

Volumetric Algorithms

Cmpt 767 Visualization

Steven Bergner

sbergner@sfu.ca



[based on slides by A. C. Telea]

Volume visualization (Chapter 10)

1. Motivation

- how to see through 3D scalar volumes?

2. Methods and techniques

- ray function (MIP, average intensity, distance to value, isosurface)
- classification
- compositing
- volumetric shading

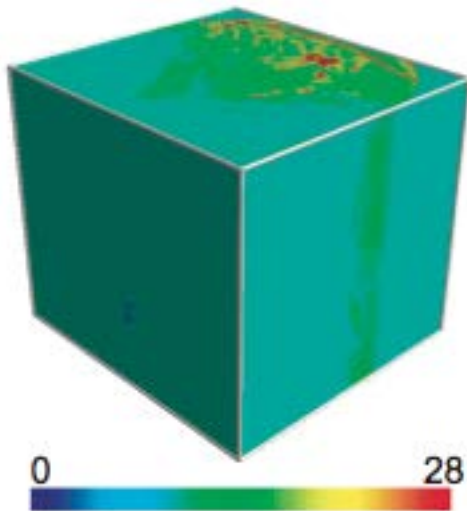
3. Advanced points

- sampling and interpolation
- classification and interpolation order
- performance issues

Motivation

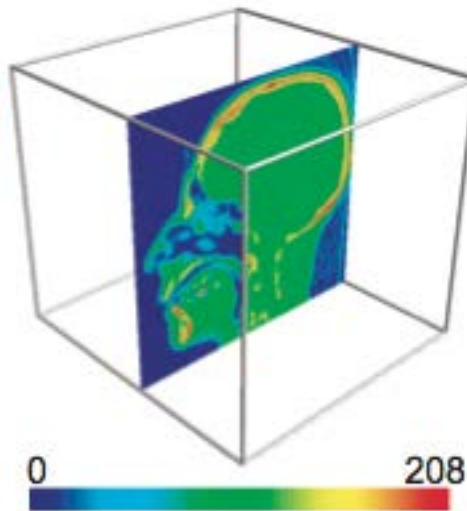
Scalar volume $s : \mathbf{R}^3 \rightarrow \mathbf{R}$

How to visualize this?



