



**National University of Sciences and Technology (NUST)**  
**School of Electrical Engineering and Computer Science**

**Department of Computing**

**EE 433: Digital Image Processing**

**Name: Amal Saqib**

**CMS: 282496**

**Class: BSCS 9C**

**Lab 3: Basic Image Processing**

**Date: 27<sup>th</sup> September 2021**

**Time: 2.00Pm to 5.00Pm**

**Instructor: Dr. Imran Malik**



## Basic Image Processing

### Task #1: Image Binarization using a predefined global threshold.

Take an RGB image (preferably from the provided ones) and convert it to binarized form (in 0/1 form) by defining a single global threshold. Repeat the experiment with the three provided images and identify why a single global binarization threshold may not be applicable in a wide variety of application scenarios.

Note :\_ you have to perform task 1 on following images B1,B2,B3

### Refer output of B3.jpg

Not all images can be binarized using the same threshold

### CODE

```
import os, sys
from PIL import Image

# decide the threshold
threshold = 164

# binarize image
def binarization(img):
    # load the pixels of the image
    pix = img.load()
    # get width and height of the input image
    width, height = img.size
    # iterate through all the pixels
    for x in range(width):
        for y in range(height):
            if pix[x, y] > threshold:
                pix[x, y] = 255
            else:
                pix[x, y] = 0
    return img

# gets all the files from the command line arguments
for infile in sys.argv[1:]:
    try:
        f, e = os.path.splitext(infile)
        # open image
        img = Image.open(infile).convert('L')
```



```
binarized_img = binarization(img)

# save binarized image
binarized_img.save(f + '_binarized' + e)
except IOError:
    print("Binarization Error")
```

```
1  import os, sys
2  from PIL import Image
3
4  # decide the threshold
5  threshold = 164
6
7  # binarize image
8  def binarization(img):
9      # load the pixels of the image
10     pix = img.load()
11     # get width and height of the input image
12     width, height = img.size
13     # iterate through all the pixels
14     for x in range(width):
15         for y in range(height):
16             if pix[x, y] > threshold:
17                 pix[x, y] = 255
18             else:
19                 pix[x, y] = 0
20     return img
21
22 # gets all the files from the command line arguments
23 for infile in sys.argv[1:]:
24     try:
25         f, e = os.path.splitext(infile)
26         # open image
27         img = Image.open(infile).convert('L')
28
29         binarized_img = binarization(img)
30
31         # save binarized image
32         binarized_img.save(f + '_binarized' + e)
33     except IOError:
34         print("Binarization Error")
```



