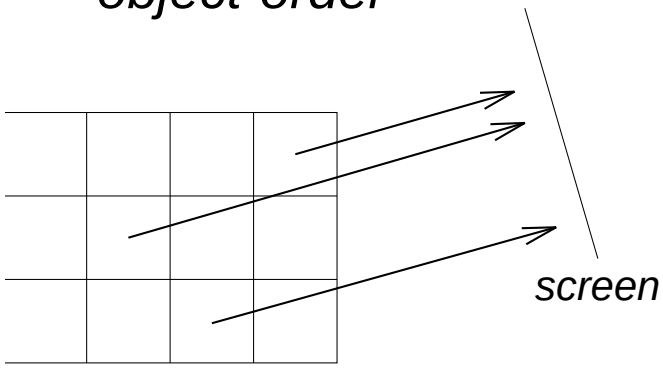
Direct Volume Rendering *Elvins*

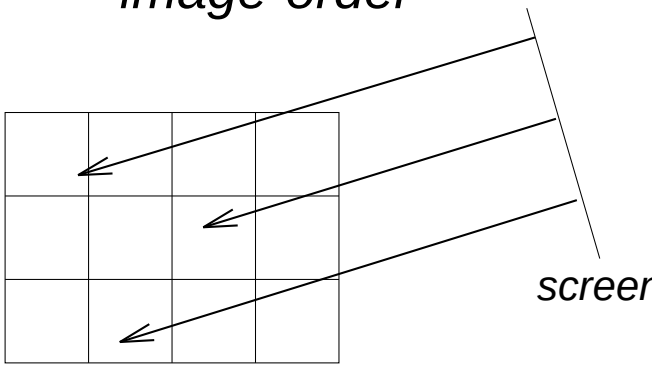
Principle: rendering of scalar volume data with cloud-like, semi-transparent effects

forward mapping
object-order



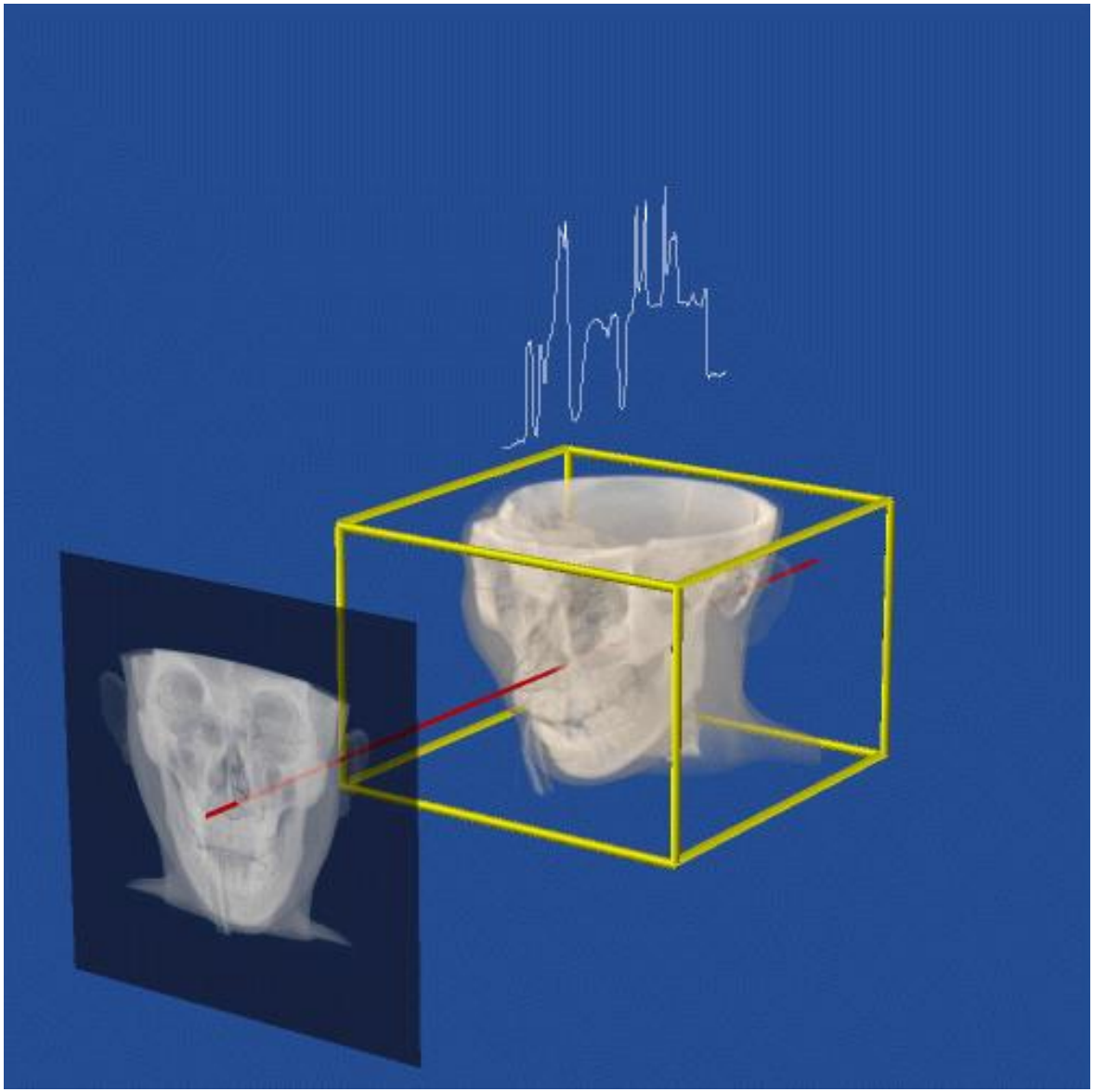
voxel projection
projected tetrahedra
splatting

