IN 4086-14 Data Visualization
Volume Visualization Project: Group 26

Dimitropoulos Georgios: 4727657
Manousogiannis Emmanouil :4727517
Mastoropoulou Emmeleia: 4743539

February 4, 2018

1 Introduction

The goal of our project is to develop and extend a volume renderer. Our work consists of three main parts. Firstly, the theoretical background of these methods is provided in combination with a connection to the corresponding implementation of our source code. In the sequel, the illustration of these methods with a variety of data samples is given. Specifically, the first part is about Raycasting and the second one concerns 2-D Transfer functions and Shading. More specifically for the Ray Casting part, the following functionalities have been implemented : 1) Tri-linear interpolation for the samples along the viewing ray. 2) Maximum Intensity Projection (MIP). 3) Compositing ray functions. 4) Responsiveness improvement during user interaction. As far as as the 2-D Transfer functions and Shading part is being concerned, the following implementations have been considered: 1) Implementation of a gradient-based opacity weighting in our ray-caster. 2) Implementation of the Phong shading illumination model. Finally, in the third part, an extension of the basic volume renderer is proposed.

The rest of the paper is organized as follows: Section 2 refers to the Raycasting, where Tri-linear interpolation, MIP, Compositing and responsiveness improvement are being considered. Section 3 presents the 2-D Transfer functions and Shading, where the implementation of a gradient-based opacity weighting and the implementation of the Phong shading model are analyzed. Section 4 presents an extension of the basic volume renderer. Section 5 provides the results of our work, followed by an analysis of the functionality of the parts which we were implemented. Section 6 summarizes our work, where interesting conclusion and future work proposals could be depicted.Finally, in Section 7 the way of our work is mentioned.

2 Ray Casting

In this section, we provided the direct visualization of the volume data (Direct Volume Rendering), without transformation into geometric primitives. The method we follow is the image order technique ray casting. For this part the following functionalities have been implemented: 1) Tri-linear interpolation of the samples along the viewing ray. 2)MIP. 3)Compositing ray functions. 4)Responsiveness improvement.

2.1 Tri-linear Interpolation for the Samples Along the Viewing Ray

Based on the skeleton code that was provided, we implemented the function getVoxelLinearInterpolate() of the class Volume so that it takes the coordinates of a point as input and returns the interpolated value through Tri-linear interpolation of the volume.

2.2 Maximum Intensity Projection (MIP)

This functionality is about the implementation of the Maximum Intensity Projection method. We implemented the traceRayMIP() function of the RaycastRenderer class, which is called for every pixel of the image to get the color. Based on the number of samples on the ray vector with a fixed sample step, it computes the value of the point where the highest intensity occurs. Based on that value, the color pixel is returned as an integer. Considering this value and by using the transfer function, we map this value with its corresponding color and opacity.

2.3 Compositing ray functions

For this task, we implemented the traceRayComposite() function in the RaycastRenderer class. We got samples on the direction of the ray vector starting from the exit point and ending to the entry point, (back to front) of the ray. Due to the fact that formulas were significantly simpler, we chose the back to front method instead of front to back. The function traceRayComposite() is being called for each pixel and returns the accumulated value of the pixel according to the formula: $C'_i = A_i * C_i + (1 - A_i) * C'_{i-1}$ with $i = 1, ..., n$ and (C_1, A_1) the color and the opacity of the exit point and (C_n, A_n) the color and the opacity of the entry point. C'_i is the accumulated color of a sample point i by taking all the samples into account. The transfer function maps each value of the sample i to an opacity A and a color C_i.

2.4 Responsiveness improvement during user interaction

Our ray-caster seems to be rather slow and is not responsive , especially when we are in compositing mode. In order to overcome this difficulty we can rather reduce the image resolution. However this, could also cause a significant reduction in the quality of our rendering. For this reason we make use of the **interactiveMode** of our RayCastRenderer class,in raycast() function.When we are in interactive mode, meaning when the user is rotating or changes the angle of view of the image, then we increase the sample step of the raycasting. However, when we are not in this mode, the image remains in its full resolution.

3 2-D Transfer functions and Shading

In the second part of this assignment, we integrate the gradient information into our transfer functions. We provided the indirect 3D visualization of the volume data through isosurfaces(Indirect Volume Rendering). More specifically, we followed the next two steps: 1) Implementation of a gradient-based opacity weighting based on [1] and 2) Implementation of the Phong shading Illumination model.

