Computer Vision Assignment-1 Report
M22MA003

Ques. 1. (Spot the diff) Given a stitched image containing two very similar scenes, find out the differences.

(a) Submit your implementation. (b) Write down your algorithm in brief.

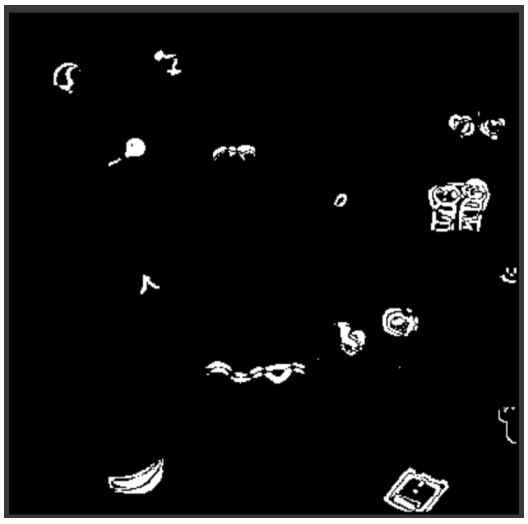
Algorithm:-

1. Slice the image into two parts.
2. Read the two sliced images.
3. Find the difference between the two slices using ImageChops.
4. Use thresholding to give a clear picture on the differences.
5. Use a kernel to dilate the differences and create a suitable object for the rectangle which will be used to mark the differences in the stitched image.

(c) Show the image where differences are suitably marked.

Given Image :

Differences in the two images:-



Dilation of the differences:-

Marking the differences with rectangles.

(d) Write down scenarios when your implementation may not work.

The implementation may not work when the given stitched image cannot be sliced into two pictures with the separating line present at the center of the stitched image.

[Q1 Colab Notebook Link]

2. (Distance in images-1) Given an image of the map of India, find out the pixel distance between two states. [Hint: use off-the-shelf OCR]

(a) Submit your implementation.

1. Imported the required libraries.
2. Used the easyocr reader model.
3. Used the readtext to fetch the text from the image.
4. Since the states' names are in capital letters, we will select only the capital letters text and put the text in the list alongwith coordinates obtained with the help of easyocr.
5. Take the average of the four coordinates of the text rectangles to obtain a center point coordinate.
6. Put all the States name and the center point coordinates in a ndarray.
7. Pick two states and find the distance between them.
8. Distance is calculated using both the methods ,i.e, Manhattan and Euclidean.

```
['KASHMIR', '172', '108'],
['HIMACHAD', '225', '137'],
['PRADESH', '226', '151'],
['PUNAB', '187', '166'],
['CHINA', '428', '163'],
['PAKISTAN', '70', '184'],
['CHANDIGARH', '184', '186'],
['(TIBET)', '429', '178'],
['UTTARAKHAND', '275', '201'],
['HARYANA', '199', '211'],
['DELH', '199', '234'],
['UTTAR PRADESH', '291', '262'],
['BHUTAN', '530', '263'],
['RAJASTHAN', '135', '290'],
['ASSAM', '592', '284'],
['NAGALAND', '655', '285'],
['BIHAR', '440', '303'],
['MEGHAL', '542', '314'],
['BANGLADESH', '530', '332'],
['IPUR', '638', '340'],
['IHARKHAND', '414', '356'],
['WEST ', '479', '356'],
['BENGAL', '475', '372'],
['MIZORAM', '612', '374'],
['GUJARAT', '84', '387'],
['MADHYA PRADESH', '250', '388'],
['MYANMAR', '655', '396'],
['(BURMA)', '655', '410'],
['ODISHA', '39
```

```
distance(arr,'BIHAR','GUJARAT')
```

```
x1,y1,x2,y2 are (440, 303, 84, 387)
Manhatten Distance between BIHAR and GUJARAT
X distance is 356 and Y distance is 84
Euclidean Distance between BIHAR and GUJARAT
365.78
```

(b) Write down the limitations of your approach.

