

Information Visualization

W09: Visualization Pipeline

Graduation School of System Informatics

Department of Computational Science

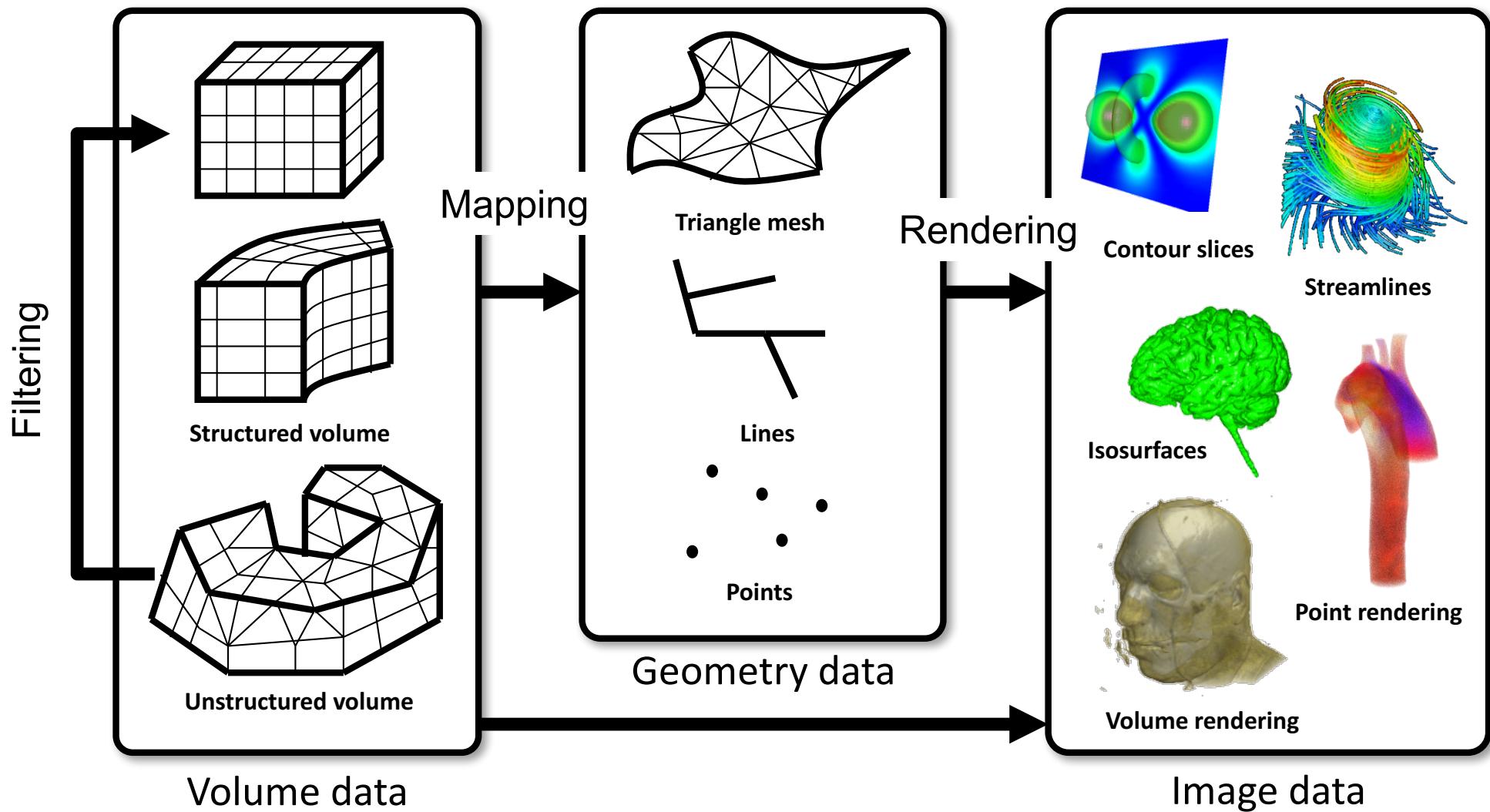
Naohisa Sakamoto, Akira Kageyama

May.16, 2017

Schedule

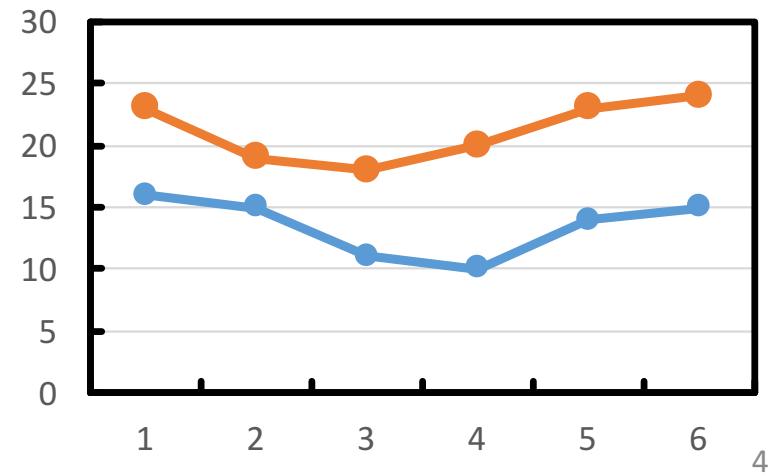
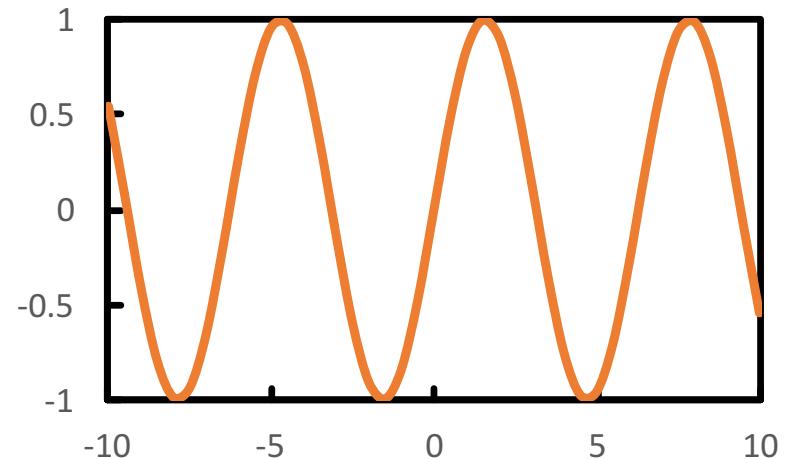
- W01 4/11 Guidance
- W02 4/12 Setup
- W03 4/18 Introduction to Data Visualization
- W04 4/19 CG Programming
- W05 4/25 Rendering Pipeline
- W06 4/26 Coordinate Systems and Transformations
- W07 5/09 Shading
- W08 5/10 Shader Programming
- W09 5/16 Visualization Pipeline
- W10 5/17 Data Model and Transfer Function
- W11 5/23 Scalar Data Visualization 1 (Isosurface Extraction)
- W12 5/24 Implementation of Isosurface Extraction
- W13 5/30 Scalar Data Visualization 2 (Volume Rendering)
- W14 5/31 Implementation of Volume Rendering
- W15 6/06 Student Presentations

Scientific Visualization



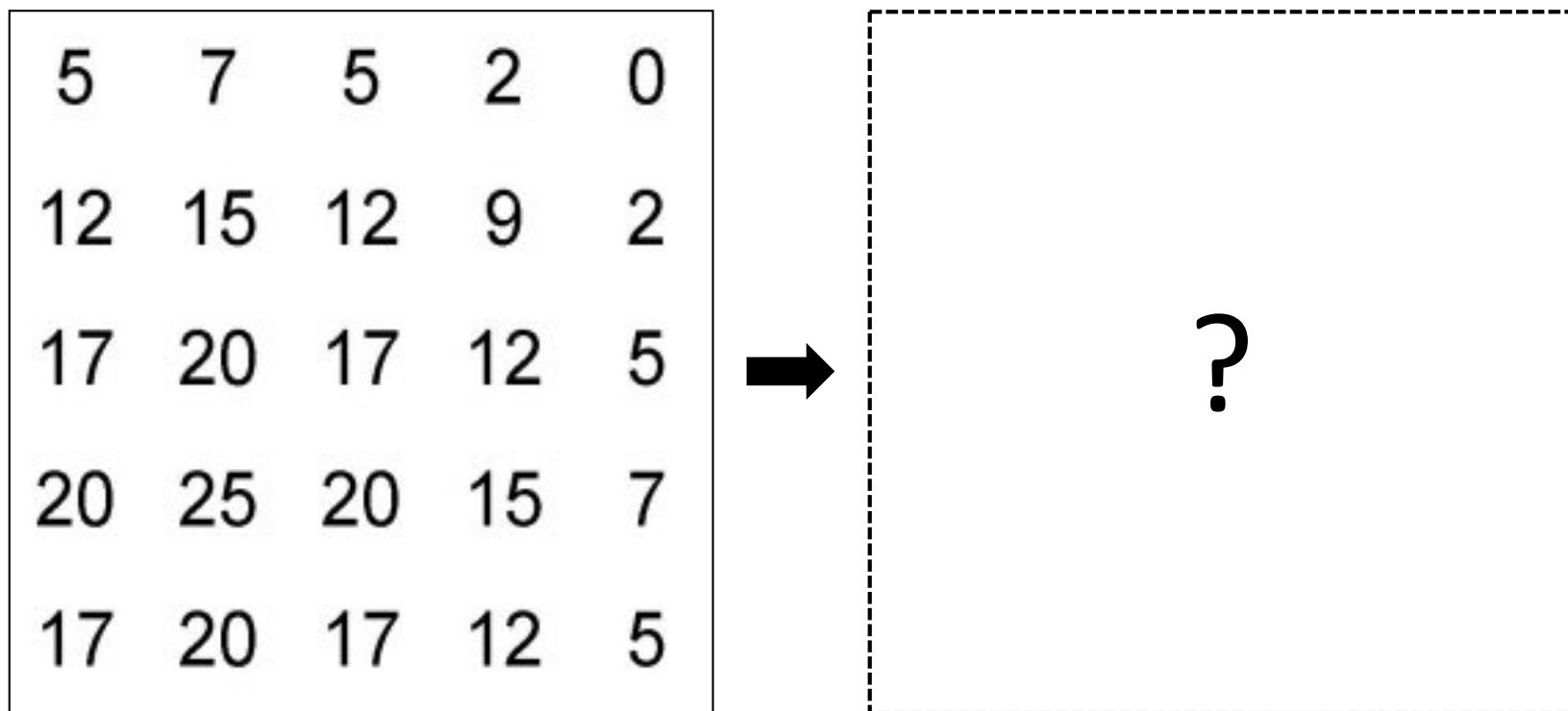
Data Visualization in 1D

- Function $f(x)$
 - $f(x) = \sin x$
- Measurement data
 - 23, 19, 18, 20, 23, 24
 - 16, 15, 11, 10, 14, 15



