EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Direct Volume Visualization

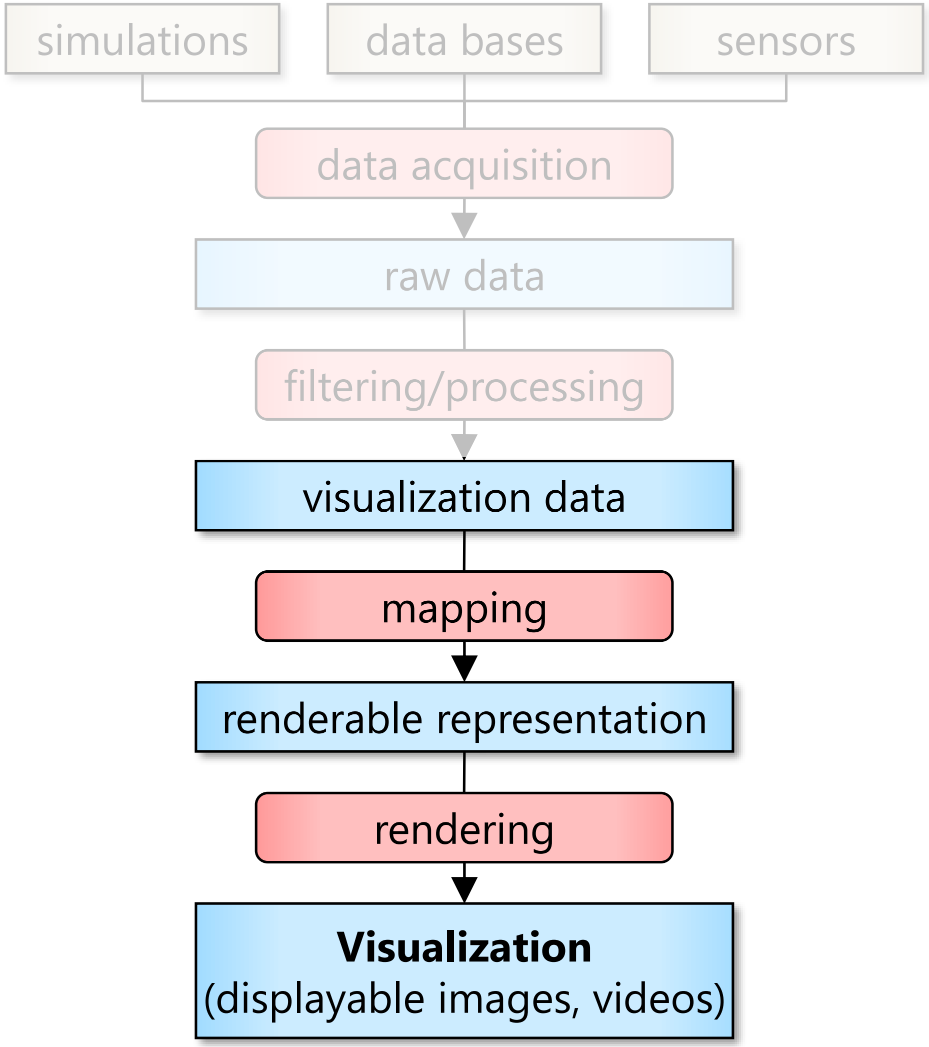
Scientific Visualization – Summer Semester 2021

Jun.-Prof. Dr. **Michael Krone**

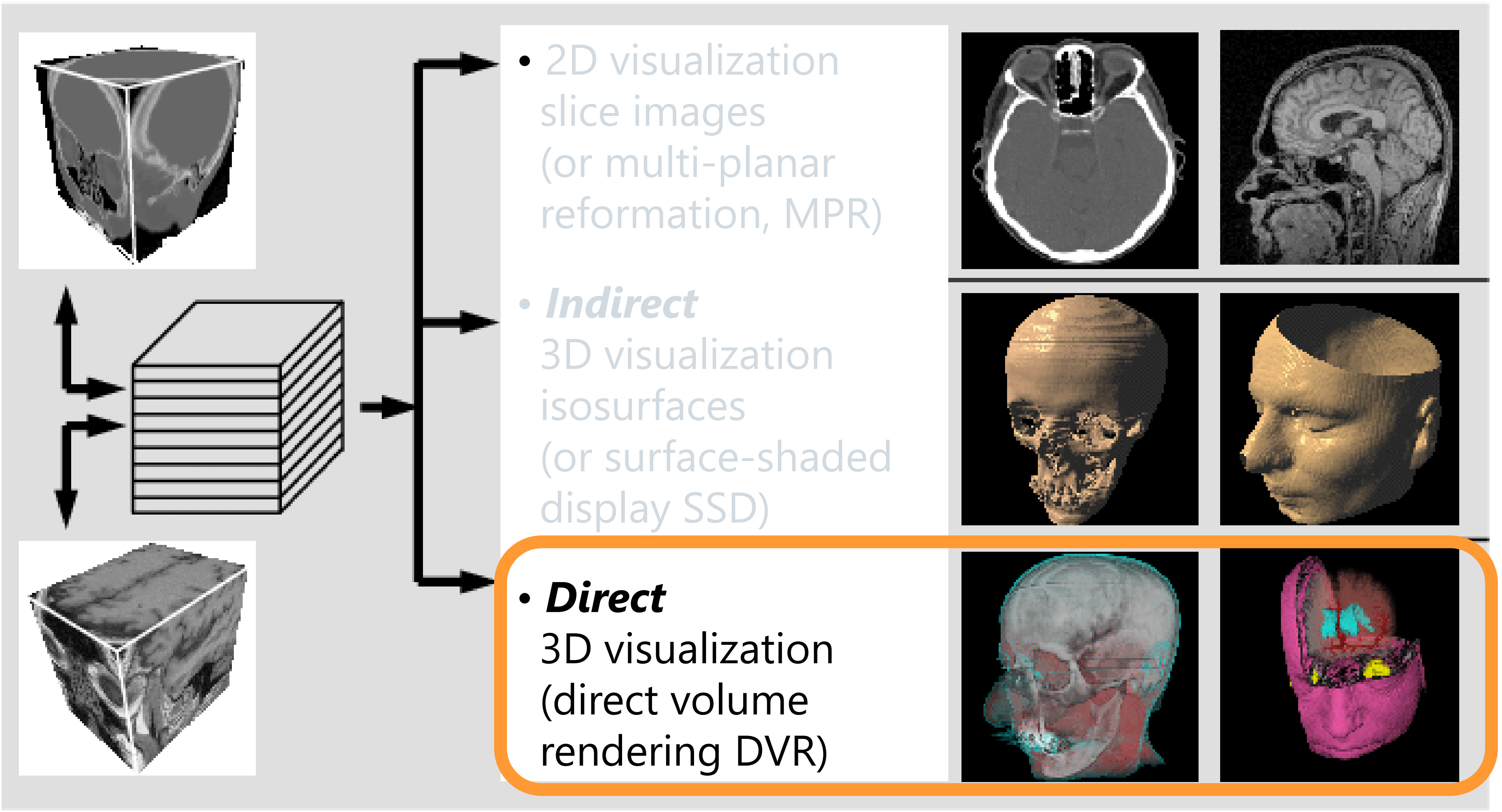
Contents

- Overview
- Volume rendering equation
- Compositing schemes
- Ray casting
- Acceleration techniques for ray casting

Focus:
Second step of visualization pipeline



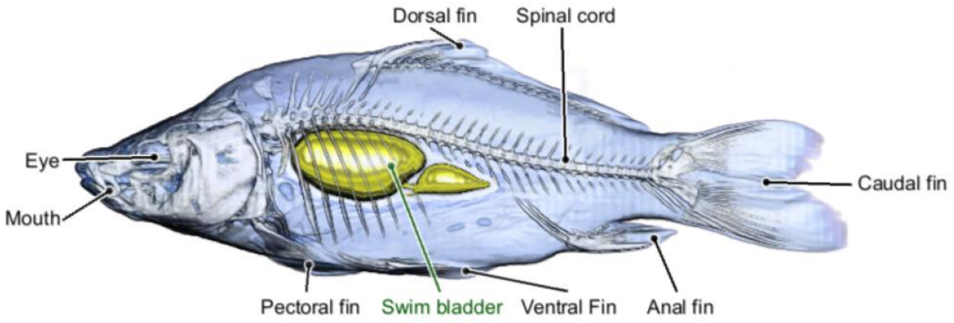
Overview – Volume Visualization



- 2D visualization slice images (or multi-planar reformation, MPR)
- ***Indirect*** 3D visualization isosurfaces (or surface-shaded display SSD)
- ***Direct*** 3D visualization (direct volume rendering DVR)

Overview

- 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



Overview

- Optical model
  - Emission q and absorption κ of light → participating media
- Volume rendering equation

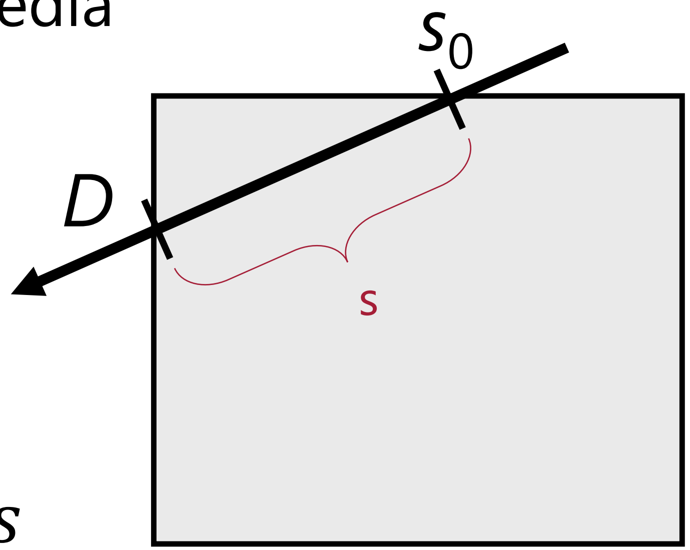
$$\frac{dI(s)}{ds} = q(s) - \kappa(s)I(s)$$

- Volume rendering integral

$$I(D) = I(s_0)T(s_0) + \int_{s_0}^{D} q(s)T(s)ds$$

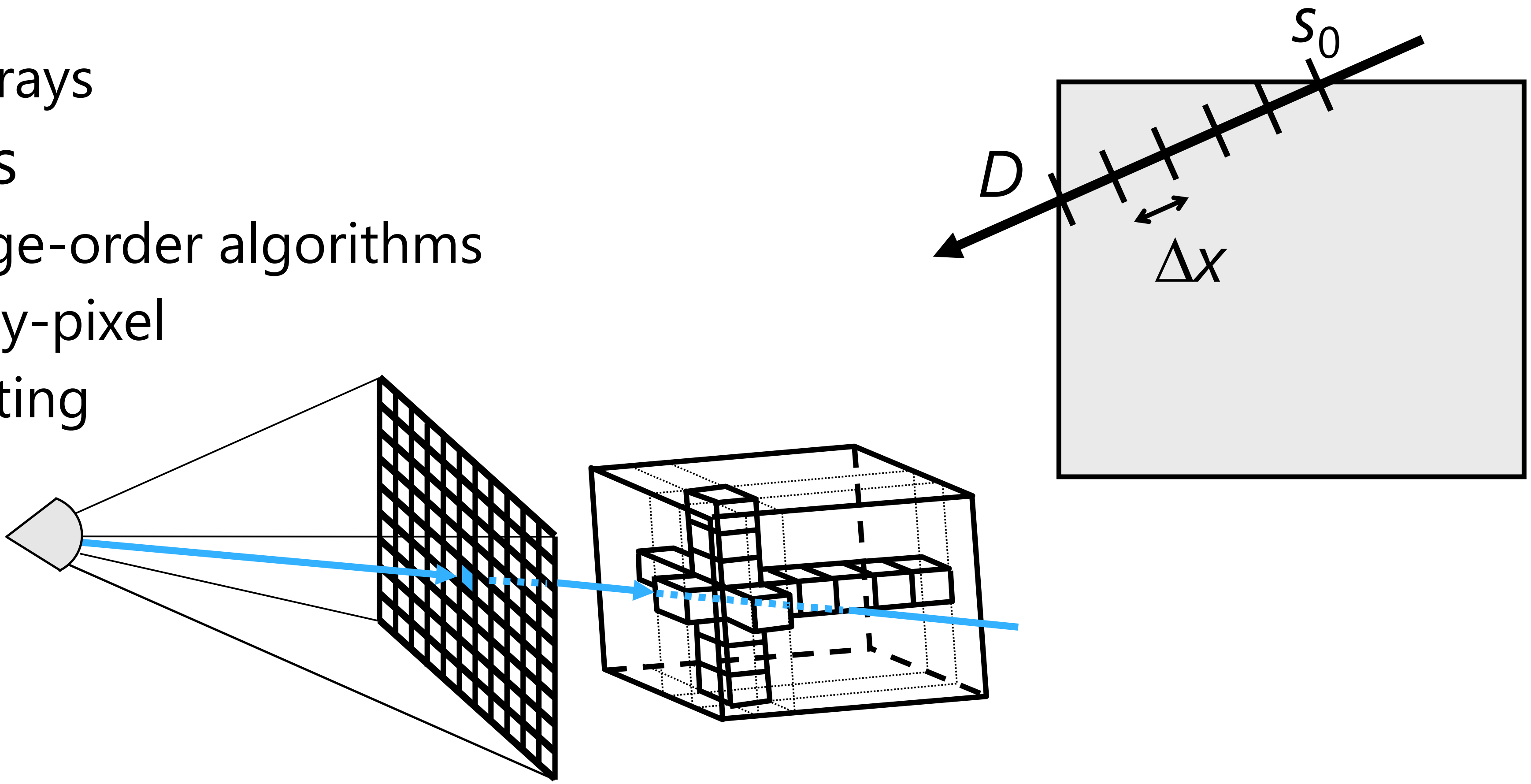
- Transparency

$$T(s) = \exp\left(-\int_{s}^{D} \kappa(t)dt\right)$$



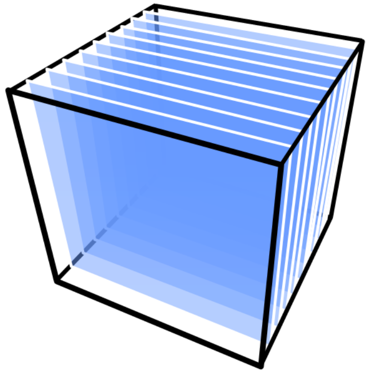
Overview

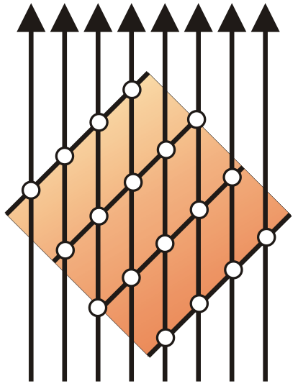
- Numerical approach to compute the volume rendering integral
  - Riemann sum
  - Sampling of light rays
- Backward methods
  - Image space, image-order algorithms
  - Performed pixel-by-pixel
  - **Example:** Ray casting

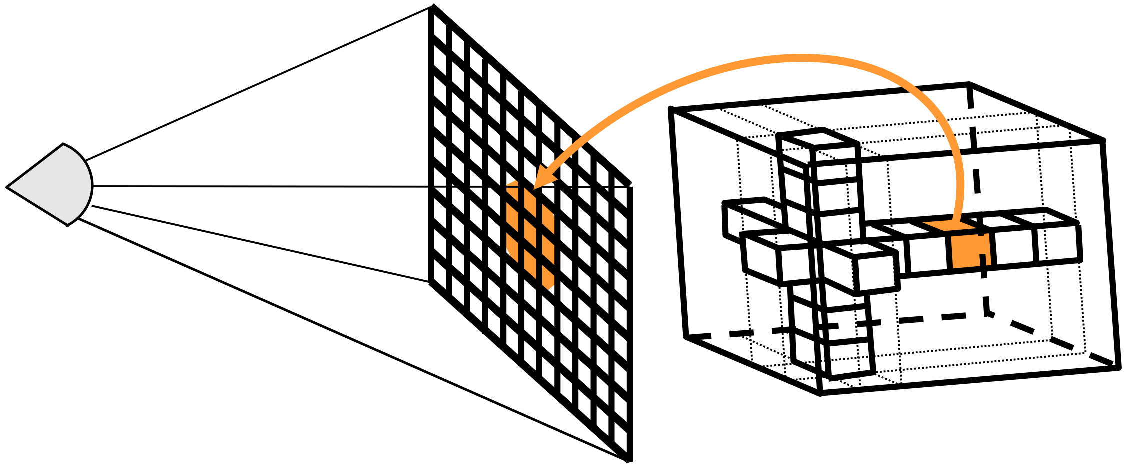


Overview

- Forward methods
  - Object-space, object-order algorithm
  - Cell projection
  - Performed voxel by voxel
  - **Examples:** Slicing, shear-warp, splatting
    → mostly outdated methods, modern volume vis usually uses ray casting





