Proposed solutions for LABexc3-ELE510-2023

September 4, 2023

1 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.

```
<p>You have to fill out code everywhere it is indicated with `...`</p>
<p>The code section under `######## a)` is answering subproblem a) etc.</p>
```

[1]:
```
# Import the packages
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

1.1 Problem 1

The histogram for an image of a **black** cilinder and **white** background is $[5600, 980, 10, 0, 0, 40, 11200, 80000]$, where 8 gray levels are used.

The radius of the cilinder is $r = 85$mm and height $h = 310$mm. The cilinder is perfectly oriented to only see the lateral area, but not the top or bottom of the cilinder.

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

Describe the steps to arrive to the solution.

Proposed answer: The area of the shown cilinder in the image is:

$A = 2 \cdot r \cdot hmm^2 = 2 \cdot 85 \cdot 310 = N \cdot \Delta x^2$, where N is number of pixels in the ball.

We can find N from the histogram:

Black regions: $5600 + 980 + 10 = 6590$ pixels. , White regions: $40 + 11200 + 80000 = 91240$ pixels.

Totally $N = 6590$ pixels and therefore $\Delta x = \sqrt{(2 \cdot 85 \cdot 310)/6590}mm = 2.83mm$

1.2 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:

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

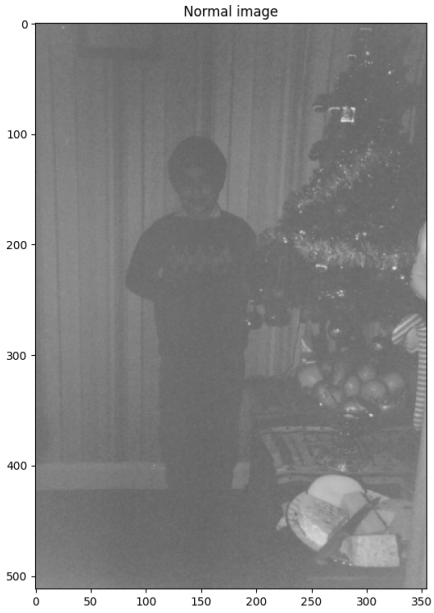
Where g_{old} is an old pixel value, and g_{new} a new pixel value.

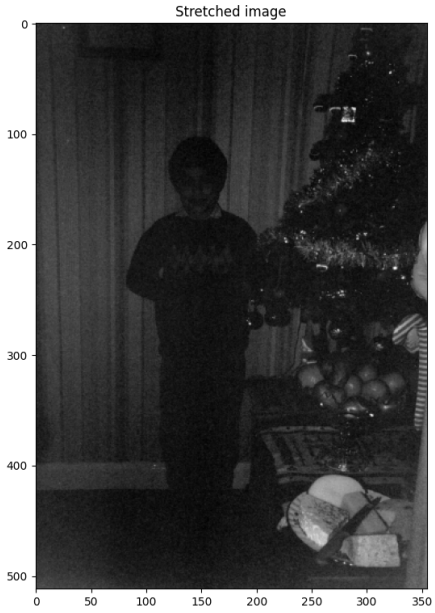
a) Make a small Python function, taking an image as input, perform histogram stretching using the previous equation, and giving the increased contrast image as output. Use an 8-bit grayscale range (G=255). Show and explain the result using `./images/christmas.png` as an example.

[2]:
```
"""
Function that takes in input an image and return the same image stretched.
"""
def histogram_stretch(img):
    G = 255 # number of grey levels
    g_min = np.min(img)
    g_max = np.max(img)
    img_stretch = np.floor((((img-g_min)/(g_max-g_min))*G) + 0.5).astype(np.
 ↪uint8)

    return img_stretch
```

[3]:
```
# Read the image and convert it to RGB channels
imagepath = os.path.join("./images/christmas.png")
```

```
img = cv2.imread(imagepath)#, cv2.IMREAD_GRAYSCALE)
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, cmap="gray")
plt.title('Normal image')
plt.subplot(222)
plt.imshow(img_stretch, cmap="gray")
plt.title('Stretched image')
plt.show()
```





1.3 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

1. Make a Python function or script that do histogram equalization cv2.equalizeHist,
   2. computes the histograms plt.hist for both the input image and the output image,

3. and displays both images with histograms in the same figure (use plt.figure and plt.subplot).

