CS 530 Visualization

Volume Visualization

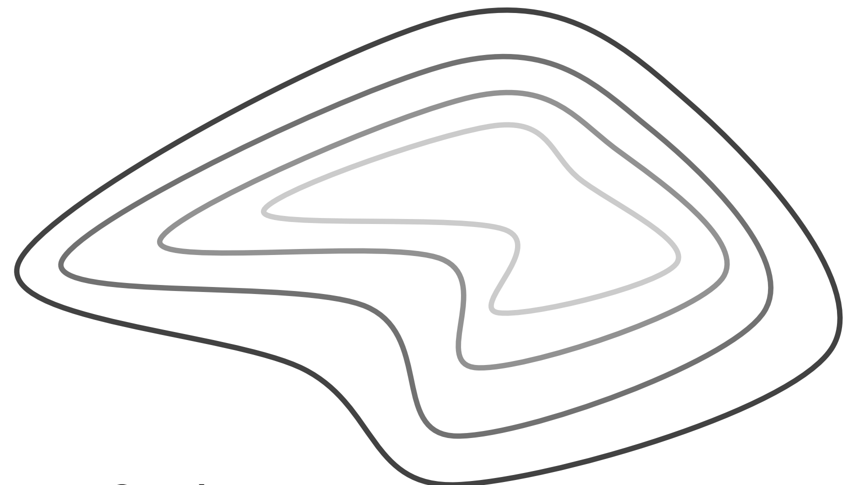
February 13, 2013

Overview

- Scalar Volumes
- Ray casting
- Unstructured Volume Rendering
- Graphics texture memory

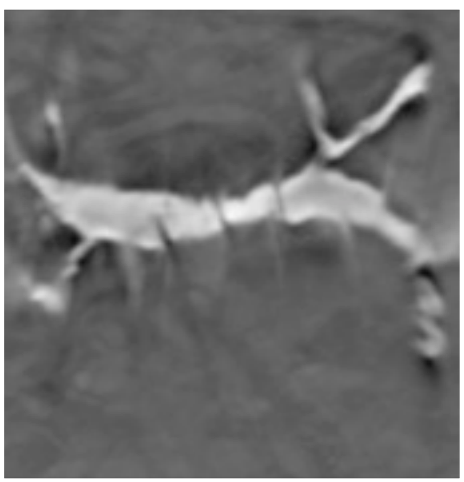
Isosurfaces

- Contour lines
  - Images → curves
  - Volumes → surfaces
- Marching cubes (Lorensen 87)
  - Triangulated surface at a given isovalue



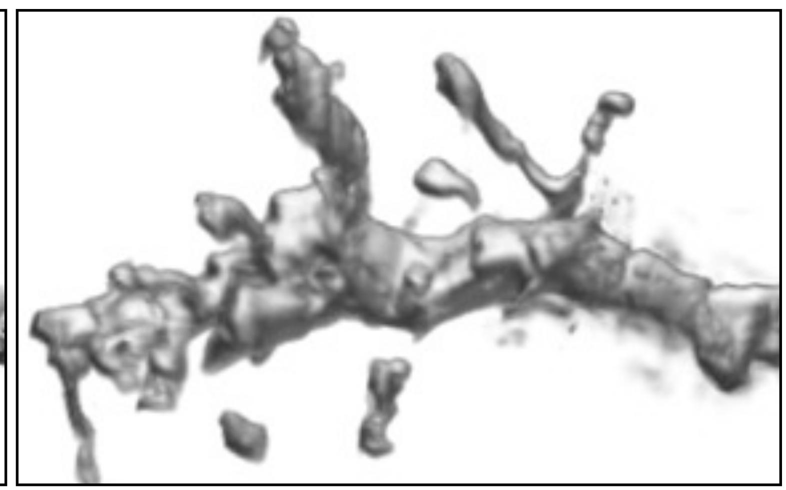
Isosurfacing is limited

- Isosurfacing is "binary"
  - point inside isosurface?
  - voxel contributes to image?
- Is a hard, distinct boundary necessarily appropriate for the visualization task?



Slice

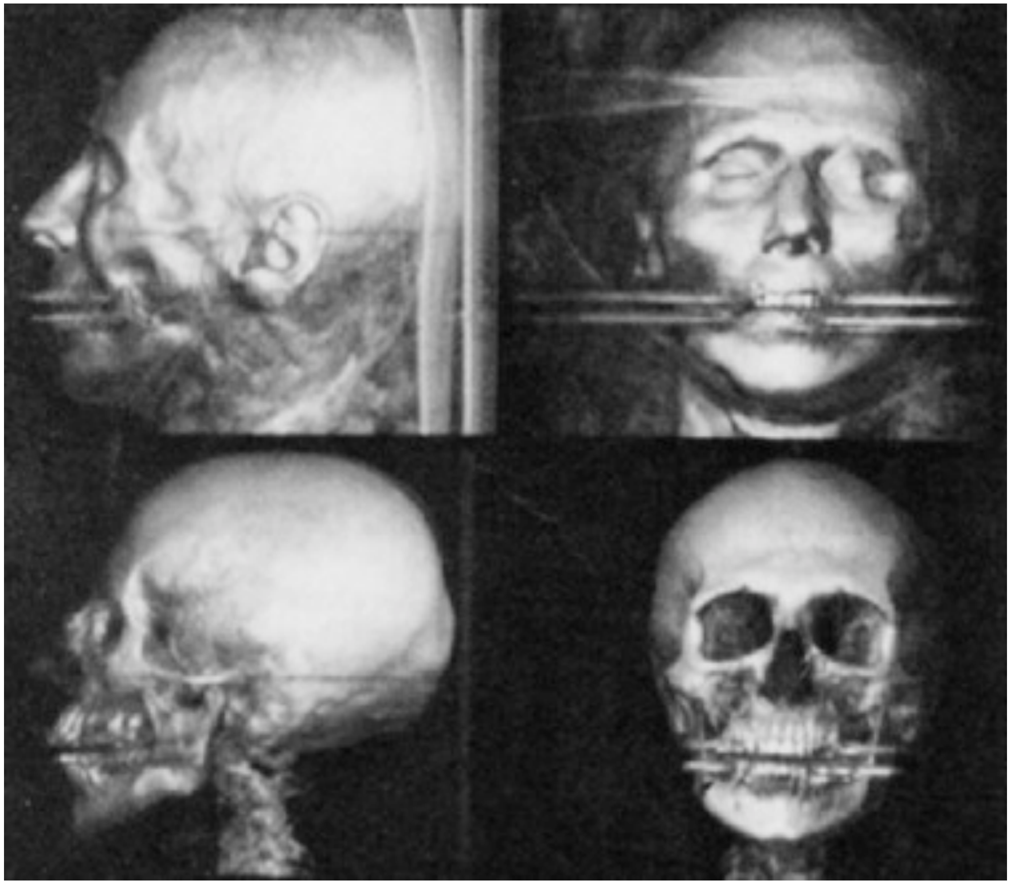
Isosurface



Volume Rendering

Spirit of volume rendering

- “‘Every voxel contributes to image"
- Greater flexibility

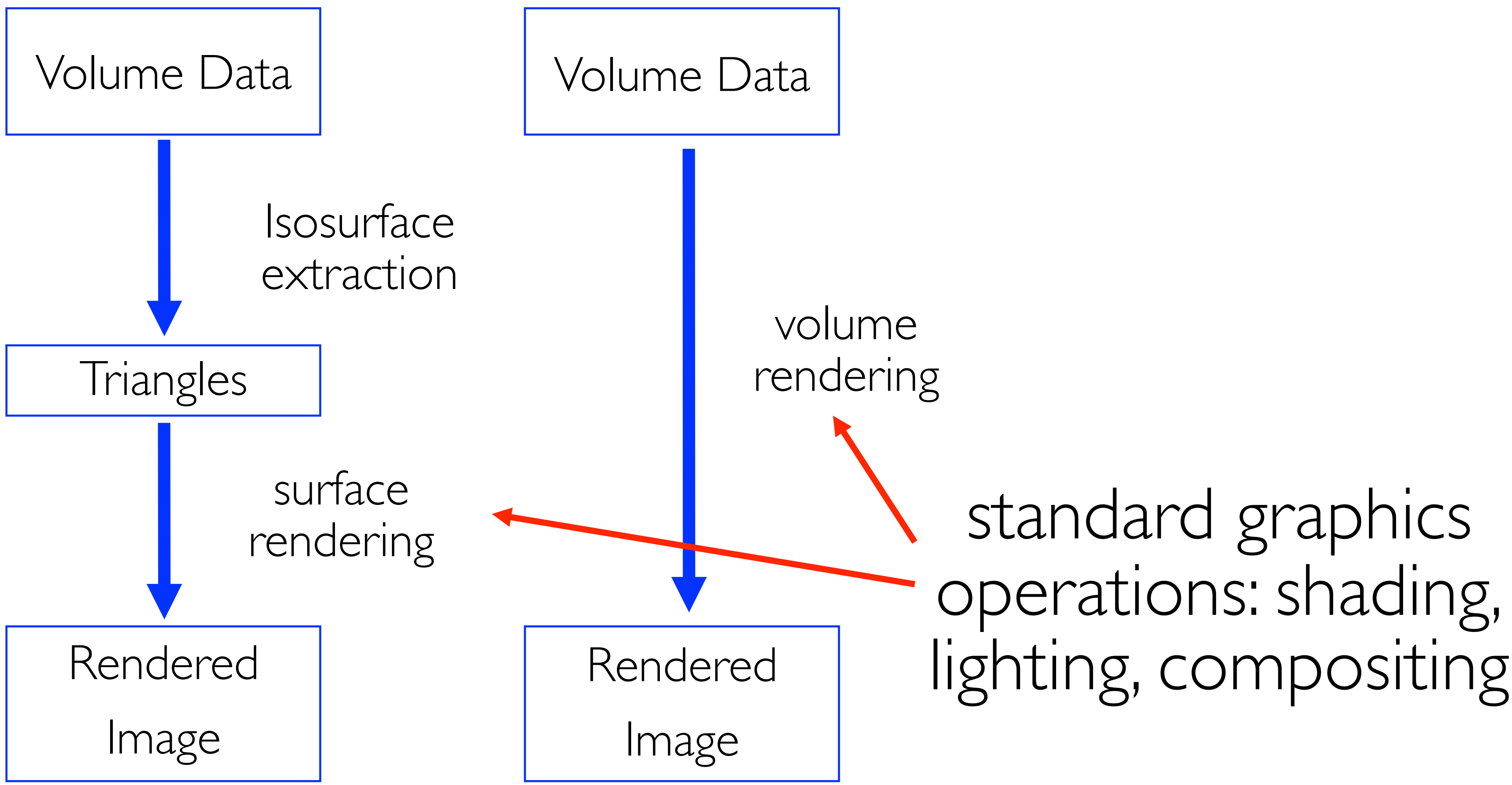


Marc Levoy, 1988

"Display of Surfaces from Volume Data"

Pipelines: Iso vs. Vol Ren

The standard line - "no intermediate geometric structures"



What is Volume Rendering?

- Any rendering process which maps from volume data to an image without introducing binary distinctions / intermediate geometry
- How do you make the data visible?

What is Volume Rendering?

- Any rendering process which maps from volume data to an image without introducing binary distinctions / intermediate geometry

- How do you make the data visible?

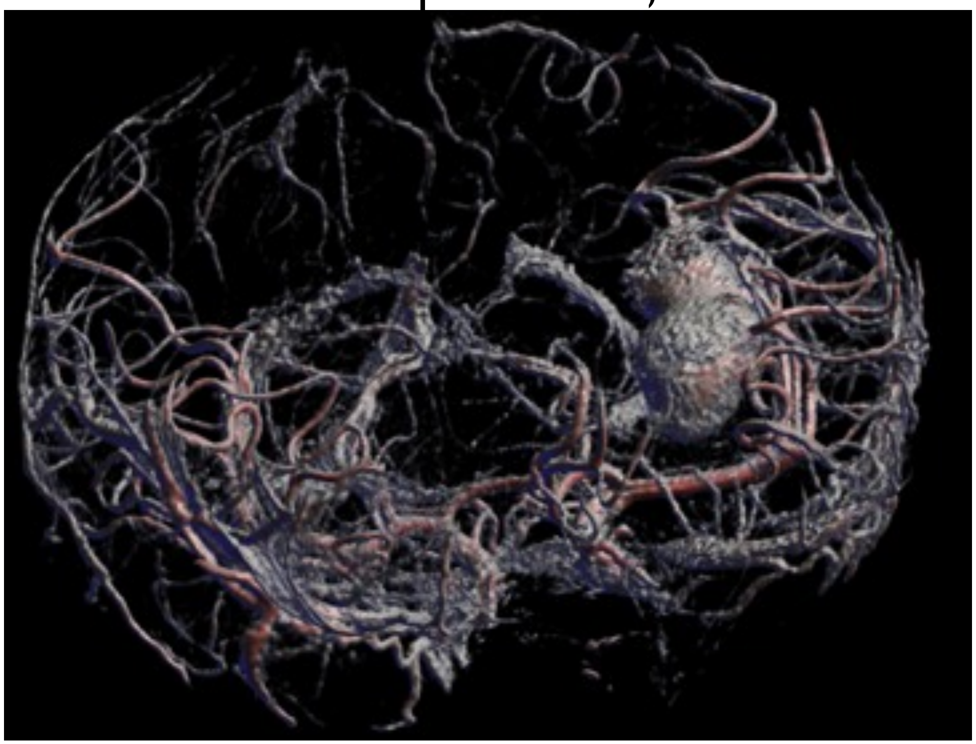
  ➡ color and opacity

Direct volume rendering

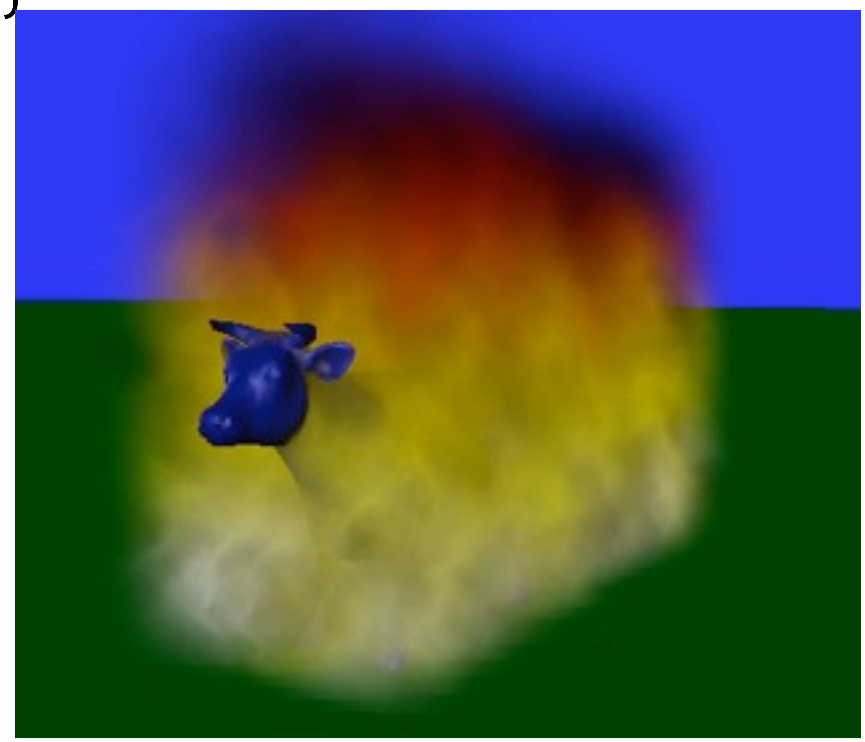
- Directly get a 3D representation of the volume data
  - The data is considered to represent a semi-transparent light-emitting medium
    - Also gaseous phenomena can be simulated
  - Approaches are based on the laws of physics (emission, absorption, scattering)
  - The volume data is used as a whole (look inside, see all interior structures)

Isosurfacing is Limited

- Isosurfacing poor for ...
  - measured, "real-world" (noisy) data
  - amorphous, "soft" objects



virtual angiography



bovine combustion simulation

Fundamentals (Physics)