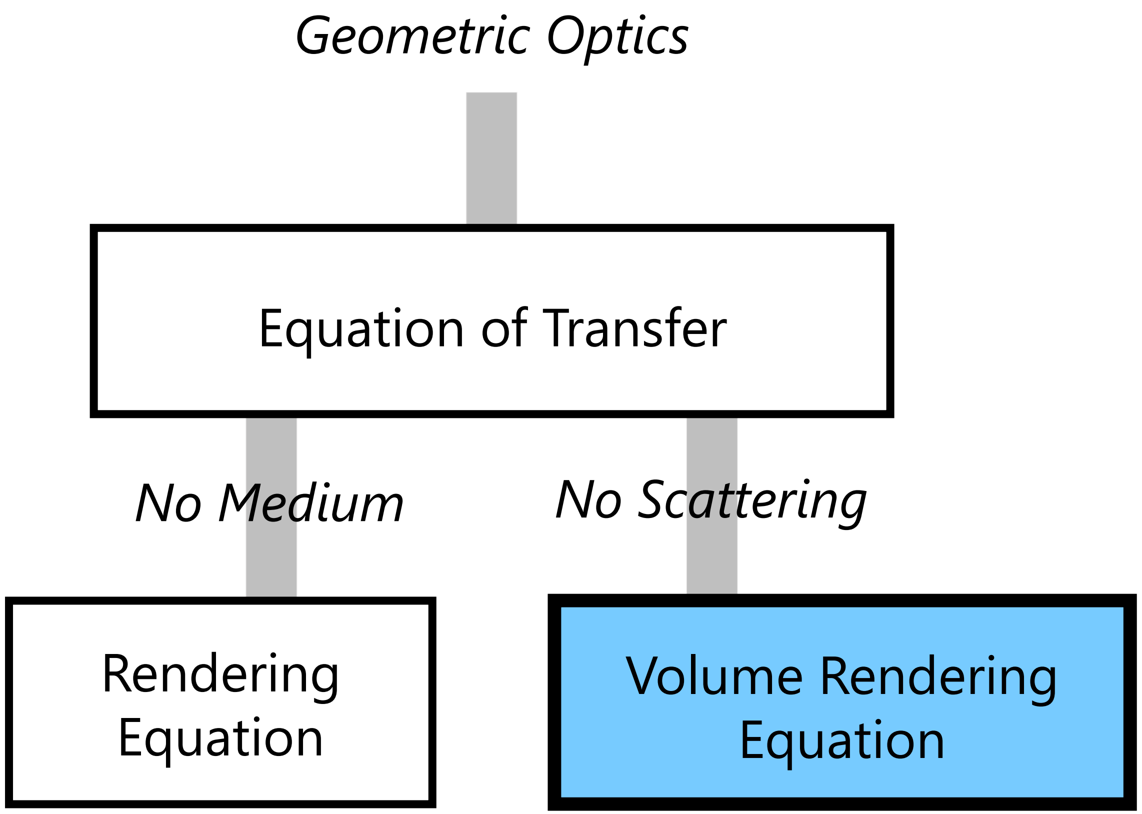


Volume Rendering Equation

- **Goal:** physical model for volume rendering
  - Emission-absorption model
  - Density-emitter model [Sabella 1988]
  - Leads to volume rendering equation
- More general approach:
  - Linear transport theory
  - Equation of transfer for radiation
  - Basis for all rendering methods
- Important aspects:
  - Absorption, Emission, Scattering
  - Participating medium

Volume Rendering Equation

- The grand scheme



Geometric Optics

Equation of Transfer

No Medium

No Scattering

Rendering Equation

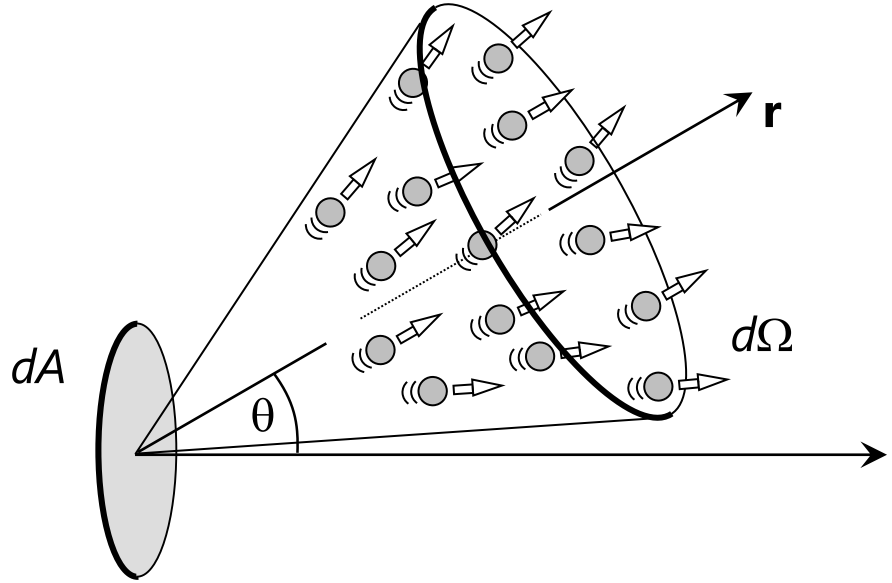
Volume Rendering Equation

Volume Rendering Equation

- Assumptions:
  - Based on a physical model for radiation
  - Geometrical optics
- Neglect:
  - Diffraction, Interference, Wave-character, Polarization
- Interaction of light with matter at the macroscopic scale
  - Describes the changes of specific intensity due to absorption, emission, and scattering
- Based on energy conservation
- Expressed by equation of transfer

Volume Rendering Equation

- Basic quantity of light: radiance I
- Sometimes called specific intensity



$$\delta E = I(\boldsymbol{x},\boldsymbol{r},\nu)\,\overbrace{\cos\theta\, dA}^{\text{projected area element}}\, d\Omega\, d\nu\, dt$$

radiative energy

radiance

position

direction

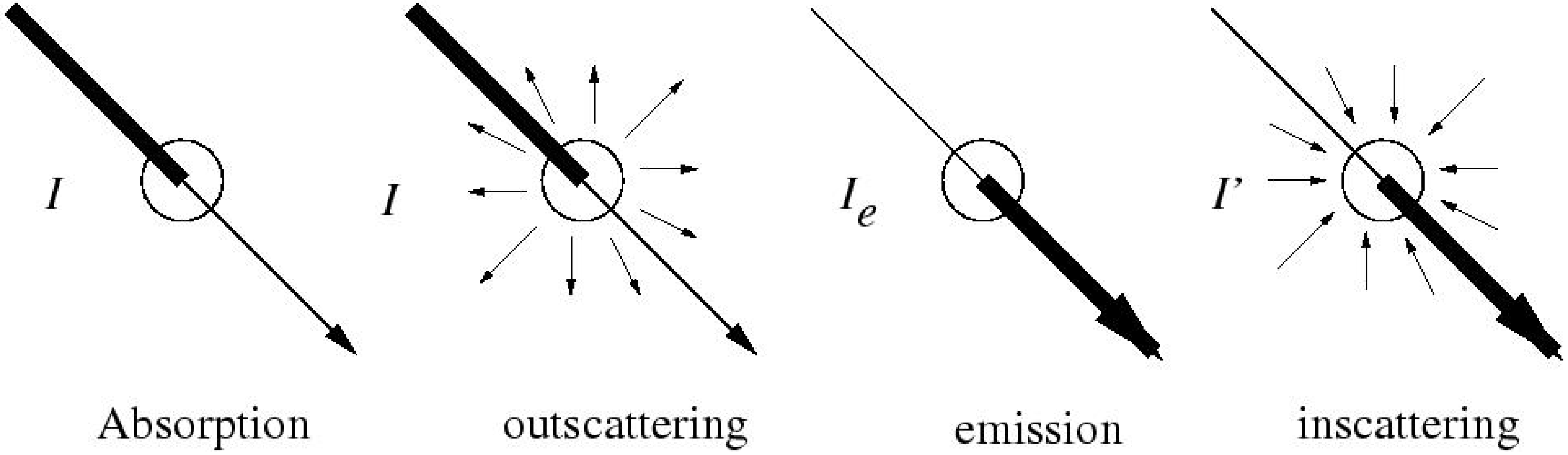
frequency

solid angle

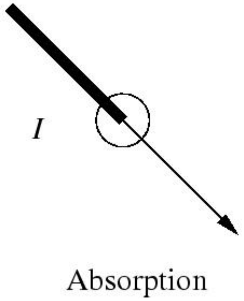
time

Volume Rendering Equation

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

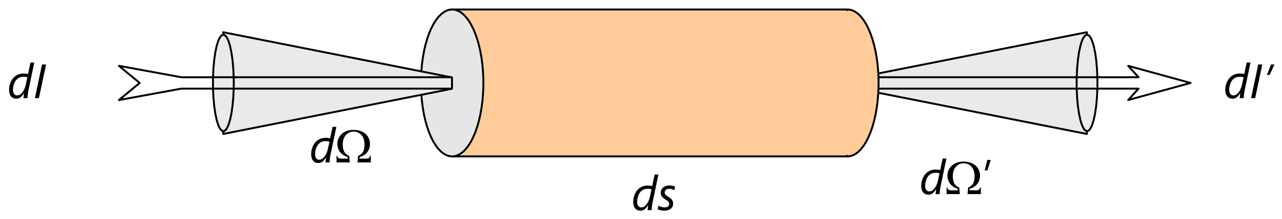


Volume Rendering Equation

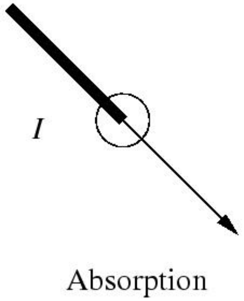


- Absorption
  - Total absorption/extinction coefficient $\chi(\boldsymbol{x}, \boldsymbol{n}, \nu)$
- Loss of radiative energy through a cylindrical volume element:

$$\delta E^{absorption} = \chi(\boldsymbol{x}, \boldsymbol{n}, \nu) I(\boldsymbol{x}, \boldsymbol{n}, \nu) ds dA d\Omega d\nu dt$$

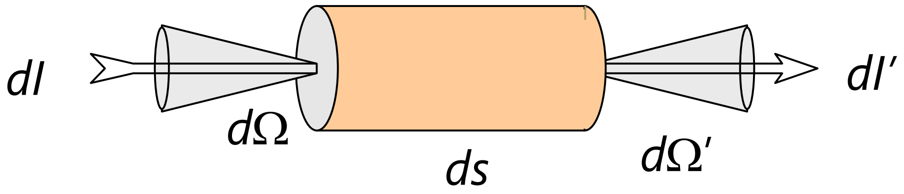


Volume Rendering Equation

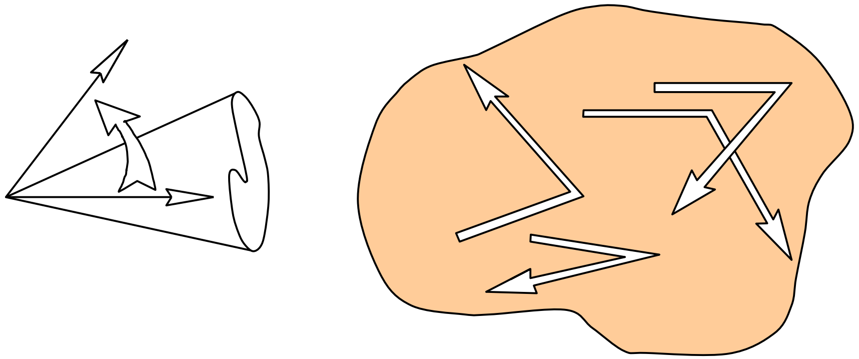


- Total absorption coefficient χ consists of:
  - True absorption coefficient $\kappa(\boldsymbol{x}, \boldsymbol{n}, \nu)$
  - Scattering coefficient $\sigma(\boldsymbol{x}, \boldsymbol{n}, \nu)$

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

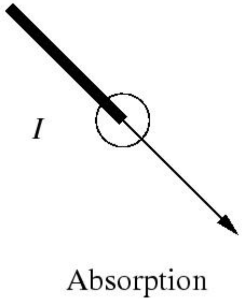


removal of radiative energy by true absorption (conversion to thermal energy)

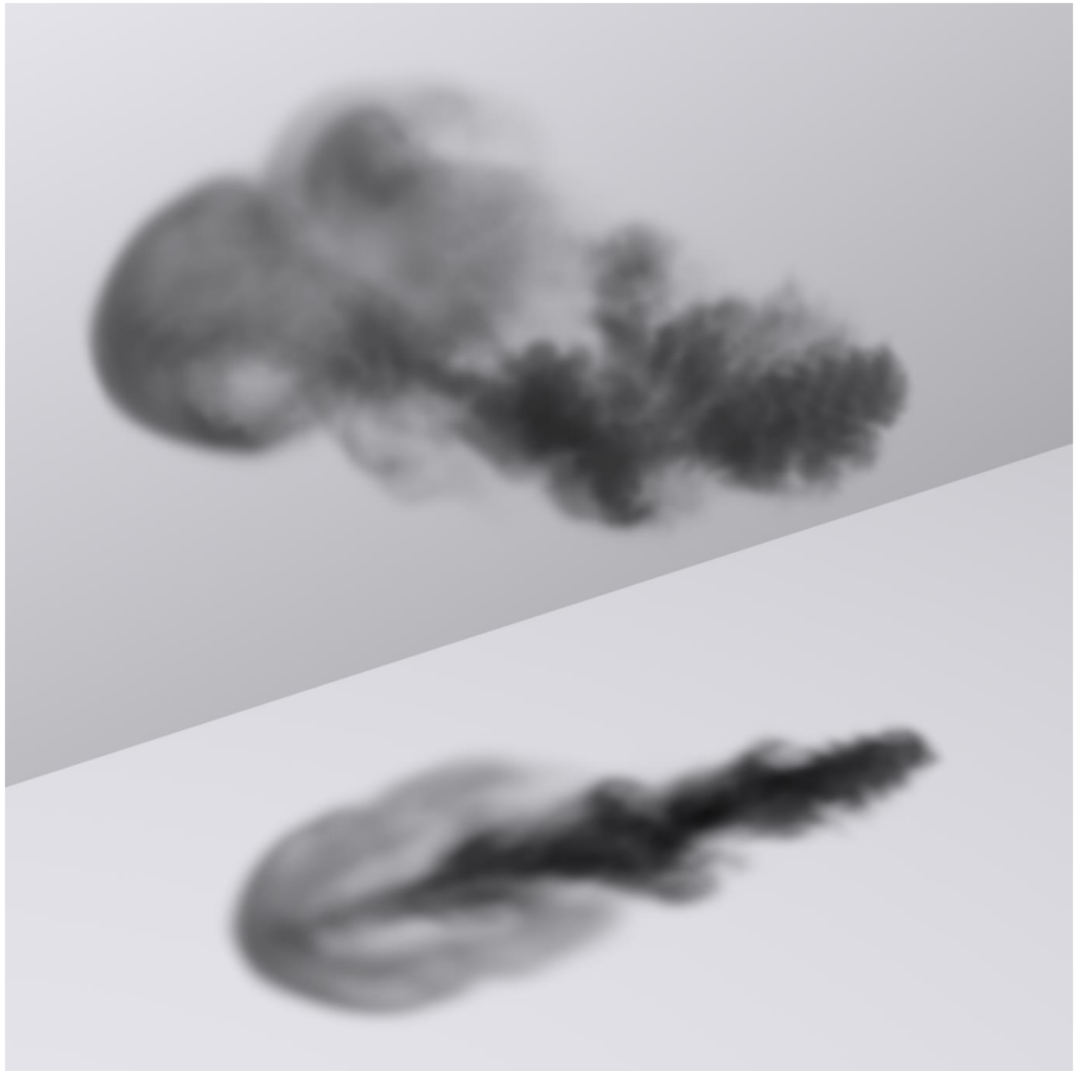


scattering out of solid angle $d\Omega$

Volume Rendering Equation

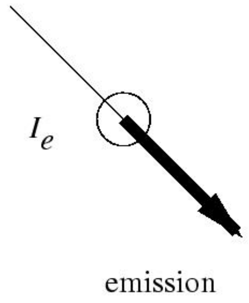


- Effect of absorption



[Pharr, Humphreys, Physically Based Rendering, 2004]

Volume Rendering Equation



- Emission
  - Emission coefficient $\eta(\boldsymbol{x}, \boldsymbol{n}, \nu)$

- Emission of radiative energy within a cylindrical volume element:

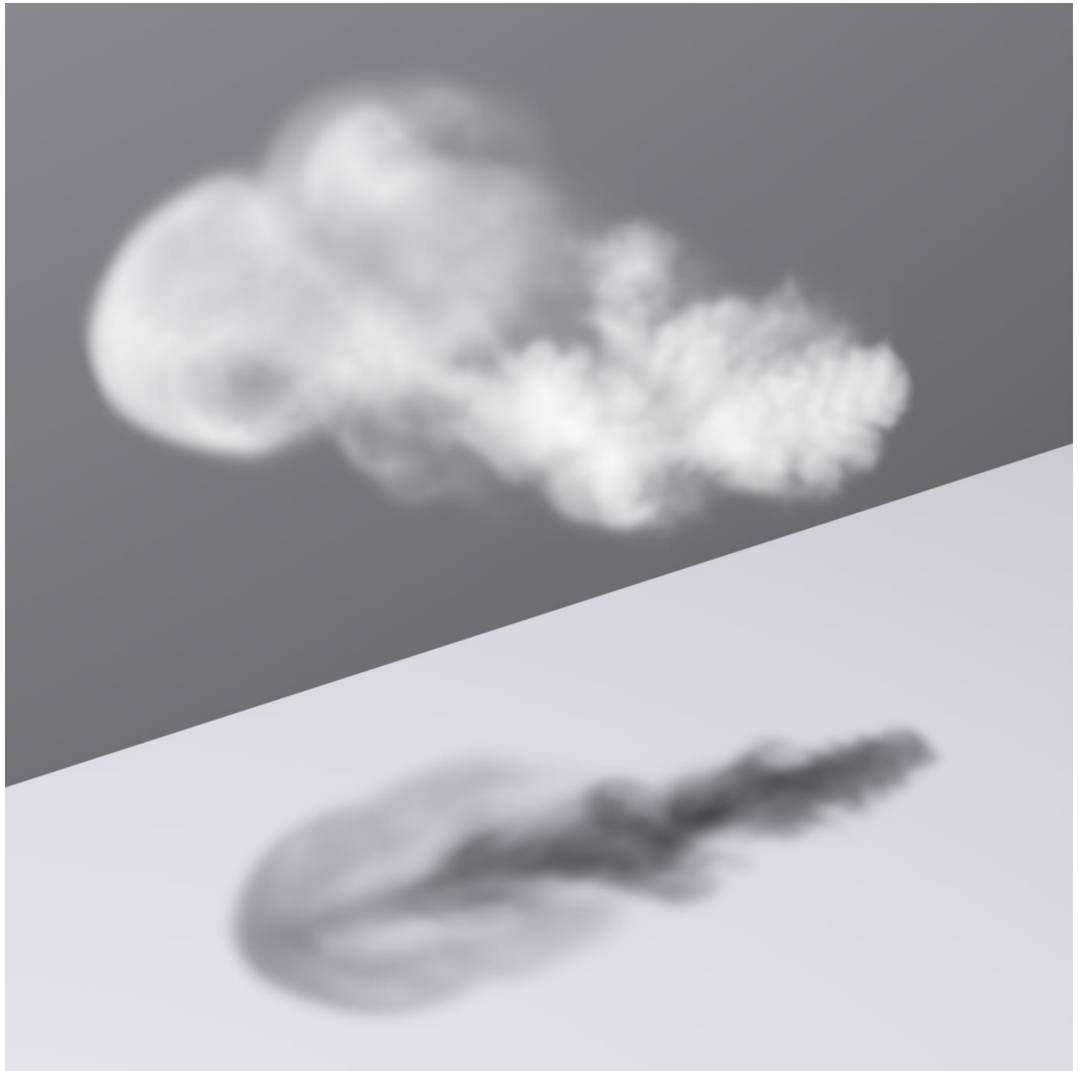
$$\delta E^{emission} = \eta(\boldsymbol{x}, \boldsymbol{n}, \nu) ds dA d\Omega d\nu dt$$

- Consists of two parts:
  - Thermal part or source term $q(\boldsymbol{x}, \boldsymbol{n}, \nu)$
  - Scattering part $j(\boldsymbol{x}, \boldsymbol{n}, \nu)$

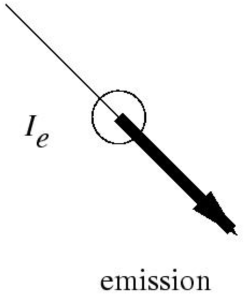
$$\eta = q + j$$

Volume Rendering Equation

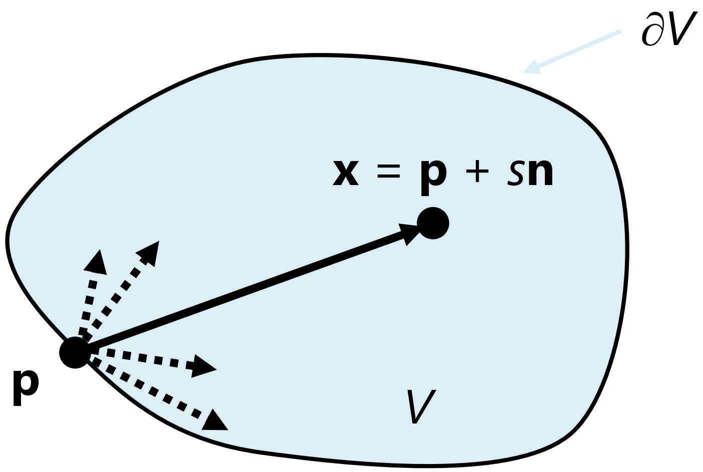
- Effect of emission



[Pharr, Humphreys, Physically Based Rendering, 2004]



Volume Rendering Equation



- Equation of transfer:

  $$\frac{\delta}{\delta s} I(\boldsymbol{x}, \boldsymbol{n}, \nu) = -\chi(\boldsymbol{x}, \boldsymbol{n}, \nu) I(\boldsymbol{x}, \boldsymbol{n}, \nu) + \eta(\boldsymbol{x}, \boldsymbol{n}, \nu)$$

  for derivative along a line $\boldsymbol{x} = \boldsymbol{p} + s\boldsymbol{n}$
- Arbitrary reference point $\boldsymbol{p}$
- Optical depth between 2 points $\boldsymbol{x_1} = \boldsymbol{p} + s_1\boldsymbol{n}$ and $\boldsymbol{x_2} = \boldsymbol{p} + s_2\boldsymbol{n}$ is

  $$\tau_\nu(\boldsymbol{x_1}, \boldsymbol{x_2}) = \int_{s_1}^{s_2} \chi(p + s'\boldsymbol{n}, \boldsymbol{n}, \nu) ds'$$

  - Optical depth: ratio of *incident* to *transmitted* radiant power through material

Volume Rendering Equation

- Using $\tau_\nu(\boldsymbol{x_0}, \boldsymbol{x}) = \tau_\nu(\boldsymbol{x_0}, \boldsymbol{x}') + \tau_\nu(\boldsymbol{x}', \boldsymbol{x})$ leads to the
- Integral form of the equation of transfer

$$\boxed{I(\boldsymbol{x}, \boldsymbol{n}, \nu) = I(\boldsymbol{x_0}, \boldsymbol{n}, \nu) \cdot e^{-\tau_\nu(\boldsymbol{x_0}, \boldsymbol{x})} + \int_{s_0}^{s} \eta(\boldsymbol{x}', \boldsymbol{n}, \nu) \cdot e^{-\tau_\nu(\boldsymbol{x}', \boldsymbol{x})} ds'}$$

- Integral equation because generally η contains I (inscattering)
- Interpretation: Radiation consists of
  - Sum of photons emitted from all points along the line segment,
  - Attenuated by the integrated absorptivity of the intervening medium, and
  - Attenuated contribution from radiation entering the boundary surface

Volume Rendering Equation

- Integral form of the equation of transfer

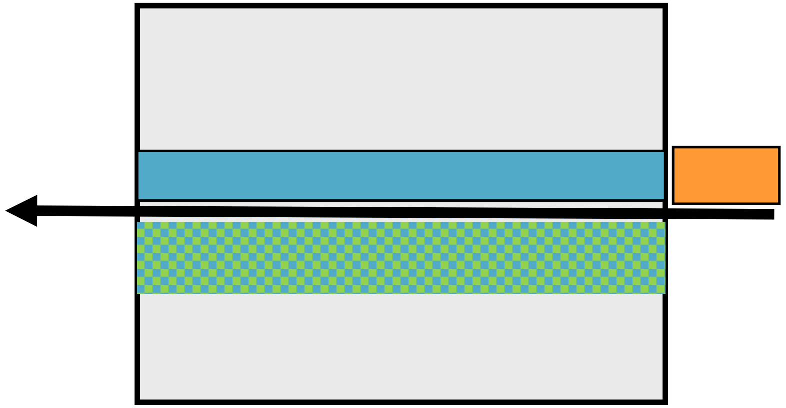
$$I(\boldsymbol{x},\boldsymbol{n},\nu) = I(\boldsymbol{x_0},\boldsymbol{n},\nu)\cdot e^{-\tau_\nu(\boldsymbol{x_0},\boldsymbol{x})} + \int_{s_0}^{s} \eta(\boldsymbol{x}',\boldsymbol{n},\nu)\cdot e^{-\tau_\nu(\boldsymbol{x'},\boldsymbol{x})} ds'$$

🟧 🟦 attenuated contribution from external radiation

🟩 sum of photons emitted along the line segment …

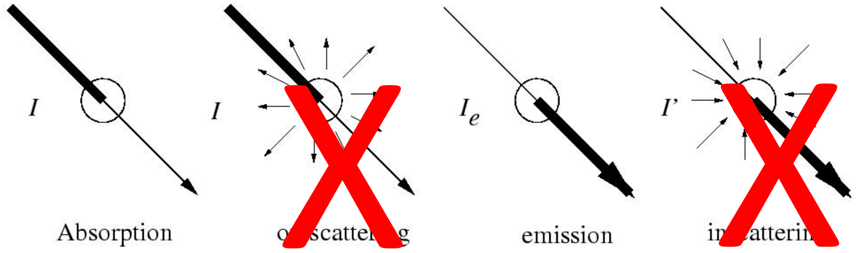
🟦 … attenuated by integrated absorptivity

- $\boldsymbol{x}$: position; $\boldsymbol{n}$: direction, ν: frequency



Volume Rendering Equation

- Special case for most volume rendering approaches:
  - Emission-absorption model
  - Density-emitter model [Sabella 1988]
  - Volume filled with light-emitting particles
  - Particles described by density function
- Simplifications:
  - No scattering
    - Emission coefficient consists of source term only: $\eta = q$
    - Absorption coefficient consists of true absorption only: $\chi = \kappa$
  - No mixing between frequencies (no inelastic effects)



Equation of Transfer for Light

- Volume rendering equation

$$\boxed{I(s) = I(s_0) \cdot e^{-\tau(s_0,s)} + \int_{s_0}^{s} q(s') \cdot e^{-\tau(s',s)} ds'}$$

with optical depth

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

Equation of Transfer for Light

- Discretization of volume rendering equation
  - Discrete steps s_k
  - Often equidistant

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

Equation of Transfer for Light

- Discretization of volume rendering equation *(cont.)*
- Define:
  - Transparency part $\theta_k = e^{-\tau(s_{k-1},s_k)} \approx e^{-\kappa(s_k)\Delta s}$
  - Emission part $b_k = \int_{s_{k-1}}^{s_k} q(s) \cdot e^{-\tau(s,s_k)} ds \approx q(s_k)\Delta s$
- Discretized volume integral:

$$\boxed{\begin{aligned} I(s_n) &= I(s_{n-1}) \cdot \theta_n + b_n = I(s_{n-1}) \cdot (1-\alpha_n) + b_n \\ &= \sum_{k=0}^{n} \left(b_k \prod_{j=k+1}^{n} \theta_j \right) \text{ with } b_0 = I(s_0) \end{aligned}}$$

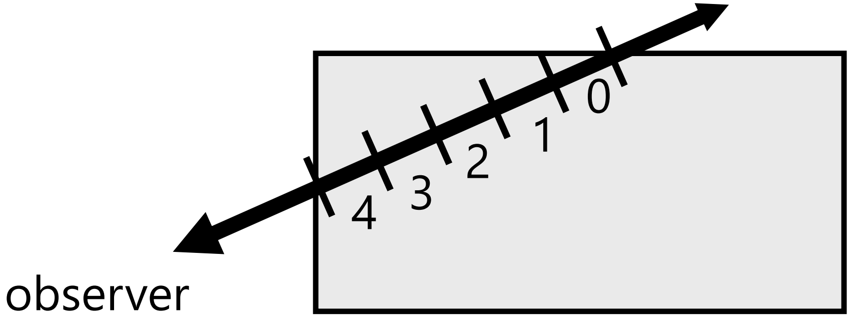
over operator with opacity $\alpha = (1\text{-}\theta)$

Code:

```
intensity = b_0;
for (k = 1 ; k <= n; k = k + 1 )
    intensity = theta_k * intensity + b_k;
```

Compositing

- Compositing = iterative computation of discretized volume integral
- Traversal strategies
  - Front-to-back
  - Back-to-front

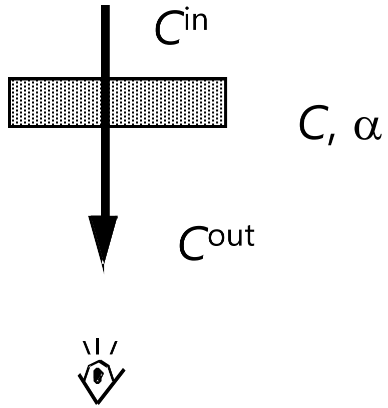


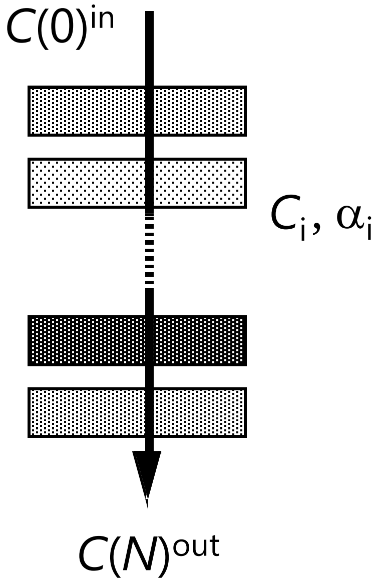
- Back-to-front compositing
  - Directly derived from discretized integral: $I(s_n) = I(s_{n-1}) \cdot (1 - \alpha_n) + b_n$
  - Just different notation: $C^{out} = C^{in} \cdot (1 - \alpha) + C'$
  - Colors C and opacity α are assigned with transfer function
  - C’ is pre-multiplied color: $C' = C \cdot \alpha$ (often denoted as C too)

