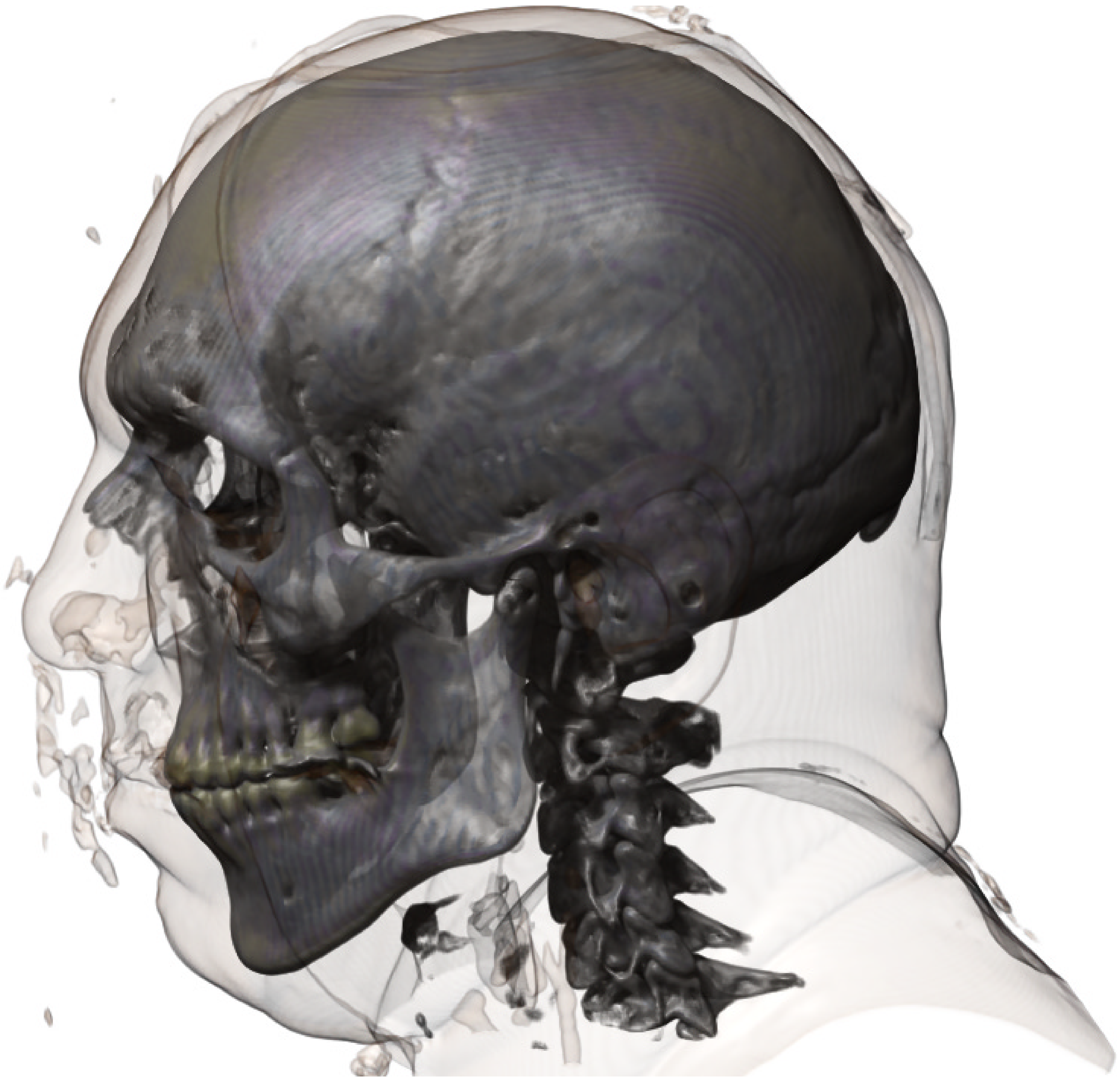
Visualizing Volume (3D) Data



[© Weiskopf/Machiraju/Möller]

