SHORELINE EXTRACTION FROM SENTINEL 2 REFLECTANCE PRODUCTS AND INTERTIDAL DEM FOR CHITTAGONG REGION

Nazmuddoha Ansary

Table of Contents

Chapter 1: Introduction

The shoreline of Bengal delta is a very dynamic, rapidly changing entity. The purpose of this work is to develop an automated workflow to extract shoreline from SENTINEL 2 REFLECTANCE PRODUCTS. SENTINEL 2 Level 2A/B Images are provided at highest 10m resolution. For processing of this very high resolution and high volume data, we have adopted the methodology presented by Matthias et al. 2018 where they have used Proba-V Images at 100m resolution. . The processing system is implemented in Python for flexible deployment and rapid development..

Objectives

The objectives of this work is -

- To develop a python based processing tool to extract the shoreline from SENTINEL 2 Level 2 data using a robust shoreline extraction algorithm.
- To implement scripts to generate intertidal DEM using the knowledge of water level along with the position of the shoreline.

Organization of the Report

The report is organized in 4 chapters.
In Chapter 2 - A brief introduction to SENTINEL 2 REFLECTANCE PRODUCTS is presented with respect to our specific needs.
In Chapter 3 - The algorithm for each step is described keeping in mind the highest possible similarity with *Matthais et.al 2018*
In Chapter 4 - The program flow is described with output results with the presented implementation.

Chapter 2: Data Description

2.1 Information about the product

Sentinel 2 reflectance products are distributed from theia.cnes.fr as a single zip file for a specific time per tile. It Contains the following files:

- A metadata file in XML format
- A quick look file in JPG format
- Multiple band files in GEOTIFF format
- A sub-repository of MASKS in GEOTIFF format

The unzipped product creates a folder which contains information about-

- Satellite instrumentation
- Date-time
- Product level
- Zone-tile-orbit number
- Metadata type
- Metadata version number

EXAMPLE:PRODUCT INTRODUCTION

Unzip folder: SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5
Satellite instrumentation :SENTINEL2B
Date(YYYYMMDD to DD-MM-YYYY):21-02-2018
Time(HHMMSS-mili S format):04:28:03s 458 ms
Product Level:L2A
Geographical Zone:T46QCK
Metadata type:D
Metadata Version:V1-5

2.2 Information about Band Data

The product comes with 19 band files with two types of resolution - 10m and 20m. The data is also divided on the basis of slope effects as following -

- SRE -image in ground reflectance without the correction of slope effects
- FRE -image in ground reflectance with the correction of slope effects

#Table: Band properties of Sentinel 2A and 2B Level 2 Dataset

BAND NUMBER	BAND TYPE	SPATIAL RESOLUTION (m)	Central Wavelength (nm)	Band Width (nm)
B2	BLUE	10	490	65
B3	GREEN	10	560	35
B4	RED	10	665	30
B5	Vegetation red edge	20	705	15
B6	Vegetation red edge	20	740	15
B7	Vegetation red edge	20	783	20
B8	NIR	10	842	115
B8A	Vegetation red edge	20	865	20
B11	SWIR	20	1610	90
B12	SWIR	20	2190	180
ATB	Atmospheric and biophysical parameters	10 and 20	443	20

B2-B12:

- All of the band from B2-B12 has a single raster band.
- General format: <Identifier>_<FRE/SRE>_B<num>.tiff

ATB:

- The ATB band contains two raster band.
- 1st raster contains the water vapor content (WVC)
- 2nd raster contains aerosol optical thickness (AOT)
- <IDENTIFIER>_ATB_R1.tif is for 10m resolution.
- <IDENTIFIER>_ATB_R2.tif is for 20m resolution

2.3 Information about Mask Data

The mask data subrepository contains mask data in two spatial resolutions and usually contains the following data:

- **SAT:** saturation mask coded over 8 bits
- **DFP:** defective pixels mask coded over 8 bits
- **CLM:** cloud mask made of 1 band coded over 8 useful bits:
    - *1st bit (CM1):* cloud_mask_all, result of a "logical OR" for all the cloud and shadow mask
    - *2nd bit (CM2):* cloud_mask_all_cloud, result of a "logical OR" for all the cloud masks
    - *3rd bit (CM3):* cloud_mask_refl, cloud mask identified by a reflectance threshold
    - *4th bit (CM4):* cloud_mask_refl_var, cloud mask identified by a threshold on reflectance variance
    - *5th bit (CM5):* cloud_mask_extension, cloud mask identified by the extension of cloudmasks
    - *6th bit (CM7):* cloud_mask_shadow, shadow mask of clouds inside the image
    - *7th bit (CM8):* cloud_mask_sahdvar, shadow mask of clouds outside the image
    - *8th bit (CM9):* cloud_mask_cirrus, cloud mask identified with the cirrus spectral band

- **MG2:** geophysical mask of level 2, made of 1 band coded over 8 useful bits:
  - *1st bit (WTR):* water mask
  - *2nd bit (CM2) :* cloud_mask_all_cloud, result of a "logical OR" for all the cloud masks
  - *3rd bit (SNW):* snow mask
  - *4th bit (logical OR between CM7 and CM8):* shadow masks of clouds
  - *5th bit (SHD):* topographical shadows mask
  - *6th bit (HID):* hidden areas mask
  - *7th bit (STL):* sun too low mask
  - *8th bit (TGS):* tangent sun mask
- **EDG:** edge mask coded over 8 bits
- **IAO:** interpolated AOT pixels mask

Depending on the version of metadata the mask subrepository may contain **IAB** instead of **IAO** (metadata version>= 1.7).Also the repository may also contain **DTF** masks for 10m and 20m resolution.

EXAMPLE: PRODUCT TREE

```
SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5
    ├── MASKS
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_CLM_R1.tif
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_CLM_R2.tif
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_DFP_R1.tif
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_DFP_R2.tif
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_DTF_R1-D09.tif
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_DTF_R1-D10.tif
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_DTF_R1-D11.tif
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_DTF_R1-D12.tif
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_DTF_R2-D09.tif
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_DTF_R2-D10.tif
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_DTF_R2-D11.tif
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_DTF_R2-D12.tif
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_EDG_R1.tif
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_EDG_R2.tif
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_IAO_R1.tif
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_IAO_R2.tif
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_MG2_R1.tif
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_MG2_R2.tif
    │   ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_SAT_R1.tif
    │   └── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_SAT_R2.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_ATB_R1.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_ATB_R2.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_FRE_B11.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_FRE_B12.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_FRE_B2.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_FRE_B3.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_FRE_B4.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_FRE_B5.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_FRE_B6.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_FRE_B7.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_FRE_B8A.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_FRE_B8.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_MTD_ALL.xml
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_QKL_ALL.jpg
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_SRE_B11.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_SRE_B12.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_SRE_B2.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_SRE_B3.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_SRE_B4.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_SRE_B5.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_SRE_B6.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_SRE_B7.tif
    ├── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_SRE_B8A.tif
    └── SENTINEL2B_20180221-042803-458_L2A_T46QCK_D_V1-5_SRE_B8.tif
```

Chapter 3: Algorithm for SHORELINE EXTRACTION

3.1 Selection of Data

Four Band Data are selected while keeping the overall processing scheme similar to Matthais et.al 2018:

#Table: Bands used for Processing

BAND NUMBER	BAND TYPE	SPATIAL RESOLUTION (m)	Central Wavelength (nm)
B2	BLUE	10	490
B4	RED	10	665
B8	NIR	10	882
B11	SWIR	20	1610

3.2 Preprocessing of Band Data

3.2.1 Cloud Mask Correction

A aggressive cloud mask is applied (i.e- All cloud except the thinnest) to the bands according to resolution.
[<IDENTIFIER>_CLM_R1.tif for B2,B4,B8]
[<IDENTIFIER>_CLM_R2.tif for B11]
The pixels with cloud are set to negative of reflectance value(i.e- reflectance=10000 for all products)

3.2.2 NAN Data Conversion

All the negative reflectance values are taken as NAN data(i.e- NoDATA/EDG and Cloud Data)

3.2.3 Normalization:

The band data is normalized as follows:

$$Band\_norm = \frac{Band - min(Band)}{max(Band) - min(Band)}$$

3.2.4 Upsampling of SWIR Band:

Since B11 is of 20m resolution, a simple nearest neighbour interpolation is done to B11 to upsample to 10m resolution.

3.3 Construction of channel Data:

Alpha applied Red,Green and Blue channel are constructed as follows with-

- B4_norm as RED
- B8_norm/NIR as Green
- B2_norm as Blue
- B11_norm/SWIR as Alpha

And applying the following equation:

$$red\_new = (1 - SWIR\_norm) + (SWIR\_norm \times red\_norm)$$
$$green\_new = (1 - SWIR\_norm) + (SWIR\_norm \times NIR\_norm)$$
$$blue\_new = (1 - SWIR\_norm) + (SWIR\_norm \times blue\_norm)$$

3.4 Construction of Hue and Value channel Data:

Hue and Value channel are constructed as the method implemented by Pekel et.al 2014[]

$$V = max(R,G,B) \quad \text{<Value channel>}$$

$$HUE = \begin{cases} 0 & ; \text{ if } R=G=B \\ [\{60 \times (G-B)\}/(V-min(R,G,B))] \bmod 360 & ; \text{ if } V=R \\ [\{60 \times (B-R)\}/(V-min(R,G,B))] + 120 & ; \text{ if } V=G \\ [\{60 \times (R-G)\}/(V-min(R,G,B))] + 240 & ; \text{ if } V=B \end{cases}$$

3.5 Construction of Binary WaterMap:

3.5.1 Binary Water Map Hue Channel:

[-]The generated Water Mask for the specific region is applied inversely to Normalized HUE channel

[-]The median and standard deviation is calculated over the Inverse water mask applied Hue channel.