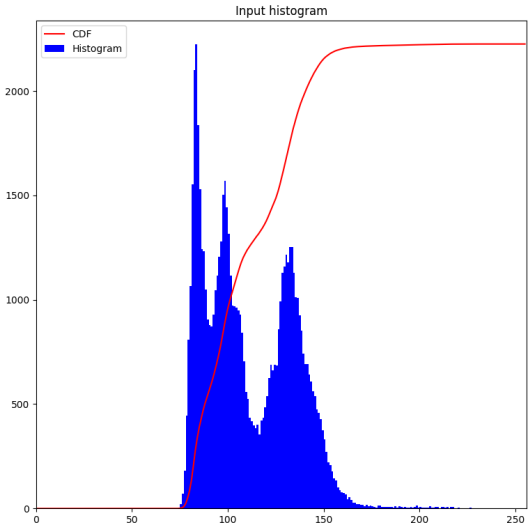
a) Use Python and find the histograms for the images. Plot the normalized cumulative distribution function (CDF) over the histogram.

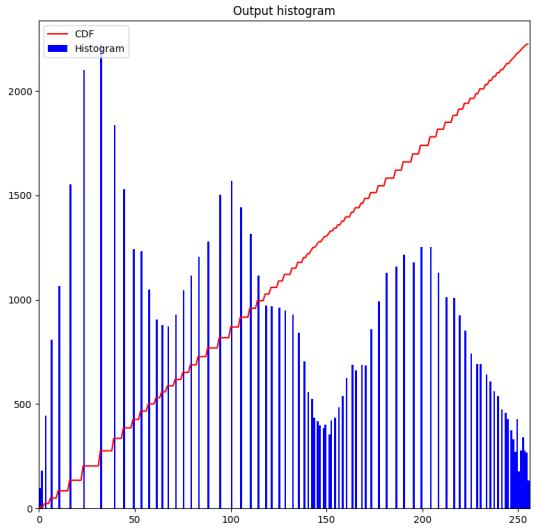
b) Perform histogram equalization of these images and find the new histograms. Plot the normalized CDF over the new histogram. Discuss the effect of the equalization over the histogram and its CDF.

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

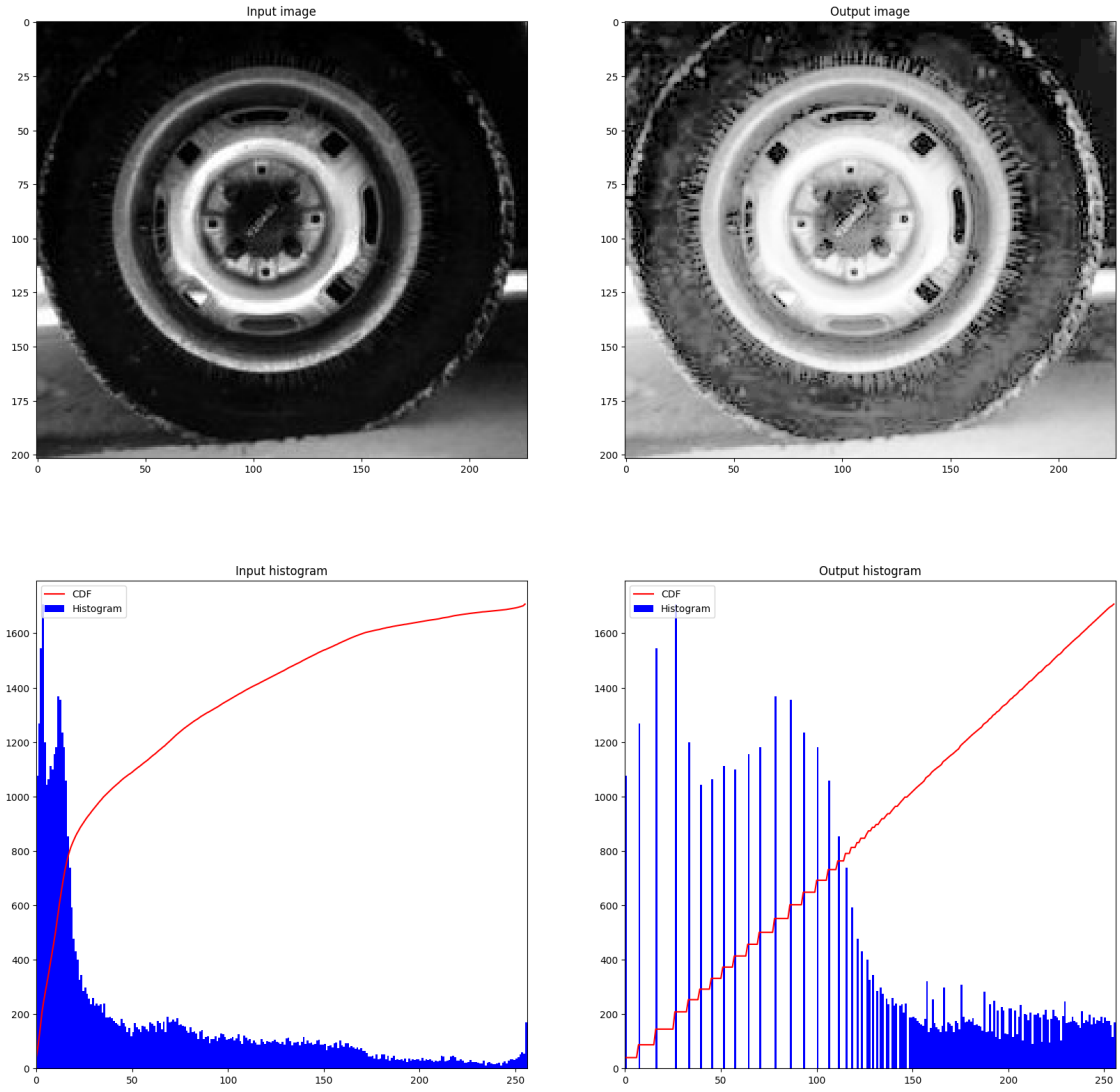
[4]:
```
#######################################################
# a) & b)
"""
Function that takes in input the image name, perform an histogram equalization␣
 ↪and display the results.
"""
def showhisteq(imgname):
    img = cv2.imread(imgname, cv2.IMREAD_GRAYSCALE) # read a grayscale image
    outimg = cv2.equalizeHist(img)

    plt.figure(figsize=(20,20))
    plt.subplot(221)
    plt.imshow(img, cmap='gray', vmin=0, vmax=255)
    plt.title('Input image')
    plt.subplot(222)
    plt.imshow(outimg, cmap='gray', vmin=0, vmax=255)
    plt.title('Output image')

    input_hist, _ = np.histogram(img.flatten(),256,[0,256])
    cdf_input = input_hist.cumsum()
