Name: Om Mahesh Vaknalli

Roll No. 18376

Subject: EES-338 (Remote Sensing & GIS Lab)

Lab – I (Introduction)

For USGS data download:

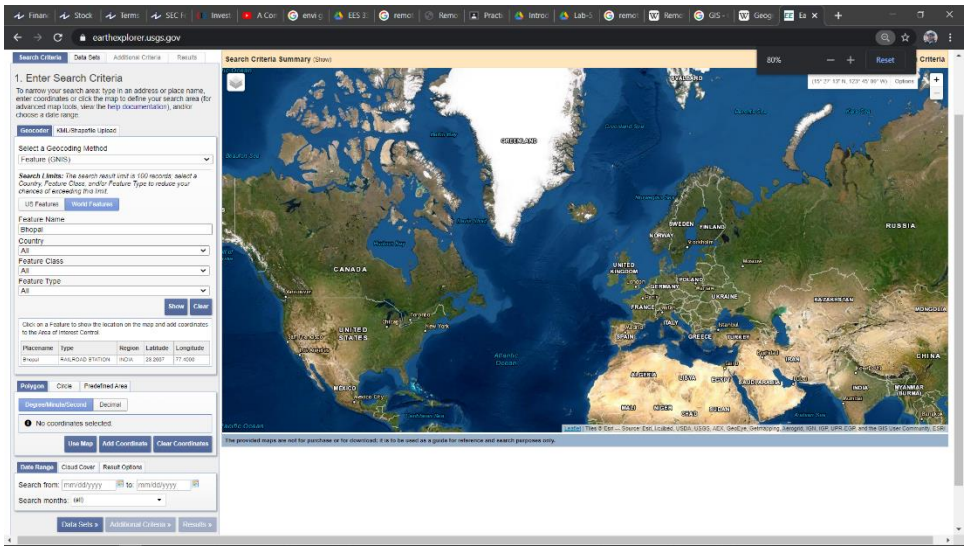


Fig. 1: Login into USGS Earth Explorer website, enter the search criteria

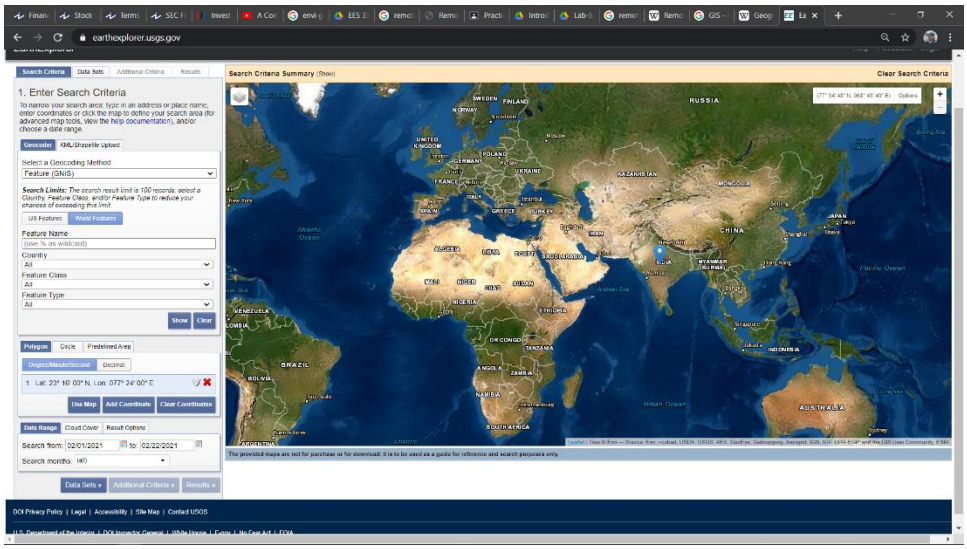


Fig. 2: Add in the search date range

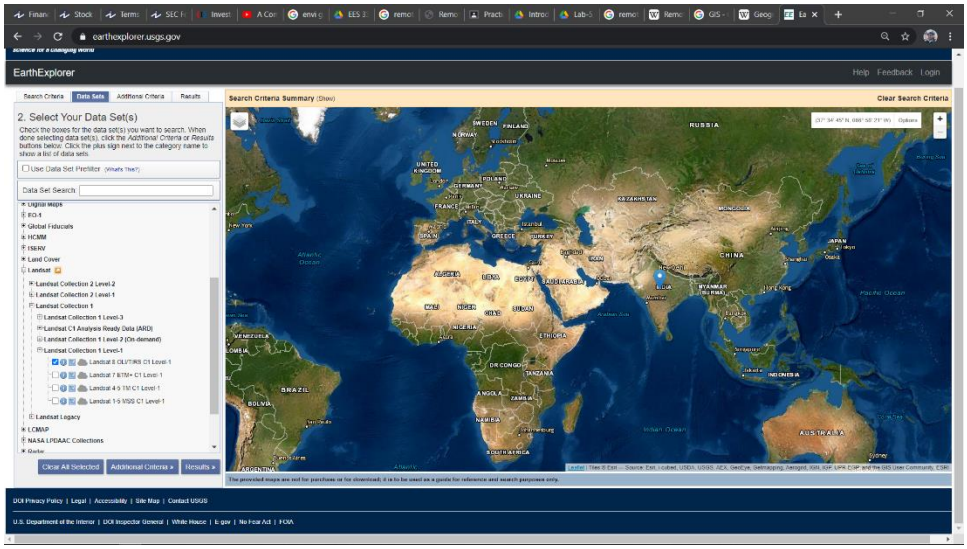


Fig. 3: Choose the required satellite as well as the band of choice

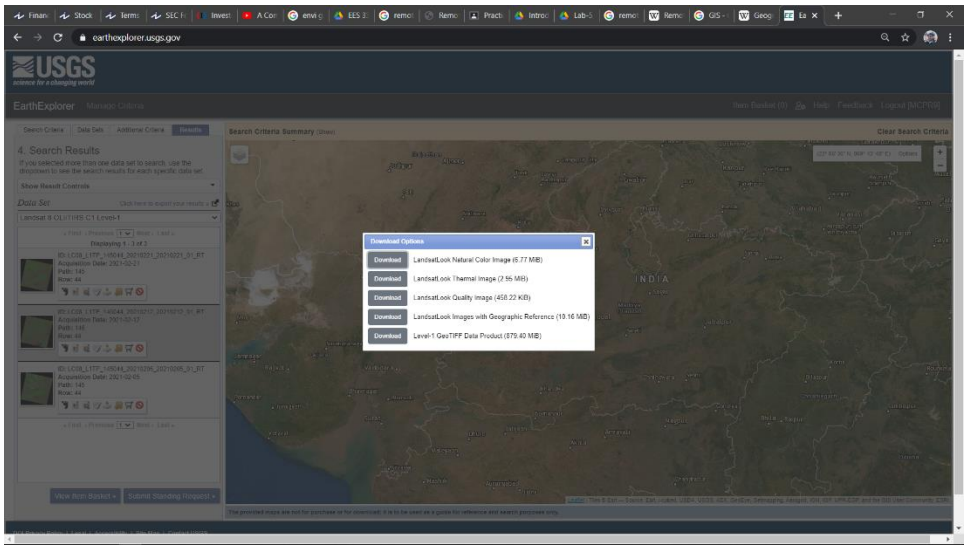


Fig. 4: Download the GeoTIFF file of the selected band image

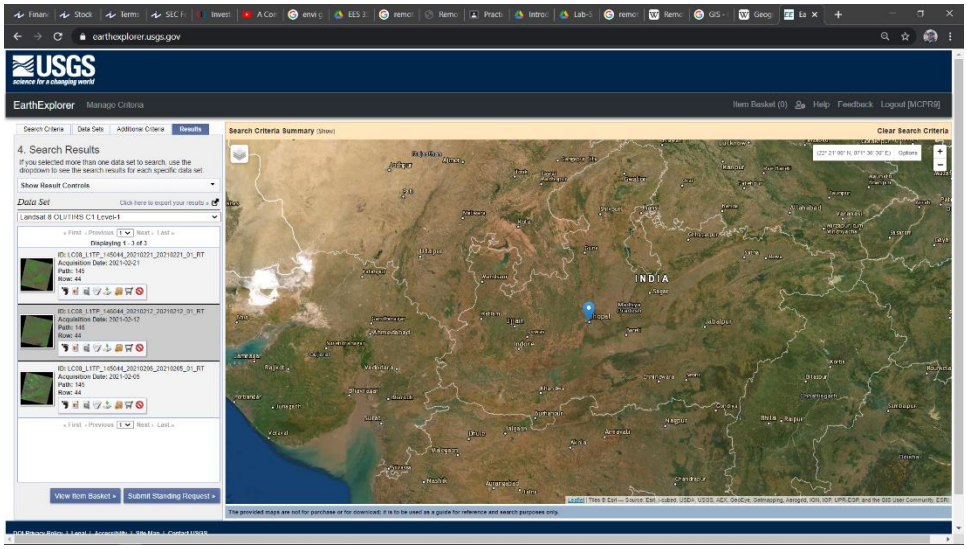


Fig. 5: Results

For Bhuvan data download:

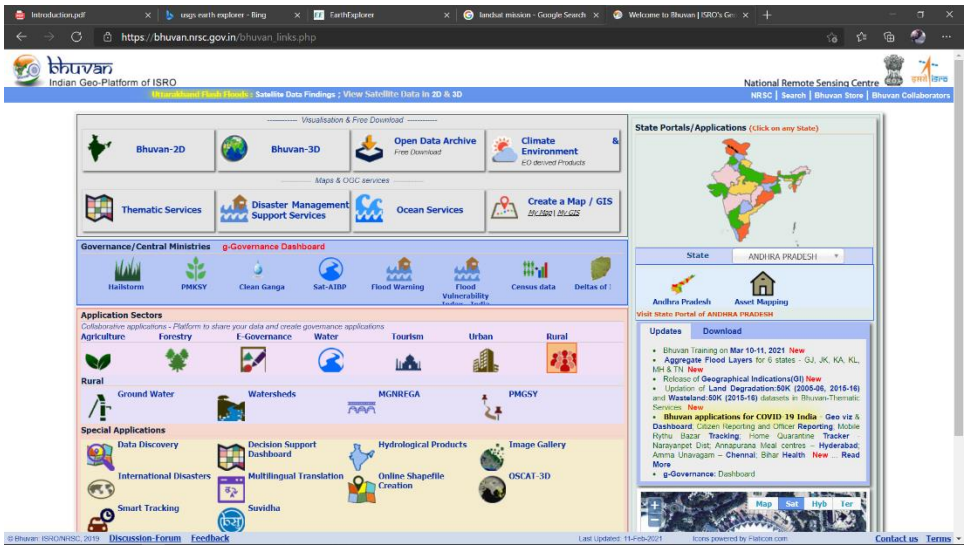


Fig. 6: **Login into Bhuvan Indian Geo-Platform**

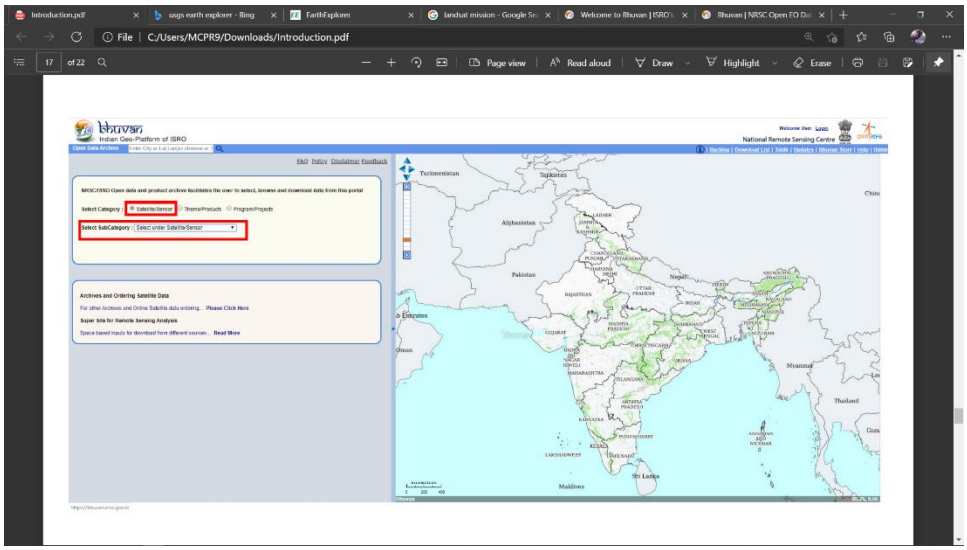


Fig. 7: **Select desired satellite and sensor**

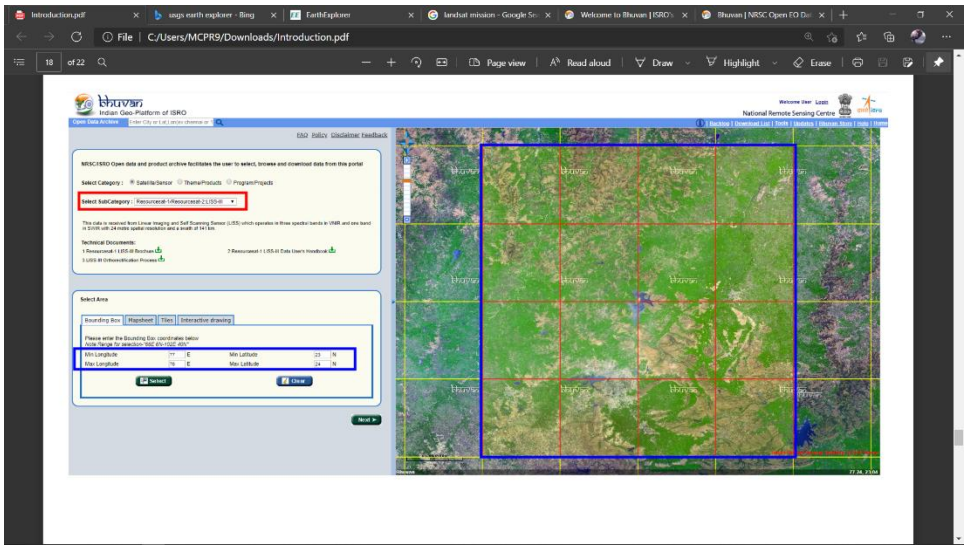


Fig. 8: **Enter the details of the Area of Interest**

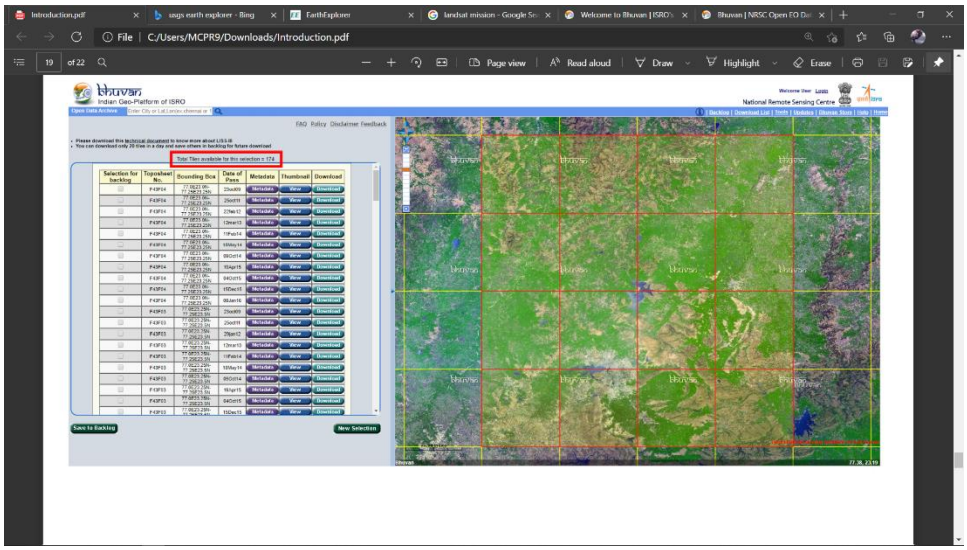


Fig. 9: **Select the date of interest**

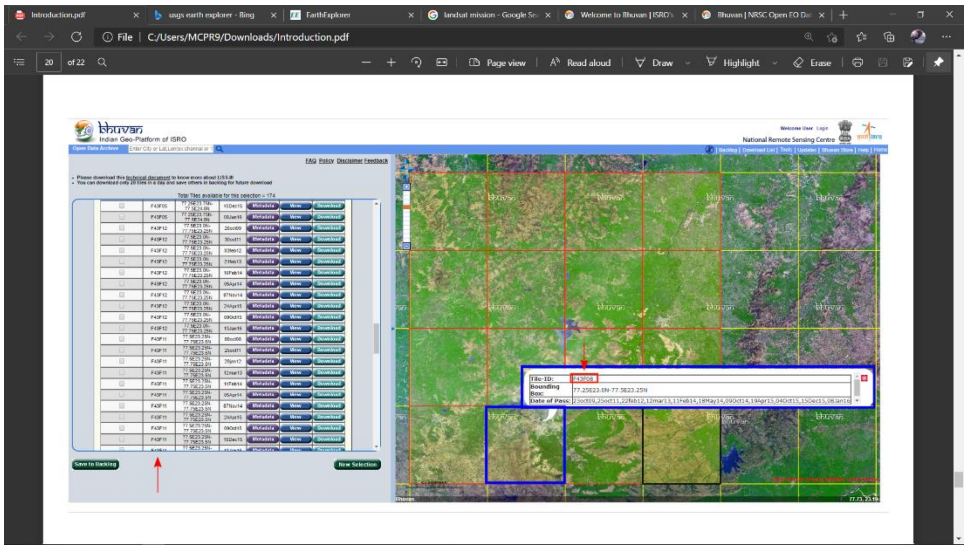


Fig. 10: **Results**

Lab – II (Image Visualization in ArcGIS)

Fig. 1: [Landsat Missions and their properties]

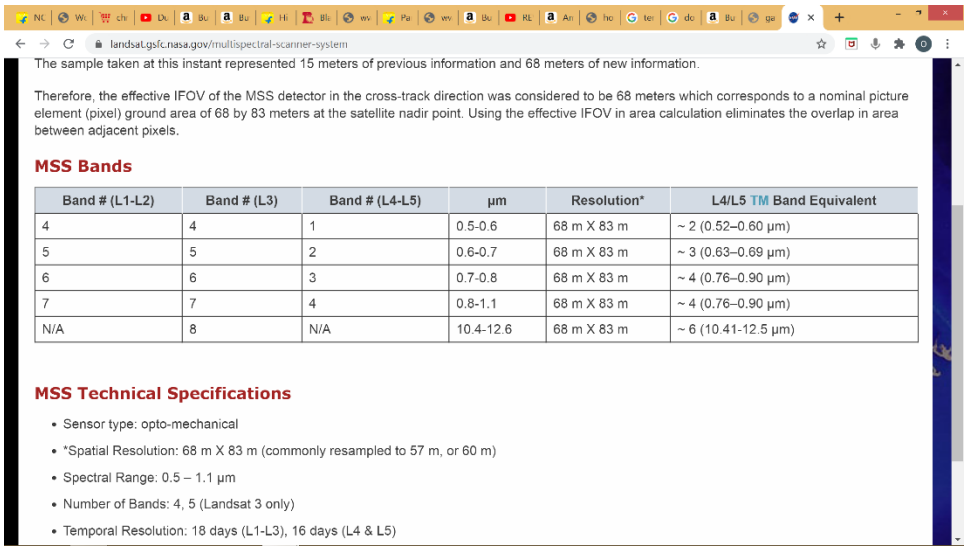
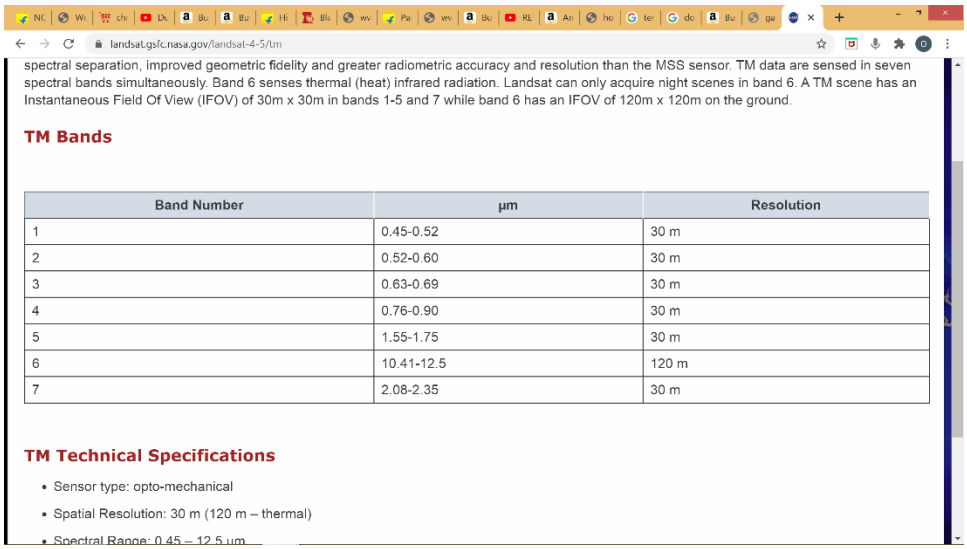


Fig. 2: **MSS Bands Spectrum**



landsat.gsfc.nasa.gov/landsat-4-5/tm

spectral separation, improved geometric fidelity and greater radiometric accuracy and resolution than the MSS sensor. TM data are sensed in seven spectral bands simultaneously. Band 6 senses thermal (heat) infrared radiation. Landsat can only acquire night scenes in band 6. A TM scene has an Instantaneous Field Of View (IFOV) of 30m x 30m in bands 1-5 and 7 while band 6 has an IFOV of 120m x 120m on the ground.

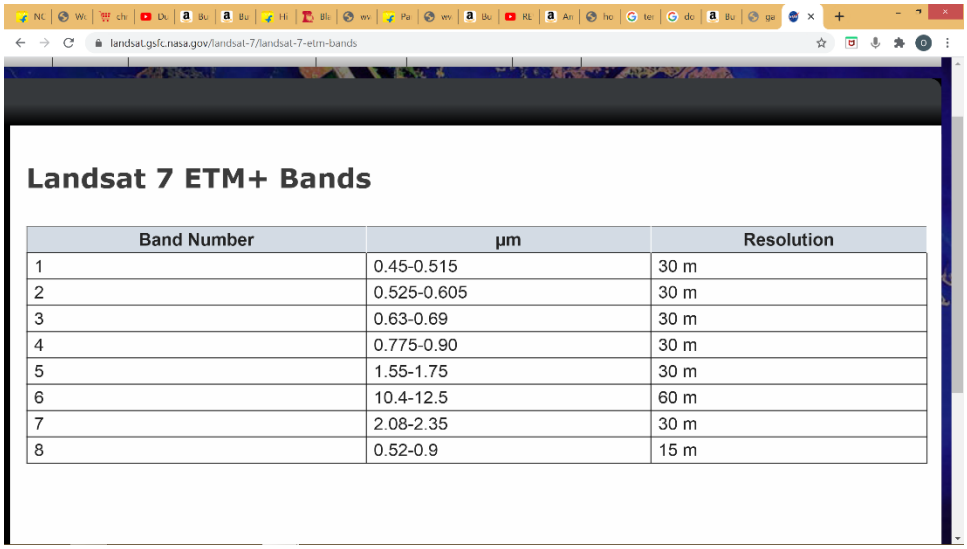
TM Bands

Band Number	µm	Resolution
1	0.45-0.52	30 m
2	0.52-0.60	30 m
3	0.63-0.69	30 m
4	0.76-0.90	30 m
5	1.55-1.75	30 m
6	10.41-12.5	120 m
7	2.08-2.35	30 m

TM Technical Specifications

- Sensor type: opto-mechanical
- Spatial Resolution: 30 m (120 m – thermal)
- Spectral Range: 0.45 – 12.5 µm

Fig. 3: **TM Bands Spectrum**

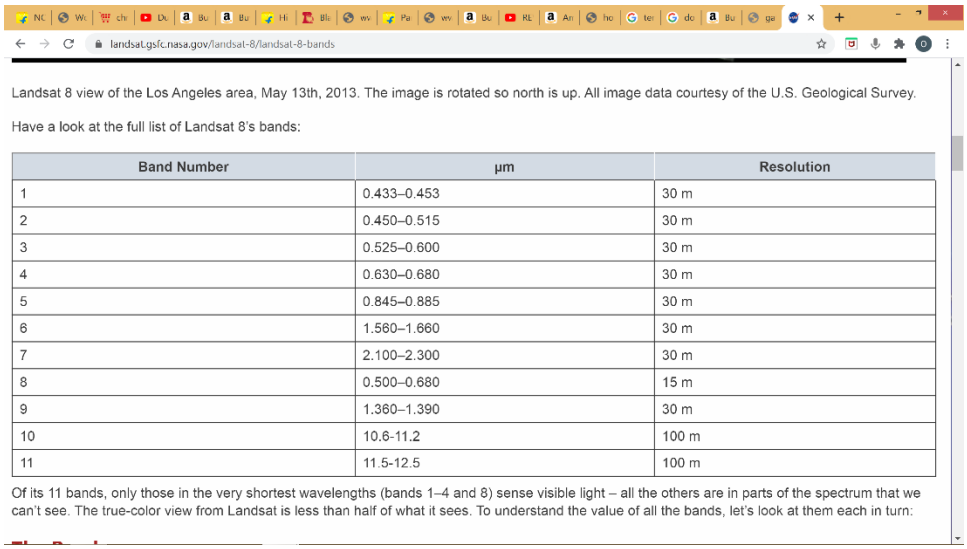


landsat.gsfc.nasa.gov/landsat-7/landsat-7-etm-bands

Landsat 7 ETM+ Bands

Band Number	µm	Resolution
1	0.45-0.515	30 m
2	0.525-0.605	30 m
3	0.63-0.69	30 m
4	0.775-0.90	30 m
5	1.55-1.75	30 m
6	10.4-12.5	60 m
7	2.08-2.35	30 m
8	0.52-0.9	15 m

Fig. 4: **ETM+ Bands Spectrum**



landsat.gsfc.nasa.gov/landsat-8/landsat-8-bands

Landsat 8 view of the Los Angeles area, May 13th, 2013. The image is rotated so north is up. All image data courtesy of the U.S. Geological Survey.

Have a look at the full list of Landsat 8's bands:

Band Number	µm	Resolution
1	0.433–0.453	30 m
2	0.450–0.515	30 m
3	0.525–0.600	30 m
4	0.630–0.680	30 m
5	0.845–0.885	30 m
6	1.560–1.660	30 m
7	2.100–2.300	30 m
8	0.500–0.680	15 m
9	1.360–1.390	30 m
10	10.6-11.2	100 m
11	11.5-12.5	100 m

Of its 11 bands, only those in the very shortest wavelengths (bands 1–4 and 8) sense visible light – all the others are in parts of the spectrum that we can't see. The true-color view from Landsat is less than half of what it sees. To understand the value of all the bands, let's look at them each in turn:

Fig. 5: **OLI & TIRS Bands Spectrum**

Band Combinations for Landsat 8

Composite Name	Bands
Natural Color	4 3 2
False Color (urban)	7 6 4
Color Infrared (vegetation)	5 4 3
Agriculture	6 5 2
Healthy Vegetation	5 6 2
Land/Water	5 6 4
Natural With Atmospheric Removal	7 5 3
Shortwave Infrared	7 5 4
Vegetation Analysis	6 5 4

Fig. 6: **Band combinations for building meaningful composite images**

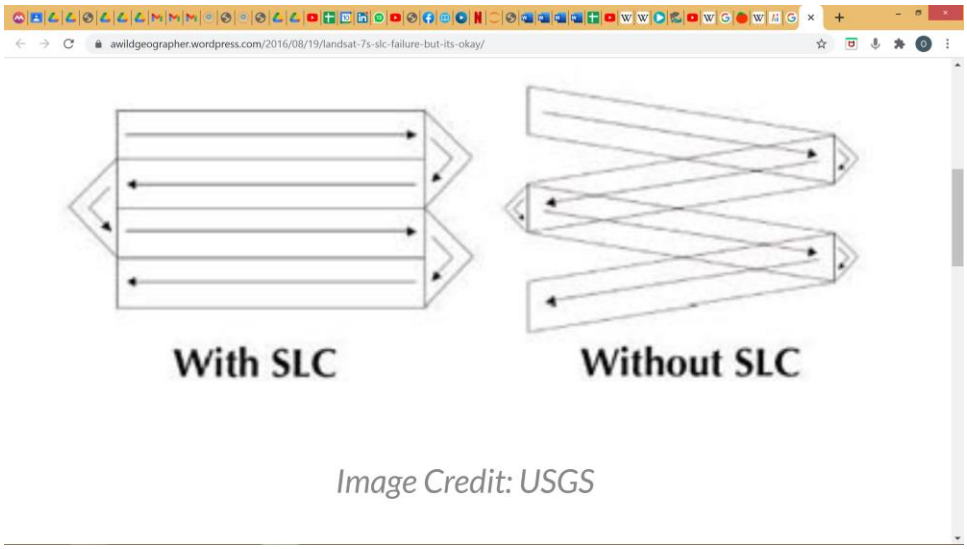


Fig. 7: **Scan-Line Compensator**

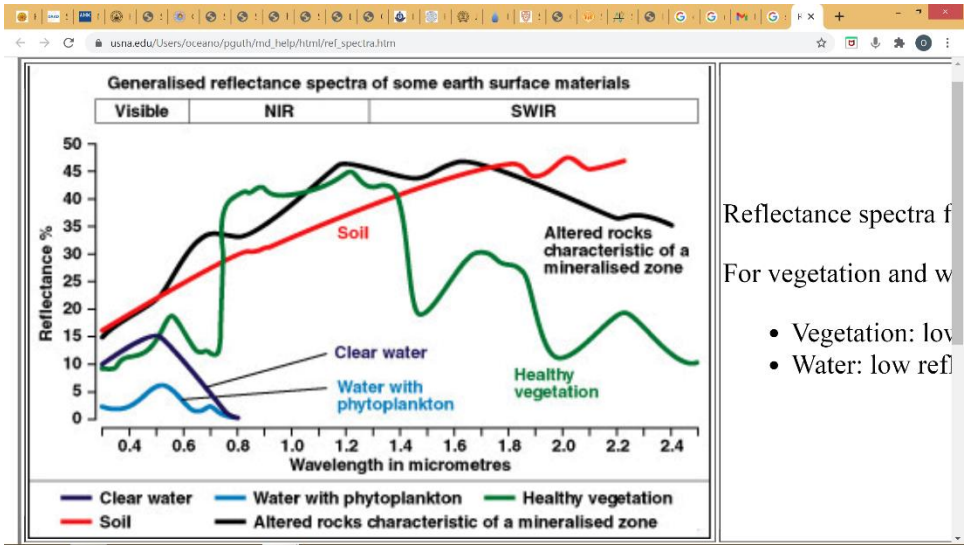


Fig. 8: **Spectral Reflectance Curve**

Finding Spatial Resolution & DN values and Locating Scan-line Error of Landsat 7:

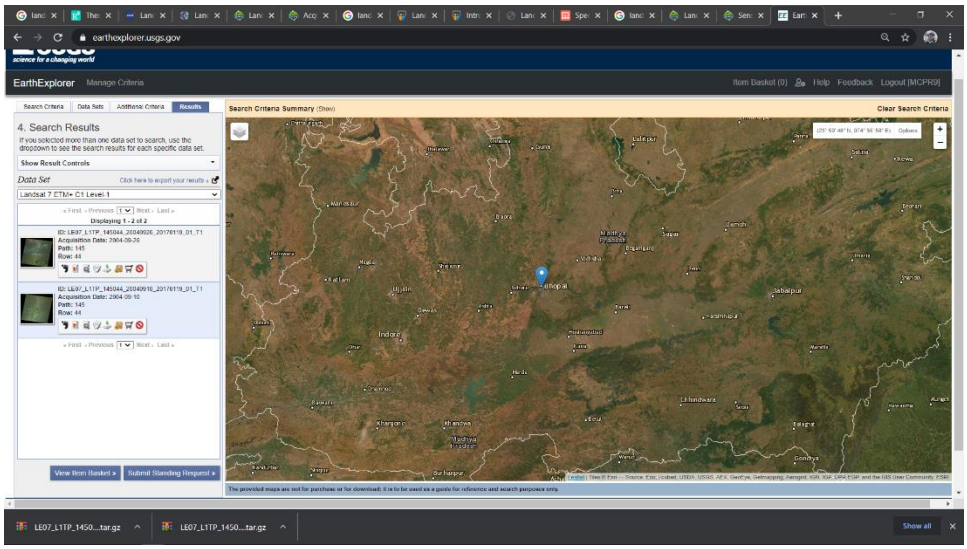


Fig. 9: Search for Landsat 7 images before and after May 2003 on USGS Earth Explorer

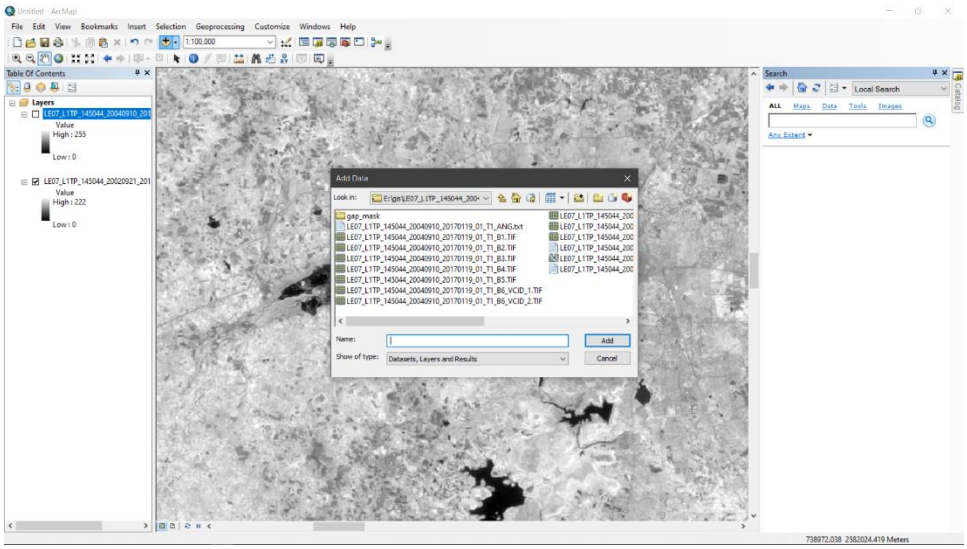


Fig. 10: Load these images on ArcGIS software

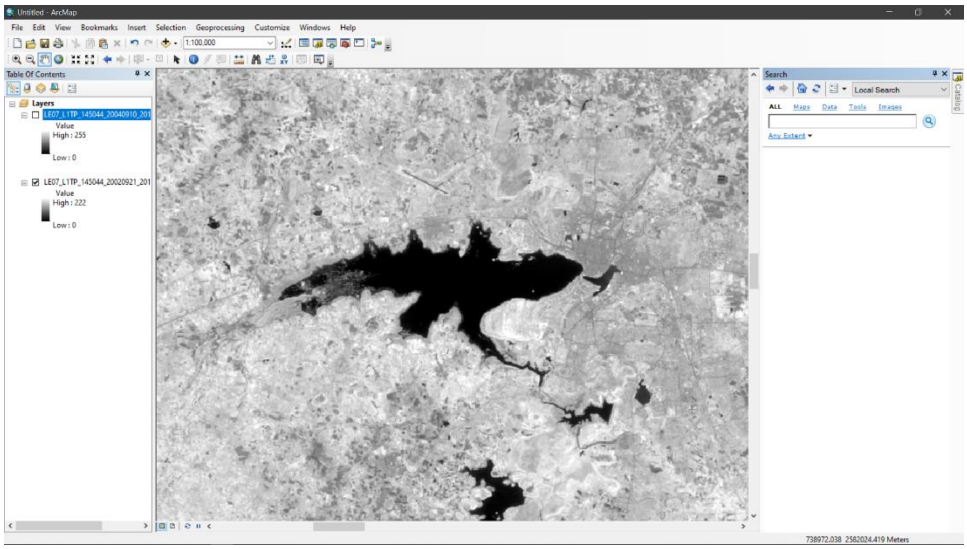


Fig. 11: Result: Pre – May 2003 image

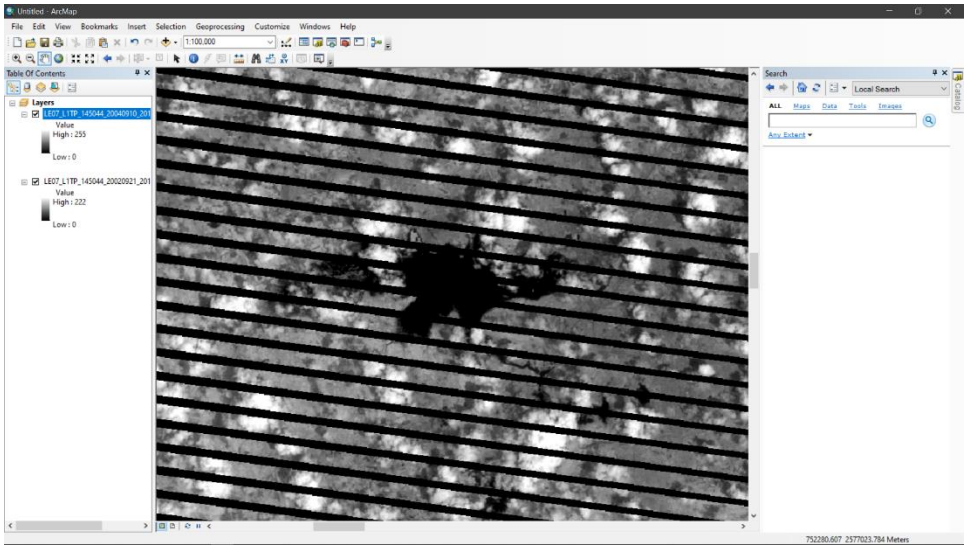


Fig. 12: Result: Post – May 2003 images

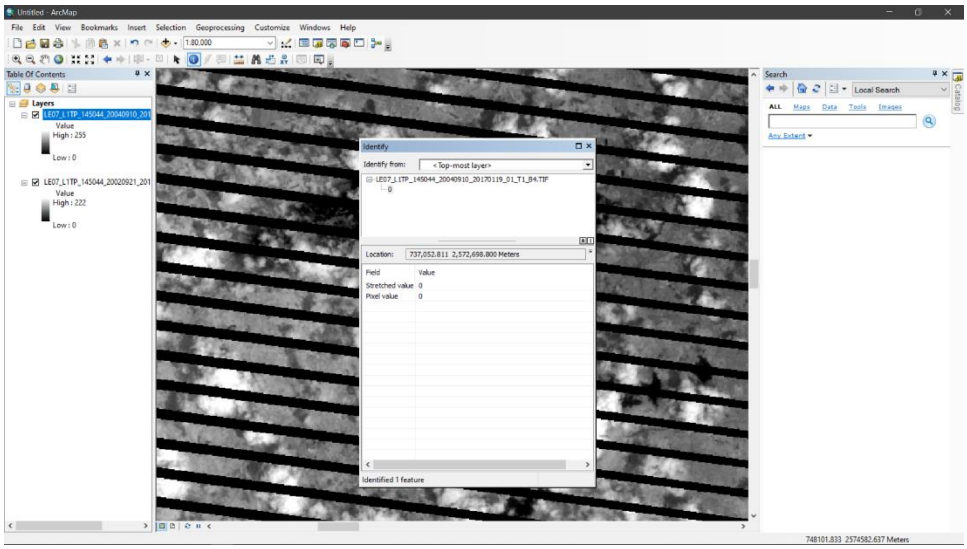


Fig. 13: **Identifying pixel values/intensity values/digital no. values of the image**

