IMAGE PROCESSING GROUP ASSIGNMENT

Coastline Changes

by Jonathan Berrett, Damian Wojtowich and Martyna Nieckarz

PRESENTATION OUTLINE

MAIN TOPICS

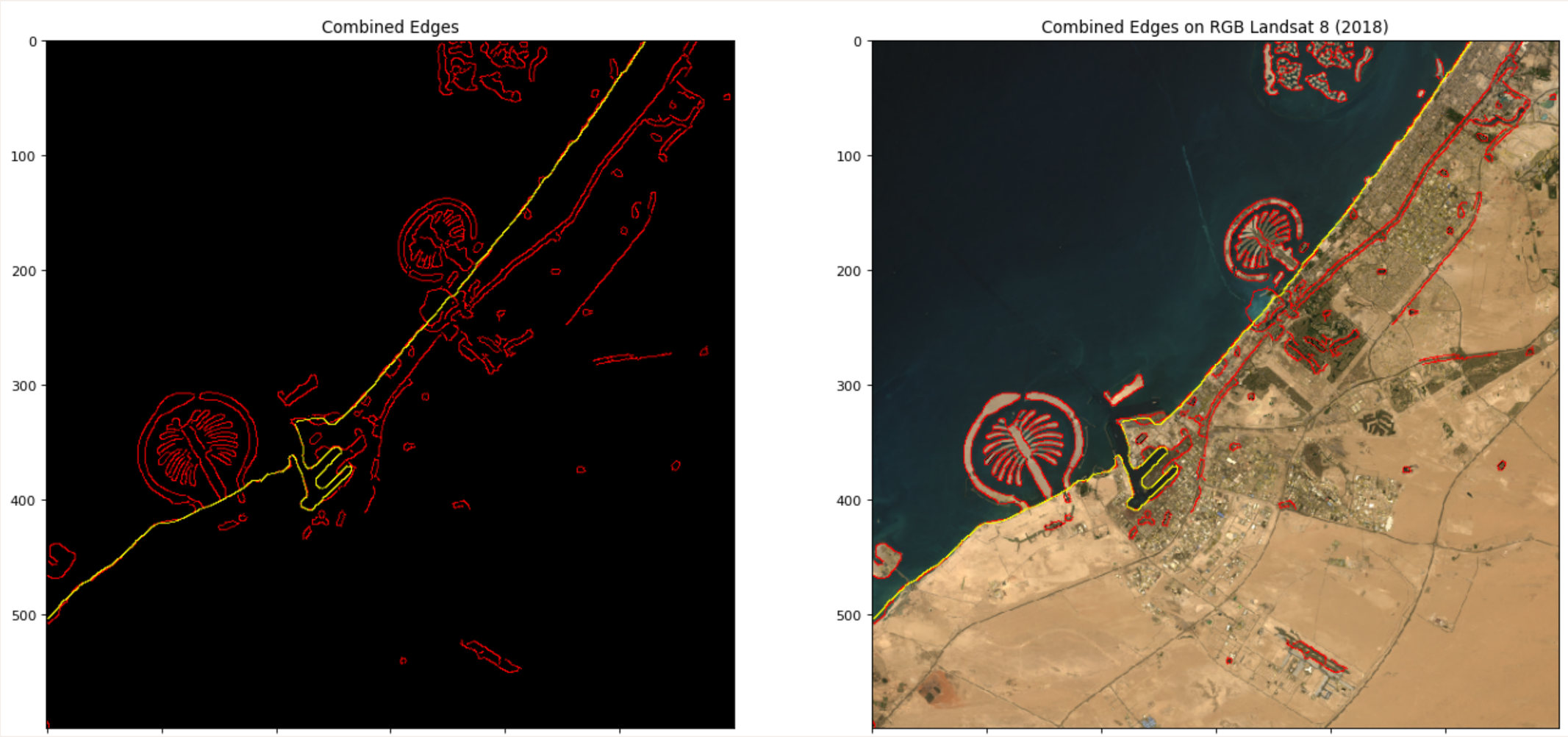
- Introduction
- Research
- Implementation
  - Image Retrieval
  - Segmentation
  - Coastline Detection
- How the program works
- Conclusion

Project Introduction

The goal of this project:

- Obtain historical and current image data from Landsat
- Automatically detect coastlines using those images
- Clearly show how they have changed over time

Final Images



Research & Existing Solutions

We began the project by doing research of existing solutions.

- First paper we have looked at studied the methods of detecting the coastline of Landsat images with image enhancement and segmentation techniques. Median filter was used.
- Another paper implemented an approach to automatically detect coastlines using edge detection and K-Means clustering.

Our Implementation

There are 3 stages of the project:

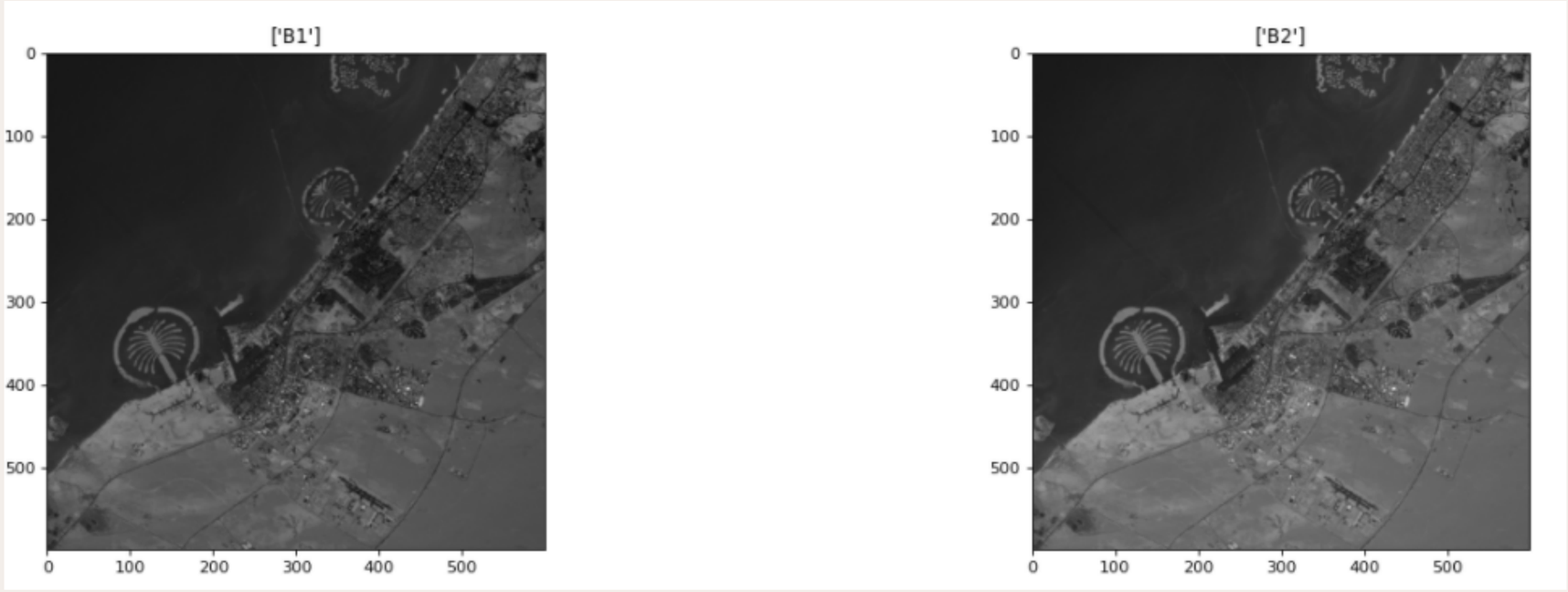
1. Image Retrieval
2. Segmentation
3. Coastline Detection

