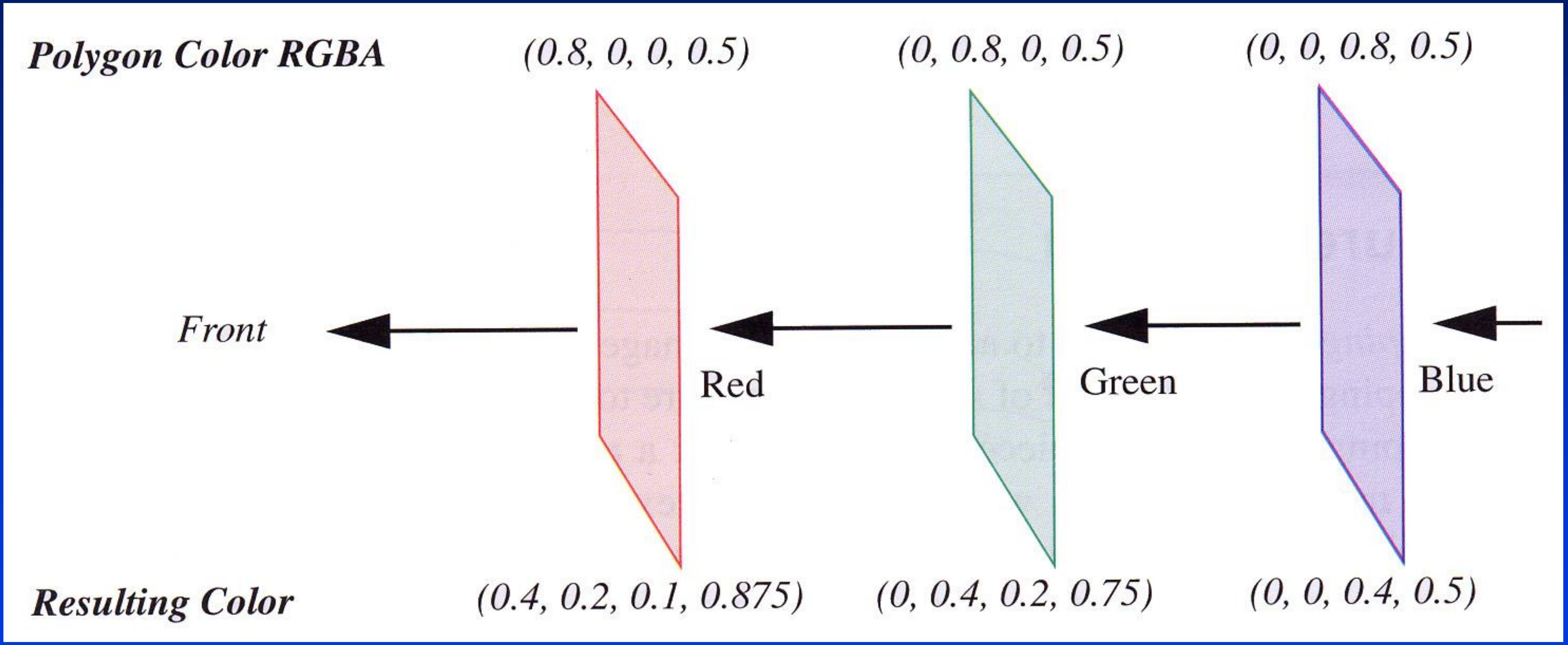
Advanced Volume Rendering Techniques

Transparency and Alpha Values

- Why is transparency useful?
- Peer into and through objects
- Volume visualization
- α = opacity
- $\alpha = 1 \rightarrow$ opaque
- Modern graphics hardware supports **alpha blending**
- Need to **composite** transparent actors
- Does order matter?
- Answer is: Yes

Alpha Compositing



Alpha Composition

$$R = A_s R_s + (1 - A_s) R_b$$
$$G = A_s G_s + (1 - A_s) G_b$$
$$B = A_s B_s + (1 - A_s) B_b$$
$$A = A_s + (1 - A_s) A_b$$

- *s* represents *surface* of actor
- *b* represents what is *behind* actor's surface
- Suppose $A_s = 0$? $A_s = 1$?

Alpha Compositing Example

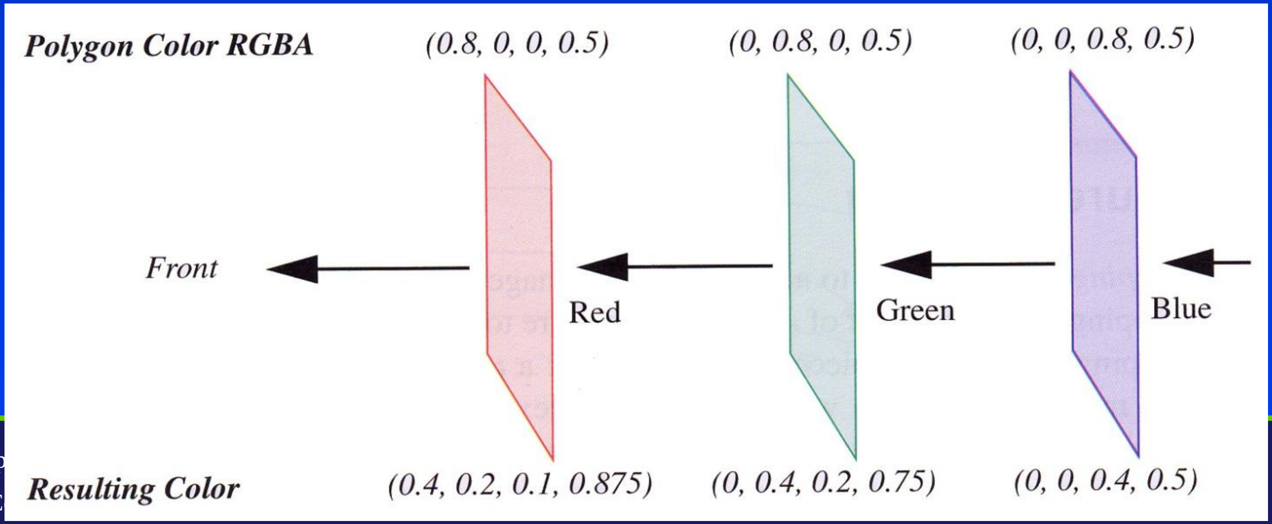
- Use α = 0.5 for all 3 polygons and work through the calculations

$$R = A_s R_s + (1 - A_s) R_b$$

$$G = A_s G_s + (1 - A_s) G_b$$

$$B = A_s B_s + (1 - A_s) B_b$$

$$A = A_s + (1 - A_s) A_b$$

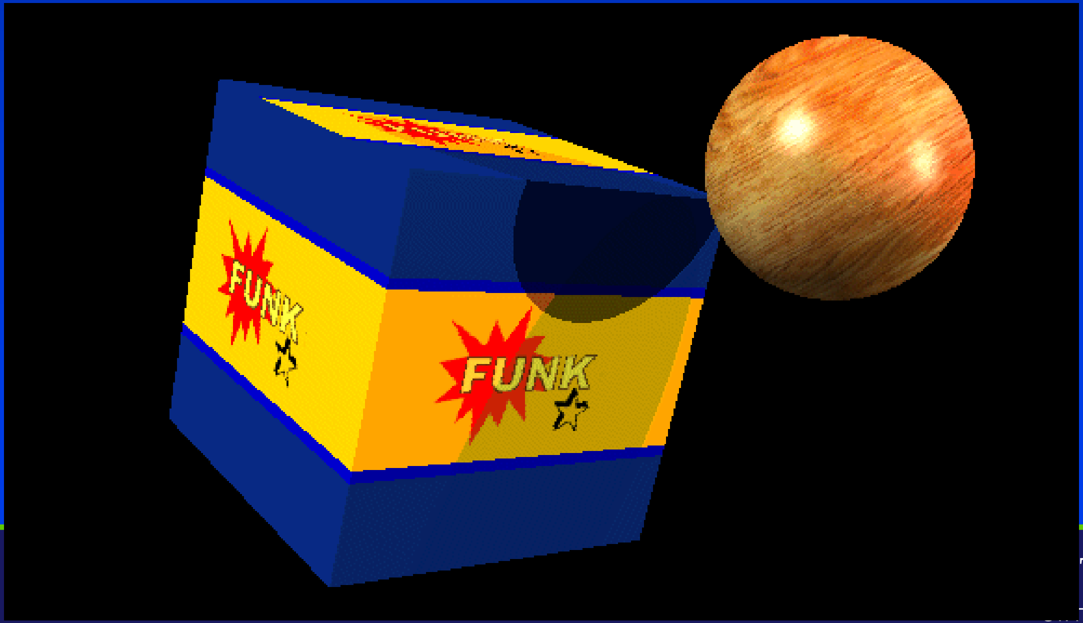


Compositing Order Matters!