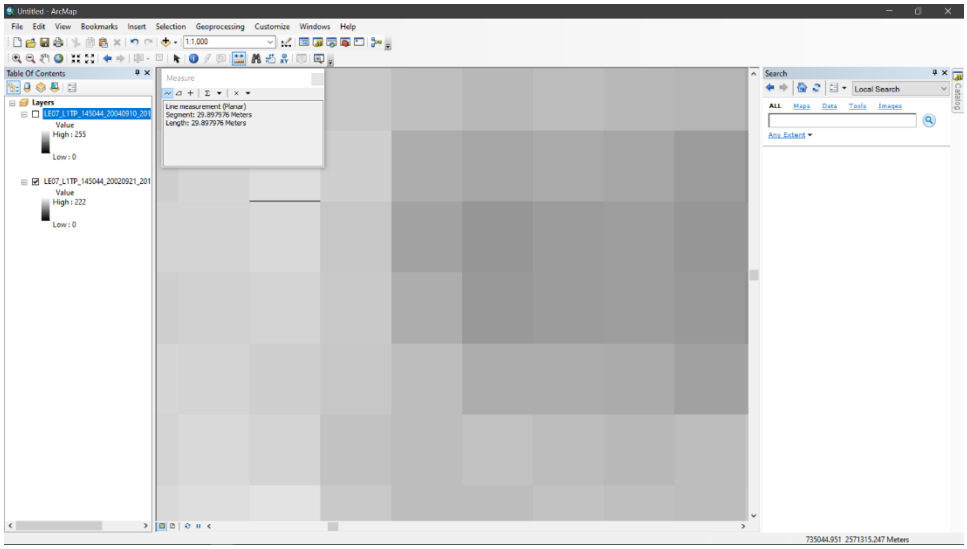


Fig. 14: **Measuring pixel size / spatial resolution**

Identifying waterbodies and vegetation in band images:

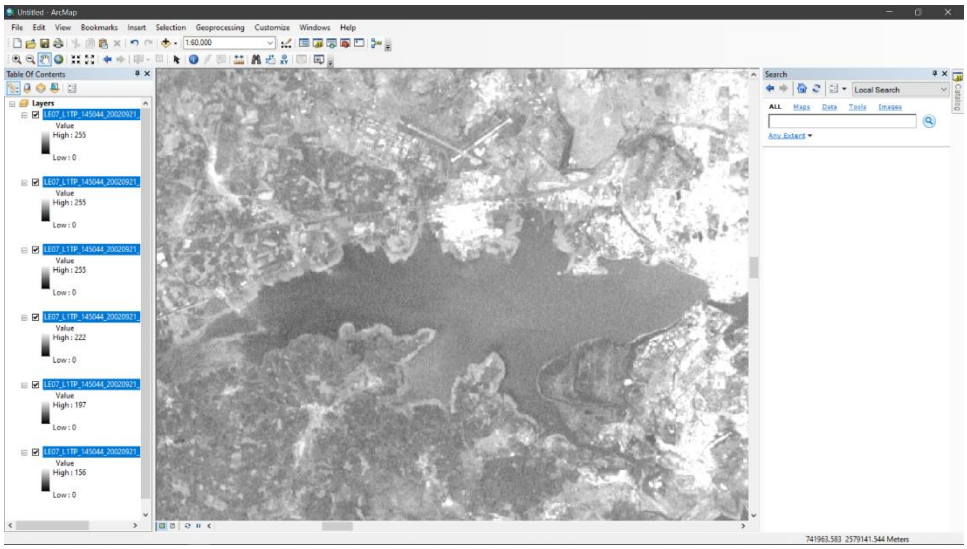


Fig. 15: **Band 1 image**

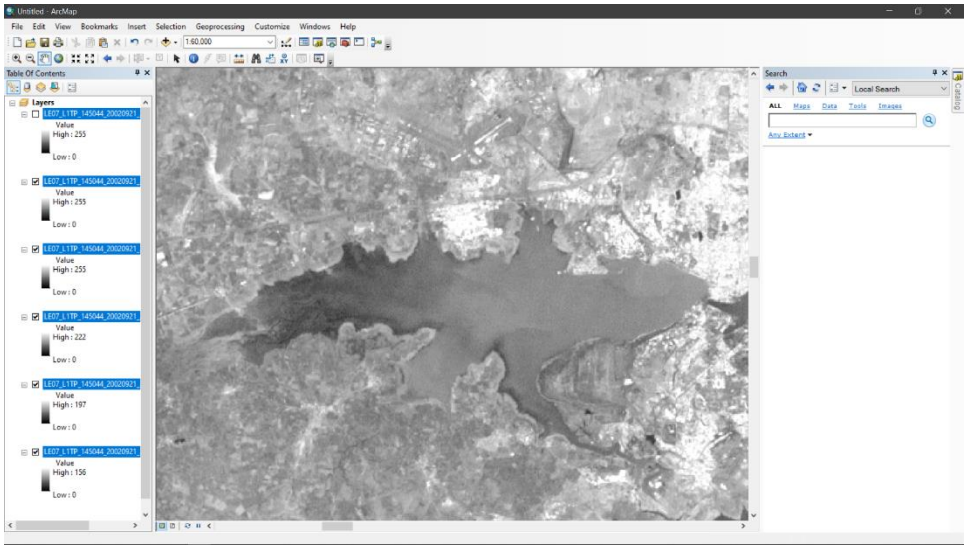


Fig. 16: **Band 2 image**

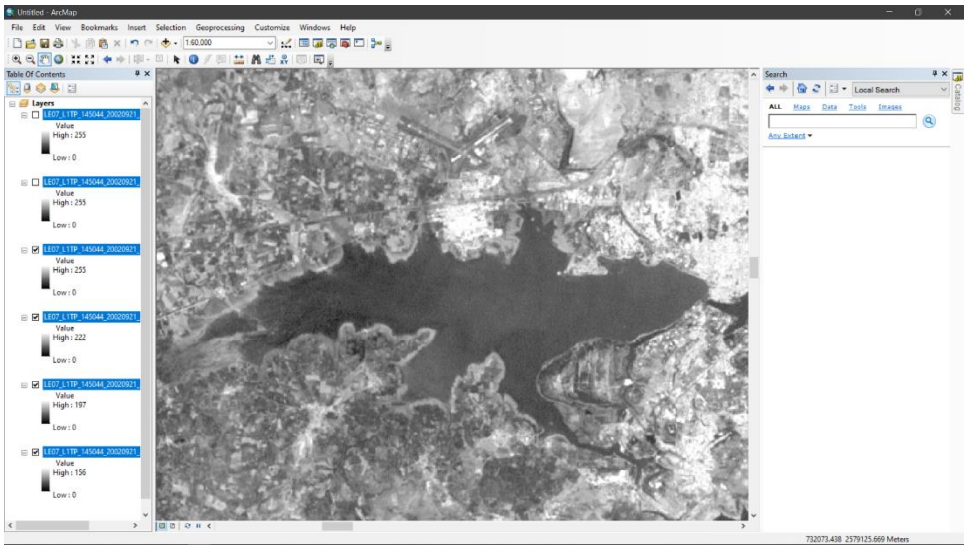


Fig. 17: **Band 3 image**

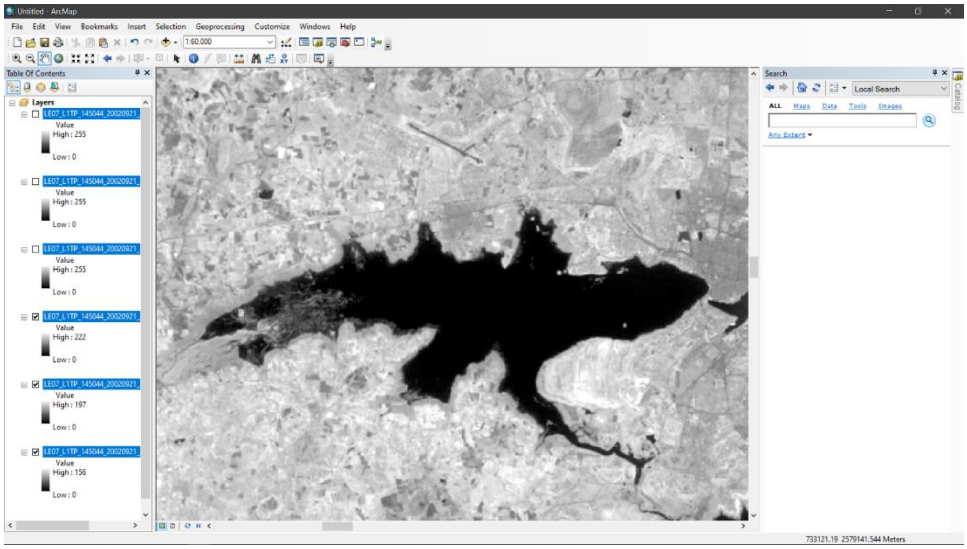


Fig. 18: **Band 4 image**

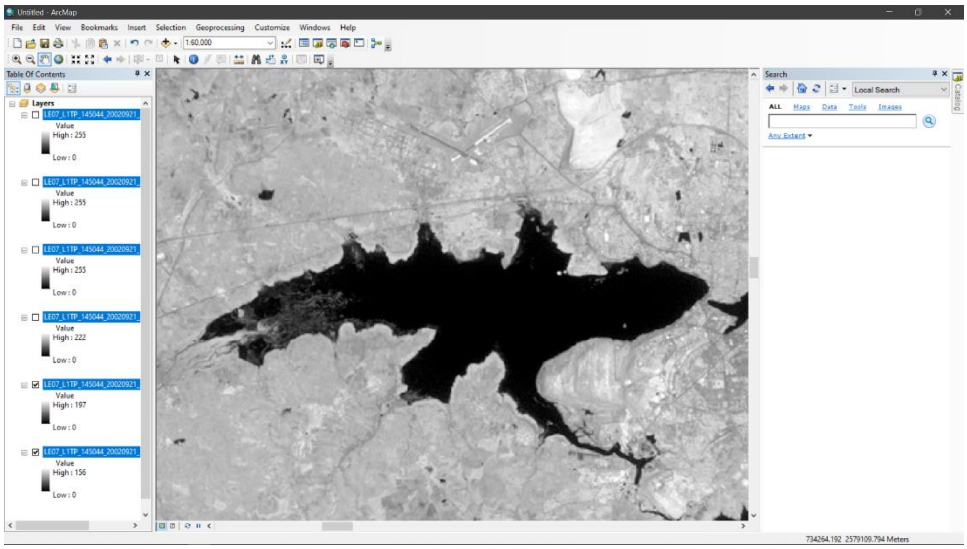


Fig. 19: **Band 5 image**

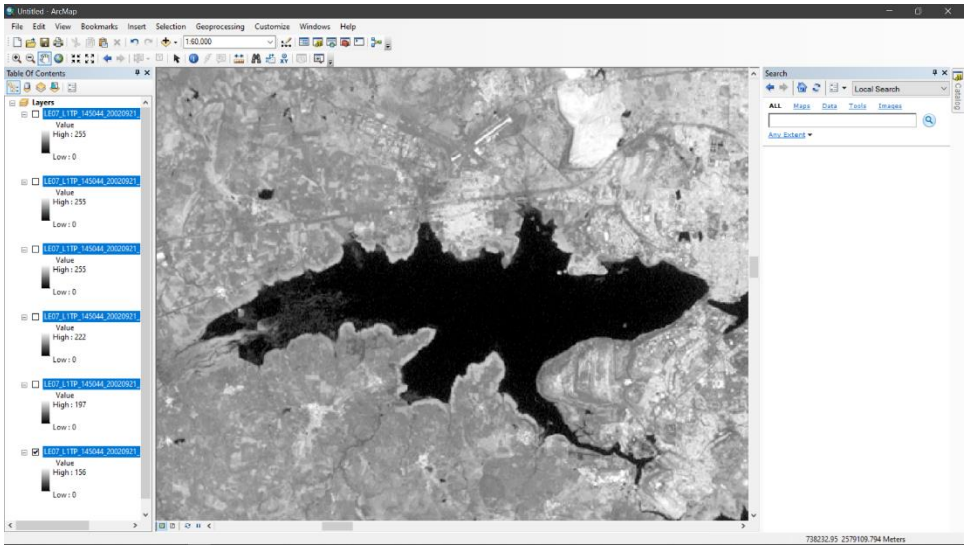


Fig. 20: **Band 7 image**

Construction of composite images:

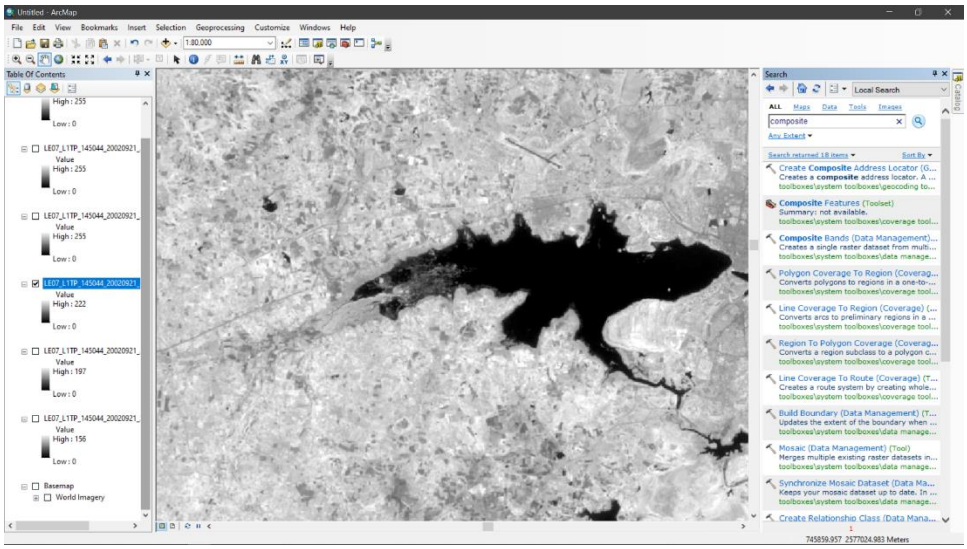


Fig. 21: **Open ‘Composite Bands’ feature**

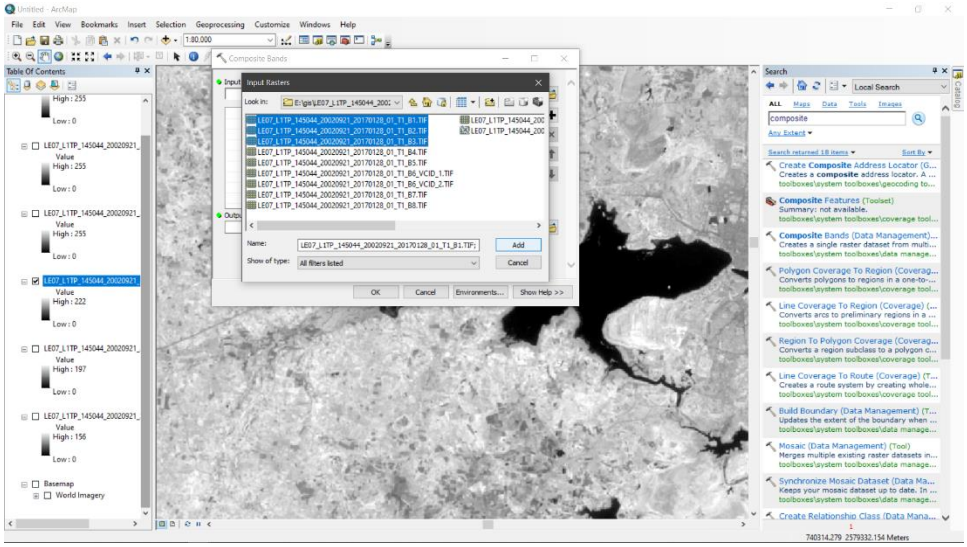


Fig. 22: **Input the desired bands**

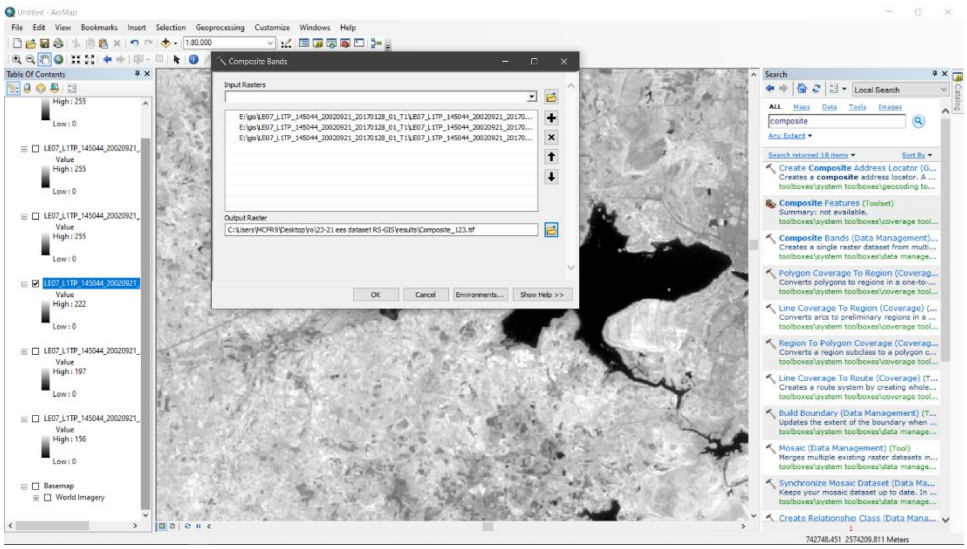


Fig. 23: Select the output file and click 'Okay'

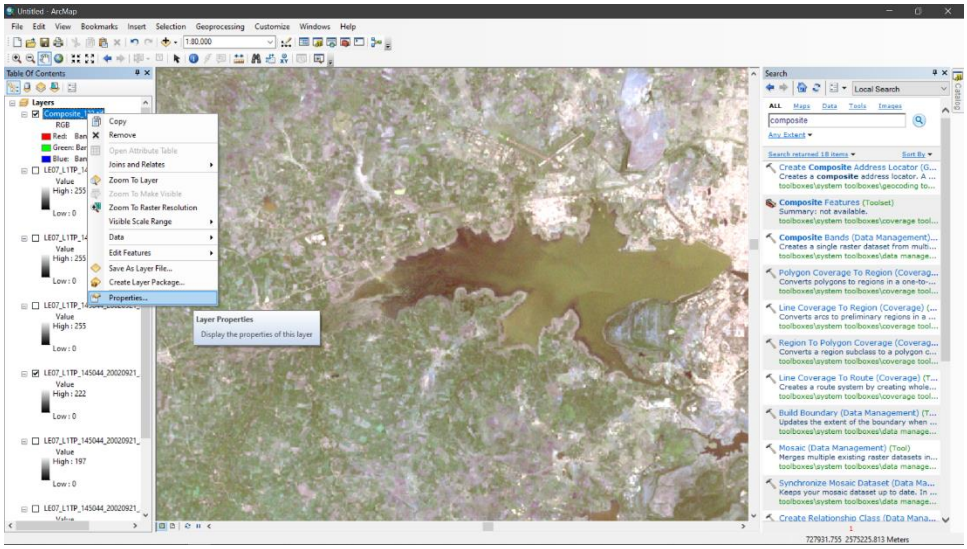


Fig. 24: View properties of the composite image

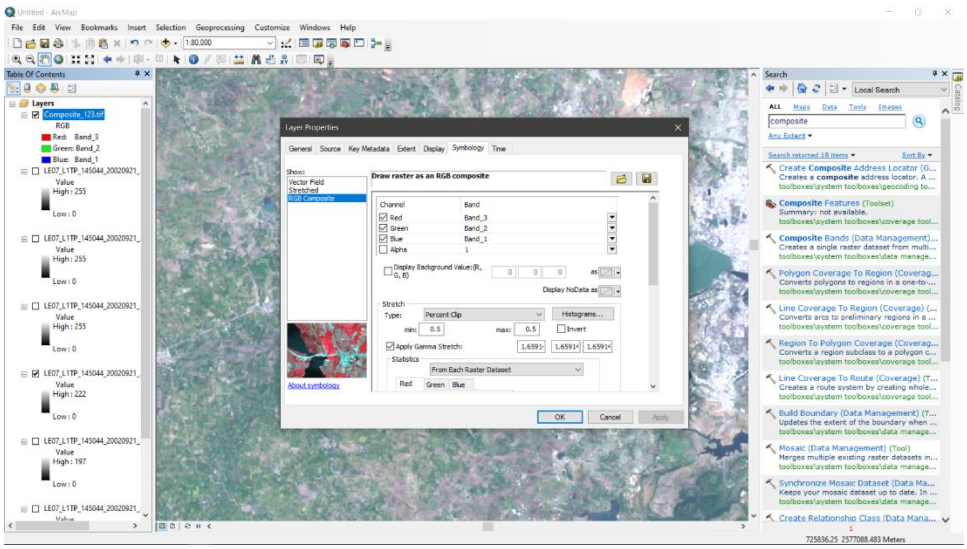


Fig. 25: Correct the symbology of the bands

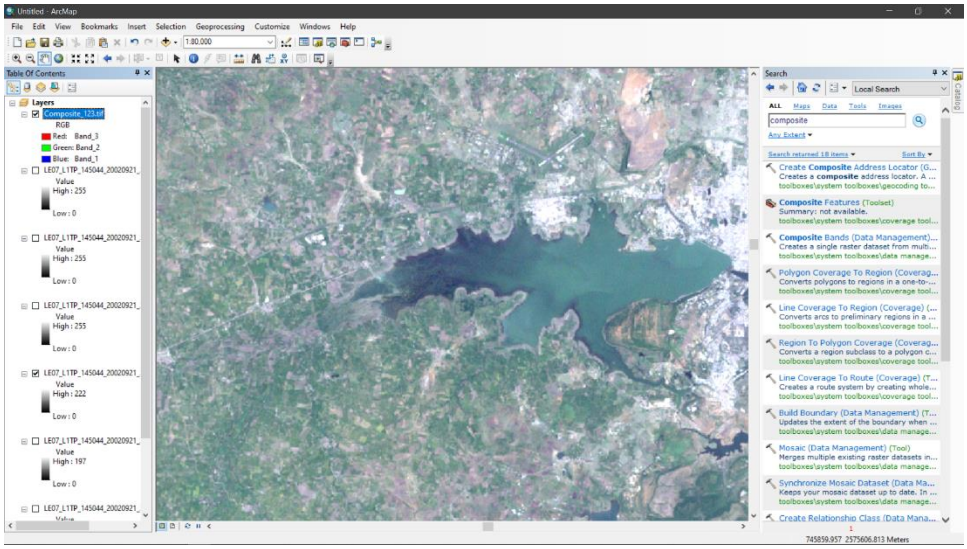


Fig. 26: Result: Natural Colour Composite of LC07

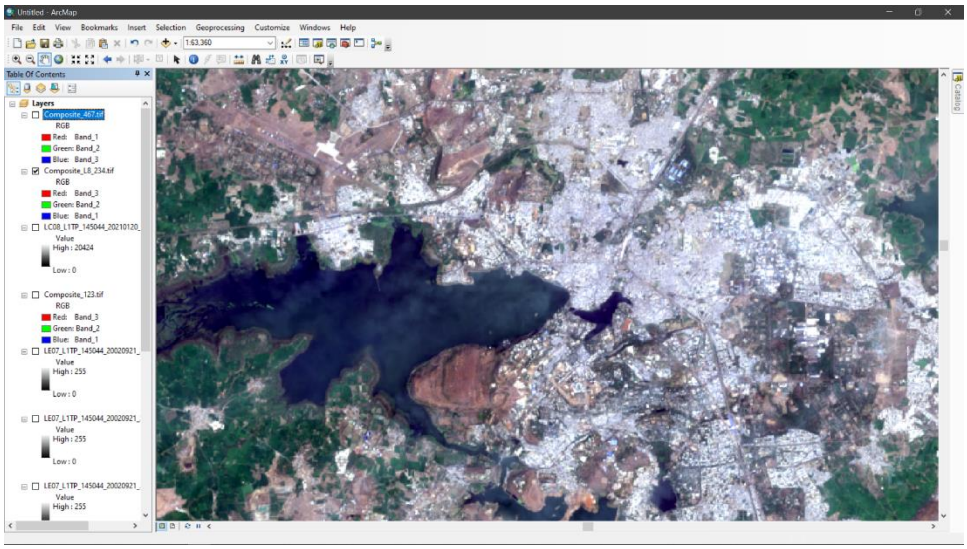


Fig. 27: Result: Natural Colour Composite of LC08

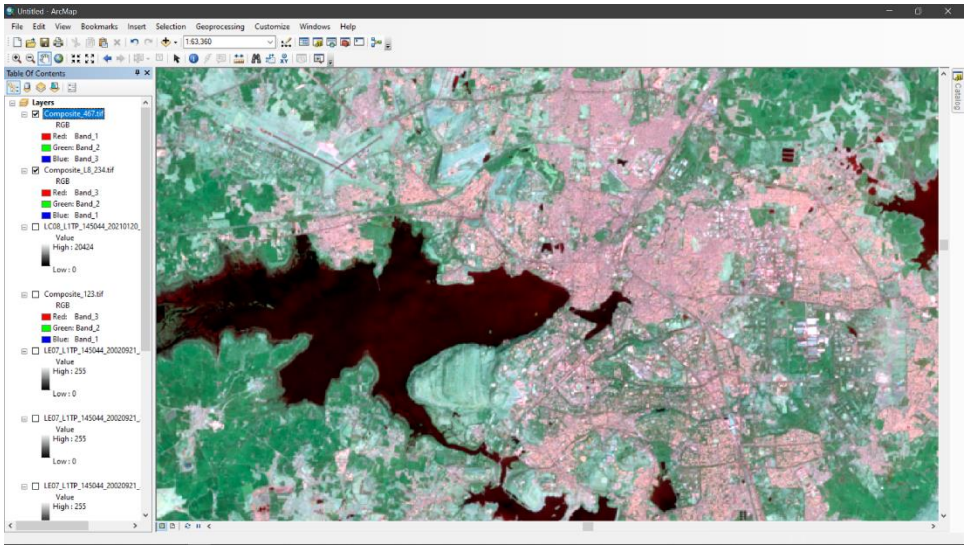


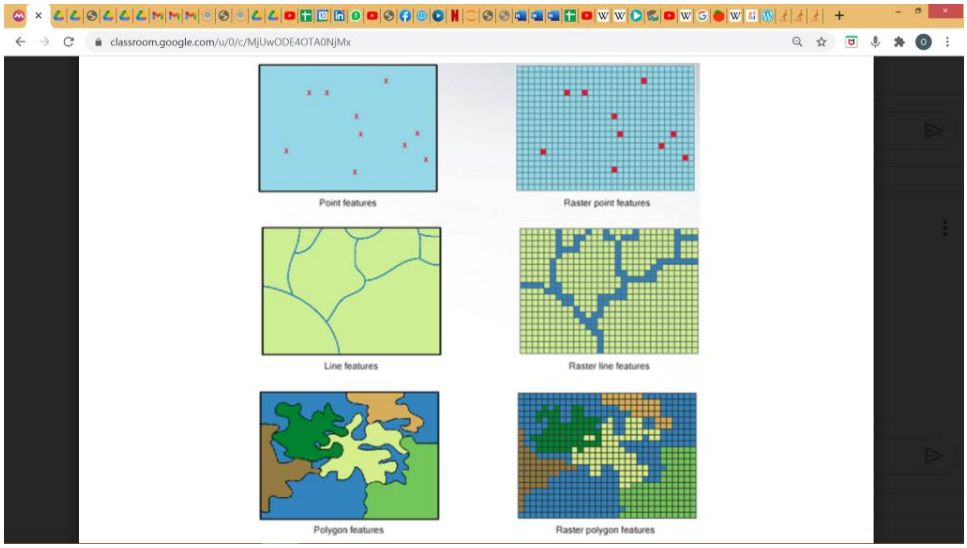
Fig. 28: Result: False Colour Composite of LC08

Name: Om Mahesh Vaknalli

Roll No. 18376

Subject: EES-338 (Remote Sensing & GIS Lab)

Lab – III (Shapefile, Vector data, Subset and Atmospheric Correction)



Disti **Distinction between Vector and Raster Data**

Recommended Solar Exoatmospheric Spectral Irradiances (ESUN) Values (in W/m2/µm)

Band Number	Landsat 7 ETM+	Landsat 5 TM	Landsat 4 TM	Landsat 1-5 MSS
1	1970	1958	1958	1848
2	1842	1827	1826	1588
3	1547	1551	1554	1235
4	1044	1036	1033	856.6
5	225.7	214.9	214.7	-
7	82.06	80.65	80.70	-
8	1369	-	-	-

Fig. 2: ESUN Values for Landsats 1-7 (Landsat 8 does not require it)

Creation of a Shapefile and an Area of Interest (AOI):

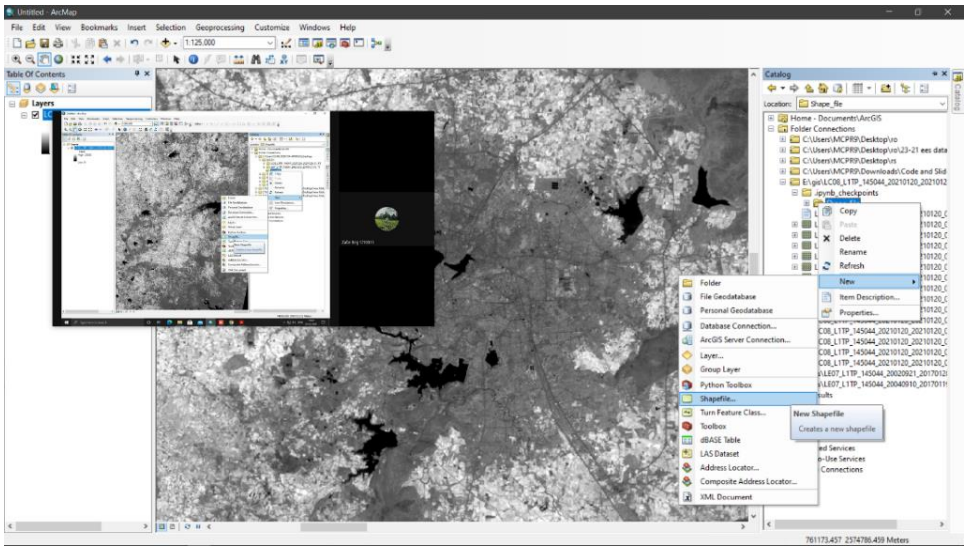


Fig. 3: Open a band image, then open the 'Create a new Shapefile' icon as shown

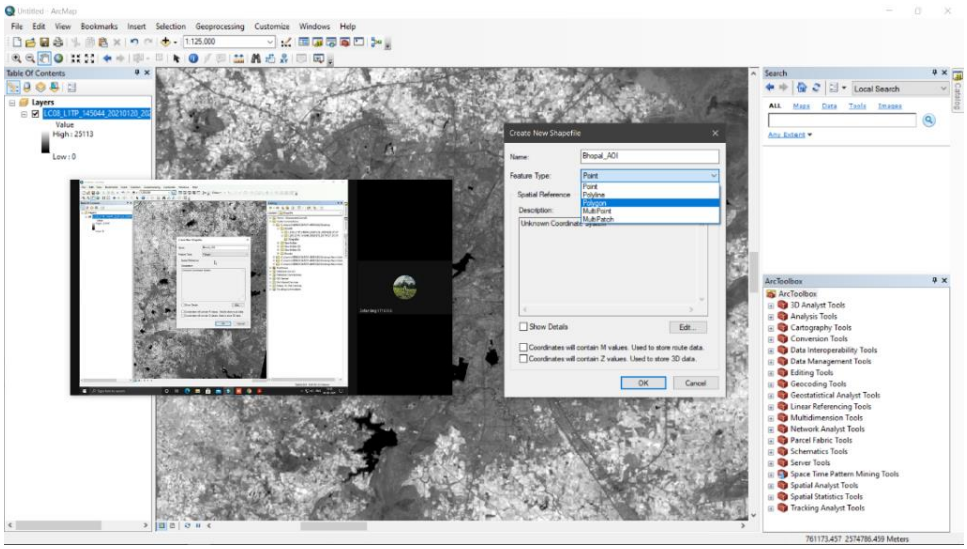


Fig. 4: Choose the type of vector feature of the shapefile

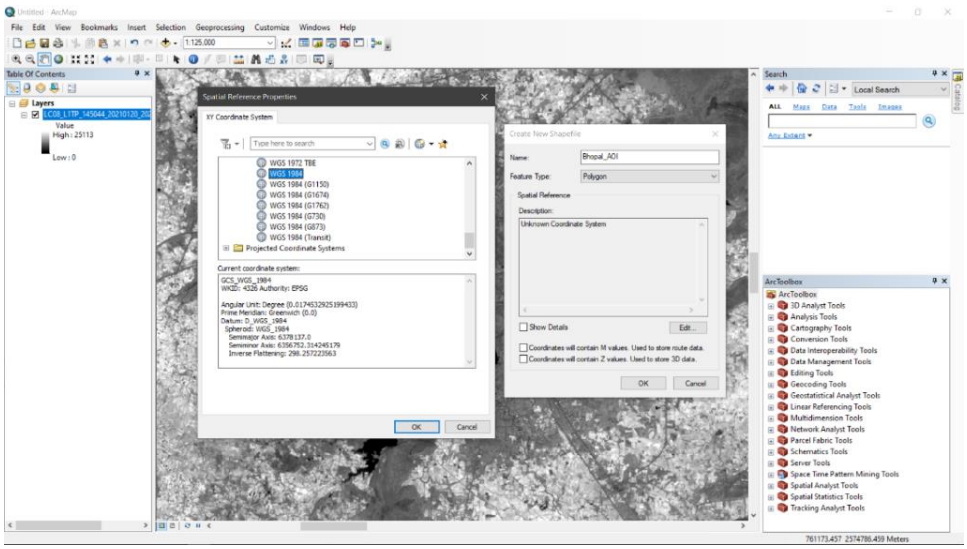


Fig. 5: **Choose the appropriate spatial reference system that matches the selected image.**

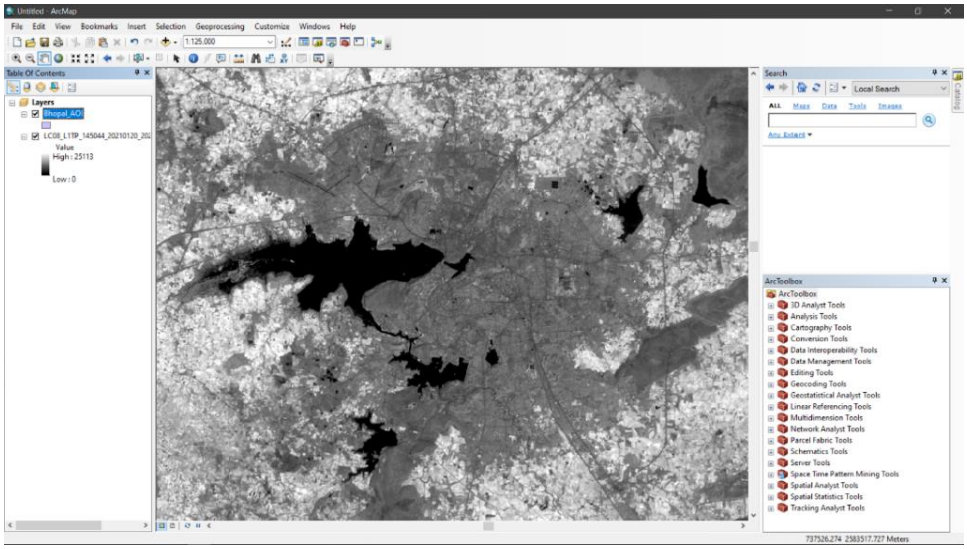


Fig. 6: **Result: Shapefile created**

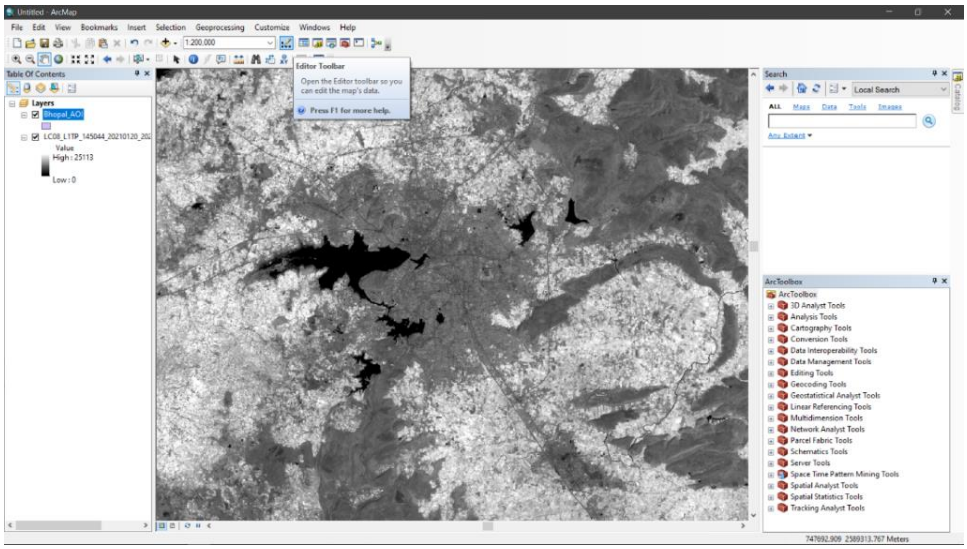


Fig. 7: Open the 'Editor Toolbar'

