

University of Reading

Department of Computer Science

John Smith

Supervisor: Dr David Burton

**Declaration**

I, **John Smith** of the Department of Computer Science, University of Reading, confirm that this is my own work and the figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person’s work, except where the works of other have been explicitly acknowledged, quoted and referenced.

I understand that, if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised according.

I give consent to a copy of my report being shared with future students as an exemplar.

I give consent for my work to be made more widely to members of UoR and the public with an interest in teaching, learning and research.

**John Smith**

**April 22, 2021**

**Acknowledgement**

This is to acknowledge the support provided by my supervisor Dr David Burton throughout the various stages of this final year project. I also wish to thank my family and friends for their support during my degree studies and in particular throughout my final year studies.

Table of Contents

1. Introduction 4

2. Literature review. 5

2.1 History of neuroimaging techniques. 5

2.1.1 CT Scan 6

2.1.2 MRI 7

2.2 Tools used in neuroimaging. 7

2.2.1 Limitations for a scientist to use neuroimaging tools. 7

2.2.2 Solutions for using neuroimaging tools. 8

2.3 Comparison of neuroimaging tools. 8

2.4 Freesurfer and its functions. 12

2.4.1 FreeSurfer Limitation. 13

2.4.2 Solution 14

3. Methodology 14

3.1 GUI connection with command-line interface. 16

3.2 Implementation of GUI commands with buttons. 17

3.2 Functions for selecting the folder and image and recon-all. 17

3.3 Recon-all Function: 19

3.4 Data logging and analysis: 20

3.5 Function for the progress bar. 21

3.6 Simple Mail Transfer Protocol function. 23

3.7 GUI Implementation 24

4. Testing and Validation 25

4.1 Environment set up test: 26

4.2 Folder name validation and testing. 27

4.3 Image selection and testing. 28

4.4 Recon-all testing. 28

4.5 Progress bar testing. 29

4.6 SMTP testing. 30

5. Results 31

6. Conclusion and future work 31

7. Reflection 32

References: 33

1. Introduction

In today’s world, we can see the enhancement of technology in each and every sector. From creating a life to destroying a life, technology plays an important role in this century. It is unbelievably evolving that someday the world might look like something out of the fair world. Even now if we compare this technology to the previous centuries people then it’s heaven in terms of services and technologies we have. In ancient times people used to die young age because there were no appropriate medical treatments available. With simple diseases like cold and flu, people used to die but now we can even cure Cancer up to a certain stage and I am sure scientists will discover them sooner or later.

People can have diseases in any part of the body and it is easy to diagnose in all the other body parts compared to our human brain. The brain is the most sophisticated and crucial part of our body without this we cannot exist or even become conscious and this is why it was extremely difficult for the person to live when they had a brain disease. So to discover what is happening in the brain and to study the brain neuroimaging techniques were evolved.

Over the past centuries, scientists have discovered many neuroimaging techniques and built a machine to examine the brain. However, the development of the Graphical User Interface for these techniques is not updated. Nowadays the world is computerized and we want to do everything with simple buttons and touch screens. Processing the neuroimaging using the command line interface and programming language takes a long time and requires immense knowledge about each and every command. It will be challenging for scientists and medical professionals to learn all the things about the computer and its language.

That is why the main aim of this project is to create a Graphical User Interface for the FreeSurfer which helps scientists and medical professionals to run the image processing without having to press single command. There are lots of software for neuroimaging processes available in the market but FreeSurfer is the only software suite that includes almost everything that is needed for the human brain analysis and visualisation.

Having all the required methods and tools for the human brain analysis FreeSurfer is very popular among scientists and medical professionals but the only drawback of this software is that it doesn’t have a Graphical user interface. This means all the processes should be done by passing the commands to the command-line interface in the OS.

To solve the problem of user interface and to save time for the thousands of researchers, medical professionals and scientists I decided to devote myself to create a GUI for the

FreeSurfer which will execute all the processes required for processing the neuroimaging, show the progress of the process and send the email to the user after the process is completed as this process is very long and take more than 6 hours to complete.

2. Literature review.

2.1 History of neuroimaging techniques.

The history of neuroimaging dates way back to the 1880s when Angelo Mosso invented a technique called ‘human circulation balance’. This technique was able to non-invasively measure the redistribution of blood flow during intellectual and emotional activity(Sousa, 2010). However, the experiments performed and the specific working of the instrument Mosso developed remained mostly unknown. Later in December 1895, W.C. Röntgen discovered X-rays(Narang and Jha, 2017). He studied the properties of the new type of radiation which can go through screens of notable thickness for several weeks after discovering X-rays. The news of the X-ray discovery spread like a wildfire and attracted enormous attention and interest from the public, scientists and physicists. The use of X-rays on patients began as early as January 1896 to investigate the skeleton and gradually the lungs and other organs of the body. This gave birth to radiology.

The first neurosurgeon to use X-rays in neuroscience was Sir Victor Horsley in the late 19th centurySpecialists in neuroscience and physicist tried to X-ray the skull of patients to diagnose lesions (type of damage to any part of the brain) in the brain. Schuller was the first person to use “Neuroradiology” as pathologies of the brain using skull radiographs. After this, the popularity grew much stronger and many papers were published showing patients with different diseases like brain tumours. With the growing popularity and research in this area, it came with limitations. The main limitation was that it needed a long period of immobility to acquire images, soft tissue pathologies could not detect, and neoplasms without bony involvement or classification were hard to visualise(Sousa, 2010)

To take any action or process the diagnosis of the lesions, patients need enough time. However, by the time pathologies were identified on skull radiographs it would be too late. So to overcome this Walter Dandy invented an idea. The idea was to diagnose pathologies and localize lesions by analysing the changes in the morphology of verticles by injecting contrast material (Corsello et al.). Later on, when he discovered that the air from the verticles escapes into the subarachnoid space, he discovered Pneumoencephalography. At that time it was a common medical procedure in which most of the cerebrospinal fluid was drained from

around the brain and replaced with air and oxygen which allows the structure of the brain to show up more clearly when an X-ray is done. Through the years of research and study, these techniques helped to identify the nature of the disease in neuroscience and more improvements to these techniques discovered by Walter Dandy led to the invention of the skull table which assisted in moving the air to the different parts of the brain as need. (Corsello et al.). Following the Walter Dandy idea, the air was used as the contrast medium for the first Myelography in the early 20th century. During that time there was a discovery of iodine as a safe radiopaque contrast. As the iodine goes will with an oily substance this leads to a case of arachnoiditis and iodine was replaced by water over the next 50 years. (Tan and Yip).The next discovery that happened after the Pneumoencephalography was Angiography. The discovery of Angiography started with a dream of Egas Moniz where he thought of making cerebral vessels opaque or brain on imaging. (Tan and Yip). To make this possible Egas Moniz conducted various experiments on animals and as he was progressing he conducted a cutdown to reach the carotid artery and injected contrast material directly.

2.1.1 CT Scan

As we can see that the process of imaging the human brain has progressed throughout ancient times but the modern era of neuroimaging began when computerized tomography (CT) was developed by Godfrey Hounsfield. Originally the idea was by physicist William Oldendorf and mathematically developed by Allan Cormack. (“1979: Allan MacLeod Cormack (1924-1998) | StJohns”)

In 1972 a woman was scanned with CT with suspicion of a tumour in her brain. She was the first person to be scanned using CT. After the first scan was done another few scans were conducted. Later on, In 1973 CT imaging of the brain was published and this discovery proved to be a ground-breaking discovery of that time. Hounsfield and Cormack discovery made them very famous and many people knew their names in the medical community worldwide. In 1979 they received the Nobel prize for their contribution. Multiple improvements and refinements were done to the original CT scanners as they showed only the imaging of the head but after the improvements, CT scanners were able to scan the entire human body which led to reduce in radiation dosage and time for scanning images. This innovation also increased the quality of the image and even today in the 20th century it is widely used and many developing countries are using this as a primary source of neuroimaging and in medical fields.

