Computer Graphics

- Volume Rendering -

Hendrik Lensch

[a couple of slides thanks to Holger Theisel]

Overview

- **Last Week**
  - Subdivision Surfaces

- **on Sunday**
  - Ida Helene
- **Today**
  - Volume Rendering
- **until tomorrow: Evaluate this lecture on**

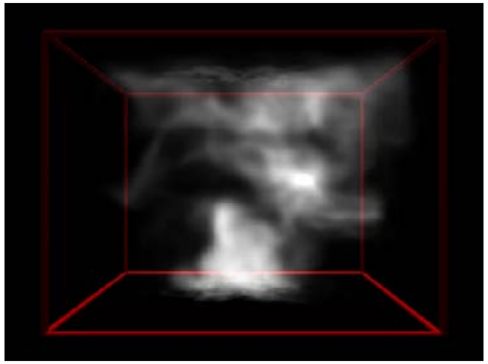
http://frweb.cs.uni-sb.de/03.Studium/08.Eva/

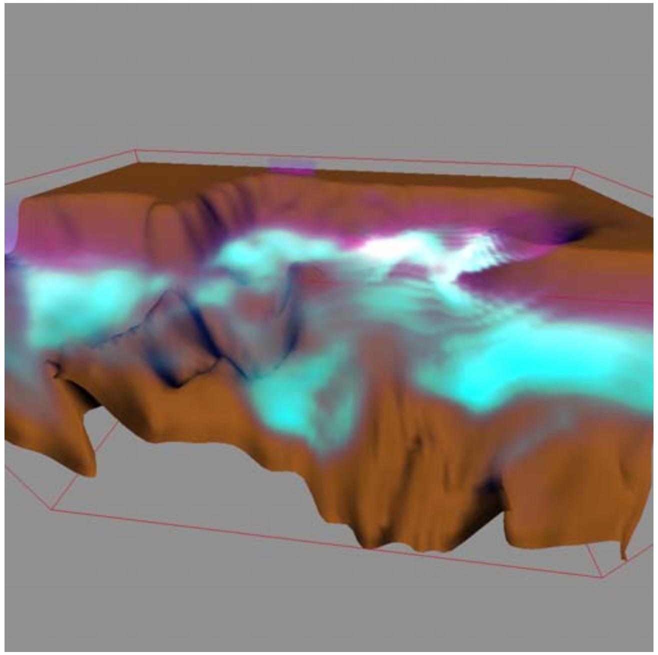
Motivation

- **Applications**
  - Fog, smoke, clouds, fire, water, …
  - Scientific/medical visualization: CT, MRI
  - Simulations: Fluid flow, temperature, weather, ...
  - Subsurface scattering
- **Effects in Participating Media**
  - Absorption
  - Emission
  - Scattering
    - Out-scattering
    - In-scattering
- **Literature**
  - Klaus Engel et al., *Real-time Volume Graphics*, AK Peters
  - Paul Suetens, *Fundamentals of Medical Imaging*, Cambridge University Press