- Recall the *z buffer* algorithm, which is used for depth?
- Will not necessarily composite polygons in right order
- Usually must use software to order actors by their increasing distance from camera

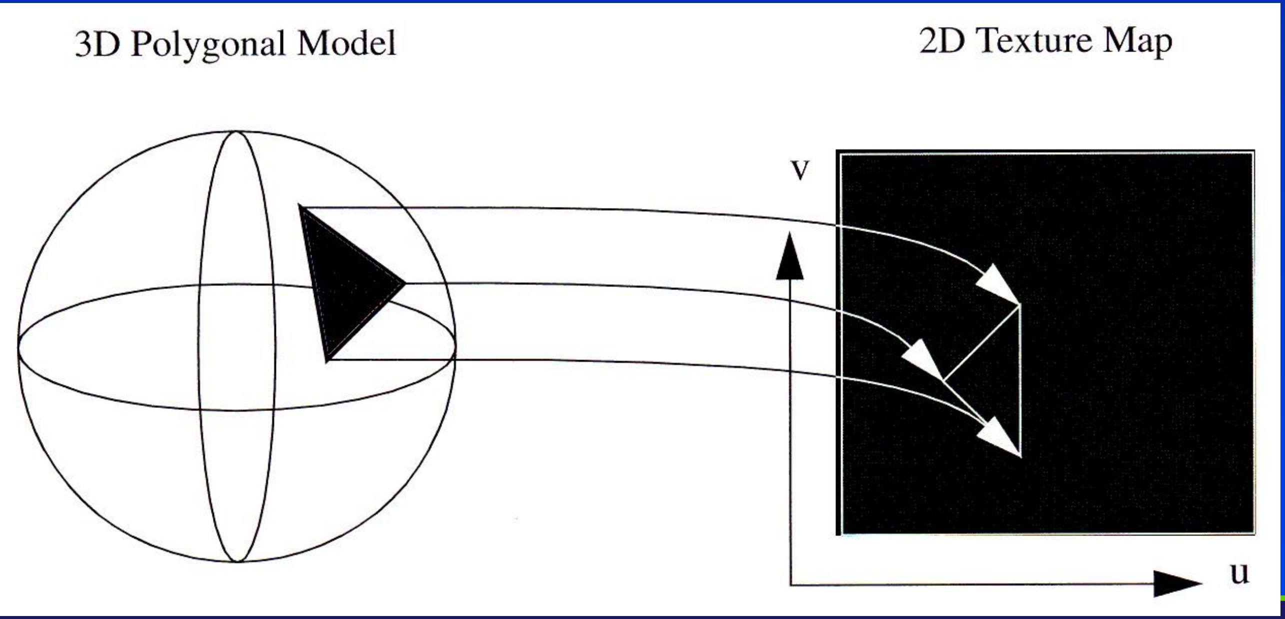
Texture Mapping

- Idea: add detail to image without requiring modeling detail
- Map picture called a **texture map** onto object
- **Texture coordinates** tell you where on object to put picture



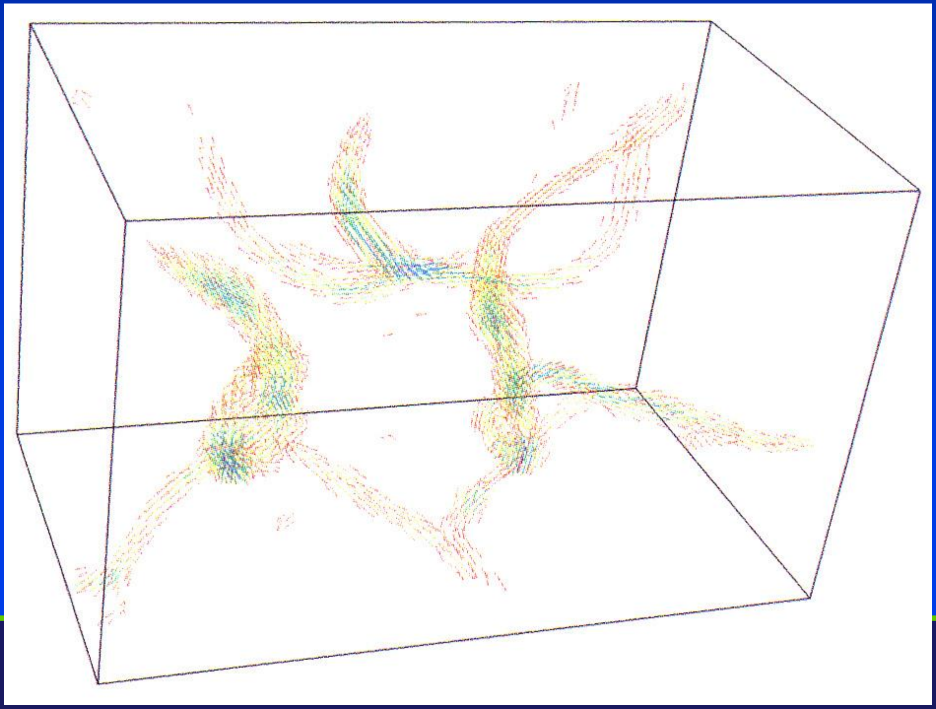
Texture Mapping

- 2D texture mapped onto 3D geometry
- Each 3D vertex assigned 2D texture coordinates, usually written (*u*,*v*)
- Texture is an RGBA image made of **texels**, texture elements



Texture Mapping in Visualization

- Animated texture maps
- Flow visualization
- Colors cycle in a loop to show direction of flow



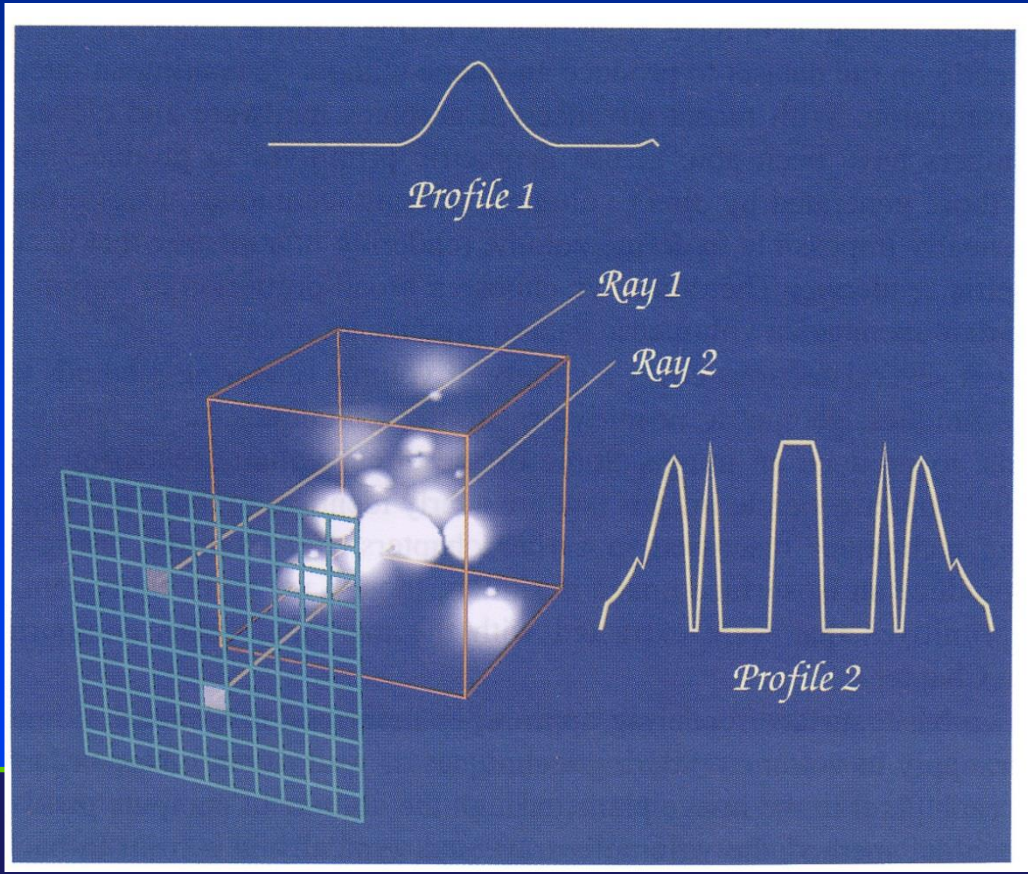
Advanced Volume Rendering

Volume Rendering

- Image-order and object-order volume rendering
- Ray-casting vs. splatting

Ray Casting

- Idea: send *viewing ray* into volume and examine data encountered to compute pixel's color

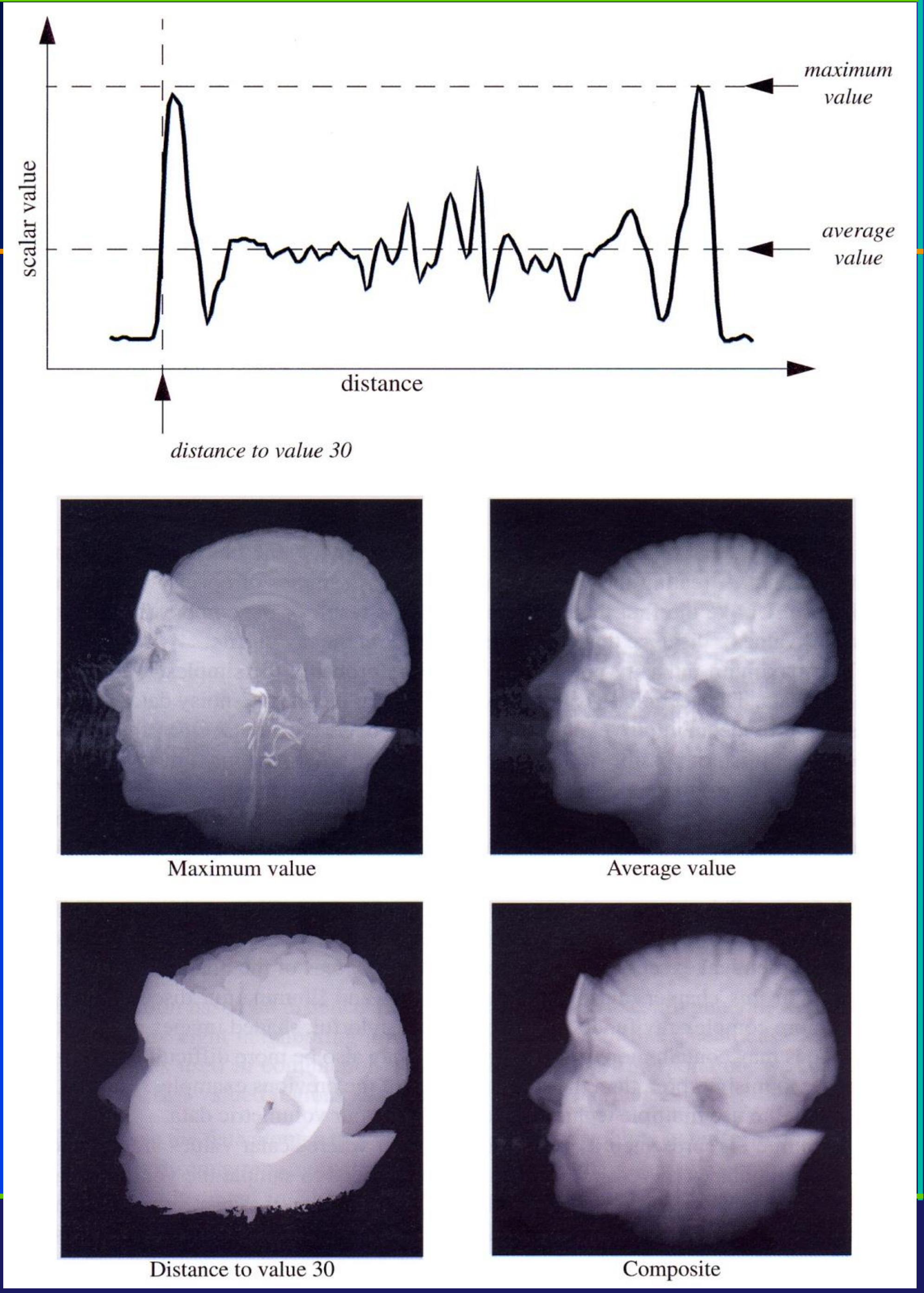


Ray Casting

- Each ray has a different profile we can draw as a 2D curve
- Essentially we will numerically integrate (?) the curve
- Material density, illumination parameters, other attributes affect this integration