- Density attenuation
  - Kajiya:

$$e^{-\tau \int_{t_1}^{t_2} \sigma(t)dt}$$

- Volume Rendering Integral

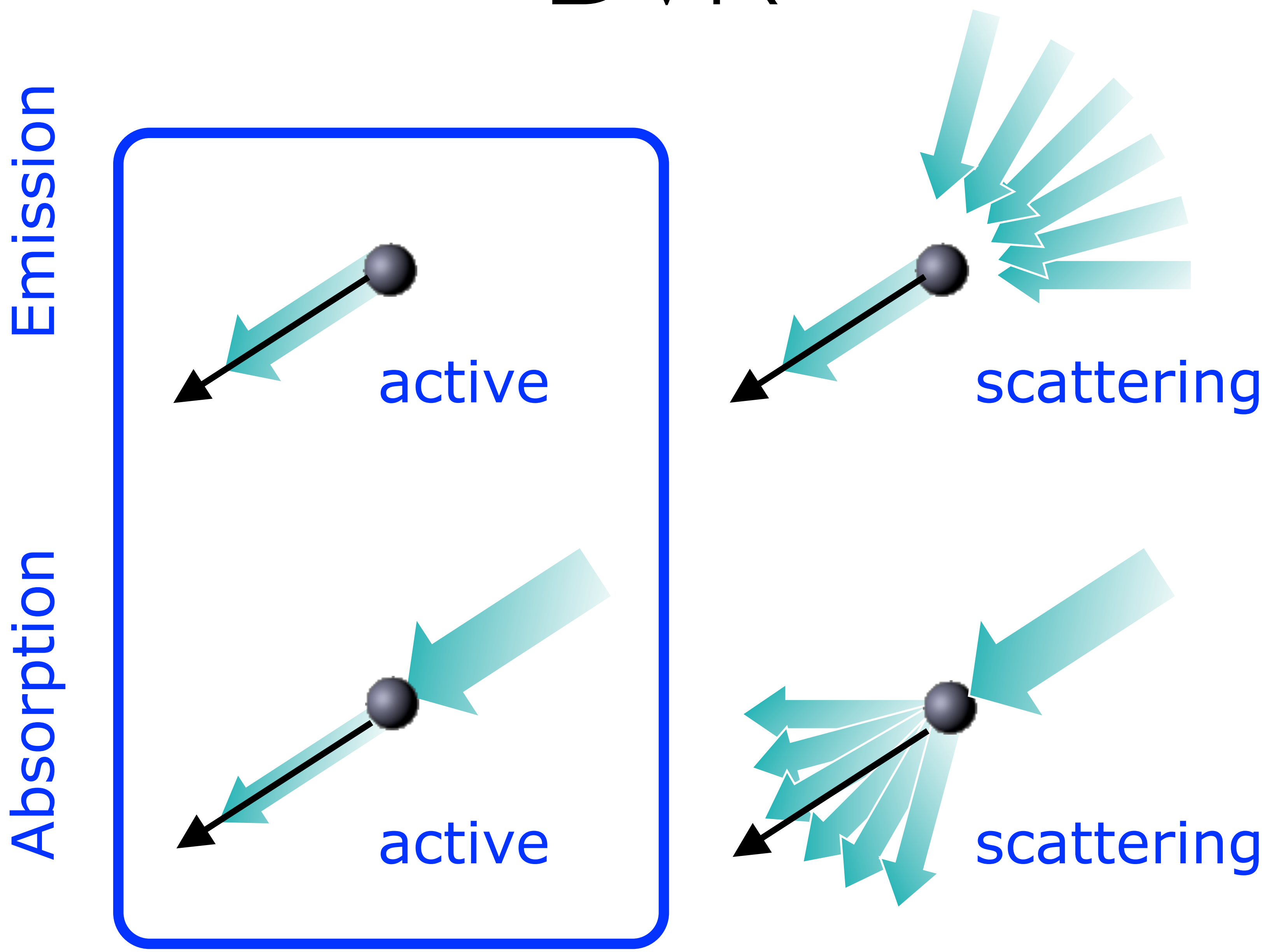
Integral along ray

emitted color

cumulative absorption

$$c(\mathbf{R}) = \boxed{\int_0^D} \boxed{c(s(x(t)))\mu(s(x(t)))} \boxed{e^{-\int_0^t \mu(s(u))du} dt}$$

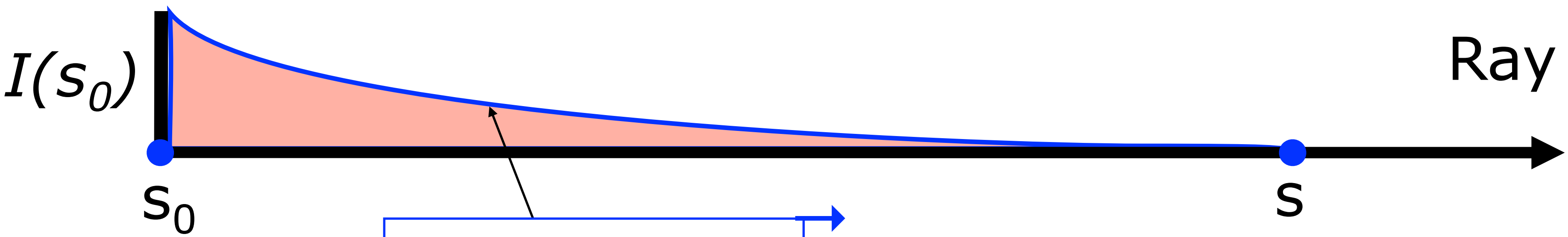
DVR



Markus Hadwiger, IEEE Visualization 2002 Tutorial Notes

DVR *Integration*

- Emission and absorption of light



Absorption along the distance s - s_0

Initial intensity (Emission) at s_0

$$\boxed{I(s) = I(s_0)e^{-\tau(s_0,s)}}$$

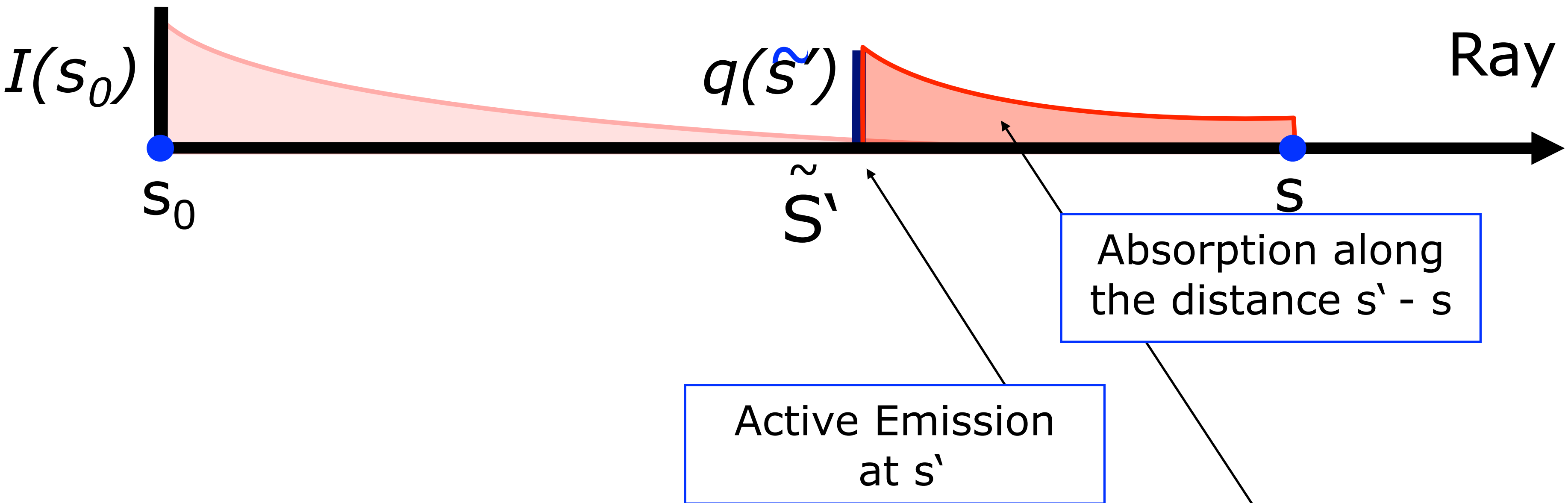
Absorption

$$\boxed{\tau(s_1, s_2) = \int_{s_1}^{s_2} \kappa(s)ds}$$

Markus Hadwiger, IEEE Visualization 2002 Tutorial Notes

DVR

- Emission and absorption of light

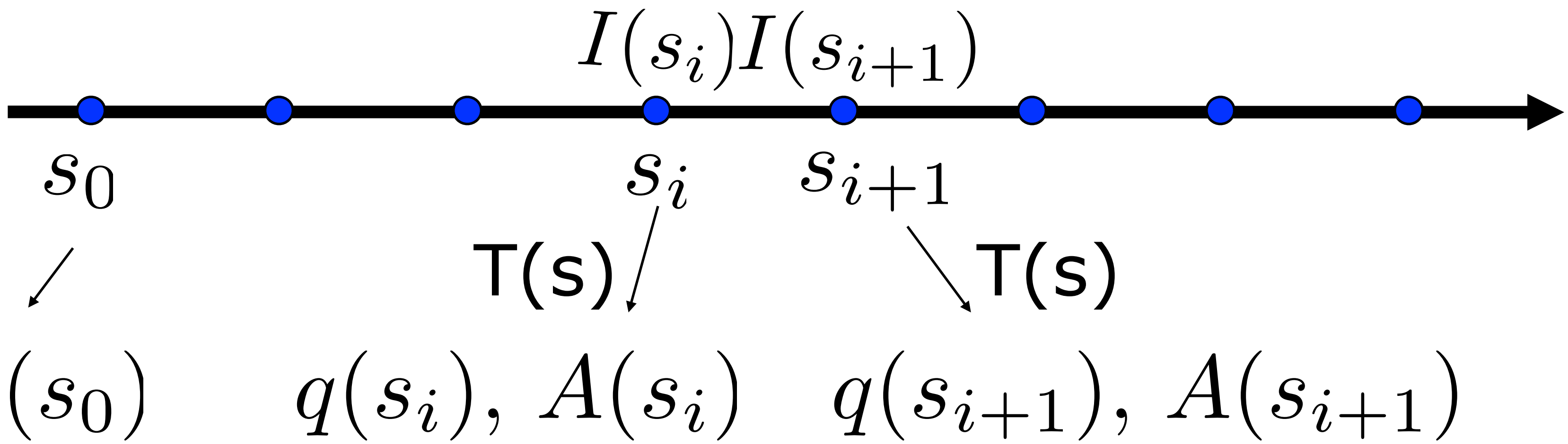


$$I(s) = I(s_0)e^{-\tau(s_0,s)} + \int_{s_0}^{s} q(\acute{s})e^{-\tau(s,\acute{s})}d\acute{s}$$

Markus Hadwiger, IEEE Visualization 2002 Tutorial Notes

DVR
Discrete Approximation

Resample along ray



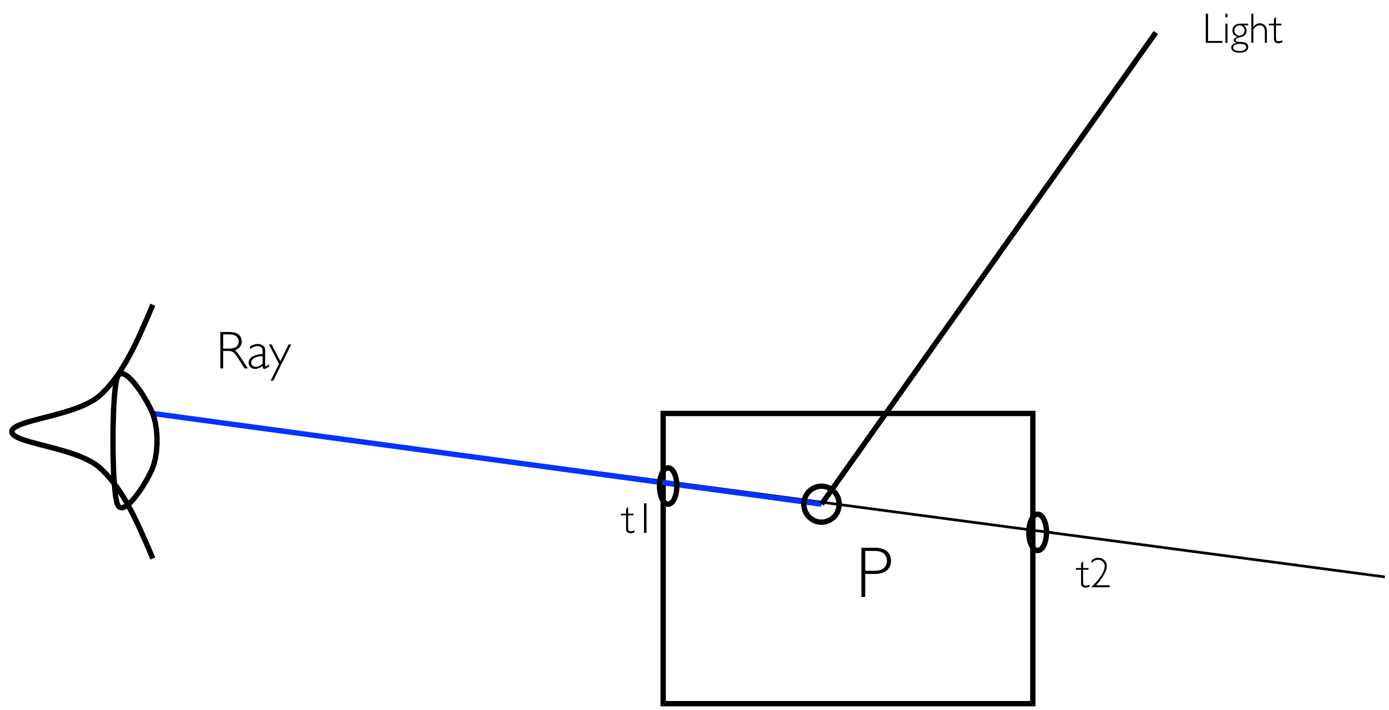
Back-to-front Compositing with $\alpha = A(s_{i+1})$

$$
\begin{aligned}
I(s_{i+1}) &= \alpha q(s_{i+1}) + (1-\alpha) I(s_i) \\
&= q(s_i + 1) \text{ OVER } I(s_i)
\end{aligned}
$$

Markus Hadwiger, IEEE Visualization 2002 Tutorial Notes

General Components

- Basic diagram

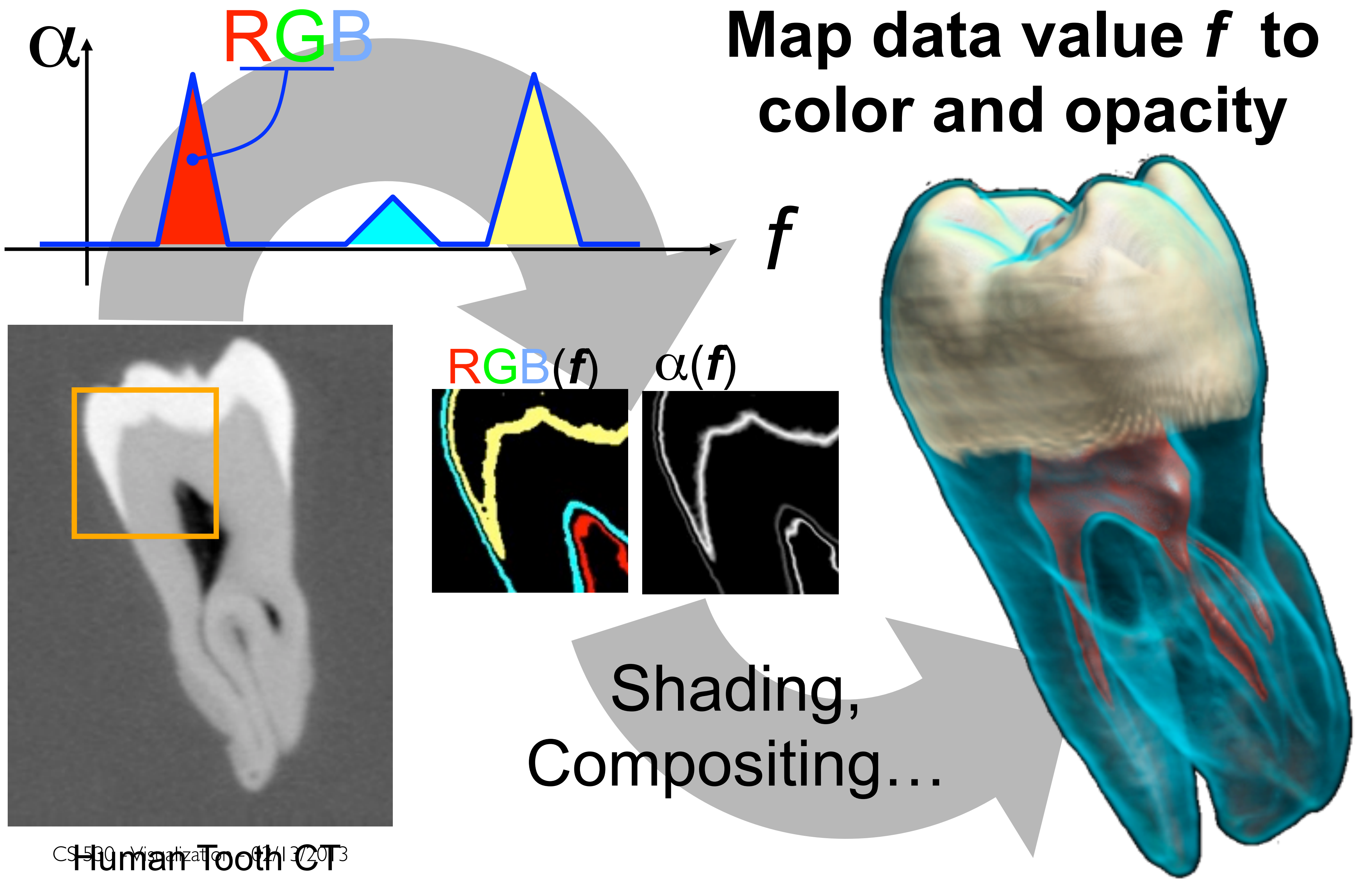


Color and Opacity Transfer Functions

- $C(p), \alpha(p)$ – p is a point in volume
- Functions of input data $f(p)$
  - $C(f), \alpha(f)$ – these are 1D functions
  - Can include lighting affects
    - $C(f, N(p), L)$ where $N(p) = grad(f)$
  - Derivatives of f
    - $C(f, grad(f)), \alpha(f, grad(f))$

