Size Invariant Ship Detection using SNAP API, Image & Signal Processing and High Performance Computing.

Ministry/ Organization name: **Indian Space Research Organisation**
Problem Statement : **Size Invariant Ship Detection (NM404)**
Team Name : **Team PirateSIHp**
Team Leader Name : **Meesum Ali** College Code : 1-3514357128

Idea / Approach details

Our idea revolves around using existing algorithms as well as well defined Deep Learning methods to accurately detect ships.

Having tinkered with the SNAP API to achieve our desired results and applied and having a proof of concept for partitioning our Orbit File into slices, we aim to reduce the errors in ship detection by filtering false results as well as parallelize the process of ship detection by cutting up the Orbit File into smaller portions to compute our algorithm on.

For a detailed explanation and code please refer :
https://github.com/makflakes/Size-Invariant-Ship-Detection

Size Invariant Ship Detection

Team PirateSIHp

(Meesum Ali, Ch Saaketh, Anushka Singh, Abdul Mateen, Qurram Zaheer, Abdul Khaliq)

01

OBTAIN SAR DATA

Data is obtained through ISRO's SIH Dataset on Size Invariant Ship Detection

02

APPLY ORBIT FILE

We apply an orbit file on the SAR image to get more accurate features for analysis.

03

VECTOR MASK

To differentiate between Land and Sea

04

CALIBRATION & ADAPTIVE THRESHOLDING

To better identify objects in the sea area of the image.

05

OBJECT DETECTION

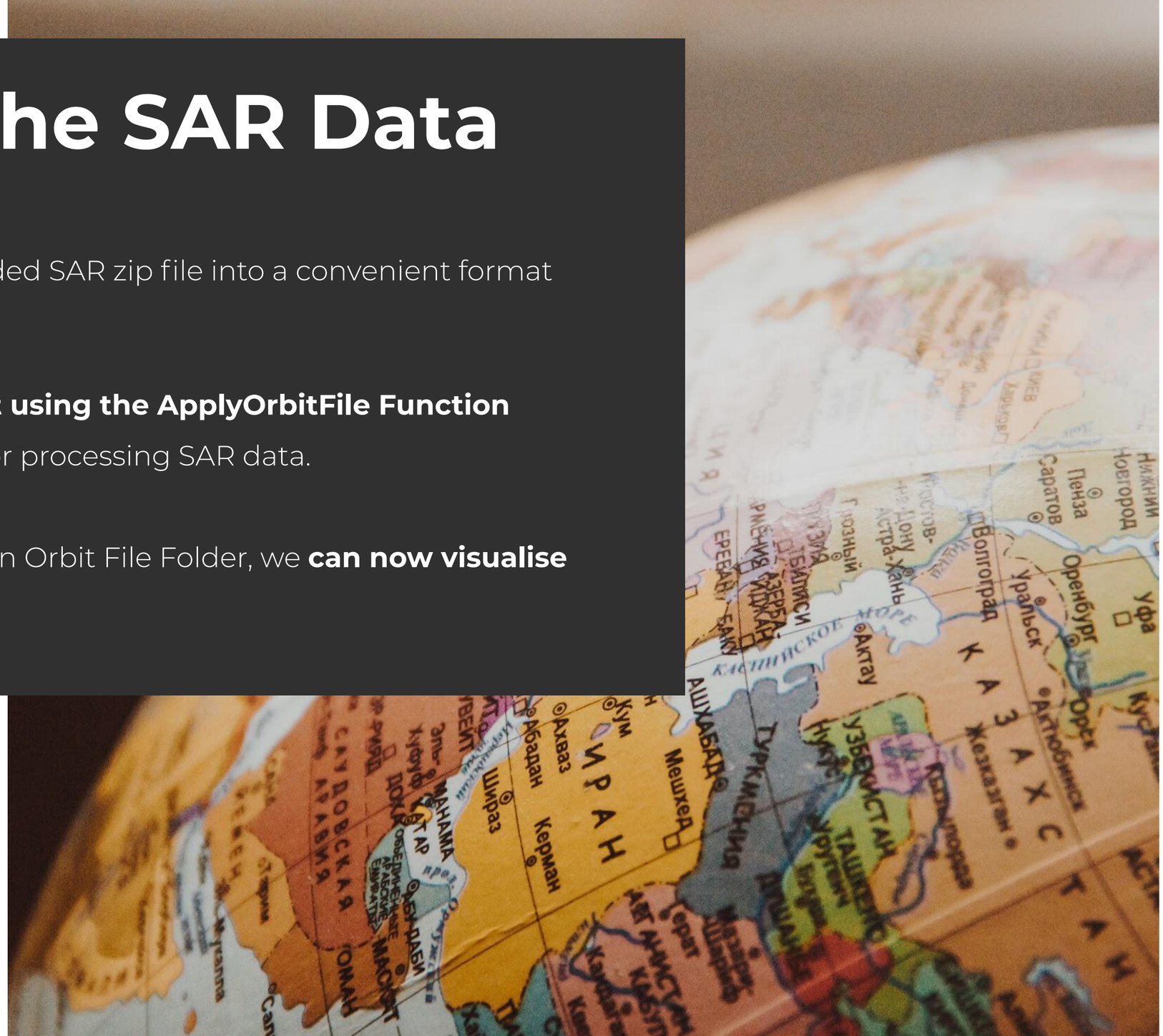
To identify objects that are only ships in the sa.

