Direct Volume Rendering (I)

D. Tost

This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

3D rendering

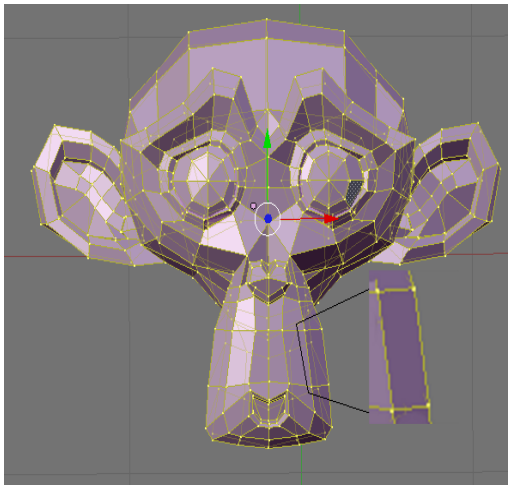
From painting to sculpting

This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

3D Models

Surface models

e.g. Polygonal meshes (vectorial) ← Last session



Volume models

e.g. Voxel models (raster)

← Today

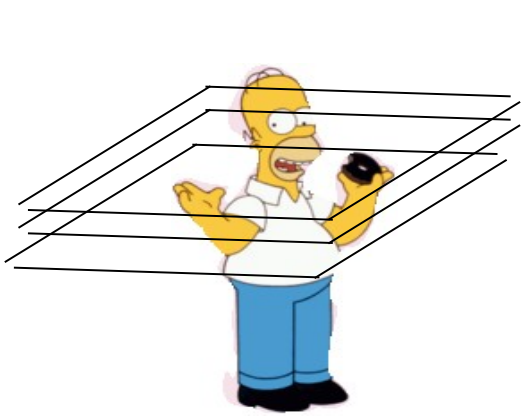
Image from Kniss at al., 2002

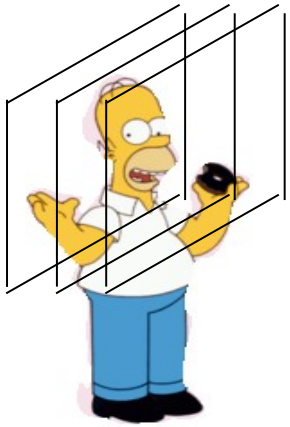
This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

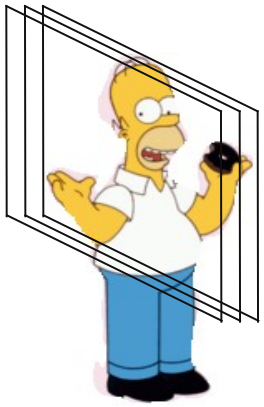
3D images

Stack of 2D images sampling a volumetric region

Different orientations: sagittal, coronal and axial

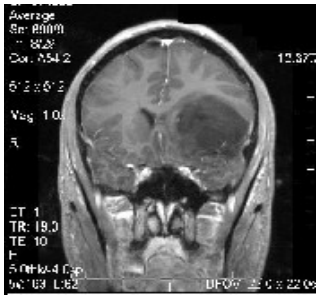






Axial

Coronal



Sagittal

This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Volume models

- **Sample space:** V closed region of R^3
- **Property space:** (dimension n) $\underbrace{R \times R \ldots \times R}_{n}$
- **Reconstruction function**:

$$f : V \rightarrow R \times R \ldots \times R, \forall\, p \in V, f(p) = v$$

v is the property value of p

Which properties? The ones captured by the medical imaging device (tissue density, activity) and mapped in a range depending on the property type

This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Volume models

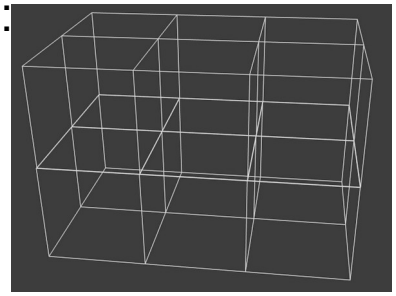
Space enumeration models
Grid-based models

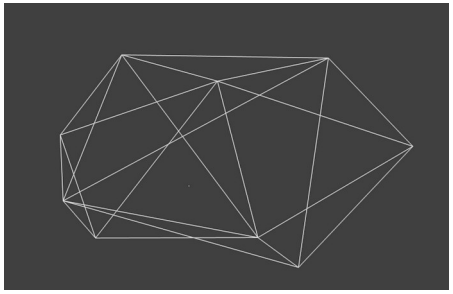
Given a grid of sampled values, the sample space V is tesselated into *nc* disjoint adjacent cells C_i such that:

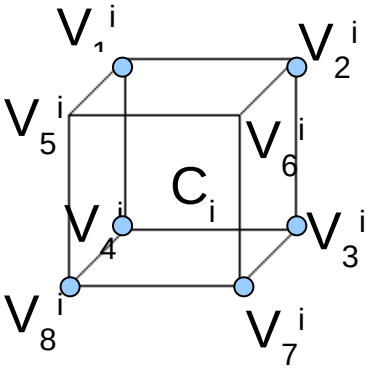
$$V = C_1 \cup C_2 \cup C_3 .... C_{nc}$$

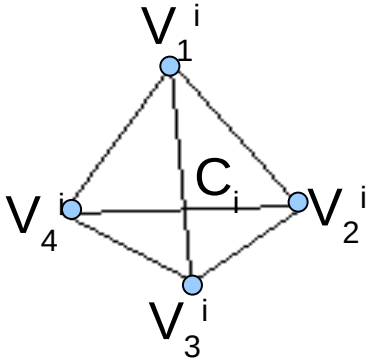
$$\forall Ci, Cj, Ci \cap Cj = \emptyset \quad *$$

$$\forall p \in V, \exists C_i, p \in C_i$$









* the cells are semi-open, a vertex belongs to only one cell

This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Volume models

Space enumeration models
Grid-based models

The reconstruction function at the cell vertices is defined as the sampled values at the corresponding locations

$$\forall V_j^i,\ f(V_j^i) = Sampled\ Value\ at\ V_j^i$$

The reconstruction function at any interior point is defined as an interpolation of the sampled values of the cell in which the point is*.

* Larger neighborhoods can be applied

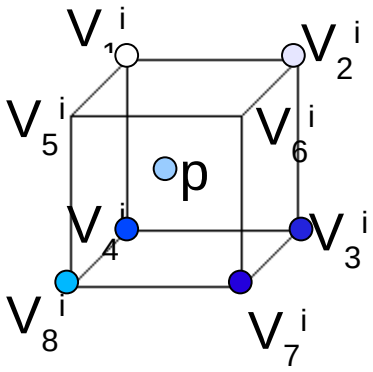
This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Volume models

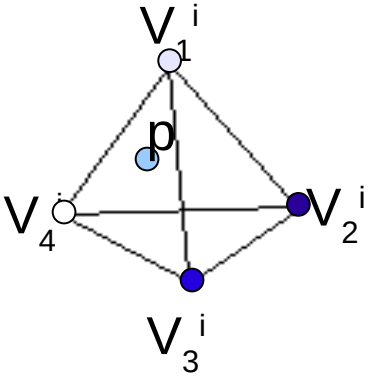
Space enumeration models
Grid-based models

The reconstruction function inside a cell is a smooth interpolation funcion of the cell vertices values.

Generally: tri-linear interpolation



C_i



Cubical cells: voxel model

Tetrahedral cells: tetrahedra model

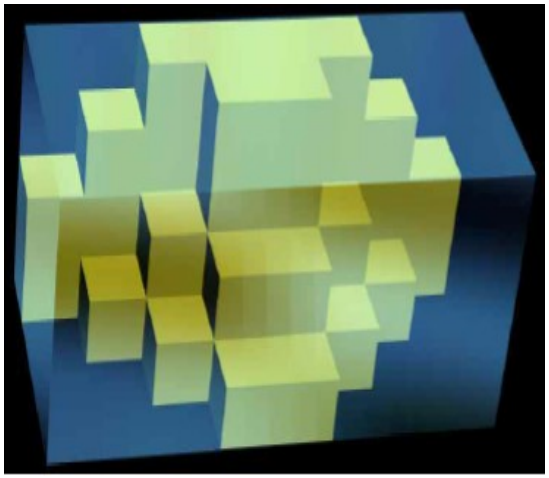
This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

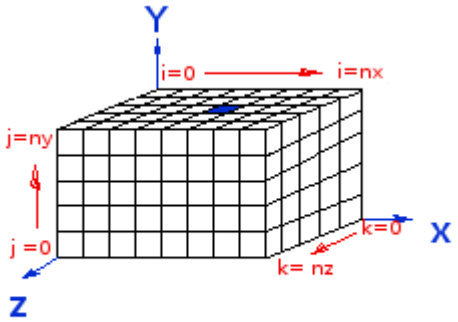
Volume models

Cubical cells

Voxel: volumetric “pixel”

Implicit geometry: it is not necessary to store the coordinates of every cell vertex, only the model coordinate system and the voxel edge-length.





vijk = (x0 + i.A, y0+j.A, z0+k.A)

Occupancy in memory of a voxel model:

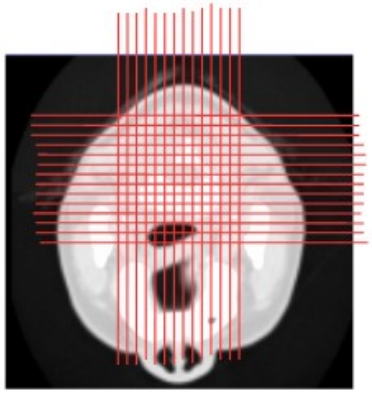
$$resx \times resy \times resz \times \sum_{i=1}^{nprop} sizeof\left(p_i\right)$$

where

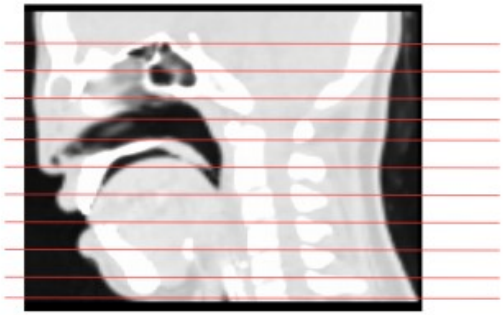
- `resx, resy and resz`: voxel resolution (number of voxels) in the x, y, z axes of the voxel coordinate system
- `nprop`: number of properties
- p_i : `i`th property

This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

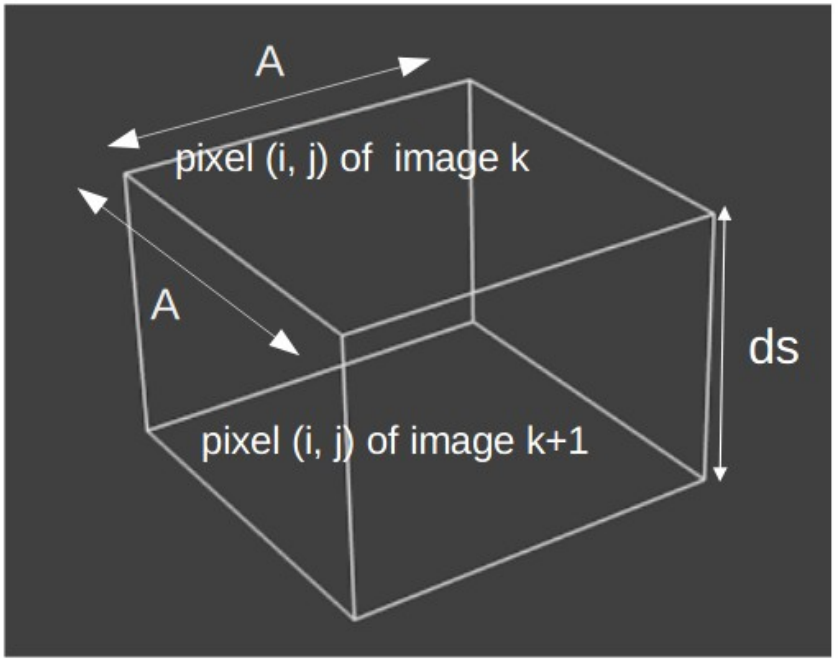
Construction of a voxel model
From 3D images



A = interpixel distance



dS = Interslice distance



This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Exercise

Exercise: What is the occupancy in memory of a voxel model of voxel resolution 256x256x185, 1-byte property per voxel?

Answer: 256*256*185*1 bytes

Exercise: What is the volume of the region represented by a set of 126 2D images of pixel resolution 234x156, 72x72 ppi and interslice distance 0.7mm?