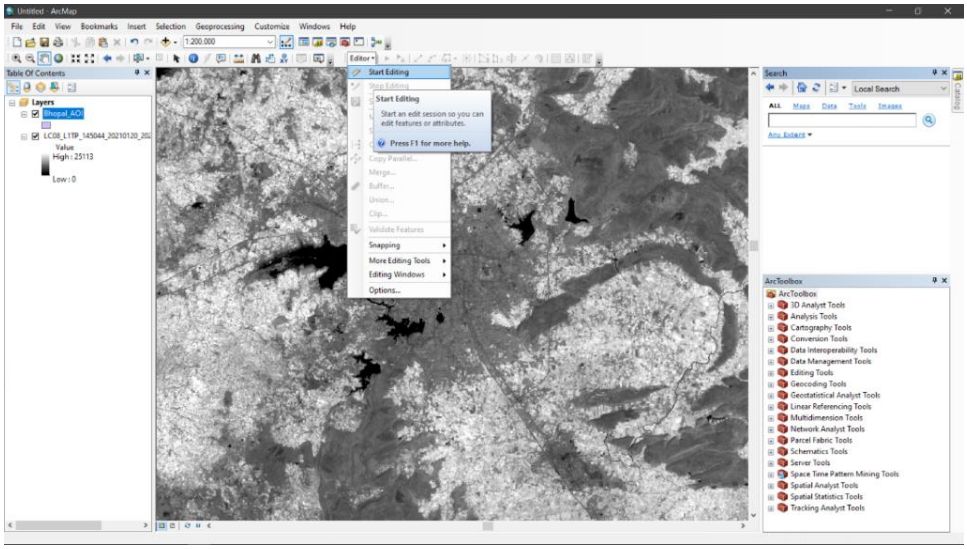


Fig. 8: Select the 'Start Editing' feature

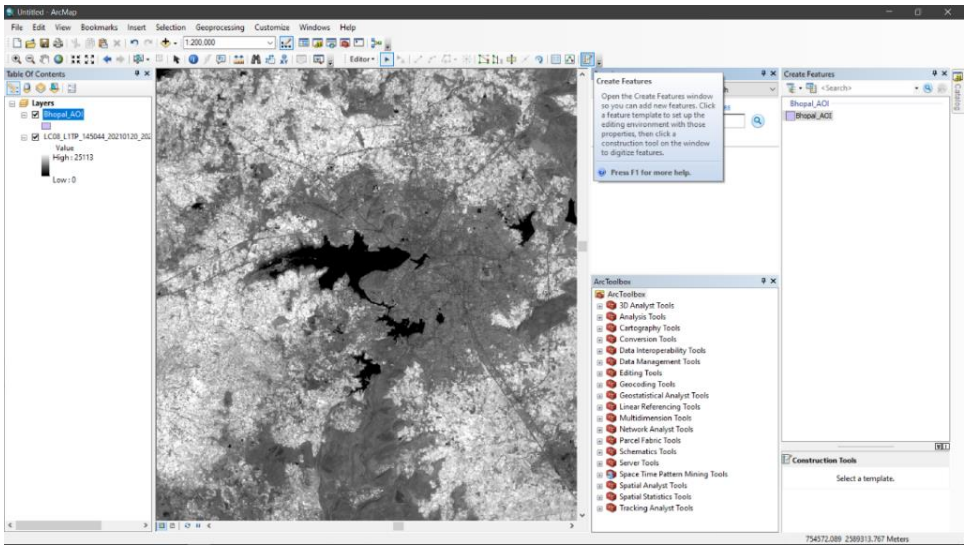


Fig. 9: Click on 'Create Features'

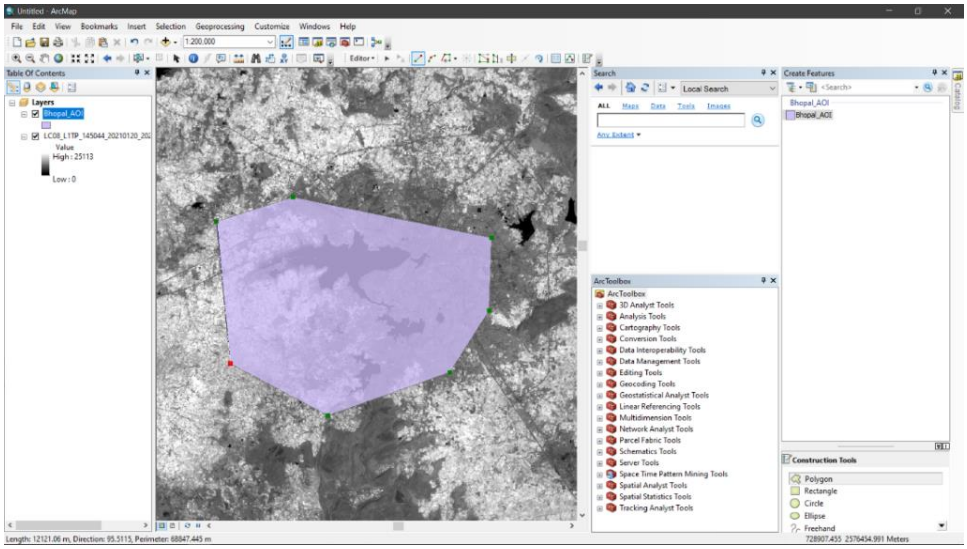


Fig. 10: Trace out the type and shape of AOI required

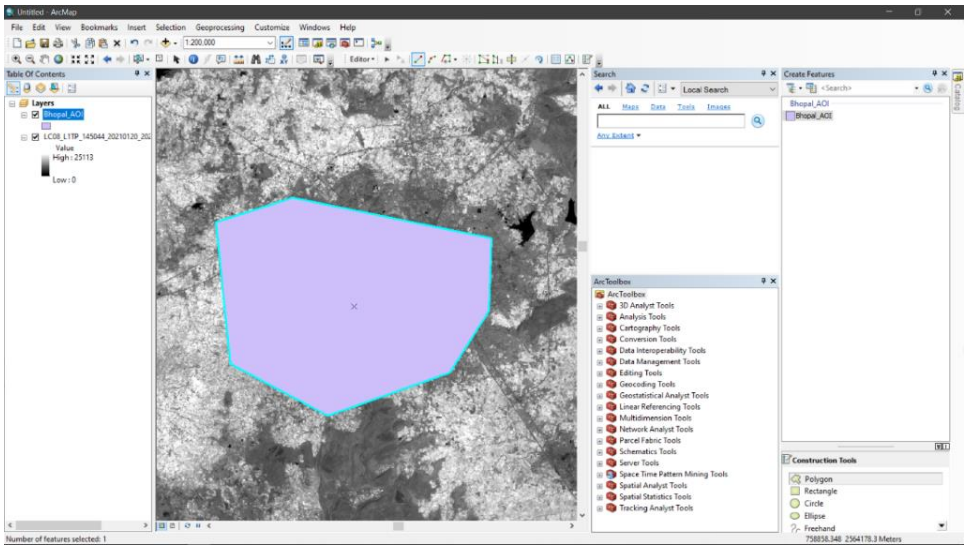


Fig. 11: **Complete Editing of the AOI**

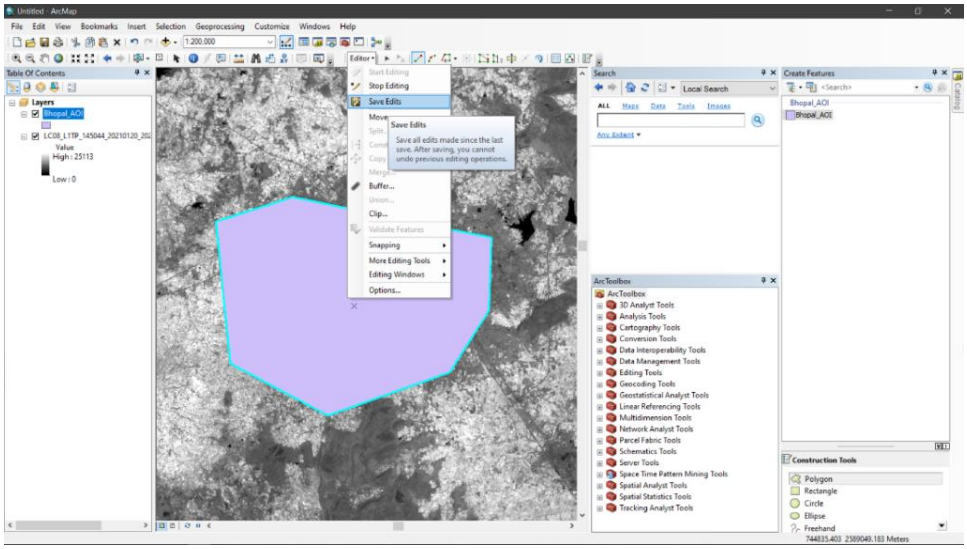


Fig. 12: **Save the created AOI**

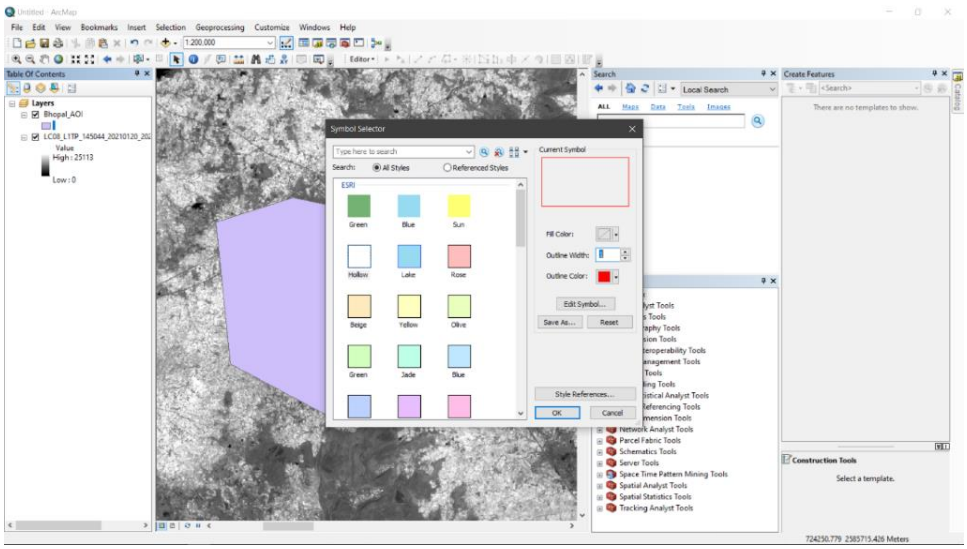


Fig. 13: Alter the colour, transparency and border-colour of the AOI

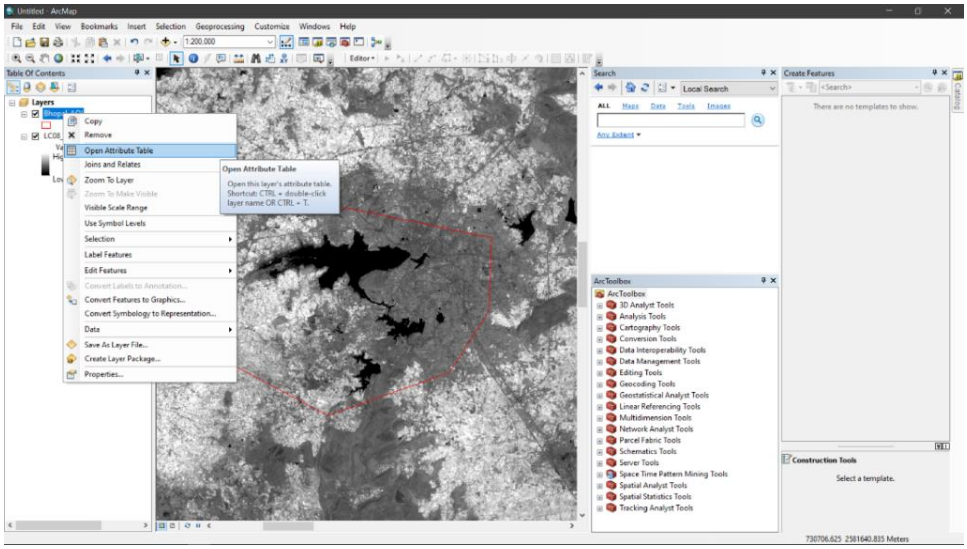


Fig. 14: Open the Attributes Table of the shapefile

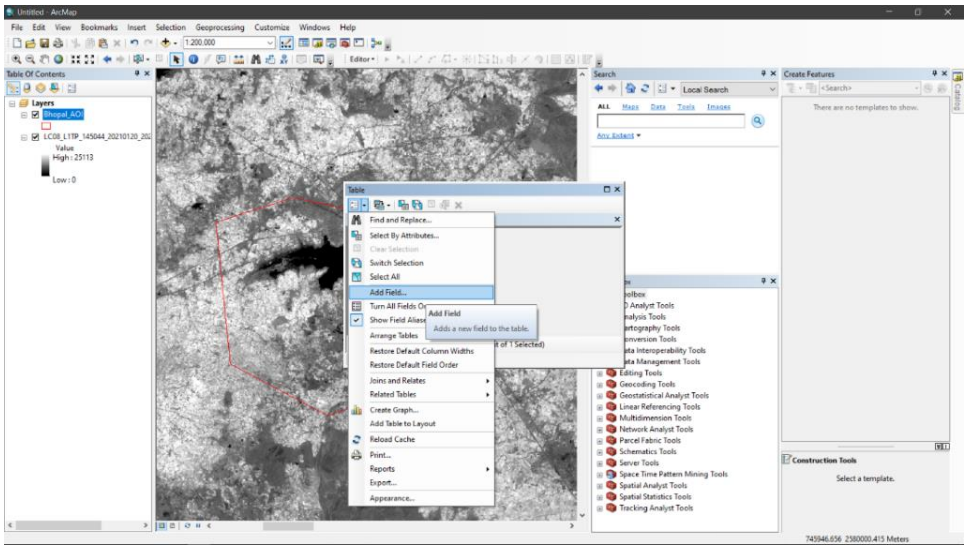


Fig. 15: Add a new field to the table

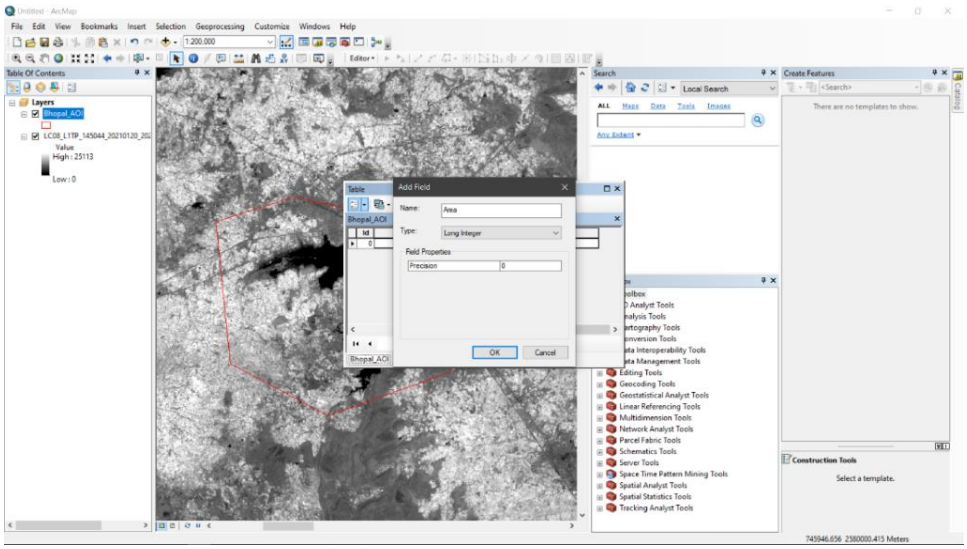


Fig. 16: Assign the area of the AOI in the new field

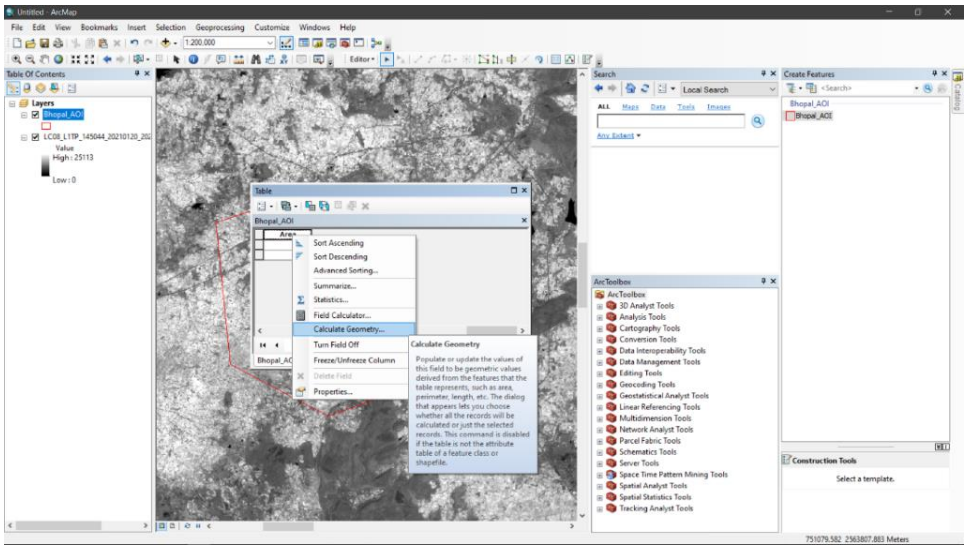


Fig. 17: Open the 'Calculate Geometry' tab

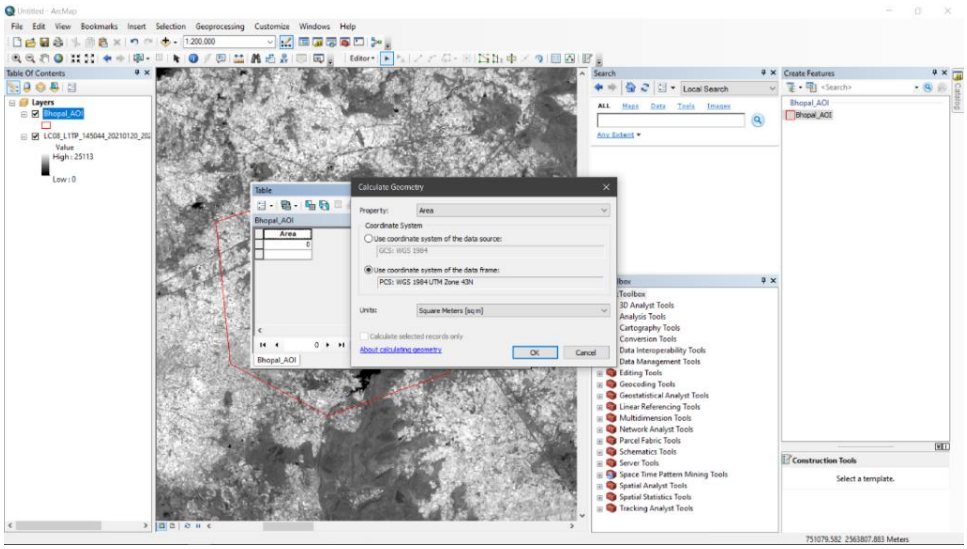


Fig. 18: Calculate the area of the AOI

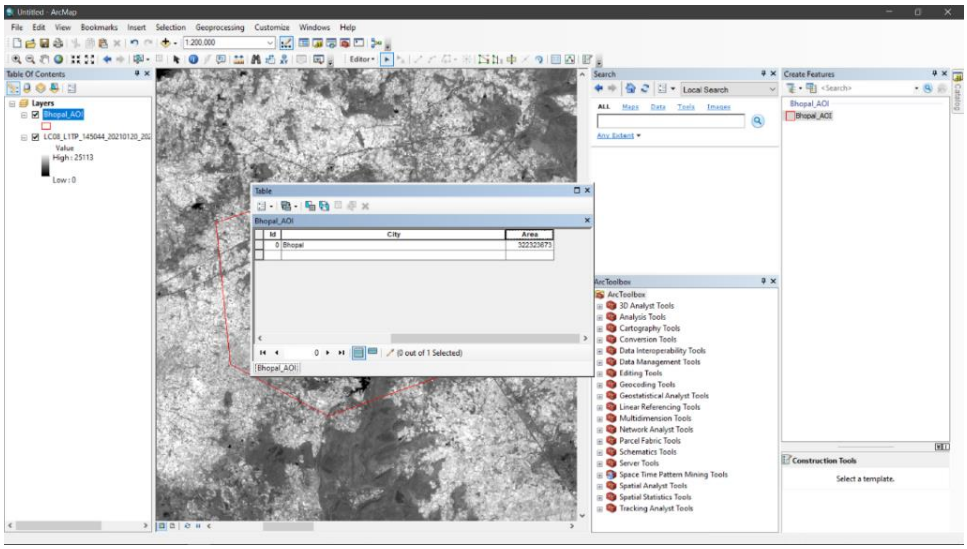


Fig. 19: **Result: AOI area calculated**

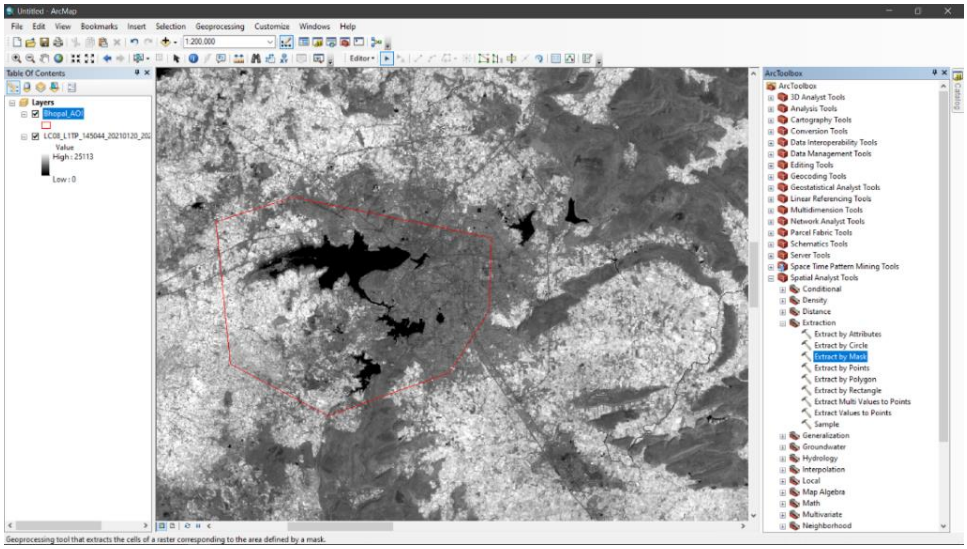


Fig. 20: **Result: AOI created as a shapefile**

Extraction of data by Vector Masking:

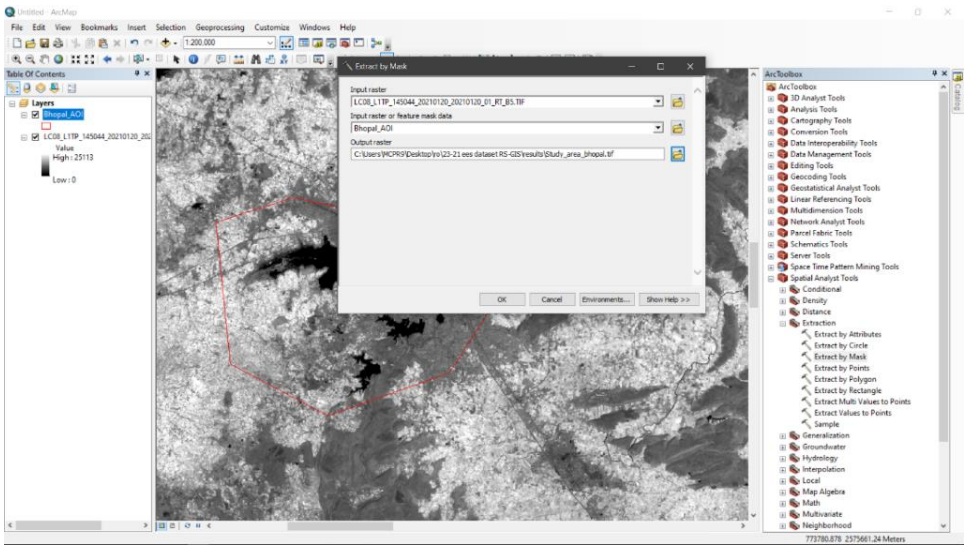


Fig. 21: Open the 'Extract by Mask' tool and enter the relevant information

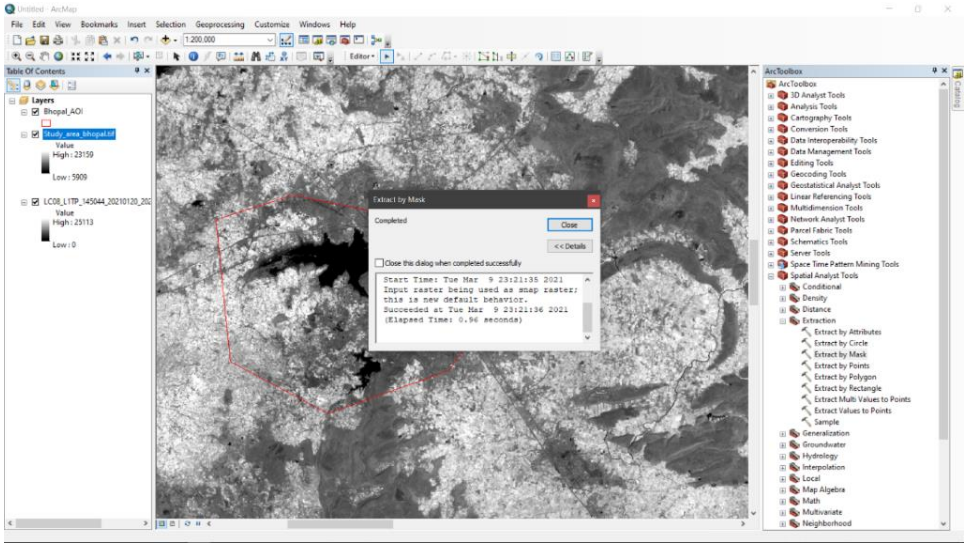


Fig. 22: Masking completed and computational time shown

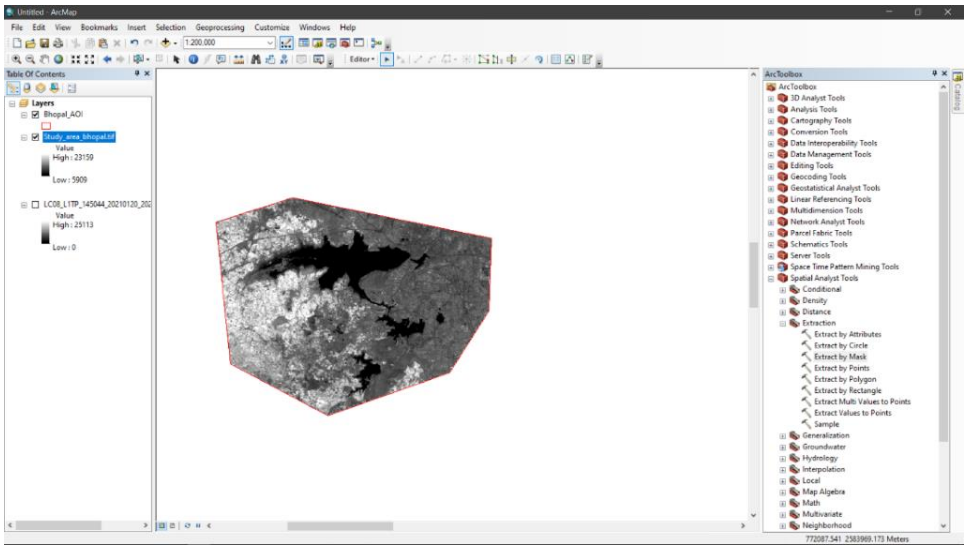


Fig. 23: **Results: Data extracted**

Performing Atmospheric Corrections:

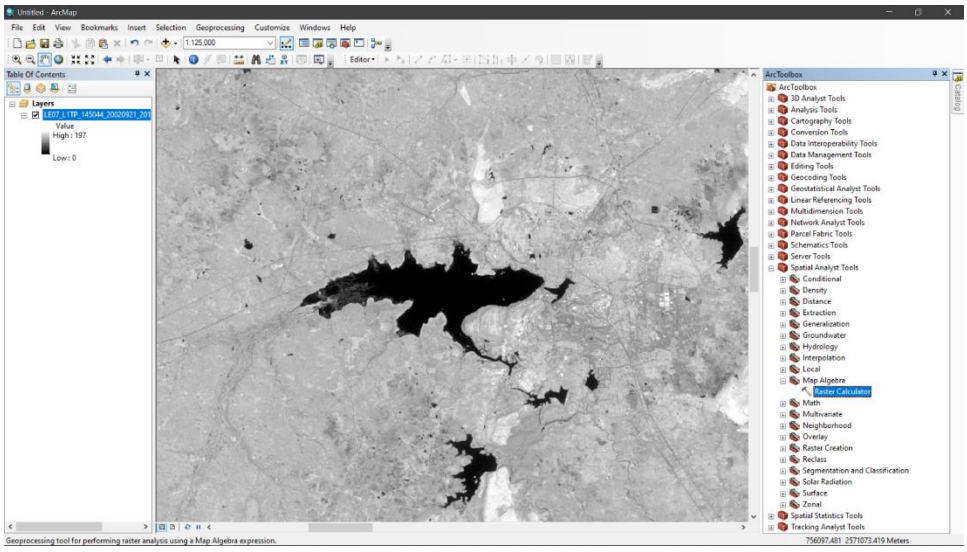


Fig. 24: **Open the Raster Calculator tool**

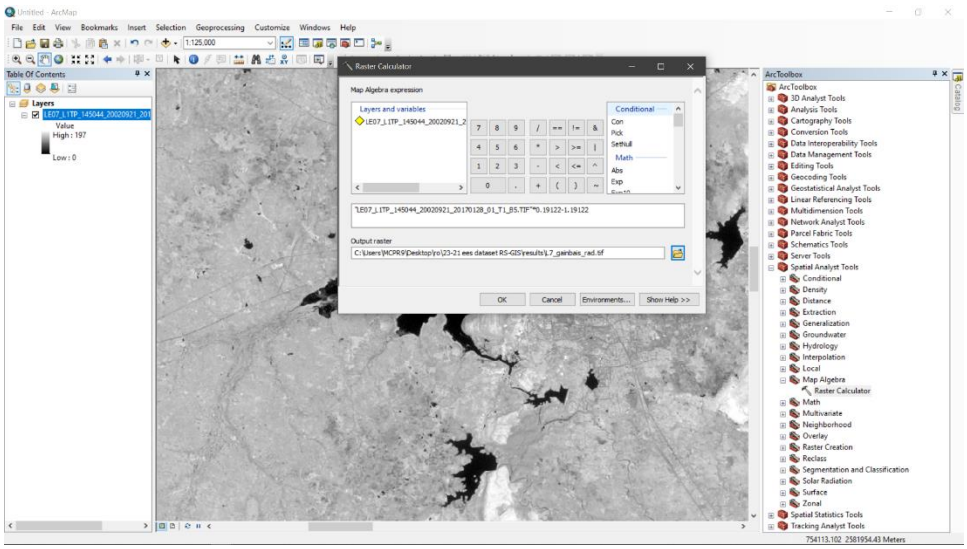


Fig. 25: Apply the Gain & Bias formula by taking the multiplicative & additive coefficient values from the metadata file

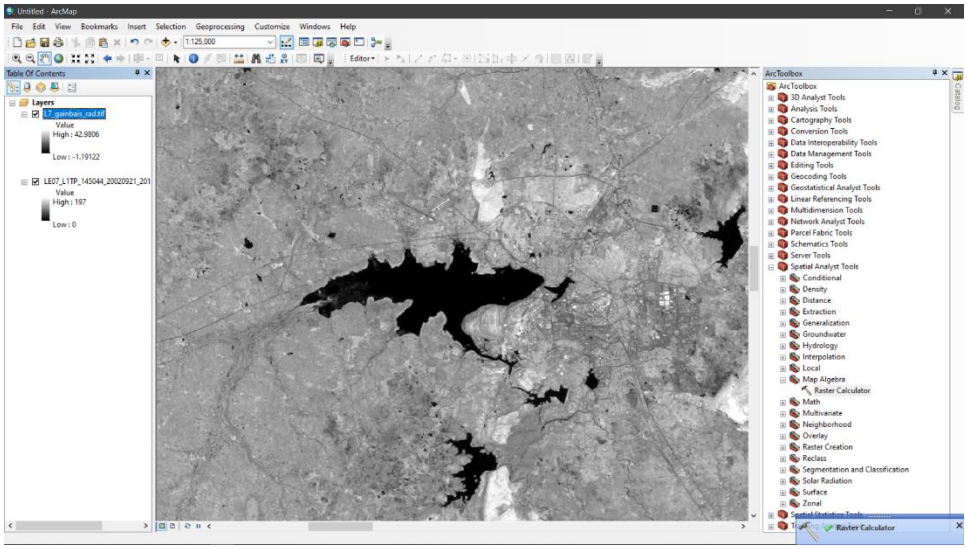


Fig. 26: Result: DN to Radiance conversion by Gain and Bias Method

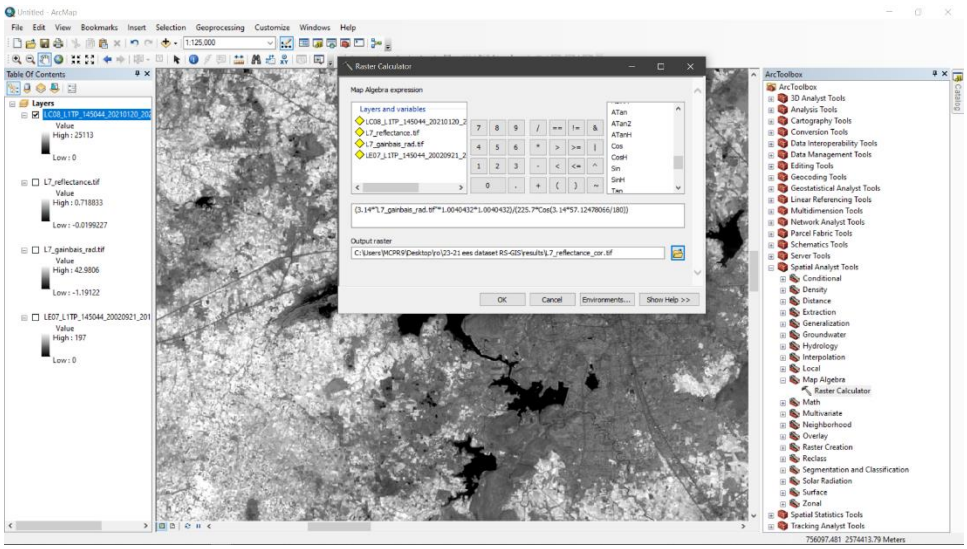


Fig. 27: Apply the Radiance to ToA reflectance formula by taking the d and φ values from the metadata file and the ESUN values from Fig. 2

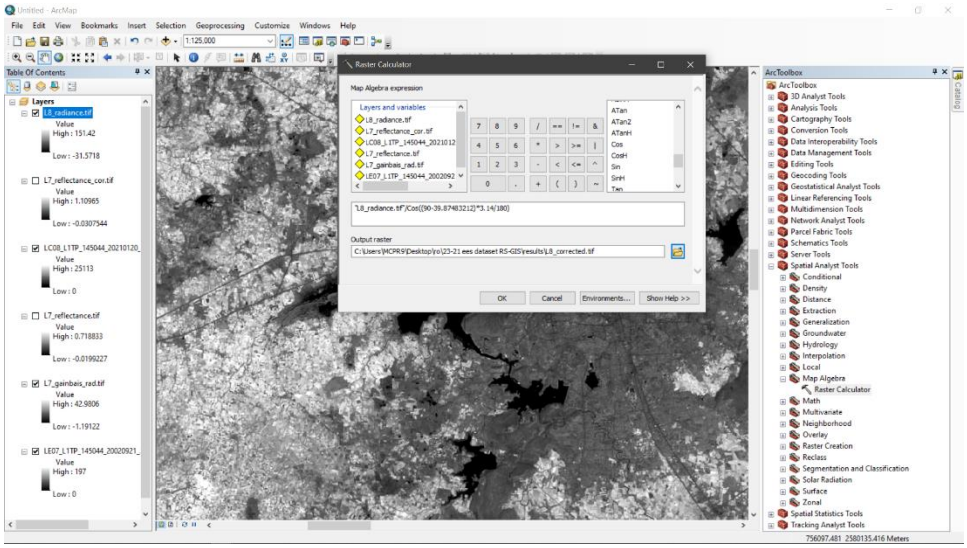


Fig. 28: Radiance to ToA reflectance for Landsat 8 calculations, φ is taken from the metadata file

