HACETTEPE UNIVERSITY

DEPARTMENT OF ELECTRICS and ELECTRONICS ENGINEERING

ELE492 IMAGE PROCESSING

MIDTERM EXAMINATION REPORT

NAME : **Ziya Tuna**

SURNAME : **Bölükbaşı**

NUMBER : **21628166**

SPRING 2023

Question-1:

Using the algorithms you learned in class, and also the algorithms you can research; devise astructured way to enhance this image and explain it. Plot the framework of the algorithms (whichone comes after the other), and explain why this process has been chosen.

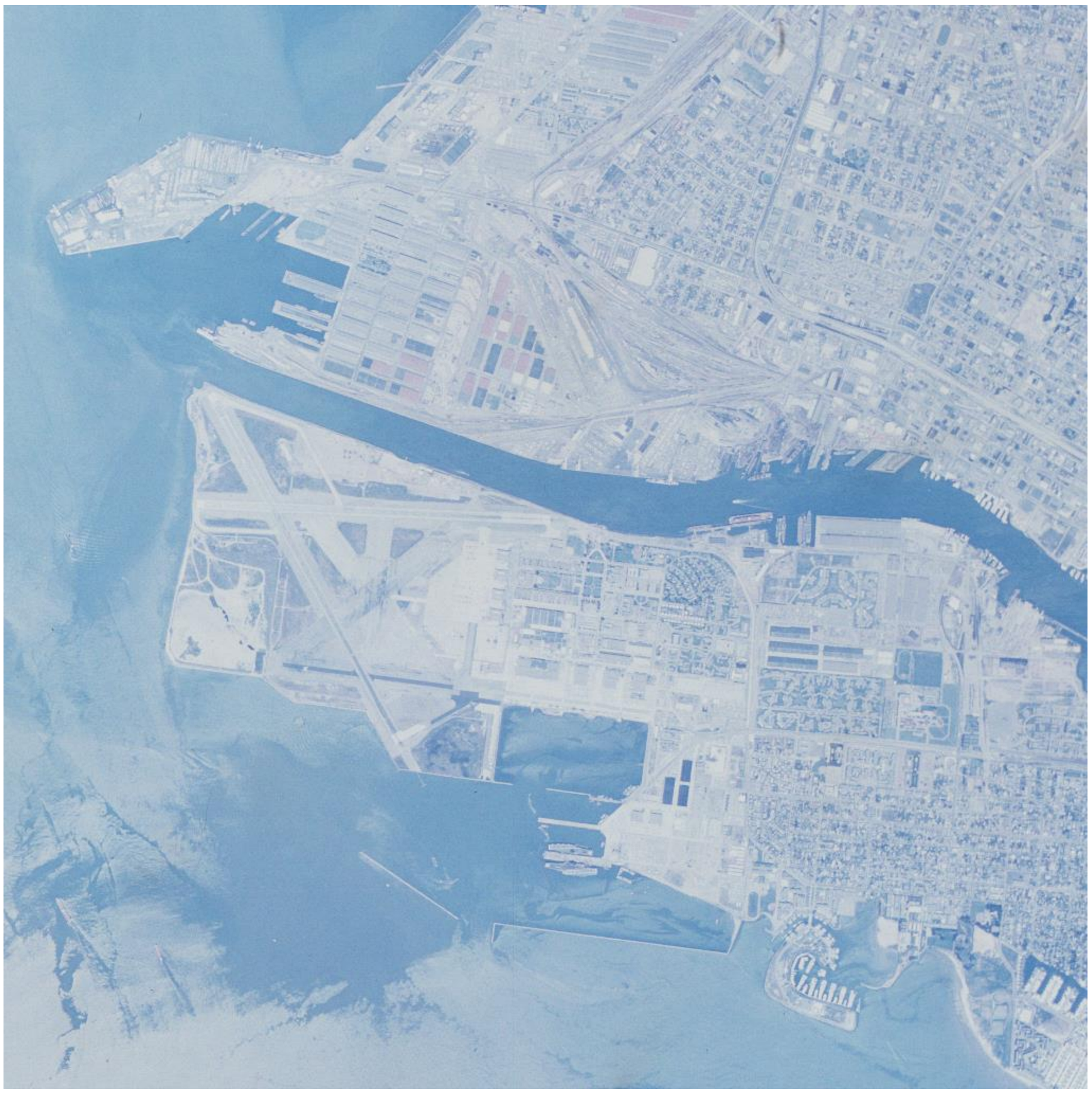

Figure 1: Image to be used for Question 1

This is my answer:

To solve this problem, I used the PIL library.First, I increased the brightness of the image by 1.2 times using the ImageEnhance.Brightness() function.Next, I increased the color of the image by 1.5 times using the ImageEnhance.Color() function.Next, I increased the contrast of the image by 1.5 times using the ImageEnhance.Contrast() function.Finally, I increased the sharpness of the image by 1.3 times using the ImageEnhance.Sharpness() function.Finally, I applied a Gaussian blur filter.