Compositing

- Back-to-front compositing *(cont.)*
  - Over operator [Porter & Duff 1984]
- Compositing equation:

$$C^{out} = C^{in} \cdot (1 - \alpha) + C' \qquad C(i)^{in} = C(i-1)^{out}$$





Compositing

- Front-to-back compositing
  - Reverse the order of summation
  - From

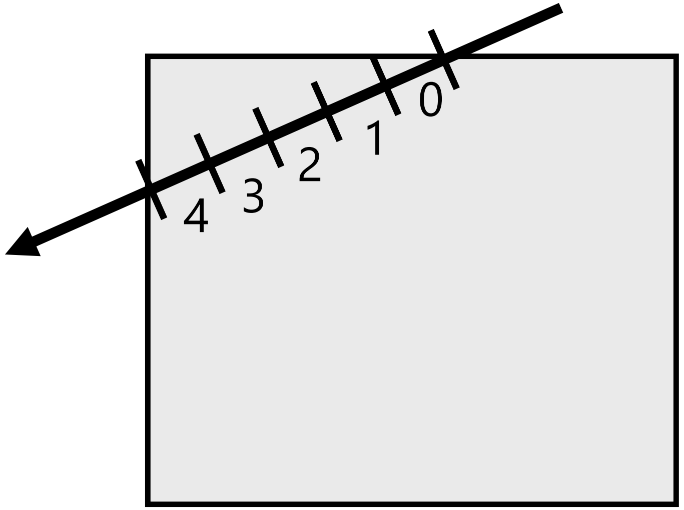
$$I(s_n) = \sum_{k=0}^{n} \left(b_k \prod_{j=k+1}^{n} \theta_j \right)$$

obtain

$$I(s_n) = I(s_{n+1}) + T_{n+1} b_n$$

$$T_n = T_{n+1} \theta_n$$

  - with accumulated transparency T_n

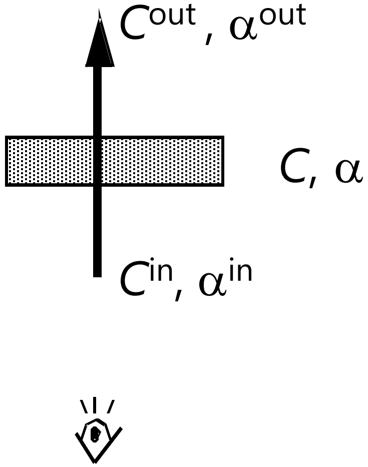


Compositing

- Front-to-back compositing *(cont.)*
  - Needs to maintain α^{in}
  - Most often used in ray casting
  - Allows for early ray termination (stop if α^{out} close enough to 1)
- Compositing equation:

$$C^{out} = C^{in} + \left(1 - \alpha^{in}\right) C'$$

$$\alpha^{out} = \alpha^{in} + \left(1 - \alpha^{in}\right) \alpha$$

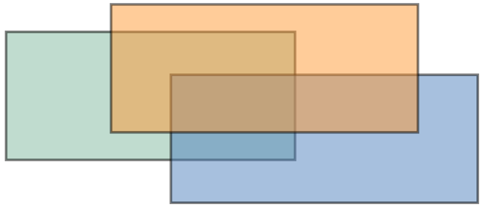


Compositing

- Associated colors
    - Color contributions are already weighted by their corresponding opacity
    - Also called pre-multiplied colors
- Non-associated colors: $C' \rightarrow C\alpha \quad (C \rightarrow C\alpha)$
    - Just substitute in compositing equations
- Yields the same results as associated colors (on a continuous level)
- **Example:** back-to-front compositing with non-associated colors:

$$C^{out} = C^{in} \cdot (1 - \alpha) + C\alpha$$

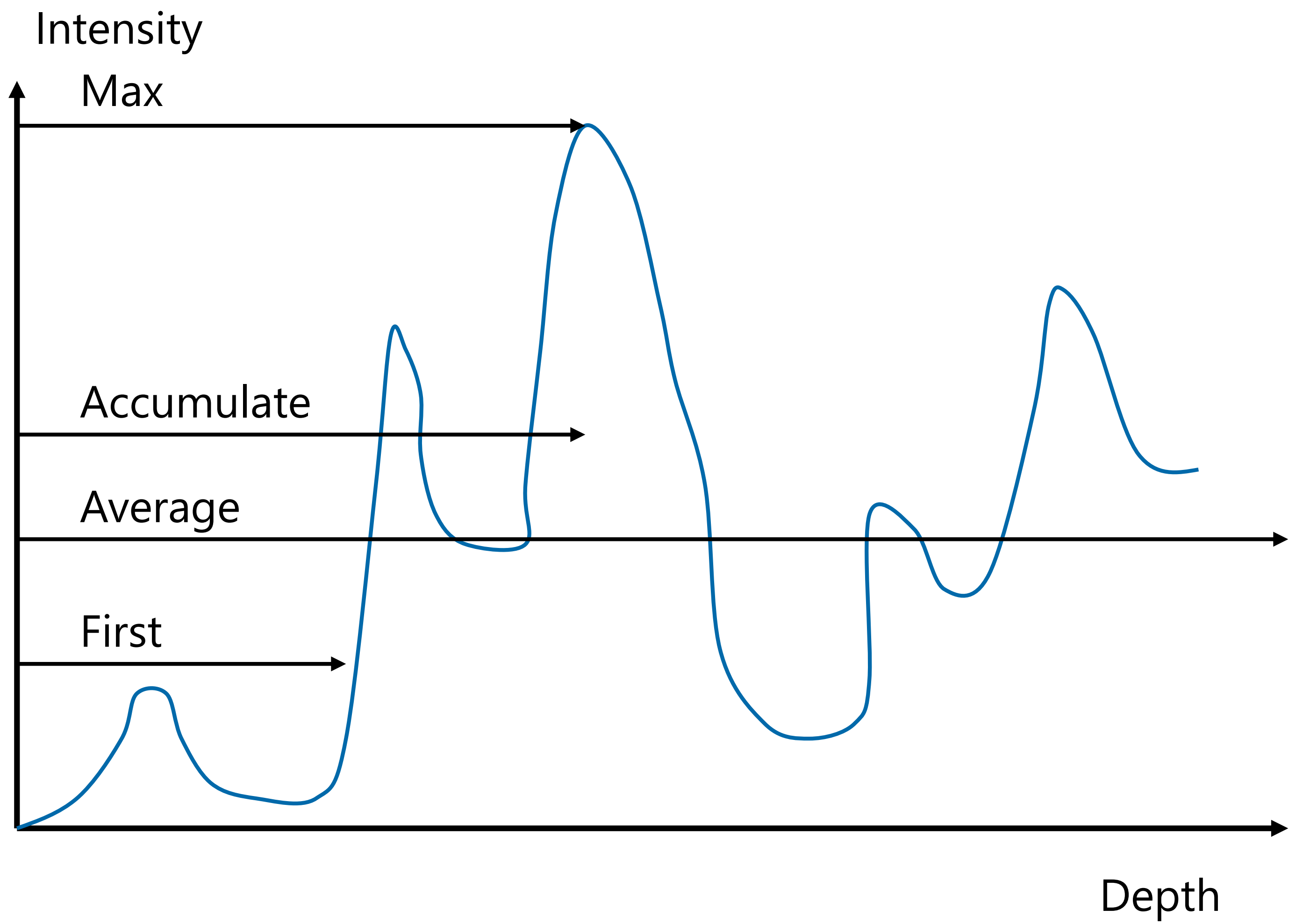
- Standard OpenGL blending for semi-transparent surfaces



Compositing

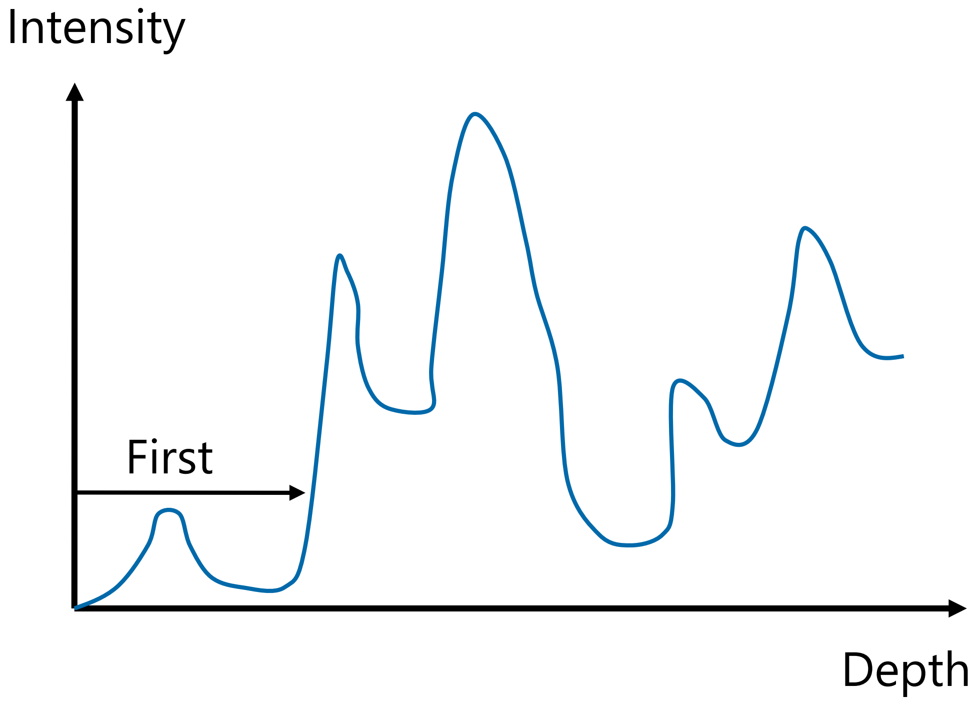
- So far: accumulation scheme
- Variations of composition schemes, e.g.:
  - First
  - Average
  - Maximum intensity projection

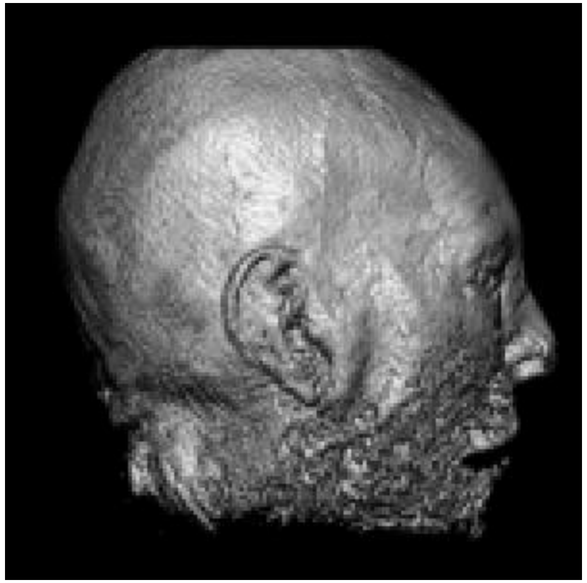
Compositing



Compositing

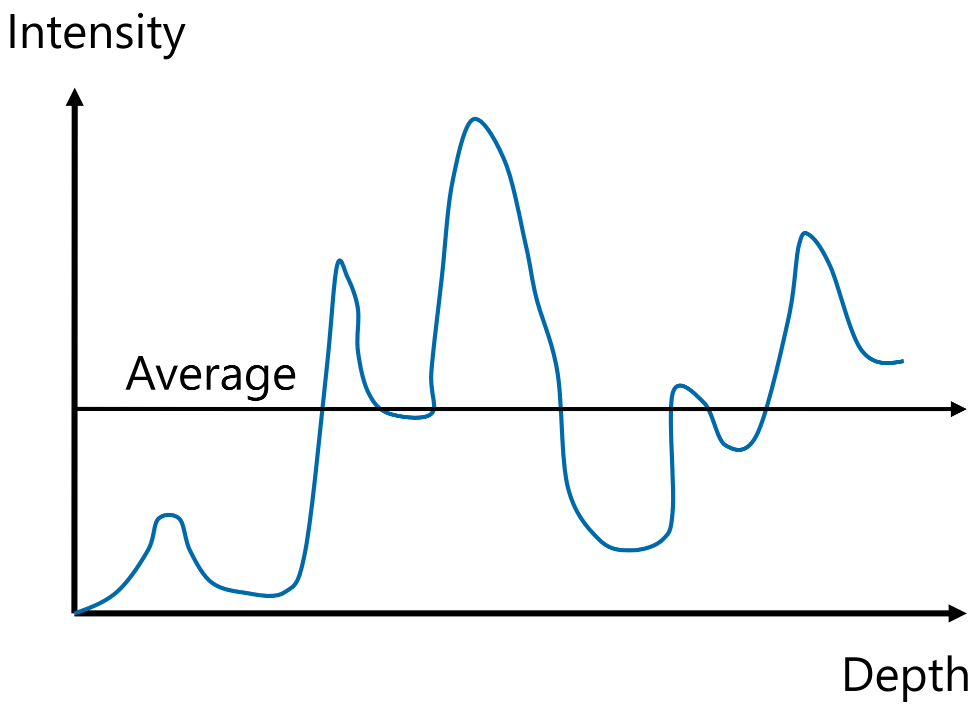
- Compositing: First (above a certain intensity)
- Extracts isosurfaces

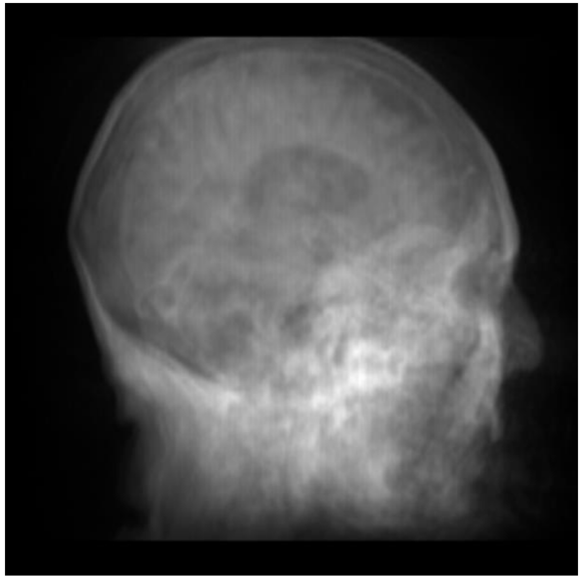




Compositing

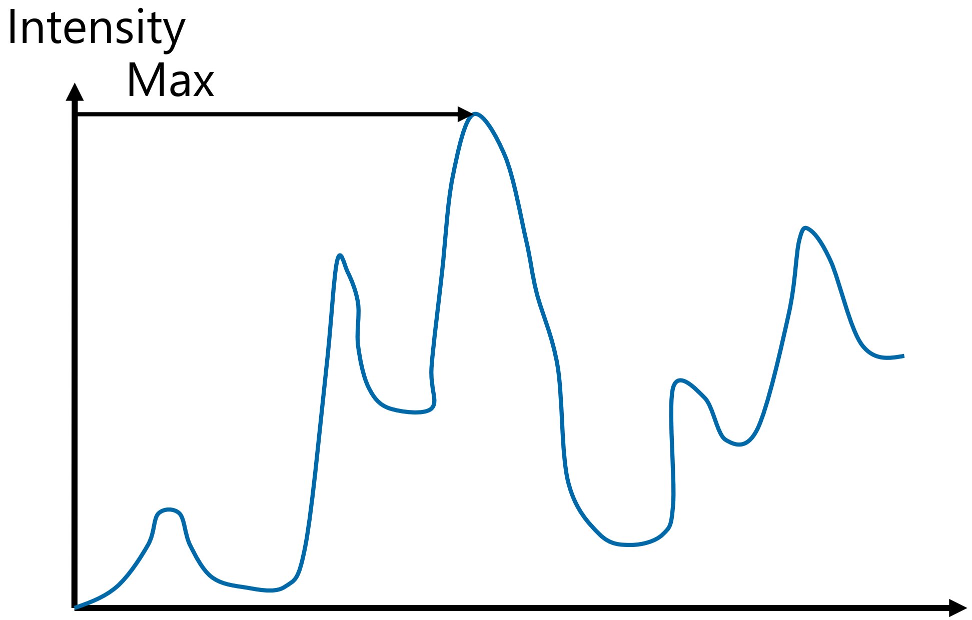
- Compositing: Average
- Produces basically an X-ray picture

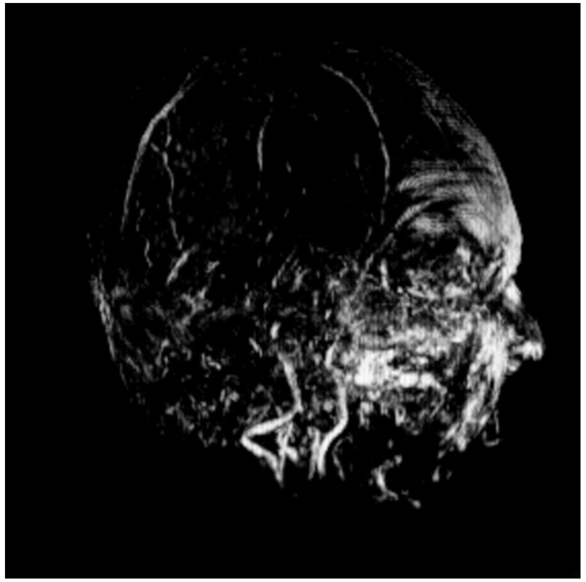




Compositing

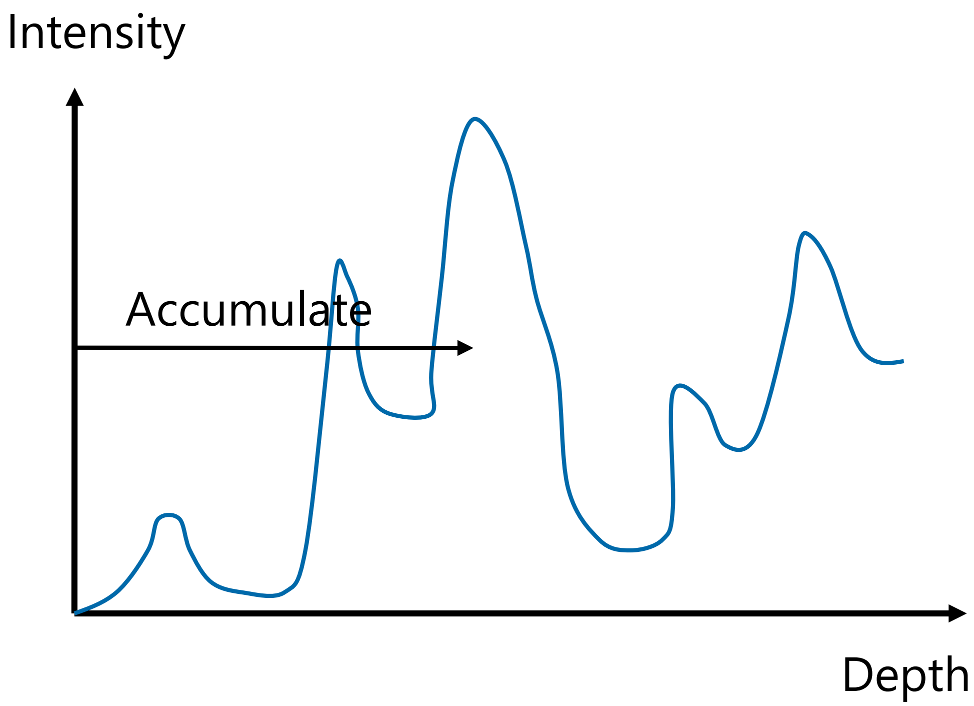
- Maximum Intensity Projection (MIP)
- Often used for magnetic resonance or CT angiograms
- Good to extract vessel structures

