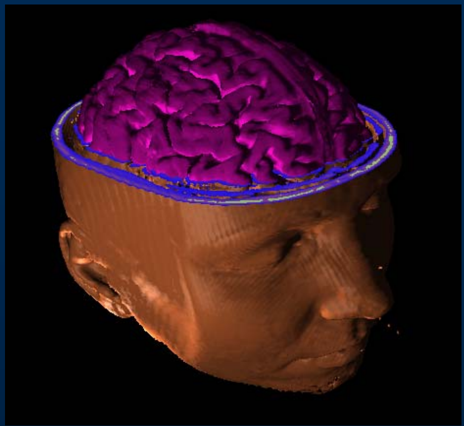
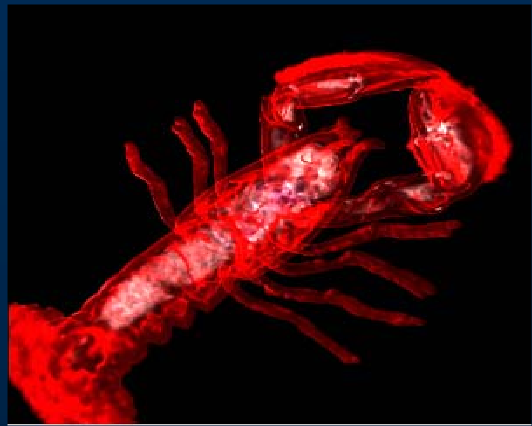
Volume Visualization

Baoquan Chen

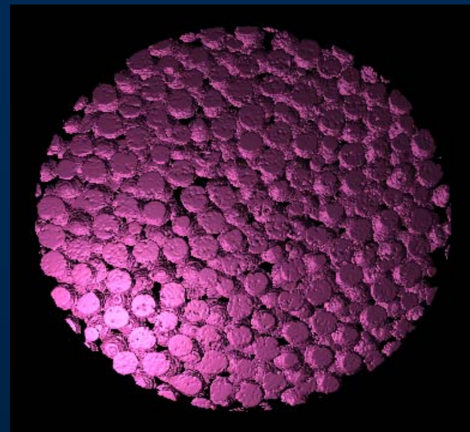
Peking University

Volume Datasets

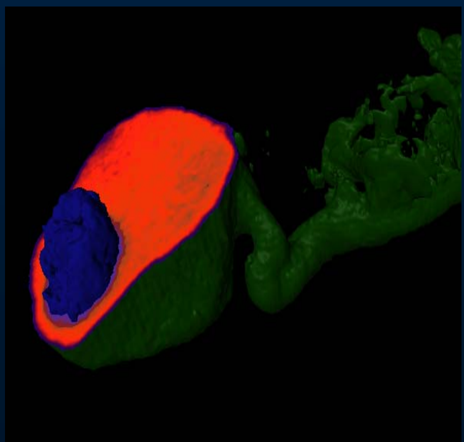




MRI / CT / PET / Ultrasonography

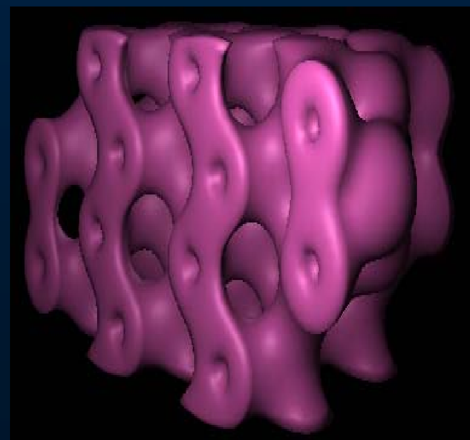


Micro-Tomography



Confocal Microscopy

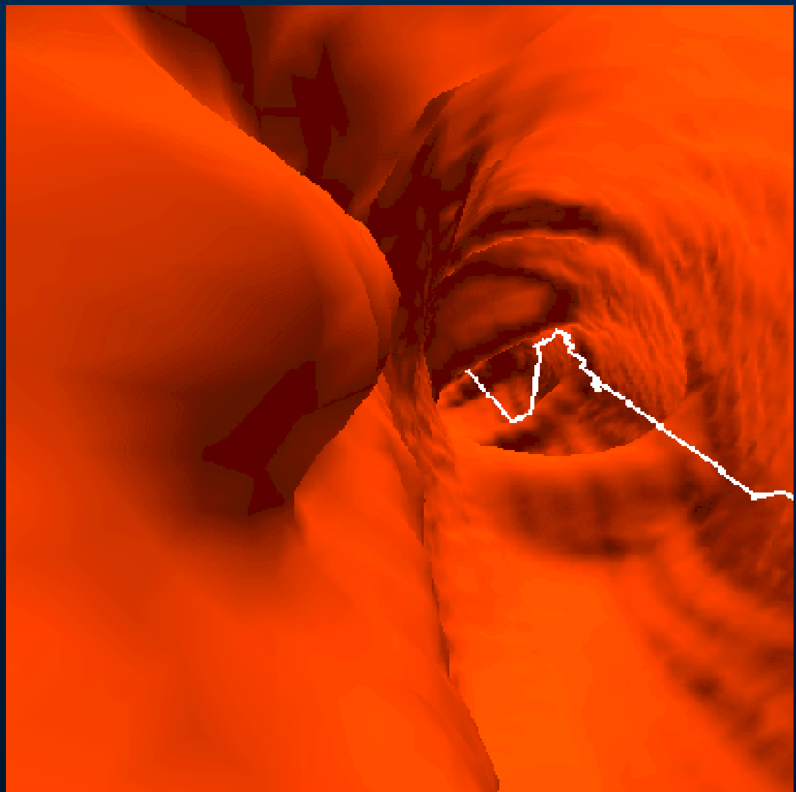
Voxelization



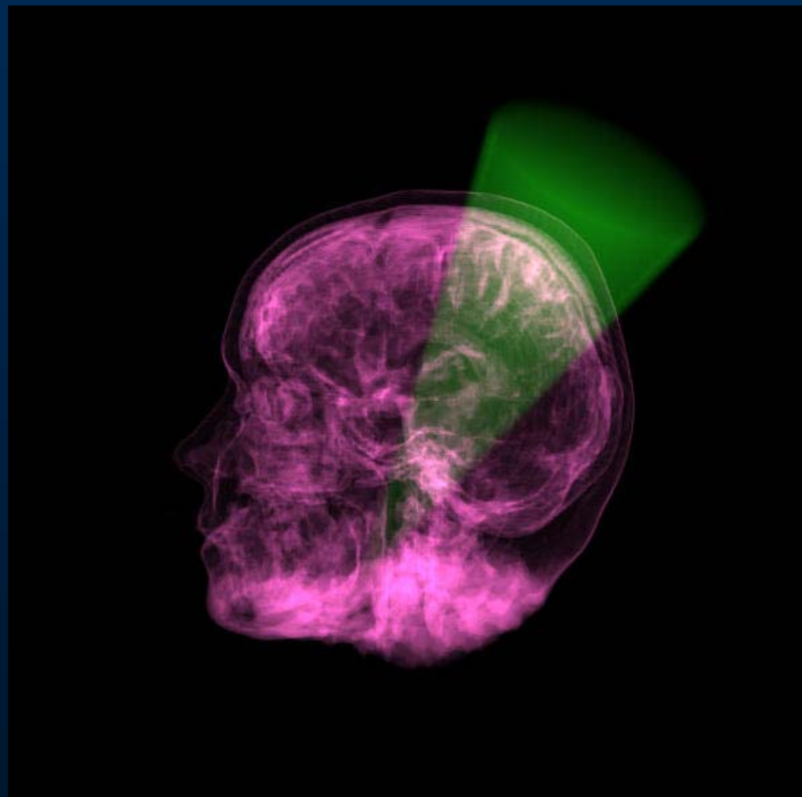
Simulation

Volumetric Application

Biomedical Visualization



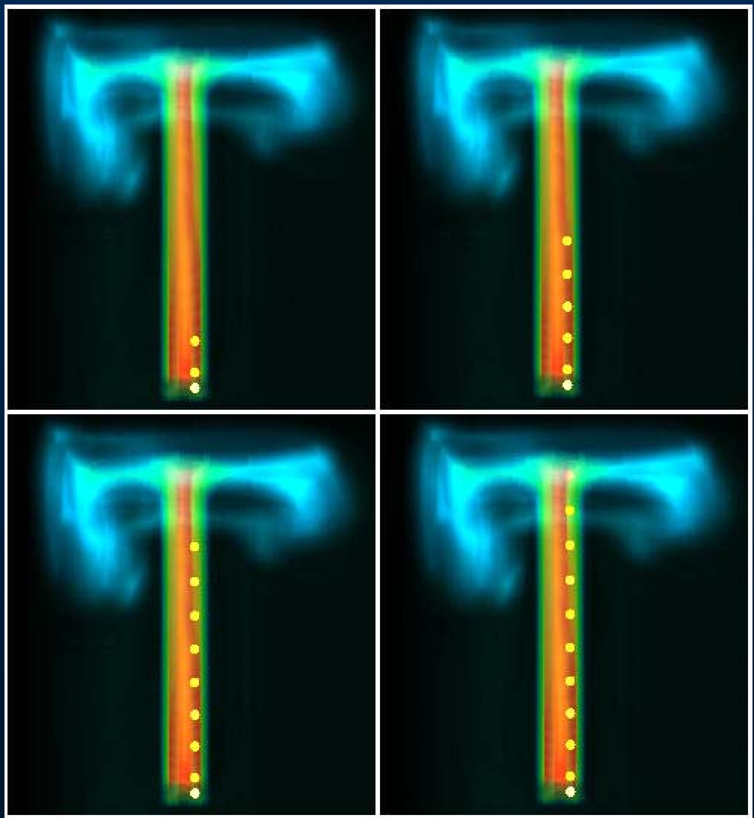
Virtual colonoscopy



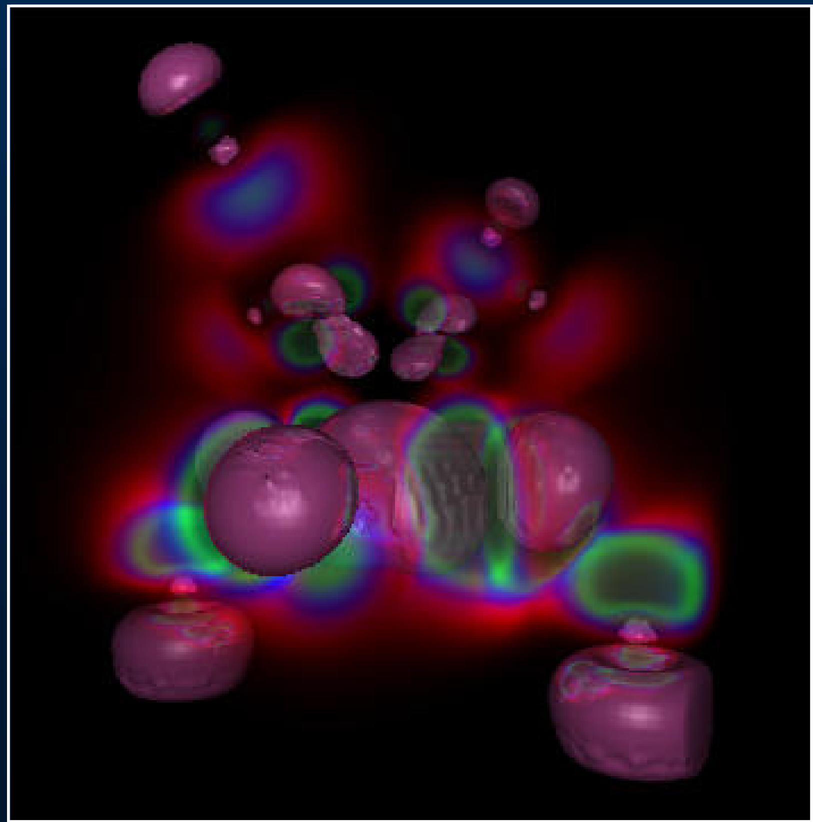
Radiation Therapy

Volumetric Application

Scientific Visualization



Computational
Fluid Dynamics (CFD)



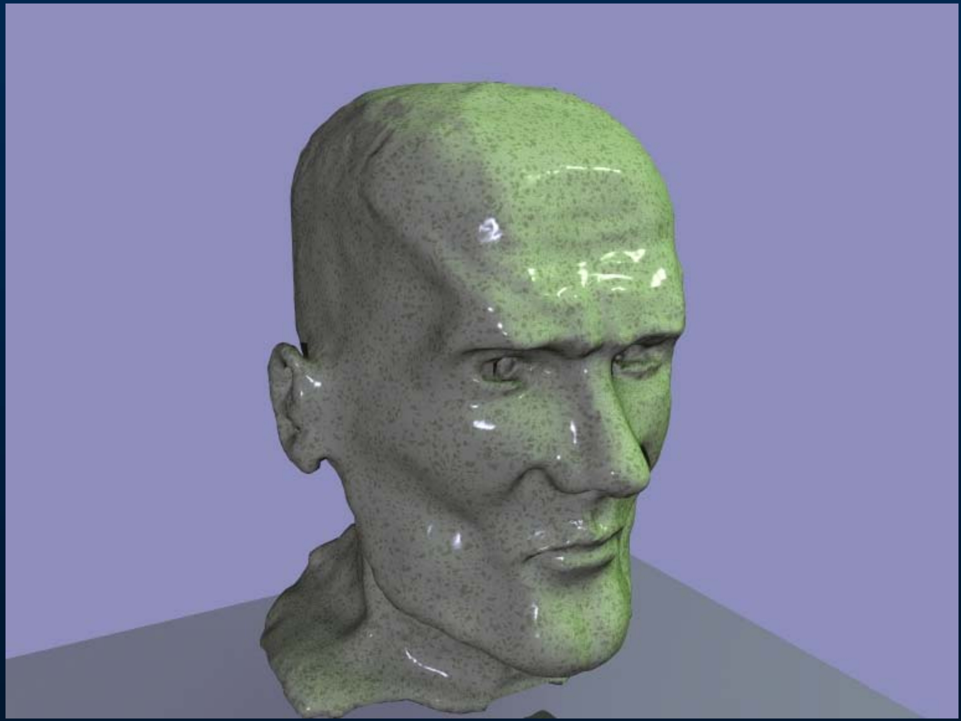
High-potential
iron proteins

Volumetric Application

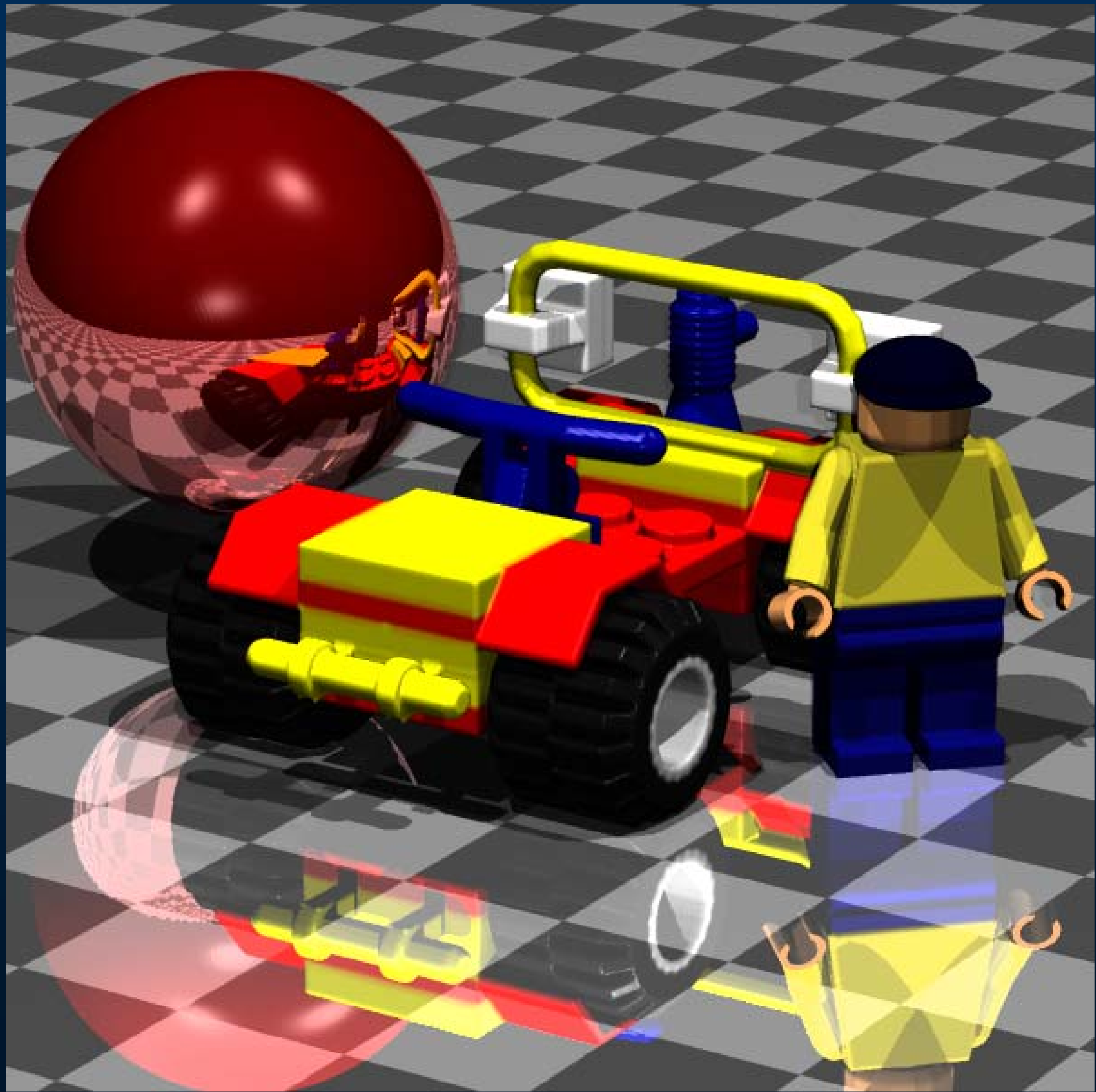
Amorphous Phenomena

Volumetric Application

Sculpting System



Volume Graphics



Volumetric objects

- have information inside it
- not consist of explicit surfaces and edges
- May be too voluminous to be represented geometrically

Volume Visualization Objective

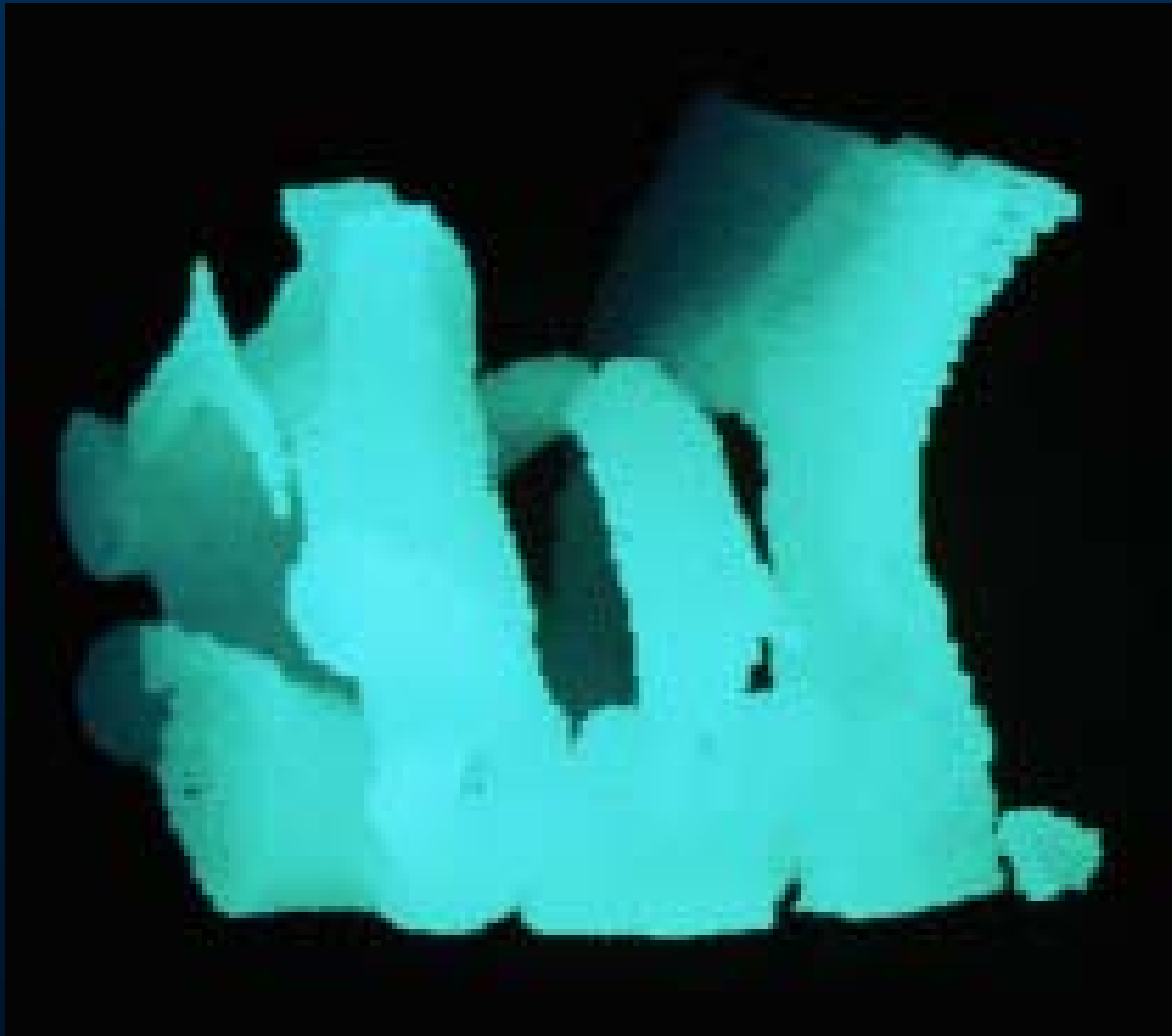
- Peer inside voumetric objects
- Probe into voluminous & complex structures

Volume Visualization

A visualization method concerned with the representation, manipulation, and rendering of volumetric data.

History of Volume Visualization

1970 First report - 3D oscillioscopic images
1970's Medical imaging
1978 3D surface presentation [Sunguroff & Greengerg]
1979 Cuberille [Herman & Liu]
1981 Depth only shading [Herman & Udupa]
1982 Octree Machine [Meagher]
Voxel processor [Goldwasser & Reynolds]
1984 Ray casting [Tuy & Tuy]
1985 Cube architecture [Kaufman & Bakalash]
Back-to-front & Front-to-back
Depth gradient shading [Gordon et al.]
Contextual shading [Chen et al.]



