Digital Image Processing

Content:

1.Introduction

我們常說一張圖勝過千言萬語，這是因為人類的視覺感官對於圖像非常強烈，往往用筆墨還不足以形容。例如，陳述一個小女還有多可愛，與其用盡所有形容詞，不如出示一張她的照片。

顯然圖像是傳達訊息非常直接又有效的方式，但是若上述照片沒有拍好(例如光線不足集拍攝拾手部顫抖等問題)，則其訊息傳達的效果可能還不如用語言形容。對於效果不佳的圖片，特別是記錄珍貴歷史性時刻者，如果能以經濟又有效的方式加以處理，使其達成原本能傳達的訊息量，是否很好呢？這就是影像處理（image processing）所要探討的問題。換言之，就是以有效（effective）而又有效率（efficient）來處理影像，使期能傳達我們所需要的訊息。

數位影像處理(digital image processing)又是什麼呢？簡言之，就是將所擷取到的影像數位化(digitized)，並且利用電腦高速計算的能力來處理影像。比起攝影師暗房技巧那一類的光學影像處理以及像調整電視機旋鈕來改變電壓使對比或亮度改變的類比影像處理，數位影像處理具有更大的彈性(flexibility)以及效率(efficiency)，如圖 1.1~1.2。數位影像處理除了可以改善影像資訊使人理解外，他還能使機器像人具有視覺能力。一影像處理系統其架構圖，如圖 1.3。

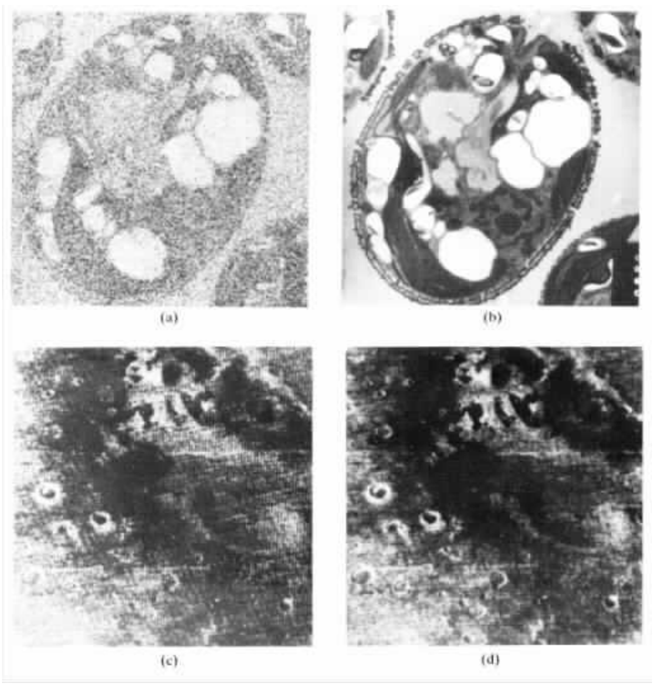


圖 1.1、數位影像處理

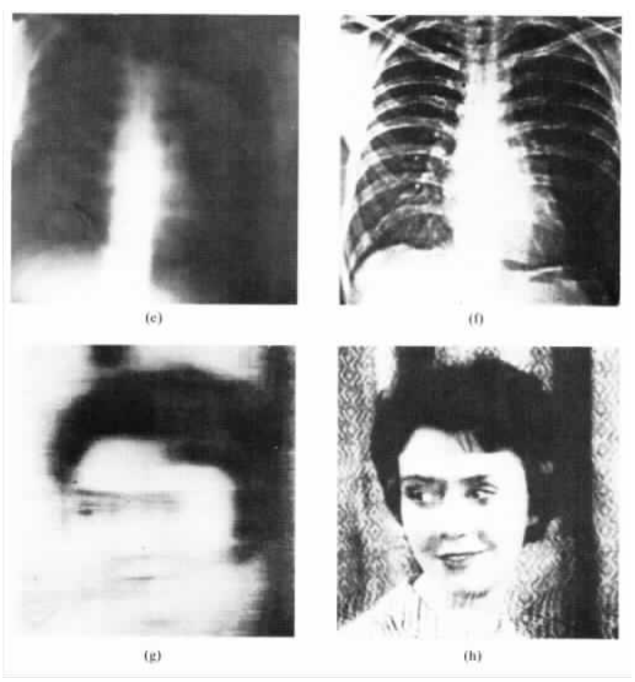


圖 1.2、數位影像處理

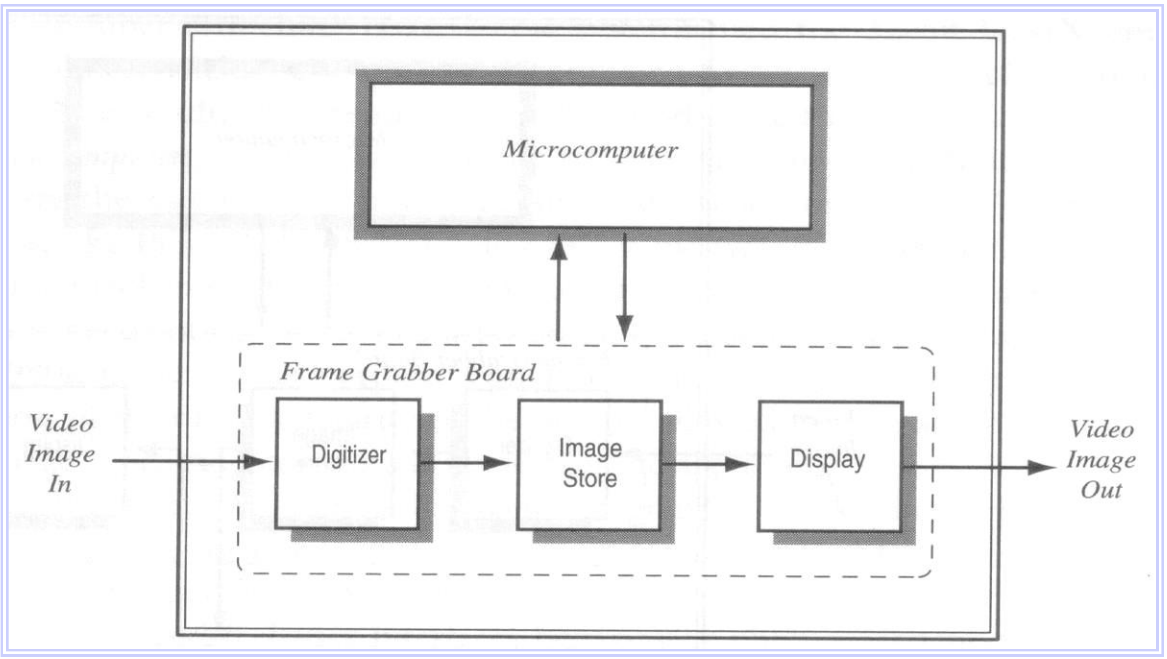


圖 1.3、影像處理系統架構圖

從 1964 年迄今，影像處理的領域有快速的發展。主要的應用領域：

➤**太空計畫：**1964 年，美國太空總署的噴射推進實驗室開始使用計算機（computer）技術改善從太空探測器所獲得的影像，當時用計算基礎理由巡航者 7 號（Ranger7）傳回的月球圖片，以校正電視攝影機所存在的幾何失真或響應失真。其後有一連串的星際探測計畫，直到現在一直持續不斷送回更多影像。

➤**生物醫學領域：**首先用於細胞分類、染色體分類和放射影像的處理。1972 年 X 光電腦斷層掃描術（Computer Tomography, CT）獲得實際應用，在醫學工程上為一大突破；1977 年白血球自動分析儀問世；1980 年發展出心臟活動立體影像的技術。

隨著醫學影像成像理論與計數之進步，醫學影像在臨床診療上扮演著愈來愈重要的角色。從簡單的 X 射線成像技術到各式的斷層掃描醫學影像提供了其他方法所難以獲得的資料，比如人體之解剖學，生理及生化功能之訊息。

醫學影像所包含的範疇十分廣泛。廣義的說，凡將人體中某種訊息以影像之方式表示而有助於醫療工作者，皆可稱之為醫學影像。例如 X 射線影像（分為一般 X 射線影像、血管 X 射線影像 i.e. 血管攝影、乳房 X 射線影像 i.e.乳房攝影、與 X 射線斷層掃描）、放射性核影像（radionuclide imaging，分為直線式掃描器、閃光照相機 scintillation camera、單光子電腦斷層掃描 Single Photon Emission Computerized Tomography、與正子放射式斷層掃描 Positron Emission Tomography）、核磁共振影像(nuclear magnetic resonance imaging, MRI)和超音波影像(ultrasound imaging) 皆為眾所皆知的醫學影像。

對這些影像增強對比度或將亮度值著色等之處理，可幫助臨床醫師診斷如肺病、腫瘤及心血管等疾病。

➤**遙測資料分析：**遙測（Remote Sensing）常以顏色為依據，由飛機、衛星及太空船上的多光譜（Multi-Spectral）掃描器攝得影像，範圍從可見光至紅外光（有時含紫外光）分成幾個頻帶，每一各頻帶所攝得的顏色都不同，可用於土地使用、作物收成、作物病害偵測森林及水資源調查、環境污染偵測、地質與地形分析、礦藏探勘及氣象預測等。由於資料量龐大，因此必須尋求處理與分析這些影像的自動方法，特別是影像對比度增強、分割（segmentation）及影像識別（recognition）之技術。

➤**科學研究：**考古學可用影像處理方法來復原模糊或其他惡化狀況的珍貴文物影像。若這些文物在拍成照片後，不幸遺失或損害，這些照片將成為唯一可用的紀錄。

➤**一般工商業應用：**例如用非破壞性影像處理方式檢查零件內部的瑕疵及焊接品質、金屬材料之成分與結構分析、紡織品品質檢測、印刷電路板（PCB）及焊點不良檢查、零件安裝檢查等。又如郵遞區號自動讀取系統、組裝零件辨識、光學字之讀取與辨認（Optical Character Recognition，OCR）、條碼讀取、與身份識別（使用指紋、瞳孔或顏面等影像）。

➤**通訊與電腦資料儲存：**例如傳真機（FAX）、影像電話（Video Phone）及視訊會議(Video Conference)等採用影像資料壓縮（image compression）的技術，使通訊更快速。此外，採用影像資料壓縮將可大幅地降低影像資料儲存量以降低儲存負擔。

整個影像處理的領域仍蓬勃發展中，其原因有：（1）電腦計算能力愈來愈強。（2）影像擷取與顯示設備更加普遍與便利。（3）影像處理的觀念日益普及，將更容易創造新的應用。

2. Basic Concept of Digital Image Processing

A. 光學影像處理（Optical processing）

例如攝影師在暗房沖洗照片，將照片沖洗的更清晰、效果更好。

Disadv.：耗時費力，效率低。

B. 類比影像處理（Analog processing）

影像轉換成類比的電子訊號，再改變這些訊號以達到影像處理的目的。例如：TV 的旋轉鈕⇨影像對比或亮度改變。

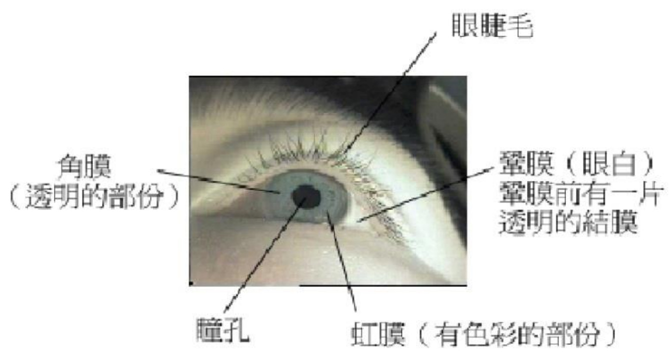
C. 數位影像處理(Digital Image Processing)

經過數位化（digitized）的影像是由許多不同的亮點所組成，每一亮點均有其特定位置及亮度值，這些數據藉著電腦的快速運算能力，使得各種複雜的影像處理得以實現，且效果更佳。

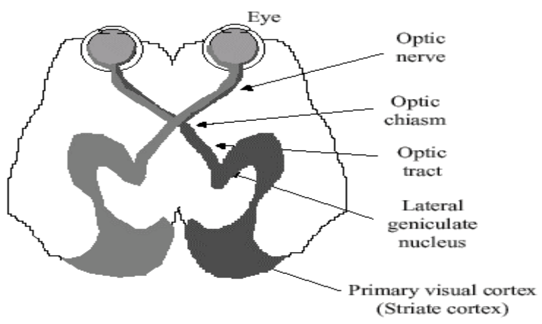
2.1 人類視覺系統

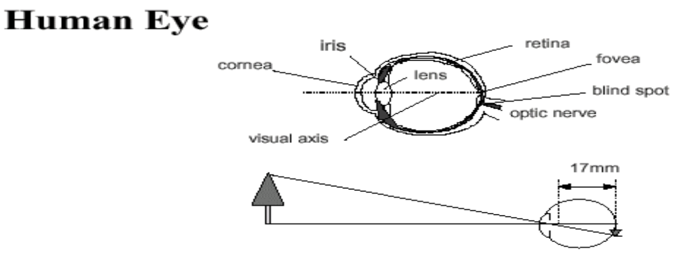
所有的影像最後都必須經由人類的視覺系統來處理。因此在改善影像品質之前，我們必須瞭解人類視覺系統的特性。

眼睛，如圖 2.1，是一個複雜的系統，他將我們所接觸到的影像轉換成神經脈衝（nerves impulse），再由我們的大腦接收並轉換成像。眼睛的主要組織，由景物所產生的光線聚集在水晶體(lens)，並投影於視網膜(retina)的表面；虹膜(iris)控制光線的通過量，水晶體與虹膜都由眼角膜(cornea)所保護，而視網膜係由對光敏感的元素所構成，也就是感體(rod)和錐體(cone)，數以億計的光接收體，將這些光轉換成神經脈衝，神經脈衝再透過視覺神經，從眼中將影像傳達大腦，而大腦則將這些神經脈衝所接收之資訊形成我們所看到的影像。因此，一物體在眼睛的成像如圖 2.2。



（A）


Eye
Optic nerve
Optic chiasm
Optic tract
Lateral geniculate nucleus
Primary visual cortex (Striate cortex)



（B）

圖 2.1（A），(B) 眼睛結構

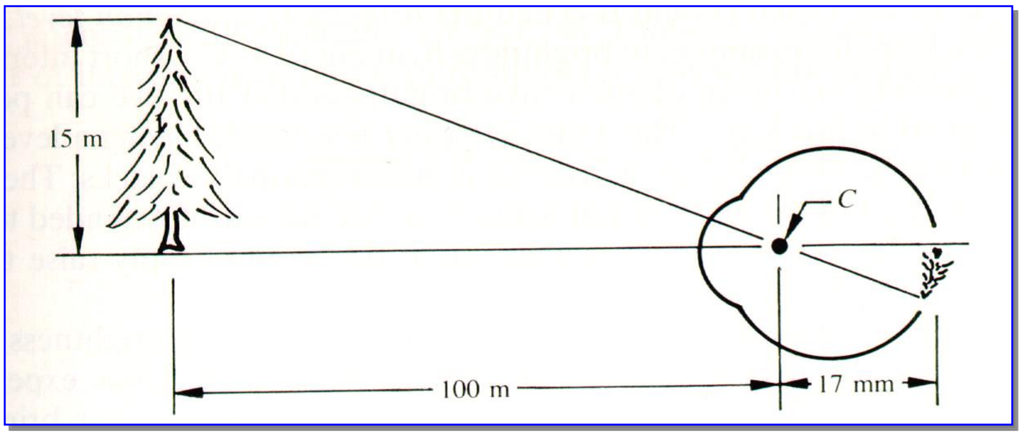
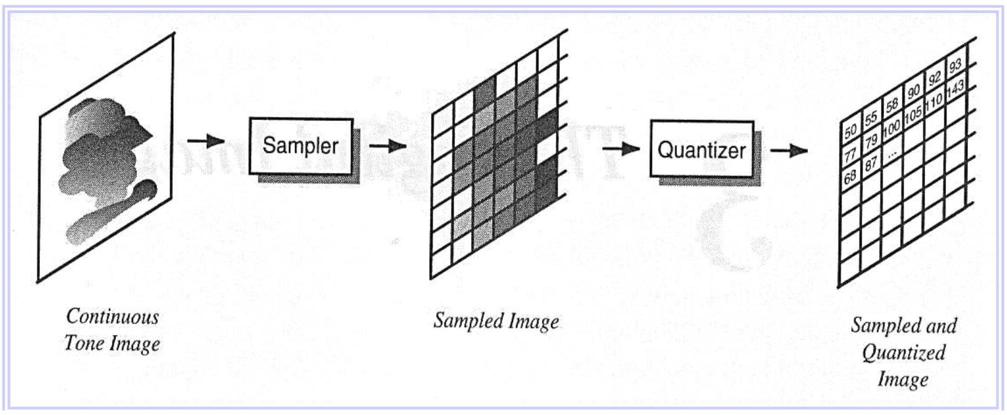