Image Retrieval

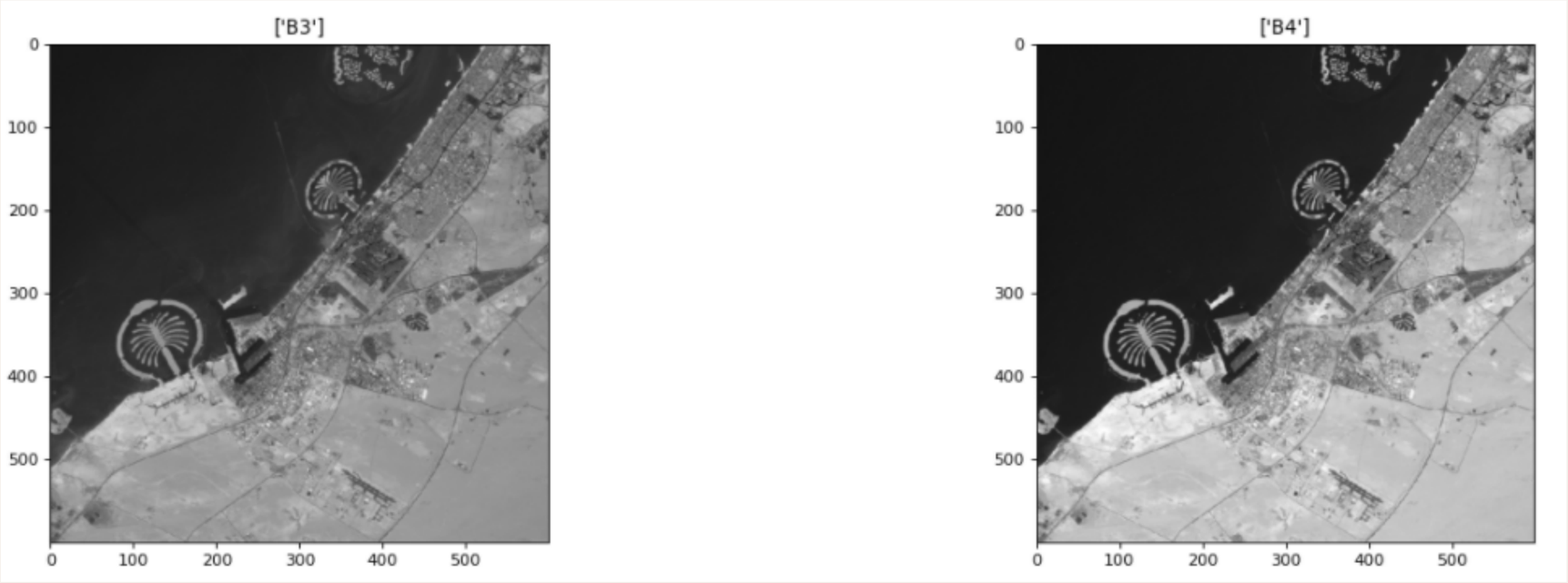
- The Landsat images were obtained via an Earth Engine API
- What is Landsat?
  - Group of various satellites used to gather data for images of the Earth's land surface and coastal regions.
  - Landsat 5: **six bands**, Landsat 8: **seven bands**
  - Bands measure different ranges of frequencies along the electromagnetic spectrum (e.g. Colour)
  - Meaning the program will be working with 13 images

```
""" ********************************************************************************************************
IMAGE RETRIEVAL
********************************************************************************************************
collections = {
'landsat5': 'LANDSAT/LT05/C01/T1_SR',
'landsat8': 'LANDSAT/LC08/C01/T1_SR'}

print("Starting retrieval")
# Center map around last drawn point
lastDrawnPoint = getLastDrawnPoint()
if lastDrawnPoint != None:
    zoom = 11
    longitude = lastDrawnPoint["longitude"]
    latitude = lastDrawnPoint["latitude"]
    print(longitude, latitude)
    regionOfInterestPoint = ee.Geometry.Point(longitude, latitude)


# Center map
roipCoords = regionOfInterestPoint.getInfo()["coordinates"]
Map.setCenter(roipCoords[0], roipCoords[1], 11)

# Select Landsat 5 collection
imageCollectionID = collections['landsat5']
startDate = '1990-01-01'
endDate = '1990-12-31'
bands = [['B1'], ['B2'], ['B3'], ['B4'], ['B5'], ['B7']]
landsat5_BandImages, landsat5_BandNames = getBandImagesFromSatellite(imageCollectionID, regionOfInterestPoint, startDate,

# Select Landsat 8 collection
imageCollectionID = collections['landsat8']
startDate = '2018-01-01'
endDate = '2018-12-31'
bands = [['B1'], ['B2'], ['B3'], ['B4'], ['B5'], ['B6'], ['B7']]
landsat8_BandImages, landsat8_BandNames = getBandImagesFromSatellite(imageCollectionID, regionOfInterestPoint, startDate,
```

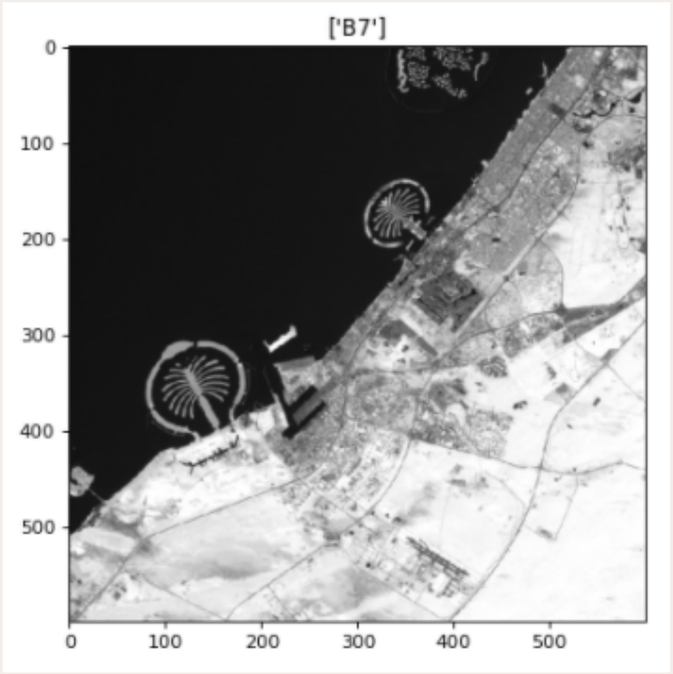
```
# Images have been retrieved so this variable is set to False. Otherwise going into fullscreen will invoke the
# algorithm.
imageRetrivalStarted = False
easygui.msgbox("Finished, Please exit fullscreen")

# Get RGB Image from the Red, Green and Blue band images
landsat5_RGB = getRGBFromImages(landsat5_BandImages[0], landsat5_BandImages[1], landsat5_BandImages[2])
landsat8_RGB = getRGBFromImages(landsat8_BandImages[1], landsat8_BandImages[2], landsat8_BandImages[3])

displayImages([landsat5_RGB, landsat8_RGB], ["landsat5_RGB", "landsat8_RGB"], 9, 2)

# Display Band Images
# displayImages(landsat5_BandImages, landsat5_BandNames, 9, 2)
displayImages(landsat8_BandImages, landsat8_BandNames, 9, 2)

combinedImages = []
combinedImageTitles = []
```