History of Volume Visualization

1986 3D Scan conversion [Kaufman & Shimony]
Grey-level shading [Hoehne & Bernstein]
1987 Marching cubes [Lorensen & Kline]
1988 Volume rendering [Levoy; Derbin; Upson & Keeler; Sabella]
Dividing cubes [Cline et al.]
1989 Chapel Hill Workshop
Splatting [Westover]
1990 San Diego Workshop
1992 Boston Workshop
1994 Washington Workshop
1996 San Francisco Workshop

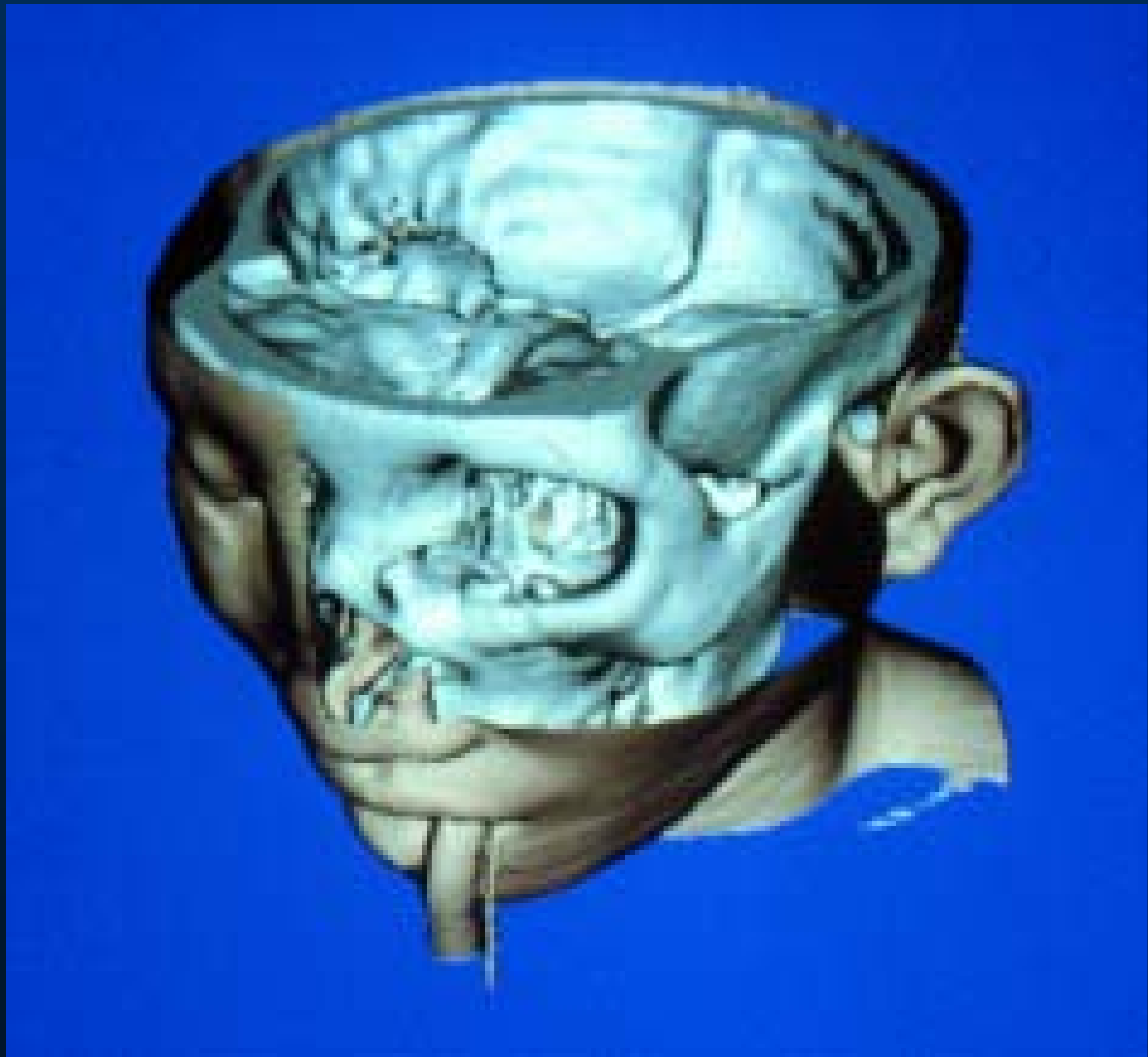
IEEE Symposium on Volume Visualization
IEEE Workshop on Volume Graphics
IEEE Visualization

Volume Visualization

- Iso-surface extracting and rendering
  - Marching Cubes (Lorensen 87)
  - Marching Tetrahedra
- Direct volume visualization
  - Ray Casting (Levoy 89)
  - Splatting (Westover 90)

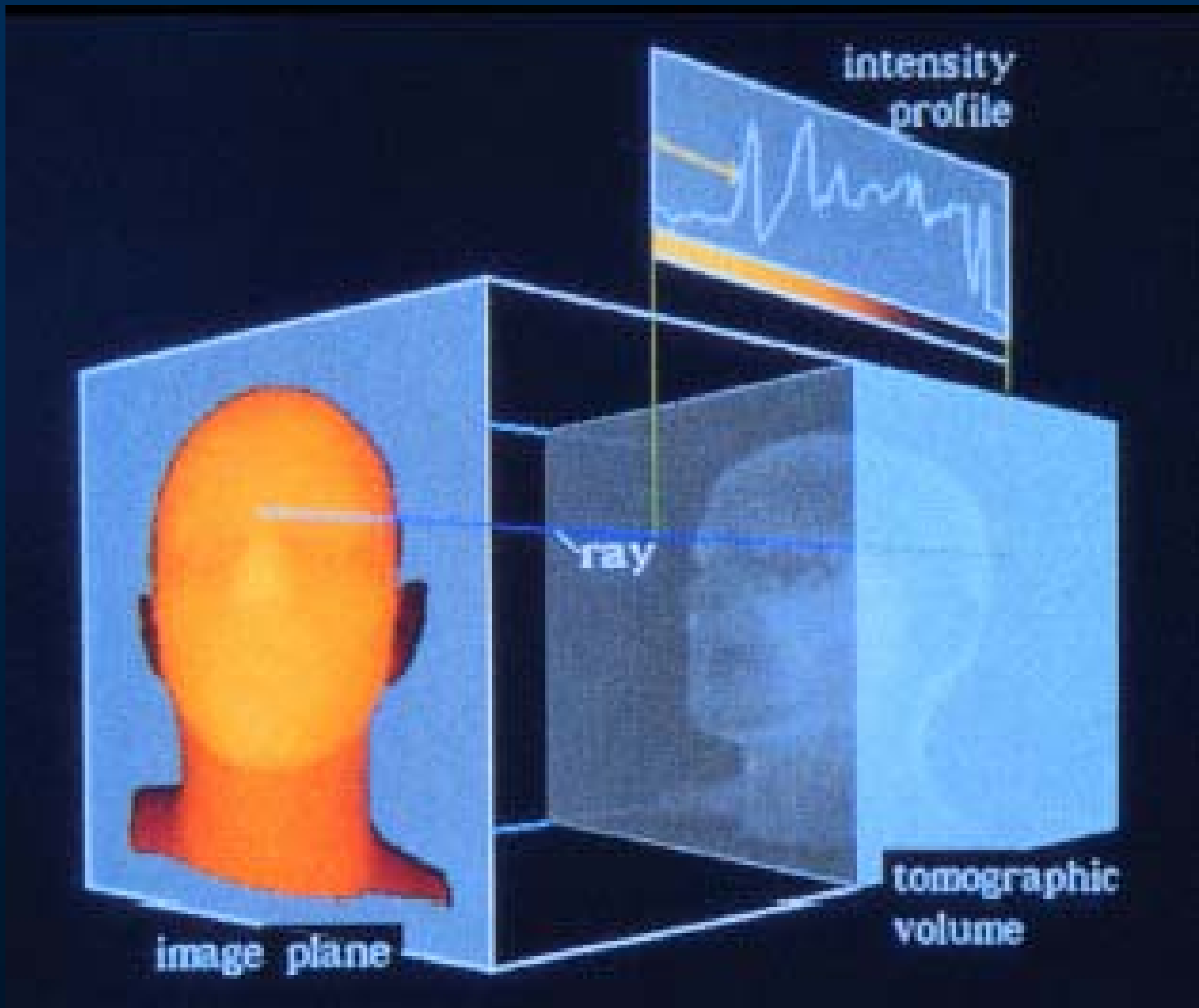
Surface Rendering

An indirect technique used for visualizing volume primitives by first converting them into an intermediate surface representation and then employing conventional computer graphics techniques to render them to the screen.



Volume Visualization

A direct technique for visualizing volume visualizing volume primitives without any intermediate conversion of the volumetric dataset to surface representation .


intensity
profile
ray
image plane
tomographic
volume

Surface Rendering

- Intermediate representation
- Tangible surfaces
- Information on surfaces
- Continuous
- Compact representation
- Fast
- Iso-surfacing

Marching Cubes

- Creates triangles
- Floating point representation
- Uses case table to create triangles
- Can use general purpose polygon-based hardware for rendering

Marching Cubes History

- Developed in 1984
- Published in Siggraph ’87
- Marching Cubes in AVS and SGI Explorer, and ***everywhere***
- 12,537 citations by google scholar
- Lorensen won achievement award at IEEE Visualization 2004

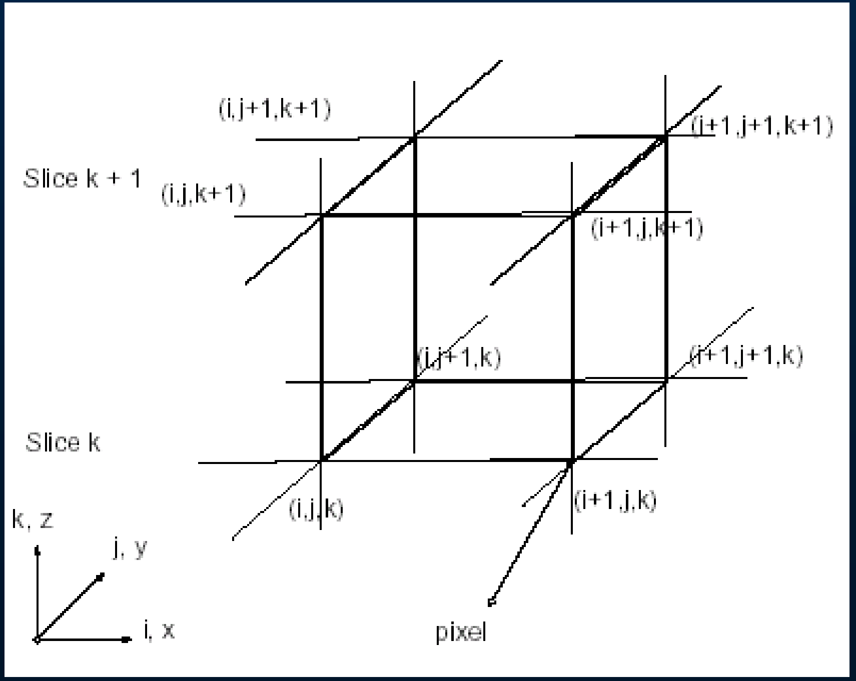
Marching Cubes Algorithm

1. Create a cube
2. Classify each vertex
3. Build an index
4. Get edge list
5. Interpolate triangle vertices
6. Calculate and interpolate normals

Marching Cubes

Step 1 - Create a cube

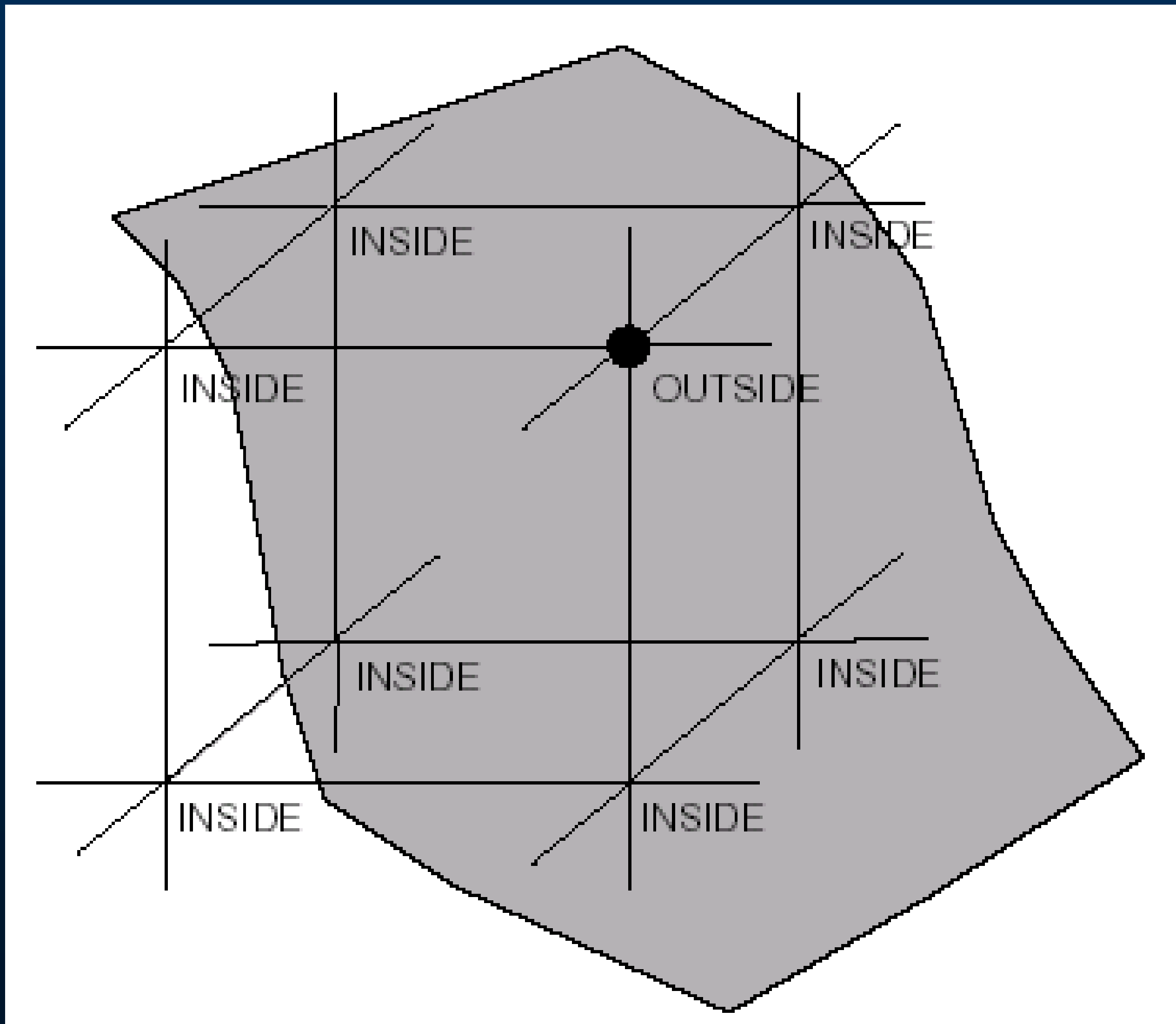
- Consider a cube defined by eight data values, four from slice k, and four from slice k + 1



Marching Cubes

Step 2 - Classify each vertex

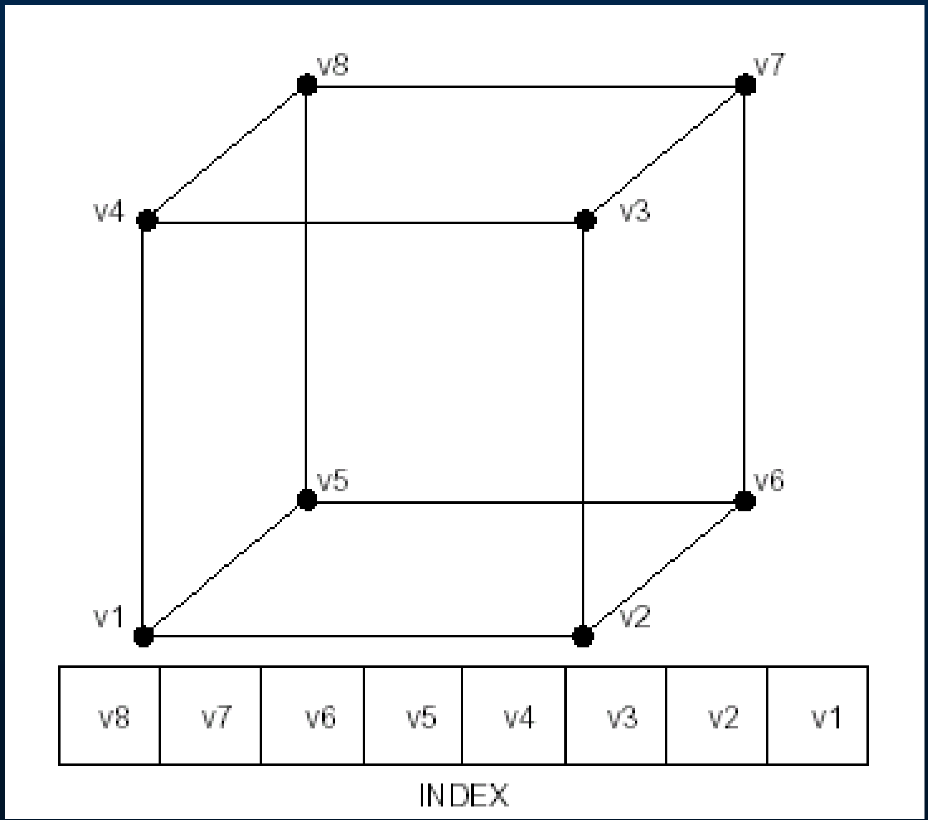
- Classify each vertex of the cube as to whether it lies outside surface or inside the surface
  - Outside if vertex value < surface value
  - Inside if vertex value >= surface


INSIDE
INSIDE
INSIDE
OUTSIDE
INSIDE
INSIDE
INSIDE
INSIDE

Marching Cubes

Step 3 - Build an index

- Create an index between 0 and 255 from the binary labeling of each vertex

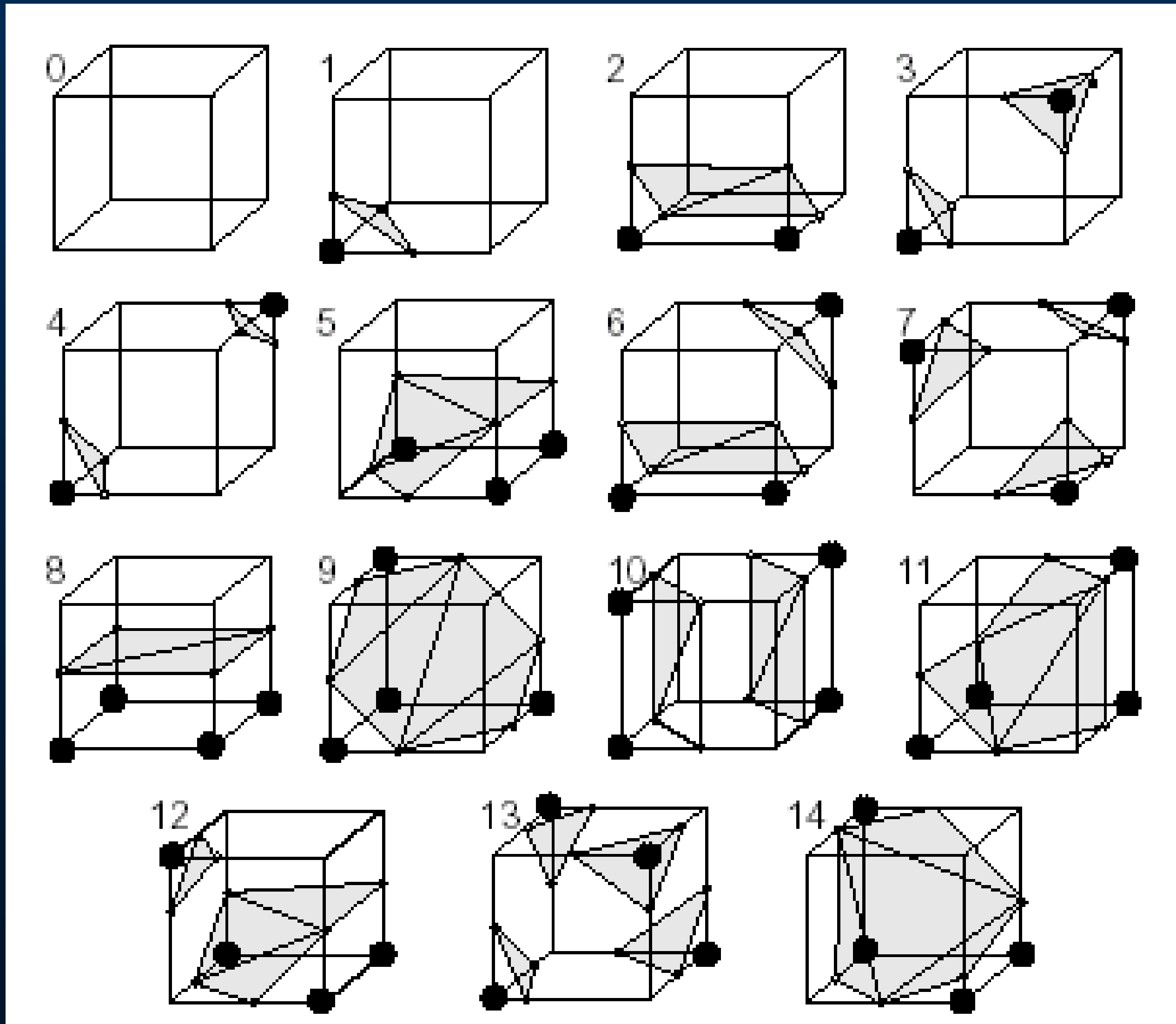


Marching Cubes

Step 4 - Get edge list

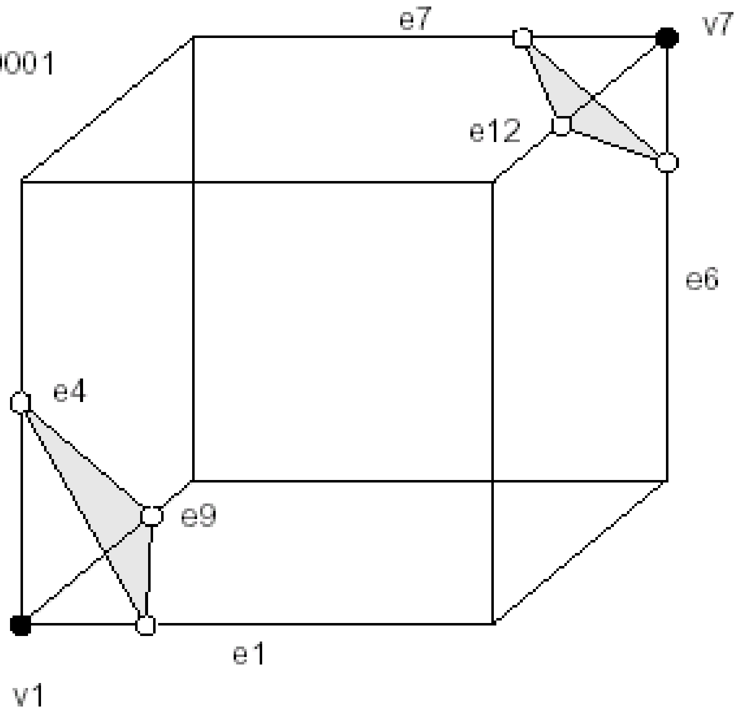
- For a given index, access a list of cubes edges that contain a triangle vertex
- Using symmetry of the cube, all 256 cases can be generated from fourteen cases

