report

March 19, 2023

1 Question 1.

a. Enhance the image with Laplacian filters if they can find the hidden objects and show the results. Explain their success or failure. - Laplacian filters are used to enhance edges and detect edges in an image.

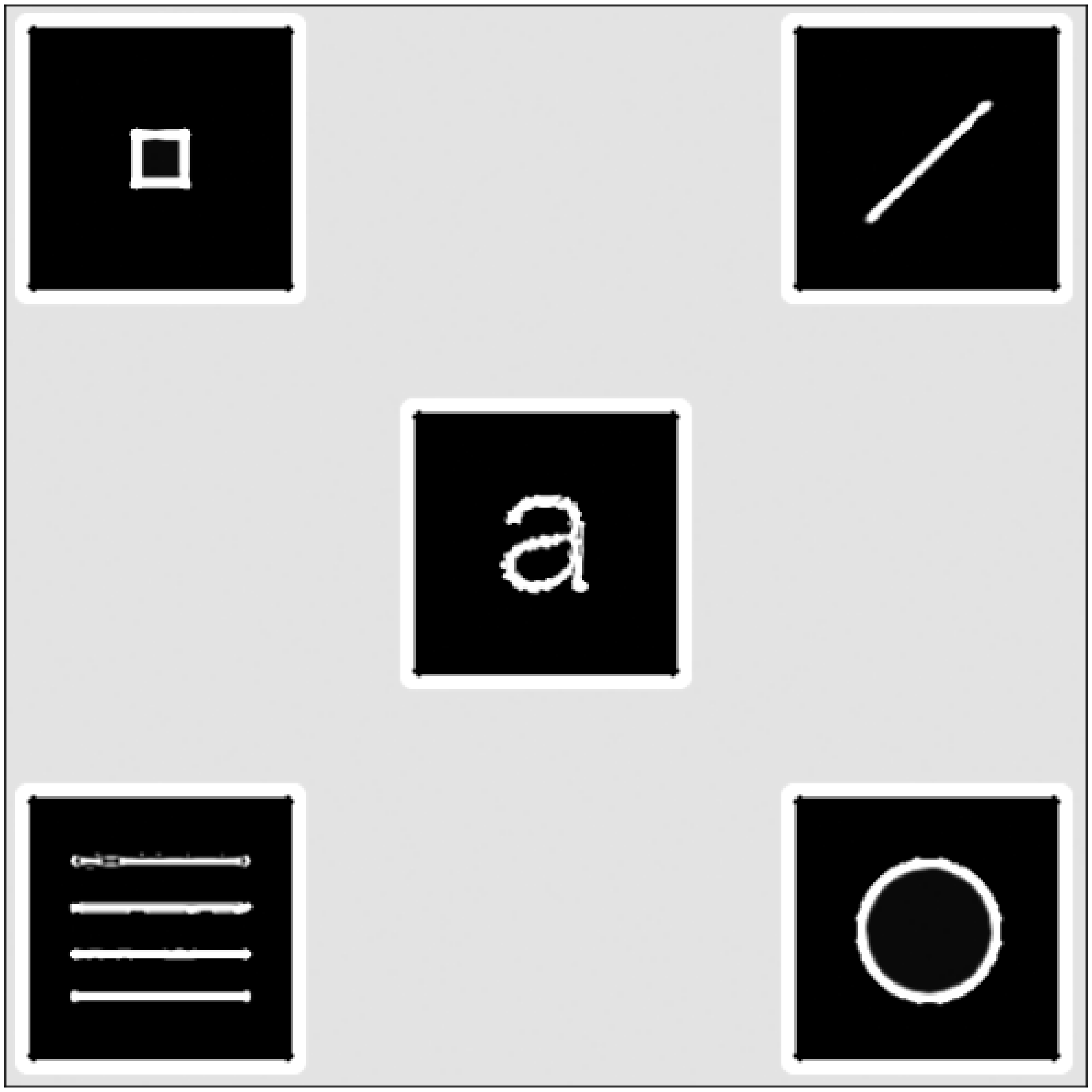
- The Laplacian filter works by finding the second derivative of the image intensity function. - It enhances areas of the image where the intensity changes quickly, such as edges or boundaries between - different objects in the image. Therefore, Laplacian filters can be effective at revealing hidden objects in an image.

[2]:
```
#  Steps
- Load the image
- Convert the GaussianBlur image
- Apply the Laplacian filter
- Subtract the normalized Laplacian filter from the original image


# Result show
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
```

[2]:
```
(Text(0.5, 1.0, 'Result = original image - Laplacian filter '),
 ([], []),
 ([], []))
```

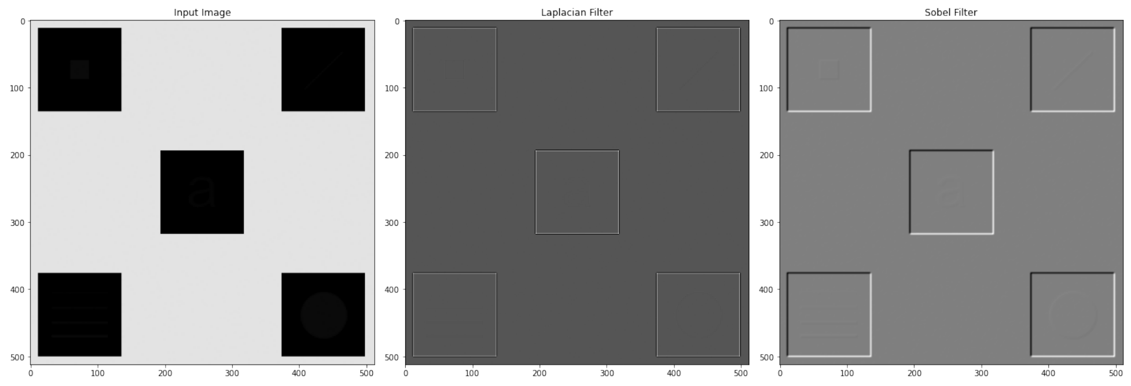
Result = original image - Laplacian filter

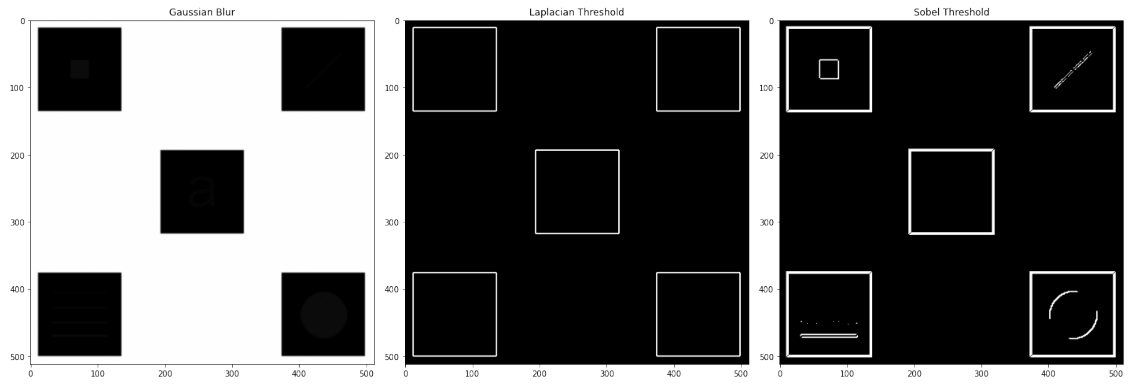


2 b. Use any other spatial filtering techniques that you think is suitable to find the hidden objects.

[3]:
```
- Load the image
- Convert the image to grayscale
- Apply a Laplacian filter to detect edges
- Apply a Sobel filter to detect edges in both directions
- Apply a Gaussian filter to smooth the image

-- Threshold the Laplacian and Sobel images to obtain binary images
```

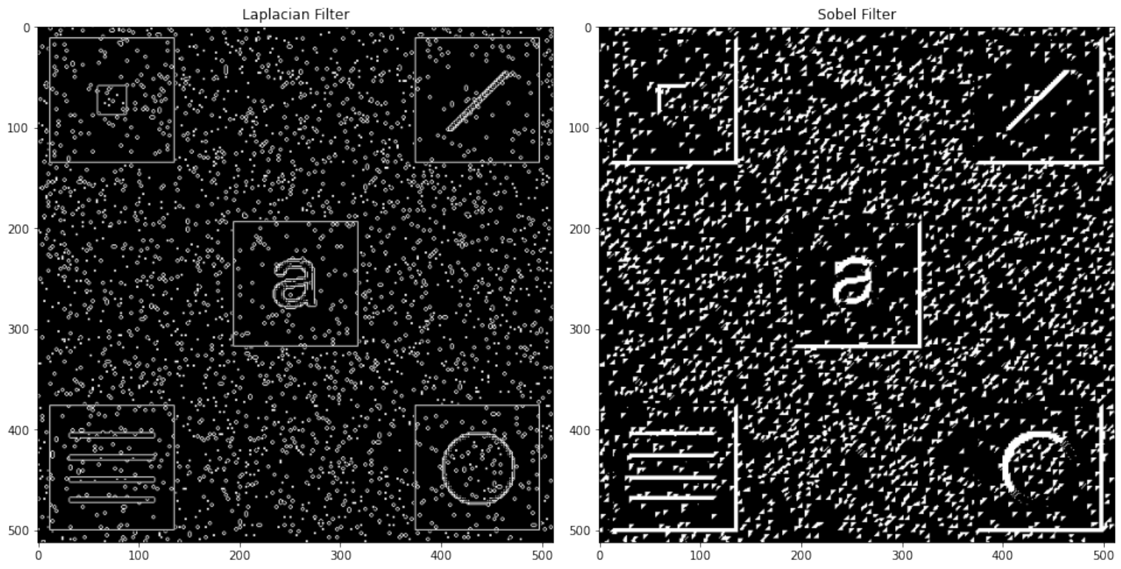
```
# Display the results
```





[7]:
```
- Apply Laplacian filter
- Apply Sobel filter

# Display the results
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
```



3 c. Write a code that implements the following method:

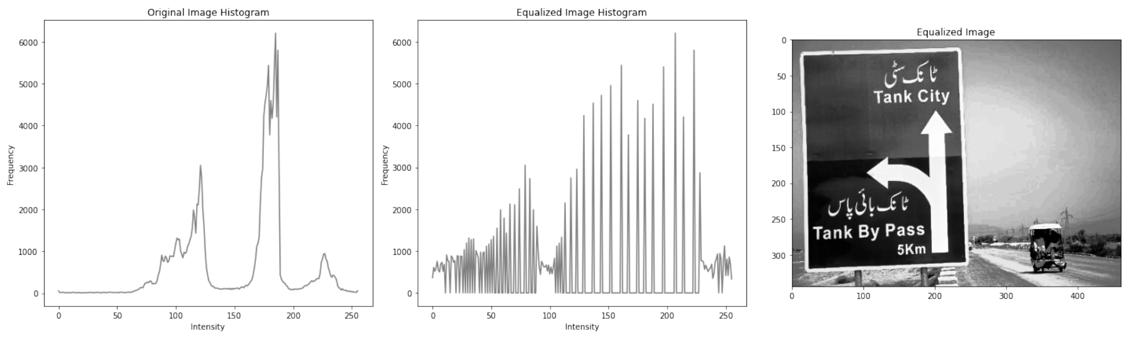
4

[8]:
```
- Compute global mean and standard deviation
- Compute local mean and standard deviation using a square neighborhood
- Compute enhanced image using the given formula
```

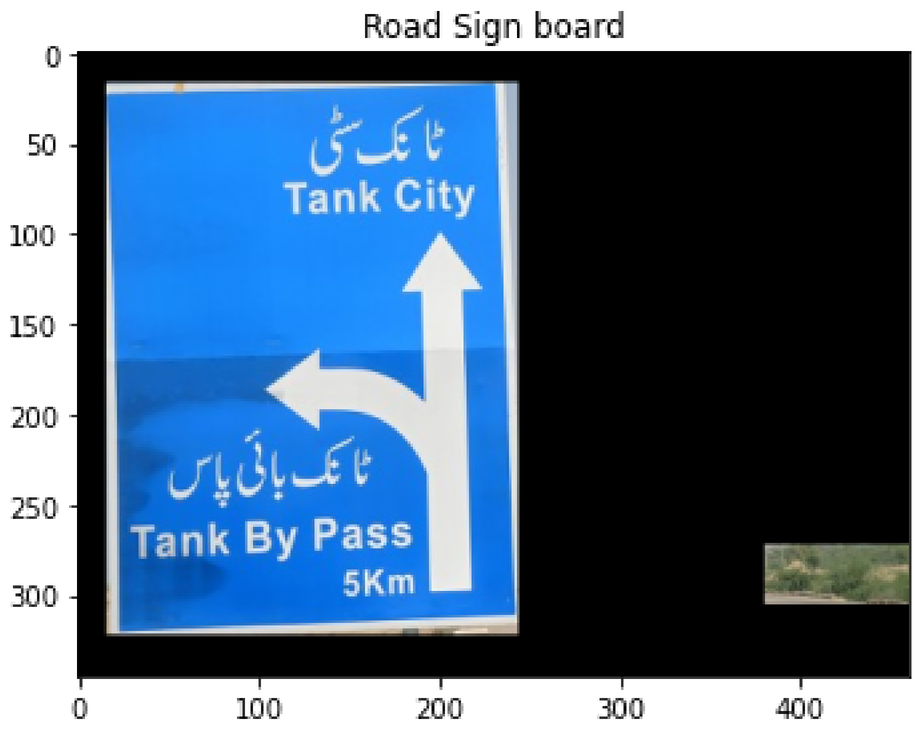
5 Question 2

6 a. Find only the blue board that has text and arrow signs. The rest of the pixels in the image should be black.

[12]:
```
- Read the image
- Calculate the histogram
- Draw the histogram

- Check if the image is too dark or too bright
- Apply histogram equalization
- Calculate the histogram of the equalized image
- Display the equalized image
```



[13]:
```
- Load the image
- Convert the image to HSV color space
- Define the range of blue color in HSV

- Create a mask for the blue board
- Perform morphological operations to remove noise

- Find contours in the image
- Loop over the contours and find the blue board that has text and arrow signs
- Create a black image of the same size as the original image
- Draw the blue board on the black image
- Add the black image to the list

- Combine all the black images using bitwise OR operation
- Mask the original image with the black image to make the rest of the pixels␣
 ↪black
```



7 b Write a program that should be able to find the background color of the sign board. It should be a generic function to identify three different primary colors: red, green or blue. It should display blue if the background color of the board is blue.

[14]:
```
- initializes a variable named "blue_count" to 0.

- It loops through each pixel in the image using two nested for loops.

- For each pixel, it unpacks the RGB values into three separate variables named␣
↪"r", "g", and "b".

- It checks whether the blue value of the pixel (b) is greater than both the␣
↪red value (r) and the green value (g). If it is, it increments the␣
↪"blue_count" variable.

- After all the pixels have been processed, it calculates the percentage of␣
↪blue pixels in the image by dividing the "blue_count" variable by the total␣
↪number of pixels in the image (width times height).
```

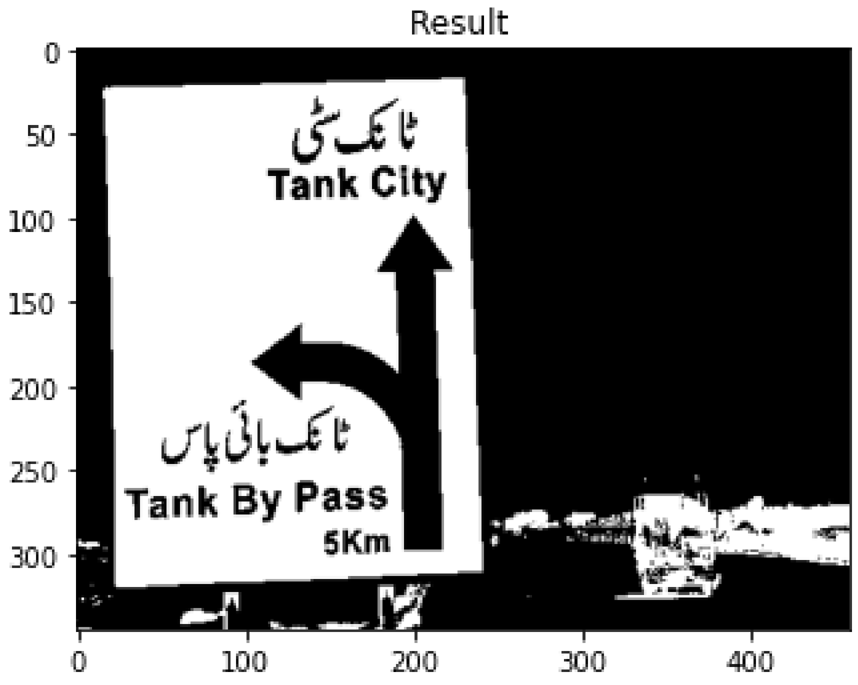
```
- If the percentage of blue pixels is greater than 50%, the function returns␣
↪the string "blue". Otherwise, it returns the string "not blue".
```

[15]:
```
- Result
```

```
blue
```

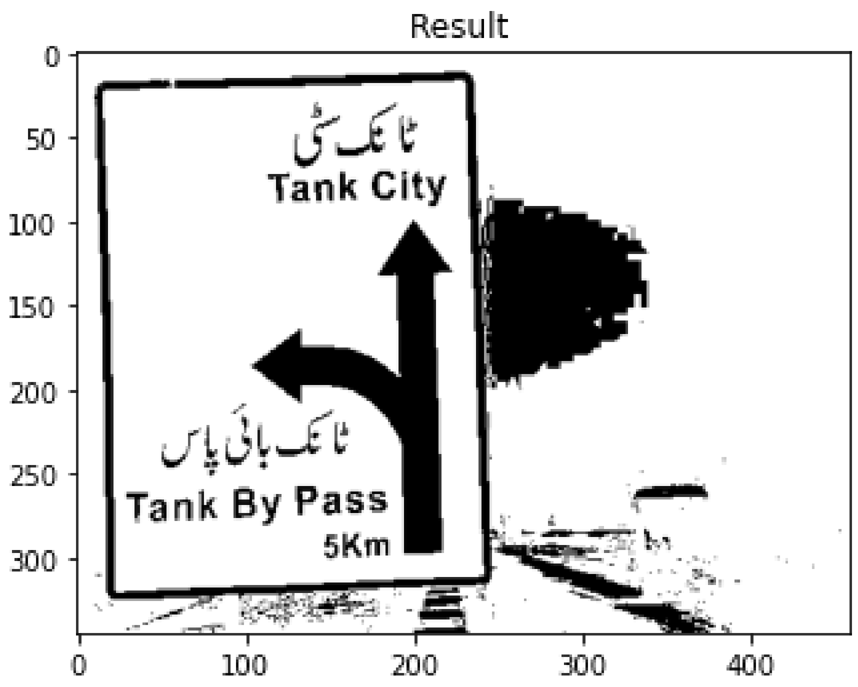
7.0.1 c. Apply thresholding to show text shown in the sign board along with arrow signs in black while the background of the sign board and rest of the image in white. You should find the threshold value automatically by calculating a histogram. The histogram ideally will have two peaks, select the value in between two peaks as the threshold value. If you use your manually entered threshold then NO MARKS will be assigned for this part.

[16]:
```

```



[17]:
```
- Read in the image
- Calculate the histogram of the image
- Find the two peaks in the histogram

- Apply thresholding to the image
- Invert the thresholded image so that text and arrow signs are black
```

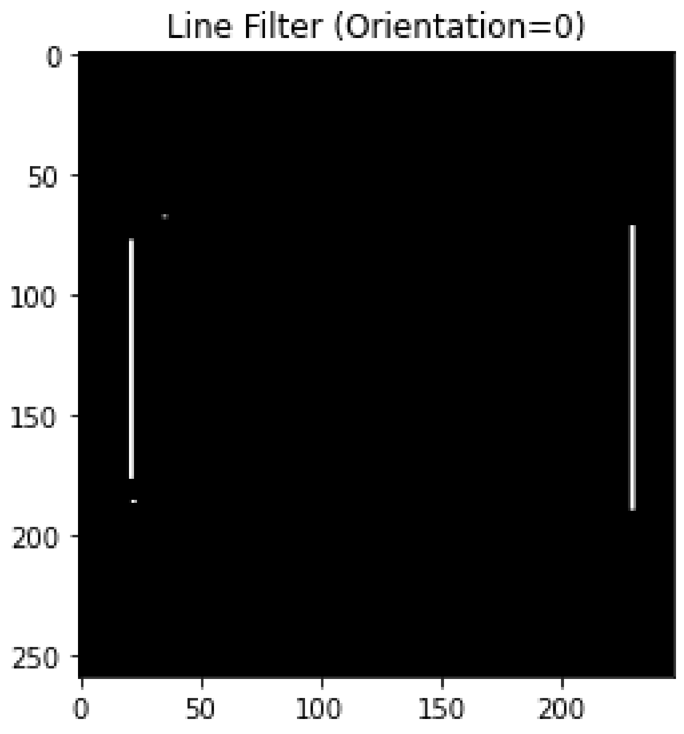
```
First peaks in the histogram 185 value 6206
second peaks in the histogram 187 value 5798
```



