Technische Universiteit Eindhoven

Visualization

2IV35

Volume rendering

Author:
Sander Kools
0848523
s.w.a.kools@student.tue.nl

Author:
Luuk Hulten
0720248
l.a.j.v.hulten@student.tue.nl

January 11, 2015

Contents

1 Speeding up interaction **3**

2 MIP **4**

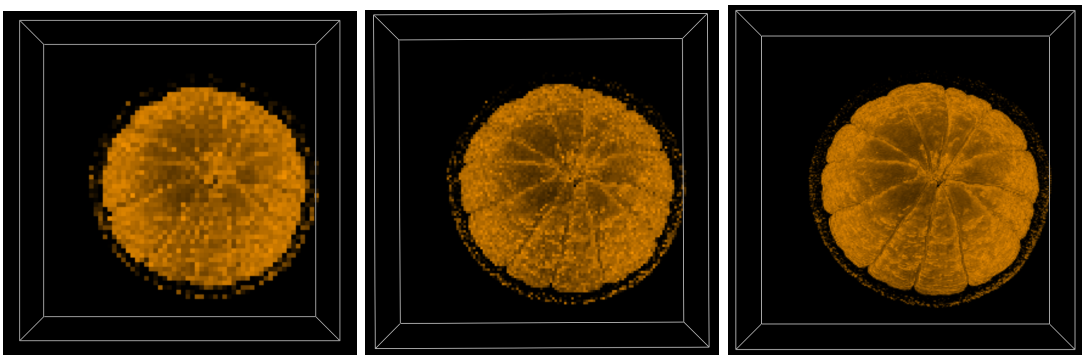
3 Trilinear Interpolation **6**

Introduction

In this report we will describe some implementation details of the volume rendering application. We will show the advantages and disadvantages of some rendering techniques at the hand of images. In section 1, we will give some details about rendering in a lower resolution to speed up interacting with the application.
In section 2, we will discuss the MIP rendering technique In section 3, we will discuss the Compositing rendering technique In section 4, we will discuss the Opacity weighting rendering technique Finally in section 5, we will give a short conclusion.

1 Speeding up interaction

Since most visualization techniques are quite slow, interacting with the visualization becomes laggy and difficult. To speed up these interaction we render the image while interacting in a lower resolution. We implemented the rendering to render the image in three different resolutions. These resolutions are $\frac{1}{4}$ and $\frac{1}{2}$ of the original resolution. At last we render the image at full resolution. This enables the user to interact with the image and see some result in low detail. For example when rotating the users sees the image in $\frac{1}{4}$ of the resolution. When he stops rotating the image is rendered at $\frac{1}{2}$ of the resolution this image is well enough for the user to determine if this is the view he liked. After the calculation is completed the image is rendered at full resolution. Normally calculating in three resolutions introduces extra overhead but since intermediate results are stored little overhead is introduced. Also the calculation of the pixels is done multi threaded this means that the user can interact with the visualization while pixels are been calculated. While interacting while pixels are calculated the pixel queue is cleared and filled with the pixels of the new visualization.

