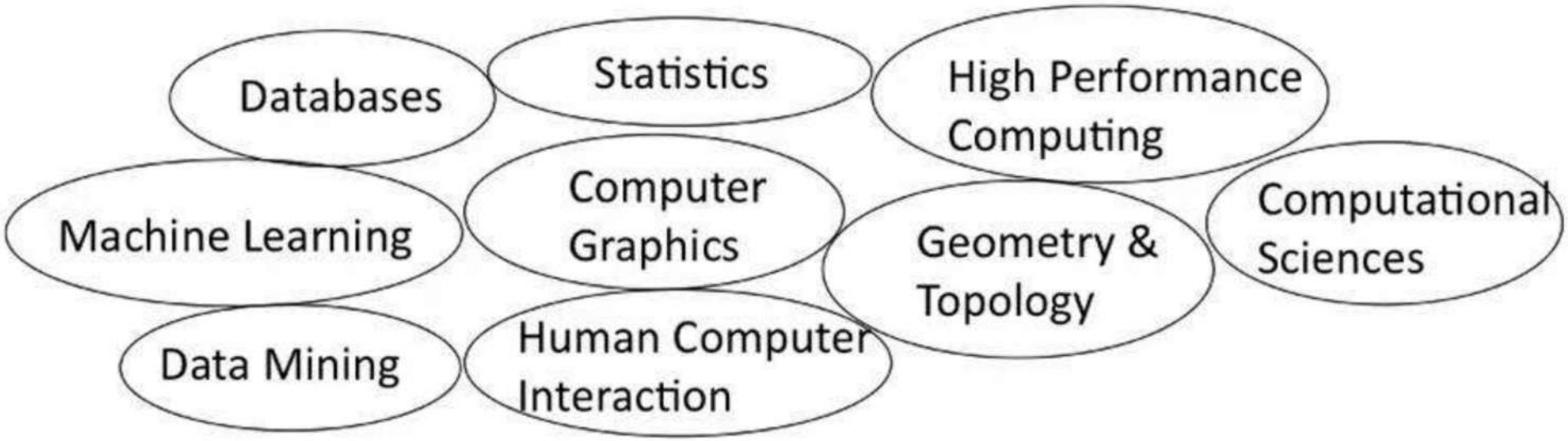
Introduction to Scientific Data Visualization and Analytics

Han-Wei Shen

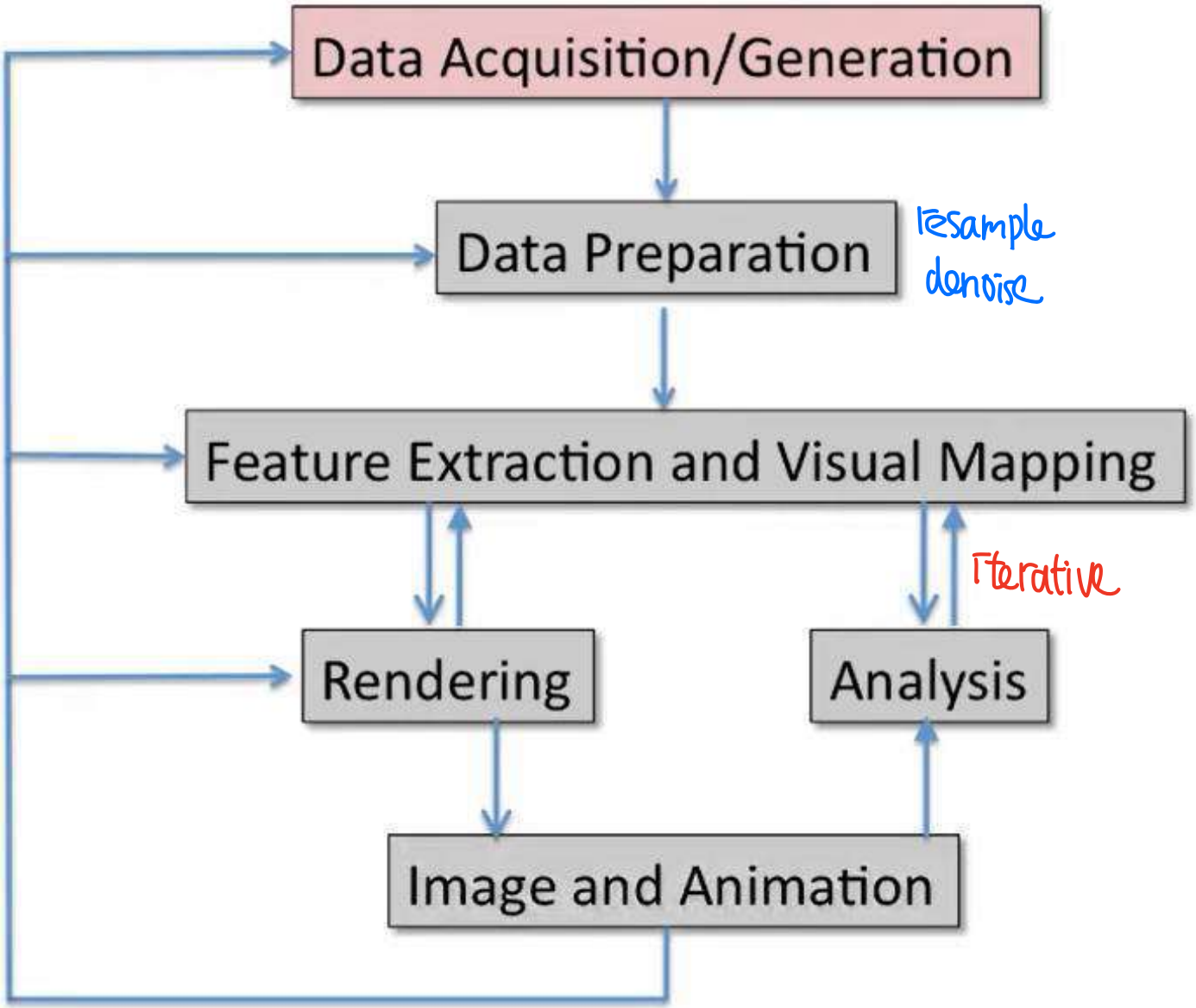
The Ohio State University

What is Visualization?

- A process of transforming numerical data to images
- The goal is to extraction information from the data, or data analytics
- Interdisciplinary

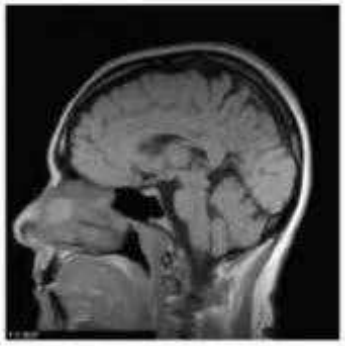


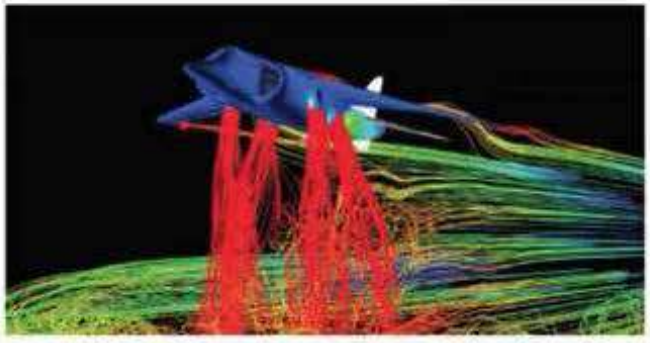
Visual Analysis Pipeline

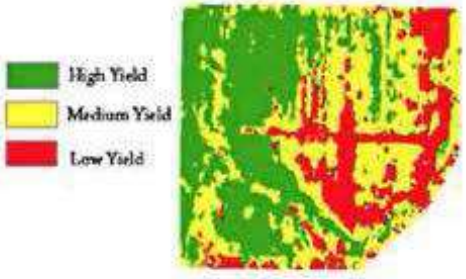


Data Acquisition and Generation

- Spatial Data
  - Medical imaging
  - Numerical simulations
  - Sensor data (Radar ,LiDAR, etc.)

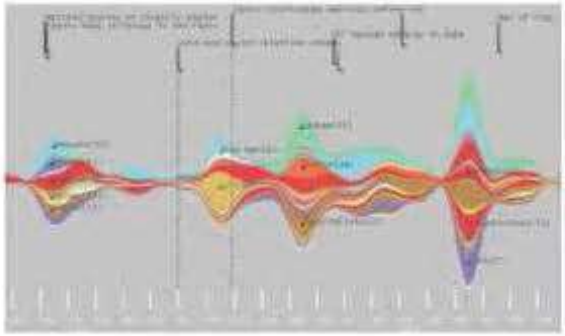


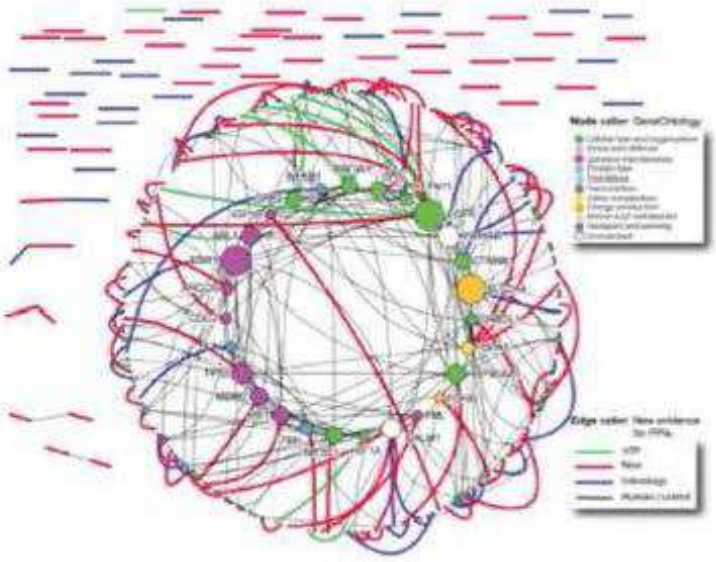




Data Acquisition and Generation

- Non-Spatial Data
  - Network Data (Graphs and Trees)
  - Text Data (Documents)
  - Matrix Data (Spreadsheets)



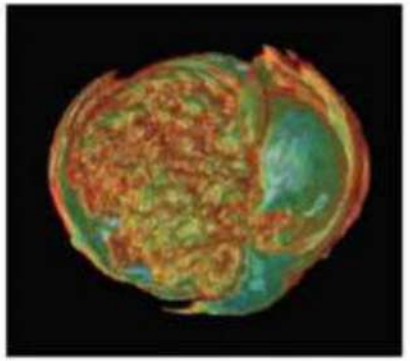


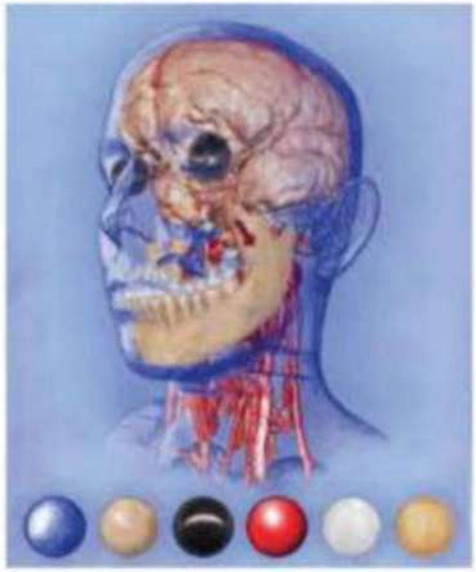
Data Preparation

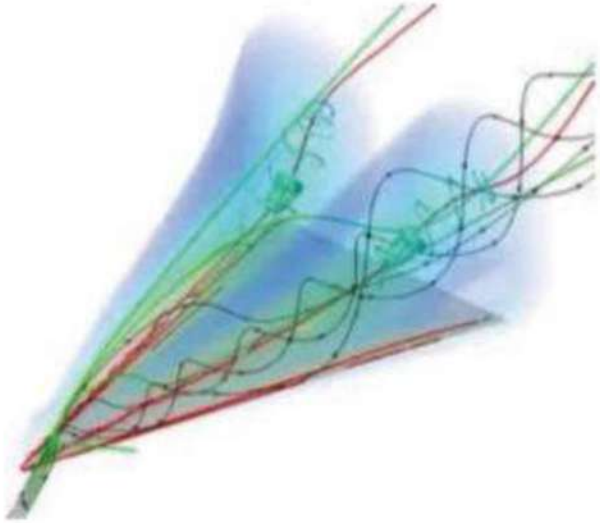
- Reconstruction
- Smoothing/De-noising
- Re-sampling to higher resolution
- Transformation (wavelet, Fourier, etc. transforms)
- Projections (to lower dimensions)
- Compression/down-sampling
- Partitioning/Bricking
- Multi-resolution
- Data distribution and file layout
- …

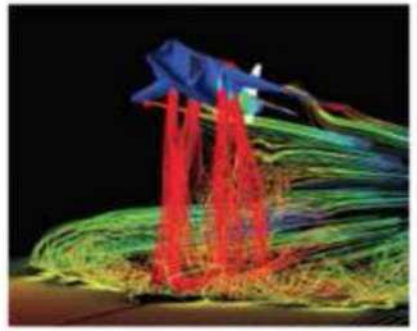
Feature Extraction

- Generic features: isosurfaces, streamlines, critical points etc.
- Specific features: vortices, material boundaries, flow separation, etc







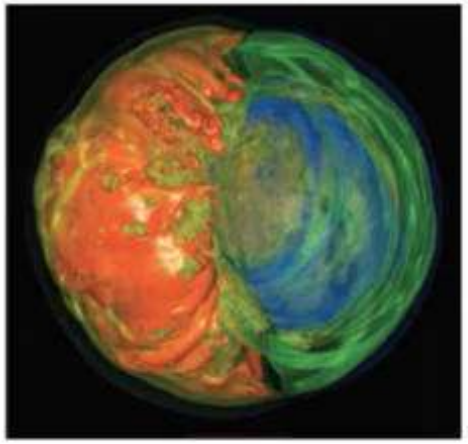


Visual Mapping and Rendering

- The process of converting features to visual forms (img)

$$I(D) = I_0 \exp^{-\int_0^D \tau(s)\,dt} + \int_0^D c(t)\tau(t)dt \exp^{-\int_s^D \tau(s)\,dt}$$

- Volume rendering $f(s, \nabla s) = (r, g, b, \alpha)$
  - Optical models
  - Transfer functions
- Polygon rendering
  - Raster graphics
  - Ray tracing



- Advanced illumination and stylized rendering

retrieve information from data

Analysis

The goal of analysis is to extract/prioritize features and raw data

- Topological
  - Surface topology: contour trees, reeb graphs
  - Vector field topology (sink, source, saddle)
- Geometry
  - Surfaces and curves: first and second fundamental forms, various types of curvatures
- Statistical
  - First, second, or higher order moments
  - Distribution, histograms
  - Entropy, information theory

Image and Animation

- Often the final output of the visualization pipeline
- The quality of the output is affected by the visualization algorithm parameters
- Many metrics are available to evaluate the images and in turn to optimize the visualization algorithm paremters
- Visualization algorithms can also be optimized if the property of the images are considered (visibility, object size etc)

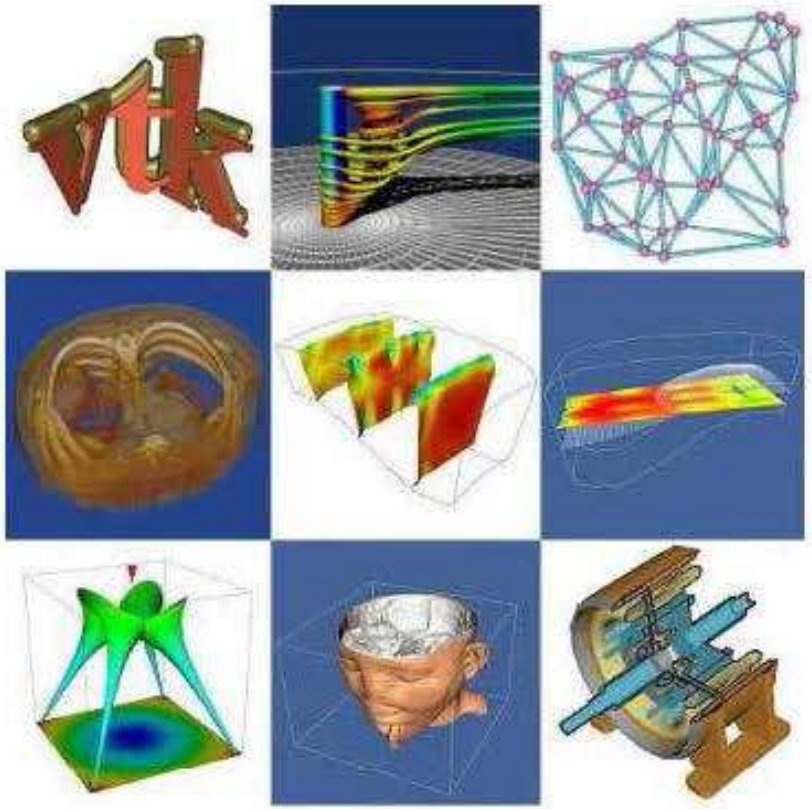
VTK: The Visualization Toolkit

Part I: Overview and Graphics Models

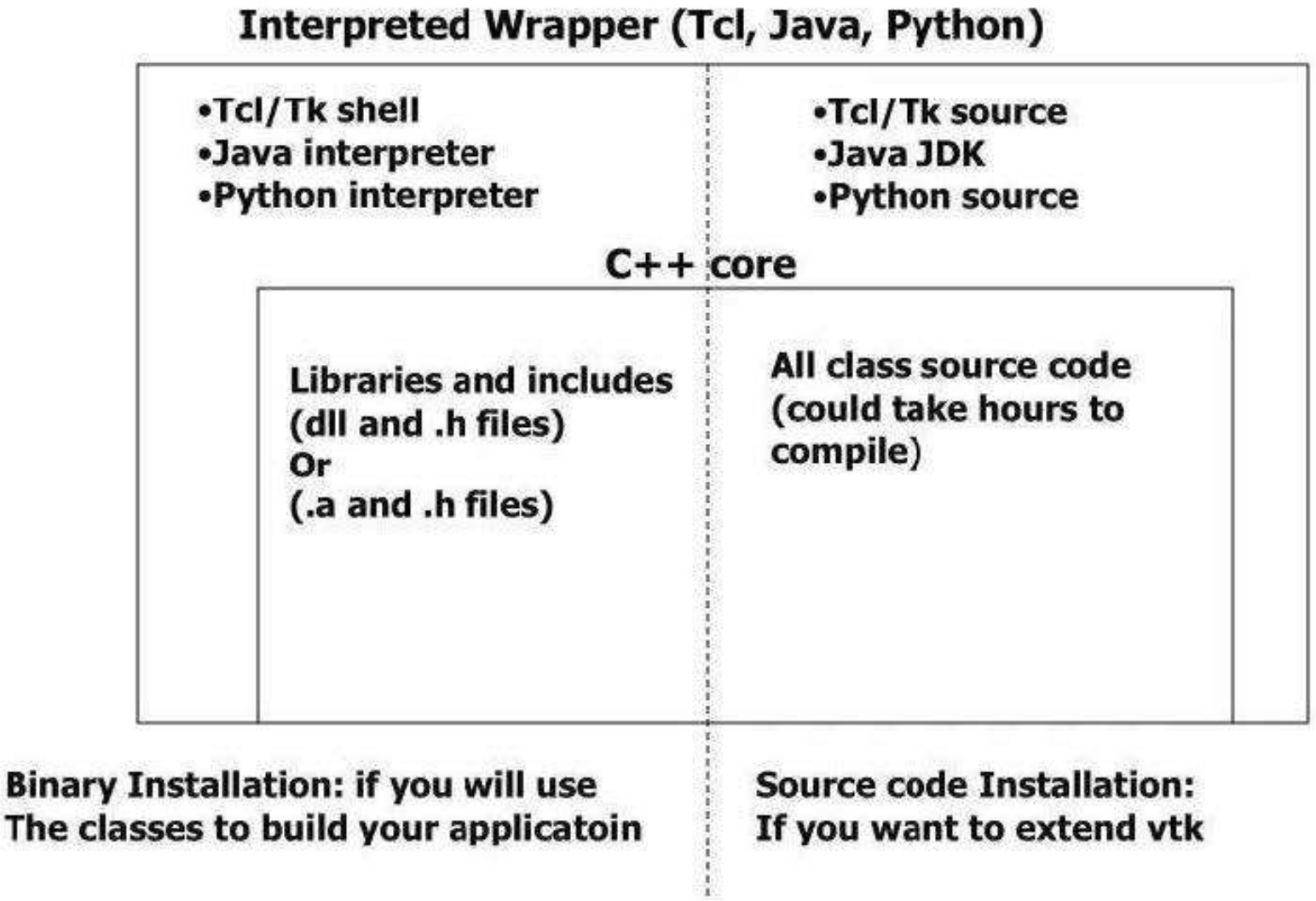
Han-Wei Shen
The Ohio State University

What is VTK?

- An open source, freely available software system for 3D graphics, image processing, and visualization.
- Support for hundreds of algorithms in visualization and image processing
- Object-oriented design with different interpreted language wrappers.



System Architecture

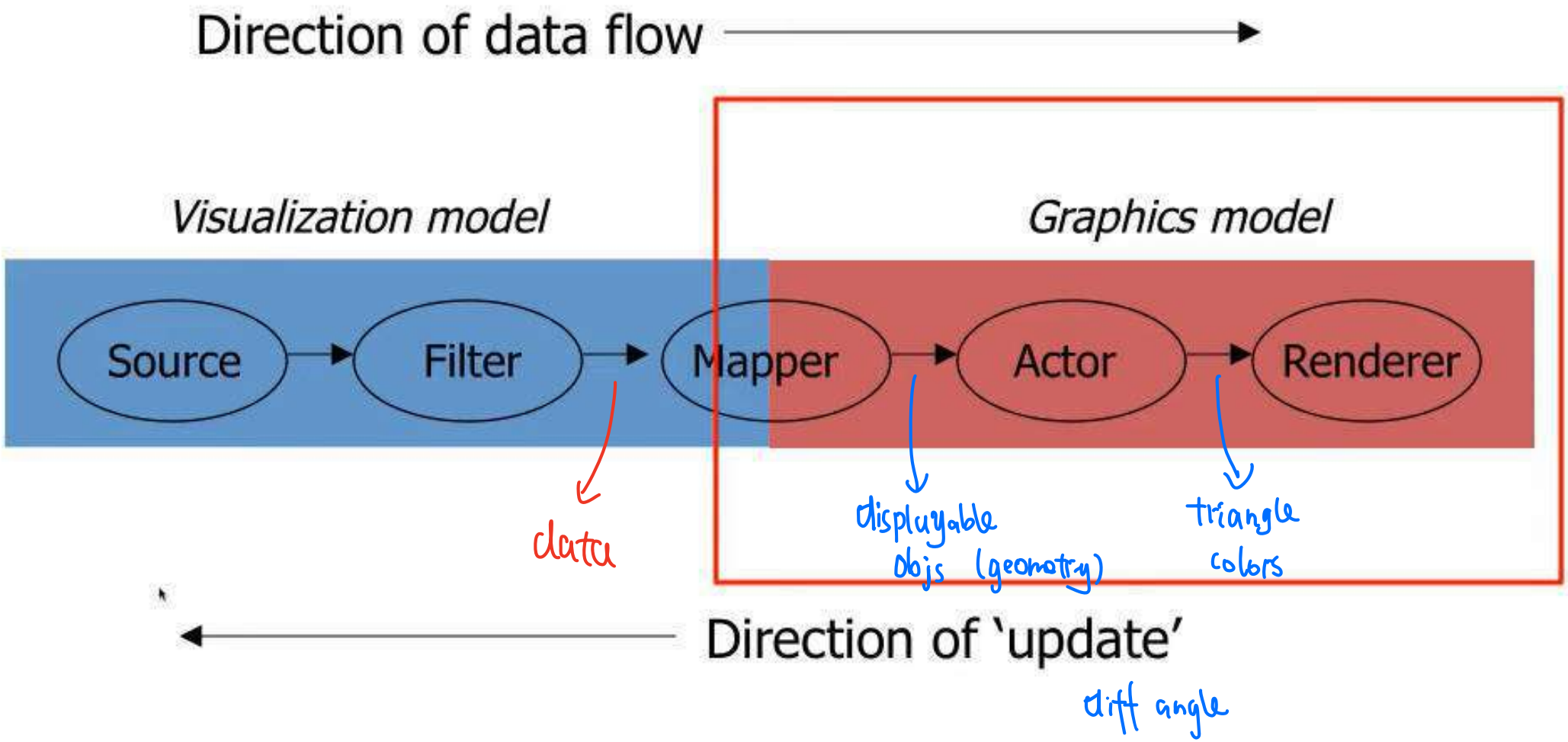


VTK Object Models

- Graphics and Visualization Models
  - Graphics objects: rendering
  - Visualization objects: generating graphical objects to represent the data