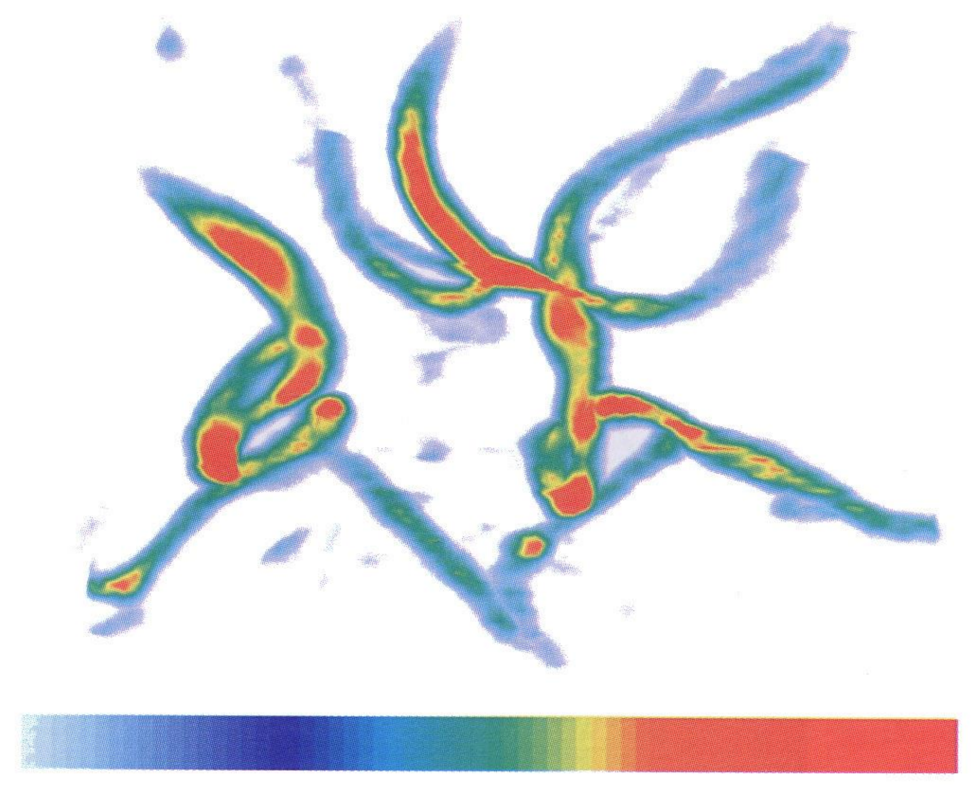
Ray Profile Example

- 8-bit density volume
- Range: 0…255
- x-axis: distance from view plane
- y-axis: density
- Image 3: distance to first voxel with 30+ density value
- Image 4: alpha compositing



Maximum Intensity Projection (MIP)

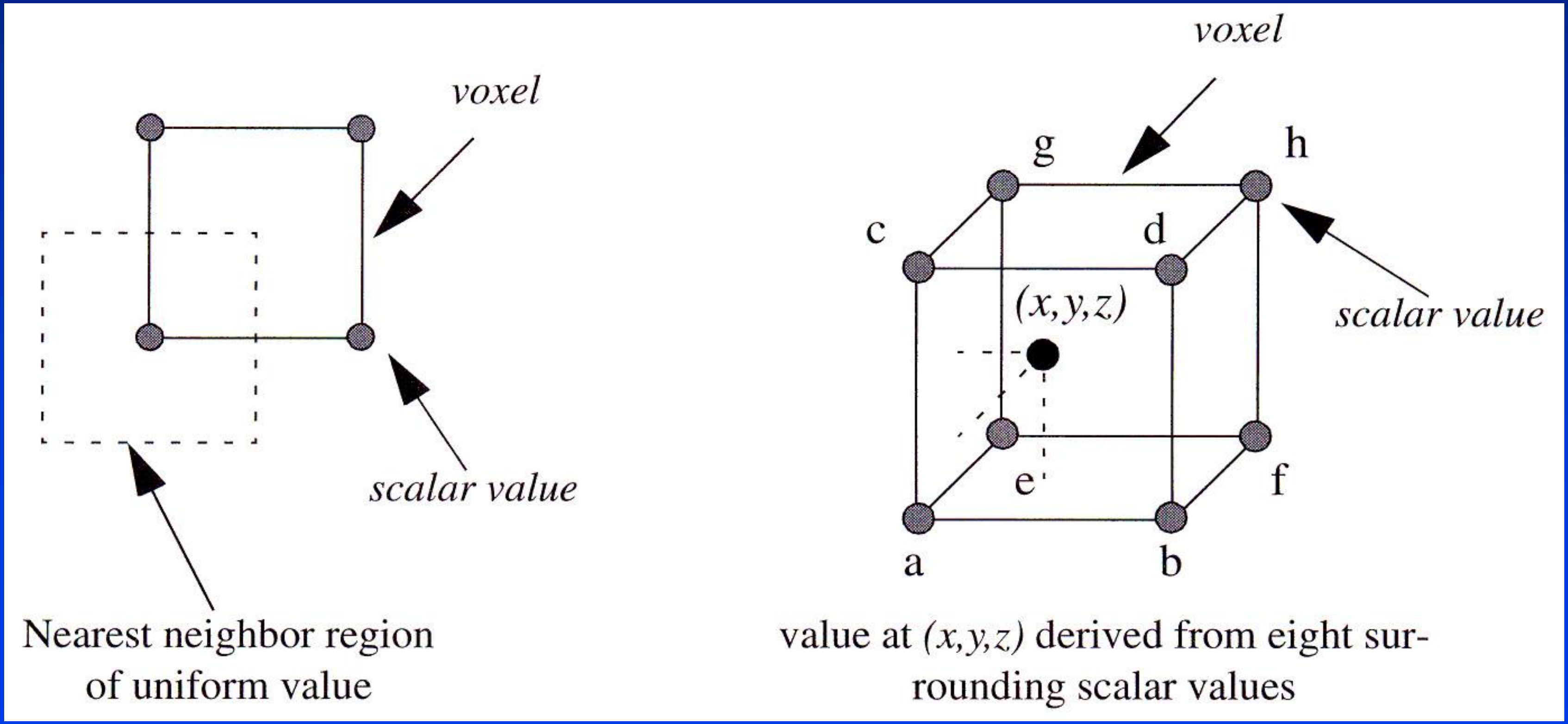
- MIP simple yet effective technique
- Depth perception lost, though
- Can do colored MIP also
- Which blood vessel is in front of the others?
- No compositing, so colors don't blend together



Ray Traversal

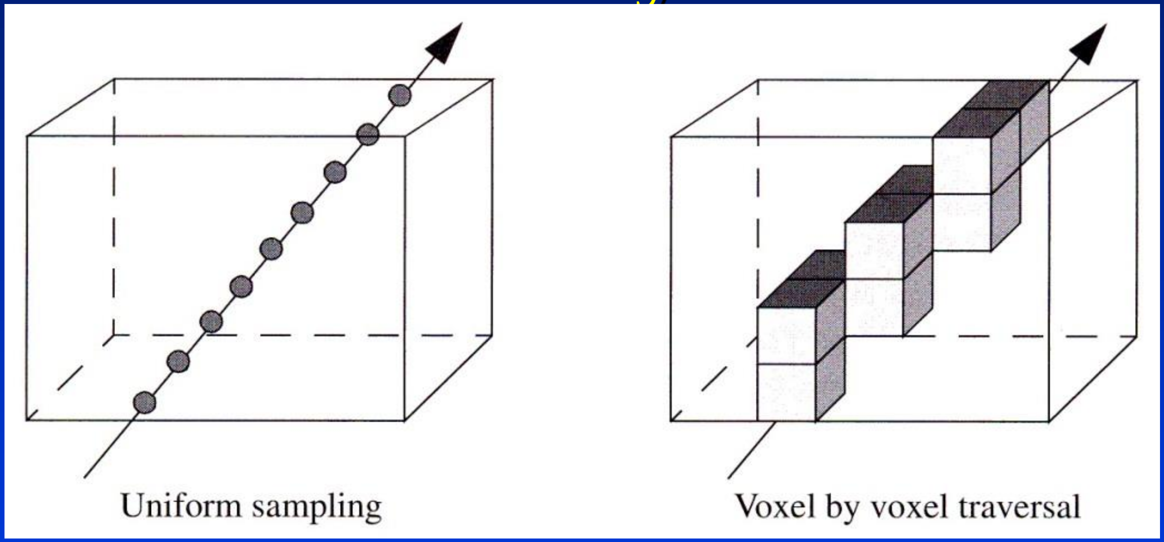
- We take small steps along the ray
- Don’t always land on a voxel
- Need to estimate density somehow (?)
- Interpolation!
- **Nearest neighbor interpolation**: just find closest voxel and use its density
- **Trilinear interpolation**: take some weighted sum of 8 nearest voxels’ densities

