UNIVERSITY OF SRI JAYEWARDENEPURA

Faculty of Science

TRAINING REPORT

Arthur C. Clarke Institute for Modern Technologies

Project name : Determination of Frequency Drift Rate of Solar Radio Bursts using CALLISTO data

From 07/01/2020 to 07/04/2020

Date of submission : 22^{nd} of June 2020

N.M.Samarawickrama

AS2016944

Department of Physics

Declaration

This dissertation describes that this industrial training report is the record of authentic work carried out by me during the period from 07th January 2020 and 07th April 2020, as an intern at Arthur C. Clarke Institute for Modern Technologies under the supervision of Industrial and Academic supervisors.

………………………..

N.M.Samarawickrama

AS2016944

…………………………………

Industrial supervisor

Mr.Janaka Adassuriya

Research Scientist,

Astronomy division,

Arthur C. Clark Institute for Modern Technologies.

………………………………….

Academic supervisor

Mr. C.H. Manatunga

Senior lecturer

Abstract

This report describes about the industrial project that was carried out by me at Arthur C. Clarke Institute for Modern Technologies. I was assigned to the project "Determination of Frequency Drift Rate of Solar Radio Bursts using CALLISTO data" under the division of Astronomy. In order to achieve this task I had to study about new term fits files, E-CALLISTO international network, python and image processing techniques used in OpenCV which comes under python language.

Fits file which included all the details about solar burst were extracted from E-CALLISTO network and the solar burst was identified from it by removing the unnecessary noise. Image processing techniques like gaussianblur, threshold, erode and dilate were used to remove the noise from solar burst. The maximum intensity points of the solar burst were located and using python commands drift rate was calculated. A simple interface was designed by me using tkinter in python to make this process more simpler and easier.

Acknowledgment

It is a great opportunity to thank all the people who are supported me in completing my 3 months of internship period successfully in a meaningful way by sharpening my career effectively.

Firstly, I would like to thank the Department of Physics, specially Prof.A.D.Kumarasinghe the Head of the Department, Dr.S.Jayawardhana the Industrial Training Coordinator of Physics Department for guiding us throughout the program by finding good training establishment places. Further I would like to pay my sincere gratitude for Mr.C.H.Manatunga the Internal Supervisor of the Internship for the guidance and giving us instructions for sharpening our career during the training period.

Not only that I am highly grateful to Astronomy Division of Arthur C. Clarke Institute for Modern Technologies (ACCIMT) for their guidance and support throughout my Internship period, Specially the Director of Astronomy Division Mr.Saraj Gunasekara and the Industrial Training Supervisor Mr.Janaka Adassuriya for the immense support,guidance and dedication to make this internship program a fruitful one. Then I would like to thank the entire stuff of ACCIMT for their daily support throughout my training period.

Finally, I would like to thank all the other technical and non-technical staff members and my other fellow trainees for their cooperation and help given during my training period.

Table of Contents

Table of figures

1. Introduction of the Institute

This gives an introduction about my training establishment, Arthur C. Clarke Institute for Modern Technologies (ACCIMT). Here it describes about ACCIMT & its history, vision and mission, services and the divisions of the Institute.

1.1 Arthur C. Clarke Institute for Modern Technologies

Arthur C. Clarke Institute for Modern Technologies is an institute for research and technology transfer in Sri Lanka. This was found by the renowned British sci-fi creator and designer Sir Arthur C. Clarke. The organization is chiefly centered around directing examination in the fields of electronics,micro-electronics,telecommunications,information technology, space advancements and mechanical technology. It is one of only a handful not many establishments of this sort in Sri Lanka.

The ACCIMT was set up in 1984 by act of parliament, the Arthur C. Clarke Center for Modern Technologies Act, No. 30 of 1984 and re-established in a corporate form in 1994 by the Science and Technology Development Act, No. 11 of 1994. The ACCIMT was chosen as the national point of convergence for space innovation applications, by the United Nations Economic and Social Commission for Asia and the Pacific around the same time 1994.In 1996 a 45 cm Go To Cassegrain reflector telescope was received to this institute as a donation by the Tokyo National Observatory which was the largest optical telescope in Sri Lanka.

Figure 1. 1: The largest telescope in Sri Lanka.

1.2 Vision and Mission

1.2.1 Vision

To be a leading innovation centre for Modern Technologies in the region.

1.2.2 Mission

“To develop, foster and facilitate the domestic base of modern technological capabilities through innovation, R & D, training, industrial services and international collaboration”.

1.3 Services

1.3.1 Research and Development

The information and communications technology, electronics, microelectronics, space technology, Astronomy and robotics are some fields which carry out researches in this institute. The majority of its research is planned for advancing most recent innovation among government and the private area ventures in Sri Lanka.

1.3.2 Consultancy

Their very much experienced and exceptionally gifted specialized staff give consultancy in overseeing and overhauling high tech industrial systems, offer their ability and facilities to the local Industry so as to help overseeing current modern frameworks, for example, microchip based hardware, telecom frameworks, information systems, PC systems and so forth. Further, the establishment offers advance symptomatic and fix benefits in the regions of their mastery.

1.3.3 Training programs

Professional Development programs are conducted for professional, experts and senior managers by ACCIMT. Further more it has additionally propelled electronic workshops and astronomy outreach programs for school students in Sri Lanka. Library facilities are accessible for college understudies and the overall population.

Figure 1. 2:Training program conducted by ACCIMT

1.4 Division of ACCIMT

1.4.1 Communication and Robotics division

The Communication Division is well furnished with present day instrumentation and computer systems, which help in applied research and product improvement related to electronics, broadcast communications and microchip based equipment.

The fields of research that carried out by this division are

- Automation Systems
- Communication Systems
- Rolling Stock Systems
- Unmanned Ground Vehicle
- Unmanned Aerial Vehicle
- Space Technology

1.4.2 Electronic and Microelectronic division

This division does research and development projects, test and estimation administrations, consultancy administrations, equipment recuperation and Continuous Professional Development (CPD) courses for the industry. The division basically centers around industry started R&D activities particularly microcontroller based system structuring, data logging and display systems, use of sensors, simple circuit plan and electrical cable data securing.

1.4.3 Industrial services

The division is for the most part settled to give consultancy support to the electronic business in the regions of Calibration of test and measuring instruments, Performance test and measurement administrations and equipment recuperation services of electronic and electrical lab instruments.

1.4.4 Information Technology

Research, IT Solutions, Technology Transfer and Training Courses are main functions carried out by the Information Technology Division. within the past more emphasis was made towards development of software to satisfy the requirements of clients and therefore the transfer of data concerning IT to general public through courses. The software developed are within the sort of processing of knowledge to hurry up tasks to offer a far better service to the general public and also within the sort of information dissemination through information systems.

1.4.5 Space Application division

Space Applications Division has been conducted the activities within the fields of RS/GIS (Remote Sensing/Geographic Information Systems).

1.4.6 Astronomy

The division is responsible for conducting operation of telescope facility and carryout education programs of observations. Astronomers within the division are working in close collaboration with foreign entities and native universities to hold out basic research in astronomy. additionally, outreach programs for astronomy and space science popularization also are conducting for public and school children.

2. Introduction of the task

This gives an introduction about the project “Determination of Frequency Drift Rate of Solar Radio Bursts using CALLISTO data” which I carried out at Arthur C.Clarke Institute for Modern Technologies and introduction to the techniques I used to do this task.

2.1 Astronomical background

2.1.1 Solar Bursts

A sudden flash of increased brightness on the Sun is known as a solar burst and usually it can be observed near Sun’s surface where a group of sunspots can be observed. In this project I worked on the type ii solar bursts which are very rare and shows a slow drift from high to low frequency.

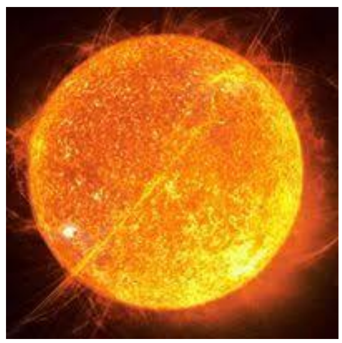

Figure 2. 1: image of a solar burst

2.1.2 Fits file

FITS is the most ordinarily used digital file format in astronomy. The standard meaning of the FITS is Flexible Image Transport System and it consists of multidimensional arrays and 2d tables. This is mostly used for transporting, analyzing, and archiving scientific data files. In this project the fits file contains all the data of a solar burst captured by station including the time(s) and frequency(MHz) range.

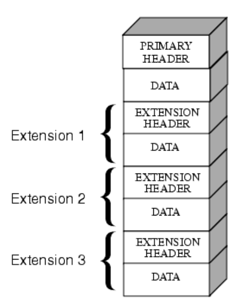

Figure 2. 2: Structure of the fits file.

2.1.3 e-CALLISTO Network

The standard meaning of CALLISTO is Compound Astronomical Low frequency Low cost Instrument for Spectroscopy and Transportable Observatory. This instrument normally operates between the frequency range 45MHz to 870MHz. So e-CALLISTO is an international network carried out by Dr. Christian Monstein which keeps the records of solar radio bursts observed using CALLISTO spectrometers. Stations that placed all over the world use this CALLISTO spectrometer and observe the solar radio spectrum for 24h through all the year. The CALLISTO instrument stored this data as fits files in e-CALLISTO network which can be access by anyone.

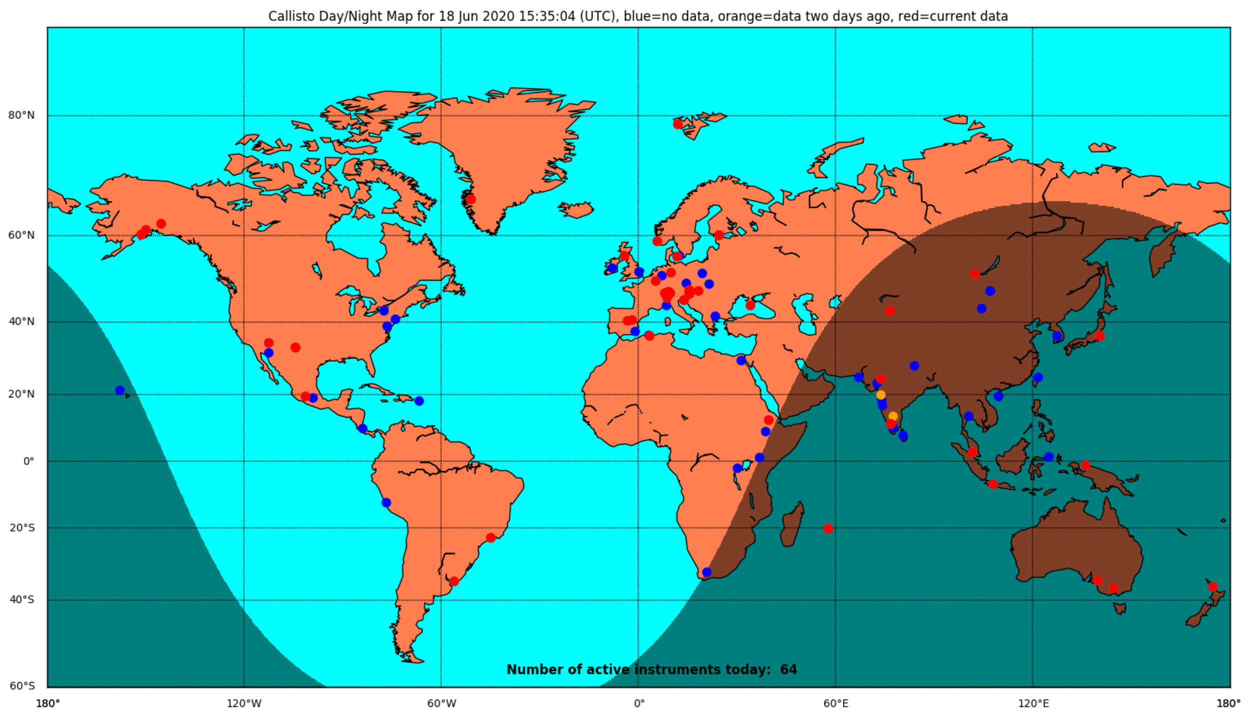

Figure 2. 3: Map of current distribution of CALLISTO instruments.