Data Flow System: Pipeline execution

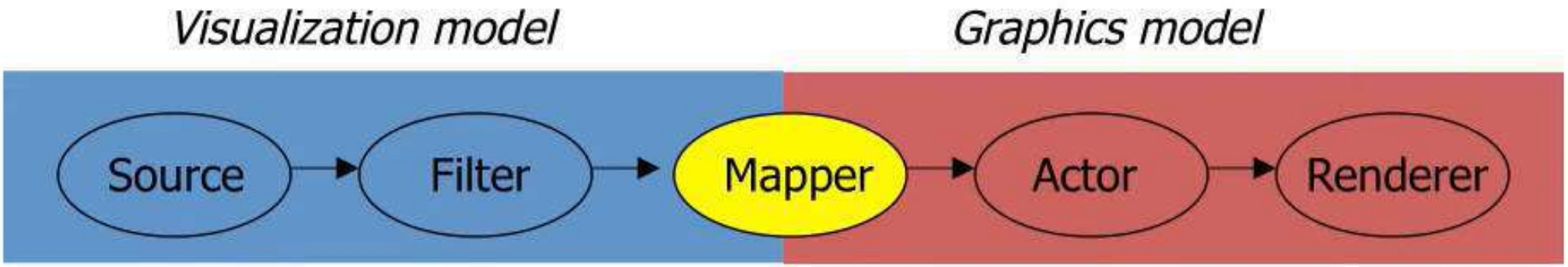
Pipeline Execution



VTK Mappers

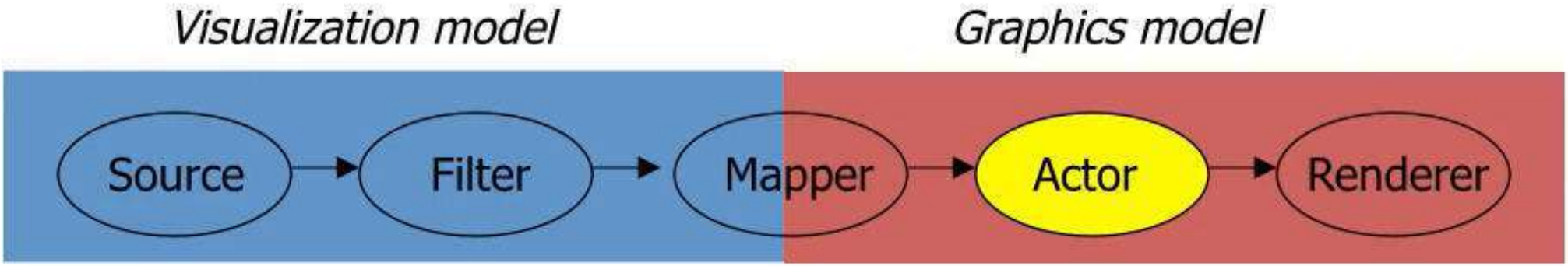
- Mappers convert data into graphical primitives or write to a file (writer)
  - Mappers require one or more input data objects
  - Mappers terminate the visualization pipeline
- Example: vtkPolyDataMapper, which takes geometry such as cylinder or cone as input and convert it to renderable geometry

a collection of triangles



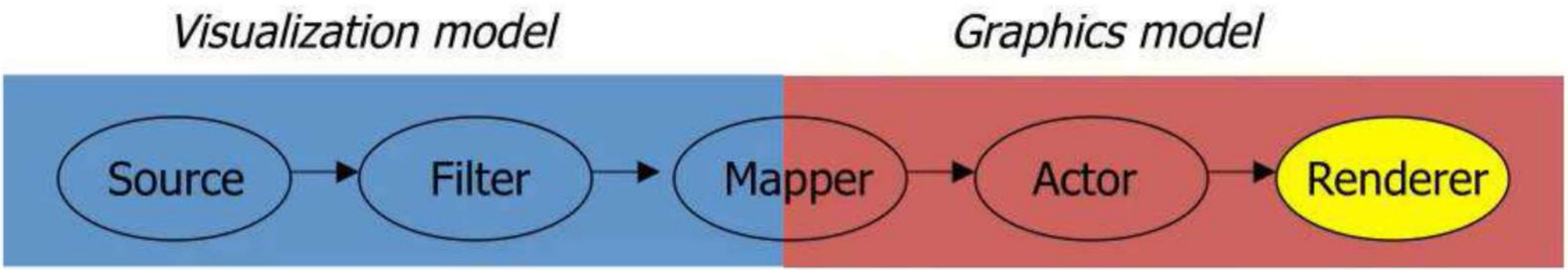
VTK Actors

- Actors represent graphical data or objects
- A VTK actor contains
  - object properties (color, shading type, etc)
  - geometry ~ triangles
  - transformations
- VTK actors need to work together with lights (vtkLight) and camera (vtkCamera) to make a scene
- The scene is then rendered to an image by a renderer (vtkRenderer)



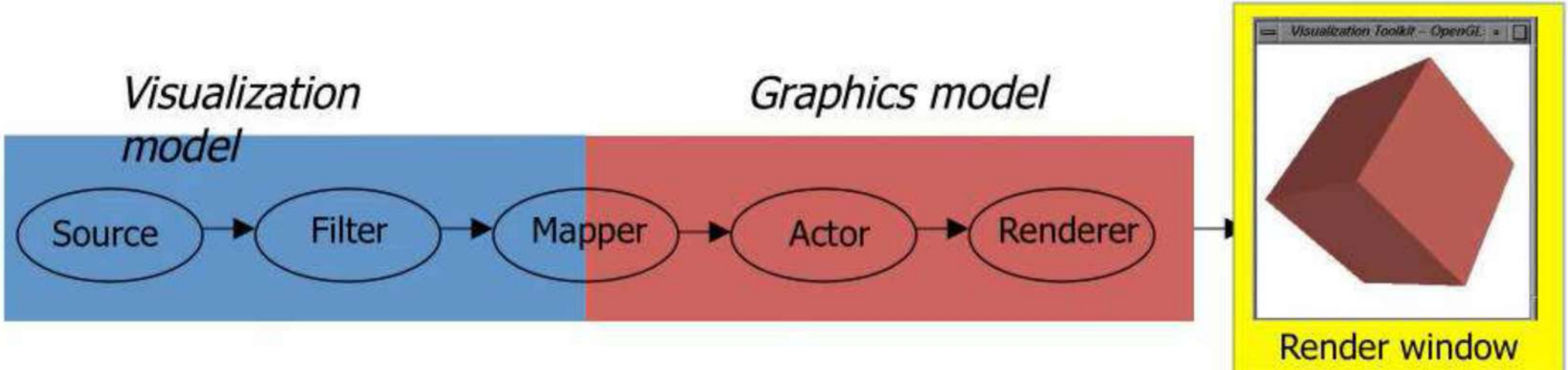
VTK Renderer

- The Renderer in VTK, vtkRenderer, is to coordinate the rendering process that involves lights, cameras, and actors
- vtkRenderer creates a default camera and lights if not present, but needs to have at least one actor
- vtkRenderer needs to be connected with a vtkRenderWindow



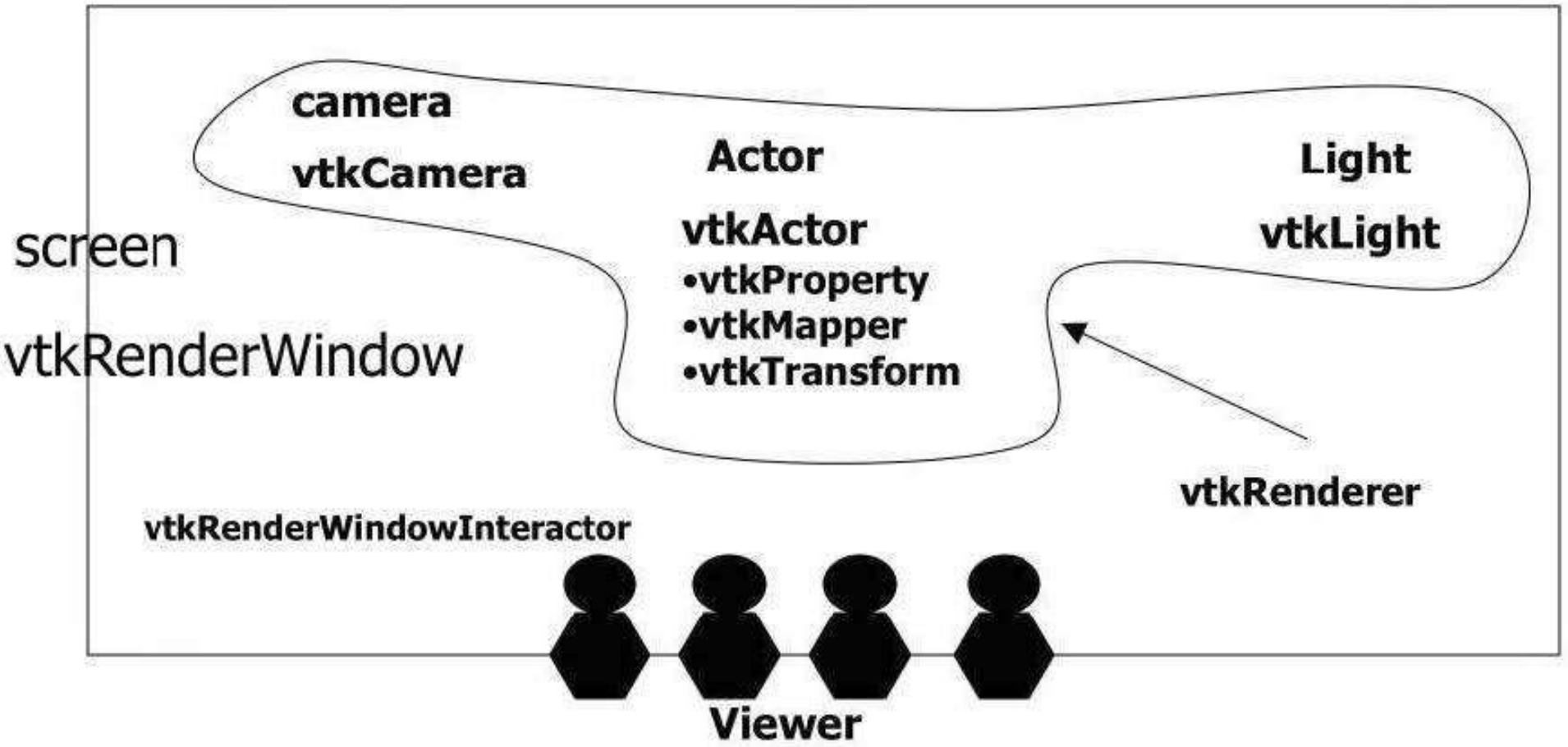
VTK Render Window

- The class, vtkRenderWindow ties the entire rendering process together
- It manages all the platform dependent window management issues and hide the details from the user
- It also stores graphics specific information such as window size, position, title, frame buffer depth, etc.



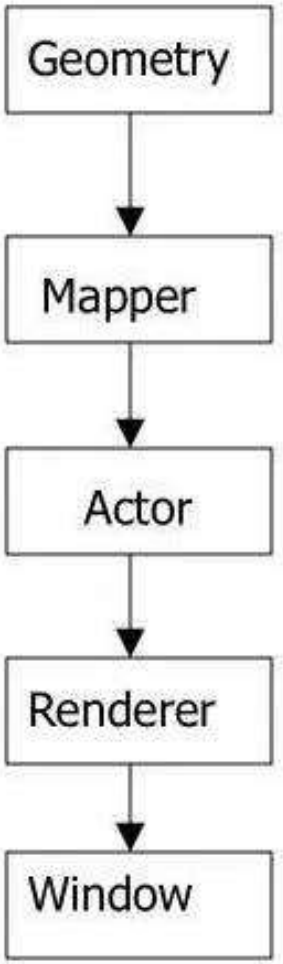
The Graphics Model

The purpose is to render the geometry (volume) on the screen



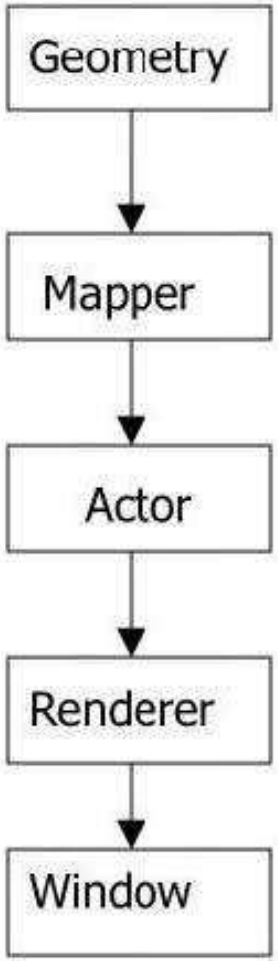
VTK Simple Pseudo Code

```
Main() {

    create geometry;
    create a mapper;
    give the geometry to the mapper;
    create an actor;
    give the mapper to the actor;
    create a renderer;
    give the actor to the renderer;
    create a window;
    give the renderer to  the window;

    window->render();
}
```

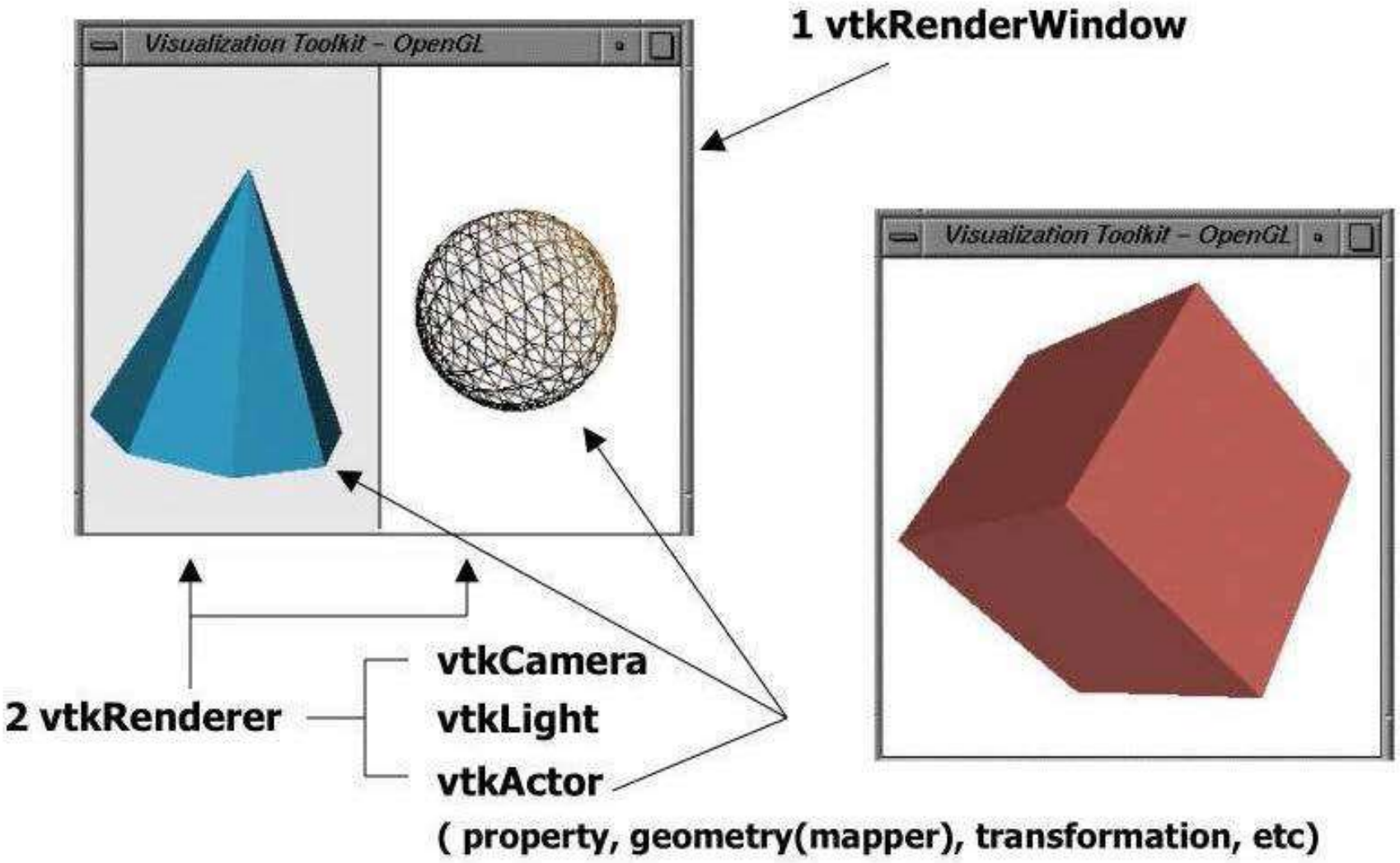


VTK Render a Cone

```
#include "vtk.h"

Main()
{
    vtkConeSource *cone = vtkConeSource::New();
    cone->SetResolution(8);

    vtkPolyDataMapper *mapper = vtkPolyDataMapper::New();
    mapper ->SetInput(cone->GetOutput());

    vtkActor *coneActor = vtkActor::New();
    coneActor->SetMapper(Mapper);

    vtkRenderer * ren = vtkRenderer::New();
    ren->AddActor(conActor);

    vtKRenderWindow *renWindow = vtkRenderWindow::New();
    renWindow->AddRenderer(ren);

    renWindow->render();
}
```



vtkRenderWindow Output



VTK: The Visualization Toolkit

Part II: Visualization Model

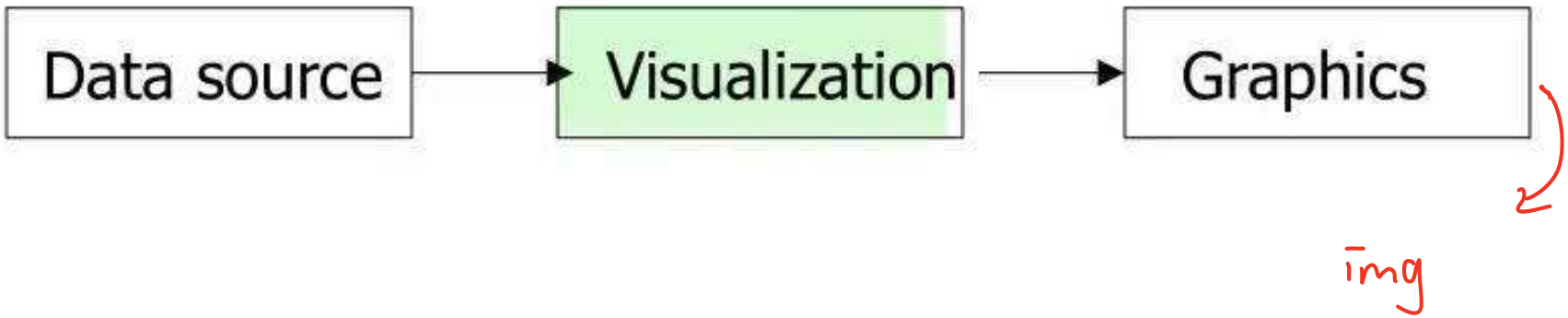
Han-Wei Shen

The Ohio State University

VTK Visualization Model

- Graphics and Visualization Model
  - Graphics model: rendering
  - Visualization model: transformation and representation

Data Flow System: Pipeline execution



The Role of Visualization Model

- Converting data from its original form into graphical form
- Deal with the issues of *transformation* and *representation*
  - Representation: internal data structures
  - Transformation: data to graphics

Example: stock price display

① Internal (computational) representation:

Arrays of stock prices

Price[i] = 17100, 16950, 17073, 17050,...

Time[i] = 10, 10:30, 11, 11:30, 12, ...

② Graphical Representation: x-y plot

3D Plot Example

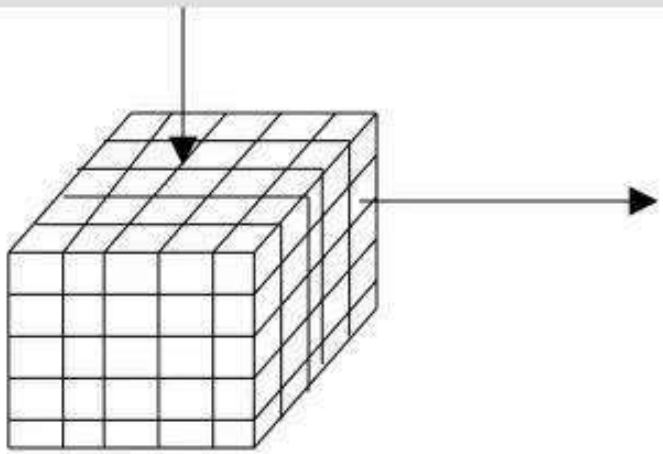
Quadric function:

$$f(x,y,z) = a_0\,x + a_1\,y + a_2\,z + a_3\,xy + a_4\,yz + a_5\,xz + a_6\,x^2 + a_7\,y^2 + a_8\,z^2$$

- Evaluate the function at a pre-determined x,y,z samples
- Store the function values into a 3D array
- Generate a 3D surface corresponding to f(x,y,z) = c (this is called an isosurface)
- Display the surface

data

geometry

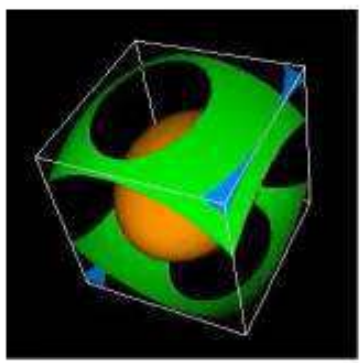


F[x][y][z] = f(x,y,z)

X	Y	Z	f
0	0	0	
1	0	0	
2	0	0	
…	…	…	

Find all (x,y,z)
f(x,y,z) = c

Transformation
(visualization)



VTK Visualization Model

How is a visualization process described in VTK?

Visualization pipeline: It describes the flow of data from its original form to the final image output

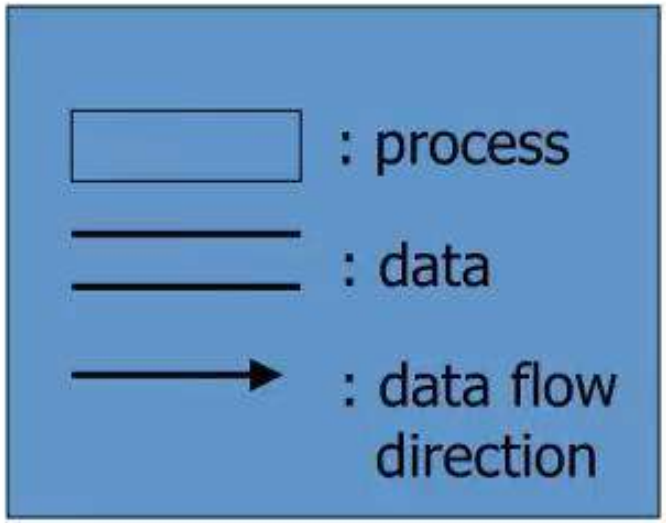
Visualization Pipeline =

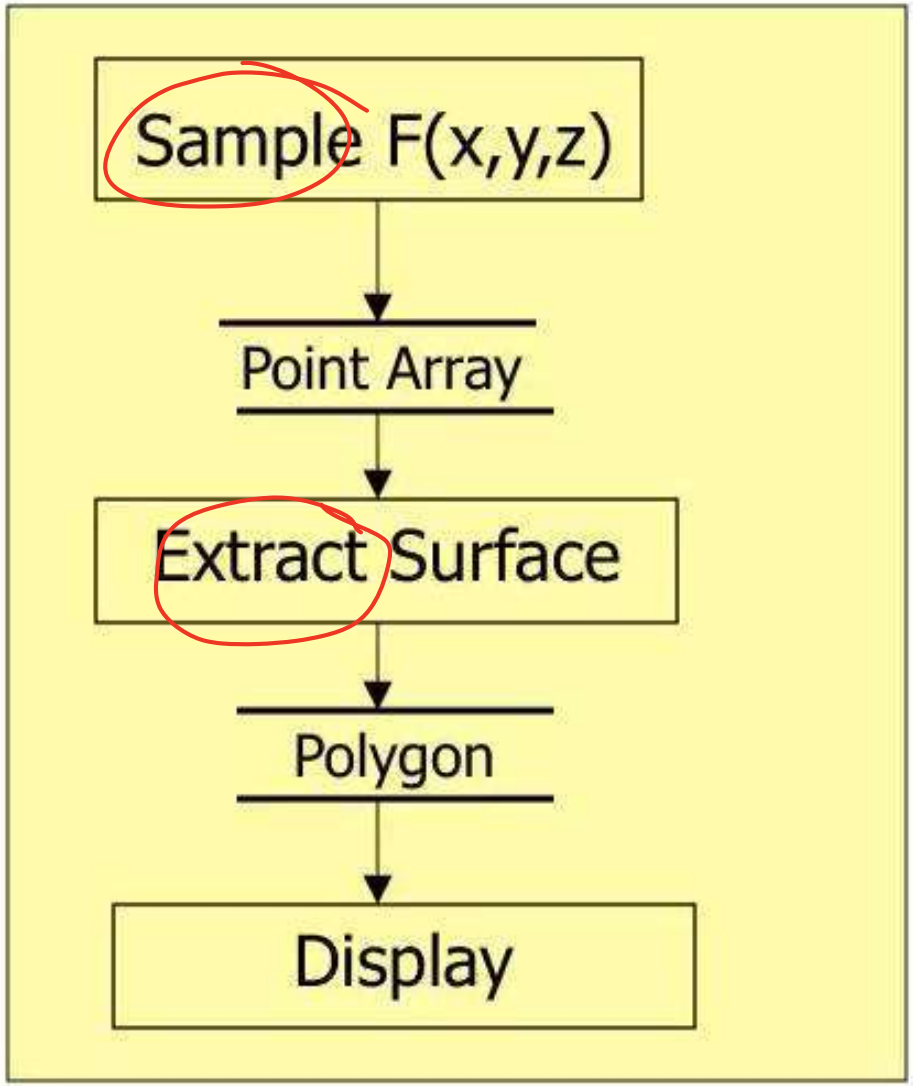
Function Model + Object Model

- Function model: Transformation
  - steps to create visualization (how data flow through the system)
- Object model: objects that participate in the function model
  - processes and internal data structures

VTK Function Model

- Function model is to Illustrate the steps to create visualization





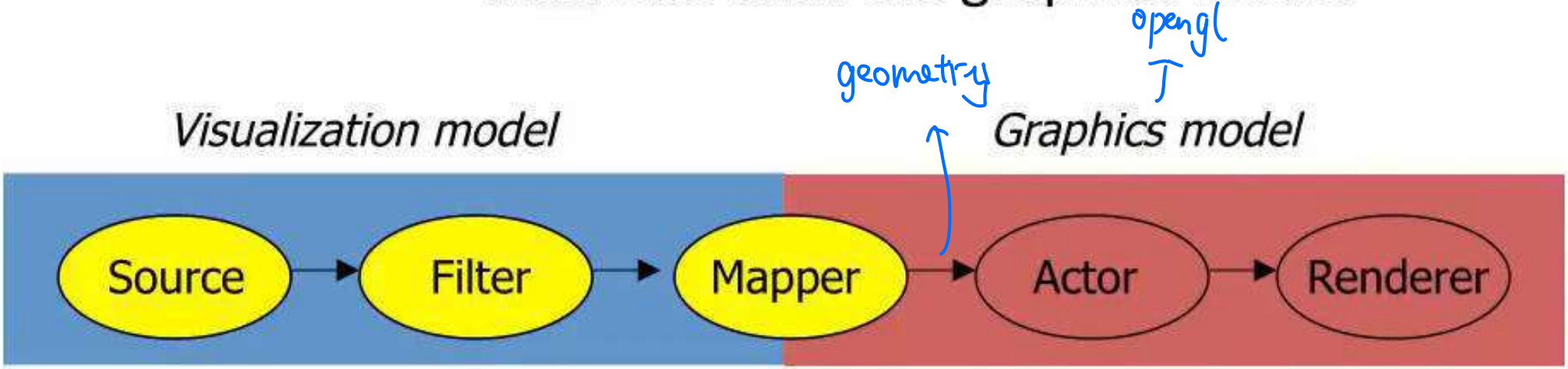
VTK Object Model

Describe which objects operate on the functional model (visualization pipeline)

