DICOM Visualizer in Python Language

Davide Console
NECST Lab
Politecnico di Milano
Email: davide.console@mail.polimi.it

I. Introduction

Due to the centrality of diagnostic imaging in the healthcare sector, this project stems from the desire to give the user an easy and rapid way to access CT and MR images to provide useful information in the clinical process. In order to achieve this goal, I created a Graphic User Interface (GUI) in Python Language that is able to read every diagnostic image in DICOM format, since DICOM images are between the most used data format in the medical field.
In this report we will discuss how the code has been created and how it operates, from the selection of the images to the various functionalities that have been implemented.
The images are shown in three different and independent views. The user can scroll the tomographies of each view using a slider. They can also use the zoom functionality to enlarge an image up to 4 times its original size or they can use the manual segmentation to point out organs or other parts of interest in the human body.

II. Main code

A. Tkinter window creation

Because a GUI works differently from an usual code, which runs once from top to bottom, it is important to set an event loop as soon as the code is executed in order to handle inputs from the keyboard or the mouse.
To do so, firstly we initialize the tkinter interpreter so that it can execute tkinter's methods without compiling first. *Tkinter* is a python library which enable to create a GUI with all its widgets.
Then we define `root` as the main window, specifing also its title as well as its geometry.
Finally, to set the event loop, it is necessary to insert at the end of the code the method `root.mainloop()`. This method will loop forever, waiting for events from the user, until the user exits the program by closing the main window.

B. Directory selection

When the user runs the code, the first thing they will see is a dialog window [Fig. 1] where they are asked to choose a directory. This directory must contain DICOM files (`.dcm`). When the directory is selected, every file inside it is scanned and the code picks out only `.dcm` files.
If no file is found or the user selects the `Cancel` button, a new box [Fig. 2] will appear to inform the user, who can then choose to retry the proceeding by pressing the button `Yes` or to exit the program by pressing `No`.

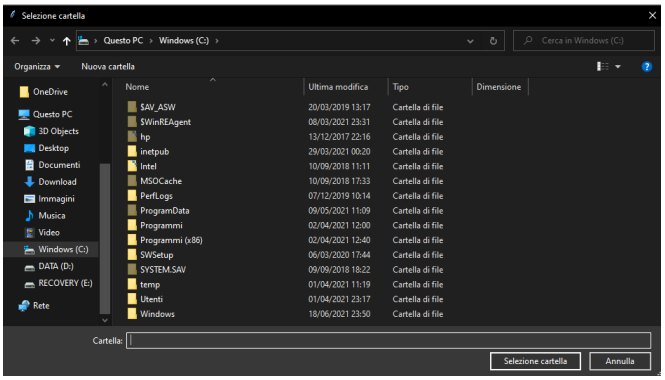

Fig. 1. Dialog window

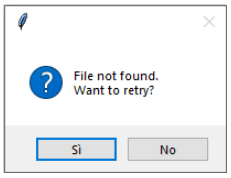

Fig. 2. Message box

C. Dataset elaboration

Once a correct directory has been chosen, the code can finally analyze the dataset. All the `.dcm` files in a folder contain a single CT or MR tomography and they are supposed to be already neatly enumerated from first to last one.
To analyze the dataset, the following steps are executed:

1) all the files' paths are saved in a python list which will have the same length as the number of `.dcm` files in the directory;
2) a 3D array is initialized where all the 2D DICOM files will be storaged moving on the z-axis of the 3D array;
3) using a *for cycle*, the list is scanned; for each path the relative image is extracted, thanks to the methods in the *Pydicom library*, and put into a slice of the 3D array
4) The images are saved as numpy.array and are shown in 8-bit greyscale. Because there are some issues caused

by a scale error of the pixels [Fig 3] which causes the images to be oversaturated, using the *cv2 library* the range of pixels is normalized in order to have values between 0 and 255 [Fig. 4];

5) lastly, it is initialized a mask, which will be useful for the segmentation [III-C].