2.2 Project background

Python language was used as the main language to achieve this task. The main task of the project was to extract fits file from e-CALLISTO network and identify the unnecessary noise that captured in solar burst. The image processing techniques in OpenCV was used to remove the noise from the solar burst and the drift rate of the solar burst was calculated.

2.2.1 Python language

Python is a high level language which allows to do any common programming tasks. This also known as a general purpose language because it can be used for developing GUI applications, software development process, data science and many more. This project was done using the Python version 3.7 and Spyder was used as the integrated development environment (IDE).

There are many libraries in python and matplotlib, astropy, pandas, numpy, PIL, skimage, scipy, sympy, datetime are some libraries imported for this project.

Figure 2. 4: Logo of the python

2.2.2 Image processing techniques – OpenCV

OpenCV is a library which implemented using C++ language to solve the problems with computer vision. Techniques like Gaussianblur, threshold, erode and dilate were used in this project.

Figure 2. 5 : Logo of the openCV

- Gaussianblur – This a low pass filter whish helps to reduce high frequency components of the image. So this technique is used to smoothen or blur the input source image.
- Threshold – In this the pixels of an image is classified according to the given threshold value. If the pixel value is greater than the threshold value usually it sets that pixel in to 255 otherwise it set to 0.
- Erode – Usually this technique is performed on binary images and it used to remove or erode pixels on boundary of the object.
- Dilate – This is the opposite of erode which used to add pixels on boundary of the object.

Figure 2. 6: Original image

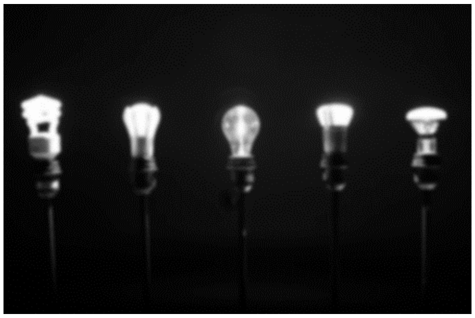

Figure 2. 7 : Gaussianblur image

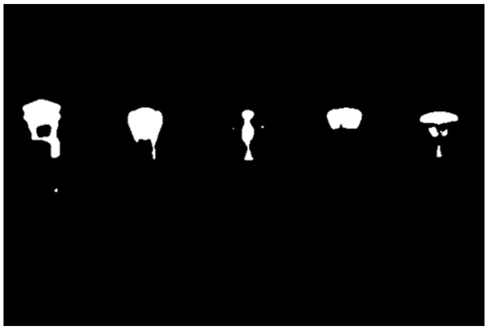

Figure 2. 8: threshold image

Figure 2. 9 : erode and dilate image

2.2.2 Tkinter GUI library

This is a standard GUI library of python. This can be used to develop simple GUI applications. In this project the GUI application was designed using this library.

The following codes represent how to make a tkinter window simply with one widget.

```python
from tkinter import *
root = Tk()                                 # Create the root (base) window
w = Label(root, text="Hello, world!")       # Create a label with words
w.pack()                                    # Put the label into the window
root.mainloop()                             # Start the event loop
```

Figure 2. 10 : Simple code to make a tkinter window

3. Objectives

3.1 Main objective

- How to identify max intensity area from an image.
- Study about the drift rates of different solar radio bursts.

3.2 specific objectives

- Study about different image processing techniques used in python-openCV.
- Learn about curve fitting techniques.
- Learn how to design a simple interface using Tkinter in python.

3.3 Academic outcomes

- Gain knowledge regarding solar bursts and their types.
- Improvement of communication skills.

4. Methodology

In this section it includes the procedure and techniques I used to make this project.

4.1 software development

4.1.1 Requirements

In the beginning there was no clear idea about the Astronomical background and project background. So at first I had to gain the following requirements to achieve this task.

- Studied about solar bursts, fits file and e-CALLISTO network.
- Learned about python commands for fits file handling.
- Learned about image processing techniques.
- Studied about curve fitting techniques.

4.1.2 Implementation

After gaining a bit idea about both Astronomical and project backgrounds I started to work on the project. The following points shows the work I did in step by step.

- At first I simply tried to read a fit files using different softwares like mathematica, matlap and python(spyder) and I studied the details included in it. Then I selected python language to carry out the works further.
- The image is a 2d array in 3600x200 size, So in order to plot the image with both axes frequency and time I had to convert them according to the ranges of both frequency & time axes. Some calculations as shown in below were carried out by me to convert the axes.

  Y axis (frequencys-MHz) = 200/frequency range of the station = y (took the round number)

  If we assume we get y number of times then I took the first y numbers of the image array and got the average. Again I got next y numbers and got the average likewise I repeat it through the 2d array of the image in order to get the y axis values.

  X axis (time-s) = 3600/time range=x (took the round number)

  As in y axis here I did the same and took the values of the x axis.

  Then I plotted the image with axes.

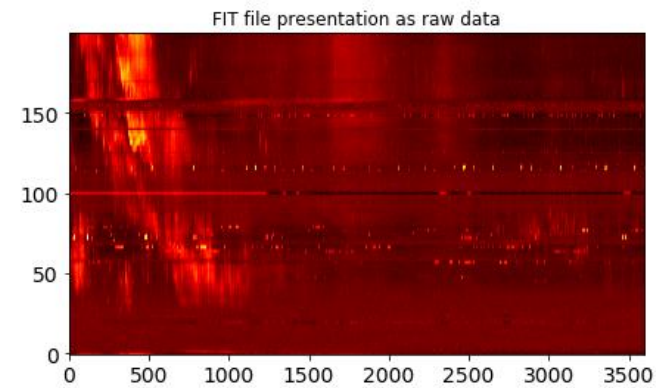

Figure 4. 1: Original image without frequency and time axes.

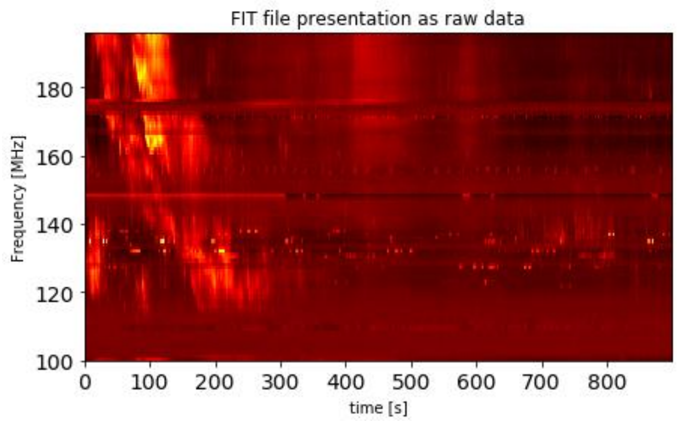

Figure 4. 2: Original image with frequency and time axes.

- Gradually I started to apply four kinds of image processing techniques like GaussianBlur, Threshold, Erode and Dilate to remove noise from the solar burst and identify the solar burst correctly from the plotted image. In here I applied four iterations of erode technique and one iteration from dilate technique.

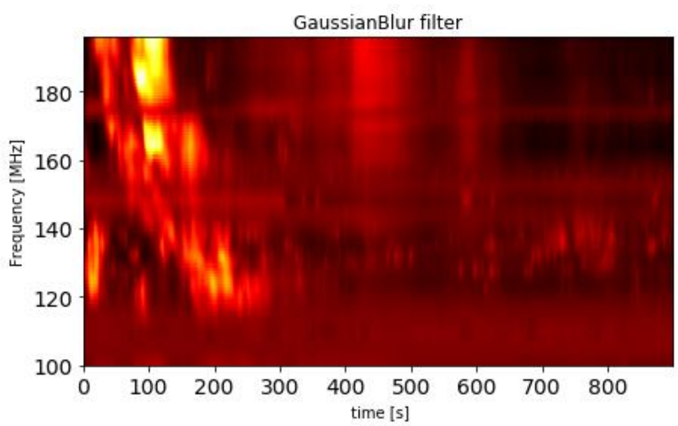

Figure 4. 3: Solar burst image after gaussianblur filter.

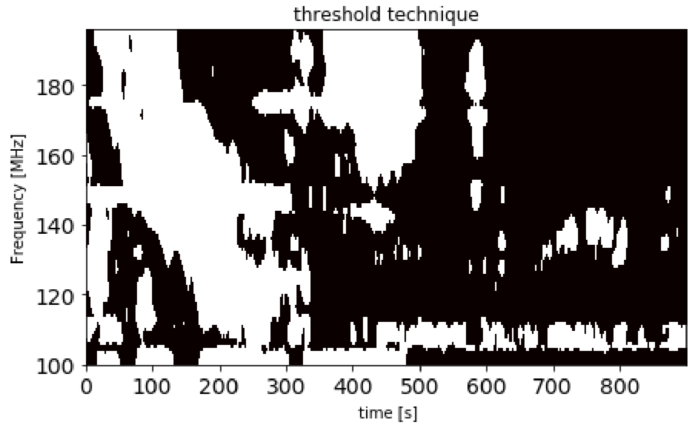

Figure 4. 4: Image after applying threshold technique.

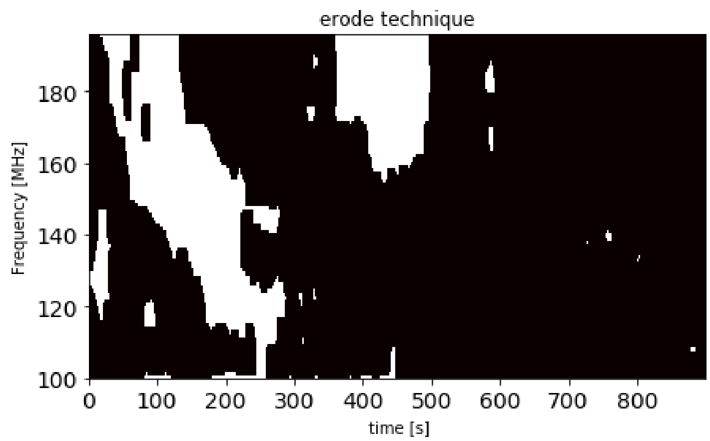

Figure 4. 5: Image after applying erode technique.

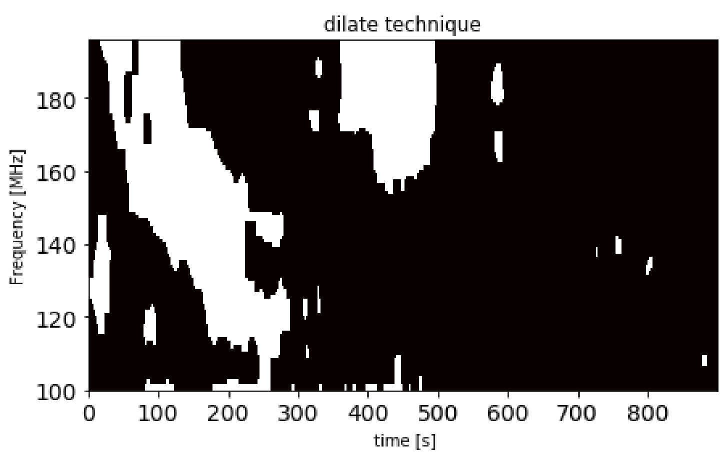

Figure 4. 6: Image after applying dilate technique.

- After that finally I was able to get a somewhat better mask of the solar burst compare to the original image. When taking this mask I had to put a condition to select the areas with pixels more than 2000.

