VolVis Project

Volume Rendering Assignment

Ajaya Adhikari, Vasileios Milias, Vivek Subramanian
Student number: 4627199, 4614526, 4601211

1. Objective

The goal of our project is to develop/extend a volume renderer to a complete and functional raycaster. Our work consists of two main parts: ***Raycasting*** and ***2-D Transfer functions and Shading***. First, the theoretical background of these methods are explained. But the implementation is not explained in detail due to the three page restriction. The code is well commented to provide a good understanding of the implementation. In the final part, these methods are illustrated with a lot of interesting example.

2. Raycasting

For this part we implemented the following functionalities:

- *Tri-linear interpolation of samples along the viewing ray*
- *MIP*
- *Compositing ray functions*
- *Responsiveness improvement*

2.1 Tri-linear interpolation of samples along the viewing ray

Based on the skeleton code that we were provided with, we edited the function *getVoxelInterpolate()* of the class *Volume* so that it takes the coordinates as input of a point and returns the interpolated value. The tri-linear interpolation method we implemented is widely used and straightforward.

2.2 MIP

This functionality is about the implementation of the Maximum Intensity Projection method. When the *mipMode* variable is set to true the *traceRayMip* function is being called for each pixel of the image to get the color. The *traceRayMip* function samples points on the ray with a fixed sample step, and saves the maximum interpolated value. The mapping of this maximum value to a color is done in two methods.
In the first approach, which we call *the **x-ray** mode*, we do not use the transfer function. Instead, when the mip returns the maximum sample value for a ray, we map this value to an opacity. Each pixel gets whiter for the higher values and more transparent for the lower values in a black background.

In the other approach we used the transfer function (***tf** mode*) to map the maximum voxel value per pixel to a color. In this way, by changing the transfer function different regions can be highlighted and emphasized.

2.3 Compositing ray functions

For this task, we sampled the ray starting from the exit point and ending to the entry point (back to front) of a ray. The reason we chose *back to front* method instead of *front to back* is because the formulas were significantly simpler. The function *composite* 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 exitpoint and (C_n, A_n) the color and the opacity of the entry point. C'_i is the accumulated color of sample point i taking all the samples from i to 1 into account. The transfer function maps each value of the sample i to an opacity A_i and color C_i. The user defined transfer function is used to get the color and opacity of each sample value.

2.4 Responsiveness improvement

To improve the responsiveness we edited the *raycast* function of the *RaycastRenderer* class. We decided to lower the resolution of the image during *interactive* mode, such that the user can rotate the object without lag. We investigated two options: increasing the sample step and decreasing the number of computed pixels. For the second option the pixels are divided into squares consisting of $i \, x \, i$ pixels. And for each of these squares only one value gets computed. We tried to find a balance between both tricks. Through experimentation, we decided to increase the sample step by five and we decrease the number of pixels by four, to find a good balance between two. This way, the interaction with the objects become more smooth without a significant decrease of the overall resolution.

3. 2-D Transfer functions and Shading

In the second part of this assignment we integrate the gradient information into our transfer functions. The use of the gradient information reveals new and interesting features of our datasets. More specifically, the steps we followed are the following:

- Implementation of a gradient-based opacity weighting based on [1]
- Implementation of the Phong shading model

3.1 Gradient based opacity weighting

The gradient volume class was extended. Firstly, the *compute* function computes the gradient of each voxel point. Secondly, the gradient of any point in the volume can be computed through trilinear interpolation of the gradients of the surrounding voxels through the *getGradient* function.