Output

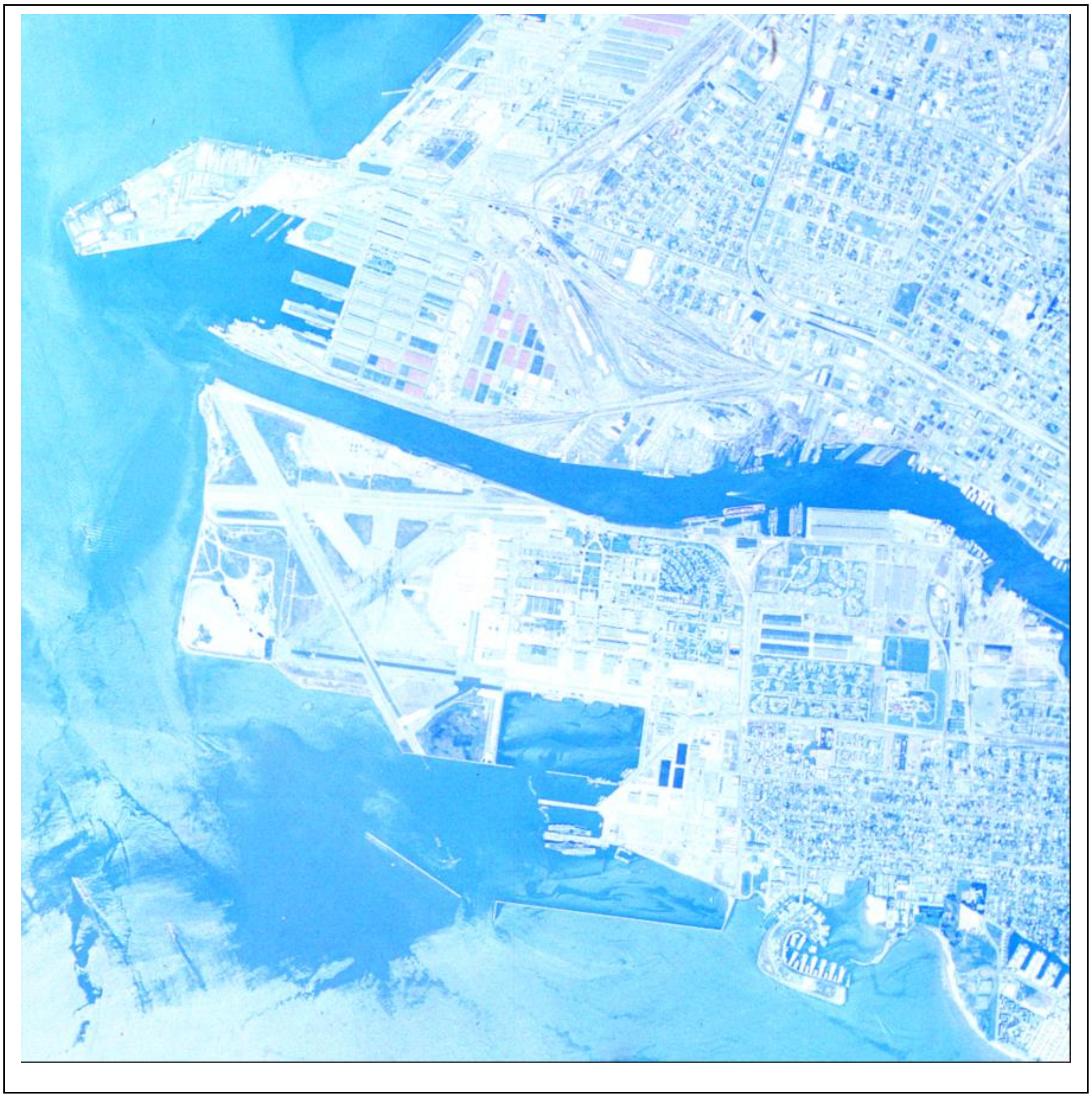

Figure 2:Enhanced Image

Question-2:

The image haze.png has been used in several publications dealing with the problem of removing fog from an image (dehazing)

Figure 3:Image to be used for Question 2

This is my answer:

I used the image_dehazer library.First of all, I determined the required values for the function that will perform the de-fogging process. I did multiple experiments to find these values:

airligthEstimation_windowSze: Window used to estimate the airlight.

boundaryConstraint_windowSze: Size of the window used to apply boundary constraints.