2.1.2 MRI

As the neuroimaging process and techniques were evolving they all used nuclear radiation but the discovery of Magnetic Resonance Imaging (MRI) did not involve nuclear radiation. MRI is based on nuclear magnetic resonance as defined by name (Edelman). MRI was developed by Felix Bloch and Edward Purcell which was based on nuclear magnetic resonance and Damadian is known for inventing the MRI machine in 1992 (“Felix Bloch | American Physicist | Britannica”). Later on, after 25 years a physician from the United States presented the idea that modality in the image could differentiate disease tissue, for example, tumours from normal tissue. Following this idea, Herman Carr published this theoretical idea based on MRI at Harvard University. Inspired by this idea, Paul Lauterbur, used this to come up with a way of creating 2-D and 3-D MRI images. (Abdallah and Alqahtani, 2019). The first MRI scan was done on a mouse. After a year a new algorithm was developed by Peter Mansfield which assisted by reducing the time taken for image generating and their quality. Finally, after a long wait in 1977, Damadian, Minkoff and Goldsmith conducted the first-ever human MRI scan (“Timeline of MRI”). Since then MRI has become the widely used technique or process for neuroimaging.

2.2 Tools used in neuroimaging.

As the neuroimaging process has evolved gradually from one simple technique like “Human Circulation Balance” to the Magnetic Resonance Imaging (MRI) it has been attracting many scientists and people from the medical field. With the curiosity about neuroimaging technology and process many investigations has been done which involves various part of the brain. In ancient times studies and experiments were only done in animal models, electrophysiological measures, examination of post-mortem and observation of the patients with trauma or neurological disorder. Because of this, there were few restrictions on researching neurophysiology.

2.2.1 Limitations for a scientist to use neuroimaging tools.

Now neuroimaging has become popular and it is utilized by large numbers of scientists. However, since scientists are not familiar with programming languages and command-line interfaces and they are not ready to learn to code and create a user interface or other functions for the process. They cannot do anything because they need to have knowledge of a programming language to take advantage of neuroimaging tools.

From selecting an image to viewing the 2D and 3D structure of an object after the process is done and even running the process alone they need to have a great knowledge of programming languages and operating system as they run quite different in each OS.

2.2.2 Solutions for using neuroimaging tools.

Neuroimaging tools were very difficult to access and needed a specific operating system to support the tools and powerful machines to run the process which was time-consuming and the images were only viewed in black and white. Now, various neuroimaging tools can show images in colour. It is very difficult to choose the most suitable neuroimaging tools to view the structure and function of the nervous system of the human brain as they still include a command-line interface for running the process and seeing what is happing and how much the process is done.

The solution for all these limitations can be a Graphical User Interface that simply allows the user to click the button and select the process they want to execute. For instance, a normal tool has more than 50 processes that need to be done for image processing which could take a long time even to study the process and execute it. So making a GUI that allows the user to view the process as they are running, and process images using simple buttons would remove all the problems and can contribute to modern neuroimaging tools.

2.3 Comparison of neuroimaging tools.

There are few neuroimaging tools available with their own uniqueness and features. Before creating a GUI for a neuroimaging tool, a comparison between these tools is needed because it will give me insight into what is included in each tool and which neuroimaging tool will be the best to make a GUI in terms of time efficiency and according to their popularity so that it could be beneficial for many scientist and doctors who wants to use the neuroimaging tools.

According to (PMCID: PMC4795522) 30 neuroimaging tools are listed. Below is the list of 30 neuroimaging tools with their cons and pros comparison.

**•3D slicer**

**Pros:** It is provided with a graphical user interface that can interact with data and it isinterfaced with multiple toolkits.

**Cons:** As this software is under development it does not have functionality.

**•AFNI**

**Pros:** It has several facilities which are provided for specific types of statistical analysis ofmultiple 3D datasets.

**Cons:** It is only available in UNIX operating system.

**•AMIDE**

**Pros:** It has a non-orthogonal interface

**Cons:** It does not have features like factor analysis and cardiac polar maps.

**•AMIRA**

**Pros:** Many functions are available like; surface reconstruction and raw data processing.

**Cons:** It is available in Linux and Windows but not in Unix.

**•BioImage Suite**

**Pros:** It has various tools that are suitable for users.

**Cons:** Although it has lots of tools that are suitable for users, to run this users need to installother packages

**•BrainVoyager**

**Pros:** It is very useful as this software provides accurate brain segmentation in a decentamount of time.

**Cons:** It is only available in Windows.

**•CAMINO**

**Pros:** It is flexible.

**Cons:** Not available in Linux and Windows

**•CONN**

**Pros:** It has a graphical user interface

**Cons:** It is based on MATLAB.

**•DataViewer3D**

**Pros:** It provides integrate results from various analysis packages.

**Cons:** It is only used for visualisation.

**•Explore DTI**

**Pros:** It accepts the output from different tracking packages

**Cons:** It has a complex user interface

**•FreeSurfer**

**Pros:** This software includes packages for various functions

**Cons:** It is slow when used in windows with VirtualBox

**•FSL**

**Pros:** It has Graphical User Interface and can be interacted with the command line

**Cons:** For this, we have to use Windows with VirtualBox

**•ISAS**

**Pros:** It has a specific function mainly for localising the region of seizure.

**Cons:** It has a limited number of functions.

**•LIPSIA**

**Pros:** We can implement fast in this software.

**Cons:** It is not available in Linux and Windows.

**•LONI PIPEline**

**Pros**: It can automatically parallelise data and can convert data format automatically.

**Cons:** Results are inaccurate and biased.

**•Mango**

**Pros:** It has a Graphical user interface

**Cons:** Results are inaccurate

**•MIPAV**

**Pros:** It allows quantitative analysis of images

**Cons:** Implementation is very slow:

**•MRIcro**

**Pros:** It has excellent functions to view images and exporting brain images

**Cons:** The user interface lacks information and it is out of date

**•MRtrix**

**Pros:** It is appropriate for defusing weighted data

**Cons:** The user interface lacks information and it is out of date.

**•MRVision**

**Pros:** It is very useful for creating a simple uncluttered layout.

**Cons:** The user interface is out of date.

**•NITRC**

**Pros:** It provides an environment for cloud computing

**Cons:** It is very slow to load and process.

**•NeuroLens**

**Pros:** Image processing is done very fast and it is flexible.

**Cons:** It provides inaccurate results.

**•Olea Medical**

**Pros:** It is su**i**table for neuro and breast

**Cons:** The user interface is very complex.

**•PyMVPA**

**Pros:** It can handle large dataset

**Cons:** It has a very slow implementation

**•Shanoir**

**Pros:** To ease the completion of metadata it provides user-friendly web access.

**Cons:** The data are very few and it lacks needed information on the website.

**•AIR**

**Pros:** It has automated registration of 3D and 2D images

**Cons:** It is very complicated to install and time-consuming to implement.

**•SDM**

**Pros:** It outputs unbiased results

**Cons:** The output produced is inaccurate.

**•SPM**

**Pros:** It is mostly designed for fMRI, PET, EEG analysis

**Cons:** It is based on MATLAB.

**•TORTOISE**

**Pros:** It is very flexible.

**Cons:** It has an outdated user interface.

Most of the neuroimaging tools listed above have their own strengths and weaknesses. However, to choose the perfect software for Making GUI from listed software, FreeSurfer seems to be appropriate to make GUI as it is used to analyse the human brain and provides different functions like; Skull stripping, B1 bias field correction, Reconstruction and Nonlinear registration of the cortical surface and statistical analysis(Fischl, 2012).

In addition, FreeSurfer was famous soon after its release and it is widely used by scientists and physicists. This is why it will be best to use FreeSurfer to make GUI.

