LAB 1: VTK

ANDRÉS MARTÍNEZ MORA

CM2004: VISUALIZATION IN MEDICAL IMAGING

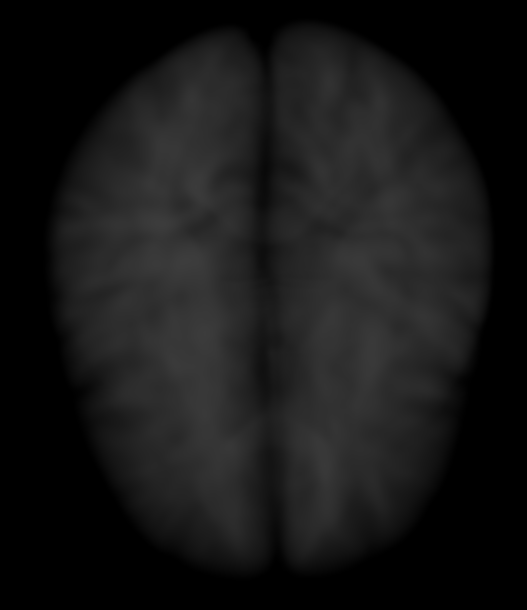
EXERCISE 1

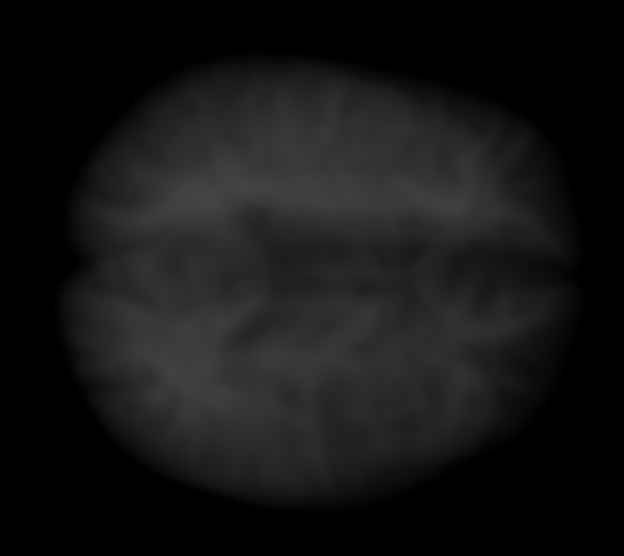
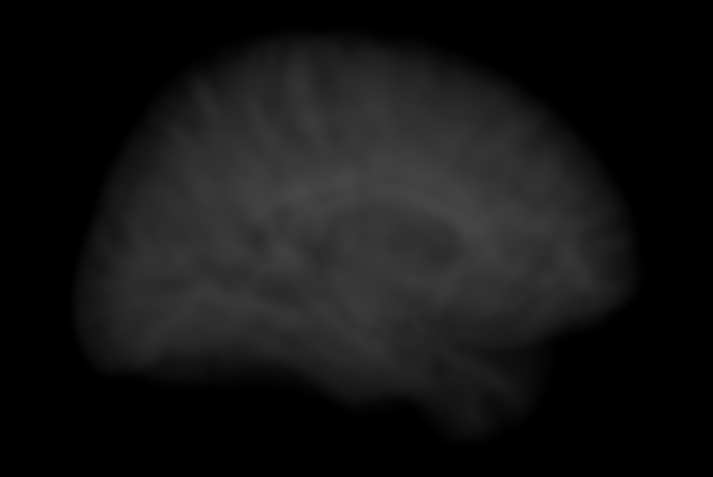
In this exercise we are asked to build a Volume Rendering tool of a MR brain volume. The provided code has to be run from the terminal with: “*ipython main\_volume.py filename*”, so the filename should be read as the first argument with *sys.argv[1]*.

The source is built with a .nii image reader and it is outputted to a mapper. The mapper is a Ray Cast Mapper. The volume in VTK must have some properties, so as we are working with Volume Rendering, it must have at least a color range and an opacity range, thus a color range is provided with some Look-up-table (LUT) and a piecewise function is provided as opacity function. The mapper and the properties are then connected to the actor.

To visualize the renderized results, a renderer is created. This renderer is connected to the actor and has some views defined by a camera that is also connected to the renderer. The renderer performs its actions in a render window that is also defined and controlled by an also created interactor.

