3

Raycasting

Direct volume rendering

Volume rendering (sometimes called direct volume rendering) stands for methods that generate images directly from 3D scalar data.

"Directly" means: no intermediate geometry (such as an isosurface) is generated.

Volume rendering techniques

- depend strongly on the grid type
- exist for structured and unstructured grids
- are predominantly applied to uniform grids (3D images).

2D or 3D image data are uniform grids with cell-centered data.

Cell-centered data

- are attributed to cells (pixels, voxels) rather than nodes
- can also occur in (finite volume) CFD datasets
- are converted to node data
  - by taking the dual grid (easy for uniform grids, n cells $\rightarrow$ n-1 cells!)
  - or by interpolating.

Raycasting

Raycasting is historically the first volume rendering technique.

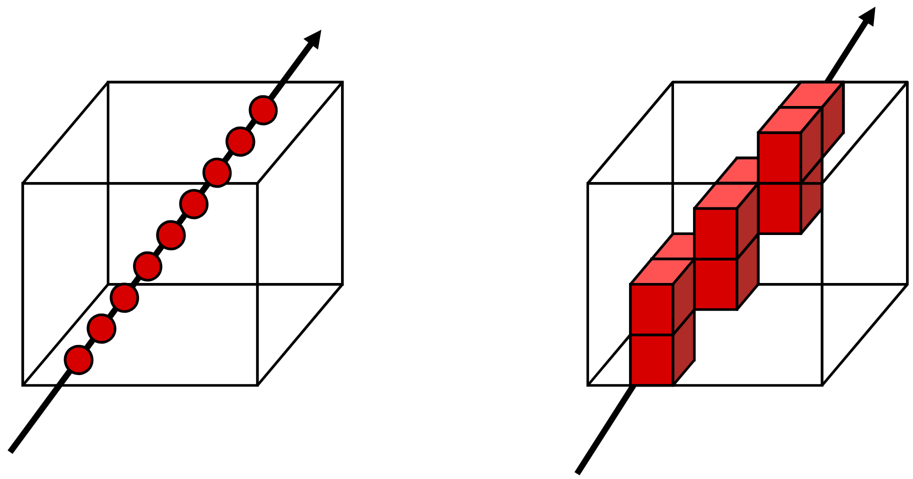
It has common with raytracing:

- image-space method: main loop is over pixels of output image
- a view ray per pixel (or per subpixel) is traced backward
- samples are taken along the ray and composited to a single color

Differences are:

- no secondary (reflected, shadow) rays
- transmitted ray is not refracted
- more elaborate compositing functions
- samples are taken at intervals (not at object intersections)

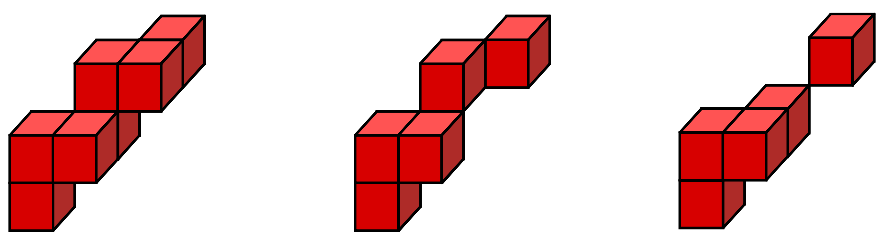
Sampling interval can be fixed or adjusted to voxels:



uniform sampling

voxel-by-voxel traversal
(faster)

Connectedness of "voxelized" rays:



6-connected
(strongest)

18-connected

26-connected
(weakest)

Ray templates

A ray template (Yagel 1991) is a voxelized ray which by translating generates all view rays.

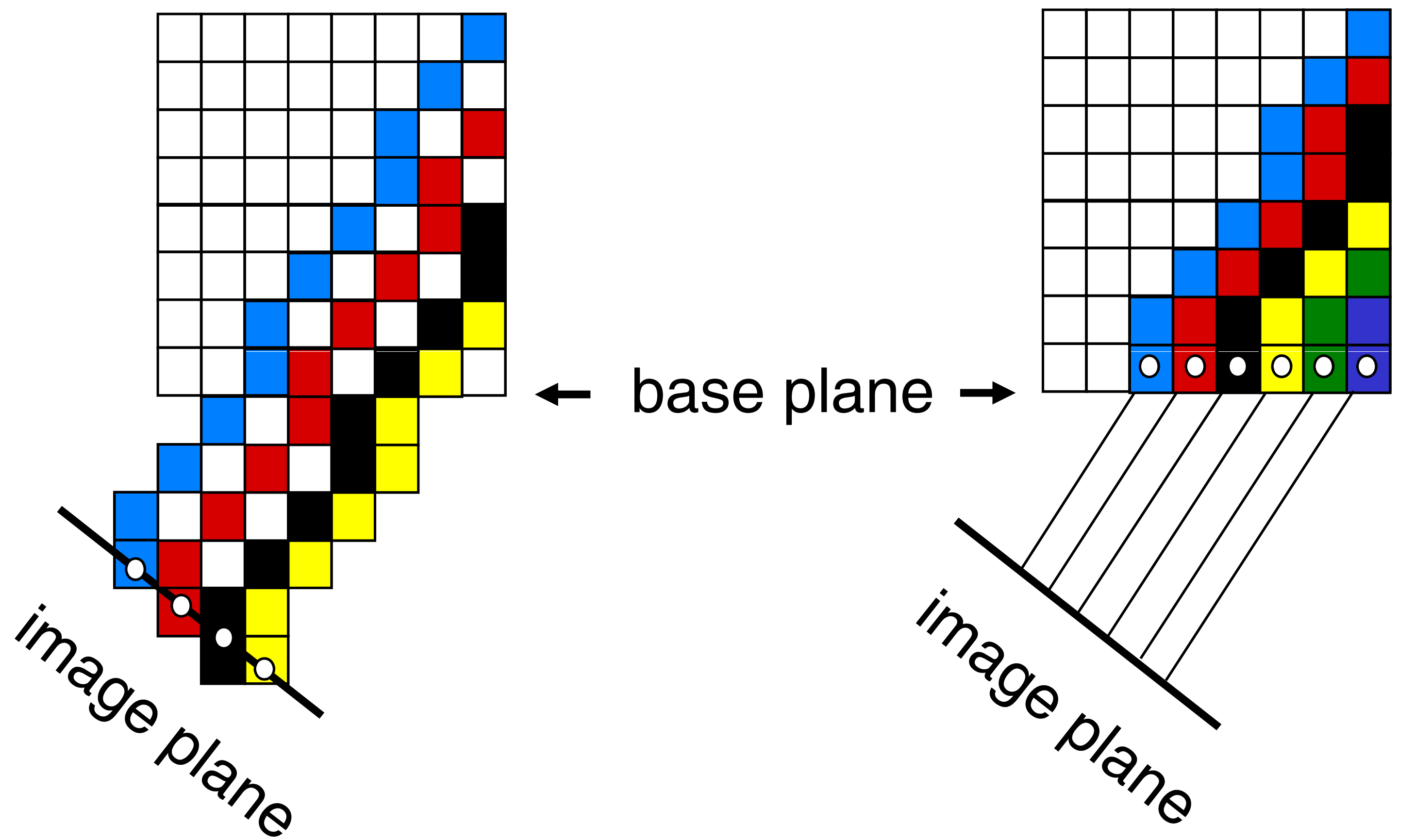
Ray templates speed up the sampling process, but are obviously restricted to orthographic views.

