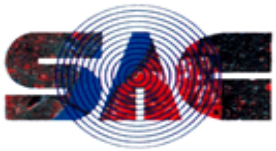


ICEBERG DETECTION & TRACKING

Report submitted for completion of training under

(TREES)

(**T**raining and **R**esearch in **E**arth **E**co-**S**ystem)
Programme of Space Applications Centre

Submitted by
JAY GORAKHIYA
(Registration No. 18CEUOS111)
B.Tech. (Computer Engineering)

Under the guidance of

Ms. Purvee Joshi & Mr. Naveen Tripathi
SCI/ENG – SD
HTDD/AMHTDG/EPSA
Space Applications Centre, (ISRO) Ahmedabad

Institution
Dharmsinh Desai University
Nadiad – 387001
Gujarat

SRTD-PPG
Space Applications Centre (ISRO)
Ahmedabad, Guajrat

06 December 2021 to 06 April 2022

Iceberg Detection and Tracking

By

Gorakhiya Jay Jagdishbhai (18CEUOS111)

A project submitted

In

partial fulfilment of the requirements

For the degree of

Bachelors of Technology

In

Computer Engineering

Internal guide

Prof. M.S. Bhatt

Associate Professor

Dept. of Computer Engg.

External guide

Ms. Purvee Joshi and Mr. Naveen Tripathi

Scientist SD and Scientist SD

Space Application Centre (ISRO)

Faculty Of Technology

Department of Computer Engineering

Dharmsinh Desai University

April 2022

Certificate

This is to certify that the project work titled

Iceberg Detection and Tracking

Is the bonafide work of

Jay Gorakhiya (18CEUOS111)

Carried out in the partial fulfilment of the degree of Bachelor of Technology in the Computer Engineering at Dharmsinh Desai University in

the academic session

December 2021 to April 2022

Prof. M.S. Bhatt
Associate Professor
Dept. of Computer Engg.

Dr. C.K. Bhensdadia
Head,
Dept. of Computer Engg.

Faculty Of Technology
Department of Computer Engineering
Dharmsinh Desai University
April 2022

Acknowledgement

The opportunity I had with Dharmsinh Desai University was a great chance for learning and professional development. Therefore, I consider myself as a very fortunate individual as I was provided with an opportunity to be a part of it. I am also grateful for having a chance to meet so many wonderful people and professionals who led me though this period.

I express my deepest thanks to **Dr. C.K.Bhensdadia** (HOD, Computer Engineering department), **Prof. Malay Bhatt** (Associate Professor) for taking part in useful decision & giving necessary advices, guidance and arranged all facilities to make this journey easier. I choose this moment to acknowledge their contribution gratefully.

It is my radiant sentiment to place on record my best regards, deepest sense of gratitude to **Ms. Purvee Joshi**(Scientist SD, Space Application Centre (ISRO)) for their careful and precious guidance, which were extremely valuable for my study, theoretically and practically.

I perceive as this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work and try to improve, in order to attain desired career objectives. Hope to continue cooperation with all of you in the future.

Table of content

Figure table

List of Tables

Chapter 1

Introduction

1.1 Polar Region

Polar region (also called frigid zone) is the area around North Pole and South Pole. Northern polar region is called the Arctic and the Southern Polar region is called the Antarctic. Characteristics of polar areas include: **Climate - long cold winters, with annual temperatures mostly below freezing**. Polar areas are often windy, with very little precipitation. Permanent ice caps cover polar landscapes. The Arctic is mostly an ocean surrounded by land. The Antarctic is mostly land surrounded by water.The ice cap of the **southern polar region** averages **6,700 feet** (about 2,000 m) in **thickness**, is underlaid by the continental landmass of Antarctica, whereas pack **ice in arctic region is 2-3 m thick**.

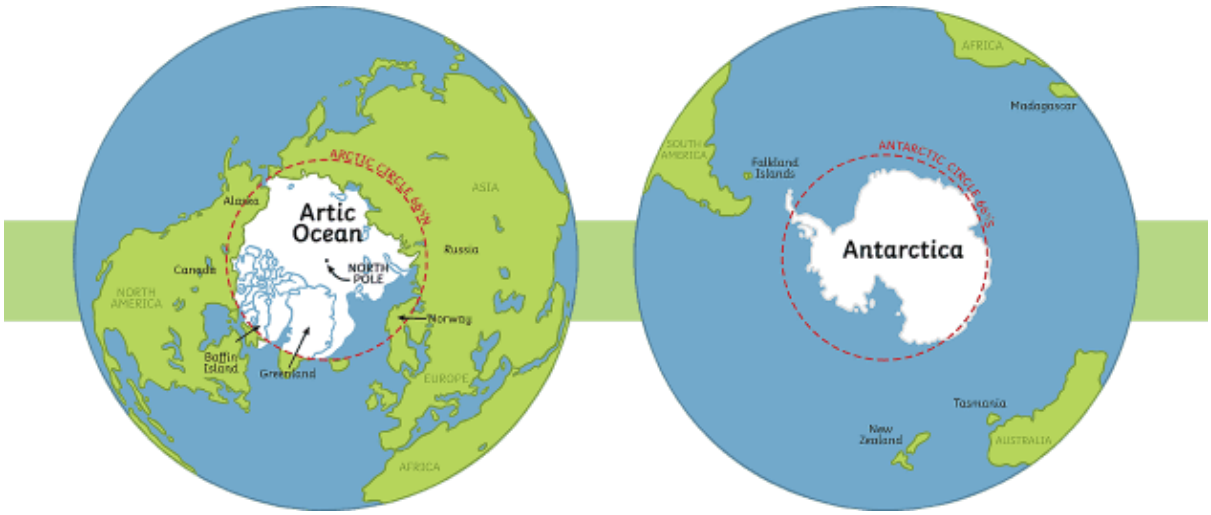


Fig 1.1: Polar region

1.2 Icebergs

A large floating mass of ice detached from a glacier or ice shelf and carried out to sea is called an iceberg. This process is called **calving**. Icebergs are made of frozen freshwater and thus float on sea water. **90%** of an iceberg is below the water surface. Most icebergs in the Northern Hemisphere break off from glaciers in Greenland. In the Southern Hemisphere, almost all icebergs calve from the continent of Antarctica. Once icebergs enter the water, they can fragment into smaller pieces through exposure to additional stresses and gradually melt. Although melting can occur at the surface when air temperatures are above freezing, the melting that occurs below the water surface is the primary control on how long it takes an iceberg to melt. There are different types of icebergs. Icebergs can be classified based on their size and shape.

category	height		length	
	metres	feet	metres	feet
growler	< 1	< 3	< 5	< 16
bergy bit	1 to 4	3 to 13	5 to 14	15 to 46
small berg	5 to 15	14 to 50	15 to 60	47 to 200
medium berg	16 to 45	51 to 150	61 to 122	201 to 400
large berg	46 to 75	151 to 240	123 to 204	401 to 670
very large berg	> 75	> 240	> 204	> 670
Source: International Ice Patrol				

Table 1.1: Iceberg size categories

Fig 1.2: Iceberg

1.3 Iceberg Drift

Since almost 90% of the iceberg is below the water surface, their movement is chiefly determined by the ocean currents. **Average drift speed** of an iceberg is around **0.7 km/h**, although speed greater than 3.6 km/h have been recorded. The dynamics of icebergs is likewise more complicated than might first seem the case. Following are the factors affecting iceberg drift:

- **Ocean currents** - It is basically the movement of seawater. As 90% of the iceberg is below water surface, it affects the drift of icebergs.

- **Coriolis force** - The Coriolis force causes objects to accelerate because of the rotation of the Earth. The Coriolis force is small at scales of tens to hundreds of kilometers, but it does affect processes that occur at the global scale Coriolis force causes objects to deflect to the right, and in the Southern Hemisphere, objects deflect to the left.
- **Pressure gradient force** - The pressure-gradient force is the force that results when there is a difference in pressure across a surface.
- **Force Imbalance** - it means force in 1 direction is more than force in opposite direction. This causes the object to change its speed or direction. There is imbalance of force in the ocean which affects iceberg drift.
- **Wind** - Wind is the primary force responsible for the sea ice motion but it does not affect the drift of iceberg to a great extent as very small portion of iceberg's total height is above water surface.

These forces vary with iceberg dimensions, resulting in the diverse drift of an ensemble of icebergs with a range of dimensions.

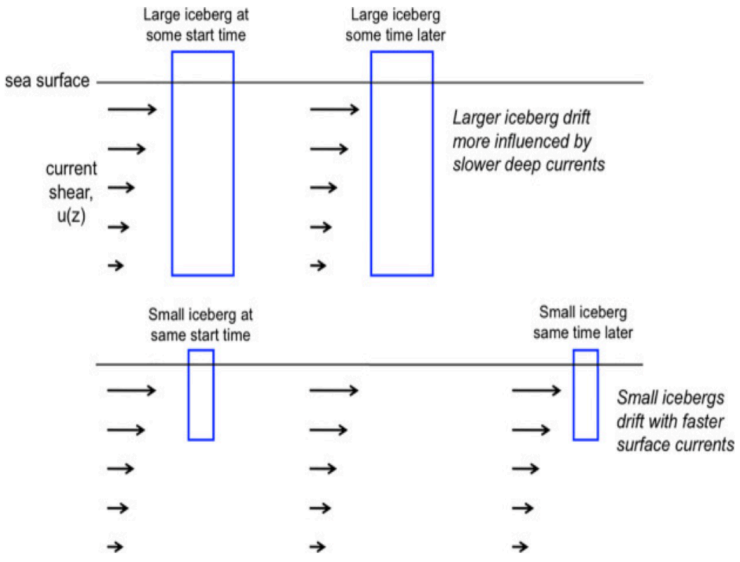


Fig: 1.3 why different iceberg sizes follow different pathways.

Chapter 2

About the system

2.1 Synthetic Aperture Radar (SAR)

SAR is a form of radar that is used to create images of landscapes in 2-D or 3-D. SAR remote sensing is an active microwave remote sensing technique which provides higher spatial resolution data. Here, a synthetic antenna will be formed using the actual small antenna and velocity of the satellite. Microwave energy is radiated from satellite, this energy is scattered back from the ground and received by the platform and then image is formed through signal processing.

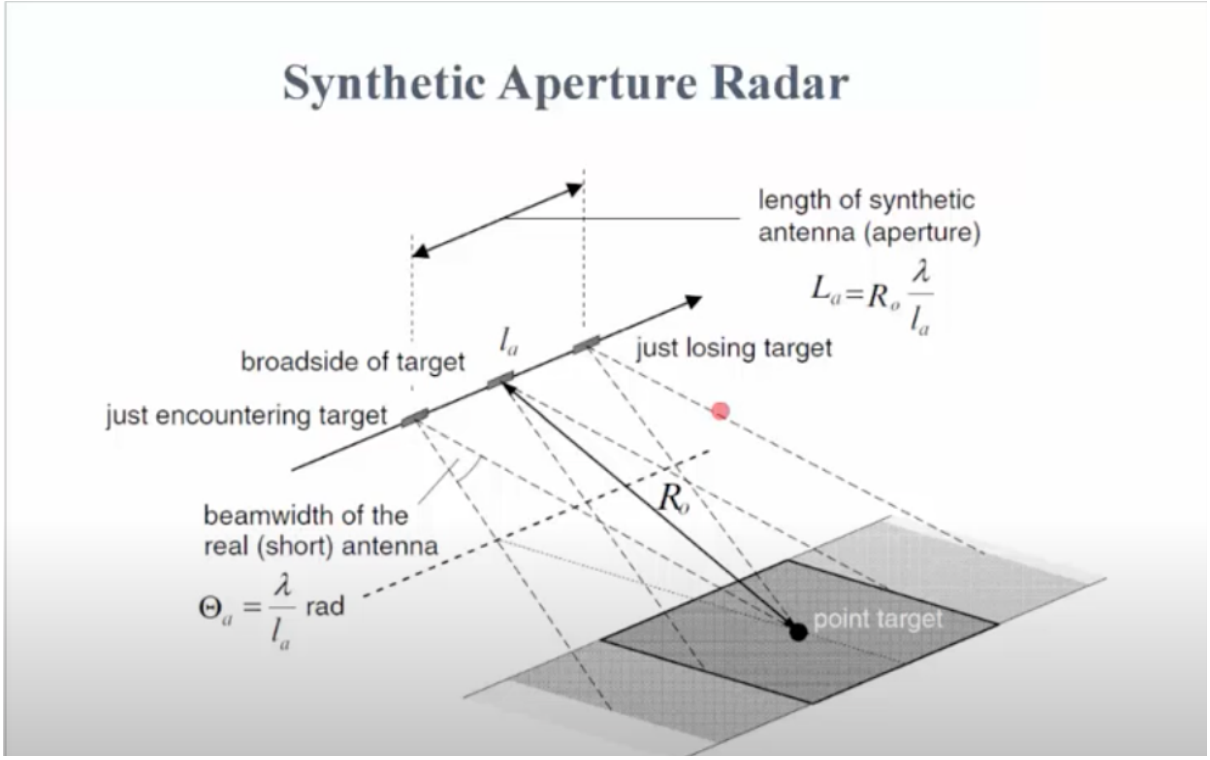


Fig 2.1: basic working of SAR

The spatial resolution of radar data is directly related to the ratio of the sensor wavelength to the length of the sensor's antenna. For a given wavelength, the longer the antenna, the higher the spatial resolution. From a satellite in space operating at a wavelength of about 5 cm (C-band radar), in order to get a spatial resolution of 10 m, you would need a radar antenna about 4,250 m long (That's over 47 football fields!).