Transfer Functions (TFs)

Map data value *f* to color and opacity

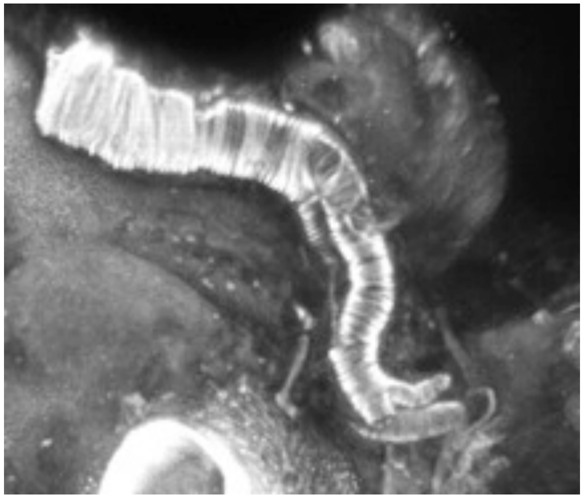


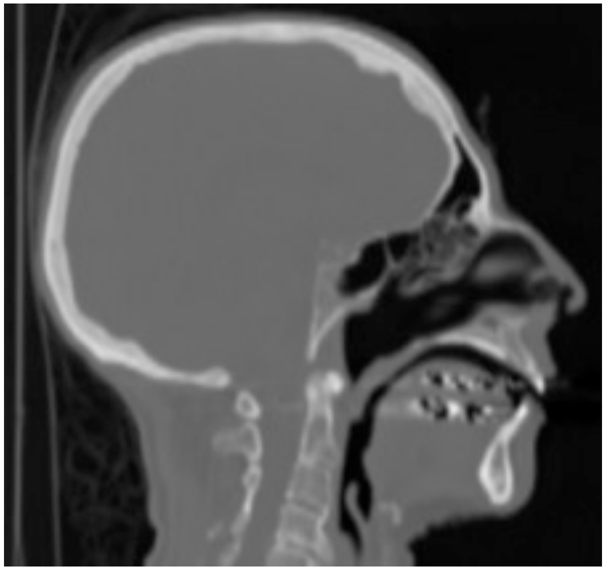
Human Tooth CT

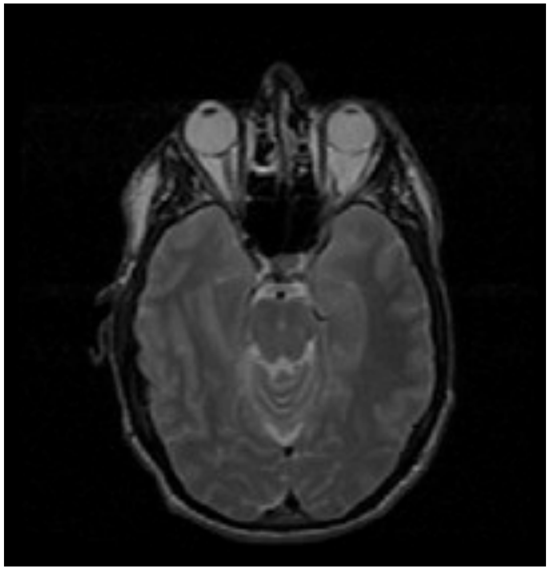
Volume Rendering Usefulness

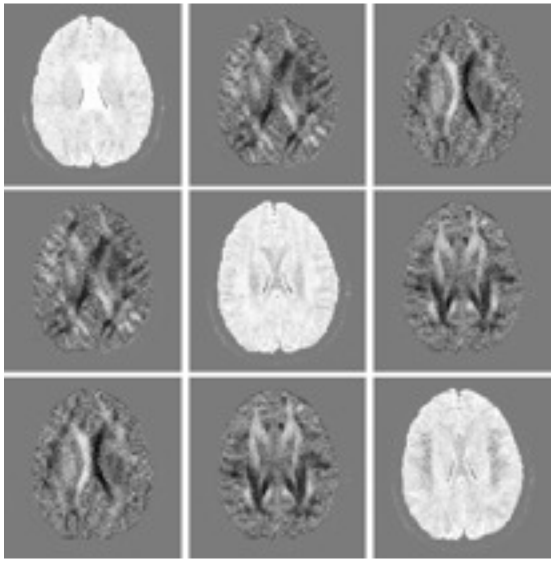
Measured sources of volume data

- CT (computed tomography)
- PET (positron emission tomography)
- MRI (magnetic resonance imaging)
- Ultrasound
- Confocal Microscopy



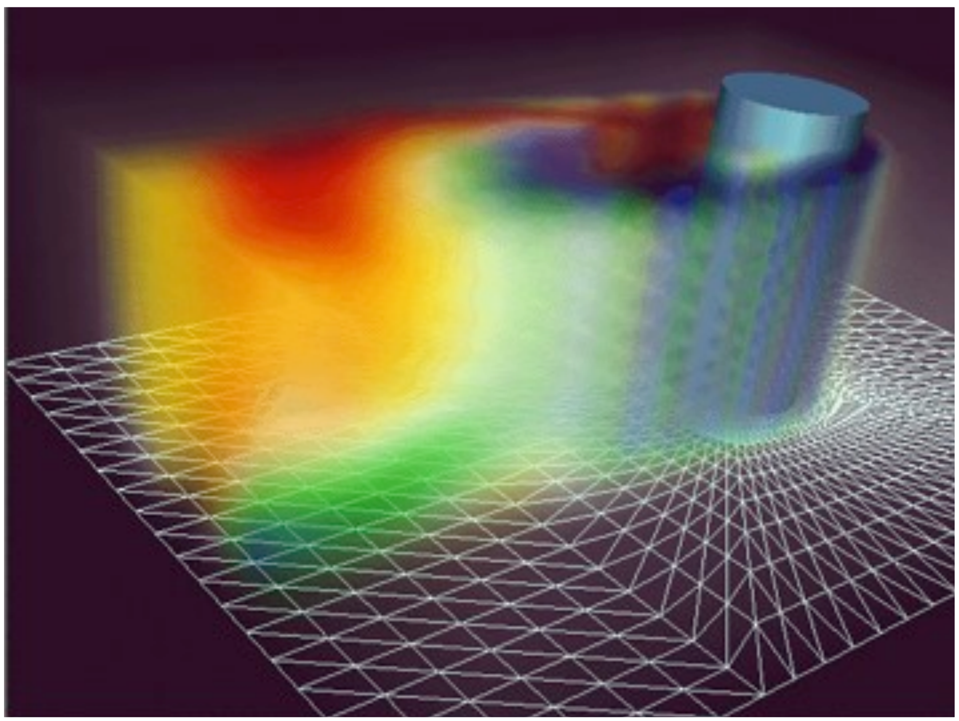


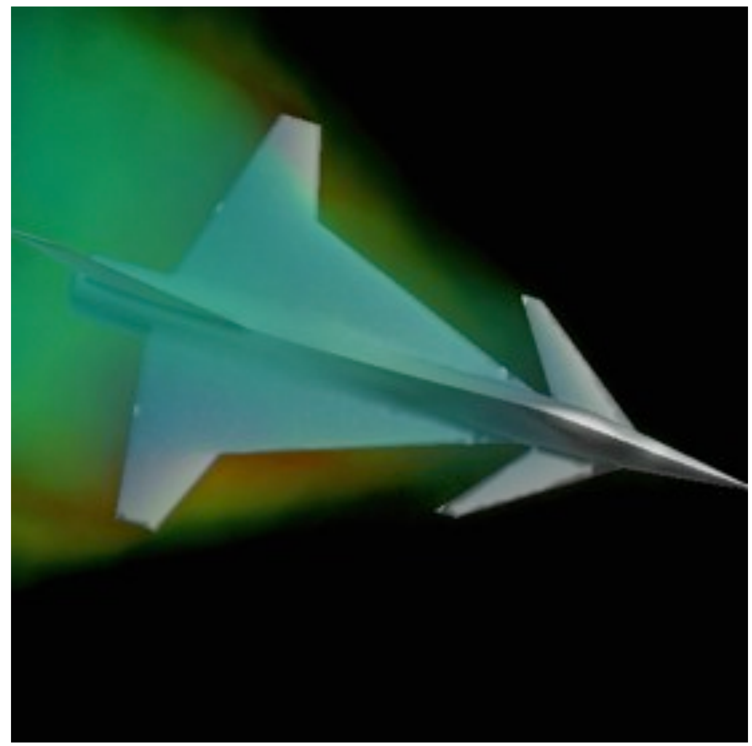




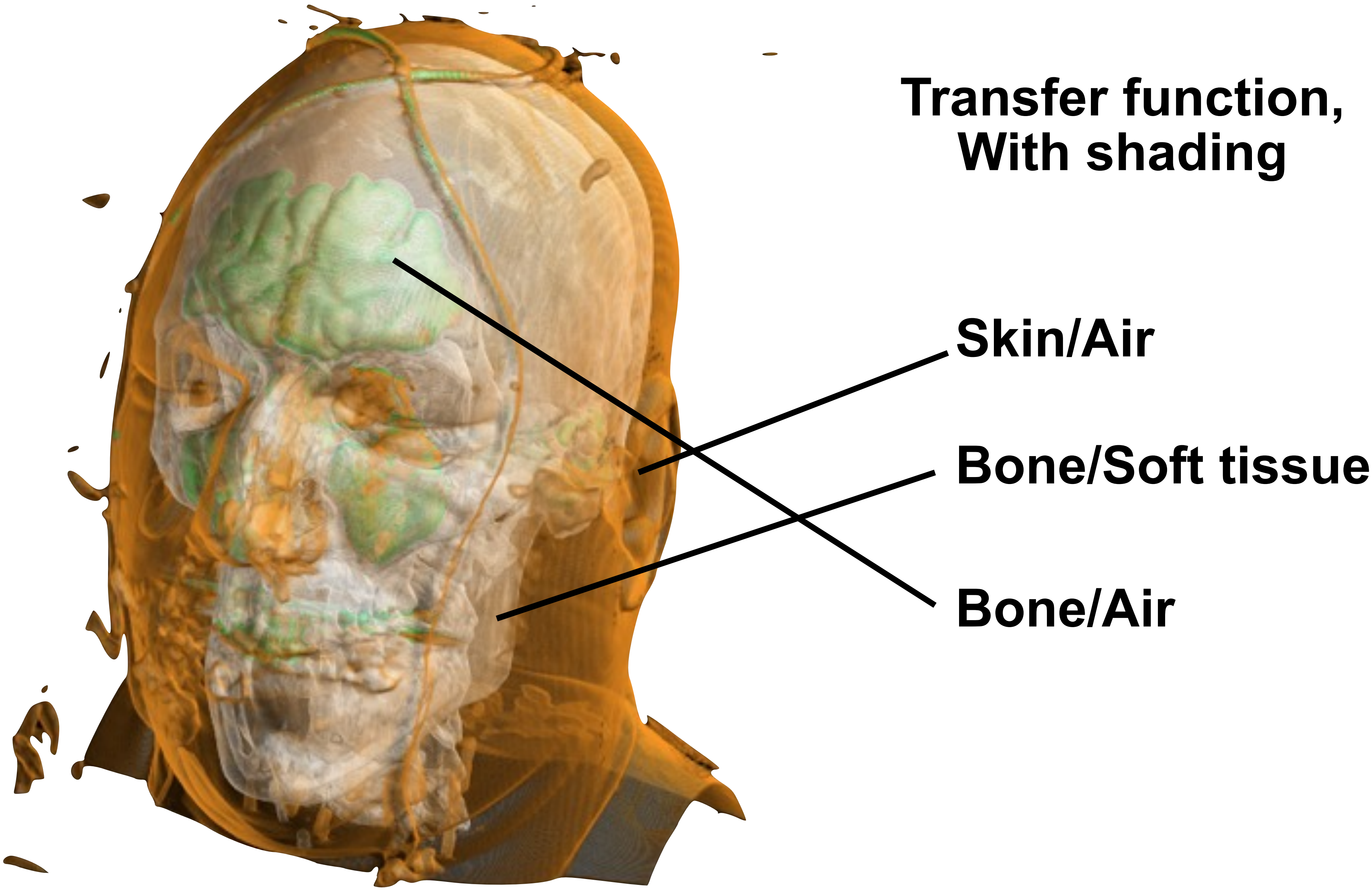
Volume Rendering Usefulness

- Synthetic sources of volume data
- CFD (computational fluid dynamics)
- Voxelization of discrete geometry



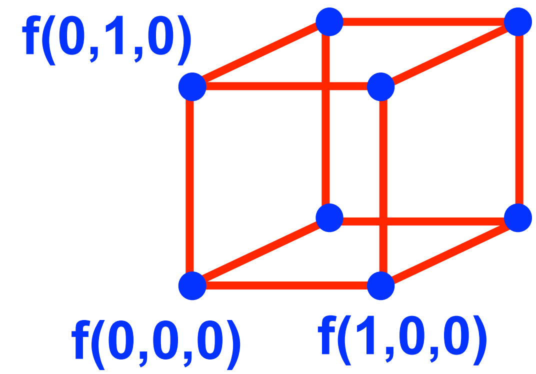


Volume Rendering: Interfaces

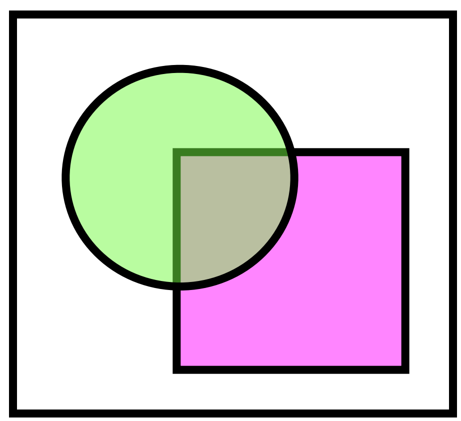


Concepts

- Voxels
  - basic unit of volume data



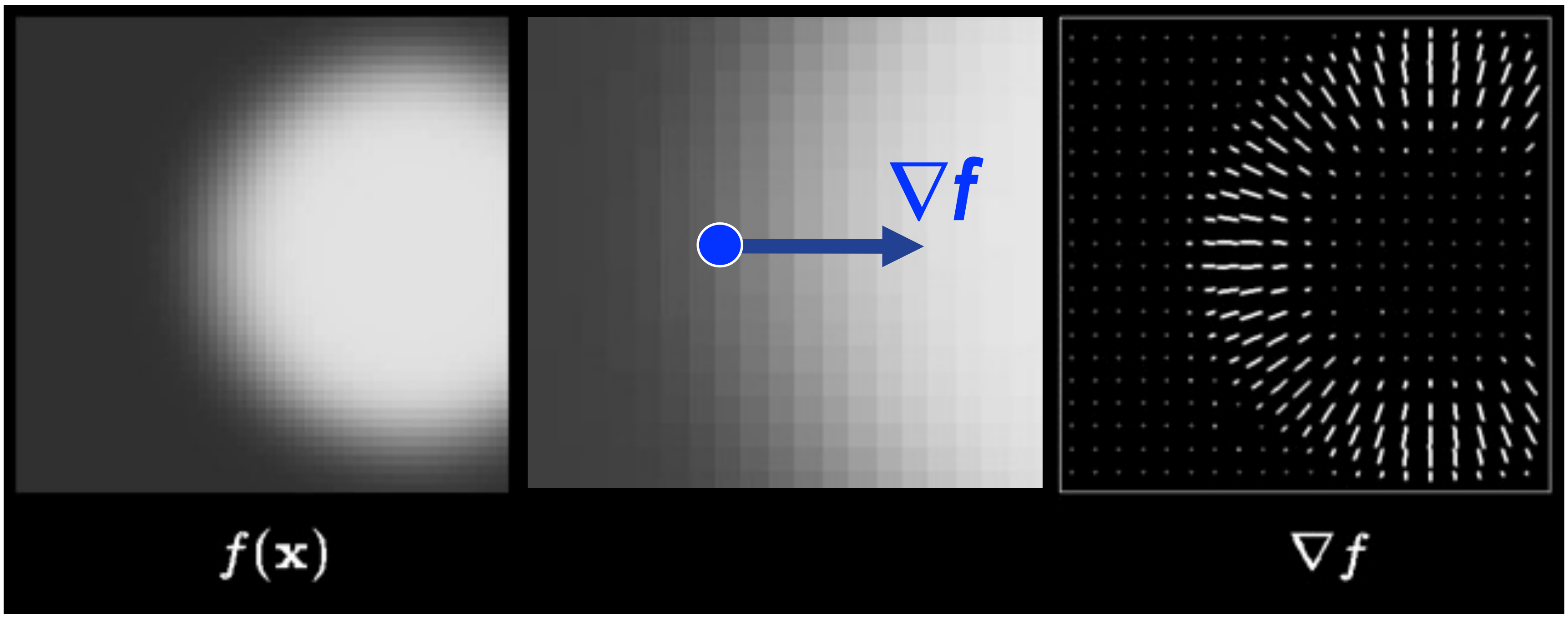
- Interpolation
  - trilinear common, others possible
- Gradient
  - direction of fastest change
- Compositing
  - "over operator"



