

```
1 NAME = "Loknath Saha"
2 ID = "28201817"
```

▼ Necessary library import

```
1 import numpy as np
2 from skimage import io, color, exposure, img_as_float
3 import matplotlib.pyplot as plt
```

▼ Task 1 - Basic Image Operation

import your image or any photo taken by you (sample.jpg) as a numpy array, save it in the variable I.

A picture taken from your phone of any scenery/streets/building is better.

remember your image name MUST Be sample.jpg

Make sure the height and the width of the image is smaller than 1000 pixels.

```
1 I = io.imread("sample.jpg") # Replace None with appropriate function call line
2
3
4 # find the height and the width of the image
5 H,W = I.shape[1:2]
6
7 print("Height is", H)
8 print("Width is", W)
9
10 Height is 750
11 Width is 750
12
13 # Normalize the image so that the gray scales are between 0 and 1. Save it to I and display the image
14 I = io.imread("sample.jpg")
15
16 ## BEGIN SOLUTION
17 I2 = color.rgb2gray(I)
18 print(I2.shape)
19 I3 = img_as_float(I2)
20 for i in range(12,255):
21     I3[i] = i/255
22
23 plt.imshow(I3,cmap='gray')
24 plt.title('Normalized Image')
25 plt.axis('off')
26 plt.show()
27
28 ## END SOLUTION
```

Normalized Image



```
1 # Increase the brightness of the image without changing the contrast.
2 # Save the resulting image in I_bright and display it.
3 I_bright = np.clip(I+0.4, 0, 1)
4 plt.imshow(I_bright, cmap='gray')
5 plt.title('Brightened Image')
6 plt.axis('off')
7 plt.show()
```

Brightened Image



```
1 # Decrease the brightness of the image without changing the contrast.
2 # Save the resulting image in I_dark and display it.
3 I_dark = np.clip(I-0.2, 0, 1)
4
5 plt.imshow(I_dark, cmap='gray')
6 plt.title('Darkened Image')
7 plt.axis('off')
8 plt.show()
9
10 #io.imsave('dark_image.jpg', (I_dark * 255).astype(np.uint8))
```

Darkened Image



```
1 # Multiply the three channels of image I with three DIFFERENT numbers between 0.3 and
2 # Save the resulting image in I_tint and display it.
```

```

3 # The resulting image should have some color shift
4 I_tint = None
5
6
7 # HINT:
8 # I_tint[:, :, 0] = np.zeros(I.shape)
9 # I_tint[:, :, 1] = ..... I[:, :, 0].....
10 # .....
11
12 ##### BEGIN SOLUTION
13 I_tint = np.zeros(I.shape,dtype=np.float64)
14 scale_r = 1.5
15 scale_g = 0.7
16 scale_b = 2.0
17
18
19 I_tint[:, :, 0] = I[:, :, 0] * scale_r
20 I_tint[:, :, 1] = I[:, :, 1] * scale_g
21 I_tint[:, :, 2] = I[:, :, 2] * scale_b
22
23 I_tint = np.clip(I_tint, 0, 255).astype(np.uint8)
24
25 plt.imshow(I_tint)
26 plt.title('Tinted Image with Color Shift')
27 plt.axis('off')
28 plt.show()
29
30 ##### END SOLUTION

```

Tinted Image with Color Shift



```

1 # Convert the image into a grayscale image.
2 # Save it to I_gray and display it
3 # (gray=0.2126*R+0.7152*G+0.0722*B
4 I_gray = None
5
6 ##### BEGIN SOLUTION
7
8 #for I_gray = 0.2126 * I[:, :, 0] + 0.7152 * I[:, :, 1] + 0.0722 * I[:, :, 2]
9 I_gray = color.rgb2gray(I)
10 I_gray = img_as_float(I_gray)
11
12
13 plt.imshow(I_gray, cmap='gray')
14 plt.title('Converted Grayscale Image')
15 plt.axis('off')
16 plt.show()
17
18 ##### END SOLUTION

```

Converted Grayscale Image



```

1 # Display the negative of the grayscale image
2
3 ##### BEGIN SOLUTION
4 I_negative = 1 - I_gray
5 plt.imshow(I_negative, cmap='gray')
6 plt.title('Negative of Grayscale Image')
7 plt.axis('off')
8 plt.show()
9 ##### END SOLUTION