Interpolation Techniques



Ray Traversal

- Usually we traverse the ray at uniform intervals



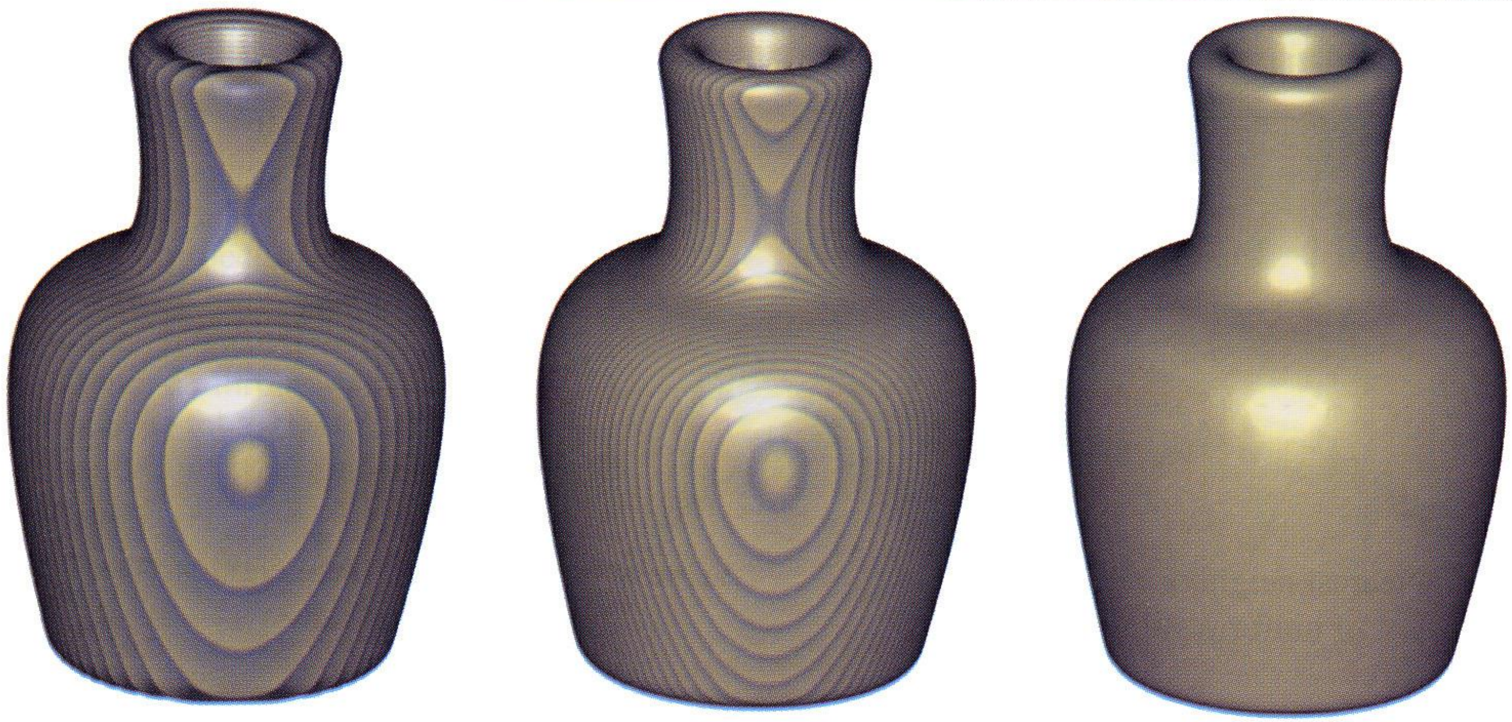
- Parametric form: $(x, y, z) = (x_0, y_0, z_0) + (a, b, c)\,t$
- (x_0, y_0, z_0) is the origin of the ray
- (a, b, c) is the normalized ray direction vector

Ray Traversal Pseudo-code

```
t = t1;
v = undefined;
while ( t < t2 )
  {
  x = x0 + a * t;
  y = y0 + b * t;
  z = z0 + c * t;
  v = EvaluateRayFunction( v, t );
  t = t + delta_t;
  }
```

- t1 and t2 are distances where ray enters and leaves volume, respectively

Step Size Affects Image Quality



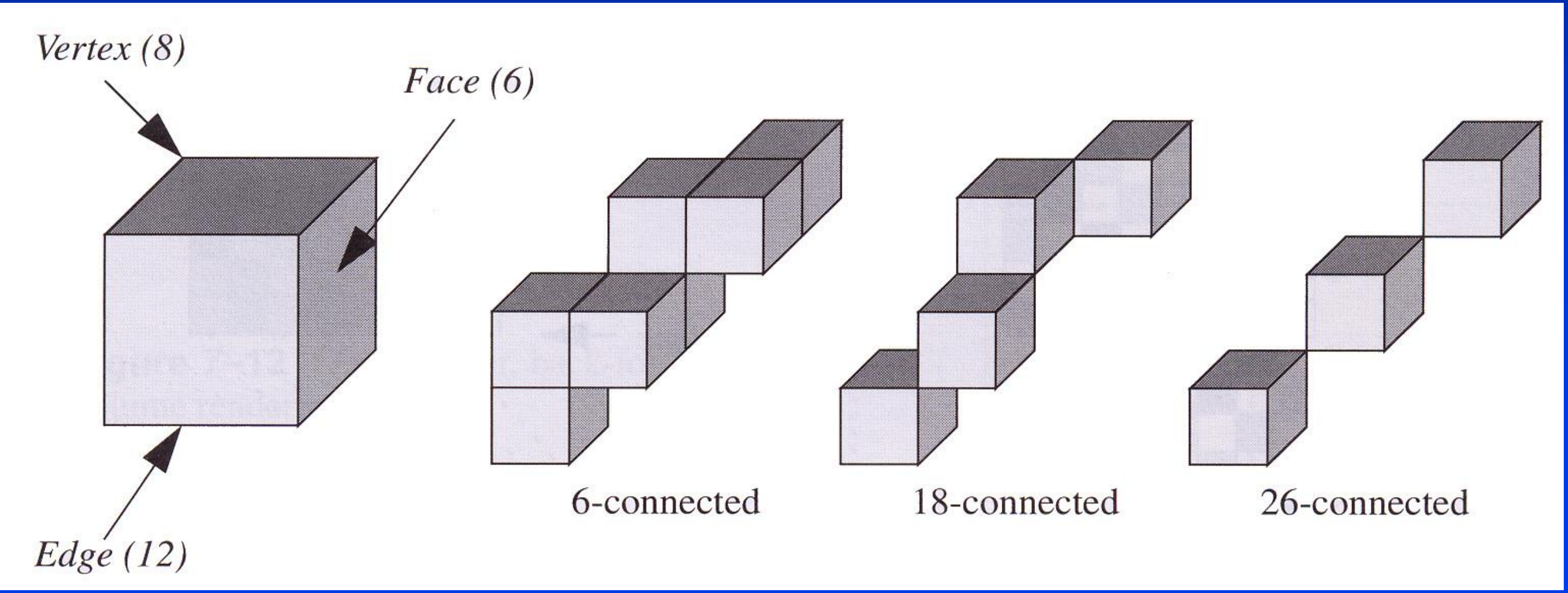
Step size = 2.0 Step size = 1.0 Step size = 0.1

Step Size

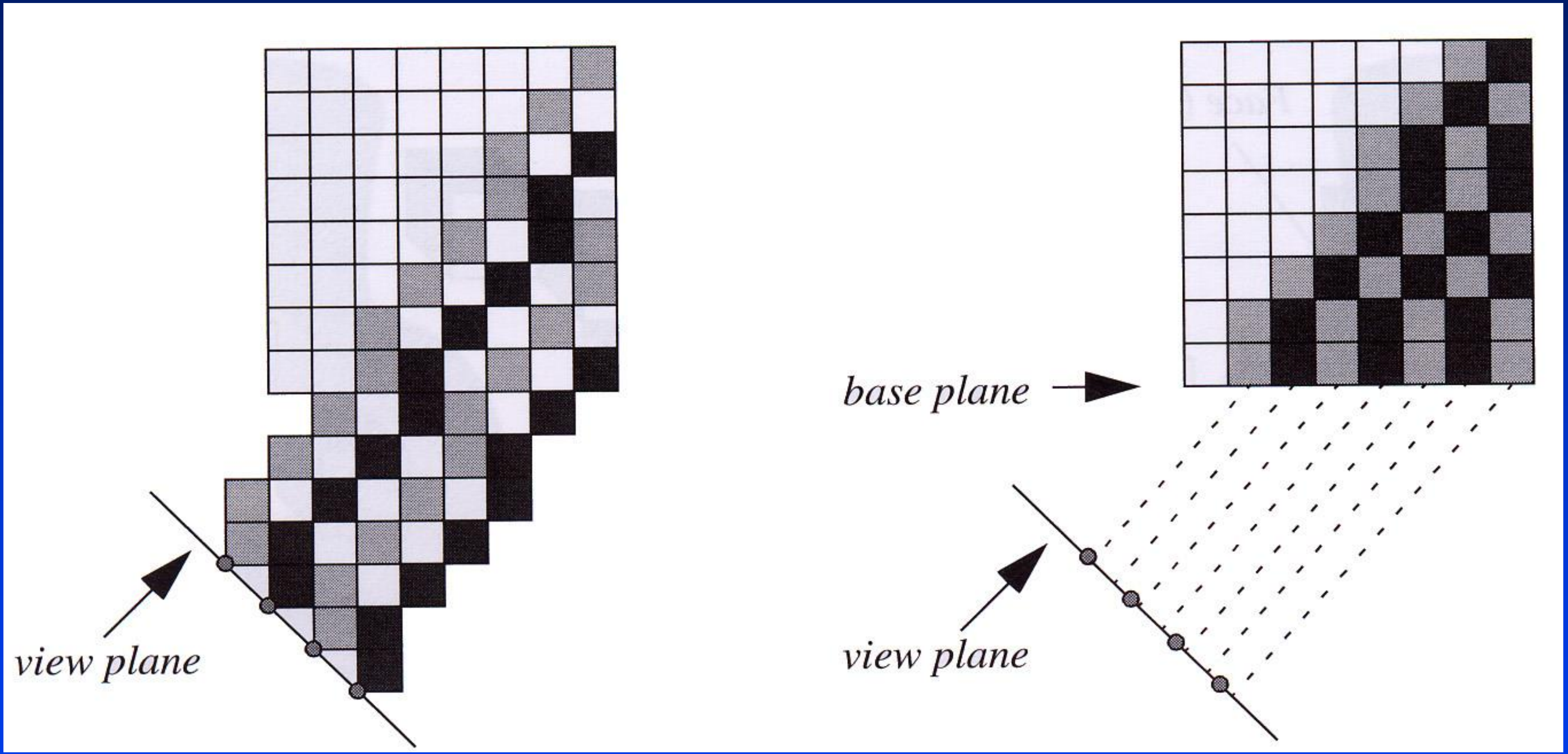
- Small step size = higher quality, slow speed
- Large step size = converse
- Large step size causes the banding effect