[-]Then the binary water map from hue is calculated as follows:

$$BW\_hue = \neg \{ (hue < T\_hue + n\_hue * \sigma\_hue) \wedge (hue > T\_hue - n\_hue * \sigma\_hue) \}$$

I_Hue = Inverse water mask applied Hue channel

T_hue = Median (I_Hue)

σ_hue = Standard deviation (I_Hue)

N_hue = Scaling Factor

(After several test and trials N_hue was selected as 0.4)

3.5.2 Binary Water Map Value Channel:

[-]The generated Water Mask for the specific region is applied to Normalized Value channel

[-]The median and standard deviation is calculated over the water mask applied Value channel.

[-]Then the binary water map from Value is calculated as follows:

$$BW\_val = (val < T\_val + n\_val * \sigma\_val) \wedge (val > T\_val - n\_val * \sigma\_val)$$

I_val = Water mask applied value channel

T_val = Median (I_val)

σ_val = Standard deviation (I_val)

N_val = Scaling Factor

(After several test and trials N_val was selected as 5)

3.5.3 And Connotation Binary Water Map:

A final binary water map is formed through and connotation of BW_hue and BW_val as follows:

$$MapWater = BW\_hue \wedge BW\_val$$

3.6 Creation of Gross Water Mask:

The creation of gross water mask is done by averaging all B11 data present for a specific zone and thresholding with 0.5 factor of std value. I.e--
Water_mask[Cumulative_B11_Normalized <= 0.5 * std of Cumulative_B11_Normalized] =1 {water}
And rest data points are considered as Land

Chapter 4: Program Flow

The complete program consists of the following parts:

- Preprocessing
- Image Processing
- Utilities

The code is run from main.py present in root directory of the program. Before executing main.py there are a few positional and optional arguments that need to be set in the main.py script and a separate script depcheck.py has to be run separately to know if any of the dependencies are missing.

4.1 Tuning the parameters:

The Section:Parameters in main.py contains the following variables:

```
indatadir = # Directory of Zipped Data
wkdir = # Directory of saving unzipped data
prepdir = # Directory of saving preprocessed data
improcdir= # Directory of saving processed data
waterleveldir= # Directory of water level dat files
vertrefdir= # Saving Intermediate data for vertical referencing

##----Preprocessing Optional Params (default values)
stdfactor = (0.5)
# Threshold for creating watermask
i.e- data[data>factor*std]=Land (Float)
MaskWater = (10000)
# Water Mask creation water blob removal threshold
MaskLand = (5000)
# Water Mask creation land blob removal threshold
additionalDirectory = (None)
# Specialized testing Directory for watermask
##----processing Optional params
```

```
hue_channel_scaling_factor                    =                    (0.4)
# Scaling Factor of hue for median thresholding
value_channel_scaling_factor=                                      (5.0)
# Scaling Factor of Value for median thresholding
blob_removal_land=                                                 (10000)
# Binary water map blob removal size for land features
blob_removal_Water=                                                (50000)
# Binary water map blob removal size for water features

##----Boolean Params of png saving
prepWmaskPNGflag= False
# Save png for Water mask creation  (True/False)
procChannelPNGflag= False
# Save png while channel construction (True/False)
procWaterMapPngFlag= False
# Save png while binary water map creation (True/False)
procblobRemovalpngFlag= False
# Save png while filtering water maps (True/False)
```

4.2 Module description:

The program contains 4 modules as follows:

4.2.1. Preprocessing module (preprep) :

Extracting data (ingest):

preprep.ingest(indatadir,wkdir) function extracts zipped data from a specified input directory (**indatadir**) and saves the unzip data through and saves them to a specified directory (**wkdir**).

Creating Water Mask (genmask):

preprep.genmask(wkdir,prepdir,dir=additionalDirectory,nstd=stdfactor,water=MaskWater,land=MaskLand,png=prepWmaskPNGflag) function creates watermask for all the zones present in the unzipped data directory and combines all B11 data after normalizing and threshold by a specific factor of standard deviation s to create watermaks for all zones.

4.2.2. Image Processing module (improc) :

Constructing channels (construct_channels):

improc.construct_channels(directory,improcdir,prepdir,png=procChannelPNGflag) function creates the alpha applied normalized red,green,blue,hue and value channels with cloud mask correction

Create binary watermap (make_watermap):

improc.make_watermap(directory,improcdir,prepdir,nhue=hue_channel_scaling_factor,nvalue=value_channel_scaling_factor,png=procWaterMapPngFlag) function creates the binary water map with median thresholding on hue and value channels

Filtering binary watermap (remove_blob):

improc.remove_blob(directory,improcdir,prepdir,nwater=blob_removal_Water,nland=blob_removal_land,png=procblobRemovalpngFlag) function filters the binary water map based on blob size in both land and water

Shoreline extraction (extract_shoreline):

improc.extract_shoreline(directory,improcdir,prepdir) function extracts the shoreline of the binary watermap and saves the latitude and longitude of the shore line

4.2.3. Utilities (utils) :

This module holds the classes and functions used internally to read geotiffs,write data,plot data and give information about the data.

4.3 Program Output:

4.3.1. Preprocessing output:

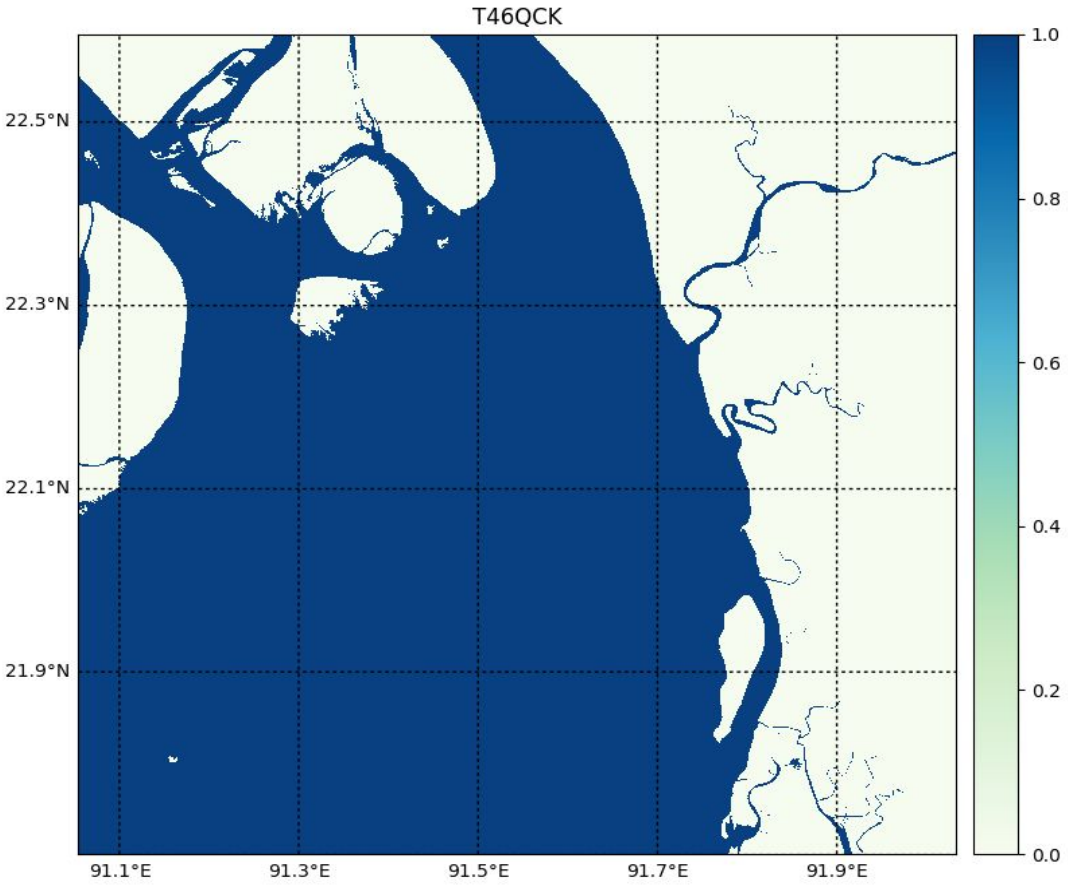
Extracting data:

Approximate time to extract data from a single zip file: 30s

Creating Water Mask:

Approximate time to process a single B11 data for creating water mask: 15s (A single zone may contain 10+ B11 data)

Example water mask for zone T46QCK:



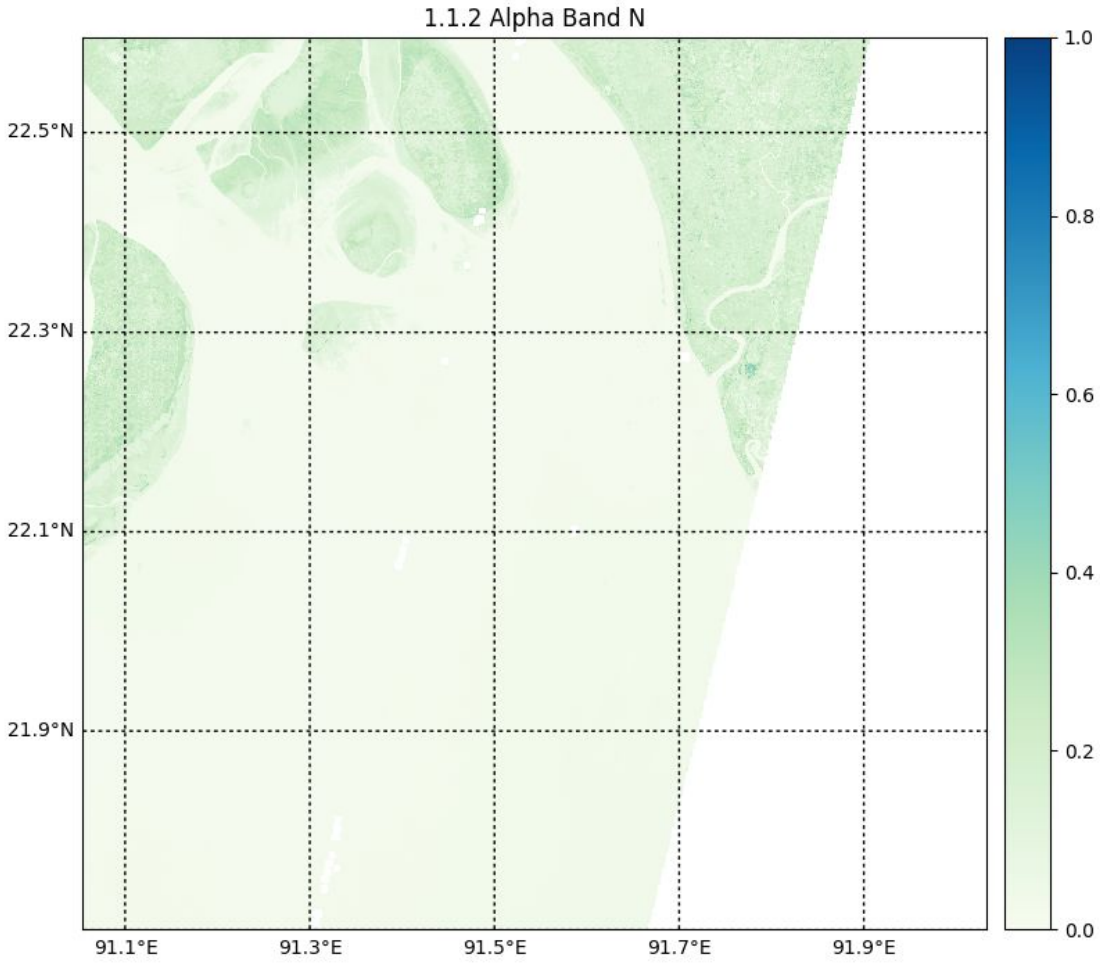
4.3.2. Image processing output:

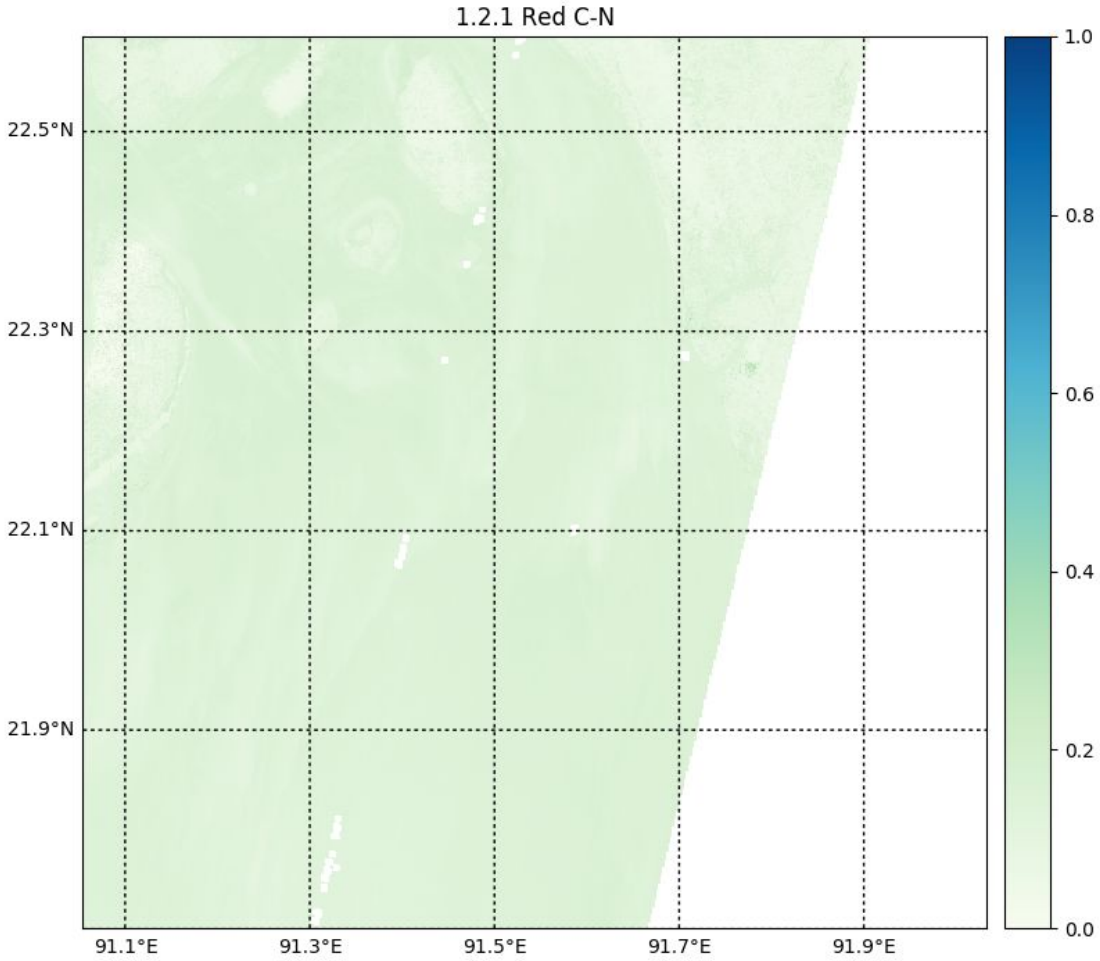
Profile of a single processed data (approximation) :

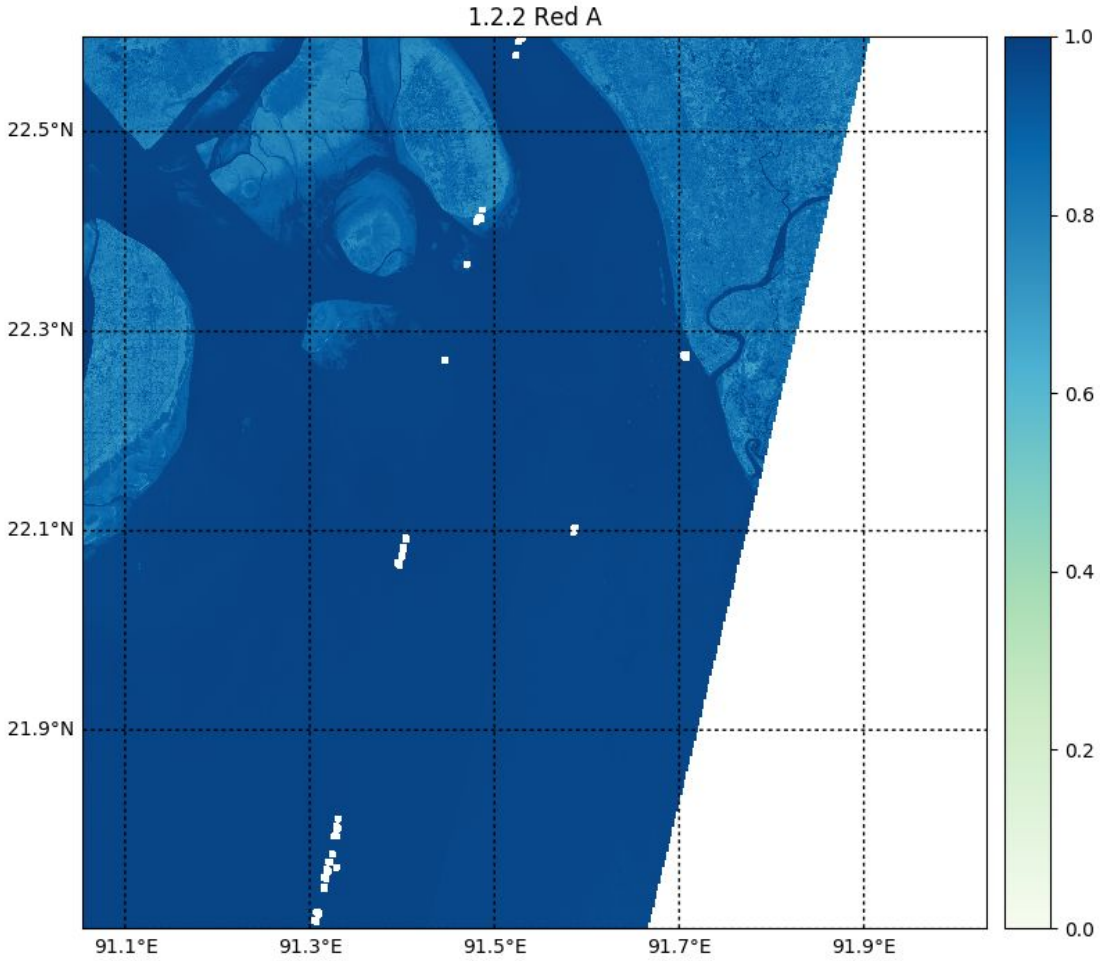
```
Total time: 112.787 s (with normal plotting)
            691.338 s (with basemap plotting)
            49.20   s (without any png plotting)
```