Antenna of such size is not practical. Hence, scientists and engineers have come up with a clever workaround — the synthetic aperture In this concept, a sequence of acquisitions from a shorter antenna are combined to simulate a much larger antenna, thus providing higher resolution data.

SAR is used in cryospheric applications (sea ice drift, iceberg detection etc), flood control, soil moisture content etc.

2.2 SAR Data

SAR image data provide information different from that of optical sensors operating in the visible and infrared regions of the electromagnetic spectrum. **SAR data consist of high-resolution reflected returns of radar-frequency energy from terrain that has been illuminated by a directed beam of pulses generated by the sensor**. The radar returns from the terrain are mainly determined by the physical characteristics of the surface features (such as surface roughness, geometric structure, and orientation), the electrical characteristics (dielectric constant, moisture content, and conductivity), and the radar frequency of the sensor. By supplying its own source of illumination, the SAR sensor can acquire data day or night without regard to cloud cover.

For detection of iceberg we use Sentinel-1 SAR data. Sentinel-1 is collection 2 satellites (1A & 1B), under European Space Agency's(ESA) sentinel mission. They carry C band (7.5 to 3.8cm 5.405GHz) SAR instrument. Sentinel-1 is freely available.

Links: https://scihub.copernicus.eu/dhus/#/home

https://search.asf.alaska.edu/#/

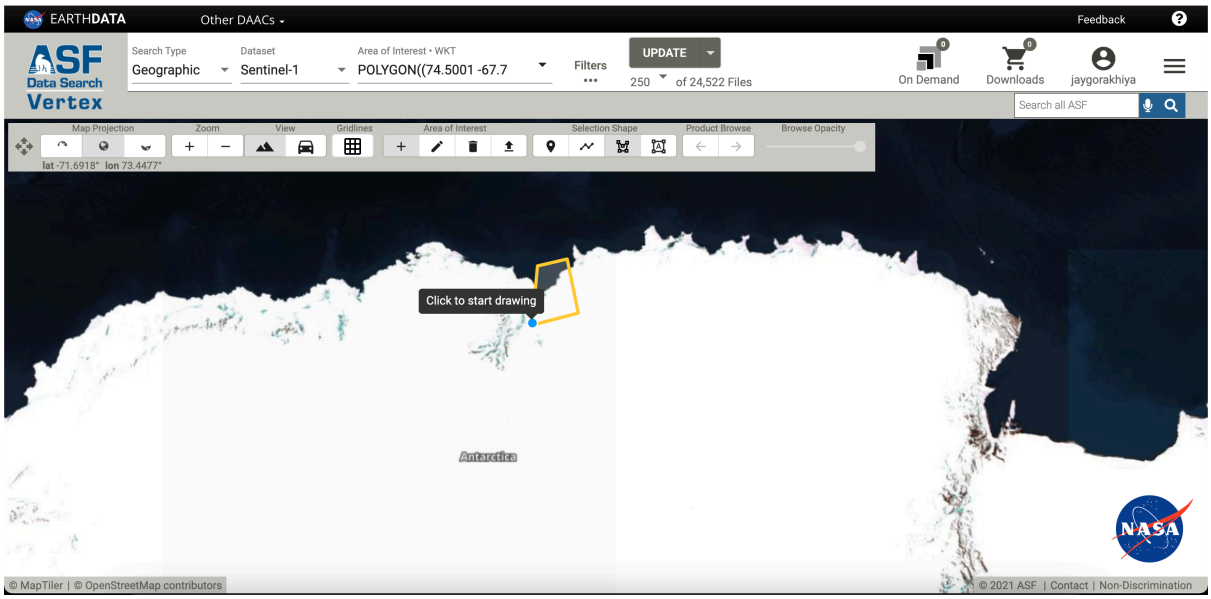


Fig 2.2: Alaska Search Facility data search dashboard

First select the area of which you want images. Yellow boundary in the above image indicates the desired area. You can also filter out the results based on date, beam mode etc. Now after clicking on search you can get all images that matches the search criteria. Select the image that you want to download, then select type of image that you need(Raw, SLC, GRD etc.) and click on the download icon near it. A zip file will be downloaded which can be opened in software like SNAP, GMTSAR, ISCE etc.

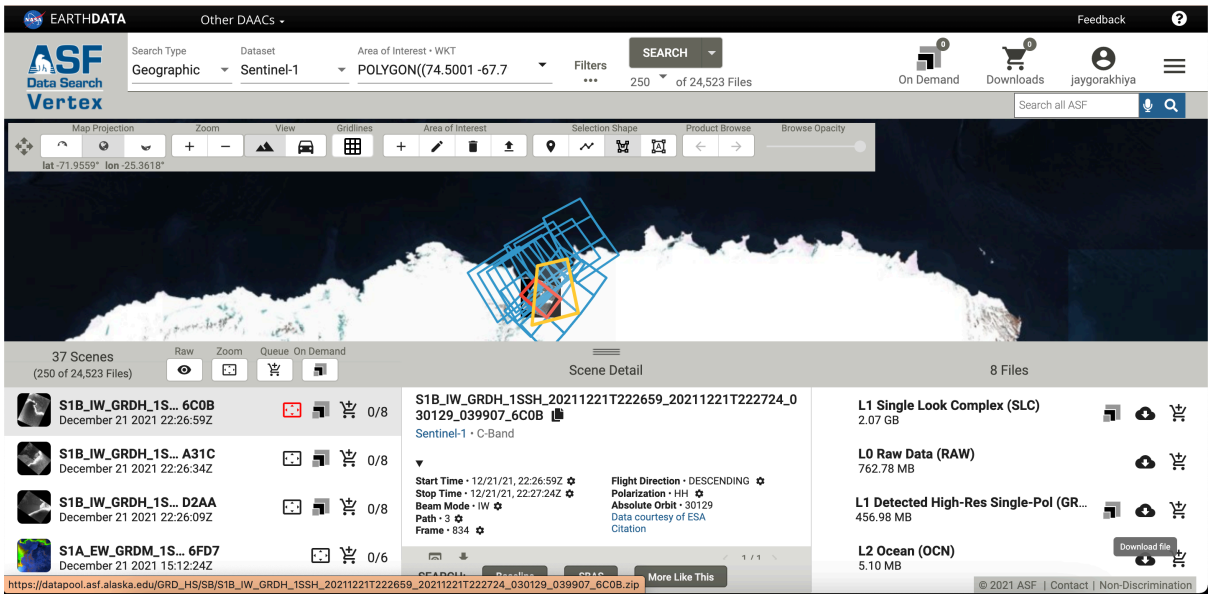
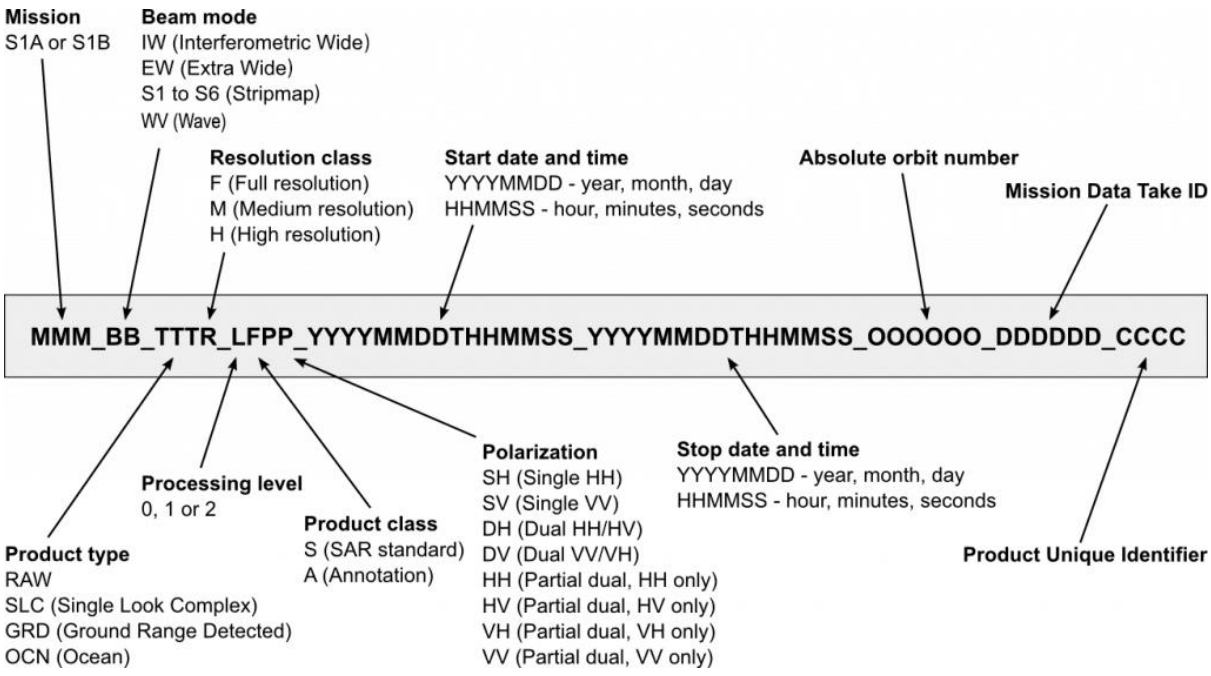


Fig 2.3: list of all images that matches the search criteria

The box indicates detail of all area that were captured by sentinel-1 satellite. Name of the file has exact meaning that is explained below.

First, you have to make one account in ASF site, after that you can download available resources.

2.3 Naming convention of SAR dataset



2.4 Beam Mode

There are four different beam modes.

1. Interferometry wide swath (IW)
2. Extra wide swath (EW)
3. Strip Map (SM)
4. Wave (WV)

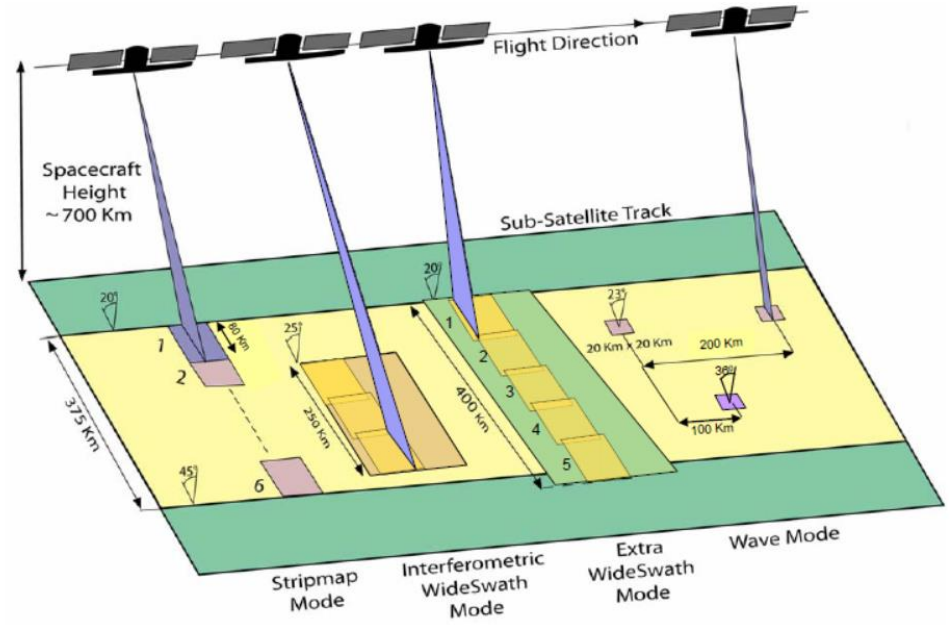


Fig 2.4: Difference between Beam modes

2.5 Polarization

Signals with components in two orthogonal or basis polarizations are needed to create a wave with an arbitrary polarization. The two most common basis polarizations are horizontal linear or H, and vertical linear or V. The radar antenna is often designed to receive the different polarization components of the EM wave simultaneously. For example, the H and V parts of an antenna can receive the two orthogonal components of the incoming wave, and the system electronics keep these two signals separate. Denoting the transmit and receive polarizations by a pair of symbols, a radar system using H and V linear polarizations can thus have the following channels:

I. HH - for horizontal transmit and horizontal receive (HH)
II. VV - for vertical transmit and vertical receive (VV)
III. HV - for horizontal transmit and vertical receive (HV)
IV. VH - for vertical transmit and horizontal receive (VH).