Voxel-based Ray Traversal

- Jump from one voxel to another instead of along a continuous ray
- Related to concept of **connectedness**

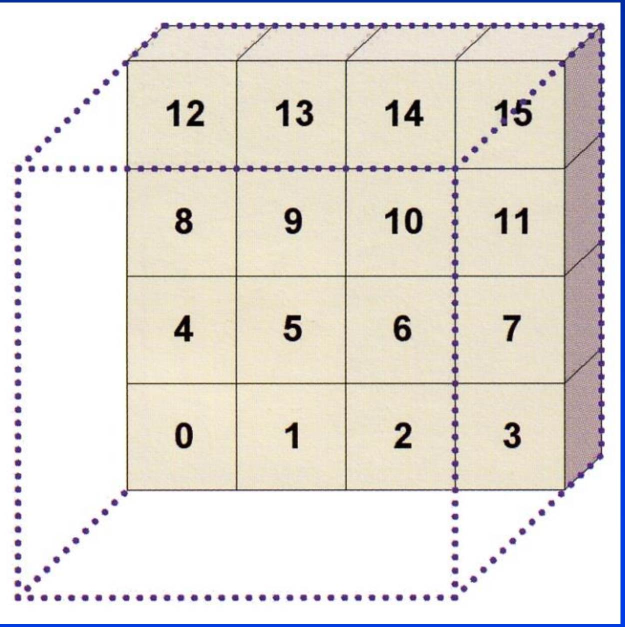


Voxel-based Ray Traversal



Object-Order Volume Rendering

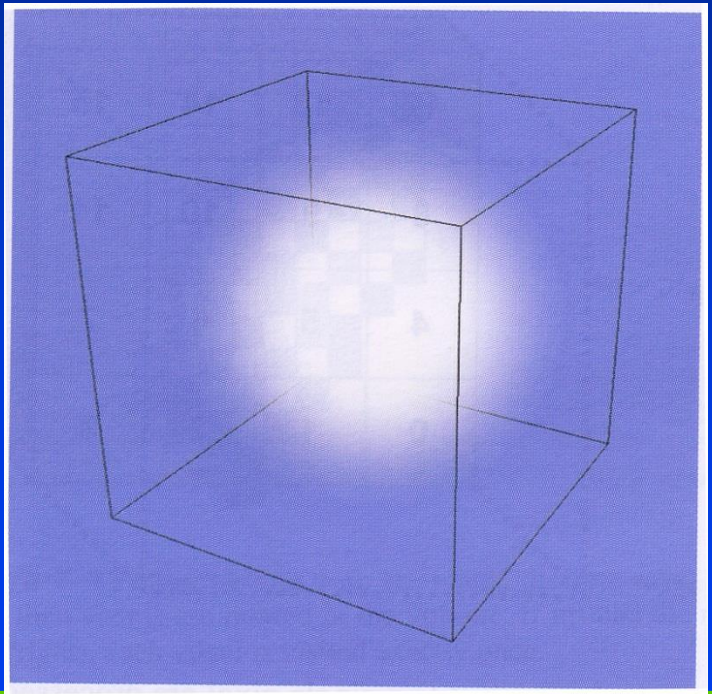
- Back-to-front or front-to-back processing of voxels
- Requires a triply nested loop
- **for** z = …. {
  for y = …. {
  for x = …. {
  ….
  }
  }
  }



- Select plane most parallel to image plane

Splatting

- Fuzzy sphere (called the kernel) placed around each voxel
- Kernel projected onto viewing plane, producing a **footprint**
- Repeat for all voxels
- Kernel size affects image quality
- Footprint *discretized* to a resolution appropriate for image resolution



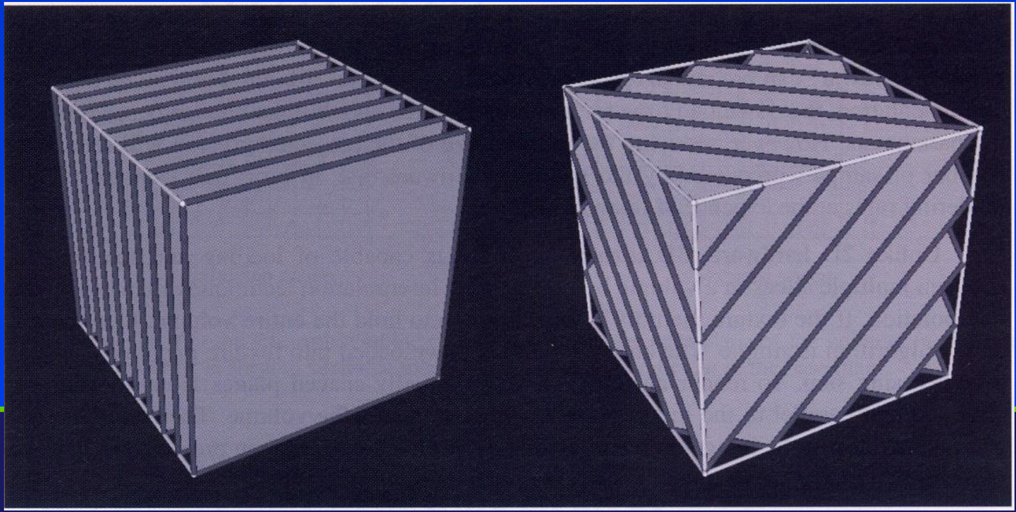
Implementing Splatting

- Software-only vs. hardware-assisted
- Footprint table – slices generic kernel into image-aligned slabs

Texture Mapping-based Volume Rendering

- In 2D: project and composite *axis-aligned* slices onto image plane
- In 3D: cut volume into slices that are *parallel* to the image plane ("image-aligned slices")
- Use interpolation and compositing in both cases