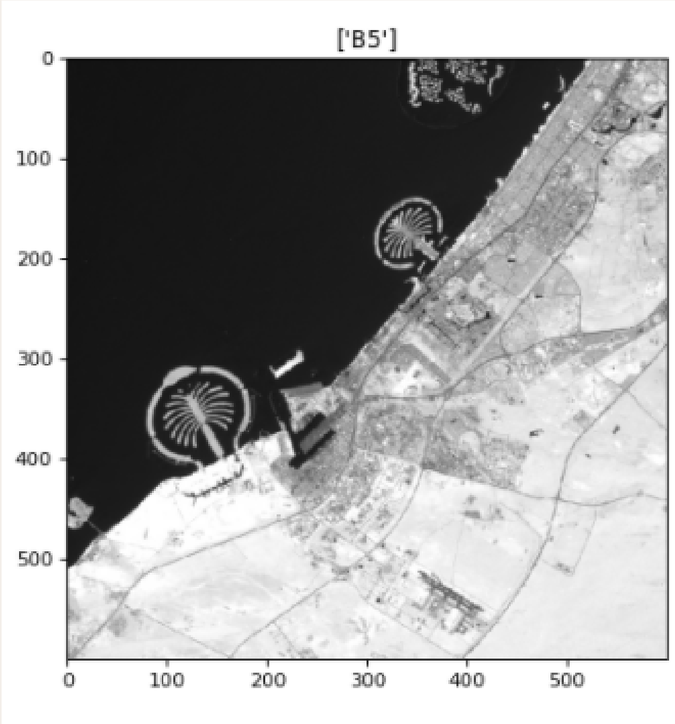


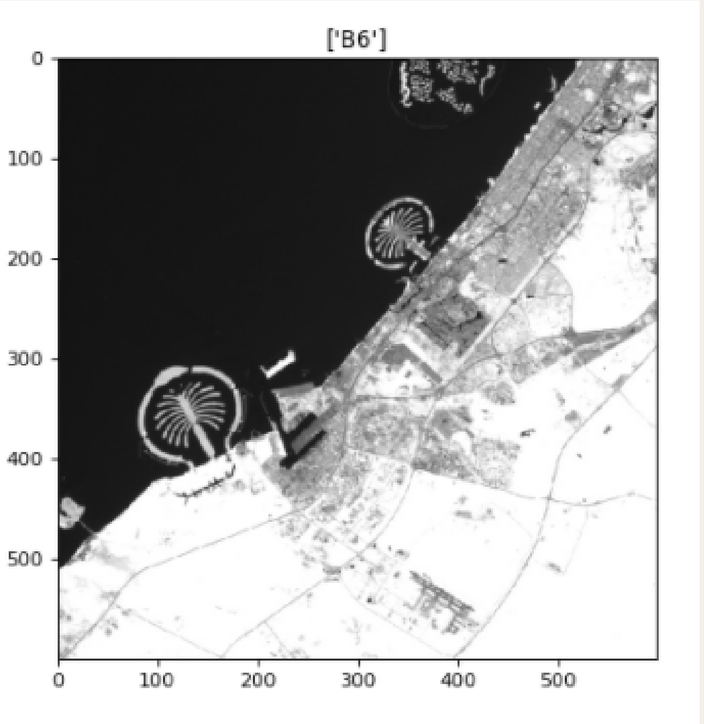
Landsat 8 Bands 1 to 4



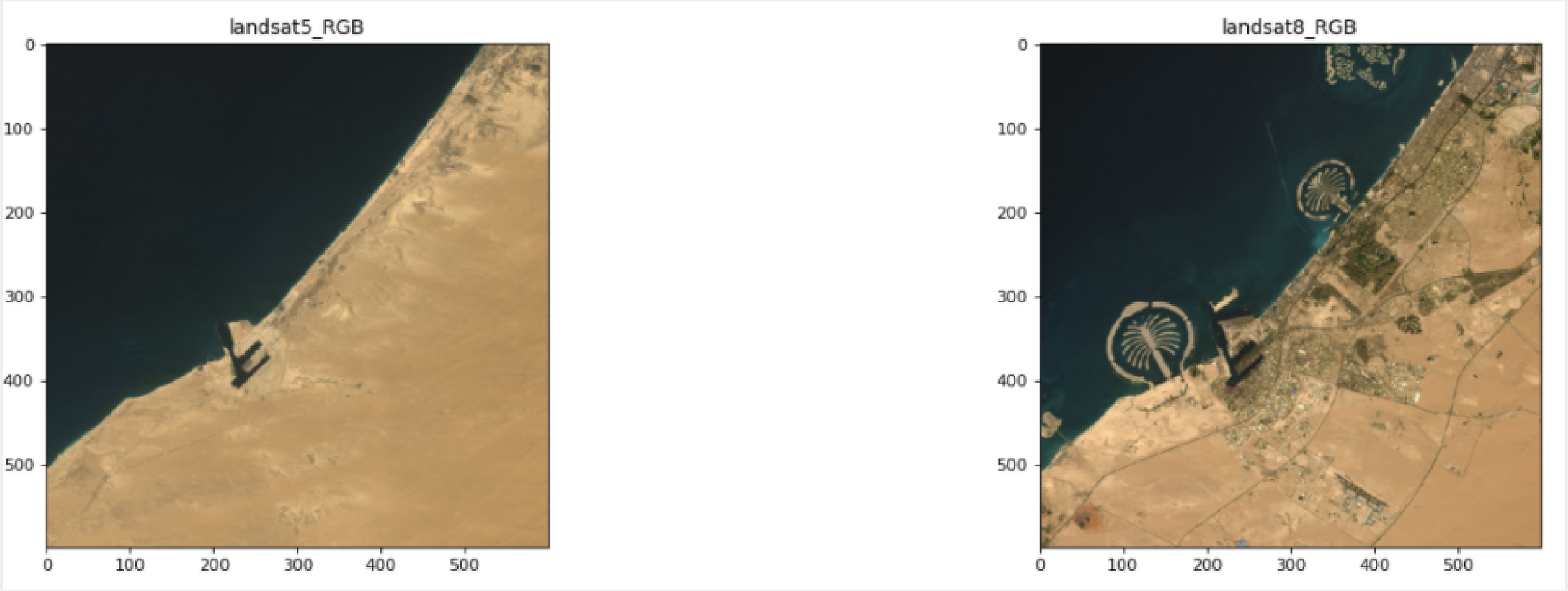
Landsat 8 Bands 5 to 7







Combined RGB Landsat 5 & 8 Images



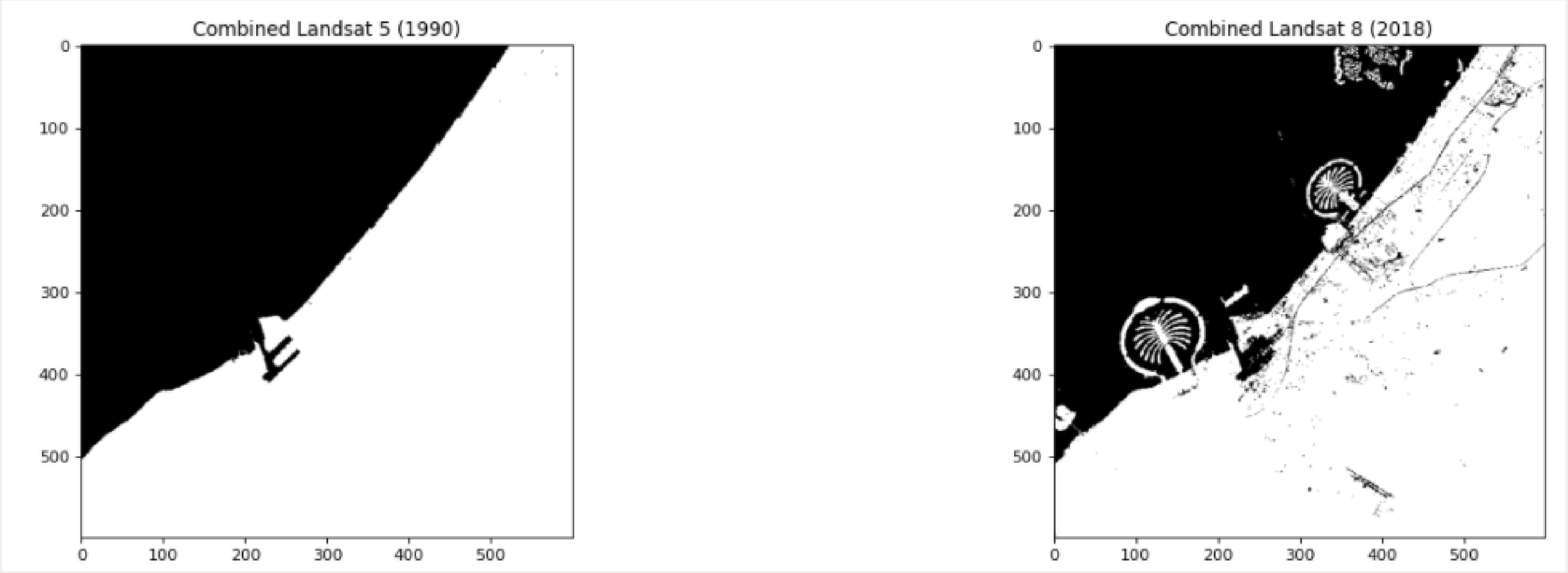
Segmentation

- After getting the image bands K-Means clustering is performed on them.
- K-Means segmentation was used to separate the water from land in each image
- Image bands are converted to binary images
- Using bitwise_or function bands are put together creating one image of Landsat 8 and one of Landsat 5
- Those combined image will be blurred with a Gaussian blur to reduce any noise