The opacity $\alpha(x_i)$ of each sample point x_i is determined by using its gradient size $|\nabla f(x_i)|$ and value $f(x_i)$. The formula for this algorithm is presented in Levoy's paper under section "Isovalue contour surfaces" [1]:

$$\alpha(\mathbf{x_i}) = \alpha_v \begin{cases} 1 & \text{if } |\nabla f(\mathbf{x_i})| = 0 \text{ and} \\ & \quad f(\mathbf{x_i}) = f_v \\ 1 - \dfrac{1}{r} \left| \dfrac{f_v - f(\mathbf{x_i})}{|\nabla f(\mathbf{x_i})|} \right| & \text{if } |\nabla f(\mathbf{x_i})| > 0 \text{ and} \\ & \quad f(\mathbf{x_i}) - r\ |\nabla f(\mathbf{x_i})| \le f_v \le \\ & \quad f(\mathbf{x_i}) + r\ |\nabla f(\mathbf{x_i})| \\ 0 & \text{otherwise} \end{cases}$$

f_v and r respectively are the selected intensity and the radius by the user. This algorithm is implemented in the function *tranferFuntion2D*.

3.2 Phong shading model

Phong shading adds an extra dimension to the image. It allows to view the structure of the surface in detail. Phong shading has three different components namely ambient, diffuse and specular light. These components are illustrated in the last section.

- The ambient reflexion only depends on the color of the surface.
- The intensity of diffuse reflection is dependent on the angle of the light ray with the surface, reflectivity and color of the surface.
- The intensity of specular reflection depends on the same factors as for diffuse reflection plus the shininess of the material and the viewer's direction in addition.

This algorithm is implemented in the function *getShade*.

4. Results & Evaluation

4.1 MIP

In *figure 1*, we show the result of the **mip *x-ray*** mode on the carp dataset. The skeleton of the carp, whose sample values are high, is whiter, while the skin, whose sample values are not so high is more transparent. This technique seems very useful for the study of the skeleton of the fish.

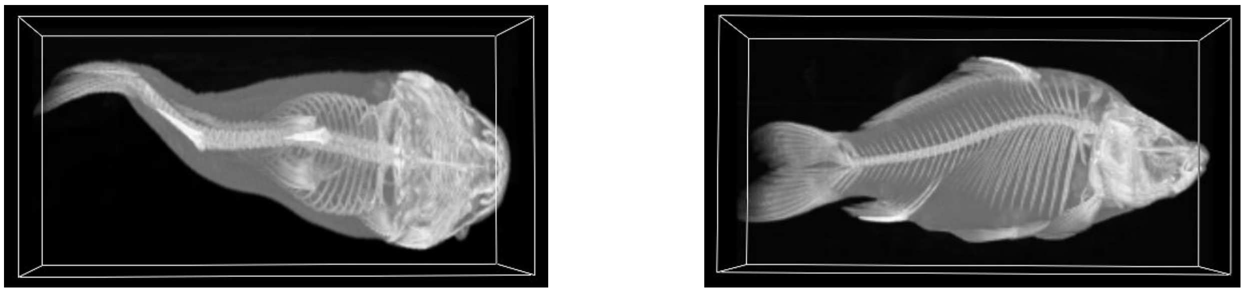


Figure 1: *MIP / carp dataset (x-ray mode)*

However, the ***x-ray*** mode of the **mip** method is not always useful as it can be seen from *figure 2*. When the dataset does not have an inner structure with high intensity values this method does not provide insightful information.