Gradient

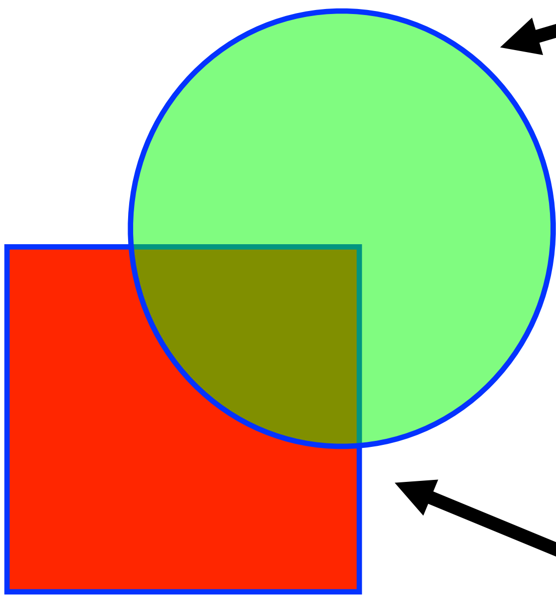
$$\nabla f = (dx, dy, dz)$$
$$= ((f(1,0,0) - f(-1,0,0))/2,$$
$$(f(0,1,0) - f(0,-1,0))/2,$$
$$(f(0,0,1) - f(0,0,-1))/2)$$

- Approximates "surface normal" (of isosurface)



Compositing: Over Operator

$C_f = (0,1,0)$
$a_f = 0.4$



$C_b = (1,0,0)$
$a_b = 0.9$

$$\boxed{\begin{gathered} c = a_f * c_f + (1 - a_f) * a_b * c_b \\ a = a_f + (1 - a_f) * a_b \end{gathered}}$$

$c_{red} = 0.4*0 + (1-0.4)*0.9*1 = 0.6*0.9 = 0.54$
$c_{green} = 0.4*1 + (1-0.4)*0.9*0 = 0.4$
$c_{blue} = 0.4*0 + (1-0.4)*0.9*0 = 0$
$a = 0.4 + (1 - 0.4)*(0.9) = 0.4 + 0.6*0.9)$

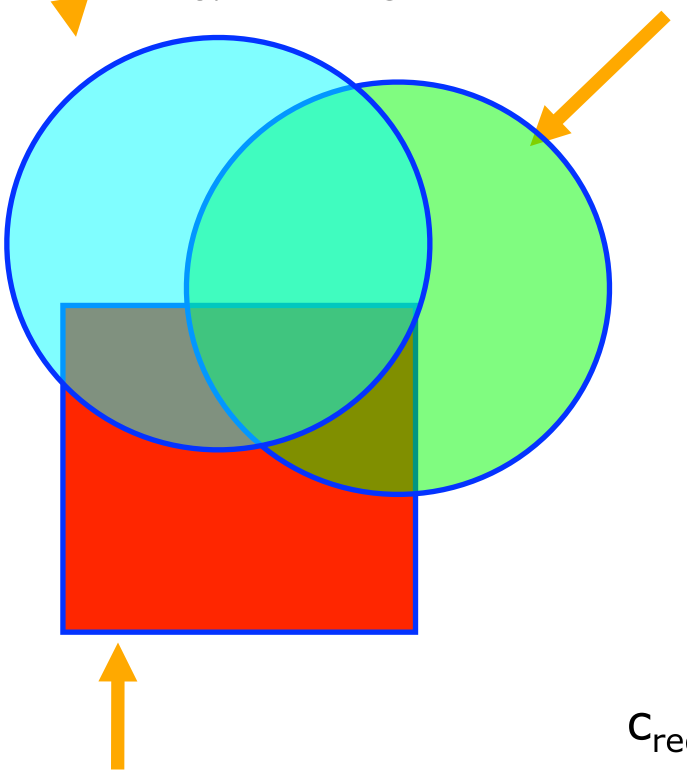
$c = (0.54,0.4,0)$
$a = 0.94$

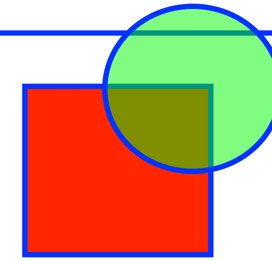
Compositing: Over Operator

$$c = a_f * c_f + (1 - a_f) * a_b * c_b$$
$$a = a_f + (1 - a_f) * a_b$$

cf = (0,1,1)
af = 0.4

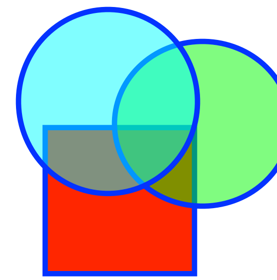
cf = (0,1,0)
af = 0.4





c_{red} = 0.4*0 + (1-0.4)*0.9*1 = 0.6*0.9 = 0.54
c_{green} = 0.4*1 + (1-0.4)*0.9*0 = 0.4
c_{blue} = 0.4*0 + (1-0.4)*0.9*0 = 0
a = 0.4 + (1 – 0.4)*(0.9) = 0.4 + 0.6*0.9)

c_b = (0.54,0.4,0)
a_b = 0.94



cb = (1,0,0)
ab = 0.9

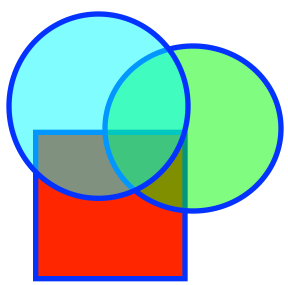
c_{red} = 0.4*0 + (1-0.4)*0.94*0.54 = 0.6*0.94*.54 = 0.30
c_{green} = 0.4*1 + (1-0.4)*0.94*0.4 = 0.6*0.94*.4 = 0.23
c_{blue} = 0.4*1 + (1-0.4)*0.94*0 = .4
a = 0.4 + (1 – 0.4)*(0.94) = 0.4 + 0.6*0.94) = .964

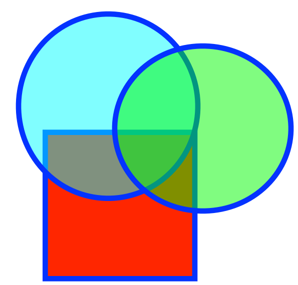
c = (0.3,0.23,0.4)
a = 0.964

Compositing: Over Operator

$$c = a_f * c_f + (1 - a_f) * a_b * c_b$$
$$a = a_f + (1 - a_f) * a_b$$

Order Matters!