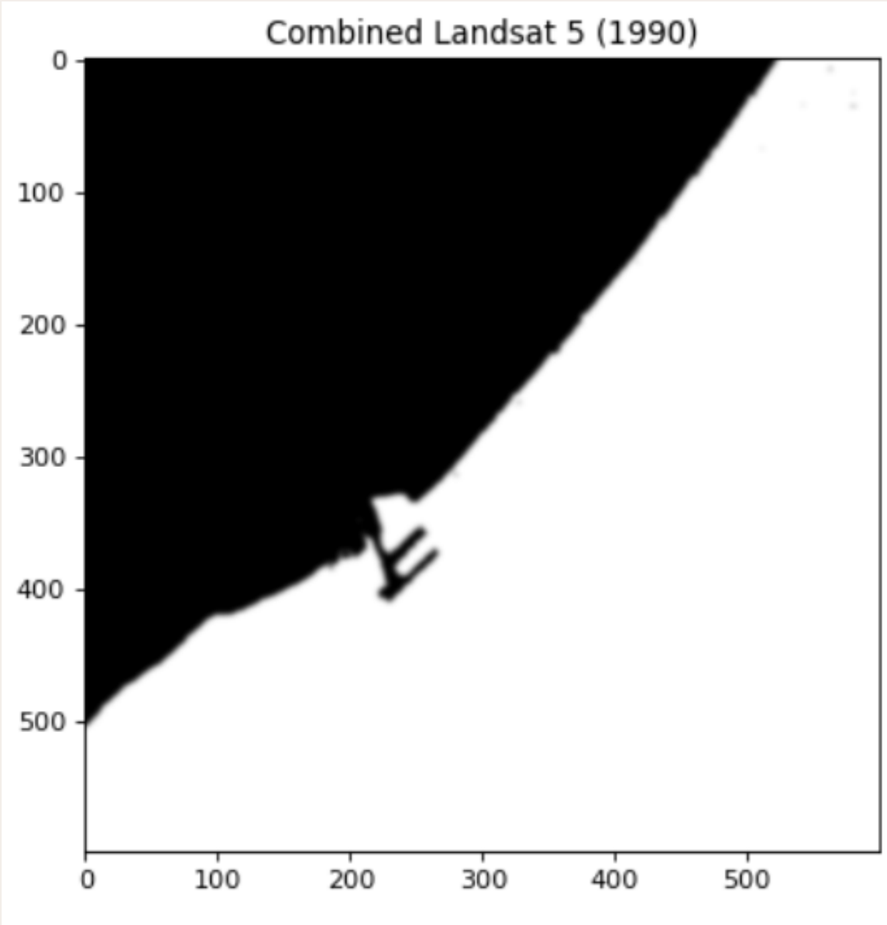
```
""" ********************************************************************************************************************
SEGMENTATION
********************************************************************************************************************

kmeansImages = getKmeansImages(landsat5_BandImages.copy())
combinedImage, binaryImages = getBinaryCombinedImage(kmeansImages)

combinedImageTitles.append("Combined Landsat 5 (1990)")
combinedImages.append(combinedImage)

# Display K-Means Images
# displayImages(kmeansImages, landsat5_BandNames, 9, 2)

# Display K-Means Images in binary
# displayImages(binaryImages, landsat5_BandNames, 9, 2)

# ------------------------------------------------------------------------------------------------------------------

kmeansImages = getKmeansImages(landsat8_BandImages.copy())
combinedImage, binaryImages = getBinaryCombinedImage(kmeansImages)

combinedImageTitles.append("Combined Landsat 8 (2018)")
combinedImages.append(combinedImage)

# Display K-Means Images
# displayImages(kmeansImages, landsat8_BandNames, 9, 2)

# Display K-Means Images in binary
displayImages(binaryImages, landsat8_BandNames, 9, 2)
```

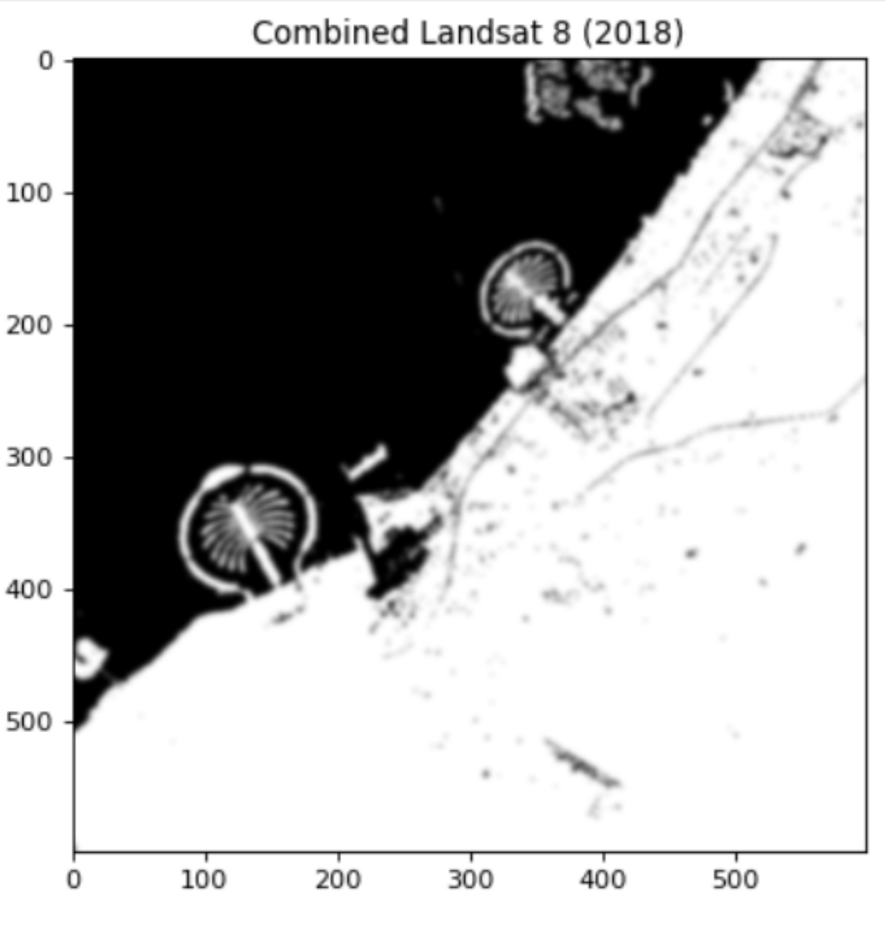
```
# ------------------------------------------------------------------------------------------------------------

displayImages(combinedImages, combinedImageTitles, 9, 2)

blurredImages = [cv2.GaussianBlur(combinedImage, (11,11), 0) for combinedImage in combinedImages]
titles = ["Blurred Combined Landsat 5 (1990)", "Blurred Combined Landsat 8 (2018)"]
displayImages(blurredImages, combinedImageTitles, 9, 2)

# Morphology test as comparison (Not used as the technique is inferior to blurring)
morphClosedImages = []
for combinedImage in combinedImages:
    shape = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3))
    # Morphological is used to denoise the image
    closedImage = cv2.morphologyEx(combinedImage, cv2.MORPH_CLOSE, shape)
    morphClosedImages.append(closedImage)

# Area counter
landsat5_area = 0
landsat8_area = 0
h, w = combinedImages[0].shape
for row in range(h):
    for col in range(w):
        landsat5_pix = combinedImages[0][row][col]
        landsat8_pix = combinedImages[1][row][col]

        if landsat5_pix == 255:
            landsat5_area += 1
        if landsat8_pix == 255:
            landsat8_area += 1

print(landsat5_area, landsat8_area)
if landsat5_area > landsat8_area:
    areaPercentage = (landsat5_area - landsat8_area) / landsat5_area
    print("The area in the image has decreased by", str(areaPercentage) + "%")
if landsat5_area < landsat8_area:
    areaPercentage = (landsat8_area - landsat5_area) / landsat5_area
    print("The area in the image has increased by", str(areaPercentage) + "%")
```