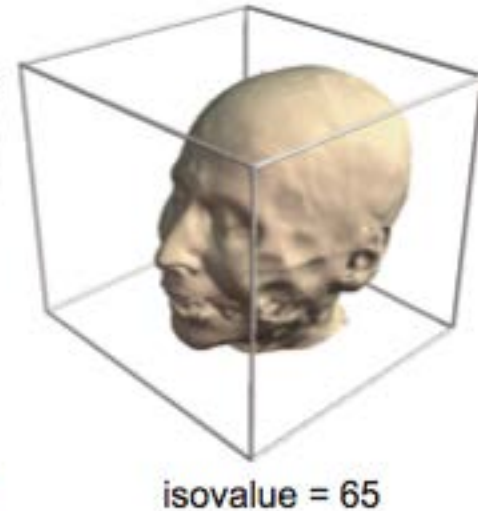
direct color mapping

- see only outer surface



slicing

- all details on slice
- no info outside slice



contouring

- all details on contour
- no info outside contour

How to visualize this so we see through the volume

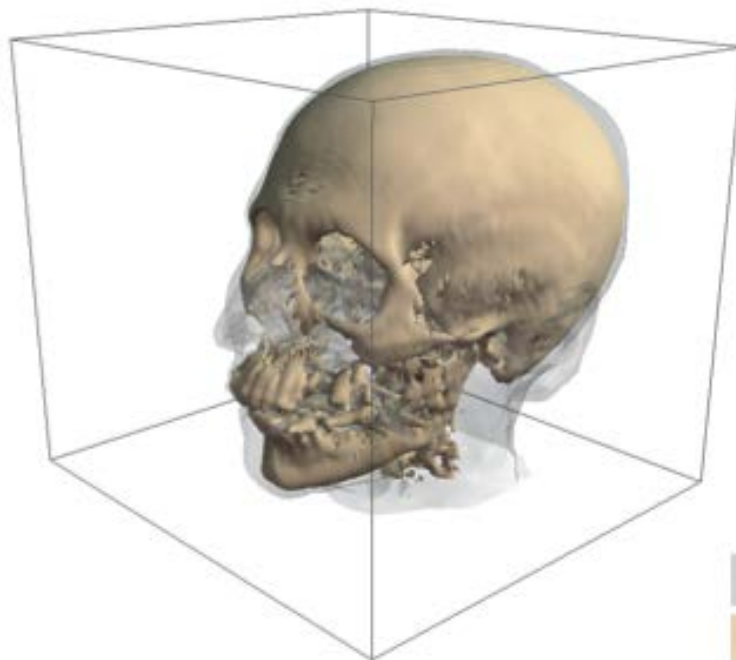
Seeing through a volume

Idea

- use known techniques (slices and contours)
- use transparency

First try

- draw several contours C_i for several values s_i
- transparency α_i proportional to scalar value s



We start seeing a little bit through the volume...

...But this won't work for too many contours (**why?**)

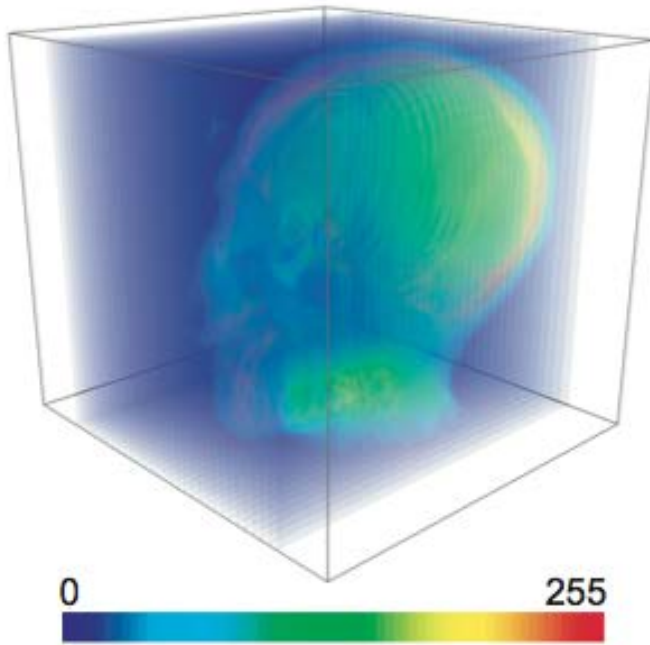
isovalue = 65
isovalue = 127

Seeing through a volume

Second try

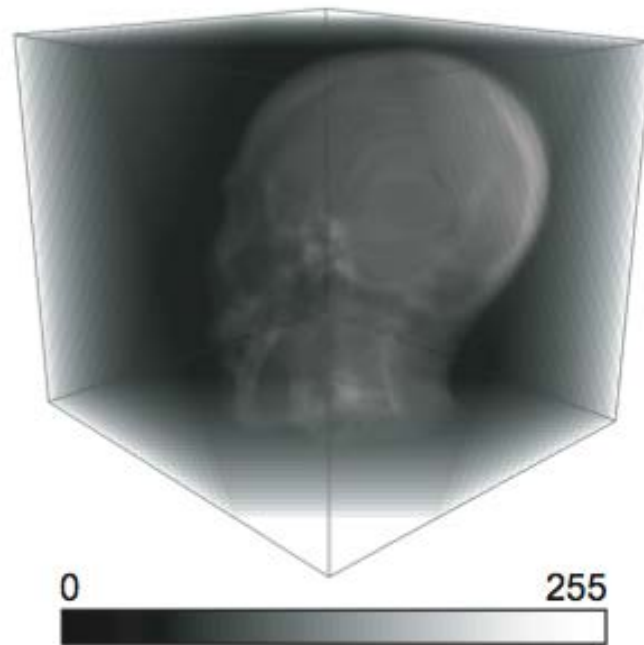
- draw several parallel slices S_i
- transparency α_i inversely proportional to number of slices