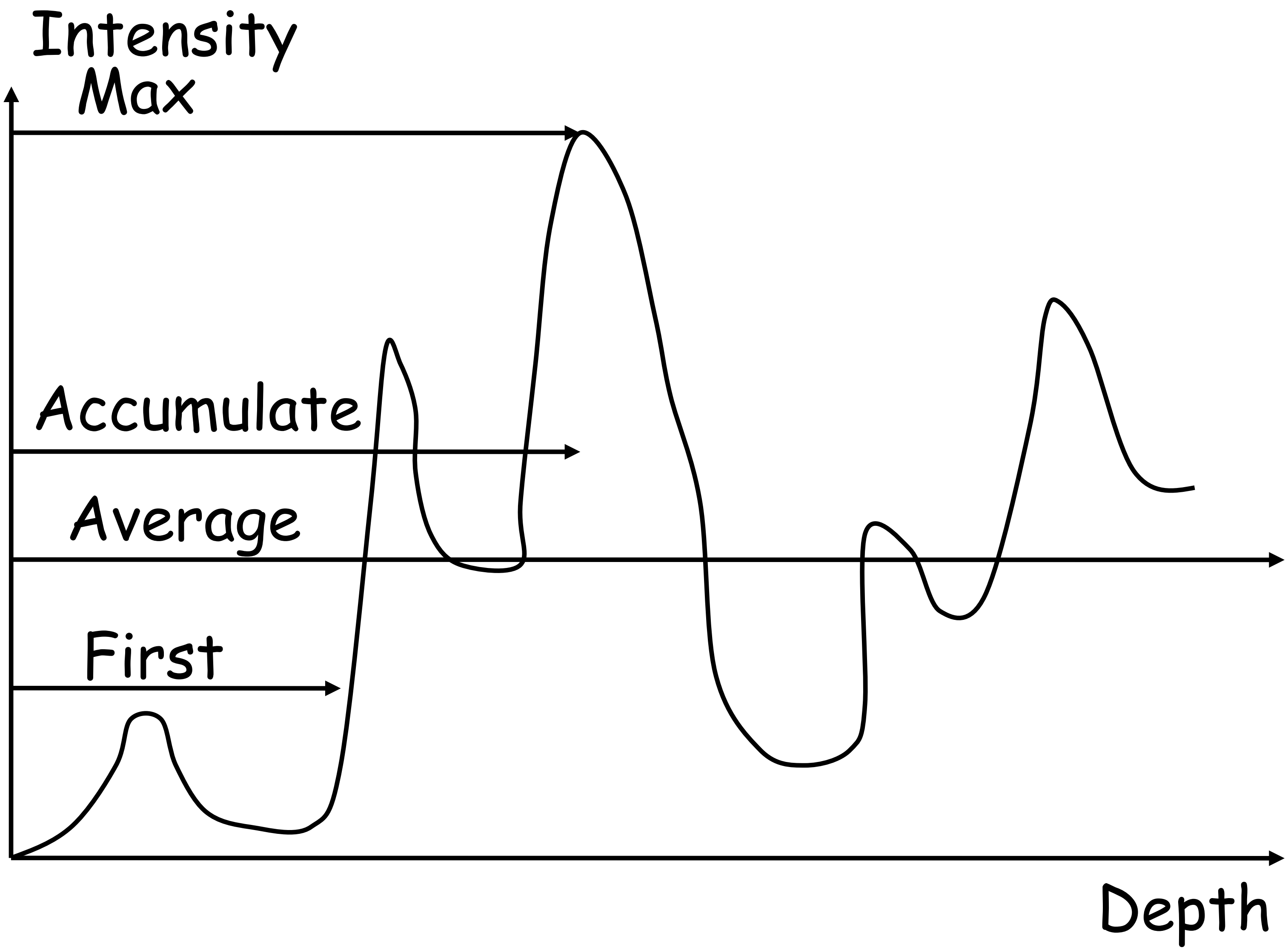




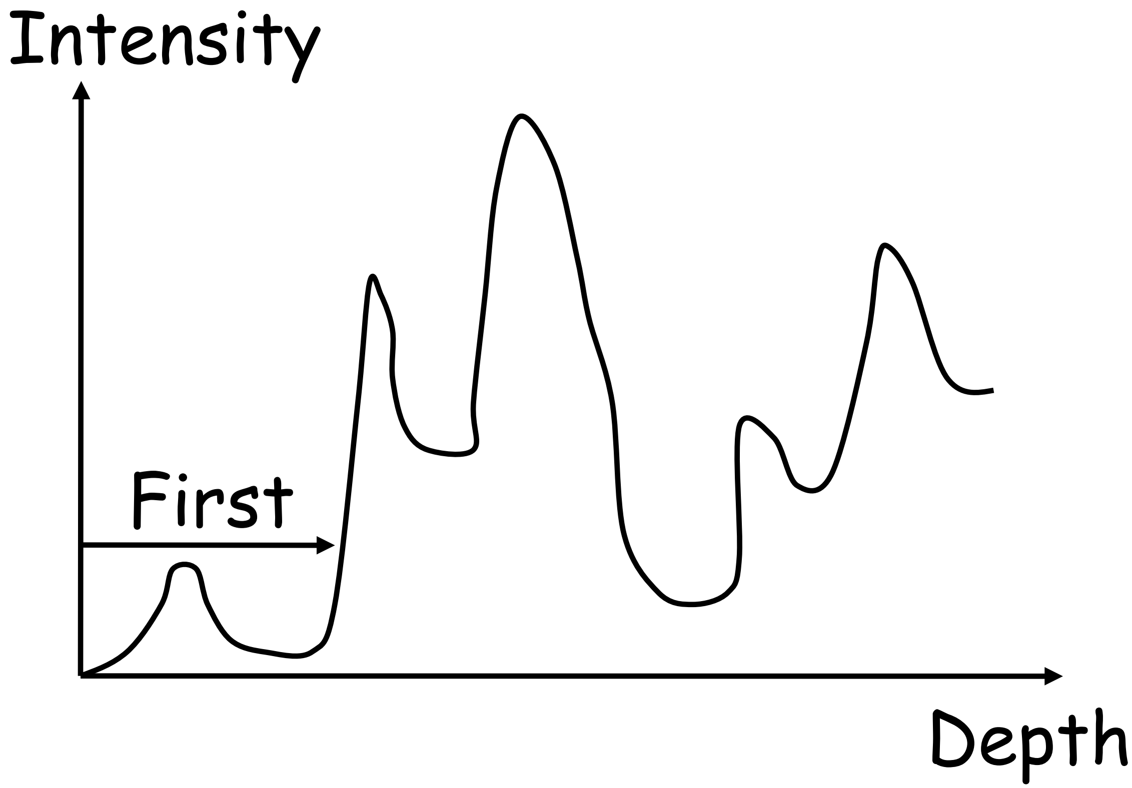
$c = (0.3, 0.23, 0.4)$
$a = 0.964$

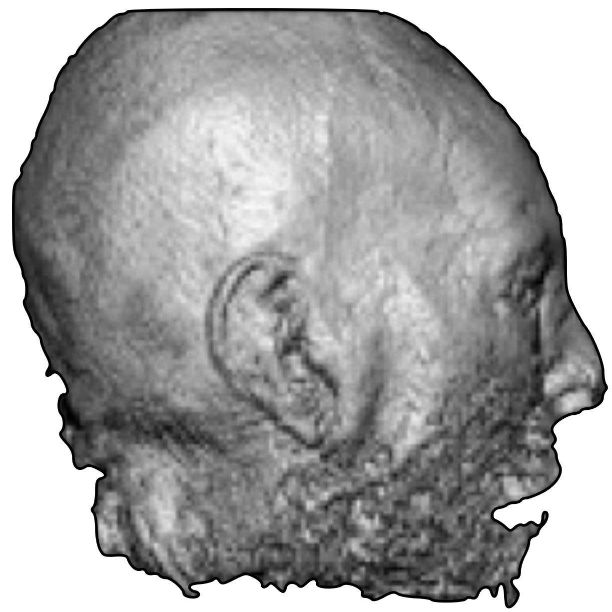
$c = (0.3, 0.23, 0.23)$
$a = 0.964$

Pixel Compositing Schemes



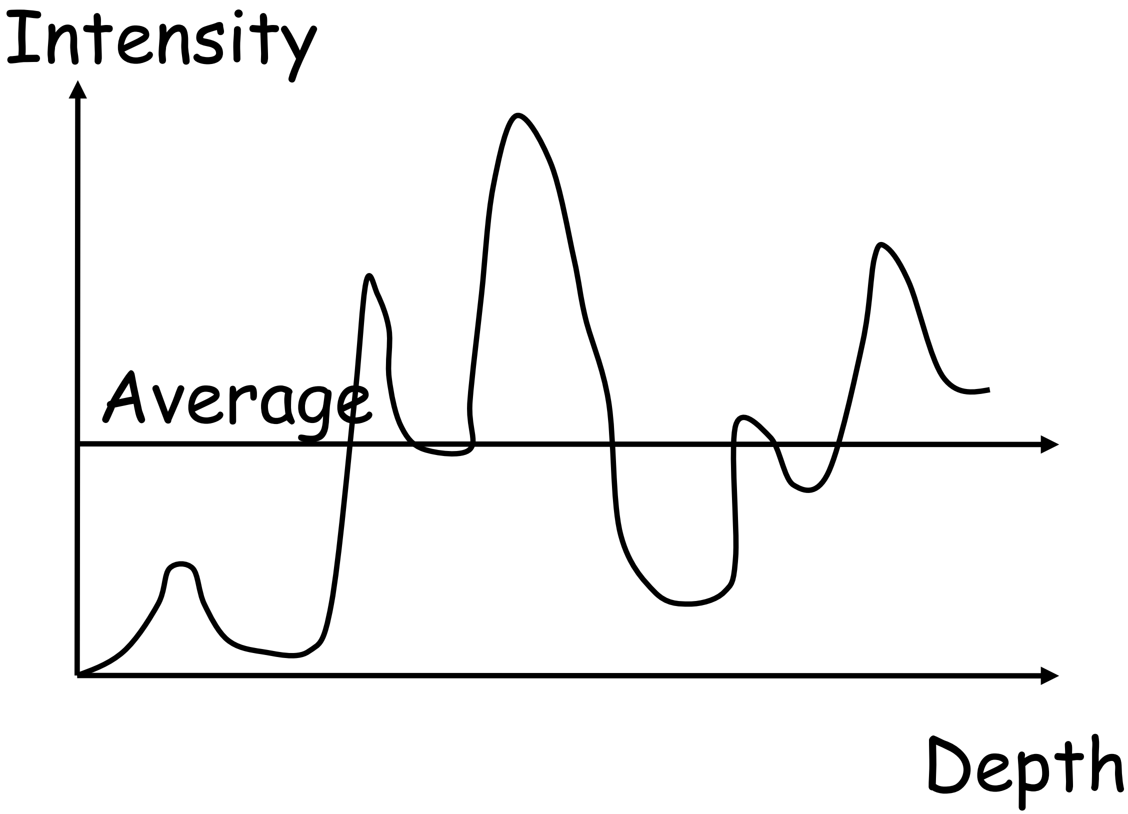
Compositing – First (Threshold)

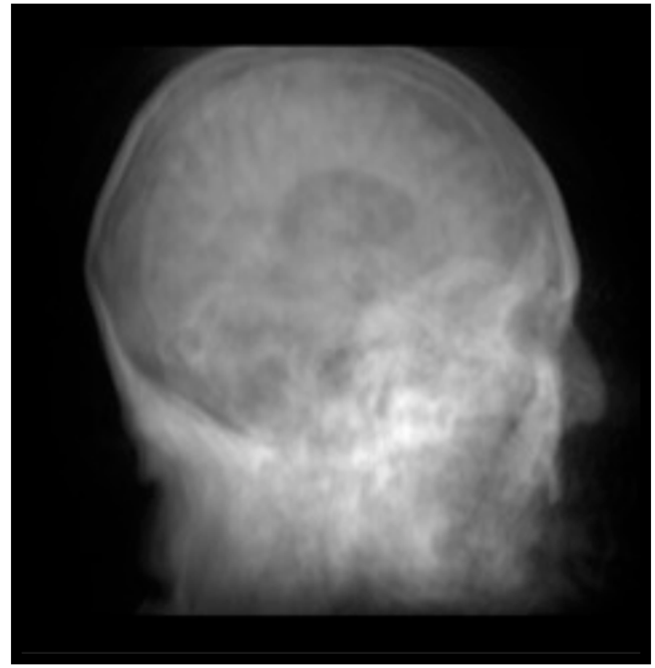




- Extracts iso-surfaces (again!)

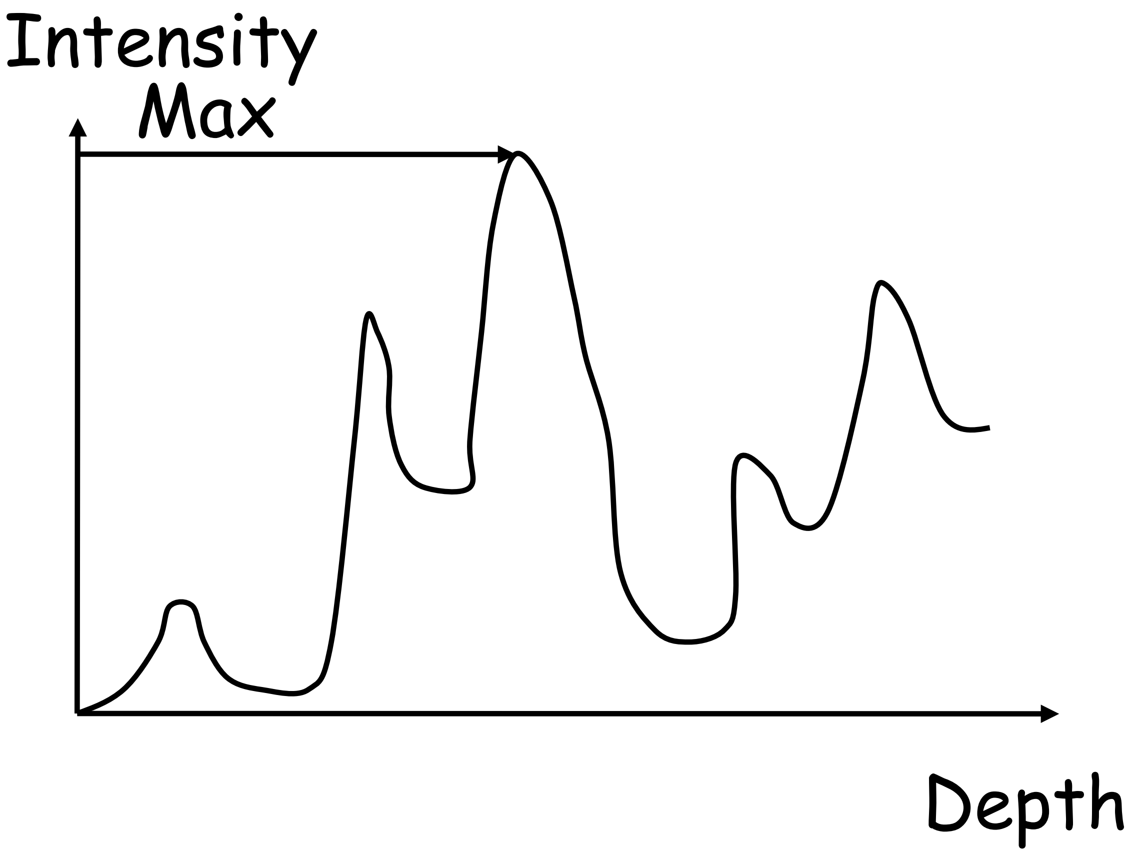
Compositing - Average





Synthetic Reprojection

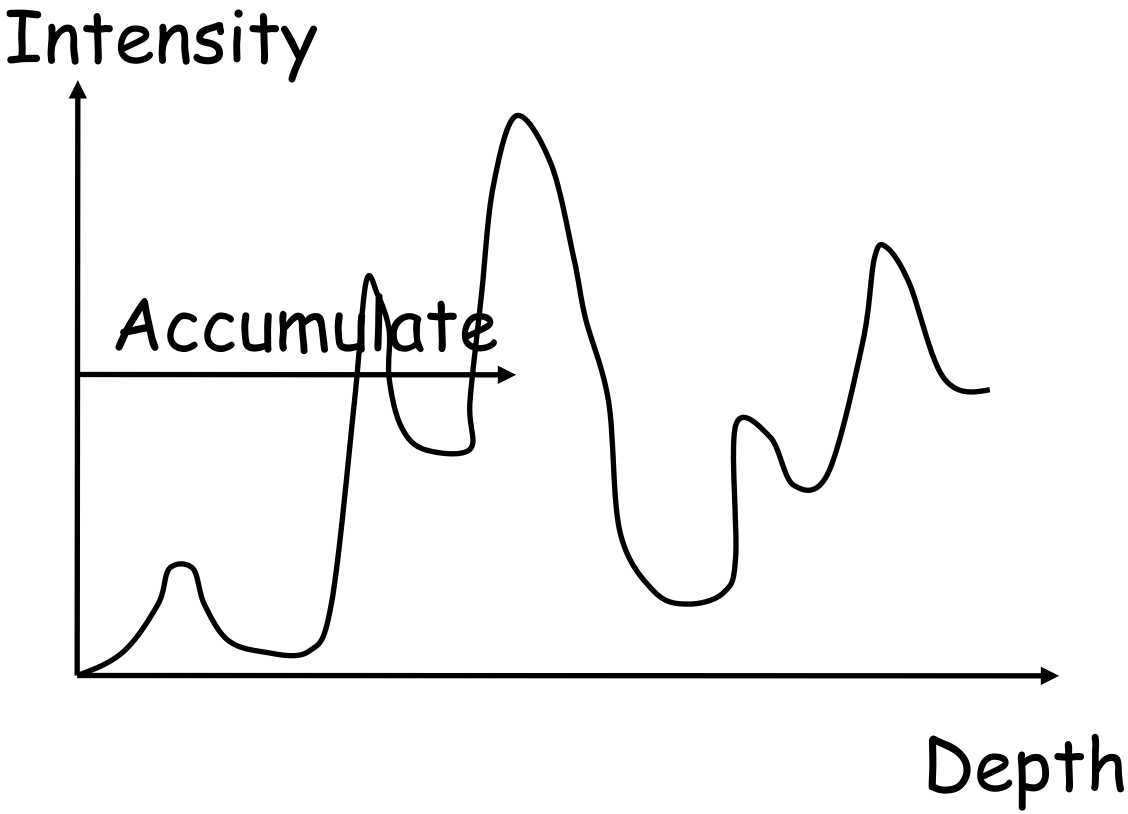
Compositing - MIP



Maximum Intensity Projection used for Magnetic Resonance Angiogram


real time ray tracer
MegaPEG
13.2fps

Compositing - Accumulate



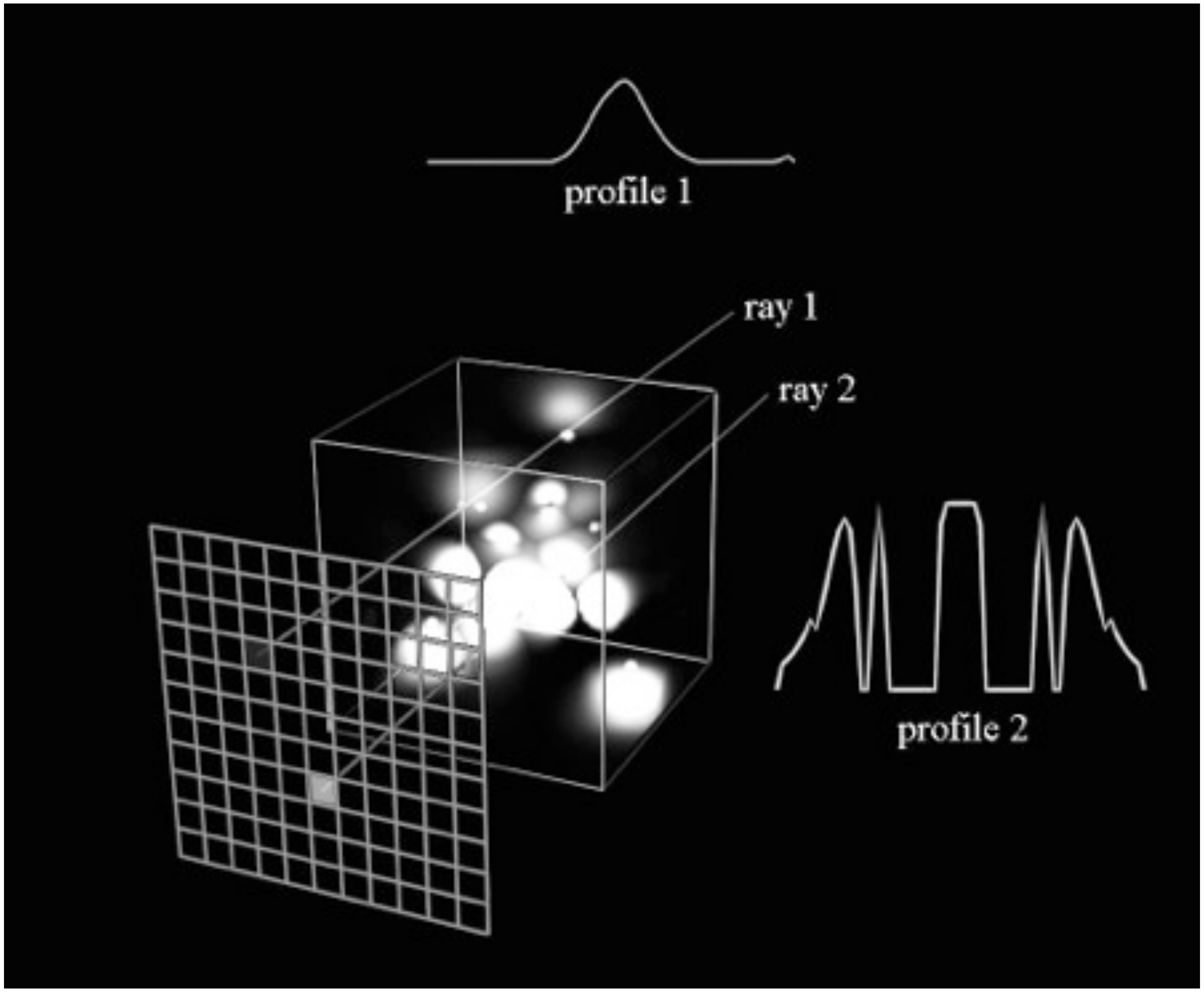
Make transparent layers visible; uses a transfer function for color/opacity

Computational Strategies

- How can the basic ingredients be combined:
  - Image Order
  - Ray casting (many options)
- Object Order
  - splatting, texture-mapping
- Combination (neither)
  - Shear-warp, Fourier

Image Order

- Render image one pixel at a time



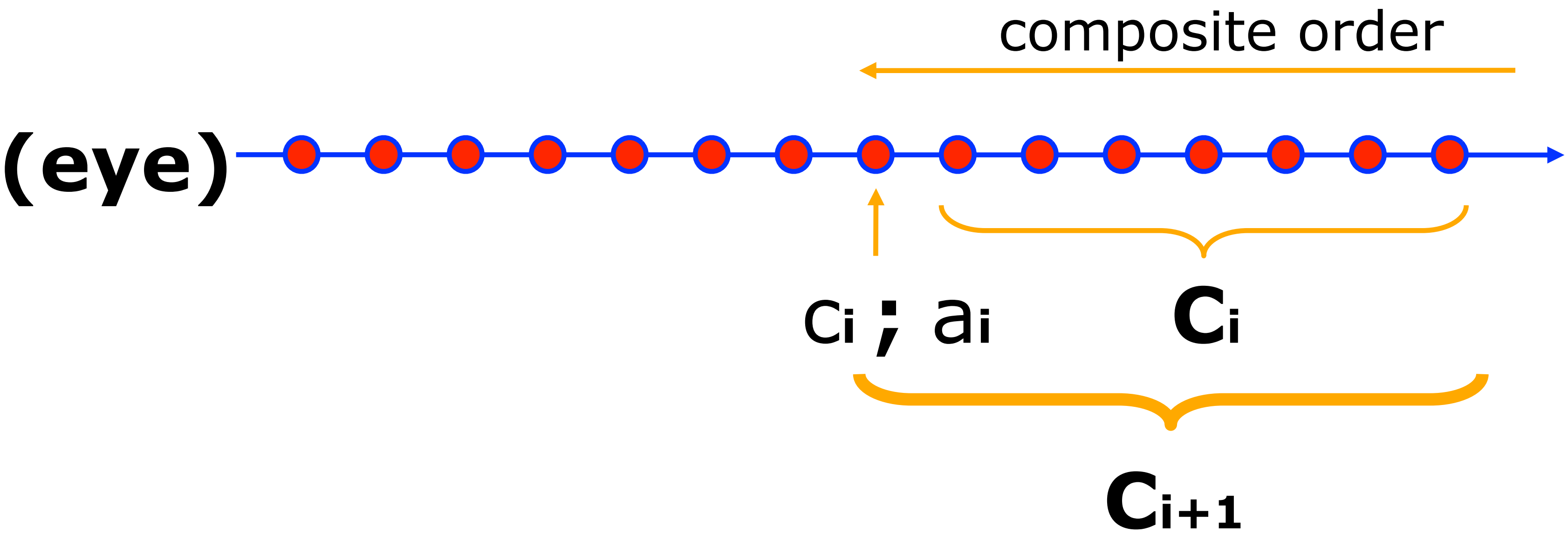
For each pixel ...
- cast ray
- interpolate
- transfer function
- composite

Raycasting

- Back to Front
  - straightforward use of over operator
  - intuitively backwards
- Front to Back
  - intuitively right
  - not simple over operator
  - facilitates early ray termination

Raycasting: compositing

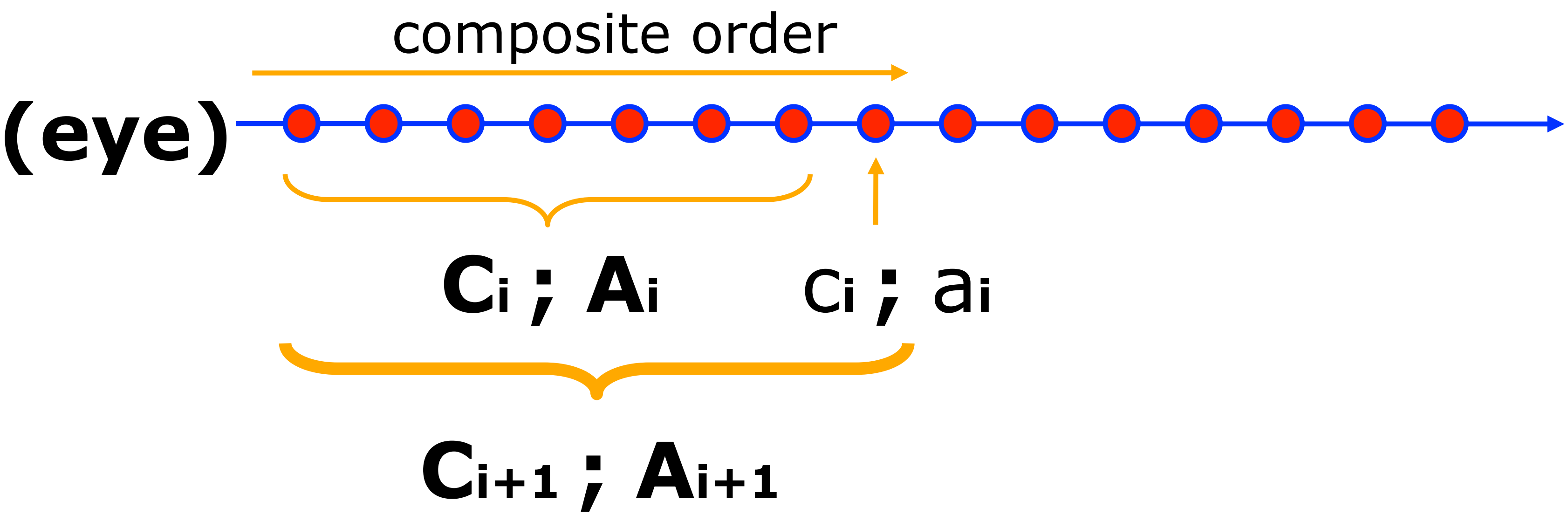
- Back to Front:



$$\boxed{C_{i+1} = a_i * c_i + (1 - a_i) * C_i}$$

Raycasting: compositing

- Front to Back:



$$C_{i+1} = C_i + (1 - A_i)*a_i*c_i$$
$$A_{i+1} = A_i + (1 - A_i)*a_i$$

Raycasting: compositing

- Which is better?

- Front to Back:

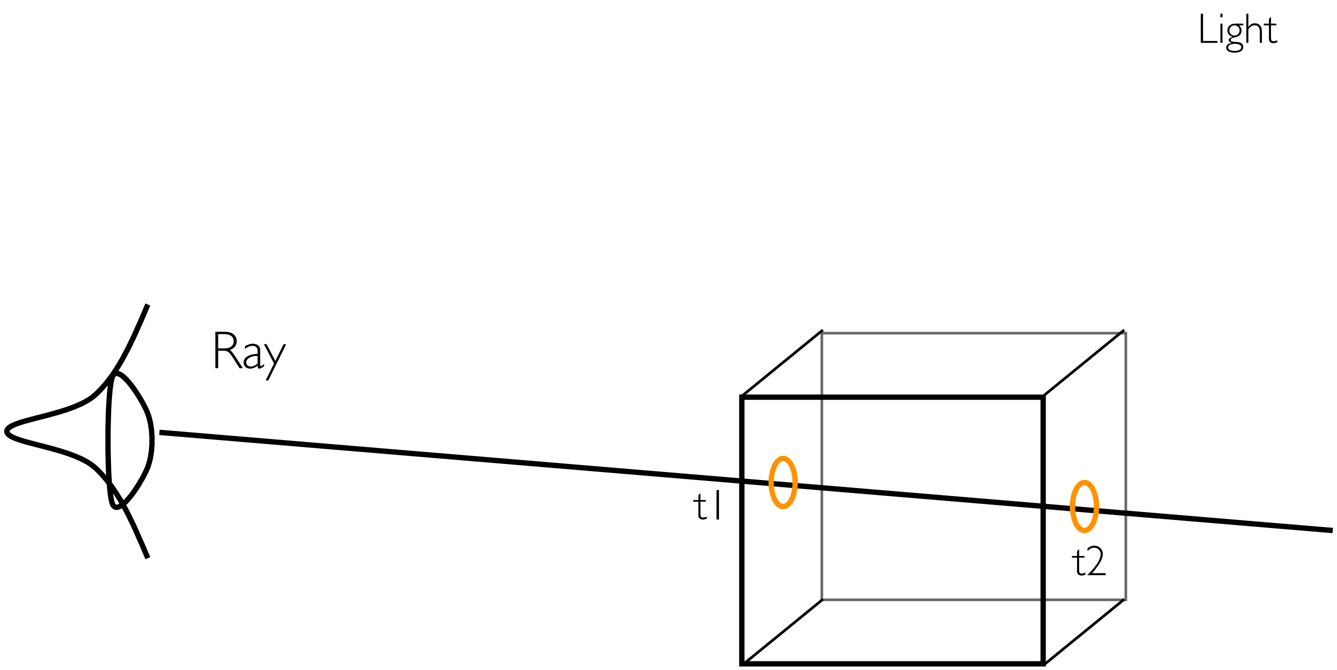
$$C_{i+1} = C_i + (1 - A_i)*a_i*c_i$$
$$A_{i+1} = A_i + (1 - A_i)*a_i$$

- Back to Front:

$$C_{i+1} = a_i*c_i + (1 - a_i)*C_i$$

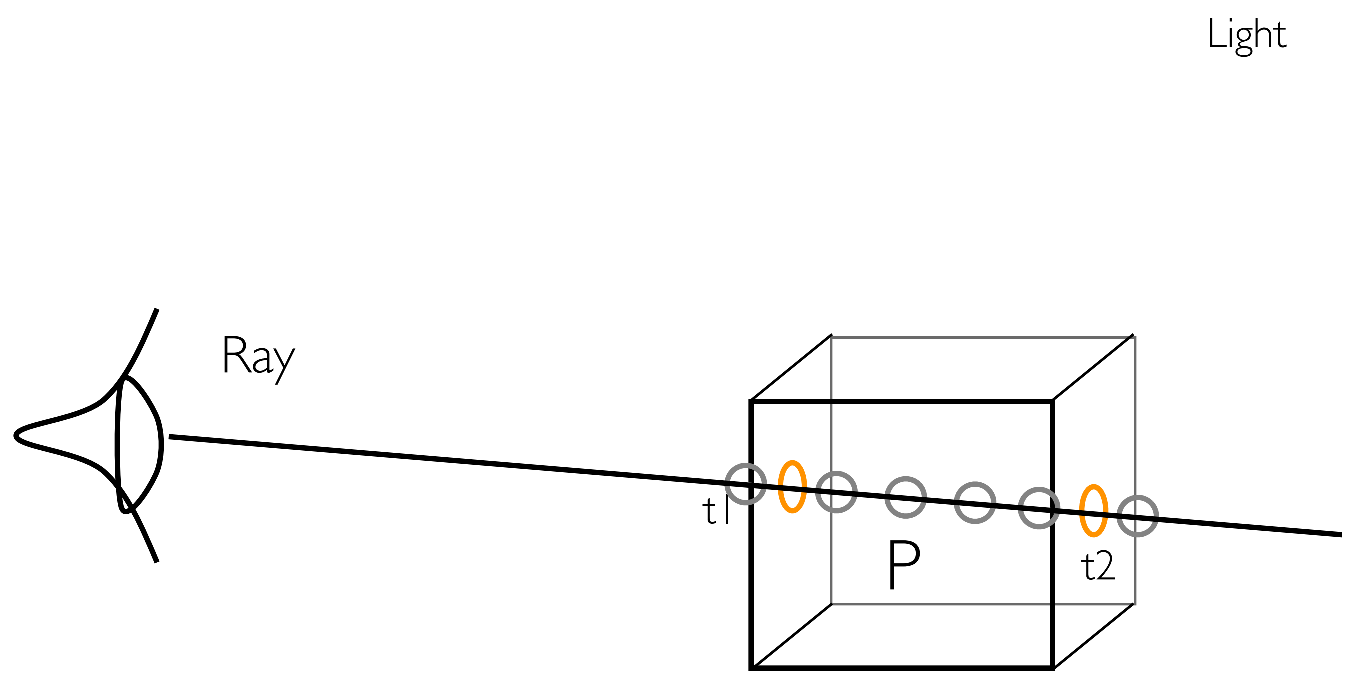
General Components

- Basic diagram



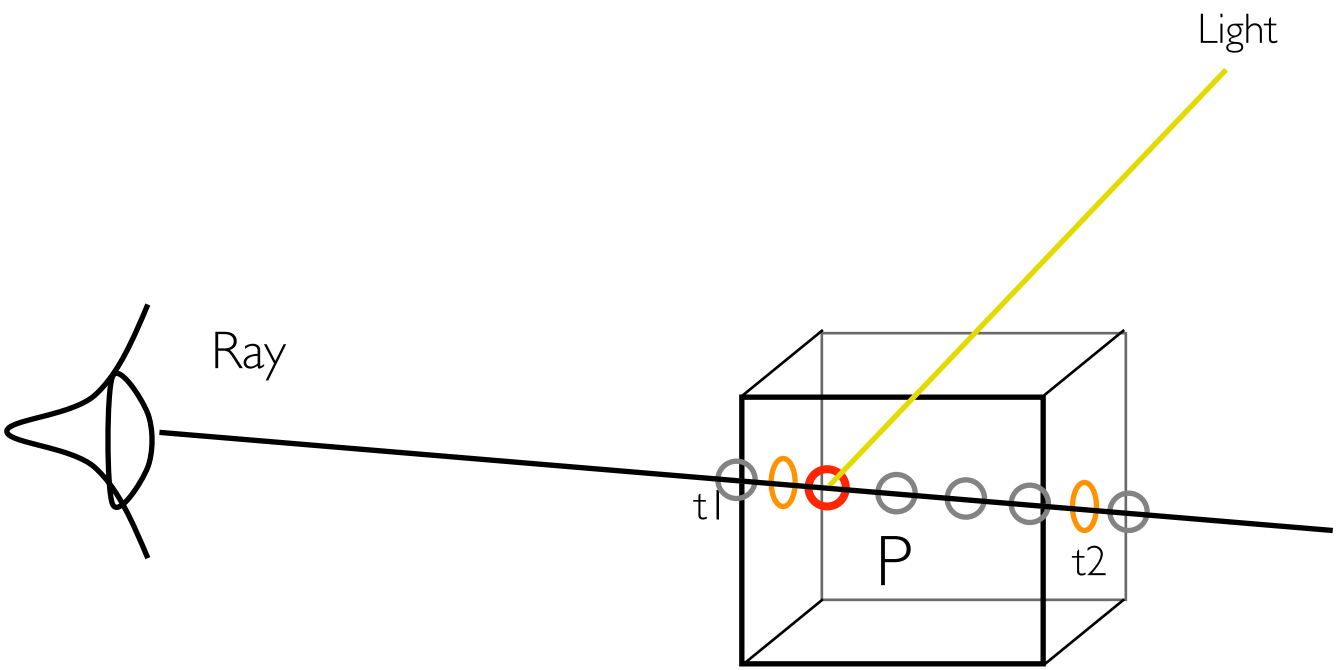
General Components

- Basic diagram



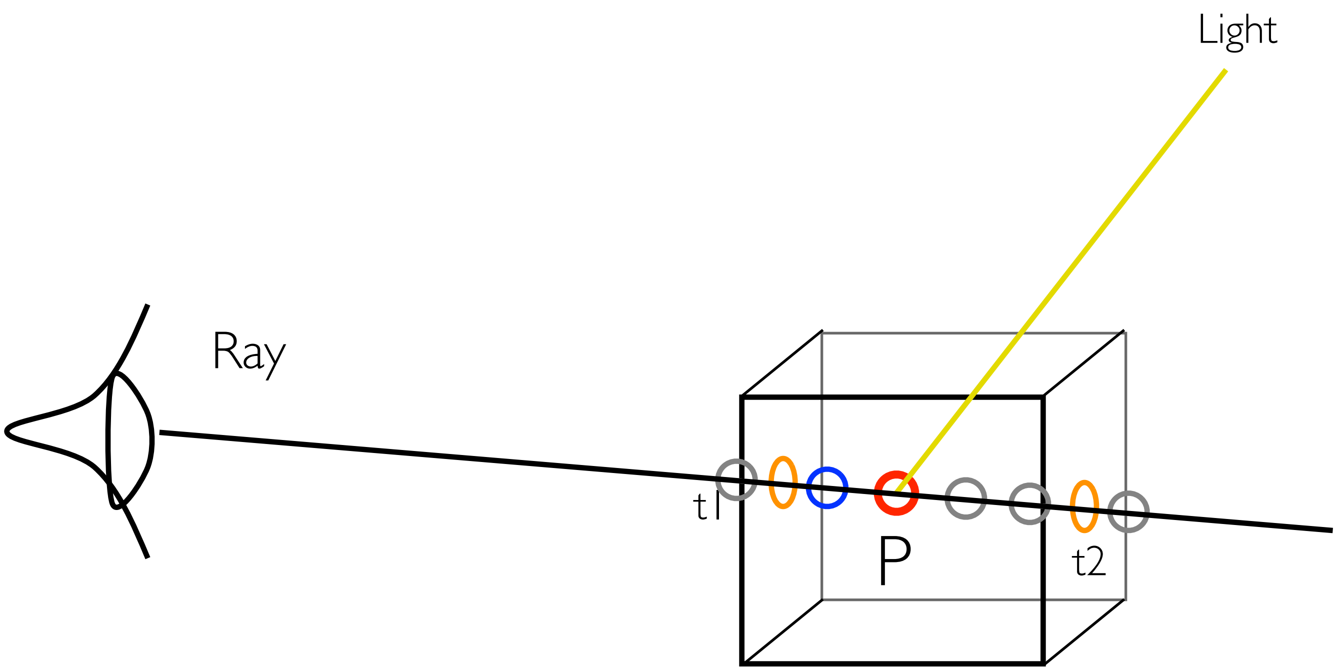
General Components

- Basic diagram



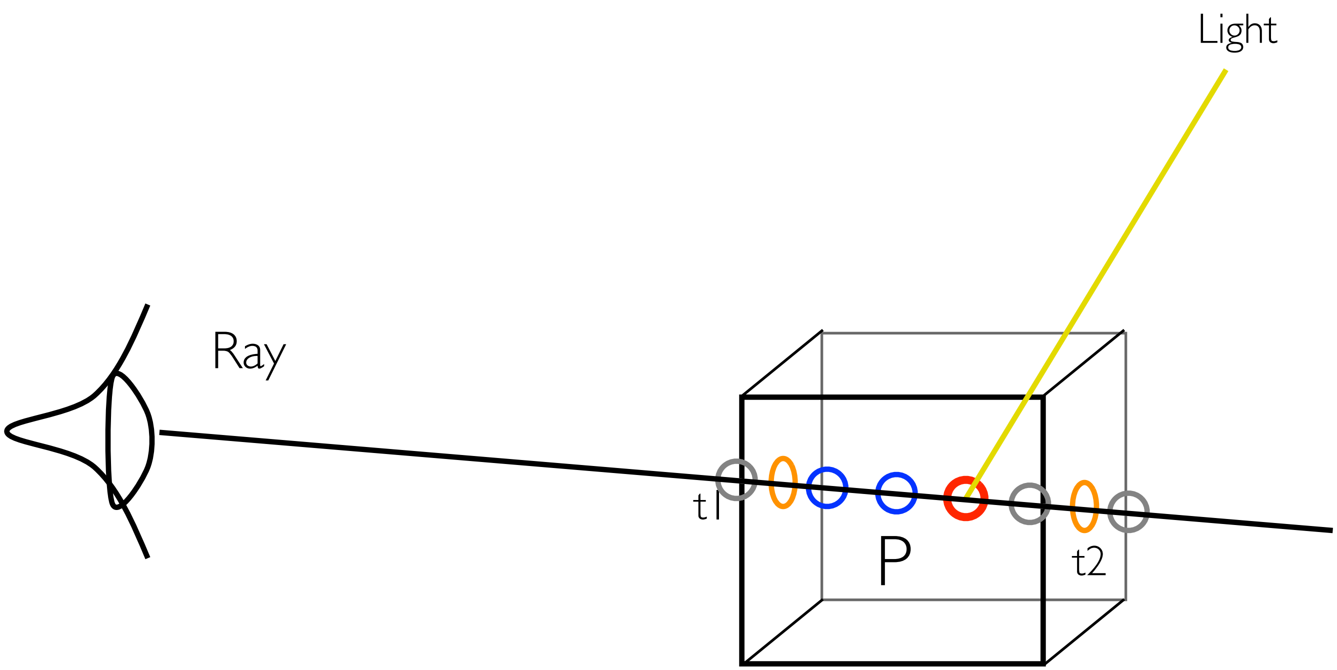
General Components

- Basic diagram



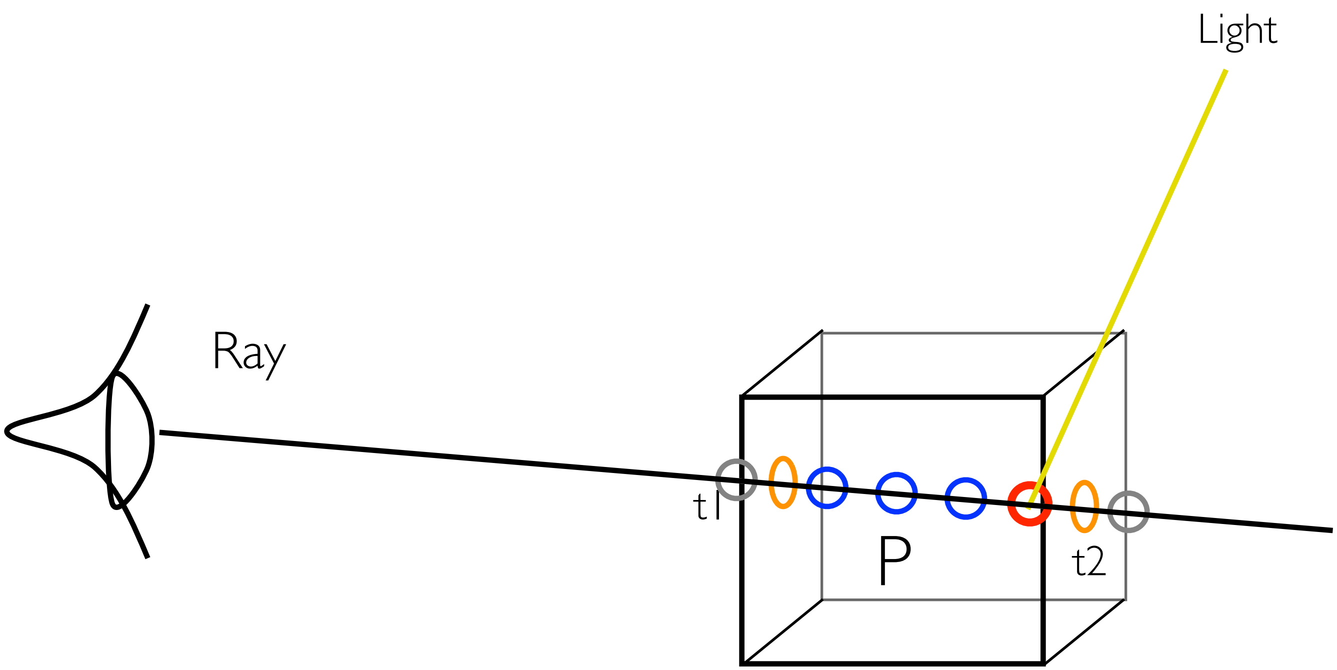
General Components

- Basic diagram



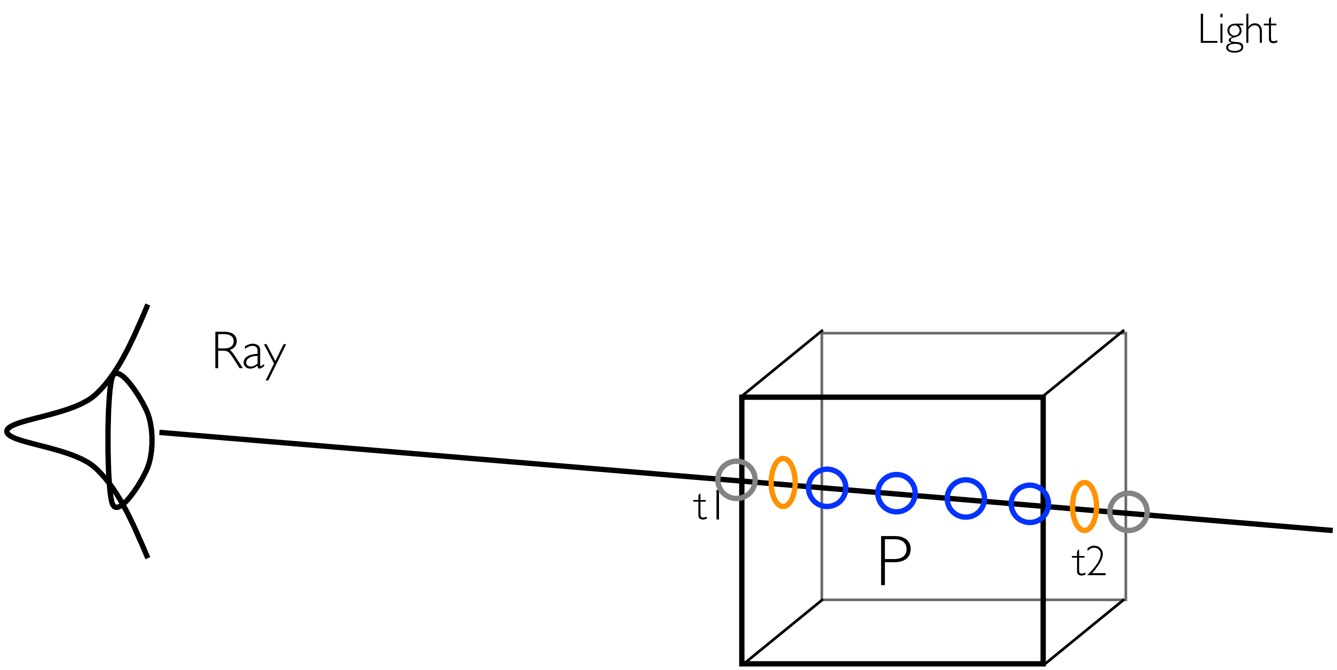
General Components

- Basic diagram



General Components

- Basic diagram



Ray-casting - Highlights

Advantages:

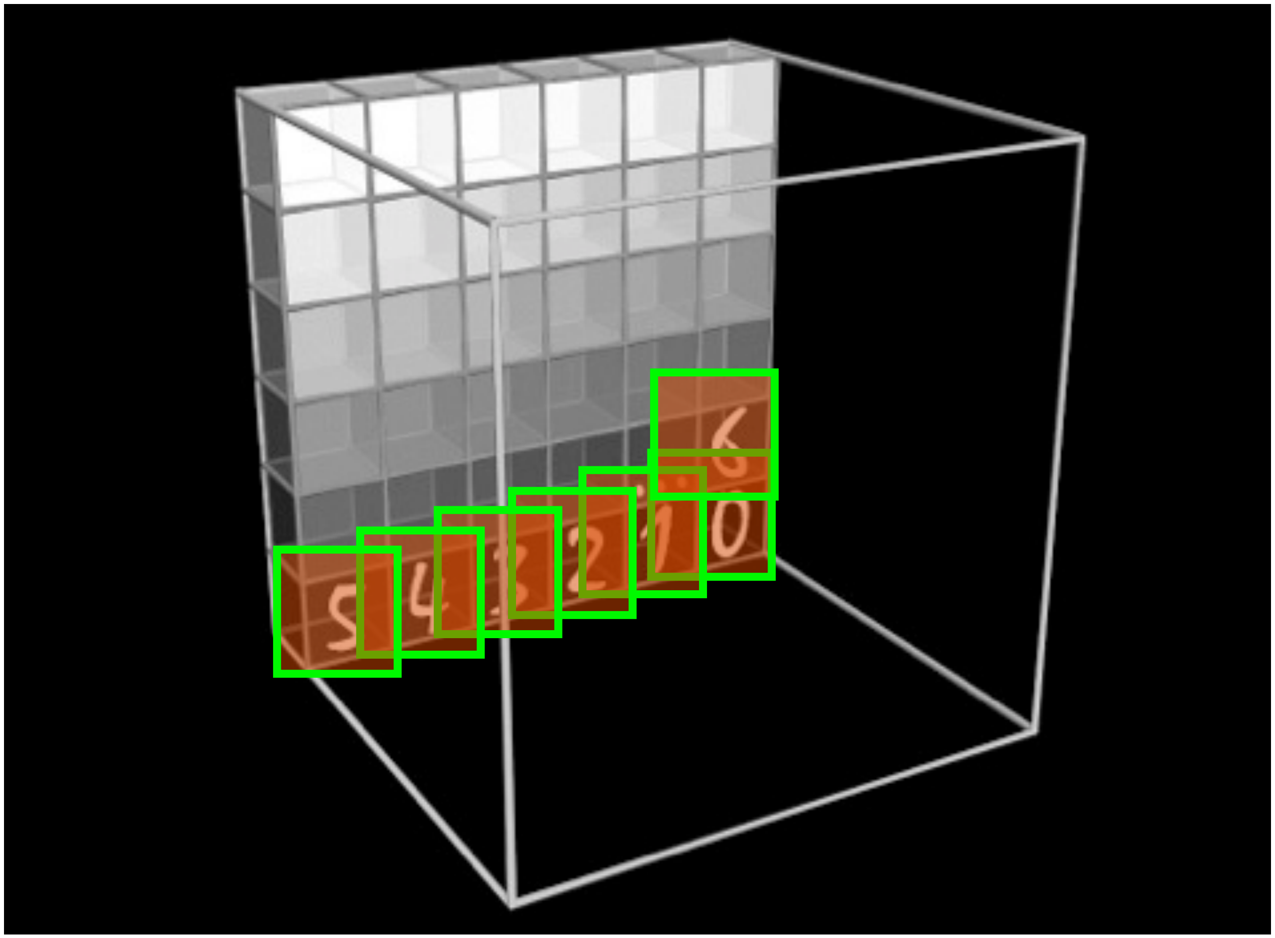
- Simple algorithm
- Inherently parallel
- Can add features (like a ray-tracer)

Disadvantages:

- Slow (lots of rays, lots of samples)
- Must sample densely
- Requires entire data set in memory

Object Order

- Render image one voxel at a time



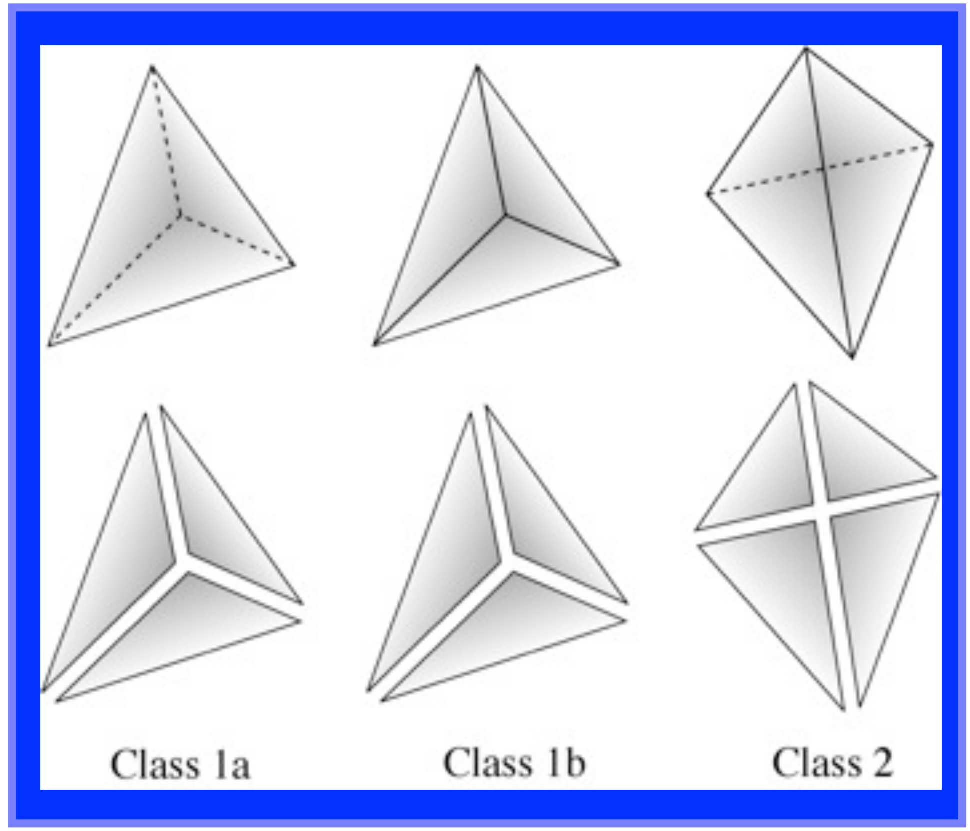
for each voxel ...
- transfer function
- determine image contribution
- composite

Unstructured Volume Rendering

- Given an irregular data set that consists of volumetric cells (typically from FEM simulation)
- How can the volume be displayed accurately?
- Numerous approaches:
  - Ray casting
  - Ray tracing
  - Sweep plane algorithms (e.g. ZSWEEP)
  - PT algorithm of Shirley and Tuchman

PT algorithm of Shirley and Tuchman

- Decompose each cell into tetrahedra
- Sort the tetrahedra in a back to front fashion
- Project each tetrahedron and render its decomposition into 3 or 4 triangles



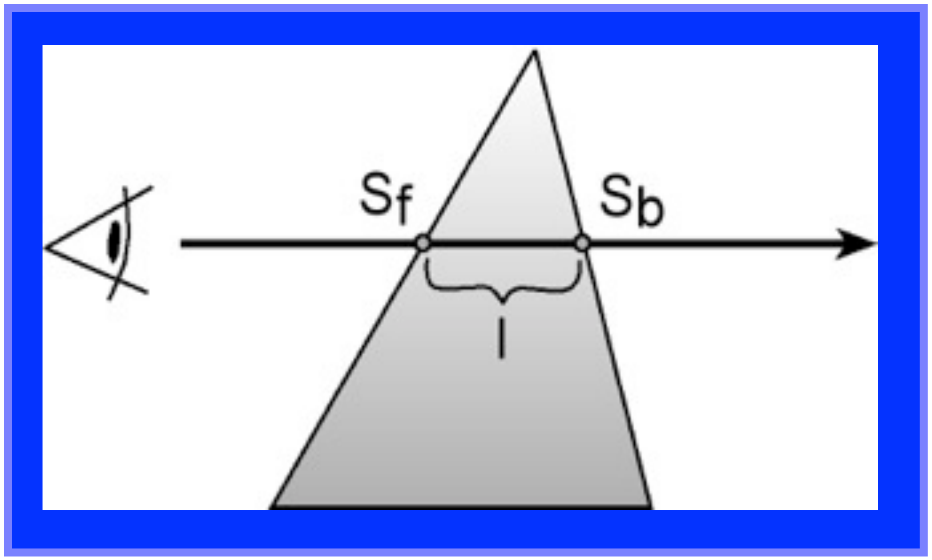
Two different non-degenerate classes of the projected tetrahedra

Volume Density Optical Model

- For the Volume Density Optical Model of Williams et al. the emission and absorption along a light ray is defined by the transfer functions $\kappa(f(x,y,z))$ and $\rho(f(x,y,z))$ with $f(x,y,z)$ being the scalar function
- Usually the transfer functions are given as a linear or piecewise linear function, or as a lookup table