Algorithm:

- Rename volume axes such that z is the one "most orthogonal" to the image plane.
- Create ray template with 3D version of Bresenham algorithm, giving 26-connected rays which are functional in z coordinate (have exactly one voxel per z-layer)
- Translate ray template in base plane, not in image plane

Incorrect: translated in image plane

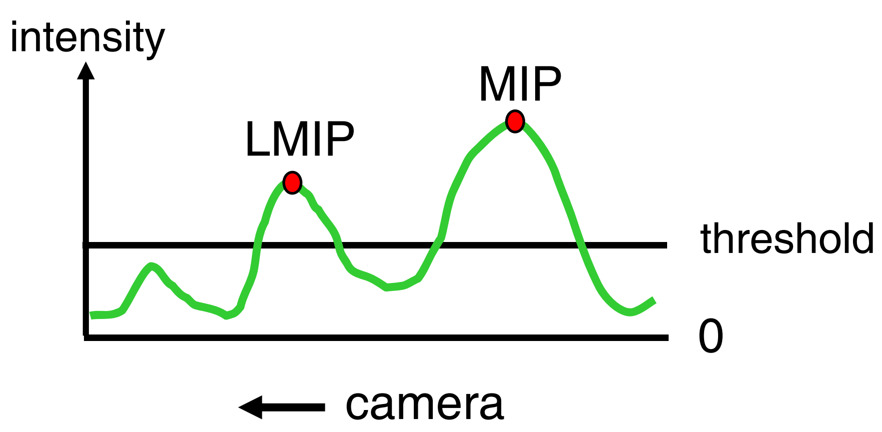
Correct: translated in base plane



Compositing

Two simple compositing functions can be used for previewing:

- Maximum intensity projection (MIP):
  - maximum of sampled vaules
  - result resembles X-ray image
- Local maximum intensity projection (LMIP):
  - first local maximum which is above a prescribed threshold
  - approximates occlusion
  - faster & better(!)



Comparison of techniques (Y. Sato, dataset of a left kidney):

Isosurface vs. raycasting with MIP, LMIP, α-compositing

fast	fast	fast	-
(1 parameter)	parameter free	(1 parameter)	-
-	full data range	full data range	full data range
-	noise insensitive	noise insensitive	noise insensitive
lighting	-	-	lighting
occlusion	-	(occlusion)	occlusion
(transparency)	-	-	transparency

α-compositing

Assume that each sample on a view ray has color and opacity:

$$(C_0, \alpha_0), \cdots, (C_N, \alpha_N) \qquad C_i \in [0,1]^3, \alpha_i \in [0,1]$$

where the 0th sample is next to the camera

and the Nth one is a (fully opaque) background sample:

$$C_N = (r, g, b)_{\text{background}}$$
$$\alpha_N = 1$$

α-compositing can be defined recursively:

Let C_f^b denote the composite color of samples $f, f+1, \ldots, b$

Recursion formula for back-to-front compositing:

$$C_b^b = \alpha_b C_b$$
$$C_f^b = \alpha_f C_f + (1 - \alpha_f) C_{f+1}^b$$

The first few generations, written with transparency $T_i = 1 - \alpha_i$

$$C_b^b = \alpha_b C_b$$

$$C_{b-1}^b = \alpha_{b-1} C_{b-1} + \alpha_b C_b T_{b-1}$$

$$C_{b-2}^b = \alpha_{b-2} C_{b-2} + \alpha_{b-1} C_{b-1} T_{b-2} + \alpha_b C_b T_{b-1} T_{b-2}$$

$$C_{b-3}^b = \alpha_{b-3} C_{b-3} + \alpha_{b-2} C_{b-2} T_{b-3} + \alpha_{b-1} C_{b-1} T_{b-2} T_{b-3} + \alpha_b C_b T_{b-1} T_{b-2} T_{b-3}$$

reveal the closed formula for α-compositing:

$$C_f^b = \sum_{i=f}^{b} \alpha_i C_i \prod_{j=f}^{i-1} T_j$$

front-to-back compositing can be derived from the closed formula:

Let T_f^b denote the composite transparency of samples $f, f+1, \ldots, b$

$$T_f^b = \prod_{j=f}^{b} T_j$$

Then the simultaneous recursion for front-to-back compositing is:

$$\begin{aligned} C_f^f &= \alpha_f C_f \\ T_f^f &= 1 - \alpha_f \\ C_f^{b+1} &= C_f^b + \alpha_{b+1} C_{b+1} T_f^b \\ T_f^{b+1} &= \left(1 - \alpha_{b+1}\right) T_f^b \end{aligned}$$

Advantage of front-to-back compositing: early ray termination when composite transparency falls below a threshold.

The emission-absorption model

How realistic is α-compositing?

The emission-absorption model (Sabella 1988) yields a basic volume rendering equation

$$L(x) = \int_{x}^{x_b} \varepsilon(x') e^{-\int_{x}^{x'} \tau(x'')dx''} dx'$$

The equation describes the radiance (power per unit area per solid angle [W/m^2/sr]) arriving along a ray at the position *x* on this ray.

