Brain Cancer Segmentation

1 Introduction

The application created has as its main theme the analysis of medical images. Specifically, *MRI images* of the brains of patients are analyzed. The application is divided into 2 main parts, **Tumor segmentation** and **Graphical User Interface** (GUI).

2 Datasets used

Two different datasets were used to implement the entire program

- Dataset **.mat format** downloaded on Figshare. This brain tumor dataset containing 3064 T1-weighted contrast-inhanced images from 233 patients with three kinds of brain tumor: meningioma (708 slices), glioma (1426 slices), and pituitary tumor (930 slices). Due to the file size limit of repository, we split the whole dataset into 4 subsets, and achive them in 4 .zip files with each .zip file containing 766 slices.The 5-fold cross-validation indices are also provided. This data is organized in matlab data format (.mat file). Each file stores a struct containing the following fields for an image:
  - *cjdata.label* : 1 for meningioma, 2 for glioma, 3 for pituitary tumor
  - *cjdata.PID* : patient ID
  - *cjdata.image* : image data
  - *cjdata.tumorBorder*
  - *cjdata.tumorMask* : a binary image with 1s indicating tumor region
- Dataset **.jpg format** downloaded on Kaggle. This contains 3264 files so divided, there are two directories: "Testing" and "Training", each directory contains other 4 directories: glioma, pituitary, meningioma, no-tumor.

3 Tumor segmentation

Tumor segmentation is a very complex task because medical images can have different light intensities and recognizing the countourn of an abnormal area within brain tissue is not always trivial. Another problem, acquiring a dataset is not easy because there is the patient privacy. To resolve the goal, we divided the problem into three main parts

- Skull Stripping
- Mean tumor color extraction
- Segmentation

3.1 Skull Stripping

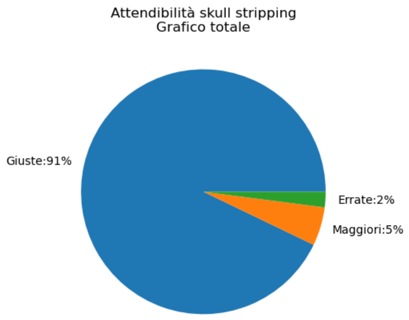
Skull Stripping is the **brain extraction** from a MRI. The goal is to eliminate the border because it influence the "*tumor detection*".

Algorithm

The skull stripping process is so implemented: It is calculated the **pixel intensity** presents with higher frequency considering a range, in a grayscale image, between 20 and 190 (pixels close to black and close to white are not considered). A *Canny filter* is applied to delimit the contours in the range that are from -10 to +10 with respect to the intensity of the pixel found earlier. We try to find the contour that most of all has the shape of a brain by taking only the brain tissue (gray) by extrapolating the **outer contour**. The returned image does not contain other parts of the border which, having different light intensities than the brain tissue, they can hinder the recognition of a tumor, as these parts could be mistaken for tumors. The goal is to have an image with only the brain, so that whatever sharp difference in intensity may be, most likely, the **segment of a tumor**.

3.1.1 Testing

To evaluate the accuracy of the program, tests were carried out on the entire .mat format dataset. To understand if the extraction of the contents of the brain was successful, the program initially calculates **the area outer contour** (brain tissue + edge), performs *skull stripping* and, on the contour that is returned, calculate *the area* and *the circularity.* If the area is too small compared to the outermost area, or if the contour is not very circular, the skull stripping process is likely did not work correctly, if not we assume the process worked correctly. There is an average case where the process most likely extracted the brain tissue but failed to completely remove the edge.



3.2 Mean brain tumor color extraction

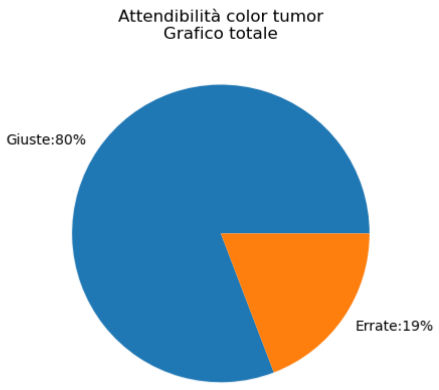
In this phase the goal is to analyze the image that contains only the **brain tissue without borders** and find the **average color of the tumor** (if any).

Algorithm

The average color of the brain is calculated by taking the intensity of the pixel present with greater frequency within image with brain tissue only. At this point all the contours are analyzed which, in addition to respecting some characteristics such as circularity and area, they have a distant average color compared to the average brain color. Difference between **average brain color** and **average contour color**, combined with *area* and *circularity* are the three properties that characterize and distinguish each contour found, the contour that respects these the most property is chosen as a possible tumor, and its average color is returned. The color of this outline may not be entirely accurate and, as we will see in the next step, it is only a further parameter which is used to compare it with the average color of the contours being analysed later more accurately.

