Volume visualization I *Elvins*

- surface fitting algorithms
  - ⋆ marching cubes
  - ⋆ dividing cubes
- direct volume rendering algorithms
  - ⋆ ray casting, integration methods
  - ⋆ voxel projection, projected tetrahedra, splatting
- hybrid rendering algorithms

Background & Motivation

- Large three-dimensional (3D) data sets arise from measurement by physical equipment, or from computer simulation.
- Scientific areas: computerized tomography (CT), astronomy, computational physics or chemistry, fluid dynamics, seismology, environmental research, non-destructive testing, etc.
- For easy interpretation volume visualization techniques are useful:
  - ⋆ view data from different viewpoints.
  - ⋆ interactive exploration in Virtual Environments.

Requirements

- Compression/simplification: visualize reduced version of data in controllable way.
- Progressive refinement: incremental visualization from low to high resolution.
- Progressive transmission: transmit data incrementally from server to client's workstation (data transfer is time-limiting factor)
- Level-of-detail (LOD): use low resolution for small, distant or unimportant parts of the data.

Volume rendering integral

- Transport of light is modelled by equations originating from physics.

- low albedo approximation for the intensity $I(\mathbf{x}, \mathbf{s})$ at position $\mathbf{x}$ integrated along the line $\mathbf{x} + t\mathbf{s}$, $t_0 \leq t \leq t_l$:

$$\boxed{I(\mathbf{x}, \mathbf{s}) = \int_{t_0}^{t_l} f(\mathbf{x} + t\mathbf{s}) e^{-\int_{t_0}^{t} \alpha(\mathbf{x}+u\mathbf{s})\mathrm{d}u} \mathrm{d}t.}$$

where t_0 is the point of entrance, and t_l the point of exit. α is the opacity (related to the density of the particles).

X-ray rendering

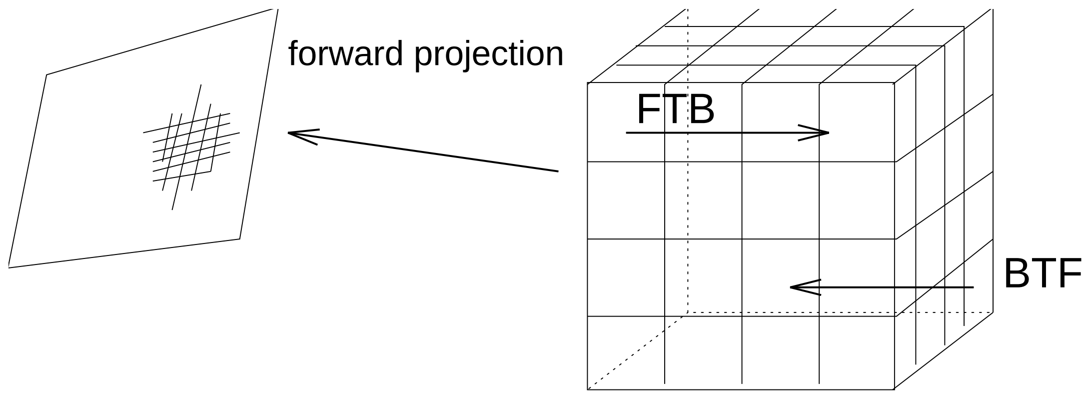
Further simplification: $\alpha = 0$.

$$\boxed{I(\mathbf{x}, \mathbf{s}) = \int_{t_0}^{t_l} f(\mathbf{x} + t\mathbf{s})\mathrm{d}t.}$$

Volume visualization II

- surface rendering
  reduce volume to isosurfaces $S(c) : f(x, y, z) = c$ of a density function $f(x, y, z)$ representing the boundary between materials.

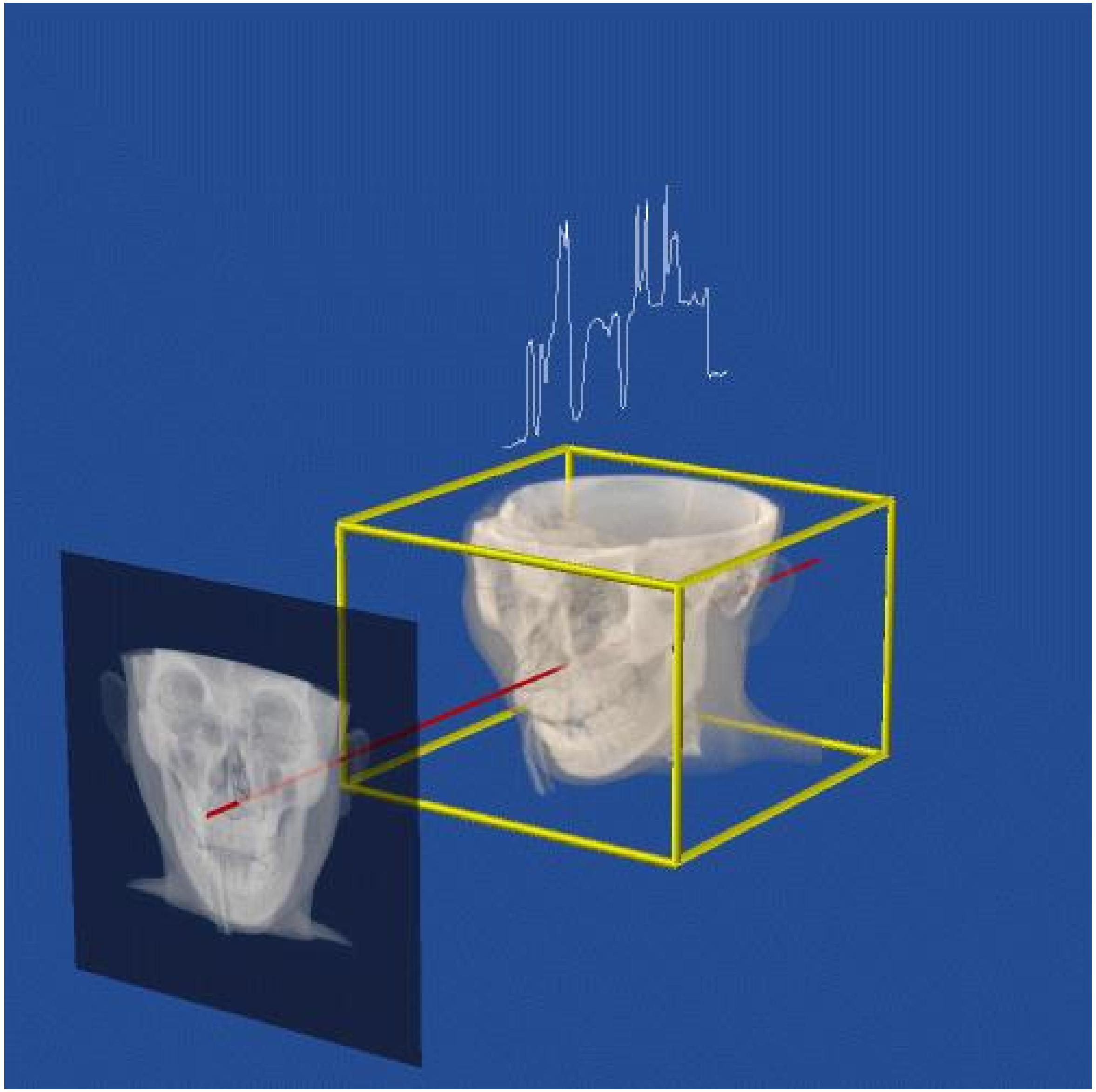
- direct volume rendering
  map volume data directly on screen (no graphical primitives) with semi-transparent effects



Direct Volume Rendering

X-ray rendering of human head CT data.