Marching Cubes



Case 4

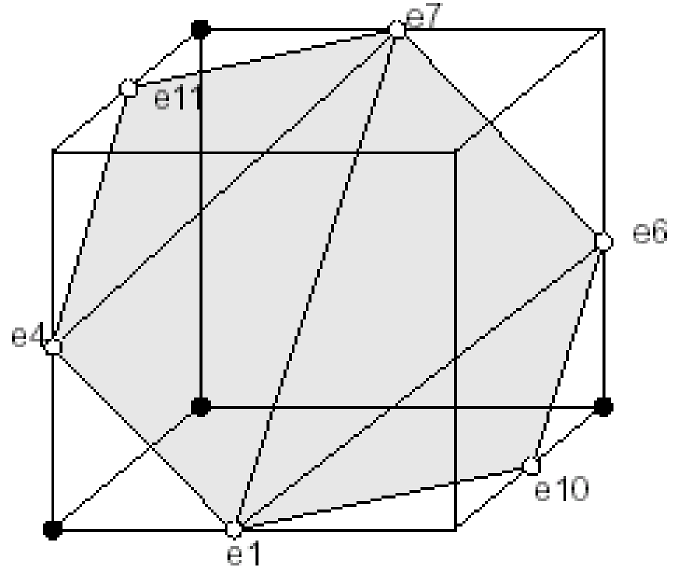
INDEX = 01000001



triangle 1 = e1, e9, e4
triangle 2 = e6, e7, e12

Case 9

INDEX = 10110001



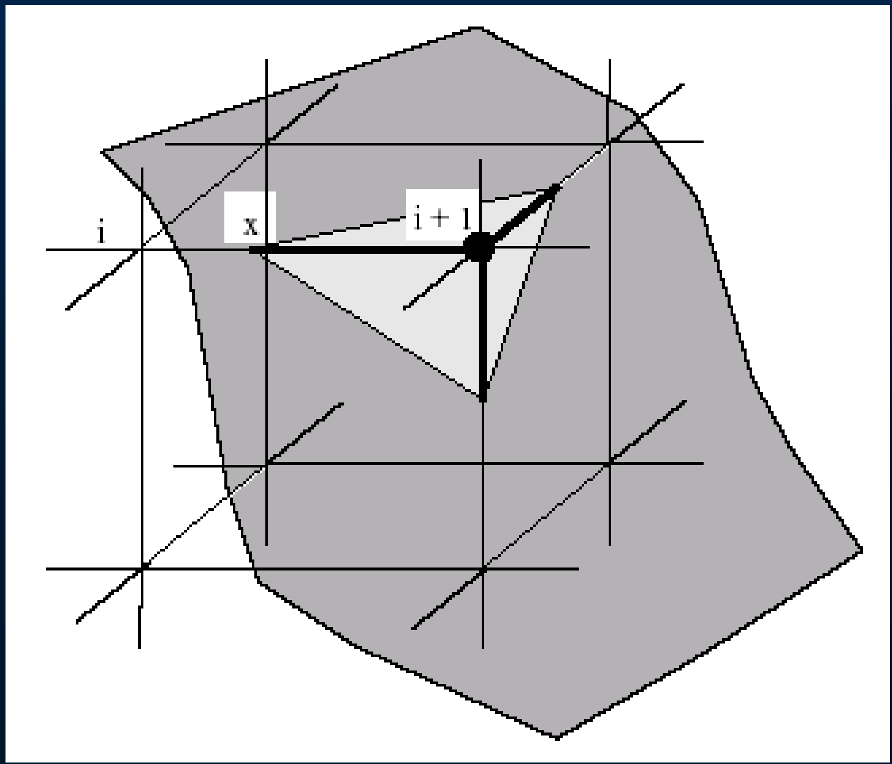
triangle 1 = e4, e7, e11
triangle 2 = e1, e7, e4
triangle 3 = e1, e6, e7
triangle 4 = e1, e10, e6

Marching Cubes

Step 5 - Interpolate triangle vertices

- For each triangle edge, find the vertex using linear interpolation of the density values

$$x = i + (\text{value} - D(i)) / (D(i + 1) - D(i))$$



Marching Cubes

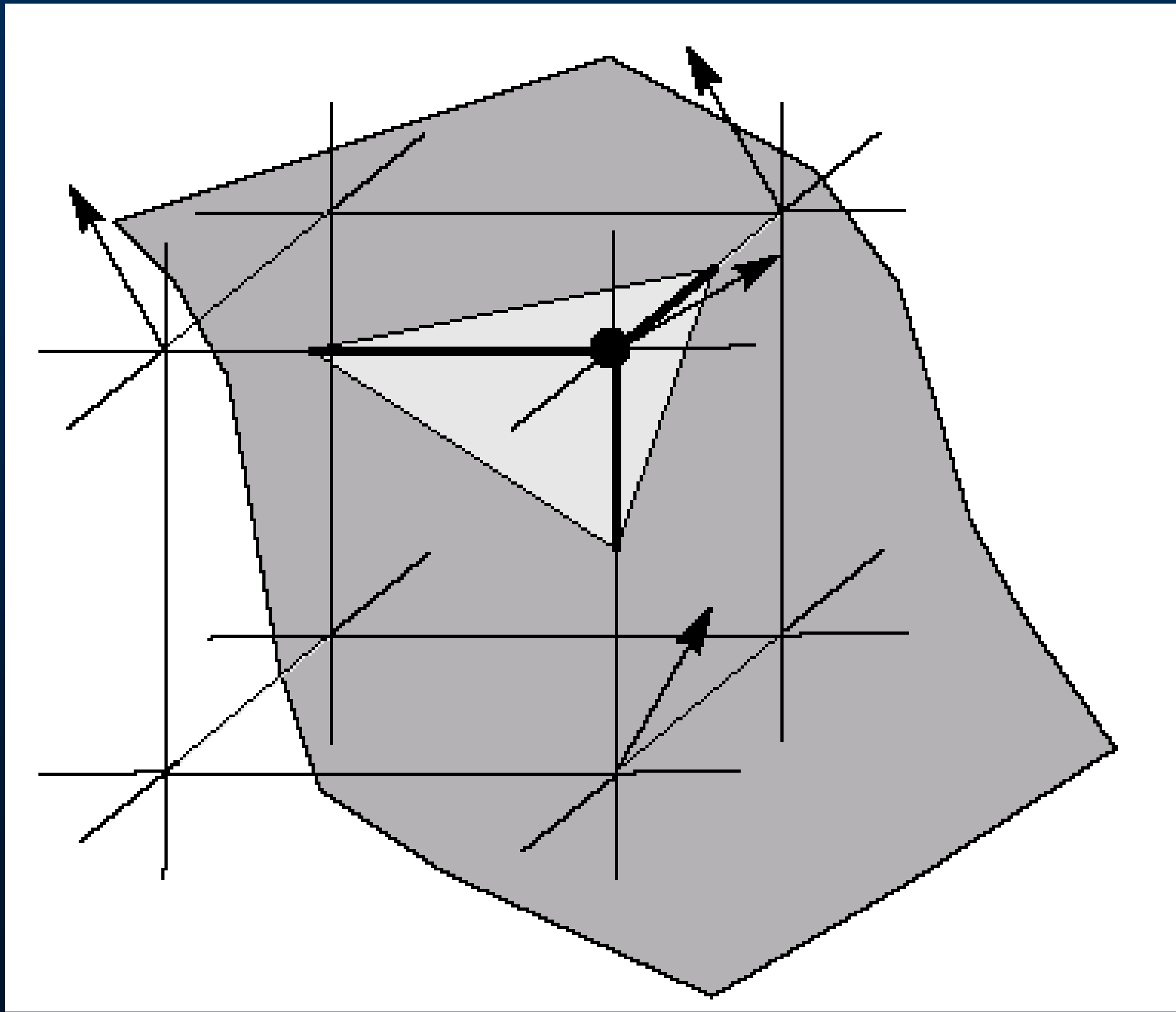
Step 6 - Calculate and interpolate normals

- For each triangle edge, find the vertex normals from the gradient of the density data using central differences

$$Gx = D(i + 1, j, k) - D(i - 1, j, k)$$

$$Gy = D(i , j + 1, k) - D(i , j - 1, k)$$

$$Gz = D(i , j, k + 1) - D(i , j, k - 1)$$



Marching Cubes

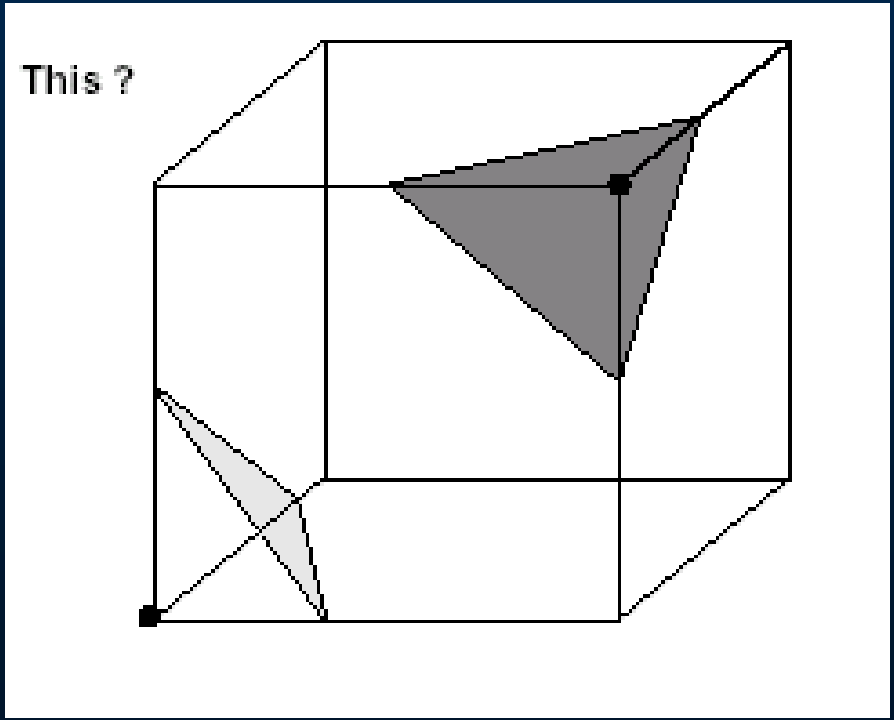
Extensions for Analysis

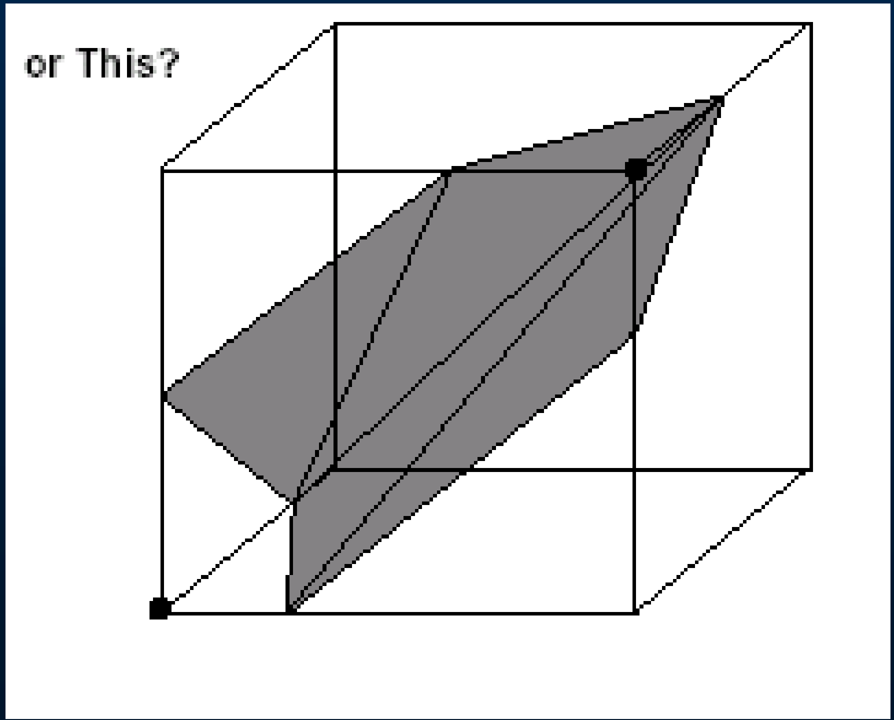
- Originally developed to produce surfaces for rendering
- Ambiguous cases can result in holes
- Many solutions proposed by many authors
  - Face patching
  - Tetrahedra
  - Function dependent triangulation

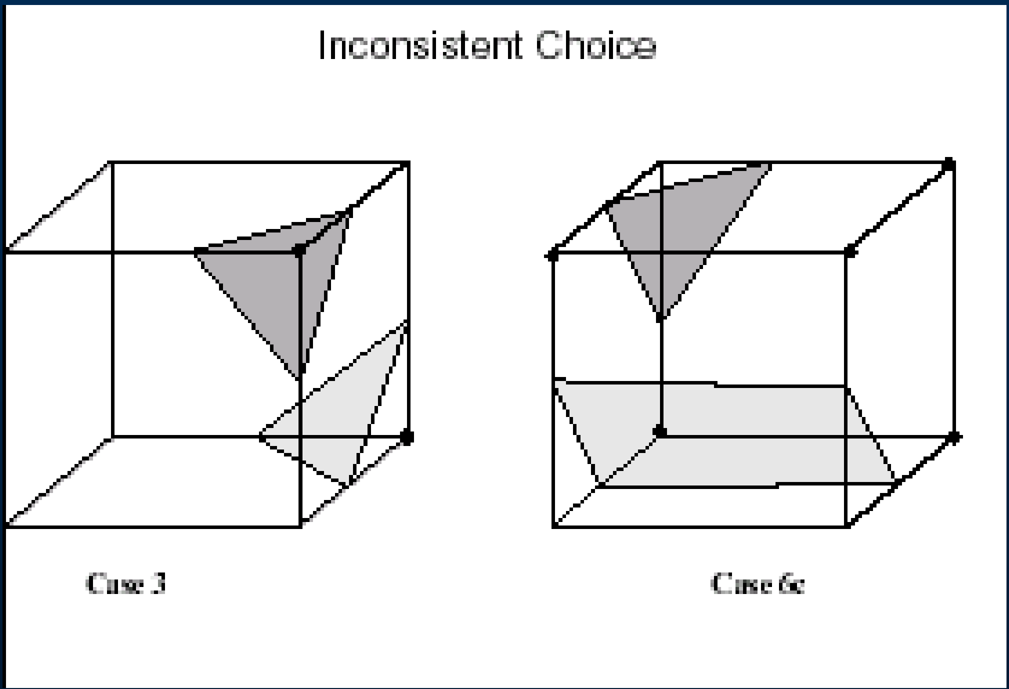
Marching Cubes

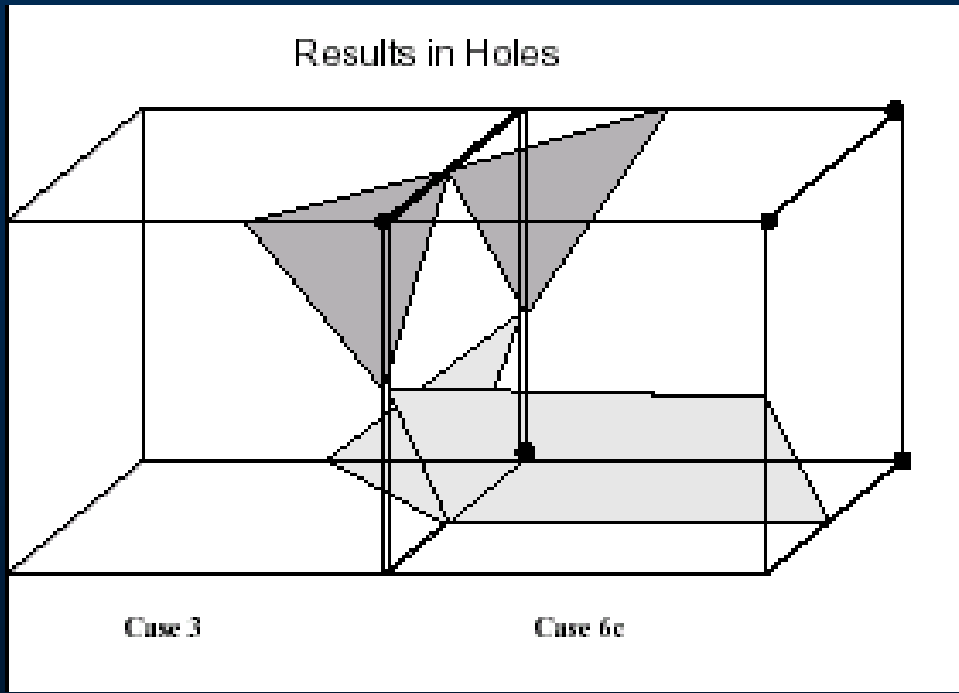
Ambiguous Cases

- Occur on any cube face that has adjacent vertices with different states, but diagonal vertices in same state
- There are six of these cases






Inconsistent Choice
Case 3
Case 6c

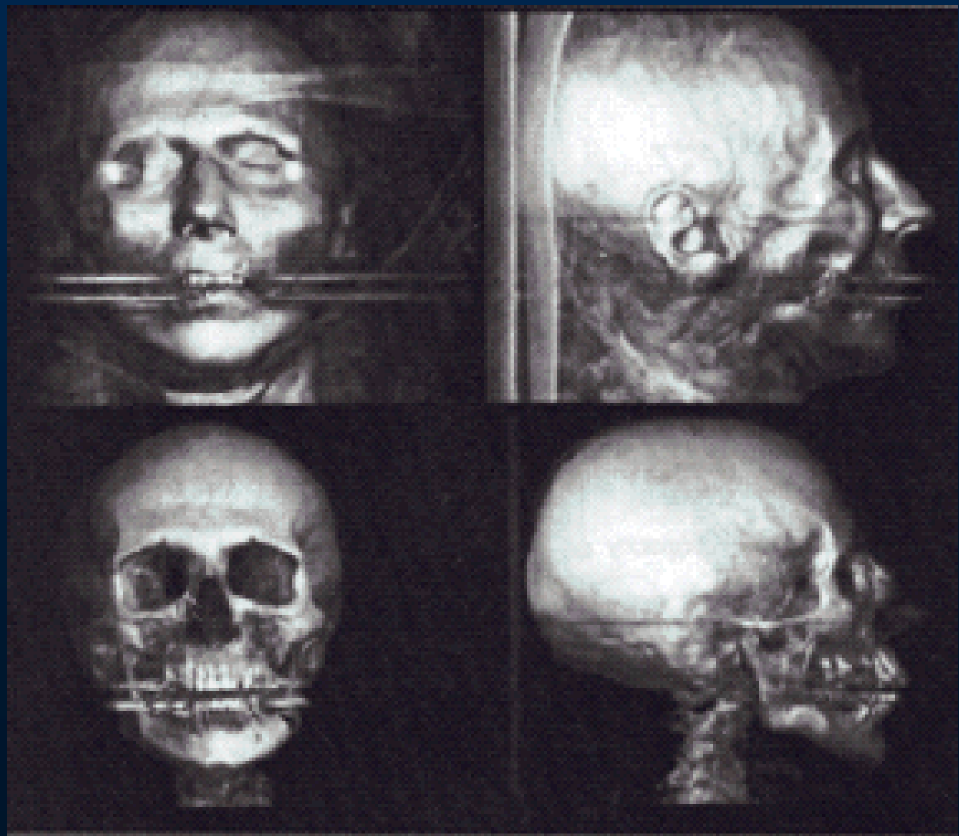

Results in Holes
Case 3
Case 6c

Volume Rendering

- Direct projection
- Translucent gel
- Information inside objects
- Discrete
- Large datasets
- Slow
- Classification
- Compositing

