6.5 Direct Volume Rendering

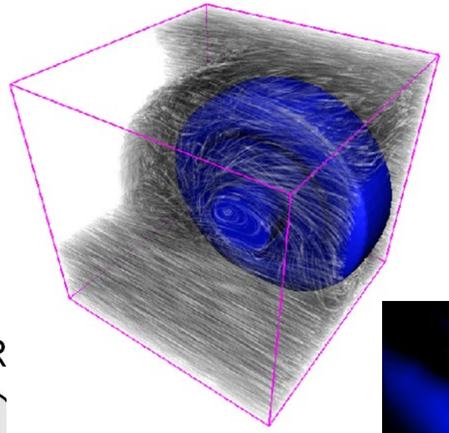
ohm

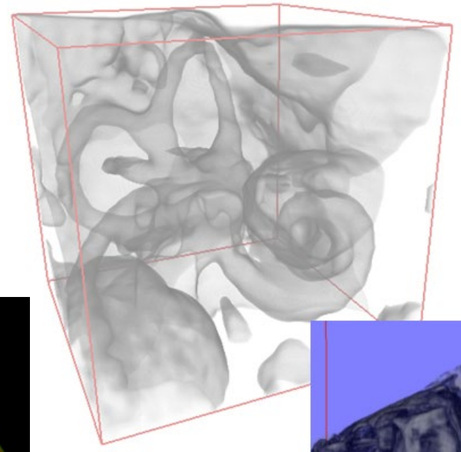
Introduction

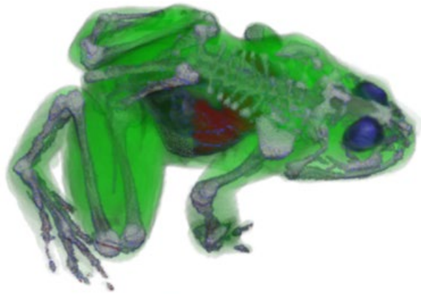
Properties (of direct volume rendering)

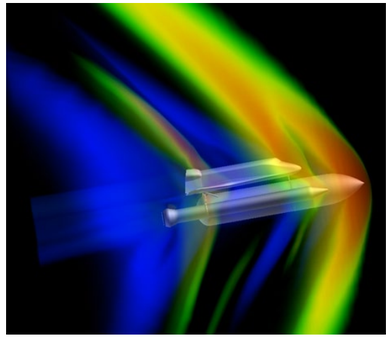
- Directly get a 3D representation of the volume data
- The data is considered to represent a semi-transparent light-emitting medium
  - Hence, also gaseous phenomena can be simulated
- Approaches are based on the laws of physics
  - Emission
  - Absorption
  - Scattering
- The volume data is used as a whole
  - Look inside, see all interior structure

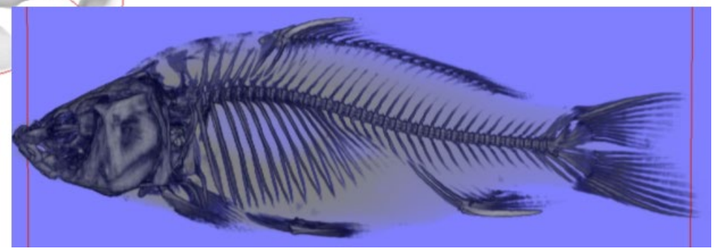
Introduction

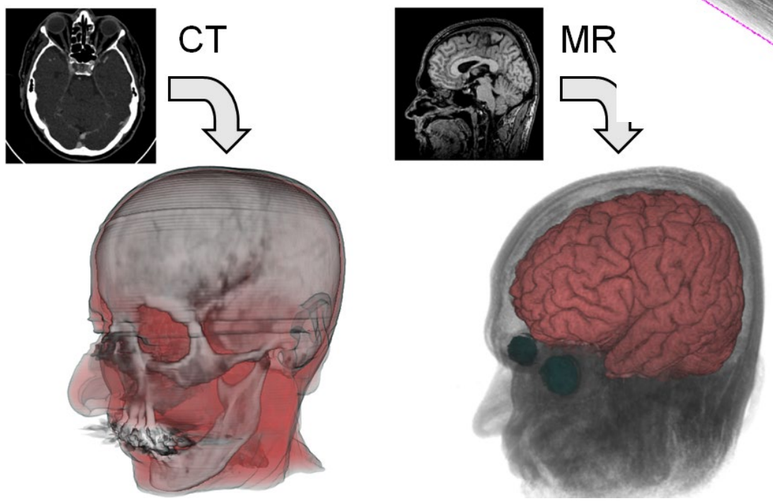


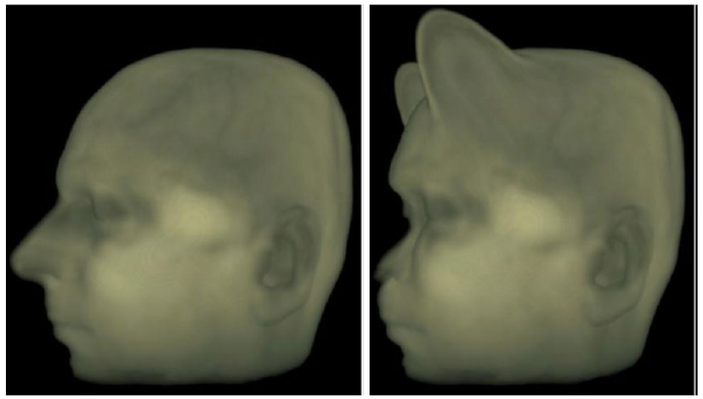


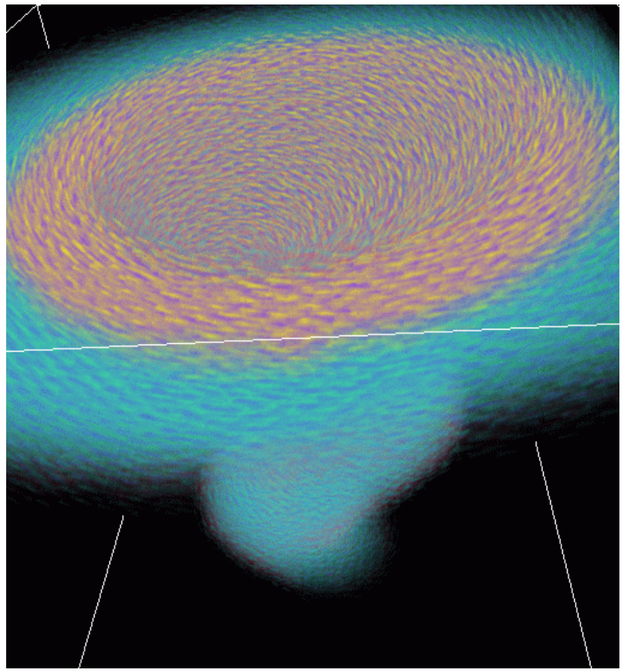


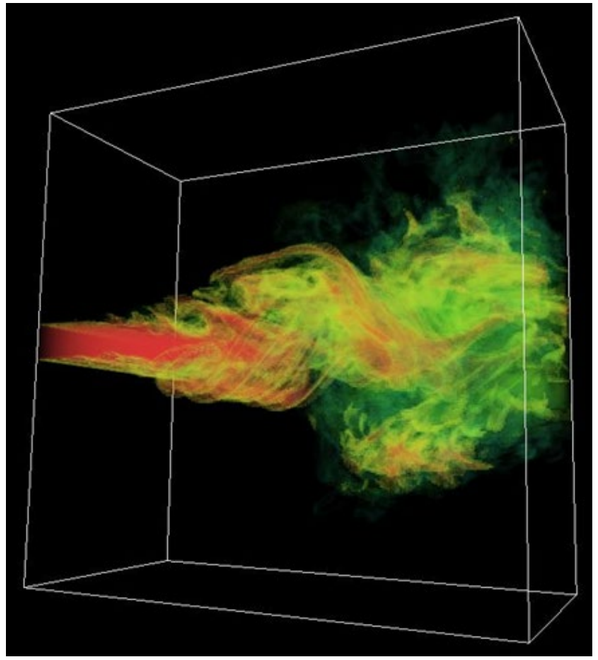












Xue et al, *Fast Dynamic Flow Volume Rendering Using Textured Splats on Modern Graphics Hardware*, Proceedings SPIE EI 2004

Introduction

Advantages

- Use all information (comprehensive representation)
- Show fuzzy and amorph objects
- This approach does not exclude iso-surfaces (combination)
  - However, often there is no need to use them
- Segmentation Visualisieren der Daten -> Segmentation => zuweisen bestimmter Bereiche mit Farben (Werte)
  - Explicit (voxel labeling)
  - Implicit (use color and opacity based on transfer functions)

Introduction

Disadvantages

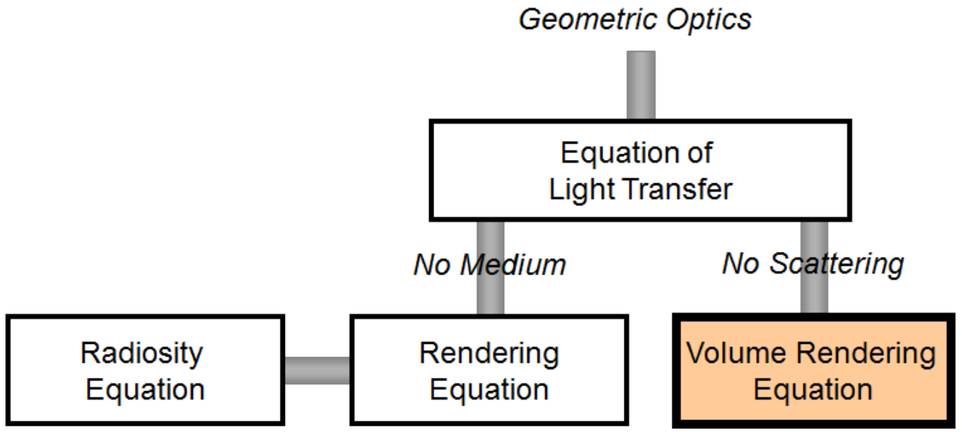
- Interactivity
  - Requires special algorithms and special hardware
- Assignment of color and opacity is difficult
- Amount of data
  - MRI
    - (2562 or 5122) px x 128 slices x 2 bytes = 16 - 64 MB
  - CT
    - 5122 px x (128 up to 800) slices x 2 bytes = 64 - 400 MB
  - Material science
    - 10242 x (128 up to 800) slices x 2 bytes = 256 - 1600 MB
  - If data is time dependent > 1GB

→ Huge amount of interpolations !

6.5.1 Volume Rendering Equation

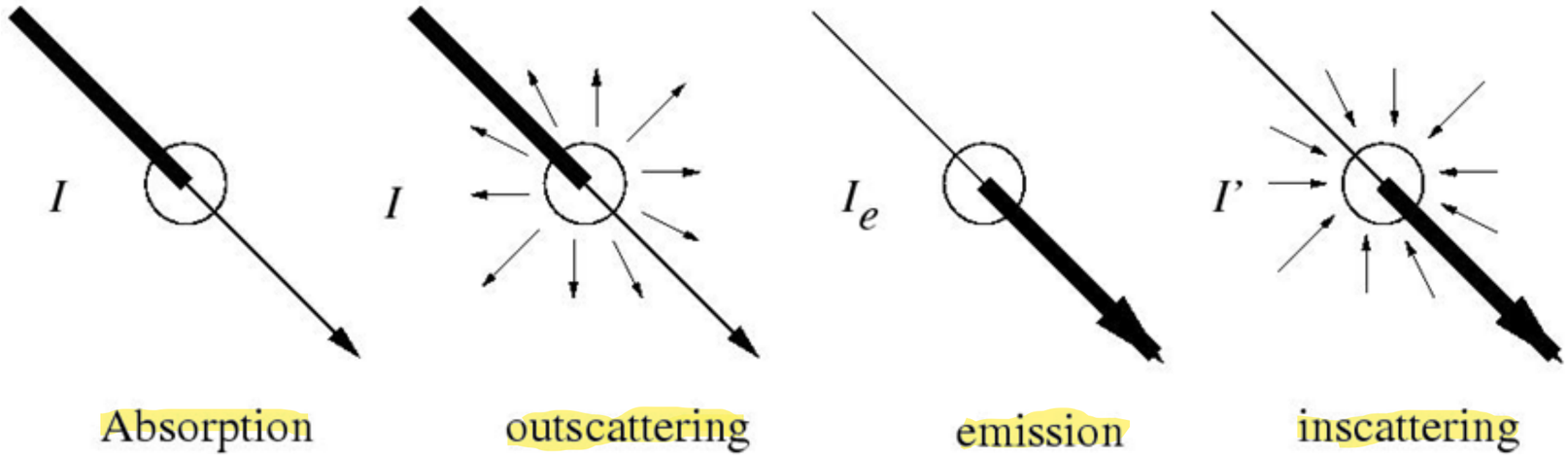
Volume Rendering

no scattering



Light transport

- Contributions to radiation at single position
  - Absorption
  - Emission
  - Scattering



ohm

Volume Rendering

Emission-absorption model

- Special case for most volume rendering
  - Also denoted: density-emitter model [Sabella 1988]
- Idea
  - Volume filled with light-emitting particles
  - Particles described by density function
- Simplifications
  - No scattering
  - Emission coefficient has source term only
  - Absorption coefficient has true absorption only

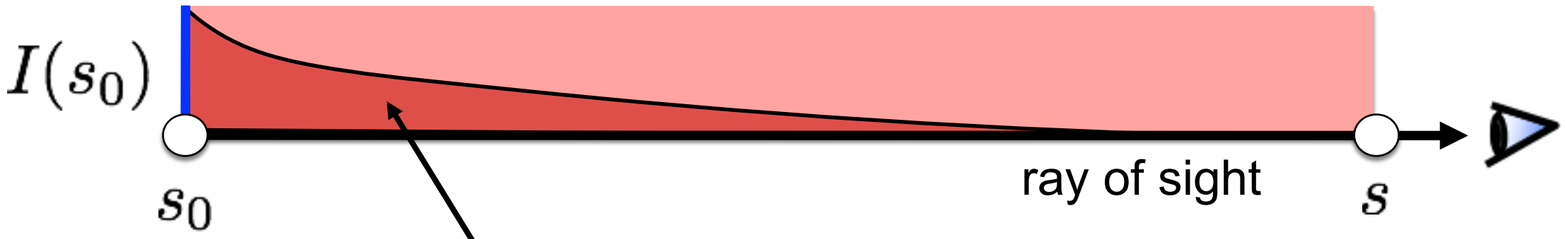
but also absorpe

Volume Rendering

optical depth τ
absorption κ

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

Absorption intensity decays exponentially



initial intensity (emission) at s_o

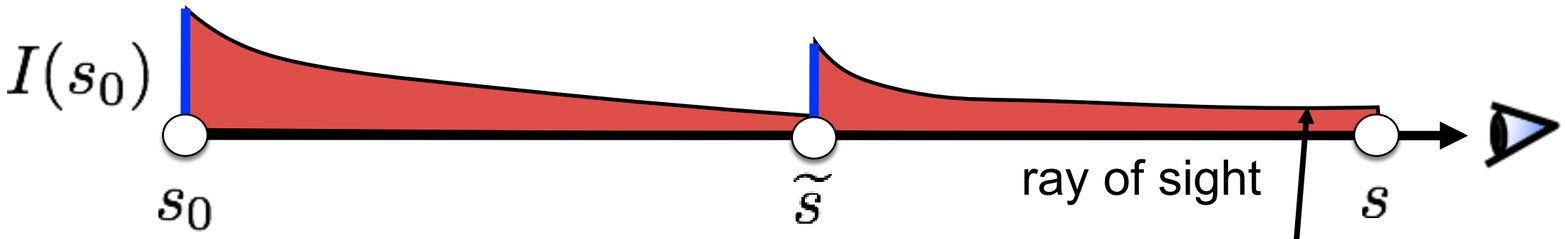
Absorption along distance s_o - s

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

Without absorption, the initial intensity would losslessly arrive at point **s** without *exponential decay*

Volume Rendering

Summe der Samples ist die Intensität



At each point $\tilde{s}$ along the ray of sight additional light is emitted

Active emission at s

absorption along distance s~s

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

Volume Rendering

Discretization

$$I(s_k) = I(s_{k-1})\, e^{-\tau(s_{k-1}, s_k)} + \int_{s_{k-1}}^{s_k} q(s)\, e^{-\tau(s, s_k)}\, ds$$

- Define
  - Transparency part

$$\theta_k = e^{-\tau(s_{k-1}, s_k)}$$

  - Emission part

$$b_k = \int_{s_{k-1}}^{s_k} q(s)\, e^{-\tau(s, s_k)}\, ds$$

  - Discretized volume integral

$$I(s_n) = I(s_{n-1})\, \theta_n + b_n = \sum_{k=0}^{n} \left(b_k \prod_{j=k+1}^{n} \theta_j \right)$$

Summe über alle Emissionen gewichtet über Absorptionen (Transparenz)

Volume Rendering

- Emission Absorption Model

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

- Discretization

$$\tilde{C} = \sum_{i=0}^{|T/\Delta t|} C_i \prod_{j=0}^{i-1} (1 - A_j)$$

over operator with opacity

- Recursive computation

$$C'_i = C_i + (1 - A_i)\, C'_{i-1}$$

Radiant energy observed at position ***i***

Radiant energy emitted at position ***i***

Absorption at position ***i***

Radiant energy observed at position ***i–1***

ohm

Volume Rendering

Compositing - Evaluation of the volume rendering equation

- Back to front

$$C'_i = C_i + (1 - A_i)C'_{i-1}$$

- Front to back

$$C'_i = C'_{i-1} + (1 - A'_{i-1})C_i$$

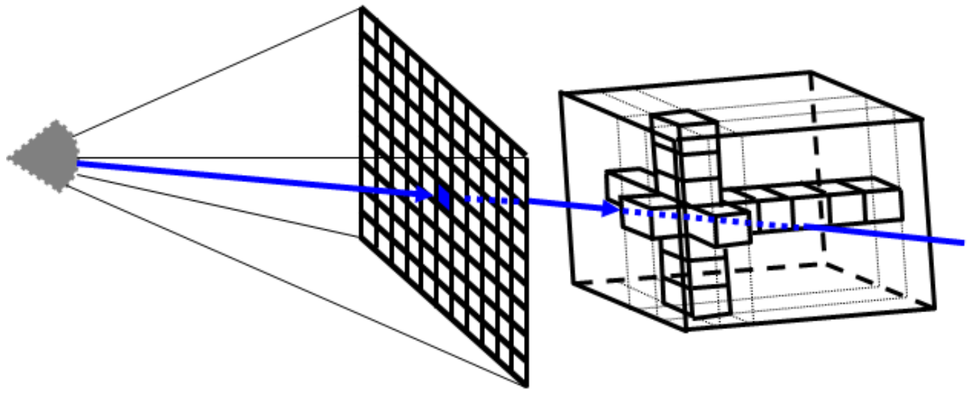
$$A'_i = A'_{i-1} + (1 - A'_{i-1})A_i$$

6.5.2 Direct Volume Rendering

Main Strategies

Image space / image order algorithms

- Performed pixel by pixel of resulting image



```
for (each pixel on image plane)
   for (each sample point on viewing ray)
    compute contribution to pixel (color, opacity)
```