$$\alpha_i = \frac{1}{\|S\|}$$



axis-aligned slices

- not OK if we view volume across slicing direction



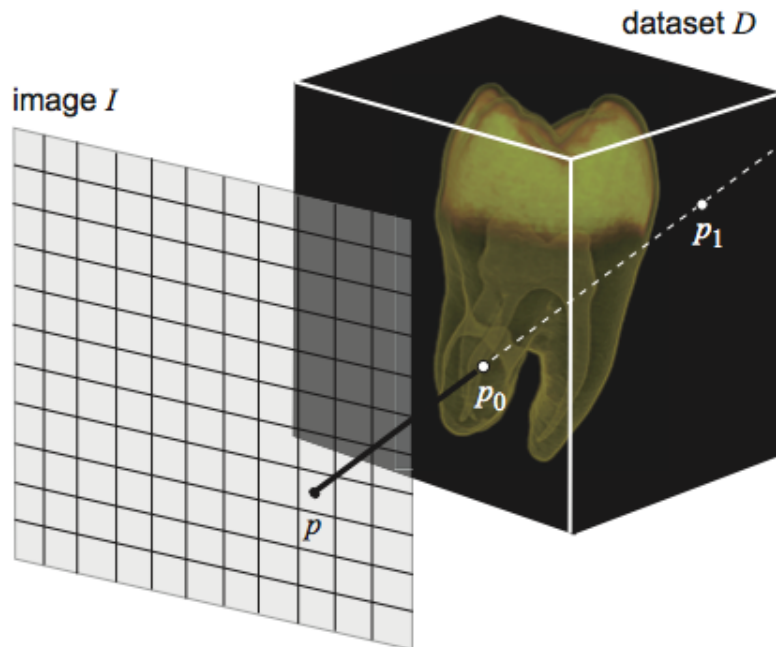
view direction-aligned slices

- any viewing direction OK
- must reslice when changing viewpoint

Volume rendering basics

Main idea

- consider a scalar signal $s : D \rightarrow \mathbf{R}$ to be drawn on the screen image I
- for each pixel $p \in I$
 - construct a ray \mathbf{r} orthogonal to I passing through p
 - compute intersection points p_0 and p_1 of \mathbf{r} with D
 - express $I(p)$ as function of s along \mathbf{r} between p_0 and p_1



1. Parameterize ray

$$p(t) = (1-t)p_0 + tp_1, \quad t \in [0,1]$$

1. Compute pixel color

$$I(p) = f(H(s(t))), \quad t \in [0,1]$$

transfer function

ray function

Volume rendering

Define a ray function

$$F : \{s(t) \mid t \in [0, 1]\} \rightarrow \mathbf{R}$$

all scalar values along ray

a single resulting scalar value

The ray function 'aggregates' all scalar values along a ray

Next, define a transfer function

$$f : \mathbf{R} \rightarrow [0, 1]^4$$

a single scalar value

an RGBA color

- same concept as color mapping (see Module 2)

Idea

- ray function: says how to combine all scalar values along a ray into a single value
 - transfer function: says how to map a single scalar value to a color
 - The process of computing all rays for an image I is called ray casting
-

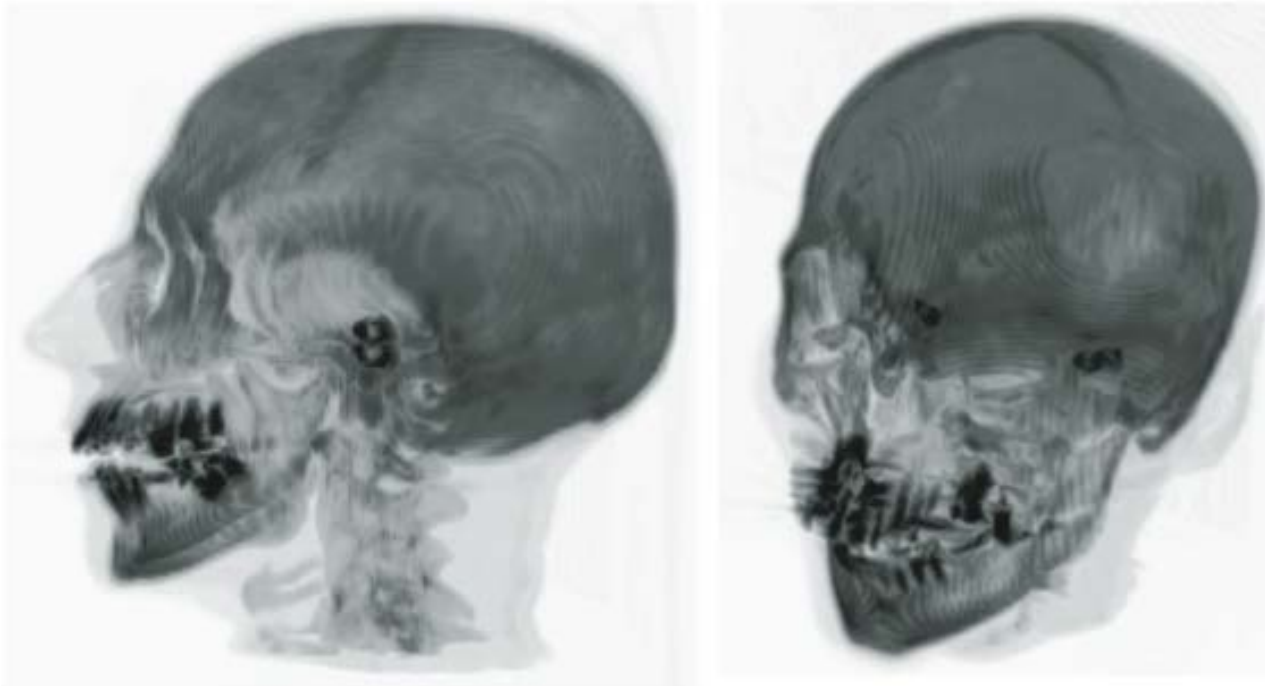
Maximum intensity projection (MIP)

