Web Interface for the MiloSAR

A web interface, dataset parser, processor and library for the miloSAR radar

Presented by:
Daniel David Zuckerman

Prepared for:
A. K. Mishra
Dept. of Electrical and Electronics Engineering
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science degree in Mechatronics.

September 13, 2021

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Signature:.....
D. D. Zuckerman

Date:.....

Acknowledgments

I would like to thank my supervisor, Amit Mishra, for his support and wisdom throughout this investigation. Replying to my emails promptly and lending advice and insight from the very beginning of development till the very end.

I would also like to thank my second supervisor, Darryn Jordan, for all the help he has given me throughout. Assisting me with the understandings of SAR radar technology and its datasets as well as walking me through any concepts I had difficulty with. He showed patience and understanding whenever I found myself struggling.

I am also grateful for my friends in the engineering department and adjacent for supporting me throughout my engineering studies. Most specifically, I would like to thank my friend, Jadon Wolffs, for tutoring me on web development and coding languages, and allowing me to consult with him.

Finally, I'd like to thank my parents. They have supported me throughout not only the development of this investigation but my entire university career. Thank you to my mother for sitting down with me to help proofread this report.

Abstract

The miloSAR is a testbed designed to implement a Synthetic Aperture Radar in a low cost and rapidly re-configurable way. The purpose of these radars is to provide important information about farmland and its surrounding areas to the agricultural industry. These SAR images provide incredibly useful data to the agricultural analytics market as the radar imagery allows for detection of biomass, plant identification and soil moisture content. Furthermore, digital elevation models and terrain displacement measurements can be made which can greatly assist any mining and hydrology studies. In order for this information to be useful to farmers and scientists, the results from a flight of the miloSAR require a certain level of processing. Therefore, a program is needed that can produce a web interface that is capable of parsing through miloSAR data sets for processing, process this data in a meaningful and useful way and then produce this processed data on the web interface in such a way that is easy to navigate and understand.

Therefore, a python program is created that could perform three major tasks. The program can parse through data and extract important information, it can process this information in important ways that show more information, and contains a web interface that allows a user to view said data and information as well as perform operations themselves to the data sets. The program takes raw miloSAR data and produces SAR images on an interactable map through a series of processes. It also uses other information regarding the flight as this information is also useful to the agricultural industry.

All code used and referenced in this report can be found under the GitHub Repository here: `https://github.com/Dzucky/EEE4022SThesis2020`
A video demonstration of the interface can be found here: `https://youtu.be/id_R3ozurJ4`

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background to the study

Synthetic Aperture Radar technology is primarily used to obtain images of landscapes. These images are incredibly useful to scientists and engineers in the agricultural industry as they provide important information of a large area of farmland. A company by the name of droneSAR (Pty) Ltd is developing a product called the radioCamera, a C-Band Synthetic Aperture Radar (SAR) that is primarily designed to be fitted onto a drone and to collect image data for this agricultural analysis sector [1] .
In order to test this radioCamera, a testbed was designed called the miloSAR testbed. Designed to be low-cost and easily re-configurable, the miloSAR was made from affordable, commercial S-band components. MiloSAR was named after the popular chocolate drink, Milo, as it uses Milo cans as antennae in its design as can be seen in figure. 1.1

The miloSAR is attached to a drone and flown over a predetermined plot of land. The testbed then captures radar imagery data and logs flight data such as the drones velocity and altitude throughout the recording as well as other important information. This data then needs to be processed so that scientists and engineers can easily view it. Thus, the miloSAR is in need of a method to display its image sets to clients as well as providing the other information in a comprehensive way. This requires a web interface to be created that parses through datasets, processing the data as needed, and displaying information that a user can interact with.

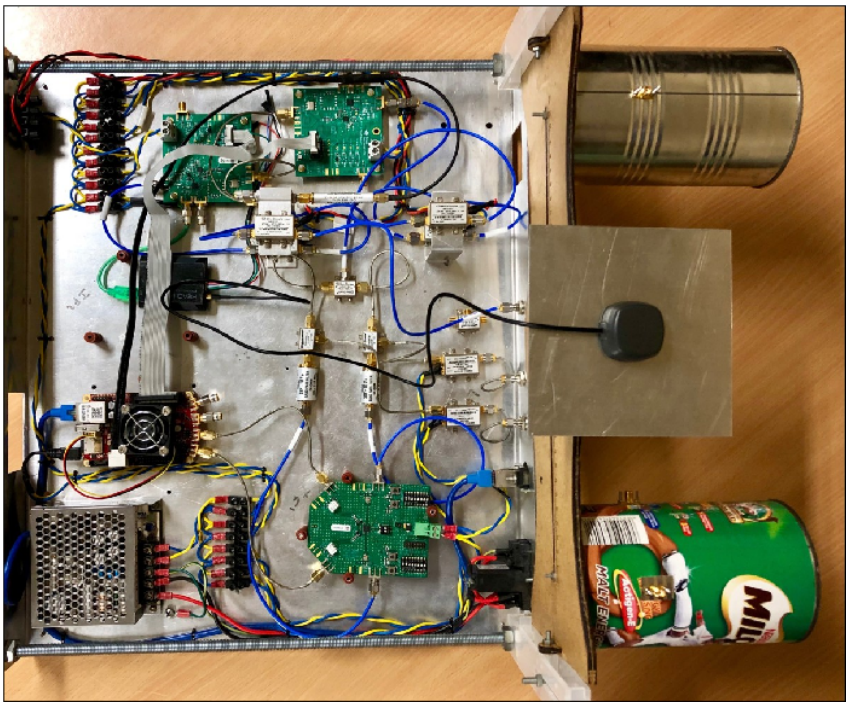


Figure 1.1: Aerial view of the miloSAR testbed radar.[1]

1.2 Objectives of this study

1.2.1 Problems to be investigated

The primary objective of this investigation is to design, develop and produce a web interface for the miloSAR and its datasets as well as overcome any challenges involved. This includes any problems associated with data parsing, data processing, and displaying this data in both an interactable and useful way to the user. The three user requirements given for the program are: Produce a web interface, acquire and display miloSAR datasets, and process this acquired data in a useful way. This requires identifying a python framework with which the web interface is built on and this is done through in-depth research.

Another major problem that needs to be solved in this investigation is that the program must have a method of taking a miloSAR image and merging it with a satellite map depending on its positional data. This requires the importing of libraries and toolboxes and therefore, research needs to be made to determine how this can be done.

1.2.2 Purpose of the study

The team behind miloSAR require a means of conveying the findings made by the miloSAR testbed radar to the scientists and engineers in the agricultural industry. This is a large target audience and therefore, a web interface that anyone can access and use to view the data for analysis is the most appropriate solution. Therefore, the purpose of this study is to deliver a fully functioning web interface that acts as a dataset parser, processor and library.

1.3 Scope and Limitations

The program was built around the assumption that the datasets were of a certain, predetermined file format as well as being specifically purposed to read and process SAR imagery. However, it was important to keep in mind from the start of development, the level of modularity of this program and how much work would be needed to re-purpose this program for other dataset formats or imagery from other sources such as satellites. This will be discussed in a later chapter. However, for the scope of this project and in terms of the requirements it needs to meet, the program will be developed for the miloSAR results which are in the form of SAR imagery and are received as one type of file format.
A limitation to this study is that the entire program, including all its processing, is run locally on a desktop computer, of which the specifications are laid out in Chapter 3.2.1. This means that often times, processing of large amounts of datasets is incredibly slow, hindering the response time of the web page as a whole.

1.4 Plan of development

The following report documents the development of this web interface for the miloSAR testbed. The following section contains a review of the relevant literature that was researched in order to design and develop such a program and provide greater understanding of the work laid out in this report. Following this section is the design and methodology which contain a review of the user requirements in order to break them down into functional requirements and then further down into specifications and develop acceptance test protocols. The chapter also describes the computational setup used to develop

the program as well as an explanation of every algorithm and process used in the web interface to fulfill the given tasks and meet the functional requirements. The next chapter will pertain to the results of the study. It contains screenshots of the interface's operation alongside explanations to demonstrate the program's implemented features. The discussion chapter then assesses the acceptance test protocols to see if all the requirements have been met and explores the relevance of the results to the overall investigation. The conclusions chapter then revisits the main purpose of the study to assess whether the program developed succeeds or fails to satisfy this purpose. The final chapter, Recommendations, explores any recommendations for any possible further development.

Chapter 2

Literature Review

This chapter assesses the literature relevant to this project. There are many ways to create a program that will perform the tasks required by this project and so, in order to decide on which methods are best, a great amount of research as well as an in-depth review of the options must be made. Furthermore, in order to understand the project fully, research must be done on synthetic aperture radar (SAR) technology as well as the miloSAR testbed. In order to meet the requirements set out, a python-based web interface needs to be created for the miloSAR radar that can act as a means to view processed SAR imagery as well as process and parse through datasets.

2.1 Existing Web Frameworks

There exist many tools and libraries that provide a framework to create a web interface in python and choosing which framework to use mainly consists of the extent of tasks needed to be done. In this review we will be looking at Django, Flask and Web2Py.

2.1.1 Django

Django is an open-source web framework for Python. It follows a mode-view-template architecture pattern which means it separates the information between three interconnected sections [8]. The advantage of using Django as a web interface is that the design philosophies surrounding Django emphasize reusability and reducing redundancy as well

as the ability for rapid development [5]. It also can provide a dynamically generated interface allowing an administrator to edit the webpage. Django is used by many popular websites including Instagram and Mozilla [7].

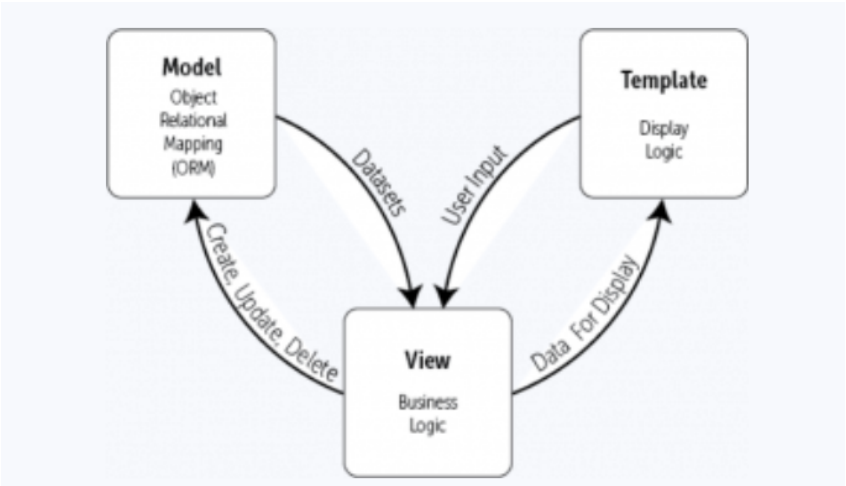


Figure 2.1: A flow diagram illustrating Django's architecture of mode, view and template. This figure shows how information flows in a Django web page.[8]

Django makes the web development process very straight forward and allows one to fully focus on the design process. However, when designing a web interface with Django, the entire web page needs to be based on the Django Object-Relational Mapper (ORM). The URL router in Django is more complicated than in other web frameworks and is regarded as very monolithic[6]. Django also requires a lot of research before being able to start production on a web page and requires the programmer to have a set out plan. This limits creativity and the person's ability to adapt their web page.

2.1.2 Flask

Flask is a web framework for python that does not require any specific tools or libraries. It does not include features such as form validation and database layers that other frameworks have. The advantage of using flask is that it supports extensions that can implement features as if it were part of flask itself [2]. Other popular websites that use flask are Pinterest [3] and LinkedIn [2]. Flask is specifically designed to be extended as the core framework is small compared to other frameworks but easier to understand. This classifies it as a 'microframework' [4].
Therefore, the advantage of using flask would be a greater degree of freedom when executing our vision in what we want our interface to do.
To sum up what flask is about, the creator of flask, Armin Ronacher, said this, "The idea of Flask is to build a good foundation for all applications. Everything else is up to you

or extensions." [2]. Flask is based on a utility library called Werkzeug and a template engine called Jinja which operates in a sandbox similar to Django. It is also compatible with many third party tools such as Google App Engine [4].

2.1.3 Web2Py

Web2Py is a full-stack web framework that focuses on three main goals: Ease of use, Rapid development and Security. This framework has everything included within it. It requires no extensions and includes a web server, database and a web-based IDE. Every function within web2py has an over-writable default behaviour and can automatically generate forms for data. It is also incredibly secure as the template language and database abstraction layer erase most of its vulnerabilities [9].
Web2Py, like Django, also uses the Model View Controller Pattern however, according to Web2Py's website, web2py differs from Django in its compactness and its ease of learning. However, according to slant.co, Django is ranked number 2 in the rankings for best backend framework while Web2Py is only ranked number 24. Flask is ranked number 5 on this list [10].
Unlike Django, Web2Py lacks a powerful ORM [11]. Therefore, things could get complicated when working with large models.

2.1.4 Template Engines

Interface controllers often render templates to generate the HTML necessary for a web page. Web2Py and Django use their own templating engine, but Flask, as well as Django, can use a template engine called Jinja2.

Jinja is a templating language that is designed to be fast and secure [12]. Jinja follows Python's principles while adding functionality that can be useful in templating environments such as for loops, if statements and variables. Not only can Jinja allow for variables to be passed through by the controller into the HTML code, but these variables can also be imported into the script tags using json, allowing for JavaScript manipulation of the variables. Using Jinja also allows for an easy method of creating HTML lists with variables using for loops and importing variables. Jinja distinguishes variables and instructions with the use of curly brackets.

Most significantly, Jinja is capable of template inheritance [12]. This allows a programmer

to build a base "parent" template that would contain common elements of the website shared between multiple different pages. It then defines code blocks that child templates can be inserted into and essentially extend the parent template. For examples of parent/child Jinja templates, see files layout.html and template.html under the GitHub repository.
Jinja also comes with other useful tools such as macros, filters and assignments.

2.1.5 Framework Used

After comparing each of these frameworks, the decided framework to proceed with is Flask. While being the only framework of the three to be a micro-framework, it allows the programmer an immense amount of freedom and flexibility. It is also one of the most used python web frameworks, meaning there are a lot of resources online to assist. Furthermore, Flask's higher level of modularity over the other frameworks is a great advantage for the task at hand as the required functionality could change throughout the course of development. Furthermore, the use of Jinja2 templates with Flask grant a great advantage in designing the web pages.

2.2 Data File Formats

An important part of the system is the data retrieval sub-system. However, this depends on what file format the data is contained in. We will now go into the different types of data files that can be used, and which are the most useful for this project.

2.2.1 Hierarchical Data Format 5 (HDF5)

HDF5 is a file format designed for storing and managing data. It also stores these hierarchical objects in a natural manner much like directories [13].
According to its website, the HDF5 was specifically designed for high volume and complex data, all sizes and types of system, flexible storage, and to be used as a file format toolkit. It also has incredible support for queries based on field matching such as searching for an attribute. However, it is not ideal for sequential processing of all records in the entire database or selecting based on co-ordinates.
What makes HDF5 unique is its ability to represent both complex data and useful metadata. They also allow the user to directly access parts of the file without having to

parse through the file's entire contents. HDF5's data model is primarily made up of two objects: a group and a dataset.

