Computer Graphics

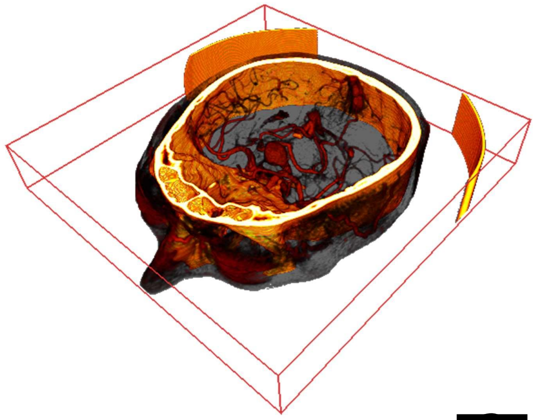
Volume Rendering

Tobias Isenberg

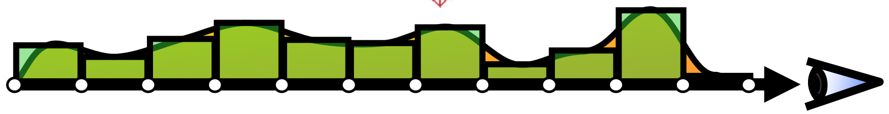
Inria informatics mathematics

Overview

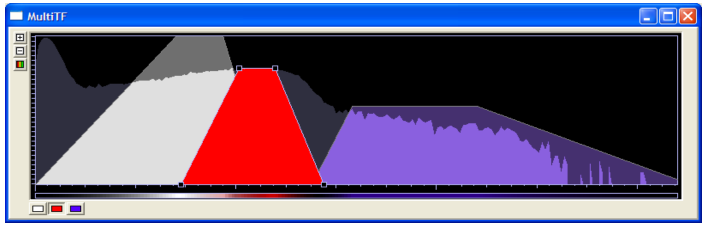
- motivation/introduction



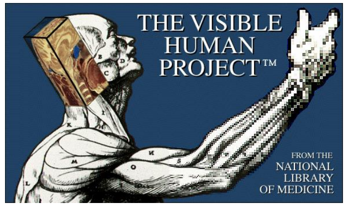
- radiative energy transfer



- volume rendering pipeline



- applications



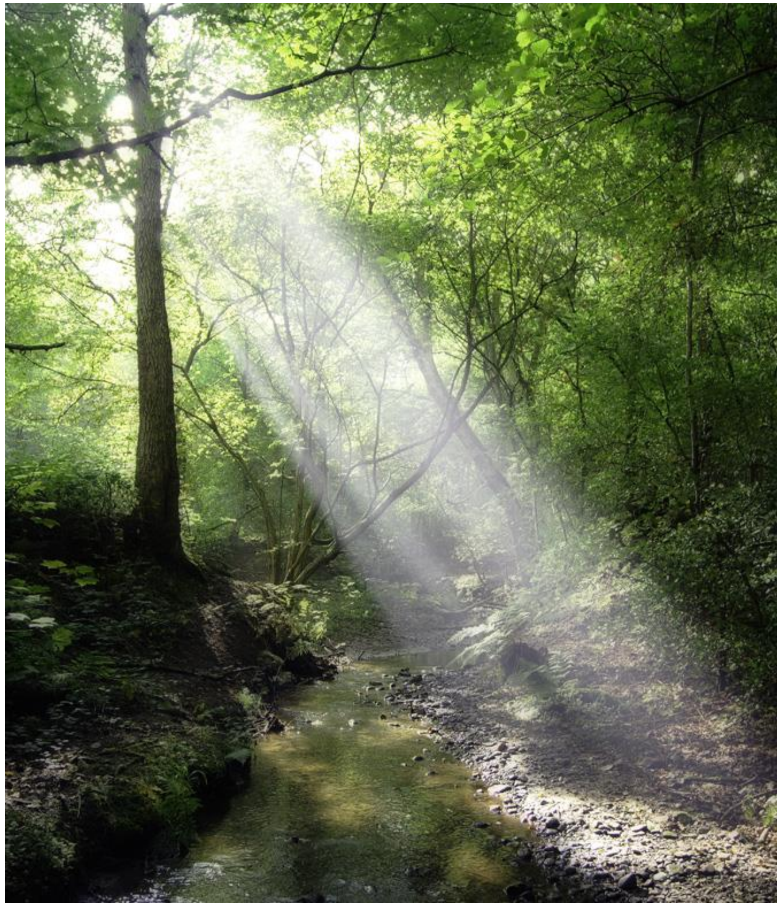
Motivation

- volumetric rendering effects

Images by H. W. Jensen

Motivation

- volumetric rendering effects

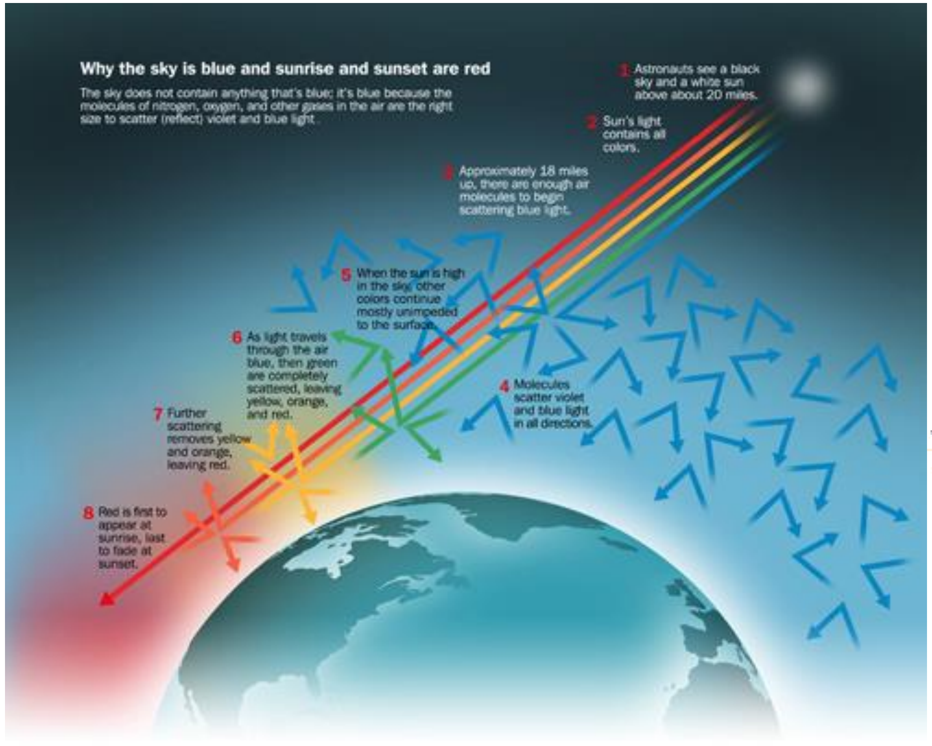


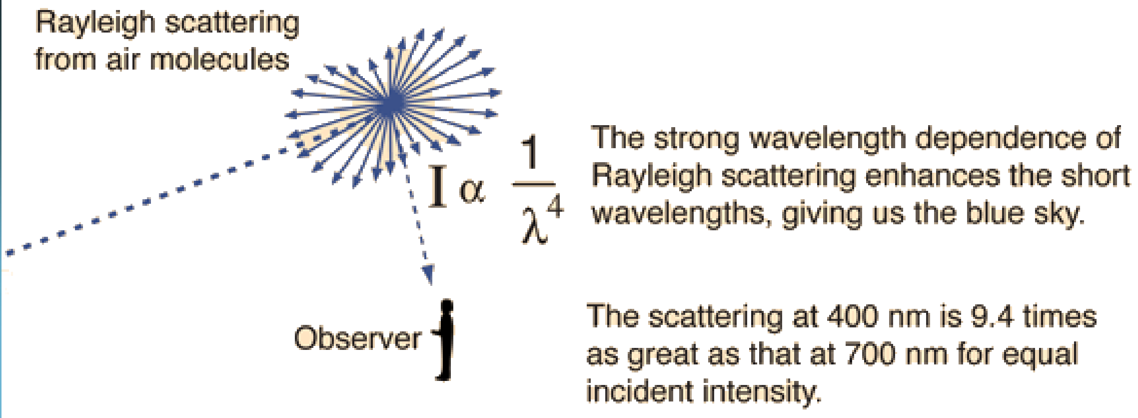
Motivation

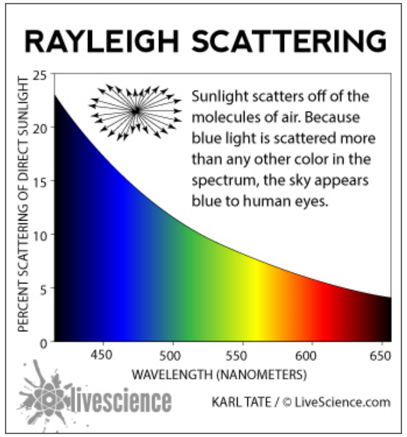
- volumetric rendering effects

  Q: Why is the sky blue?

  A: Rayleigh scattering.

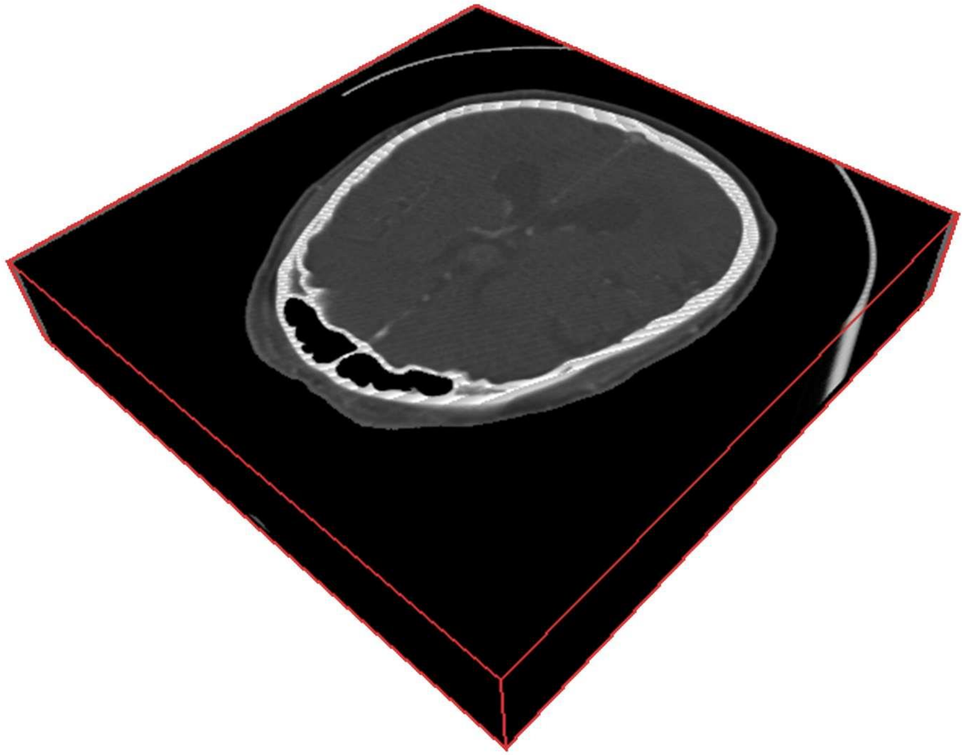


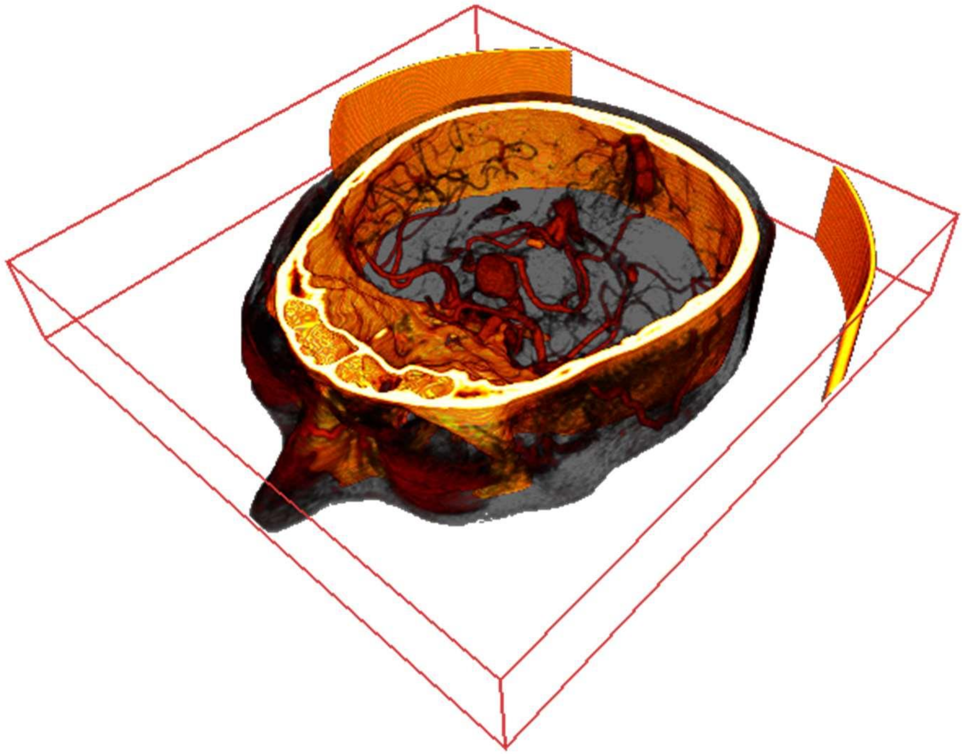




Motivation

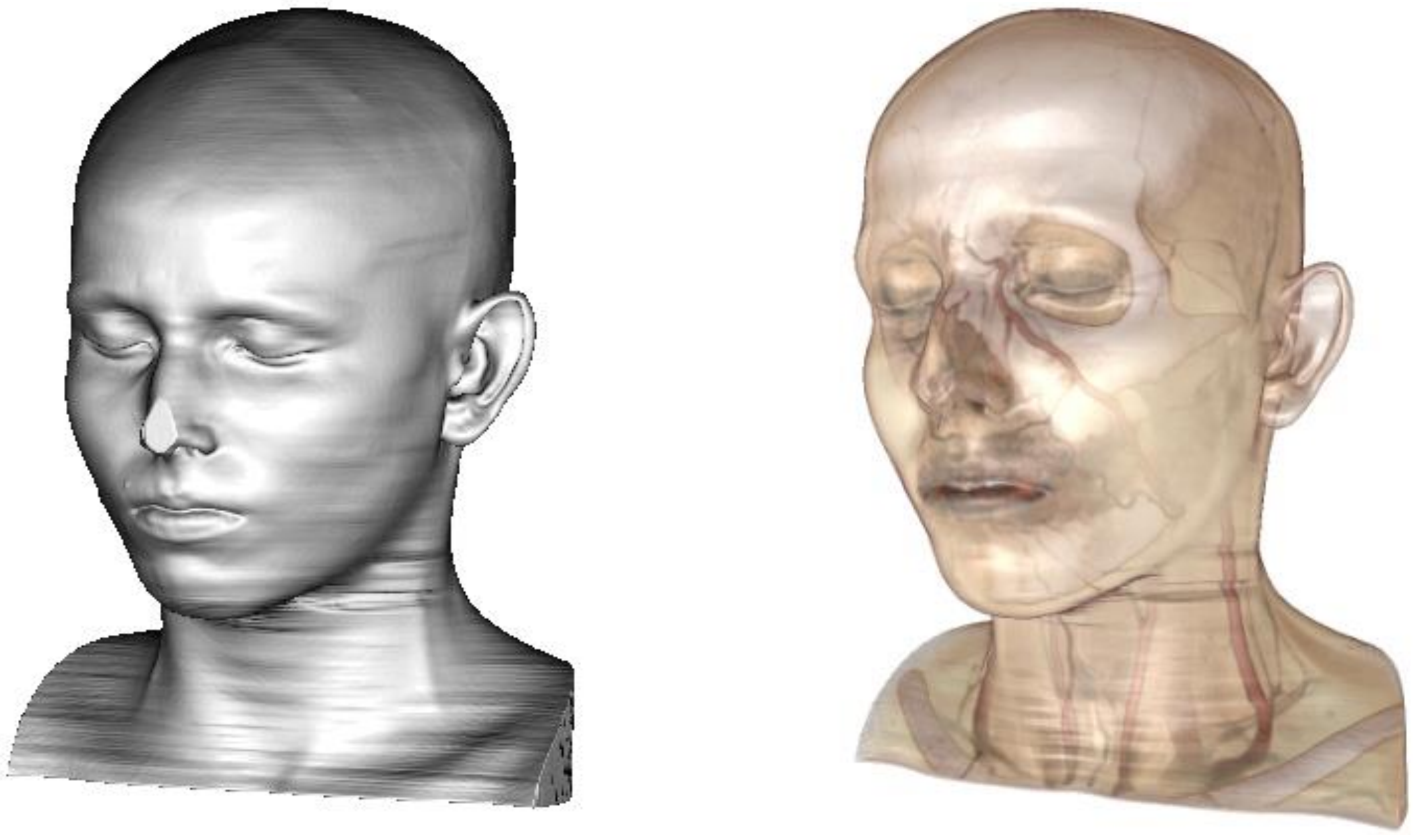
- volumetric datasets





Motivation

- iso-surface extraction + surface rendering not enough

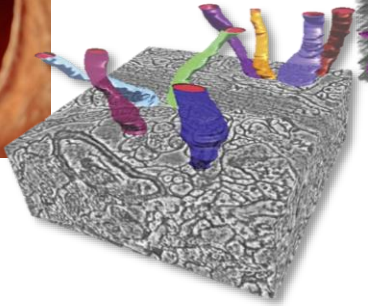
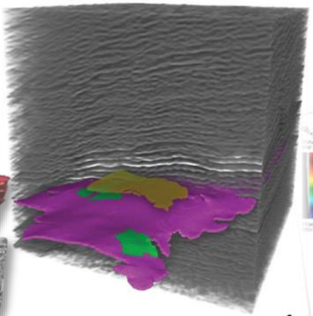
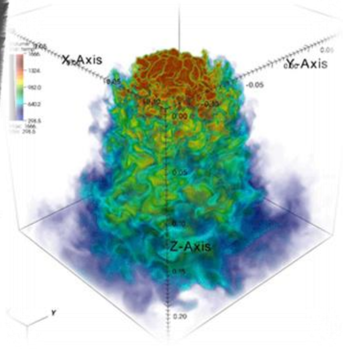


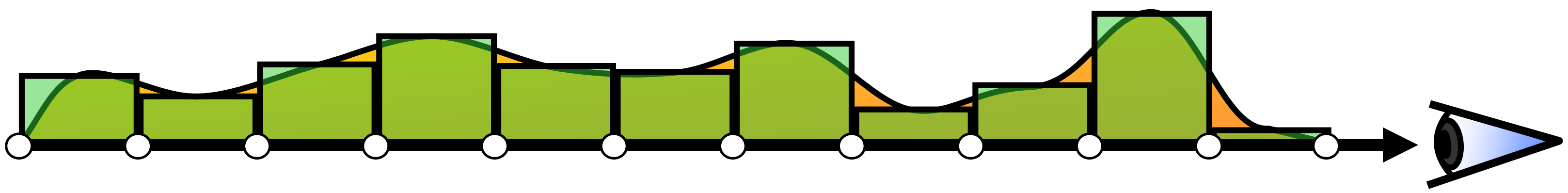
Motivation/applications

- realistic rendering of (e.g.):
  - atmospheric effects
  - fluid effects
  - translucent solids
  - biological tissues




- “scientific” visualization of (e.g.):
  - medical data
  - biological data
  - geological data
  - engineering data

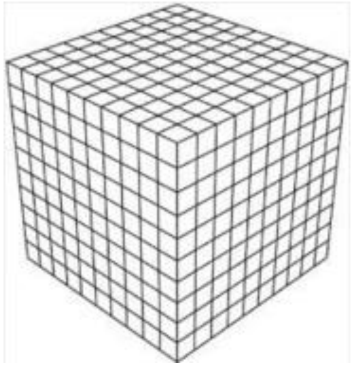





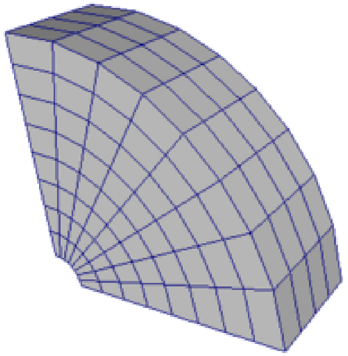


(Approximation of the) Radiative Energy Transfer

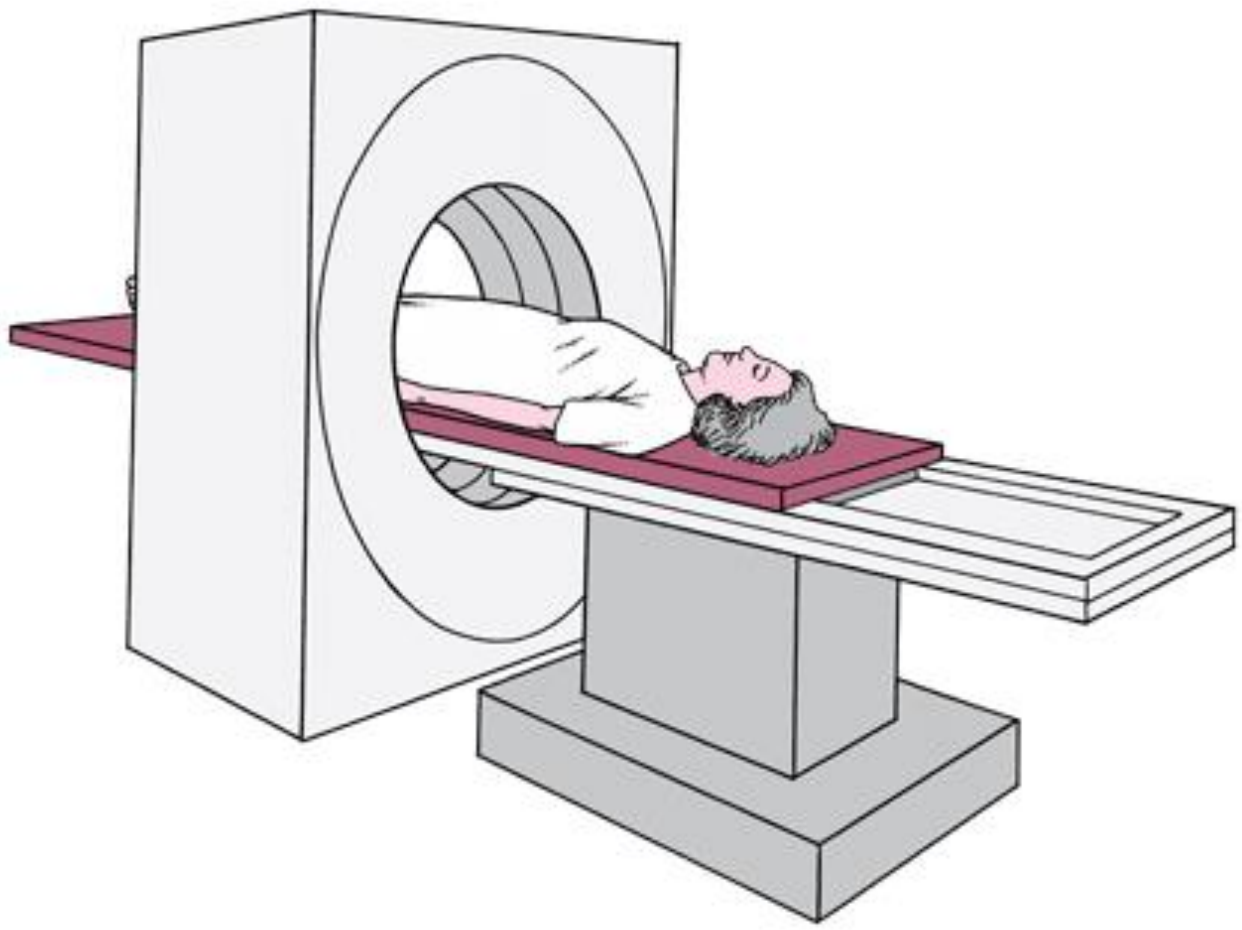
Volume rendering

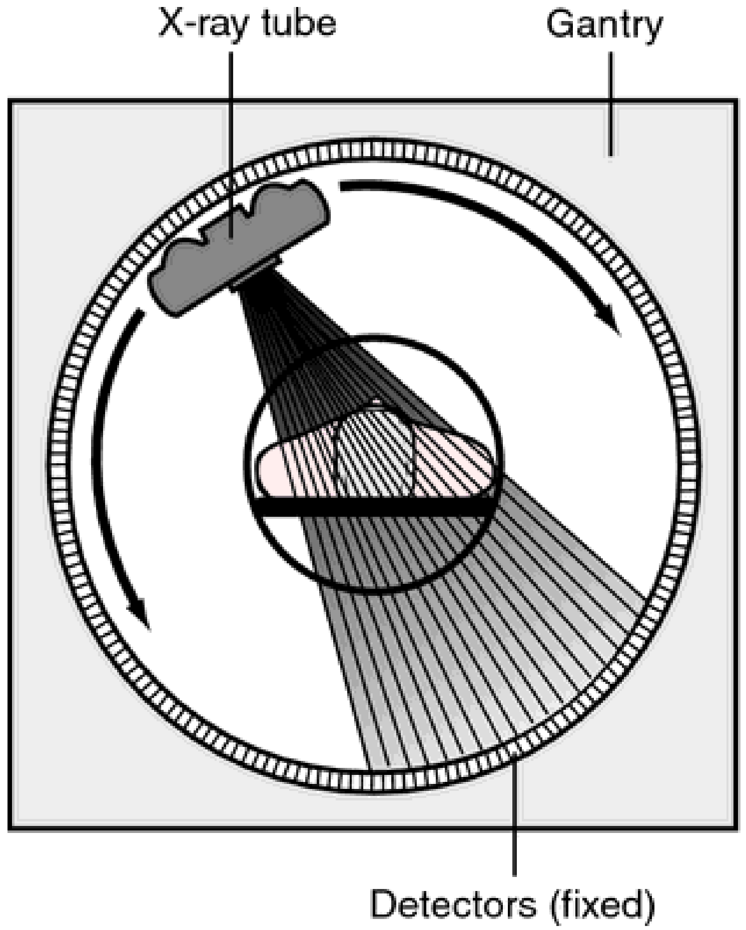
- rendering of a 3D (scalar) field
  - no explicitly defined surfaces
  - 3D data is projected onto a 2D image
  - 2D pixel → 3D voxel
- most commonly based on sampled representation
  - regular grids
  - curvilinear grids
  - point clouds



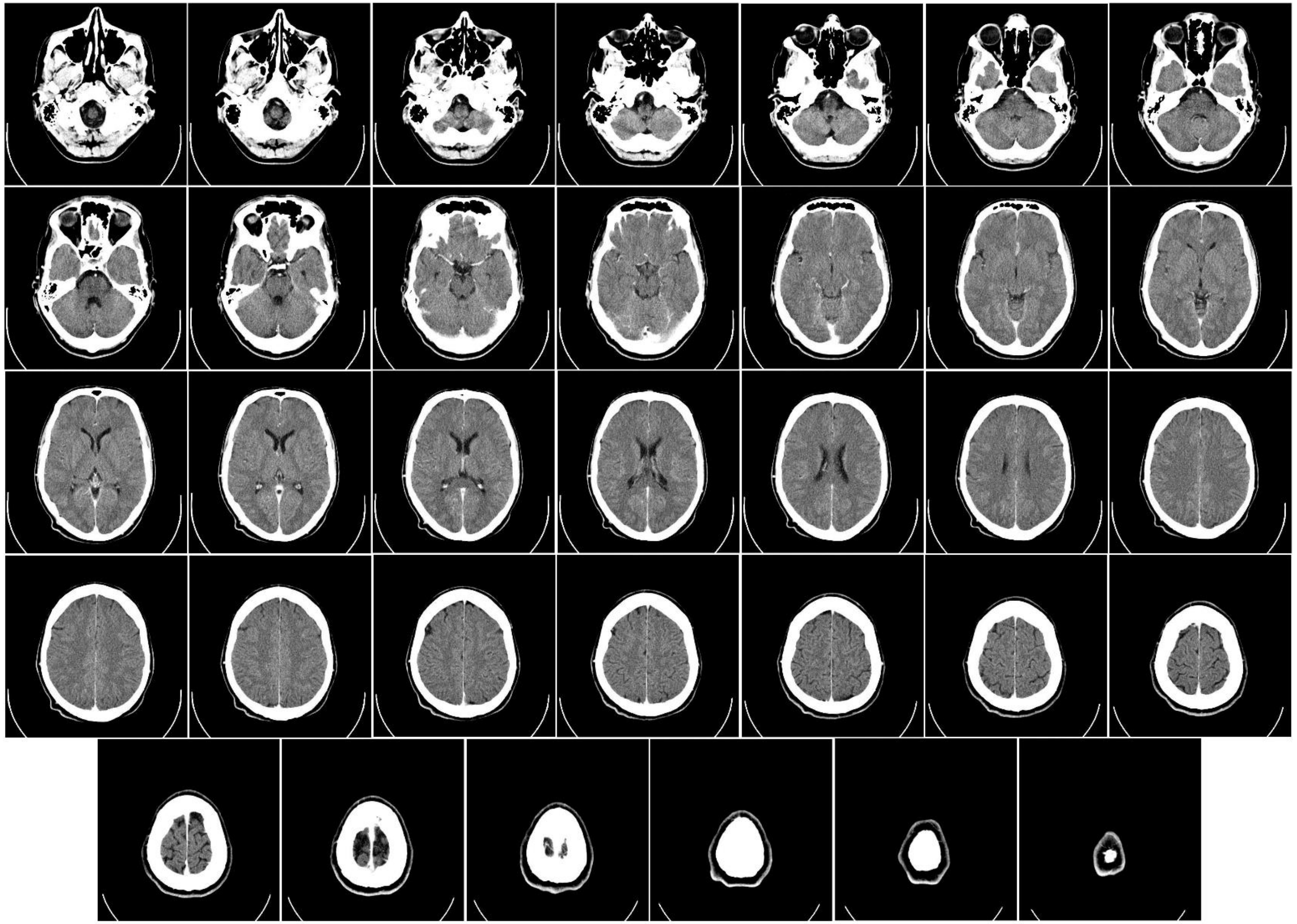


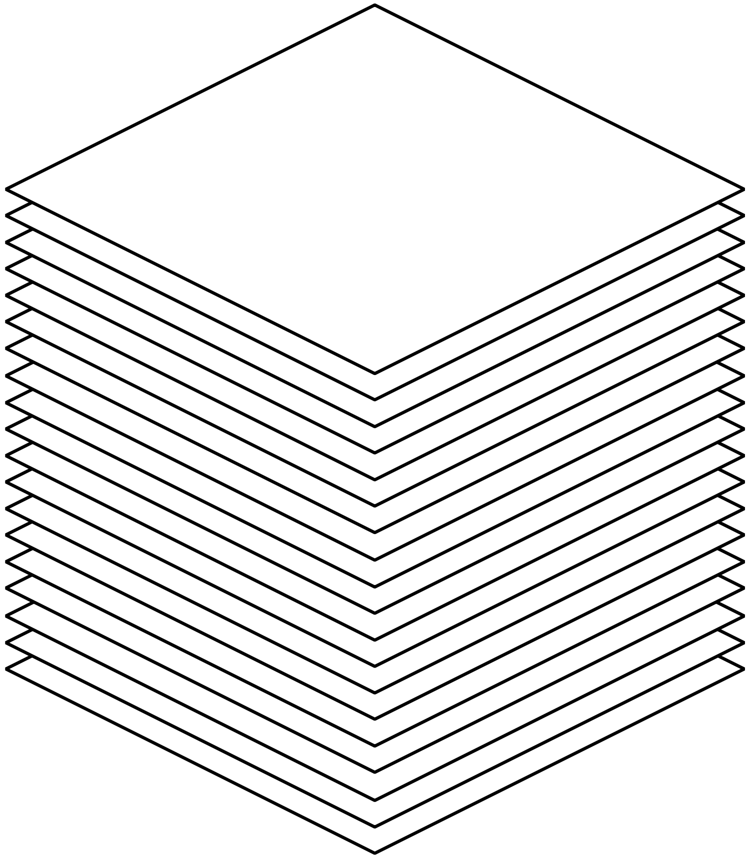
Example: Computed Tomography (CT)