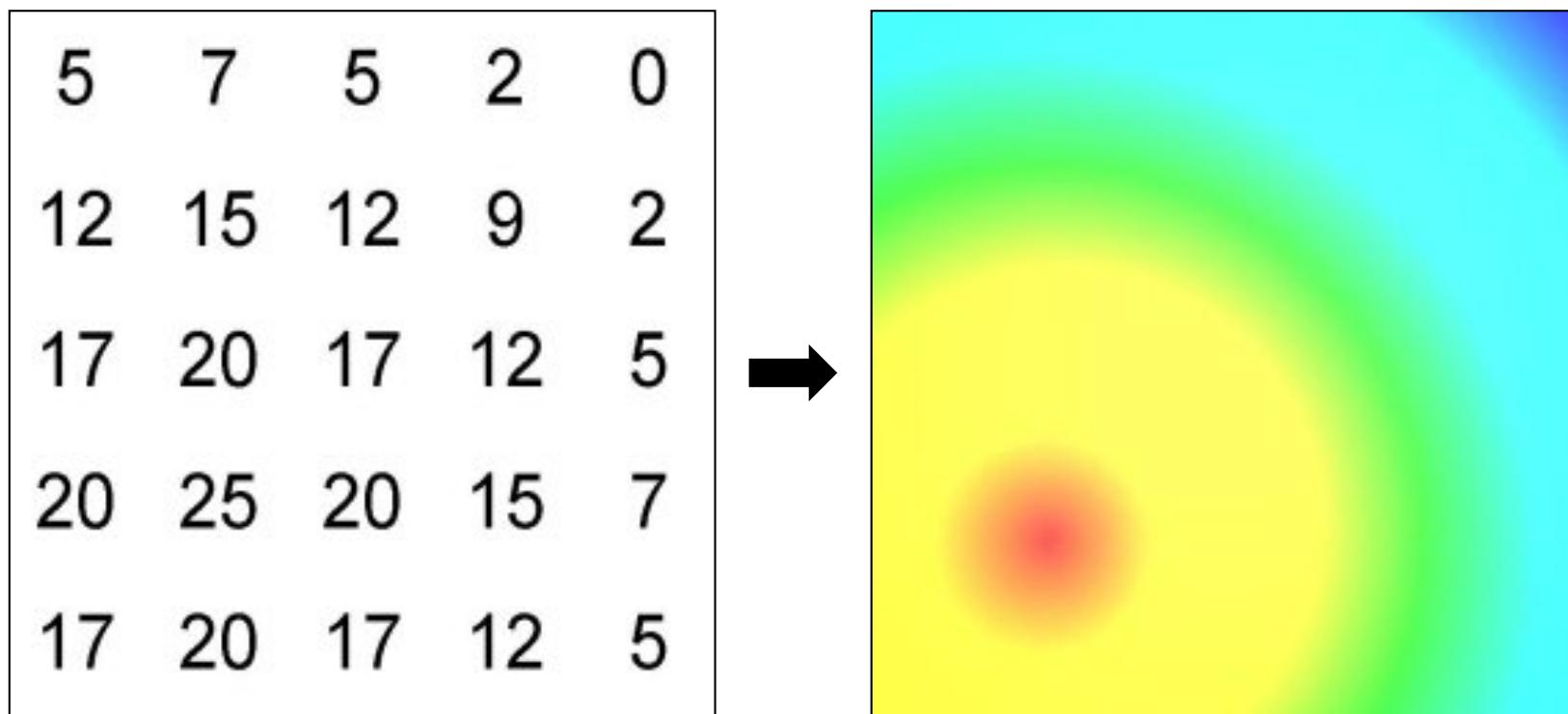
Data Visualization in 2D

- Function $f(x,y)$
- Numerical data



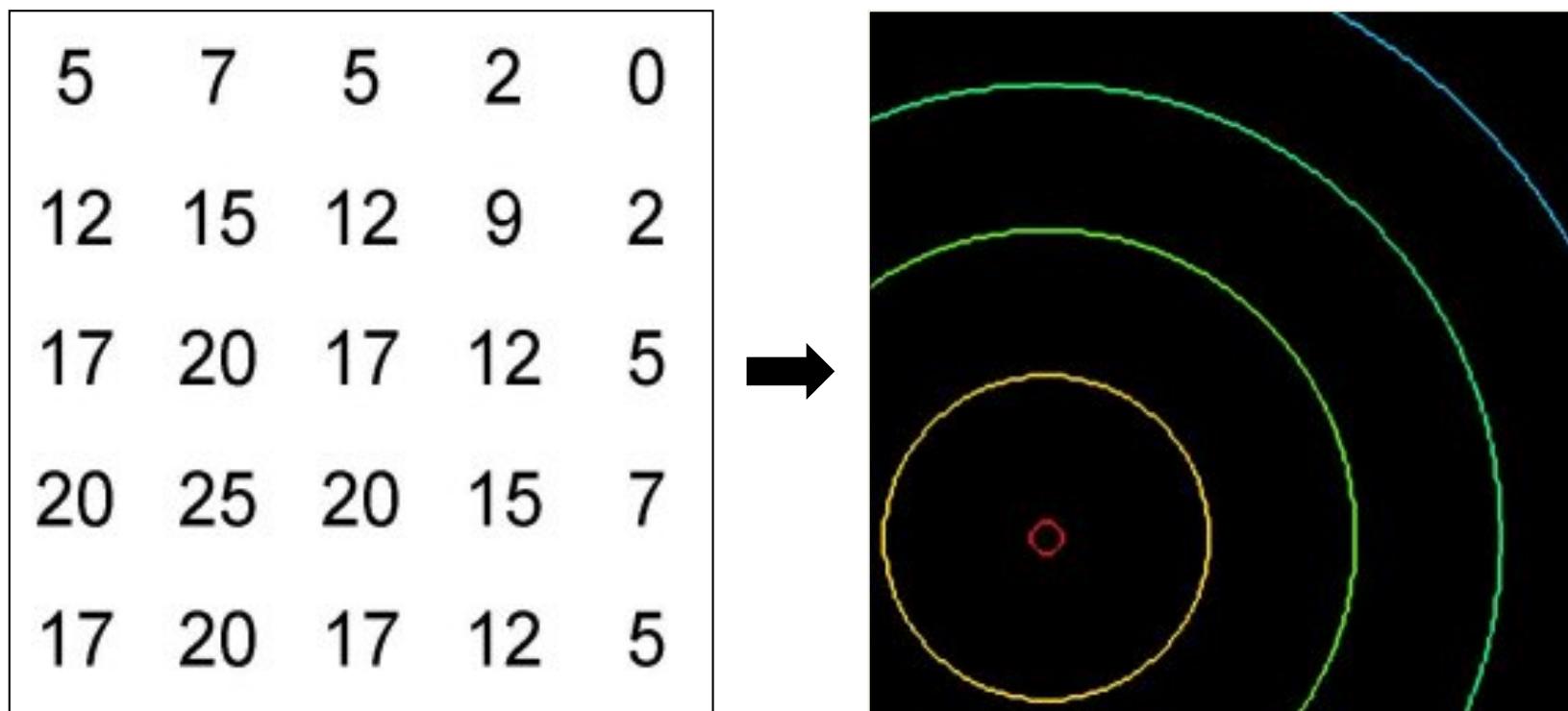
Data Visualization in 2D

- Function $f(x,y)$
- Numerical data



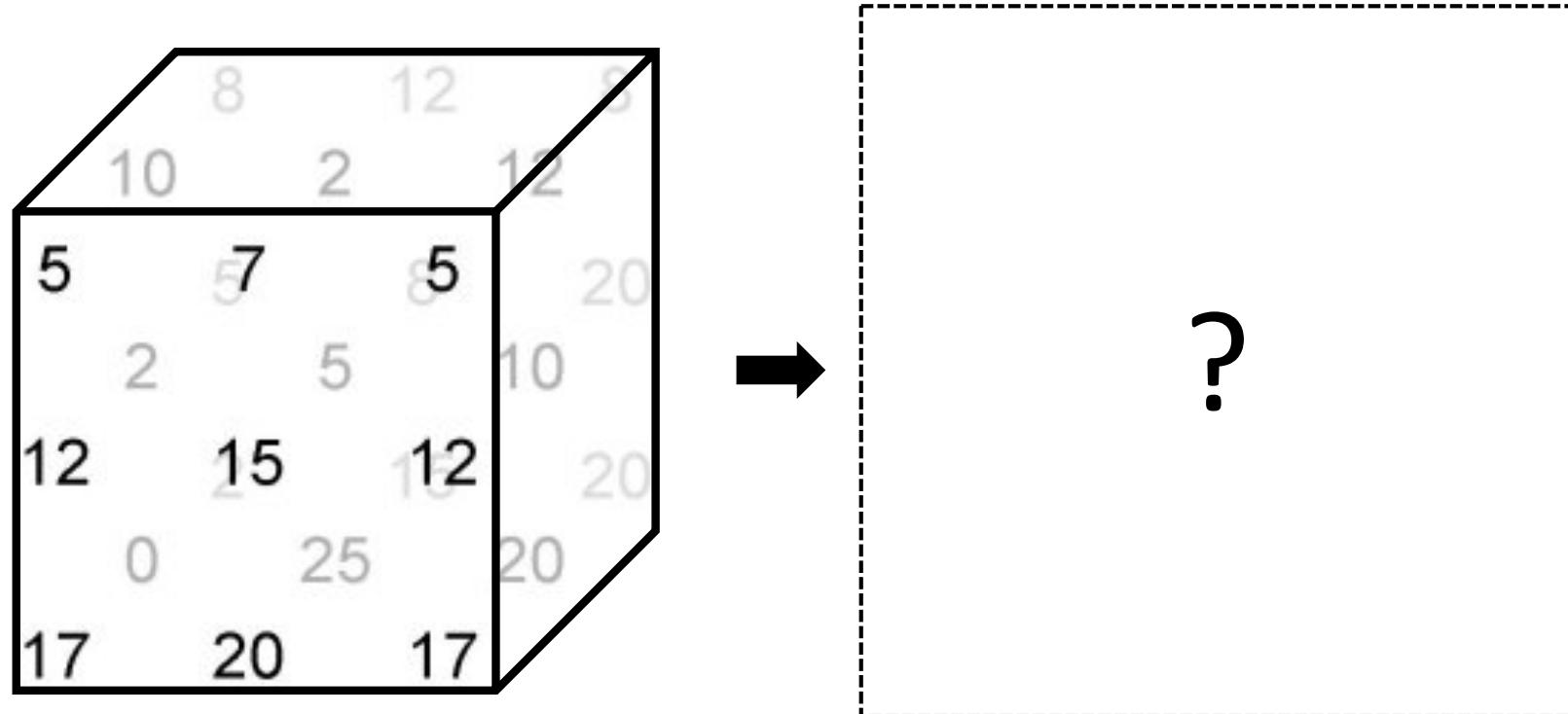
Data Visualization in 2D

- Function $f(x,y)$
- Numerical data



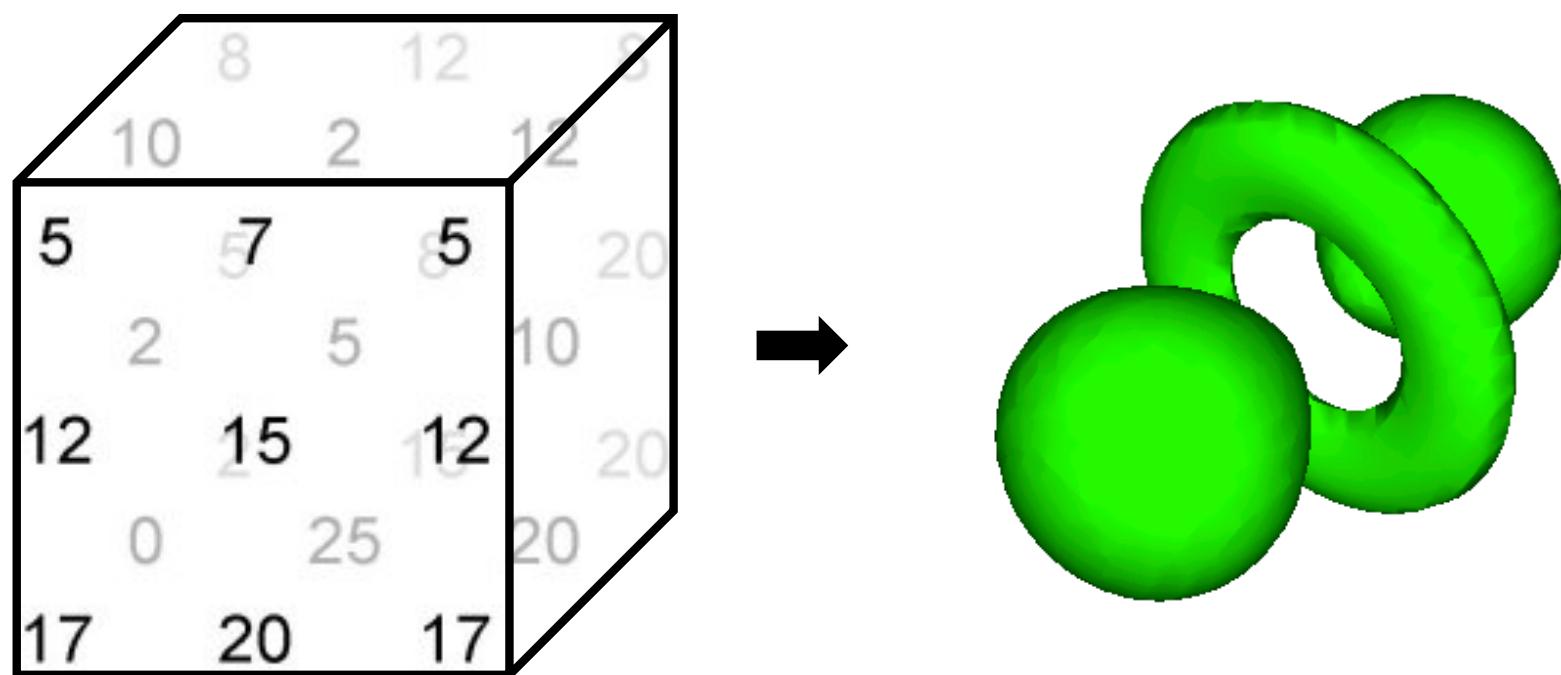
Data Visualization in 3D

- Function $f(x,y,z)$
- Numerical data



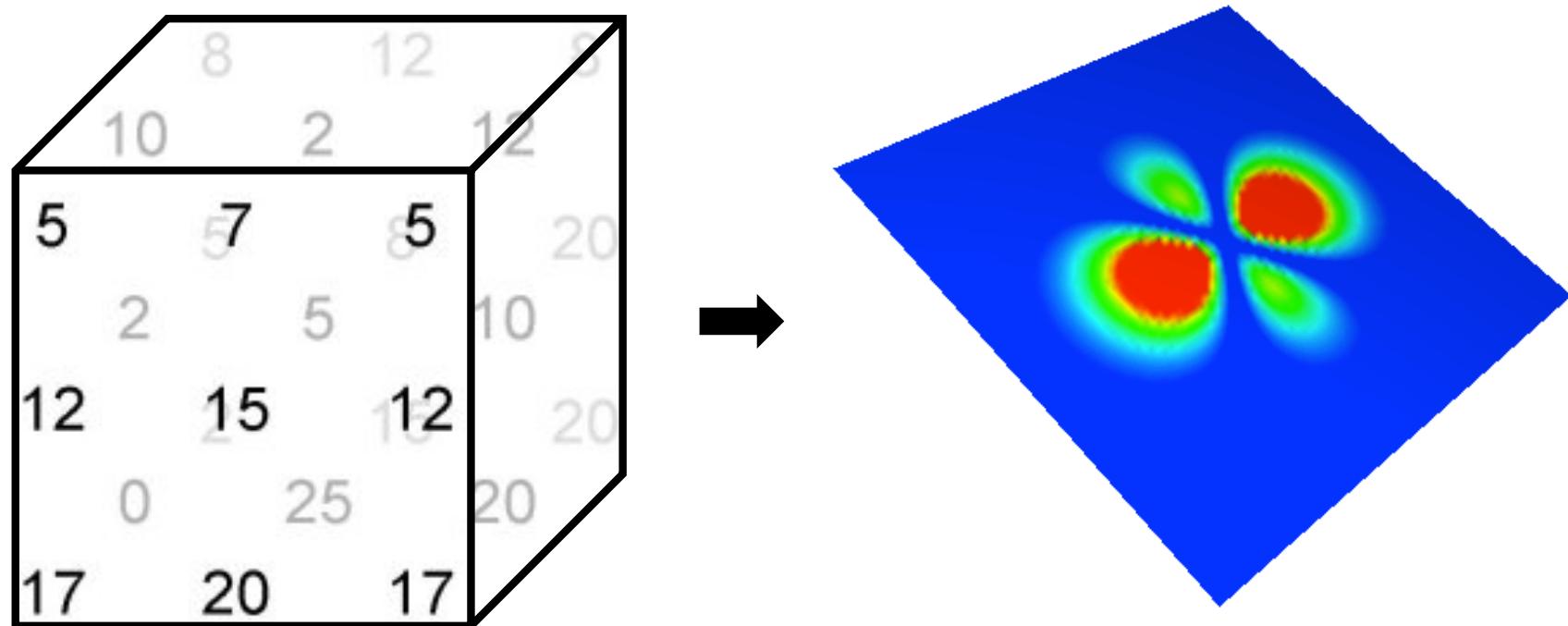
Data Visualization in 3D

- Isosurfaces
 - Marching Cubes



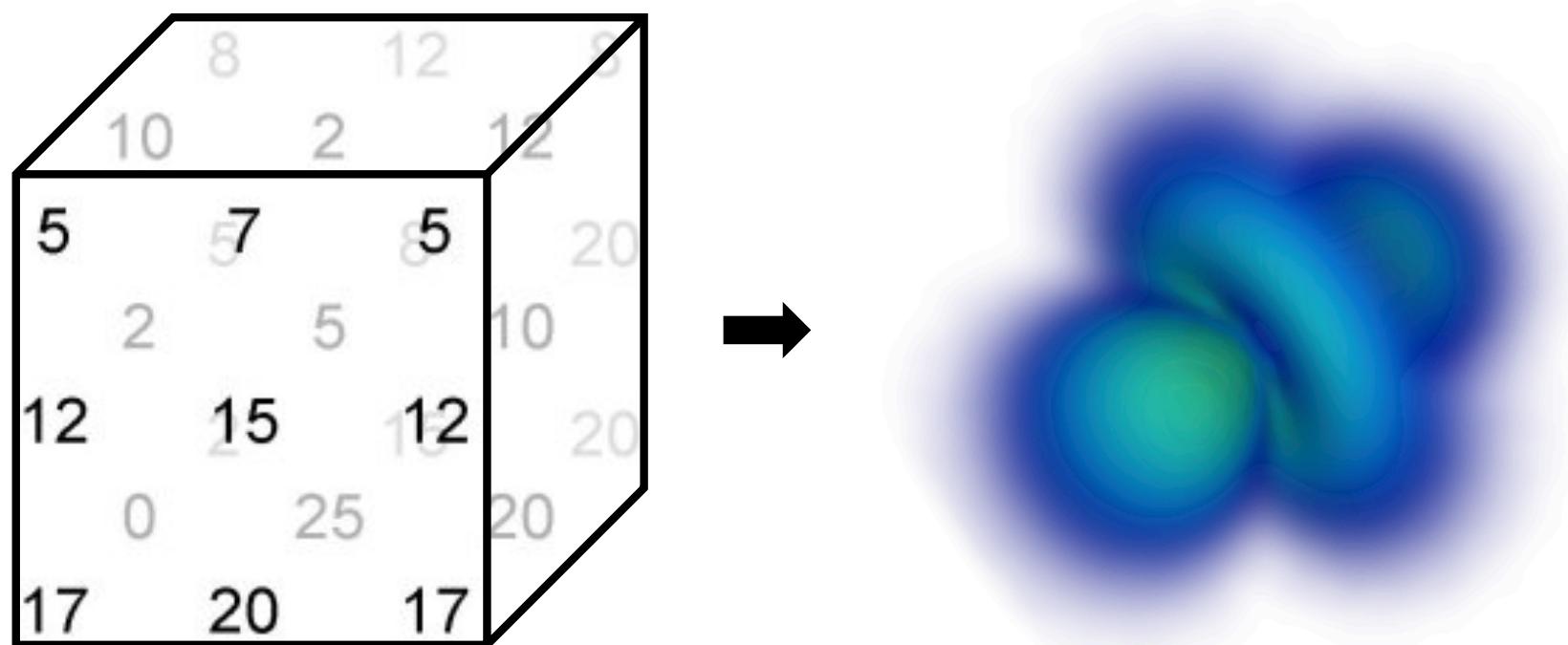
Data Visualization in 3D

- Slice Plane
 - Slice Plane extracted with Isosurfacing algorithm



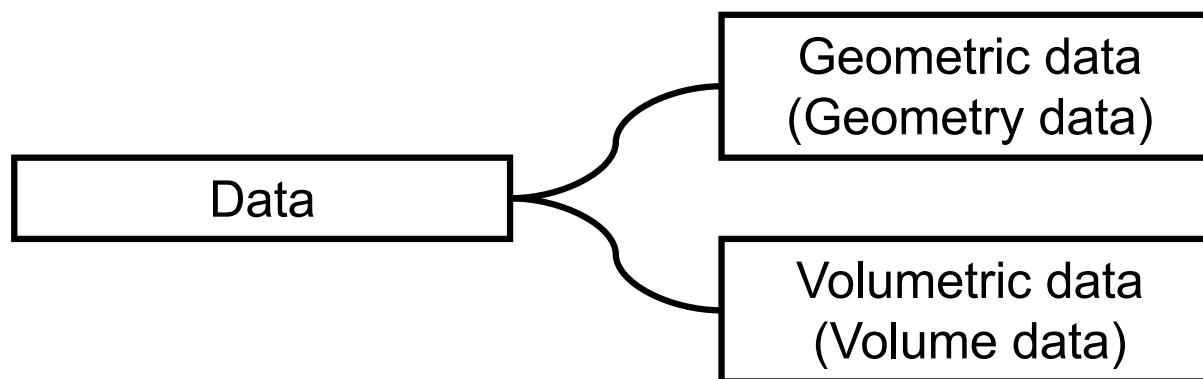
Data Visualization in 3D

- Direct Volume Rendering
 - Ray Casting



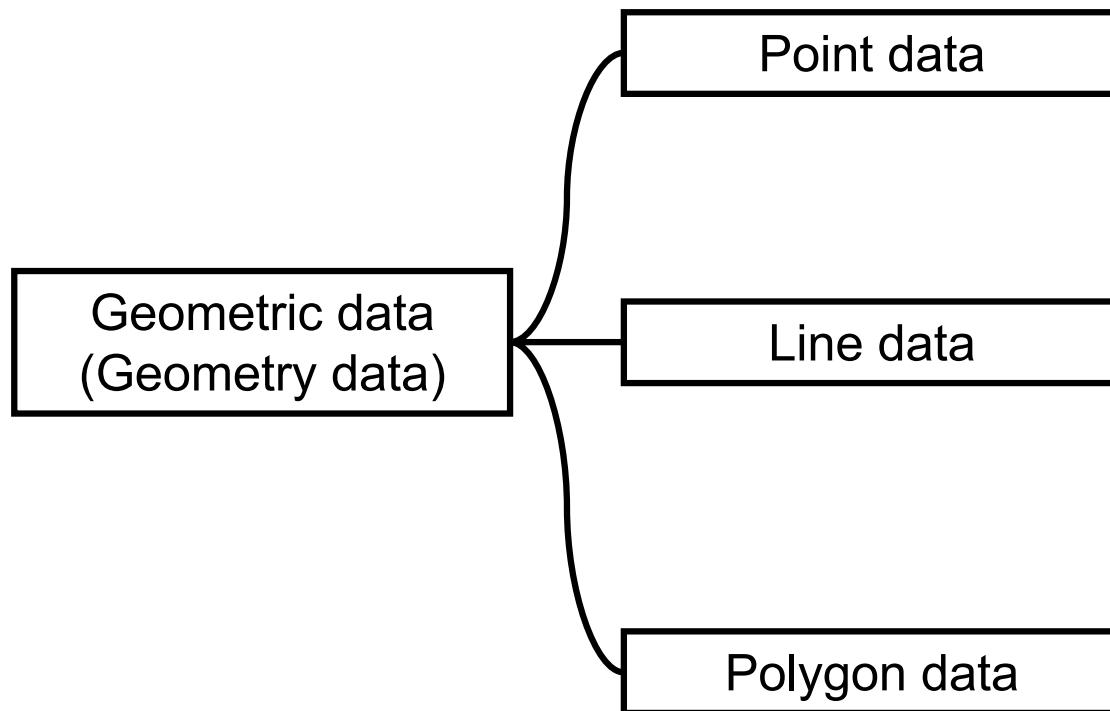
Data

- Data for visualization are classified into geometric data and volumetric data.



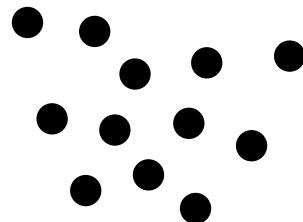