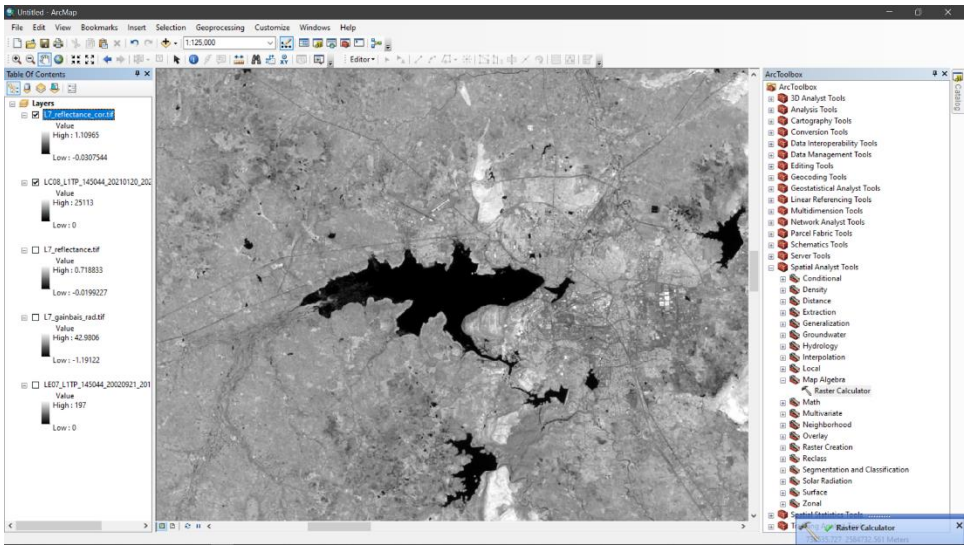


Fig. 29: **Result: Radiance to ToA reflectance conversion for Landsat 7**

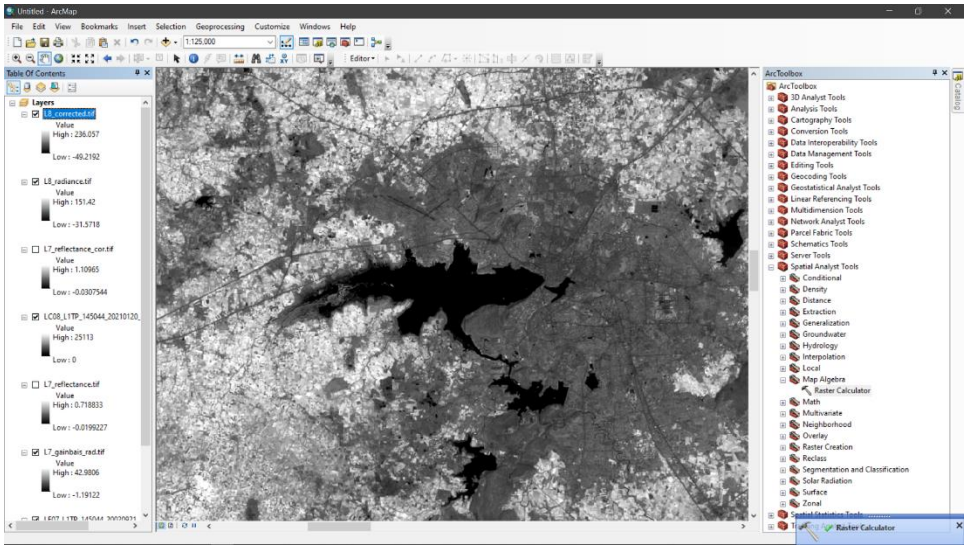


Fig. 30: **Result: Radiance to ToA reflectance conversion for Landsat 8**

Performing Dark Object Subtraction:

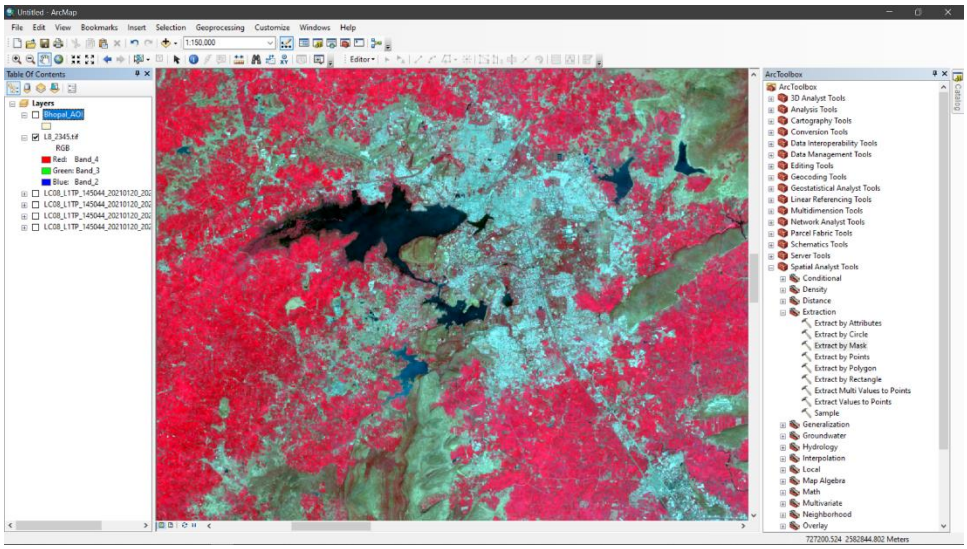


Fig. 31: Create a composite image of bands 2, 3, 4 & 5 from Landsat 8 that contain at least one water body

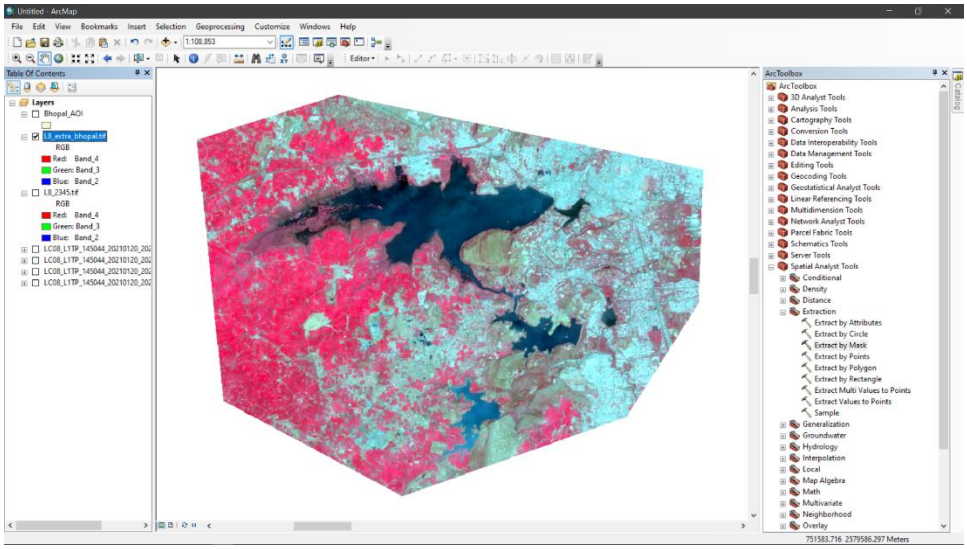


Fig. 32: Extract an AOI datafile using mask extraction from the composite image

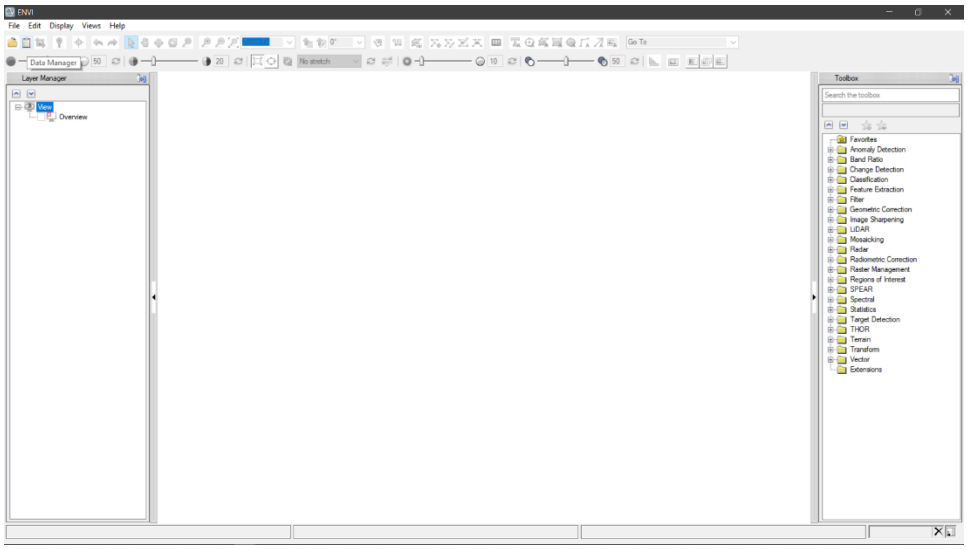


Fig. 33: Open the ENVI software

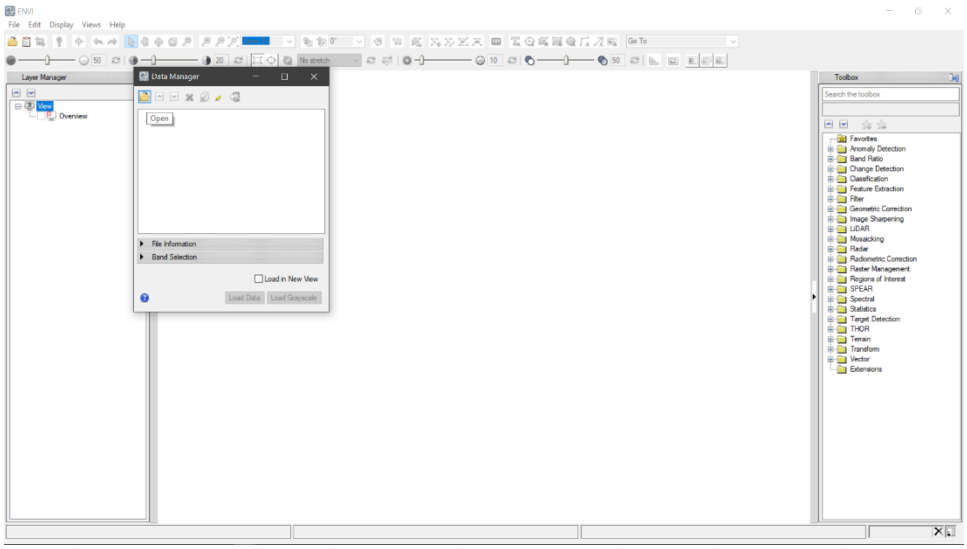


Fig. 34: Import the AOI datafile using the 'Data Manager' tool

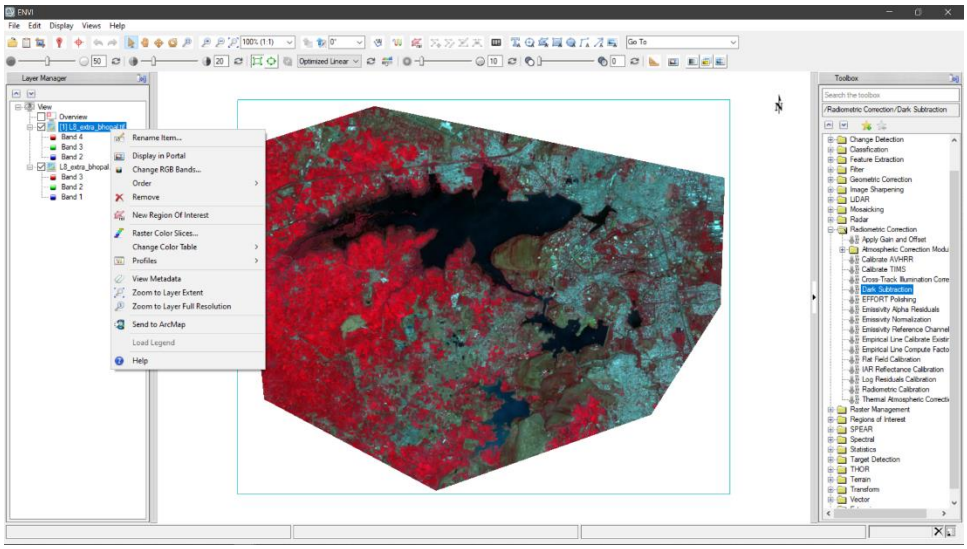


Fig. 35: Open the 'New Region of Interest' tool

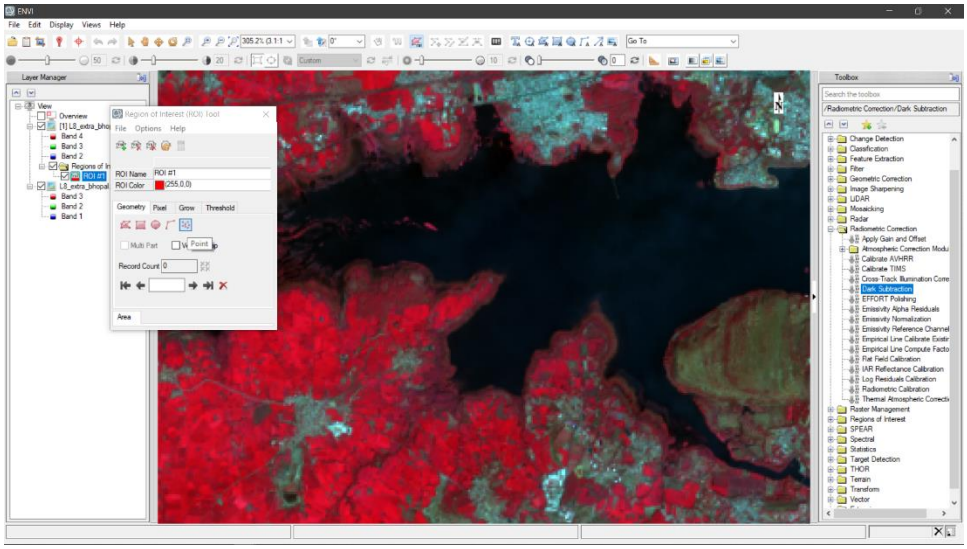


Fig. 36: Select the geometry of selection as 'point'

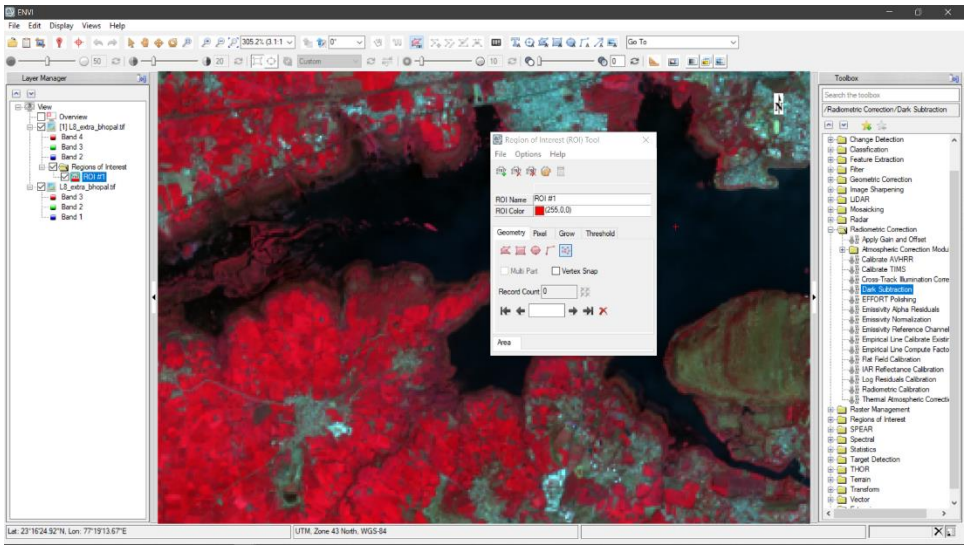


Fig. 37: Select one the darkest points (point with a very low DN value) of the image (mostly in the water body) and save it

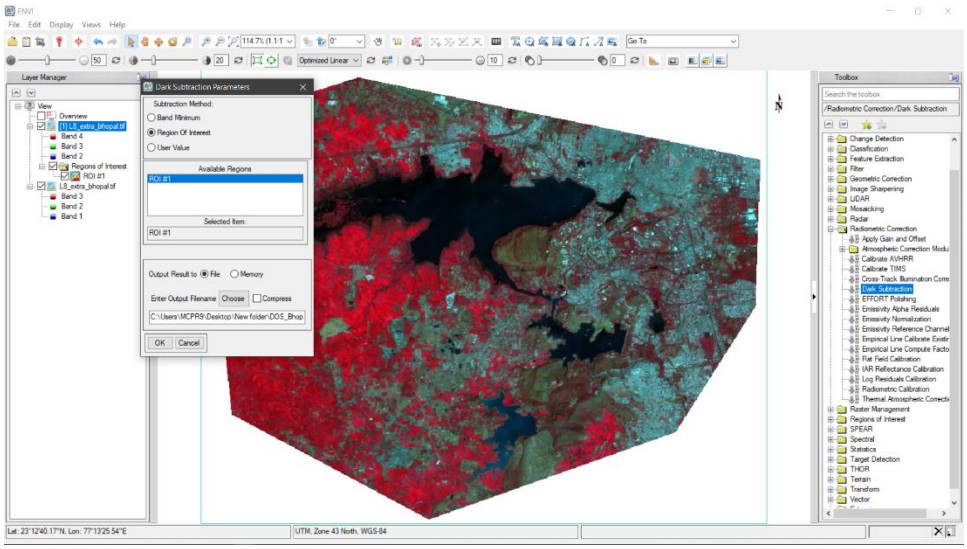


Fig. 38: Open the 'Dark Subtraction' tool and select the saved point, then choose the location for saving the output file

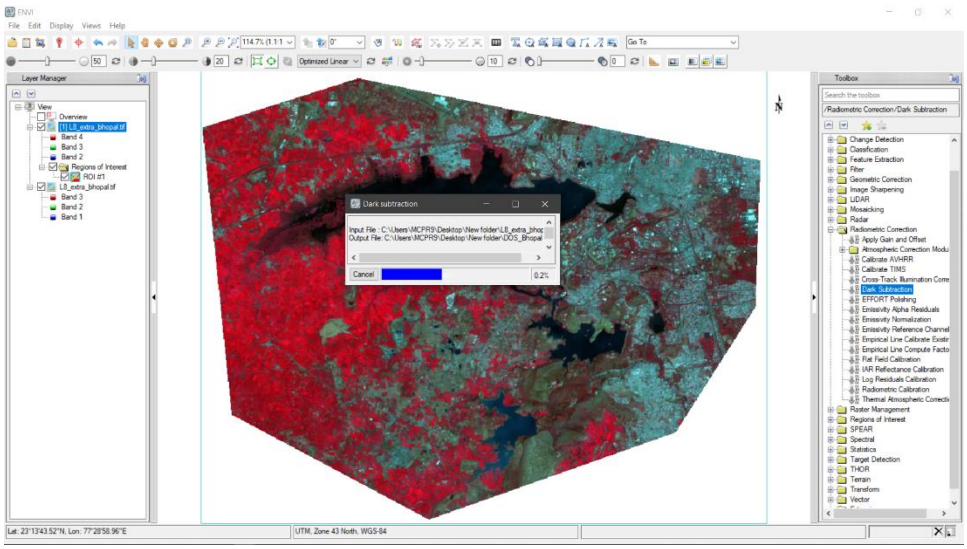


Fig. 39: Running the tool performs the Dark Subtraction

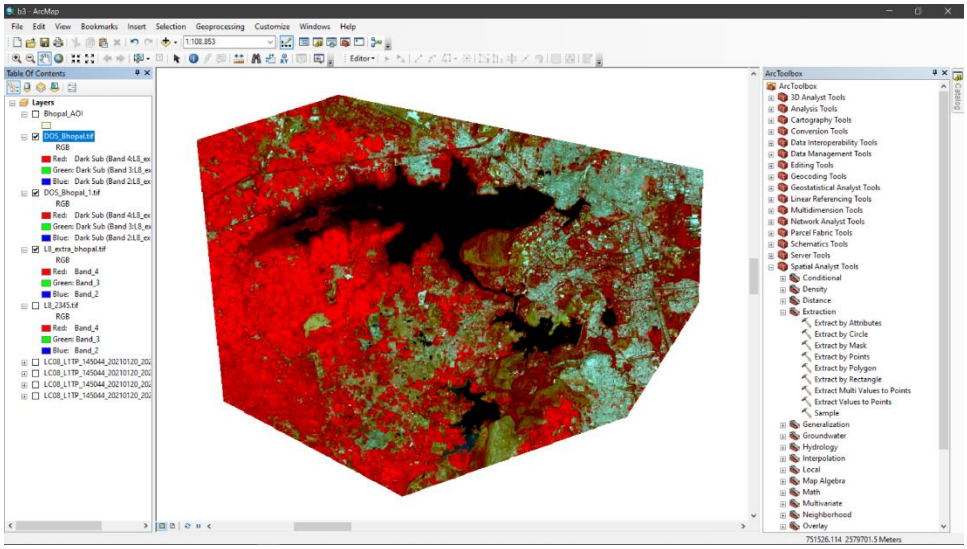


Fig. 40: Result: Dark Subtraction of Image completed

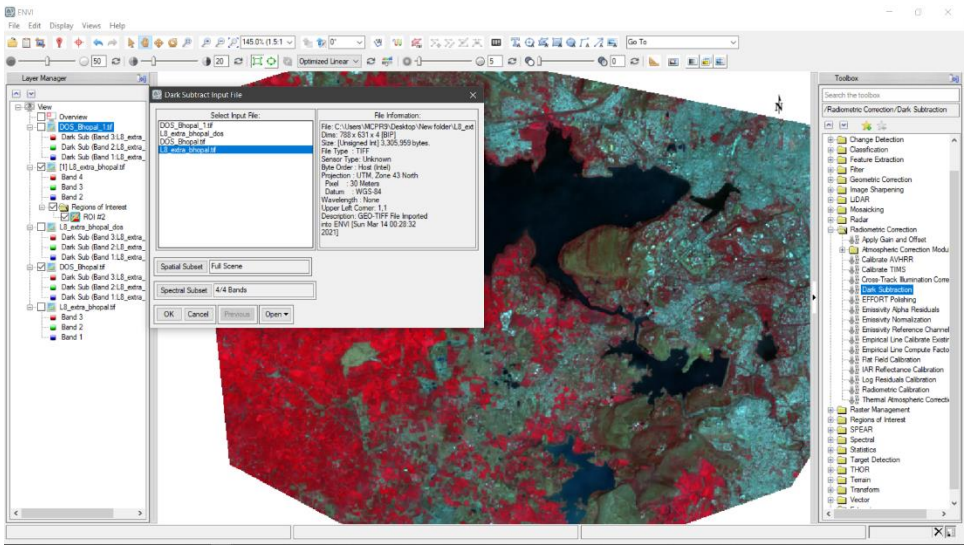


Fig. 41: Properties of all the opened images can be seen

Name: Om Mahesh Vaknalli

Roll No. 18376

Subject: EES-338 (Remote Sensing & GIS Lab)

Lab – IV (Georeferencing and Image Resampling)

Georeferencing using Geometrical Correction:

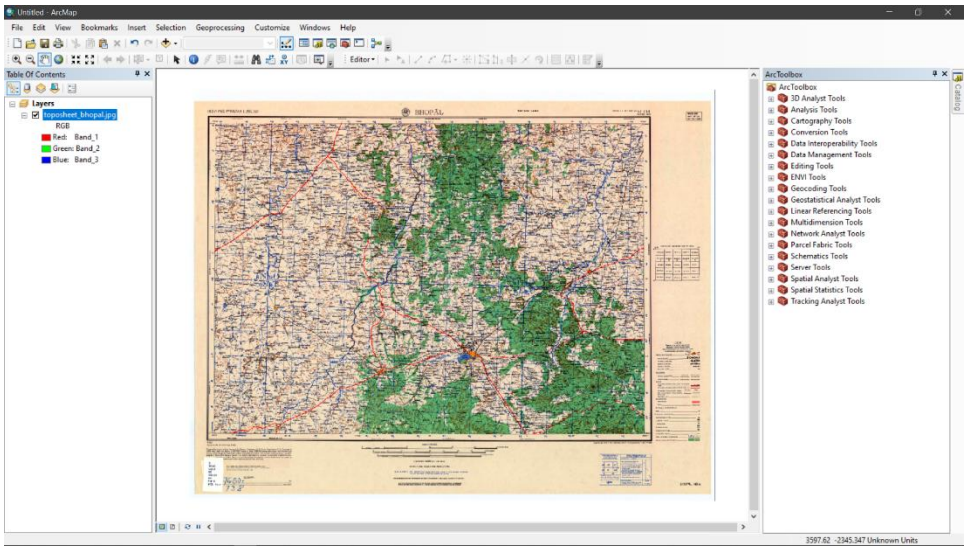


Fig. 1: Load a toposheet on ArcGIS

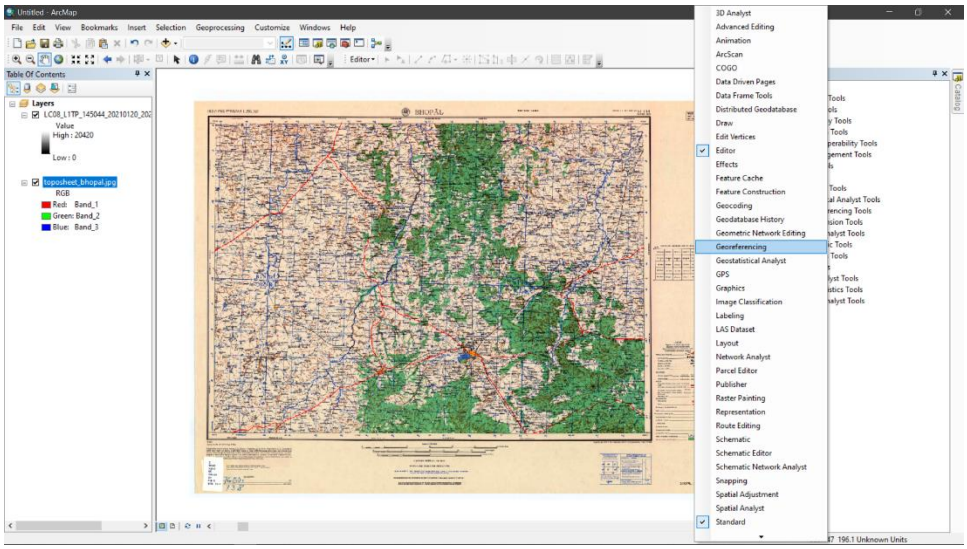


Fig. 2: Load another already georeferenced toposheet of the same location to cross-check the georeferencing of the toposheet at hand after it is done. Enable the 'Georeferencing' tool

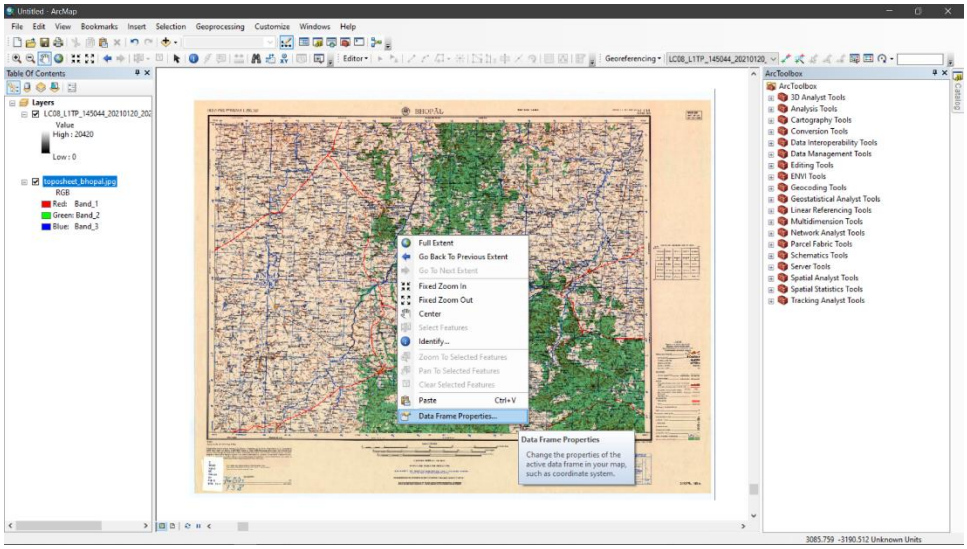


Fig. 3: View the data frame properties of the image

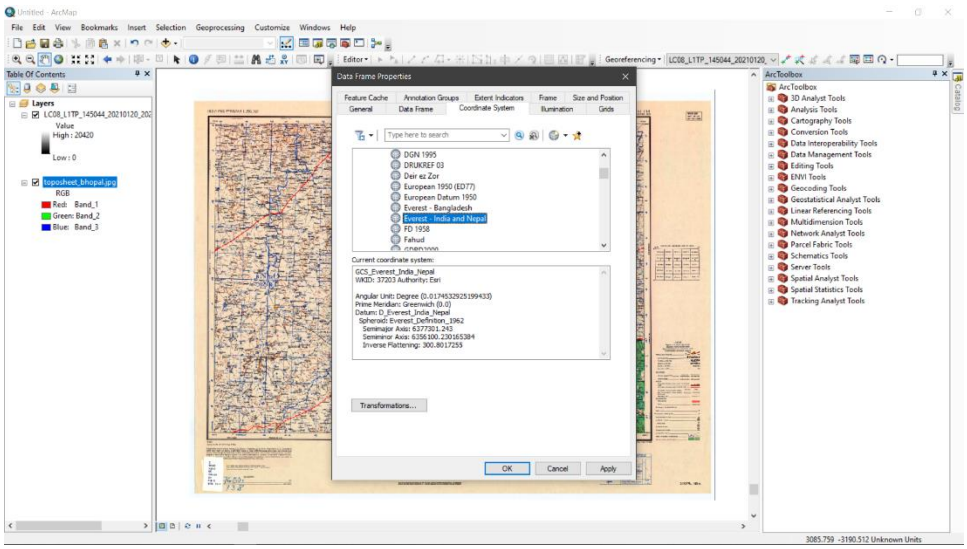


Fig. 4: Select the appropriate coordinate system that matches the map and click 'OK'

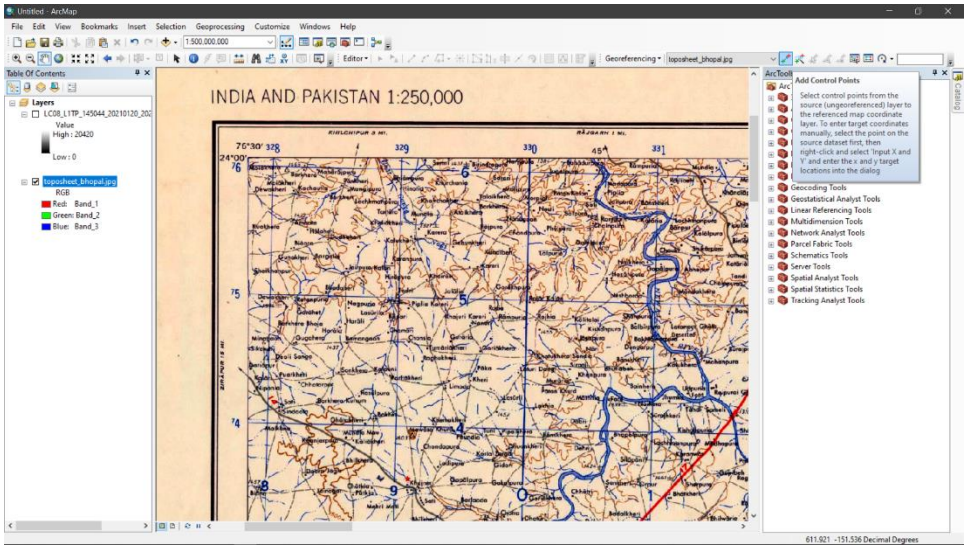


Fig. 5: Click on the 'Add Control Points' tab

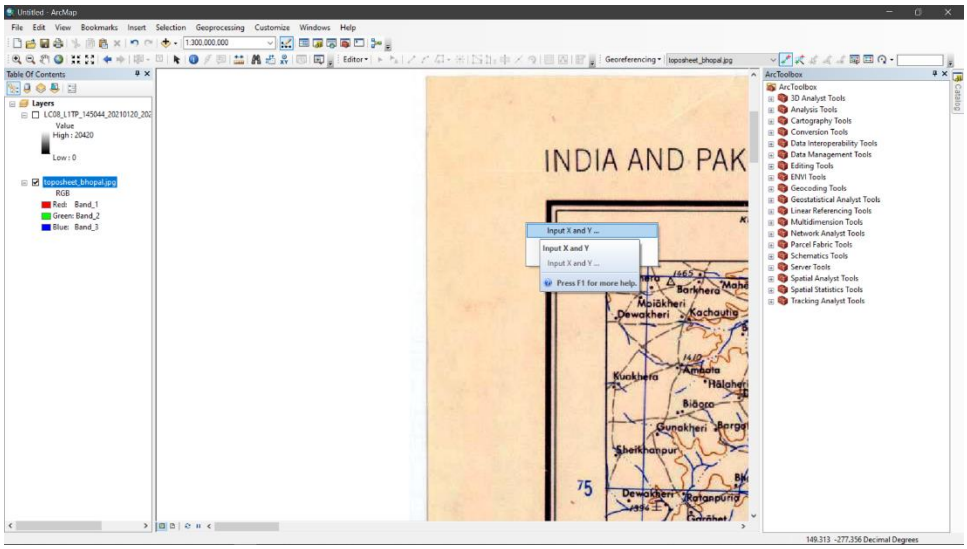


Fig. 6: Plot the appropriate number of GCPs (according to the formula). Select those points on toposheet as GCPs whose coordinates are mentioned on the map or known otherwise. To plot a GCP, right click on the desired location and select 'Input X and Y'

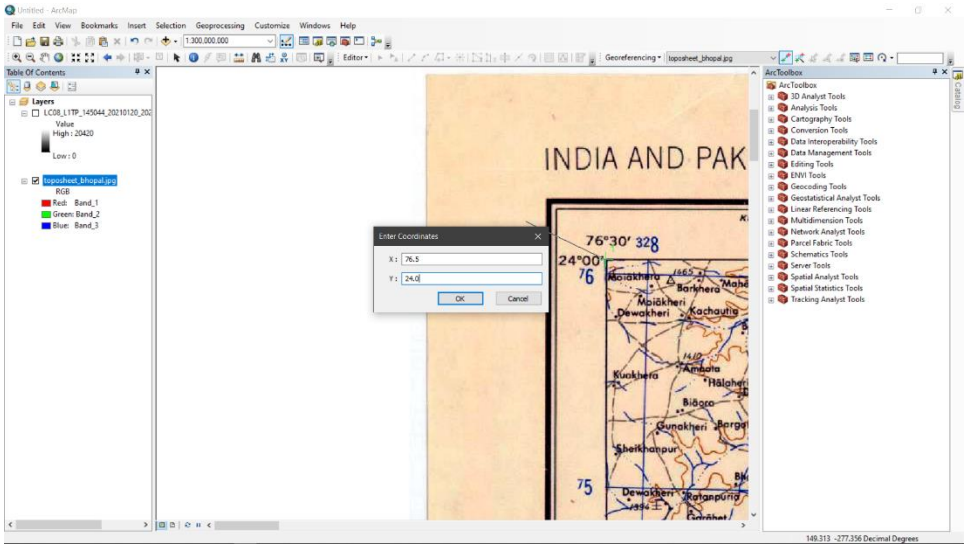


Fig. 7: Enter the local coordinates of the point as mentioned on the toposheet or known otherwise