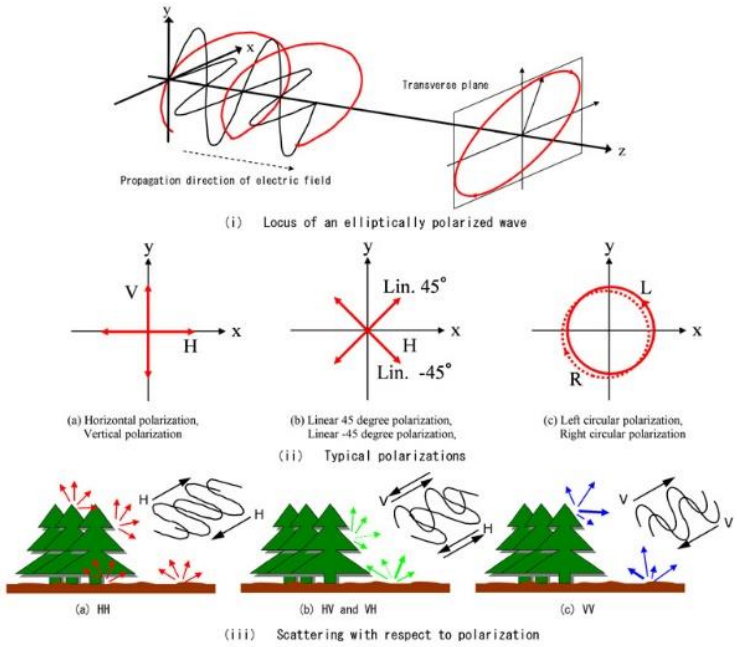


Fig 2.5: Difference between HH, HV, VH, & VV polarization

2.6 Product Type

There three types of products.

1. Level–0
2. Level–1
3. Level–2

Level-1 products can be one of two product types - either Single Look Complex (SLC) or Ground Range Detected (GRD). Level-2 Ocean (OCN) products can have different components available depending on the acquisition mode. Products are designated based on their acquisition mode, product type and in the case of Level-1 GRD also its resolution.

All products are processed directly from the Level-0 product. Each mode can potentially generate Level-1 SLC, Level-1 GRD and Level-2 Ocean products. For WV mode, the Level-0 products are not distributed.

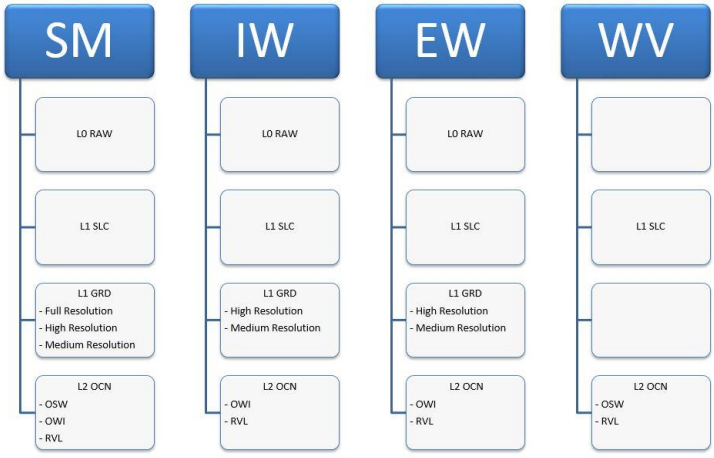


Fig 2.6 product usage in beam modes

2.7 Resolution (level 1 ground range detected)

Level-1 GRD products are available in one of three spatial resolutions:

1. Full Resolution (FR) for SM mode

Mode	Resolution rg x az	Pixel spacing rg x az	Number of looks	ENL
SM	9x9 m	3.5x3.5 m	2x2	3.7

Table 2.1: For FR beam mode parameters

2. High resolution (HR) for SM, IW, EW modes

Mode	Resolution rg x az	Pixel spacing rg x az	Number of looks	ENL
SM	23x23 m	10x10 m	6x6	29.7
IW	20x22 m	10x10 m	5x1	4.4
EW	50x50 m	25x25 m	3x1	2.7

Table 2.2: For HR beam mode parameters

3. Medium Resolution (MR) for SM, IW, EW, WV modes

Mode	Resolution rg x az	Pixel spacing rg x az	Number of looks	ENL
SM	84x84 m	40x40 m	22x22	398.4
IW	88x87 m	40x40 m	22x5	81.8
EW	93x87 m	40x40 m	6x2	10.7
WV	52x51 m	25x25 m	13x13	123.7

Table 2.3: For MR beam mode parameters

2.8 SNAP (Sentinel Application Platform)

SNAP is a software provided by ESA to process SAR images. It is GUI based software. In the image view Panel, you can open your desire band of image. You can also work with multiple bands simultaneously.

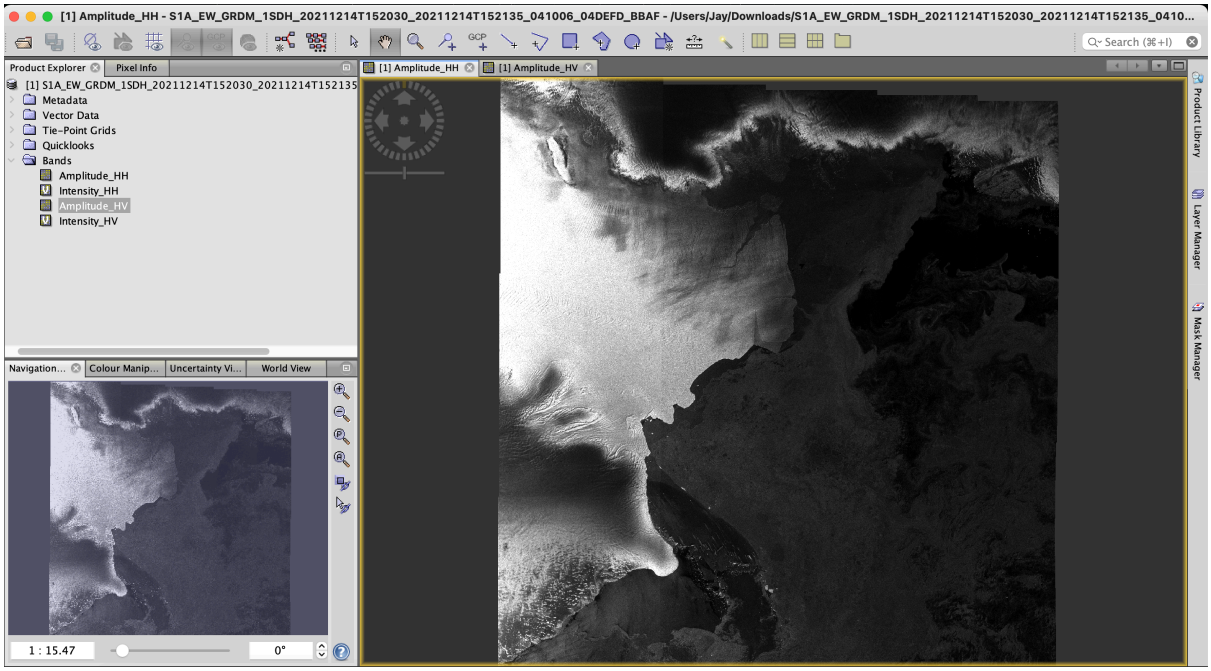


Fig 2.7: SAR image opened in SNAP

Before using the image for iceberg detection, it must be preprocessed which can be done using SNAP.

2.8.1 Data preprocessing in SNAP

Following are the steps of SAR data preprocessing:

1. Border noise removal (if present)
2. Apply orbit file
3. Thermal noise removal
4. Calibration
5. Speckle filtering (remove salt & pepper noise)
6. Multi looking
7. Ellipsoid correction
8. Linear to db

Output of the following steps is **sigma0 image** of the selected band from original image.

Sigma0 - It is Scattering coefficient, or the conventional measure of the strength of radar signals reflected by a distributed scattered, usually **expressed in db**. It is a normalised dimensionless number, comparing the strength observed to that

expected from an area of one square meter. Sigma nought is defined with respect to the nominally horizontal plane, and in general has a significant variation with incidence angle, wavelength, and polarization, as well as with properties of the scattering surface itself

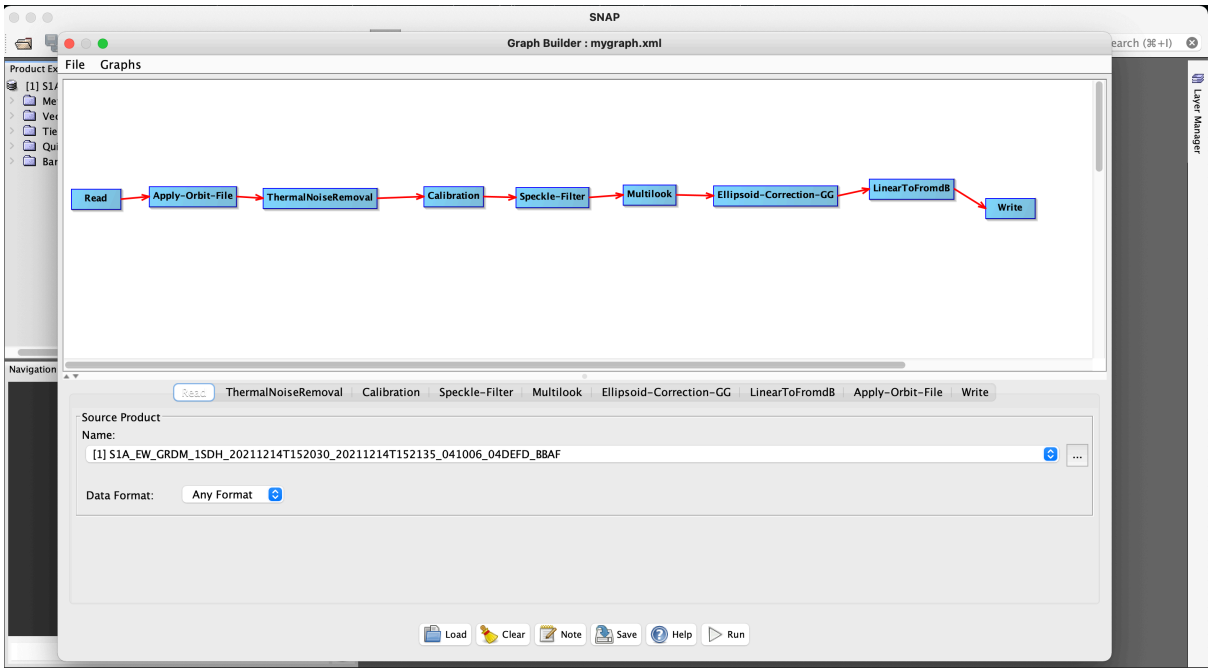


Fig 2.8: graph showing preprocessing steps in order

Fig 2.9: amplitude_HV band of S1A_EW_GRDM_1SDH_20211214T152030_20211214T152135_041006_04DEFD_BBAF