The emission function $\varepsilon(x)$ describes the photons "emitted" by the volume along the ray.

The absorption function $\tau(x)$ is the probability that a photon traveling over a unit distance is lost by absorption.

The emission-absorption model is based on Boltzmann's transport equation in statistical physics, but completely ignores scattering.

In more general models $\tau(x)$ is an extinction function having both an absorption term and a scattering term.

Instead, in the emission-absorption model:

- incident scattering is modeled by the emission function
- loss by scattering can be thought to be part of the absorption.

Discrete version of emission-absorption model

$$L(x) = \sum_{i=0}^{n} \varepsilon_i \Delta x e^{-\sum_{j=0}^{i-1} \tau_j \Delta x} = \sum_{i=0}^{n} \varepsilon_i \Delta x \prod_{j=0}^{i-1} e^{-\tau_j \Delta x}$$

matches the α-compositing formula

$$C_f^b = \sum_{i=f}^{b} \alpha_i C_i \prod_{j=f}^{i-1} \left(1 - \alpha_j\right)$$

and gives interpretations of "opacity" and "color":

$$\alpha_i = 1 - e^{-\tau_i \Delta x}$$

$$\alpha_i C_i = \varepsilon_i \Delta x$$

The product $\tilde{C}_i = \alpha_i C_i$ is called a premultiplied or associated color.

Transfer functions

Transfer functions map raw voxel data to opacities and colors as needed for the α-compositing.

Inputs of TF (one or more):

- voxel value $s(\mathbf{x})$
- gradient magnitude $\|\nabla s(\mathbf{x})\|$
- higher derivatives of $s(\mathbf{x})$

Opacity transfer function $\alpha(s(\mathbf{x}),\|\nabla s(\mathbf{x})\|,\cdots)$

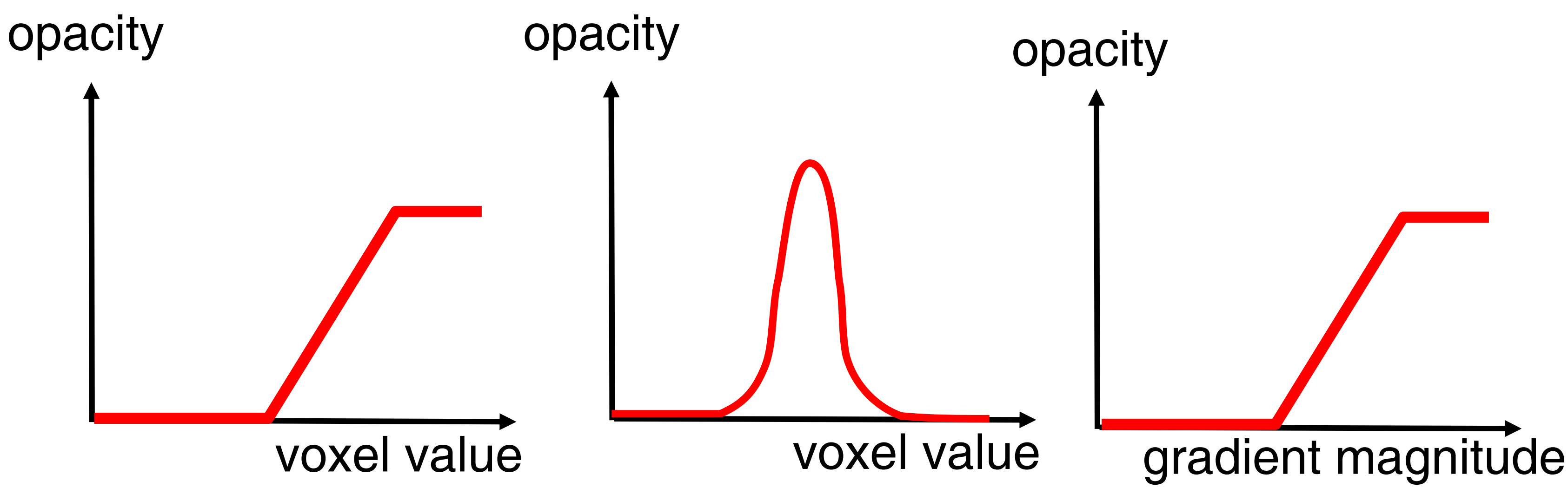
Color transfer function $C(s(\mathbf{x}),\|\nabla s(\mathbf{x})\|,\cdots)$

- or premultiplied: $\tilde{C}(s(\mathbf{x}),\|\nabla s(\mathbf{x})\|,\cdots)$

In general TF don't depend on spatial location, exception: for focus+context techniques