backward mapping
image-order

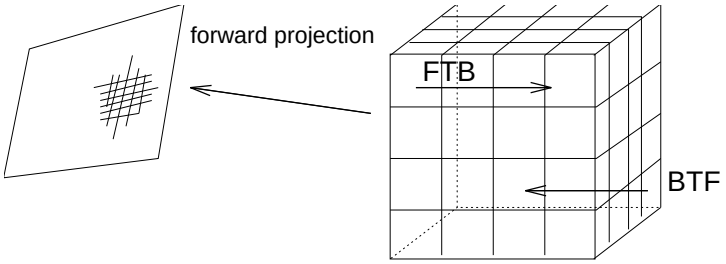


ray casting
integration methods

Direct Volume Rendering



Voxel projection



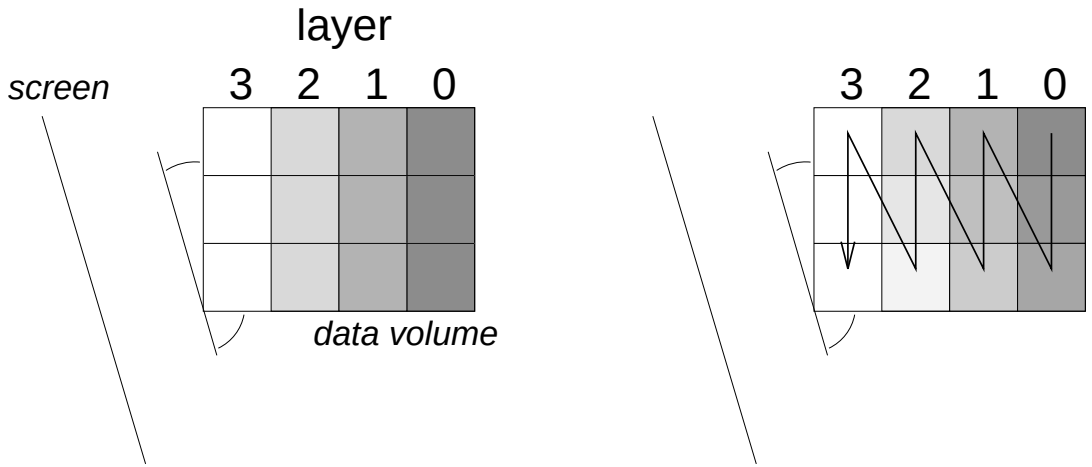
FTB = Front to Back
BTF = Back to Front

For each voxel layer:

for each voxel from layer:

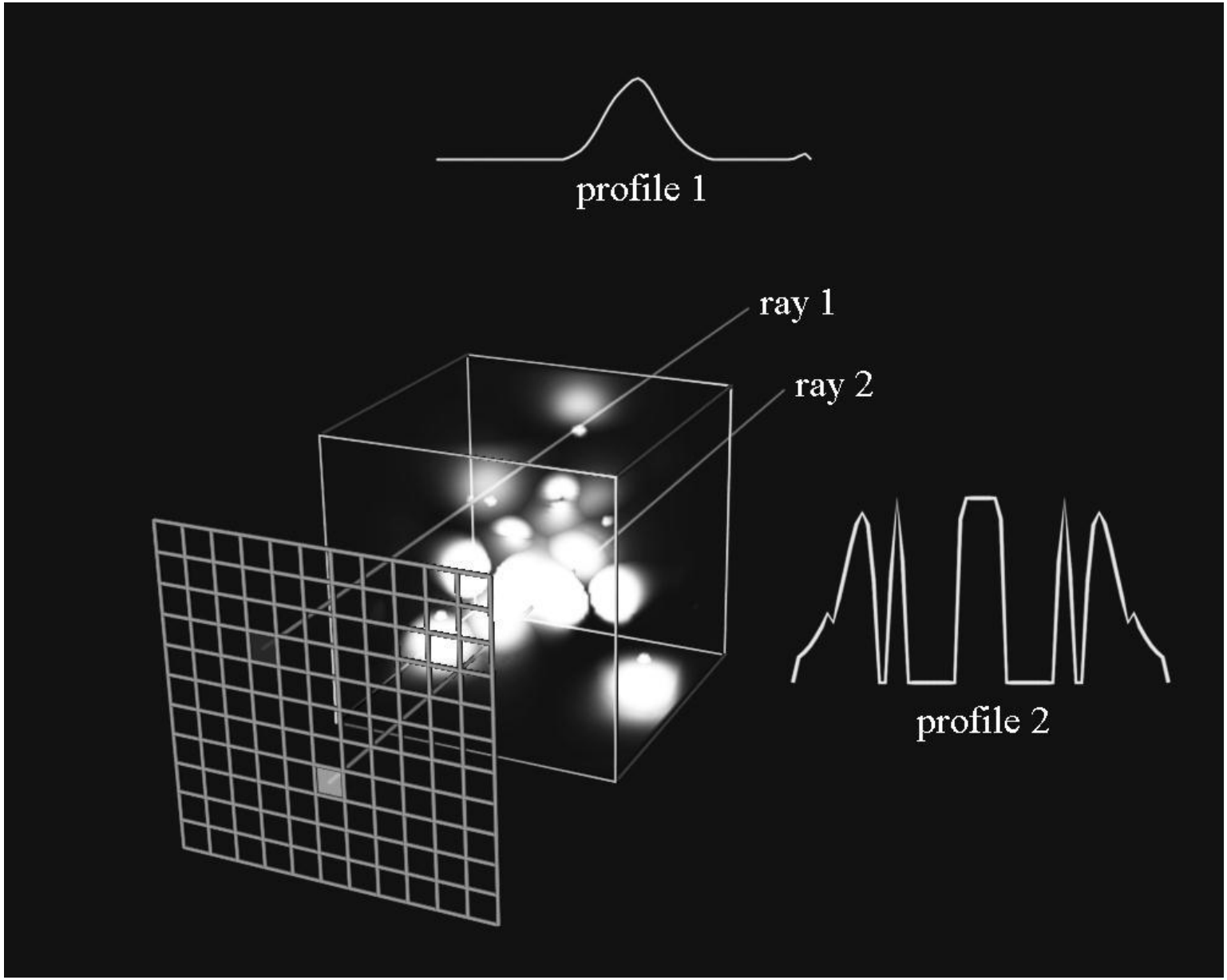
- classify voxel value
- if voxel selected:
  - compute color from classification and shading
  - project to screen

Projection order BTF = Back to Front



- determine axis of data volume with smallest angle with screen
- start with most distant layer
- Order of voxel in layer: start with most distant voxel

Ray casting



Ray casting algorithm

Fire a ray and then:

1. find intersection with voxel and use interpolation to obtain value
2. use transfer function to find colour and opacity
3. compute shading and opacity attenuation (optional)
4. apply weighting formula to combine colour and opacity
5. step along the ray

Composite transfer function

Steps:

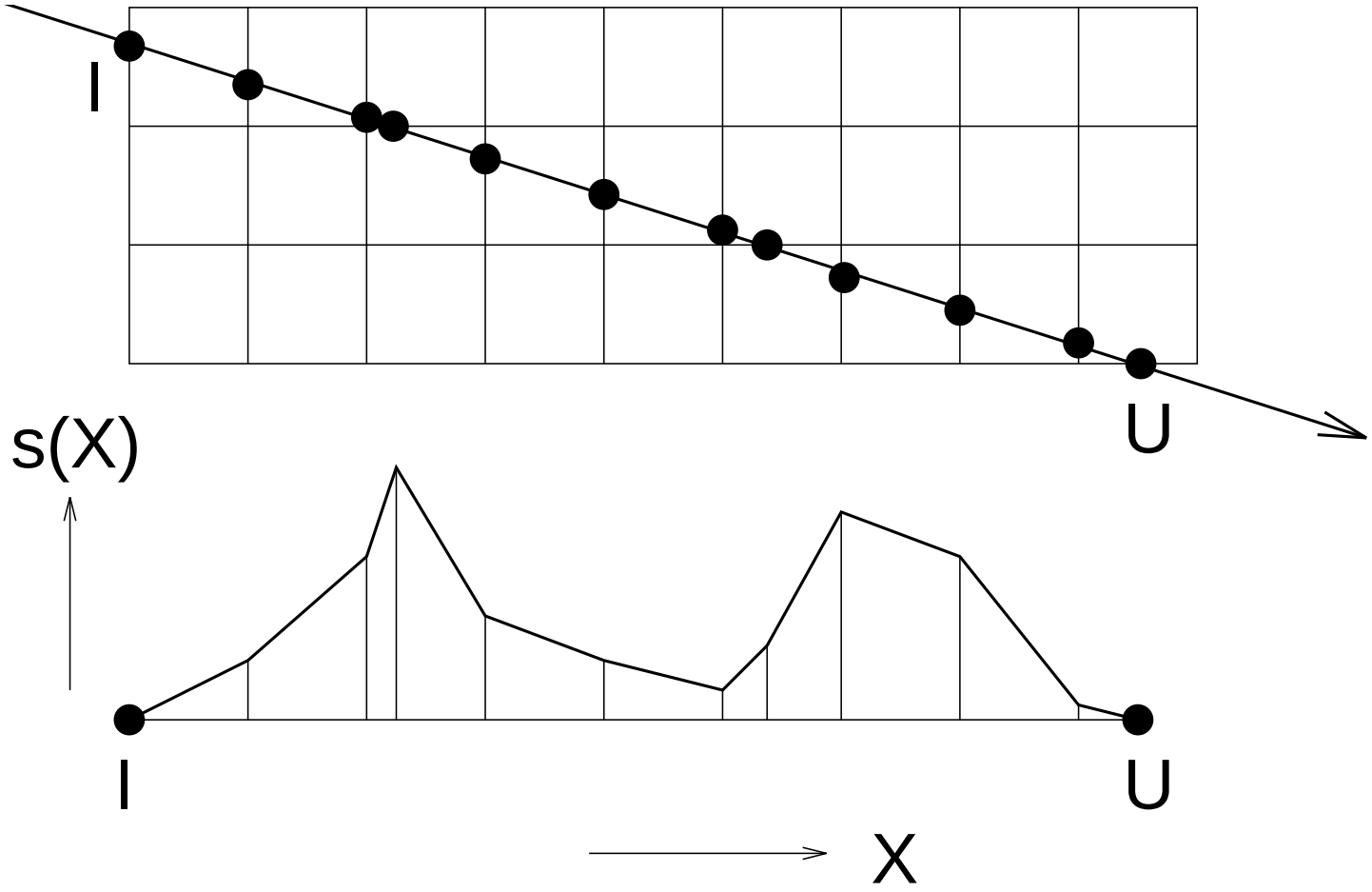
- Classification: determine transfer function.
  TF: map scalar values to colour, opacity and material

- Ray casting

How to determine the transfer function?

- determine what is present in the data and what you want to see
- find the boundaries between materials by thresholding
- make up a value for the opacity (depends on what you want to see)
- make a test image
- repeat the procedure until satisfied

Ray casting



Color merging

Merging of semi-transparent objects in depth order:
C_i, O_i = accumulated color, opacity after adding object i with color c_i and opacity o_i

- front-to-back

  $C_i = (1 - O_{i-1})c_i + C_{i-1}$

  $O_i = (1 - O_{i-1})o_i + O_{i-1}$

- back-to-front

  $C_i = (1 - o_i)C_{i-1} + c_i$

  $O_i = (1 - o_i)O_{i-1} + o_i$

Ray casting: (dis)advantages

\+ high image quality

– dependent on number of pixels

– viewpoint dependent

– computationally intensive

Ray casting options

- thresholding: detection of certain value $\longrightarrow$ iso-value!
- cutting: map values at specific depth
- maximum intensity projection: per ray localize and map maximal value
- visible objects: voxels with certain membership label

Maximum intensity projection (MIP)

✔ works well on noisy data

✘ 3D structure difficult to understand → sequence of images

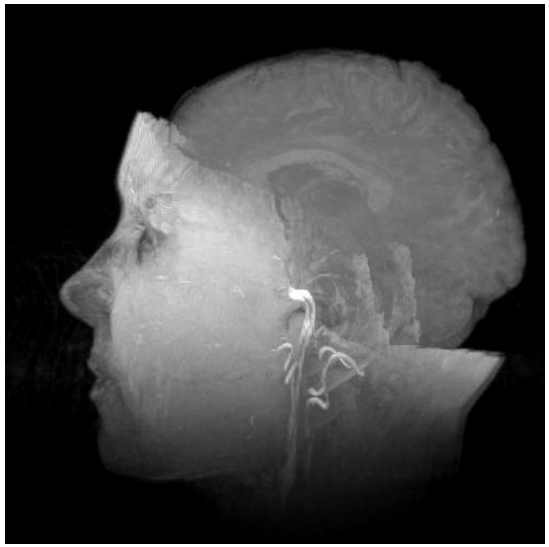
Useful to get an impression of what data looks like.

3D Magnetic Resonance Angiography

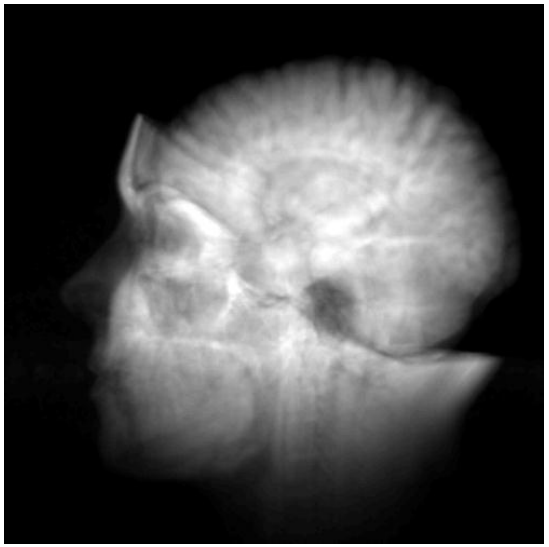
MIP with 'Closest Vessel Projection'. 256x256x148 MRA data (Courtesy: AZU).

