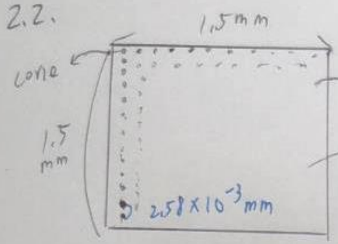
數位影像處理 DIP CH2 Homework (100 pts)

1. (P113; Problem2.2)(30)

2.2* Using the background information provided in Section 2.1, and thinking purely in geometrical terms, estimate the diameter of the smallest printed dot that the eye can discern if the page on which the dot is printed is 0.2 m away from the eyes. Assume for simplicity that the visual system ceases to detect the dot when the image of the dot on the fovea becomes smaller than the diameter of one receptor (cone) in that area of the retina. Assume further that the fovea can be modeled as a square array of dimension 1.5 mm on the side, and that the cones and spaces between the cones are distributed uniformly throughout this array.

2.2.



Cones: 581×581

due to textbook 150,000 cones per mm² at fovea.

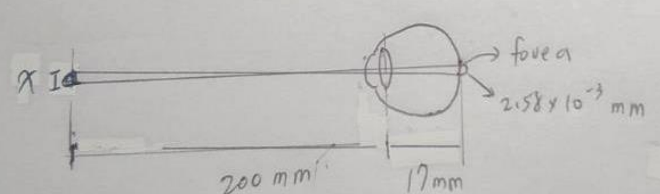
⇒ so there are $150{,}000 \times 1.5 \times 1.5 = 337500$ numbers of cones at fovea.

$337500 \cong 580 \times 580$

⇒ so there are 580 numbers of cones on one side of fovea.

space between two cones

$$\Rightarrow \frac{1.5\ mm}{580+1} = 2.58 \times 10^{-3}\ mm$$

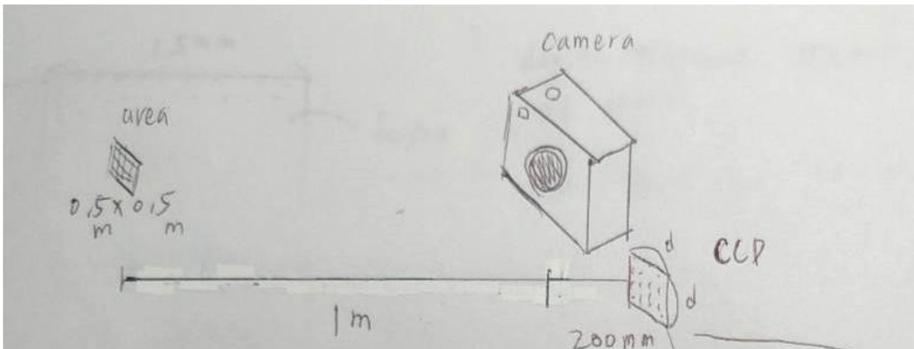


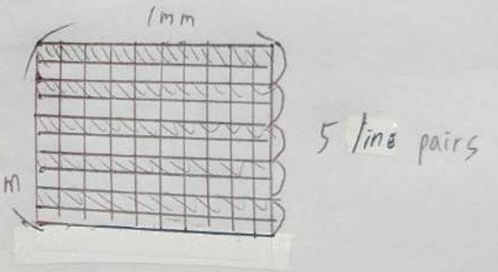
$$\frac{x}{200} = \frac{2.58 \times 10^{-3}}{17} \quad \text{(units : mm)}$$

$x = 0.03$ mm ※

2. (P114; Problem2.8)(35)

2.8* Suppose that a given automated imaging application requires a minimum resolution of 5 line pairs per mm to be able to detect features of interest in objects viewed by the camera. The distance between the focal center of the camera lens and the area to be imaged is 1 m. The area being imaged is 0.5×0.5 m. You have available a 200 mm lens, and your job is to pick an appropriate CCD imaging chip. What is the minimum number of sensing elements and square size, $d \times d$, of the CCD chip that will meet the requirements of this application? (*Hint:* Model the imaging process as in Fig. 2.3, and assume for simplicity that the imaged area is square.)





$$\frac{500}{1000} = \frac{d}{200} \quad (\text{units: mm})$$

$d = 100$ mm ※

(5x2)

$\underbrace{10 \times 100}_{\text{Sensing elements for one side}} \times 10 \times 100 = 10^6$ ※ minimum number of sensing elements in size $d \times d$. (100×100 mm mm)

3. (P117; Problem2.36)(35)

2.36 With reference to Table 2.3, provide single, composite transformation functions for performing the following operations:

(a)* Scaling and translation.