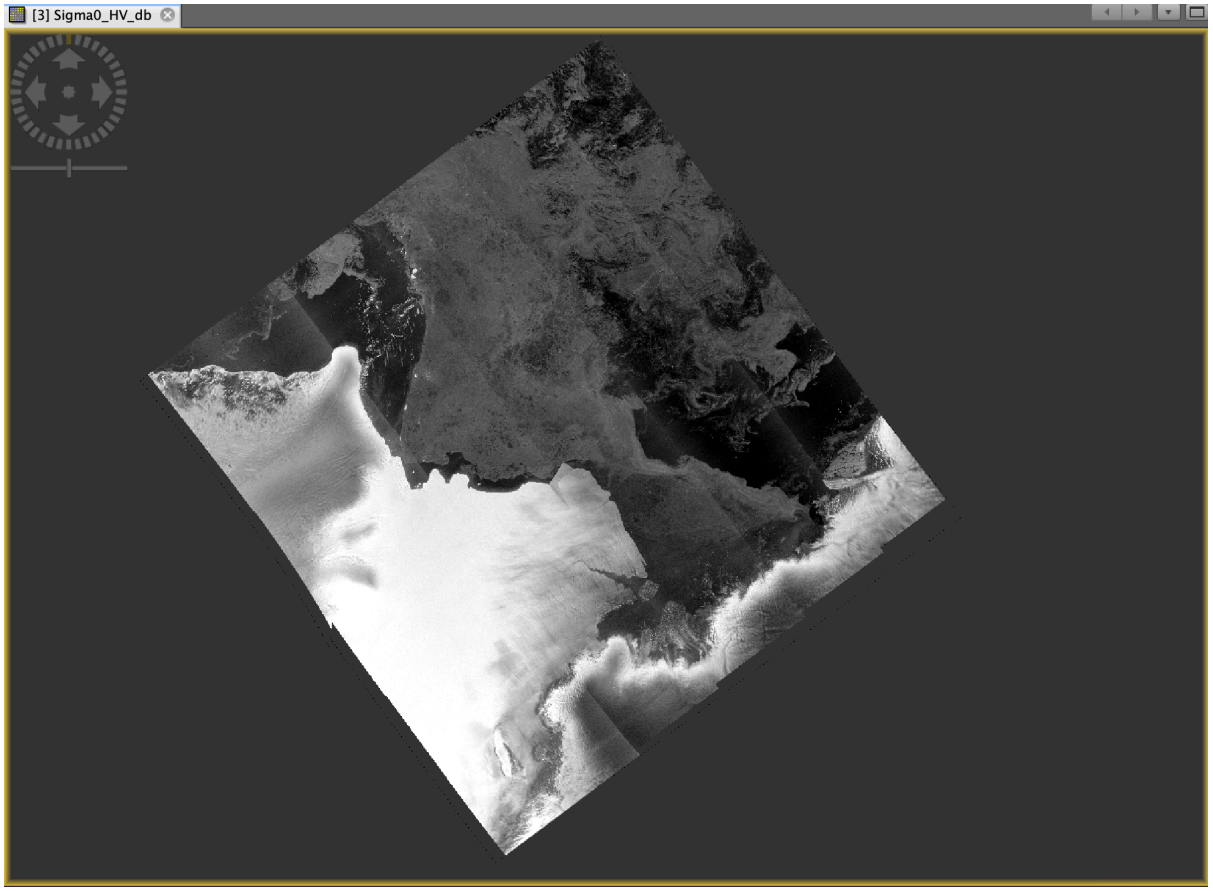


Fig 2.10: Sigma0_HV_db (processed image) of S1A_EW_GRDM_1SDH_20211214T152030_20211214T152135_041006_04DEFD_BBAF

Chapter 3

Tools & Libraries

3.1 Tools

below is the list of tools used in this project of iceberg detection and tracking:

1. SNAP (Sentinel Application Platform)
2. Google Colab
3. Google Earth Engine

SNAP is a software developed by ESA for processing sentinel images. Here, in our project we have used sentinel-1 images which uses SAR. SAR images can be preprocessed using SNAP.

However, we have used SNAP for only study purposes because for data processing in SNAP you have to download SAR image manually and then you have to preprocess it using SNAP GUI. This not only uses local machine's memory but also processing speed is bottlenecked by the processor of the machine.

All the SAR images i.e sentinel-1 SAR dataset is available in Earth Engine, and these images available in earth engine are already preprocessed. So, we can use them directly in our project. This saves lot of memory and processing time. We have used Google colab as our code editor for python language.

3.2 Libraries

Following are the **python libraries** used in this project:

1. Earth engine library (ee)
2. Folium library
3. Pandas
4. Numpy
5. Matplotlib
6. Datetime

We have used earth engine python API in our implementation of the iceberg detection and tracking project. It has many client side libraries which can be used in retrieving, manipulating and processing the SAR images. We can also get (preprocessed) SAR images using earth engine library. Below is a list of some earth engine library classes and methods used commonly:

ee.Image - An object to represent an Earth Engine image. Its constructor accept arguments as strings, list, array or an image and returns an. Some of its methods are:

- **Image.bandNames()** : Returns a list containing the names of the bands of an image.
- **Image.clip(geometry)** : Clips an image to a Geometry or Feature.
- **Image.cumulativeCost(source, maxDistance, *geodeticDistance*)** : Computes a cumulative cost map based on an image containing costs to traverse each pixel and an image containing source locations.
- **Image.paint(featureCollection, color, width)** : Paints the geometries of a collection onto an image.
- **Image.reduceToVectors()** : Convert an image to a feature collection by reducing homogeneous regions.

ee.ImageCollection - An object to represent collection of more than 1 Image. ImageCollections can be constructed using a string, single image or list of images. Some of its methods are:

- **ImageCollection.filter()** : used to apply filter to the collection.
- **ImageCollection.filterBounds(geometry)** : used to filter a collection by intersection with geometry.

ee.Filter - Constructs a new filter. The constructor accepts another filter, a list of filters, computed object returning filter as its argument**.**

ee.Geometry - it creates a geometry by passing a geoJSON object as argument to its constructor. It can create geometry is like rectangle, polygon, multi polygon, point etc. some of its methods are:

- **Geometry.area(maxError, proj)** : Returns the area of the geometry.
- **Geometry.centroid(maxError, proj)** : Returns a point at the center of the highest-dimension components of the geometry.
- **Geometry.distance(right, *maxError*, *proj*)** : Returns the minimum distance between two geometries.

ee.Feature - Features can be constructed from one of the arguments as ee.Geometry, geoJSON object or computed object plus an optional dictionary of properties.

ee.FeatureCollection - An object representing collection of more than 1 feature. It can be constructed from arguments as string, single geometry, a feature, a list of features or a geoJSON object.

Folium makes it easy to visualize data that's been manipulated in Python on an interactive leaflet map. It enables both the binding of data to a map for choropleth visualizations as well as passing rich vector/raster/HTML visualizations as markers on the map. Thus, we can add raster layers and vector layers to map in order to visualize the iceberg images and its trajectory.

numpy is used for scientific computing like creating histograms and calculating difference between 2 histograms. **Matplotlib** is used for visualisation of histograms and **pandas** is used to read geoJSON files into a dataframe and extract iceberg information from it. **Datetime** library is used to work with date & time of images and iceberg locations.

Chapter 4

Iceberg detection

4.1 SNIC (Simple Non Iterative Clustering)

To detect the target iceberg from the Sentinel-1 images, we employ a superpixel image segmentation method named simple non-iterative clustering (SNIC)[2]. This approach uses similar concept to k-means clustering(simple linear iterative clustering), but in contrast, it implements connectivity from the start.

4.1.1 Initialization of superpixel

In the SNIC image segmentation, all image pixels should be grouped into small clusters of connected pixels, which are named **superpixels**. The **centroids of superpixels**, referred to as **seeds**, are initialized with a given number of pixels chosen a priori on a regular grid. The size of each segment is determined by the seed value. A smaller seed value can distinguish small icebergs but takes a longer computation time while a larger seed value is used for detection of large icebergs and can reduce computation time.

4.1.2 Distance measure

After initialisation of the superpixel centroid or seed value, affinity of a pixel to a centroid is measured using a distance in five dimensional space of colour and spatial coordinates. The affinity of the jth pixel to the kth superpixel centroid is calculated by using the distance between them ($d_{j,k}$):

$$d_{j,k} = \sqrt{\frac{\left\| X_j - X_k \right\|^2}{s} + \frac{\left\| C_j - C_k \right\|^2}{m}},$$

Here, X_j is a geocoordinate (latitude and longitude) of the jth pixel, X_k is a geocoordinate of the kth superpixel centroid, C_j is the HH band backscatter of the jth pixel, C_k is the HH band backscatter of the kth superpixel centroid. S and M are the normalising factors for spatial and colour (backscatter) distances, respectively. If an image has N pixels and K superpixel centroids, S in the equation should be set $\sqrt{N/K}$

The value of m, also called the *compactness factor*, is user-provided. A higher value results in more compact superpixels at the cost of poorer boundary adherence, and vice versa.

4.1.3 Evolution of centroids

SNIC uses **priority queue** to choose the next pixel to add to a cluster. The priority queue is populated with candidate pixels that are 4 or 8-connected to a currently growing superpixel cluster. Popping the queue provides the pixel candidate

that has the smallest distance d from a centroid. Here, we update the centroid in single iteration.

4.2 Algorithm

Input: Input image I, K initial centroids $C[k] = \{\mathbf{x}_k, \mathbf{c}_k\}$ sampled on a regular grid, color normalization factor m
Output: Assigned label map L

Initialize $L[:] \leftarrow 0$
for $k \in [1, 2, ...K]$ **do**
  Element $e \leftarrow \{\mathbf{x}_k, \mathbf{c}_k, k, 0\}$
  Push e on priority queue Q
end for
while Q is not empty **do**
  Pop Q to get e_i
  if $L[\mathbf{x}_i]$ is 0 **then**
    $L[\mathbf{x}_i] = k_i$
    Update centroid $C[k_i]$ online with $\mathbf{x}_i$ and $\mathbf{c}_i$
    for Each connected neighbor $\mathbf{x}_j$ of $\mathbf{x}_i$ **do**
      Create element $e_j = \{\mathbf{x}_j, \mathbf{c}_j, k_i, d_{j,k_i}\}$
      if $L[\mathbf{x}_j]$ is 0 **then**
        Push e_j on Q
      end if
    end for
  end if
end while
return L

Fig 4.1: SNIC segmentation algorithm

The initial K seeds C[k] = {x_k,c_k} are obtained as for SLIC on a regular grid over the image. Using these seed pixels, K elements e_i = {x_i, c_i, k, $d_{i,k}$} are created, wherein each label k is set to one unique superpixel label from 1 to K, and each distance value $d_{i,k}$, representing the distance of the pixel from the kth centroid, is set to zero. A priority queue Q is initialized with these K elements. When popped, Q always returns the element e_i whose distance value $d_{i,k}$ to the kth centroid is the smallest.

While Q is not empty, the top-most element is popped. If the pixel position on the label map L pointed to by the element is unlabelled, it is given the label of the centroid. The centroid value, which is the average of all the pixels in the superpixel, is updated with this pixel. In addition, for each of its 4 or 8 neighbours that have not been labeled yet, a new element is created, assigning to it the distance from the connected centroid and the label of the centroid. These new elements are pushed on the queue.

As the algorithm executes, the priority queue is emptied to assign labels at one end and populated with new candidates at the other. When there are no remaining unlabelled pixels to add new elements to the queue and the queue has been emptied, the algorithm terminates.

SNIC visits all pixels only once, except for those at the borders of the superpixels. Thus, **the number of visits is N plus a value that is dependent on the number of desired superpixels K**.The priority queue, when implemented using a heap data structure, is known to have logarithmic complexity for pushing and popping elements.

After completing the SNIC image segmentation, we detect the segments of icebergs by applying the mean brightness filter to all segments. Icebergs commonly show higher backscatter intensity(-12 to -5 sigma0_HH intensity_db) compared to the surrounding sea water or sea ice. Thus, we identify iceberg segments as the segments satisfying the following equation:

$$\gamma/\alpha > -0.2$$

Here, γ is the. Backscatter of SAR image and α is incidence angle of SAR image.

4.3 Implementation

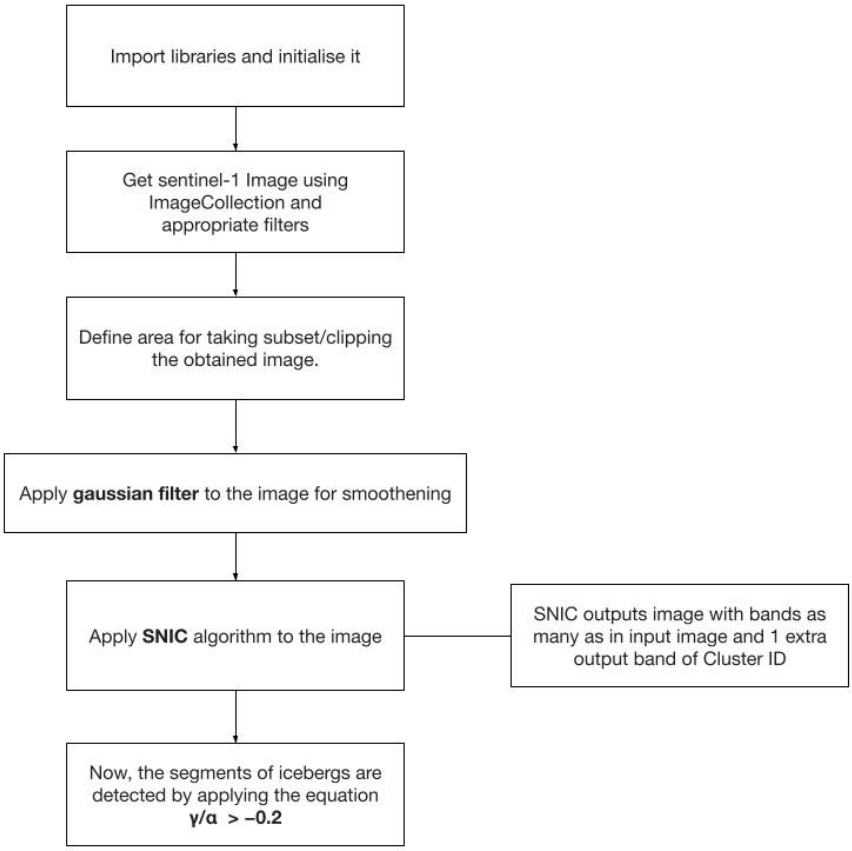


Fig 4.2: Flowchart of implementation steps

Implementation is done using Earth engine library (ee) and Folium library in python programming language.