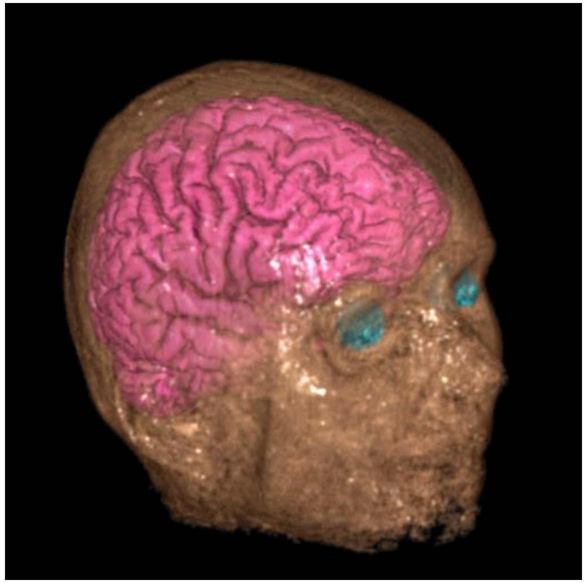




Compositing

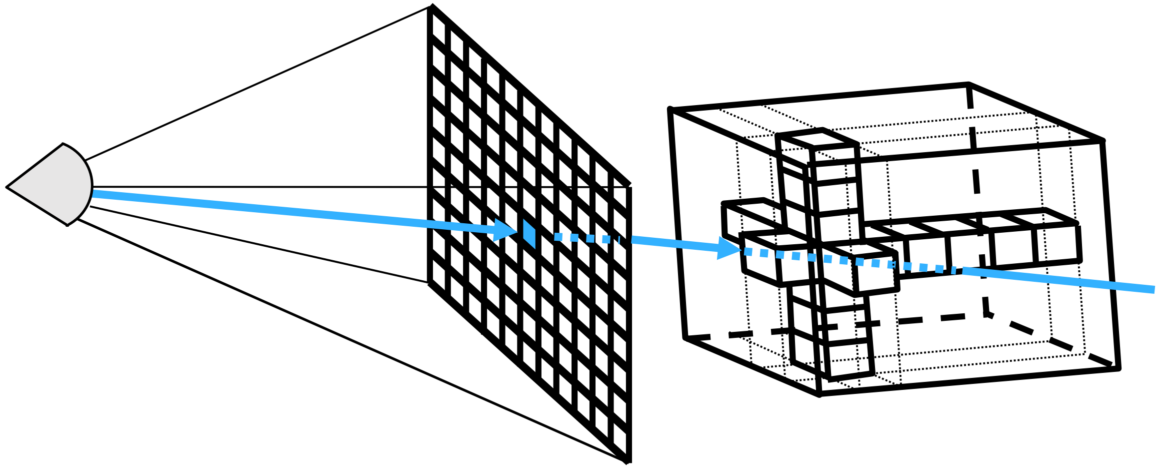
- Compositing: Accumulate
- Emission-absorption model
- Make transparent layers visible (see volume classification)





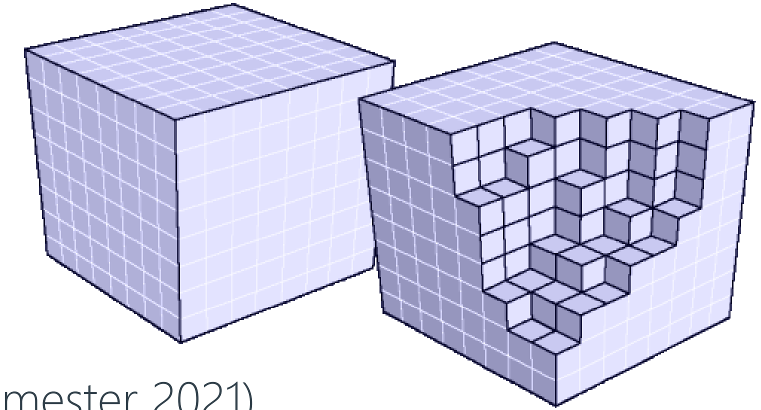
Ray Casting

- Similar to ray tracing in surface-based computer graphics
- In volume rendering we only deal with primary rays; hence: *ray casting*
- Natural image-order technique
- As opposed to surface graphics – how do we define and calculate the ray/object intersection?



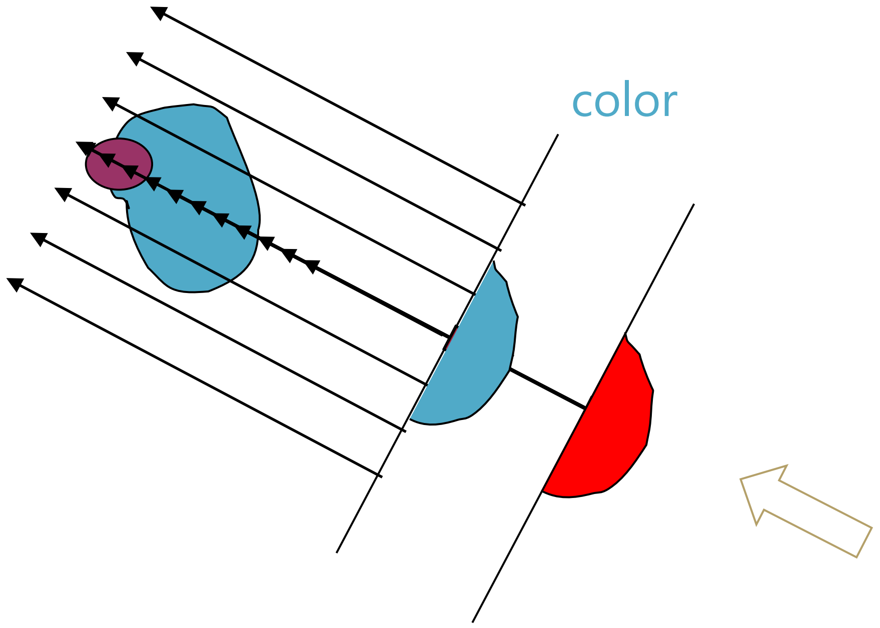
Ray Casting

- Since we have no surfaces, we need to carefully step through the volume
- A ray is cast into the volume, sampling the volume at certain intervals
  - Sampling intervals usually are equidistant, but don't have to be
- At each sampling location, a sample is interpolated / reconstructed from the voxel grid → *also called "ray marching"*
  - Popular filters are: nearest neighbor (box), trilinear (→ GPU), or more sophisticated (Gaussian, cubic spline)
- First: Ray casting in uniform grids
  - Implicit topology
  - Simple interpolation schemes



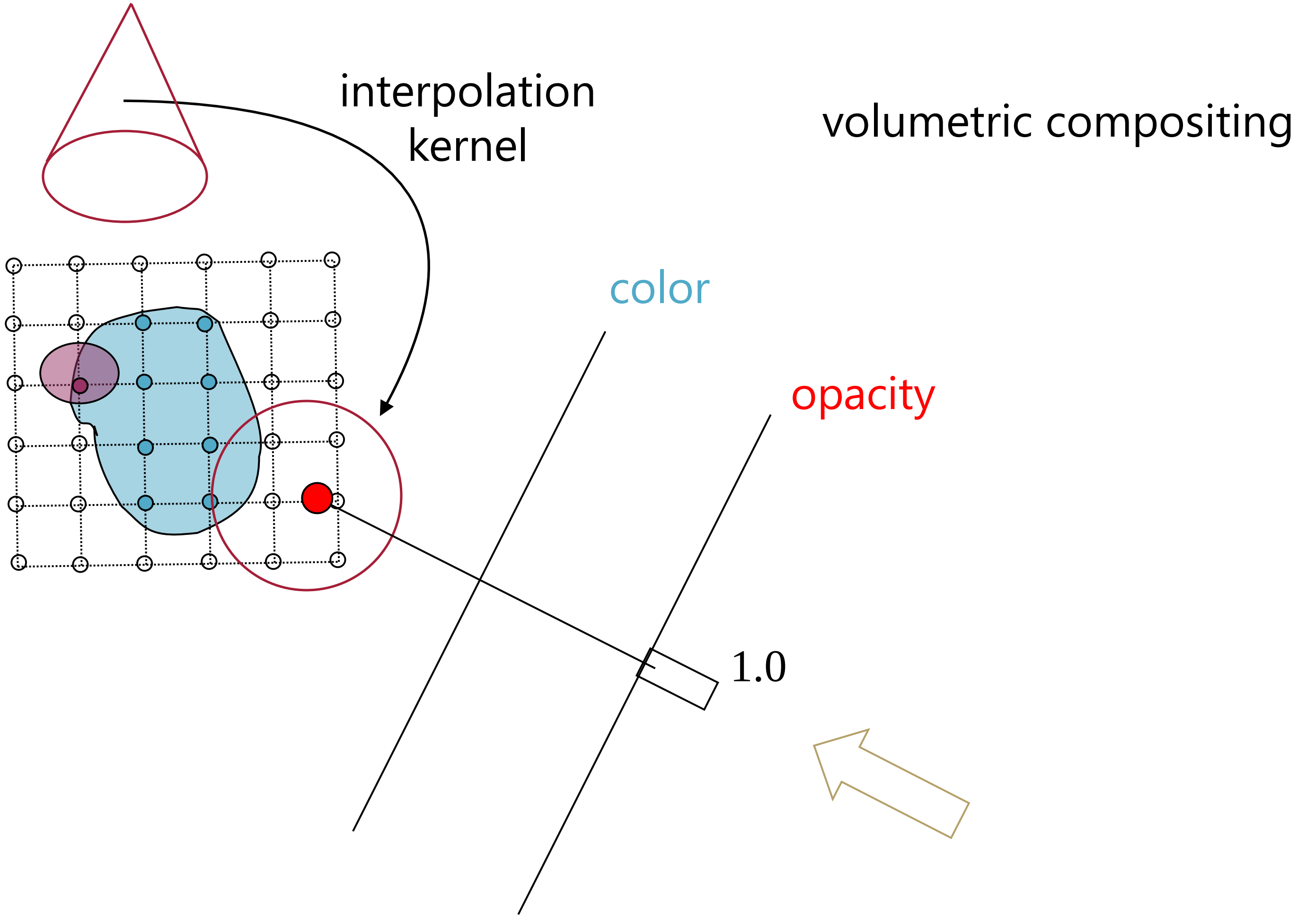
Ray Casting / Ray Marching

- Volumetric ray integration:
  - Tracing of rays
  - Accumulation of color and opacity along ray: compositing (front to back)



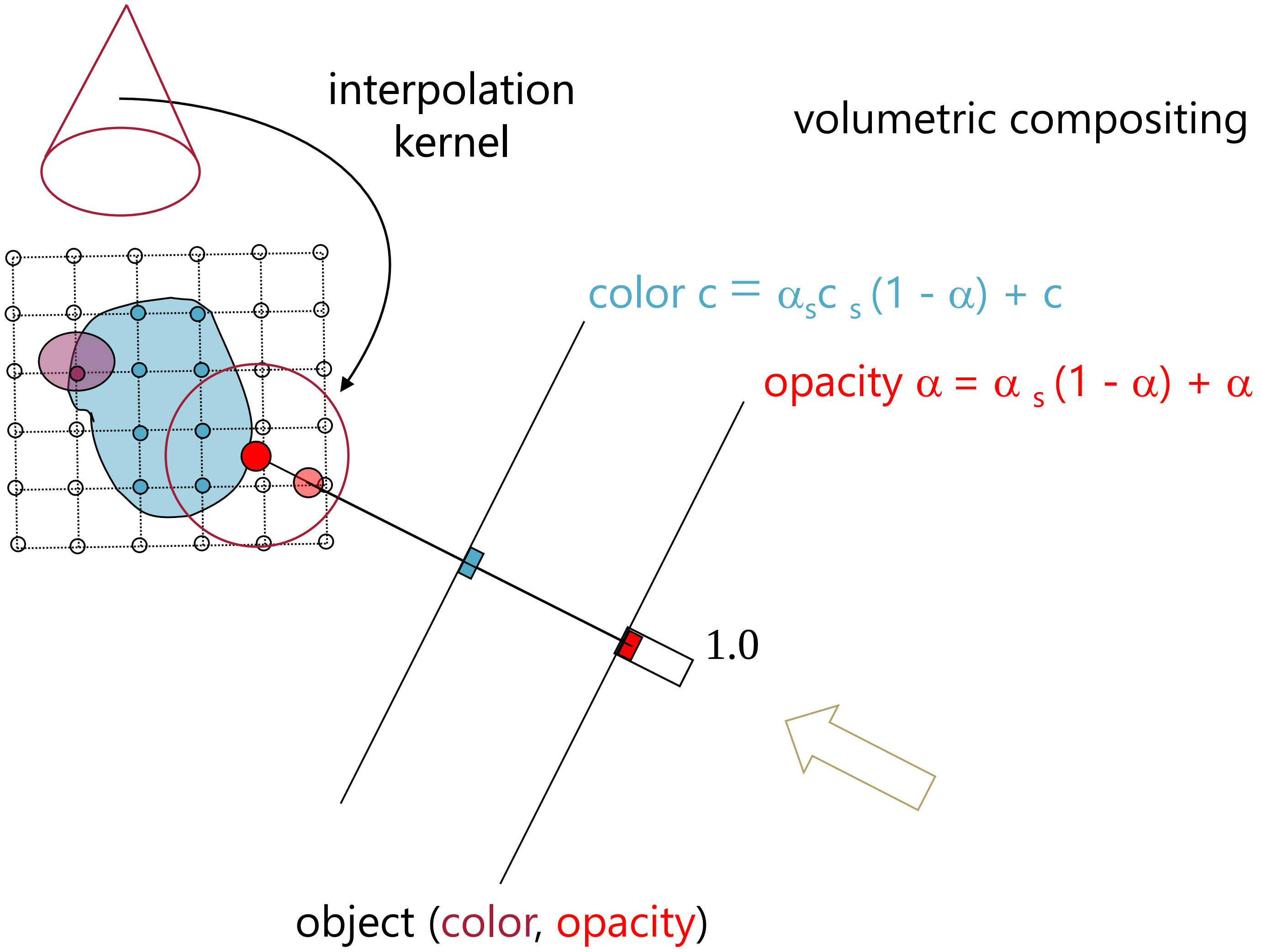
object (color, opacity)

Ray Casting / Ray Marching



object (color, opacity)

Ray Casting / Ray Marching



interpolation kernel

volumetric compositing

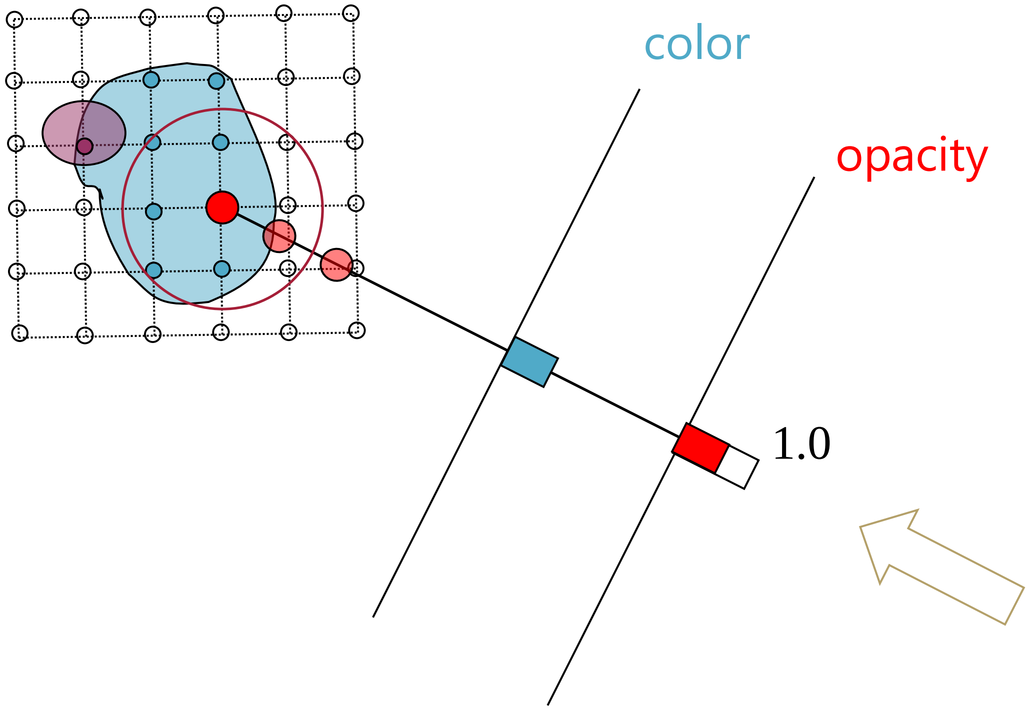
color $c = \alpha_s c_s (1 - \alpha) + c$

opacity $\alpha = \alpha_s (1 - \alpha) + \alpha$

object (color, opacity)

