

Information Visualization

W13: Scalar Data Visualization 2 - Volume Rendering

Graduation School of System Informatics

Department of Computational Science

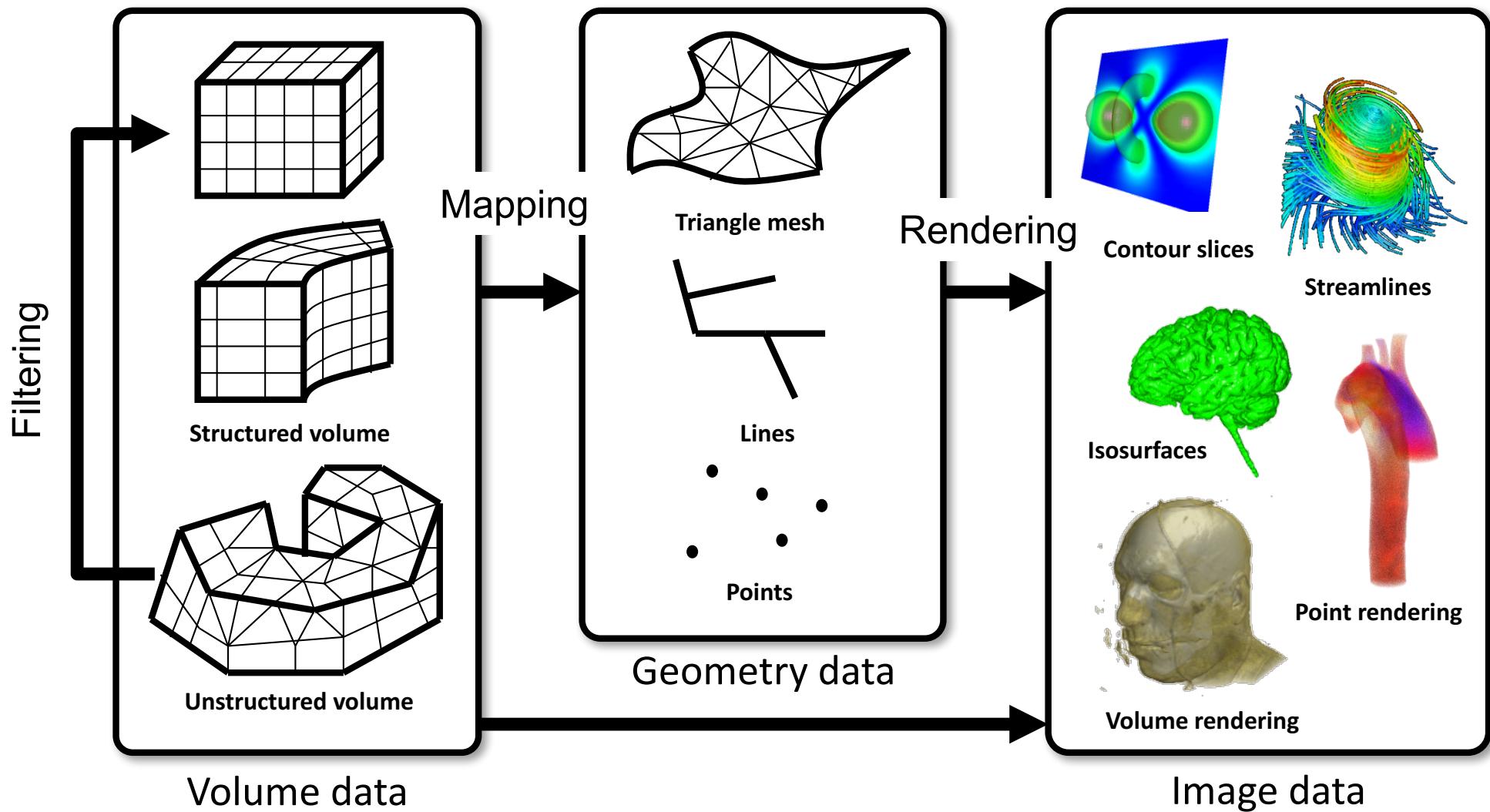
Naohisa Sakamoto, Akira Kageyama

May.30, 2017

Schedule

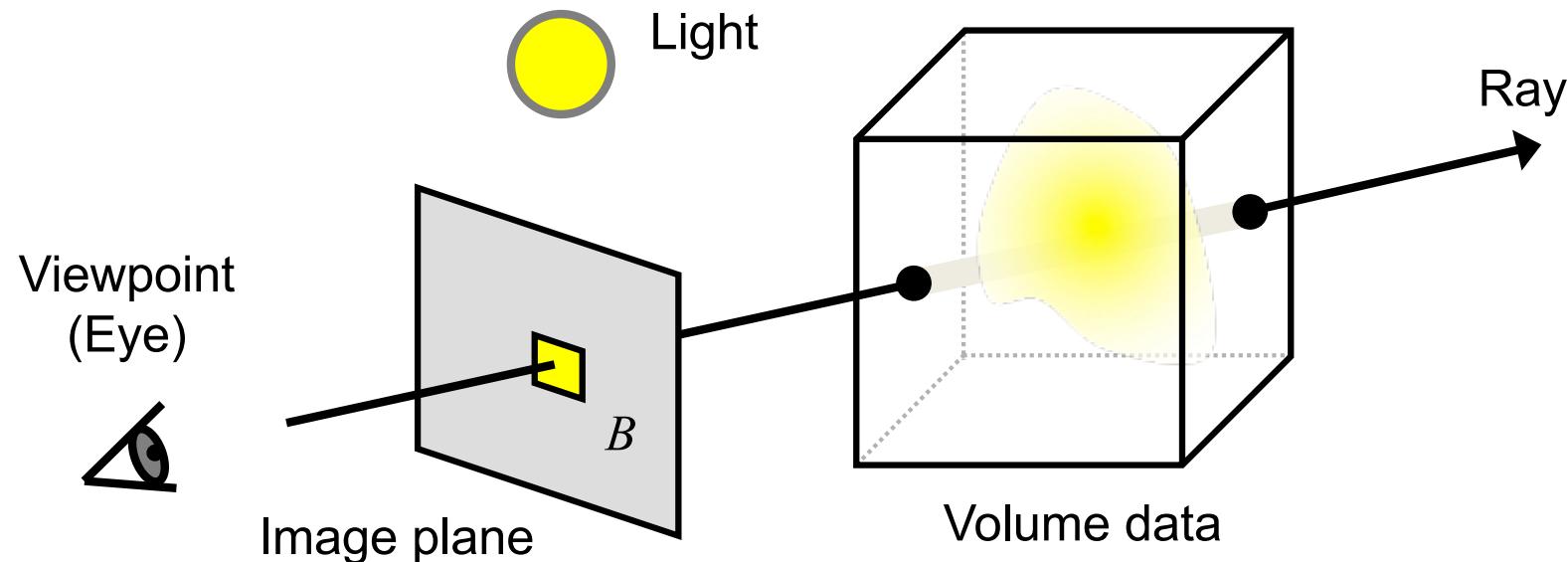
- W01 4/11 Guidance
- W02 4/12 Setup
- W03 4/18 Introduction to Data Visualization
- W04 4/19 CG Programming
- W05 4/25 Rendering Pipeline
- W06 4/26 Coordinate Systems and Transformations
- W07 5/09 Shading
- W08 5/10 Shader Programming
- W09 5/16 Visualization Pipeline
- W10 5/17 Data Model and Transfer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

Scientific Visualization



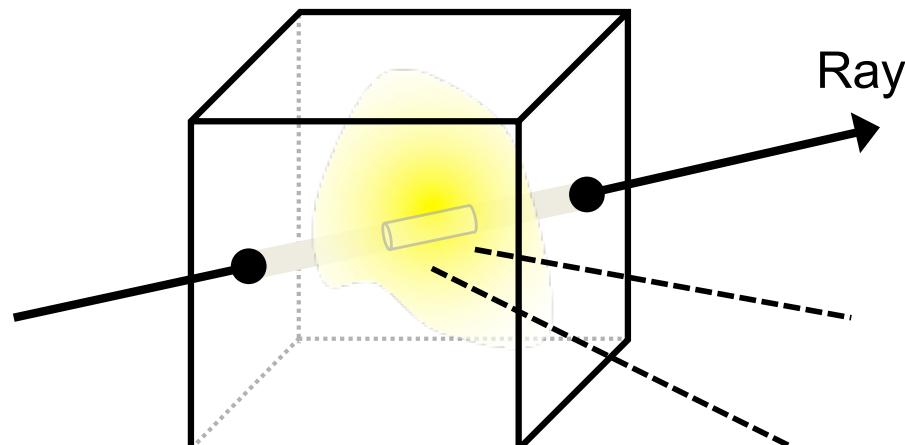
What is volume rendering?

- Volume rendering is a technique to generate images from volume datasets based on the density emitter model.

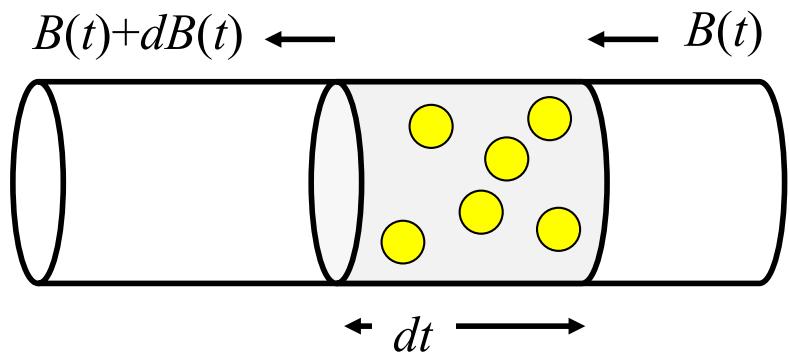


What is volume rendering?