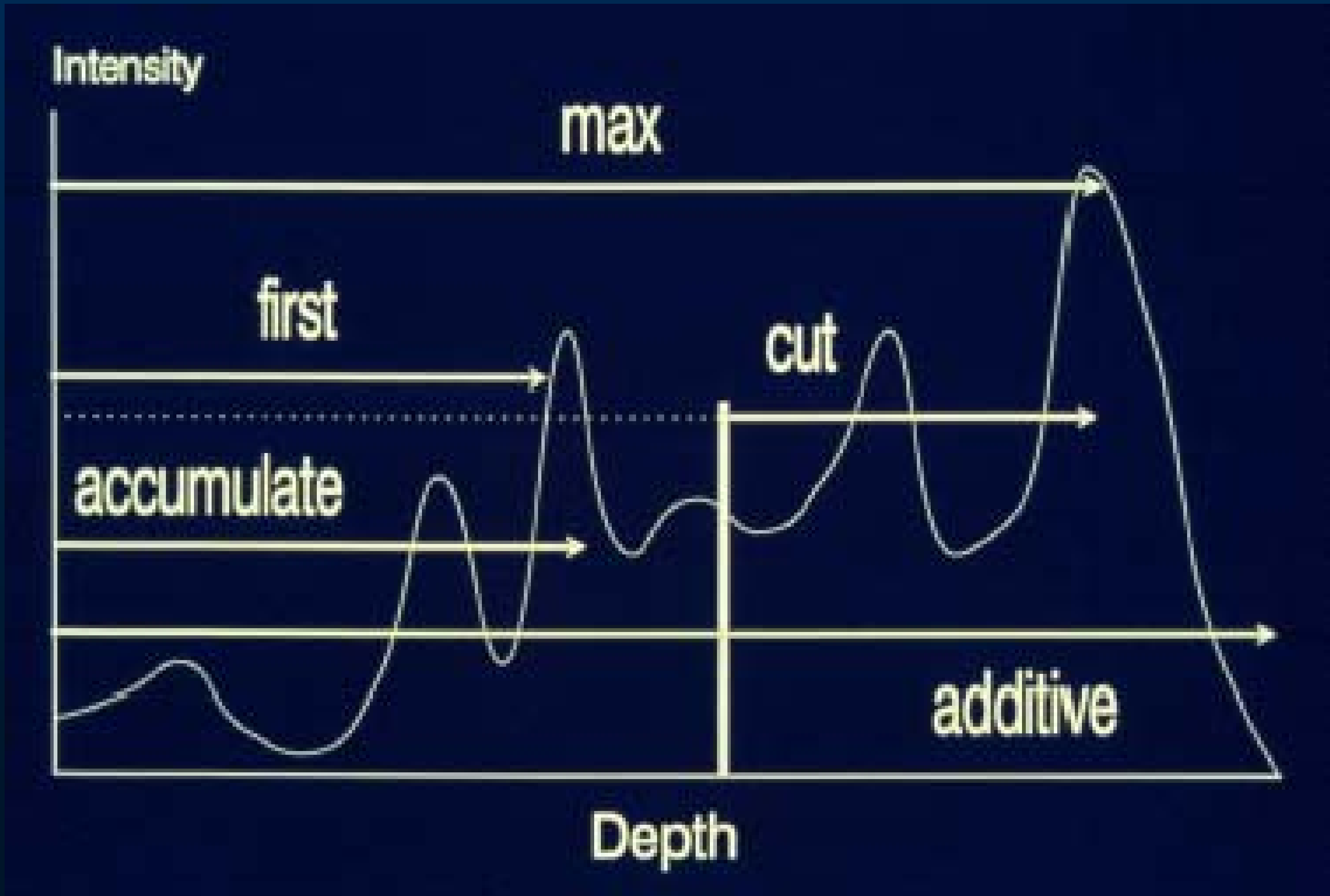
Direct Volume Rendering

- Ray Casting
- Levoy 89 CG&A

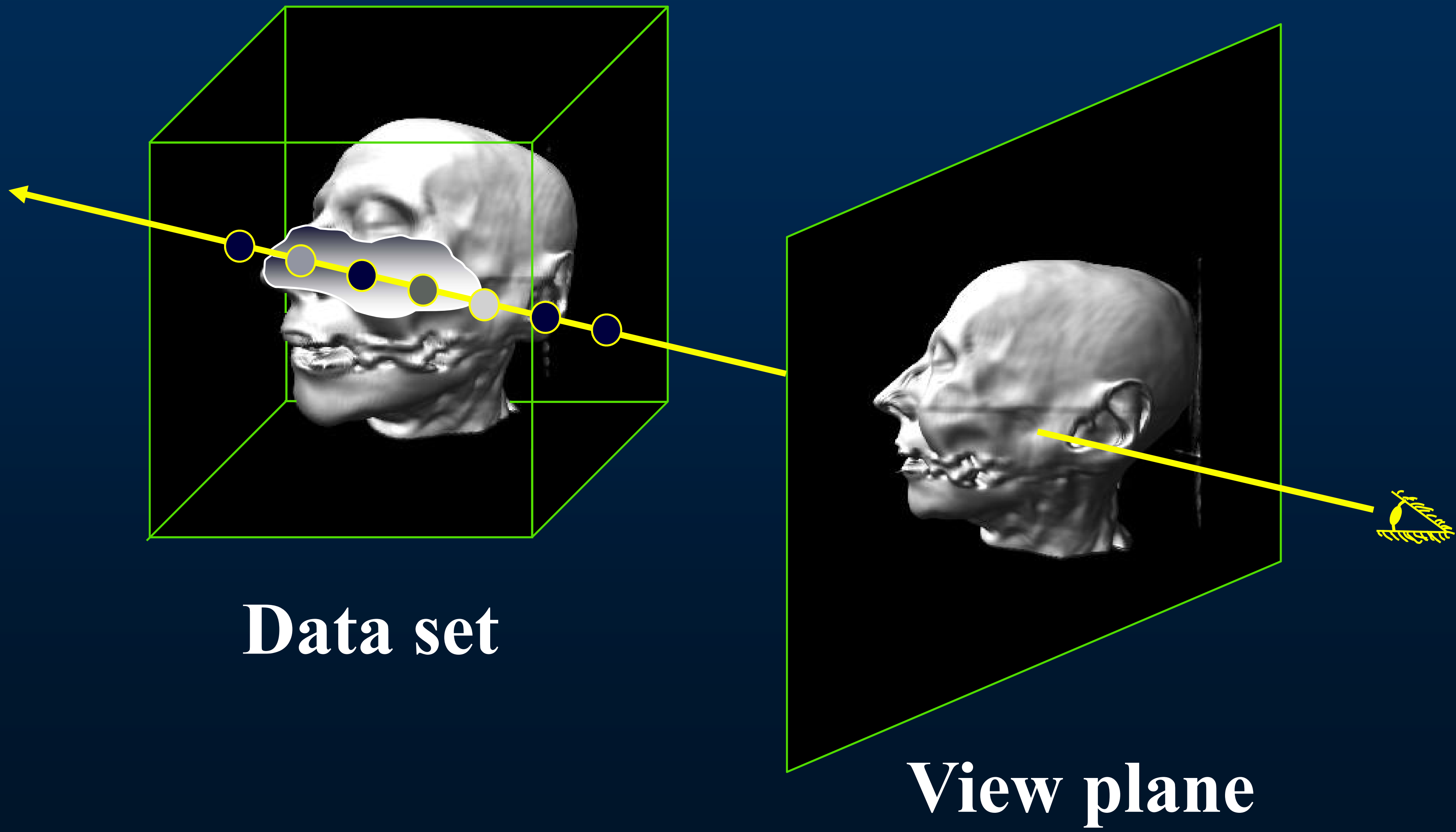


Ray Traversal Methods

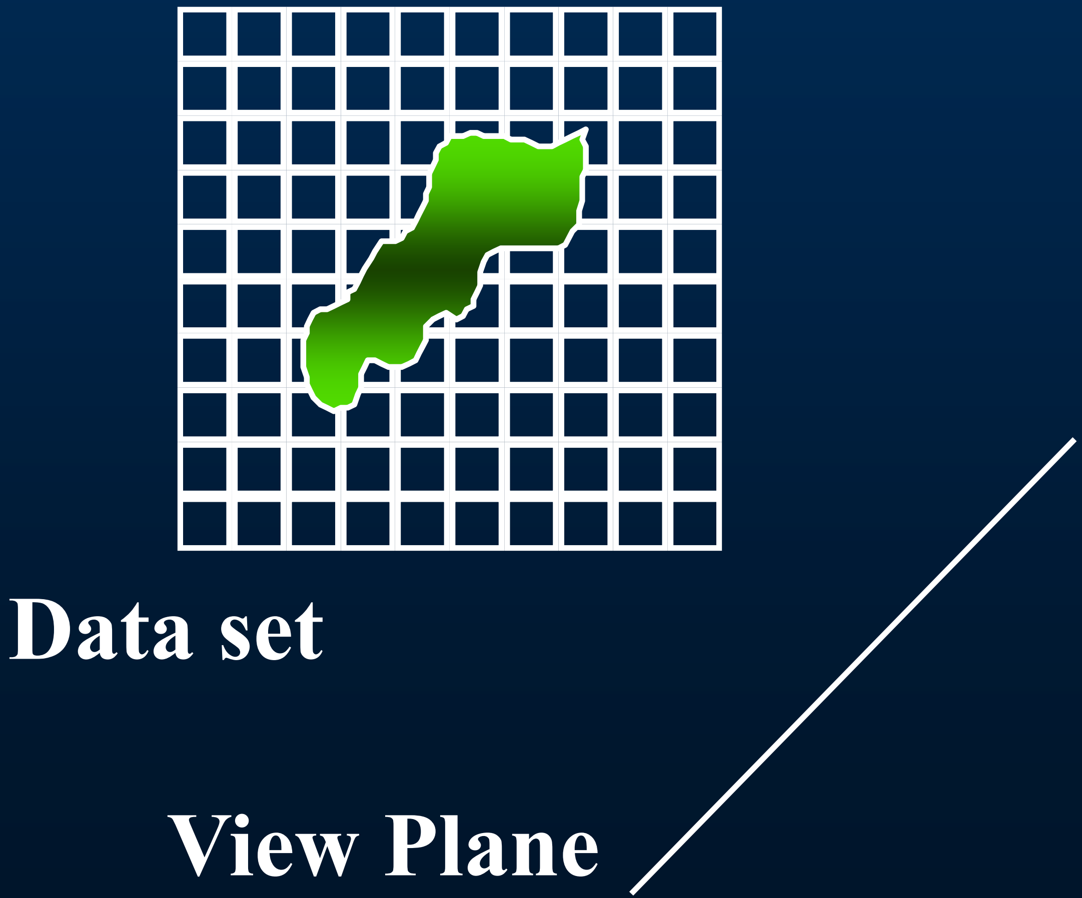
- **Around that time:**



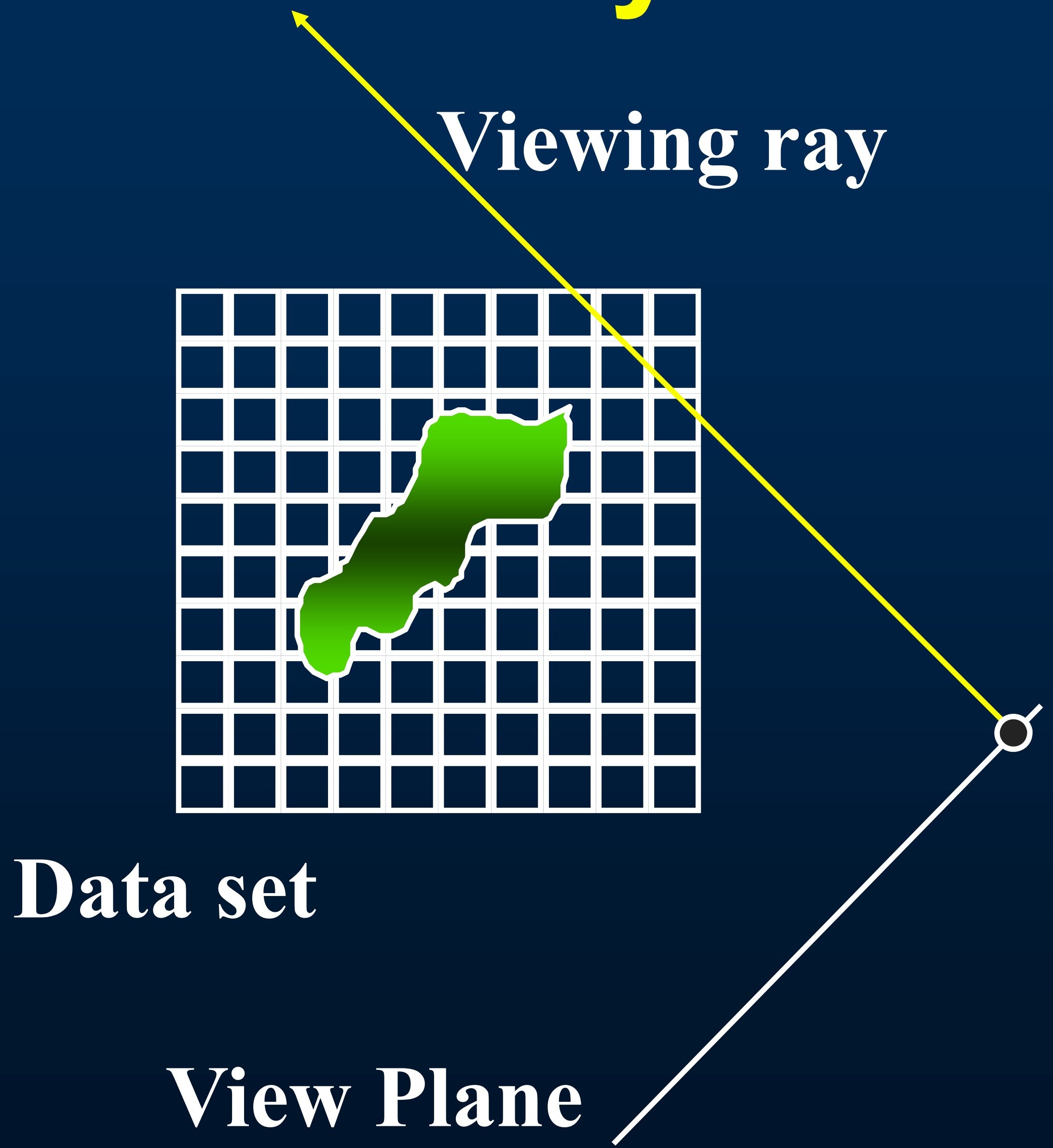
Volumetric Ray-Casting



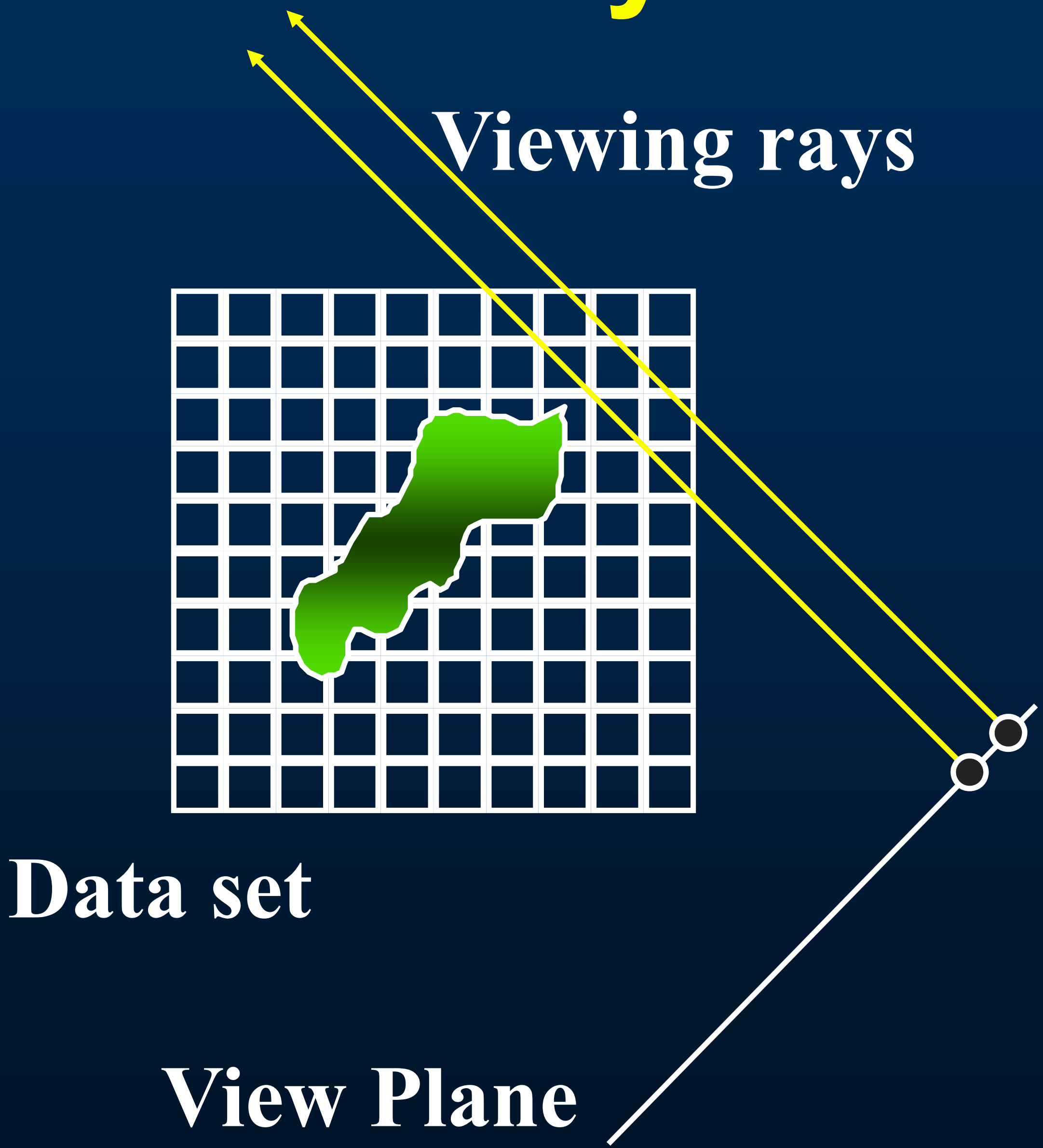
Basic Ray-Casting Algorithm



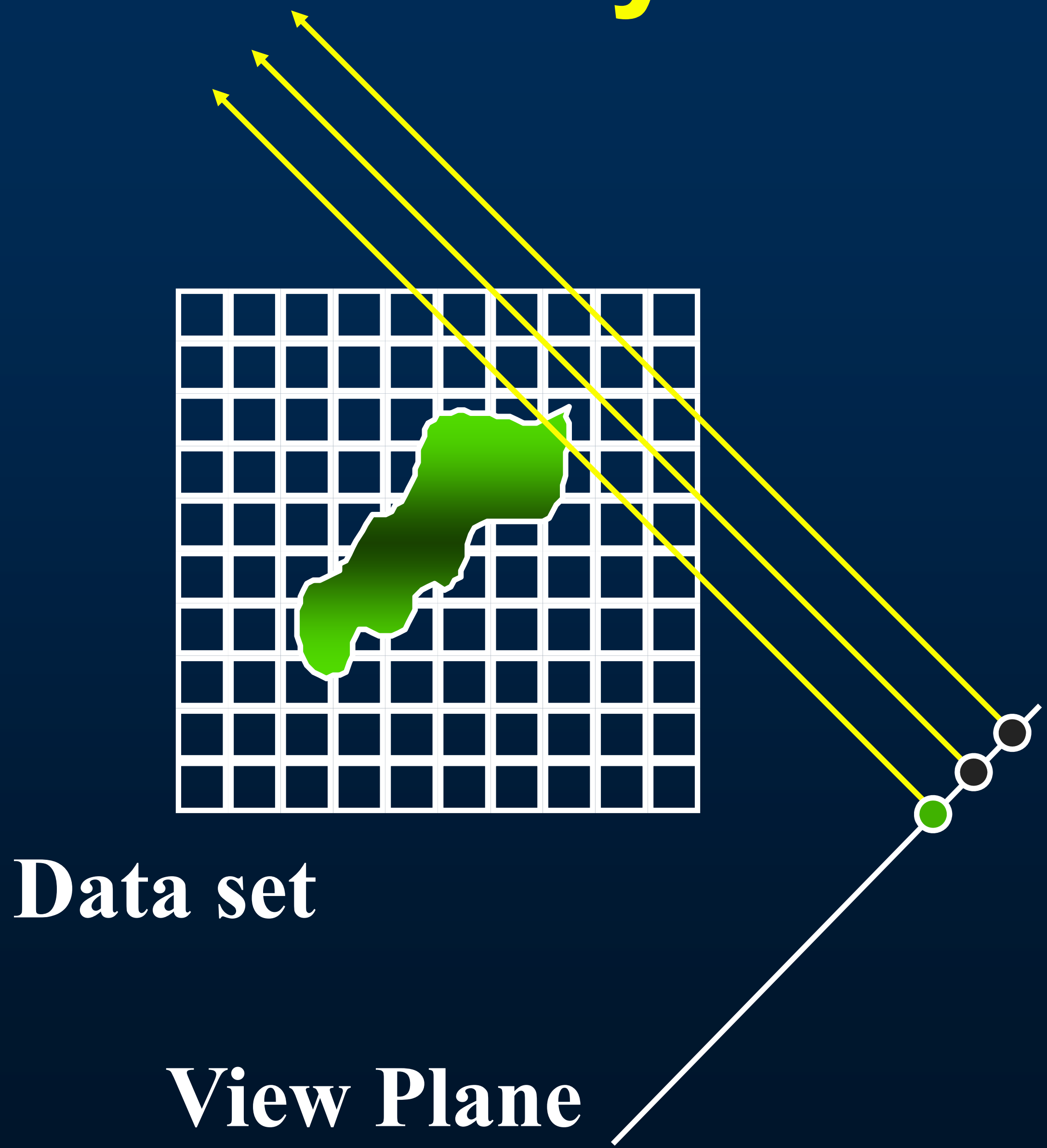
Basic Ray-Casting Algorithm



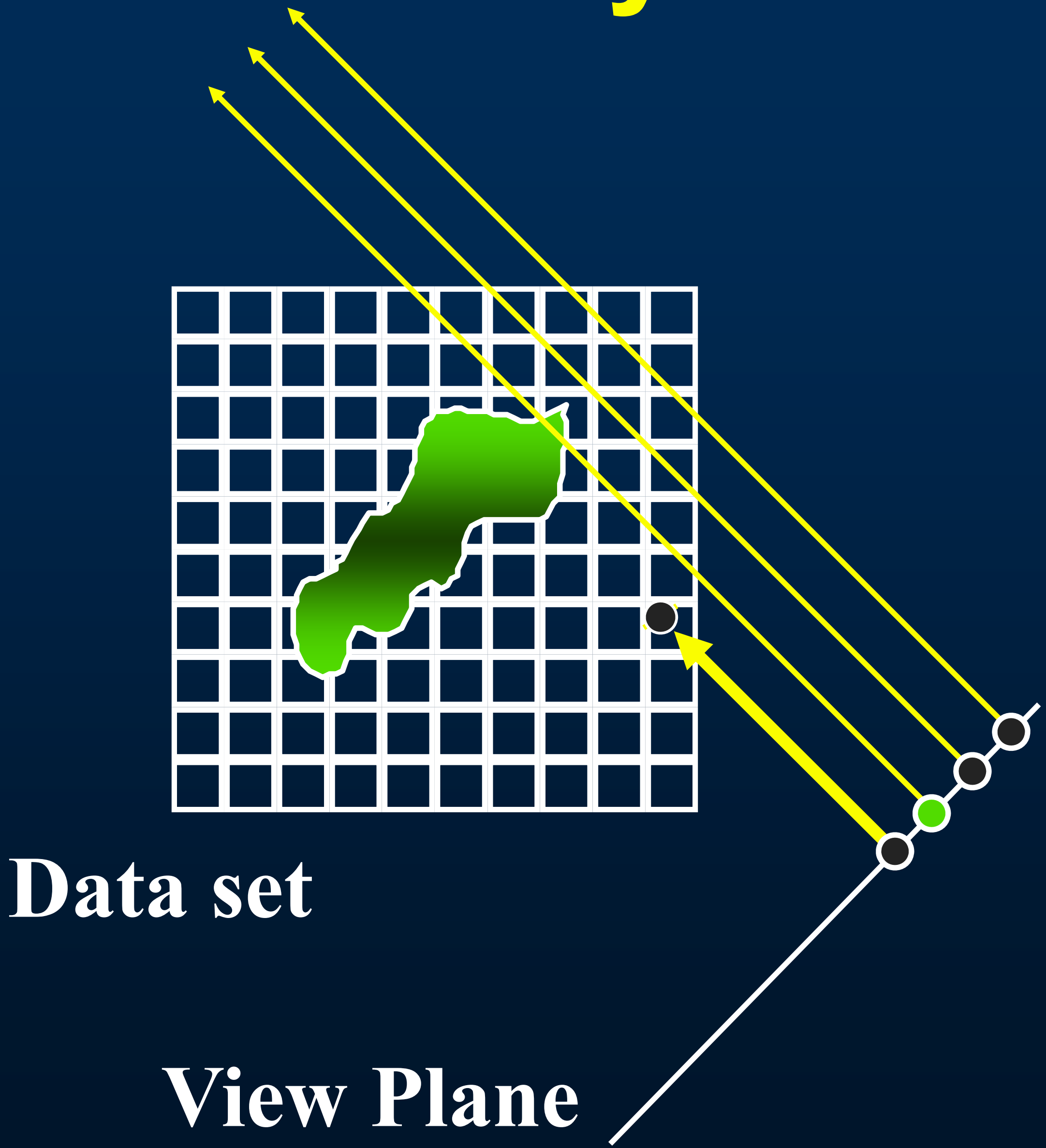
Basic Ray-Casting Algorithm



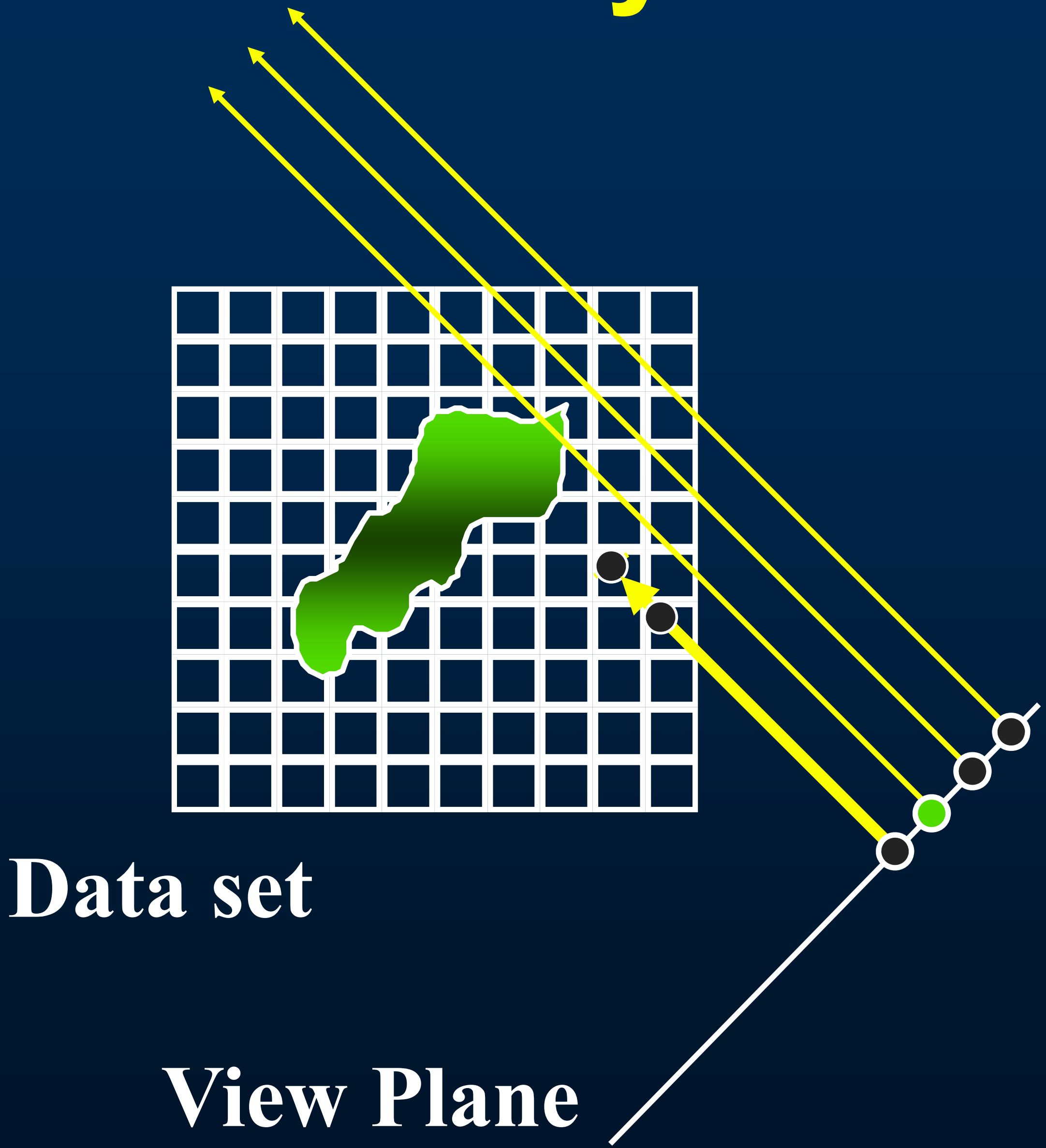
Basic Ray-Casting Algorithm



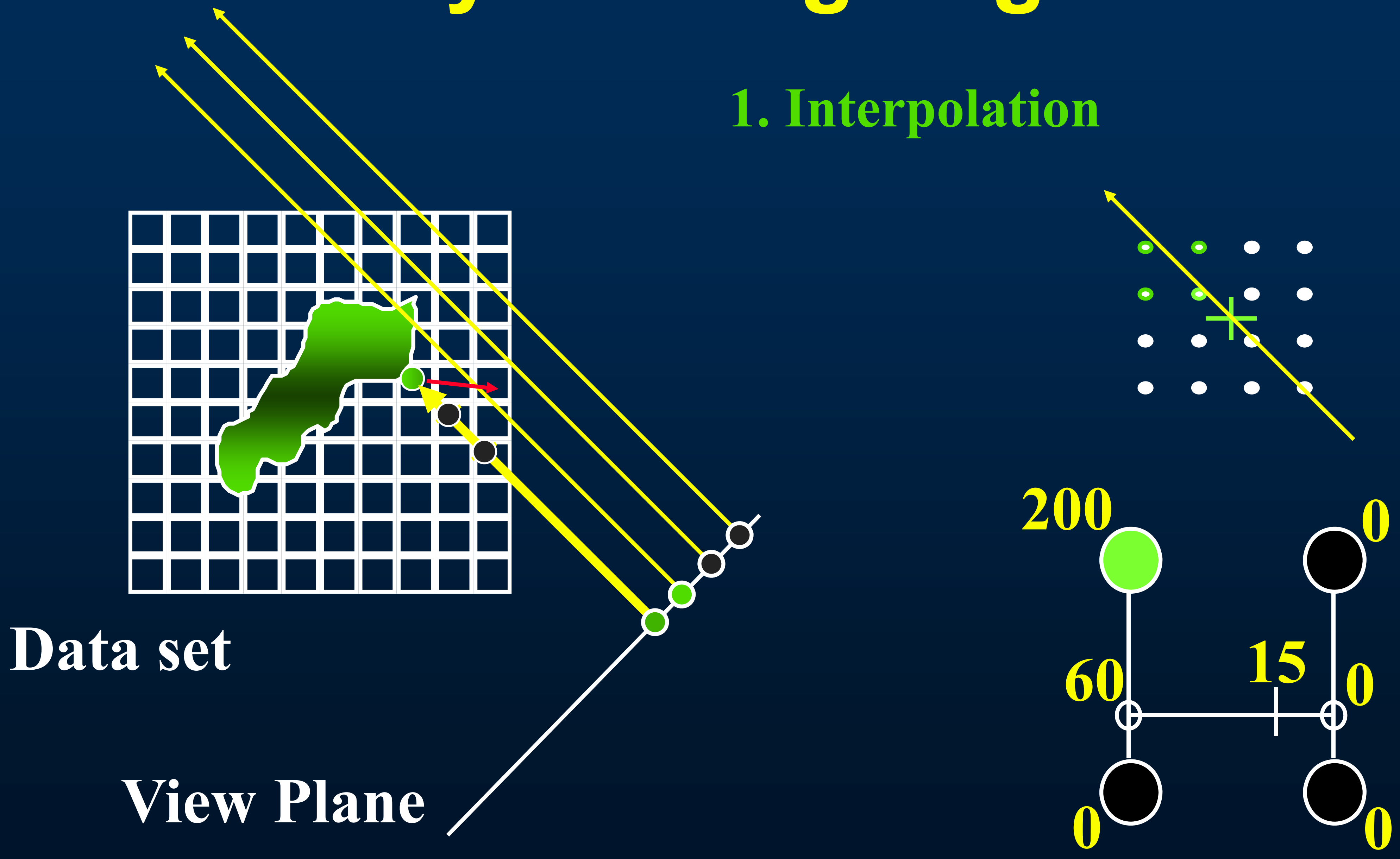