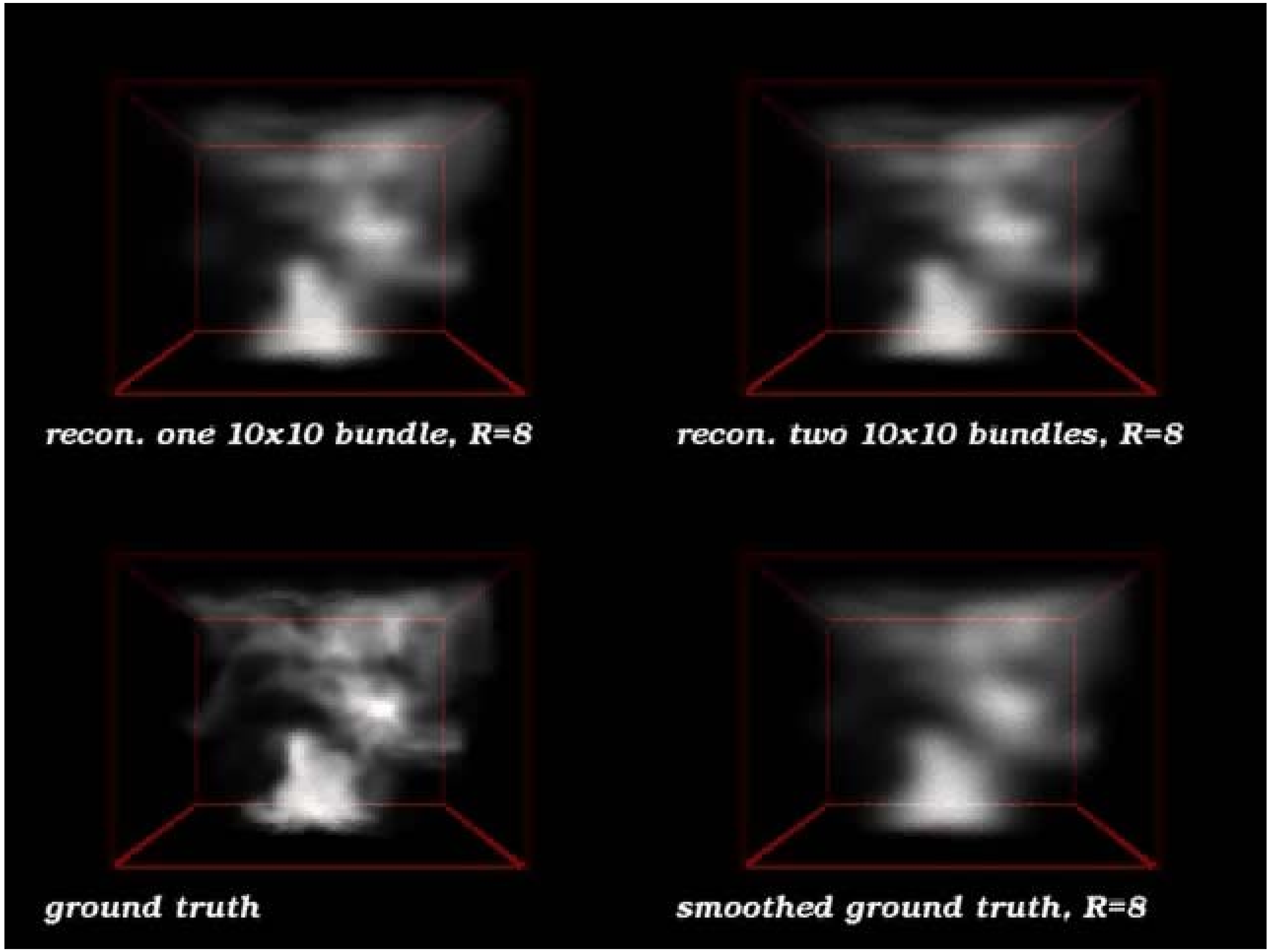
Motivation Volume Rendering

- **Examples of volume visualization:**

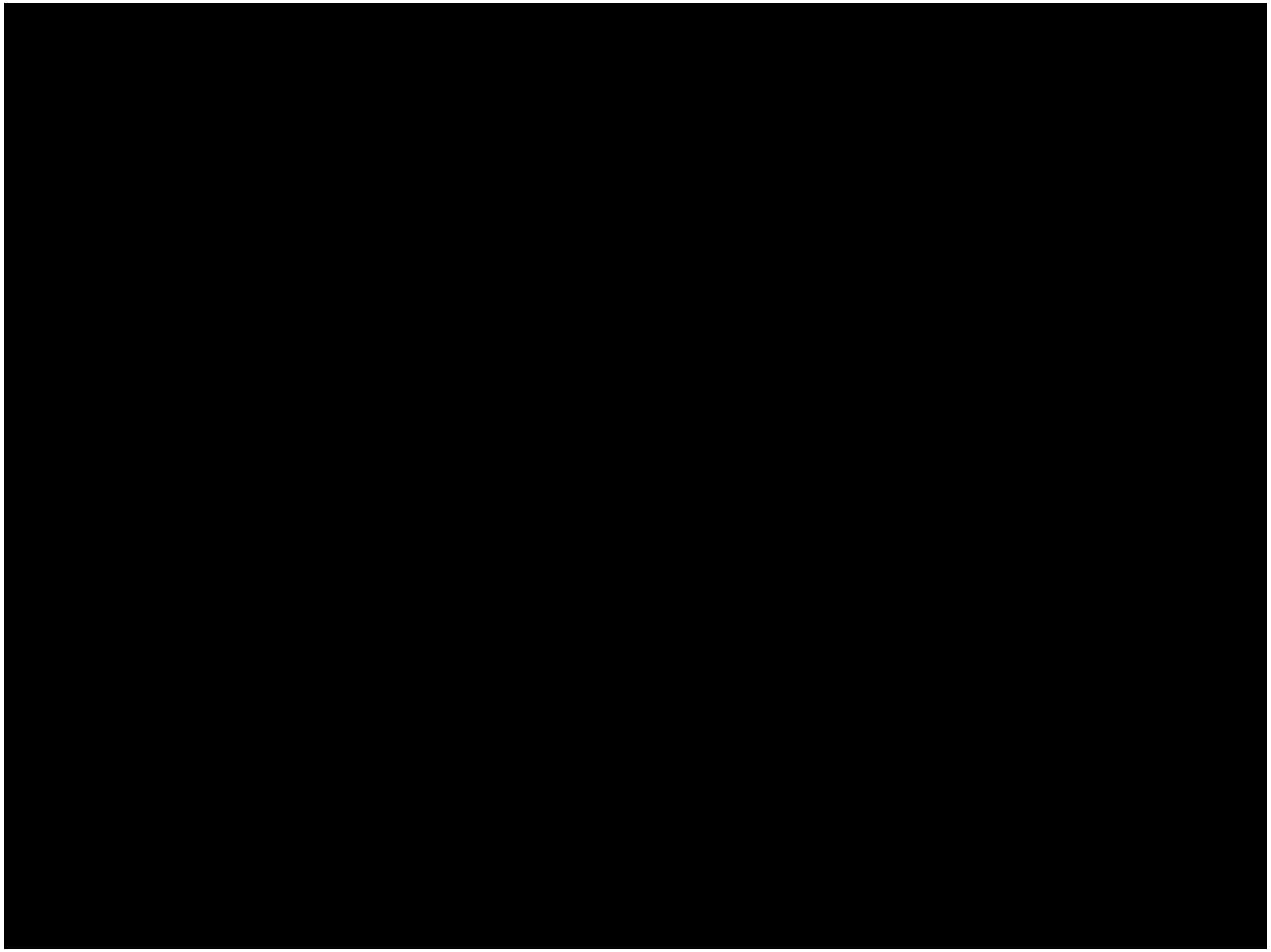




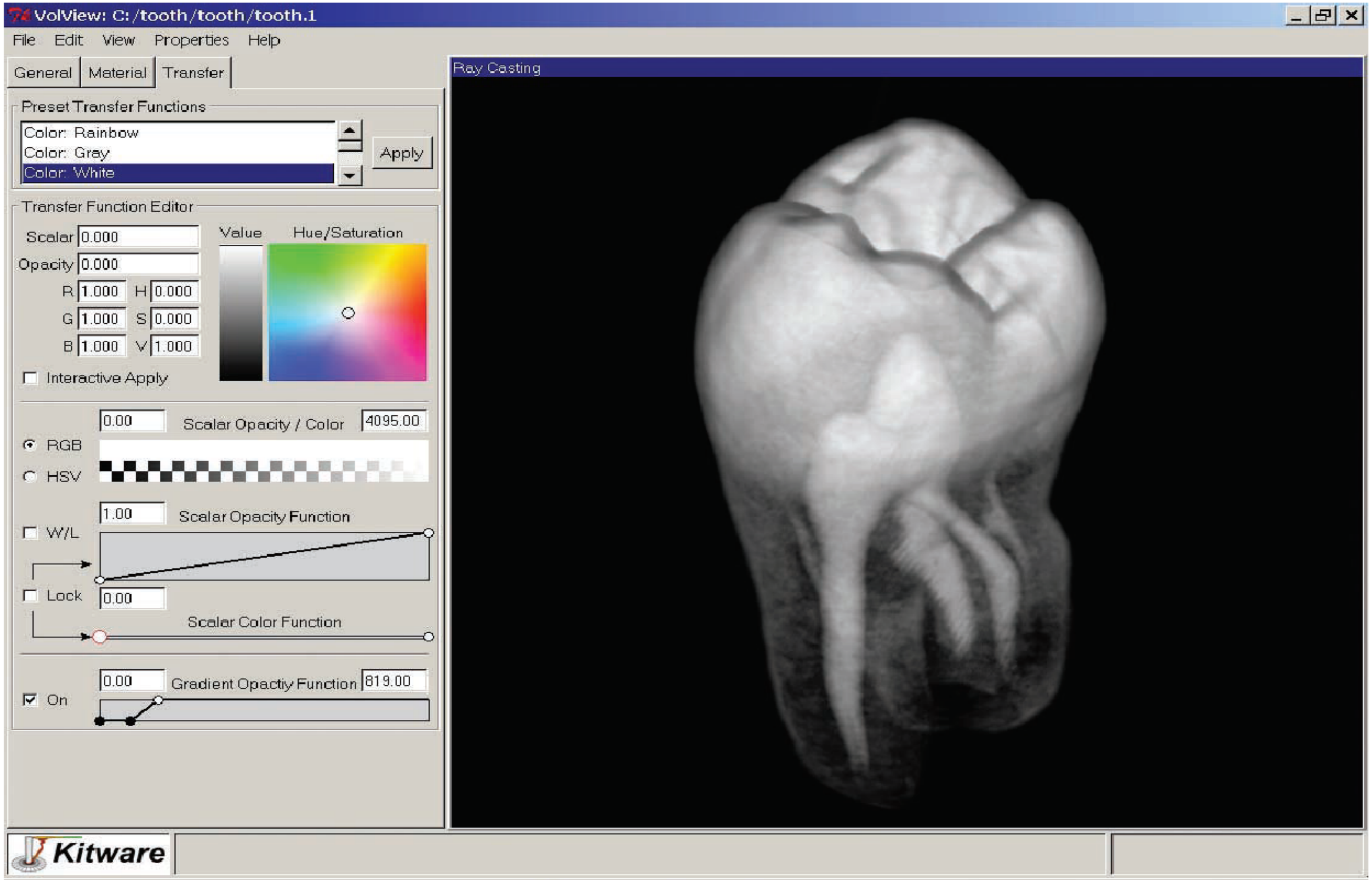
Direct Volume Rendering



Volume Acquisition

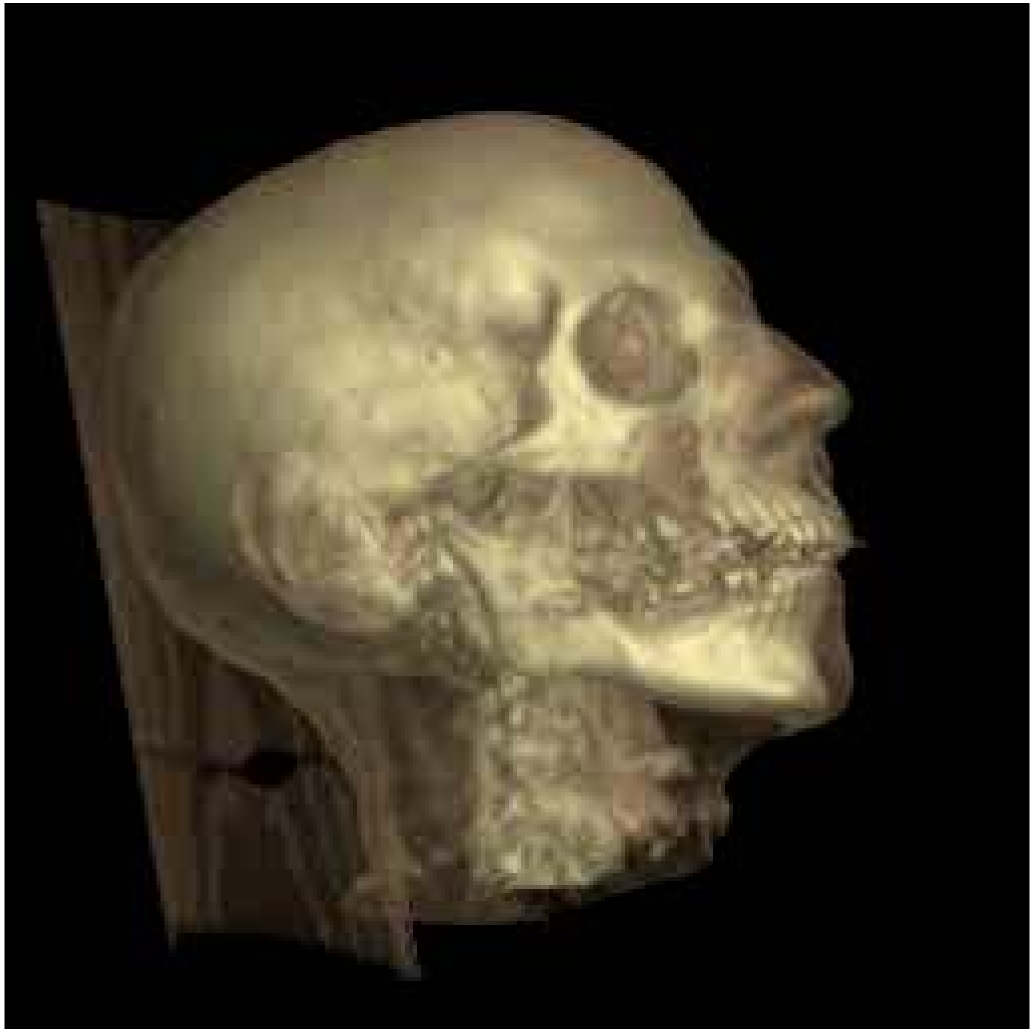


Direct Volume Rendering



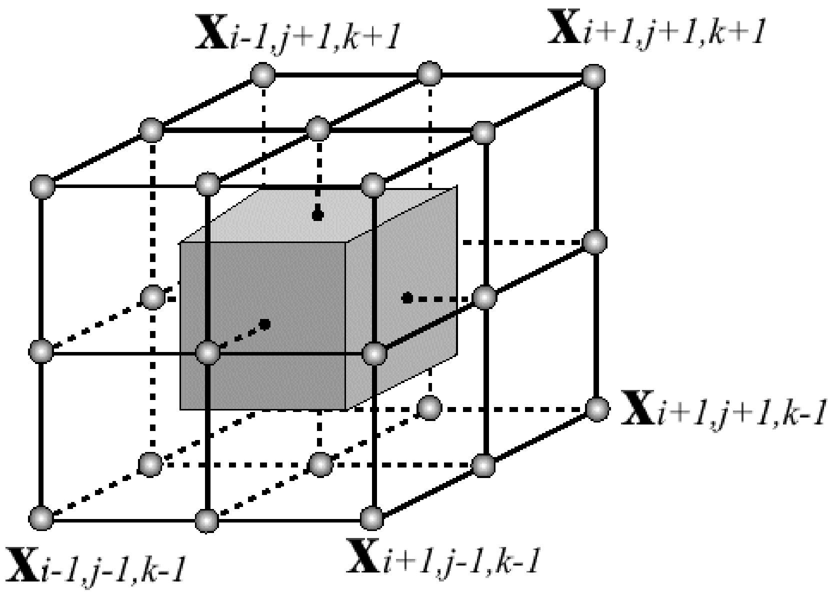
Direct Volume Rendering

- Shear-Warp factorization (Lacroute/Levoy 94)

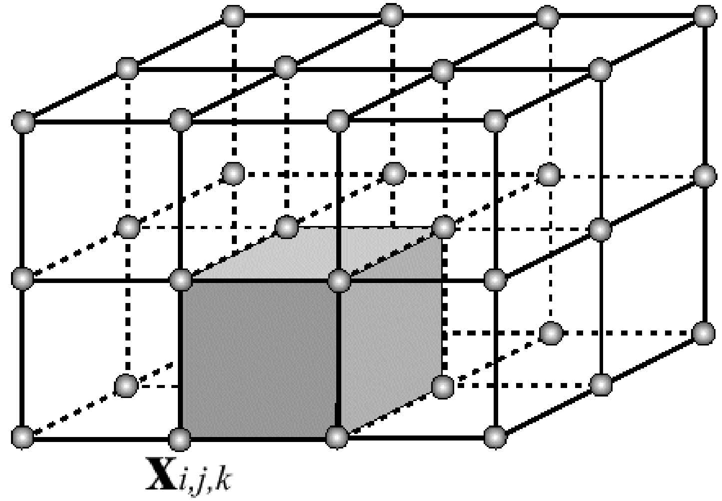


Volume Representations

- Cells and voxels



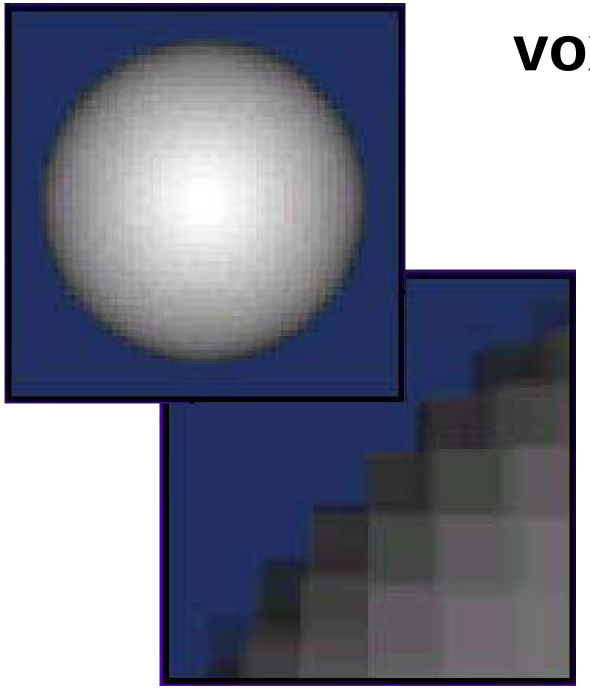
voxels: represent homogeneous areas



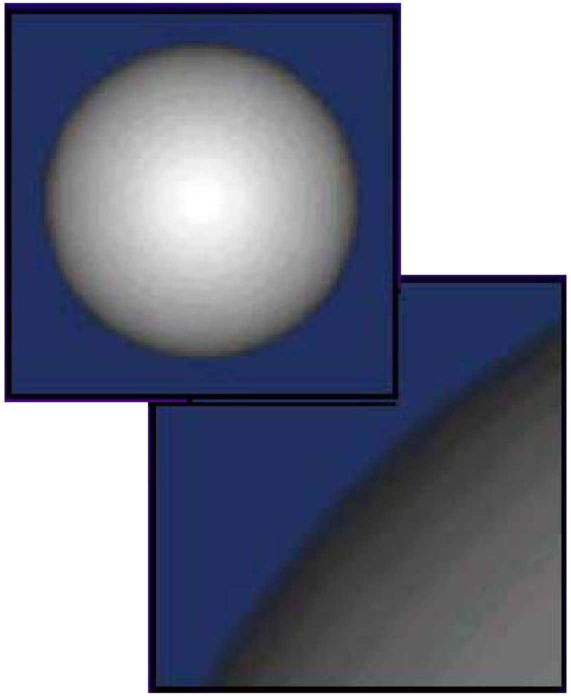
cells: represent inhomogeneous areas

Volume Representations

- **Cells and voxels**



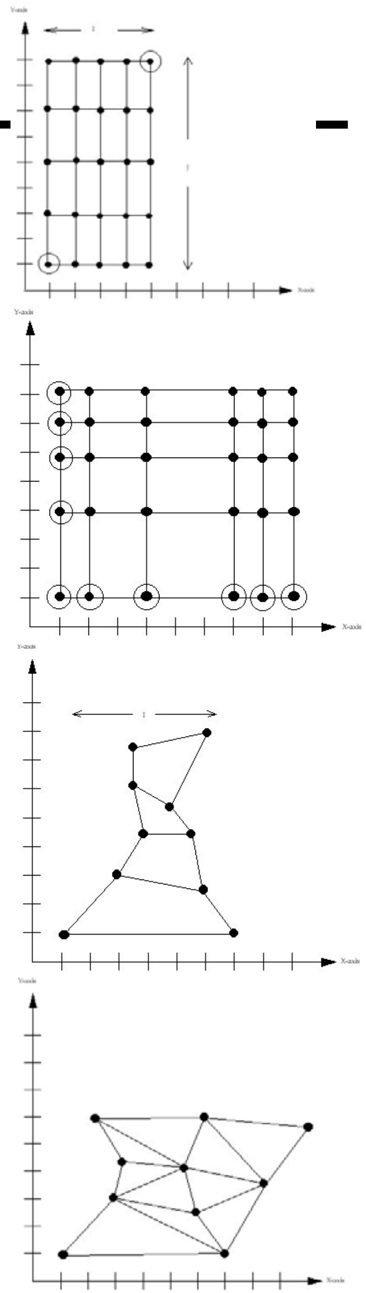
voxels: represent homogeneous areas

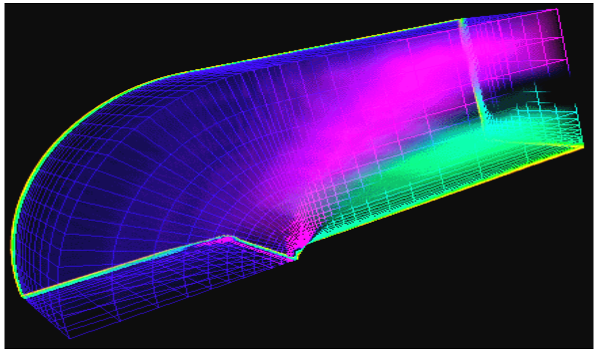


cells: represent inhomogeneous areas

Volume Representations

- **Simple shapes with procedural solid texture**
  - Ellipsoidal clouds with sum-of-sines densities
  - Hypertextures [Perlin]
- **3D array**
  - Regular (uniform) or rectilinear (rectangular)
  - CT, MRI
- **3D meshes**
  - Curvilinear grid (mapping of regular grid to 3D)
    - "Computational space" is uniform grid
    - "Physical space" is distorted
    - Must map between them (through Jacobian)
  - Unstructured meshes
    - Point clouds
    - Often tesselated into tetrahedral mesh)

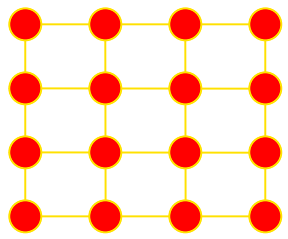




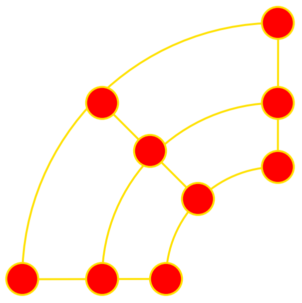
Curvilinear grid

Volume Organization

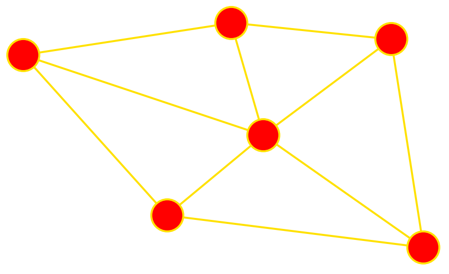
- **Rectilinear Grid:**
  - Wald et al.
  - Implicit kd-trees



- **Curvilinear Grid:**
  - Warped Rectilinear Grid
  - Hexahedral cells



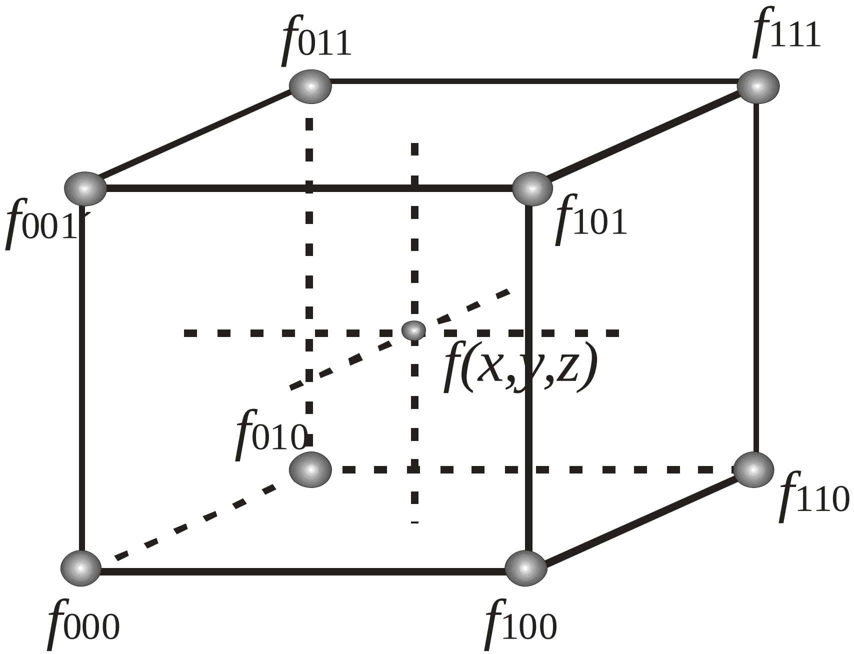
- **Unstructured Mesh:**
  - Tetrahedral cells



Trilinear Interpolation

- Cells

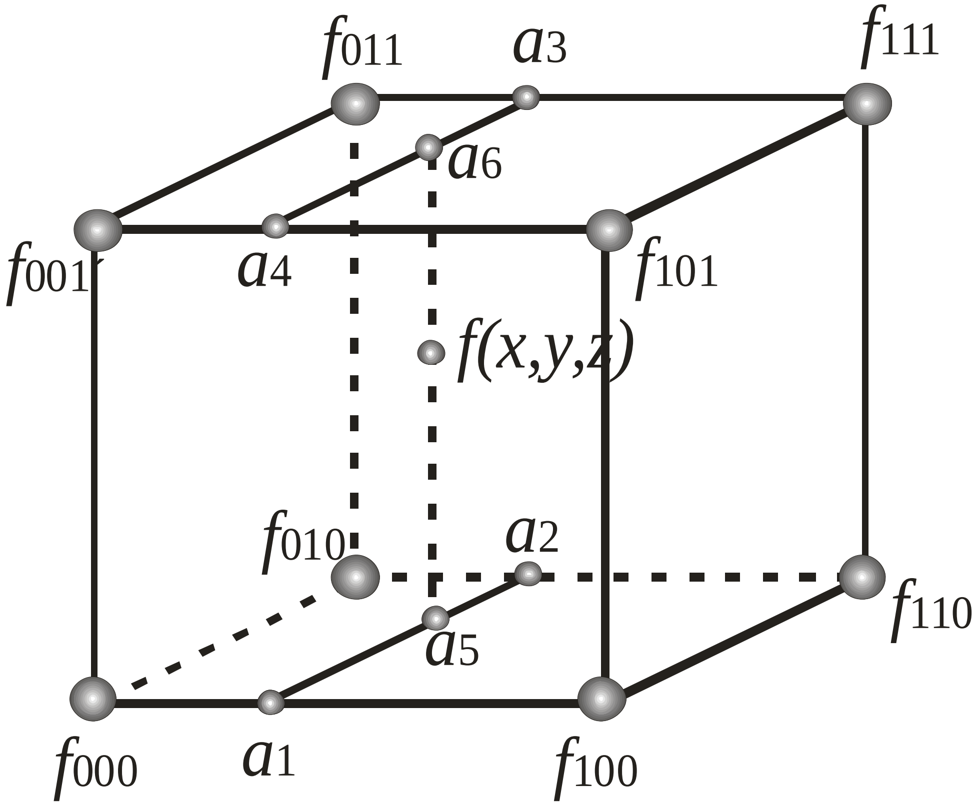
Data values inside a cell have to be computed by interpolation.