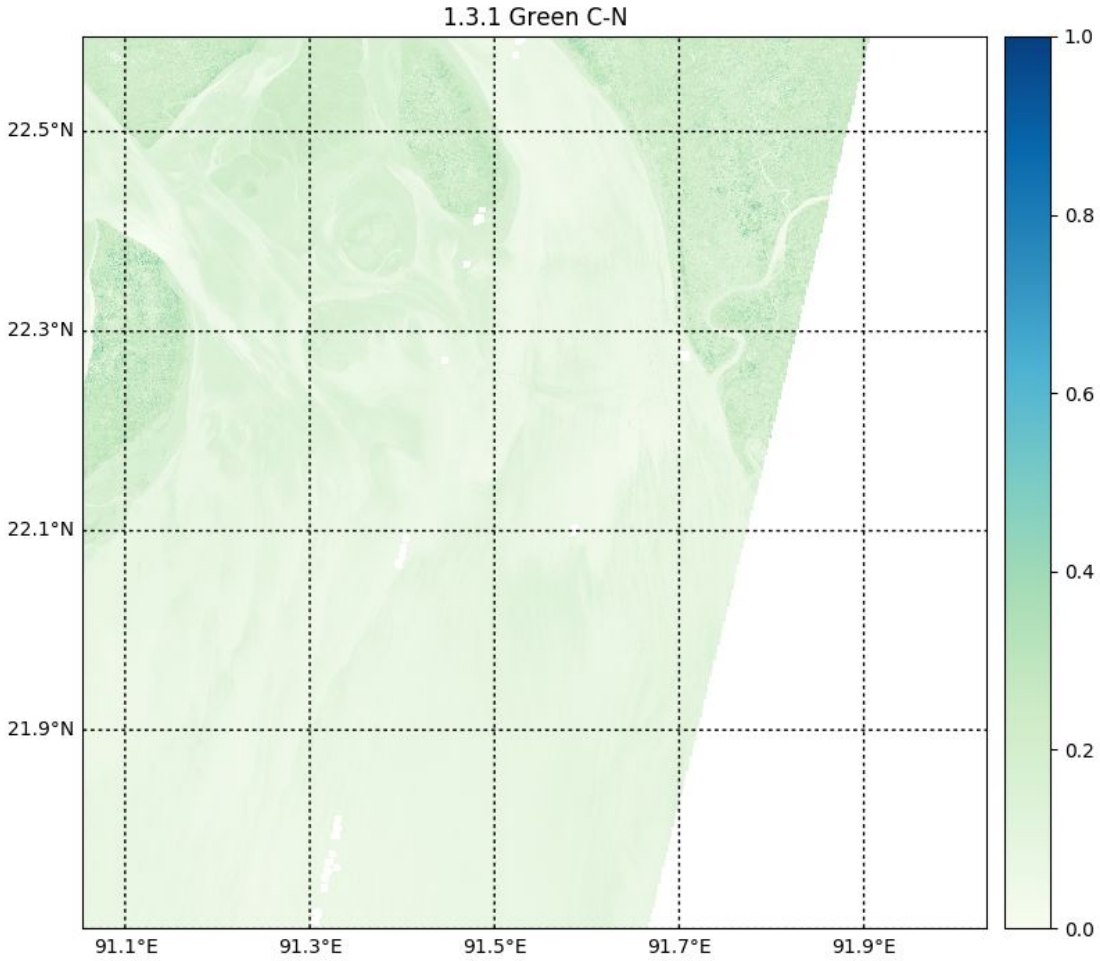
Example of a step by step processing output:

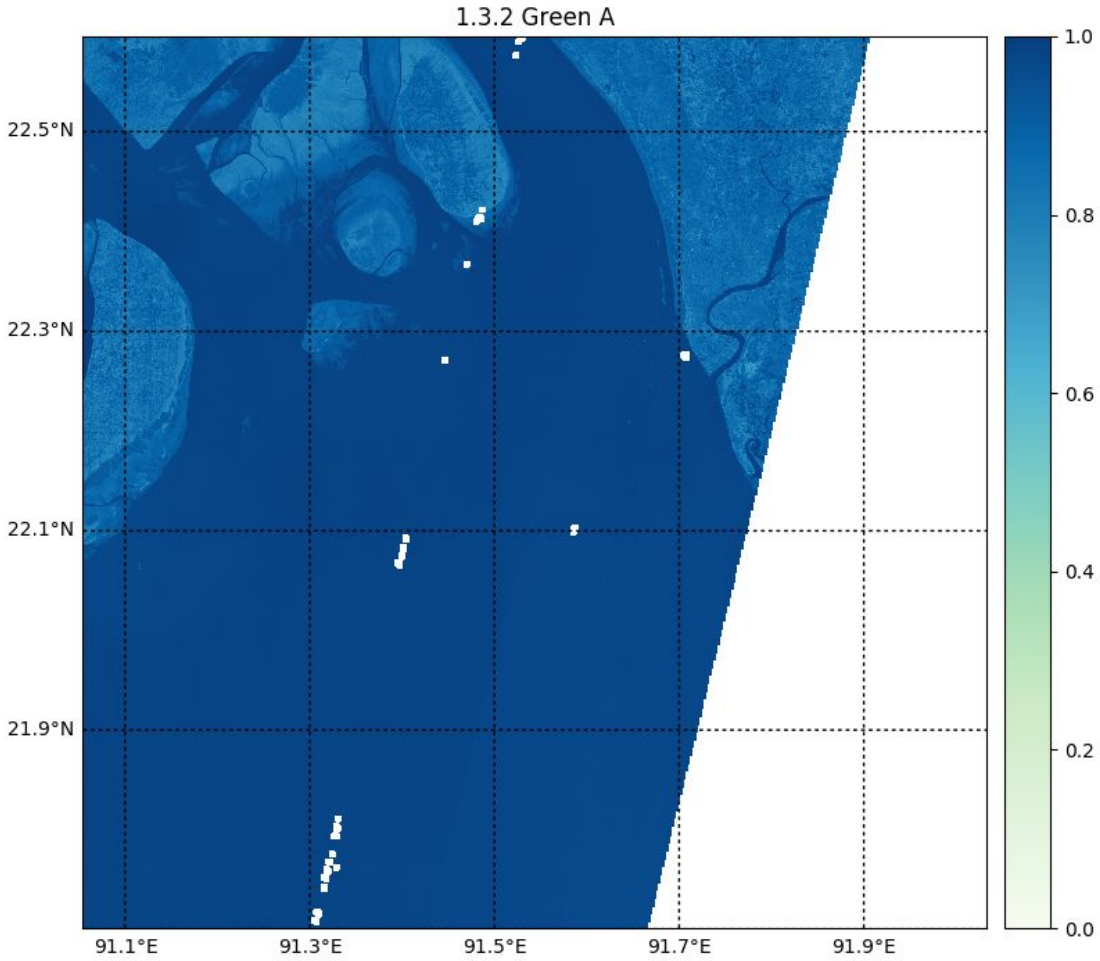
Alpha Band = B11 Cloud mask corrected and Upsampled
(N = Normalized,C= Cloud Mask Corrected)
Channel Construction:

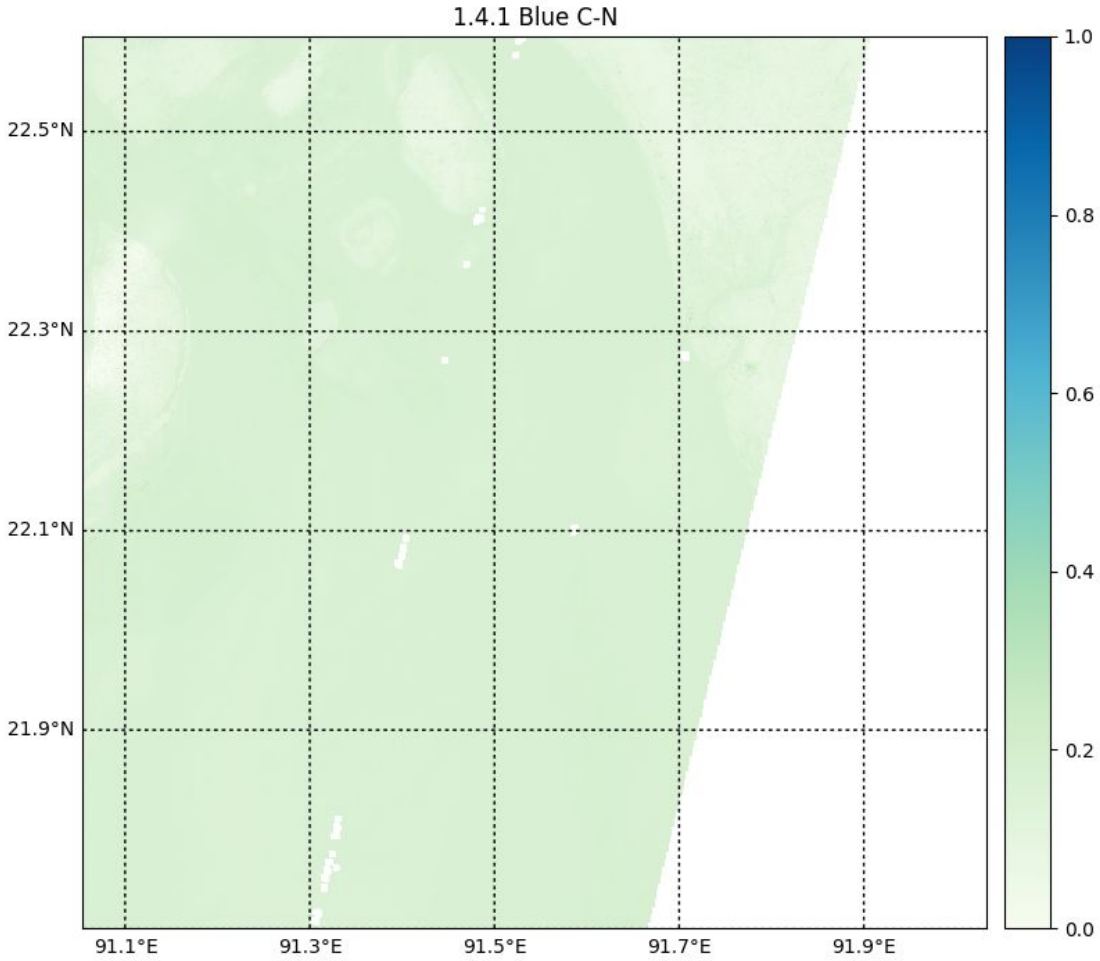


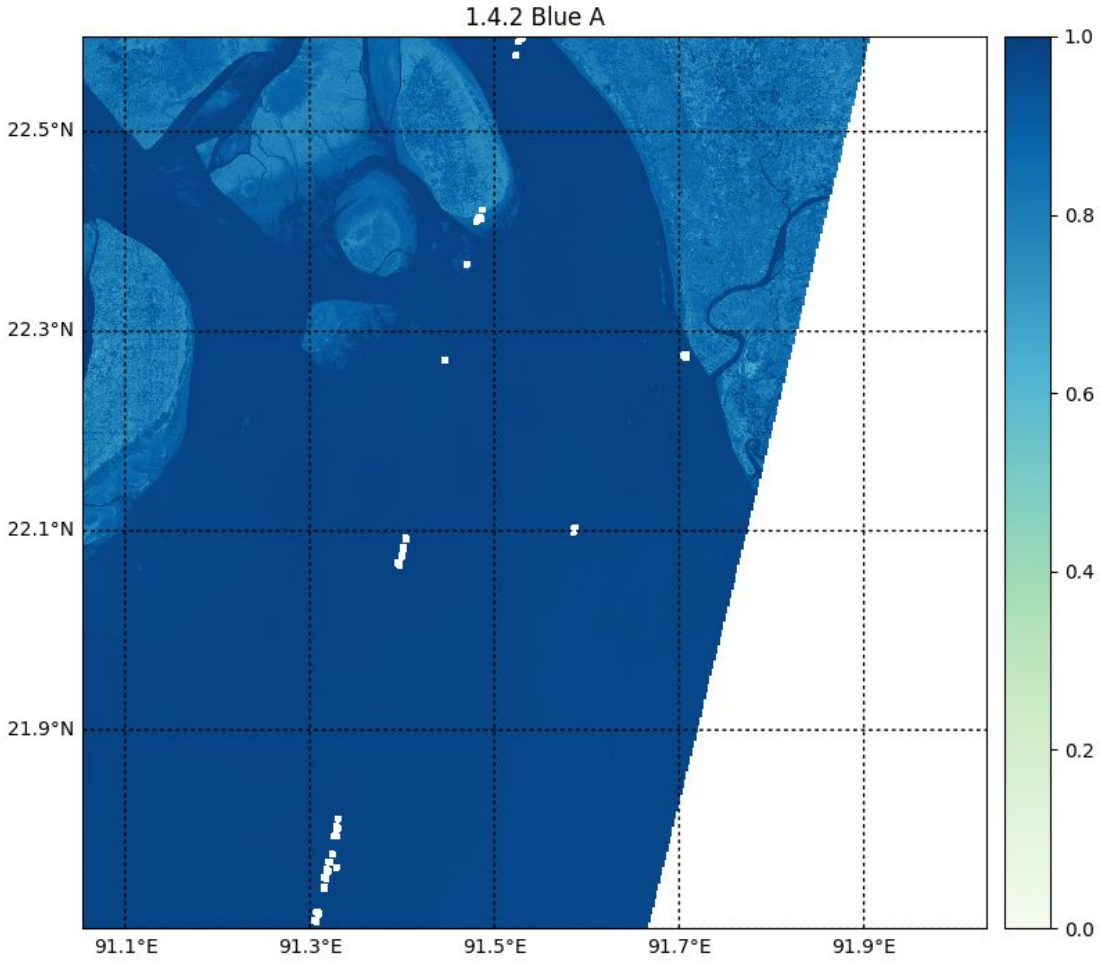

1.2.1 Red C-N
22.5°N
22.3°N
22.1°N
21.9°N
91.1°E
91.3°E
91.5°E
91.7°E
91.9°E
1.0
0.8
0.6
0.4
0.2
0.0

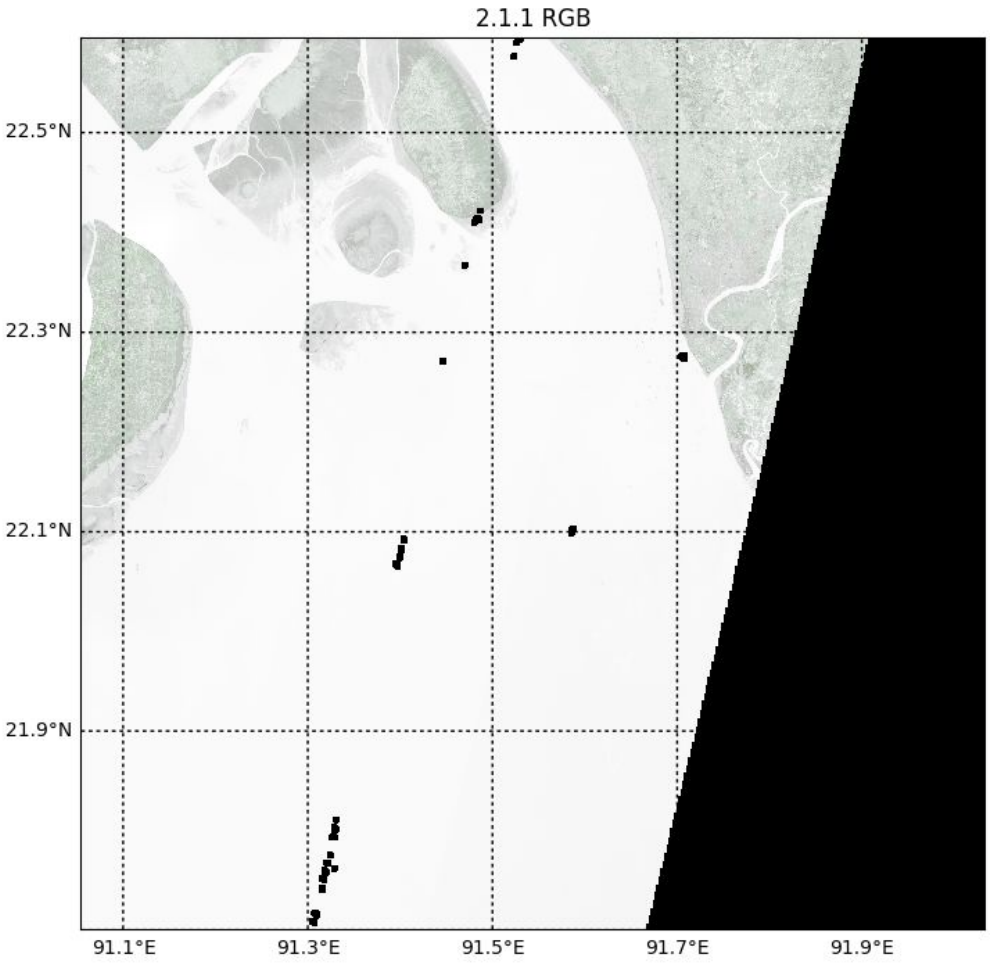

1.2.2 Red A
22.5°N
22.3°N
22.1°N
21.9°N
91.1°E
91.3°E
91.5°E
91.7°E
91.9°E
1.0
0.8
0.6
0.4
0.2
0.0

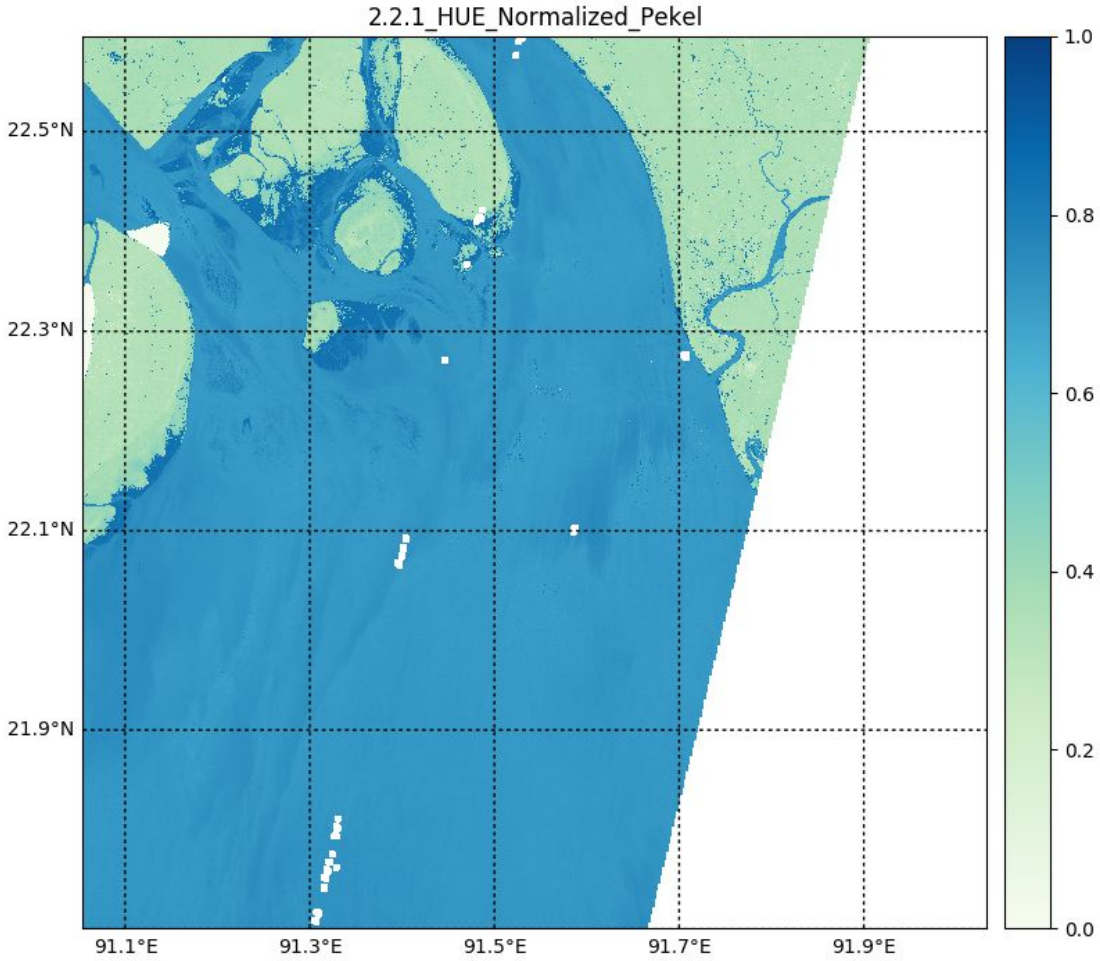

1.3.1 Green C-N