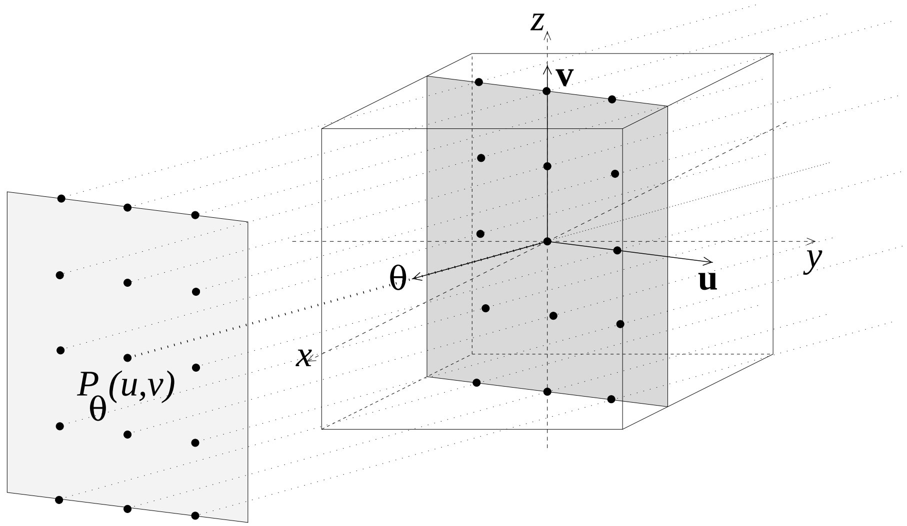
X-ray rendering



X-ray rendering

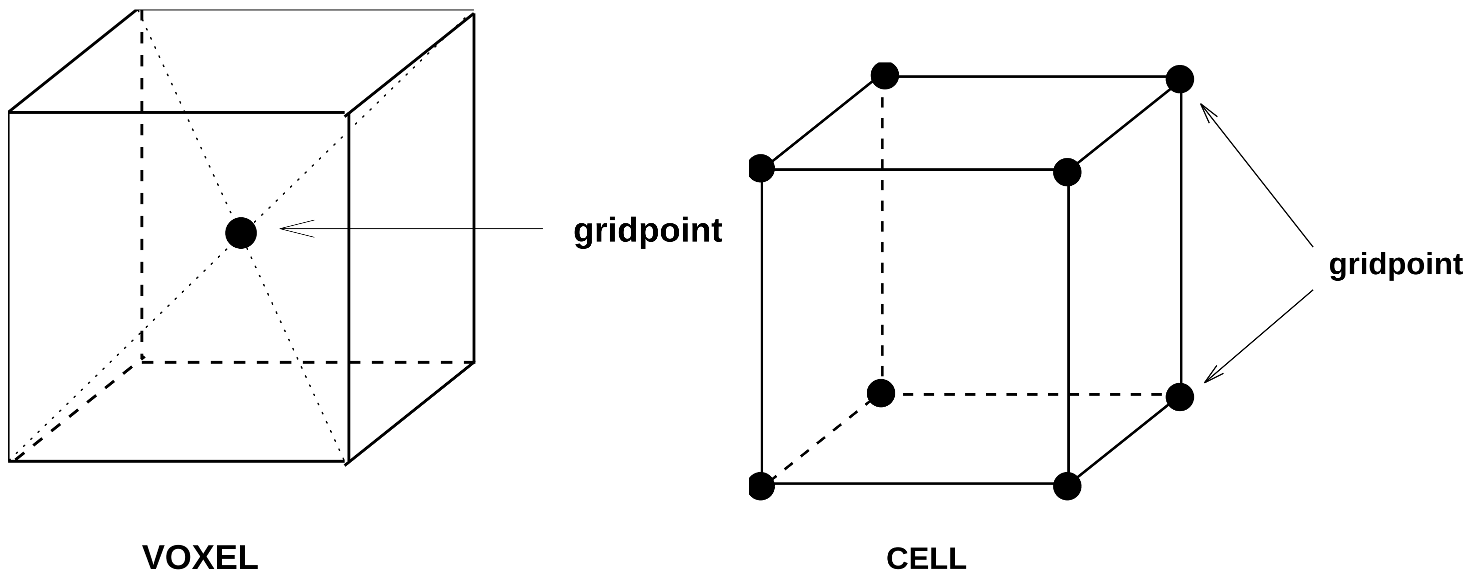
- integrate the density f along the line of sight
- Mathematical concept: X-ray transform:

$$\boxed{\mathcal{P}_{\theta} f(u, v) = \int_{\mathbb{R}} f(u\mathbf{u} + v\mathbf{v} + t\theta)\,\mathrm{d}t\,.}$$



Voxel vs. Cell model

Represents the 3D division of space by a 3D array of grid points.



- Voxel: grid point in center, constant value in voxel
- Cell: grid points at vertices, value within cell varies

Generalized Voxel model *Höhne et al.*

3D array with data about intensity values of different materials or information about class membership of certain organs Medical data sources:

MRI soft tissue: fat, muscle

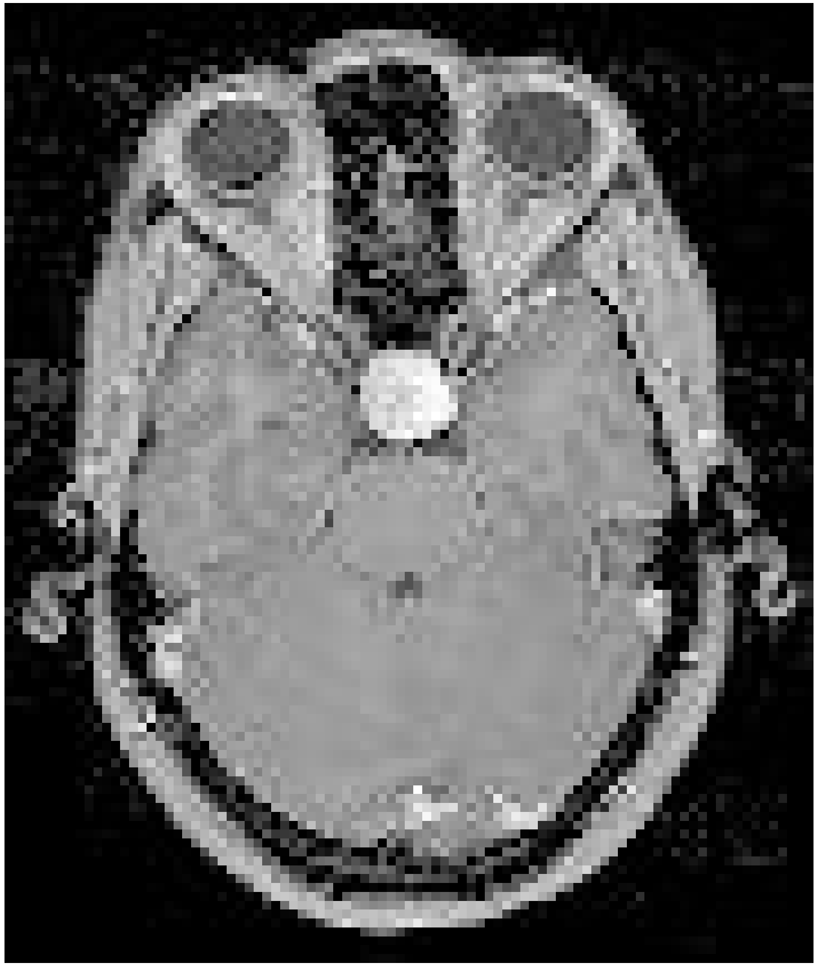
CT hard tissue: bone

PET energy emission, fluid flow, physiology

CT vs. MRI

CT image: 512 x 512 pixels, 2 bytes/voxel.