Geometric Data

- A geometric data is a data that is either a point data, a line data or a polygon data.



Point Data

- A point data is an object that is represented as a set of points, with coordinates and colors for each vertex.



```
var coords = [  
    [x0, y0, z0],  
    [x1, y1, z1],  
    [x2, y2, z2]  
];
```

```
var colors = [  
    [r0, g0, b0],  
    [r1, g1, b1],  
    [r2, g2, b2]  
];
```

Memory Layout

- Structure of Array (SoA)

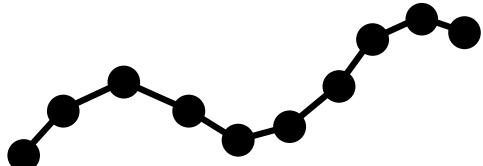
```
var coords = [    // Array
    [x0, y0, z0], // Structure (Point #0)
    [x1, y1, z1], // Structure (Point #1)
    [x2, y2, z2]  // Structure (Point #3)
];
```

- Array of Structure (AoS)

```
var coords = [    // Structure (Coordinate values)
    [x0, x1, x2], // Array (x coordinate values)
    [y0, y1, y2], // Array (y coordinate values)
    [z0, z1, z2]  // Array (z coordinate values)
];
```

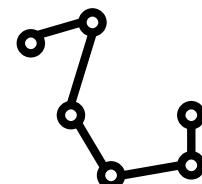
Line Data

