CS-GY 6643 Computer Vision
Assignment 1

Due on Feb 27, 2018 11:55 pm

Professor Guido Gerig

Tianyu Wang

Part A - Theoretical Questions

A1) Image formation

A professional full-frame digital camera uses an image size of 36mm x 36mm and standard focal length of 50mm. Let us say that the square sensor provide 16 megapixels. Now you buy smartphone with a 16 megapixel sensor (assuming a square image too), but given a focal length of 4mm so that the phone fits into your pocket.

- Using the pinhole camera projection equation, calculate the size of the light-sensitive image sensor of your smart-phone. Calculate the ratio of this size relative to the professional camera sensor size.
- Calculate the size of a sensor pixel element for the professional and your smart-phone cameras. Provide a short discussion of eventual advantages/disadvantages of your resulting measures, and reasons why some professionals or amateurs favor more expensive large cameras.
- Calculate the storage requirement assuming storage of raw images with color RGB channels, for both cameras.

A2) Connectivity foreground/background

One solution to digitization paradoxes is to mix connectivities. Using 8-neighborhood for foreground and 4-neighborhood for background, examine the paradoxes shown in the book (Fig. 2.7). Discuss the number of components of fore- versus background given this choice. Also discuss the #of components when either using 4-n (and also 8-n) for both fore- and background, and when reversing the notion and using 8-n for background and 4-n for foreground. You can discuss in words and also include sketches of your thoughts to this section.

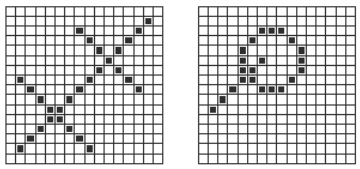

Figure 2.7: Paradoxes of crossing lines. © *Cengage Learning 2015.*

A3) Histogram equalization

Remember the main goal of histogram equalization to result in a uniform intensity distribution (histogram). Below you see the images from the book referring to image equalization. Explain why the histogram of a discrete image is not flat after histogram equalization. (Hint: You may first work on the practical part to get a closer insight).

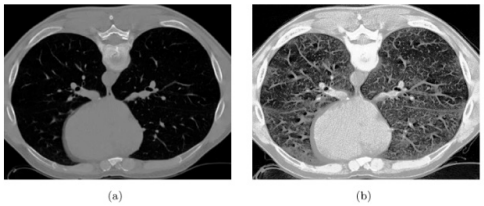

Figure 5.3: Histogram equalization. (a) Original image. (b) Equalized image. © *Cengage Learning 2015.*

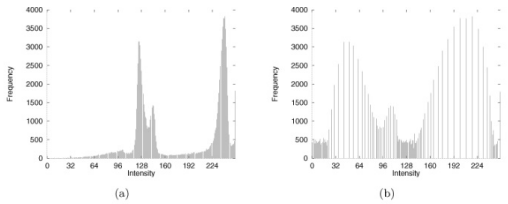

Figure 5.4: Histogram equalization: Original and equalized histograms corresponding to Figure 5.3a,b. © *Cengage Learning 2015.*

Answers

A1) Image formation

- According to the projection equation, we have:

$$\begin{aligned}\frac{Length\ of\ the\ edge}{Focal\ length} &= \frac{36\,mm}{50\,mm} \\ &= \frac{x}{4\,mm} \\ \Rightarrow\ x\ &=\ 2.88\,mm\end{aligned}$$

  So the size of the light-sensitive image sensor of your smart-phone should be $2.88\,mm \times 2.88\,mm$. And the ratio should be

$$ratio\ =\ \frac{(4\,mm)^2}{(50\,mm)^2}\ =\ \frac{4}{625}$$

- The square sensor provides 16 megapixels for a square image. So the size of the square image should be 4000×4000.

  (i) The size of a sensor pixel element for the professional digital camera:

$$\begin{aligned}Length\ of\ the\ edge\ of\ a\ pixel\ element\ &=\ \frac{36\,mm}{4000}\ =\ 0.009\,mm\ =\ 9\,\mu m \\ Size\ of\ a\ pixel\ element\ &=\ (Length\ of\ the\ edge\ of\ a\ pixel\ element)^2 \\ &= 81\,\mu m^2\end{aligned}$$

  (ii) The size of a sensor pixel element for the smart-phone camera:

$$\begin{aligned}Length\ of\ the\ edge\ of\ a\ pixel\ element\ &=\ \frac{2.88\,mm}{4000}\ =\ 0.00072\,mm\ =\ 0.72\,\mu m \\ Size\ of\ a\ pixel\ element\ &=\ (Length\ of\ the\ edge\ of\ a\ pixel\ element)^2 \\ &= 0.5184\,\mu m^2\end{aligned}$$

  (iii)

	Advantages	**Disadvantages**
Professional digital camera	**Quality**: high	**Size**: large
Smart-phone	**Size**: small	**Quality**: low

  (p.s: 'size' means 'size of a pixel element', not the real 'image size', which is the same for both.)