Examples of transfer functions

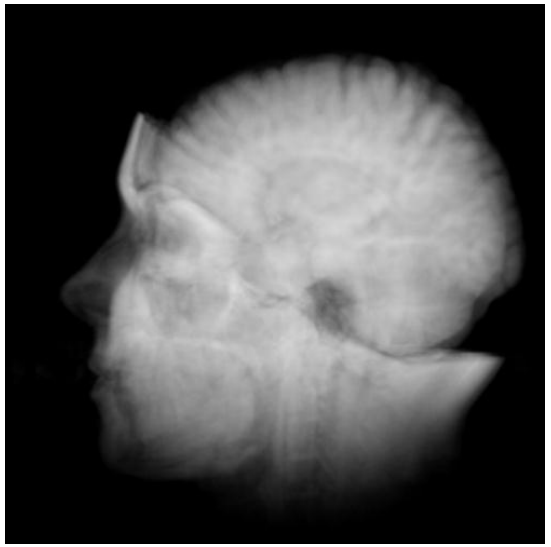
maximum



average



composite



Example: foot data

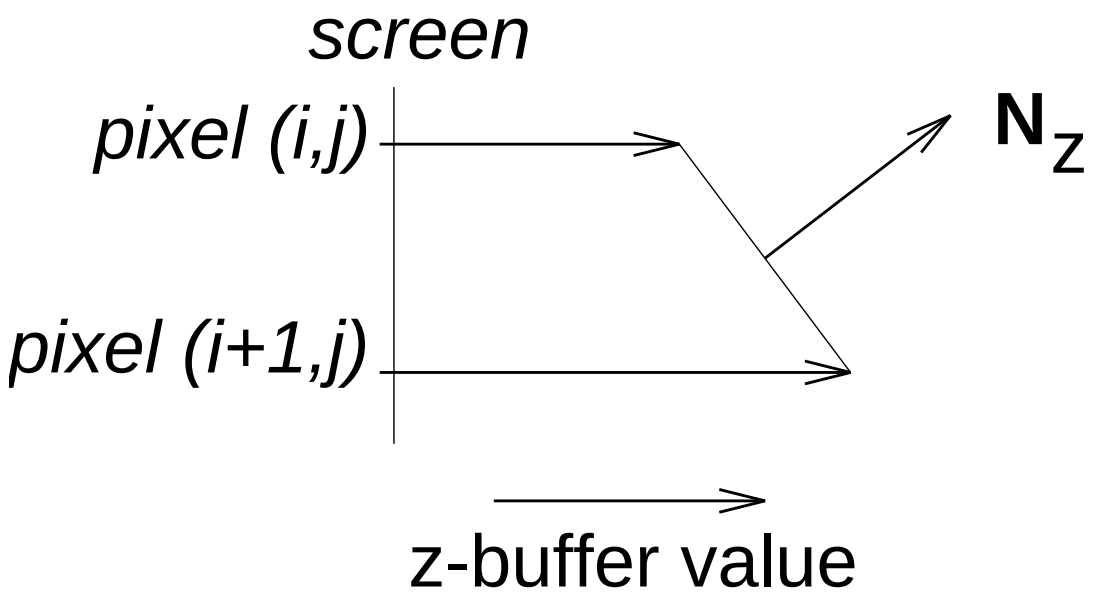
Tissue	Value	Opacity
skin	25–35	50%
bone	70–80	100%

Gradient shading

Principle: determine normal of surfaces in scalar data volumes for shading of semi-transparent clouds

* Z-buffer gradient shading surfaces based on depth-buffer
* Depth value $Z(x, y)$, gradient $\nabla Z = (\frac{\partial Z}{\partial x}, \frac{\partial Z}{\partial y}, 1)$.
* Approximate by:

$w(Z(x, y) - Z(x - 1, y))$ $\quad$ $(w =$ weight function$)$



Gradient shading

N = normal of surface for determination of color

(in the sense of shading)

$|N|$ = magnitude of surface for computation of opacity

(in the sense of presence, visibility)