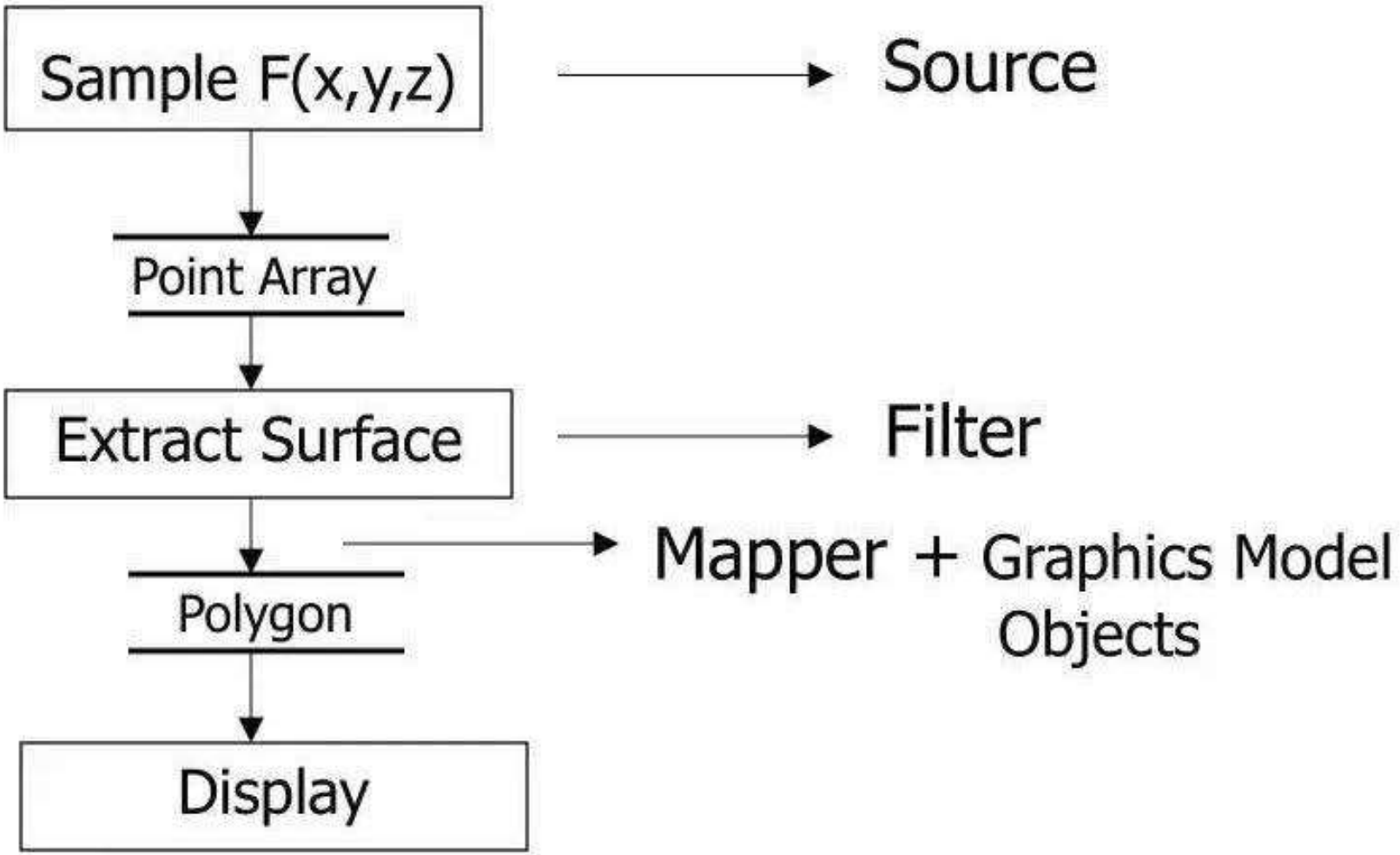
- Process Object: Operates on (transform) the data objects
  - Example: generating surfaces from F(x,y,z)
- Data Object: Represent *information* and *methods* to create, access, and delete the information
  - Example: the 3D point array F(x,y,z)

Process Objects

- Source: Interface to external data sources or generate data procedurally
- Filter: Transform the data input
- Mapper: The sink of the functional model, interface with the graphics model



Process Objects (cont'd)



Data Objects

Data Object: Dataset

- Structures: how the information is organized
  - Topology
  - Geometry

mesh
vertexes
cells

- Attributes: store the information we want to visualize. e.g. function values

Data Objects: structures

seq of connection info

- Topology: $P_1 \to P_2 \to P_3$ ~ Cell Data ⇒ triangle
  - Invariant under geometric transformation (rotation, translation, scaling etc)
  - Topological structures: Cells

connections info

Vertex
line
triangle
triangle strip

- Geometry: $P_1(x_1, y_1)$ $P_2(x_2, y_2)$ $P_3(x_3, y_3)$
  - The instantiation of the topology
  - Geometric structures: cells with positions in 3D space

3D coordinates

Visual

VTK Data Attributes

- The information stored at each corner of the cell
  - Scalars: temperature, pressure, etc
  - Vector: velocitys
  - Normals: surface directions
  - Texture coordinates: graphics specific
  - Tensors: matrices

Color mapping + isocontour

Streamlines

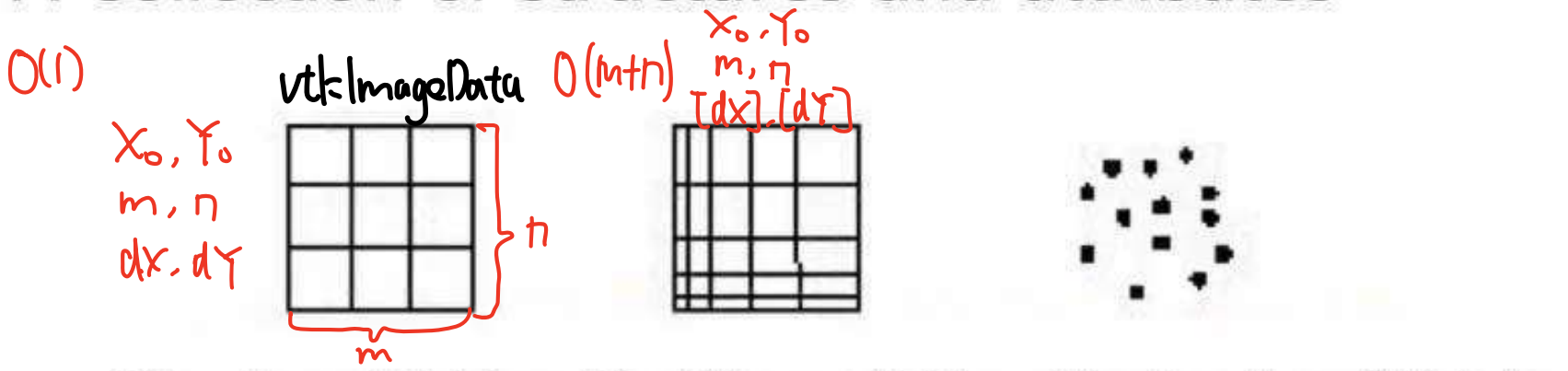
Glyph

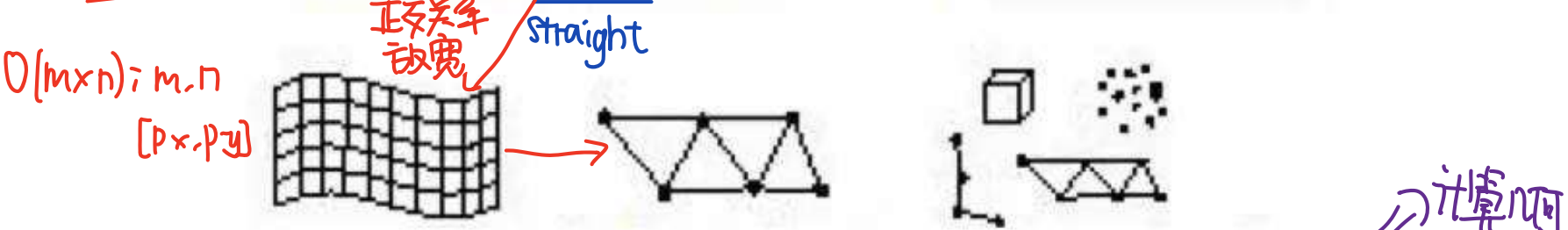
Topology

VTK Datasets

Organizing Structure
- Geometry
- Attrs
  - point
  - cell

A Collection of structures and attributes





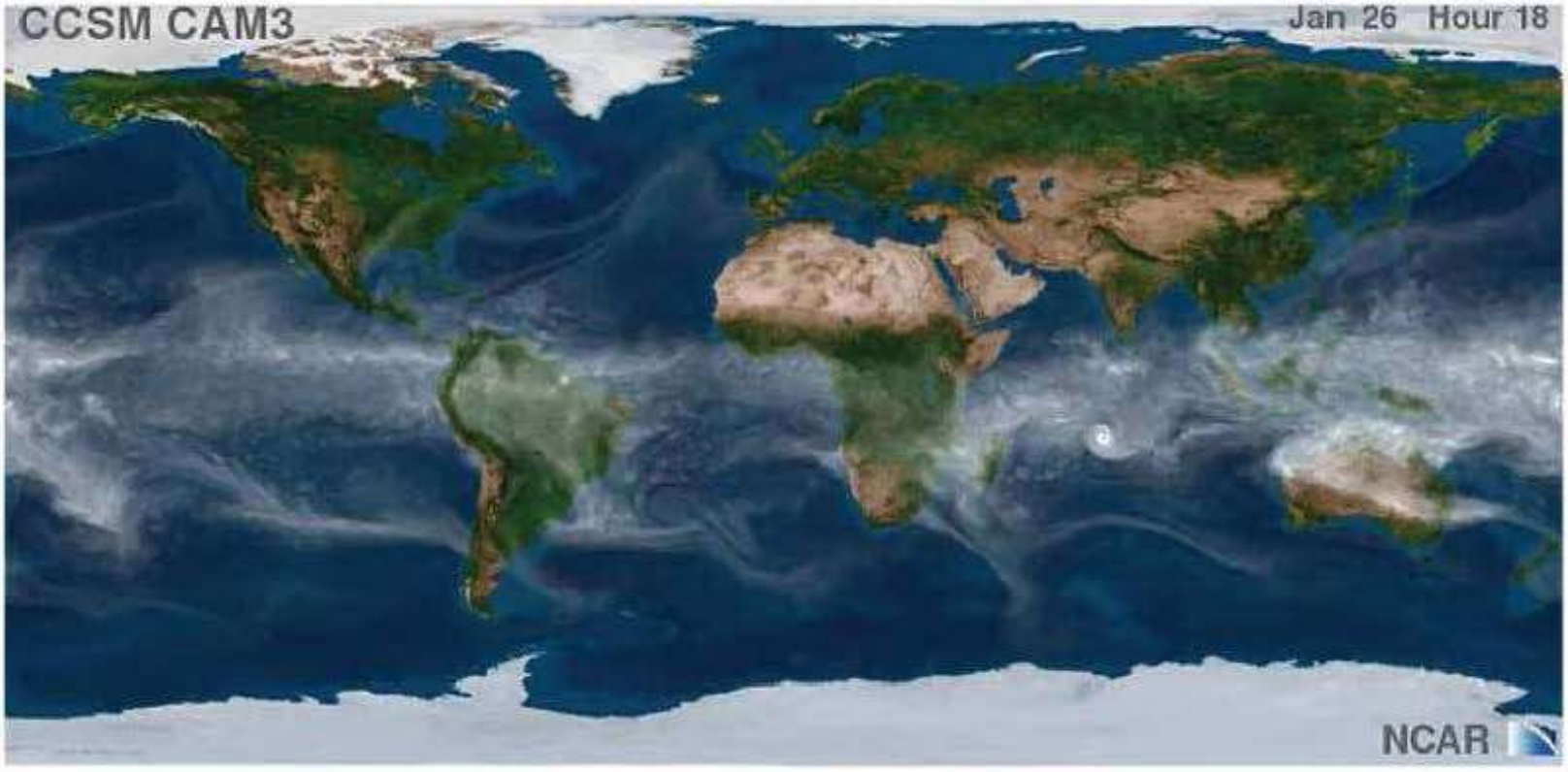
流体流动
热量传输

Scientific Data Model

Han-Wei Shen

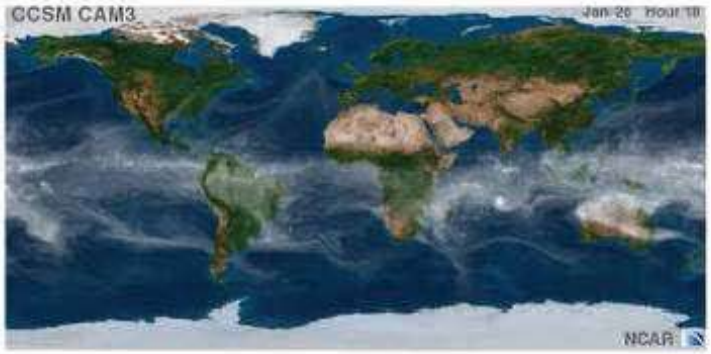
The Ohio State University

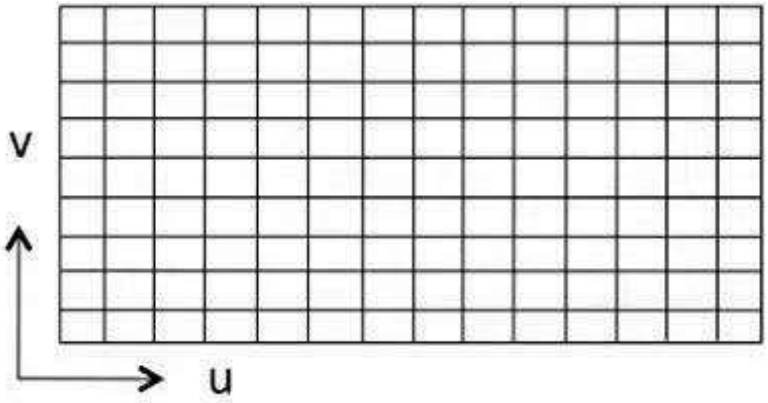
What is a Data Model?



How do you describe the data represented by this image?

Data Model



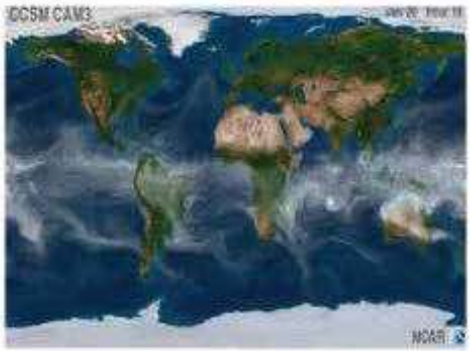


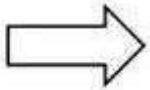
(u,v)

Temperature
Pressure
Cloud density
...

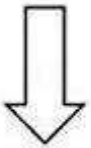
- Describe the objects represented by the data
  - Structures of the objects
  - Properties of the objects
  - Relationships between the objects

Scientific Data Model





Data Model



- Data set – a single or multiple valued function

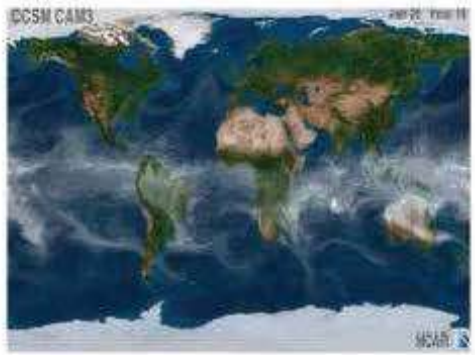
m dependent variables x_i (i=1..m)
n independent variable v_j (j = 1..n)

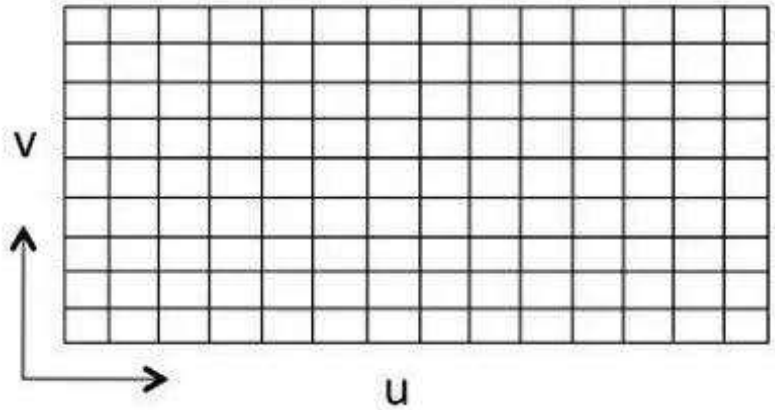
Temperature
Pressure
Cloud density
...

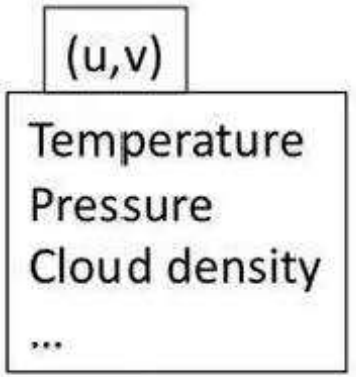
$$\mathbf{y}_1 = f_1(x_1, x_2, x_3, \ldots, x_n)$$
$$\mathbf{y}_2 = f_2(x_1, x_2, x_3, \ldots, x_n)$$
$$\vdots$$
$$\mathbf{y}_m = f_m(x_1, x_2, x_3, \ldots, x_n)$$

Each dependent variable y_i can have a tensor rank k
- k = 0 : scalar; k = 1: vector; k = 2; 2D matrix, etc.

Scientific Data Model

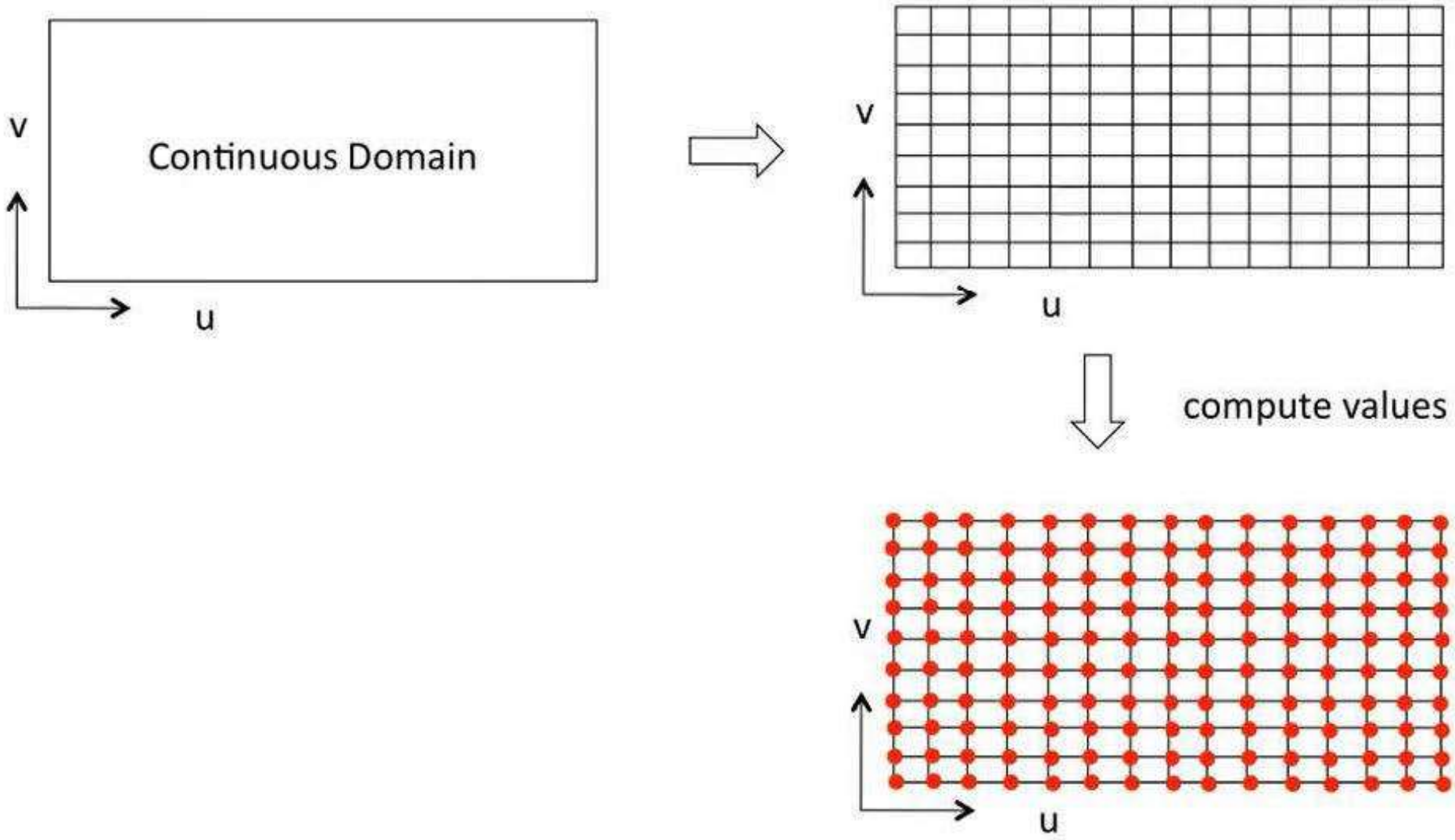




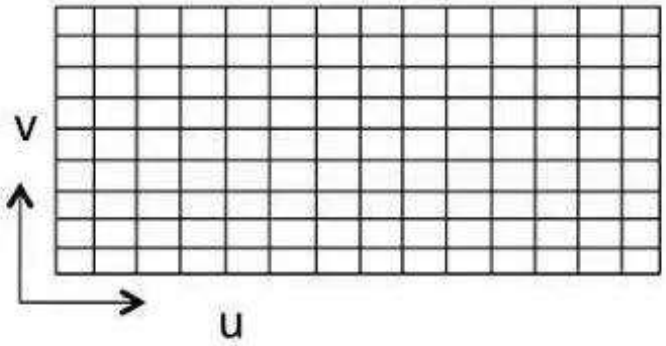


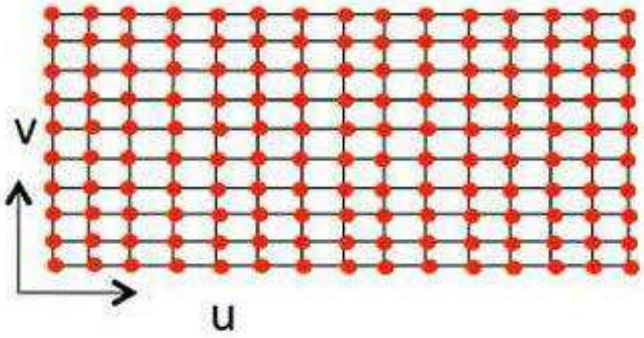
- Data set – a single or multiple valued function
  - **Independent variables** (dimensions)
    - Spatial coordinates (longitude, latitude, height)
    - Time
    - Zone ID
    - ...
  - Dimensionality - number of independent variables
  - **Dependent variables**
    - The function values of independent variables
    - The number of values associated with each dependent variable can be described by its *tensor rank*
      - 0: scalar
      - 1: vector
      - 2: n x n matrix ...

Domain Discretization



Scientific Data Set





Scientific Data Set =

Domain Structure +

Attributes

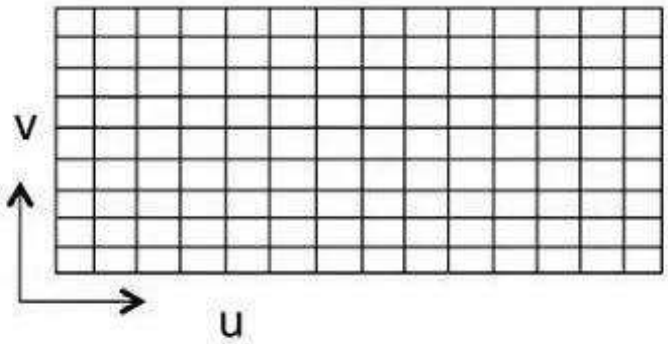
Domain Structure

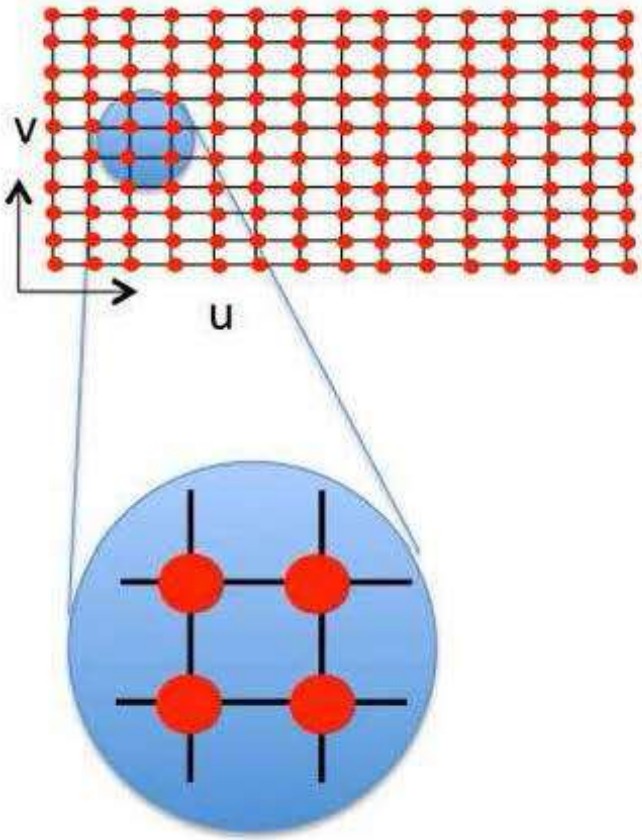
- Topology: property invariant under transformation
- Geometry: instantiation of topology with specific positions
- Consists of *Points* and *Cells*, which define the *Mesh*

Attributes

One or multiple values (scalars, vectors, tensors) defined at points or cells

Domain Structure - Cell

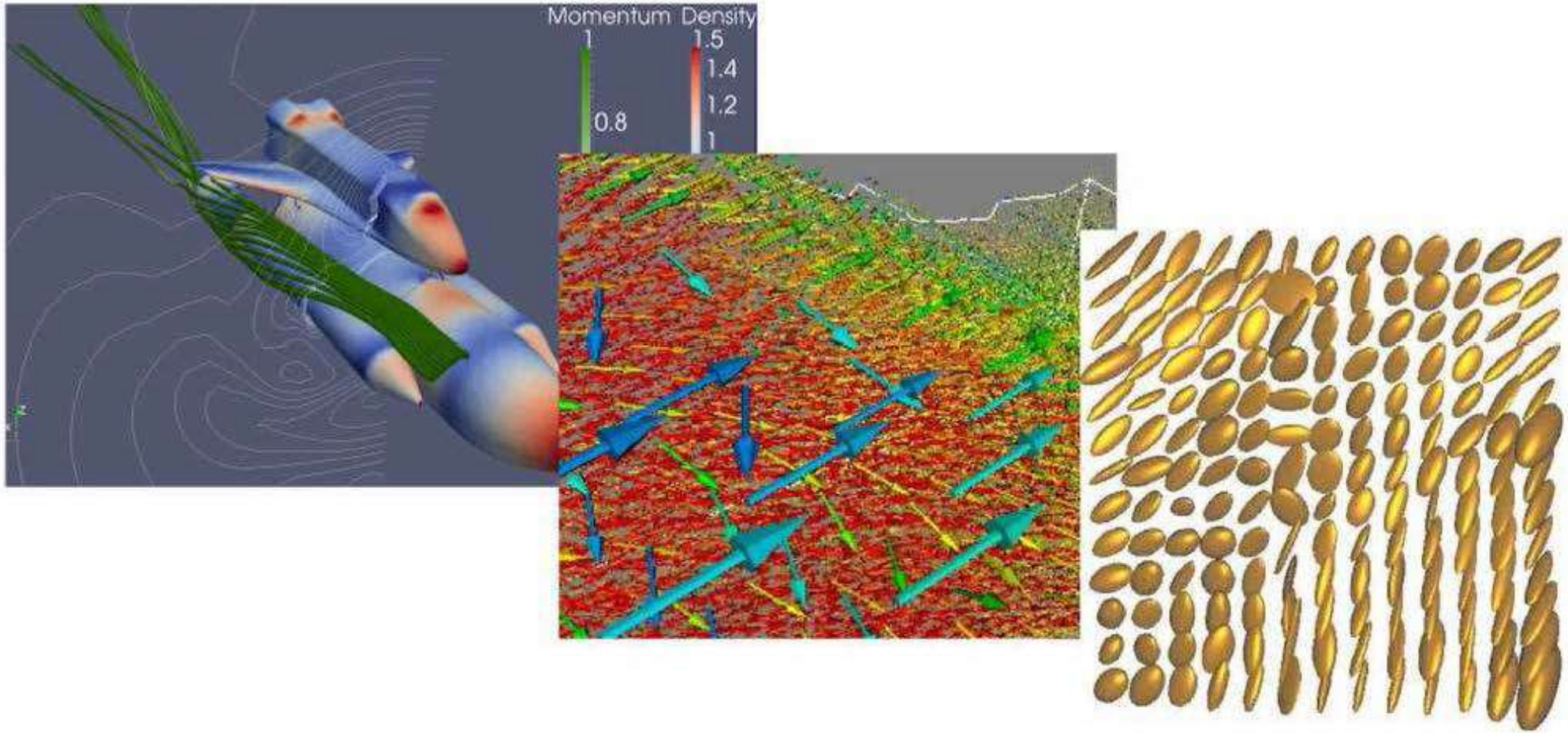




- Cells are the fundamental building blocks of scientific data sets
- Cells define how points are connected together to form the basis for interpolation
- Cells can be of different dimensionality
  - 0 D: Vertices
  - 1 D: Line; Polylines;
  - 2 D: Triangle; Quadrilateral; Polygon
  - 3 D: Tetrahedron; Hexahedron; Voxel;