Ray Casting / Ray Marching

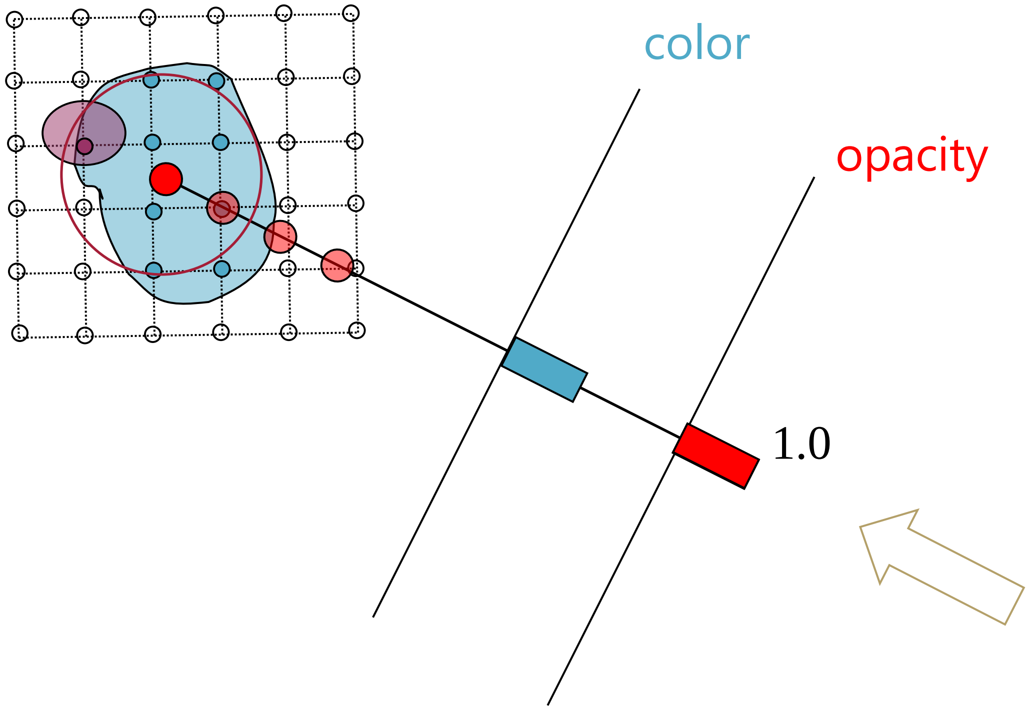
volumetric compositing



object (color, opacity)

Ray Casting / Ray Marching

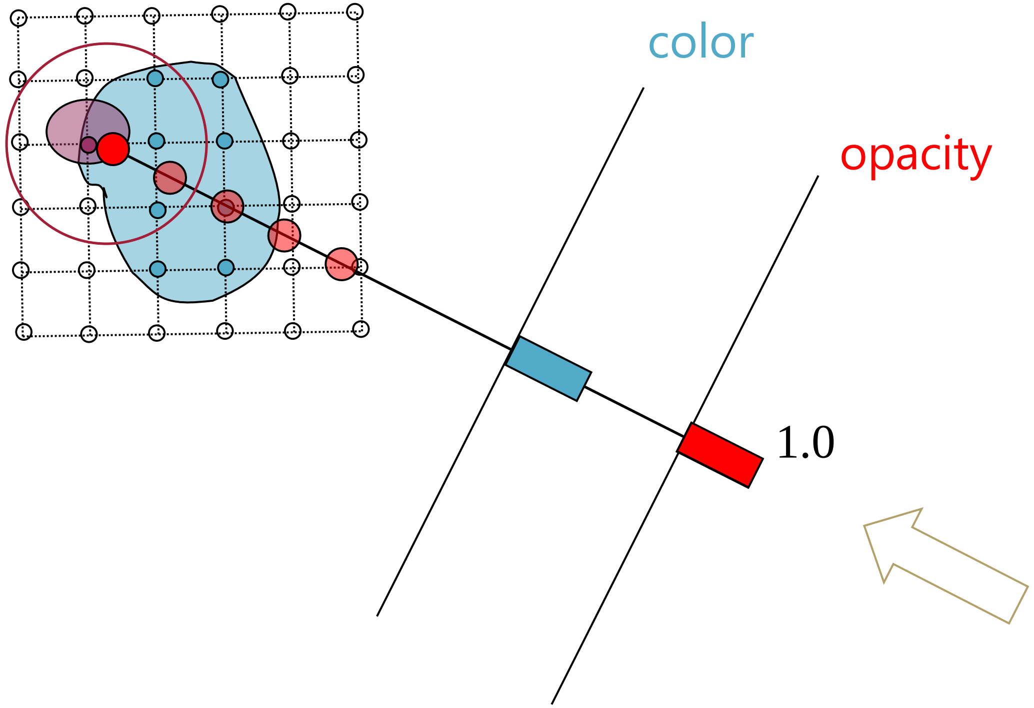
volumetric compositing



object (color, opacity)

Ray Casting / Ray Marching

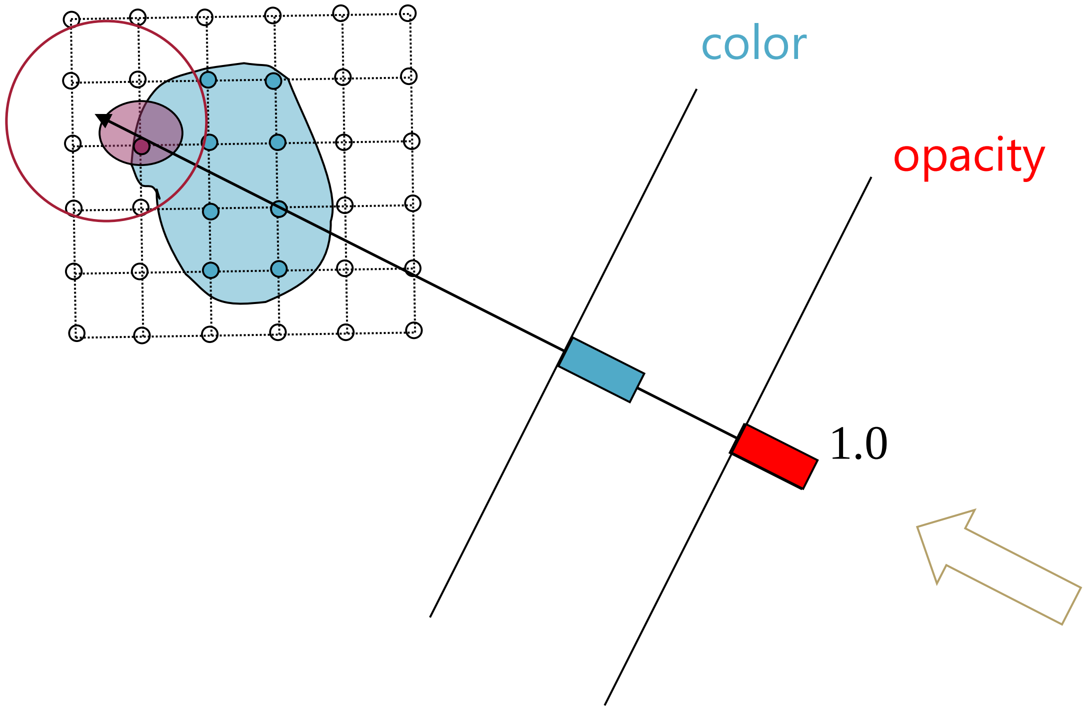
volumetric compositing



object (color, opacity)

Ray Casting / Ray Marching

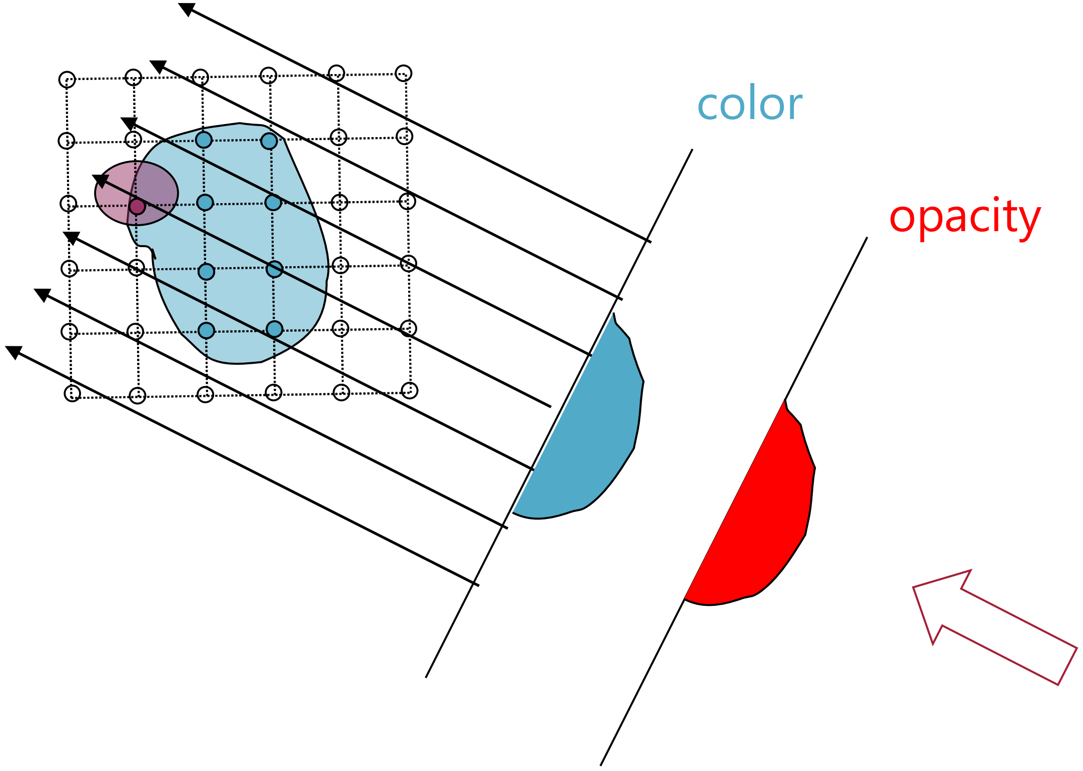
volumetric compositing



object (color, opacity)

Ray Casting / Ray Marching

volumetric compositing



object (color, opacity)

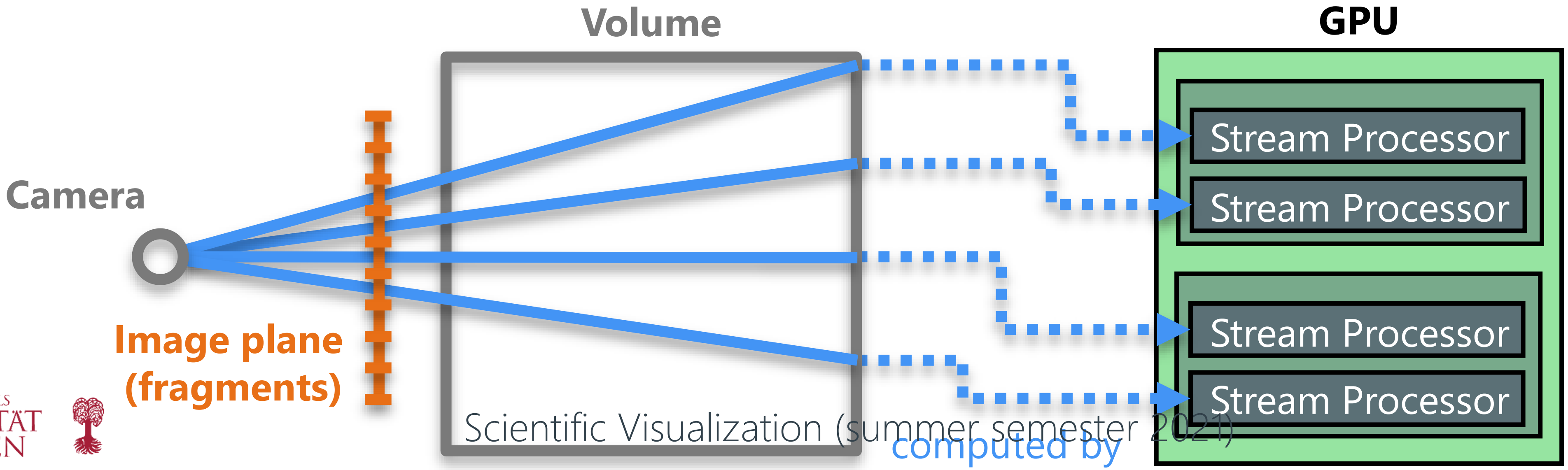
Ray Casting / Ray Marching

- How is color and opacity at each integration step determined?
- Opacity and (emissive) color in each cell according to classification
- Additional color due to external lighting
  - According to volumetric shading (e.g., Blinn-Phong, normal from gradient)
- No shadowing, no secondary effects captured so far
  - Requires additional steps, e.g., secondary rays

- Rays can be traced completely independently from each other
  - Straightforward parallelization on multicore CPUs and GPUs
  - One ray can be computed by one thread

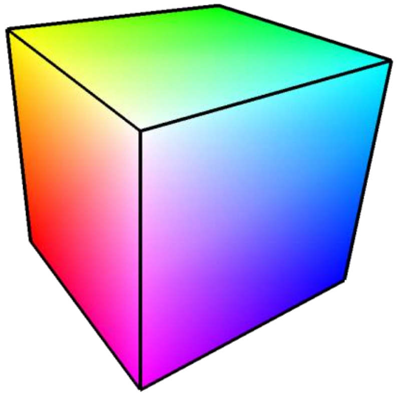
Acceleration Techniques for Ray Casting

- GPUs can be used for ray casting
- Essential idea
  - (Fragment) shader loop implements ray marching
- Benefits from
  - High processing speed and parallelism of GPUs
  - Built-in trilinear interpolation in 3D textures

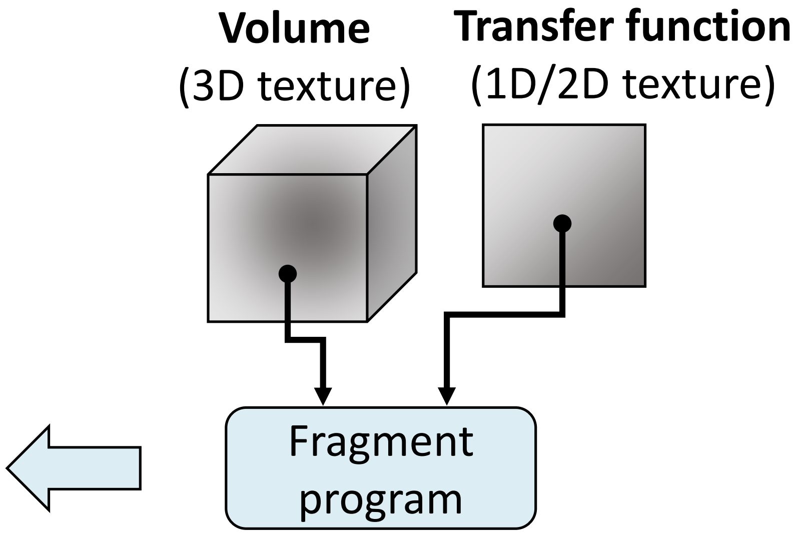


GPU Ray Casting/Marching: Ray Traversal

- Single-pass approach
  - Complete computation in a single fragment program
  - Shader loop to step along ray
- Algorithm
  - Render front faces of volume bounding box
  - Issue raster position with each vertex