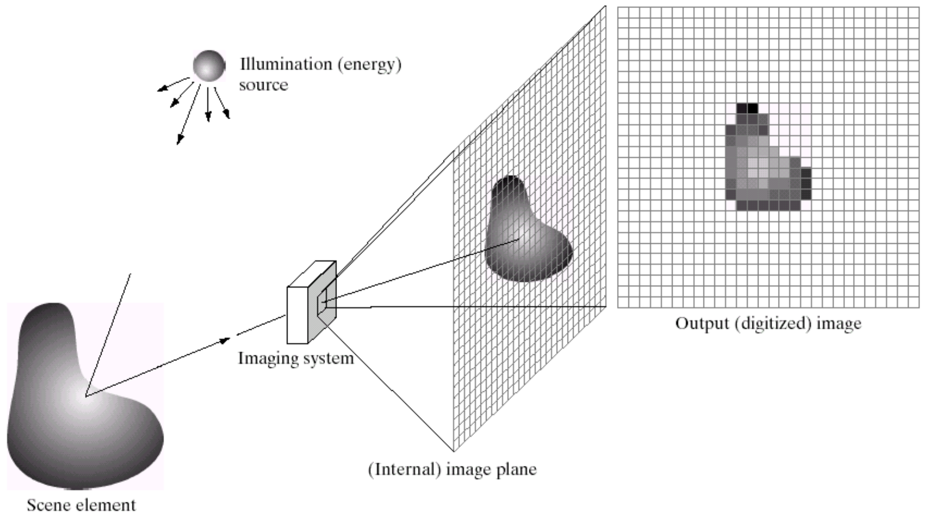
圖 2.2 一物體在眼睛的成像。

2.2 數位化影像（Digitized Images）

要利用電腦處理影像，首先一定要將影像轉換為電腦可以存取和處理的數據。黑白照片是由黑到白的連續性濃淡所構成。它是由不可分割的明暗帶所組成的。因此數位化影像就是轉換連續性資料成為數位化訊號。數位化的程序可由下圖 2.3 來表示：

$$\begin{tikzcd}
\text{類比影像信號} \arrow[r] & \boxed{\text{取樣器 (sampler)}} \arrow[r] & \boxed{\text{量化 (Quantization)}} \arrow[r] & \text{數位化影像}
\end{tikzcd}$$



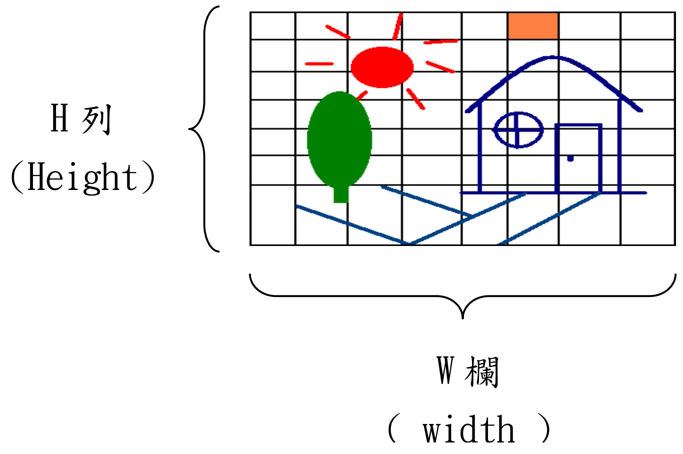


a b c d e

FIGURE 2.15 An example of the digital image acquisition process. (a) Energy ("illumination") source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

圖 2.3 影像的數位化程序

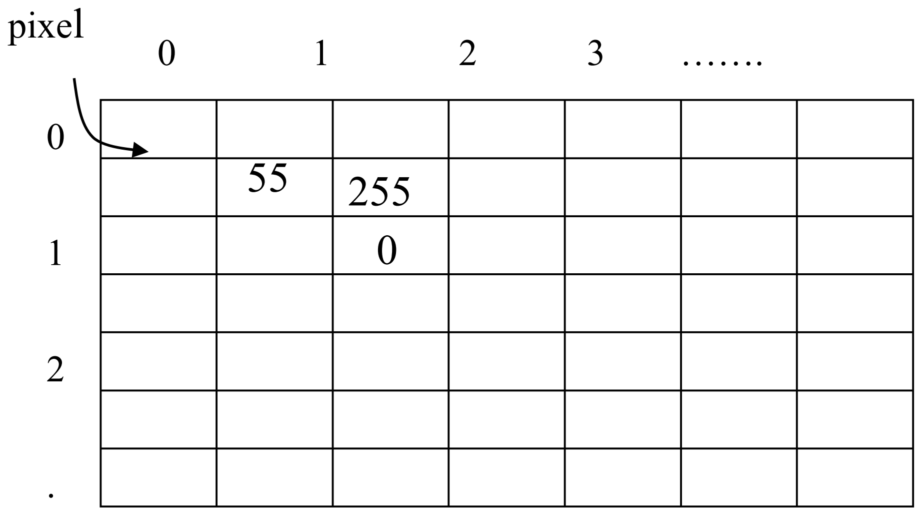
A.取樣（Sampling）一小格,一像素



因為一張影像被分割成一小格一小格水平被分割成 W 點，垂直分割 H 點.因此 共有 W*H 點

空間解析度：常用單位
dpi，dots per inch
ppi，pixels per inch

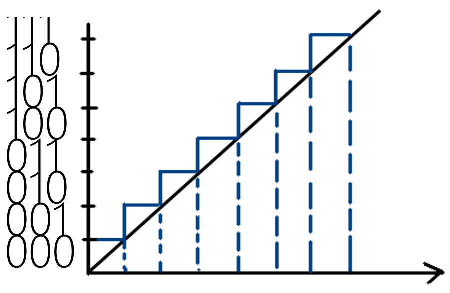
對每一個攝取到的資料，賦于一個數值代表他的濃淡及其對應位置。通常我們稱此一最小的單位為影像元素（picture element）簡稱像素（pixel），它是組成數位影像的最基本單位。



現在，我們的影像已被分割成獨立且不連續的點，且每一個點均有其固定的位置及其座標和亮度值。

B.量化（Quantization）

影像取樣完成後,每一像素點的樣本值（亮度 or 色彩）將在某一正確的誤差範圍內，被轉換成一對應的整數值。



取樣點數值

量化的目的是使電腦可以 B 個位元來表示一取樣值儲存,所以一般將取值範圍分成

2^B 個區間

ex: B= 8,10,12,16

量化: 亮度（色彩）解析度

ex. B=8 , 2^8=256 個灰階度

通常，量化的區間越多(越細),則由量化的樣本值(即電腦內的數位 image)將與實際的影像越接近。當區間不夠時,在影像的灰度值度變化緩慢的區域內將出現原 image 上沒有的假輪廓 false contour。

Q1：當我們在處理影像的數位化時，此一代表性影像和原來的影像作比較，它是達到何種相近的程度？

Ans. 解析度（resolution）是將某物分割其組成的基本原見的大小。在影像處理中，解析度可分為兩種形式，一種是空間的解析度（spatial resolution），另一種則是亮度(色彩)的解析度(brightness or color resolution）。

◆空間的解析度 (Spatial Resolution)

空間是指所謂的二維空間（two-dimensional space），二維空間的物體擁有一固定的高度與寬度。當討論空間解析度時，即是描述他可能劃分成多少點。也就是說，當此解析度劃分的越細，那麼所得到的是一個更加接近影像的外觀。最佳的狀況是經數位化後，沒有遺失任何資訊於轉換過程之中，也就是說，適當地顯示此數位化的影像，對觀察者而言，他和原始的影像一致的。

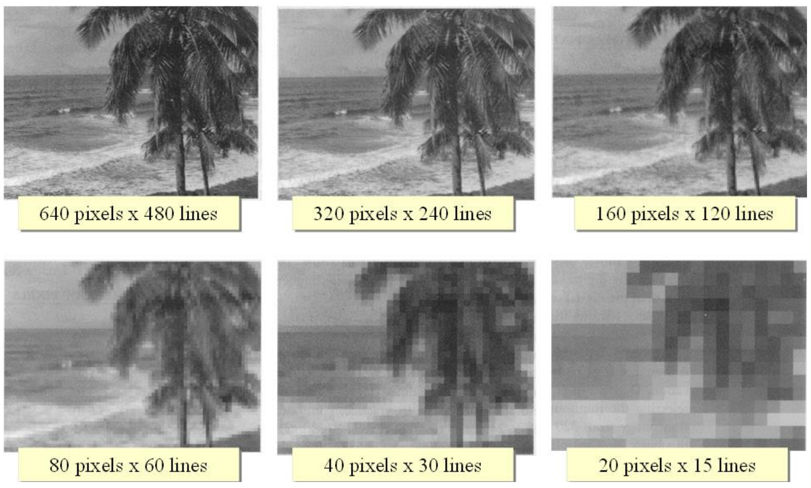


圖 2.4 A image displayed in various spatial resolutions.

◆亮度的解析度 (Brightness Resolution)

影像的第二種解析度，就是所謂的亮度。每一個像素代表原始影像在某一點抽樣的亮度。亮度解析度的觀念即是討論數位像素對於原影像在同一個位置其亮度的正確性。數位化的構成，其抽樣是在事先預定的方格位置。一個抽樣的亮度就被轉換成一個相等的整數值，即是所謂的量化（quantization）。

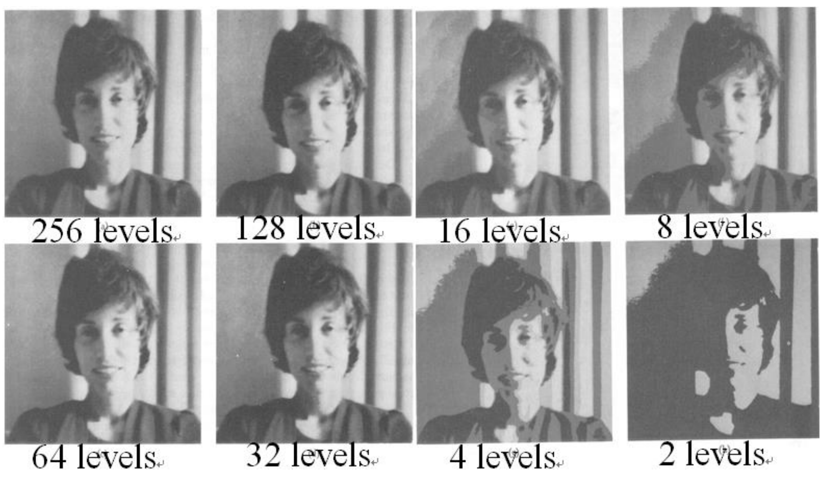


圖 2.5 A image displayed in 256,128,64,32,16,8,4 and 2 levels

Q: 如何計算一張數位化後 image 的儲存空間?

Ans:B= 8 (1 byte, 256 色) 所以 共 W*H*1 bytes

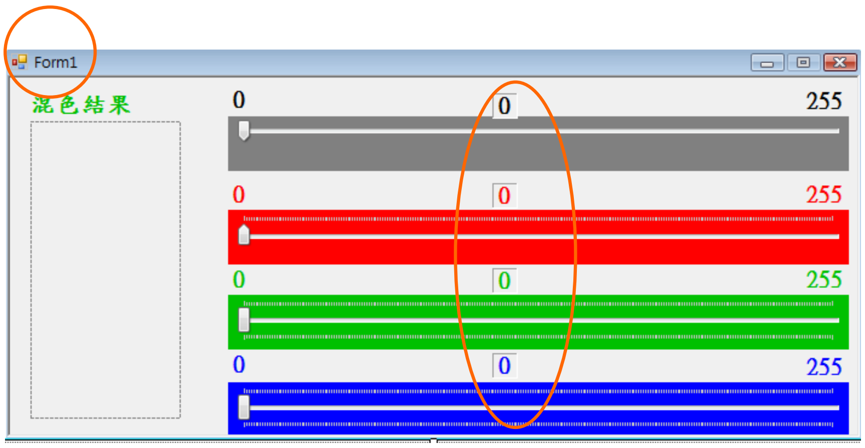
Ex. W=256, H=256, 所以 256*256*1= 64K bytes
Ex. B=24, W=512, H=512 , 所以 512*512*(24/8) = 768K bytes

◆Color & Pallet 色彩與調色盤

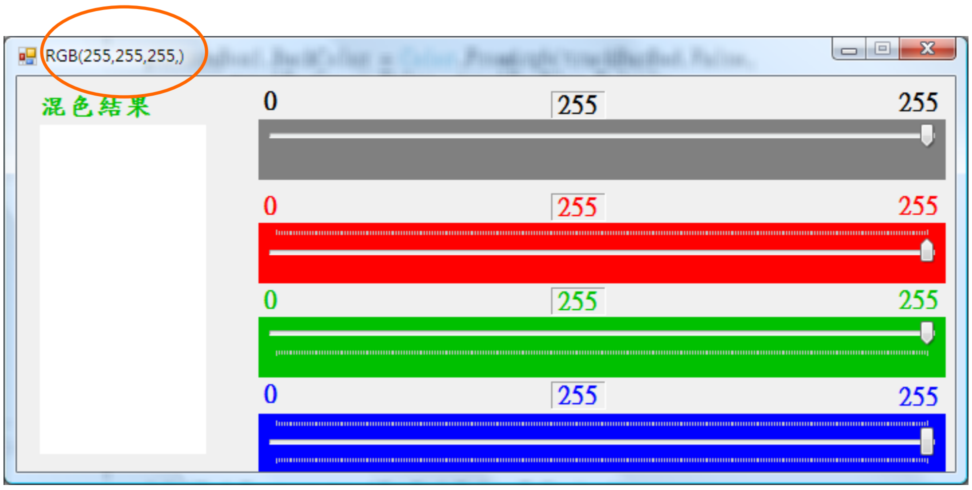
由彩虹或光稜鏡顯示出的真實光譜內,我們常會發現光譜內光線的排列次序為紅色→ 橙色→ 黃色→綠色→藍色→紫色