- A line data is an object that is represented as a set of line segments and contains data related to the connections between vertices in addition to the data included in a point data.



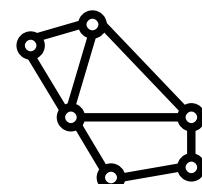
Line Strip

[0,1,2,3,4,5]



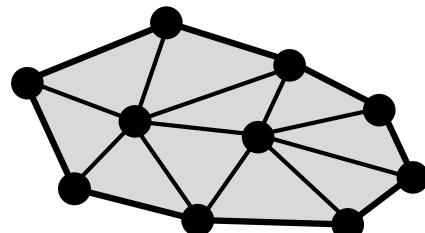
Line Piece

```
[0,1, // piece #0  
 1,2, // piece #1  
 0,2, // piece #2  
 ...  
 ]
```



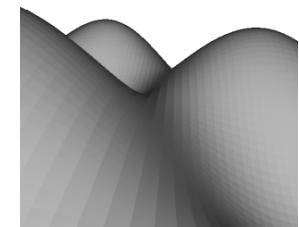
Polygon Data

- A polygon data is an data that is represented as a set of triangular faces and includes the normal vectors for each face or vertex in addition to the data included in a line data.



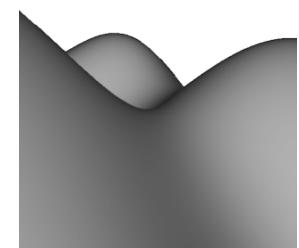
Face Normal

```
[nx0, ny0, nz0], // for face #0  
[nx1, ny1, nz1], // for face #1  
[nx2, ny2, nz2] // for face #2  
...
```



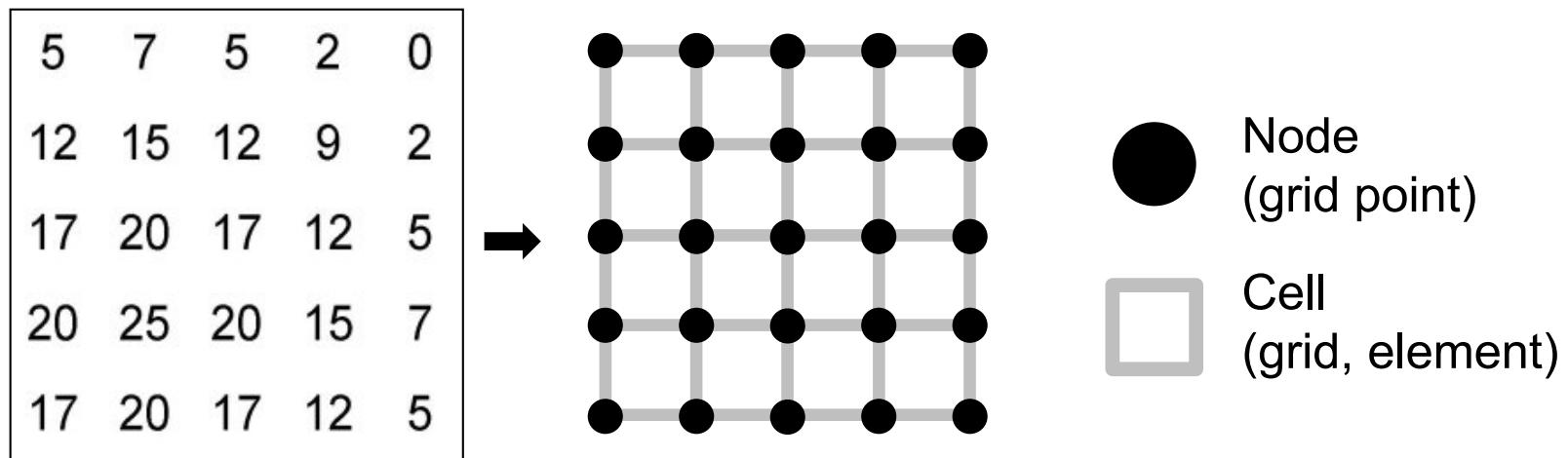
Vertex Normal

```
[nx0, ny0, nz0], // for vert #0  
[nx1, ny1, nz1], // for vert #1  
[nx2, ny2, nz2] // for vert #2  
...
```