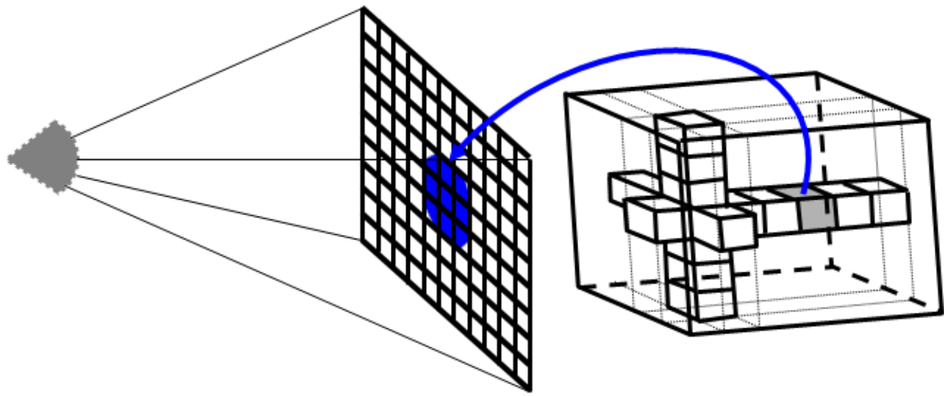
- Example: ray casting

Main Strategies

Object space / object order algorithms

- Performed voxel by voxel of volume data

Projizieren Objekt auf die Bildfläche



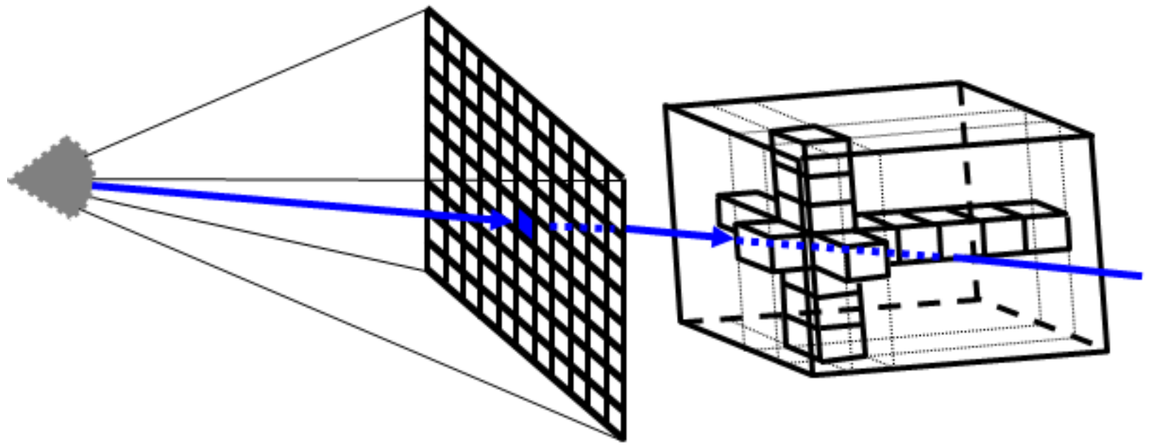
```
for (each voxel in volume data)
   for (each pixel of image plane projected onto)
      compute contribution to pixel (distribute info!)
```

- Example: projection

Raycasting

Raycasting

- Characteristics
  - Similar to ray tracing in surface-based graphics
  - In volume rendering, only primary rays are used (speedup)
    - Hence: ray casting
  - Natural image order technique



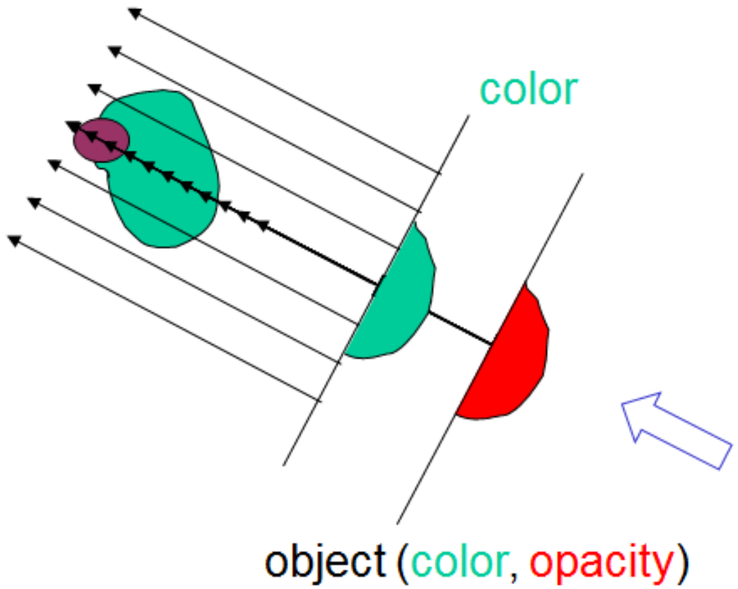
ohm

Raycasting

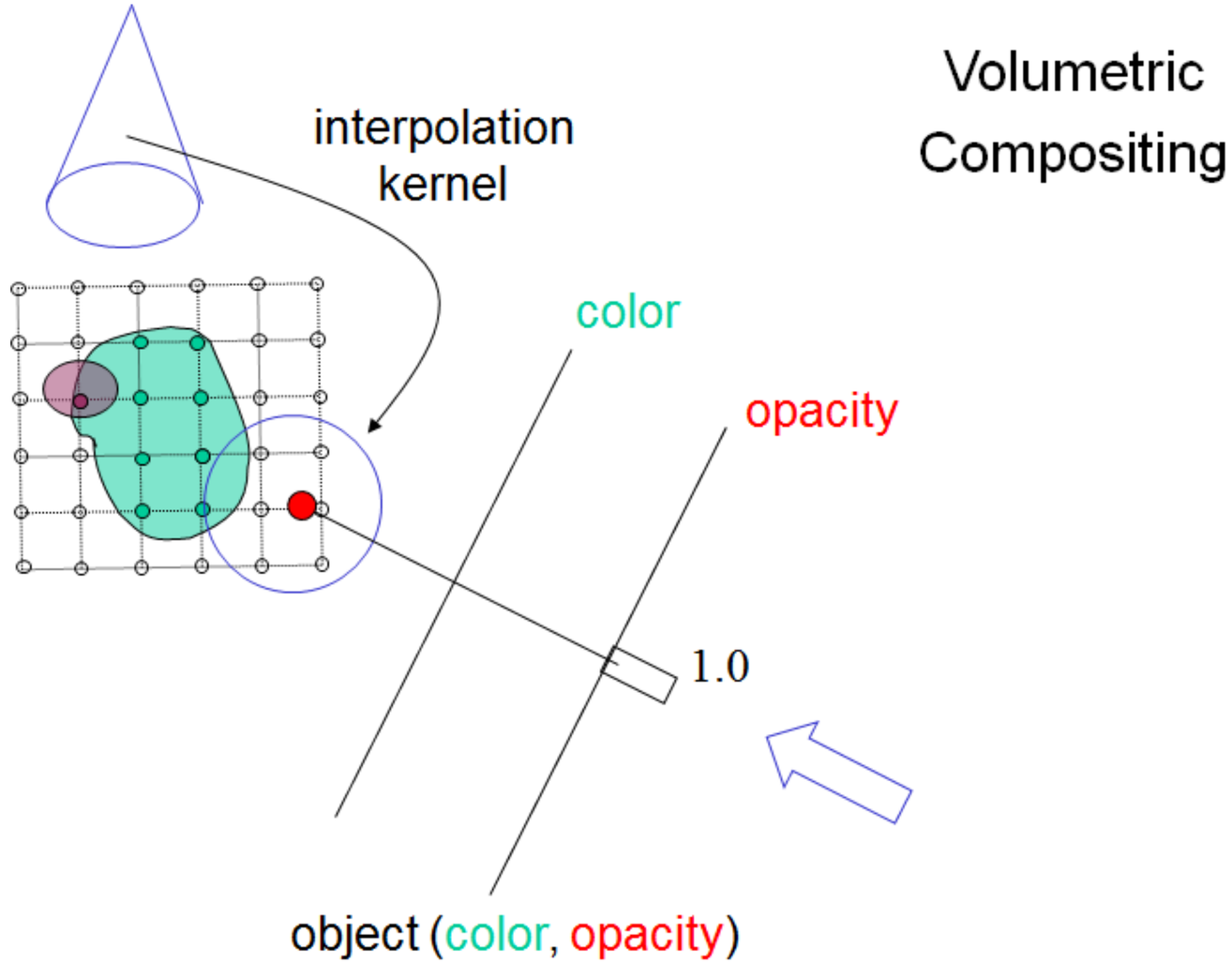
- Approach
  - Cast a ray into the volume
    - Sampling at certain intervals
    - Often equidistant sampling
      - More sophisticated sampling schemes exist
  - At each sampling location
    - Interpolation from voxel grid
      - Nearest neighbor
      - Trilinear
      - Or more sophisticated (Gaussian, cubic spline)

Raycasting

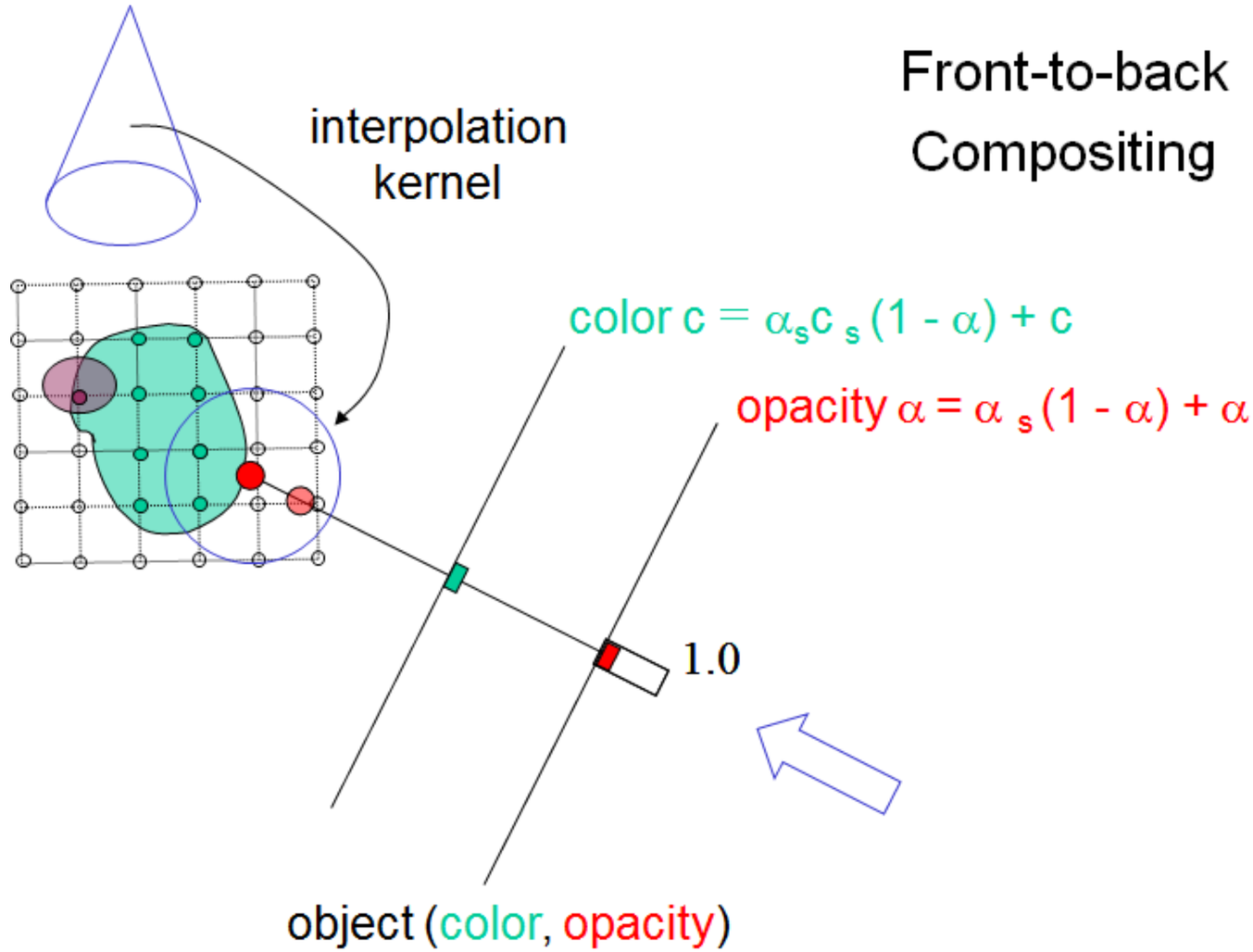
- Volumetric ray integration (*compositing*)
  - Tracing of rays
  - Accumulation of color & opacity along ray



Raycasting

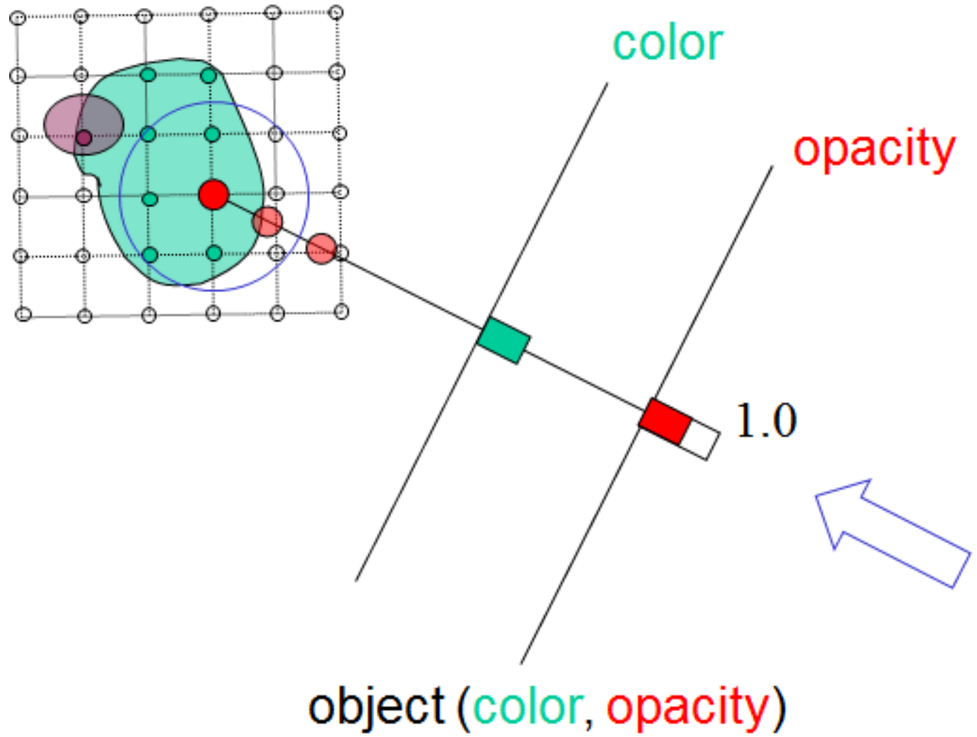

Volumetric
Compositing
interpolation
kernel
color
opacity
1.0
object (color, opacity)

Raycasting


interpolation
kernel
Front-to-back
Compositing
color c = $\alpha_s c_s (1 - \alpha) + c$
opacity $\alpha = \alpha_s (1 - \alpha) + \alpha$
1.0
object (color, opacity)