Visualizing Volume (3D) Data

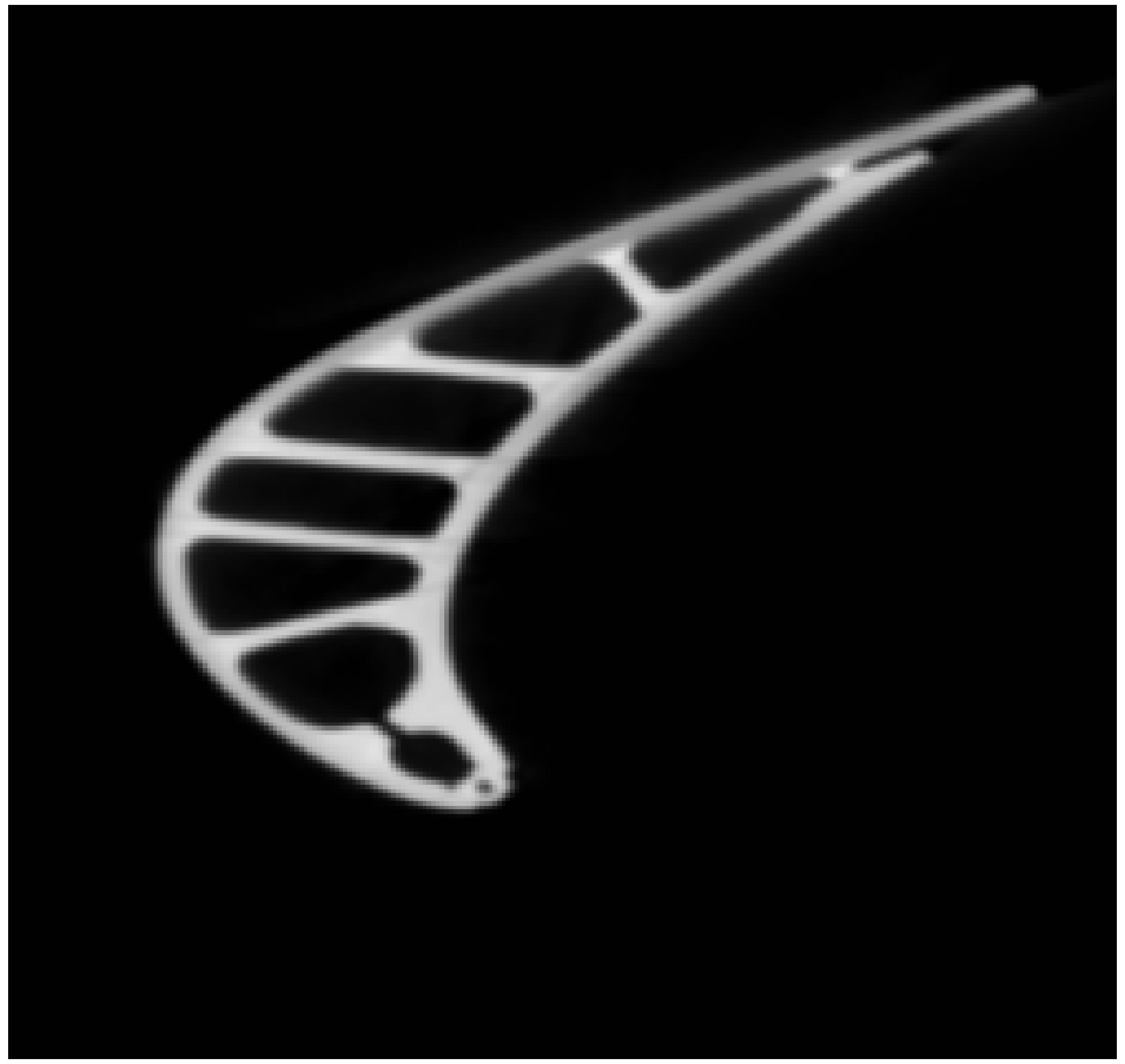
(a) 2D slice