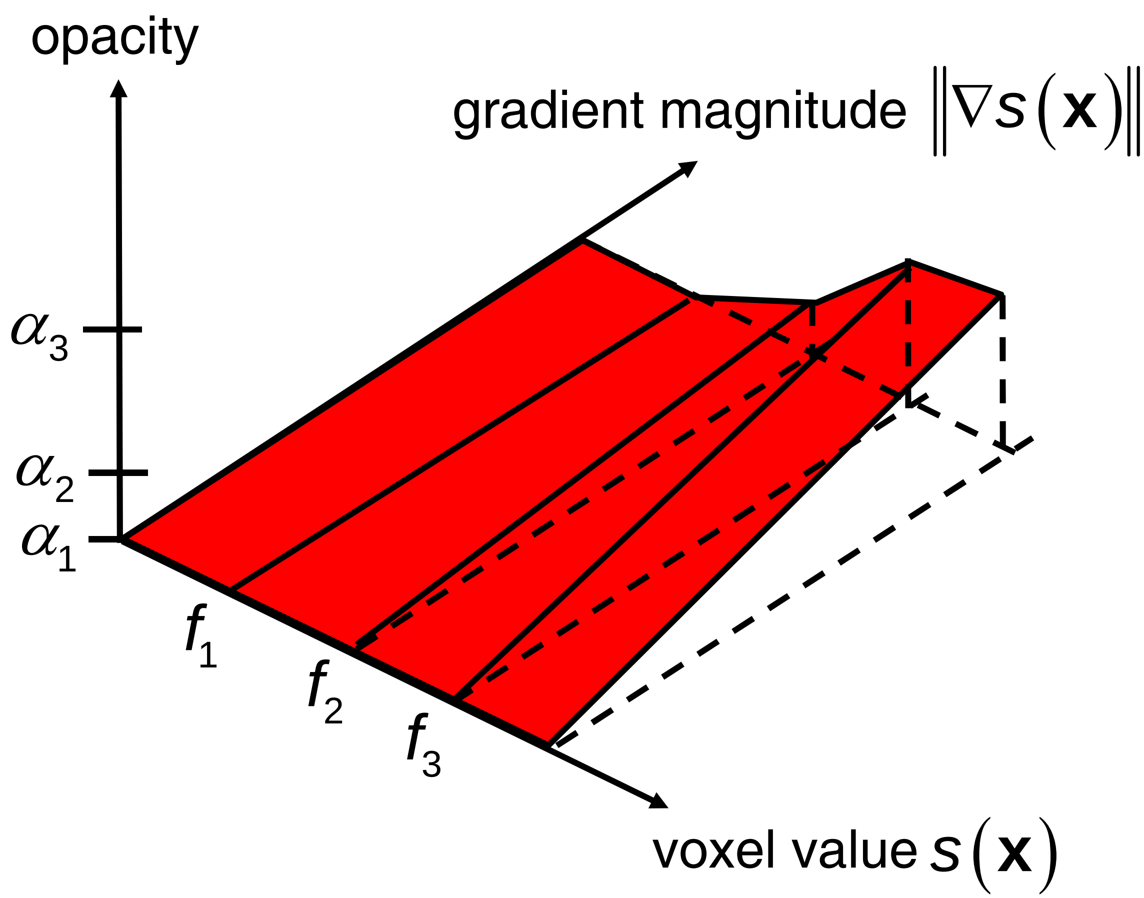
By choosing different opacity transfer functions different types of applications can be achieved.

Examples:

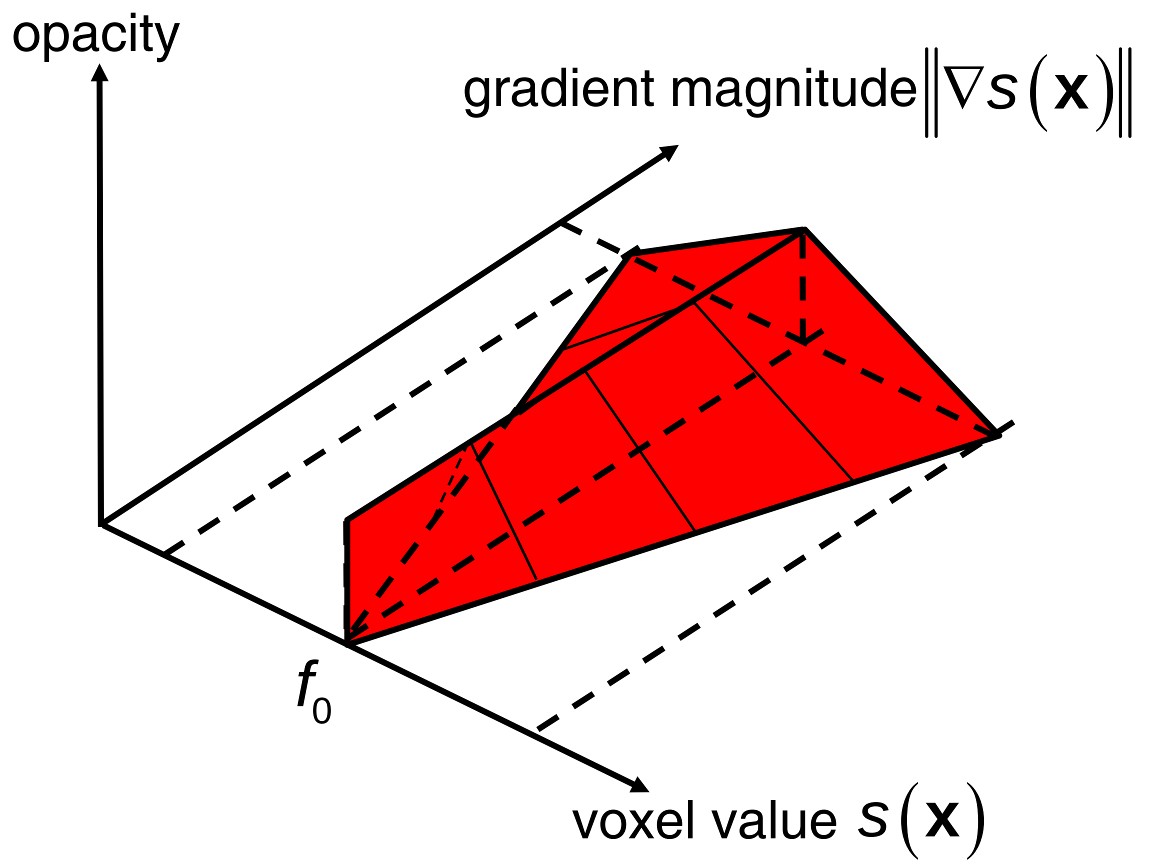


standard application | isosurface | 3D edge detector

Example of a bivariate (=2D) transfer function:

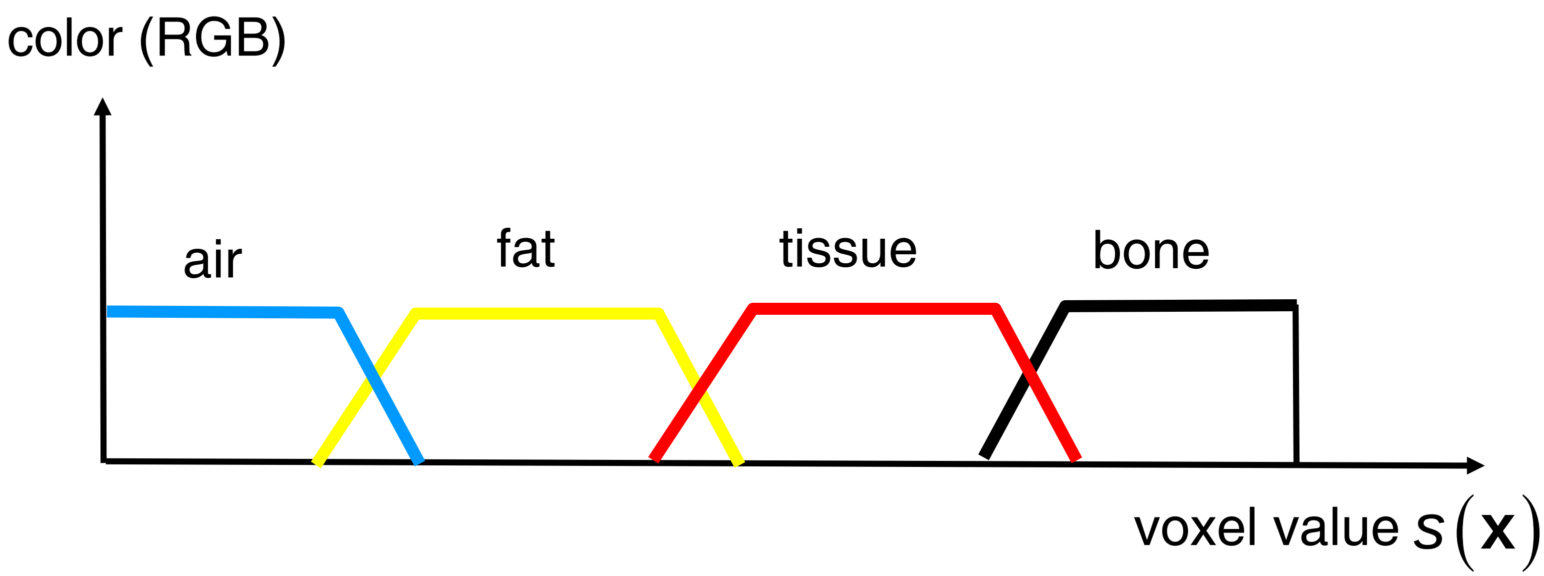


Example: bivariate transfer function for isosurface of constant "thickness".



The color transfer function allows to make a simple classification.

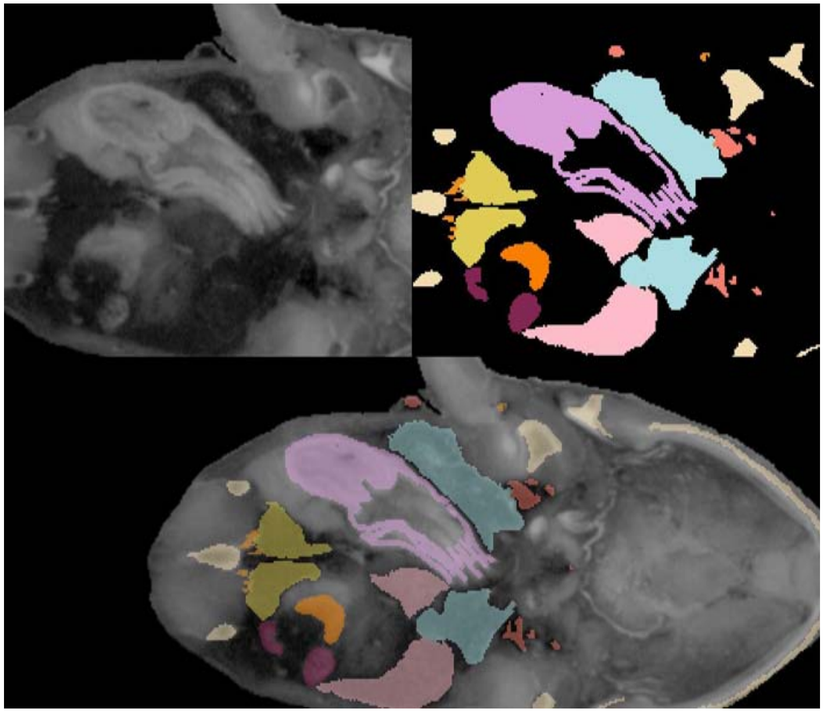
Example:



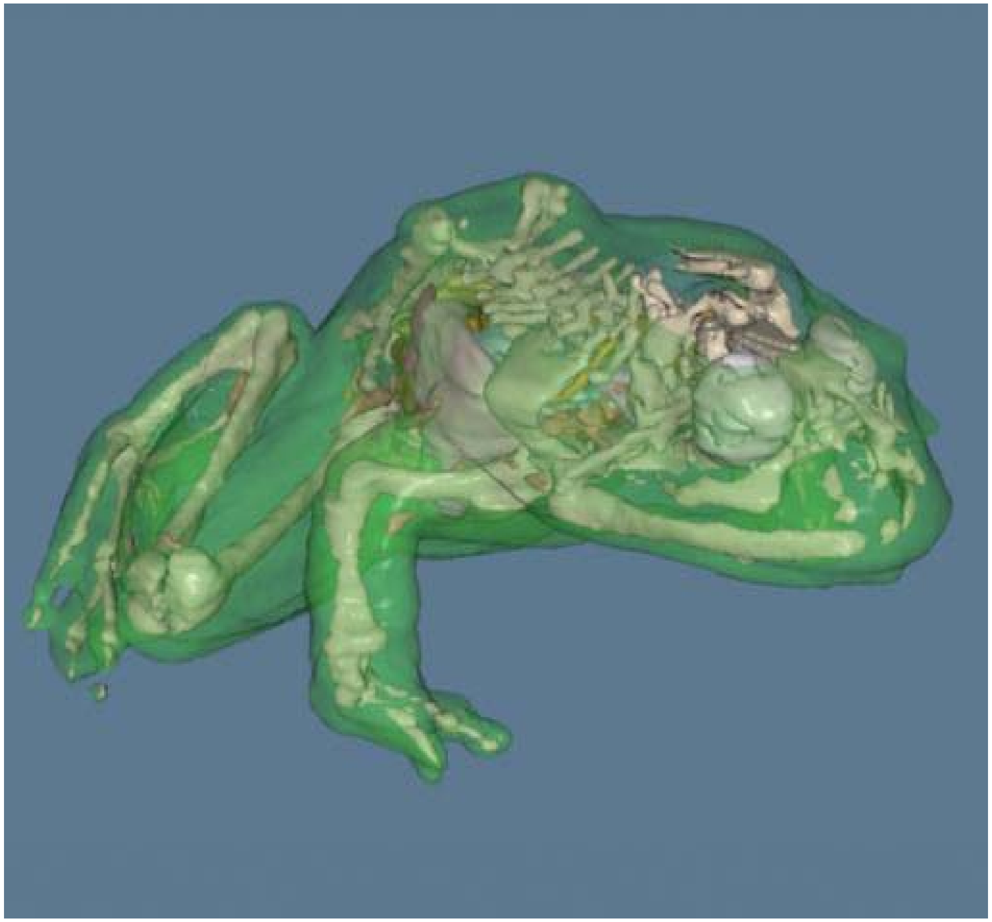
Better (but more expensive) classification than with a transfer function is obtained by segmentation (typically slice by slice, semi-automatic).