- Density emitter model [Sabella, 1988]



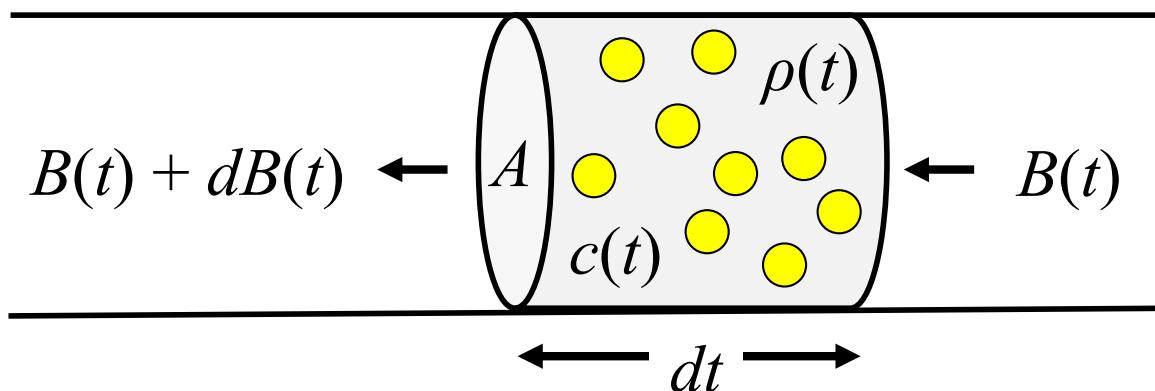
- Emissive and opaque particles are assumed
- Not particle but their densities are modeled



$$B = \int_{t_n}^{t_0} c(t) \times \pi r^2 \rho(t) \times \exp \left(\int_t^{t_0} \pi r^2 \rho(\lambda) d\lambda \right) dt$$

Density Emitter Model

- Brightness equation



$B(t)$: Brightness
 $\rho(t)$: Particle density
 $c(t)$: Luminosity
 r : Particle radius

$$db(t)A = -\text{absorbed} + \text{emitted}$$

$$= (-B(t) + c(t)) \times \pi r^2 \rho(t) A dt$$

$$\frac{dB(t)}{dt} = (-B(t) + c(t)) \times \pi r^2 \rho(t)$$

Brightness Equation

- Multiplying both sides by a exponential term.

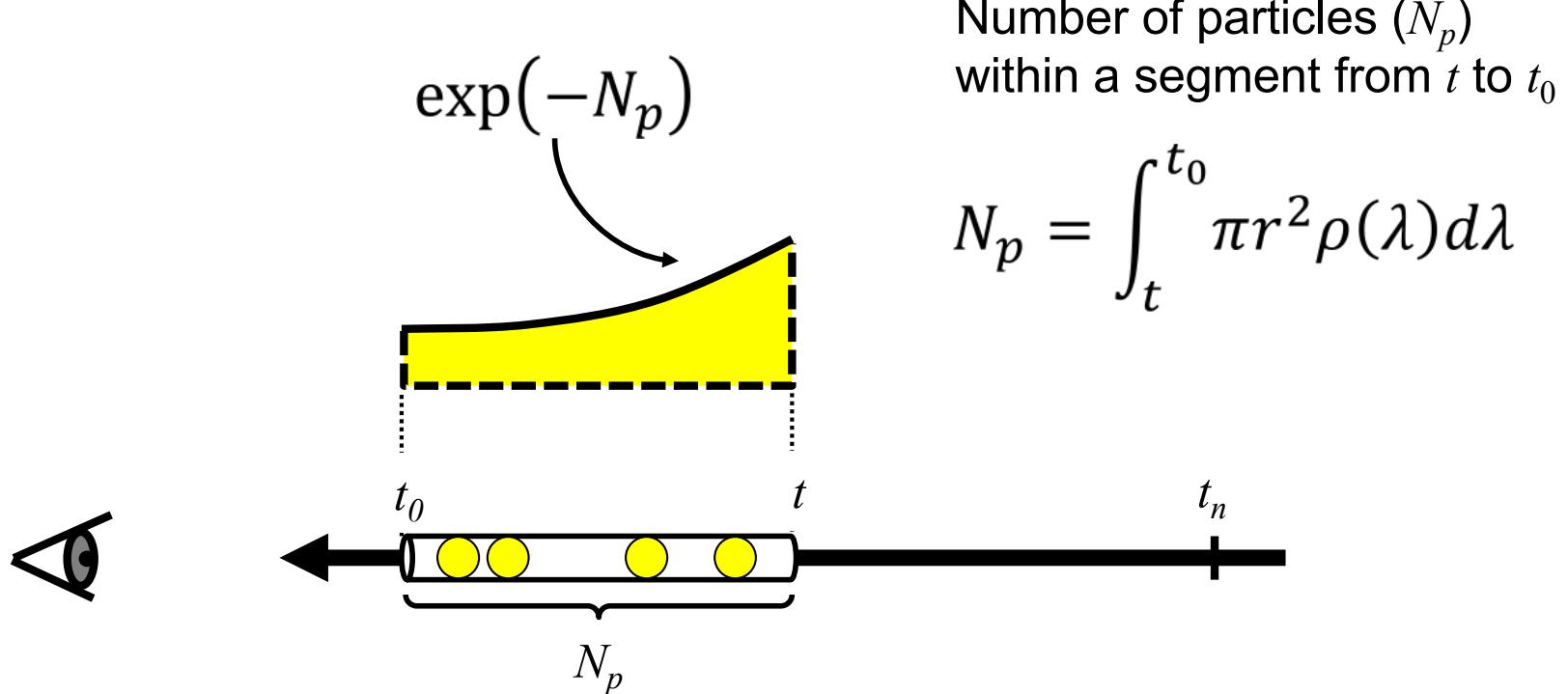
$$\frac{dB(t)}{dt} = \{-B(t) + c(t)\} \times \pi r^2 \rho(t)$$

$$\exp\left(\int_{t_n}^t \pi r^2 \rho(u) du\right) \frac{dB(t)}{dt} = \exp\left(\int_{t_n}^t \pi r^2 \rho(u) du\right) \{-B(t) + c(t)\} \times \pi r^2 \rho(t)$$
$$\Rightarrow B = B(t_0) = \int_{t_n}^{t_0} c(t) \times \pi r^2 \rho(t) \times \exp\left(-\int_t^{t_0} \pi r^2 \rho(\lambda) d\lambda\right) dt$$



Brightness Equation

$$B = \int_{t_n}^{t_0} c(t) \times \pi r^2 \rho(t) \times \exp\left(-\int_t^{t_0} \pi r^2 \rho(\lambda) d\lambda\right) dt$$



Brightness Equation

- Poisson distribution
 - In the density emitter model, the Poisson distribution has been assumed for the number of particles in a segment.
 - If the expected number of particles in the segment is N_p , then the probability P that there are k particles is represented as follows:

$$P(N = k) = \frac{\exp(-N_p)N_p^k}{k!}$$

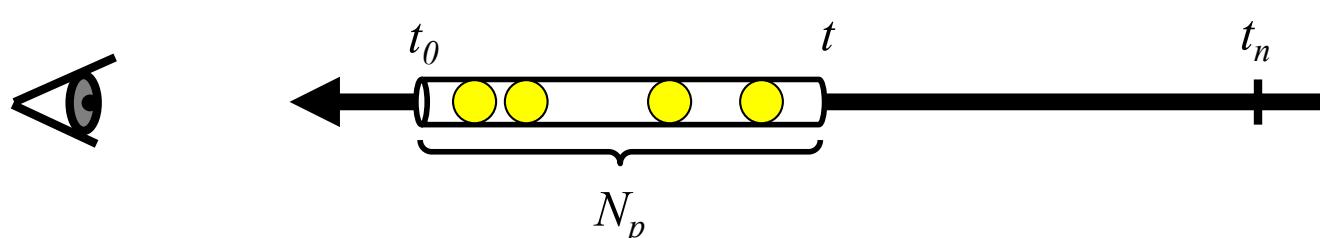
Brightness Equation

- Definition of opacity
 - The probability that there is no particles is defined as transparency

$$0 \leq P(N = 0) = \exp(-N_p) \leq 1$$

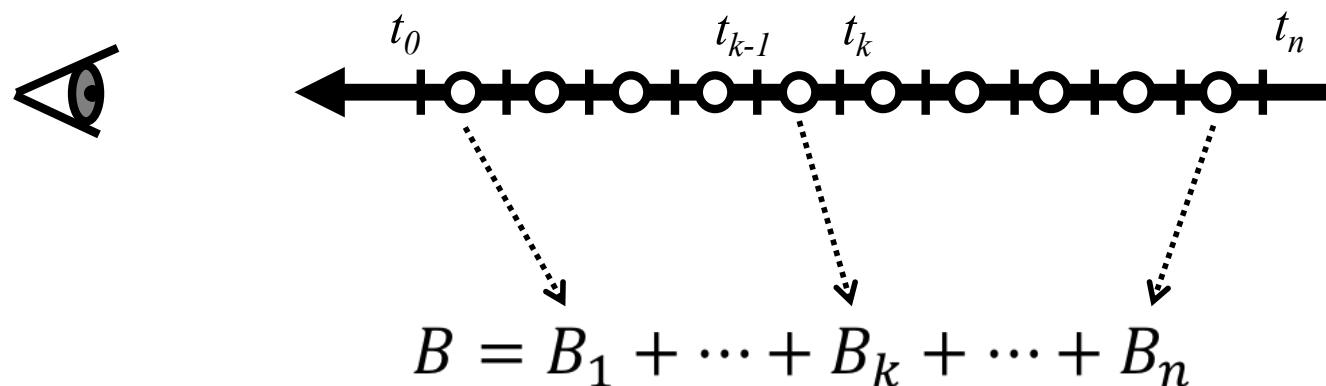
- The probability that there are more than a particle is defined as opacity

$$0 \leq 1 - P(N = 0) = 1 - \exp(-N_p) \leq 1$$



Brightness Equation

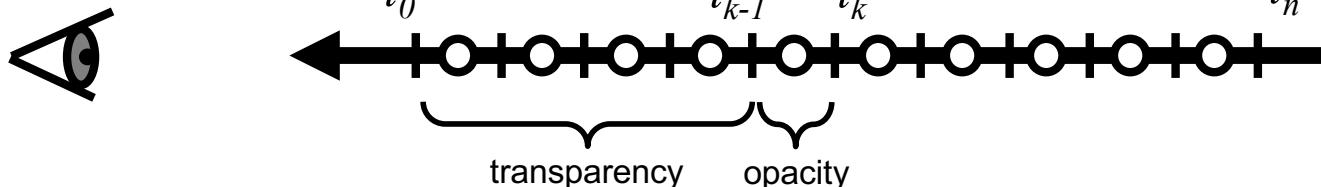
- Discretization
 - Brightness equation is numerically calculated by subdividing a viewing ray into n ray segments.