(b)* Scaling, translation, and rotation.

(c) Vertical shear, scaling, translation, and rotation.

(d) Does the order of multiplication of the individual matrices to produce a single transformations make a difference? Give an example based on a scaling/translation transformation to support your answer.

2.36

(a)

$$\overset{\text{Scaling}}{\begin{bmatrix} C_x & 0 & 0 \\ 0 & C_y & 0 \\ 0 & 0 & 1 \end{bmatrix}} \overset{\text{translation}}{\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}} = \begin{bmatrix} C_x & 0 & C_x t_x \\ 0 & C_y & C_y t_y \\ 0 & 0 & 1 \end{bmatrix} ※$$

(b)

$$\begin{bmatrix} C_x & 0 & C_x t_x \\ 0 & C_y & C_y t_y \\ 0 & 0 & 1 \end{bmatrix} \overset{\text{rotation}}{\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}} = \begin{bmatrix} C_x\cos\theta & -C_x\sin\theta & C_x t_x \\ C_y\sin\theta & C_y\cos\theta & C_y t_y \\ 0 & 0 & 1 \end{bmatrix} ※$$

(c)

$$\overset{\text{vertical shear}}{\begin{bmatrix} 1 & S_v & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}} \begin{bmatrix} C_x\cos\theta & -C_x\sin\theta & C_x t_x \\ C_y\sin\theta & C_y\cos\theta & C_y t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_x\cos\theta + S_v C_y\sin\theta & -C_x\sin\theta + S_v C_y\cos\theta & C_x t_x + S_v C_y t_y \\ C_y\sin\theta & C_y\cos\theta & C_y t_y \\ 0 & 0 & 1 \end{bmatrix} ※$$

(d) yes, $\because$ 矩陣乘法不具交換律 ※

$$\overset{\text{translation}}{\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}} \overset{\text{then}}{} \overset{\text{Scaling}}{\begin{bmatrix} C_x & 0 & 0 \\ 0 & C_y & 0 \\ 0 & 0 & 1 \end{bmatrix}} = \begin{bmatrix} C_x & 0 & t_x \\ 0 & C_y & t_y \\ 0 & 0 & 1 \end{bmatrix} \neq \overset{\text{Scaling then translation}}{\begin{bmatrix} C_x & 0 & C_x t_x \\ 0 & C_y & C_y t_y \\ 0 & 0 & 1 \end{bmatrix}}$$

數位影像處理 DIP Chapter3 Homework (100 pts)

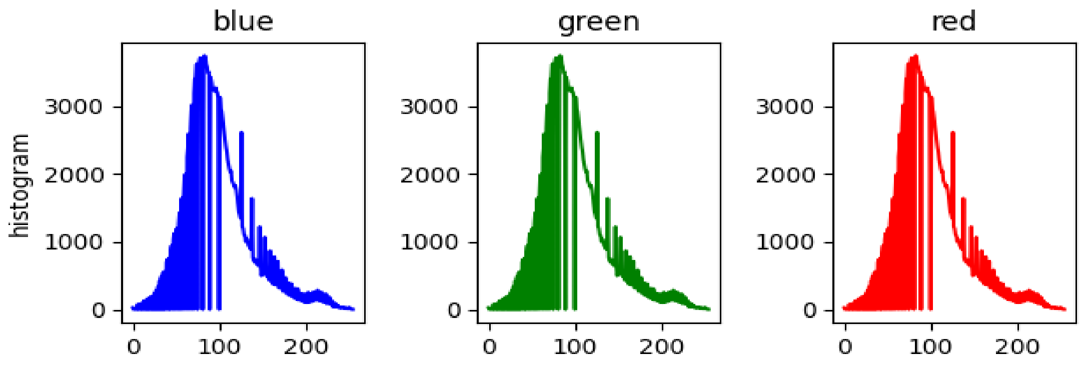
1. Please depict the **histogram** and **graph** of the assigned image "aerial_view.tif", and print out the source code? (10)

2. Please plot the **histogram** and **graph** of the image **<u>after Histogram Equalization</u>**, and print out the source code? (30)

3. Please plot the **histogram** and **graph** of the image **<u>after Histogram Matching (specificiation) by</u>** $p_{z(z_q)} = c \cdot z_q^{0.4}$, and print out the source code? (**NOTE**: the parameter, c, needs to calculate in advance) (40)