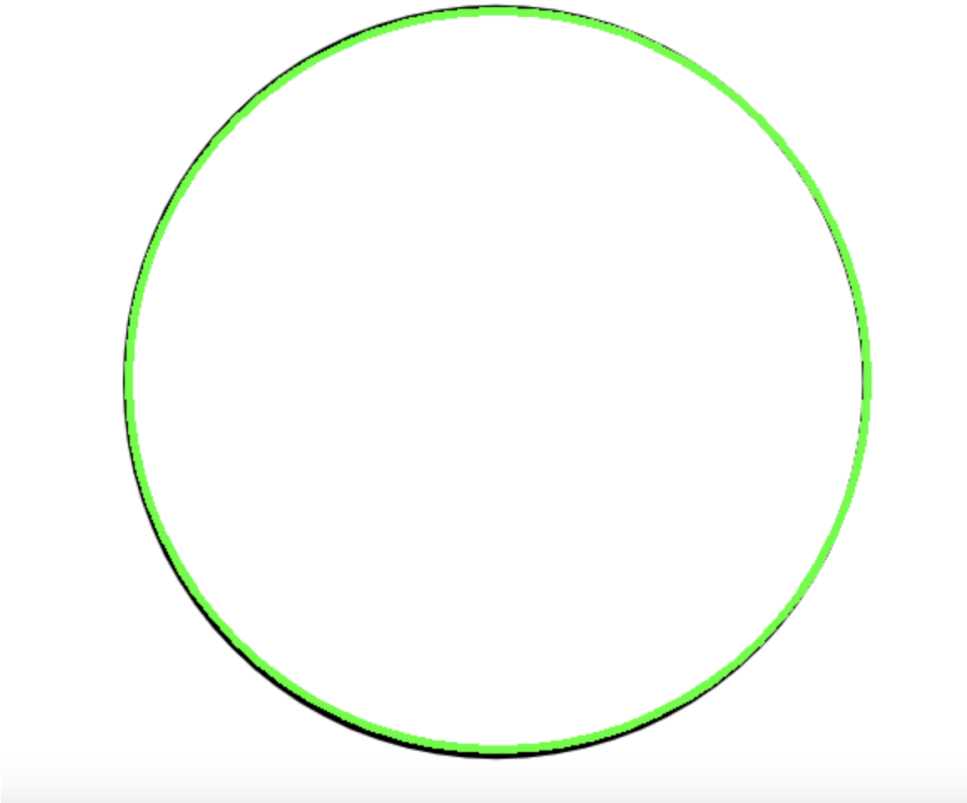
1. If there are special characters in the text, that will also be read by the easyocr reader. We need to remove the special characters text as well which could also result in removing the required text.
2. The 'easyocr' library is not able to detect the text which are overlapping with the state boundaries in the image.
3. The text is fetched from the image independent of the case or any condition. If it is a text, it will be captured regardless of the fact whether it is an actual word or not.

Q2 Colab Notebook Link

3. (Distance in images-2) Given an image of a circle, find out the area and perimeter in the pixel unit. Submit your implementation such that it takes the image file as an argument and prints the area and perimeter in new lines.

1. Imported the required libraries.
2. Used the cvtColor to convert the image into gray and used medianBlur to enhance the difference between the circle and the image.
3. Used HoughCircles transform to detect the circle in the image and then checked the threshold value to manage the correct circle count and accuracy of detection.
4. Fetch the radius from the detected circle and use the fundamental formula to find the perimeter and area of the circle.
5. The whole computation is done by defining method findValues(input) which takes input as the image.

```
Shape of the INPUT image is (526, 590, 3)
Number of circles detected are 1
Radius is 221
Perimeter of circle in pixel unit is 1388.584
Area of circle in pixel units are 153438.527
```



[Q3 Colab Notebook Link]

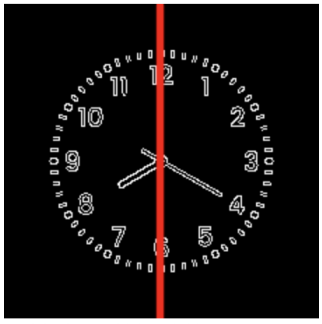
4. (Towards reading time) Given an image of a clock find out the angle between the hour and minute hands. (a) Submit your implementation.

