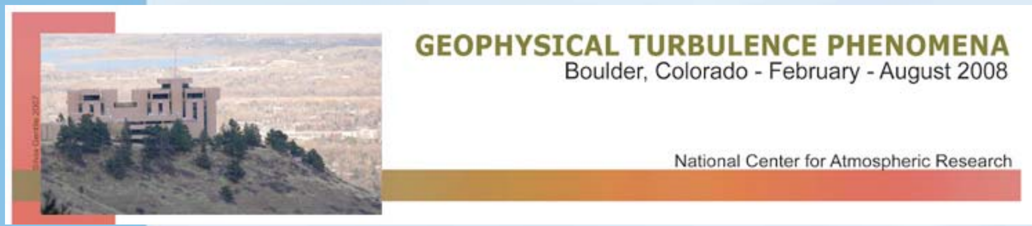


NCAR

Scalar Data Visualization for Volumetric (3D) Fields

John Clyne
clyne@ucar.edu
National Center for Atmospheric Research

Goals

NCAR

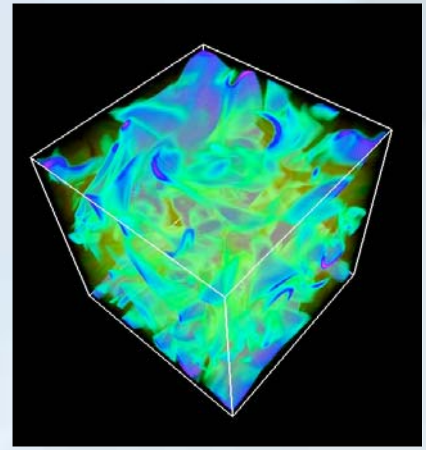
- Provide you enough information about Direct Volume Rendering so that you can:
  1. Create “useful” volume visualizations of your data
  2. Interpret what you are seeing
  3. Avoid rendering artifacts when possible
  4. Recognize and understand then when you can’t

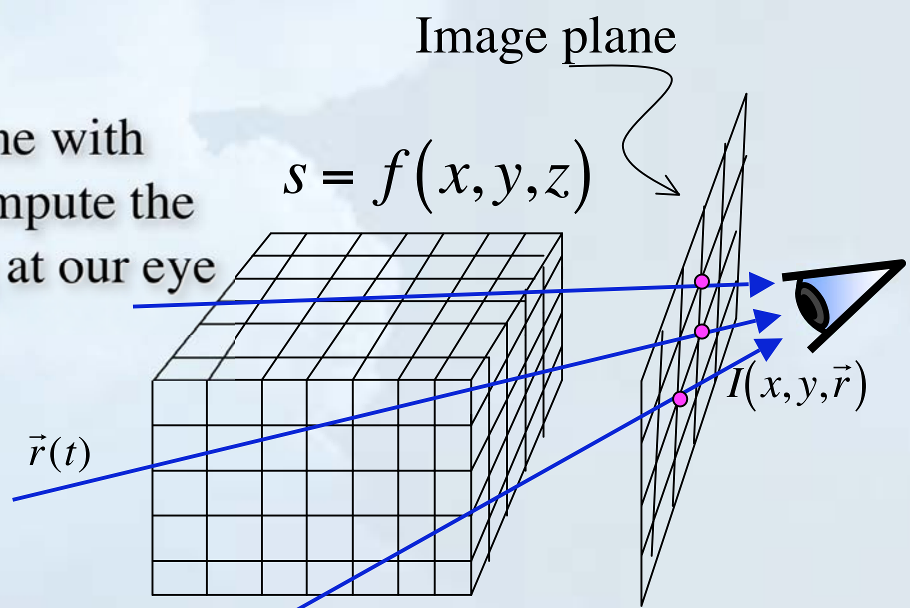
Outline

- Theoretical foundations
- Implementation
- Understanding and avoiding sources or error

Direct Volume Rendering

- Assumptions
  - Discretely sampled continuous scalar 3D field:
    $$s = f(x,y,z);\ \text{where } x,y,z \in \mathrm{Z}$$
  - The field f is a gaseous *participating medium* that interacts with light:
    - Scattering
    - Emitting
    - Absorbing
  - For each pixel in an image plane with coordinates *x,y*, we wish to compute the light intensity, *I(x,y,r)* arriving at our eye along a ray *r*





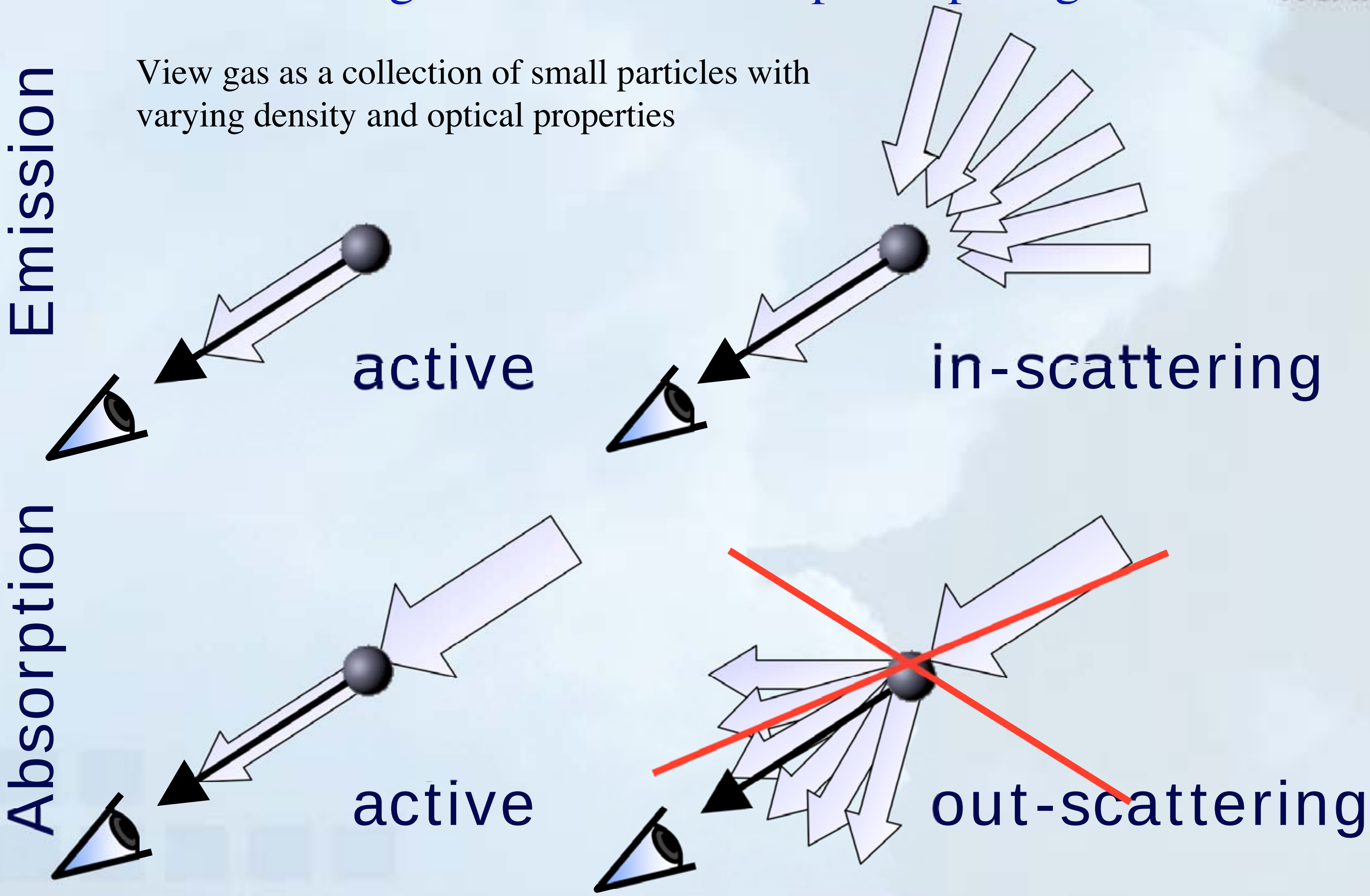
Basic idea: model light transport in gaseous materials

Photorealistic rendering vs scientific visualization of volumetric data

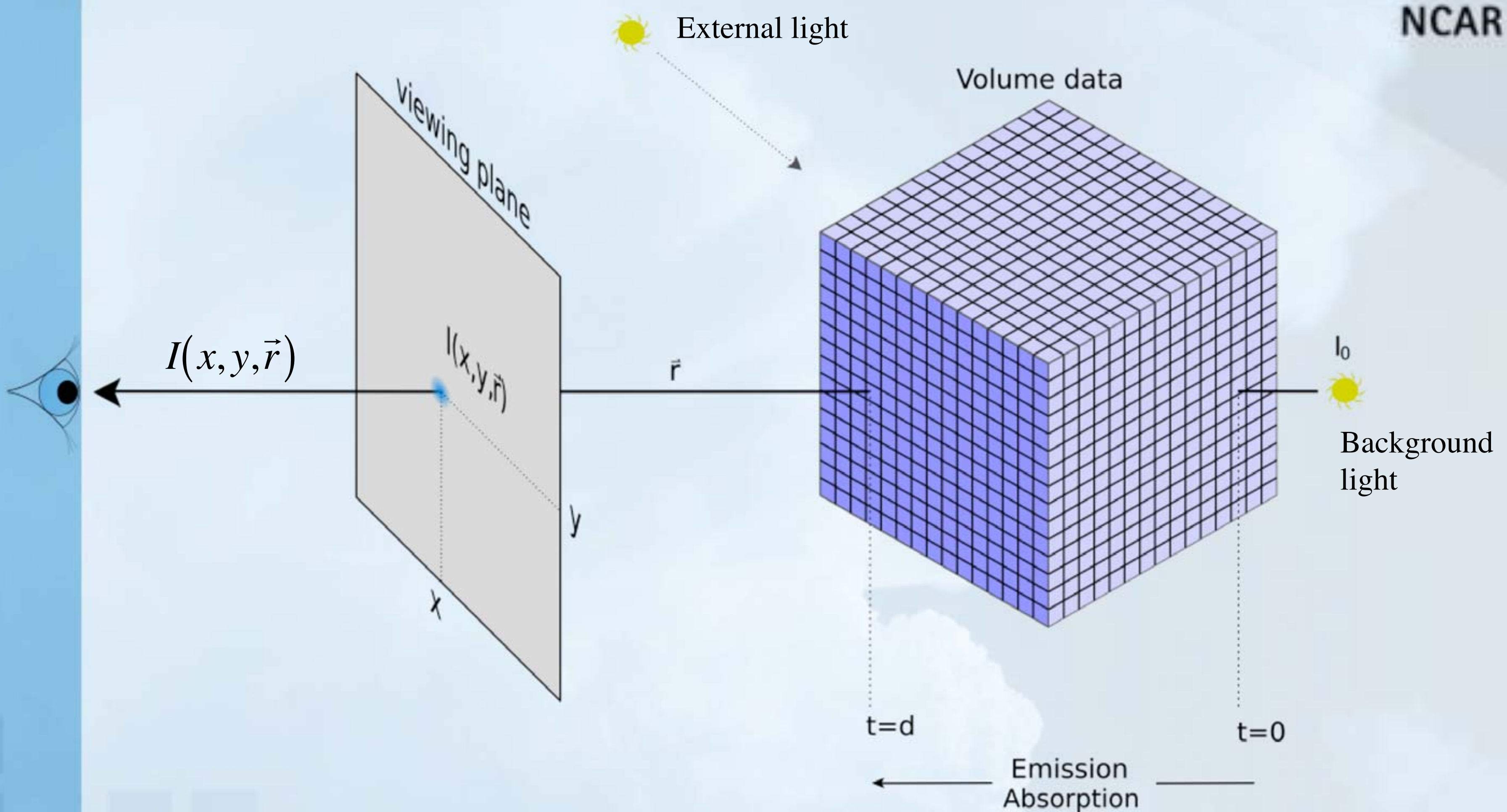
- Photorealism
  - Imitates the look of realistic gases
  - Requires physically accurate description of the participating medium
- Scientific visualization
  - Visually extracting information from a 3D scalar field
  - Requires mapping the field to physical quantities that describe light interaction at a point in 3D space