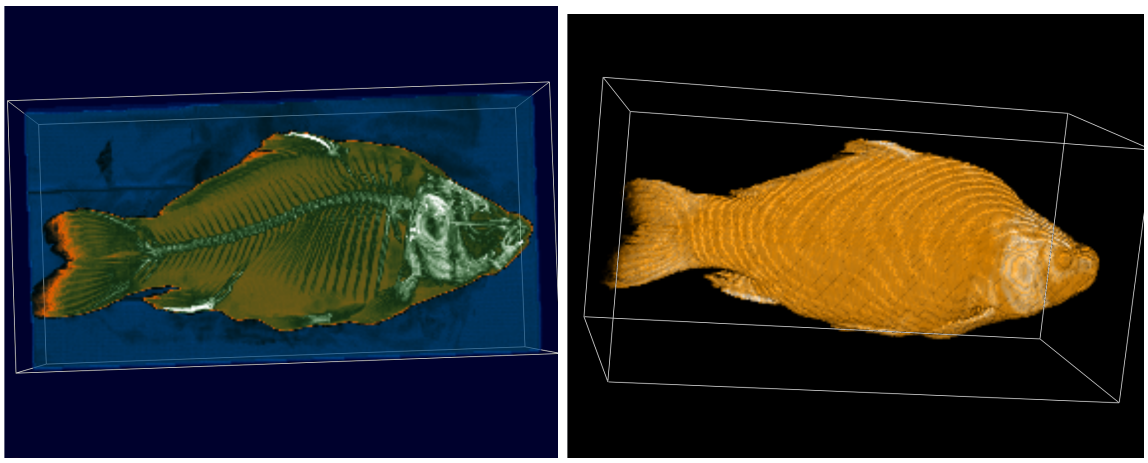


(a) $\frac{1}{4}$ Resolution (b) $\frac{1}{2}$ Resolution (c) Full resolution

Figure 1: Pie Charts

2 MIP

In scientific visualization, a maximum intensity projection (MIP) is a volume rendering method for 3D data. It consists of projecting the voxel with the highest attenuation value on every view throughout the volume onto a 2D image.
This method tends to display bone and contrast materialfilled structures preferentially, and other lower-attenuation structures are not well visualized. The primary clinical application of MIP is to improve the detection of pulmonary nodules and assess their profusion. MIP also helps characterize the distribution of small nodules. In addition, MIP sections of variable thickness are excellent for assessing the size and location of vessels, including the pulmonary arteries and veins.
For the implementation of the MIP we sampled every pixel by casting a ray through the data and set the voxel to the maximum value. First we iterate over every pixel. At every pixel we record the corresponding voxel value and if it's the highest value we've encountered, we save it as being the highest value until we possibly find a higher one later in the iteration.
Below in figure 2a there is an example of a MIP rendering using the fish dataset and the composition as comparison in figure 2b.



(a) Fish MIP rendering (b) Fish Composition rendering

Figure 2: Fish rendering, MIP and composition

We see that MIP works well when the datasets have objects with high density, such as the Fish or the Skeleton, for instance in figure 3 you are able to see the contents of the Piggy bank using this rendering form, which wasn't able with the composition rendering.

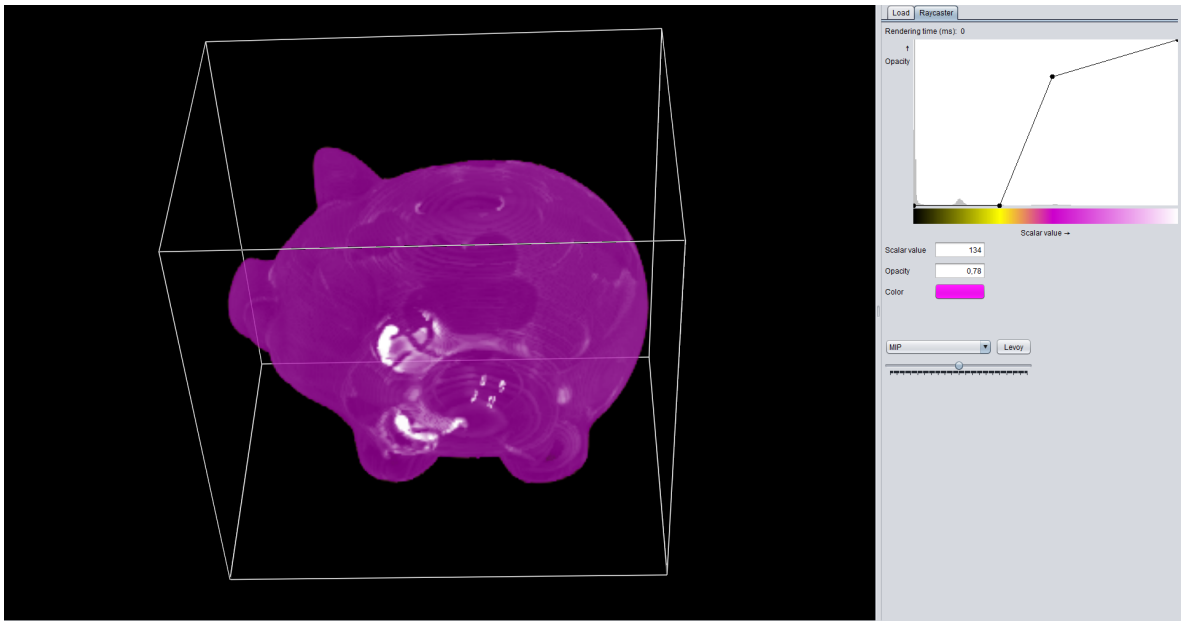
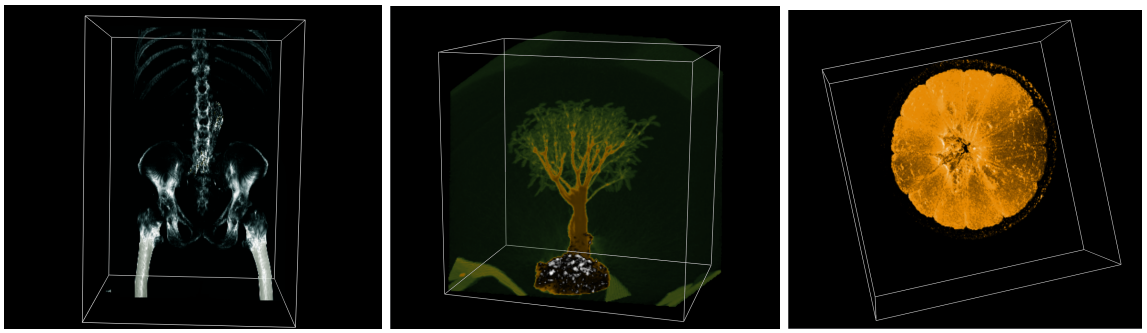