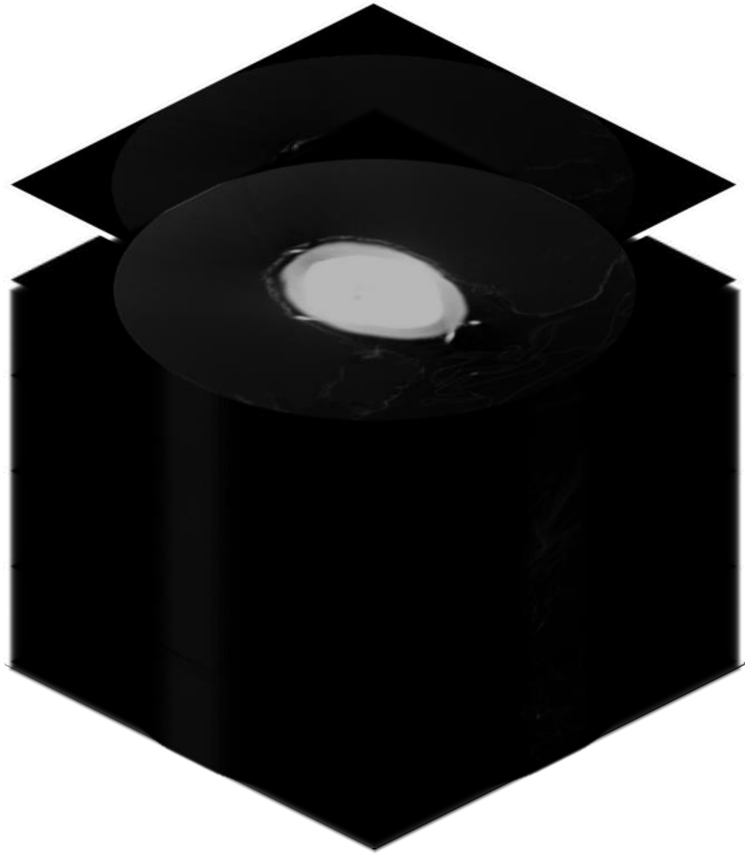


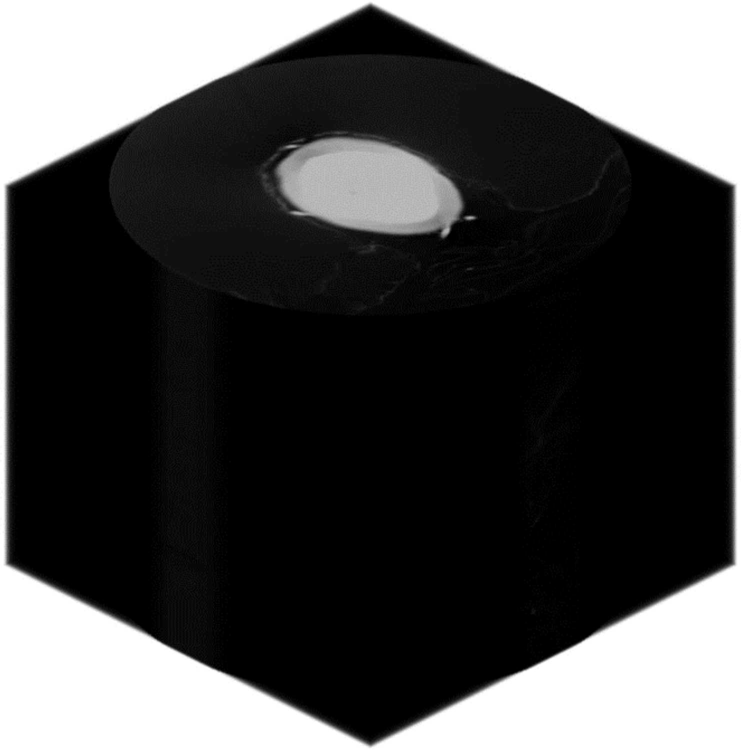


Example: Computed Tomography (CT)

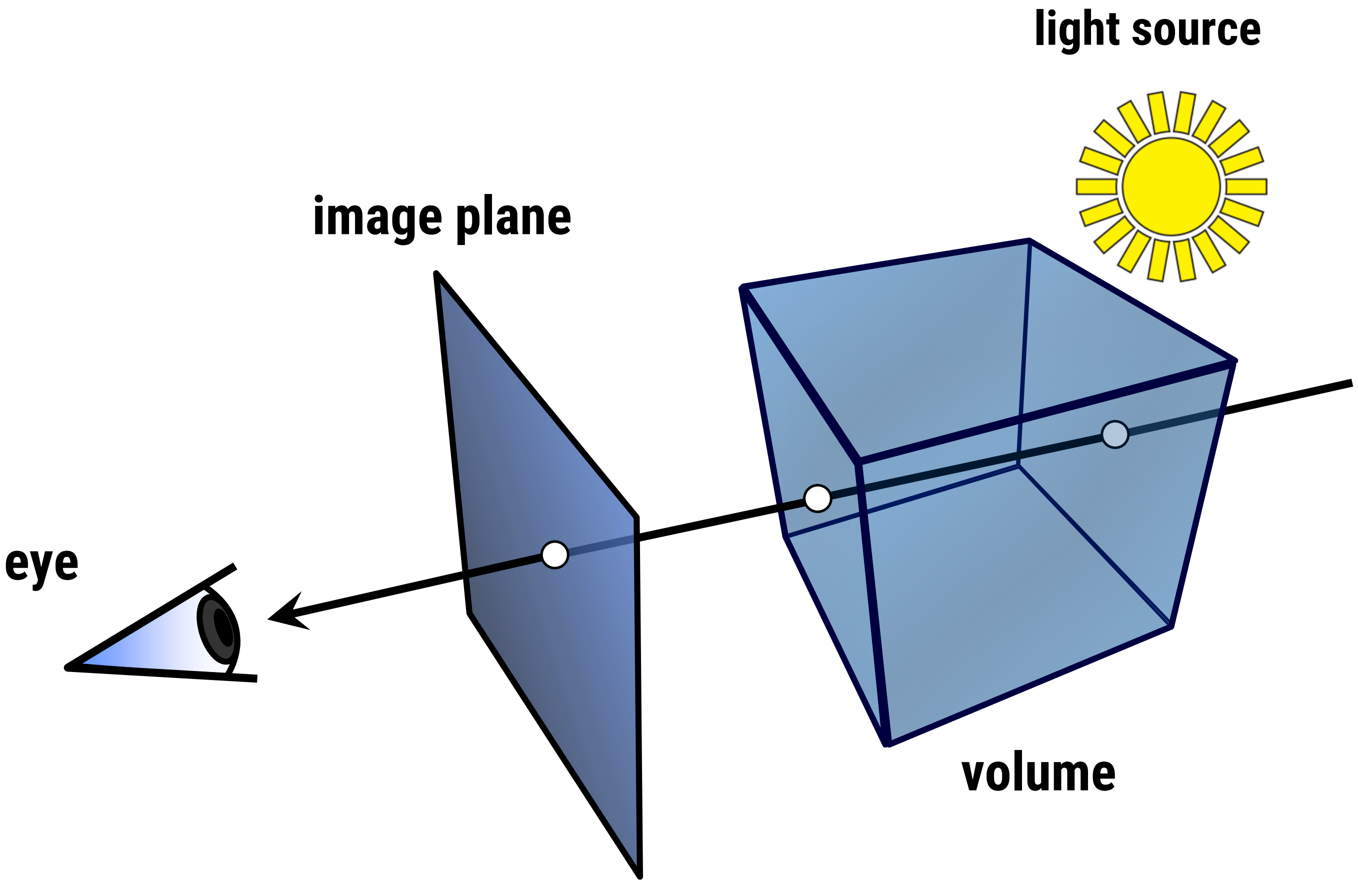








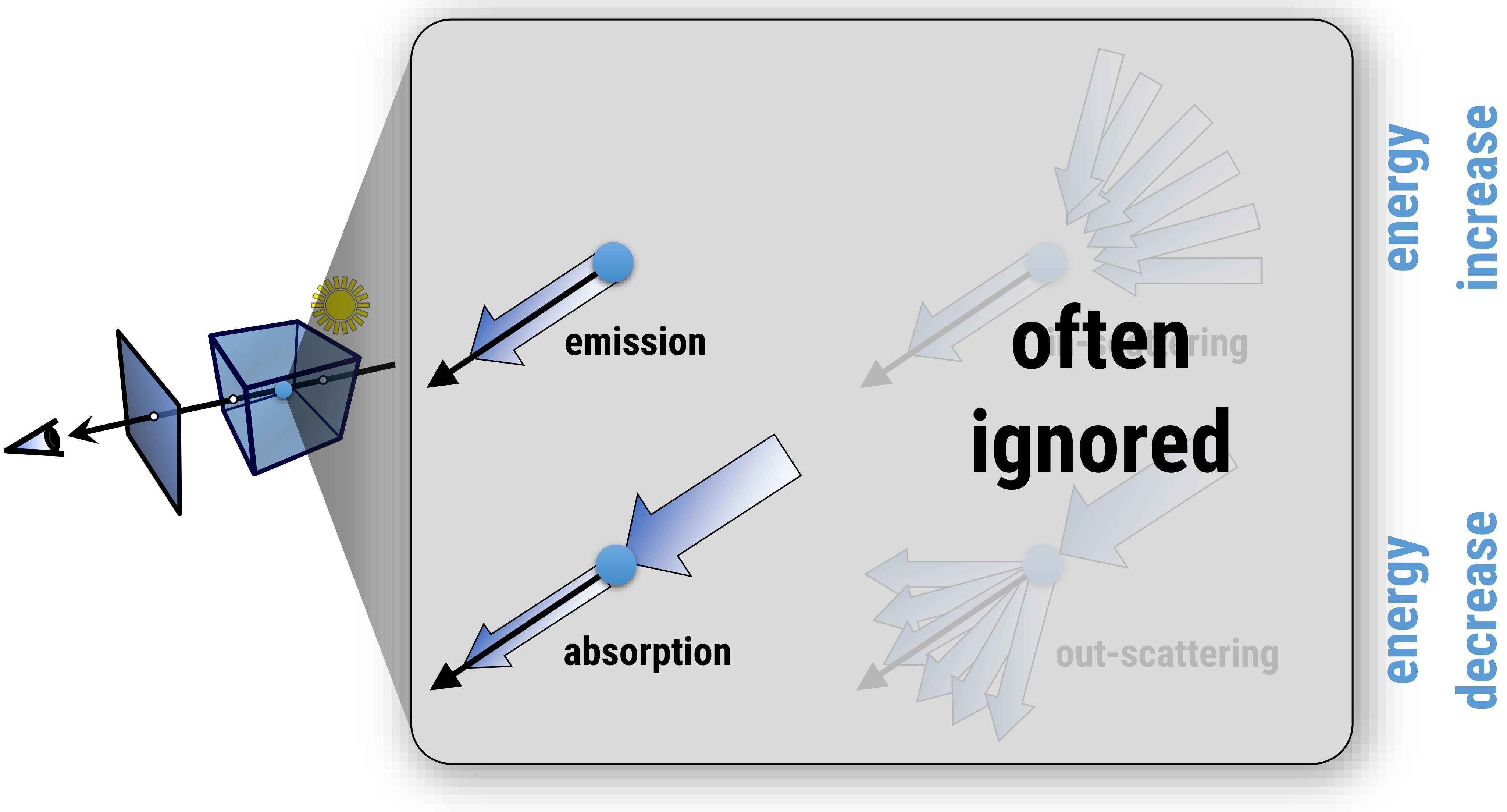
Radiative transfer in volumes



Radiative transfer in volumes

- light potentially interacts with the medium at every point
- for each pixel on the image plane, we want to compute how much light reaches the eye
- we need to solve the **volume rendering integral**
  - analytically only possible in very simple cases (e.g., homogeneous media)
  - usually, when dealing with inhomogeneous volumes, we use an numerical approximation
  - even then, we often use various simplifications to enable a real-time solution

Radiative transfer in volumes



Analytical model

$$I(s) = I(s_0)$$

without absorption all the initial radiant energy would reach the point s

Analytical model

absorption between s_0 and s

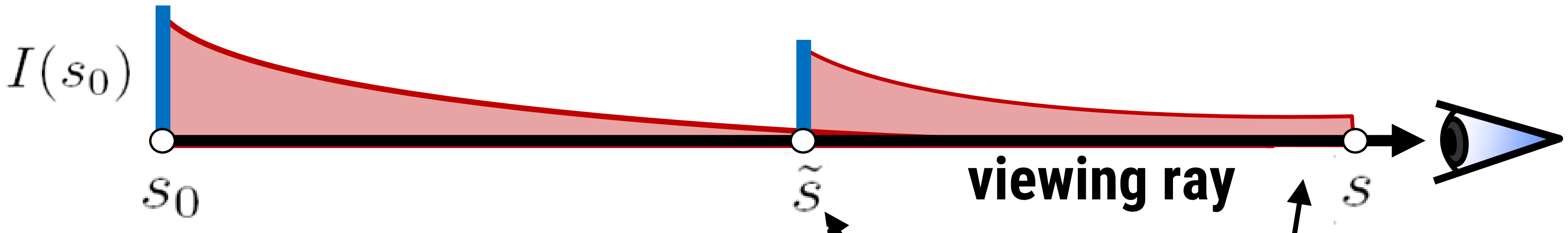
$$I(s) = I(s_0)\, e^{-\tau(s_0, s)}$$

Extinction τ

Absorption κ

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

Analytical model



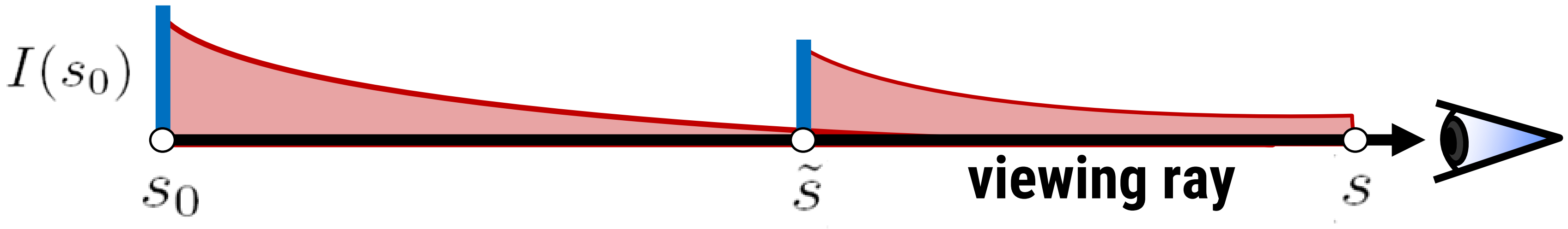
one point $\tilde{s}$ along the viewing ray emits additional radiant energy

active emission at point $\tilde{s}$

absorption between $\tilde{s}$ and s

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

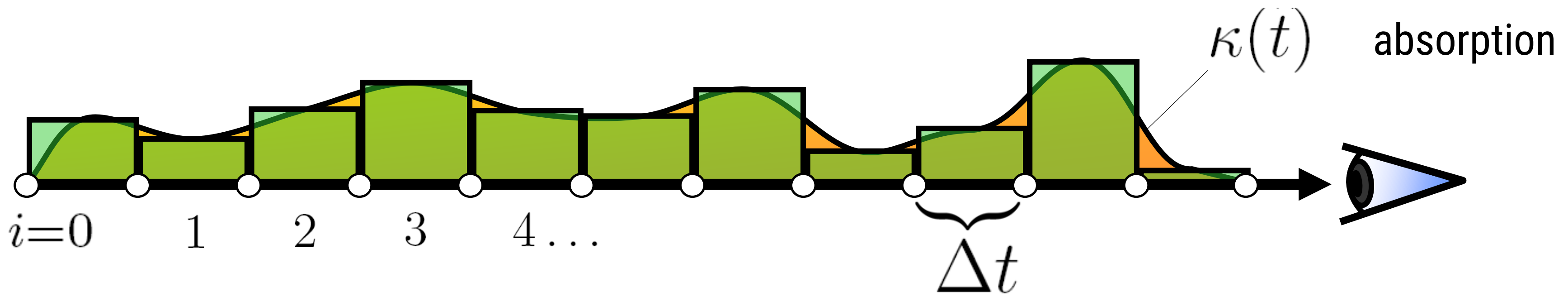
Analytical model



every point $\tilde{s}$ along the viewing ray emits additional radiant energy

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

Numerical approximation

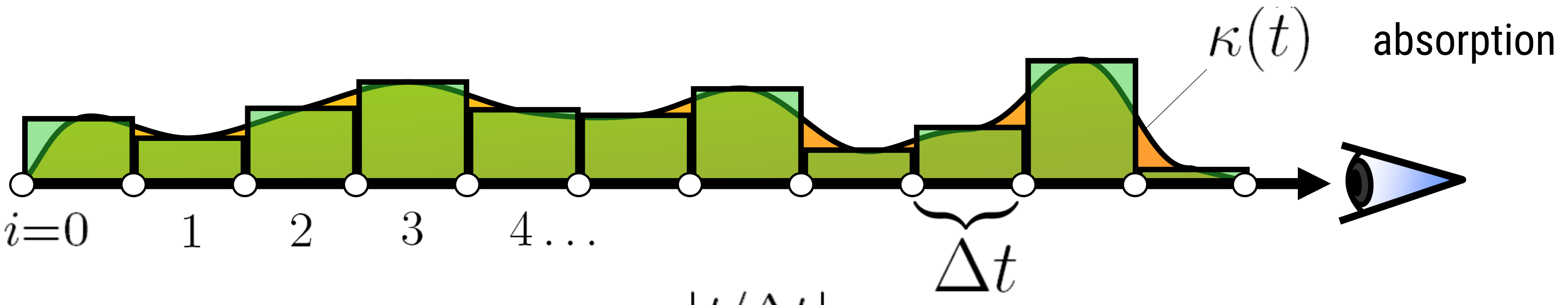


Extinction: $$\tau(0,t) = \int_0^t \kappa(\hat{t})\, d\hat{t}$$

approximate integral by Riemann sum:

$$\tau(0,t) \approx \sum_{i=0}^{\lfloor t/\Delta t \rfloor} \kappa(i \cdot \Delta t)\, \Delta t$$

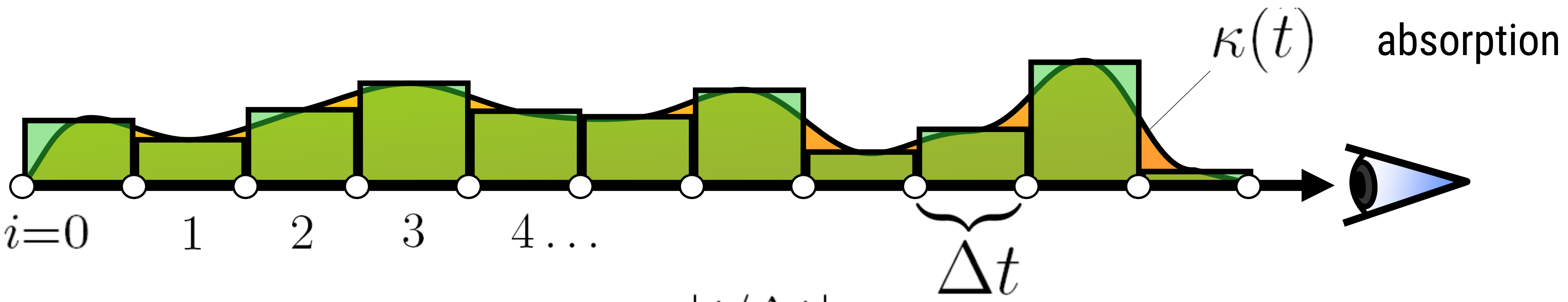
Numerical approximation



$$\tau(0,t) \approx \tilde{\tau}(0,t) = \sum_{i=0}^{\lfloor t/\Delta t \rfloor} \kappa(i \cdot \Delta t)\, \Delta t$$

$$e^{-\tilde{\tau}(0,t)} = e^{-\sum_{i=0}^{\lfloor t/\Delta t \rfloor} \kappa(i \cdot \Delta t)\, \Delta t}$$

Numerical approximation



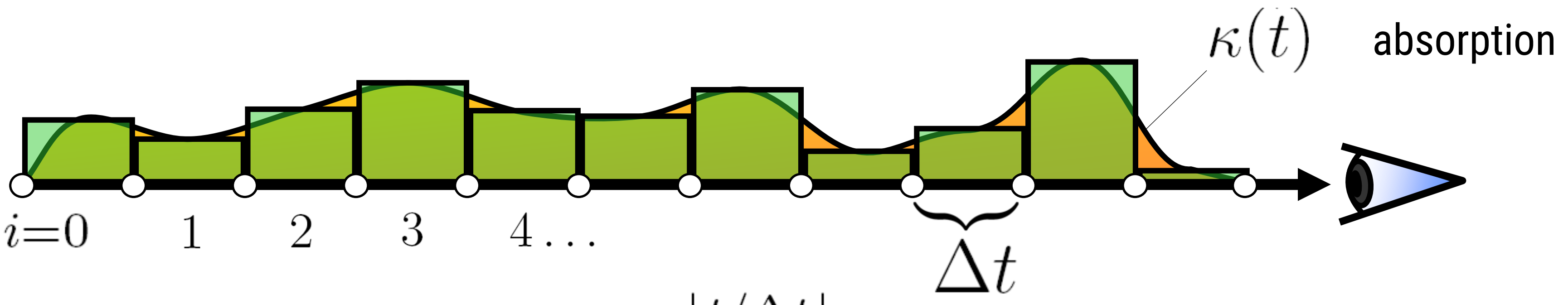
$$\tau(0,t) \approx \tilde{\tau}(0,t) = \sum_{i=0}^{\lfloor t/\Delta t \rfloor} \kappa(i \cdot \Delta t)\, \Delta t$$

$$e^{-\tilde{\tau}(0,t)} = \prod_{i=0}^{\lfloor t/\Delta t \rfloor} e^{-\,\kappa(i \cdot \Delta t)\, \Delta t}$$

now we introduce opacity: (light absoption at one sample step)

$$A_i = 1 - e^{-\kappa(i \cdot \Delta t)\, \Delta t}$$

Numerical approximation



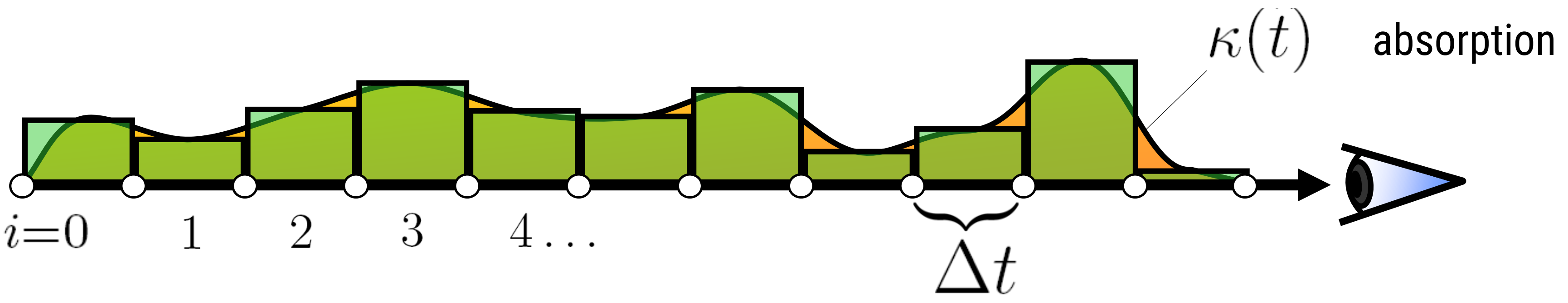
$$\tau(0,t) \approx \tilde{\tau}(0,t) = \sum_{i=0}^{\lfloor t/\Delta t \rfloor} \kappa(i \cdot \Delta t)\, \Delta t$$

$$e^{-\tilde{\tau}(0,t)} = \prod_{i=0}^{\lfloor t/\Delta t \rfloor} \boxed{e^{-\kappa(i \cdot \Delta t)\, \Delta t}}$$

now we introduce opacity: (light absoption at one sample step)

$$(1 - A_i) = \boxed{e^{-\kappa(i \cdot \Delta t)\, \Delta t}}$$

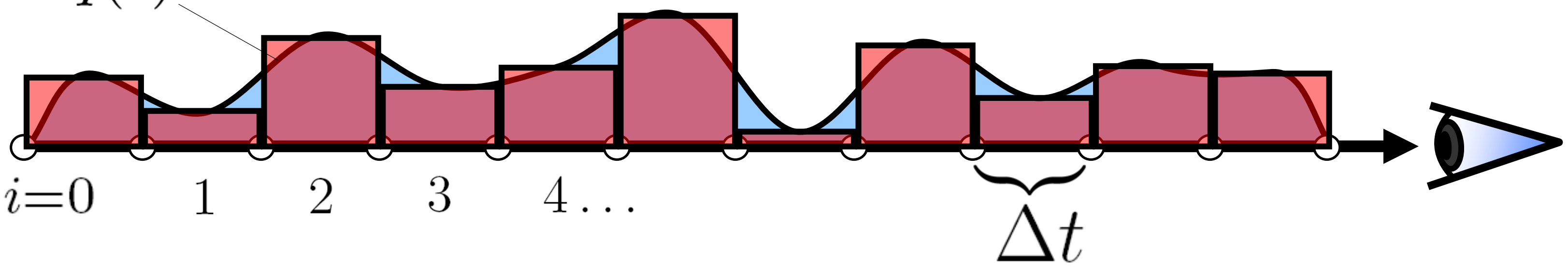
Numerical approximation



$$e^{-\tilde{\tau}(0,t)} = \prod_{i=0}^{\lfloor t/\Delta t \rfloor} (1 - A_i)$$

Numerical approximation

$q(t)$ (emitted light energy for a given point)

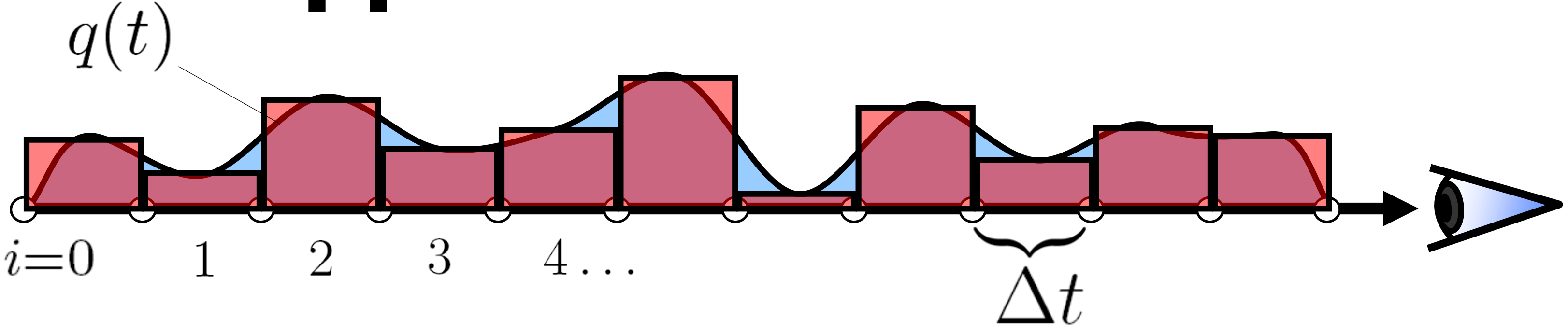


$$e^{-\tilde{\tau}(0,t)} = \prod_{i=0}^{\lfloor t/\Delta t \rfloor} (1 - A_i)$$

$$q(t) \approx C_i = c(i \cdot \Delta t)\,\Delta t$$ (emitted light energy at a given interval)

$$\tilde{C} = \sum_{i=0}^{\lfloor T/\Delta t \rfloor} C_i \, e^{-\tilde{\tau}(i,t)}$$ (light arriving at the viewer)

Numerical approximation

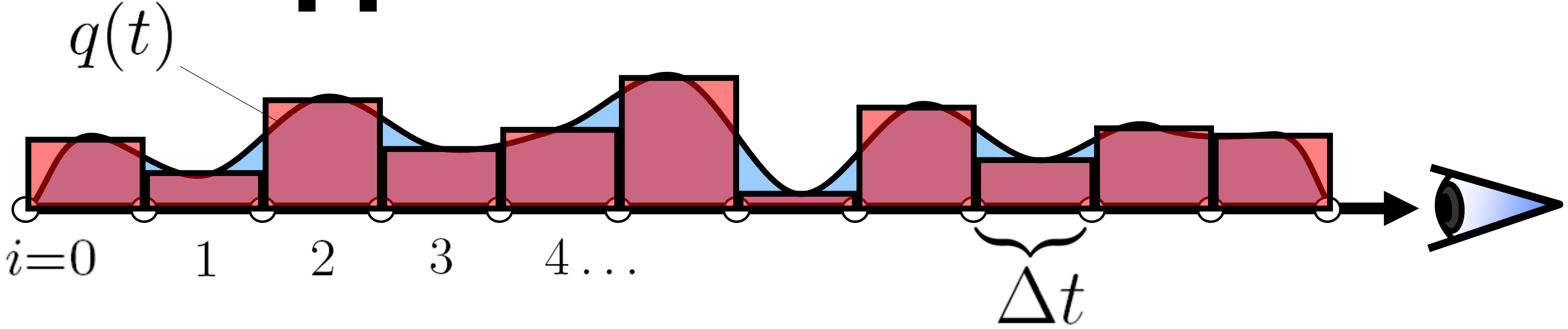


$$e^{-\tilde{\tau}(0,t)} = \prod_{i=0}^{\lfloor t/\Delta t \rfloor} (1 - A_i)$$

$$q(t) \approx C_i = c(i \cdot \Delta t)\,\Delta t$$

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

Numerical approximation



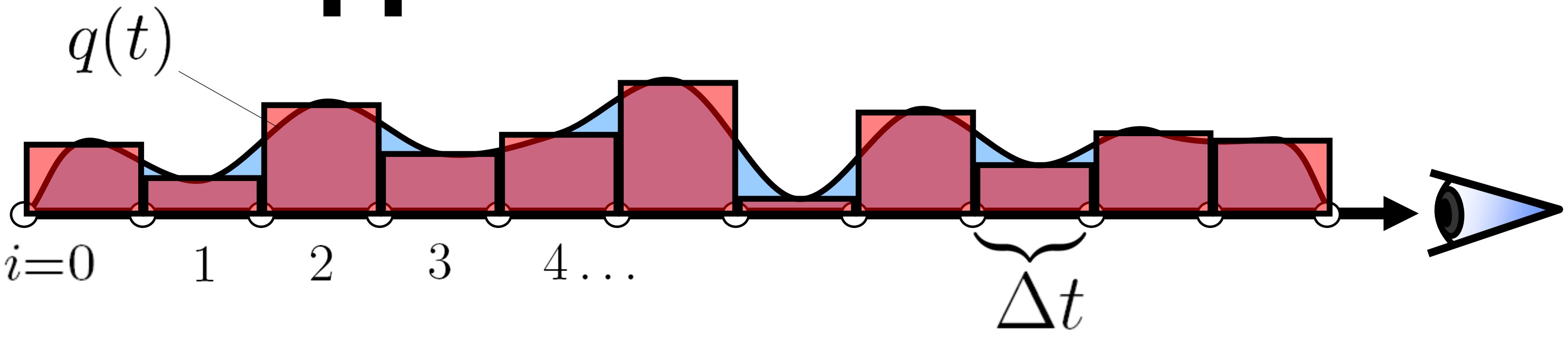
$$\tilde{C} = \sum_{i=0}^{\lfloor T/\Delta t \rfloor} C_i \prod_{j=0}^{i-1} (1 - A_j)$$

can be computed recursively:

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

- C'_i: radiant energy observed at position i
- C_i: radiant energy emitted at position i
- A_i: absorption at position i
- C'_{i-1}: radiant energy observed at position $i-1$