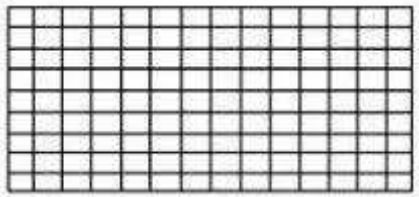
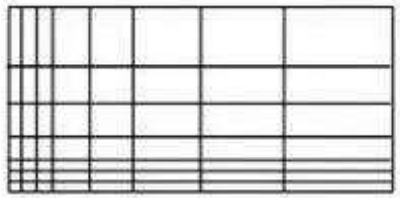
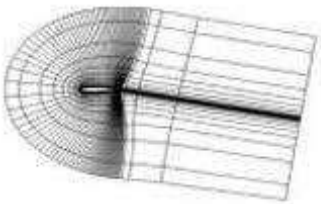
Attributes

- Scalars (e.g. density), Vectors (e.g. momentum), , Tensors (e.g. stress tensor)



Scientific Dataset Types

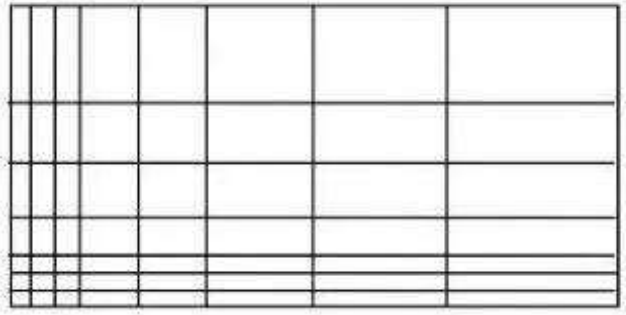
- Data sets are categorized into different types based on their underlying grid (domain structures)
  - Structured Grid

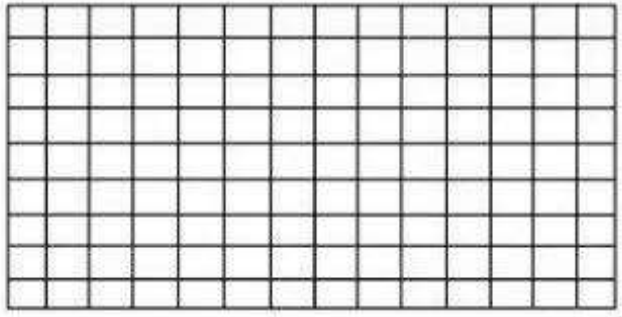
    - Consisting of a collection of points and cells arranged on a regular lattice
    - Every point in the structured grid can be indexed by (i,j) in 2D, (i,j,k) in 3D, etc.
    - The position of the points, and hence the geometry of the cells, can be either implicitly defined (Cartesian gird), or explicitly specified (rectilinear or curvilinear grid)

Scientific Dataset Types

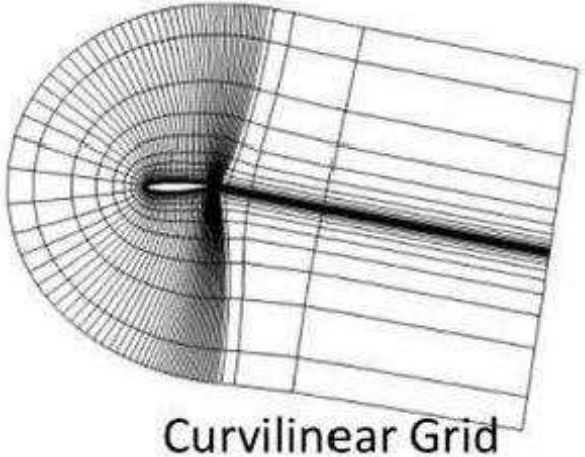
- Data sets are categorized into different types based on their underlying grid (domain structures)
  - Structured Grid
    - Cartesian mesh
    - Rectilinear mesh
    - Curvilinear mesh



Rectilinear Grid



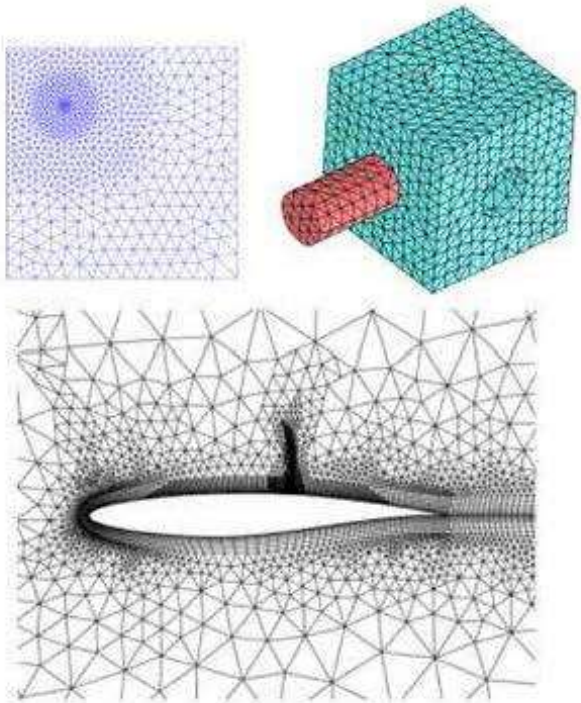
Cartesian Grid



Curvilinear Grid

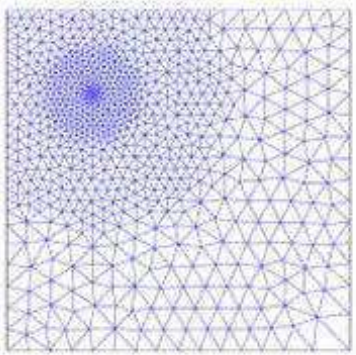
Scientific Dataset Types

- Data sets are categorized into different types based on their underlying grid (domain structure)
  - Unstructured Grid
    - Also called irregular grid data
    - Unstructured grid points are irregular located in space
    - It is often a result of space tessellation with simple shapes
    - Explicit connectivity information to form cells is necessary

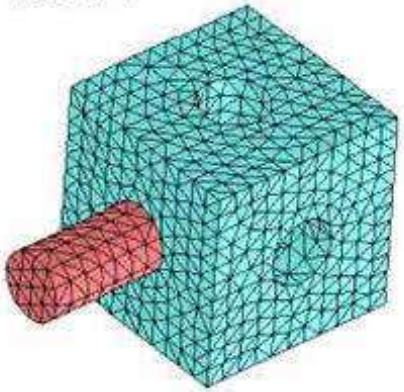


Scientific Dataset Types

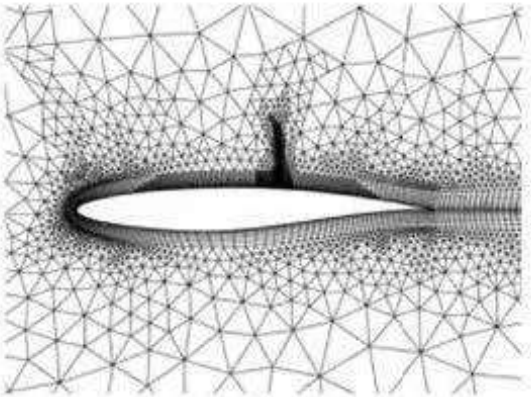
- Data sets are categorized into different types based on their underlying grid (domain structure)
  - Unstructured Grid
    - Polygonal mesh
    - Tetrahedral mesh
    - Hybrid Mesh



Polygonal mesh



Tetrahedral mesh



Hybrid mesh

Scientific Data File Formats

Han-Wei Shen

The Ohio State University

Scientific File Format

- Common scientific data file formats
  - VTK
  - NetCDF
  - Plot3D

- Similar conceptual data models, but with different
  - organizations for storing domain structures, attributes, and meta-data
  - degrees of extensibility to store one, multiple, or groups of data objects
  - Application Program Interfaces (API) to read/write the data

The Visualization Toolkit (VTK)

- The Visualization Toolkit (VTK) is an open-source, freely available software system for visualization, 3D computer graphics, and image processing
- Produced by the company Kitware (www.kitware.com)

VTK Simple Legacy Format

- Contains two different style: Legacy and XML

MyFile.vtk

File version and identifier

vtkDataFile Version x,x

Header: comment, 256 char max

This is where I put my comments \n

Type: type of data

ASCII (or BINARY)

Structure: describes domain Structure

... (more in the following slides)

Attributes

... (more in the following slides)

Specifying Domain Structure

- Specifies the structure of the domain

DATASET STRUCTURED_POINTS **or**
STRUCTURED_GRID
UNSTRUCTURED_GRID
RECTILINEAR_GRID
POLYDATA
FIELD

- Then give the information related to the type of the structure
  - Dimensions
  - Spacing
  - Coordinates
  - ...

Specifying Domain Structure

- Specifies the structure of the domain

DATASET ***STRUCTURED_POINTS*** **or**

STRUCTURED_GRID

UNSTRUCTURED_GRID

RECTILINEAR_GRID

POLYDATA

FIELD

File version and identifier
vtkDataFile Version x.x

Header: comment, 256 char max
This is where I put my comments \n

Type: type of data
ASCII (or BINARY)

Structure: describes domain Structure
... (more in the following slides)

Attributes
... (more in the following slides)

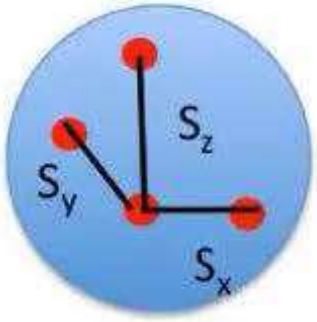
VTK Structured Point Type

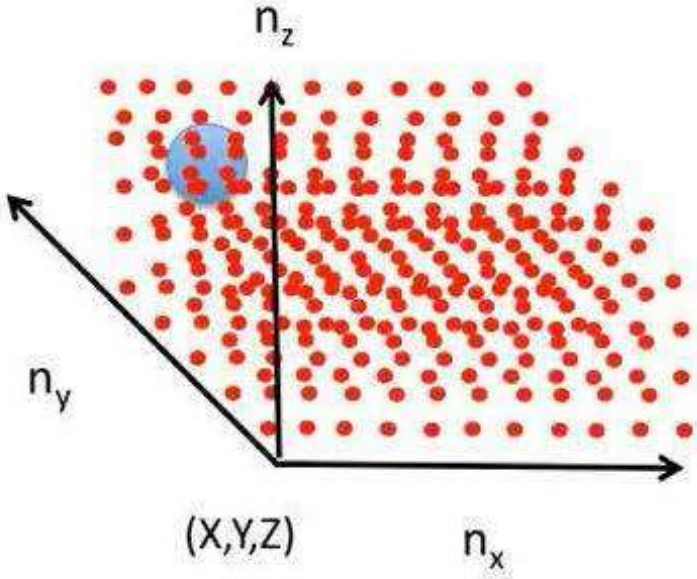
DATASET STRUCTURED_POINTS

DIMENSIONS $n_x\ n_y\ n_z$

ORIGIN $X\ Y\ Z$

SPACING $S_x\ S_y\ S_z$





Specifying Domain Structure

- Specifies the structure of the domain

DATASET STRUCTURED_POINTS **or**

STRUCTURED_GRID

UNSTRUCTURED_GRID

RECTILINEAR_GRID

POLYDATA

FIELD

File version and identifier
vtkDataFile Version x.x

Header: comment, 256 char max
This is where I put my comments \n

Type: type of data
ASCII (or BINARY)

Structure: describes domain Structure
... (more in the following slides)

Attributes
... (more in the following slides)

VTK Rectilinear Grid Type

DATASET RECTILINEAR_POINTS

DIMENSIONS $n_x\ n_y\ n_z$

X_COORDINATES n_x *dataType*

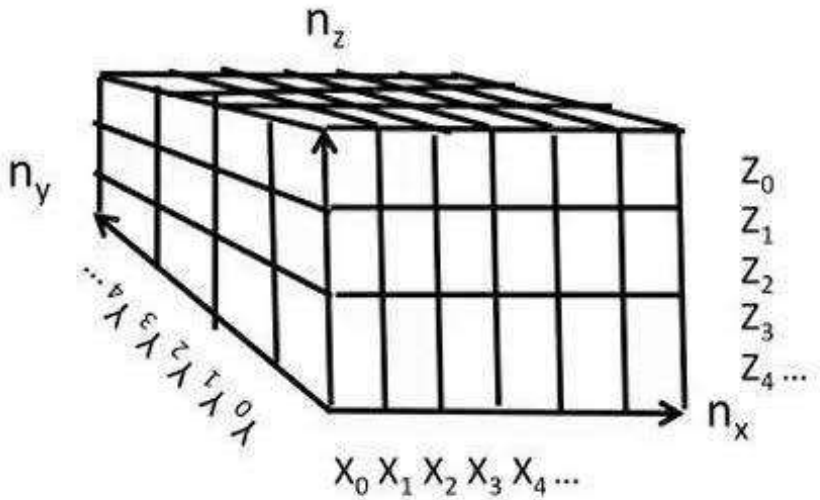
$x_0\ x_1\ \ldots\ x_{nx-1}$

Y_COORDINATES n_y *dataType*

$y_0\ y_1\ \ldots\ y_{nx-1}$

Z_COORDINATES n_z *dataType*

$z_0\ z_1\ \ldots\ z_{nx-1}$



Specifying Domain Structure

- Specifies the structure of the domain

DATASET STRUCTURED_POINTS **or**
STRUCTURED_GRID
UNSTRUCTURED_GRID
RECTILINEAR_GRID
POLYDATA
FIELD

File version and identifier
vtkDataFile Version x.x
Header: comment, 256 char max
This is where I put my comments \n
Type: type of data
ASCII (or BINARY)
Structure: describes domain Structure
... (more in the following slides)
Attributes
... (more in the following slides)

VTK Unstructured Grid Type

dataType can be *unsigned_char, char, unsigned_short, short, unsigned_int, int, unsined_long, long, float, double*

DATASET UNSTRUCTURE_GRID
POINTS *n dataType*
P_{0x} P_{0y} P_{0z}
P_{1x} P_{1y} P_{1z}
...
$P_{(n-1)x}$ $P_{(n-1)y}$ $P_{(n-1)z}$

CELLS *n size*
$numPoints_0$ i,j,k,l,.... (first cell)
$numPoints_1$ i,j,k,l ... (second cell)
...
$numPoints_{n-1}$ i,j,k,l,.... (n-1th cell)

CELL_TYPES *n*
$Type_0$
$Type_1$
$Type_2$
...
$Type_n$

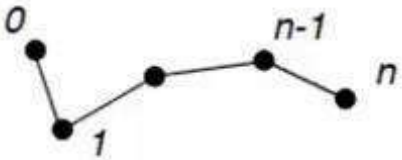
VTK Cell Types

VTK_VERTEX (=1)

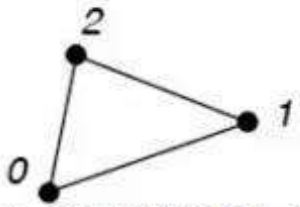
VTK_POLY_VERTEX (=2)

VTK_LINE (=3)

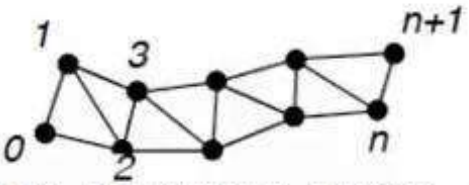
保存多边形的顶点
不如转成 triangle_strip



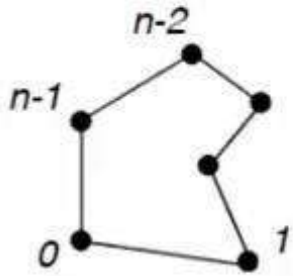
VTK_POLY_LINE (=4)



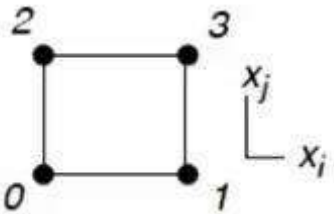
VTK_TRIANGLE(=5)



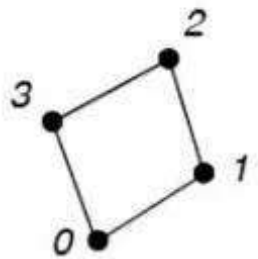
VTK_TRIANGLE_STRIP (=6)



VTK_POLYGON (=7)



VTK_PIXEL (=8)



VTK_QUAD (=9)

VTK Data Attributes

- Attributes: the values stored at the grid points
- Supported types:
  - scalars, vectors, 3x3 tensors, normals, texture coordinates (1/2/3D), and fields.
- Format:

MyFile.vtk

File version and identifier
vtkDataFile Version x.x

Header: comment, 256 char max
This is where I put my comments \n

Type: type of data
ASCII (or BINARY)

Structure: describes domain Structure
... (more in the following slides)

Attributes
... (more in the following slides)

attributeType	dataName	dataType	attributeValues
SCALARS VECTORS TENSORS NORMALS TEXTURE_COORDINATES	Any name you like	unsigned_char char unsigned_short short unsigned_int int unsined_long long float double	Described in the following slides

VTK Data Attributes

- Attributes: the values stored at the grid points
- Supported types:
  - scalars, vectors, normals, texture coordinates (1/2/3D), 3x3 tensors, and fields.
- Scalar Attribute

MyFile.vtk

File version and identifier
vtkDataFile Version x.x

Header: comment, 256 char max
This is where I put my comments \n

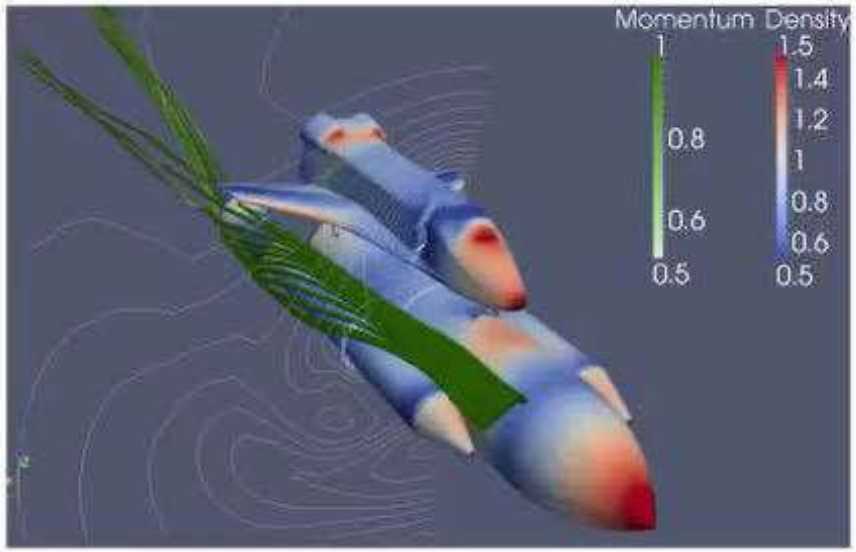
Type: type of data
ASCII (or BINARY)

Structure: describes domain Structure
... (more in the following slides)

Attributes
... (more in the following slides)

Optional, default = 1

SCALARS *dataName* *dataType* *numComp*
LOOKUP_TABLE *default*
S_0
S_1
...
S_{n-1}



VTK Data Attributes

- Attributes: the values stored at the grid points
- Supported types:
  - scalars (with optional RGB lookup table) vectors, normals, texture coordinates (1/2/3D), 3x3 tensors, and fields.

MyFile.vtk

File version and identifier
vtkDataFile Version x.x

Header: comment, 256 char max
This is where I put my comments \n

Type: type of data
ASCII (or BINARY)

Structure: describes domain Structure
... (more in the following slides)

Attributes
... (more in the following slides)

- Vector attribute

VECTORS dataName dataType

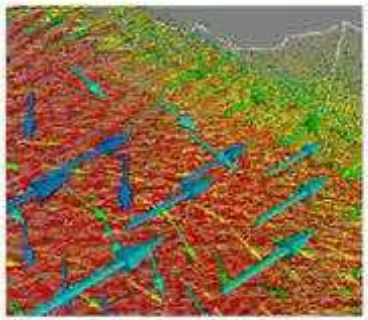
v_{0x} v_{0y} v_{0z}
v_{1x} v_{1y} v_{1z}
...
$v_{(n-1)x}$ $v_{(n-1)y}$ $v_{(n-1)z}$

- Tensors attribute

TENSORS *dataName dataType*

t_{00} t_{01} t_{02}
t_{10} t_{11} t_{12}
t_{20} t_{21} t_{22}

n 3 x 3 tensors



VTK Data Attributes

- Attributes: the values stored at the grid points
- Supported types:
  - scalars (with optional RGB lookup table) vectors, normals, texture coordinates (1/2/3D), 3x3 tensors, and fields.

MyFile.vtk

File version and identifier
vtkDataFile Version x.x

Header: comment, 256 char max
This is where I put my comments \n

Type: type of data
ASCII (or BINARY)

Structure: describes domain Structure
... (more in the following slides)

Attributes
... (more in the following slides)

- Normals

NORMALS *dataName dataType*

$N_{0x}\ n_{0y}\ n_{0z}$

$n_{1x}\ n_{1y}\ n_{1z}$

...

$n_{(n-1)x}\ n_{(n-1)y}\ n_{(n-1)z}$

- Texture Coordinates

TEXTURE_COORDINDATES *dataName dim dataType*

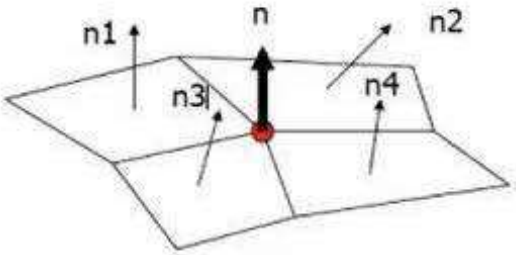
$t_{00}\ t_{01}\ t_{0(dim-1)}$

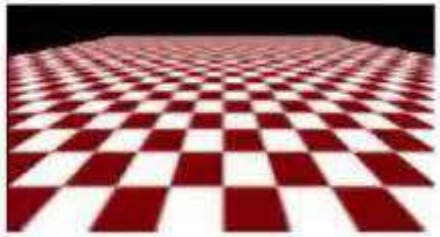
$t_{10}\ t_{11}\ t_{1(dim-1)}$

...