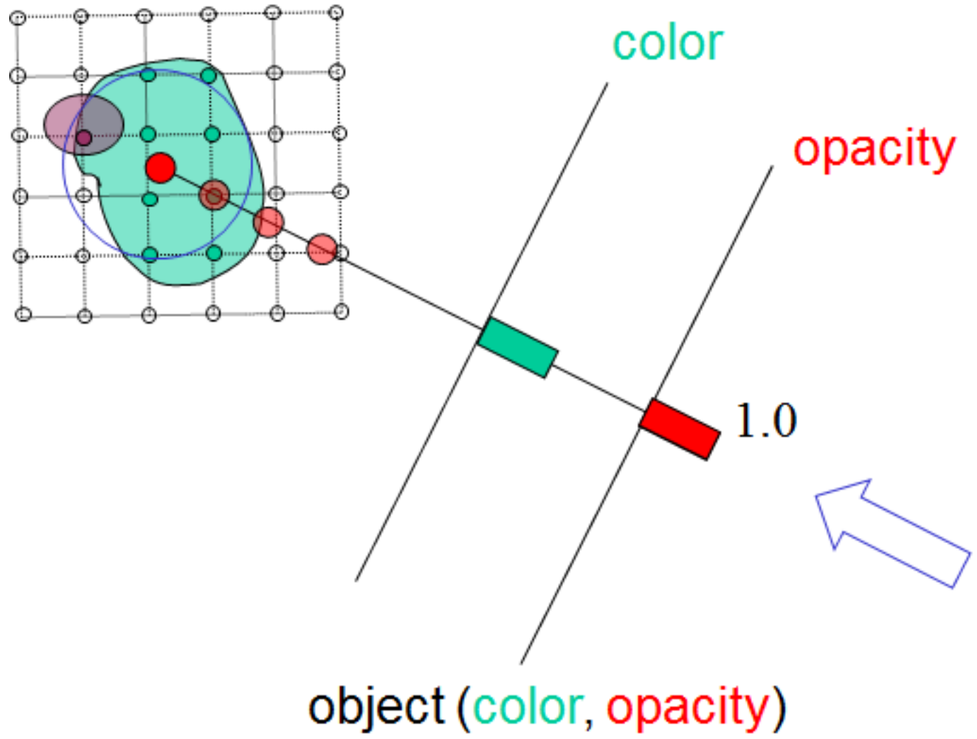
Raycasting

Front-to-back
Compositing

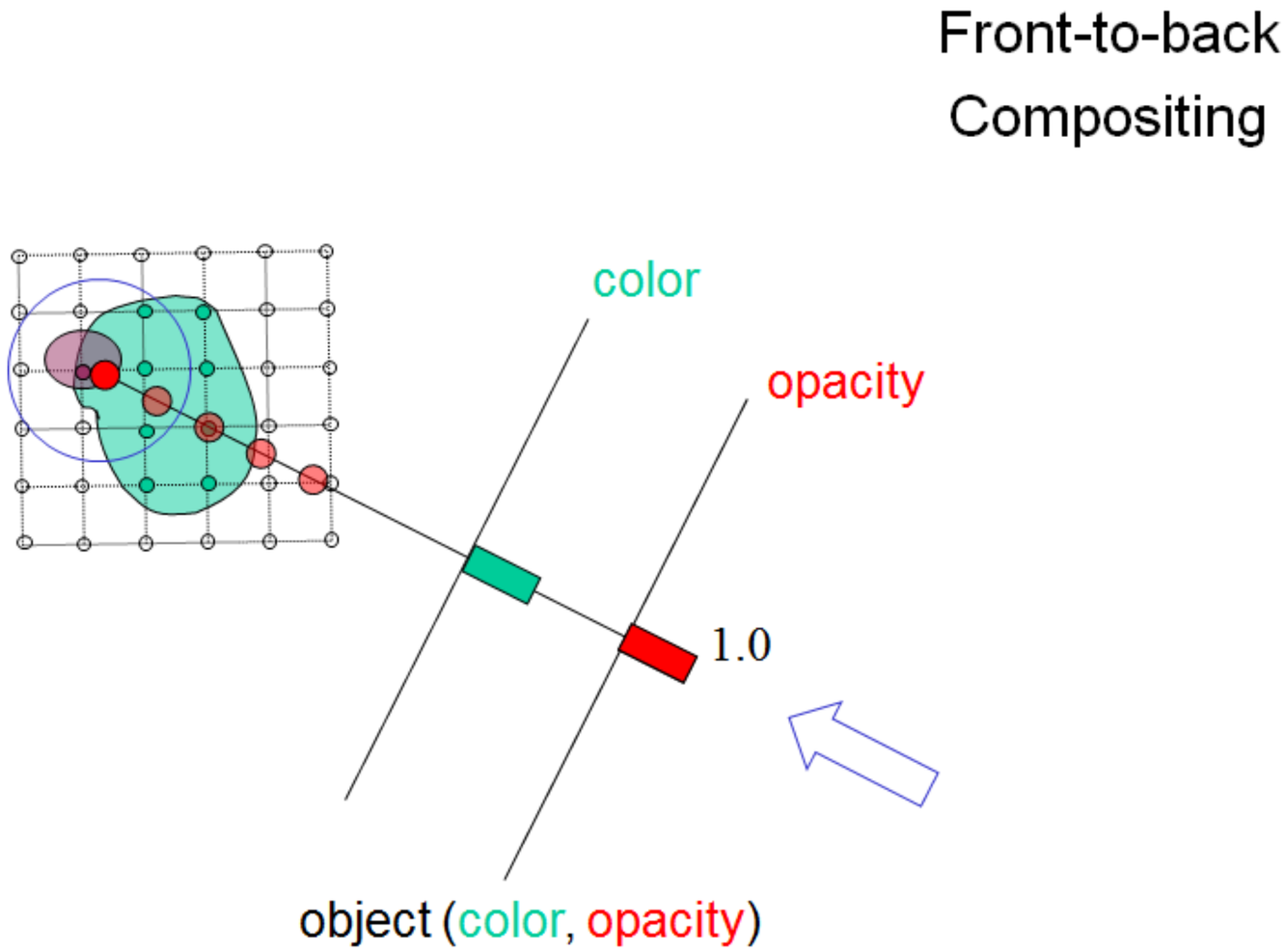


Raycasting

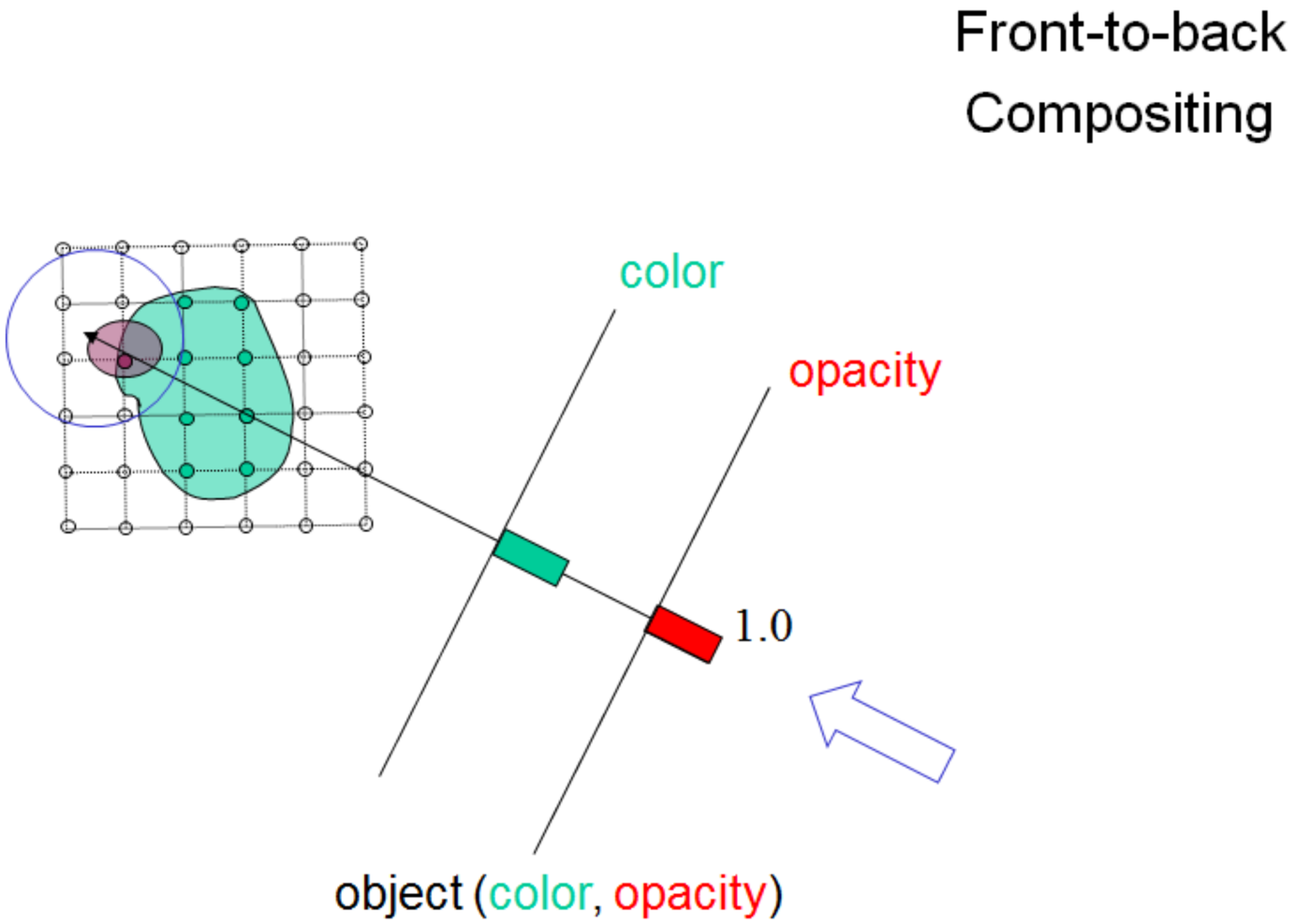
Front-to-back
Compositing



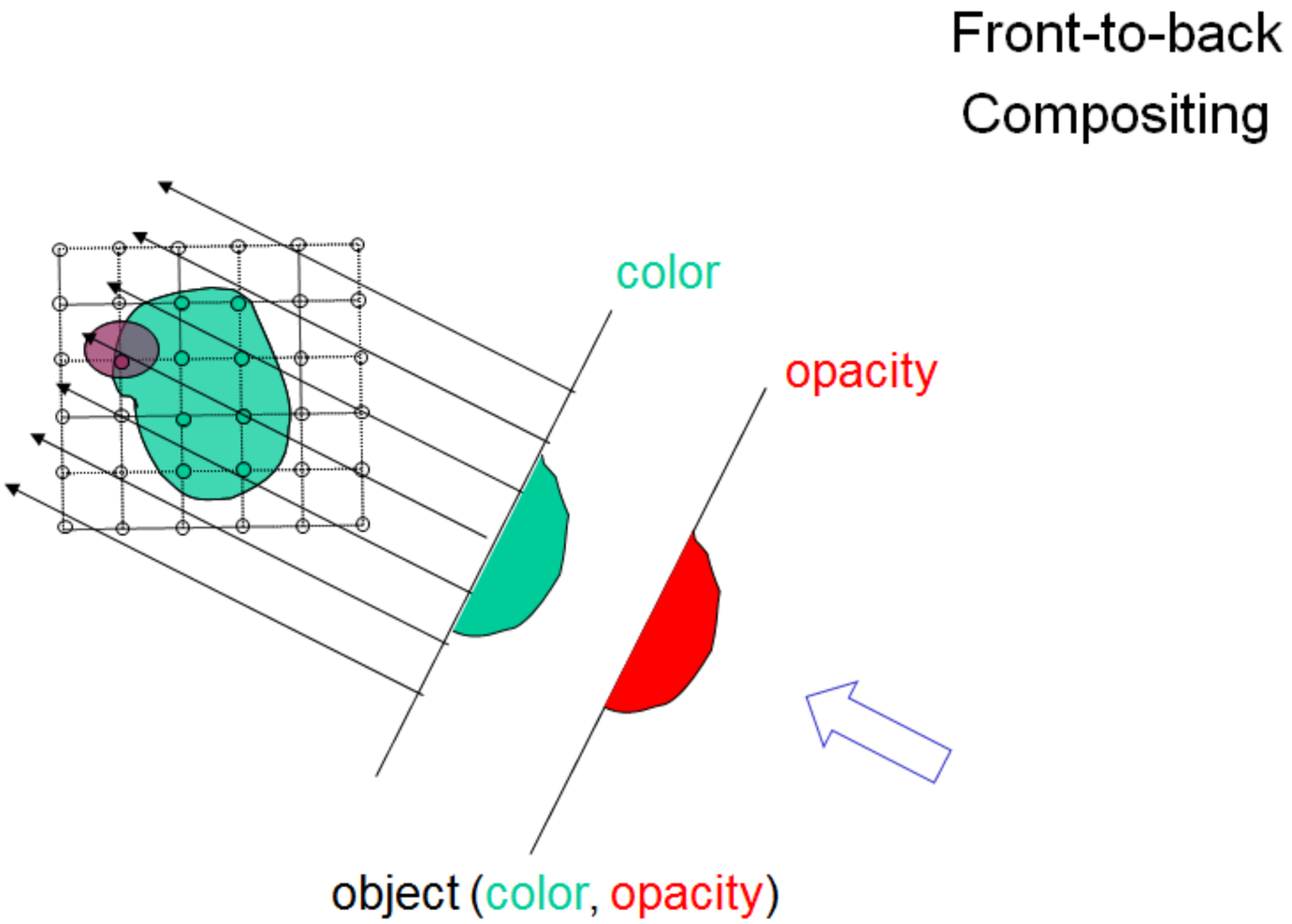
Raycasting


Front-to-back
Compositing
color
opacity
1.0
object (color, opacity)

Raycasting


Front-to-back
Compositing
color
opacity
1.0
object (color, opacity)

Raycasting


Front-to-back
Compositing
color
opacity
object (color, opacity)

Raycasting

- Processing steps
  - Ray discretization
    - Sampling of intensities
  - Assignment of color and opacity
    - Adjustment of transfer functions
  - Integration

Texture-based Volume Rendering

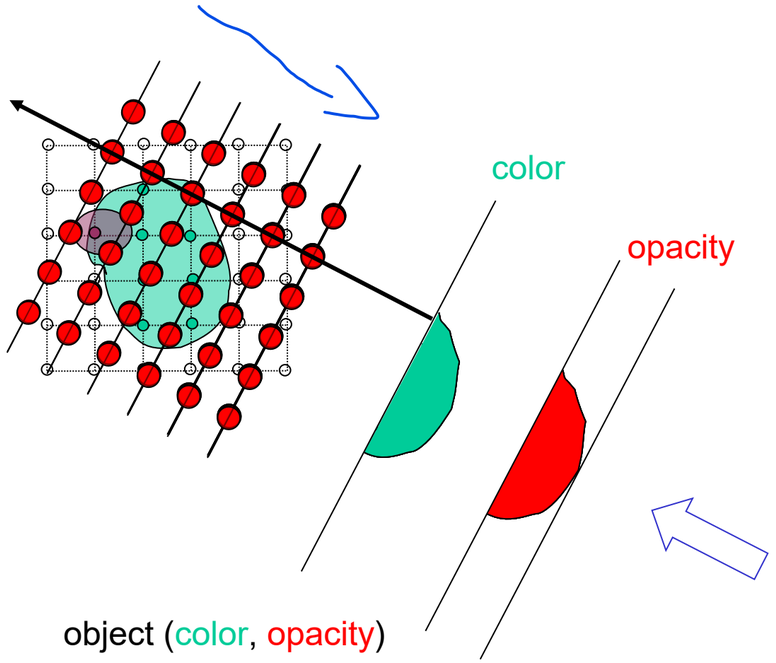
ohm

Texture-based DVR

- Characteristics
  - Object order approach
  - Based on graphics hardware
    - Rasterization
    - Texturing
    - Blending
  - Use proxy geometry
    - Due to missing volumetric primitives
    - Calculate slices trough volume

Texture-based DVR

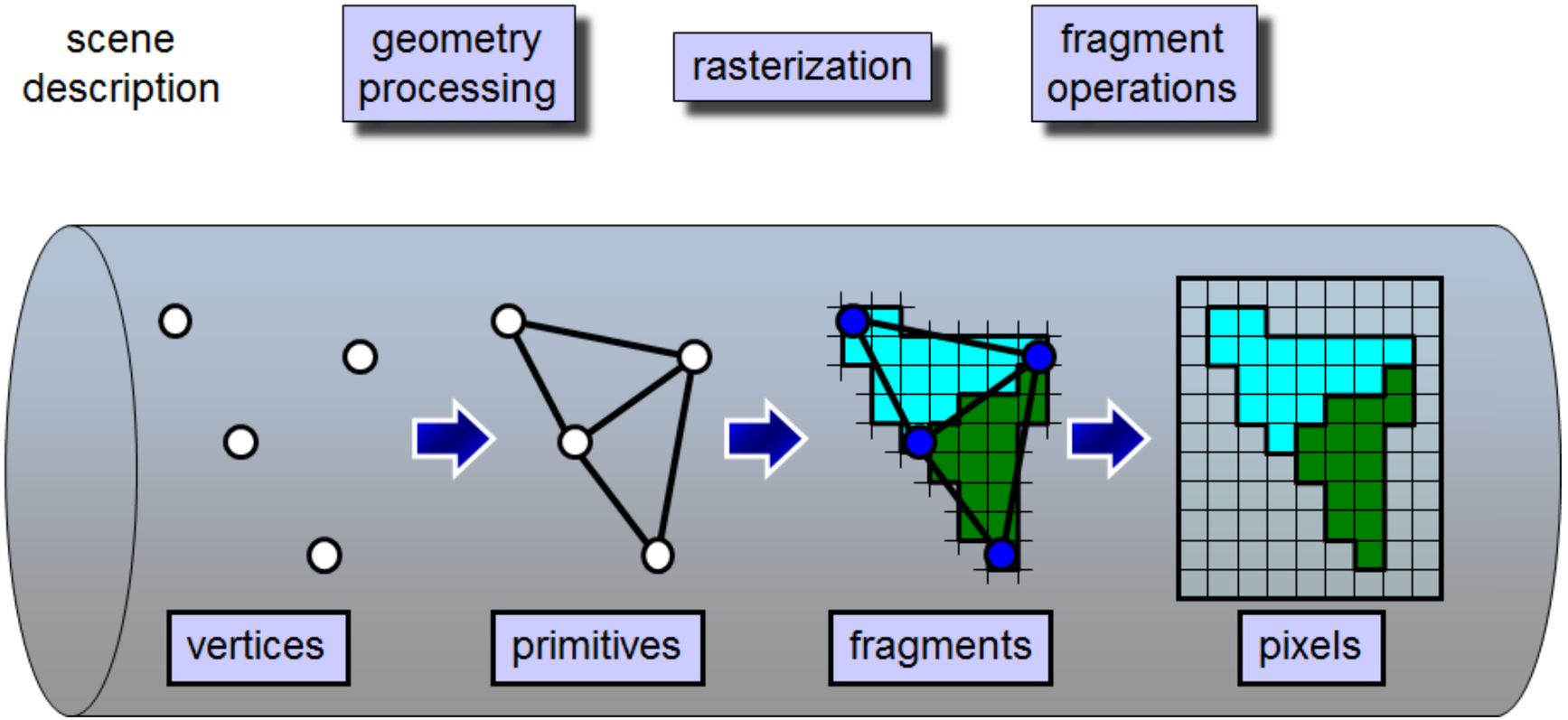
- Slice-based rendering



Ergebnis => Überlagerung der Textur durch die Schnitte im Volumen (Back to Front)