Numerical approximation



back-to-front compositing

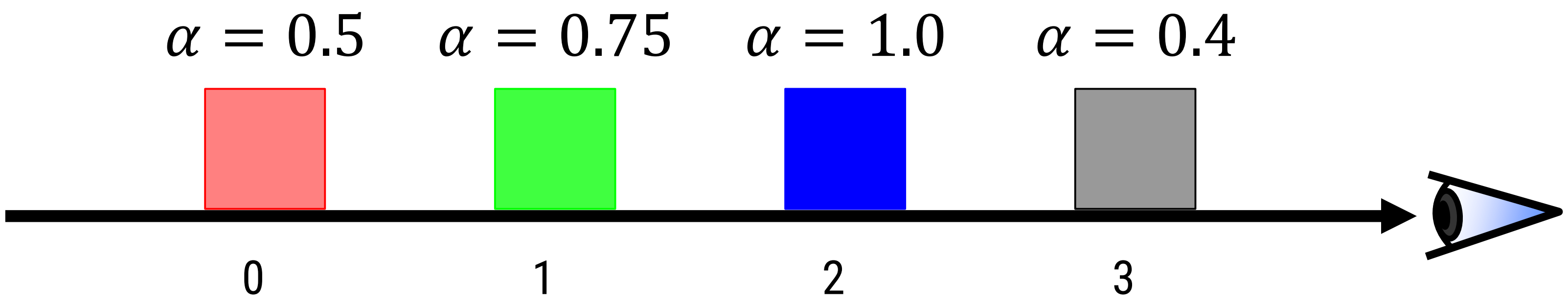
$$C'_i = C_i + (1 - A_i) C'_{i-1}$$

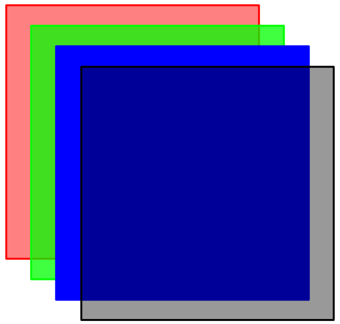
front-to-back compositing

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

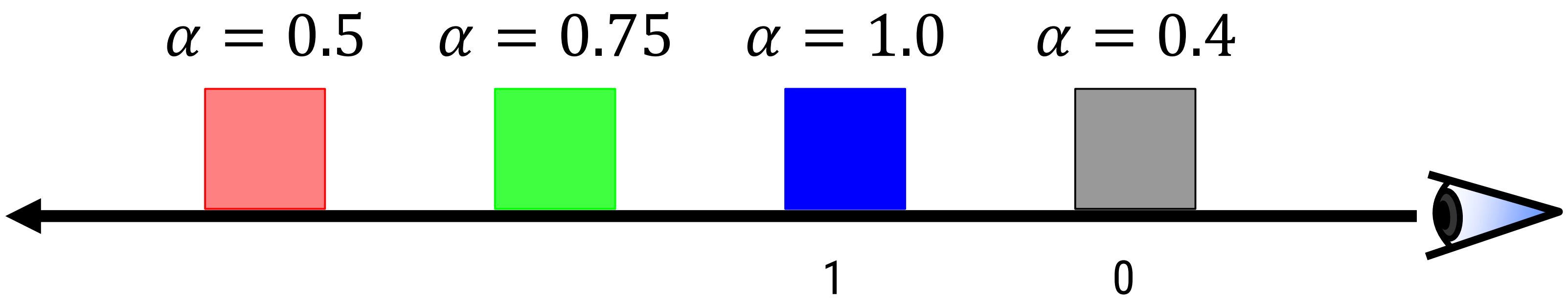
early ray termination: stop the calculation when $A'_i \approx 1$

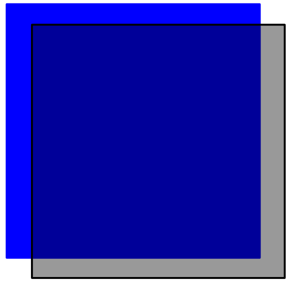
Back-to-front compositing





Front-to-back compositing





Summary of the approximation

- emission absorption model

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

- numerical solutions **[pre-multiplied alpha assumed]**

back-to-front iteration

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

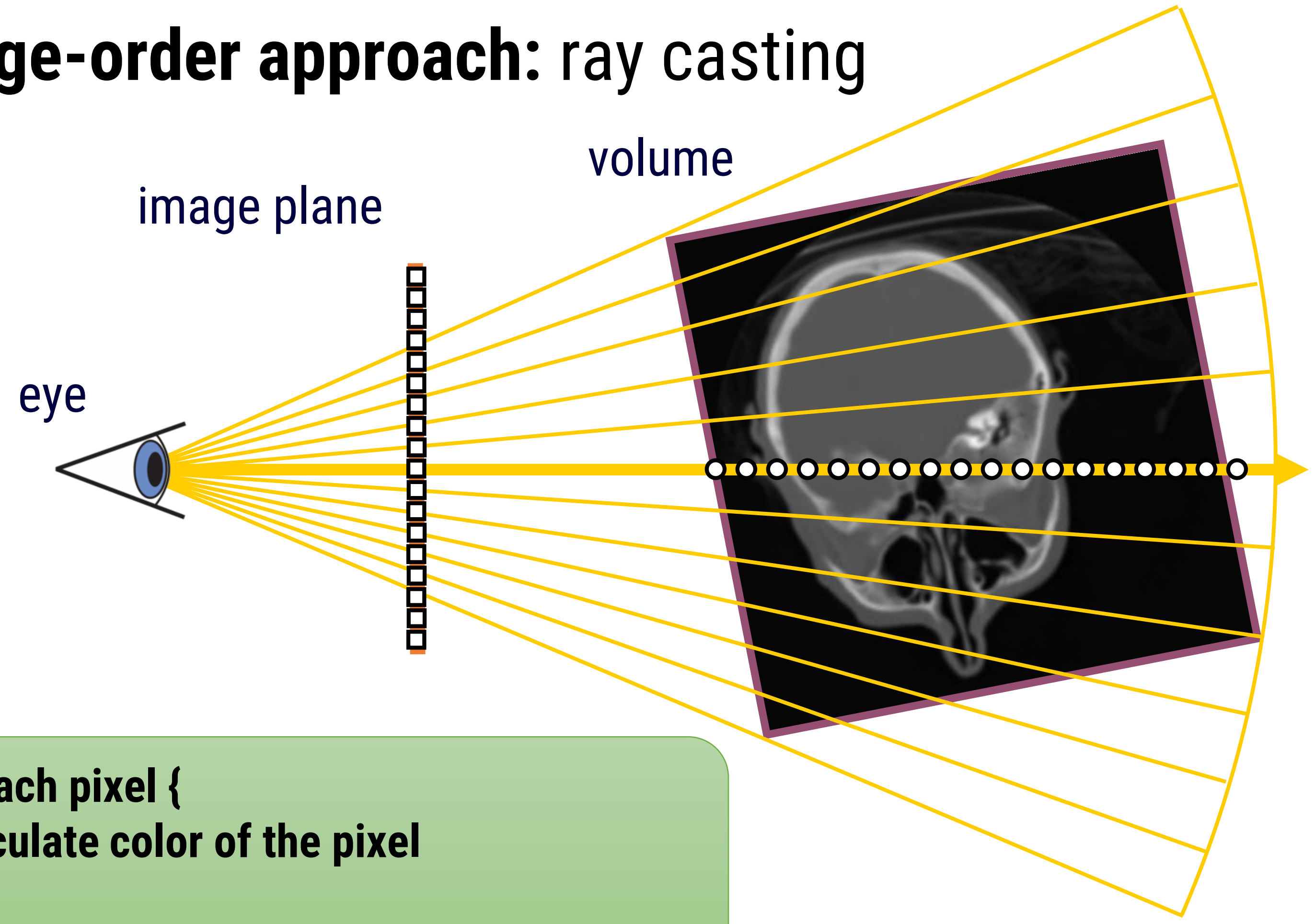
front-to-back iteration

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

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

Image vs. object order

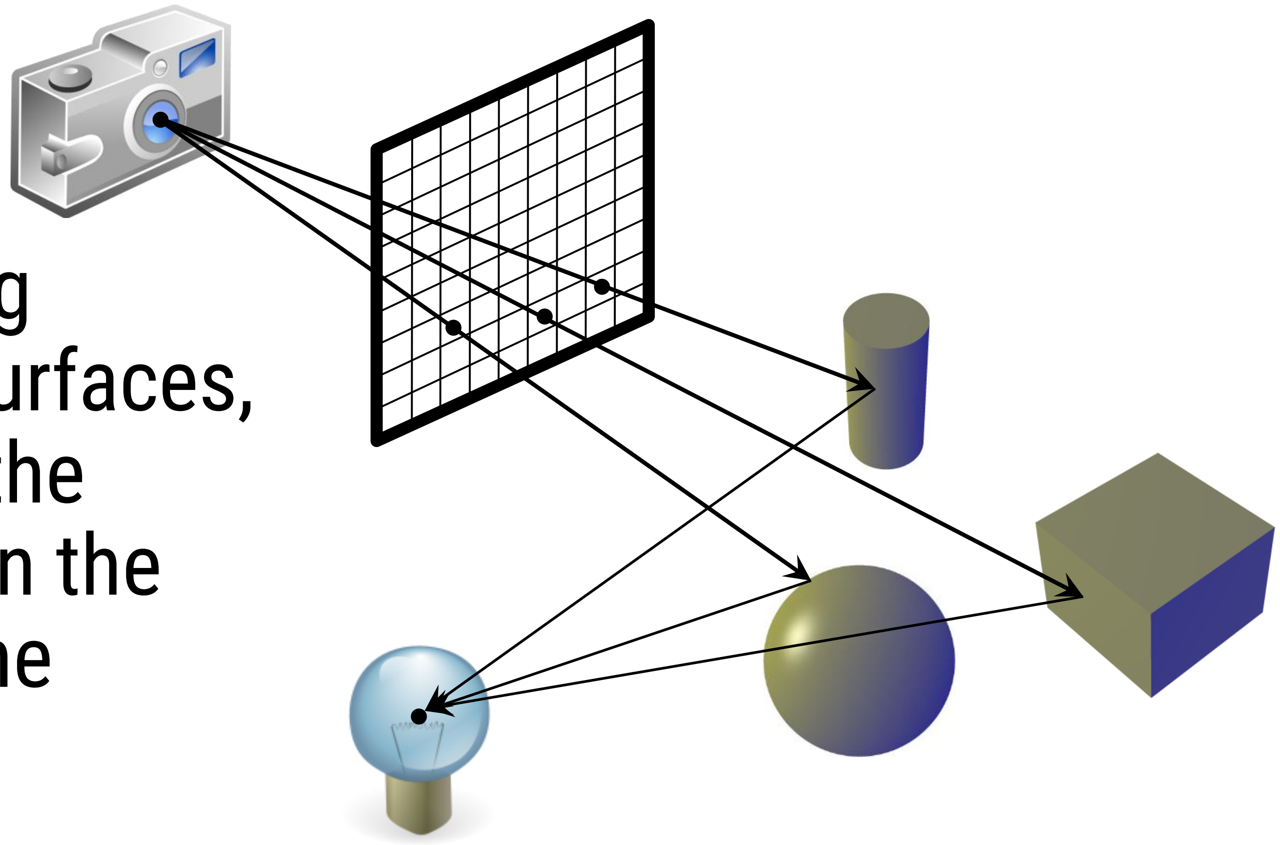
image-order approach: ray casting

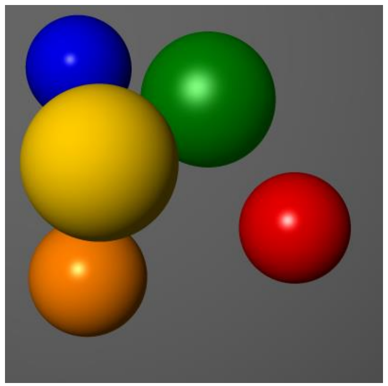


For each pixel {
  calculate color of the pixel
}

Side note: Difference to “Raytracing”

- in traditional ray casting and ray tracing we use only surfaces, so we ignore the space between the camera and the first hit point





Side note: Difference to "Raytracing"

- in ray casting for direct volume rendering, we consider all samples between the eye and the end of the dataset

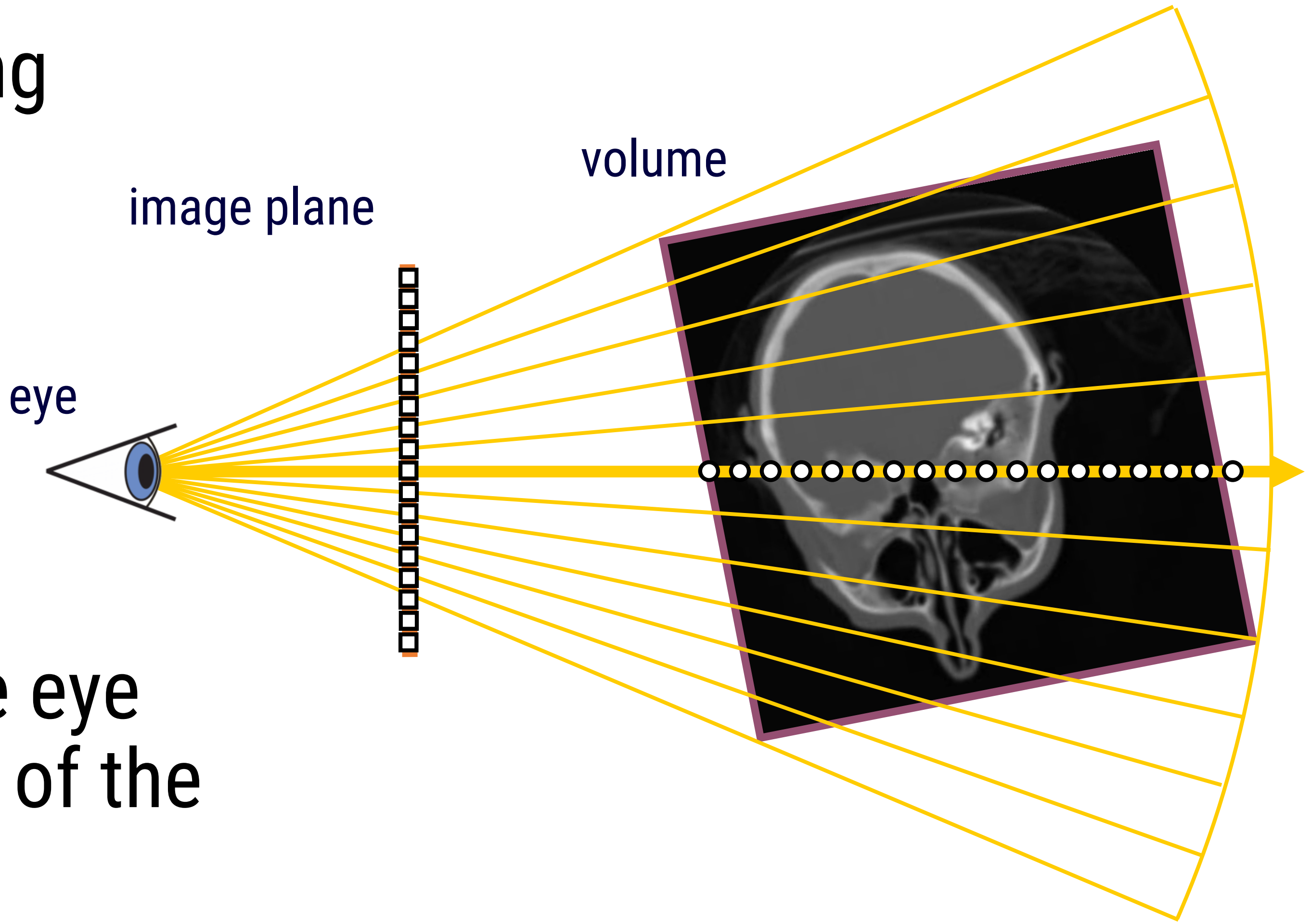
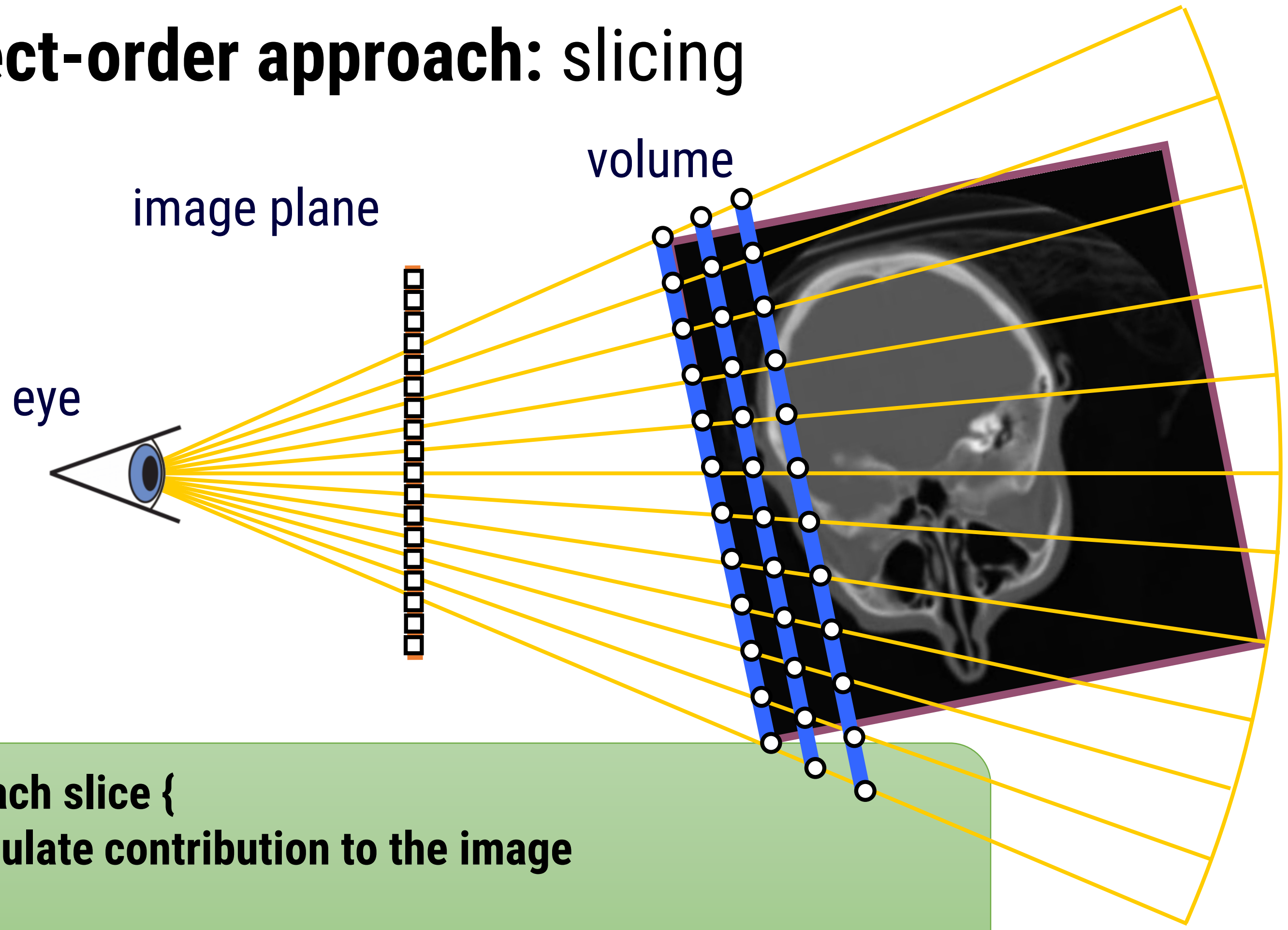


Image vs. object order

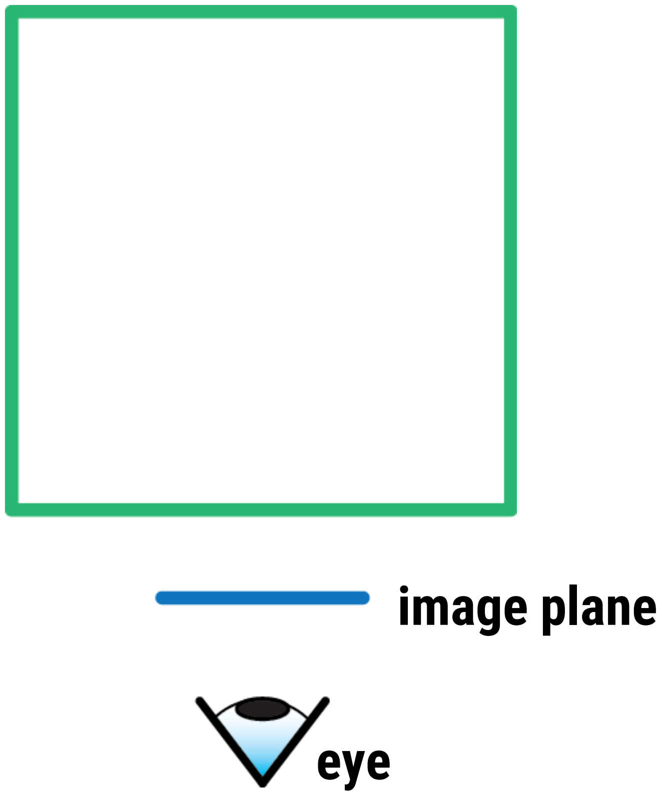
object-order approach: slicing



For each slice {
calculate contribution to the image
}

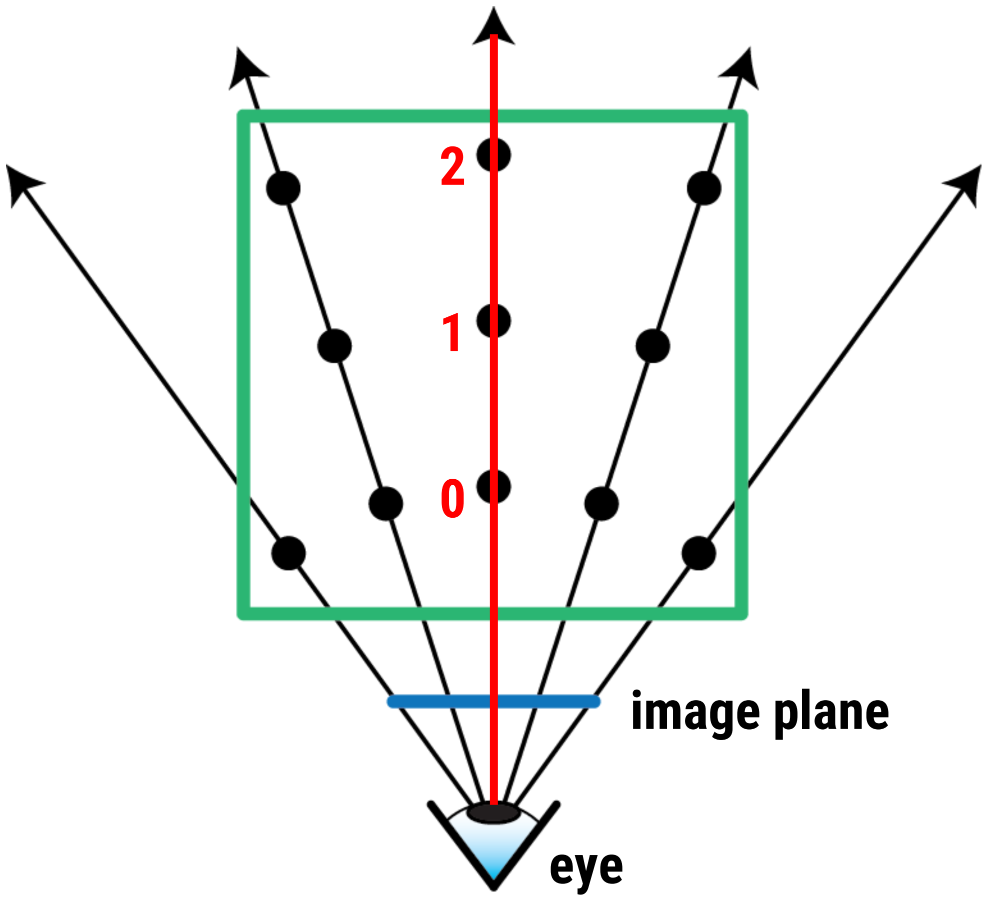
Ray casting vs. slicing

- today, standard ray casting can be implemented on the GPU
- on previous hardware generations, only slicing was possible



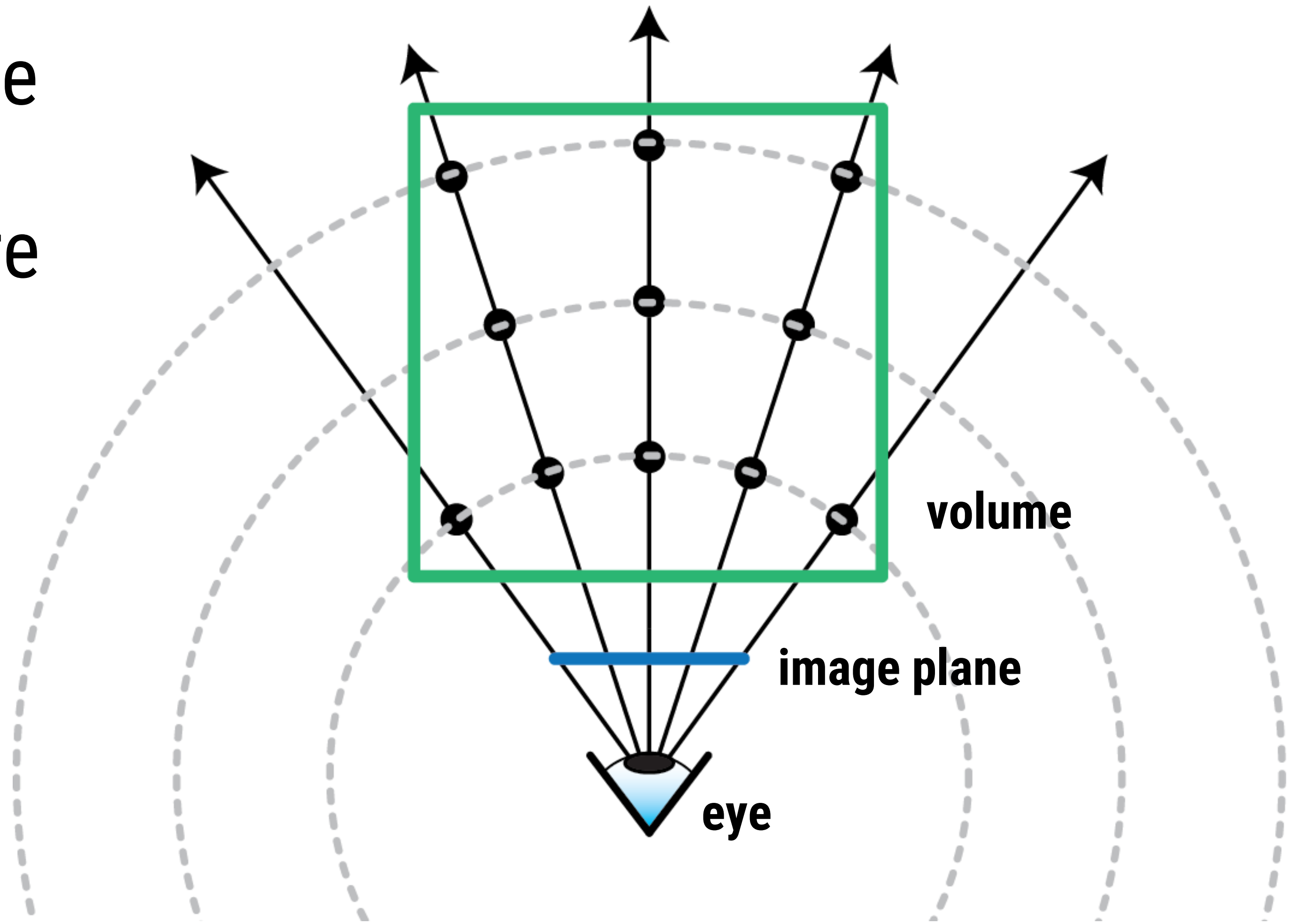
Ray casting vs. slicing

- in ray casting, sampling is typically performed at equidistant points along each ray



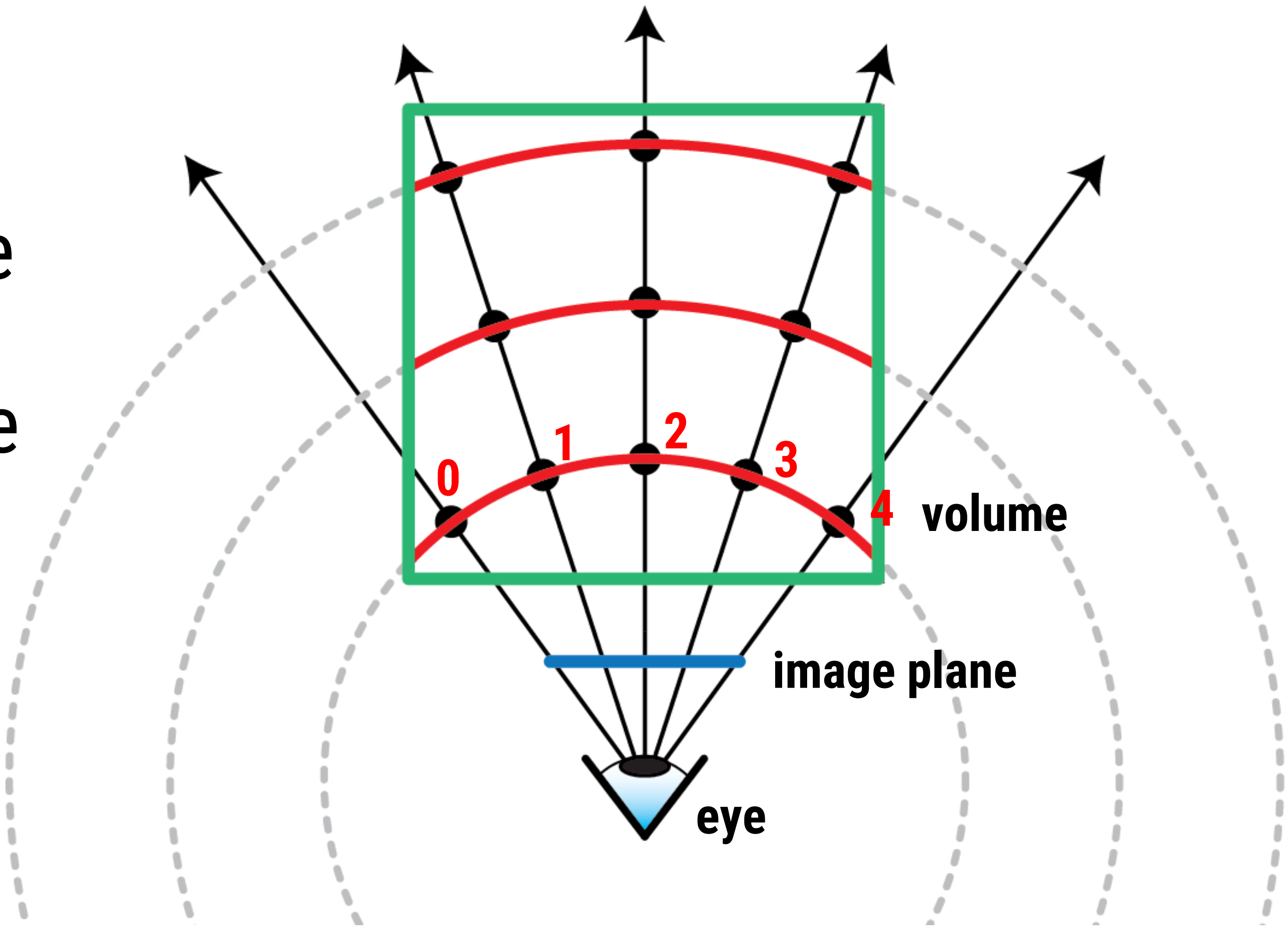
Ray casting vs. slicing

- under perspective projection, these sample points are located on concentric spherical shells