3.2.1 Testing

In this phase, tests were carried out on 200 images taken from one of the 4 directories of the dataset format .mat. To know if the color found was correct or not, it was compared by analyzing the **average color of the tumor mask** present on every image in the dataset (*cjdata.tumorMask*).



3.3 Segmentation

Once the **average color of the tumor** and **the image of the brain tissue** have been acquired, it becomes easier to analyze the image to perform the "*tumor detection*" and finally the "*tumor segmentation*".

Algorithm

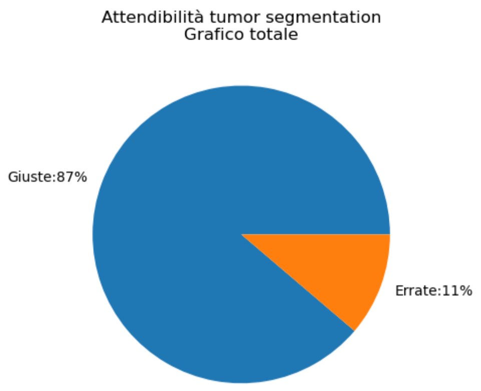
The **K-Means Clustering** algorithm is used with a k=6 which partitions the image so that each pixel belongs to one of the **6 clusters**, each cluster has a different **color intensity**. For each intensity (cluster), I create a **mask**, i.e. a black image in which only the pixels that refer to the current cluster I'm analyzing are entirely white. I find the **outlines of the current mask** and, for each outline, I check that it respects some properties, such as: circularity of a certain value, area with respect to the outermost contour (previously calculated), not too small and not too large, not too high difference between the average color of the contour, calculated with respect to the initial image and the average color of the tumor calculated in the previous step. All the contours that respect these properties are associated with a **unique numerical value** which is calculated at starting from these characteristics through a **weighted average**, the contour that has the highest weighted average e above a certain value it is considered a **tumor**. If the highest weighted average does not exceed this value then it means that the program has not found any tumors.

3.3.1 Testing

Here too tests were carried out to evaluate the precision of the segmentation on 200 images taken from one of the 4 directories of the .mat format dataset. To evaluate if, given an input image, the program finds the tumor correctly, the following images were compared

- Black image with only **the tumor found by the program**
- Black image with **the tumor present in the .mat file** (*cjdata.tumorMask*)

The image comparison function returns a **p** from 0 to 1 which indicates how equal two images are (the more p is close to 1 and the more similar the images are), for each p greater than or equal to 0.97 we assume that the tumor was accurately captured, in all other cases no. Therefore, cases in which the segmentation is not precise or cases in which the tumor is present but the program does not find it are considered wrong.



Segmentation conclusion

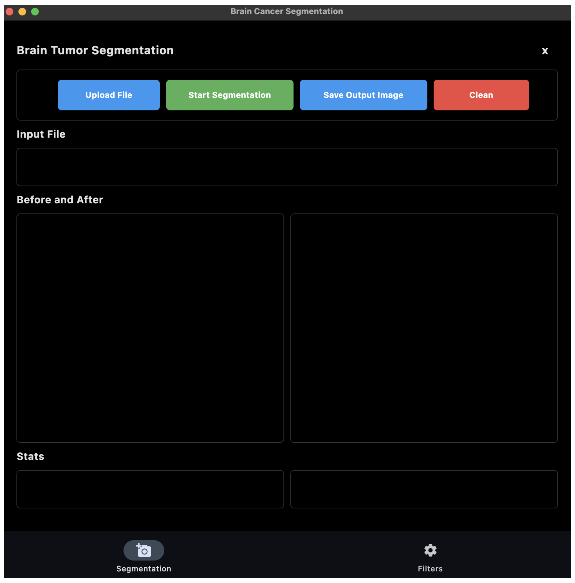
It is possible to conclude that, with a certain approximation, the segmentation of the tumor has a **percentage accuracy of 90%**, to try to improve even more by going by raising the percentage it is possible to try to improve *the accuracy of the skull stripping* and/or *average color extraction*, this because they influence the segmentation as the segmentation program takes this data as input.

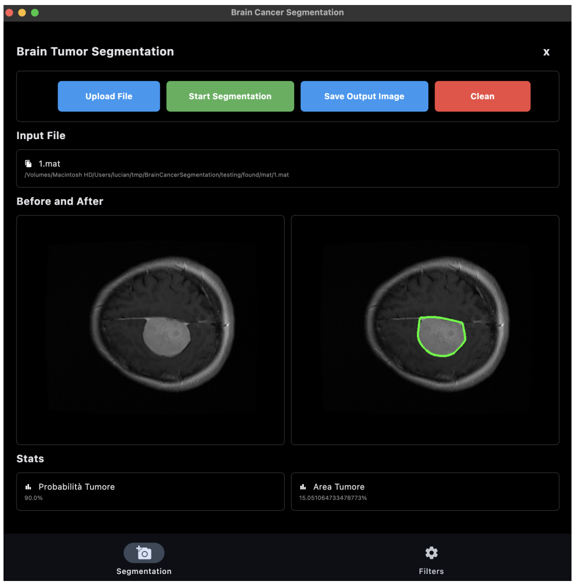
4 GUI

The graphical interface (*GUI*) in turn is divided into 2 sections **Brain Tumor Segmentation** and **Segmentation Filters**.