1. Imported the required libraries.
2. Read the image into ndarray and get the values of height and width to get an idea on the size of the clock and the hands length. This size will help in keeping the threshold value for the detection of the lines in the image.
3. Used Canny edge detector to identify the edges and the HoughLines to get the lines in the image, constraining the lines attributes like minimum length and threshold value.
4. Identified the hands lines and used the HoughLines return value to get the rho and theta values.
5. Defined method findAngle(lines) to find the angle between the lines.
6. Converted the angle from radian to degrees and projected the inside angle and the outside angle between the lines.

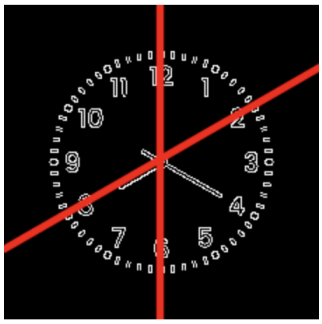
Clock Image 1

```
showLines(edges1, lines_1)
```

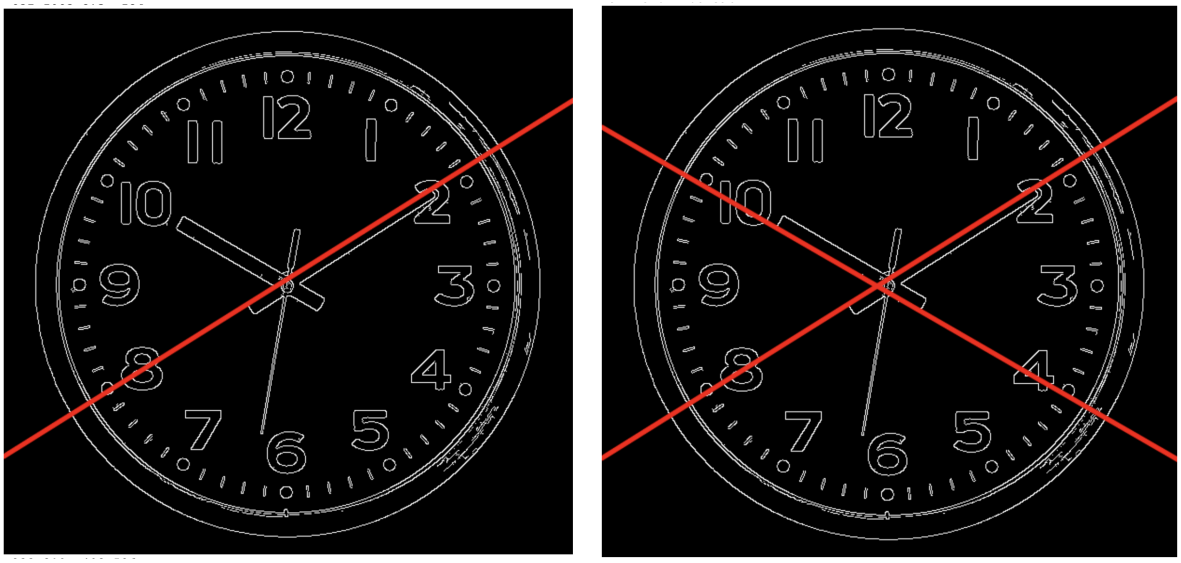
```
111 111 1000 -1000
```



```
-790 941 630 -369
```



Clock Image 2



Results:-

```
[180] findAngle(lines_1)

      Angles between minute hand and hour hand are
      in_angle is 60.0 degrees
      out_angle is 300.0 degrees

[181] findAngle(lines_2)

      Angles between minute hand and hour hand are
      in_angle is 62.0 degrees
      out_angle is 298.0 degrees
```

(b) Write down your approach for finding out the angle.

Using basic fundamentals of the geometry, it was found that the angle between the hands would be the difference between the angles obtained from the HoughLines (these angles are actually the angle between the perpendicular from origin to line and the x axis).

(c) Write down the limitations of your approach.