3.1 Gradient-based opacity weighting

Firstly, we implemented the function compute() in the GradientVolume class. This function calculates the gradients of all data points. We need those gradients, as we will use them to find the normal vector of the isosurfaces (triangles) through interpolation. Secondly, we implemented the function interpolate() in the same class which takes as an input two given gradients and the position of a sample point among them and computes the gradient in this position. This function is extensively called in the getGradient() function, which is implemented in the same class, where we perform Tri-linear interpolation to find the gradient of an inside point of the cube. The opacity $a(x_i)$ of each sample point is determined by using its gradient size $|\nabla f(x_i)|$ and the voxel value $f(x_i)$. The formula for this algorithm is presented in [1] and is as follows:

$$a(x_i) = a_v \begin{cases} 1 & |\nabla f(x_i)| = 0 \cap f(x_i) = f_v \\ (1 - \frac{1}{r} * \frac{|f_v - f(x_i)|}{|\nabla f(x_i)|}) & |\nabla f(x_i)| > 0 \cap f(x_i) - (r * |\nabla f(x_i)|) <= f_v <= f(x_i) + (r * |\nabla f(x_i)|) \\ 0 & otherwise \end{cases}$$

where f_v is the material value and r is the desired thickness. The above method had been implemented through the computeLevoyOpacity() function of the RaycastRenderer class.

3.2 Phong shading illumination model

Phong shading adds an extra dimension to the image. The structure of the surface can be viewed in detail. Phong shading has three different components namely ambient, diffuse and the specular light which are illustrated in the results and evaluation section. The ambient reflection only depends on the color of the surface. The intensity of diffuse reflection depends on the angle of the light ray with the surface, on the reflectivity and on the color of the surface. The intensity of specular reflection depends on the same factors as the diffuse reflection depends on, plus the shininess of the material and the viewer's direction in addition. We have implemented this algorithm in the computePhongShading() function in the RaycastRenderer class.

4 Extension of the basic volume renderer

In this part we tried to find a useful extension or improvement to our basic volume renderer.We mainly focused on the extension and improvement of our shading, which is implemented using the Phong model.As noticed during our experimentation, and after reading related research , we realized that the Phong model suffers from a high computational cost. This makes it slow, and sometimes difficult to perform. A main reason that the computational cost of this model is so high is that it takes into account diffuse,specular and ambient coefficients in order to determine the total shading.
From those three, the specular coefficient requires exponential computational time , since it is dependent on the $cos(\phi)^n$ parameter, where ϕ is the angle between the Reflect vector and the View vector and n is a constant indicating the shininess factor.
In order to reduce this computational cost, we provide an alternative, introduced by Christophe Schlick in the paper "A Fast Alternative to Phong's specular model" [2]. In this alternative we compute the specular coefficient as follows:
$\frac{t}{n-nt+t}$,
where t is the $cos\phi$ of the angle between the Reflect vector and the View vector.This computation requires just 1 addition, 1 multiplication, 1 subtraction and 1 division.
Below we provide two shaded images (figures 1-2) in the Composite mode,where the quality of shading seems to be equal, with much lower computational cost.This may have a significant impact in large data samples.

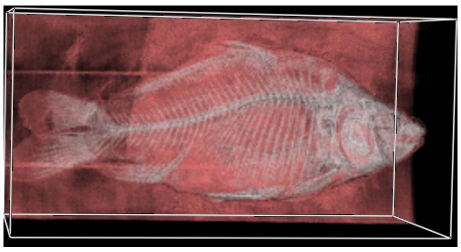


Figure 1: MIP/Phong shading

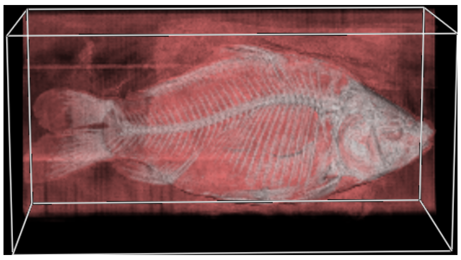


Figure 2: MIP/Alternative specular calculation

Apart from the above, we deployed a complete alternative of the Phong shading model, named Blinn shading model [3]. The function is developed in RayCastRenderer class and its name is **computeBlinnShading**.The function is not currently used, but it can replace the traditional Phong function we implemented if the developer decides it is useful.The Blinn model is considered to be computationally more efficient specially with some specific conditions,where the viewer and light are treated to be very remote, such as approaching or at infinity.