  (iv) **Conclusion**: Large professional digital cameras have better image quality. They have larger image element size, which means that the raster in a camera can absorb more light than smart-phone. Although cameras are much heavier and larger than a smart-phone, they can take better pictures and provide more details in a photo.

- The size of image taken by a digital camera or smart-phone is same with each other. So for one raw image:

$$\begin{aligned}Storage\ of\ raw\ image\ &=\ Pixels\ in\ this\ photo \times Storage\ of\ per\ pixel \\ &= (4000 \times 4000) \times (8\,bits \times 3\,channels) \\ &= 48,000,000\,bytes \approx 48\,\text{MB}\end{aligned}$$

A2) Connectivity foreground/background

(a) 1) D_8 for foreground and D_4 for background
Number of components in foreground: 1
Number of components in background: 1

2) D_4 for foreground and D_4 for background
Number of components in foreground: 25
Number of components in background: 1

3) D_8 for foreground and D_8 for background
Number of components in foreground: 1
Number of components in background: 1

4) D_4 for foreground and D_8 for background
Number of components in foreground: 25
Number of components in background: 1

	D_4	D_8
# of components in foreground	25	1
# of components in background	1	1

(b) 1) D_8 for foreground and D_4 for background
Number of components in foreground: 1
Number of components in background: 2

2) D_4 for foreground and D_4 for background
Number of components in foreground: 11
Number of components in background: 2

3) D_8 for foreground and D_8 for background
Number of components in foreground: 1
Number of components in background: 1

4) D_4 for foreground and D_8 for background
Number of components in foreground: 11
Number of components in background: 1

	D_4	D_8
# of components in foreground	11	2
# of components in background	1	1

(c) **Thoughts**:

- **Foreground**: First, most of the lines/circles are not perfectly vertically/horizontally straight, or we can say, pixels are at the same line. So if we use D_4 to count the connected components in a line, we may wrongly separate these pixels into different parts. Furthermore, when we come across the situation where two lines are crossed with half pixels (see Figure 1), pixels inside of the purple square are counted into the same connected component. Therefore D_8 is better to be used to count the number of connected components in foreground.
- **Background**: There's no circle in Figure 2.7(a) that can divide the background into 2 parts (using D_4). But in Figure 2.7(b), we can see that the circle divides the plane into 2 different parts. And if we use D_8 to calculate connected components, the area inside of the circle would be connected with the outside one. So we usually choose D_4 to count the number of connected components in background.

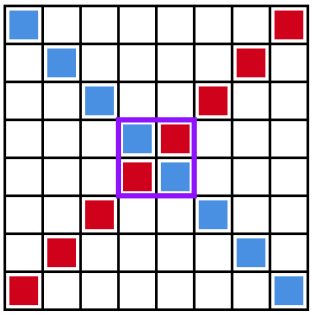

Figure 1: Two lines cross with each other

A3) Histogram equalization

There are mainly two reasons that would cause this problem:

1) No values are mapped into some particular values of the new equalized histogram (the black dots in Figure 2(a)).
2) Different values are mapped into the same value of the new equalized histogram (the white dot in Figure 2(b)).