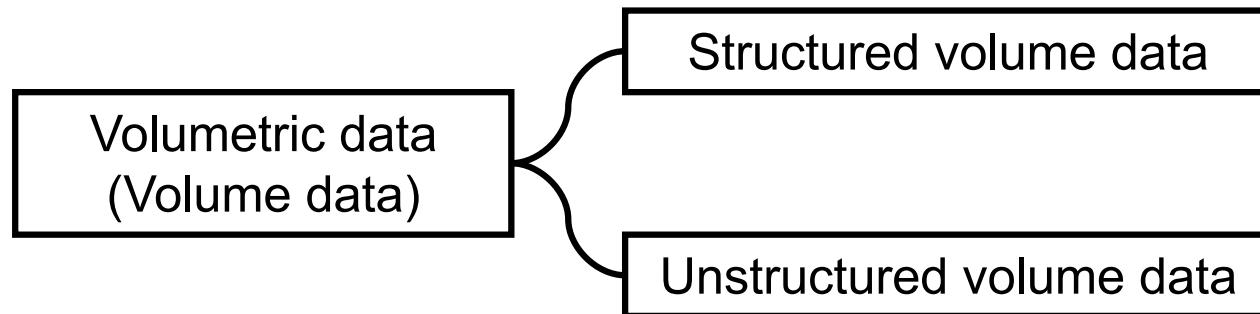
Volumetric Data

- A volume data is a data composed of nodes and cells.
- A node is a vertex of a cell, and a physical quantity, which is represented as a scalar, vector or tensor, is defined on each node.



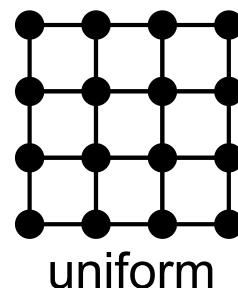
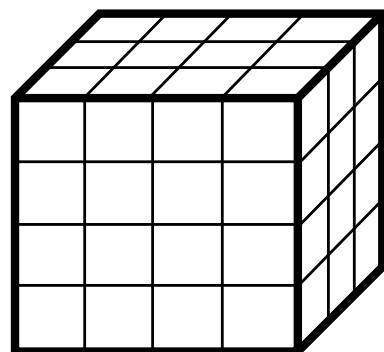
Volumetric Data

- Based on the cell shape, volume data are divided into two types, known as structured volume data and unstructured volume data.

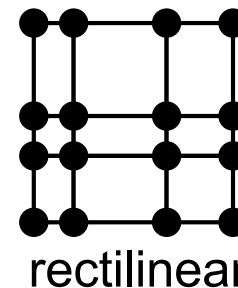


Structured Volume Data

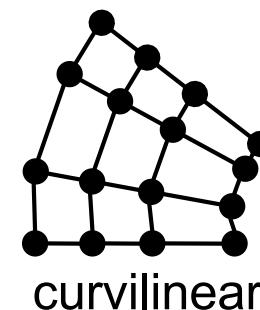
- A structured volume data is a data that is represented as a set of hexagonal cells and has a grid structure that defines the alignment of the nodes.
- There are three types of grid structure available for structured volume data: uniform, rectilinear and curvilinear.



uniform



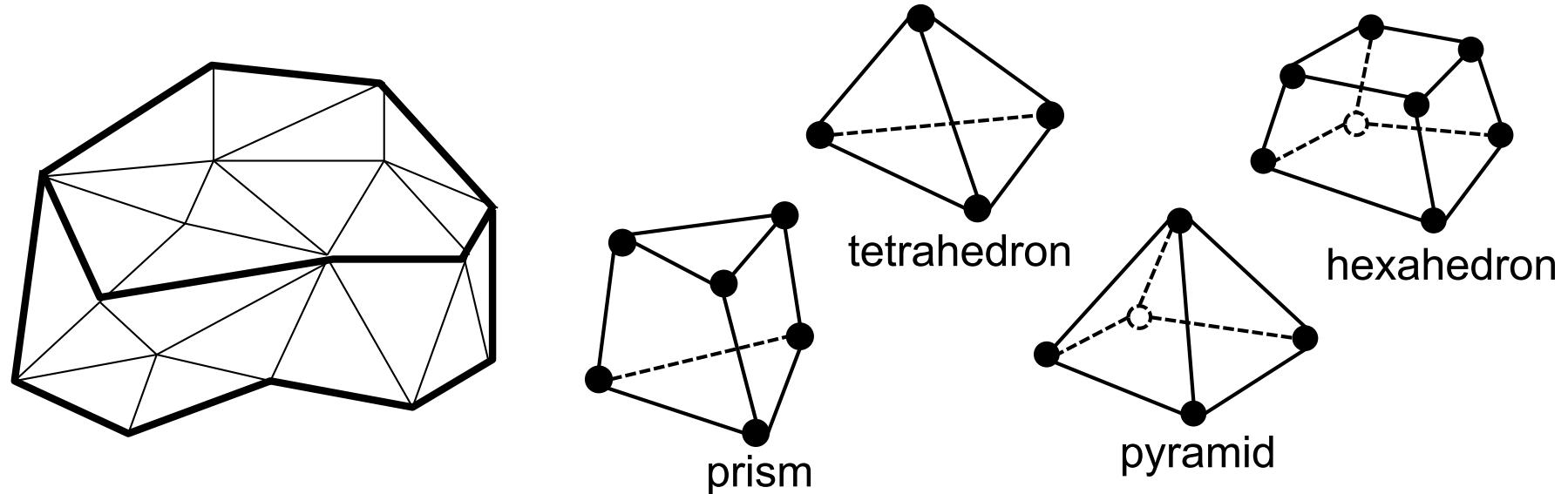
rectilinear



curvilinear

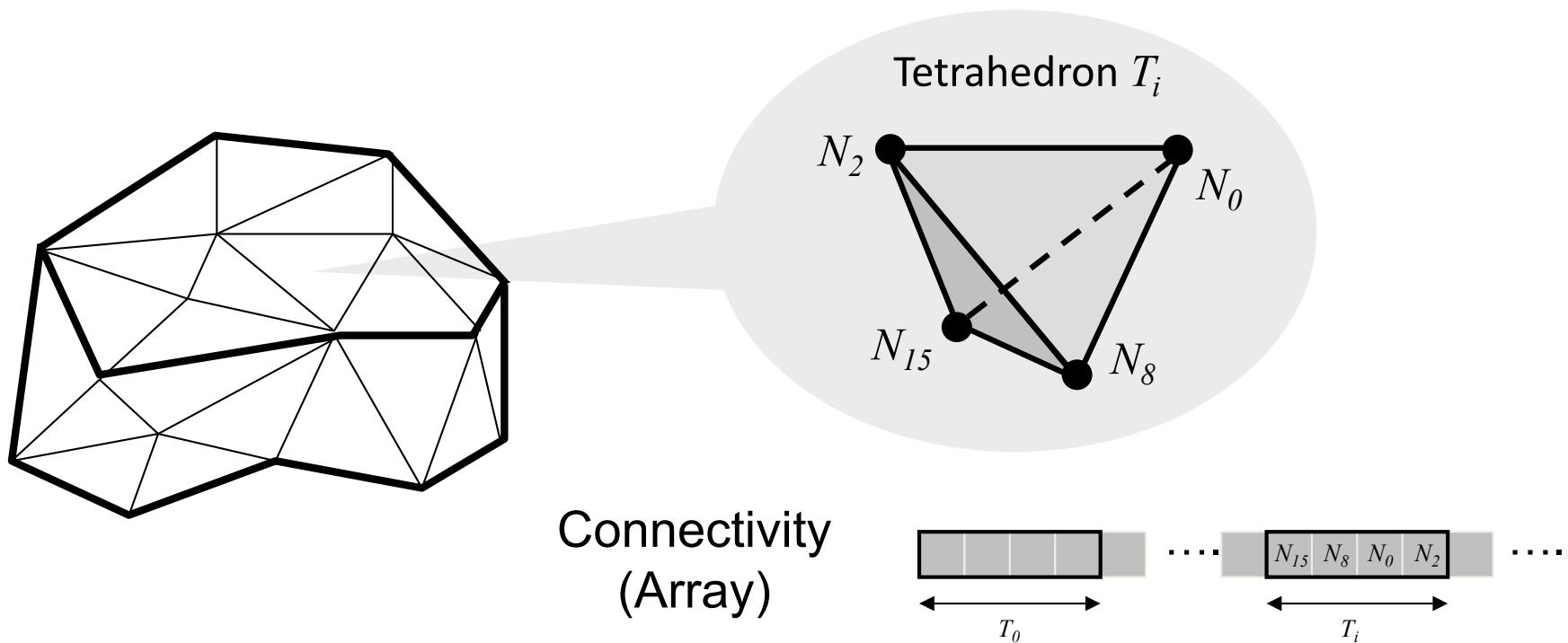
Unstructured Volume Data

- An unstructured volume data is represented as a set of arbitrarily shaped cells.
- Several types of the cell are defined, such as tetrahedral and hexahedral cells.