We first take input image using the **ee.ImageCollection** class. Sentinel-1 preprocessed images are already available in Earth engine library so we do not need to preprocess the images separately. We can get the desired image by applying date filters, area filters etc. We can also take a subset of the image/clip the image by keeping it to a specific area. Now the obtained image can be viewed in an **interactive map** using the **folium** library.

In implementation, before applying SNIC to the image **we first apply guassian filter to SAR image**. The image segmentation process can misinterpret shaded areas associated with the iceberg surface(sea ice formed near boundary) topography as artificial iceberg boundaries and split the iceberg into several pieces. To reduce this effect, we first smooth images by applying a Gaussian filter kernel.

After applying gaussian filter to the image, SNIC is applied. Implementation of the SNIC algorithm is available in earth engine library. It takes image, compactness factor, connectivity, neighbourhoodSize and seed value as function arguments and Outputs a band of cluster IDs and the per-cluster averages for each of the input bands.now, segments of the iceberg are detected by applying the equation $\gamma/\alpha > -0.2$.

```
#guassian kernel for softening the image
kernel = ee.Kernel.gaussian(3)
img1 = clipped_Image1.convolve(kernel)
img = img1.select(['HH', 'angle'], ['HH', 'angle'])

#SNIC image segmentation
snic = ee.Algorithms.Image.Segmentation.SNIC(img, 20, 5, 8, 256, seeds)\
        .select(["HH_mean", "angle_mean", "clusters"], ["HH", "angle", "clusters"])
ratio = snic.select("HH").divide(snic.select("angle"))
icebergs = ratio.gt(-0.2).selfMask()
show_iceberg = icebergs.clip(D28)
```

Fig 4.3: code snippet of SNIC image segmentation

As shown here, gaussian filtered image is passed as argument to the method **ee.Algorithms.Image.Segmentation.SNIC()**, **connectivity** is set to **8**, **compactness** factor is set to **5** and **seed value** is initialised to **100** (relatively large value) as the iceberg we are detecting, Iceberg D28, is large in size has an approximate area of 1600 km^2 . Smaller seed value is used when iceberg is smaller in size.

Below are the input and output images of the iceberg detection.

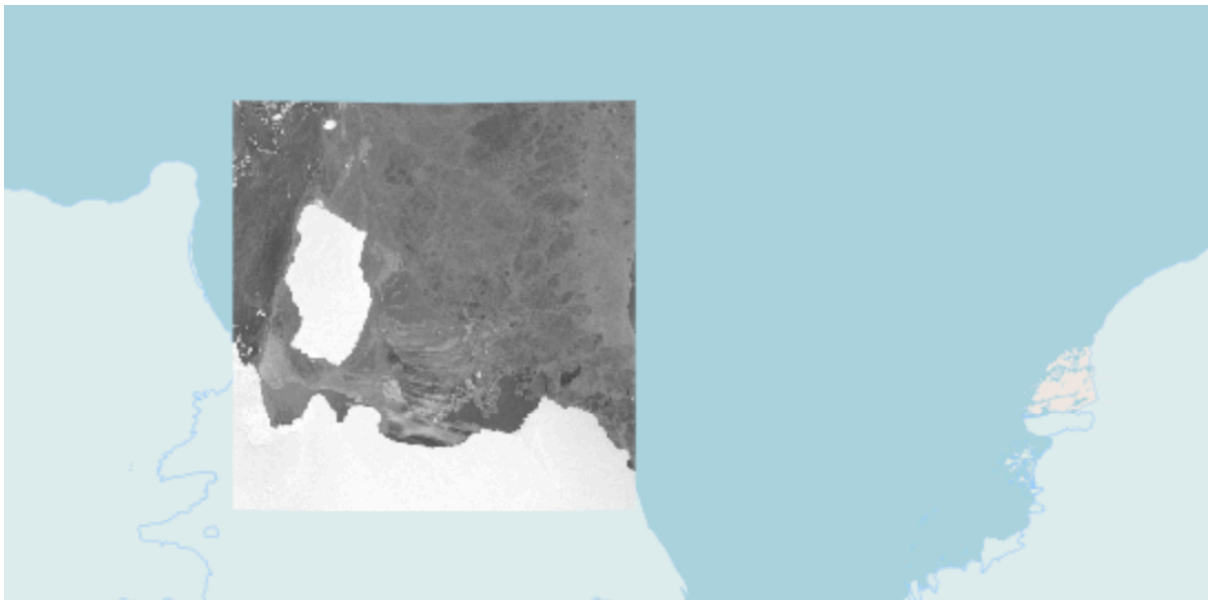


Fig 4.4: Input image

Fig 4.5: output image (with cluster id band)

In the above image, we can see the clusters formed after SNIC image segmentation. Both these images are shown on interactive map using folium library.

We can confirm/cross check the detection of iceberg D28 by seeing the area of the detected iceberg. Here, calculated area of the detected iceberg is 1602.5 km^2 .


```
{'area': 5777537370.092439, 'centroid': {'type': 'Point', 'coordinates': [71.86683806084024, -68.83814865759291]}, 'count': 3635525, 'label': 1}
{'area': 1249487.4228872724, 'centroid': {'type': 'Point', 'coordinates': [70.54217000383113, -68.70115980913414]}, 'count': 786, 'label': 1}
{'area': 14826884.595294125, 'centroid': {'type': 'Point', 'coordinates': [70.96913836003495, -67.59207551312217]}, 'count': 9326, 'label': 1}
{'area': 1602506158.8435507, 'centroid': {'type': 'Point', 'coordinates': [70.9581971365636, -68.18864320507402]}, 'count': 1008054, 'label': 1}
```


Fig 4.6: area of detected iceberg

Chapter 5

Iceberg tracking

5.1 Tracking Algorithm

In order to automatically detect and track only the target iceberg among multiple icebergs, we adopt a feature extraction method based on centroid distance function. We create a centroid distance histogram of the iceberg, reference and detected, and then compare both histograms to know the similarity between them[1].

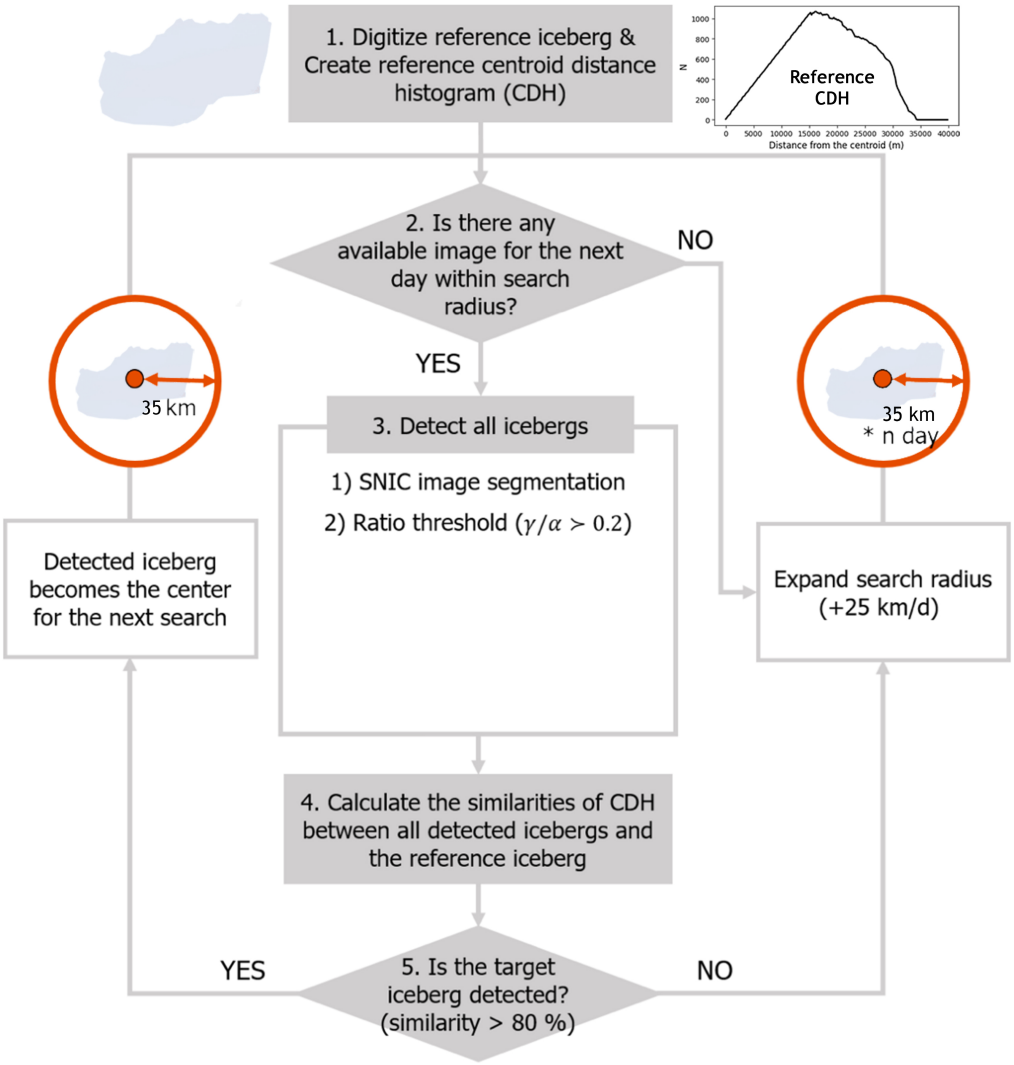


Fig 5.1: Tracking algorithm flowchart

Step1 - First, we define the target iceberg that will be tracked, which in our case is iceberg D28. We manually digitize the polygon of the reference iceberg from a Sentinel-1 image and calculate the distances from the iceberg centroid for all iceberg pixels. We make a histogram of these centroid distances for all pixels in the

iceberg; this **centroid distance histogram (CDH)** represents the area and shape of the iceberg.

Step2 - Specify the initial search radius(25 or 35km from reference iceberg) and search for available sentinel-1 images in the radius for the next day. If the image for the next day is not available, the search radius is extended by 25 km each day until a Sentinel-1 image is found.

Step3 - Once a Sentinel-1 image is found, all candidate icebergs are detected in the image by using the process described in chapter 4 i.e SNIC image segmentation.

Step4 - Once all icebergs are detected from an image, we calculate and create Centroid Distance Histograms (CDH) of all detected icebergs. Then we compare the CDHs of detected icebergs with the reference iceberg to calculate similarities between them. The similarity of CDH between iceberg A and reference iceberg (r (A, ref)) is calculated by equation:

$$r(\mathrm{A}, \mathrm{ref}) = \left(1 - \frac{\sum_{i=1}^{m} |N_{\mathrm{A}}(i) - N_{\mathrm{ref}}(i)|}{\sum_{i=1}^{m} N_{\mathrm{ref}}(i)} \right) \times 100\,(\%),$$

where m is the bin size of the histogram, and $N_A(i)$ and $N_{ref}(i)$ are the number of pixels in the ith bin for iceberg A and reference iceberg, respectively. We select the iceberg showing similarity greater than 80 % with the reference iceberg. If multiple icebergs show similarity above 80 % with the reference iceberg, we select the iceberg with the highest similarity.

Step5 - Once we select the iceberg with most similarity, the centroid of this new detected iceberg becomes the centre for the search radius for the next day and goto step2. However if no iceberg is detected then we goto step2, increase the search radius and repeat the same steps until a target iceberg is detected.

In above steps, if no iceberg is found in sentinel-1 images of 5 to 6 consecutive days and the search radius becomes too large to be processed then in that case manual intervention is needed to find the next image. Also when a significant mass of the iceberg breaks from the iceberg, the shape of the iceberg changes considerably. In these cases, the initial CDH would no longer be representative of the iceberg and a new reference iceberg would have to be manually set.

5.2 Implementation

Implementation is done using Earth engine library (ee) and Folium library in python programming language.

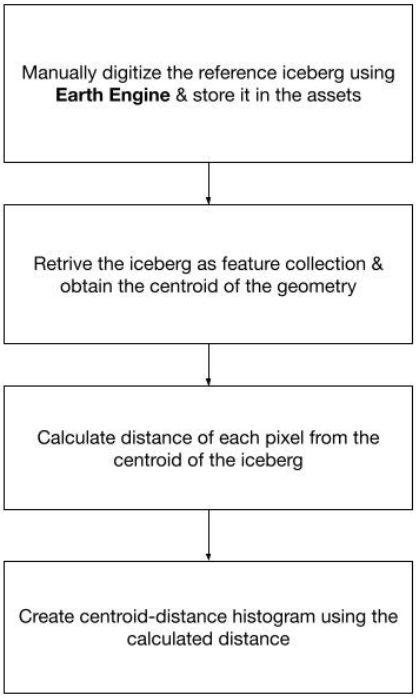


Fig 5.2: Flowchart of reference CDH implementation.