Brightness Equation

- Numerical Solution

$$\begin{aligned} B_k &= \int_{t_k}^{t_{k-1}} c_k \times \pi r^2 \rho(t) \times \exp\left(-\int_t^{t_0} \pi r^2 \rho(\lambda) d\lambda\right) dt \\ &= \exp\left(-\int_{t_{k-1}}^{t_0} \pi r^2 \rho(\lambda) d\lambda\right) \int_{t_k}^{t_{k-1}} c_k \times \pi r^2 \rho(t) \times \exp\left(-\int_t^{t_{k-1}} \pi r^2 \rho(\lambda) d\lambda\right) dt \\ &= \prod_{i=0}^{k-2} \underbrace{\exp\left(-\int_{t_{i+1}}^{t_i} \pi r^2 \rho(\lambda) d\lambda\right)}_{\text{transparency}} \times c_k \times \underbrace{\left(1 - \exp\left(-\int_{t_k}^{t_{k-1}} \pi r^2 \rho(\lambda) d\lambda\right)\right)}_{\text{opacity}} \end{aligned}$$

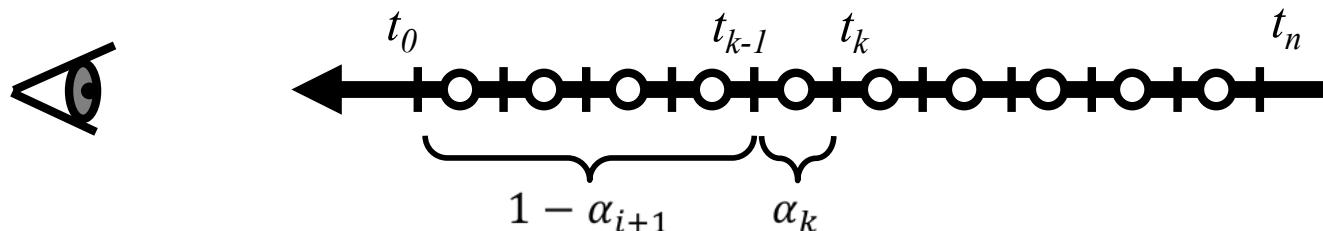


Brightness Equation

- Numerical Solution

$$B_k = \prod_{i=0}^{k-2} \underbrace{\exp\left(-\int_{t_{i+1}}^{t_i} \pi r^2 \rho(\lambda) d\lambda\right)}_{\text{transparency} = 1 - \alpha_{i+1}} \times c_k \times \left(1 - \exp\left(-\int_{t_k}^{t_{k-1}} \pi r^2 \rho(\lambda) d\lambda\right)\right) \underbrace{\quad}_{\text{opacity} = \alpha_k}$$

$$= c_k \alpha_k \prod_{i=0}^{k-2} (1 - \alpha_{i+1}) = c_k \alpha_k \prod_{i=1}^{k-1} (1 - \alpha_i)$$



Brightness Equation

- Opacity correction
 - Opacity along the k-th segment

$$\alpha_k = 1 - \exp\left(- \int_{t_k}^{t_{k-1}} \pi r^2 \rho(\lambda) d\lambda\right)$$
$$\approx 1 - \exp(-\pi r^2 \rho_k \times \Delta t)$$

where, Δt is the length of the segment and ρ_k is the density at its center.

- If an actual length $\Delta t'$ is different from the Δt , the opacity is corrected as follows:

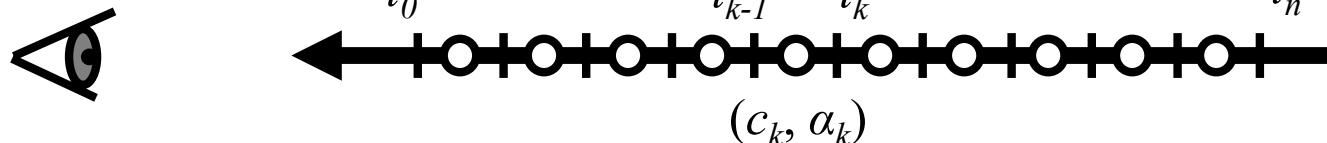
$$\alpha'_k = 1 - (1 - \alpha_k)^{\frac{\Delta t'}{\Delta t}}$$

Brightness Equation

- Discretized brightness equation

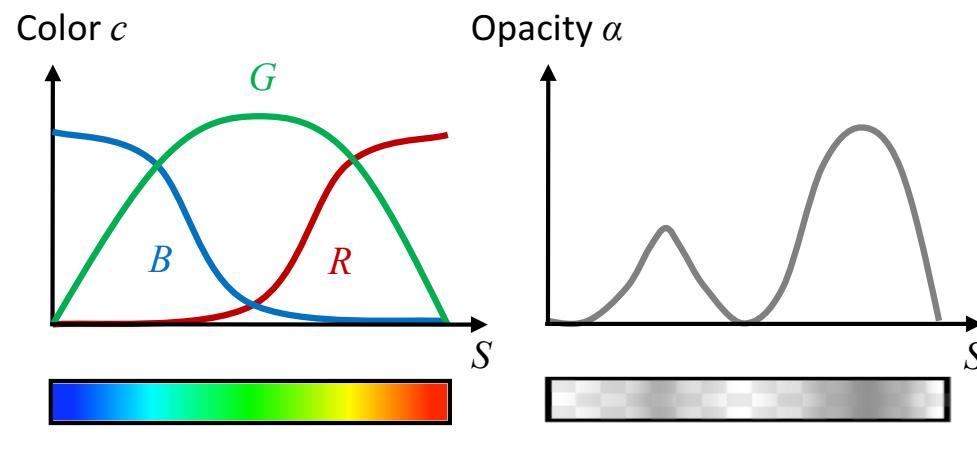
$$B = B_1 + \cdots + B_k + \cdots + B_n$$

$$= \sum_{j=1}^n c_j \alpha_j \prod_{i=1}^{j-1} (1 - \alpha_i)$$



Brightness Equation

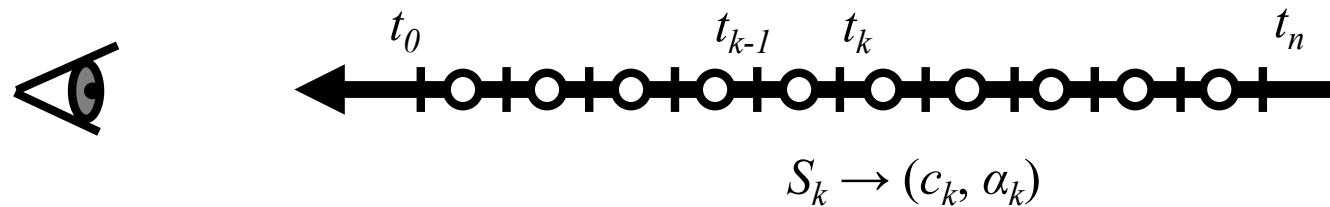
- Transfer function
 - In each segment, a scalar value S is mapped to color c and opacity α with transfer function.



$$S_k \rightarrow c_k = (R_k, G_k, B_k)$$

$$S_k \rightarrow \alpha_k$$

$$B = \sum_{j=1}^n c_j \alpha_j \prod_{i=1}^{j-1} (1 - \alpha_i)$$

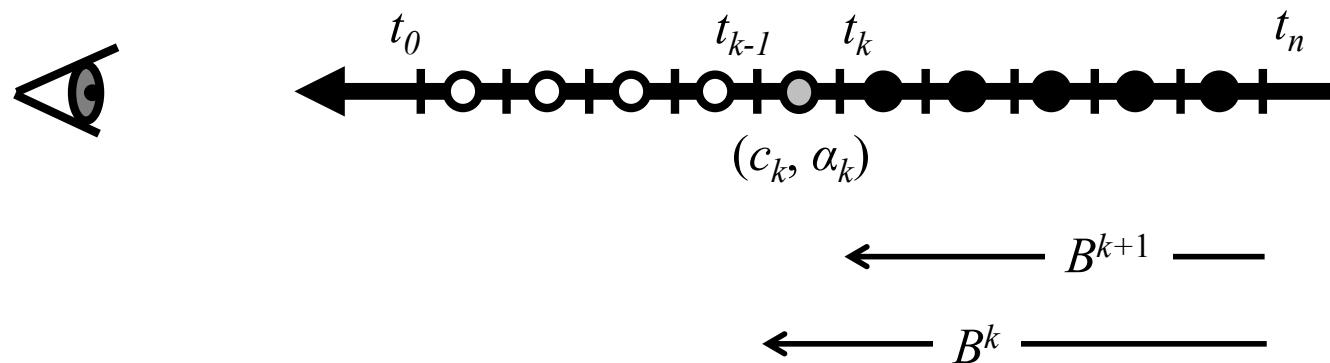


Brightness Equation

- Recurrence formula
 - Back-to-front compositing

$$B^k = c_k \alpha_k + (1 - \alpha_k) B^{k+1}$$

where c_k and α_k are the color and opacity for the k -th segment, and B_k is the accumulated brightness (color) from the back of the volume.



