Principle and Applications of Digital Image Processing

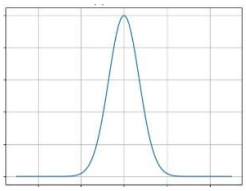
Homework 2 Report 林東甫 R12631055

Part 1: (30%)

2.5:
Ans:
(a) 2048/50=40.96 line pairs per mm
(b) 2048/2=1024 dpi
2.12:

Ans:考慮該式為自然對數 e 的負數次方，則該自然對數將會在 x=x0 及 y=y0 時來到最大 e^0=1，如下圖所示。

已知人眼可觀測 2^3 層強度變化，256=2^8，8-3=5 因此 k<5。



2.18
Ans:
(a) V={0,1},

4-path:不存在,如果我們從 p 開始,那我們可以先選繼續往右或往上移動,若是往上,則明顯沒有路可以走,若是往右則我們只能選擇先往右到 1 再往上到 1,這時又有往上和往右可以選擇,往右這次也是死路,因此只能往上到 0,這裡也沒有路可以到 q,因此 4-path 從 p 到 q 不存在.

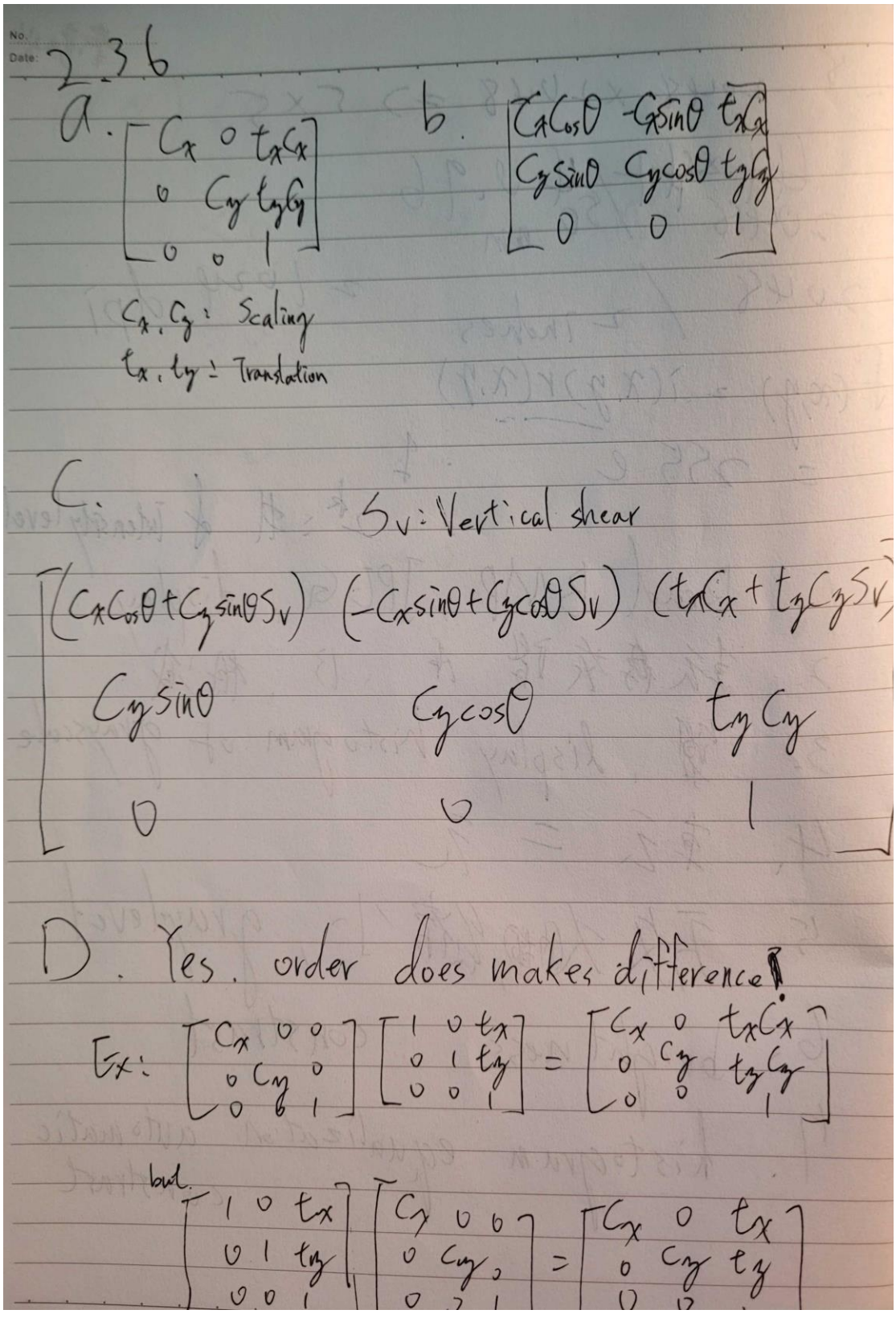
8-path=4,實際做法與上述方法相似。

m-path=5

(b) V={1,2}
4-path=6 , 8-path=4 , m-path=6.

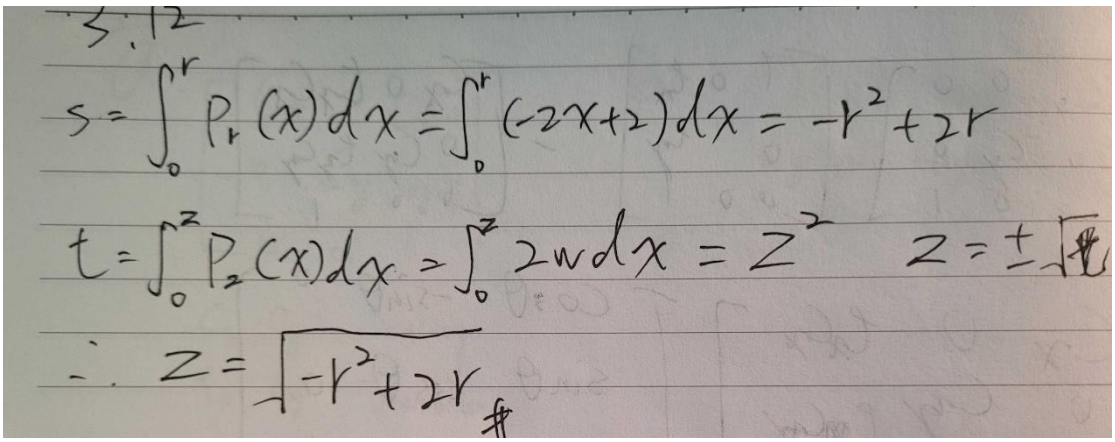
2.36:

Ans:


No.
Date:
2.36
a.
$\begin{bmatrix} C_x & 0 & t_x C_x \\ 0 & C_y & t_y C_y \\ 0 & 0 & 1 \end{bmatrix}$
b.
$\begin{bmatrix} C_x\cos\theta & -C_x\sin\theta & t_x C_x \\ C_y\sin\theta & C_y\cos\theta & t_y C_y \\ 0 & 0 & 1 \end{bmatrix}$
C_x, C_y: Scaling
t_x, t_y: Translation
C.
S_v: Vertical shear
$\begin{bmatrix} (C_x\cos\theta + C_y\sin\theta S_v) & (-C_x\sin\theta + C_y\cos\theta S_v) & (t_x C_x + t_y C_y S_v) \\ C_y\sin\theta & C_y\cos\theta & t_y C_y \\ 0 & 0 & 1 \end{bmatrix}$
D. Yes, order does makes difference!
Ex: $\begin{bmatrix} C_x & 0 & 0 \\ 0 & C_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_x & 0 & t_x C_x \\ 0 & C_y & t_y C_y \\ 0 & 0 & 1 \end{bmatrix}$
but.
$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_y & 0 & 0 \\ 0 & C_y & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_x & 0 & t_x \\ 0 & C_y & t_y \\ 0 & 0 & 1 \end{bmatrix}$