C0: Minimum value of the haze transmission map.

C1: A parameter that controls the slope of the transmission map.

regularize_lambda: A regularization parameter that controls the smoothness of the transmission map.

sigma: The standard deviation of the Gaussian filter used to estimate the aurora.

delta: A parameter that controls the strength of the soft thresholding process used to estimate the transmission map.

showHazeTransmissionMap: A flag that determines whether to display the haze transmission map.

Output:

Figure 4:Dehaze Image

Question-3:

Devise a way to detect and count the number of pools in the following image.

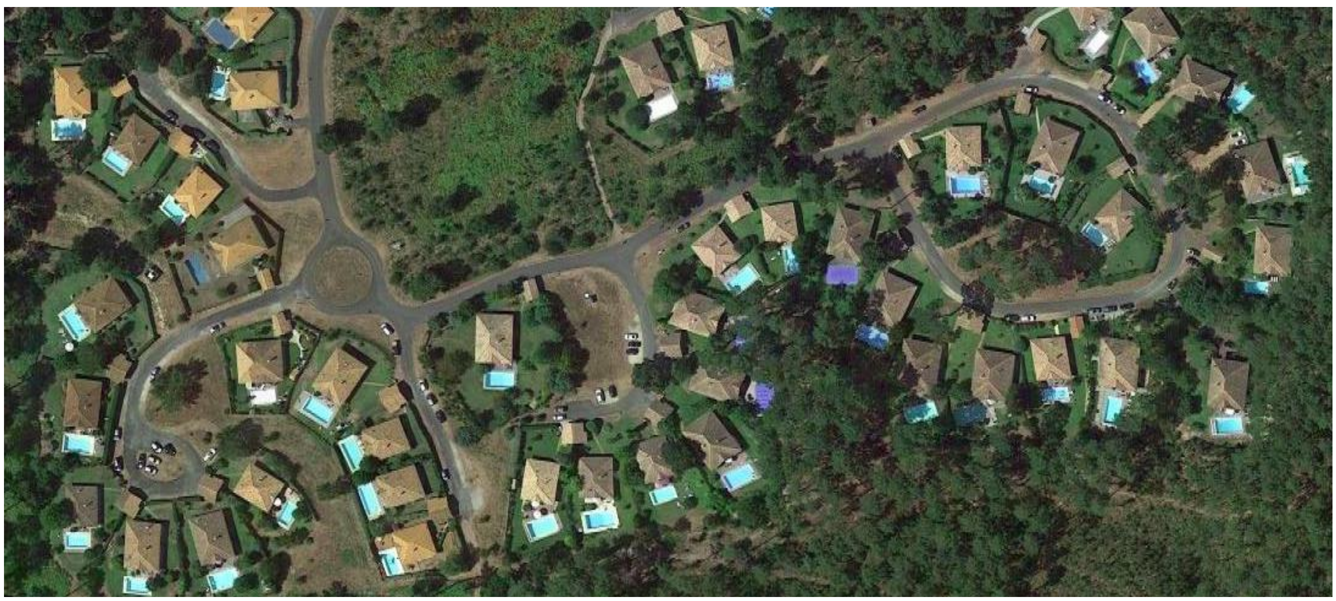

Figure 5: Image to be used for Question 3

This is my answer:

In this question, I followed these steps to identify the pools in the photo:

First I converted the photo to HSV color space, using the cvtColor function in the Opencv library.Secondly, I determined the blue color range in which the pools can be identified with rgb codes and defined them in arrays with the numpy library.Then, with these arrays I created, I created a mask with the inRange function from the Opencv library.Then I applied morphological operations to reduce noise using the morphologyEx function from the opencv library.I found the contours of the blue regions, i.e. pools, with the findContours function from the Opencv library.I used the Rectangle function from the opencv library to enclose the pools in a rectangle. While doing this, I increased the counter variable, which I initially defined as 0, by one and thus found the number of pools.Finally, I opened the image showing the rectangled pools.

