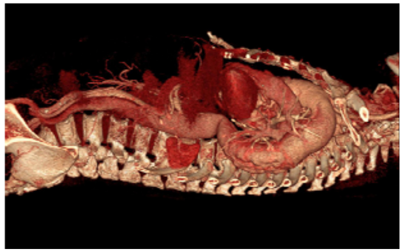
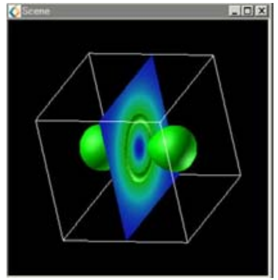
Volume Data

- 3D array of data. Each volume element (voxel) is a value representing some measurement or calculation
- e.g. Magnetic resonance imaging (MRI) data: Strong magnetic field aligns magnetization of hydrogen atoms, and then measures radio waves they emit (body tissue contains a lot of hydrogen (water)). Computed Tomography (CT) data: Interior images of the body are calculated using many X-rays of the body.

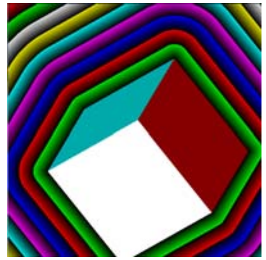


Volume Data

- e.g. simulation: probability of electron position around a H_2 molecule
- Machine representation simple: 3D array of char/integer/real, etc.
- Conversion to volume data: scanned (MRI/CT), numerical simulation (hydrogen data), or distance fields
- Rendering to screen (e.g. MIP – later slides)



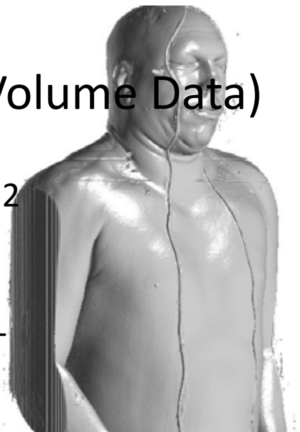
Hydrogen data



Distance around object

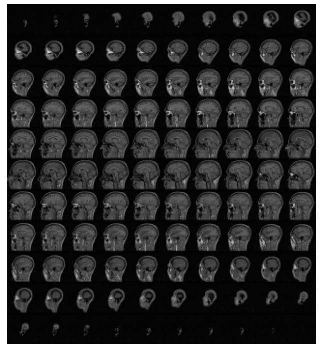
Visible Human Project (Volume Data)

- CT scan 1800+ slices, each 512x512 voxels. Each voxel is 2 bytes (integer) ~ 1GB. MRI scan similar
- Body then photographed at 1800+ 1mm intervals (each photograph represents a slice through the body)



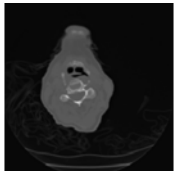
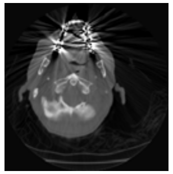
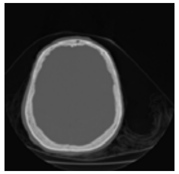
Volume Rendering Input

- The data consists of a number of images representing slices through the body
- Each pixel in each image has the measured quantity (X-ray absorption for CT scans)
- Stacking the images creates a 3D array of “voxels” = volume elements (pixels=picture elements)

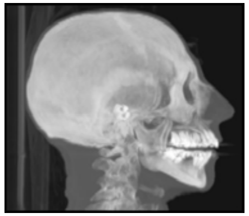


Maximum Intensity Projection

- Type of rendering algorithm for volume data
- See lecture for example application for reading data
- Typical slices from CThead – bit.ly/cthead

Maximum Intensity Projection



- Example for side view from CThead
- Rays are traced through the data set
- For each ray find the maximum value
- Then map it from [min,max] to [0,255]

Maximum Intensity Projection

- Algorithm (for 256x256x113 data set – CThead):

  e.g. for top view, first loop over all the pixels:

```
for j=row 0 to row 255
   for i=column 0 to column 255
now set the maximum to a default value
     maximum=a small value
now for the ray (i,j) going through the data from slice 0 to
   slice 112, find the maximum
     for k=slice 0 to slice 112
              maximum=max(cthead[k][j][i],maximum)
still, for each ray (i,j), now the maximum has been found, set
   the colour of the pixel as usual (as in the code)
```