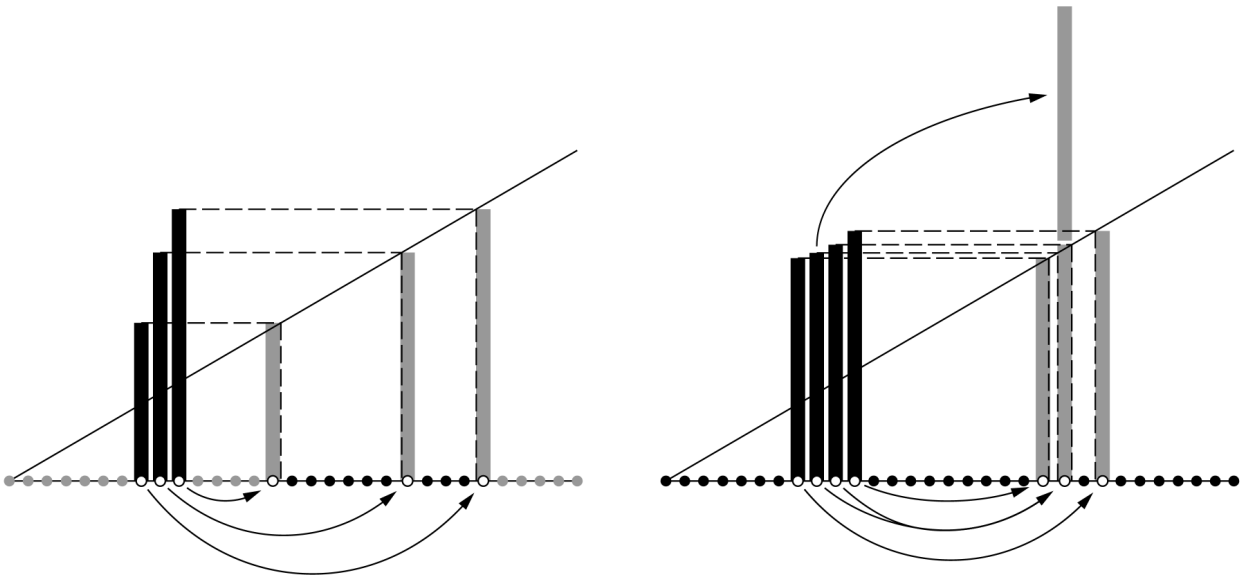

Figure 2: (a) no values are mapped into the specific value
(b) different values are mapped into the same value

Therefore, in the equalized histogram/pdf, we can see that there would be some peak values and also with some empty values.

(Image Reference: `http://geomatica.como.polimi.it/corsi/rs_ia/02_Histogram_manipulation.pdf`)

Part B - Practical Programming Assignments

B1) Compute a Histogram and CDF

Write code that reads a 2D image as input and returns a 1D array of the relative frequencies of occurrence of greylevels in your image. Provide a choice for quantizing a binning of the greylevels into n quantized bins between 0 and the maximum value (please remember that for an 8bit image, this is the range 0 … 255 for the range 0 … L-1).

- Calculate the histogram of an image of your choice, please note that a color image first needs to be converted into black-and-white.
- Normalize the histogram by the image size to present a probability density function (pdf), plot the pdf.
- Calculate the cumulative distribution function CDF from your pdf and plot the function.
- Creatively experiment with a second image that may show different structures.
- Write a short report that shows the original images, and the corresponding pdf and CDF plots. Provide a short discussion if the shape of the histograms that may reflect some of the visible properties of the image, and discuss differences between results from the two images.

B2) Histogram Equalization

Use the histogram code as developed above, and provide an additional function for histogram equalization.

- Follow instructions as in the book and course notes to calculate the histogram, pdf, cdf and then a binning of the frequency axis into n bins that determines the mapping of intensities to form a uniform distribution.
- Apply your histogram equalization code to the images used before. Calculate and plot the new histogram after equalization.
- Add an additional section to the report by showing images, pdf and cdf before/after equalization. Briefly discuss what you see in the histogram equalized images and the corresponding plots of pdf's and cdf's.

B3) Histogram Matching

Following the course notes, develop code that maps intensity values of a preferably bad image into intensity distribution of a good looking image.

- Select an image with somewhat poor contrast or visibility of structures. Select a second image which looks good.
- Calculate histograms, pdf's, cdf's of both images. Follow course instructions to map the intensity distributions of the first image into those of the second image (histogram matching).
- Add a section to the report that shows original images and plots of pdf's and cdf's. Then show the results of the adjusted first image, and its pdf and cdf's.
- Provide a short discussion of what you see and if the procedure resulted in the anticipated result.

Report

Before starting our project, we have to set up environment: I choose Python 3 and OpenCV-Python. You can use Jupyter or Google Colab to open my `.ipynb` file. I've also provided an online version (https://goo.gl/Bp3yYw) of this assignment for running.

B1) Compute a Histogram and CDF

In this part, I picked up two images.