using 2D texture mapping hardware

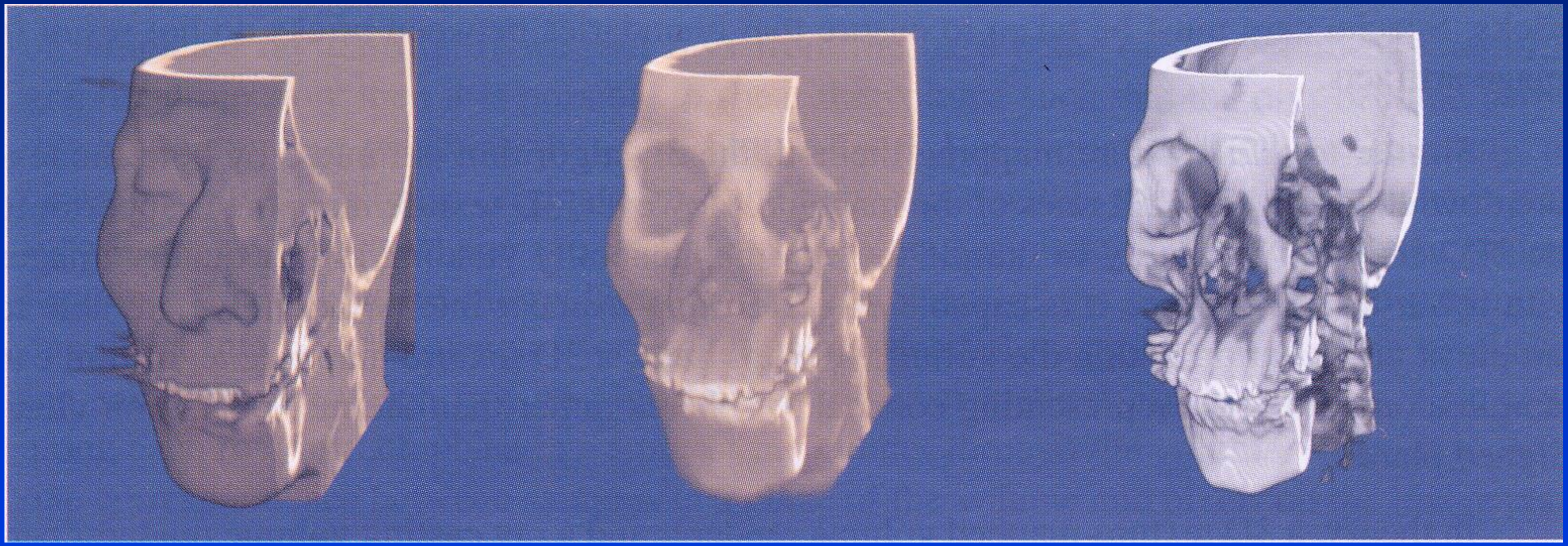


using 3D texture mapping hardware

Department of Computer Science
Center for Visual Computing

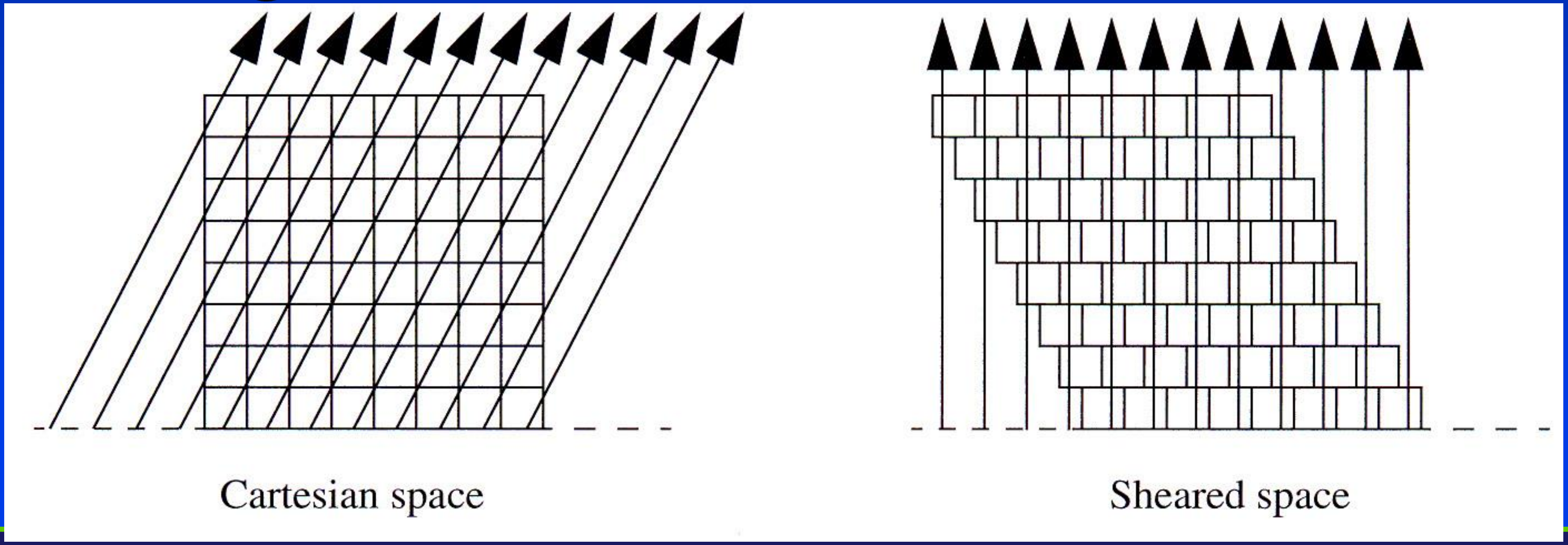
STONY BROOK
STATE UNIVERSITY OF NEW YORK

2D Texture-Mapped Volume Rendering Example



Shear-Warp Volume Rendering

- Hybrid technique – aspects of object-order and image-order rendering
- Idea: convert a rotation of the camera into a *shearing* of the volume

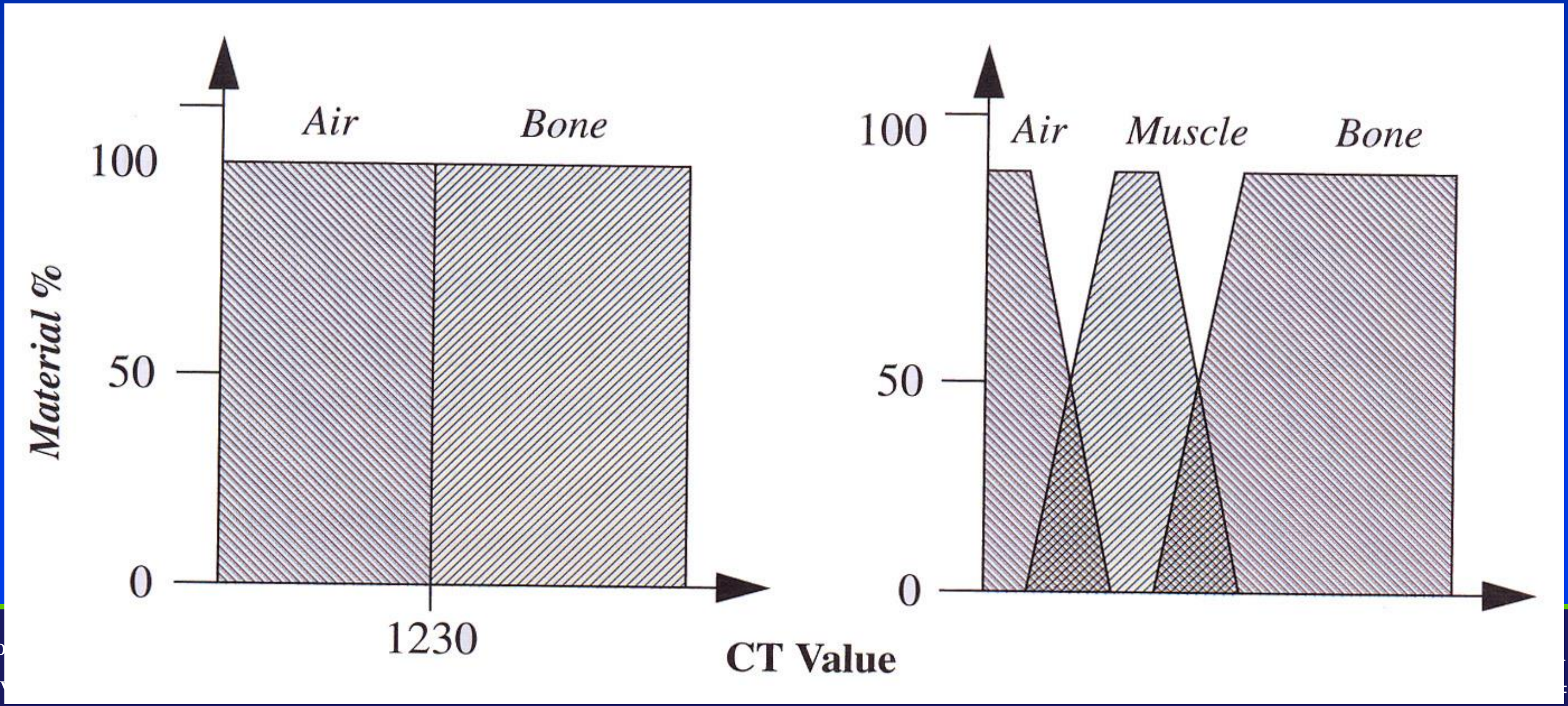


Shear-Warp Volume Rendering

- Need to use **bilinear interpolation** to resample the slices
- Front-to-back ray traversal
- Essentially a very efficient form of ray-casting
- Downside? Extra interpolations introduce error and hurt image quality
- Requires 3 copies of the volumes so we can shear volume along direction most parallel to image plane
- Shear in xy-plane, xz-plane or yz-plane
- Need to be able to process raster in any order

Volume Classification

- Assignment of density ranges to categories
- Represented by transfer functions
- "Material percentage" transfer functions:



Volume Classification

- Usually we classify a volume using red, green, blue and opacity transfer functions
- Two possibilities to apply classification during ray traversal:
  1. Interpolate voxel densities and then compute color
  2. Assign colors to voxels and then interpolate colors
- Option 1 tends to make nicer looking images

Volume Classification

- We can also compute the **gradient** of the density field and use that to modulate the color
- Gradient is a vector that tells you how the material is changing at a position
- Vector of first partial derivatives in x, y and z:

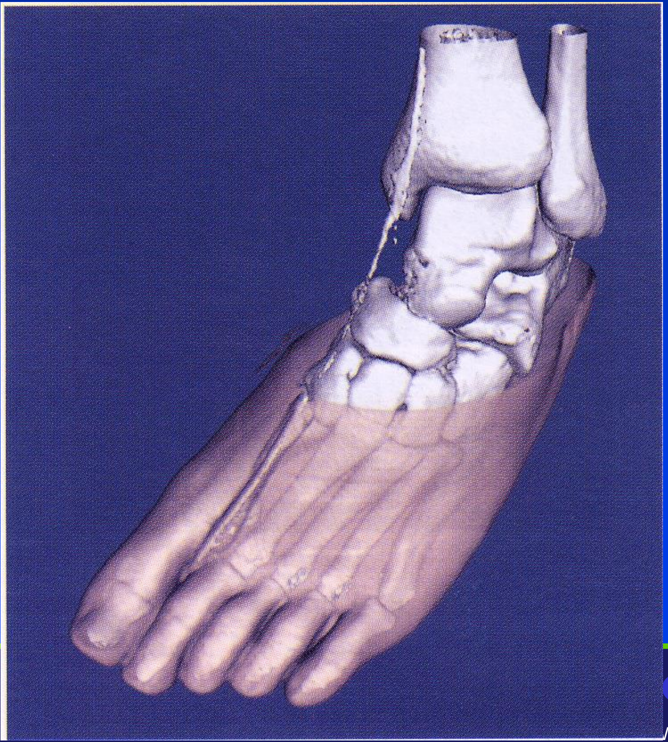
$$g_x = \frac{f(x+\Delta x, y, z) - f(x-\Delta x, y, z)}{2\Delta x}$$