Basic Ray-Casting Algorithm



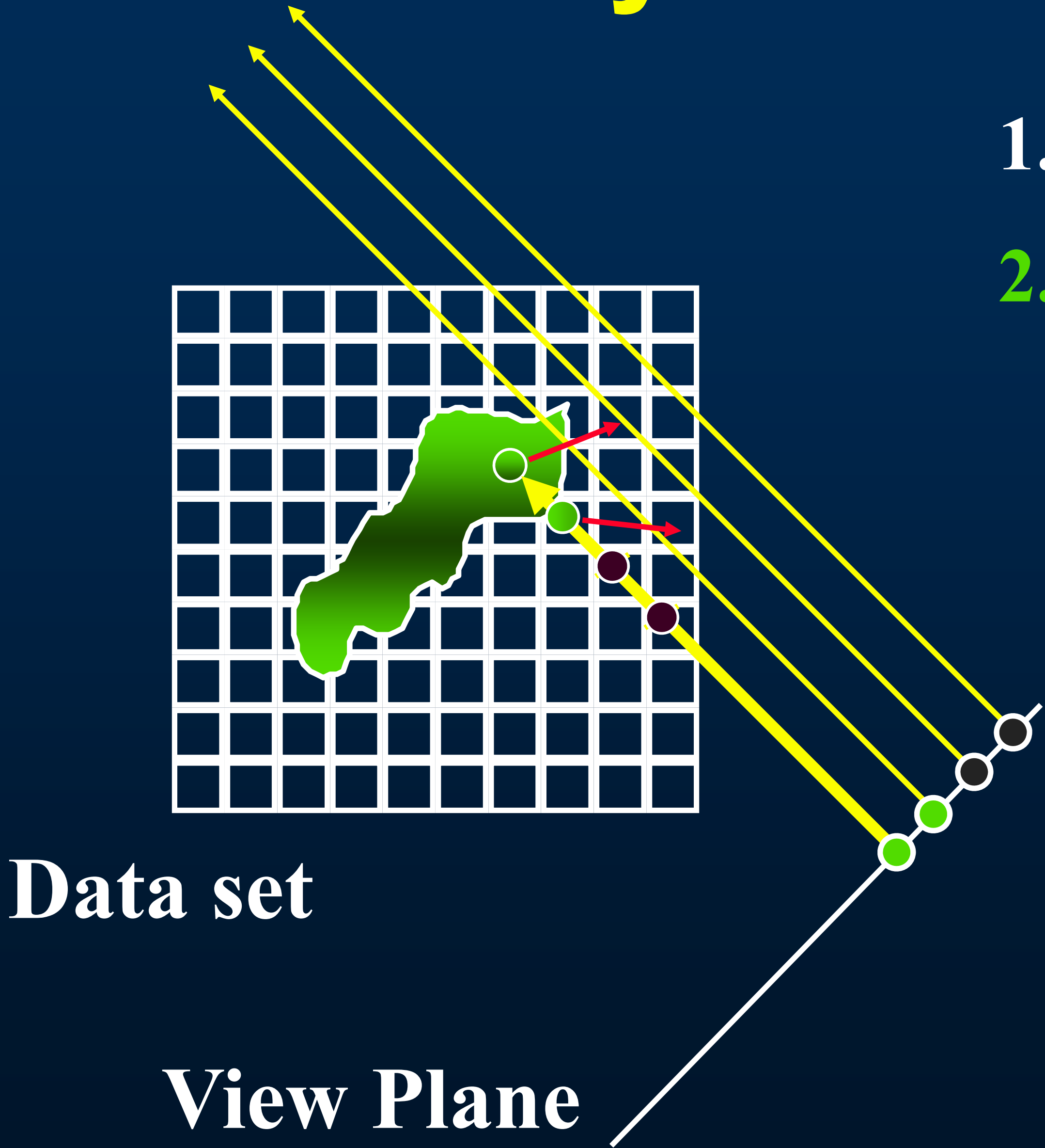
Basic Ray-Casting Algorithm



Basic Ray-Casting Algorithm

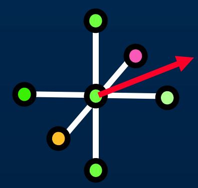


Basic Ray-Casting Algorithm



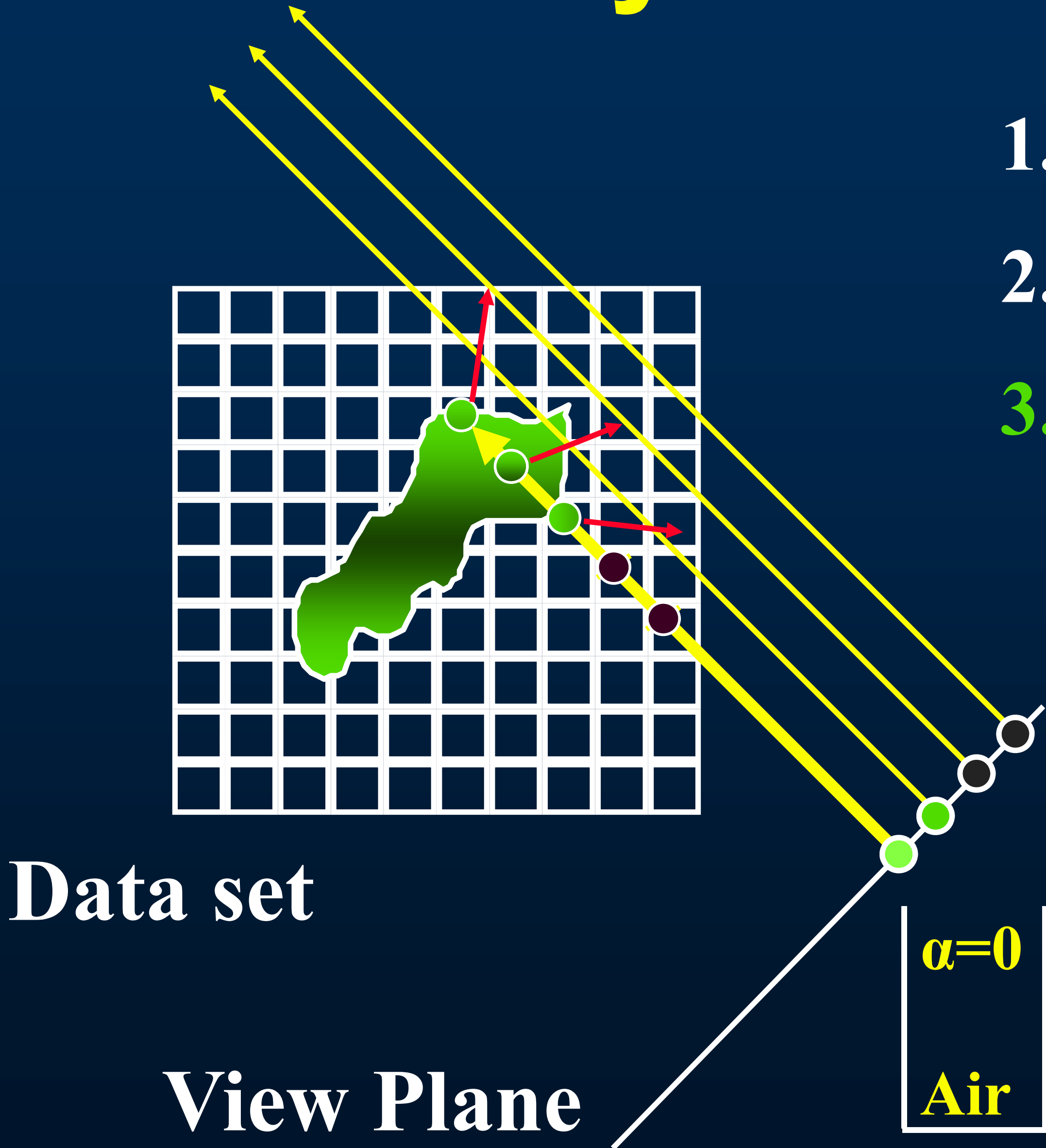
1. Interpolation

2. Gradient estimation

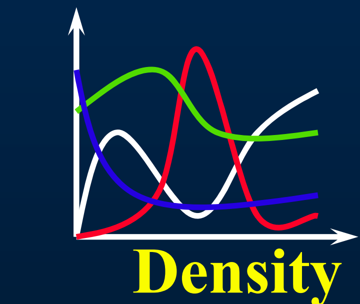


Estimated Gradient
= (Δx, Δy, Δz)

Basic Ray-Casting Algorithm



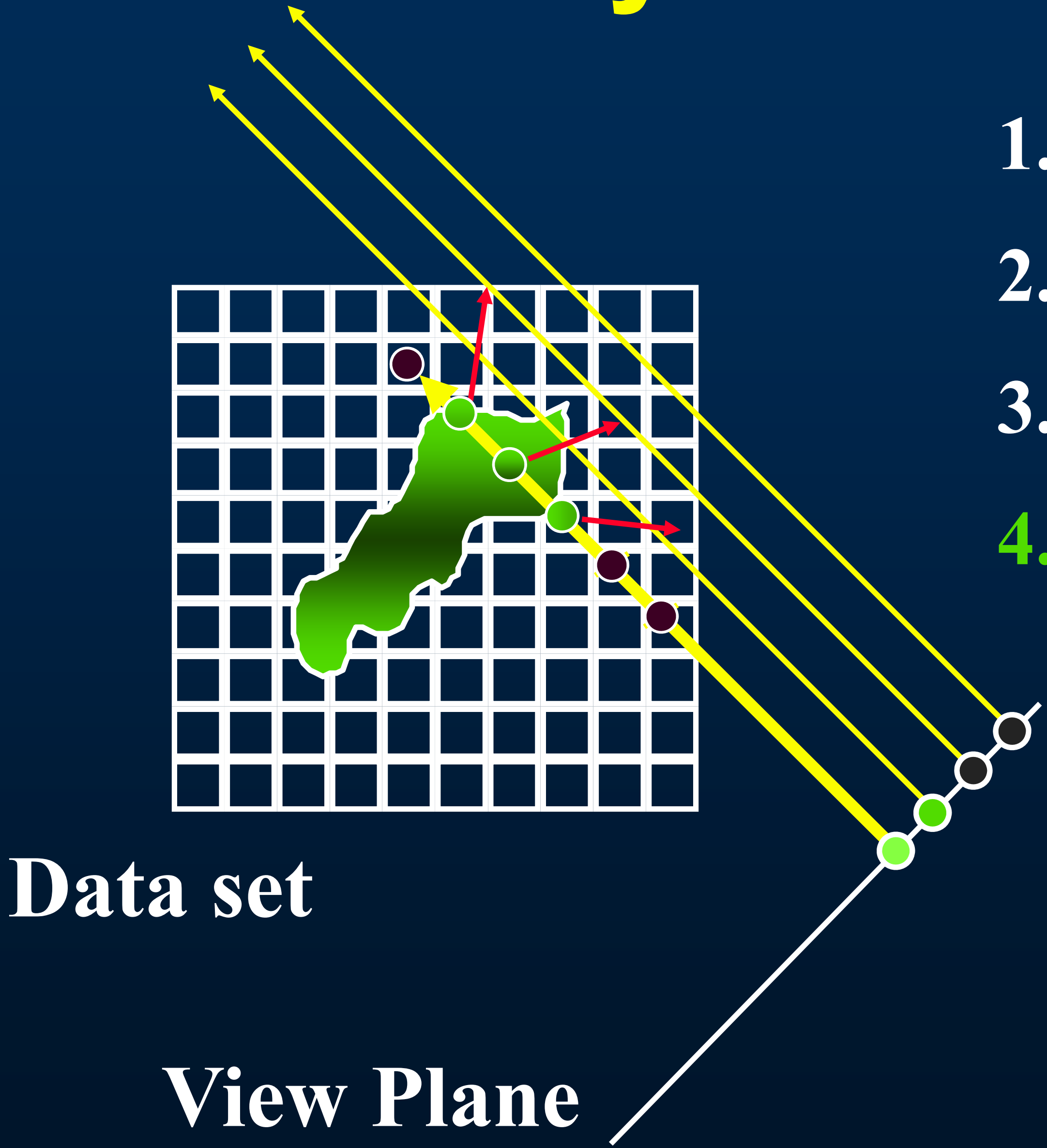
1. Interpolation
2. Gradient estimation
3. Classification



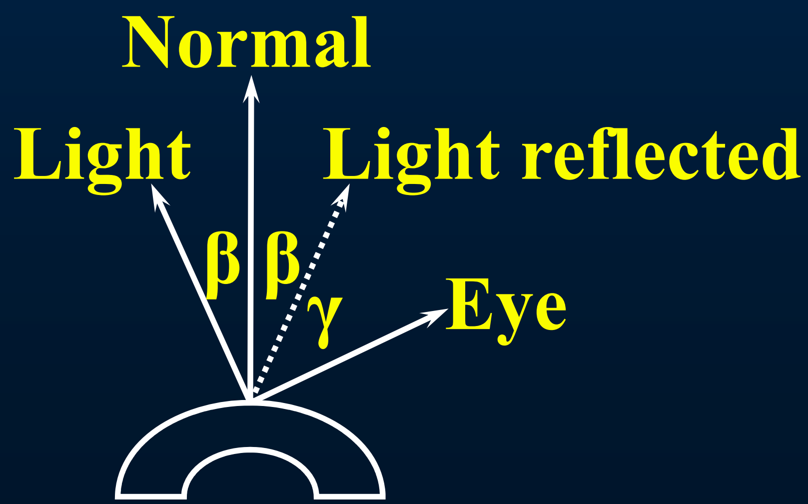
α=0	α=1 gray	α=.2 pink	α=.5 red
Air	Bone	Muscle	Blood