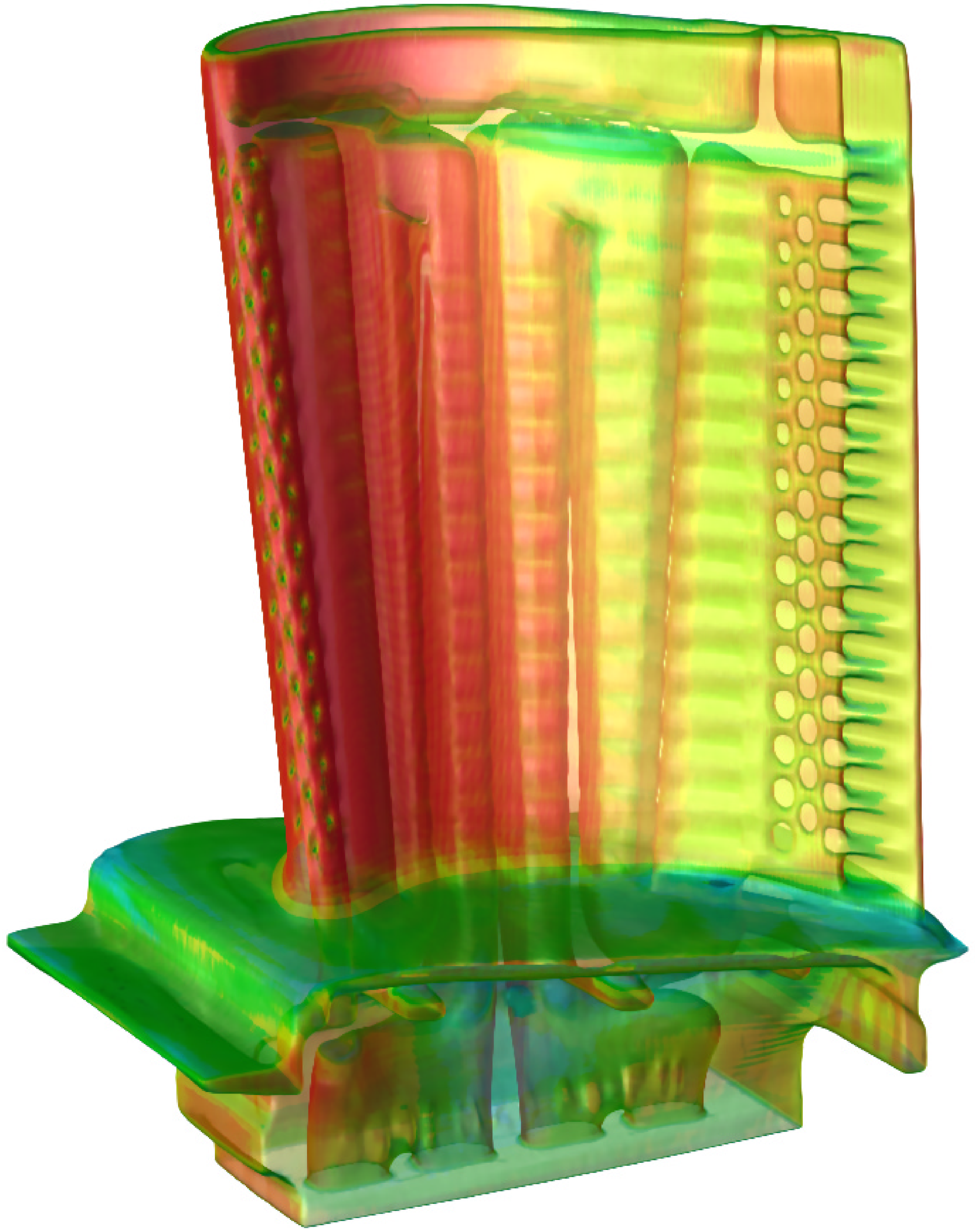
(b) Volume Rendering

[J. Kniss, 2002]