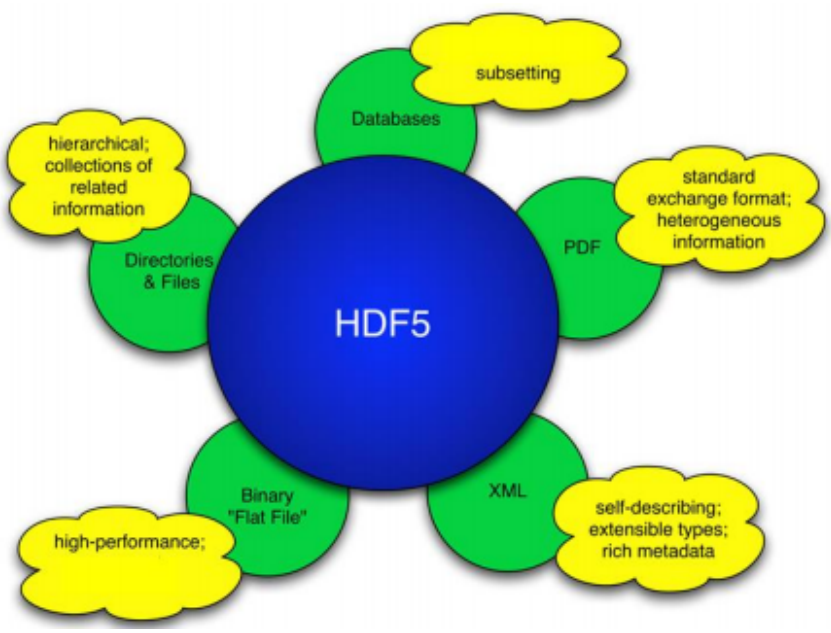


Figure 2.2: A diagram illustrating the types of datasets, and their related data, one HDF5 file is capable of containing.[13]

Datasets are a multidimensional array of data elements containing the actual data as well as supporting metadata that describes the data in the form of attributes. These elements can be many different types of data including images, graphs or even PDFs.
Groups are "folders" that can contain more groups or datasets. Groups also come with their own metadata. Groups can also share datasets with other groups, allowing for shared information between different data clusters.
The metadata can contain extra information about the data such as its data type, properties and attributes. Attributes allow for the parameters of the data to be easily accessed without having to actually view all the data. An example of the architecture of an HDF5 file is better illustrated in Figure 2.3 below.

2.2.2 JavaScript Object Notation (JSON)

JSON is a data file format that is based on a subset of the JavaScript programming language [14]. It is easy for humans to read and understand and easy for machines to parse through. Json files are built on two structures: An object (represented in the form of name/value pairs) and a list or array (An ordered list of values). Its basic data types are number, string, Boolean and null. Therefore, a Json file would not be able to include other data types such as PDFs or Images as the HDF5 file type can.

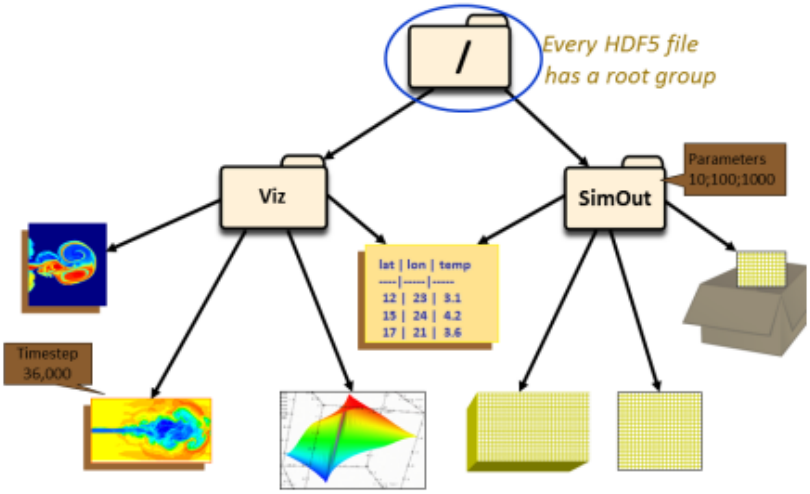


Figure 2.3: A diagram of the architecture of an HDF5 file, demonstrating two different groups sharing the same dataset as well as different file types within these groups. [13]

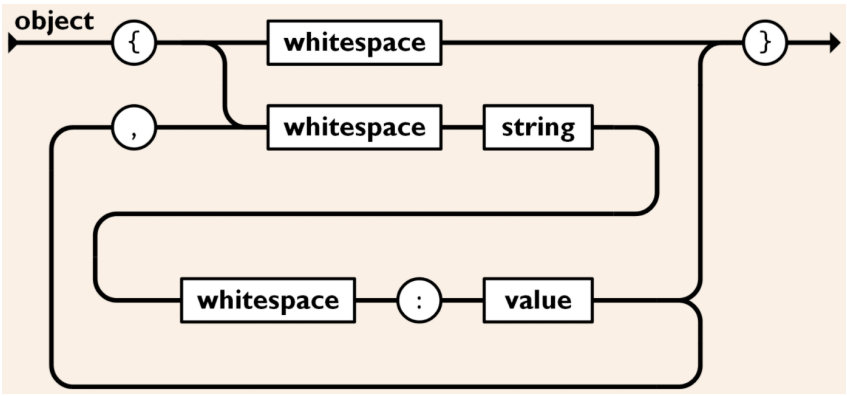


Figure 2.4: An object's format described within a json file. [14]

As mentioned earlier, the notation of the json files is very easy to understand. It follows the convention of: "parameter": value. This does mean that it is necessary to read through the entire file to view the parameters and their values.
JSON is a good choice if we had small amounts of uncomplex data. However, various Json parsing implementations have been known to suffer from denial-of-service attacks and JSON files are not suited if we need to combine data from different systems [15].

2.2.3 Yet Another Mark-up Language (YAML)

YAML, like JSON, is a file format that is easy for humans to understand and was designed to be accessible regardless of the programming language used [16]. According to YAML's website, yaml.org, the necessity for YAML to be extensible and easy to implement and

use are two of the developers' main design goals. YAML builds upon concepts from many other languages and formats including JSON. Like Python, YAML uses indentation to distinguish hierarchical levels. This assists in its easy interpretation to people however, this doesn't advantage the machine's interpretation in anyway. The difference between YAML and JSON lie in their priorities. JSON prioritises simplicity and being universal while YAML prioritises human readability. This makes JSON easy to generate and parse, but the process is more complex with YAML. YAML is essentially a superset of JSON as it is a more complete model.
It is important to note that YAML's indentation format can be seen as a hindrance by some developers and, compared to JSON and HDF, it is a lot younger and therefore, still contains some inconsistencies.

2.2.4 The Ideal File Format

Having researched the above formats, it is clear that HDF5 is the best suited and most interesting file format to use for this project. It's unique representation of metadata allows for parameters/attributes to be easily accessed and parsed along with the necessary images that the SAR radar will provide. The HDF5's structure of groups makes it incredibly easy to provide multiple datasets in one file and is useful for query searching based on parameters which is one of the functional requirements of the project.

2.3 Synthetic Aperture Radar Images (SAR)

Synthetic-Aperture radar (SAR) is a form of radar that is typically used to create 2-D images or 3-D reconstructions of objects like terrain and landscapes [1]. SAR is capable of remote sensing with a high degree of resolution regardless of weather, altitude and time of day. Synthetic-aperture radars are imaging radars that are mounted on a moving platform. Most of the time it is mounted onto a plane or drone to take images of the earth below. The radar transmits electromagnetic waves and the 'echoes' that bounce off the objects on the ground are recorded and digitized so that the data may be processed into an image. The way in which the signals are received at different positions creates a 'synthetic' aperture that is greater in width than the actual antenna [19]. 2-D images are generated by processing the azimuth and range data. Azimuth is the angle between a reference direction, usually north, and a line drawn from the observer to the point of interest projected onto the same plane as the reference direction. Once the 2-D image

is created, a 3-D model can further be generated by using a digital elevation model to determine height information. Thus, the third dimension is elevation. For this project, most if not all the imagery will be 2-D.

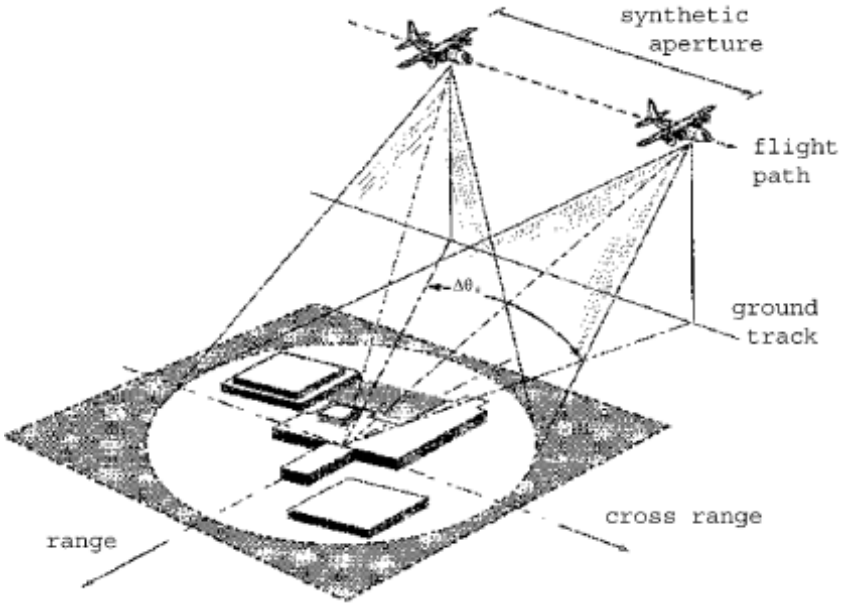


Figure 2.5: A diagram of an aircraft taking SAR readings. [19]

SAR images are used in many fields other than geology and topography. It is used in oceanography, glaciology, and volcano/earthquake monitoring. The SAR image of a radar campaign comes in the form of a linear power image (W). This needs to be converted into dBm in order to have a logarithmic image for breaking further breakdown of information. The method by which to do this is converting first to dB with the formula:

$$X = 20\log(P)$$

Where X is the decibel magnitude and P is the linear magnitude. Then using the formula:

$$dBm = dB + 30$$

This is necessary to create our SAR image and to create other informative graphs such as a histogram of magnitude pixel data [20]. This gives important information to the user with regards to the amount of SAR image being represented. Furthermore, the display of the image can be optimized by reprocessing the pixel data or changing the colour map of the data to gray scale, green scale or an image with multiple colour bands [20].

2.4 Google Earth Engine

Google earth engine is a program that allows visualization and scientific analysis of geographical datasets. According to the google earth engine website, earth engine's purpose is to provide an interactive platform for geo-spatial algorithm development, make substantial progress on global challenges involving large geographical datasets, and to enable high-impact, data-driven science [18].
Google earth engine's catalogue contains thousands of free to use geo-spatial datasets that can prove extremely useful for the purpose of this project. This library also includes Sentinel-1 SAR data that could be useful to compare with miloSAR results.
There already exist many resources on how one can access Earth Engine's API through python, as well as how to represent this on an interactive window using Folium [17].

2.5 Folium

Folium allows python processed data to be visualized on a leaflet map using leaflet.js, a JavaScript library [24]. Folium allows the program to generate an interactable satellite map of the world. With this map as a base layer to the leaflet, raster layers, such as a SAR image, can easily be overlaid using the raster layer's "ImageOverlay()" function. Folium also allows for a layer controller which can allow the user to toggle each layer's visibility on the map.
Folium does require a custom method of displaying earth engine tiles to a folium map as it has no default method of handling earth engine data. This is done by adding a custom layer to the folium.Map module. To see how this is done please look at the Design and Methodology section and the python file "ee_demo.py" found in the GitHub repository.
Most useful to this investigation however, is the ability for the folium map that is created to be represented in HTML format. This allows for a flask function to render the folium map inside a web browser [24].

2.6 Viewing HDF5 files

There exist many ways of viewing and creating an HDF5 file. The most suitable method is based on what the data is intended for.

2.6.1 HDFView

HDFView is a software tool produced by the HDFGroup, written in Java, that allows HDF5/4 files to be visually represented to the user [21]. The tool can browse through and edit the hierarchical tree of the files as well as its datasets and groups along with their accompanying metadata. The Graphical User Interface is simple and intuitive to use and makes for a suitable medium of displaying the HDF5 files to a human being in the form of text or an image. However, if one wanted to use this information for a program or process the data in any way, another method needs to be used.

2.6.2 HDF Programming model

The HDF5 library is programmed in C as it contains useful mechanisms such as using objects as data structures [22]. However, the library uses its own pointers or identifiers to call operations to be made on a specific instance of an object. This method is shared with other object-oriented languages however, the syntax is shared with C. The programming model allows a user to create and manipulate HDF5 files in C. Operations include creating an HDF5 file, writing to or reading from a dataset, defining compound data types as well as many other operations. Since C objects do not collect all methods for an object under one name space, the HDF5 library's API simulates this by using a common name prefix corresponding to the class of object that the function operates on. The API naming scheme is shown below in Table 2.1.

Prefix	**Operates On**
H5A	Attributes
H5D	Datasets
H5E	Error reports
H5F	Files
H5G	Groups
H5I	Identifiers
H5L	Links
H5O	Objects
H5P	Property lists
H5R	References
H5S	Dataspaces
H5T	Datatypes
H5Z	Filters

Table 2.1: Table of the HDF5 library's API naming schemes and their object classes. [22]

2.6.3 H5py

H5py is a package for python that allows python programs to interface with the hdf5 format of data files. The main advantage of using this package is that it makes working with the datasets much easier as it allows the data to be stored and manipulated in the same way as a NumPy array with all the same syntax [23]. Most importantly, files created with h5py are made in a standard binary format that is used by many libraries and languages. This allows other people using other programs to use these files as well. The H5py library allows all of the operations capable with the HDF programming model to be done in python. Therefore, the HDF5 files to be used, that contain groups and datasets as well as attributes could be created within a python program and could then be read within the flask controller program thanks to this H5py package.

Chapter 3

Design and Methodology

3.1 Requirements Analysis

The study being made is for a web interface that needs to fulfill certain functions described in the form of user requirements. Therefore, an analysis needs to be made on every user requirement so that each of them can be broken down into functional requirements.Those can then be further broken down into design specifications. Finally, acceptance test protocols can be derived to make sure all the design specifications and in turn, the user requirements are being met.

3.1.1 User Requirements

UR1-0-001	**Produce a web interface for the miloSAR.**
Requirement	The system must be represented by a web interface that is easy to understand and navigate and that the user solely interacts.
Rationale	The main deliverable for this project is to produce a web interface for the miloSAR radar that a person could use in order to see information pertaining to the miloSAR's readings and flights. This is so that a better understanding can be made of the surrounding area's agricultural and geographical features.
Refined By	FR1-1-001 FR1-1-002 FR1-2-005
Verification	Requirement is verified by demonstration of the web interface functioning as intended by an outside user.

UR1-0-002	**Acquire datasets of the miloSAR's flights and display them.**
Requirement	The system shall be able to acquire datasets of information as well as display them as the user requires.
Rationale	The information required for the interface will be stored separately as a series of files containing datasets. The program needs to retrieve these files and interpret the data within them so that the necessary information can be displayed.
Refined By	FR1-2-001 FR1-2-002 FR1-2-003 FR1-2-004 FR1-2-005
Verification	Requirement is verified by demonstration of the interface's ability to read and display the datasets correctly.

UR1-0-003	**Process data in a useful way for the user.**
Requirement	The program should process some of the data in a useful way for the user such that more information is available or that the information is represented in a different way.
Rationale	The interface needs to produce interactable graphs and images. In order to produce these, the program needs to process the data given from the datasets and perform some calculation to produce results that can be useful to the user.
Refined By	FR1-3-001 FR1-3-002
Verification	Requirement is verified by demonstration of these features.

Table 3.1: The user requirements of the interface. These are broad requirements that need to be expanded on in order to be met.

3.1.2 Functional Requirements

Multiple functional requirements can be derived from the main user requirements. The functional requirements are more specific requirements of what the program needs to be able to accomplish.

FR1-1-001	**miloSAR about page.**
Requirement	The web interface will contain a section that will briefly describe the miloSAR, what it does, how it works and any other important information.
Refines	UR1-0-001 (Web Interface) states that the web interface needs to be a representation of the information obtained by the miloSAR that will help the user gain a better understanding. By having an about page for the miloSAR, a user can better understand the equipment used and the results obtained.
Refined By	DS1-1-001
Verification	Requirement is verified by inspection.