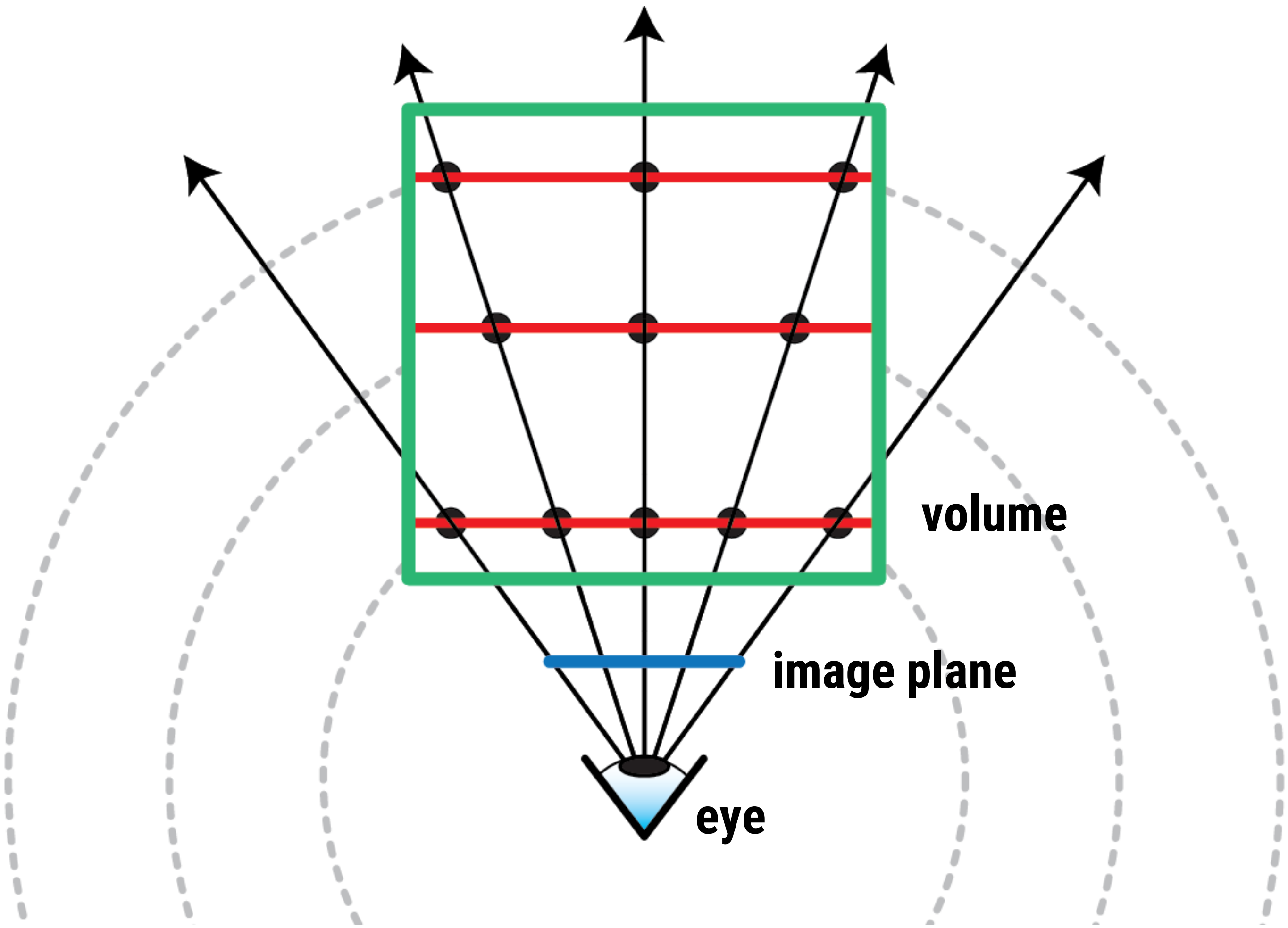
Ray casting vs. slicing

- this is equivalent to sampling all the points on one shell before proceeding to the next one



Ray casting vs. slicing

- in slicing, the shells are typically approximated by view-aligned planes



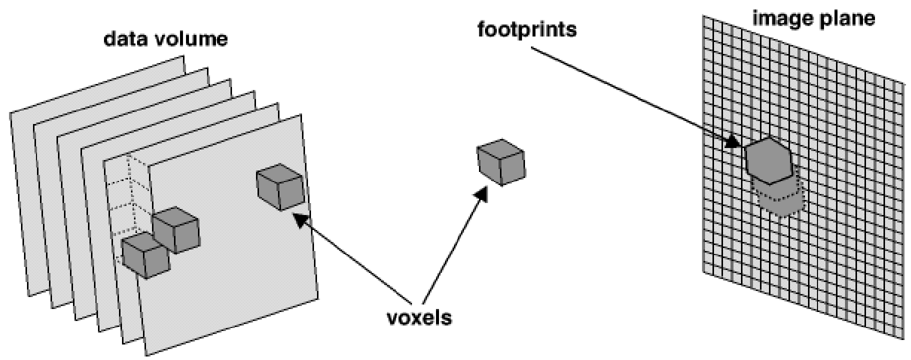
Ray casting vs. slicing

- slicing
  - advantages in terms of memory access pattern
  - sampling pattern can cause artifacts
  - optimizations (e.g., early ray termination) difficult

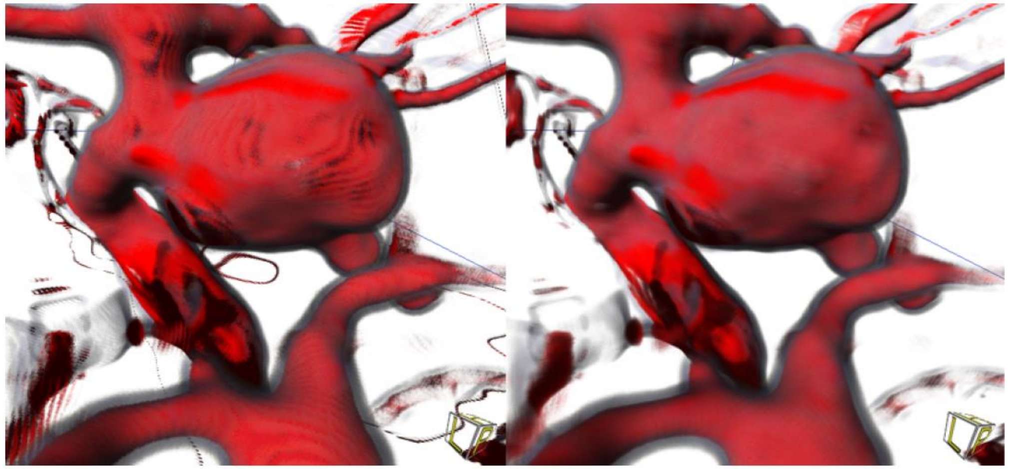
- ray casting
  - easier to implement on current hardware
  - early ray termination is trivial
  - more flexible than slicing

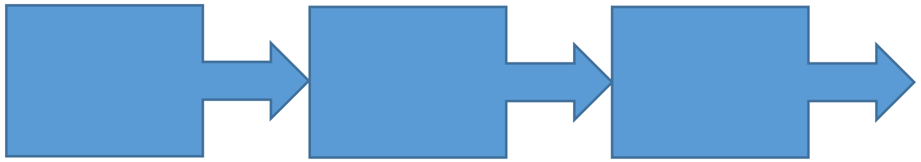
Splatting

- alternative object-order approach
  - project voxel footprints (splats) onto image plane
  - traverse only non-transparent parts of the volume
  - most useful if data is very sparse → few splats
  - rarely used nowadays (does not map well to GPUs)
  - can have quality advantages



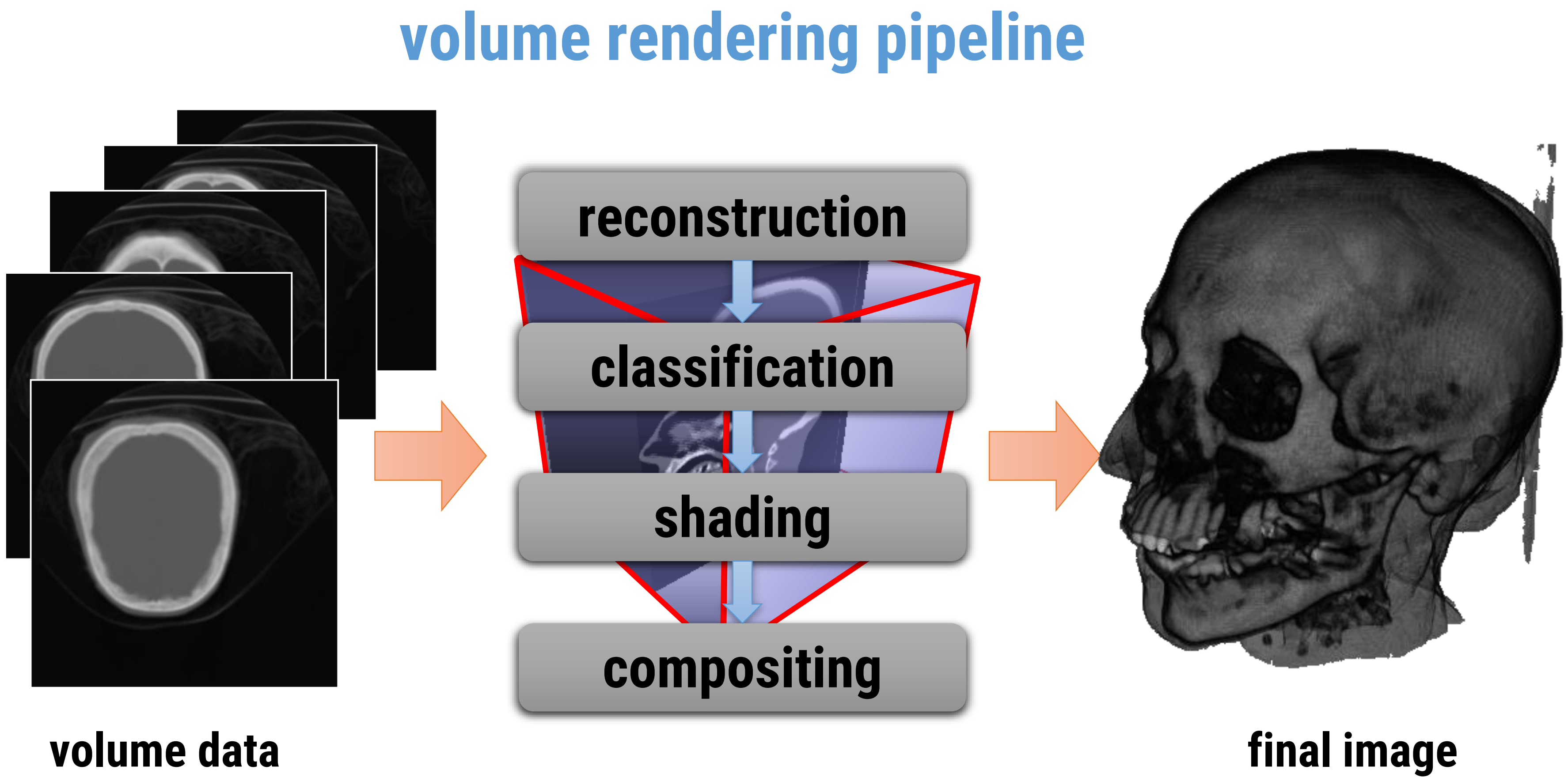
slicing (left) vs. splatting (right)





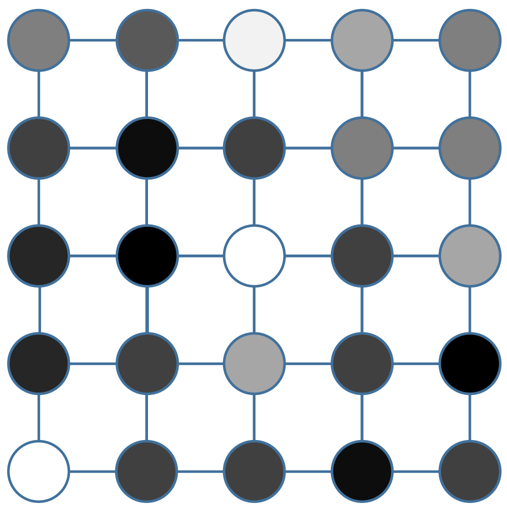
Volume Rendering Pipeline

Volume rendering pipeline



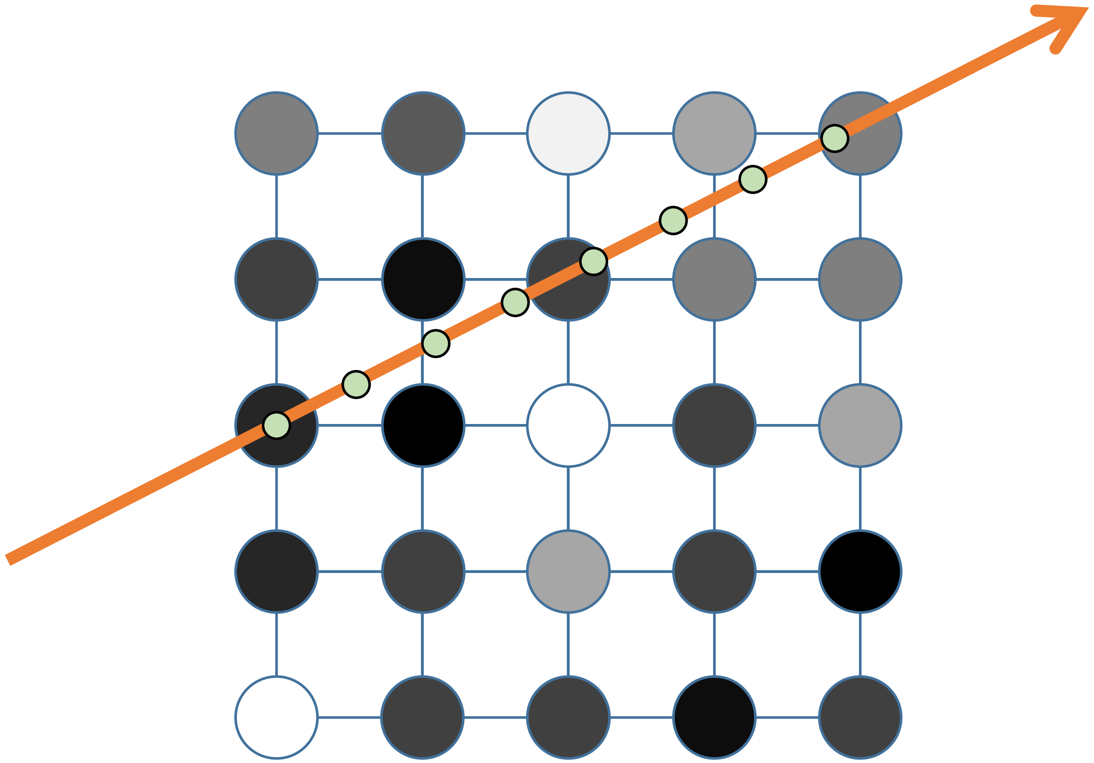
Reconstruction

- usually, volume data sets are a grid of discrete samples
  - for rendering purposes, we want to treat them as continuous three-dimensional functions



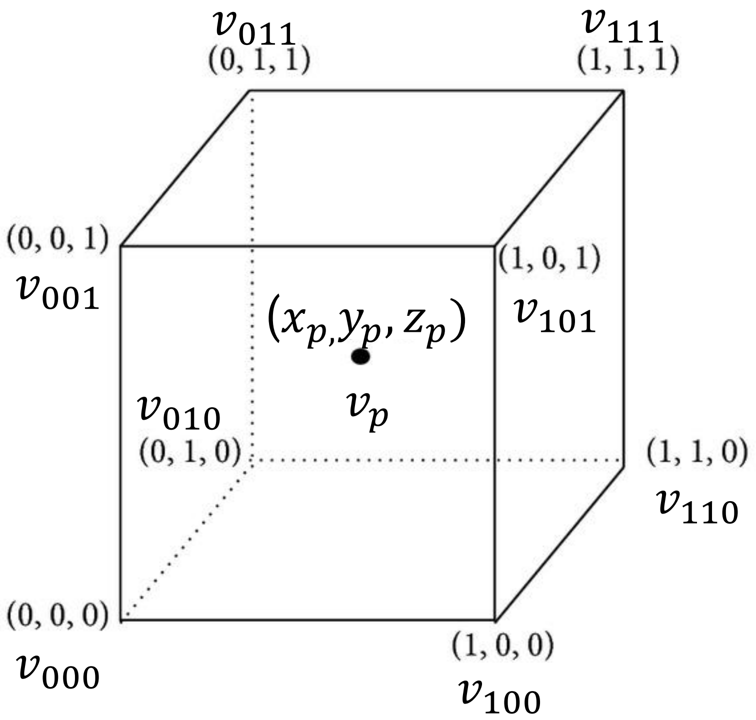
- we need to choose an appropriate reconstruction filter (i.e., way of interpolation between the samples)
  - requirements: high-quality reconstruction, but small performance overhead

Reconstruction



Reconstruction

- simple extension of linear interpolation to three dimensions
- advantage: current GPUs automatically do trilinear interpolation of 3D textures



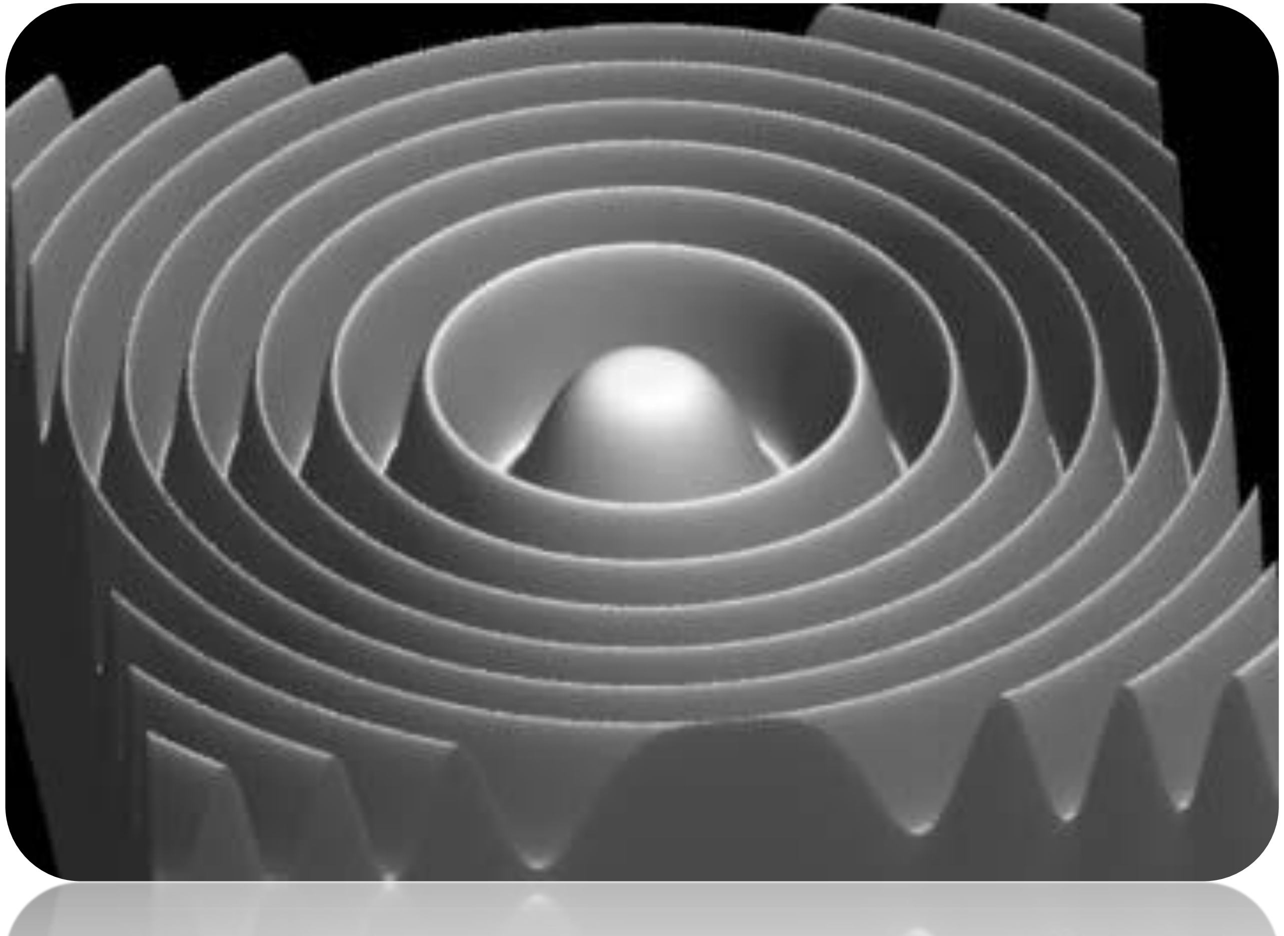
$$\begin{aligned} v_p \quad = \quad & v_{000}(1-x_p)(1-y_p)(1-z_p)+ \\ & v_{100}x_p(1-y_p)(1-z_p)+ \\ & v_{010}(1-x_p)y_p(1-z_p)+ \\ & v_{001}(1-x_p)(1-y_p)z_p+ \\ & v_{011}(1-x_p)y_pz_p+ \\ & v_{101}x_p(1-y_p)z_p+ \\ & v_{110}x_py_p(1-z_p)+ \\ & v_{111}x_py_pz_p \end{aligned}$$

Reconstruction filters

- if **higher quality** is needed, **more complex reconstruction filters** may be required; need **quality test**:
  - **Marschner-Lobb** function is a common test signal to evaluate the quality of reconstruction filters [Marschner and Lobb 1994]
  - the signal has a high amount of its energy near its Nyquist frequency
  - Nyquist frequency: the minimum rate at which a signal can be sampled without introducing errors, which is twice the highest frequency present in the signal
  - makes it a very demanding test for accurate reconstruction

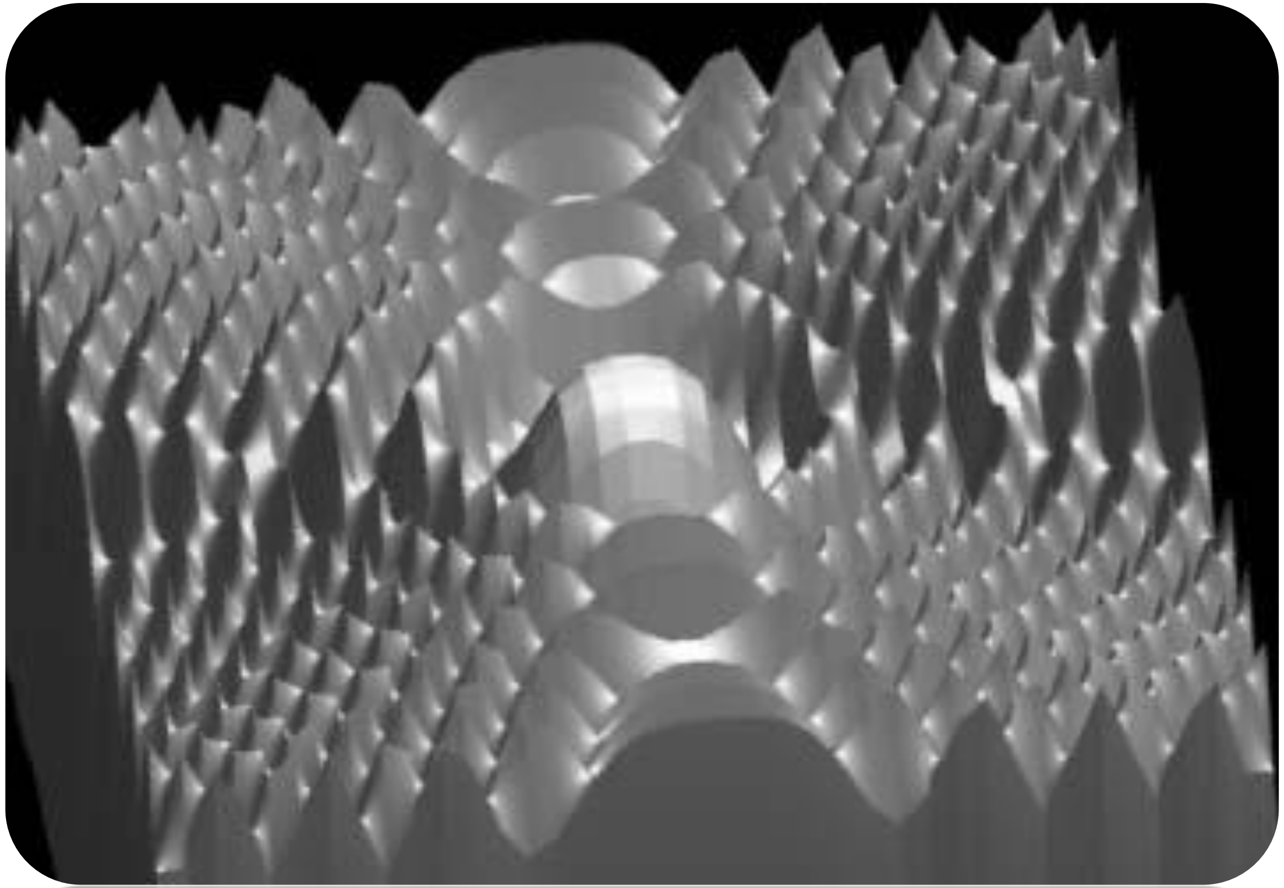
Reconstruction filters

- **Marschner-Lobb** test signal (analytically evaluated)