Most common interpolation for cells: trilinear interpolation

Trilinear Interpolation

Let $f_{ijk} = f(i,j,k)$ for $i,j,k \in \{0,1\}$. Then the value $f(x,y,z)$ for a certain point $(x,y,z) \in [0,1]^3$ inside the cell is computed by trilinear interpolation as:

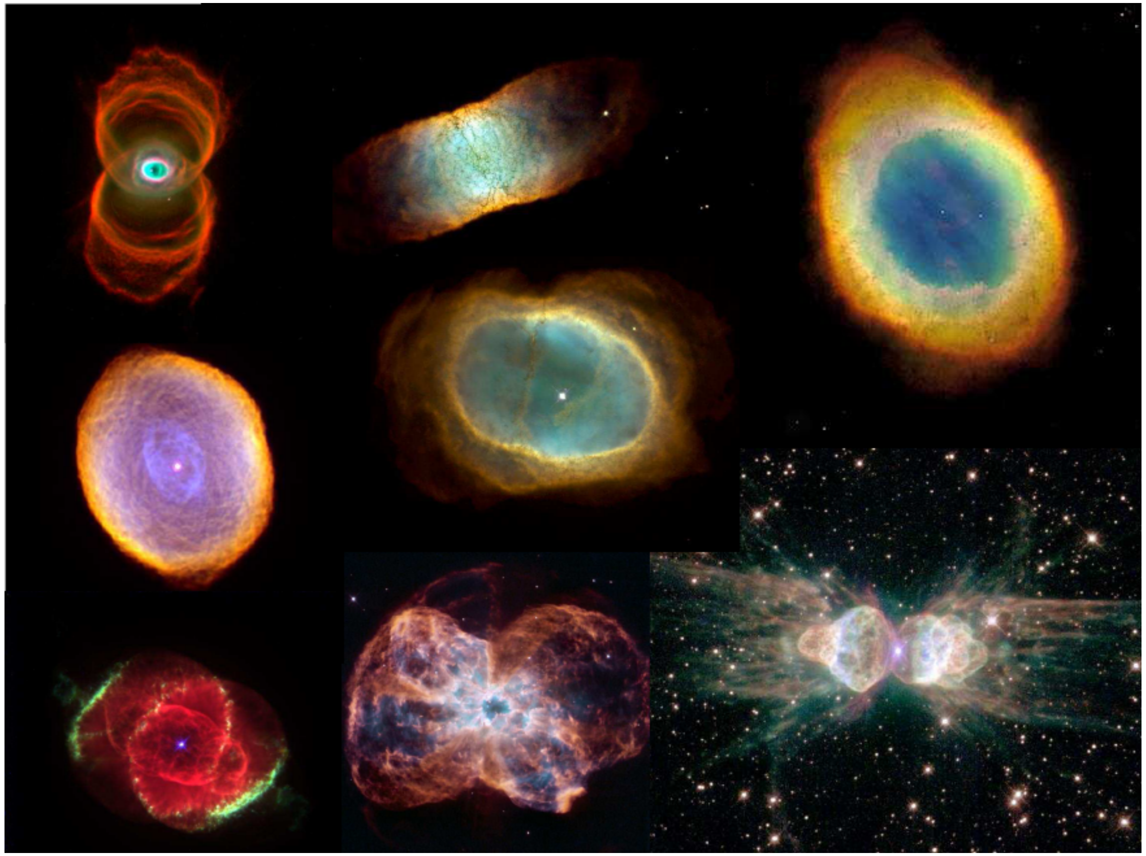


$a_1 = (1-x)*f_{000} + x*f_{100}$
$a_2 = (1-x)*f_{010} + x*f_{110}$
$a_3 = (1-x)*f_{011} + x*f_{111}$
$a_4 = (1-x)*f_{001} + x*f_{101}$
$a_5 = (1-y)*a_1 + y*a_2$
$a_6 = (1-y)*a_4 + y*a_3$

$$f(x,y,z) = (1-z)*a_5 + z*a_6$$

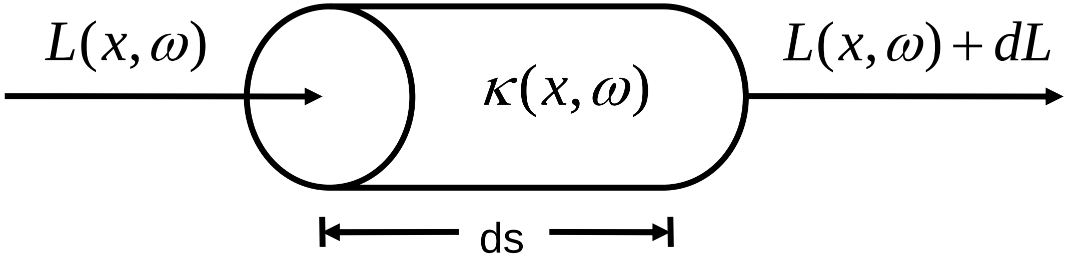
Participating Media

- **Absorption**
- **Emission**
- **In-Scattering**
- **Out-Scattering**
- **Multiple Scattering**



Absorption

- **Absorption Coefficient κ(x,ω)**
  - Probability of a photon being absorbed at x in direction ω per unit length



$$dL(x,\omega) = -\kappa(x,\omega)L(x,\omega)ds$$

$$\frac{dL}{ds}(x,\omega) = -\kappa(x,\omega)L(x,\omega)$$

  - Optical depth τ of a material of thickness s
    - Physical interpretation:
      - Measure for how far light travels before being absorbed

$$\tau(s) = \int_0^s \kappa(x+t\omega,\omega)dt \quad [= \kappa s, \quad iff \;\; \kappa = const]$$

Transparency and Opacity

- **Integration Along Ray**

$$\frac{dL}{ds}(x,\omega) = -\kappa(x,\omega)L(x,\omega) \quad and \quad \tau(s) = \int_0^s \kappa(x+t\omega,\omega)dt$$

$$L(x+s\omega,\omega) = e^{-\tau(s)}L(x,\omega) = T(s)L(x,\omega)$$

- **Transparency (or Transmittance)**

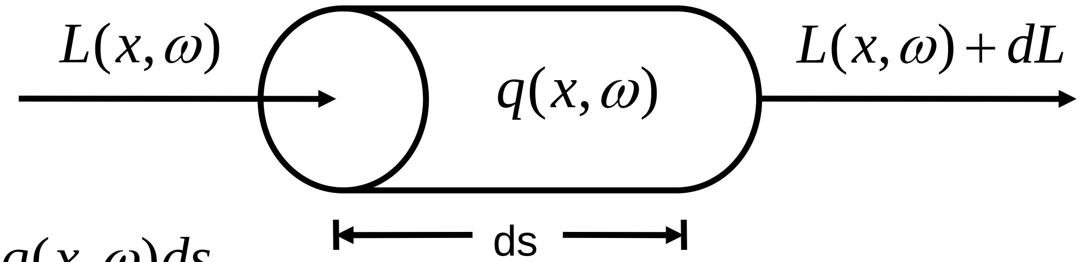
$$T(s) = e^{-\tau(s)} = e^{-\int_0^s \kappa(x+t\omega)dt}$$

- **Opacity**

$$O(s) = 1 - T(s)$$

Emission

- **Emission Coefficient q(x,ω)**
  - Number of photons being emitted at x in direction ω per unit length



$$dL(x,\omega) = q(x,\omega)ds$$

$$\frac{dL}{ds}(x,\omega) = q(x,\omega)$$

Emission-Absorption Model

- **Emission-Absorption Model**
  - Kombines absorption and emission only
- **Volume Rendering Equation**
  - In differential form

$$\frac{dL}{ds}(x,\omega) = -\kappa(x,\omega)L(x,\omega) + q(x,\omega)$$

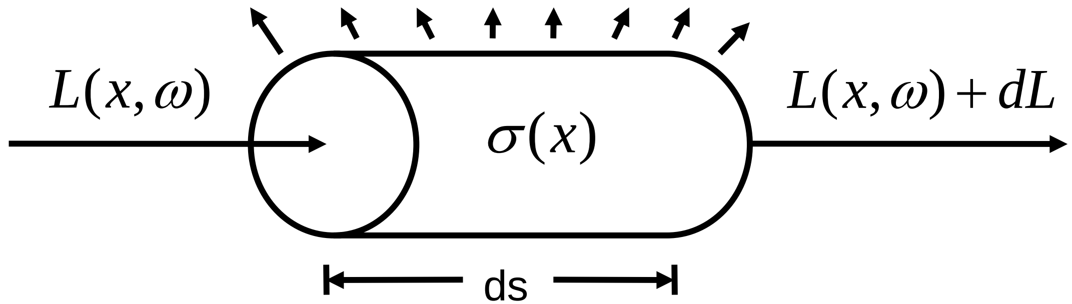
- **Volume Rendering Integral**

$$L(x+s\omega,\omega) = L(x,\omega)e^{-\int_0^s \kappa(t)\,dt} + \int_0^s q(s')e^{-\int_{s'}^s \kappa(t)\,dt}\,ds'$$

  - Incoming light is absorbed along the entire segment
  - Emitted light is only absorbed along the remaining segment
  - Must integrate over emission along the entire segment

Out-Scattering

- **Scattering cross-section σ(x,ω)**
  – Probability of a photon being scattered out of direction per unit length



  – Total absorption (extinction): true absorption plus out-scattering

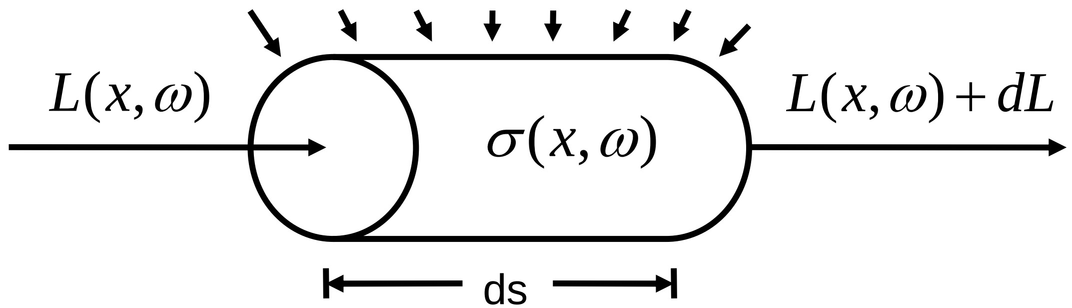
$$\chi = \kappa + \sigma$$

  – Albedo ("Weißheit", measure for reflectivity or ability to scatter)

$$W = \frac{\sigma}{\chi} = \frac{\sigma}{\kappa + \sigma}$$

In-Scattering

- **Scattering cross-section σ(x,ω)**
  - Number of photons being scattered into path per unit length
  - Depend on scattering coefficient (probability of being scattered) and the phase function (directional distribution of out-scattering events)



$$j(x,\omega) = \int_{S^2} \sigma(x,\omega_i)\, p(x,\omega_i,\omega) L(x,\omega_i)\, d\omega_i$$

  - **Total Emission: true emission q plus in-scattering j**

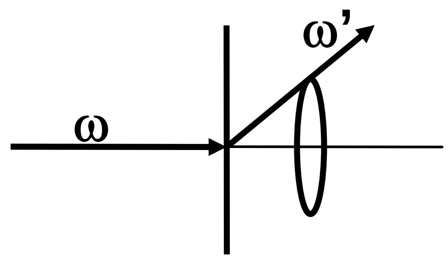
$$\eta(x,\omega) = q(x,\omega) + j(x,\omega)$$

  - **Phase function** (essentially the BRDF for volumes)

$$p(x,\omega_i,\omega)$$

Phase Functions

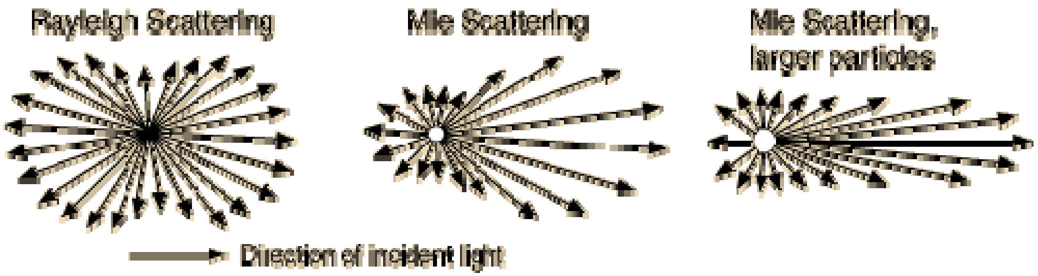
- **Phase angle is often only relative to incident direction**
  - cos θ = ω·ω'



- **Reciprocity and energy conservation**

$$p(x,\omega_i,\omega) = p(x,\omega,\omega_i)$$

$$\frac{1}{4\pi}\int_{S^2} p(x,\omega_i,\omega)d\omega = 1$$



- **Phase functions**
  - Isotropic

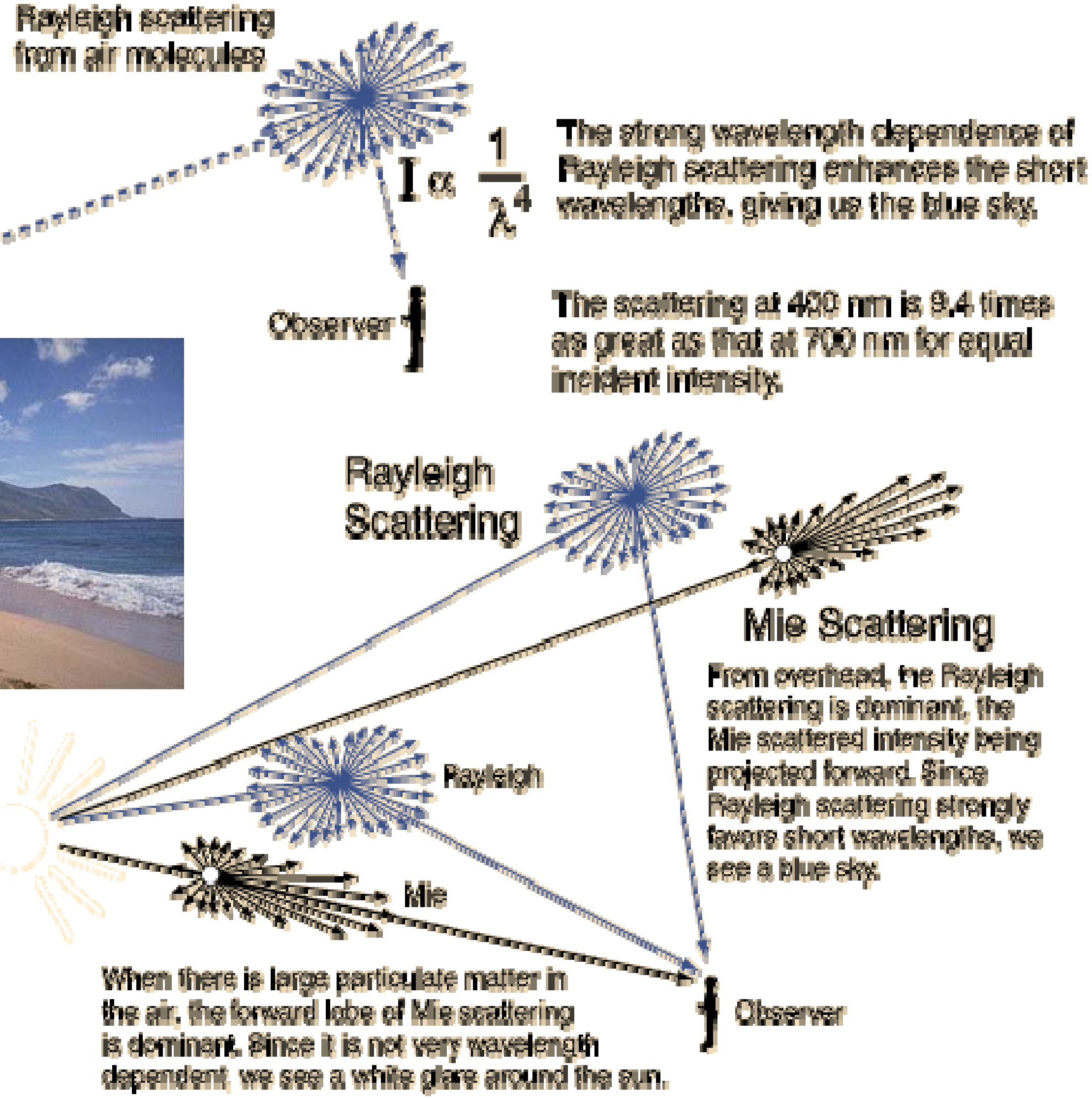
$$p(\cos\theta) = 1$$

  - Rayleigh (small molecules)
    - Strong wavelength dependence

$$p(\cos\theta) = \frac{3}{4}\frac{1+\cos^2\theta}{\lambda^4}$$

  - Mie scattering (larger spherical particles)

Rayleigh and Mie Scattering



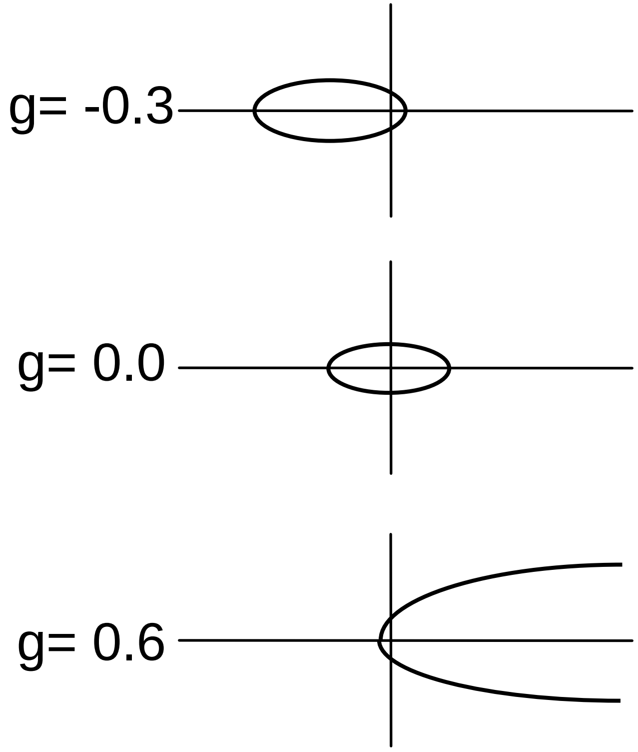
Henyey-Greenstein Phase Function

- **Empirical Phase Function**
  - Often used for interstellar clouds, tissue, and similar material

$$p(\cos\theta) = \frac{1}{4\pi}\frac{1-g^2}{\left(1+g^2-2g\cos\theta\right)^{3/2}}$$