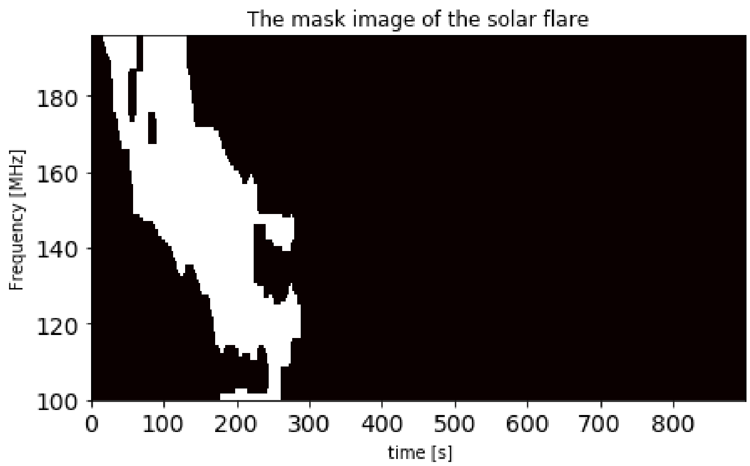

Figure 4. 7: Final identified mask of the solar burst.

- After getting the noise free image of the solar burst then I left with locating the maximum intensity points of the solar burst and following calculations were carried out to select the points.

1. First I selected the non empty positions (which represent 255 value in the array) of the mask array.
2. Then I compared those positions with the original image array and got the pixel intensity value stored in that array.
3. An array was formed using above values and got the average from those values.
4. Then I gave a command to select the values which are higher than the average value and formed a new array using that values.
5. When forming the above array I gave another two command to form array with x axis(time) values and y axis(frequency) values.
6. Finally I gave command to plot x and y arrays in order to get the following image given below.

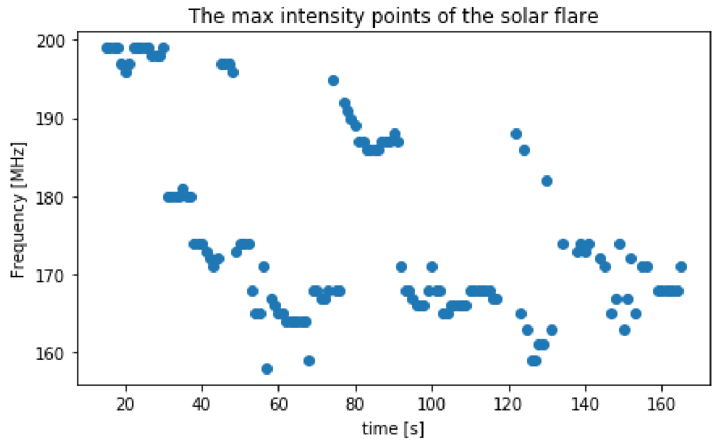

Figure 4. 8: Maximum intensity points of the identified solar burst.

- Finally I had to fit a curve to the located points in order to find the drift rate. Additionally I plot a residual graph to observe best fit for the points.

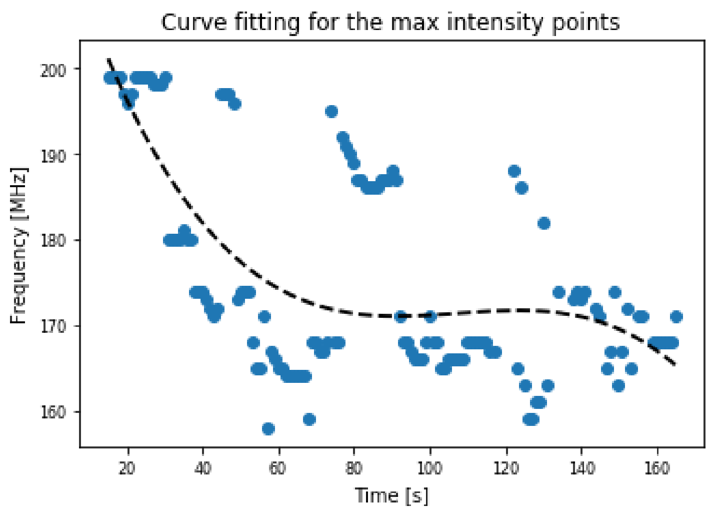

Figure 4. 9: Curve fit for the max intensity points of the identified solar burst.

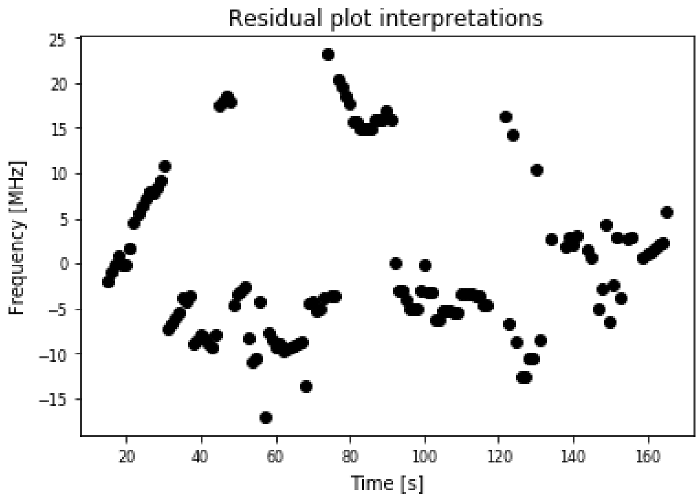

Figure 4. 10: Residual graph of the curve fit.

- Then finally the drift rate of was calculated by taking the derivative of function of the curve.

4.2 User Interface development

4.2.1 Requirements

I had to design a Graphical User Interface for this project as an additional requirement. At first in order to fulfill this task I studied about Tkinter GUI library and some techniques as follows.

- Studied how to make a simple window, frames, buttons and layouts.
- Studied how to connect different frames using buttons.
- Learned how to clear previous frames.

4.2.2 Implementation

Before starting to design the interface using tkinter I simply sketched the necessary frames using a pencil. Then I started to design them according to following steps.

- After studying more about tkinter I simply made a window for the introduction of the application.

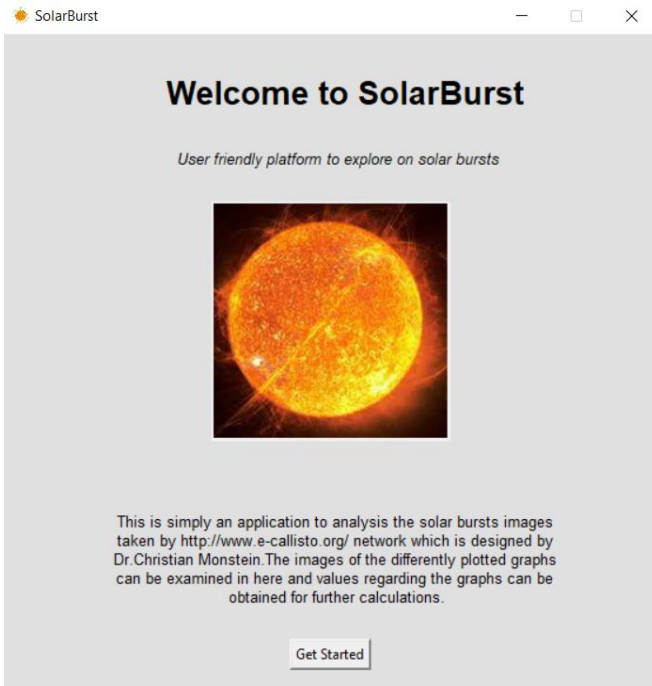

Figure 4. 11: The start page of the application.

- Then I made a page with two options to insert fits file either
    1. manually or
    2. directly extract from http://www.e-callisto.org/ website.

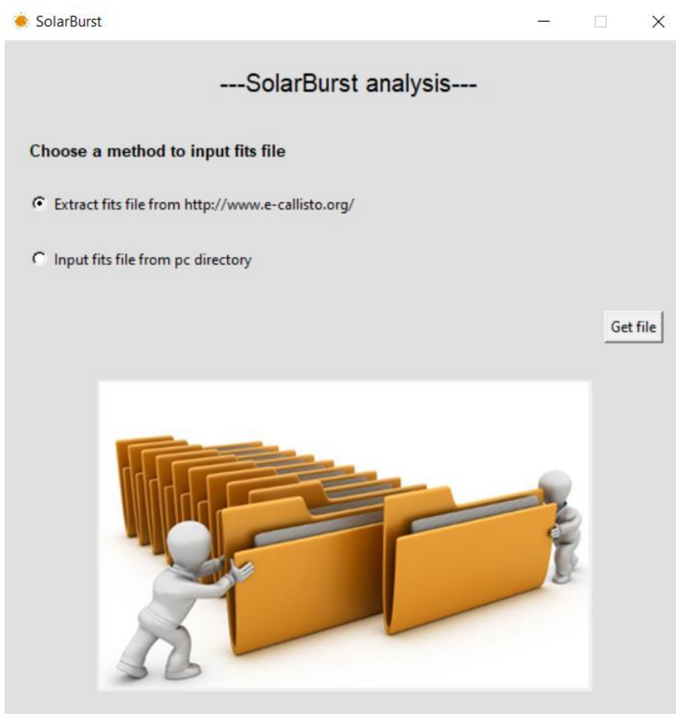

Figure 4. 12: Interface to choose option to input file.

- First option brings user to this page as shown in below. Here I made three entries. User had to insert the date , station and time in the given format to extract the fits file.

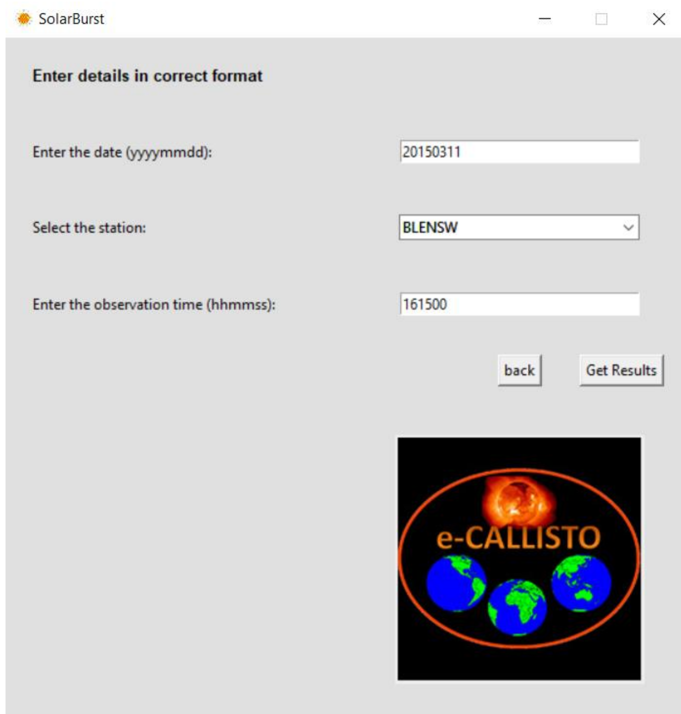

Figure 4. 13: Interface to enter details of fits file to extract it from e-CALLISTO network.