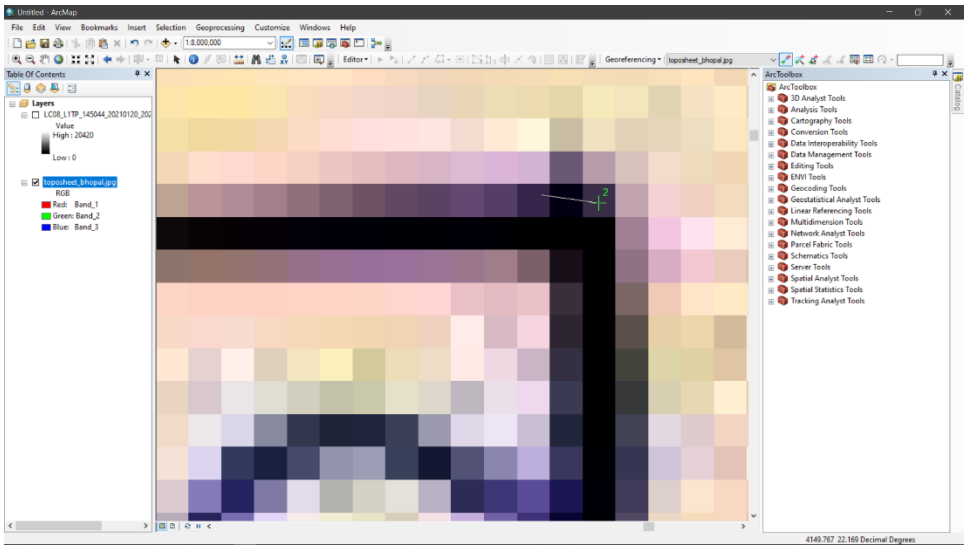


Fig. 8: Zoom into the toposheet to mark GCPs with greater accuracy

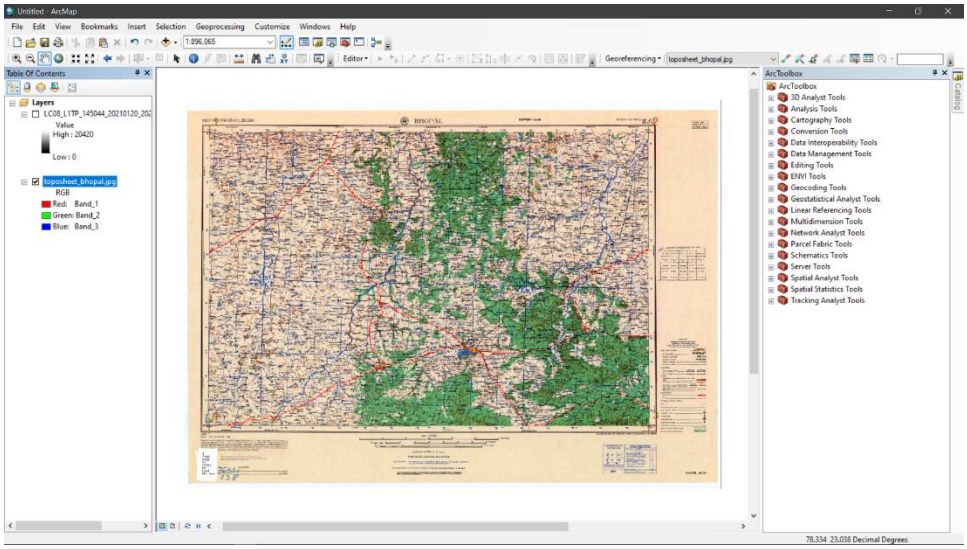


Fig. 9: Result: GCP marking completed

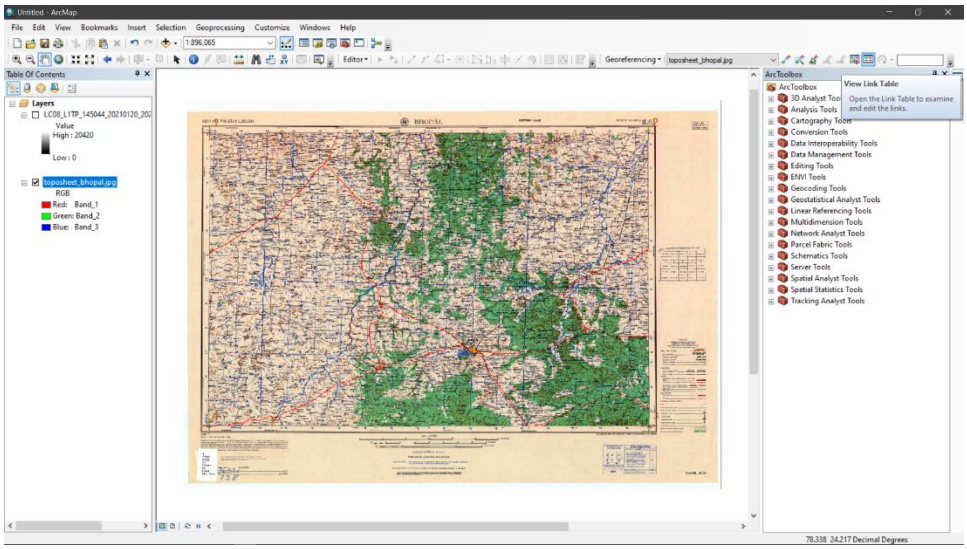


Fig. 10: Open the link table to view the details about the added GCPs

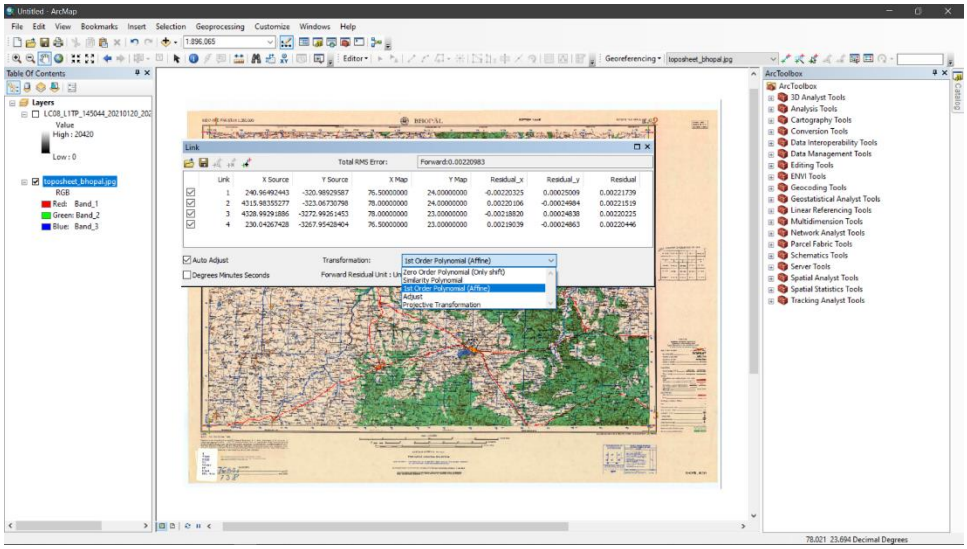


Fig. 11: Select the appropriate type of transformation to decide the accuracy of georeferencing (at the cost of computational time and file size)

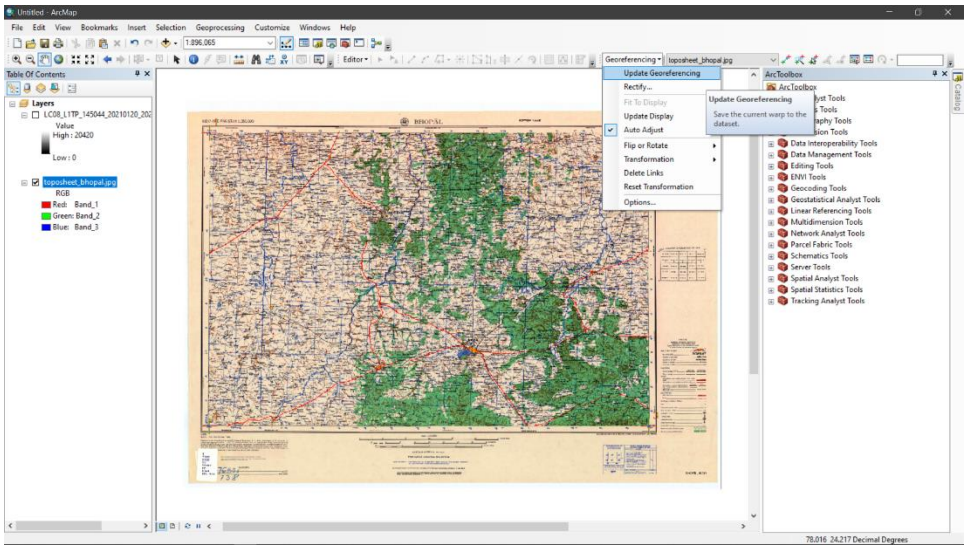


Fig. 12: Update the georeferencing

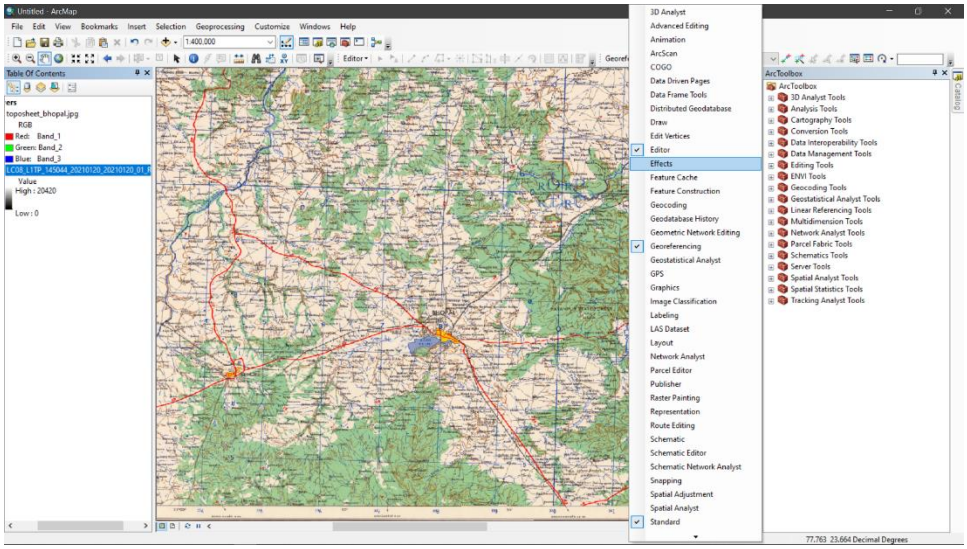


Fig. 13: Open the 'Effects' tool

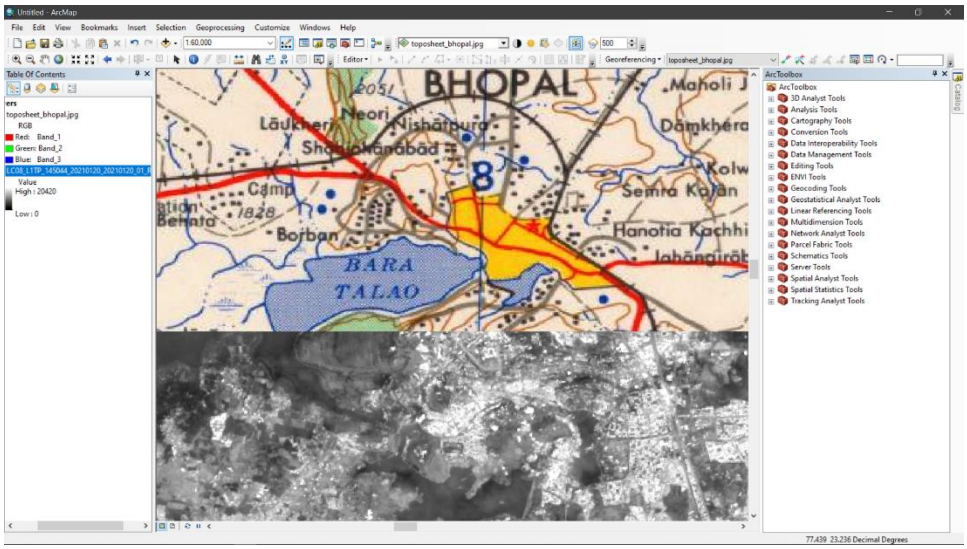


Fig. 14a: Cross-reference the presently georeferenced toposheet with the other one by sliding it up & down to check their mutual alignments

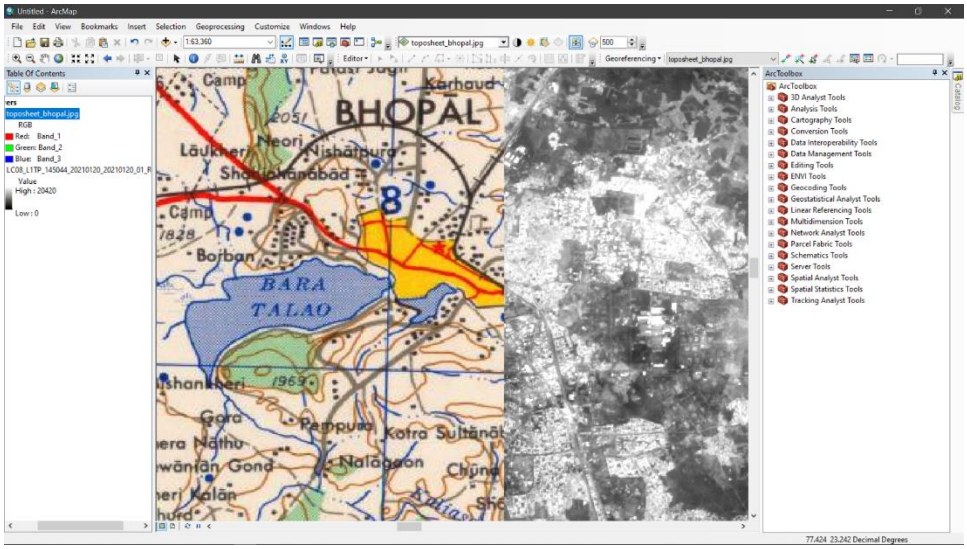


Fig. 14b: Cross-reference the presently georeferenced toposheet with the other one by sliding it sideways to check their mutual alignments

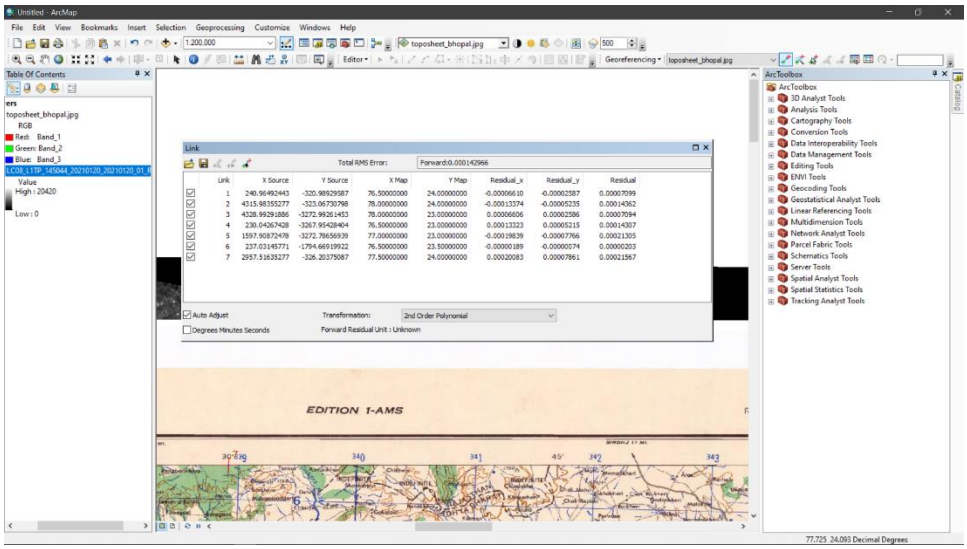


Fig. 15: **Open the link table, check the RMS error value and add more GCPs accordingly if needed.**

Georeferencing using Image to Image Registration:

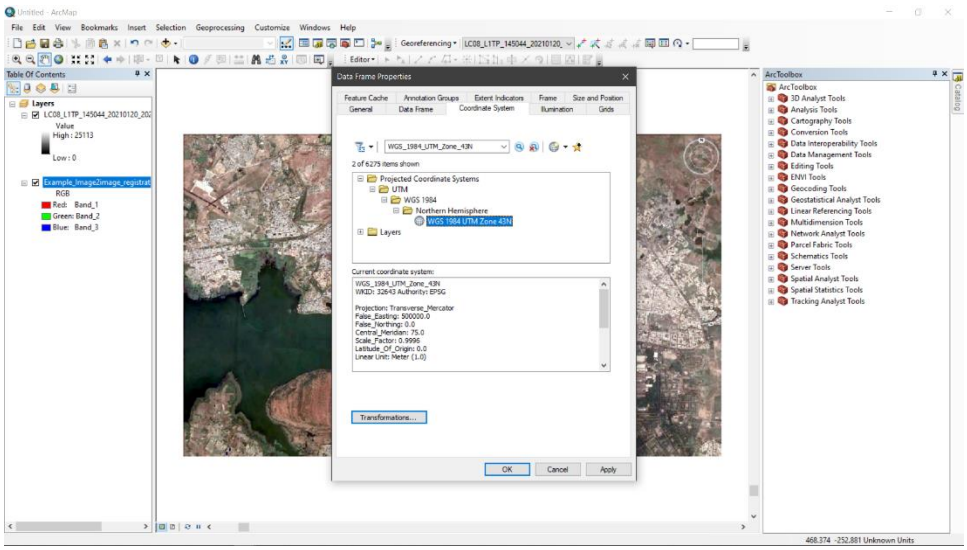


Fig. 16: **Follow the procedures shown in Fig. 1 to 5**

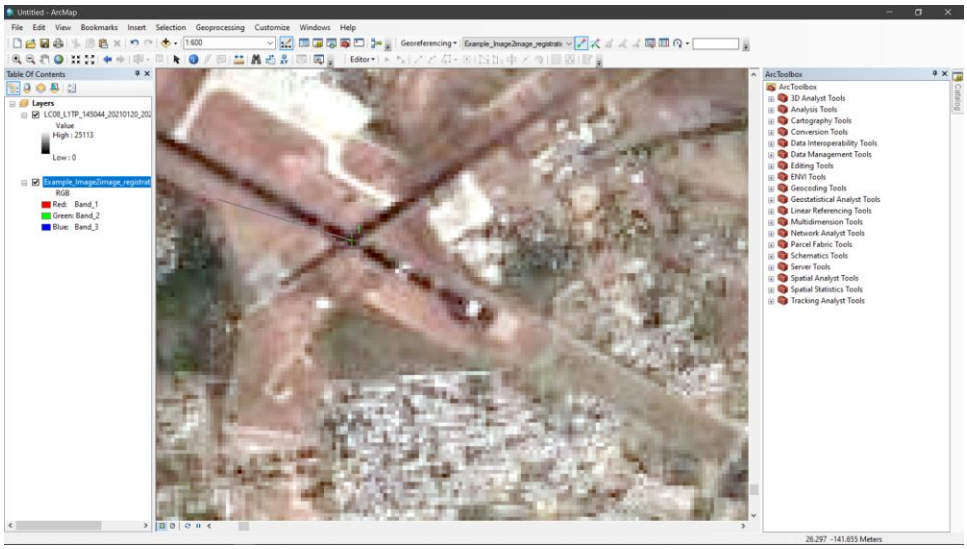


Fig. 17a: Plot the appropriate number of GCPs (according to the formula). Select those points on the toposheet as GCPs that can be identified in the other (already georeferenced) toposheet as well. To plot a GCP, click on the desired location...

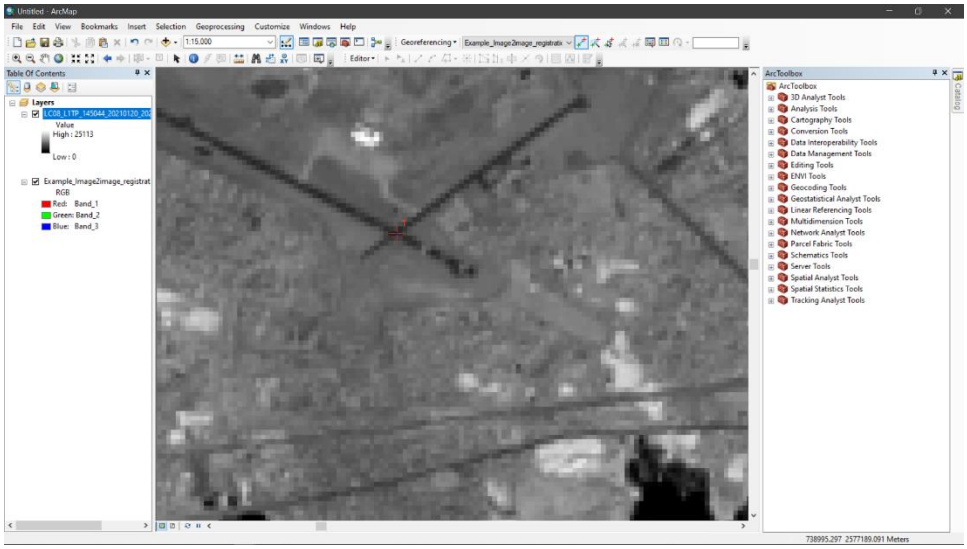


Fig. 17b: ...and trace the cursor onto the corresponding point on the other map. Follow the rest of the procedure in Fig. 8 to 13.

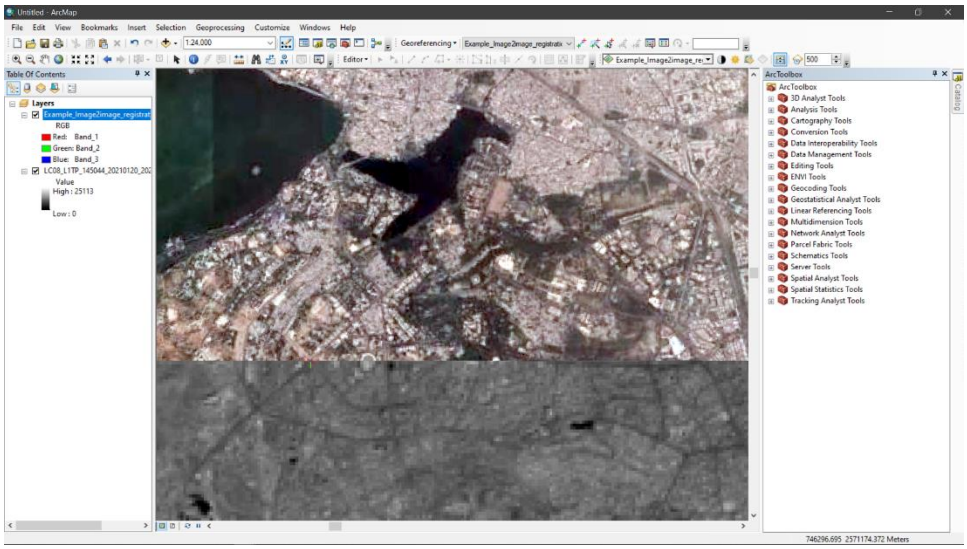


Fig. 18a: Follow the procedure in Fig. 14a

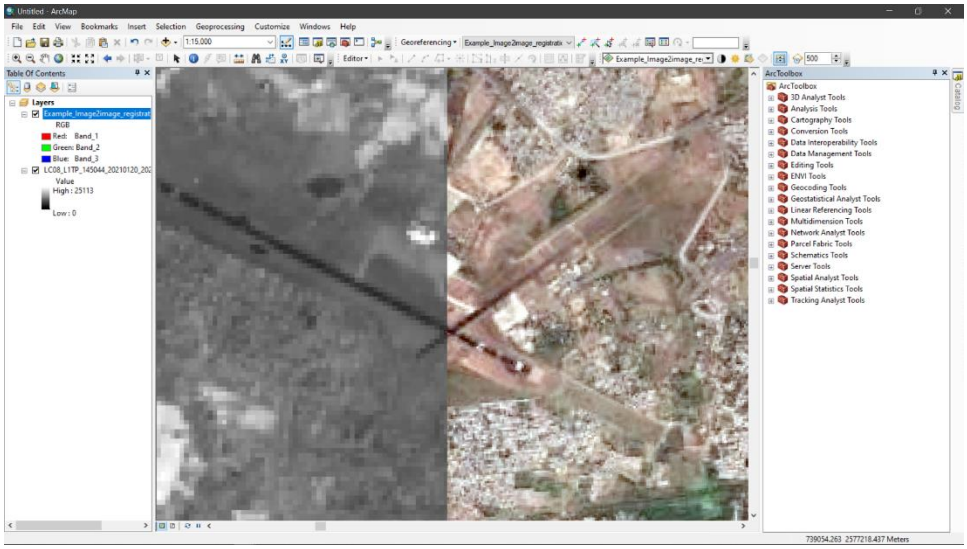


Fig. 18b: Follow the procedure in Fig. 14b

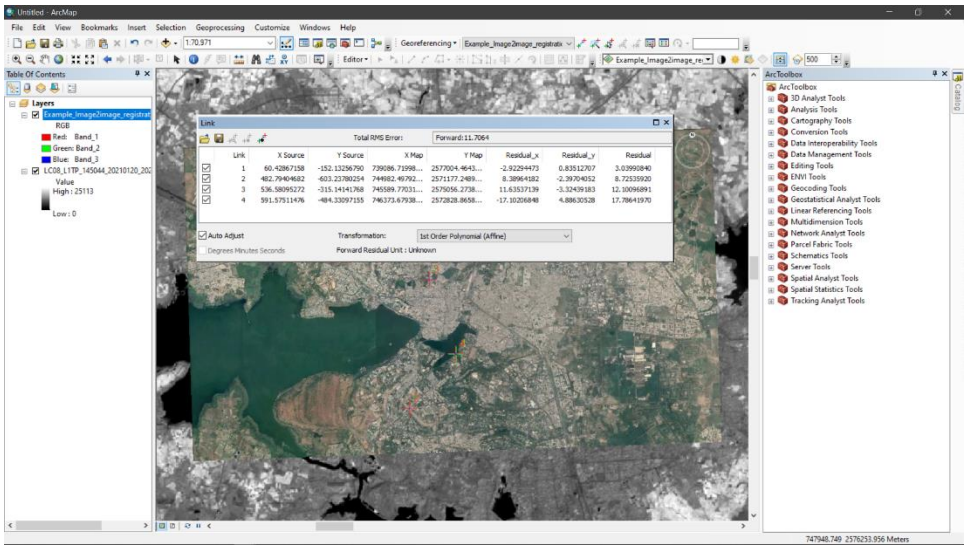


Fig. 19: **Follow the procedure in Fig. 15**

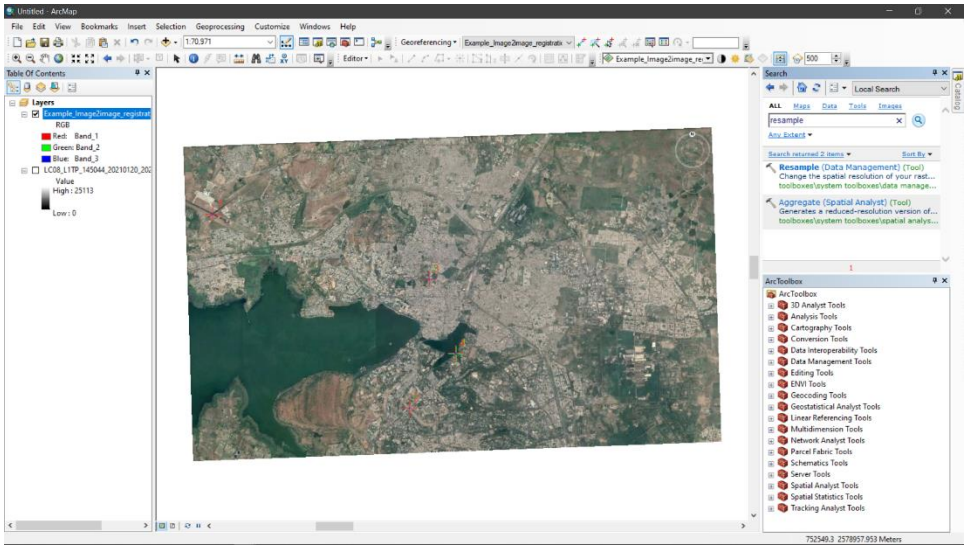


Fig. 20: **Result: Image to Image Registration completed**

Image Resampling:

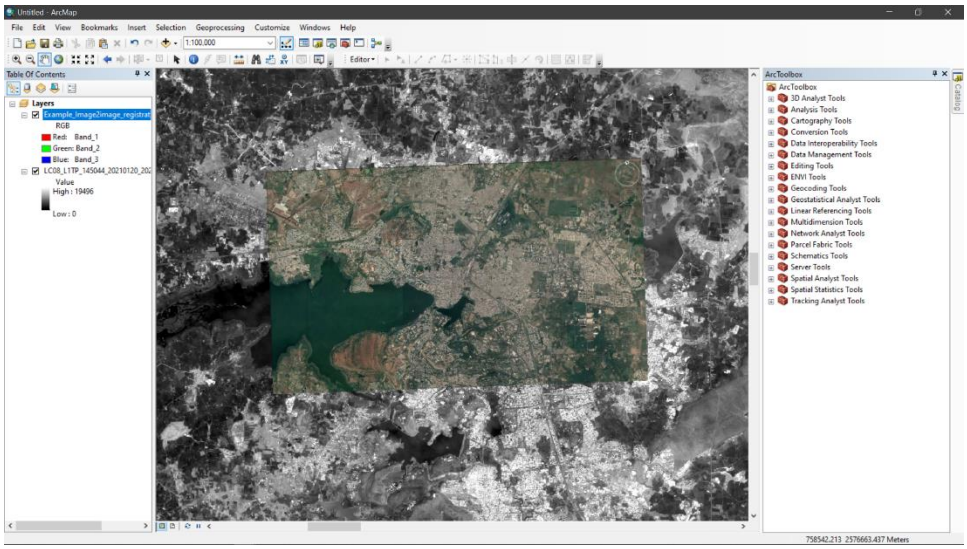


Fig. 21: **Enable both images (reference image and image to be resampled)**

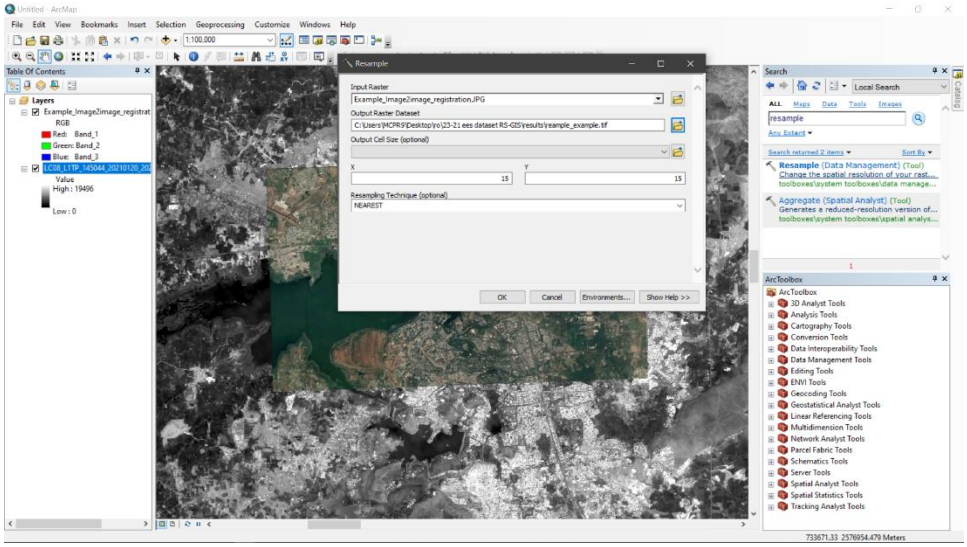


Fig. 22: **Open the resampling tool and enter the input & output files, the resampling resolution and the choice of technique**

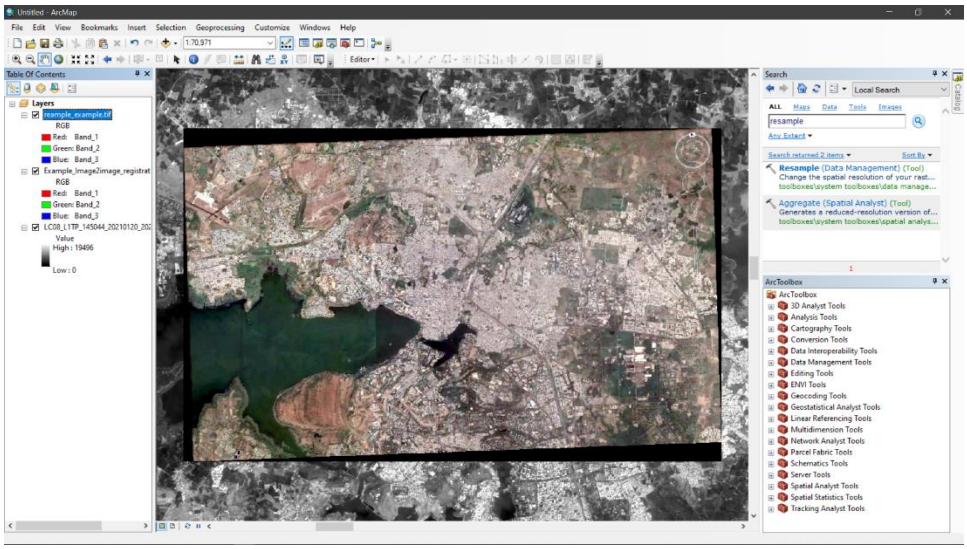
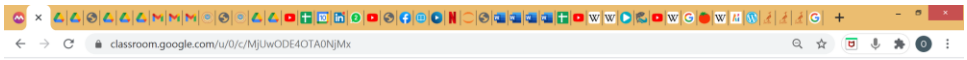


Fig. 23: **Result: Image resampling completed**

Task:



GCPs	Longitude	Latitude	Landmark
1	77.413963	23.267780	Railway station (Bhopal)
2	77.277303	23.280096	IISERB gate
3	77.409312	23.200665	Kaliyasot Dam (Spillway)
4	77.296580	23.217373	Electricity sub-station
5	77.259449	23.247806	Road Junction (Shivhare dhaba)
6	77.342874	23.202118	DPS school
7	77.335631	23.290907	Airport (Bhopal)

Fig. 24: **Task: Georeference the google earth image provided by geometric correction (using the above points as GCPs) as well as by image-to-image registration (using the already georeferenced image in class)**

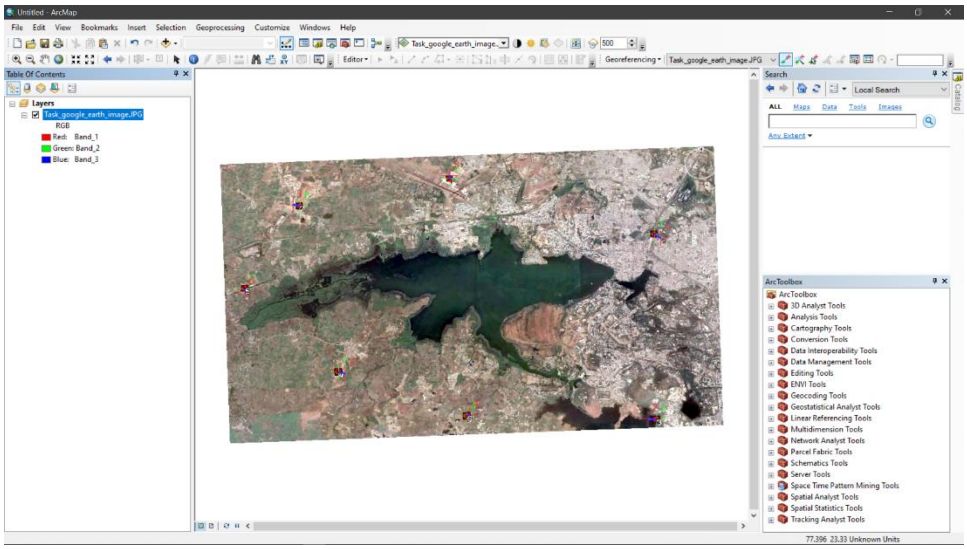


Fig. 25a: **Result: Georeferencing task by geometric correction completed**

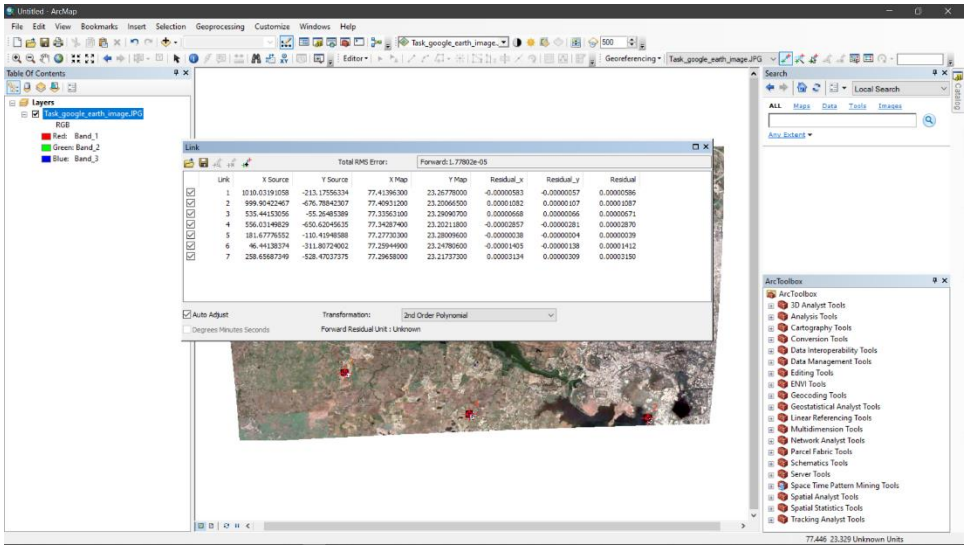


Fig. 25b: **Result: Georeferencing task by geometric correction completed**

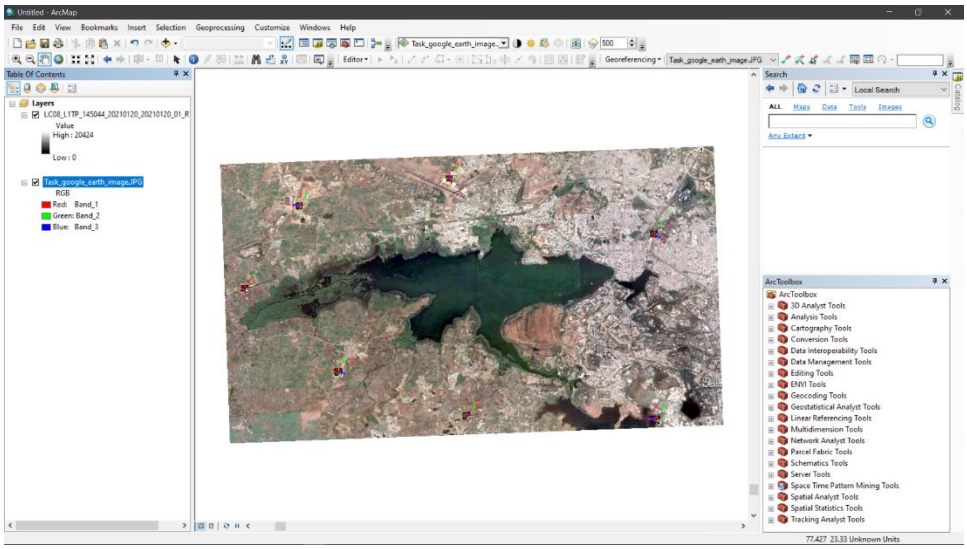


Fig. 25c: Result: Georeferencing task by image to image registration completed

Name: Om Mahesh Vaknalli

Roll No. 18376

Subject: EES-338 (Remote Sensing & GIS Lab)

Lab – VI (Report on Filters)

Applying Inbuilt Filters:

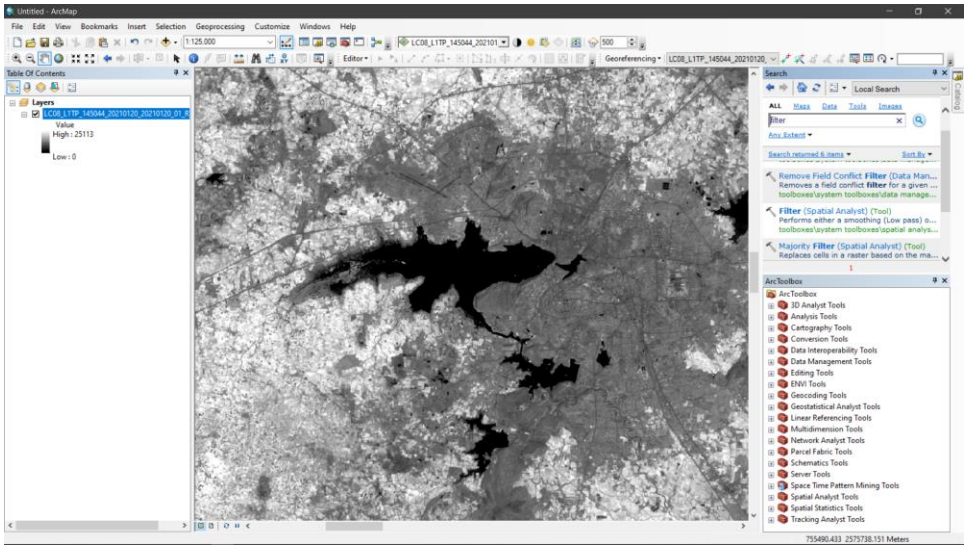


Fig. 1: Load the base image on ArcGIS on which all filter computations will be done

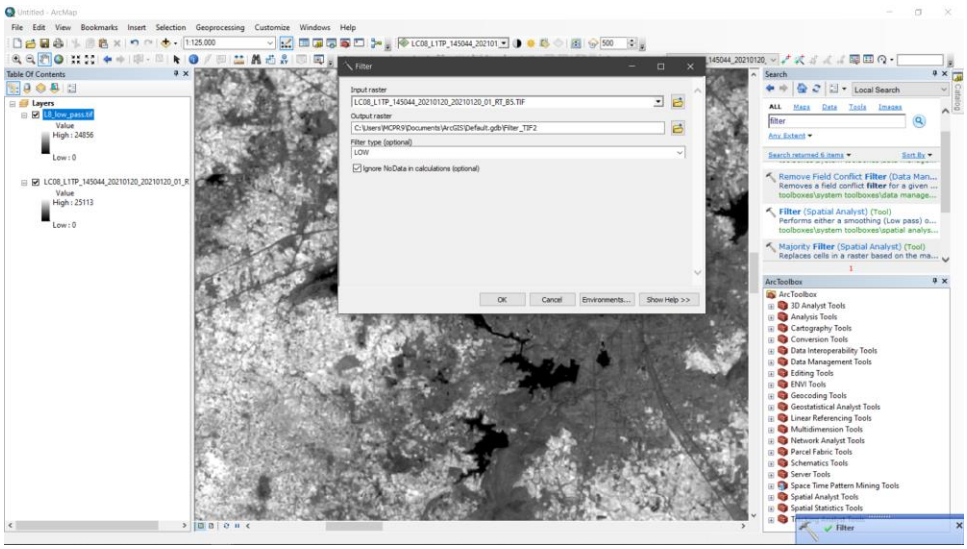


Fig. 2: Open the 'Filter' tool and input the base raster image along with the type of filter that has to be applied, then click 'OK'

Analysis of various filters:

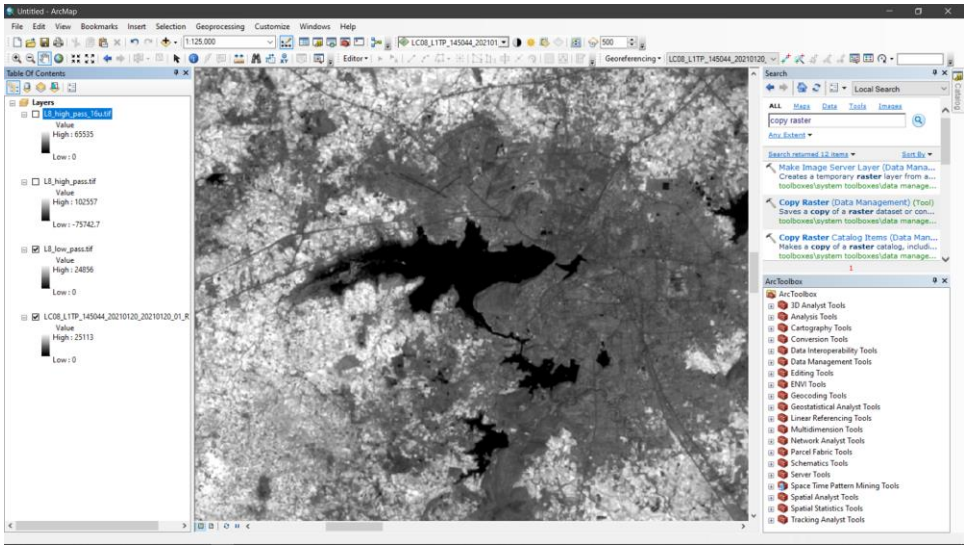


Fig. 3: Result: Application of 1x1 low pass filter

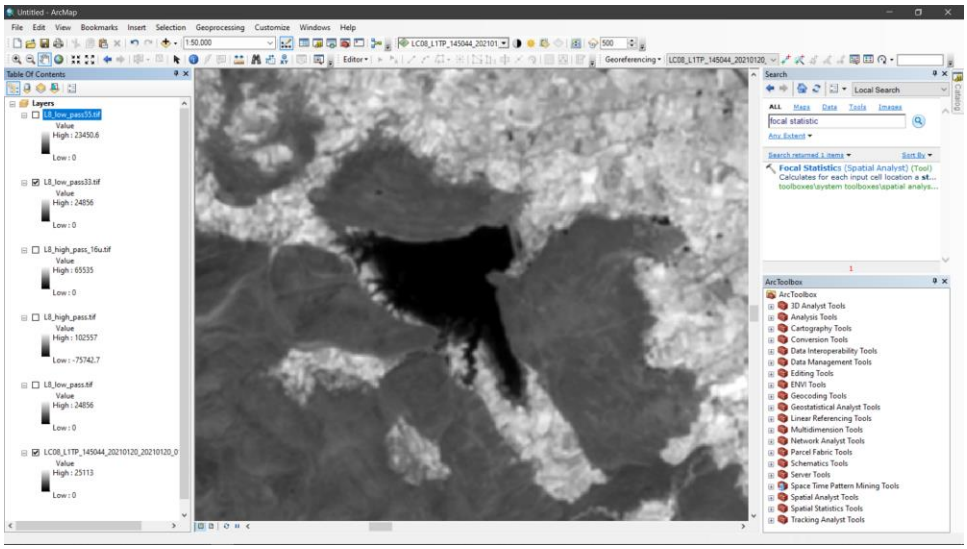


Fig. 4: Result: Application of 3x3 low pass filter (zoomed in image for better visualization)

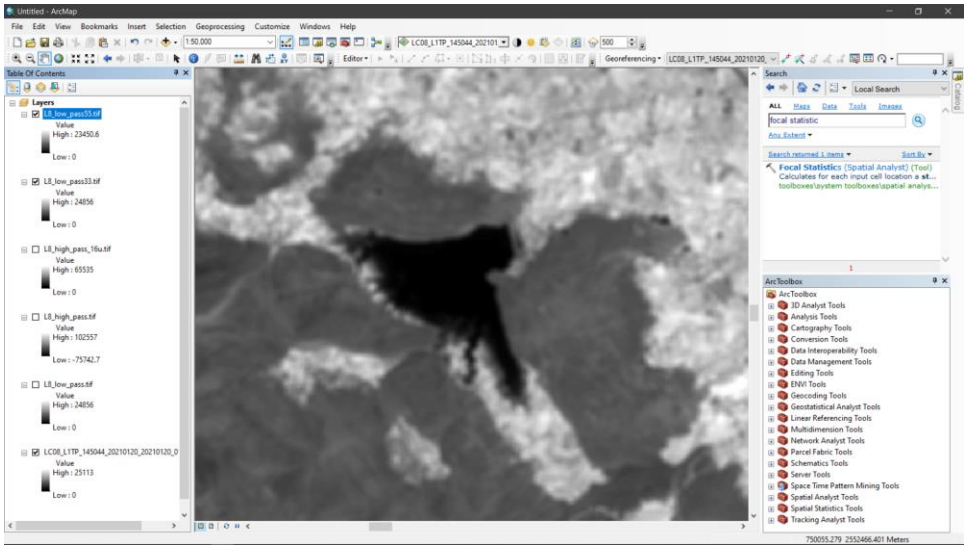


Fig. 5: Result: Application of 5x5 low pass filter (zoomed in image for better visualization)

Observations & Inferences: -

As observed in Fig. 3, the image blurriness has increased (when compared with the original image), but the smoothening of the image has also occurred. The edges of the image however, have been distorted (in their intensity values) thereby disrupting the image statistics.

In Fig. 4 and 5, the kernel size has been increased to 3x3 and 5x5 respectively. This has resulted in increased consecutive blurriness of the images (blurriness: 5x5 > 3x3 > 1x1). The same trend is noticed for image smoothening and edge distortions as well.

Kernel size modifications can be done with all kinds of filters and will show similar relative trends of blurriness, edge distortion and image smoothening as seen in the case of low pass filters.

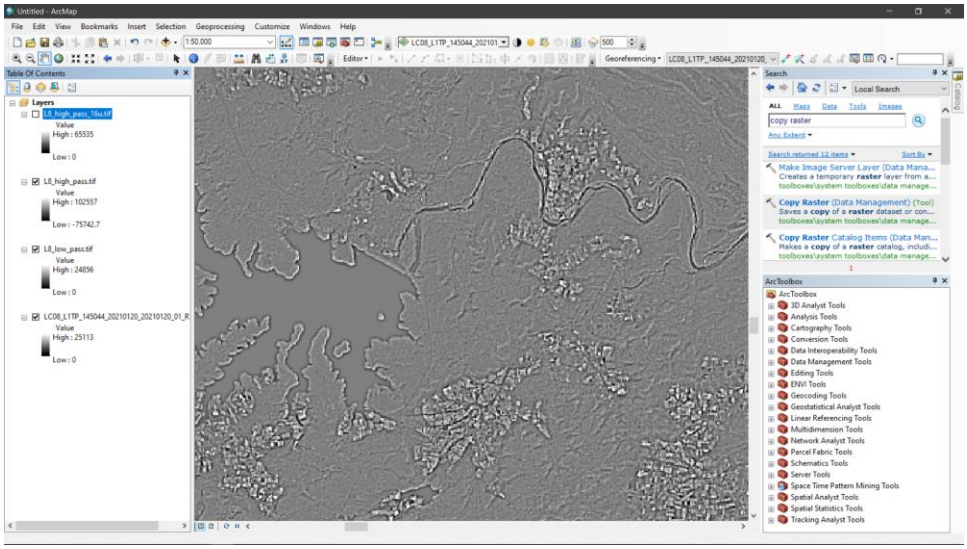


Fig. 6: **Result: Application of high pass filter**

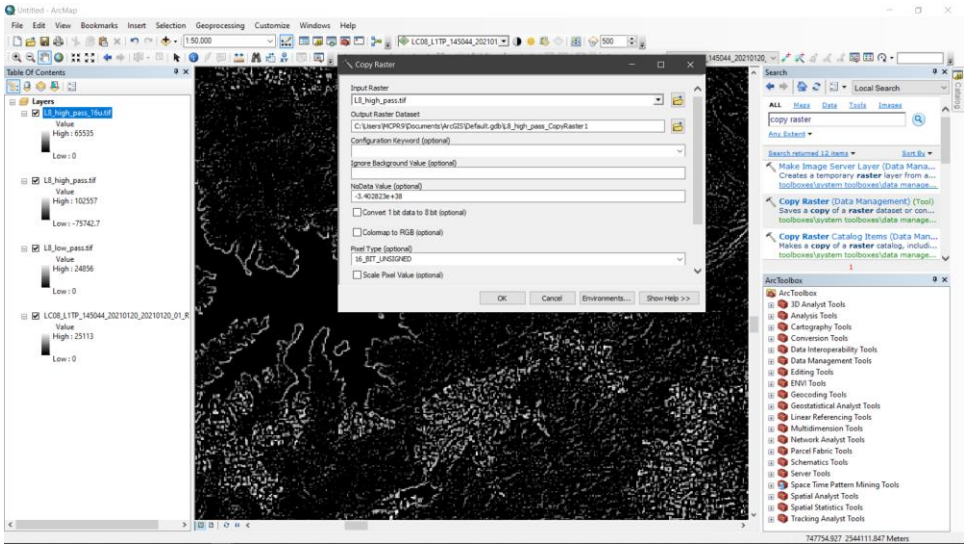


Fig. 7: **Open the 'Copy Raster' tool and enter the input raster, the output location and choose the pixel type as 'Unsigned' with the choice of bit size, then click 'OK'**

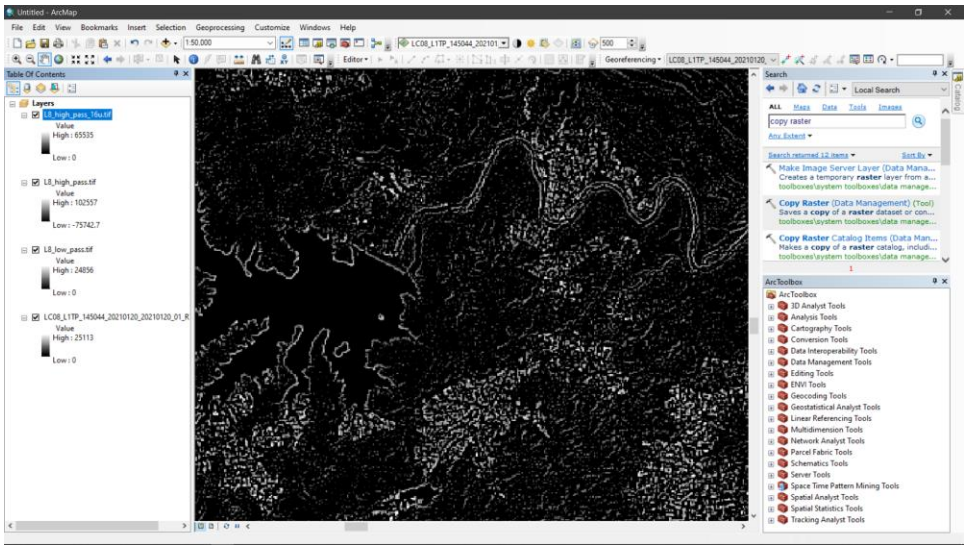


Fig. 8: Result: Unsigned high pass filter

Observations & Inferences: -

In Fig. 6, the sharpness has increased compared to the original image, while the edge of the image has been preserved.

In Fig. 7 & 8, the pixel type is changed to 'Unsigned Integer' since not all the intensity values of the image are positive, and they need to be converted to positive so that the edges are preserved further and can be seen in the results of Fig. 8.

Change of pixel type to Unsigned Integer can be done with all kinds of filters and will show a similar trend of edge preservation as seen in the case of high pass filters.

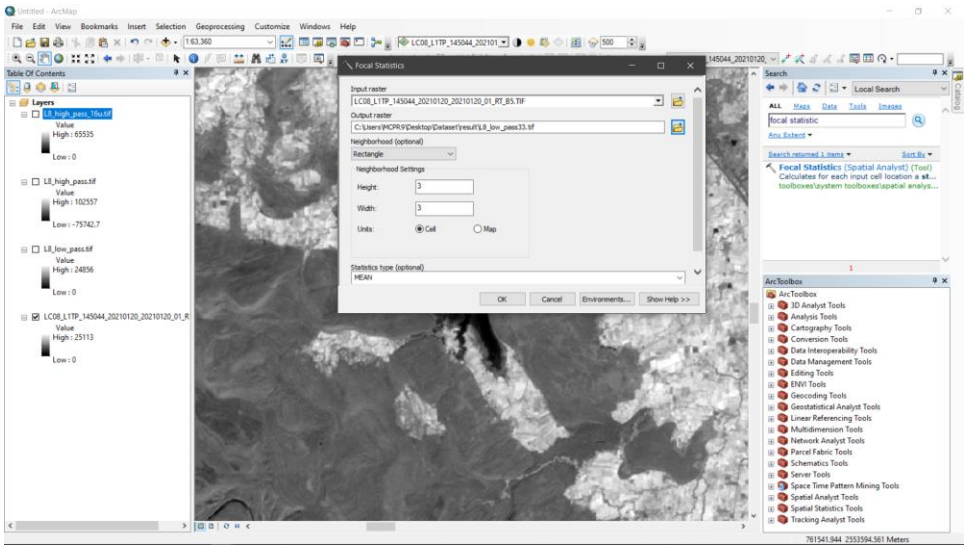


Fig. 9: **Apply the 'Focal Statistics' tool under the Spatial Analyst kit, give the input raster and the output file location as well as the preferred statistics type (mean in this case)**

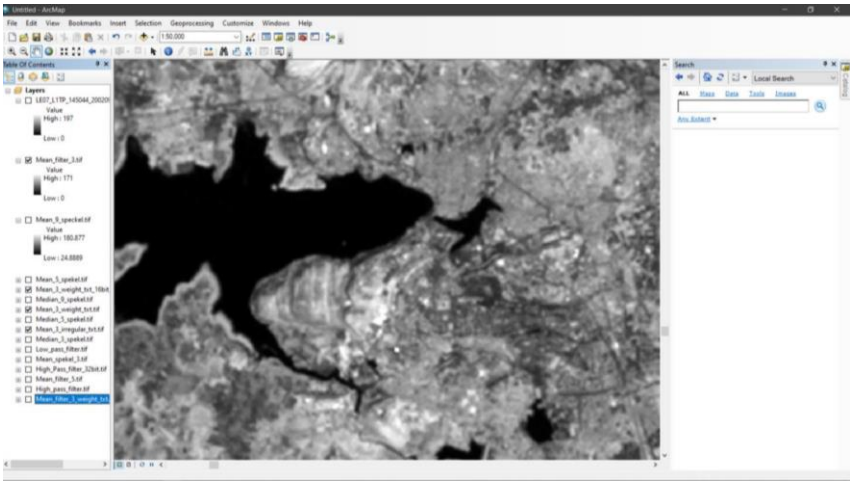


Fig. 10: **Result: Application of 3x3 mean filter**

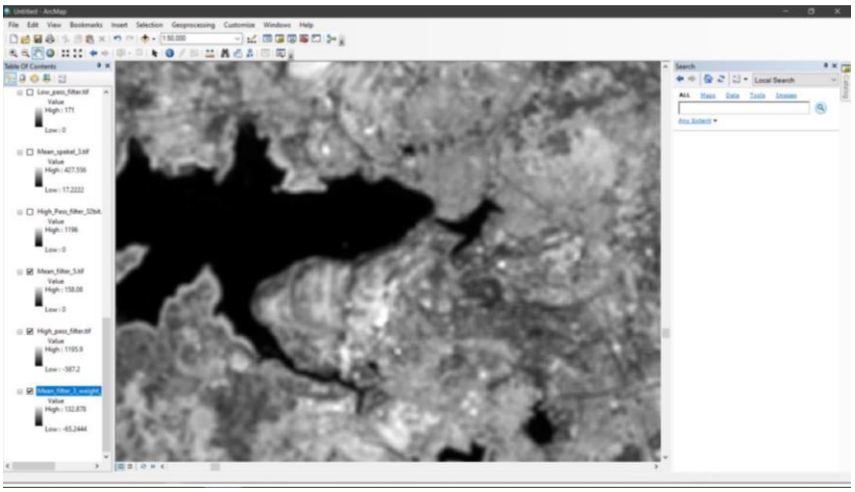


Fig. 11: **Result: Application of 5x5 mean filter**

Observations & Inferences: -

The image effects obtained by mean filters in Fig. 10 & 11 are similar to that of low pass filters. The effects of kernels too are similar to that as seen in case of low pass filters.

Median filters however, work similar to high pass filters in the sense that they preserve the edges while providing less blurriness compared to the mean filter and low pass filters.

Also unlike the mean and low pass filters, median filters showed that the edges were relatively more preserved with relatively less blurriness with an increase in kernel size.

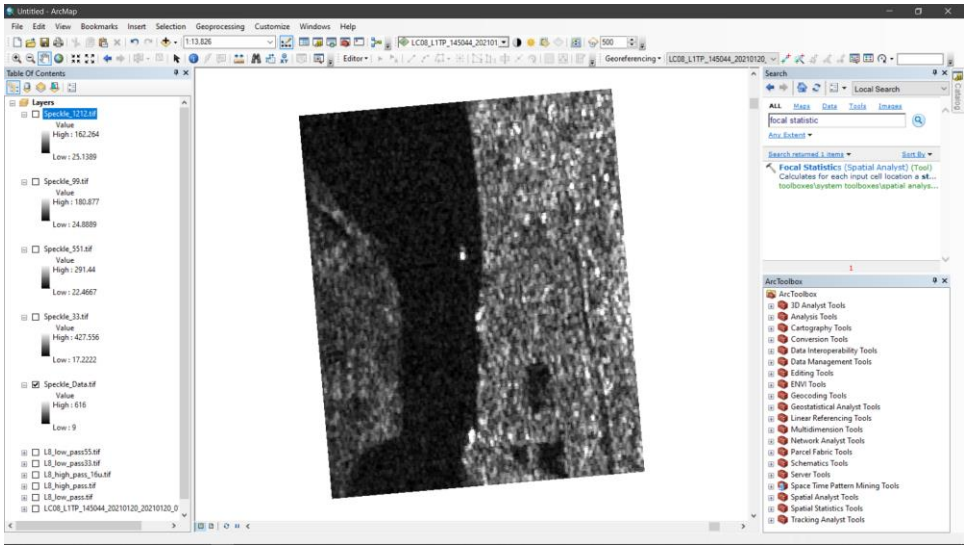


Fig. 12: Image with speckle noise

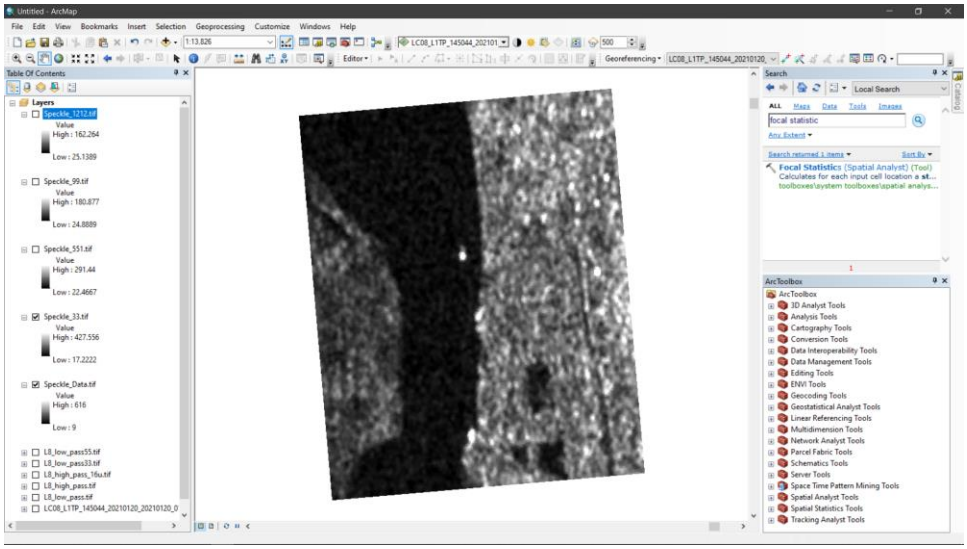


Fig. 13: Result: Speckle noise with a 3x3 filter

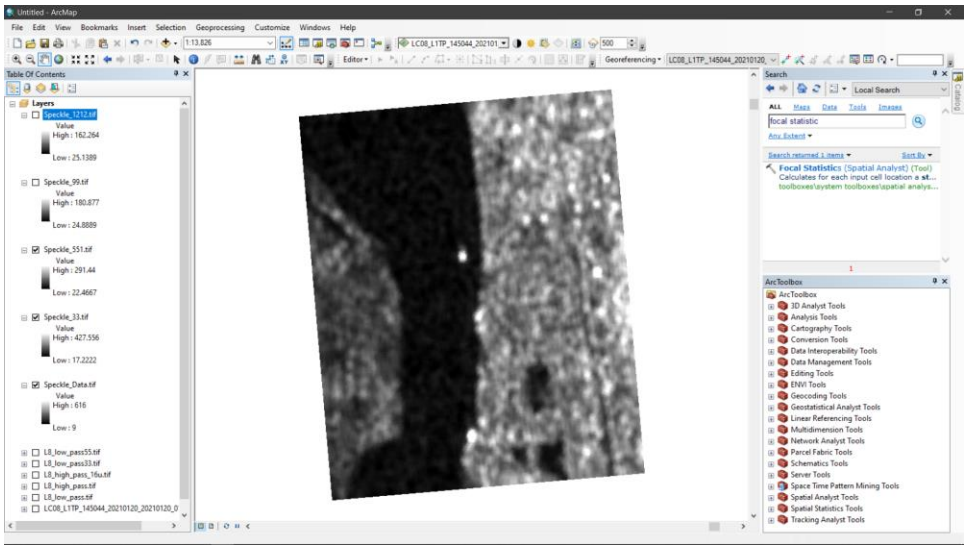


Fig. 14: **Result: Speckle noise with a 5x5 filter**

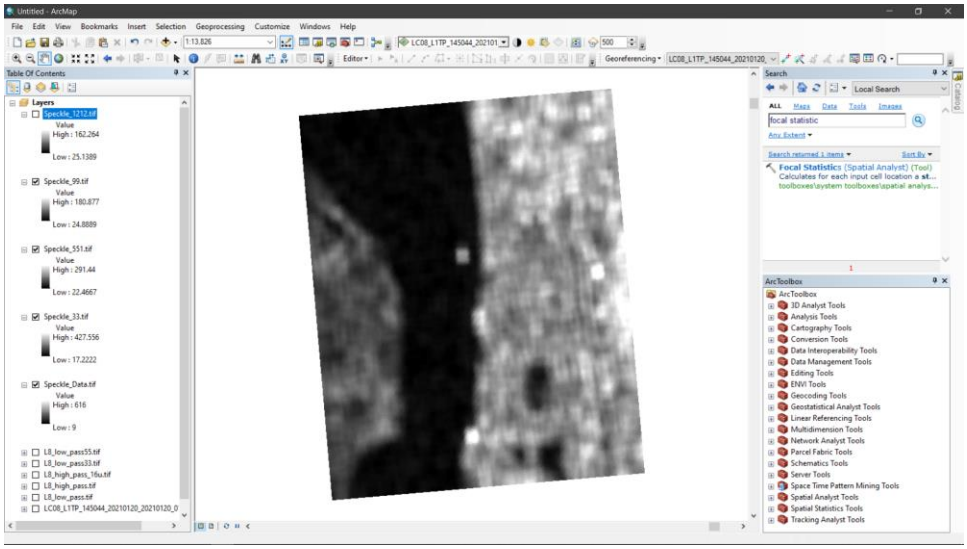


Fig. 15: **Result: Speckle noise with a 9x9 filter**

Observations & Inferences: -

In Fig. 13, 14 & 15, the speckle noise is consecutively reduced with increasing kernel size compared to the original noise in Fig. 12. This observation is irrespective of the type of filter applied to the image.

However, for a more focused inference, the above images were applied with a median filter which also highlighted its edge preserving abilities effectively. Although the edges were more distorted with increasing kernel sizes, their preservation with respect to other mean and low pass filters was much higher.

User Made (Manual) Filters:

Fig. 16: Kernel size and coefficients of filters (eg. mean filter in this case) should be saved in a text file

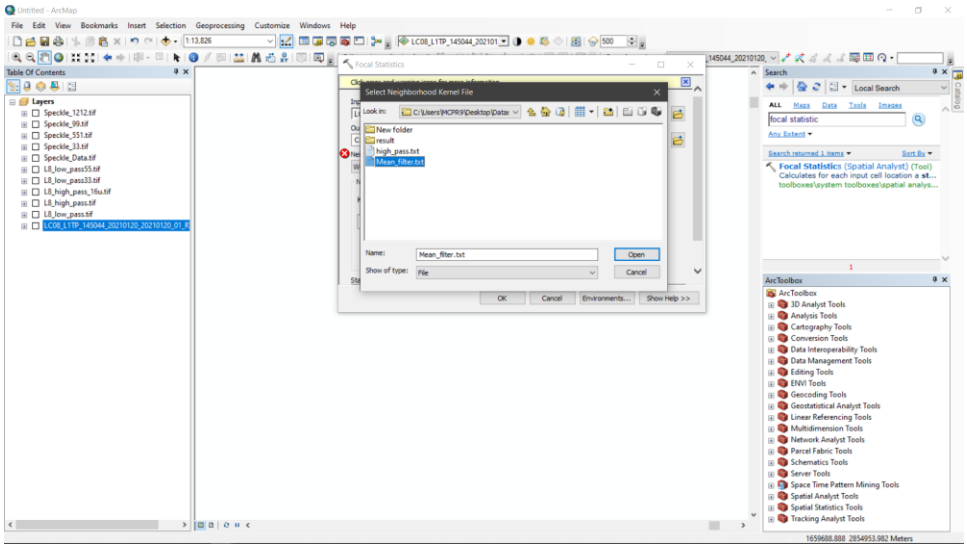


Fig. 17: Open the 'Focal Statistics' tool and input the raster and output file location, following which the neighbourhood should be selected as 'Weight' and the neighbourhood kernel file should be selected as the text file made in the previous step

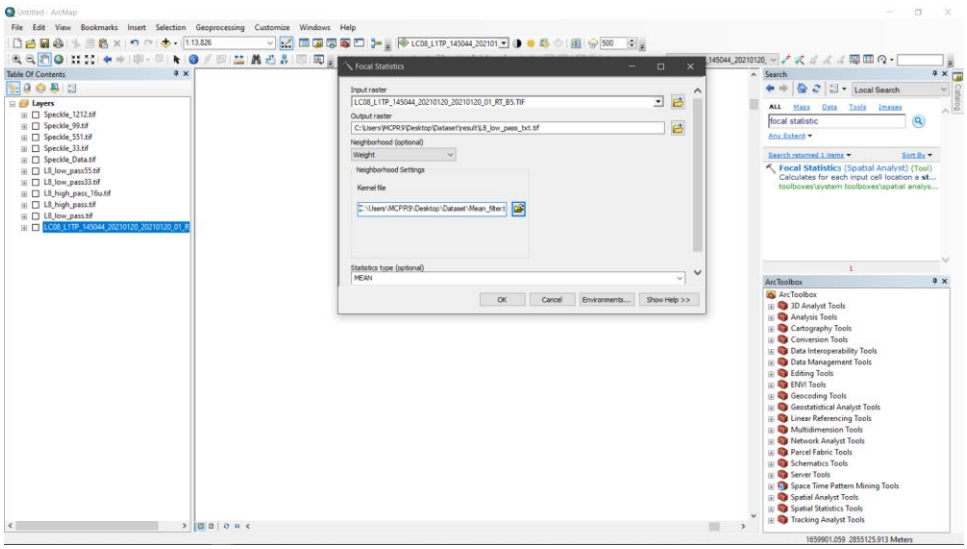


Fig. 18: **After clicking 'OK', select the desired statistics type and again click 'OK'**

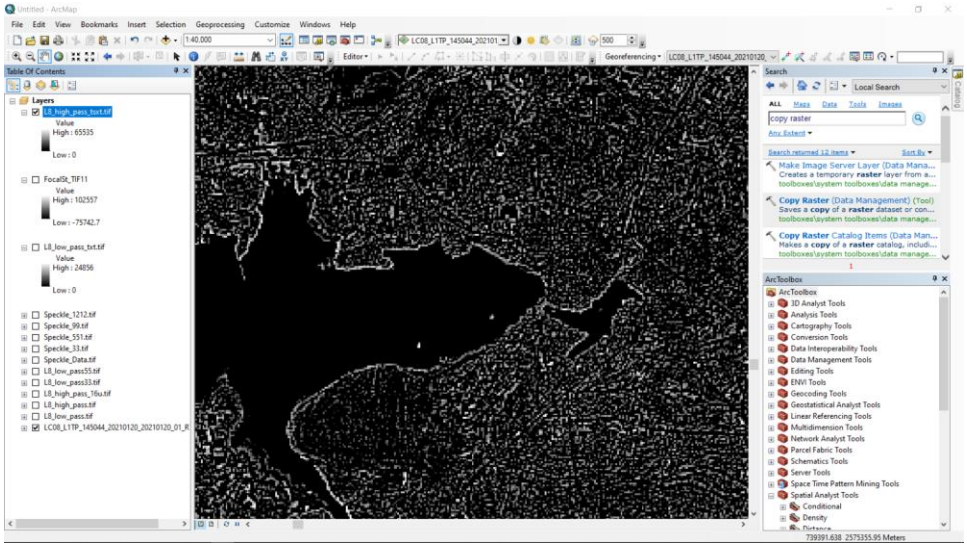


Fig. 19: **Result: Application of a user defined (manual) filter**

Observations & Inferences: -

The neighbourhood for the manual filters can be defined in 2 ways, (i) Weight and (ii) Irregular. Weight filter considers float values from the coefficients that are present while irregular filters consider these values from either 0 or 1 from the coefficients.

Weight filter along with the use of the 'sum' statistic type produces a filter that is identical to the inbuilt filters whereas the 'mean' statistic type will yield in a filter that is similar but not identical to an inbuilt filter with the difference that the manual filter will have different intensity values (and therefore intensity statistics) compared to an inbuilt filter as seen in Fig. 19.