```

Negative of Grayscale Image



```

1 # Artificially degrade the grayscale image by reducing its contrast
2 # You can do so by recaling the gray values and concentrating them in a narrow range
3 # say between 0.3 and 0.6.
4 # Save the image as I_degraded and display it
5 # HINT: SEE lec-4-demo-codes
6
7 new_min = new_max = 0.4, 0.5
8 I_degraded = img_as_float(I_gray)
9
10 I_degraded = (I_degraded * (new_max - new_min)) + new_min
11 I_degraded = np.clip(I_degraded, 0, 1)
12
13
14 plt.figure(figsize=(10, 5))
15 plt.subplot(1, 2, 1)
16 plt.imshow(I_gray, cmap='gray')
17 plt.title('Original Image')
18 plt.axis('off')

```

```

19
20 plt.imshow(l, 2, 2)
21 plt.imshow(l_degraded, cmap='gray')
22 plt.title('Degraded Contrast Image')
23 plt.axis('off')
24
25 plt.tight_layout()
26 plt.show()
27
28

```



```

1 # Complete the following function to perform Piecewise Linear Contrast stretching
2 # That is, implement the map shown in Slide 17 of Lecture 3
3
4 # Prototype: piecewise_contrast_stretch(I_gray, r1, r2, s1, s2)
5 # Assuming both input and output images are normalized between 0 and 1
6
7 def piecewise_contrast_stretch(l, r1, r2, s1, s2):
8     pass
9
10 # Write your code here
11 l_stretched = np.zeros_like(l)
12
13 # Apply the first mapping [0, r1] -> [0, s1]
14 mask1 = l <= r1
15 l_stretched[mask1] = (s1 / r1) * l[mask1]
16
17 # Apply the second mapping [r2, 1] -> [s2, 1]
18 mask2 = l >= r2
19 l_stretched[mask2] = ((1 - s2) / (1 - r2)) * (l[mask2] - r2) + s2
20
21 # Apply the third mapping [r1, r2] -> [s1, s2]
22 mask3 = (l >= r1) & (l <= r2)
23 l_stretched[mask3] = ((s2 - s1) / (r2 - r1)) * (l[mask3] - r1) + s1
24
25 return l_stretched
26
27

```

```

1 # To test your implementation, contrast stretch the degraded image l_degrade
2 r1 = 0.2
3 r2 = 0.7
4 s1 = 0.1
5 s2 = 0.9
6 l_stretched = piecewise_contrast_stretch(l_degraded, r1, r2, s1, s2)
7
8 plt.imshow(l_stretched, cmap='gray')
9 plt.title('Piecewise Linear Contrast Stretched Image')
10 plt.axis('off')
11 plt.show()

```



Task 2 - Histogram and Equalization

```

1 # Plot the Image and its histogram + cdf of the original image I
2 # Note that it is a color image, so it will have three different histograms
3
4 ## BEGIN SOLUTION
5 I = io.imread("sample.jpg")
6
7 R = I[:, :, 0]
8 G = I[:, :, 1]
9 B = I[:, :, 2]
10
11 def plot_histogram_and_cdf(channel, color_name):
12     hist, bins = exposure.histogram(channel, nbins=256, normalize=False)
13     cdf, cdf_bins = exposure.cumulative_distribution(channel, 256)
14     plt.plot(bins, hist, label=f'{color_name} Histogram', color=color_name)
15
16     plt.twinx()
17     plt.plot(cdf_bins, cdf, label=f'{color_name} CDF', color=color_name, linestyle='--')
18
19 plt.figure(figsize=(12, 5))
20 plt.subplot(2, 2, 1)
21 plt.imshow(I)
22 plt.title('Original Image')
23 plt.axis('off')
24
25 plt.subplot(2, 2, 2)
26 plt.hist(R, 'red')
27 plt.title('Red Channel Histogram + CDF')
28
29 plt.subplot(2, 2, 3)
30 plt.hist(G, 'green')
31 plt.title('Green Channel Histogram + CDF')
32
33 plt.subplot(2, 2, 4)
34 plt.hist(B, 'blue')
35 plt.title('Blue Channel Histogram + CDF')
36
37 plt.tight_layout()
38 plt.show()