2.4 Freesurfer and its functions.

Freesurfer is a software suite written in C++ which has a collection of 79 functions that are bundled together for the analysis of neuroimaging. FreeSurfer contains almost everything that is needed for analysing the human brain. In this software editing and visualisation tools are included which provide high-quality images and visual interaction to the users. It is an open sources package available in Windows, Linux and Unix. (Fischl, 2012)

Analysis of key features of the human brain is done by FreeSurfer. Some of the key features involved in FreeSurfer are; Volumetric Segmentation, Segmentation of hippocampal

subfields, Inter-subject alignment based on cortical folding patterns, Parcellation of cortical using diffusion MRI, Inter-subject alignment based on cortical folding patterns, Parcellation of cortical folding patterns, Estimation of architectonic boundaries, Mapping of the thickness of cortical grey matters and the construction of surface models of the human cerebral cortex. The idea of “Construction of cortical surface models” was an inspiration that made the development of FreeSurfer possible.

David Van Essen and Eric Schwartz worked on the “cortical cartography” with the help of Heather Drury, they came up with the algorithm which became the foundation for the CARET package. CARET package is now very famous for creating a predictive model in machine learning. This package contains various tools which are used for data splitting, pre-processing and feature selection.

After the development of the CARET package, the computational flattening algorithm was developed by Erick, who was the first person to understand the importance of surfaced-based analysis and this helped him to develop the algorithm which is the basis of FreeSurfer and used in features like “regularizing spherical transformation and registration”.

FreeSurfer consist of many features and the most amazing thing about this software is that there is a command to process every feature separately or the user can do the process with a single command. This gives scientists and medical professionals flexibility in specifying the process they require.

2.4.1 FreeSurfer Limitation.

FreeSurfer has tools for both visualisation and software editing but to process the images, the users need to have a knowledge of the command-line terminal and setting up is quite difficult if the user doesn’t have the necessary knowledge.

FreeSurfer is headless software which means it doesn’t have a Graphical User Interface. To use the FreeSurfer, Firstly, the user needs to set up the environment by giving the command to the command-line interface and this has to be done every time when using FreeSurfer. The commands for setting the FreeSurfer to the environment are provided in the guide for FreeSurfer installation in the webpage. After the environment is set up and FreeSurfer is running, users need to pass the command according to their needs. For example, if they want to view the image then they need to pass “ freesurfer -v imagename.nii” command to the command-line interface and then the visualisation tool will open. There are lots of commands one needs to learn before actually using the tools and features available in the FreeSurfer.

2.4.2 Solution

Therefore, creating a GUI to simplify the process by simply clicking the buttons and conducting the process will be beneficial for Scientist and users who wants to use FreeSurfer. This GUI can save the hassle of looking into the FreeSurfer Wiki on how to use the commands and spending hours figuring out how to use it.

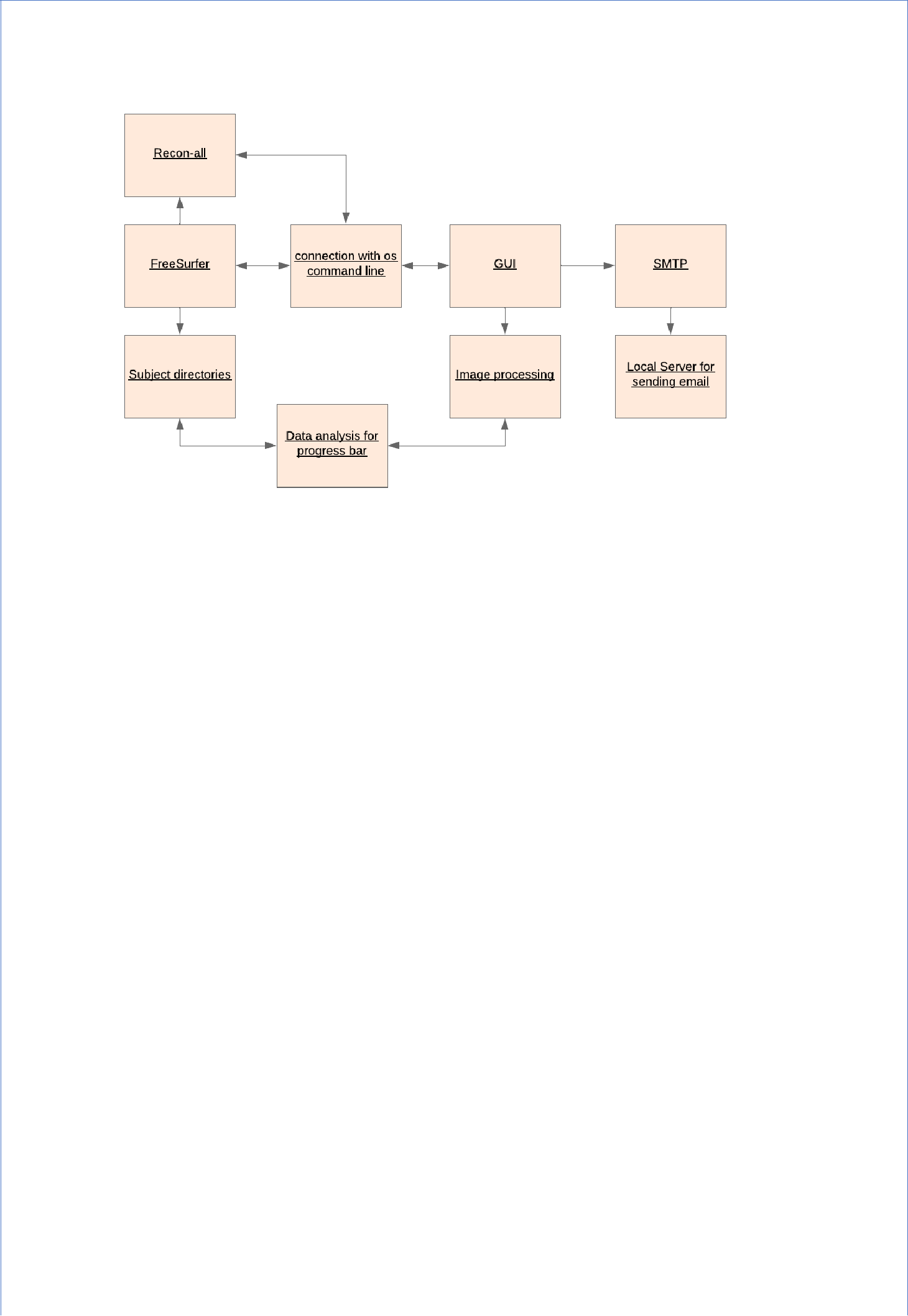
Secondly, after the process command is passed to the command-line interface there is no indication of how the process is going. The users cannot see what is happening because it will output lots of lines that are impossible to understand. So making a GUI that tells users how much process is done and how much more process is still needed to be completed will help them to track the process and analyse the process that is done even if the process couldn’t complete.

Lastly, image processing takes hours. On average it takes 6 to 8 hours to complete the process depending on the machine that the user is using. After the process command is passed the user will not have any idea when it will be completed and they have to wait till the process is completed and check the screen from time to time. Therefore, GUI that sends an email to the user after the process is completed will save their time as they don’t have to look at the screen to see when it will be completed. Users can simply run the process and do other work they need to do in the meantime and after the process is completed it will automatically send a message to the user saying the process is completed.

3. Methodology

A neuroimaging software suite with all the required features needed for human brain analysis and visualisation, FreeSurfer provides extraordinary tools for users. However, to use the tools required users must have a knowledge of the command-line interface. Users who don’t know how to use the interface must learn all the commands provided by FreeSurfer to process, visualise and analyse the image. It can be time consuming and frustrating for those who are new to programming. When the programming is running FreeSurfer does not show the progress of the process and it is quite challenging to set up the environment without proper guidance. The GUI that I have created will overcome all the challenges that users can come across.