Optical Models
Our view of how light interacts with a participating medium

View gas as a collection of small particles with varying density and optical properties



Direct Volume Rendering



Volume Rendering Integral (1)

emission-absorption optical model (Hege 1993, Glassner 1995)

$$I(D) = \overbrace{I_0 e^{-\int_{s_0}^{D} k(t)\,dt}}^{A} + \overbrace{\int_{s_0}^{D} q(s) e^{-\int_{s}^{D} k(t)\,dt}\, ds}^{B}$$

No scattering

A : light from background attenuated by volume

B : integral contribution of source terms attenuated by the particpating medium along the remaining distance to the eye

$\vec{r}(s): s_0 \le s \le D,$ where s_0 is the ray start, D is ray end

I_0 : the light entering the volume from background at $s = s_0$

$I(D)$: radiance leaving the volume at $s = D$

$k(t)$: the absorption (enegy loss) of light at $\vec{r}(t)$

$q(s)$: the emmision (e.g., from thermal excitation) of light at $\vec{r}(s)$

Volume Rendering Integral (2)
(emission-absorption optical model)

NCAR

$$I(D) = \overbrace{I_0 e^{-\int_{s_0}^{D} k(t)\,dt}}^{A} + \overbrace{\int_{s_0}^{D} q(s) e^{-\int_{s}^{D} k(t)\,dt}\, ds}^{B}$$

No scattering

$$\tau(s_1, s_2) = \int_{s_1}^{s_2} k(t)\,dt$$

Optical depth between s_1 and s_2: distance light may travel before being absorbed. Small values indicate more transparent material, while large values correspond to more opaque media

$$T(s_1, s_2) = e^{-\tau(s_1, s_2)} = e^{-\int_{s_1}^{s_2} k(t)\,dt}$$

Transparency between s_1 and s_2

$$\boxed{I(D) = I_0 T(s_0, D) + \int_{s_0}^{D} q(s) T(s, D)\, ds}$$

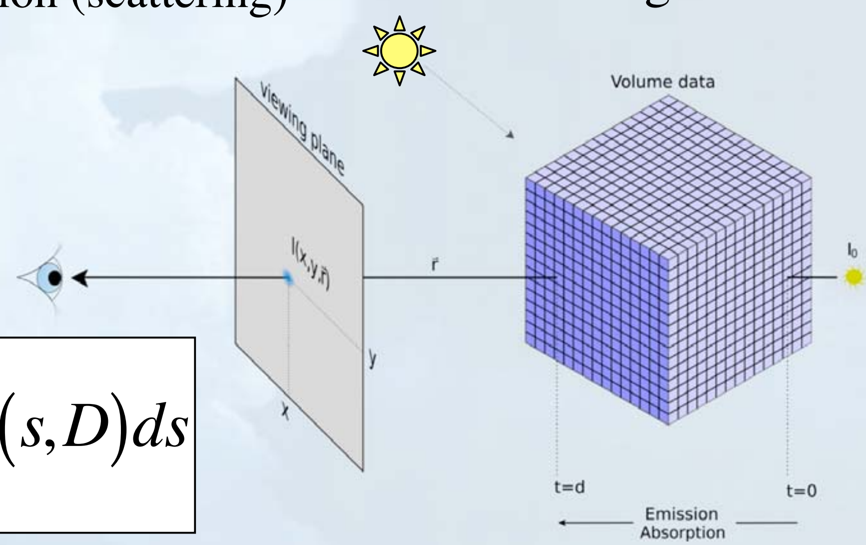
Including scattering in the volume rendering integral (emission-absorption-scattering optical model)

- Scattering light that comes from an external light source (not from emission from the volume itself)

- Assume external light passes through volume unimpeded (without absorption or scattering), reaching all points in the volume with equal intensity

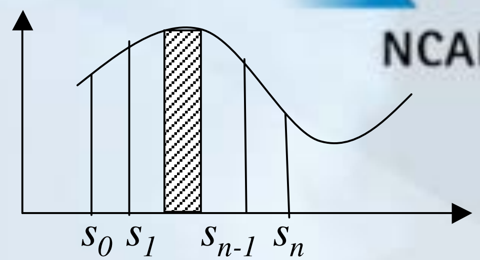
- We simply replace $q(s)$ with $q_E(s) + q_S(s)$, where
  - $q_E(s)$: light emitted a s
  - $q_S(s)$: additional light from local reflection (scattering)

External light source



$$\boxed{I(D) = I_0 T(s_0, D) + \int_{s_0}^{D} \left(q_S(s) + q_E(s)\right) T(s, D)\, ds}$$

Numerical approximation of volume rendering integral



Split integration domain into n subintervals, $s_0<s_1<\ldots<s_{n-1}<s_n$

$$I(s_i) = I(s_{i-1})T(s_{i-1},s_i) + \int_{s_{i-1}}^{s_i} q(s)T(s,s_i)\,ds$$ Light transport within $[s_{i-1},s_i]$

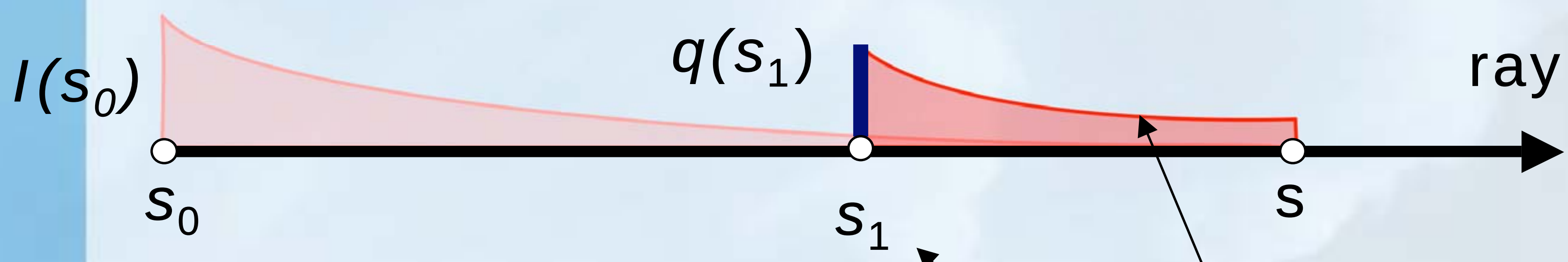
$$T_i = T(s_{i-1},s_i)$$ Transparency contribution of i^{th} interval

$$c_i = \int_{s_{i-1}}^{s_i} q(s)T(s,s_i)\,ds$$ Color (light) contribution of i^{th} interval

$$I(D) = I(s_n) = I(s_{n-1})T_n + c_n = \left(I(s_{n-2})T_{n-1} + c_{n-1}\right)T_n + c_n = \ldots,$$

$$I(D) = \sum_{i=0}^{n} c_i \prod_{j=i+1}^{n} T_j, \text{ where } c_0 = I(s_0)$$

Integration subintervals



Initial intensity (emission) at s_0

Emission at s_1

Absorption a long the distance s_1 - s

$$I(s_1) = I(s_0)T(s_0,s_1) + \int_{s_0}^{s_1} q(s)T(s,s_1)ds$$

$$I(s_1) = I(s_0)T_1 + c_0$$

Slide credit: Markus Hadwiger

Evaluation of transparency and color contributions in each subinterval

- Integration domain is now segmented into *n* discrete intervals
- Need to evaluate the color and transparency of each interval
- Approximate with a Riemann sum of *n* equidistant segments of length Δx

$T_i \approx e^{-k(s_i)\Delta x}$, transparency of ith segment

$c_i \approx q(s_i)\Delta x$, color contribution of ith segment

where

$\Delta x = (D - s_0)/n$

Outline

- Theoretical foundations
- Implementation
- Understanding and avoiding sources or error

Practical considerations for evaluating the volume integral

- Classification
- Reconstruction
- Illumination
- Alternatives to volume rendering integral
- Hardware implementation

Classification

How do we obtain the emission and absorption functions: *q(s)* and *k(s)*? Previous discussion assumes we have them. We don't!!