Extracting The SAR Data

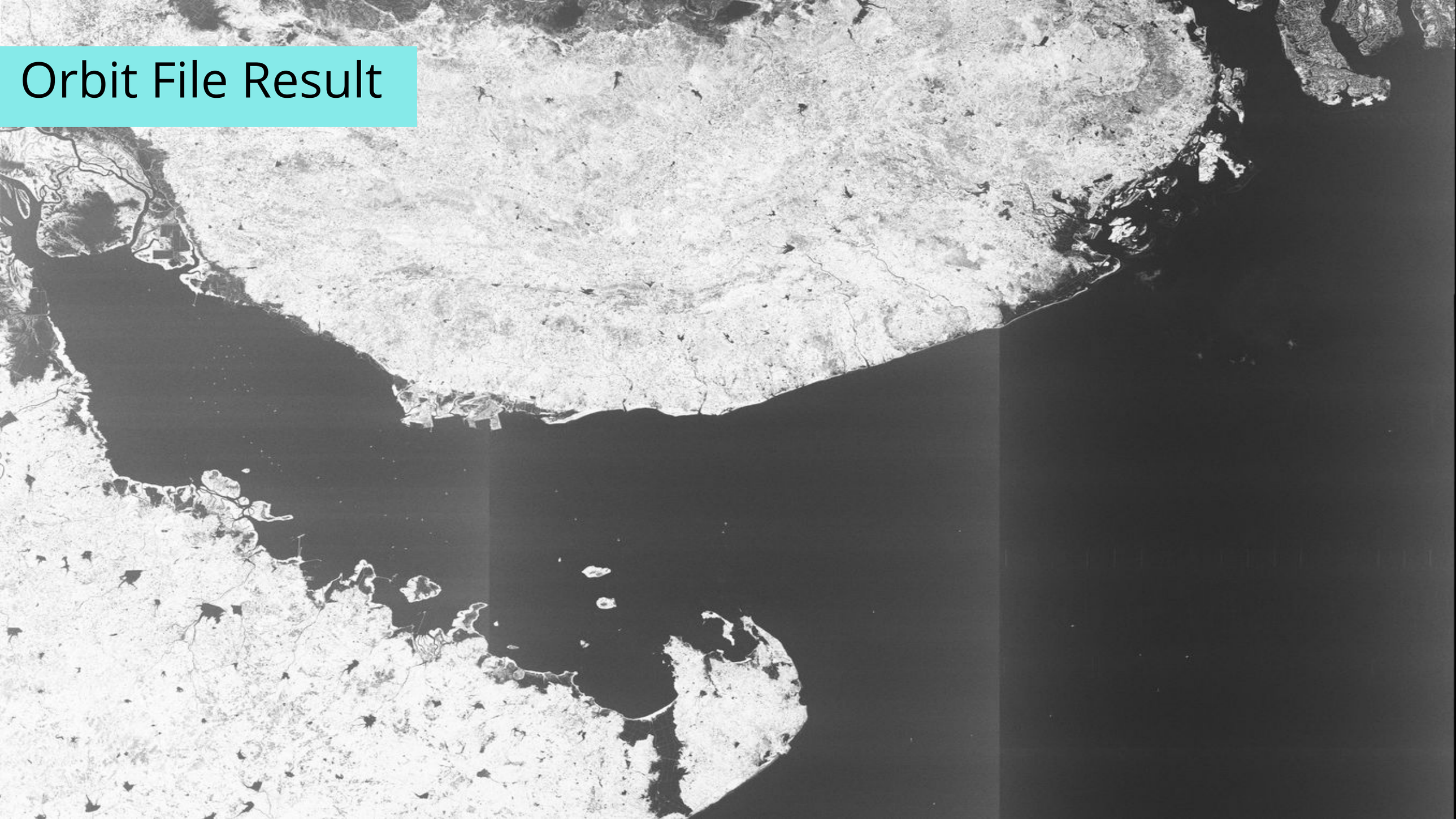
Our first step is to extract the provided SAR zip file into a convenient format for our purposes.

In this particular case, **we extract it using the ApplyOrbitFile Function provided in the snappy module** for processing SAR data.

After extracting the SAR data into an Orbit File Folder, we **can now visualise the separate bands**.