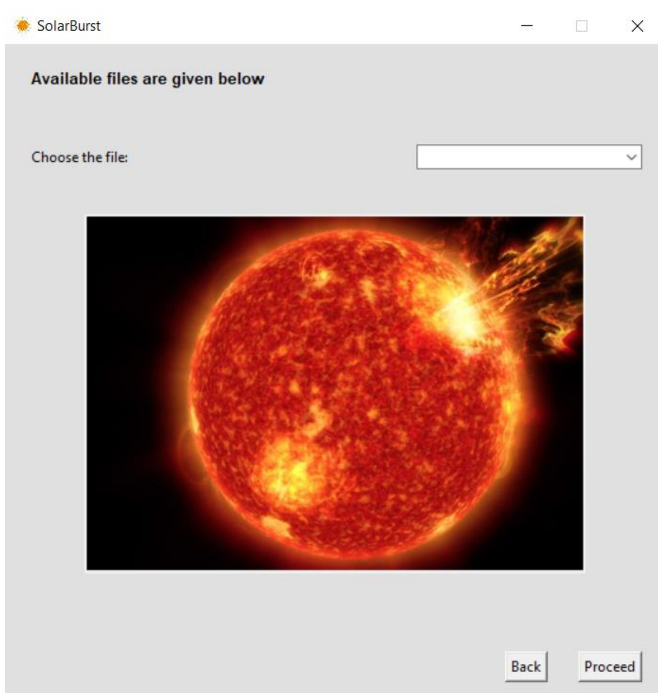

Figure 4. 14: Available fits file presentation.

- Otherwise it will leads to this page. In here user have to simply upload the fits file from pc directory.

Figure 4. 15: Interface to browse fits file from PC directory.

- In here user can select the graph that wanted plot using the drop down menu.

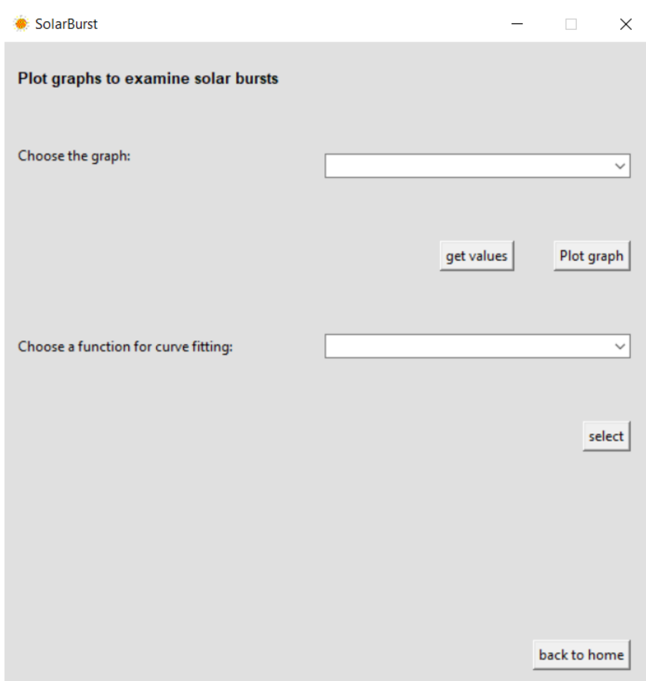

Figure 4. 16: Interface to select graphs for the plot.

- I made a button named “get values” which allows user to save.txt file with the values of x and y axes of the graph.

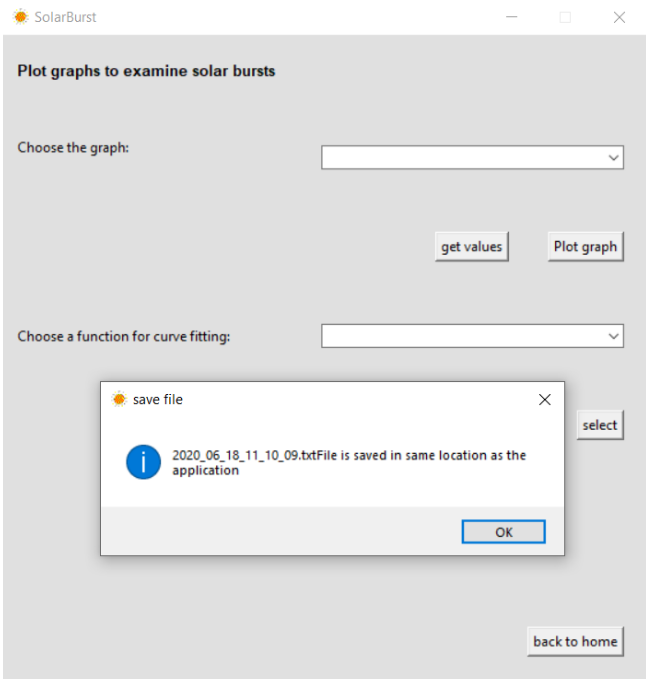

Figure 4. 17: Message display for file save.

- Then I made a option to observe the curve fitted plots.

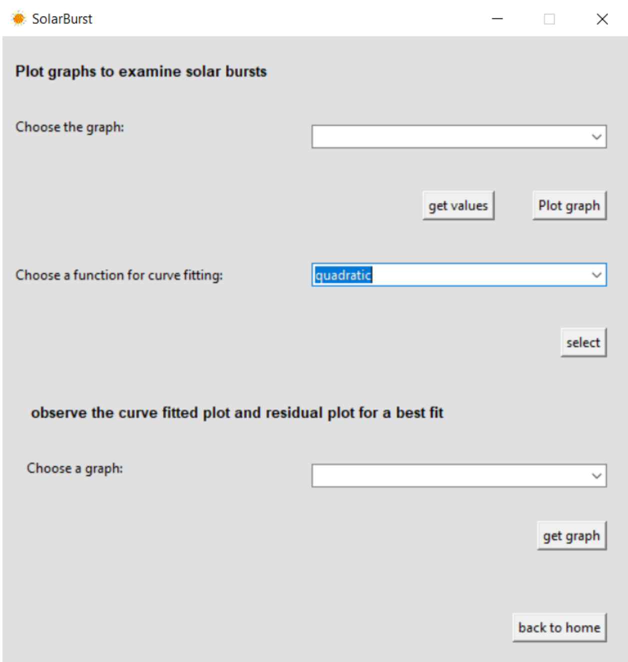

Figure 4. 18: Extended interface with more plot options.

- Finally I designed this window to display the selected graph. I made this window with tool bar which gives different options like zoom, pan axis and save figure.

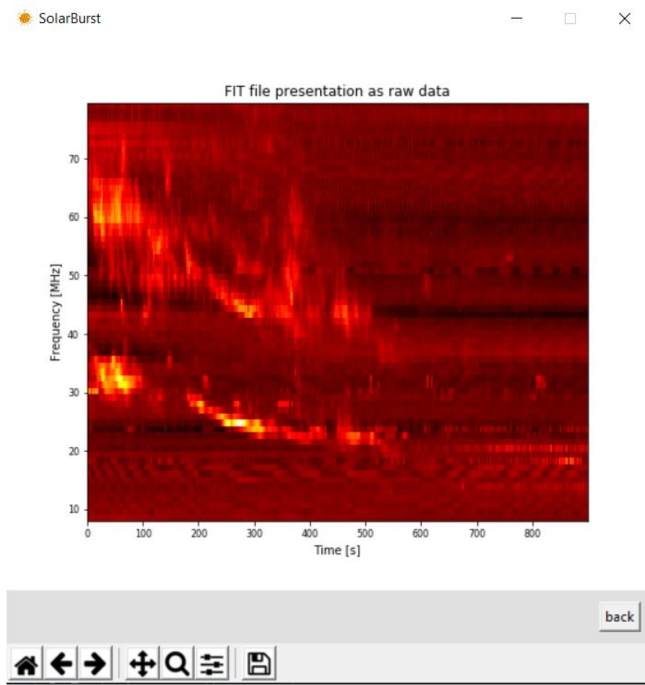

Figure 4. 19: The plot of the image.

- The drift rate can be calculated in this interface by giving the time in seconds.

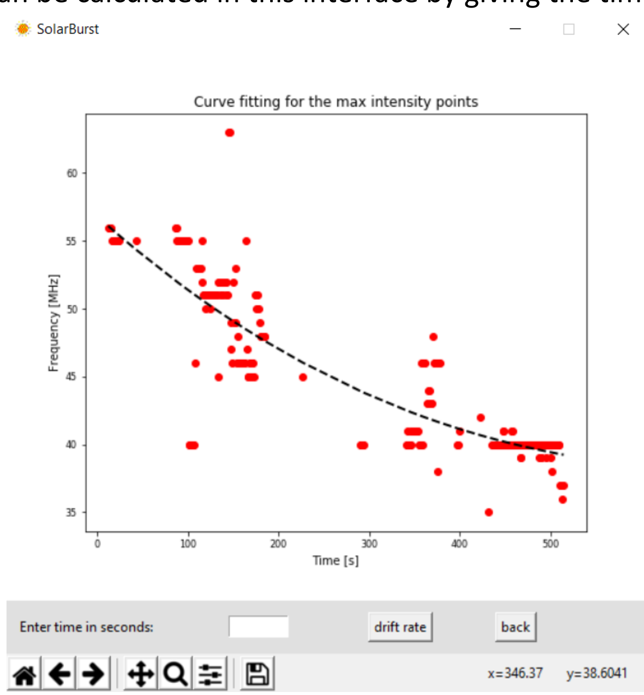

Figure 4. 20: The final Page of the application.

5. Gantt Chart

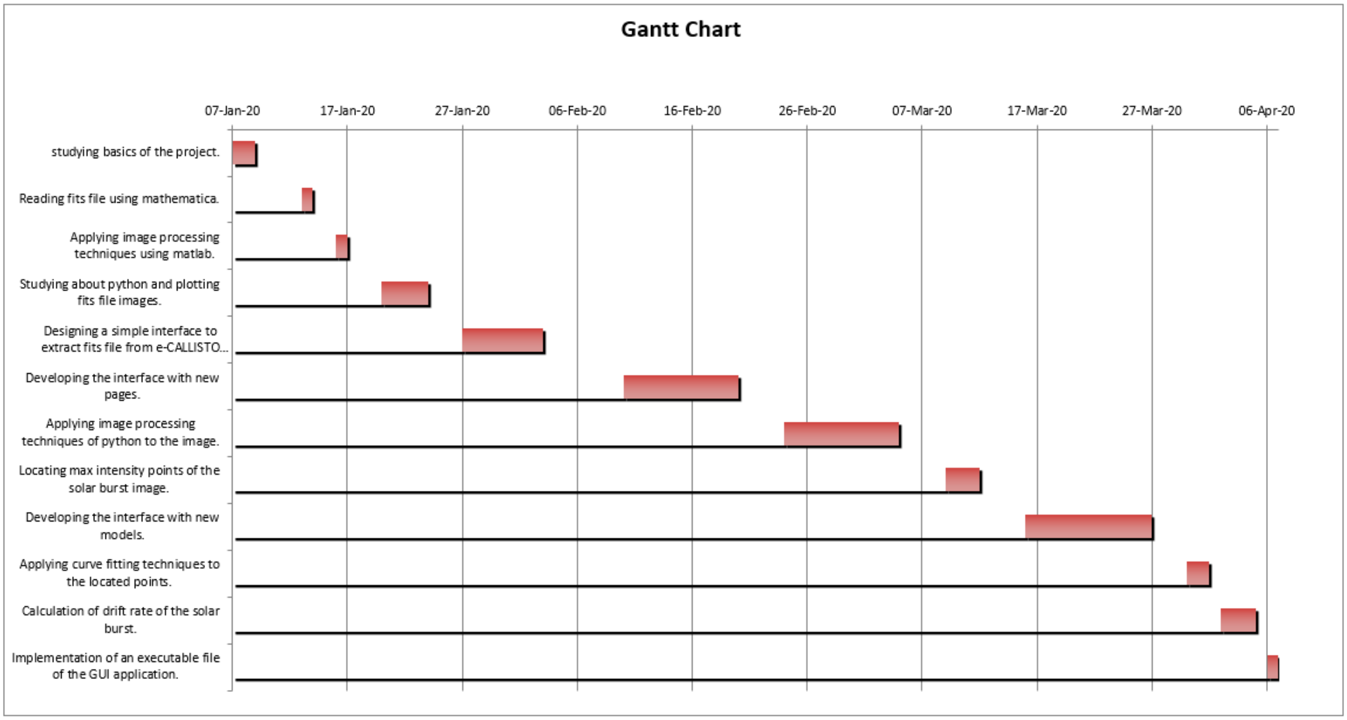

Figure 5. 1: The work schedule during 3 months.