Classification: Mapping from data to opacities (absorption) and color (emission)

Range of interest: high opacity (more opaque)
not-so-interesting range: translucent or transparent

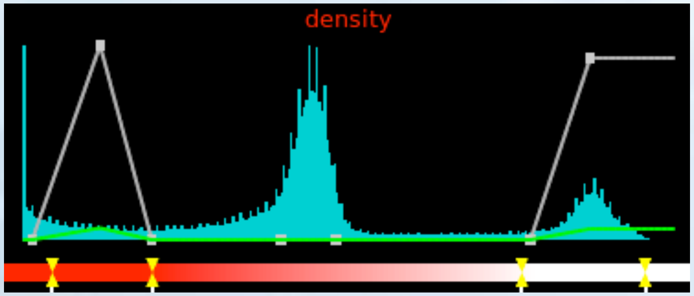
Classification is in practice performed via a *Transfer Function*

Classification
Transfer Functions

$T_f : R \rightarrow R^4$ One-dimensional -- maps scalar values to 4-tuples (r,g,b,α)

$q(s)$ $k(s)$

Transfer Functions make volume data visible by mapping data values to optical properties.

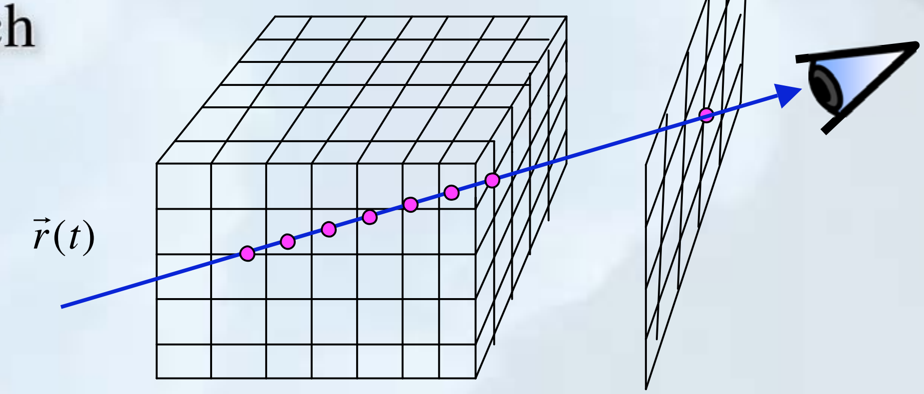


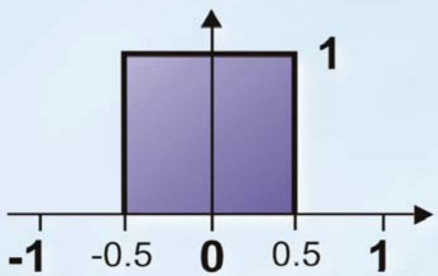
Slide credit: Kenny Gruchalla

Reconstruction

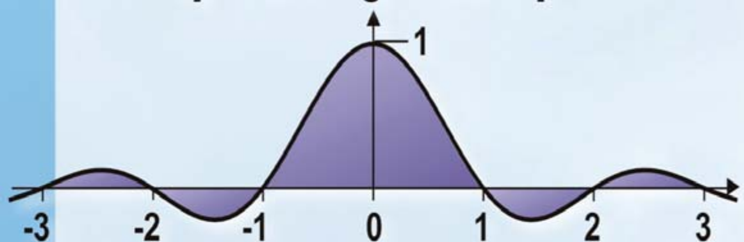
- Reconstruct values of field *f* at arbitrary locations along each ray with a convolution filter

$$s = f(x, y, z)$$

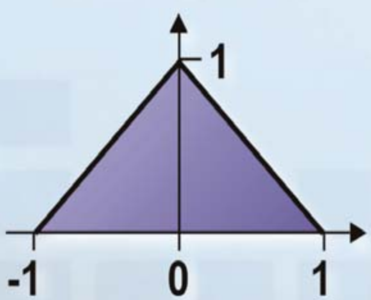




Box



Sinc

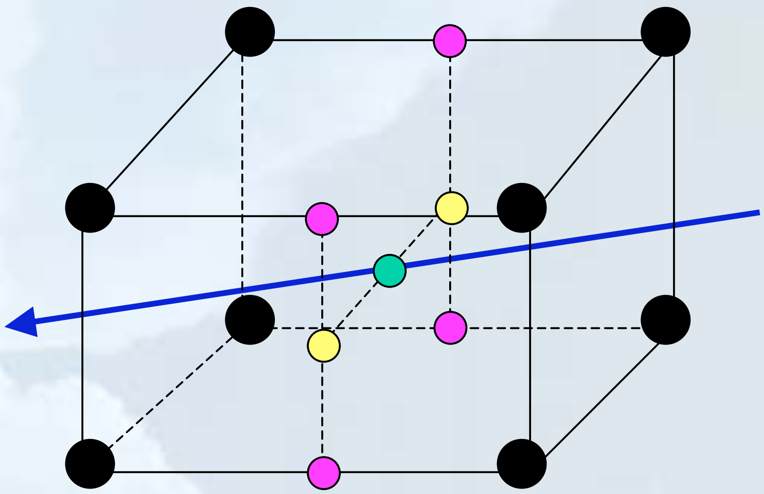


Tent

Higher order filters produce more accurate results but with greater computational cost

Reconstruction
Tri-linear interpolation for a rectilinear cell

- Seven linear interpolations per cell
- Most commonly used interpolation method



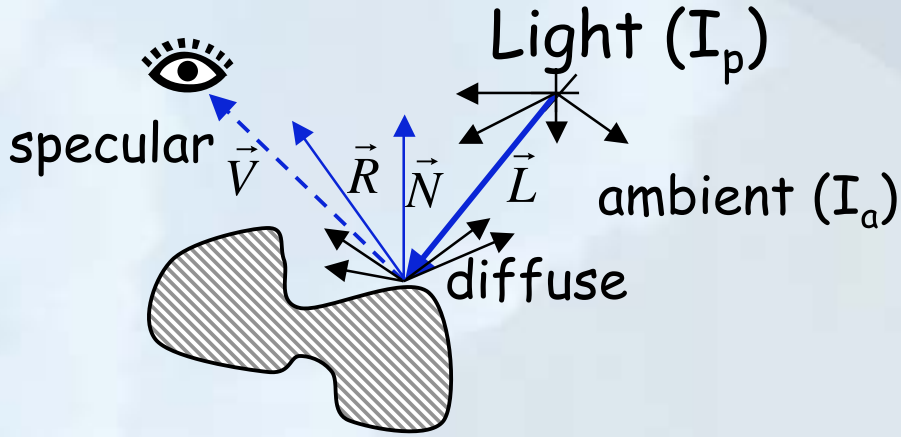
Illumination

NCAR

- If we want our volume rendering integral to include scattering we need to include an optical model to compute the scattering term, $q_I(s)$
- The Phong model works well in practice (Phong 1975)
- How do we compute the surface normal, $\boldsymbol{N}$?

Remember, illumination can modulate color in unexpected ways

Phong illumination



$$I = Ambient + Diffuse + Specular$$

$$Ambient = I_a k_d$$ emission

$$Diffuse = I_p k_d \left(\vec{N} \cdot \vec{L}\right)$$ scattering

$$Specular = I_p k_s \left(\vec{R} \cdot \vec{V}\right)^n$$

$$\left(0.0 \le k \le 1.0\right)$$

Illumination
Normal estimation

NCAR

- Estimate normal with field's gradient
- Central difference estimator is most typical
- Reconstruction of field required to compute gradient
- Lighting model requires normalized normal vector:

$$\vec{N}(x) = \frac{\nabla f(x)}{\left\|\nabla f(x)\right\|}$$

- What happens when gradient vector magnitude is zero?

$$\vec{N} \approx \nabla f(x) = \begin{Bmatrix} \dfrac{\partial f(x)}{\partial x} \\ \dfrac{\partial f(y)}{\partial y} \\ \dfrac{\partial f(z)}{\partial z} \end{Bmatrix}$$

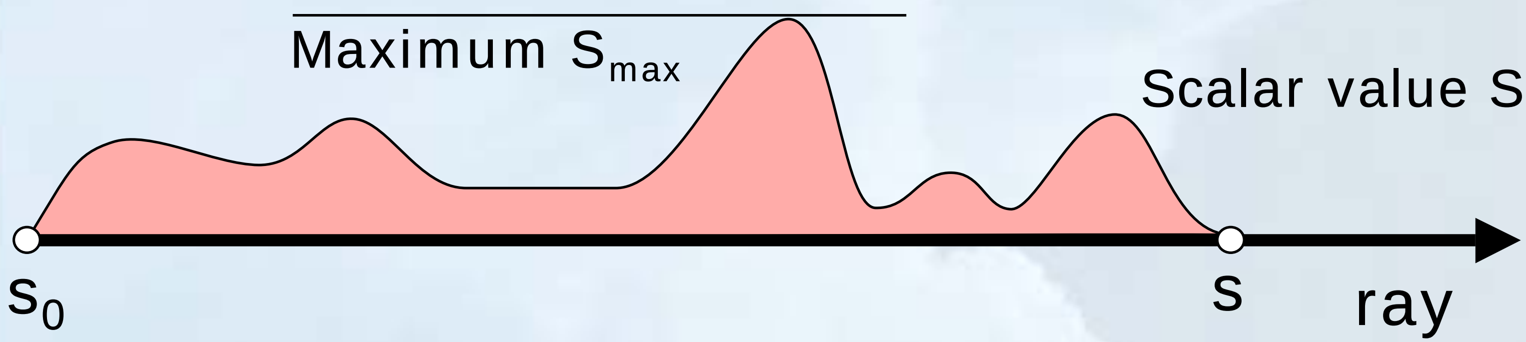
$$\vec{N} \approx \nabla f(x) \approx \frac{1}{2h} \begin{Bmatrix} f(x+h,y,z) - f(x-h,y,z) \\ f(x,y+h,z) - f(x,y-h,z) \\ f(x,y,z+h) - f(x,y,z+h) \end{Bmatrix}$$

Ray traversal schemes

- Accumulate (what we’ve been doing)
- Maximum Intensity Projection
- Threshold

Ray Traversal
Maximum Intensity Projection (MIP)

