ELE510 Image Processing with robot vision: LAB, Exercise 3, Histogram and point transformations

Purpose: *To learn about the image histogram, histogram equalization and image noise.*

The theory for this exercise can be found in chapter 3 of the text book [1]. Supplementary information can found in chapter 1, 2 and 3 in the compendium [2]. See also the following documentations for help:

- OpenCV
- numpy
- matplotlib

IMPORTANT: Read the text carefully before starting the work. In many cases it is necessary to do some preparations before you start the work on the computer. Read necessary theory and answer the theoretical part frst. The theoretical and experimental part should be solved individually. The notebook must be approved by the lecturer or his assistant.

Approval:

The current notebook should be submitted on CANVAS as a single pdf file.

To export the notebook in a pdf format, goes to File -> Download as -> PDF via LaTeX (.pdf).

Note regarding the notebook: The theoretical questions can be answered directly on the notebook using a *Markdown* cell and LaTex commands (if relevant). In alternative, you can attach a scan (or an image) of the answer directly in the cell.

Possible ways to insert an image in the markdown cell:

```
![image name]("image_path")
```

```
<img src="image_path" alt="Alt text" title="Title text" />
```

Under you will find parts of the solution that is already programmed.

You have to fill out code everywhere it is indicated with `...`

The code section under `######## a)` is answering subproblem a) etc.

Problem 1

The histogram for an image of a **black** and **white** football and **grey** background is $[0, 520, 920, 490, 30, 40, 5910, 24040, 6050, 80, 20, 80, 440, 960, 420, 0]$, where 16 gray levels are used.

The diameter of the football is $d = 230\text{mm}$.

Use these information to find the pixel size, $\Delta x = \Delta y$.

Describe the steps to arrive to the solution.

▶ **Click here for an optional hint**

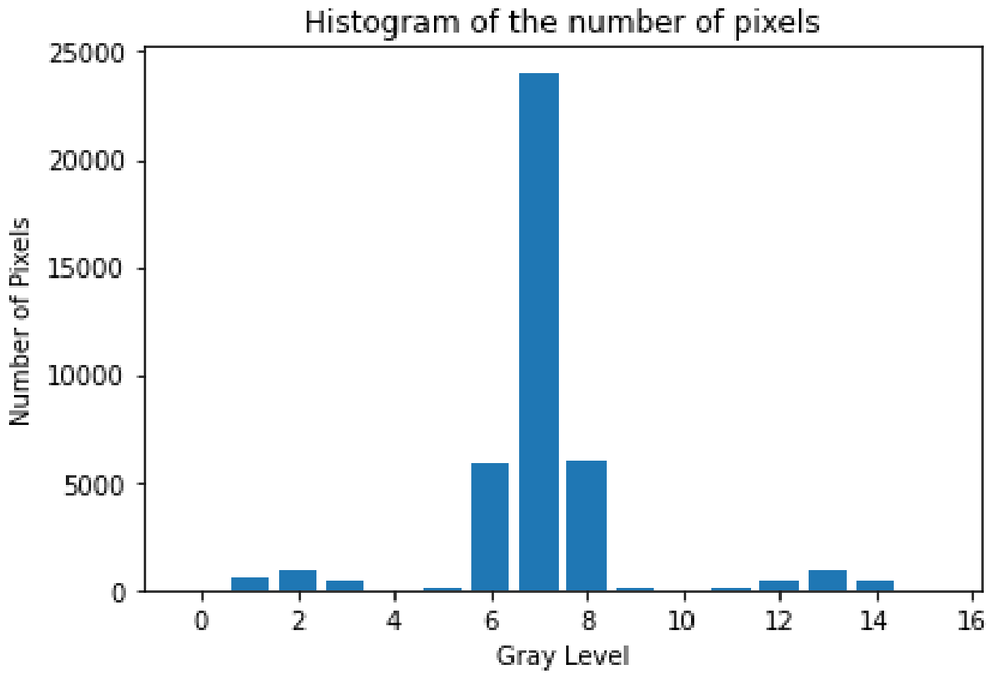
In [ ]:
```
import matplotlib.pyplot as plt
import math
```

In [ ]:
```
d = 230
A = round(math.pi * (d/2)**2, 2)
print("A = ", A, ", d = ", d)
```

```
A =  41547.56 , d =  230
```

In [ ]:
```
histogram = [0,520,920,490,30,40,5910,24040,6050,80,20,80,440,960,420,0]
gray_levels = 16
plt.bar(range(gray_levels), histogram)
plt.xlabel('Gray Level')
plt.ylabel('Number of Pixels')
plt.title('Histogram of the number of pixels')
```