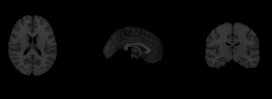
These are the final results after following all the instructions. All images are screenshots from different positions:





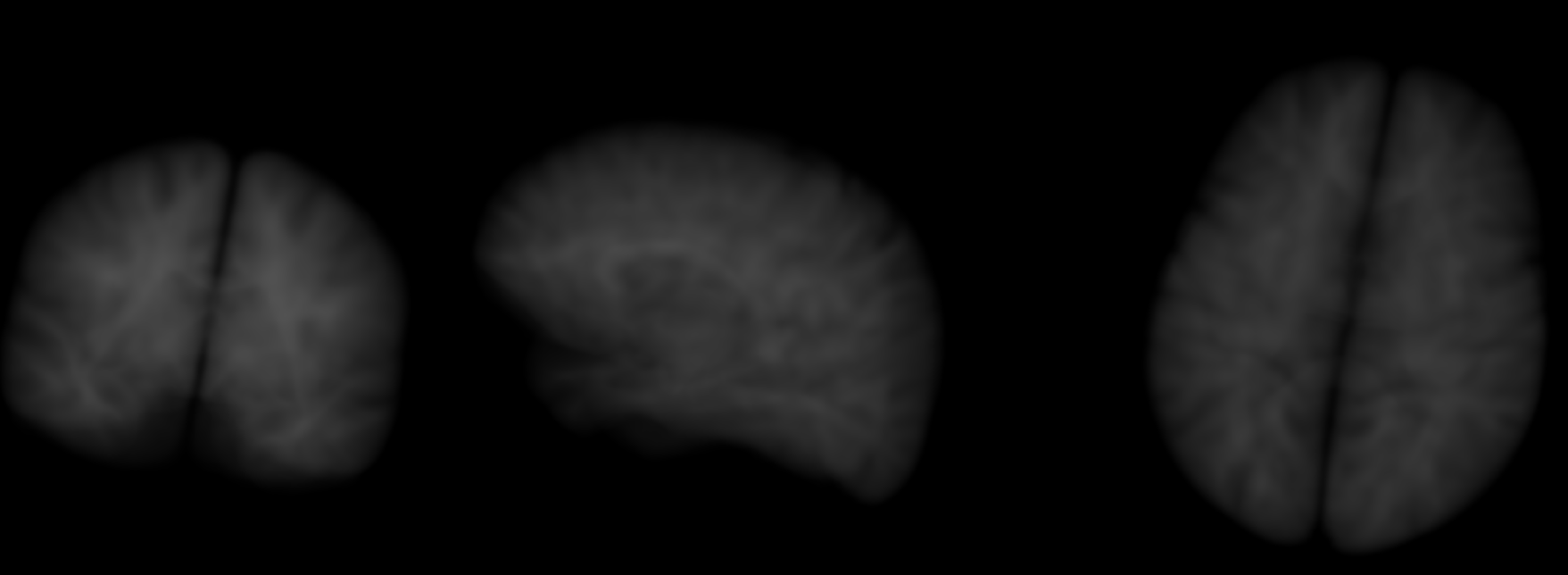
EXERCISE 2

In this exercise we try to generalize the proposed pipeline in Exercise 1. First of all, we run a code provided that shows the axial, coronal and sagittal views of MR slices from the same volume as in Exercise 1. In here, we work with different combinations of grid displays for several results at the same time. This is one of the results:

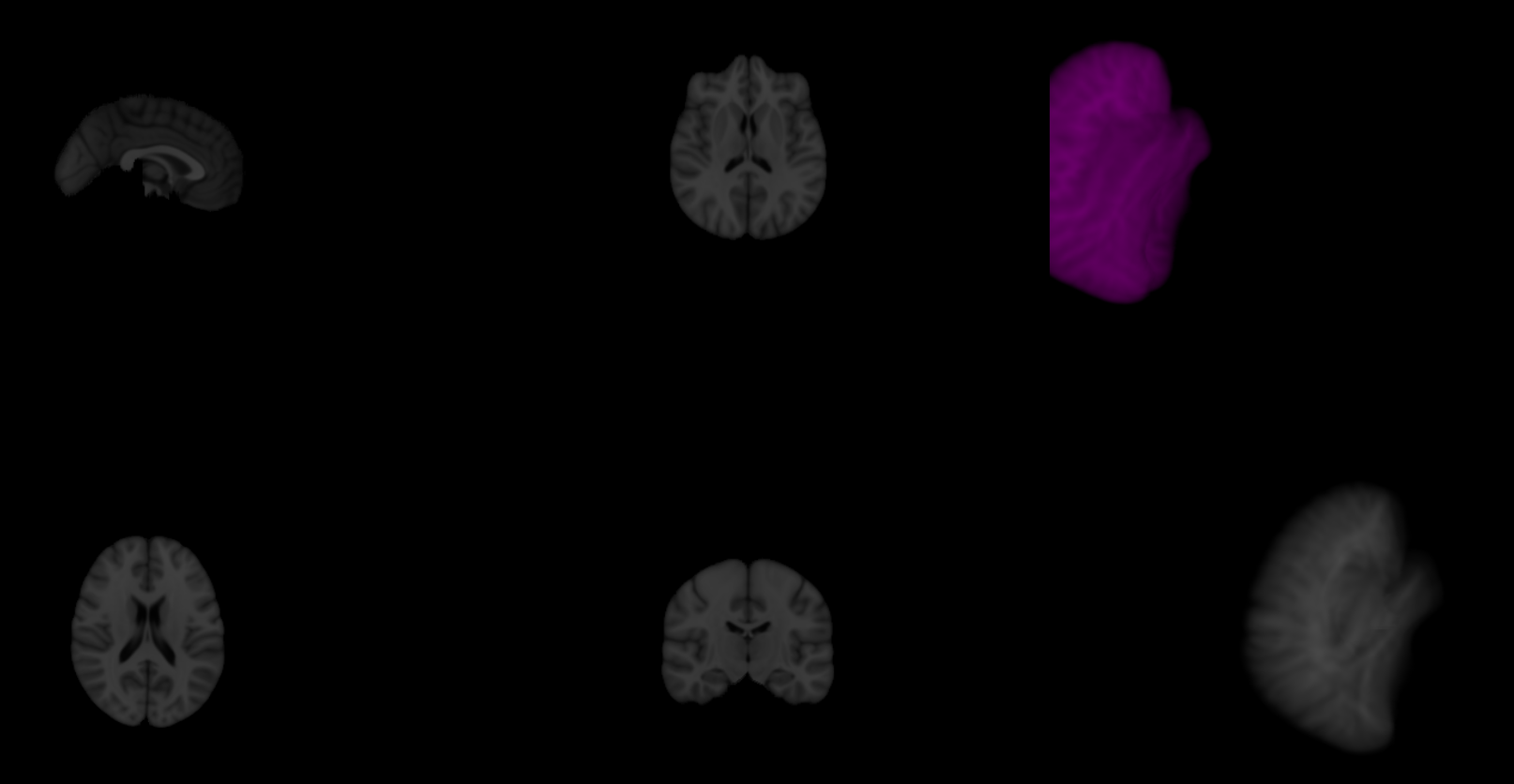


This code is known as Multi-Renderer, receiving the image source as input, then internally in a render class providing the mapper, the actor, the volume properties and the renderer. The output is the information display, being manipulated in another code called *GridWindow.py* with the class *CombinedWindow* so to display results in several windows. Slice interaction is made through mouse clicks so to rotate the slices and left and right buttons to change the slice.

The next step was to generalize the code from Exercise 1 into a render class, as in the case of the slice rendering above. This class gets as input only the source where the image is loaded. Then it connects the source to a mapper and to the volume properties (color and opacity), providing also a mapper that works with them. The proposed class also sets the properties of the camera. The Multi-Renderer code then displays the results with a render window and a given interaction setup. Interaction is made just mouse movements and clicks on the volume. Multi-rendering results with the generalization of the code from Exercise 1 is shown below with 3 examples:



A third render class was also coded in the same way as in Exercise 1, with some modifications in the Look Up Tables, the opacity values and the camera parameters. The multi-render display was used to show images for slice visualization (axial, coronal, sagittal and oblique), the original volume rendering that has been generalized and a modified version of the previous rendering. The results from this visualization are displayed below:



It can be observed that the modified volume-rendered visualization has been completed in such a way that it has “purple”-ish color and it is far away from being centered.