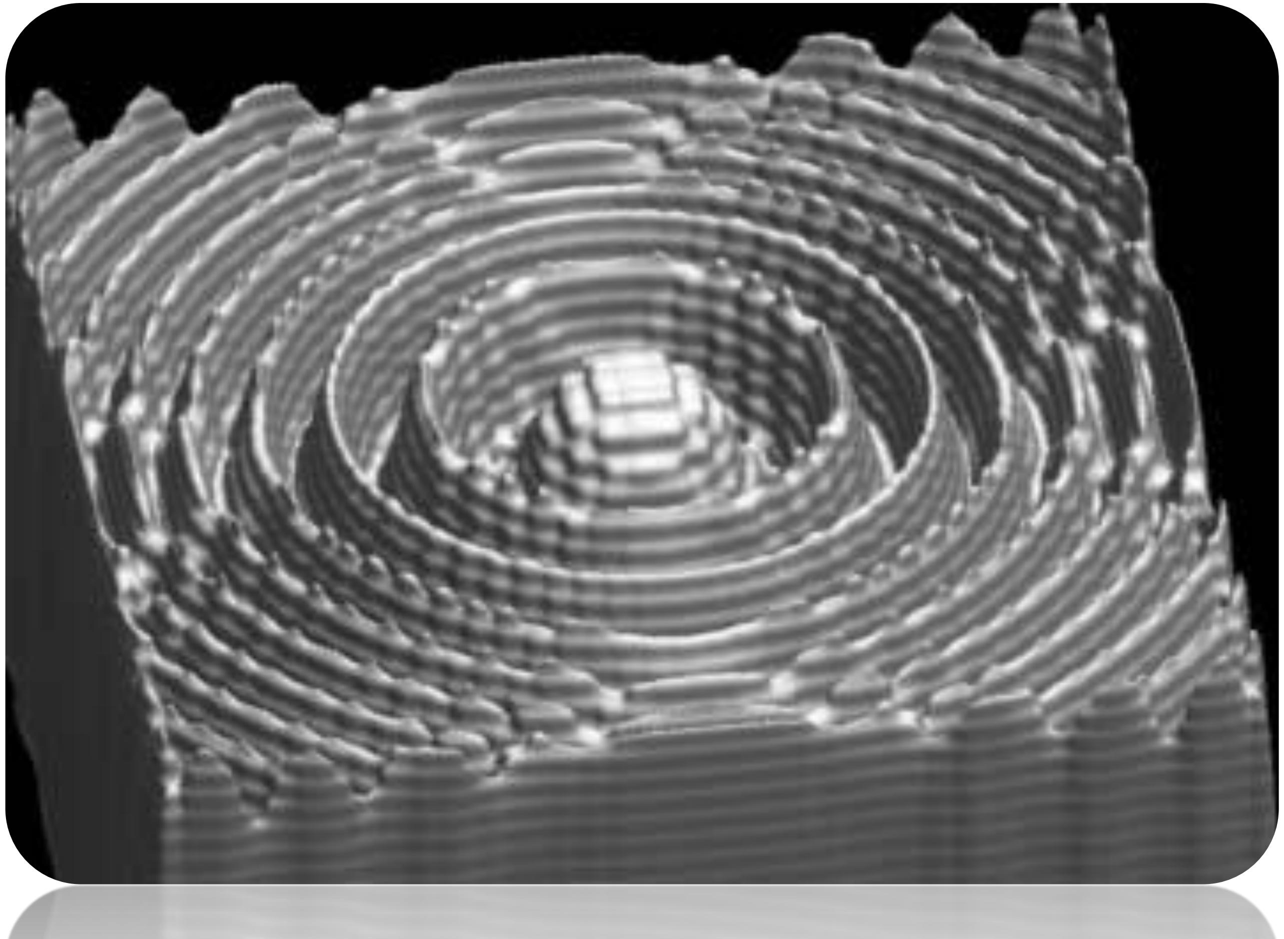
Reconstruction filters

- **trilinear** reconstruction of Marschner-Lobb test signal



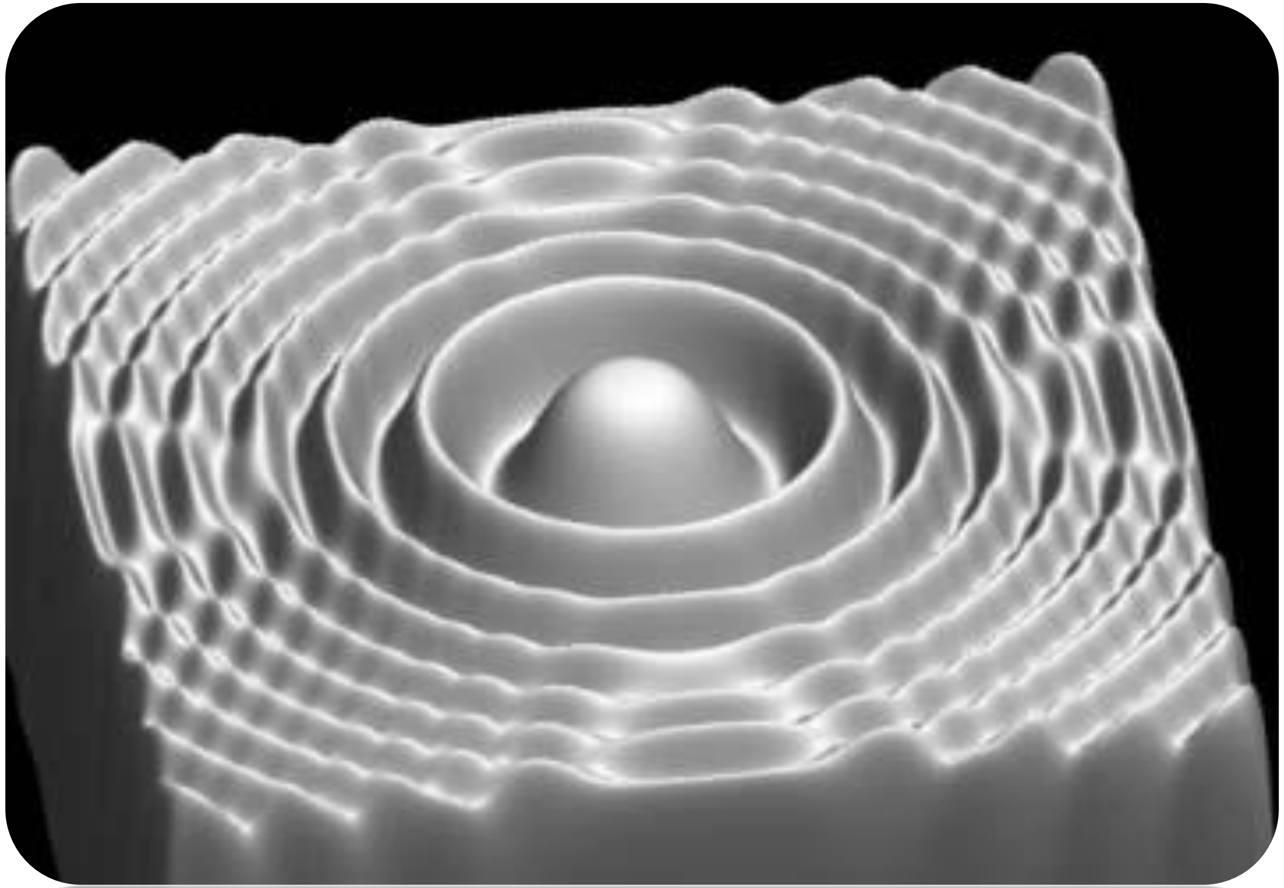
Reconstruction filters

- **cubic** reconstruction of Marschner-Lobb test signal



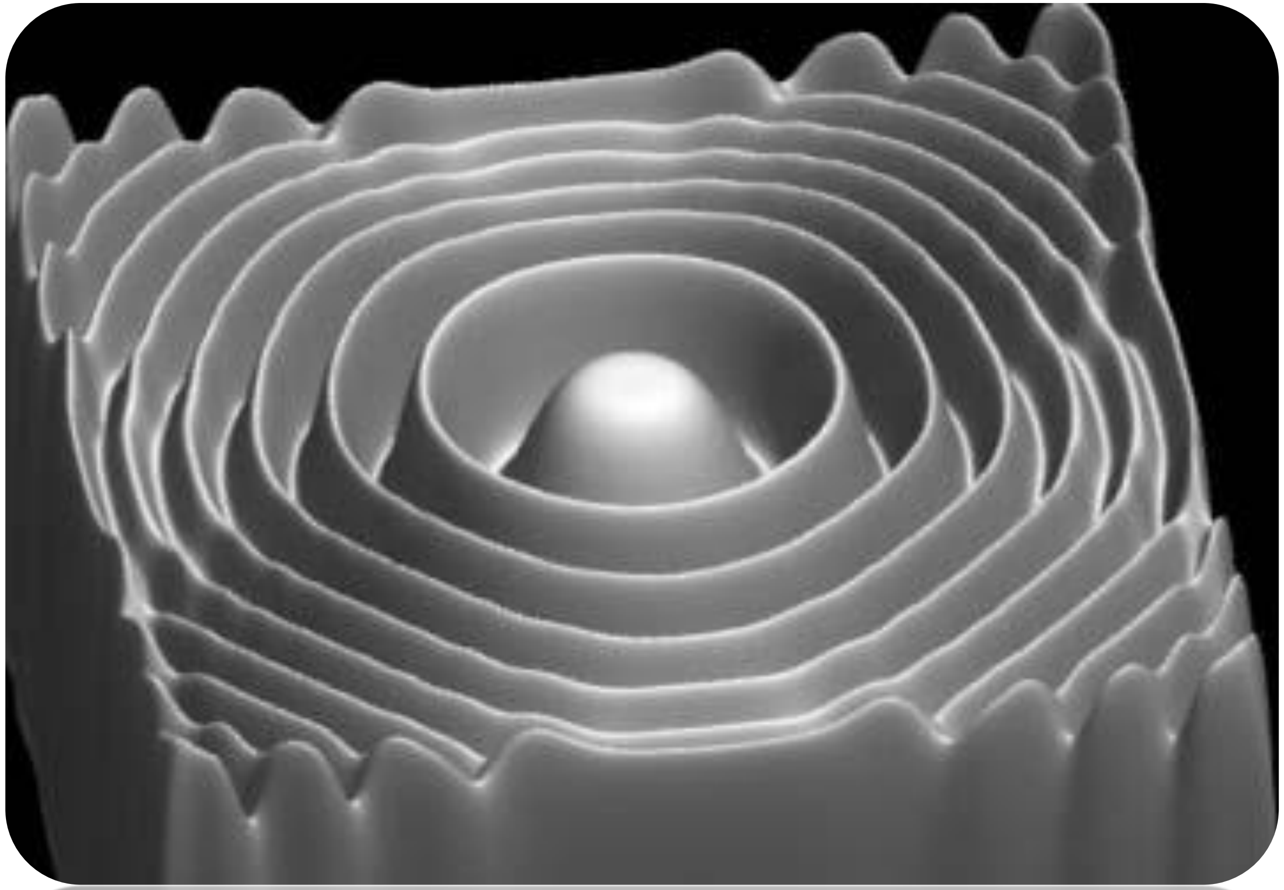
Reconstruction filters

- **b-spline** reconstruction of Marschner-Lobb test signal



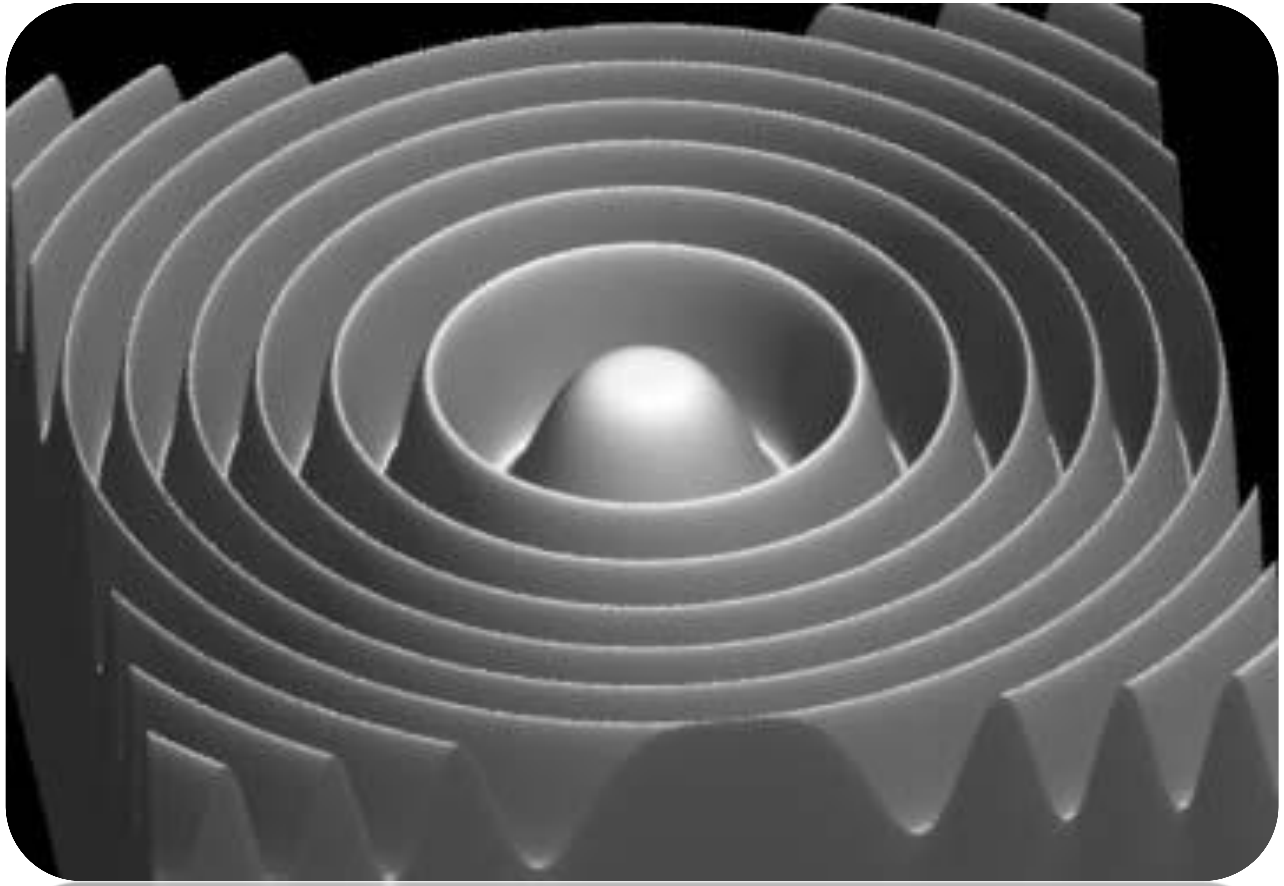
Reconstruction filters

- **windowed sinc** reconstruction of Marschner-Lobb test signal



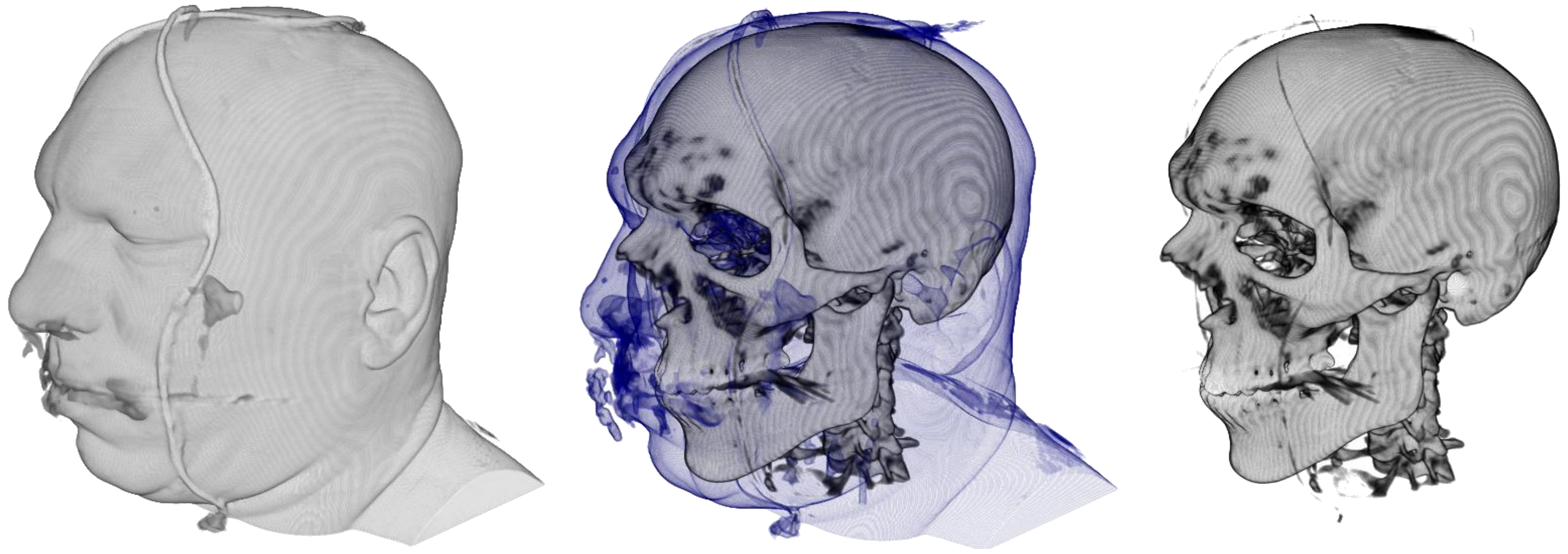
Reconstruction filters

- Marschner-Lobb test signal (analytically evaluated)



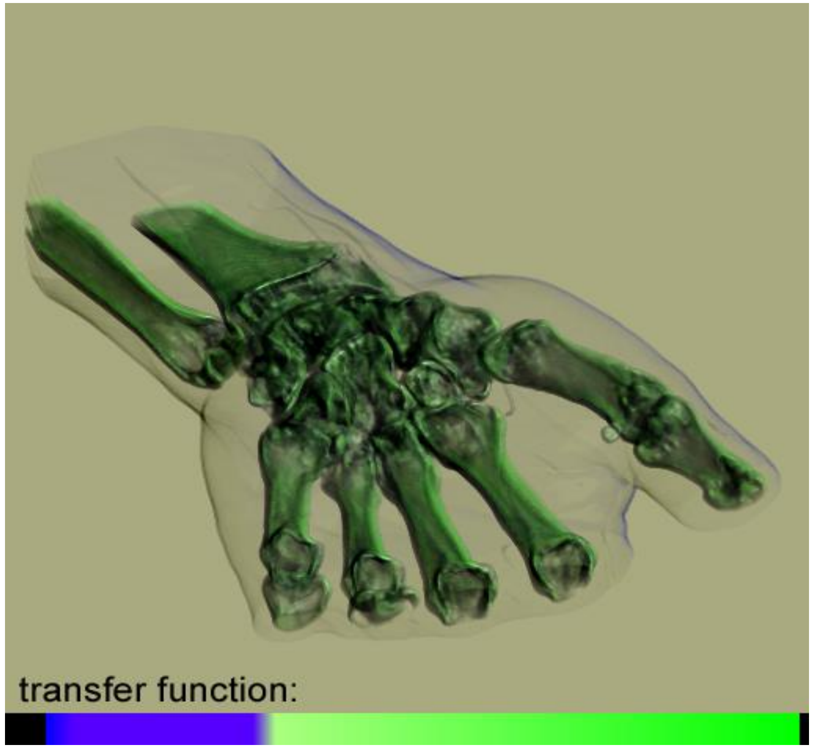
Classification

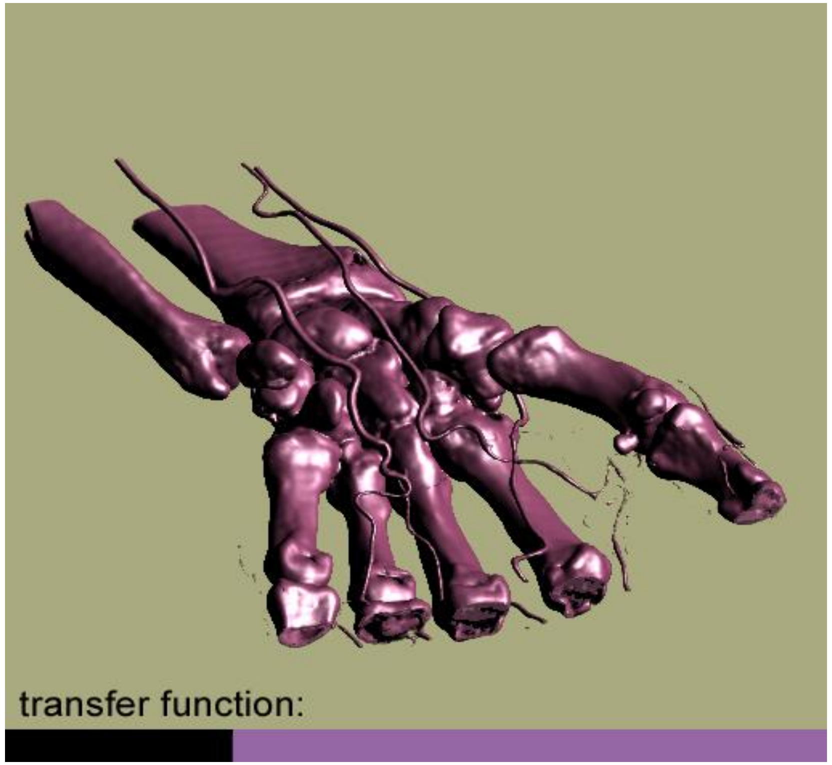
- definition of the appearence of the data by the user
  - Which parts are transparent?
  - Which parts have which color?



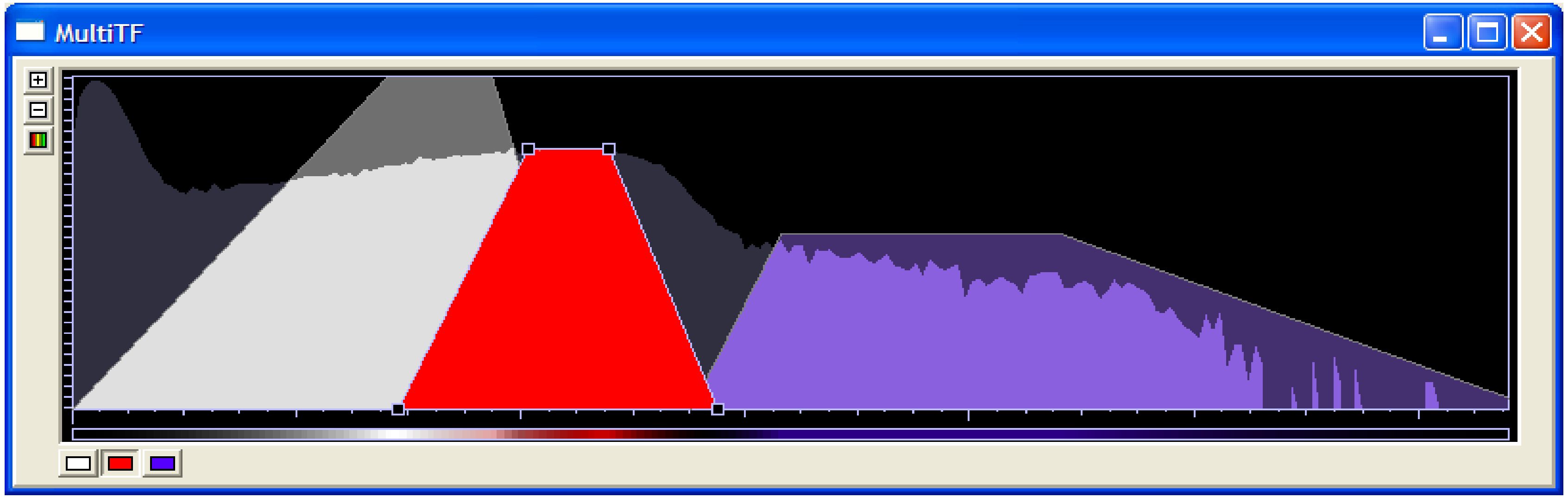
Classification: Transfer functions

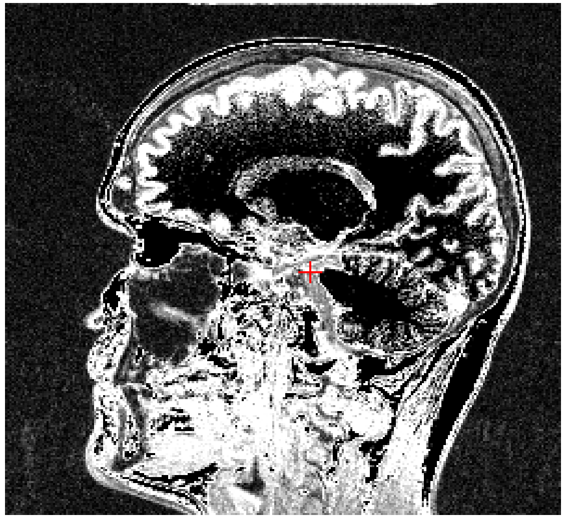
- mapping of data attributes to optical properties
- simplest: color table with opacity over data value

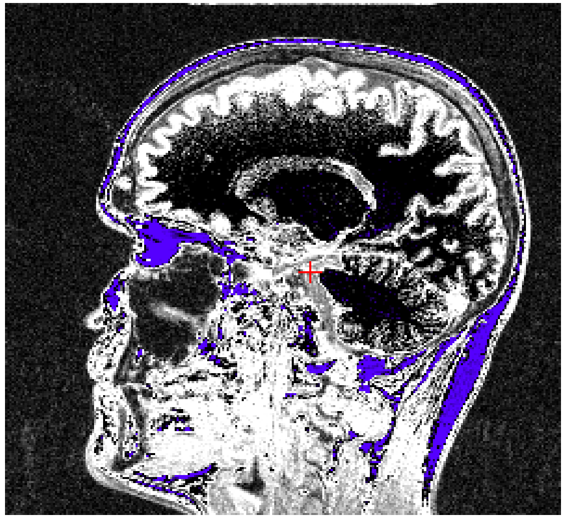


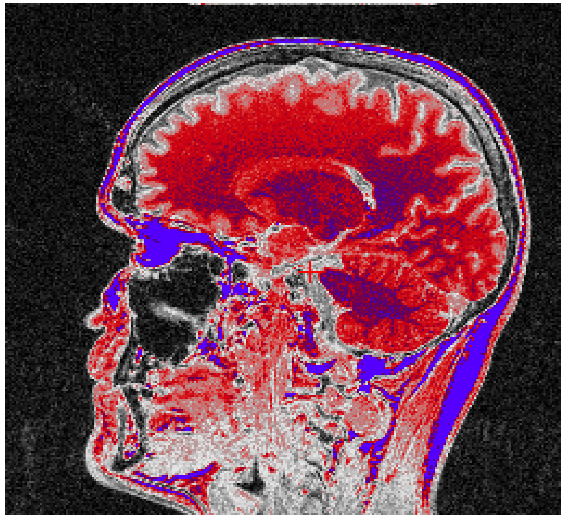


Classification: Transfer functions

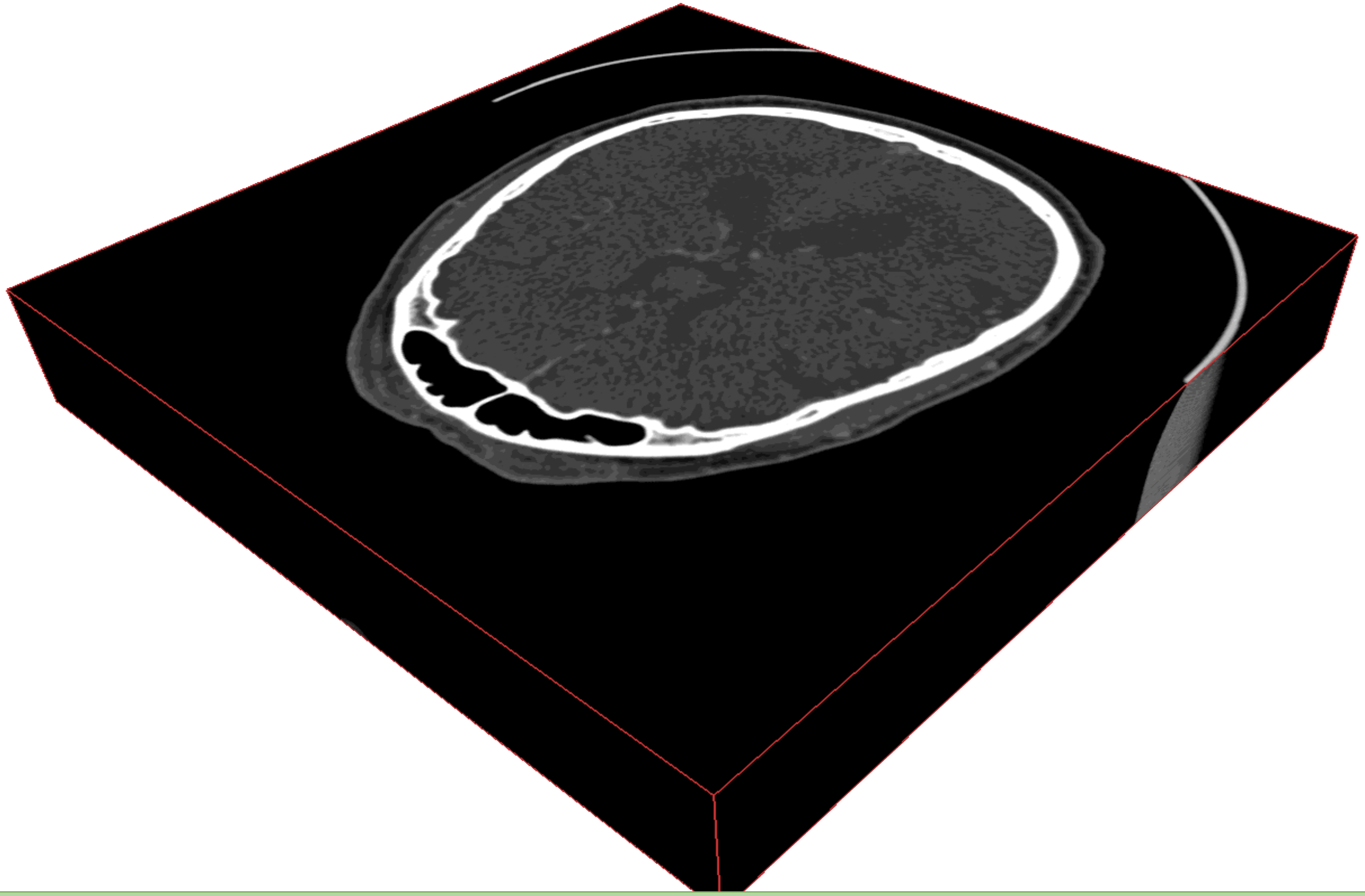








Classification: Transfer functions



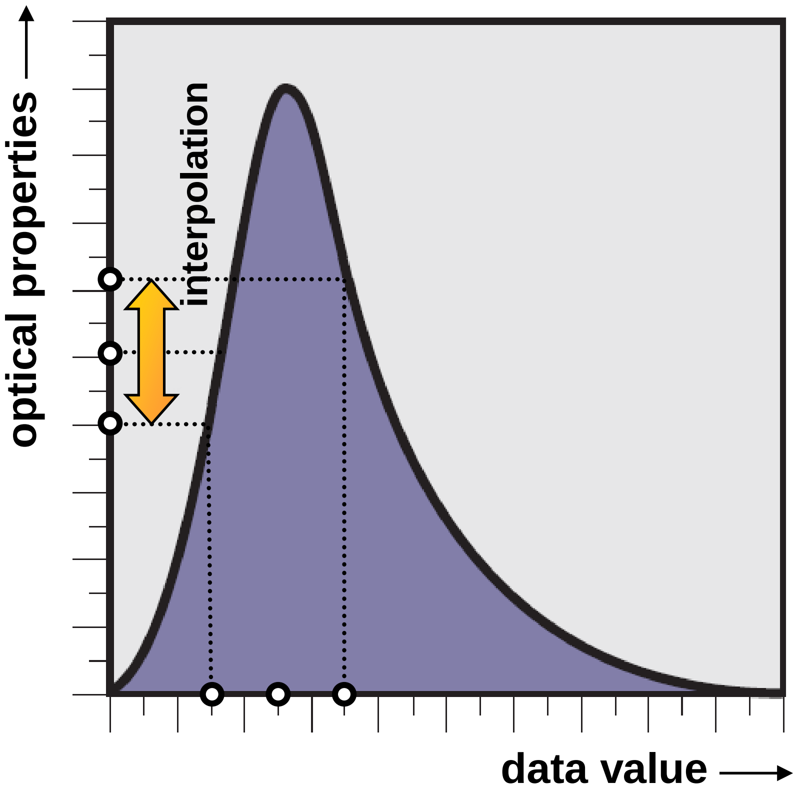
real-time update of the transfer function important

Classification order

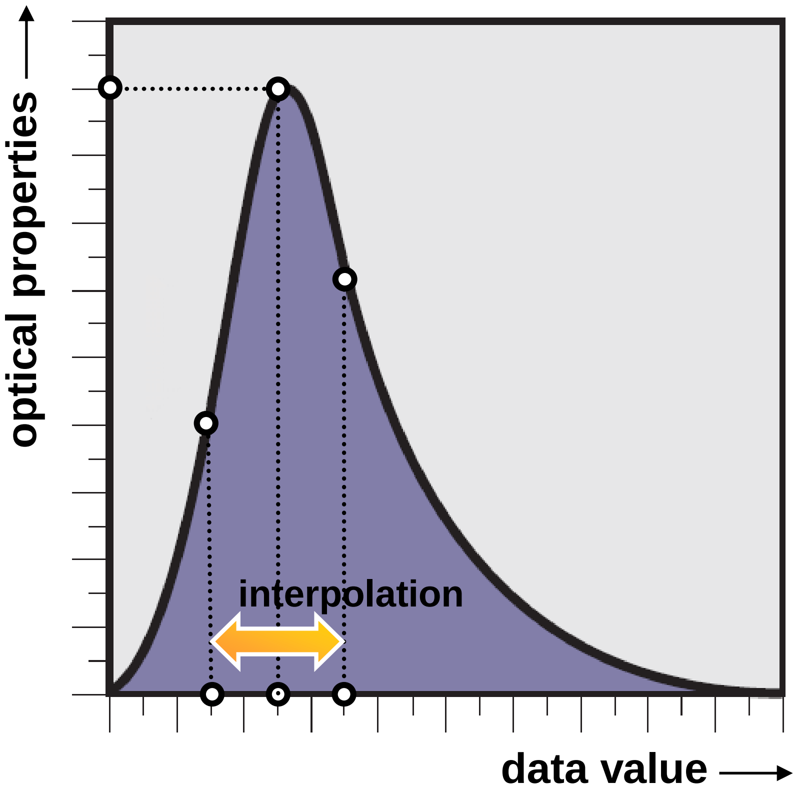
- classification can occur before or after reconstruction
  - this decision has significant impact on quality
- **pre-interpolative classification**
  - classify all data values and then interpolate between RGBA-tuples
- **post-interpolative classification**
  - interpolate between scalar data values and then classify the result