Out[ ]:
```
Text(0.5, 1.0, 'Histogram of the number of pixels')
```



In [ ]:
```
# num_pizels is first four and last four gray levels
num_pixels = sum(histogram[:4] + histogram[-4:])
num_pixels
```

Out[ ]:
```
3750
```

To find the pixel size we need to calculate the following:

- d = 230mm
- number_of_pixels = 3750
- $A = \pi \cdot (d/2)^2 = (number\_of\_pixels \cdot area\_single\_pixel)$
- $\Delta x = \sqrt{\frac{A}{number\_of\_pixels}}$

In [ ]:
```
# Pixel size then becomes
pixel_size = round(math.sqrt(A / num_pixels), 2)
pixel_size
```

Out[ ]:
```
3.33
```

The solution of problem 1 should be:

```
Δx = Δy = 3.33mm or, if you have taken different level of gray into consideration, Δx = Δy = 3.22mm
```

Problem 2

For images, such as `./images/christmas.png`, some processing is normally desired to improve the contrast. The simplest approach for doing this is called histogram stretching. For a given image, where the used pixel values range from $g_{\min}$ to $g_{\max}$ we can spread these so they cover the entire $[0, G-1]$ range. The formula for histogram stretching is given by:

$$g_{\text{new}} = \left\lfloor \frac{g_{\text{old}} - g_{\min}}{g_{\max} - g_{\min}} G + 0.5 \right\rfloor$$

Where g_{old} is an old pixel value, and g_{new} a new pixel value.

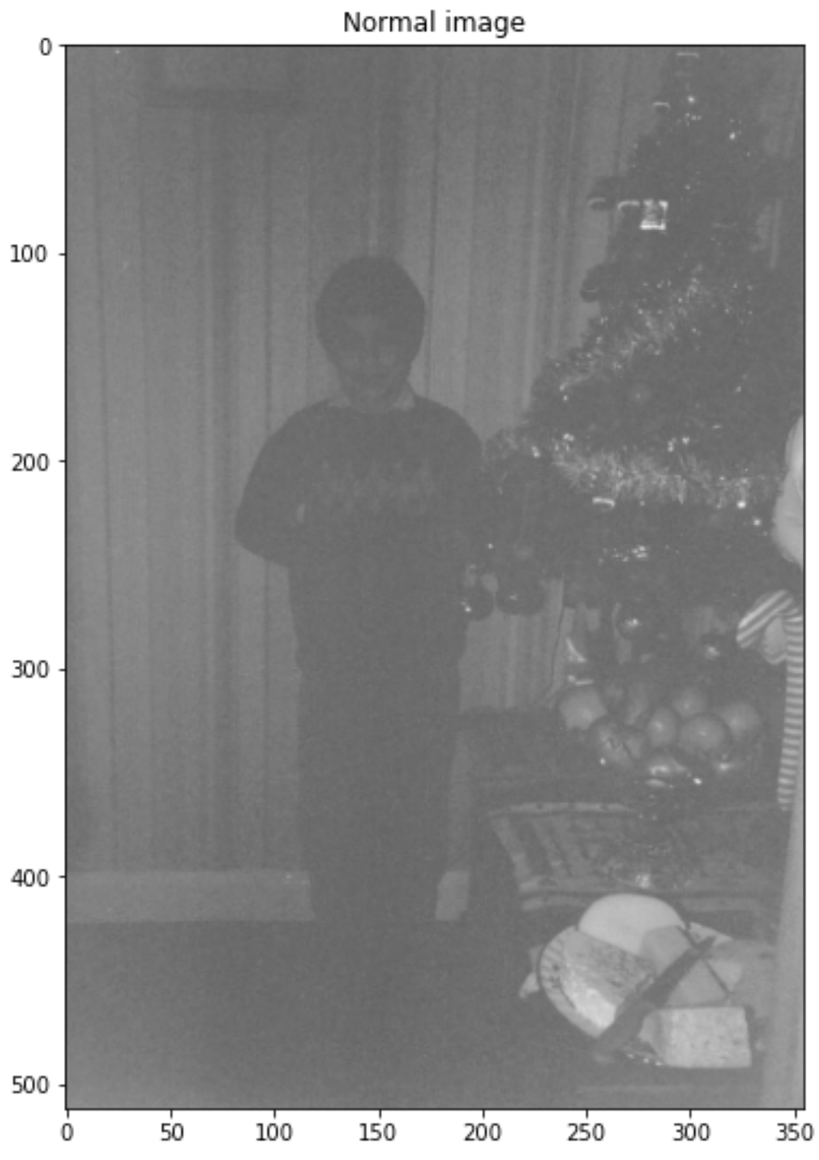
a) Make a small Python function, taking an image as input, perform histogram stretching using the previous equation, and giving the increased contrast image as output. Use an 8-bit grayscale range (G=255). Show and explain the result using `./images/christmas.png` as an example.