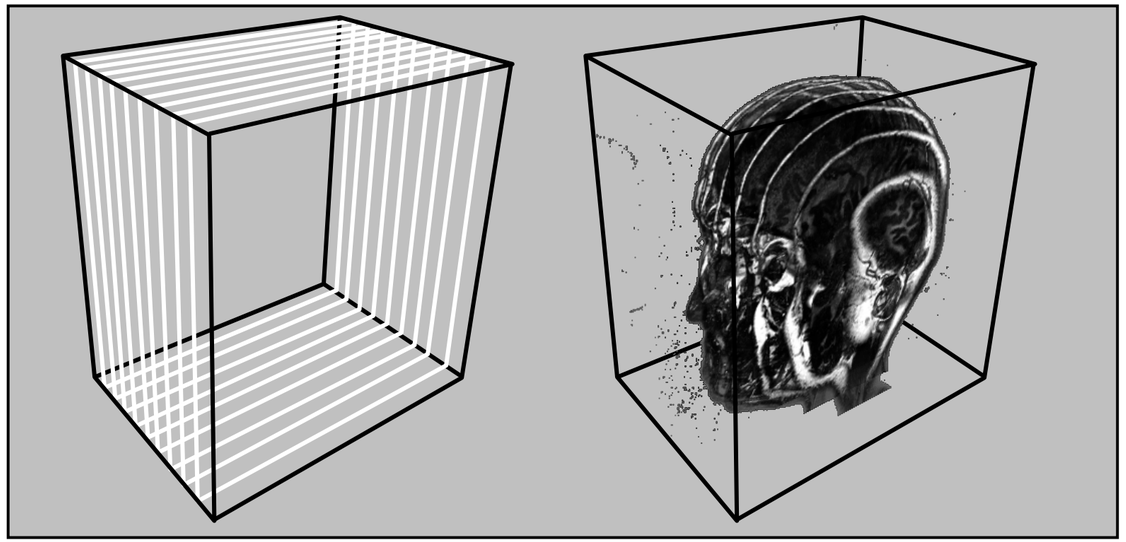
Texture-based DVR

- Rendering pipeline (recap)



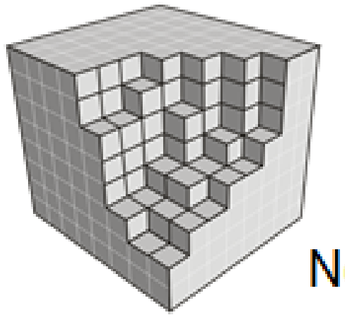
Texture-based DVR

- Proxy geometry
  - Stack of texture-mapped slices
  - Generate fragments
  - Most often back-to-front traversal

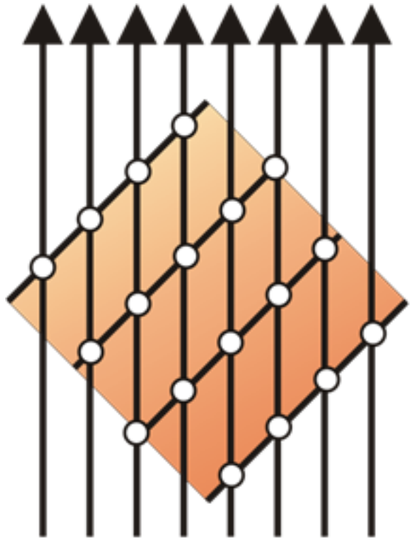


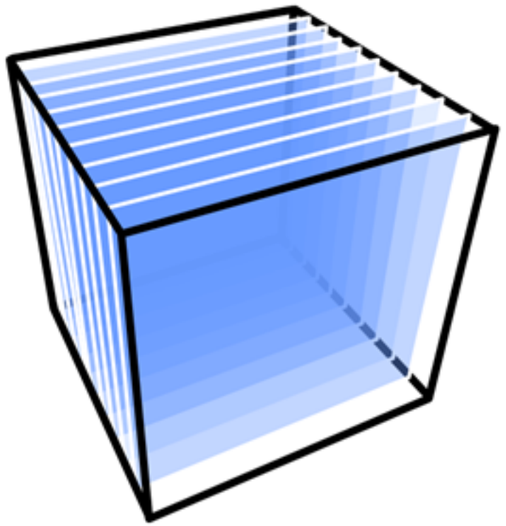
Texture-based DVR

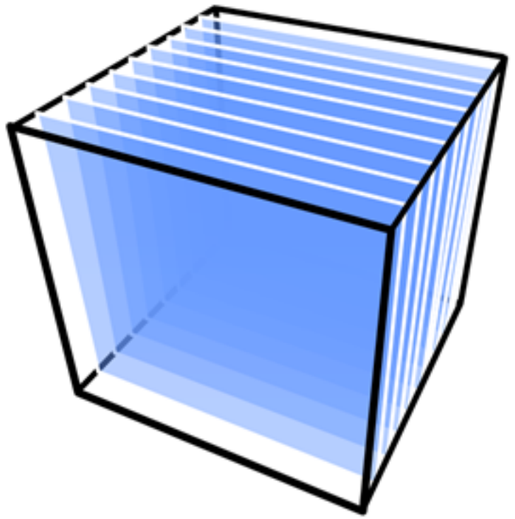
- 2D textured slices
  - Object-aligned slices
  - Three stacks of 2D textures
  - Bilinear interpolation



Note: Scalar data on uniform rectilinear grid



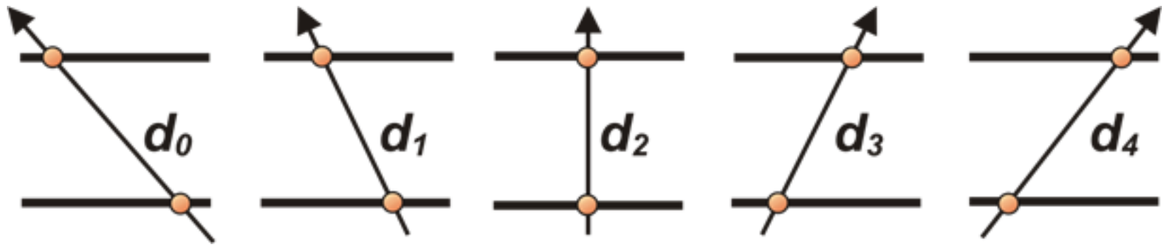




Texture-based DVR

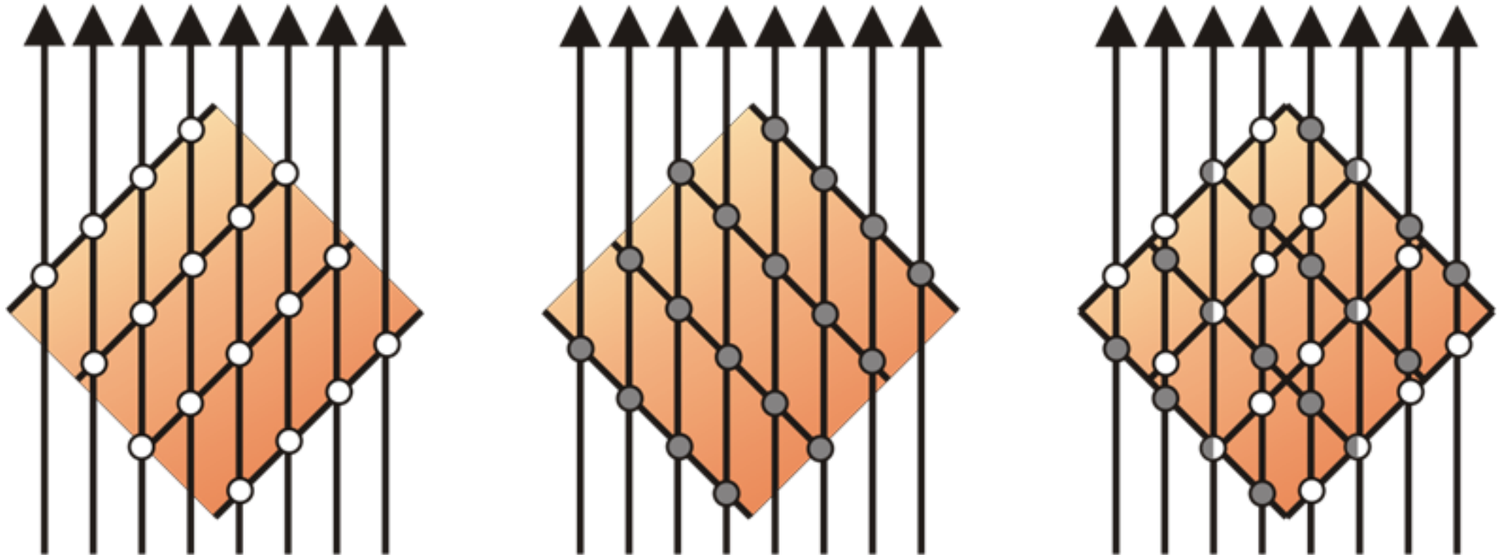
- Stack of 2D textures
  - Works on older graphics hardware without 3D textures
    - Needs 3 stacks of textures: 3 times texture memory!
  - Only bilinear interpolation within slices
    - Fast
    - Problems with image quality
  - Sampling distance between slices along a ray
    - Depends on viewing direction
    - Apparent brightness changes if opacity is not corrected

abhängig von Blickwinkel damit nicht zwischen die Schichten geschaut werden kann



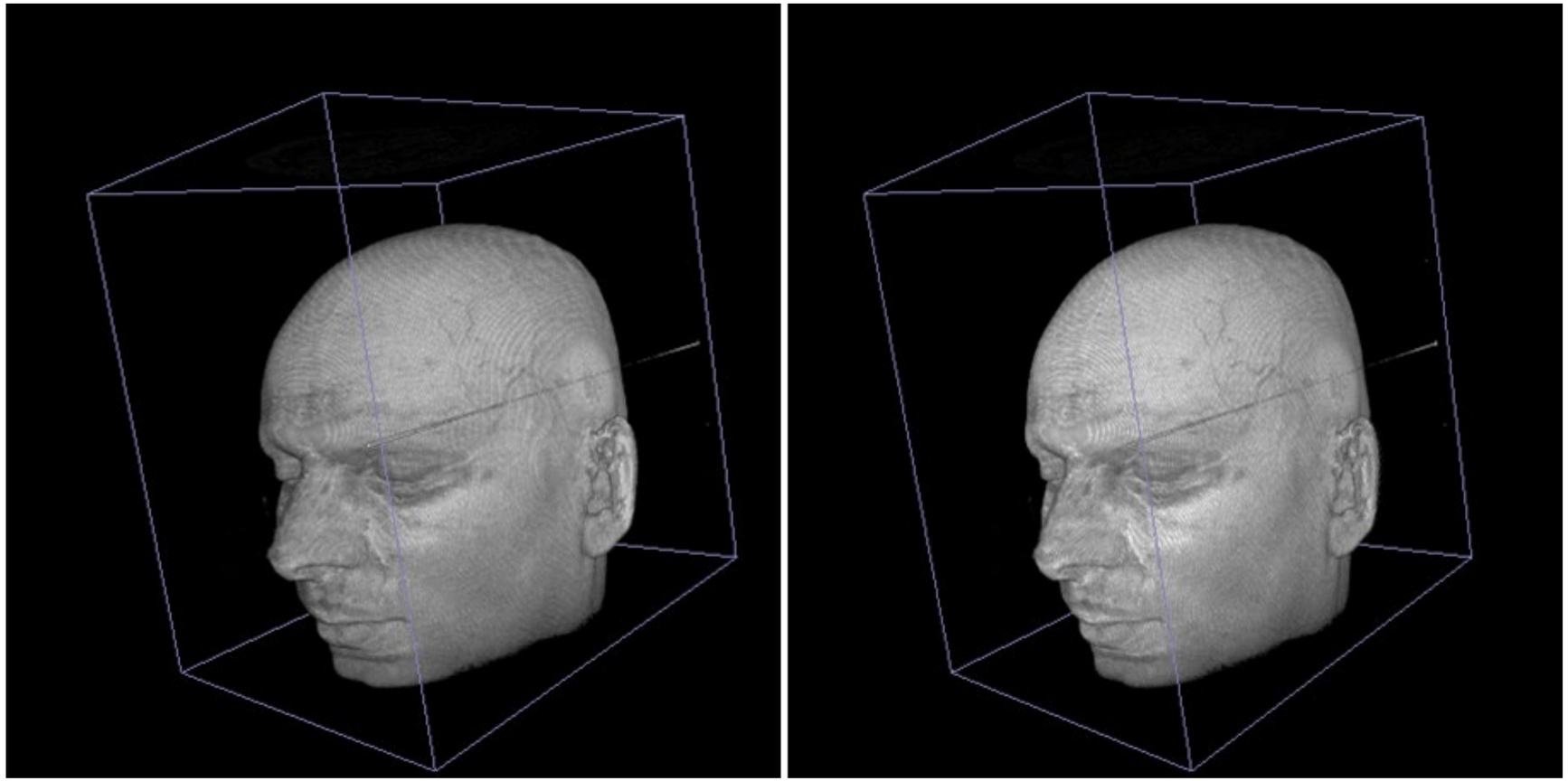
Texture-based DVR

- Stack of 2D textures
  - Artifacts when stack is viewed close to 45 degrees
    - Locations of sampling points may change abruptly



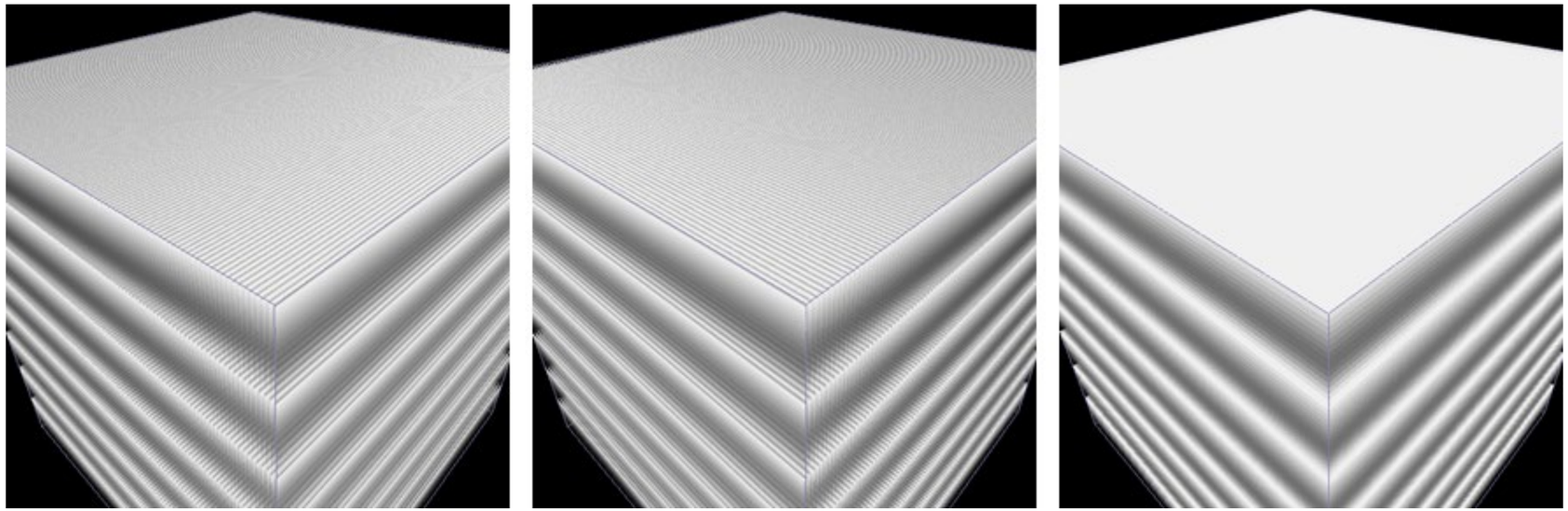
Texture-based DVR

- Brightness changes in 2D texture rendering



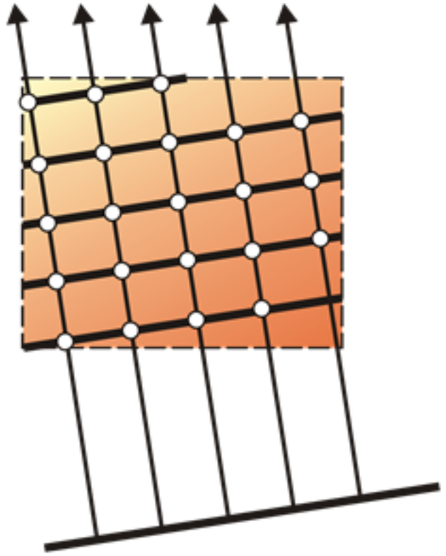
Texture-based DVR

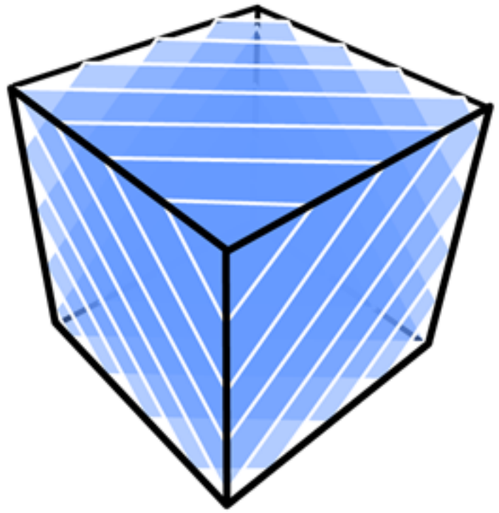
- Artifacts
  - When stack is viewed close to 45 degrees

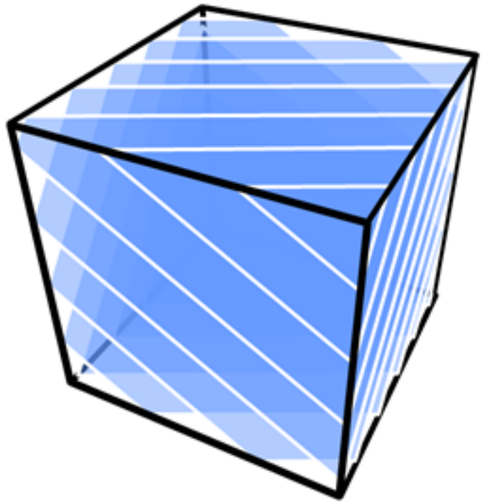


Texture-based DVR

- 3D textured slices
  - View-aligned slices
  - Single 3D texture
  - Trilinear interpolation