8 Question 3

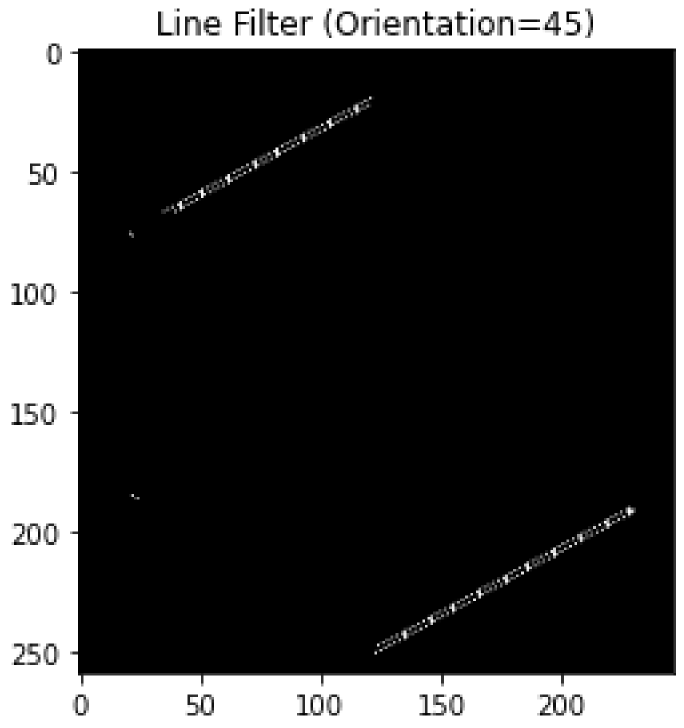
[18]:
```
- Convert image to grayscale if not already in grayscale
- Calculate gradient magnitude and direction
- Define threshold for angle
- Define kernels for different orientations

- Filter image with kernel and angle threshold

- Threshold the image to convert it to black and white
```

[19]:
```
```


Line Filter (Orientation=0)
0
50
100
150
200
250
0
50
100
150
200

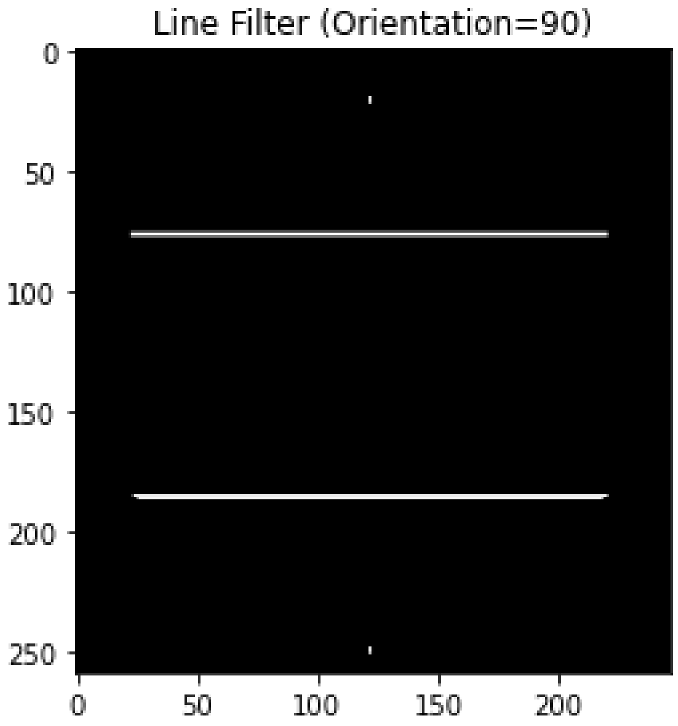
[20] :

```

```


Line Filter (Orientation=45)
0
50
100
150
200
250
0
50
100
150
200

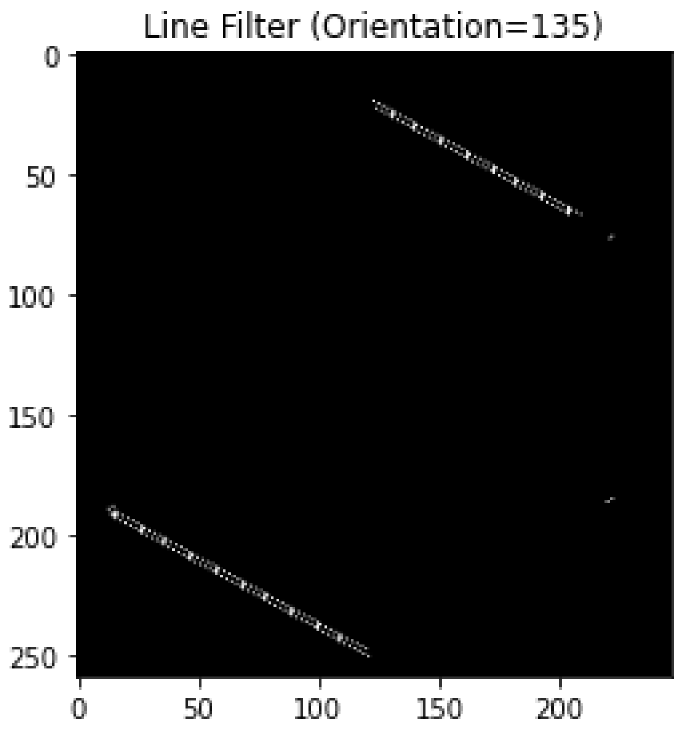
[21]:

```
```


Line Filter (Orientation=90)
0
50
100
150
200
250
0
50
100
150
200

[22]:

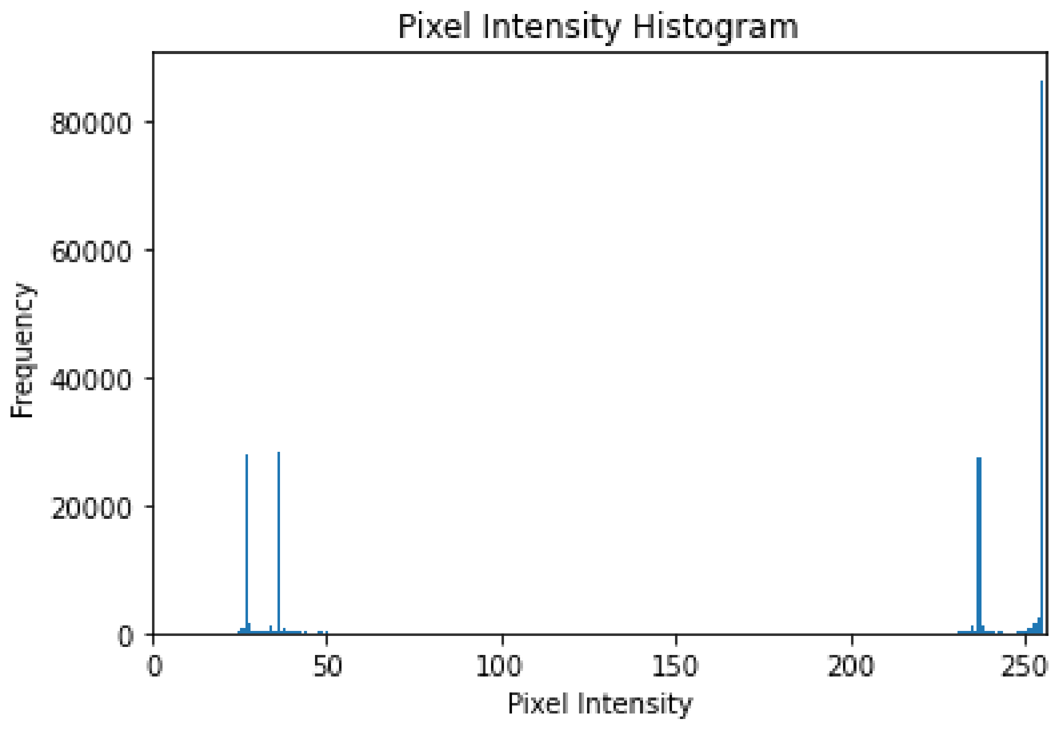
```
```



9 Question 4

10 Find percentage of the area covered by the hexagon and display it.

[23]:
```
- Create a histogram of the pixel intensities
- Plot the histogram as a bar graph

- Add labels to the plot

- Display the plot
```

