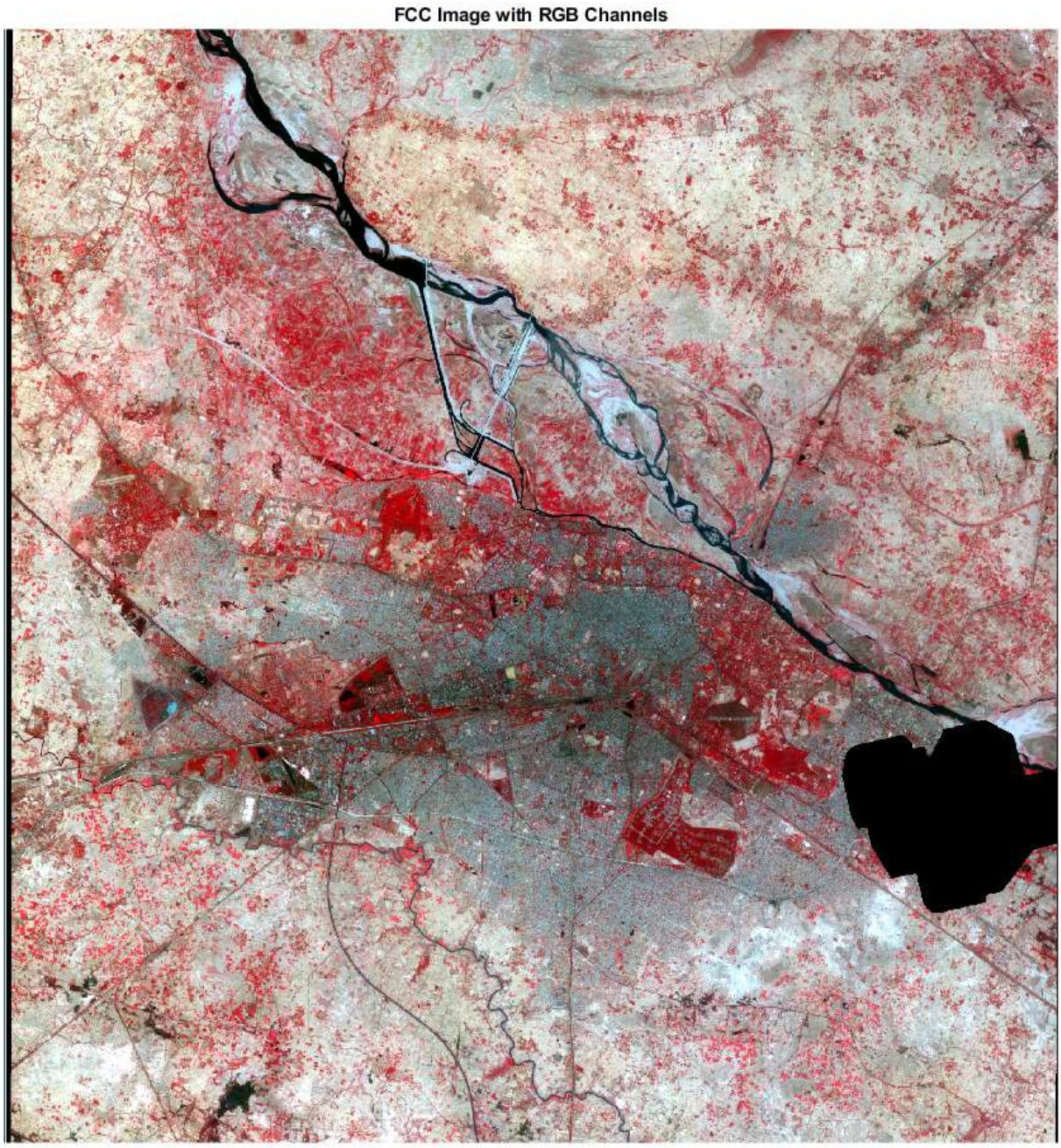
Contents


```
% Modified MATLAB Code for Remote Sensing Image Analysis
```

Step 1: Load and Visualize FCC Image using RGB Channels

```
% Load images corresponding to RGB channels
img_green = imread('C:\Users\Srijan\OneDrive\Documents\MATLAB\L4_tiff\imagery1.tif'); % Band 2: Green
img_red = imread('C:\Users\Srijan\OneDrive\Documents\MATLAB\L4_tiff\imagery2.tif');   % Band 3: Red
img_nir = imread('C:\Users\Srijan\OneDrive\Documents\MATLAB\L4_tiff\imagery3.tif');   % Band 4: NIR

% Check dimensions of the images
fprintf('Dimensions of Band 2: %s\n', num2str(size(img_green)));
fprintf('Dimensions of Band 3: %s\n', num2str(size(img_red)));
fprintf('Dimensions of Band 4: %s\n', num2str(size(img_nir)));

% Verify the images are RGB
[rows_g, cols_g, ch_g] = size(img_green);
[rows_r, cols_r, ch_r] = size(img_red);
[rows_nir, cols_nir, ch_nir] = size(img_nir);

% Ensure all images have 3 color channels
if ch_g ~= 3 || ch_r ~= 3 || ch_nir ~= 3
    error('One or more images do not have three channels.');
end

% Confirm the images are of the same dimensions
if ~isequal([rows_g, cols_g], [rows_r, cols_r]) || ~isequal([rows_r, cols_r], [rows_nir, cols_nir])
    error('Images must be of identical size.');
end

% Generate FCC image by combining bands into an RGB image
fcc_img = cat(3, img_nir(:,:,1), img_red(:,:,1), img_green(:,:,1)); % Change channels as needed

% Validate FCC image dimensions
[rows_fcc, cols_fcc, ch_fcc] = size(fcc_img);
fprintf('FCC Image Dimensions: %s\n', num2str(size(fcc_img)));

% Check that FCC image is an RGB image
if ch_fcc ~= 3
    error('FCC image should have three color channels.');
end

% Display the FCC image
figure;
imshow(fcc_img);
title('FCC Image with RGB Channels');
```

```
Dimensions of Band 2: 5275  4992     3
Dimensions of Band 3: 5275  4992     3
Dimensions of Band 4: 5275  4992     3
FCC Image Dimensions: 5275  4992     3
```