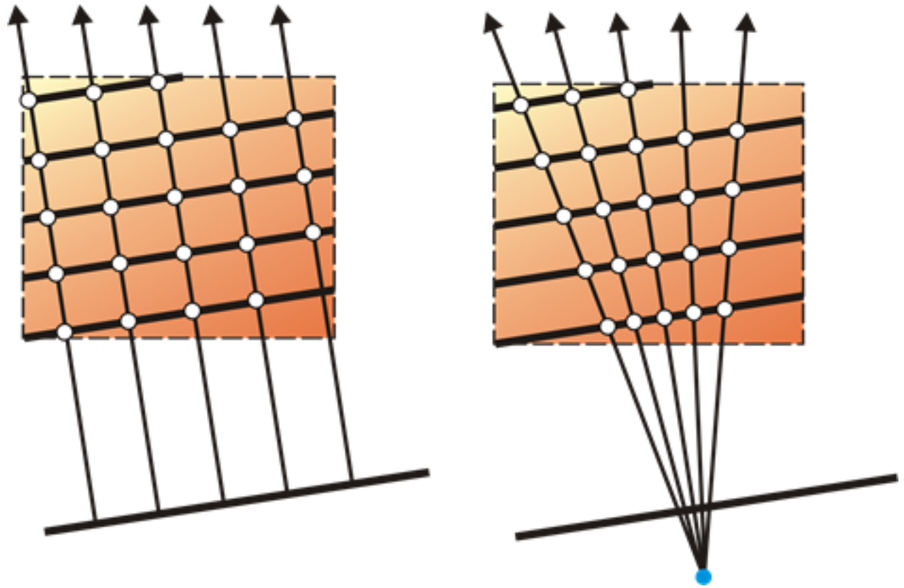






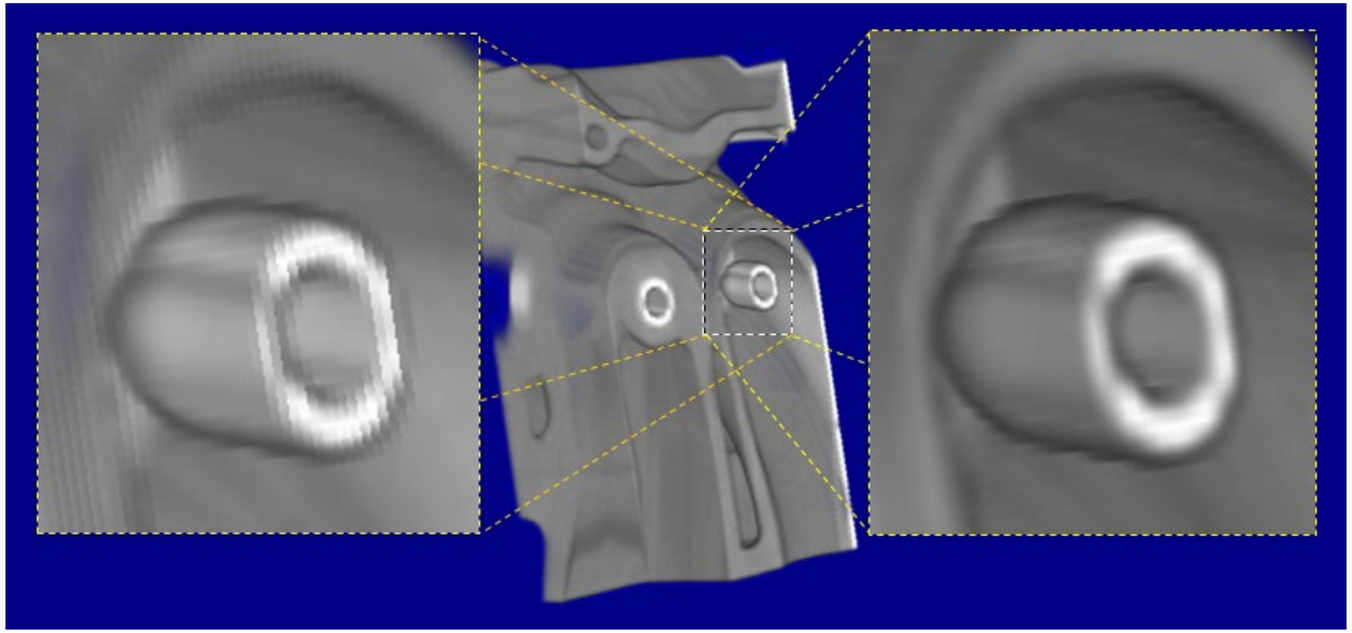
Texture-based DVR

- 3D textures
  - Needs support for 3D textures
  - Data set stored only once (not 3 stacks!)
  - Trilinear interpolation within volume
    - Slower
    - Good image quality
  - Constant Euclidian distance between slices along a ray
    - Constant sampling distance (except for perspective projection)

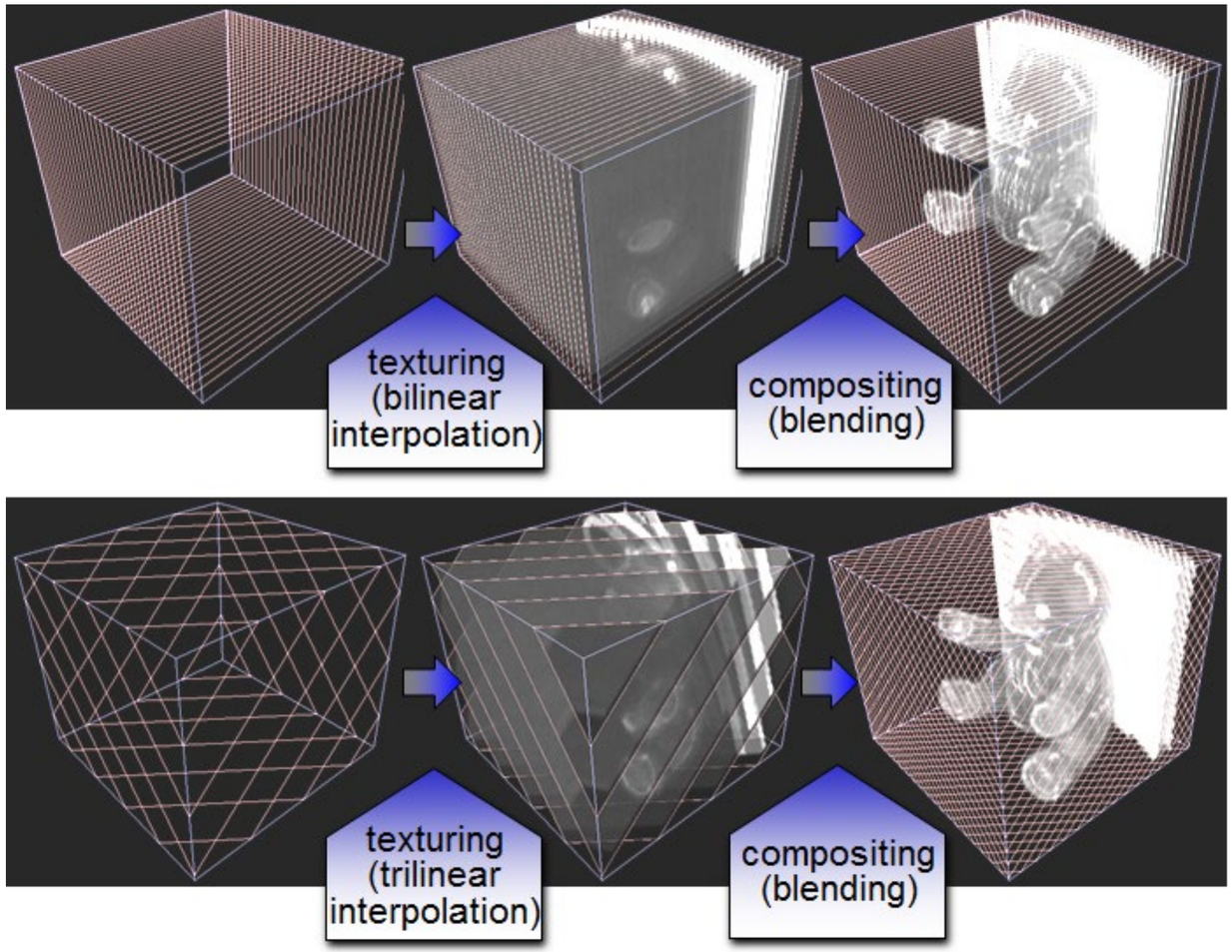


Texture-based DVR

- 3D textures
  - No artifacts due to inappropriate viewing angles
  - Increase sampling rate → more slices
    - Easy with 3D textures



Texture-based DVR



2D textures
axis-aligned

3D texture
view-aligned

ohm

Texture-based DVR

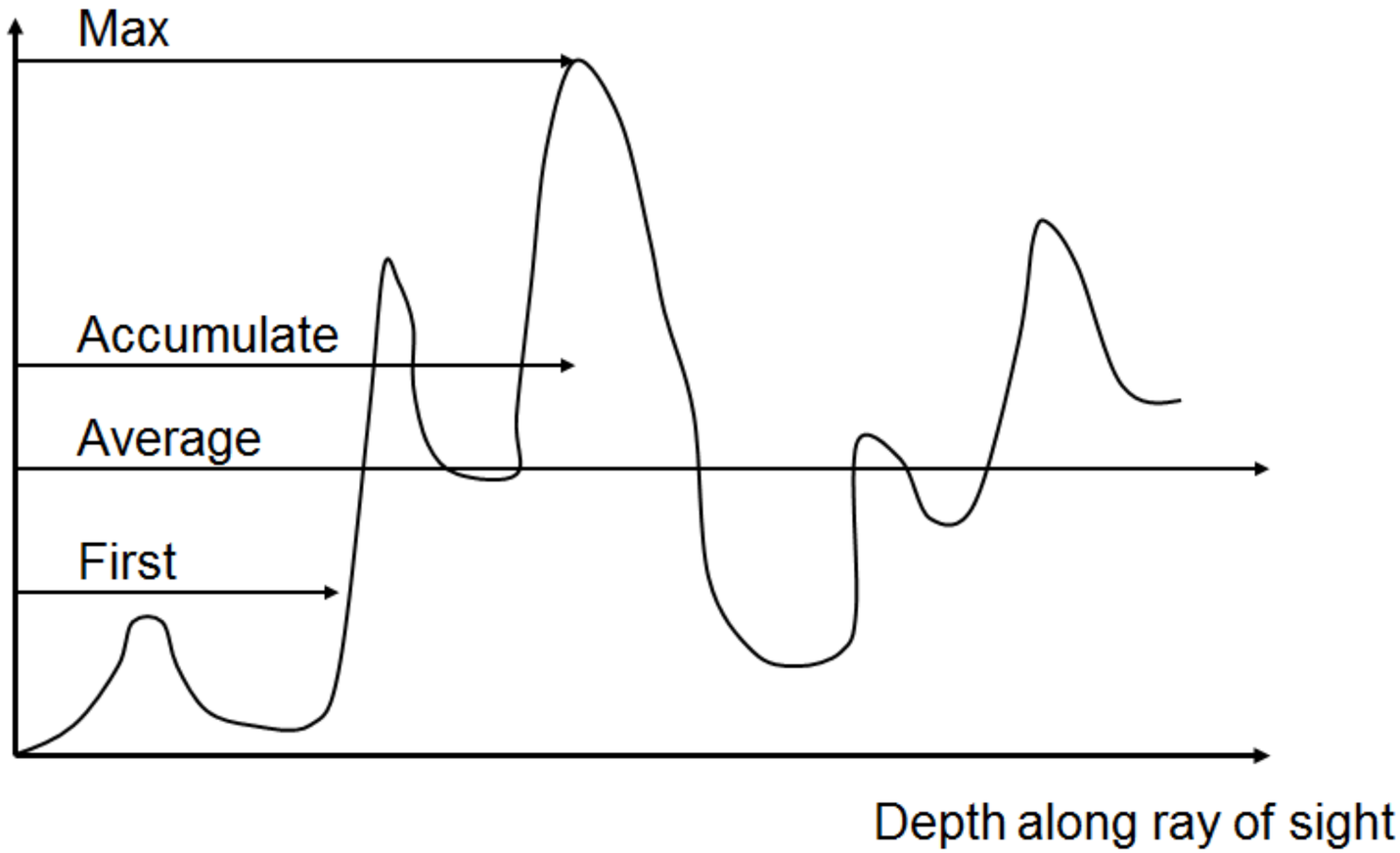
- Advantages
  - Exploits graphics hardware
  - Fast for moderately sized data sets
  - Combines surface and volumetric representations
    - Allows for mixture with opaque geometries
- Disadvantages
  - Limited by texture memory
    - Use bricking at the cost of additional texture downloads
  - Brute force representation by slices
  - No acceleration techniques as for ray casting
  - Rasterization speed and memory access can be problematic

6.5.3 (Other) Compositing Schemes

Compositing Schemes

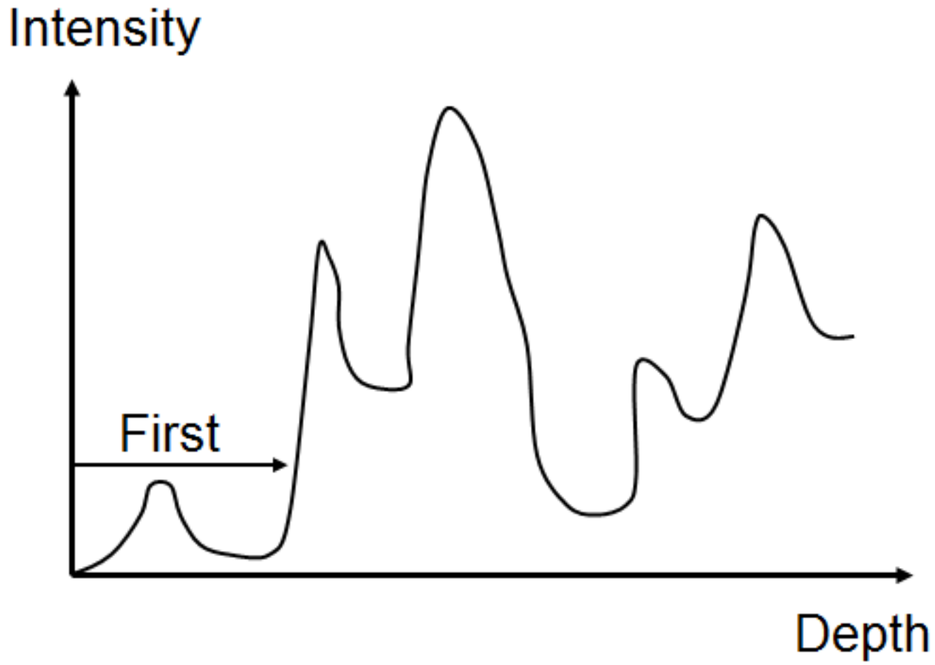
- Other approaches for determining the final color exist
  - Simpler calculations
  - Different visualization results
  - Application dependent
- Variations of composition schemes
  - First
  - Average
  - Maximum intensity projection
  - Accumulate (i.e. evaluation of the integral, chapter 6.5.2)

Compositing Schemes



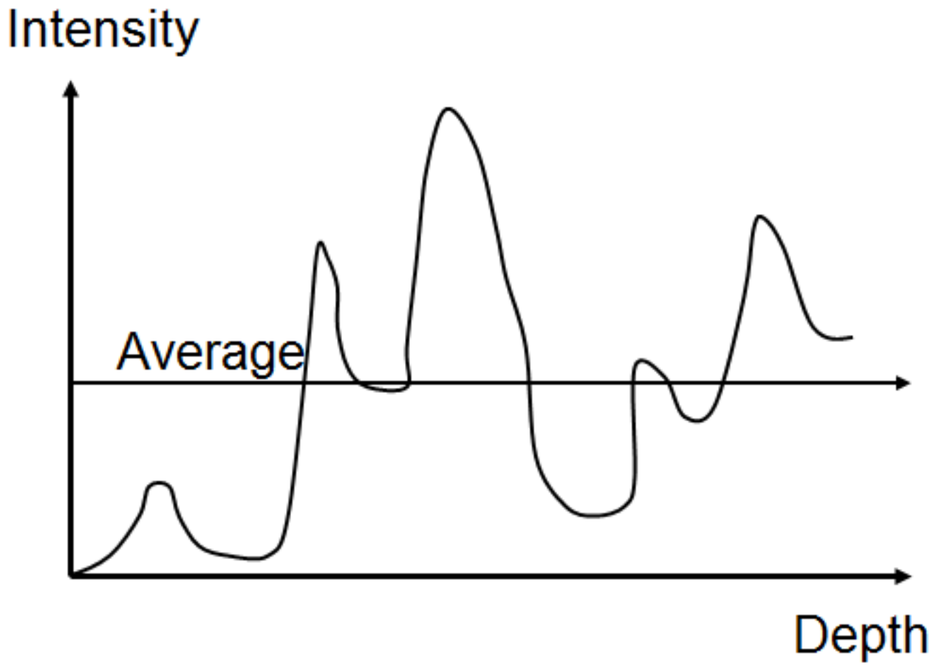
Compositing Schemes

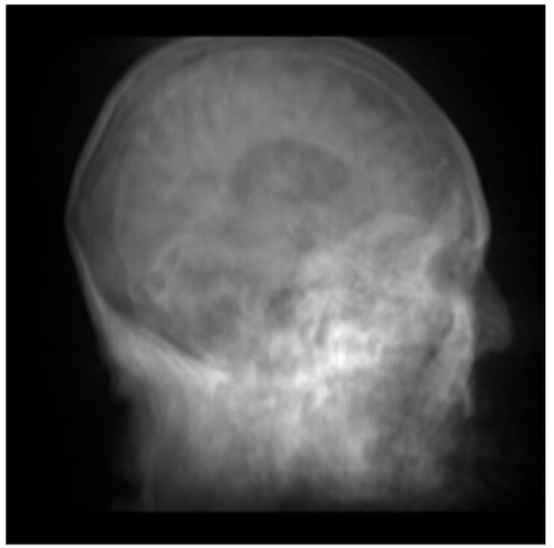
- First hit of iso-value along a ray
  - Extracts iso-surfaces