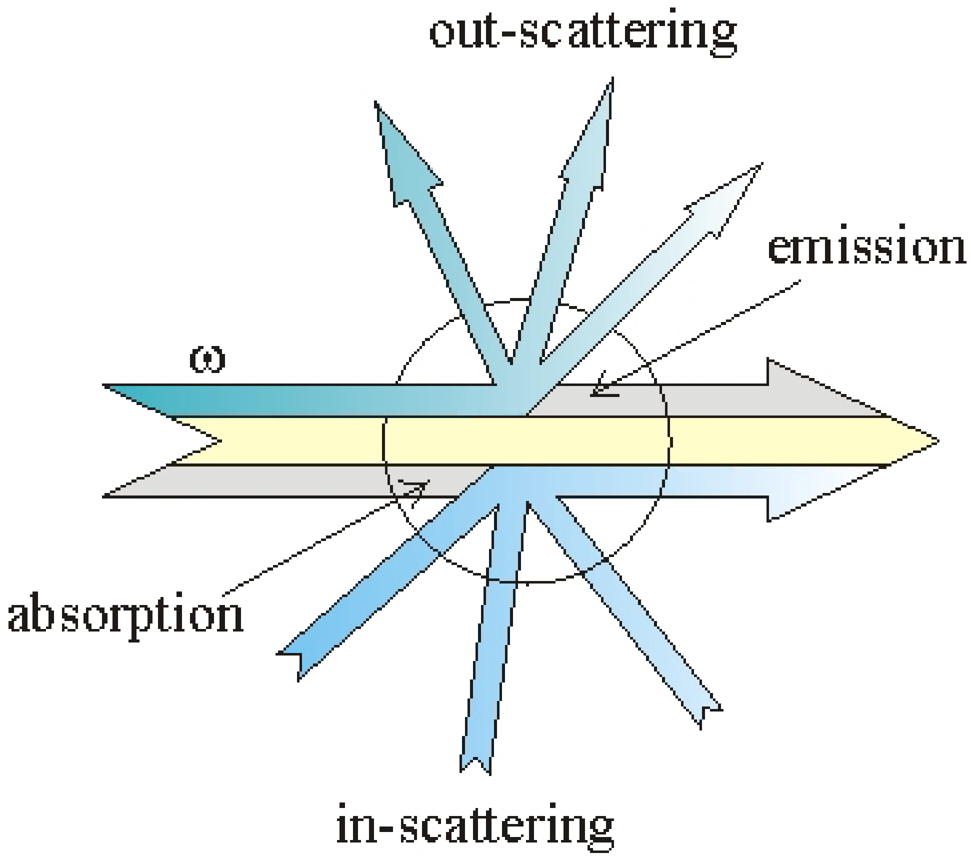
  - Average cosine of phase angle

$$g = 2\pi\int_0^{\pi} p(\cos\theta)\cos\theta\, d\theta$$



Summary

- **Scattering in a volume**



Full Volume Rendering

- **Full Volume Rendering Equation**

$$\omega \cdot \nabla_x L(x,\omega) =$$

$$\frac{\partial L(x,\omega)}{\partial s} = -\chi(x,\omega)L(x,\omega) + q(x,\omega) + \int_{S^2} \sigma(x,\omega_i)\, p(x,\omega_i,\omega) L(x,\omega_i)\, d\omega_i$$

$$\nabla_x = \left(\partial/\partial x, \partial/\partial y, \partial/\partial z\right) \quad \text{at point x}$$

- **Full Volume Rendering Integral**

$$L(x+s\omega,\omega) = \int_0^s e^{-\int_{s'}^{s} \chi(x+t\omega,\omega)\,dt}\, \eta(x+s'\omega,\omega)\, ds'$$

Attenuation:
(absorption &
out-scattering)

Source Term:
in-scattering,emission,
and background
(η(0,ω)= L(x,ω)δ(x))

Simple Atmosphere Model

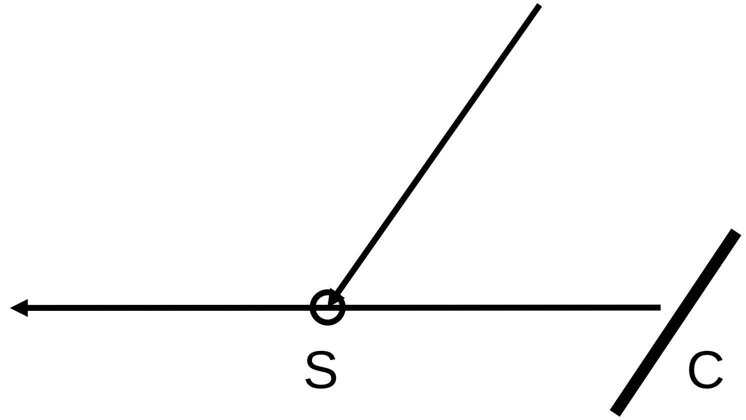
- **Assumptions**
  - Homogeneous media (κ = const)
  - Constant source term q (ambient illumination)

$$\frac{\partial L(s)}{\partial s} = -\kappa L(s) + q$$

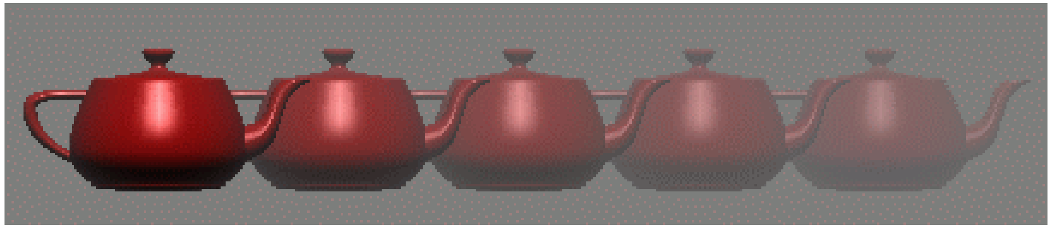
$$L(s) = e^{-\kappa s} C + \int_0^s e^{-\kappa s'} q \, ds'$$

$$L(s) = e^{-\kappa s} C + \left(1 - e^{-\kappa s}\right) q$$

$$L(s) = T(s) C + (1 - T(s)) q$$



- **Fog and Haze (in OpenGL)**
  - Affine combination of background and fog color
  - Depending on distance



Volume Visualization

Two ways of graphical representation of volume data

1) extracting geometry

-> Isosurfaces

-> different extraction approaches

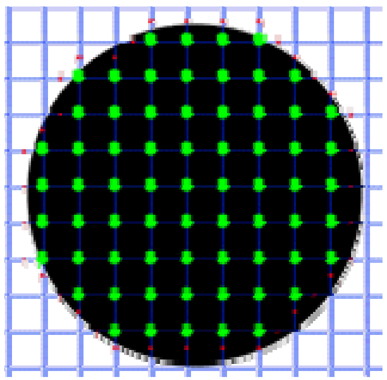
-> Most famous: Marching Cubes

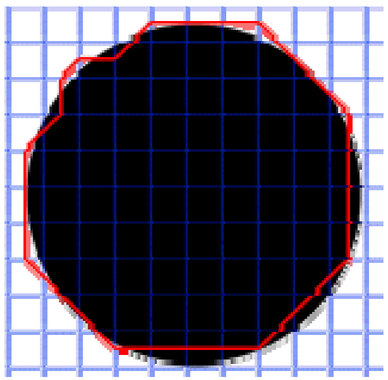
2) direct rendering of the whole volume (direct volume rendering)

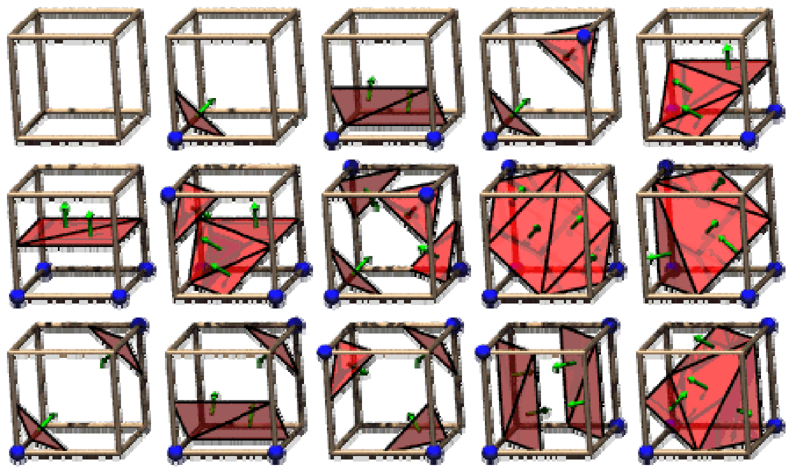
-> here in more detail

Indirect Volume Rendering

- **Iso-Surfaces**
  - Compute iso-surface for v(x,y,z)= C and shade as normal
- **Ray Tracing**
  - Intersect ray with cubic surface defined by values at vertices
  - Several accurate and/or fast algorithms
- **Marching Cubes algorithm**
  - Iterate over all voxels
  - Classify voxel into 15 classes (by symmetry) ➔ surface topology
  - Compute vertex location by interpolation
  - Render as triangle mesh







The 15 Cube Combinations

Raycarsting vs. Projection

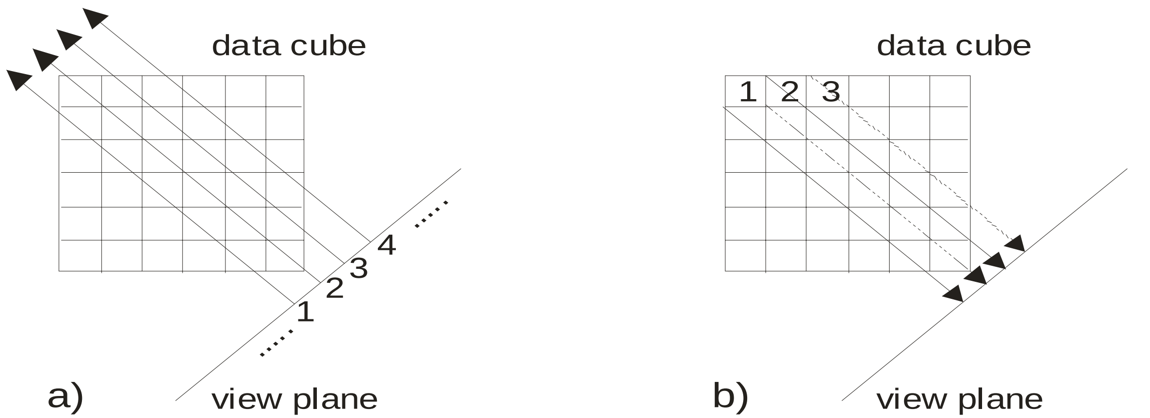
- Two Methods for Direct volume rendering

1. **Raycasting**
   (send a ray through the data volume; evaluation of the color distribution concerning the hit volume elements)

   for each ray do
   for each voxel-ray intersection d
   calculate pixel contribution

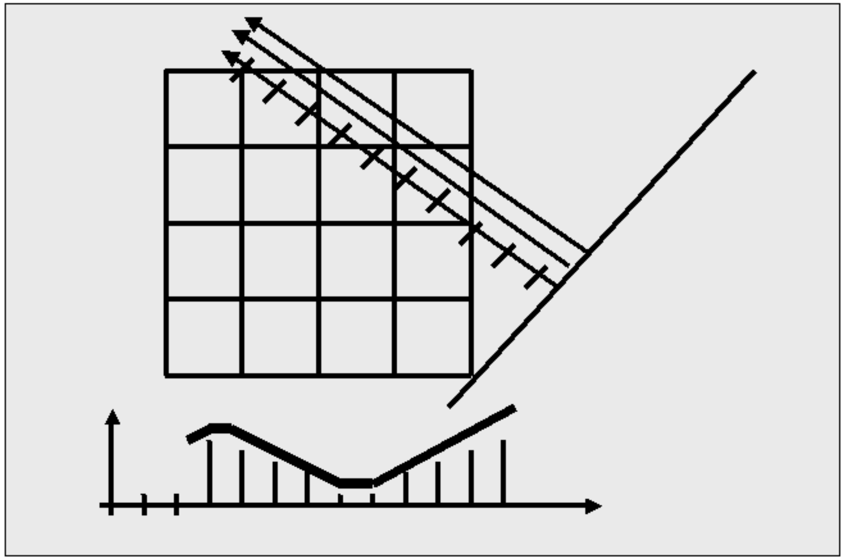
2. **Projection of the volume elements onto screen**

   for each voxel or cell do
   for each pixel projected onto do
   calculate pixel contribution

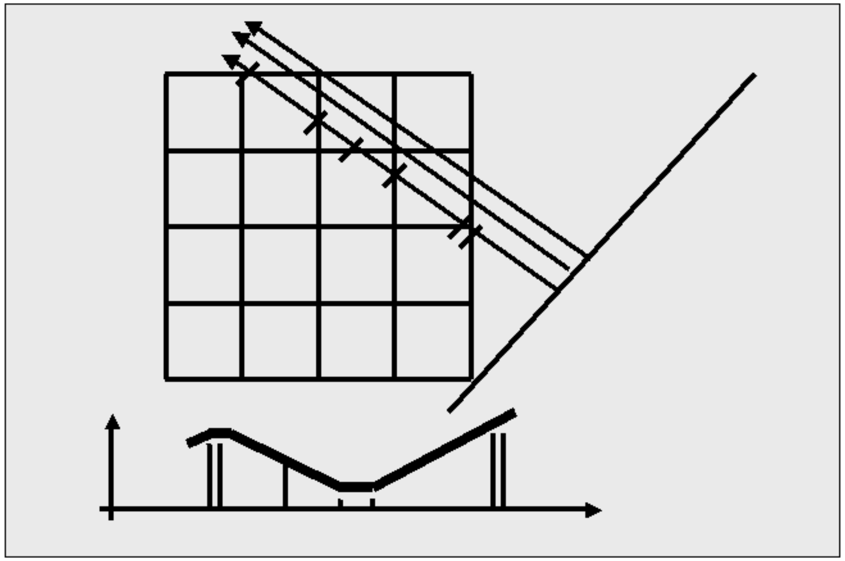


Raycasting

There are two ways to evaluate color and transparency properties for raycasting:



equidistant stepsize

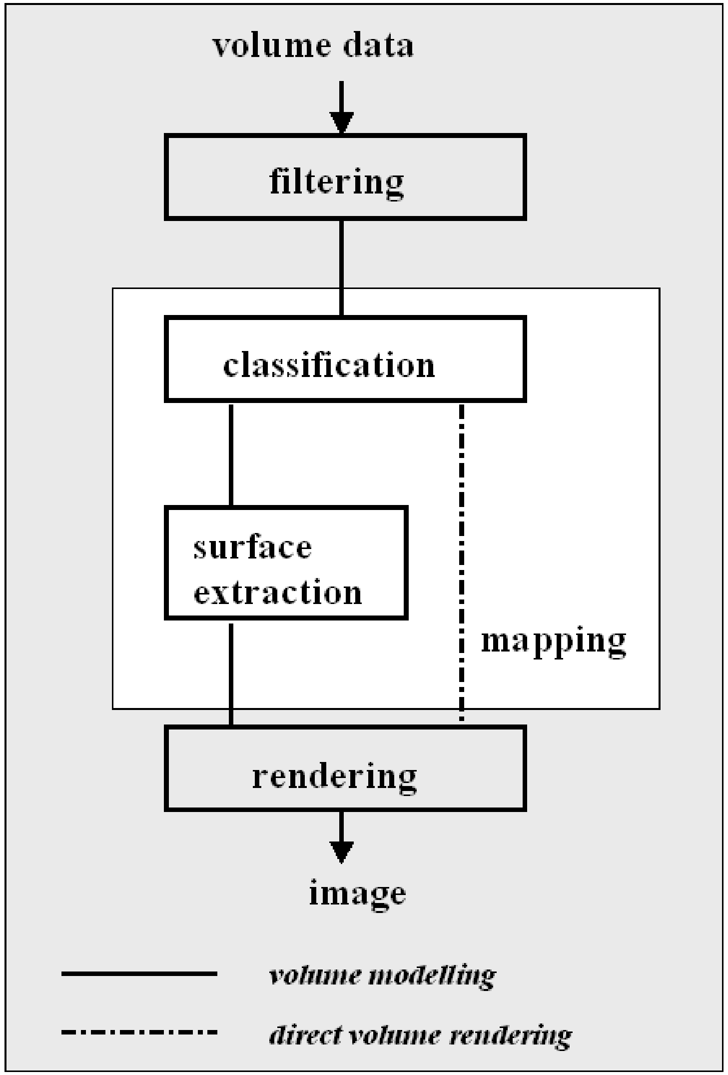


intersection ray / volume element

Transfer Functions

- **Classification using transfer functions**
  - Map value given in the volume to optical properties
  - Typical: One-dimensional transfer functions
    - $\kappa(x,\omega)= T_\kappa(v(x))$ and $q(x,\omega)= T_q(v(x))$
  - Multidimensional transfer functions
    - Depend on value v(x) and its gradient grad(v(x))
    - $\kappa(x,\omega)= T_\kappa(v(x), grad(v(x)))$ and $q(x,\omega)= T_q(v(x), grad(v(x)))$
- **When to apply them**
  - Before (pre-) or after (post-classification) interpolation?
  - Post-classification is more appropriate
    - Transfer function generally modifies frequency spectrum of volume
    - Sampling of volume is chosen according to data not for any high-frequency modulation of it
- **Pre-Integrated Transfer Functions**
  - Assume linear interpolation of κ and q inside small segments
  - Precompute integral value for all tuples $(v_0,v_1,\Delta s)$