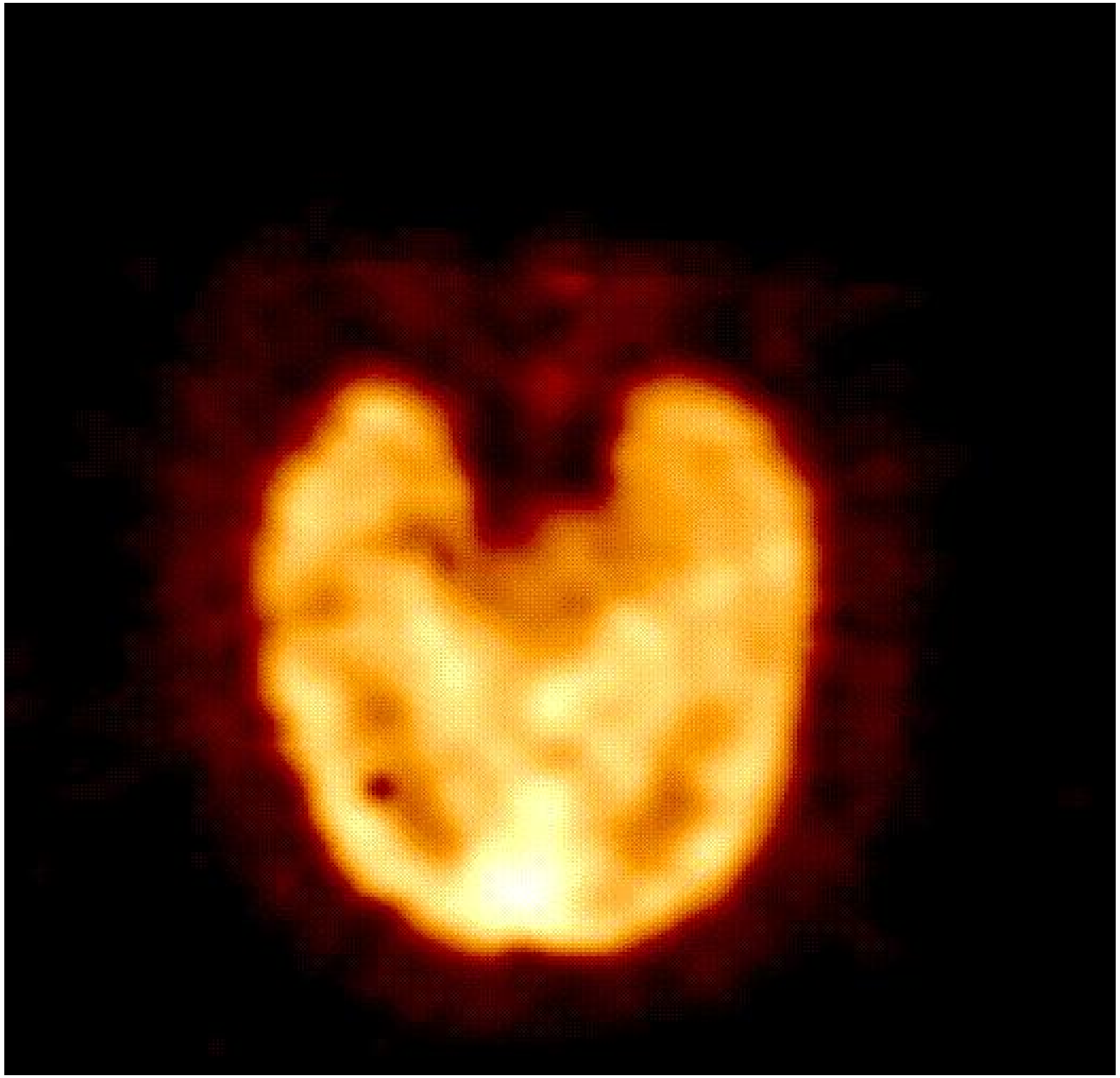
CT vs. MRI



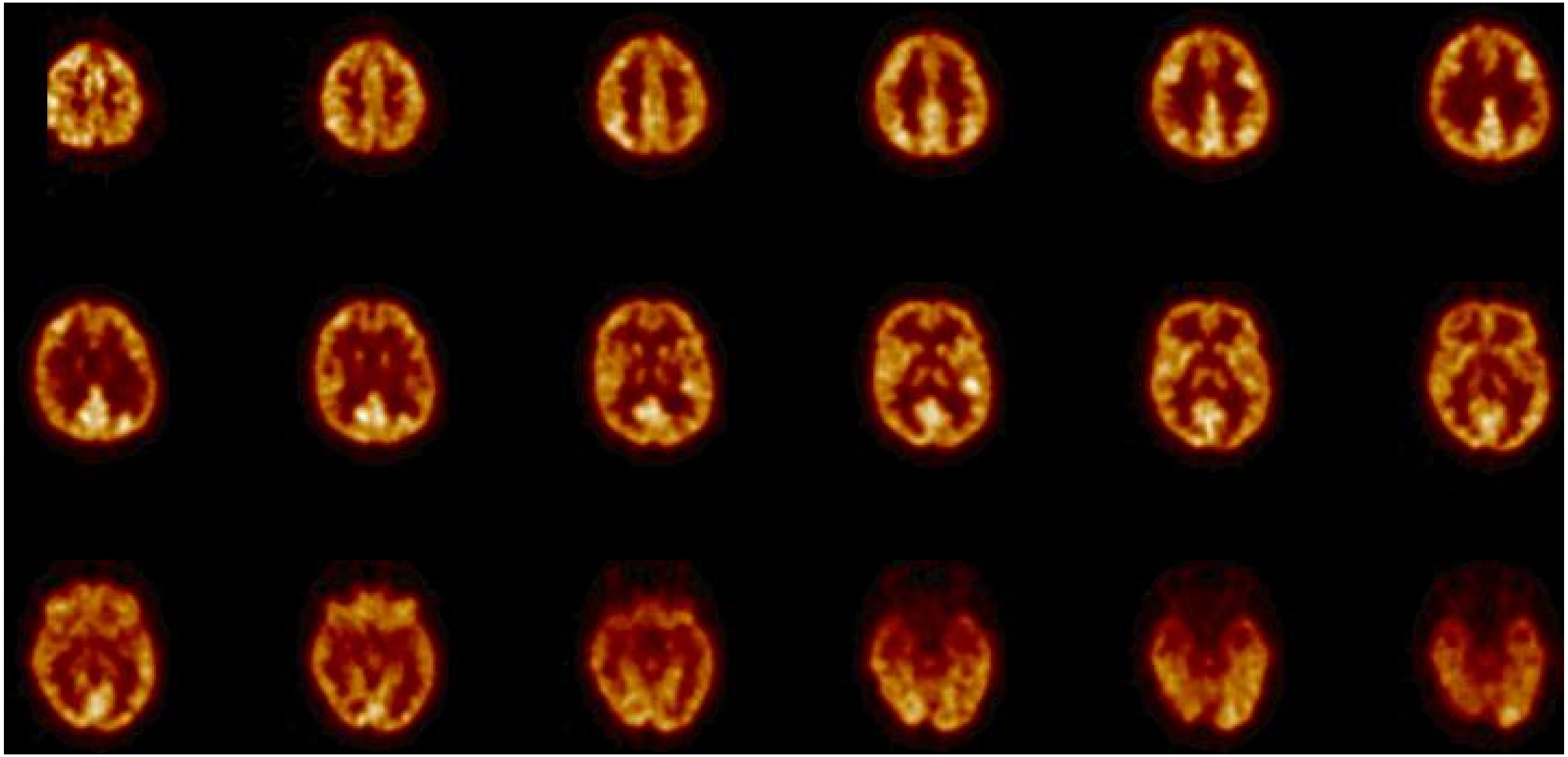
MR image: 256 x 256 pixels, 2 bytes/voxel.