Compositing Schemes

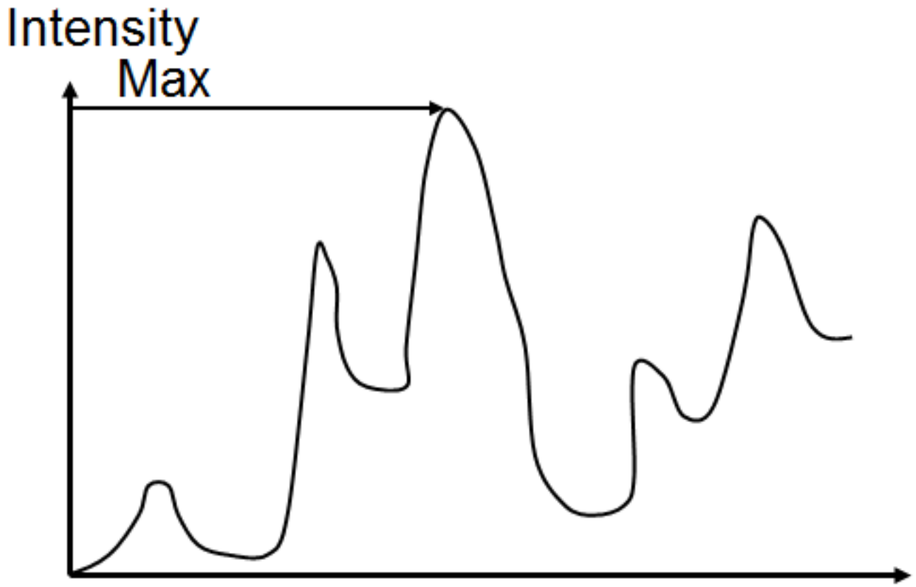
- Average of all values along a ray
  - Produces basically an X-ray picture

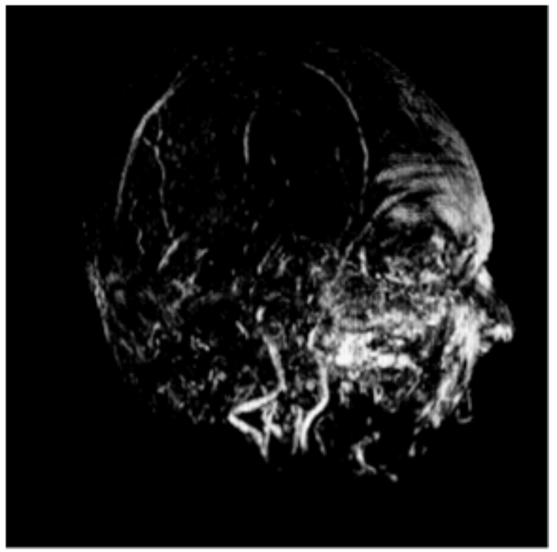




Compositing Schemes

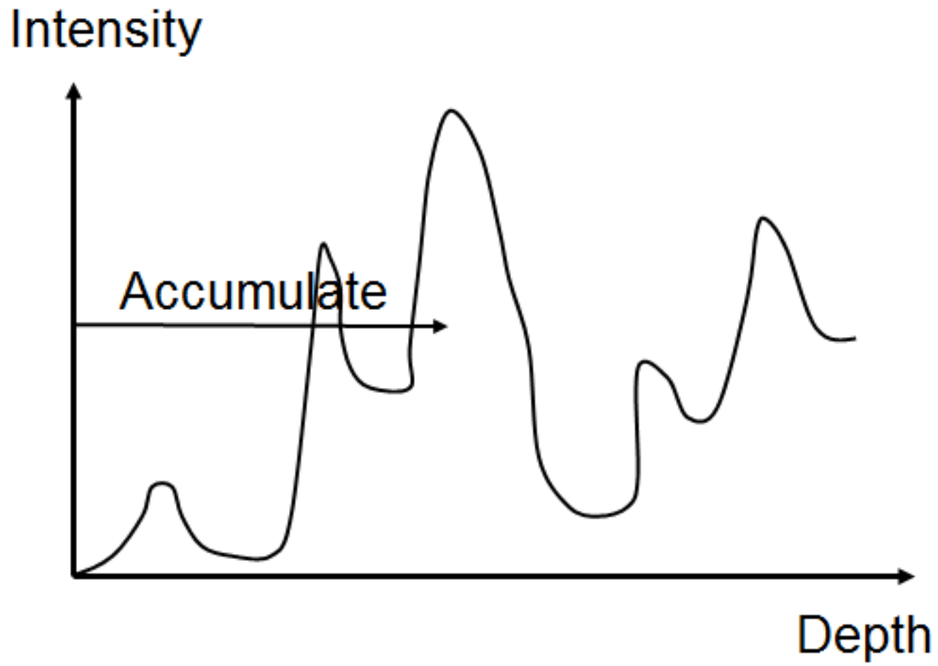
- Maximum value along a ray
  - Maximum intensity projection (MIP)
    - Good for structures with values higher than the rest

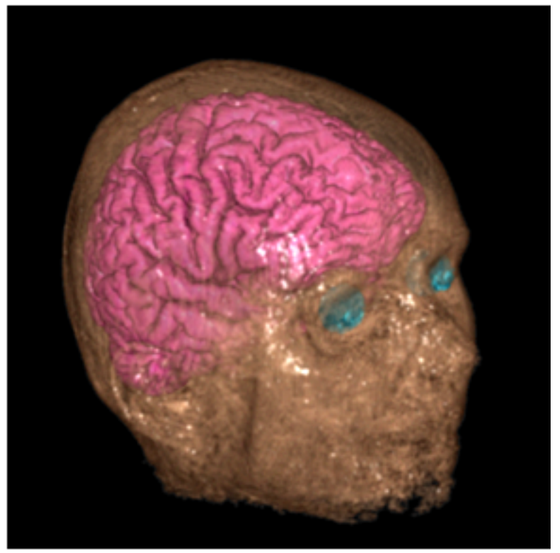




Compositing Schemes

- Accumulate all values along a ray
  - Emission-absorption model (integration of volume rendering equation)



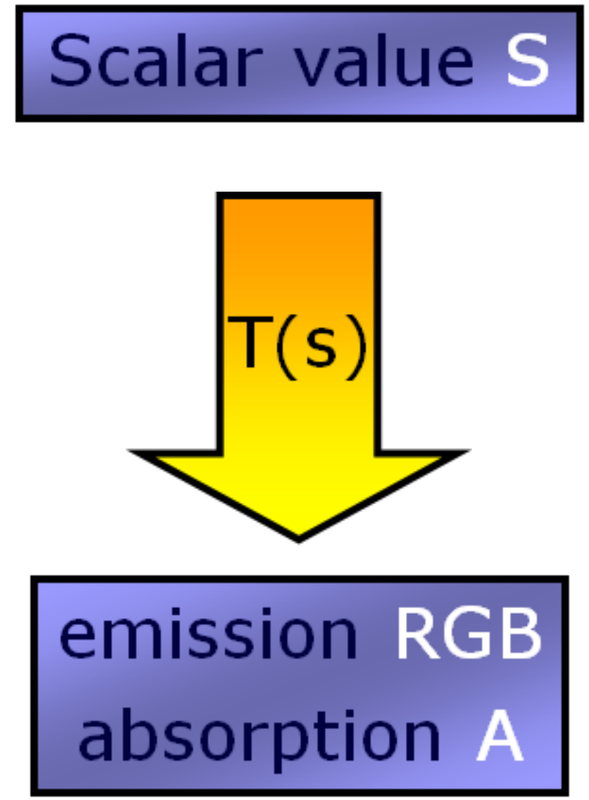


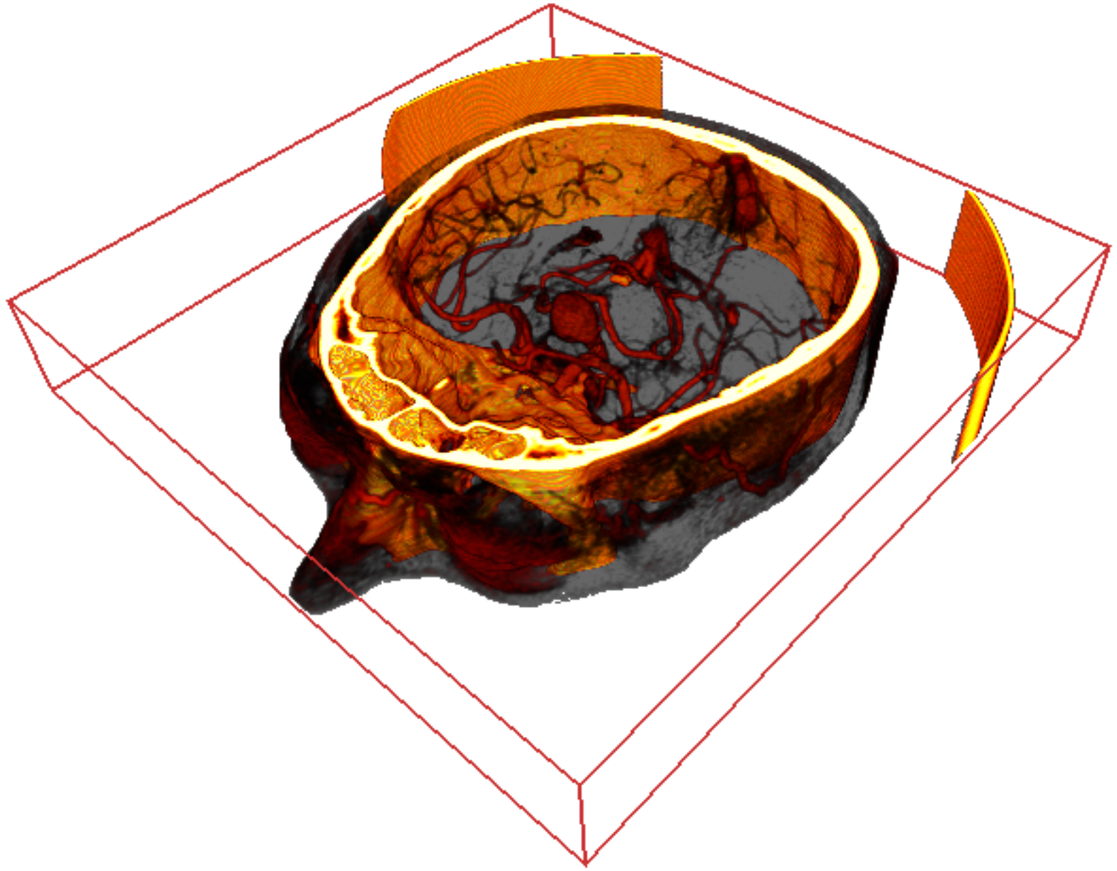
6.5.4 Classification

Classification

Wie werden Werte gesetzt dass es richtig Visualisiert wird

Transfer Funktion

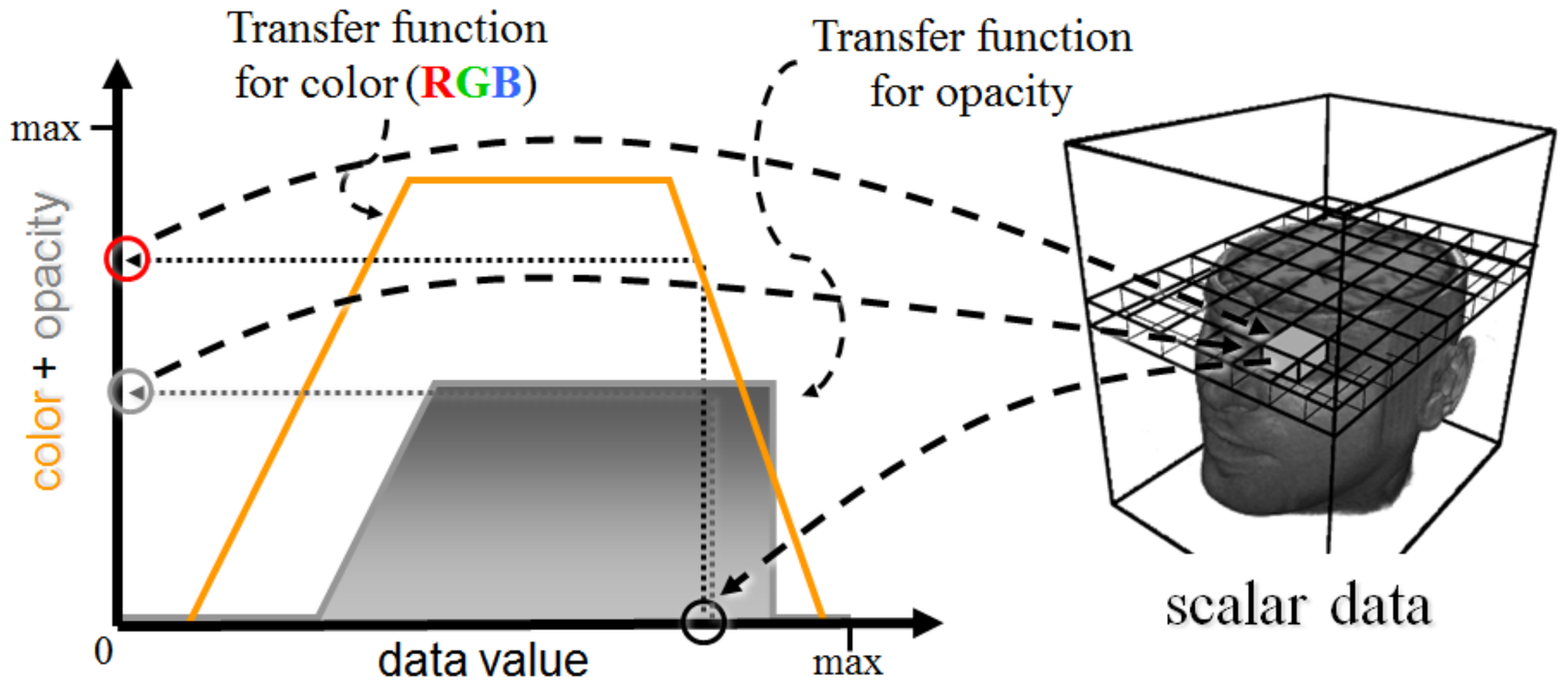




Classification

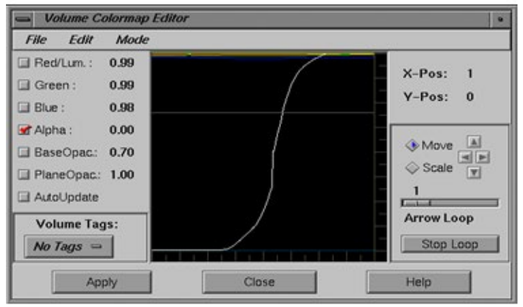
- Transfer functions
  - Map data value to color and opacity

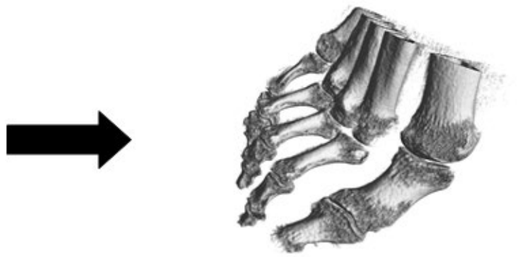
look up Wert von Farbe/Grauwert für Scalar data