0 30 100 130 255

Basic Ray-Casting Algorithm

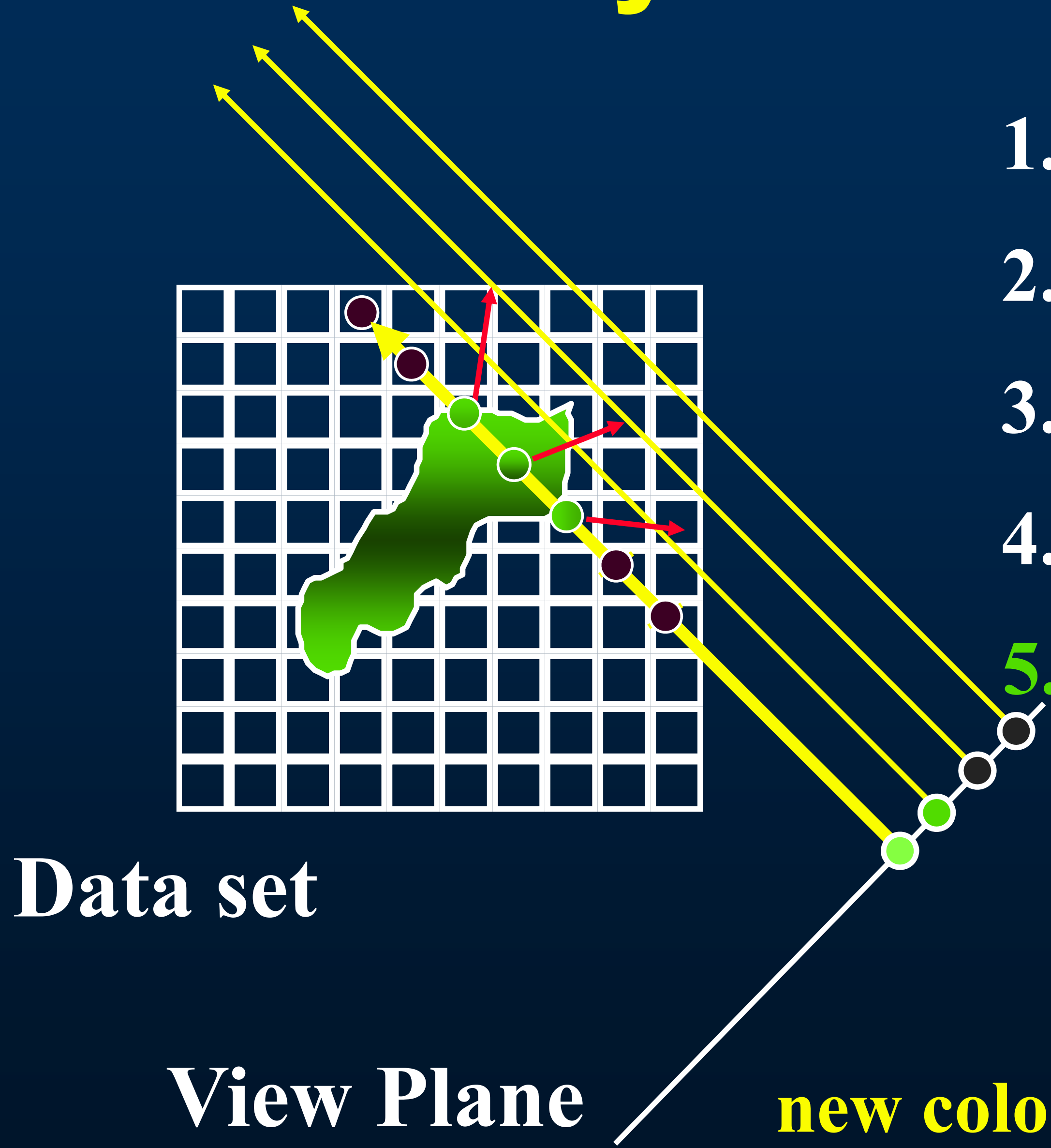


1. Interpolation
2. Gradient estimation
3. Classification
4. Shading



New Intensity = Intensity • cos(γ)

Basic Ray-Casting Algorithm

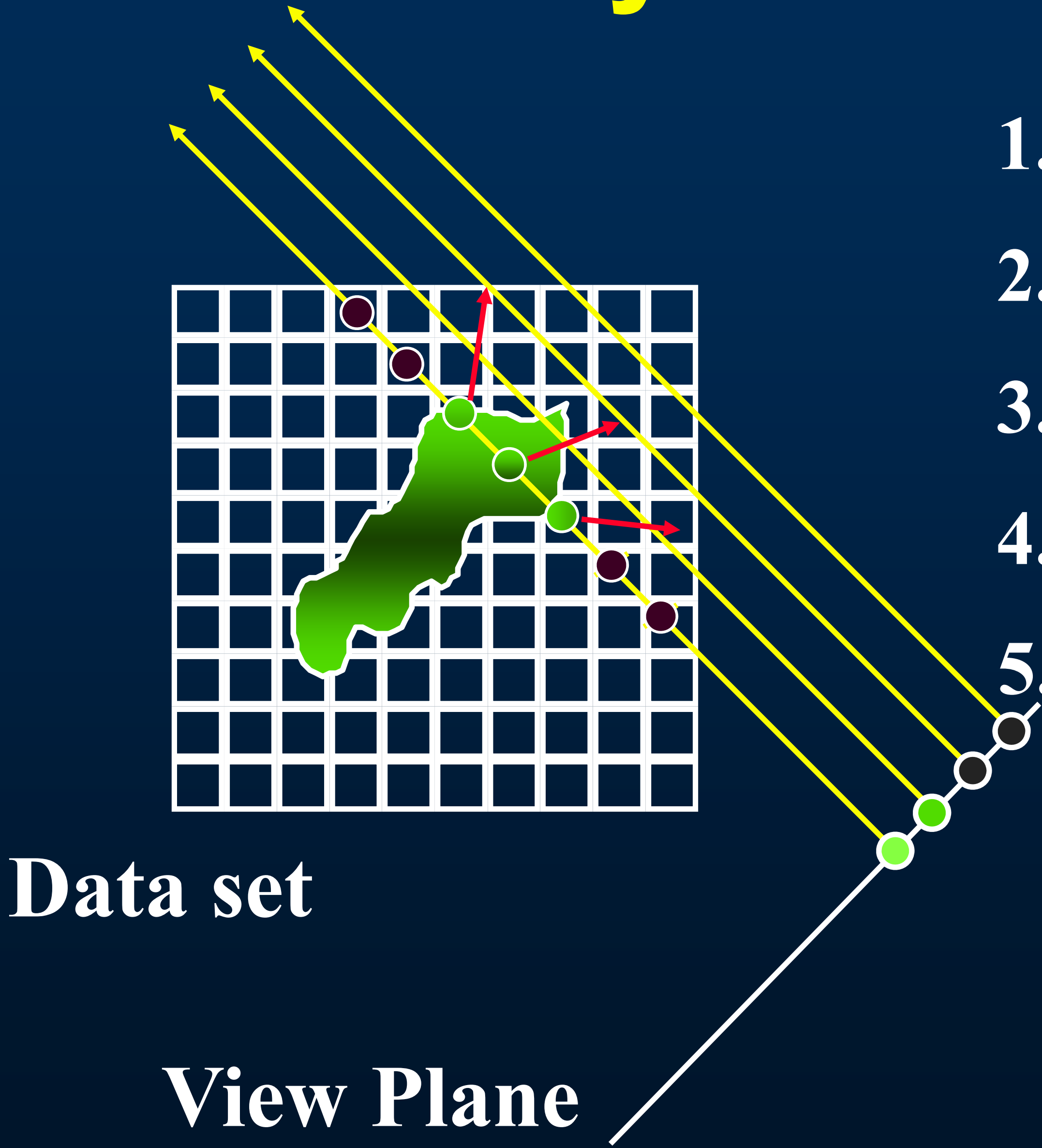


1. Interpolation
2. Gradient estimation
3. Classification
4. Shading
5. Compositing

Back-to-Front compositing :

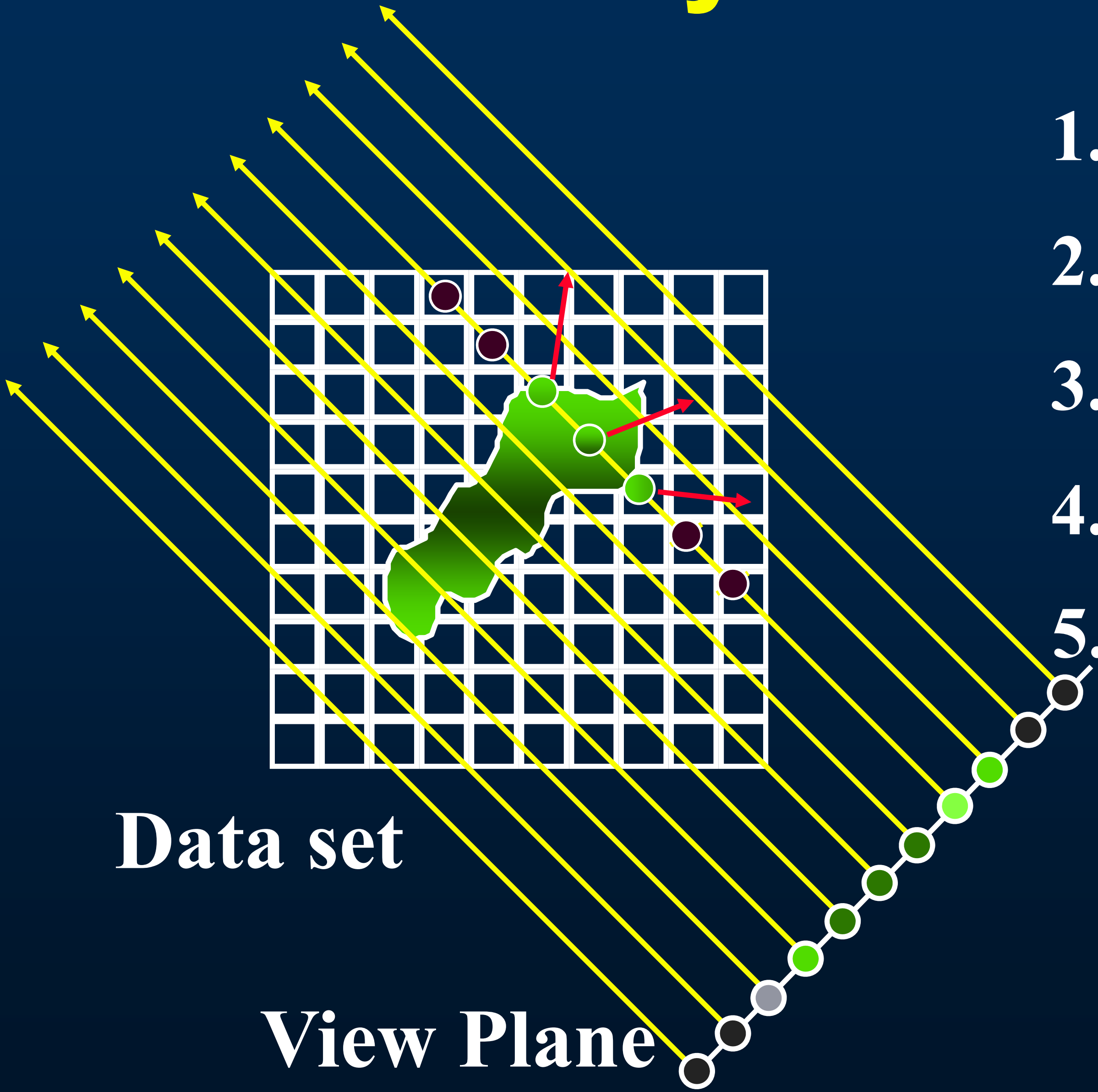
new color = front color • front α
+ back color • (1 - front α)

Basic Ray-Casting Algorithm



1. Interpolation
2. Gradient estimation
3. Classification
4. Shading
5. Compositing

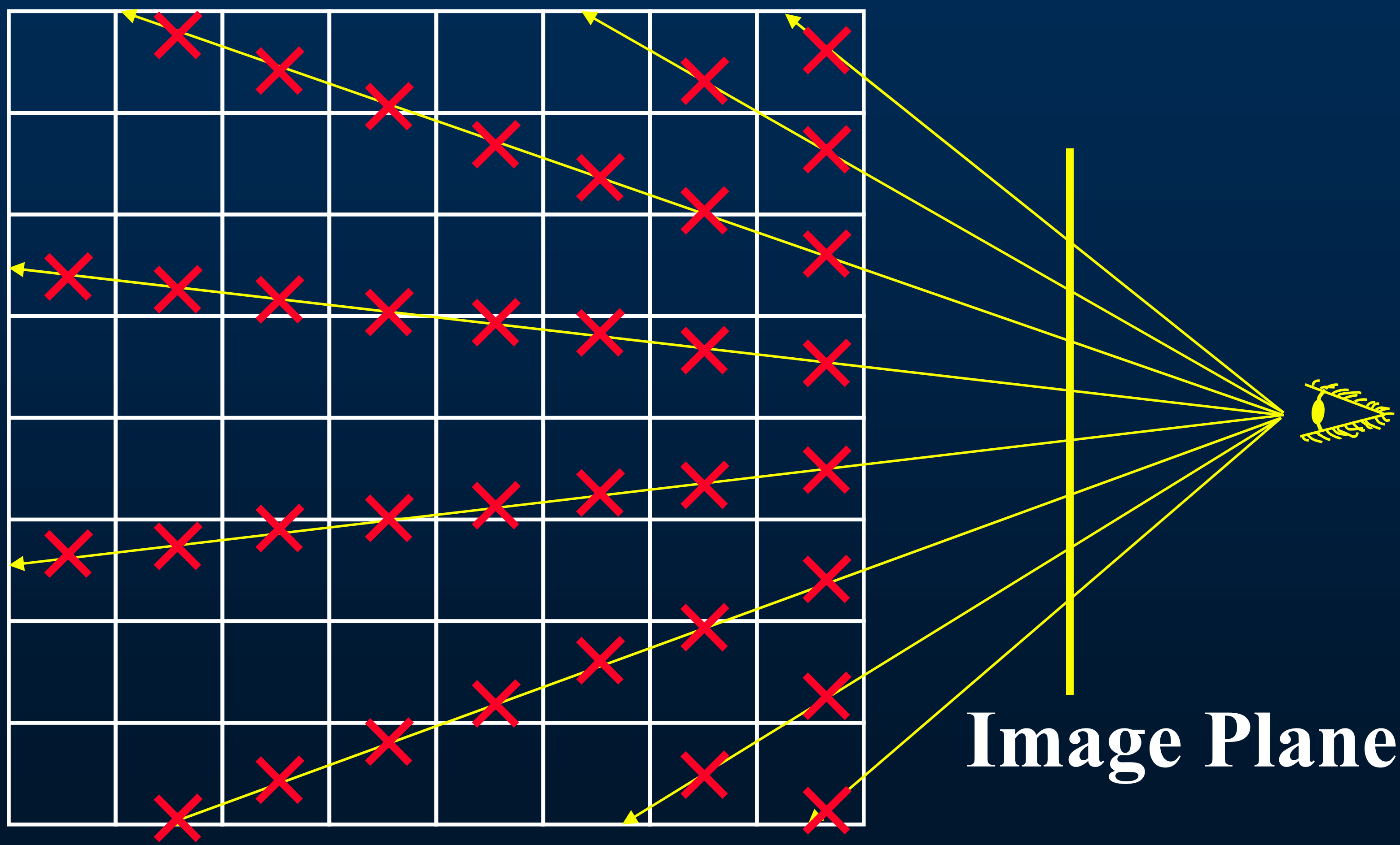
Basic Ray-Casting Algorithm



1. Interpolation
2. Gradient estimation
3. Classification
4. Shading
5. Compositing

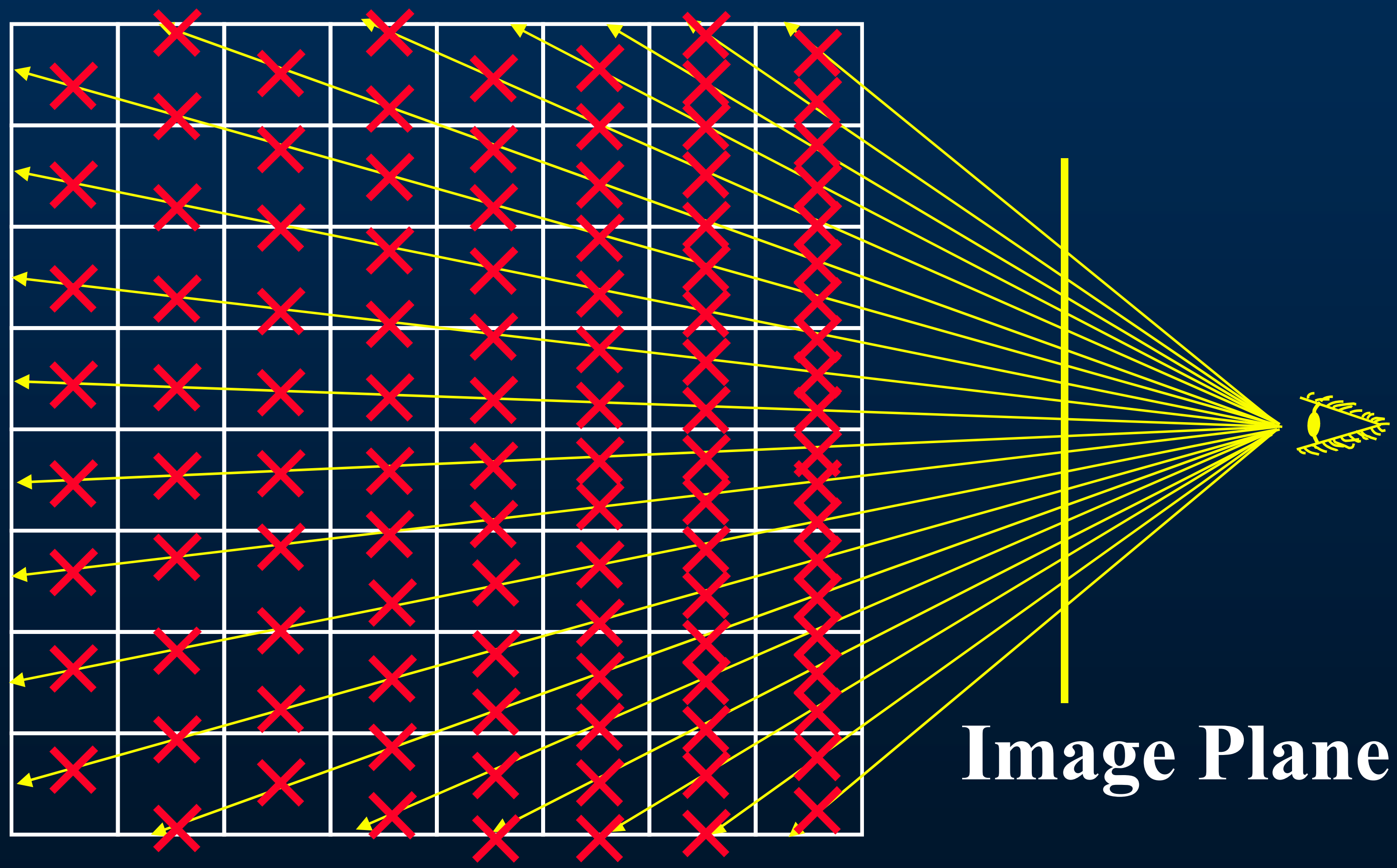
Perspective Projection

Aliasing!



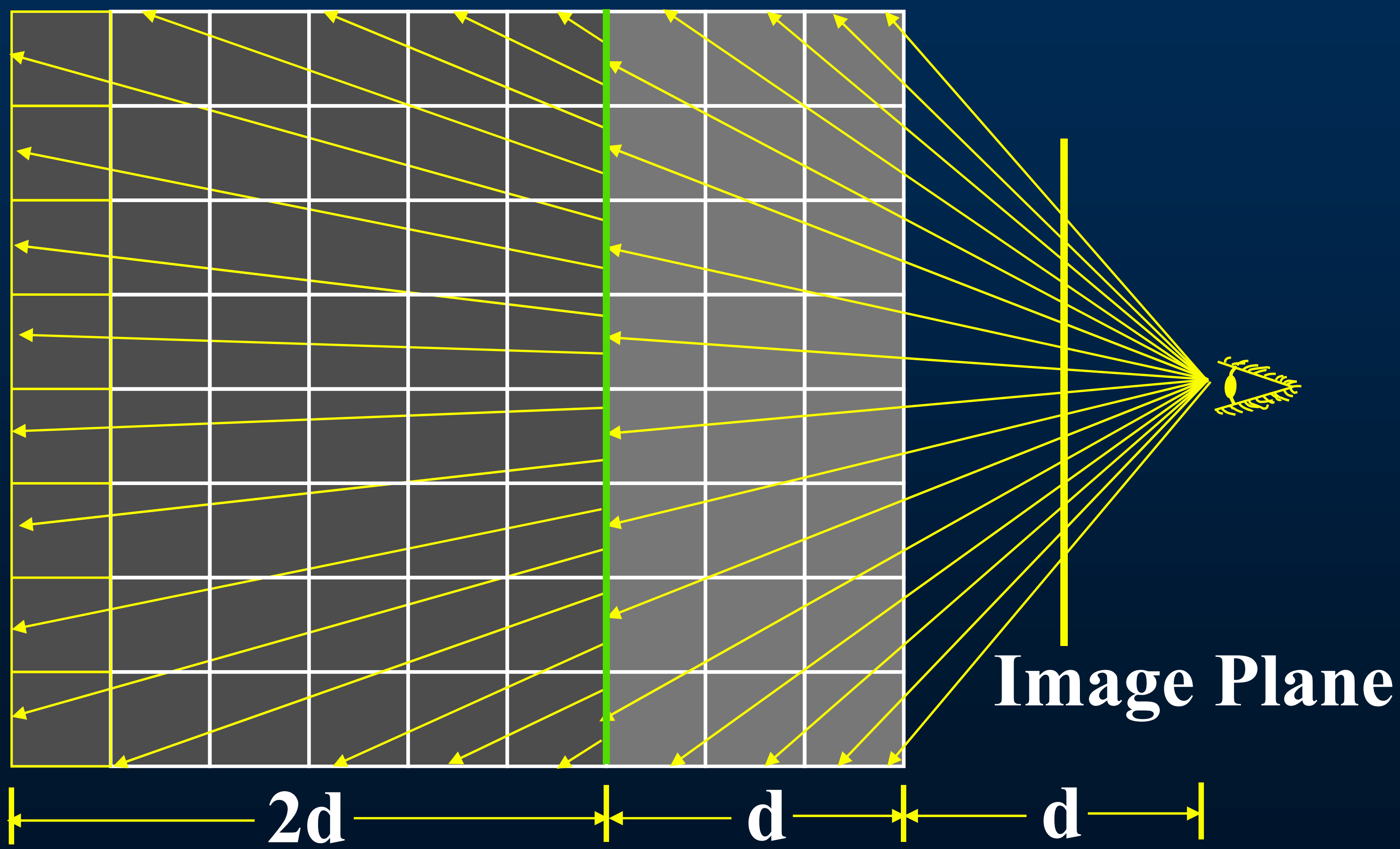
Supersampling

Too Expensive!

