CV Final Project Report

3D Reconstruction of Segmented Image Slices

Richanshu Jha rj1469@nyu.edu
Tejas Shetty trs389@nyu.edu

Contents

Introduction

In this work we have created a module that performs 3D reconstruction of the human brain. We initially perform image segmentation MRI scans of the brain and then generated a 3D model by compositing the segmentation results into a point cloud and representing it in an interactive 3D interface. This work acts as a proof of concept for 3-D reconstruction and can be used as a model for any 2-D sliced data. For example, we can use this moel to reconstruct any real-life organs provided we get their sliced images.
Such reconstruction can help to visualize the organ more intuitively and thus in case of any medical issues with the organs, helps in better diagnosis.

High Level Details

- On a high level, the project involves two modules, the segmentation module and the reconstruction module.
- The segmentation module implements the Snake segmentation module in python and generates contours for each sliced image.
- The reconstruction module uses Open GL to stitch the points between contours of successive layers and generate the reconstructed image. Here, we have used the triangulation method, i.e. generate a triangle between two points of one layer and one on the next layer.

Configurations and controllable parameters

- We have made the implementation as generic as possible by providing a configuration json that controls the behavior of the codes.
- It contains attributes like alpha, beta, gamma, max iterations, folder location of the images, the y-separation (horizontal distance) between layers and a unique identifier for each run called session name. More details about the configuration json can be found in the README file.

- The 3D reconstruction that we have performed provides an interactive interface via pygame. We have provided controls to change color, orientation, display/hide lines/triangles/solid lines formed, expand/contract the image etc. The README file provides more details about the possible controls. The images shown below in results section shows images

``

Datasets

One of the primary problems of this project was to find a dataset of MRI images that have a sliced image of the brain. While MRI images are available in abundance, a set which iteratively contains cross sections of the brain could not be found. We found datasets that contain around 4-5 layers of the human brain for a number of subjects. However, a dataset with fewer subjects and more layers could not be found.

A second issue that we faced was that datasets have a large size (~GBs) and thus could be used for processing. As an example, consider the dataset from the NYU Langone site (Link). The extracted dataset was ~200 GBs and thus could not be used for processing. In this dataset, we also had issues to extract the data as the final file did not contain any extension and thus could not be used.

We thus used the below methods to generate data:

1) We generated a set of 147 sliced images of the brain. For this, we used a processed a video file and split it into images using Open CV.
2) A second dataset containing 40 images was generated from the above set and used during development of code.

We found that results using dataset-1 were a better representation of the reconstructed brain and they have been described in the sections below.