```

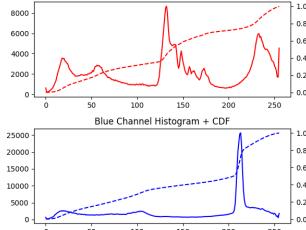
```

39
40
41 plt.subplot(2, 2, 4)
42 plot_histogram_and_cdf(b, 'blue')
43 plt.title('Blue Channel Histogram + CDF')
44
45 plt.tight_layout()
46 plt.show()
47
48 ## END SOLUTION

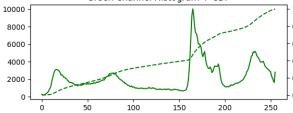
```



Red Channel Histogram + CDF



Green Channel Histogram + CDF



Blue Channel Histogram + CDF


```

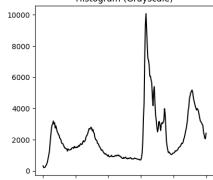
1 # Plot the Image and its histogram + cdf of the grayscale image l_gray
2
3 ## BEGIN SOLUTION
4
5 hist, bins = exposure.histogram(l_gray, nbins=256, normalize=False)
6 cdf, cdf_bins = exposure.cumulative_distribution(l_gray, 256)
7
8 plt.figure(figsize=(12, 4))
9 plt.subplot(1, 3, 1)
10 plt.imshow(l_gray, cmap='gray')
11 plt.title('Original Image')
12 plt.axis('off')
13
14 plt.subplot(1, 3, 2)
15 plt.plot(bins, hist, color='black')
16 plt.title('Histogram (Grayscale)')
17
18 plt.subplot(1, 3, 3)
19 plt.plot(cdf_bins, cdf, color='red')
20 plt.title('CDF (Grayscale)')
21
22 plt.tight_layout()
23 plt.show()
24 ## END SOLUTION

```

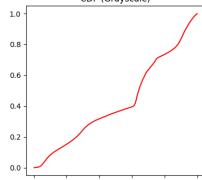
Grayscale Image



Histogram (Grayscale)



CDF (Grayscale)

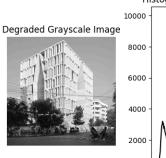


```

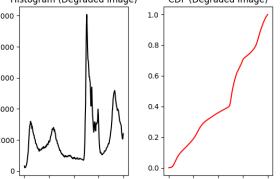
1 # Plot the Image and its histogram + cdf of the degraded image l_degraded
2
3 ## BEGIN SOLUTION
4 hist, bins = exposure.histogram(l_degraded, nbins=256, normalize=False)
5 cdf, cdf_bins = exposure.cumulative_distribution(l_degraded, 256)
6
7 plt.figure(figsize=(8,4))
8
9 plt.subplot(1, 3, 1)
10 plt.imshow(l_degraded, cmap='gray')
11 plt.title('Degraded Grayscale Image')
12 plt.axis('off')
13
14 plt.subplot(1, 3, 2)
15 plt.plot(bins, hist, color='black')
16 plt.title('Histogram (Degraded Image)')
17
18 plt.subplot(1, 3, 3)
19 plt.plot(cdf_bins, cdf, color='red')
20 plt.title('CDF (Degraded Image)')
21
22 plt.tight_layout()
23 plt.show()
24
25 ## END SOLUTION

```

Degraded Grayscale Image



Histogram (Degraded image)



CDF (Degraded image)


```

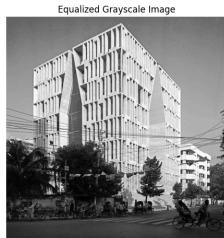
1 # Equalize the histogram of the degraded image l_degraded
2 # Save the result in l_recon_gray, display the image along with its histogram
3
4 l_recon_gray = exposure.equalize_hist(l_degraded)
5 hist_recon, bins_recon = exposure.histogram(l_recon_gray, nbins=256, normalize=False)
6
7 plt.figure(figsize=(10, 5))
8
9 plt.subplot(1, 2, 1)
10 plt.imshow(l_recon_gray, cmap='gray')