Answer:

25.4 mm → 1 inch → 72 pixels

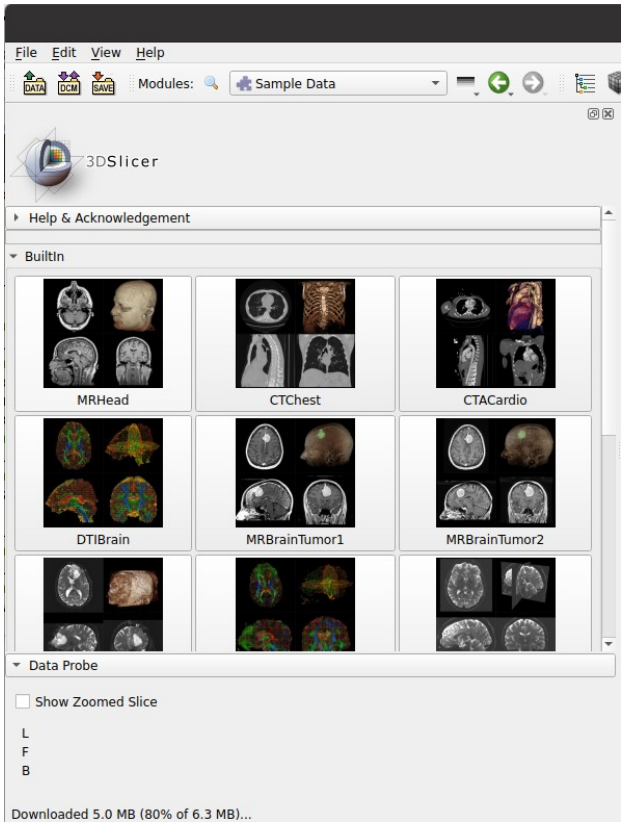
0.35 mm = 25.4*1/72 mm ← 1/72 inch ← 1 pixel

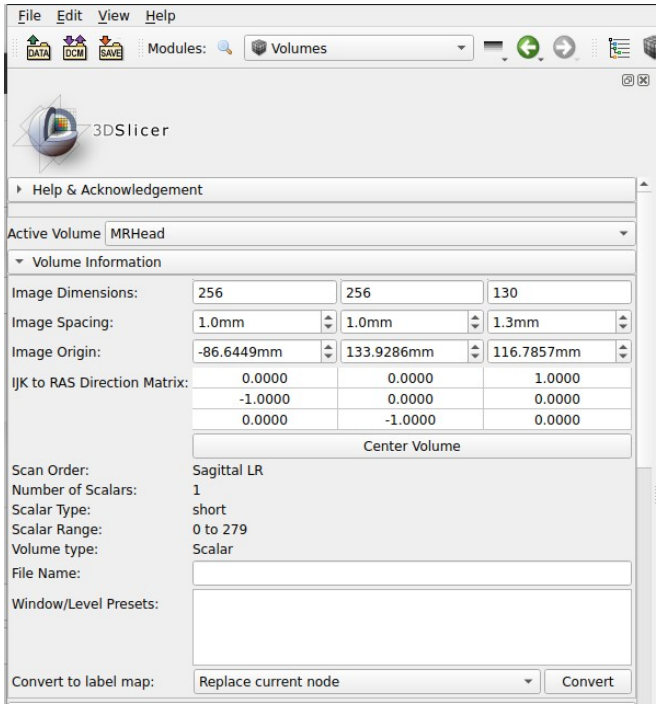
0.35*234*0.35*156*0.7*(126-1) = 397512.64 mm3 = 397.5 cm3

This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Practice

Open Slicer and download a sample dataset (MRHead) and explore it.





This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

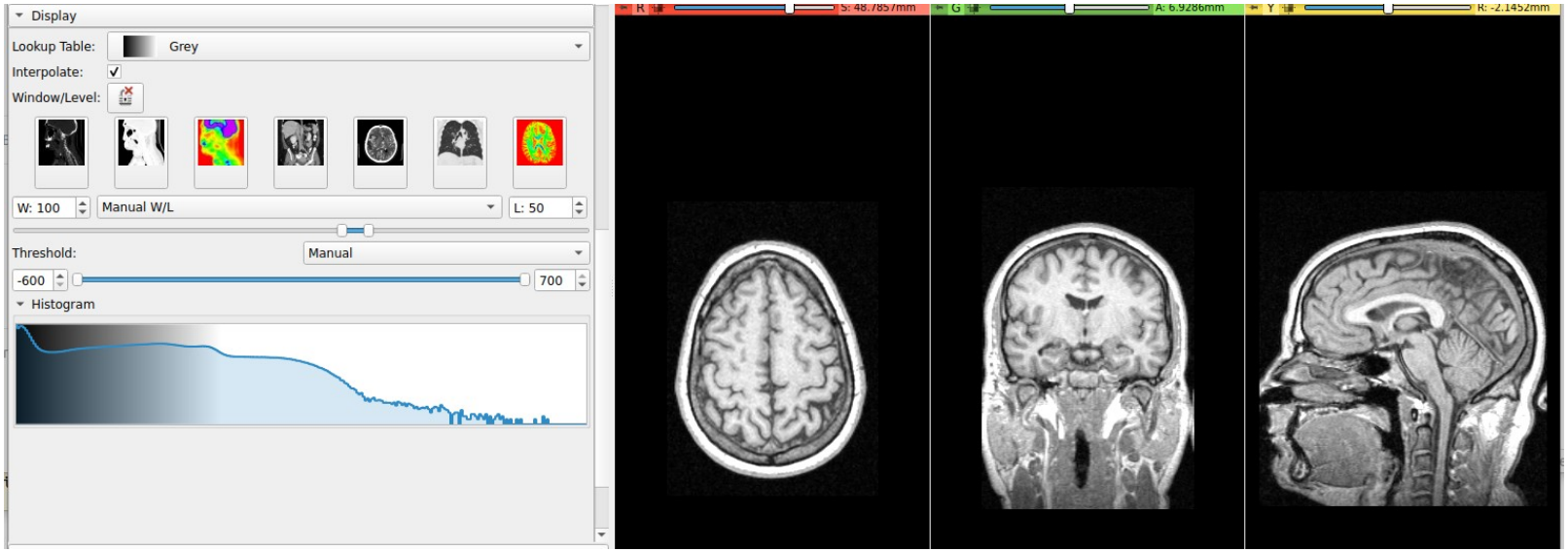
Understand the slices

Move in the slices. Try to understand the correspondance between color and slice orientation. Can you change it?

Look at the histogram. Can you conclude anything?

Play with the threshold. What are you doing? Understand the effect of changing it.

Play with Window/Level. Does it remind you something you implemented in the 2D lab?



This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

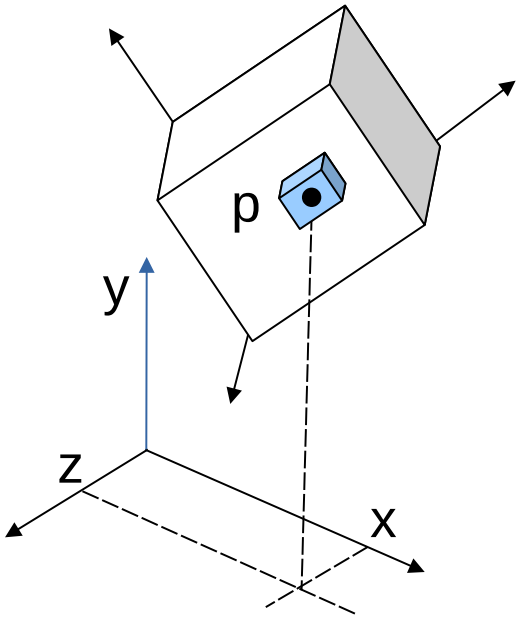
Voxel model

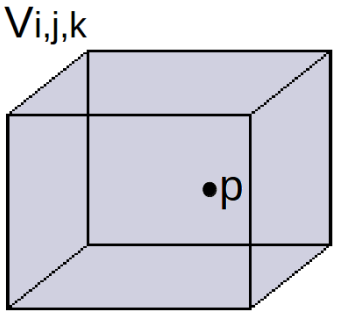
Computation of the property value at a point p

Nearest neighbor approach

1. Computation of the point's cell indexes (i, j, k)

2. Computation of the cell property values at the cell: MV[i, j, k]





This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

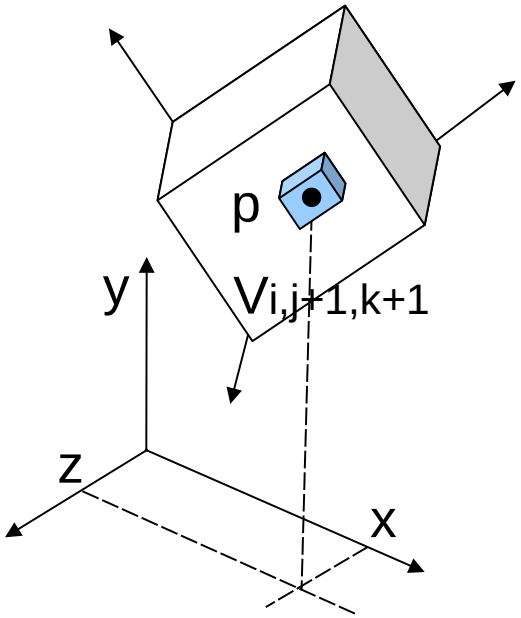
Voxel model

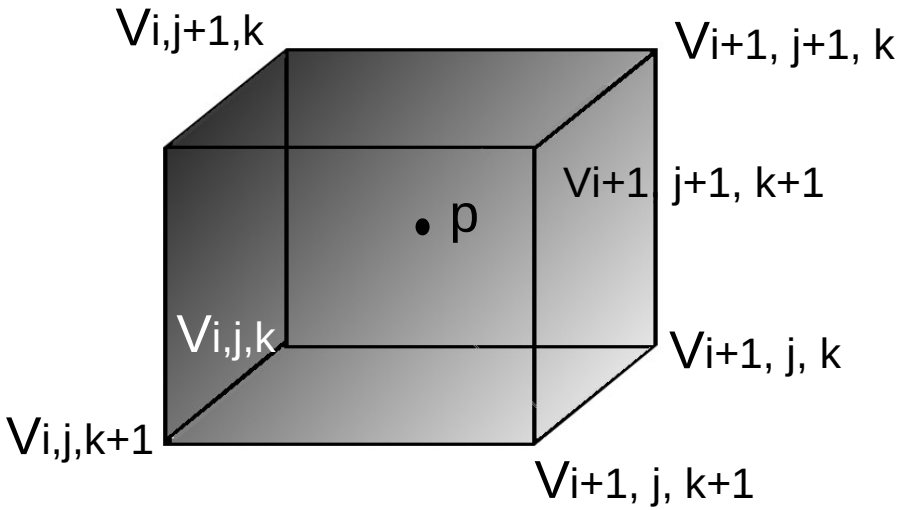
Computation of the property value at a point p
Tri-linear interpolation

1. Computation of a point's cell

2. Computation of the point's cell vertices property values

3. Tri-linear interpolation of the cell vertices property values at p

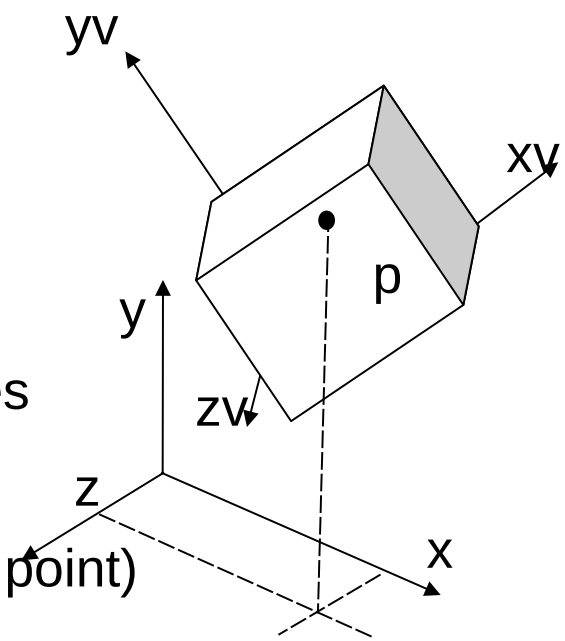




This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Voxel model

Computation of voxel indexes



1. Change of coordinate system

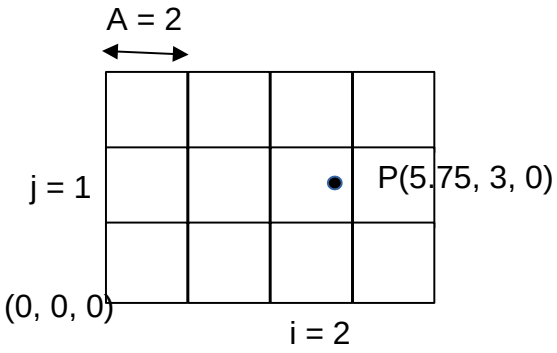
   p(x, y, z) world coordinates

   ➡ p(xv, yv, zv) voxel model coordinates

2. Obtention of the voxel indexes (rasterization of the point)

   i = int (xv/A)
   j = int (yv/A)
   k= int (zv/A)

   dx = (xv-i*A)/A
   dy = (yv-j*A)/A
   dz = (zv-k*A)/A



Voxel indexes:
(i, j, k) = (2, 1, 0)

Relative coordinates of P inside the voxel:
(dx, dy, dz) = (0.875, 0.5, 0)

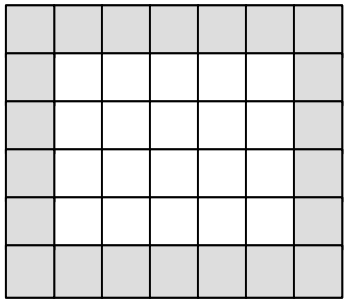
This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Voxel model

Computation of the voxel vertexs properties

MV[i, j, k], MV[i+1, j, k], MV[i+1, j+1, k], MV[i, j+1, k],
MV[i, j, k+1], MV[i+1, j, k+1], MV[i+1, j+1, k+1], MV[i, j+1, k+1]

Be aware of the boundary values!