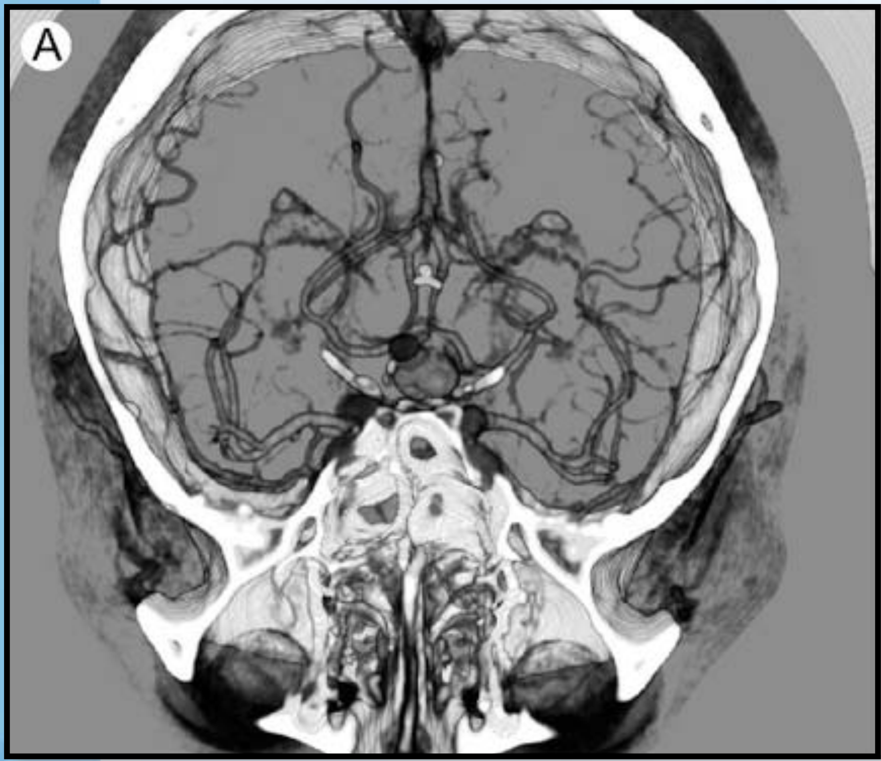
- No **emission** or **absorption**
- Pixel value is the **maximum** scalar value along the viewing ray

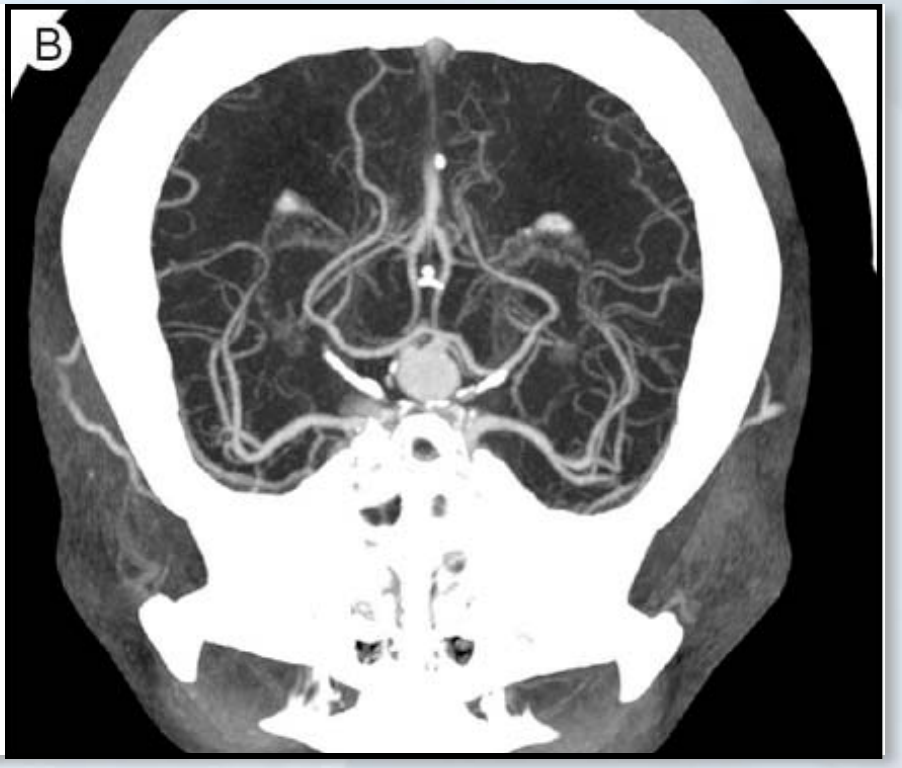


- **Advantage:** transfer function not required
- **Drawback:** misleading depth information

Slide credit: Markus Hadwiger

Maximum Intensity Projection





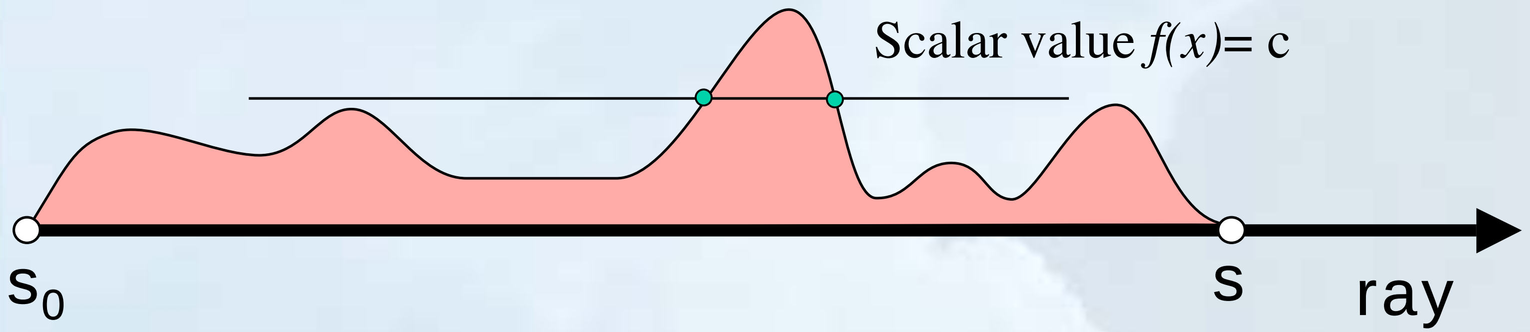
Emission/Absorption

Maximum Intensity Proj.

Slide credit: Markus Hadwiger

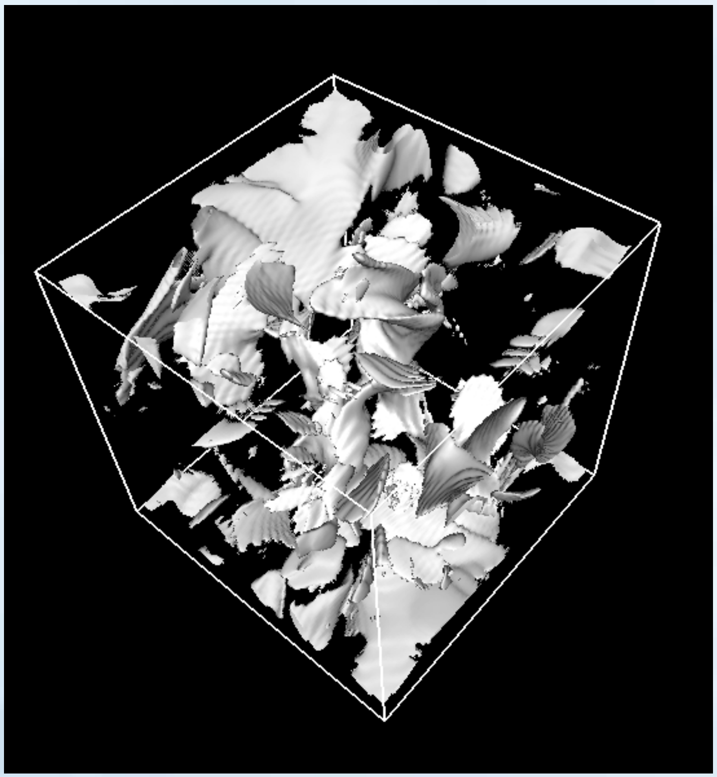
Ray Traversal
Threshold

- Isosurfaces without geometry
- No **emission** or **absorption**
- Pixel value is the first sample to meet a threshold value (or the accumulation of all threshold crossings if transparency included)

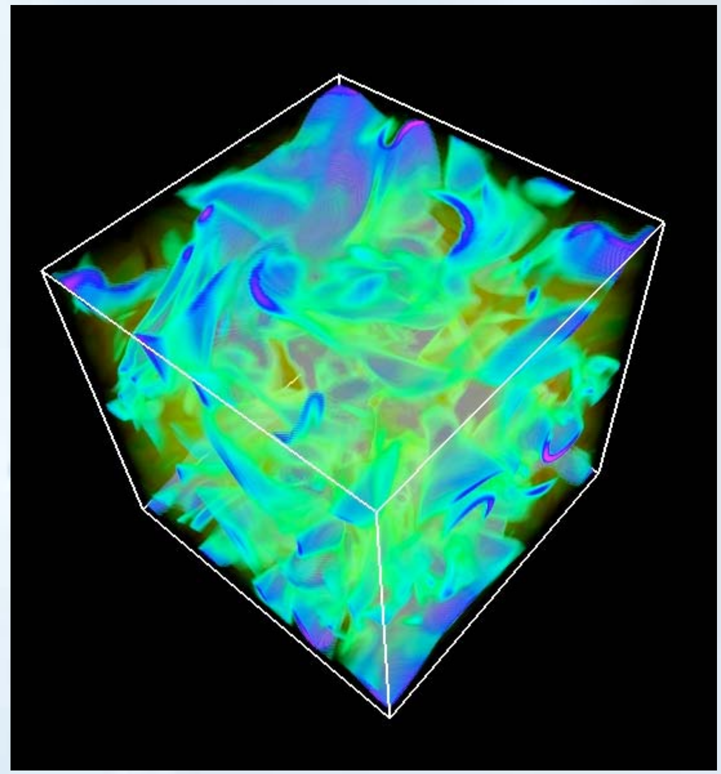


- **Advantage:** Hardware implementation facilitated

Slide credit: Markus Hadwiger



Threshold



Absorption/emission

Hardware implementation on the graphics processing unit

- Modern GPUs offer orders of magnitude more computing power than CPUs. But…
- Inputs to GPUs
  - Simple geometry: triangles, quads, lines, etc.
  - Geometric attributes: position, color, transparency, etc.
  - Textures: 1D, 2D, 3D arrays of 4-tupples (r,g,b,α)
    - Most commonly 2D textures are images, photographs, etc.
    - Textures cannot be rendered (drawn) directly
      - They are attributes that may be mapped to geometric primitives
- Programming interface is highly restricted

- How do we volume render?