Combined K-Means Images



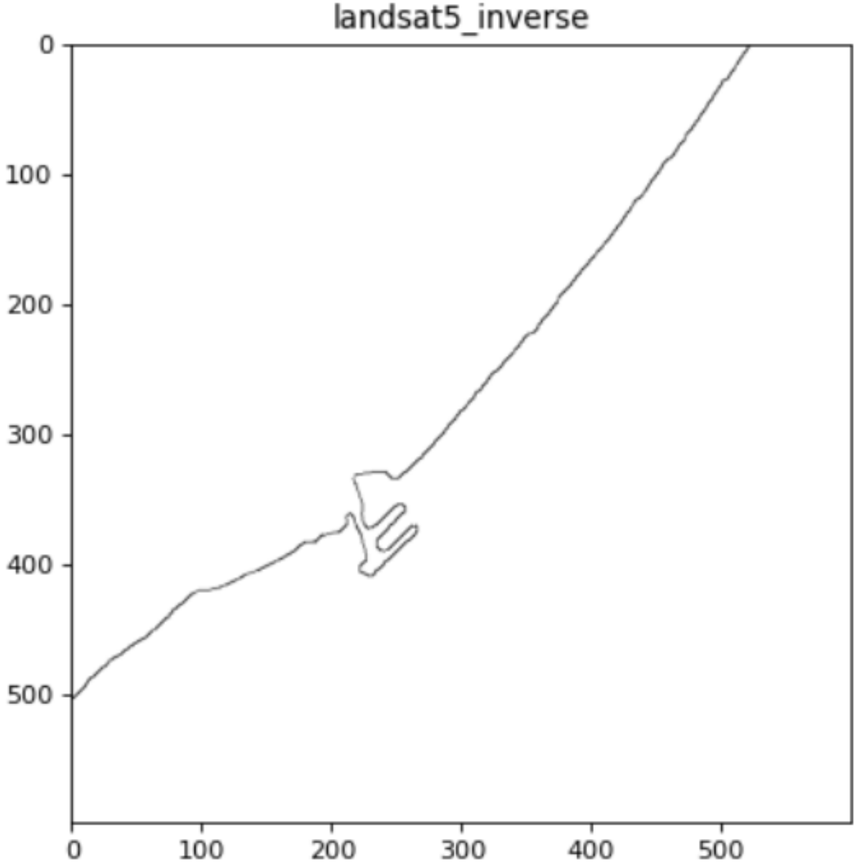
Gaussian filtered Landsat 5 & 8 Images

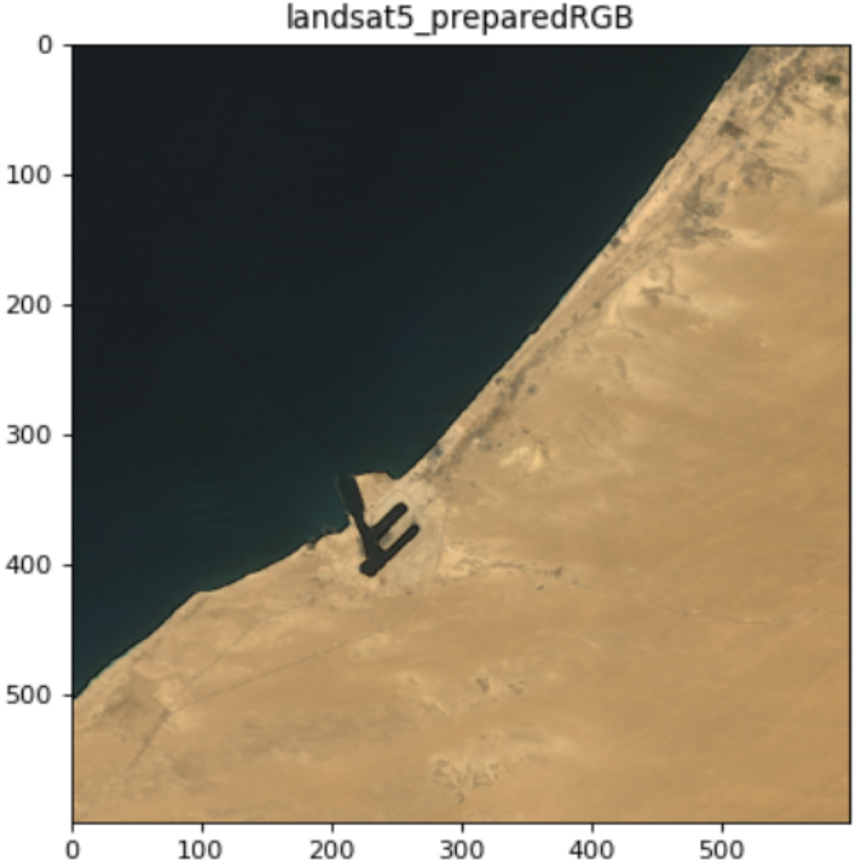




Coastline Detection

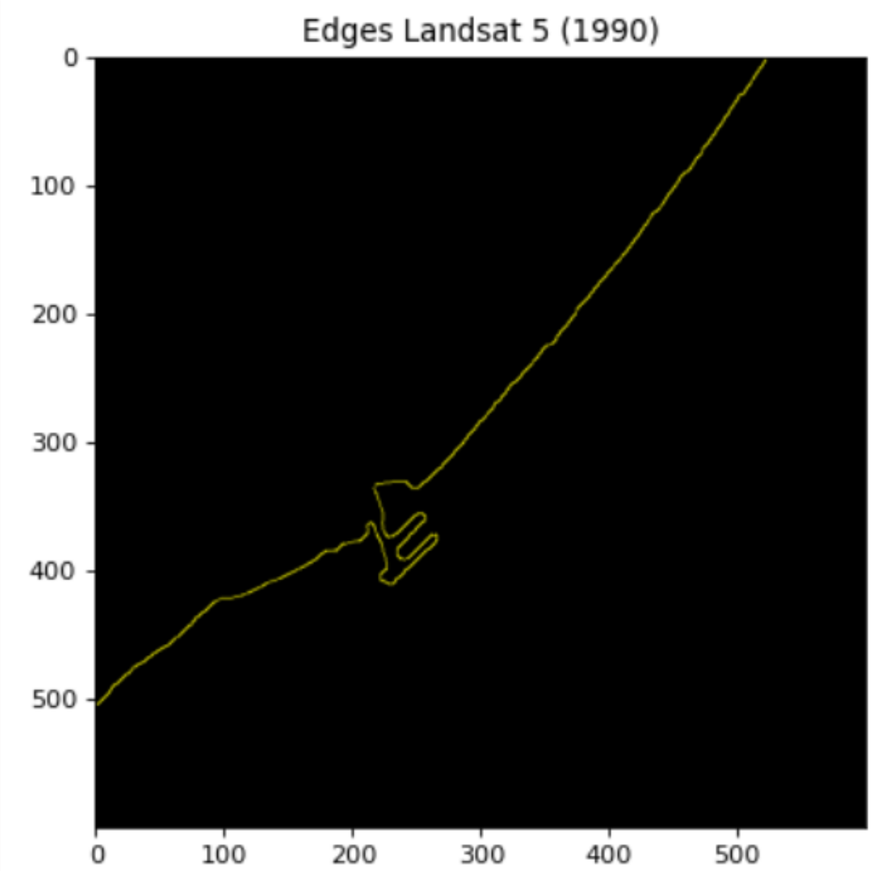
- Using the two combined images (one from Landsat 5 & one from Landsat 8) coastlines will be obtained of both using Canny edge detection
- Contours will be redrawn in a different colour using Contours functions.
- The contours will be displayed on top of the RGB images
- Finally, the contours will be combined and drawn on top of the present day image to show how much the coastline has changed

