SVVR Assignment 2

Sander Nugteren (6042023) Merel de Groot (6103677)

November 17, 2014

1 Visualization Pipeline

The assignment is to create a visualization pipeline with VTK. The source will be a dataset consisting of images from a CT scanner. These 2D images combined form a 3D image of a body. Using VTK we will create a pipeline that takes the dataset as input and creates a 3D visualization of the body.

Source

The data consists of 94 images created by a CT scanner. Each file is 131,072 bytes and the images are 256x256 or 512x512 pixels in size. Since 131,072 devided by 512x512 is only a half, we know that this is not possible. Deviding 131,072 by 256x256 however, gives us 2 bytes, which is a reasonable size of data per pixel. The collection of binary images is read by using VTKImageReader2.

By default voxels are isotropic, but this has to be adjusted to get a proper view of the head. In this case the voxels need to be adjusted to $(1, 1, 2)$ to make the data spacing more accurate and prolong the head to a reasonable size.

Isosurfaces

Next a contour filter is used to create the isosurface. The range of the values is $[0, 65535.0)]$. The suggestion to start with a halfway value (about 30000) only renders a jaw, i.e. the dense bone. But by lowering the value softer tissues become visible. Around 700 the skin is smooth, showing the soft tissue of the body and when increased to 1200 we can see the bones. So the isosurfaces correspond to the density of the material. Then, the data from this filter is given to the mapper to create graphical primitives.

Colours

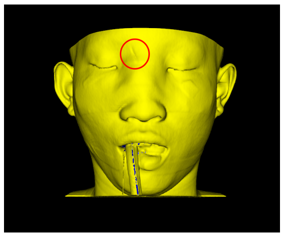
Unfortunately VTK maintains a default of a dark blue object on a black background. Therefore the colours are adjusted in the mapper stage. There are options to set these colours manually, but to have the colours change over the isosurfaces, we chose to adjust the scalar range of the colour mapping.

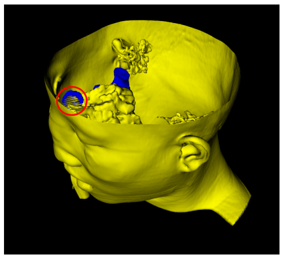
Setting the Scene

The renderer configures the visualization in the window, so in this stage a camera is added to define our point of view. As a default setting we decided to point the camera straight towards the face of the victim. We also tried to configure the size of the renderer window, but using the FullScreenOn()-function caused the window to be uncloseable. Stretching out the window is still possible, so it's just a minor inconvenience.

2 Results

In figure 1 we see two isosurfaces displayed. The head is displayed from two angles, so that we can see both surfaces in figure 1b. In figure 1a we see a tiny scar. In figure 1b we see that behind this scar a tiny blue object is lodged inside the skull. This might be what killed our victim. For clarity's sake, we displayed the blue parts without the skin in figure 1c. These images are generated with the script ctvis_static.py. We also included a script that animates the different isosurfaces (ctvis.py).


(a) Frontal picture of the head. Notice the scar circled in red.


(b) Foreign object lodged into head at scar position.

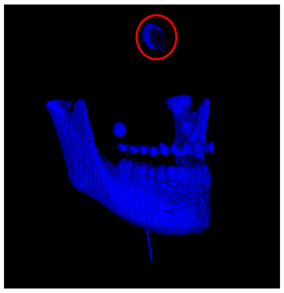

(c) Jaw and foreign object only.

Figure 1: Skin is displayed in yellow (isosurface of 700), dense bones and foreign object displayed in blue (isosurface of 3000)