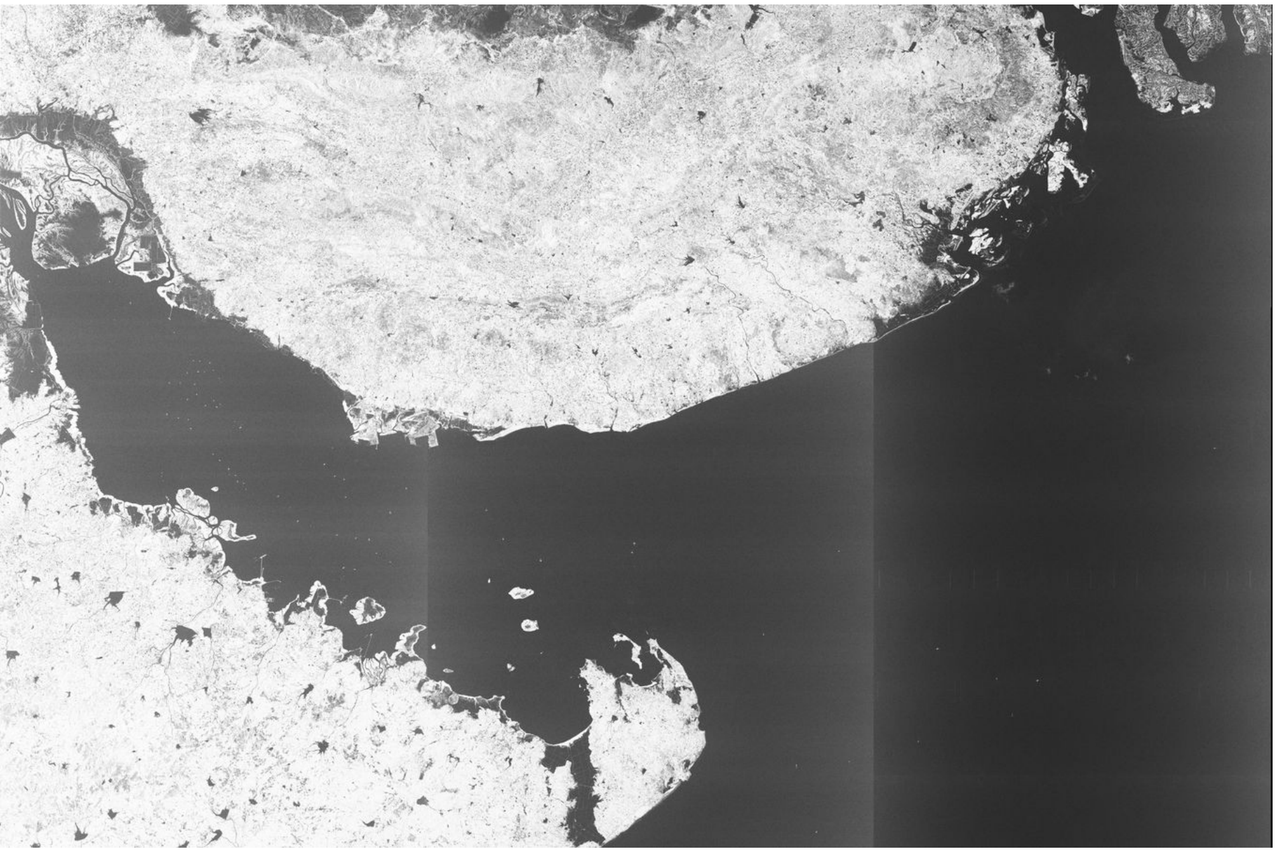
Orbit File Result



Add Vector Mask (Land-Sea)

The first step towards identifying objects (ships) in the given data is land-sea discrimination. **we achieve this by comparing our data with the SRTM** (Shuttle Radar Topography Mission) information, which accurately **maps the elevation of all regions around the globe**. Thus, **using latitude and longitude position**, we are able to mask out all **areas above sea level.**

Land Sea Mask



Original File

Land-Sea Mask

Calibration and Adaptive Thresholding

In this step, we apply a **radiometric calibration** to the Land-Sea Mask. This allows us to provide an imagery where the **values of the pixels are directly mapped to the radio backscatter of the SAR scene**.

We then perform **Adaptive Thresholding**. Adaptive Thresholding **tests each pixel under a central pixel, a new threshold value is created based on statistical analysis of the pixel's local background**. If the **pixel value is above the threshold, the pixel is then classified as a target pixel**. We use the **Two-Parameter Constant False Alarm Rate (CFAR) Detector algorithm** for this adaptive thresholding.

Object Detection

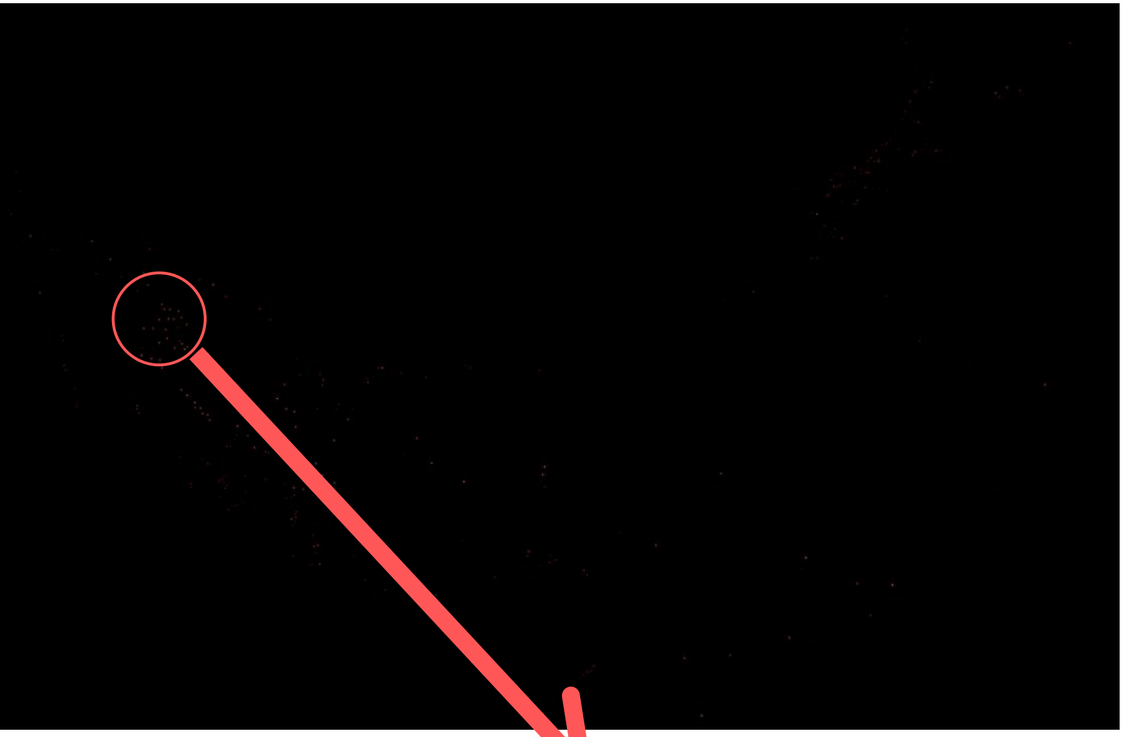
After Adaptive Thresholding, we are left with a clean, binarized image where the sea objects (ships) are easily distinguishable. We also set a minimum and maximum target size to help get rid of cases such as islands or large floating debris.