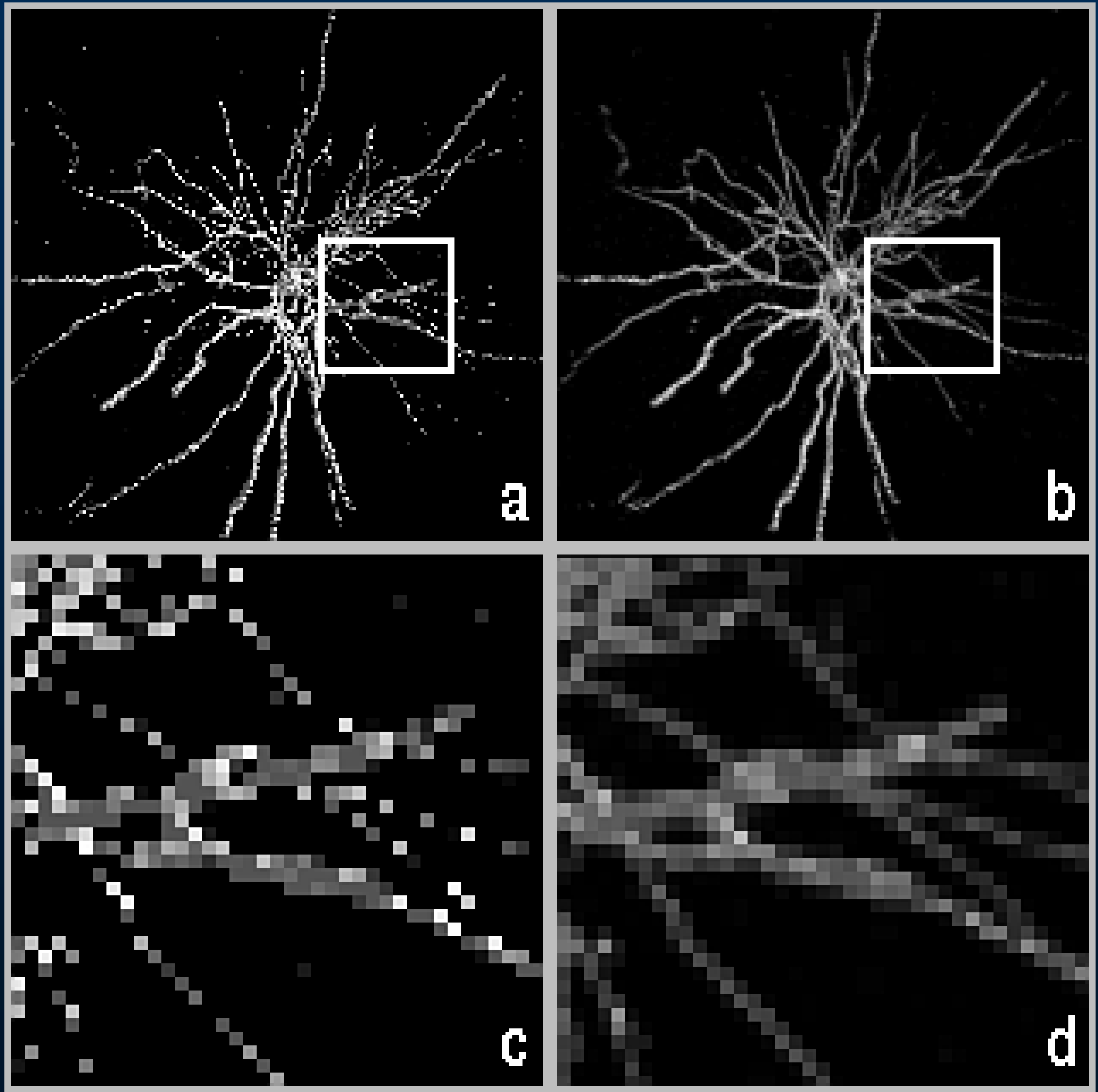


Adaptive Sampling

Kreeger, Dachille, Chen, Bitter, Kaufman, VolVis 98



Perspective Projection



LGN Nerve Cell

a) Undersampling

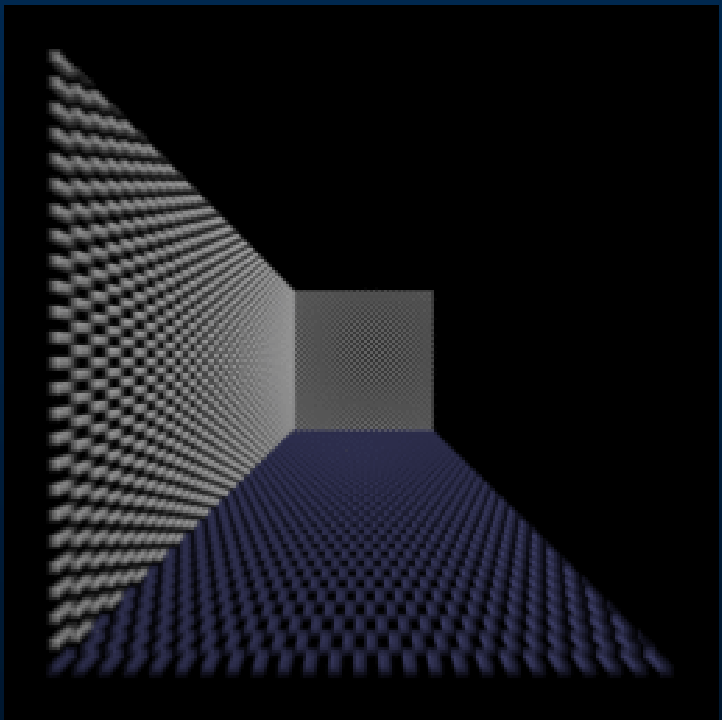
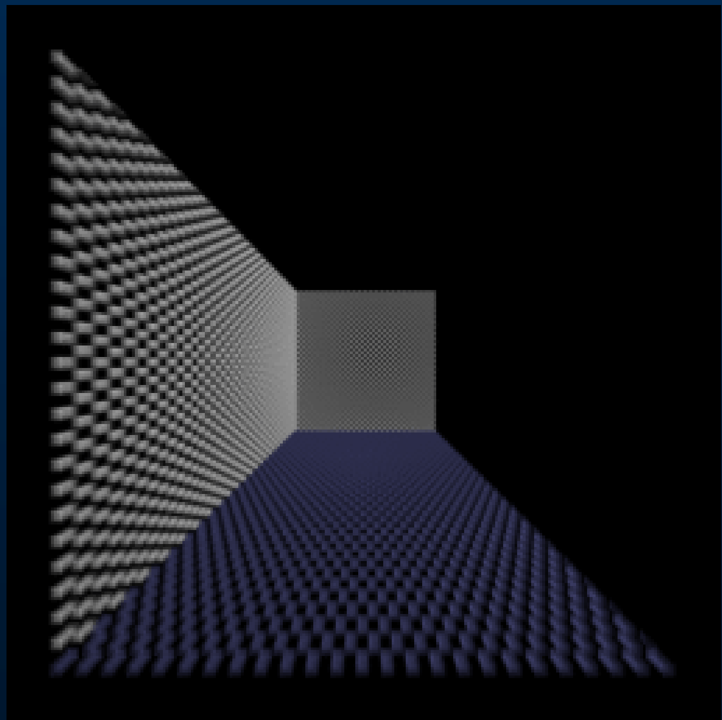
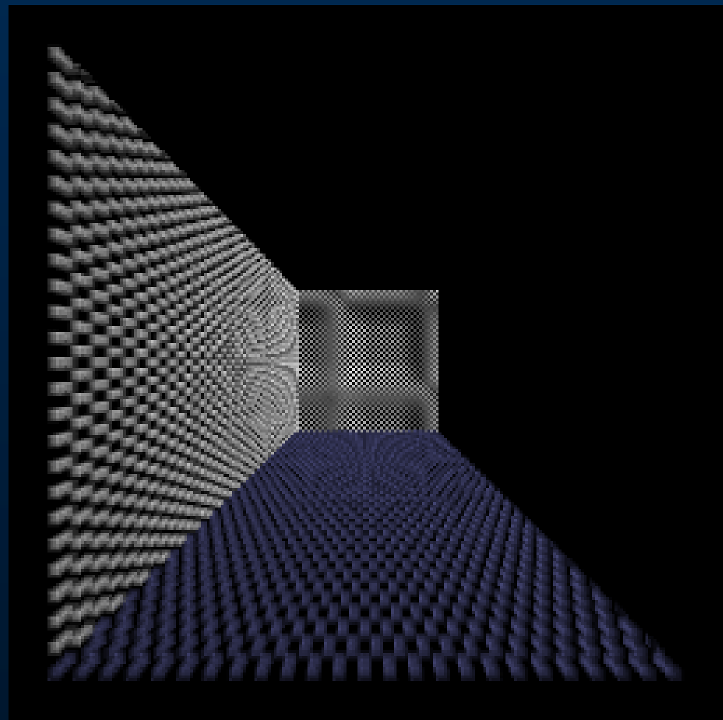
b) Adaptive Sampling

c) Undersampling Zoom

d) Adaptive Sampling Zoom

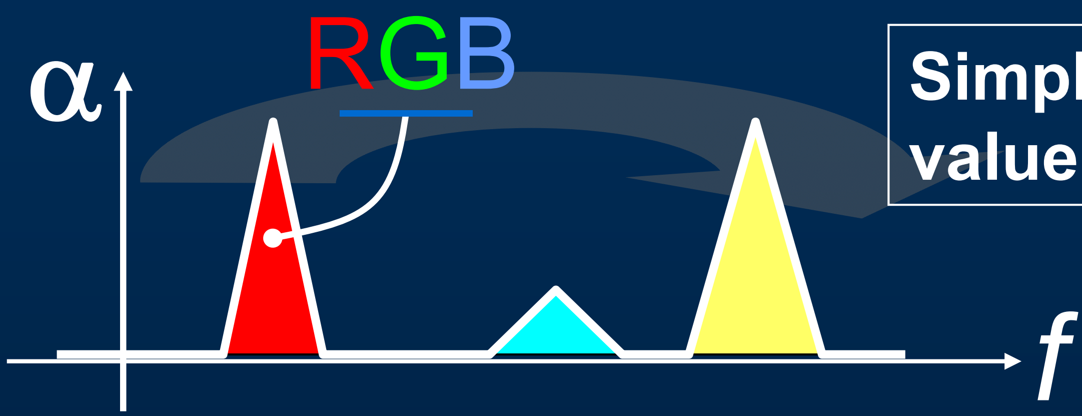
Perspective Projection

Oversampling **Adaptive Sampling** **Undersampling**

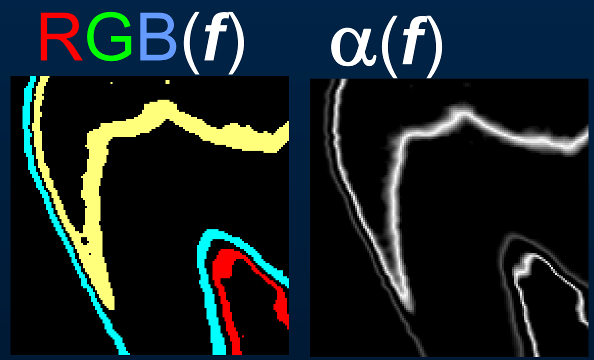
5^3 checker box room (128^3 volume)

Transfer Functions (TFs)

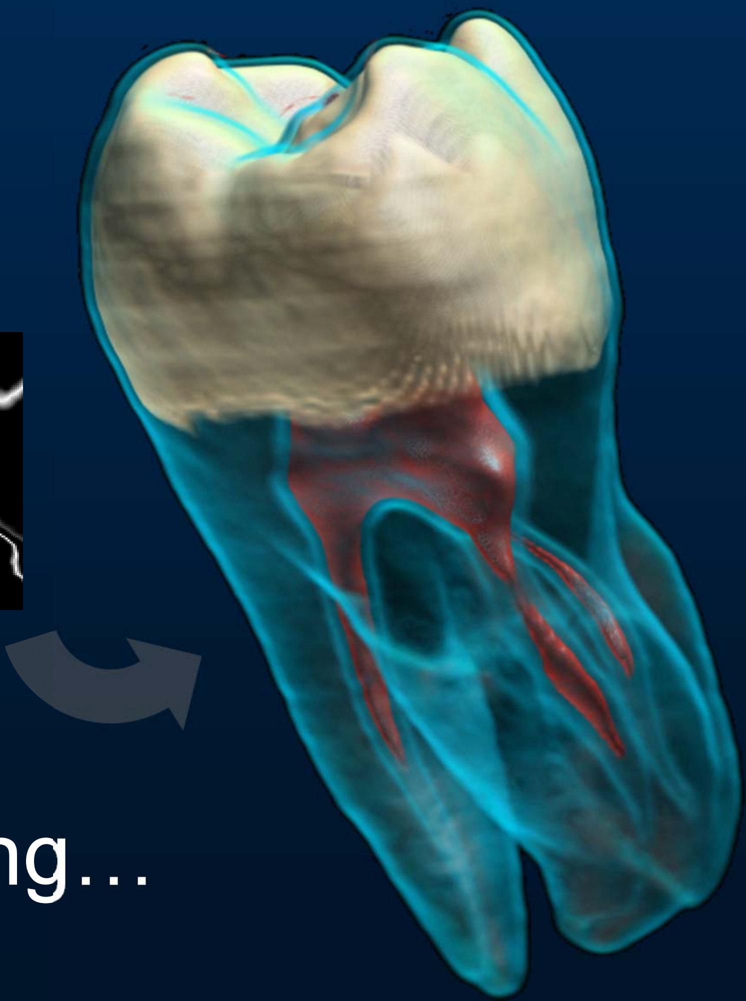


Simple (usual) case: Map data value *f* to color and opacity

Human Tooth CT

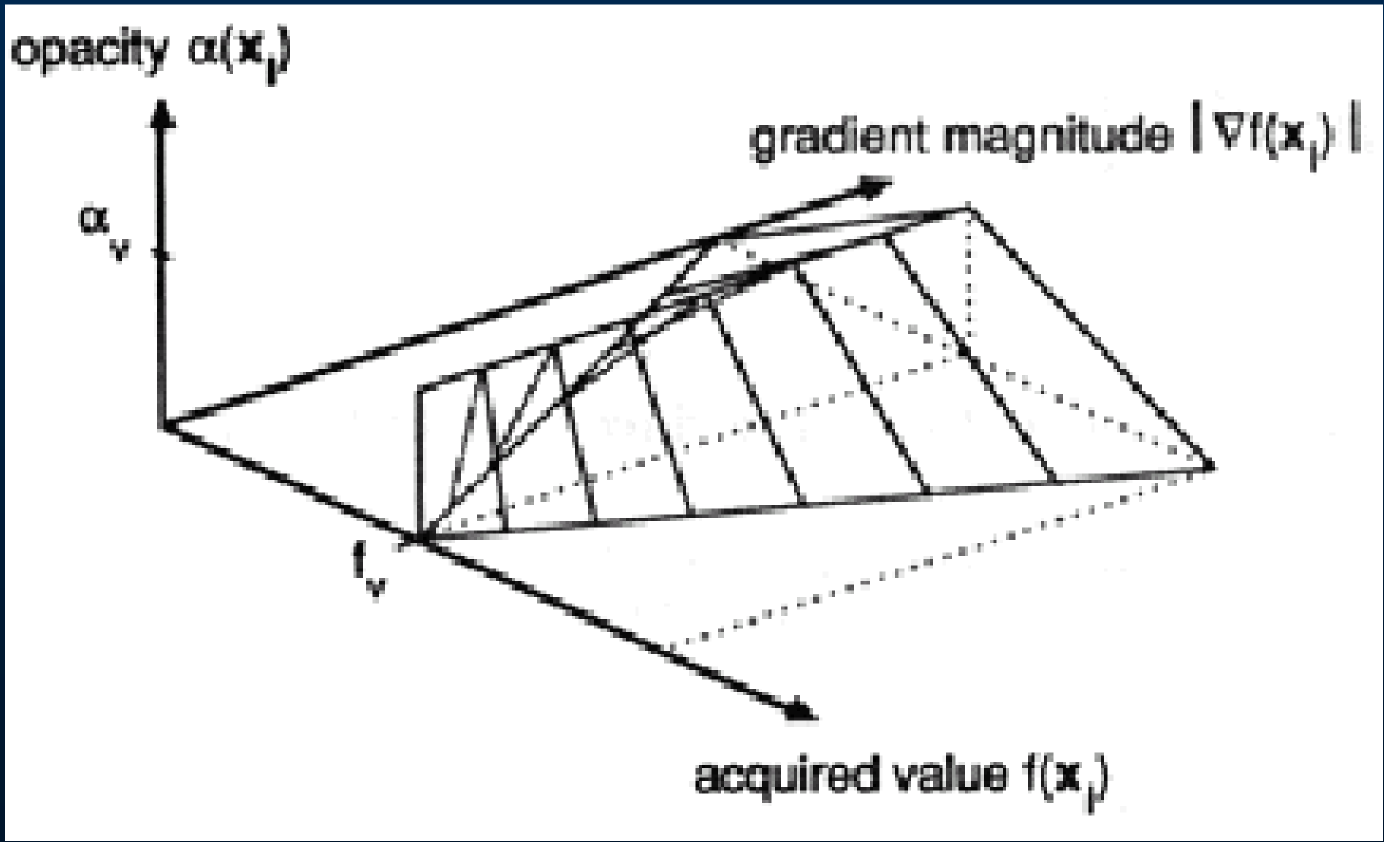


Shading, Compositing…



Classification

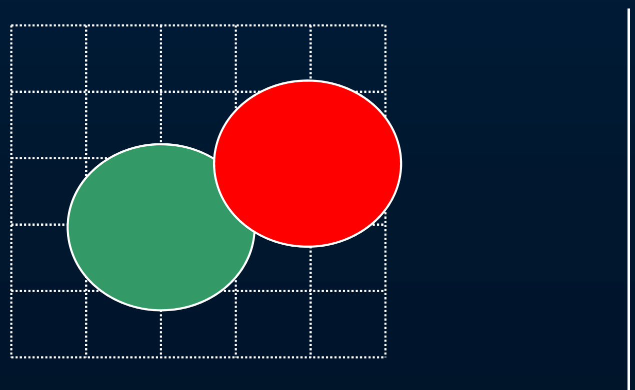
- **Transfer Function**



Volume Splatting

- **Volume = field of 3D interpolation kernels**
  - **One kernel at each grid voxel**
- **Each kernel leaves a 2D *footprint* on screen**
  - **Voxel contribution = footprint ·(C, opacity)**
- **Weighted footprints accumulate into image**

voxel kernels

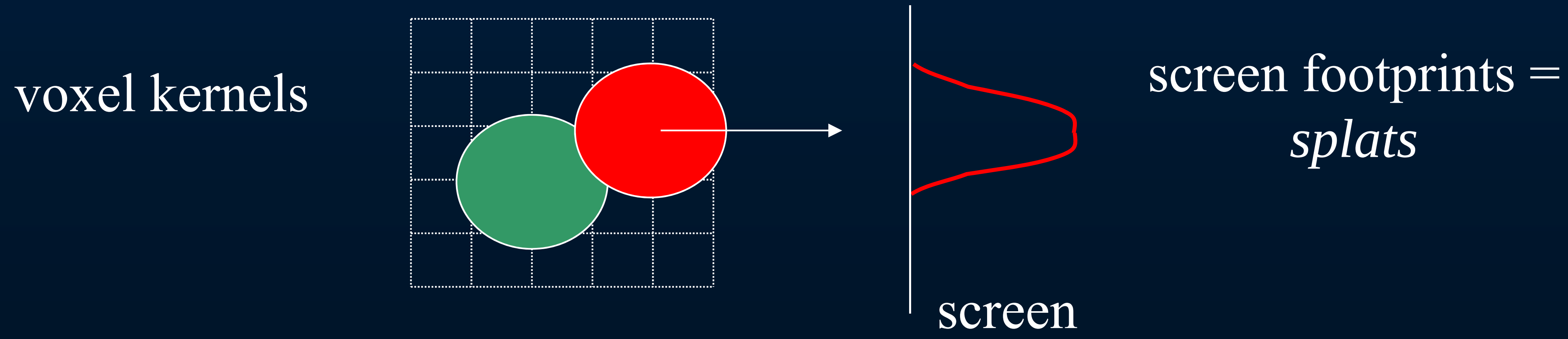


screen footprints = *splats*

screen

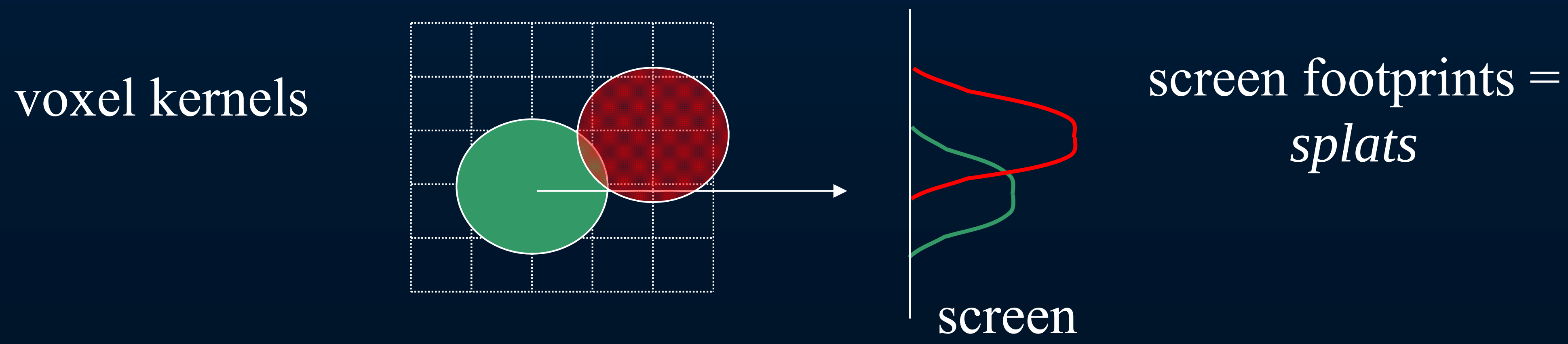
Volume Splatting

- **Volume = field of 3D interpolation kernels**
  - **One kernel at each grid voxel**
- **Each kernel leaves a 2D *footprint* on screen**
  - **Voxel contribution = footprint ·(C, opacity)**
- **Weighted footprints accumulate into image**



Volume Splatting

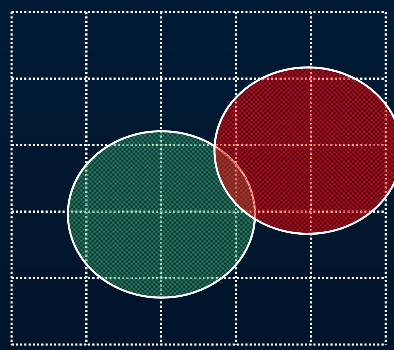
- **Volume = field of 3D interpolation kernels**
  - **One kernel at each grid voxel**
- **Each kernel leaves a 2D *footprint* on screen**
  - **Voxel contribution = footprint ·(C, opacity)**
- **Weighted footprints accumulate into image**