Pre-classified volume data.

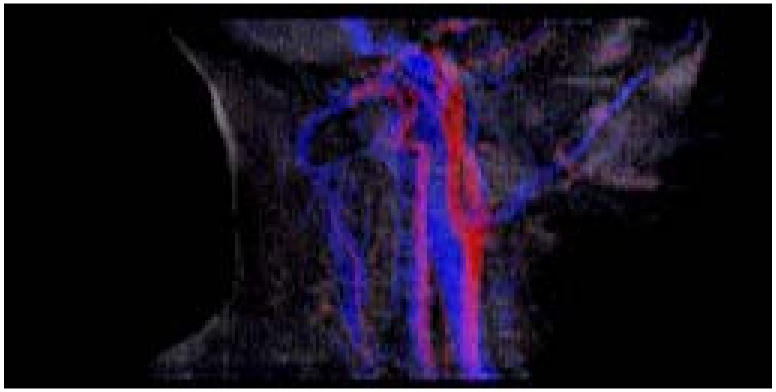
Example: "virtual frog" dataset (Lawrence Berkeley Labs).



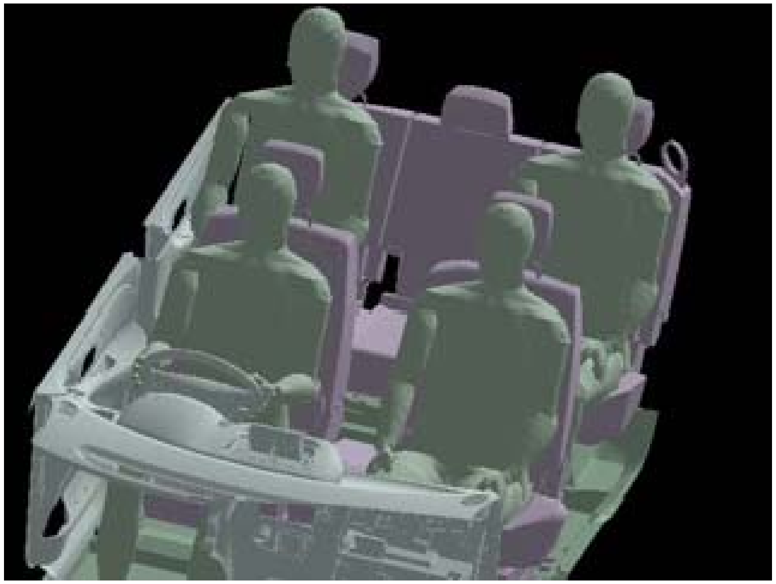
Volume rendering of segmented volume data (from VTK book).



Example: classifications with different transfer functions (McGill Univ.)



Example: temperature data for opacity and color TF (Nissan Res.Ctr.)



In pre-classification, the voxels can also be lit:

- The gradient is perpendicular to the local isosurface. It can be used as a normal vector for a Phong lighting (without rendering the isosurface itself).
- Reflection coefficients can be assigned by a separate transfer function ("materials" instead of colors only).
- The diffuse lighting can be applied to the entire volume dataset as a pre-processing since it is independent of the viewing direction.

Pre- vs. post-classification

For quality reasons, current volume rendering implementations often use post-classification.

Pre-Classification:

1. Transfer functions are applied to voxels
2. Results are interpolated to sample locations.

Post-Classification:

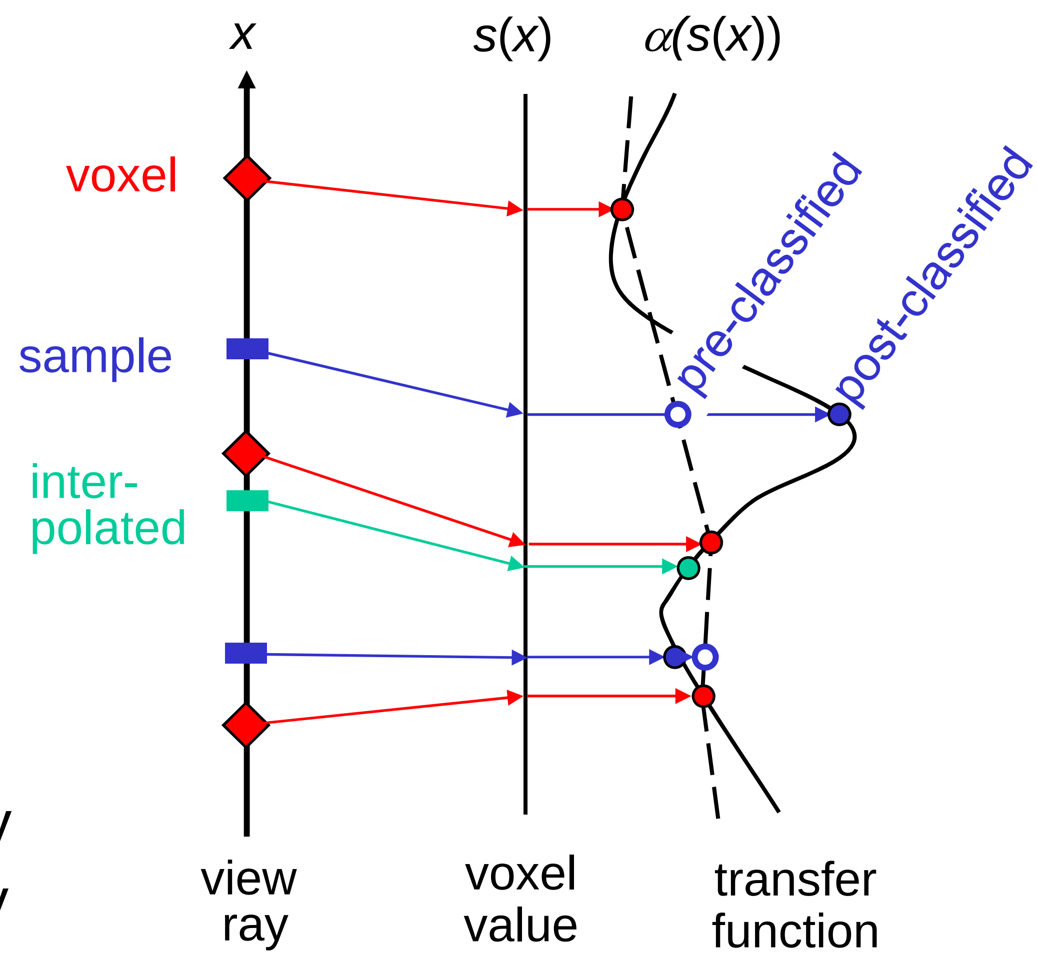
1. Raw data are interpolated to sample locations.
2. Transfer functions are applied to sampled data.