Classification order

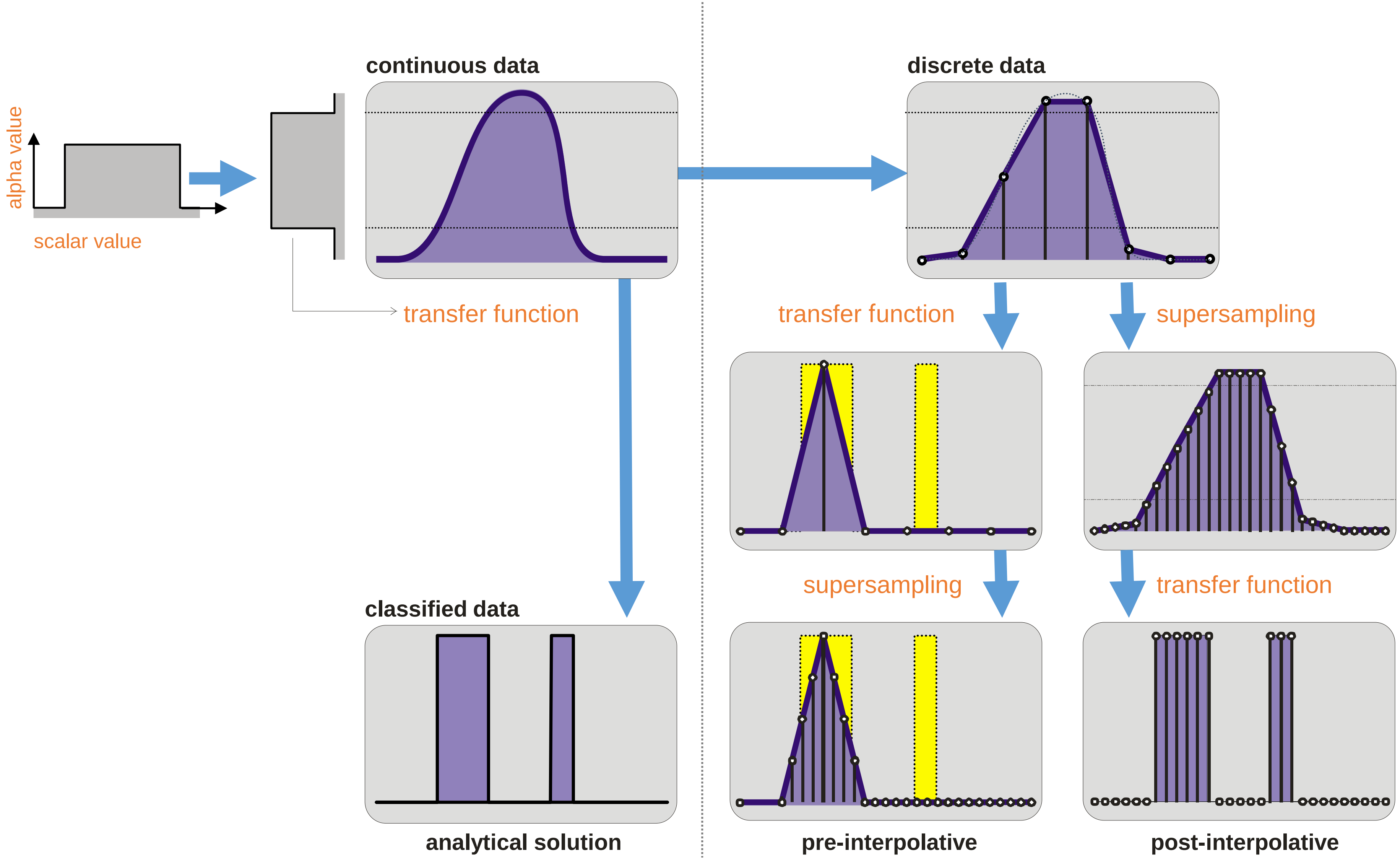
PRE-INTERPOLATIVE



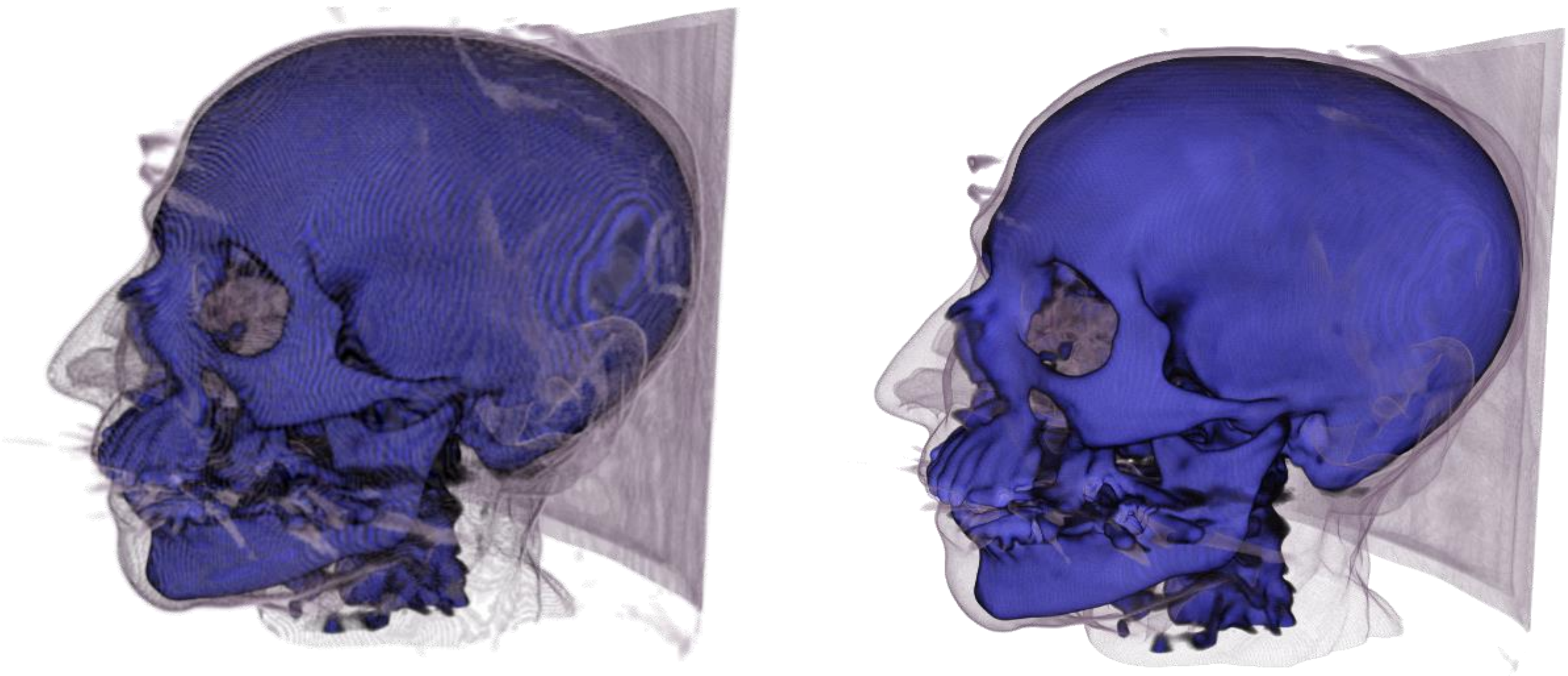
POST-INTERPOLATIVE



Classification order



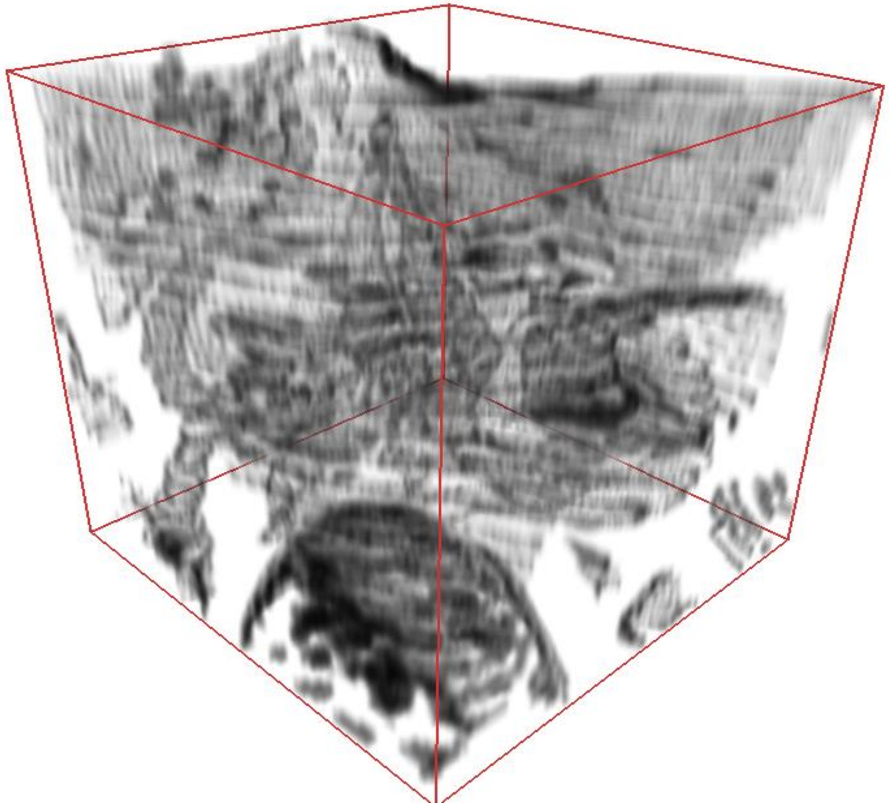
Classification order



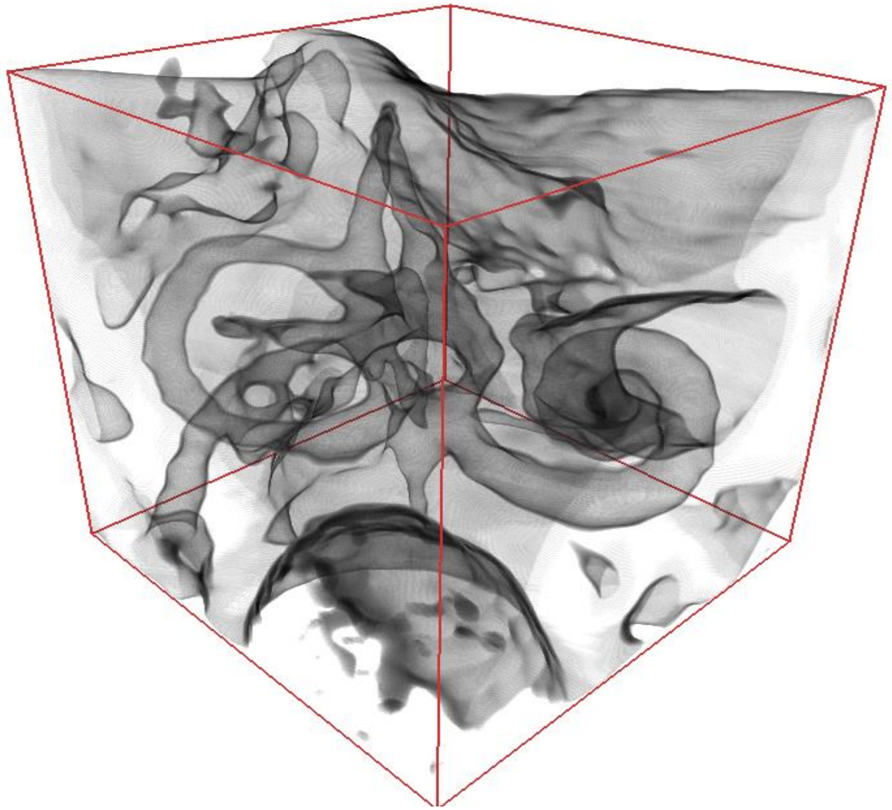
pre-interpolative | post-interpolative

same transfer function, resolution, and sampling rate

Classification order



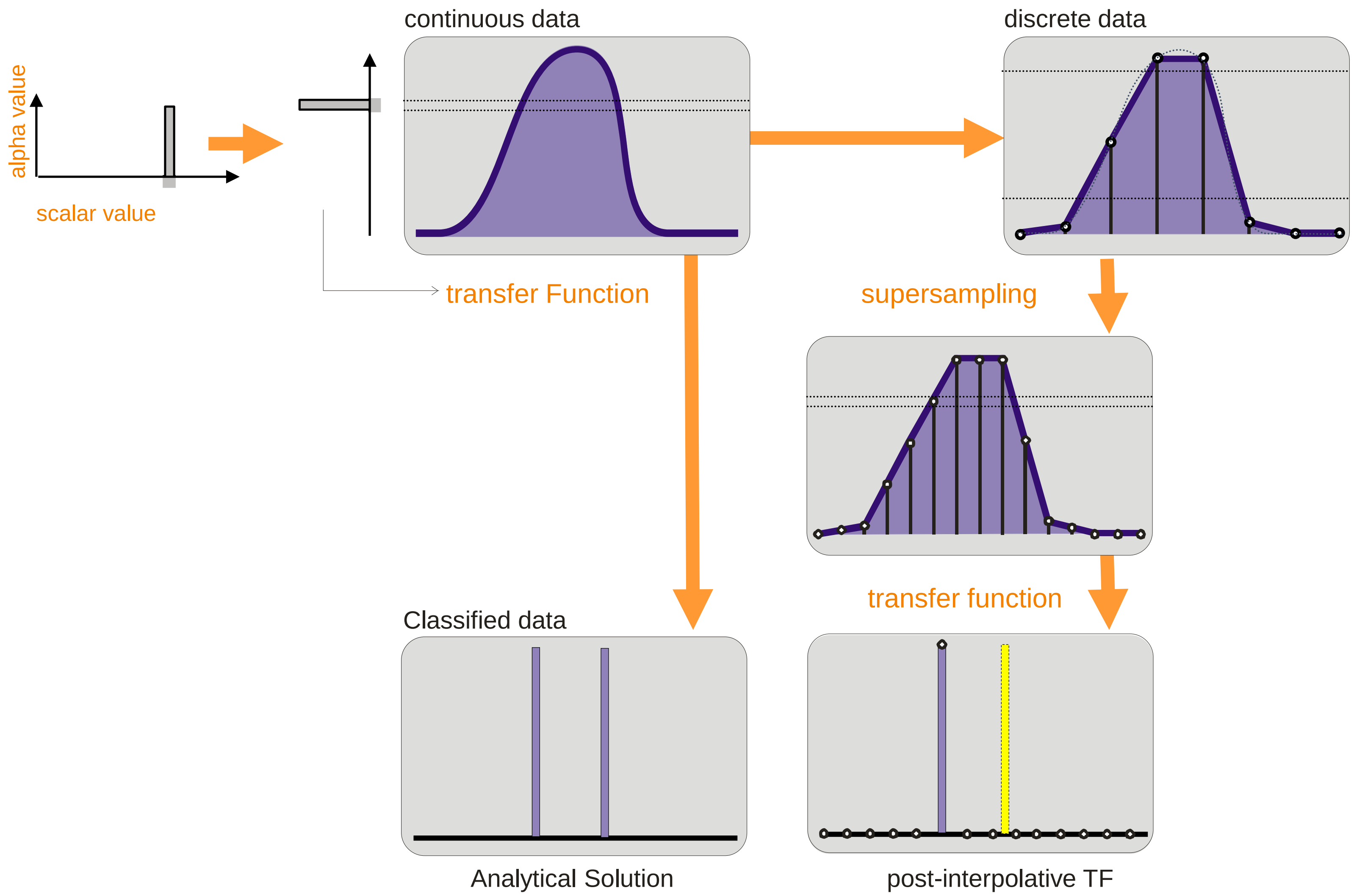
pre-interpolative



post-interpolative

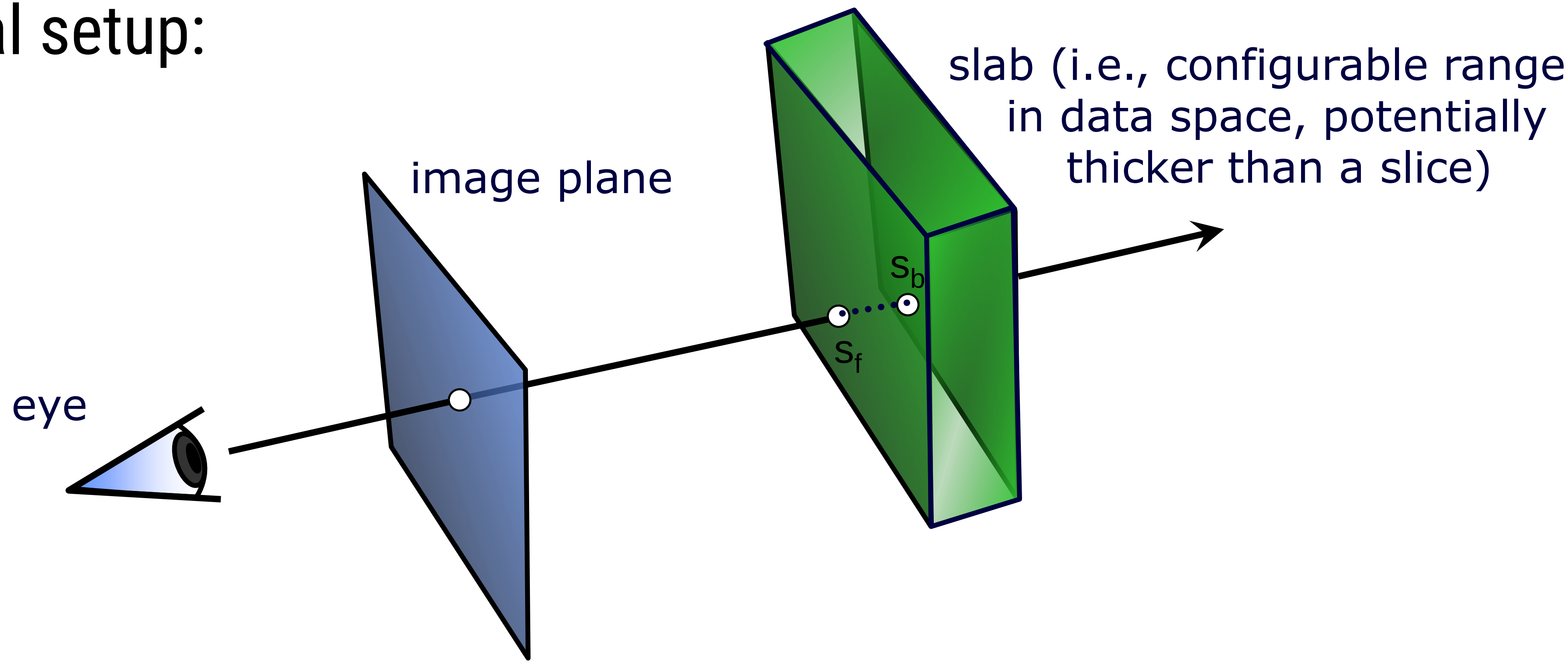
same transfer function, resolution, and sampling rate

Problem with small TF windows



Solution: Pre-integration

- assume the following general setup:

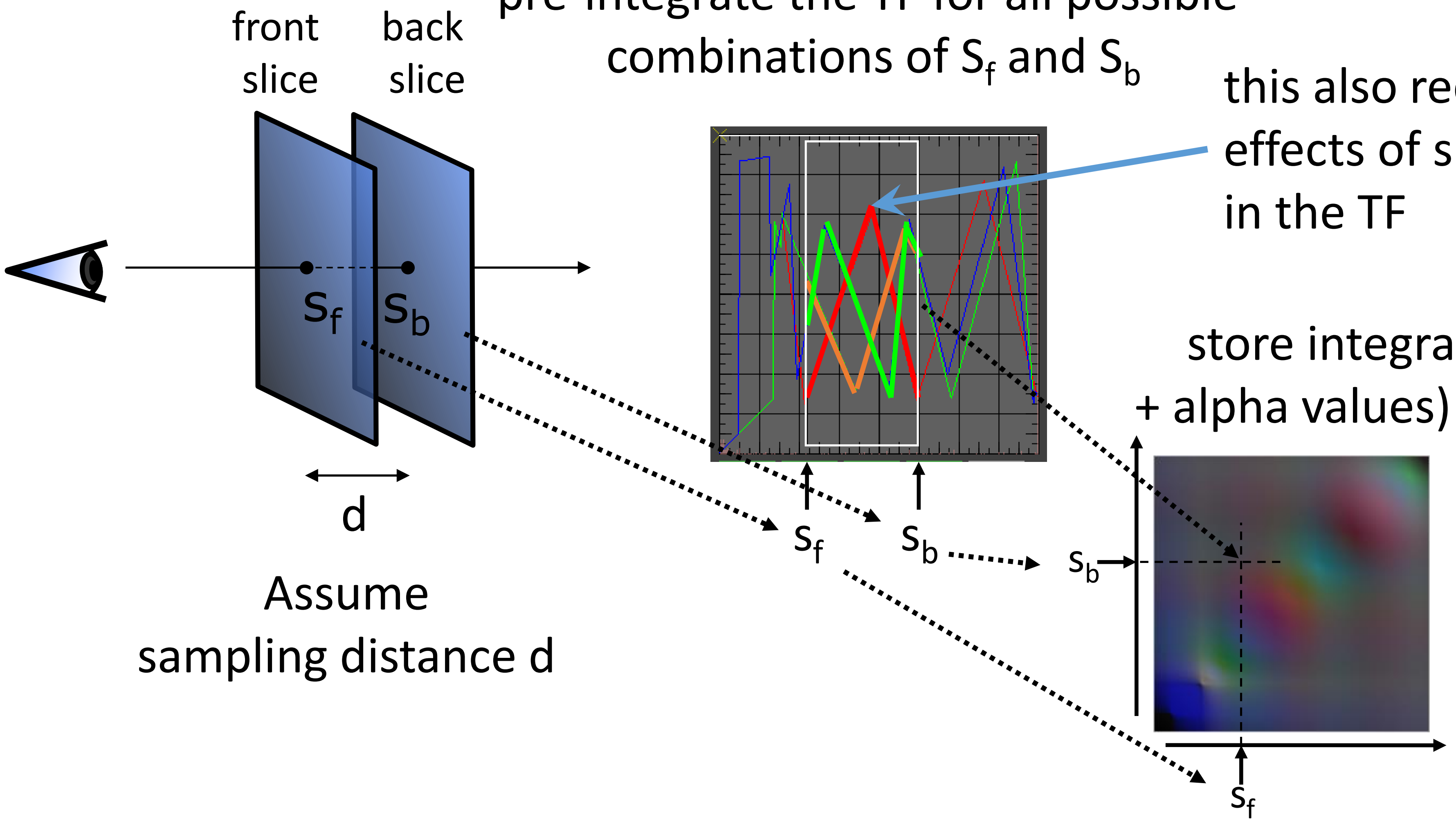


Pre-integration

instead of only evaluating the TF at S_f and S_b at runtime, pre-integrate the TF for all possible combinations of S_f and S_b

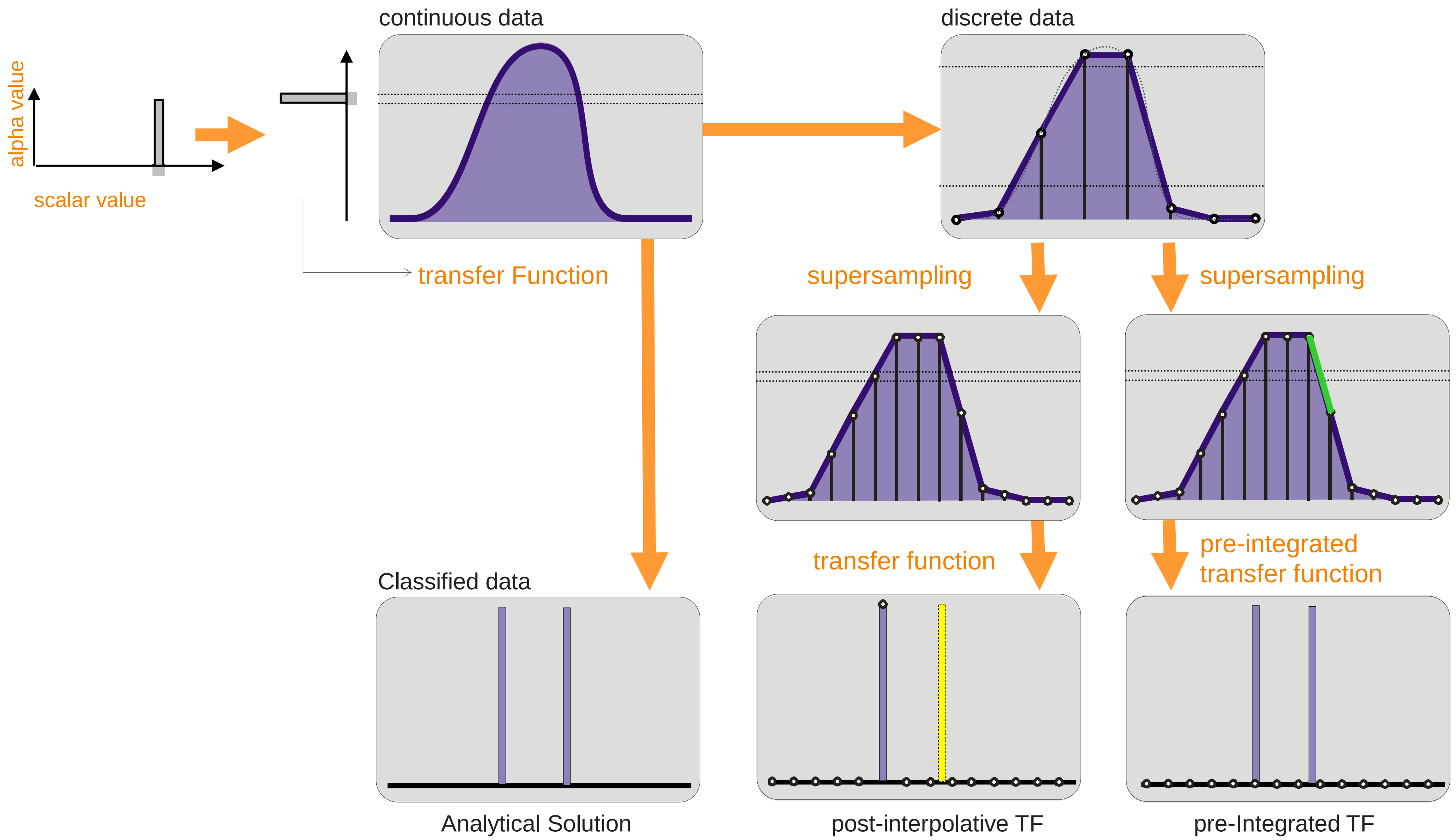
this also records the effects of small peaks in the TF

store integral (color + alpha values) into table

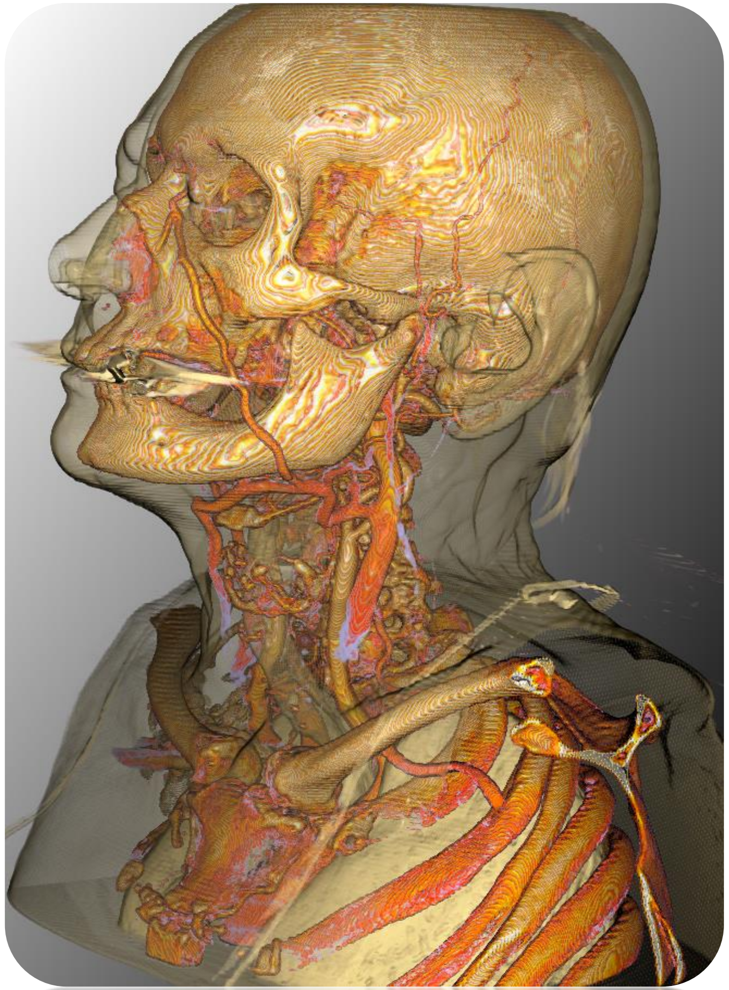


Assume sampling distance d

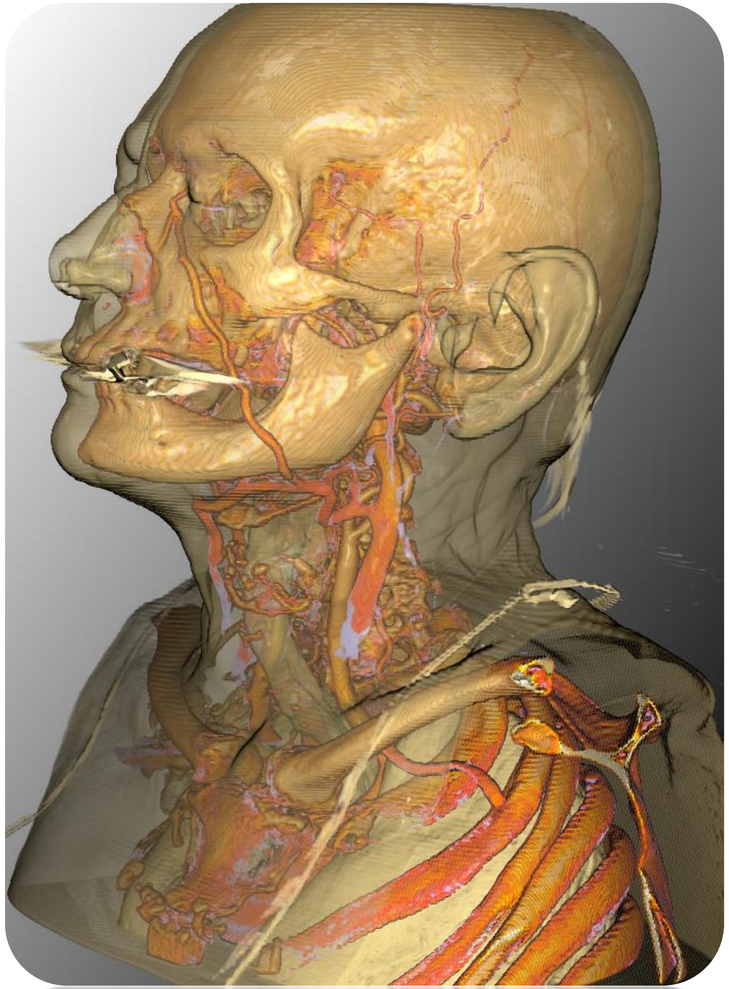
Pre-integration: Comparison



Pre-integration

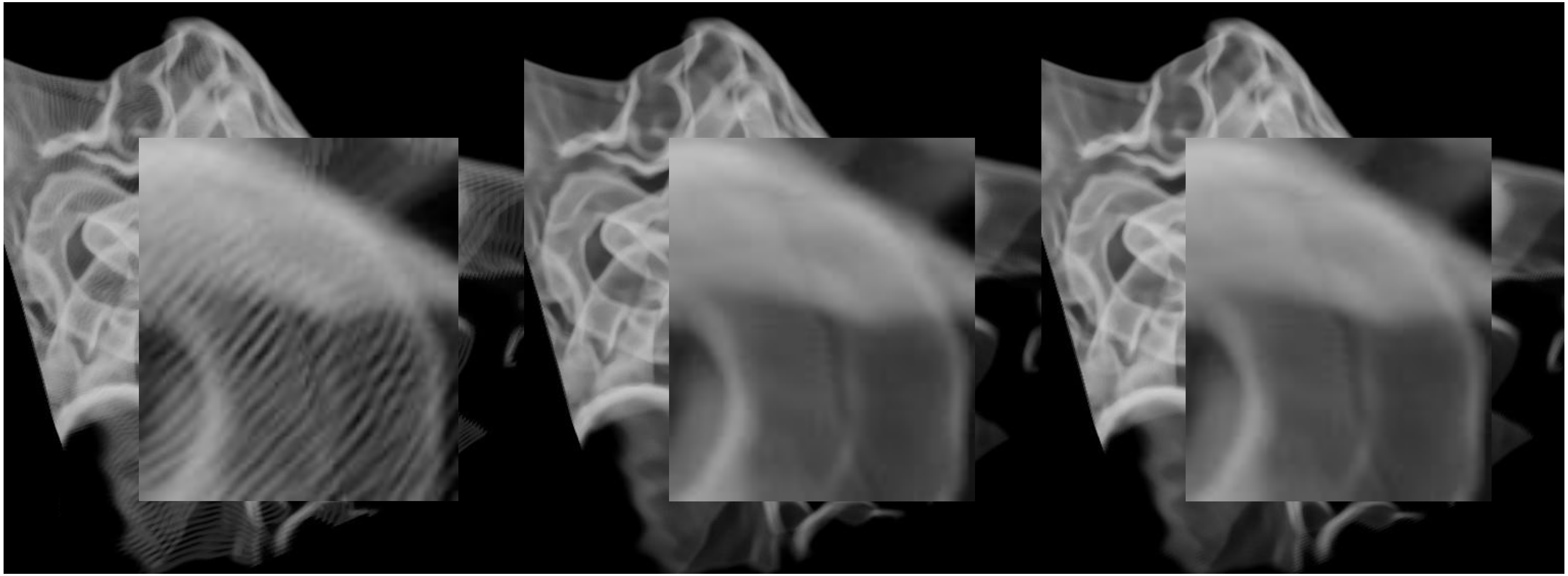


no pre-integration



pre-integration

Pre-integration



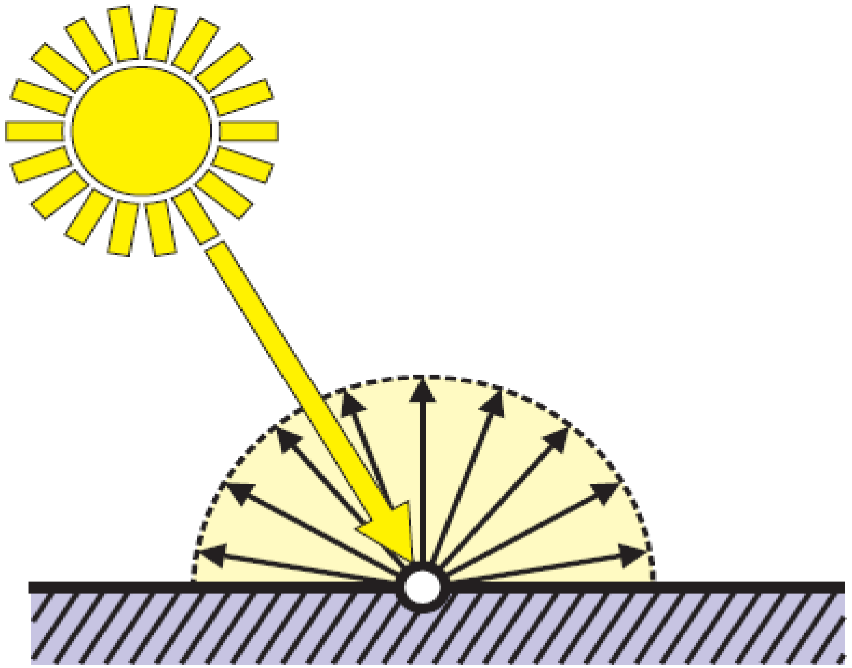
128 slices | 284 slices | 128 slabs

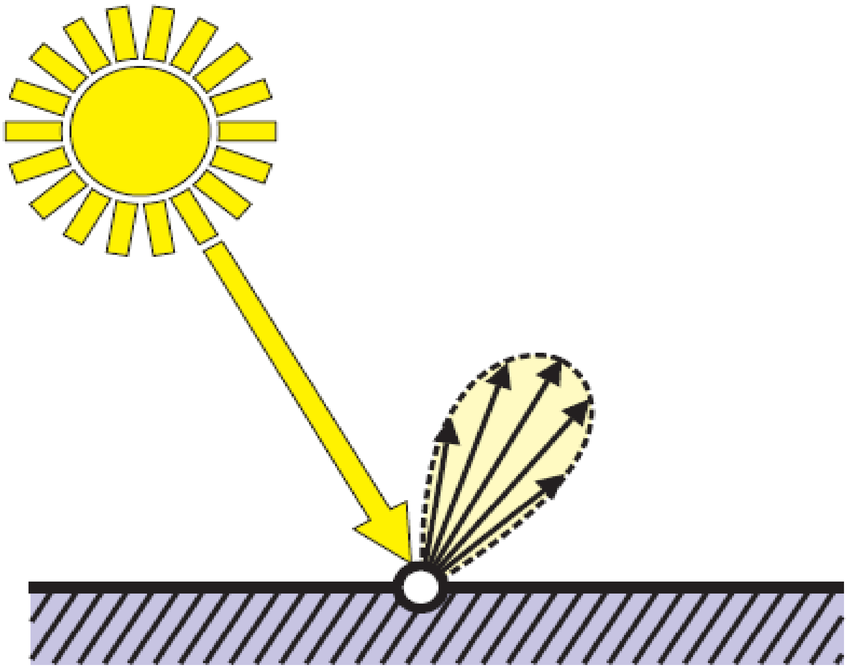
Shading

- make structures in volume data sets more realistic by applying an illumination model
  - shade each sample in the volume like a surface
  - any model used in real-time surface graphics suitable
  - common choice: Blinn-Phong illumination model