$T_{(n-1)0}\ t_{(n-1)1}\ t_{(n-1)(dim-1)}$

Dimensions of the texture





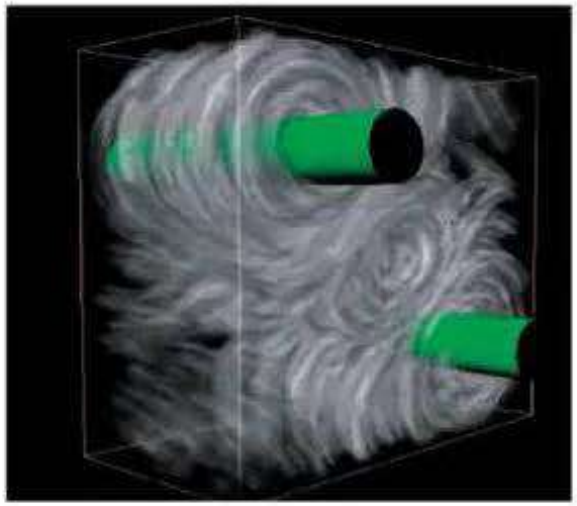
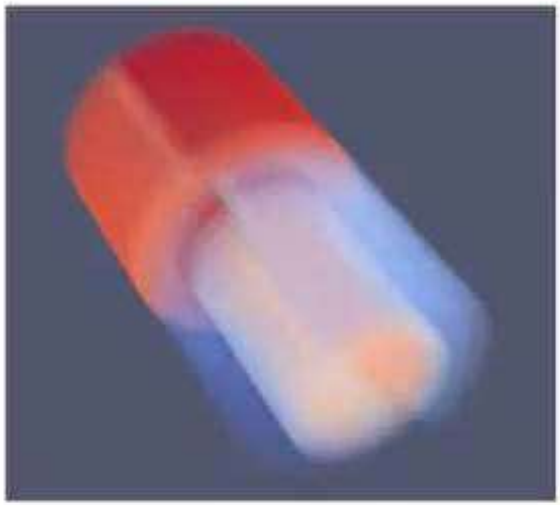
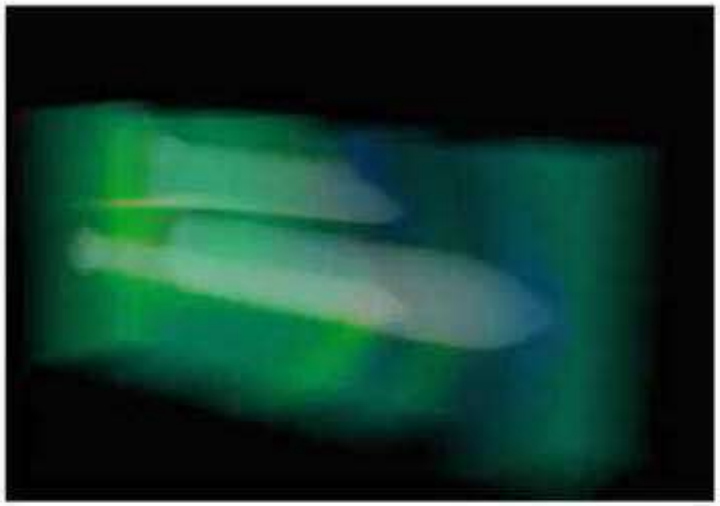
Direct Volume Rendering
Optical Model

Han-Wei Shen

The Ohio State University

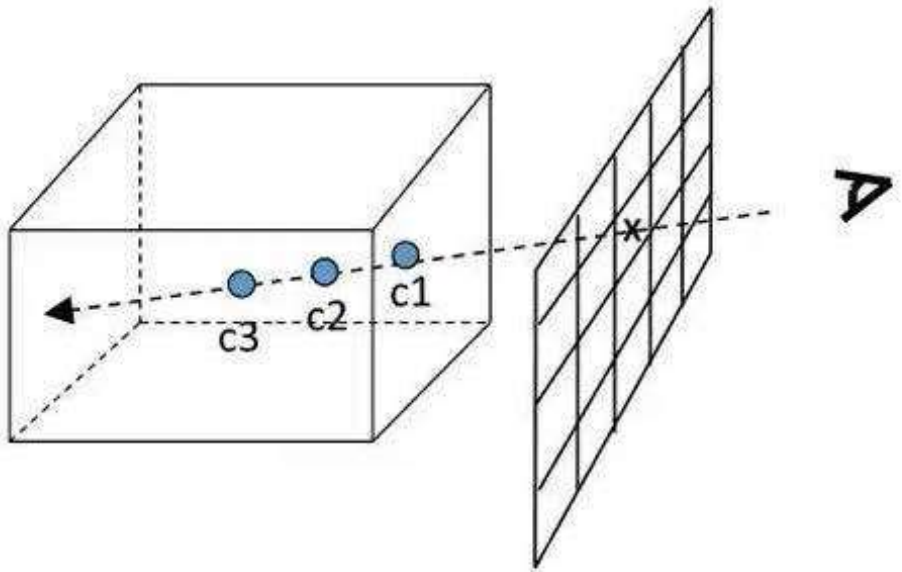
Volume Rendering

- A method to visualize the entire 3D data set by simulating light transport across the volume
- A 2D projection of 3D discrete samples

Direct Volume Rendering

- Simulate light transport through a continuous volume
- Data are interpolated from the samples at the grid points
- Optical properties such as colors and opacities are assigned to the interpolated data
- Optical properties must be integrated along each viewing ray

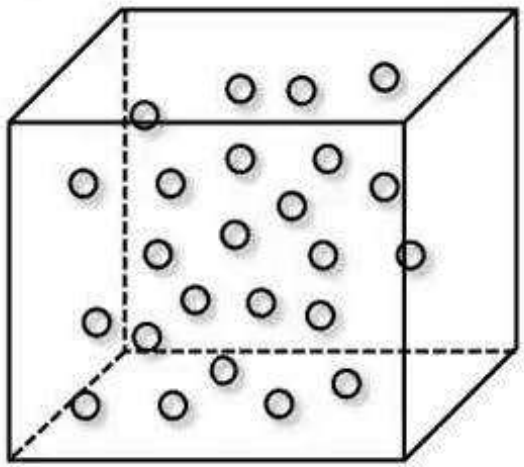


Optical Model

Participating medium (the voxels) can absorb, emit, or both absorb and emit light

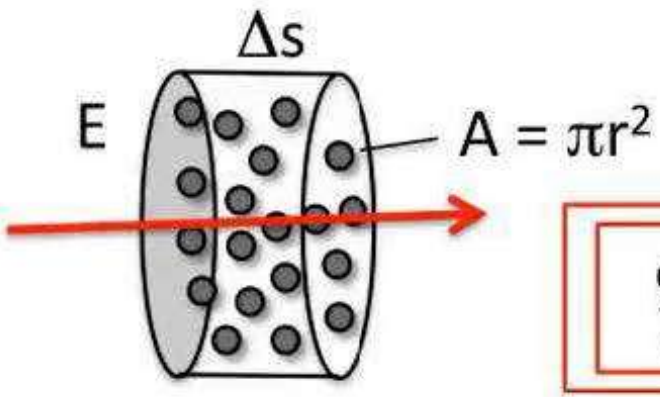
How much light will reach the eye?

- Absorption Model
- Emission Model
- Absorption + Emission



Absorption Only

- The simplest participating medium
- Consists of perfectly black particles that absorb all the light that they intercept
- Assume
  - each particle has an area of $A = \pi r^2$
  - Number of particles per unit volume = ρ
  - A small cylindrical slab with a base area E and thickness Δs

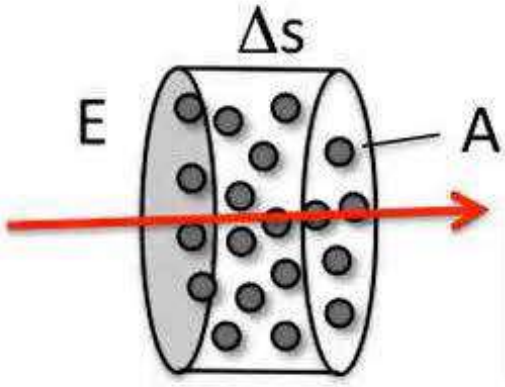


- Total number of particles = $E\,\Delta s\,\rho$
- Total area occluded by particles = $A\,E\,\Delta s\,\rho$
- The fraction of occluded area = $AE\Delta s\rho / E = A\,\Delta s\,\rho$

$$\frac{d\,\mathbf{I(s)}}{ds} = -\,I(s) \times \frac{A\,\Delta s\,\rho(s)}{\Delta s} = -\,A\,\rho\,(s)\,\mathbf{I}(s)$$

Absorption Only

- The simplest participating medium
- Consists of perfectly black particles that absorb all the light that they intercept
- Assume
  - each particle has an area of $A = \pi r^2$
  - Number of particles per unit volume = ρ
  - A small cylindrical slab with a base area E and thickness Δs

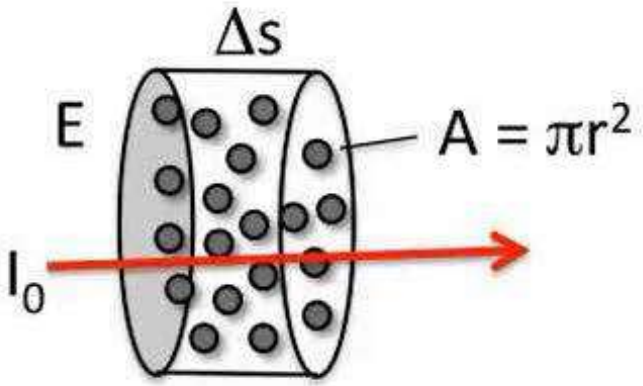


$$\frac{dI}{ds} = -A\,\rho(s)\,I(s)$$

Solve this ODE

$$I(s) = I_0 \times e^{-\int_0^s \rho(t) A dt} = I_0 \times e^{-\int_0^s \tau(t) dt}$$

Absorption Only



$$I(s) = I_0 \times e^{-\int_0^s \rho(t)Adt} = I_0 \times e^{-\int_0^s \tau(t)dt}$$

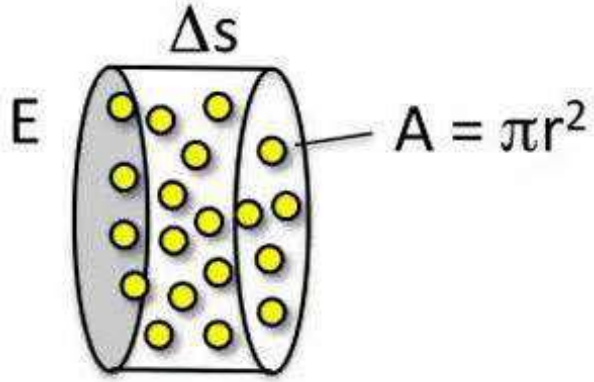
$\rho(t)A = \tau(t)$: extinction coefficient

I_0 = Initial light intensity

$e^{-\int_0^s \tau(t)dt}$ ← Can be seen as the transparency, or 1 – opacity of the medium from 0 to s

Emission Only

- Each particle will glow diffusively with an intensity C
- In a small cylindrical slab, the total area occupied by the particles is $AE\Delta s\rho$
- So the glow flux will be $CAE\Delta s\rho$
- So the glow per unit area is $CAE\Delta s\rho / E = CA\Delta s\rho$



$$\frac{dI}{ds} = C(s)\, A\, \rho(s) = C(s)\tau(s) = g(s)$$

→ Solve this ODE

$$I(s) = I_0 + \int_0^s g(t)dt = I_0 + \int_0^s C(t)\tau(t)dt$$

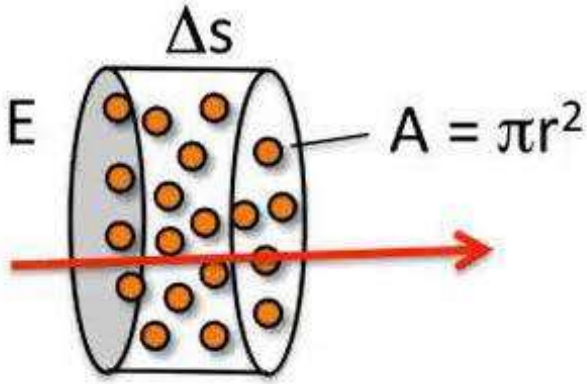
Emission plus Absorption

- Simply add emission and absorption together

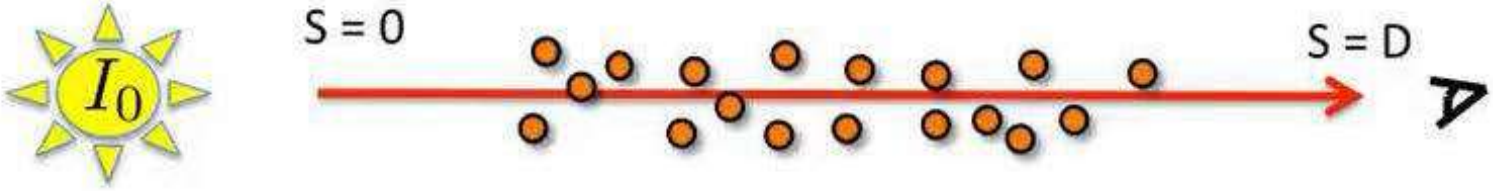
$$\frac{dI}{ds} = C(s)\tau(s) - A\,\rho\,(s)\,I(s) = g(s) - \tau(s)I(s)$$

emission absorption

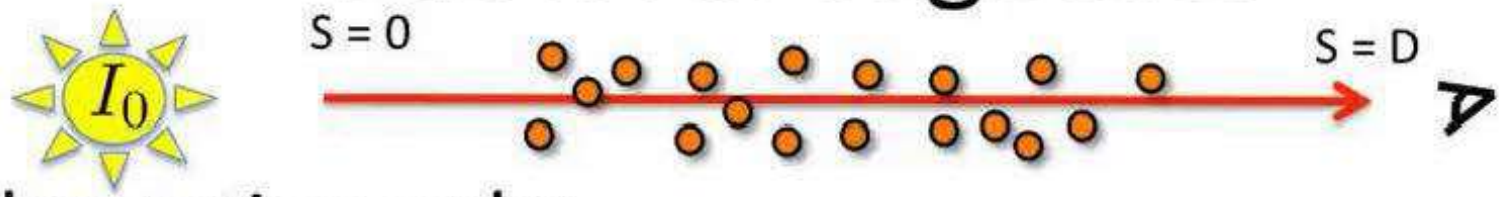
Solve this ODE



$$I(D) = I_0 \times e^{-\int_0^D \tau(t)dt} + \int_0^D g(s)e^{-\int_s^D \tau(t)dt}ds$$



Put It All Together



- Absorption only:

$$I(s) = I_0 \times e^{-\int_0^s \rho(t) A dt} = I_0 \times e^{-\int_0^s \tau(t) dt}$$

- Emission only:

$$I(s) = I_0 + \int_0^s g(t)dt = I_0 + \int_0^s C(t)\tau(t)dt$$

- Emission plus absorption:

$$I(D) = I_0 \times e^{-\int_0^D \tau(t)dt} + \int_0^D g(s) e^{-\int_s^D \tau(t)dt} ds$$

Look More Closely

$$I(D) = I_0 \times e^{-\int_0^D \tau(t)dt} + \int_0^D g(s) e^{-\int_s^D \tau(t)dt} ds$$

I_0 : background light

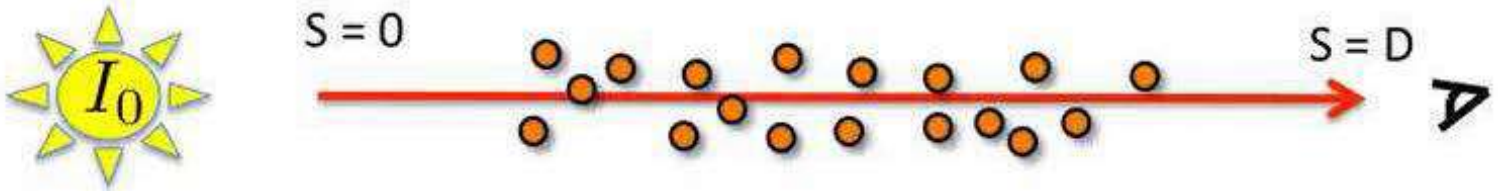
$\tau(t)$: extinction coefficient at t , related to the rate that light is occluded

D : total distance light will travel

$e^{-\int_0^D \tau(t)dt}$: transparency of medium between 0 and D

$1 - e^{-\int_0^D \tau(t)dt} = \alpha$: opacity of medium between 0 and D

$g(s)$: source term at point s, typically derived from the data value



Direct Volume Rendering

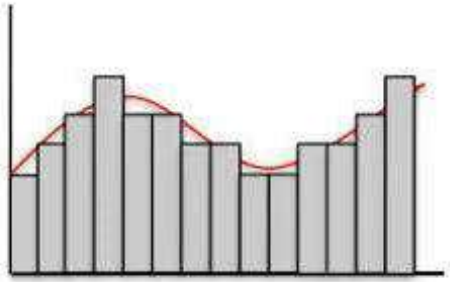
Discrete Implementation

Han-Wei Shen

The Ohio State University

Discrete Implementation

- Numerical integration:



$$\int_0^D h(x)dx = \sum_{i=1}^{i=n} h(x_i)\Delta x$$

(Riemann Sum)

$$e^{-\int_0^D \tau(t)dt} = e^{-\sum_{i=1}^{i=n} \tau(t_i)\Delta t}$$

$$= e^{-\sum_{i=1}^{i=n} \tau(i\Delta x)\Delta x} = \prod e^{-\tau(i\Delta x)\Delta x}$$

$$= \prod_{i=1}^{i=n}(1-\alpha_i)$$

(Remember $1 - e^{-\int_0^D \tau(t)dt} = \alpha$)

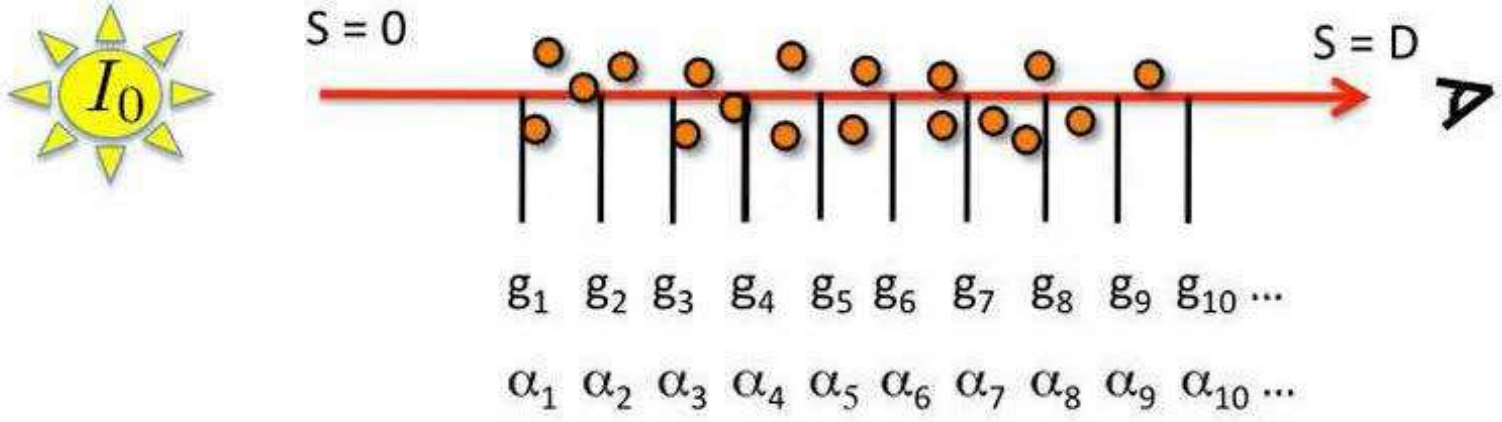
Discrete Implementation

$$I(D) = I_0 \times e^{-\int_0^D \tau(t)dt} + \int_0^D g(s) e^{-\int_s^D \tau(t)dt} ds$$

$$I_0 \prod_{i=1}^{i=n} (1 - \alpha_i) \qquad \sum_{i=1}^{i=n} g_i \times \prod_{j=i+1}^{n} (1 - \alpha_i)$$

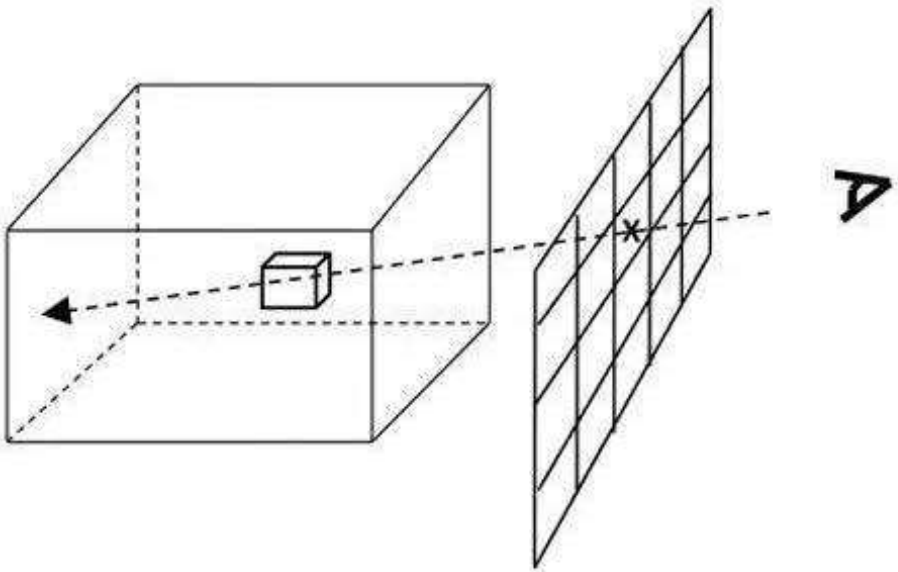
$= g_n + (1-\alpha_n)(g_n\text{-}1+(1-\alpha_{n-1})(g_{n-2}+ (1-\alpha_{n-2}(\ldots. (1-\alpha_2)(g_2+(1-\alpha_1)(g_1+I_0)))))\ldots)))))$

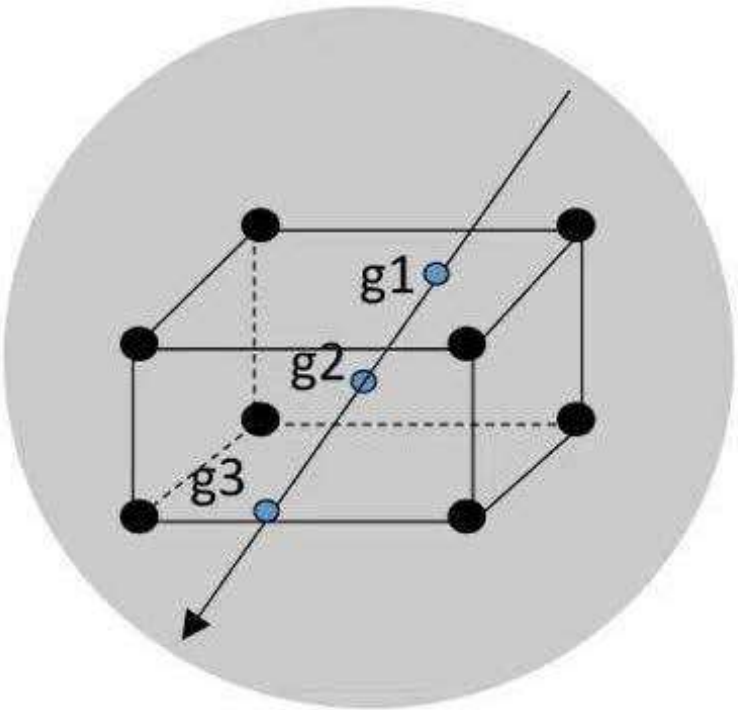
This is called - Back to Front Compositing



Ray Casting Algorithm

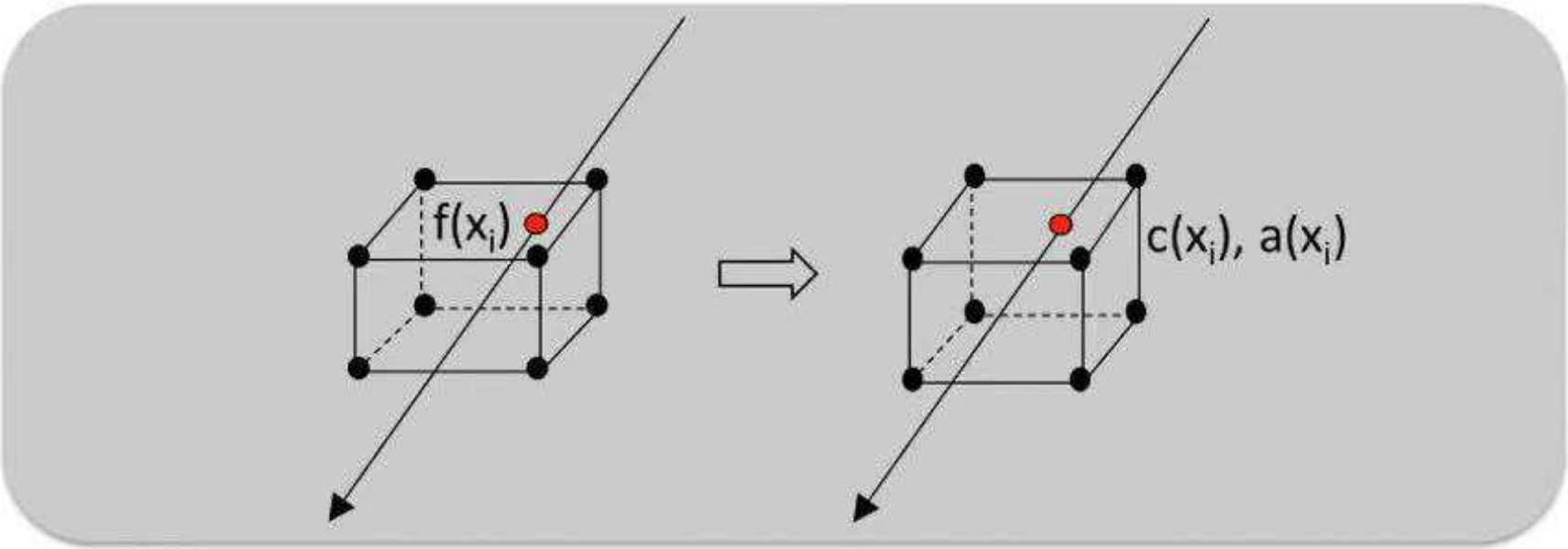
- For each pixel
  - Cast a ray into the volume
  - Linearly interpolate data values from cell (voxel) corners
  - Convert the data values to optical properties (color and opacity)
  - Composite the optical properties
  - Return the final color





Shading and Classificaton

- Shading: computer a color for every sample in the volume
- Classification: computer an opacity for every sample in the volume



- This is often done through a table (transfer function) lookup

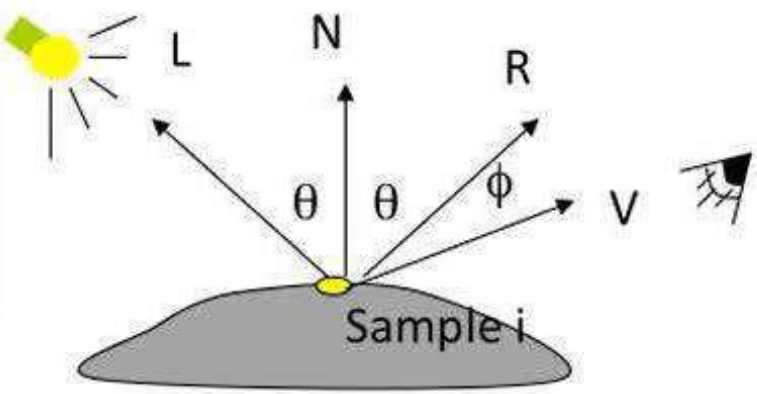
Shading

- Use the Phong illumination model

  illumination = ambient + diffuse + specular

  $= \boxed{C(x_i) \times I_a} + \boxed{C(x_i) \times I_d \times (N.L)} + \boxed{C(x_i) \times I_s \times (R.V)^n}$

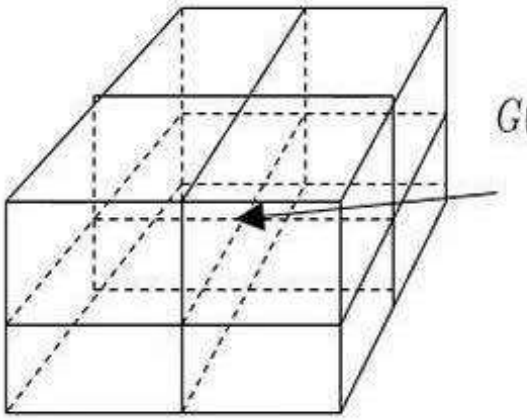
$C(x_i)$: color of sample i
I_a, I_d, I_s: light's ambient, diffuse, and specular colors (usually set as white)
N: normal at sample i
V: vector from sample point to eye
L: light vector, from sample to light source
R: reflection vector of light vector
n: shininess



Normal Estimation

- How to compute the sample normal N?
  - Normal: a vector that is perpendicular to the local surface, which is the gradient of the sample point

1. Compute the gradient G at the cell corners using *central difference*
2. Linearly interpolate the gradients



$$G(x,y,z) = (\frac{f(x+1,y,z) - f(x-1,y,z)}{2}, \frac{f(x,y+1,z) - f(x,y-1,z)}{2}, \frac{f(x,y,z+1) - f(x,y,z-1)}{2})$$

Classification

- Classification: mapping from data values to opacities

$$I(D) = I_0 \times e^{-\int_0^D \tau(t)dt} + \int_0^D g(s) e^{-\int_s^D \tau(t)dt} ds$$