Steps in Volume Visualization



Volume Processing Pipeline

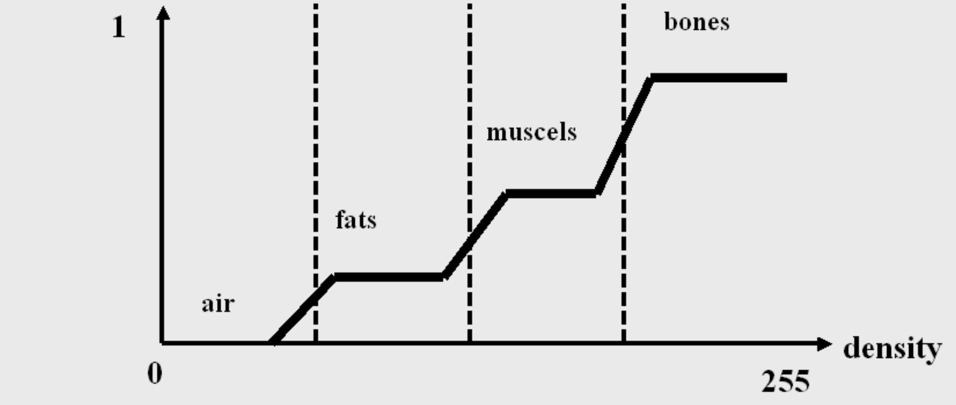
1. Filtering

- data acquisition
- data conversion
- data completion
- data reduction
- filter operators

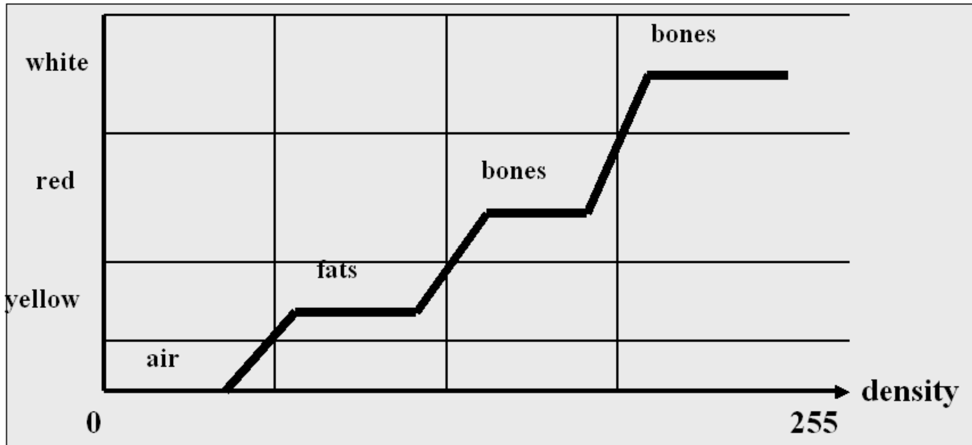
2. Classification

- for each volume element the distribution of the containing materials is computed
- for each material transparency and color is specified
- multiply the percentage of materials with assigned properties

Transfer Functions



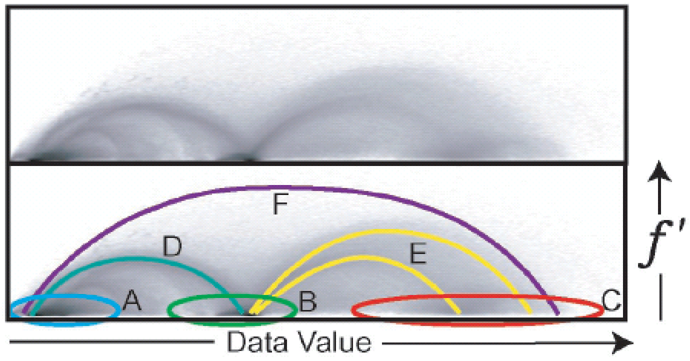
Assignment of transparencies



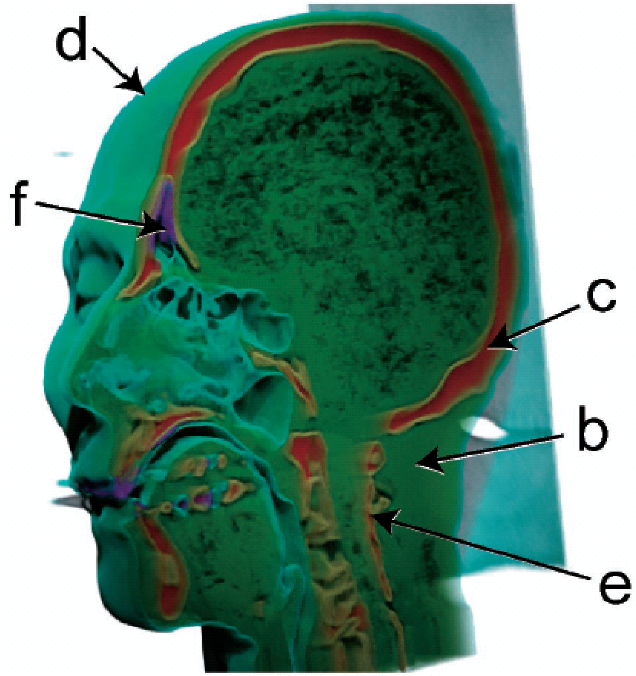
Assignment of colors

Transfer Functions

- 2D Transfer functions:
- **make transfer functions depend not only on scalar value but also on magnitude of the gradient**
- **emphasizes material boundaries**
- **strong gradient -> more opacity**
- **diminishes homogenuous areas**



(b)



[Kniss et al, 2002]

Direct Volume Rendering

- Idea: **collect contributions (using a local lightning model) along a viewing ray**
- **for surface rendering the normals are necessary; they can be computed using *gradients*.**
- **The gradient *grad f* over a scalar function *f = f(x,y,z)* is definied as:**

$$grad\ f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)^T = \left(f_x, f_y, f_z \right)^T$$

Gradients

Gradients express the difference of the data values along the axes.

They are perpendicular to the isosurfaces *f(x,y,z)=const*; thus they can be used to estimate the surface normals.

For piecewise trilinear scalar fields, gradients are computed using central differences:

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

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

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

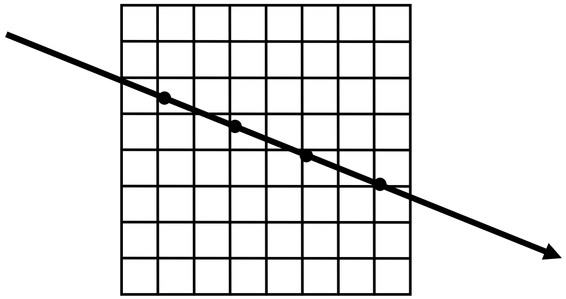
where G_x, G_y and G_z are the components of the gradients and s_x, s_y, s_z is the stepsize along the regular grid in x-, y- and z-direction.

Direct Volume Rendering

- Properties:
  - No binary classification
  - Show small details
  - Compute complexity depends on volume size; but parallelization possible;
  - combination with geometrical data not trivial, no traditional rendering

Compositing Along a Ray

- **Incremental compositing algorithm**
  - As seen from the viewer (s_n is at front)
- **Two Approaches**
  - Front to back (start at s_n) and back to front (start at s_0)
  - Accumulate color and opacity

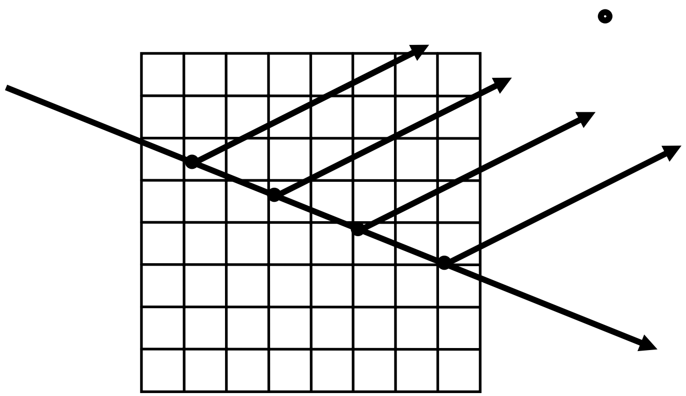


- **Algorithm (front to back)**
  - Allows for early ray termination
  - $C = C_n$, $\alpha=0$ (Opacity)
  - for (i=n-1; i >= 0; i--)
  - $C \mathrel{+}= (1-\alpha)*c_i$
  - $\alpha \mathrel{+}= (1-\alpha)(1-T_i)$
  - if (α > threshold) break
  - $C \mathrel{+}= (1-\alpha)C_{background}$

- **Algorithm (back to front)**
  - Does not allow for termination
  - $C = C_{background}$
  - for (i=0; i <= n; i++)
  - $C = (1-T_i)C + c_i$

Single Scattering

- **Single scattering approximation**
  - Compute illumination via shadow ray
    - Accumulate transparency along the way
    - Multiply with scattering coefficient, phase function, and light radiance
  - Accumulate front to back
    - Illumination from light source
    - Weight with transparency
    - Accumulate transparency
  - Add background illumination times transparency




```
T=1
L=0
for (s=0; s < 1; s+= ds)
    j= σ(s) * p(ω, ω_L) * L_s * T_S
    L += T*j*ds
    T *= (1- τ(v(s)))
L+= T * L_0
```

Shadow Ray:

```
T_S=1
for (t=0; t < 1; t+= dt)
    T_S *= (1- τ(v(t)))*dt
```

Multiple Scattering

- **Highly computationally demanding**
  - Zonal method (FE-Technique) [Rushmeier'87]
    - Assume constant, isotropic scattering in voxels
    - Set up linear system (a la radiosity) and solve numerically
    - Also includes surface interactions (SS, SV, VS, VV)
  - P-N (P_N) method [Kajiya'84]
    - Represent light distribution at each point in Spherical Harmonics (SH)
    - Compute interactions of SH-coefficients an solve numerically
  - Discrete Ordinate method [Languénou'95]
    - Choose M fixed directions to redistribute energy in
    - Can generate "ray effects" due to fixed directions
      - Should distribute in solid angle
  - Diffusion process [Stam'95]
    - Assumes optically dense medium → much scattering → uniform diffusion
    - Recently also used for sub-surface scattering approximation
    - E.g. computes Point Spread Function (PSF)

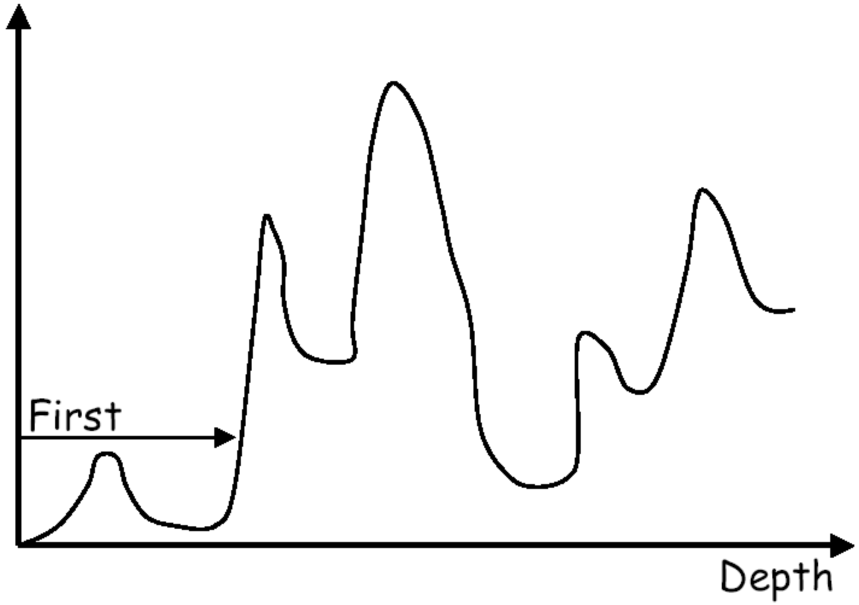
Cost Reduction for Ray Casting

- Early Ray Termination**:**

  check transparency; if beyond certain threshold: stop process;
- Increase number of sent rays adaptively
  Ray is sent for group of pixels, i.e. 3*3; if values of adjacent rays differ significantly: additional rays are sent.
- Discretization of rays
  describe a ray as set of 3D points (artifacts possible)
- 3D distance transformations
  per volume element: coding the distance to the next volume element -> skip areas of low interest.

Cost Reduction for Composition

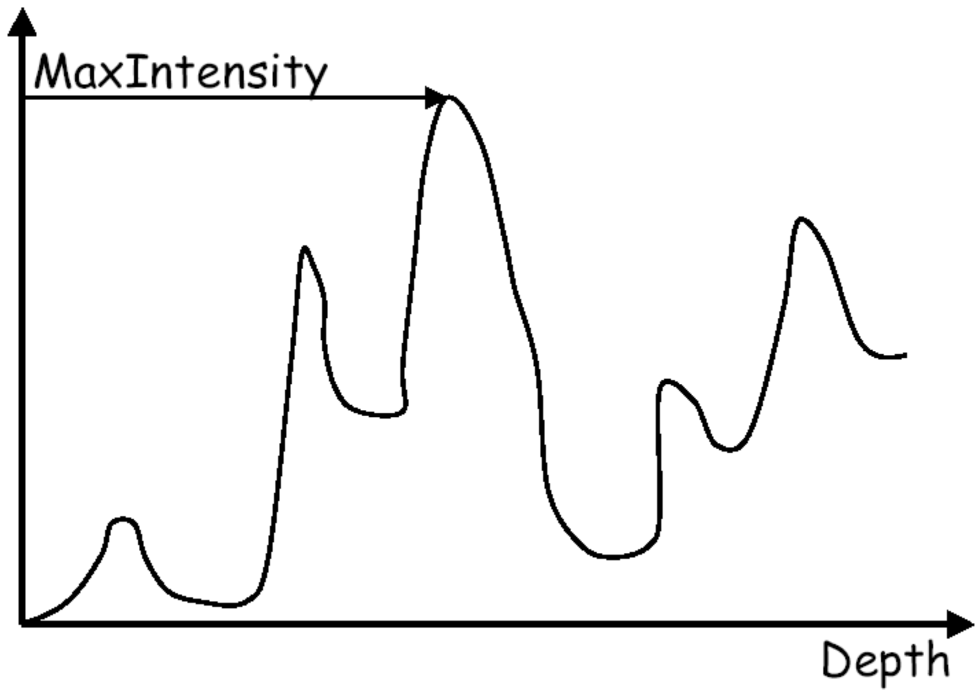
- first hit

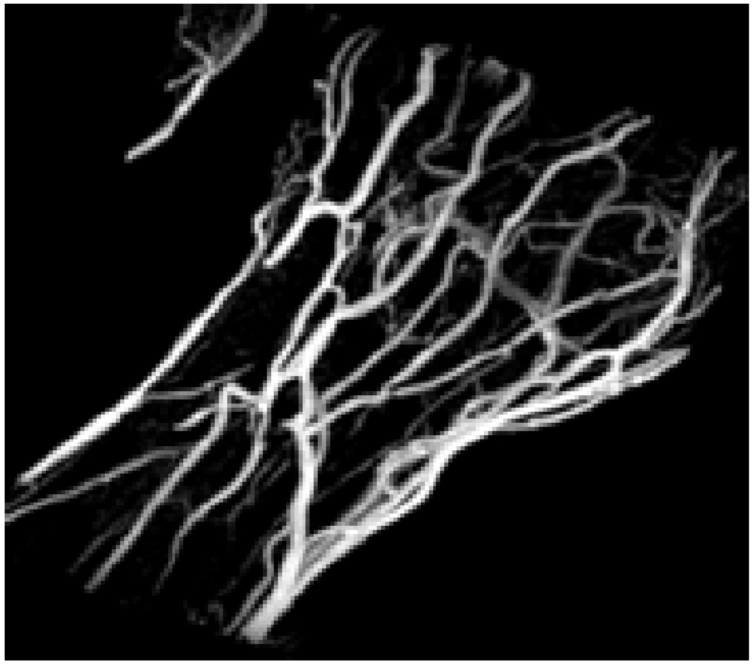


First: Extracts iso-surfaces (again!), done by Tuy&Tuy '84

Cost Reduction for Composition

- **maximum intensity projection**

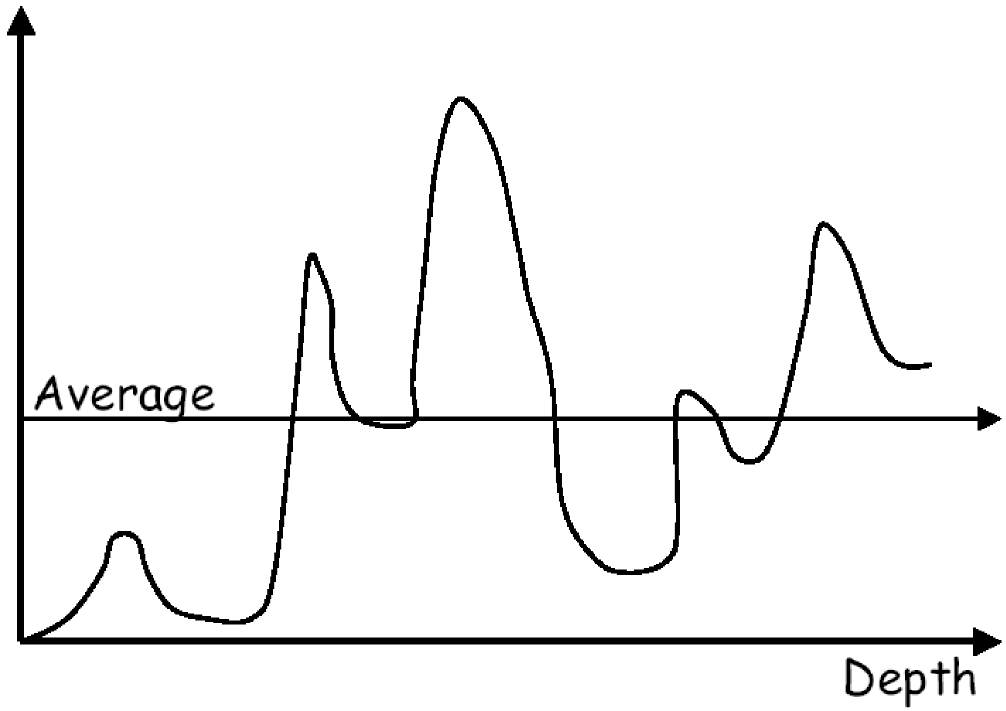




Max: Maximum Intensity Projection used for Magnetic Resonance Angiogram, for example