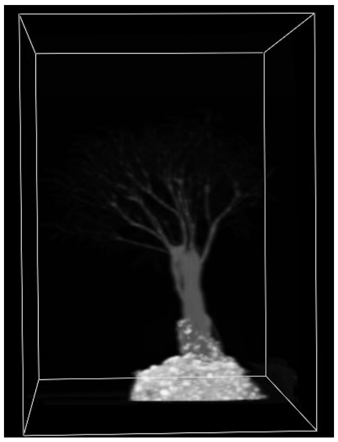
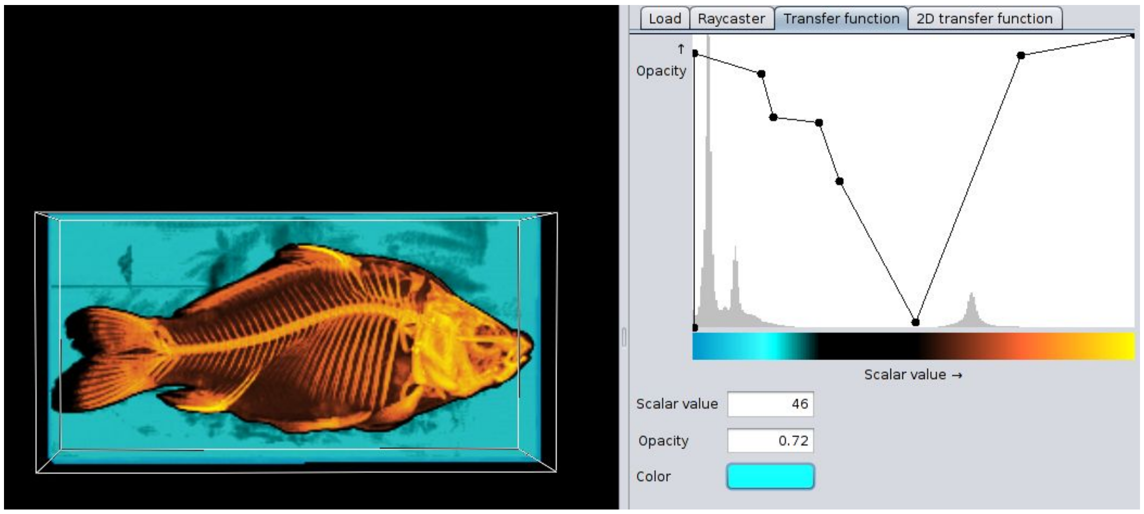


Figure 2: *MIP / bonsai dataset (x-ray mode)*

The ***tf** mode* of the **mip** method is also interesting. By editing the transfer function we have the option of highlighting the different intensity surfaces. In *figure 3*, we present the results of this method on the carp dataset. The information we can access is the same, but now emphasizing and focusing on any part maximum intensity part is possible.



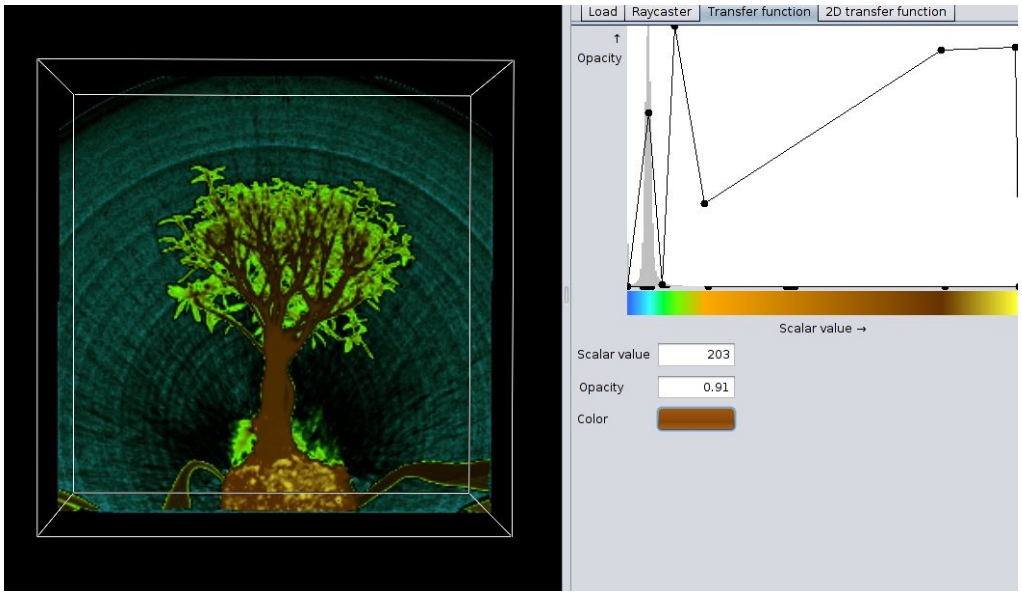
Scalar value 46

Opacity 0.72

Color

__Figure 3__: MIP / carp dataset (tf- mode) / minimum density mapped to light blue (environment), the skin is mapped to black and the skeleton is mapped to white

Furthermore, we investigated the proper way to manipulate the transfer function so that we can get more information on the bonsai dataset. It looks like the leaves of the bonsai have a very low intensity. Thus, in the ***x-ray*** mode the leaves end up having the same color with the background. However, in the case of the *tf* mode we can select a color for them and therefore see them. In *figure 4* we present the abovementioned.