FR1-1-002	**Interactable web page.**
Requirement	The web interface should contain buttons, combo boxes, drop-down menus and hyperlinks that allow the user to interact with the data and navigate the interface.
Refines	UR1-0-001 (Web Interface) requires the user to be able to interact with the interface so that the user can navigate the page and see the desired information. This will be done through the use of interactable GUI components such as buttons and drop-down menus.
Refined By	DS1-1-002 DS1-1-003 DS1-3-001
Verification	Requirement is verified by demonstration.

FR1-2-001	**Interpret Data.**
Requirement	The program must interpret offsite files containing the datasets of information so that they can be displayed with their appropriate attributes and can be processed.
Refines	UR1-0-002 (Acquire and Display data) requires the program to interpret the data files to a certain extent so that the appropriate information can be extracted and displayed correctly.
Refined By	DS1-2-001
Verification	Requirement is verified by demonstration.

FR1-2-002	**Display SAR Imaging Information.**
Requirement	The program must display the information captured by the miloSAR radar in the form of images, and text according to the attributes and information obtained from the interpreted data.
Refines	UR1-0-002 (Acquire and Display data) requires that the information captured by the miloSAR, and then interpreted from the files, needs to be displayed appropriately so that the user can fully understand it.
Refined By	DS1-1-003 DS1-2-002
Verification	Requirement is verified by demonstration.

FR1-2-003	**Display miloSAR flight information.**
Requirement	The program will also display information pertaining to the flight of the miloSAR reading as part of the dataset that is interpreted.
Refines	UR1-0-002 (Acquire and Display data) states that all necessary data be available for the user. The flight information is useful and important to the user.
Refined By	DS1-2-003
Verification	Requirement is verified by demonstration.

FR1-2-004	**Display Multiple Datasets.**
Requirement	The program shall be able to display multiple datasets so that the user can compare and view multiple images at the same time.
Refines	UR1-0-002 (Acquire and Display data). The ability to view multiple datasets at once is incredibly useful as it allows the user to compare different images and different areas of the map at the same time.
Refined By	DS1-2-004
Verification	Requirement is verified by demonstration.

FR1-2-005	**Query Search by Attributes.**
Requirement	The program will allow the user to search for a specific dataset by searching the dataset's name.
Refines	UR1-0-001 (Web Interface) states that the user needs to be able to interact with the web page and allowing them to query search gives the user the ability to refine what information they want to see. UR1-0-002 (Acquire and Display data). This function requires the program to interpret the data such that the parameters of the information is available such as the dataset's timestamp name.
Refined By	DS1-2-005
Verification	Requirement is verified by demonstration.

FR1-3-001	**Image Processing.**
Requirement	The program must process a SAR image so that additional information can be viewed in an easily interactable way.
Refines	UR1-0-003 (Process Data) requires that the program process the data in some way. By processing data from a SAR image, the user can view information related to the image such as azimuth and range graphs at certain locations on the image.
Refined By	DS1-3-001
Verification	Requirement is verified by demonstration.

FR1-3-002	**Graphs.**
Requirement	The program will process data in order to produce graphs that can better convey the data to the user.
Refines	UR1-0-003 (Process Data) states that the program should process the data to be more useful to the user. By introducing graphs made from flight path and radar parameters.
Refined By	DS1-3-002 DS1-3-003
Verification	Requirement is verified by inspection.

Table 3.2: Tables of the functional requirements of the system. Derived from the user requirements.

3.1.3 Specifications

The design specifications are the specific features the program will have in order to fulfill the functional requirements previously defined.

DS1-1-001	**Link to miloSAR about page.**
Requirement	The web page will contain a link that will redirect the user to a page consisting of a short description of the miloSAR radar and how it works as well as some images.
Refines	FR1-1-001 (miloSAR about page) requires there to be a section somewhere within the interface that contains information about the miloSAR radar.
Verification	Requirement is verified by demonstration.

DS1-1-002	**Coding Languages.**
Requirement	The web interface will be made using python. Specifically, Flask as a web framework along with HTML, css and JavaScript in order to create the web interface.
Refines	FR1-1-002 (Interactable to the user) states that the project's deliverable needs to be a web interface. This can be created using the Flask web framework in python which renders Jinja2 HTML templates.
Verification	Requirement is verified by inspection.

DS1-1-003	**Folium Map.**
Requirement	The program will use the google earth engine to provide satellite images of the same location as the SAR image so that they can be overlaid onto each other for better context. These images will be placed on a folium map which will be interactable in a similar way to google maps.
Refines	FR1-1-002 (Interactable to the user) requires the web interface to have some interaction available and the folium library provides such interaction in a familiar way as most people have used google maps. FR1-2-002 (Display SAR imaging Information) requires the program to display the SAR radar image. The earth engine assists in conveying context about the SAR radar image to the user and provides a better understanding of what the user is looking at.
Verification	Requirement is verified by demonstration.

DS1-2-001	**HDF5 File Format.**
Requirement	The file format used will be HDF5 as this format is a hierarchical data format that can contain the datasets required as well as the parameters needed, and these are easy to obtain using python.
Refines	FR1-2-001 (Interpret Data) requires the program to have some way of reading data from a file and interpreting it in such a way that it can be conveyed to the user. By using HDF5, the program can read the parameters of the dataset and dynamically display the information depending on the unique datasets.
Verification	Requirement is verified by demonstration.

DS1-2-002	**Displaying Image Information.**
Requirement	The program will display the SAR image overlaid onto a folium interactive map. Furthermore, the information relating to the radar image will be displayed alongside it, correctly labelled corresponding to the parameters given by the HDF5 file.
Refines	FR1-2-002 (Display SAR Imaging Information) requires that the web interface displays miloSAR radar images as well as the information pertaining to those images.
Verification	Requirement is verified by demonstration.

DS1-2-003	**Displays Flight Information.**
Requirement	The program will contain a link to a page containing information on the miloSAR's flight for the selected dataset's campaign based on parameters given by the HDF5 file.
Refines	FR1-2-003 (Display miloSAR flight information) requires that the program shows flight data relating to when the radar image was captured and that that information be available to the user.
Verification	Requirement is verified by demonstration.

DS1-2-004	**View Multiple datasets.**
Requirement	The program will have the ability to select to view more than one dataset at a time. This will mean both SAR images will be displayed on the interactable map at once.
Refines	FR1-2-004 (Display Multiple Datasets) requires the program to allow the user to view multiple datasets at the same time so that the user can better compare images and determine which land is more useful.
Verification	Requirement is verified by demonstration.

DS1-2-005	**Query Search.**
Requirement	The program will include a feature that will allow the user to search for a dataset by its name. The user will be able to search for a specific dataset and select it from the menu.
Refines	FR1-2-005 (Query Search by Attributes) states that the program should allow the user to search for specific datasets via a query.
Verification	Requirement is verified by demonstration.

DS1-3-001	**Image Compare Feature.**
Requirement	The program will include a feature that processes the information displayed in a radar image. It will display the azimuth and range over power graphs depending on where on the image the user selects.
Refines	FR1-1-002 (Interactable web page) Requires the web interface to be interactable by the user. This feature allows the user to further interact with the images and graphs on the interface. FR1-3-001 (Image Processing) requires that the radar image is processed in some way to convey extra information. This program will do that by allowing the user to view the azimuth vs normalized power and range vs normalized power graphs depending on where on the image the user has selected.
Verification	Requirement is verified by demonstration.

DS1-3-002	**Graphs.**
Requirement	The program will produce velocity, altitude and Q-point graphs from the flight data as well as the normalised power vs azimuth and range vs normalized power graphs from the SAR radar imaging data.
Refines	FR1-3-002 (Graphs) requires the program to produce some information graphically so that the user can better understand the information. This requires some form of processing to be done by the program.
Verification	Requirement is verified by inspection.

DS1-3-003	**Chartjs.**
Requirement	The program will make use of the JavaScript library, Chartjs and python libraries such as numpy and matplotlib in order to properly display these graphs from the web controller to the template.
Refines	FR1-3-002 (Graphs) requires the interface to produce graphs. These need to be done using libraries available.
Verification	Requirement is verified by inspection.

Table 3.3: Tables of the design specifications which have been refined from the functional requirements

3.1.4 Acceptance Test Protocols

ATP	Test Description.
UR1-0-001:FR1-1-001:DS1-1-001	Verify that there is information somewhere within the web interface that consists of general information about the miloSAR radar as well as images of the radar itself.
UR1-0-001:FR1-1-002:DS1-1-002	Verify that the web interface has been made using the correct languages by inspecting the web page as well as the program's code.
UR1-0-001:FR1-1-002:DS1-1-003	Ensure that the folium map window is interactable and that the selected SAR image will align correctly with the co-ordinates of the satellite map as well as successfully inserting satellite imagery from the Google Earth Engine database.
UR1-0-002:FR1-2-001:DS1-2-001	Ensure that all datasets are available for the user to select and verify that the selected dataset displays the correct information.
UR1-0-002:FR1-2-002:DS1-2-002	Verify that the SAR imagery is displayed within the folium window and that the information displayed as well as their parameters match the dataset selected.
UR1-0-002:FR1-2-003:DS1-2-003	Verify that the flight information is being displayed correctly and is easily viewed by the user.
UR1-0-002:FR1-2-004:DS1-2-004	Ensure that the interface displays multiple SAR images on the folium map at the same time.
UR1-0-002:FR1-2-005:DS1-2-005	Ensure that the search feature retrieves the correct datasets when typing into the search bar of the dataset menu.
UR1-0-003:FR1-3-001:DS1-3-001	Ensure that the graphs produced are appropriate representation of the area selected on the SAR image as well as updating when the user selects another area.
UR1-0-003:FR1-3-002:DS1-3-002	Verify that graphs are being produced of the information regarding flight data and SAR radar imaging.
UR1-0-003:FR1-3-002:DS1-3-003	Verify that the appropriate libraries are being used by inspecting the code of the program and making sure that the libraries stated are in fact being used correctly.

Table 3.4: A table of the acceptance test protocols of the system

3.2 Subsystem Development

It was decided, for the purpose of modularity and ease of design, that the program's conceptual design should be split up into three subsystems. The project is divided into three subsystems: The user interface, Data acquisition and Data Processing. The design of how these subsystems interact with one another and what they encompass is illustrated in the Object Process Methodology (OPM) diagram shown below as Figure 3.1.

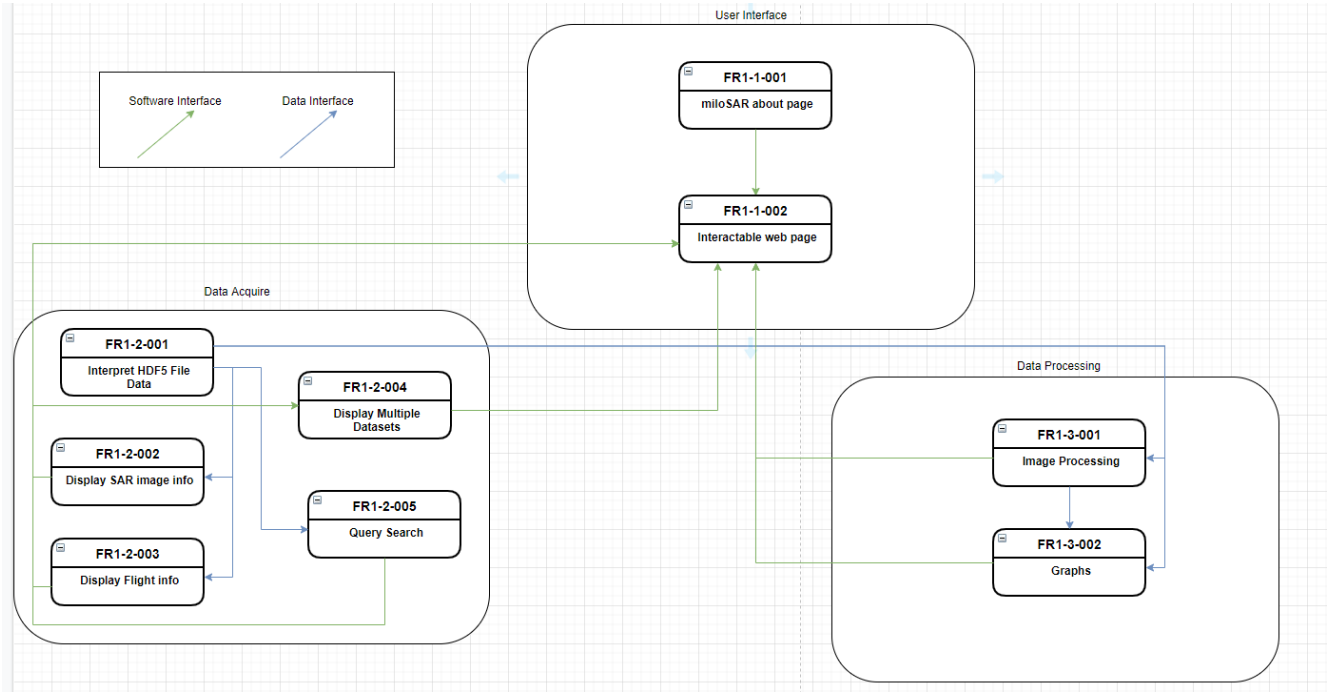


Figure 3.1: An OPM diagram showing the design layout of the program with its three subsystems.

3.3 Computational Setup

The web page is run from my home desktop computer. The hardware specifications of this computer is as follows:

- RAM: 8.00 GB
- Processor: AMD FX(tm)-8350 Eight-Core Processor 4.00 GHz
- Graphics card: NVIDIA Geforce GTX 1060 6 GB

- Operating System: Windows 10 64-bit
- Storage: 250 GB Solid State Disk

3.4 Extracting information

The datesets produced by the miloSAR are in the form of multiple files of different file types. Therefore, the relevant information needs to be extracted from the various files and compiled into an HDF5 file for the interface to read. In order to do this, a python program was written and can be found within the GitHub repository as "hdf5.py".
The files produced from each miloSAR campaign contain many parameters, some of which are not relevant to the study. Therefore, the variables shown in lines 155-188 of python file are the only variables that were necessary to capture. The program accesses and captures data from:

- a cmd file (lines 49-64). The cmd file contains parameters in the form of "parameter => value". Therefore, the algorithm searches line by line for the desired keyword value, finds the location of the "=>" characters, and reads the remainder of the line after that "arrow" as the variable's value.
- a log file (lines 66-81). The log file was simple text containing the parameters in the format of "parameter: value". When opened in python as a text file, the "\n" characters that signify a new line are visible. Therefore, the file needs to be split on the "\n" character which will essentially split the file up by lines. Then the program uses a similar approach as with the cmd file, only this time it is finding the position of the 'colon' in the line instead of the 'arrow'.
- a json file (lines 83-98). The python library, json, allows python programs to read json files with ease. It allows for the program to easily search each line for the required keyword and append the value to an array. The "ns" variable only needs to be taken from the first line as it is repeated as the same value throughout the flight.
- a geoJson file (lines 100-105). The geoJson file is another json file renamed to geojson as it contains the top left and bottom right co-ordinates of the SAR image. Therefore, the method used to capture this data was the same as the json file before.
- summary.ini (lines 109-123). The summary.ini file was read using an imported library called configparser. Configparser allows for the program to read the ini file

and search for specific parameters and their values without having to parse through strings.

- the image bin file (line 126). The bin file is a complex64 array and needs to be read as such using numpy.
- a txt file (lines 131-146). The text file contains the data in the form of html code. Therefore, the use of the package, BeautifulSoup, was most efficient to easily capture the data.

Once every necessary parameter is saved under a variable, the program creates a hdf5 file using h5py (line 150). It then creates a group for the image to be stored under in case any future files may be used with more than one image (line 151). It then writes the image file in binary format under the group (line 152) and proceeds to set the attributes of the bin dataset with the previously compiled parameters (lines 155-188).

3.5 Designing the Web Interface

3.5.1 Image Processing

The SAR image that is represented by the complex64 numpy and that is found in the newly created hdf5 file needs a level of processing done so that the final, observable image can be viewed and so that other information such as azimuth and range values can be extracted. In order to do this, a function was created called generatePNG and saved in an external python file called process_to_image.py. This file can be found inside the GitHub repository and the algorithm's flow chart can be seen below, Figure 3.2.