Visualizing Volume (3D) Data



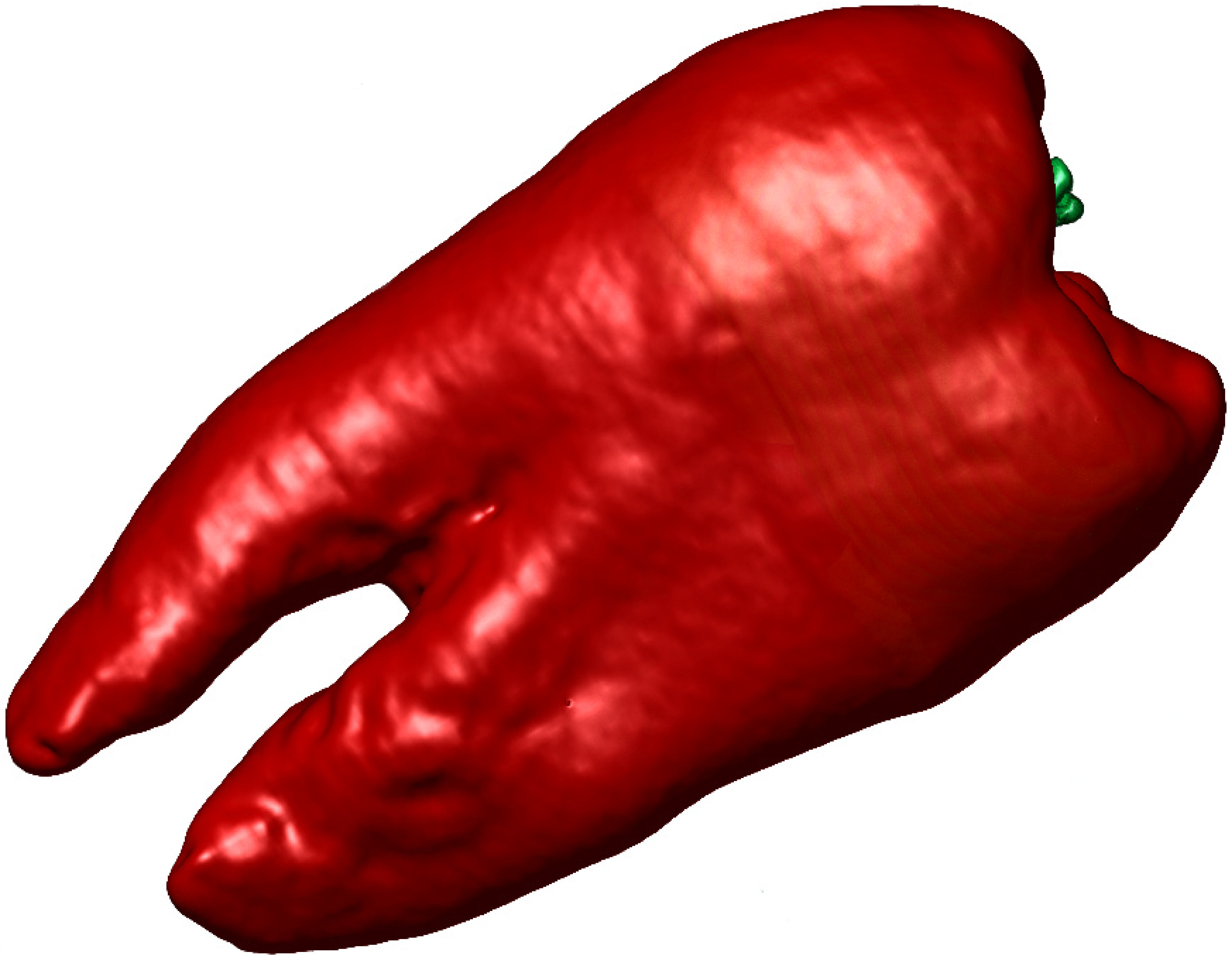
(a) 2D slice



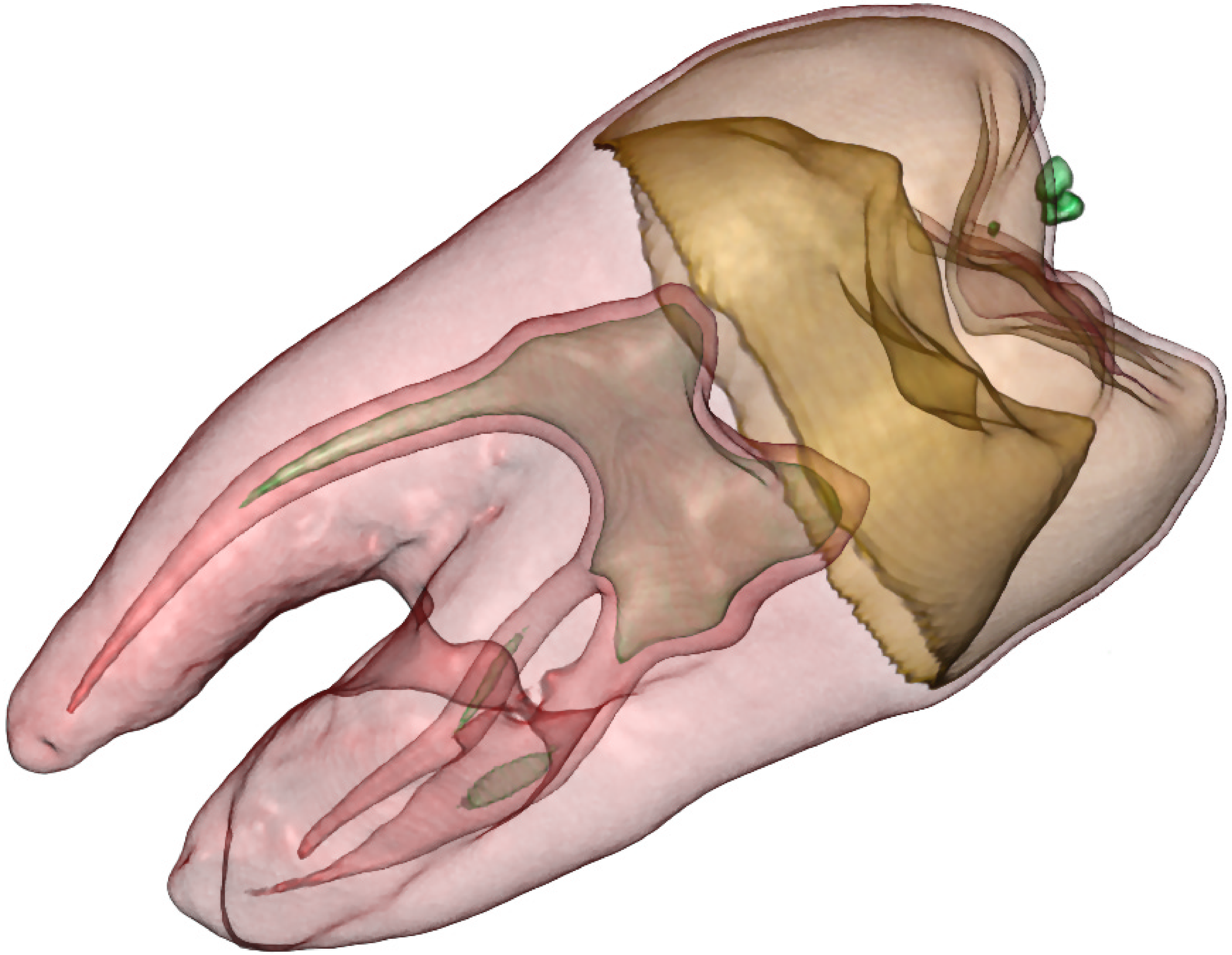
(b) Volume Rendering

[J. Kniss, 2002]

Visualizing Volume (3D) Data



(a) An isosurfaced tooth.