1. The threshold values needs to be customized for different images, thus the process is not so dynamic.
2. Even after putting thresholding values, there were multiple lines obtained from the HoughLines reason being that there could be parallel lines in the hands of the clock. So, there was need to select just two lines from the set.

Q4 Colab Notebook Link

5. (Fun with Landmarks) Choose three images of a world landmark from the Google Landmark dataset (Link: https://storage.googleapis.com/gld-v2/web/index.html). The name of your chosen landmark should begin with the first letter of your first name. For example: If your name is Adhrit, you could choose Amarnath.

Landmarks Chosen : Beaver Lake, Beth Shalom Temple, Bodhi Tree.

Beaver Lake, Montreal :-

Beth Shalom Temple, Cuba :-

Bodhi Tree :-

(a0) Resize all images to 256 × 256. Convert it to gray.

Results:-

(a) Show the average of all three images.

Average of the image is taken using the fundamental formula (image1+image2+image3)/3.
Result:-

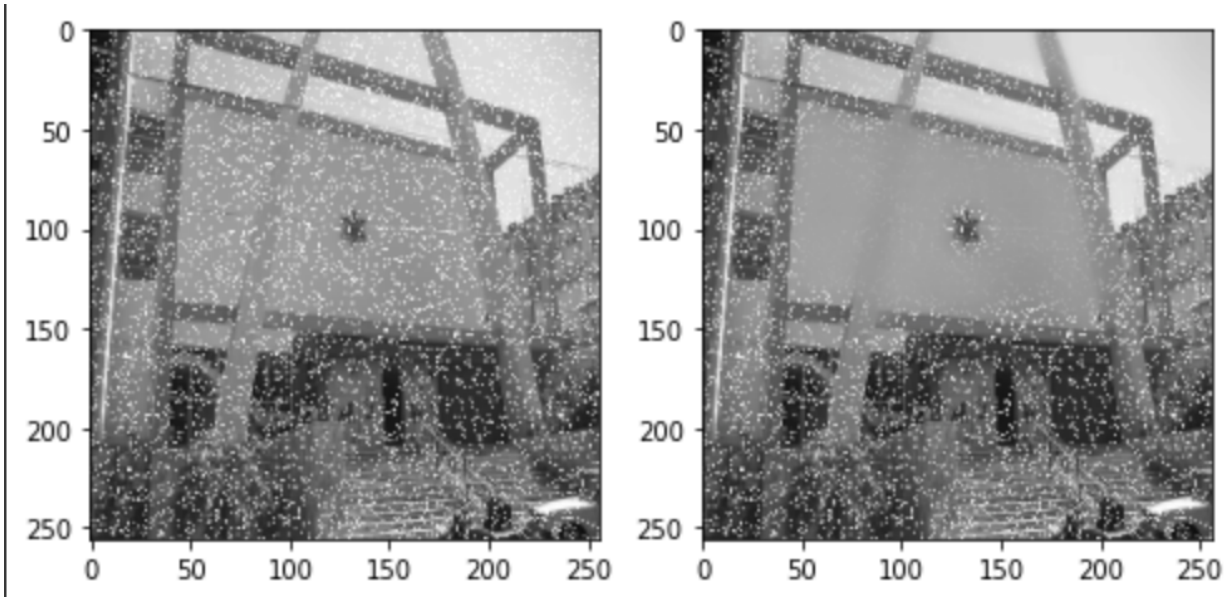
(c) Subtract Image 1 with Image 2.

OpenCV subtract method is used to subtract Image1 with Image 2.

(d) Add salt noise with 5% probability in one of the images.