5 Results/Evaluation

5.1 MIP

In figure 3, the result of the slicer mode on the carp8 dataset is shown whereas in figure 4 the result of the MIP mode same dataset is presented. The skeleton of the carp, whose sample values are high, is more white, while the skin, whose sample values are not so high is more transparent. Hence, this method seems very useful for the study of the skeleton of the fish.

However, MIP method is not always useful as it can be depicted from figure 5. In case that the dataset does not have an inner structure with high intensity values, this method does not provide insightful information.

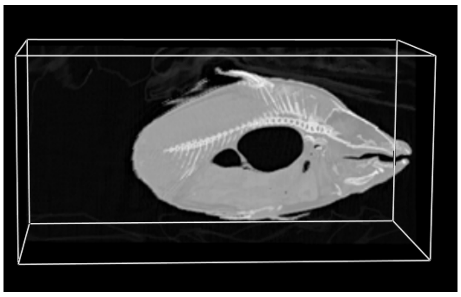


Figure 3: Slicer/Carp8 Dataset

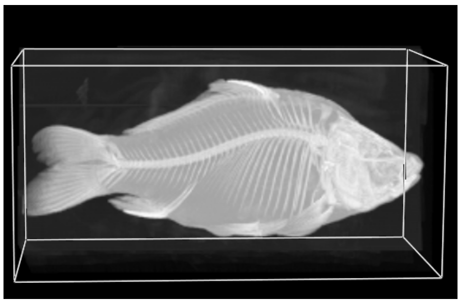


Figure 4: MIP/Carp8 Dataset

In general, MIP functionality is relatively fast and interactive.Hence it is easy to experiment with this method and analyze the objects on different perspectives.

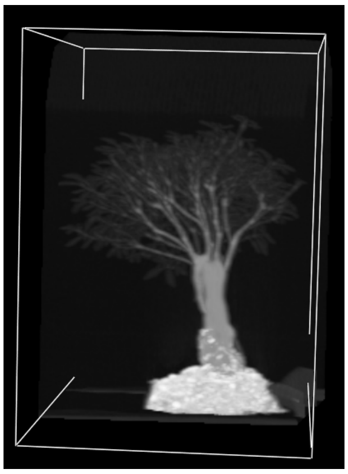


Figure 5: MIP/Bonsai Dataset

5.2 Compositing

The transfer function mode of the composite method is also interesting. By editing the transfer function we also have the option of highlighting the different intensity surfaces. In figure 6, we present the results of this method on the carp dataset. The information we can access is the same, but now we emphasize and focus on any part where maximum intensity occurs.

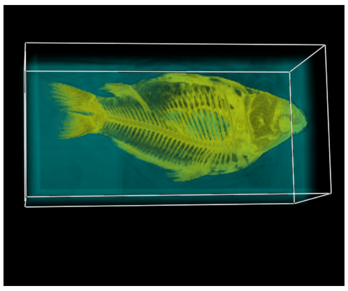


Figure 6: Composite/Corp Dataset

Furthermore, In figure 7 and 8 results of the compositing functionality on the pig dataset are presented. The transfer function that we used is also shown in figure 8 . In figure 7, only the outside layer of the piggy bank can be seen, while in figure 8 some patterns on the surface of the object as well as the coins inside it can be depicted. Also in figure 9 the results of the compositing functionality on the Bonsai dataset is depicted.

It is easily observed that this method provides more flexibility than MIP. Apart from that, we have the option to see other levels of the object, in contrast to MIP. In addition, the interpretation of the sense of depth is significantly better in compositing than in MIP. Finally, while MIP is computationally much more efficient, compositing method is still usable and responsive.

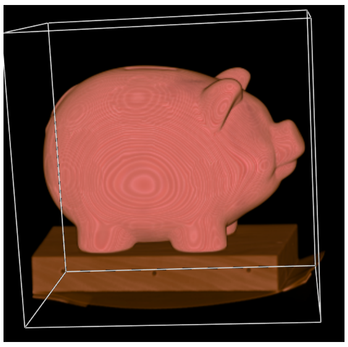
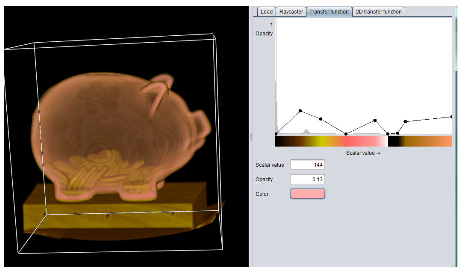