6. Results and discussion

6.1 Results

The main outcome of this project was to identify the solar burst and locate maximum intensity points of the solar burst image to get the drift rate by doing a curve fit through the points.

So in this project I was able to locate the maximum intensity points of the selected solar burst.

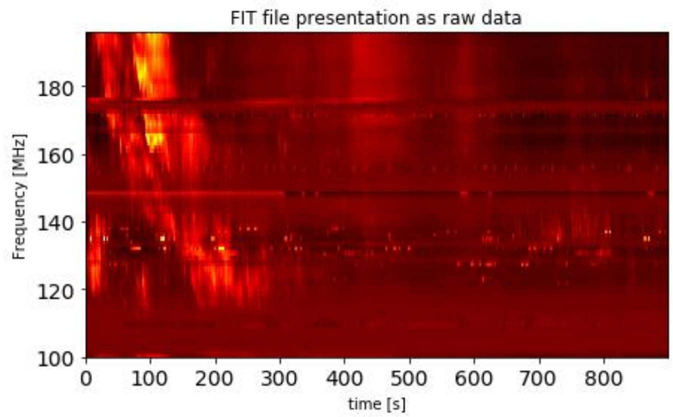

Figure 6. 1: The Selected solar burst

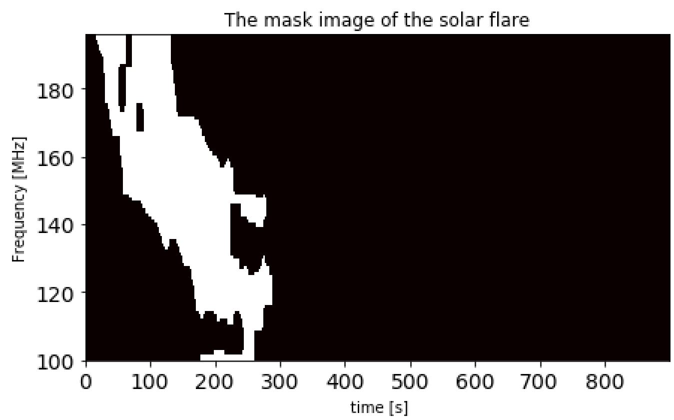

Figure 6. 2: The identified solar burst

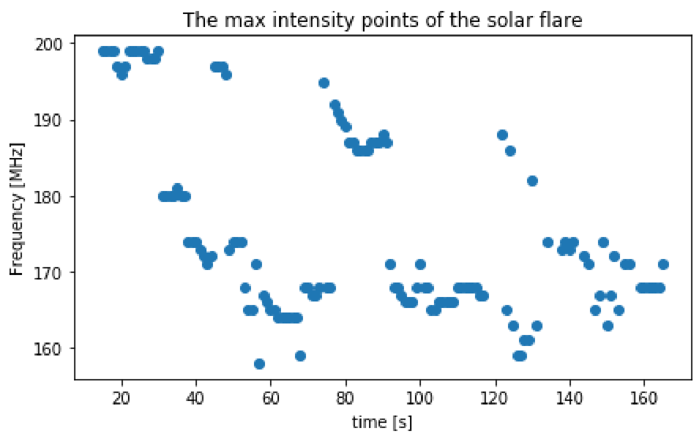

Figure 6. 3: Located max intensity points of the Solar burst.

The diagram of the result of best curve fit for the located points is given below.

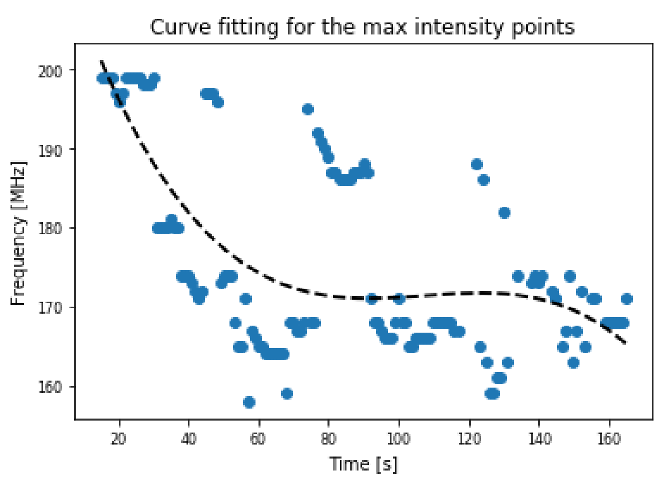

Figure 6. 4: The best curve fit for the points.

The drift rates of different solar flares using the function of the curve was calculated according to the user input time.

6.2 Discussion

6.2.1 Problems encountered

At first I started to do this project using matlab but after I met with founder of e-CALLISTO network Dr.Christian Monstein, he told me that it is better to do the project using python and he gave me some python tutorials in handling fits files. So I started to learn about python to work out on this project. The main purpose of this industrial project was to find the maximum intensity points of the type ii solar bursts and fit a curve through the points using a matching function in order to calculate the drift rate. During implementing process I encountered many problems. I could resolve these problems with the help of my supervisor and my colleagues.

- When running python file dealt with the errors occurred due to misusing expressions, using class variables incorrectly, specifying parameters incorrectly.
- Had to try numerous image processing techniques in order to get better results.
- Faced some issues with implementing the executable file of the final GUI application./

Although I was able to extract the solar burst from the image by removing the noise I hope there are more image processing techniques and curve fitting techniques to get better results Due to limited time It was hard to try every possible techniques. Somehow I was able to complete the given tasks through the given period of time at a better successful rate.

6.2.1 Further Improvements

In the given limited time I had to do this project while learning the new things from the start. So because of that I was unable to make a better GUI application to make this process a successful one. So in future I am planning to obtain better results with new image processing techniques and develop the interface with some new replacements.

- Update interface with error messages when user inputs incorrect data, when there is issues with the app.
- Try different image processing techniques to get better results compare to this one.
- Give user input to select the number of pixels that should select when taking the mask of the solar burst.
- Sometimes the solar burst is captured in two frames. So I am planning to merge the two frames in to one so user can carry out calculations correctly.
- Suggestion to make it a success application and distribute among public audience.

7. Conclusion

- Python can be used by any beginners to do their projects related to programming as it is a simple and easy to learn language.
- The brightest spot of an image can be identified using image processing techniques like Gaussianblur, threshold, erode and dilate which can be found in OpenCV python package.
- Tkinter can be used by beginners who are interested make simple interfaces as it is also easy to learn by beginners.

8. Feedback

The 12 weeks that I spend as a trainee in the Arthur C. Clarke Institute for Modern Technologies (ACCIMT) was a very fruitful period. During that time, I learnt lot of new knowledgeable facts and technologies within a short period of time. So here I got a huge experience with related to new technologies in compare to what I expected. It was all thanks to the physics special degree course. I was able to get many experiences regarding different fields specially in Astronomy field which I never had experiences. I didn't have a clear idea about solar flares but thanks to this project I was able to gain some knowledge regarding them. I had only a little experience with python but because of my project I was able to know more about python and able to make an interface using python for the first time. I was also self-learnt about many things in order to complete this task and that helped me in improving my reading and comprehension skills.

The staff of the ACCIMT was a huge support in completing my project successfully. Specially Mr.Janaka Adassuriya, industrial supervisor of the project gave me advices and support in order to make it a success. Apart to the project in there I was able to observe different planets and stars with the support from the staff there. I was able participate in workshops and training programs held for school students. This helps me to gain some knowledge about Astronomy field and also improved some of my soft skills which is a huge help for my future career works.

So as an overall view I was satisfied with my internship at ACCIMT and I recommend this institute for future undergraduates in physics special degree course.

9. Summary

Type ii Solar Bursts are very rare type of solar bursts which show very slow drift from high frequency to low frequency. This report provides how to calculate drift rate of a type ii solar burst by removing noise and identifying the solar burst maximum intensity points. A simple interface was designed to carry this task. Commands in Python language, image processing techniques gaussian blur, threshold, erode, dilate in OpenCV and Tkinter GUI package was used to make this task a successful one. All the python commands can be found in the annexes. Results shows that OpenCV is better for image processing techniques and also curve fitting techniques in python can be used to achieve better results. This report concludes that python language and its' different techniques in image processing can be used by any beginners to do any programming related projects as it is easy to learn and gives better results as we expected.

10. References

[1] C. Monstein, "Catalog of dynamic electromagnetic spectra observed with callisto".

[2] "The fits support office," [Online]. Available: https://fits.gsfc.nasa.gov/. [Accessed 08 01 2020].

[3] "e-CALLISTO," [Online]. Available: http://www.e-callisto.org/index.html. [Accessed 08 01 2020].

[4] S. M. White, "Solar radio bursts and space weather".

[5] "Frequency Drift Rate Investigation of Solar Radio," [Online]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/180/1/012048/pdf. [Accessed 12 01 2020].

[6] "fits file handling," [Online]. Available: https://docs.astropy.org/en/stable/io/fits/. [Accessed 20 01 2020].

[7] "Python-GUI programming," [Online]. Available: https://www.tutorialspoint.com/python/python_gui_programming.htm. [Accessed 29 01 2020].

[8] "Python GUI examples," [Online]. Available: https://likegeeks.com/python-gui-examples-tkinter-tutorial/. [Accessed 29 01 2020].

[9] "Introduction to GUI programming with tkinter," [Online]. Available: https://python-textbok.readthedocs.io/en/1.0/Introduction_to_GUI_Programming.html. [Accessed 30 01 2020].

[10] "Basics image processing in python," [Online]. Available: https://www.codementor.io/@innat_2k14/image-data-analysis-using-numpy-opencv-part-1-kfadbafx6. [Accessed 20 02 2020].

[11] "Detecting multiple bright spots in an image with Python and OpenCV," [Online]. Available: https://www.pyimagesearch.com/2016/10/31/detecting-multiple-bright-spots-in-an-image-with-python-and-opencv/. [Accessed 10 03 2020].

[12] "Fitting curves," [Online]. Available: https://scientific-python-101.readthedocs.io/scipy/fitting_curves.html. [Accessed 28 03 2020].

[13] "Basic Curve Fitting of Scientific Data with Python," [Online]. Available: https://towardsdatascience.com/basic-curve-fitting-of-scientific-data-with-python-9592244a2509. [Accessed 29 03 2020].

[14] "Curve Fitting with Linear and Nonlinear Regression," [Online]. Available: https://blog.minitab.com/blog/adventures-in-statistics-2/curve-fitting-with-linear-and-nonlinear-regression. [Accessed 29 03 2020].

[15] "Solving Equations and Writing Expressions with SymPy and Python," [Online]. Available: https://pythonforundergradengineers.com/sympy-expressions-and-equations.html. [Accessed 02 04 2020].

[16] "Auto py to exe," [Online]. Available: https://dev.to/eshleron/how-to-convert-py-to-exe-step-by-step-guide-3cfi. [Accessed 06 04 2020].