    cdf_input_normalized = cdf_input * input_hist.max()/ cdf_input.max()
    output_hist, _ = np.histogram(outimg.flatten(),256,[0,256])
    cdf_output = output_hist.cumsum()
    cdf_output_normalized = cdf_output * output_hist.max()/ cdf_output.max()

    plt.subplot(223)
    plt.plot(cdf_input_normalized, color = 'r')
    plt.hist(img.flatten(),256,[0,256], color = 'b')
    plt.xlim([0,256])
    plt.legend(('CDF','Histogram'), loc = 'upper left')
    plt.title('Input histogram')
    plt.subplot(224)
    plt.plot(cdf_output_normalized, color = 'r')
    plt.hist(outimg.flatten(),256,[0,256], color = 'b')
    plt.xlim([0,256])
    plt.legend(('CDF','Histogram'), loc = 'upper left')
    plt.title('Output histogram')
    plt.show()
```

[5]:
```
imagepath = os.path.join("./images/pout.jpg")
showhisteq(imagepath)
```

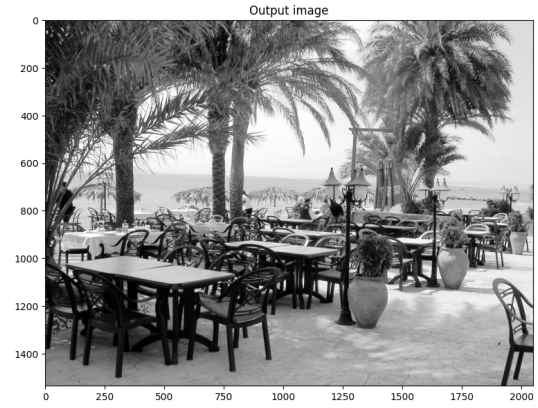


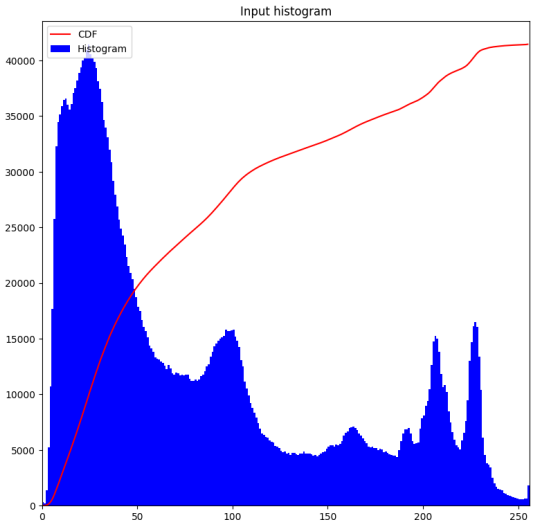


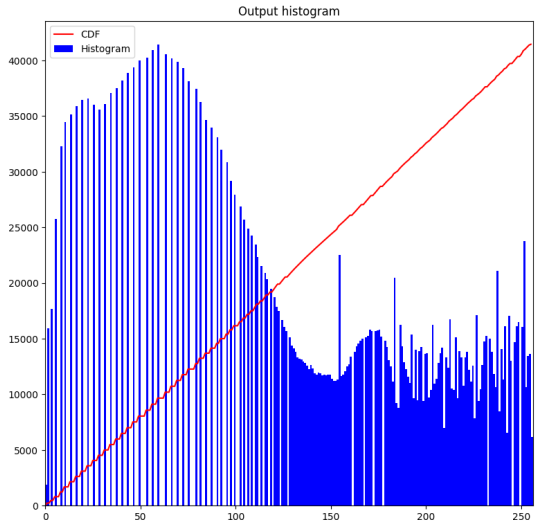
[6]:
```
imagepath = os.path.join("./images/tire.jpg")
showhisteq(imagepath)
```



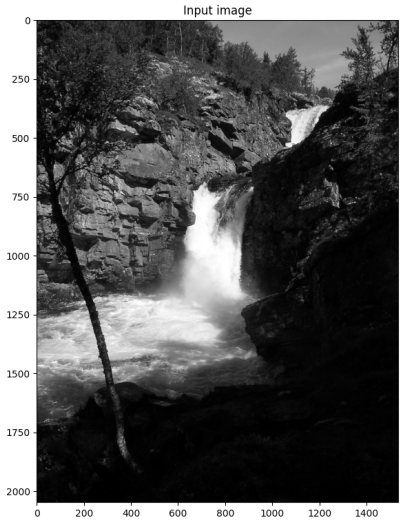
[7]:
```
imagepath = os.path.join("./images/restaurantSpain.jpg")
showhisteq(imagepath)
```

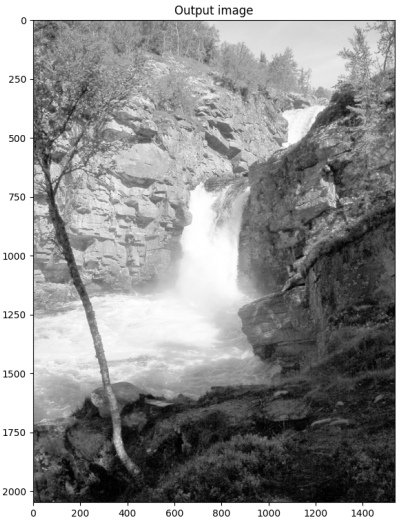