Shading

- local illumination, similar to surface lighting
  - **diffuse/Lambertian reflection**
    light is reflected equally in all directions
  - **specular reflection**
    light is reflected scattered around the direction of perfect reflection

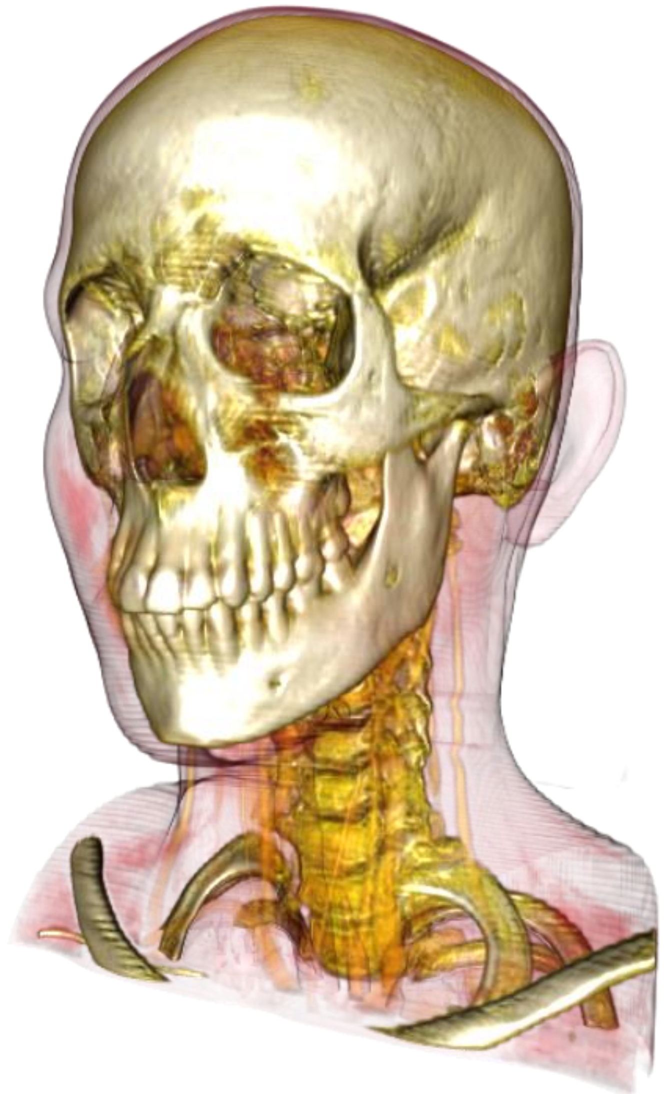




Shading

unshaded volume rendering

shaded volume rendering



Gradient estimation

- normalized gradient vector of the scalar field used to **substitute for the surface normal**
- gradient vector: the first-order derivative of the scalar field

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f(\mathbf{x})}{\partial x} \\ \frac{\partial f(\mathbf{x})}{\partial y} \\ \frac{\partial f(\mathbf{x})}{\partial z} \end{pmatrix}$$

partial derivative **in x-direction**

partial derivative **in y-direction**

partial derivative **in z-direction**

Gradient estimation

- estimate the gradient vector using finite differencing schemes, e.g. central differences:

$$\nabla f(x,y,z) \approx \frac{1}{2h} \begin{pmatrix} f(x+h,y,z) - f(x-h,y,z) \\ f(x,y+h,z) - f(x,y-h,z) \\ f(x,y,z+h) - f(x,y,z-h) \end{pmatrix}$$

- noisy data may require more complex estimation schemes

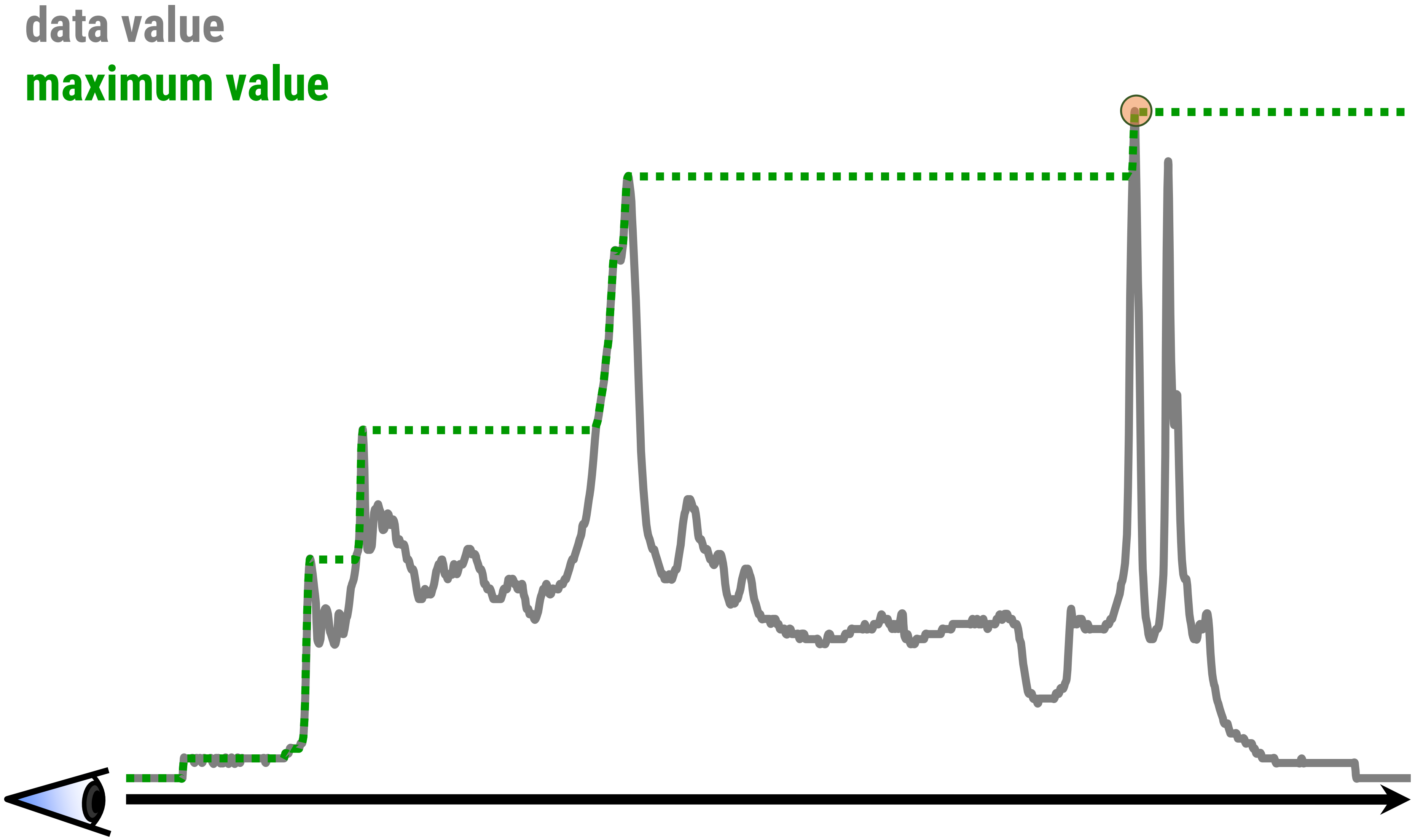
Gradient magnitude

- magnitude of gradient vector can be used to measure the “surfaceness” of a point
  - strong changes → high gradient magnitude
  - homogenity → low gradient magnitude
- applications
  - use gradient magnitude to modulate opacity of sample
  - interpolate between unshaded and shaded sample color using gradient magnitude as weight

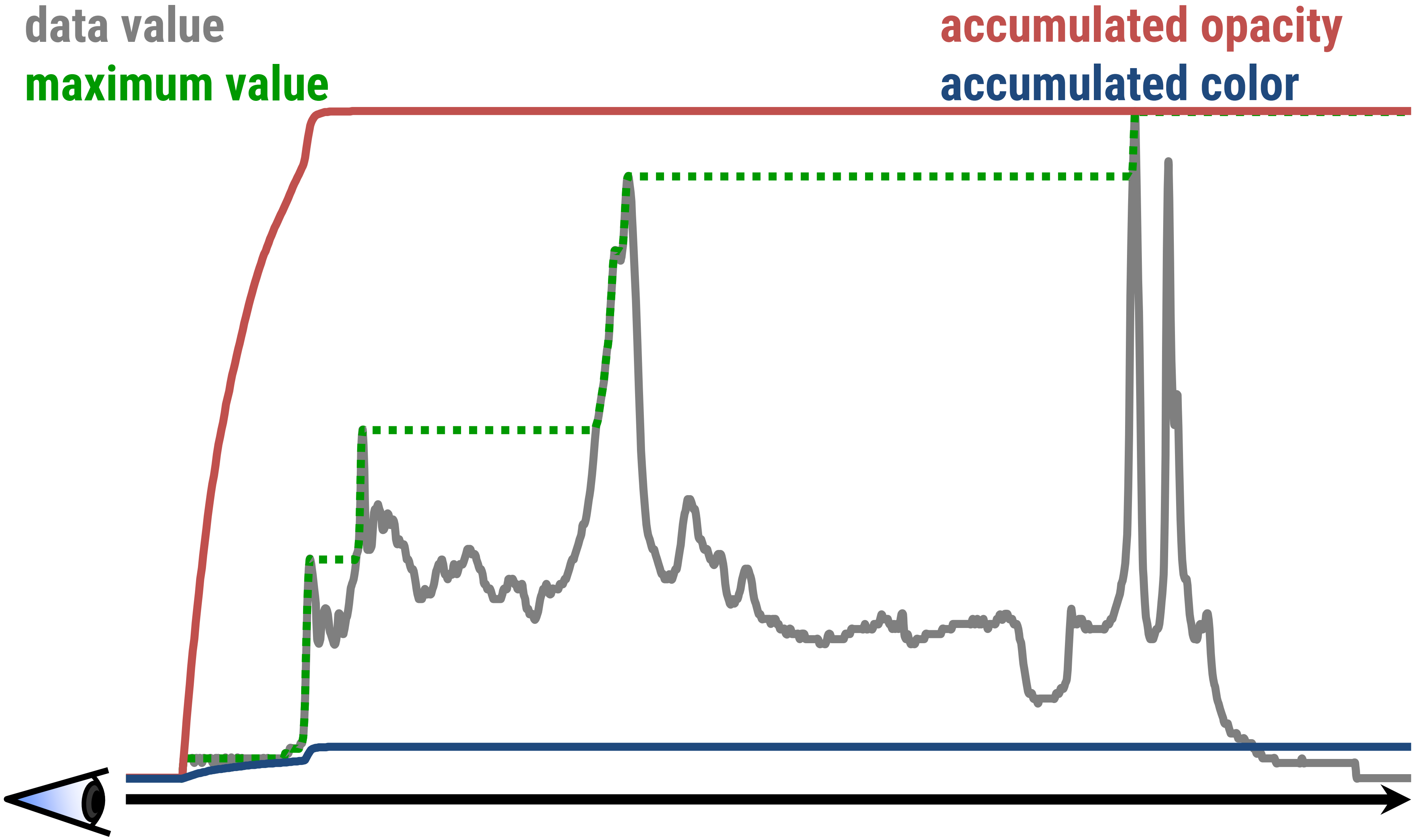
Compositing

- alpha compositing
  - physically-based
  - optical model for emission & absorption
  - may require complex transfer function
  - visual cues due to accumulation and shading
- maximum intensity projection
  - practically-motivated
  - project maximum value along each viewing ray
  - suffices with window/level setting
  - spatial ambiguities caused by order-independency

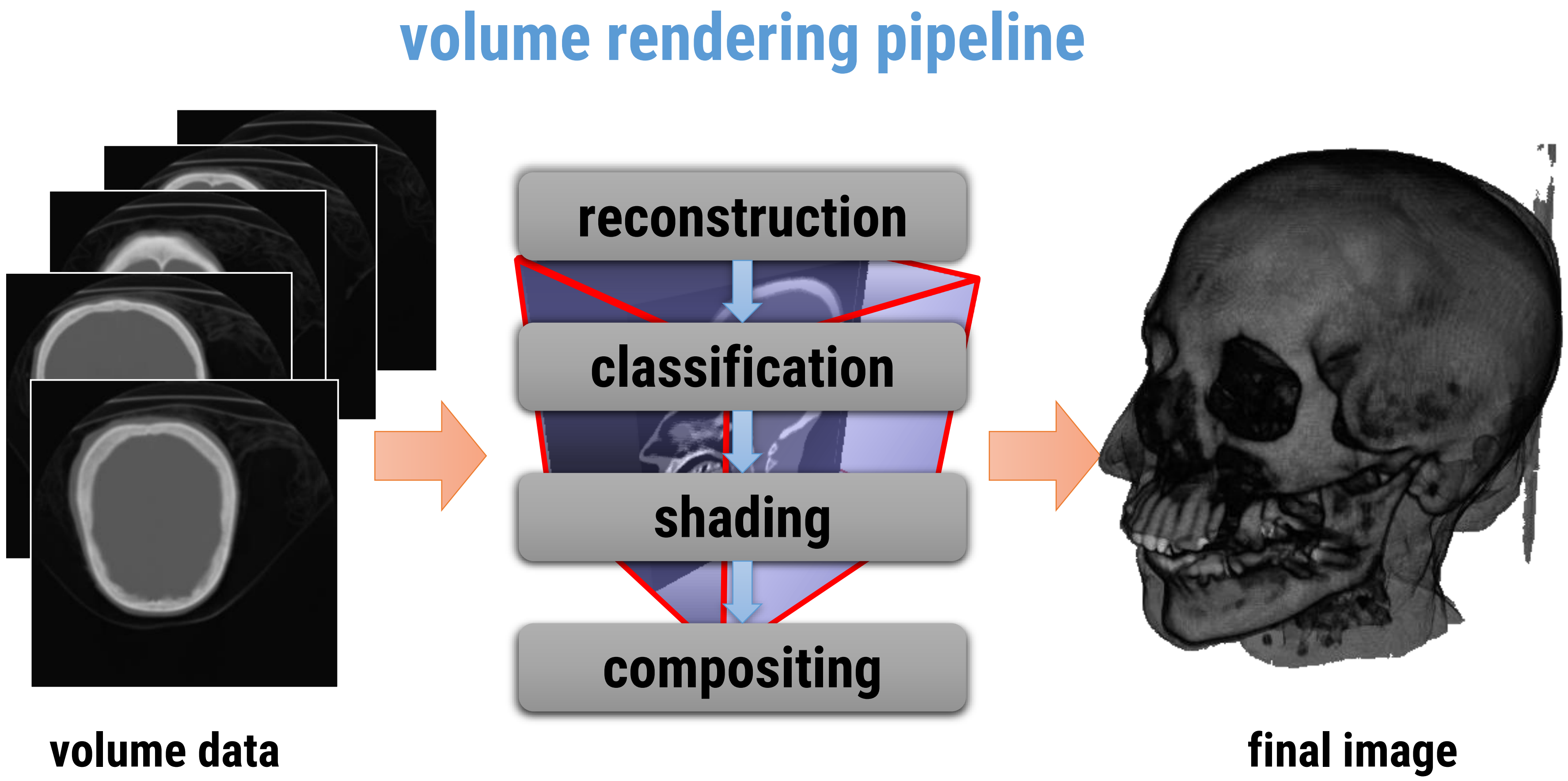
Maximum intensity projection



Alpha compositing



Volume rendering pipeline

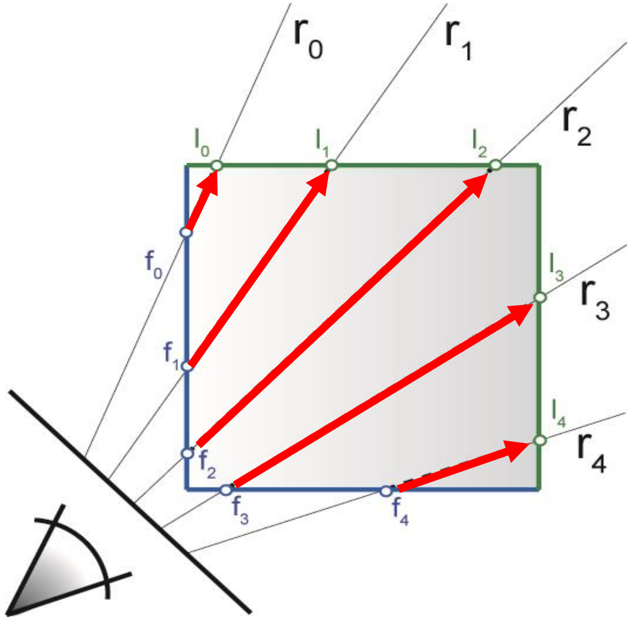


Implementation Considerations

Basic ray setup & termination

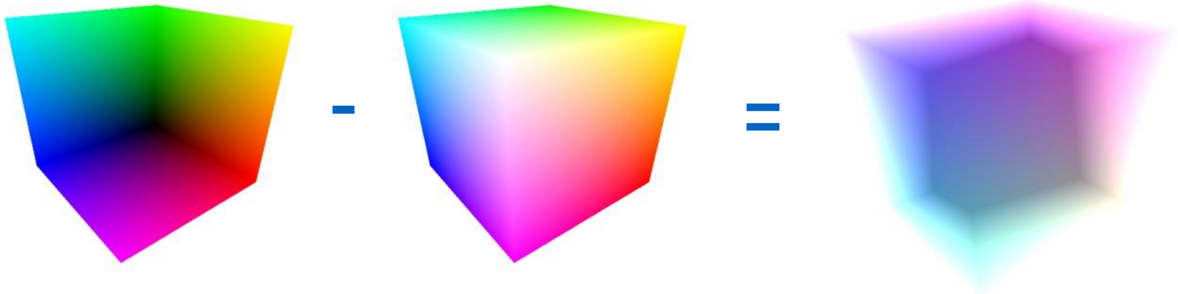
- two main approaches:
  - procedural ray/box intersection [Röttger et al., 2003], [Green, 2004]
  - rasterize bounding box [Krüger and Westermann, 2003]

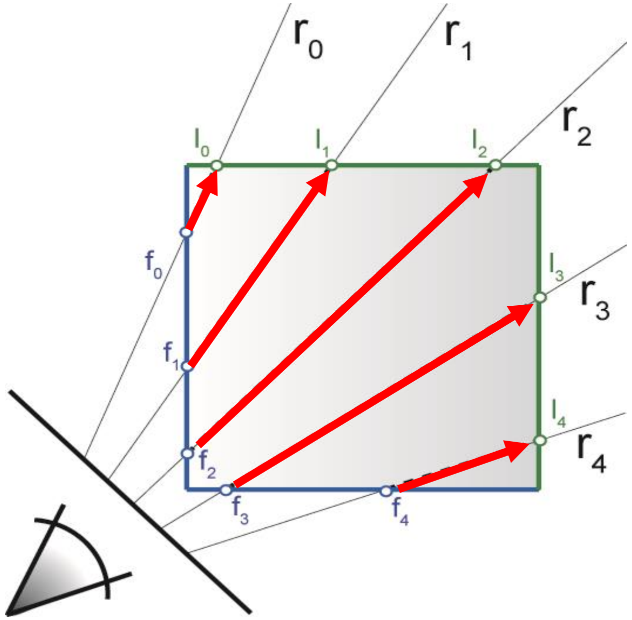
- some possibilities
  - ray start position and exit check
  - ray start position and exit position
  - ray start position and direction vector



Rasterization-based ray setup

- fragment = ray
- need ray start position, direction vector
- rasterize bounding box of volume data

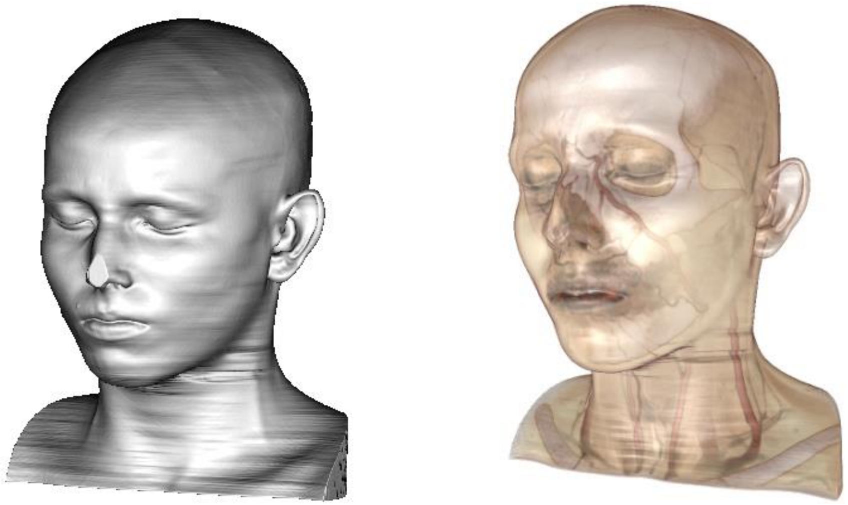




- identical for orthogonal and perspective projection!

Ray casting optimizations

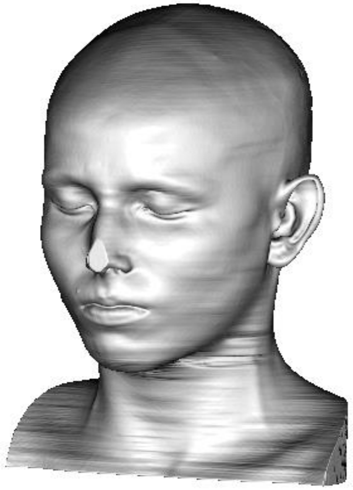
- early ray termination
  - isosurfaces: stop when surface hit
  - direct volume rendering:
    stop when opacity >= threshold



- several possibilities
  - older GPUs: multi-pass rendering with early-z-test
  - shader model 3+: break out of ray-casting loop
  - current GPUs: early loop exit works well

Ray casting optimizations

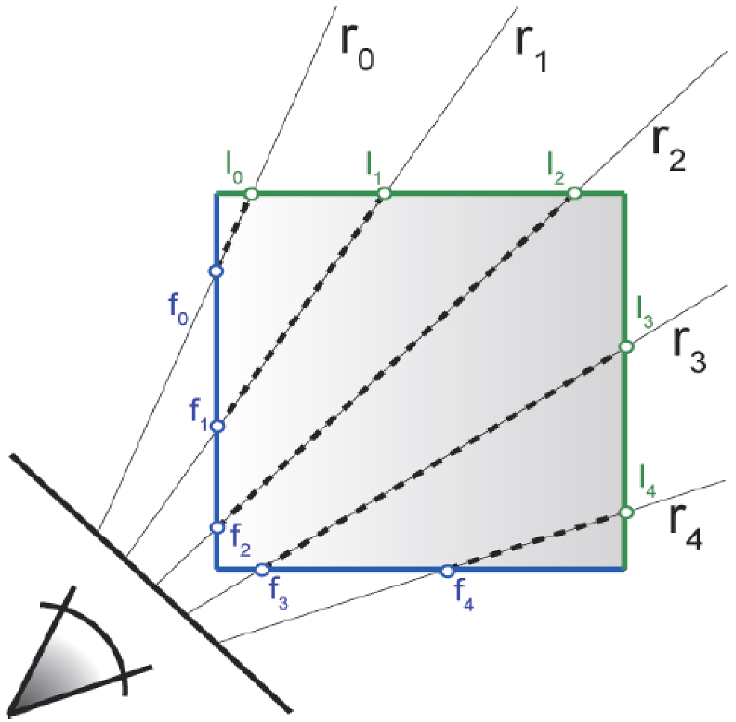
- empty space skipping
  - skip transparent samples
  - depends on transfer function
  - start casting close to first hit



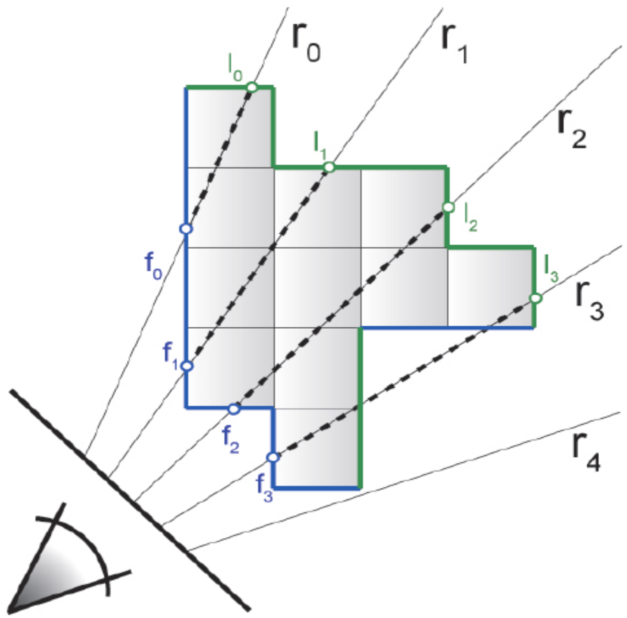
- several possibilities
  - per-sample check of opacity (expensive)
  - traverse hierarchy (e.g., octree) or regular grid
  - use tighter bounding geometry

Empty space skipping

- modify initial rasterization step



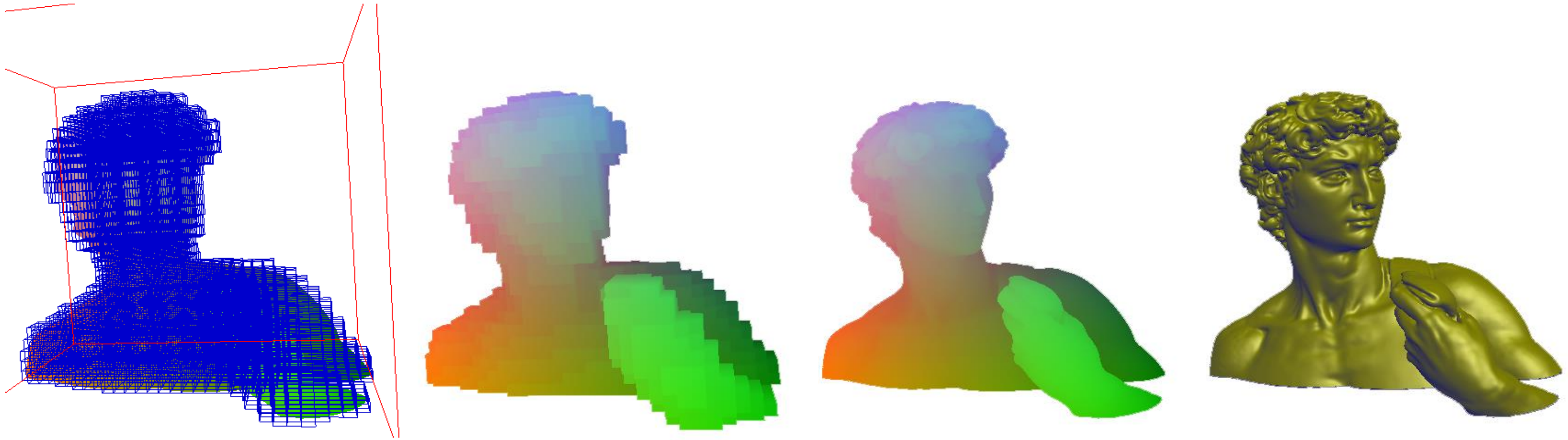
rasterize bounding box



rasterize “tight" bounding geometry

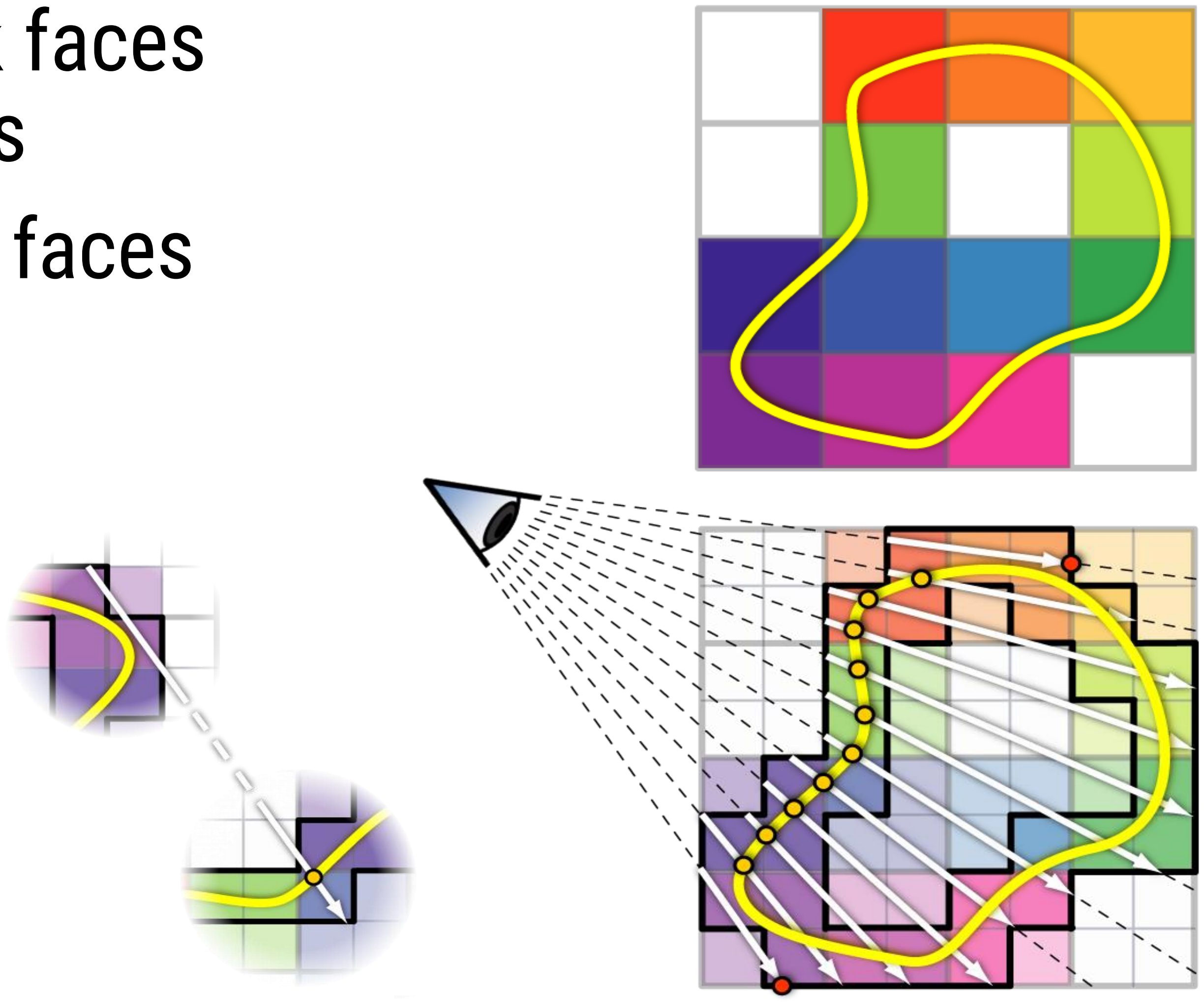
Empty space skipping

- store min-max scalar data values of volume blocks
- cull blocks against transfer function
- rasterize front and back faces of active blocks