11. Annexes

```python
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg,NavigationToolbar2Tk
import astropy.io.fits as fits
import pandas as pd
import numpy as np
from matplotlib import cm
import tkinter as tk
from tkinter import ttk
from scipy.ndimage.filters import gaussian_filter
from PIL import Image,ImageTk
import threading
from tkinter.filedialog import askopenfile
import cv2
import imutils
from skimage import measure
import sys,os
from scipy.optimize import curve_fit
from tkinter import messagebox
from sympy import *
import datetime

LARGE_FONT= ("Verdana", 12)


class main(tk.Tk):

    def __init__(self, *args, **kwargs):

        tk.Tk.__init__(self, *args, **kwargs)
        container = tk.Frame(self)
        container.pack(side="top", fill="both", expand = True)
        self.title("SolarBurst")
        self.iconbitmap(self.resource_path(r'sun.ico'))
        self.geometry("560x560")
        self.resizable(0,0)

        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)

        self.frames = {}

        for F in (HomePage,StartPage,ExtraPage, PageOne,PageTwo,PageThree):

            frame = F(container, self)

            self.frames[F] = frame

            frame.grid(row=0, column=0, sticky="nsew")

        self.show_frame(HomePage)

    def show_frame(self, page_name):
        for fme in self.frames:     # Remove all frames
            self.frames[fme].grid_remove()
        frame = self.frames[page_name]
        frame.grid()

    def get_page(self, page_name):
        for page in self.frames.values():
            if str(page.__class__.__name__) == page_name:
                return page
```

```
        return None

    def resource_path(self,relative_path):

        try:

            base_path = sys._MEIPASS
        except Exception:
            base_path = os.path.abspath(".")

        return os.path.join(base_path, relative_path)




class HomePage(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self,parent)
        self.controller = controller
        self.display(controller)
        self.config(bg = 'gray88')

        self.grid_columnconfigure(0, weight=1)
        self.grid_rowconfigure(0, weight=1)


    def display(self,controller):
        #Start page
        label= tk.Label(self, text = "   Welcome to SolarBurst",bg="gray88",justify=tk.CENTER)
        label.grid(column=0, row=0, sticky=tk.S, pady=20, padx=20)
        label.config(font=("",20,"bold"))

        label= tk.Label(self, text = "   User friendly platform to explore on solar bursts",bg="gray88",justify=tk.CENTER)
        label.grid(column=0, row=1, sticky=tk.S, pady=5, padx=20)
        label.config(font=("",10,"italic"))

        quote1 = """

        This is simply an application to analysis the solar bursts images
        taken by http://www.e-callisto.org/ network which is designed by
        Dr.Christian Monstein.The images of the differently plotted graphs
        can be examined in here and values regarding the graphs can be
        obtained for further calculations."""

        label1= tk.Label(self, text = quote1,bg="gray88")
        label1.grid(column=0, row=4, sticky=tk.W, pady=5, padx=60)
        label1.config(font=("", 10))


        canvas1 = tk.Canvas(self,height=200,width=200)
        canvas1.grid(column=0, row=3, pady=20, padx=50)
        canvas1.background = ImageTk.PhotoImage(Image.open(controller.resource_path("ww1.jpg")))
        canvas1.create_image(1,1,image=canvas1.background,anchor='nw')

        button1 = tk.Button(self, text="Get Started",
                        command=lambda: [controller.show_frame(ExtraPage),
                                         ExtraPage.get_input(self.controller.get_page("ExtraPage"),controller)])

        button1.grid(column=0, row=5, pady=20,padx=20, sticky=tk.S)

