COMPUTER VISION  
ASSIGNMENT-3

**AMOGH GARG – 2020UCO1688**

Consider a grey level image f(x, y) of size 256x256 with 1< x, y ≤256, which has the following intensities:

**f(x,y)** = r+1 1≤x≤12andl≤ y ≤12

r 13≤x≤16,1≤ y ≤16 and   
  
 r+2 1≤x≤12,13≤ y ≤16 elsewhere

with 0 ≤r≤253.

(i) Sketch the image f(x, y) and comment on its visual appearance. Justify your answer.

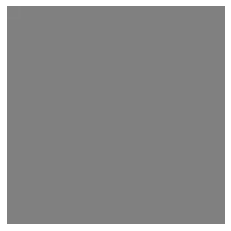
**IMPORTING LIBRARIES**

import numpy as np  
import matplotlib.pyplot as plt  
from PIL import Image as im  
import cv2

**DEFINING FUNCTIONS**

def create\_image(size, r):  
 image = np.zeros((size, size))  
 for x in range(1, size + 1):  
 for y in range(1, size + 1):  
 if 1 <= x <= 12 and 1 <= y <= 12:  
 image[x-1, y-1] = r + 1  
 elif 13 <= x <= 16 and 1 <= y <= 16:  
 image[x-1, y-1] = r + 2  
 elif 1 <= x <= 12 and 13 <= y <= 16:  
 image[x-1, y-1] = r + 2  
 else :  
 image[x-1, y-1] = r;  
 return image  
  
def display\_image(img):  
 plt.imshow(img, cmap='gray')  
 plt.axis('off')  
 plt.show()  
   
def hist\_equalize(img):  
 # Convert the input image to grayscale  
 gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
  
 # Compute the cumulative distribution function  
 hist, bins = np.histogram(gray.flatten(), 256, [0, 256])  
 cdf = hist.cumsum()  
 cdf\_normalized = cdf \* hist.max() / cdf.max()  
  
 # Use the CDF to equalize the histogram  
 cdf\_m = np.ma.masked\_equal(cdf,0)  
 cdf\_m = (cdf\_m - cdf\_m.min())\*255/(cdf\_m.max()-cdf\_m.min())  
 cdf = np.ma.filled(cdf\_m,0).astype('uint8')  
 equalized\_img = cdf[gray]  
   
 return equalized\_img  
  
def local\_hist\_equalize(img, patch\_size=16):  
 # Convert the input image to grayscale  
 gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
  
 # Create a copy of the input image  
 equalized\_img = np.zeros(gray.shape, dtype=img.dtype)  
  
 # Divide the image into patches of size patch\_size x patch\_size  
 for i in range(0, img.shape[0], patch\_size):  
 for j in range(0, img.shape[1], patch\_size):  
 patch = gray[i:i+patch\_size, j:j+patch\_size]  
  
 # Compute the cumulative distribution function for each patch  
 hist, bins = np.histogram(patch.flatten(), 256, [0, 256])  
 cdf = hist.cumsum()  
 cdf\_normalized = cdf \* hist.max() / cdf.max()  
  
 # Use the CDF to equalize the histogram for each patch  
 cdf\_m = np.ma.masked\_equal(cdf,0)  
 cdf\_m = (cdf\_m - cdf\_m.min())\*255/(cdf\_m.max()-cdf\_m.min())  
 cdf = np.ma.filled(cdf\_m,0).astype('uint8')  
 patch = cdf[patch]  
 equalized\_img[i:i+patch\_size, j:j+patch\_size] = patch  
  
 return equalized\_img

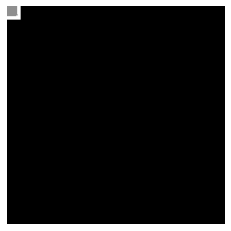
size = 256  
r = 128  
image1 = create\_image(size, r)  
image = im.fromarray(image1)  
cv2.imwrite('Img-2.jpg',image1)  
display\_image(image)



We can observe that the visual appearance of the image is same throughout because pixel intensities varies only in the small range of [r,r+2].

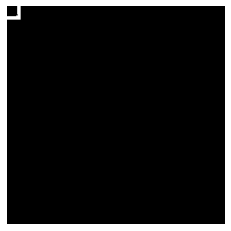
(ii) Apply global histogram equalisation on the above image. Comment on the visual appearance of the resulting equalised image.

img = cv2.imread("Img-2.jpg")  
equalized\_img = hist\_equalize(img)  
display\_image(equalized\_img)



(iii) Apply local histogram equalisation on the above image using non-overlapping image patches of size 16×16. Comment on the visual appearance of the resulting locally equalised image.

img = cv2.imread("Img-2.jpg")  
equalized\_img = local\_hist\_equalize(img)  
display\_image(equalized\_img)



(iv) Based on the above observations, which of the two types of equalisation processes would you choose for the visual improvement of the particular image? Justify your answer.

Global histogram equalization is best suited for images with low contrast, where the intensity values are not distributed evenly across the whole image. This technique transforms the intensity values to produce a uniform distribution of intensities, which results in an increase in contrast and a better visual appearance of the image.

On the other hand, local histogram equalization is best suited for images with high contrast, where some regions of the image have a high dynamic range of intensities. This technique equalizes the histogram of small regions of the image, rather than the entire image, which helps to preserve the details in the high-contrast regions while still improving the overall visual appearance of the image.

In our case, applying global histogram equalisation is a better choice as in case of global equalisation, the 16 x 16 patches are not being divided by sharp boundaries as in case of local equalisation (which decreases the overall sharpness and visual qual

# Let's apply these on random images  
contrast\_img = cv2.imread("Img-1.jpg")  
# applying global equlaization on low\_contrast\_img  
global\_eq\_img = hist\_equalize(contrast\_img)  
# applying local equalization on low\_contrast\_img  
local\_eq\_img = local\_hist\_equalize(contrast\_img)

# original\_image  
display\_image(contrast\_img)



# after applying global optimization  
display\_image(global\_eq\_img)



# after applying global optimization  
display\_image(local\_eq\_img)