```

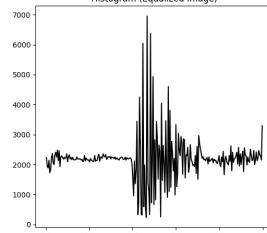
```

11 plt.title('Equalized Grayscale Image')
12 plt.axis('off')
13
14 plt.subplot(1, 2, 2)
15 plt.plot(bins_recon, hist_recon, color='black')
16 plt.title("Histogram (Equalized Image)")
17
18 plt.tight_layout()

```



Equalized Grayscale Image

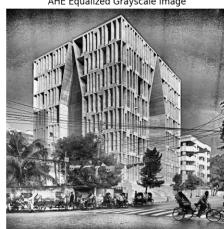


```

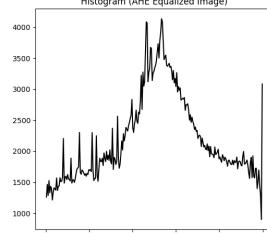
1 # Equalize the histogram of the degraded image I_degraded using AHE
2 # Save the result in I_recon_gray_2, display the image along with its histogram
3
4 I_recon_gray_2 = exposure.equalize_adapthist(I_degraded,kernel_size=(128, 128),clip_limit=0)
5
6 hist_recon_2, bins_recon_2 = exposure.histogram(I_recon_gray_2, nbins=256, normalize=False)
7
8 plt.figure(figsize(18, 5))
9
10 plt.subplot(1, 2, 1)
11 plt.imshow(I_recon_gray_2, cmap='gray')
12 plt.title('AHE Equalized Grayscale Image')
13 plt.axis('off')
14
15 plt.subplot(1, 2, 2)
16 plt.plot(bins_recon_2, hist_recon_2, color='black')
17 plt.title("Histogram (AHE Equalized Image)")
18
19 plt.tight_layout()
20 plt.show()

```

AHE Equalized Grayscale Image



Histogram (AHE Equalized Image)



```

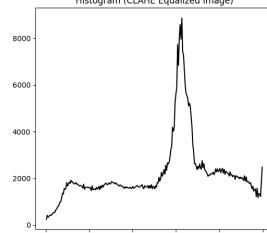
1 # Equalize the histogram of the degraded image I_degraded using CLAHE
2 # Save the result in I_recon_gray_3, display the image along with its histogram
3
4 I_recon_gray_3 = exposure.equalize_adapthist(I_degraded,kernel_size=(64, 64), clip_limit=0.01)
5
6 hist_recon_3, bins_recon_3 = exposure.histogram(I_recon_gray_3, nbins=256, normalize=False)
7
8 plt.figure(figsize(18, 5))
9
10 plt.subplot(1, 2, 1)
11 plt.imshow(I_recon_gray_3, cmap='gray')
12 plt.title('CLAHE Equalized Grayscale Image')
13 plt.axis('off')
14
15 plt.subplot(1, 2, 2)
16 plt.plot(bins_recon_3, hist_recon_3, color='black')
17 plt.title("Histogram (CLAHE Equalized Image)")
18
19 plt.tight_layout()
20 plt.show()

```

CLAHE Equalized Grayscale Image



Histogram (CLAHE Equalized Image)



```

1 # Artificially degrade the original **RGB image** by reducing its contrast
2 # You can do so by recalling the values of the L channel (in LAB color space)
3 # and concentrating them in a narrow range, say between 0.3 and 0.6.
4 # Save the image as I_rgb_degraded and display it
5
6
7
8 *** BEGIN SOLUTION
9 I_lab = color.rgb2lab(I)
10 l = I_lab[:, :, 0]
11

```

```

12 new_min, new_max = 0.3 * 100, 0.6 * 100
13
14 L_norm = (L - L.min()) / (L.max() - L.min())
15
16 L_degraded = L_norm * (new_max - new_min) + new_min
17
18 L_lab_degraded = L_lab.copy()
19 L_lab_degraded[:, :, 0] = L_degraded
20
21 L_rgb_degraded = color.lab2rgb(L_lab_degraded)
22
23 plt.figure(figsize(12, 6))
24 plt.imshow(L_rgb_degraded[1, 2, 1])
25 plt.show()
26 plt.title('Original RGB Image')
27 plt.axis('off')
28
29 plt.subplot(1, 2, 2)
30 plt.imshow(L_rgb_degraded)
31 plt.title('Degraded RGB Image')
32 plt.axis('off')
33
34 plt.tight_layout()
35 plt.show()
36
37 print("Original L channel range: [{L.min():.2f}, {L.max():.2f}]")
38 print("Degraded L channel range: [{L_degraded.min():.2f}, {L_degraded.max():.2f}]")
39
40 ## END SOLUTION