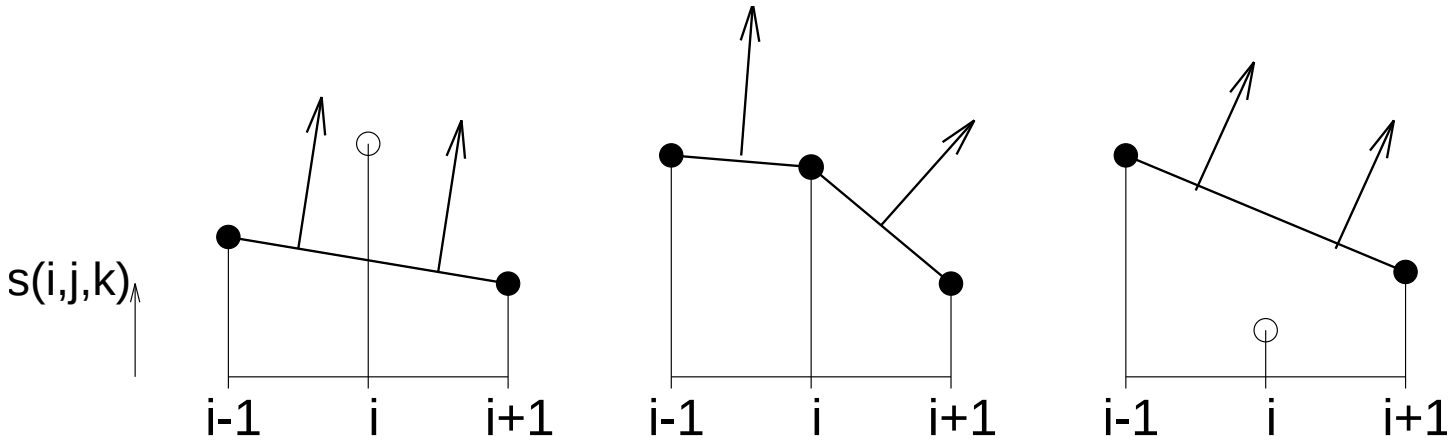
Grey-level gradient shading

Principle: surfaces based on data itself: density surfaces

$$N_s = \nabla s \approx \begin{pmatrix} s(i+1,j,k) - s(i-1,j,k) \\ s(i,j+1,k) - s(i,j-1,k) \\ s(i,j,k+1) - s(i,j,k-1) \end{pmatrix}$$

Unshaded vs. shaded

Adaptive gradient shading



Goal: removing small artifacts, like sharp edges

(Dis)advantages of shading

✘ May interfere with interpretation

MIP: dark $\longrightarrow$ low values in scalar data

shaded: dark $\longrightarrow$ low values in scalar data or gradient points away from light source

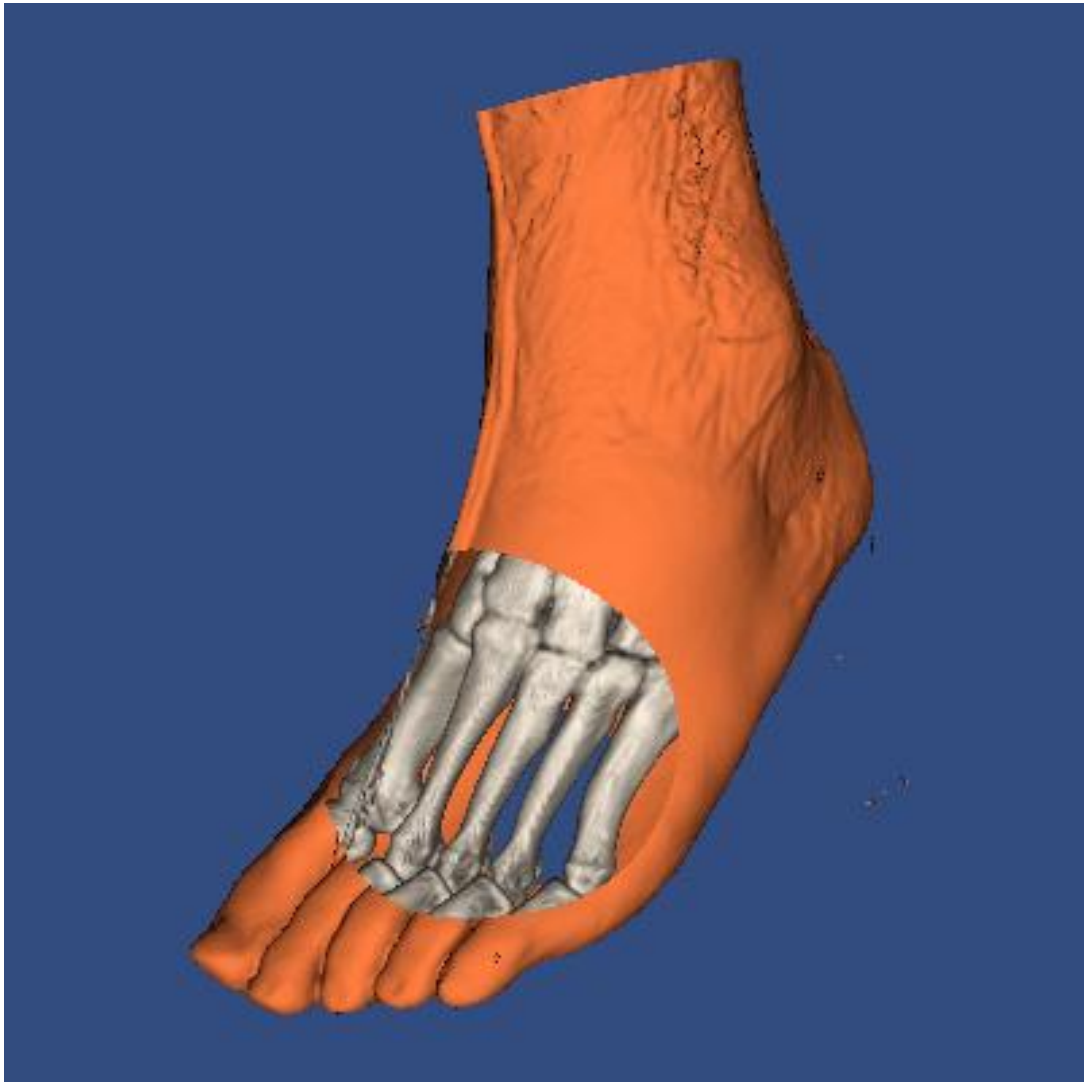
✔ 3D structure in most cases clear (occlusion, lighting, motion parallax)