MRI

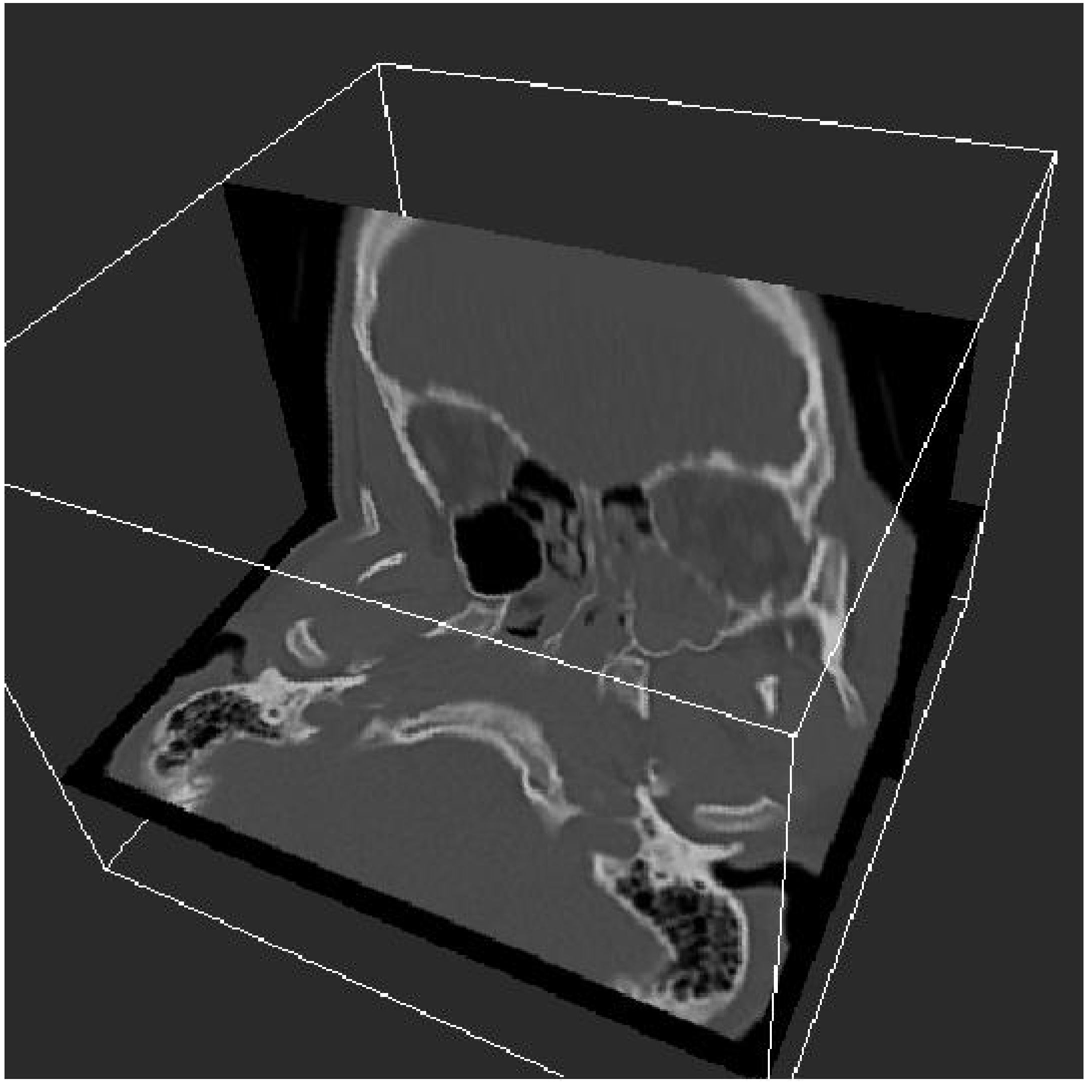
PET



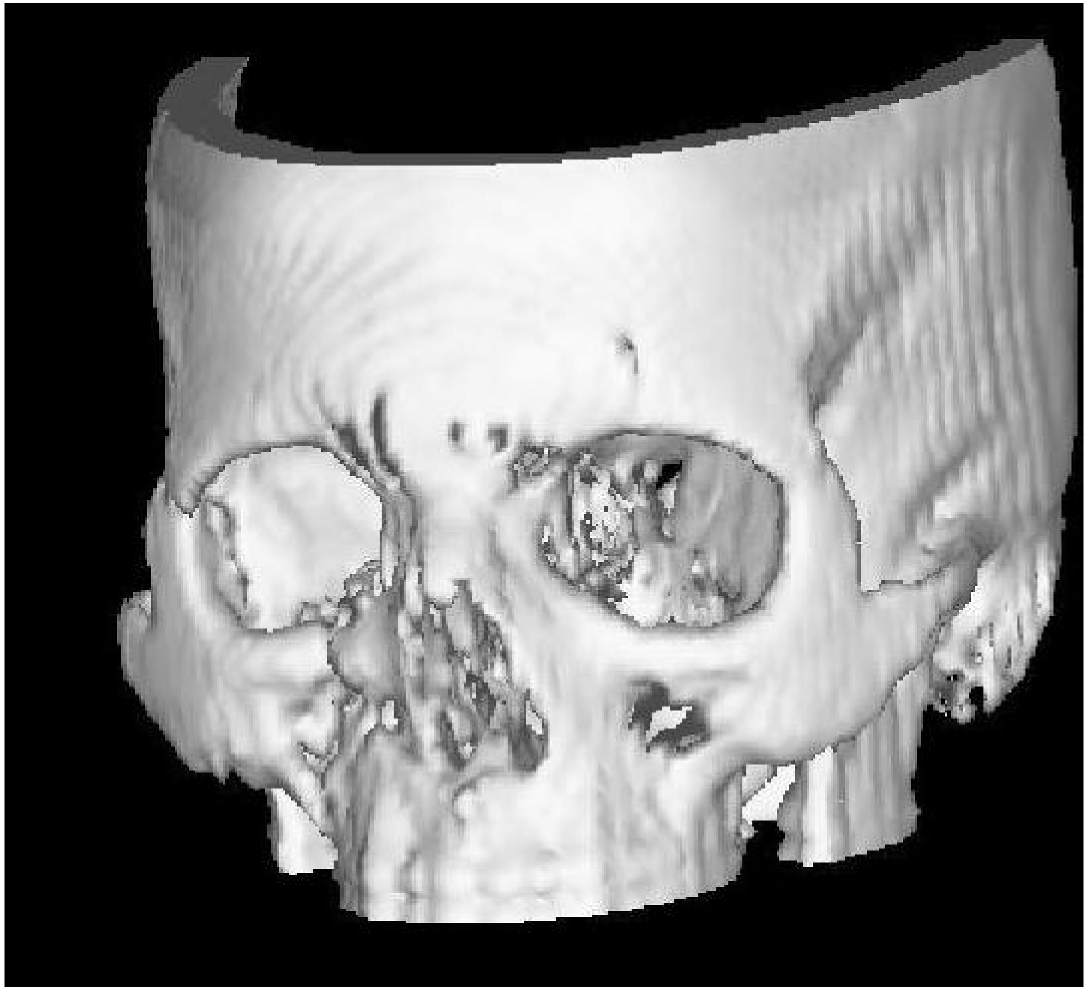
Series of PET scans



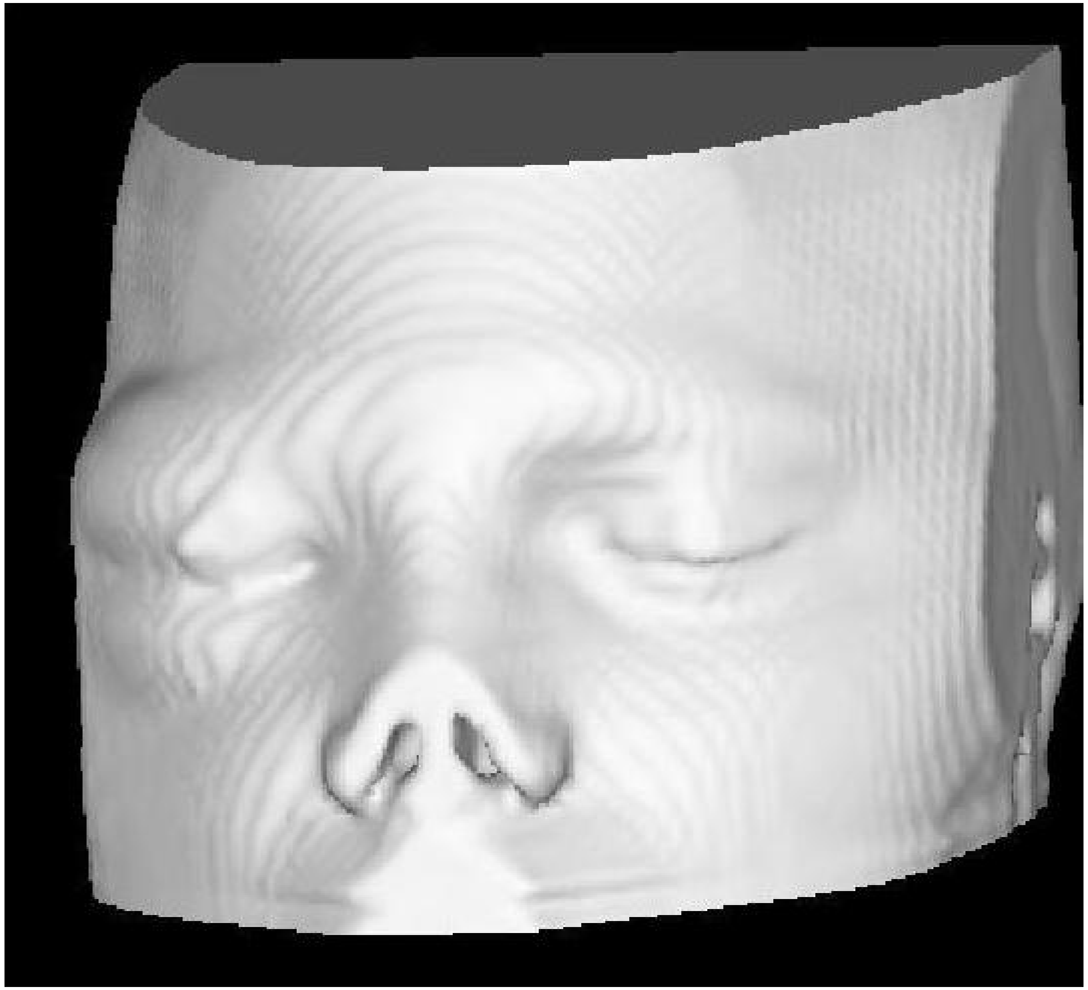
Orthogonal slices



Iso-surface: bone



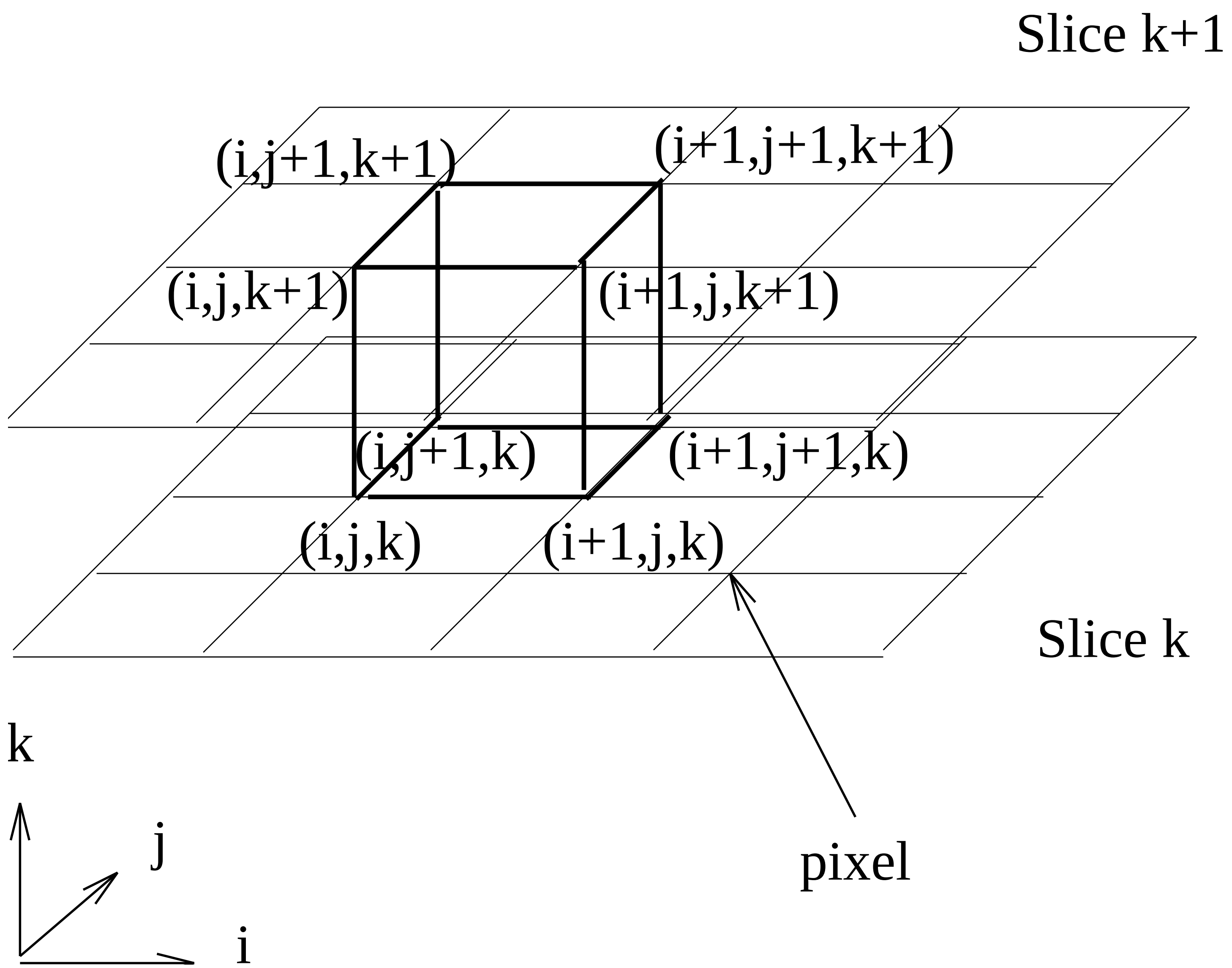
Iso-surface: skin



Marching Cubes *Lorensen & Cline*

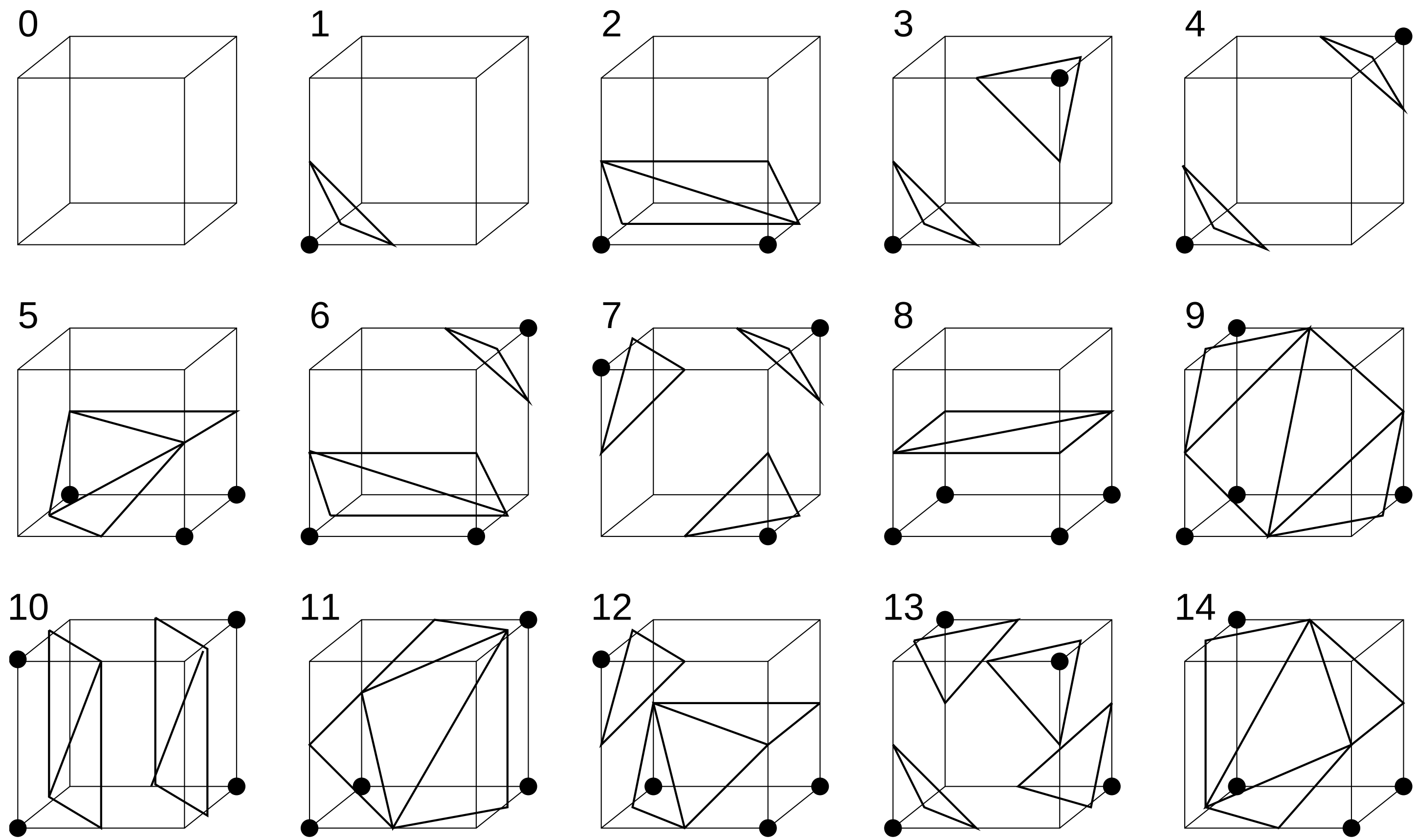
1. read 4 successive slices into memory
2. consider cube with 8 data points as vertices : 4 in first slice, 4 in next slice
3. classify vertices 1=inside, 0=outside surface w.r.t. iso-value to determine index of the cube
4. use index to retrieve intersection- and triangulation pattern from look-up table
5. determine exact cube side intersections and surface normals by interpolation of vertex values
6. for each triangle from table, pass the computed 3 vertex values to graphical hardware