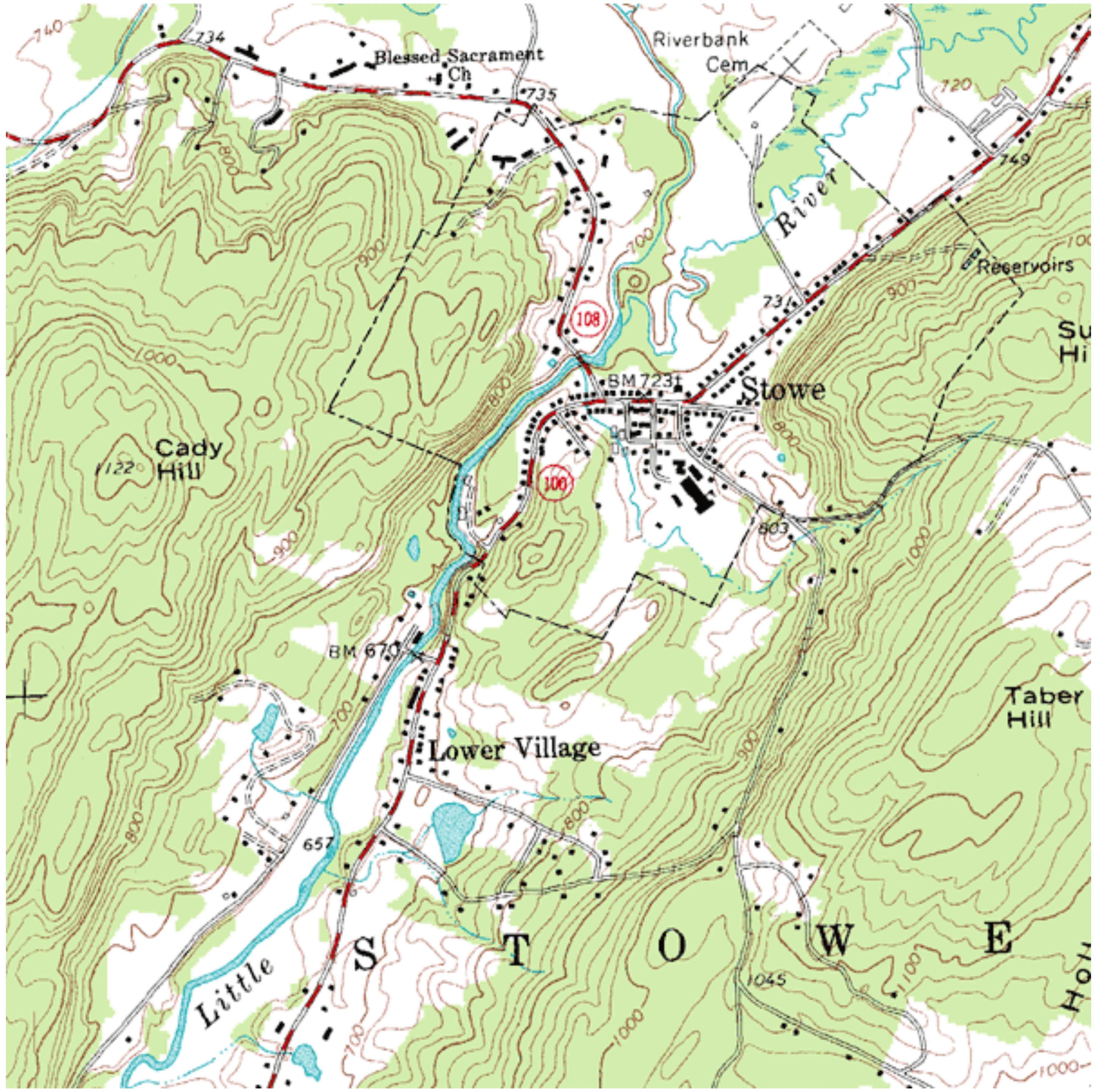
(b) Multiple isosurfaces.