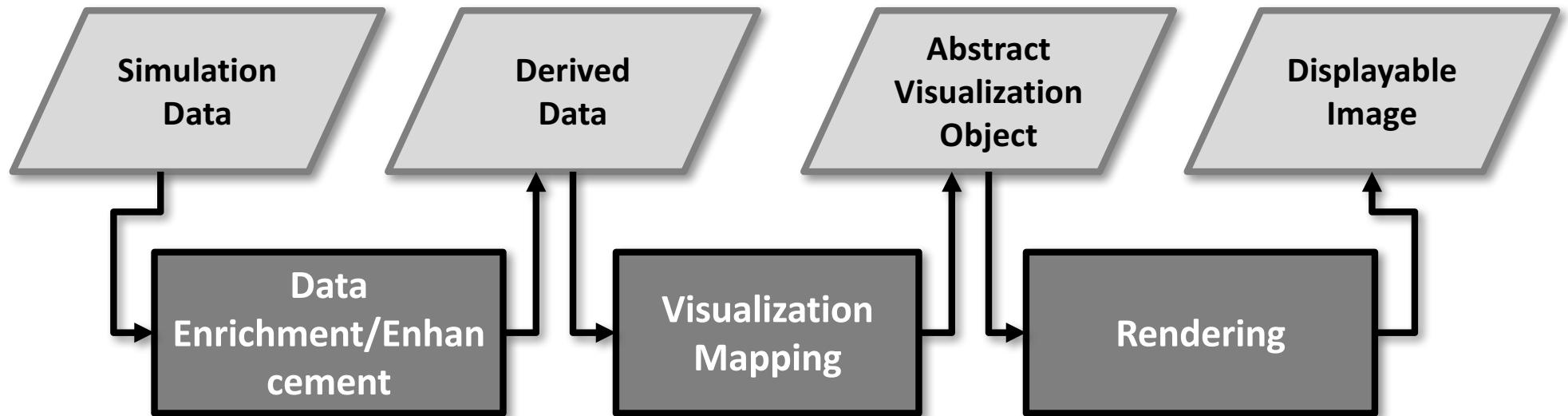
Unstructured Volume Data

- An unstructured volume data also contains cell connectivity data for each cell in addition to the coordinates and physical quantities defined for each node.

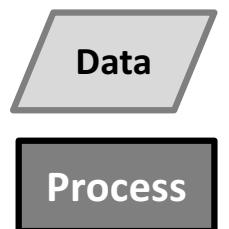


Visualization Pipeline

- Visualization process model

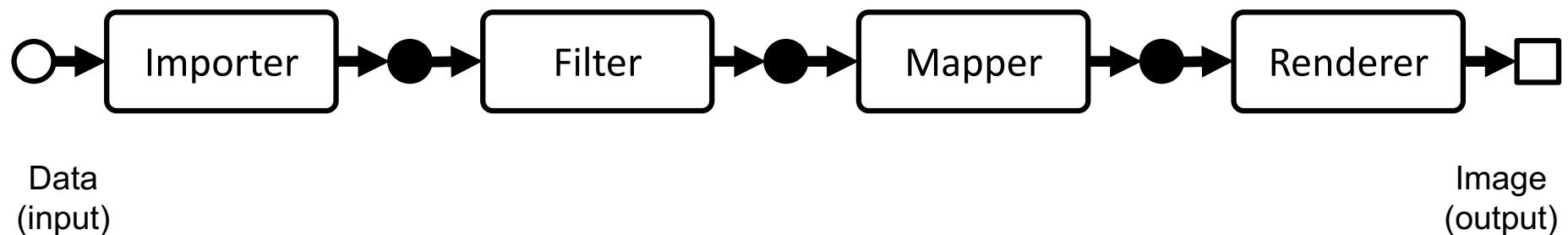


Habber-McNabb dataflow model [Habber et al., 1990]



Visualization Pipeline

- In a lot of visualization applications, each visualization method is implemented as a module, and these methods are divided into several types of processes.
- Each module has an input and an output port for an object.
- The ports of each module can be connected interchangeably, thus allowing the user to easily build a visualization pipeline.

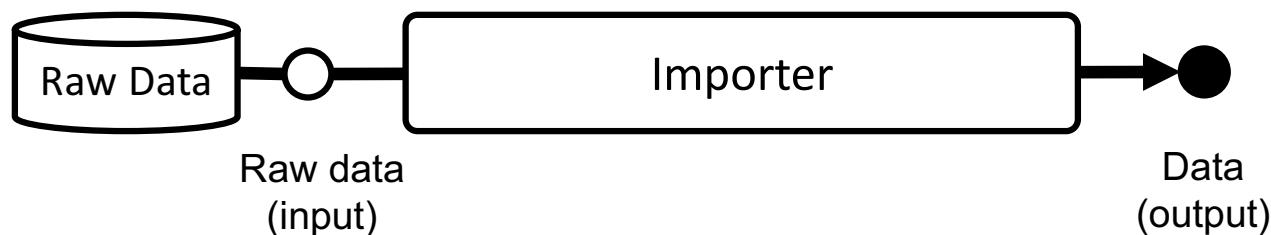


Module

- Visualization modules are divided into four types of processes:
 - Importer
 - Filter
 - Mapper
 - Renderer

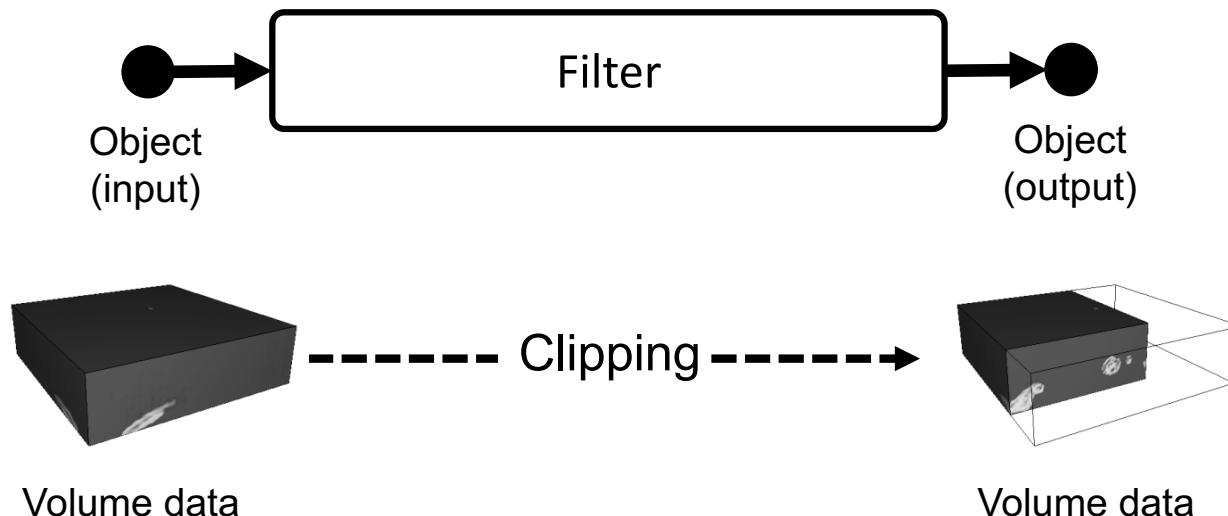
Importer

- An importer module represents the first process that is performed to import the input data into the visualization process.



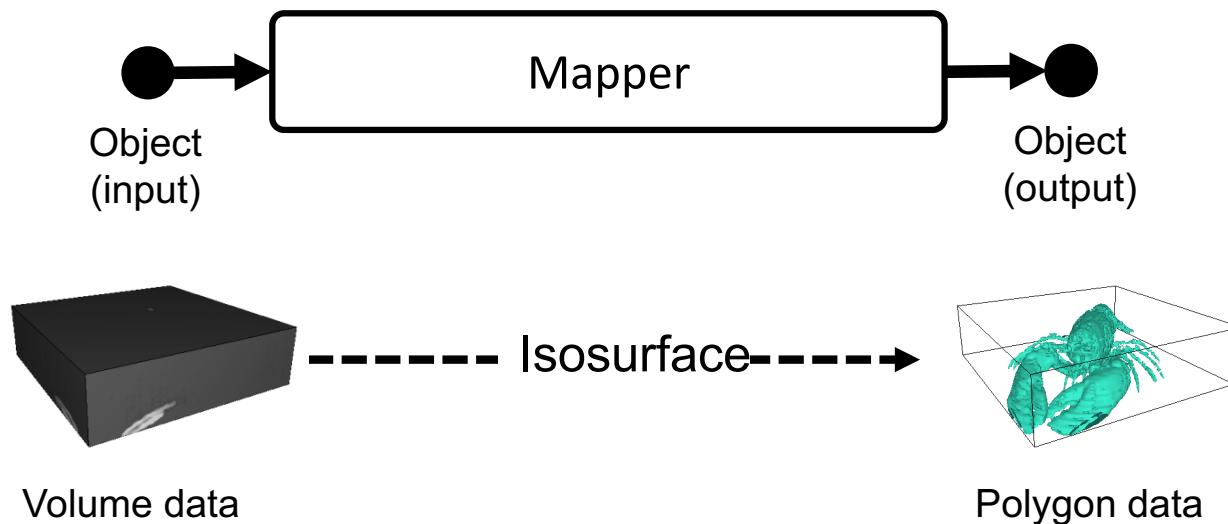
Filter

- A filter module represents a data transformation process, such as clipping, denoising, resampling, interpolation or segmentation.