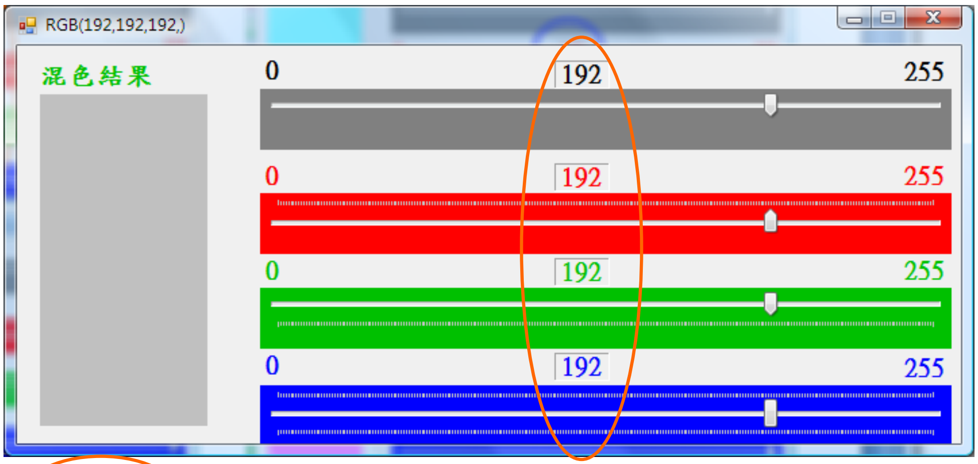
Q.電腦 monitor 只有 **Red**、**Green**、**Blue**(三原色),why?
Ans. 因為人類眼睛的視網膜 (Retina) 對這三種顏色特別敏感。

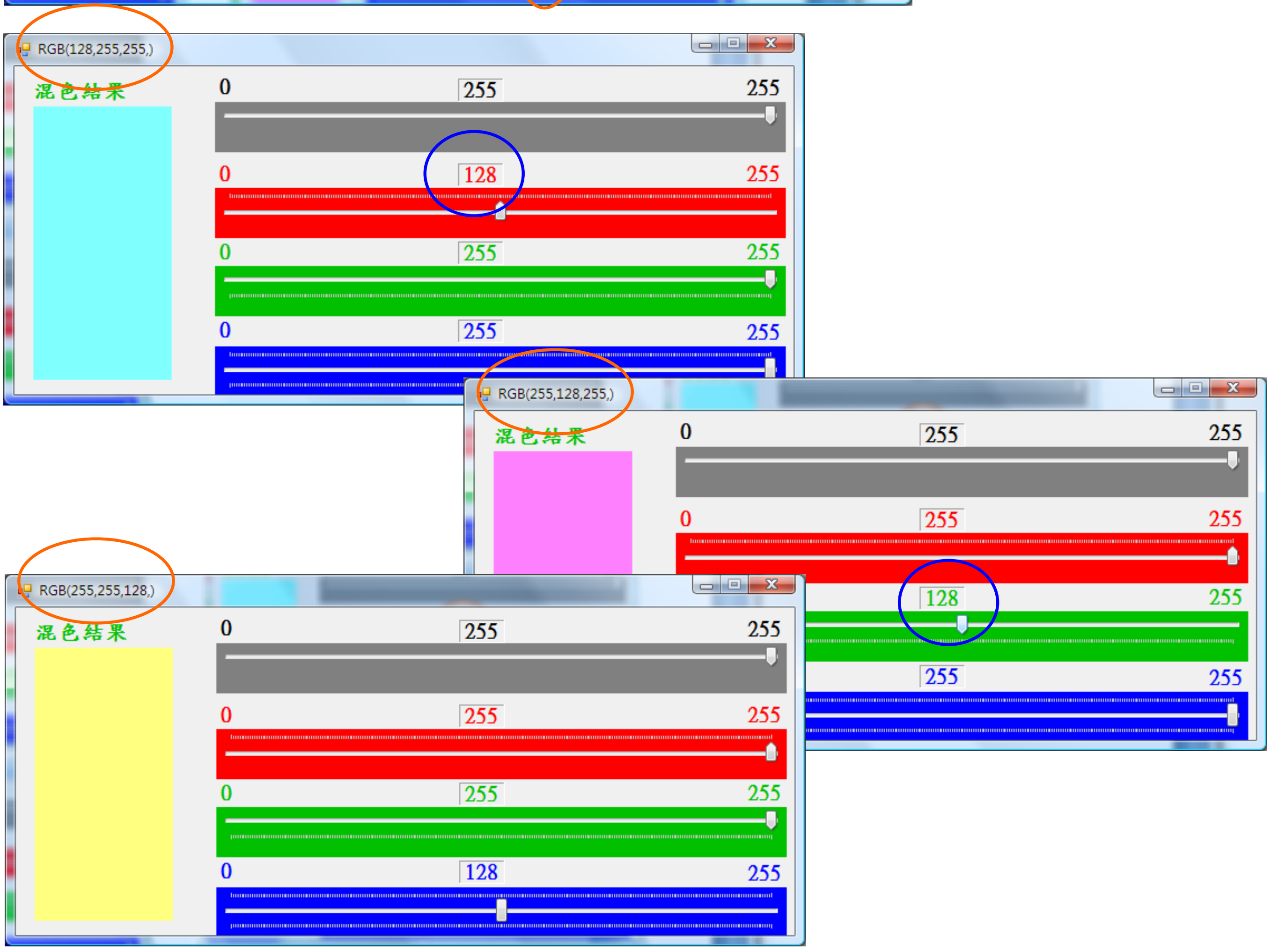
Ex. Color by **Red**、**Green**、**Blue**
設計階段:

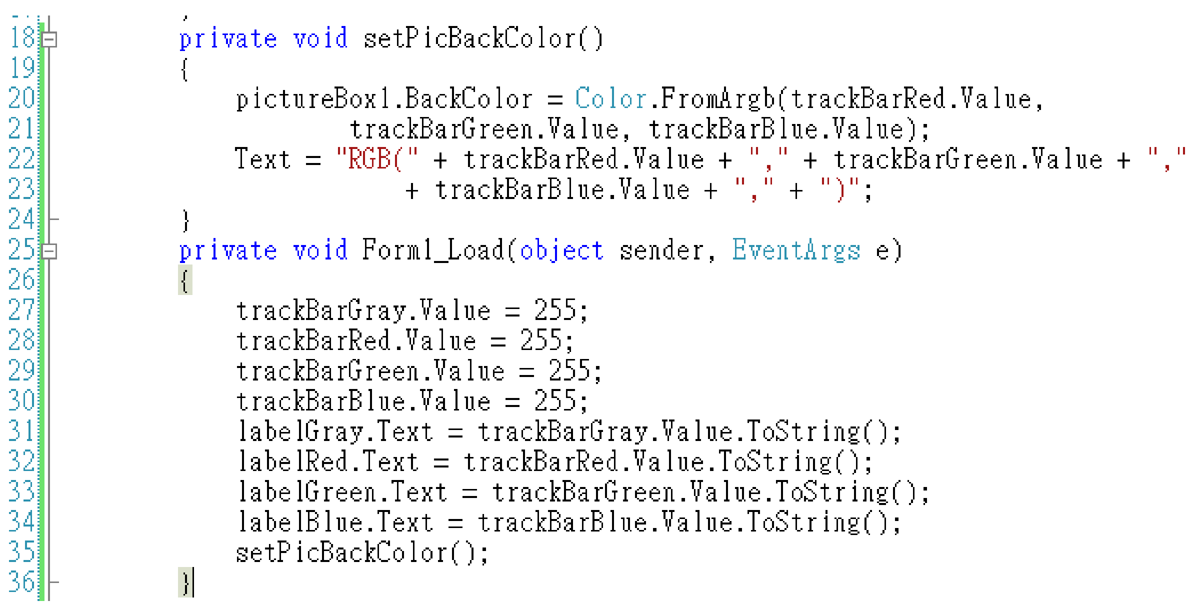


執行階段：

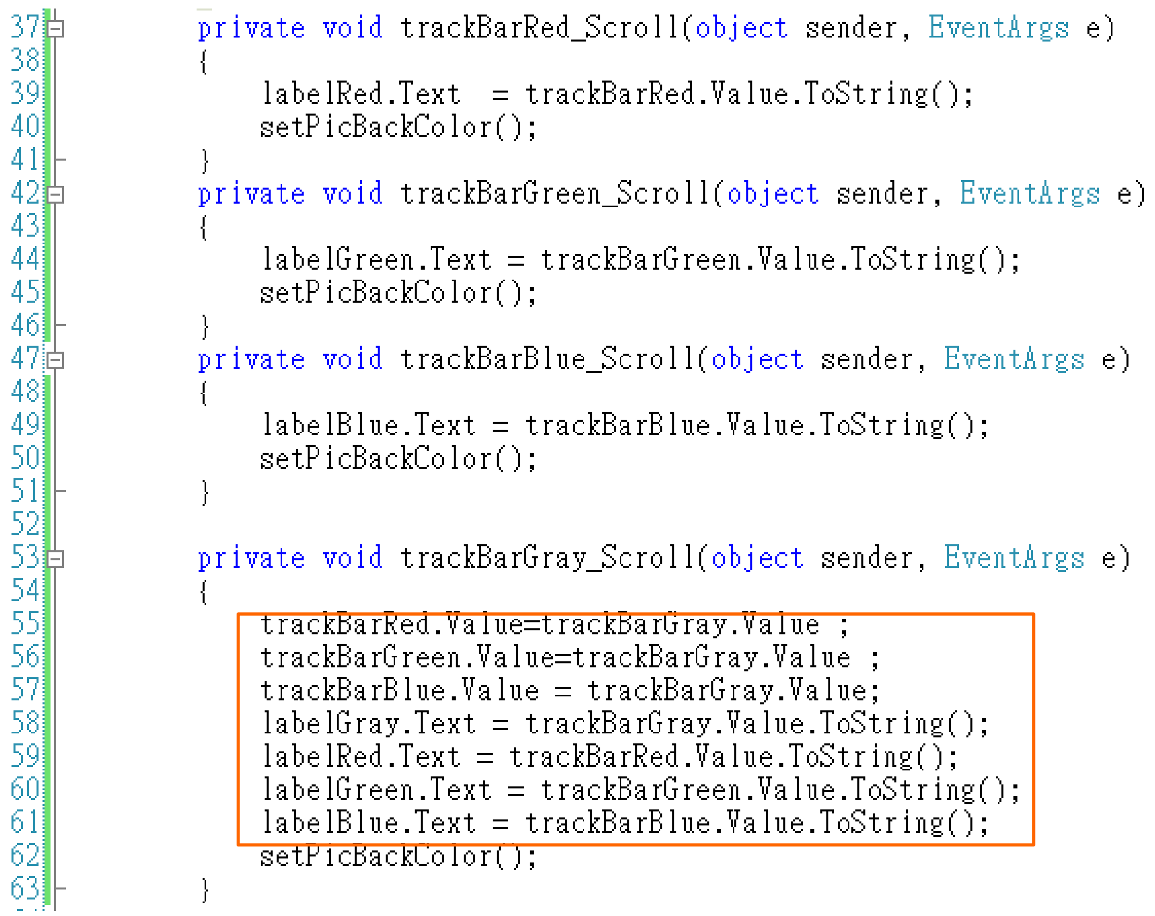

RGB(255,255,255,)
混色結果
0 255 255
0 255 255
0 255 255
0 255 255


RGB(192,192,192,)
混色結果
0 192 255
0 192 255
0 192 255
0 192 255


RGB(128,255,255,)
混色結果
0 255 255
0 128 255
0 255 255
0 255 255
RGB(255,128,255,)
混色結果
0 255 255
0 255 255
128 255
255 255
RGB(255,255,128,)
混色結果
0 255 255
0 255 255
0 255 255
0 128 255



```
        private void setPicBackColor()
        {
            pictureBox1.BackColor = Color.FromArgb(trackBarRed.Value,
                    trackBarGreen.Value, trackBarBlue.Value);
            Text = "RGB(" + trackBarRed.Value + "," + trackBarGreen.Value + ","
                        + trackBarBlue.Value + "," + ")";
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            trackBarGray.Value = 255;
            trackBarRed.Value = 255;
            trackBarGreen.Value = 255;
            trackBarBlue.Value = 255;
            labelGray.Text = trackBarGray.Value.ToString();
            labelRed.Text = trackBarRed.Value.ToString();
            labelGreen.Text = trackBarGreen.Value.ToString();
            labelBlue.Text = trackBarBlue.Value.ToString();
            setPicBackColor();
        }
```



```
        private void trackBarRed_Scroll(object sender, EventArgs e)
        {
            labelRed.Text  = trackBarRed.Value.ToString();
            setPicBackColor();
        }
        private void trackBarGreen_Scroll(object sender, EventArgs e)
        {
            labelGreen.Text = trackBarGreen.Value.ToString();
            setPicBackColor();
        }
        private void trackBarBlue_Scroll(object sender, EventArgs e)
        {
            labelBlue.Text = trackBarBlue.Value.ToString();
            setPicBackColor();
        }

        private void trackBarGray_Scroll(object sender, EventArgs e)
        {
            trackBarRed.Value=trackBarGray.Value ;
            trackBarGreen.Value=trackBarGray.Value ;
            trackBarBlue.Value = trackBarGray.Value;
            labelGray.Text = trackBarGray.Value.ToString();
            labelRed.Text = trackBarRed.Value.ToString();
            labelGreen.Text = trackBarGreen.Value.ToString();
            labelBlue.Text = trackBarBlue.Value.ToString();
            setPicBackColor();
        }
```

Ex.01 Open an image

Step.1 check in install open-cv by conda list

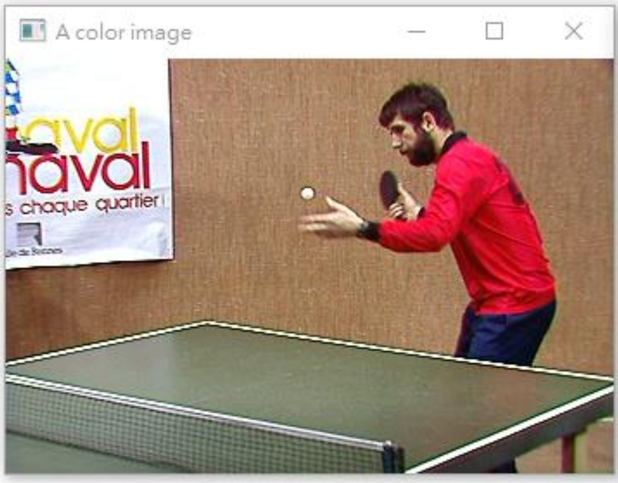
```
opencv-python        4.5.5.62                  pypi_0    pypi
```

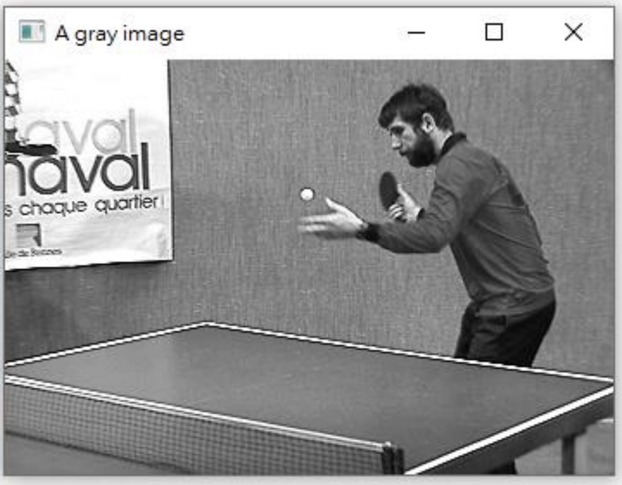
Step.2 install open-cv

```
pip install opencv-python
```

```python
import numpy as np
import cv2

imgPath=r'd:\myCode\MCU_HealthSignal\ImgBank'
imgfn='stennis70.bmp'
fn=imgPath+'\\'+imgfn
img=cv2.imread(fn)
cv2.imshow("A color image", img)
img2=cv2.imread(fn,cv2.IMREAD_GRAYSCALE)
cv2.imshow("A gray image", img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```





img	Array of uint8	(240, 352, 3)	[[[255 247 173]
img2	Array of uint8	(240, 352)	[[226 138 248 ... 94 89 81]