```

```python

class ExtraPage(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self,parent)
        self.controller = controller
        self.config(bg = 'gray88')
        self.grid_columnconfigure(0, weight=1)


    def get_input(self, controller):
        #second page
        label= tk.Label(self, text = " ---SolarBurst analysis---",bg="gray88",justify=tk.CENTER)
        label.grid(column=0, row=0, sticky=tk.S, pady=20, padx=20)
        label.config(font=("",15))


        label=tk.Label(self, text="""Choose a method to input fits file""",justify = tk.LEFT,bg="gray88")
        label.grid(column=0, row=1, sticky=tk.W, pady=10, padx=20)
        label.config(font=("",10,"bold"))

        #radio button inputs to select fits file
        self.v = tk.IntVar()
        self.v.set(1)
        rb1=tk.Radiobutton(self,text="Extract fits file from http://www.e-callisto.org/",
                           variable=self.v, value=1,bg="gray88")
        rb1.grid(column=0, row=2,sticky=tk.W, pady=10,padx=20)
        rb2=tk.Radiobutton(self,text="Input fits file from pc directory", variable=self.v,  value=2,bg="gray88")
        rb2.grid(column=0, row=3,sticky=tk.W, pady=10,padx=20)
        button = tk.Button(self, text="Get file",
                           command=lambda:[controller.show_frame(StartPage),
                                           StartPage.quit_loop(self.controller.get_page("StartPage"),controller)])
        button.grid(column=0, row=4, pady=20,padx=20, sticky=tk.SE)

        canvas2 = tk.Canvas(self,height=250,width=400)
        canvas2.grid(column=0, row=5, pady=10, padx=20)
        canvas2.background = ImageTk.PhotoImage(Image.open(controller.resource_path("file2.jpg")))
        canvas2.create_image(1,1,image=canvas2.background,anchor='nw')

class StartPage(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self,parent)
        self.controller = controller
        self.config(bg = 'gray88')

        self.grid_columnconfigure(0, weight=1)
        self.grid_rowconfigure(0, weight=1)
        self.grid_rowconfigure(1, weight=1)
        self.grid_rowconfigure(2, weight=1)
        self.grid_rowconfigure(3, weight=1)

    def on_cancel(self):
        #destroy all the widgets in the current frame
        for wid in self.winfo_children():
            wid.destroy()

    def quit_loop(self,controller):
        extrapage = self.controller.get_page("ExtraPage")
        global selection
        selection =extrapage.v.get()
```

```python

class ExtraPage(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self,parent)
        self.controller = controller
        self.config(bg = 'gray88')
        self.grid_columnconfigure(0, weight=1)


    def get_input(self, controller):
        #second page
        label= tk.Label(self, text = " ---SolarBurst analysis---",bg="gray88",justify=tk.CENTER)
        label.grid(column=0, row=0, sticky=tk.S, pady=20, padx=20)
        label.config(font=("",15))


        label=tk.Label(self, text="""Choose a method to input fits file""",justify = tk.LEFT,bg="gray88")
        label.grid(column=0, row=1, sticky=tk.W, pady=10, padx=20)
        label.config(font=("",10,"bold"))

        #radio button inputs to select fits file
        self.v = tk.IntVar()
        self.v.set(1)
        rb1=tk.Radiobutton(self,text="Extract fits file from http://www.e-callisto.org/",
                           variable=self.v, value=1,bg="gray88")
        rb1.grid(column=0, row=2,sticky=tk.W, pady=10,padx=20)
        rb2=tk.Radiobutton(self,text="Input fits file from pc directory", variable=self.v,  value=2,bg="gray88")
        rb2.grid(column=0, row=3,sticky=tk.W, pady=10,padx=20)
        button = tk.Button(self, text="Get file",
                           command=lambda:[controller.show_frame(StartPage),
                                           StartPage.quit_loop(self.controller.get_page("StartPage"),controller)])
        button.grid(column=0, row=4, pady=20,padx=20, sticky=tk.SE)

        canvas2 = tk.Canvas(self,height=250,width=400)
        canvas2.grid(column=0, row=5, pady=10, padx=20)
        canvas2.background = ImageTk.PhotoImage(Image.open(controller.resource_path("file2.jpg")))
        canvas2.create_image(1,1,image=canvas2.background,anchor='nw')

class StartPage(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self,parent)
        self.controller = controller
        self.config(bg = 'gray88')

        self.grid_columnconfigure(0, weight=1)
        self.grid_rowconfigure(0, weight=1)
        self.grid_rowconfigure(1, weight=1)
        self.grid_rowconfigure(2, weight=1)
        self.grid_rowconfigure(3, weight=1)

    def on_cancel(self):
        #destroy all the widgets in the current frame
        for wid in self.winfo_children():
            wid.destroy()

    def quit_loop(self,controller):
        extrapage = self.controller.get_page("ExtraPage")
        global selection
        selection =extrapage.v.get()
```

```
                                                  "Malaysia_UKM",
                                                  "Melibea",
                                                  "NEWZEALAND-AUT",
                                                  "NORWAY-NY-AALESUND",
                                                  "NORWAY-RANDABERG",
                                                  "OOTY",
                                                  "OSRA",
                                                  "RCAG",
                                                  "ROSWELL-NM",
                                                  "SOUTHAFRICA-SANSA",
                                                  "SSRT",
                                                  "SWISS-BLEN5M",
                                                  "SWISS-HB9SCT",
                                                  "SWISS-HEITERSWIL",
                                                  "SWISS-IRSOL",
                                                  "SWISS-Landschlacht",
                                                  "SWISS-MUHEN",
                                                  "TRIEST",
                                                  "USA-ARIZONA-ERAU",

                                                  ])
            self.drop.grid(column=2, row=2, padx=20, pady=10)

            dateString = tk.StringVar()
            timeString = tk.StringVar()
            #input entry to date of the solar burst
            self.entryDate = tk.Entry(self, width=32, textvariable=dateString)
            self.entryDate.grid(column=2, row=1, padx=20, pady=10)
            #input entry to time of the solar burst
            self.entryTime = tk.Entry(self, width=32, textvariable=timeString)
            self.entryTime.grid(column=2, row=3, padx=20, pady=10)

            button = tk.Button(self, text="Get Results",
                                   command=lambda:[ controller.show_frame(PageOne),
                                                   PageOne.start_thread(self.controller.get_page("PageOne"),controller)])
            button.grid(column=2, row=4, pady=10,padx=20, sticky=tk.SE)

            btn3 = tk.Button(self, text="back",
                        command=lambda:[controller.show_frame(ExtraPage),self.on_cancel()])

            btn3.grid(column=2, row=4, pady=10,padx=120, sticky=tk.SE)

            canvas3 = tk.Canvas(self,height=200,width=200,bg="gray88")
            canvas3.grid(column=2, row=5, pady=30, padx=20)
            canvas3.background = ImageTk.PhotoImage(Image.open(controller.resource_path("callisto.jpg")))
            canvas3.create_image(1,1,image=canvas3.background,anchor='nw')


        #if fits file is upload from PC directory
        elif selection==2:

            label=tk.Label(self, text="""Browse or Enter the file location""",justify = tk.LEFT,bg="gray88")
            label.grid(column=0, row=0, sticky=tk.W, pady=10,padx=40)
            label.config(font=("",10,"bold"))

            fileString=tk.StringVar()
            #input entry for location of the fits file
            self.ent1=tk.Entry(self,width=60, textvariable=fileString)
            self.ent1.grid(column=0,row=1,sticky=tk.W,padx=40, pady=20)

            def browsefunc():
```

```
                file = askopenfile(filetypes =[('fit Files', '*.fit')])
                self.ent1.insert(tk.END, file.name)



            btn1 = tk.Button(self, text ='browse', command = lambda:browsefunc())
            btn1.grid(column=0, row=1, pady=20,padx=20, sticky=tk.SE)

            btn2 = tk.Button(self, text ='proceed', command = lambda:[ controller.show_frame(PageTwo),
                                                                      PageTwo.final_out(self.controller.get_page("PageTwo"),controller)])
            btn2.grid(column=0, row=3, pady=20,padx=20, sticky=tk.SE)

            btn3 = tk.Button(self, text="back",
                        command=lambda:[controller.show_frame(ExtraPage),self.on_cancel()])

            btn3.grid(column=0, row=3, pady=20,padx=100, sticky=tk.SE)

            canvas1 = tk.Canvas(self,height=300,width=420,bg="gray88")
            canvas1.grid(column=0, row=2,pady=20)
            canvas1.background = ImageTk.PhotoImage(Image.open(controller.resource_path("filebrowse.jpg")))
            canvas1.create_image(0,0,image=canvas1.background,anchor='nw')



class PageOne(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self,parent)
        self.controller = controller
        self.config(bg='gray88')

        self.grid_columnconfigure(0, weight=1)
        self.grid_rowconfigure(0, weight=1)
        self.grid_rowconfigure(1, weight=1)
        self.grid_rowconfigure(2, weight=1)
        self.grid_rowconfigure(3, weight=1)

    def start_thread(self,controller):
            th = threading.Thread(target=self.get_output(controller))
            th.start()


    def on_cancel(self):
        for wid in self.winfo_children():
            wid.destroy()

    def get_output(self,controller):
        startpage = self.controller.get_page("StartPage")
        d1 = startpage.drop.get()  #get input station
        d2 = startpage.entryDate.get()  #get input date
        d3 = startpage.entryTime.get() #get input time
        url_test = 'http://soleil.i4ds.ch/solarradio/callistoQuicklooks/?date={s}' #url for the e-CALLISTO network
        url = url_test.format(s=d2) #format link using input
        tables = pd.read_html(url) #read table in the link to a list
        df = tables[0] #convert list to a dataframe
        input_list = [d1, d2, d3]# inputs in an array
        seperator = '_'
        p1=seperator.join(input_list)#join inputs with string "_"
        df1 = df[df[0].astype(str).str.match(p1)]# extract files related to inputs
        choices = df1[0].tolist() #convert it in to a list
        self.drop1 = ttk.Combobox(self, value=choices, width = 28)#enter the list in to a drop down menu
```

```
        choices = df1[0].tolist() #convert it in to a list
        self.drop1 = ttk.Combobox(self, value=choices, width = 28)#enter the list in to a drop down menu
        self.drop1.grid(column=0, row=1,sticky=tk.SE, pady=10,padx=20)
        labelselection1 = tk.Label(self, text = "Available files are given below", bg="gray88",justify=tk.LEFT)
        labelselection1.grid(column=0, row=0, sticky=tk.W,  pady=10, padx=20)
        labelselection1.config(font=( "",10, "bold"))
        labelselection2 = tk.Label(self, text = "Choose the file: ", bg="gray88")
        labelselection2.grid(column=0, row=1, sticky=tk.SW, padx=20,pady=10)
        button1 = tk.Button(self, text="Back",
                        command=lambda: [controller.show_frame(StartPage),self.on_cancel()])
        button1.grid(column=0, row=3, pady=20,padx=100, sticky=tk.SE)
        button2 = tk.Button(self, text="Proceed",
                           command=lambda: [ controller.show_frame(PageTwo),
                                            PageTwo.final_out(self.controller.get_page("PageTwo"),controller)])
        button2.grid(column=0, row=3,pady=20,padx=20, sticky=tk.SE)


        canvas = tk.Canvas(self,bg='gray88',height=300,width=420)
        canvas.grid(column=0, row=2, pady=20, padx=10)
        canvas.background = ImageTk.PhotoImage(Image.open(controller.resource_path("sf4.jpg")))
        canvas.create_image(1,1,image=canvas.background,anchor='nw')




class PageTwo(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self,parent)
        self.controller = controller
        self.config(bg='gray88')

        self.grid_columnconfigure(0, weight=1)
        self.grid_rowconfigure(0, weight=1)
        self.grid_rowconfigure(1, weight=1)
        self.grid_rowconfigure(2, weight=1)
        self.grid_rowconfigure(3, weight=1)
        self.grid_rowconfigure(4, weight=1)
        self.grid_rowconfigure(5, weight=1)
        self.grid_rowconfigure(6, weight=1)
        self.grid_rowconfigure(7, weight=1)
        self.grid_rowconfigure(8, weight=1)


    def on_cancel(self):
        for wid in self.winfo_children():
            wid.destroy()

    def final_out(self,controller):

        extrapage = self.controller.get_page("ExtraPage")
        global selection
        selection =extrapage.v.get()

        labelselection1 = tk.Label(self, text = "Plot graphs to examine solar bursts", bg="gray88",justify=tk.LEFT)
        labelselection1.grid(column=0, row=0, sticky=tk.NW,  pady=20, padx=10)
        labelselection1.config(font=( "",10, "bold"))
        labelselection2 = tk.Label(self, text = "Choose the graph: ", bg="gray88")
        labelselection2.grid(column=0, row=1, sticky=tk.NW,pady=1, padx=10)
        #list of graphs to plot
```

```
        choices=["FIT file presentation as raw data",
                 "Solar burst image with Gaussian blur filter",
                 "Image after applying threshold technique",
                 "Image after applying erode technique",
                 "Image after applying dilate technique",
                 "Identified mask of the Solar Burst"]
        self.drop2 = ttk.Combobox(self, value=choices, width = 40)
        self.drop2.grid(column=0, row=1, sticky=tk.NE,pady=10,padx=20)

        labelselection2 = tk.Label(self, text = "Choose a function for curve fitting: ", bg="gray88")
        labelselection2.grid(column=0, row=3, sticky=tk.NW,pady=10, padx=10)
        #List of functions
        choices1=["linear",
                  "quadratic",
                  "cubic",
                  "polynomial"]
        self.drop3 = ttk.Combobox(self, value=choices1, width = 40)
        self.drop3.grid(column=0, row=3, sticky=tk.NE, pady=10,padx=20)




        if selection ==1:
            btn3 = tk.Button(self, text="back to home",
                        command=lambda:[controller.show_frame(ExtraPage),self.on_cancel()])

            btn3.grid(column=0, row=9, pady=20,padx=20, sticky=tk.NE)

            button1 = tk.Button(self, text="Plot graph",
                        command=lambda:[controller.show_frame(PageThree),
                                        PageThree.calculations1(self.controller.get_page("PageThree"),controller),
                                        PageThree.calculations(self.controller.get_page("PageThree"),controller),
                                        PageThree.graphs(self.controller.get_page("PageThree"),controller)])
            button1.grid(column=0, row=2, pady=20,padx=20, sticky=tk.NE)


            button2 = tk.Button(self, text="select",
                            command=lambda:[controller.show_frame(PageTwo),
                                            PageThree.calculations1(self.controller.get_page("PageThree"),controller),
                                            PageThree.calculations(self.controller.get_page("PageThree"),controller),
                                            PageThree.graphs2(self.controller.get_page("PageThree"),controller),
                                            self.show(controller)
                                            ])
            button2.grid(column=0, row=4, pady=20,padx=20, sticky=tk.NE)
            button3 = tk.Button(self, text="get values",
                        command=lambda:[controller.show_frame(PageTwo),
                                        PageThree.calculations1(self.controller.get_page("PageThree"),controller),
                                        PageThree.calculations(self.controller.get_page("PageThree"),controller),
                                         PageThree.save_file(self.controller.get_page("PageThree"),controller)])
            button3.grid(column=0, row=2, pady=20,padx=120, sticky=tk.NE)




        elif selection==2:
            btn3 = tk.Button(self, text="back to home",
                        command=lambda:[controller.show_frame(ExtraPage),self.on_cancel()])

            btn3.grid(column=0, row=9, pady=20,padx=20, sticky=tk.NE)

            button1 = tk.Button(self, text="Plot graph",
```

```
            button1 = tk.Button(self, text="Plot graph",
                        command=lambda:[controller.show_frame(PageThree),
                                        PageThree.calculations2(self.controller.get_page("PageThree"),controller),
                                        PageThree.calculations(self.controller.get_page("PageThree"),controller),
                                         PageThree.graphs(self.controller.get_page("PageThree"),controller)])
            button1.grid(column=0, row=2, pady=20,padx=20, sticky=tk.NE)