1) The Pork Image (Algorithm Part):

It's quite easy for us to show the histogram, just aggregating all of the values from 0 to 255($L-1$). Then we can try to calculate the corresponding PDF and CDF. And here's the simple python code to generate histogram, PDF and CDF.

```
import cv2
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np

path = "CV-Assignment-1-Dataset/images/Pork/Original.jpg"

L = 2**8

def show_image(image):
  plt.axis('off')
  plt.imshow(image)
  plt.show()

def show_gray_image(image):
  plt.axis('off')
  plt.imshow(image, cmap = 'gray')
  plt.show()

def get_hist(image):
  hist = [0] * L

  for index, pixel in np.ndenumerate(image):
    hist[pixel] += 1

  return hist

def get_pdf(image):
  hist = [0] * L
  hist = get_hist(image)

  pdf = [0] * L
  height = 0
  width = 0
```

```
35    size = 0
36    (height, width) = image.shape[:2]
37    size = height * width
38
39    for i in range(0, L):
40      pdf[i] = hist[i] / size
41
42    return pdf
43
44  def get_cdf(image):
45    pdf = [0] * L
46    pdf = get_pdf(image)
47    cdf = [0] * L
48
49    for i in range(0, L):
50      cdf[i] = cdf[i - 1] + pdf[i]
51
52    return cdf
53
54  def show_hist(image):
55    plt.plot(get_hist(image), color = 'r')
56    plt.xlim([0, L])
57    plt.ylim(ymin=0)
58    plt.title("Histogram of Image")
59    plt.show()
60
61  def show_hist_bins(image, bins=L):
62    data = []
63    if bins > L:
64      bins = L
65    for index, pixel in np.ndenumerate(image):
66      data.append(pixel)
67    plt.hist(data, bins=bins, range=[0, L-1])
68    plt.title("Histogram of Image with " + str(bins) + " bins")
69    plt.show()
70
71  def show_pdf(image):
72    plt.plot(get_pdf(image), color = 'g')
73    plt.xlim([0, L])
74    plt.ylim(ymin=0)
75    plt.title("Probability Density Function of Image")
76    plt.show()
77
78  def show_cdf(image):
79    plt.plot(get_cdf(image), color = 'b')
80    plt.ylim(ymin=0, ymax=1.1)
81    plt.xlim([0, L - 1])
82    plt.title("Cumulative Distribution Function of Image")
83    plt.show()
```

```
84
85  image = cv2.imread(path)
86  gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
87
88  # # Original Image
89  # show_image(image)
90
91  # Gray Image
92  show_gray_image(gray_image)
93
94  # Histogram
95  show_hist(gray_image)
96
97  show_hist_bins(gray_image, 256)
98  show_hist_bins(gray_image, 64)
99
100 # PDF
101 show_pdf(gray_image)
102
103 # CDF
104 show_cdf(gray_image)
```

Figure 3: The original image

Figure 4: The gray-scaled image

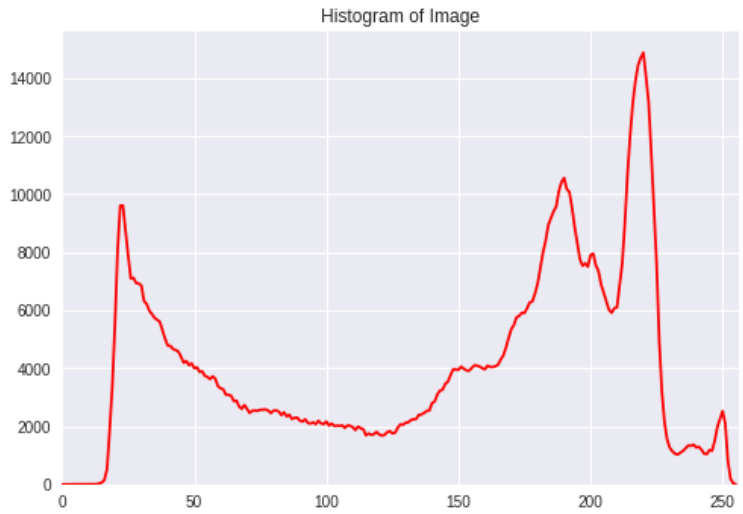

Figure 5: Histogram (using `matplotlib.plot`) of the gray-scaled image

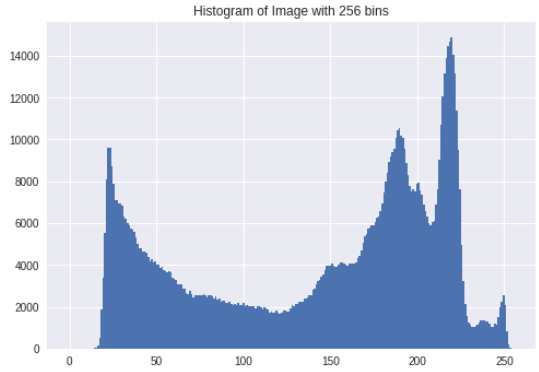

Figure 6: Histogram (using `matplotlib.hist`) of the gray-scaled image, # of bins = 256 (L)