# National University of Sciences and Technology (NUST) School of Electrical Engineering and Computer Science

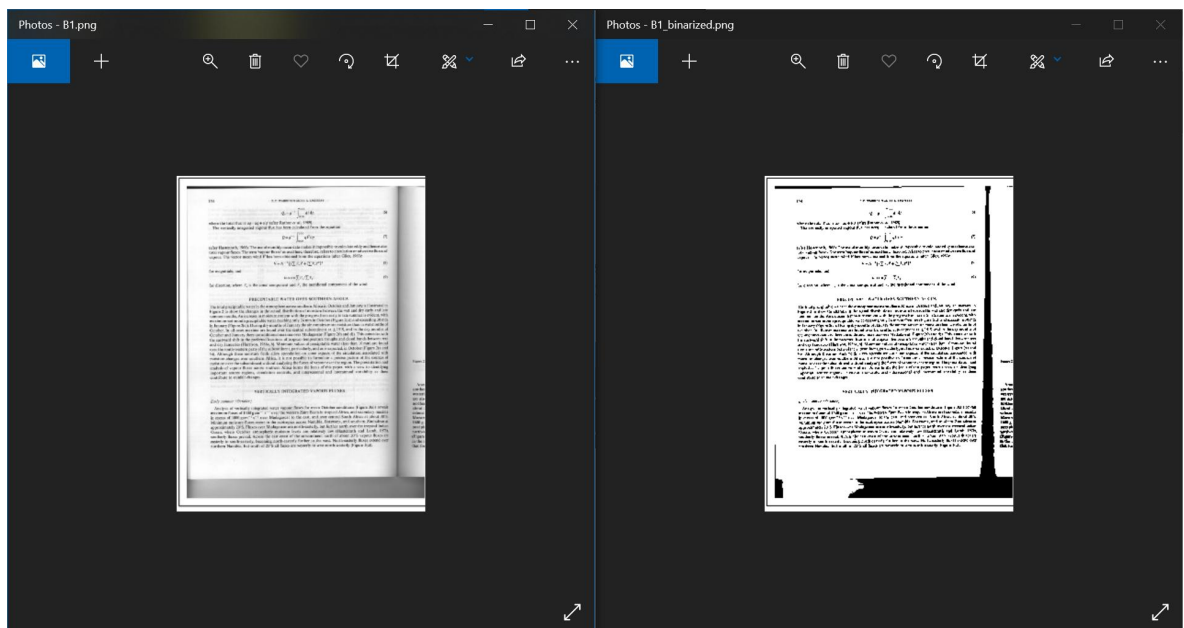
## OUTPUT

B1.png

```
Command Prompt

D:\Folders\SEECs\Semester 5\DIP\Lab\Lab 3>python Task1.py B1.png

D:\Folders\SEECs\Semester 5\DIP\Lab\Lab 3>_
```



B2.jpg

```
Command Prompt

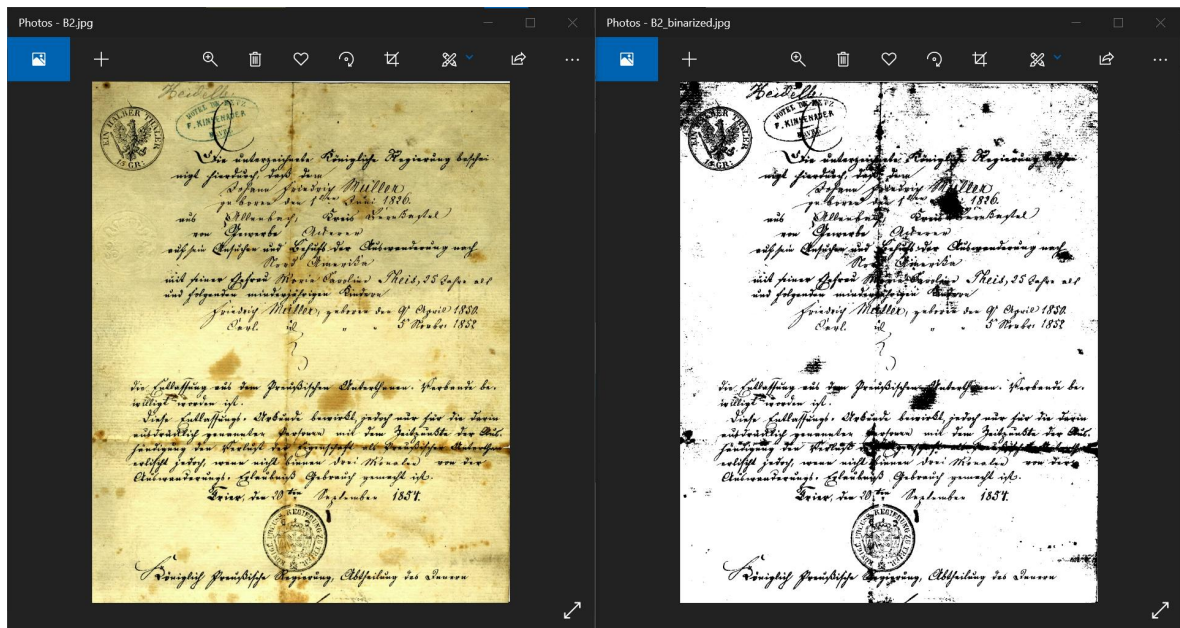
D:\Folders\SEECs\Semester 5\DIP\Lab\Lab 3>python Task1.py B2.jpg

D:\Folders\SEECs\Semester 5\DIP\Lab\Lab 3>_
```