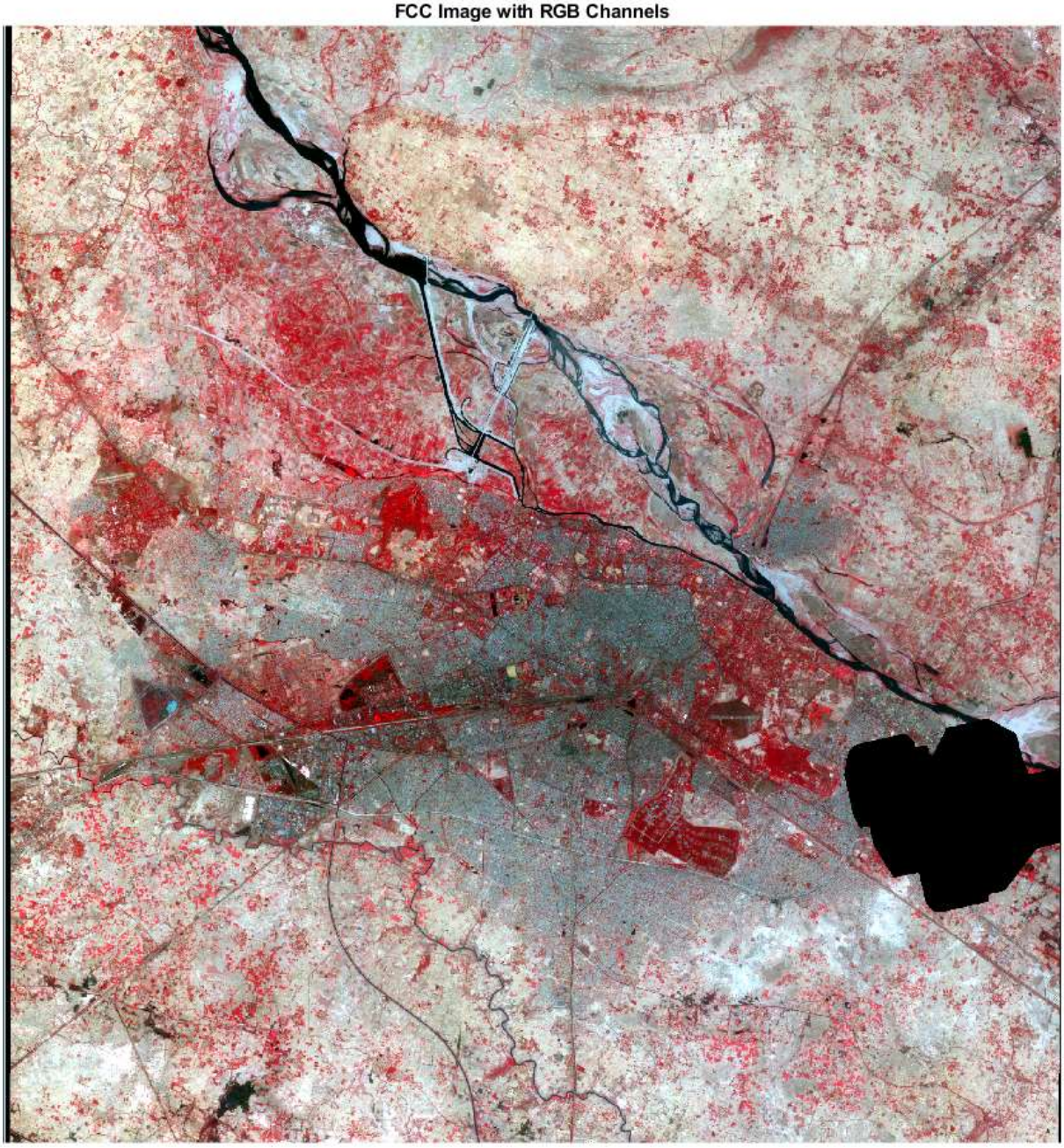
Step 2: Analyze Pixel Values and Comment on Distribution

```
% Interactive pixel value visualization
impixelinfo;

% Get pixel values programmatically at a specific location
x_loc = 100;
y_loc = 100;

% Extract pixel values from each band
val_green = squeeze(img_green(y_loc, x_loc, :));
val_red = squeeze(img_red(y_loc, x_loc, :));
val_nir = squeeze(img_nir(y_loc, x_loc, :));

fprintf('Pixel values at (%d, %d) - Green: [%d], Red: [%d], NIR: [%d]\n', ...
        x_loc, y_loc, val_green(1), val_red(1), val_nir(1));

% Plot histograms of pixel intensities for each image
figure;
subplot(3,1,1);
imhist(rgb2gray(img_green)); % Convert to grayscale for histogram
```

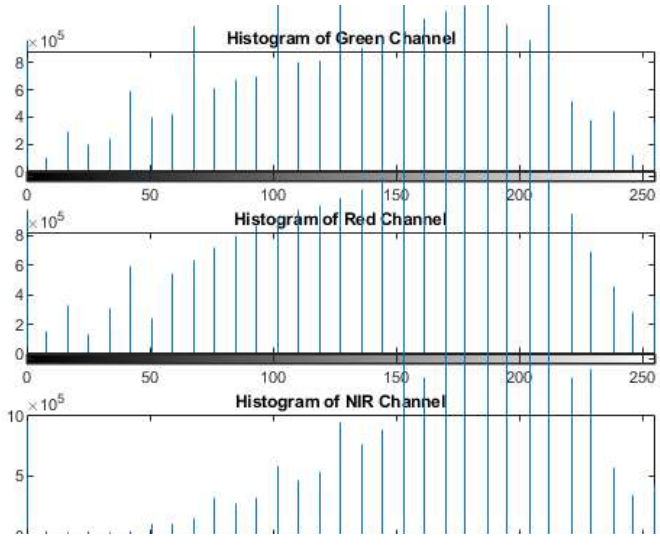
```
title('Histogram of Green Channel');

subplot(3,1,2);
imhist(rgb2gray(img_red)); % Convert to grayscale for histogram
title('Histogram of Red Channel');

subplot(3,1,3);
imhist(rgb2gray(img_nir)); % Convert to grayscale for histogram
title('Histogram of NIR Channel');
```

```
Pixel values at (100, 100) - Green: [229], Red: [238], NIR: [246]
```