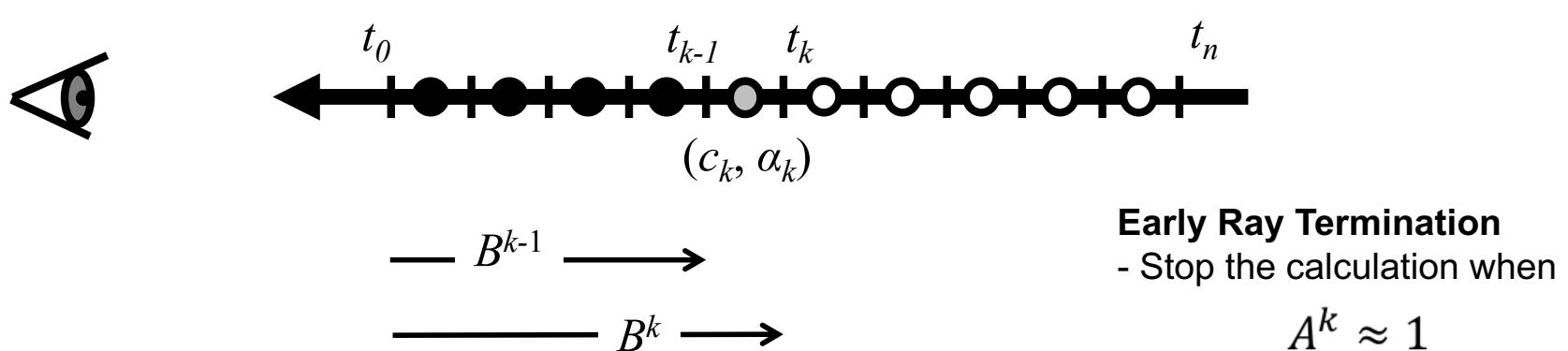
Brightness Equation

- Recurrence formula
 - Front-to-back compositing

$$B^k = c_k \alpha_k (1 - A^{k-1}) + B^{k-1}$$

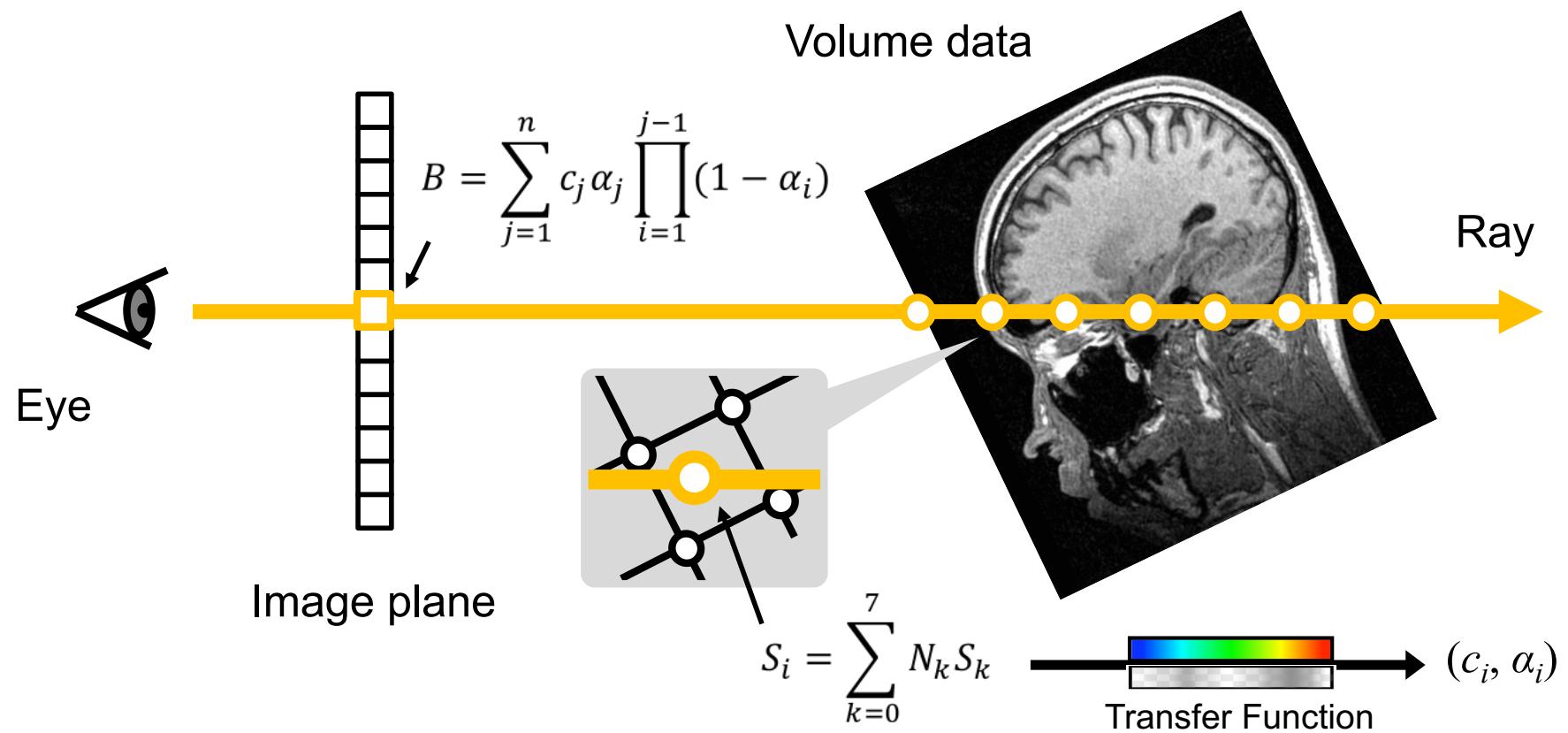
$$A^k = \alpha_k (1 - A^{k-1}) + A^{k-1}$$

where B_k and A_k are the accumulated brightness (color) and opacity from the front of the volume.



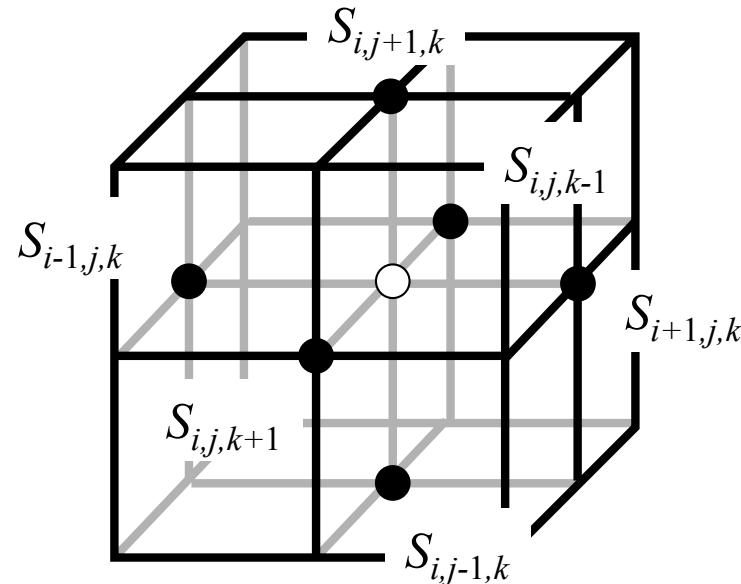
Volume Rendering

- Calculate the color and opacity for each pixel based on the brightness equation



Volume Rendering

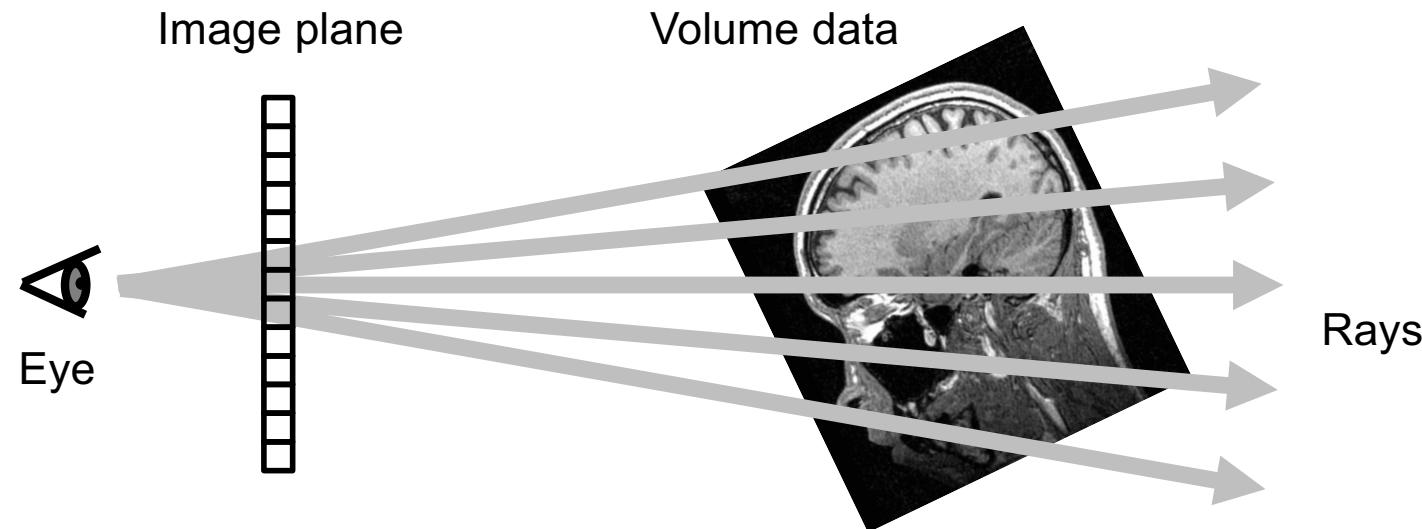
- Normal vector \mathbf{N} at the grid point can be calculated as scalar gradient.



$$\begin{aligned}\mathbf{N} = \nabla S &= \begin{pmatrix} \partial S / \partial x \\ \partial S / \partial y \\ \partial S / \partial z \end{pmatrix} \\ &= \begin{pmatrix} S_{i+1,j,k} - S_{i-1,j,k} / \Delta x \\ S_{i,j+1,k} - S_{i,j-1,k} / \Delta y \\ S_{i,j,k+1} - S_{i,j,k-1} / \Delta z \end{pmatrix}\end{aligned}$$

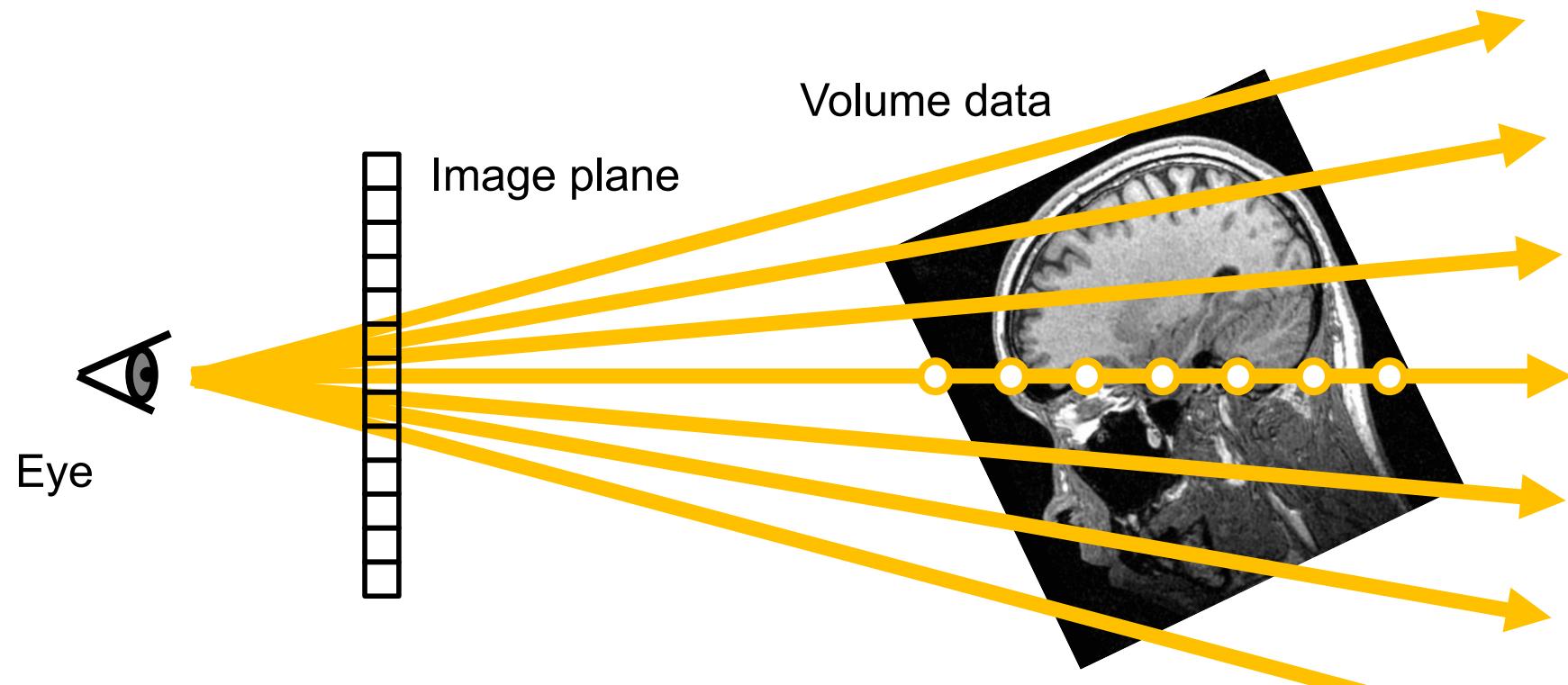
Volume Rendering

- Two approaches of volume rendering techniques
 - Image-order approach
 - Object-order approach



