Challenge 2: Cortical thickness challenge

Gabriel Morariu - 002246879
GMORARIU18@ubishops.ca

Punan Chowdhury - 002323073
PCHOWDHURY21@ubishops.ca

Dept. of Computer Science
Bishop's University
Sherbrooke, QC Canada

Submitted to Dr. Russell Butler
CS590 – Masters Project

Project Summary:

The goal of this challenge is to develop an algorithm to estimate a cortical thickness map based on a raw T1-weighted image of a brain. The cortical thickness corresponds to the thickness of the brain's gray matter at every point on it. The thickness is defined by the distance between the white matter surface of the brain (the surface that divides the white matter and gray matter) and the pial surface (the outer gray matter surface).

This project was made using Python and limited libraries.
Libraries used:

- Numpy
- Matplotlib
- Nibabel
- Skimage
- Default Python math library
- Scipy

The Program:

We started by importing the libraries and installing nibabel. Once done, we loaded the raw t1 map for subject 1 as well as the thickness map that were provided. We wanted to create the code with subject 1 so that we could verify our result's accuracy with the subject 1 map before we ran the code with subject 2.

As shown in the video presentation had with the professors, we next applied a gaussian filter to the raw t1 image to reduce noise and help with segmenting it into different surfaces and areas as needed.

The first step to getting our desired result is to segment the white matter boundary/surface. We did this my first isolating the entire white matter with a threshold. We then applied a median filter to help with the dilation that is applied right after. Finally, we subtracted the median white matter from the dilated median white matter to get the resulting white matter boundary.

The second step is very similar to the first, but instead of segmenting the white matter, we segment the gray matter. As before, we isolated the gray matter with threshold boundaries and made sure that the resulting values were floats (this will avoid errors later on). We then applied a median filter and dilation as before and subtracted the gray matter from the result to get the gray matter boundary.

The third step builds on the second step and helped us find the pial surface (outer gray matter boundary). We first dilated the white matter surface found in the first step and them subtracted it from the gray matter boundary found in the second step. This resulted in an array with only the outer surface of the gray matter boundary. One extra step that we needed to perform was using a threshold to remove the negative values that appeared as a result of the previous subtraction.

After finding the white matter surface and pial surface, we then got arrays of the non-zero coordinates of each array in order to find the distance values between them.
We started by finding the distance between the white matter boundary and pial surface for each white matter boundary point. To do this we used a for loop and the formula for Euclidean distance. We also broadcasted the white matter and pial surface coordinates in order to use the Euclidean formula. Once we got all distances between one white matter point and all pial surface points, we found the minimum distance and recorded that value to the same point on the white matter boundary (which corresponds to the cortical thickness at each white matter boundary point).

Once all the white matter point distance values were recorded, we moved on to filing the rest of the gray matter points with the value taken from the closest white matter boundary point. This was achieved by first getting the array of non-zero gray matter points. Then, we used a for loop to iterate trough them and find the closest white matter boundary point for each one. This was achieved using the KDTree method to find the closest neighbor to each gray matter point on the white matter boundary. Once found, the gray matter point would inherit the distance value/cortical thickness from the white matter point. This resulted in our final image of a gray matter cortical thickness map.

Results:

We first ran the program with the subject 1 raw t1 file and got a result not too different from the subject 1 thickness map that was provided with darker green being a small thickness and lighter green/yellow being a larger thickness. The main difference were the colors and some of the shape or the thickness map. When viewing the middle slice of the brain, our result has a few extra features in the middle of the brain and at the top and bottom. These are most likely because we did not use enough filters or optimize the thresholds when segmenting the gray matter and white matter. The colors being different between our result and the provided map is likely due to differences in distance calculation methods. In fact, the brighter yellow areas on our result signify a larger cortical thickness despite not being very thick. This likely occurred because the gray matter pixels took the distance value of a white matter pixel that was further away because of unoptimized segmentation.

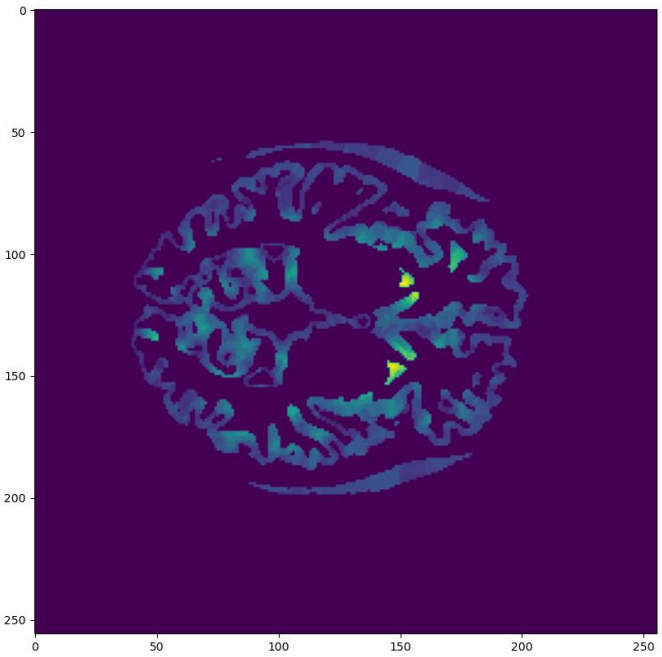

Figure 1: Middle slice of obtained subject 1 cortical thickness map

We then ran the program with the subject 1 raw t1 file and got around the same kind of distance map results as with subject 1. This time we also decided to change the display color map to better notice the differences in color (with darker red being less thick and lighter red/orange/yellow being more thick).

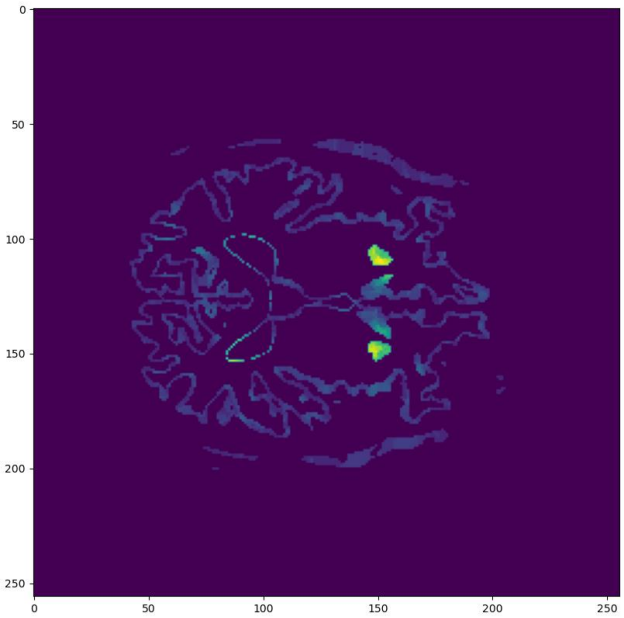

Figure 2: Middle slice of obtained subject 2 cortical thickness map (green color map)

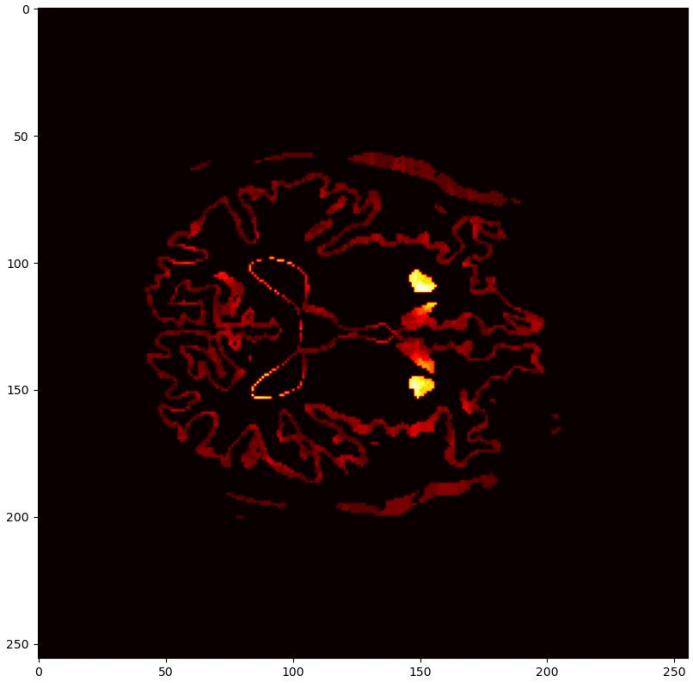

Figure 3: Middle slice of obtained subject 2 cortical thickness map (red color map)

Challenges faced:

We faced quite a few challenges during the development of our solution.

First, we got stuck when segmenting the pial surface as we did not realize that we could remove the negative values of the pial surface array after subtracting the white matter boundary.

Second, we got stuck on how to find the distance between the two surfaces and when we did find the Euclidean distance method, we got stuck when trying to implement it as we hadn't found the list of non-zero coordinates first. This caused a lot of confusion with trying to get the units and shapes of different arrays to be compatible for calculations; a constant problem we faced throughout the project.

Third, while we did find the KDTree solution quite easily and thought that it would result in a relatively quick run time, the coloring of all gray matter pixels took upwards of 8 hours to complete. We tried to add conditions to speed up the runtime, such as adding an upper bound to the distance calculation, but nothing seemed to speed it up. If we were to continue to iterate on this project, this would be the first problem that we would tackle.