```
(base) C:\Users\Wesley>python
Python 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "C:\Users\Wesley\anaconda3\lib\site-packages\cv2\__init__.py", line 9, in <module>
    from .cv2 import _registerMatType
ImportError: cannot import name '_registerMatType' from 'cv2.cv2' (C:\Users\Wesley\anaconda3\lib\site-packages\cv2\cv2.c
p38-win_amd64.pyd)
```

pip uninstall opencv-python

pip install opencv-python==4.1.2.30

```
(base) C:\Users\Wesley>python
Python 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>>
```

◆特性分佈 (Histogram)

在影像處理分析技術中，經常會運用影像特性的數值資料來作分析應用。影像特性分佈（image histogram）則是其中之一的著名工具。它的意義即一組數字集合的出現頻率分佈圖，因而提供一張數位影像中，有多少像素分佈於不同的亮度區域內。由於它是提供影像對比的形態，所以可提供數位影像品質評估的一種指標。

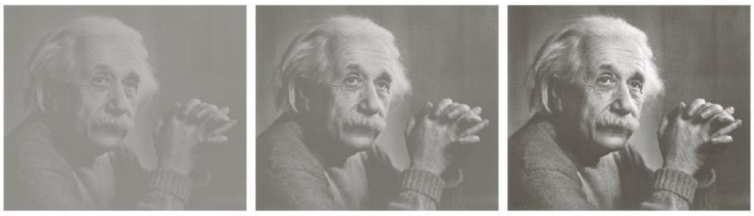
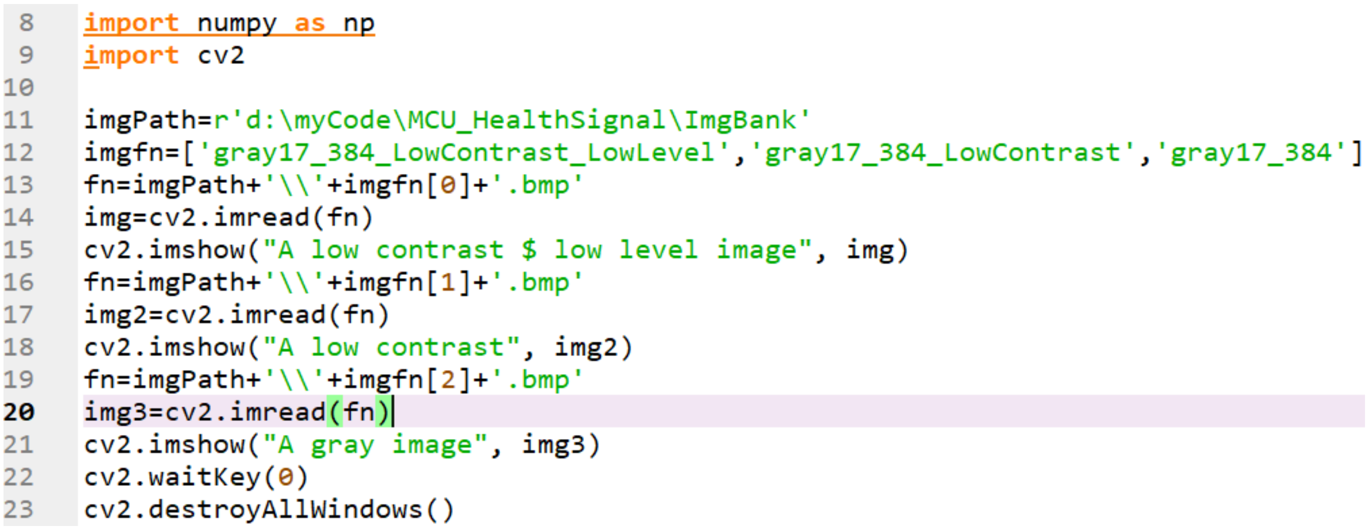


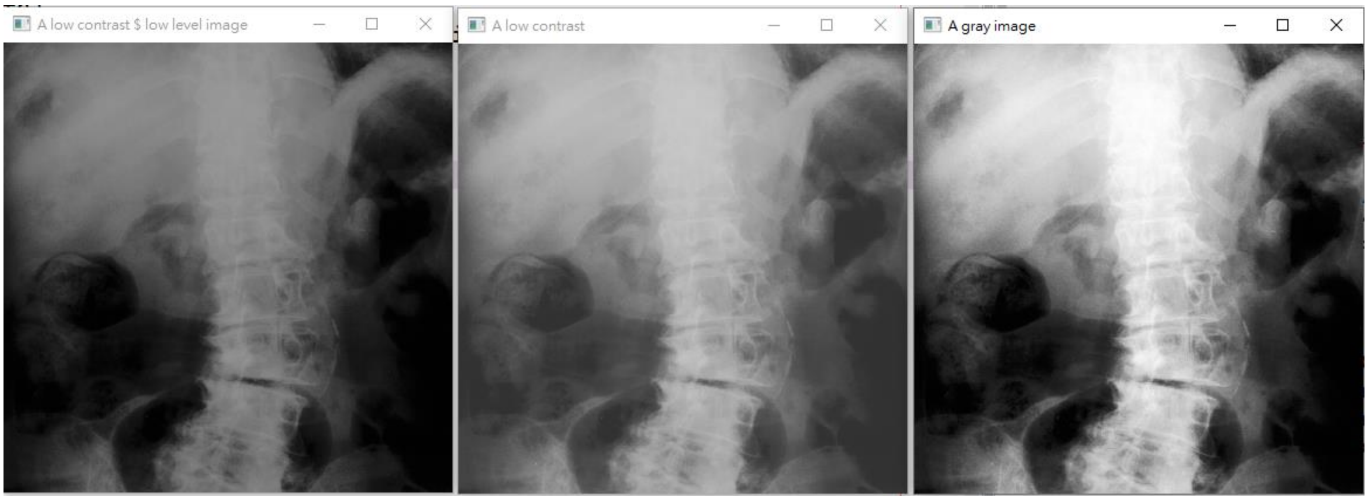
圖 2.6 different contrast images

Ex02



```python
import numpy as np
import cv2

imgPath=r'd:\myCode\MCU_HealthSignal\ImgBank'
imgfn=['gray17_384_LowContrast_LowLevel','gray17_384_LowContrast','gray17_384']
fn=imgPath+'\\'+imgfn[0]+'.bmp'
img=cv2.imread(fn)
cv2.imshow("A low contrast $ low level image", img)
fn=imgPath+'\\'+imgfn[1]+'.bmp'
img2=cv2.imread(fn)
cv2.imshow("A low contrast", img2)
fn=imgPath+'\\'+imgfn[2]+'.bmp'
img3=cv2.imread(fn)
cv2.imshow("A gray image", img3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



3. Image Filtering (影像濾波處理)

將影像的頻率組成(即亮度變化)，予以區分。所謂的頻率組成就是空間頻率。

高頻率：亮度(gray level)變化較大的部分。 ex.眼睛：眼白&瞳孔

低頻率：亮度變化較少的區域。ex.臉頰

運作原理：

Given a mask

ex. 3*3 mask

$$\text{遮罩}\quad\begin{array}{ccc} w1 & w2 & w3 \\ w4 & w5 & w6 \\ w7 & w8 & w9 \end{array}$$

核心 (Kernel)

mask愈大處理的自由度〈彈性〉愈大，但運算時間增加，所以通常使用3*3 mask。

Q：mask如何使用到image上？

A：以w5為中心點對上Image的每一pixel

$$\begin{array}{lll} r(x-1,y-1) & r(x-1,y) & r(x-1.y+1) \\ r(x,y-1) & r(x,y) & r(x.y+1) \\ r(x+1,y-1) & r(x+1,y) & r(x+1.y+1) \end{array}$$

故

$$\begin{aligned} s(x,y)= {} & w1*r(x-1,y-1)+ w2* r(x-1,y) + w3* r(x-1.y+1) \\ & +w4*r(x,y-1) + w5* r(x,y) + w6* r(x.y+1) \\ & +w7*r(x+1,y-1)+ w8* r(x+1,y) + w9* r(x+1.y+1) \end{aligned}$$

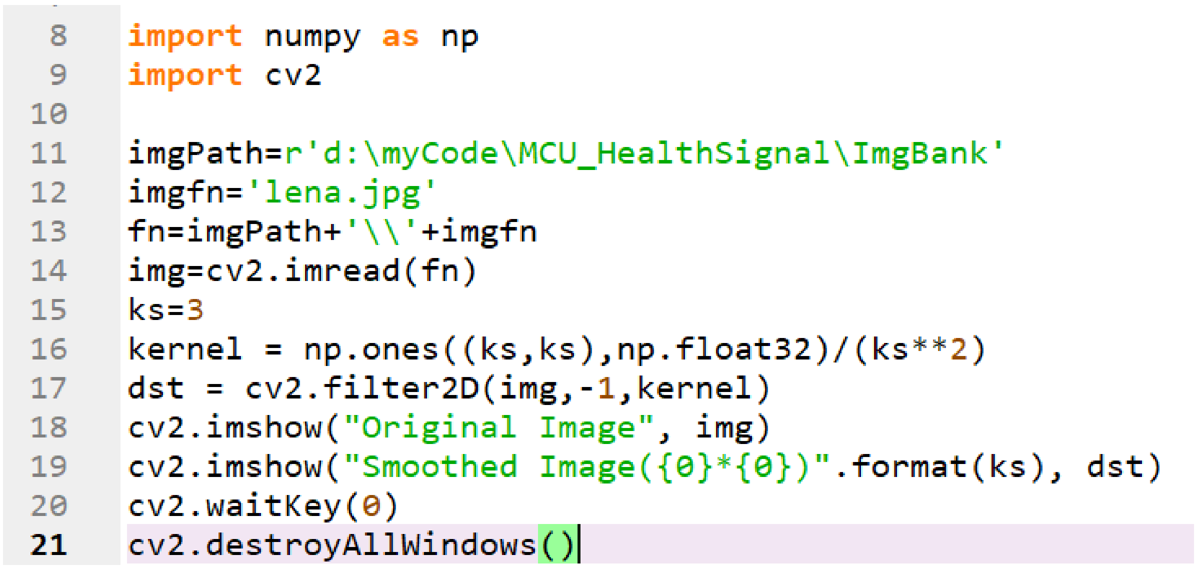
4.3 平滑濾波(smoothing filtering)

低通濾波器(low pass filtering)：它主要用來降低image中的 noise，但會使image模糊。因為強化變化平緩的部份〈低頻部份〉抑制變化較快的成份〈高頻部份〉。

1. mask= 1/9 $\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array}$ 2. mask= 1/10 $\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{array}$

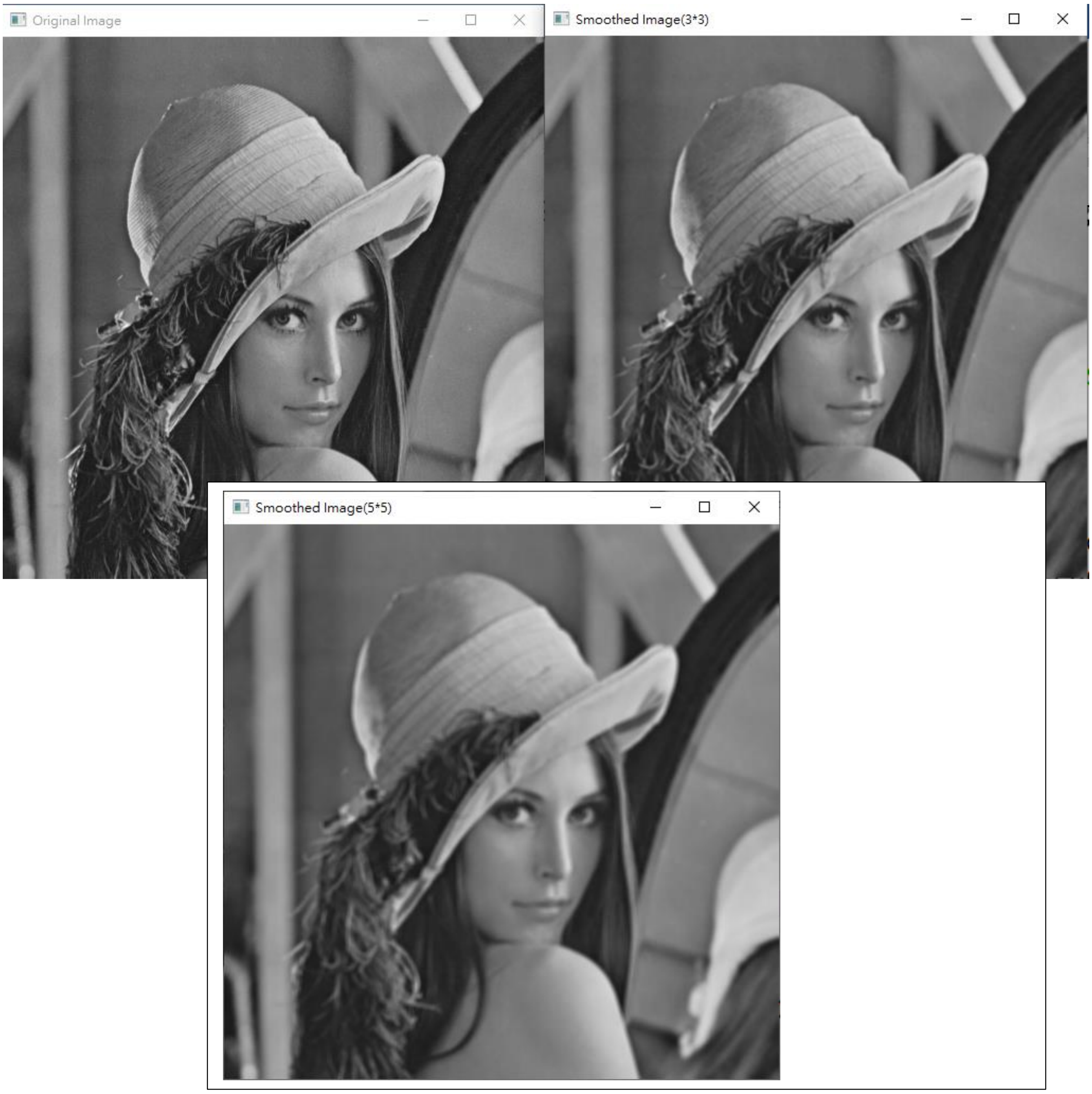
3. mask= 1/16 $\begin{array}{ccc} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{array}$ 4. mask= 1/8 $\begin{array}{ccc} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{array}$