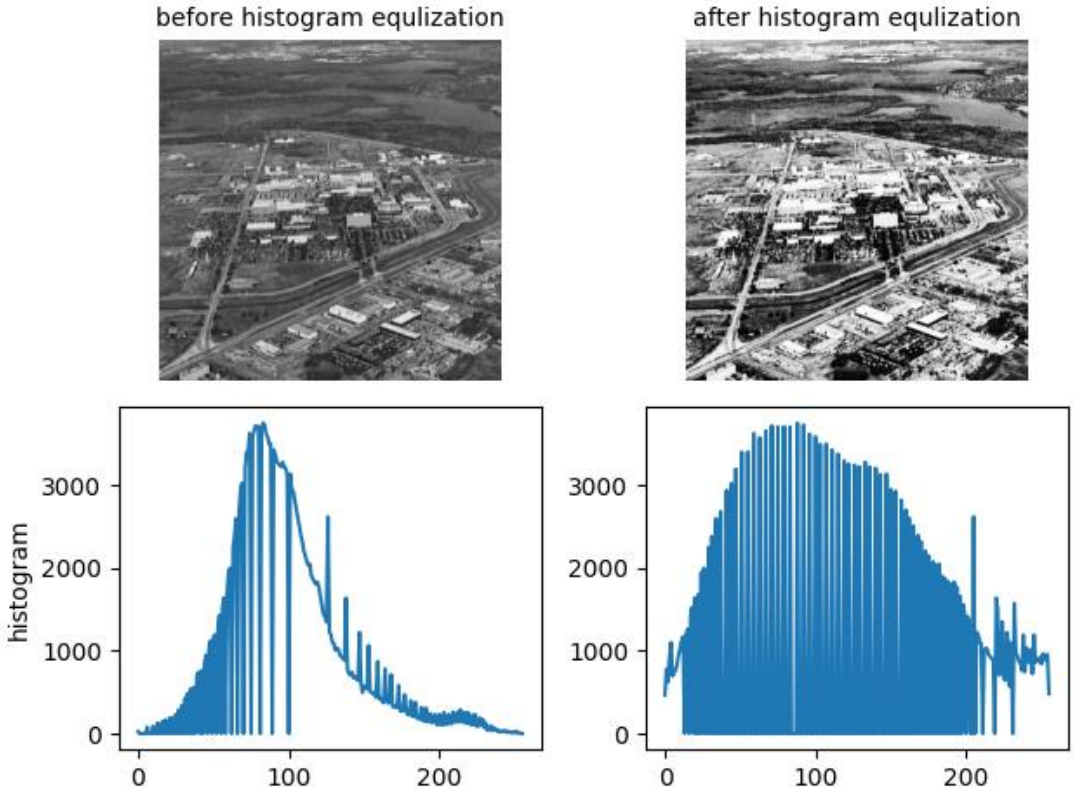
4. Please **comment** the original, the histogram-equalized and the histogram-matching images ? (20)

1.



```python
import cv2
from matplotlib import pyplot as plt

img = cv2.imread("aerial_view.tif")
print(img.shape)
cv2.imshow("aerial_view", img)
cv2.waitKey()
hist_b = cv2.calcHist([img], [0], None, [256], [0, 256])
hist_g = cv2.calcHist([img], [1], None, [256], [0, 256])
hist_r = cv2.calcHist([img], [2], None, [256], [0, 256])

fig, axs =  plt.subplots(nrows=1, ncols=3)

axs[0].plot(hist_b, 'b')
axs[1].plot(hist_g, 'g')
axs[2].plot(hist_r, 'r')

axs[0].set_ylabel('histogram')
axs[0].set_title("blue")
axs[1].set_title("green")
axs[2].set_title("red")

plt.tight_layout()
plt.show()
```