Before moving forward to the methods and techniques used in creating GUI. Here is the simple diagram which shows the interactivity of the GUI with users and FreeSurfer.



*Figure 1: FreeSurfer GUI interaction with the OS command line and FreeSurfer*

Figure 1 shows the background process of how the GUI will function. Firstly, when the user installs the FreeSurfer software, an environmental setup should be done and to do that GUI will pass the command to the OS command line and it will be setup up. After that, to select the images and folder the user wants to process GUI will ask for the folder and image name and fetch the path from the OS.

Secondly, when the environment is set up and the user provides an image and folder, GUI will run the “Recon-all” process with the help of the OS command-line interface. As the process is running it will access the subject directories of the FreeSurfer and analysis how much process is done and show the output to the user.

In the end, after the process is complete it will use the SMTP server which currently is local to send the email to the user and end the program.

FreeSurfer GUI is developed using python. I have used the Tkinter package to create the user interface. Tkinter is a standard Python interface to the Tcl/Tk graphical user interface toolkit. Python is the most effective language for GUI programming and a high-level language with a large standard library it helps to create GUI and also connects the GUI with a command-line interface and FreeSurfer.

There are lots of methodologies used when creating a FreeSurfer GUI. From setting up the environment to sending an email to the user when the process is completed. I have classified these methods into 4 categories;

1. GUI connection with Command Line Interface
2. Command implementation using GUI
3. Data logging and analysis
4. Simple Mail Transfer Protocol.

3.1 GUI connection with command-line interface.

The first thing that the GUI should be able to do is set up the FreeSurfer environment. Environment variables are a set of variables (which are not in order) in an area of memory to begin each process. When a process creates a new process, the child’s environment is copied from the parent’s environment. The result from this will have a new process with an environment that is similar to its parent but it will only be a copy. [XXX]. This plays a significant role when the program is using the bash script because to start a new process from the bash script the process needs to inherit the program script environment.

FreeSurfer is developed in C++ and uses the bash script to run the program. So whenever running the program environment variables should be called. FreeSurfer has provided code that sets the environment variables (figure 3). However, when using Python for GUI to run the command, it was not possible to use that command because there are two different commands first to export and the second to the source.

Using the **OS** library in Python it was possible to execute only one command but in order to create an environment variable child variable should be able to copy the parent environment and this needed both commands to run without changing the environment. So to this, I created a function where both commands run at the same time without changing the environment (figure 2)

1. **def update\_env**(script\_path):
2. **if** "--child" **in** sys.argv:# executed in the child environment
3. pprint(dict(os.environ))
4. **else**:
5. python, script = quote(sys.executable), quote(sys.argv[**0**])
6. os.execl("/bin/bash", "/bin/bash", "-c", "source %s; %s %s --
7. child" %
8. (script\_path, python, script))

*Figure 2: Code for setting up the environment variable in GUI*

1. export FREESURFER\_HOME=$HOME/freesurfer
2. source $FREESURFER\_HOME/SetUpFreeSurfer.sh

*Figure 3: Code provided by FreeSurfer to set up the environment variable.*

The libraries that I have used to create an Environment Variable are; **OS** and **sys**. OS library allows the program to interact with the operating system and this is used to execute the function that I have created and run the commands provided by the FreeSurfer. System-specific parameters (sys) library allows variables in the program to be manipulated in different parts of the Python runtime environment. [XXX] this helped me to interpret with an operating system to gain access to the variable and set up the environment of the program same so that the child can copy the process.

To make sure if the bash file passed to the function for creating an environment is working, I created a function called “openfile” which allows the user to open a bash file. After selecting the file, the file path is called in the Environment function and if the bash file is working properly then the GUI will prompt a window saying FreeSurfer is set up to the environment.

3.2 Implementation of GUI commands with buttons.

The first process was to connect the GUI with FreeSurfer and it was done using the environment variable. The next process is to process the MRI images. The image processing that includes all the analysis of the human brain is **Recon-all**. For this, the user needs to have an image they want to process and select the folder name where they want to store the processed image results. To do this process without GUI the user needs to type commands. It is quite challenging because they need to lookup for commands to execute in FreeSurfer Wiki.

To select the folder, in GUI there is a **text box** and a button named “**select this folder”**. The user needs to enter the folder name where they want all the output of the processed image. After that, there is a button named “**select image and Recon-all**”. On click, this function prompts the file dialogue and asks the user to select the image. After they select the image the recon-all process will begin.

3.2 Functions for selecting the folder and image and recon-all.

For the recon-all to process, I created three different functions. Image function to select image, folder function to create a folder and do\_recon function to run the function.

**Image function:**

Figure 4 shows the function for selecting MRI images for processing. Firstly, it opens the file dialogue and asks to select an image. It is done by importing the extinction of the Tkinter library called “filedialog”. When the image is selected it will get the path of the image but to run the recon-all we need only the name of the image currently located in the GUI program directory. So to do this I used “**os.path.split**” to split the image name from the path and stored it into the variable named img\_name. I need to use the image name later when doing recon-all so I return the img\_name.

1 **def** **image**():

1. file = filedialog.askopenfilename()
2. os.path.split(file)
3. img\_name = os.path.split(file)[**1**]
4. # print(img\_name)
5. **return** img\_name

*Figure 4: Function for selecting an image.*

**Folder Function:**

Figure 5 shows the function for selecting a folder to store the output of the processed image and it is essential for running the “**recon-all**” command. Firstly, a text box is created using Tkinter and the parameters are set to (1.0, and “end-1c”) what this means is it will get all the strings passed to the text box. After getting the string from the text box it stores it in the file variable and returns it so that it can be used later.

1 **def** **folder**():

1. file = textBox.get(**1.0**, "end-1c")
2. **print**(file)
3. **return** file

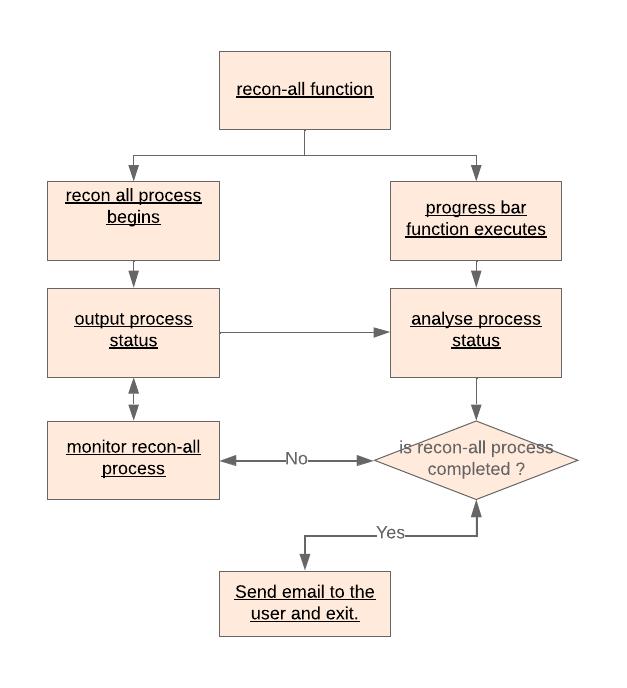
*Figure 5: Function for a folder name.*

**Limitation of running the recon-all command in GUI.**

The recon-all process takes roughly around 6 to 8 hours to complete. The process can be faster or slower depending on the user’s machine. Running this long process will not allow the program to run other functions and we need to wait for the function to complete. The GUI

should be able to show the progress bar when the process is running and to overcome this I came up with a different methodology.

The method is to run two programs simultaneously. So that when the recon-all function is executed it will execute the function for data logging and analysis automatically and run both at the same time. This will be beneficial because when the recon-all process completes another function for the progress bar will also be complete. Figure 6 shows the workflow of how two program works simultaneously.