FCC Image with RGB Channels



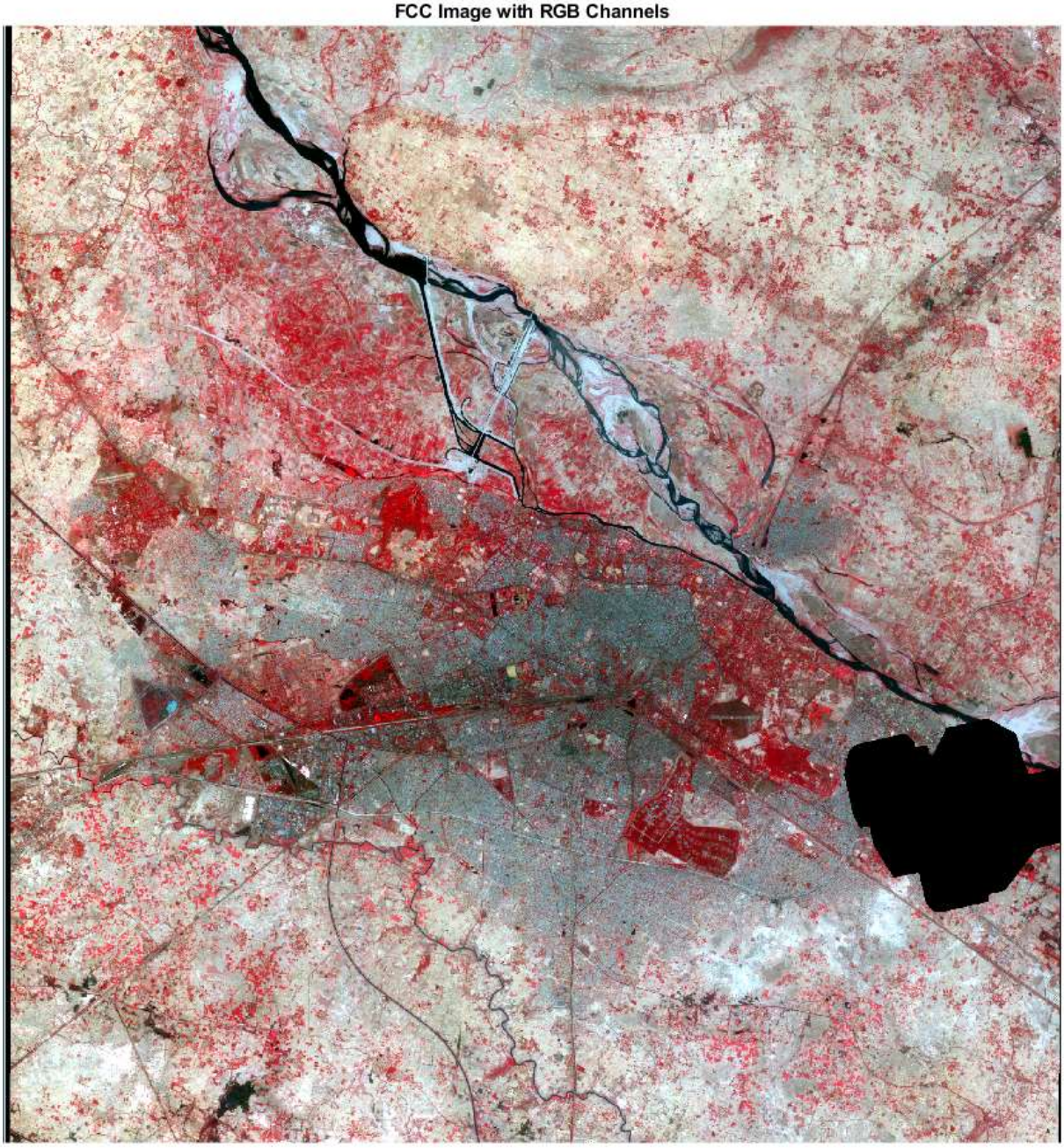
Pixel info: (2169, 4597) [229 221 195]



Step 3: Explore Different RGB Band Combinations

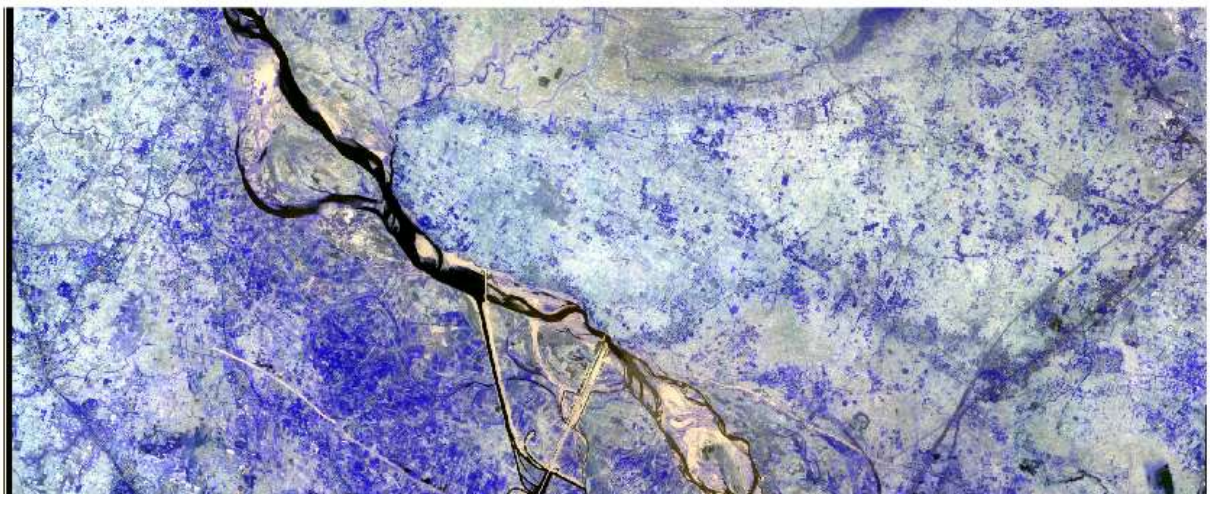
```
% Define combinations of bands for different RGB settings
band_combos = {...
    {img_green(:,:,1), img_red(:,:,1), img_nir(:,:,1)}, ... % Green (R), Red (G), NIR (B)
    {img_green(:,:,1), img_nir(:,:,1), img_red(:,:,1)}, ... % Green (R), NIR (G), Red (B)
    {img_red(:,:,1), img_green(:,:,1), img_nir(:,:,1)}, ... % Red (R), Green (G), NIR (B)
    {img_red(:,:,1), img_nir(:,:,1), img_green(:,:,1)}, ... % Red (R), NIR (G), Green (B)
    {img_nir(:,:,1), img_green(:,:,1), img_red(:,:,1)}, ... % NIR (R), Green (G), Red (B)
    {img_nir(:,:,1), img_red(:,:,1), img_green(:,:,1)} ...  % NIR (R), Red (G), Green (B)
};

% Display the combinations
for idx = 1:length(band_combos)
    figure;
    imshow(cat(3, band_combos{idx}{:}));
    title(sprintf('RGB Combination %d', idx));
end
```