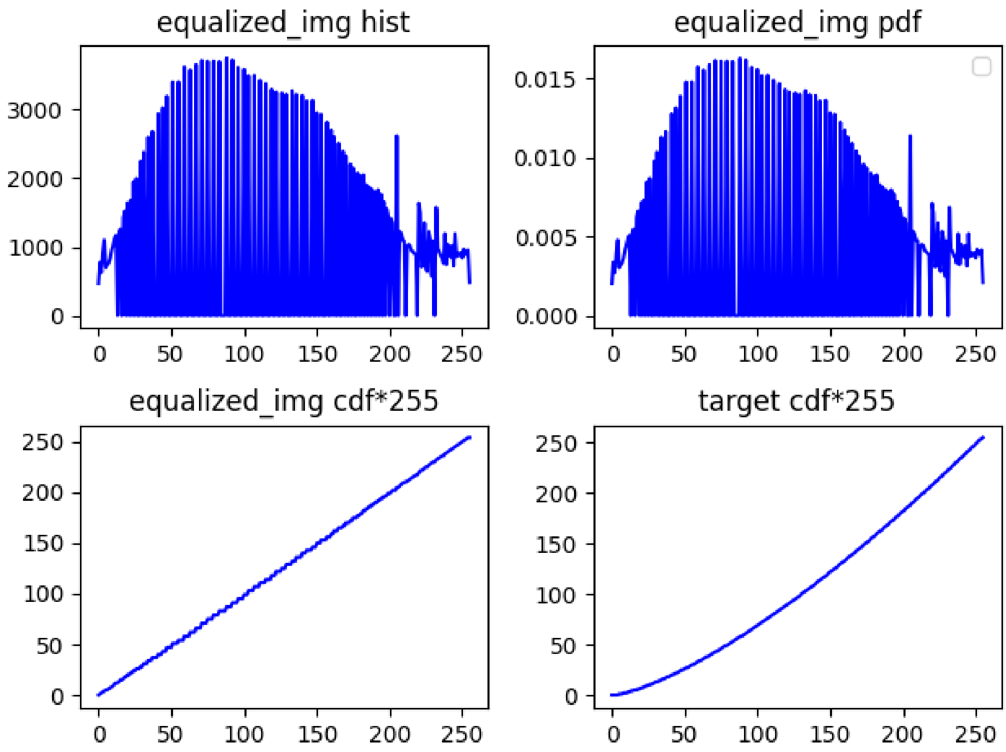
2.




```python
import cv2
from matplotlib import pyplot as plt

img = cv2.imread("aerial_view.tif")

# Note: When performing histogram equalization with OpenCV,
# we must supply a grayscale/single-channel image.
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
equalized_img = cv2.equalizeHist(gray)

# calculate histogram
hist = cv2.calcHist([gray], [0], None, [256], [0, 256])
equalized_hist = cv2.calcHist([equalized_img], [0], None, [256], [0, 256])

# plot result
fig, axs =  plt.subplots(nrows=2, ncols=2)
plt.subplot(221), plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)), plt.axis("off")
plt.subplot(222), plt.imshow(cv2.cvtColor(equalized_img, cv2.COLOR_BGR2RGB)), plt.axis("off")
axs[1, 0].plot(hist)
axs[1, 1].plot(equalized_hist)

axs[0, 0].set_ylabel('graph')
axs[1, 0].set_ylabel('histogram')

axs[0, 0].set_title("before histogram equlization", fontsize=10)
axs[0, 1].set_title("after histogram equlization", fontsize=10)

plt.tight_layout()
plt.show()
```

3.