As can be seen in figure 3.2 and the python file, the function accepts input parameters of an image array, the file name, the number of rows or in other words, the number of range bins to process, the number of columns or the number of pulses, the azimuth and range dimensions, and the dynamic range.
The program starts by creating two arrays of range and azimuth points ranging from 0 to their maximum value with the amount of points in-between being the number of rows and number of columns respectively (lines 10 and 11). The program then reshapes the image array from a 1 dimensional array of numbers to an array that matches the number of rows and columns given (line 13). The program then converts the linear array into dB (line 14) and then to dBm (line 15). It then clips the image to the dynamic range. Lines

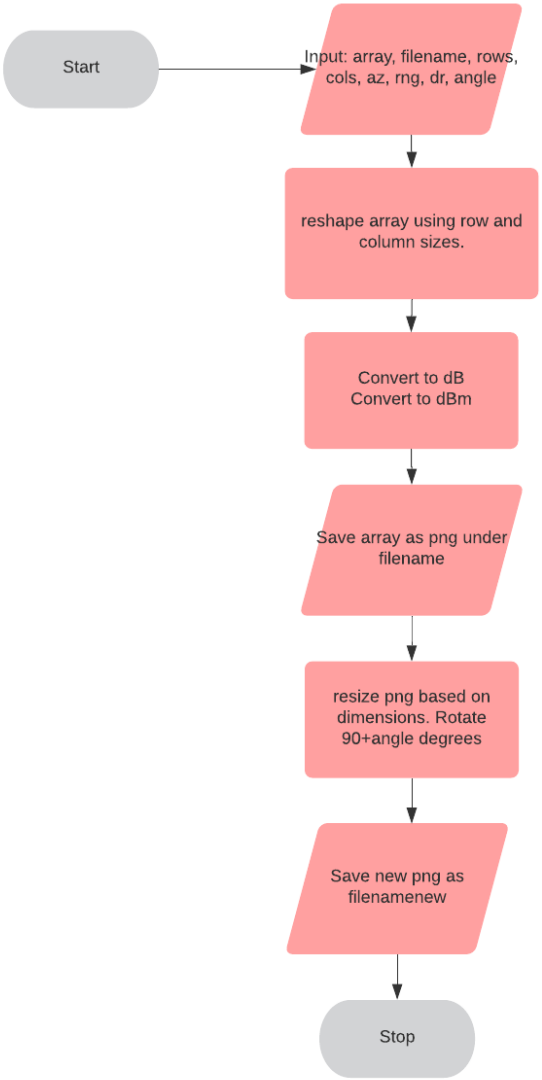


Figure 3.2: A flow diagram showing the logical operation of the algorithm used to save a png image from the SAR image array.

20-24 save the newly processed image as a png file under the filename given, which will be the name of the dataset, using Matplotlib's imsave() function. The program then uses the pillow library to resize the image to the dimensions given and rotate the image based on the angle given as the course attribute, which is the flights average heading during the campaign. It then saves this corrected image as a separate file.

3.5.2 Generating the Map

Once the image is saved as a PNG file, the program then needs an algorithm that will take a PNG picture and overlay it over a map. The function, generate_map, was created and can be viewed from the GitHub repository under "ee_demo.py". The logic of this

function's algorithm is shown below in Figure 3.3. Also included in this python file is the function, generate_map_2, which would add two images as separate layers to the map to allow for a potential comparison to be made.

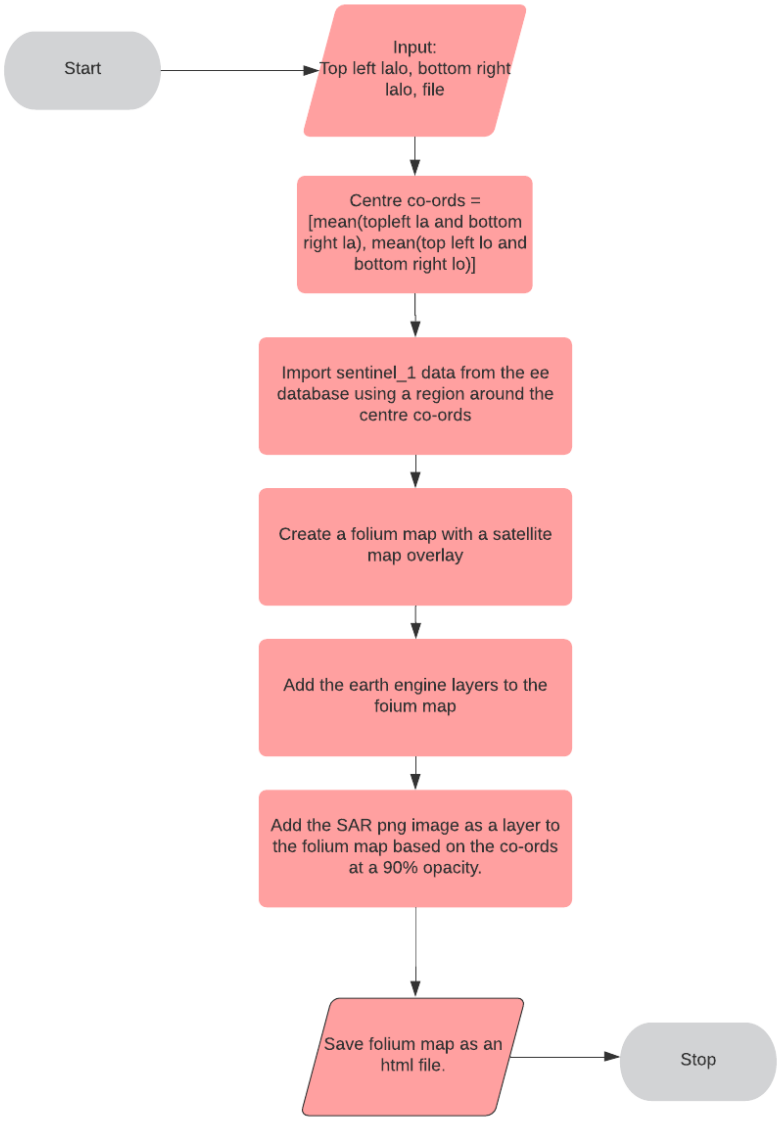


Figure 3.3: A flow diagram showing the logical operation of the algorithm used to create a map containing the SAR image and Earth Engine data.

This function accepts the latitude and longitude co-ordinates of the top left and bottom right corners of the image as well as the name of the dataset. The process of the function is as follows. The folium library is used to create the map on which these SAR images are placed. However, folium does not have a default method of adding tiles from Google Earth Engine. Therefore, lines 11-20 define a custom method of adding these tiles to the folium map followed by line 22 which adds this function to the folium.Map module. The program then finds the centre co-ordinates of the image by calculating the mean of the

two latitude co-ordinates and the mean of the two longitude co-ordinates (lines 25-28). Then the program will import Google Earth Engine data from the Sentinel 1 satellite in a buffered region around the centre of the image. The program takes both single and dual polarisation images filtered within the month of November 2018. All of this can be seen in lines 30-45. Line 47 creates the folium map with the starting location set to the centre co-ordinates. Lines 50-59 add the standard optical map of the world as a tile layer to the folium map. This provides a 'base layer' for our SAR image and Earth Engine data to be placed over. Then, lines 62 and 63 add the sentinal data to the folium map using the previously defined function. Lines 66-72 add the png file, based on the inputted file name, to the folium map using the given co-ordinates as bounds with an opacity of 0.9 so that the optical image of the landscape can also be seen behind it. Line 75 adds a control option to the map which allows users to toggle visibility on each layer and finally, line 77 saves this folium map as an html file so that the flask web page may view it as an interactable map. The second function present in "ee_demo.py", generate_map_2, works the same as the first function, however, it accepts an extra set of co-ordinates and a file name as inputs and adds a second SAR image to the map with those extra parameters so that a user may view two SAR images on the same map simultaneously.

It was intentional for the program to save the html map under one specific name. This way, whenever a new map needs to be generated, with a different image, the program will write over the old map. This was intentionally done so that the Flask web page can make use of an iframe with a static location URL and that whenever the map is changed, the iframe merely refreshes within the web page and thus, the entire web page does not need to refresh every time a new dataset is selected or a process is made to change the image.

3.5.3 The Flask Controller

The Flask controller is where all of the processing needs to be done. The algorithm of the controller follows the flow chart shown below in figure 3.4. What the controller does is contain each app route or page of the website which individually renders an html document using jinja2 and passes all the necessary information through to the templates with the use of variables. It handles all the processing needed to complete the functions of the web page by directing to app routes that do not render a new template, essentially running processes in the background of the interface. This allows for the script blocks in the html files to call functions and receive information using ajax requests. All of the code of the Flask controller can be found in in the GitHub repository by the file name,

"controller.py".

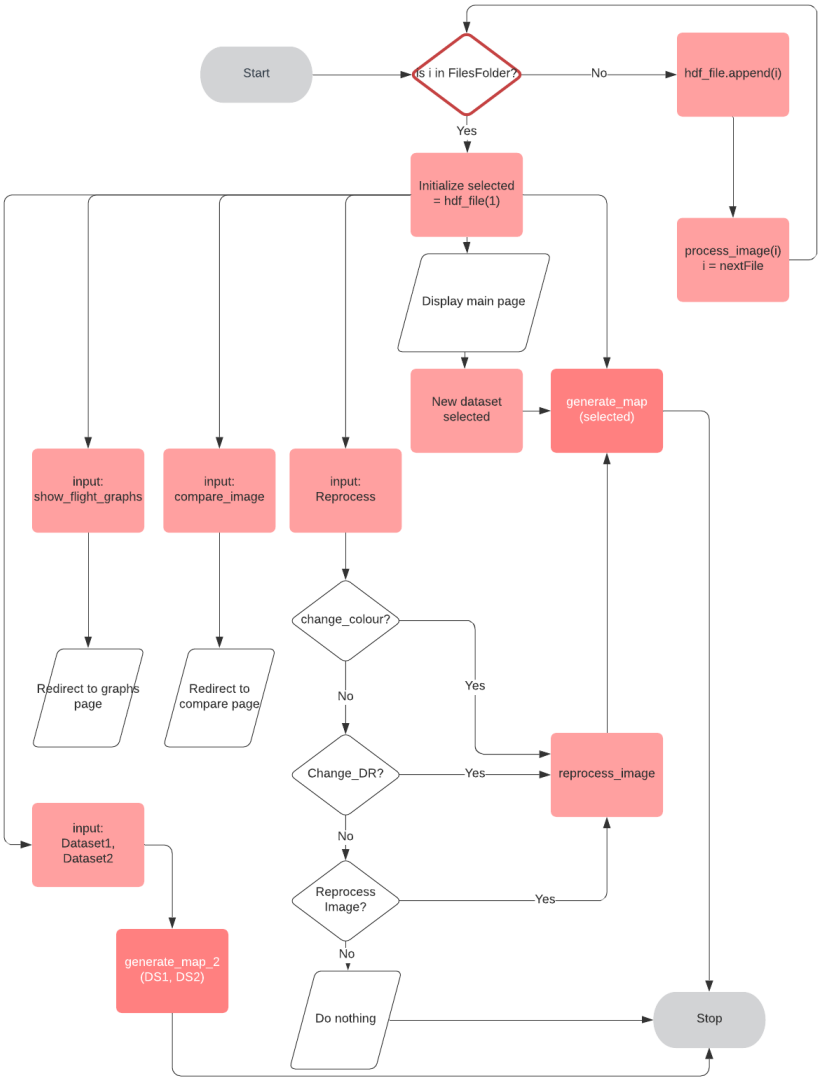


Figure 3.4: A flow diagram showing the logical operation of the flask controller when it is first run as well as the functions that can be inputted by the user.

When the controller first runs, it reads through every hdf5 file in the designated folder, "datafiles". It reads the bin array containing all the image information as well as every attribute and its values. The controller also reads specific attributes' values as these will be useful to the program later and the h5py function, "Get", is a much easier method to acquire these values rather than using for loops to run through every attribute in the list after the file is closed. After that, all the read values are appended to a list. The program then directs the user to the default app route which is the main page.

3.5.4 The Main Page

The main page is rendered by the Jinja2 template.html template file which extends the parent template layout.html. Both of these files can be viewed under the GitHub repository. The layout.html template is used in every page and contains all the css that the child templates use as well as a navigation bar which contains links directing the user to the main page, compare page and about page. The main page contains a menu of the datasets available, allowing the user to choose which dataset to view as well as being able to search by name for a specific dataset. It also contains the interactable map, the parameters of the dataset selected as well as some functions such as reprocessing the SAR image, viewing two images on the same map, and displaying flight graphs. When the controller renders this html, it sends the template the URL for the folium map's src, a list containing all attributes of every dataset as well as a list of all the values of those attributes, the title of the page as well as a list of the dataset names, and the first name in the list.
The template starts by generating the datasets menu by using a Jinja2 for loop and populating a list with the names of the datasets. Also generated is an iframe, which will load the html of a file given to it. In this case the iframe will load the generated folium map. Also generated are an input text box and a button to change the process by which the image is generated, along with this text box is a tooltip that tells the user what functions are available to them when the user hovers over the area. Also included are two combo boxes and another button allowing for the viewing of two dataset images at once as well as every attribute of the dataset and their values inside uneditable text areas. It is important to note that all attributes and values are generated within the page with their display hidden. The styling of the menu and other html objects can be seen under the css of the parent html document, layout.html.

Each list item also contains an action that calls the JavaScript function "reload()" when the item is clicked. The reload function is also called when the web page is first rendered. If the window's value is not set, the function will be called with the first dataset name in the list. When reload is called, the selected item's name is also sent as a parameter. The function then uses ajax to send the selected dataset name to the flask app route, "backgroundreload". Backgroundreload contains the function update(). This function receives the dataset name selected, finds its position within the names array so that it may have the index of the selected dataset, and then uses the generate_map function to generate the folium map with the selected dataset's image and co-ordinates.
Once the update function has completed, the reload function runs the code under ".always" which only triggers once the python function has completed its process. The JavaScript

program creates a tail of random numbers and replaces the URL that the src of the map iframe points to. This essentially forces a refresh of the iframe so that the new map shows on the page without the entire page refreshing.
The function then goes through every list item in the menu and removes the active class from the item, thereby removing the "highlight" and essentially deselecting the item. Then it finds the list item that was selected by matching the element ID with the input name and sets that item's class to active, which highlights the item in the menu. After doing this the script sets the window.value to current which will make the selected dataset a globally available value. Finally, the function gets the index of the selected item in the list and uses that index to set the display of the list of attributes and values relevant to the dataset selected to visible while making all other attribute lists invisible.
Also present in the JavaScript code is a querySearch() function which allows the user to search for a specific dataset by name. The function loops through every list item in the menu and hides the items that do not match the query letters.

Changing the image process

Upon clicking the "Execute command" button, the program will execute the ChangeProcess() JavaScript function. This function captures the "command" that is inputted into the input element and splits it up into the command and the parameters by splicing on the parentheses. It then sends these variables along with the window.value to a flask app route "processCommand" using the same ajax method as in the reload function.
The flow of this algorithm is shown above in Figure 3.4.

The processCommand app route contains a function called processCommand(). This function receives the command, parameters and the window.value. The program then finds which dataset is selected by finding the index of the window.value within the names array. The program then does an operation based on what the command string is:

- If the command is "setColourMap", the processPNG function is called. This function is exactly the same as the generatePNG function found in "process_to_image.py", however, the imsave() function saves the image with the inputted colour map specified as the command's parameter. Therefore, the user can input any recognized Matplotlib colour map as an input parameter to the function.
- If the command is "gaussian", the program opens the appropriate png image and performs the gaussian function found in the skimage library. This performs a gaussian sweep over the image with a standard deviation specified in the command's

parameter. The program then overwrites that image with the new gaussian image.

- If the command is "changeDR", the program runs the generatePNG function using all the same parameters as when the file was originally processed however, now the dynamic range is inputted as a different set of values given by the command's parameters.
- If the command is "originalImage", the program calls the generatePNG function with all the parameters that were given in the HDF5 file.