Scalar value 203

Opacity 0.91

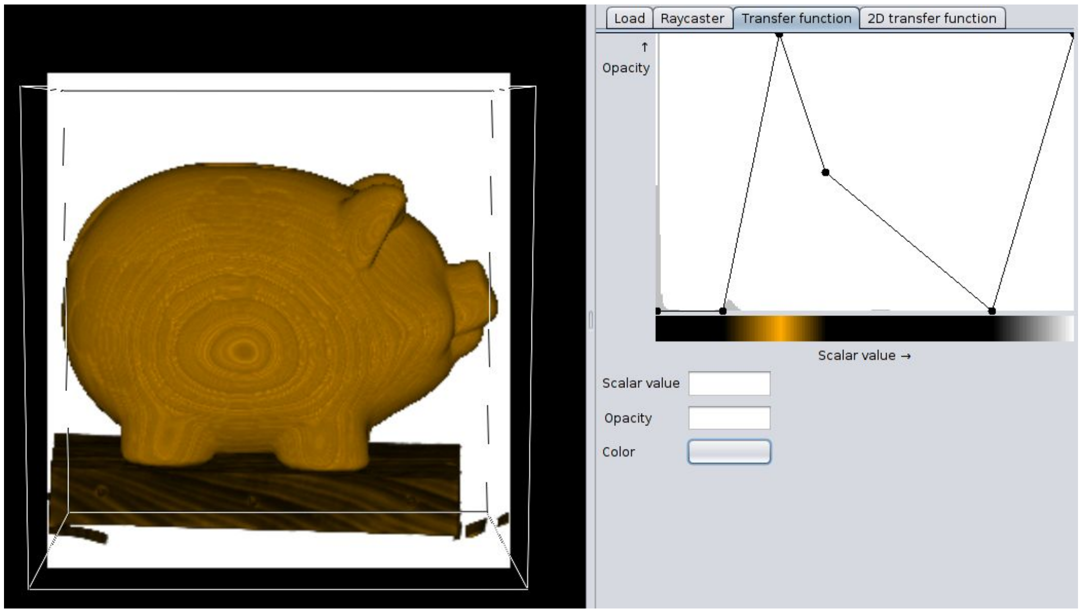
Color

__Figure 4__: MIP of the orange dataset from different angles

In general, the **mip** functionality is relatively fast and interactive. It is easy to experiment with this method and analyze the objects on different perspectives.

4.2 Compositing

In *figure 5* and *6* we present the results of the **compositing** functionality on the pig dataset. For each figure we show the transfer function that we used. In *figure 5*, only the outside layer of the piggy bank can be seen, while in *figure 6* we can also distinguish some patterns on the surface of the object as well as the coins inside it.

