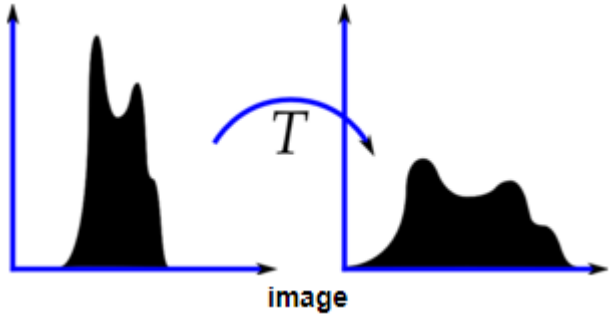
Explanation

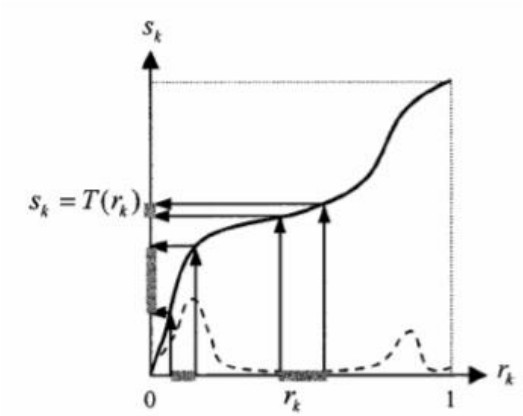
第四張圖是偏亮的圖、第五張圖是偏暗的圖

我用的兩種方法是 Global equalization 和 Contrast Limited Adaptive Histogram Equalization

原理：

Global equalization：





Transformation function for histogram equalization.

$$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j), \quad = \sum_{j=0}^{k} \frac{n_j}{n},$$
$$0 \le r_k \le 1 \quad \text{and} \quad k = 0, 1, \ldots, L-1$$

讓影像的強度分配平均舒展，而不是集中於小範圍的強度，此時的轉換取縣是用長張圖計算出來

code：

```
image = cv2.imread('5_resize.jpg',0) # Only for grayscale image

eqa = cv2.equalizeHist(image)
#eqa = np.hstack((image,eqa)) #stacking images side-by-side
cv2.imwrite('5_brighter_global_histequ.jpg', eqa)
```

Contrast Limited Adaptive Histogram Equalization:

(參數可調整的 histrogram equalization)

a. To understand the algorithm, consider a matrix

$$A=\begin{bmatrix}2&1&0\\0&1&3\\4&1&4\end{bmatrix}$$ and the range be 0-4

b. Copy the matrix A into another matrix B and pad it with zero on all sides

$$B=\begin{bmatrix}0&0&0&0&0\\0&2&1&0&0\\0&0&1&3&0\\0&4&1&4&0\\0&0&0&0&0\end{bmatrix}$$

c. Now consider a window of size 3 by 3. It starts tracing from the first position(1,1)

$$B=\begin{bmatrix}\boxed{\begin{matrix}0&0&0\\0&2&1\\0&0&1\end{matrix}}&0&0\\0&4&1&4&0\\0&0&0&0&0\end{bmatrix}$$

d. Find the probability density of each value inside the matrix.
Probability of (0/4) =6/9. Similarly, the probability of (1/4) = 2/9.
The probability of (2/4) = 1/9.
The probability of (3/4) =0
The probability of (4/4) =0

e. Now find the cumulative distribution of each value.
For 0, 6/9*4= 3.
a. For 1, pdf of 0+ pdf of 1. (ie) 8/9*4=4
b. 2=>9/9*4=4 3=>4 4=>4

f. Find the middle element in the window

$$B=\begin{bmatrix}\boxed{\begin{matrix}0&0&0\\0&\textcircled{2}&1\\0&0&1\end{matrix}}&0&0\\0&4&1&4&0\\0&0&0&0&0\end{bmatrix}$$

Replace value of 2 with the value obtained by CDF. It is 4. Similarly, the computation is done for the whole matrix.