4.1 Brain Tumor Segmentation

By executing the **run.sh** file located in the main directory, an interface opens where the user has the possibility to load an image from the file system (accepted formats .mat/.jpg/.png). The chosen image will appear in the space reserved for the two "*Before and After*" images, under the "*Before*" section. Subsequently, it is possible to start **segmentation by selecting** the appropriate "*Start Segmentation*" button. *The image with the tumor* (*if any*) segmented will appear on the right, under the "*After*" section. Furthermore, **the area of the tumor found** and **the probability** that the program actually found a tumor. If the program does not find any tumor, the message "*TUMOR NOT FOUND*" will appear on the right. Finally, the user can save the image of the segmented tumor, selecting the "*Save Output*" button, choosing, inside your computer, where to go to store it.





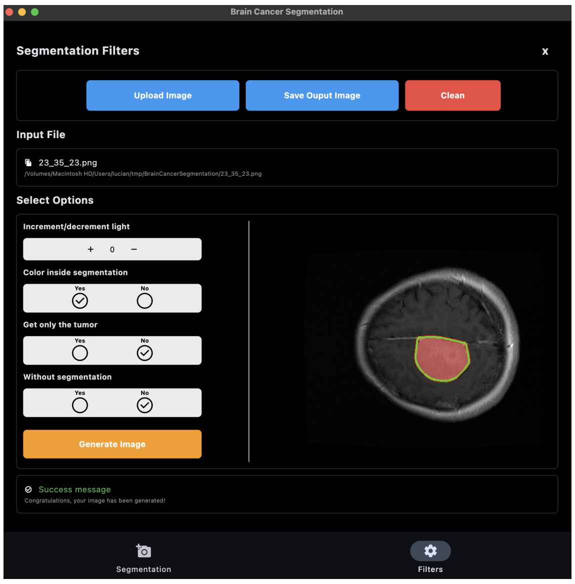
1.mat
/Volumes/Macintosh HD/Users/lucian/tmp/BrainCancerSegmentation/testing/found/mat/1.mat

Probabilità Tumore
90.0%

Area Tumore
15.051064733478773%

4.2 Segmentation Filters

The user can load an image that the program has generated at first, i.e. an image with the **segmented tumor** and apply filters. For example, it is possible to **increase/decrease the color of the tumor** in order to increase the contrast with the rest of the photo. It is possible to see and save the image **containing only the tumor**, on a black background and **color the inside of the segmentation**. Finally, it is possible to **eliminate the segment of the tumor** that the program had created, bringing the image as it was before the segmentation.



23_35_23.png
/Volumes/Macintosh HD/Users/lucian/tmp/BrainCancerSegmentation/23_35_23.png

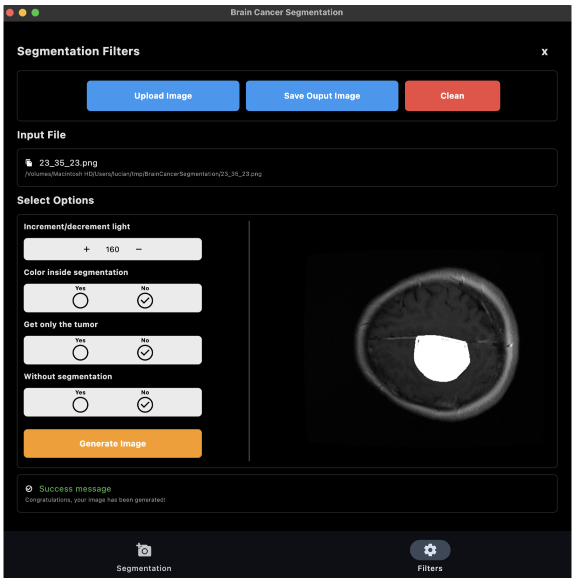
Increment/decrement light: + 0 −

Color inside segmentation: [✓ Yes] No

Get only the tumor: Yes [✓ No]

Without segmentation: Yes [✓ No]

Success message
Congratulations, your image has been generated!



23_35_23.png
/Volumes/Macintosh HD/Users/lucian/tmp/BrainCancerSegmentation/23_35_23.png

Increment/decrement light: + 160 −

Color inside segmentation: Yes [✓ No]

Get only the tumor: Yes [✓ No]

Without segmentation: Yes [✓ No]

Success message
Congratulations, your image has been generated!

4.3 Execution example

We illustrate in the following link an example of application execution.