Manara university
Engineering college

Faculty of Informatics engineering

Computer vision

2025 - 2026

Lab session 4:
exp function and histogram equalization

Tasks:

1. Read images P1 or P2, in the gray scale and store it as (img) array.
2. Initialize zeros [256, 1] vectors called (idx, h1, h2, h3, h1_sum and hn).
3. Get the size of (img) array and store it in (r, c) variables.
4. Initialize zeros [r, c] array called equalized_img as the same size of (img) array.
5. Apply exponential function to the (img) and store the result in (exp_img) array using the two formulas:

   exp_image[i, j] = c * (np.exp(img[i, j]) – 1),

   where: c is a constant, (c= 1 for example)

6. Calculate the histogram of (img) and store it in a [256, 1] vector called (h1).
7. Calculate the histogram of (exp_img) and store it in a [256, 1] vector called (h2).
8. Display (img) and (exp_img) with corresponded histograms using (plt.subplots).
9. Fill (h1_sum) vector to represent the accumulator sum of the histogram (h1) (i.e. each element in h1_sum represents the corresponded value of (h1) in addition to the value of the last element of the accumulator vector (h1_sum).
10. Calculate the normalized_sum (hn) vector using the formula:

    hn = h1_sum * np.max(img)/ np.max(h1_sum)

11. Replace the gray levels of (img) array with the new calculated levels in (hn) vector and store the result in the (equalized_img) array.
12. Calculate the histogram of (img) after the equalization process and store it in (h3) vector.
13. Display the two histograms (h1 and h2) and P1 image before and after the exponential transformation in one figure using subplots and bar methods.
14. Display the two histograms (h1 and h3) and P1 image before and after the equalization process in one figure using subplots and bar methods.

Needed syntaxes:

 # casting exp_image array into uint8 one:

exp_image = np.array(exp_image, dtype = np.uint8)

 # creation of multiple sub-figures and sub-plots to display the images and histograms:

fig, axis = plt.subplots(2, 2)

axis[0][0].set_title('before log')

axis[0][0].imshow(img, cmap='gray')

…

axis[1][1].bar(idx[0], h2[0])

plt.show()

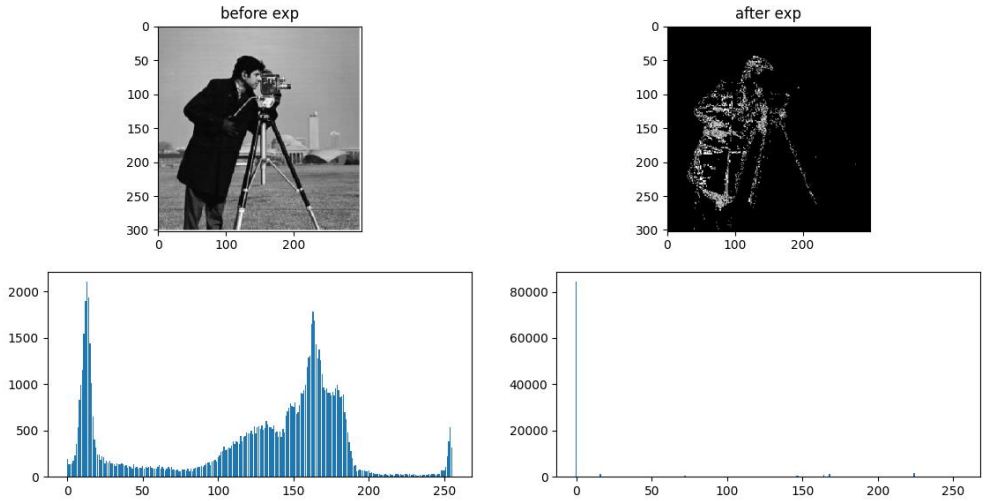
code:

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

# Read an image
path = r" \\"
img = cv2.imread(path+"cameraman.png", 0)

# Apply log transformation method
c = 1
exp_image = c * (np.exp(img)-1)

# Specify the data type so that
# float value will be converted to int
exp_image = np.array(exp_image, dtype = np.uint8)
```

```
r, c = img.shape
h1 = np.zeros((1, 256), dtype = int)
h2 = np.zeros((1, 256), dtype = int)
idx = np.zeros((1, 256), dtype = int)

for i in range(r):
    for j in range(c):
        h1[0][img[i, j]] = h1[0][img[i, j]]+1
        h2[0][exp_image[i, j]] = h2[0][exp_image[i, j]]+1

for i in range(256):
  idx[0][i] = i

# Display both images
fig, axis = plt.subplots(2, 2)
axis[0][0].set_title('before exp')
axis[0][0].imshow(img, cmap='gray')
axis[0][1].set_title('after exp')
axis[0][1].imshow(exp_image, cmap='gray')
axis[1][0].bar(idx[0], h1[0])
axis[1][1].bar(idx[0], h2[0])
plt.show()

r, c = img.shape
h3 = np.zeros((1, 256), dtype = int)
h1_sum = np.zeros((1, 256), dtype = int)
equalized_img = np.zeros((r, c), dtype=int)
hn = np.zeros((1, 256), dtype = int)
```

```
# sum of histogram
h1_sum[0][0] = h1[0][0]
for i in range( len(h1[0])-1 ):
    h1_sum[0][i+1] = h1_sum[0][i] + h1[0][i+1]

normalized_sum = np.round( h1_sum * np.max(img)/ np.max(h1_sum) )

for i in range(r):
    for j in range(c):
        equalized_img[i, j] = normalized_sum[0][img[i, j]]

for i in range(r):
    for j in range(c):
        hn[0][equalized_img[i, j]] = hn[0][equalized_img[i, j]]+1

fig, axes = plt.subplots(2, 2)
fig.suptitle('Results')
axes[0][0].bar(idx[0], h1[0])
axes[0][1].bar(idx[0], hn[0])
axes[1][0].imshow(img, cmap='gray')
axes[1][1].imshow(equalized_img, cmap='gray')
plt.show()
```

Results:



Histogram Equalization :