Results: -

- **The procedure for applying inbuilt filters was successfully understood.**
- **The observations related to the application of various kinds of filters such as Low Pass, High Pass, Mean and Median filters were noted.**
- **The effects of changes in kernel size and changes in datatype of intensity values of the images was successfully observed.**
- **The influence of filter application on reduction of Speckle noise was discussed at length.**
- **The procedure to build user made or manual filters was effectively understood.**

Lab – VII (PCA and Convolution Theorem)

Principal Component Analysis (PCA):

```
clc
clear all

%%
img = uint16(imread('final_com.tif')); %Image date: 26-12-2020
FCC_before_PCA = cat(3, img(:,:,5), img(:,:,4), img(:,:,3)); %NIR(5) RED(4) GREEN(3)
%%
FCC_before_PCA = imlocalbrighten(FCC_before_PCA); %Histogam stretching
%%
subplot(221)
imshow((FCC_before_PCA))
title('FCC LANDSAT-8 (Before PCA)')
%%

[r c] = size(img(:,:,1)); %Preserving the shape
for i = 1:11
shape_img = reshape(img(:,:,i),[],1); %Converting each band into a column vector
shape_final_data(:,i)=shape_img;
end
%%
subplot(222)
nComp = 1; %input('Please enter the no. of PCs: \n')
shape_final_data = double(shape_final_data);
shape_final_data_m = mean(shape_final_data);
shape_final_data_m_adjusted = (shape_final_data - shape_final_data_m);
```

COMMAND WINDOW

```
[r c] = size(img(:,:,1)); %Preserving the shape
for i = 1:11
shape_img = reshape(img(:,:,i),[],1); %Converting each band into a column vector
shape_final_data(:,i)=shape_img;
end
%%
subplot(222)
nComp = 1; %input('Please enter the no. of PCs: \n')
shape_final_data = double(shape_final_data);
shape_final_data_m = mean(shape_final_data);
shape_final_data_m_adjusted = (shape_final_data - shape_final_data_m);
[coeff, score, latent, ~, explained] = pca(shape_final_data_m_adjusted);

%The next two statements forms the formula for reconstruction of the data
reconstruction_data = score(:, 1:nComp)*coeff(:, 1:nComp)';
final_reconstruction = (reconstruction_data + shape_final_data_m);
for i = 1:11
    Reconstructed_final = reshape(final_reconstruction(:, i), r, c);
    img_final(:, :, i) = Reconstructed_final
end
img_final = uint16(img_final);
FCC_after_PCA = cat(3, img_final(:, :, 5), img_final(:, :, 4), img_final(:, :, 3));
FCC_after_PCA = imlocalbrighten(FCC_after_PCA);
imshow((FCC_after_PCA))
title('FCC LANDSAT-8 (with PCs = 1, 2, & 3, Variance = 95%)')
```

COMMAND WINDOW

Fig. 1: MATLAB code to perform PCA

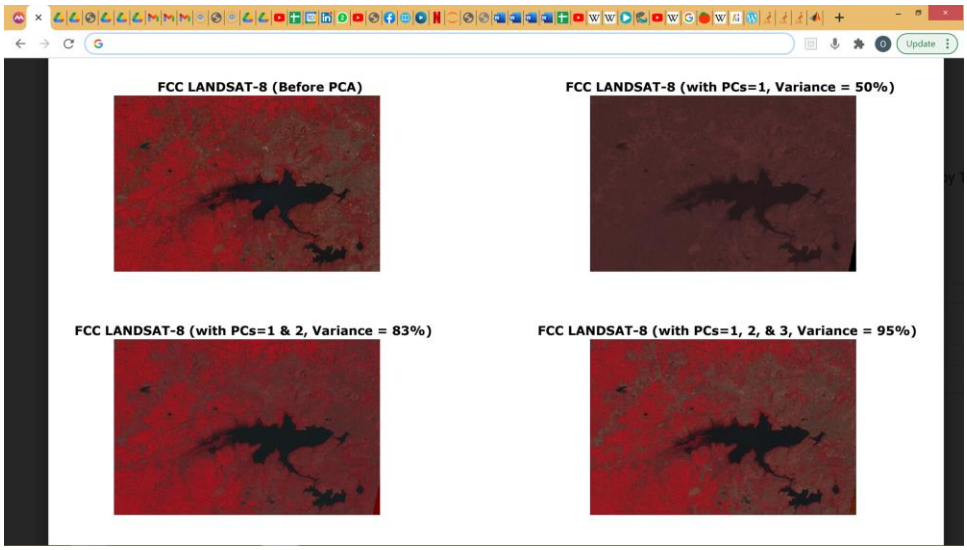


Fig. 2: The input and outputs (results) for PCA

Methodology, Observations & Inferences: -

Fig. 1 shows the MATLAB code used to perform PCA on any image. The working of the program is detailed as follows:

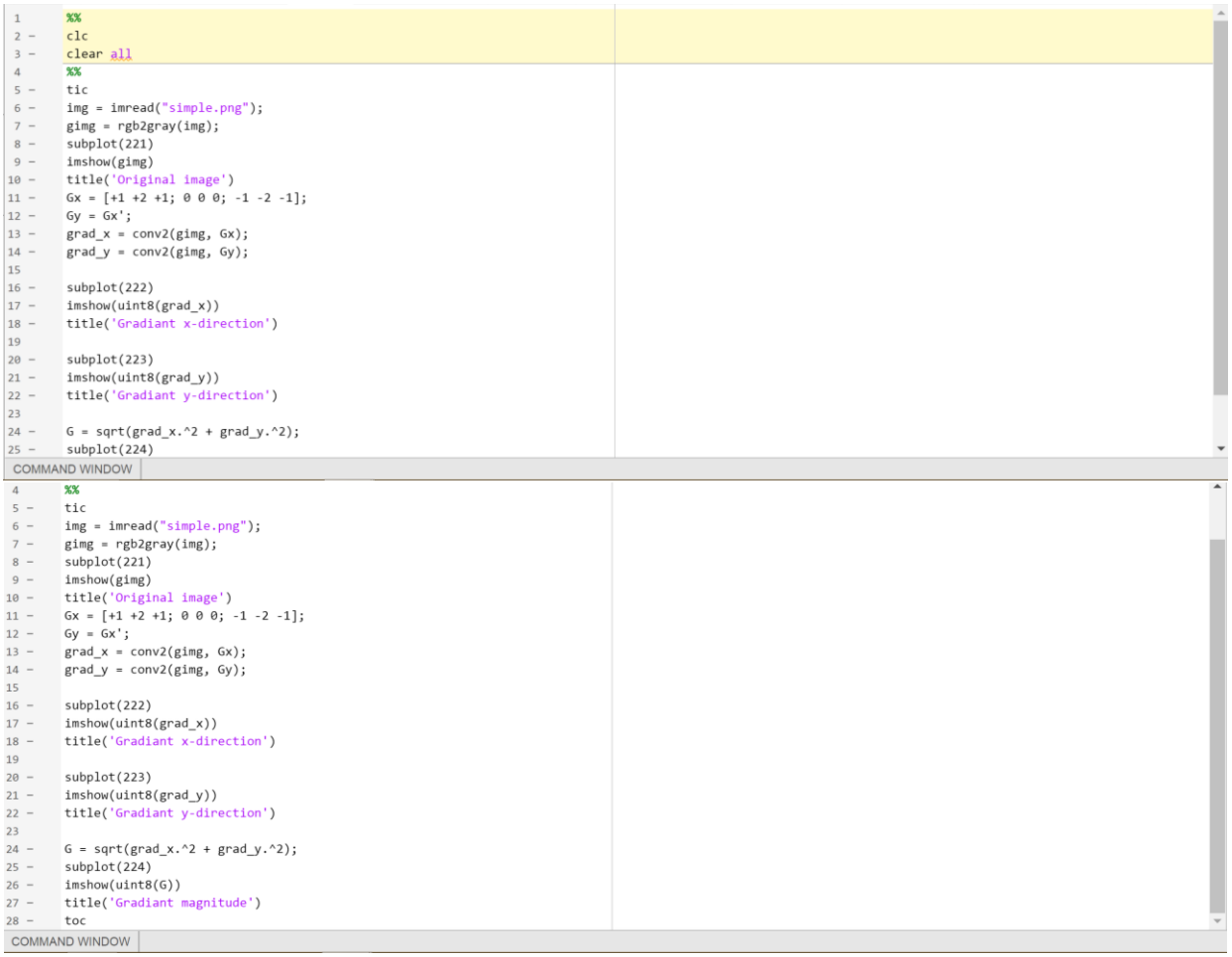
- ***Line 5*** **imports the data into MATLAB for processing. This data consists of a composite image of all the bands of a Landsat 8 file. The data is then converted into 16 bit unsigned integer datatype.**
- **In *line 6*, the information of the bands is resolved such that bands 3, 4 and 5 are grouped together to form an FCC of their own.**

- **This FCC is subjected to histogram stretching in *line 8* so that the individual normalised histograms of the respective bands can be effectively compared and computed upon.**
- ***Lines 10 - 12* plot this FCC.**
- ***Lines 15 – 19* now try to convert the FCC 3D data array into a 2D matrix with dimensions of bands x observations. This is done by arranging the bands into column vectors and then stacking them together. However, before the rearranging of the data, a mechanism is set up to preserve the shape of the original data file which will be used in a later step.**
- ***Line 22* decides the no. of PCs to assign for the analysis.**
- ***Lines 23 – 25* convert the data into 'double' datatype, then take its mean and finally subtract individual values from the mean to obtain a normalised dataset. This also helps to bring all the different variables to the same platform or range.**
- ***Line 26* extracted the meaningful terms such as latent, explained, score, coeff and ~ using the inbuilt MATLAB PCA command. These are useful terms in computing the PCs further.**
- **The reconstruction of the data into its original shape is done in *lines 29 – 34*. Note that this data also incorporates the recently found PCs.**
- **The modified data is transformed into the unsigned integer datatype format in *line 35*.**
- **The FCC is reconstructed back from this data using the same 3 bands but in their modified state according to the *lines 35 – 39*. The process also retraces the steps of histogram stretching and plotting of the results.**
- **Lines 22 – 39 should be repeated more times with varying the number of PCs and variance to be analysed.**

Fig. 2 discusses the results obtained from the above procedure. The result constructed by 1 PC gathers most of the information of the original image with a 50% variance. The result with 2 PCs gives an even more accurate representation of the original image, but the difference in variance between this image and the one obtained through only 1 PC is lesser when compared with the variance difference between only PC 1 image and the original image. Similarly, when using 3 PCs, the variance difference is the least among the 3 comparisons, even though it gives the closest representation of the original image. It effectively conveys that the higher order PCs contribute lesser and lesser to the overall variance of the original image compared to the initial ones. However, with that being said, the results with initial PCs have a smaller file size compared to the other results or the original image. Thus, this main motive of this exercise i.e. size reduction through PCA is satisfied.

The PCA obtained through programming is not the same as the PCA that is manually done. The programmed PCA is more accurate and provides a reversed sign compared to the manual PCA. The programmed PCA parameters can also be translated into meaningful, physically significant terms of the manual PCA. Secondly, the manual PCA uses Eigen Value Decomposition (EVD) while the built-in MATLAB PCA uses Single Value Decomposition (SVD). In a general sense, SVD is broader and is more compatible to perform tasks which EVD cannot.

Verification of Convolution Theorem (Tasks 1 & 2):



```
%%
clc
clear all
%%
tic
img = imread("simple.png");
gimg = rgb2gray(img);
subplot(221)
imshow(gimg)
title('Original image')
Gx = [+1 +2 +1; 0 0 0; -1 -2 -1];
Gy = Gx';
grad_x = conv2(gimg, Gx);
grad_y = conv2(gimg, Gy);

subplot(222)
imshow(uint8(grad_x))
title('Gradiant x-direction')

subplot(223)
imshow(uint8(grad_y))
title('Gradiant y-direction')

G = sqrt(grad_x.^2 + grad_y.^2);
subplot(224)
```

COMMAND WINDOW

```
%%
tic
img = imread("simple.png");
gimg = rgb2gray(img);
subplot(221)
imshow(gimg)
title('Original image')
Gx = [+1 +2 +1; 0 0 0; -1 -2 -1];
Gy = Gx';
grad_x = conv2(gimg, Gx);
grad_y = conv2(gimg, Gy);

subplot(222)
imshow(uint8(grad_x))
title('Gradiant x-direction')

subplot(223)
imshow(uint8(grad_y))
title('Gradiant y-direction')

G = sqrt(grad_x.^2 + grad_y.^2);
subplot(224)
imshow(uint8(G))
title('Gradiant magnitude')
toc
```

COMMAND WINDOW

Fig. 3: MATLAB code to verify the Convolution Theorem (spatial domain part)

```
%%
clc
clear all
%%
tic
img = imread("simple.png");
gimg = rgb2gray(img);
subplot(221)
imshow(gimg)
title('Original image')
Gx = [+1 +2 +1; 0 0 0; -1 -2 -1];
Gy = Gx';
PQ = (size(gimg));
F = fft2(double(gimg), PQ(1), PQ(2));
H = fft2(double(Gx), PQ(1), (PQ(2)));
F_fh = H.*F;
ffix = ifft2(F_fH);

%Display results (show all values)
imshow(uint8(ffix))
title('Frequency x')
%%
subplot(223)

F = fft2(double(gimg), PQ(1), PQ(2));
```

COMMAND WINDOW

```
F_fh = H.*F;
ffix = ifft2(F_fH);

%Display results (show all values)
imshow(uint8(ffix))
title('Frequency x')
%%
subplot(223)

F = fft2(double(gimg), PQ(1), PQ(2));
%F = fft2(double(f));
H = fft2(double(Gy), PQ(1), PQ(2));
%H = fft2(double(H));
F_fH = H.*F;
ffiy = ifft2(F_fH);

%Display results (show all values)
imshow(uint8(ffiy))
title('Frequency y')
%%
G = sqrt(iffx.^2 + iffy.^2);
subplot(224)
imshow(uint8(G))
title('Frequency Domain')
toc
```

COMMAND WINDOW

Fig. 4: MATLAB code to verify the Convolution Theorem (frequency domain part)

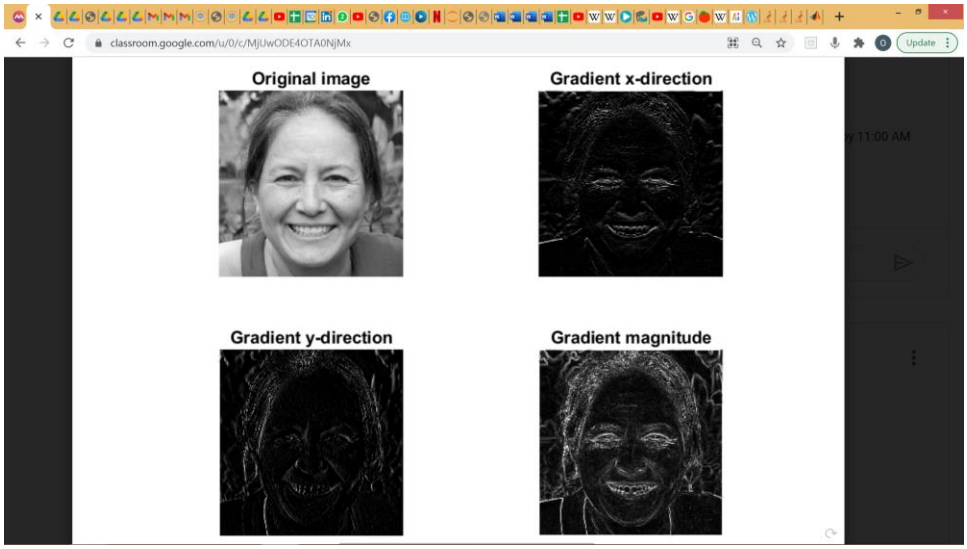


Fig. 5: The input and outputs (results) for convolution in the spatial domain

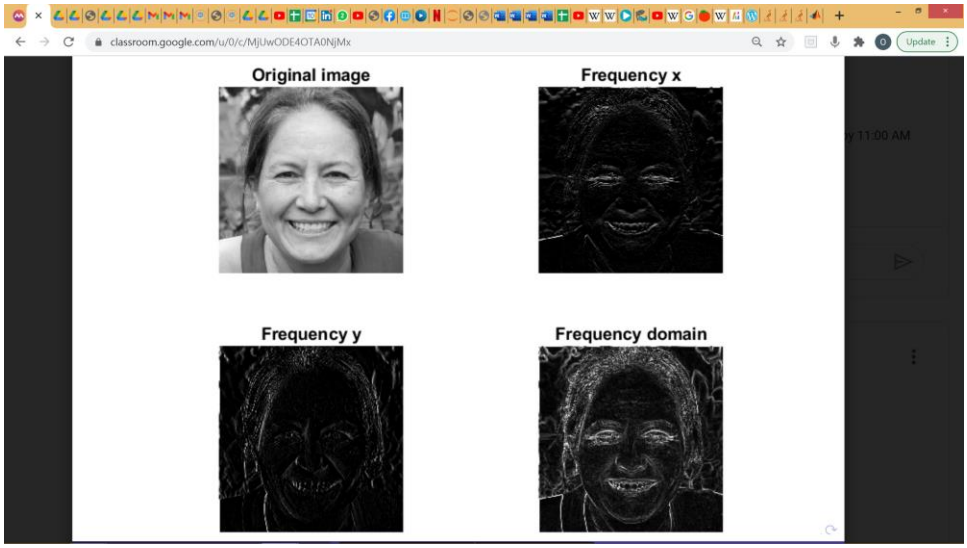


Fig. 6: **The input and outputs (results) for multiplication in the Fourier space**

Methodology, Observations & Inferences: -

Consider the detailed analysis of the spatial domain code shown in Fig. 3:

- **The 'tic' - 'toc' function in *lines 5 & 28* indicate the running time of the program**
- ***Line 6* imports the input image.**
- ***Line 7* converts to colour image into a grayscale image using an inbuilt MATLAB function.**
- **The plotting of this grayscale image is done in *line 8 – 10*.**
- ***Lines 11 – 12* define the gradient matrix (filter) in the x and y direction which result from the convolution of the grayscale image.**
- **The convolution functions are obtained in the lines 13 – 14 by using the inbuilt MATLAB 'conv()' function. These output functions are resolved in the x and y directions and saved in 'grad_x' and 'grad_y' variables respectively.**
- **These gradients are then plotted ass illustrated in *lines 16 – 22*.**
- **Finally, the gradient magnitude is also plotted according to *lines 24 – 27*.**

Now consider the frequency domain counterpart of this code shown in Fig. 4:

- ***Lines 5 – 12* exactly imitate the program of the spatial domain.**
- ***Line 13* finds the dimensions of the grayscale image.**
- **The conversion of the input image and the x-directional gradient matrix (filter) into its frequency domain counterpart is carried out in *lines 14 – 15* (the fft() function is used for Discrete Fourier Transformation (DFT)), which are then multiplied in *line 16*. This multiplication product is then reconverted into its spatial domain counterpart in *line 17* (the ifft() function is used for Inverse Discrete Fourier Transform (IDFT)).**

- **Lines 19 – 21 then plot this multiplication product.**
- **The process in the above 2 points in repeated for the y-direction in *lines 23 – 34*.**
- **Like in the spatial domain, the frequency domain magnitude of the multiplication product is obtained in the *lines 36 – 39*.**

By observing Fig. 5 & 6, it is clear that both these figures are the same, i.e., the convolution in the spatial domain is equal to the multiplication in the Fourier space in the frequency domain. Thus, the Convolution Theorem is successfully verified.

Ideal, Butterworth and Gaussian Low & High Pass Filters (Tasks 3 & 4):

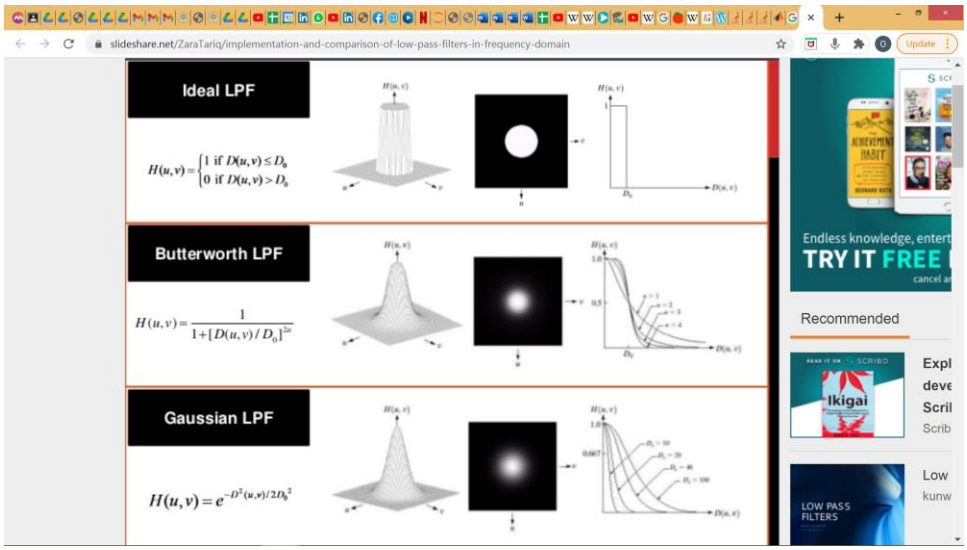


Fig. 7: Differences between Ideal, Butterworth and Gaussian LPFs. These similar principles are applicable for HPFs as well.

```
%%
clc
clear all

%%
img = imread('Fake_AI_Generated_Image.png');
gimg = rgb2gray(img);
%gimg = zeros(256, 256)
%gimg(125:126, 125:126) = 1;
imshow(gimg)

%% Ideal low pass
P = size(gimg);
M = P(1); N = P(2);
F = fft2(gimg, M, N);
subplot(332)
%imshow(uint8(abs(fftshift(F))));
imshow(uint8(abs((F))));
u0 = 100; %remove freq. greater than u0
u = 0: (M-1);
v = 0: (N-1);
idx = find(u > M/2);
u(idx) = u(idx) - M;
idy = find(v > N/2);
v(idy) = v(idy) - N;
```

COMMAND WINDOW

```
v(idy) = v(idy) - N;
[V, U] = meshgrid(v, u);
D = sqrt(U.^2 + V.^2);
H = double(D <= u0);
%display
subplot(333)
imshow(abs(fftshift(H)), []);
subplot(334)
G = H.*F;
g = (ifft2(G));
imshow(uint8(g))

%% Butterworth low pass
n = 1; %Butterworth filter of order n
H = 1./(1 + (D./u0).^(2*n));
subplot(335)
imshow(abs(fftshift(H)));
subplot(336)
G = H.*F;
g = (ifft2(G));
imshow(uint8(g))

%% Gaussian low pass filter
H = exp(-(D.^2)./(2*(u0^2)));
subplot(337)
```

COMMAND WINDOW

```
g = (ifft2(G));
imshow(uint8(g))

%% Butterworth low pass
n = 1; %Butterworth filter of order n
H = 1./(1 + (D./u0).^(2*n));
subplot(335)
imshow(abs(fftshift(H)));
subplot(336)
G = H.*F;
g = (ifft2(G));
imshow(uint8(g))

%% Gaussian low pass filter
H = exp(-(D.^2)./(2*(u0^2)));
subplot(337)
imshow(abs(fftshift(H)));
subplot(338)
G = H.*F;
g = (ifft2(G));
imshow(uint8(g))

% For high pass filter exercise, do H = 1 - given function

```

COMMAND WINDOW

Fig. 8: MATLAB code to apply the Ideal, Butterworth and Gaussian low pass filters

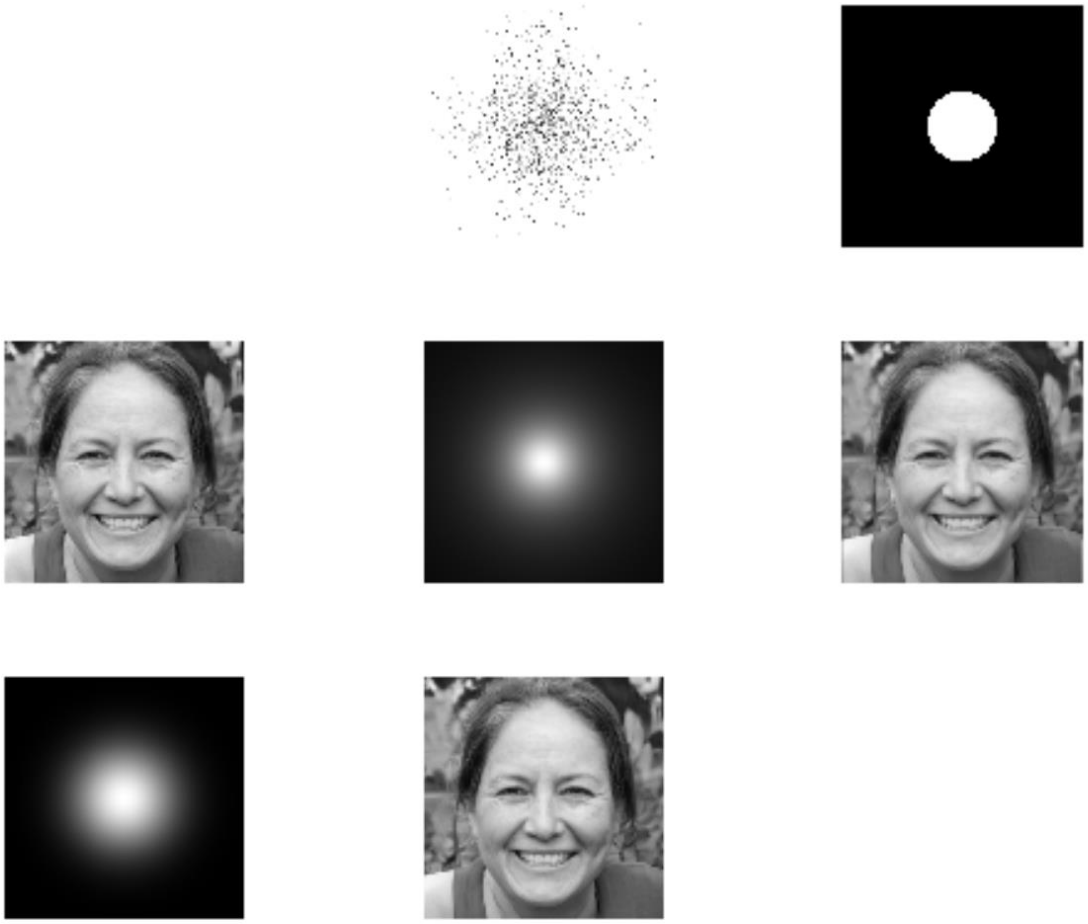


Fig. 9: **Results: Ideal, Butterworth and Gaussian Low Pass Filters**

```
%%
clc
clear all

%%
img = imread('Fake_AI_Generated_Image.png');
gimg = rgb2gray(img);
%gimg = zeros(256, 256)
%gimg(125:126, 125:126) = 1;
imshow(gimg)

%% Ideal high pass
P = size(gimg);
M = P(1); N = P(2);
F = fft2(gimg, M, N);
subplot(332)
%imshow(uint8(abs(fftshift(F))));
imshow(uint8(abs((F))));
u0 = 100; %remove freq. greater than u0
u = 0: (M-1);
v = 0: (N-1);
idx = find(u > M/2);
u(idx) = u(idx) - M;
idy = find(v > N/2);
v(idy) = v(idy) - N;
```

COMMAND WINDOW

```
v(idy) = v(idy) - N;
[V, U] = meshgrid(v, u);
D = sqrt(U.^2 + V.^2);
H = 1 - double(D <= u0);
%display
subplot(333)
imshow(abs(fftshift(H)), []);
subplot(334)
G = H.*F;
g = (ifft2(G));
imshow(uint8(g))

%% Butterworth high pass
n = 1; %Butterworth filter of order n
H = 1 - (1./(1 + (D./u0).^(2*n)));
subplot(335)
imshow(abs(fftshift(H)));
subplot(336)
G = H.*F;
g = (ifft2(G));
imshow(uint8(g))

%% Gaussian high pass filter
H = 1 - exp(-(D.^2)./(2*(u0^2)));
subplot(337)
```

COMMAND WINDOW

```
subplot(333)
imshow(abs(fftshift(H)), []);
subplot(334)
G = H.*F;
g = (ifft2(G));
imshow(uint8(g))

%% Butterworth high pass
n = 1; %Butterworth filter of order n
H = 1 - (1./(1 + (D./u0).^(2*n)));
subplot(335)
imshow(abs(fftshift(H)));
subplot(336)
G = H.*F;
g = (ifft2(G));
imshow(uint8(g))

%% Gaussian high pass filter
H = 1 - exp(-(D.^2)./(2*(u0^2)));
subplot(337)
imshow(abs(fftshift(H)));
subplot(338)
G = H.*F;
g = (ifft2(G));
imshow(uint8(g))
```

COMMAND WINDOW

Fig. 10: MATLAB code to apply the Ideal, Butterworth and Gaussian high pass filters

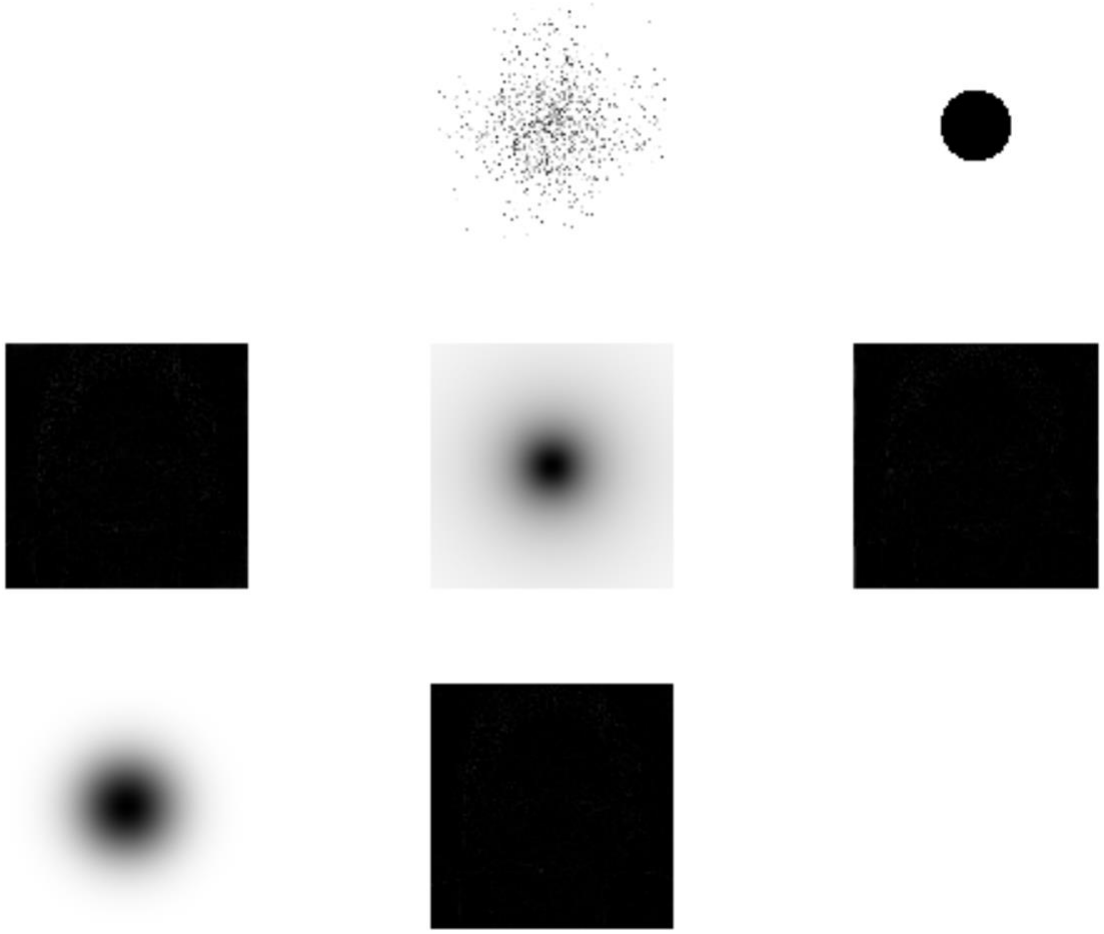


Fig. 11: **Results: Ideal, Butterworth and Gaussian High Pass Filters**

Methodology, Observations & Inferences: -

Fig. 8 & 10 process the method of applying these filters:

- **Import the image data, convert it to grayscale and plot it using the *lines 6 to 10*.**
- **Separately store the dimensions of the grayscale image and convert it into frequency domain according to *lines 11 – 18*. Plot this frequency domain image.**
- **Introduce a threshold in *line 19*, and create a radial distance function with the limiting condition of thresholding as explained in *lines 20 – 27*.**
- **After this point, the separate functions can be created and plotted as required using their distinct definitions. The high and low pass conditions can be taken into account by modifying H appropriately.**

The results for applying these functions can be seen in Fig. 9 & 11. The thresholding limits for both high as well as low pass filters are clearly visible in these figures and well as in Fig. 7. The image smoothening is greatly observed in the Butterworth filter followed by a lesser extent in the Gaussian filter while there is no smoothening in the Ideal filter. As a result, the ringing effect is greatly observed in the Ideal filters while least observed in the Butterworth filters.

Name: Om Mahesh Vaknalli

Roll No. 18376

Subject: EES-338 (Remote Sensing & GIS Lab)

Lab – VIII (Principal Component Analysis (PCA) and Unsupervised Classification)

Aim: (i) Performing Principal Component Analysis

(ii) Performing Unsupervised Classification

(iii) Understanding and performing Thresholding

Theory :-

- Principal Component Analysis (PCA): It is a process of dimension reduction of dataset. It facilitates an increase of interpretability while reducing information loss as well as data size. This is done by introducing new variables in the dataset and assigning them to the maximum/optimal output variance of the existing dataset variables. The covariance between these variables (the principal components) is then calculated and the corresponding eigen values and vectors are obtained. The least important variable is then dropped out and the data is reassembled in terms of the original variable.

- Unsupervised Classification: The classification involves the user input of number of classes (and not the actual defined classes) by the software and it automatically classifies the given data based on the numerical information available in the data. In the case of images, the classification is done using the digital number values as variables. Several classification techniques and their combinations are used by the software for the purpose of classification based on the type and variability of the data.

- Thresholding: It is an image segmentation process to segregate the foreground of data (the relatively more important data) from the background of the data (or the noise). Otsu method is a well-known segmentation method used to conduct automatic image thresholding. The method does this by maximizing the inter-class variance (or minimizing the intra-class variance).

Procedure :-

Performing PCA:

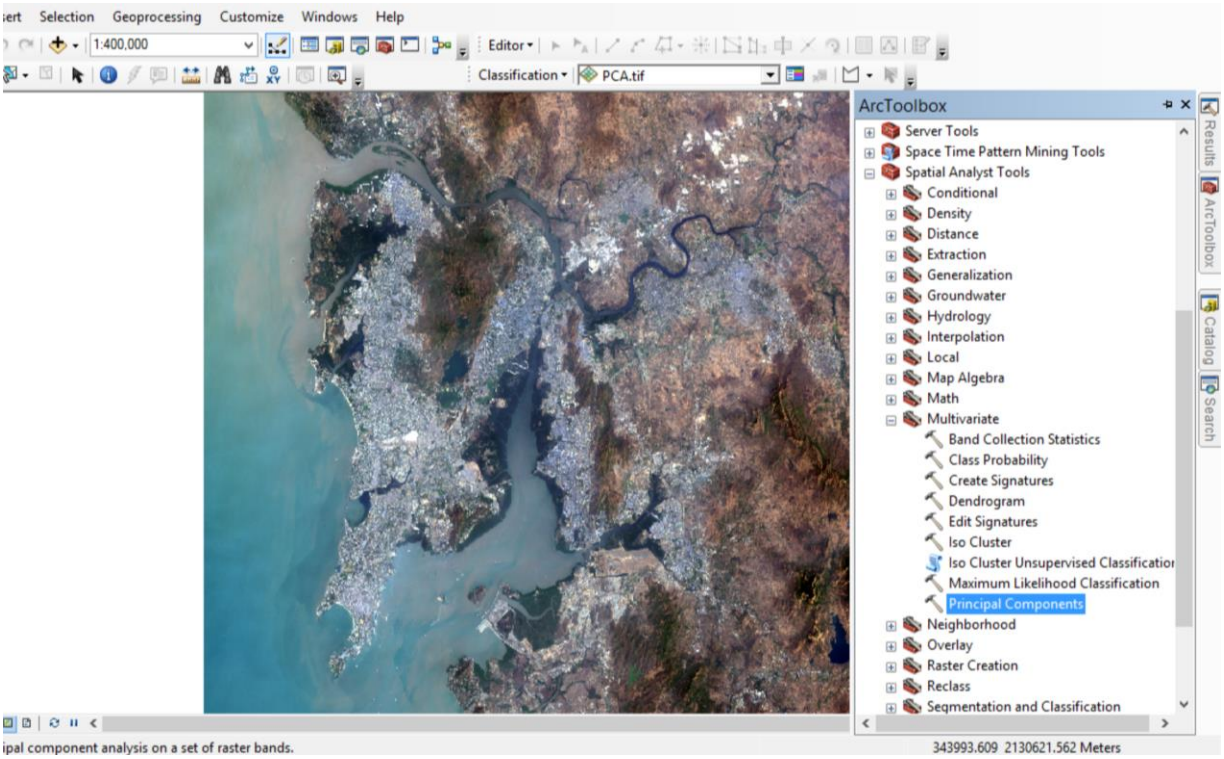


Fig. 1: Open the 'Principal Components' tab from the Arc Toolbox

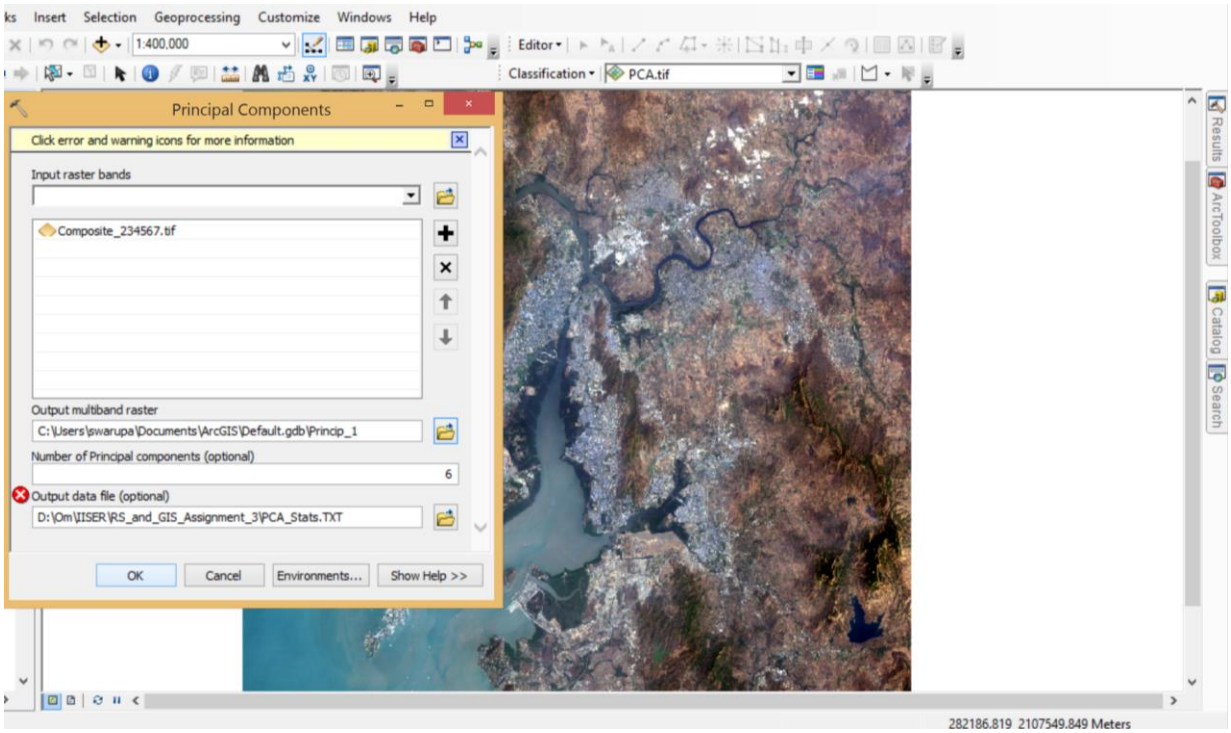


Fig. 2: Fill in the input raster (the composite of bands), output raster location, the no. of PCs available (no. of bands used) and location for the statistics file of the PCA

Fig. 3: **Result: PCA**

Performing Iso-Cluster Unsupervised Classification:

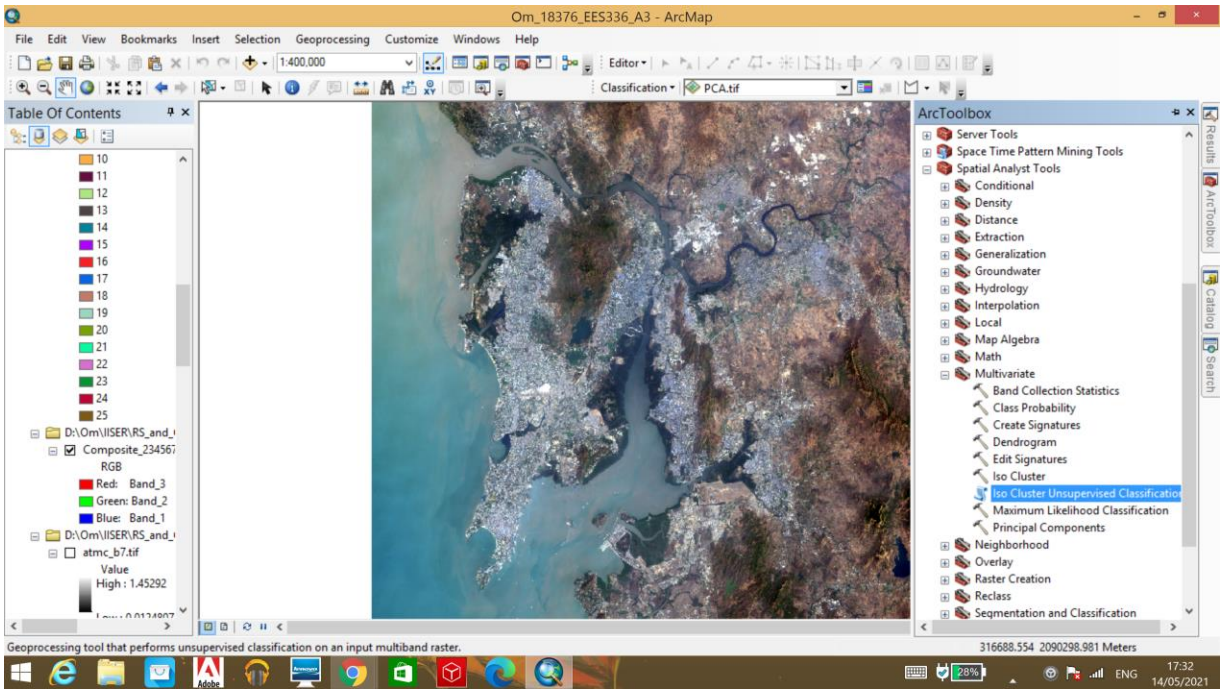


Fig. 4: Open the 'Iso Cluster Unsupervised Classification' tab from the Arc Toolbox