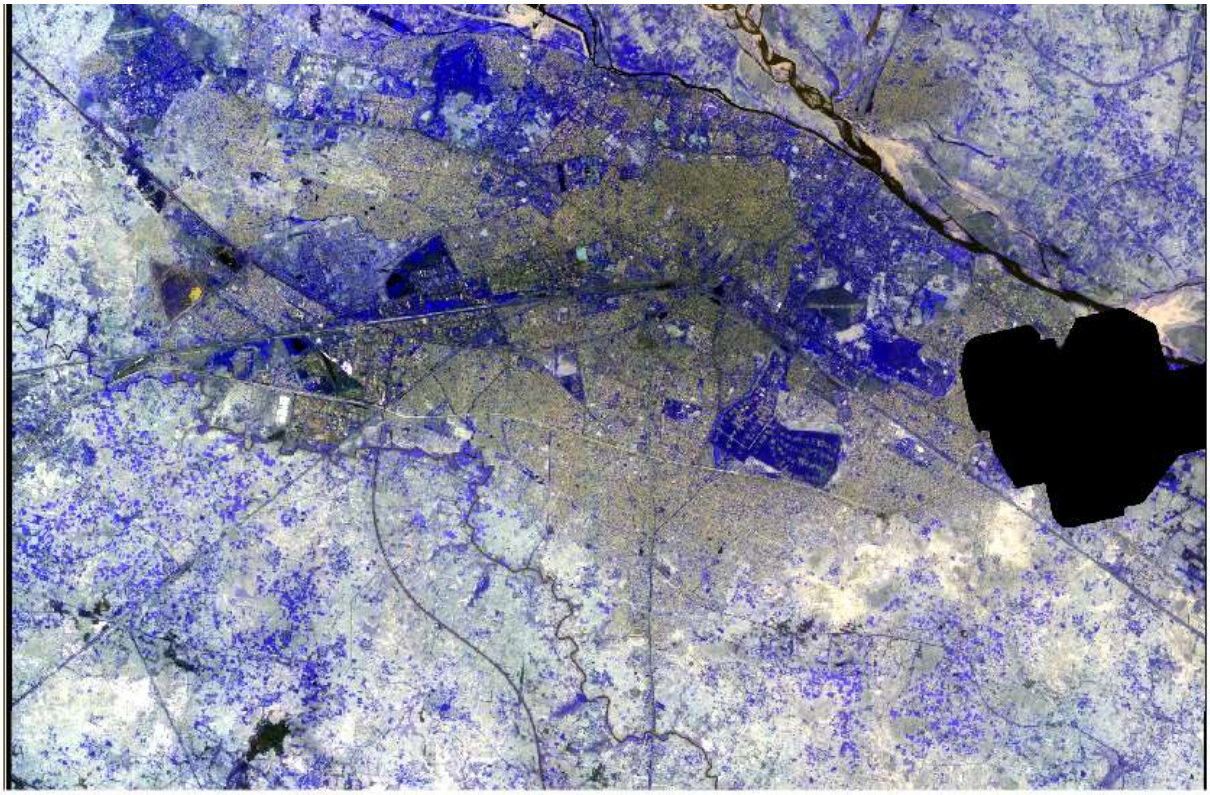
FCC Image with RGB Channels



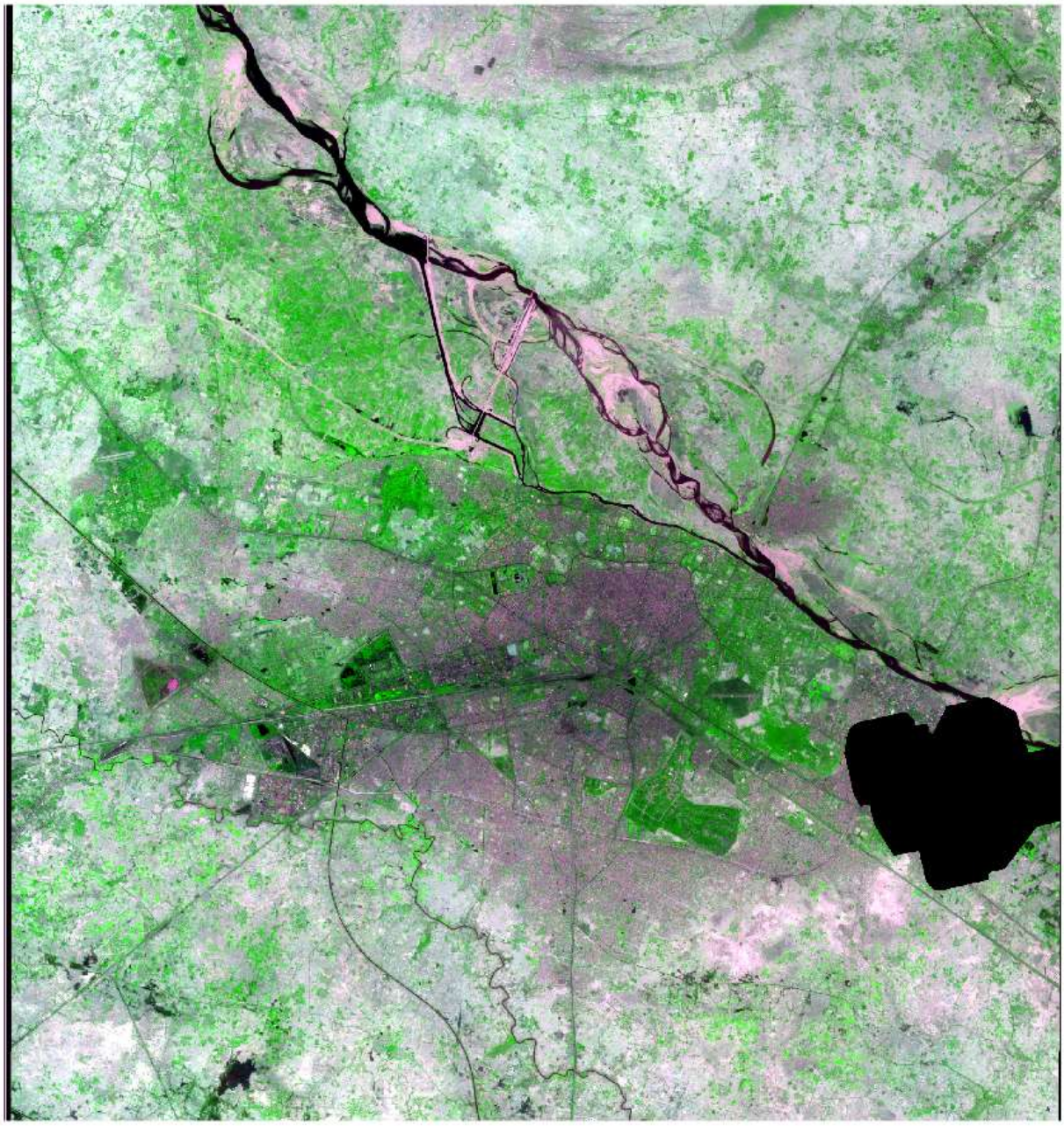
Pixel info: (X, Y) [R G B]