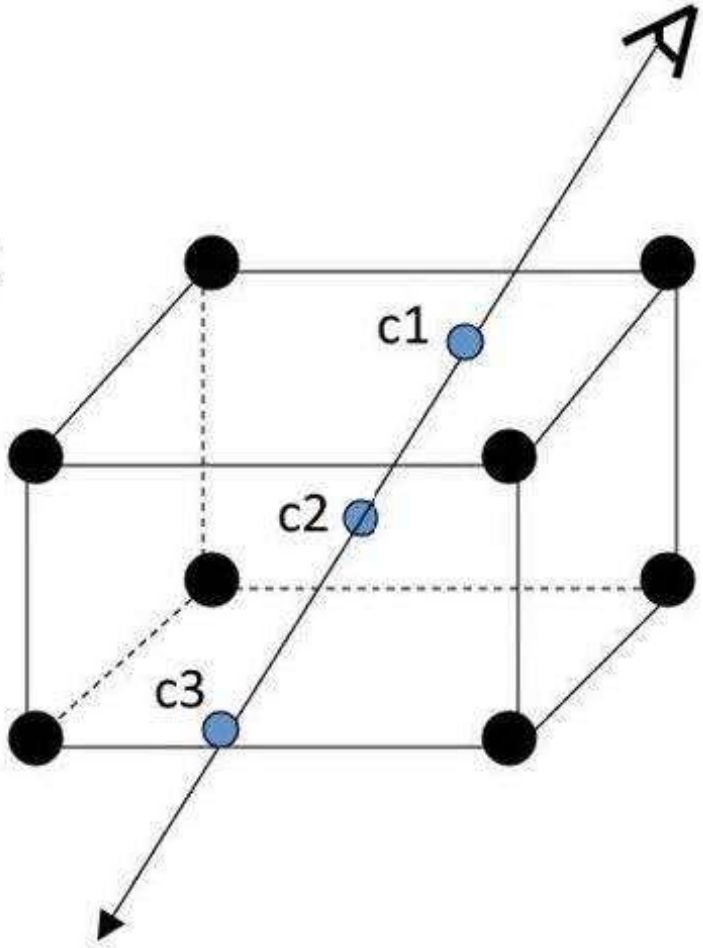
$I_0 \prod_{i=1}^{i=n} (1 - \alpha_i)$

$\sum_{i=1}^{i=n} g_i \times \prod_{j=i+1}^{n} (1 - \alpha_i)$

  - Region of interest: high opacity
  - Rest: translucent or transparent
- The opacity function, or called transfer function, is given by the user

Ray Sampling

- Sample the volume at discrete points along the ray
- Perform tri-linear interpolation to get the sample values
- Look up the transfer function to get the color and opacity
- Compositing the color/opacity (front-to-back or back-to-front)

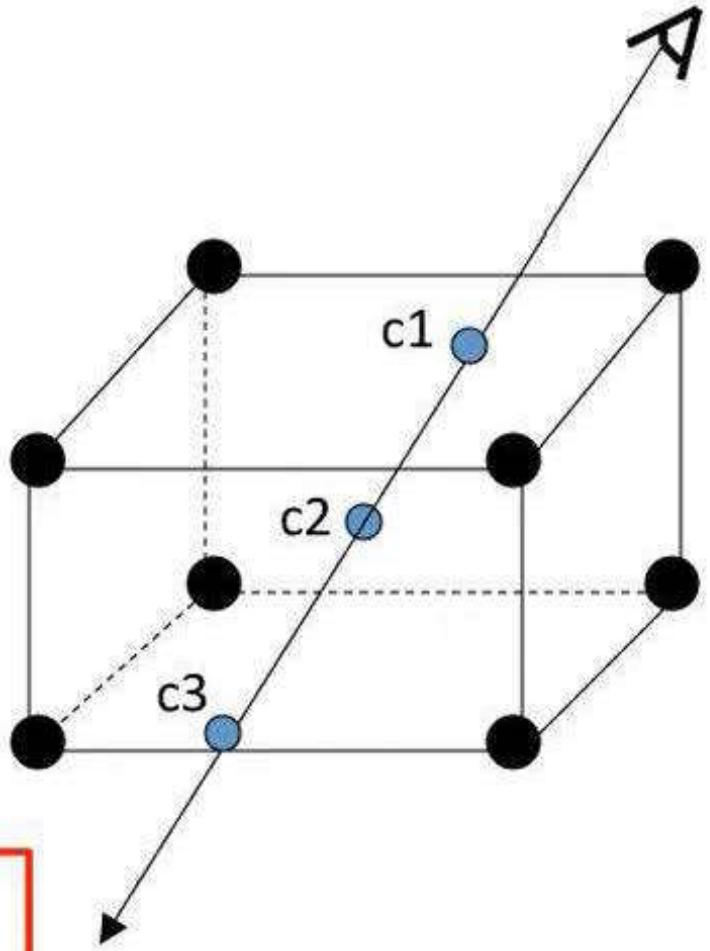


Back-to-Front Compositing

The initial pixel color = Black

Back-to-Front compositing:
use 'under' operator

C = C1 'under' background
C = C2 'under C
C = C3 'under C
...



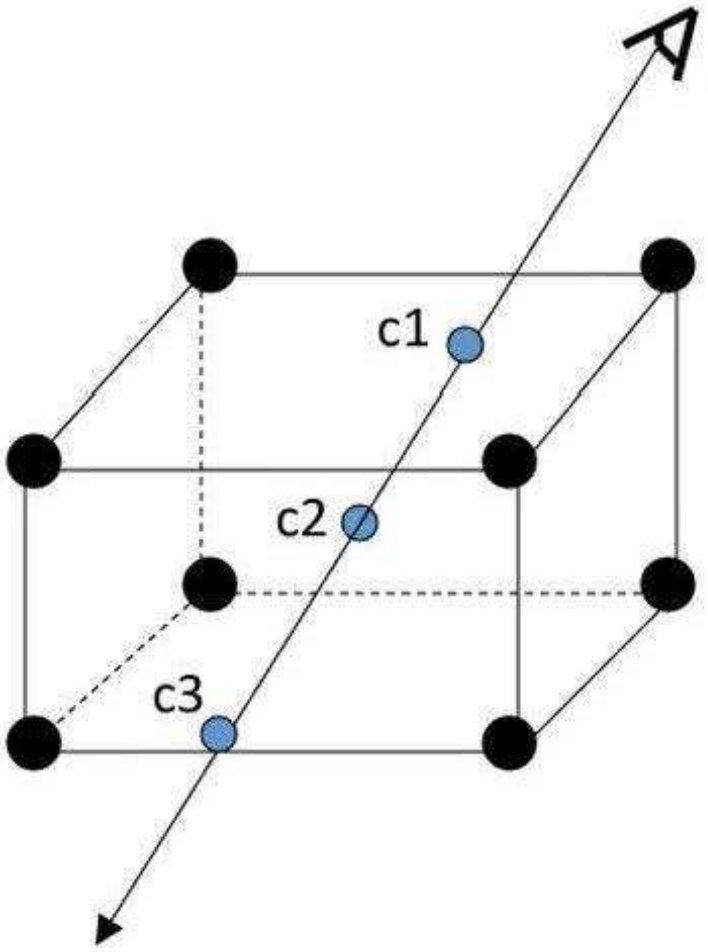
$C_{out} = C_{in} * (1-\alpha(x)) + C(x)*\alpha(x)$
(this is the alpha blending formula)

Front-to-Back Compositing

Front-to-Back compositing:
use 'over' operator

C = background 'over' C1
C = C 'over' C2
C = C 'over' C3
...

$$C_{out} = C_{in} + C(x)\,\alpha(x) * (1 - \alpha_{in});$$
$$\alpha_{out} = \alpha_{in} + \alpha(x) * (1 - \alpha_{in})$$



Direct Volume Rendering

Transfer Function Design

Han-Wei Shen

The Ohio State University

Transfer Function

- Map a data sample to color and opacity

$$f_{color}(s) = (r, g, b)$$

$$f_{opacity}(s) = \alpha$$

- The sample could be
  - A single value (scalar)
  - Multiple values (scalar, gradient magnitude, etc)

Transfer Function in Rendering Equation

$$I(D) = I_0 \times e^{-\int_0^D \tau(t)dt} + \int_0^D g(s) e^{-\int_s^D \tau(t)dt} ds$$

$$I_0 \prod_{i=1}^{i=n} (1 - \alpha_i)$$

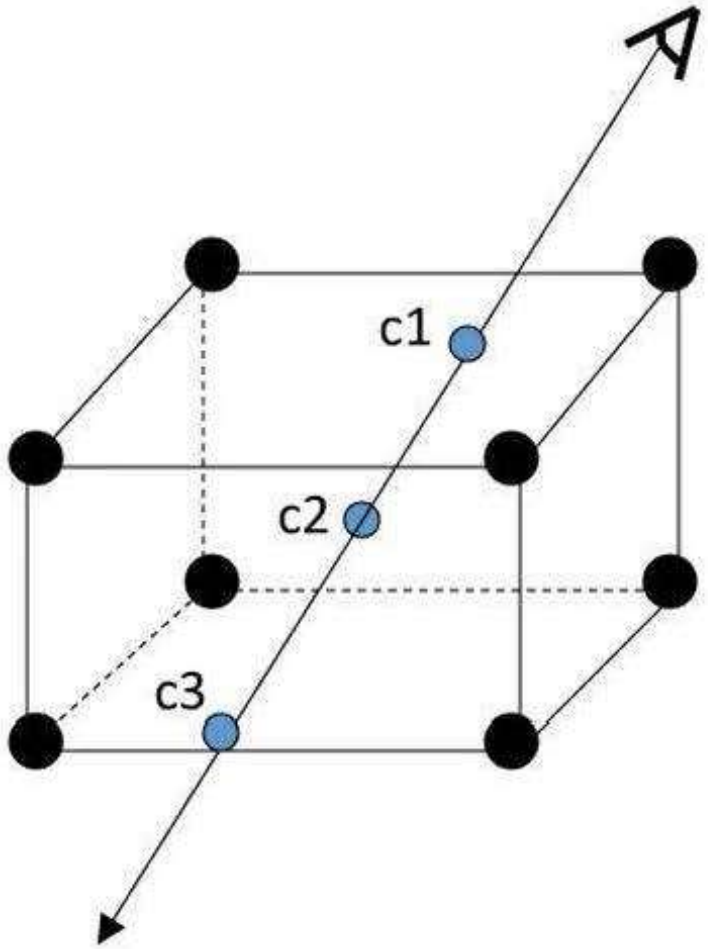
$$\sum_{i=1}^{i=n} (g_i \times \prod_{j=i+1}^{n} (1 - \alpha_i))$$

Transfer Function in Front-to-Back Compositing

Front-to-Back compositing:
use ‘over’ operator

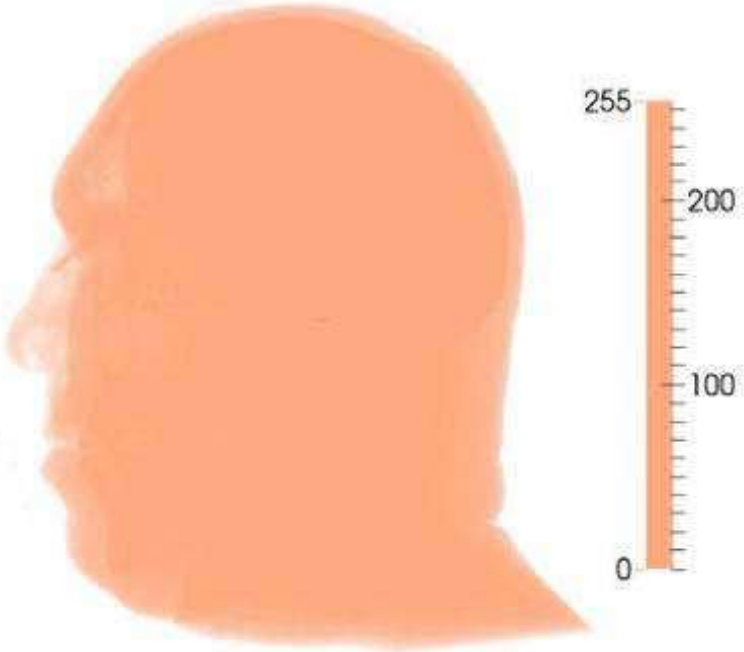
C = background ‘over’ C1
C = C ‘over’ C2
C = C ‘over’ C3
…

$$C_{out} = C_{in} + C(x)*(1- \alpha_{in});$$
$$\alpha_{out} = \alpha_{in} + \alpha(x) *(1-\alpha_{in})$$

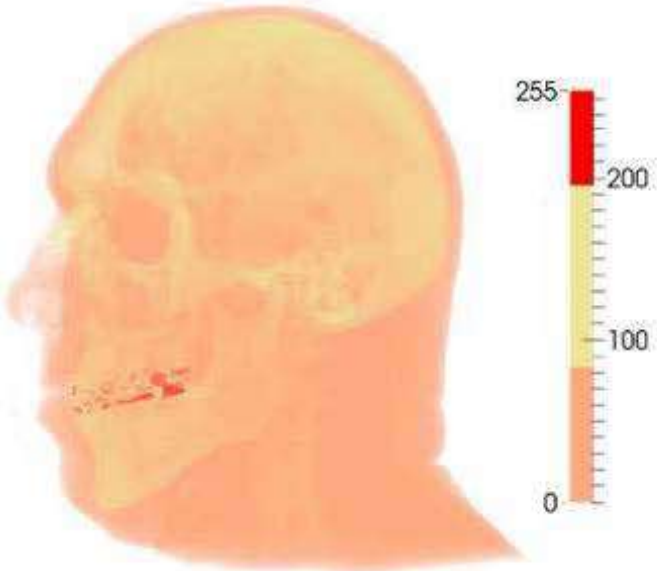


Color in Transfer Function

- Color
  - Distinguish different materials



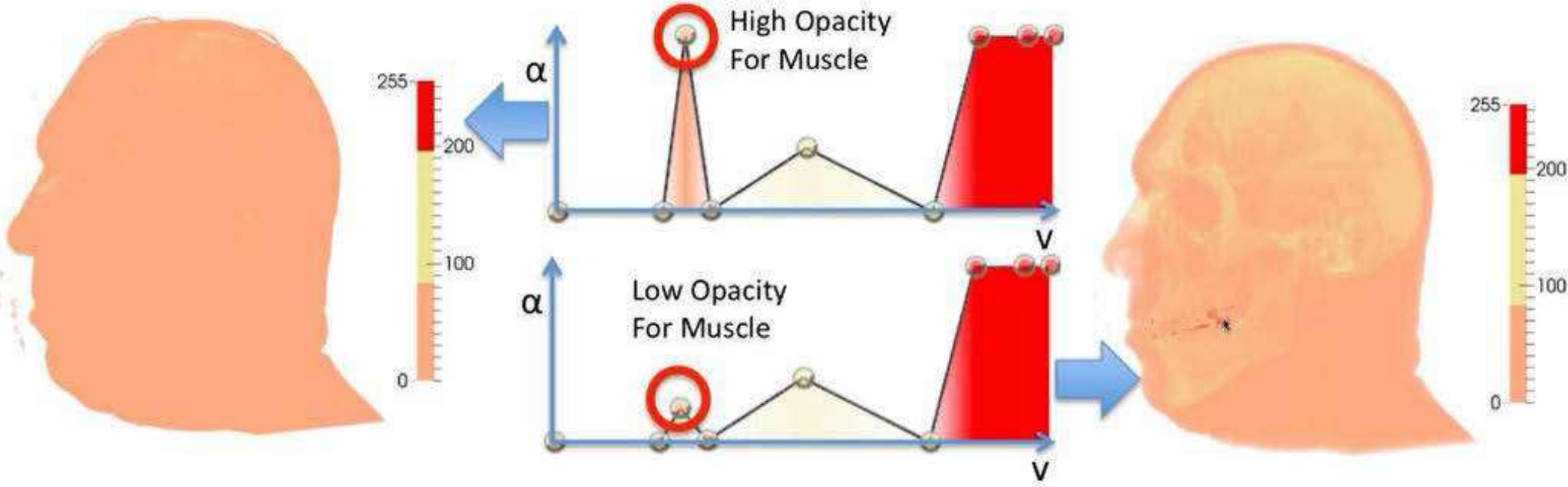
- All Materials uses same color
- See no features



- Muscle: orange
- Bone: yellow
- Tooth: red

Opacity in Transfer Function

- Opacity
  - Opacity (transparency) of each sample
  - That multiple materials is shown in the rendered image provides more context



- Material inside the muscle is occlude
- See no features

- Make the muscle transparent
- See the bone and tooth

Transfer Function Design

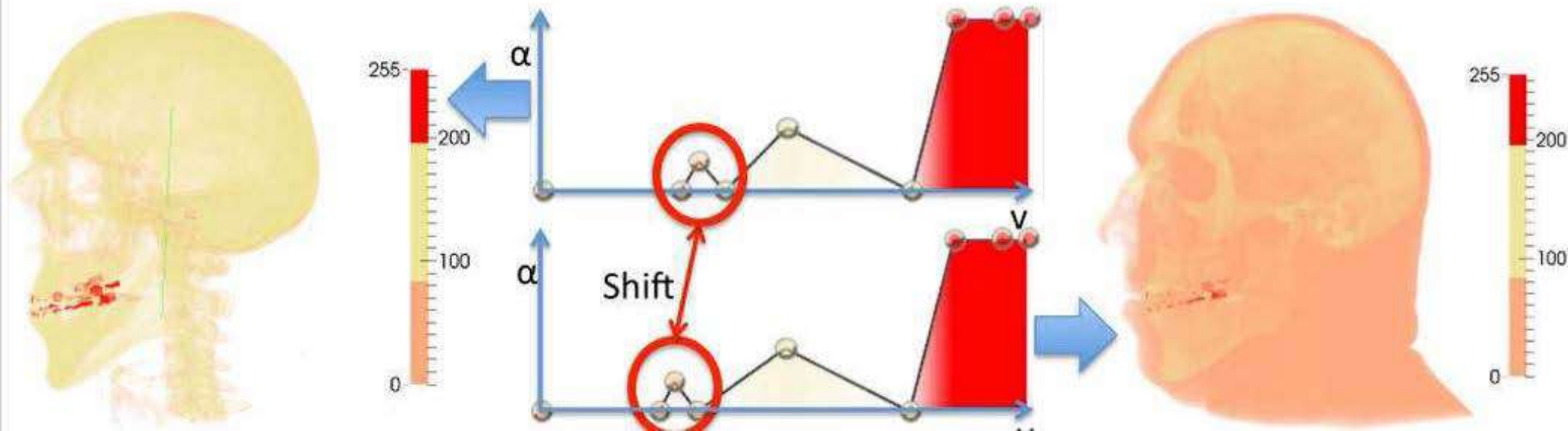
- Goal
  - Using transfer function to emphasize salient structures and de-emphasize other
- Challenges
  - Without knowing the data how to design a good transfer function?
  - A small difference in the transfer function could produce very different images
  - Some features are not easy to show without a lot of tweaking

Need algorithms and strategies to assist users to find the desired transfer function in a huge transfer function search space

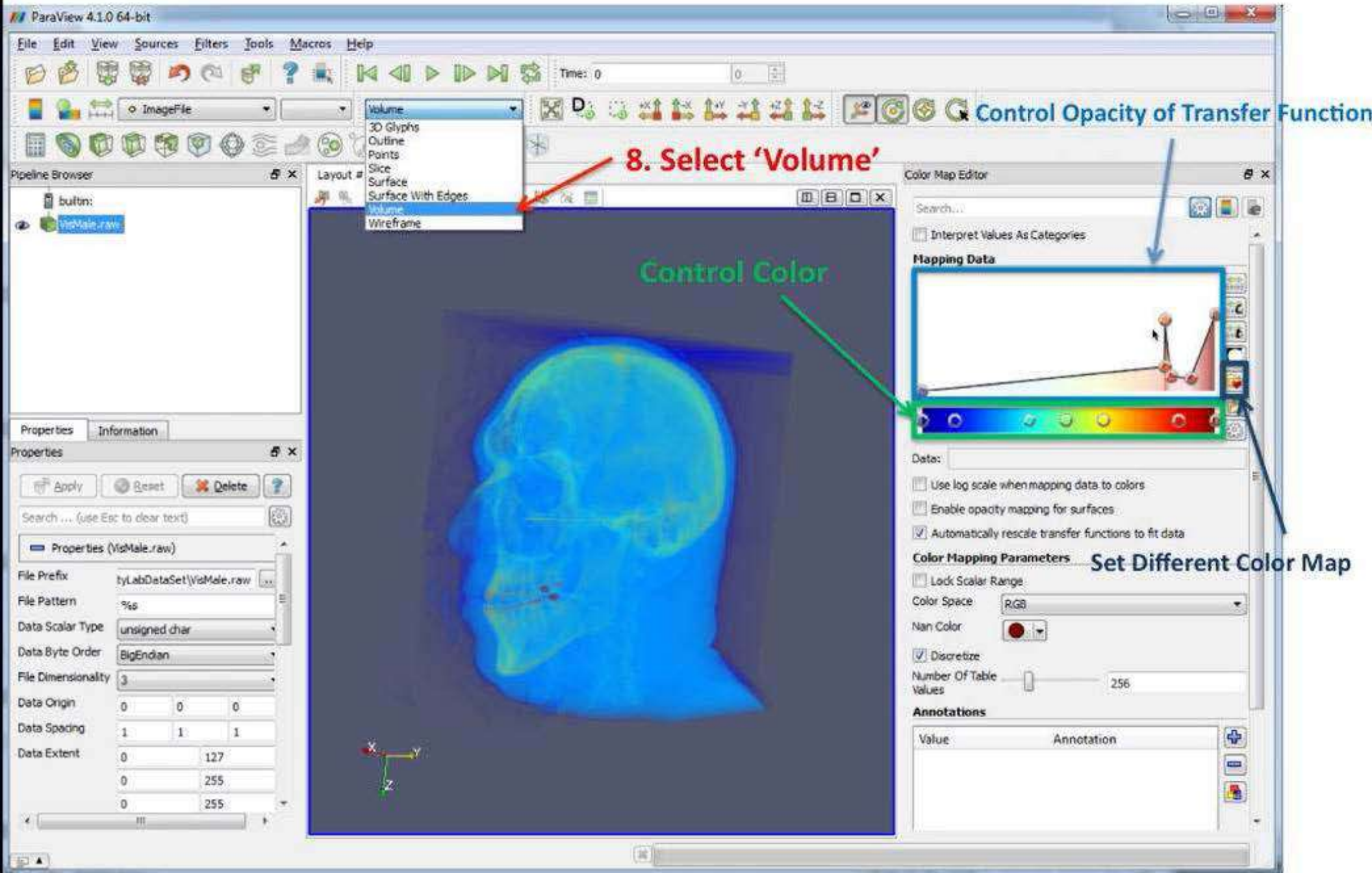
Trial and Error

- Manually control the color and opacity of each scalar value using graphical user interface (GUI)
  - Very tedious work and inefficient
  - Small transfer function change can produce very different images

It could be difficult to get a good image without enough prior knowledge

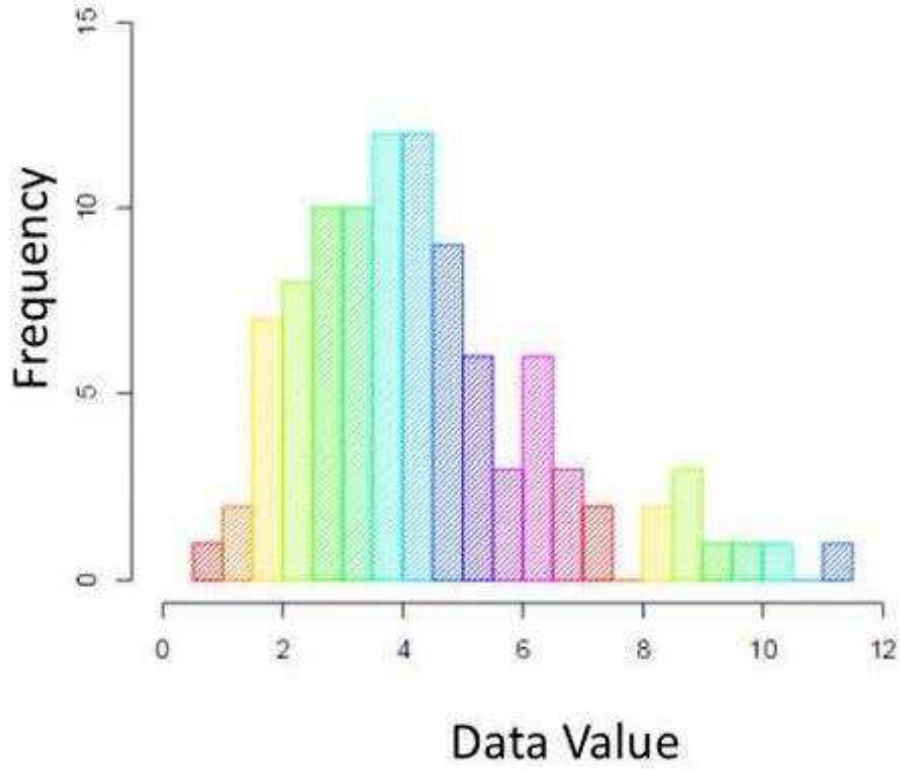


Paraview


ParaView 4.1.0 64-bit
File Edit View Sources Filters Tools Macros Help
Time: 0
ImageFile
Volume
3D Glyphs
Outline
Points
Slice
Surface
Surface With Edges
Volume
Wireframe
8. Select 'Volume'
Control Opacity of Transfer Function
Control Color
Set Different Color Map
Pipeline Browser
builtin:
VisMale.raw
Properties
Information
Apply
Reset
Delete
Search ... (use Esc to clear text)
Properties (VisMale.raw)
File Prefix
File Pattern
Data Scalar Type
Data Byte Order
File Dimensionality
Data Origin
Data Spacing
Data Extent
Color Map Editor
Search...
Interpret Values As Categories
Mapping Data
Use log scale when mapping data to colors
Enable opacity mapping for surfaces
Automatically rescale transfer functions to fit data
Color Mapping Parameters
Lock Scalar Range
Color Space
RGB
Nan Color
Discretize
Number Of Table Values
256
Annotations
Value
Annotation