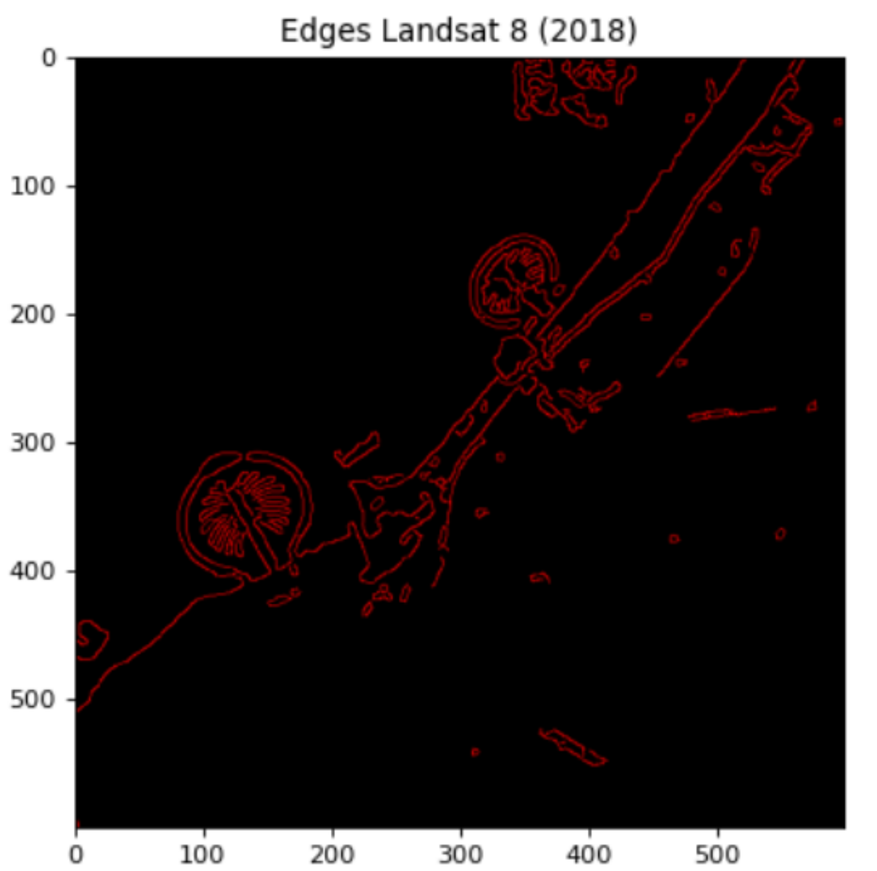

landsat5_inverse
0
100
200
300
400
500
0
100
200
300
400
500


landsat5_preparedRGB
0
100
200
300
400
500
0
100
200
300
400
500

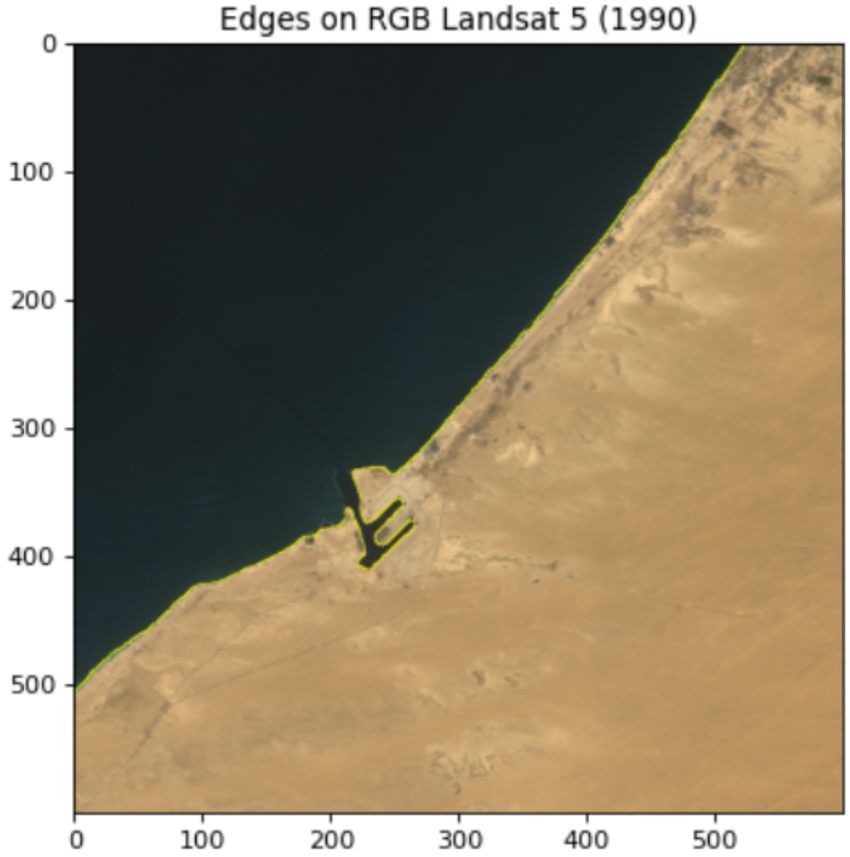
```
# Get contours and redraw with a different colour
# edged image is just the image from Canny Edge detection.
# contoured image is the same thing but redrawn with a different colour
landsat5_edged, landsat5_contoured = obtainAndDrawingEdges(blurredImages[0], (255, 255, 0))
landsat8_edged, landsat8_contoured = obtainAndDrawingEdges(blurredImages[1], (255, 0, 0))

# Combine the contours with their respective RGB satellite image
# Landsat 5
# An inverse mask is created so that the coloured contours stand out and do no blend with the image
landsat5_inverse = ~landsat5_edged
# The pixels where the contours go on top of the RGB image are turned black to prevent blending
landsat5_preparedRGB = cv2.bitwise_and(landsat5_RGB, landsat5_RGB, mask = landsat5_inverse)
# Contoured image and the RGB image (post mask) are combined
edgesOnRGB_landsat5 = cv2.bitwise_or(landsat5_contoured, landsat5_preparedRGB)

# Landsat 8
landsat8_inverse = ~landsat8_edged
landsat8_preparedRGB = cv2.bitwise_and(landsat8_RGB, landsat8_RGB, mask = landsat8_inverse)
edgesOnRGB_landsat8 = cv2.bitwise_or(landsat8_contoured, landsat8_preparedRGB)

# Put both contours on the most recent RGB satellite image (landsat 8 RGB + Both contours)
# Contours are combined
combinedContours = cv2.bitwise_or(landsat5_contoured, landsat8_contoured)
# An inverse mask is used to prepare the RGB image for the contours to go on top
combinedContoursMask = cv2.bitwise_or(landsat5_edged, landsat8_edged)
inverseCombinedMask = ~combinedContoursMask
landsat8_RGB_final = cv2.bitwise_and(landsat8_RGB, landsat8_RGB, mask = inverseCombinedMask)
final = cv2.bitwise_or(combinedContours, landsat8_RGB_final)
```