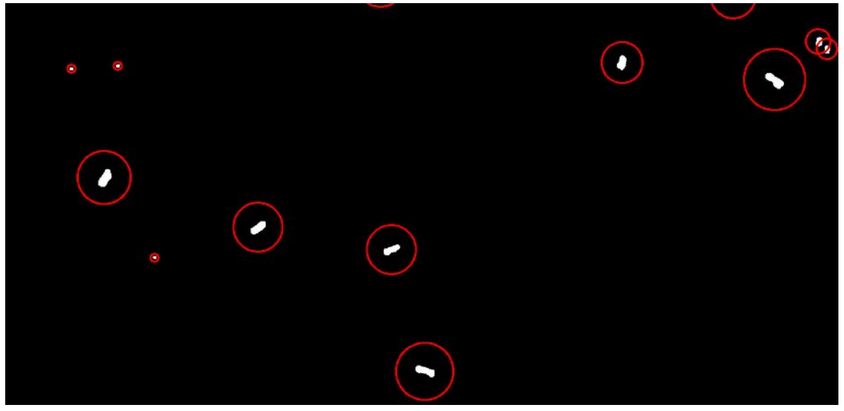
Minimum Target Size (m): 30 (target with dimension smaller than this threshold is eliminated)
Maximum Target Size (m): 600 (target with dimension larger than this threshold is eliminated)

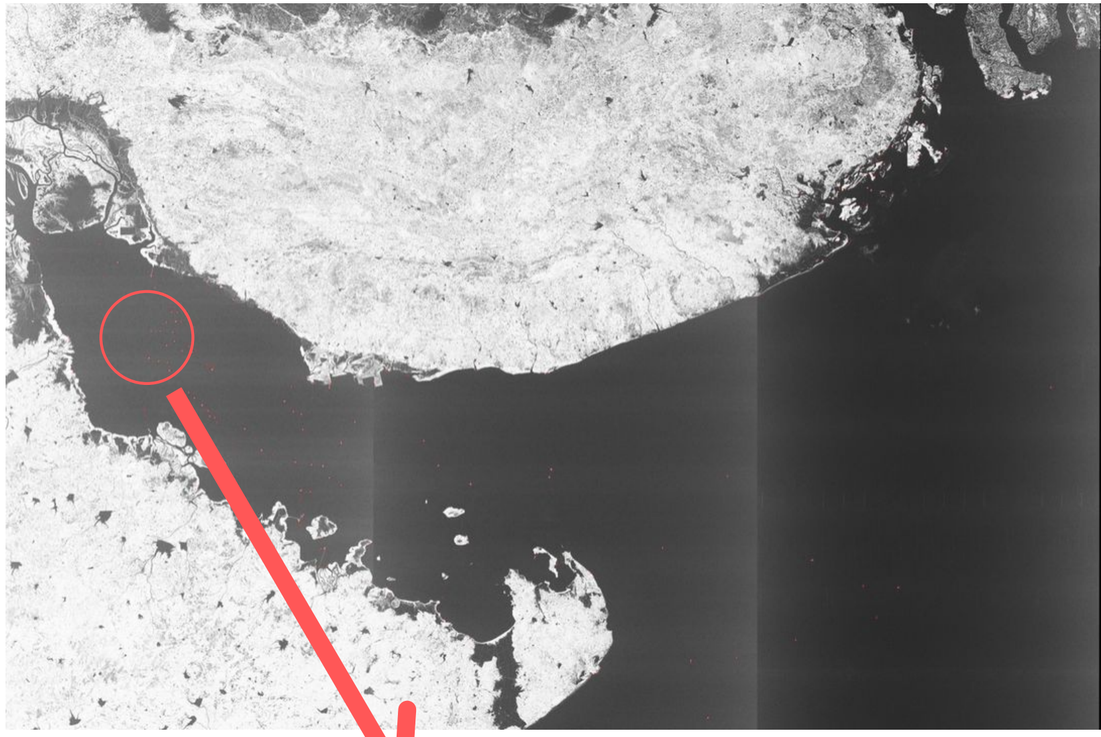
This will then give us an output file highlighting only those objects in the sea-area that fit this dimensional requirement.

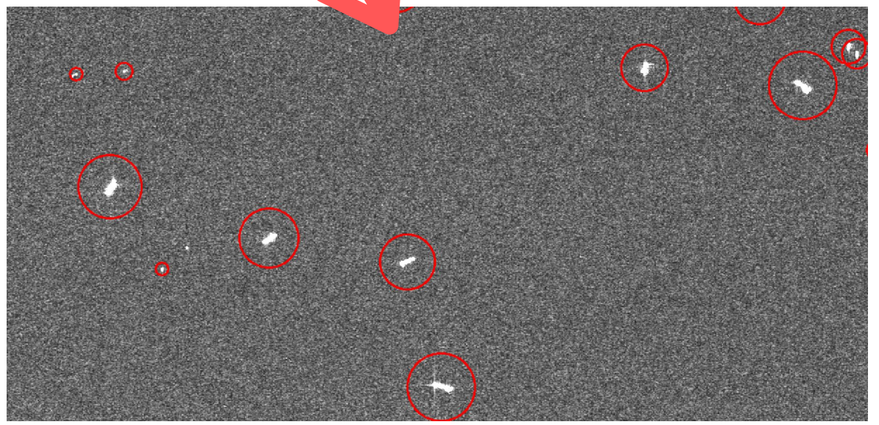
For the purpose of clarity, we wrote a script to put red circles around the ships so as to differentiate them.