*Figure 6: Two programs working simultaneously.*

3.3 Recon-all Function:

Figure 7 shows the function for running the process “recon-all”. As mentioned earlier to run the command recon-all, folder name, and image name are required. Firstly, a variable called “cd” is created to store the command. In this variable, I have called both the folder and image function to get their name.

Secondly, to overcome the limitations of running the recon-all command in GUI I created a new file with the **f** variable and used the **truncate** function to delete all the data from the file if it is previously created and used. After that, I wrote all the commands used for running the recon-all command and passed the function that is used for setting the environment because

the file will be new which means all the environment variables I created earlier will be erased and I need to create it again for it to work.

After all the commands are written in the new file, I passed the cd variable and used it for executing the process. “os.system(‘” + cd + “’)” executes the command and I run the program using the subprocess library from the python.

Finally, when the new program runs the program sleeps for 25 seconds. This is done because soon after the new program starts it takes about 10-15 seconds to execute and there will be no process to view. After 25 seconds progress bar function will run which is named **path()** in the code. The path function is used for data analysis and logging to show progress.

1. **def do\_recon**():
2. # cd = ("recon-all -i " + image + " -s
3. cd = ("recon-all -s " + folder() + " -i
4. f = open("reconall.py", "a")
5. f.truncate(**0**)
6. f.write("import subprocess **\n**")
7. f.write("import sys **\n**")
8. f.write("import os **\n**")
9. f.write("from main import update\_env **\n**
10. # f.write("sd =" + filepath() + "\n")
11. f.write("update\_env('bash.sh')**\n**")
12. f.write("os.system('" + cd + "')")

13

1. f.close()
2. subprocess.run("python3 reconall.py &",
3. time.sleep(**25**)
4. **print**(cd)
5. path()
6. **return** cd

* + folder()
  + + image()

")

shell=True)

* " -all")
* " -all")

*Figure 7: recon-all function*

3.4 Data logging and analysis:

To show the progress of the user data analysis is conducted. In the FreeSurfer when recon-all is executed and the image processing starts. The recon-all is quite a long process and users don’t know how much of the process is done and how much is left to do. The process status is stored under the folder name which is chosen by the user in the beginning.

Altogether there are 79 different features that FreeSurfer execute when the recon-all process is started and they are: MotionCor, Talairach Monm, Talairach Failure Detection, Nu Intensity Correction, Intensity Normalization, Skull Stripping, EM registration, CA normalize, CA Reg, SubCort, CC seg, Merg Aseg, Intensity Normalization2, Mask BFS, WM Segmentation, Fill, Tessellate lh, Tessellate rh, Smmoth1 lh, Smooth 1rh, Inflation1 lh,

Inflation1 rh, Qsphere lh, Qspere rh, Fix Topology lh, Fix Topology rh, Smooth2 lh, Smooth rh, Inflation2 lh, Inflation2 rh, Curv .H and .K lh, Curv .H and .K rh, Sphere lh, Sphere rh, Surf Reg lh, Surf Reg rh, Jacobian white lh, Jacobian white rh, AvgCurv lh, AvgCurv rh, Cortical Parc lh, Cortical Parc rh, white curv lh, white curv rh, white area lh, white area rh, pial curv lh, pial curv rh, pial area lh, pial area rh, thickness lh, thickness rh, area and vertex vol lh, area and vertex vol rh, Curvature Stats lh, Curvature Stats rh, Cortical ribbon mask, Cortical Parc 2 lh, Cortical Parc 2 rh, Cortical Parc 3 lh, Cortical Parc 3 rh, WM/GM Contrast lh, WM/GM Contrast rh, Relabel Hypointensities, APas-to-ASeg, APas-to-ASeg aarc, AParc-to-ASeg aparc.a2009s, AParc-to-ASeg aparc.DKTatlas, WMParc, Parcellation Stats lh, Parcellation Stats rh, Parcellation Stats 2 lh, Parcellation Stats 2 rh, Parcellation Stats 3 lh, Parcellation Stats 3rh, ASeg Stats, BA\_exvivo Labels lh and BA\_exvivo Labels rh.

When the function for the progress bar executes in the recon-all function (do\_recon), it looks for the process that has been done in the status log of the FreeSurfer and gets the last line from the log file. After getting the line the function will determine how many processes are done and how much more processes are there to run.

3.5 Function for the progress bar.

Figure 8 shows the function for the progress bar which analyse the data from the log file and shows the progress of the recon-all process. Firstly, variables are defined for email, message and recipient which will be described in the SMTP methodology. To access the log file generated by the FreeSurfer, GUI needs to know the path to the folder so to access the path to the folder I used **“os.path.expanduser(~)”** function which will simply get the path of the home directory. The next variable is **filepath** which assembles all the string that makes a full path. For instance, the **folder()** function returns the name of the folder that the user used at the beginning of the process.

After getting the file path, the function will look into the path and search for the file called recon-all-status.log. The name of the file will be the same in all the projects. If the program exists then it will run the program and check the output in that log file.

As mentioned earlier FreeSurfer has 79 different processes and when each process completes, the process name will be automatically written into the log file. When the program reads the output of the process that has been done it will show the user how many processes have been completed out of 79 processes. Reading the lines was done using the **readlines** function in python which reads the line in the designated file and helps the user to analyse it.

Each process has a name and its name is linked with a number. They are always performed in the same order so it was quite easy to know which process comes first. This gave me an idea of how to show progress in the progress bar.

The progress bar value is 100 and there are 79 processes. 100/79 roughly gives me around 1.27 and I gave a 1.27 value of the progress bar to each function. So when one process completes and the function reads the name it will have its number and it will multiply with the number 1.27 and give the number calculated to the progress bar to show increment. For example, in line 17 the progress value is 1.27 because this is the first process so the value is not multiplied. However, in line 23, the progress bar value is multiplied by 2 because this is the second process.

To make this work I created around 80 if and else if function, in the Figure below there is only a few but full code will be available in the appendix of this report. The process of the progress bar itself is automated and it loads every 2 minutes. By any chance, if the user wants to update the bar then there is a button to update the progress.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

|  |  |  |
| --- | --- | --- |
| **def path**(): | | |
| e | = | "freesurferemail@gmail.com" |
| m | = | "Your process has been completed sucessfully" |
| r | = | recipent() |

cd = os.path.expanduser('~')

# print(cd)

filepath = (cd + "/freesurfer/subjects/" + folder() + "/scripts/recon-all-status.log")

**with** open(filepath) **as** f:

line = subprocess.check\_output(['tail', '-1', filepath])

last\_line = f.readlines()[-**1**]

**if** 'MonitorCor' **in** last\_line:

l1.config(text=line)

l2.config(text="MonitorCor is in process **\n** 1/79 process is

done")

progress['value'] = **1.27**

**elif** 'Talairach' **in** last\_line:

l1.config(text=line)

l2.config(text="Talairach is in process **\n** 2/79 proceess is

done")

progress['value'] = (**1.27** \* **2**)

.

.

.

.

.

.

1. **elif** 'finished without error' **in** last\_line:
2. l1.config(text=line)
3. l2.config(text="recon all is finished without error/n
4. sending email to the user")
5. l3.config(**print**("recon-all is finishing sending email to
6. user"))
7. progress['value'] = **100**
8. sendmail(e, r, m)
9. exit()
10. **if** 'without error' **in** last\_line:

|  |  |  |  |
| --- | --- | --- | --- |
| 35 | progress['value'] | = | **100** |

1. **elif** 'finished with error' **in** last\_line:
2. l1.config(text='Error Occured Please Try Again')
3. Timer(**120**, path).start()

*Figure 8: progress bar function*

3.6 Simple Mail Transfer Protocol function.

Figure 9 shows the function for the Simple Mail Transfer Protocol that will be used to send mail to the user after the process is completed. For this function to work, three arguments should be passed. Email, Recipient Email and Message to send. I created a variable named, **e**, **r** and **m** in the function for the progress bar and called the function there so that when theprogress is 100 or in other words when the recon-all process is completed it will call the SMTP function and send an email to the user.