Cost Reduction for Composition

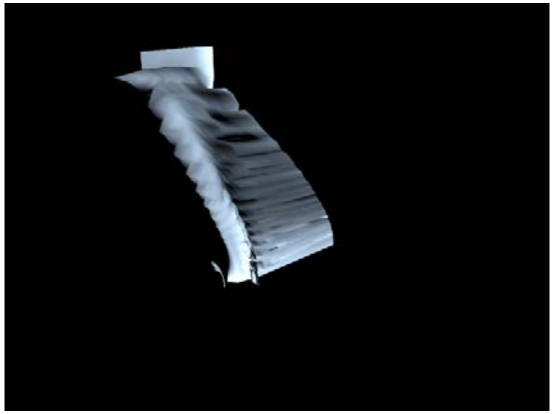
– average



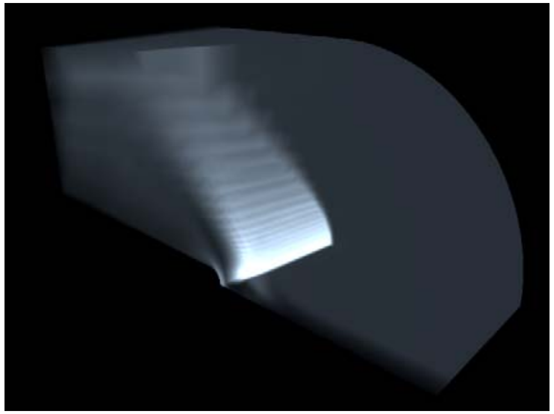
Average: Produces basically an X-ray picture

Volume Visualization Techniques

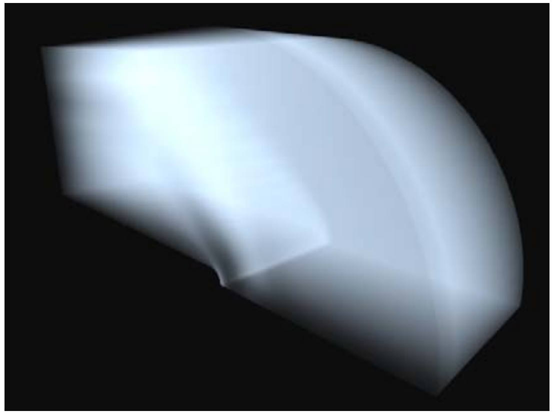
- **Rendering Volume Data**
  - Isosurface Rendering (implicit surface)
  - Maximum-Intensity-Projection
    - Render the larges volume value along a ray
  - Direct or Emission-Absorption Volume Rendering (x-ray)



Isosurface Rendering



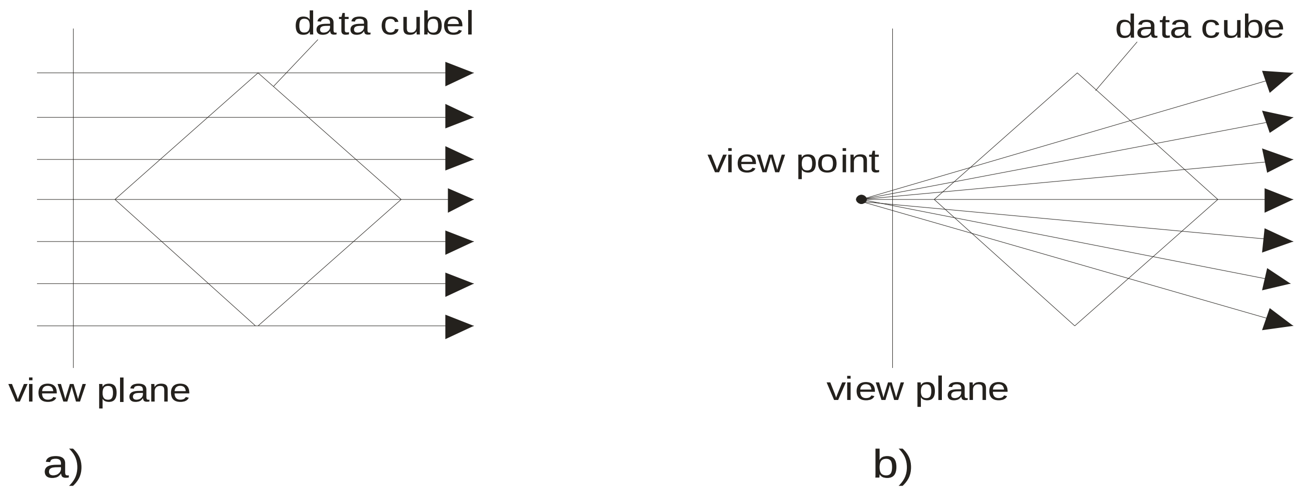
Maximum-Intensity-P.



E-A Volume Rendering

Cost Reduction through Transformation

use parallel- instead perspective projection



- Transform the data volume such that rays are parallel to coordinate axes.

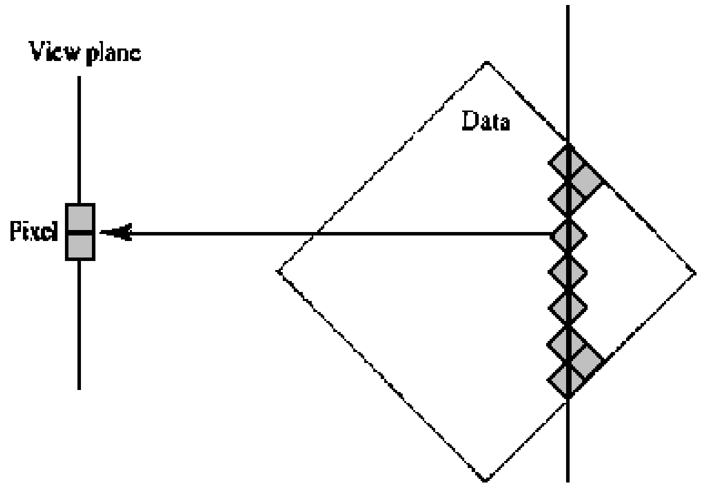
Projection

- Projection and rasterization of cells, Voxels, planes
  - plane composing
  - voxel projection
  - cell projection
  - shear warp

Volume Slicing

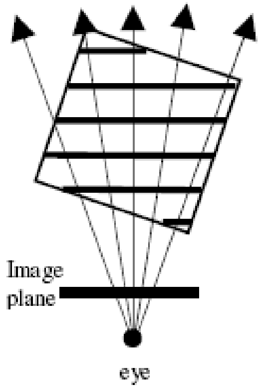
- **The plane composing (or “slicing”) method, divides the volume into slices.**
  During the rendering process, the slices are composes one over the other, producing the image.

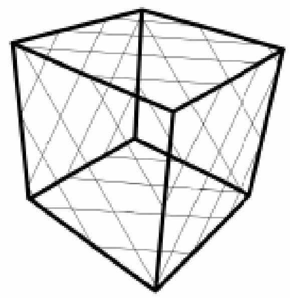
Basic Complexity = VolumeSize



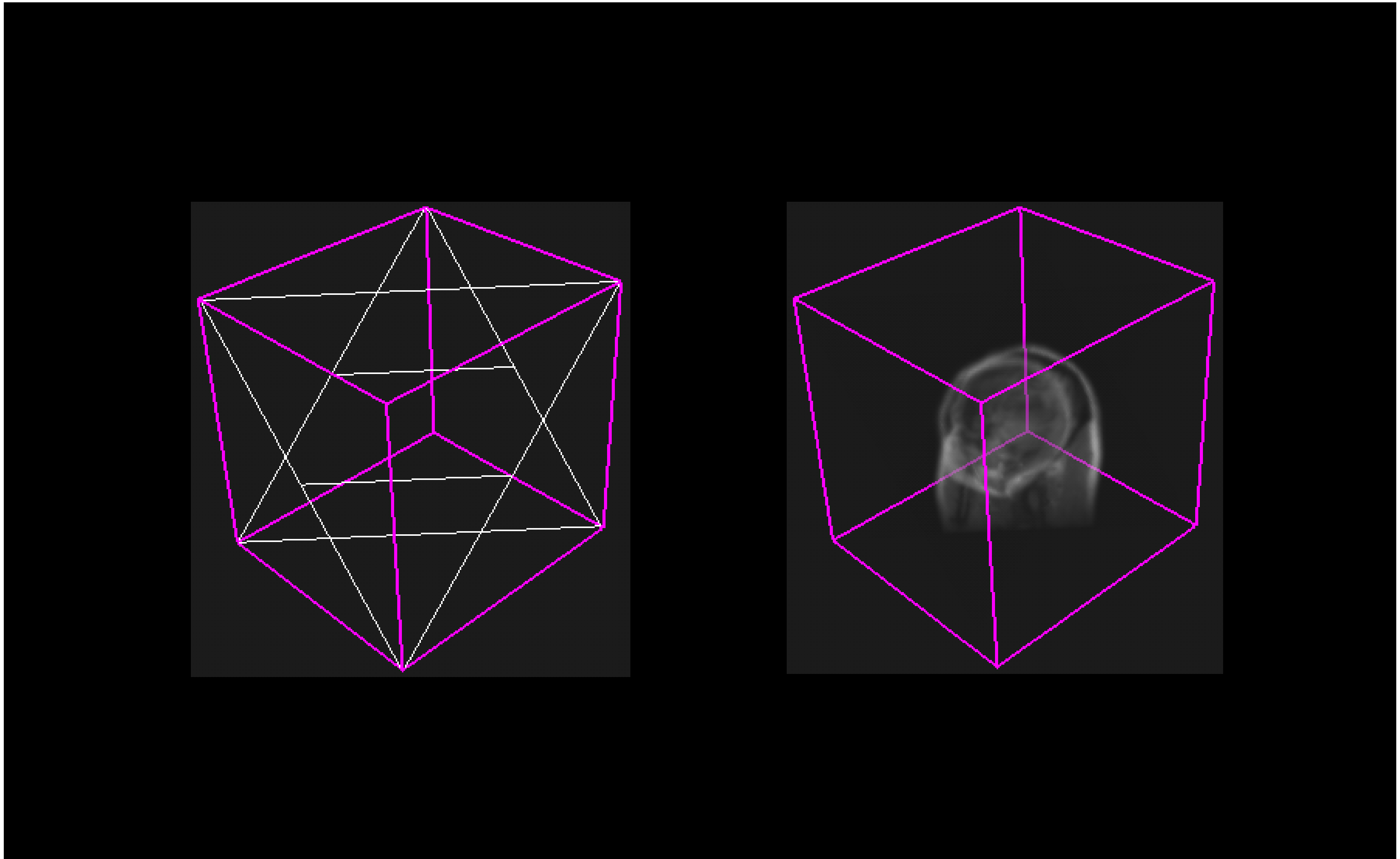
Volume Rendering on GPU

- **Volume Rendering with 3D-Textures**
  - Given volume data set as 3D texture
  - Slice bounding box of 3D texture with planes parallel to viewing plane
  - Render with back to front approach
    - With compositing set appropriately (does not need Alpha buffer)
    - FB_color = FB_color * (1-fragment_alpha) + fragment_color
- **Using 2D Texture**
  - Same technique but use 2D slices of of volume directly
  - Needs three copies (xy, xz, yz) to always use best orientation