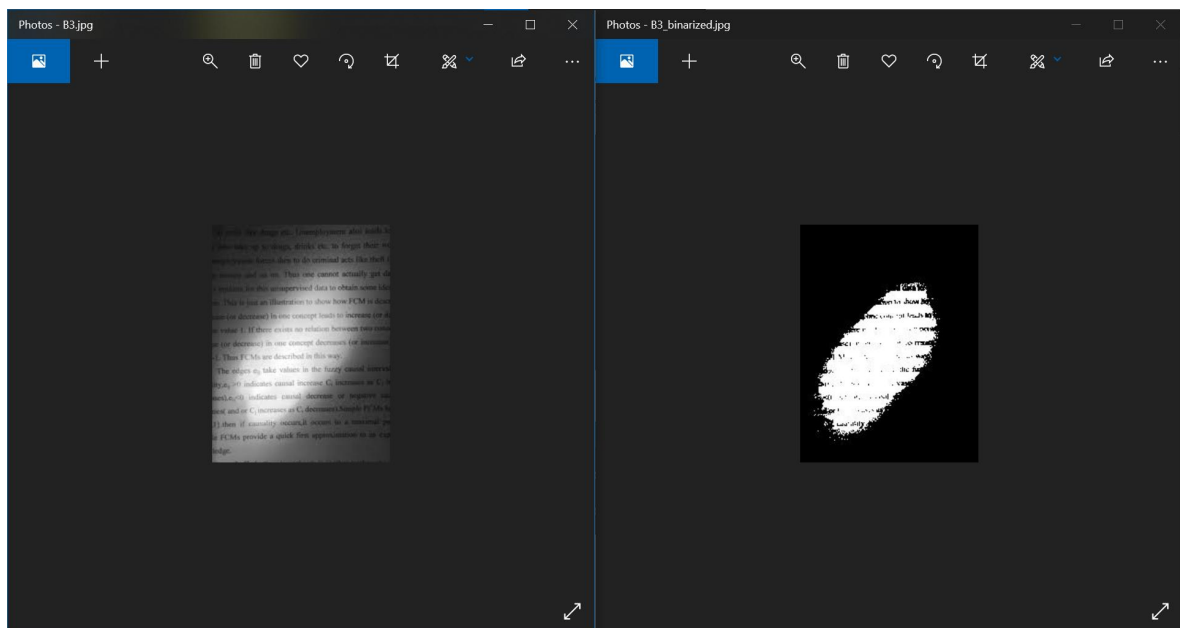
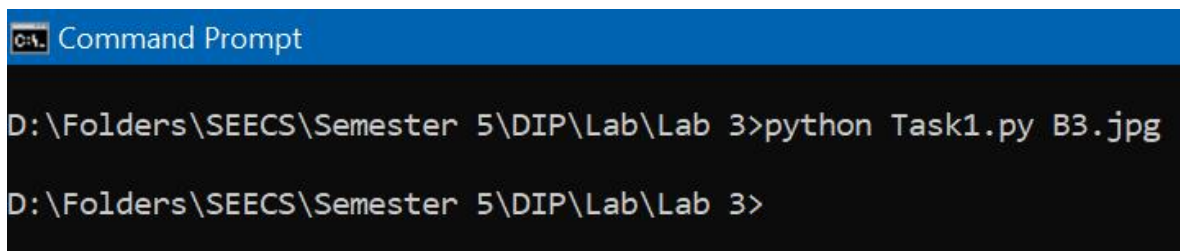


# National University of Sciences and Technology (NUST)

## School of Electrical Engineering and Computer Science



B3.jpg



## Task #2: Create Intensity Histogram from a Greyscale image





Hint: As discussed during the lab demo.

## **CODE**

```
import os, sys
from PIL import Image
from matplotlib import pyplot as plt

# get all the pixels in an array
def histogramArray(img):
    histogram_array = []
    # load the pixels of the image
    pix = img.load()
    # get width and height of the input image
    width, height = img.size

    # iterate through all the pixels
    for x in range(width):
        for y in range(height):
            histogram_array.append(pix[x,y])
    return histogram_array

def makeHistogram(array):
    # make histogram
    plt.hist(array, bins = [i for i in range(0,256)])

# gets all the files from the command line arguments
for infile in sys.argv[1:]:
    try:
        f, e = os.path.splitext(infile)
        # open image and convert it to grayscale
        img = Image.open(infile).convert('L')

        makeHistogram(histogramArray(img))

        # save histogram
        plt.savefig(f + "_histogram" + e)

    except IOError:
        print("Error")
```



```
1  import os, sys
2  from PIL import Image
3  from matplotlib import pyplot as plt
4
5  # get all the pixels in an array
6  def histogramArray(img):
7      histogram_array = []
8      # load the pixels of the image
9      pix = img.load()
10     # get width and height of the input image
11     width, height = img.size
12
13     # iterate through all the pixels
14     for x in range(width):
15         for y in range(height):
16             histogram_array.append(pix[x,y])
17     return histogram_array
18
19  def makeHistogram(array):
20     # make histogram
21     plt.hist(array, bins = [i for i in range(0,256)])
22
23  # gets all the files from the command line arguments
24  for infile in sys.argv[1:]:
25     try:
26         f, e = os.path.splitext(infile)
27         # open image and convert it to grayscale
28         img = Image.open(infile).convert('L')
29
30         makeHistogram(histogramArray(img))
31
32         # save histogram
33         plt.savefig(f + "_histogram" + e)
34
35     except IOError:
36         print("Error")
```