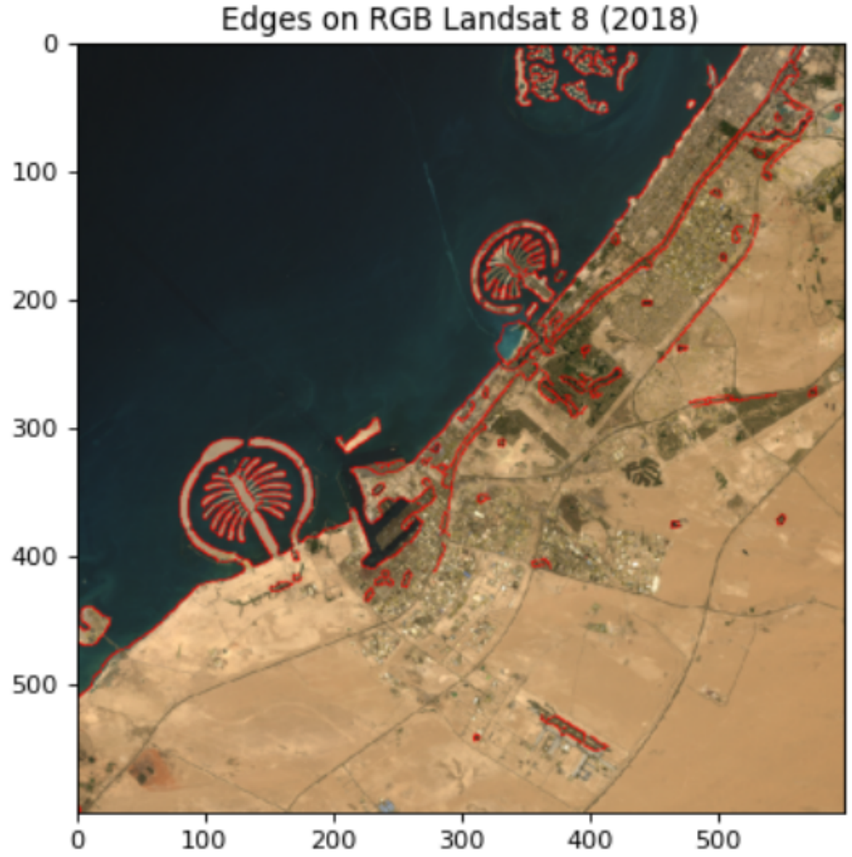
Contoured Edges of Landsat 5 & 8 Images



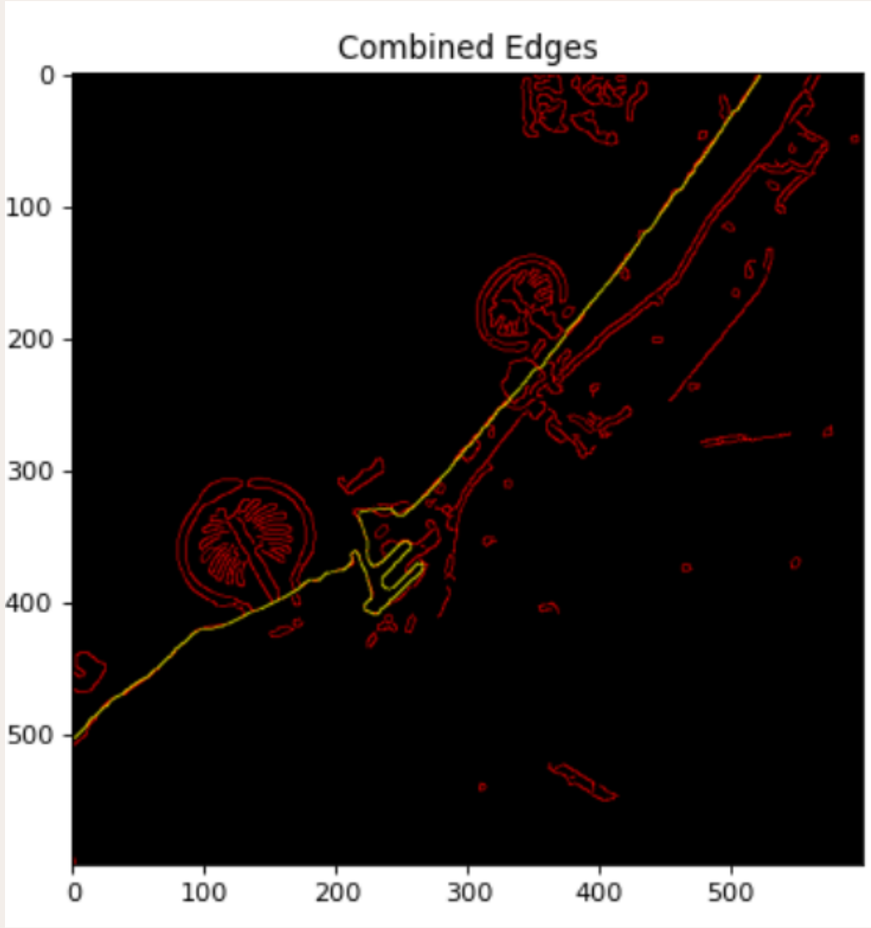


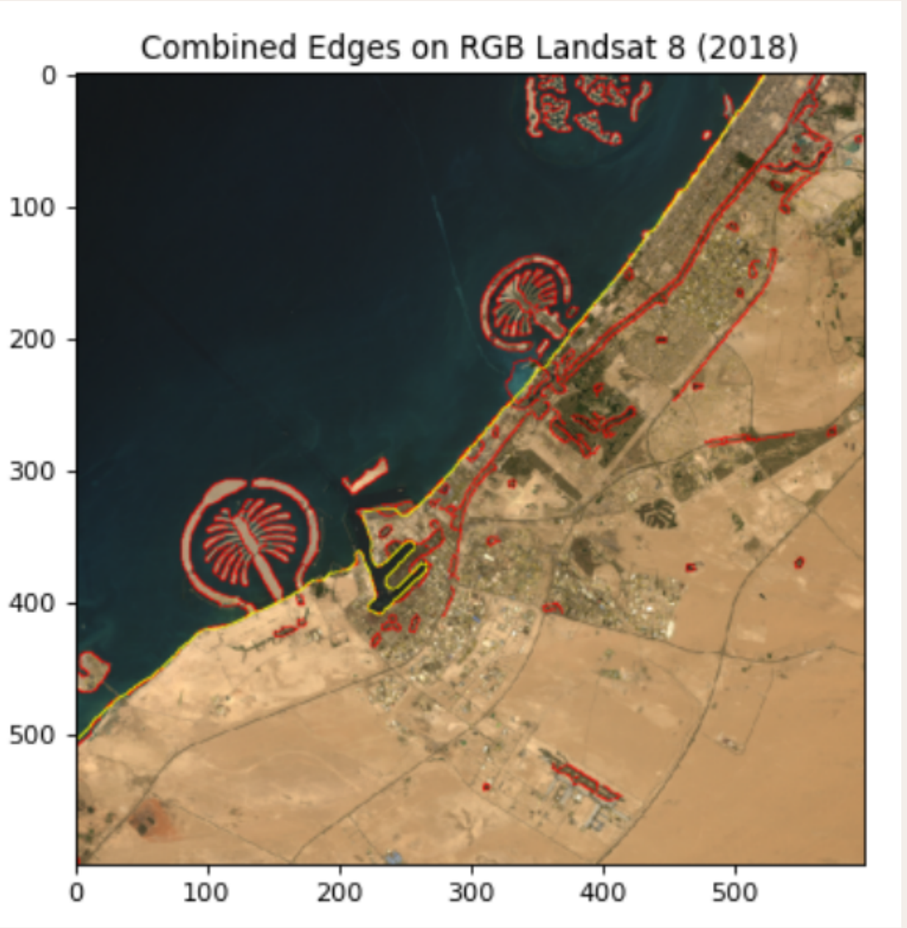
Contoured Edges on corresponding RGB Images





