Volume Rendering Assignment

(Anna Vilanova based on TU/e course by Michel Westenberg)

In this assignment, you will develop a volume renderer based on the raycasting approach as described during the lectures. The skeleton code is in Java and provides a set-up of the whole graphics pipeline, a reader for volumetric data, and a slice-based renderer. The main task is to extend this to a full-fledged raycaster based on the provided code skeleton.

1 Getting the skeleton code running

We provide skeleton code to help you get started on the assignments. The code is in Java, and provided as Netbeans projects. It should be possible to run the project out-of-the-box on Mac OS X, Linux, and Windows systems. For OpenGL graphics, the code relies on the JOGL libraries, which are included in the archive. Netbeans also provides means to debug the code, breakpoints and watching variables, have a look at this possibilities.

You will need algorithmic and programming skills for this project, however, you do not need deep knowledge of Java given that we have provided the basic skeleton. If needed you can have a look at http://www.javatpoint.com/simple-program-of-java.

You need a basic knowledge of object oriented programing (what is a class, and what are the functions in a class). The syntax should be similar to most other programing languages you might have been using before, but you might need some time to get used to it.

2 Skeleton code

The skeleton has the familiar set-up of a main application (VolVisApplication), which holds the visualization and user interface. The helper classes Volume and VolumeIO represent volumetric data and a data reader (AVS field files having extension .fld), respectively.

The class GradientVolume is partially implemented, and provides methods to compute (not implemented!) and retrieve gradient vectors.

To make working with 1-D and 2-D transfer functions easier, the classes TransferFunction, TransferFunctionEditor, and TransferFunction2DEditor provide methods to store and graphically edit transfer functions. The class VectorMath offers a number of static methods for vector computations. Finally, TrackballInteractor provides a virtual trackball to perform 3-D rotations.

The class RaycastRenderer provides a simple renderer that visualizes a view-plane aligned slice through the center of the volume data. The renderer also draws a bounding box to give visual feedback of the orientation of the data set. This class is where most of the raycasting algorithm will take place.

3 Assignments

3.1 Part I: Ray casting

To start with study the method slicer() in the class RaycastRenderer, we will use this class as a basis to develop a raycaster.

Study the function raycast to see how the different types of visualizations can be activated and the initial direction and position of the ray are calculated.

The coordinate system of the view matrix is explained in the code. Furthermore, the code already provides a transfer function editor so you can more easily specify colors and opacities to be used by the compositing ray function.

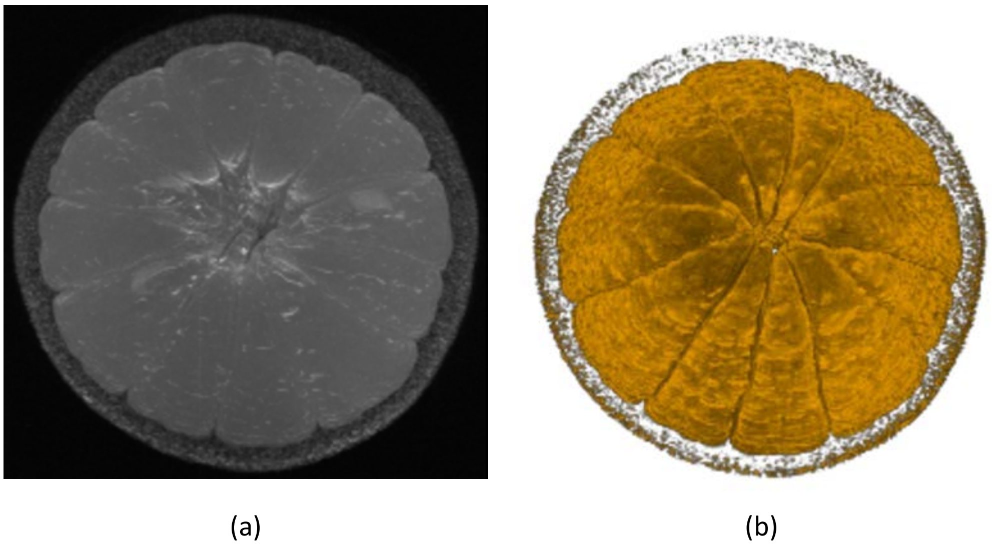


Figure 2.1: *(a) MIP of the orange dataset. (b) Result of the compositing ray function on the orange dataset with the default settings in the transfer function editor.*

Look at the function slicer() and activate the Transfer Function (TF) in order to understand how it is used. Figure 2.1 shows reference result images for the orange dataset based on the MIP ray function (Fig. 2.1(a)) and the compositing ray function using the default settings of the transfer function editor (Fig. 2.1(b)). We also provide some data sets that you can use. We recommend to use *Carp8* for testing given its small size.

Implement the following functionalities:

1. Tri-linear interpolation of samples along the viewing ray (function getVoxelInterpolate() in the class Volume). This function is also used in slicer() you can see the effects already there.
2. MIP (function traceRayMIP in the RaycastRenderer)
3. compositing ray functions (modify raycast and create the necessary functions within RaycastRenderer)

4. As you may notice, a software raycaster is quite slow. Make the application more responsive during user interaction. A straightforward way to do this is to compute the image at a lower resolution. You can think about other strategies. (hint you can use the interactiveMode attribute of the renderer)

Experiment with various data sets, try to highlight interesting features in these datasets, and compare pros and cons of MIP and compositing.