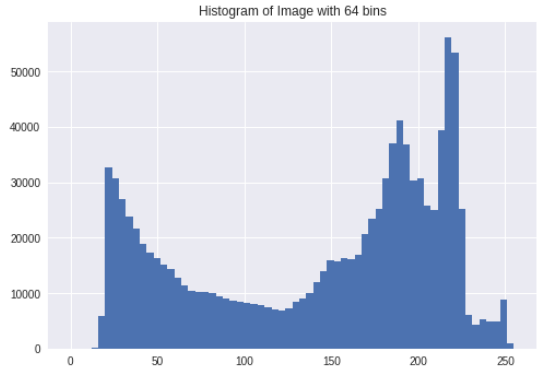

Figure 7: Histogram (using `matplotlib.hist`) of the gray-scaled image, # of bins = 64 ($\frac{L}{4}$)

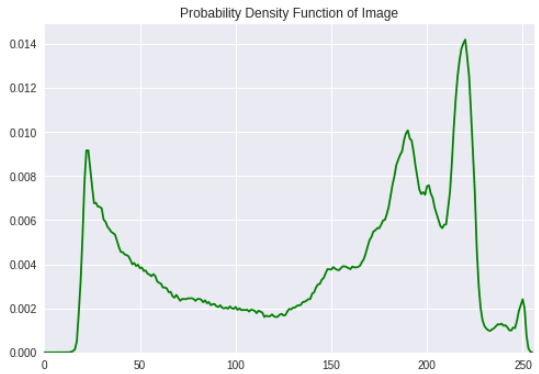

Figure 8: Probability Density Function (PDF) of the gray-scaled image

Figure 9: Cumulative Distribution Function (CDF) of the gray-scaled image

2) The Mountain Image from OpenCV Website (Comparison part):

I chose the Figure 10 and 12 to make a comparison on histogram. We can see that although they are different in pixels, they still share the same histogram, pdf and cdf. So histogram, pdf, as well as cdf are based on the pixels, not the position.

Figure 10: The original image

Figure 11: The gray-scaled image

Figure 12: The clipped image

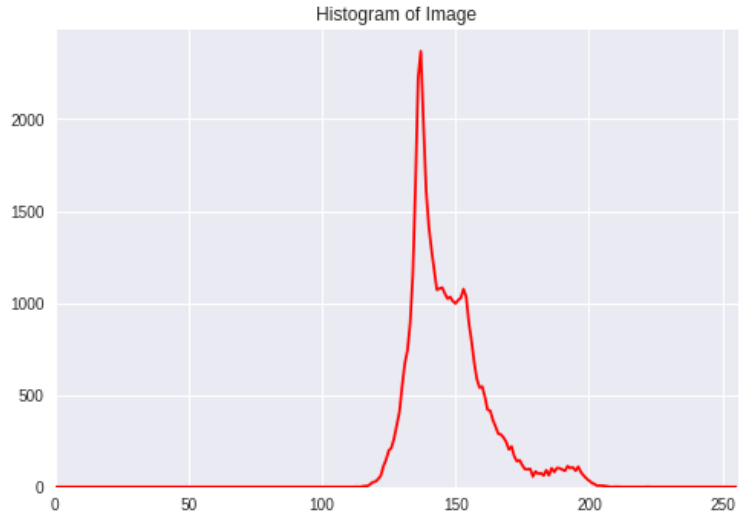

Figure 13: Histogram (using `matplotlib.plot`) of the gray-scaled image

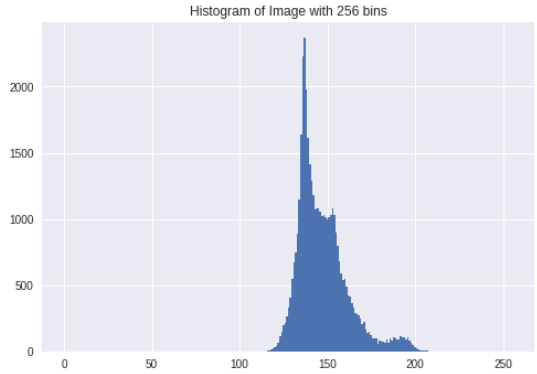

Figure 14: Histogram (using `matplotlib.hist`) of the gray-scaled image, # of bins = 256 (L)