The whole tracking process can be divided into two major steps. The first step is creating reference icebergs histogram. For that, we first digitize the reference iceberg using Earth engine code editor and store the digitized iceberg as an asset in Earth engine.

Then we retrieve the asset as a feature collection using Earth engine library in python language in Google Colab and get its centroid value. After that we get a Sentinel-1 image of a particular date(23/10/2019) and clip it to the reference iceberg geometry using **ee.ImageCollection**. After that we define the centroid as a source for calculating distance of each pixel from centroid by converting it to ee.Image. We use **Image.cumulativeCost()** method to calculate distance of each pixel from its centroid and this method is applied on the image obtained and Image of source(centroid of the iceberg) from which distance is to be calculated is passed as an argument along with **max distance** (for computation in meters) which is set as **35000**.

```
# Compute the cumulative cost to traverse the land cover.
# | (distance of each pixel from centroid)
cumulativeCost = cover.select('HH').cumulativeCost(
    source = sources,
    maxDistance = 35000,
    geodeticDistance = True
).clip(reference)
```

Fig 5.3: code snippet showing cumulativeCost()

Now, the distances calculated are converted to a list using **Reducer.toList()** which reduces an image to a list and using this list, the centroid distance histogram **(CDH**) of the reference iceberg is created using numpy, where **range** of distance is set from **0 to 40000** and **bin size** is **200**. Now this histogram can be visualised using matplotlib.

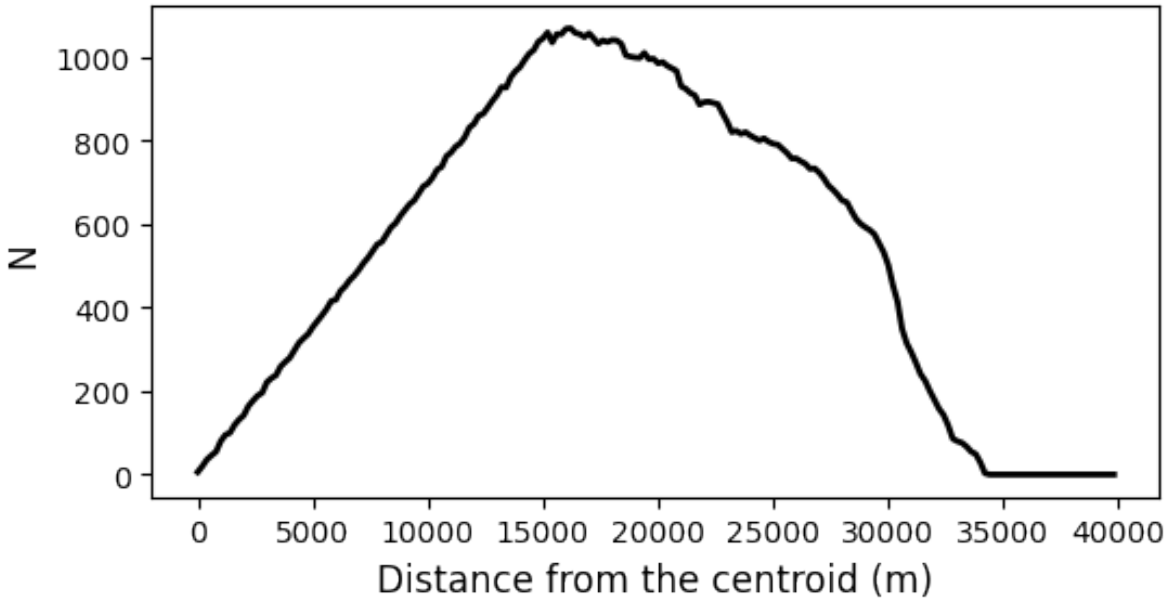


Fig 5.4: CDH of reference iceberg (23/10/2019)

We can also get the area of reference iceberg from the feature collection using **geometry().area()** method. The area of the reference iceberg dated 23/10/2019 calculated using above method is 1602.5 km^2.

After creating the CDH of the reference iceberg, we now start tracking the iceberg. We first provide the start date and end date as 1st and 2nd date of a month and initialize the search radius by providing the center of the search area. Then we initialize the map and set the buffer area (buffer distance set as 35000m) for searching images.

We search for sentinel-1 images by filtering it using the start & end date and the buffer area. If there is no image available for current date then we increase the search radius by 25000 and search for available sentinel-1 image in next day. If an image is available then we apply SNIC image segmentation on the obtained image to detect icebergs from the image using same process described in chapter 4.

After applying SNIC algorithm on the image, all detected iceberg's rasters are converted to vectors using **Image.reduceToVector()** and their area and centroid information is calculated using **geometry().area()** and **geometry.centroid()** methods. Now, from all the detected icebergs only those whose area is between **1.1**

to 0.9 times the area of reference iceberg are filtered out and rest of the icebergs (if there) are not considered.

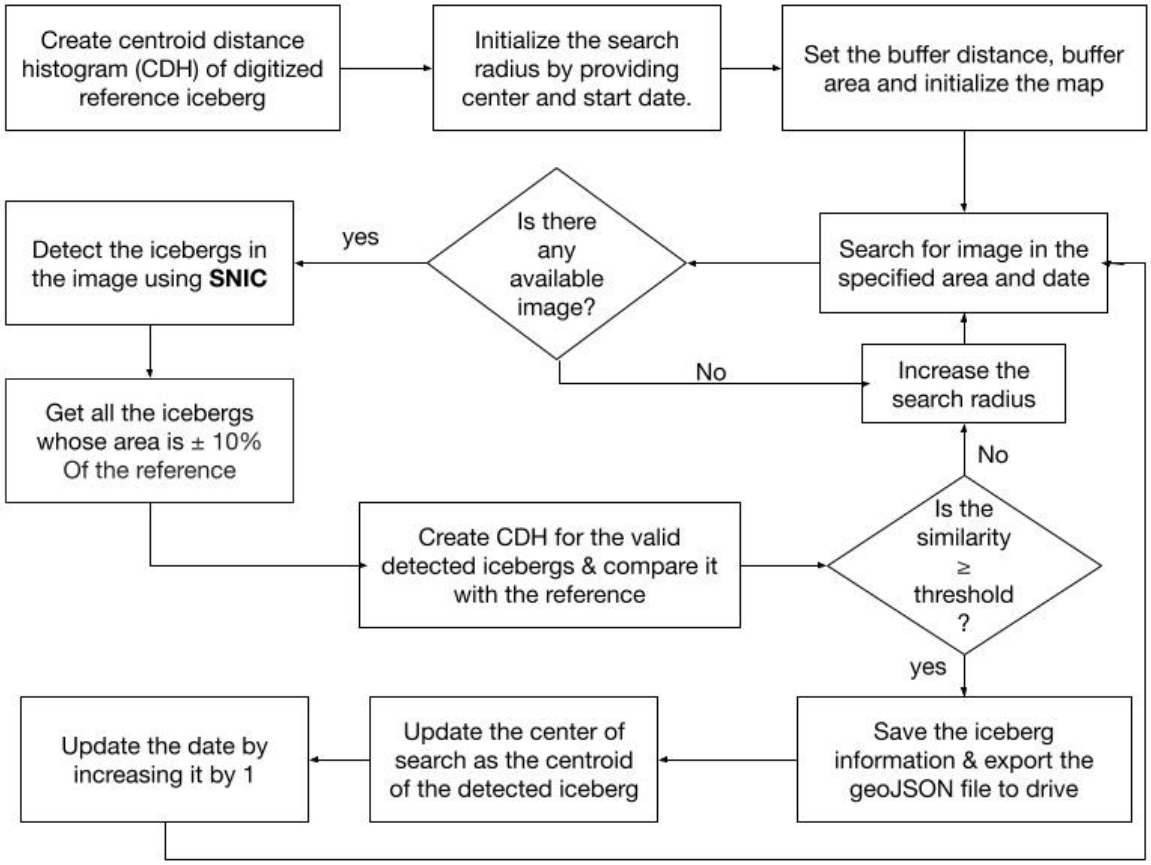


Fig 5.5: flowchart of tracking implementation

Now, if there are valid icebergs filtered in above step then their centroid distance histogram (CDH) is calculated using the same process described earlier in this section. Now, the CDH of detected iceberg is compared with CDH of the reference iceberg and similarity between both is calculated using numpy.

```
2019-11-02 ==========================================
Feature 0 >>> area: 1591.53 km^2, centroid: [70.366, -68.575], similarity: 0.466
Feature 1 >>> area: 1606.8 km^2, centroid: [70.857, -68.28], similarity: 0.979
... exporting task for 2019-11-02
2019-11-04 ==========================================
Feature 0 >>> area: 1610.89 km^2, centroid: [70.867, -68.27], similarity: 0.979
... exporting task for 2019-11-04
```

Fig 5.6: Output showing information of detected icebergs

The area, centroid and similarity of the detected iceberg are printed in output along with the date on which the iceberg was detected

if the **similarity** between the CDH of detected iceberg and reference iceberg is **greater than or equal to 70%**, then their area and centroid information is saved and a geoJSON file containing details of the iceberg detected is exported to Drive.

After that, center of search for next day is updated as centroid of the detected iceberg. If there is no iceberg whose similarity is greater than or equal to 70% then again search radius is increased by 25000 and search for available sentinel-1 image in next day.

As it can be seen in fig 5.6, there are 2 icebergs detected on 02/11/2019, one with similarity 0.466 and another with similarity 0.979. So, the one with **similarity 0.466 is not considered** further for CDH. Below is the image of CDHs of all valid detected icebergs in the month of November 2019.

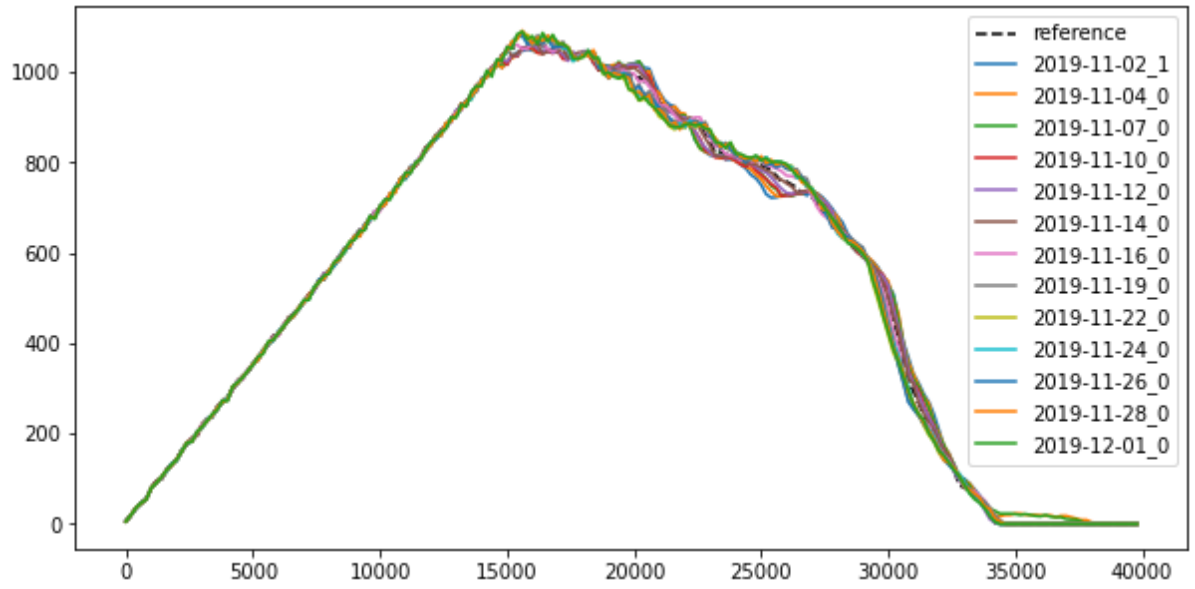


Fig 5.7: CDH comparison for month November 2019

The CDH of detected icebergs is almost similar to that of the reference CDH because the iceberg D28 calved from emery ice shelf in september 2019

The process explained here for tracking is followed for 1 month and after that again start date and end date is set to 1st and 2nd date of next month which in our case is 1/12/19 and 2/12/19 and center of the search is now centroid of last detected iceberg in the month of November 2019 and so on the process continues. The trajectory of the iceberg in the month is also showed in map using folium.

After the tracking of 1 month is completed we display three things, first the information of all detected icebergs, then CDH comparison between all valid icebergs and the reference iceberg and also the trajectory of the iceberg for that month.