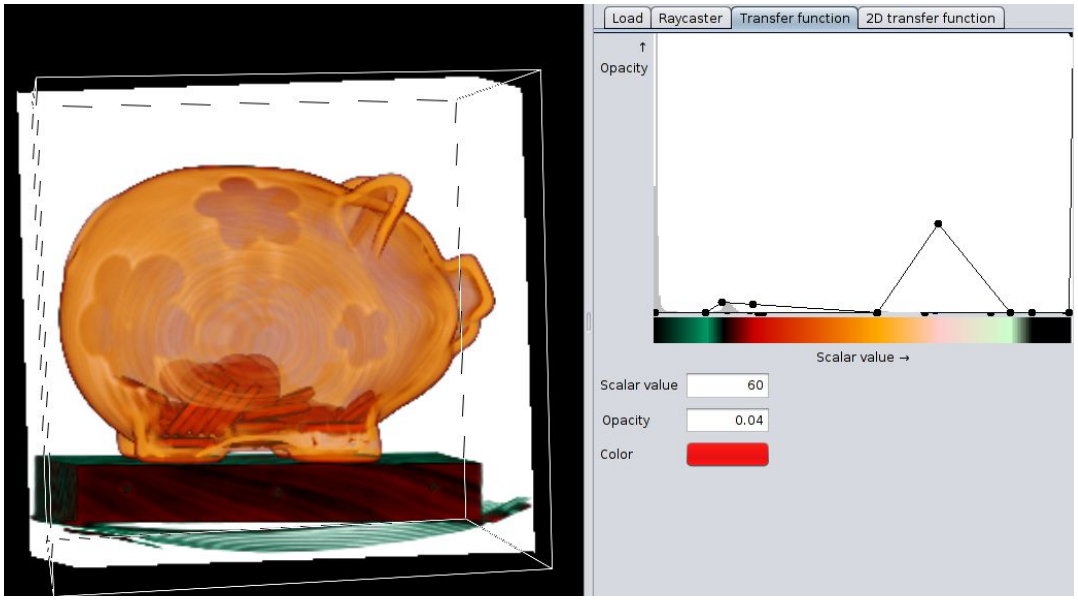


Scalar value

Opacity

Color

Figure 5: Compositing / Pig / Default transfer function



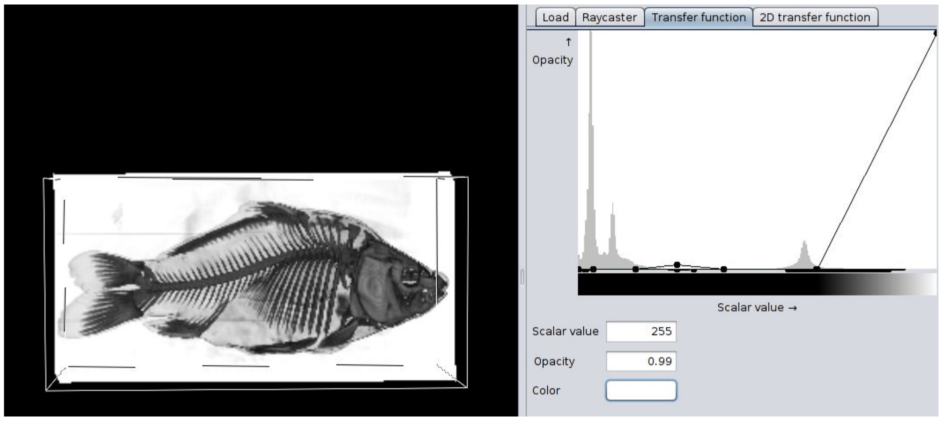
Scalar value 60

Opacity 0.04

Color

Figure 6: Compositing / Pig / Edited transfer function

It is easily observed that this method provides more flexibility than the **mip**. For example, in *figure 6* we show the skeleton of the carp by using the **compositing** method. Apart from that, we have the option to see other levels of the object, while with **mip** we do not. In addition, the interpretation of the sense of depth is significantly better in **compositing** than in **mip.** Finally, while **mip** method is computationally much more efficient, **compositing** method is still usable and responsive.