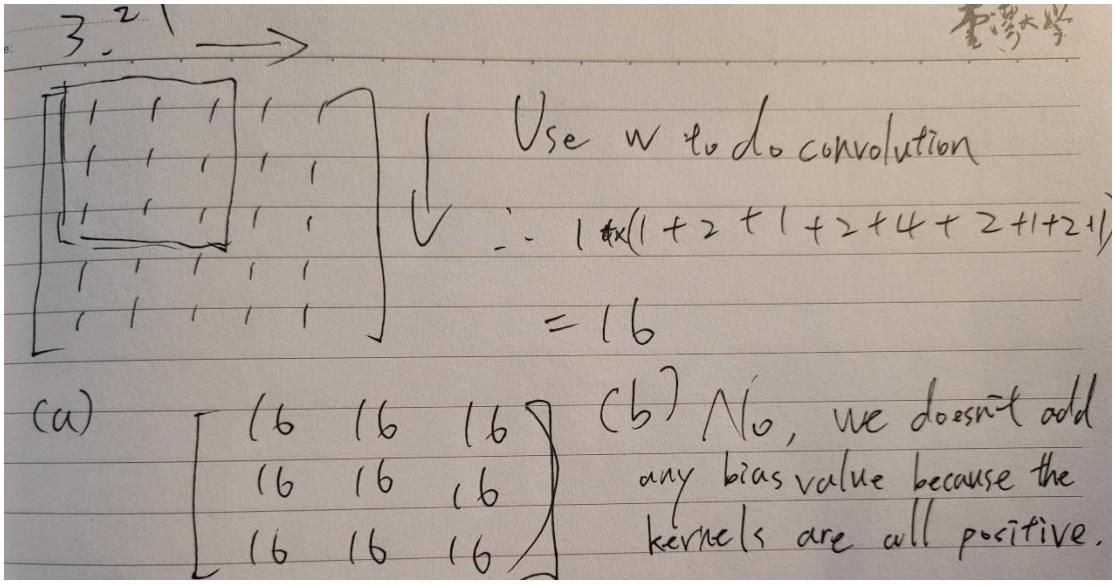
3.12

Ans:



3.21

Ans:



Part 2: (70%) Image File Reading, Display and Basic Processing

1. Read a color BMP or JPEG image file and display it on the screen.

```
Mat img = cv::imread(fileName.toStdString());
Img = cv::imread(fileName.toStdString());
cvtColor(img, img, COLOR_BGR2RGB);
```