As the iceberg travels in the ocean, it gradually melts due to several reasons like higher air temperature, rise in water temperature or it may break due to interaction with ocean waves. This results in the change of centroid distance

histogram of the iceberg. So, similarity consecutively decreases on the further months.

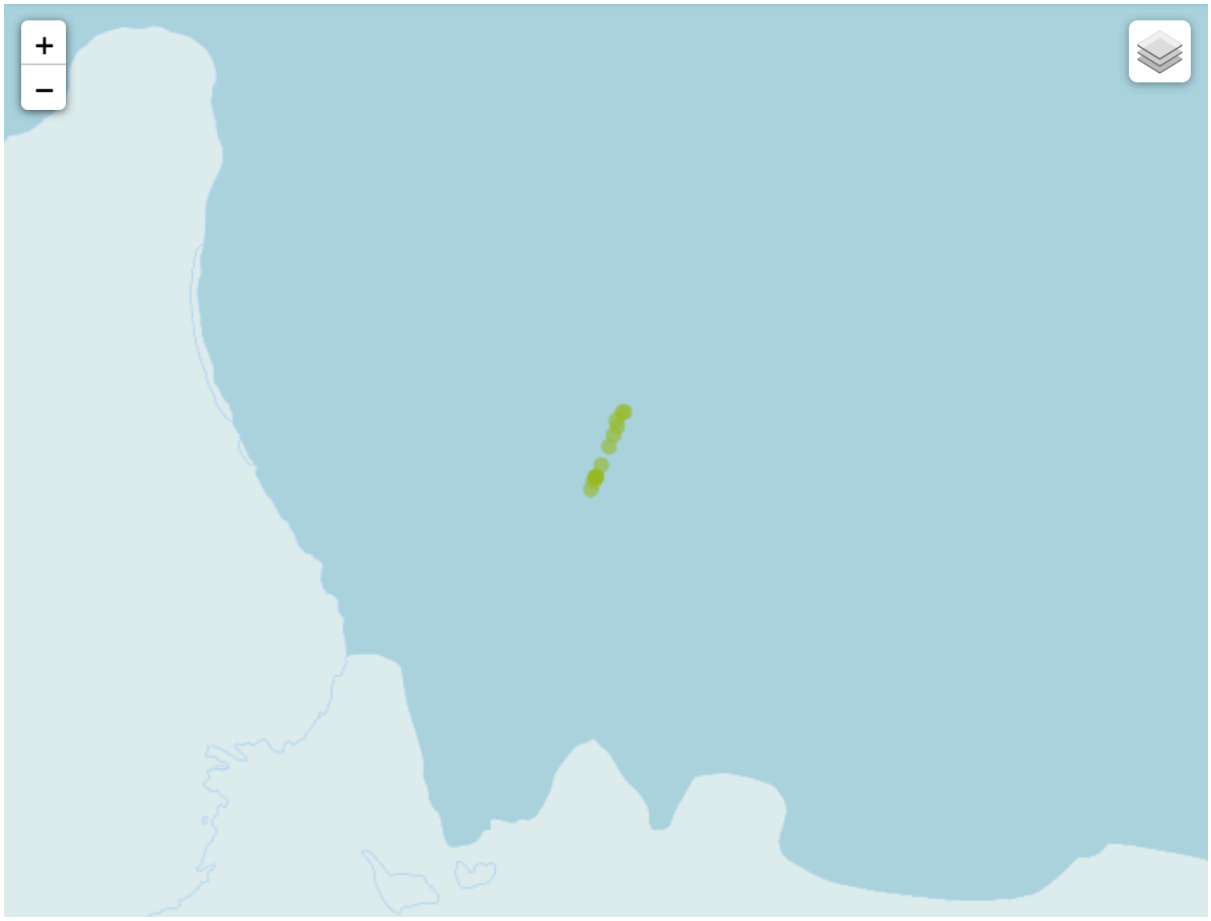


Fig 5.8: Trajectory of iceberg in the month of November 2019

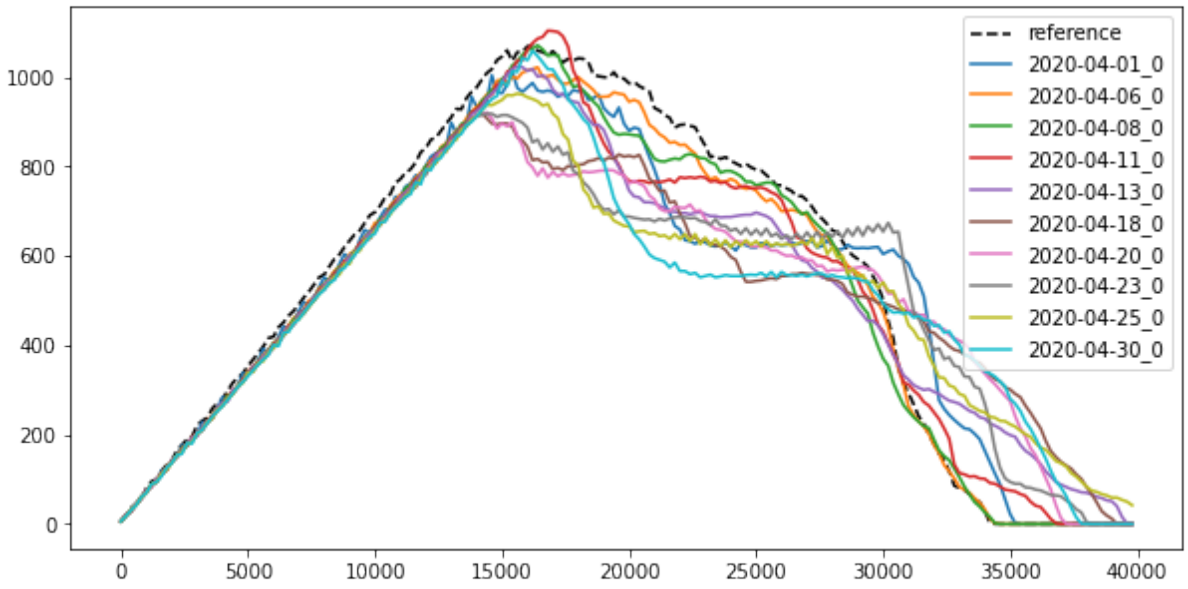


Fig 5.9: CDH comparison for month April 2020

It can be clearly seen in fig 5.9 that CDHs of detected icebergs in April 2020 are less similar to the reference CDH than that in November 2019.

In above process, if no image is found in 5 to 6 days then the radius for searching the image becomes too large (350000 m) to be processed so the code stops in such case and we have to manually find the next image and iceberg from that image.

Also after sometime, there is a possibility of melting the iceberg too much or breaking off the iceberg that it changes its shape significantly, in that case the first reference CDH would no longer be representative of the iceberg and a new reference iceberg would have to be manually set and its CDH has to be calculated. In our project, we changed the reference iceberg CDH after 1 year of tracking.

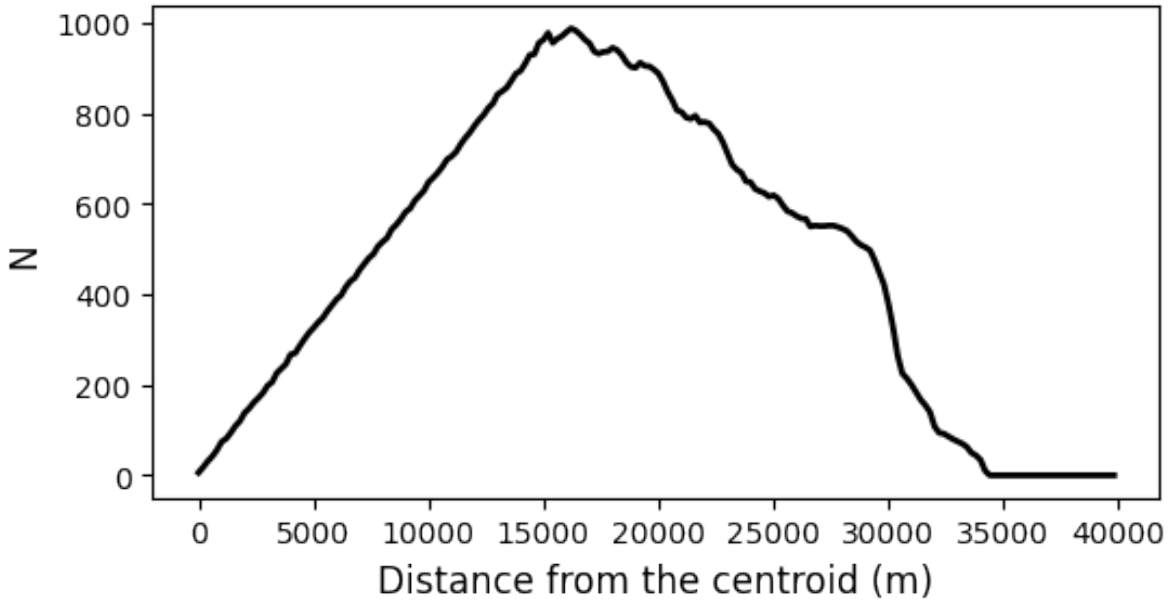


Fig 5.10: CDH of new reference iceberg (27/09/2020)

As the iceberg which is being tracked is very large there is not much change in CDH but still there is some change in area. The area of the new reference iceberg is 1587.73 km^2.

5.3 Trajectory of the iceberg

The path followed by the iceberg during the course of its journey throughout the ocean is called the trajectory of the iceberg. It shows us where the iceberg went from its initial position (from where it was calved) to its current position.

In the tracking process, we saved all the valid detected iceberg information as a geoJSON file and exported it to drive. This information is used to display the trajectory of the iceberg. We use the **os** library to get all the **geoJSON files** and then iterate over them to get centroid information of each iceberg and then convert the centroid to a **ee.FeatureCollection** and then add colour to the centroid according to the month by using **FeatureCollection.draw(color code)** method which converts it to ee.Image and then it is displayed on map using folium.

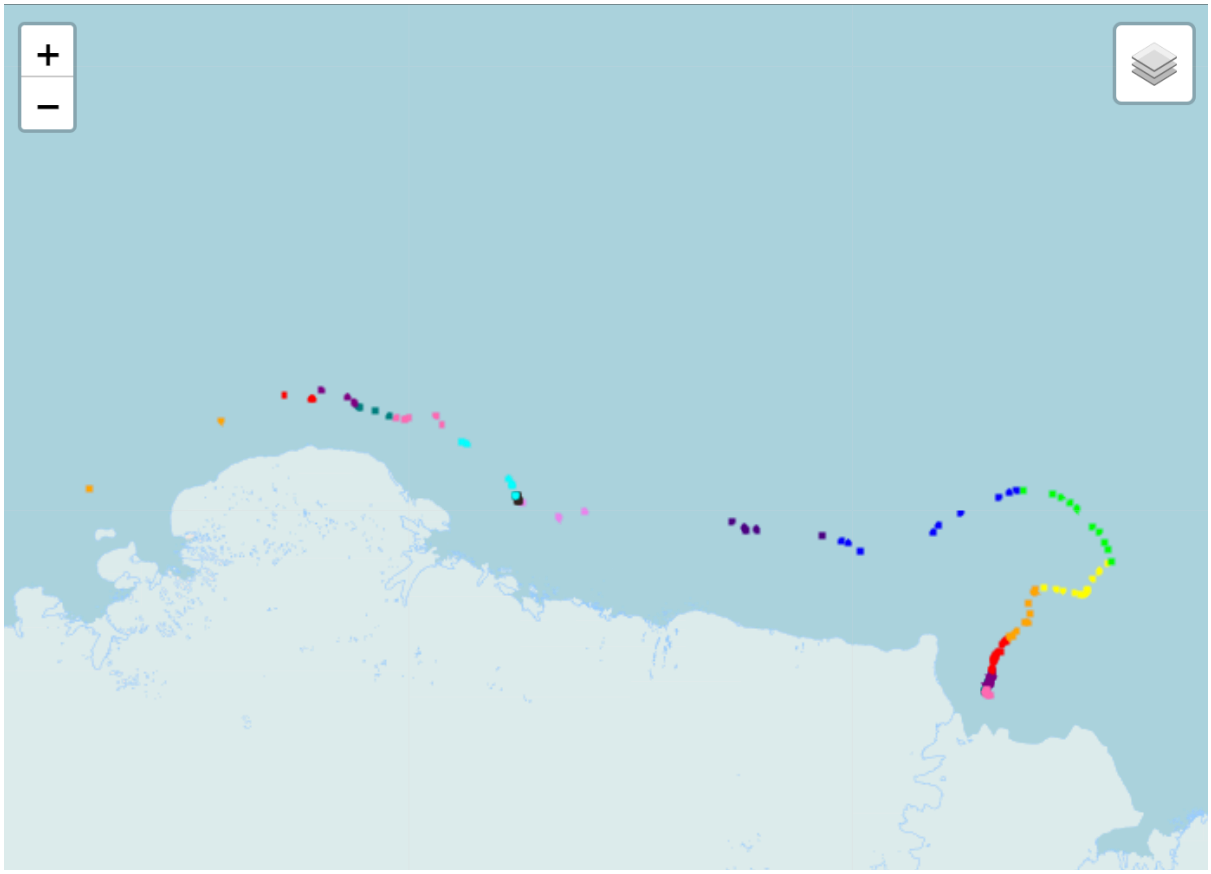


Fig 5.11: Trajectory of iceberg D28 (upto February 2021)

The dots are color coded according to month of the year

January - Red	July - Violet
February - Orange	August - Brown
March - Yellow	September - Cyan
April - Green	October - Pink
May - Blue	November - Teal
June - Indigo	December - Purple

The above image shows the trajectory of iceberg D28. Many observations can be made from this trajectory. The iceberg was calved in September 2019 so we started tracking from October 2019. In start, from October 2019 to December 2019, the movement of the iceberg is very slow which can be observed in the trajectory. After that in 2020 the movement of the iceberg became quite fast. Also it can be observed that in 2020 the some points in august and September are overlapped, it is because the iceberg was grounded from august to mid September in 2020.