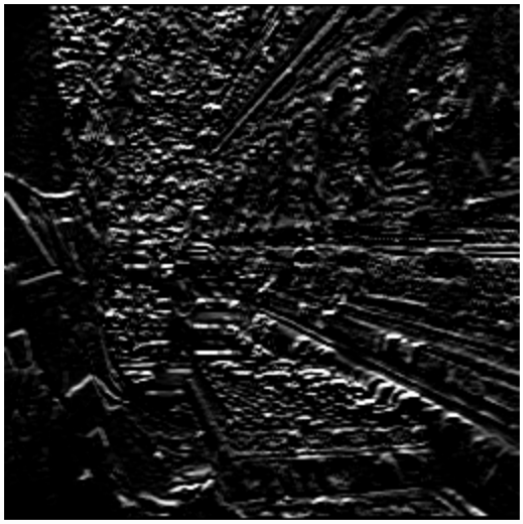
Salt noise can be applied once the image is turned to grayscale. So first, I converted the image to grayscale. Find the dimension of the image and then chosen 5% probability to add the noise to the image by randomly picking pixels.

(e) Re-move the noise.
When the noise is gaussian, we use fastNlMeansDenoisingColored Denoising function.



(f) Use the following 3×3 kernel: {−1, −1, −1; 0, 0, 0; 1, 1, 1} for performing convolution in one of the images and show the output.

```
kernel_val = np.array([[-1, -1, -1],
                       [0, 0, 0],
                       [1, 1, 1]])
```



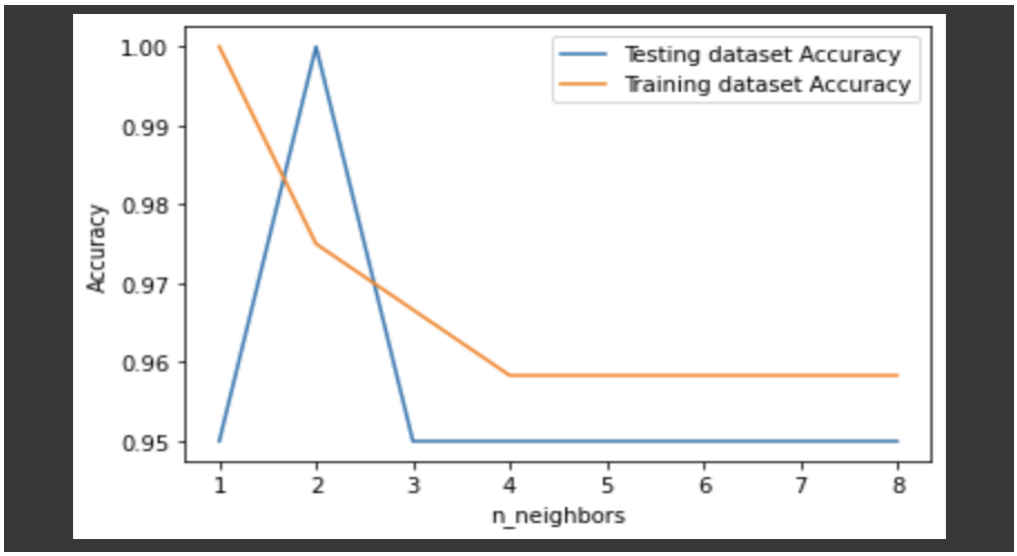
[Q5 Colab Notebook Link]

6. (Digit Recognition) You will be given 100 handwritten images of 0 and 1. You have to compute horizontal projection profile features and use Nearest Neighbour and SVM classifiers to recognize the digits. Report accuracy and show some visual examples. Dataset (choose only 0 and 1): https://github.com/myleott/mnist_png.git

Used 2000 images to train the model and then 100 images to test the model.

1. Imported the required libraries.
2. Mounted the Google Drive and converted the images pixel range from 0-255 to 0-1.
3. Applied horizontal projection and looped over the dataset.
4. Marked the data points with labels for the train dataset.
5. Used sklearn library to import teh KNN and SVM models.
6. Fitted the model on the train dataset. In KNN, checked a few values for the neighbour k value to obtain good accuracy. k=3 resulted out the perfect number.
7. Evaluated the accuracy from both KNN and SVM and below are the results.