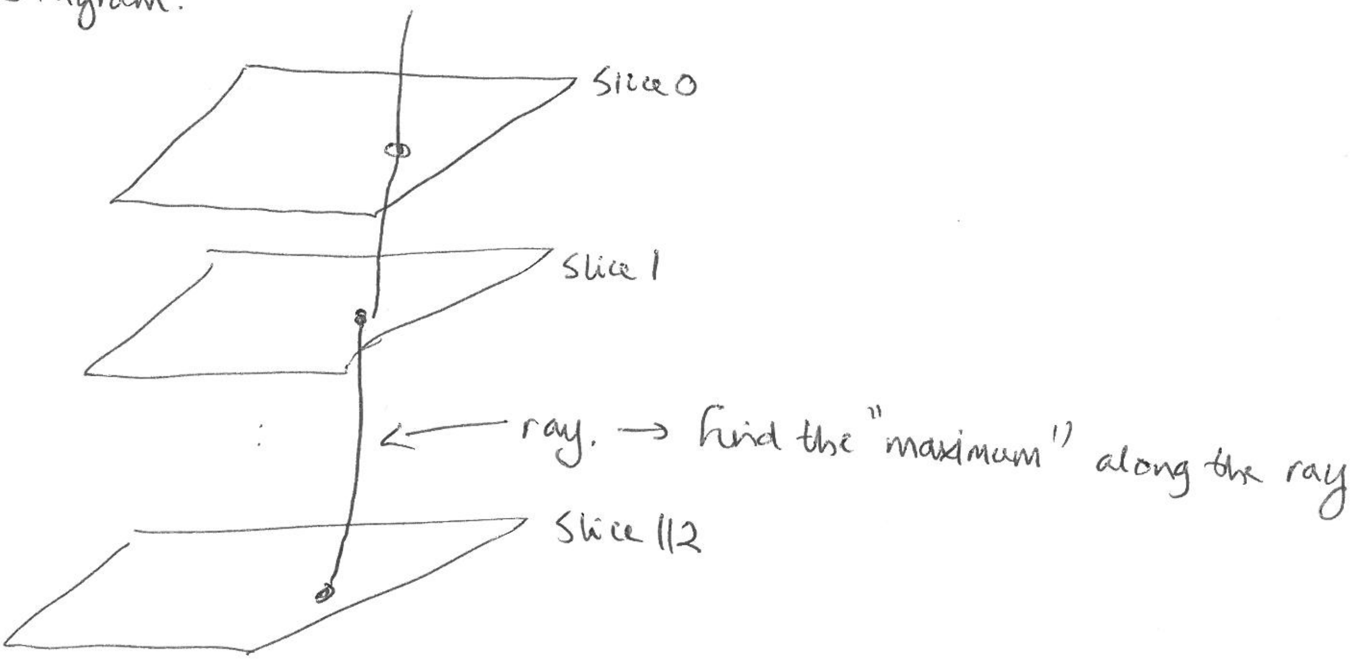
Maximum Intensity Projection

- Further definitions
- each position in the 3D data set is known as a voxel
- a ray is a straight line “fired” through the data set
- Maximum intensity projection (MIP) has the advantage that it is close to a doctors understanding of an X-Ray
- Its disadvantage is that the depth of the structure is hard to distinguish (sometimes depth weighting is used)

MIP

See slides for loops / algorithm. Its your task to convert it into code.

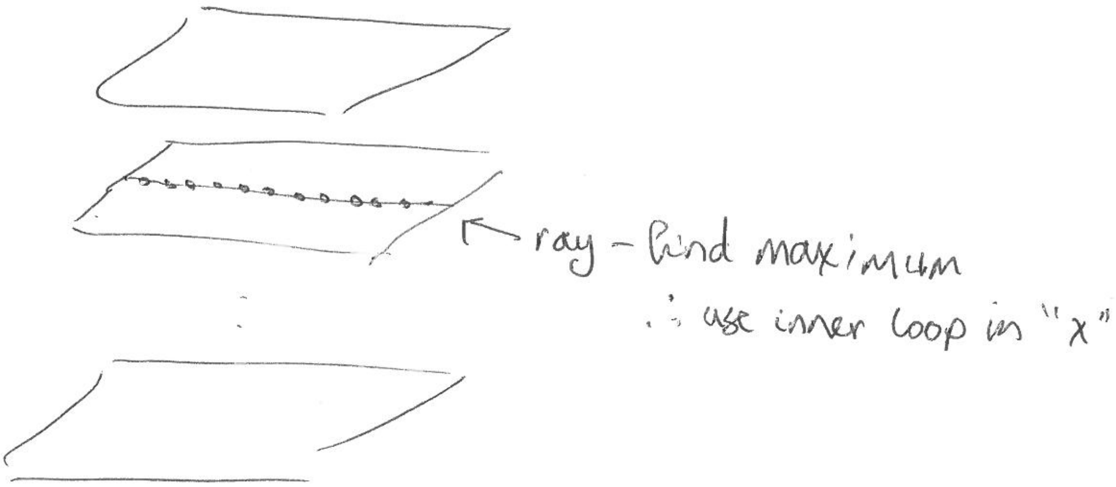
Diagram.