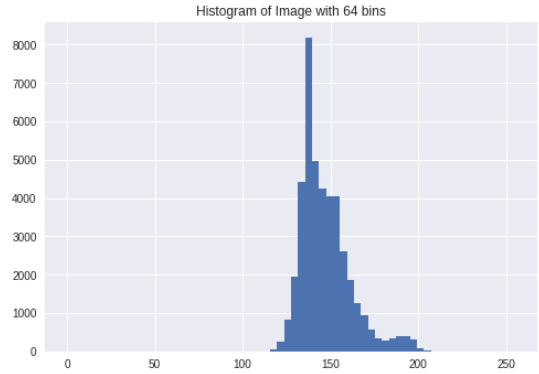

Figure 15: Histogram (using `matplotlib.hist`) of the gray-scaled image, # of bins = 64 ($\frac{L}{4}$)

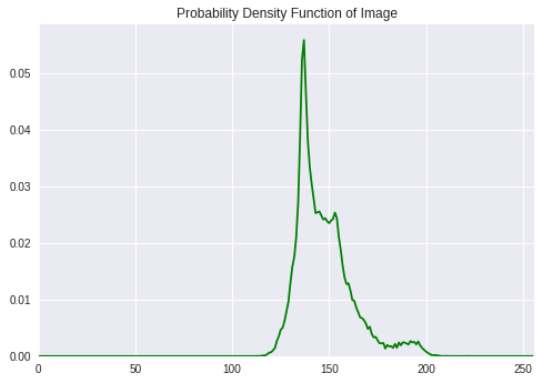

Figure 16: Probability Density Function (PDF) of the gray-scaled image

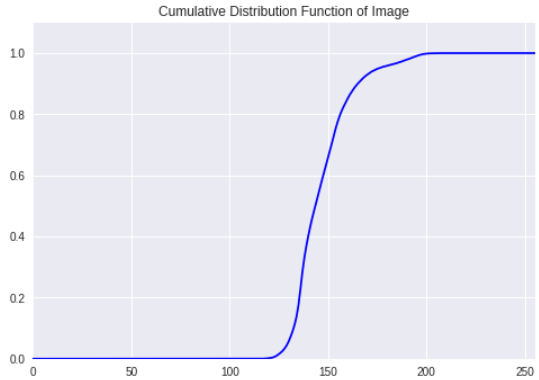

Figure 17: Cumulative Distribution Function (CDF) of the gray-scaled image

B2) Histogram Equalization

I still used Figure 10 to continue this part. The algorithm of histogram equalization and matching are almost the same. We can iterate all of the possible values of pixels to find $CDF_1(x) = CDF_2(x')$, then we replace value x with x'. In equalization part, $CDF_2(x) = \frac{1}{L}x$.

We could see that, after equalization, the value distribution of PDF is approaching equally. Also, the CDF could somehow be represented as $CDF(x) = \frac{1}{L}x$. But we can find that there are some stairs in the plot of CDF, because some of the values could not find the relative/mapping values (which is discussed at Theory Part A2).

The original gray-scaled image, after histogram equalization, shows more details on the light part. The brightness of the dark part comes down.