First example of ray function

- find maximum scalar along ray, then apply transfer function to its value

$$I(p) = f\left(\max_{t \in [0, T]} s(t)\right)$$

- useful to emphasize high-value points in the volume



Example
MIP of human head CT

- white = low density (air)
- black = high density (bone)

OK, but gives no depth cues

Average intensity projection

Second example of ray function

- compute average scalar along ray, then map it to color

$$I(p) = f \left(\frac{\int_{t=0}^T s(t) dt}{T} \right)$$

- useful to emphasize average tissue type (e.g. density in a CT scan)



maximum intensity projection



average intensity projection

Example Human torso CT

- black = low density (air)
- white = high density (bone)

Average intensity projection
is equivalent to an X-ray

Distance to value function

Third example of ray function

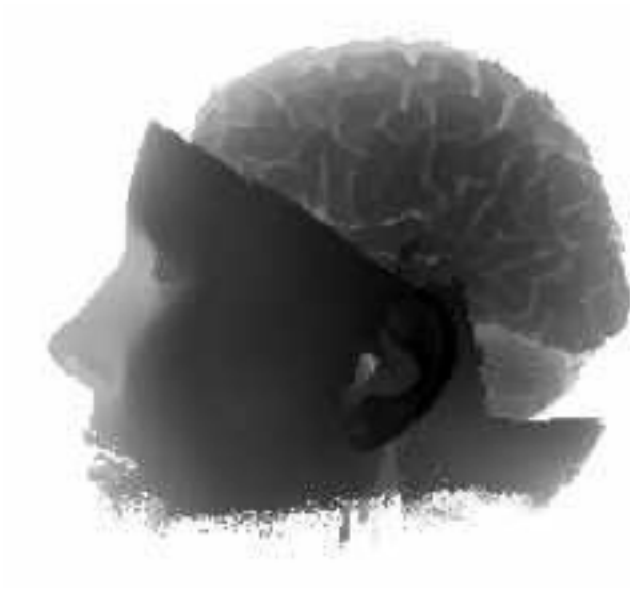
- compute distance along ray until a specific scalar value σ

$$I(p) = f \left(\min_{t \in [0, T], s(t) \geq \sigma} t \right)$$

- useful to emphasize depth where some specific tissue is located



distance to value 20



distance to value 50

Example
Human head CT

- black = low distance
- white = high distance

Isosurface function

Fourth example of ray function

- compute whether a given isovalue σ exists along ray

$$I(p) = \begin{cases} f(\sigma), & \exists t \in [0, T], s(t) = \sigma, \\ I_0, & \text{otherwise.} \end{cases}$$

- produces same result as marching cubes, but with a higher accuracy



isosurface
(marching cubes)



isosurface
(software ray casting)



isosurface
(hardware ray casting)