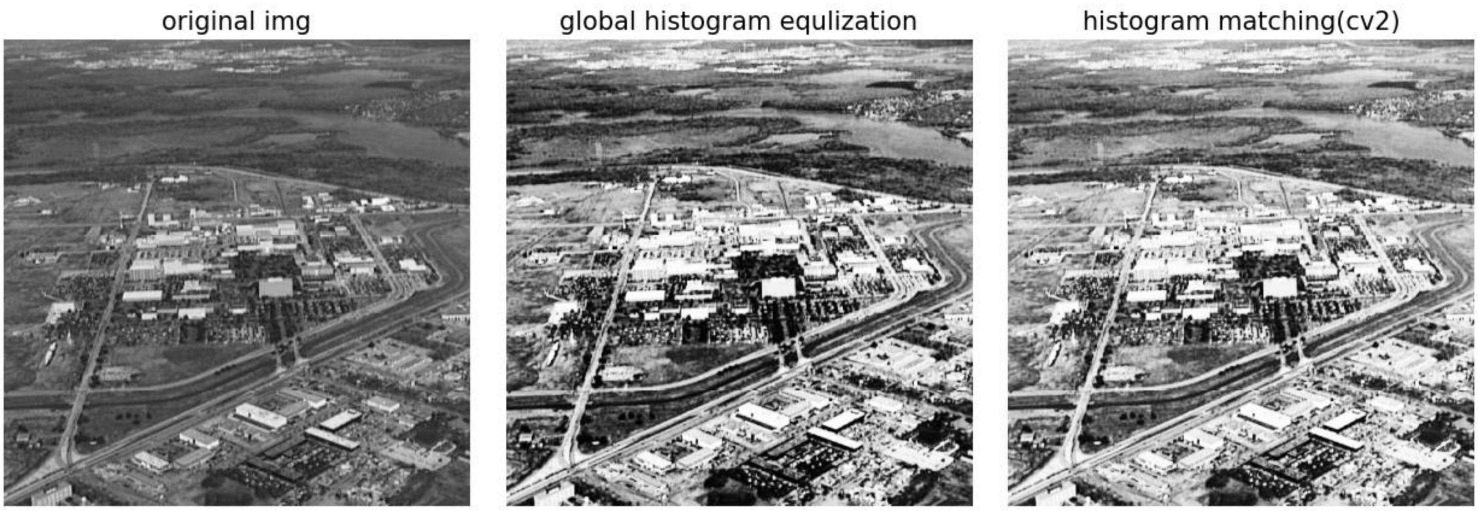
c = 0.0006




```python
import cv2
from matplotlib import colors, pyplot as plt
import numpy as np

img = cv2.imread("aerial_view.tif")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
equalized_img = cv2.equalizeHist(gray)
rows, cols = equalized_img.shape

hist = cv2.calcHist([equalized_img], [0], None, [256], [0, 256])
pdf = hist/equalized_img.size
cdf_255 = np.uint8(np.cumsum(pdf)*255)

cdf_Zq_255 = np.uint8(0.0006*np.cumsum(np.arange(256)**0.4)*255)  # c =  0.0005968222208221872
cdf_Zq_255[-1]  = 255    # 這邊須注意的是，因最後一個累積值算出來是256.35，超過 uint8 範圍因此將它指定為255
```