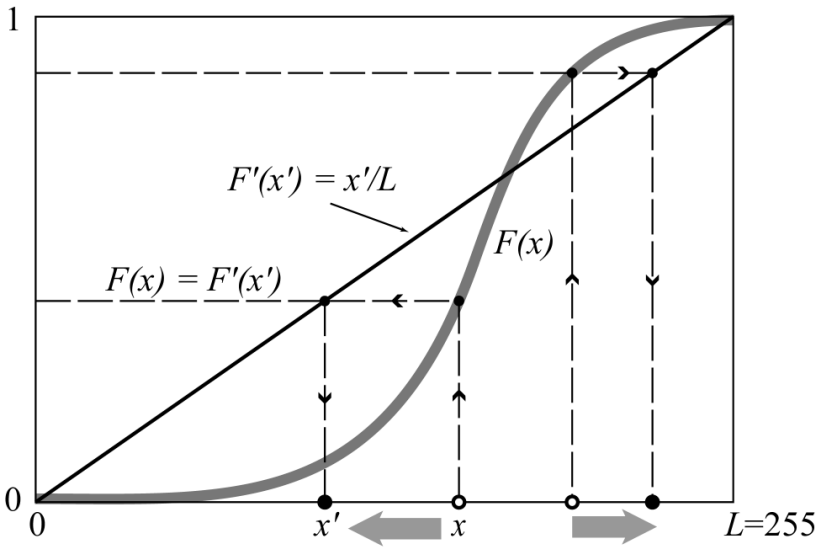

Figure 18: Histogram Equalization Algorithm

```
mapping = [0] * L

cdf = get_cdf(gray_image)

for i in range(0, L):
  mapping[i] = int(round(cdf[i] * L))

for index, pixel in np.ndenumerate(gray_image):
    gray_image[index] = mapping[pixel]

# Image
show_gray_image(gray_image)

# Histogram
show_hist(gray_image)

show_hist_bins(gray_image, 256)
show_hist_bins(gray_image, 64)
show_hist_bins(gray_image, 32)
show_hist_bins(gray_image, 16)

# PDF
show_pdf(gray_image)

# CDF
show_cdf(gray_image)
```

Figure 19: The original image

Figure 20: The gray-scaled image

Figure 21: The equalized image

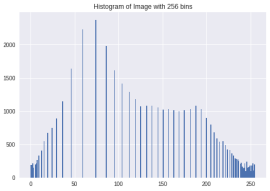

Figure 22: Histogram, # of bins = 256 (L)

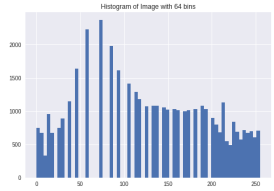

Figure 23: Histogram, # of bins = 64 ($\frac{L}{4}$)

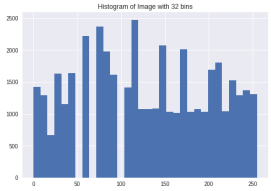

Figure 24: Histogram, # of bins = 32 ($\frac{L}{8}$)

Figure 25: Histogram, # of bins = 16 ($\frac{L}{16}$)

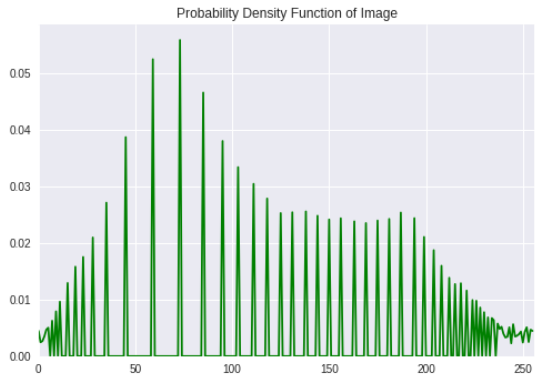

Figure 26: Probability Density Function (PDF) of the equalized gray-scaled image

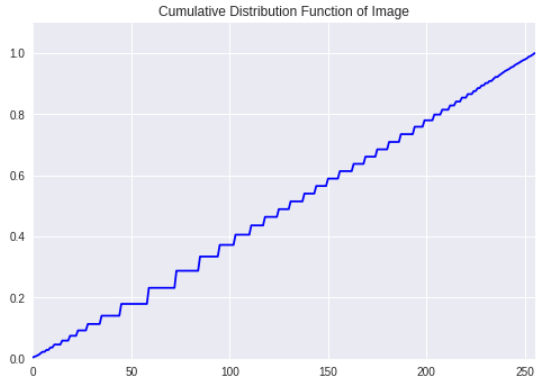

Figure 27: Cumulative Distribution Function (CDF) of the equalized gray-scaled image

B3) Histogram Matching

In this part, I picked up an anime photo as the original image. The colored image looks too bright and I use the Pork as the matching one. The algorithm of histogram matching is same as histogram equalization. We can iterate all of the possible values of pixels to find $CDF_1(x) = CDF_2(x')$, then we replace value x with x'. In this part, $CDF_2(x) = \frac{1}{L}x$.

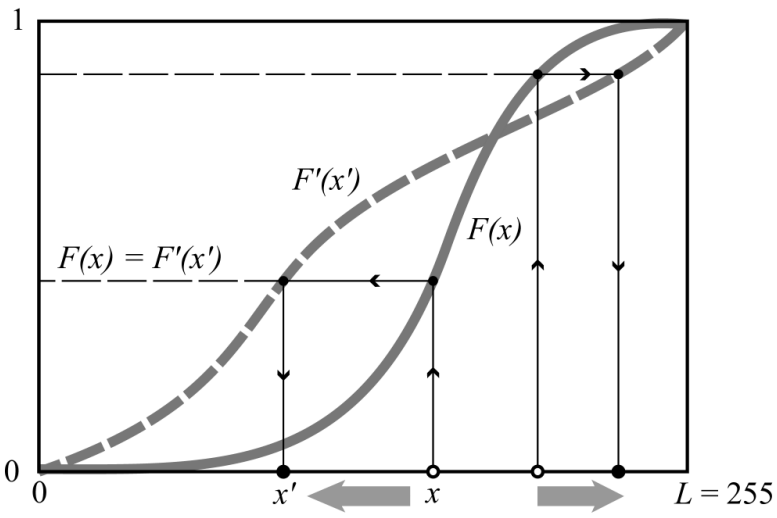

Figure 28: Histogram Matching Algorithm

We could see that after histogram matching, the image contrast was balanced and we got the really nice photo.