Volume Rendering

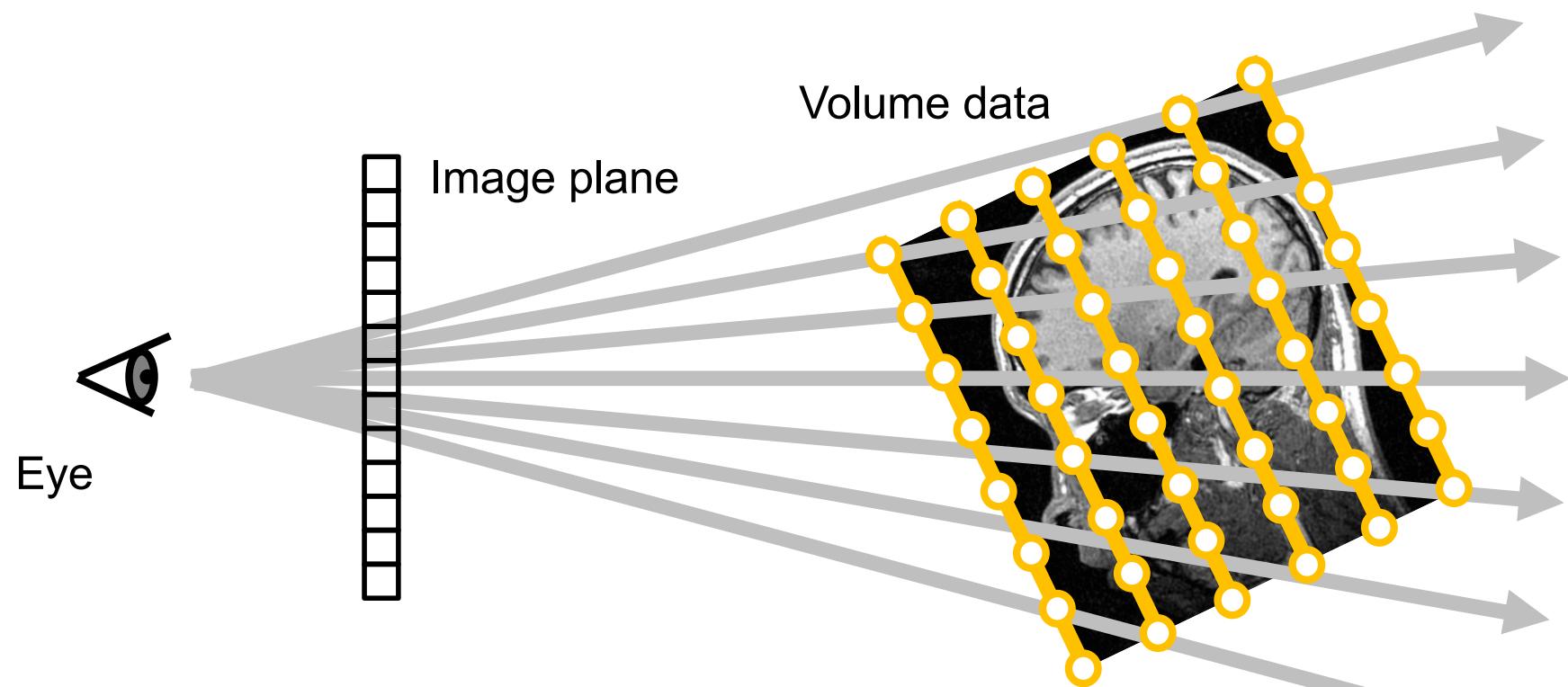
- Image-order approach



```
for each pixel  
{  
    // Calculation  
}
```

Volume Rendering

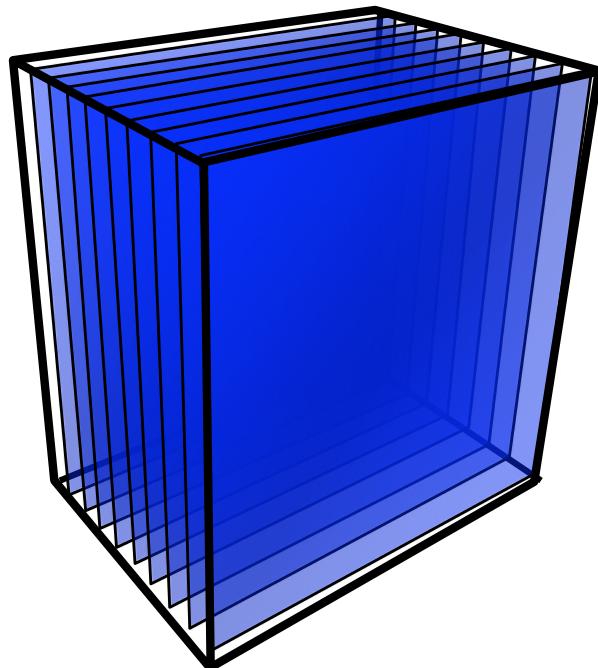
- Object-order approach



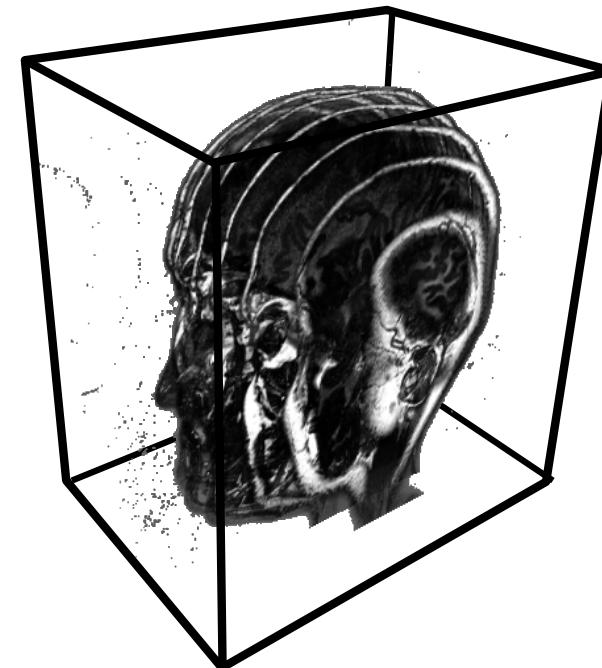
```
for each slice
{
    // Calculation
}
```

Implementation

- Texture-based approach
 - Object-order approach



Proxy Geometry

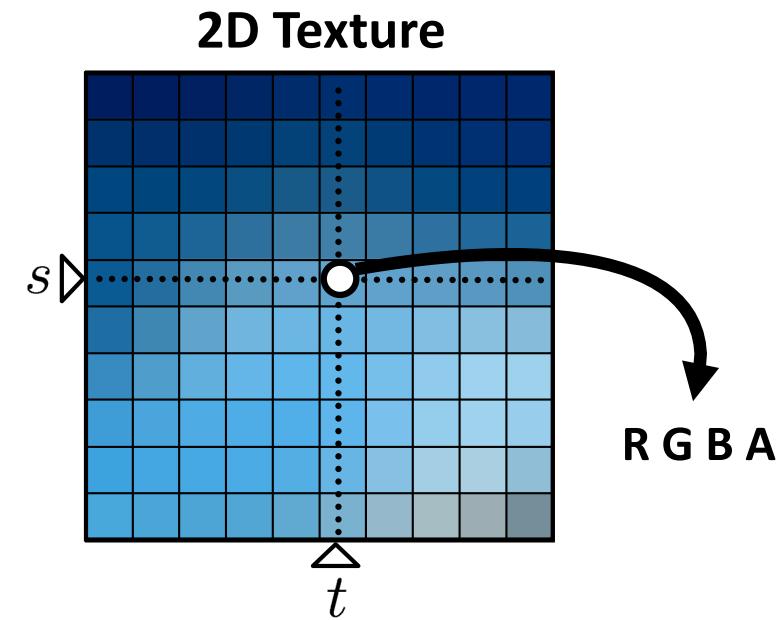
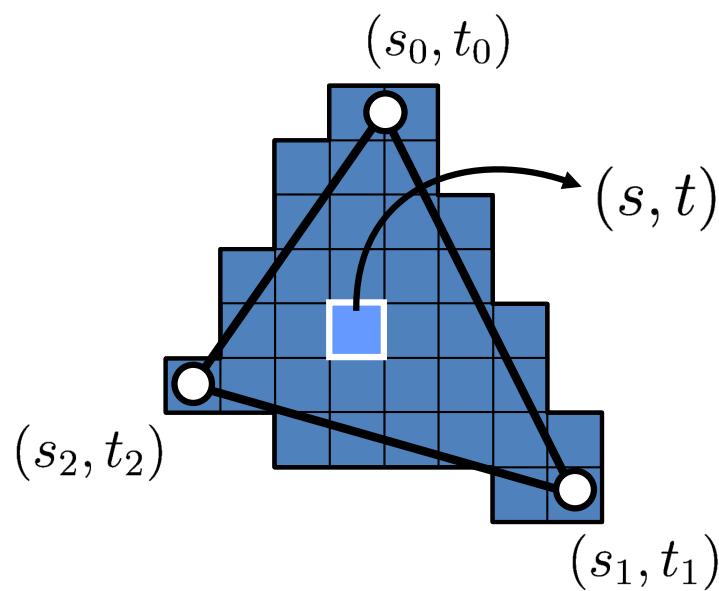