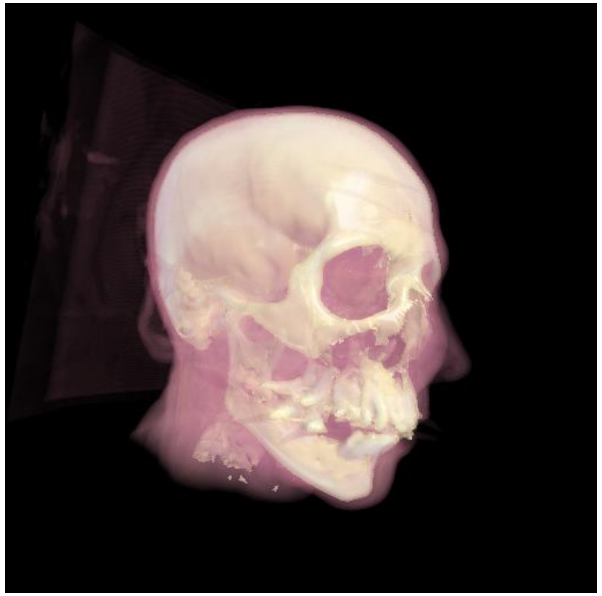


```
FOR EACH fragment
    Compute volume entry position
    Compute ray of sight direction
    WHILE in volume
            Lookup data at ray position in volume texture
            Accumulate color and opacity
            Advance along ray
```

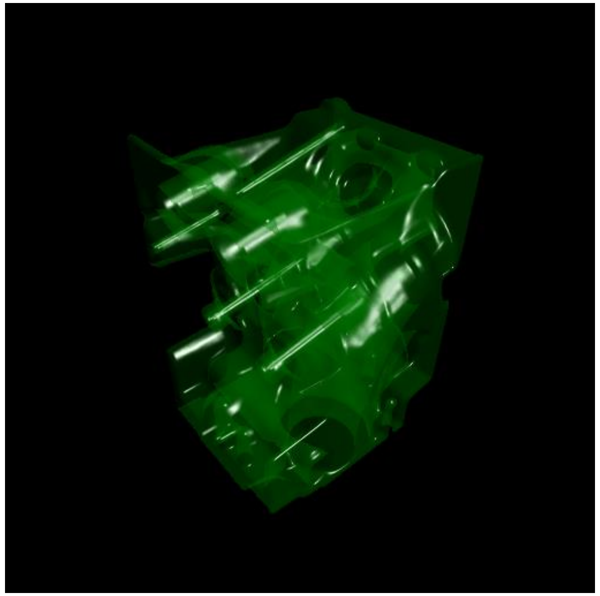


GPU Ray Casting: Examples

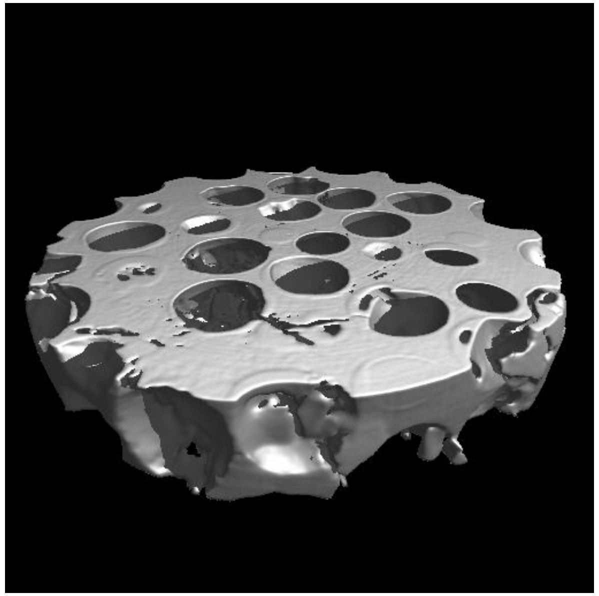
- High flexibility
  - Shading models
  - Acceleration techniques (early ray termination, empty space leaping, etc.)



Direct volume rendering



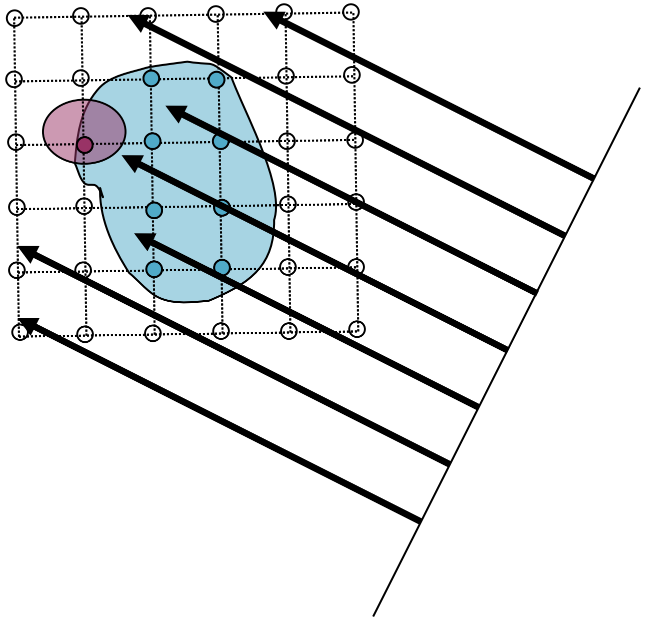
Transparent, illuminated isosurfaces



Isosurface with self-shadowing

Acceleration Techniques for Ray Casting

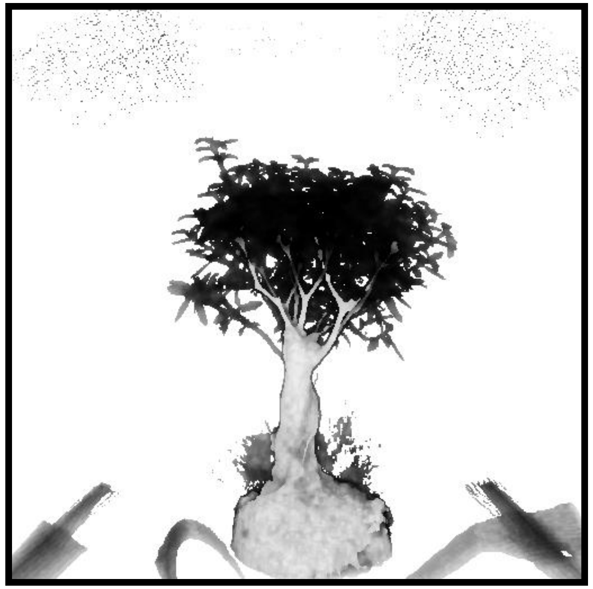
- **Problem:** ray casting/marching is time consuming
- **Idea:**
  - Neglect "irrelevant" information to accelerate the rendering process
  - Exploit coherence
- Early-ray termination
  - Colors from faraway regions do not contribute if accumulated opacity is already high
  - Stop traversal if contribution of sample becomes irrelevant
  - User-set opacity level for termination
  - Front-to-back compositing



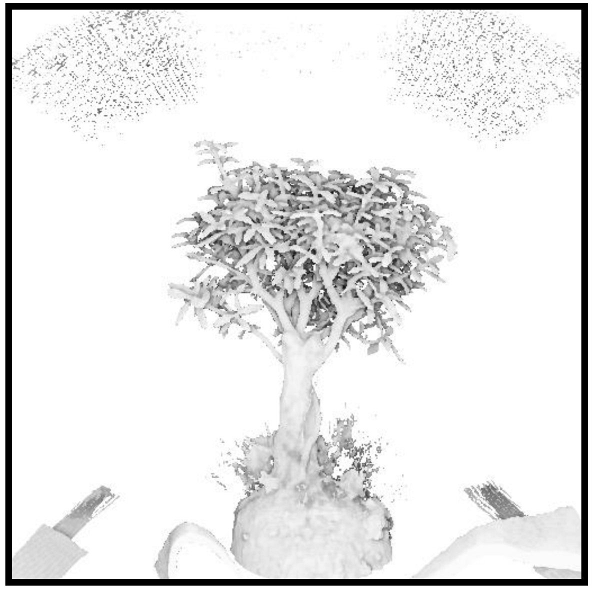
Acceleration Techniques for Ray Casting

- Effect of early-ray termination

Example image



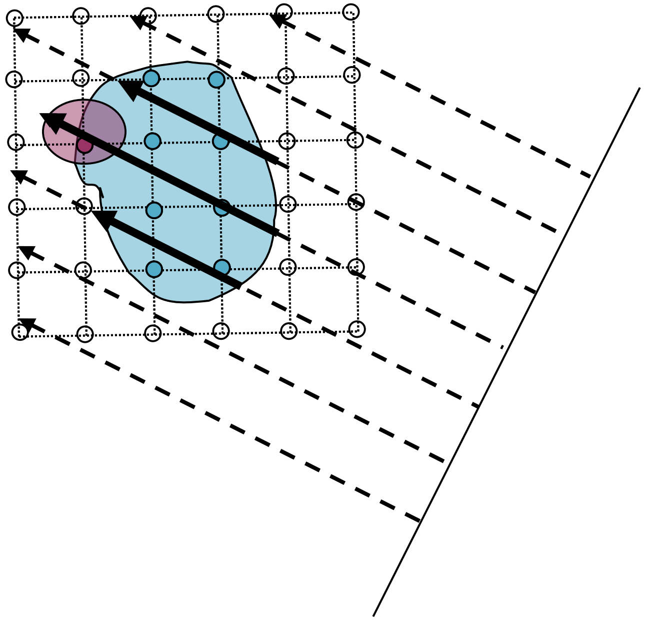
# Sample points
(semi-transparent)



# Sample points
(opaque)

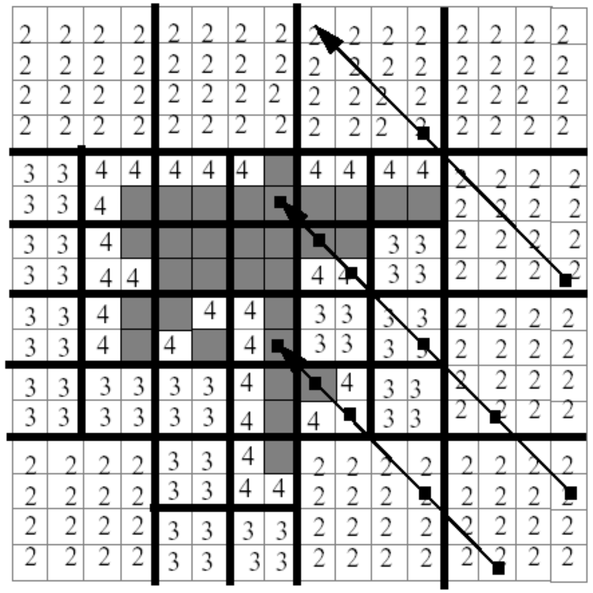
Acceleration Techniques for Ray Casting

- Space leaping
  - Skip empty cells
- Homogeneity-acceleration
  - Approximate homogeneous regions with fewer sample points



Acceleration Techniques for Ray Casting

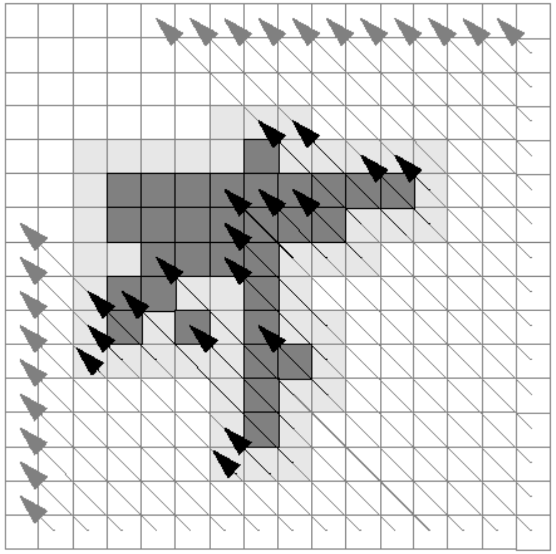
- Hierarchical spatial data structure
  - Octree
  - Mean value and variance stored in nodes of octree



(number encodes octree level)

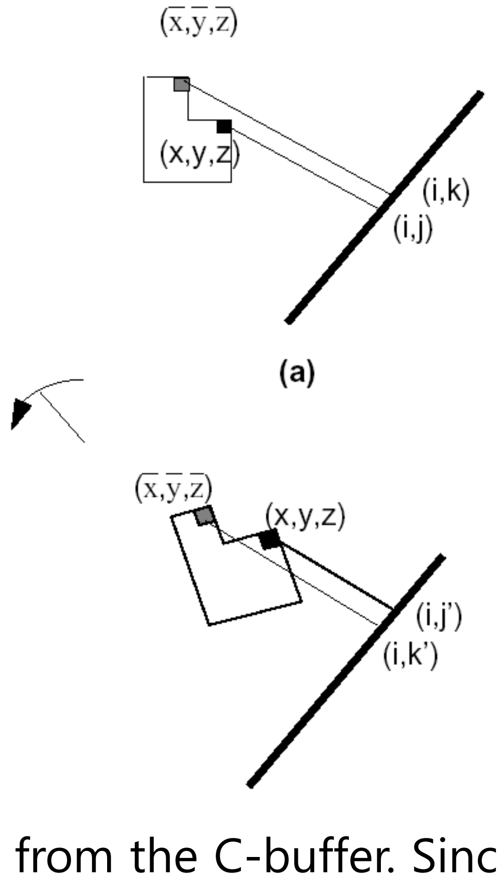
Acceleration Techniques for Ray Casting

- Adaptive ray traversal
  - Different “velocities” for traversal
  - Different distance between samples
  - Based on vicinity flag
  - Layer of “vicinity voxels” around non-transparent parts of the volume



Acceleration Techniques for Ray Casting

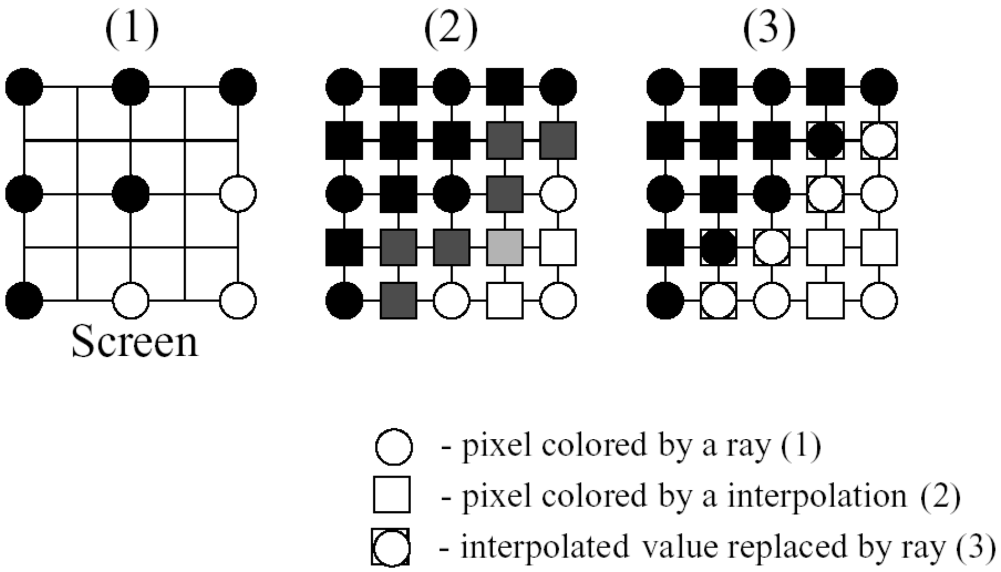
- Exploiting temporal coherence in volume animations
  - C-buffer (Coordinates buffer)
  - Store coordinates of first opaque voxel
  - Removing potentially hidden voxels
  - Or adding potentially visible voxels
  - Criterion: change of position on image plane


($\overline{x},\overline{y},\overline{z}$)
(x,y,z)
(i,k)
(i,j)
(a)
($\overline{x},\overline{y},\overline{z}$)
(x,y,z)
(i,j')
(i,k')
(b)

Removing potentially hidden coordinates from the C-buffer. Since the relationship between the two voxels in (a) changed, it serves as an indicator that the other voxel is potentially hidden (b).