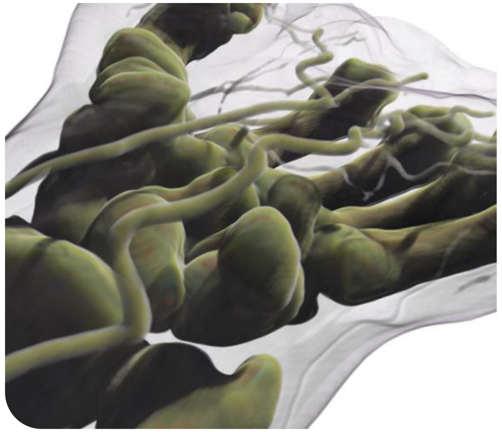
Empty space skipping

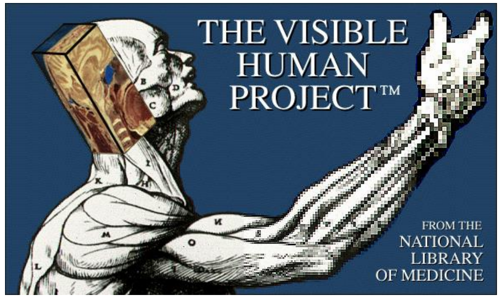
- rasterize front and back faces of active min-max bricks
- start rays on brick front faces
- terminate when
  - full opacity reached, or
  - back face reached
- not all empty space is skipped



Additional techniques/considerations

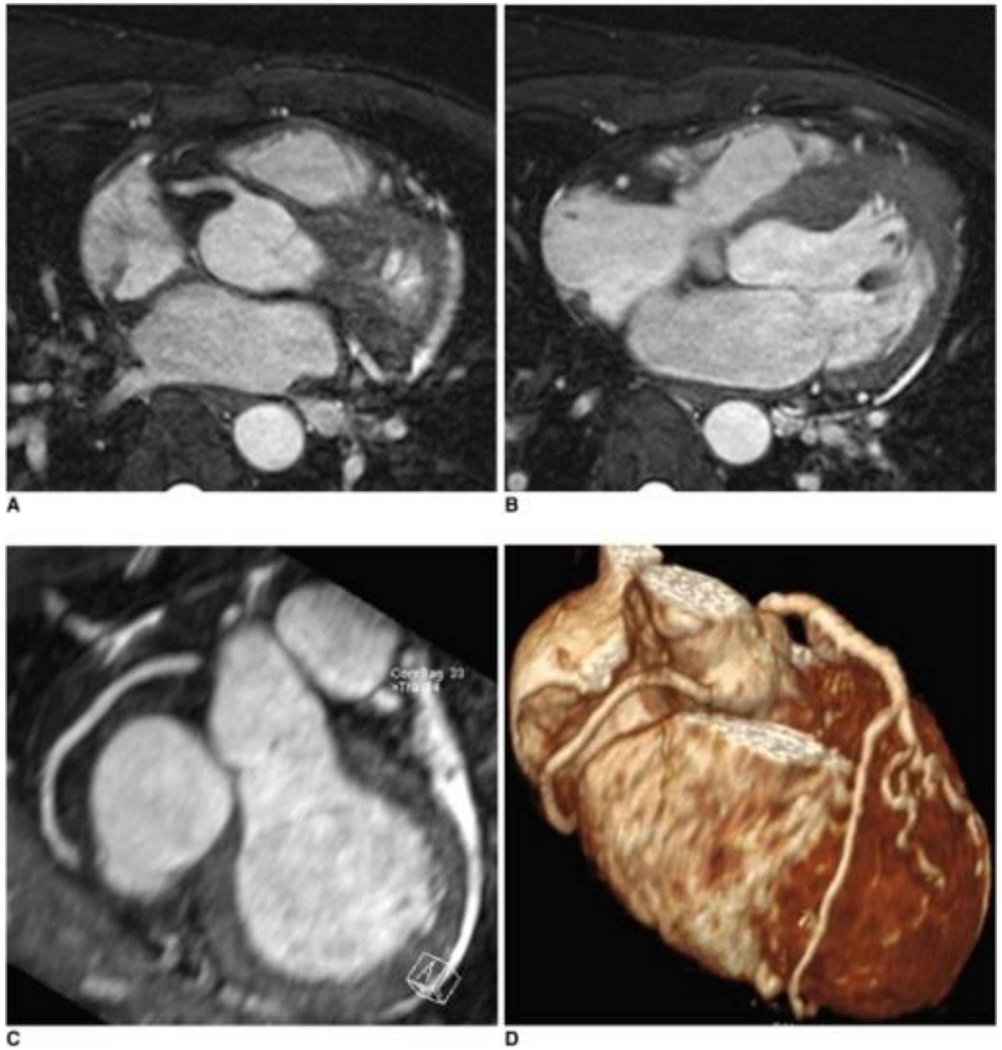
- global illumination effects
  - in- and out-scattering (which we previously ignored)
  - shadows
  - photon mapping
  - …

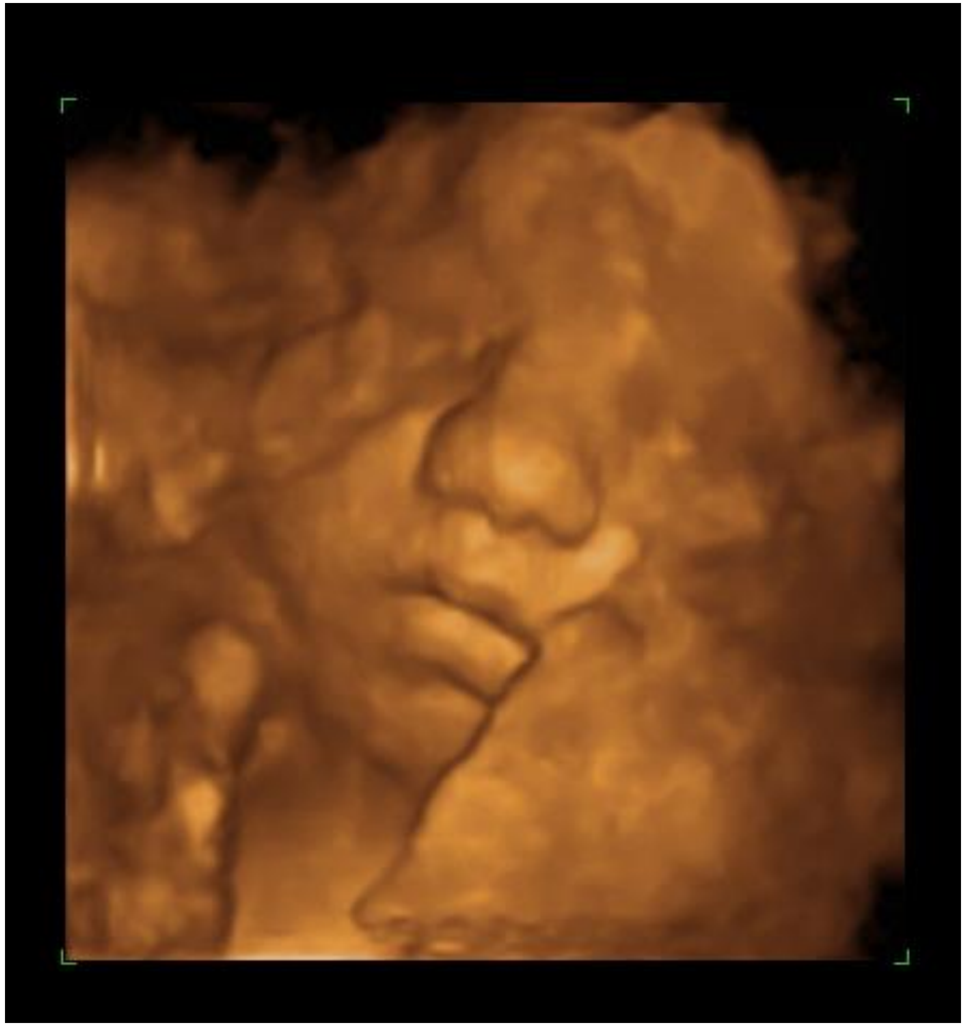




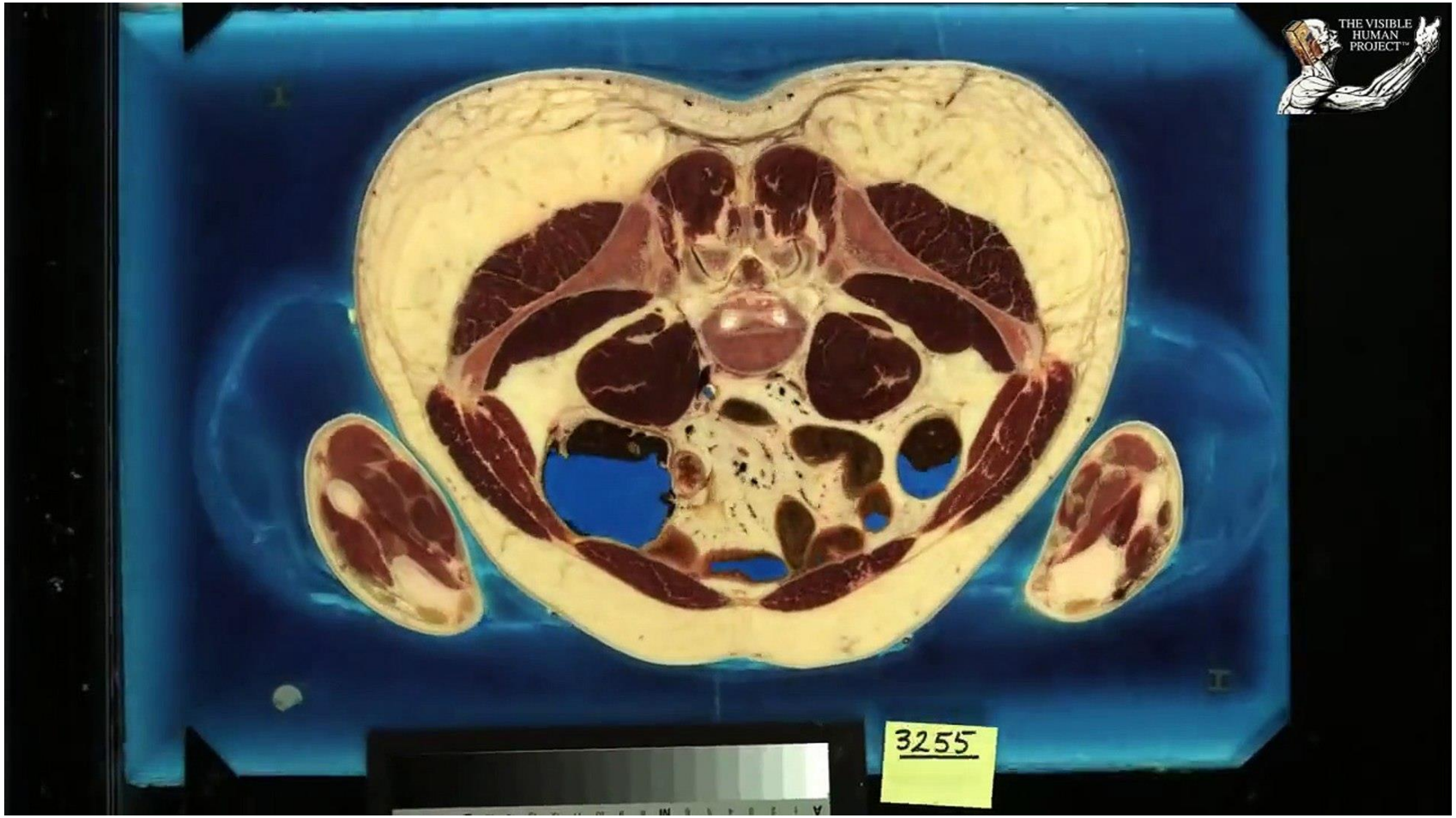
Applications

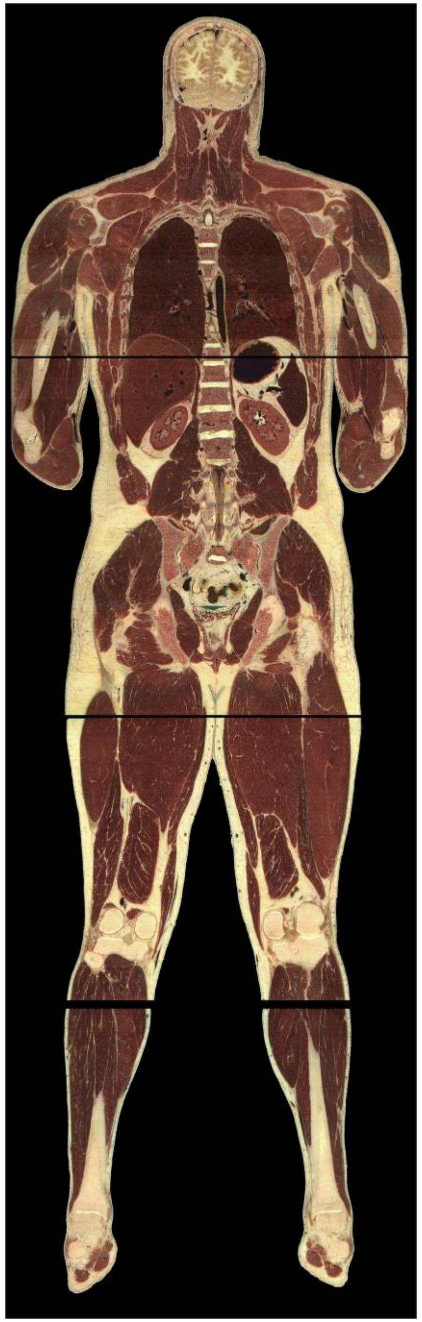
Medical applications



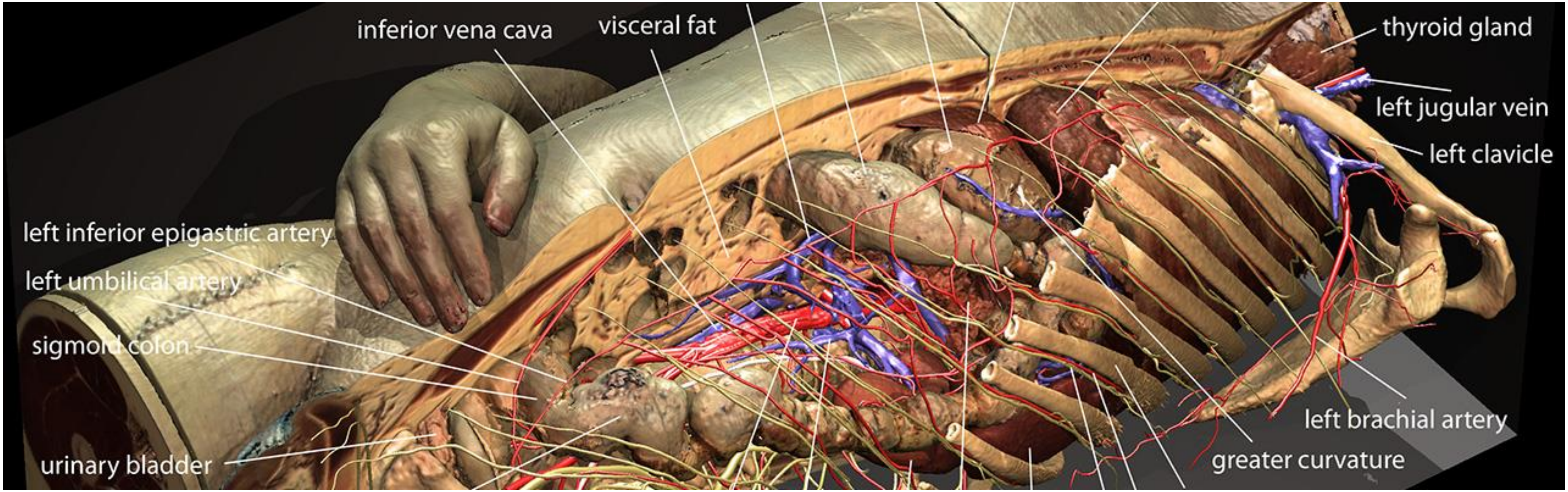


Medical + graphics research

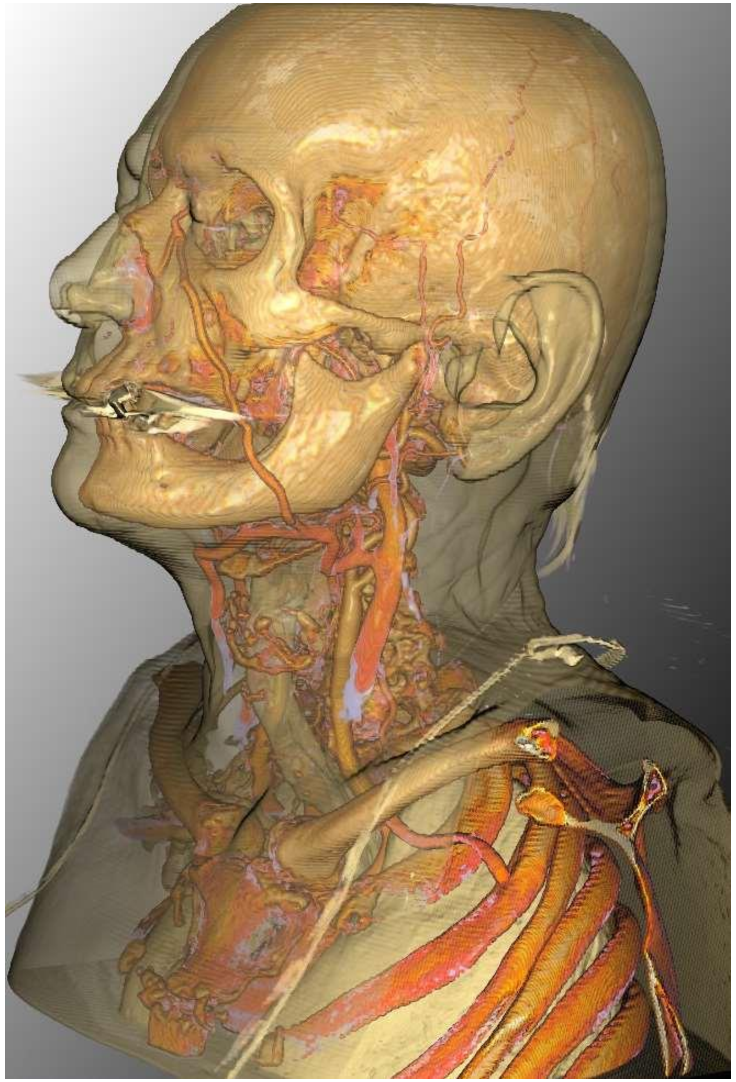


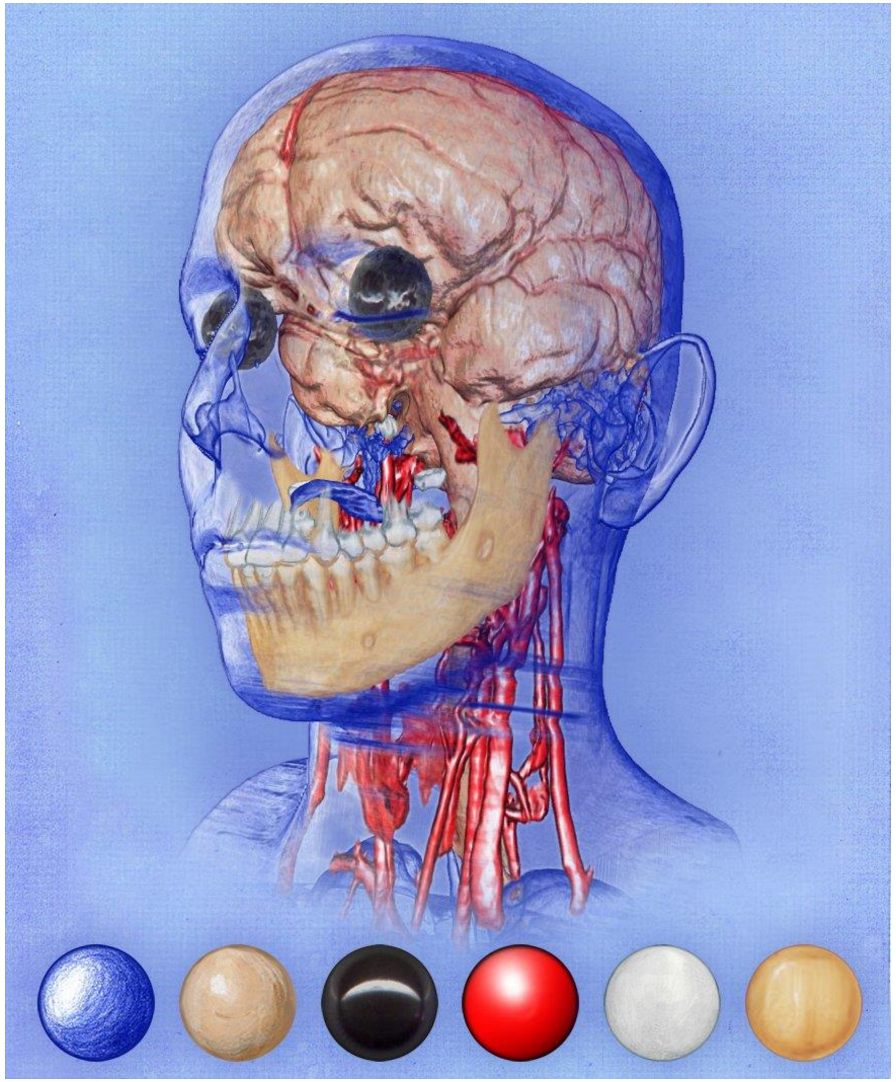


Medical + graphics research

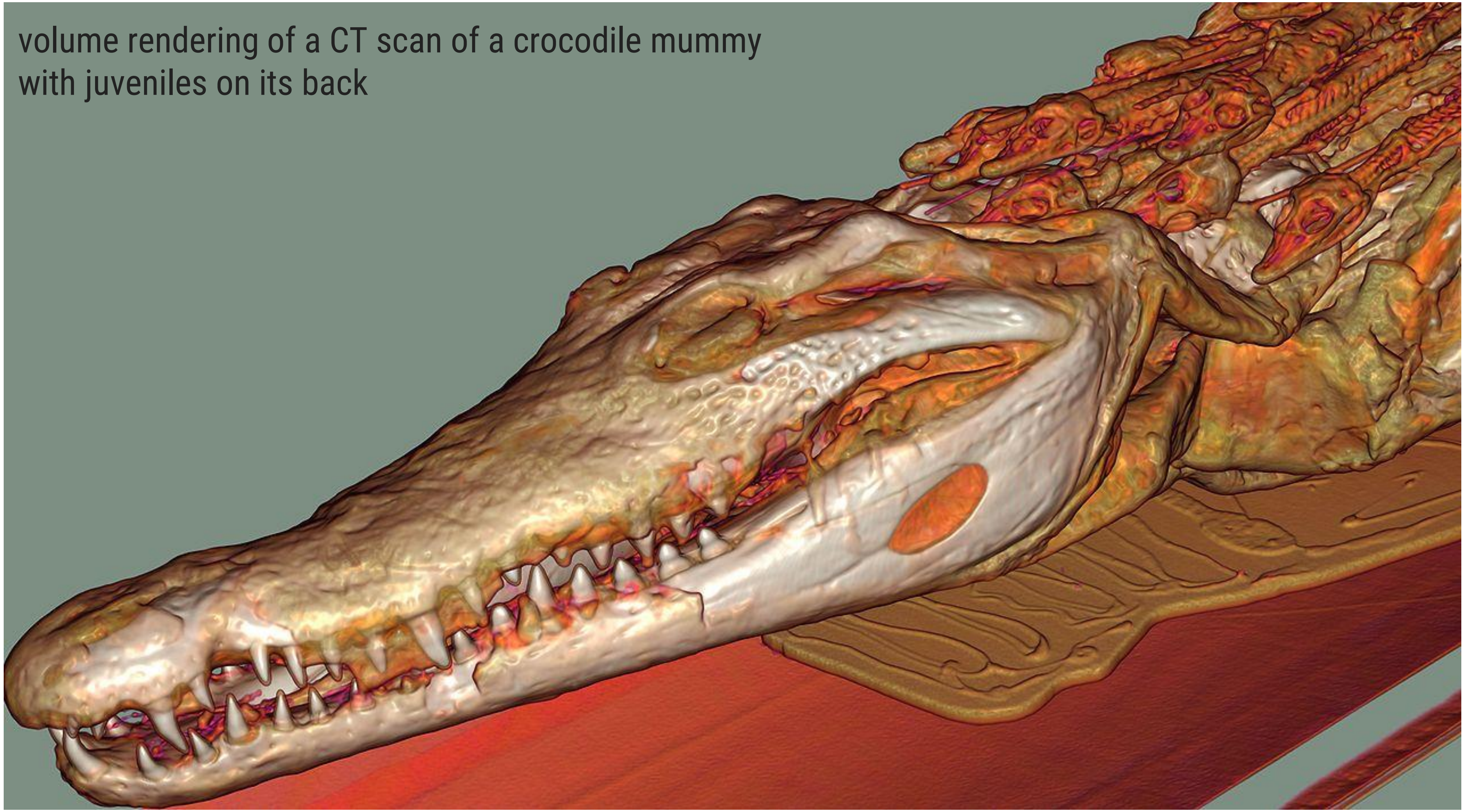


Anatomy visualization and illustration



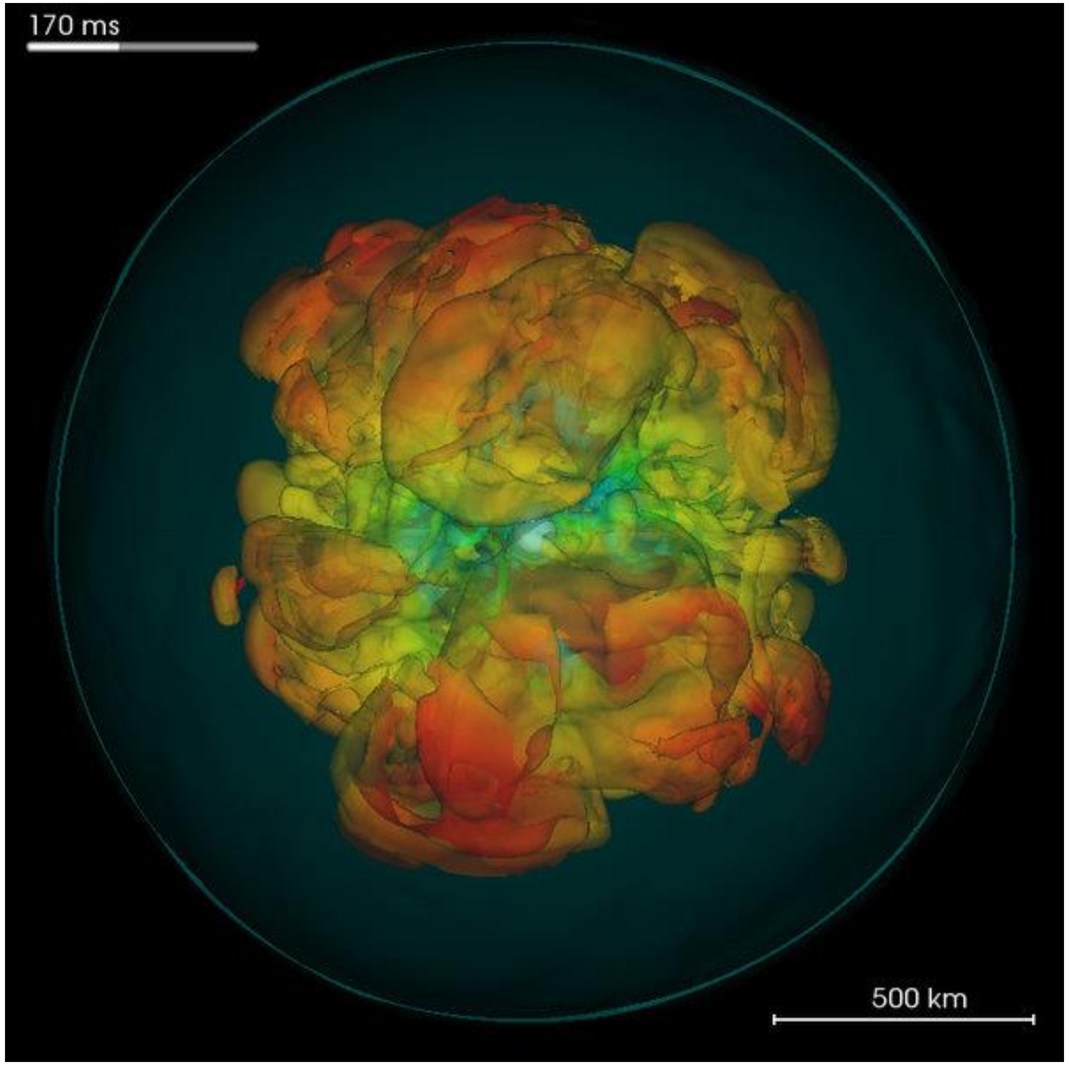


Applications in archaeology



volume rendering of a CT scan of a crocodile mummy
with juveniles on its back

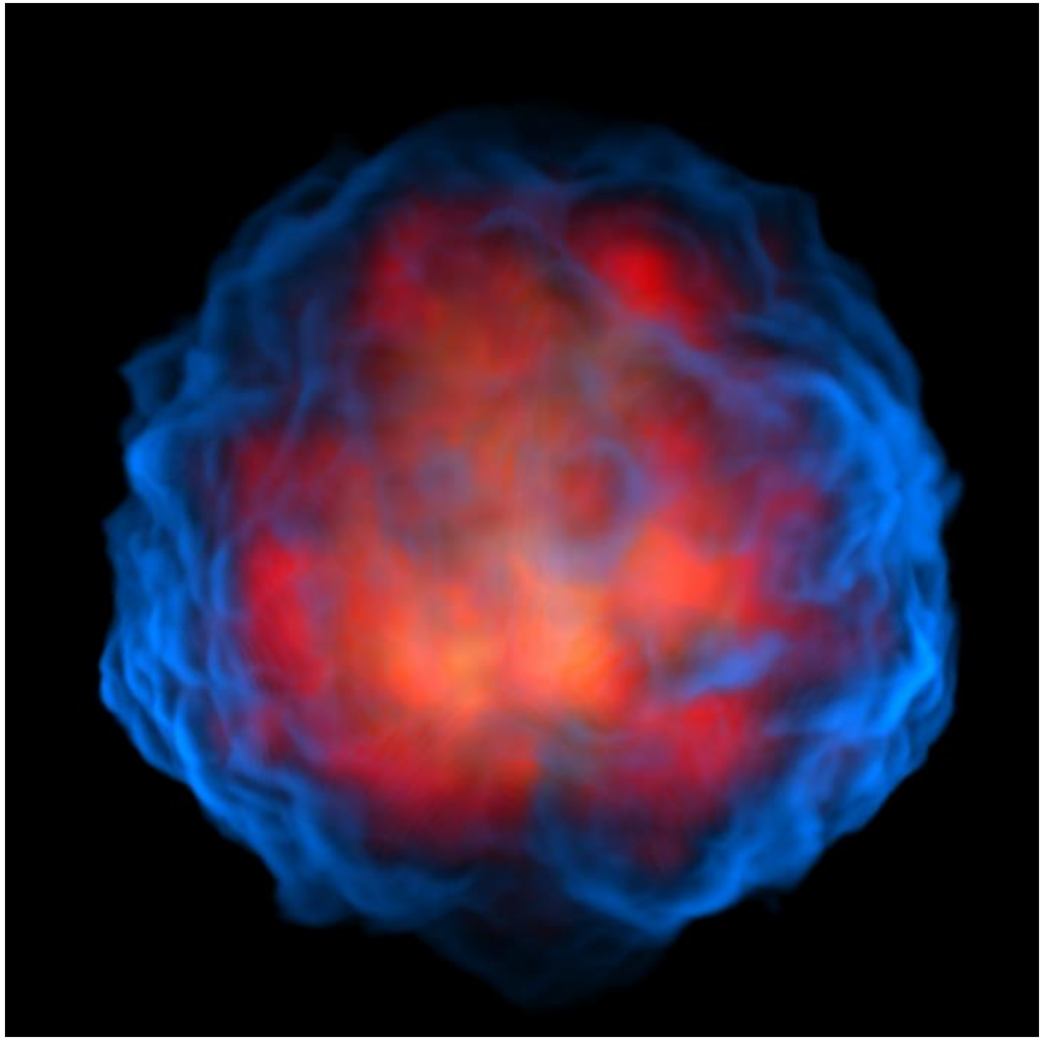
Applications for physical simulations



explosion in the stellar interior
of a 9.6 solar-mass star
170 ms after core bounce

Volume rendering in games

- volume rendering for dedicated objects such as procedural flames and fire balls



Volume rendering in games

Volume rendering in games