Marching Cubes: setup

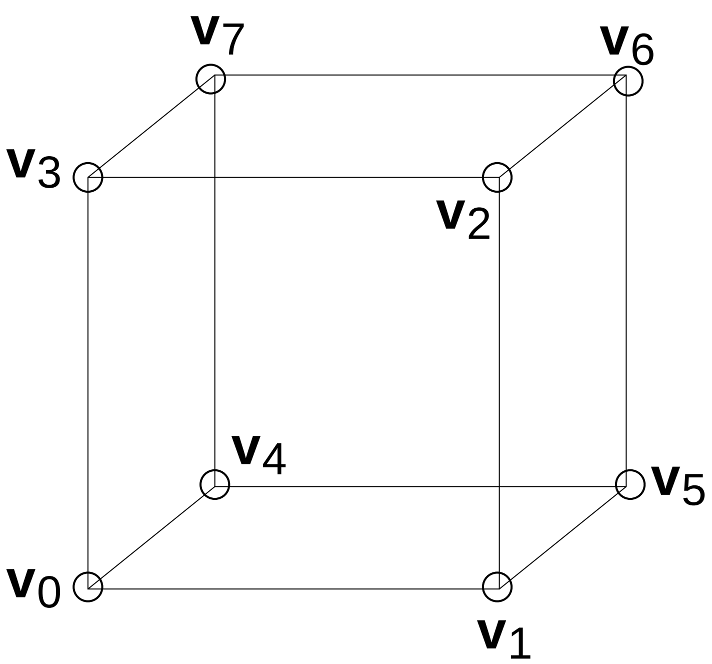


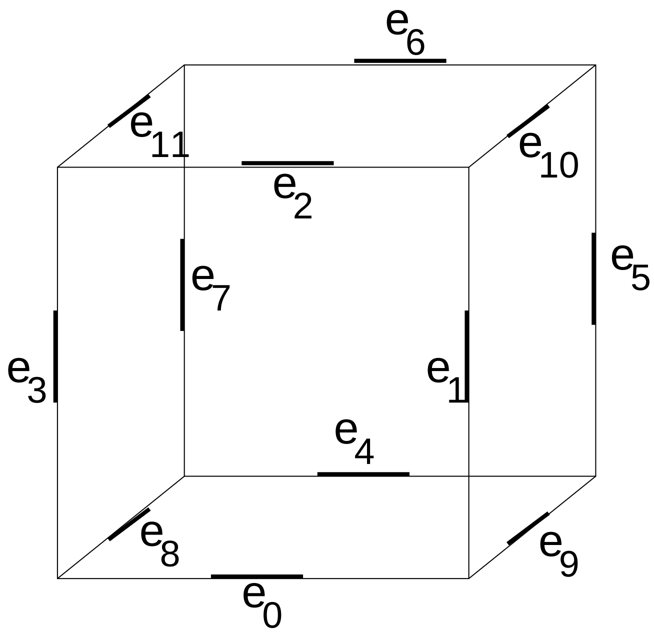
Marching Cubes

256 possible patterns :
by inversion and rotation reduce to 15 basic patterns



Marching Cubes: test cube





index	$\mathbf{v}_7$	$\mathbf{v}_6$	$\mathbf{v}_5$	$\mathbf{v}_4$	$\mathbf{v}_3$	$\mathbf{v}_2$	$\mathbf{v}_1$	$\mathbf{v}_0$

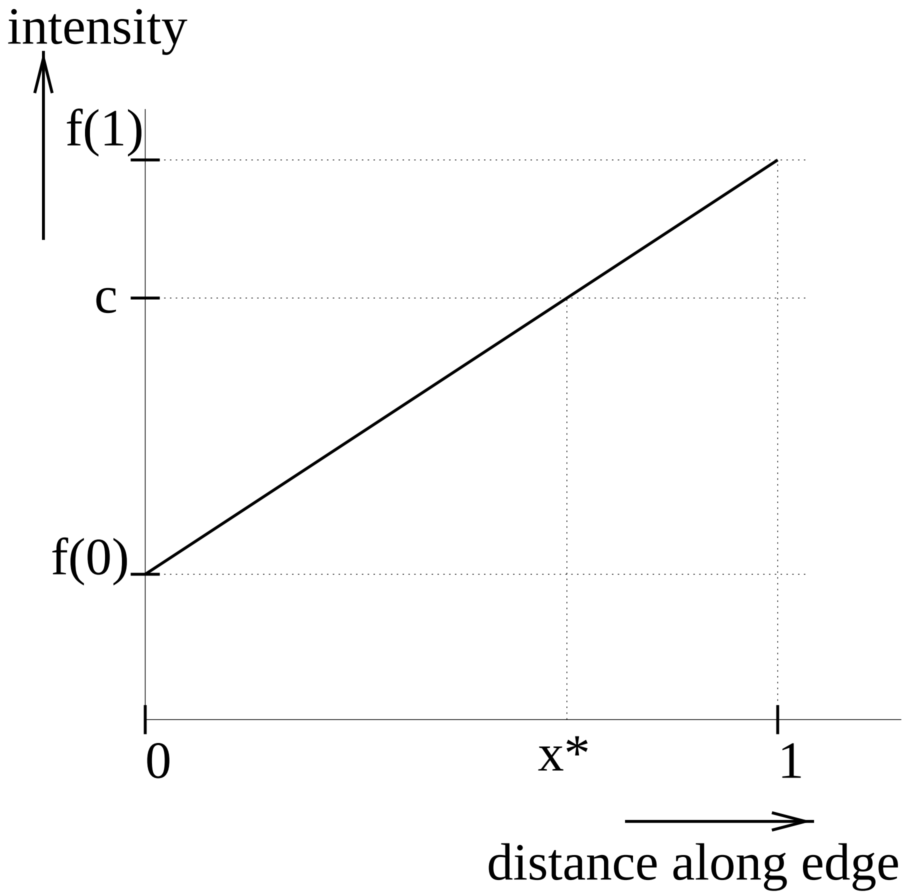
Marching Cubes: index table

index	basic pattern	inversion	rotation
0	0	0	0, 0, 0
1	1	0	0, 0, 0
2	1	0	1, 0, 0
3	2	0	0, 0, 0
$\vdots$	$\vdots$	$\vdots$	$\vdots$
255	0	1	0, 0, 0

basic pattern	intersection of sides	triangulation pattern
0	–	–
1	$e_1e_4e_9$	$e_1e_9e_4$
2	$e_2e_4e_9e_{10}$	$e_2e_{10}e_4, e_{10}e_9e_4$
$\vdots$	$\vdots$	$\vdots$
14	$e_1e_2e_6e_7e_9e_{11}$	$e_1e_2e_9,\ e_2e_7e_9,$ $e_2e_6e_7,\ e_9e_7e_{11}$

Cube side intersections

The exact cube side intersections are determined by linear interpolation.



Surface normals

- The surface normals are determined by linear interpolation of vertex normals.

- Each vertex normal is computed by central differences:

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

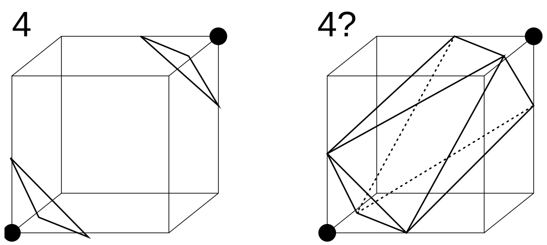
where $s(i,j,k)$ is the data array.

Marching Cubes: (dis)advantages

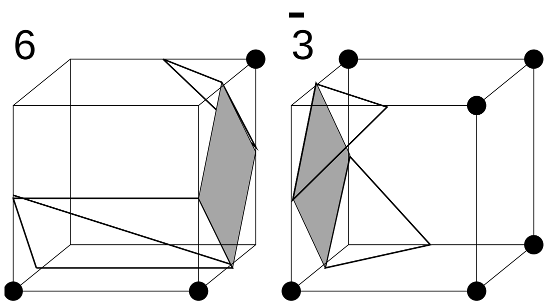
+ data-volume scanned only once for 1 iso-value

+ determination of geometry independent of viewpoint

– ambiguity in possible triangulation pattern



– holes in geometry may arise



Dividing Cubes

Principle: large numbers of small triangles (projection smaller than pixel) to be mapped as points

- look for cells with vertex values not all equal
- subdivide cells if projection larger than a pixel
- interpolate gradient vector from vertices
- map surface point to pixel with computed light intensity
- no surface primitives needed; so also no surface-rendering hardware

Surface rendering

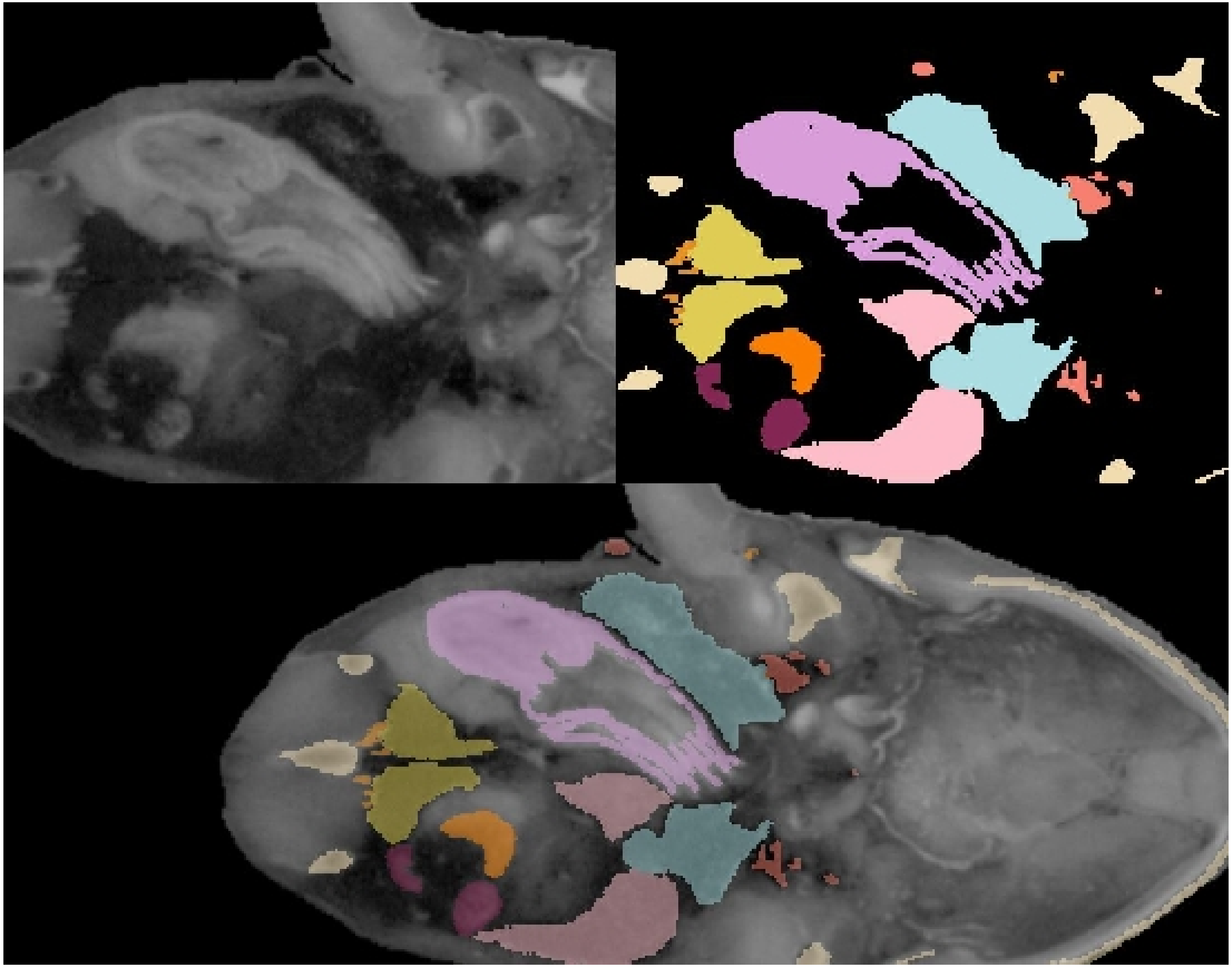
Example: Visualization of segmented data (a frog).