Composing of the Tetrahedra

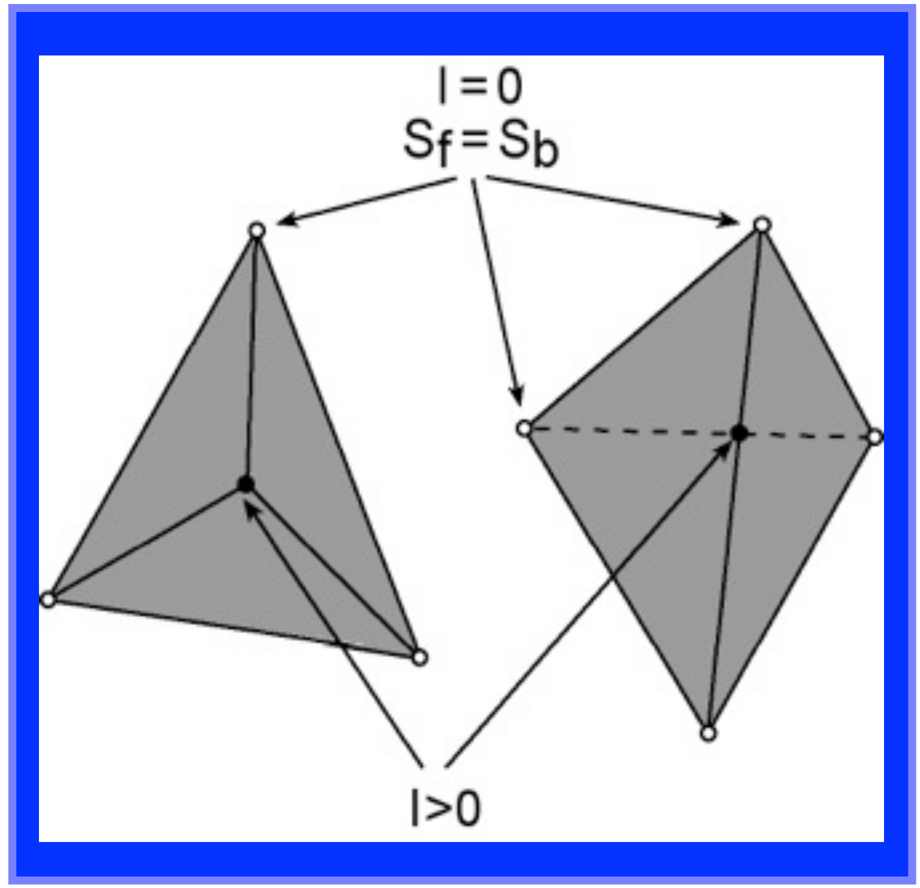
- For each rendered pixel the ray integral of the corresponding ray segment has to be computed



- Observation: The ray integral depends only on S_f, S_b, and l for the Volume Density Optical Model of Williams et al.

3D Texturing Approach

- Compute the three-dimensional ray integral by numerical integration and store the integrated chromaticity and opacity in a 3D texture
- Assign appropriate texture coords (S_f,S_b,l) to the projected vertices of each tetrahedron



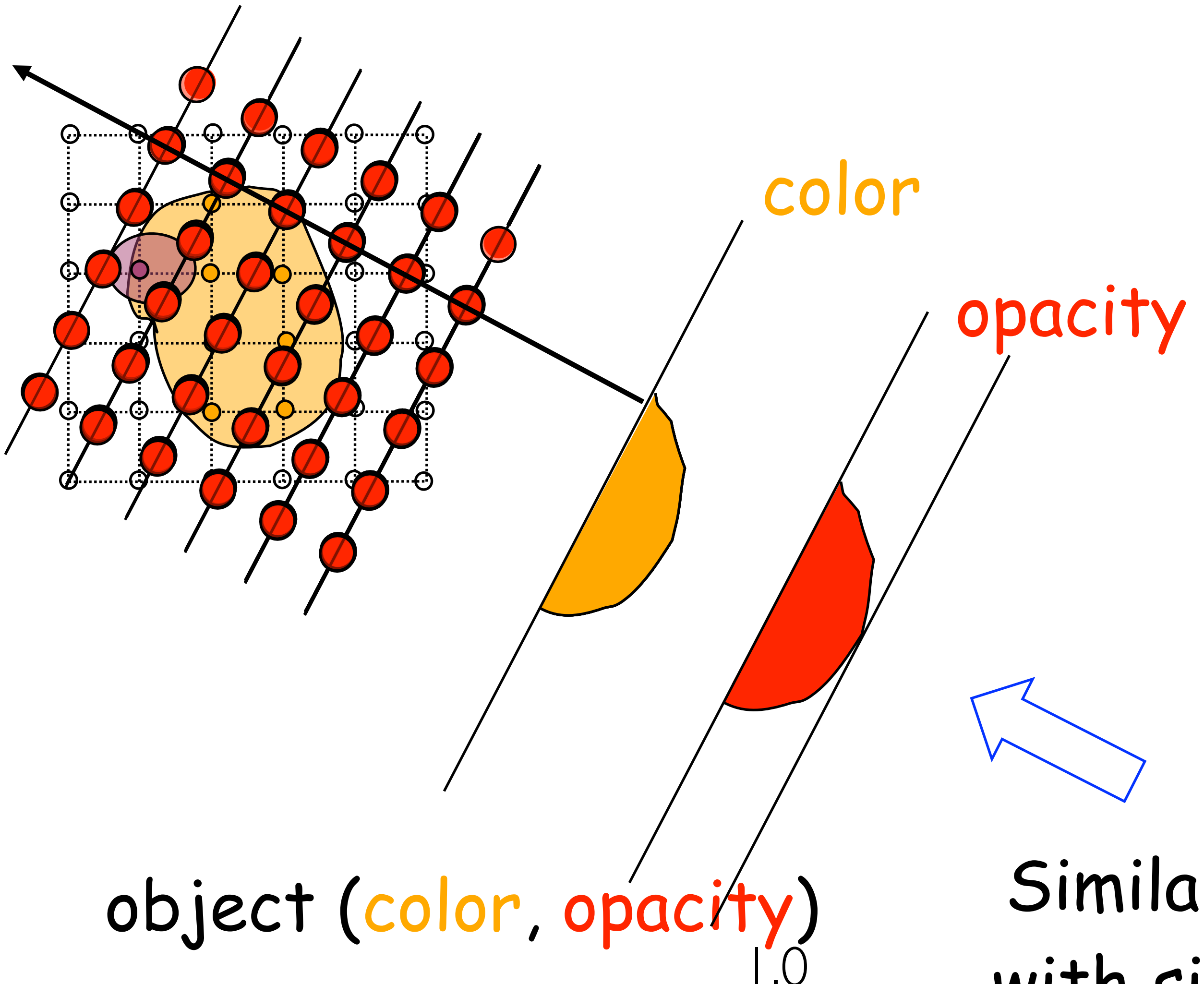
Pros / Cons

- Pros:
  - Object order method
  - Hardware-accelerated approach
  - Per-pixel exact rendering
- Cons:
  - Sort Required
  - Slower than uniform volume rend.

Rendering by Slicing - Why?

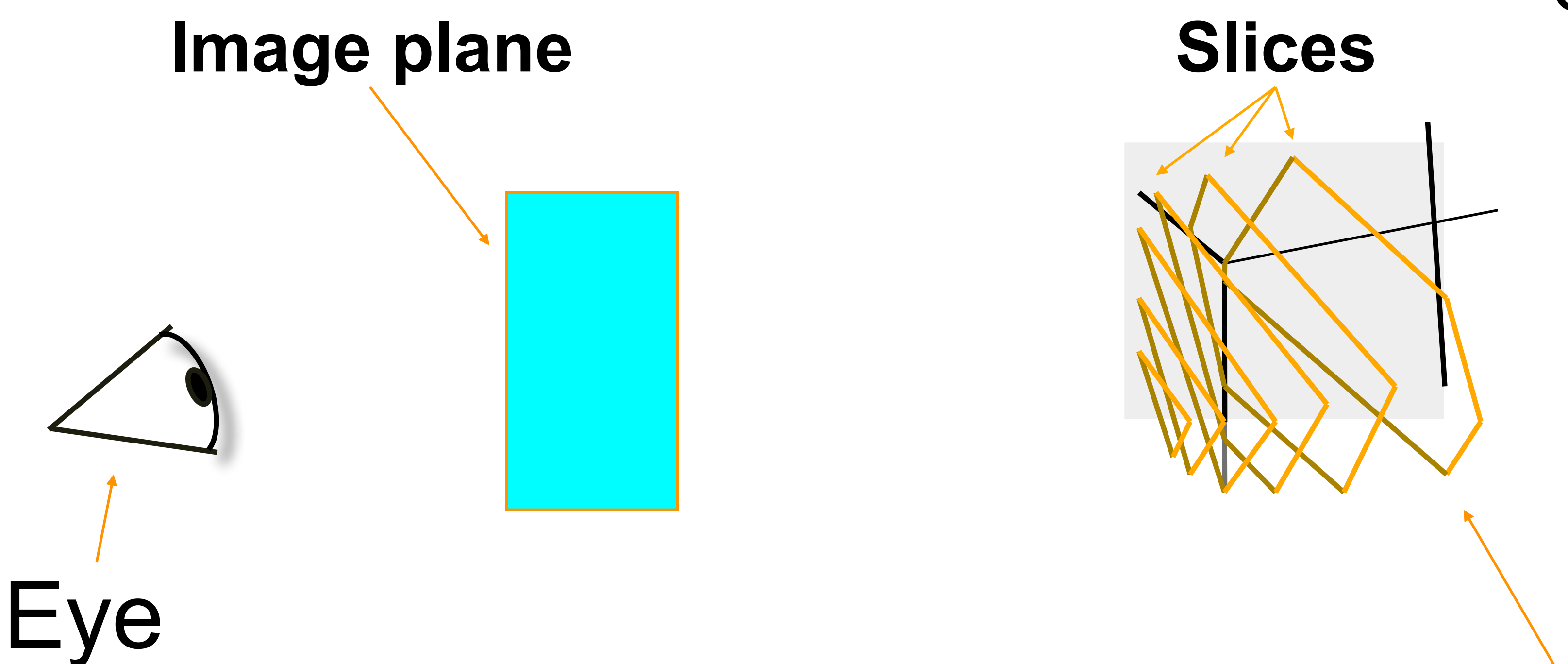
- Store volume in solid texture memory
- Hardware steps:
  - Slicing of the volume (proxy geom)
  - Composite the slices in a BTF order

Slice Based Rendering



Similar to ray-casting with simultaneous rays

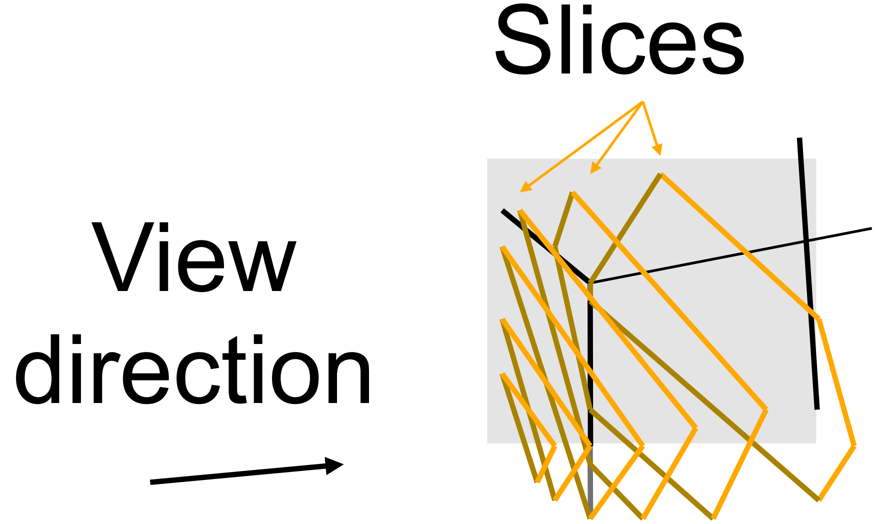
Slice Based Rendering



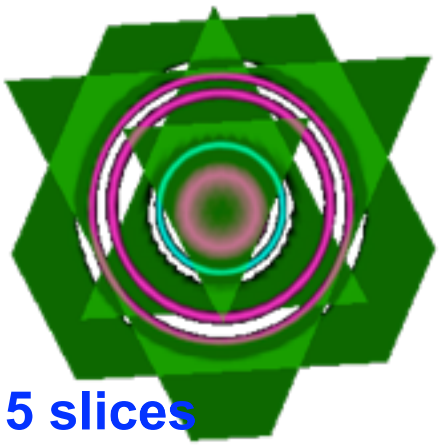
Graphics Hardware Volume Data

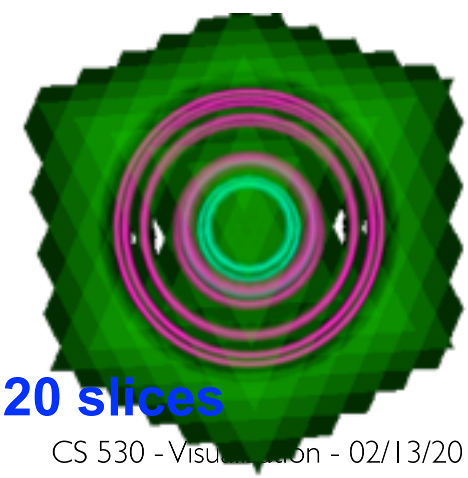
- **Polygons – Proxy geometry**
- **Textures – Data & interpolation**
- **Blending operations – Numerical integration**

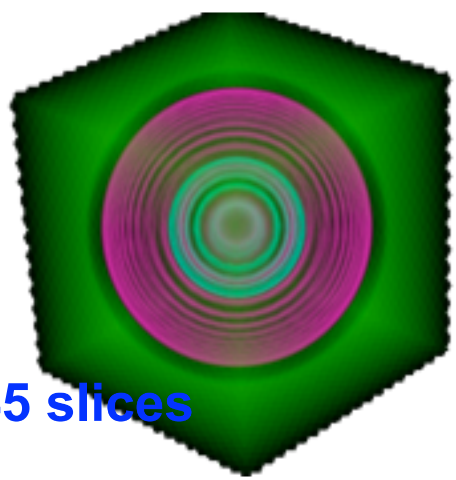
Slice Based Rendering

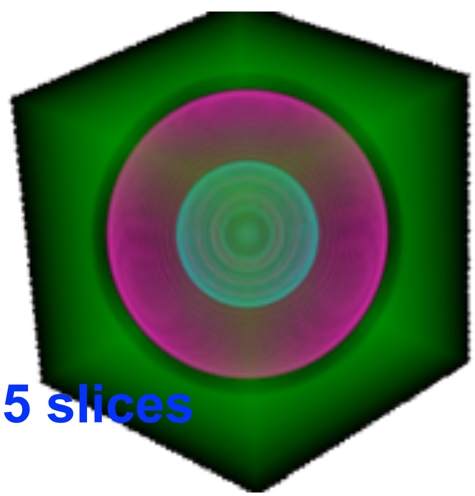


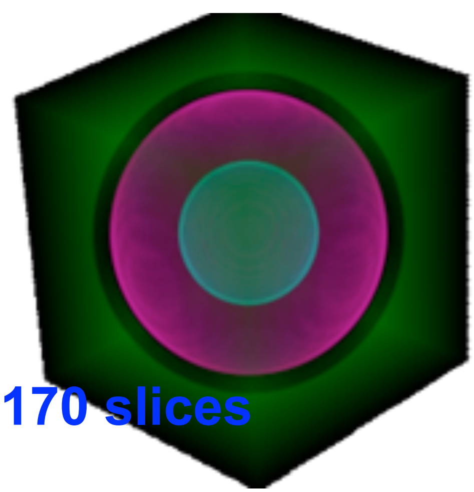

1 slice


5 slices


20 slices


45 slices


85 slices


170 slices

Slice Based Problems?

Does not perform correct

- Illumination
- Accumulation - but can get close

Can not easily add correct illumination and shadowing

- See the Van Gelder paper for their addition for illumination
    - Stored in LUT quantized normal vector directions

Summary

- Volume Ray Casting:
  - Slow (unless implemented on GPU)
  - Back Projection
    - Requires entire data set in memory
  - Can produce reflections, shadows, and complex illumination “relatively” easily
  - Easily parallelizable

Summary

- Hardware Texture Mapping
  - Extremely Fast
  - Can't do correct illumination
  - Approximate accumulation
  - Difficult to add detailed color and texture