Modern day graphics processor (GPU)

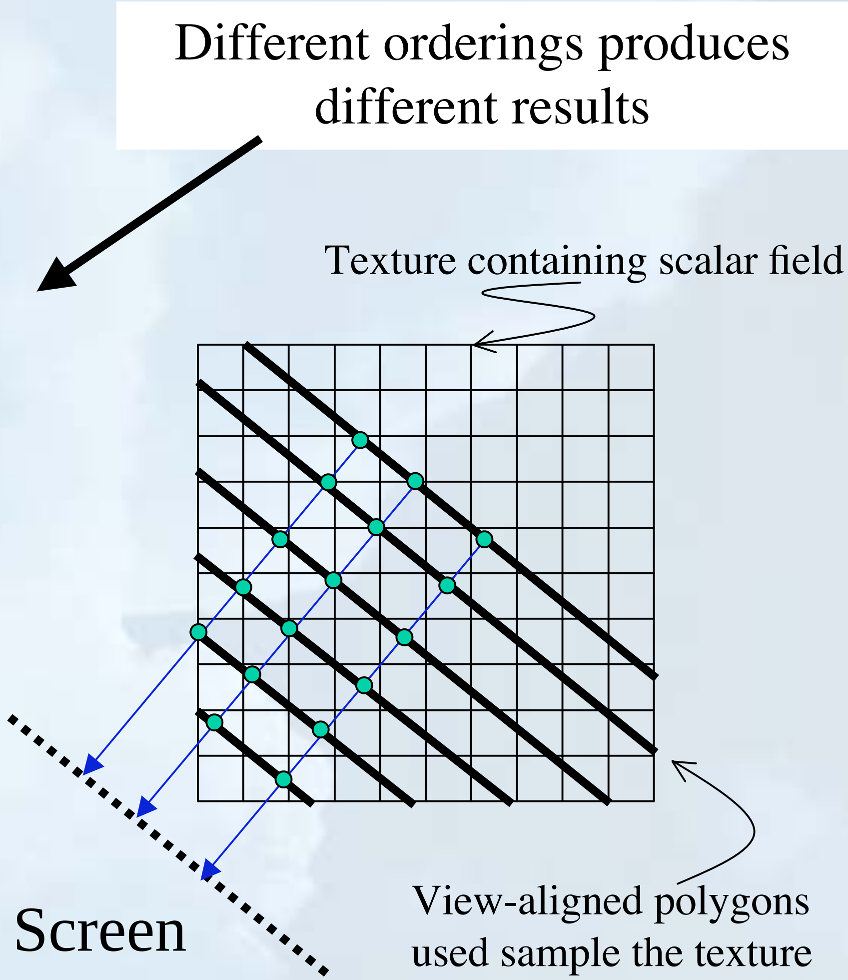
- Capabilities
  - Processing cores: *~256*
  - Memory bandwidth: *~100 GBs*
  - Floating point performance: *~1000* GFLOPS
  - Video (data) memory: ~1GB
  - Max number of instructions: ~2048
- Cost: ~$500
- Bus interface: PCIe x16 (4-8 GBs)
- Programming model: streaming vector processing (data viewed as streams, computation as kernels)
  - Graphics APIs (OpenGL, Direct3D) + shader language
    - Specialized 4GL languages for graphics, not general computing
  - More general purpose languages (e.g. CUDA) starting to appear
- Primary market: computer gaming
  - If the gaming industry doesn't need it, don't expect to see it on a GPU

Opportunities for parallelism

Many times faster than today's CPUs (10-40x), but offer a limited and complex programming mode

Hardware volume rendering
2D data example

- Volume data are loaded as a graphics texture
- Polygons are constructed to sample the texture
- Texture samples are mapped through a user-defined transfer function
- Polygons are rendered from front to back (or back to front)
- Textures are resampled and applied to polygons
- Polygons are blended together using a compositing operation that approximates the discrete volume rendering integral



Front to back compositing

We approximate:

$$I(D) = \sum_{i=0}^{n} c_i \prod_{j=i+1}^{n} T_j, \text{ where } c_0 = I(s_0)$$

with some hand waving by iterative application of:

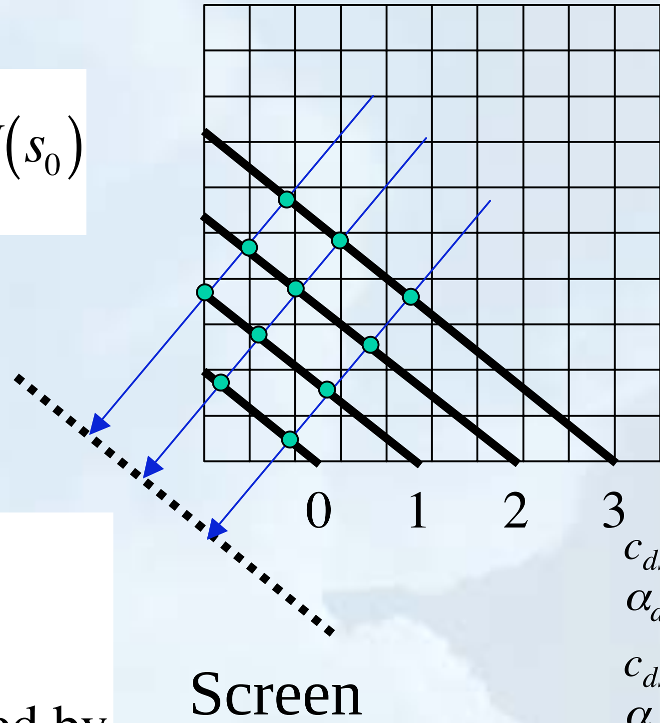
$$C_{dst} \leftarrow C_{dst} + (1-\alpha_{dst})C_{src}$$

$$\alpha_{dst} \leftarrow \alpha_{dst} + (1-\alpha_{dst})\alpha_{src}$$

where α_{dst} and C_{dst} are initialized by

$$C_{dst} = c_0$$

$$\alpha_{dst} = \alpha_0 \qquad (\alpha = 1 - T)$$



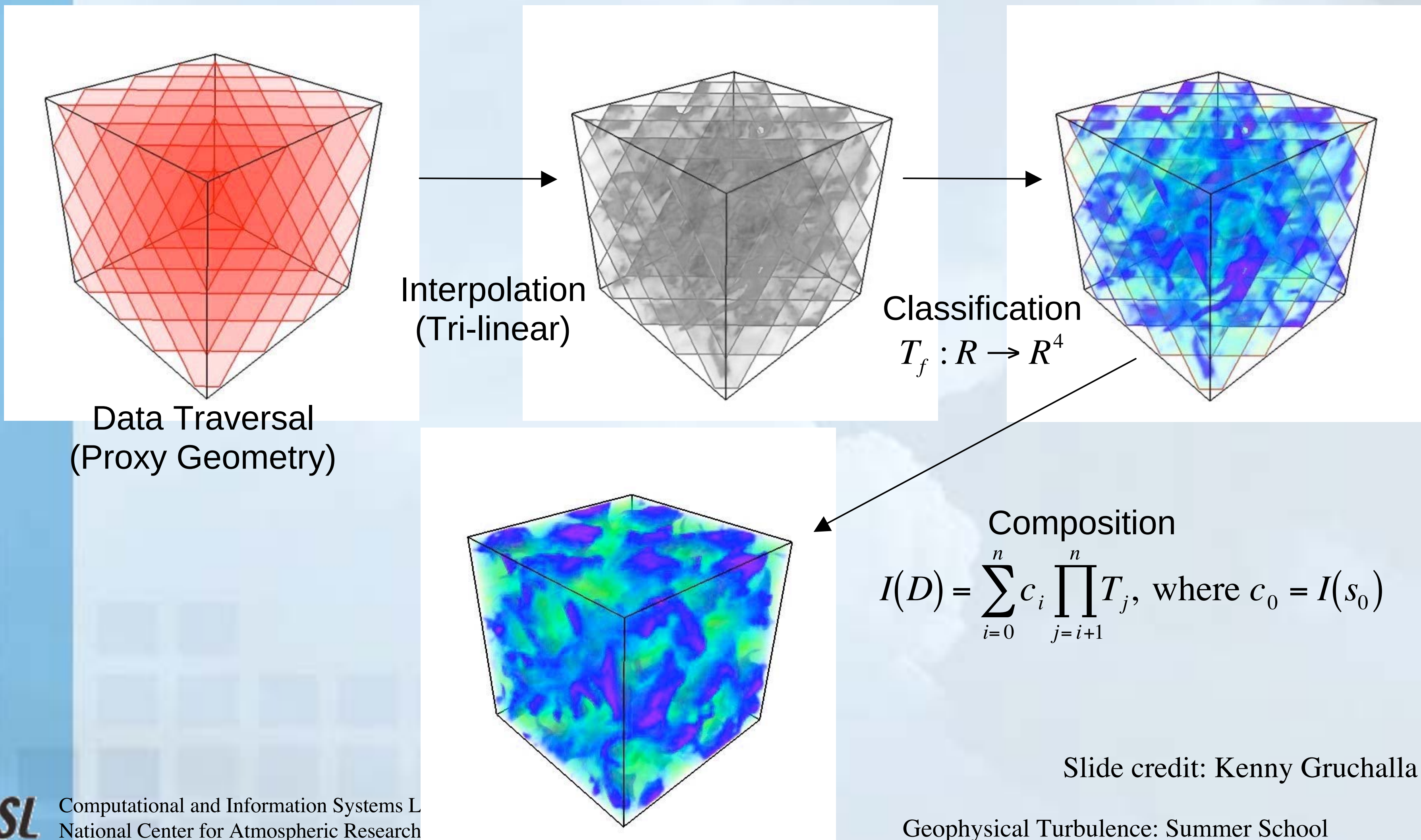
$c_{dst} = c_{dst} + (1- \alpha_{dst})c_0$
$\alpha_{dst} = \alpha_{dst} + (1- \alpha_{dst})\alpha_0$

$c_{dst} = c_{dst} + (1- \alpha_{dst})c_1$
$\alpha_{dst} = \alpha_{dst} + (1- \alpha_{dst})\alpha_1$