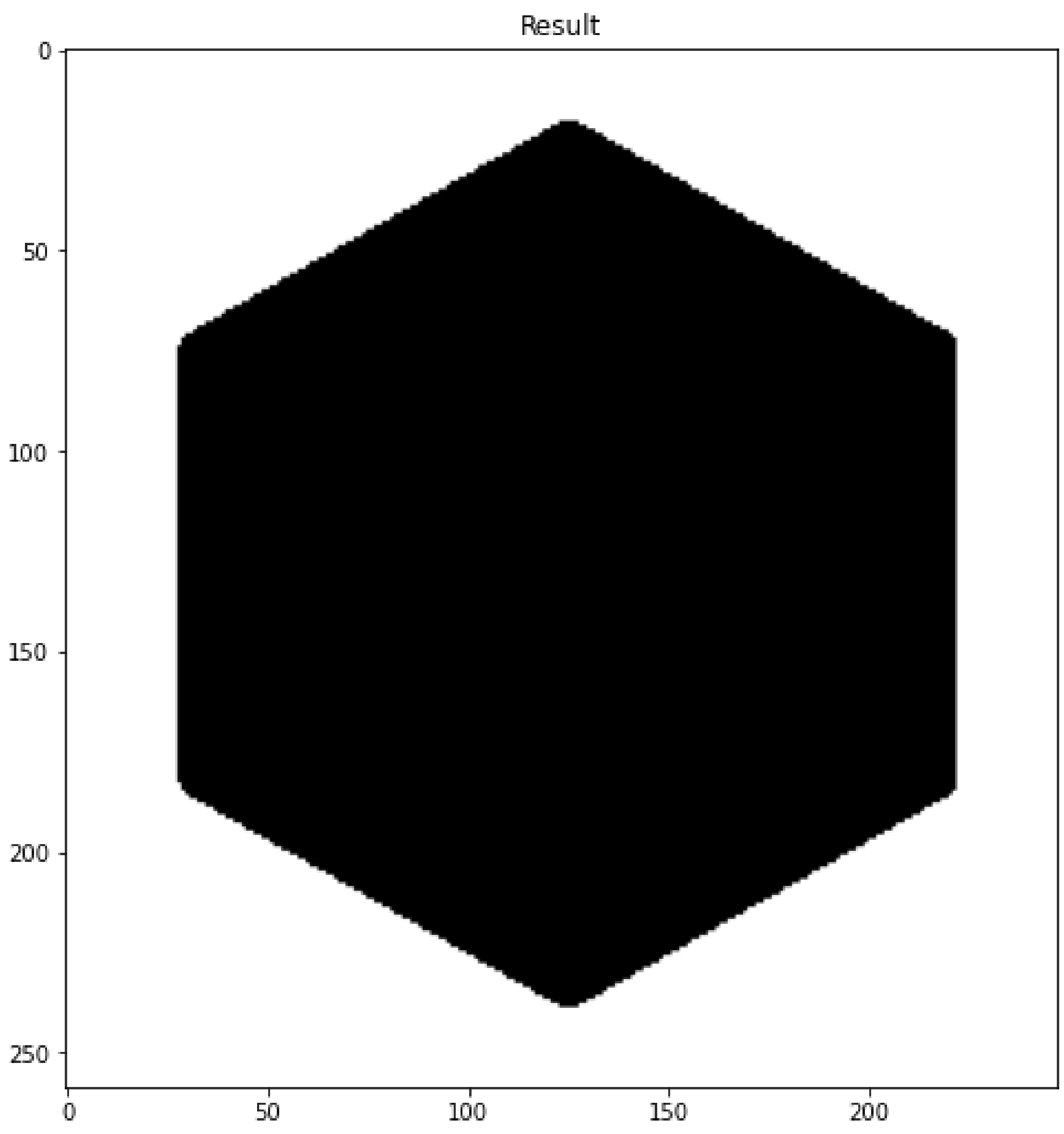


[24]:
```
- Convert the image to grayscale
- Threshold the grayscale image to create a binary image
- Find the contours of the binary image
- Find the hexagonal contour
- Approximate the contour to a polygon
- Check if the polygon has 6 sides (a hexagon)


- Draw the hexagonal contour on the original image
- Find the area of the hexagon
- Calculate the percentage of area covered by the hexagon
- Display the image and the percentage of area covered by the hexagon
```

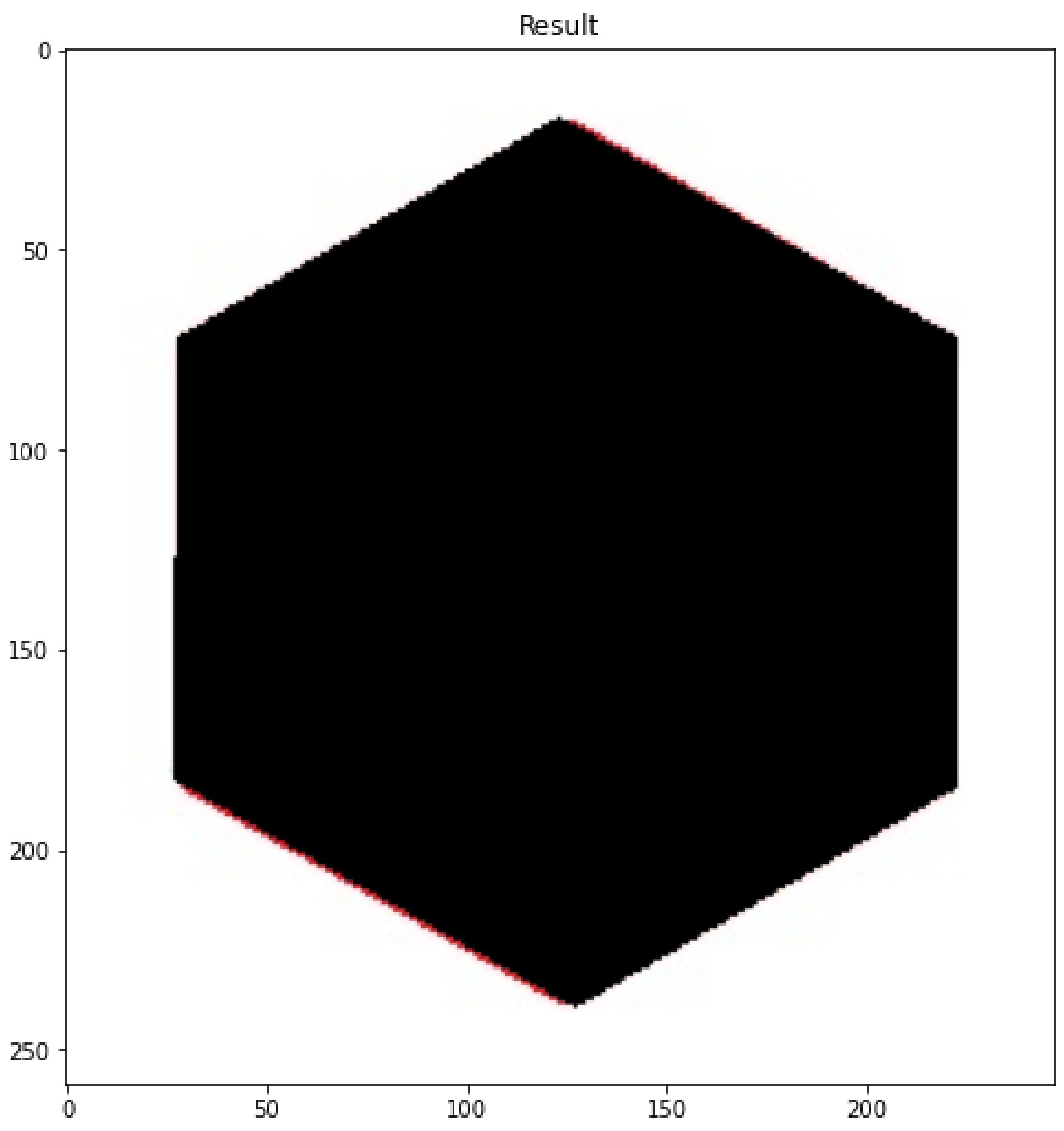
```
The hexagon covers 50.62% of the image.
```

11 b. Set all the pixels of the hexagon to ZERO and display the resulting image.

[25]:
```
- Convert the image to grayscale
- Threshold the grayscale image to create a binary image
```



[26]:
```
- Find the contours of the binary image
- Find the hexagonal contour

- Approximate the contour to a polygon
- Check if the polygon has 6 sides (a hexagon)
- Create a mask image of the same size as the original image
- Draw the hexagonal contour on the mask image
- Set all pixels of the hexagon to zero in the original image
```