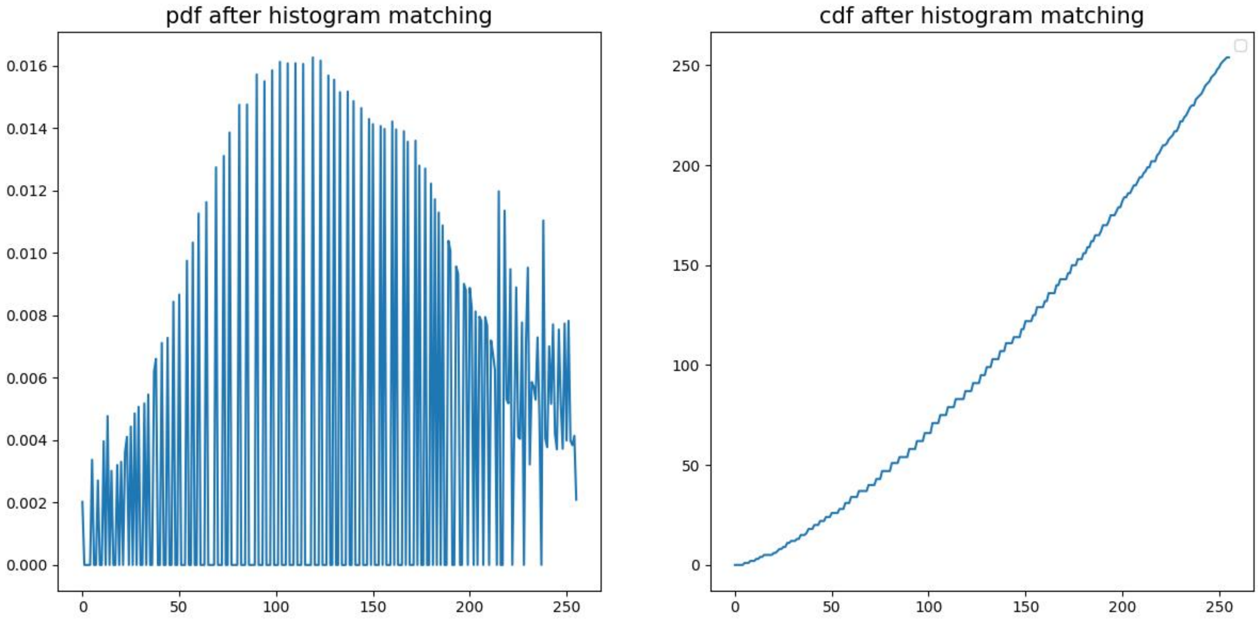
```python
# plot result
plt.subplot(221), plt.plot(hist, 'b'), plt.title("equalized_img hist")
plt.subplot(222), plt.plot(pdf, 'b'), plt.legend(), plt.title("equalized_img pdf")
plt.subplot(223), plt.plot(cdf_255, 'b'), plt.title("equalized_img cdf*255")
plt.subplot(224), plt.plot(cdf_Zq_255, 'b'), plt.title("target cdf*255")

plt.tight_layout()
plt.show()
```

```python
# calculate c parameter
cdf_Zq = np.cumsum(np.arange(256)**0.4)
#print(cdf_Zq_cv2_2[-1])  #1675
print("c = ", 1/cdf_Zq[-1])  # c =  0.0005968222208221872
```




```python
# implement histogram mapping
output_img = np.zeros_like(equalized_img)
all_vals_in_equalized_img = list(set(equalized_img.ravel()))

for val in all_vals_in_equalized_img:
    if val in cdf_Zq_255:
        index = 0
        for t in (cdf_Zq_255 == val):
            if t == True:
                break
            else:
                index += 1
        pixel_val = index
        indices = (equalized_img == val)
        output_img[indices] = pixel_val
    else:
        value_abs_diff = [np.abs(int(val) - int(cdf_Zq)) for cdf_Zq in cdf_Zq_255]
        pixel_val = np.argmin(value_abs_diff)           # Will print out smallest index meet the condition
        indices = (equalized_img == val)
        output_img[indices] = pixel_val
```

```python
#plot result in matplotlib
fig, axs =  plt.subplots(nrows=1, ncols=3)
plt.subplot(131), plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)), plt.axis("off")
plt.subplot(132), plt.imshow(cv2.cvtColor(equalized_img, cv2.COLOR_BGR2RGB)), plt.axis("off")
plt.subplot(133), plt.imshow(cv2.cvtColor(output_img, cv2.COLOR_BGR2RGB)), plt.axis("off")

axs[0].set_title("original img", fontsize=15)
axs[1].set_title("global histogram equlization", fontsize=15)
axs[2].set_title("histogram matching(cv2)", fontsize=15)

plt.tight_layout()
plt.show()
```




```
#plot result in matplotlib
fig, axs =  plt.subplots(nrows=1, ncols=2)
output_hist = cv2.calcHist([output_img], [0], None, [256], [0, 256])
output_pdf = output_hist/output_img.size
output_cdf_255 = np.uint8(np.cumsum(output_pdf)*255)

plt.subplot(121), plt.plot(output_pdf) ,plt.title("pdf after histogram matching", fontsize=15)
plt.subplot(122), plt.plot(output_cdf_255), plt.legend(), plt.title("cdf after histogram matching", fontsize=15)
plt.tight_layout()
plt.show()
```

4.

可以由 target cdf*255 的曲線(下凹)知道，原本亮度值較小的 Pixel 在做完 histogram matching 之後數目會減少(佔的機率減少)，而結果圖猶如推論，確實相較”global histogram equalization”暗區在做完 matching 後變亮了。總而言之，global histogram equalization 能調整對比對， histogram matching 則可以根據想要的 histogram 分布(or pdf of pixel values)去調整想要的像素值分布，彈性更大。

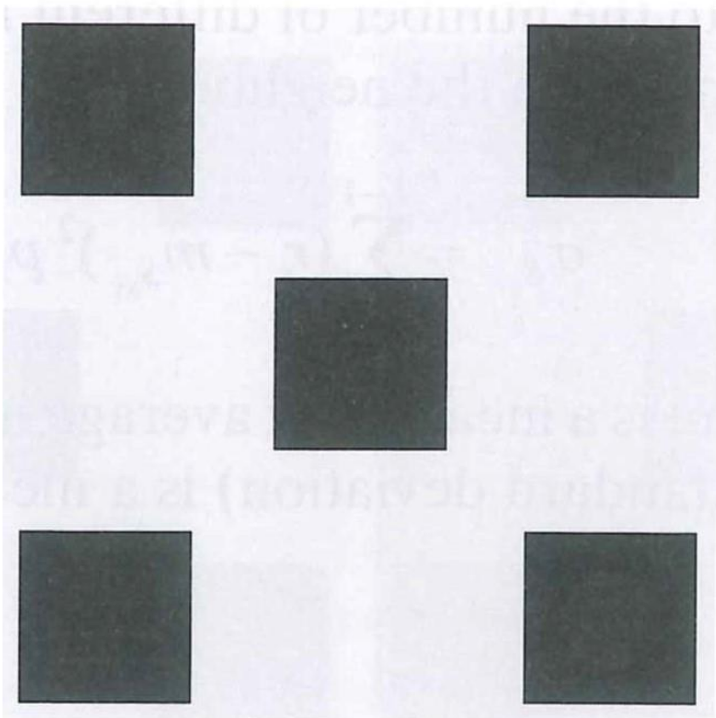
數位影像處理 DIP Homework Chapter 3_2 (100 pts)