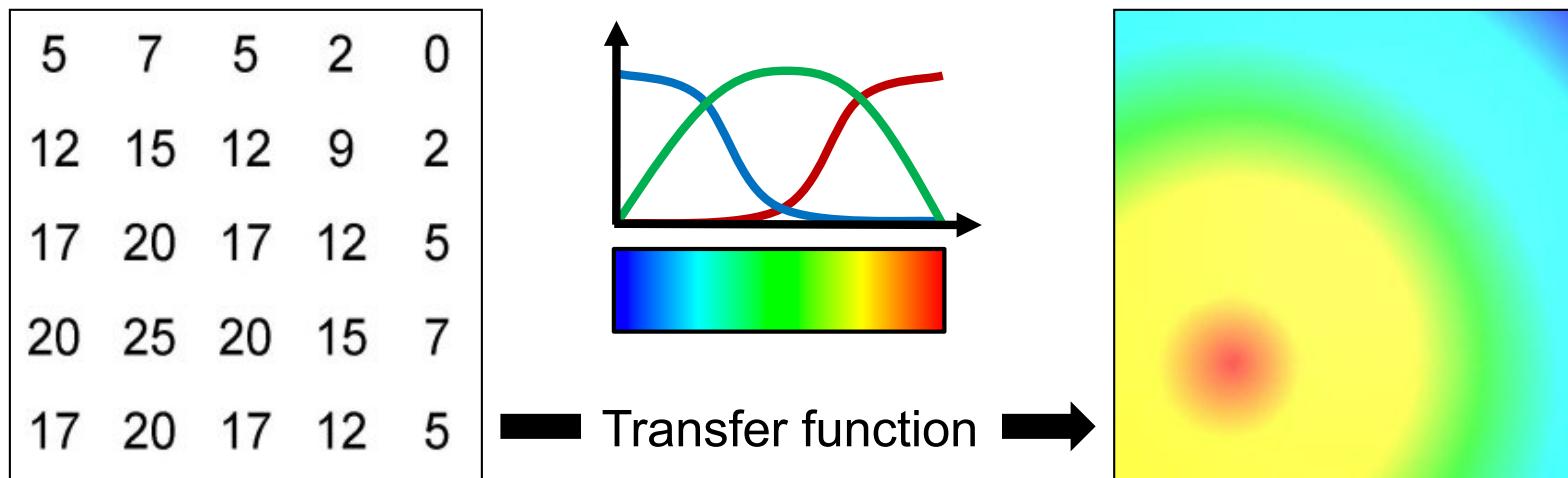
Mapper

- A mapper module represents a process for converting data into graphical primitives that can be directly drawn in a subsequent rendering process.



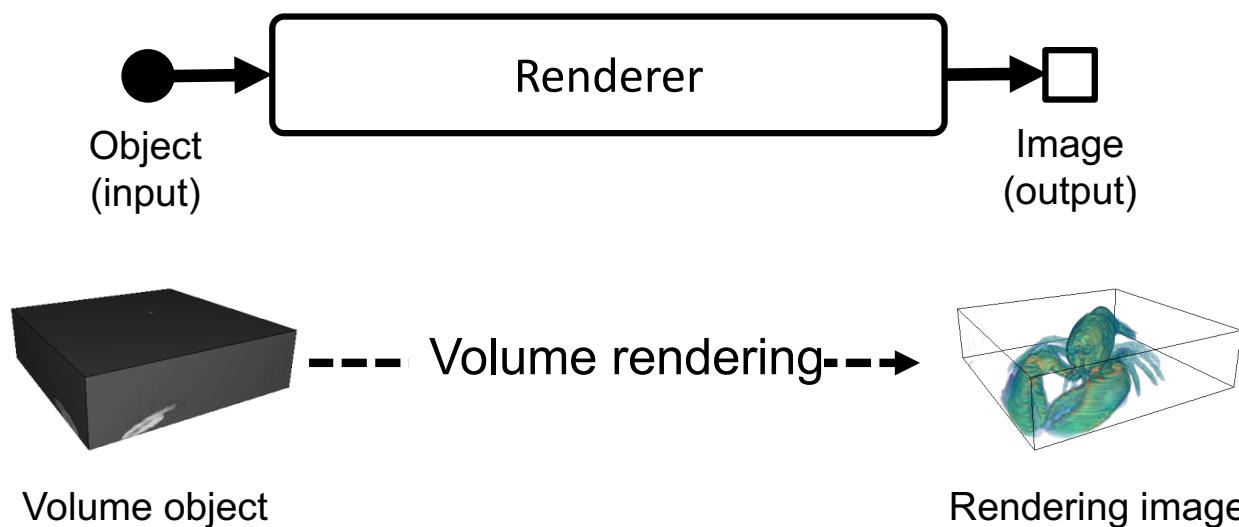
Mapper

- In the mapping process, color and opacity values will be assigned to a volume object based on physical quantities through the application of a transfer function



Renderer

- A renderer module represents the final process that is performed to draw graphical primitives on the image plane. In the rendering process, image data are generated and drawn into a frame buffer on a GPU.

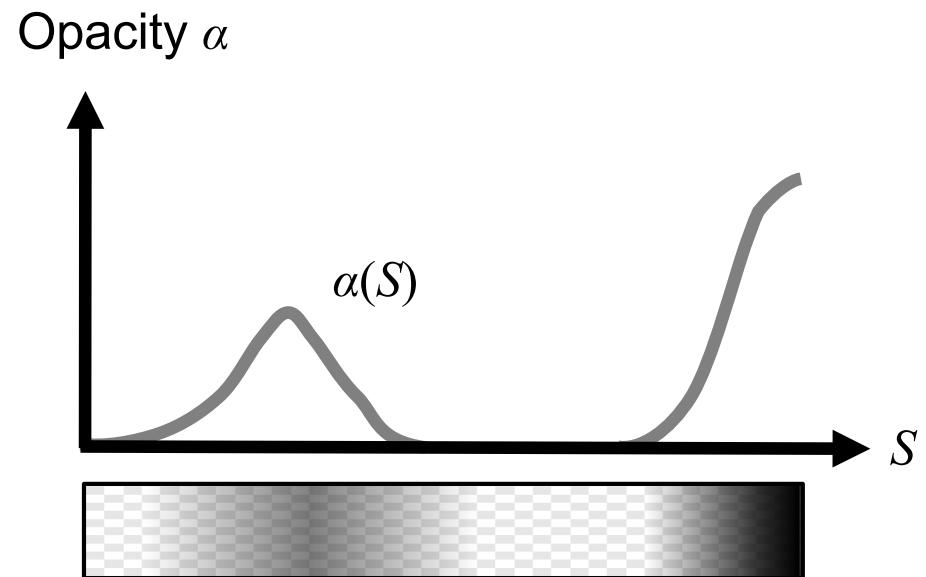
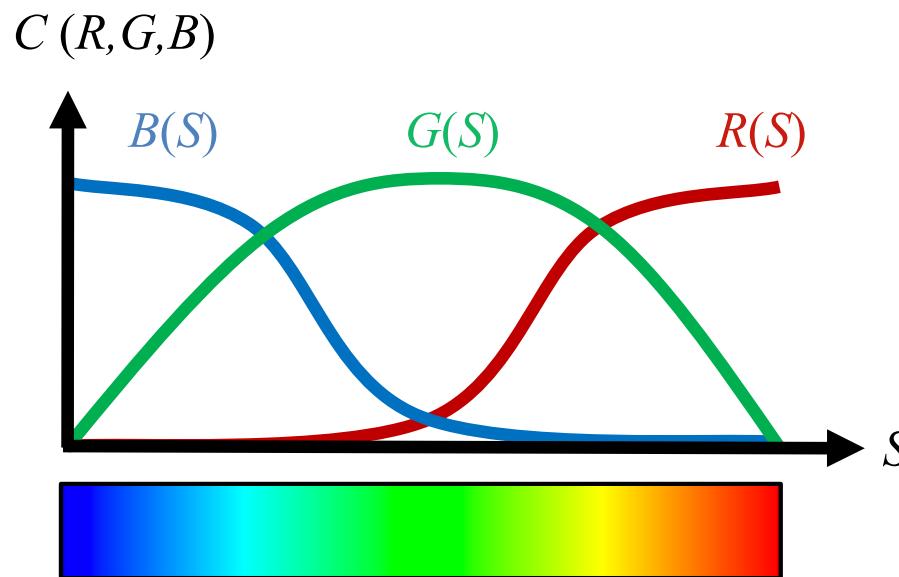


Renderer

- Although a ‘geometry-to-image’ transformation can be executed using the standard APIs provided in OpenGL, it is necessary to provide a renderer for transformations of the ‘volume-to-image’ type; such transformations are typically referred to as direct volume rendering.

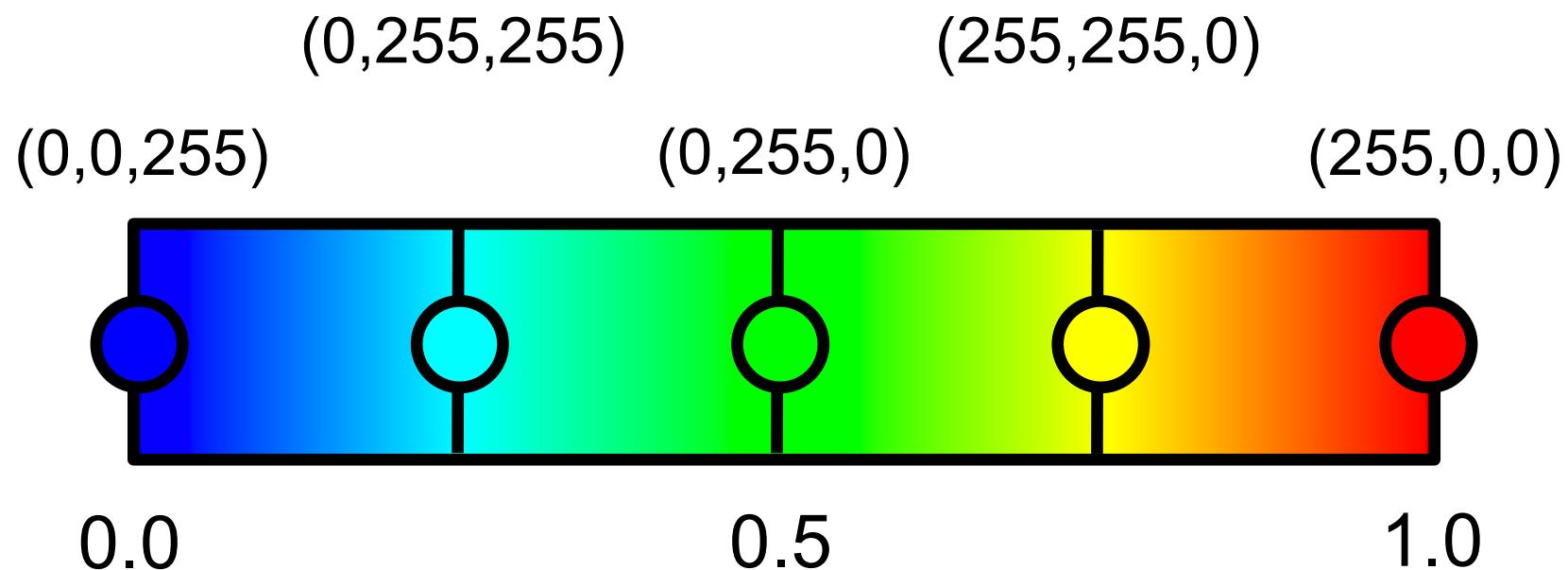
Transfer Function

- Transfer functions make volume data visible by mapping data values to optical properties.
 - Convert scalar values to color and opacity values
 - Represented as table (array) structures



