MEGA – UE1

Preliminary Instructions

Please check the document `PythonInstructions.txt` for information on how to start your Python environment for the exercise.

For **parts 1 and 2** of this exercise you will use `Python 3.9` and the following libraries (with their respective versions, if needed): `tkinter`, `pydicom (2.4.3)`, `matplotlib`, and `scipy (1.11.3)`.

For **part 3**, you will need to download the framework MeVisLab (https://www.mevislab.de/download).

Part 1: Noise Suppression

The script `DICOMViewer.py` is a simple application for viewing DICOM images. It allows you to load a folder containing DICOM files, display the images, and navigate through the slices interactively using a slider. Use the application to load any of the datasets that we are providing you in the folder `Datasets_UE1`. Inspect the datasets and pick one that you would like to work with further.

Check the `scipy.ndimage` library out (https://docs.scipy.org/doc/scipy/reference/ndimage.html). It contains different functions for multidimensional image processing. Using this library, we can apply different filters for noise suppression. Adapt the code given in `DICOMViewer.py` to implement three filters: median, average, and Gaussian. Apply each filter with at least three different parameterizations (e.g., kernel sizes for median/average, sigma for Gaussian). **Deliver:**

- Your implementation.
- Screenshots of three sets of filtered images (one set per filter; for each set employ different parameterizations).
- Explain shortly: *Which structures in the specific dataset you are using become harder/easier to see under each filter? Relate your observations to the parameter choices.*
- Reflect: *What might be the clinical risk of choosing a filter/parameterization that "looks good" visually but over-smooths in the context of this dataset?*

Part 2: Resampling

Take again the original script `DICOMViewer.py` (in its state before the changes you introduced in Part 1). Check out in the `scipy.ndimage` library the filter called `zoom`. This filter is used to perform zooming or resampling of an image or multi-dimensional array. It can be used to both upsample or downsample an image or array by resampling the data at different intervals.

a) Perform a downsampling of the images by a factor of 4 and subsequently, an upsampling of the downsampled images also by a factor of 4. *Is the final image the same as the original and why?* **Deliver** your code implementation and briefly explain why, using a few screenshots to justify your answer.

b) Use different interpolation method within the `zoom` filter and explain their effect. The `order` parameter determines the interpolation order, which affects the quality of the resized image. The available options for the order parameter are 0 for a nearest-neighbor interpolation, 1 for a bilinear interpolation, 3 for a cubic interpolation. **Deliver** your code implementation and briefly explain the effect

of the chosen interpolation method, backing it up with screenshots that showcase specific examples of the effect of the interpolation methods on the selected dataset.

Part 3: Image Segmentation

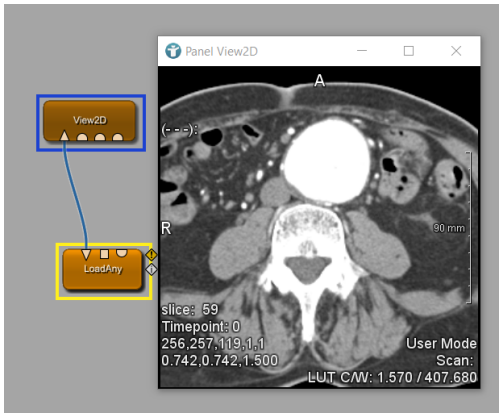
In this exercise, you will learn how to use the rapid prototyping framework MeVisLab to load and segment a structure from a medical image using a *thresholding* technique. Then, you will also learn how to perform a *region growing* for segmentation purposes. MeVisLab is a powerful tool for medical image processing, visualization, and analysis. This exercise will help you become familiar with loading medical images, applying basic image processing, and segmenting structures of interest. Deliver answers to the questions highlighted in pink below.

a) Thresholding

- For this exercise, we will use the **aneurysm.vti** data set (from the `Datasets_UE1_part3` folder).

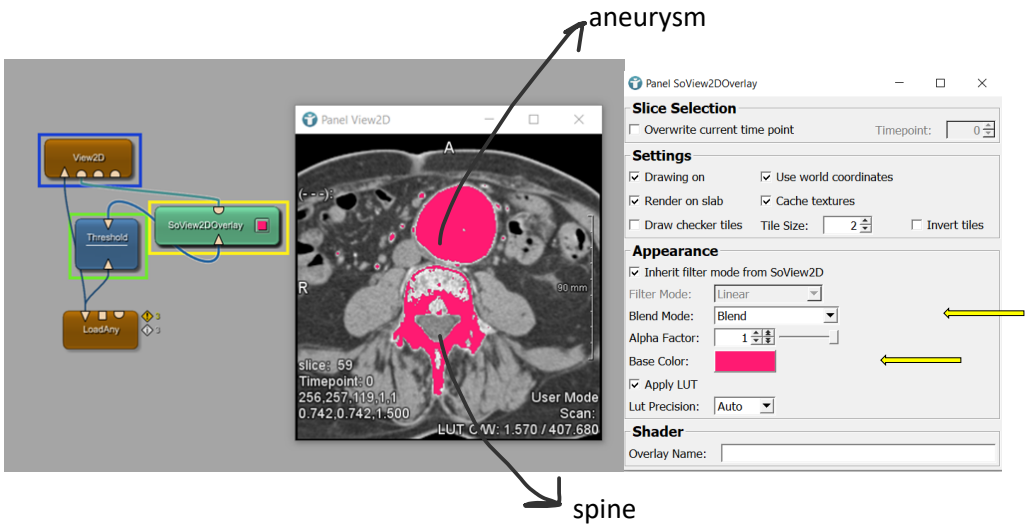
- Launch MeVisLab and look at the top of the MeVisLab window – there should be a search bar. Type `LoadAny` in. You should now have on your canvas an orange box (see screenshot below). If you double-click it, a window pops up where you can import a medical image into your MeVisLab project. Press `Load`.

- In the same way as before, bring to your canvas a `View2D` module. Link the two modules by connecting their input/output, as denoted with a triangle (see screenshot below). If you double-click on the `View2D`, you can see your dataset and scroll through it.



- Now, load and link a `Threshold` module between the loader and the viewer. Double-click on the `Threshold` module to check out its properties. Change the value of the field `Threshold` and see what it does.

- Now, load a `SoView2DOverlay` module and reproduce the network in the figure below. Make sure that the blend mode in the `SoView2DOverlay` is set to `Blend` and that you have chosen a color different than white/black/grey for the overlay. *Please mind that you will need to find what is the appropriate value of the threshold to be able to distinguish the aneurysm (the blob in magenta in the screenshot below) and the spine from the rest of the image.* What is a good value of the threshold? Deliver also your network.



- Now, add a `View3D` module at the output of the `Threshold` module and double-click it. This is a 3D rendering of the segmented area. We will learn more about it soon ☺

b) Region Growing

- Load a clean canvas. Use the same `LoadAny` module and load the same data set as before. Link to its output a `RegionGrowingMacro` module (**NOT** the simple `RegionGrowing`). Double-click the module and inspect what the different settings do. Try to reach a similar result as before. *Please ensure that you activate the following two options: Main Panel → General Parameters → Auto-Update and Advanced 1 Panel → Markers → Persistent markers. These settings ensure that you are preserving the selected seed points (markers) when saving the network.* Can you actually separate the aneurysm from the spine? Yes/no and why? Deliver also your code for this network. Provide screenshots to back up your answers.