RGB Combination 1

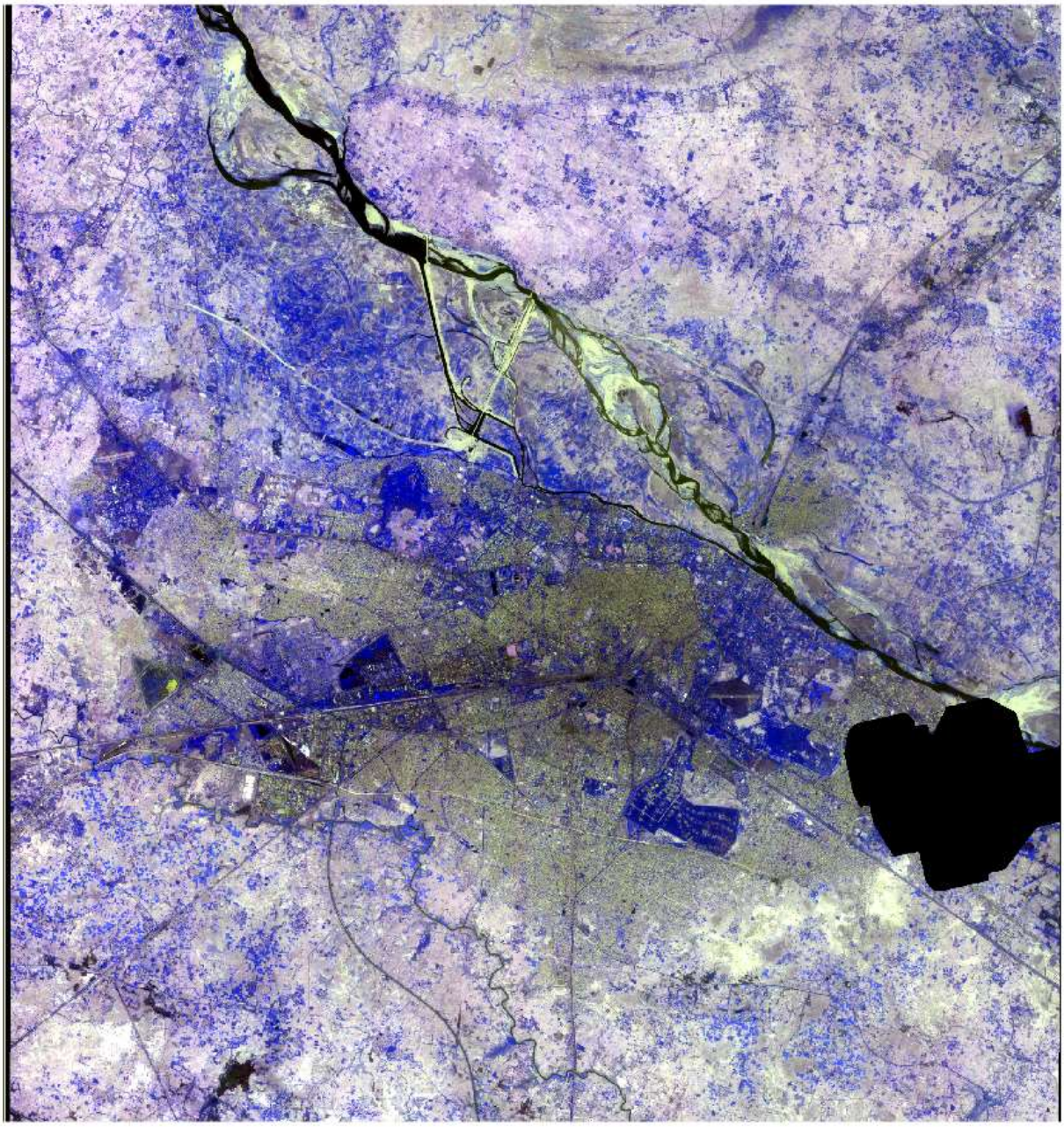




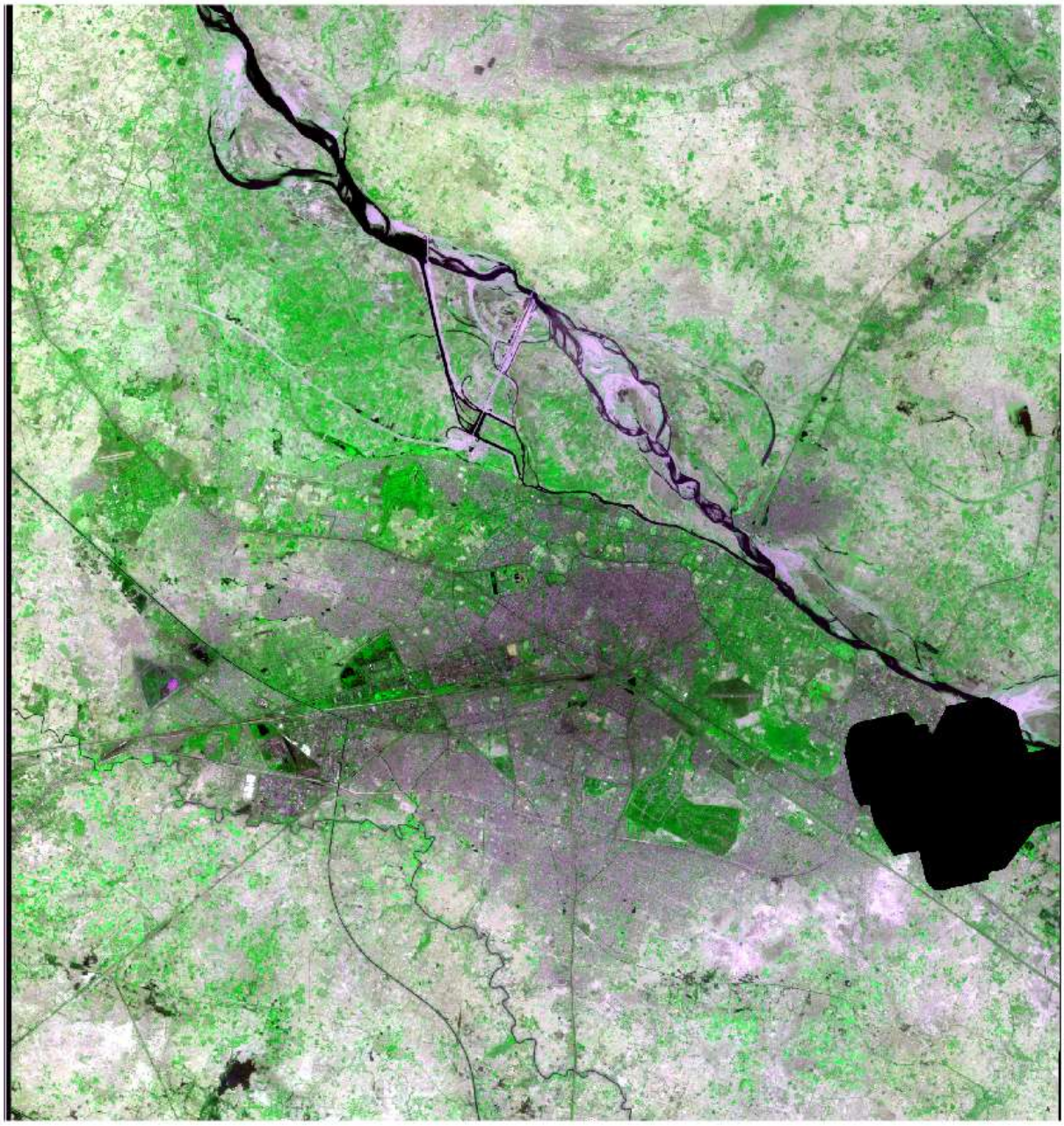
RGB Combination 2



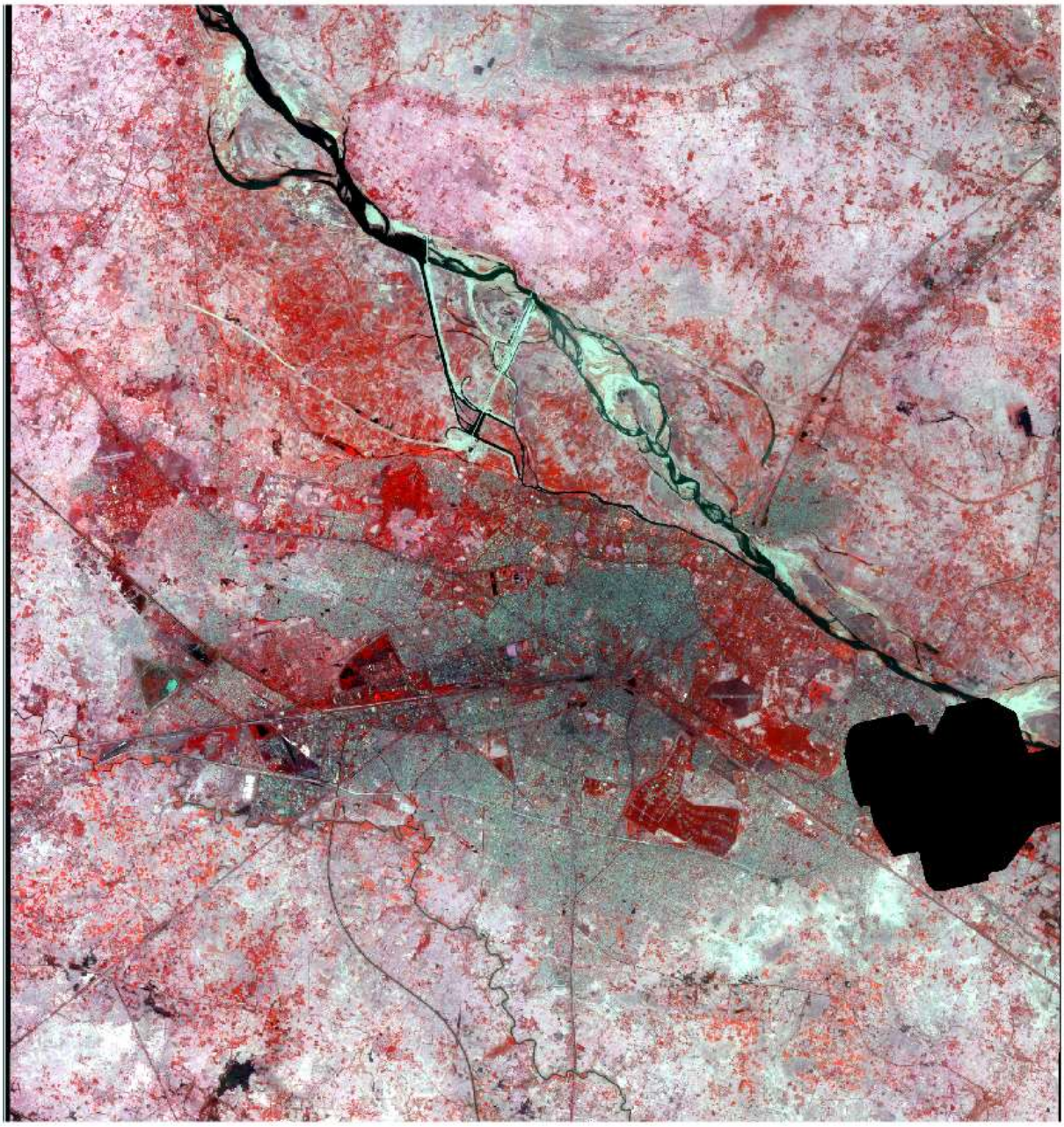
RGB Combination 3

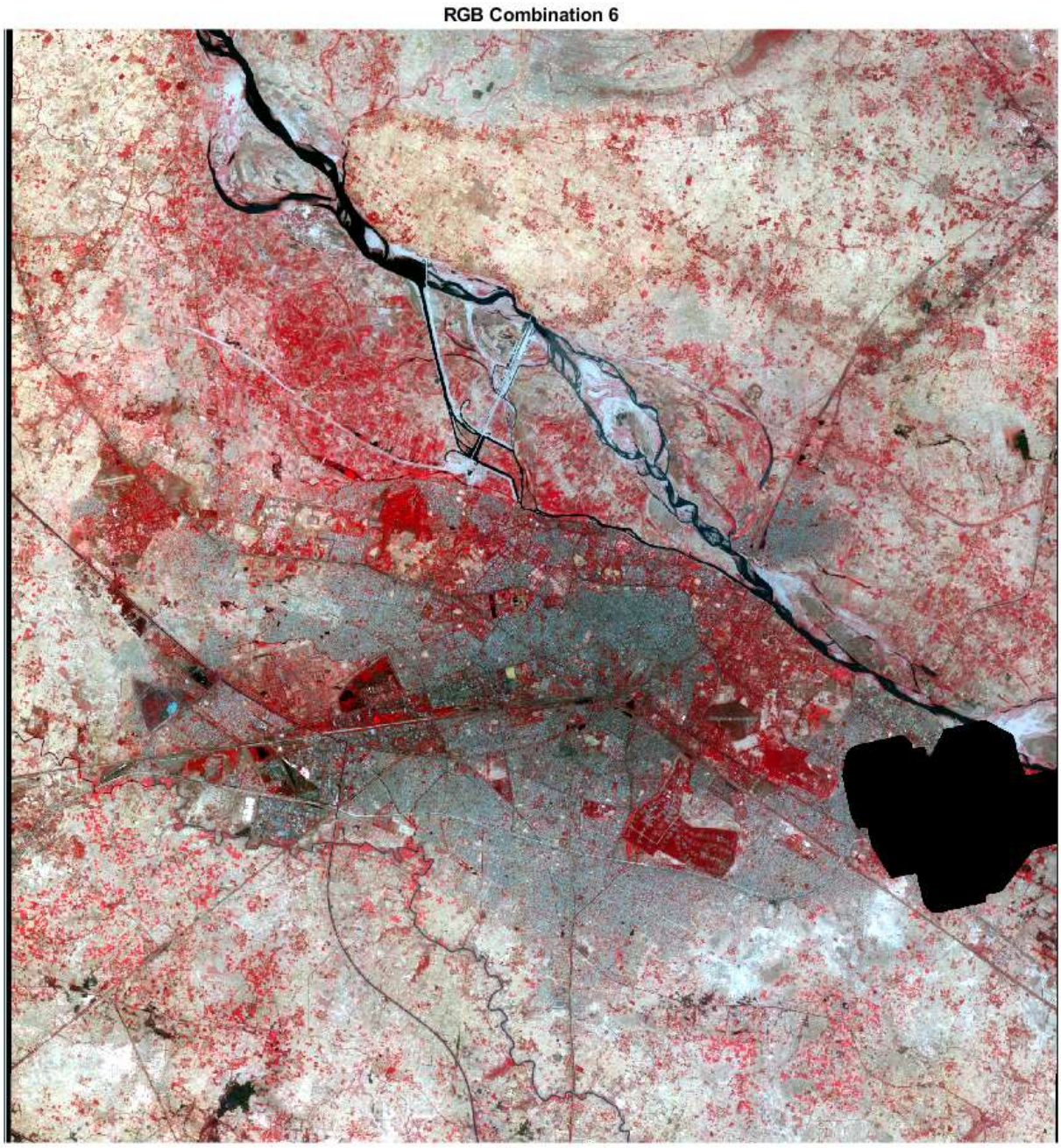


RGB Combination 4