Once the python function is complete, the javascript function calls the reload function again so that the new map is generated and the iframe is updated.

Viewing multiple images

When the "View Both Images" button is pressed, the script executes the generate2() function which captures the selected values in the combo boxes, which are both populated with the same list of datasets as the menu is. It then sends these names to the Flask controller under the app route, gen2maps, using ajax.
The app route contains the python function gen2maps() which receives the two dataset names and finds their indexes within the names array, then runs the generate_map_2 function with their appropriate co-ordinates which can be found in "ee_demo.py" and whose operation is described in Section 3.5.2.

Viewing the graphs

When the "Show Graphs" button is pressed, the user is redirected to a different app route in the flask controller, "graphs". Graphs finds the appropriate altitude, velocity and change in time values based on the index of the dataset selected and creates an array of time values based on the time steps given. The function then renders the graphs.html file, found in the GitHub Repository, and sends these arrays as lists to the template.
The template generates three canvas elements, each one representing a different graph. The JavaScript then uses json to import the lists passed through by the controller. scatter and line graphs require the datasets to be in a specific format. Each point of the dataset needs to be of the form "x: xvalue, y: yvalue". However, the x values and y values are in separate lists at the moment. Therefore, the datarize() function accepts two lists and combines them, taking each value in the x and y lists and combining them into the proper

format and then appending it into a new list. Assuming the x and y lists are of equal length, once the for loop has gone through every value in the lists, the function returns the new combined list. The script then uses chartJS to create line graphs with the lists. When creating a chart, the first parameter is the ID of the canvas element that the chart is being created on. The second parameter contains the type, which specifies the type of graph such as bar, line, scatter, and others. Also within the second parameter are the datasets. Each dataset requires a label, the list of the data, and optional parameters such as the properties of the points of data. The final aspect of the second parameter is options. This specifies any other properties required for the graph such as the x and y axes' labels and scales. This function is used three times to create the three graphs with velocity represented as a blue line, altitude as green and Q-points as black.

3.5.5 Compare Image

The interface contains a web page that will allow the user to analyze, in-depth, the SAR image of their choice. This web page allows the user to select slices of the image and view its range and azimuth values.
When the user clicks on the compare image link in the navigation bar, the user is redirected to the "compare" app route, containing the compare function in the Flask controller. The controller then finds the index of the selected dataset name and does the same series of processing and operations as the generate_PNG function from process_to_image.py. This is so that the program has the full processed array of the image which is then saved as a global array. Then the initial indices for the axes are found so that the program has a way of slicing the processed image array for the x-axis of the range graph and the y-axis of the azimuth graph. The program then renders the chartjs.html template and sends the SAR image URL as a string and the range and azimuth arrays as lists. The operation of the compare image web page is shown below in Figure 3.5.

Chartjs.html, can be found under the GitHub repository. The html generates three canvas elements in such a way that the range graph will be the same height as the image and the azimuth graph will be the same length. The JavaScript code then draws the SAR image upside down onto the first canvas. The script then receives the range and azimuth x and y axes via the tojson method. These are then converted into the format "x: xvalue, y: yvalue" for the scatter graph with the same datarize() function described in Section 3.5.4. These are then used to create the scatter graphs for range and azimuth with one significant difference in the range graph. The range graph's y-axis needs to be flipped, in other words, starting from 0 at the top and increasing downwards, so that the values

match the image. In order to do this, the "ticks" property of yAxes is set to "reverse: True".

Updating the Graphs on mouse click

A feature of this web page is that the range and azimuth scatter graphs update based on where the user is slicing the image. This is decided based on where the user clicks on the SAR image.
In order to do this, the JavaScript section of the html contains an event listener which listens for a click event on the image canvas. A JavaScript function, getMousePos(), was made that gets the bounds of the canvas and returns the mouse's x and y positions on the canvas by subtracting the mouse's x and y positions by the canvas' left and top bounds. The listener function then sends the x and y co-ordinates to an app route in the Flask controller called updateGraphs.
The updateGraphs app route executes the python function of the same name. The function receives the x and y positions and scales the x and y value to the image resolution by dividing by the canvas' dimensions and multiplying by the range and azimuth maximums. Then, using the find_nearest_index function, the program finds the closest value in the array to the x and y values selected. Then, the program finds the new range and azimuth lists, sliced from the global processed image array, and combines them into one list to send to the template. In order to send the list back to the template, the combined list needs to be converted into a python string using json.dumps() and then that string can be returned as the function's output.
Once the python function has completed and the template has received the new lists, the function converts the new lists into the "x: x, y: y" format and uses the custom functions, removeData() and addData() to remove the previous scatter plots and replace the charts with the new ones.

3.5.6 About Page

The about page is entered when the user clicks on the link in the navigation bar, directing them to the "about" app route which renders the template about.html which can be found in the GitHub repository. The template is a very basic HTML document, containing a heading, a picture of the miloSAR and a short paragraph describing the testbed and the company behind it. It also contains within this paragraph, a link to the company's website.

3.5.7 Data flow

Below are the data flow diagrams that give a general understanding of the processes involved within this system. Figure 3.6 shows the interaction between the user and the web interface powered by the flask controller. Figures 3.7 and 3.8 show a more detailed description of the process involved in Figure 3.6.

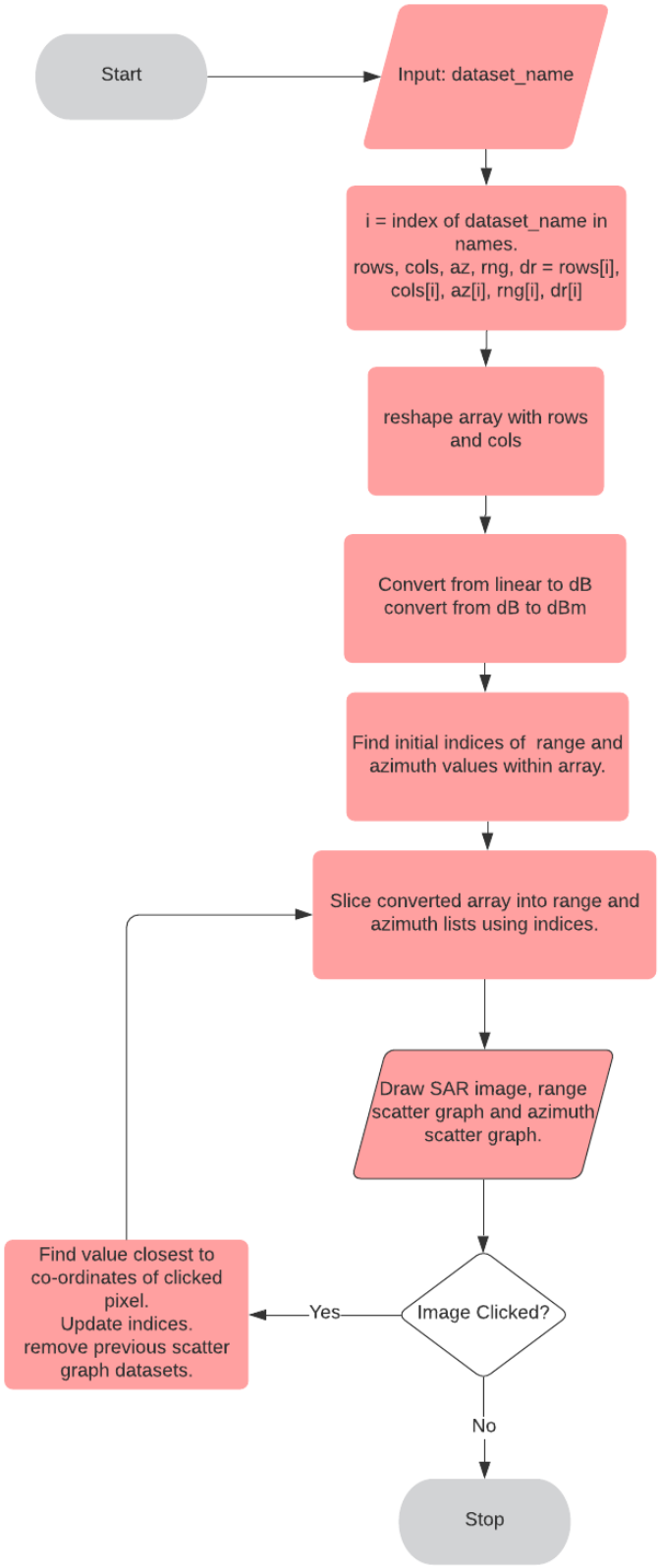


Figure 3.5: A flow diagram showing the logical operation of the "Compare Image" web page that allows the user to view the range and azimuth values of the image.

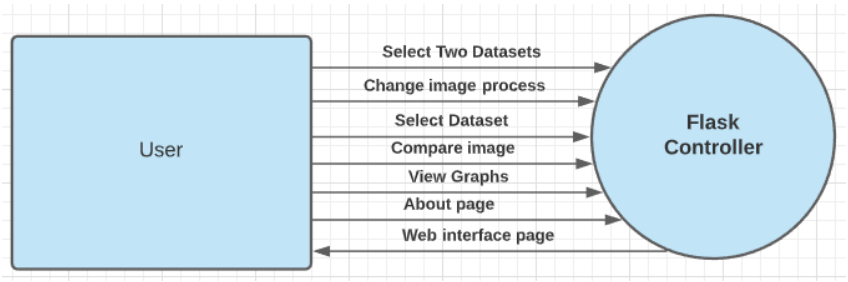


Figure 3.6: A level 0 data flow diagram of the overall web interface's processes.

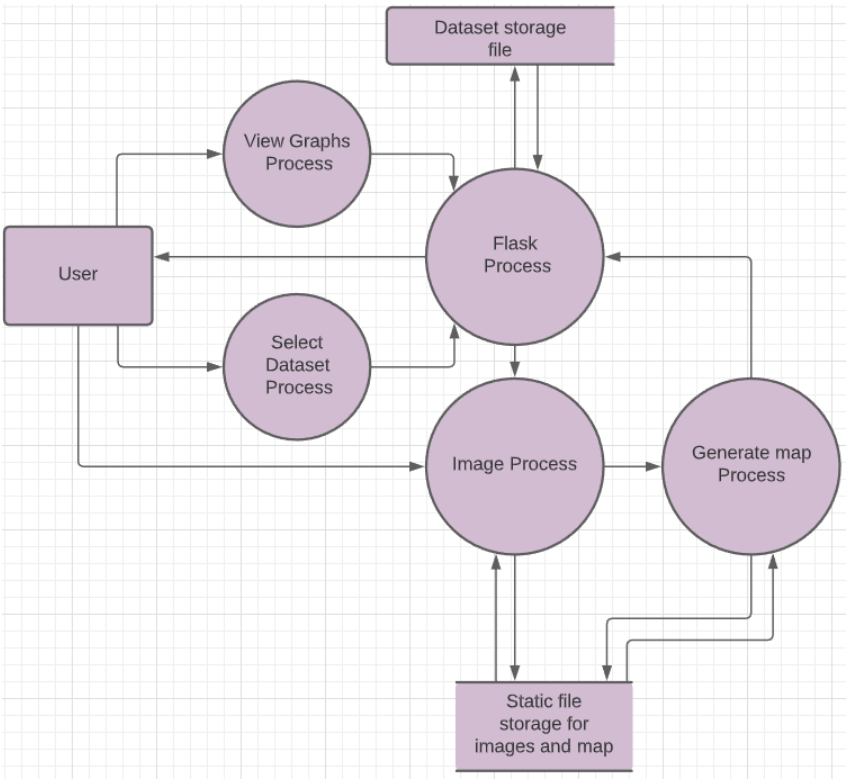


Figure 3.7: A level 1 data flow diagram demonstrating the processes within the flask controller concerning process from the main page.

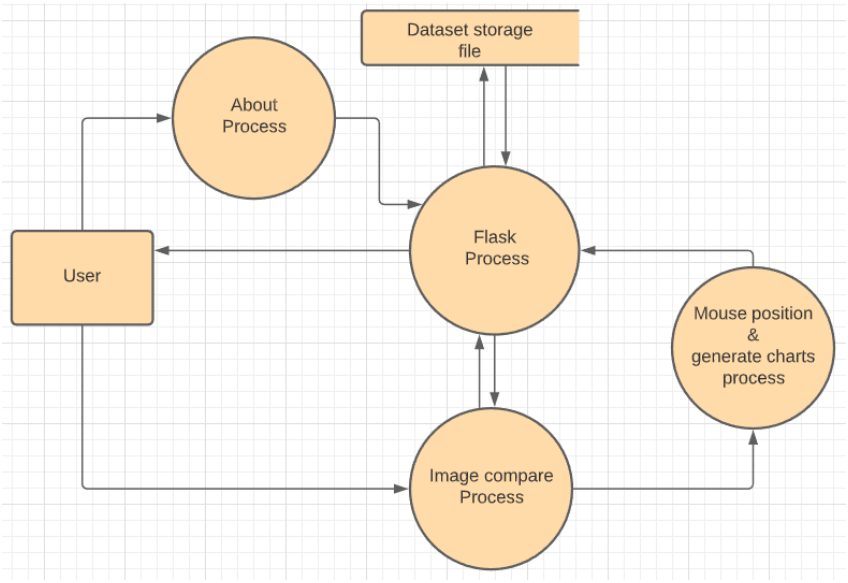


Figure 3.8: A level 1 data flow diagram demonstrating the processes within the flask controller concerning the other web pages of the interface.

Chapter 4

Results

The following are screenshots of the resulting interface that the program creates. These screenshots are what the user sees within their web browser and each screenshot is taken after a single action to demonstrate the function being used.

4.1 Producing the HDF5 file Results

Once the HDF5.py program, found in the repository previously linked, is run, the result is an HDF5 file generated under the "datafiles" folder. This hdf5 file can then be viewed using the HDFViewer program as shown below in Figure 4.1. The program has created an HDF5 file with a group called MyGroup. Within that group is the dataset containing the binary array of the SAR unprocessed image and as can be seen, the attributes of the dataset have been assigned.

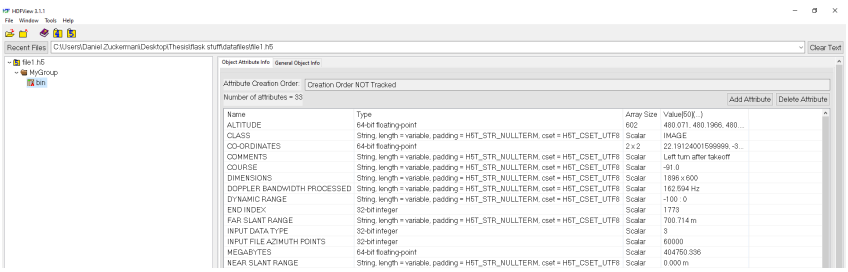


Figure 4.1: A view of the hdf5 file that is used for the program. This file has been generated by extracting information from the various files outputted during the dataset's campaign.

4.2 Image Processing Results

Once the function, generatePNG, imported from proccess_to_image.py is finished executing, the result is the image shown below in Figure 4.2. This image has been processed from the binary dataset of the HDF5 file, resized and then rotated. The image below is of a different dataset to better demonstrate the rotation as the campaign of the first dataset is only rotated by 1 degree and would be harder to demonstrate the effectiveness of the program.

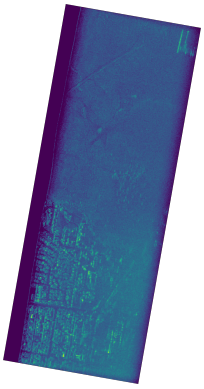


Figure 4.2: The resized and rotated image of the SAR data.

4.3 Resulting Generated Map

Below, Figure 4.3, is a screenshot of the folium map that is generated as a result of running the generate_map function imported from ee_demo.py. Also generated on the folium map are layers of image data from the Sentinel 1 from Google Earth Engine. These layers are shown in Figure 4.11.