In the implementation, often, to avoid having to check if a voxel is at the boundary, a larger model is implemented with replicated values of the boundary faces, such that all queried voxels are internal.

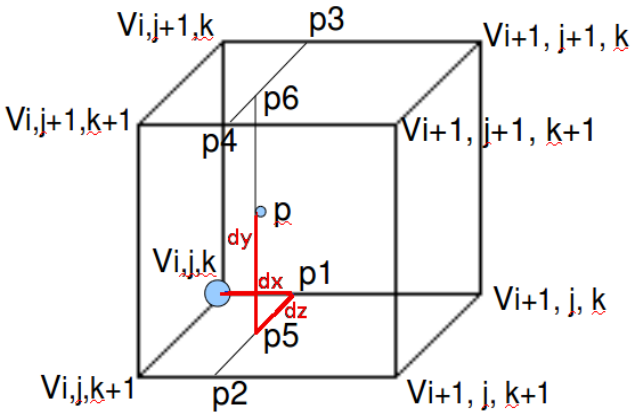
This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Voxel model

Trilinear interpolation

$f(p1) = dx * f(V_{i+1, j, k}) + (1-dx) * f(V_{i, j, k})$
$f(p2) = dx * f(V_{i+1, j, k+1}) + (1-dx) * f(V_{i, j, k+1})$
$f(p3) = dx * f(V_{i+1, j+1, k}) + (1-dx) * f(V_{i, j+1, k})$
$f(p4) = dx * f(V_{i+1, j+1, k+1}) + (1-dx) * f(V_{i, j+1, k+1})$
$f(p5) = dy * f(p1) + (1-dy) * f(p2)$
$f(p6) = dy * f(p3) + (1-dy) * f(p4)$

$f(p) = dz * f(p5) + (1-dz) * f(p6)$



This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Computation of the property value

Computation of a point's voxel indexes:

p(x, y, z) world coordinates

Change of coordinate system

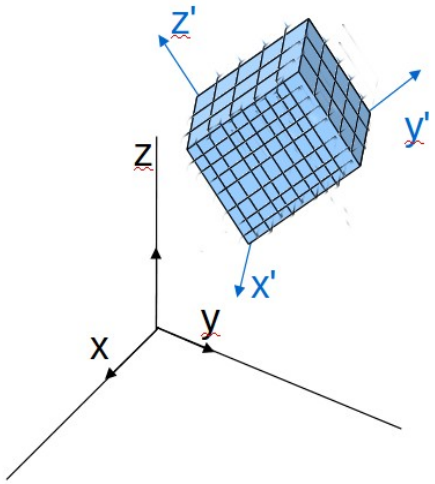
p(x', y', z') voxel model coordinates

Discrete indexes computation (rasteritzation)

(i, j, k) voxel index

Interpolation: nearest neighbour or trilinear

property value



This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

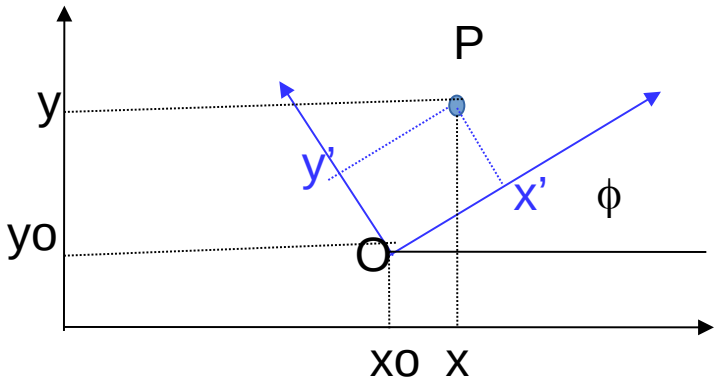
Computation of the property value

Change of coordinate system

p(x, y, z) world coordinates → p(x', y', z') voxel model coordinates

It can be expressed as a composition of 4 geometric transformations:
- a translation of minus the coordinates of the origin of the voxel model
- three axis rotations (Euler's angles)

Example 2D:



- Translation of (-ox, -oy)
- Rotation of -ϕ

This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Geometric transformations

Geometric transformations can be expressed with matrices:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Identity Matrix

$$\begin{pmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

glTranslatef(tx,ty,tz)

$$\begin{pmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

glScalef(sx,sy,sz)

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(d) & -\sin(d) & 0 \\ 0 & \sin(d) & \cos(d) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

glRotatef(d,1,0,0)

$$\begin{pmatrix} \cos(d) & 0 & \sin(d) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(d) & 0 & \cos(d) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

glRotatef(d,0,1,0)

$$\begin{pmatrix} \cos(d) & -\sin(d) & 0 & 0 \\ \sin(d) & \cos(d) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

glRotatef(d,0,0,1)

A change of coordinate system can be expressed as a product of a vector by a matrix:

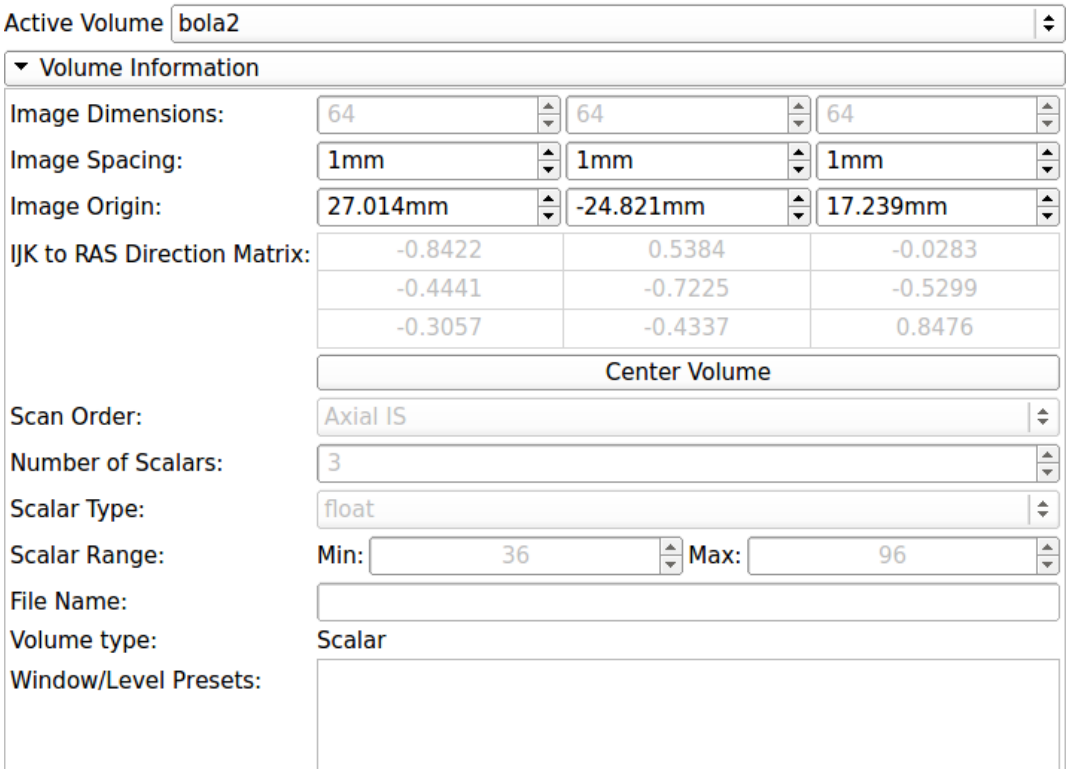
$$\begin{pmatrix} X' \\ Y' \\ Z' \\ H \end{pmatrix} = M * \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

being M the product matrix of 3 rotation and a translation matrix.

http://math.hws.edu/graphicsbook/c3/s5.html

This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Geometric transformations



This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Computation of the property value

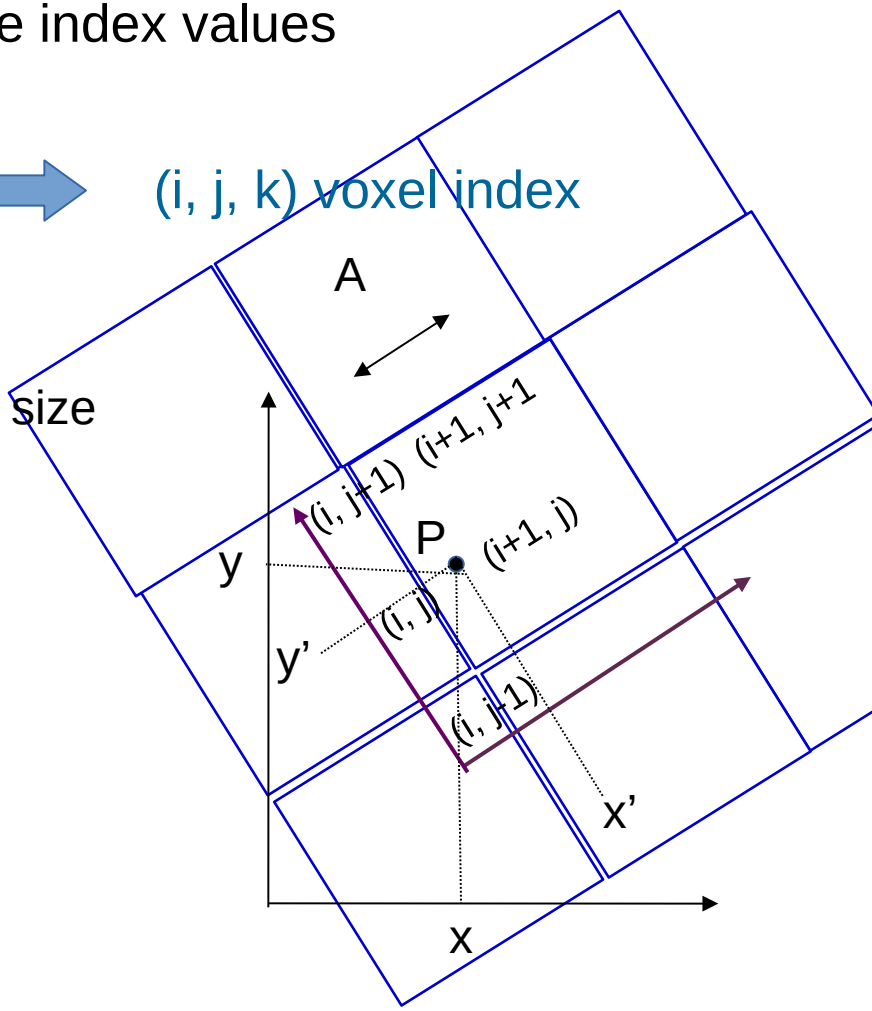
Computation of the discrete index values

p(x', y', z') voxel model coordinates → (i, j, k) voxel index

i = (int) x'/A
j = (int) y'/A
k = (int) z'/A

Where A is the voxel edge size (supposed cubic)

Scaling + Cast to int

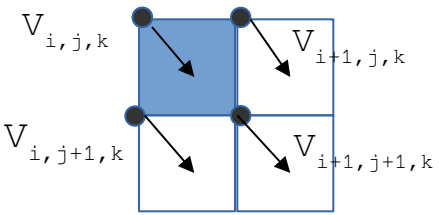


This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Computation of the property value