Screenshot of the KNN accuracy with varying neighbor value



Accuracy results are as follows:-

```
print(f"KNN Accuracy with 3 neighbours is {knn.score(x_test, y_test)*100}%")
```

```
KNN Accuracy with 3 neighbours is 95.0%
```

```
print(f"KNN Accuracy with 2 neighbours is {knn.score(x_test, y_test)*100}%")
```

```
KNN Accuracy with 2 neighbours is 100.0%
```

```
[90] print(f"SVM Accuracy is {clf.score(x_test,y_test)*100}%")
```

```
SVM Accuracy is 92.5%
```

Example from each model prediction:-

```
img=cv2.imread("/content/gdrive/MyDrive/HorizontalProjection/testing/0/10.png")
cv2_imshow(img)
y=knn.predict([x_test[0]])
print(f"predicted value of the image is {y}")
```

```
predicted value of the image is [0]
```

```
[78] img=cv2.imread("/content/gdrive/MyDrive/HorizontalProjection/testing/1/107.png"
     cv2_imshow(img)
     y=clf.predict([x_test[20]])
     print(f"predicted value of the image is {y}")
```

```
predicted value of the image is [1]
```

[Q6 Colab Notebook Link]

7. (White on Black or Black on White): Given a word image find out if the word is bright text on a dark background or dark text on bright background.

Idea used - Parse in the center row of the image. If the gradient value increases then the text is light and the background color is dark. If the gradient value decreases then the text is dark and background color is light.

1. Imported the required libraries.
2. Read the text using the easyocr reader model.
3. The image is converted to Binary or Binary Inverted to enhance the gradient value difference.
4. Later the gradient is calculated using Sobel Filter.
5. Text will be present the center of the image, so fetched the center row pixel by the dividing the height into half and checked the gradient values to identify the background contour.

Dark backgorund

Light backgorund

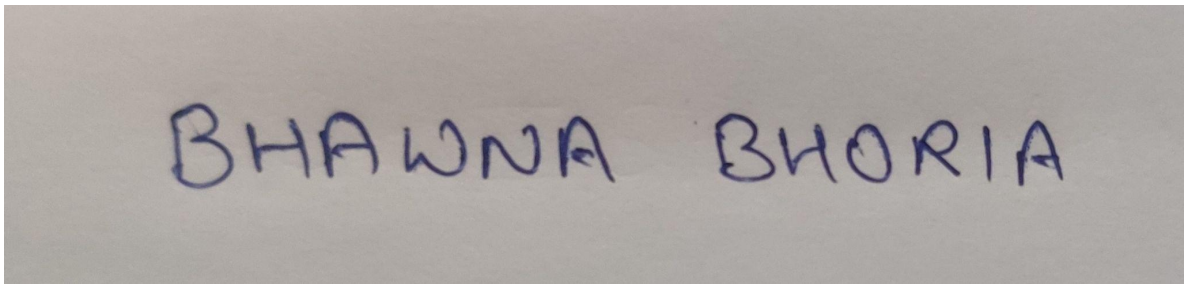
[Q7 Colab Notebook Link]

8. (Template Matching) Write your name in capital letters on a piece of white paper and a random letter from your name. Click photographs of these. Implement the Template Matching algorithm and discuss your observation.

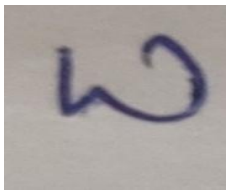
1. Imported the required libraries.
2. Read the image and the template images.
3. Used easyocr reader model to identify the text from the image.

4. Converted the template and image to grayscale. Made sure that the template size is smaller than the image.
5. Template is matched with the image to find any matches and the threshold value is set to 0.9
6. Rectangle is created over the region where the match is found.