Advanced volume visualization

Combining surface rendering with ray casting.
Steps involved:

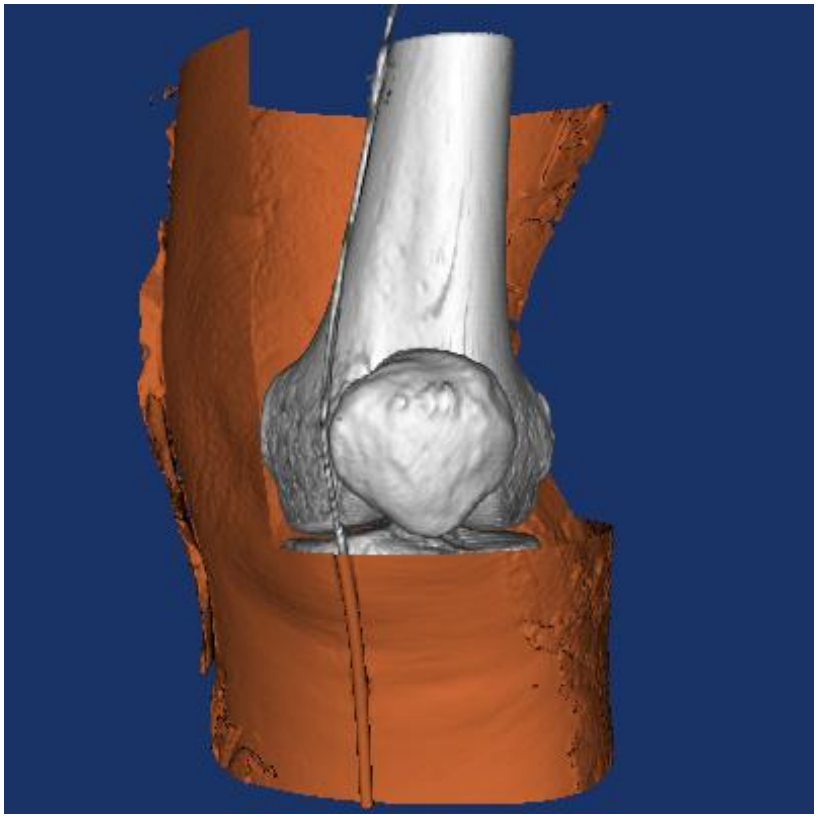
1. render geometry
2. use depths in Z-buffer as start and end points of rays
3. perform ray casting
4. combine results into one image