Data acquired by physically slicing the frog and photographing the slices.

Data consists of 136 slices of resolution 470×500.

Each pixel labelled with tissue number for a total of 15 tissues.

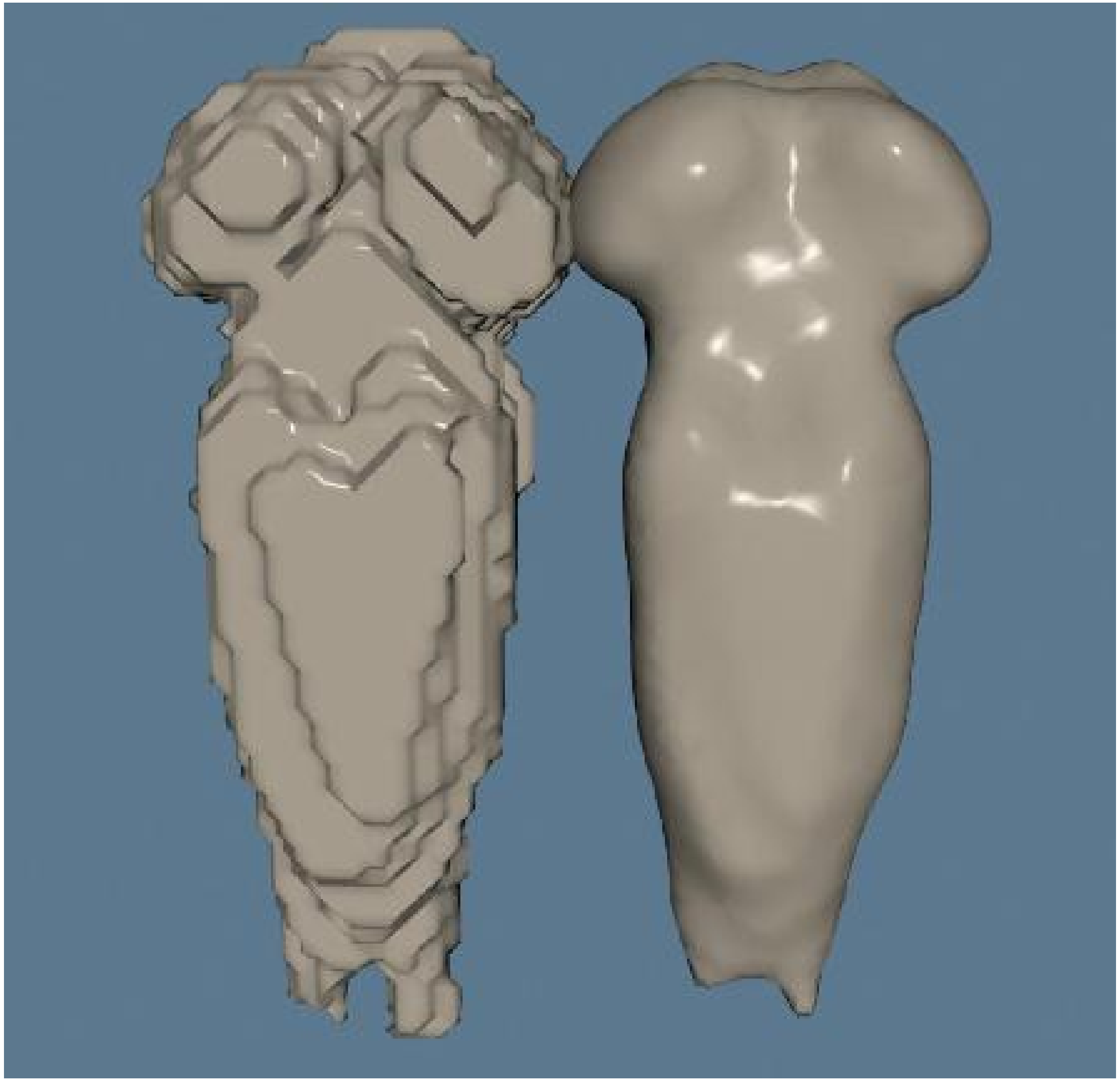
Segmented slice of frog



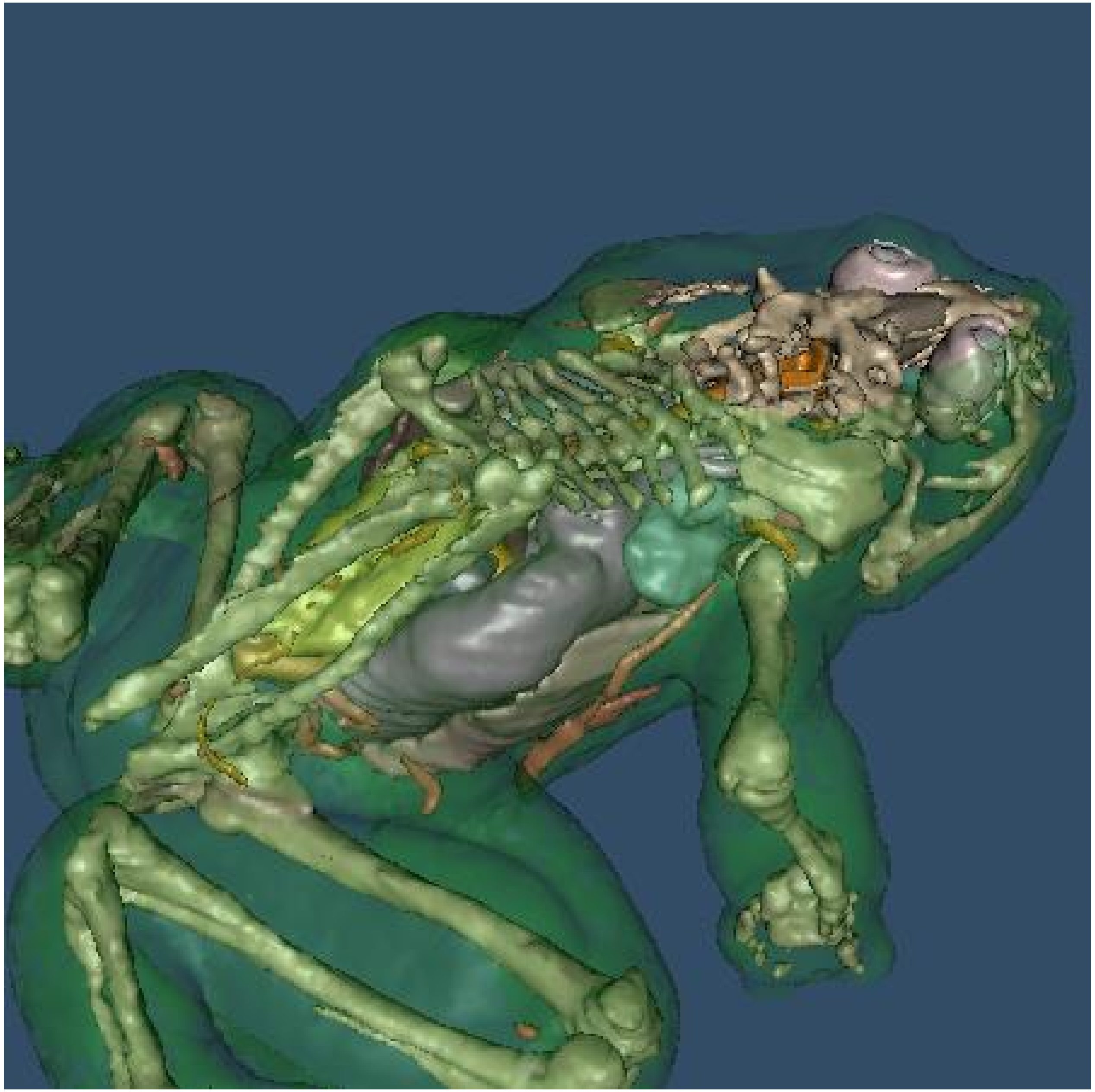
Simplified visualization pipeline

1. read segmentation data
2. remove islands (optional)
3. select tissue by thresholding
4. resample volume (optional)
5. apply Gaussian smoothing filter
6. generate surface using Marching Cubes
7. decimate surface (optional)
8. write surface data