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

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

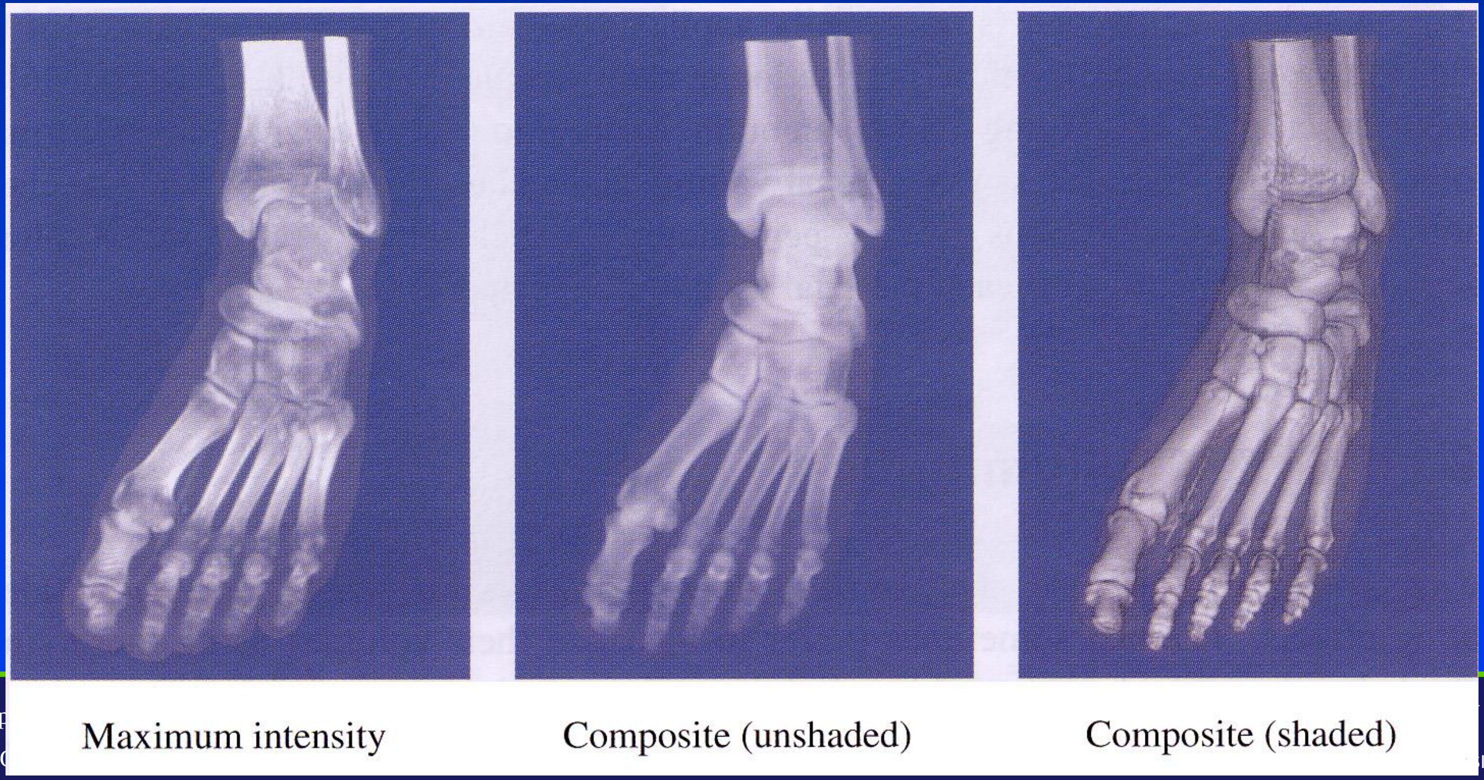
Volume Classification

- If the vector **g** is long, that means the material is changing quickly
- Example: boundary between bone and flesh
- Implies presence of a surface
- Modulate color based on gradient magnitude to ignore regions of **homogeneous** material distributions
- Small magnitude = little or no change of material



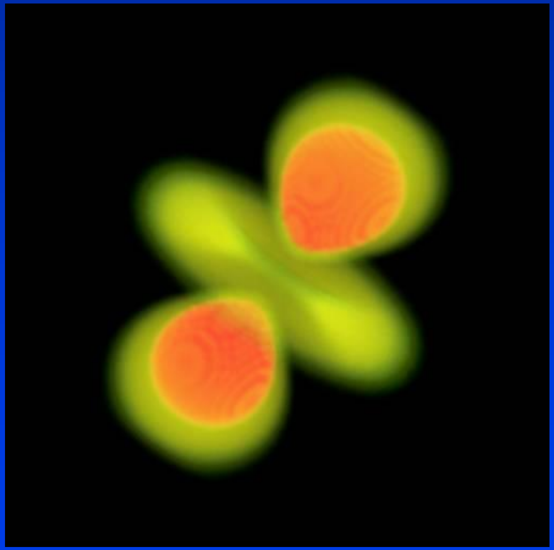
Uses of the Gradient Vector

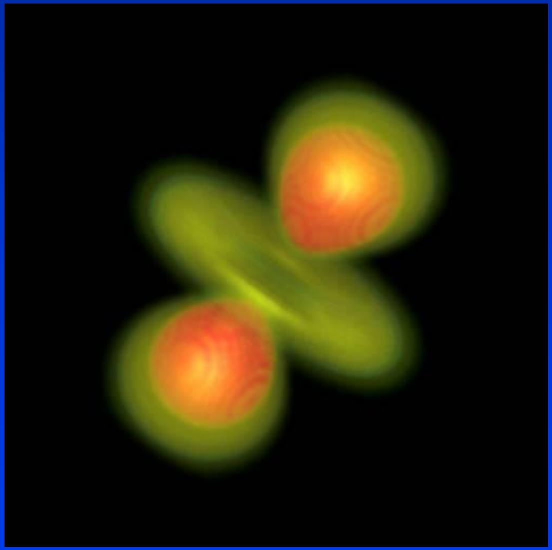
- We can treat the gradient as a normal vector and evaluate the lighting equation to shade and illuminate volumes

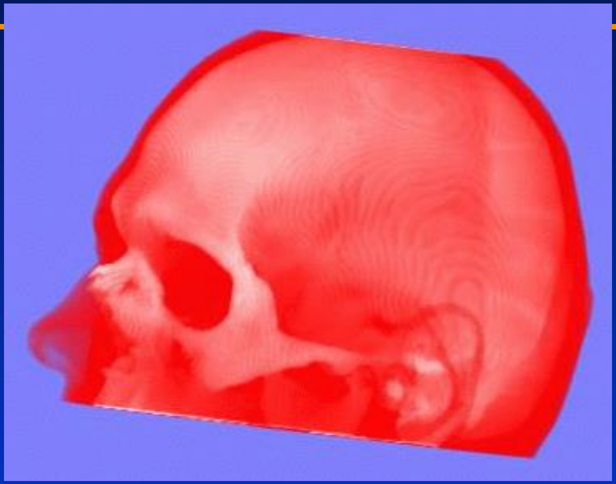


Volumetric Shading

- Can reveal surfaces inside the data
- Compare:







Volumetric Shading: How?

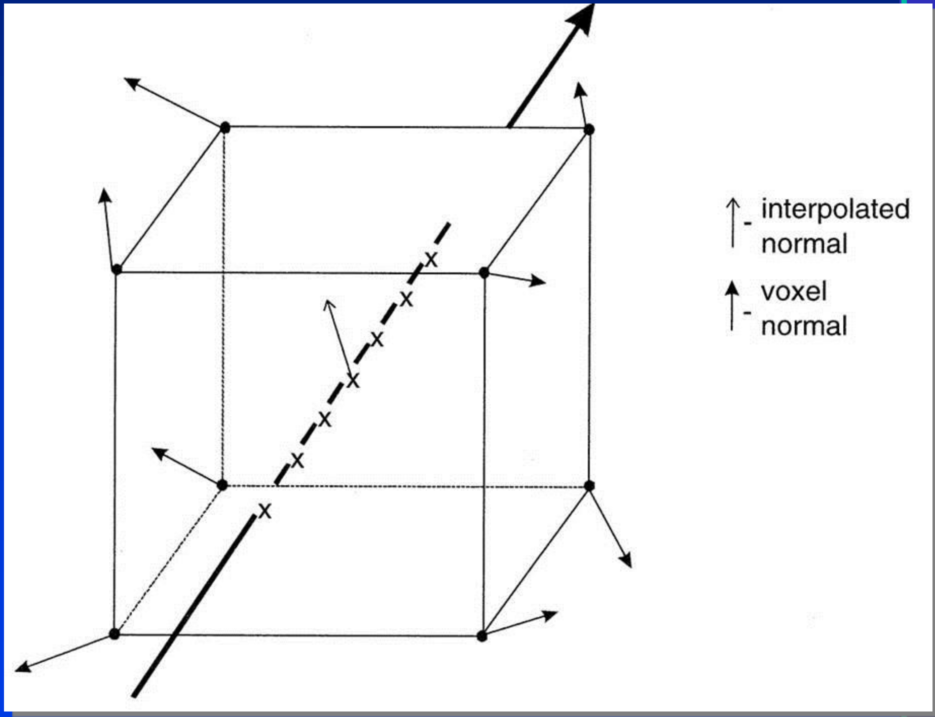
- Gradient allows us assigns a direction vector to each voxel
- This (normalized) vector is used just like the normal vector in surface graphics
- It will modulate the color we assign to samples and thereby allow us to create 3D effects
- Look at the skull on the right

Volumetric Shading

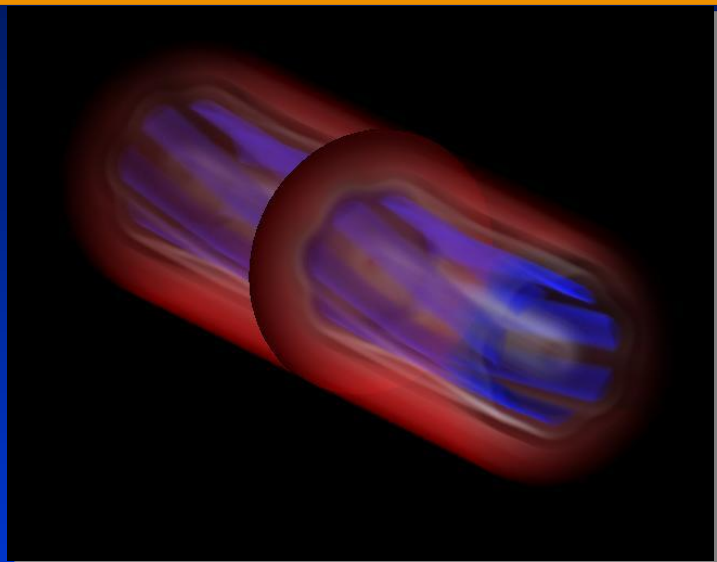
- But how do we incorporate color, opacity, and shading information?
- First we interpolate the density at a given sample position
- Interpolate gradient at same position
- Then assign color and opacity to each sample, and shade using interpolated gradient
- When we shoot the rays through the volume, we have to composite all these samples together