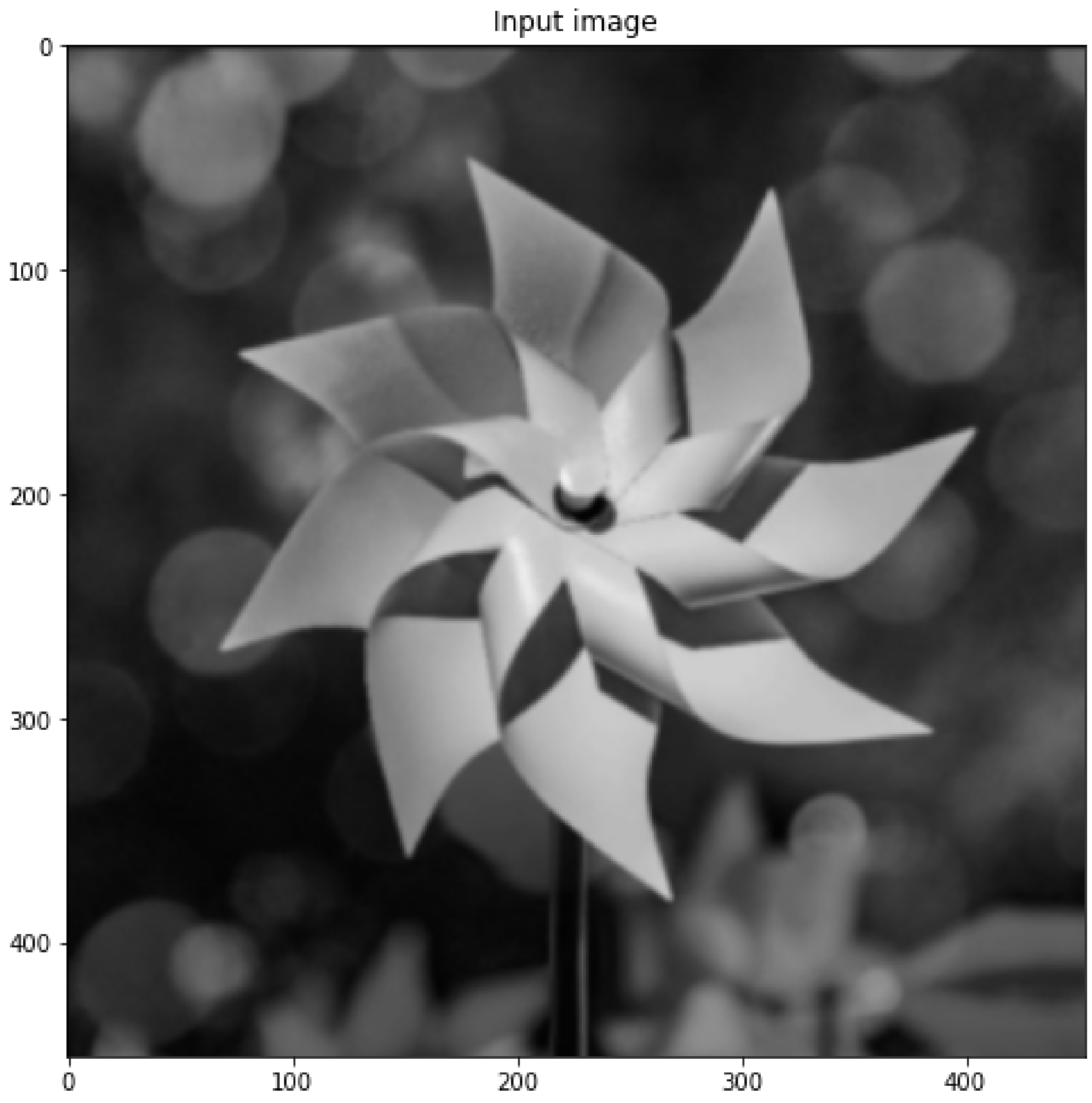

Result
0
50
100
150
200
250
0
50
100
150
200

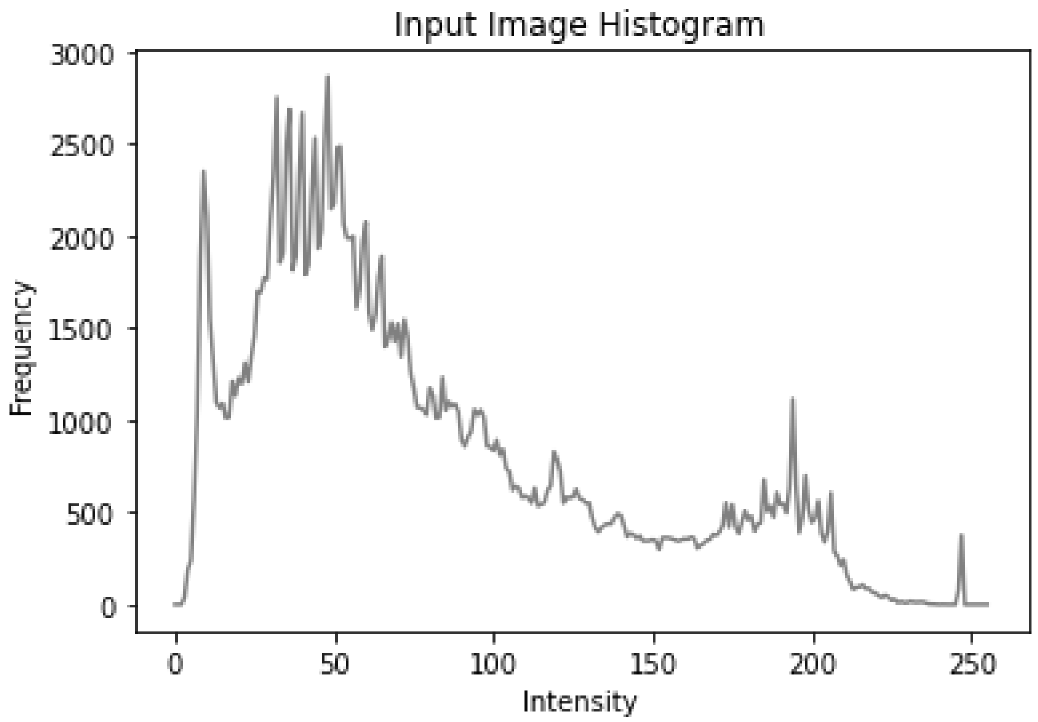
12 Question 5.

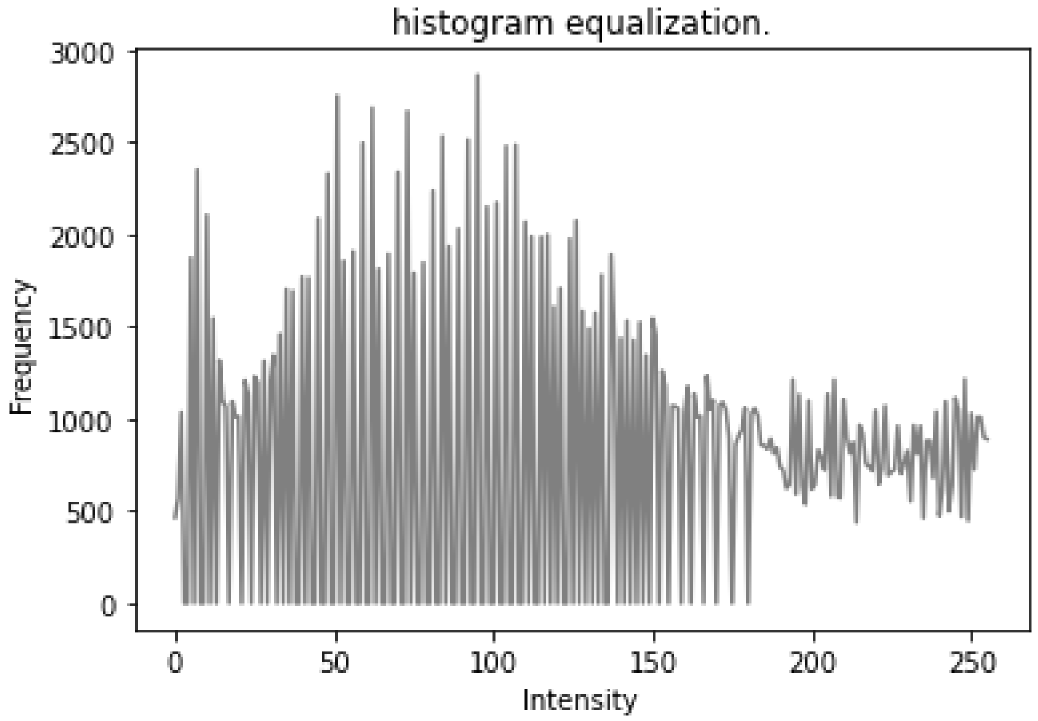
13 a. Write your own custom function that should be able to do histogram equalization. You should not use any library to perform histogram equalization.

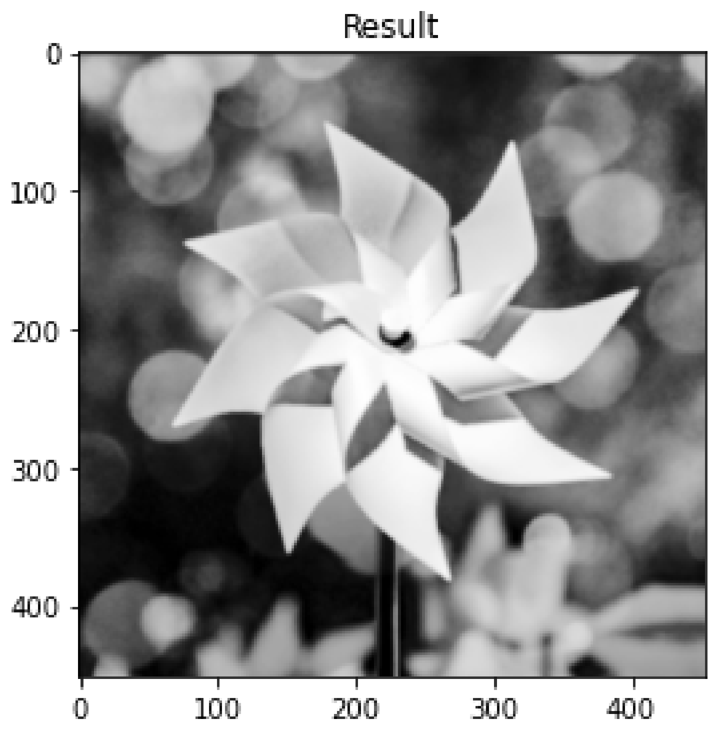
[27]:
```
- Flatten the image into a 1D array
- Calculate the histogram of the flattened image

- Draw the histogram
- Calculate the cumulative distribution function (CDF) of the histogram

- Normalize the CDF to have values between 0 and 1
    # Interpolate the CDF to get the new pixel values
    # Reshape the 1D array back into the original image shape
    # Calculate the histogram of the flattened image
```

[28]:
```
# Result
```


Input image
0
100
200
300
400
0
100
200
300
400


Input Image Histogram
Frequency
Intensity
3000
2500
2000
1500
1000
500
0
0
50
100
150
200
250


histogram equalization.
Frequency
Intensity
3000
2500
2000
1500
1000
500
0
0
50
100
150
200
250



14 b. Write a function that takes two images as input and display "Similar" if two images are similar otherwise display "Different".

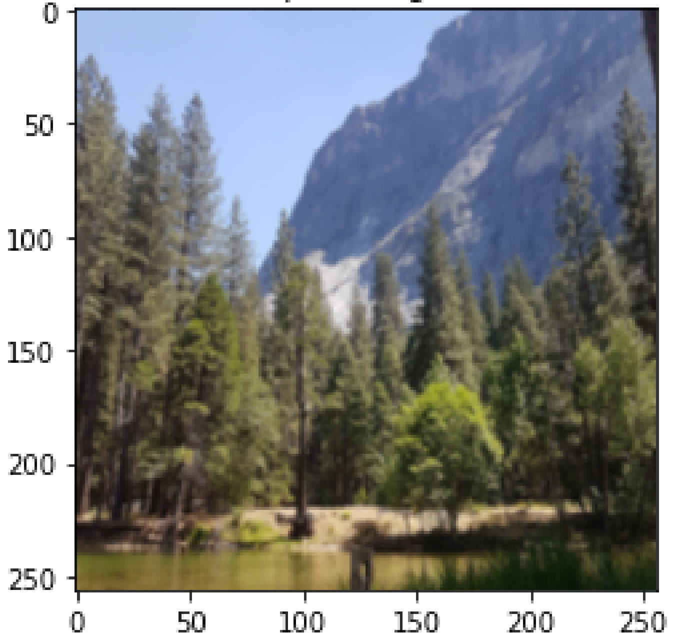
15 Note* you can use histogram to find similarity.

[29]:
```
- Convert images to grayscale
- Normalize images
    # Calculate histograms
    # Plot the histograms using Matplotlib

    # Calculate similarity score

    # Return similarity score and result
```