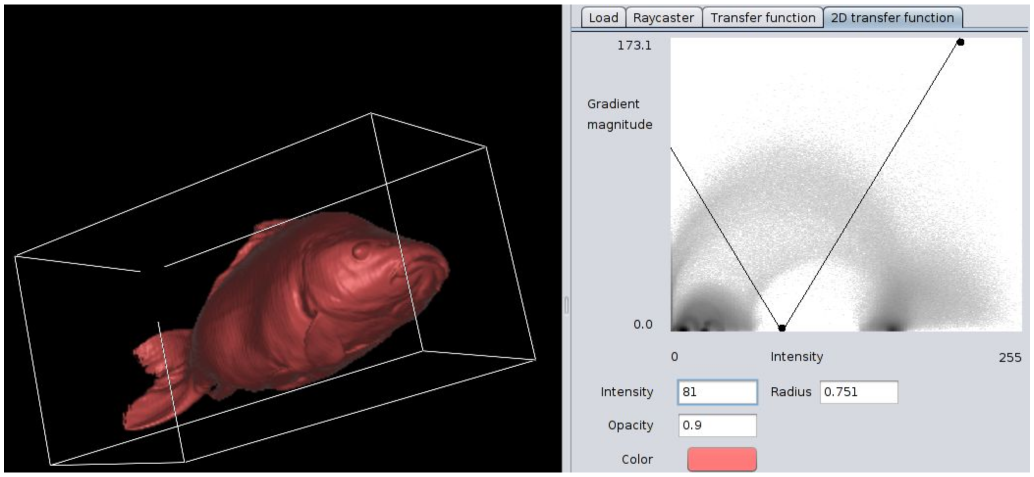


Scalar value: 255
Opacity: 0.99
Color

Figure 6: Compositing / Carp / Edited transfer function

4.1 2D - Transfer Function combined with shading

Moreover, we explored the insights we could gain by using the 2D-transfer function in combination with the volume shading. In *figure 7*, we present the result of this technique on the carp dataset, which is probably the most intriguing dataset to be analyzed with the specific method. To verify the correctness of this method, in *figure 8* we present a real image of a carp.



Intensity 81
Radius 0.751
Opacity 0.9

Figure 7: 2D - Transfer Function / Carp / Edited transfer function

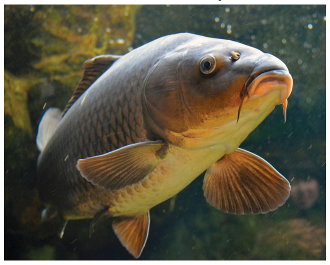
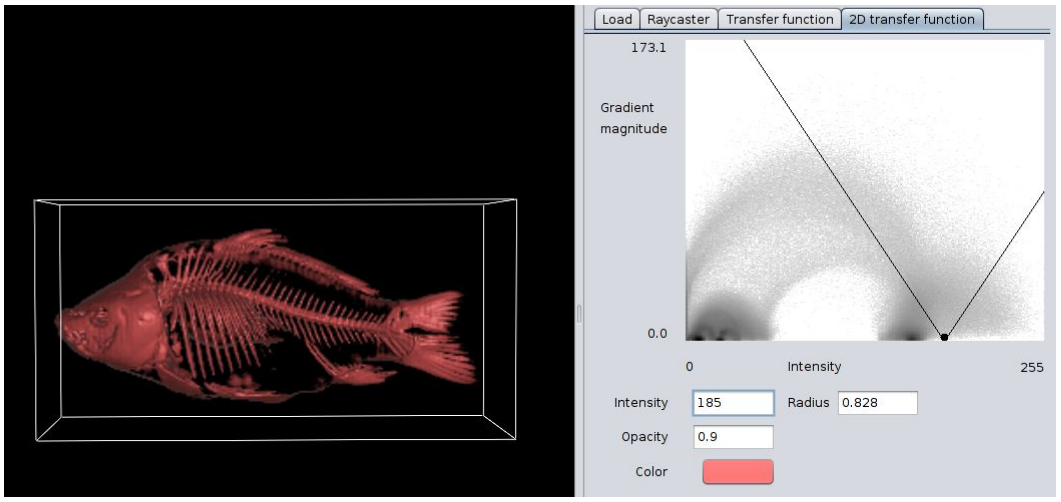


Figure 8: Carp's photograph [3]

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, as can be seen in *figure 9*. In addition, to clearly understand the power of shading its each component is illustrated in *figure 10*. On the downside, this method is computationally expensive and the responsiveness when the image is being rotated is not very high.