Combined surface rendering – ray casting



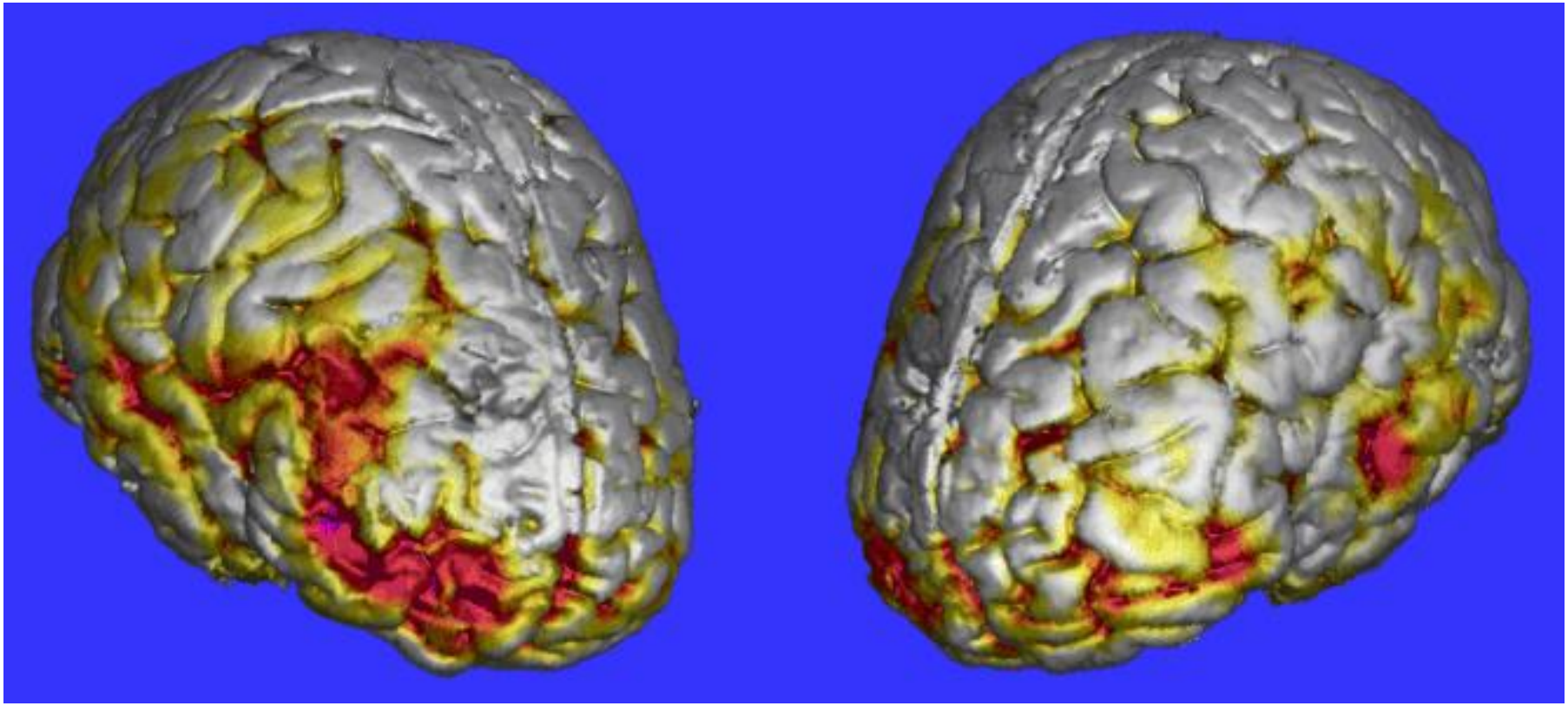
Mpeg movie: skin: surface rendering bones: ray casting

Combined surface rendering – ray casting



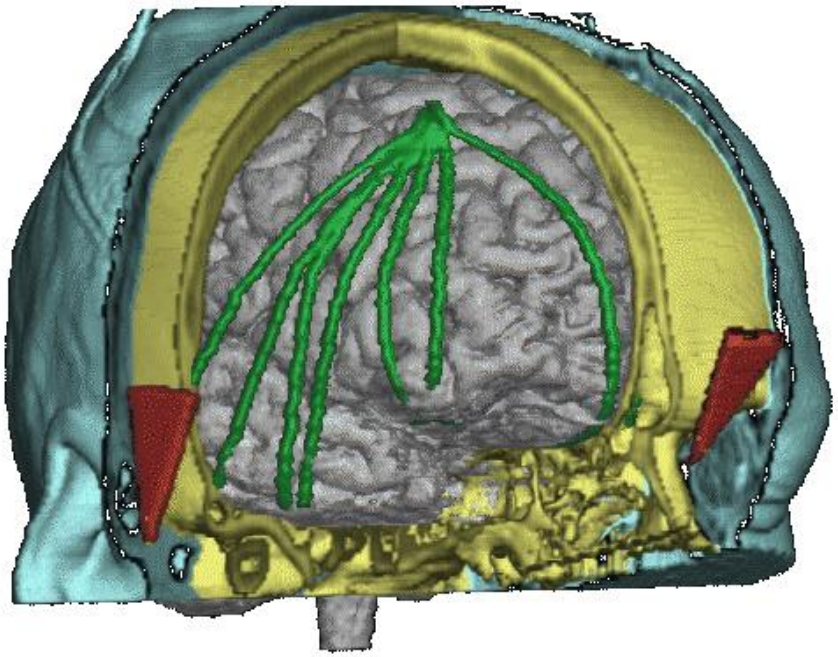
Data: visible human
(www.nlm.nih.gov/research/visible/visible_human.html)

Multimodality 3D rendering



SPECT functional data color coded on MR anatomical data (courtesy, AZU).

Multimodality 3D rendering



Cortical surface and the skin from MR data, bone and cortical surface electrode from CT data (courtesy, AZU).