Figure 7: Composite 1/Pig Dataset



Scalar value 144
Opacity 0,13

Figure 8: Composite 2/Pig Dataset

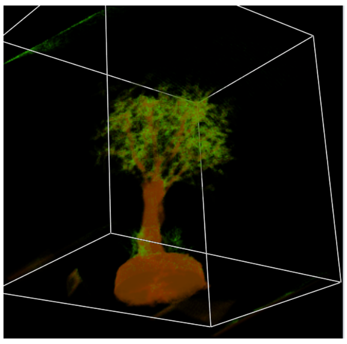


Figure 9: Composite/Bonsai Dataset

5.3 2D - Transfer Function combined with shading

Apart from the above, we explored the insights we could gain by using the 2D-transfer function in combination with the volume shading. The shading, provides a lot of new information and a realistic sense of depth. Furthermore, the different intensity layers could also be viewed and examined. The result of this technique are presented on the carp dataset where the power of each component of shading is illustrated in figures 10-13.

On the downside, this method is not only computationally expensive, but also the responsiveness in case that image is being rotated is not very high.

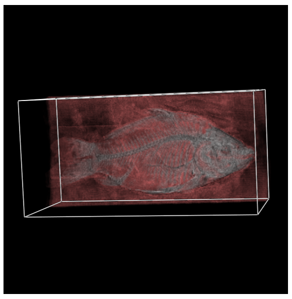


Figure 10: Ambient

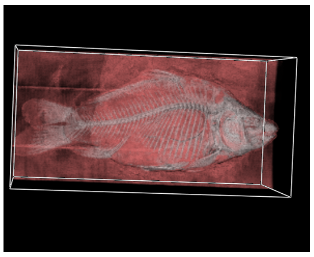


Figure 11: Diffuse

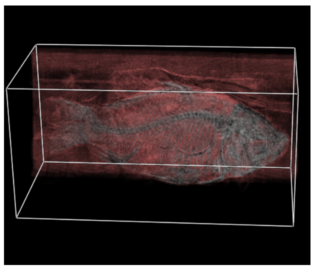


Figure 12: Specular

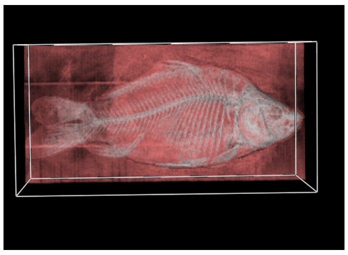


Figure 13: Ambient+Diffuse+Specular

6 Conclusion and Future Work

In this project we extended a given basic volume renderer to a complete and functional raycaster. Our work consisted of three main parts namely Raycasting, 2-D Transfer functions and Shading and an extension of the basic Volume Renderer. The theoretical background of these methods was explained first. Secondly, the results of the implementation were illustrated to get more insights on the various volume data samples and the workings of the different algorithms. It led us to develop and explore our artistic skills through experimentation with transfer functions and in combination with understanding of direct and indirect volume visualization algorithms and techniques. Through Phong Shade model, we saw that light can break down into three components. The first one is diffuse, which has to do with the way light falls off an object. The second one is specular which concerns the shininess of the object. The third one is ambient, which represents the minimum amount of light used to simulate global illumination. A possible future extension of the Phong shading model could be the addition of an extra coordinate to the aforementioned sum. One possible coordinate could be the emit, which has to do with the glowing effect. An alternative possible future extension could be to extent the basic triangle widget that is proposed in [4].

7 Individual reports

During the whole project we worked together as a team. Initially, we had some meetings to have an overview of the renderer, the classes provided, how they are connected and what their role is. After that, we started working together on every part of the project.Each one of us, made a small research regarding each issue that we had to solve and then we discussed about the exact solution we would prefer to implement.Every one of us was involved in all parts of the project.

References

[1] *M. Levoy. Display of surfaces from volume data. In IEEE Computer Graphics and Applications. 8(3), 29–37, 1988*

[2] *C. Schlick. A Fast Alternative to Phong's Specular Model. Academic Press Professional, Inc. San Diego, CA, USA 385-387, 1994*

[3] *J F Blinn. Models of light reflection for computer synthesized pictures.,v11, p192-198*

[4] *J. Kniss, G. L. Kindlmann and C. D. Hansen. Multidimensional transfer functions for interactive volume rendering. In IEEE Trans. Visualization and Computer Graphics. 8(3), 270–285, 2002*