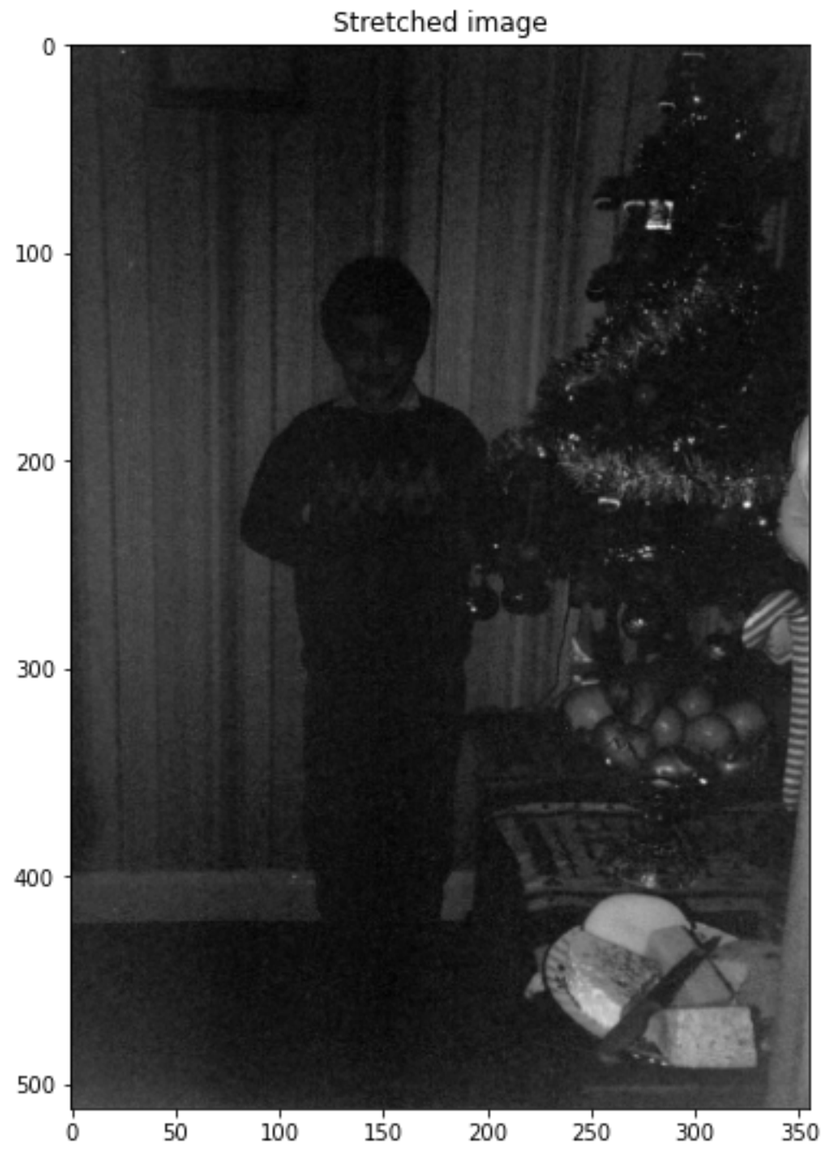
In [ ]:
```
# Import the packages
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
"""
Function that takes in input an image and return the same image stretched.
"""
def histogram_stretch(img):
    G, g_min, g_max = 255, np.min(img), np.max(img)
    return  ((img - g_min) / (g_max - g_min) * G + 0.5).astype(np.uint8)
```

In [ ]:
```
# Read the image and convert it to RGB channels
img = cv2.imread("./images/christmas.png")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

