Tumor Detection in MRI Scans

Annabelle Sauve

101080402

COMP 4102: Computer Vision

Rosa Azami

Carleton University

April 14, 2021

Contents

Tumor Detection in MRI Scans

Abstract

The goal of this project was to use computer vision concepts to create a program that detects tumors in MRI brain scans. After a patient gets imaging done, these images are analyzed by trained medical professionals which can then give a proper diagnosis. In recent years, computer vision used in medical imaging has received lots of attention due to the fact that it could be very helpful for busy doctors. Although there already exists some software that can, to a certain extent, analyze scans (MRI, CT, X-RAY, etc.), I wanted to learn more about how these algorithms actually work and see if I would be able to produce strong results.

Introduction

The problem of tumor detection itself is very challenging because the brain is a complex organ, and it can be difficult to analyse solely by looking at images without the help of computer vision algorithms. On the other hand, automatic tumor detection is also challenging because it requires lots of image manipulation and different steps before getting results and can often be computationally expensive. Since medical imaging is a very popular method for obtaining information about a patient, it makes sense that there has been an increase in demand for working tumor detection and analysis algorithms.

Background

This project was inspired by Vadmal & al.'s paper on magnetic resonance imaging (MRI) image analysis. In this paper, they describe different methods that are used to analyse these scans in hopes of finding brain tumors automatically. These methods are part of already existing

software such as Insight ToolKit (ITK) and FMRIB Software Library (FSL). The analysis process is divided into three main steps: preprocessing, segmentation, and feature extraction. I have followed these steps and have been able to write a successful tumor detector algorithm using the OpenCV Python library.

Approach

After the image is imported into the program, it goes through the preprocessing function. The results of this step can be observed in figures 1 and 2. First, noise is removed by applying the OpenCV non-local means denoising function with a filter of 5. I found that this filter number was the most efficient because even though I wanted to remove some noise, I still needed to keep some details so that the tumor could be detected properly. Then, bias correction was applied. I first tried a simple method of adjusting the brightness and contrast manually. Although this was working for a singular image, I wanted a program that was efficient for whatever image you would put through. After trying different methods, I decided to use histograms. First, the image histogram and its distribution are calculated. Then, the histogram is clipped at 3%. The percentage at which it was clipped was a very important factor when it came to the tumor detection. I tried different values and found that 3% was the most efficient one for all the data. Finally, from the clipped histogram, the minimum and maximum gray values were calculated and were then used to obtain gain and bias value (brightness and contrast), which were applied using the convertScaleAbs function. In this step I also applied a

Figure 1. Original Image

Figure 2. Image after preprocessing

sharpening filter, but then realized that it was not necessary and was actually making it harder for the program to detect a tumor, and therefore was removed.

After the preprocessing function, the skull stripping function is applied. The goal of this function is literally to strip the skull out of the image, meaning that the output will be the brain by itself (see Figure 3). First, the image is thresholded using an Otsu threshold (a type of binary threshold, see Figure 4). Then, the skull is found by computing the maximum connected component of the Otsu image. Since the skull is usually very bright in MRI images, and it goes around the brain (and is therefore connected), the maximum connected component method is very effective.

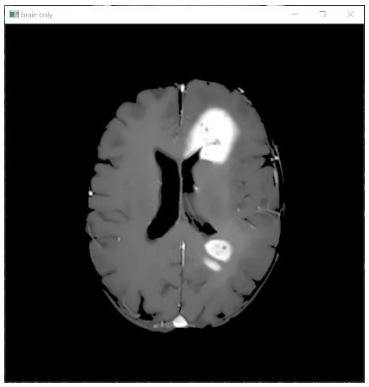

Figure 3. Brain only

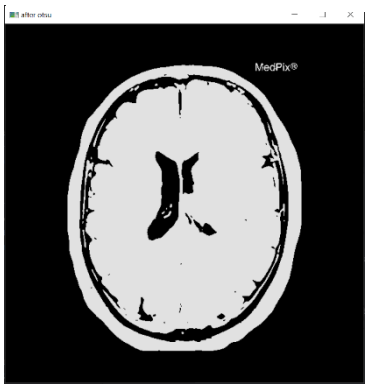

Figure 4. Otsu threshold

Now that all of the image preprocessing steps are done, the program can go on to the tumor detection, which starts at segmentation. In this step, the image is once again thresholded but this time with a Tozero threshold from values 125-255, which gives gradients (instead of black and white, as seen in a binary threshold). Segmentation separates the image into layers, as can be seen in Figure 5.