22.5°N
22.3°N
22.1°N
21.9°N
91.1°E
91.3°E
91.5°E
91.7°E
91.9°E
1.0
0.8
0.6
0.4
0.2
0.0

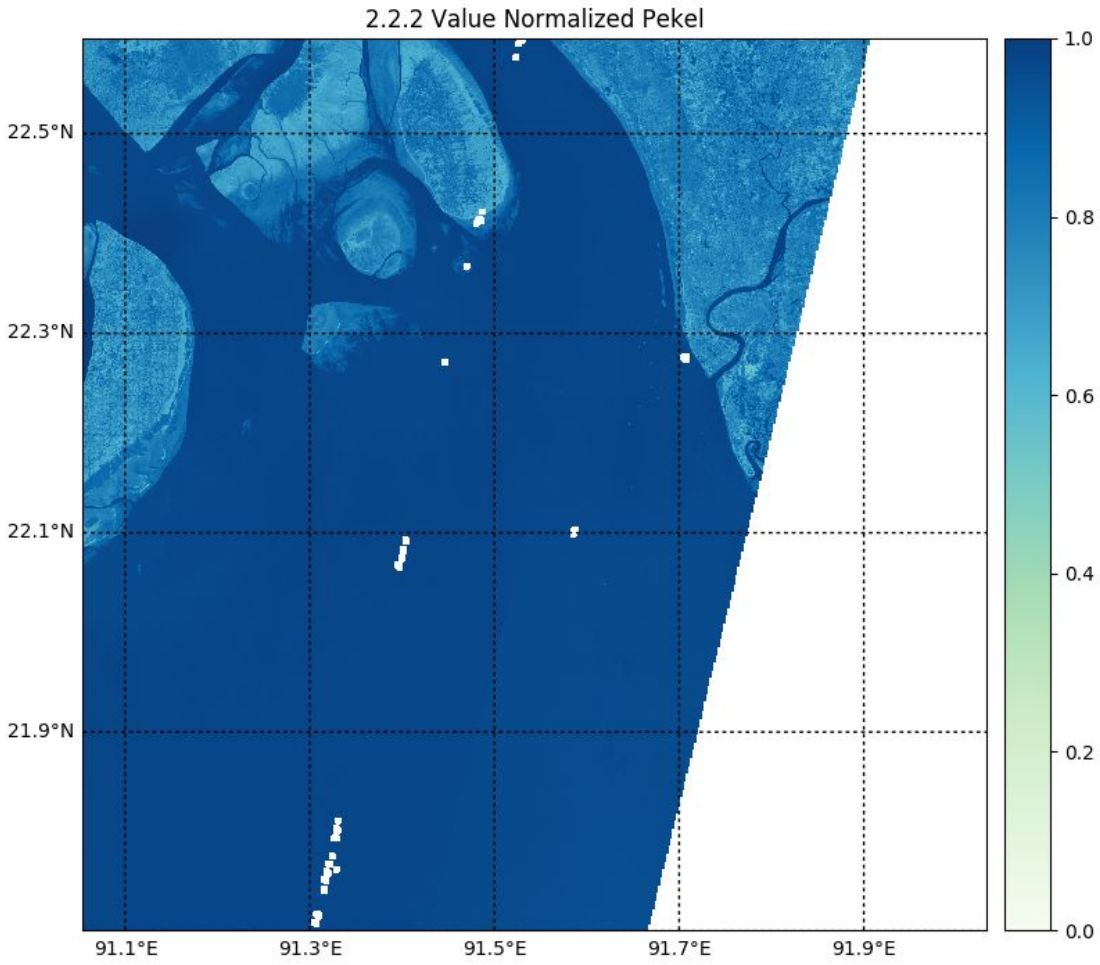

1.3.2 Green A
22.5°N
22.3°N
22.1°N
21.9°N
91.1°E
91.3°E
91.5°E
91.7°E
91.9°E
1.0
0.8
0.6
0.4
0.2
0.0


1.4.1 Blue C-N
22.5°N
22.3°N
22.1°N
21.9°N
91.1°E
91.3°E
91.5°E
91.7°E
91.9°E
1.0
0.8
0.6
0.4
0.2
0.0

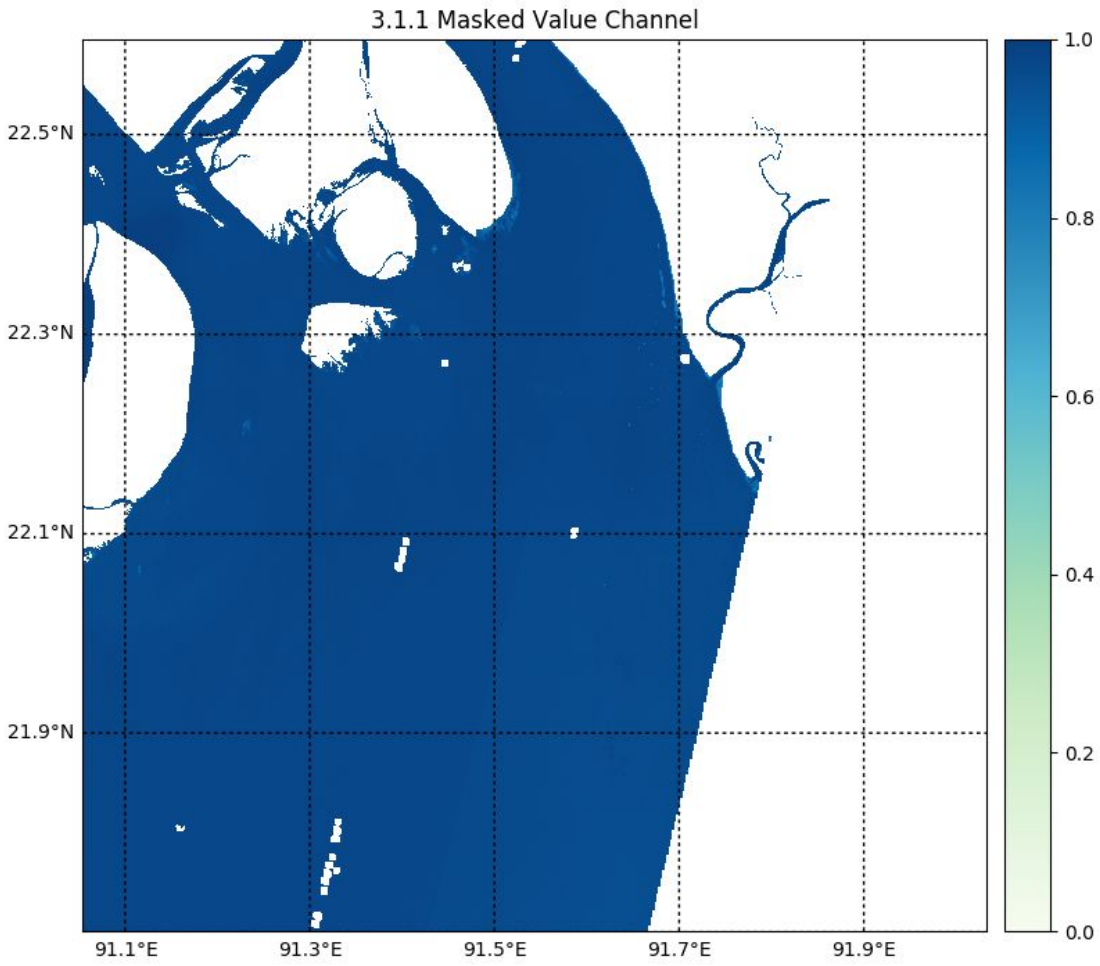