Nearest neighbour

$f(p) = f(V_{i, j, k})$

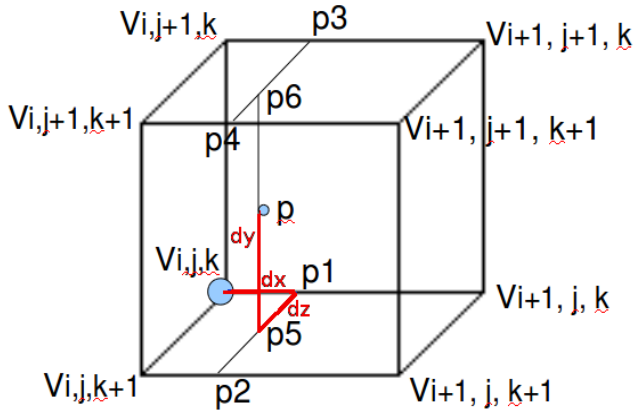


Trilinear interpolation

$f(p1) = dx * f(V_{i+1, j, k}) + (1-dx) * f(V_{i, j, k})$
$f(p2) = dx * f(V_{i+1, j, k+1}) + (1-dx) * f(V_{i, j, k+1})$
$f(p3) = dx * f(V_{i+1, j+1, k}) + (1-dx) * f(V_{i, j+1, k})$
$f(p4) = dx * f(V_{i+1, j+1, k+1}) + (1-dx) * f(V_{i, j+1, k+1})$

$f(p5) = dy * f(p1) + (1-dy) * f(p2)$
$f(p6) = dy * f(p3) + (1-dy) * f(p4)$

$f(p) = dz * f(p5) + (1-dz) * f(p6)$



The values of the property at the vertices of voxel $V_{i, j, k}$ are the values of the neighbour voxels

This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Computation of the property value

Exercice

A voxel model of voxel resolution 100x100x100, 1 uint8 property value per voxel represents the rasterization of a uniform central sphere of radius 40 and value 255.
Suppose that the voxel model is aligned with the world coordinate system (WCS) and that its left, inferior, anterior vertex is located at the origin of the WCS.

What is the property value at the following points, if they are inside the voxel model:

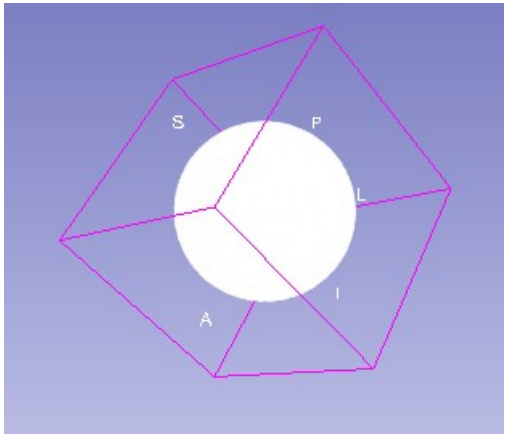
(-5, 10, 20) → external

(5, 5, 5) → 0

(50, 30, 10) → 0

(50.5, 60.7, 70.2) → 255

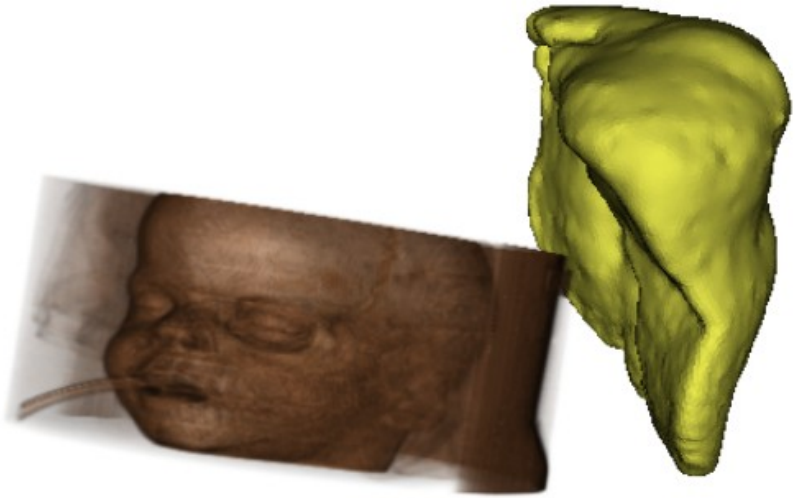
(70, 60, 40) → 255



This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

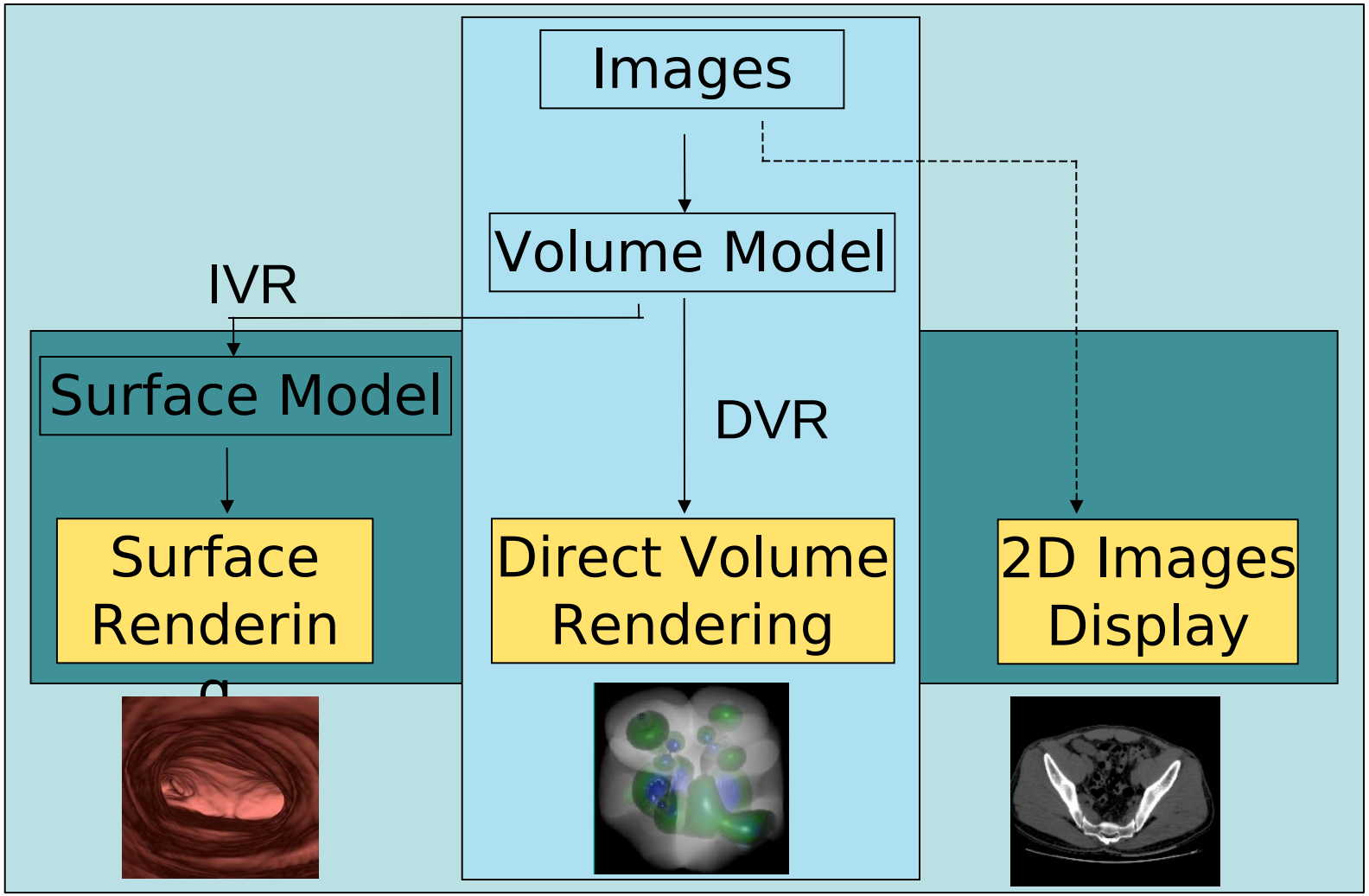
Rendering

The process of obtaining an image from a 3D model (either raster o vectorial)



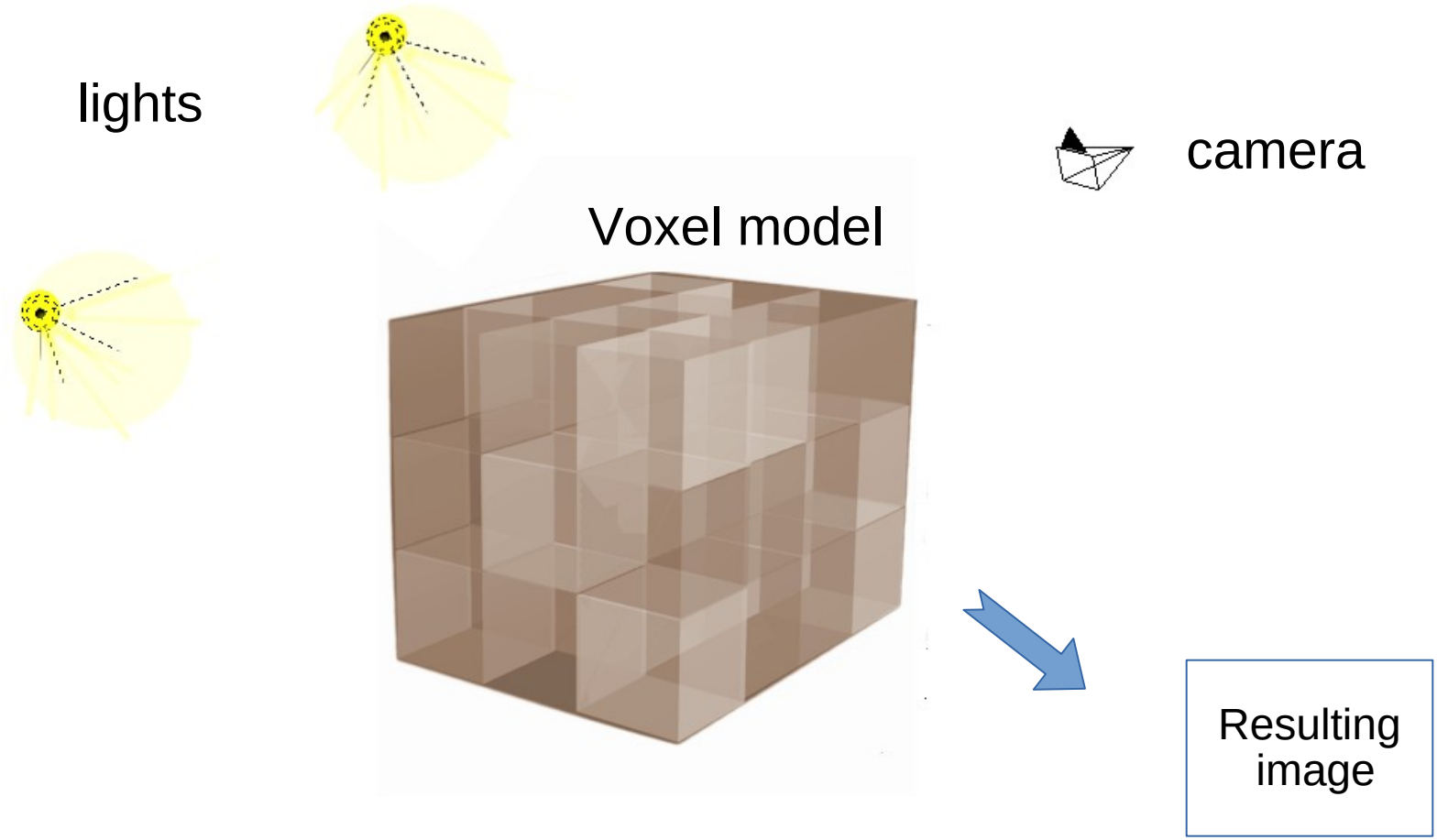
This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Volume rendering