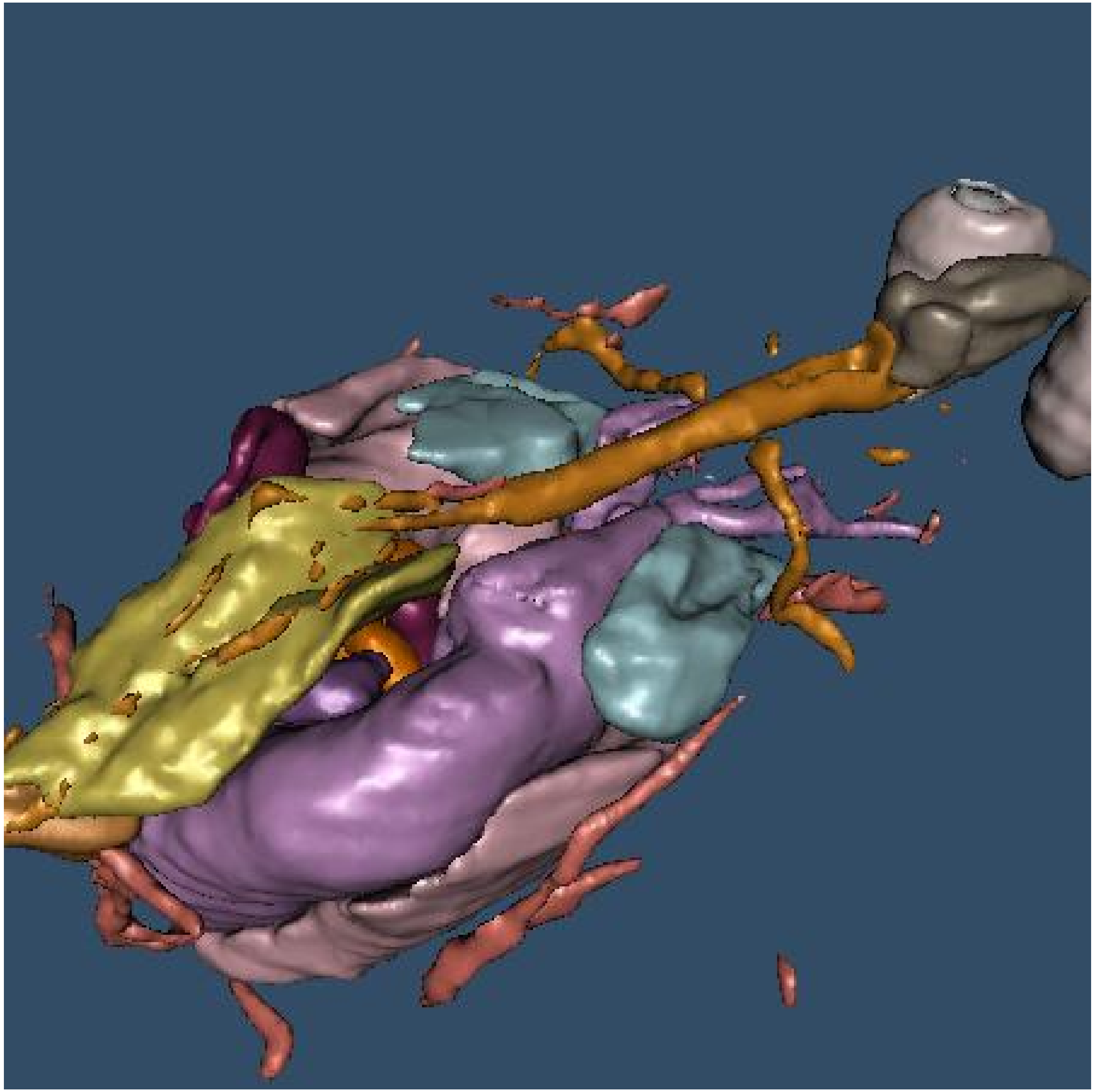
Why smooth the volume?



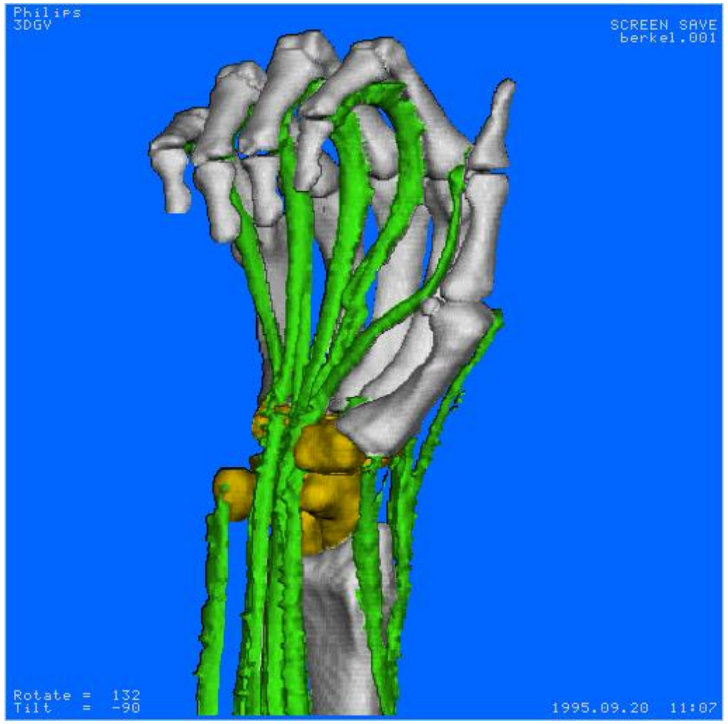
Frog

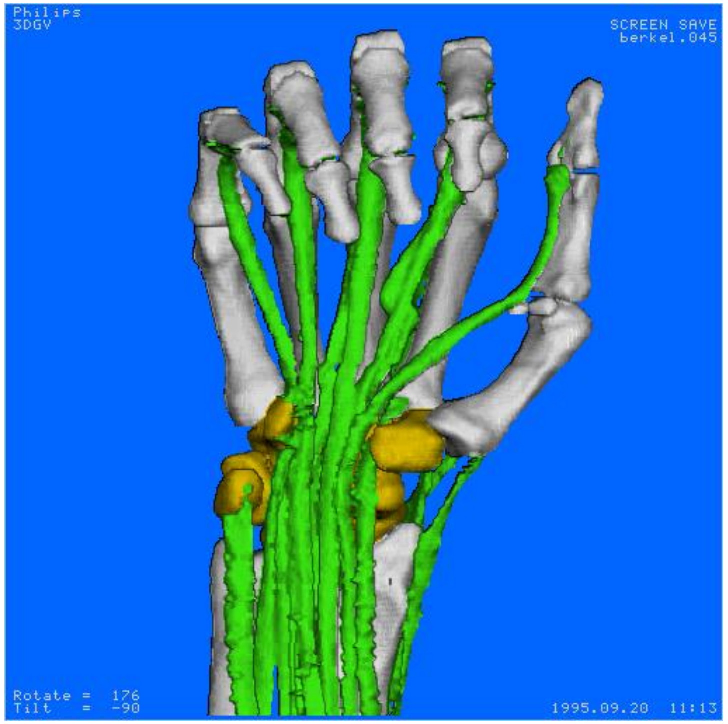


Frog Organs



3D CT imaging





CT images of bones and tendons of a hand. All tendons and bones are separately segmented. (Courtesy: prof. Frans Zonneveld).

Drawbacks of surface rendering

- ✘ only approximation of surface
- ✘ only surface means loss of information
- ✘ amorphous phenomena have no surfaces, e.g. clouds
- ✘ MR also difficult to visualize: different tissues map to the same scalar value