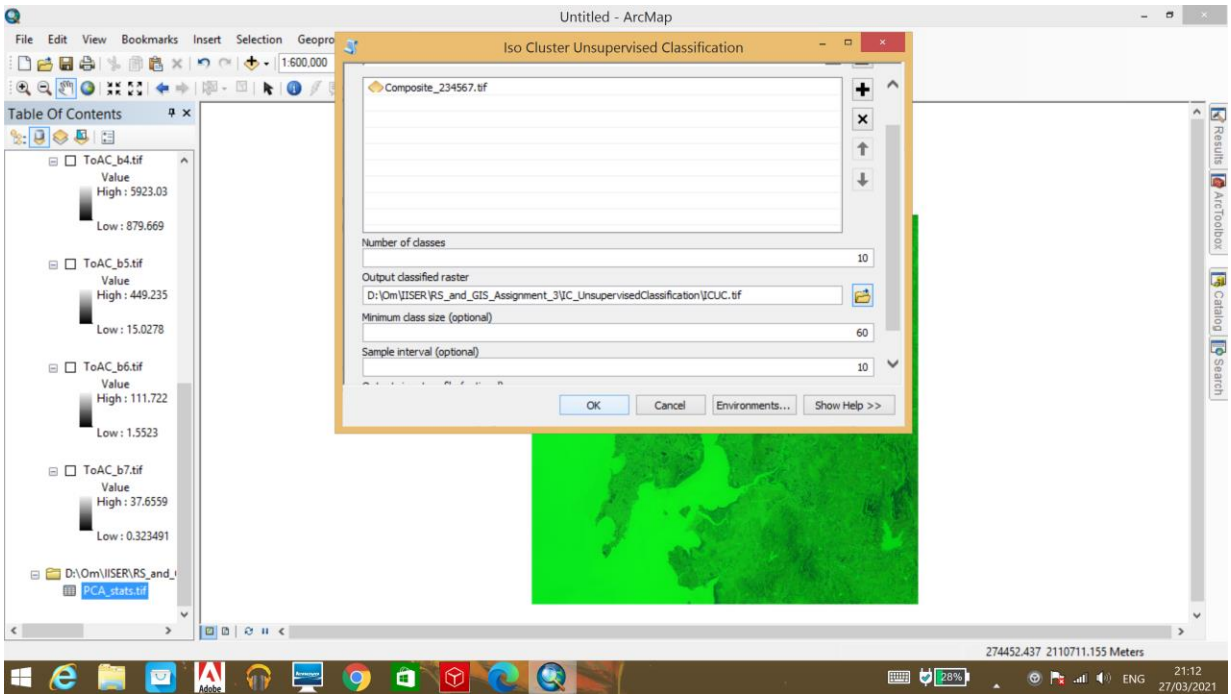


Fig. 5: Fill in the input raster (composite image), no. of desired classes, the output raster location, the minimum class size (usually considered 10 times the no. of bands) and the sample interval

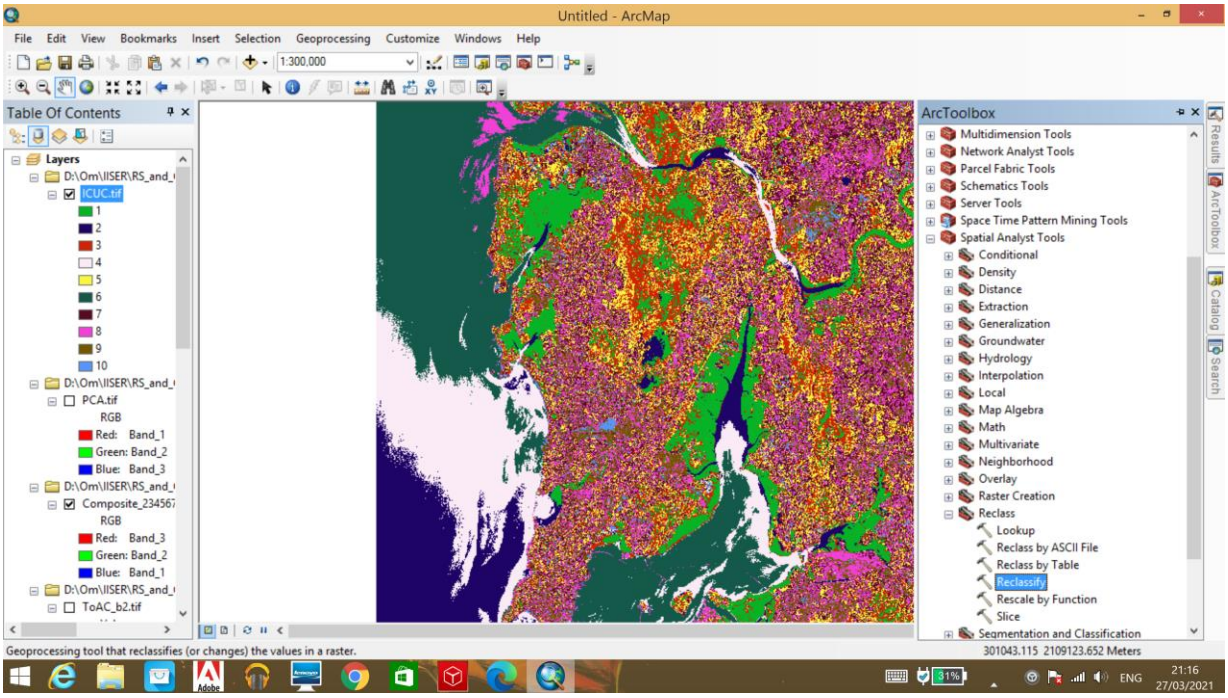


Fig. 6: Result: Iso-Cluster Unsupervised Classification (ICUC)

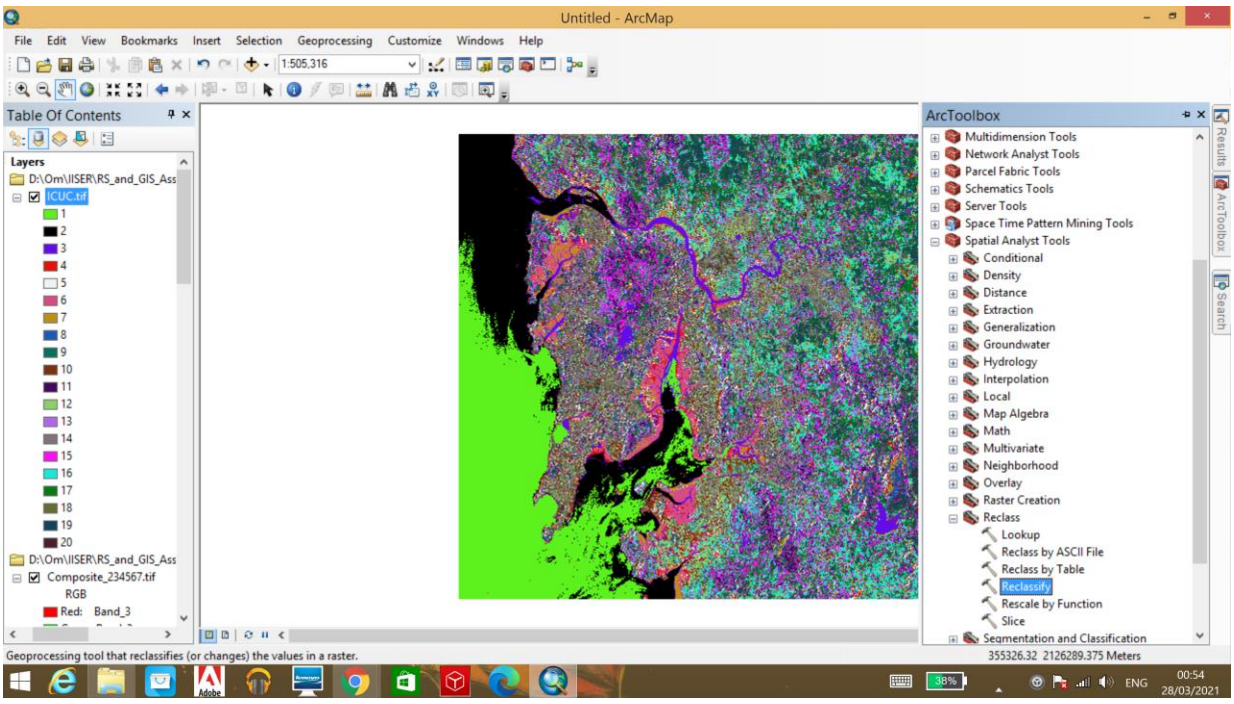
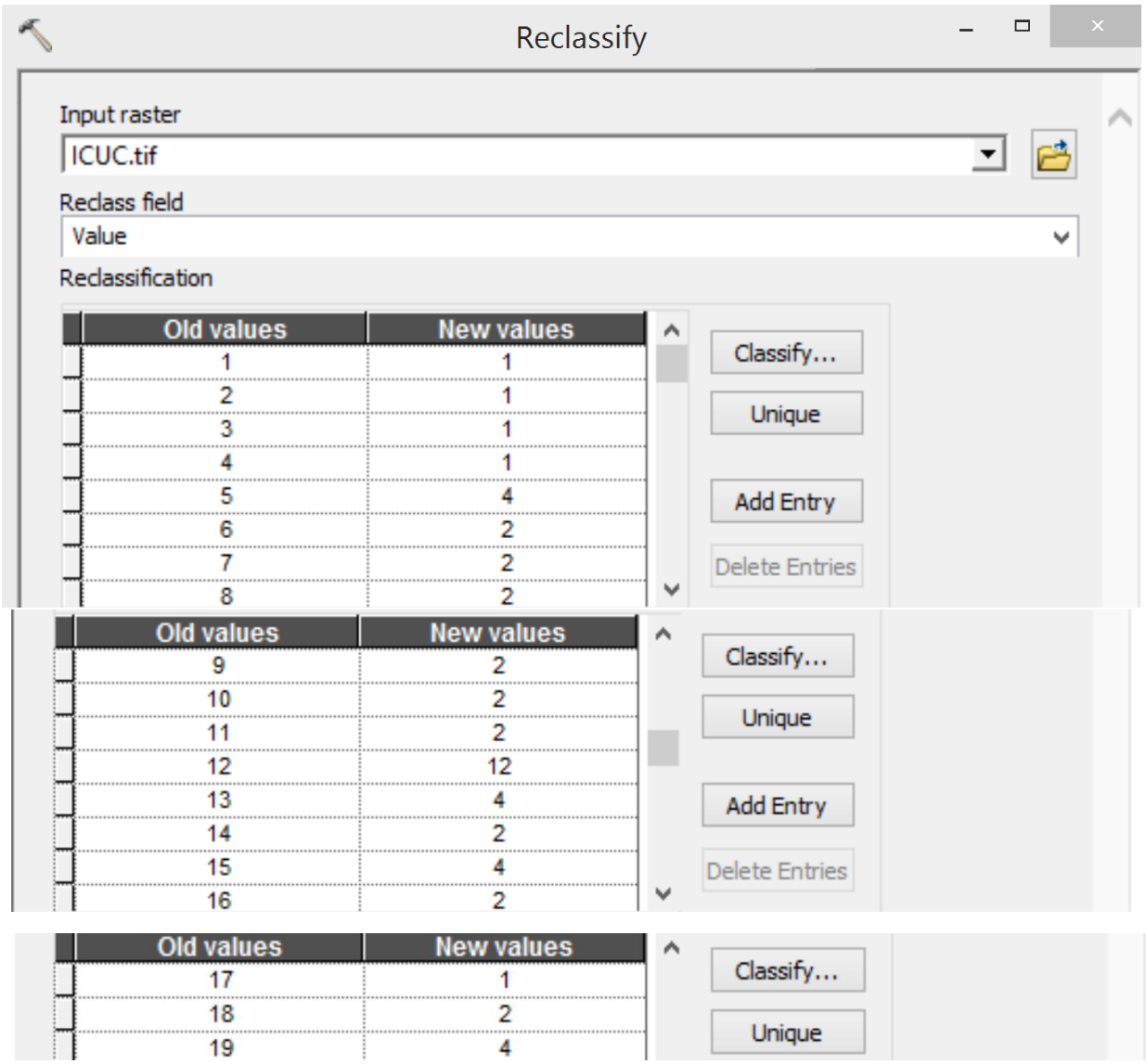


Fig. 7: Select the 'Reclassify' tab in the Arc Toolbox


Reclassify
Input raster
ICUC.tif
Reclass field
Value
Reclassification
Old values
New values
1 1
2 1
3 1
4 1
5 4
6 2
7 2
8 2
9 2
10 2
11 2
12 12
13 4
14 2
15 4
16 2
17 1
18 2
19 4
Classify...
Unique
Add Entry
Delete Entries

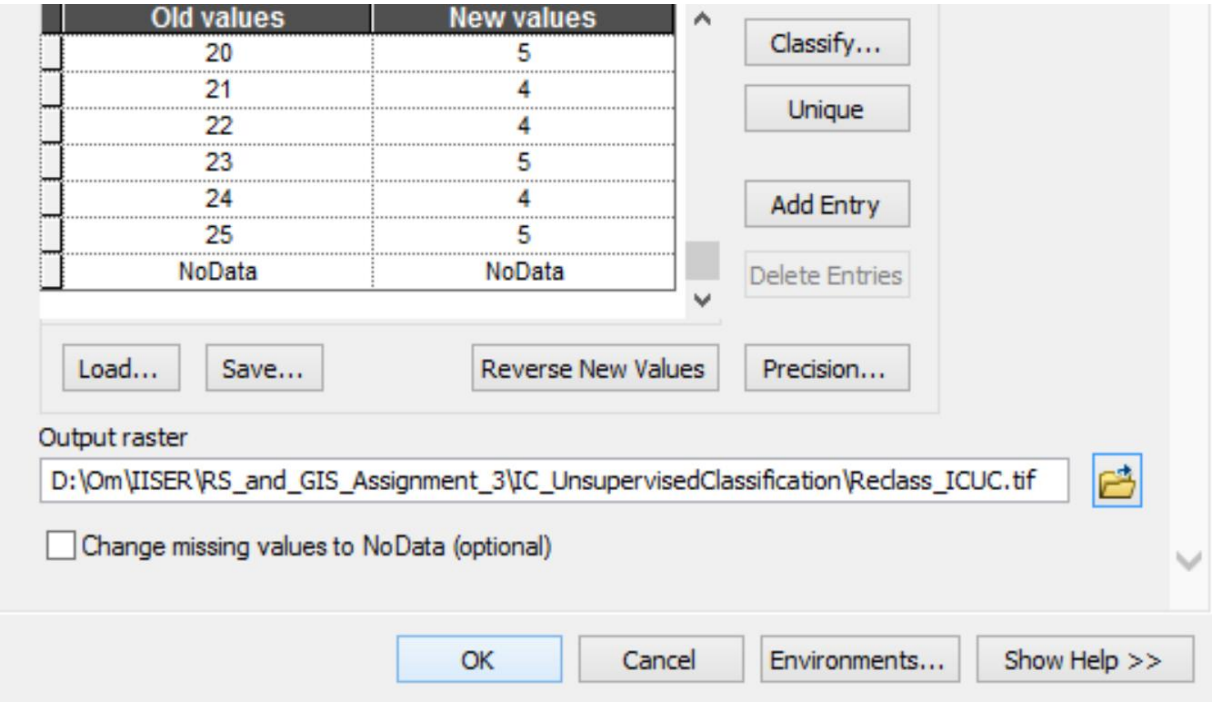


Fig. 8: Reclassify the produced classes as desired according to the various land use classes found in the AOI and select the output raster location

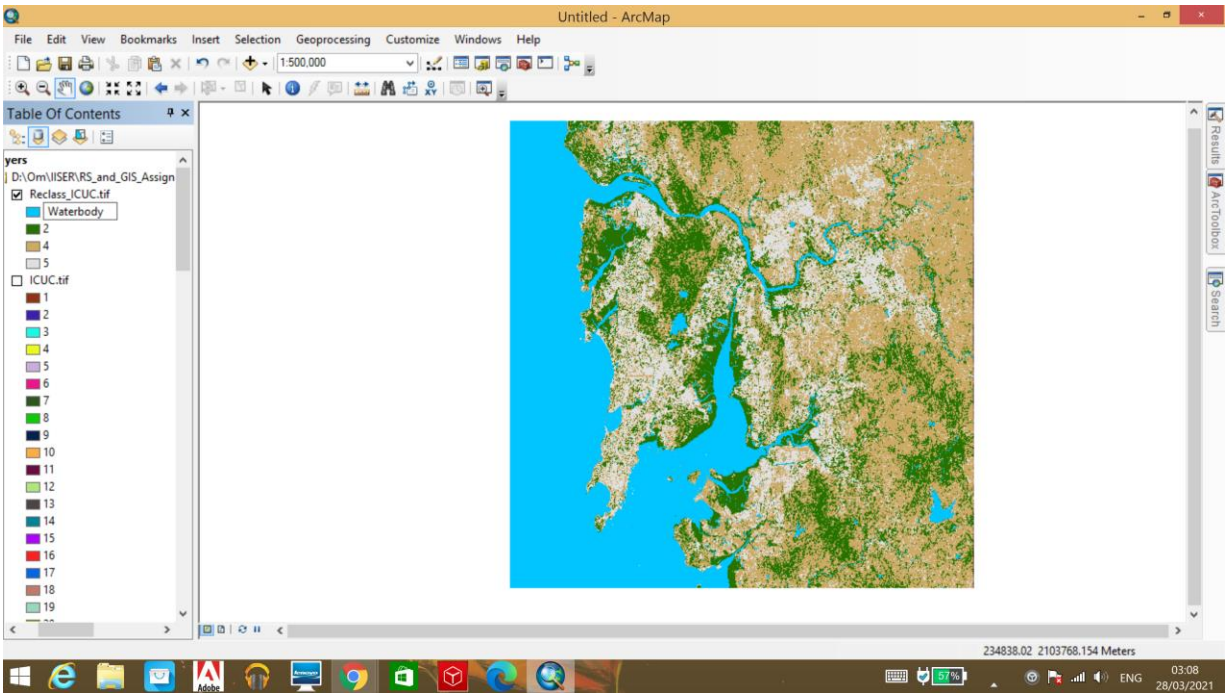


Fig. 9: Rename and recolour the classes as desired

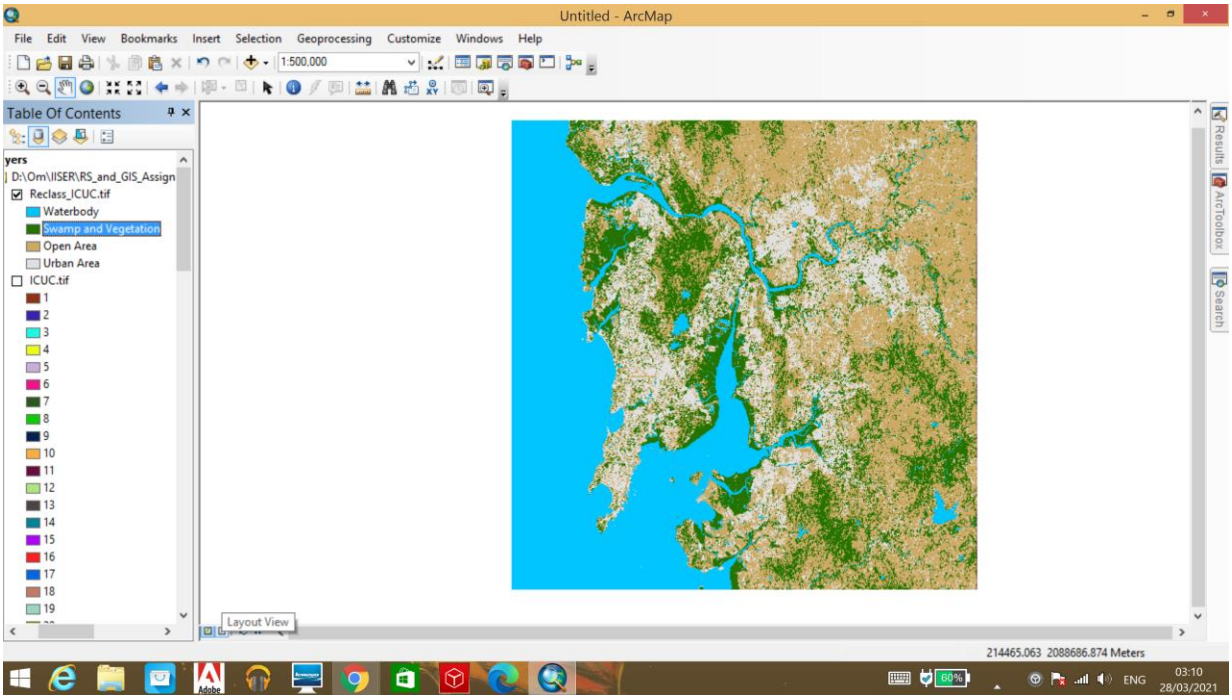


Fig. 10: Result: Reclassed ICUC

Performing Thresholding in ArcGIS:

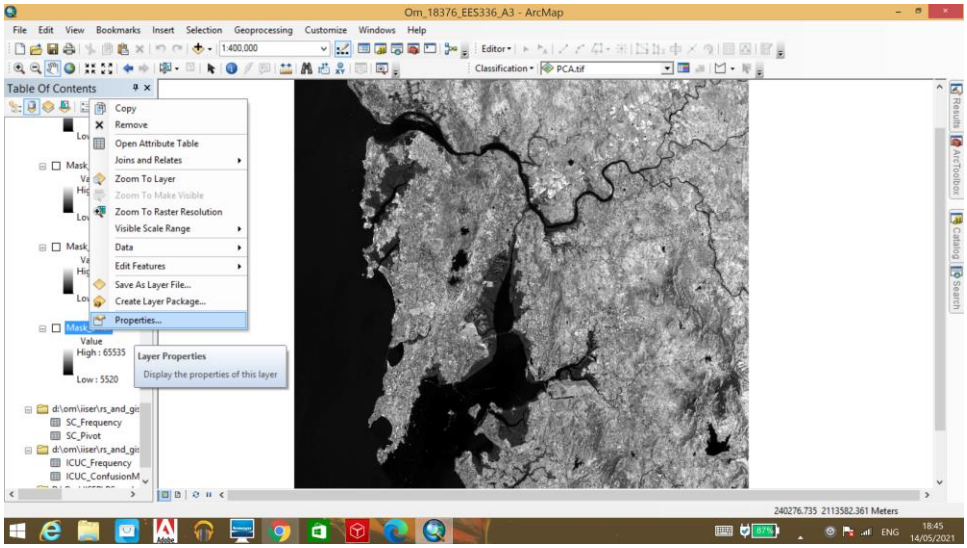


Fig. 11: Open layer properties of the desired band

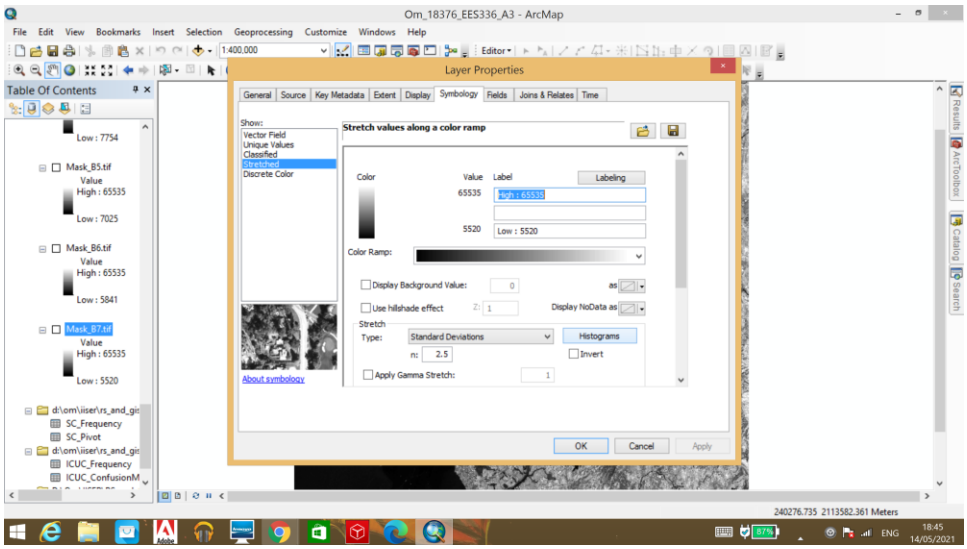


Fig. 12: **Open the histogram for the band**

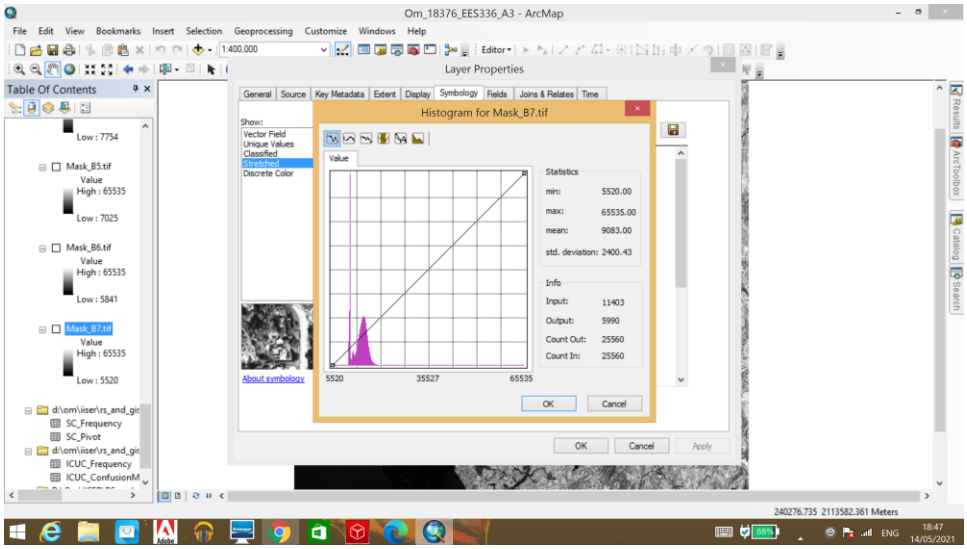


Fig. 13: **Identify the input threshold value between the foreground and the background**

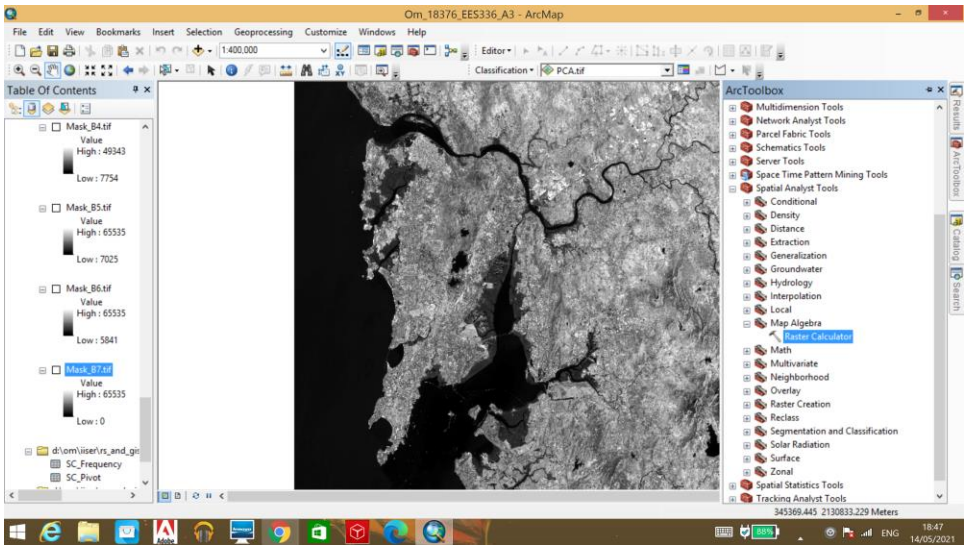


Fig. 14: Open the 'Raster Calculator' from the Arc Toolbox

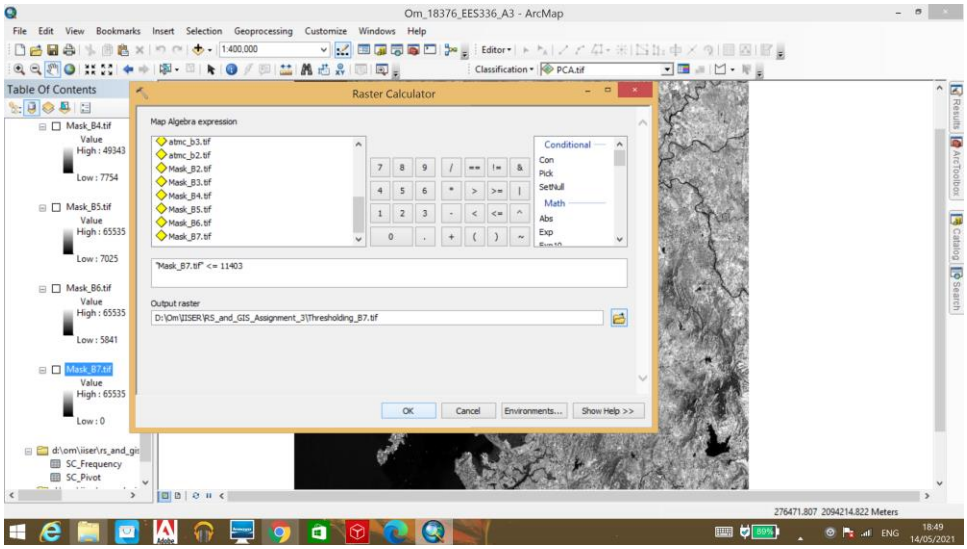


Fig. 15: Enter the mathematical statement to enable the thresholding and select the output raster location

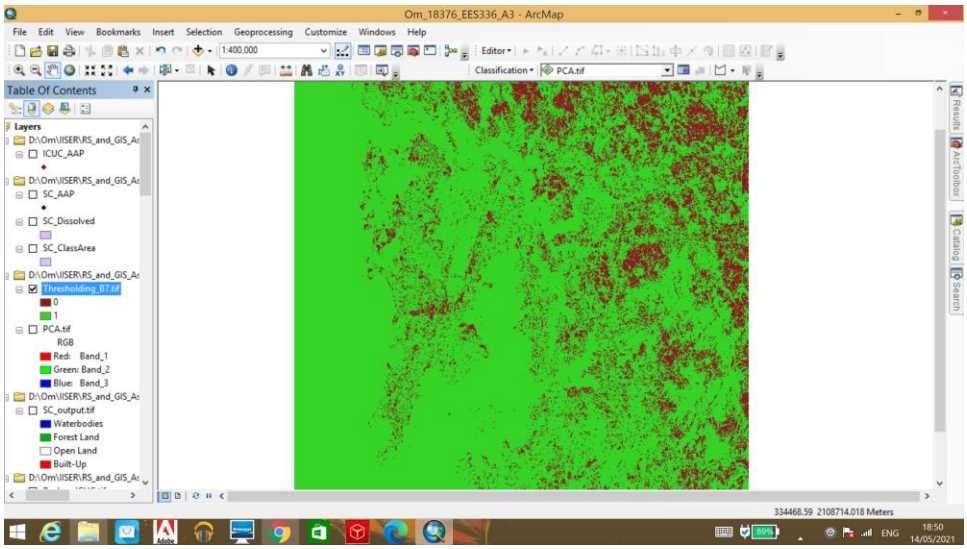


Fig. 16: **Result: Thresholding**

Performing Automatic Thresholding in MATLAB:

```
B=imread('Fake_AI_Generated_Image.png')
```

```
B = 3599×3568×3 uint8 array
B(:,:,1) =

  Columns 1 through 1,666

    53    53    53    53    53    53    53    53    54    55    56    57    58    59    61    62    64
    ⋮
```

```
B=rgb2gray(B);
V=reshape(B,[],1);
H=hist(V,0:255);
plot(H)
```

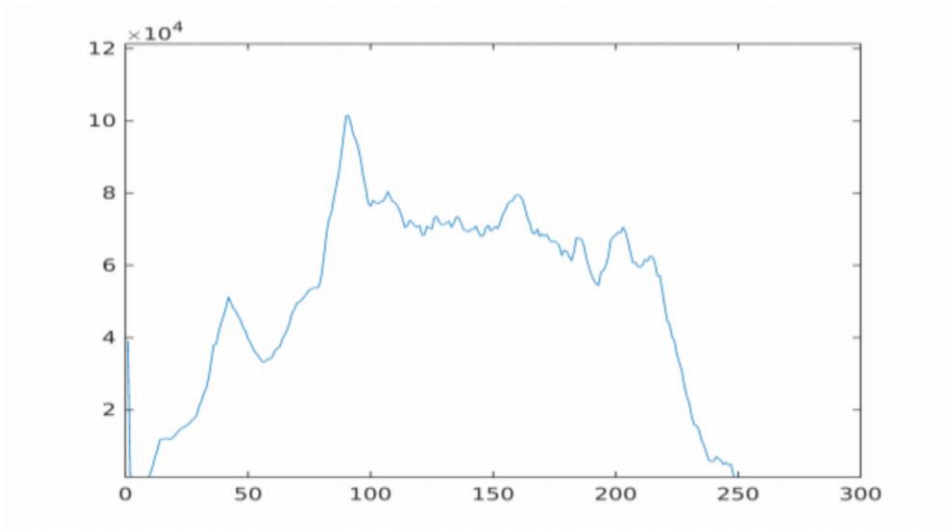


Fig. 17:

- **'imread()' function obtains the image's DN values and displays them.**
- **'rgb2gray()' function converts the image to grayscale.**
- **A histogram of the frequency vs. DN values is plotted in order to determine the thresholding value.**

```
%local function

function [binary]=convert2binary(B, threshold)
B=rgb2gray(B);
[x y]=size(B);
binary=zeros(x,y);
for i=1:x
    for j=1:y
        if B(i,j)>=threshold
            binary(i,j)=1;
        else
            binary(i,j)=0;
        end
    end
end
end
```

```
%% convert2binary
clc
clear all
B=imread('Fake_AI_Generated_Image.png');
threshold=122;
binary=convert2binary(B,threshold);
imshow(binary)
title('convert2binary')
```

```
figure
bin_im=imbinarize(rgb2gray(B),122/256);
imshow(bin_im)
title('inbuit function')
```

Fig. 18:

- **Define a function to convert the grayscale image to Binary using the thresholding value obtained from the histogram**
- **Use the inbuilt function 'imbinarize()' to automatically do the above step and compare the results from the two steps**

Fig. 19: **Result: Automatic Thresholding in MATLAB using user-defined function and inbuilt function**

Performing Thresholding in MATLAB by OTSU method:

```
%%
clc                % To clear the Command Window
clear all          % To clear the Workspace
%%
B=imread('Fake_AI_Generated_Image.png');  % Reading the RGB image
B=rgb2gray(B);                            % Converting the RGB image to Grayscale imag
%%
V=reshape(B,[],1);                        % Converting the array (B) to a column vecto
H=hist(V,0:255);                          % Computing the histogram of V
%%
for i=0:255
 [wbk,mbk,varbk]=calculate(H,1,i);
 [wfg,mfg,varfg]=calculate(H,i+1,255);
 sigma_S_w(i+1)=(wbk*varbk)+(wfg*varfg);  %Equation 13
 sigma_S_b(i+1)=(wbk*wfg)*((mfg-mbk)^2);  %Equation 14
end
lambda_obj_func=(sigma_S_b./sigma_S_w); %First objective function of Equation 12.
 %Find the max value in the array (i.e., minimum of the weighted within-class variance
 [value,k]=min(sigma_S_w);        % k is the threshold level
 tval=(k)/256;
 bin_im=imbinarize(B,tval);
 subplot(1,2,1)
 imshow(bin_im);
 title('Step by step Implementation')
 subplot(1,2,2)
 temp=graythresh(B);
 temp_bin_im=im2bw(B,temp);
 imshow(temp_bin_im)
 title('Using inbuilt Matlab function (graythresh)')
```

```
%% Function for calculating the zeroth (probability), first (mean) and second (varianc
```

```
 function [w,mean,var]=calculate(H,m,n)
i=[0:255];
%Weight Calculation
N=sum(H); %n1+n2+n3+...+nL
w=sum(H(m:n))/N; % (summation of ni)/N;
%Mean Calculation
     value=(i(m:n).*H(m:n))./N;
     total=sum(value); %Mean
     mean=total/w;      %Weighted mean
%Variance calculation.
    value2=(i(m:n)-mean).^2;
     numer=sum(value2.*H(m:n))/N;
     var=numer/w;
 end
```

Fig. 20: Implementing the iterative method as well as using the inbuilt function 'graythresh()' ad comparing their results

Fig. 21: Result: Thresholding by OTSU method

Observation :-

The results obtained from the comparative analysis by both the methods (automatic as well as OTSU) showed minor variations in some pixels. This is because the firstly, both the methods involve different algorithms for computation and hence the rounding errors significantly add up at the end of the process, resulting in the differences. Secondly, both these processes use/generate some amount of randomization (for eg. In the numerical iterations) which can again generate cumulative errors in the process and thereby the differences.

Results :-

The Principal Component Analysis, Unsupervised Classification and Thresholding were successfully studied and applied.