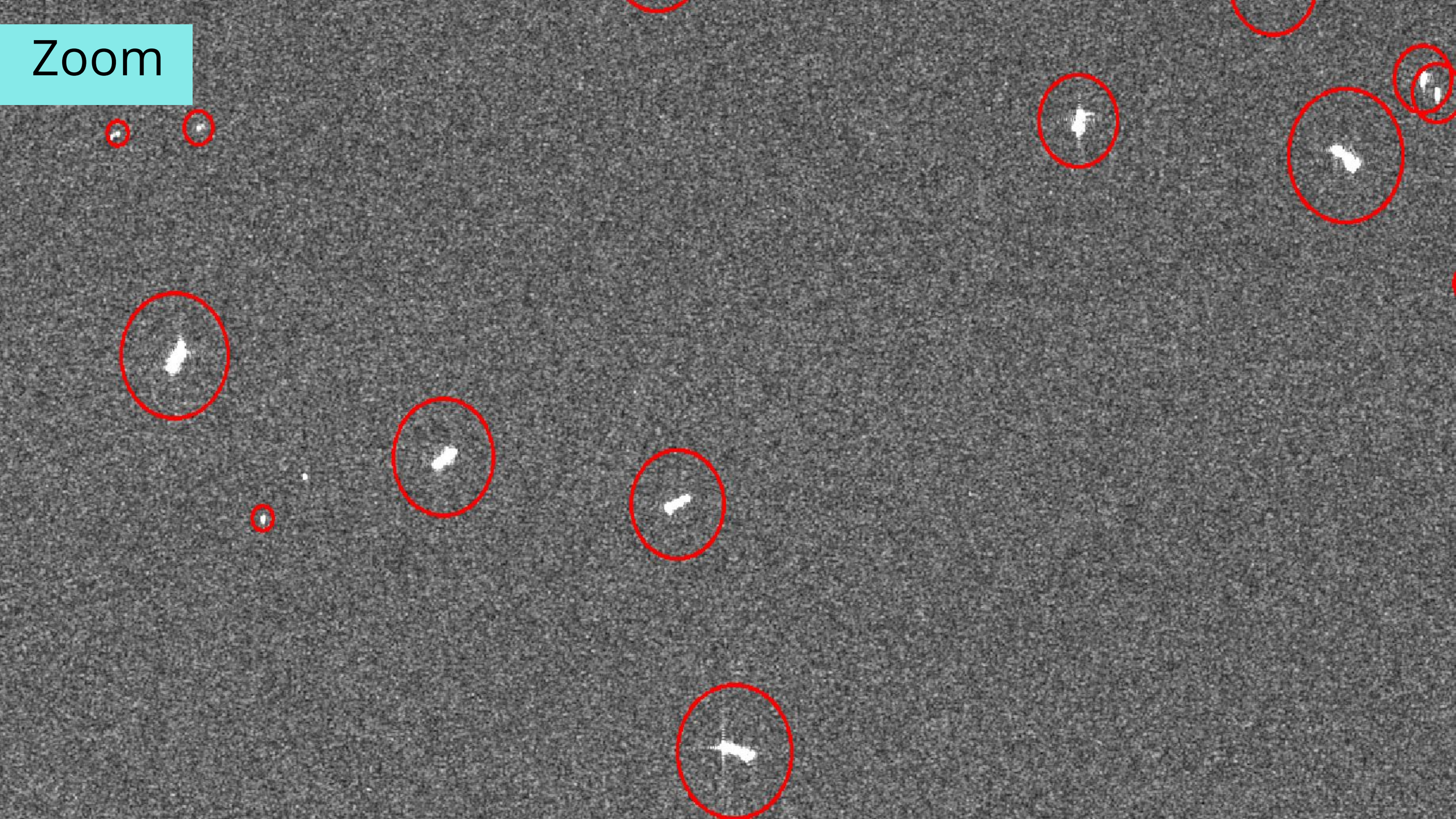
Ships Detected on the mask and Superimpose