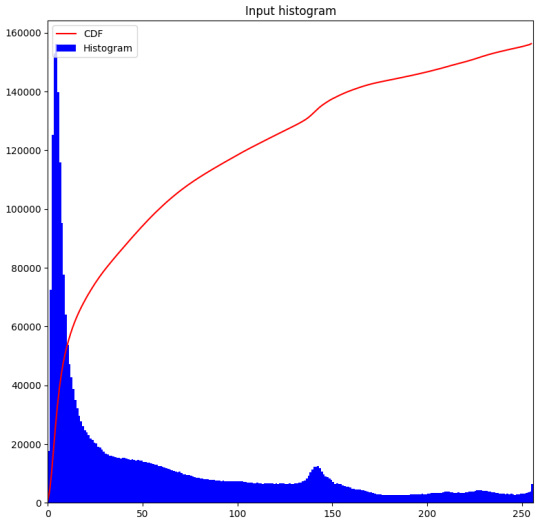


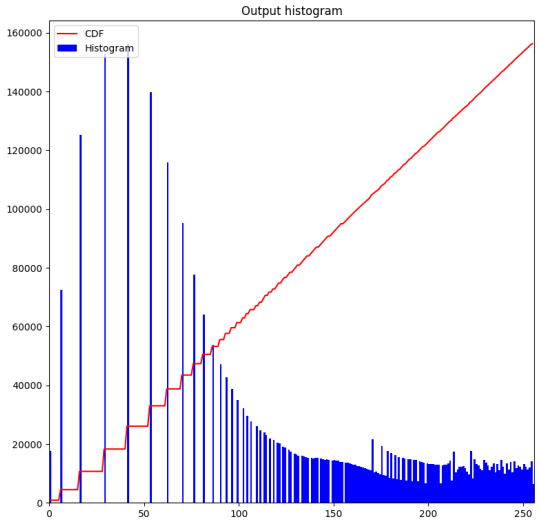


[8]:
```
imagepath = os.path.join("./images/waterfall2.jpg")
showhisteq(imagepath)
```









b) We can observe that we have achieved a more uniform distribution. While in the input images the pixel value density is heavily represented in a specific set of values, the pixel values in the equalized images is more evenly distributed. This is further reflected in the CDF, which can be aproximated by a linear function such as $y = a \cdot x$, being a the slope constant.

Proposed answers: **c)** The discrete histogram cannot be completely flat when the input image has varying number of pixels for the different grey levels. The number of pixels in each bin cannot change; it can only be moved to another position or added to another bin.

1.4 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.

Proposed answer: Equation (1) gives:

$$\bar{g} = \frac{1}{K}\sum_{k=1}^{K} g_k = \frac{1}{K}\sum_{k=1}^{K} f_k + \frac{1}{K}\sum_{k=1}^{K} \eta_k$$

Then:

$$E\{\bar{g}\} = \frac{1}{K}\sum_{k=1}^{K} \underbrace{E\{f_k\}}_{\substack{E\{f_k\}=f, \text{ since the} \\ \text{image is unchanged.}}} + \frac{1}{K}\sum_{k=1}^{K} \underbrace{E\{\eta_k\}}_{\substack{E\{\eta_k\}=0, \text{ since the noise is} \\ \text{assumed to have zero mean.}}}$$