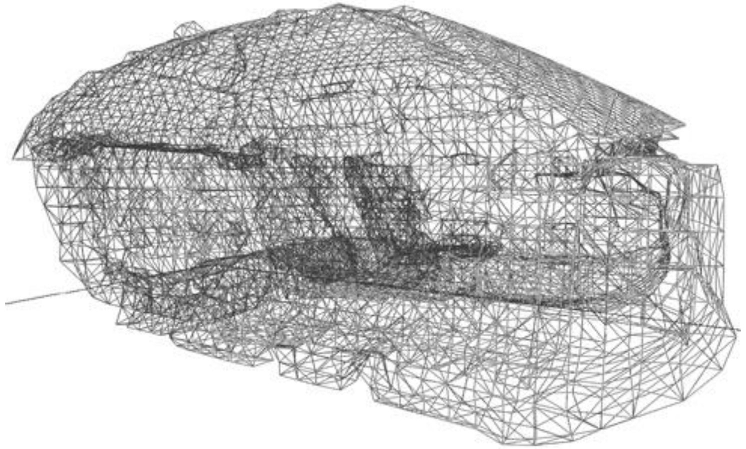
Acceleration Techniques for Ray Casting

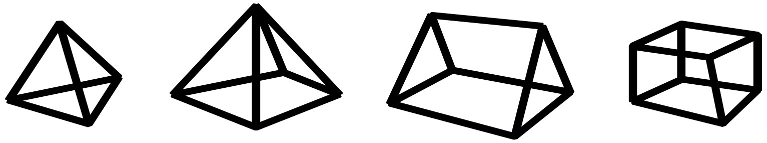
- Adaptive screen sampling [Levoy 1990]
  - Rays are emitted from a subset of pixels (on image plane)
  - Missing values are interpolated
  - In areas of high value gradient additional rays are traced


(1)
(2)
(3)
Screen
○ - pixel colored by a ray (1)
□ - pixel colored by a interpolation (2)
- interpolated value replaced by ray (3)

Ray Casting

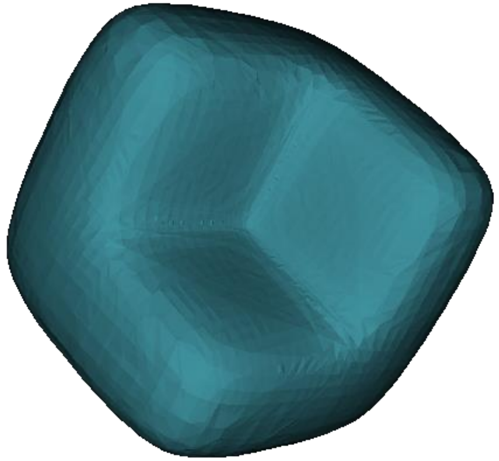
- Ray casting in tetrahedral grids
  - Linear interpolation within cells
  - Slightly modify the traversal through the grid, compared to uniform grids
  - Algorithm by M. P. Garrity
    ["Raytracing irregular volume data", VolVis, 1990]

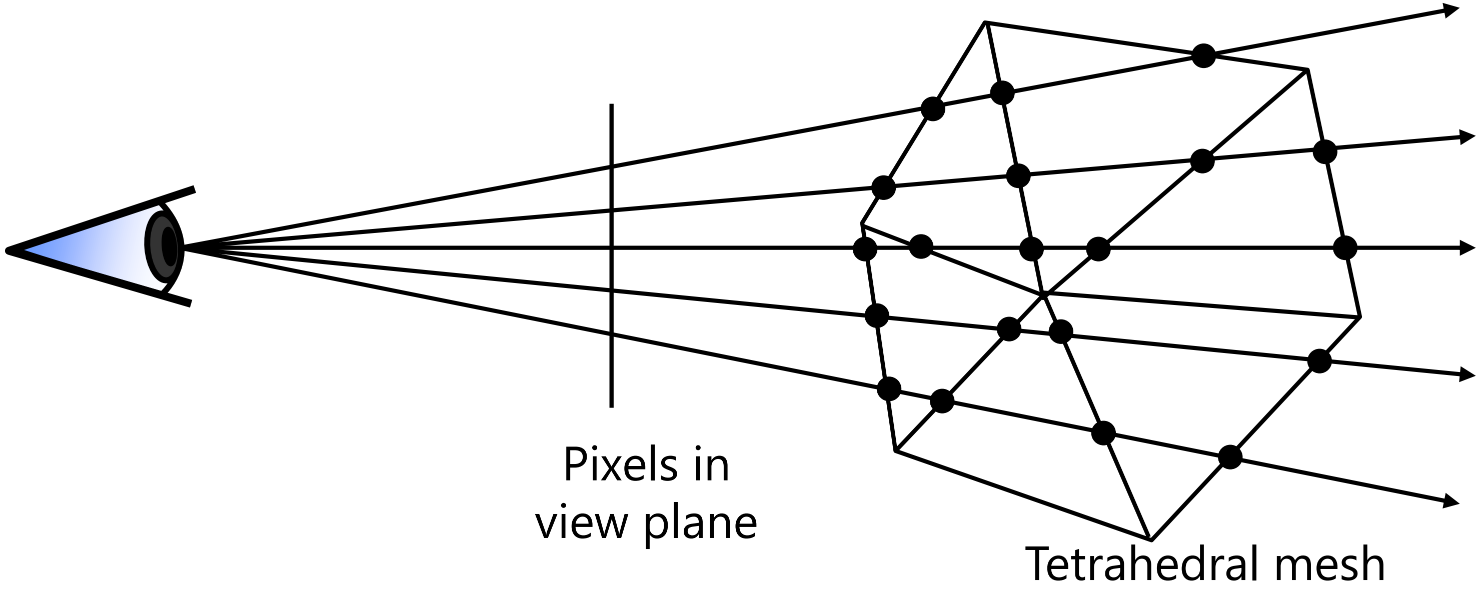




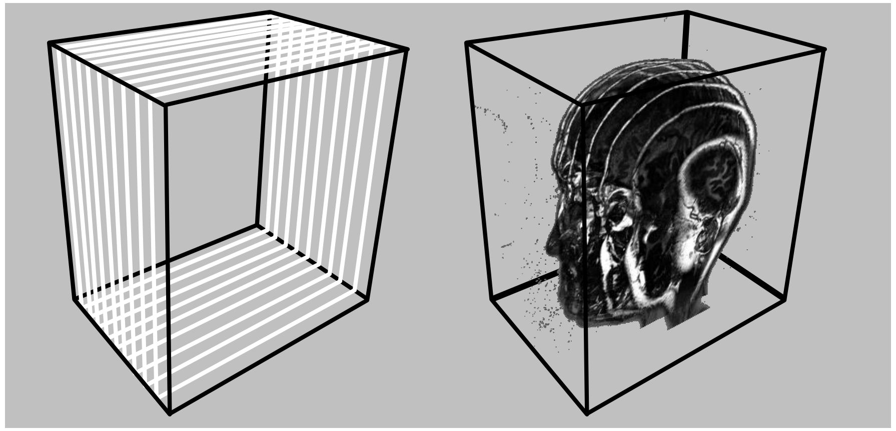
Ray Casting

- Ray casting in tetrahedral grids
  - Traverse rays front-to-back
  - Stop at intersected cell faces
  - Compute color and opacity for current ray segment
  - Accumulate volume colors and opacities

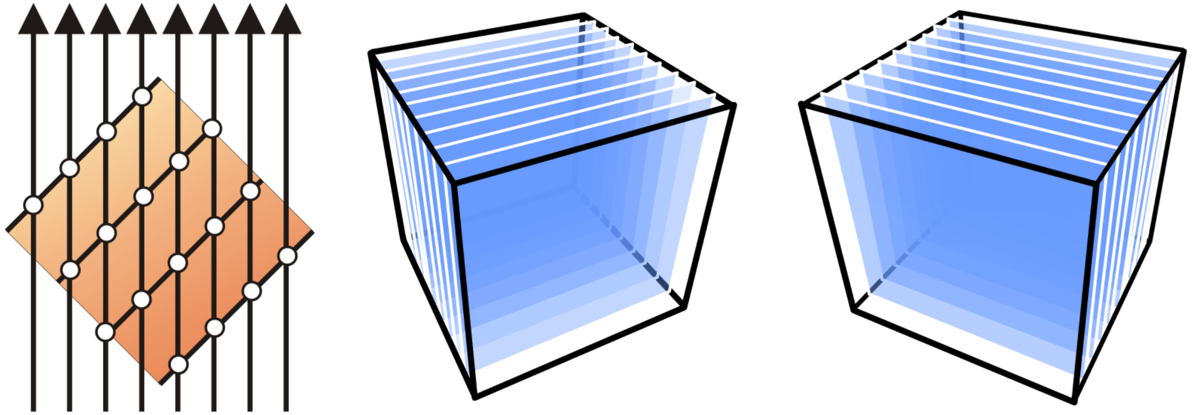




Texture-Based Volume Rendering

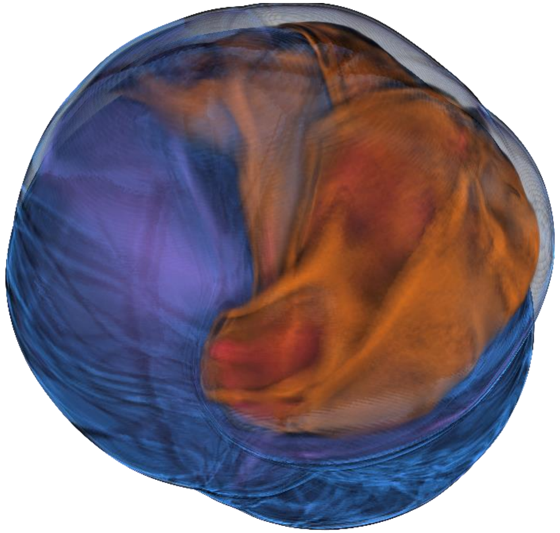


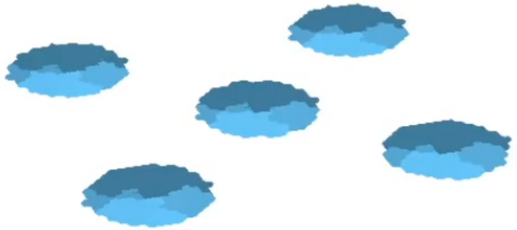
- Object-space approach
- Based on graphics hardware:
  - Rasterization, Texturing, Blending
- Proxy geometry → there are no volumetric primitives in graphics hardware
- **"Historic" Example:** 2D textured slices through the volume
  - Object-aligned slices
  - Three stacks of 2D textures
  - Bilinear interpolation
  - Back-to-front traversal (blending)



Outlook: Time-dependent Volume Data

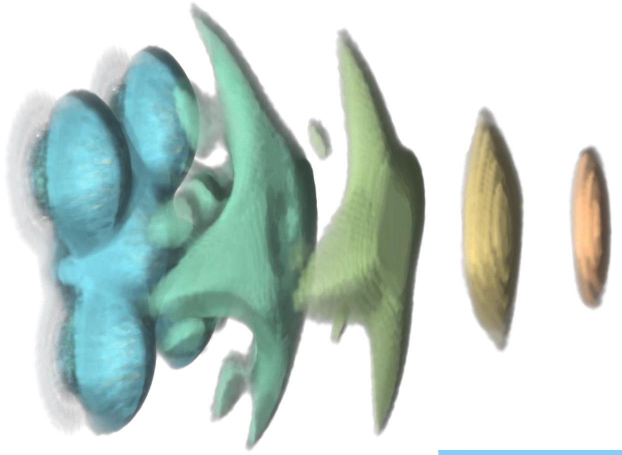
- Videos/Animation ineffective for visual analysis
- Compositing of all time steps: occlusion and visual clutter
- **Idea:** find a meaningful static representation

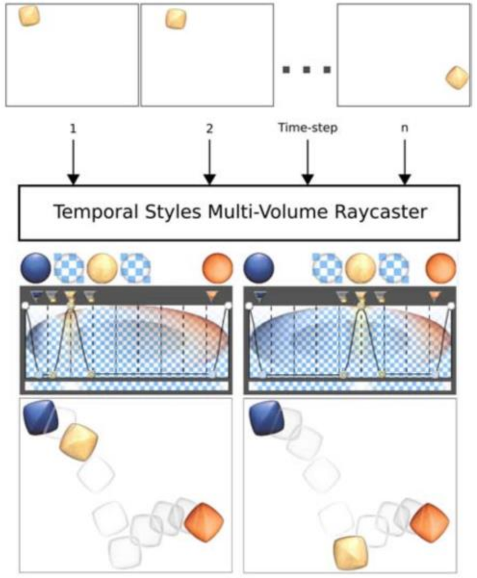


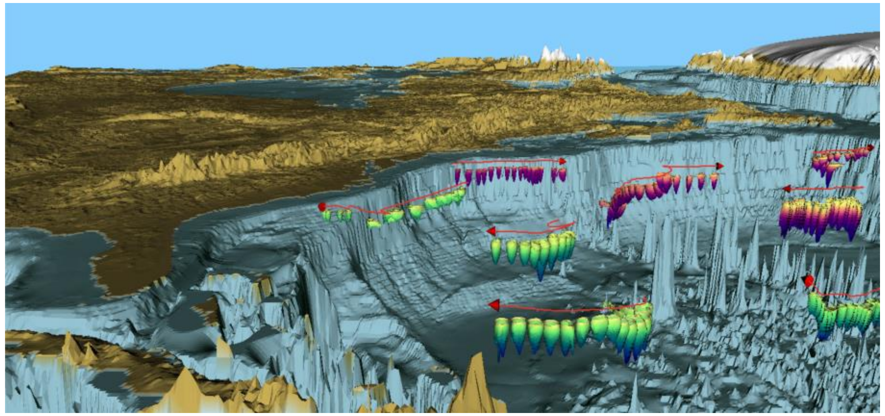


Outlook: Time-Dependent Volume Visualization

- Treat data as space-time hypercube
  - Slicing and projection
  - Dedicated transfer functions
- Time step selection
  - Based on selection metrics
- Feature extraction and visualization
  - e.g., Illustration-inspired techniques

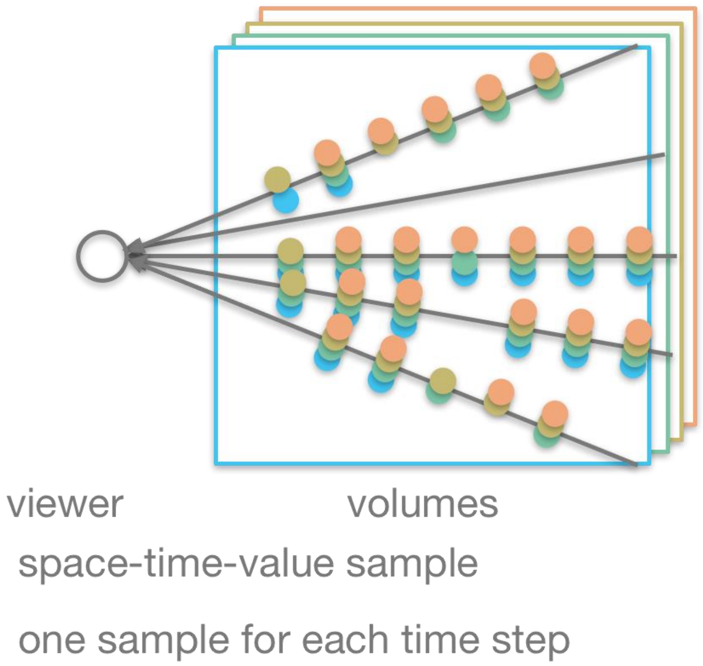


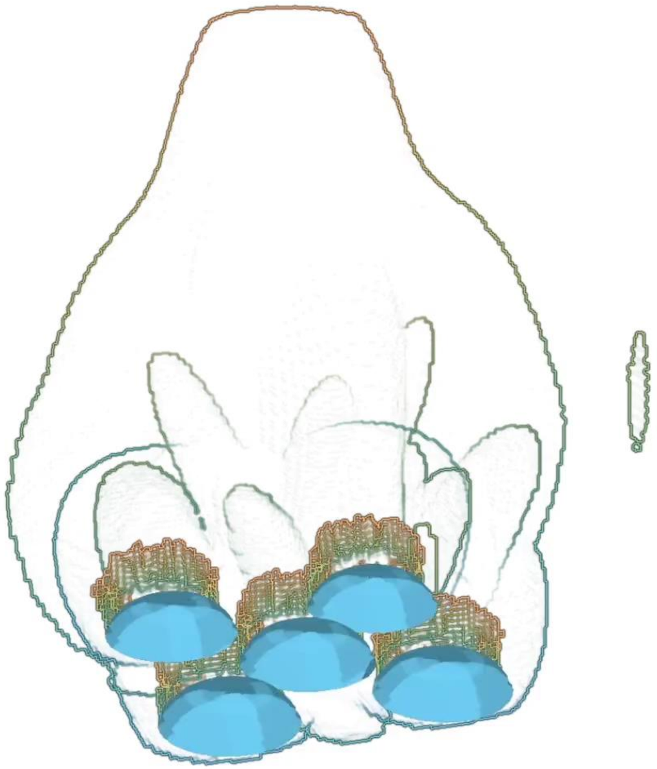




Spatio-Temporal Contours

- Idea: compute differences between sets of samples computed along each ray in space and time [S. Frey, EuroVis 2018]
  - Visualize large differences between sample sets as contours





Outlook: Volume Ensembles

- Multifield data (ensembles, time-dependent, etc.) a focus of research
- Often based on feature extraction
  - Higher dimensional data, clustering, graphs, etc.
  - Connection to „Information Visualization“