This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

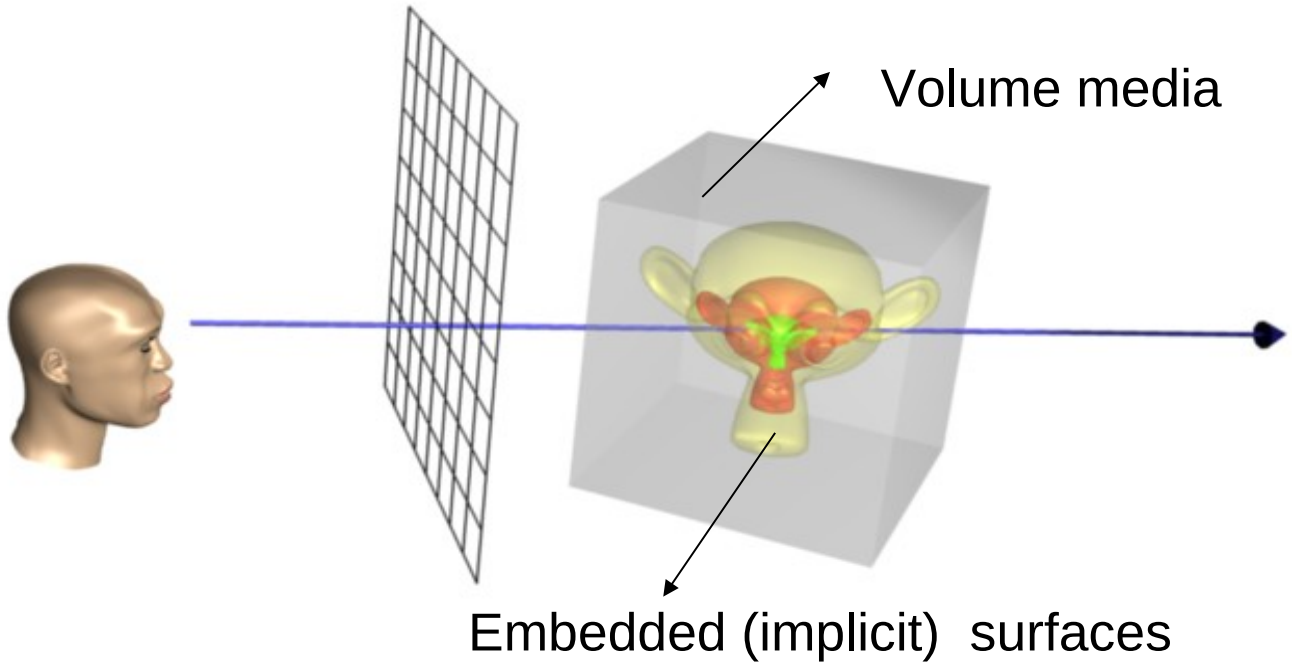
Volume rendering (IVR and DVR)



This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Direct volume rendering (DVR)

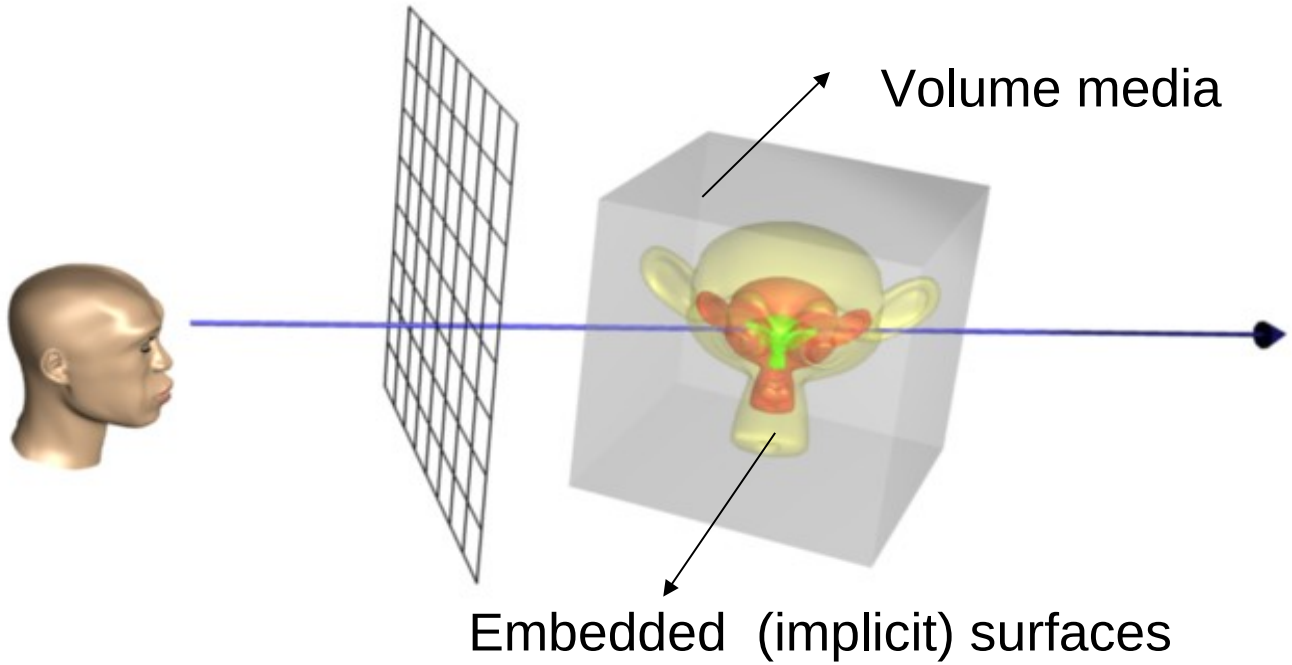
The color of each pixel of the rendered image is a mapping of the radiance of light perceived by the viewer at the corresponding area of the projection plane.



This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Direct volume rendering (DVR)

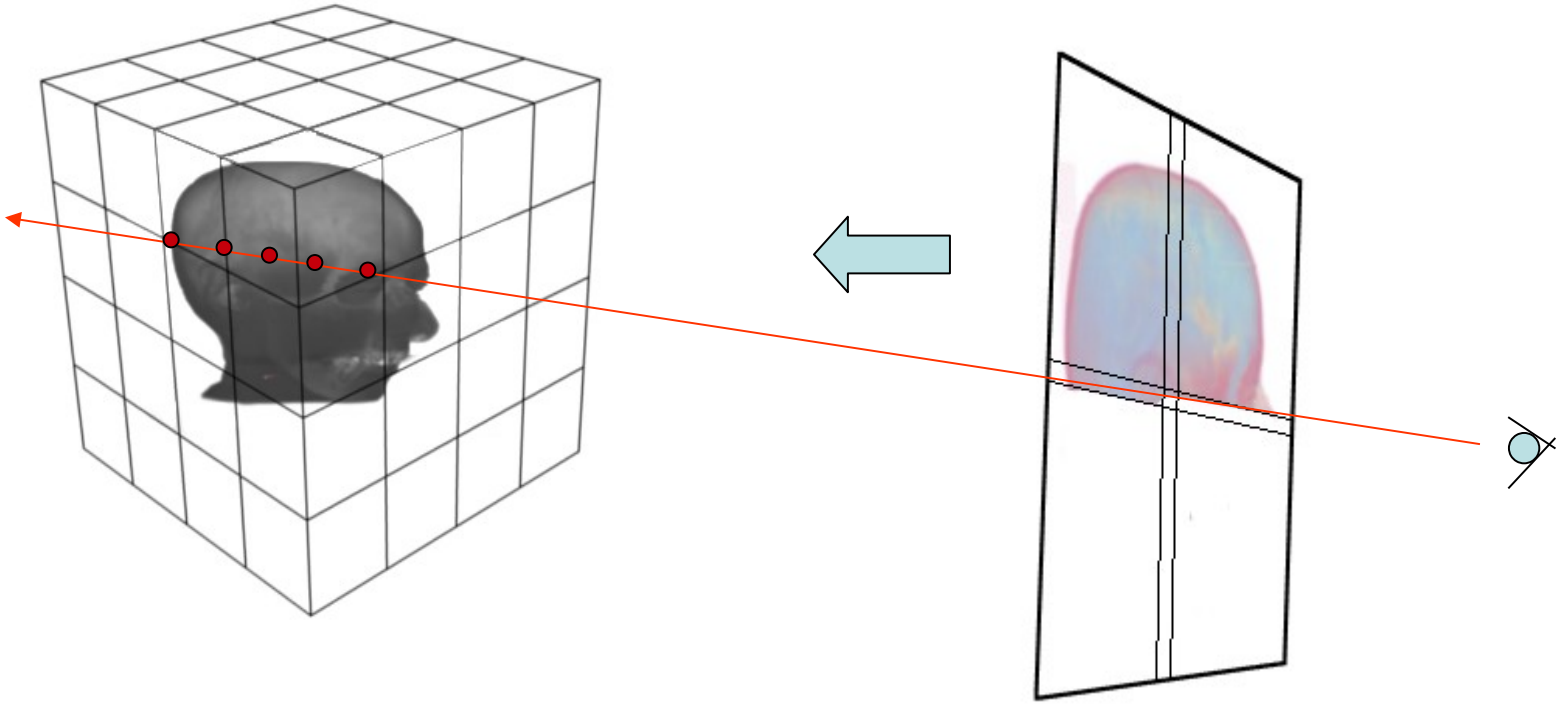
Need to compute the transport of light through volume media (volume shading) and on embedded surfaces (surface shading)



This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Volume Ray-Casting

Algorithm for direct volume rendering. For each pixel of the image to be rendered, compute one on more ray of vision, integrate light along these rays, weight the computed light and assign to the pixel the corresponding color value.

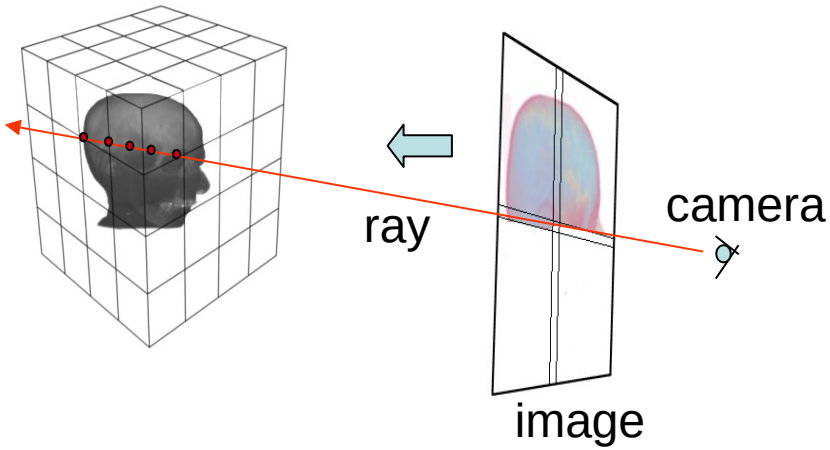


This document is distributed under Creative Commons Attribution Sh se

Volume Ray-Casting

Pseudo code

```
for each pixel (i, j) of the image do
    L = 0, 0, 0                                          (RGB)
    for k in [1…. number_of_rays] do
        ray = compute_ray(k, camera, (i, j))             Geometric computations
        L = L + integrate_light_along_ray (ray, voxel_model)   Shading
    endfor
    color = mapping(L)
    image[i, j] = color
endfor
```



Actually, this is done using GPU programming using shaders

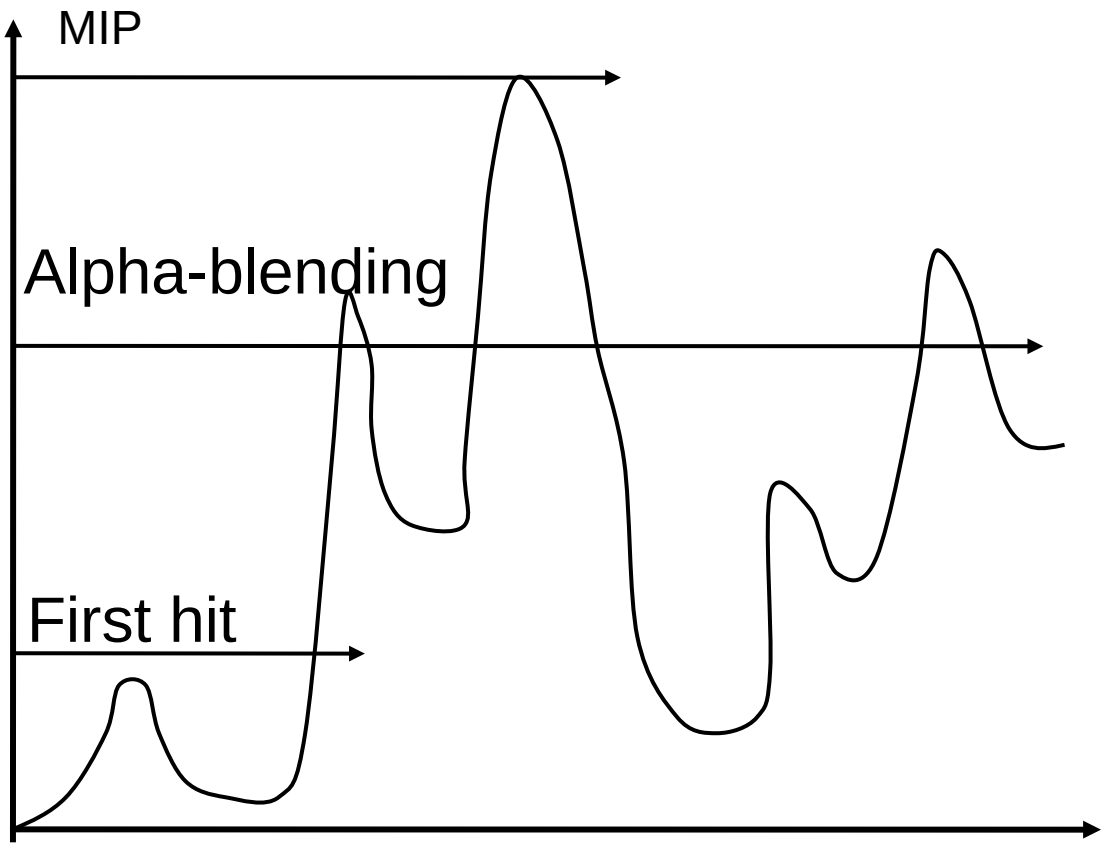
This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Compositing