使用cv的imread將jpg或bmp讀進來

2. Convert a color image into a grayscale image using the following

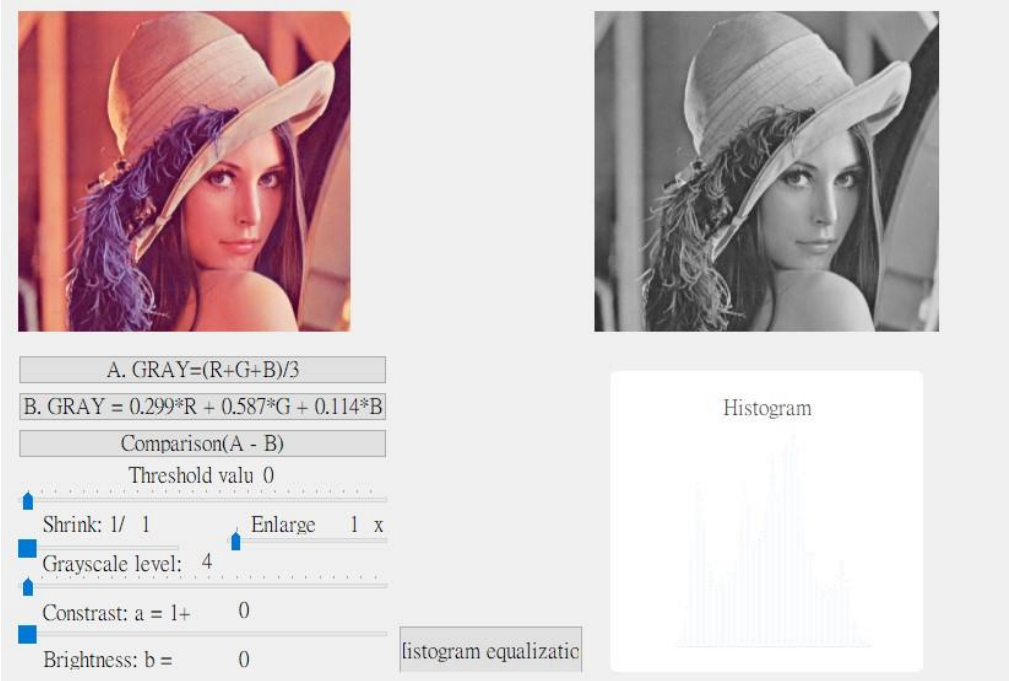
equations: A. GRAY = (R+G+B)/3.0

B. GRAY = 0.299*R + 0.587*G + 0.114*B

Compare the grayscale images obtained from the above equations. One way to compare difference between two images is by image subtraction

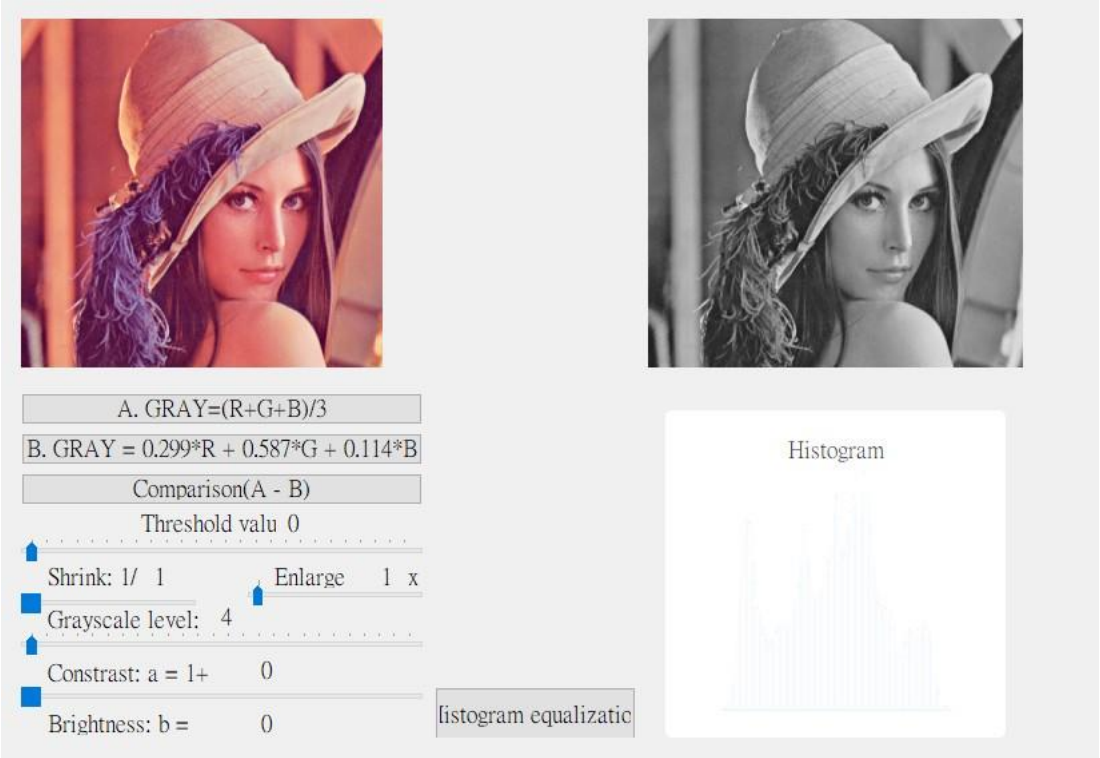
A GRAY = (R+G+B)/3

```
int m = (img.at<Vec3b>(i,j)[0] + img.at<Vec3b>(i,j)[1] + img.at<Vec3b>(i,j)[2])/3;
img2.setPixel(j,i,qRgb(m,m,m));
```