Figure 5, 8, 9 are the histogram, pdf and cdf of the Pork image. Figure 34, 35, 36 are the histogram, pdf and cdf of the anime image before matching. And Figure 37, 38, 39 are the histogram, pdf and cdf of the anime image after matching.

Figure 29: The original Pork image

Figure 30: The gray-scaled Pork image

Figure 31: The original image

Figure 32: The gray-scaled image

Figure 33: The matched image

Here's the code of histogram matching.

```
import math

path1 = "CV-Assignment-1-Dataset/chino.png"
path2 = "CV-Assignment-1-Dataset/images/Pork/Original.jpg"

image1 = cv2.imread(path1)
gray_image1 = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)

image2 = cv2.imread(path2)
gray_image2 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)

pdf1 = get_pdf(gray_image1)
pdf2 = get_pdf(gray_image2)

cdf1 = get_cdf(gray_image1)
cdf2 = get_cdf(gray_image2)

mapping = [0] * L

for i in range(0, L):
```

```
21      delta = 1
22      for j in range(0, L):
23        if math.fabs(cdf1[i] - cdf2[j]) < delta:
24          delta = abs(cdf1[i] - cdf2[j])
25          mapping[i] = j
26
27  for index, pixel in np.ndenumerate(gray_image1):
28      gray_image1[index] = mapping[pixel]
29
30  show_gray_image(gray_image1)
31
32  show_hist(gray_image1)
33
34  show_hist_bins(gray_image, 256)
35  show_hist_bins(gray_image, 64)
36  show_hist_bins(gray_image, 32)
37
38  show_pdf(gray_image1)
39
40  show_cdf(gray_image1)
```

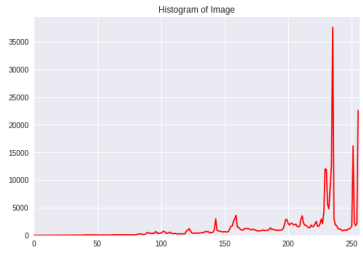

Figure 34: Histogram (before)

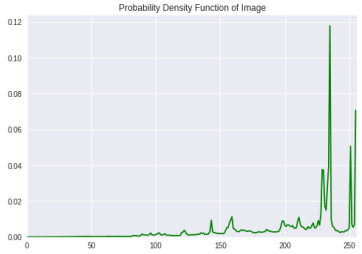

Figure 35: PDF (before)

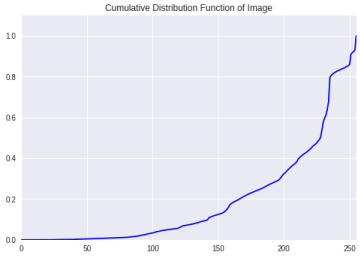

Figure 36: CDF (before)

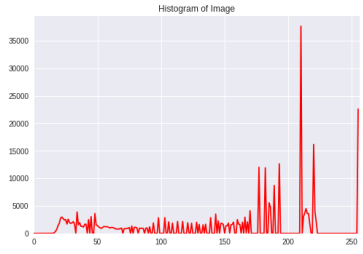

Figure 37: Histogram (after)

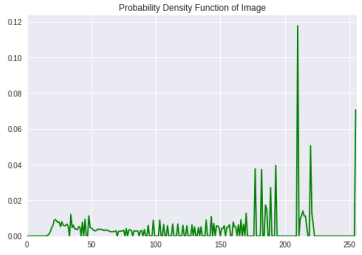

Figure 38: PDF (after)

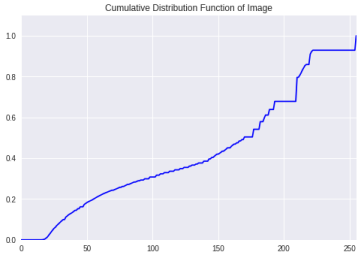

Figure 39: CDF (after)