2D Textures

Implementation

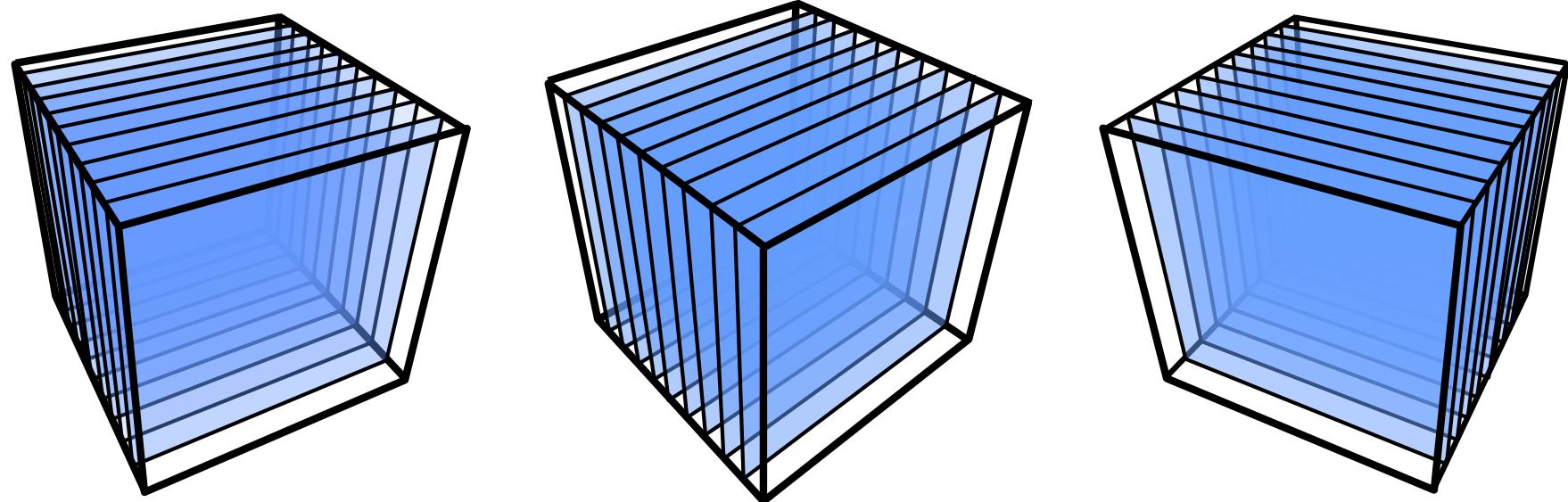
- **Texture**

- 1D Texture = 1D array data structure, linear interpolation
- 2D Texture = 2D array data structure, bi-linear interpolation
- 3D Texture = 3D array data structure, tri-linear interpolation



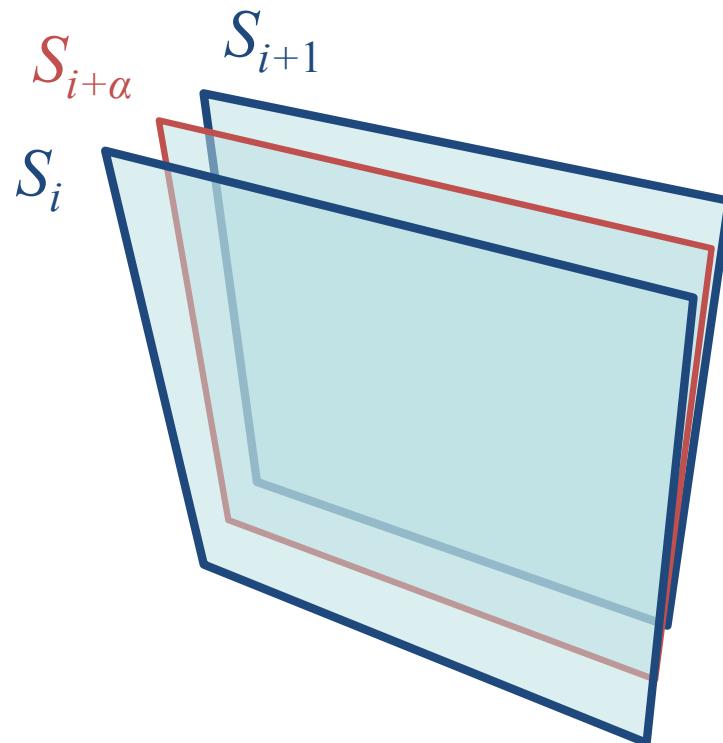
Object-order Approach

- Axis-aligned slices
 - Volume data is stored on GPU as a stack of 2D textures.
 - Three stacks of 2D textures for x,y,z principle directions.



Object-order Approach

- Axis-aligned slices
 - 2D Multi-Textures

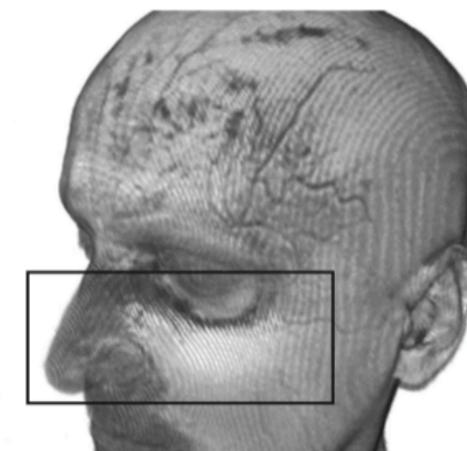
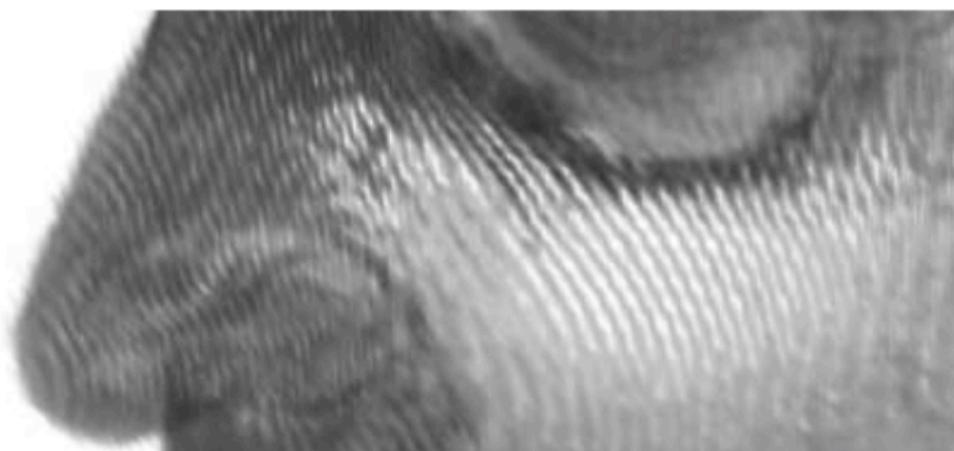


1. Bilinear Interpolation by 2D Texture Unit
2. Blending of two adjacent slice images
→ Trilinear Interpolation

$$S_{i+\alpha} = (1 - \alpha) S_i + \alpha S_{i+1}$$

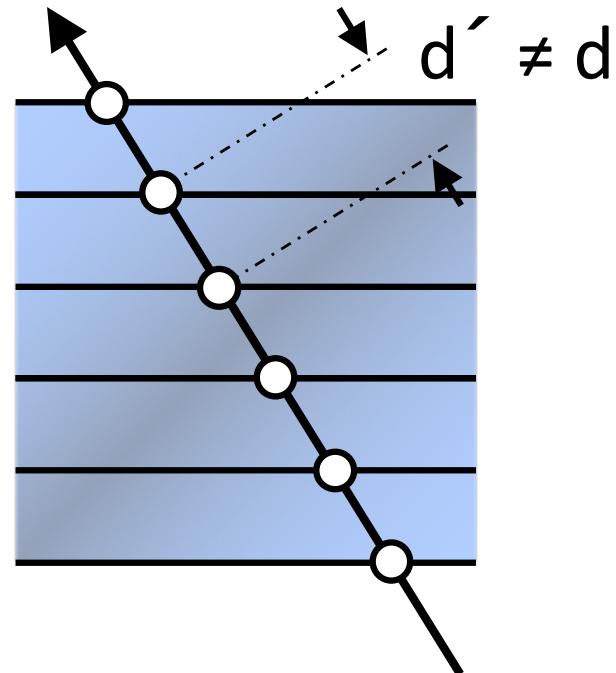
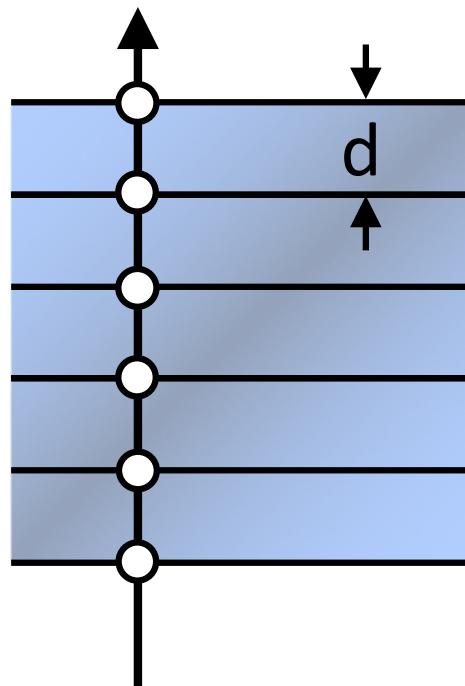
Object-order Approach

- Drawbacks of axis-aligned slices
 - Aliasing artifacts become visible at the edges of the slice polygons.