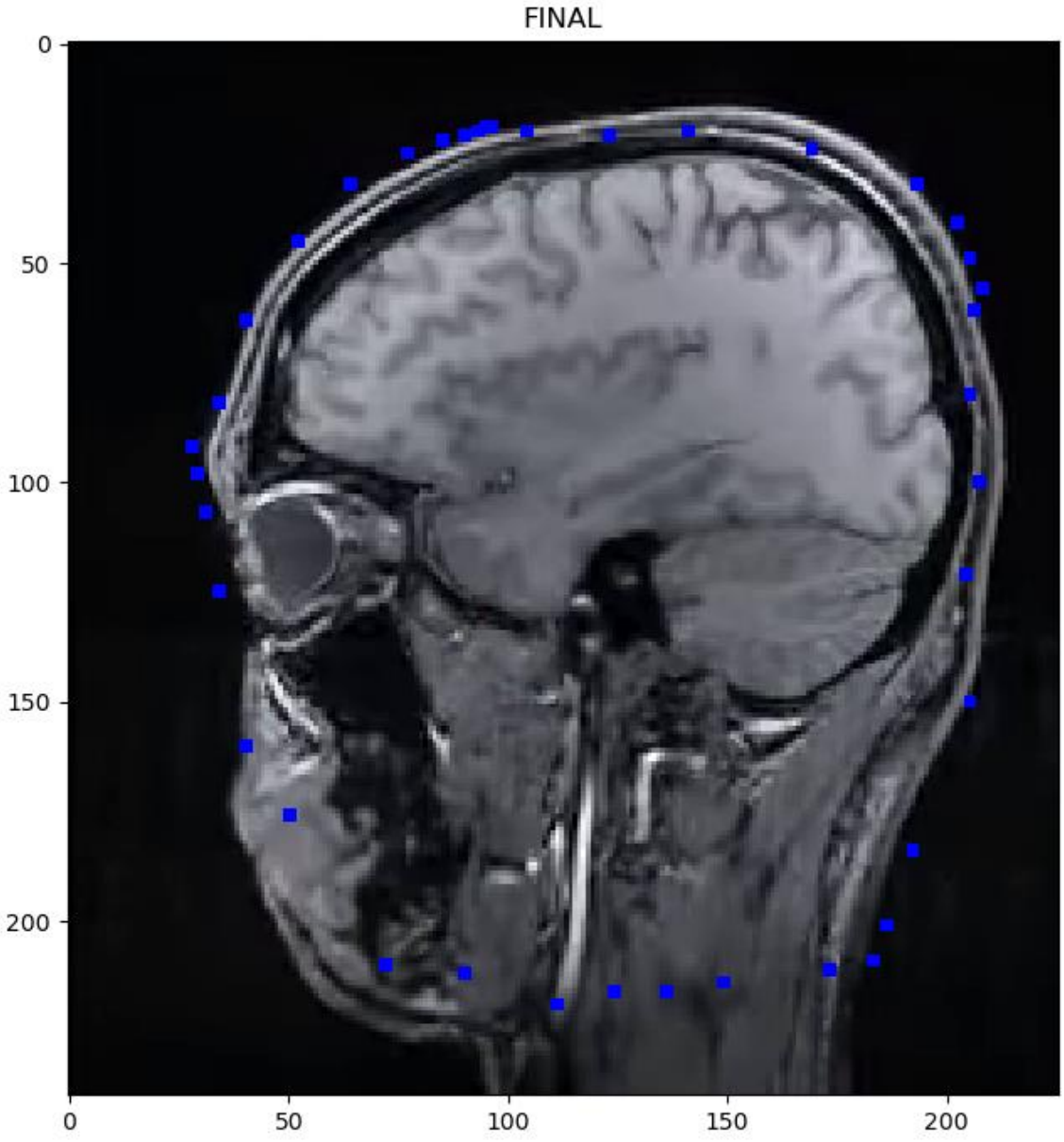
Detailed Workflow

1) The codebase contains two files, segmentor.py and viewer.py The flow starts in viewer.py where we get the configurations to be used. We have provided two options here:
    a. Use a preprocessed data file which contains the segmentation results and the settings used and generate the 3D model
    b. Read the images, perform segmentation and then generate the 3D model.
2) In the second approach, the results of segmentation are iteratively stored in a json file. Thus, the code flow is same for both the approaches w.r.t. reconstruction. Below, we describe the second approach. The flow for the first approach can be intuitively understood.
3) The workflow starts from viewer.py where we first perform the settings for OpenGL like display size, color, rotation, keyboard controls etc. Then, we set a forever running while loop. In each iteration, a sliced image is loaded and its contour is generated using Snake segmentation. It then stores this contour in the json file mentioned above.
4) This contour is then used by OpenGL functions to reconstruct the 3 D model. It basically constructs triangles using two adjacent points on the first contour and the nearest point in the next contour. This process is continued for all the points in the contour.
5) After all the images are processed, the json generated above is stored.