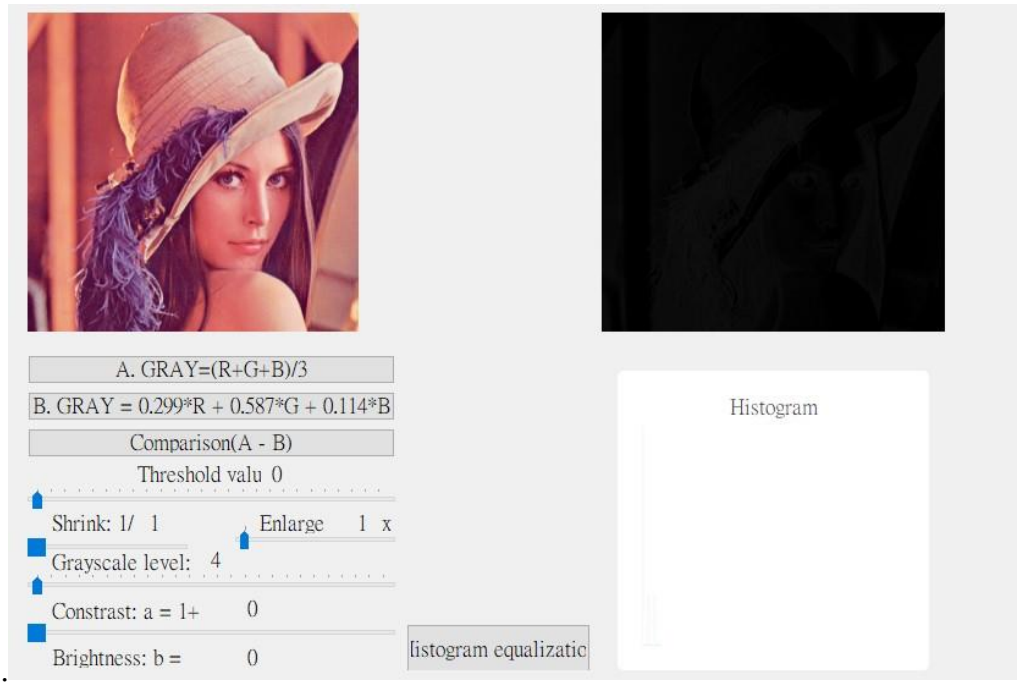
B GRAY = 0.299*R + 0.587*G + 0.114*B

```
int m = img.at<Vec3b>(i,j)[0]*0.114 + img.at<Vec3b>(i,j)[1]*0.587 + img.at<Vec3b>(i,j)[2]*0.299;
img2.setPixel(j,i,qRgb(m,m,m));
```



比較(AB相減,用qBound避免值小於零)

```
int m = img.at<Vec3b>(i,j)[0]*0.22 + img.at<Vec3b>(i,j)[1]*(-0.254) + img.at<Vec3b>(i,j)[2]*0.034;
int n = qBound(0, m, 255);
img2.setPixel(j,i,qRgb(n,n,n));
```



