Volume rendering

In this assignment, you will develop a volume renderer based on a raycasting approach. The skeleton code 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.

Getting the skeleton code running

We provide skeleton code to help you get started on the assignments. The code is in Python. After setting up the environment and installing some additional libraries (see separate document), the project should run out-of-the-box on Mac OS X, Linux, and Windows systems.

Skeleton code

The skeleton provides data readers, a graphical user interface, and several helper functions related to interaction and mapping images to the screen. You should only have to look at the file `implementation.py`; there is no need to understand nor change any of the other parts of the framework.

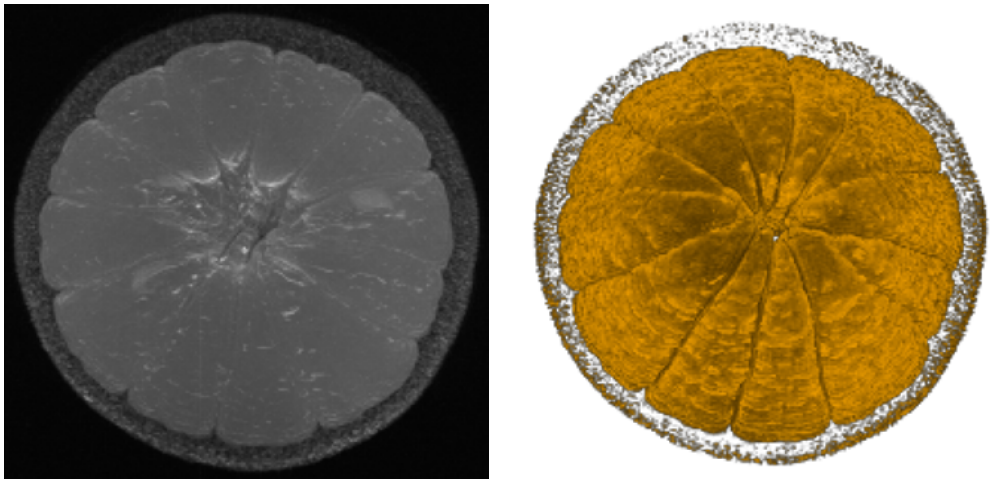
The class `RaycastRendererImplementation` provides a simple renderer (via the function `render_slicer` that visualizes a view-plane aligned slice through the center of the volume data. The renderer also draws a bounding box to give visual feedback of the orientation of the data set.

Assignments

Ray casting implementation

Study the method `render_slicer` in the class `RaycastRendererImplementation`, and use this as a basis to develop a raycaster that supports both *Maximum Intensity Projection* and *compositing* ray functions. We have already provided placeholder functions `render_mip` and `render_compositing` in the skeleton, and these functions will be called automatically upon pressing the corresponding buttons in the user interface.

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 1 shows reference result images for the orange dataset based on the MIP ray function (Fig. 1(a)) and the compositing ray function using the default settings of the transfer function editor (Fig. 1(b)).



(a) (b)

Figure 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.*

- Implement the following functionalities:
  1. Tri-linear interpolation of samples along the viewing ray.
  2. MIP and compositing ray functions.
- Demonstrate that your MIP and compositing ray functions work as intended using the orange dataset.

Apply your volume renderer to a real use case

The Scientific Visualization community runs a yearly visualization challenge, see `http://sciviscontest.ieeevis.org` for an overview.

This year, you will work with the 2013 contest data concerning the area of developmental neuroscience. The website of the contest can be found at `http://sciviscontest.ieeevis.org/2013/VisContest/index.html`. As you can read there, the dataset is very large and there are many research questions and visualization challenges to overcome. Therefore, we have provided a subset of the data, which contains a number of relevant or interesting sets of genes.

Your assignment is to produce an insightful visualization of (some of) this data. The document that describes the dataset provides quite some inspiration. The challenges are to:

- Visualize multiple volume data sets of the energy simultaneously.
- Perform volume rendering on the annotation data, which is a labeled volume data set. Each voxel has a label that is the ID of a particular neural structure.
- Combine the rendering of the energy volume data and the annotation volume data.

Deliverables

Write a report of maximum **5 pages** when typeset on A4 using an 11-point font size and normal margins (about 1 inch on all sides). The report should demonstrate that your raycaster

works properly using the orange dataset (i.e. show the data from multiple angles and vary the transfer function etc.). The remainder of the report should be about interesting insights that you obtained on the contest data set, and how you obtained them (make sure your results are reproducible by including enough details about parameters and transfer function settings).

The report and code should be handed in by **December 8, 2019**. Upload the report (in PDF) and code (in ZIP or other archive) as separate files into Canvas. If the report is not uploaded separately into Canvas, we cannot grade it, because it will be invisible for the built-in grading tool.