From top down, for a ray going through the slices, we need an inner loop going from 0..112 finding the maximum. (Note there is already a variable called max, so use a different name for this maximum).

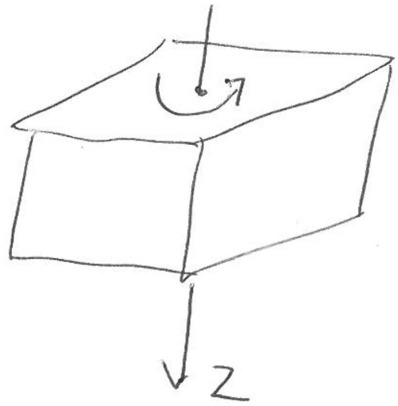
Use that maximum value to give a value for the pixel in the image you wish to display. Example image is given in the notes.

Doing MIP from the side.



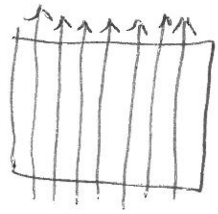
Rotation view. [advanced]

From previous a ray goes in x for side view, z for vertical view, y for front view. For a rotation about z axis

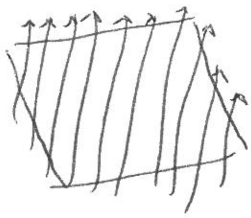


we need to work out how the rays go through a slice.

Let's look at one slice and no rotation (0°)



and say 25°



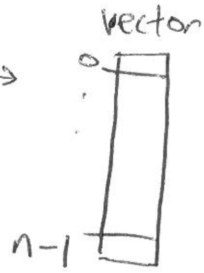
Now recall 2D rotations from GCSE

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

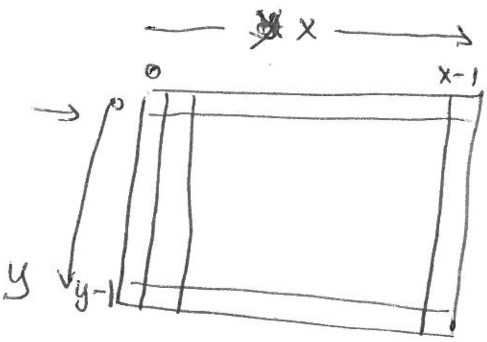
This is a hint how to do it.

3D Arrays.

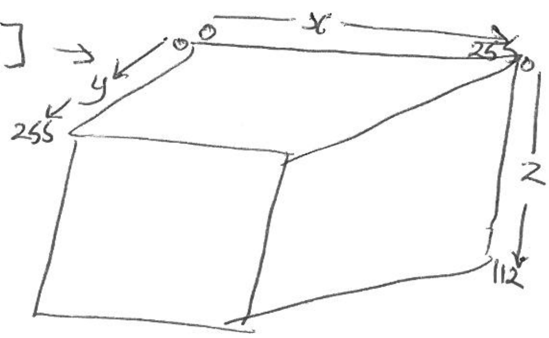
1D vector[n] →



2D image[y][x] →



3D cthead[z][y][x] →



cthead[k][j][i] k indicates a single cross sectional slice
(i, j) is a single pixel/voxel/element within that slice.

(Advanced)
(of course its all a 1D vector of memory (i + j*width + k*width*height)*sizeof(short))

fixed
So cthead[z][j][i] give a slice looking down (the z direction).
loops

fixed
cthead[k][y][i] gives a slice looking from front (the y direction)
loops

cthead[k][j][x] gives a slice looking from the side (the x direction)
loops fixed