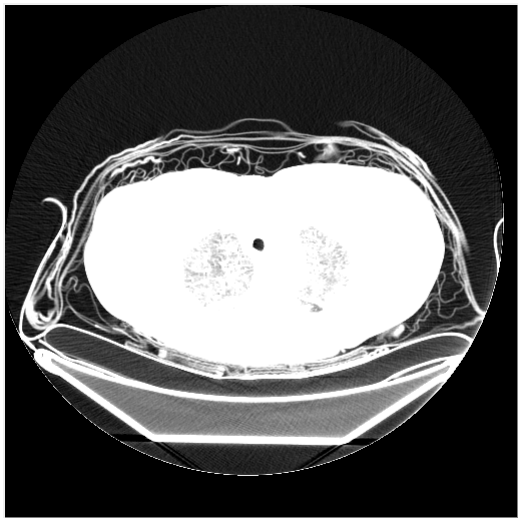

Fig. 3. DICOM image

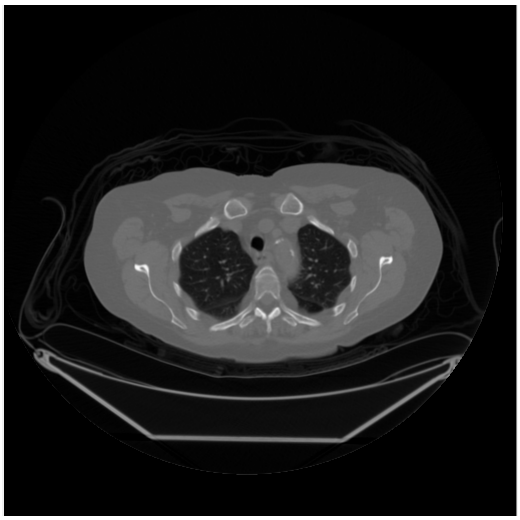

Fig. 4. DICOM image normalized

D. Tag Extraction and first screen

DICOM is not simply a format for storing images. It can contain many information about the pixels, the color palette, specifics about the imaging scanner used, information about the patient and many others.
This information can be extracted by any image using the right tag [1]. In the Graphic Interface will be shown in particular:

- the path from which the files have been taken;
- the dimensions of the 3D array;
- the modality of imaging;
- the date of the exam;
- the description of the exam.

All these information will be shown using tkinter's labels and entries. The first ones are used to display text boxes with fixed texts, while the latter ones are used to accept variable text.
Once all the operations discussed until this point are executed, the user will see the following screen [Fig. 5].

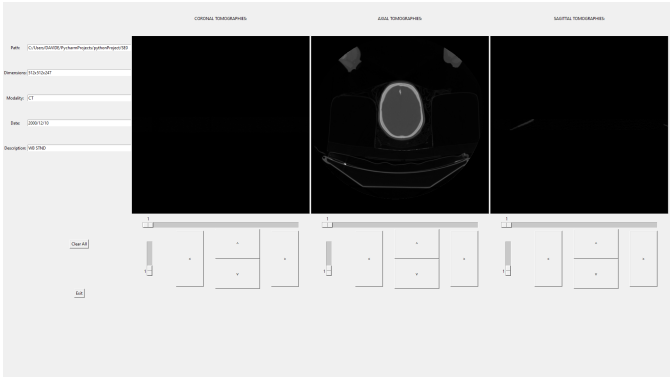

Fig. 5. first GUI screen

As we can see from the figure 5, there are three canvases on the User Interface. Each one of them shows the images on a specific axis. The first one is for the coronal plane, the second is for the axial plane and the third is for the sagittal plane. A canvas is a tkinter widget for displaying images. Because tkinter does not support numpy.array, using the *Pillow library*, the images are converted to a PIL-image format, which is supported.
For each view the images are created by slicing the 3D array on a different axis.

III. Functionalities

There are three different functionalities that are built for the user:

- the slider for changing image [III-A];
- the zoom slider to enlarge the image and arrows to move the image [III-B];
- the manual segmentation that can be done using the mouse [III-C].

A. Slider

For each canvas, there is a slider which can assume different values (depending on the dimensions of the image and the number of files in the chosen folder) by simply scrolling it with the mouse. Every time a new value is selected, the widget calls a function which deletes the previous image on the canvas and replaces it with the one corresponding to the selected number. On the sagittal and coronal axis, the image is also interpolated in order to have those images with the same dimensions of the axial ones.

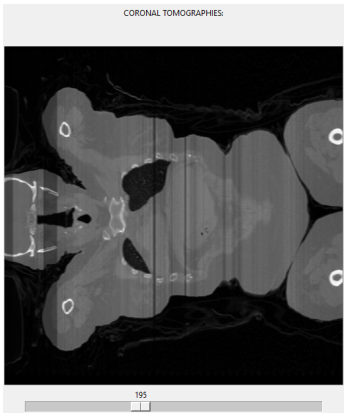

Fig. 6. The canvas shows the 195th image as indicated by the slider

B. Zoom

The image can be zoomed for a maximum of four times its original dimensions. The access to this function is granted by a slider where the user can select the zoom scale factor [Fig. 10]. Once the scale factor is selected, the previous image on the canvas is deleted and it is replaced by a new one. This picture represents the same slice, which has though been enlarged using the method `scipy.ndimage.zoom`, which uses spline interpolation to enlarge the image.
If the user selects a scale value different from `1` then the slider for changing images [III-A] is disabled. In fact, it is possible to move between slices only if the zoom slider is set to `1`.
Similarly, if the scale value is different from `1`, the segmentation [III-C] can not be done because the *Scipy library*'s zoom method does not interpolate masks.
While the image is zoomed, it can not be seen all at once because the image is much larger then the canvas on which it is shown. For this reason there is the possibility to move along the shown image.
This can be done using directional buttons [Fig. 10] that are located under every one of the three canvas.
There is a button for every cardinal direction (top, bottom, left, right). These four buttons can be pressed only when the image is actually zoomed, otherwise they will give no response.
When one of these buttons is pressed, the image is shifted of a step in the corresponding direction on the canvas.