[J. Kniss, 2002]

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

Isolines (2D)

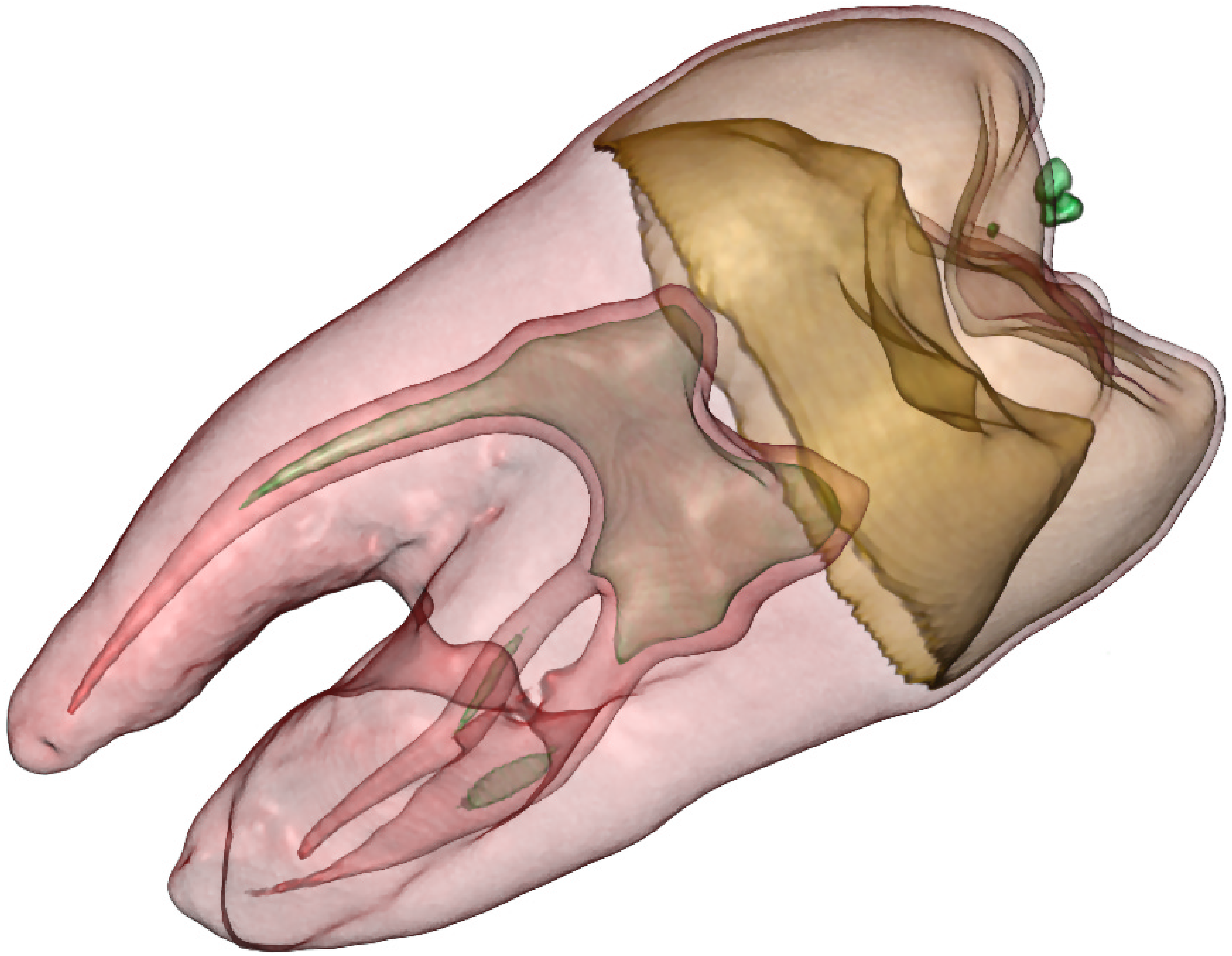
- Isoline: a line that has the same scalar value at all locations
- Example: Topographical Map



[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]