Composite function

Fifth example of ray function

- compute a color at each point along the ray (apply transfer function *first*)
- blend (compose) all colors to get the final pixel color (ray function=alpha blending)

$$I(p) = F(\{f(s(t) | t \in [0, 1]\})$$

transfer function (applied to all pixels along ray)
ray function (blends all colors produced by transfer function along ray)

Idea

- **transfer function**: controls color+transparency of all material types
- **ray function**: blends together all material colors+transparencies along ray
- most powerful (but most computationally expensive) ray function
- allows huge range of effects (depending on type of transfer function)
- designing 'good' transfer functions is however non-trivial

Volumetric shading

Shading

- is required if we compute e.g. isosurfaces
- but can also be useful for composite ray function

Method

- instead of simply using the colors $I(t) = f(s(t))$
- composite the shaded colors (see Chapter 2, Phong lighting)

$$I(t) = c_{\text{amb}} + c_{\text{diff}}(t) \max(-\mathbf{L} \cdot \mathbf{n}(t), 0) + c_{\text{spec}}(t) \max(-\mathbf{r} \cdot \mathbf{v}, 0)^\alpha$$

How to implement

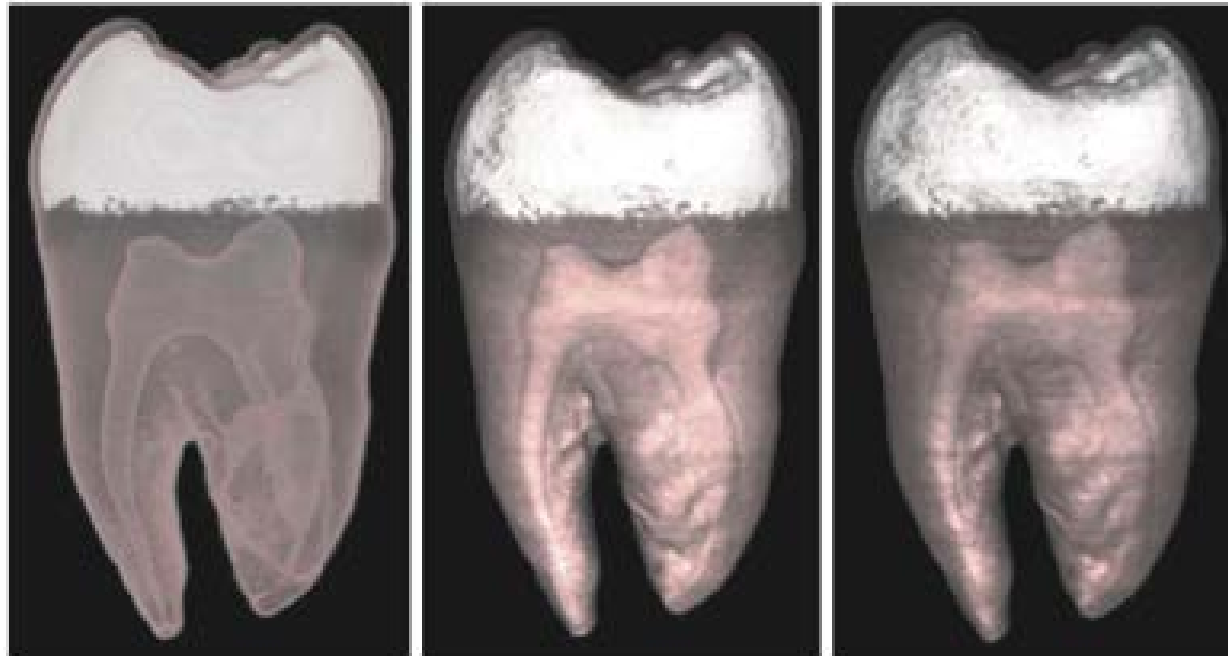
- lighting coefficients c and light vector \mathbf{L} : user sets them as desired
- surface normal \mathbf{n} : compute from gradient of scalar value

$$\nabla s(t) = \frac{\partial s(t)}{\partial x} + \frac{\partial s(t)}{\partial y} + \frac{\partial s(t)}{\partial z}$$

(we did the same for isosurfaces, see Module 3)