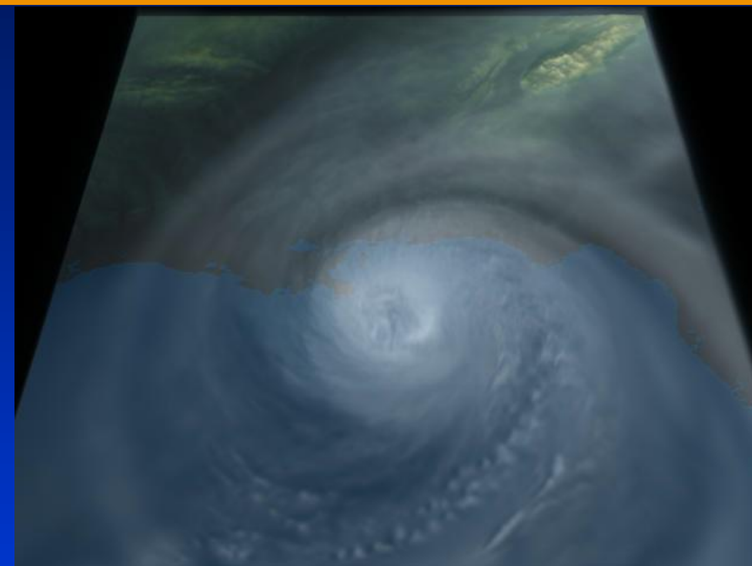
Gradient Interpolation

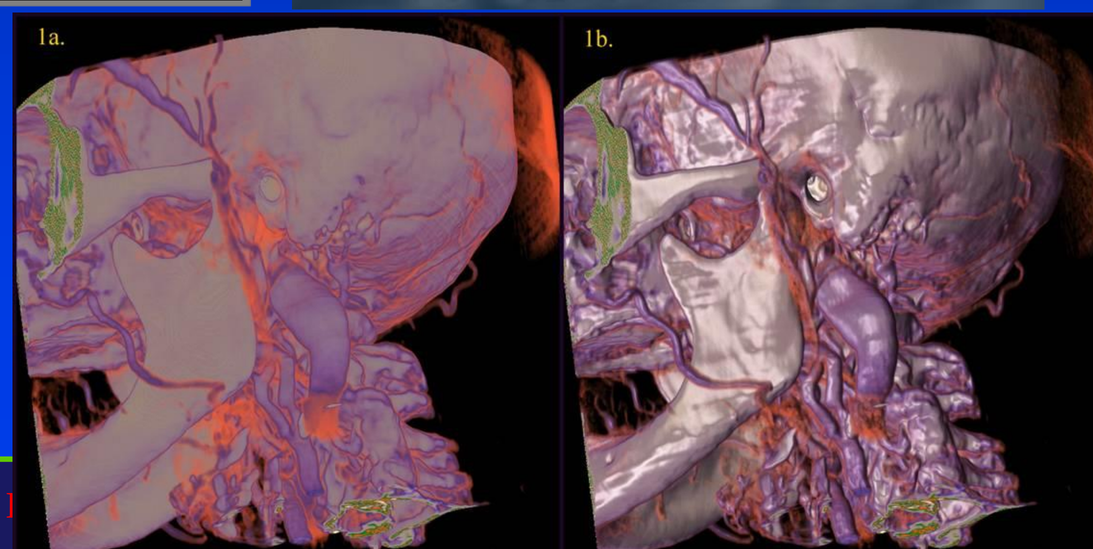
- At start of processing, compute gradient at each voxel
- During ray traversal, estimate gradient with trilinear interpolation
- Like densities (and unlike colors), gradients are **intrinsic** attributes of models



Volumetric Shading Examples







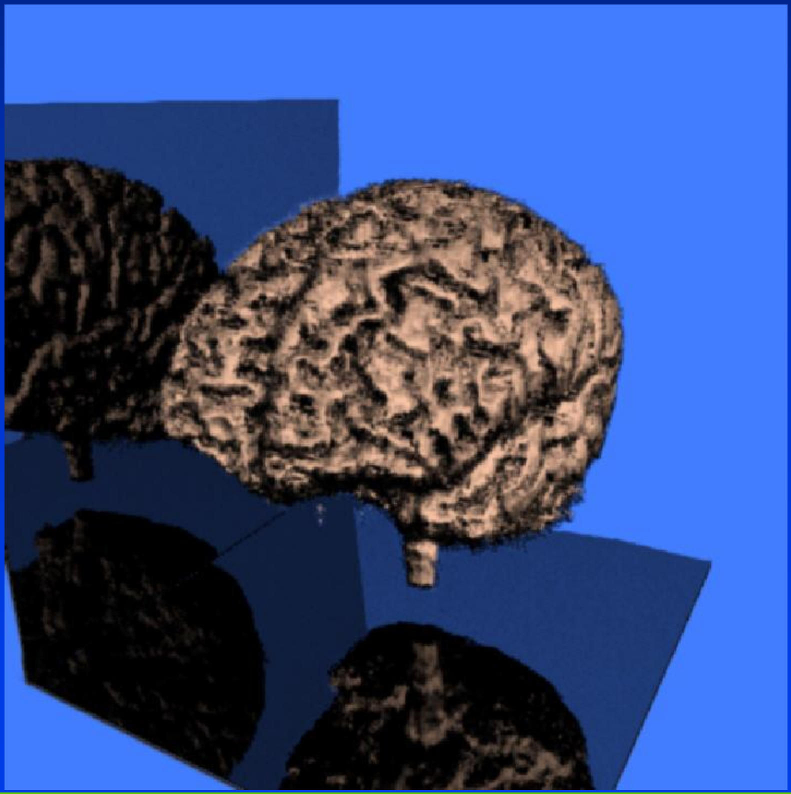
Gradient Modulation

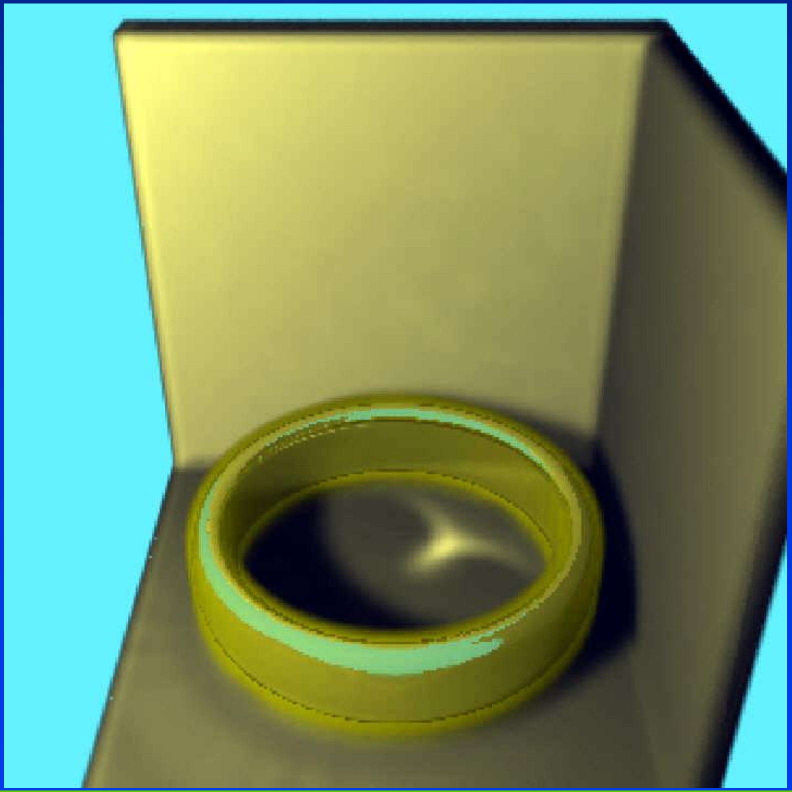
- With **gradient modulation** we modulate opacity/color of a voxel by gradient
- We multiply opacity and color by some function of gradient magnitude (or given by a transfer function, #5)
- Regions of high gradient magnitude increase opacity; regions of low gradient magnitude decrease opacity

- Explain this image

Volumetric Global Illumination

- Global illumination refers to reflections, shadows, and other effects that cannot be computed locally





Regions of Interest (ROIs)

- An ROI is simple a portion of the data-set of particular importance
- Use **cropping planes** to reveal interior
- Simple idea, but very, very useful
- Eliminate sets of voxels from consideration

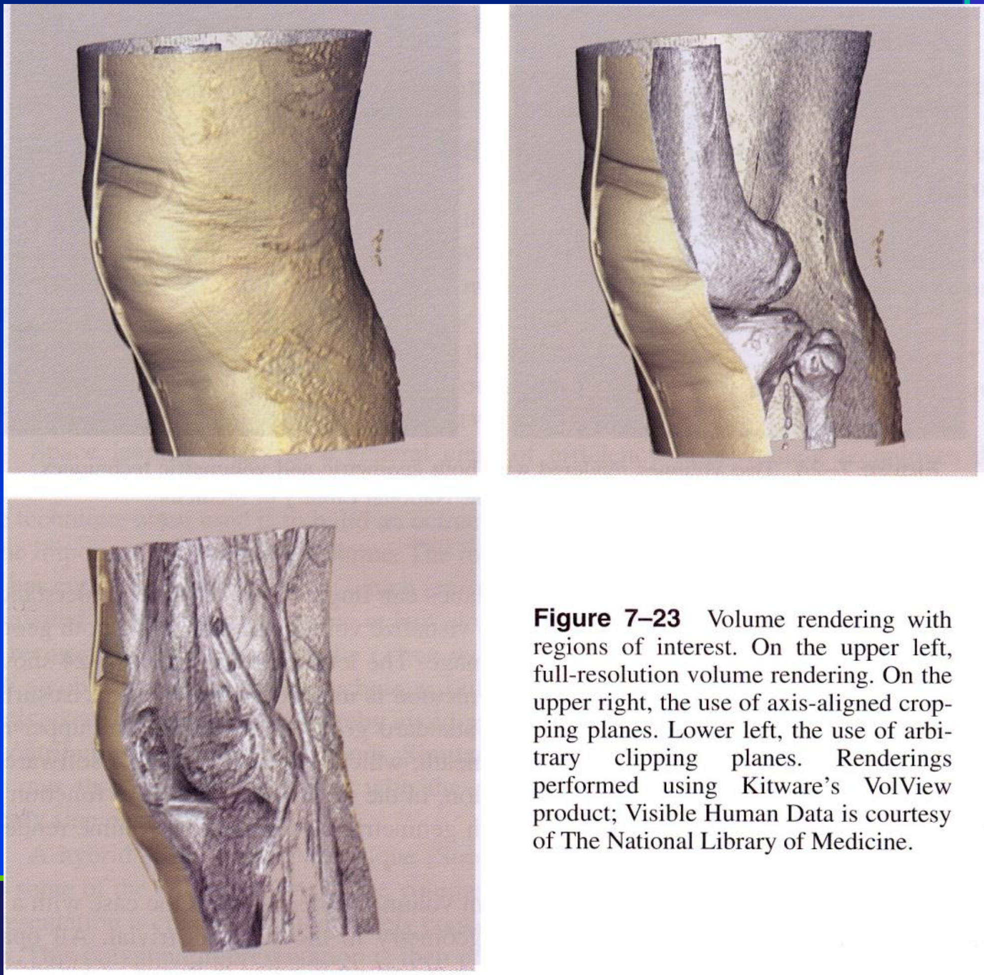


Figure 7–23 Volume rendering with regions of interest. On the upper left, full-resolution volume rendering. On the upper right, the use of axis-aligned cropping planes. Lower left, the use of arbitrary clipping planes. Renderings performed using Kitware's VolView product; Visible Human Data is courtesy of The National Library of Medicine.