$c_{dst} = c_{dst} + (1- \alpha_{dst})c_2$
$\alpha_{dst} = \alpha_{dst} + (1- \alpha_{dst})\alpha_2$

$c_{dst} = c_{dst} + (1- \alpha_{dst})c_3$
$\alpha_{dst} = \alpha_{dst} + (1- \alpha_{dst})\alpha_3$

3D Texture-Based Volume Rendering



$$I(D) = \sum_{i=0}^{n} c_i \prod_{j=i+1}^{n} T_j, \text{ where } c_0 = I(s_0)$$

Slide credit: Kenny Gruchalla

Hardware volume rendering caveats

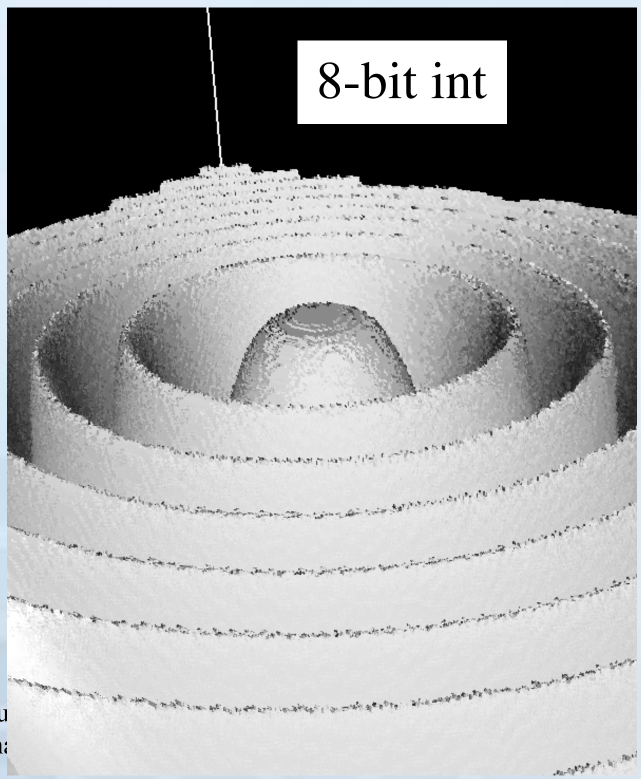
- Irregular grids are difficult to support
  - Rectilinear grids with isotropic geometry are most widely supported by far
- Video memory has limited capacity (<= 1GB)
  - Large data volumes will require memory virtualization, which introduces complexity and computational cost
- Quantization
- Aliasing
  - Several sources

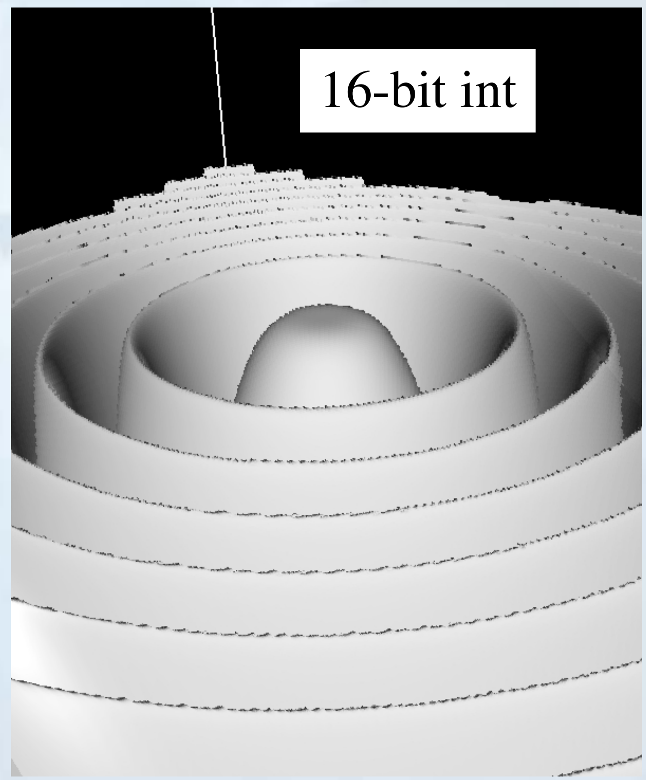
Outline

- Theoretical foundations
- Implementation
- Understanding and avoiding sources or error

Quantization errors

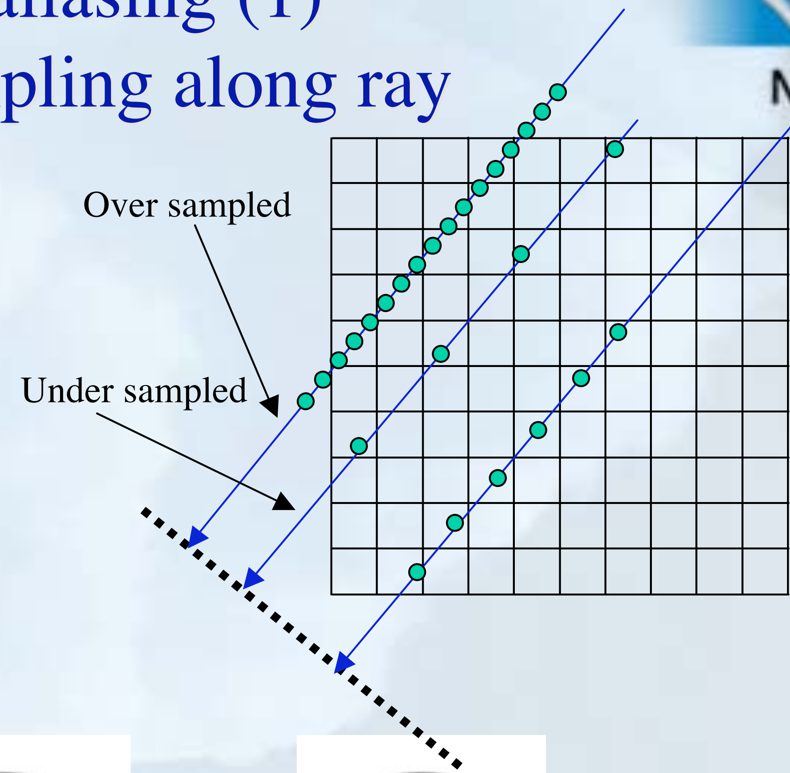
- Texture elements on current graphics cars support limited precision and/or type
  - 8, 16 bits
  - integer only (no floating point)
- High-dynamic range data may not quantize well
- Gradient calculation also suffers

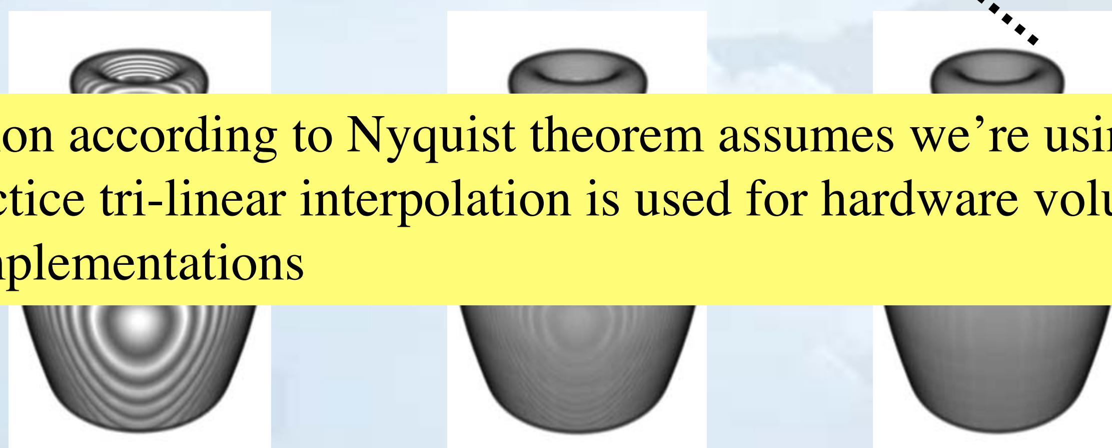




Sources of aliasing (1)
Insufficient sampling along ray

- Sampling theory tells us to sample at greater than *Nyquist frequency*
  - More computation
  - Better quality
- Trade off between speed and accuracy