Volumetric shading

Results



no shading

diffuse lighting

diffuse and specular
lighting

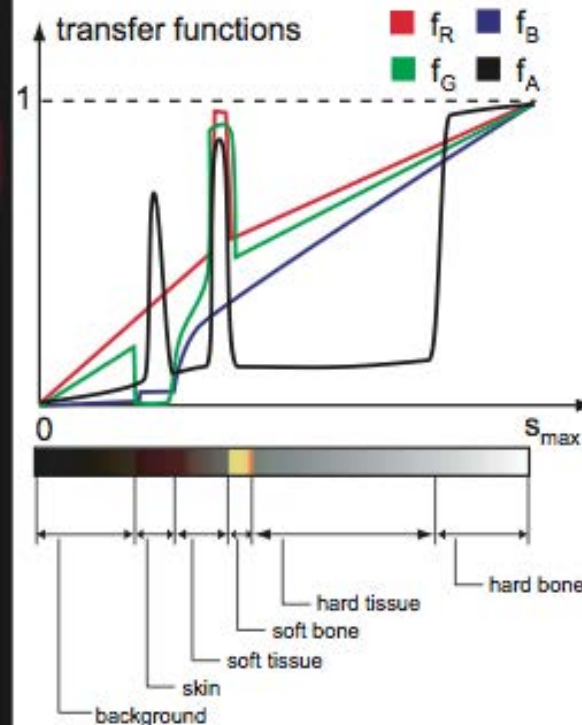
Shading

- gives very good cues of depth and shape structure
 - is quite cheap and simple to compute
-

Transfer functions

Extremely powerful modeling tool (mainly when using composite ray function)

- design four functions f_R, f_G, f_B, f_A
- use color and transparency to emphasize desired material properties (e.g. tissue type)
- use any ray function described so far

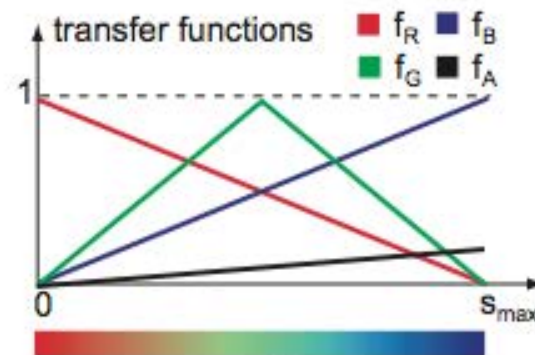
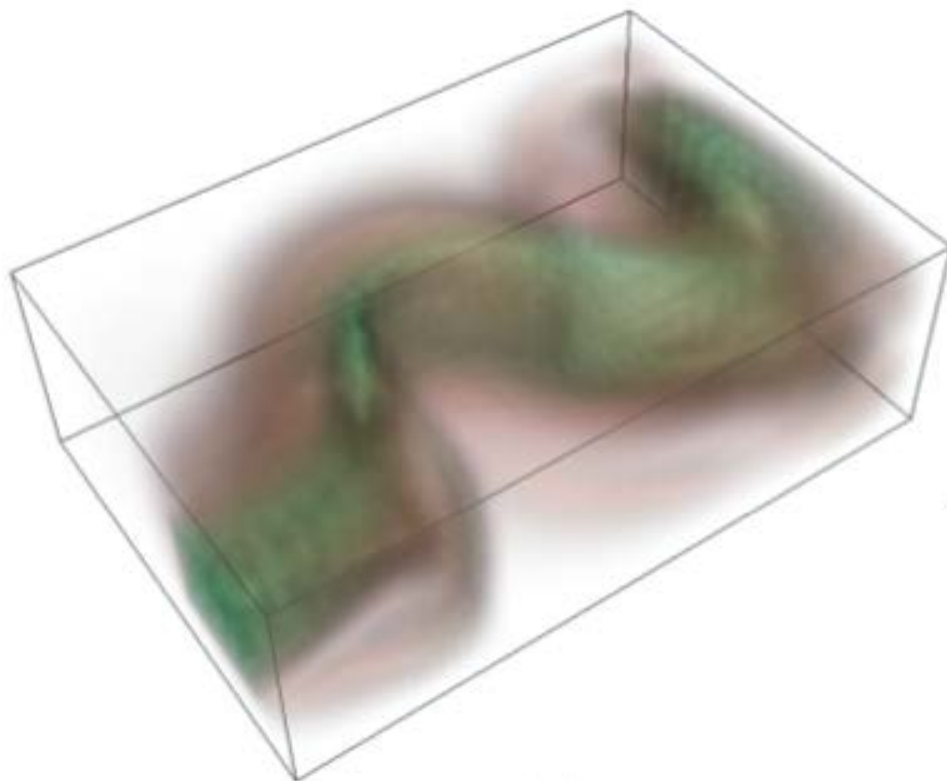