4.4 Web Interface Results

When the flask controller program is run, the web interface, shown in the below figures, is produced. When the user visits the site, the program first opens on the home page, shown in Figure 4.4. As you can see, the navigation bar at the top of the screen has been generated according to the html template, layout.html, found in the templates folder of the GitHub repository. The rest of the page is the result of rendering the html template,

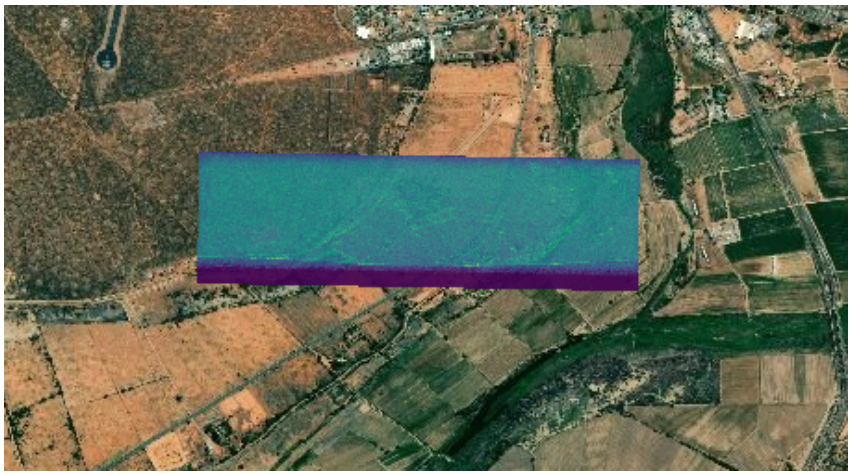


Figure 4.3: A screenshot of the folium map created from the result of running the generate_map function.

template.html which can be found in the same folder. As can be seen in Figure 4.4 and Figure 4.5, the web page correctly displays the folium map as well as all the parameters of the dataset's campaign.

Figure 4.4: A screenshot of the home page that the user sees when entering the website. This is the first page to be shown when the main program is run.

Also demonstrated, in Figure 4.6, is the query search feature. When the user enters the name of the dataset required, the datasets with matching timestamps are visible while the others are made invisible. In the case of this figure, the user has the first dataset, "18_11_21_10_04_13," selected but is searching for the second dataset, "18_11_21_10_13_11," and, as can be seen in the figure, only that dataset shows.

Figure 4.5: The main page showing that all of the dataset's parameters are displayed.

4.4.1 Processing Commands

The user has the ability to alter the process by which the image is displayed. The user does this by inputting a command with a parameter as shown in a tooltip that appears when the user hovers over the text field. Once the user has entered the desired function, the user must click on the "Execute Command" button. Figure 4.7 shows the result when the user inputs the setColourMap() command with the input parameter, "gray". Figure 4.8 shows the result when the user inputs the gaussian() command with the input standard deviation as 1. Figure 4.9 is the result of the user changing the dynamic range by which the image is processed. In the case shown, the user has inputted the changeDR() function with the lower value equal to -50 and the upper value equal to 0. Figure 4.10 demonstrates the result of the user entering the originalImage() function, successfully reprocessing the image with its "default" parameters.

Figure 4.11 shows the result of the user toggling the two satellite image layers on the folium map. The images were successfully imported from the Google Earth Engine's

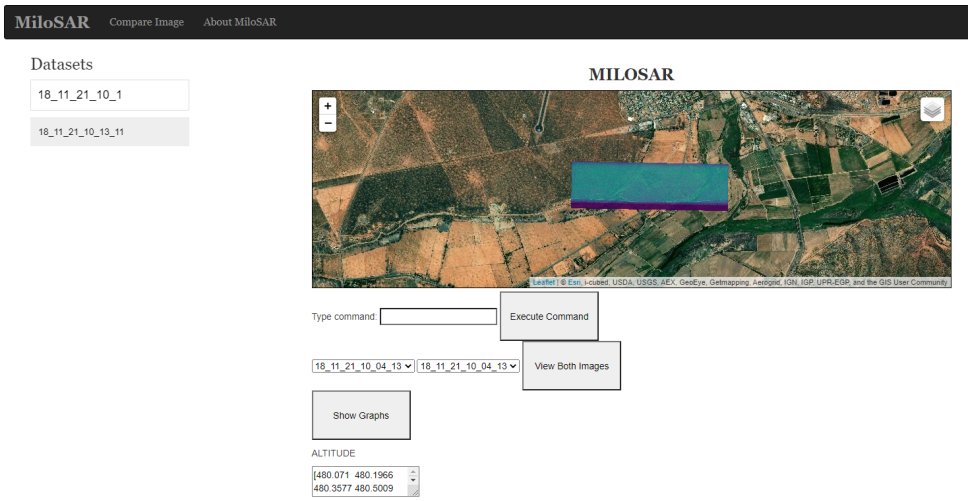


Figure 4.6: Demonstrating the search feature of the datasets menu. All datasets that match the name query are shown while the others become invisible.

database of the Sentinel 1 satellite imagery.

The following figure, Figure 4.12, is the result of the user selecting two different datasets from the combo box and pressing the "View Both Images" button. The result is the interactable map displaying both datasets' SAR images for the user to analyze.

Figure 4.13 is displayed when the user selects the "Graphs" button. The program redirects them to this page which is generated by the "graphs.html" template, shown under the templated directory of the GitHub repository.

Figure 4.14 shows the resulting page once the user has selected the second dataset in the menu. The folium map is updated to show the second dataset's SAR image as well as displaying the dataset's list of attributes and their values below.

When the user selects the "Compare Image" link in the navigation bar, the user is directed to the page as shown in Figure 4.15. The data and image shown are based on which dataset is selected when the user clicks the link. The result is the SAR image, along with the range vs normalized power and normalized power vs azimuth graphs. These graphs are representing the range and azimuth values through vertical and horizontal slices of the SAR image through the centre.
Once the user clicks on a location within the image, the range and azimuth graphs change to represent their respective values through the vertical and horizontal slices through the position of the mouse click. As can be seen in Figure 4.16, the user has clicked their mouse in the top left corner of the image. This has resulted in the range graph changing but more noticeably the azimuth graph is a constant -100 dB. This is because the horizontal

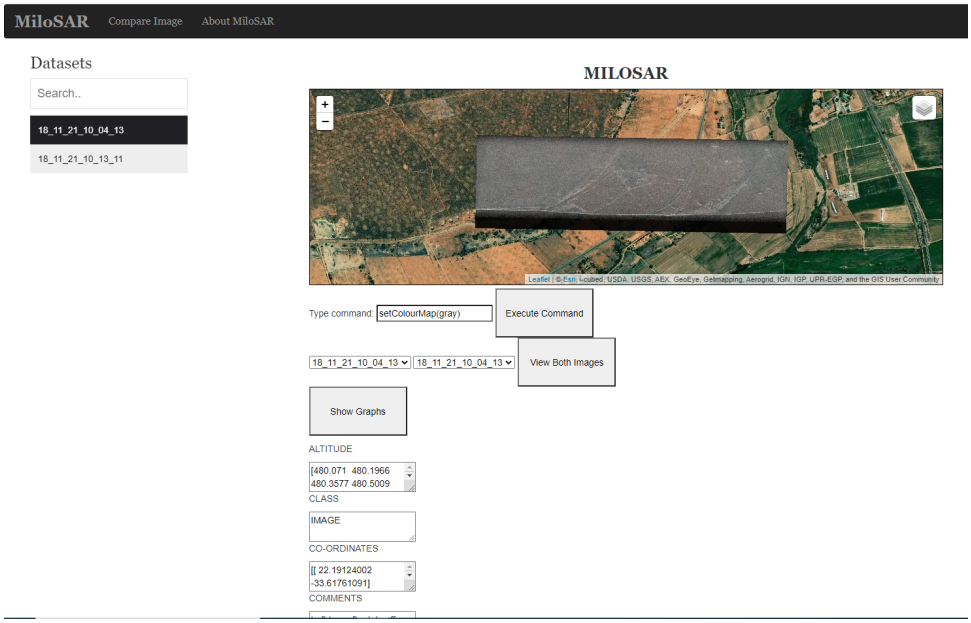


Figure 4.7: The result of the user changing the colour map of the SAR image to gray.

line through the mouse position is a constant black area.

When the user selects the "About MiloSAR" link in the navigation bar at the top of the screen, the user is redirected to a new page, generated by the "about.html" template shown in the GitHub repository. As can be seen in the figure, this page contains an image of the miloSAR and a brief description of what the miloSAR is capable of and what the company behind it is all about.

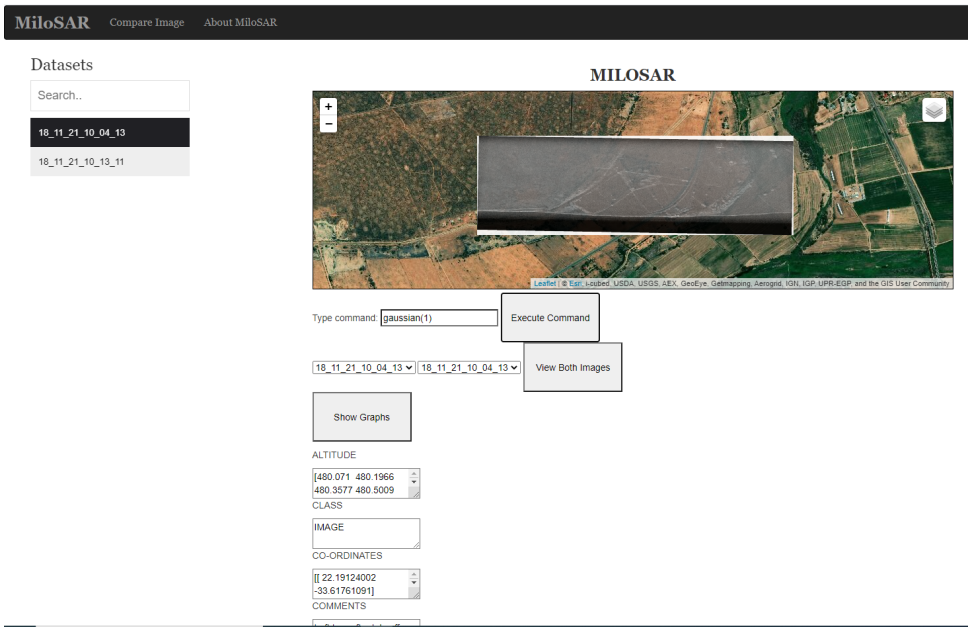


Figure 4.8: The result of the user performing a Gaussian sweep with a standard deviation of 1 on the SAR image.

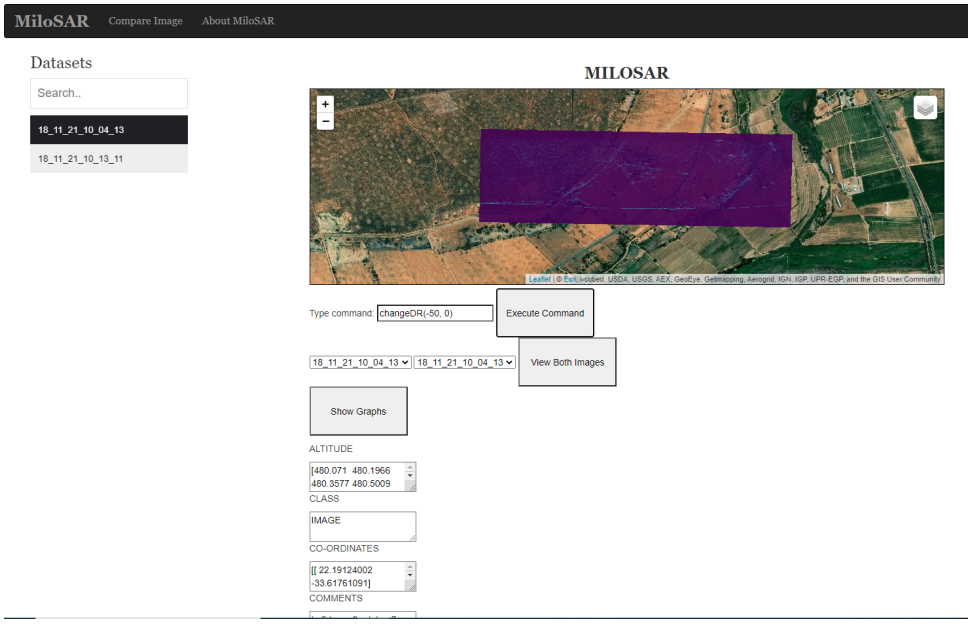


Figure 4.9: The result of the user changing the dynamic range with which the SAR image is processed.

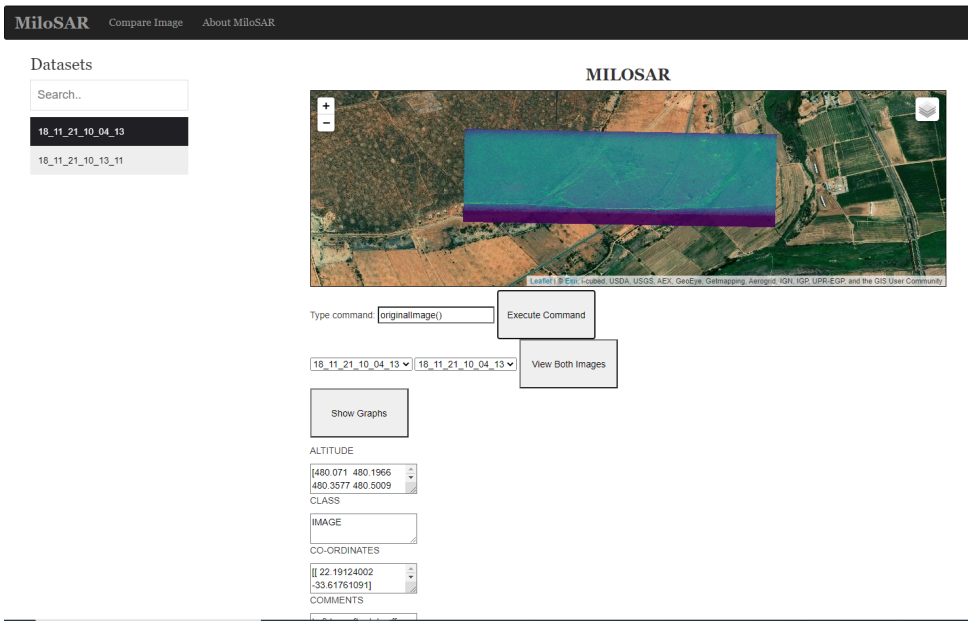


Figure 4.10: The result of the user inputting the command to revert the SAR image back to the original image with its predefined attributes.

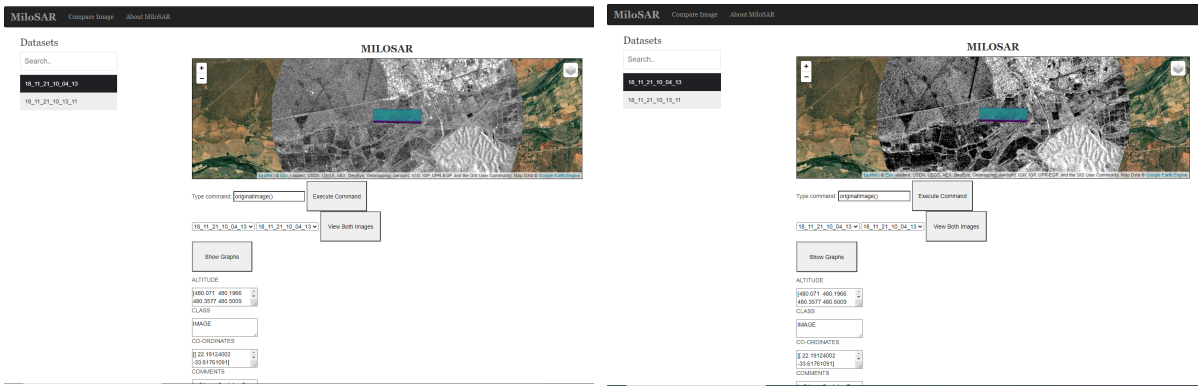


Figure 4.11: The result of the generate_map function adding two layers of the Google Earth Engine's Sentinel 1 satellite imagery to the folium map.