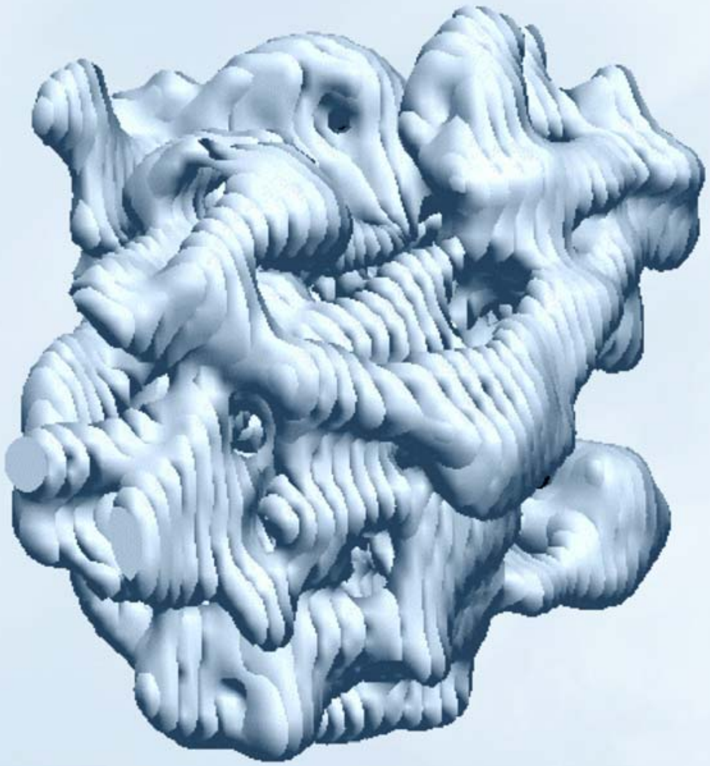




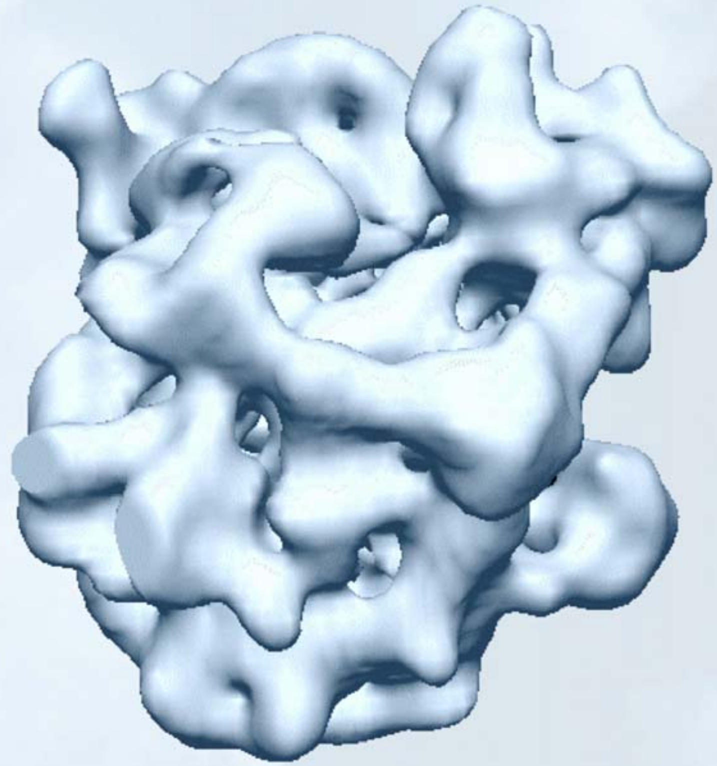
1/20x 1/10x 1x

Reconstruction according to Nyquist theorem assumes we're using a sinc filter. In practice tri-linear interpolation is used for hardware volume rendering implementations

Sources of aliasing (1)
Insufficient sampling along ray



64 samples

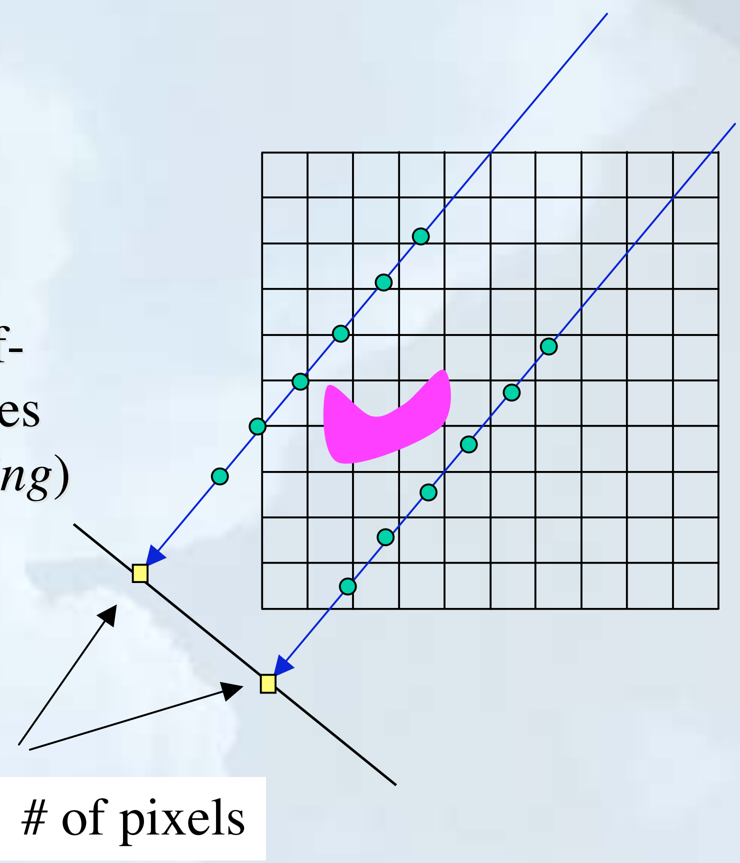


640 samples

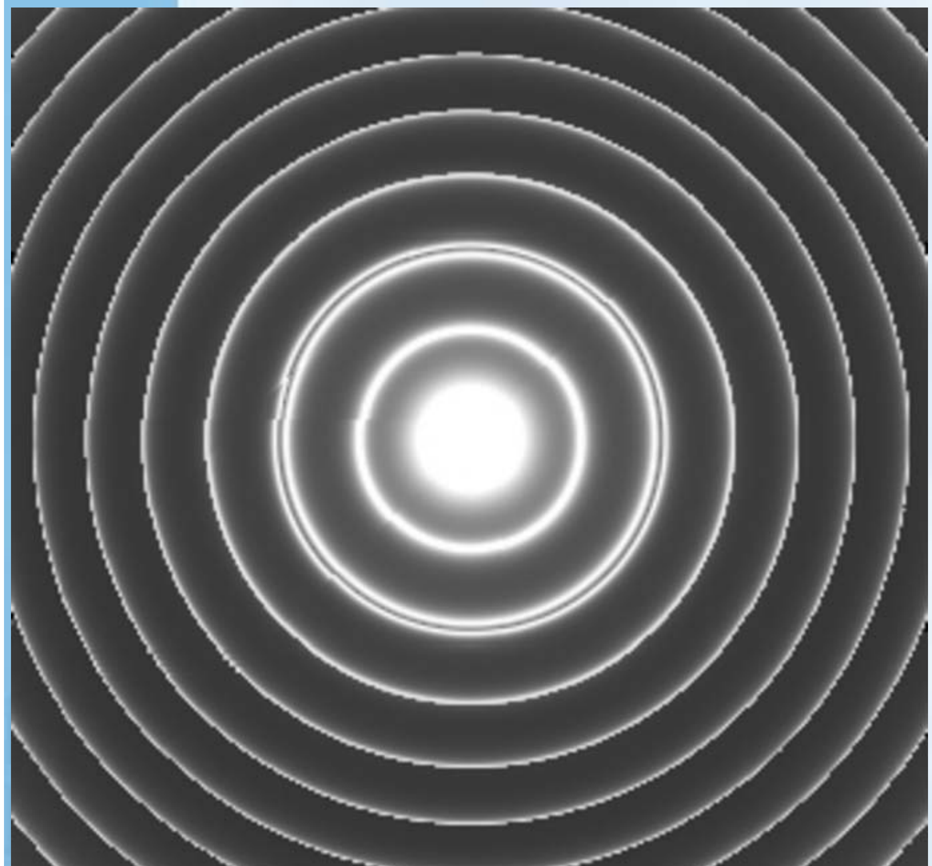
Sources of aliasing (2)
Insufficient number of rays

For hardware volume rendering the number of rays is most typically determined by the display window resolution

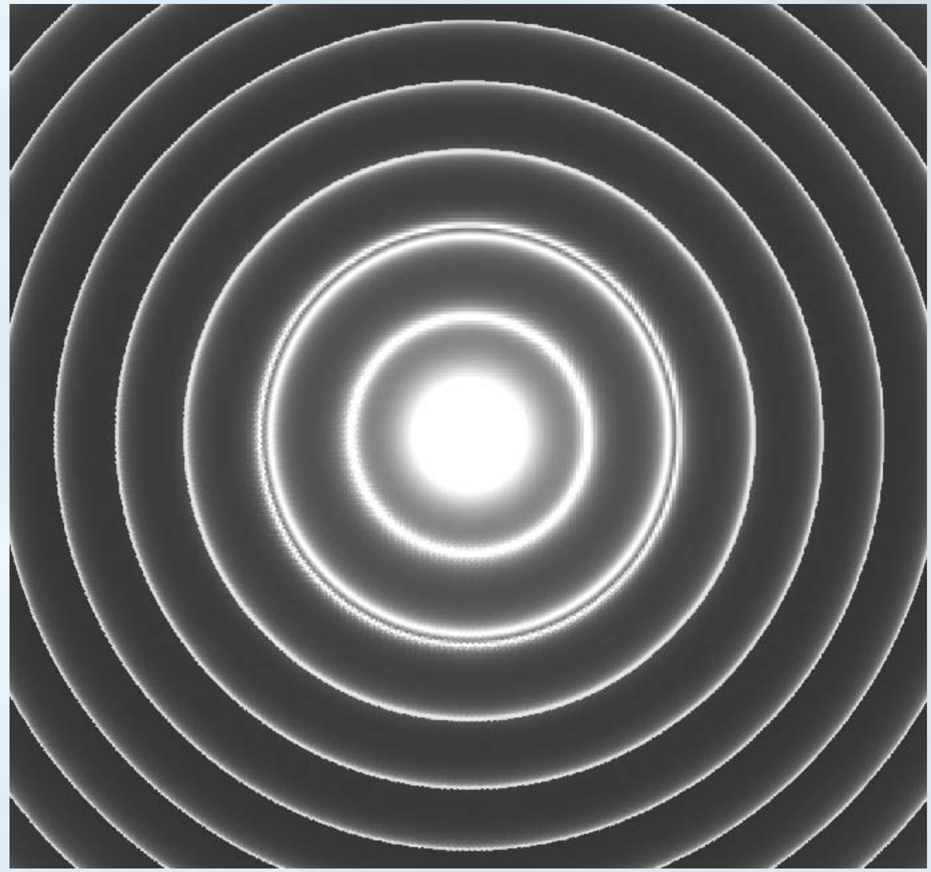
Some implementations may support off-screen rendering at higher sampling rates than the displayed image (*super sampling*)