Example
Human head CT

- emphasize bone
- show also muscles

Transfer functions

Volume rendering can be used to visualize also **vector datasets**



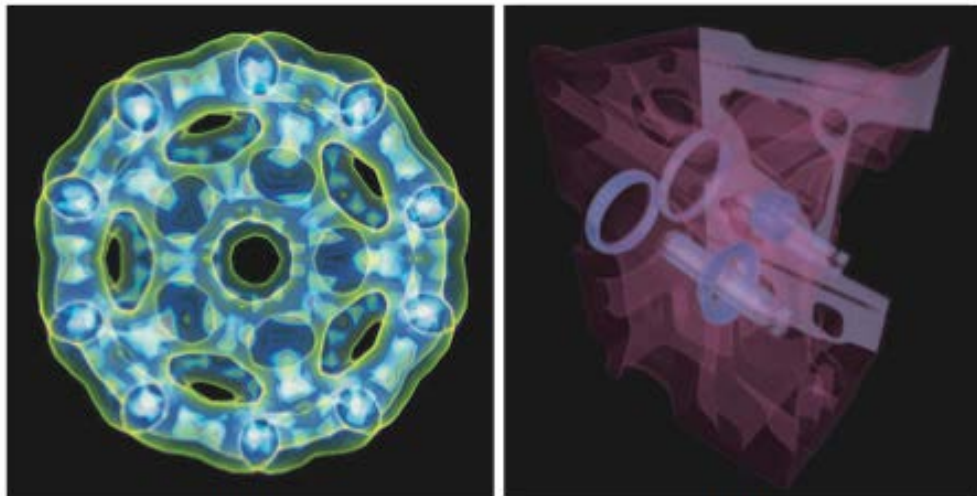
Volume rendering of fluid flow vector field magnitude

- red = slow flow
- green = more rapid flow
- blue = fastest flow

Question: why is the blue hard to see?

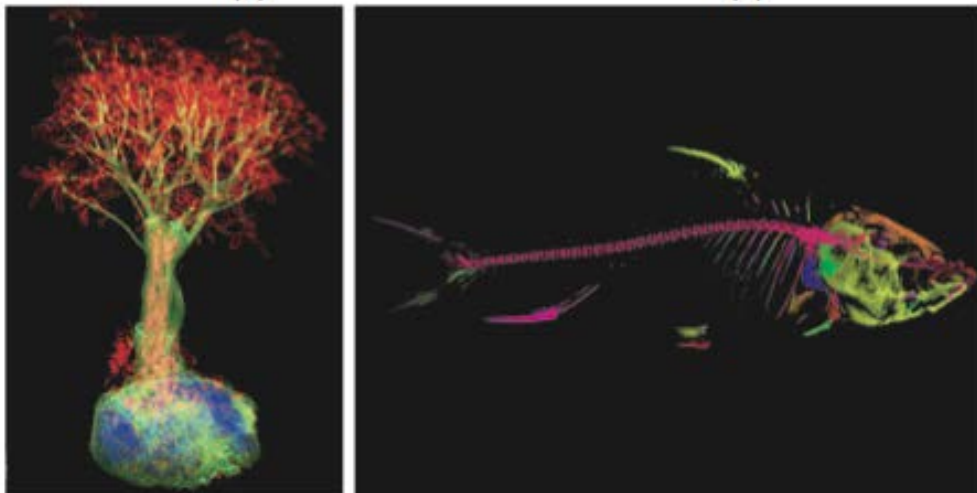
Transfer functions

Further examples of (artistic) volume rendering



(a)

(b)



- a) electron density
- b) car engine part
- c) bonsai tree (scanned)
- d) fish

Transfer functions

...and some extreme examples of volume rendering



Volume rendering of human MRI dataset

- shading: mimics natural lighting
- backdrop added for extra effect

Beautiful result...