Therefore:

$$E\{\bar{g}\} = \frac{1}{K} K f = f$$

All images are the same and the noise has zero mean.

From statistics we have that **the variance of the sum of uncorrelated random variables is the sum of the variances**.

Then, from the equation above, we get:

$$
\begin{aligned}
Var\{\bar{g}\} &= Var\{\frac{1}{K}\sum_{k=1}^{K} g_k\} \\
&= \frac{1}{K^2} Var\{\sum_{k=1}^{K} g_k\} \\
&= \frac{1}{K^2} \sum_{k=1}^{K} Var\{g_k\} \\
&= \frac{1}{K^2} \sum_{k=1}^{K} (Var\{f_k + \eta_k\})
\end{aligned}
$$

Then:

$$
Var\{\bar{g}\} = \frac{1}{K^2} \sum_{k=1}^{K} \left(\underbrace{Var\{f_k\}}_{\substack{Var\{f_k\}=0, \text{ since the} \\ \text{image is unchanged.}}} + \underbrace{Var\{\eta_k\}}_{Var\{\eta_n\}=\sigma_\eta^2} \right)
$$

$$
Var\{\bar{g}\} = \frac{1}{K^2} \left(0 + K\sigma_\eta^2\right)
$$

Therefore:

$$
Var\{\bar{g}\} = \frac{1}{K}\sigma_\eta^2
$$

1.5 Contact

1.5.1 Course teacher

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

1.5.2 Teaching assistant

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

1.6 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.