Sources of aliasing (2)
Insufficient number of rays



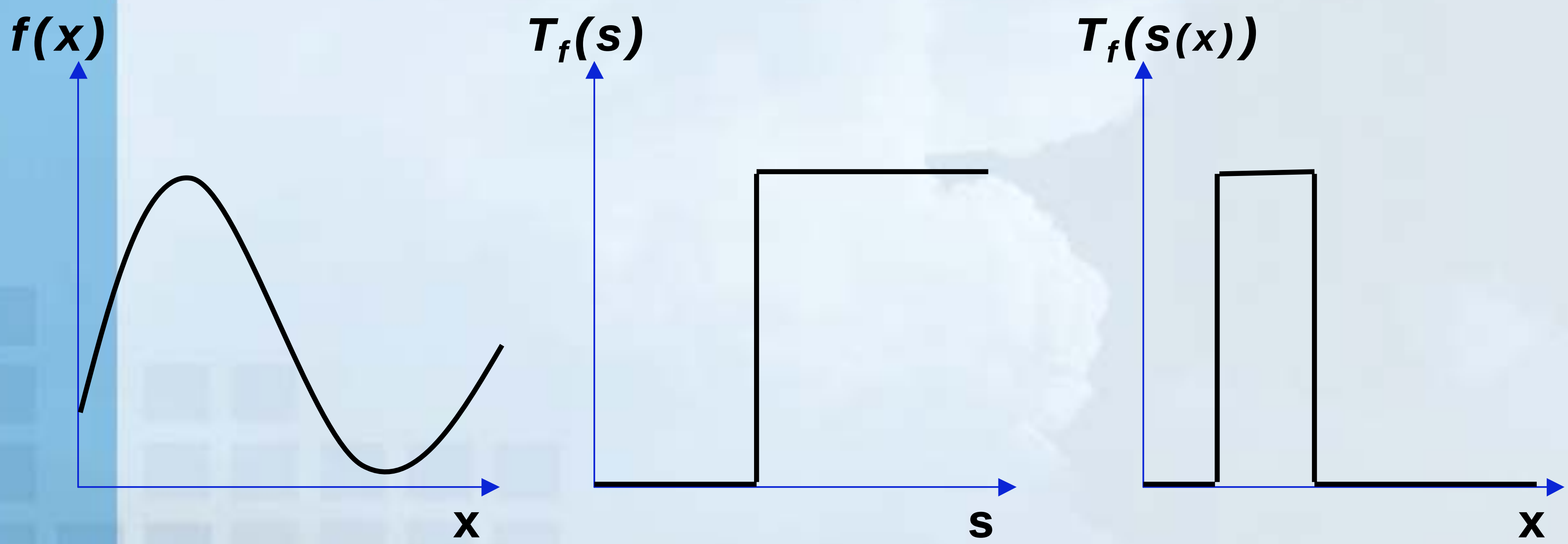
1/2x sampling



1x sampling

Sources of aliasing (3)
Classification

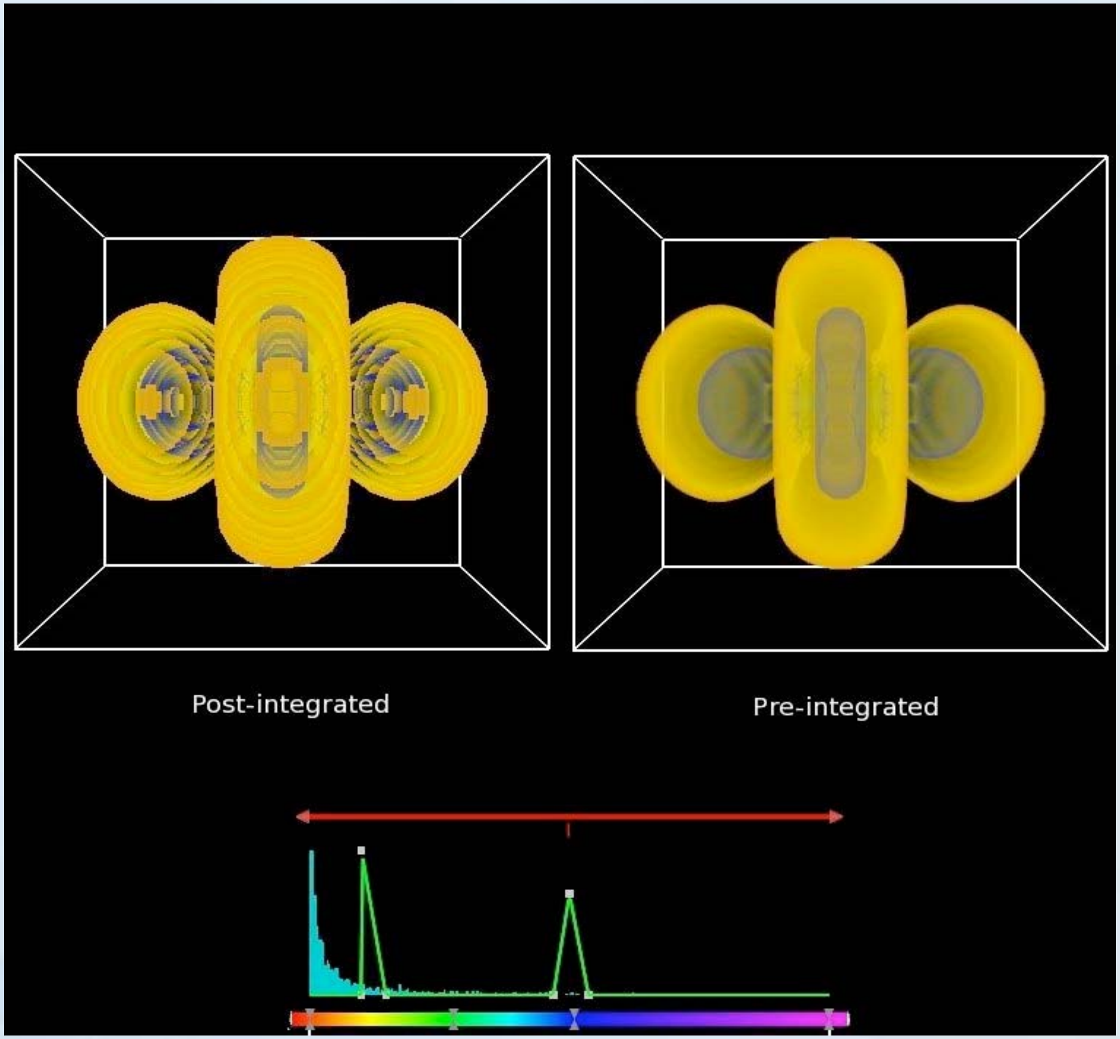
- We assume sampling theory has led to proper discretization of the data set
- However, the transfer function can introduce frequencies higher than those contained in our original function
- Higher sampling rate needed
- Increased sampling => increased expense



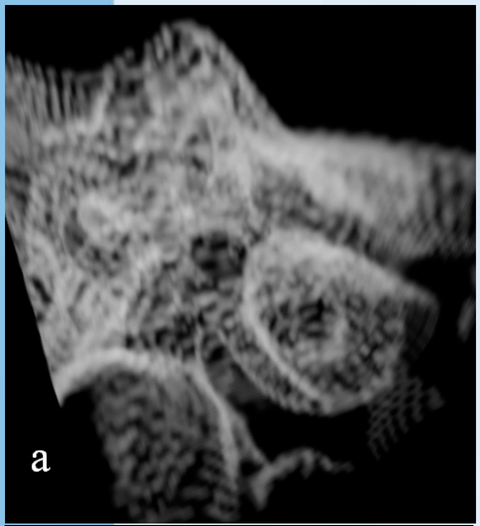
Sources of aliasing (3)
Pre-integrated classification (Engle 2001, Rottger 2000)

- Hardware volume rendering solution to high-frequency transfer functions
- Split numerical integration into two:
  1. One integration for the scalar field at sampling rate prescribed by the grid
  2. One (pre) integration of transfer function with integration step size determined by sampling of transfer function
     - Requires “advanced” programmable GPU
     - Transfer integration table must be re-calculated for every change to the transfer function.

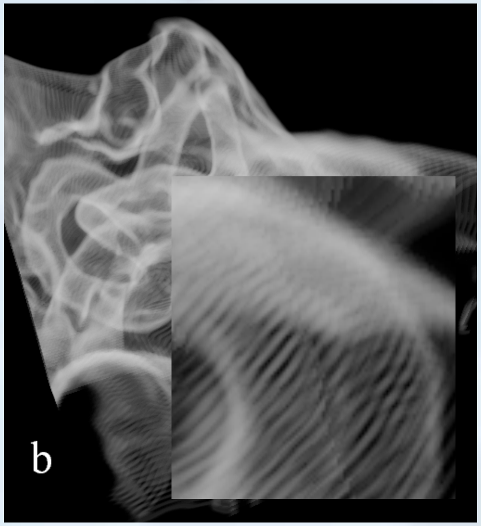
Sources of aliasing (3)
Pre-integrated classification



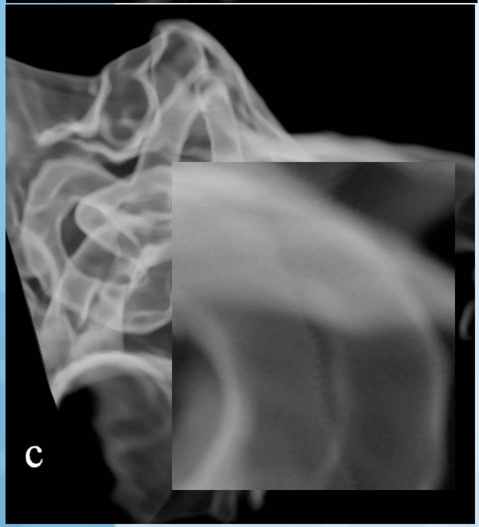
Pre-Integration Quality



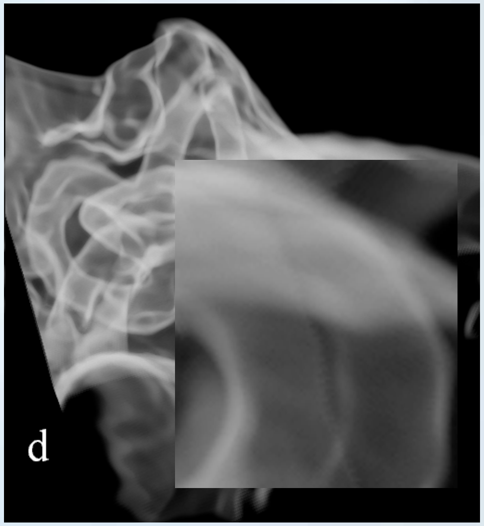
128 slices
pre-
classification



128 slices
post-
classification



284 slices
post-
classification



128 slices
pre-integrated

Slide credit: Markus Hadwiger

Summary

- Volume rendering treats a 3D scalar field as if it were a gas, modeling light transport through that gas
- Assignment of optical properties to the scalar field is known as *Classification* and is performed via a *Transfer Function*
- Hardware implementations are the most common for interactive performance
- Numerous sources of error that may be introduced by the implementation as well as the specification of the Transfer Function (avoid abrupt transitions, etc.)
- Understanding and recognizing rendering artifacts is essential for accurate interpretation of results