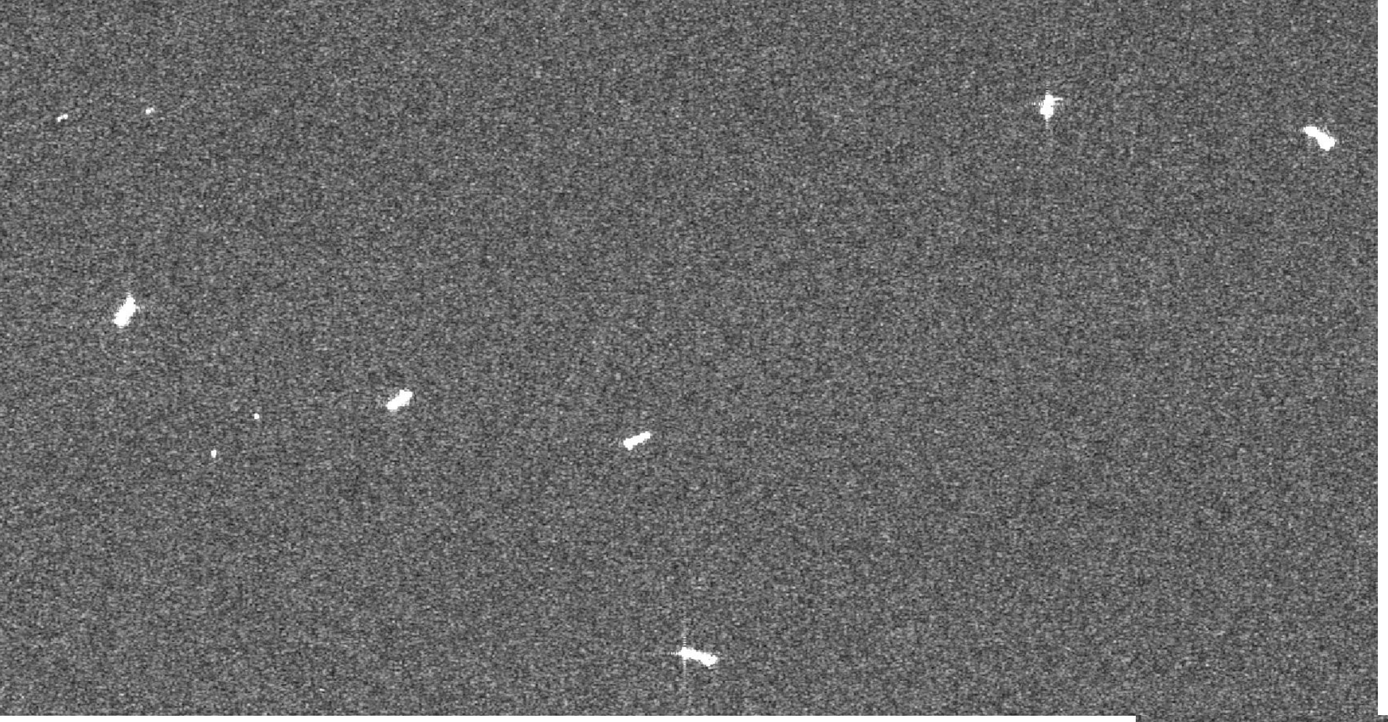






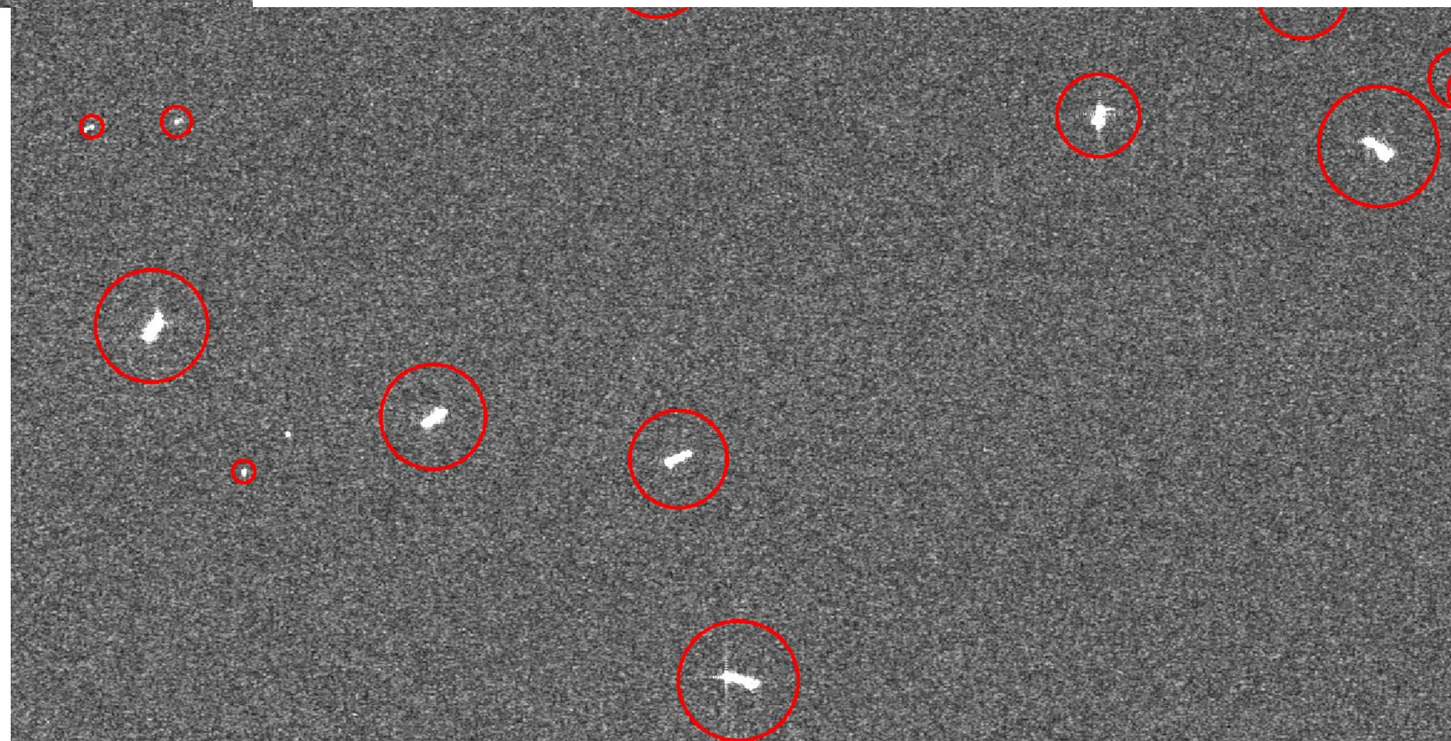



Zoom



Original Objects

Detected Ships



Resulting Data

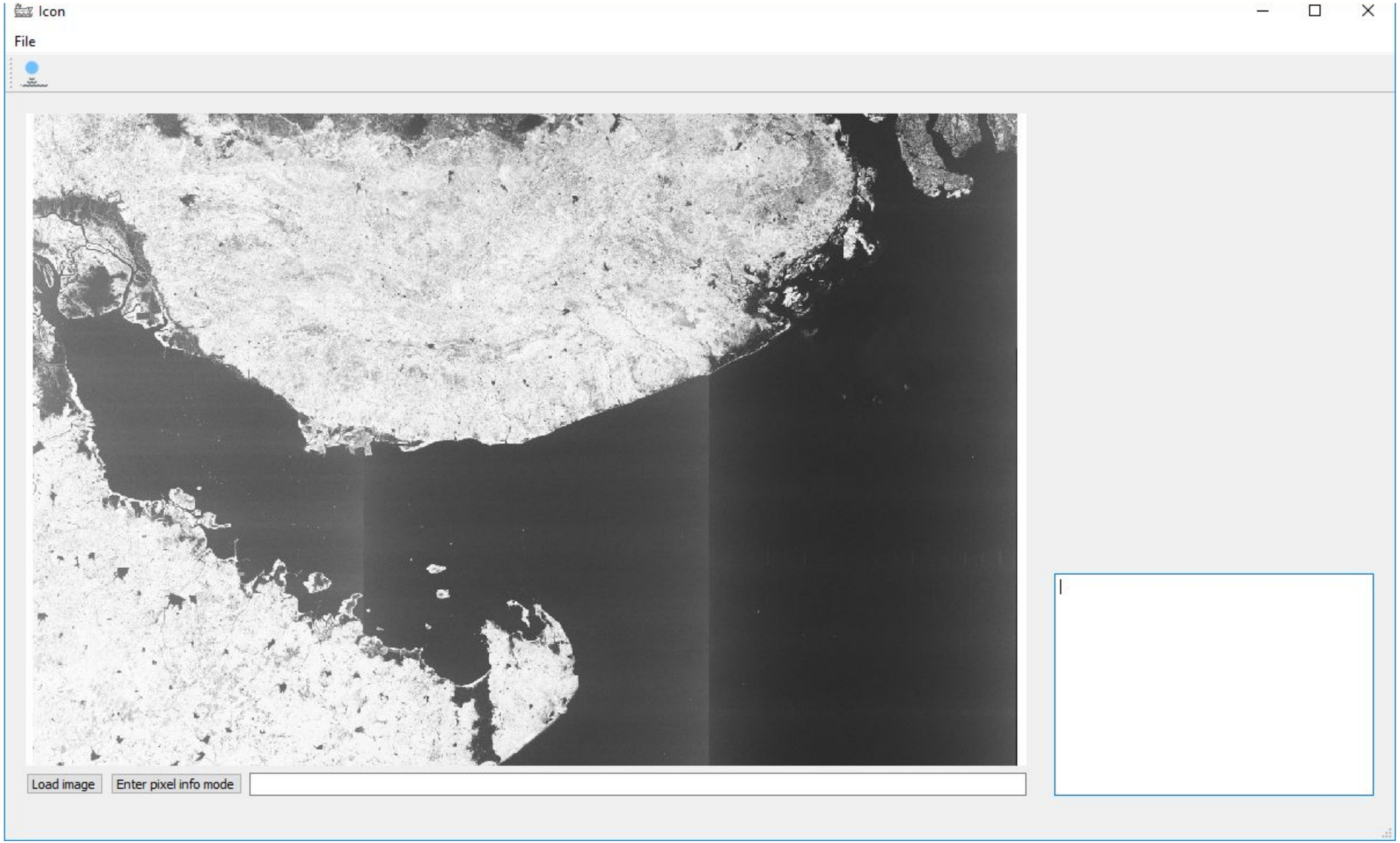
ShipDetections	geometry	Detected_x	Detected_y	Detected_lat	Detected_lon	Detected_width	Detected_length
target_000	POINT (21134 55)	21134	55	25.04990134	66.71050684	90	120
target_001	POINT (21327 301)	21327	301	25.03080193	66.68727268	160	180
target_002	POINT (22504 151)	22504	151	25.06295549	66.57509492	60	60
target_003	POINT (22728 27)	22728	27	25.07766081	66.55546918	50	40
target_004	POINT (21786 1369)	21786	1369	24.94187467	66.62367694	70	70
target_005	POINT (21211 1421)	21211	1421	24.9280128	66.67873504	100	60
target_006	POINT (22226 1372)	22226	1372	24.9485706	66.58068105	80	90
target_007	POINT (14042 1782)	14042	1782	24.77991216	67.37021664	230	200
target_008	POINT (15673 1784)	15673	1784	24.80634791	67.21153992	50	110
target_009	POINT (18378 1648)	18378	1648	24.86231537	66.9507129	50	70
target_010	POINT (18370 1663)	18370	1663	24.86078948	66.95121396	70	80
target_011	POINT (18127 1764)	18127	1764	24.84778149	66.9730623	90	80
target_012	POINT (21290 1516)	21290	1516	24.92075099	66.66936185	80	50
target_013	POINT (21258 1564)	21258	1564	24.91587106	66.67162064	280	160
target_014	POINT (21243 1635)	21243	1635	24.90927843	66.67183286	60	70