1. Please use a **Histogram Statistics method** and a **local enhancement method** to extract the hidden image of the image, 'hidden object.jpg.' Please describe the your parameters and method in detailand print out the source code? (40+40)

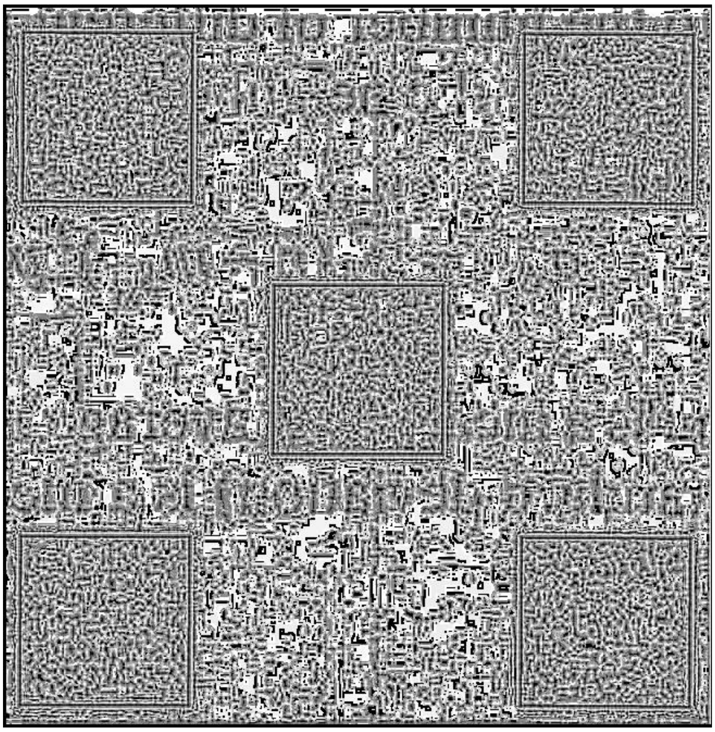
2. Please **comment** and **compare** Histogram Statistics method and a local enhancement method? (20)

1.

Original image (in gray level)