Next, to use this function I created a Gmail account for the FreeSurfer as a sender and to do

so I imported smtplib. smtplib allows users to send mail with a simple mail transfer protocol

to any computer or device with the internet. The function

**smtplib.SMTP\_SSL(“smtp.gmail.com”, 465)** set up the server and port number for the

email transmission.

Secondly, the **server. the login** function is used to log in to the FreeSurfer account so that the program itself could log in and send an email to the user. In the default setting of Gmail, untrusted software cannot access the email unless the setting is changed. So for my program to log in to the account, I allowed less secured software to log in. After this, I used the **server. SendMail** function which will use the arguments for this function to login to theFreeSurfer account, select the recipient email and the message the user gets.

Finally, after sending the message the function quits, all the processes will be completed and the program stops.

1 **def** **sendmail**(email, recipent, message):

1. server = smtplib.SMTP\_SSL("smtp.gmail.com", **465**)
2. server.login("freesurferemail@gmail.com", "freesurfer123!")
3. server.sendmail(email,

|  |  |
| --- | --- |
| 5 | recipent, |
| 6 | message |
| 7 | ) |
| 8 | server.quit() |

*Figure 9: SMTP function for sending mail to the user*

3.7 GUI Implementation

Python is widely known for its simple API that allows developers to create user interfaces using different Python libraries and applications. To implement the GUI in python I used Tkinter. It is a Python package for the graphical user interface. There are various options for creating GUI but Tkinter is the most used package and it is also very fast and easy to create GUI.

To create a GUI in Python using Tkinter, I imported the modules of Tkinter. It has multiple widgets that can be used for the implementation of complex functions and designing the window as well to make it look standard.

Figure 10 shows the code that has been used to make all the buttons and text boxes in the GUI.

1 greetings = Label(root, text="Welcome to FreeSurfer GUI")

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | # this is button for opening a file for setting up freesurfer | | |
| 3 b = tk.Button(root, text="select file for set up", borderwidth=**3**, | | | |
| 4 relief="ridge", command=openfile) | |  |  |
| 5 | # this is for getting folder name | for recon | all |
| 6 textBox = Text(root, height=**1.5**, width=**30**) | | |  |
| 7 | #this is the button for selecting | a folder |  |
| 8 buttonCommit = tk.Button(root, text="select | | | this folder", |
| 9 borderwidth=**3**, relief="ridge", command=folder) | | | |
| 10 | #toolbox to write recipient email |  |  |
| 11 textBox1 = Text(root, height=**1.5**, | | width=**30**) |  |
| 12 | #this button is fr selecting recipent email | |  |

13 buttonCommit1 = tk.Button(root, text="select recipent", borderwidth=**3**,

14 relief="ridge", command=recipent)

15 #this button is for recon-all

16 recon = tk.Button(root, text="select image and Recon all",

1. borderwidth=**3**, relief="ridge", command=do\_recon)
2. #this is for progress bar

20 progress = Progressbar(root, orient=HORIZONTAL, length=**400**,

1. mode='determinate')
2. #this is a button to show the progress.
3. showProgress = tk.Button(root, text="Show Progress", borderwidth=**3**,
4. relief="ridge", command=path)
5. #these are the labels to show the text in the gui

*Figure 10: Button implementation.*

4. Testing and Validation

Testing plays an important role when developing software or program. Testing a program can show the areas of weakness and the nature of the product created. When testing, errors can be found which is a way of improving the quality of the program. After the research on the neuroimaging tools and features, the way of solving the problem is identified and with the methodology the design and implementation of the software became effortless. Just like that when the methods are implemented and a program is created, testing is needed to see if the program has any errors and to identify its quality. Creating software or developing a program is a very systematic process that not only helps the developer but also the users to know more about the program.

FreeSurfer GUI that I have developed to overcome the hurdles of scientists and users by saving their efforts to learn programming languages and time that could consume when running vast features in the FreeSurfer. FreeSurfer GUI is specially designed with the concept of providing a user-friendly graphical interface that gives clarity on running the process and its progression.

Graphical User Interface, which means the users are pressing buttons instead of passing the command into the command-line interface. Creating a GUI means creating a button and textbox which works correctly without any disruption and does the right process. For the testing of the FreeSurfer GUI, I have conducted tests on seven different buttons and two text

boxes independently and after knowing that every button works I have tested the GUI itself from start to the end. The seven different tests that I have done are:

1. Environment set up testing
2. Folder name validation and testing
3. Image selection and testing
4. Recon-all testing
5. Progress bar testing
6. SMTP testing
7. GUI grids and clarity testing.

4.1 Environment set up test:

To run the FreeSurfer after installation, the user needs to select a bash file that contains a script on setting up the environment and running the FreeSurfer. In GUI there is a button that allows the user to select the button and choose the file.

**Testing process:**

I tested the button by pressing it to see if it opens the file browser where I can browse the file and select the file and run it successfully.

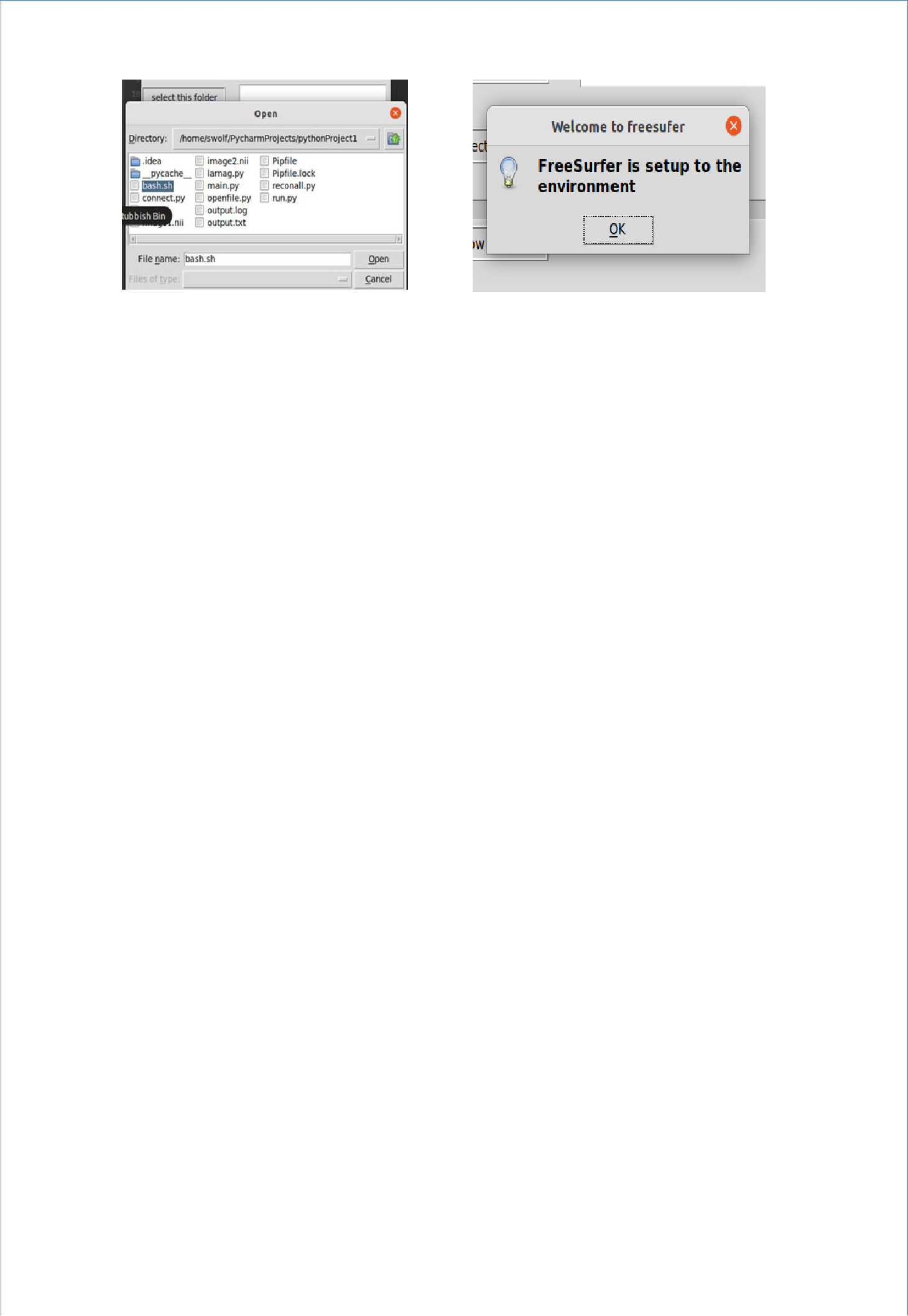
**Expected Outcome:**

I expect the button to work properly by allowing me to choose files and set up the environment.