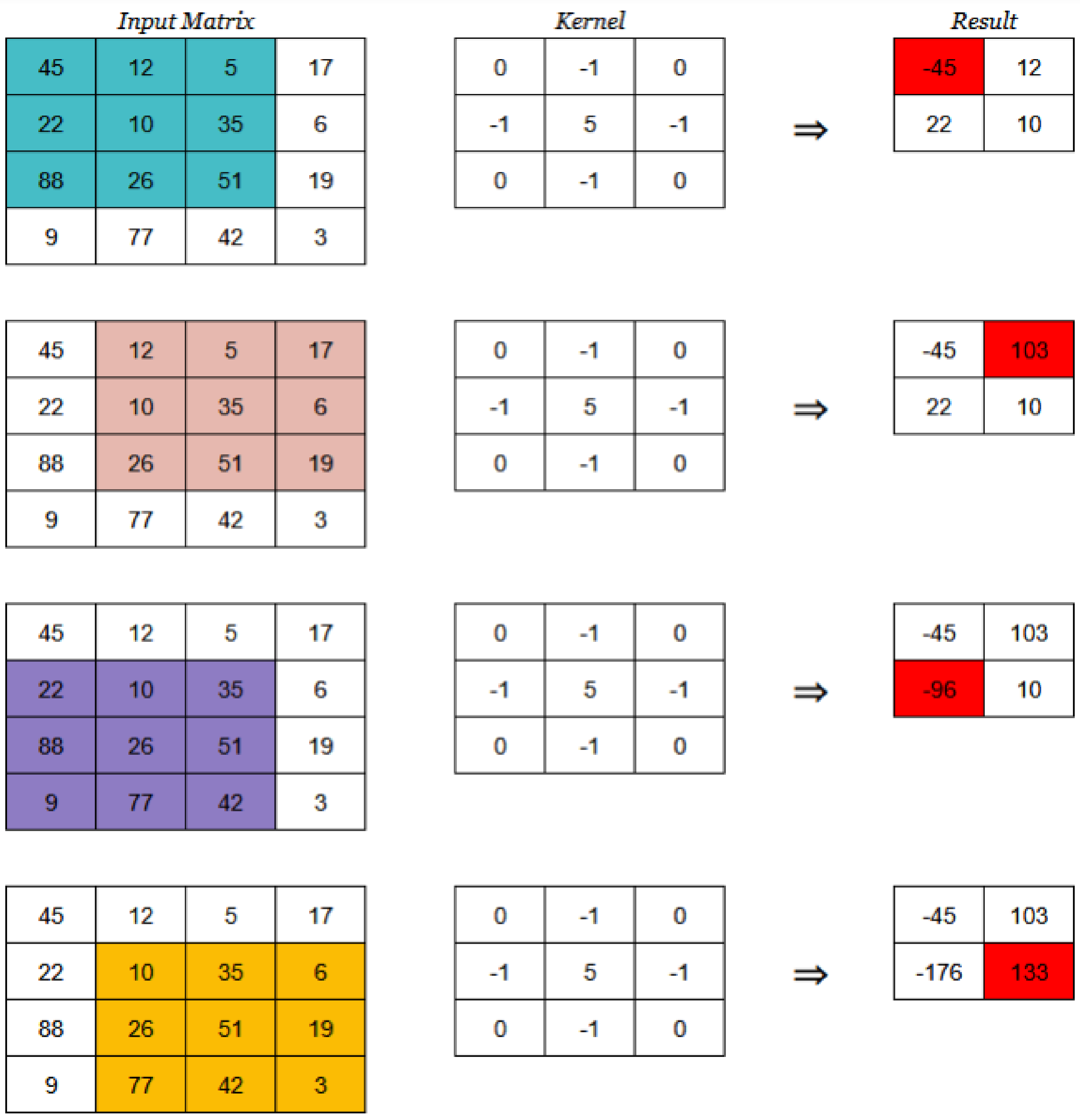
Ex.03 Low-Pass filtering (Smoothing)



```python
import numpy as np
import cv2

imgPath=r'd:\myCode\MCU_HealthSignal\ImgBank'
imgfn='lena.jpg'
fn=imgPath+'\\'+imgfn
img=cv2.imread(fn)
ks=3
kernel = np.ones((ks,ks),np.float32)/(ks**2)
dst = cv2.filter2D(img,-1,kernel)
cv2.imshow("Original Image", img)
cv2.imshow("Smoothed Image({0}*{0})".format(ks), dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```





Input Matrix

45	12	5	17
22	10	35	6
88	26	51	19
9	77	42	3

Kernel

0	-1	0
-1	5	-1
0	-1	0

$\Rightarrow$

Result

-45	12
22	10

45	12	5	17
22	10	35	6
88	26	51	19
9	77	42	3

0	-1	0
-1	5	-1
0	-1	0

$\Rightarrow$

-45	103
22	10

45	12	5	17
22	10	35	6
88	26	51	19
9	77	42	3

0	-1	0
-1	5	-1
0	-1	0

$\Rightarrow$

-45	103
-96	10

45	12	5	17
22	10	35	6
88	26	51	19
9	77	42	3

0	-1	0
-1	5	-1
0	-1	0

$\Rightarrow$

-45	103
-176	133

Convolution in action

4.4 Median filter 中間值濾波

用來除去孤立的雜質(noise)且保持image本身的銳利度(sharpness)。
作法：
若取mask是3*3的大小則其處理的結果是將9個pixel的中間值取出
i.e.9個點中，取gray level為第5大的。

因為　1.取出3*3共9個點　　　ex.　10　20　5
　　　　2.做排序（sorting）　　　　15　**210**　20
　　　　3.取出第5點做輸出　　　　　15　10　210

1. 10 20 5 15 210 20 15 10 210

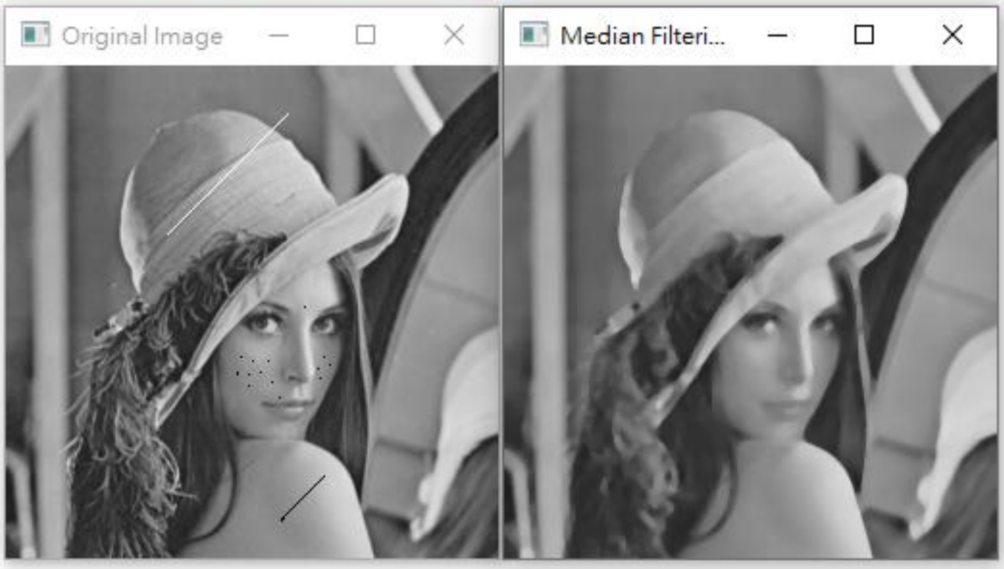
2.sorting:　5　10　10　15　**15**　20　20　210
　　　　　　　　　　　　　　　(中間值)

3.所以 s=15　取代原有210

Ex04.

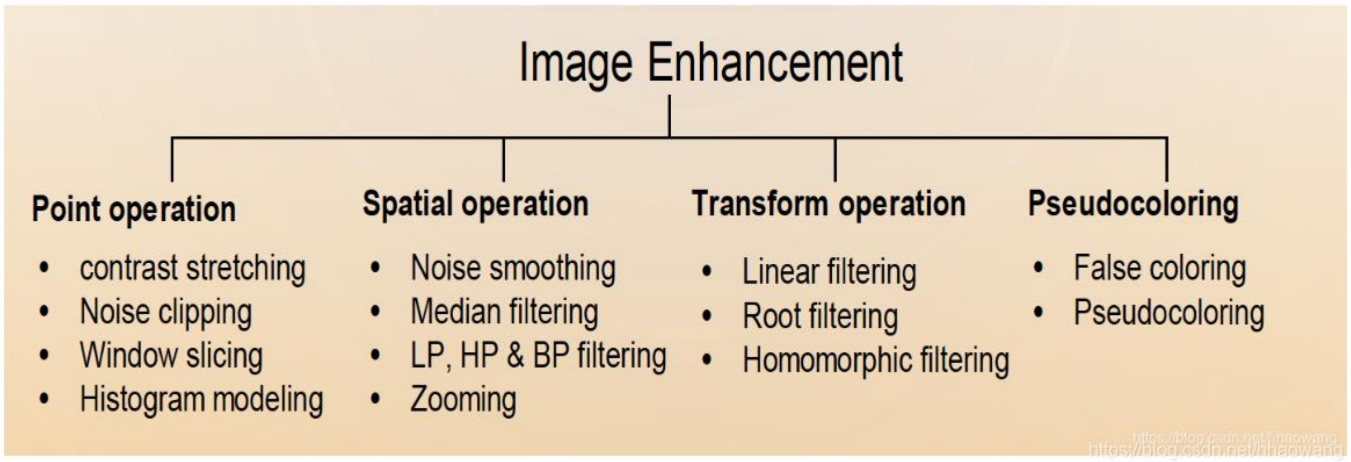
```python
import numpy as np
import cv2

imgPath=r'd:\myCode\MCU_HealthSignal\ImgBank'
imgfn='lena256_noise.bmp'
fn=imgPath+'\\'+imgfn
img=cv2.imread(fn)
dst = cv2.medianBlur(img,5)
cv2.imshow("Original Image", img)
cv2.imshow("Median Filtering",dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



4. Image Enhancement

要找到一種適當的影處理方法,首先要了解輸入影像的性質,而空間領域(spatial domain)影像的性質主要表現在灰度值的變化(variations)與分佈(distribution)。



4.1 長條圖(Histrogram)

長條圖是 image gray level 的機率函數，一般皆以長條柱表示.每一 gray level (r_k)機率的大小，如圖 4.1。

$$P(r_k) = \frac{(\text{gray level} = r_k)\text{的 pixel 數}}{\text{All pixels}}$$

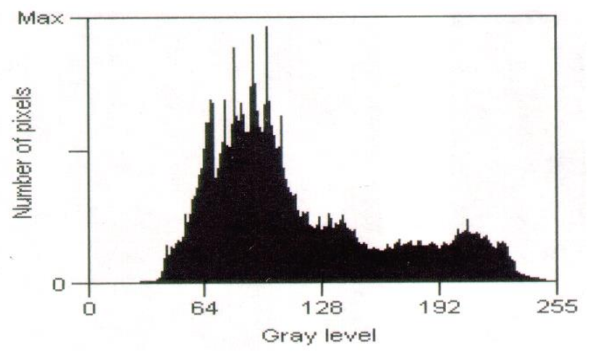
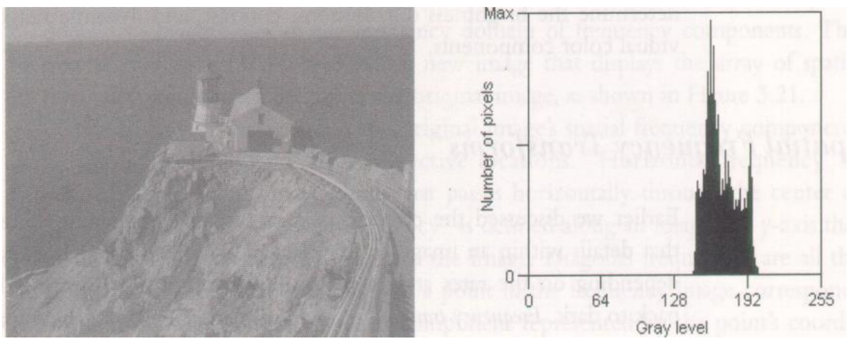


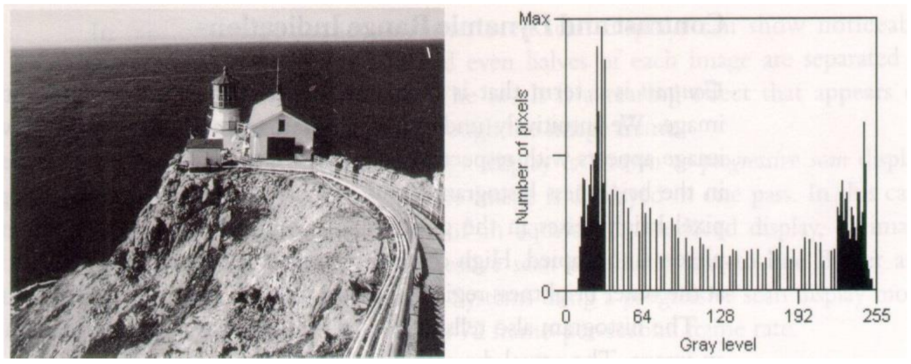
圖 4.1 影像之長條圖。

由長條圖,我們可了解此 image 的灰度值動態範圍,是否偏暗或偏亮？以及對比強弱如何？

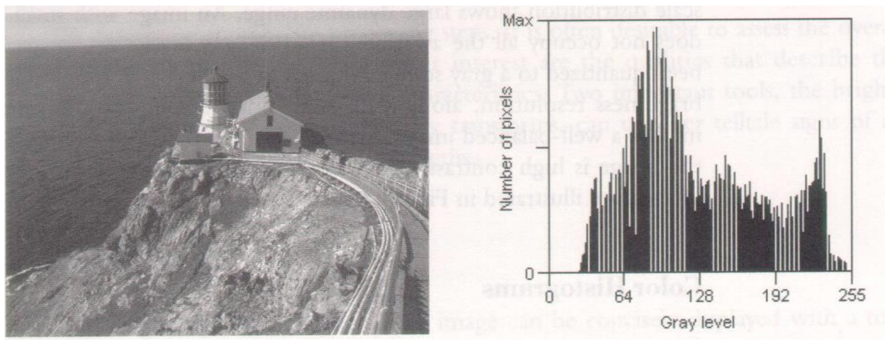
A. Low contrast and low dynamic range image

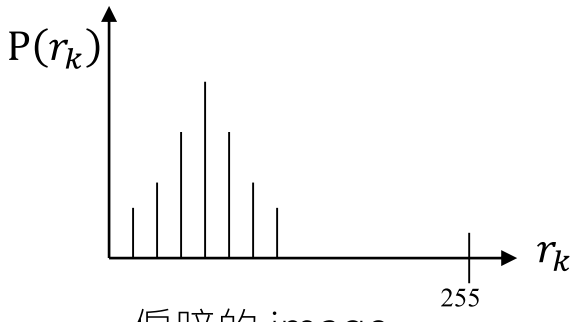


B. High contrast and high dynamic range image

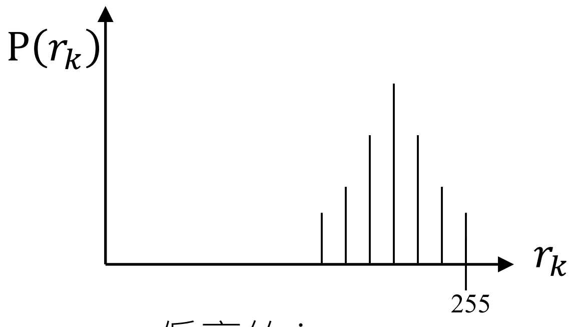


C. Good contrast and high dynamic range image

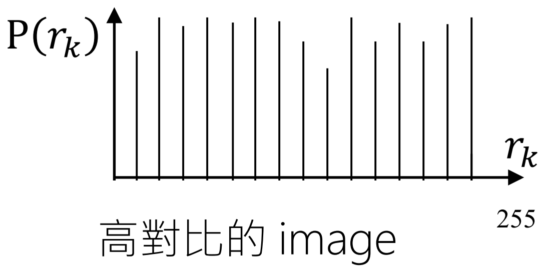




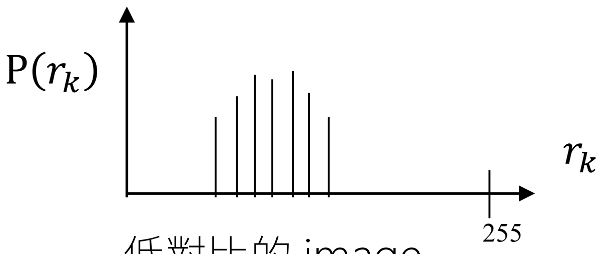
偏暗的 image



偏亮的 image



高對比的 image



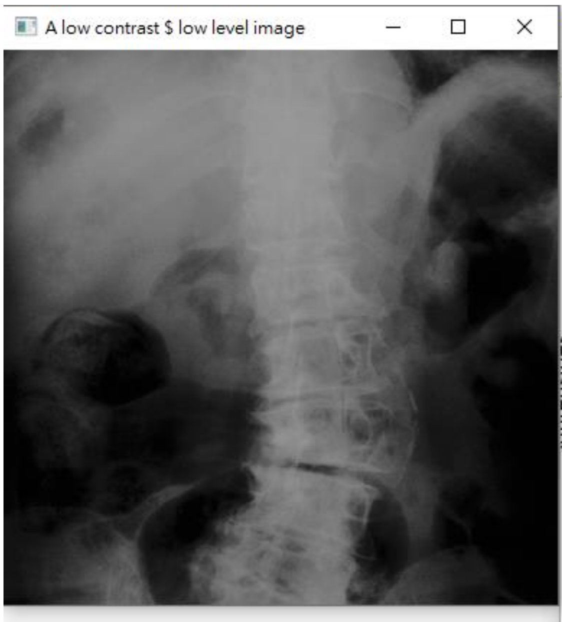
低對比的 image

Ex.05 Display Histogram

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

def showHist(x):
    if x.ndim !=3:
        hist=cv2.calcHist([x],[0], None,[256],[0,256])
        plt.plot(hist)
    else:
        color=('b','g','r')
        for i,col in enumerate(color):
            hist=cv2.calcHist([x],[i], None,[256],[0,256])
            plt.plot(hist,color=col)
        plt.xlim([0,256])
        plt.xlabel('Intensity')
        plt.ylabel('#Intensities')
        plt.show()