Volume Splatting

- **Volume = field of 3D interpolation kernels**
  - **One kernel at each grid voxel**
- **Each kernel leaves a 2D *footprint* on screen**
  - **Voxel contribution = footprint ·(C, opacity)**
- **Weighted footprints accumulate into image**

voxel kernels



screen

screen footprints = *splats*

Volume Splatting: Highlights

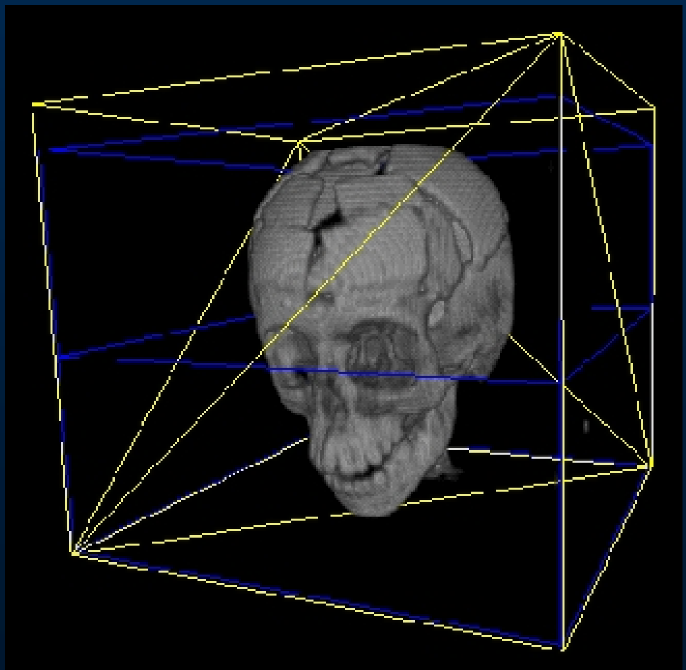
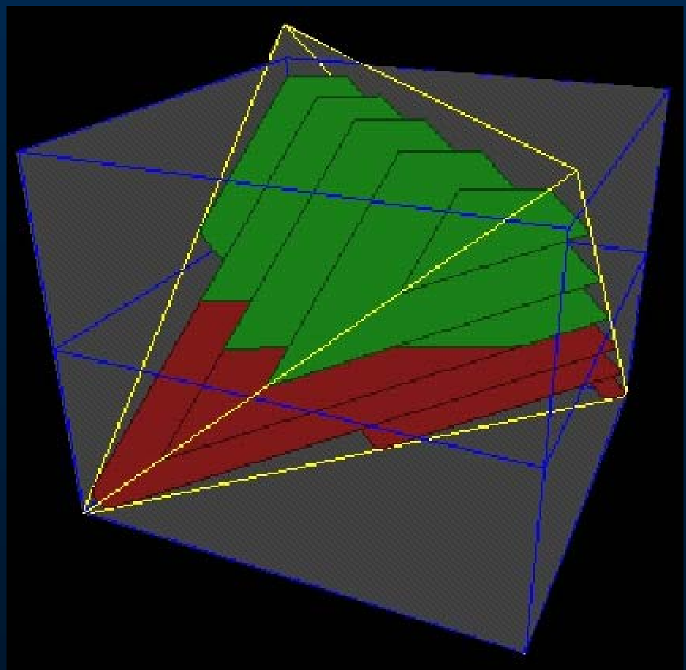
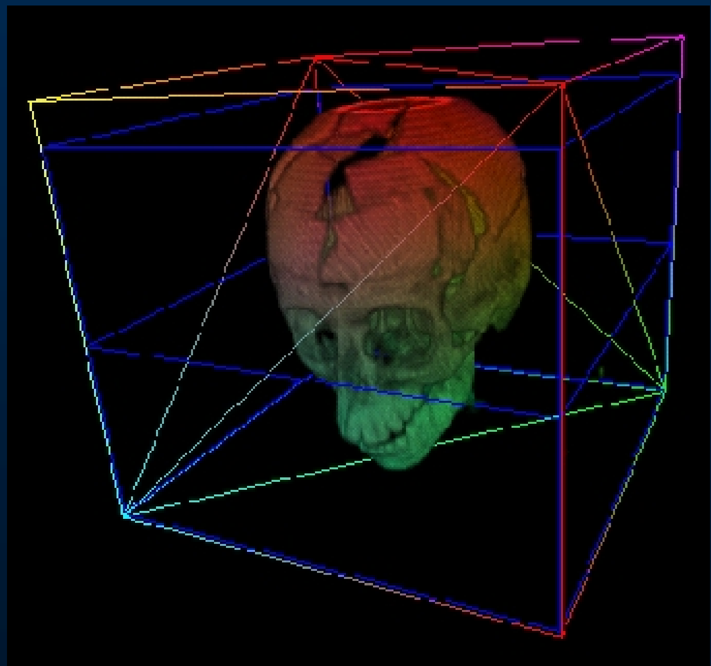
- Footprints can be pre-integrated
  - fast voxel projection
- Advantages over raycasting:
  - Fast: voxel interpolation is in 2D on screen
  - Hardware acceleration
  - More accurate reconstruction (afford better kernels)
  - Only relevant voxels must be projected

Volume Rendering Expenses

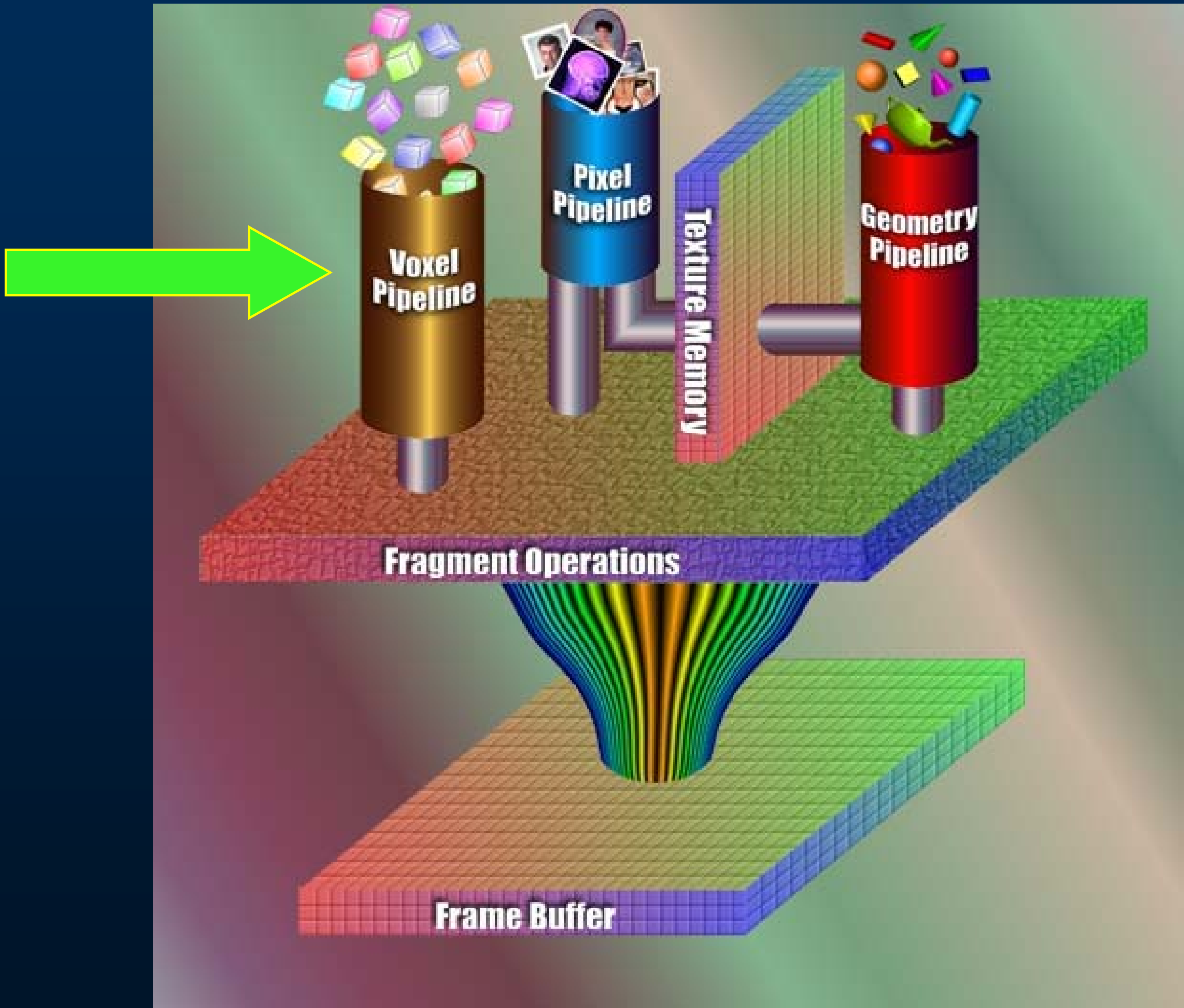
1024^3 16-bit volume @ 30 Hz

- 2GBytes storage
- 60GBytes/second memory bandwidth
  (one access per voxel)
- 900 billion instructions/second
  (30 instructions per voxel)

Volume Rendering Using Conventional Graphics Hardware

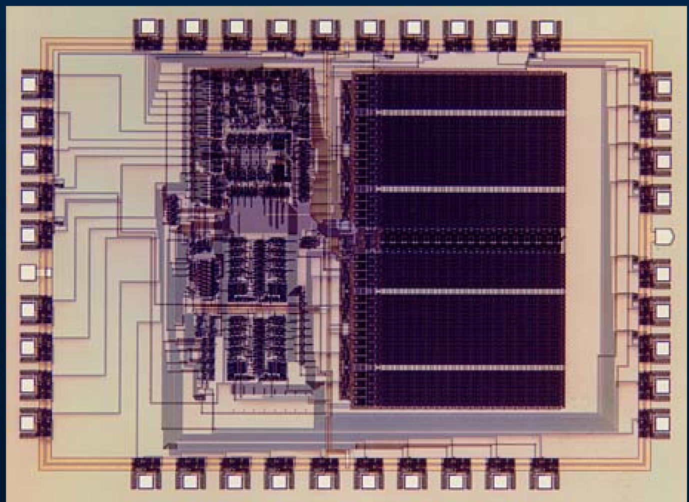
HP *Voxelator* Architecture



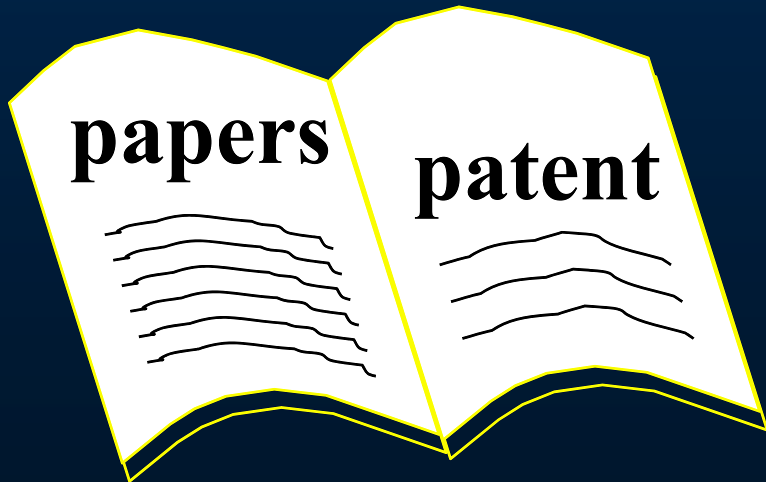
Cube Architecture Design

Cube-1

VolumePro (Cube-4)

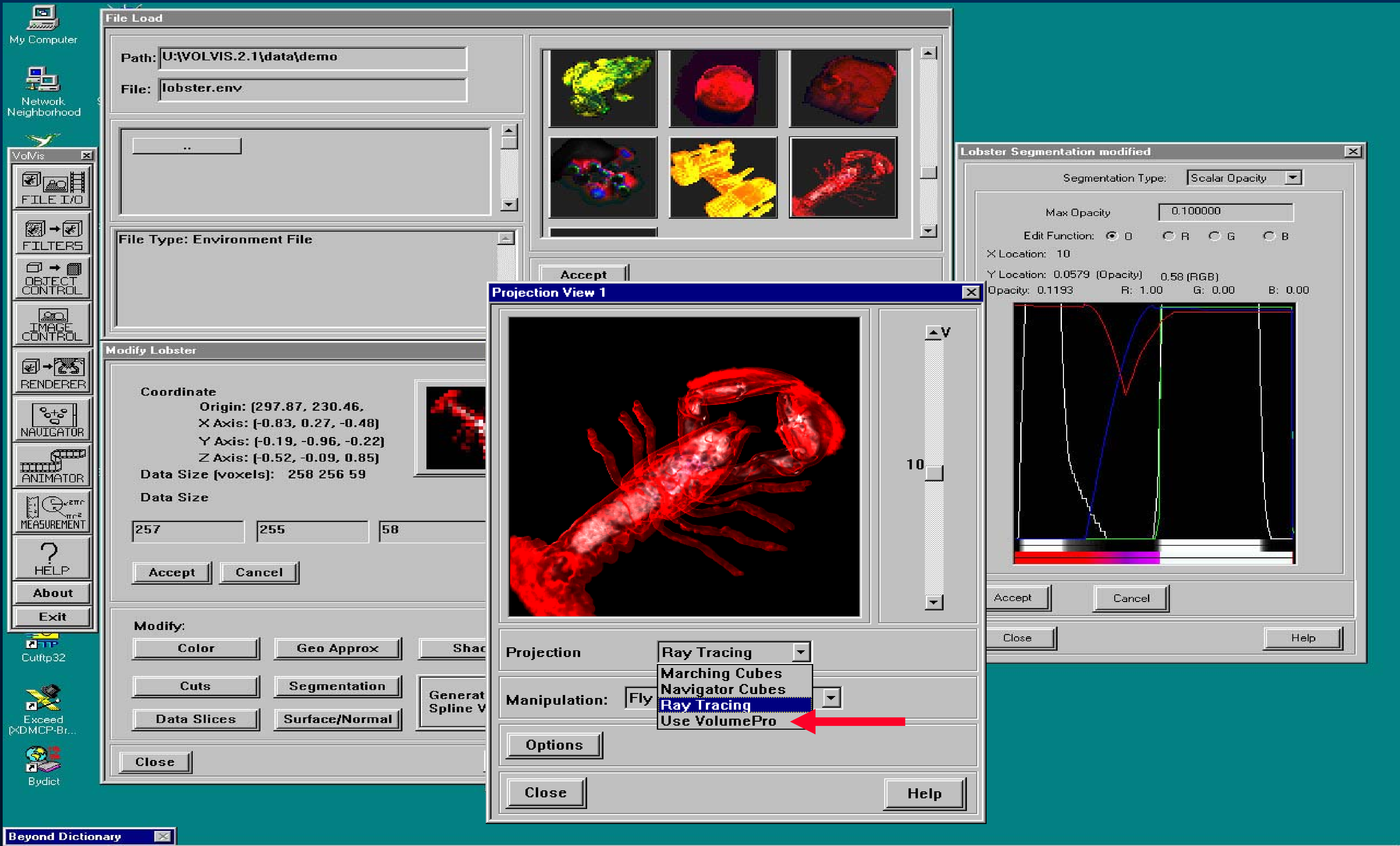


Cube-2

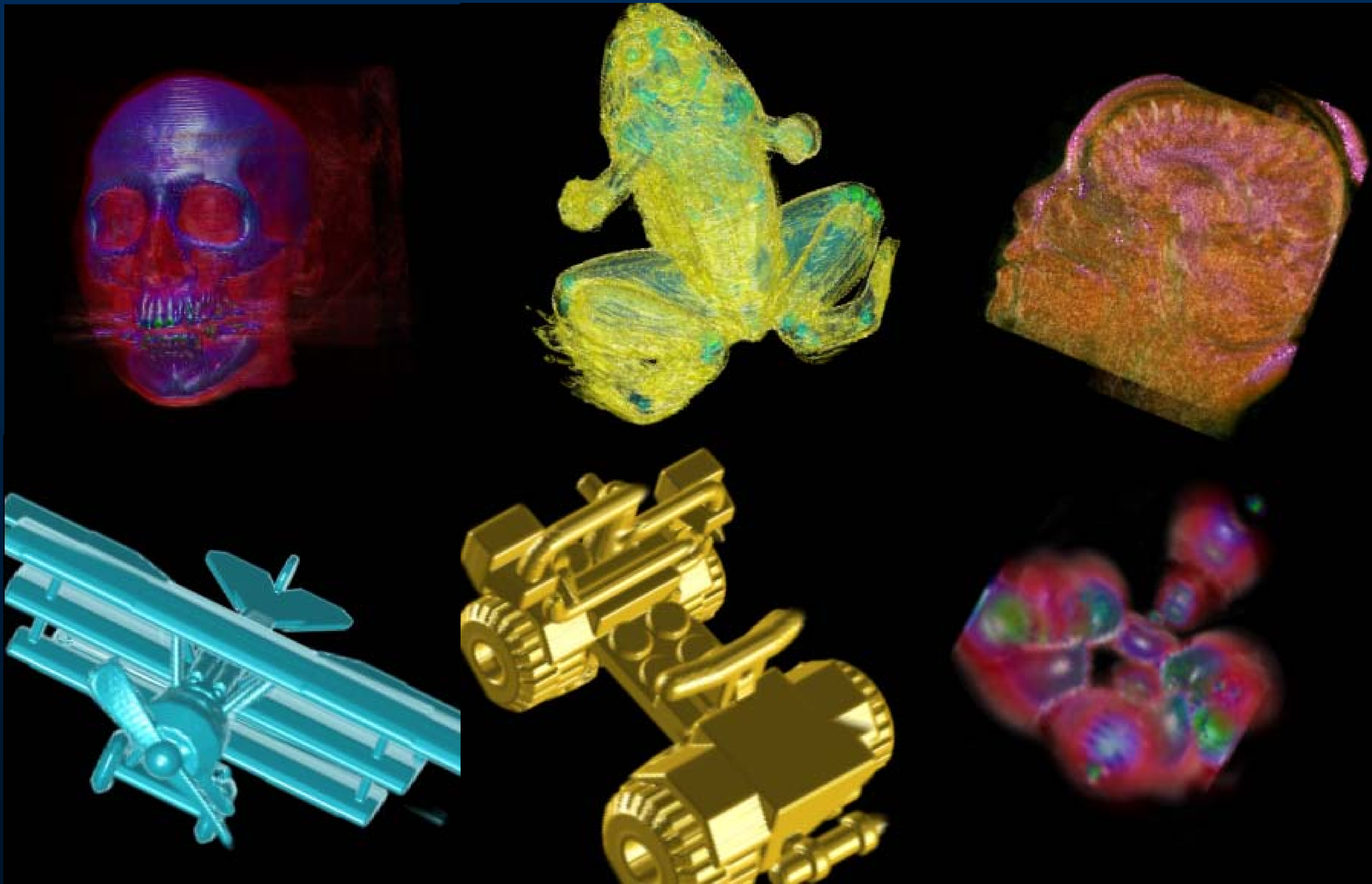


Cube-5

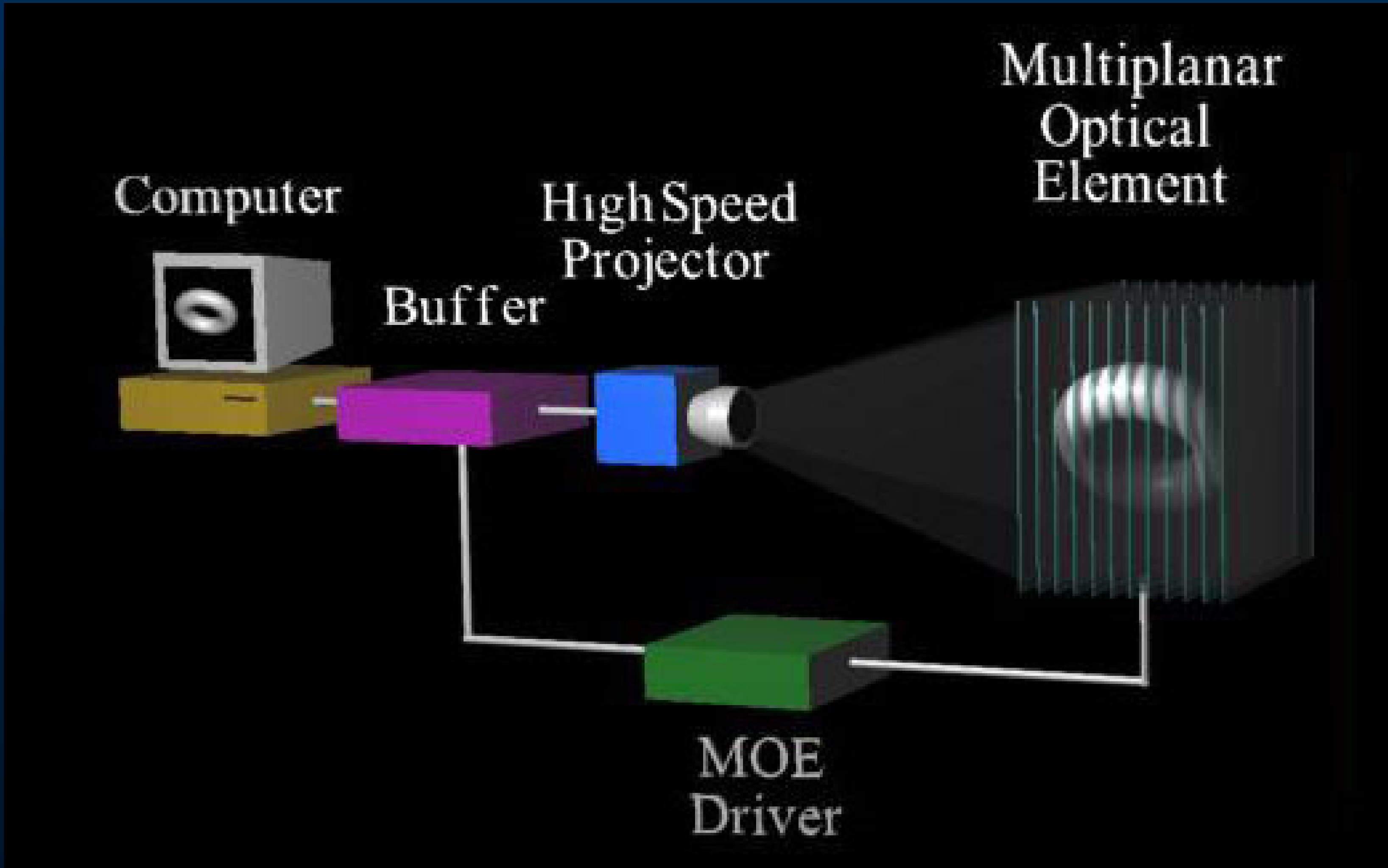
VolumePro / VolVis



VolumePro / VolVis



Volumetric Display



http://www.lightspacetech.com/