**Actual Outcome:**

The actual outcome was the same as the expected outcome. It successfully set up the environment after selecting the correct file.

**Evidence:**



*Figure 11 and 12 : Evidence of environment testing.*

4.2 Folder name validation and testing.

The user must provide a folder name to run the recon-all process. In GUI there is a text box that allows the user to enter the name of the folder and a button to get the name of the folder when it is pressed.

**Testing process:**

I tested the text box by typing the folder name and pressing the button to see if it get the folder name or not.

**Expected outcome:**

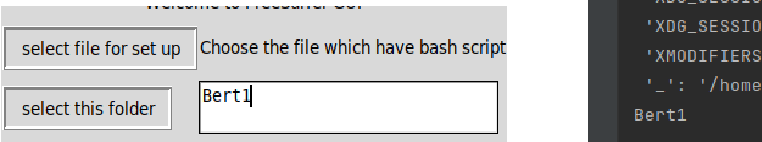
I expect the button to get the folder name that I entered in the text box.

**Actual outcome:**

As expected the button gets the name of the folder I entered in the text box.

**Evidence:**

The first image shows the text box working and the second image shows that the button successfully fetched the data from the text box.



*Figure 13 and 14: Evidence for folder name testing*

4.3 Image selection and testing.

The user needs to select an image to process the neuroimaging analysis and recon-all. In GUI there is a button for selecting an image. It should open the file browser and it should be able to get the name of the image only and not the path of the image.

**Testing process:**

I tested the button by pressing it to see if it opens the file explorer to select the image and get only the name.

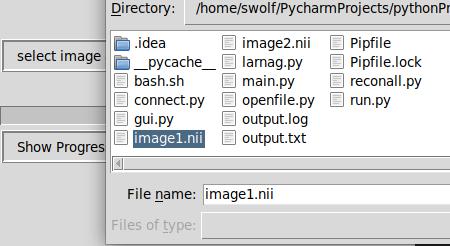
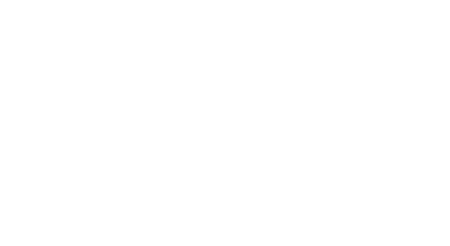
**Expected outcome:**

I expect the button to work properly by allowing the user to select the image and get the name of the image.

**Actual outcome:**

The actual outcome was the same as the expected outcome. It opens up the file explorer for a user to select the image and gets only the name of the image.

**Evidence:**



*Figure 15: Evidence for image selection testing*

4.4 Recon-all testing.

After getting the folder name and image name the recon-all process needs to be implemented for image processing. In GUI there is a button for both selecting an image and doing recon-all by pressing this button the process starts.

**Testing process:**

I tested the button if it works or not by pressing it. It should first ask for the image and as soon as the image is selected it should start the process of image analysis.

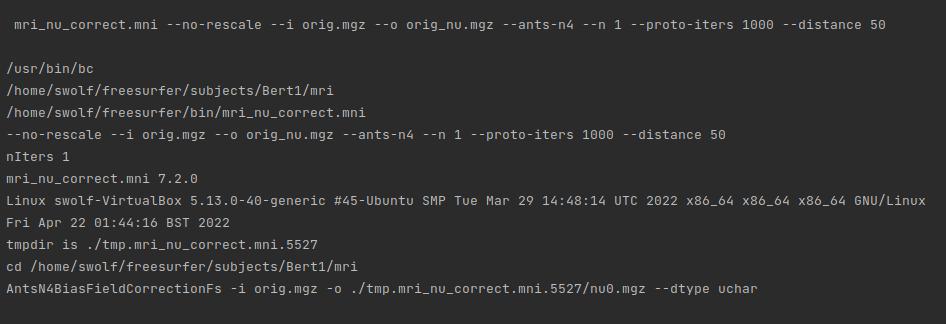
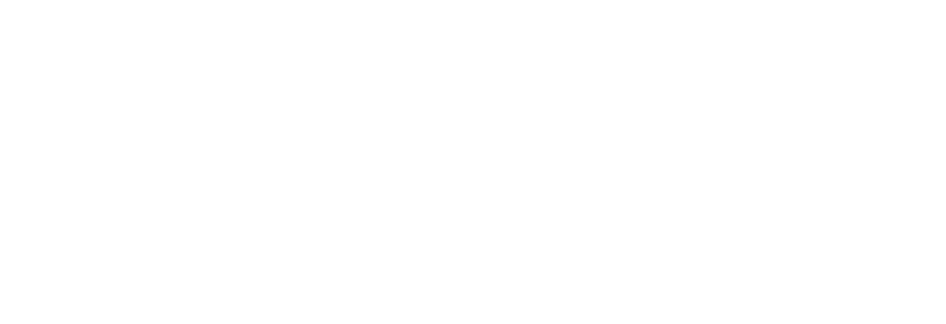
**Expected outcome:**

The testing on image selection is already carried out in the previous testing. So I expect that right after choosing the button it starts the process.

**Actual outcome:**

As expected right after the image is selected the process starts.

**Evidence:**



*Figure16: Evidence for Recon-All Testing*

4.5 Progress bar testing.

The progress will monitor the recon-all process as it is running and show the progress of the process in the progress bar with details. In GUI it should work automatically right after the recon-all process starts.

**Testing process:**

I tested the progress by running the recon-all process to see if it automatically increase the value.