            button2 = tk.Button(self, text="select",
                            command=lambda:[controller.show_frame(PageTwo),
                                            PageThree.calculations2(self.controller.get_page("PageThree"),controller),
                                            PageThree.calculations(self.controller.get_page("PageThree"),controller),
                                            PageThree.graphs2(self.controller.get_page("PageThree"),controller),
                                            self.show(controller)
                                            ])
            button2.grid(column=0, row=4, pady=20,padx=20, sticky=tk.NE)
            button3 = tk.Button(self, text="get values",
                        command=lambda:[controller.show_frame(PageTwo),
                                        PageThree.calculations2(self.controller.get_page("PageThree"),controller),
                                        PageThree.calculations(self.controller.get_page("PageThree"),controller),
                                         PageThree.save_file(self.controller.get_page("PageThree"),controller)])
            button3.grid(column=0, row=2, pady=20,padx=120, sticky=tk.NE)


    def show(self,controller):
        label=tk.Label(self, text=""" observe the curve fitted plot and residual plot for a best fit""",justify = tk.LEFT,bg="gray88")
        label.grid(column=0, row=5, sticky=tk.NW,  pady=20, padx=10)
        label.config(font=("",10,"bold"))

        labelselection3 = tk.Label(self, text = "Choose a graph: ", bg="gray88")
        labelselection3.grid(column=0, row=6, sticky=tk.NW,pady=10, padx=10)
        choices2=["curve fitted plot",
                  "residual plot"]
        self.drop4 = ttk.Combobox(self, value=choices2, width = 40)
        self.drop4.grid(column=0, row=6, sticky=tk.SE, pady=10,padx=20)

        button3 = tk.Button(self, text="get graph",
                            command=lambda:[controller.show_frame(PageThree),
                                            PageThree.graphs2(self.controller.get_page("PageThree"),controller),
                                            PageThree.graphs3(self.controller.get_page("PageThree"),controller)
                                            ])
        button3.grid(column=0, row=7, pady=20,padx=20, sticky=tk.NE)




class PageThree(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self,parent)
        self.controller = controller
        self.config(bg='gray88')

        self.grid_columnconfigure(0, weight=1)
        self.grid_rowconfigure(0, weight=1)

    def on_cancel(self):
        for wid in self.winfo_children():
            wid.destroy()

    def calculations1(self,controller):
```

```
        #if fits file is extract from e_CALLISTO network
        startpage = self.controller.get_page("StartPage")
        dd = startpage.entryDate.get()
        pageOne = self.controller.get_page("PageOne")
        year = (dd[0:4])#extract year from date
        month = (dd[4:6])#extract month from date
        day = (dd[6:8])#extract day from date
        df2=pageOne.drop1.get()
        target_url = 'http://soleil.i4ds.ch/solarradio/data/2002-20yy_Callisto/{a}/{b}/{c}/{d}'
        self.new_url =target_url.format(a=year, b=month, c=day, d=df2)#extract the correct fits file according to the give inputs

    def calculations2(self,controller):
        #if fits file is uploded
        startpage = self.controller.get_page("StartPage")
        self.file_loc=startpage.ent1.get() #extract location from upload fits file



    def calculations(self,controller):
        extrapage = self.controller.get_page("ExtraPage")
        global selection
        selection =extrapage.v.get()

        if selection==1:
            file=self.new_url


        elif selection==2:
            file=self.file_loc

        hdu = fits.open(file)#read fits file
        image_dB = hdu[0].data.astype(np.float32)/255.0*2500.0/25.4  #gets primary data
        mini_dB = np.min(image_dB) # find lowest value
        rel_dB = image_dB - mini_dB# set background 0
        rel_dB = np.flip(rel_dB,0) #flip file
        freqs = hdu[1].data['Frequency'][0] # extract frequency axis
        time  = hdu[1].data['Time'][0] # extract time axis
        hdu.close()

        self.extent = (time[0], time[-1], freqs[-1], freqs[0])
        rel_dB = rel_dB - rel_dB.mean(axis=1, keepdims=True)  # subtract average
        freqdiv=round(200/(freqs[0]-freqs[-1])) #gets constant value to convert y-axis
        timediv=round(3600/(time[-1]-time[0])) #gets constant value to convert x-axis
        #covert time axis
        full=[]
        for y in rel_dB:
            half=[]
            val1= 0.0
            count = 0
            for x in y:
                count = count + 1
                val1 = val1 + x
                if count == timediv:
                    val1 = val1/timediv
                    half.append(val1)
                    val1=0.0
                    count=0

            full.append(half)
```

```python
        full = np.array(full)


        #convert freq axis
        full1=[]
        for y in full.T:
            half1=[]
            val1= 0.0
            count = 0
            for x in y:
                count = count + 1
                val1 = val1 + x
                if count == freqdiv:
                    val1 = (val1/freqdiv)
                    half1.append(val1)
                    val1=0.0
                    count=0

            full1.append(half1)

        #convert in to an array
        self.full1 = np.array(full1)
        self.full1 =self.full1.T

        #image proccesing techniques
        self.blurred = cv2.GaussianBlur(self.full1, (11, 11), 0,0) #blurred image
        self.thresh = cv2.threshold(self.blurred, 0, 255, cv2.THRESH_BINARY)[1]
        self.thresh1 = cv2.erode(self.thresh, None, iterations=4)
        self.thresh2 = cv2.dilate(self.thresh1, None, iterations=1)


        # perform a connected component analysis on the thresholded
        # image, then initialize a mask to store only the "large"
        # components
        labels = measure.label(self.thresh2, neighbors=8, background=0)
        self.mask = np.zeros(self.thresh2.shape, dtype="uint8")
        # loop over the unique components
        for label in np.unique(labels):
            # if this is the background label, ignore it
            if label == 0:
                continue
            # otherwise, construct the label mask and count the
            # number of pixels
            labelMask = np.zeros(self.thresh2.shape, dtype="uint8")
            labelMask[labels == label] = 255
            numPixels = cv2.countNonZero(labelMask)
            # if the number of pixels in the component is sufficiently
            # large, then add it to our mask of "large blobs"
            if numPixels > 10000:
                self.mask = cv2.add(self.mask, labelMask)

        #finding max intensities and extracting their locations
        x=[]
        y=[]
        val=[]
        m=self.mask.T
        for i in range(len(m)):
            f=[]
            for j in range(len(m[i])):
                if m[i][j]==255:
                    f.append(j)
```

```
            if len(f)>0:
                maxval=0
                maxy=0
                for z in f:
                    if self.full1.T[i][z] > maxval:
                        maxval=self.full1.T[i][z]
                        maxy=z
                val.append(maxval)
                y.append(maxy+freqs[-1])
                x.append(i)

        val=np.array(val)
        x=np.array(x)
        y=np.array(y)
        print(val)
        valmean=val.mean()#average intensity value

        #get intensity points greater than overange intensity values
        x1=[]
        y1=[]
        for i in range(len(val)):
            if val[i] > valmean:
                y1.append(y[i])
                x1.append(x[i])
        self.x1=np.array(x1)
        self.y1=np.array(y1)

    #save a .txt file with y and x axis date
    def save_file(self,controller):
        coordinate = list(zip(self.x1,self.y1))
        timestr= str(datetime.datetime.now().strftime('%Y_%m_%d_%H_%M_%S'))
        for element in coordinate:
            file1 = open(timestr +".txt","a")
            file1.write(f"{element} \n")
            file1.close()
        messagebox.showinfo("save file",timestr+ ".txt"+ "File is saved in same location as the application")



    def graphs(self,controller):
        pageTwo = self.controller.get_page("PageTwo")
        graph=pageTwo.drop2.get()

        #plot different graphs
        if graph=="FIT file presentation as raw data":
            figure1 = plt.figure(figsize=(7,4))
            ax = figure1.add_subplot(111)
            ax.imshow(self.full1, aspect = 'auto', extent = self.extent, cmap=cm.hot, origin='lower')
            ax.tick_params(labelsize=8)
            chart_type = FigureCanvasTkAgg(figure1, self)
            chart_type.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH)
            ax.set_title('FIT file presentation as raw data')
            ax.set_xlabel('Time [s]')
            ax.set_ylabel('Frequency [MHz]')

        elif graph=="Solar burst image with Gaussian blur filter":
            figure1 = plt.figure(figsize=(7,4))
            ax = figure1.add_subplot(111)
            ax.imshow(self.blurred, aspect = 'auto', extent = self.extent, cmap=cm.hot, origin='lower')
            ax.tick_params(labelsize=8)
            chart_type = FigureCanvasTkAgg(figure1, self)
```

```
            chart_type.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH)
            ax.set_title('Solar burst image with Gaussian blur filter')
            ax.set_xlabel('Time [s]')
            ax.set_ylabel('Frequency [MHz]')

        elif graph=="Image after applying threshold technique":
            figure1 = plt.figure(figsize=(7,4))
            ax = figure1.add_subplot(111)
            ax.imshow(self.thresh, aspect = 'auto', extent = self.extent, cmap=cm.hot, origin='lower')
            ax.tick_params(labelsize=8)
            chart_type = FigureCanvasTkAgg(figure1, self)
            chart_type.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH)
            ax.set_title('Image after applying threshold technique')
            ax.set_xlabel('Time [s]')
            ax.set_ylabel('Frequency [MHz]')

        elif graph=="Image after applying erode technique":
            figure1 = plt.figure(figsize=(7,4))
            ax = figure1.add_subplot(111)
            ax.imshow(self.thresh1, aspect = 'auto', extent = self.extent, cmap=cm.hot, origin='lower')
            ax.tick_params(labelsize=8)
            chart_type = FigureCanvasTkAgg(figure1, self)
            chart_type.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH)
            ax.set_title('Image after applying erode technique')
            ax.set_xlabel('Time [s]')
            ax.set_ylabel('Frequency [MHz]')

        elif graph=="Image after applying dilate technique":
            figure1 = plt.figure(figsize=(7,4))
            ax = figure1.add_subplot(111)
            ax.imshow(self.thresh2, aspect = 'auto', extent = self.extent, cmap=cm.hot, origin='lower')
            ax.tick_params(labelsize=8)
            chart_type = FigureCanvasTkAgg(figure1, self)
            chart_type.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH)
            ax.set_title('Image after applying dilate technique')
            ax.set_xlabel('Time [s]')
            ax.set_ylabel('Frequency [MHz]')

        elif graph=="Identified mask of the Solar Burst":
            figure1 = plt.figure(figsize=(7,4))
            ax = figure1.add_subplot(111)
            ax.imshow(self.mask, aspect = 'auto', extent = self.extent, cmap=cm.hot, origin='lower')
            ax.tick_params(labelsize=8)
            chart_type = FigureCanvasTkAgg(figure1, self)
            chart_type.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH)
            ax.set_title('Identified mask of the Solar Burst')
            ax.set_xlabel('Relative time [s]')
            ax.set_ylabel('Frequency [MHz]')


        toolbar = NavigationToolbar2Tk(chart_type, self)
        toolbar.update()
        chart_type._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)


        button2 = tk.Button(self, text="back",
                        command=lambda:[ controller.show_frame(PageTwo),self.on_cancel()])

        button2.pack(side=tk.RIGHT,padx=10,pady=50)
```

```
    def graphs2(self,controller):
        pageTwo = self.controller.get_page("PageTwo")
        function=pageTwo.drop3.get()

        #different functions for curve fitting
        if function=="linear":
             def gaussian(x, a,b):
                 return a*x+b

             pars,cov =curve_fit(f=gaussian, xdata=self.x1, ydata=self.y1,p0=[0, 0])

        elif function=="quadratic":
             def gaussian(x, a, b, c):
                 return a*np.power(x,2)+b*x+c

             pars,cov = curve_fit(f=gaussian, xdata=self.x1, ydata=self.y1,p0=[0, 0,0])

        elif function=="cubic":
             def gaussian(x, a, b, c,d):
                 return a*np.power(x,3)+b*np.power(x,2)+c*x+d
             pars,cov =curve_fit(f=gaussian, xdata=self.x1, ydata=self.y1,p0=[0, 0,0,0])

        elif function=="polynomial":
             def gaussian(x, a, b, c,d,e):
                 return a*np.power(x,4)+b*np.power(x,3)+c*np.power(x,2)+c*x+d
             pars,cov =curve_fit(f=gaussian, xdata=self.x1, ydata=self.y1,p0=[0, 0,0,0,0])

        self.res = self.y1 - gaussian(self.x1, *pars)
        self.para = gaussian(self.x1, *pars)
        x2=[]
        y2=[]
        for i in range(len(self.res)):
            if -1<self.res[i]<1:
                y2.append(self.y1[i])
                x2.append(self.x1[i])
        self.y2=np.array(y2)
        self.x2=np.array(x2)

    def graphs3(self,controller):
        pageTwo = self.controller.get_page("PageTwo")
        plot=pageTwo.drop4.get()



        if plot=="curve fitted plot":
            figure1 = plt.figure()
            ax = figure1.add_subplot(111)
            ax.scatter(self.x1,self.y1,color="red")
            ax.plot(self.x1, self.para, linestyle='--', linewidth=2, color='black')
            ax.tick_params(labelsize=8)
            chart_type = FigureCanvasTkAgg(figure1, self)
            chart_type.get_tk_widget().pack(side=tk.TOP,expand=True)
            ax.set_title('Curve fitting for the max intensity points')
            ax.set_xlabel('Time [s]')
            ax.set_ylabel('Frequency [MHz]')

        elif plot=="residual plot":
            figure1 = plt.figure()
            ax = figure1.add_subplot(111)
            ax.plot(self.x1,self.res,'o', color='black')
```

```
            ax.tick_params(labelsize=8)
            chart_type = FigureCanvasTkAgg(figure1, self)
            chart_type.get_tk_widget().pack(side=tk.TOP,expand=True)
            ax.set_title('Residual plot interpretations')
            ax.set_xlabel('Time [s]')
            ax.set_ylabel('Frequency [MHz]')

        toolbar = NavigationToolbar2Tk(chart_type, self)
        toolbar.update()
        chart_type._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)


        time = tk.Label(self, text = "Enter time in seconds: ",bg="gray88")
        time.pack(side=tk.LEFT,padx=10,pady=10)
        timeString = tk.StringVar()
        self.time = tk.Entry(self, width=8, textvariable=timeString)
        self.time.pack(side=tk.LEFT,padx=20,expand=True)
        button2 = tk.Button(self, text="drift rate",
                        command=lambda: [controller.show_frame(PageThree),self.find_drift_rate(controller)])

        button2.pack(side=tk.LEFT,padx=20,pady=10)

        button1 = tk.Button(self, text="back",
                        command=lambda: [controller.show_frame(PageTwo),self.on_cancel()])

        button1.pack(side=tk.LEFT,padx=5,pady=10,expand=True)

        self.driftRate = tk.Label(self, text = "",bg="gray88")
        self.driftRate.pack(side=tk.LEFT,expand=True)

    def find_drift_rate(self,controller):
        pageTwo = self.controller.get_page("PageTwo")
        function=pageTwo.drop3.get()
        time11=self.time.get()
        time11=float(time11)


        if function=="linear":
            def gaussian(x, a,b):
                return a*x+b



            pars,cov = curve_fit(f=gaussian, xdata=self.x1, ydata=self.y1, p0=[0, 0], bounds=(-np.inf, np.inf))

        elif function=="quadratic":
            def gaussian(x, a, b, c):
                return a*np.power(x,2)+b*x+c

            pars,cov = curve_fit(f=gaussian, xdata=self.x1, ydata=self.y1, p0=[0, 0, 0], bounds=(-np.inf, np.inf))

        elif function=="cubic":
            def gaussian(x, a, b, c,d):
                return a*np.power(x,3)+b*np.power(x,2)+c*x+d
            pars,cov = curve_fit(f=gaussian, xdata=self.x1, ydata=self.y1, p0=[0, 0, 0, 0], bounds=(-np.inf, np.inf))

        elif function=="polynomial":
            def gaussian(x, a, b, c,d,e):
                return a*np.power(x,4)+b*np.power(x,3)+c*np.power(x,2)+c*x+d
            pars,cov = curve_fit(f=gaussian, xdata=self.x1, ydata=self.y1, p0=[0, 0, 0, 0, 0], bounds=(-np.inf, np.inf))

```

```

        x= Symbol('x')
        rr=gaussian(x,*pars)
        #get drift rate
        drift_rate = rr.diff(x)
        new_df=np.abs(drift_rate.subs(x,time11))
        new_df2=round(new_df,4)

        self.driftRate.config(text="Drift rate(MHz/s):{}".format(new_df2))

if __name__ == "__main__":
    app = main()
    app.mainloop()
```