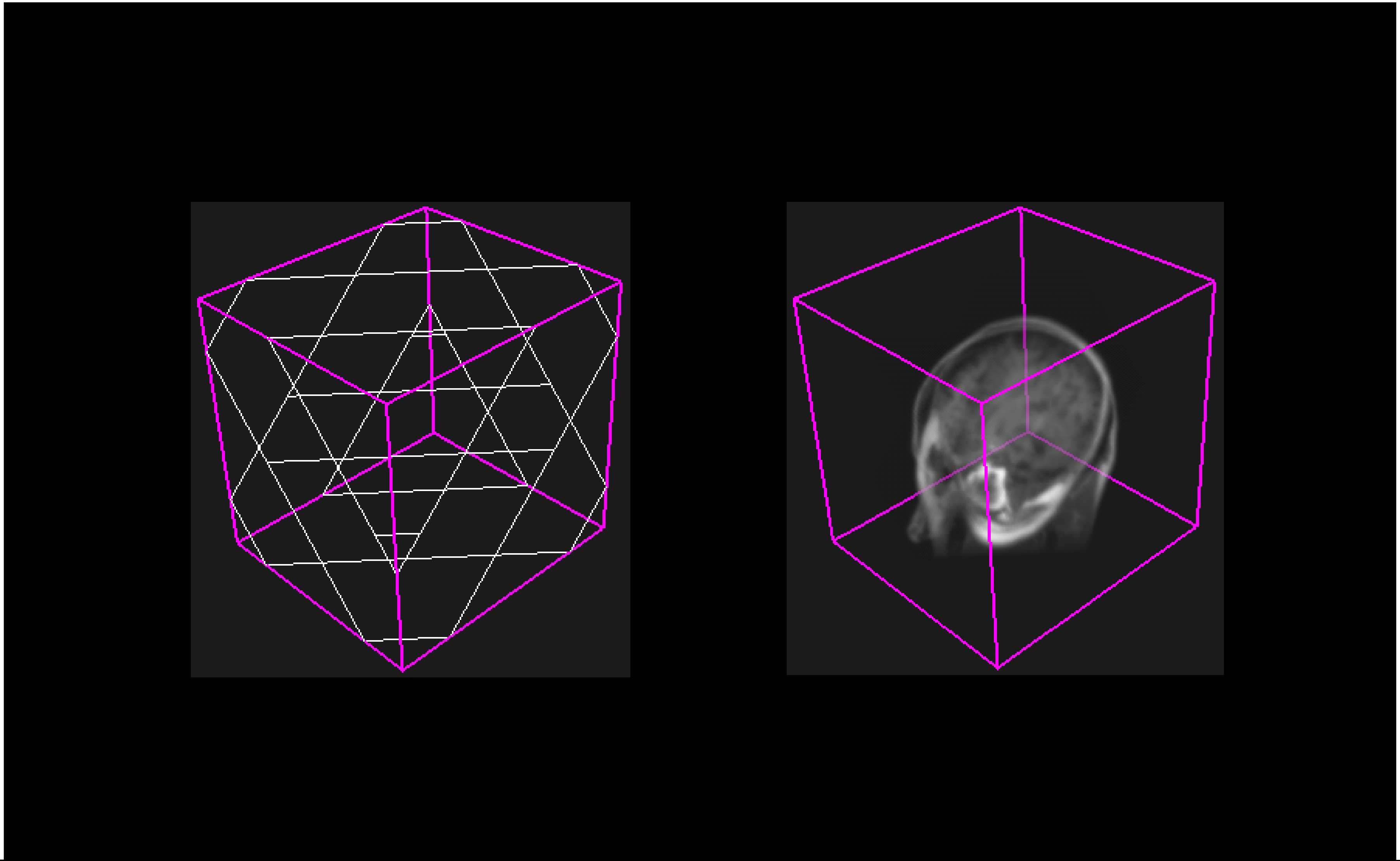




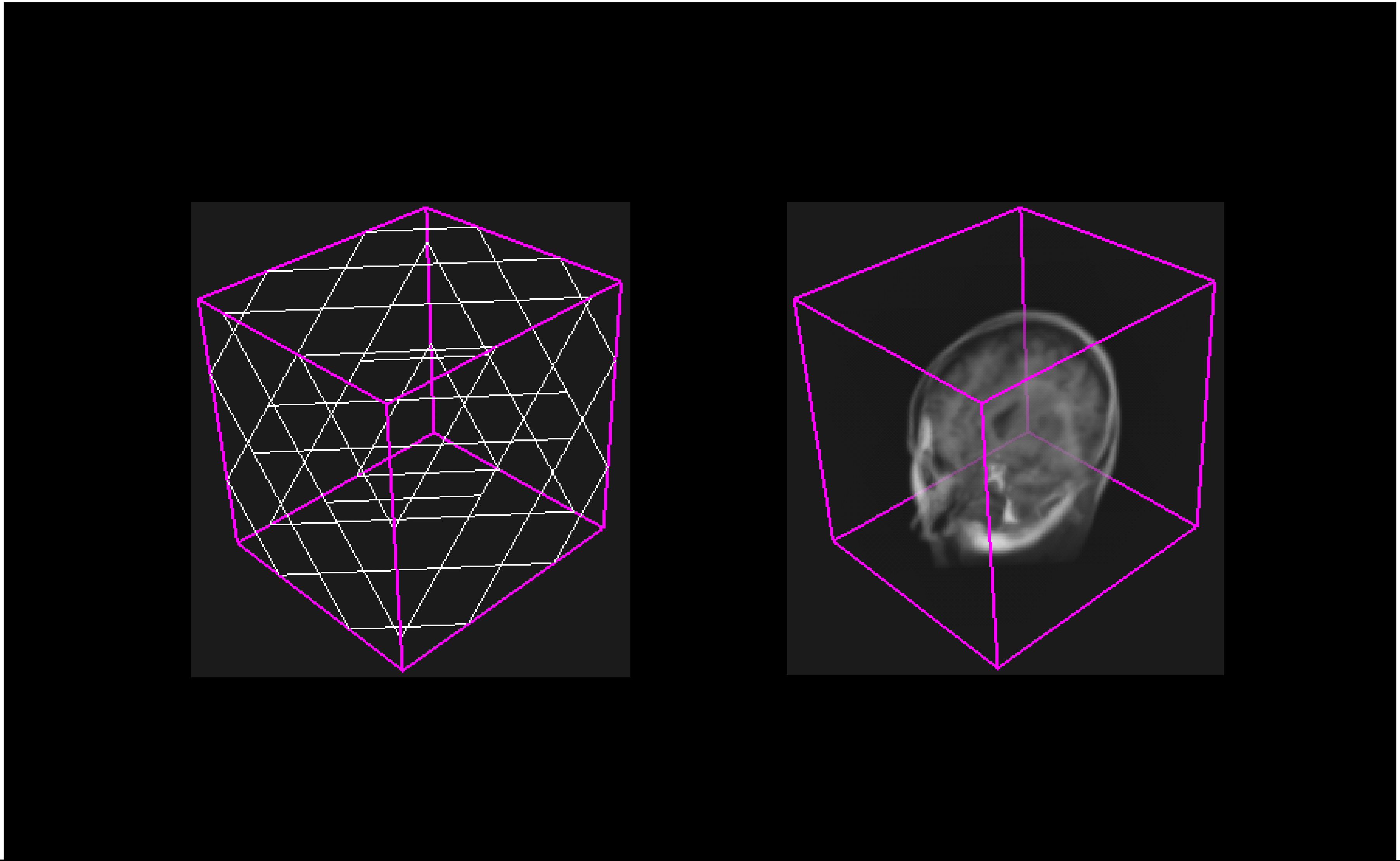
Volume Slicing



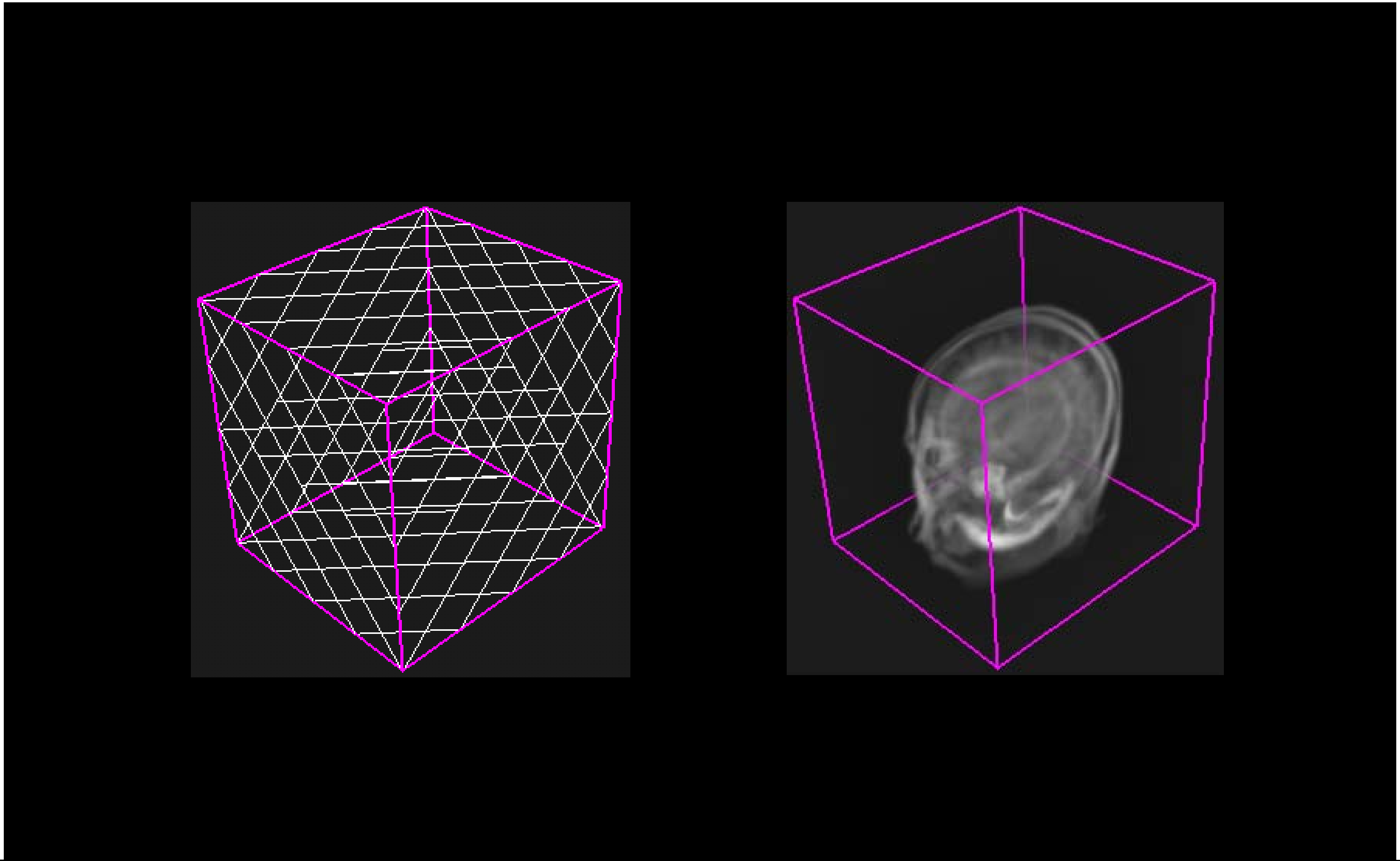
Volume Slicing



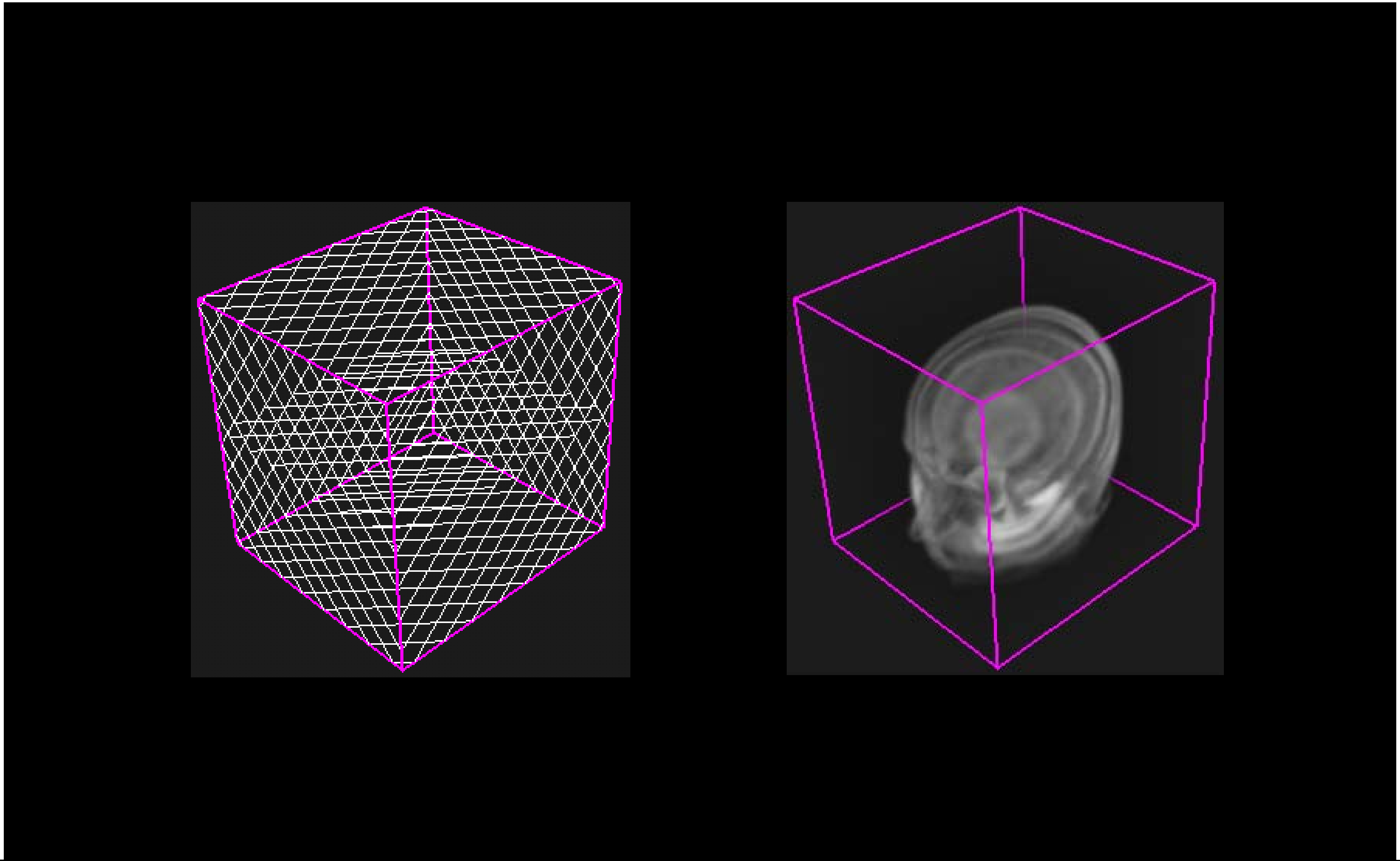
Volume Slicing



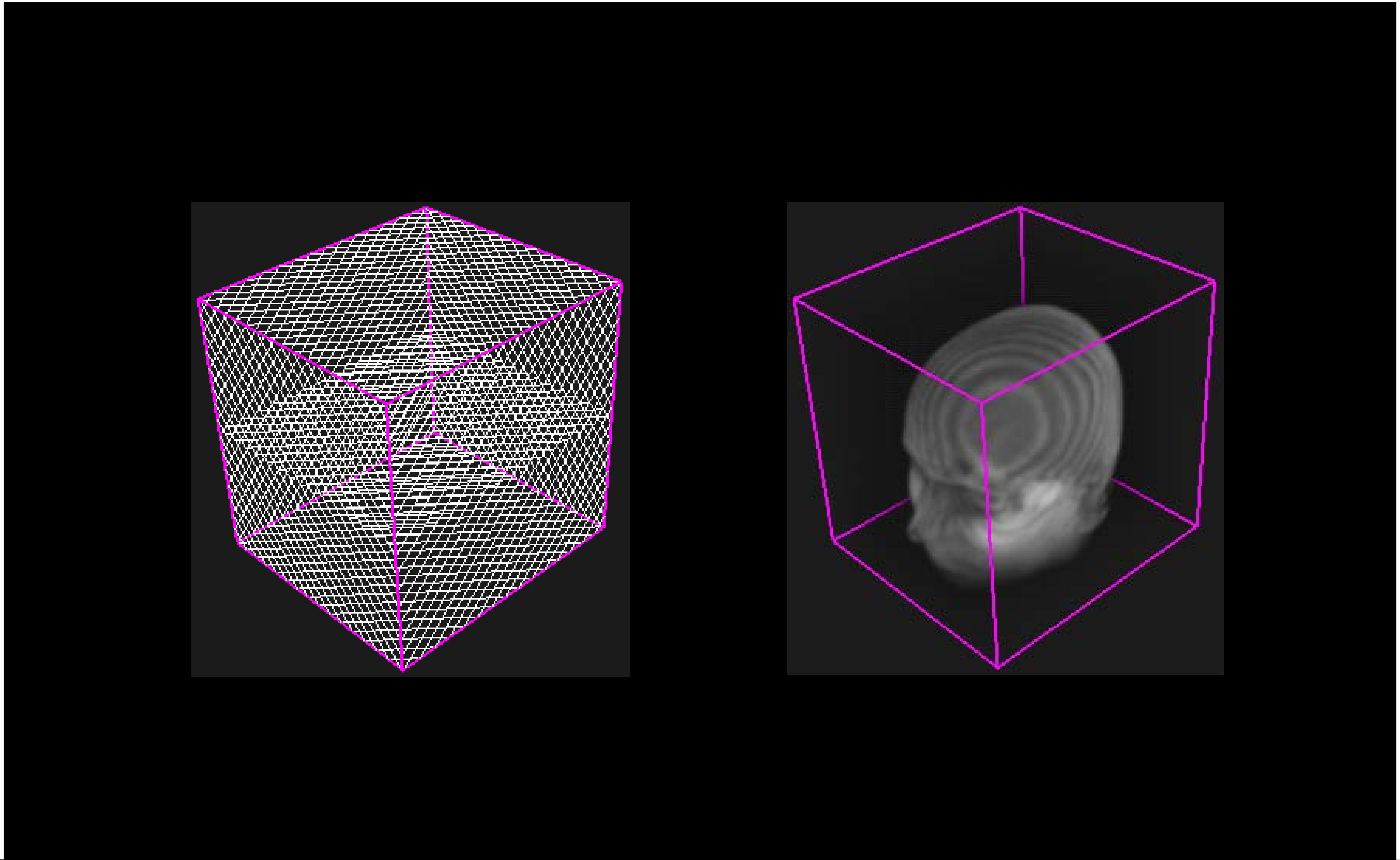
Volume Slicing



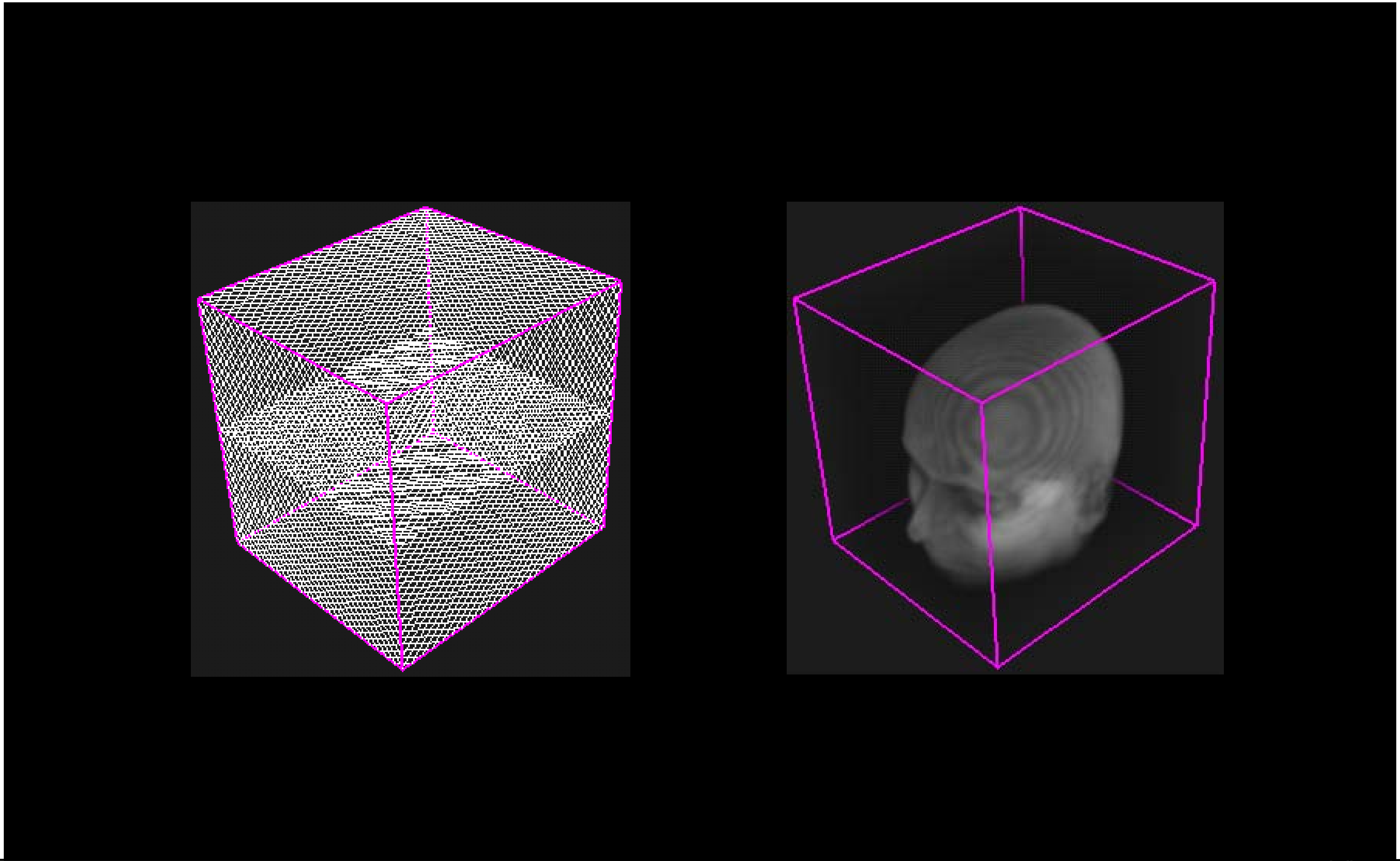
Volume Slicing



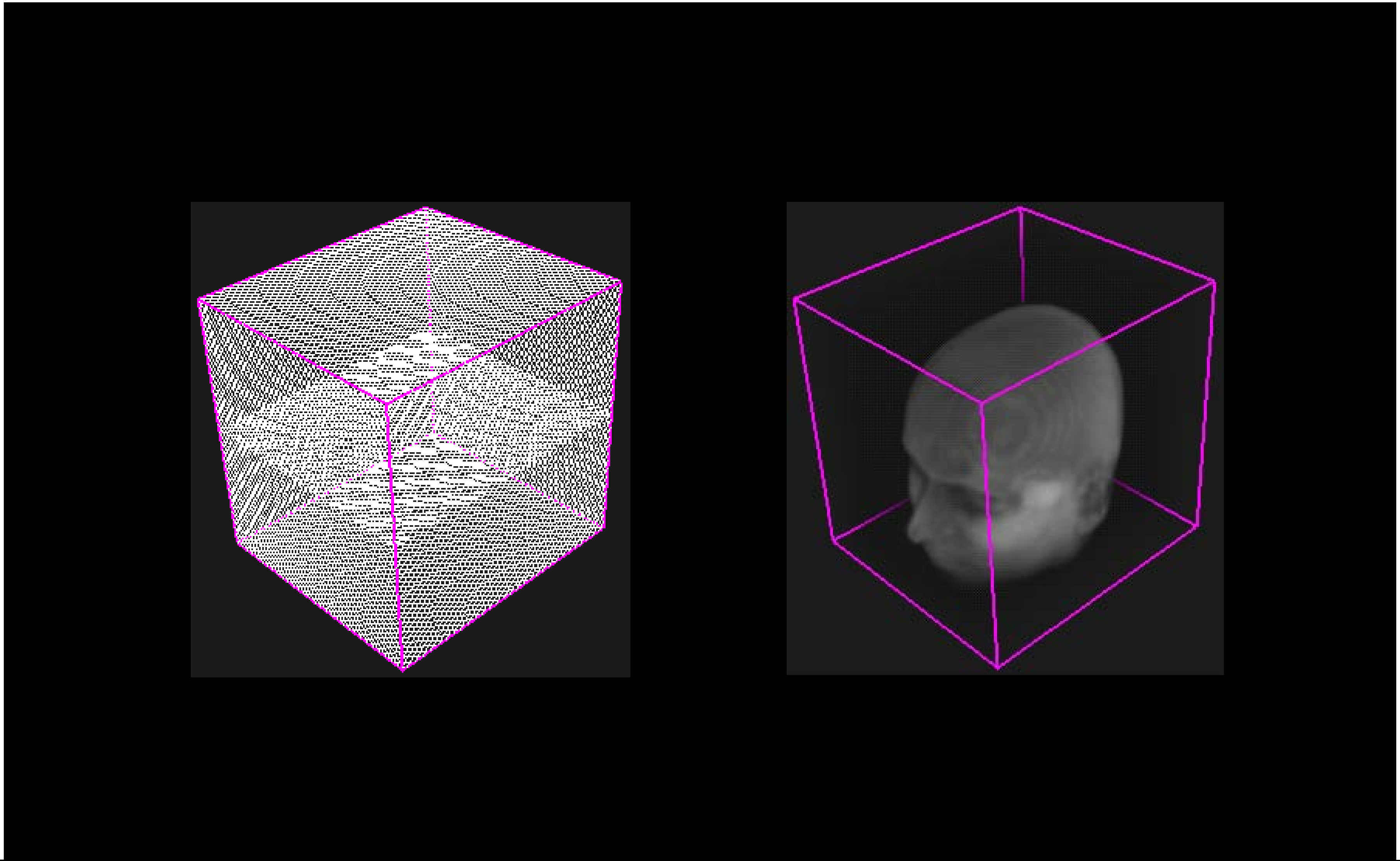
Volume Slicing



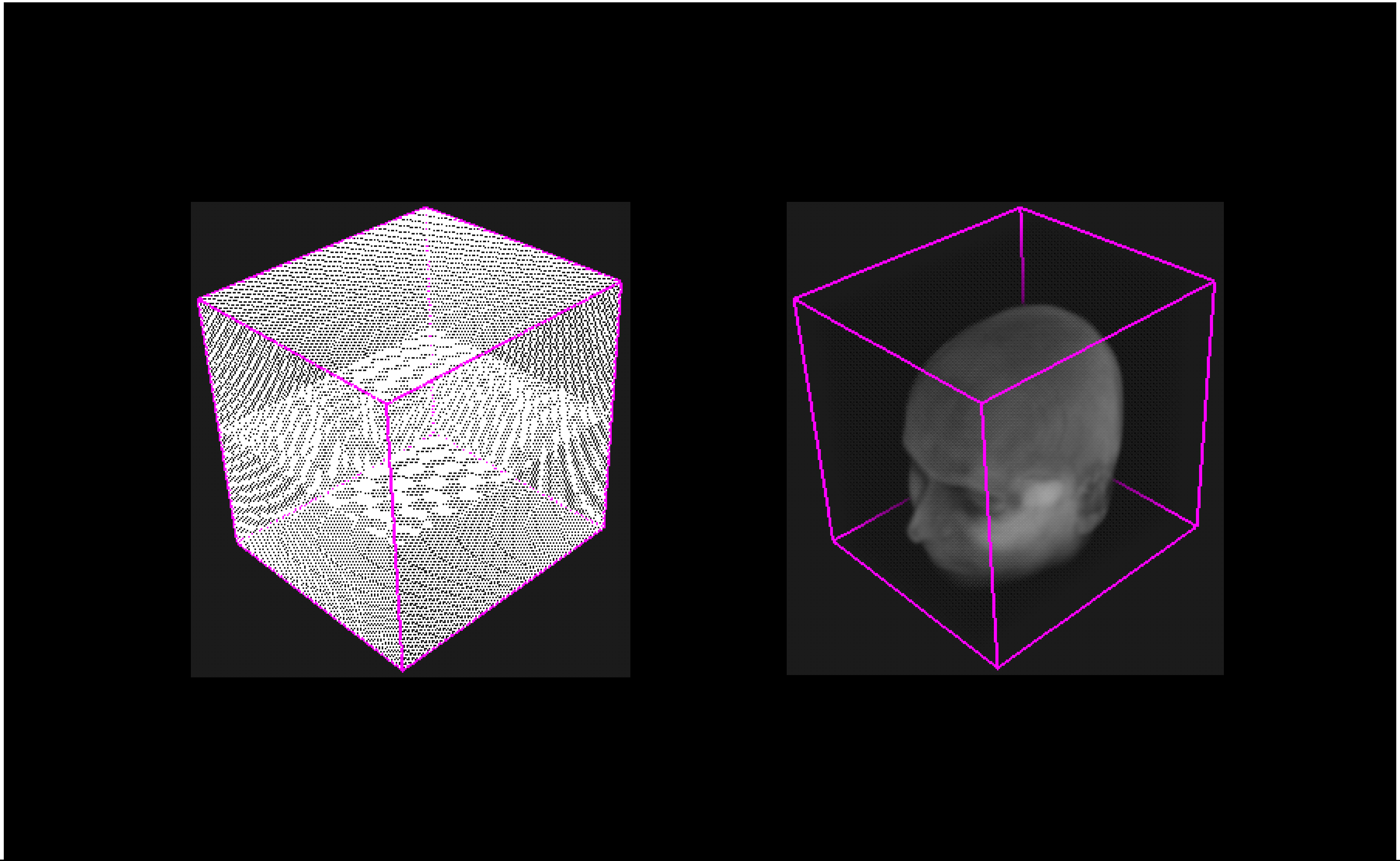
Volume Slicing



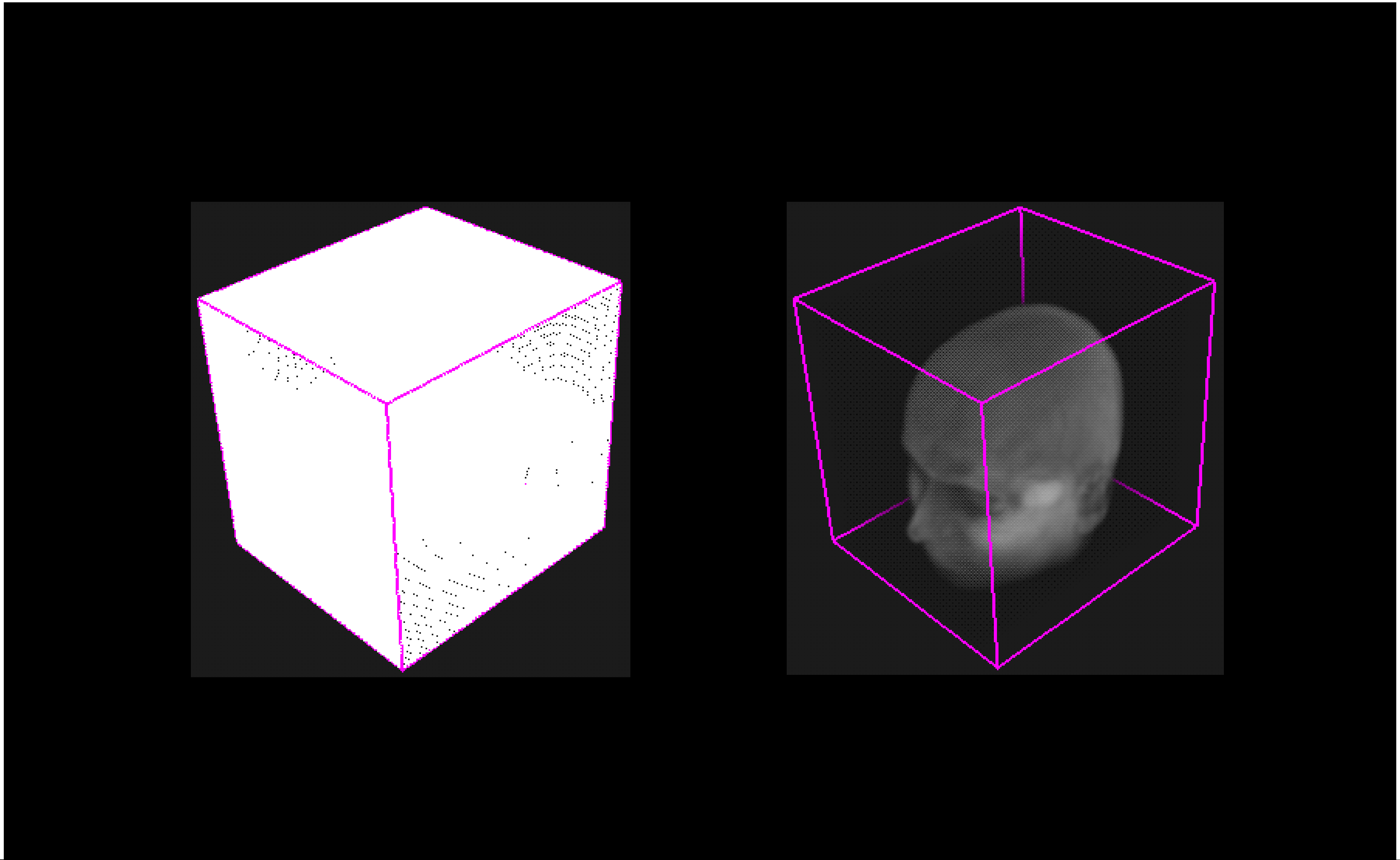
Volume Slicing



Volume Slicing



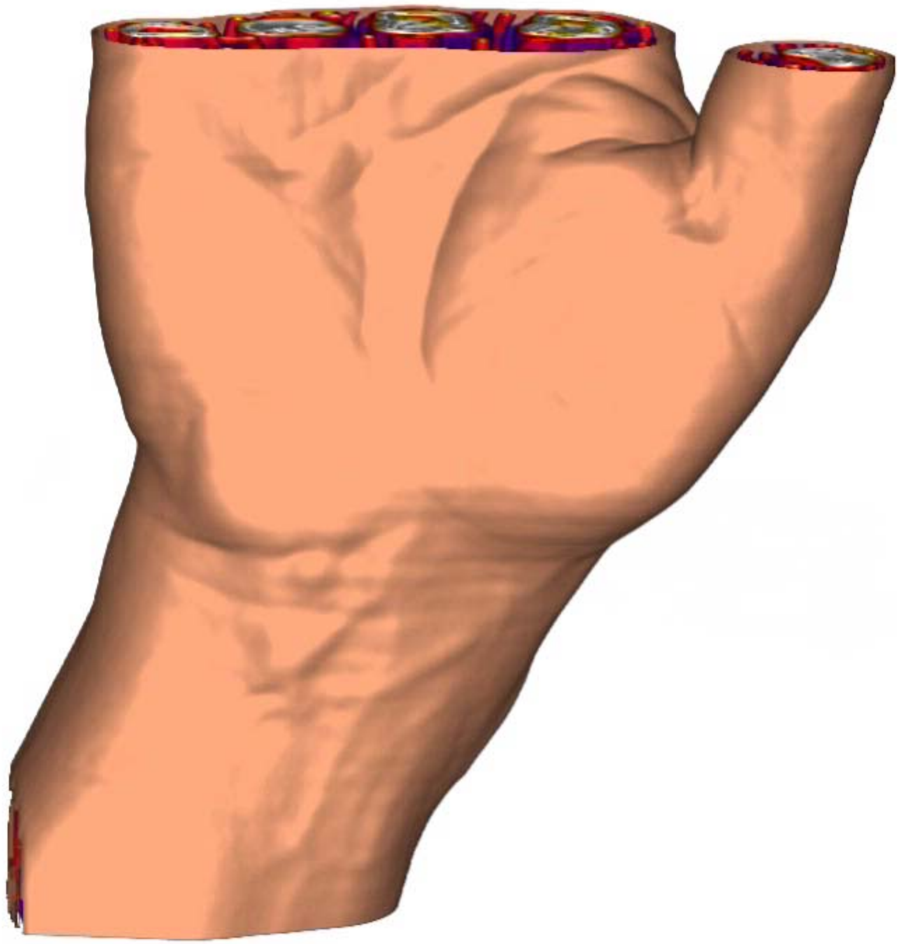
Volume Slicing



Volumes and Surfaces

- **Interactions**
  - Surface/Volume
    - Intersect with surfaces → ray segment
    - Perform volume rendering along segment
    - Add contribution from surface
    - Must handle surfaces within volumes correctly
  - Volume/Volume
    - Parallel traversal necessary if volumes overlap
      - Opacity combines from both volumes
- **Comparison**
  - Surfaces:
    - Complex traversal operations
    - Single intersection per ray → few complex shading operations
  - Volumes
    - Often simple traversal
    - Constantly shading but often simple shading algorithms

Context Aware Volume Rendering



[Bruckner & Groeller 2005]