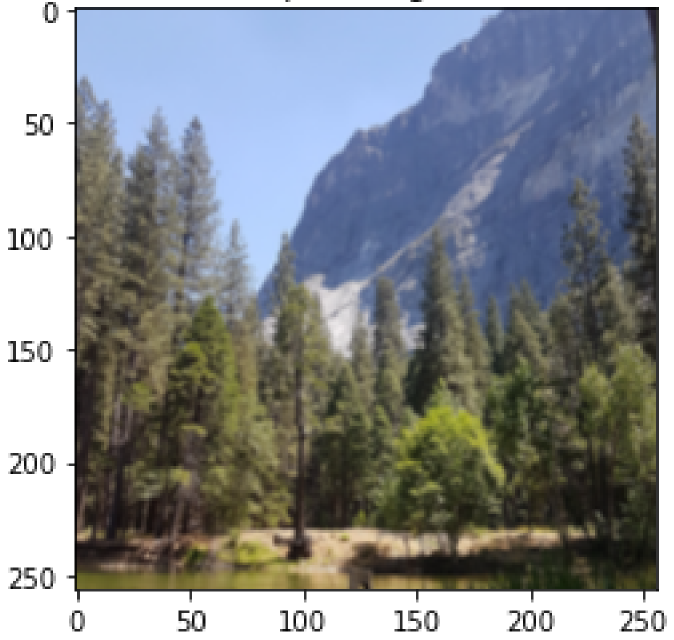
[30]:
```
# Result
```

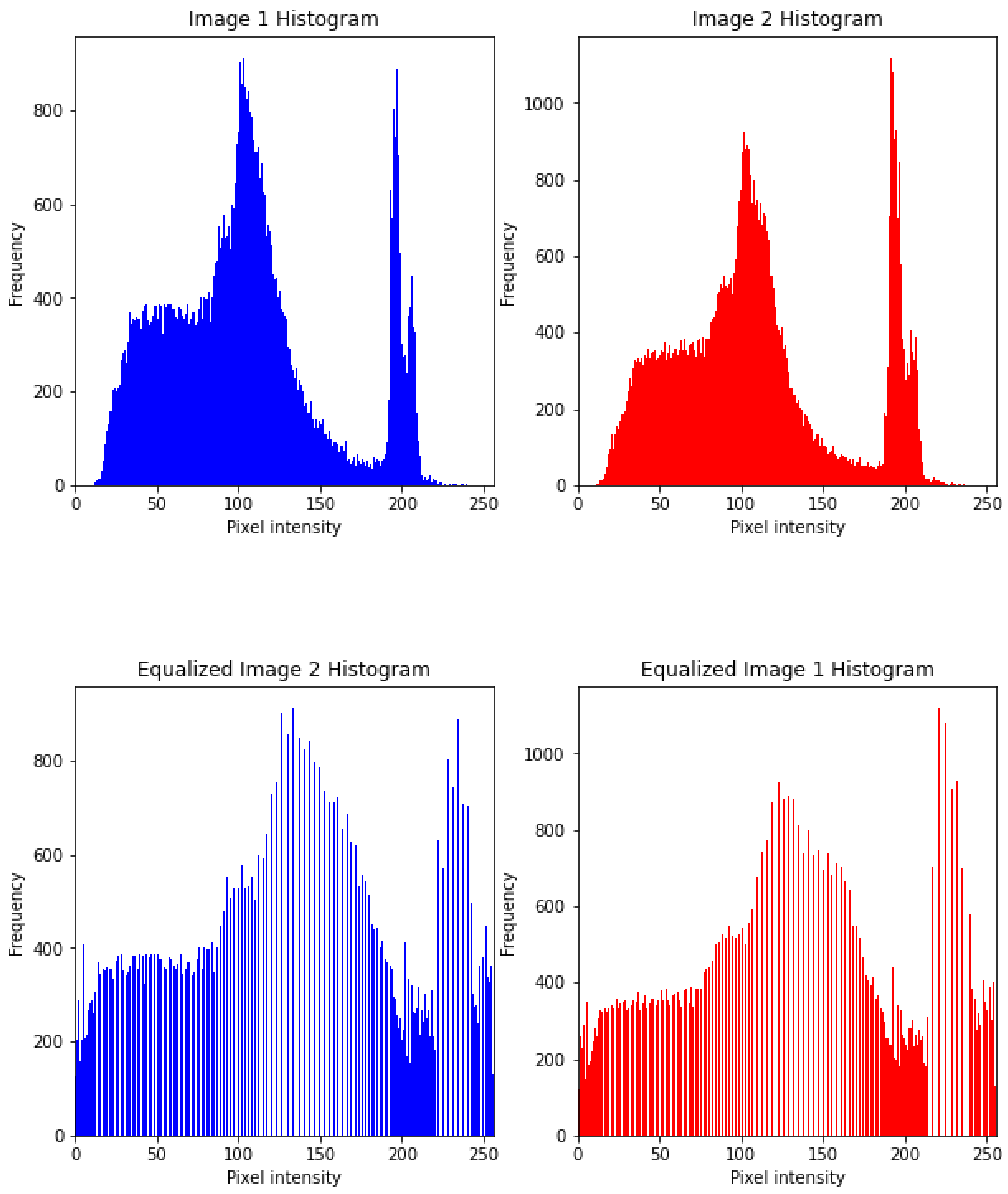
```
Image 1 shape (269, 294, 3)
Image 2 shape (270, 288, 3)
```

Input image 1



Input image 2





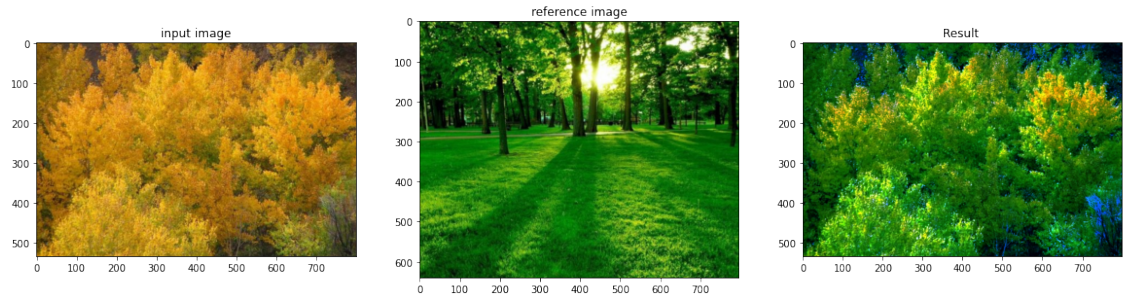
```
[30]: (0.9310303, 'Similar')
```

16 c. Write a function that enhances an image using histogram matching. Essentially, you can ask for two images as input (input image and reference image).

[1]:
```
# Get the cumulative sum of the elements



    # Normalize the cdf








    # Split the images into the different color channels
    # b means blue, g means green and r means red

    # Compute the b, g, and r histograms separately
    # The flatten() Numpy method returns a copy of the array c
    # collapsed into one dimension.


    # Compute the normalized cdf for the source and reference image


    # Make a separate lookup table for each color


    # Use the lookup function to transform the colors of the original
    # source image

    # Put the image back together
```

[3]:
```
# Display the results
```

[3]:
```
Text(0.5, 1.0, 'Result ')
```



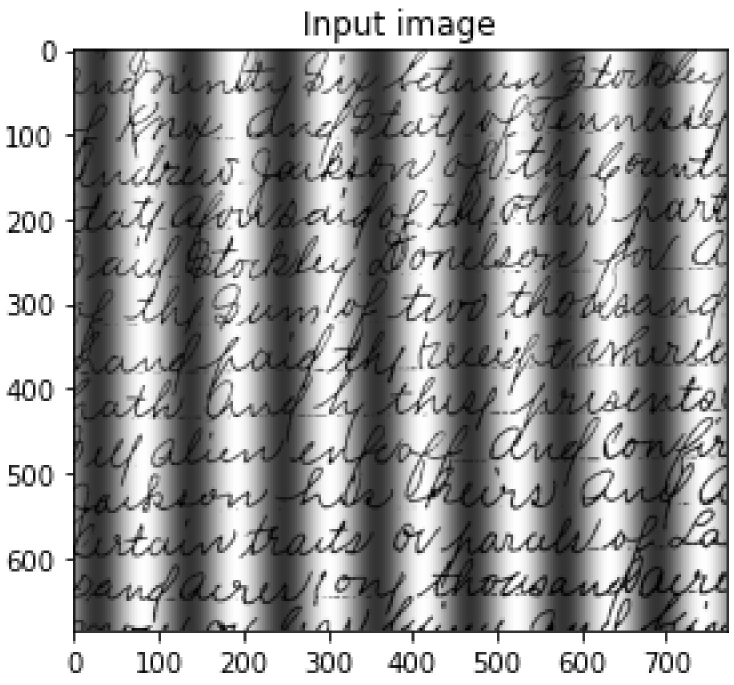
17 Question 6.

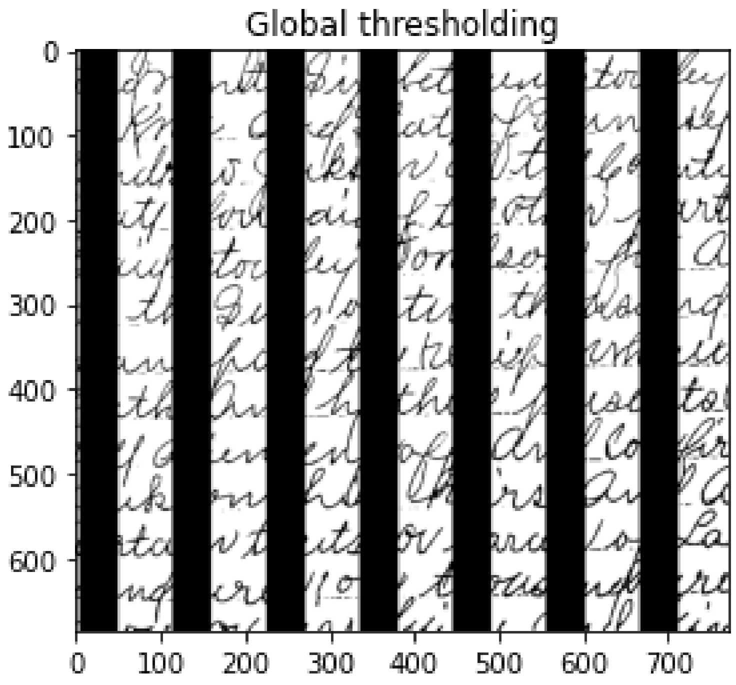
18 Global Thresholding. All the methods we discussed so far selects only one threshold for the whole image and apply it to get the final output.

19 a. Apply a global threshold on the above images and see the output. Probably it will fail. Explain your reasons for failure of the global thresholding.

20 Local Thresholding: In local thresholding we will select a small window and then apply a thresholding with respect to that small window. You can try implement your own custom function for the following technique to see if local thresholding will work.