Intensity 185 Radius 0.828

Opacity 0.9

***Figure 9*:** 2D - Transfer Function / Carp dataset / edited Transfer Function

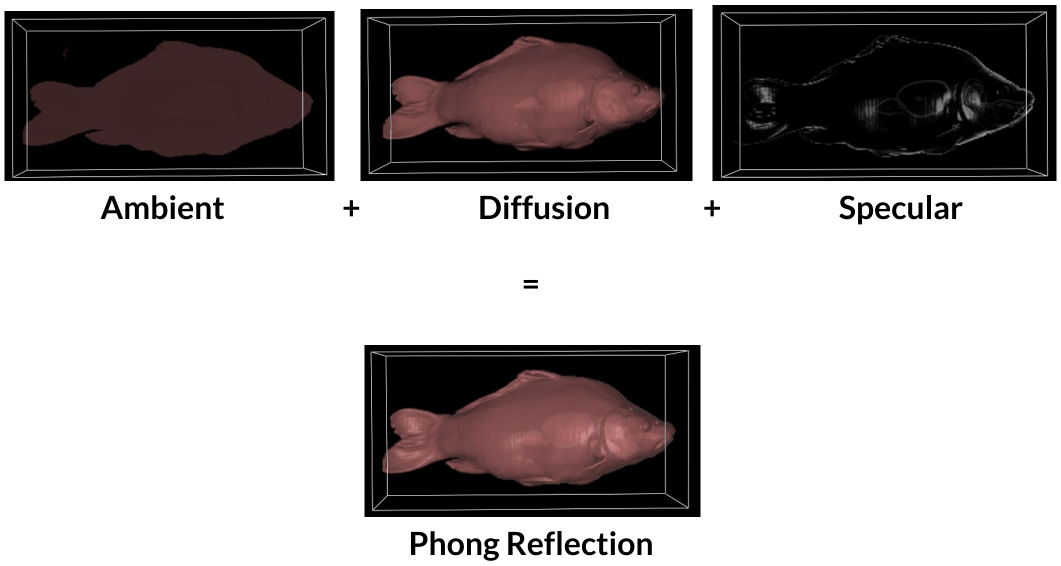


***Figure 10*:** 2D - Transfer Function / Carp dataset / Ambient-Diffusion-Specular

Finally, we wanted to investigate if we the 2D - transfer function method let us understand more about the flower patterns we found on the pig with the compositing method. From the compositing method it was not clear whether the patterns were stickers, concaves or convexes. On *figure 11* it is clear that the flower patterns have concave structures.

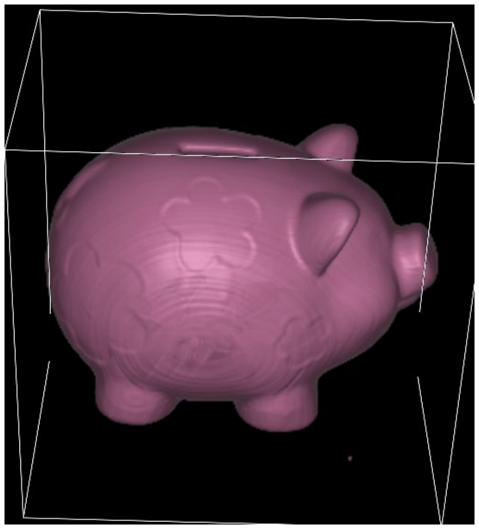


Figure 11: 2D - Transfer Function / Pig dataset

5. Conclusion

In this project we extended a given basic volume renderer to a complete and functional raycaster. Our work consisted of two main parts namely *Raycasting* and *2-D Transfer functions and Shading*. 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 datas and the workings of the different algorithms. The pros and cons of each method has also been discussed. Overall this was a very fun project, due to the beautiful images we could produce. It lead us to develop and explore our artistic skills by experimenting with the transfer functions, together with the understanding of volume visualisation algorithms.

6. References

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

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

[3] Image of a carp
https://s-media-cache-ak0.pinimg.com/originals/97/fb/5a/97fb5a5a1e3dfcc59ee921abb0a5bfc0.jpg