target_015	POINT (21818 1800)	21818	1800	24.90354689	66.61288893	100	110
target_016	POINT (21866 1820)	21866	1820	24.90245837	66.60784866	20	40
target_017	POINT (21106 1611)	21106	1611	24.90925608	66.68565606	50	50
target_018	POINT (14280 1863)	14280	1863	24.77649236	67.34559369	70	40
target_019	POINT (14406 2031)	14406	2031	24.76339497	67.3302758	70	180
target_020	POINT (15276 2076)	15276	2076	24.77352036	67.24485073	30	40
target_021	POINT (14969 1858)	14969	1858	24.78822824	67.27863204	160	70
target_022	POINT (14781 1877)	14781	1877	24.78345004	67.29657165	40	30
target_023	POINT (14697 1928)	14697	1928	24.77747302	67.30386225	270	150
target_024	POINT (14620 1973)	14620	1973	24.77216766	67.31048216	200	190
target_025	POINT (15965 2082)	15965	2082	24.78416752	67.17771928	70	60
target_026	POINT (15643 2120)	15643	2120	24.7755166	67.20835395	50	40
target_027	POINT (15964 2151)	15964	2151	24.77793746	67.17651692	100	80
target_028	POINT (18874 1952)	18874	1952	24.84282838	66.89696705	70	80
target_029	POINT (17468 2167)	17468	2167	24.80086879	67.02996021	90	50
target_030	POINT (16741 2177)	16741	2177	24.78819489	67.10046182	60	60
target_031	POINT (16741 2177)	16741	2177	24.78819489	67.10046182	60	60
target_032	POINT (18576 2028)	18576	2028	24.83123609	66.92462269	70	30

The result of the entire process is the metadata file that gets generated.