Maximum intensity projection: only the highest value is shaded and rendered

The intensities are accumulated according to their opacity until the opacity 1 is reached

The first non-empty region (or the one in a given range of property values) is rendered



This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Shading

Various models more or less complex

Volume shading
- emission only
- absorption only
- **emission+ absorption**
- volume scattering (single or multiple)

Surface shading:
- **diffuse reflection**
- **specular reflection**
- glossy reflection

(from last week)

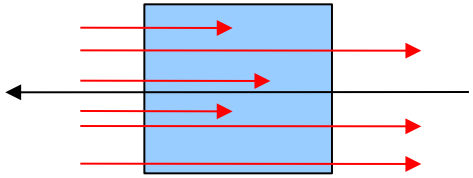
Surface shading is applied at the voxels that are at the boundaries of volumetric regions

This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Volume shading

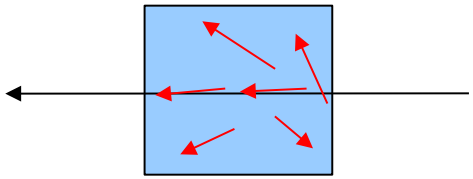
Absorption only

The volume absorbs light
It doesn't emit light, nor it reflect it
The observer perceives the attenuation of light through the volume



Emission only

The volume emits light.
It doesn't absorb it, nor reflect it
The observer perceives the sum of all the emitted light along each vision ray.

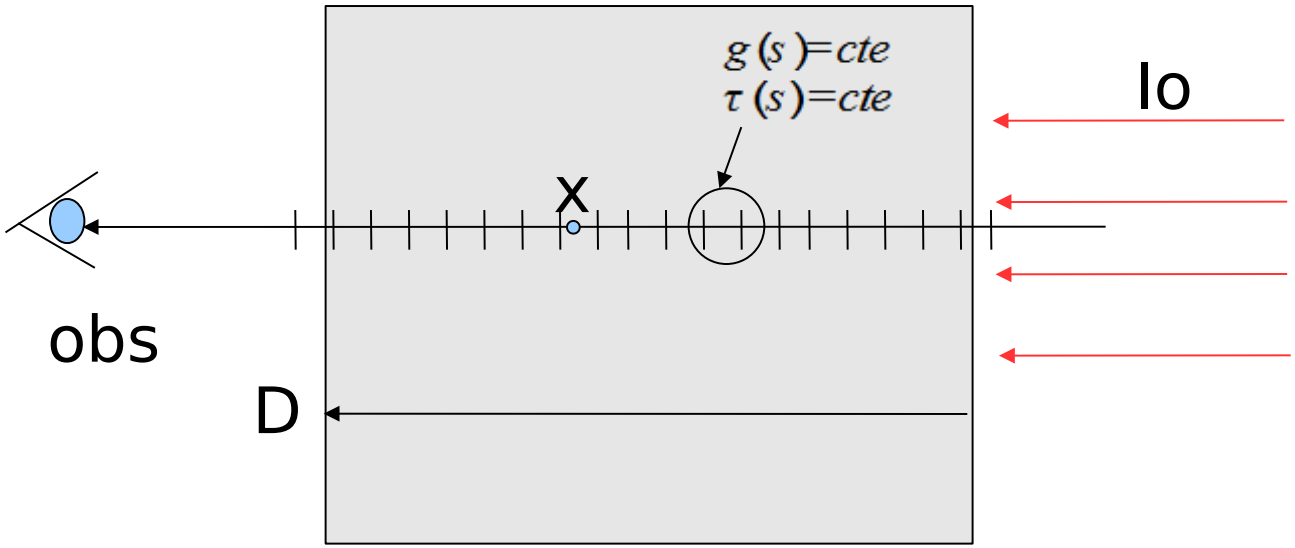


This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Volume shading

Simplification of the volume rendering equation

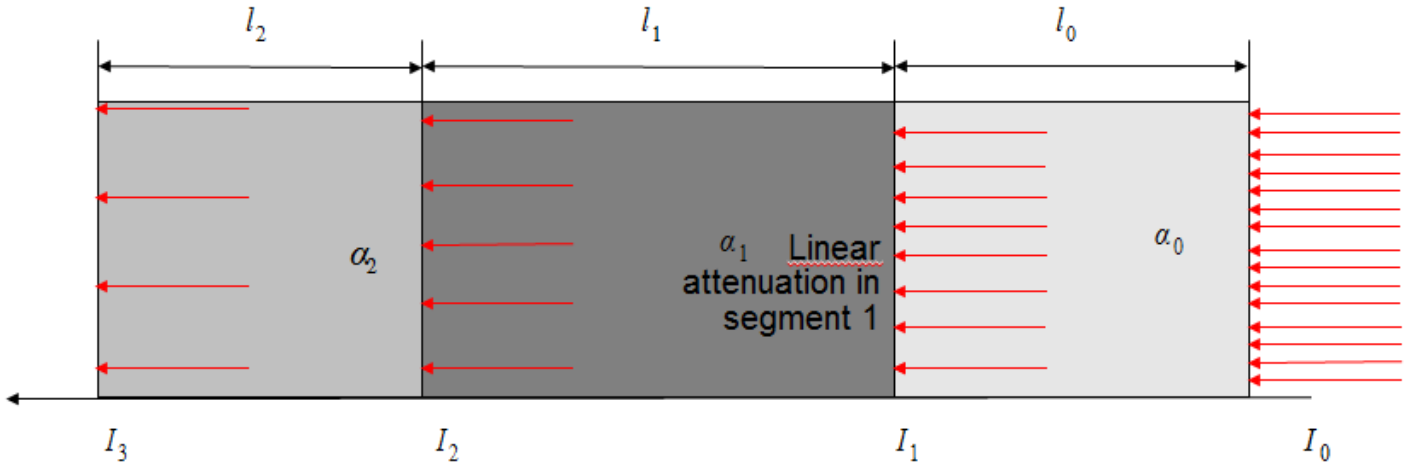
- Emission+ absorption only
- Discretization of the ray into intervals such that the emission and absorption are constant in each interval.
- Linear emission and attenuation along the ray.



This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Volume shading

Absorption only
Linear attenuation- Constant optical property within a sampling interval



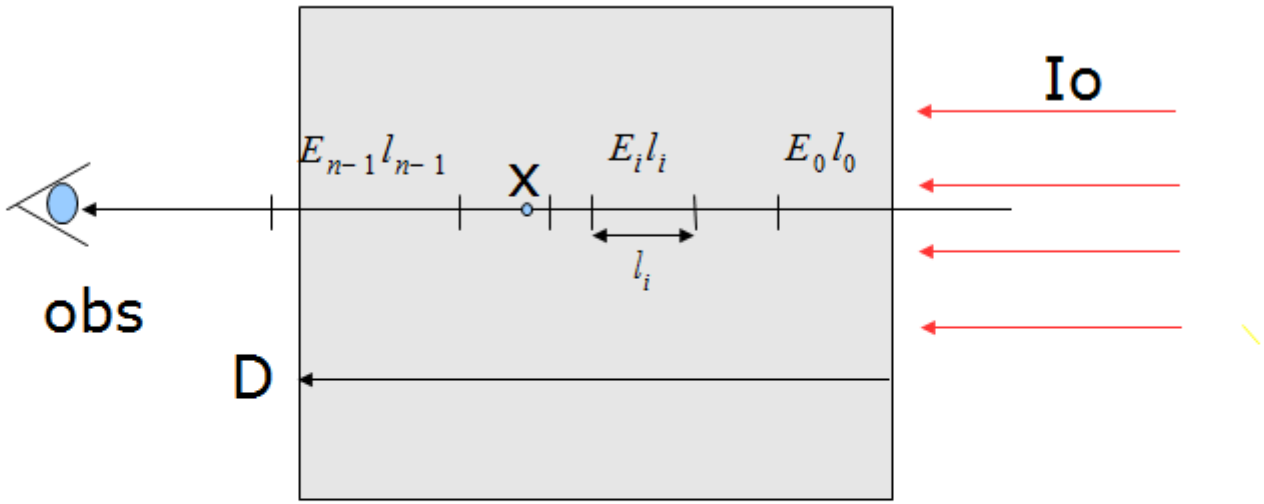
$$I_3 = I_2(1-\alpha_2 l_2) \qquad I_2 = I_1(1-\alpha_1 l_1) \qquad I_1 = I_0(1-\alpha_0 l_0)$$

$$I_n = I_o \prod_0^{n-1} (1-\alpha_i l_i)$$

This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Volume Shading

Emission only: numerical approximation with linear emission



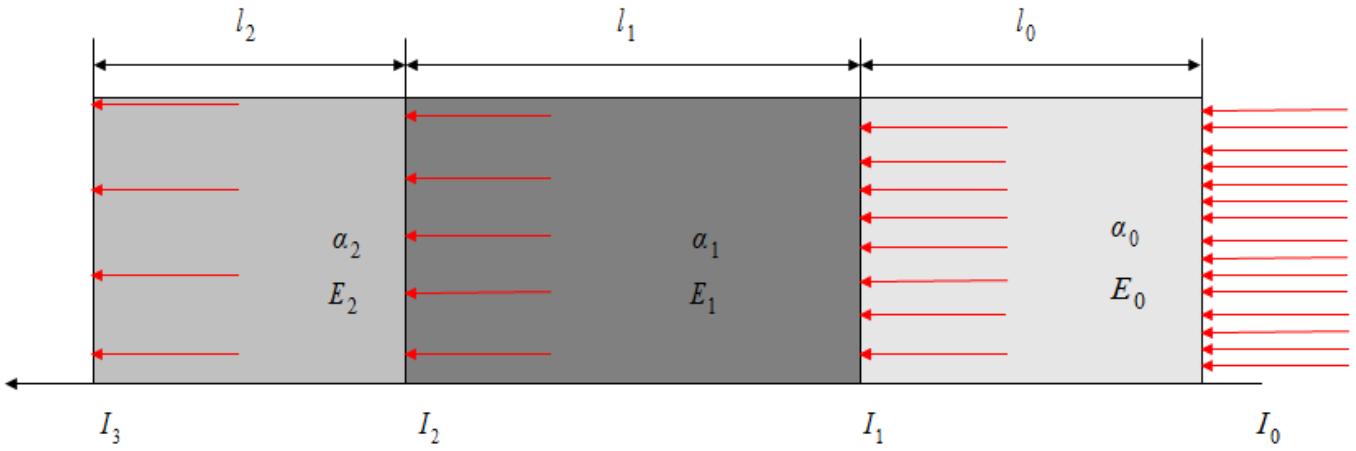
$$I(s) = I_o + \int_0^s g(t)\,dt$$

$$I_n = I_o + \sum_{i=0}^{n-1} E_i\,li$$

This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Volume Shading

Emission + absorption : numerical approximation with linear emission and attenuation



$$I_2 = I_1(1-\alpha_1 l_1) + E_1 L_1 \alpha_1 \qquad I_1 = I_0(1-\alpha_0 l_0) + E_0 l_0 \alpha_0$$

$$I_n = \sum_{i=0}^{n-1} I_0 + E_i l_i \alpha_i \prod_{j=0}^{i-1} (1-\alpha_j l_j)$$

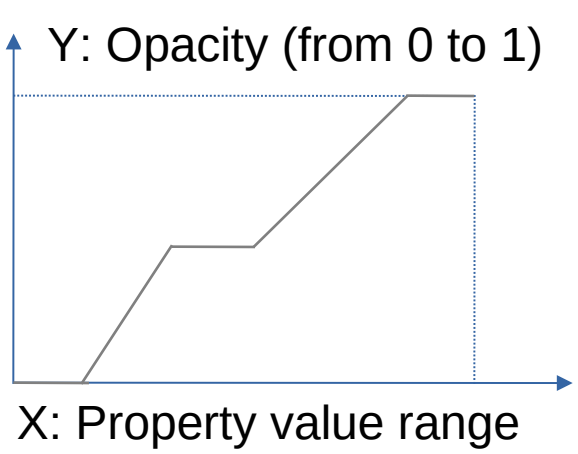
This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

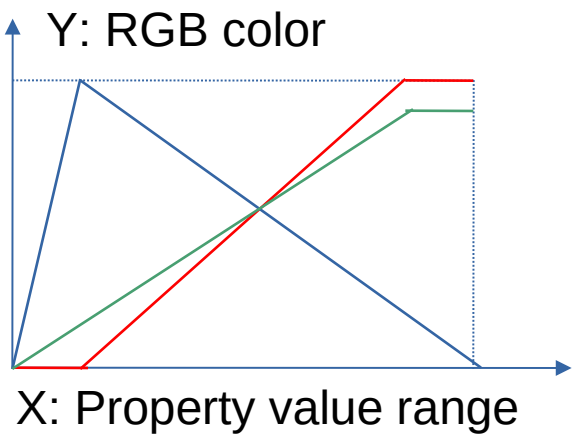
Sample shading

To shade a sample is equivalent to compute its color and opacity and to apply a shading model (by now the emission + absorption model). How do we compute the emission and absorption at the voxel?