Histogram Assisted Transfer Function Design

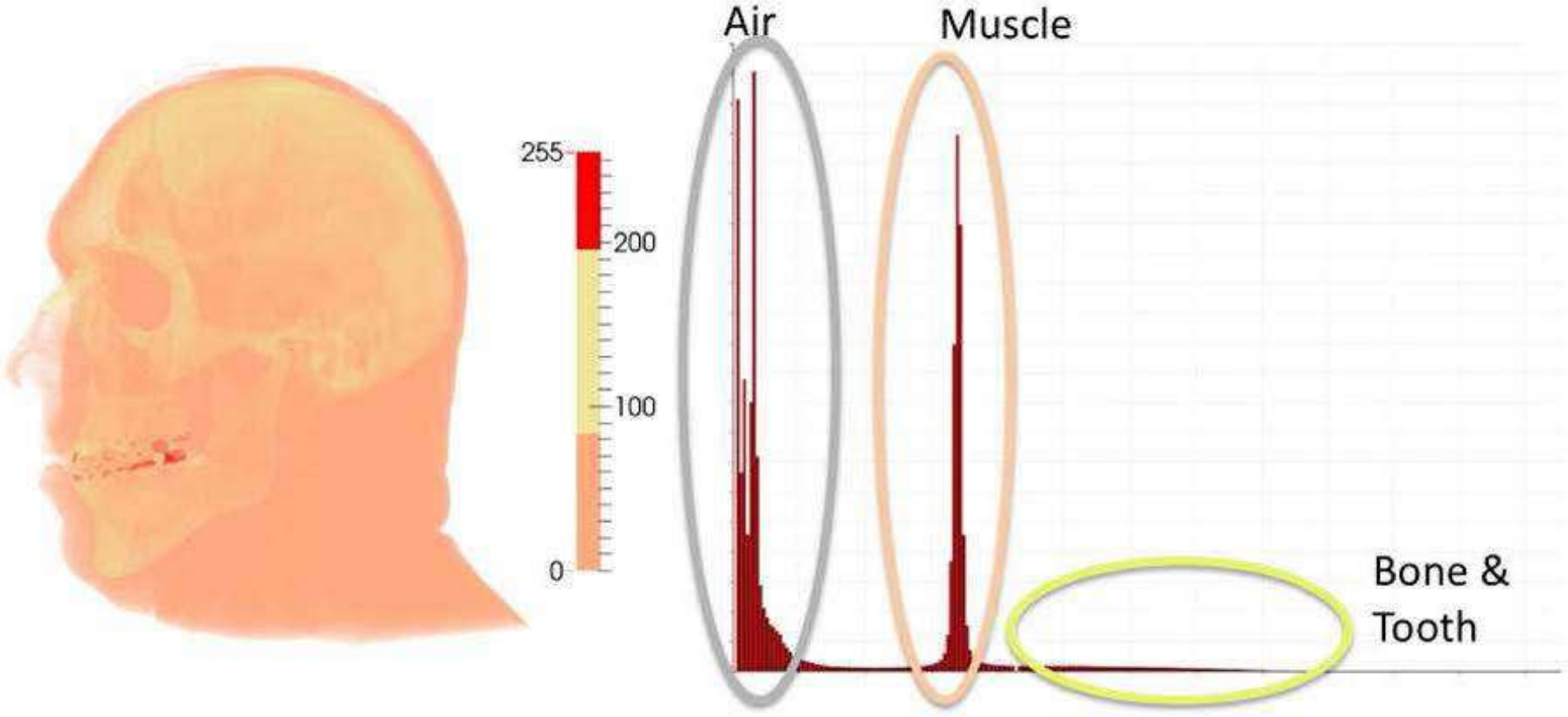
- Histogram
  - Divide the data range into finite intervals(bins)
  - Frequency of a bin is the number of samples whose values are in the interval



*http://en.wikipedia.org/wiki/Histogram

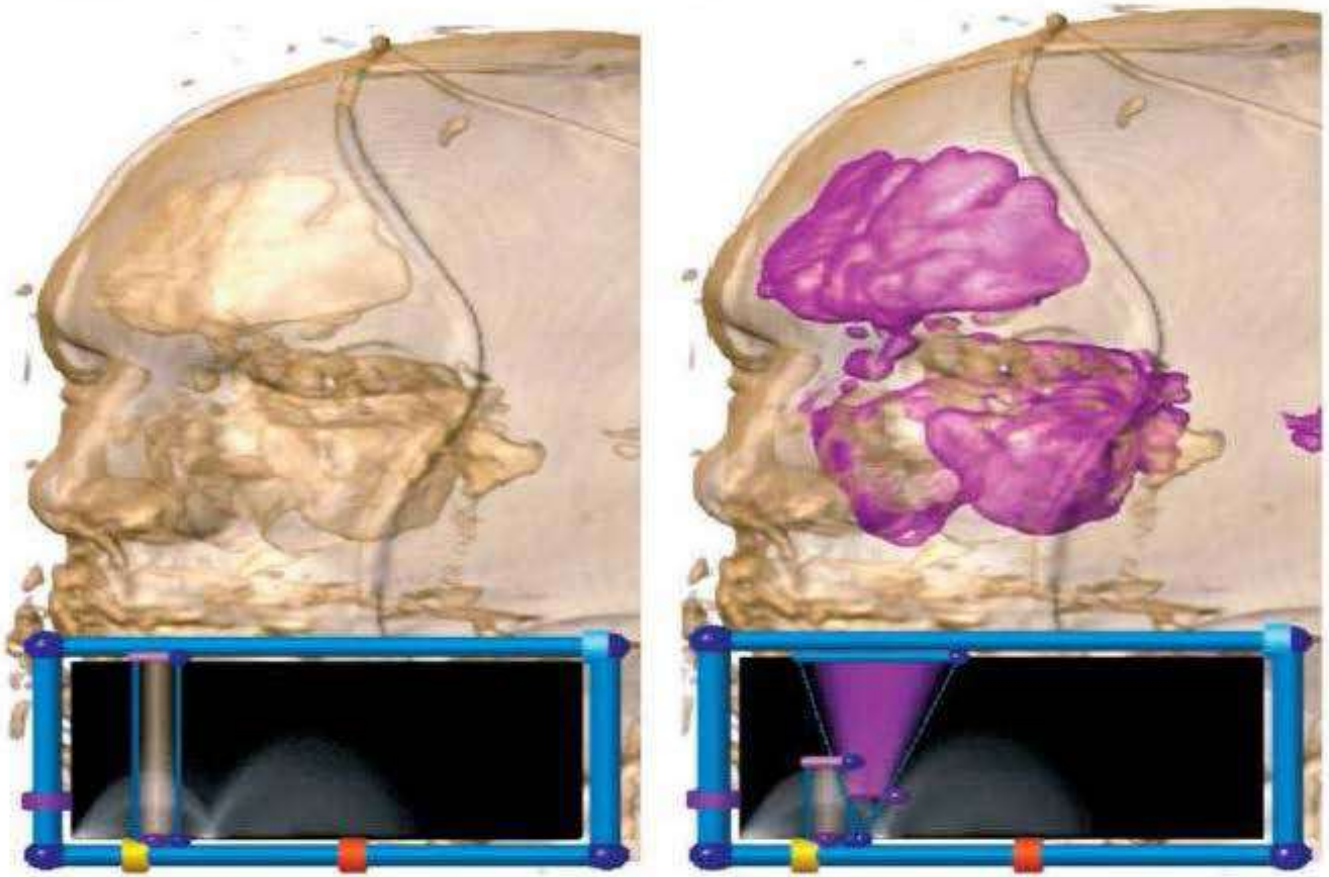
Histogram Assisted Transfer Function Design

- Different features in the data set sometimes have values in different scalar ranges
- If this is the case, value clusters can be seen from the histogram
- Different value clusters can be assigned with different colors and opacities



Multi-Dimensional Transfer Functions

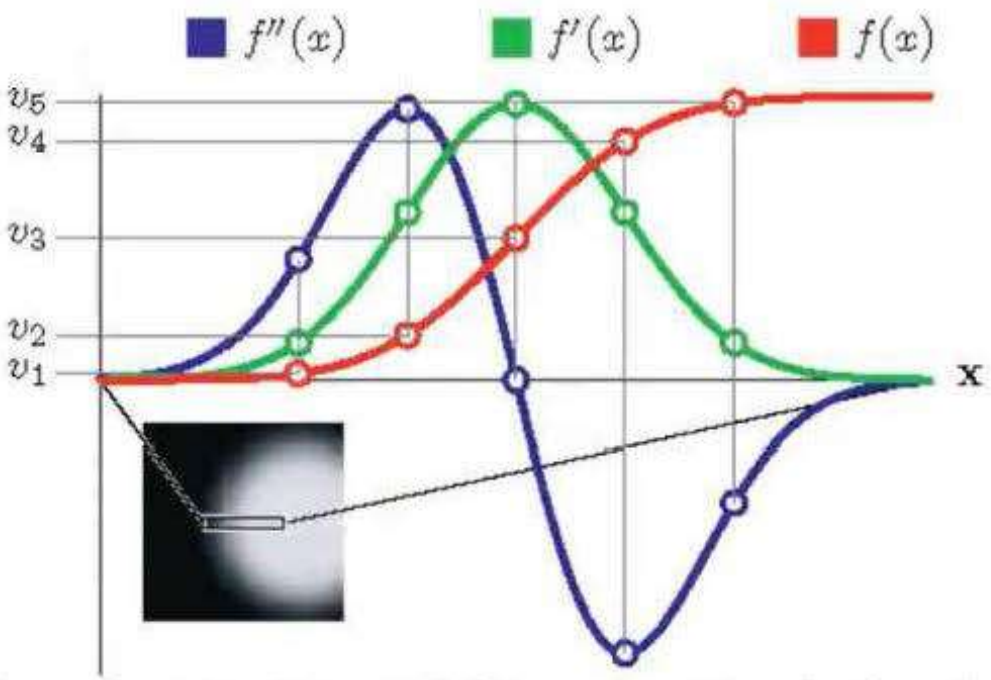
- Certain features cannot be captured by 1D histograms
  - boundary between two materials
    - Ex: emphasize the boundary between sinuses and tissue



*Interactive Volume Rendering Using Multi-Dimensional Transfer. Functions and Direct Manipulation Widgets. Joe Kniss. Gordon Kindlmann. Charles Hansen. 2001

Multi-Dimensional Transfer Functions

- How to detect/capture the boundaries
  - Values: step function
  - Gradients: local maximum
  - 2nd derivatives: zero crossing

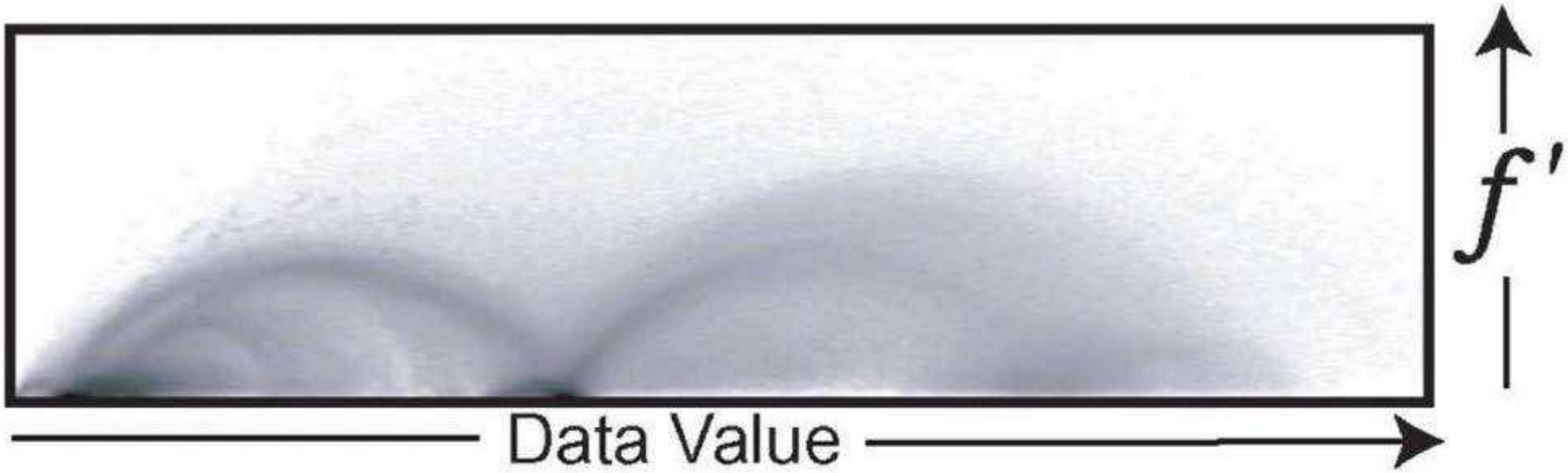


point

*Interactive Volume Rendering Using Multi-Dimensional Transfer. Functions and Direct Manipulation Widgets. Joe Kniss. Gordon Kindlmann. Charles Hansen. 2001

Multi-Dimension Transfer Function Design

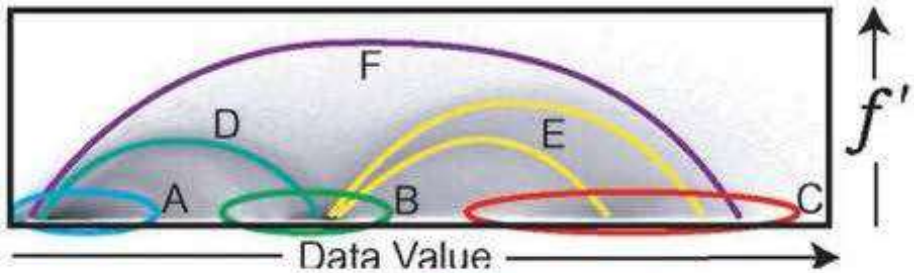
- 2D Histogram
  - X-axis – data value
  - Y-axis – gradient
  - Color intensity – frequency of the histogram (darker means more here)

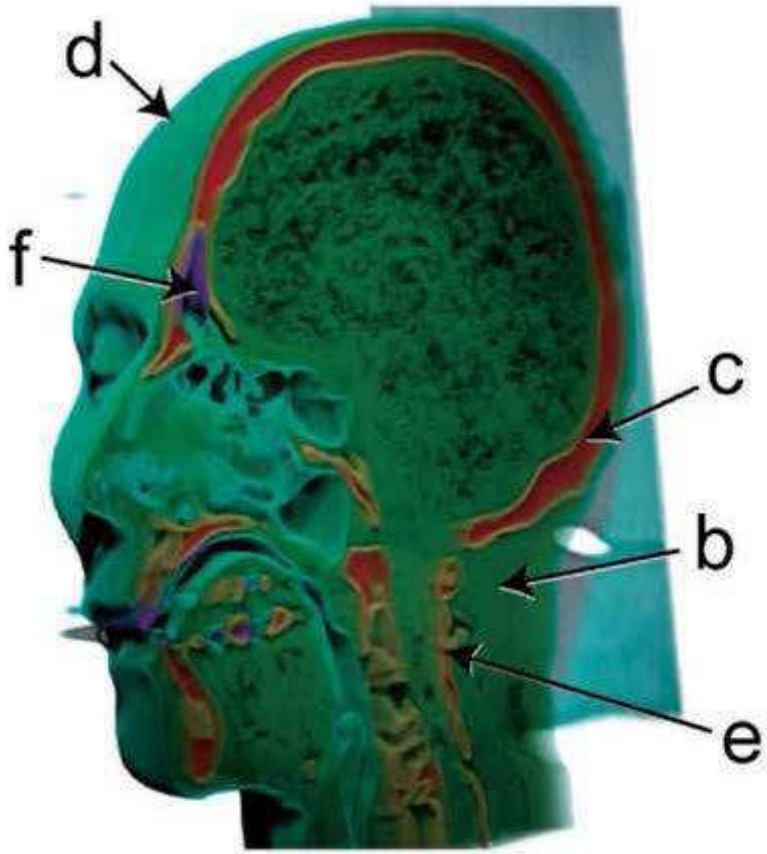


*Interactive Volume Rendering Using Multi-Dimensional Transfer. Functions and Direct Manipulation Widgets. Joe Kniss, Gordon Kindlmann, Charles Hansen, 2001

Multi-Dimensional Transfer Functions

- 1D histogram can capture homogeneous region only
  - A : air
  - B : tissue
  - C : bone
- 2D histogram can capture
  - D : air and tissue boundary
  - E : tissue and bone boundary
  - F : air and bone boundary



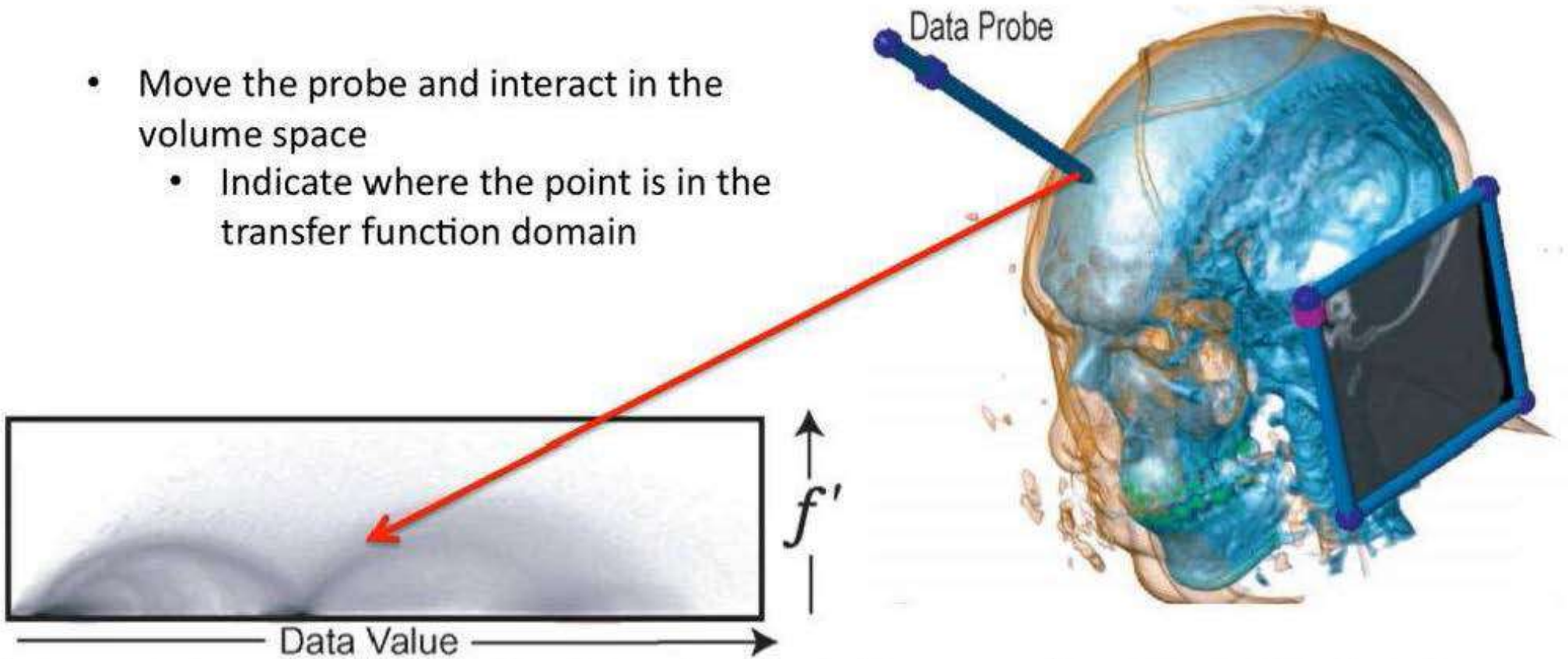


*Interactive Volume Rendering Using Multi-Dimensional Transfer. Functions and Direct Manipulation Widgets. Joe Kniss. Gordon Kindlmann. Charles Hansen. 2001

Interface for Transfer Function Design

- Use graphical interface to map 3D points back to the transfer function domain

- Move the probe and interact in the volume space
  - Indicate where the point is in the transfer function domain



*Interactive Volume Rendering Using Multi-Dimensional Transfer. Functions and Direct Manipulation Widgets. Joe Kniss, Gordon Kindlmann, Charles Hansen, 2001

Isocontours

Marching Cubes

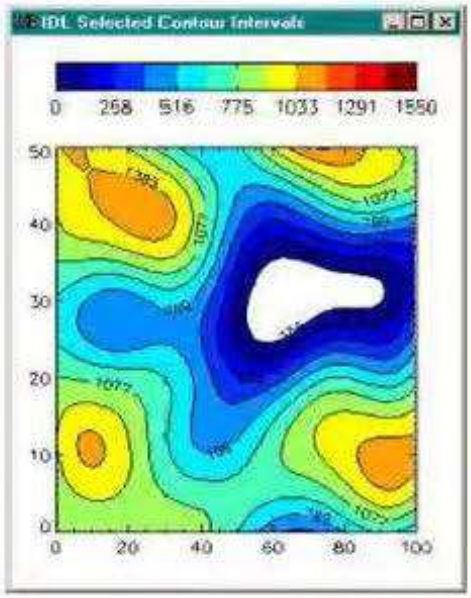
Han-Wei Shen

The Ohio State University

What is an iso-contour?

- Points in a scalar field (pressure, temperature, etc.) that have a constant value
  - 2D: isoline
  - 3D: isosurface
- It is also called a level set

$$L_f(c) = \{x | f(x) = c\}$$

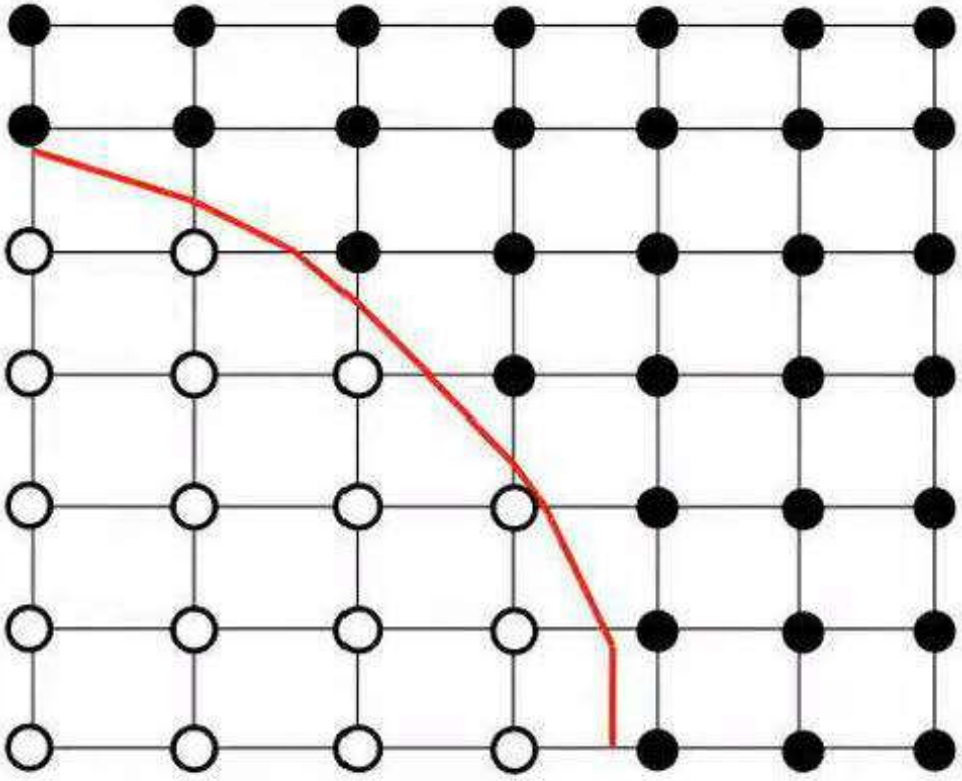


isolines

Isosurfaces

2D Isocontour

- Given a 2D scalar field, computing a 2D isocontour can be achieved cell by cell



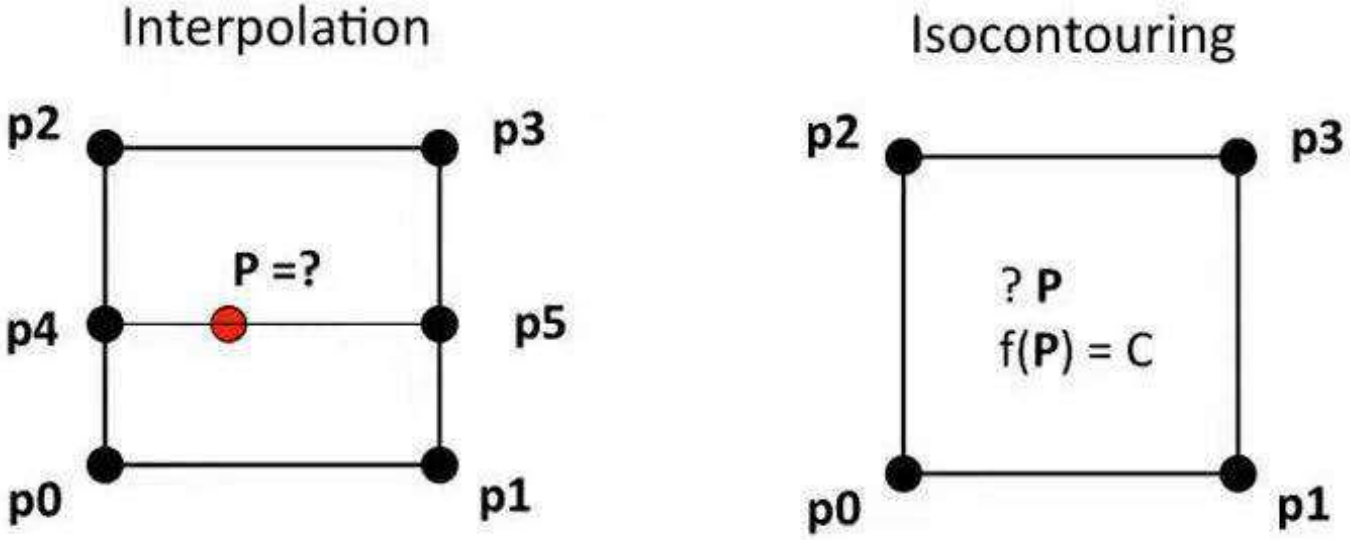
Contour Value = C

● : Value > C

○ : Value < C

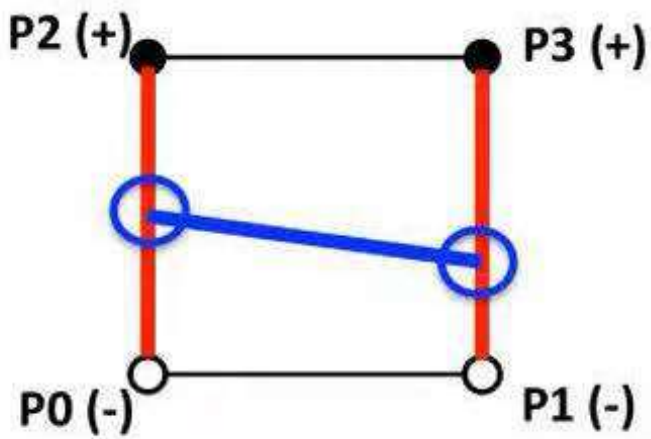
Isocontouring

- Isocontouring in a cell is an inverse problem of value interpolation



Isocontouring by Linear Interpolation

- We can compute isocontour within a cell based on linear interpolation



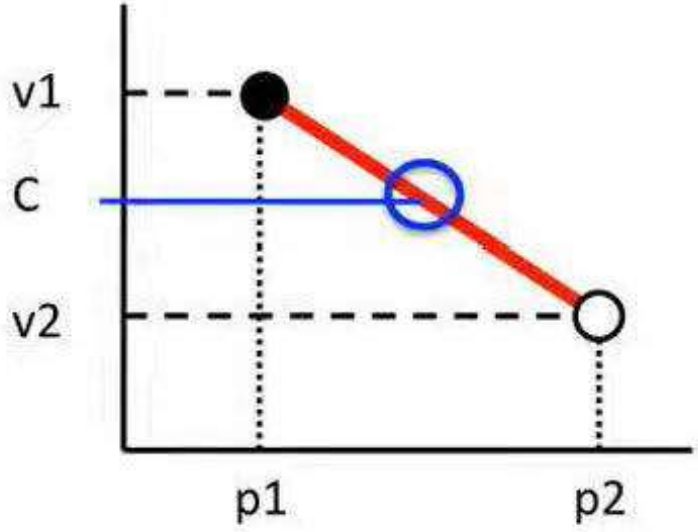
(1) Identify edges that are 'zero crossing'
- Values at the two end points are greater (+) and smaller (-) than the contour value