1.4.2 Blue A
22.5°N
22.3°N
22.1°N
21.9°N
91.1°E
91.3°E
91.5°E
91.7°E
91.9°E
1.0
0.8
0.6
0.4
0.2
0.0

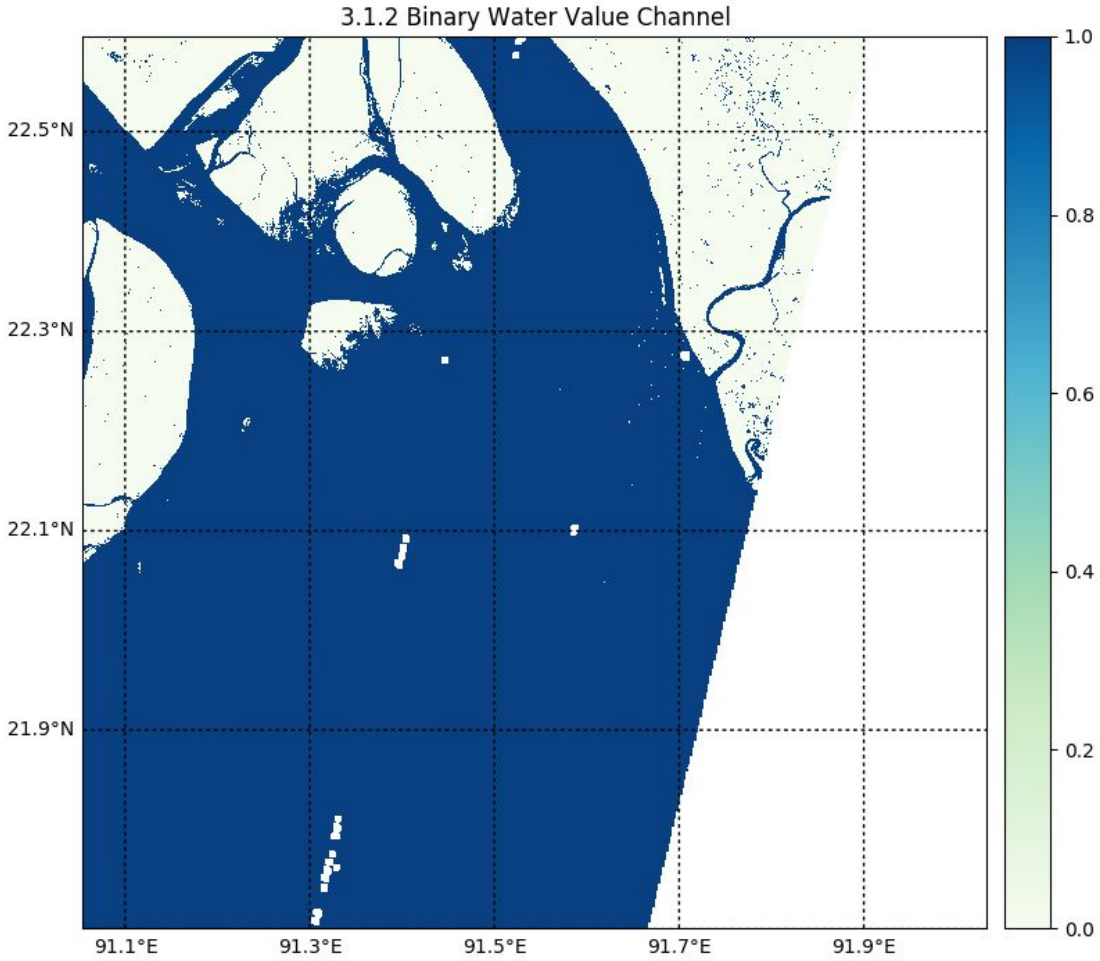

2.1.1 RGB
22.5°N
22.3°N
22.1°N
21.9°N
91.1°E
91.3°E
91.5°E
91.7°E
91.9°E

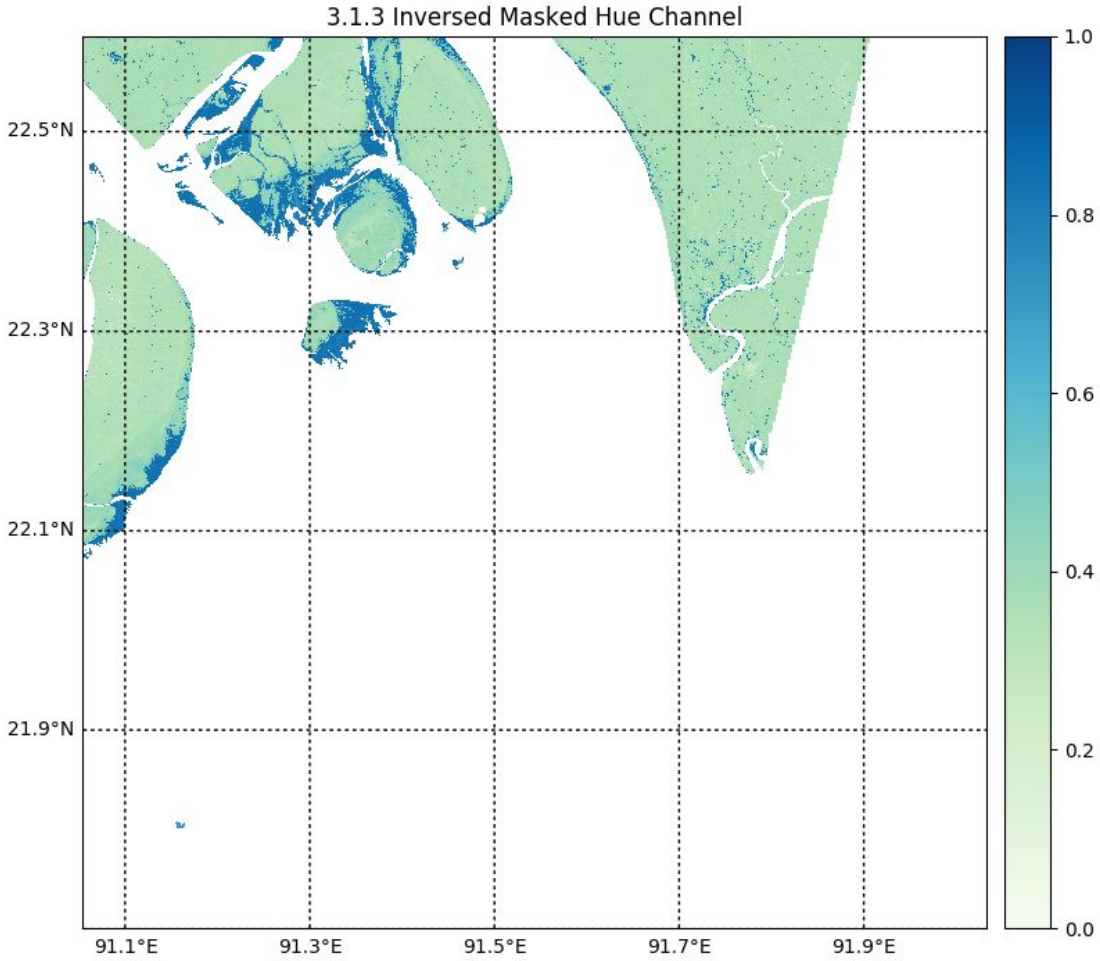

2.2.1_HUE_Normalized_Pekel
22.5°N
22.3°N
22.1°N
21.9°N
91.1°E
91.3°E
91.5°E
91.7°E
91.9°E
1.0
0.8
0.6
0.4
0.2
0.0