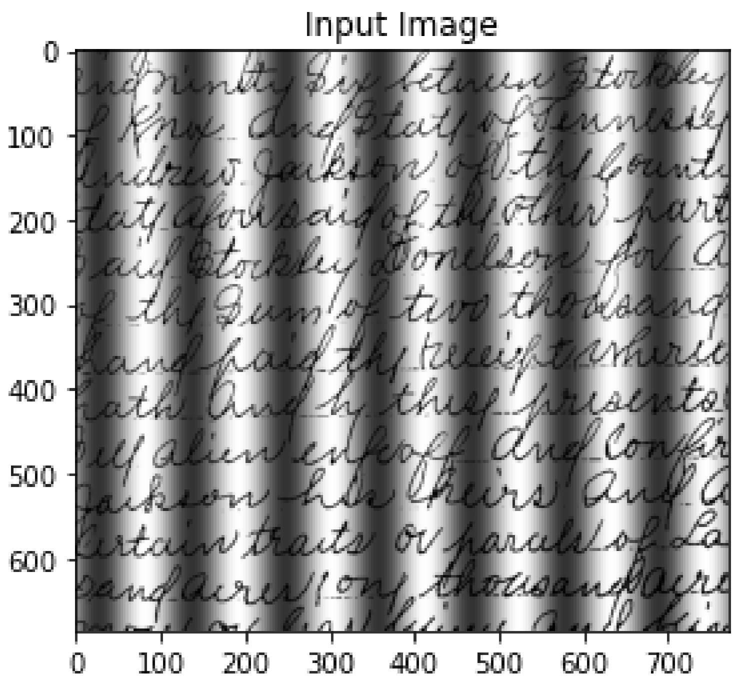
[33]:
```python
# Read the image in grayscale

# Apply global thresholding
```

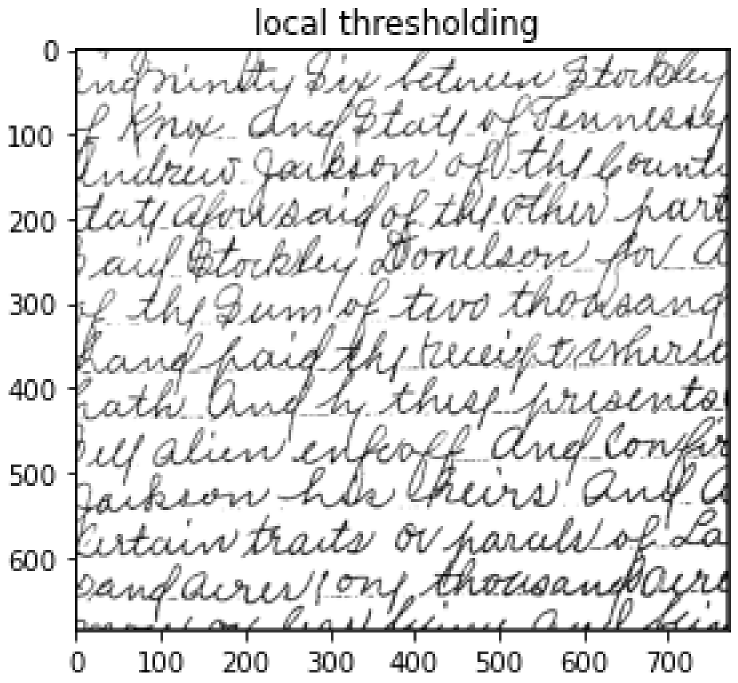




Reason for the failure of global thresholding on paper text images, it is because of the variability in the background color and lighting conditions in paper text images. This makes it difficult to choose a single threshold value that can separate the foreground (text) from the background accurately.

[34]:
```
# Read the image in grayscale



# Define the size of the window for local thresholding

# Define a custom function for local thresholding

# Apply local thresholding using the custom function
# Display the thresholded image
```





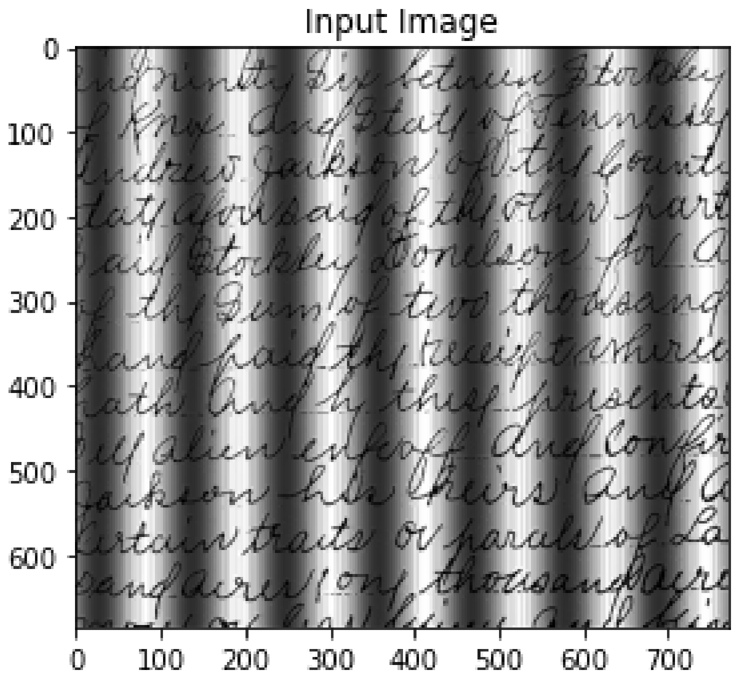
b. Consider a small neighborhood such as 3x3, 5x5 or anything that suits for you. For each location, find the average value of the pixels in the neighbourhood: AvgNHOOD.

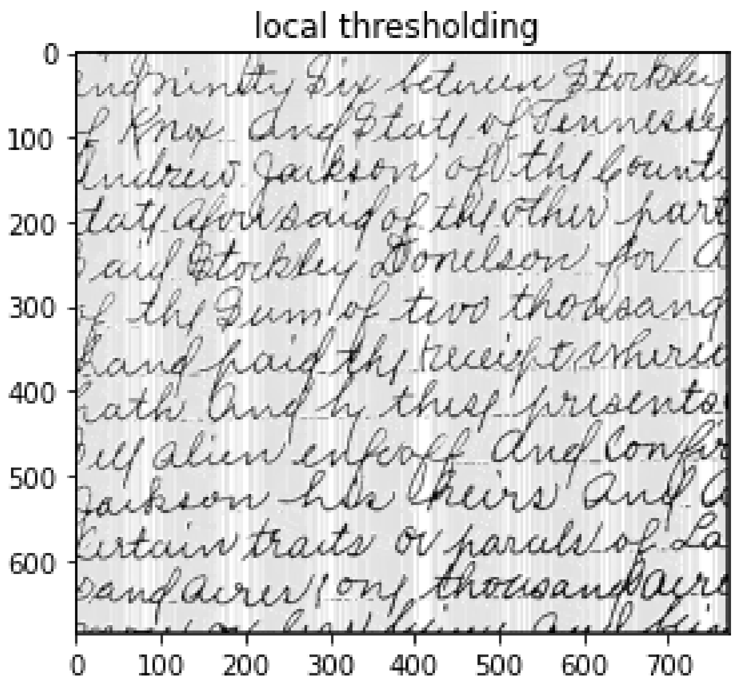
If the center pixel value is greater than AvgNHOOD then set it to ONE otherwise set it to ZERO in the output image. Then move the window to other locations and repeat for all the pixels in the image and display the output.

21 g(x, y) = { 1, if f(x, y) > AvgNHOOD

22 0, x 0

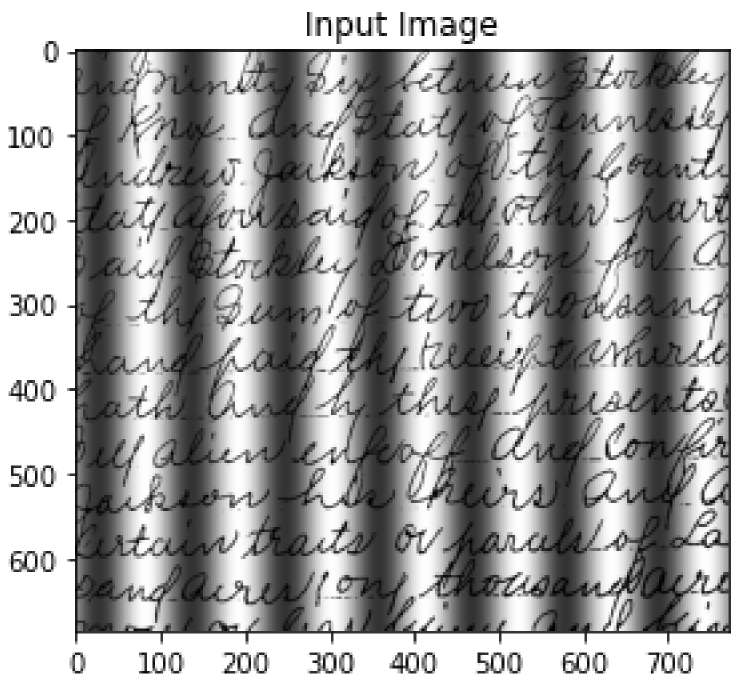
[35]:
```
# Read the image in grayscale
# Define the size of the neighborhood

# Define a custom function for local adaptive thresholding

# Apply local adaptive thresholding using the custom function
```

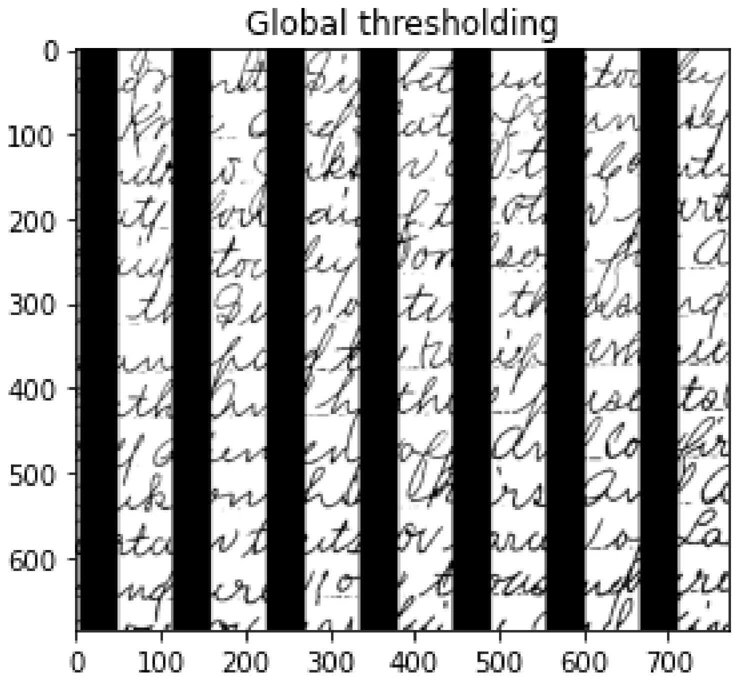


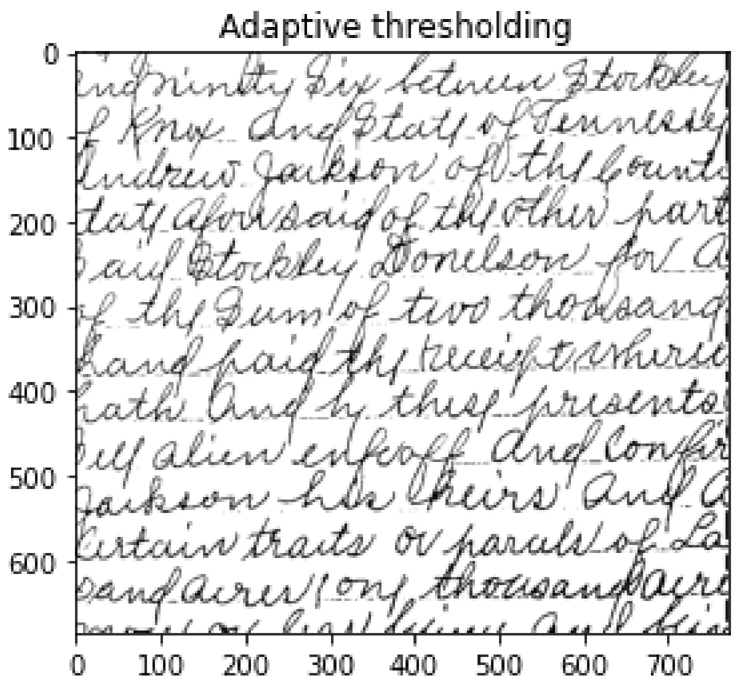


22.0.1 c. In OpenCV Adaptive Thresholding is provided. Undersand the working of Adaptive Thresholding.You can try adaptive thresholding and display your results. Compare and comment on the results obtained for your method, adaptive method and global thresholding method. Which seems more suitable for the above images.