imgPath=r'd:\myCode\MCU_HealthSignal\ImgBank'
imgfn=['gray17_384_LowContrast_LowLevel','gray17_384_LowContrast','gray17_384']
fn=imgPath+'\\'+imgfn[0]+'.bmp'
img=cv2.imread(fn)
cv2.imshow("A low contrast $ low level image", img)
showHist(img)
plt.figure()
#
fn=imgPath+'\\'+imgfn[1]+'.bmp'
img2=cv2.imread(fn)
cv2.imshow("A low contrast", img2)
showHist(img2)
plt.figure()
#
fn=imgPath+'\\'+imgfn[2]+'.bmp'
img3=cv2.imread(fn)
cv2.imshow("A gray image", img3)
showHist(img3)
#
cv2.waitKey(0)
cv2.destroyAllWindows()
```


A low contrast $ low level image

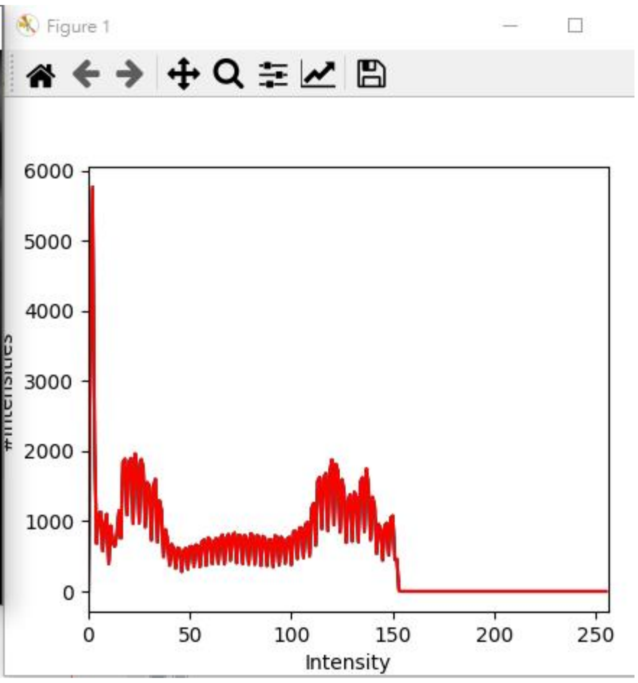
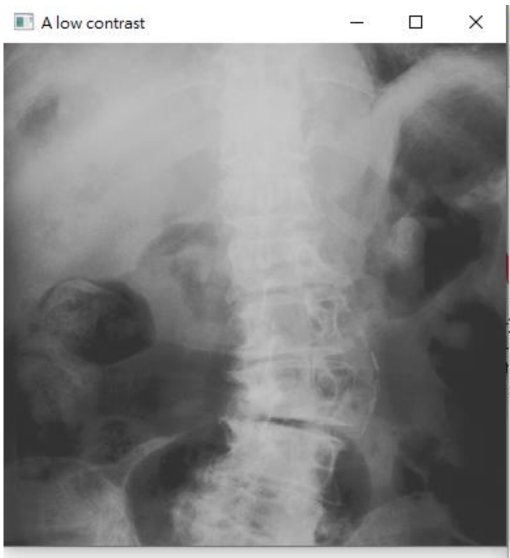


Figure 1
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250
Intensity
#Intensities


A low contrast

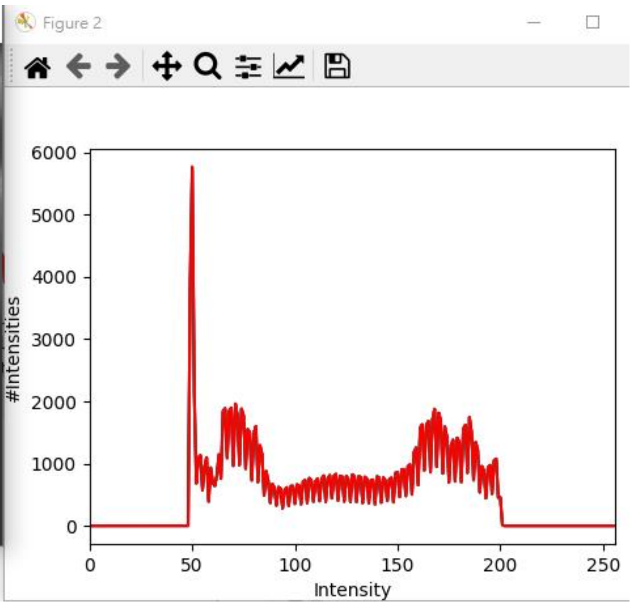
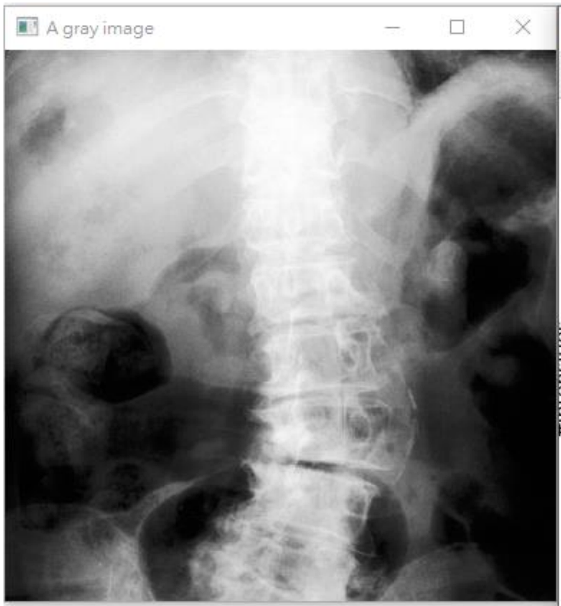


Figure 2
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250
Intensity
#Intensities


A gray image

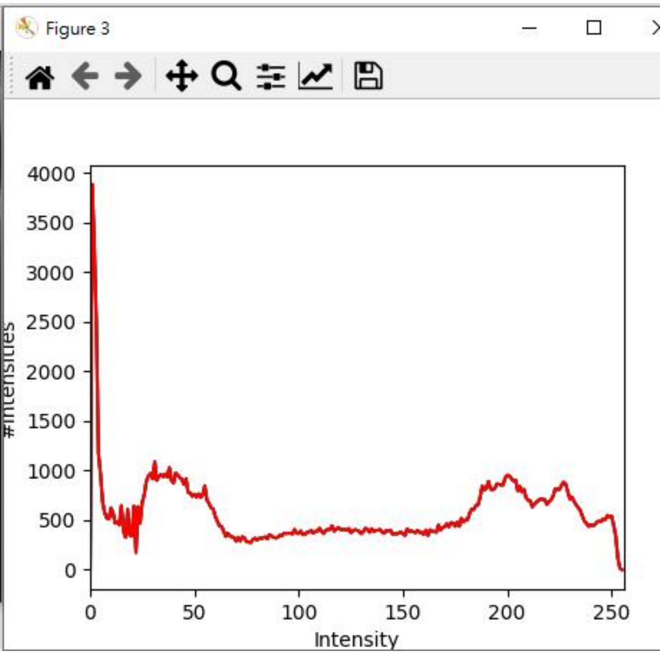
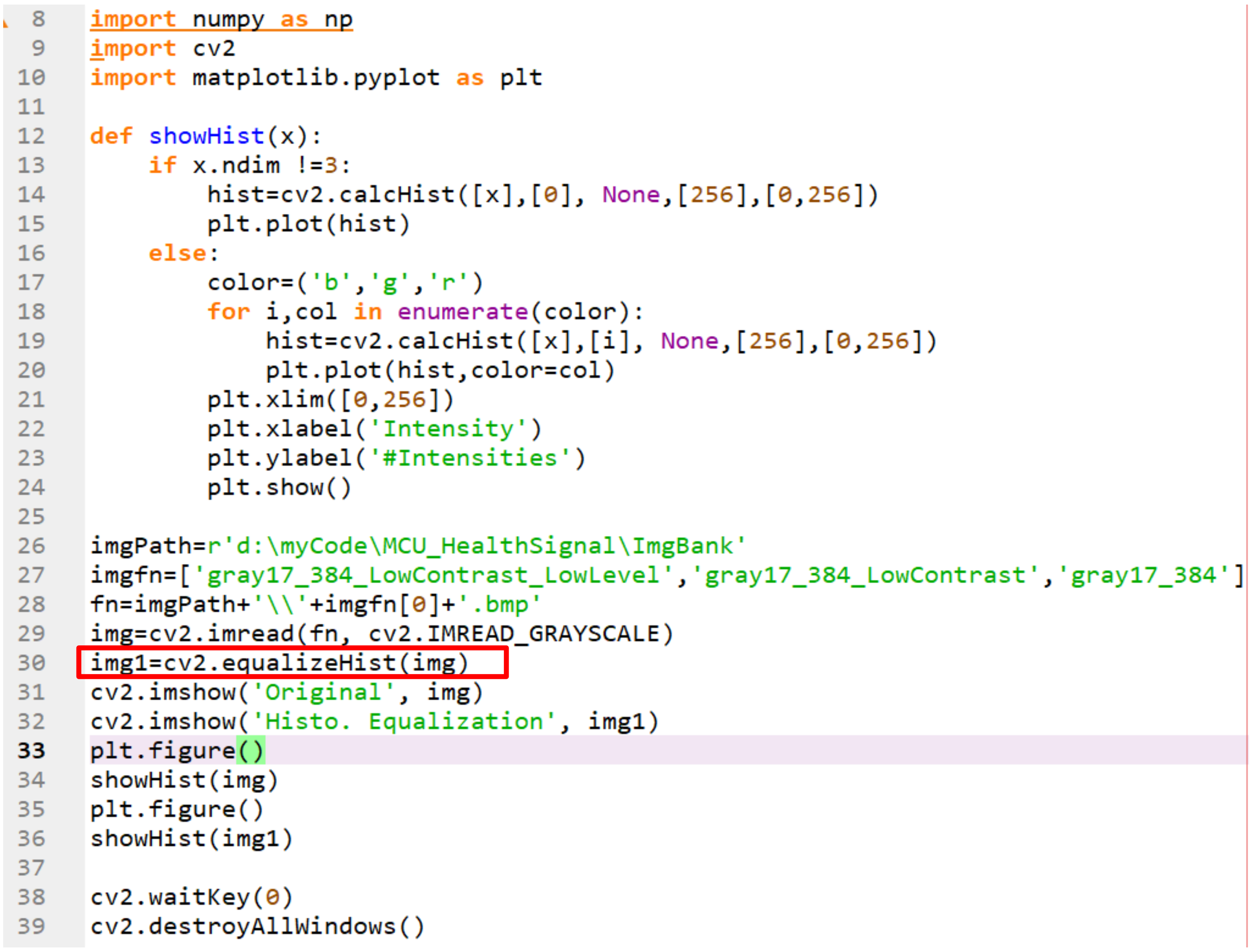


Figure 3
4000
3500
3000
2500
2000
1500
1000
500
0
0
50
100
150
200
250
Intensity
#Intensities

Ex.06 Histogram Equalization

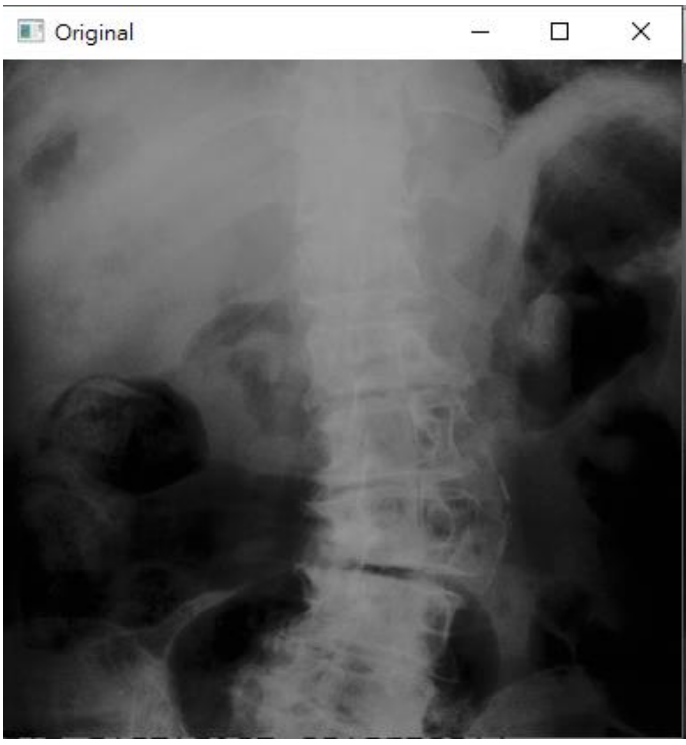


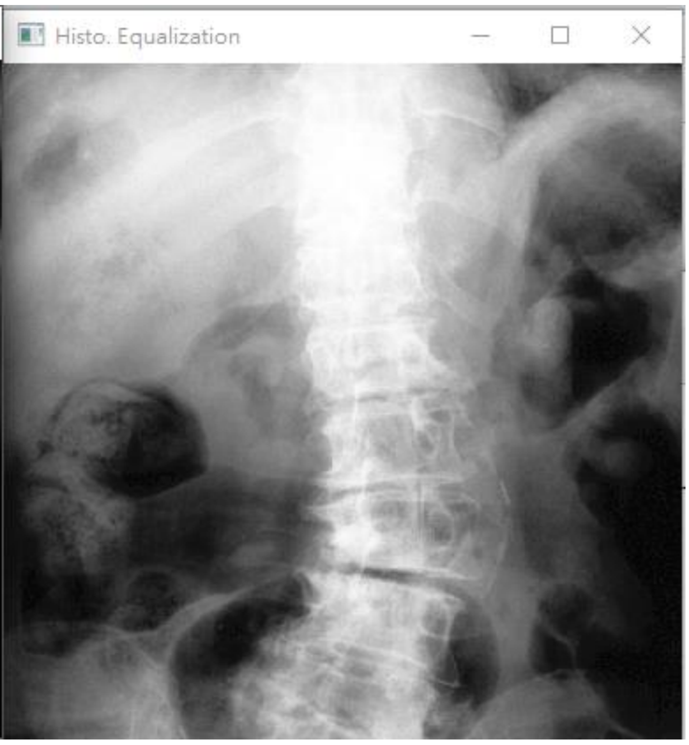
```python
import numpy as np
import cv2
import matplotlib.pyplot as plt