```

Original RGB Image



Degraded RGB Image



Original L channel range: [0.00, 100.00]
Degraded L channel range: [30.00, 60.00]

```

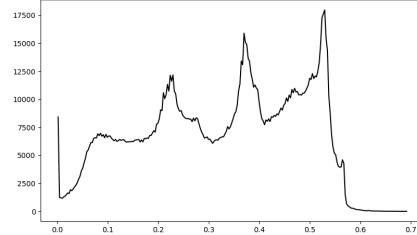
1 # Equalize the histogram of the degraded color image L_rgb_degraded using CLAHE
2 # Save the result in L_recon_color, display the image along with its histogram
3 # Hint: You have to convert to LAB first
4 # See the lecture and lecture-4-demo-codes
5 L_lab_degraded = color.rgb2lab(L_rgb_degraded)
6
7 m1 = L_lab_degraded[:, :, 0].min()
8 m2 = L_lab_degraded[:, :, 0].max()
9
10 L_degraded = L_lab_degraded[:, :, 0] / m2
11
12 L_degraded_clahe = exposure.equalize_adapthist(L_degraded,kernel_size=(64, 64), clip_limit=0.01)
13
14 L_lab_recon = L_lab_degraded.copy()
15 L_lab_recon[:, :, 0] = L_degraded_clahe * m2
16
17
18 L_recon_color = color.lab2rgb(L_lab_recon)
19
20
21 hist_recon_color, bins_recon_color = exposure.histogram(L_recon_color, nbins=256, normalize=False)
22
23
24 plt.figure(figsize(15, 5))
25
26
27 plt.subplot(1, 2, 1)
28 plt.imshow(L_recon_color)
29 plt.title('CLAHE Equalized RGB Image')
30 plt.axis('off')
31
32 plt.subplot(1, 2, 2)
33 plt.plot(hist_recon_color, bins_recon_color, color='black')
34 plt.title('Histogram (CLAHE Equalized Image)')
35
36 plt.tight_layout()
37 plt.show()

```

CLAHE Equalized RGB Image



Histogram (CLAHE Equalized Image)



▼ Task 3 - Open Ended

There are 3 images in the drive directory below. Look at the questions from the brackets [].

Answer them in the provided text cell at the bottom.

link: <https://drive.google.com/drive/folders/1f3XrO-MGxhL2PfcLCFOjU0wvLAv4?usp=sharing>

```

1 # Dark_Room.jpg = very dark [The windows are on walls. How does the wall look like?]
2 # Foggy_Road.jpg = washed out/foggy [How many vehicles do you think there are?]

```

```

3 # Read_the_code.jpg = Dark RGB Barcode [What is hidden in the Barcode?
4 #                                     Make it scanable, scan it and say something about the hidden message.
5
6 # Your task is to improve these images using
7 # contrast stretching, histogram equalization, AHE or CLAHE
8 # try different combination of parameter settings to see which produces the best result
9
10 ## BEGIN SOLUTION
11

```

⌄ Your answers:

abc abc abc ...

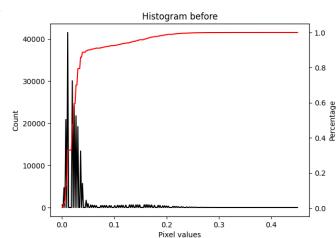
```

1 I_dark_room = io.imread('Dark_Room.jpg')
2 I_dark_room.shape
3 (424, 600, 3)

1 I_dark_room_gray = color.rgb2gray(I_dark_room)

1 hist, bins_hist = exposure.histogram(I_dark_room_gray, nbins=256, normalize=False)
2 plt.plot(bins_hist, hist, 'k')
3 plt.xlabel("Pixel values")
4 plt.ylabel("Count")
5 plt.title("Histogram before")
6
7 plt.twinx()
8 img_cdf, bins = exposure.cumulative_distribution(I_dark_room_gray, 256)
9 plt.plot(bins, img_cdf, "red")
10 plt.ylabel("%Percentage")
11 plt.show()