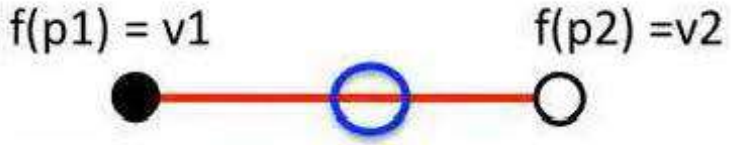
(2) Calculate the positions of **P** in those edges

(3) Connect the points with lines

Step 1: Identify Edges

- Edges that have values greater (+) and less (-) than the contour values must contain a point P that has f(p) = c
  - This is based on the assumption that values vary linearly and continuously across the edge



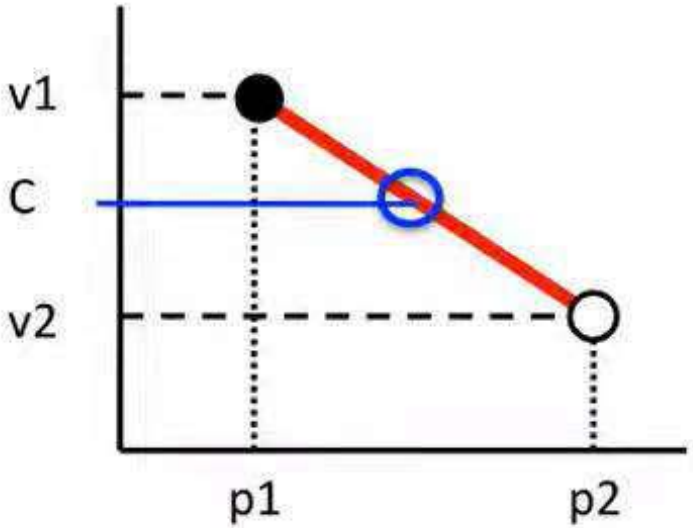


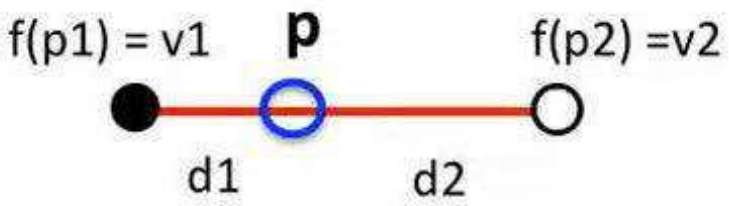
Step 2: Compute Intersection

- The intersection point **f(p) = c** on the edge can be computed by linear interpolation

$$d1/d2 = (v1-C) / (C-v2) \Rightarrow (\mathbf{p} - \mathbf{p}1)/(\mathbf{p}2-\mathbf{p}1) = (v1-C) / (v1-v2)$$

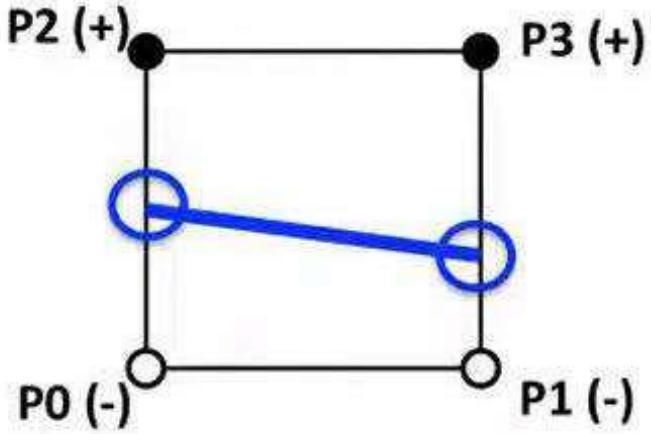
$$\mathbf{p} = (v1-C)/(v1-v2) * (\mathbf{p}2-\mathbf{p}1) + \mathbf{p}1$$





Step 3: Connect the Dots

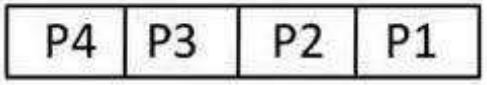
- Based on the principle of linear interpolation, all points along the line $\mathbf{p}_4\mathbf{p}_5$ have values equal to C



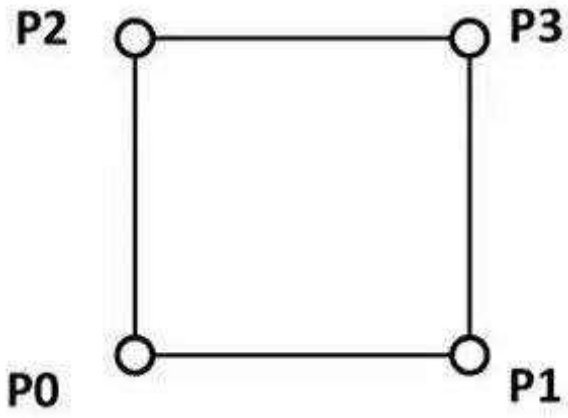
Repeat Step1 – Step 3 for all cells

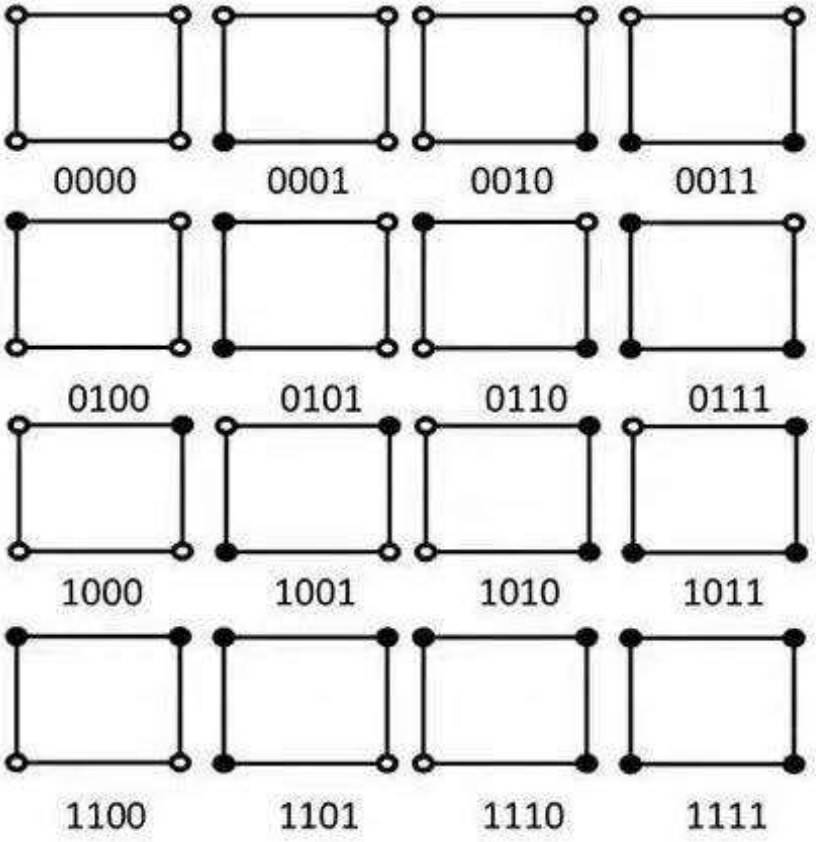
Isocontour Cases

- How many way can an isocontour intersect a linear rectangular cell?



1: > C
0: < C

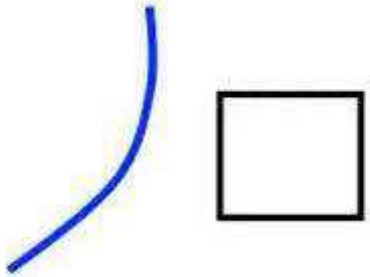




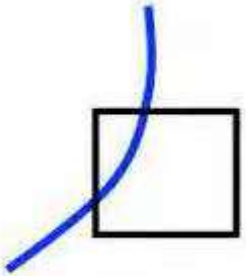
- The value at each vertex can be either greater or less than the contour value
- So there are 2 x 2 x 2 x 2 = 16 cases

Unique Topological Cases

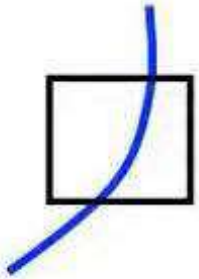
- There are only four unique topological cases



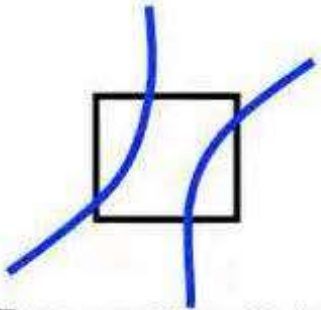
(1) No intersection



(2) Intersect with two adjacent edges



(3) Intersect with two opposite cases

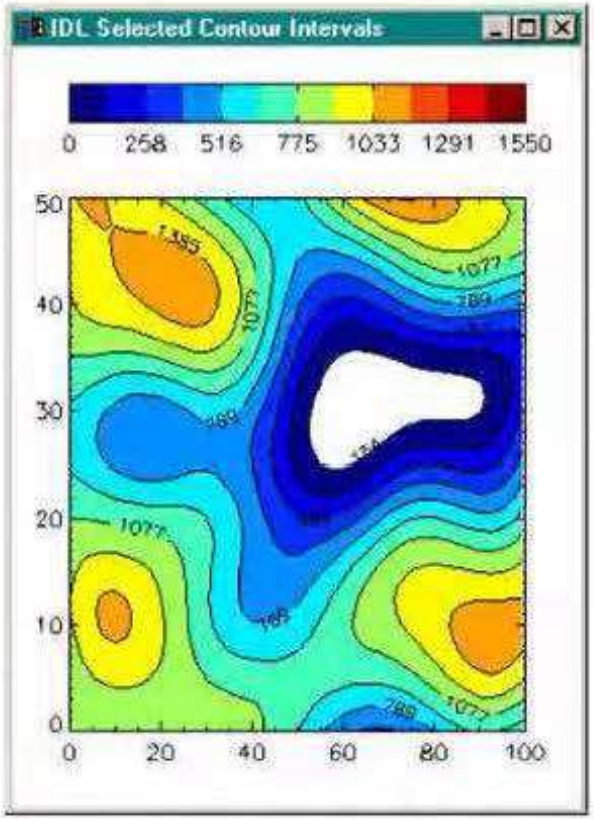


(4) Two contours pass through the cell

What are the possible bit strings for each of the cases here?

Put It All Together

- 2D Isocontouring algorithm:



1. Process one cell at a time
2. Compare the values at 4 vertices with the contour value C and identify intersected edges
3. Linear interpolate along the intersected edges
4. Connects the interpolated points together

3D Isocontour (Isosurface)

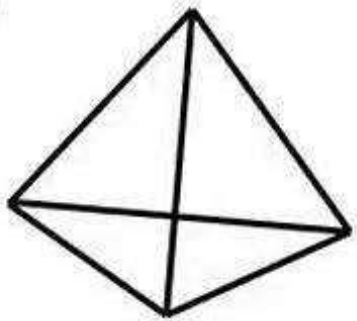
骨密度~软组织

Isosurface Extraction

- Extend the 2D algorithm presented above to three dimensions
  - 3D cells

  - Linear interpolation along edges in active cells
    - Active cells: cells that intersect with the isosurface
  - Compute surface patches within each cell
    - Depend on which edges are intersected by the isosurface

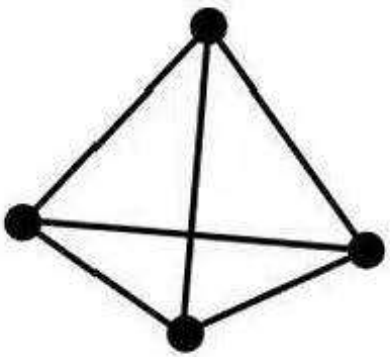
Tetrahedral Cells



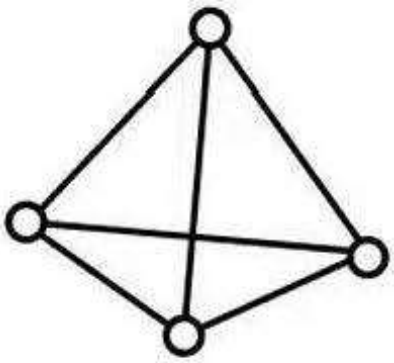
- Active cells: min < C < Max
- Mark cell vertices that are greater than the contour value as "+", and smaller than the contour value as "-"
- Each cell has four vertices
  - Each cell can have a value greater (+) or smaller (-) than the contour value
  - A total of 2 x 2 x 2 x 2 = 16 combinations, but there are only three unique topological cases

Case 1: No Intersection (all outside or inside)

- Values at all cell vertices are greater or smaller than the contour values
  - If we call cell values greater than the contour value as 'outside' and smaller as 'inside', then all cell vertices are either completely inside or outside of the isosurface



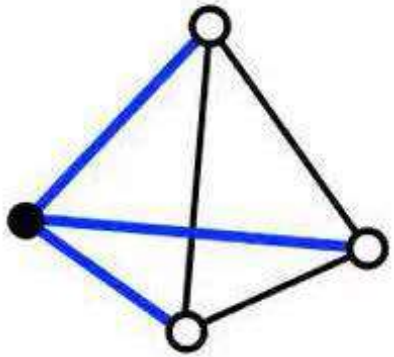
All Vertices Outside



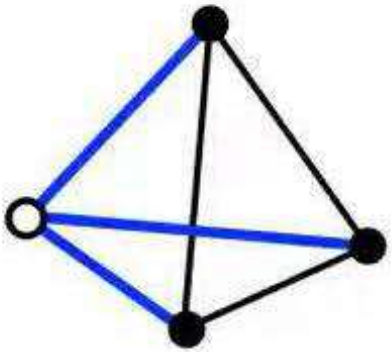
All Vertices Inside

Case 2: One Outside (or Inside)

- Only one of the vertices are outside or inside of the isosurface



One Outside

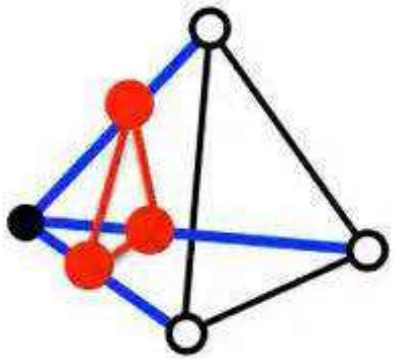


One Inside

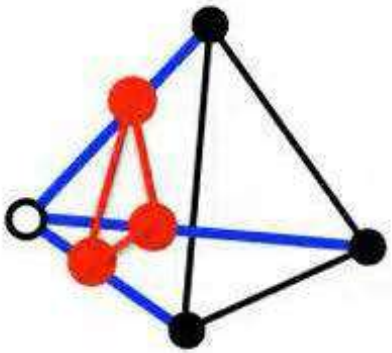
- Isosurface only intersects with edges that have '+' and '–' vertices at two ends

Case 2: Triangulation

- Compute the intersection points on the active edges



One Outside



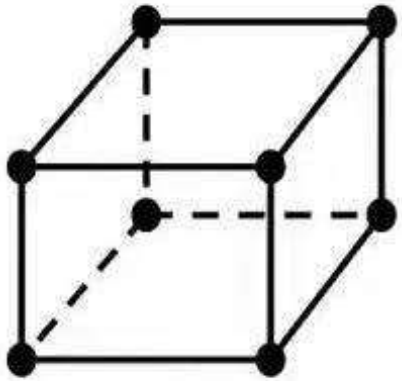
One Inside

- Connect the intersection points into a single triangle

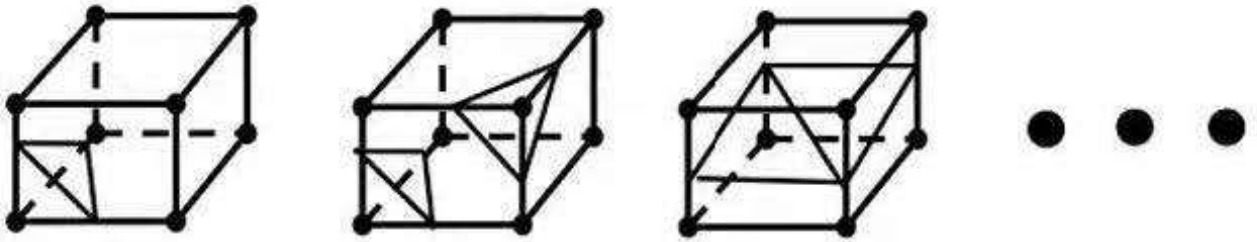
Put It All Together

1. Iterate through all tetrahedral cells
   a) Compare the values at the four corners of each cell
   b) Determine the edges that intersect with the isosurface if any
   c) Compute the surface-edge intersection points using linear interpolation
   d) Triangulate the intersection points based on the cases discussed above

Rectangular Cells



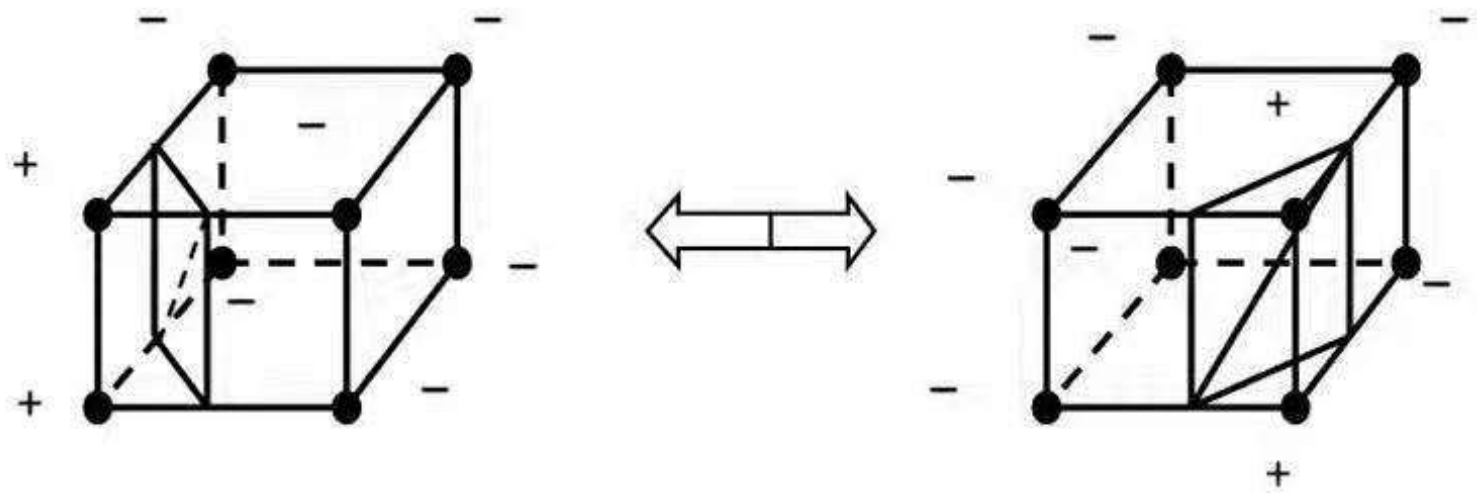
With 8 vertices in a cell, each having a value greater or smaller than the contour value, there can be $2^8 = 256$ possible cases



But the total number of unique topological cases is much smaller than 256

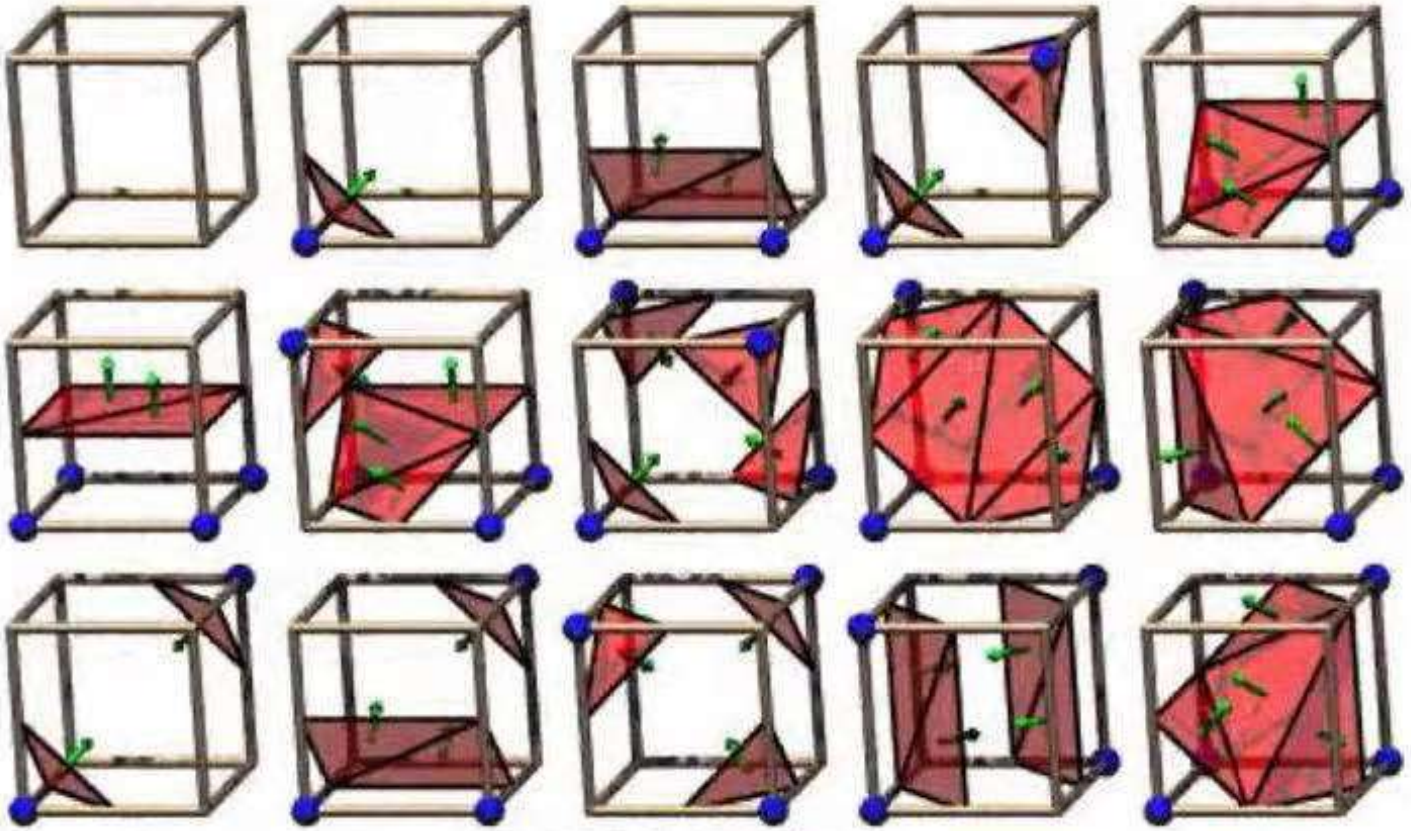
Example of Case Reduction

Rotation Symmetry



By inspection, we can reduce the number of cases from 256 to 15

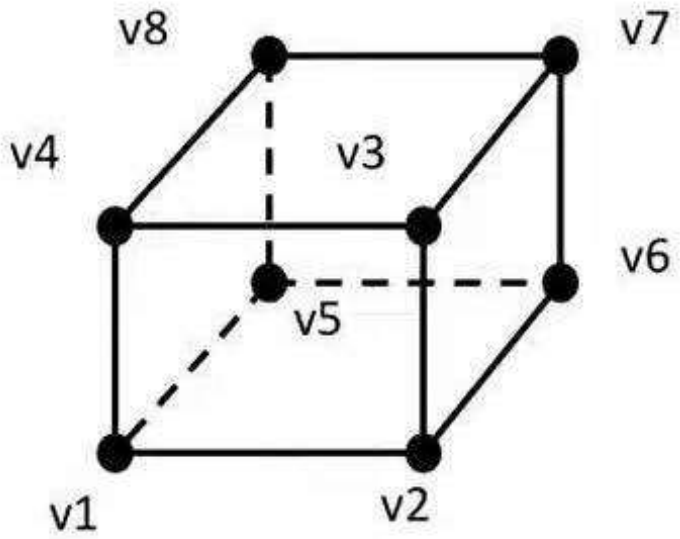
Isosurface Cases



The 15 Cube Combinations

The Marching Cubes Algorithm

- By Lorensen and Cline in 1987



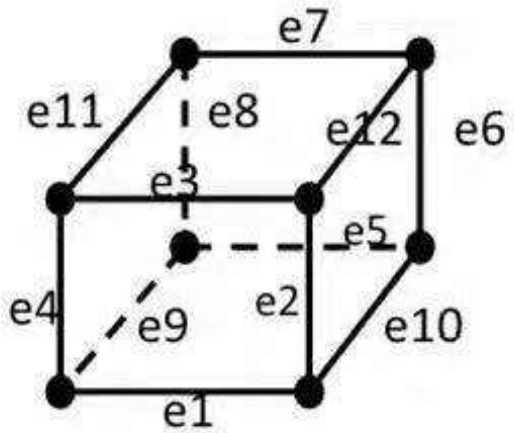
Vi is '1' or '0' (one bit)
1: > C; 0: <C (C= sovalue)

Each cell has an index mapped to a value ranged [0,255]

Index =	v8	v7	v6	v5	v4	v3	v2	v1

Marching Cubes Table Lookup

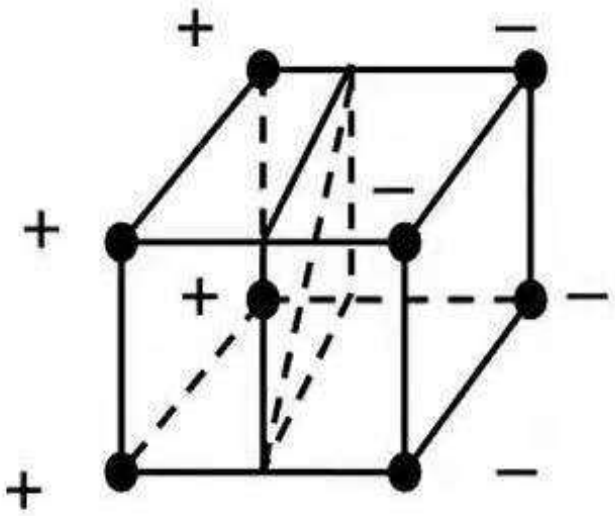
- Based on the values at the vertices, map the cell to one of the 15 cases
- Perform a table lookup to see what edges have intersections



Index	intersection edges
0	e1, e3, e5
1	…
2	
3	
	• • •
14	

Compute Intersections

- Perform linear interpolation to compute the intersection points at the edges
- Connect the points by looking up the marching cubes table



Marching the Cubes

- Sequentially scan through the cells – row by row, layer by layer
- You can re-use the intersection points for neighboring cells