## OUTPUT

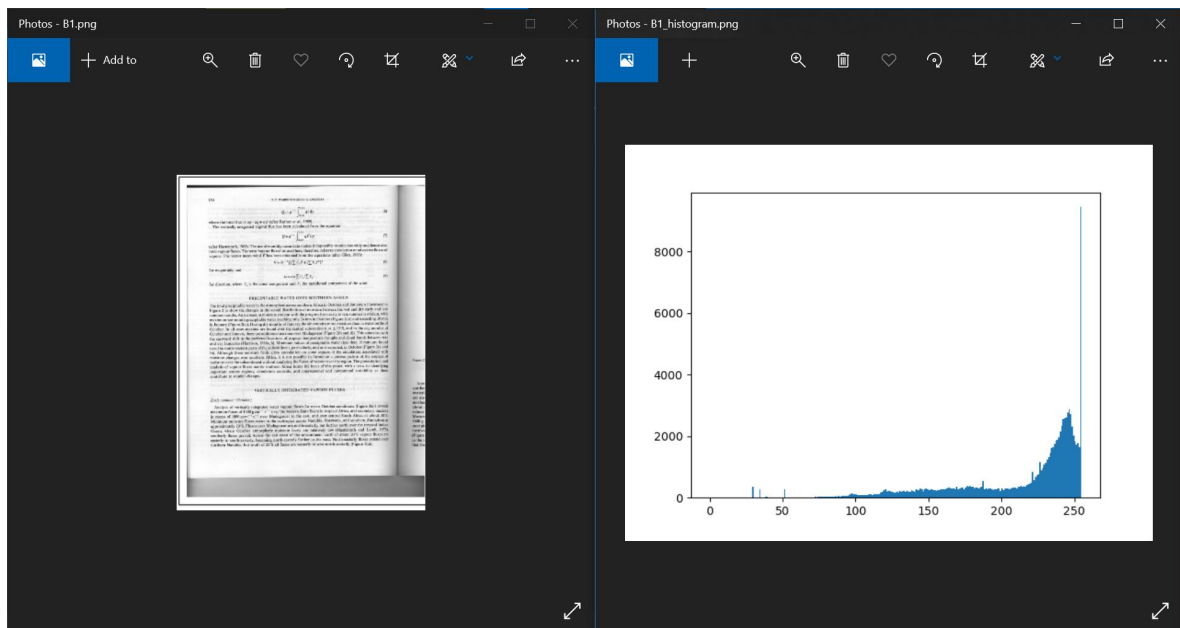
B1.png



# National University of Sciences and Technology (NUST) School of Electrical Engineering and Computer Science

## Command Prompt

```
D:\Folders\SEECs\Semester 5\DIP\Lab\Lab 3>python Task2.py B1.png  
D:\Folders\SEECs\Semester 5\DIP\Lab\Lab 3>
```