```

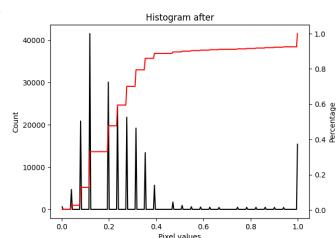


```

1 I_stretched = np.clip(10*(I_dark_room_gray), 0, 1)

1 hist, bins_hist = exposure.histogram(I_stretched, nbins=256, normalize=False)
2 plt.plot(bins_hist, hist, 'k')
3 plt.xlabel("Pixel values")
4 plt.ylabel("Count")
5 plt.title("Histogram after")
6
7
8 plt.twinx()
9 img_cdf, bins = exposure.cumulative_distribution(I_stretched, 256)
10 plt.plot(bins, img_cdf, "red")
11 plt.ylabel("%Percentage")
12 plt.show()

```



```

1 plt.figure(figsize(8, 4))
2 io.imshow(I_stretched)
3 plt.axis("off")
4 plt.show()

1 I_dark_room_clahe = exposure.equalize_adapthist(I_dark_room_gray,kernel_size=(64, 64), clip_limit=0.03
2
3
4 hist, bins_hist = exposure.histogram(I_dark_room_clahe , nbins=256, normalize=False)
5
6 plt.figure(figsize(10, 4))
7 plt.plot(bins_hist, hist, 'k')
8 plt.xlabel("Pixel values")
9 plt.ylabel("Count")
10 plt.title("Histogram before")
11
12 plt.twinx()
13 img_cdf, bins = exposure.cumulative_distribution(I, 256)
14 plt.plot(bins, img_cdf, "red")
15 plt.ylabel("%Percentage")
16 plt.show()

```



```

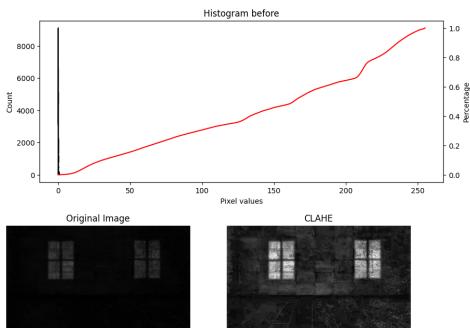
1 I_dark_room_clahe = exposure.equalize_adapthist(I_dark_room_gray,kernel_size=(64, 64), clip_limit=0.03
2
3
4 hist, bins_hist = exposure.histogram(I_dark_room_clahe , nbins=256, normalize=False)
5
6 plt.figure(figsize(10, 4))
7 plt.plot(bins_hist, hist, 'k')
8 plt.xlabel("Pixel values")
9 plt.ylabel("Count")
10 plt.title("Histogram before")
11
12 plt.twinx()
13 img_cdf, bins = exposure.cumulative_distribution(I, 256)
14 plt.plot(bins, img_cdf, "red")
15 plt.ylabel("%Percentage")
16 plt.show()

```

```

17
18 plt.figure(figsize=(10, 10))
19
20 plt.subplot(1, 2, 1)
21 plt.imshow(I_dark_room)
22 plt.title('Original Image')
23 plt.axis('off')
24
25 plt.subplot(1, 2, 2)
26 plt.imshow(I_dark_room_clahe, cmap='gray')
27 plt.title('CLAHE')
28 plt.axis('off')
29
30 plt.show()
31

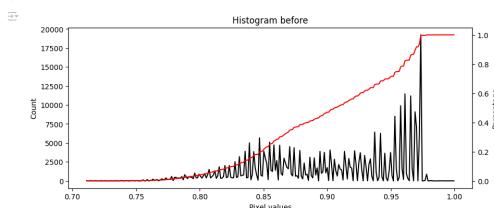
```



```

1 I_foggy_road = io.imread('Foggy_Road.jpg')
2
3 I_foggy_road_gray = color.rgb2gray(I_foggy_road)
4
5
6 hist, bins_hist = exposure.histogram(I_foggy_road_gray, nbins=256, normalize=False)
7
8 plt.figure(figsize=(10, 4))
9 plt.plot(bins_hist, hist, 'k')
10 plt.xlabel("Pixel values")
11 plt.ylabel("#Count")
12 plt.title("Histogram before")
13
14 plt.twinx()
15 img_cdf, bins = exposure.cumulative_distribution(I_foggy_road_gray, 256)
16 plt.plot(bins, img_cdf, "red")
17 plt.ylabel("%Percentage")
18 plt.show()
19

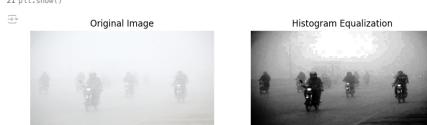
```



```

1 # Apply Histogram Equalization
2 img = exposure.equalize_hist(I_foggy_road_gray)
3
4 plt.figure(figsize=(10, 5))
5 plt.subplot(1, 2, 1)
6 plt.imshow(I_foggy_road)
7 plt.title('Original Image')
8 plt.axis('off')
9
10 plt.subplot(1, 2, 2)
11 plt.imshow(img)
12 plt.title('Histogram Equalization')
13 plt.axis('off')
14
15
16 plt.subplot(1, 2, 2)
17 plt.imshow(img, cmap='gray')
18 plt.title('Histogram Equalization')
19 plt.axis('off')
20
21 plt.show()