```
# Use the function to improve the contrast of the image
img_stretch = histogram_stretch(img)

plt.figure(figsize=(20,20))
plt.subplot(221)
plt.imshow(img)
plt.title('Normal image')
plt.subplot(222)
plt.imshow(img_stretch)
plt.title('Stretched image')
plt.show()
```





Problem 3

In this experiment we use **four** images. The two first are gray level images, `./images/pout.jpg` and `./images/tire.jpg`. The other two are colour images captured with a standard digital camera. We want to study image enhancement with histogram equalization.

We simplify by using only gray level images. Therefore, the colour images are first read to gray level; a grey level image can be imported using the flag (`cv2.IMREAD_GRAYSCALE`). The colour images are `./images/waterfall2.jpg` and `./images/restaurantSpain.jpg`, available from CANVAS.

▶ **Click here for optional hints**

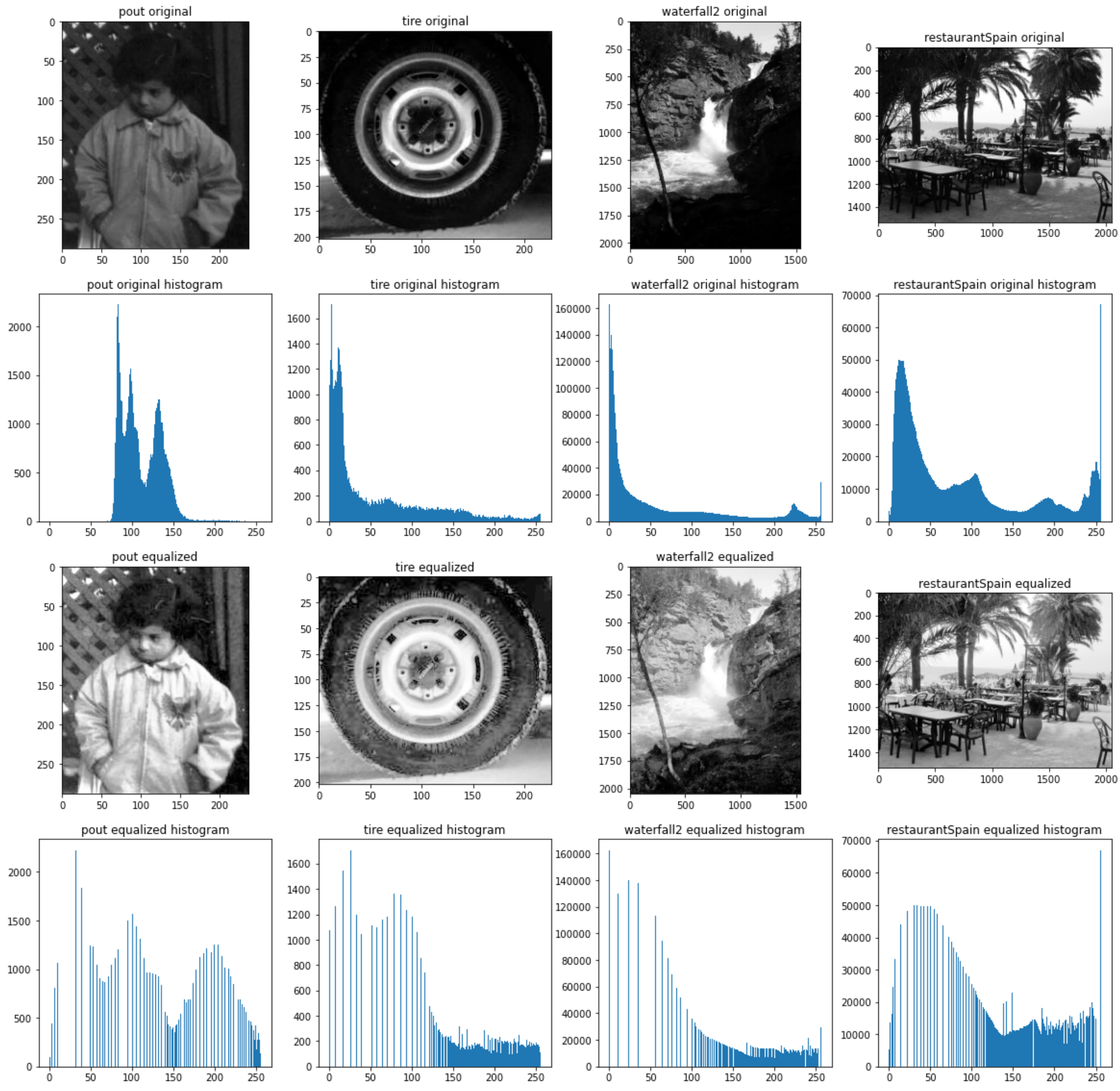
a) Use Python and find the histograms for the images.

In [ ]:
```
img_path = "./images/"
Images = ["pout", "tire", "waterfall2", "restaurantSpain"]
gral_lvl_images = ('pout', 'tire')
colour_images = ('waterfall2', 'restaurantSpain')
```

b) Perform histogram equalization of these images and find the new histograms.

In [ ]:
```
def Histogram_equalization(images):
    fig, axes = plt.subplots(4, len(images), figsize=(20,20))

    for i, image in enumerate(images):
        image_path = img_path + image + ".jpg"
        image_data = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE) if image in gral_lvl_images else cv2.imread(image_path
        image_equalized = cv2.equalizeHist(image_data)

        axes[0, i].imshow(image_data, cmap='gray')
        axes[0, i].set_title(image + " original")

        axes[1, i].hist(image_data.ravel(), 256, [0, 256])
        axes[1, i].set_title(image + " original histogram")

        axes[2, i].imshow(image_equalized, cmap='gray')
        axes[2, i].set_title(image + " equalized")
```