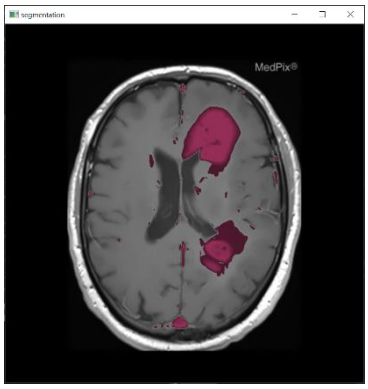

Figure 5. Image after segmentation

To find the tumor a closing function is first applied with a 10x5 kernel, followed by erosion and dilation to remove additional noise. Afterwards, the edges of the image are detected using a built-in canny edge detector. Then, the edge image is passed into a function to find its contours. The maximum contour found is assumed to be the

tumor, since it is the biggest detectable feature in the original brain image. Finally, the colored tumor contour is added to the image for easy readability.

Figure 6. Brain atlas

I had originally given myself the extra challenge of locating the brain structures the tumors were affecting. To do so, I superimposed a brain atlas (see Figure 6) onto the image and iterated through the tumor contour pixels to identify the brain structures they were a part of. Since I was not able to find a way perfectly align the atlas to the brains in the images, this worked semi-successfully. If I had more time, this is something I would have tried to fix.

Perfecting this method for tumor detection in MRI scans took many trials and errors for the main reason that I wanted to implement a program that would be universally effective across many different images. Although simple, I believe this solution is still very strong, with 13/16 tumors correctly detected in my dataset (see Figures 7.1-7.16).

Results

The figures in Table 1 show the results of the tumor detection program. The tumors are outlined in green. Figures 7.1, 7.2, and 7.4 are the only three images which failed. I believed 7.1 was no successful since the tumor is not as apparent as others. Instead of detecting the tumor in the bottom right, it assumed that the ventricles and the corpus callosum in the middle of the brain were part of the tumors, since they were the brightest feature. I assume something similar happened in Figure 7.4, since the tumor on the left does not really have a clear outline. As for Figure 7.2, I am honestly unsure why the algorithm was not successful since the tumor is clearly outlined.

Table 1. Results of tumor detection algorithm.

Figure 7.1 Unsuccessful detection.

Figure 7.2 Unsuccessful detection.

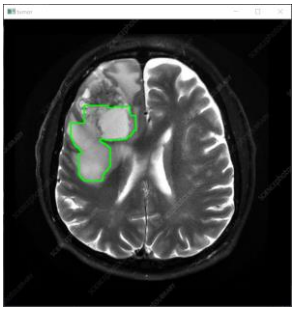

Figure 7.3 Successful detection.

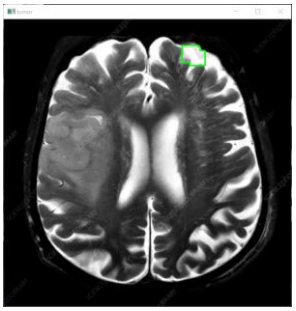

Figure 7.4 Unsuccessful detection.