Combined Contours on Landsat 8 RGB Image

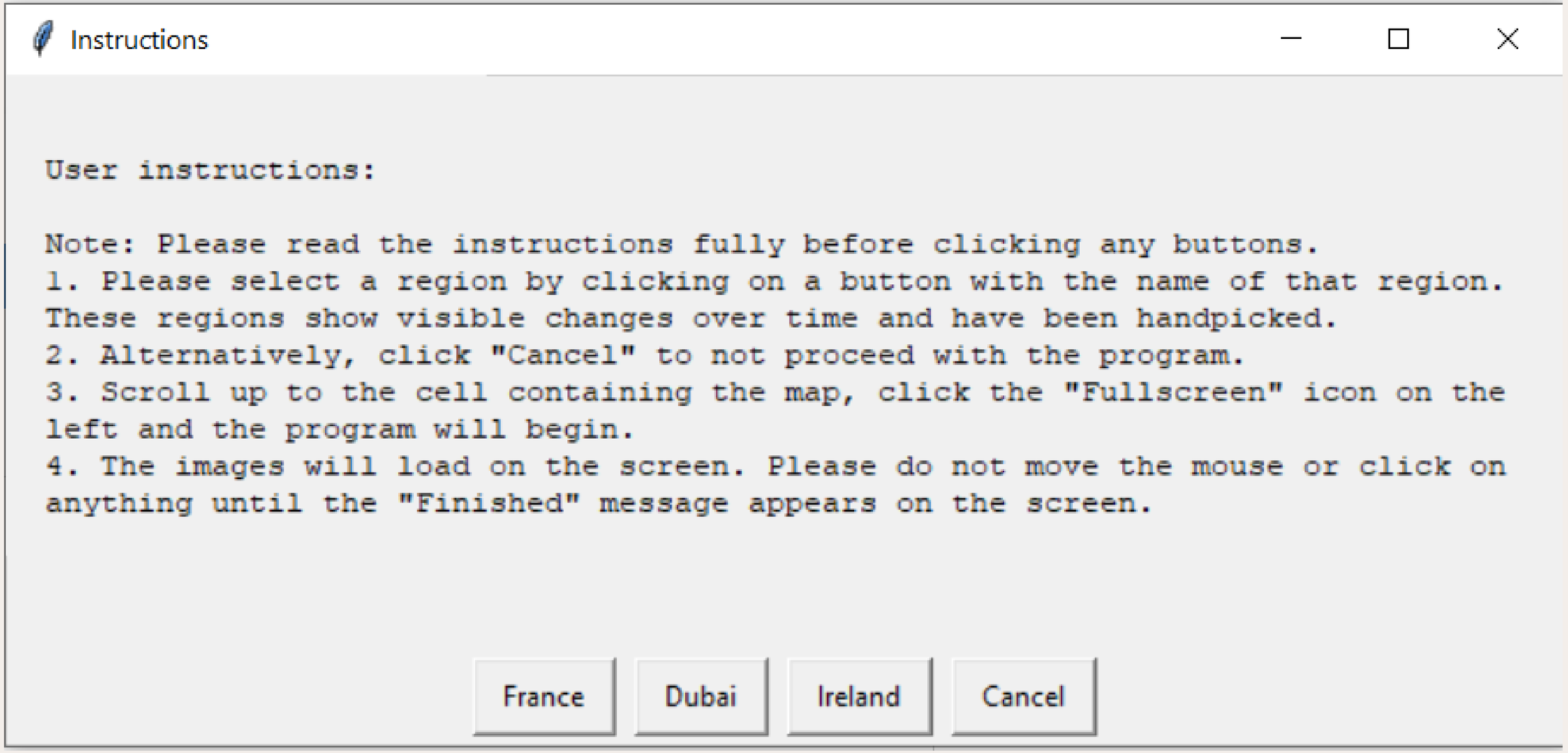




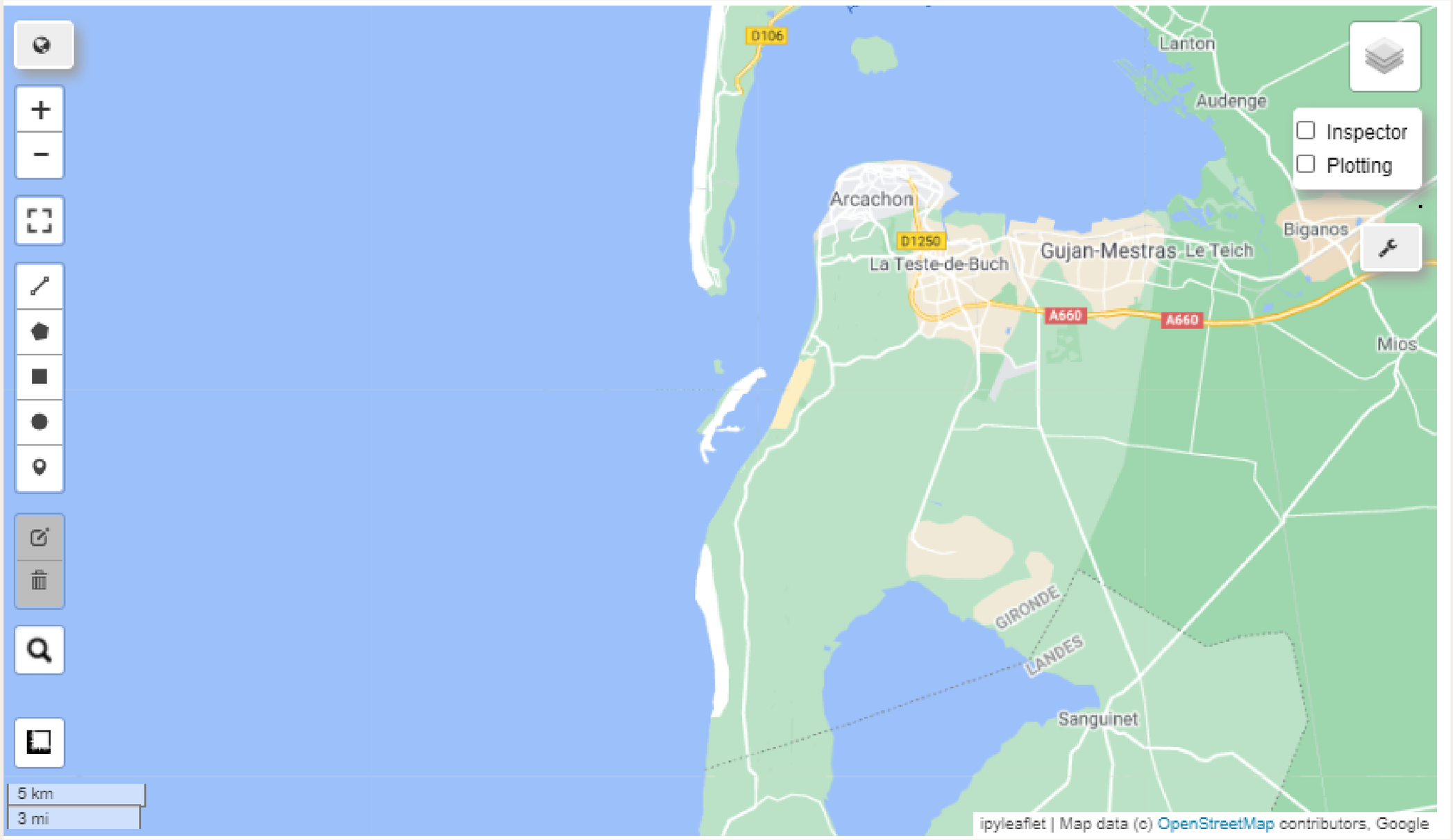
How the program works

- When the program is run the user will be given instructions and a choice between 3 hand-picked locations: France, Dubai and Ireland. Those locations show clear cloastline changes over the years.
- After choosing, the user will press the full-screen icon on the map and will wait until all band images are obtained.
- The user can leave the fullscreen and the algorithm will be performed on all the images
- Lastly, the results images will be displayed at the bottom of the program.

User Instructions



The map


D106
Lanton
Audenge
Inspector
Plotting
Arcachon
D1250
La Teste-de-Buch
Gujan-Mestras
Le Teich
Biganos
A660
A660
Mios
GIRONDE
LANDES
Sanguinet
5 km
3 mi
ipyleaflet | Map data (c) OpenStreetMap contributors, Google

Code for generating the map

```
"""
The map is generated and displayed on screen for the user to interact with. This method centers the map on a region.
"""
def generateMap(longitude, latitude, zoom):
    global regionOfInterestPoint
    if zoom == None:
        zoom = 11

    if longitude == None or latitude == None:
        Map = geemap.Map(center=(44.580014, -1.248944), zoom=zoom)
        regionOfInterestPoint = ee.Geometry.Point(-1.248944, 44.580014)
    else:
        Map = geemap.Map(center=(latitude, longitude), zoom=zoom)
        regionOfInterestPoint = ee.Geometry.Point(longitude, latitude)

    return Map
```

Conclusion

- Design requirements have been fully satisfied and the desired output was achieved
- Program works as expected on all tested regions
- Overall, our team is very satisfied with the results of this group project. The coastlines are being detected very accurately.