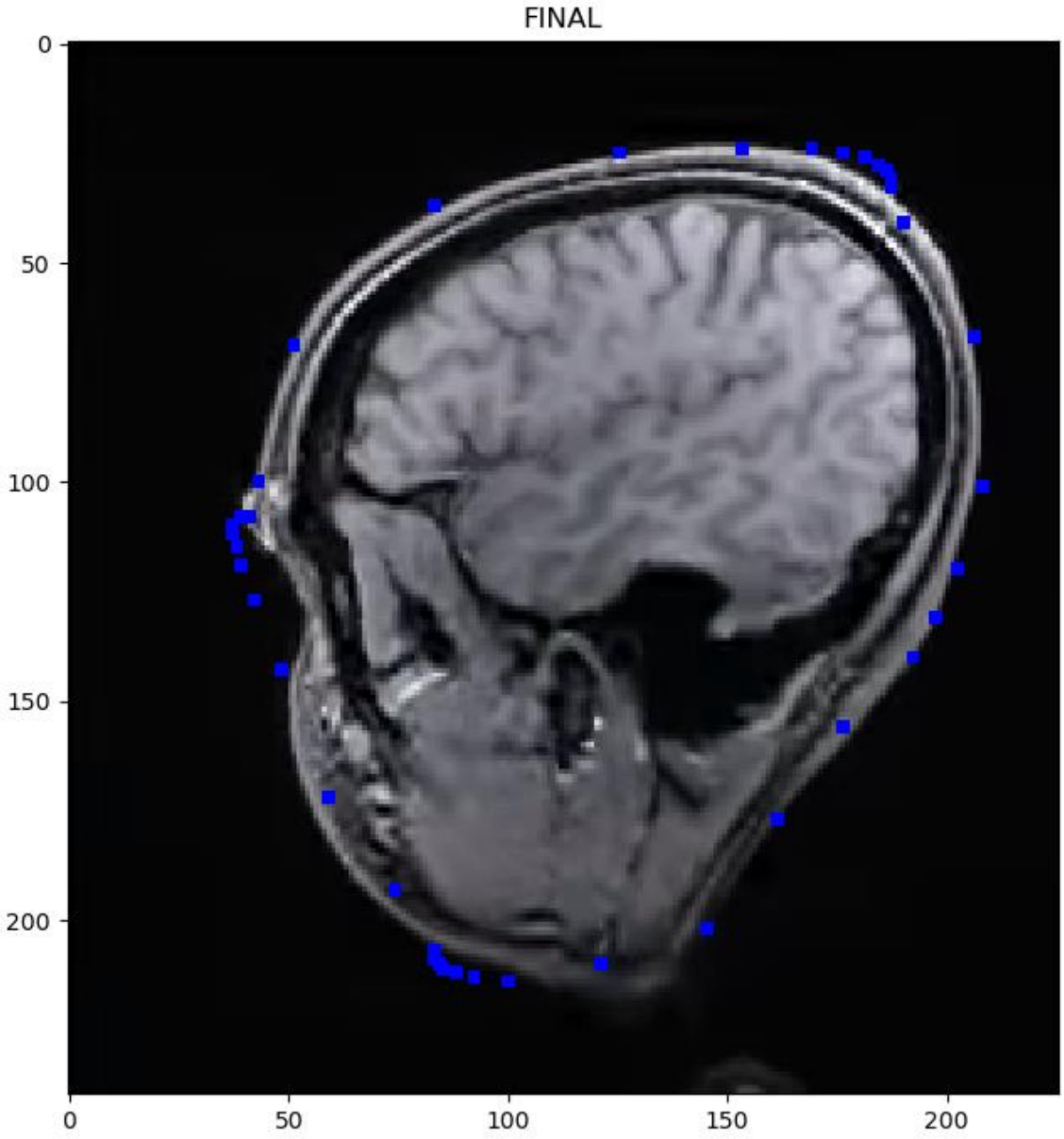
Results

Below are the results of this implementation. We see below the images of the, segmented images, 3D rendering in progress and the final reconstructed brain.

Image Segmentation images:



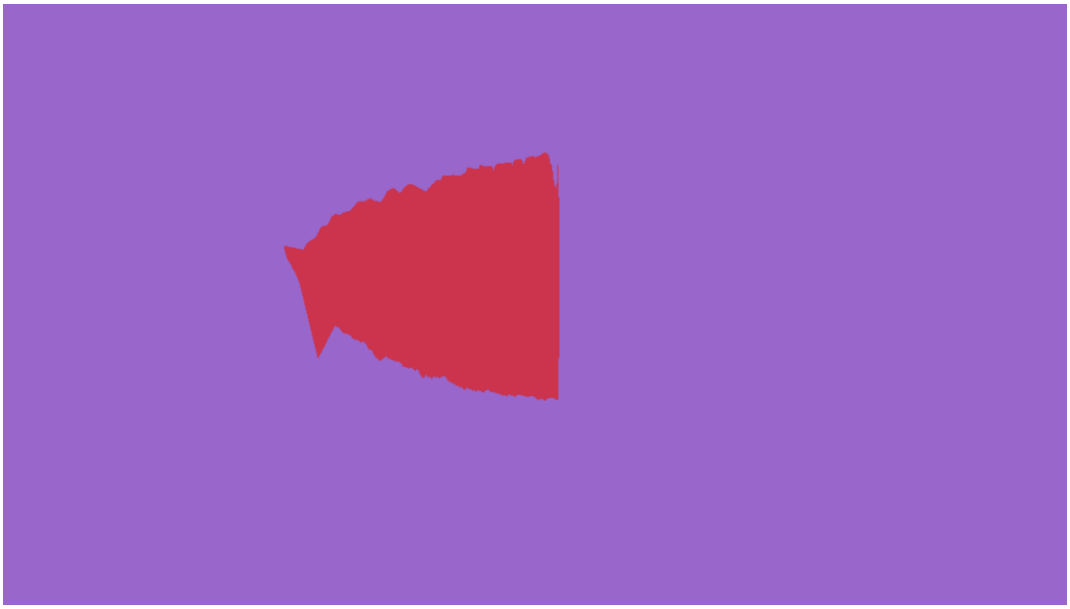
Segmented Image -1 of the brain



Segmented image-2 of the brain

Brain Reconstruction Images – In Progress

Brain reconstruction image -1 In progress



Brain reconstruction image -2 In progress

Final reconstructed images

Reconstructed brain image – Side View

Reconstructed brain image – Front view (Triangles shown for visual effect)

Challenges Faced

We faced various challenges during this implementation which are mentioned below:

1) The data collection part had some issues which have been described above. Since we are running the project on a personal computer, we have processing limitations to adhere to.
2) The processing time needed for a render was very large. For the image set with 147 images, the rendering took about 45 mins to complete. Thus, we had to invest a lot of time fine-tuning the final results.
3) Snake segmentation parameter adjustment took some efforts since we had to get optimal parameters for 147 images.
4) We learnt OpenGL in the process since we had limited experience and we had to learn on the way during the project work.

Strengths of this implementation

- We were able to reconstruct the 3-D image of the brain by using two relatively simple ideas, image segmentation and later stitching using OpenGL. The section below shows the results generated and they give a decent reconstruction of the brain, a complex object to reconstruct.
- This idea can be extended to any other organs as well since it does not link to the brain in particular. Thus, it can be used to reconstruct organs like the liver, heart etc. using medically scanned images.
- We further posit that this idea could be used in any other filed where we have sliced images available.
- The design of the code is very generic and configuration driven, thus we can set the segmentation parameters, image directories etc very easily and run the codebase. The codes also have been written in a modular fashion so that changes/additions in the future can be handily incorporated.
- The implementation is light on module requirements, we just need python with OpenGL, pygame, Open CV and associated python libraries. Thus, it can be ubiquitously run on a variety of systems.

Weaknesses in this implementation

- This technique uses images that are obtained for multiple cross sections of a 3D object. Cases where such data is not present, this implementation would not be useful.
- We use snake segmentation for finding the contour. This technique is sensitive to the parameters used and thus may not provide accurate contours for every set of image. We can mitigate this issue by thresholding, filtering beforehand for better boundary detection.
- We have used PyOpenGL here which does not use GPU inherently. It is a python based implementation of Open GL and thus is not optimized enough. A more robust implementation would be to use C++ which provides direct integration with OpenGL thus greater control and consecutively more scope for optimization.

Future Scope

- As discussed above, we can implement the model on a GPU system using native Open GL giving greater control and faster results.
- The segmentation module implemented does not do thresholding or filtering. We can try these methods and then check if the results improve.
- The initial contour that we set up for snake segmentation was very close to the actual image boundary. It did not have a large impact on our results however, padding the image and then trying the same could give us better results.
- The code has been set in a way that it reads inputs from a json file and performs operations. This provides it the scope for using the codebase as an API by writing a wrapper on top that accepts inputs from the user and writes them to the json files.