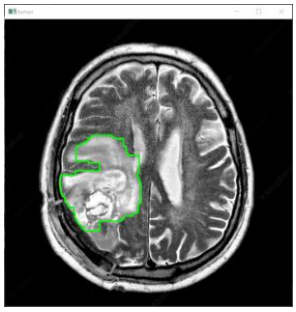

Figure 7.5 Successful detection

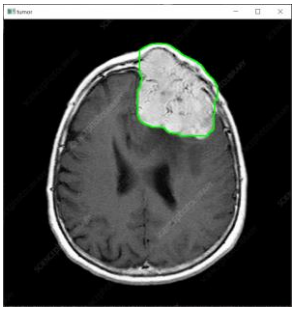

Figure 7.6 Successful detection

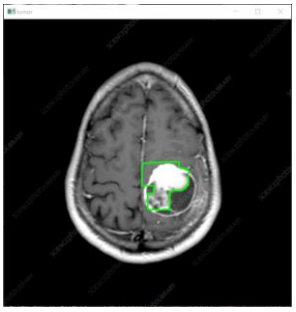

Figure 7.7 Successful detection

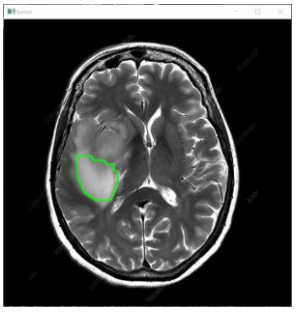

Figure 7.8 Successful detection

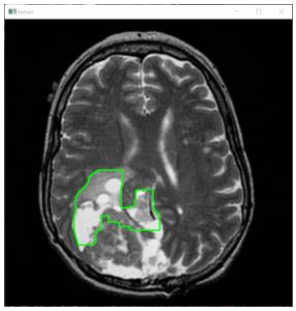

Figure 7.9 Successful detection

Figure 7.10 Successful detection

Figure 7.11 Successful detection

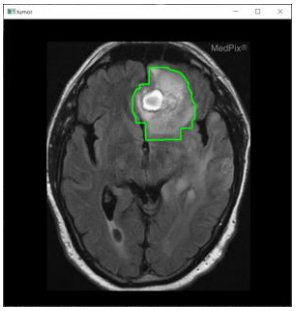

Figure 7.12 Successful detection

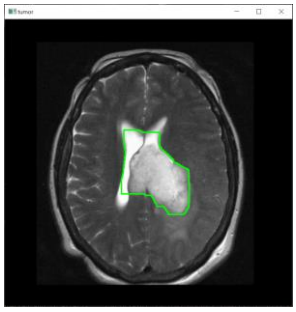

Figure 7.13 Successful detection

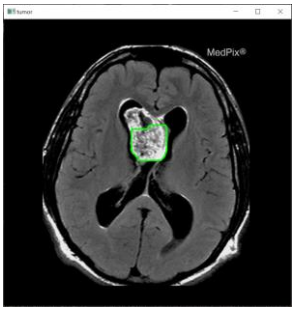

Figure 7.14 Successful detection

Figure 7.15 Successful detection

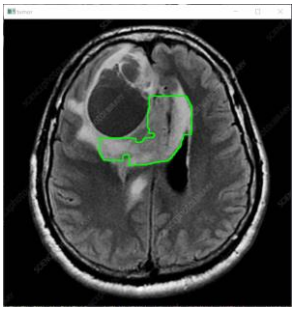

Figure 7.16 Successful detection

As mentioned, although the location and classification of the tumors were not perfect, I was still able to get interesting results. As an example, Figure 8 shows the location output for the tumor in Figure 7.13. When looking at the tumor image, I can confidently conclude that it does in fact affect the basal ganglia, the corpus callosum and the temporal lobe. Additionally, each tumor image has its own text window which contains location information. This can be seen in the attached screen recording.

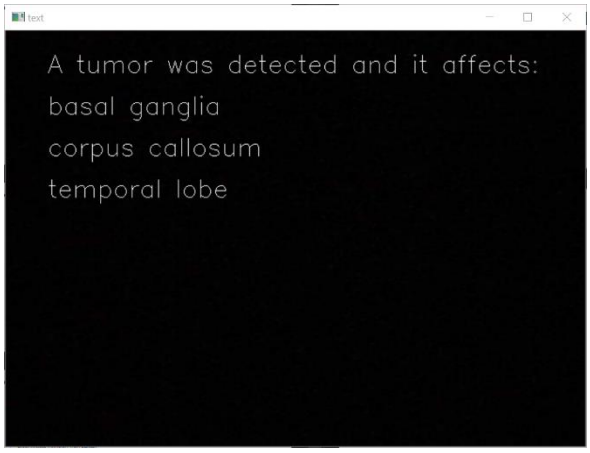

Figure 8. Location output for Figure 7.13.

GitHub page

https://github.com/annasauve/COMP4102-FinalProject

Please refer to the README.md document for directive on how to run the code locally. Additionally, a screen recording video showing the code and its output has been submitted with this project (COMP 4102 – Project screen recording.mp4).

References

National Library of Medicine *MedPix* database. https://medpix.nlm.nih.gov/home

Science Photo Library database. https://www.sciencephoto.com/

Vadmal, V., Junno, G., Badve, C., Huang, W., Waite, K. A., Barnholtz-Sloan, J. S. (2020, April). MRI image analysis methods and applications: an algorithmic perspective using brain tumors as an exemplar. ***Neuro-Oncology Advances***, 2(1). https://doi.org/10.1093/noajnl/vdaa049