We need to **classify** samples: determine to which tissue they belong. Then, we associate to each tissue a given emission and absorption.

A simple way of doing that is value-based classification through 1D **transfer – functions**. We define manually value-based (1D) **transfer functions** that associate to each property value a color (emission) and opacity (absorption).

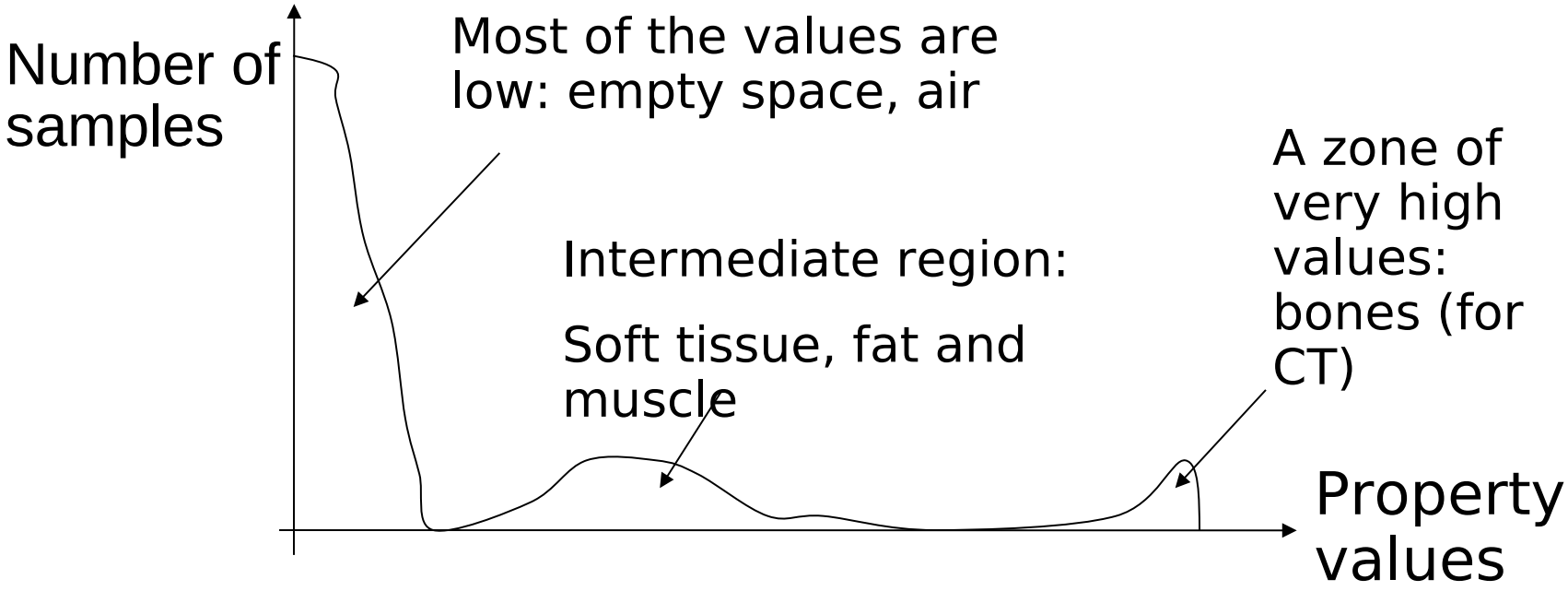




This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

1D Transfer functions

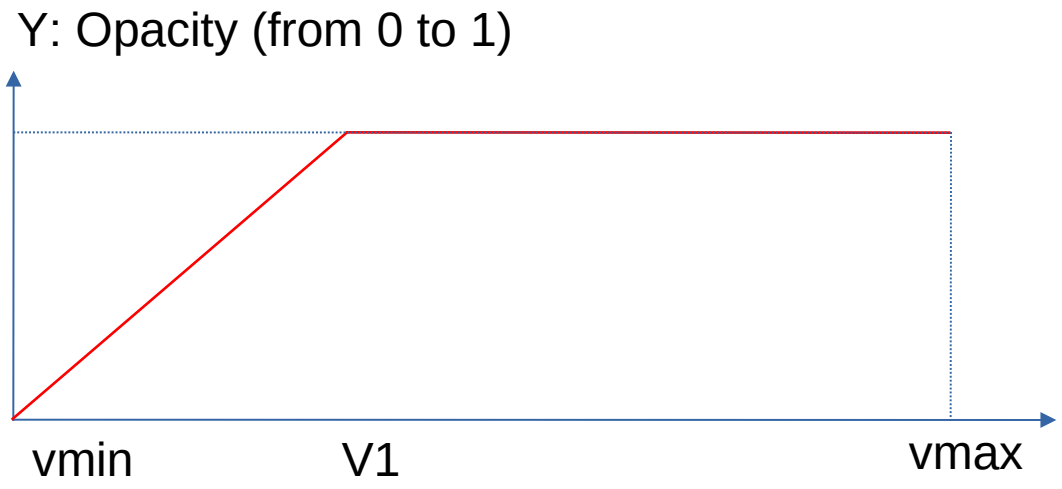
- Manual editing
- Analysis of the histogram



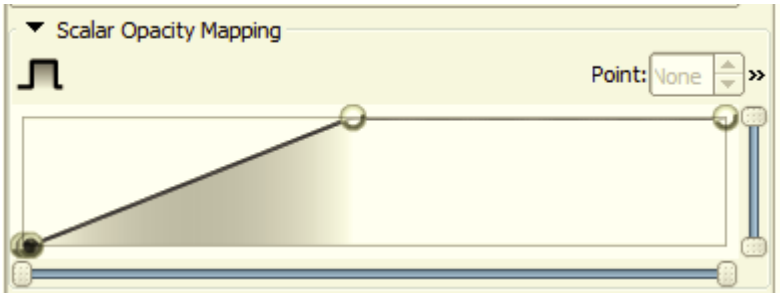
This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Sample shading

Scalar opacity mapping transfer function, an example.



X: Property value range (e.g. 0 to 255 for an unsigned char)

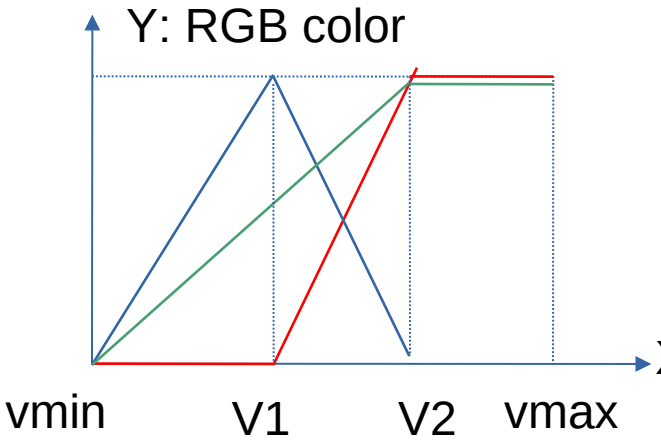


In this example, the opacity of voxels between 0 and v1 increases linearly from 0 to 1. It is 1 for any value between v1 and vmax

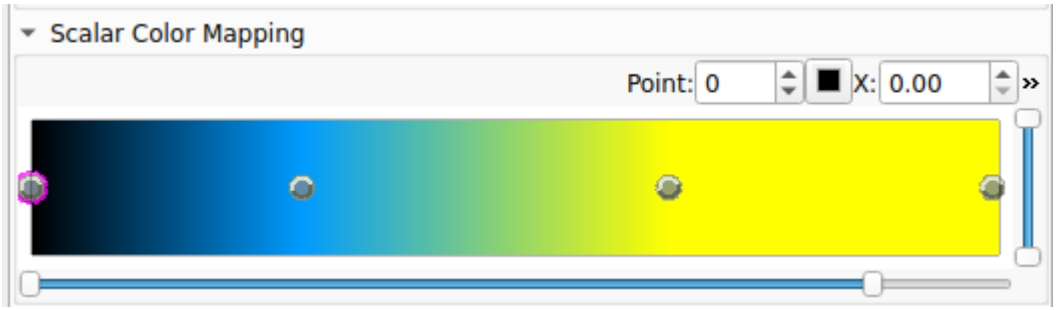
This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Sample shading

Color opacity mapping transfer function, an example.

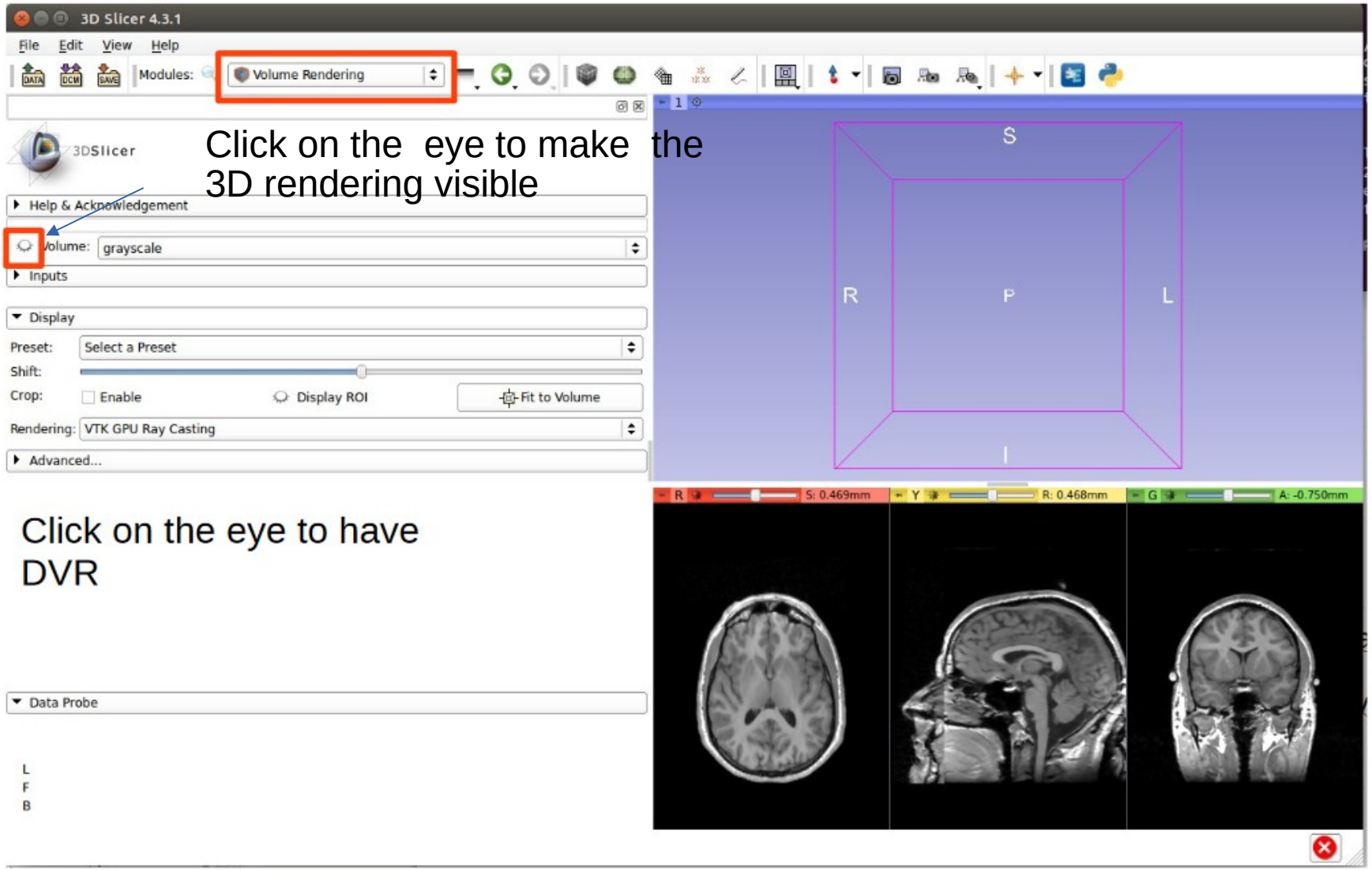


In this example, the color will vary from cyan
It will be pure yellow between v2 and vmax