...but not directly usable by e.g. clinicians

Implementation issues

Sampling density

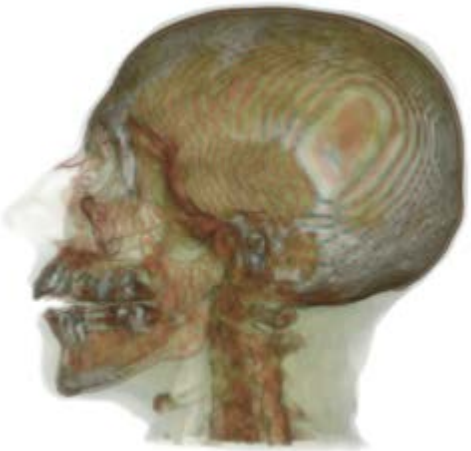
- recall the ray parameterization $q(t) = (1-t)q_0 + tq_1$, $t \in [0,1]$
- we need to sample along the ray (e.g. integrate, compute min/max, etc)
- how small should we take the sampling step $\delta = dt$?



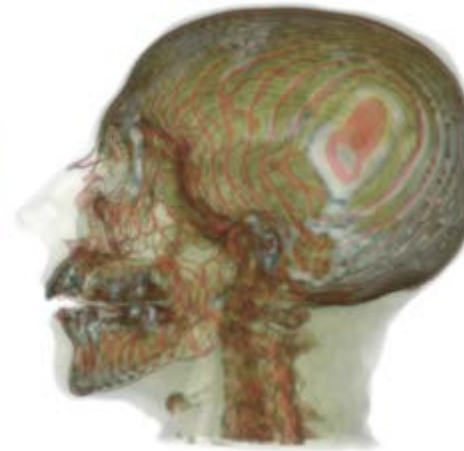
(a) $\delta = 0.1$



(b) $\delta = 0.5$



(c) $\delta = 1.0$



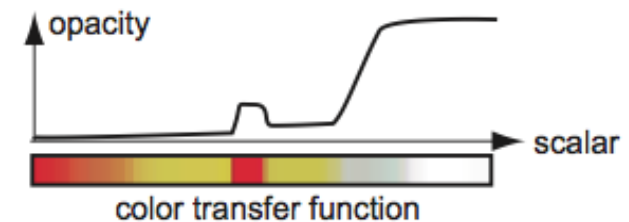
(d) $\delta = 2.0$

Human head CT, four different δ values

- smaller δ : more accuracy
- too small δ : slow rendering

Practical guideline

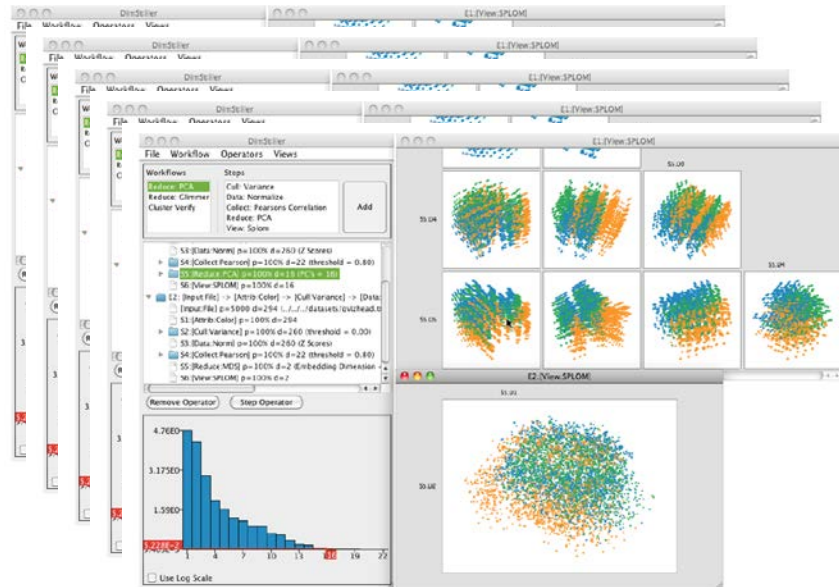
- δ should never exceed a voxel size (otherwise we skip voxels while traversing the ray...)



Summary

Volume visualization (book Chapter 10)

- Extends classical scalar visualization to 'see through' 3D volumes
 - ray functions and transfer functions
- Evaluation
 - produces highly r
 - requires quite so
 - can be easily acc
 - good transfer fun



CUDA)
nt, hard

Thank you for your interest!