Pre-classification:

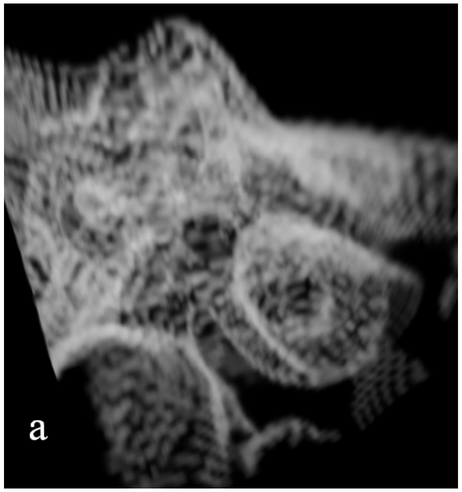
- can be done as pre-processing, e.g. segmentation, diffuse lighting

Post-classification:

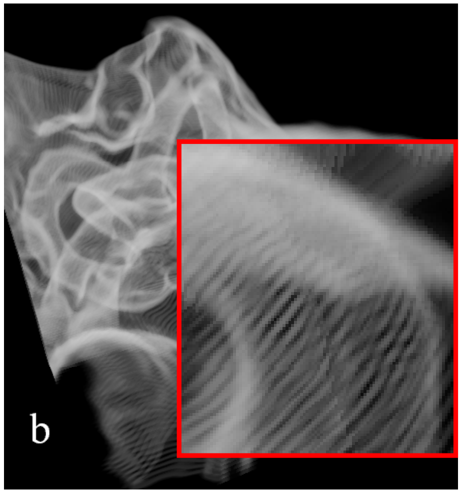
- Interpolation is in the correct space
- Additional samples can be interpolated on the fly to improve output quality



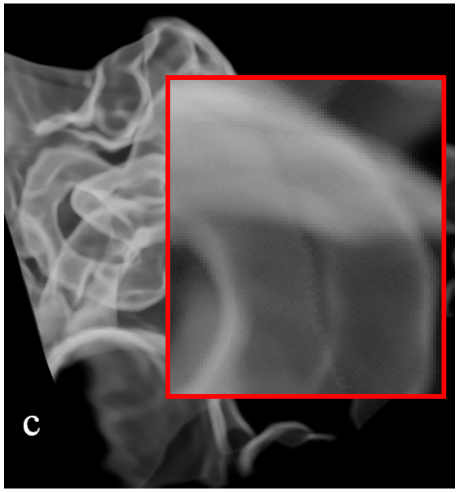
Pre- vs. post-classification



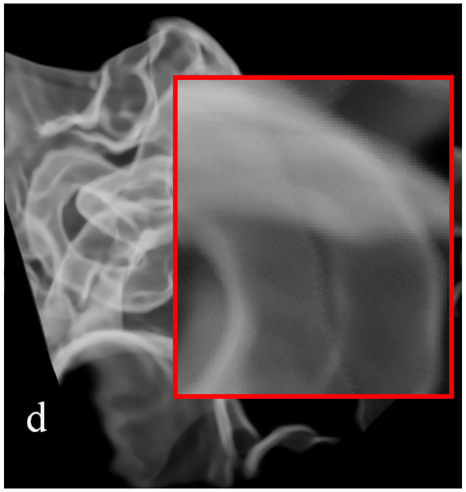
128 slices pre-classification



128 slices post-classification



284 slices post-classification



128 slices preintegrated

Image credit: K. Engel, U. Stuttgart

Preintegration

Idea (Engel 2001):

- simulate infinitely many interpolated samples between two successive samples $s_i = s(\mathbf{x}_i)$ and $s_{i+1} = s(\mathbf{x}_{i+1})$
- assuming:
  - field $s(\mathbf{x})$ varies linearly between samples
  - transfer functions don't depend on derivatives

The discrete formula for opacity at a sample was

$$\alpha_i = 1 - e^{-\tau_i \triangle x}$$

The continuous version, for a sample interval [x_i, x_{i+1}], is

$$\alpha_i = 1 - e^{-\int_{x_i}^{x_{i+1}} \tau(s(x))dx}$$

Assuming now $s(\mathbf{x})$ to be linear between samples, we get

$$\alpha_i = 1 - e^{-\frac{d}{s_{i+1}-s_i}\int_{s_i}^{s_{i+1}} \tau(s)ds}$$ with $$d = \left\| \mathbf{x}_{i+1} - \mathbf{x}_i \right\|$$

which is called a preintegrated opacity transfer function.