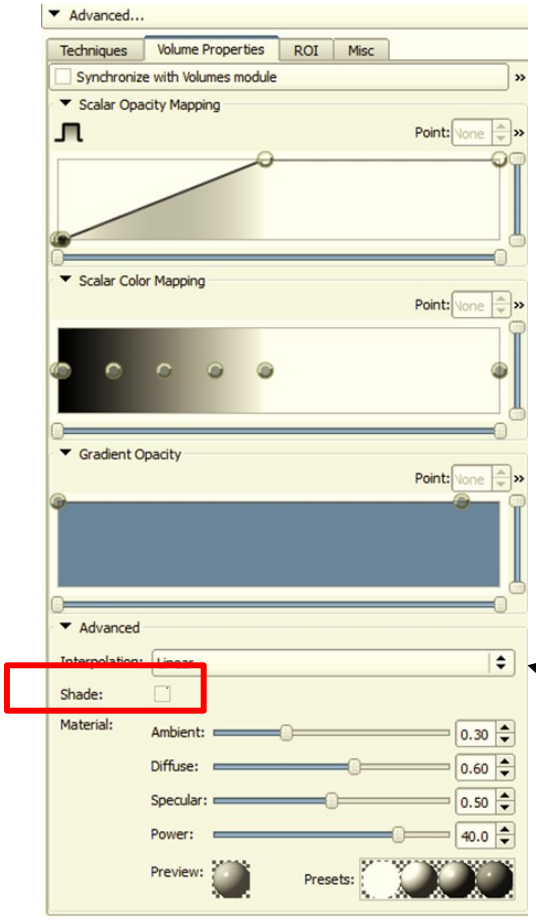
This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Practice



This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Practice



Turn off surface shading

We will work only with emission + absorption by now

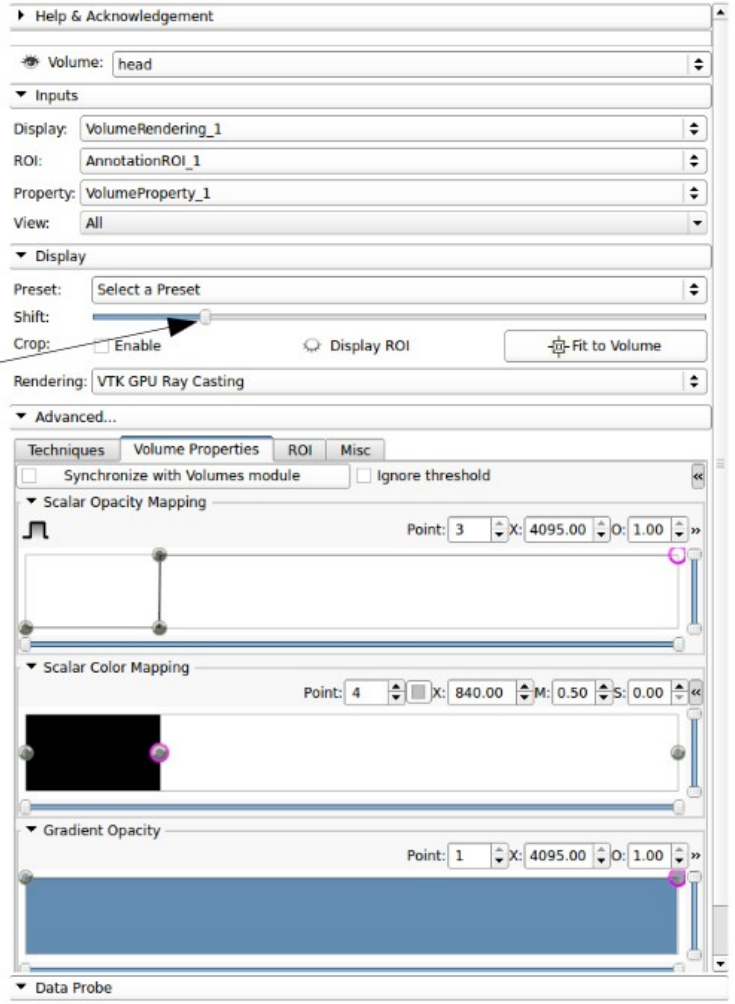
See the type of interpolation used to compute sample values

This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Practice

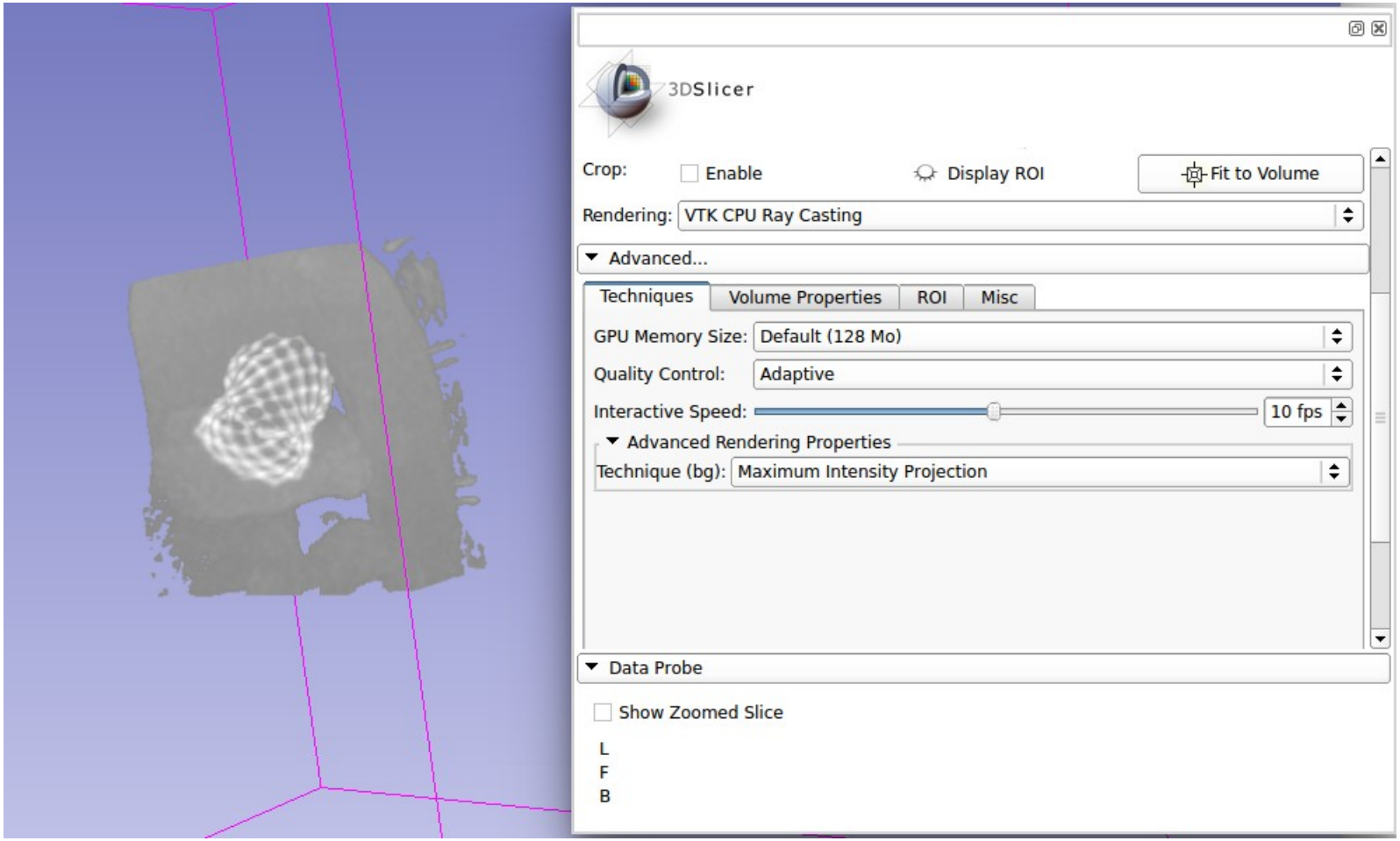
Apply a single-step opacity transfer function

Move the shift to vary the dataset value at whch opacity is 1.0



This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Change the rendering technique



This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Value-based classification

Drawbacks

- The histogram does not represent the spatial distribution of the voxels
- The “surface” or interphase component between materials is lost.
- Need to take into account the gradient, and even the second and third derivative.

Next week we’ll work on that applying surface shading

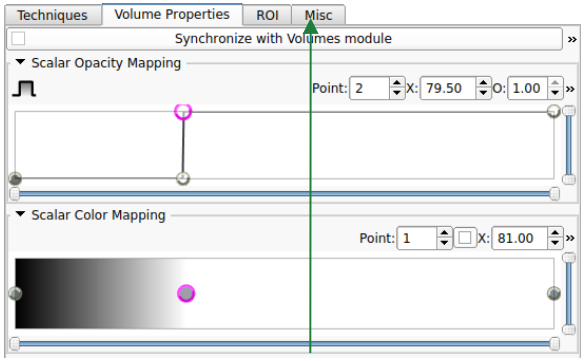
This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Exercise

What is the intensity of light at a point P located in voxel V of value 130 and gradient vector G=(1, 0, 2) taking into account the following rendering conditions:

- nearest-neighbor interpolation => value(P) = value(voxel(P)) = 130

- emission + absorption volumetric shading (see TF at right)



130

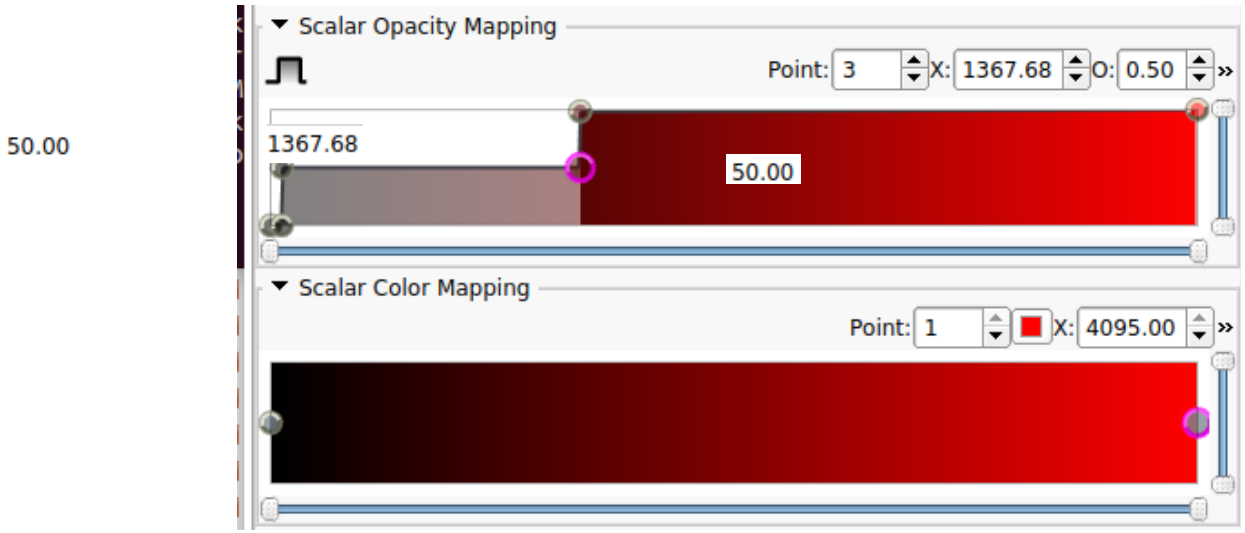
alpha = 1.0, E =(1.0, 1.0, 1.0)

Answer: I(P) = (1.0, 1.0, 1.0)

This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise

Exercise 6

Same exercise with the following transfer function (doubts next week)



This document is distributed under Creative Commons Attribution Share Alike unless stated otherwise