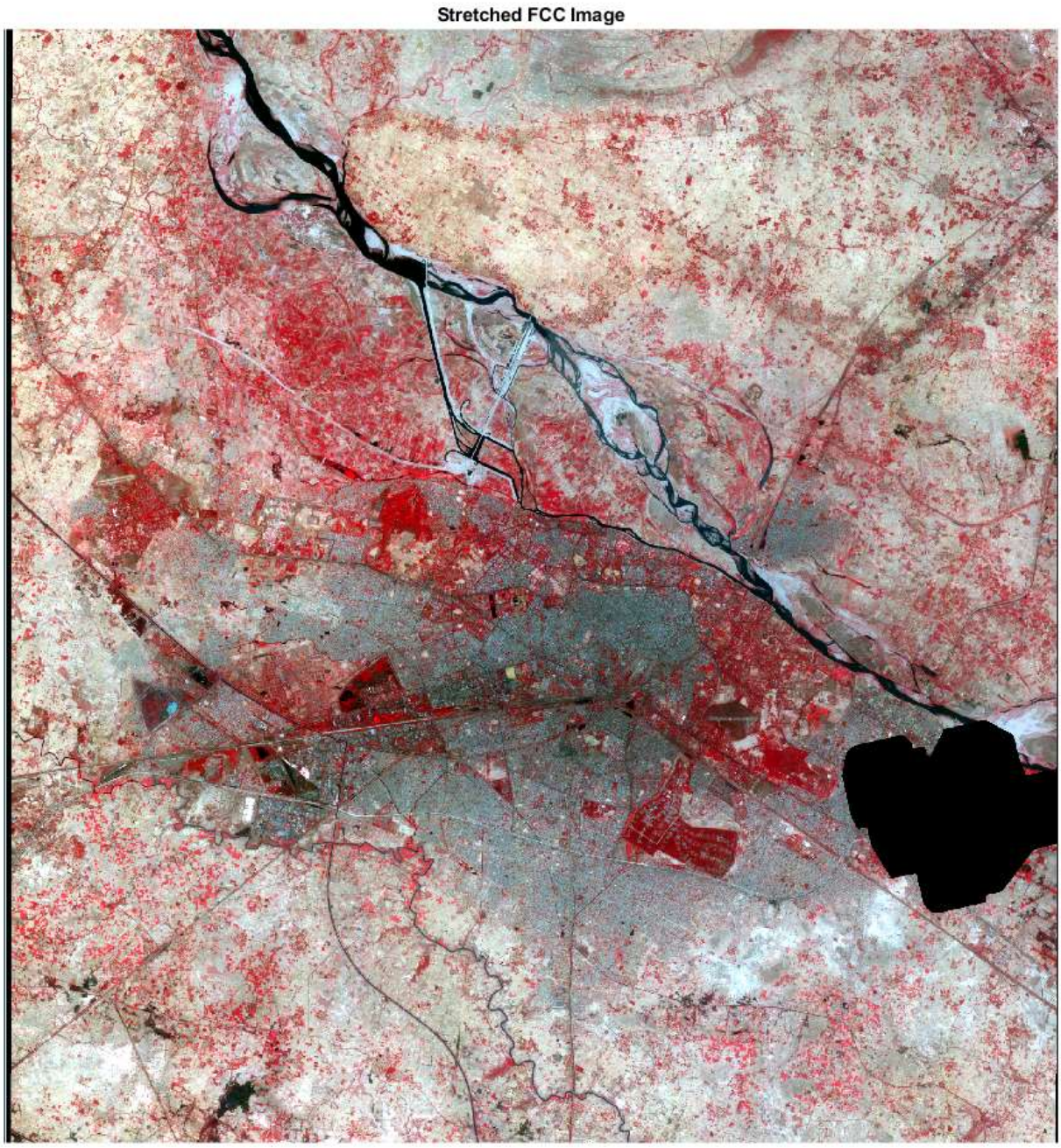
RGB Combination 5





Step 4: Perform Linear Stretching on Each Band

```
% Apply linear stretching to each image
stretched_green = apply_linear_stretch(img_green);
stretched_red = apply_linear_stretch(img_red);
stretched_nir = apply_linear_stretch(img_nir);

% Create a new FCC image using the stretched images
stretched_fcc_img = cat(3, stretched_nir(:,:,1), stretched_red(:,:,1), stretched_green(:,:,1));

% Display the stretched FCC image
figure;
imshow(stretched_fcc_img);
title('Stretched FCC Image');
```