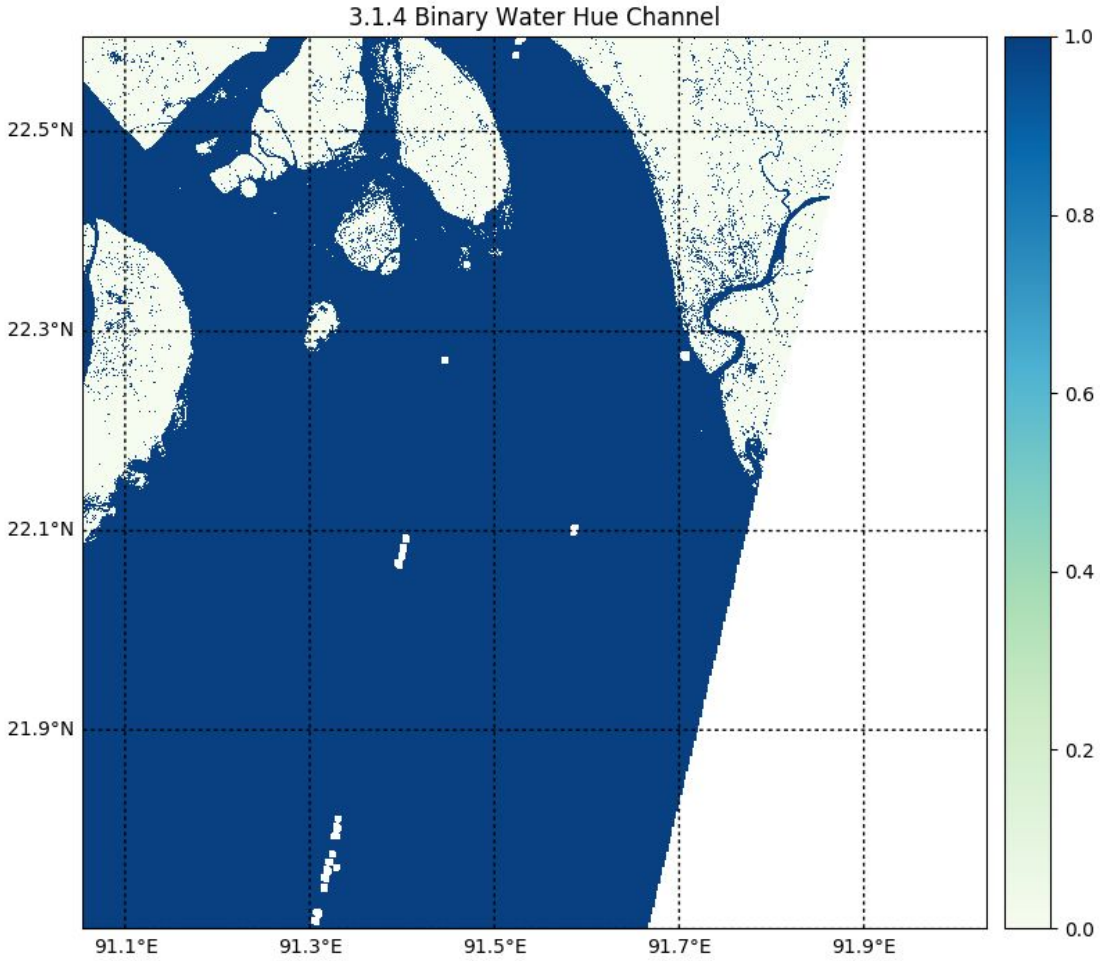

2.2.2 Value Normalized Pekel
22.5°N
22.3°N
22.1°N
21.9°N
91.1°E
91.3°E
91.5°E
91.7°E
91.9°E
1.0
0.8
0.6
0.4
0.2
0.0

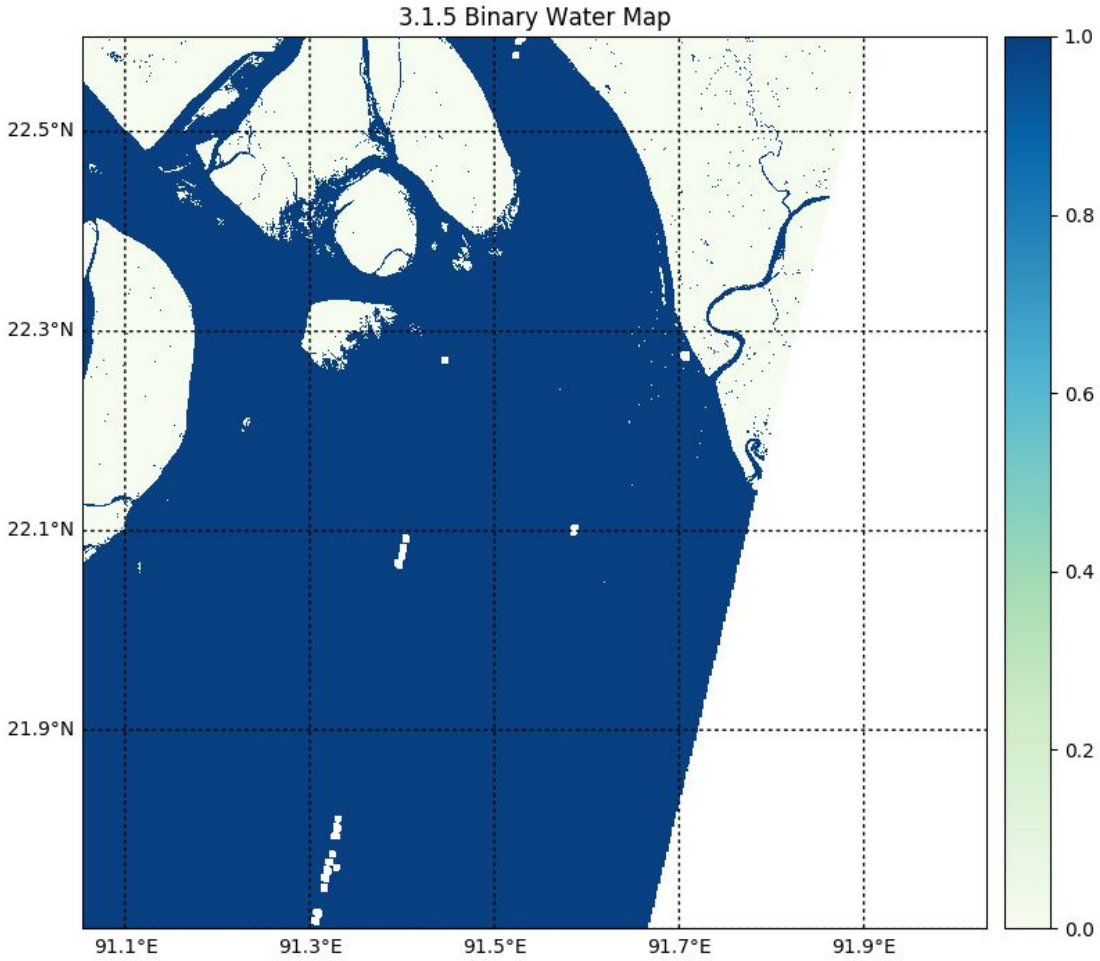
Binary Water Map Construction:



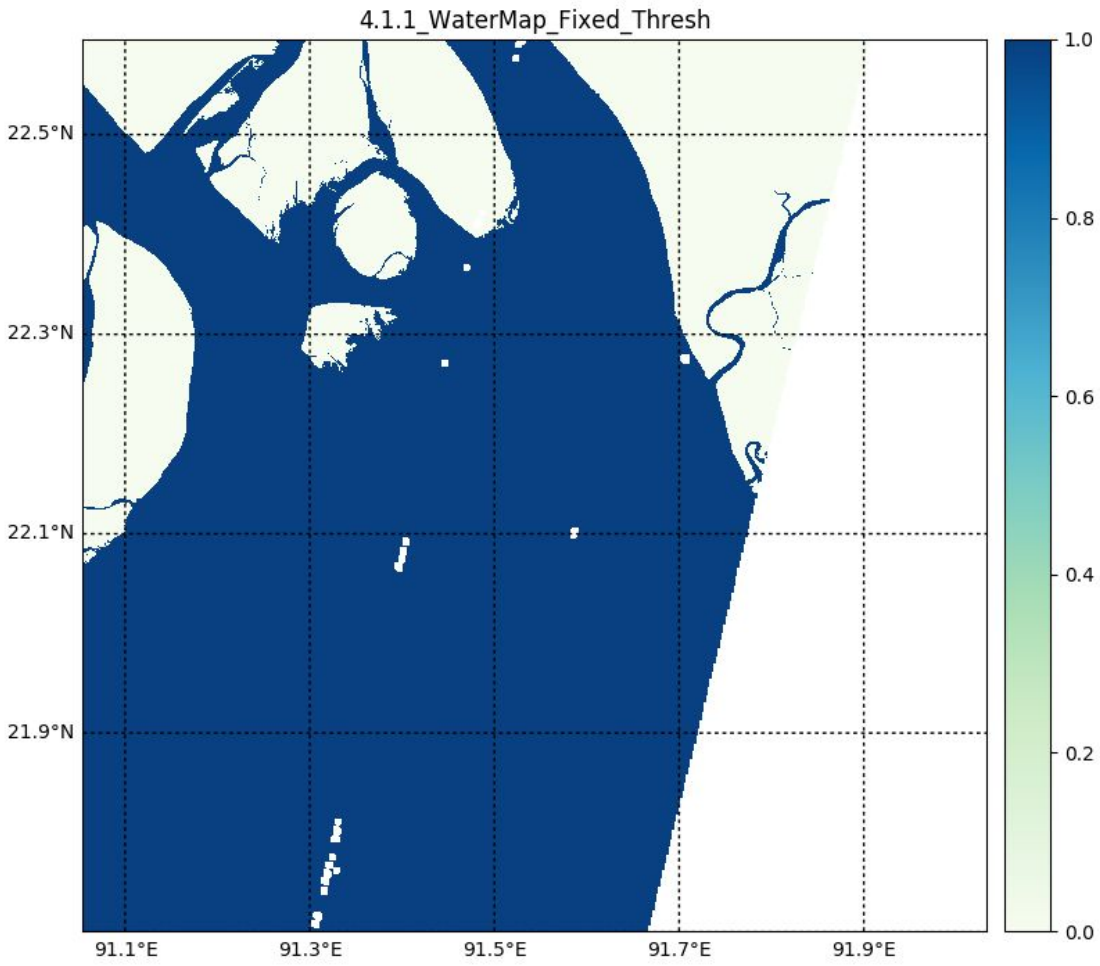

3.1.2 Binary Water Value Channel
22.5°N
22.3°N
22.1°N
21.9°N
91.1°E
91.3°E
91.5°E
91.7°E
91.9°E
1.0
0.8
0.6
0.4
0.2
0.0


3.1.3 Inversed Masked Hue Channel
22.5°N
22.3°N
22.1°N
21.9°N
91.1°E
91.3°E
91.5°E
91.7°E
91.9°E
1.0
0.8
0.6
0.4
0.2
0.0


3.1.4 Binary Water Hue Channel
22.5°N
22.3°N
22.1°N
21.9°N
91.1°E
91.3°E
91.5°E
91.7°E
91.9°E
1.0
0.8
0.6
0.4
0.2
0.0


3.1.5 Binary Water Map
22.5°N
22.3°N
22.1°N
21.9°N
91.1°E
91.3°E
91.5°E
91.7°E
91.9°E
1.0
0.8
0.6
0.4
0.2
0.0

Blob Removal:



```
Mem usage    Increment   Line Contents
================================================
146.8 MiB      0.0 MiB   Starting of memory profiling
955.6 MiB    808.7 MiB   improc.construct_channels
827.4 MiB   -128.2 MiB   improc.make_watermap
827.6 MiB      0.2 MiB   improc.remove_blob
828.1 MiB      0.5 MiB   improc.extract_shoreline
828.1 MiB      0.0 MiB   Ending of memory profiling
```

Test PC Config:
--OS: Ubuntu 18.04 LTS/ Bionic Beaver
--Processor: Intel® Core™ i7-7700 CPU @ 3.60GHz × 8
--Ram: 24 GB (DDR 4)

The processing time can increase depending on location of data (i.e- if the data is processed from/in a removable hard drive it can take longer due to saving of necessary tiff saving. These intermediate tiff are necessary to avoid memory error to occur in runtime)

Changing the default optional values might also result in changing of processing time. For example while filtering choosing very low values as blob size for removal will take longer as significant blob number increases with decrease of size.