def showHist(x):
    if x.ndim !=3:
        hist=cv2.calcHist([x],[0], None,[256],[0,256])
        plt.plot(hist)
    else:
        color=('b','g','r')
        for i,col in enumerate(color):
            hist=cv2.calcHist([x],[i], None,[256],[0,256])
            plt.plot(hist,color=col)
        plt.xlim([0,256])
        plt.xlabel('Intensity')
        plt.ylabel('#Intensities')
        plt.show()

imgPath=r'd:\myCode\MCU_HealthSignal\ImgBank'
imgfn=['gray17_384_LowContrast_LowLevel','gray17_384_LowContrast','gray17_384']
fn=imgPath+'\\'+imgfn[0]+'.bmp'
img=cv2.imread(fn, cv2.IMREAD_GRAYSCALE)
img1=cv2.equalizeHist(img)
cv2.imshow('Original', img)
cv2.imshow('Histo. Equalization', img1)
plt.figure()
showHist(img)
plt.figure()
showHist(img1)

cv2.waitKey(0)
cv2.destroyAllWindows()
```





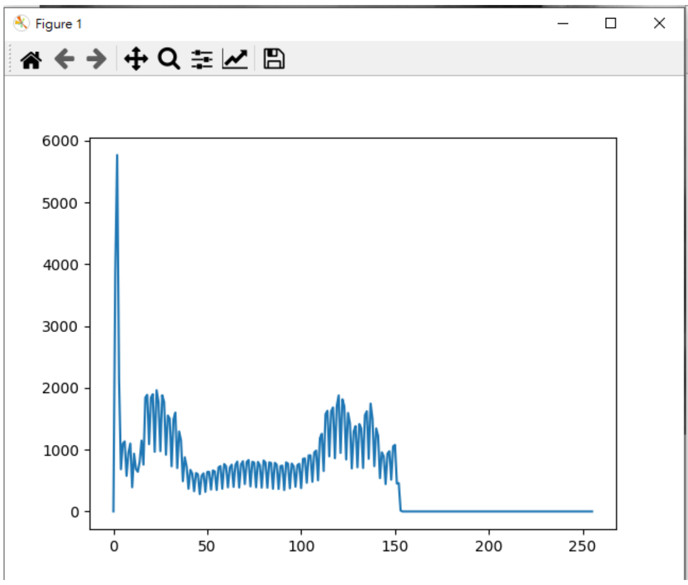


Figure 1
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250

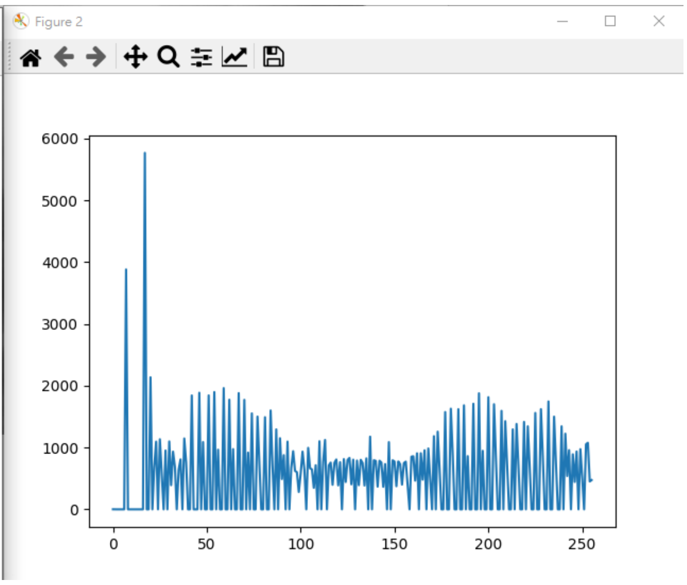


Figure 2
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250

4.2 點運算

(1)函數轉換　　(2)算術邏輯運算

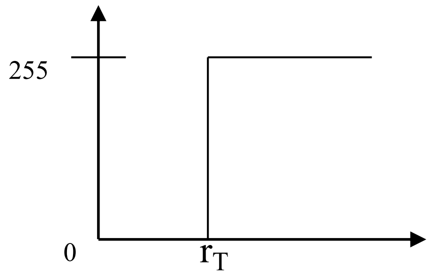
A. 函數轉換

將輸入的 gray level r, 根據函數下映射至 S

S= T(r)

a. 二值化函數 (黑白影像)

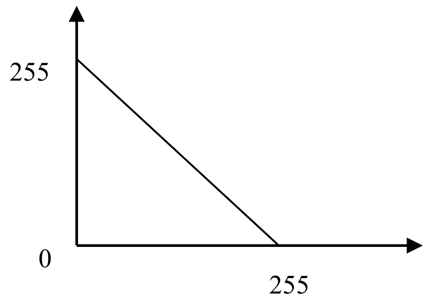
$$T(r) = \begin{cases} 0 & r < r_T \\ 255 & r >= r_T \end{cases}$$



b. 反白函數 (負片效果)

將 image 中的灰度值黑的變白,白的變黑

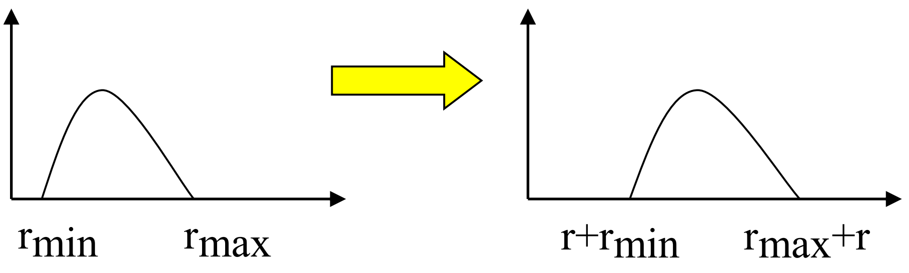
S = T(r) = 255- r



c. 特性的平移

對原 image 作變暗或變亮(對比沒有改變)

S = T (r) = r + r_0



d.影像強化

$$S = T(r) = \frac{r - r_{min}}{r_{max} - r_{min}} * 255$$

改變 image 的對比及動態範圍

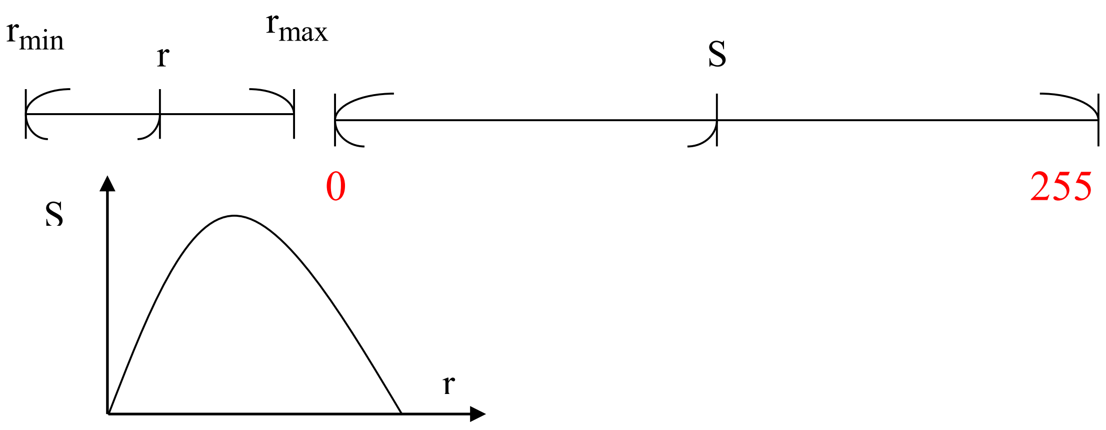


Image Processing

- A general image processing operator is a function that takes one or more input images and produces an output image.
- Image transforms can be seen as:
  - Point operators (pixel transforms)
  - Neighborhood (area-based) operators

Pixel Transforms

- In this kind of image processing transform, each output pixel's value depends on only the corresponding input pixel value (plus, potentially, some globally collected information or parameters).
- Examples of such operators include *brightness and contrast adjustments* as well as color correction and transformations.

Brightness and contrast adjustments

- Two commonly used point processes are *multiplication* and *addition* with a constant:

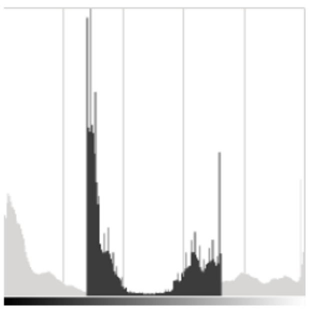
$$g(x) = \alpha f(x) + \beta$$

- The parameters $\alpha > 0$ and β are often called the *gain* and *bias* parameters; sometimes these parameters are said to control *contrast* and *brightness* respectively.
- You can think of $f(x)$ as the source image pixels and $g(x)$ as the output image pixels. Then, more conveniently we can write the expression as:

$$g(i,j) = \alpha \cdot f(i,j) + \beta$$

where i and j indicates that the pixel is located in the *i-th* row and *j-th* column.

The α parameter will modify how the levels spread. If $\alpha < 1$, the color levels will be compressed and the result will be an image with less contrast.



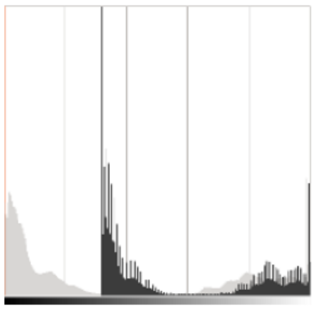
In light gray, histogram of the original image, in dark gray when contrast < 0 in Gimp

Note that these histograms have been obtained using the Brightness-Contrast tool in the Gimp software. The brightness tool should be identical to the β bias parameters but the contrast tool seems to differ to the α gain where the output range seems to be centered with Gimp (as you can notice in the previous histogram).

It can occur that playing with the β bias will improve the brightness but in the same time the image will appear with a slight veil as the contrast will be reduced. The α gain can be used to diminue this effect but due to the saturation, we will lose some details in the original bright regions.

Brightness and contrast adjustments

Increasing (/ decreasing) the β value will add (/ subtract) a constant value to every pixel. Pixel values outside of the [0 ; 255] range will be saturated (i.e. a pixel value higher (/ lesser) than 255 (/ 0) will be clamped to 255 (/ 0)).



In light gray, histogram of the original image, in dark gray when brightness = 80 in Gimp

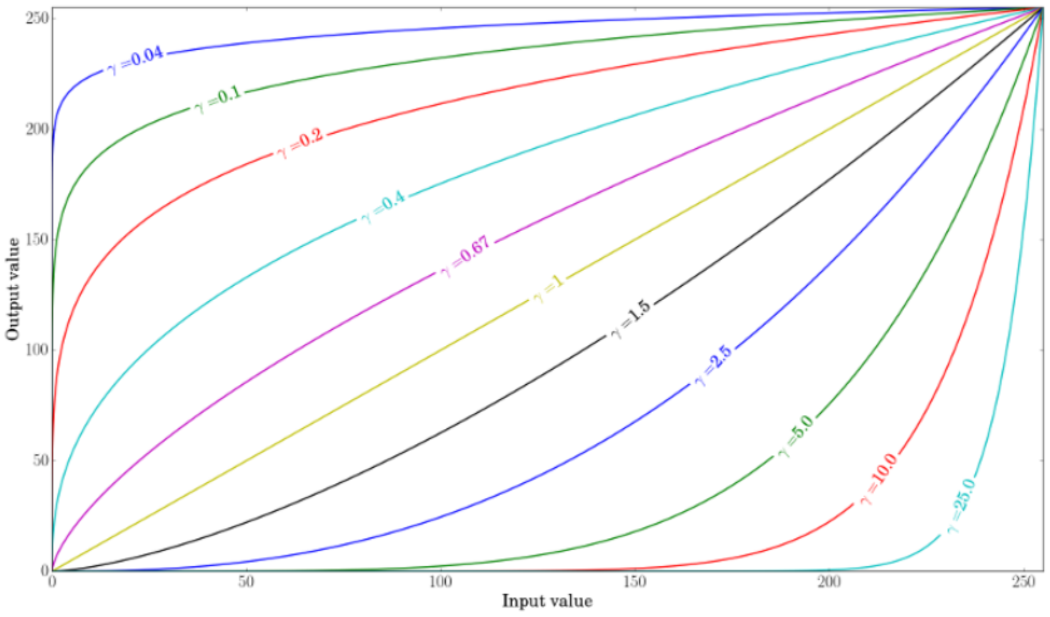
The histogram represents for each color level the number of pixels with that color level. A dark image will have many pixels with low color value and thus the histogram will present a peak in its left part. When adding a constant bias, the histogram is shifted to the right as we have added a constant bias to all the pixels.

Gamma correction

Gamma correction can be used to correct the brightness of an image by using a non linear transformation between the input values and the mapped output values:

$$O = \left(\frac{I}{255} \right)^{\gamma} \times 255$$

As this relation is non linear, the effect will not be the same for all the pixels and will depend to their original value.

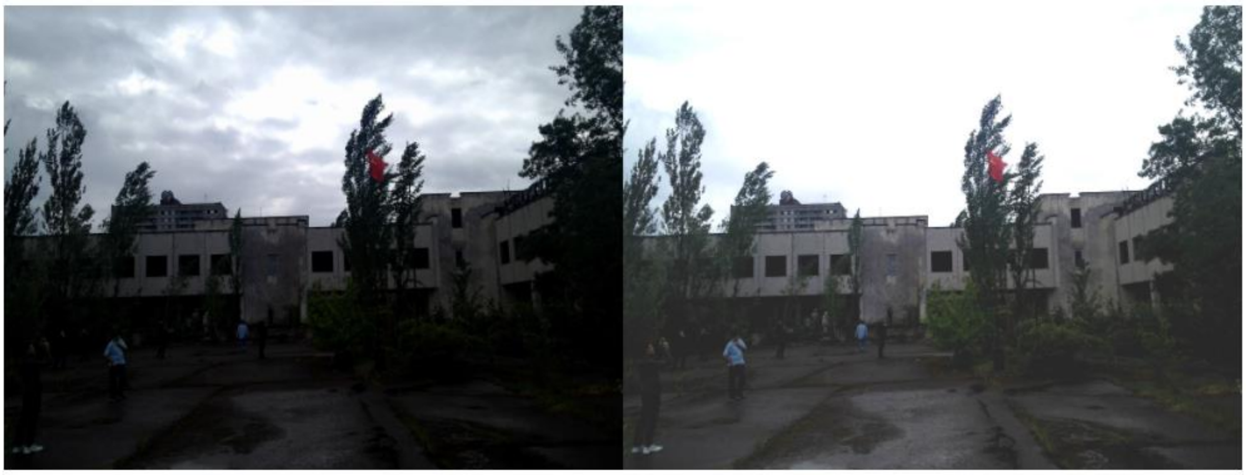


Plot for different values of gamma

When $\gamma < 1$, the original dark regions will be brighter and the histogram will be shifted to the right whereas it will be the opposite with $\gamma > 1$.

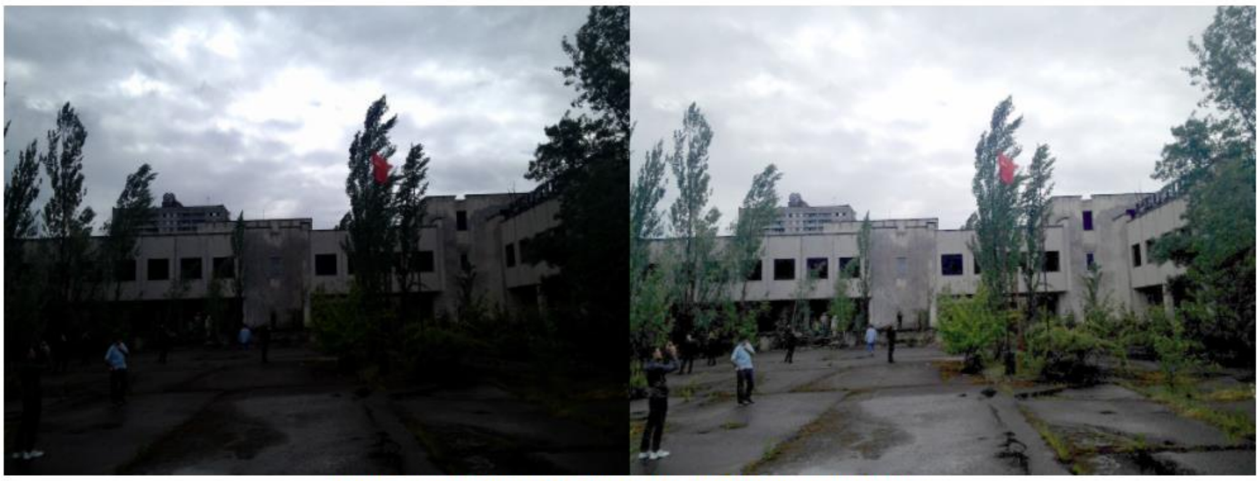
Correct an underexposed image

The following image has been corrected with: $\alpha = 1.3$ and $\beta = 40$.


By Visem (Own work) [CC BY-SA 3.0], via Wikimedia Commons

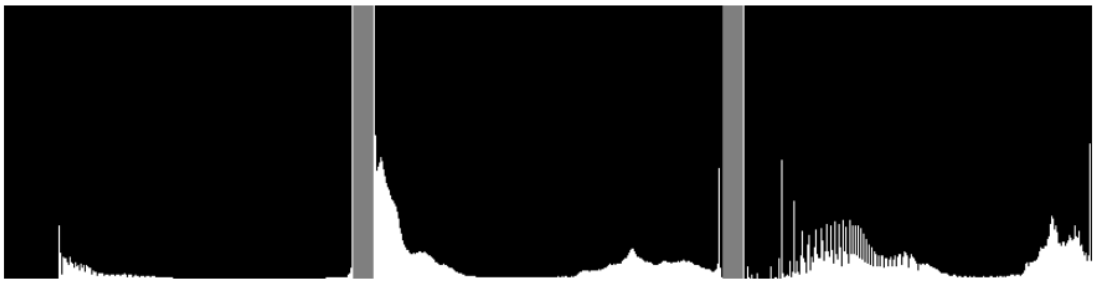
The overall brightness has been improved but you can notice that the clouds are now greatly saturated due to the numerical saturation of the implementation used (highlight clipping in photography).

The following image has been corrected with: $\gamma = 0.4$.


By Visem (Own work) [CC BY-SA 3.0], via Wikimedia Commons

The gamma correction should tend to add less saturation effect as the mapping is non linear and there is no numerical saturation possible as in the previous method.

The gamma correction should tend to add less saturation effect as the mapping is non linear and there is no numerical saturation possible as in the previous method.


Left: histogram after alpha, beta correction ; Center: histogram of the original image ; Right: histogram after the gamma correction

The previous figure compares the histograms for the three images (the y-ranges are not the same between the three histograms). You can notice that most of the pixel values are in the lower part of the histogram for the original image. After α, β correction, we can observe a big peak at 255 due to the saturation as well as a shift in the right. After gamma correction, the histogram is shifted to the right but the pixels in the dark regions are more shifted (see the gamma curves figure) than those in the bright regions.

In this tutorial, you have seen two simple methods to adjust the contrast and the brightness of an image. **They are basic techniques and are not intended to be used as a replacement of a raster graphics editor!**

Ex.07 Enhancement

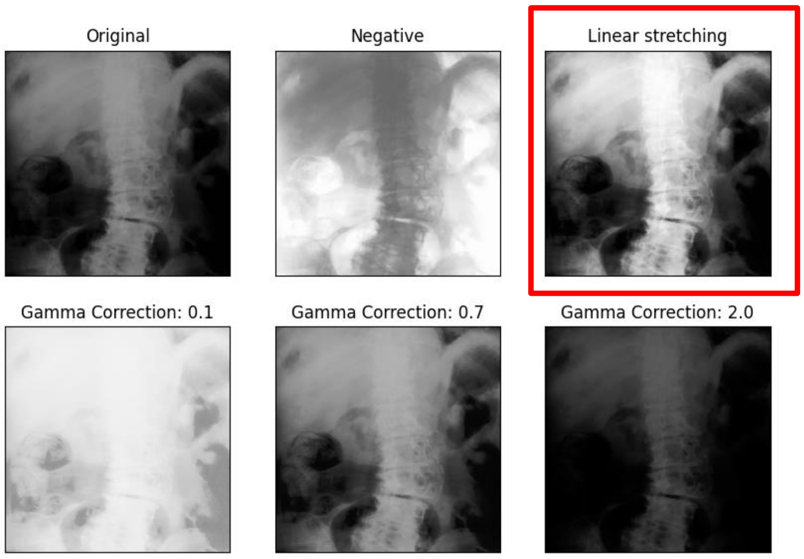
```python
import numpy as np
import cv2
import matplotlib.pyplot as plt