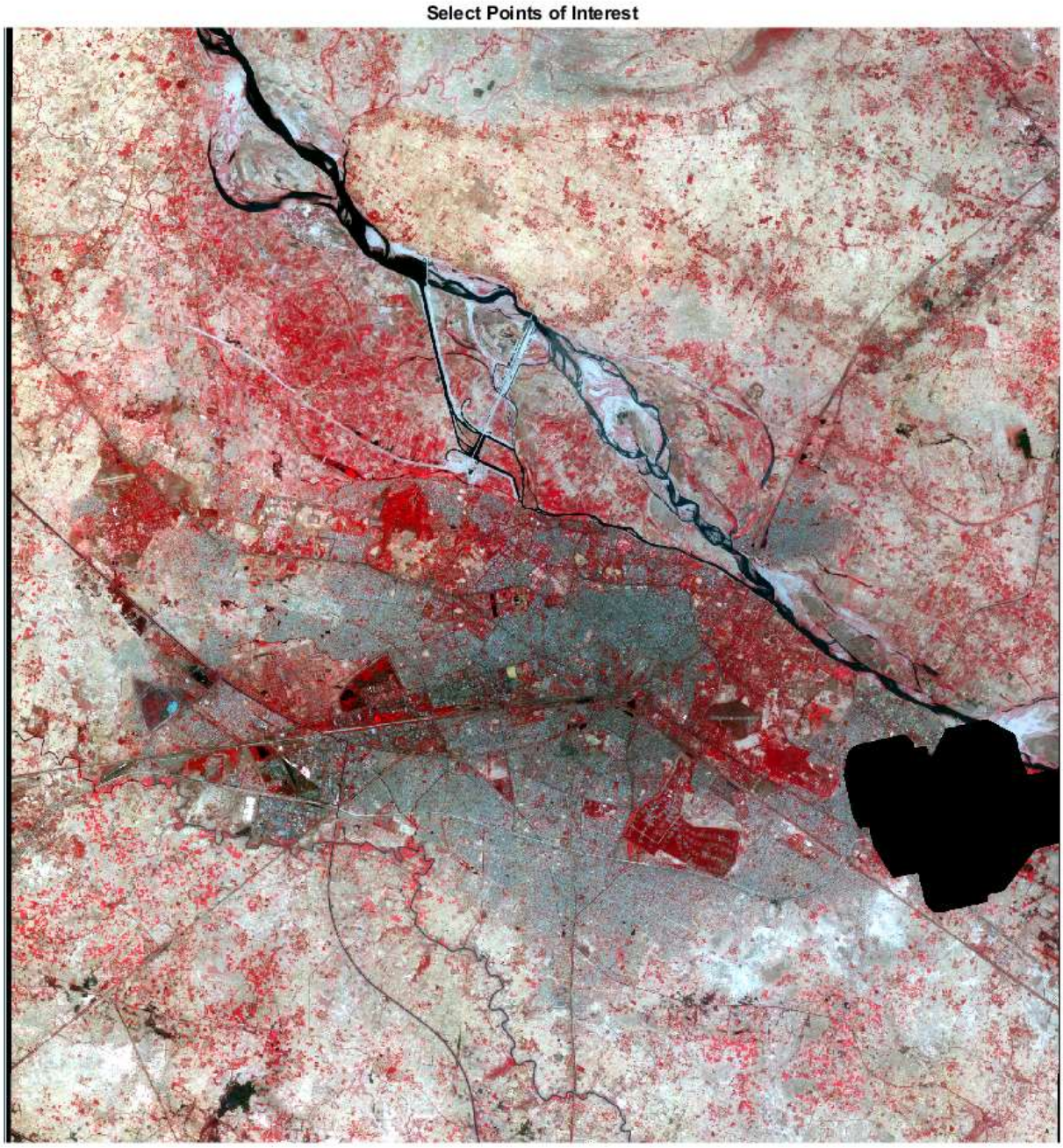
Step 5: Select and Analyze Pixels

```
% Select pixels interactively from the FCC image
figure;
imshow(fcc_img);
title('Select Points of Interest');
[x_coords, y_coords] = ginput(5); % Select five points

% Initialize arrays for storing information
pixel_indices = (1:5)';
pixel_locations = cell(5, 1);
original_rgb_values = cell(5, 1);
original_hex_colors = cell(5, 1);
stretched_rgb_values = cell(5, 1);
stretched_hex_colors = cell(5, 1);

% Loop through each selected point
for i = 1:length(x_coords)
    x_p = round(x_coords(i));
    y_p = round(y_coords(i));
```

```
    % Ensure selected point is within image bounds
    if x_p < 1 || x_p > size(fcc_img, 2) || y_p < 1 || y_p > size(fcc_img, 1)
        fprintf('Selected point (%d, %d) is out of bounds.\n', x_p, y_p);
        continue;
    end

    % Retrieve original RGB values
    orig_r_val = img_nir(y_p, x_p); % NIR as Red
    orig_g_val = img_red(y_p, x_p); % Red as Green
    orig_b_val = img_green(y_p, x_p); % Green as Blue

    % Retrieve stretched RGB values
    str_r_val = stretched_nir(y_p, x_p); % Stretched NIR as Red
    str_g_val = stretched_red(y_p, x_p); % Stretched Red as Green
    str_b_val = stretched_green(y_p, x_p); % Stretched Green as Blue

    % Store information for table
    pixel_locations{i} = [x_p, y_p];
    original_rgb_values{i} = [orig_r_val, orig_g_val, orig_b_val];
    stretched_rgb_values{i} = [str_r_val, str_g_val, str_b_val];

    % Convert RGB values to hex colors
    original_hex_colors{i} = convert_rgb_to_hex([orig_r_val, orig_g_val, orig_b_val]);
    stretched_hex_colors{i} = convert_rgb_to_hex([str_r_val, str_g_val, str_b_val]);

    % Display pixel values for debugging
    fprintf('Pixel %d - Location: (%d, %d), Original RGB: [%d, %d, %d], Stretched RGB: [%d, %d, %d]\n', ...
        i, x_p, y_p, orig_r_val, orig_g_val, orig_b_val, str_r_val, str_g_val, str_b_val);
end

% Define LULC (Land Use/Land Cover) labels for selected pixels
land_use_labels = {'Roof', 'Tree', 'River', 'Road', 'Farm'};

% Create table with pixel information
pixel_table = table(pixel_indices, pixel_locations, ...
    land_use_labels', ... % Assign the LULC labels
    original_rgb_values, ...
    original_hex_colors, ...
    stretched_rgb_values, ...
    stretched_hex_colors);

% Display the table
disp(pixel_table);
```

Function for Linear Stretching

```
function img_stretched = apply_linear_stretch(img)
    % Identify the minimum and maximum values in the image
    min_val = double(min(img(:)));
    max_val = double(max(img(:)));

    % Check if min and max values are the same, in which case return original image
    if min_val == max_val
        img_stretched = img;
    else
        % Perform linear stretching on the image
        img_stretched = uint8(255 * (double(img) - min_val) / (max_val - min_val));
    end
end
```



Function to Convert RGB to Hexadecimal

```
function hex_val = convert_rgb_to_hex(rgb)
    % Convert RGB array to hexadecimal string (#RRGGBB)
    hex_val = sprintf('#%02X%02X%02X', rgb(1), rgb(2), rgb(3));
end
```

```
Pixel 1 - Location: (2379, 1969), Original RGB: [204, 195, 229], Stretched RGB: [204, 195, 229]
Pixel 2 - Location: (1623, 1399), Original RGB: [187, 136, 136], Stretched RGB: [187, 136, 136]
Pixel 3 - Location: (2037, 937), Original RGB: [221, 195, 170], Stretched RGB: [221, 195, 170]
Pixel 4 - Location: (3897, 3331), Original RGB: [102, 119, 119], Stretched RGB: [102, 119, 119]
Pixel 5 - Location: (1011, 2587), Original RGB: [178, 153, 127], Stretched RGB: [178, 153, 127]
```

pixel_indices	pixel_locations	Var3	original_rgb_values	original_hex_colors	stretched_rgb_values	stretched_hex_colors
1	{[2379 1969]}	{'Roof' }	{[204 195 229]}	{'#CCC3E5'}	{[204 195 229]}	{'#CCC3E5'}
2	{[1623 1399]}	{'Tree' }	{[187 136 136]}	{'#BB8888'}	{[187 136 136]}	{'#BB8888'}
3	{[2037 937]}	{'River'}	{[221 195 170]}	{'#DDC3AA'}	{[221 195 170]}	{'#DDC3AA'}
4	{[3897 3331]}	{'Road' }	{[102 119 119]}	{'#667777'}	{[102 119 119]}	{'#667777'}

```
    5          {[1011 2587]}     {'Farm' }      {[178 153 127]}         {'#B2997F'}         {[178 153 127]}          {'#B2997F'}
```

Published with MATLAB® R2023a