g. Final Result: $\begin{bmatrix}4&3&3\\2&2&4\\4&3&4\end{bmatrix}$

$$\mathrm{ACE} = k_1\left[\frac{m_{I(r,c)}}{\sigma_I(r,c)}\right]\left[I(r,c) - m_I(r,c)\right] + k_2\, m_I(r,c)$$

where $m_{I(r,c)}$ = is the mean for the entire image $I(r,c)$
σ_I = local standard deviation (in the window under consideration)
m_I = local mean (average in the window under consideration)
k_1, k_2 = constants, vary between 0 and 1

主要是 local 形式，小區間的轉換曲線的是由那個區塊計算而出，是根據該區域(而非整張影像)的資訊來增強對比。

code：

```
image = cv2.imread('5_resize.jpg',0) # Only for grayscale image

# create a CLAHE object (Arguments are optional).
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
cl1 = clahe.apply(image)
# cl1 = np.hstack((image,cl1))
cv2.imwrite('5_brighter_local_histequ.jpg', cl1)
```

Reference

https://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html

Histograms Equalization in OpenCV

OpenCV has a function to do this, **cv2.equalizeHist()**. Its input is just grayscale image and output is our histogram equalized image.

Below is a simple code snippet showing its usage for same image we used :

```
1 img = cv2.imread('wiki.jpg',0)
2 equ = cv2.equalizeHist(img)
3 res = np.hstack((img,equ)) #stacking images side-by-side
4 cv2.imwrite('res.png',res)
```