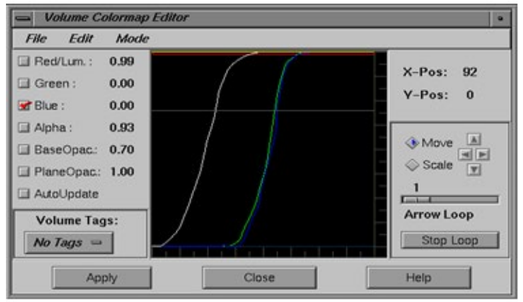


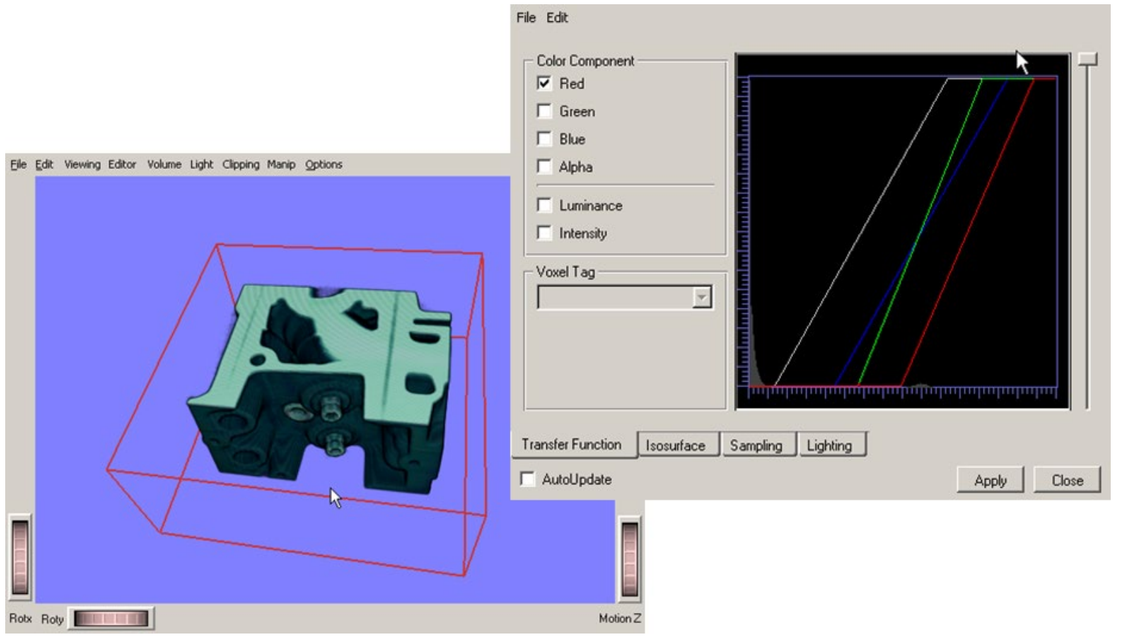
Classification

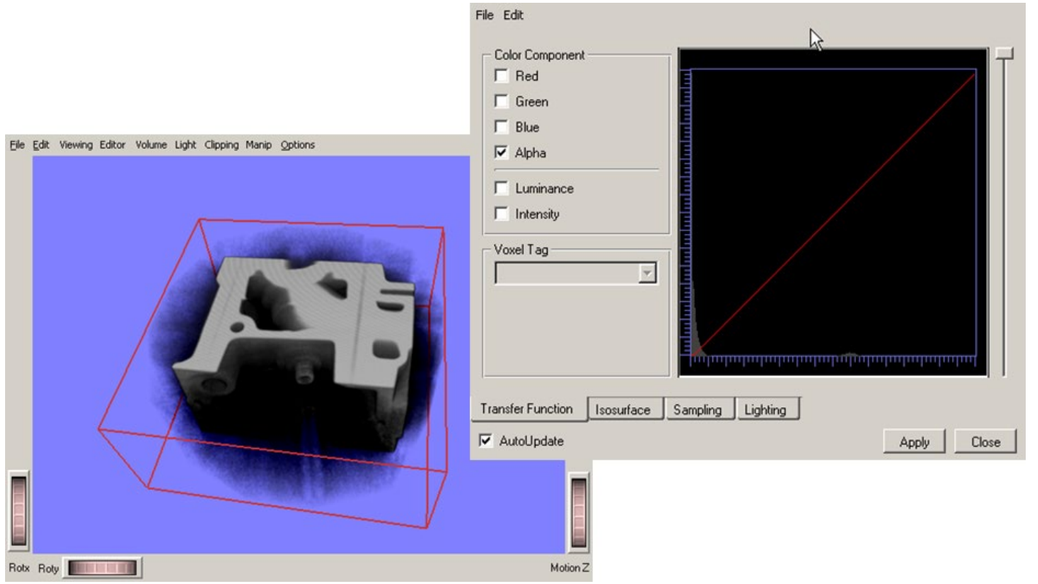
Examples





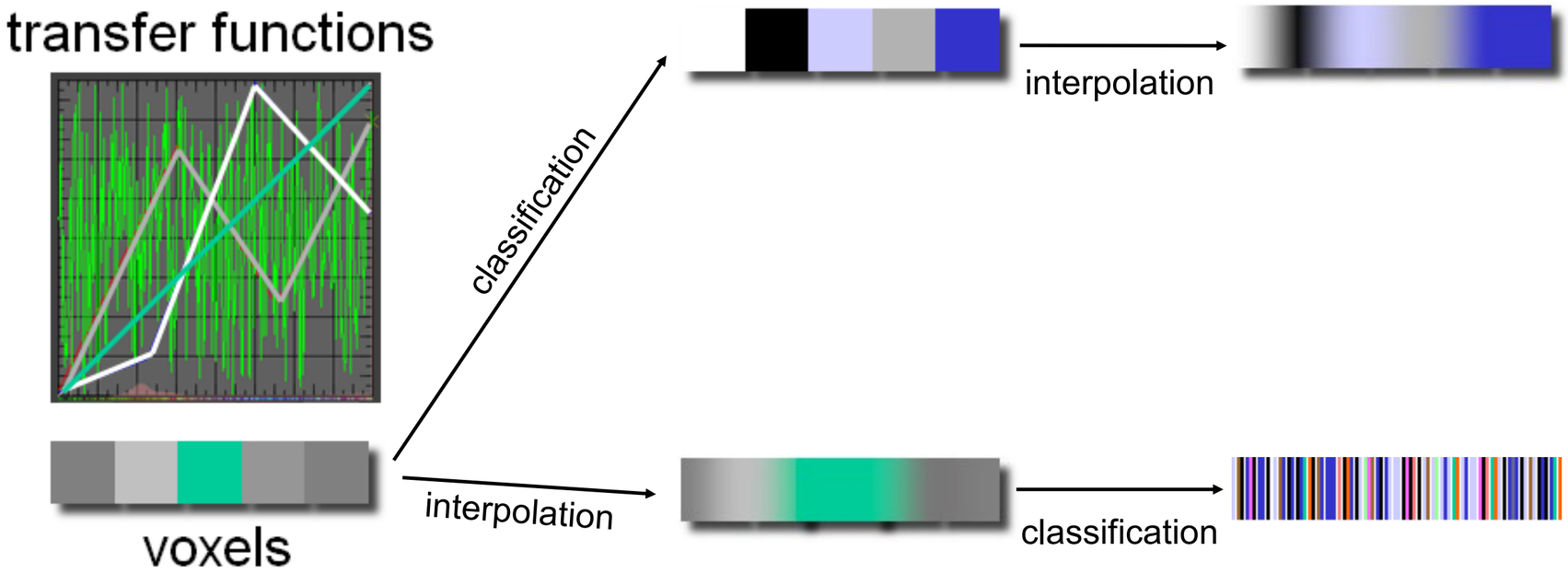


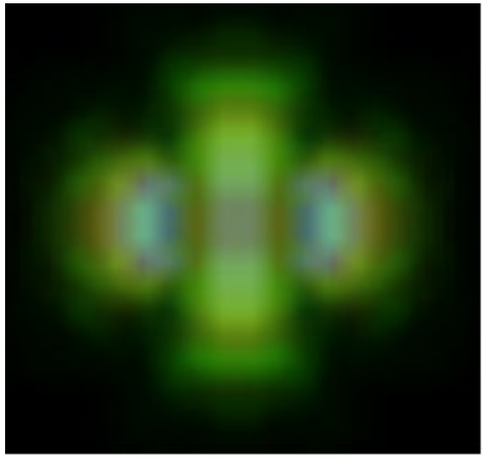


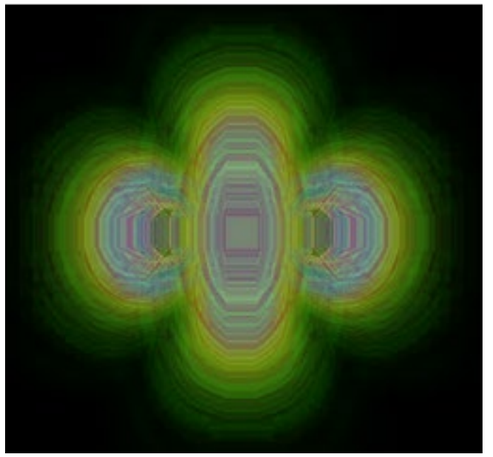


Classification

Pre- vs. post-classification



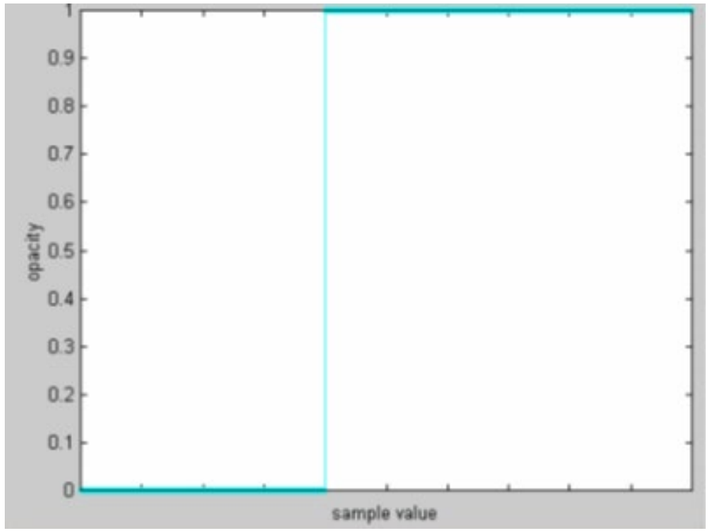


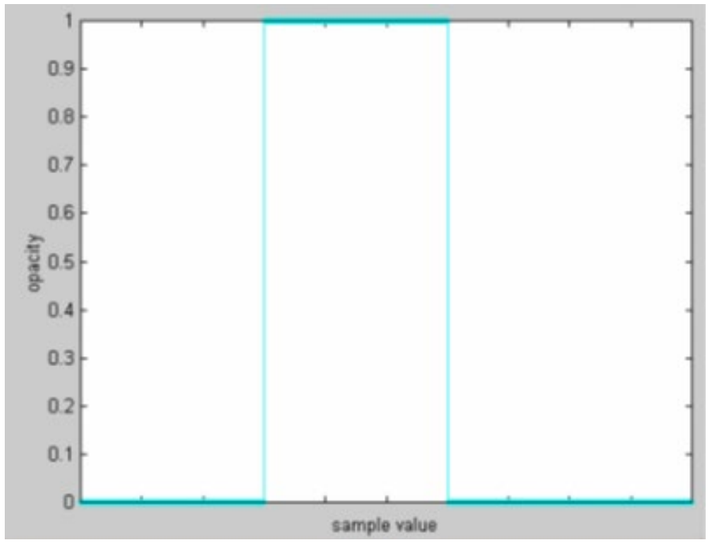


Classification

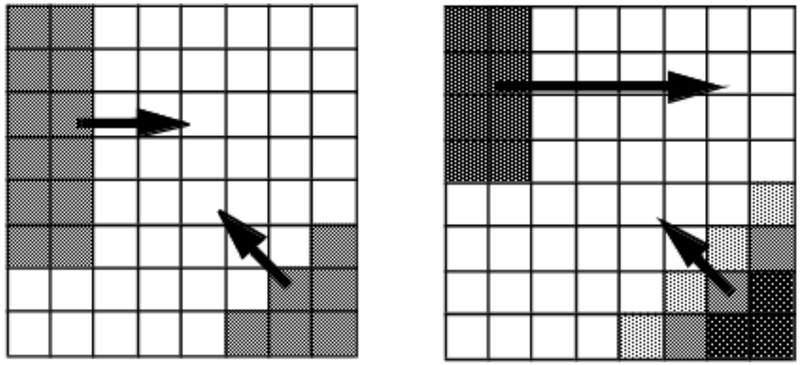
Isovalue contour surfaces

- Render opaquely all voxels with values > threshold
  - Unable to display multiple concentric surfaces
- Use a window instead of a threshold
  - To narrow → holes
  - Too wide → restricted display of multiple surfaces
  - Generates artifacts that are not part of the original data





Classification



- Considerations
  - Usually not only interested in a particular iso-surface but in regions of "change"
  - Feature extraction
    - High value of opacity in regions of change
    - Homogenous regions less interesting - transparent
  - Surface "strength" depends on gradient
  - Gradient of the scalar field is taken into account

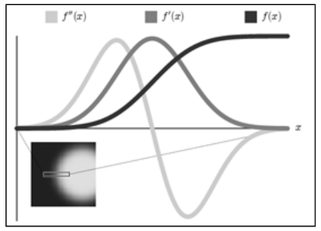
Classification

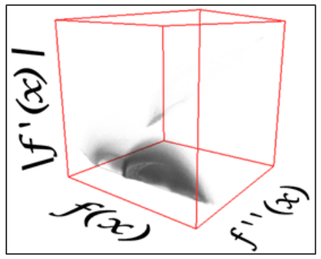
- In order to emphasize iso-surfaces
  - Opacity of a_v = max to voxel with isovalue f_v
  - Opacity to zero to remaining voxels
  - To avoid aliasing artifacts
    - Assign opacities close to av to voxels with values close to f_v
- Most pleasing if transition region stays constant
  - Opacity fall off at a rate inversely proportional to magnitude of local gradient
    - e.g.

$$\alpha(x_i) = \alpha_v \begin{cases} 1 & \text{if } f(x_i) = f_v \text{ and } |\nabla f(x_i)| = 0 \\ 1 - \frac{1}{r}\left|\frac{f_v - f(x_i)}{|\nabla f(x_i)|}\right| & \text{if } f_v - f(x_i) \le r|\nabla f(x_i)| \\ 0 & \text{otherwise} \end{cases}$$

Classification

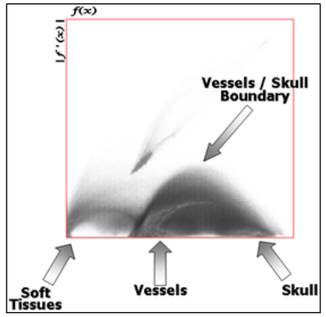
Multidimensional transfer functions



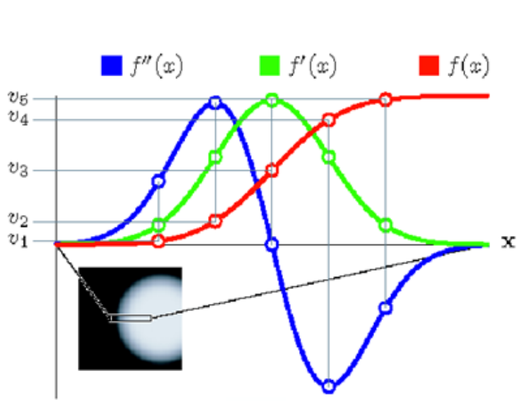


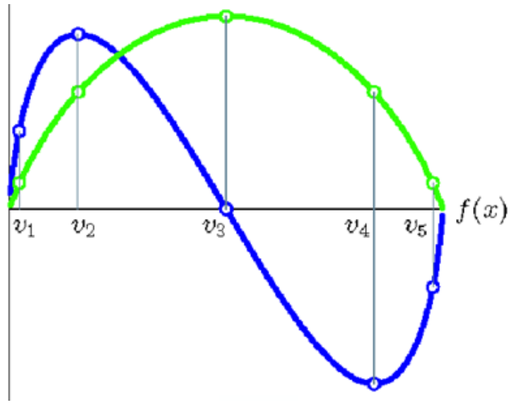
Representation in a 3D histogram

2D representation by integration in direction of $f''(x)$



- Problem: How to identify boundary regions/surfaces
- Approach: 2D/3D transfer functions, depending on
  - Scalar value, gradient magnitude
  - Second derivative along the gradient direction



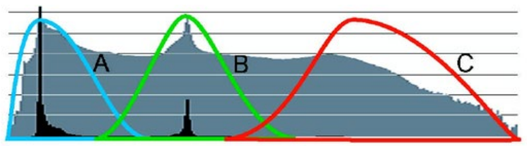


Classification

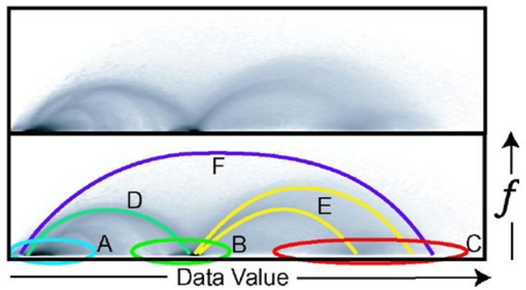
Multidimensional transfer functions

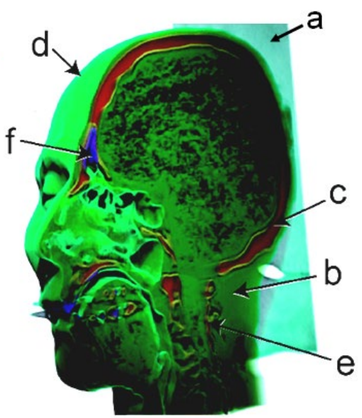
- Yields high quality transfer functions at the expense of restricted usability

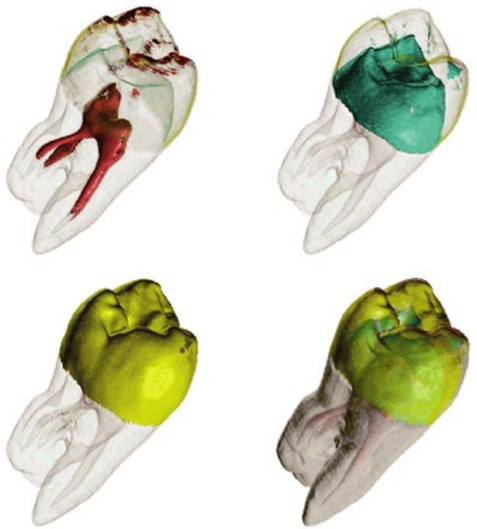
1 transfer function with 1D histogram: basic materials in colored regions (A,B,C)

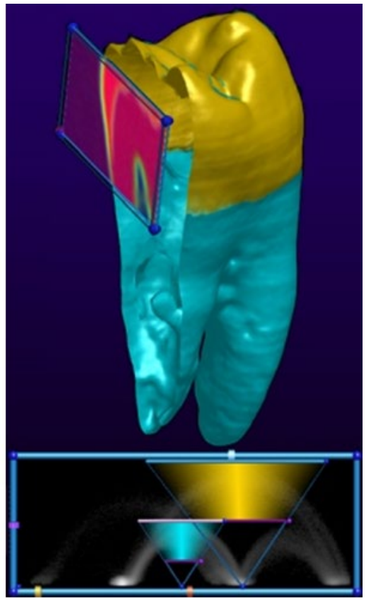


2D transfer function with 2D joint histogram:
materials (A,B,C), material boundaries (D,E,F)









Segmentation

Segmentation

zu meinem Datensatz noch Datensätze die Angeben wo andere Informationen sind (Color, Structure, ...)

