Team Glacies Indicium

Technology Feasibility Document

10/23/2020

Project Sponsors	**Team Members**	**Faculty Mentor**
Dr. Heather Lynch	Beck Bohnker	Andrew Abraham
Dr. Mark Salvatore	Joe Carter	
Brian Szutu	Kass Coxen	
	Logan Garrett	
	Zach Spielberger	

Table of Contents

1 Introduction

Antarctica is one of the last great unexplored wildernesses. As the coldest, driest, and windiest continent on Earth, in addition to being isolated by hundreds of kilometers of frigid Southern Ocean on all sides, the vast and remote Antarctic is very inhospitable to humans. Because of this, Antarctica has been difficult to research, leaving much about the continent unknown. Geological studies of the continent have the potential to significantly increase our understanding of the climate and geological composition of the region, but researchers have been constrained by its inaccessibility and extreme weather conditions. With the development of satellites and advancements in optics and sensor technologies, however, satellite imagery has become a vital asset to advancing research of the continent.

Our clients, Dr. Heather Lynch and Dr. Mark Salvatore, are planetary and ecological scientists who are trying to learn more about Antarctic geology. They currently have a large dataset of high-resolution satellite photographs of Antarctica, which they want to use to document and study Antarctica's physical properties and surface composition. While the data is currently accessible to them, it is difficult and inefficient to access, analyze, and visualize, largely due to the large number and size of data and data format. Our product aims to make this information readily available in a format that is easy and intuitive for members of the scientific community, who may lack computer experience or knowledge, to interact with. Increasing accessibility of the data will aid not only our clients, but other researchers of the Antarctic as well, in progressing towards the collective goal of understanding Antarctica.

Currently, our clients search for new insights into Antarctic geology by using specialized software to examine multi-layered satellite imagery layer-by-layer, one image at a time. Right now, they can only interact with individual images which cover small areas, making it difficult to study geological phenomena on a large-scale. That's where our team comes in: we have been hired to build an internet application that allows scientists everywhere explore all the data at once by overlaying the image data onto an interactive web-based map. The tool will allow users to individually toggle any of the layers of data in each satellite image, providing access to a wide range of information including wavelength reflectance, rock type, and presence of snow, ice, and shadow. The tool will provide a GUI with which to view and filter the data over definable areas, further enhancing its contribution to the discovery of new science. The belief is that such an application will facilitate the global scientific study of Antarctica by making this vast trove of data interactive and easily accessible to scientists working in institutions across the globe.

Before we begin implementation work on this project, there are several technological challenges that we need to address. The purpose of this document is to enumerate these challenges and set forth our team's perspectives regarding potential solutions. The following section contains a broad overview of the primary technical components of this project, after which proceeds a detailed discussion of each individual challenge, and a solution proposal researched and written by the team members assigned to that component.

2 Technological Challenges

At this early stage in the project, we are in the process of identifying possible alternatives relative to these respective challenges and selecting which of those alternatives are the most promising solutions. In this Technological Feasibility Analysis document, we begin by analyzing major technological challenges we expect to encounter while creating our interactive map. The major problems we anticipate include:

- Condensing original GeoTIFF data into "summary" data, exchanging resolution for internet delivery speed
- Organizing and storing the data such that it can be served to a web application
- Drawing the structured data onto an interactive map utilizing interactive map tools from open source third-party providers
- Designing a GUI for viewing the data in a single map, where data is filterable and can be queried based on geological attributes
- Exporting selected areas and filtered versions of the map into various formats, including raster images.
- Encapsulating technology dependencies within the tool to ensure the product remains functional and maintenance-free over time (integration)

3 Technology Analysis

3.1 Technological Challenge: Downsampling

Satellite technology gives us the capability to take highly accurate images of the surface of the Earth. Most modern commercial satellites from the United States have a maximum spatial resolution of one square meter.

The satellite images that we will be dealing with, also called raster images, will only have resolution

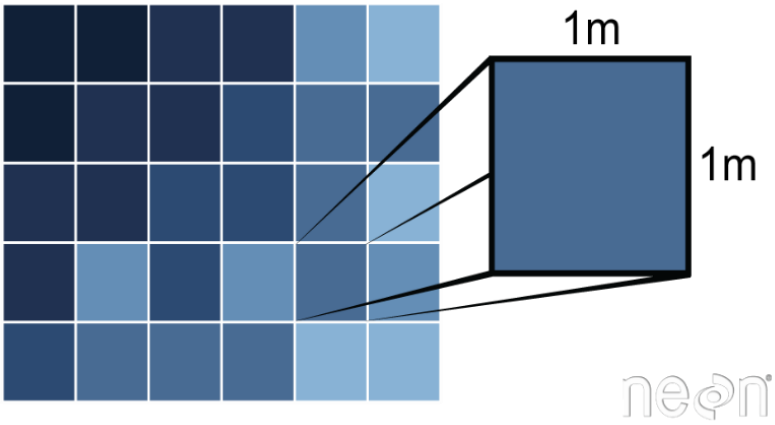

FIGURE 1: SPATIAL RESOLUTION IS A REPRESENTATION OF HOW MUCH AREA OF LAND EACH PIXEL OF AN IMAGE REPRESENTS

where one pixel is equivalent to one square meter, or a 1 m spatial resolution, compared to the 0.3 m resolution of some higher resolution commercial satellites.

Additionally, typical images contain three layers each consisting of red, blue, or green values, which are overlayed onto each other to form a color image (see below). On the visible light spectrum, the wavelength of light determines the color visible. When color images are captured, the camera records the reflectance in each range of wavelengths, which form layers or bands of an image.

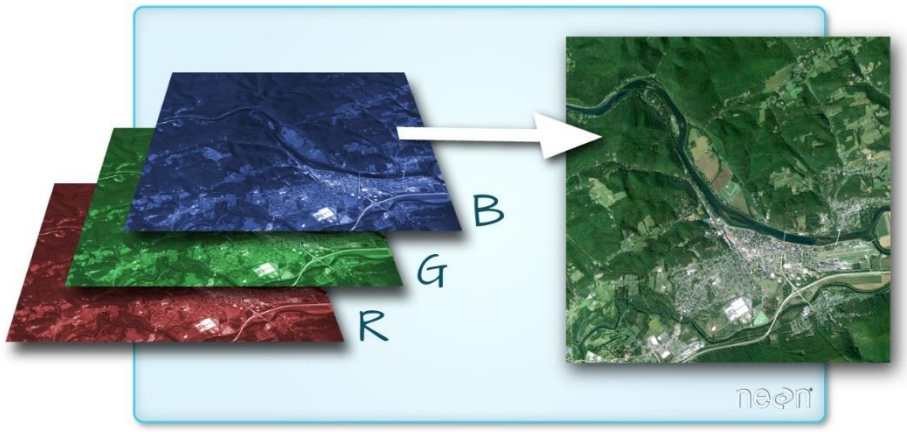

FIGURE 2: RED GREEN AND BLUE LAYERS ARE RECORDED, THEN OVERLAYED TO FORM A TYPICAL COLOR IMAGE. IN SATELLITE DATA, EVEN MORE BANDS OF LIGHT ARE RECORDED.

With our data, each pixel value is not only recorded for the three red, green, and blue bands, but for up to 18 bands, which are used for highlighting various geological attributes. Eight of these bands represent different spectral wavelengths, nine represent presence of geological phenomena, and one band is an error band representing standard deviation and likeliness of correct approximation. When these images cover hectares of land (10,000 m^2), they can start to contain incredibly large amounts of data, with each image ranging from around half a gigabyte to two or three gigabytes of data. When it takes around 60,000 images to completely survey Antarctica, we run into a serious technical hurdle in the sense of storage and scale. To reduce the magnitude of this challenge, the client is aiming for us to survey only a specific area of Antarctica, known as the Dry Valleys. This area can be surveyed with anywhere from 60-100 images. However, it is still not feasible for an interactive map on a web application to be served 100s of gigabytes in real time.

3.1.1 Desired Characteristics

Ideally, we need a way to resample these images to reduce the resolution, and correspondingly reduce the file size, so the data they hold will be easily servable across the web. We will need to serve data in the megabyte range rather than the gigabyte range to deliver a product that is responsive. An ideal solution is outlined below:

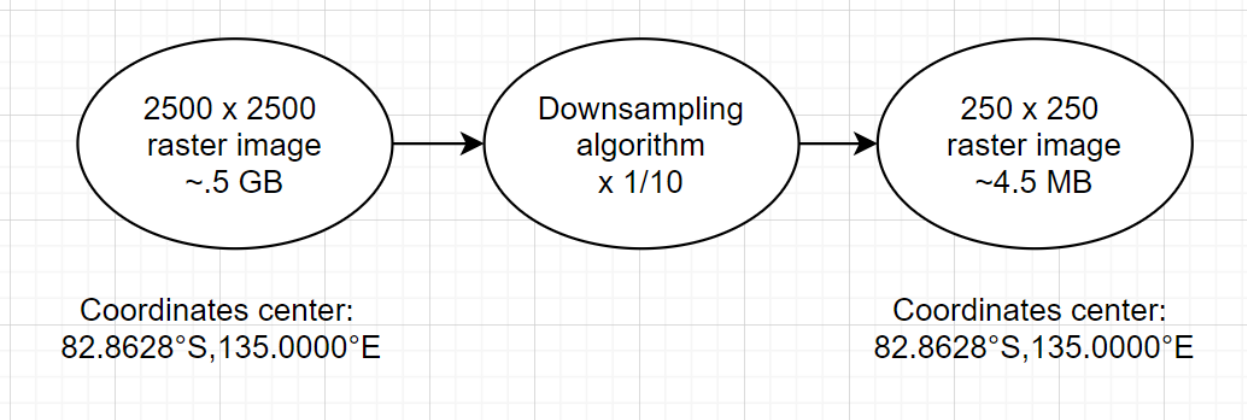

FIGURE 3: DOWNSAMPLING ALGORITHMS TAKE IN THE ORIGINAL IMAGE AND OUTPUTS AN IMAGE WITH A LOWER RESOLUTION AND FILE SIZE. WITH SATELLITE IMAGERY, THE SPATIAL EXTENT, OR AREA OF LAND THE IMAGE COVERS, IS NOT AFFECTED, SINCE DOWNSAMPLING AFFECTS THE NUMBER OF PIXELS USED TO REPRESENT THAT AREA.

In this example, the algorithm would downsample the data by a factor of 1/10, however the data size is reduced by 1/100. The relative latitude and longitude values should remain the same. Accuracy of the image would be up to 100 square meters rather than accuracy of 1 square meter.

3.1.2 Alternatives

Rasterio (python module)

Raster image data is not able to be parsed or viewed as raw values like in a text or comma-separated value file. This python module allows you to turn a raster image into a 3-dimensional array with the 18 corresponding bands of values for each x,y coordinate in the image. It's then easy to downsample this dataset. Then, we can either use those values in our database or convert them back into a raster image.

R (programming language)

R is a statistical programming language commonly used in statistical analysis. The benefit of this language is that it has a large repository of packages, which provide functions for data manipulation, including ones for raster data. The main packages we would need are the package `raster`, which has many functions for importing and manipulating raster images, and, `sf`, which allows us to work with spatial vector data. These have dependencies on the packages `GDAL`, `GEOS`, and `PROJ`, which allow for reading and writing geographical data, geometric operations, and projection conversions respectively.

3.1.3 Analysis

Factor	# bands	# x-pts	# y-pts
2	18	462	482
0.5	18	115	120
1	18	231	241

FIGURE 4: RASTERIO TRIAL RUN

Rasterio:

To the left, you can see a trial example of using rasterio. Generally, the first three numbers represent the number of bands and the number of rows and columns respectively. The first set of numbers is the data upsampled by a factor of 2, the second set downsampled by a factor ½, and then not sampled at all but just stored in the 3-dimensional array.

R:

In R, downsampling can be performed using either the `aggregate()` or `resample()` functions, which are both included in the base version of R (meaning that no library is required).

Reading in raster data, we see that a `RasterStack` object is created for multi-band raster images with the following output:

Class	RasterStack			
Dimensions	Number of rows	Number of columns	Number of cells	Number of layers
	1809	2568	4645512	8
Resolution	x		y	
	8.0216		9.046624	
Extent	X min	X max	Y min	Y max
	-60707.85	-40108.38	-569844	-553478.7

TABLE 1: OUTPUT FROM RAW DATA INPUT USING RASTER() TO CREATE A RASTER OBJECT

Calling `aggregate()` on the `RasterStack` object outputs a `RasterBrick` object with a lower resolution, but still covering the same spatial extent.

Class	RasterBrick			
Dimensions	Number of rows	Number of columns	Number of cells	Number of layers
	905	1284	1162020	8
Resolution	x		y	
	16.0432		18.09325	
Extent	X min	X max	Y min	Y max
	-60707.85	-40108.38	-569844	-553478.7

TABLE 2: OUTPUT AFTER CALLING AGGREGRATE() ON THE RASTERSTACK OBJECT.

With this method, we face challenges trying to visualize the data in a map interface. The output of plots results in separate plots for each band of one image. When trying to map the `RasterBrick` object using the package leaflet, mentioned in more detail later in this document, the different bands are not preserved.

It is possible that the leaflet map may have a plugin for the GUI side which allows for selecting different bands to use once the map is created, and that the default is to visualize only the first band. Another solution for this is to convert the data to GeoJSON format, using a GDAL. In GEOJSON, the data would be more easily accessible in a database, and when used in R, vector objects could be created instead of raster objects. Research on this topic says that this would be an optimal solution considering that higher resolution raster images are needing for retaining quality of the image in the map visualization when zooming into the map further, but at that point the file sizes become very large. Converting to GeoJSON would help retain the data in smaller file size, but the resulting visualization would not be of the raw images, and instead color areas on the map based on the values from the original images. R also has other packages called mapview and mapbox that may make it easier to integrate all of the images and various bands.

3.1.4 Chosen Approach

We are going to primarily use rasterio, as it was relatively simple to downsample the data and view the data in each band. Having an array with the respective data is also extremely useful in terms of transforming that into a legitimate database. We will learn the basics of doing the same in R in the case that we need to transition as stated previously.

3.1.5 Proving feasibility

Our plans for moving beyond preliminary analysis are to validate that the data we gained from the raster image was successfully downsampled by displaying a downsampled raster image in a GIS system, such as ArcGIS. From there we would want to make sure we can extrapolate the data from these 3-dimensional arrays. The following section outlines the extrapolation and storage of this data.

3.2 Technological Challenge: Organizing/storing data

- Relational vs. Non-relational database
    - Organization of multi-layered pixel data
- Cloud-optimized GeoTIFFs
- MapServer
- Hosting options

3.2.1 Desired Characteristics

To achieve the high speed and responsiveness needed to make this tool useful to scientists, we want to strictly limit the amount of data that needs to be sent over the internet. This means that, ideally, the processed data which the application uses will be stored on the server where the application is hosted. The question then is, how should the data be organized and stored?

3.2.2 Alternatives

Relational Database (SQL)

One potential solution is to import the processed data into a relational database system like SQL, distributing the bands of data across tables which share an identifying primary key. Such an implementation has the potential to increase the speed with which requested data can be provided to the web application, by breaking it up into component parts and serving it piecewise as the application requests it.

Cloud-Optimized GeoTIFFs

An alternative approach might be to structure our processed data into a format called a “cloud optimized GeoTIFF” (COG). This alternative format for storing GeoTIFF data is designed specifically for serving it over the internet. The data is reorganized in such a way that a web server with the right capabilities can serve data from source files in a piecewise manner, offering a speedup to the service of the data by a similar mechanism to the previous database proposal.

MapServer

A final option in this regard is MapServer, an open-source server-side application for rendering fast-loading image data from large geographical data files like GeoTIFFs. The MapServer software receives query parameters from HTTP requests, queries a datastore on the server-side (like a large

collection of GeoTIFFs, for instance) and returns little images containing only the requested data over the requested area to the web server to give to the client.

3.2.3 Analysis

SQL databases are simple, fast, and reliable. For that reason, it's easy to think that pulling data from our downsampled raster files and copying it into a database for piecewise querying might be a smart move. The key issue with this approach is that the data is already structured from the beginning, within the GeoTIFF files themselves. Taking the data out of the source files and restructuring it into a custom-made SQL database structure seems like it could quickly balloon in complexity with little return on performance.

The COG is very much like a standard GeoTIFF. The major difference between the two is that the data in a COG is reorganized such that a properly equipped web server can easily read only the data it needs from the files it needs, as opposed to all the data from all the files at once. This is intended to significantly reduce the amount of data that needs to load while a user interacts with the application. The one significant advantage to this approach is that it is an already-existing, purpose-built solution to the complex problem of serving large GeoTIFFs over the web. In other words, utilizing the COG format will prevent us from re-inventing the wheel.

MapServer is a specialized server-side application which takes queries from a web request, digs through the underlying data set for what the server is asking for, and builds static images of the requested data spanning the requested area which it returns to the web server. This is a very agile process and could offer us the performance we're looking for. There is, however, a drawback to this approach which can't be ignored: our clients want users to be able to export subsets of the original data for their own research purposes. Given that MapServer just sends back images constructed from the raw data, we might need to build a separate subsystem to collect data for export, which would be redundant and dumb; the ideal situation would be to draw our maps from the same data which users are able to export, for the sake of both simplicity and consistency.

3.2.4 Chosen Approach

The best solution to this challenge is the cloud optimized GeoTIFF. COGs are widely supported by most popular raster processing libraries, and online demonstrations of the COG technology indicate that this format will allow our web application to access and serve data from our large source rasters very efficiently. It makes sense; that's what COGs are designed for. We believe that rendering our downsampled rasters as COGs and building our web application around the COG format will offer the best performance for the smallest cost in development time and effort.

3.2.5 Proving Feasibility

To support this system, we need to look at options for short- and long-term hosting. We will need a server hosting the web application and database, and if we take the cloud optimized GeoTIFF approach, the server will need to support HTTP range queries. While the source raster data files from which we'll be drawing our data are quite large, the hope is that we can reduce the size of the data significantly in our pre-processing step and organize the data for a web server speedup via cloud-optimized GeoTIFFs. In any case, we will need a web server with sufficient onboard storage to host the data in whatever form it ends up taking. Our clients have suggested that they could easily fund an AWS hosting plan for this purpose, and we may pursue that for our short-term development period. Over the longer term, the plan is to "sell" this product to the organization that curates the original

ICEBERG source data, and that that organization would take care of hosting the final product into the future.

3.3 Technological Challenge: Drawing the structured data onto an interactive map

One of the largest requirements for this project is to be able to load roaster images into an interactive map. Currently, the ICEBERG system only allows for scientists to view one raster image at a time and the goal for this project is to load over a hundred raster images onto a web-based map that allows for users to zoom in and out, pan over different areas and to be able to toggle different bands on the map.

3.3.1 Desired characteristics

The desired characteristics for this web application are to be able to quickly load different raster files onto the web page and then be able to toggle different data types once they are loaded. The map needs to be able to offer a large amount of interactivity to the user in a timely manner. This will be accomplished by utilizing an existing JavaScript library which needs to be able to display Raster images on the browser because they are not natively supported like other file formats such as JPG and PNG files. This is very important as scientists need to be able to access this large amount of data quickly and without a proper library, this process will take a very long time.

3.3.2 Alternatives

There are already several existing libraries that allow for raster images to be displayed to a web browser and they also include options for toggling different layers as well as zooming in on a particular area of the map.

Leaflet

Leaflet is a well-supported JavaScript library whose primary purpose is to create mapping applications on the web. This library was created in 2011 and since then has been used by many different companies including Facebook, GitHub and Etsy. Leaflet also includes support for dozens of plugins which can perform different functions, and the most important for this project is to be able to display raster images.

OpenLayers

OpenLayers is another library for creating map interfaces. Unlike Leaflet, OpenLayers immediately seems like a less supported library by the lack of a list of companies who use it in their applications. One benefit for OpenLayers is the support for raster images natively without a third-party plugin which reduce the number of necessary downloads in the final product.

3.3.3 Analysis

Both libraries will be evaluated on a scale from one to five in the following categories, with five being the highest score possible.

	Ease of installation	Ease of access to needed information from library	Community support	Library code Maintainability
Leaflet	4	5	5	5
OpenLayers	3	3	5	3

TABLE 3: SHOWS ON A 1-5 SCALE THE STRENGTHS AND WEAKNESS OF LEAFLET AND OPENLAYERS, WHEN COMPARED TO EACH OTHER

Leaflet

The above table, Table 3 represents values obtained from creating the starter application based off the Leaflet tutorials to display one raster image on the web.

Leaflet is widely supported and has many available plugins that support the visualization of Raster files onto the map in the browser. This is a benefit because there are a large amount of resources and tutorials to help learn about Leaflet and its functionalities properly. Leaflet also provides well documented tutorials and provides many examples to learn more about the library. The installation process is also very quick, as only one CSS and JavaScript file are required to use this library. Leaflet also contains many of the features needed for the project's minimum viable product including togglable layers, zoom, and the ability to draw polygons and other shapes.

OpenLayers

Table 3 represents the concluded results from evaluating the OpenLayers library. Like the above library, these values were based upon creating a simple application to display one raster image in the file.

OpenLayers is another well documented library for displaying maps in the browser however, there are many downsides with choosing this library. One of the first issues is that the example code provided from OpenLayers is much longer and more difficult to understand than Leaflet, which will decrease the project's maintainability over time. The example code from OpenLayers needs 275 lines of code to run whereas Leaflet can do the same but in 78 lines of code. One benefit with using OpenLayers is that it supports Geotiff images without the need for a plugin, but the code is much longer.

3.3.4 Chosen Approach

Based upon the above tables, Leaflet appears to be the better option and the one that we will use moving forward in the project. Leaflet's documentation was not only clearer that OpenLayer's but Leaflet's code was also much more readable and easier to comprehend. This project also requires a codebase that is easy to maintain, which is another reason to choose Leaflet.

3.3.5 Proving feasibility

Moving forward, our team will create more complex programs with Leaflet such as being able to retrieve and display images from a database. Another future goal with Leaflet is to have the ability to load multiple raster images onto a webpage at one time, which is an important goal for this project.

3.4 Technological Challenge: Graphical User Interface (GUI)

The user interface has to be able to interact with the data provided from the backend by having zoom buttons, a search bar to find a specific area in the map and another button or checkbox list to toggle different layers in the raster files. There are dozens of mapping web applications available that will be used to evaluate their overall style and functionality which be used to help design the graphical user interface in this project.

3.4.1 Desired Characteristics

The user interface has to be able to have a lot of functionality readily available without having to navigate to different pages or wait for a long amount of time to update the map after zooming in or out or after searching for a specific area. The graphical user interface must also be easily understandable for any user's that may not have any experience with tools like Google Earth.

3.4.2 Alternatives

There are many different web-based mapping tools available, but currently there are only two projects that really pertain to this project. The first is, Mapping Application for Penguin Population Data (MAPPD). MAPPD is a tool that currently uses OpenLayers to display population data of penguin colonies in Antarctica. Another example is a map from Carbon Brief who used Leaflet to display a large amount of information to show different types of carbon emissions within the United States. Both tools provide many important functions that will be needed in this project.

3.4.3 Analysis

The below image, Figure 3 depicts the current user interface in MAPPD which provides an option for toggling different layers in the upper right corner. The MAPPD tool also provides a query function to search by polygons, species or by specific region, which is something required in this project.

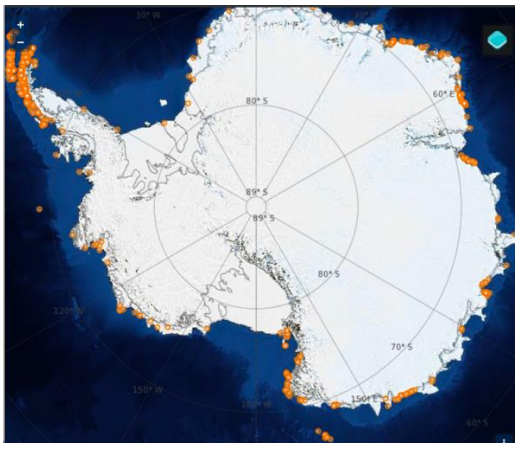

FIGURE 5: MAPPD'S GRAPHICAL USER INTERFACE AVAILABLE AT, HTTP://WWW.PENGUINMAP.COM/MAPPPD.

Figure 5 shows the map displayed from Carbon Brief. Immediately, this design seems much more modern with the vibrant colors and a user interface that is intuitive and does not require much navigating.

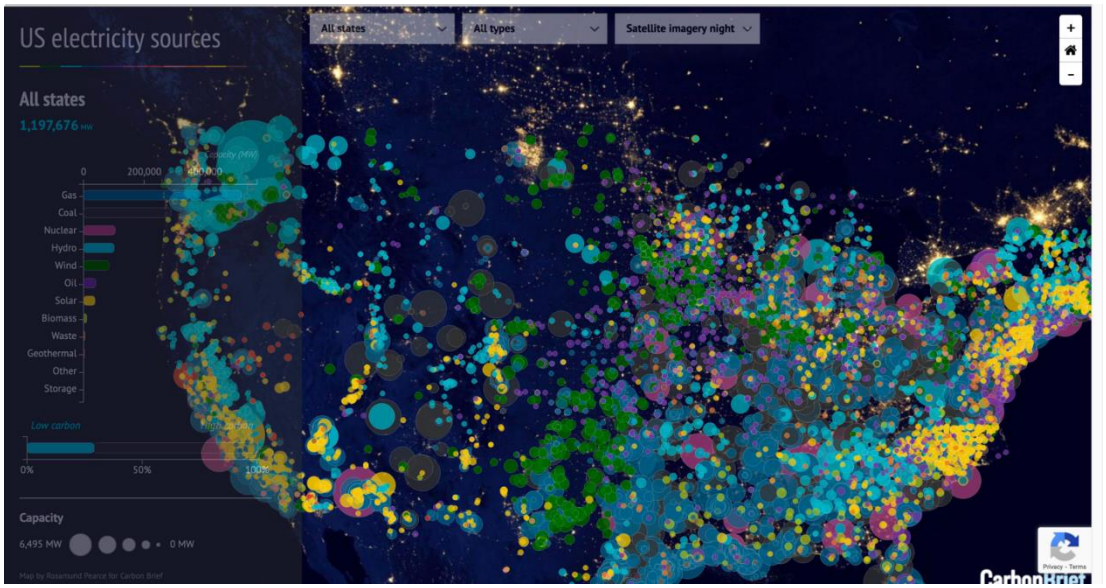

FIGURE 6: CARBON BRIEF'S GRAPHICAL USER INTERFACE AVAILABLE AT, HTTPS://WWW.CARBONBRIEF.ORG/MAPPED-HOW-THE-US-GENERATES-ELECTRICITY.

Another benefit with Carbon Brief's example is that Leaflet is used to display the data and a GitHub repository is given which will be a useful resource in learning about how other projects use this library. Similarly, like MAPPD Carbon Brief also provides tools for toggling layers which can be seen in the upper middle of Figure 6. These allow users to filter by states, carbon types and satellite imagery types as well. Even though Carbon Brief does provide functionality for filtering, they do not provide a query tool which may be a better option instead of having long dropdowns to scroll and choose from.

3.4.4 Chosen Approach

The chosen approach for the user interface in this project will be a combination of both MAPPD and the Carbon Brief map. While Carbon Brief's map is more visually appealing, they do not provide some of the necessary functionalities that MAPPD does. For this reason, our finished product's interface will include a search bar and toggling button similarly to MAPPD and create a map like Carbon Brief to design a more modern looking application.

3.4.5 Proving feasibility

To prove the feasibility of merging ideas from both applications, the process will be broken up into smaller sections. The first step will be to include a toggle button that can call different bands from the raster images and then, a zoom button can be added to zoom in or out on the map. Finally, a search query tool will be added that can filter all the raster images by their location in Antarctica.

3.5 Technological Challenge: Exporting Data from maps

We will need a way for users to download the raster images from the map. This is important because there is other software that exists offline that allows for more individual analysis but doesn't allow for the view of the whole continent. There will be different ways to download images, users will either

be able to export raster images with just the bands they have been looking at or they can export the raw data with all 18 bands.

3.5.1 Desired Characteristics

The desired way for raster images to be exported is to have users use a selection tool and then get one raster image with just that area on it. Having the ability to export only the selected image, allows scientists to focus on a particular area on our map. Then when looking at the exported image, they do not need to refocus because they only have what they selected.

3.5.2 Alternatives

The main alternative to a mosaic of different raster images is to have all the selected images merged into one raster image. With this method we would have a message appear that lets the user know the latitude and longitude of the corners that they originally selected, with this information available they will be able to find what they were looking at before. We would still be able to export one raster image with the entirety of the image in it.

3.5.3 Analysis

The main issue is having a specific area of a map be exported rather than the entire dataset. Since users will be able to select sections that contain more than one raster image, they need to be cut apart. It is possible to break up raster images into sections of the original, the problem arises from breaking them up into non-uniform sections. The main way for raster images to be broken up is to base them off a pre-existing image of squares, this is not possible to have since any area can be selected.

3.5.4 Chosen Approach

The approach we are going to use is merging the complete raster images together from the selected section. We are going to do this because there is not a good way to ensure the user gets desired sections otherwise. We will then have the coordinates they selected available to them, so they are able to refocus on what was being looked at.

3.5.5 Proving feasibility

The main problem is having only one image exported of the selected area. This is easily achieved by a pre-existing R function that will merge raster images that have the same origin and resolution, which all our will. The next challenge is to display the coordinates of the corners selected. This is possible because every pixel of the raster images has its latitude and longitude coordinates as part of their data.

Technology Integration

All the challenges above present a unique hurdle but once they are implemented, we just need to integrate them together to have a fully-fledged web application. These solutions were decided by their ability to merge with other components of the system like connecting Python to AWS and then sending the information to the front end.

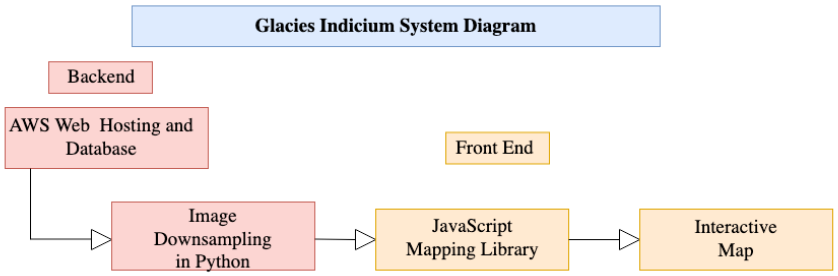

FIGURE 7: CURRENT SYSTEM DIAGRAM DESIGN

The solutions moving forward will be merged in a linear order. As seen in figure 7, the data organizing and storing will be handled on the AWS server and will be called by the image downsampling classes created in Python and the rasterio library. After the image's sizes are done processing, the JavaScript library, Leaflet will take the downsampled files to quickly display an interactive map to the user. Once displayed, the user can zoom in or out, toggle different bands in a raster image, and query a specific region in Antarctica.

Conclusion

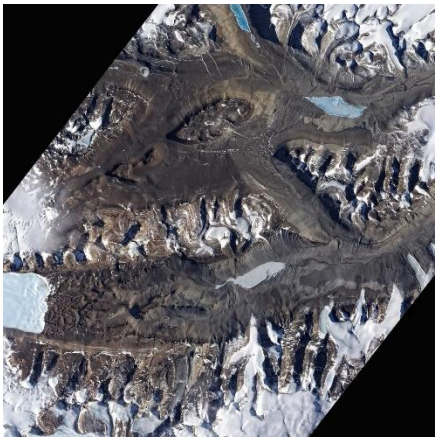

FIGURE 8: MAP OF DRY VALLEYS

The Antarctic is home to much geological diversity and scientists around the globe, particularly ones utilizing data related to climate change, could really benefit from being able to analyze it on the continental scale. If our product is successful it will be sold to the Polar Geospatial Center and scaled up from just the Dry Valleys to the entire continent of Antarctica. As is evident from the above sections we've identified the major obstacles that lie in this project's way. The next step is to prove that each of our solutions are feasible. In summary, we need to ensure that we can successfully downsample an image from GB size to the MB range and to test this we can display our newly downsampled raster in a map projection program such as ArcGIS. From there we need to take the data we've gathered from multiple downsampled images and successfully organize it in either a database or a Cloud-Optimized Geotiff file. We then need to take this data and project it onto the Dry Valleys region of Antarctica. We can test this by comparing sections of our projected map with the projections of the original raster image. If our 'muddy' projection matches up with the original data, we will be successful. Features that we outlined in the GUI sections would then need to be implemented so the data would be easily filterable. Lastly, we need to export sections of our projections so that scientists could use this data in the field, as different areas may serve as inspiration for new expeditions. From here scientists across the world could use this application to further their research and embark on new endeavors to better understand the region.