## B2.jpg

## Command Prompt

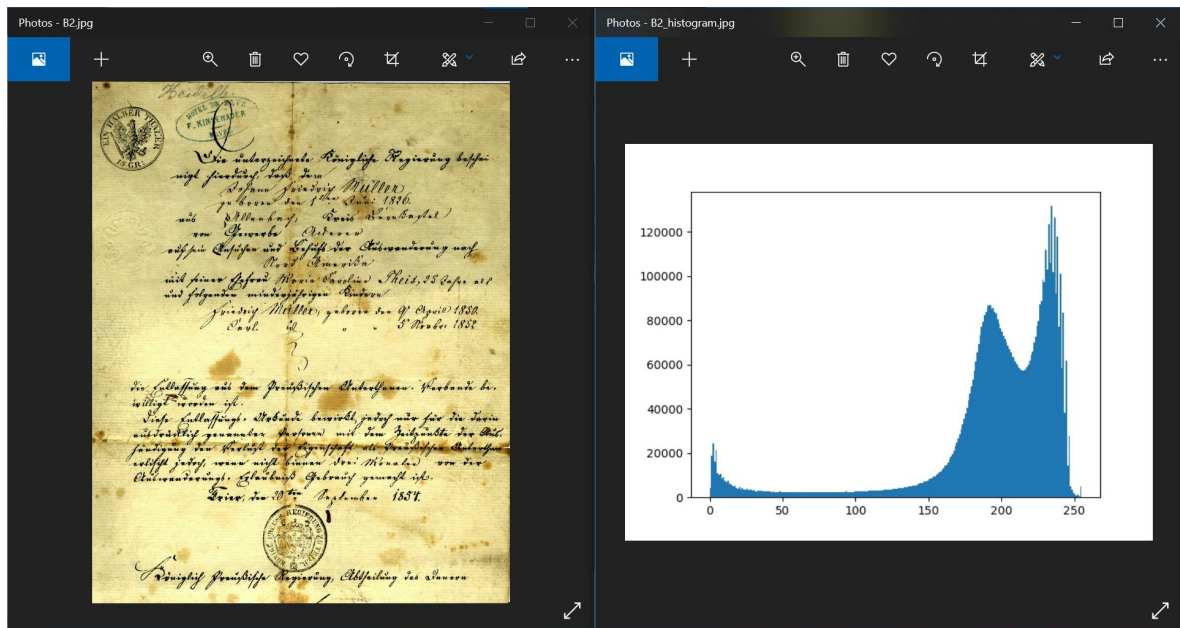
```
D:\Folders\SEECs\Semester 5\DIP\Lab\Lab 3>python Task2.py B2.jpg  
D:\Folders\SEECs\Semester 5\DIP\Lab\Lab 3>
```





# National University of Sciences and Technology (NUST)

## School of Electrical Engineering and Computer Science

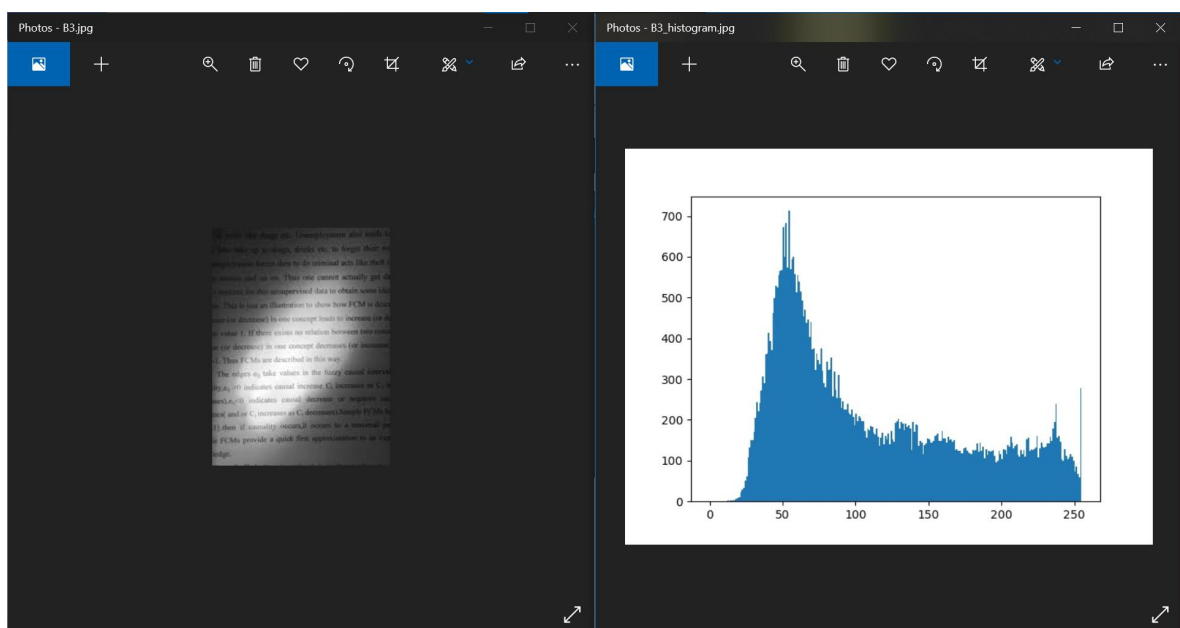


B3.jpg

```
Command Prompt

D:\Folders\SEECs\Semester 5\DIP\Lab\Lab 3>python Task2.py B3.jpg

D:\Folders\SEECs\Semester 5\DIP\Lab\Lab 3>
```