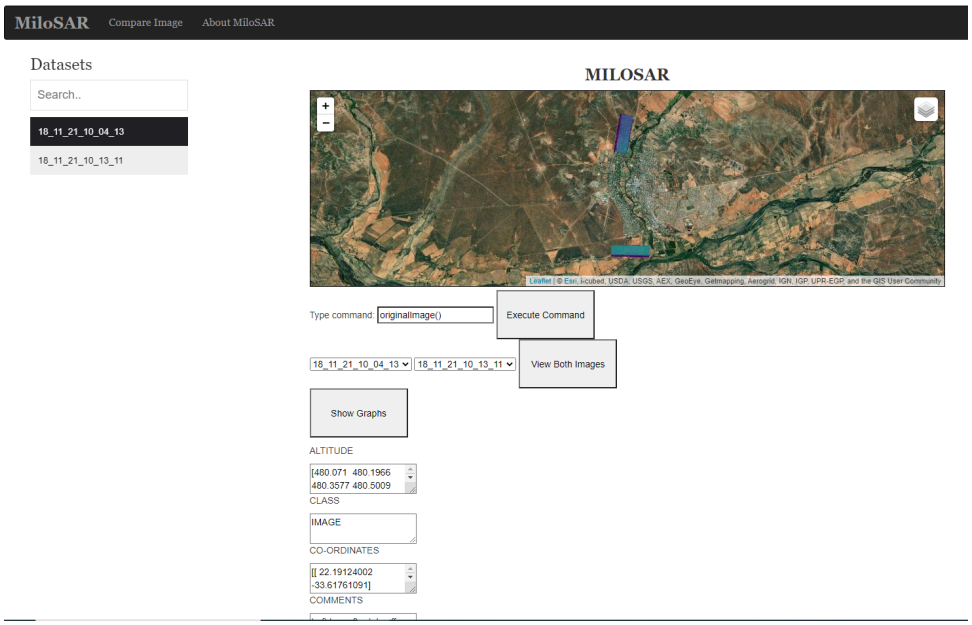


Figure 4.12: The result of the user selecting two datasets and pressing "View Both Images". The result is the folium map displaying both SAR images at once in separate layers.

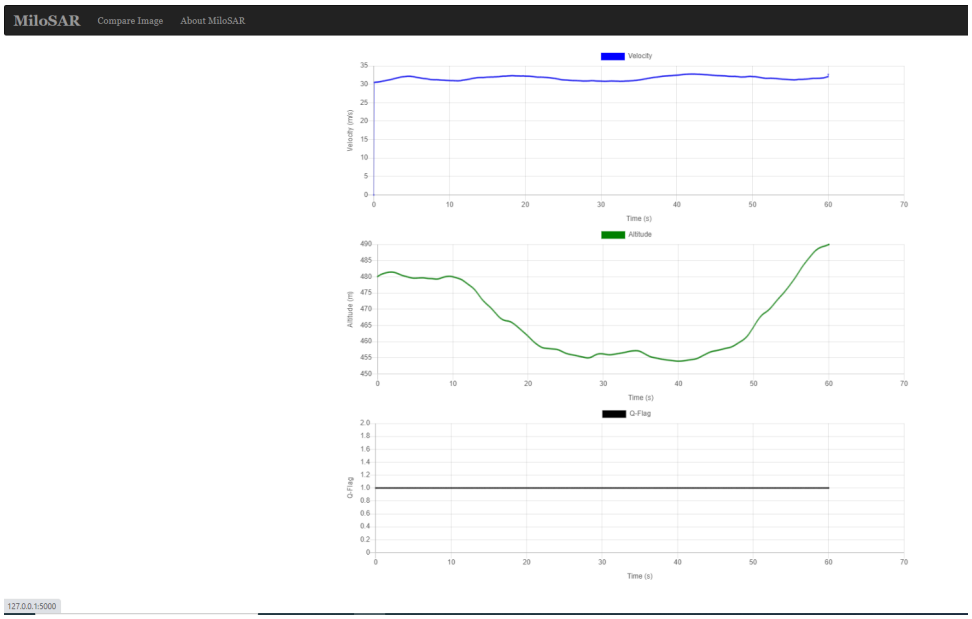


Figure 4.13: The result of the user clicking the "Graphs" button which redirects them to a page displaying graphical representations of the campaign's flight.

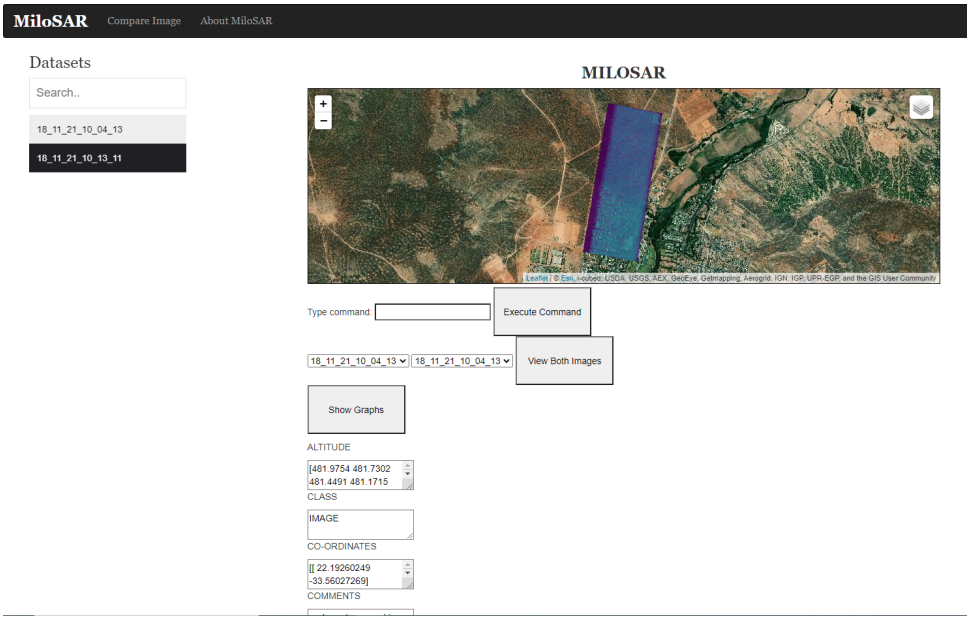


Figure 4.14: The result of the user selecting another dataset from the menu. The new image is displayed on the map and the parameters shown have updated to the specific dataset.

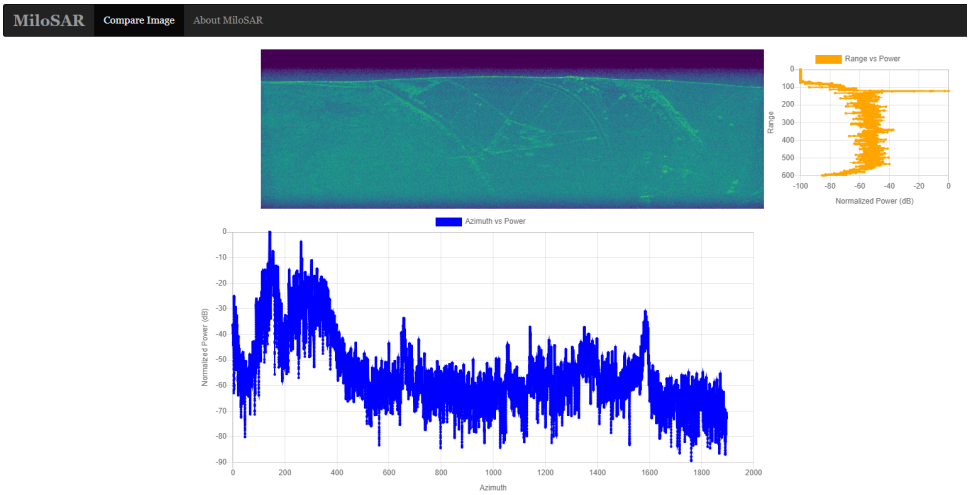


Figure 4.15: The result of the user navigating to the "Compare Image" page. The resulting image shown is based on which dataset is selected in the main page and the resulting range and azimuth graphs are based on the slices through the centre of the image.

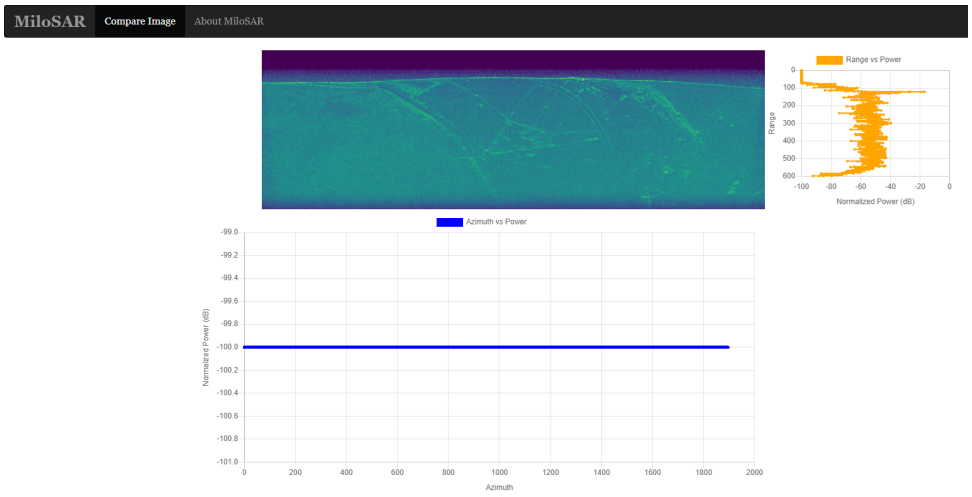


Figure 4.16: The result of the user clicking on a new position within the image. The result is the graphs changing to reflect slices of the image through the point where the mouse was clicked.

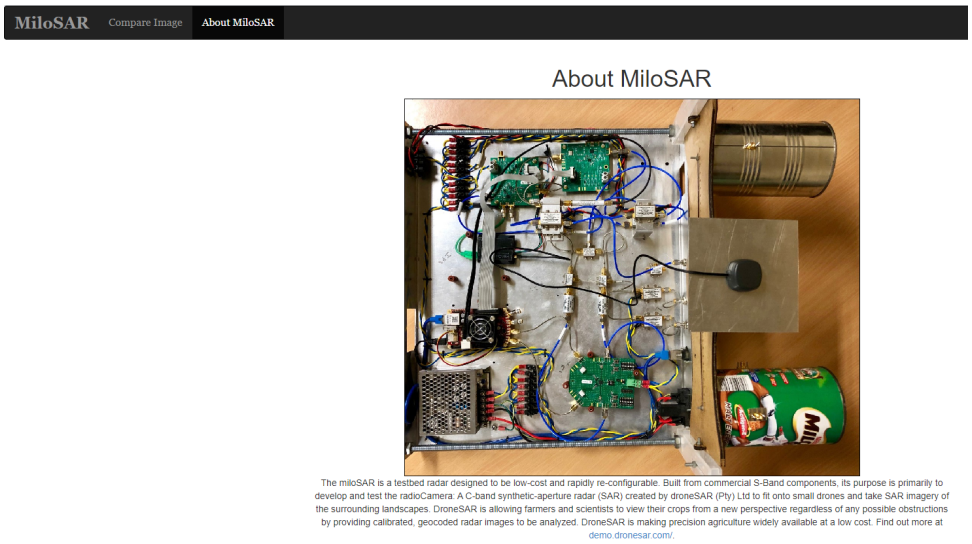


Figure 4.17: The result of the user navigating to the "About MiloSAR" page. The page outputs a picture of the miloSAR as well as a short description.

Chapter 5

Discussion

Within this chapter, the study will go into interpretation and analysis of the results and determine if the Acceptance Test Protocols derived in Chapter 3.1.4 and shown in Table 3.4 have been met and to what extent. It will also look at the extent of modularity of the program.

5.1 Acceptance Test Protocols

Using the results of the previous chapter, we can now assess whether the acceptance test protocols (ATPs), shown in Table 3.4, have been passed.

ATP UR1-0-001:FR1-1-001:DS1-1-001

This ATP is to test if the web interface contains general information about the miloSAR including images. Figure 4.17 verifies that the website does indeed contain an about page containing a short paragraph describing the miloSAR. The website does also contain an image of the miloSAR. Therefore, this ATP is considered as passed.

ATP UR1-0-001:FR1-1-002:DS1-1-002

This ATP aims to verify that the main program and the web interface was made using the python programming language. As can be seen within the GitHub repository, the main controller of the web interface is created in python using the Flask library. Therefore, this ATP is passed as well.

ATP UR1-0-001:FR1-1-002:DS1-1-003

The third ATP wishes to verify that the SAR image data has been successfully layered over an interactable folium map along with the appropriate Google Earth Engine data. Figures 4.4 and 4.11 show that the program is indeed correctly displaying the SAR image along with earth engine data on an interactable folium map with which the user can toggle layers and navigate the map in the same way one would navigate google maps.

ATP UR1-0-002:FR1-2-001:DS1-2-001

This ATP makes sure that the web interface contains a type of menu containing all the datasets available and that the user can select a dataset and view its information. Figure 4.14 demonstrates that there is a dataset menu on the main page and when a user selects the second dataset, the parameter and image data is different to that of the first dataset from Figure 4.4. This passes the ATP in question.

ATP UR1-0-002:FR1-2-002:DS1-2-002

This ATP wishes the verify that the SAR image shown on the map is the image relating to the dataset selected. Figures 4.4 and 4.14 show that the SAR image changes depending on which dataset is selected and is therefore, the correct information shown.

ATP UR1-0-002:FR1-2-003:DS1-2-003

This ATP aims to determine if the interface successfully contains data pertaining to the miloSAR's flight and redirects the user to view this information. Figure 4.5 contains

the parameters of the dataset including the velocity, altitude and time as well as Figure 4.13, which shows graphs of the datasets velocity, altitude and Q-point throughout the campaign. This shows that the program passes this ATP as it successfully contains and services a way of viewing the flight information.

ATP UR1-0-002:FR1-2-004:DS1-2-004

The interface is also required to display multiple SAR dataset images on one folium map at once. The results shown in Figure 4.12 show that, using the combo boxes and the "View Both Images" button, the user is able to regenerate the map with the SAR images from both datasets on the same map as different layers. This passes the ATP as it shows that the interface caters for the viewing of multiple images at the same time.

ATP UR1-0-002:FR1-2-005:DS1-2-005

The interface is also required to allow the user the ability to search for a specific dataset based on its name. The results shown in Figure 4.6 verify that the user is able to type the required name into the search bar above the dataset menu, and only the matching datasets containing the entered string will appear for the user to select.

ATP UR1-0-003:FR1-3-001:DS1-3-001

This ATP concerns the Compare Image feature of the interface. It aims to verify that when the user enters the page, the correct SAR image is shown and the range and azimuth graphs show the values through the centre of the image. It also requires that the program update the graphs to the values through a new point based on a mouse click on the image. Figure 4.15 shows the result of the interface when the user first enters the page with the first dataset selected. The figure shows that the SAR image of the first dataset is shown correctly and that the graphs reflect the values through the centre of the image. Figure 4.16 demonstrates the effect of the user clicking the mouse on the top left area of the image. The range graph changes and the azimuth graph becomes a constant power of -100 dB. This verifies the ATP by showing the graphs update instantly when the user clicks on the image with their mouse.

ATP UR1-0-003:FR1-3-002:DS1-3-002

This ATP wishes to verify that the altitude, velocity, Q-point, azimuth and range data is shown in the form of graphs. This is visibly shown by the results in Figures 4.13 and 4.15.

ATP UR1-0-003:FR1-3-003:DS1-3-003

This ATP merely wishes to ensure that the libraries necessary for the tasks of the project are being used within the program and as can be seen from code shown in the GitHub repository, all libraries have been imported and used correctly.

The web interface has passed all the ATPs that were originally derived.