```



```

1 # Apply CLAHE
2 I_foggy_road_clahe = exposure.equalize_adapthist(I_foggy_road_gray,kernel_size=(64, 64), clip_limit=0.01
3
4
5 plt.figure(figsize=(10, 5))
6
7 plt.subplot(1, 2, 1)
8 plt.imshow(I_foggy_road)
9 plt.title('Original Image')
10 plt.axis('off')
11
12 plt.subplot(1, 2, 2)
13 plt.imshow(I_foggy_road_clahe, cmap='gray')
14 plt.title('CLAHE')
15 plt.axis('off')

```



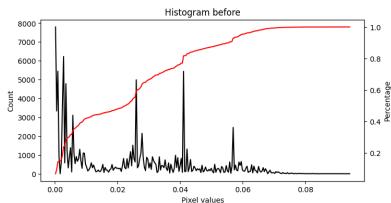
CLAHE



```

1 l_read_code = io.imread('Read_the_code.jpg')
2 l_read_code_gray = color.rgb2gray(l_read_code)
3
4
5
6 hist, bins_hist = exposure.histogram(l_read_code_gray, nbins=256, normalize=False)
7 plt.figure(figsize=(10, 4))
8 plt.plot(bins_hist, hist, 'k')
9 plt.xlabel("Pixel values")
10 plt.ylabel("%Count")
11 plt.title("Histogram before")
12
13 plt.twinx()
14 img_cdf = exposure.cumulative_distribution(l_read_code_gray, 256)
15 plt.plot(img_cdf, "red")
16 plt.ylabel("Percentage")
17 plt.show()
18

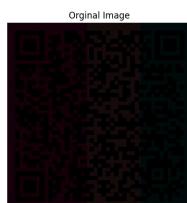
```



```

1 l_eq = exposure.equalize_hist(l_read_code_gray)
2 plt.figure(figsize=(10, 5))
3
4
5 plt.subplot(1, 2, 1)
6 plt.imshow(l_read_code)
7 plt.title('Original Image')
8 plt.axis('off')
9
10 plt.subplot(1, 2, 1)
11 plt.imshow(l_eq)
12 plt.title('Original Image')
13 plt.axis('off')
14
15
16 plt.subplot(1, 2, 2)
17 plt.imshow(l_eq, cmap='gray')
18 plt.title('Histogram Equalization')
19 plt.axis('off')
20
21 plt.show()

```



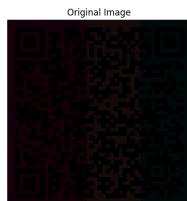
Histogram Equalization



```

1 l_read_code_clahe = exposure.equalize_adapthist(l_read_code_gray,kernel_size=(40, 40), clip_limit=0.02)
2
3
4
5 plt.figure(figsize=(10, 5))
6 plt.subplot(1, 2, 1)
7 plt.imshow(l_read_code)
8 plt.title('Original Image')
9 plt.axis('off')
10
11 plt.subplot(1, 2, 2)
12 plt.imshow(l_read_code_clahe, cmap='gray')
13 plt.title('CLAHE')
14 plt.axis('off')
15
16 plt.show()

```



CLAHE