### Task #3: Recursive XY-cut algorithm



Hint: As discussed in the demo.  
Apply it on the following image :- xycuts

## CODE

```
import os
from PIL import Image
from matplotlib import pyplot as plt

# returns the array that is used to make the histogram
def histogramArray(img):
    histogram_array = []
    # load the pixels of the image
    pix = img.load()
    # get width and height of the input image
    width, height = img.size

    # iterate through all the pixels
    for x in range(height):
        # keeps count of the number of pixels in one row
        black_pixels = 0
        for y in range(width):
            # if pixel is black, increment black_pixels
            if pix[y, x] == 0:
                black_pixels += 1
        # append the number of black_pixels in a row
        histogram_array.append(black_pixels)

    return histogram_array

# make histogram
def makeHistogram(array, img):
    plt.bar([i for i in range(0, img.height)], array)

# draw a line in rows with no or less black pixels
def makeLines(img):
    pix = img.load()
    for x in range(img.height):
        if x == 135 or x == 260 or x == 390:
            for i in range(img.width):
                pix[i, x] = 0
    return img

infile = "XY-cuts.png"
try:
    f, e = os.path.splitext(infile)
    # open image and convert it to grayscale
    img = Image.open(infile).convert('L')
```



```
makeHistogram(histogramArray(img), img)  
img = makeLines(img)
```

```
# save histogram  
plt.savefig(f + "_histogram" + e)
```

```
# save the new image  
img.save(f + "_separated lines" + e)
```

```
except IOError:  
    print("Error")
```