```
        axes[3, i].hist(image_equalized.ravel(), 256, [0, 256])
        axes[3, i].set_title(image + " equalized histogram")

    plt.show()

Histogram_equalization(Images)
```



c) Discuss the results. Are the results as expected?

Looking at the different images and their original and equalized histograms,
we see that they are different. This is due to the fact that the histogram is equalized,
which increases the contrast by spreading the pixel values across the range of levels.
This was as expected because in the original image the pixels are concentrated in a region or area, while
the equalized images have spread the pixels which explains the increase of contrast.

d) Explain why the discrete histogram equalization usually do not give a completely flat histogram.

Due to the fact that discrete histogram equalization produces an approximated flat outbut because of the discretization of the histogram,
the histogram is not completely flat. This is because it in eccense is a approximation of continous probability distribution function, and will therefore not produce new intensity levels when equalizing the histogram.

Problem 4

Noise is a common problem in digital images. In this problem we want to study estimation of camera noise. We have a set of K images. The only difference between the images is the noise value at each pixel.

Assume that the noise is additive and uncorrelated with the gray values of the image such that for each image point we have $g_k = f + \eta_k,\ k = 1, 2, \cdots, K$, where g_k is image number k with noise η_k.

The image without noise, f, is unchanged (here we have not shown the indexes (x, y)). The mean image is given by the average value:

$$\overline{g(x,y)} = \frac{1}{K}\sum_{k=1}^{K} g_k(x,y). \tag{1}$$

Then it can be shown that

$$E\{\overline{g(x,y)}\} = f(x,y) \tag{2}$$

and

$$\sigma^2_{\overline{g(x,y)}} = \frac{1}{K}\,\sigma^2_{\eta(x,y)}. \tag{3}$$

a) Show how to derive these two results using the first equation and the information given in the text.

$$E(x) = \sum_x x \cdot p(x) = \mu_x$$

$$\Rightarrow E\{\overline{g(x,y)}\} = \frac{1}{K}\sum_{k=1}^{K} g_k(x,y) = \frac{K}{K} f(x,y) + \frac{1}{K}\sum_{k=1}^{K} \eta(x,y)_k \approx f(x,y)$$

$$\sigma^2 = E(x-\mu)^2 = E(x^2) - E(x)^2$$

$$\Rightarrow \sigma^2_{\overline{g(x,y)}} = f(x,y)^2 + \left(\frac{1}{K}\sum_{k=1}^{K}\eta(x,y)_k\right)^2 + 2\left(f(x,y)\cdot\frac{1}{K}\cdot\sum_{k=1}^{K}\eta(x,y)_k\right) - f(x,y)^2$$

$$\Rightarrow \sigma^2_{\overline{g(x,y)}} \approx \left(\frac{1}{K}\sum_{k=1}^{K}\eta(x,y)_k\right)^2 + 2\left(f(x,y)\cdot\frac{1}{K}\cdot\sum_{k=1}^{K}\eta(x,y)_k\right)$$

$$= \frac{1}{K}\left(\sum_{k=1}^{K}\eta(x,y)_k^2 + 2\cdot f(x,y)\cdot\sum_{k=1}^{K}\eta(x,y)_k\right)$$

$$= \frac{1}{K}\,\sigma^2_{\eta(x,y)}$$

Delivery (dead line) on CANVAS: 23.09.2022 at 23:59

Contact

Course teacher

Professor Kjersti Engan, room E-431, E-mail: kjersti.engan@uis.no

Tomasetti Luca, room E-401 E-mail: luca.tomasetti@uis.no

Teaching assistant

Tomasetti Luca, room E-401 E-mail: luca.tomasetti@uis.no

Saul Fuster Navarro, room E-401 E-mail: saul.fusternavarro@uis.no

Jorge Garcia Torres Fernandez, room E-401 E-mail: jorge.garcia-torres@uis.no

References

[1] S. Birchfeld, Image Processing and Analysis. Cengage Learning, 2016.

[2] I. Austvoll, "Machine/robot vision part I," University of Stavanger, 2018. Compendium, CANVAS.