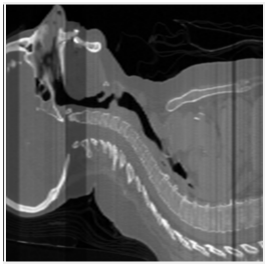

Fig. 7. Zoom scale: 2x

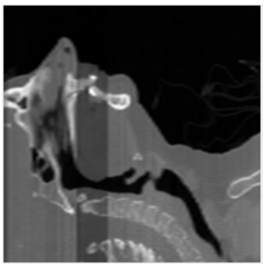

Fig. 8. Zoom scale: 3x

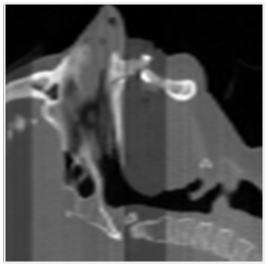

Fig. 9. Zoom scale: 4x

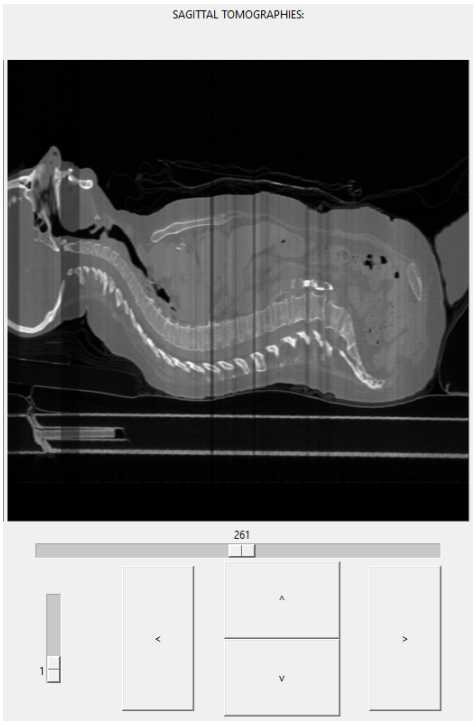

Fig. 10. Zoom slider and directional buttons

C. Manual segmentation

As mentioned in paragraph II-C, when the software first reads the dataset, it creates also a mask.
A mask is an array which contains only binary values. When the mask is overlapped to the 3D array it creates a masked array where all the 3D array's values which has been masked with a `1` are replaced by a pixel with a given value, while the others, masked with a `0`, gets maintained unaltered. For this GUI, the chosen value for a masked pixel is `255`, which is substantially a white pixel.
At first the mask has all `0`. The user can select the pixels to mask by simply pressing the left button of the mouse and moving it on a canvas. This event activates a function that works as follows:

1) the function selects a 5x5 pixel square with at the centre the pixel selected by the user with the mouse;
2) for every pixel in the square, if the pixel is contained in the picture's dimensions[1], the mask element with the same coordinates gets masked (`value = 1`);
3) the 3D array gets overlapped with the just modified mask creating a masked array;
4) the images contained on all three canvases get deleted;
5) on each canvas gets shown the selected slice of the masked array.

Because the masking is made not on a single 2D image, but directly on the 3D array, the modifications are seen contemporary on every plane even though the user interacts only with one canvas.
Finally, the result should be similar to the one in Fig. 11.
The manual segmentation has to be done separately from the zoom [III-B] as this function can not interpolate the mask. For this reason, even if an image on any of the canvases is zoomed, once the segmentation began, the image returns to its original dimensions.
There is in addition a button for deleting the changes to the mask. Once it is pressed, the mask returns to its initial configuration (all `0`) and simultaneously the images shown on the canvas are deleted and replaced with new images of the same slices without the old masked pixels.

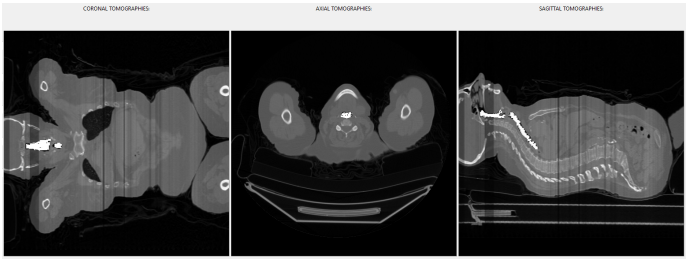

Fig. 11. Manual segmentation of the oropharynx and the tracheobronchial tree

[1]This is done because if the last pixel of the array is selected, the pixel count overflows and the function selects also the first pixels in the same line of the array

IV. Conclusion

The DICOM visualizer is efficient and it could be very helpful for doctors to diagnose the cause of the patient's symptoms, to monitor how well the body is responding to a treatment for a disease or condition, or to screen for different illnesses, such as cancer or heart disease. It can also be used to identify health conditions before the presence of any symptoms.
To achieve these purposes, the user is supported by the GUI. In fact, it can be used to study an organ or a tissue from different points of view simultaneously to have a more specific knowledge of the patient's health. The zoom can be used to investigate small details, which is crucial in the identification of possible diseases. Finally, the manual segmentation can be utilized to isolate and stress the desired object from the image.

References

[1] Innolitics, *DICOM Standard Browser*, (accessed May, 2021). [Online]. Available: https://dicom.innolitics.com/ciods/cr-image/general-series/00080060