3.2 Part II: 2-D transfer functions and Shading

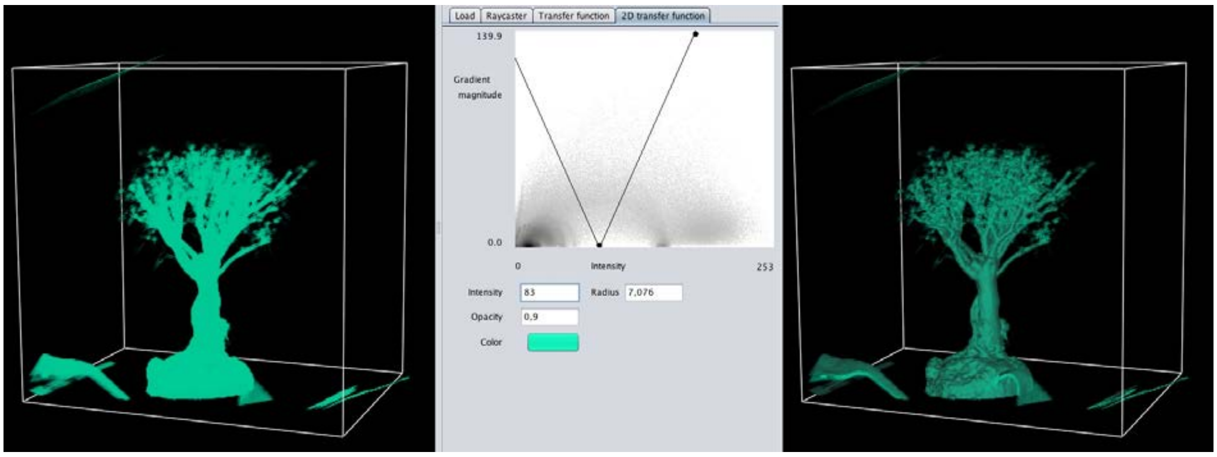
As discussed in the lectures, simple intensity-based transfer functions do not allow to highlight all features in datasets. By incorporating gradient information in the transfer functions, you gain additional possibilities.

Below are the requirements:

1. Implement gradient-based opacity weighting in your raycaster, as described in Levoy's paper [2] in the section entitled "Isovalue contour surfaces", eq. (3). The code already contains a 2-D transfer function editor which shows an intensity vs. gradient magnitude plot, and provides a simple triangle widget to specify the parameters. The class GradientVolume should be extended so that it actually computes the gradients.

2. Implement an illumination model, e.g., Phong shading as discussed in the lectures. An example result with and without illumination is shown in Fig. 2.2. The Phong shading parameters in this example are $k_{\text{ambient}} = 0.1$, $k_{\text{diff}} = 0.7$, $k_{\text{spec}} = 0.2$, and $\alpha = 10$.

3. Extend the basic triangle widget by the approach described by Kniss [1], which introduces extra parameters to control the range of gradient magnitudes that are taken into account in the computation of the transfer function.



Intensity 83
Radius 7,076
Opacity 0.9
Color 🟩

Figure 2.2: *Gradient-based opacity weighting result on the bonsai data. The image on the left is without illumination.*

Experiment with various data sets and compare the results with those obtained by the approach in the previous exercises. Feel free to implement any extensions to the framework that you would find interesting.

We provide some data sets that you can use, but feel free to search for more (notice that we read a specific format so this would mean you need to build a converter or reader).

- https://www.cg.tuwien.ac.at/xmas/#Download%20data Xmas tree.
- http://medvis.org/datasets medical data sets.
- At www.vtk.org you can find example data.
- http://www.osirix-viewer.com/datasets/ medical data sets.
- http://www.sci.utah.edu/download/IV3DData.html

4 Deliverables

- **Source code** with comments. We should be able to understand your code to be able to evaluate it, so make sure that the code is well structured, variable names have meaning, and that you have comments indicating what you are doing for each function.

- **Report** indicating:
    - What has exactly been implemented. Describe in a theoretical level and in your own words, what you implemented and where this implementation can be found in the source code.
    - The report should include a results and evaluation section where you present and analyze different volumetric data sets of your choice with the tool you have developed. Show features that would be interesting to visualize from the data, show images and explain how you achieved the visualization, e.g., indicate the transfer function. Comment on the usability of your tool (i.e., times, interaction, effectiveness, etc.)
    - **The Report** should be a maximum of **3 pages A4 size** of text. The pages should have similar font size and margins as in this project description document. The text has to be to the point. So 2 well written clear pages are better than 3 pages with the extra redundant content. Images will not be counted as part of the 3 pages so include as many as you consider adequate. Illustrating what you can achieve is important
- **Individual reports:** there should be a max. 300 words document for each of the group members describing what you did individually in the project. It is important that we know that your partner also agrees with it therefore submit it together with the rest of the project. If you work in all aspects of the project together, which is desirable, just indicate that. You do not need to extend yourself unnecessarily.

Bibliography

[1] 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.

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