[36]:
```
# Read the image in grayscale
# Apply global thresholding
# Apply adaptive thresholding using OpenCV
```


Input Image
0
100
200
300
400
500
600
0
100
200
300
400
500
600
700


Global thresholding
0
100
200
300
400
500
600
0
100
200
300
400
500
600
700


Adaptive thresholding
0
100
200
300
400
500
600
0
100
200
300
400
500
600
700

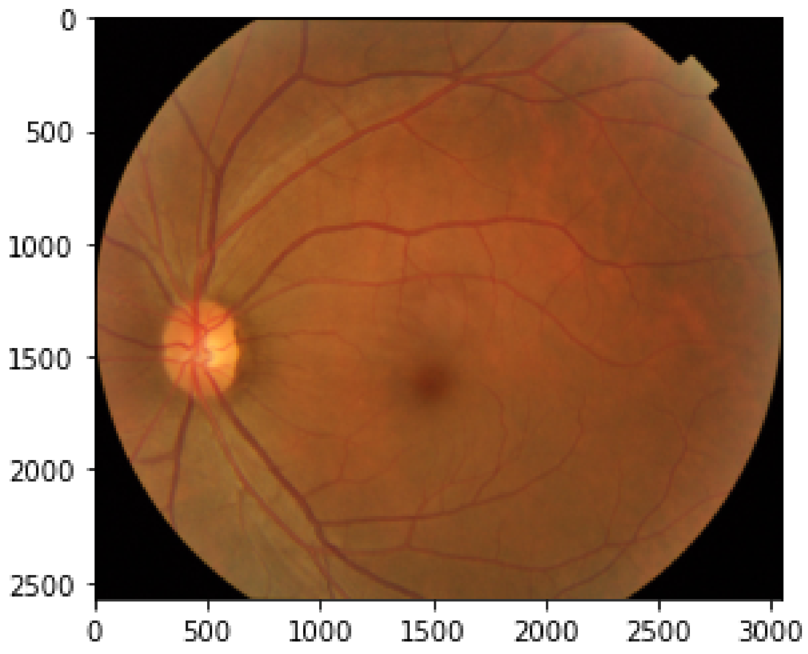
nd ninety Six between Stokley
f Knox And State of Tennessee
Andrew Jackson of the Count
tate aforesaid of the other part
aid Stokley Donelson for a
f the Sum of two thousand
hand paid the receipt whereo
hath And by these presents
ell alien enfeoff And Confir
Jackson his heirs and a
certain tracts or parcels of La
sand acres (one thousand acre

22.0.2 **Adaptive thresholding is a method of thresholding that calculates the threshold value for each pixel based on a small neighborhood around it. This method can be more effective than global thresholding because it takes into account the local variability in the image. By selecting a small neighborhood around each pixel, the adaptive thresholding method can adapt to changes in lighting and background color and produce more accurate results.**

23 Question 7.

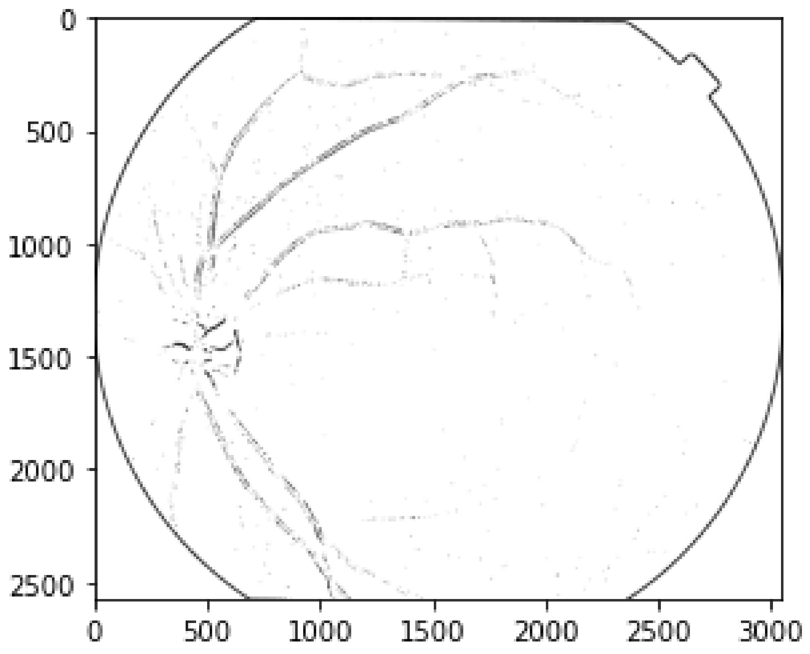
[37]:
```
# input image
```

[37]:
```
<matplotlib.image.AxesImage at 0x23048d34520>
```



[38]:
```
# Convert the image to grayscale
# Apply a Gaussian blur to the image to remove noise
# Apply adaptive thresholding to the image to separate the veins
# Apply morphological operations to the thresholded image to enhance the veins
# Display the veins in white and the rest of the pixels in black
```

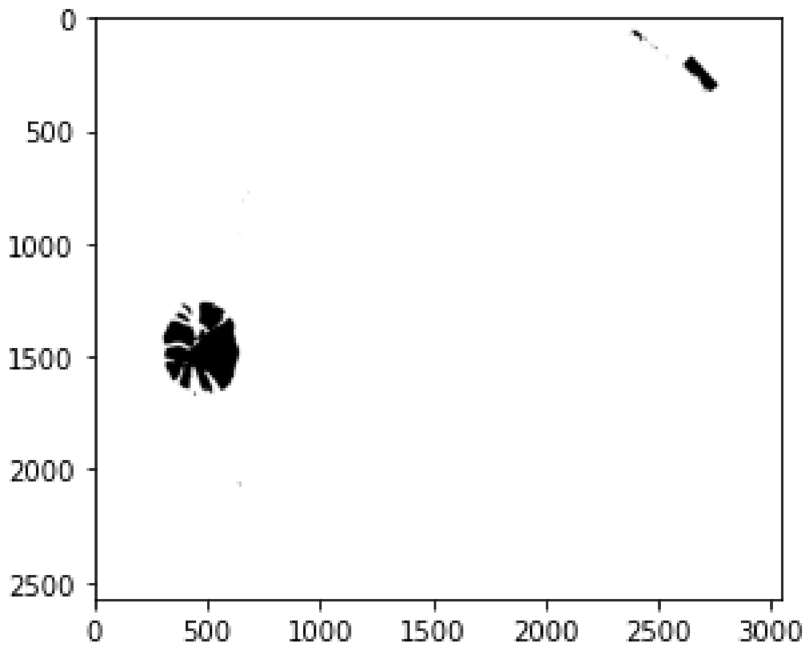
[38]:
```
<matplotlib.image.AxesImage at 0x23048ed1760>
```



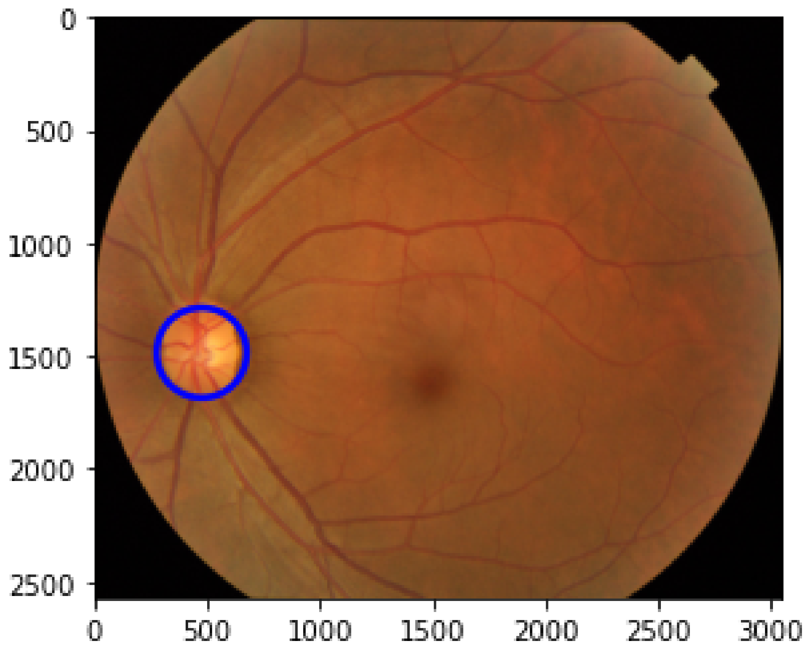
24 b. Find the diameter of the eye ball (shown with arrow) in terms of pixels and display it.

[39]:
```
# Load the fundus image as grayscale
# Convert the image to grayscale
# Apply a Gaussian blur to reduce noise
# Threshold the image to create a binary image
# Find contours in the binary image
# Sort the contours by area in descending order
# Find the pupil (the contour with the smallest area)
# Find the iris (the contour with the second-smallest area)
# Draw circles around the pupil and iris
# Draw a circle inside the iris to highlight the inside of the eye ball
# adjust the size of the circle as desired
```

```
Diameter of pixels: 398
```



0
500
1000
1500
2000
2500
0
500
1000
1500
2000
2500
3000



0
500
1000
1500
2000
2500
0
500
1000
1500
2000
2500
3000

```
[ ]:
```