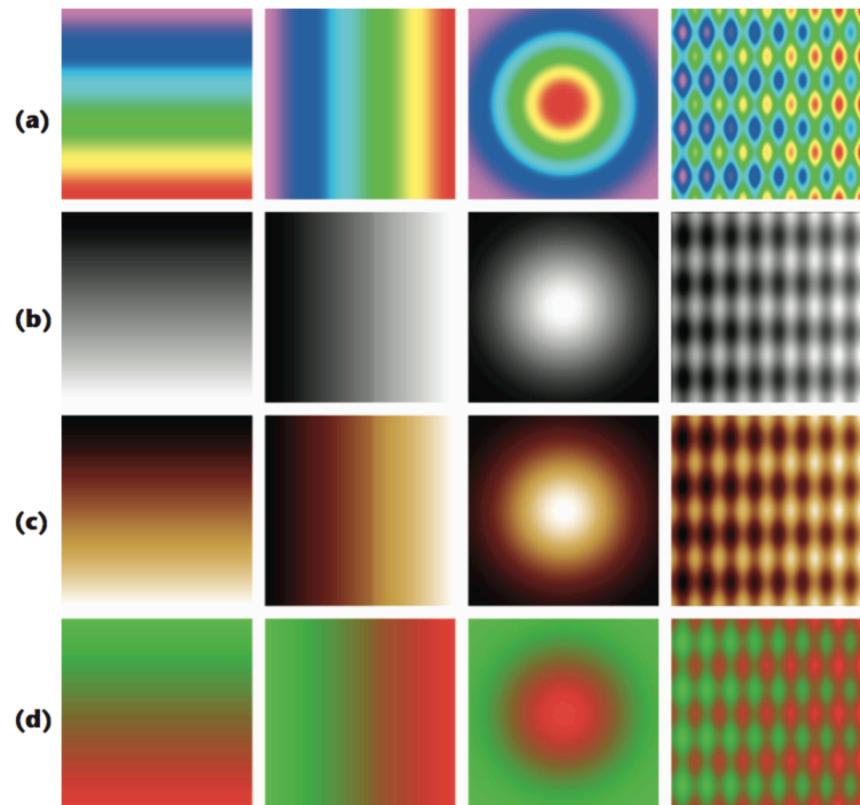
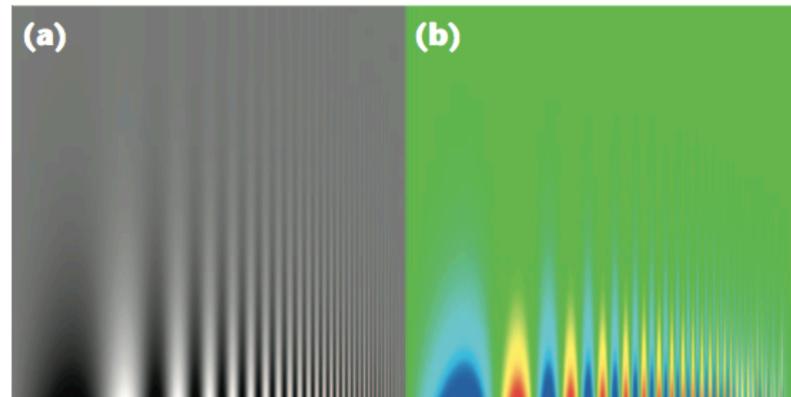
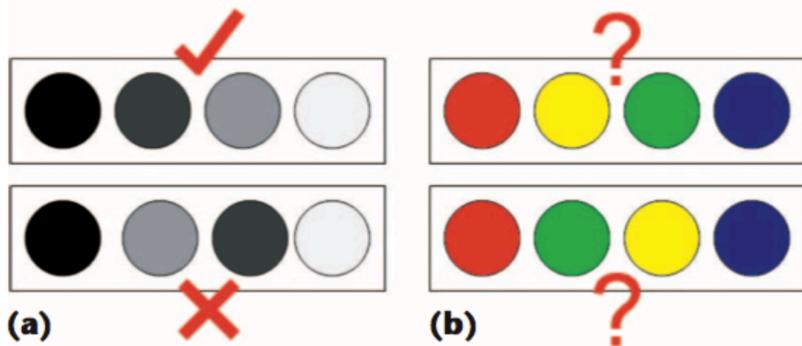
Color Map

- Rainbow color map
 - Rainbow color maps commonly appear in data visualizations in many different scientific and engineering communities.



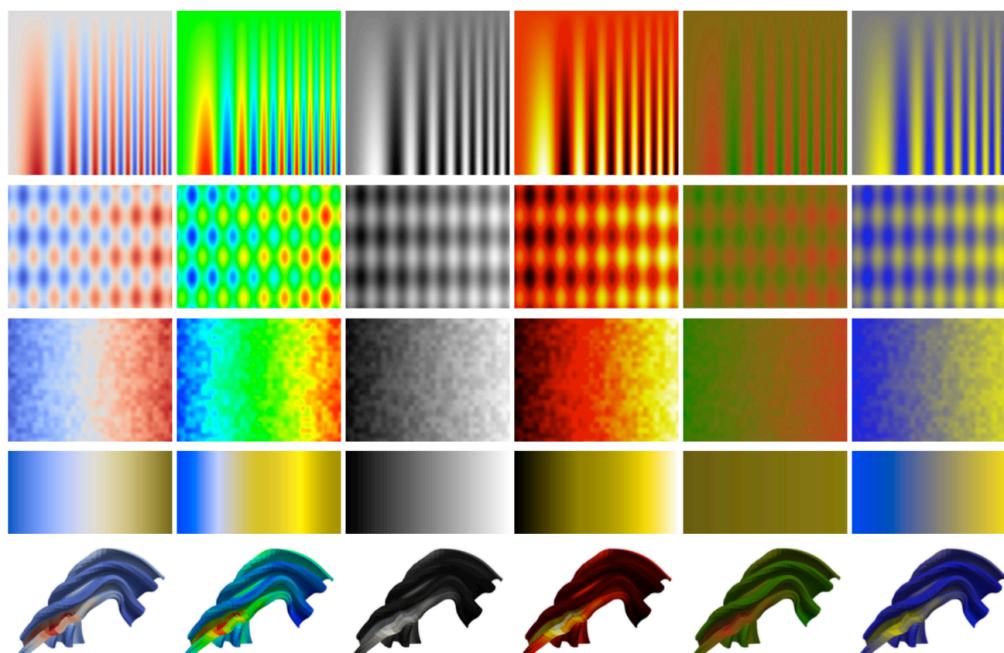
Color Map

- Problems of rainbow color map



Color Map

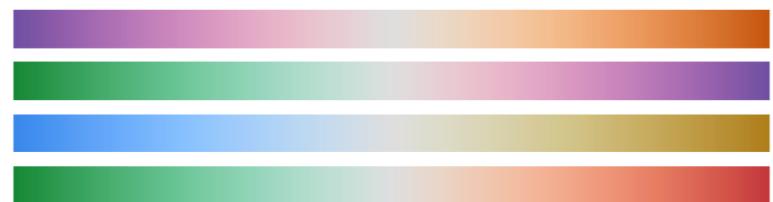
- Diverging color map



(a) Simply pick two endpoint colors to create a diverging color map in between them.

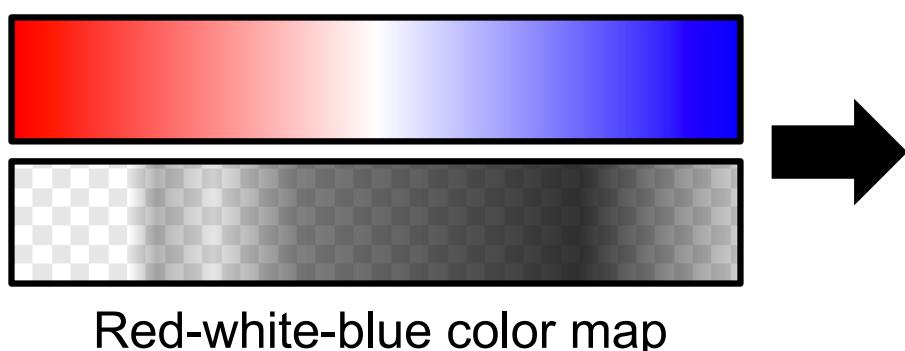
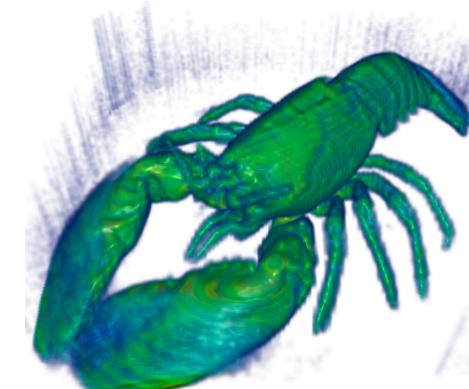
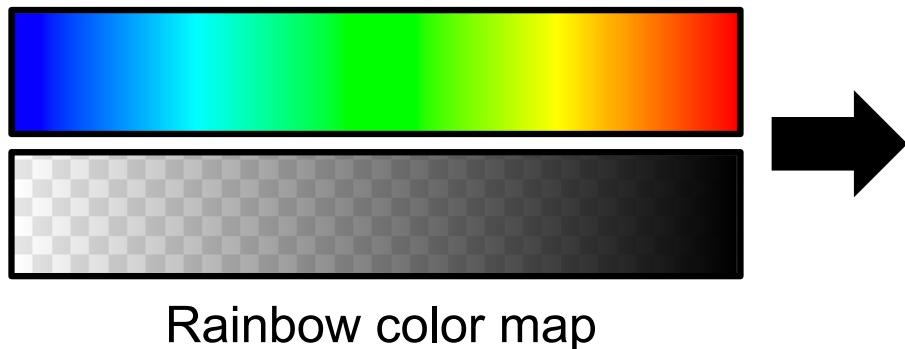
(b) Creating a white or gray control point allows you to define the intensity and location of the “middle” of the diverging color map.

(c) Adding a control point in a colored area allows you to stretch and compress regions.



Transfer Function

- Example cases for volume rendering



Polling

- Take the poll
 - Student ID Number
 - Name