The integral

$$\int_{s_i}^{s_{i+1}} \tau(s)ds = \int_0^{s_{i+1}} \tau(s)ds - \int_0^{s_i} \tau(s)ds$$

can be evaluated by two lookups in a precomputed table of

$$\int_0^{s} \tau(s')ds'$$

Alternatively, the preintegrated opacity TF could be tabulated for all possible combinations of (s_i, s_{i+1}, d), especially if the sampling distance d is chosen constant.

The composite color of the same interval

$$C_i = \int_{x_i}^{x_{i+1}} \varepsilon\left(s(x)\right) e^{-\int_{x_i}^{x} \tau\left(s(x')\right)dx'} dx$$

simplifies for linear $s(\mathbf{x})$ to

$$C_i = \frac{d}{s_{i+1} - s_i} \int_{s_i}^{s_{i+1}} \varepsilon\left(s\right) e^{-\frac{d}{s_{i+1}-s_i}\int_{s_i}^{s} \tau(s')ds'} ds$$

which is a preintegrated color transfer function.

Again, it can be tabulated for all combinations of (s_i, s_{i+1}, d), or it can be approximately calculated from tabluated values of

$\int_0^s \tau(s')ds'$ and $\int_0^s \varepsilon(s')ds'$ (moving the exponential term out of the integral).

Raycasting hardware

Today's GPUs are well suited for volume rendering.

Object-space methods are more straight-forward, but also some raycasting implementations on GPUs have been done.

In contrast, two examples of special raycasting hardware:

- VIZARD (1997, U. Tübingen):
  - FPGA-based system
  - long pre-processing
  - 10 Hz for 256^3 volume
  - perspective view raycasting

Raycasting hardware

volumePRO (1999, MERL):

- PCI board
- no pre-processing
- 30 Hz for 256^3 volume
- orthographic raycasting
- ray templates
- shear-warp method
- *z*-supersampling

Example: with / without supersampling

ASIC chip of VolumePRO vg500 board has 4 parallel pipelines with stages:

- interpolation
- gradient estimation
- classification/shading
- compositing

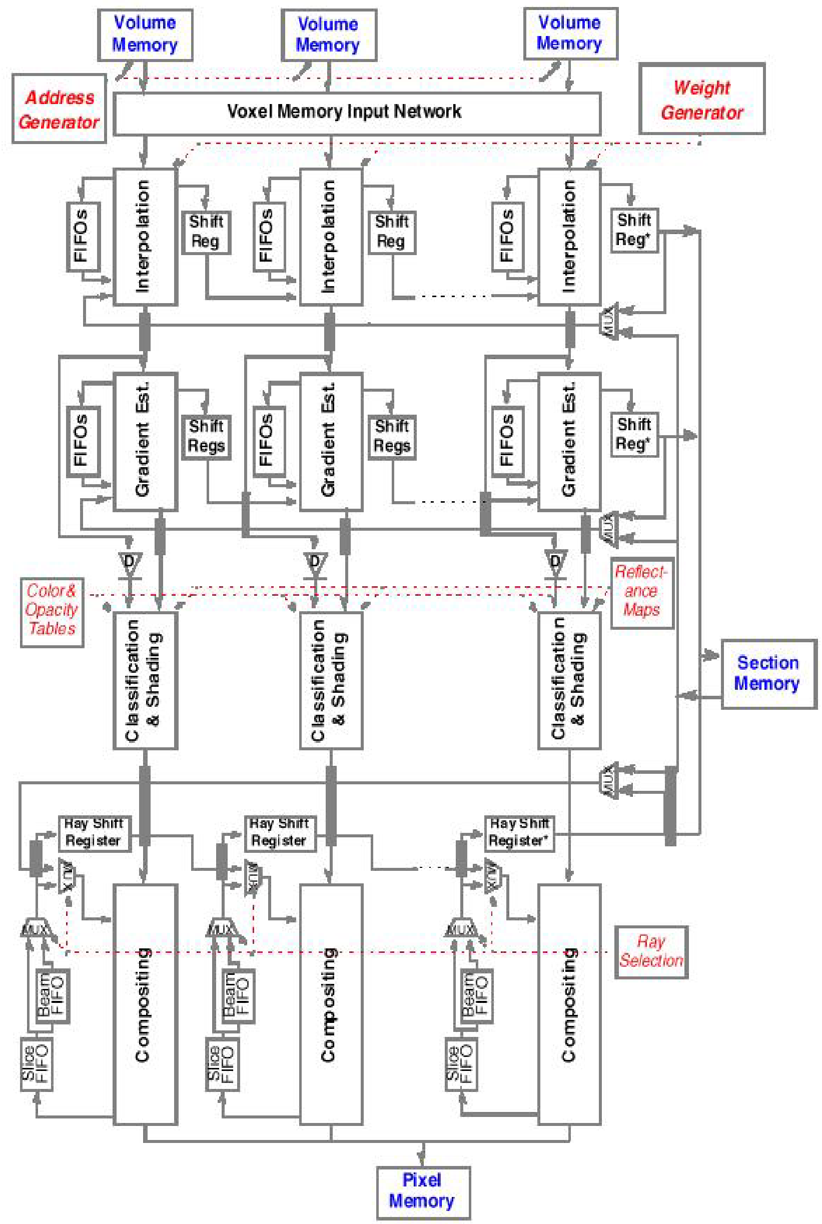
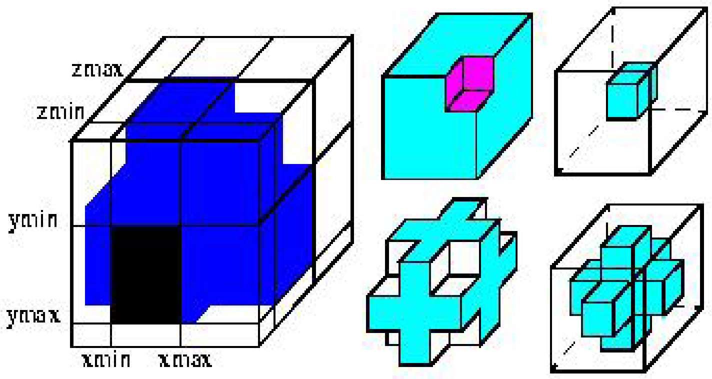


Image is rendered in a single pass through the volume.

Off-chip memory for slices (shown in blue)

Some special features:

- cropping
- clipping planes



some possible combinations of cropping and clipping planes