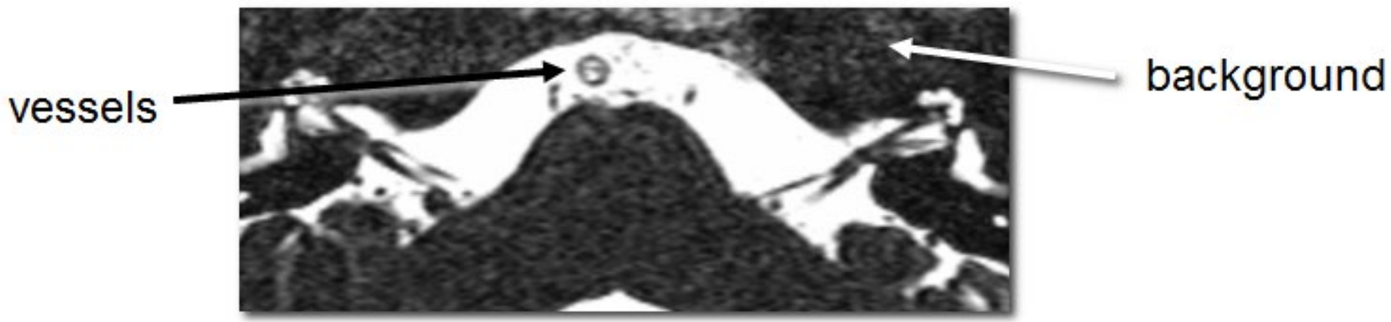
Explicit separation of structures

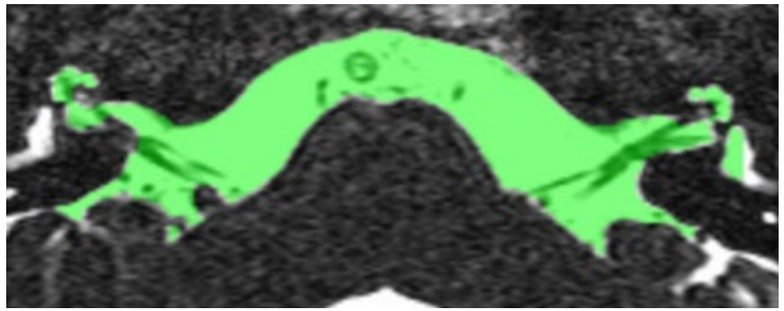
- Application areas
  - If classification with a transfer function fails
    - E.g. different structures, identical data values
  - In case of indirect volume rendering
    - E.g. if polygonal models are reconstructed out of volume data
- Label voxels indicating a type
  - Preprocessing
  - Semi-automatic process
- Note the difference
  - Implicit segmentation ⟷ classification with color and opacity
  - Explicit segmentation ⟷ labeling of structures

Segmentation

Example

- Implicit separation of background ⟷ vessels impossible
  - Both structures have identical data values





Assignment of individual T to each area

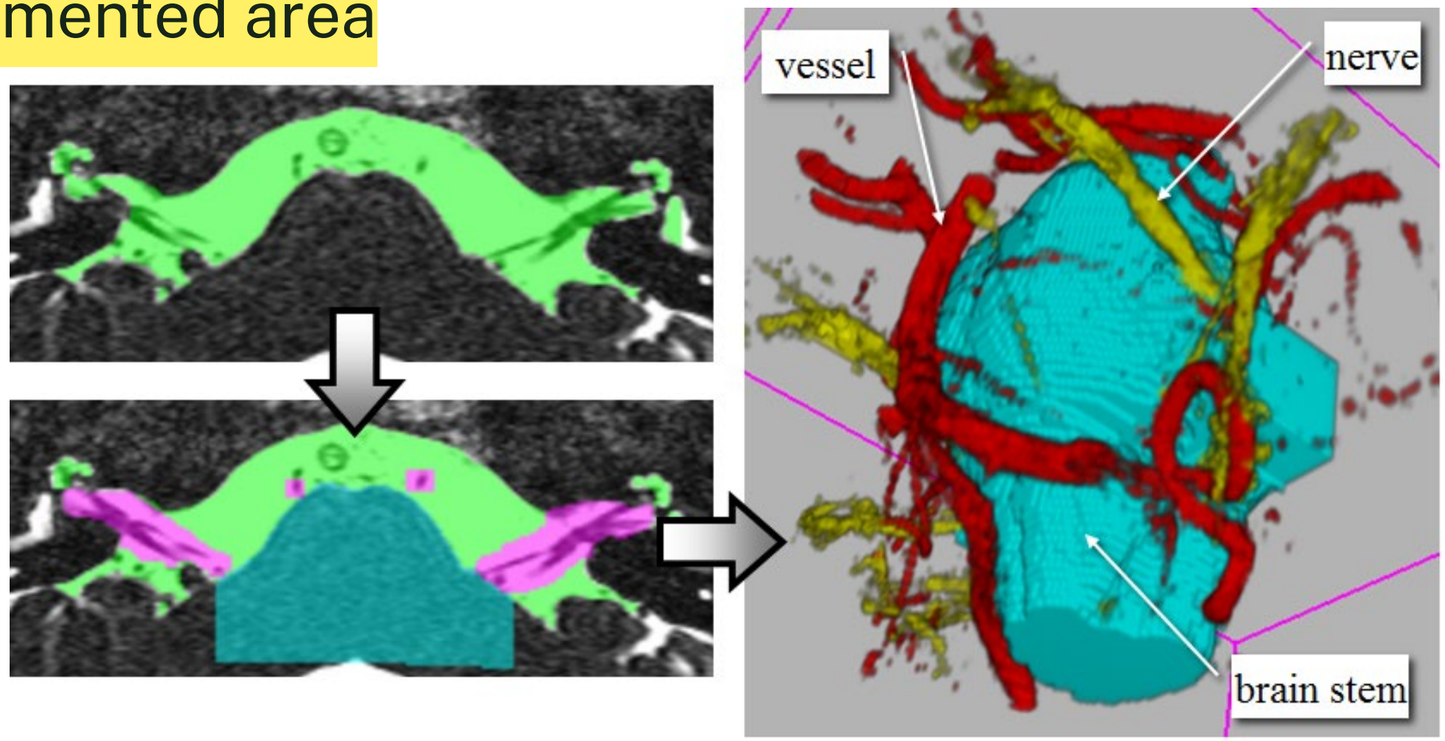
- Explicit segmentation of white area including vessels

Segmentation

Example (cont.)

jedem der Label kann eine eigene Transferfunktion zuweisen

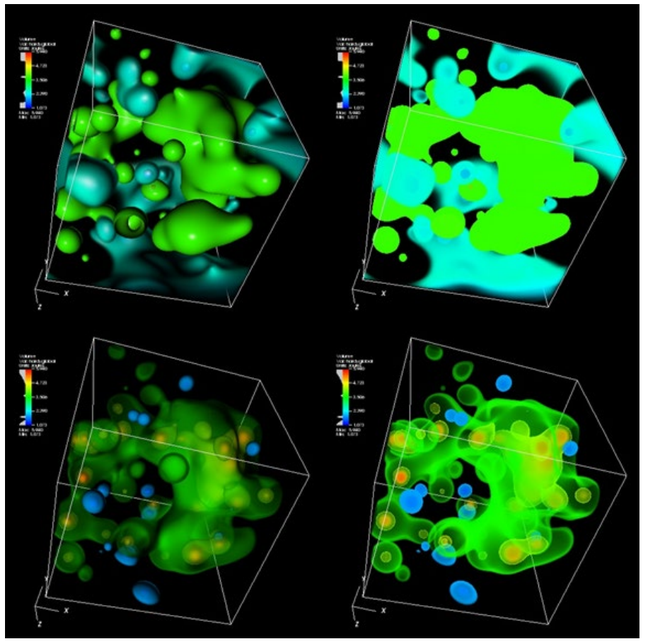
- TF 1: background to transparent
- TF 2: explicitly segmented area



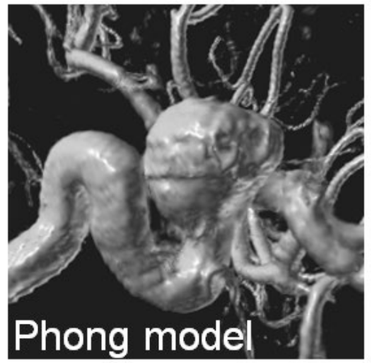
Volumetric Shading

Volumetric Shading

- Shading
  - Simulate reflection of light, simulate effect of color
  - Make use of human visual system's ability to efficiently deal with shaded objects
- What is the normal vector in a scalar field?
  - Use the gradient (perpendicular to iso-surface)
  - Numerical computation of the gradient
    - Central difference
      - Not isotropic - length is 1 to sqrt(3)
      - Needs normalization
    - Intermediate difference (forward/backward difference)
      - Very cheap
      - Noisy data means less good gradients
      - Also not isotropic



$K_a = 0.1$
$K_d = 0.5$
$K_s = 0.4$



6.5.5 Outlook

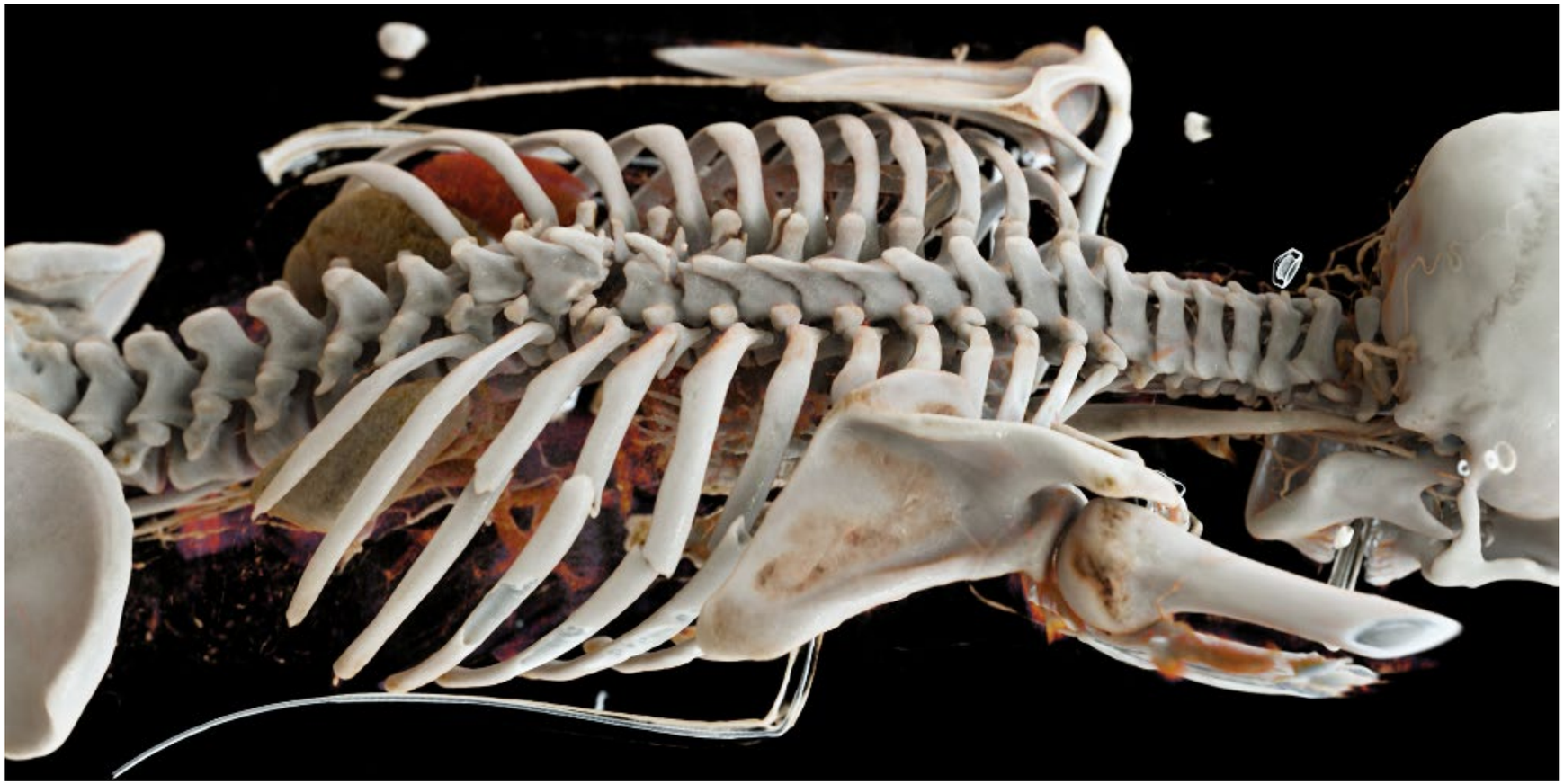
ohm

Cinematic Rendering

Physically based rendering

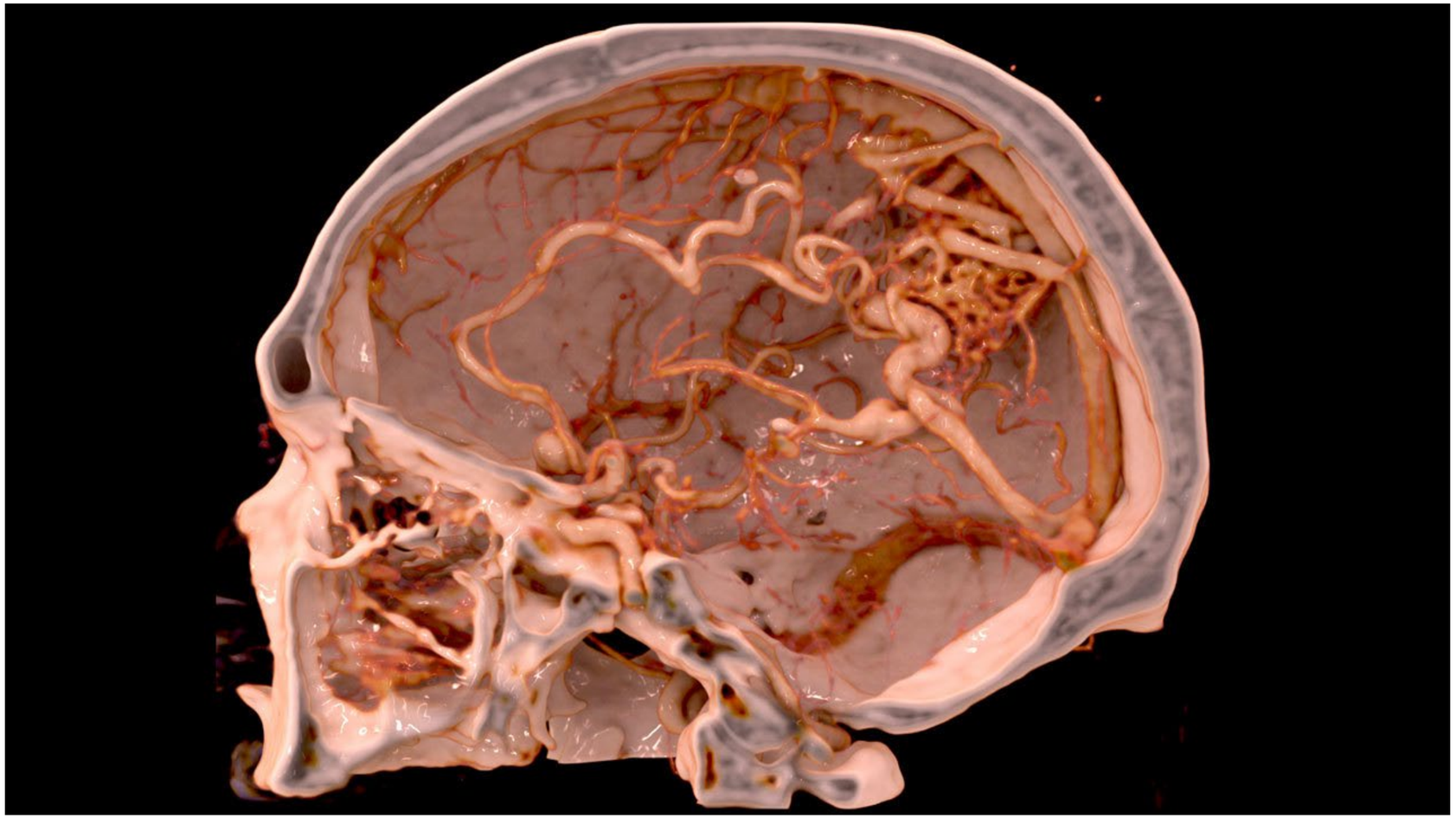
- Photorealistic rendering based on physically correct lighting calculations
- Light transport equation
  - $L_o(p, \omega_o) = L_o(p, \omega_o) + \int_{S^2} f(p, \omega_o, \omega_i) L_i(p, \omega_i) |\cos \Theta_i| d\omega_i$
- Can be extended to include volumetric scattering
- Cannot be evaluated analytically (except for the most simplest scenes)
- Monte-Carlo Ray-Tracing
  - $E[F_N] = E\left[\frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)}\right] = \int_a^b f(x) dx$
- Requires many samples to yield a good estimate
- Computers have become reasonably fast to do this

Cinematic Rendering



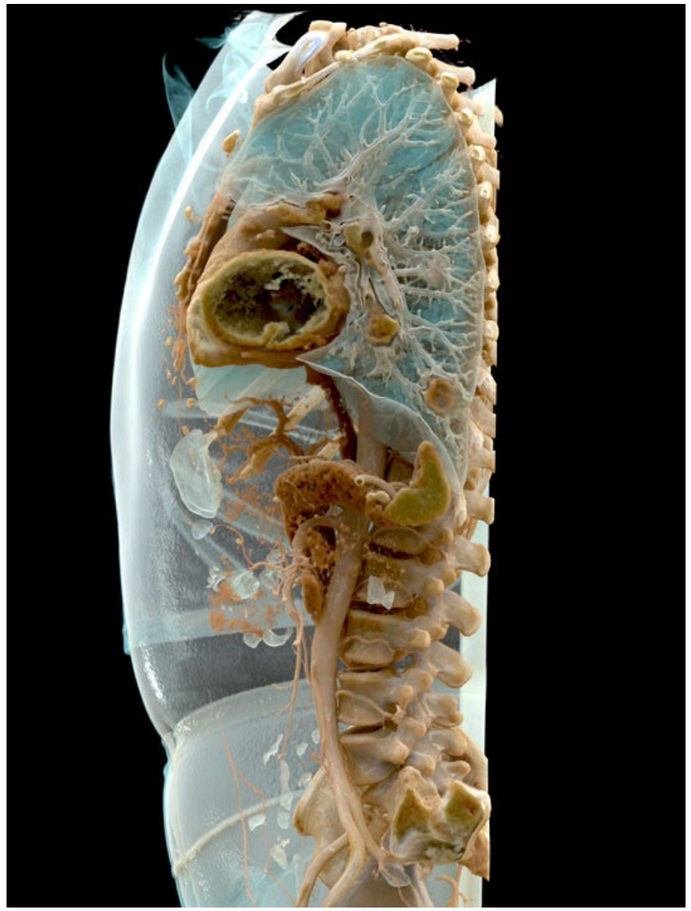
Source: https://www.aec.at/postcity/en/cinematic-rendering/

Cinematic Rendering



Source: https://www.healthcare.siemens.de/magazine/mso-cinematic-rendering.html

Cinematic Rendering



Source: https://www.healthcare.siemens.de/magazine/mso-cinematic-rendering.html

Cinematic Rendering

Source: https://www.siemens.com/innovation/en/home/pictures-of-the-future/health-and-well-being/medical-imaging-cinematic-vrt.html