Output:

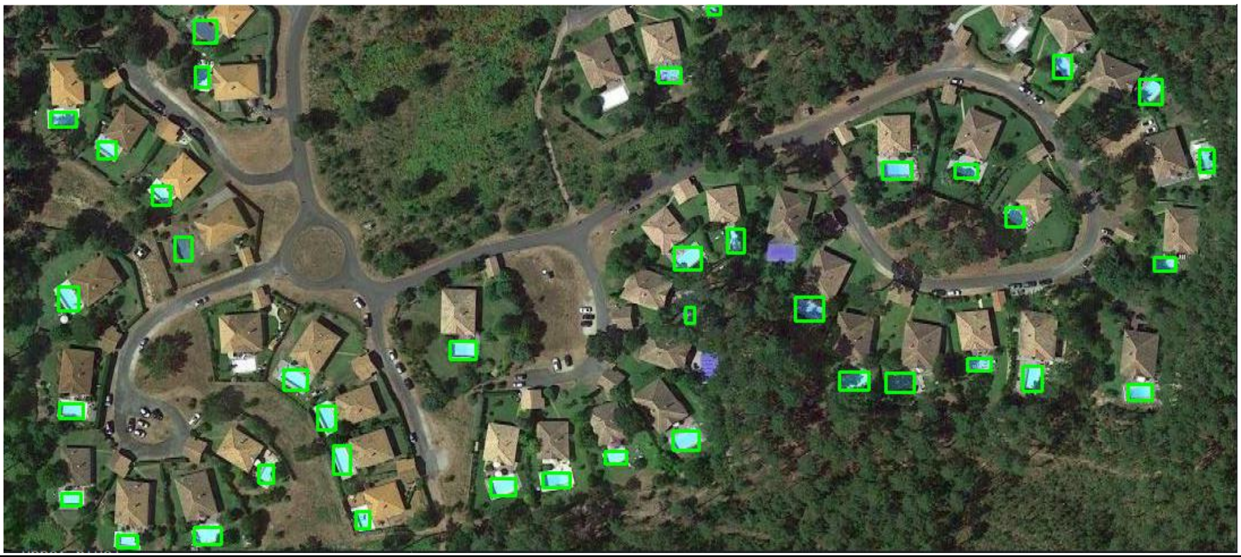

Figure 6: Pools enclosed in a rectangle

```
How many pools are there in this image: 39
```

Figure 7: Output showing how many pools are in the picture

Appendix

1)

```
from PIL import Image
from PIL import ImageEnhance, ImageFilter

image = Image.open('2.2.07.tiff')
image.show()

curr_bri = ImageEnhance.Brightness(image)
new_bri = 1.2
img_brightened = curr_bri.enhance(new_bri)

curr_col = ImageEnhance.Color(img_brightened)
new_col = 1.5
img_colored = curr_col.enhance(new_col)

curr_con = ImageEnhance.Contrast(img_colored)
new_con = 1.5
img_contrasted = curr_con.enhance(new_con)

curr_sharp = ImageEnhance.Sharpness(img_contrasted)
new_sharp = 1.3
img_sharped = curr_sharp.enhance(new_sharp)

filtered_img =
img_sharped.filter(ImageFilter.GaussianBlur(radius=0.5))

filtered_img.show()
```

2)

```
import image_dehazer
import cv2

HazeImg = cv2.imread('haze.png')
airlightEstimation_windowSze = 35
boundaryConstraint_windowSze = 9
C0 = 1
C1 = 500
regularize_lambda = 0.3
sigma = 0.1
delta = 0.9
showHazeTransmissionMap = False

HazeCorrectedImg, HazeMap =
image_dehazer.remove_haze(HazeImg,
airlightEstimation_windowSze,

boundaryConstraint_windowSze, C0, C1,

regularize_lambda, sigma, delta,

showHazeTransmissionMap)


cv2.imshow('input image', HazeImg)
cv2.imshow('enhanced_image', HazeCorrectedImg)
cv2.waitKey(0)
```

3)

```
import cv2
import numpy as np

img = cv2.imread('moliets.png')
counter=0

hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
lower_blue = np.array([80,50,50])
upper_blue = np.array([115,255,255])

mask = cv2.inRange(hsv, lower_blue, upper_blue)

kernel = np.ones((5,5),np.uint8)
mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)

contours, _ = cv2.findContours(mask, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

for cnt in contours:
    x, y, w, h = cv2.boundingRect(cnt)
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
    counter+=1
cv2.imshow('Result', img)
print('How many pools are there in this image:',counter)
cv2.waitKey(0)
cv2.destroyAllWindows()
```