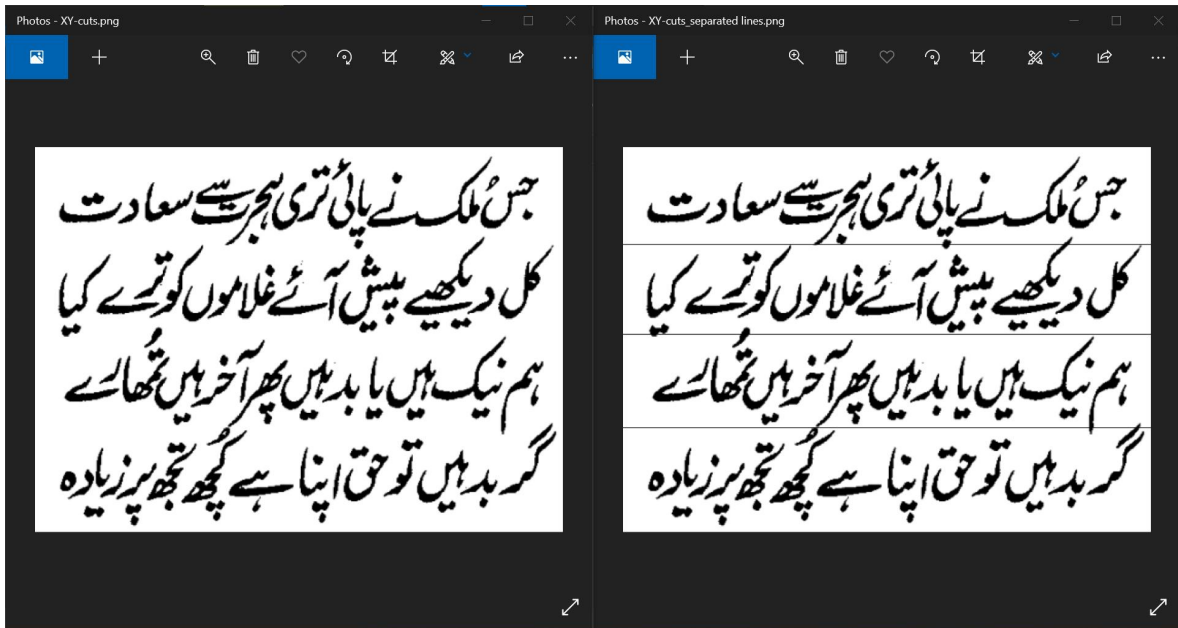
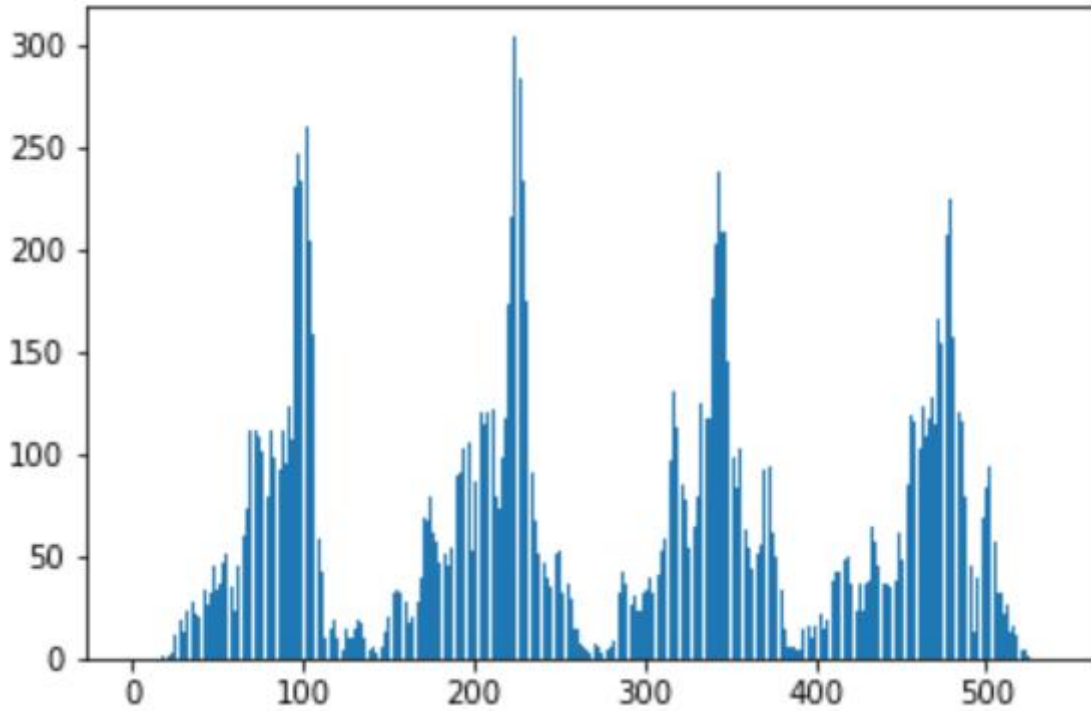
```
1  import os  
2  from PIL import Image  
3  from matplotlib import pyplot as plt  
4  
5  # returns the array that is used to make the histogram  
6  def histogramArray(img):  
7      histogram_array = []  
8      # load the pixels of the image  
9      pix = img.load()  
10     # get width and height of the input image  
11     width, height = img.size  
12  
13     # iterate through all the pixels  
14     for x in range(height):  
15         # keeps count of the number of pixels in one row  
16         black_pixels = 0  
17         for y in range(width):  
18             # if pixel is black, increment black_pixels  
19             if pix[y, x] == 0:  
20                 black_pixels += 1  
21         # append the number of black_pixels in a row  
22         histogram_array.append(black_pixels)  
23  
24     return histogram_array  
25
```



```
26 # make histogram
27 def makeHistogram(array, img):
28     plt.bar([i for i in range(0, img.height)], array)
29
30 # draw a line in rows with no or less black pixels
31 def makeLines(img):
32     pix = img.load()
33     for x in range(img.height):
34         if x == 135 or x == 260 or x == 390:
35             for i in range(img.width):
36                 pix[i, x] = 0
37     return img
38
39 infile = "XY-cuts.png"
40 try:
41     f, e = os.path.splitext(infile)
42     # open image and convert it to grayscale
43     img = Image.open(infile).convert('L')
44
45     makeHistogram(histogramArray(img), img)
46     img = makeLines(img)
47
48     # save histogram
49     plt.savefig(f + "_histogram" + e)
50
51     # save the new image
52     img.save(f + "_separated lines" + e)
53
54 except IOError:
55     print("Error")
```

OUTPUT





Hand in

Submit a lab report containing both the code and screenshots of output.

To Receive Credit

1. By showing up on time for lab, working on the lab solution, and staying to the end of the class period, only then you can receive full credit for the lab





## National University of Sciences and Technology (NUST) School of Electrical Engineering and Computer Science

assignment.

2. Comment your program heavily. Intelligent comments and a clean, readable formatting of your code account for 20% of your grade.

**Due date:**  
**Check on lms**