def GammaCorrection(f,gamma=2.0):
    g=f.copy()
    nr,nc=f.shape[:2]
    c=255.0/(255.0**gamma)
    table=np.zeros(256)
    for i in range(256):
        table[i]=round(i**gamma*c,0)
    if f.ndim !=3:#gray
        for y in range(nr):
            for x in range(nc):
                g[y,x]=table[f[y,x]]
    else:
        for y in range(nr):
            for x in range(nc):
                for k in range(3):
                    g[y,x,k]=table[f[y,x,k]]
    return g



imgPath=r'd:\myCode\MCU_HealthSignal\ImgBank'
imgfns=['WalkingMan.bmp','gray17_384_LowContrast_LowLevel.bmp','barbara.bmp','gray17_384.bmp']
fn=imgPath+'\\'+imgfns[1]
img=cv2.imread(fn)
```

```python
img2=255-img;
imax=np.max(img)
imin=np.min(img)
img3=np.uint8((img-imin)*255.0/(imax-imin))
img4=GammaCorrection(img,0.1)
img5=GammaCorrection(img,0.7)
img6=GammaCorrection(img,2.0)
laplacian = cv2.Laplacian(img,cv2.CV_64F)
sobelx = cv2.Sobel(img,cv2.CV_64F,1,0,ksize=3)
sobely = cv2.Sobel(img,cv2.CV_64F,0,1,ksize=3)
plt.subplot(2,3,1),plt.imshow(img,cmap = 'gray')
plt.title('Original'), plt.xticks([]), plt.yticks([])
plt.subplot(2,3,2),plt.imshow(img2,cmap = 'gray')
plt.title('Negative'), plt.xticks([]), plt.yticks([])
plt.subplot(2,3,3),plt.imshow(img3,cmap = 'gray')
plt.title('Linear stretching'), plt.xticks([]), plt.yticks([])
plt.subplot(2,3,4),plt.imshow(img4,cmap = 'gray')
plt.title('Gamma Correction: 0.1'), plt.xticks([]), plt.yticks([])
plt.subplot(2,3,5),plt.imshow(img5,cmap = 'gray')
plt.title('Gamma Correction: 0.7'), plt.xticks([]), plt.yticks([])
plt.subplot(2,3,6),plt.imshow(img6,cmap = 'gray')
plt.title('Gamma Correction: 2.0'), plt.xticks([]), plt.yticks([])
plt.show()
```



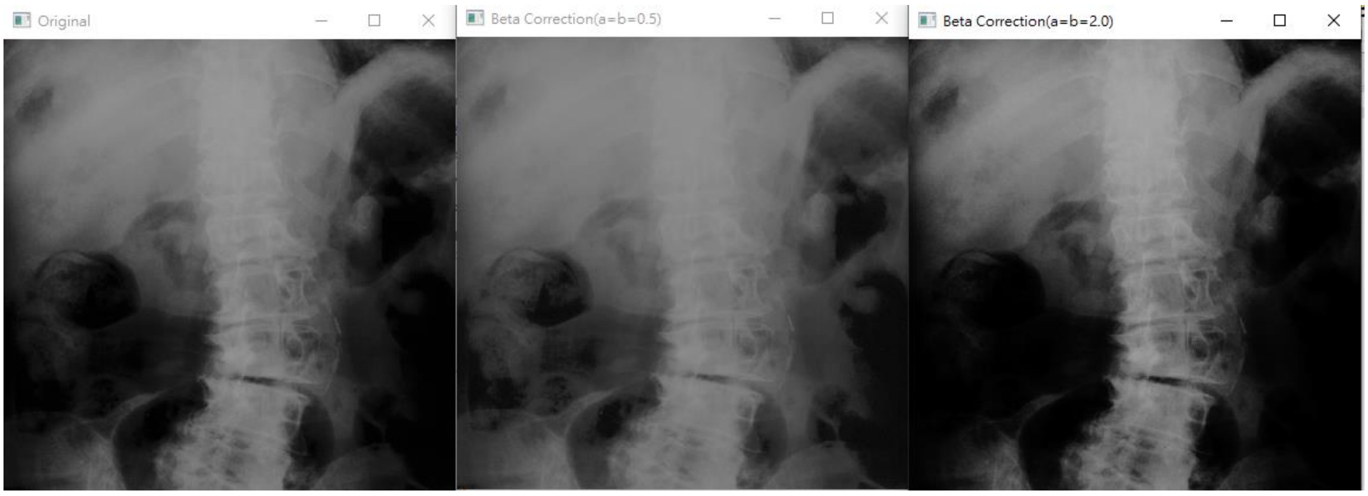
Ex.08 Beta Correction

```python
import numpy as np
import cv2
import scipy.special as special

def betaCorrection(f, a=2.0,b=2.0):
    g=f.copy()
    nr,nc=f.shape[:2]
    x=np.linspace(0,1,256)
    table=np.round(special.betainc(a,b,x)*255,0)
    if f.ndim !=3:#gray
        for y in range(nr):
            for x in range(nc):
                g[y,x]=table[f[y,x]]
    else:
        for y in range(nr):
            for x in range(nc):
                for k in range(3):
                    g[y,x,k]=table[f[y,x,k]]
    return g

imgPath=r'd:\myCode\MCU_HealthSignal\ImgBank'
imgfn=['gray17_384_LowContrast_LowLevel','gray17_384_LowContrast','gray17_384']
fn=imgPath+'\\'+imgfn[0]+'.bmp'
img=cv2.imread(fn, cv2.IMREAD_GRAYSCALE)
img1=betaCorrection(img, a=0.5,b=0.5)
img2=betaCorrection(img, a=2.0,b=2.0)
cv2.imshow('Original', img)
cv2.imshow('Beta Correction(a=b=0.5)', img1)
cv2.imshow('Beta Correction(a=b=2.0)', img2)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



4.3 邊緣偵測

Image gradients

$$\nabla f = \begin{bmatrix} f_x \\ f_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$\text{Magnitude } M(x,y) = \sqrt{f_x^2 + f_y^2}$$

$$\text{Or } M(x,y) \approx |f_x| + |f_y|$$

$$\text{Direction } \theta = tan^{-1}\left(\frac{f_y}{f_x}\right)$$

Gradient operators:

Sobel

1	0	-1
2	0	-2
1	0	-1

1	2	1
0	0	0
-1	-2	-1

Scharr

3	0	-3
10	0	-10
3	0	-3

3	10	3
0	0	0
-3	-10	-3

Prewitt

1	0	-1
1	0	-1
1	0	-1

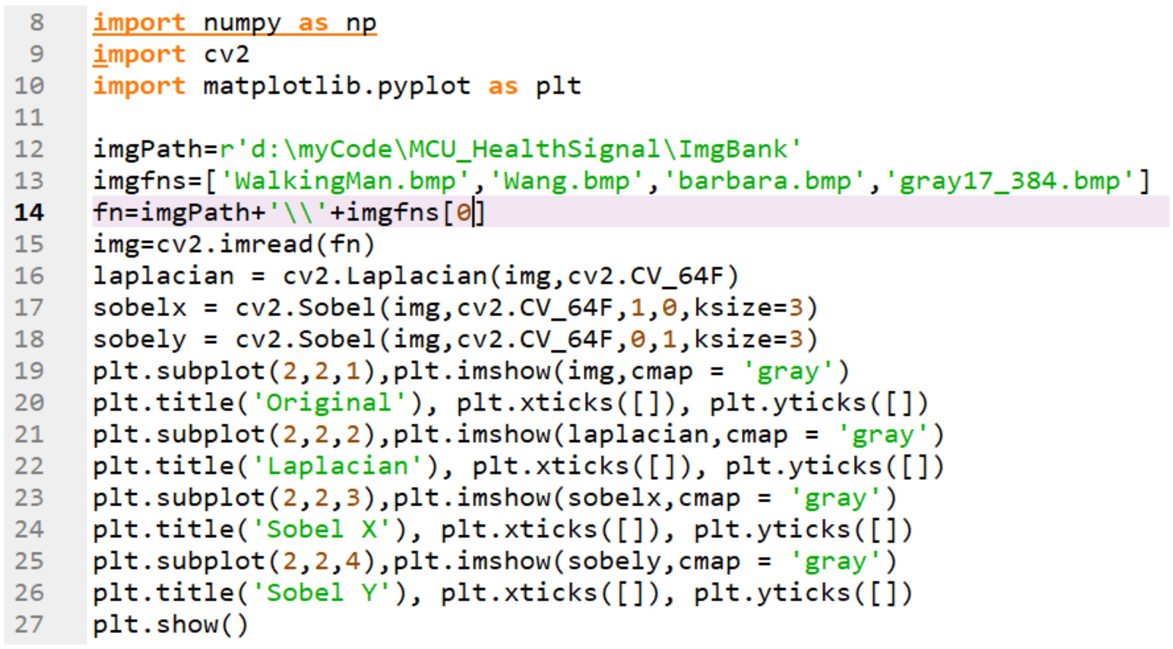
1	1	1
0	0	0
-1	-1	-1

Roberts

0	1
-1	0

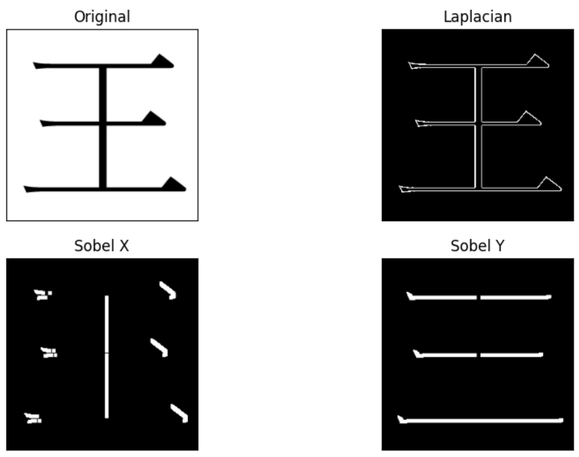
1	0
0	-1

Ex.09Edge Detection



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

imgPath=r'd:\myCode\MCU_HealthSignal\ImgBank'
imgfns=['WalkingMan.bmp','Wang.bmp','barbara.bmp','gray17_384.bmp']
fn=imgPath+'\\'+imgfns[0]
img=cv2.imread(fn)
laplacian = cv2.Laplacian(img,cv2.CV_64F)
sobelx = cv2.Sobel(img,cv2.CV_64F,1,0,ksize=3)
sobely = cv2.Sobel(img,cv2.CV_64F,0,1,ksize=3)
plt.subplot(2,2,1),plt.imshow(img,cmap = 'gray')
plt.title('Original'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,2),plt.imshow(laplacian,cmap = 'gray')
plt.title('Laplacian'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,3),plt.imshow(sobelx,cmap = 'gray')
plt.title('Sobel X'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,4),plt.imshow(sobely,cmap = 'gray')
plt.title('Sobel Y'), plt.xticks([]), plt.yticks([])
plt.show()
```


Original
Laplacian
Sobel X
Sobel Y