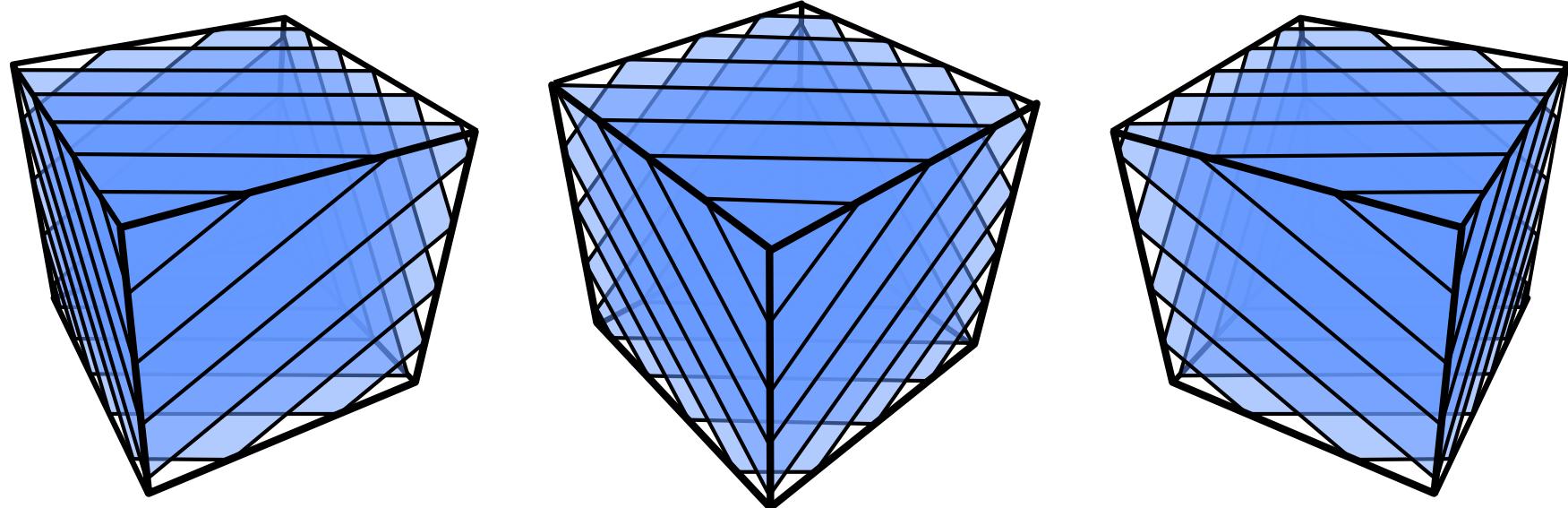
Object-order Approach

- Drawbacks of axis-aligned slices
 - Sampling rate is inconsistent.
 - Color and opacity slightly incorrect.



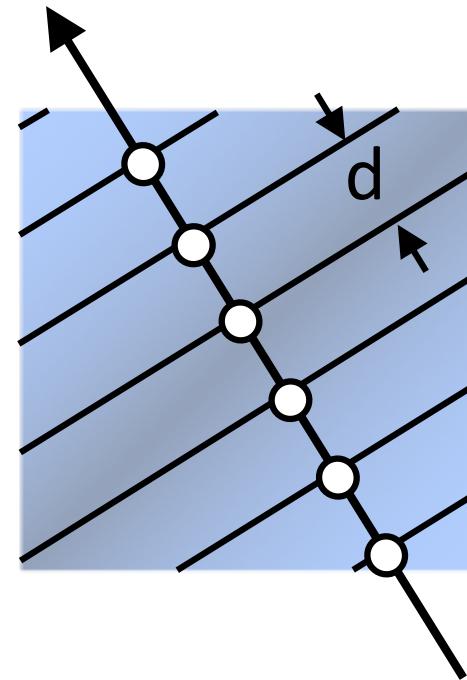
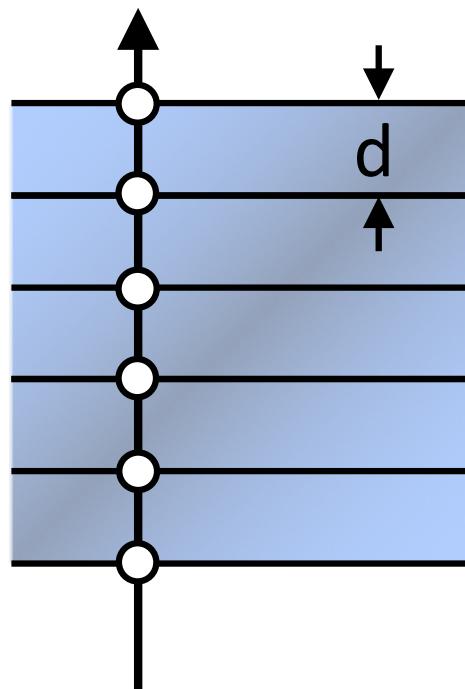
Object-order Approach

- View-aligned slices
 - Volume data is stored on GPU as a 3D texture.
 - Slices parallel to the image plane.



Object-order Approach

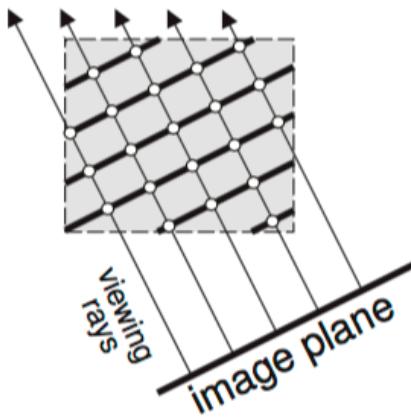
- Resampling via 3D Textures
 - Sampling rate is constant.



Object-order Approach

- Drawbacks of view-aligned slices
 - In the case of perspective projection, the sampling rate is still not consistent for all rays.

A. Parallel Projection



B. Perspective projection

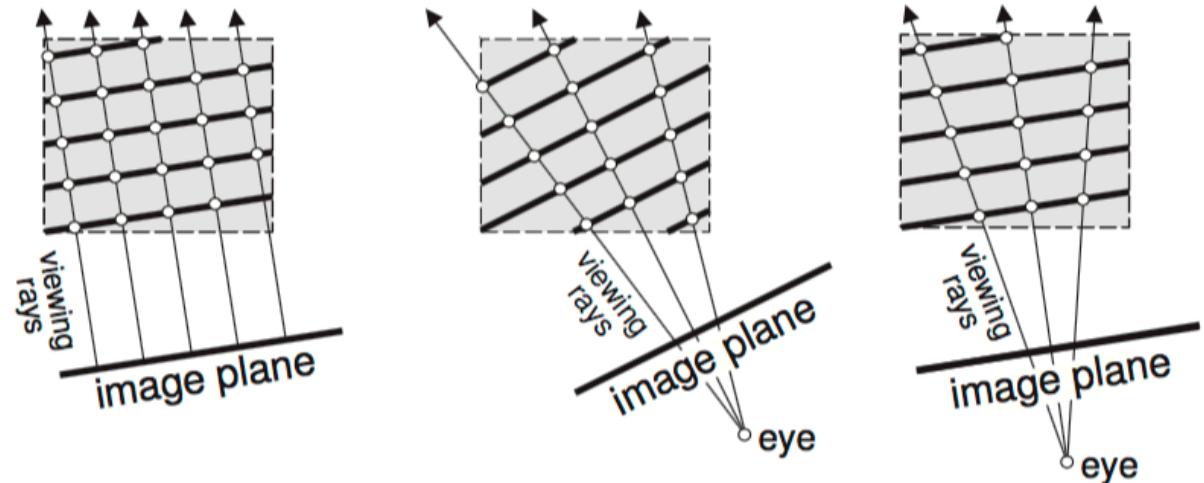


Image-order Approach

- Ray-casting approach
 - Simple implementation
 - Very flexible
 - Correct perspective projection

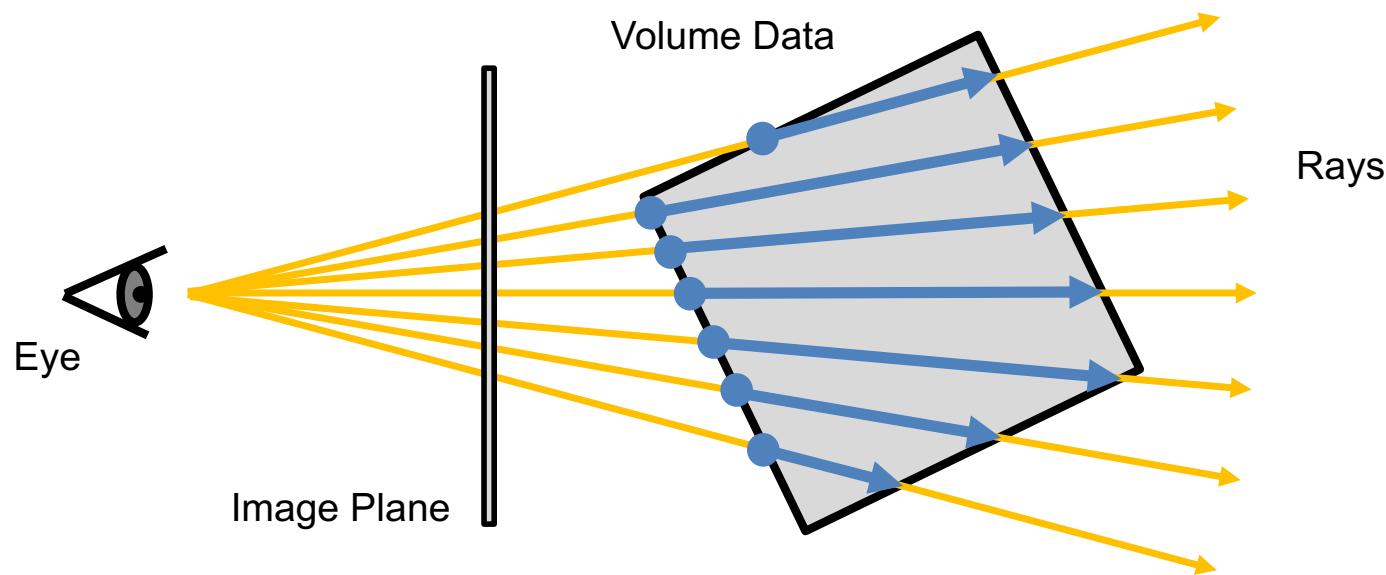
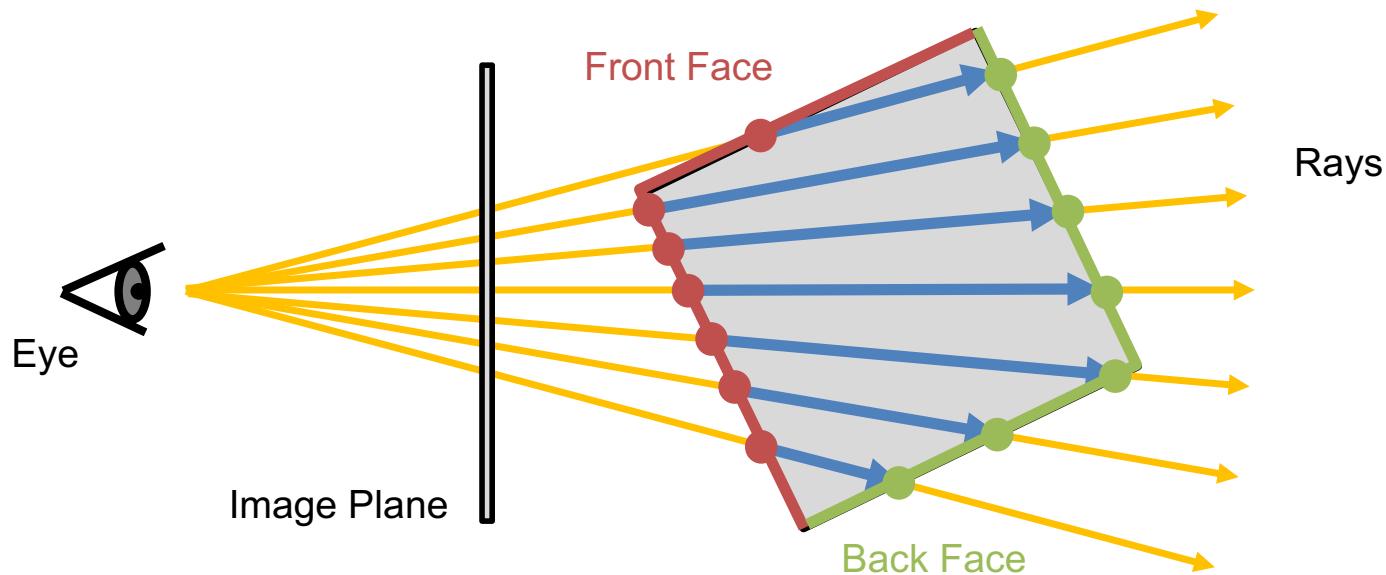