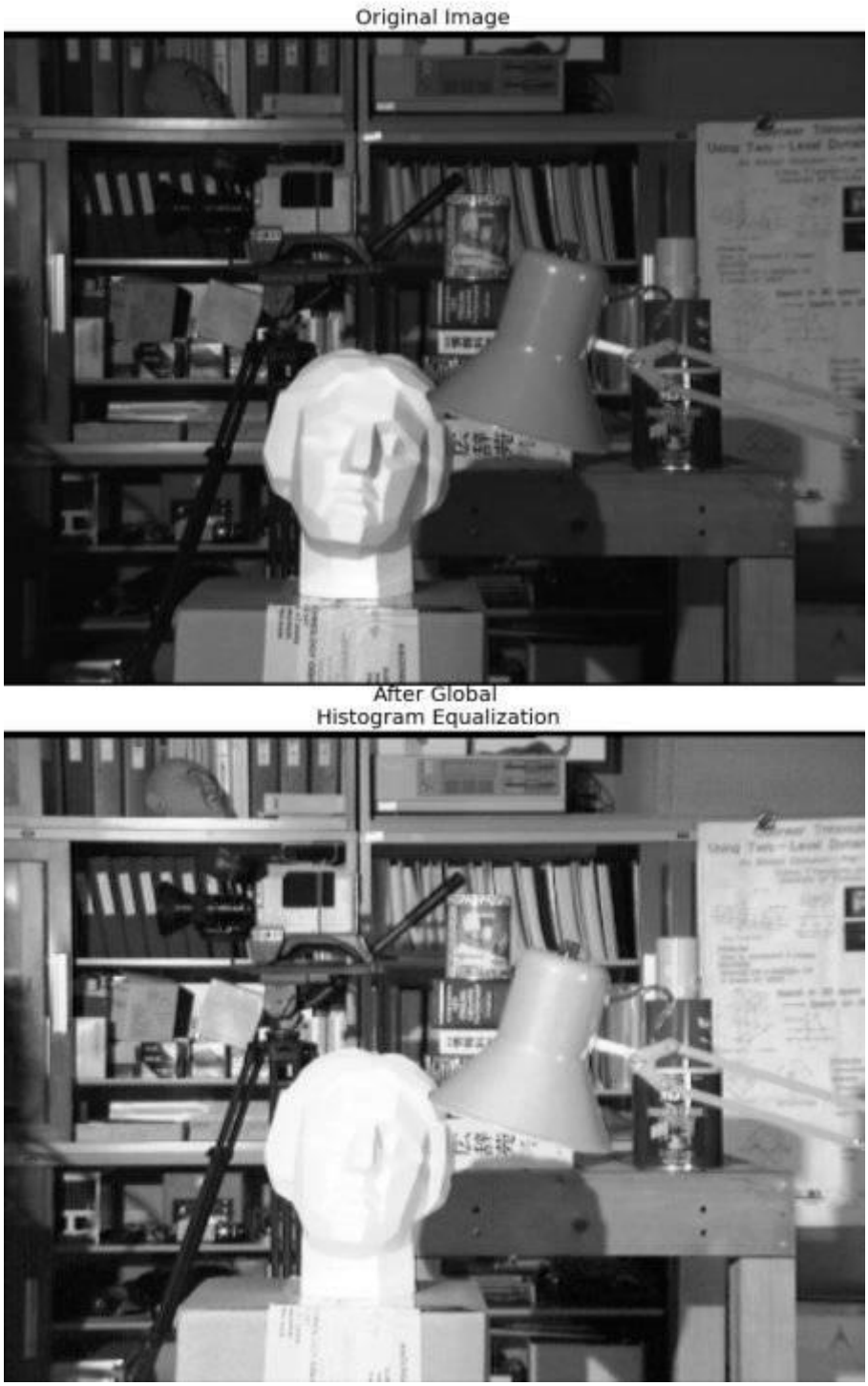
image

So now you can take different images with different light conditions, equalize it and check the results.

Histogram equalization is good when histogram of the image is confined to a particular region. It won't work good in places where there is large intensity variations where histogram covers a large region, ie both bright and dark pixels are present. Please check the SOF links in Additional Resources.

CLAHE (Contrast Limited Adaptive Histogram Equalization)

The first histogram equalization we just saw, considers the global contrast of the image. In many cases, it is not a good idea. For example, below image shows an input image and its result after global histogram equalization.



image

It is true that the background contrast has improved after histogram equalization. But compare the face of statue in both images. We lost most of the information there due to

over-brightness. It is because its histogram is not confined to a particular region as we saw in previous cases (Try to plot histogram of input image, you will get more intuition).

So to solve this problem, **adaptive histogram equalization** is used. In this, image is divided into small blocks called "tiles" (tileSize is 8x8 by default in OpenCV). Then each of these blocks are histogram equalized as usual. So in a small area, histogram would confine to a small region (unless there is noise). If noise is there, it will be amplified. To avoid this, **contrast limiting** is applied. If any histogram bin is above the specified contrast limit (by default 40 in OpenCV), those pixels are clipped and distributed uniformly to other bins before applying histogram equalization. After equalization, to remove artifacts in tile borders, bilinear interpolation is applied.

Below code snippet shows how to apply CLAHE in OpenCV:

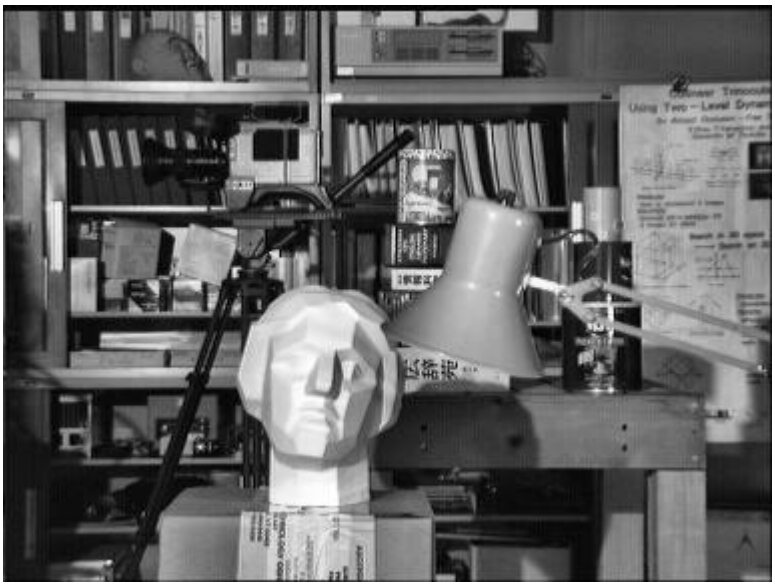
```
import numpy as np
import cv2

img = cv2.imread('tsukuba_l.png',0)

# create a CLAHE object (Arguments are optional).
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
cl1 = clahe.apply(img)

cv2.imwrite('clahe_2.jpg',cl1)
```

See the result below and compare it with results above, especially the statue region:



image