Figure 3: Pig MIP



(a) Skeleton MIP (b) Bonsai MIP (c) Orange MIP

Figure 4: Multiple MIP renderings

3 Trilinear Interpolation

We also implemented tilinear interpolation, this allows us to be able to get an approximate value of a voxel, which ordinarily might not correspond with the raw data points.
Trilinear interpolation is the extension of linear interpolation and bilinear interpolation and operates in 3 dimensions, in figure 5 you can see an example representation of trilinear interpolation.
Below is the algorithm to compute the point c in figure 5 that we used in the program.

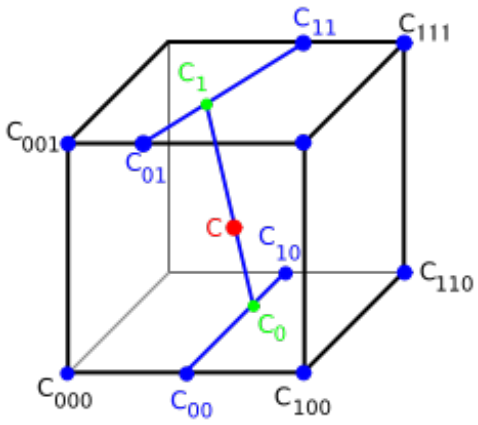


Figure 5: representation of trilinear interpolation

$$
\begin{aligned}
x_d &= \tfrac{x-x_0}{x_1-x_0} \\
y_d &= \tfrac{y-y_0}{y_1-y_0} \\
z_d &= \tfrac{z-z_0}{z_1-z_0} \\
c_{00} &= V_{x_0,y_0,z_0} \cdot (1-x_d) + V_{x_1,y_0,z_0} \cdot x_d \\
c_{10} &= V_{x_0,y_1,z_0} \cdot (1-x_d) + V_{x_1,y_1,z_0} \cdot x_d \\
c_{01} &= V_{x_0,y_0,z_1} \cdot (1-x_d) + V_{x_1,y_0,z_1} \cdot x_d \\
c_{11} &= V_{x_0,y_1,z_1} \cdot (1-x_d) + V_{x_1,y_1,z_1} \cdot x_d \\
c_0 &= c_{00} \cdot (1-y_d) + c_{10} \cdot y_d \\
c_1 &= c_{01} \cdot (1-y_d) + c_{11} \cdot y_d \\
c &= c_0 \cdot (1-z_d) + c_1 \cdot z_d
\end{aligned}
$$