Image-order Approach

- GPU accelerated ray-casting approach
 - Rasterize front and back faces of volume bounding box.
 - Entry points of ray-casting are located on the front faces.
 - Exit points of ray-casting are located on the back faces.
 - Direction vectors can be calculated by back – front faces.



[Krugger and Westermann, 2003]

Image-order Approach

- Positions are stored in 2D Textures for the front and back faces

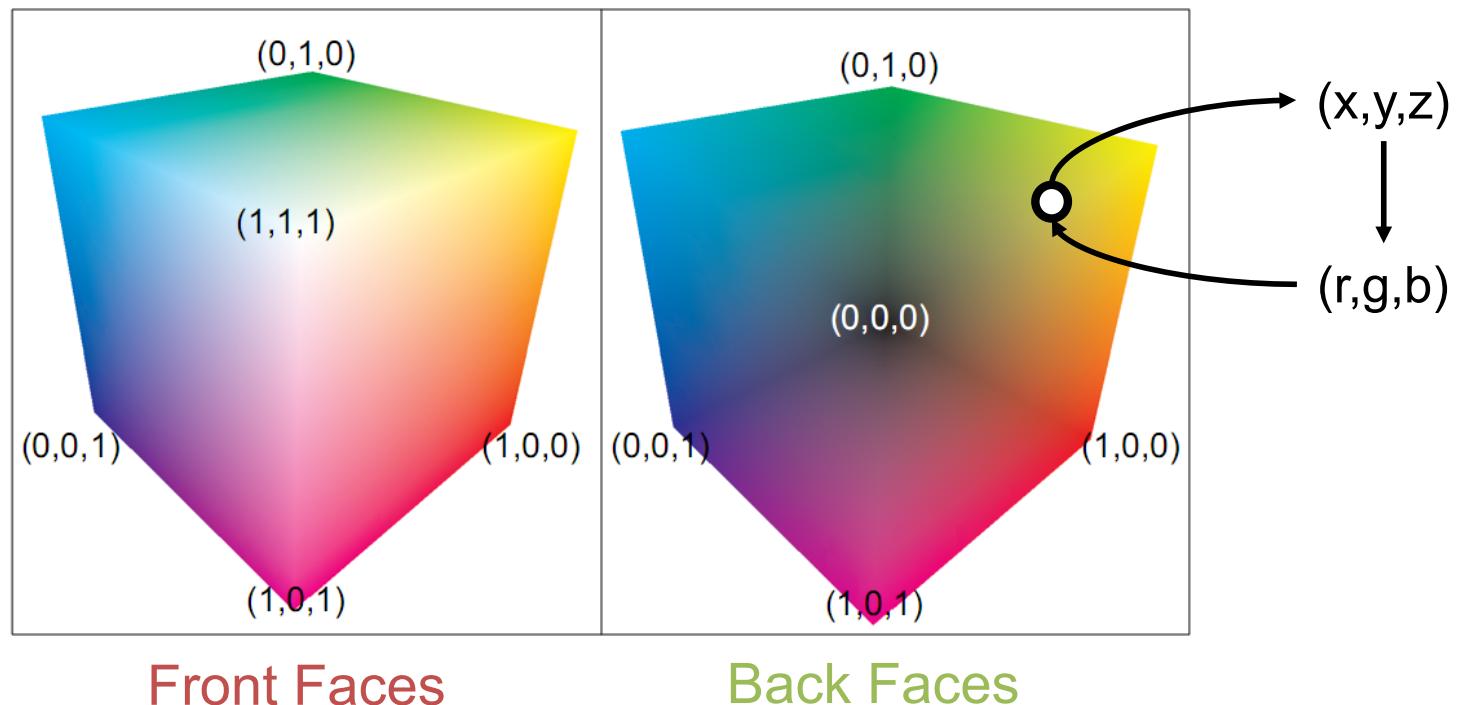


Image-order Approach

- Ray direction = Back faces – Front faces

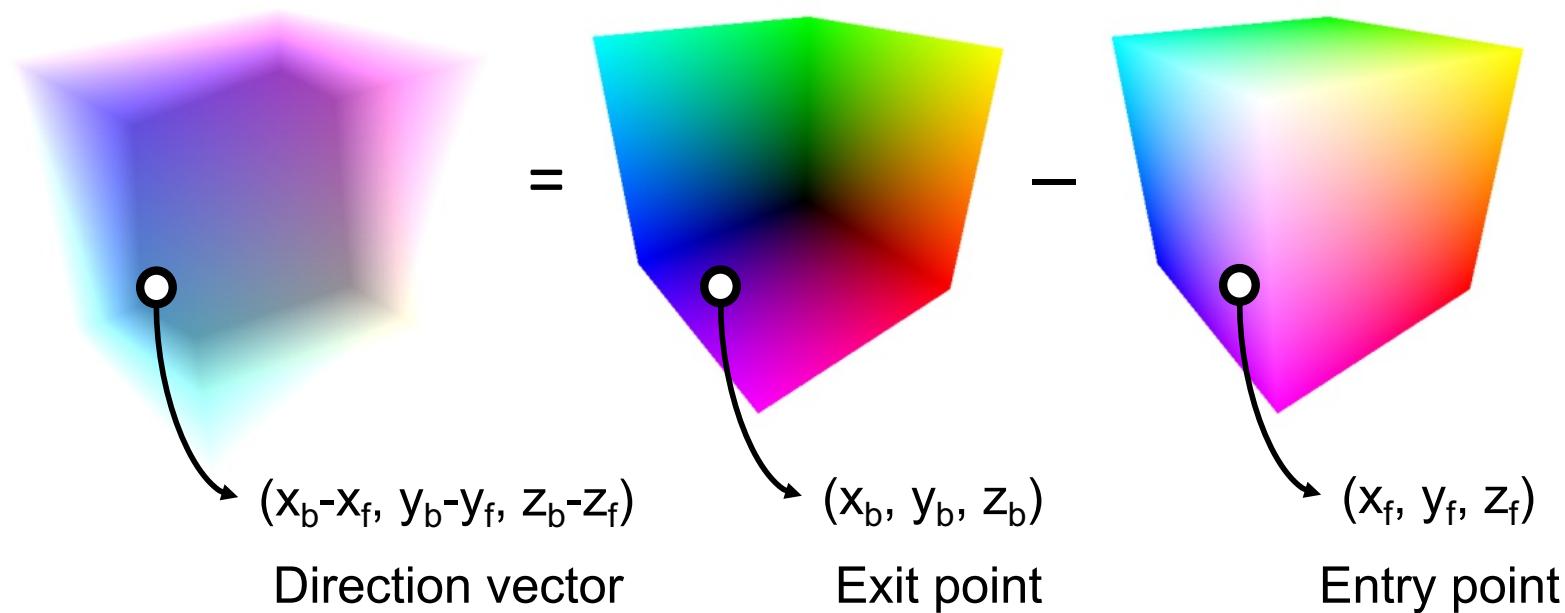
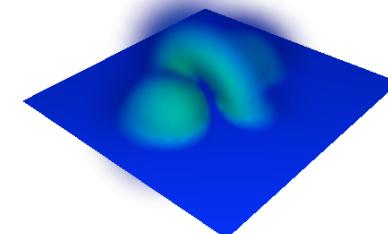
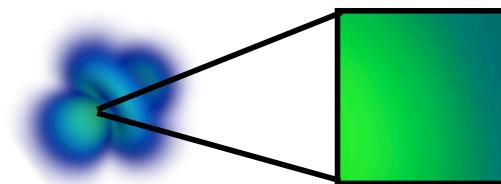
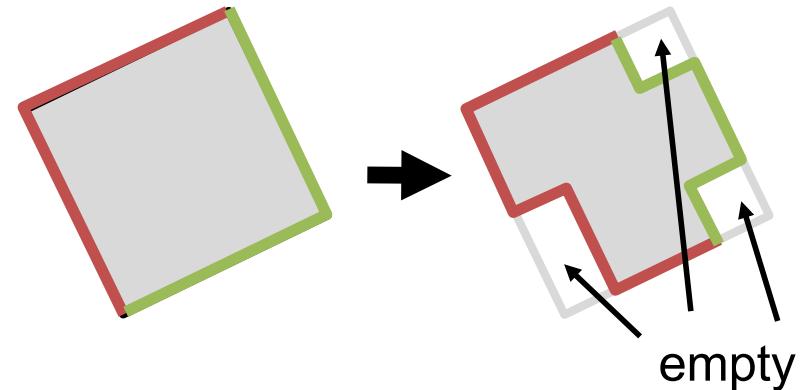


Image-order Approach

- Optimizations
 - Early Ray Termination
 - Empty Space Skipping
 - ...
- Enhancements
 - Moving into the volume
 - Rendering with geometry data
 - ...



Polling

- Take the poll
 - Student ID Number
 - Name