We can also tell about availability of the images containing iceberg. From October 2019 to may 2020, we can see continuous points of iceberg centroid i.e images were available with good frequency (10 to 12 images per month). After that, frequency of image gets reduced to 3 to 5 images per month.

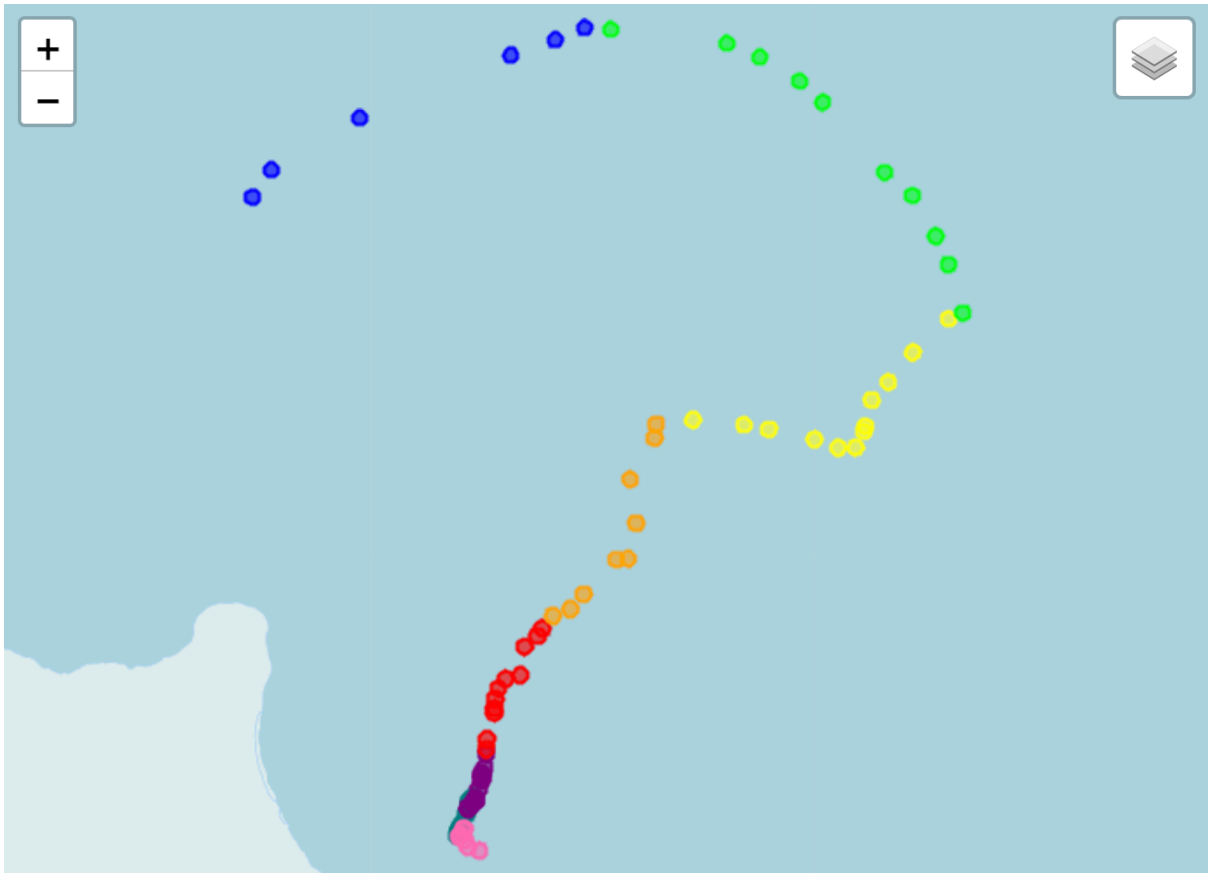


Fig 5.12: trajectory from October 2019 to may 2020 showing good image frequency

There is large gap between June 2020 and July 2020 because there was no image found between 20/06/2020 to 10/07/2020. This could mean that iceberg movement was fast in this duration due to several reasons. The grounding event occurred around 06/08/2020 to 15/09/2020 [3].

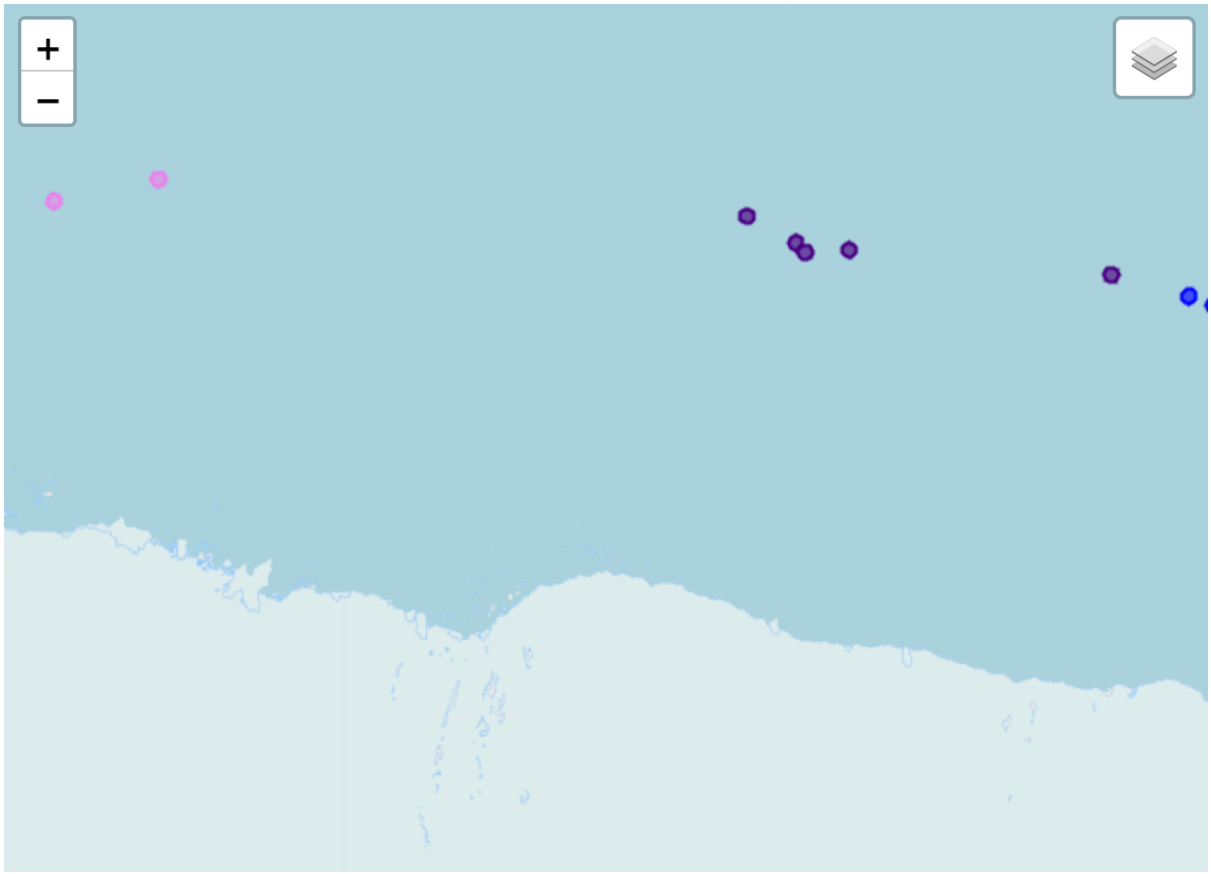


Fig 5.13: image showing gap in trajectory between June to July 2020

Here, the trajectory is shown upto February 2021 as after February 2021 no iceberg was detected by our algorithm in march, so we had to find the image of iceberg in march manually. We found only 1 image in march 2021(28/03/2021) manually which is not shown in the trajectory and its coordinates are [44.13779034624673, -67.14908415025835] (long, lat). The last image in February 2021 was found on 26/02/2021 and its coordinates are [48.13312978712529, -66.29489767494789]. The distance between them is 200km which is very huge. The tracking of the iceberg is continued to find further trajectory.

We can also calculate the total distance and average speed of the iceberg. Total distance can be calculated by summing the distances between consecutive images by using **Geometry.distance(geometry2)** method which calculates distance between 2 geometries. **Total distance travelled by icebergD28** from October 2019 to March 2021 is approximately **2156.484 km**. Average speed can be calculated by taking average of speed obtained from distance of 2 consecutive images. The **average speed** of the iceberg is calculated to be **2.582 km/day**.

Chapter 6

Future extension and Conclusion

6.1 Future extension

This tracking algorithm can be used generally for detecting any iceberg. We have successfully tracked the iceberg D28 but there are many ways in which we can extend and add new features to this project.

As the iceberg travels in the ocean, its orientation changes due to direction of ocean currents and other reasons. We can find a way to compare 2 images and find out the change in orientation of the iceberg. We can find by how much degrees has the iceberg rotated and changed its orientation from the reference. Thus, detecting whether the iceberg has rotated or not can be an extension to the project.

Also, we can find how the iceberg is melting, from which side it has melted, variation in the iceberg area using the images and the centroid distance histograms. We can also find the the height & thickness of the iceberg using Altimeter data from satellite altimeters like **CryoSat-2 and ICESat-2.**

We can also try to optimize the code of the project by reducing time complexity but that would require detailed analysis of the code in terms of time & space complexity.

6.2 Conclusion

At the polar region, icebergs calve from glaciers and ice shelf. The aim of my project was to detect the icebergs calved and track it to find its trajectory and its current position. This data can be useful to avoid icebergs in the path of a ship, if any, and the ship can move in the right direction.

For detecting the icebergs sentinel-1 SAR data is used. I learnt about SAR technique of remote sensing and about SAR data. I also learnt about preprocessing of SAR data to be able to use it in iceberg detection & tracking. After learning about SAR, I read research paper and learnt about SNIC algorithm for iceberg detection and how I can use earth engine library to my benefit. I learned about iceberg tracking method from the research papers and implemented the iceberg tracking algorithm to successfully detect and plot the trajectory of the iceberg D28.

Bibliography

1. Koo, Y., Xie, H., Ackley, S. F., Mestas-Nuñez, A. M., Macdonald, G. J., and Hyun, C.-U.: Semi-automated tracking of iceberg B43 using Sentinel-1 SAR images via Google Earth Engine, The Cryosphere, 15, 4727–4744, https://doi.org/10.5194/tc-15-4727-2021, 2021.

2. Achanta, R. and Süsstrunk, S.: Superpixels and polygons using simple non-iterative clustering, Proc. – 30th IEEE Conf. Com- put. Vis. Pattern Recognition, CVPR 2017, January 2017, 4895– 4904, https://doi.org/10.1109/CVPR.2017.520, 2017.

3. Liu, X.; Cheng, X.; Liang, Q.; Li, T.; Peng, F.; Chi, Z.; He, J. Grounding Event of Iceberg D28 and Its Interactions with Seabed Topography. *Remote Sens.* **2022**, *14*, 154. https://doi.org/10.3390/rs14010154rs14010154 .

4. Article by Kelsey Herndon, Franz Meyer, Africa Flores, Emil Cherrington, and Leah Kucera in collaboration with the Earth Science Data Systems, (2020, April 16), what is Synthetic Aperture Radar, https://earthdata.nasa.gov/learn/backgrounders/what-is-sar .

5. Earth engine python api library. https://github.com/google/earthengine-api/tree/master/python .

6. Earth engine API reference, (2021, may 5), https://developers.google.com/earth-engine/apidocs .

7. Folium documentation, https://python-visualization.github.io/folium/ .