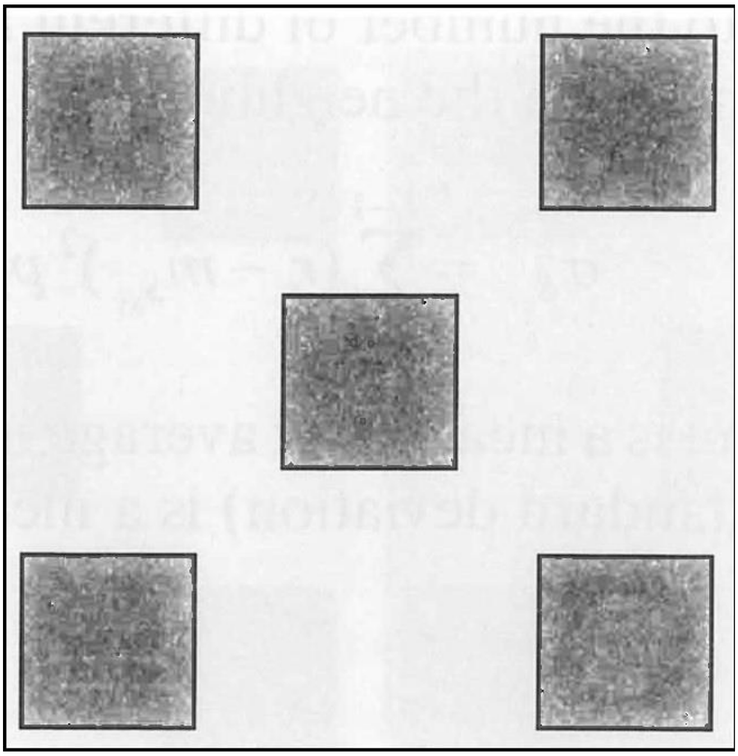
local enhancement method



```python
import cv2
import numpy as np
from matplotlib import pyplot as plt
from itertools import product


img = cv2.imread("hidden_object.jpg")  # shape: (665, 652, 3)
# print(img[:,:,0].all()==img[:,:,1].all()==img[:,:,2].all())  #True R=G=B
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
equalized_img = cv2.equalizeHist(gray)



cv2.imshow("img", img)
cv2.imshow("global_histogram_equalization", equalized_img)
#cv2.imwrite("global_histogram_equalization.jpg", equalized_img)


# use local enhancement method
kernel_size = 5
output_img = np.zeros_like(gray)
#padded_img = cv2.copyMakeBorder(gray, kernel_size//2, kernel_size//2, kernel_size//2, kernel_size//2,cv2.BORDER_DEFAULT)
rows, cols = gray.shape
```

```python
for row in np.arange(kernel_size//2+1, rows - (kernel_size//2+1)):  #652
    for col in np.arange(kernel_size//2+1, cols - kernel_size//2+1):
        area = gray[(row-(kernel_size//2)) : (row+(kernel_size//2+1)), (col-(kernel_size//2)) : (col+(kernel_size//2+1))]
        if area.size == kernel_size*kernel_size:
            equalized_area = cv2.equalizeHist(area)
            output_img[row, col] = equalized_area[kernel_size//2, kernel_size//2]   # (kernel_size, kernel_size) 中心的那個像素值
        else:
            output_img[row, col] = gray[row, col]

cv2.imshow("local_histogram_equalization", output_img)
#cv2.imwrite("local_histogram_equalization.jpg", output_img)
cv2.waitKey()
cv2.destroyAllWindows()
```

Histogram Statistics method



使用的公式:

$$g(x,y)=\begin{cases} C f(x,y) & \text{if } k_0 m_G \le m_{S_{xy}} \le k_1 m_G \text{ AND } k_2\sigma_G \le \sigma_{S_{xy}} \le k_3\sigma_G \\ f(x,y) & \text{otherwise} \end{cases} \tag{3-29}$$

使用的參數 :

k0, k1, k2, k3 = 0, 0.5, 0, 0.1

kernel_size = 5

C = 5

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
from itertools import product


img = cv2.imread("hidden_object.jpg")  # shape: (665, 652, 3)
# print(img[:,:,0].all()==img[:,:,1].all()==img[:,:,2].all())  #True R=G=B
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

global_mean = np.mean(gray)
print(global_mean)

# calculate global std
#global_variance = np.sum((bins-global_mean)**2*(pdf_cv2[:,0]))
global_std = np.std(gray)
print(global_std)
```

```python
for row in np.arange(kernel_size//2+1, rows - (kernel_size//2+1)):  #652
    for col in np.arange(kernel_size//2+1, cols - kernel_size//2+1):
        area = gray[(row-(kernel_size//2)) : (row+(kernel_size//2+1)), (col-(kernel_size//2)) : (col+(kernel_size//2+1))]
        local_mean = np.mean(area)
        local_std =np.std(area)

        if (local_mean>= k0 and local_mean <= k1 *global_mean) and (local_std >= k2 and local_std <= k3*global_std):
            output_img[row, col] = gray[row, col]* 5
            #output_img[(row-(kernel_size//2)) : (row+(kernel_size//2+1)), (col-(kernel_size//2)) : (col+(kernel_size//2+1))] = area*5
        else:
            output_img[row, col] = gray[row, col]
            #output_img[(row-(kernel_size//2)) : (row+(kernel_size//2+1)), (col-(kernel_size//2)) : (col+(kernel_size//2+1))] = area


cv2.imshow("Histogram Statistics", output_img)
#cv2.imwrite("Histogram_Statistics.jpg", output_img)
cv2.waitKey()
cv2.destroyAllWindows()
```

2.

覺得結果做得不是很好，推論可能是圖源的關係，雜訊太多，五個方塊中隱藏的圖案像素值與背景太像，因此很難以像素質及對比度做區分。

不過仍可經比較得知，用同樣的 kernel size 大小，欲找到原圖的隱藏圖案，用 **Histogram Statistics method** 相較 **local enhancement method** 能得到較好的結果。