Some important data the metadata file contains are :

1. Latitude and Longitude of detected ships.
2. No. of ships.
3. Estimated length and width of the ship.

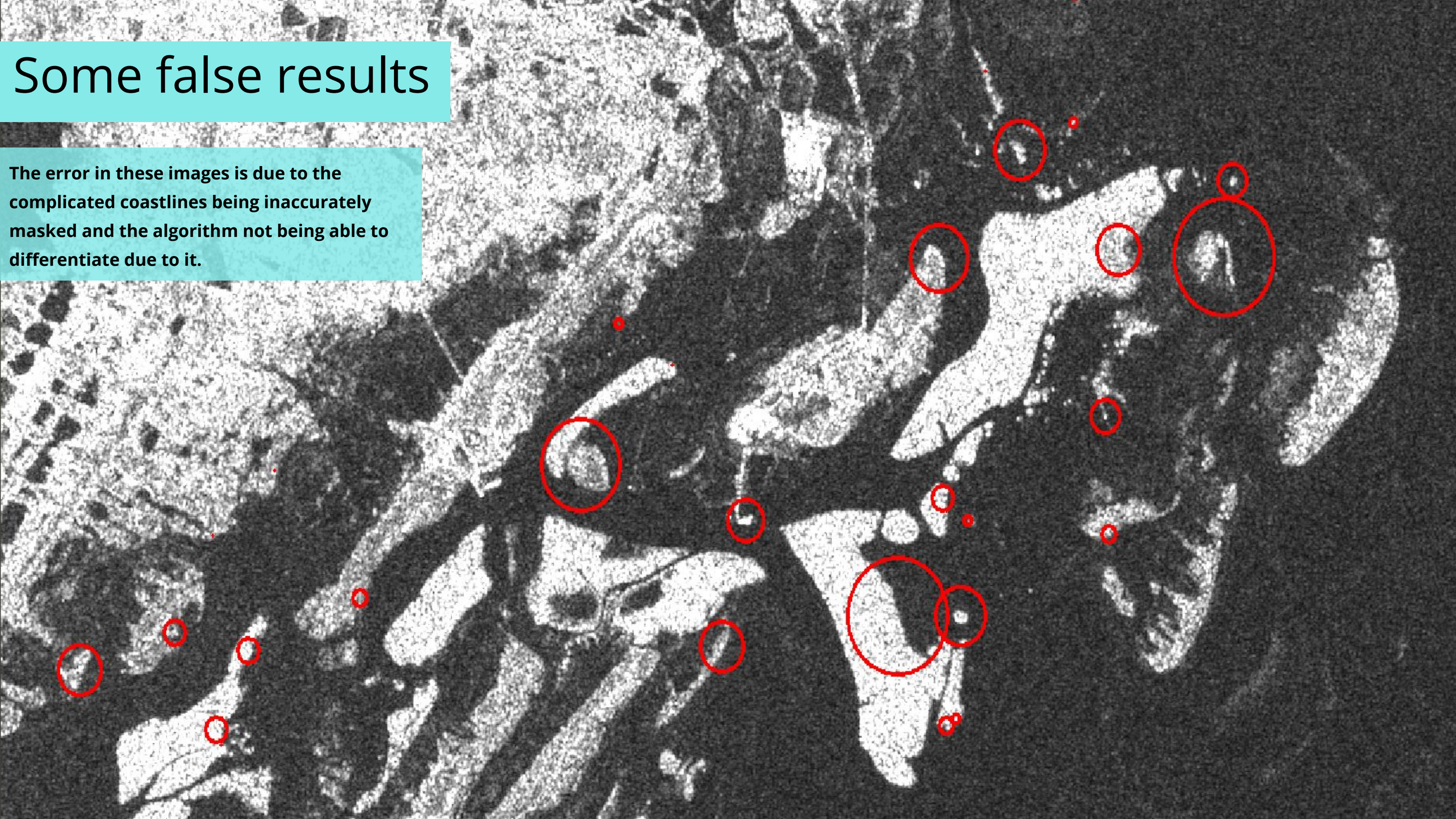
GUI Demo



Demo of the PyQT5 GUI, which is part of our end-to-end solution for SAR based ship detection. (Still a work in progress)

Some false results

The error in these images is due to the complicated coastlines being inaccurately masked and the algorithm not being able to differentiate due to it.



Problems Encountered and Showstoppers

Plan to improve upon the following :

- Improving accuracy of the Land-Sea Mask around coastlines.
- Improving accuracy of ship detection.
- Improve Adaptive Thresholding Algorithm to retain ship dimensions.
- Filtering out false detections of other sea objects.
- Cutting up the images into smaller pieces to parallelize the process and increase speed.

Technology Stack

Technology Stack :

Python
OpenCV
Tensorflow
snappy
ESA SNAP
pillow
Flask
PyQT 5

Use Cases and Conclusion

Use Cases :

Military Intel

Border Security

ETA for ships

Threat Detection

Debris detection

Rescue Missions

As it stands, we are able to extract the number of detected ships and their sizes. Our current solution aims to improve upon the accuracy of ship detection (improving landmask, adaptive thresholding etc) implementing the model in parallel by splitting the image into smaller parts and computing our algorithm on each part in parallel. This would mean a faster and more accurate solution.

Direction of progress:

-Make the object detection model a lot better by involving Deep learning to classify ships using the training data provided.

-Determining type of ship based on parameters like size (of region in the SAR image), shape etc.

-Dividing the very large image into smaller images in the GUI so we can parallelize the process of detecting ships and make it exponentially faster, possibly even in real time.