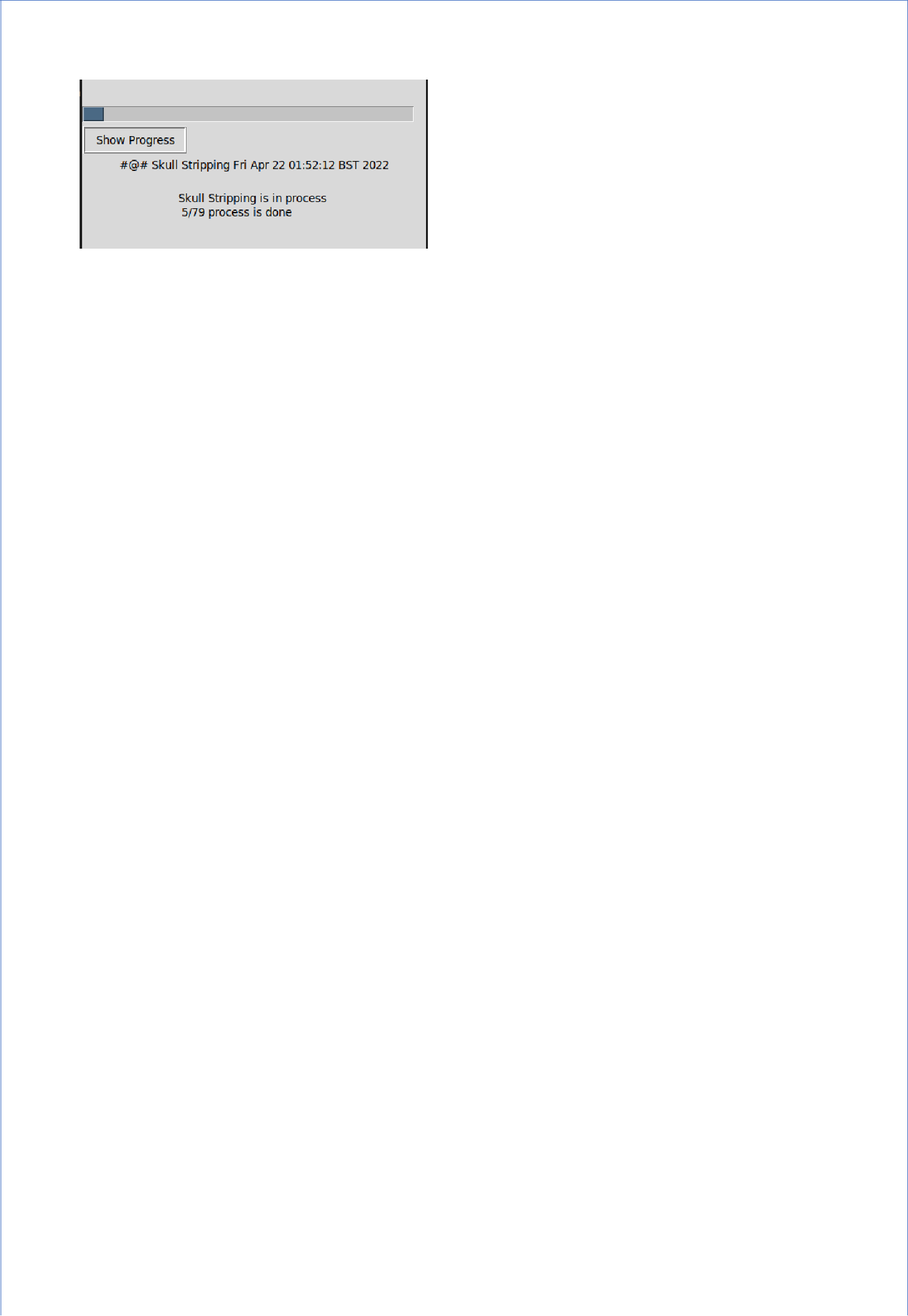
**Expected outcome:**

It should automatically show the progress of the process with the details of how much the process is completed.

**Actual outcome:**

As expected it runs automatically right after the recon-all process starts and gives the detail of the process and the name of the feature it’s implementing.

**Evidence:**



*Figure 17: Evidence of progress bar.*

4.6 SMTP testing.

At the moment the SMTP server is local and uses Gmail to send an email to the user. In GUI when the process is completed and the value of the progress bar reaches 100, it automatically sends the email to the user. After selecting the folder there is a text box and button which uses the same method as selecting the folder.

**Testing process:**

To test the Simple Mail Transfer Protocol I provided my email address and waited until the process is completed to see if I get the email or not.

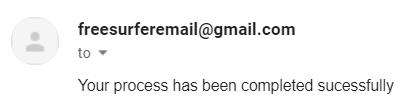
**Expected outcome:**

I should get an email to my personal email address right after the process is completed.

**Actual outcome:**

As expected I received an email with the message “Your process has been completed successfully”.

**Evidence.**



*Figure 18: Evidence for SMTP testing*

5. Results

Building a GUI for complex software like FreeSurfer is quite challenging and sometimes results might not be what we expect them to be. FreeSurfer GUI has various multiple methods but the most unique ones are Environment Variables and Running two programs simultaneously.

The expected outcome of the environment was that it successfully create the parent and child process in the same environment so that the GUI will be able to connect with FreeSurfer. After the implementation of the method, the result was the same as I expected and it is the main reason behind FreeSurfer GUI working.

The second method was to run multiple programs at the same time when they interact with each other for file logging and data analysis. At first, I could only run one command at a time but to see the progress bar another function needed to execute. So to run multiple commands at the same time a method is created where the running program will create another python file and run it so that the program which is running can execute the progress bar command.

The output for this method was the same as expected it creates the file and executes it and when the new file is executed progress bar function is called and they work together until the end of the process.

6. Conclusion and future work

In conclusion, this project has helped me understand the principles and science behind the Graphical User Interface. This project has enhanced my knowledge of Environment Variables and their importance when creating a GUI or even running any simple software. I was unaware that when starting the new process or the software an environment needs to be set up and only after that we can run the software. In the beginning, it was quite challenging for me to set up the environment for the FreeSurfer and connect my GUI. However with a full understanding of environment variables, parents and child processes I was able to create a function for setting up the environment which I think is a great achievement when I was doing this project. The new perspective and clarity have been opened up and this has changed the way I think about the software.

Many scientists/physicists who are doing research on neuroimaging and want to use the neuroimaging analysing software are unable to do it because they do not have the required knowledge of programming language and this was a huge barrier for them. To overcome this

I did research on the history of the neuroimaging techniques and their evolution over time to know how the image analysis was done throughout the centuries. During the time of MRI, computers were becoming popular around the world. To see if any neuroimaging tools use GUI for image processing, I researched 30 different neuroimaging tools and searched a software suitable for creating a Graphical User Interface to solve my problem. When conducting research I gained knowledge of different neuroimaging processes, techniques and tools that are widely well known at this time. After doing the research and knowing how I can solve the problem, I created methodologies to make a GUI which is described in this report. When creating methodologies, I learned many things about software production, how the software functions, and how software is created. Knowing the methodology of the GUI it was quite easy to implement it. During the implementation, I acquired knowledge of python libraries, functions and the Tkinter Package. Most importantly I learned how the software is able to execute multiple commands simultaneously.

FreeSurfer is a software suite and has many features and tools. Image processing is one of the tools that I made a GUI for. For future work, using this GUI as the foundation I would like to create a GUI for image Visualisation. Having a GUI for both Visualisation and Image processing can make the FreeSurfer extremely popular among scientists and people who have an interest in neuroimaging.

Overall, this project was challenging and at the same time, it was quite fun to work on. I managed to resolve all the problems, however, there is still one thing that I would do in the future and this is to be able to select multiple images and process them.

7. Reflection

On reflection, with the understanding gained in completing this report a similar approach to the GUI would be taken in the future if I came across a similar project. Overall the learning experience was exciting and challenging at the same time. There were times when I was stuck in one method and there were times when a new perspective occurred and I was feeling joy doing research to find a solution for my problem.

The **Project Initiation Document(PID)** which was submitted at the beginning of the term was submitted before having the conversation with the supervisor and there was a lot of confusion on how the Final Year Project is structured but after the meeting with my supervisor all the doubt were clear and I knew what I need to do. For Background Research, I

changed it to; History of neuroimaging techniques, Tools used for neuroimaging, FreeSurfer speciality. For analysis/design, I change it to; GUI connection with a command-line interface, Command implementation in GUI, Data logging and analysis, SMTP and GUI implementation.

There have been a few changes to the GUI which were demonstrated in the **Poster.** At first, I thought of creating a tab and under the tab section, there will be options to choose a file and image. However, those are the essential part of the process so I displayed them in the front window so that the user won’t have to look up the tabs for each process. The previous GUI wasn’t user friendly and it was confusing for users. The new GUI has everything upfront and users can easily know what they are supposed to do.

Figure 19 shows the FreeSurfer GUI.



*Figure 19: FreeSurfer GUI*

To compare me before and after the project, I would say this project has given me a clear sense of clarity on what to choose in the future and gave me hope that everything is possible if we do research and keep working hard.

References: