Retrieving Iceberg Characteristics From Satellite Images

Nuzhat Khan
Hosts: Olga Sergienko, Alex Huth, Alistair Adcroft
{nuzhat.khan, olga.sergienko, alexander.huth, alistair.adcroft}@noaa.gov

1 Introduction

Large tabular icebergs are a major source of ice mass loss in the Antarctic Ice Sheet. These icebergs, which are typically considered to be those that are greater than $3km^2$, vastly influence the climate, as they are the primary contributors of iceberg meltwater in the ocean. Despite being a major component of ice-sheet mass loss and significant source of freshwater, large tabular icebergs have not been well represented in climate models because of a greater focus on smaller icebergs. Since these two classes of icebergs behave quite differently from each other, this opens a window for inaccuracy in current models. Unlike small icebergs, which calve frequently and fracture infrequently, large tabular icebergs behave conversely, calving infrequently and fracturing frequently. They fracture mainly through collisions and the footloose mechanism, which describes the slow shedding of small icebergs from the sides of the giant tabular iceberg.

Observational data can be useful to further the understanding of the behavior of giant tabular icebergs. Research has been done to track the changes of the icebergs throughout their lifetimes. Namely, the data collected by the National Ice Center (NIC) and Brigham Young University (BYU) are the most notable examples of research on large tabular icebergs. The NIC gathered data by calculating iceberg characteristics manually, whereas BYU took a more streamlined approach by using scatterometer data to do calculations.

These two sources formed the basis for my project, as my goal was to use MODIS and VIIRS satellite images to measure iceberg characteristics. Since satellite imagery provides a boundless array of observational data, they can be manipulated to isolate icebergs of interest and track changes over time. The disadvantage of using satellite imagery is that there is often cloud cover that interferes with measurements. To overcome this, I introduce automated methods in Python that mask out cloud cover while retaining the iceberg. This image processing allows for the quantitative analysis of iceberg breakup.

2 Data

2.1 Data Collection

Prior to image processing, I first had to find a way to download large amounts of satellite images. NASA Worldview provides a snapshot tool that can be used to download images from specific dates and in specific locations. A download link can be generated for any unique image, so constructing links by filling out information for each field available on the site (e.g. projection mode, location) would allow for automated download. Github user Leifdenby had already developed a method for downloading geographically projected images via the NASA snapshot tool, so I adapted his program to suit the download of

GEOtiff images in antarctic projection mode. I created a similar method in which I started with pre-filled information in a base URL, and parameterized other information such as dates and coordinates. Instead of downloading images individually, I developed another method to loop through BYU and NIC data, which I had downloaded as CSV files from their website, and fill in any missing fields. Since the spatial coordinates in the CSV's were all lat/lon coordinates, I used the Pyproj library to convert from the lat/lon EPSG:4326 code to the desired polar stereographic EPSG:3031 code. Data retrieval is certainly a slow process, but compared to requesting downloads for satellite swath products in HDF format, this method was faster.

2.2 Data Types

Following data retrieval, I began the image processing step. Rasterio, a library based on GDAL, was another library that I used to open GEOtiff images as a preliminary step to image processing. Rasterio is a useful tool to look at the metadata contained in the GEOtiffs and is an easier interface to use than GDAL. Initially, the most important pieces of information that I gathered from Rasterio were the coordinate reference system (CRS) of the image and the bands that comprised the image. The CRS of all the images I downloaded was EPSG:3031, part of the WGS84 reference system. The unit of measurement was meters, so each pixel coordinate in the image could be converted to coordinates measured in meters on the CRS.

The bands in the image were different from each satellite product. In total, I worked with seven different satellite products. The products are as listed:

MODIS Corrected Reflectance (Bands 3-6-7)
MODIS Corrected Reflectance (Bands 7-2-1)
Terra MODIS Corrected Reflectance (True Color)
Aqua MODIS Corrected Reflectance (True Color)
VIIRS Suomi NPP Corrected Reflectance (True Color)
VIIRS NOAA-20 Corrected Reflectance (Bands M11-I2-I1)
VIIRS NOAA-20 Corrected Reflectance (True Color)

Based on the bands in the images, the products could be separated into three different groups, true color images, images in which icebergs and clouds both appear blue (MODIS 7-2-1 and VIIRS M11-I2-I1), and images in which icebergs appear red (MODIS 3-6-7). Each band of an image is associated with a different wavelength, and the combination of these wavelengths results in these groups. Though I processed the three groups in the same way, I used different color bands when opening them with Rasterio. For MODIS 3-6-7, band 3, which is the red band, was isolated. for MODIS 7-2-1, band 1, which is the blue band, was isolated. This was also the case for VIIRS M11-I2-I1. For the true colors, all color bands were taken and binarized.

3 Method

3.1 Image Processing

Once I correctly opened the images and gathered relevant metadata, I processed the images using various algorithms from the OpenCV library. The first algorithm I applied was a binary threshold, which sets all pixels with intensity values below a specified threshold to 0, and anything above that threshold to 255. Through trial and error, I determined that a reasonable threshold for most images was 210, though I created a parameter to change this value in case the iceberg's pixels were far less intense than 210. This step removed a significant amount of cloud cover but often left fragmented white spots (fig. 1-2). A connected component analysis, which uses graph theory to determine the connectivity of white 'blob-like' regions of an image, helped remove these spots. The blobs that exceeded a set pixel area of 800 were filtered out, effectively removing small fragments (fig. 3).

Though the analysis was a necessary step for most images, it sometimes resulted in unwanted holes in the iceberg. I used image dilation to mend these holes. Image dilation expands all the white parts of an image but can make pixels appear rough and blockish (fig. 4). An image blur after the dilation helped to smoothen it out (fig. 5).

Finally, I used an Otsu threshold to further separate the foreground and background of the image (fig. 6). This algorithm is an expensive step in the process, as it involves generating a histogram of pixel intensity peaks to automatically determine a threshold for the foreground and background separation.

3.2 Iceberg Isolation

Ideally, the image processing steps would leave only one or a few elements of the original image, the largest one being that of the iceberg. OpenCV provides a findContours function that captures the contours of these elements. These contours are matrices of pixel coordinates. I used the contourArea function to identify the contour with the largest area, then masked out anything except for that contour (fig. 7-8). There were many images for which this produced unsatisfactory results. Often, cloud cover that was not removed during the image processing step would end up as the largest contour (fig. 9-10). To remedy this issue, I performed a round of reprocessing. I calculated the median once I obtained the areas of a particular year. During the reprocessing, I used the median as an area threshold for the contours and specified an area cut-off value that none of the contours could exceed. This cut-off was usually 2,000 added to the median. Since reprocessing could take quite a bit of time, I avoided it unless I saw that the median of a certain satellite type's dataset greatly differed from the datasets of the other satellite types in that same year.

The image mask could be used to mask out the same area in the original image to display the result, though this step was not necessary for area calculation. To calculate the pixel area, I simply summed the non-zero pixels of the mask.

3.3 True Area Estimation

Aside from pixel area, I developed a few different methods to estimate true area. For true area estimates, the mask was not

needed and the contour matrix was sent to different functions. The first area calculation method I developed was a very simple one, in which I used the pixel area, image dimensions, and the distance covered by the bounding box during download (300km x 300km) to estimate the area.

The second method involved calculating the geodesic area using the Pyproj and Shapely libraries. I first converted the pixel coordinates to polar stereographic coordinates using Rasterio's transformation matrix. The transform method uses the CRS to map the pixel coordinates to coordinates on a map. From what I read online, I discovered that area calculation is very difficult with polar stereographic projection because it is not area-conserving and the grid is non-uniform. Most sources I came across recommended converting to a uniform grid to calculate area. For this reason, I converted the polar stereographic coordinates to latitudinal and longitudinal coordinates. Nevertheless, I also created a separate third method in which I directly created a geoJSON polygon using the polar stereographic coordinates and calculated area in kilometers using the Shapely library. As for the second method, I created a geoJSON polygon using the reprojected coordinates, projected this polygon onto a WGS84 ellipsoid, and calculated area using the Shapely library. Shapely does not take map distortions into account.

My host, Alex, examined the contours produced by the coordinates of the lat lon method and found that the contours had a flipped orientation. This was due to the polar stereographic coordinate transformation, in which I purposely flipped the x and y coordinates because of suggestions from online sources, including the University of Minnesota's PGC coordinate converter. I created separate methods with unflipped x and y coordinates for both the second and third methods to check whether I would get different results.

4 Results

Using the methods I developed, I was able collect yearly data for all of the satellite image products. I plotted the data to determine whether I could detect fracturing events of iceberg A68a. Figures 11-17 display the pixel counts in 2020 for all seven of the satellite products. Most pixel areas fell between 6,000-9,000 pixels. There was a negative trend across the year for each of the graphs, though it was clearer in some products than in others. In December 2020, there was a dramatic decrease in the data, as the pixel areas fell more below 6,000. This signaled a fracturing event, so I observed the contours closest during this month to confirm the event. Figures 30-33 show the iceberg's contours from December 10, 2020 to January 6, 2021. In the beginning of the month, the iceberg just rotated, but by the end, the finger-like extension had broken off, resulting in A68e.

I compared the pixel time series to the time series of the true area estimates. Figures 18-21, 22-25, and 26-29, show the areas calculated using polar stereographic coordinates, lat lon coordinates, and flipped lat lon coordinates, respectively. The results were similar for these methods, with most of the areas falling between 5,000-7,000 km^2. The polar stereographic method was a bit

different from the other methods in that it yielded lower estimates. All showed that there was an overall negative trend in the data, just as the pixel areas had shown, though this trend is less consistent across the year.

5 Discussion

5.1 Time Series

Despite the data products producing similar plots overall, there were two that were quite different from the other products. It is clear from the plots (figs. 12, 15, 19, 23, & 27) that the blue channel images produced especially poor results. For all of the methods used, the plots were much sparser than the other products. This is because the blue channel images were challenging to process. The clouds and icebergs were usually the same color and intensity, so many images had large chunks of cloud cover still present even after processing. The exaggerated area estimates resulting from these images were removed from the final dataset, resulting in sparse graphs.

5.2 Area Calculation

Although I used pixel areas to detect fracturing events in my end product, for most of the duration of my internship, I dealt with true area estimates. The results using the true area estimates were fair, but there were several potential issues with it, the main one being area distortions. If an iceberg happened to have a greater trajectory (such as iceberg B15), the area estimates would become more inaccurate due to distortions in the coordinate system. Even the lat lon method did not properly account for these distortions, and introduced other sources of inaccuracy due to the reprojection of the coordinates and its sensitivity to rotation. The area estimates could be slightly different depending on the iceberg's position despite retaining its shape.

To solve the problem of distortion, what I hoped to do was calculate the true area covered by each individual pixel in the iceberg and sum them, but upon contacting Ryan Boller, the head of the NASA snapshot team, I realized that this would not be straightforward.

The images are composed of many different image swaths projected onto a large map projection. The swaths may also have their own projection modes before being projected onto the map. These factors affect the "purity" of the spatial resolution, making the determination of individual pixel resolutions an extremely arduous task. In short, calculating area using the NASA snapshot GEOtiffs is a very difficult task.

Since the pixel counts were more reliable and produced more consistent plots, in terms of detecting fracturing events, they sufficed.

5.3 Sources of Error

I have already mentioned the sources of error for the true area estimation methods, which is that they are susceptible to inaccuracy resulting from distortions, rotations, and reprojections (in the case of the lat lon method). Other sources of error arise from the data products themselves. For all of the plots, the data was quite sparse in the middle of the year. Those months are during austral winters, so the satellites don't really capture anything in the Antarctic. Most images during these months were

zero-value images, simply meaning that they are blank. Of course, this lack of data didn't affect the plots of the blue band images very much, as those were already quite sparse. This is a significant source of error because a fracturing event during the winter would go unnoticed if there's not enough data to detect it.

I attempted to test iceberg B15 to determine how severe this error truly is, but I faced a few complications when doing so. The data for B15 spans over a decade, meaning that I spent a lot of time just downloading the images. On top of this, B15 spends a lot of time around Antarctica, meaning that it's around other huge pieces of ice. It was challenging enough to separate the iceberg from cloud cover; if the Antarctic ice shelf also appeared in the image, it was almost guaranteed that the ice shelf would be part of the iceberg contour because they had the same pixel intensities. Significant time went into downloading years' worth of data for B15, such as the data for 2003 and 2004, but only upon processing the images did I realize that the data wouldn't be consistent because the iceberg was still too close to the ice shelf. Thus, this revealed another source of error; the image processing methods I used may be limiting when applied to different icebergs.

5.4 Future Work

For future work, it is important to address the sources of error. For the area distortion issue, a potential solution may be to figure out what swaths make up the images and find the spatial resolution within those swaths. I heeded Ryan Boller's advice to look into level 2 and 3 swaths, but it simply simply took too much time to hone in on the location and time period I was interested in. NASA's Earthdata Search tool would get very slow as I filled in more fields to narrow down my search, and selecting the location froze the application altogether. The results that I was able to obtain were in the form of hdf files instead of GEOtiffs, which need to be opened and processed differently. So, finding a solution to the area distortion might be an entirely new project itself.

The other source of error to address is the lack of data during austral winters. An improvement to this project would be to discover a satellite that captures images during the austral winters.

Lastly, a major improvement to the project would be the incorporation of machine learning. At the start of the project, I wanted to accomplish something similar to the Photoshop select subject tool, which is hosted on Adobe Sensei, Photoshop's AI platform. Currently, it can't detect icebergs very well, but this is due to the fact that the tool is trained mainly on human subjects. Perhaps something trained specifically on iceberg images would yield better results. Since there was no data to train a model on at the start of my project, I could not pursue this route. Now that there is a way to download large amounts of data and process them, machine learning would be interesting to look into.

6 Conclusion

To summarize, I developed automated methods to retrieve bulk data from the NASA snapshot tool and process this data using various image processing techniques. I calculated the pixel areas of the

icebergs within the images and plotted time series, finding that blue channel data products produced the worst results and the true color data products produced the best results. The time series helped me to detect fracturing events, which I was able to confirm by taking a closer look at the contours.

Appendix

Fig 1. *Modis 3-6-7 image of A68a on 20/03/02*

Fig 3. *Modis 3-6-7 image of A68a on 20/03/02 after connected component analysis*

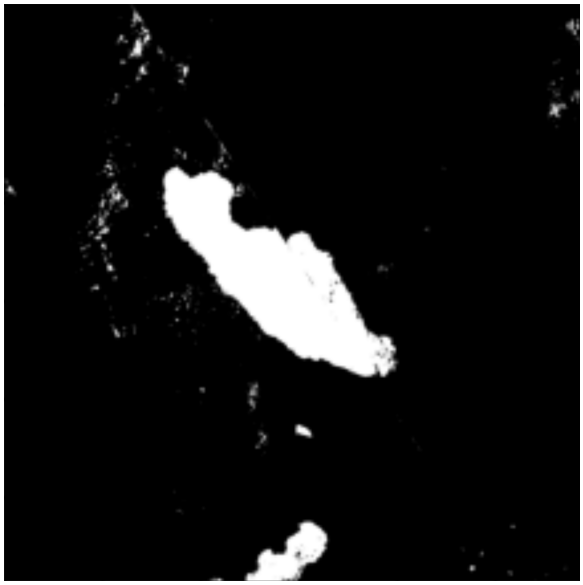

Fig 2. *Modis 3-6-7 image of A68a on 20/03/02 after binary threshold*

Fig 4. *Modis 3-6-7 image of A68a on 20/03/02 after image dilation*

Fig 5. *Modis 3-6-7 image of A68a on 20/03/02 after image blur*

Fig 6. *Modis 3-6-7 image of A68a on 20/03/02 after Otsu threshold*

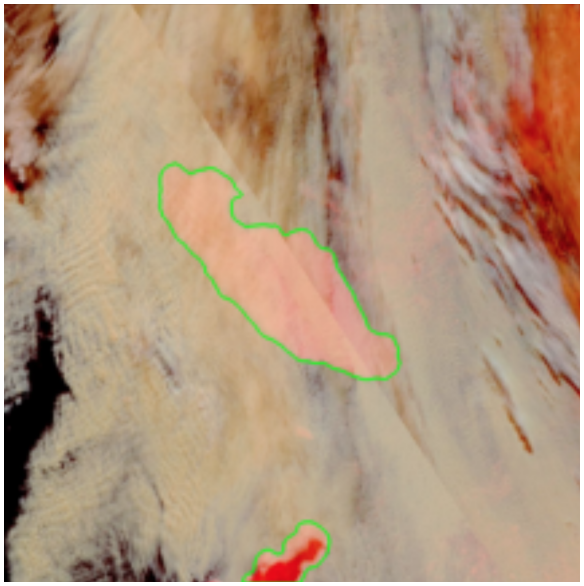

Fig 7. *Modis 3-6-7 image of A68a on 20/03/02 with detected contours*

Fig 8. *Largest contour captured in Modis 3-6-7 image of iceberg A68a on 20/03/02*

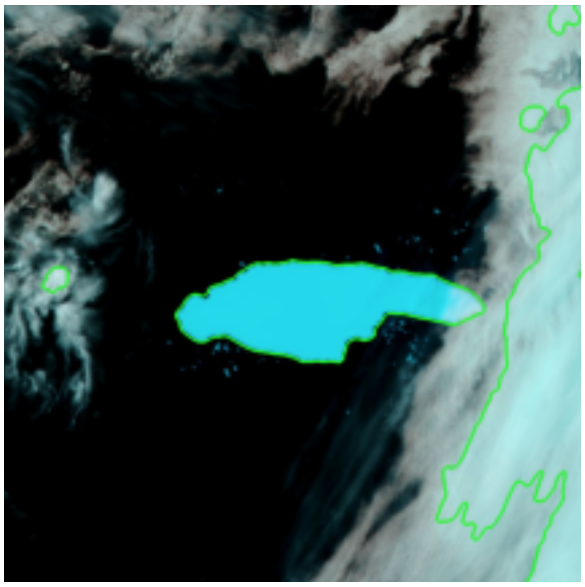

Fig 9. *VIIRS NOAA M11-I2-I1 image of A68a on 20/10/19 with detected contours*

Fig 10. *Largest contour captured in VIIRS NOAA M11-I2-I1 image of A68a on 20/10/19*

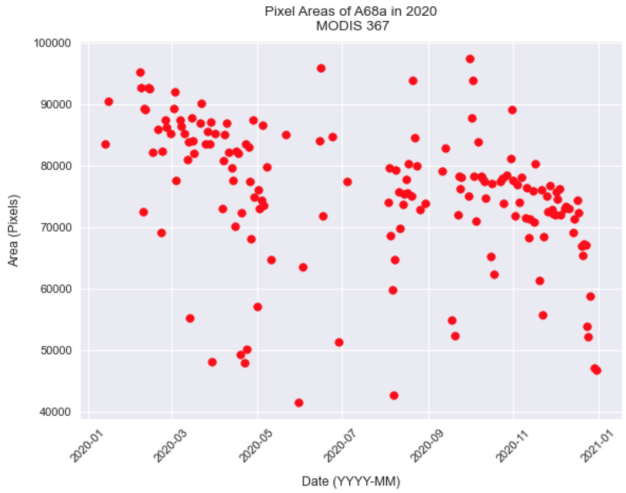

Fig 11. *Pixel counts of A68a in 2020 using MODIS 3-6-7*

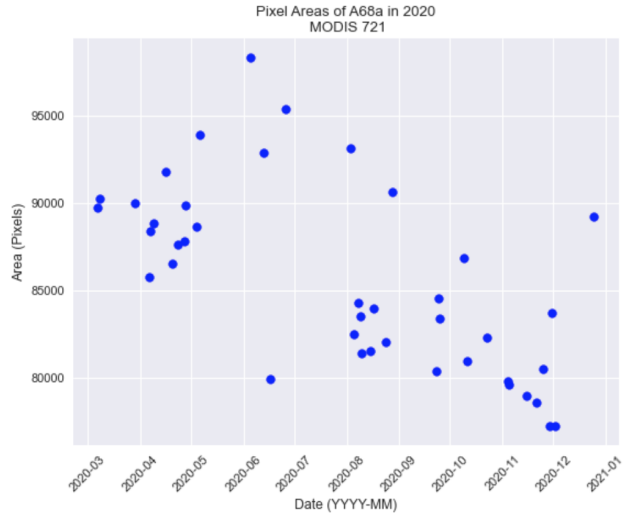

Fig 12. *Pixel counts of A68a in 2020 using MODIS 7-2-1*

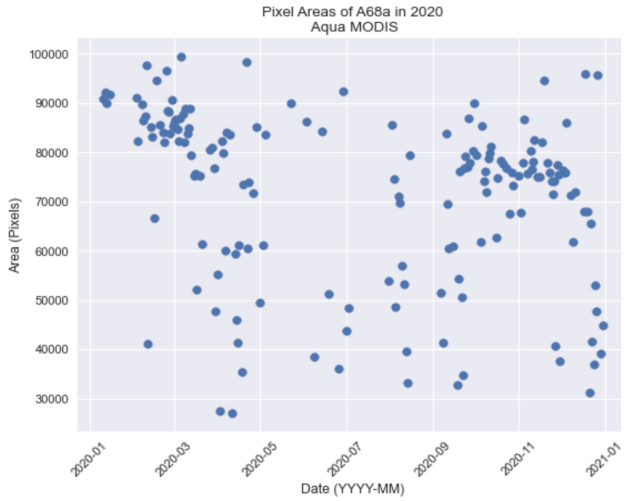

Fig 13. *Pixel counts of A68a in 2020 using Aqua MODIS*

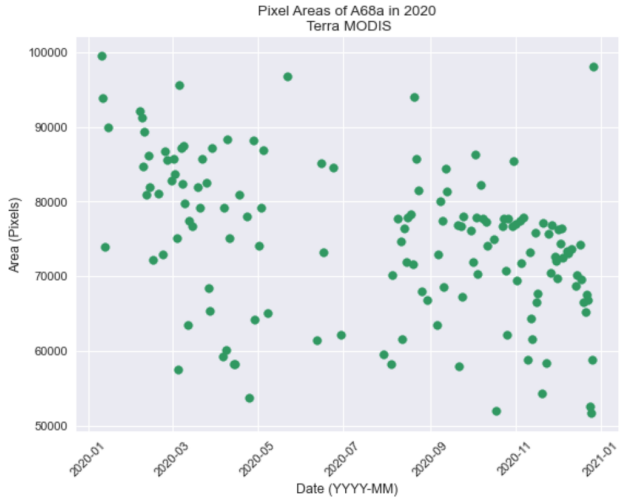

Fig 14. *Pixel counts of A68a in 2020 using Terra MODIS*

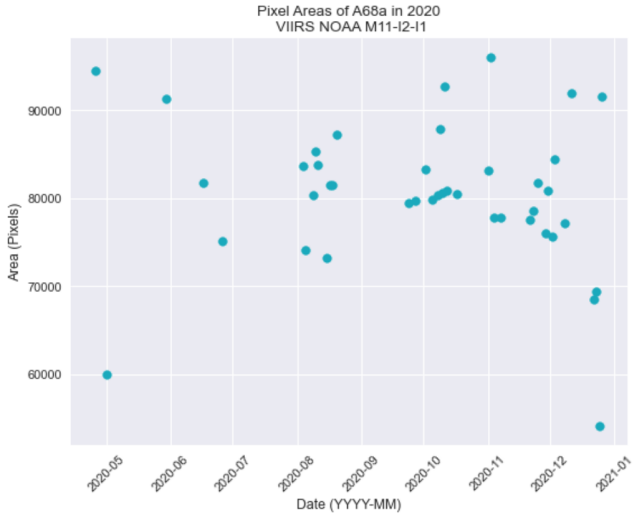

Fig 15. *Pixel counts of A68a in 2020 using VIIRS NOAA M11-I2-I1*

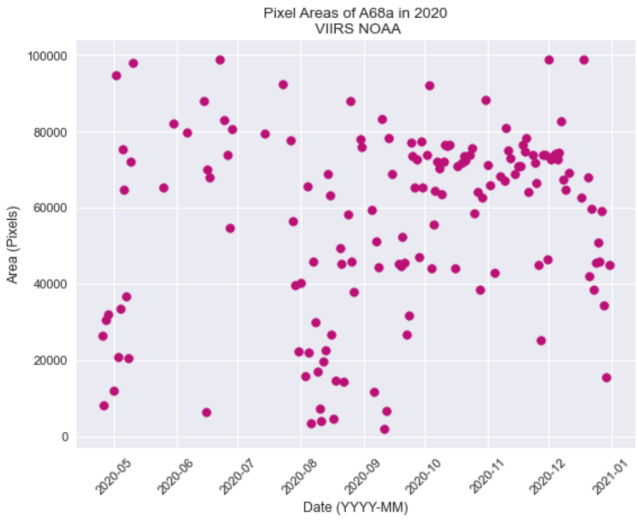

Fig 16. *Pixel counts of A68a in 2020 using VIIRS NOAA*

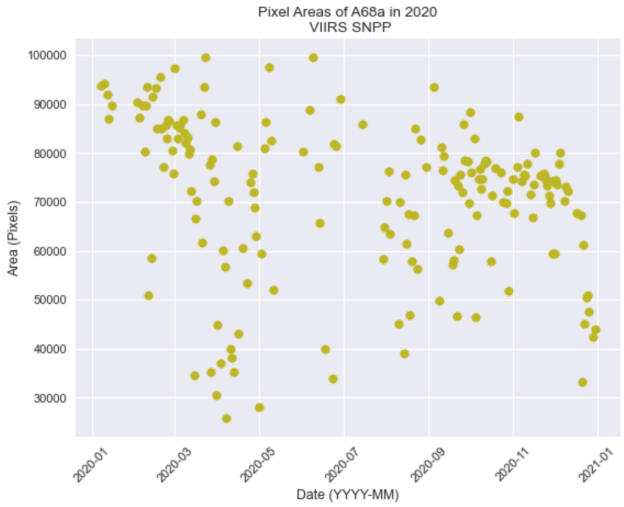

Fig 17. *Pixel counts of A68a in 2020 using VIIRS SNPP*

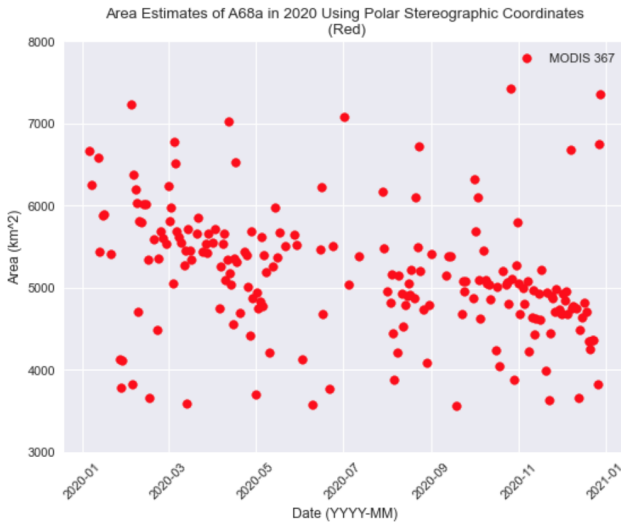

Fig 18. *Estimated true area (km²) of A68a in 2020 using MODIS 3-6-7 and polar stereographic coordinates*

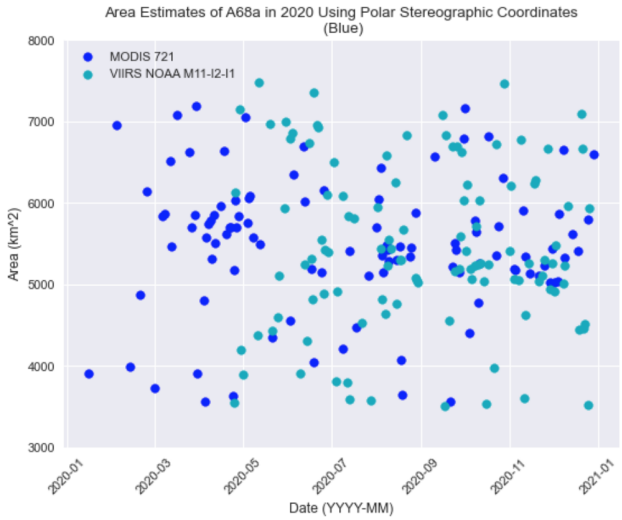

Fig 19. *Estimated true area (km²) of A68a in 2020 using blue channel images and polar stereographic coordinates*

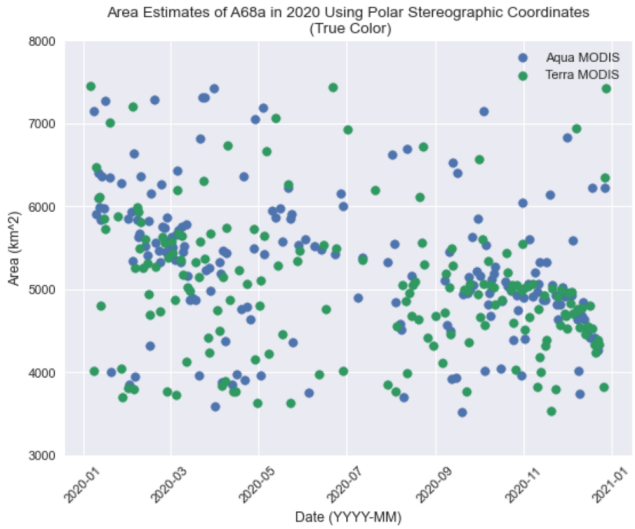

Fig 20. *Estimated true area (km²) of A68a in 2020 using true color images and polar stereographic coordinates*

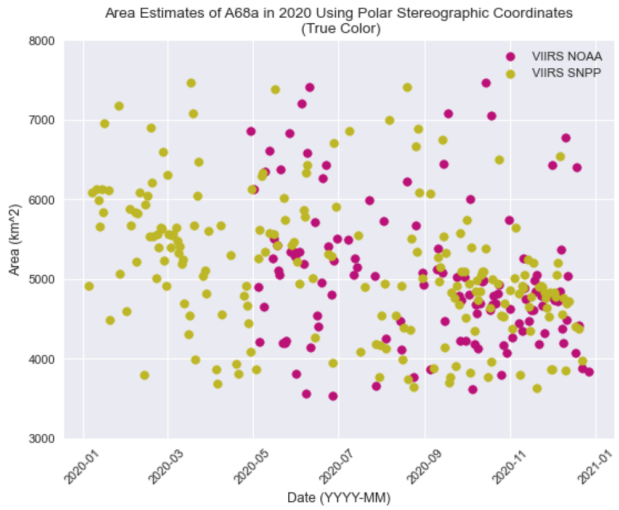

Fig 21. *Estimated true area (km²) of A68a in 2020 using true color images and polar stereographic coordinates*

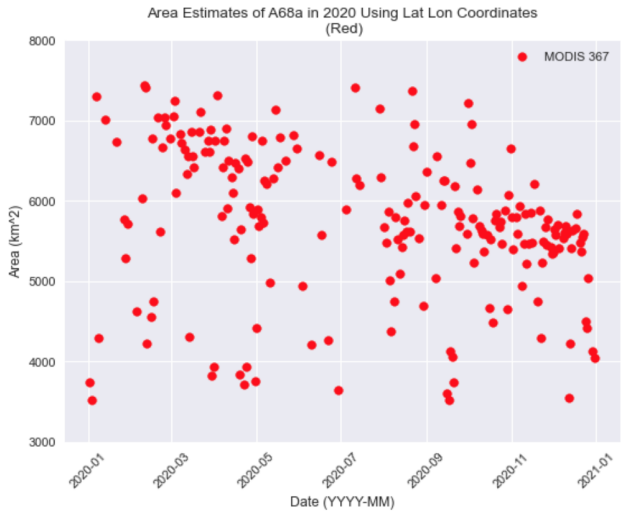

Fig 22. *Estimated true area (km²) of A68a in 2020 using MODIS 3-6-7 and lat lon coordinates*

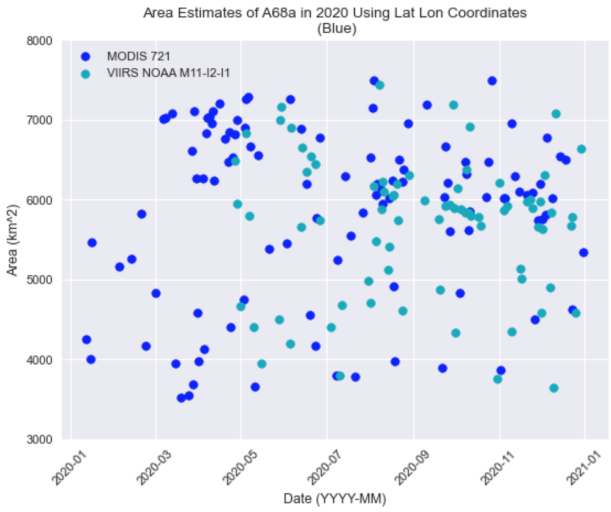

Fig 23. *Estimated true area (km²) of A68a in 2020 using blue channel images and lat lon coordinates*

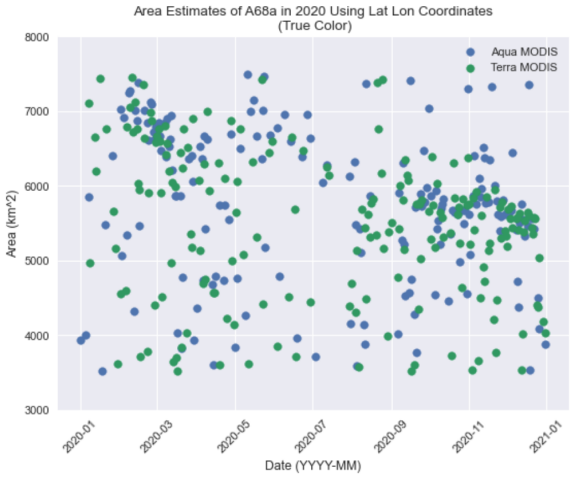

Fig 24. *Estimated true area (km²) of A68a in 2020 using true color images and lat lon coordinates*

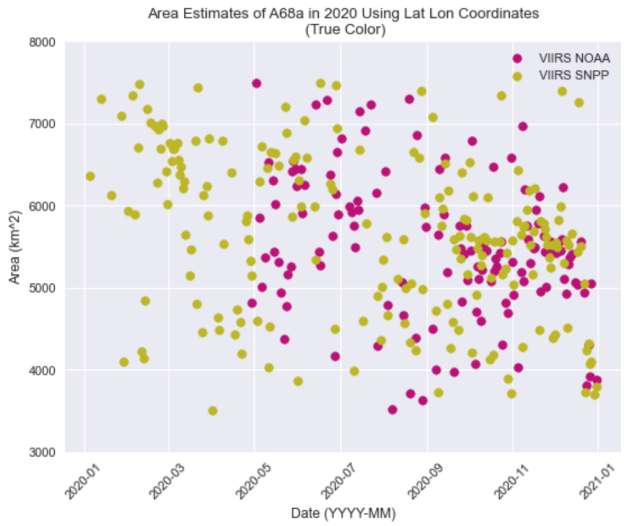

Fig 25. *Estimated true area (km²) of A68a in 2020 using true color images and lat lon coordinates*

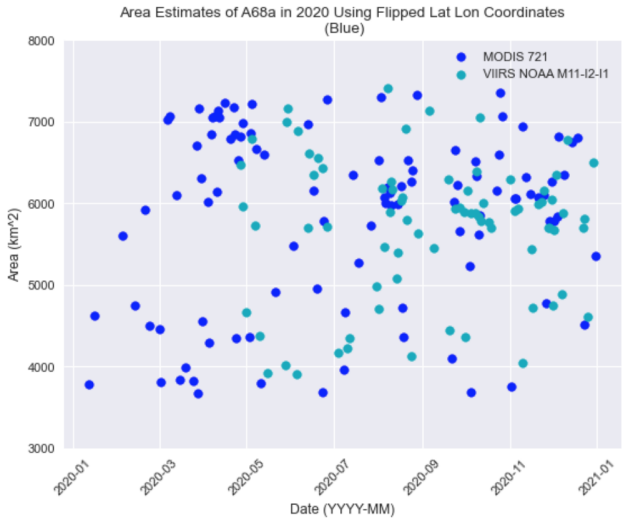

Fig 27. *Estimated true area (km²) of A68a in 2020 using blue channel images and flipped lat lon coordinates*

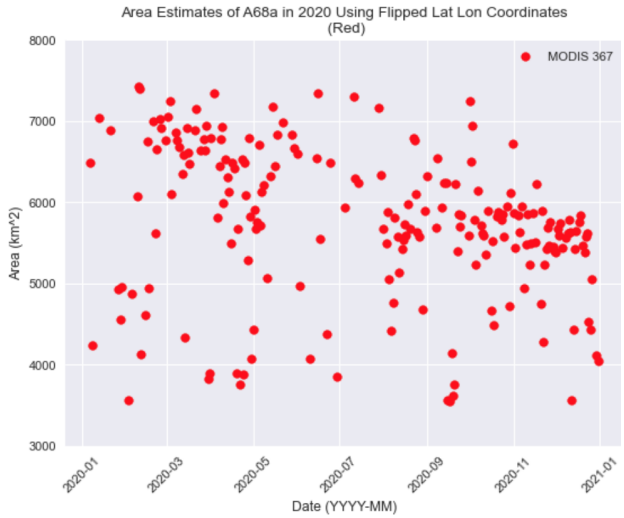

Fig 26. *Estimated true area (km²) of A68a in 2020 using MODIS 3-6-7 and flipped lat lon coordinates*