5.2 Modularity

An important thing to consider throughout the design of this program was: how modular can the program be? Keeping in mind the subsystems laid out in Figure 3.1, the program's modularity can be assessed on how much work needs to be made to adapt this program when an element of a subsystem is changed.

If someone wished to adapt the interface to display satellite imagery, then the process by which the image is developed would be different. Essentially, the only thing that would need to be altered in the program would be the generatePNG and processPNG functions from process_to_image.py as these contain processes specific to SAR imagery data. Furthermore, the flight data present in the miloSAR campaigns would not be present in a satellite campaign. Instead other data pertaining to the satellite would be present in the HDF5 file. This isn't a problem as the chartjs graphs that are produced as part of the graphs section of the interface can accept different datasets with ease. The attributes shown on the main page of the interface are dependent on the attributes provided in the HDF5 file. Allowing for flexability when it comes to any individual dataset's parameters.

Another example would be if someone wanted to use this interface as a library for viewing and processing facial recognition data. Again, the function called to process the image

would need to be different as well as new functions made, as the current functions of the web page are useful for radar image data but not so useful for imagery such as this.

If the data format with which the datasets were stored was not an HDF5 file and instead was some other file format, for example Json, the program would only need a small change to its main controller code. Lines 31-71 of controller.py contain the code necessary for extracting the datasets' information from the HDF5 files. If the file format were different, the program would need to extract the necessary parameters from each file in the specified folder. However, the designer of the program would have to know every parameter in the file specifically in order to extract the attributes. Furthermore, if this were the case, the parameters and the image data of each dataset would be stored seperately as opposed to the HDF5 file which allows for both to be stored under one dataset simultaneously in the file.

Chapter 6

Conclusions

As can be seen from the results, specifically the results of Figure 4.3, the web interface has effectively succeeded in fulfilling the tasks set out at the beginning of this study. The program successfully combines positional and radar image data and outputs this in the form of an interactable map. Furthermore, the interface successfully serves as a library where users can view miloSAR measurement campaigns, select whichever dataset they choose, and reprocess the data in a customised way. As seen in the Discussion chapter, the web interface has also satisfied every acceptance test protocol.
One important conclusion from the testing done is that the desktop computer used to run the program struggled with a lot of the processing done. Specifically, whenever the image needed reprocessing or every image was generated for the first time on start up of the program. Therefore, the program's operation would greatly benefit from a more powerful processor and, more importantly, more memory. However, if we look at the results of the program, regardless of execution time, the web interface can be considered a success. Although the primary objective of this study was to create the web interface, this investigation also demonstrates the efficiency and usefulness of using HDF5 as a file format for containing datasets and their parameters. Creating the HDF5 file and reading the data within the file was done with relative ease with the use of the python package supported by the HDF group, h5py.

While the investigation was specific to radar imaging data, the web interface designed is modular and can be adapted to many different forms of data by changing some functions that the main program uses. The program can also accept different file formats as long as the initial code when the program is first run is altered to accommodate the specific file format.

The overall purpose of the study was to design and develop a web interface that would act as a library, data processor and parser for the miloSAR testbed radar. Having analyzed the results, it can be concluded that the program designed succeeds in delivering such a web interface.

Chapter 7

Recommendations

After reviewing the results and concluding the investigation, there exist some recommendations on how this study could be taken further.

7.1 Cloud Based Processing

Due to the long run times of the processing of the program, the interface could benefit greatly from cloud based processing services such as Amazon Web Service (AWS) which provides cloud computing for a pay-as-you-go fee. These online services provide processing power which will greatly increase the response time of the website for a user. This is mainly due to the fact that whenever a new dataset is selected, the folium map is regenerated and replaced. Also to note is that the user has the ability to alter the parameters by which the imagery is processed. This involves working with complex arrays and takes a considerable amount of time to execute on a local computer.

7.2 Data File Format Wrapper

As mentioned previously, the program could accept the dataset under any file format, so long as the initialization code is changed to accomodate it. Therefore, the interface's modularity would increase with the use of a file wrapper. A file wrapper would allow the dataset to come in a number of different file format types. The program would need a custom file wrapper to be made that will interpret the data based on the file type in

order to create the arrays of image data and parameters that are needed for the interface.

7.3 Multiple Image Types

The web interface does not only have to be used for radar datasets. Any positional image data that can benefit from merging with a map can be used with this program. For example, the program could easily be used as a library, processor and parser for the Sentinel 1 satellite image datasets.

7.4 Reprocessing SAR Imagery

When dealing with SAR images, it is incredibly useful to take the SAR image and render it in a manner that can convey more information to a human being. There exist various mappings that allow a 16-bit SAR pixel magnitude image to be converted into an 8-bit pixel size that will alter the image and the histogram of the SAR image. Such mappings include histogram-stretched, quarter-power, logarithmic and arc-tangent image. These mappings could be implemented into the web interface as a function that the user could input. The option for the user to input a command already exists and thus, a mere addition to the input's 'if' branch is all that is needed.

Bibliography

[1] D. Jordan, P. Cheng, M. Inggs, A. Langman and Y. A. Gaffar, "Development of the miloSAR Testbed for the One Kilogramme radioCamera SAR for Small Drones," 2019 IEEE Radar Conference (RadarConf), Boston, MA, USA, 2019, pp. 1-6, doi: 10.1109/RADAR.2019.8835721.

[2] R. Sanders, "Developing Flask Extensions", PyCon 2014, 2014.

[3] S. Cohen, "What challenges has Pinterest encountered with Flask?", Quora.com, 2015. [Online]. Available: https://www.quora.com/What-challenges-has-Pinterest-encountered-with-Flask/answer/Steve-Cohen?srid=hXZd&share=1.

[4] Flask User's Guide, Pallets, 2010, [Online]. Available: https://flask.palletsprojects.com/en/1.1.x/.

[5] "Design philosophies — Django documentation", Docs.djangoproject.com. [Online]. Available: https://docs.djangoproject.com/en/2.0/misc/design-philosophies/.

[6] S. Hansen, "Advantages and Disadvantages of Django", Hackernoon.com, 2017. [Online]. Available: https://hackernoon.com/advantages-and-disadvantages-of-django-499b1e20a2c5.

[7] "Django introduction", MDN Web Docs, 2019. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction.

[8] Y. Nader, "What is Django? Advantages and Disadvantages of using Django", Hackr.io, 2020. [Online]. Available: https://hackr.io/blog/what-is-django-advantages-and-disadvantages-of-using-django.

[9] M. Di Pierro, Web2Py Complete Reference Manual, 5th ed. Chicago IL: Massimo Di Pierro, 2013. [Online]. Available: http://web2py.com/book

[10] "web2py vs Django detailed comparison as of 2020", Slant, 2020. [Online]. Available: https://www.slant.co/versus/1397/1746/~web2py_vs_django.

[11] Python Web Development Libraries. Tutorials Point (I) Pvt. Ltd., 2018, p. Chapter 8.

[12] Jinja Documentation, Release 2.11.2, Pallets, Jun 2020. [Online]. Available: https://jinja.palletsprojects.com/_/downloads/en/2.11.x/pdf/

[13] High Level Introduction to HDF5, The HDF Group, 2016. [Online]. Available: https://support.hdfgroup.org/HDF5/Tutor/HDF5Intro.pdf.

[14] D. Crockford, "Introducing JSON", Json.org. [Online]. Available: https://www.json.org/json-en.html.

[15] D. Priest, "Pros and Cons of JSON vs HTML", Avato, 2019. [Online]. Available: https://avato.co/developers/pros-cons-json-vs-xml/.

[16] O. Ben-Kiki, C. Evans and I. Net, YAML Ain't Markup Language (YAML™) Version 1.2, 3rd ed. Oren Ben-Kiki, Clark Evans, Ingy döt Net, 2009.

[17] JavaScript and Python Guides. Google Earth Engine, 2020. [Online]. Available: https://developers.google.com/earth-engine/guides.

[18] N. Gorelick, M. Hancher, M. Dixon, S. Ilyushchenko, D. Thau and R. Moore, "Google Earth Engine: Planetary-scale geospatial analysis for everyone", Remote Sensing of Environment, vol. 202, pp. 18-27, 2017. Available: https://doi.org/10.1016/j.rse.2017.06.031.

[19] S. Ramakrishnan, V. Demarcus, J. Le Ny, N. Patwari and J. Gussy, "Synthetic Aperture Radar Imaging Using Spectral Estimation Techniques", University of Michigan, 2002.

[20] Doerry, Armin. (2019). SAR Image Scaling, Dynamic Range, Radiometric Calibration, and Display.

[21] HDFView User's Guide, 3rd ed. The HDF Group, 2018. [Online]. Available: https://portal.hdfgroup.org/display/HDFVIEW/HDFView+3.x+User%27s+Guide.

[22] HDF5 User's Guide, Release 1.10, The HDF Group, July 2019. [Online]. Available: https://portal.hdfgroup.org/display/HDF5/HDF5+User+Guides?preview=/53610087/53610088/Users_Guide.pdf.

[23] HDF5 for Python, Andrew Collette and contributors, 2014. [Online]. Available: https://docs.h5py.org/en/stable/.

[24] Folium 0.11.0 Documentation, 2013. [Online]. Available: https://python-visualization.github.io/folium/.

Appendix A

Addenda

A.1 Ethics Forms

A.1.1 Ethics Form Submission

Project Title

Web Interface for miloSAR, dataset parser, processor and library

08/18/2020

id. 17255304

by **Daniel Zuckerman** in **EBE Electrical Submissions Undergraduate**

ZCKDAN001@myuct.ac.za

Original submission

08/18/2020

Project Aims	**The aim of this project is to produce a web interface for the miloSAR radar that displays and produces information based on datasets from the radar.**
Ethical Issues	**There are no ethical issues regarding this thesis other than plagiarism of other people's code or designs.**
Application Checklist	**Read the EBE Ethics in Research Handbook before completing this application** **Questionnaire to be used in the research (where applicable)** **Consent form where (where applicable see Addendum 2)** **If needed a letter motivating an expedited review. This letter should be included in the cover letter.**
Researcher(s)	**Daniel Zuckerman**
Department	**Electrical Engineering**
E-mail	**ZCKDAN001@myuct.ac.za**
Status of Applicant	**Student**
Degree Being Studied (For Students Only)	**BSc mechatronic engineering**
Name of Supervisor (For Students Only)	**Amit Mishra, and Darryn Jordan**
Review Track	**Normal**
Motivation for an Expedited Review	n/a

Completed Ethics Application Form

EBE_EiRC_signature_form.docx

SECTION 1: Overview of ethics issues in your research project	n/a
Question 1: Harm to Third Parties	**No**
Question 2: Human Subjects as Sources of Data	**No**
Question 3: Participation or Provision of Services To Communities	**No**
Question 4: Conflicts of Interest	**No**
	If you have answered YES to any of the above questions, please ensure that you append a copy of your Research Proposal (Addendum 1), as well as any interview schedules or questionnaires and consent documentation (Addendum 2) and complete further addenda as appropriate. **I hereby undertake to carry out my research in such a way that:** **1. there is no apparent legal objection to the nature or the method of research; and** **2. the research will not compromise staff or students or the other responsibilities of the University;** **3. the stated objective will be achieved, and the findings will have a high degree of validity;** **4. limitations and alternative interpretations will be considered;** **5. the findings could be subject to peer review and publicly available; and** **6. I will comply with the conventions of copyright and avoid any practice that would constitute plagiarism**
ADDENDUM 1 Supporting documents	n/a
Research Proposal	**Research_Proposal_for_Honours_Thesis.docx**

ADDENDUM 2 To be completed if you answered YES to question 2 in section 1	**It is required that you read the UCT Code for Research involving Human Subjects in order to be able to answer the questions in this addendum.** **Ethical research should safeguard the interests of society and the welfare of all who participate in the research, be they individuals or groups. In this section the researcher is asked to consider the implications of their research on participants in the research. The researcher should outline risks that participants will face by being involved in the research.** **When a research involves vulnerable people, a researcher is expected to obtain informed consent from participants. This informed consent should be signed by the participants. Informed consent is intended to protect the interest of both participants and the researcher should something go wrong or should conflict arise between the researcher and the participant.**
Question 2.1: Discrimination	n/a
Question 2.2: Participation of socially or physically vulnerable people	n/a
Question 2.3: Informed consent	n/a
Question 2.4: Confidentiality	n/a
Question 2.5: Anonymity	n/a
Question 2.6: Risks of physical, psychological or social harm	n/a
Question 2.7: Payments and giving of gifts	n/a
Interview Schedule	n/a
Consent Form	n/a
Additional Comments	n/a

ADDENDUM 3 To be completed if you answered YES to question 3 in section 1	**Research may sometimes interfere with the organization, progress or advancement of communities. In this section the researcher is asked to consider the effect of their research on a community or communities involved in the research. Attention should be paid to whether the research will disrupt or interrupt the normal activities of the community and how the research will influence communities in the long term.**
Question 3.1: Community participation	n/a
Question 3.2: Termination of economic or social support	n/a
Question 3.3: Provision of sub standard services	n/a
Additional Comments	n/a
ADDENDUM 4 To be completed if you answered YES to question 4 in section 1	**A conflict of interest may compromise the conduct or outcome of a research project. It may also infringe on the interests of other researchers. In this section the researcher is asked to consider if their research may be compromised by the inclusion of certain individuals or groups in the research. The researcher is also asked to consider whether the inclusion of certain individuals or groups in the research will compromise the research of others at the university. For example, if any participants in the proposed research project are also involved in other projects at the university, have you considered if this participation will negatively affect their work?**
Question 4.1: Conflicts of interest	n/a
Question 4.2: Sharing of information	n/a
Question 4.3: Conflict of interest with other research	n/a
Additional Comments	n/a

A.1.2 EBE Ethics Signature Form

Application for Approval of Ethics in Research (EiR) Projects
Faculty of Engineering and the Built Environment, University of Cape Town

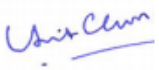
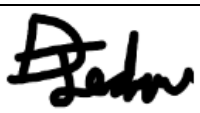
ETHICS APPLICATION FORM

Please Note:
Any person planning to undertake research in the Faculty of Engineering and the Built Environment (EBE) at the University of Cape Town is required to complete this form **before** collecting or analysing data. The objective of submitting this application *prior* to embarking on research is to ensure that the highest ethical standards in research, conducted under the auspices of the EBE Faculty, are met. Please ensure that you have read, and understood the **EBE Ethics in Research Handbook** (available from the UCT EBE, Research Ethics website) prior to completing this application form: http://www.ebe.uct.ac.za/ebe/research/ethics1

APPLICANT'S DETAILS		
Name of principal researcher, student or external applicant		Daniel Zuckerman
Department		Electrical Engineering
Preferred email address of applicant:		ZCKDAN001@myuct.ac.za
If Student	Your Degree: e.g., MSc, PhD, etc.	BSc Mechatronic Engineering
	Credit Value of Research: e.g., 60/120/180/360 etc.	40
	Name of Supervisor (if supervised):	Amit Mishra, Darryn Jordan
If this is a research contract, indicate the source of funding/sponsorship		
Project Title		Web Interface for miloSAR, dataset parser, processor and library.

I hereby undertake to carry out my research in such a way that:

- there is no apparent legal objection to the nature or the method of research; and
- the research will not compromise staff or students or the other responsibilities of the University;
- the stated objective will be achieved, and the findings will have a high degree of validity;
- limitations and alternative interpretations will be considered;
- the findings could be subject to peer review and publicly available; and
- I will comply with the conventions of copyright and avoid any practice that would constitute plagiarism.

APPLICATION BY	Full name	Signature	Date
Principal Researcher/ Student/External applicant	Daniel Zuckerman		14/08/2020
SUPPORTED BY	Full name	Signature	Date
Supervisor (where applicable)	Amit Mishra		16-aug-2020
	Darryn Jordan		18/08/2020

APPROVED BY	Full name	Signature	Date
HOD (or delegated nominee) Final authority for all applicants who have answered NO to all questions in Section 1; and for all Undergraduate research (Including Honours).			
Chair: Faculty EIR Committee For applicants other than undergraduate students who have answered YES to any of the questions in Section 1.			