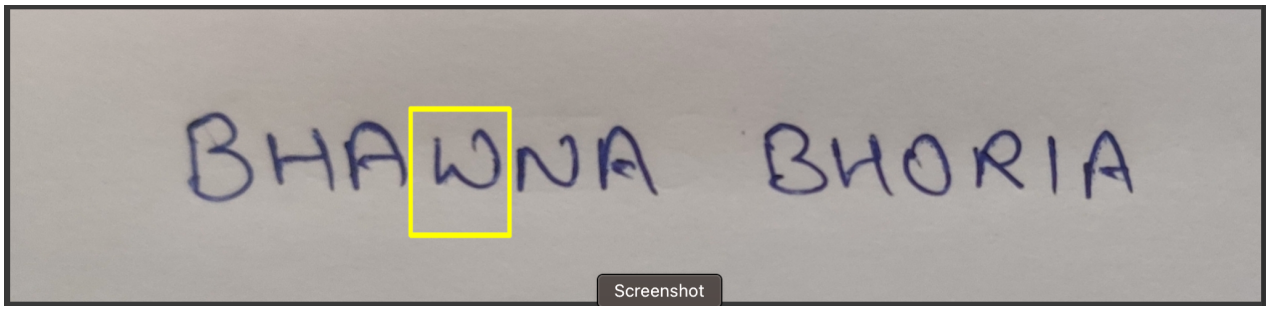
Name Image:-



Template Image:-



Results:-



[Q8 Colab Notebook Link]

9. (Histogram Equalization) Choose one image from Problem 5. Show histogram of pixel values with bin size 10. Perform histogram equalization and show the output image.

1. Import the required libraries.
2. Read the image and use the histogram function of numpy.
3. Range of histogram is 0-255 and number of bins are kept 10.
4. Using pyplot, plotting the figure of the histogram.
5. Histogram equalisation is done using the cv2 library.
6. Horizontally stacking the original image and histo equalized image to show the difference.

Chosen Image:-
Beaver Lake, Montreal

Histogram:-

```
BinEdges are [  0.   25.5  51.   76.5 102.  127.5 153.  178.5 204.  229.5 255. ]
```

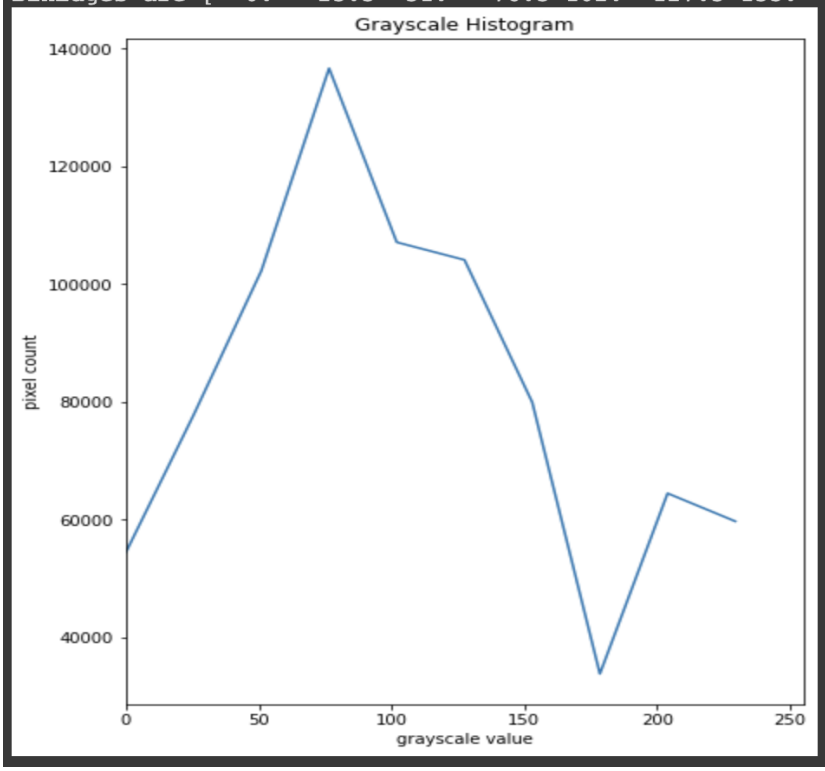


Image before and after histogram equalization :-

[Q9 Colab Notebook Link]

10. (Reading Mobile Number) You will be given image of a mobile number.
Use off-the-shelf OCR and find out the last three digits of the mobile number.

1. Imported the required libraries.
2. Used easyocr reader model to identify the mobile number in the image.
3. Selected the required substring (last 3 digits) from the obtained number.

Image1

Image2

Q10 Colab Notebook Link