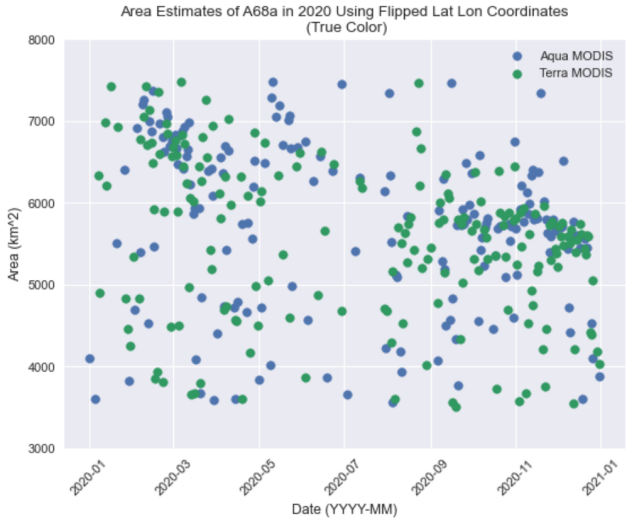

Fig 28. *Estimated true area (km²) of A68a in 2020 using true color images and flipped lat lon coordinates*

Fig 29. *Estimated true area (km^2) of A68a in 2020 using true color images and flipped lat lon coordinates*

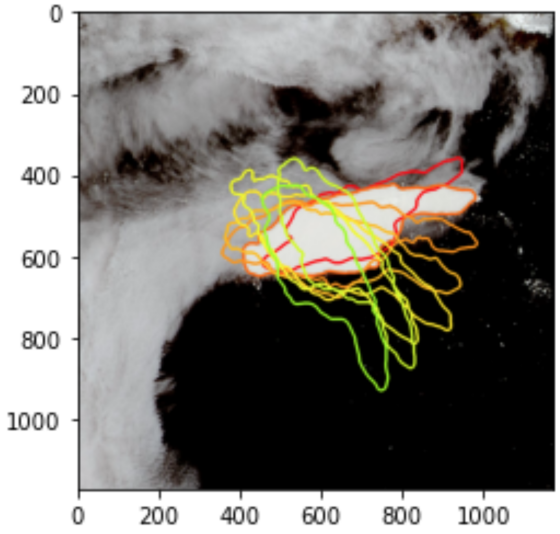

Fig 30. *Contours of A68a from 20/12/10-20/12/17*

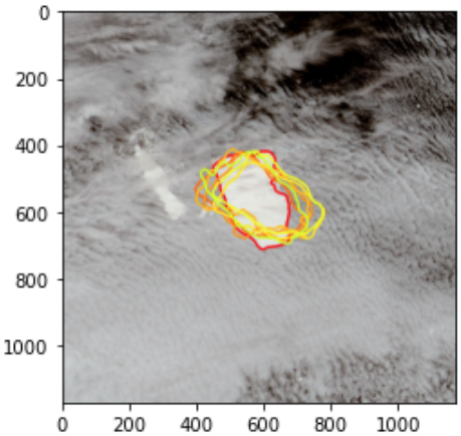

Fig 33. *Contours of A68a from 20/12/30-21/01/06*

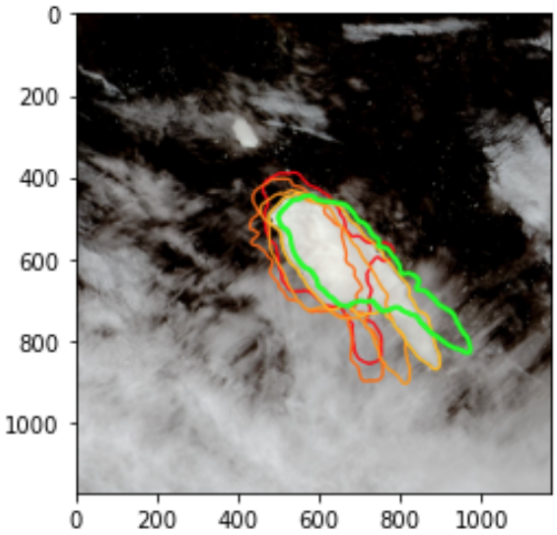

Fig 31. *Contours of A68a from 20/12/18-20/12/22*

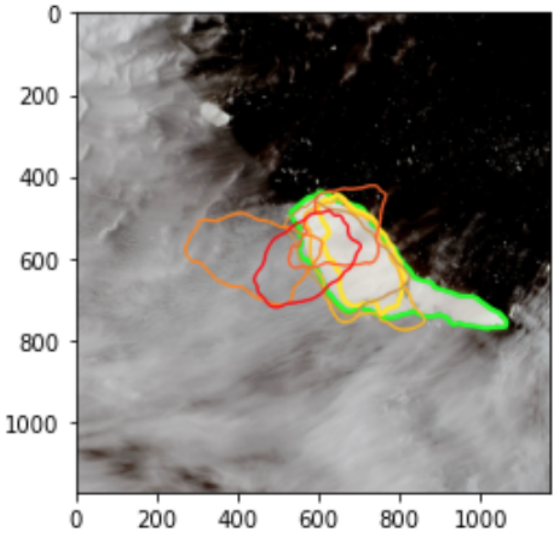

Fig 32. *Contours of A68a from 20/12/23-20/12/29*