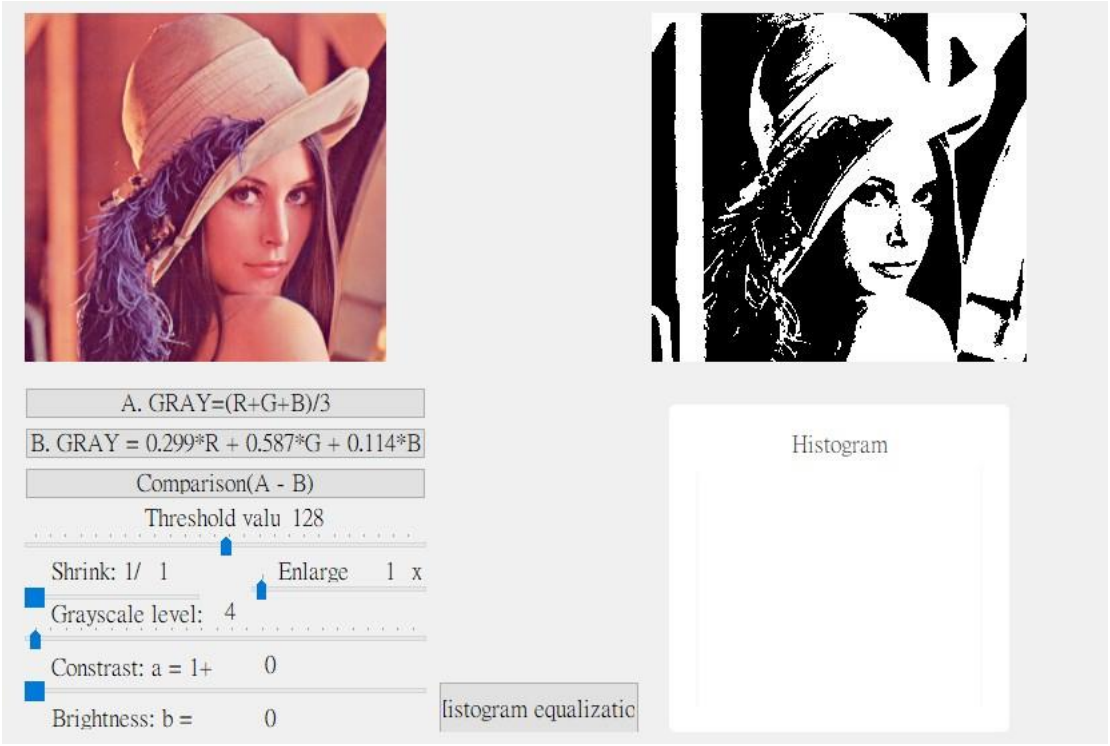
.

3.Determine and display the histogram of a grayscale image.

如上述各圖之histogram欄位

4. Implement a manual threshold function to convert a grayscale image into a binary image.

```
int m = img.at<Vec3b>(i,j)[0]*0.114 + img.at<Vec3b>(i,j)[1]*0.587 + img.at<Vec3b>(i,j)[2]*0.299;
if (m >= position)
    m = 255;
else
    m = 0;

img2.setPixel(j,i,qRgb(m,m,m));
```



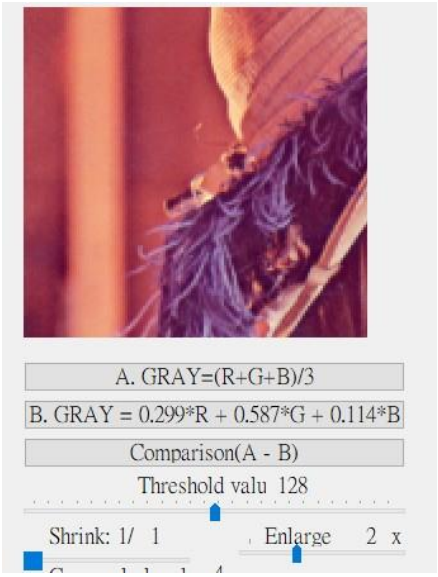
使用2(b)公式作為灰階值,如圖,設定當threshold為128時

5.Implement a function to adjust the spatial resolution (enlarge or shrink) and grayscale levels of image. Use an interpolation method on enlarging.

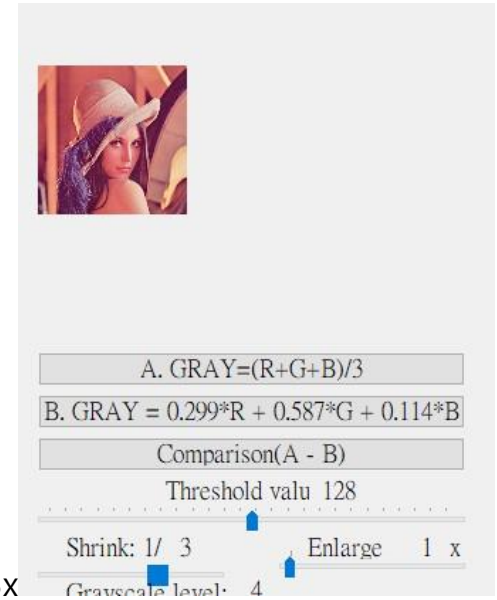
Enlarge:在放大時使用內插法，賦值方法為放大n倍時,使用原圖中每組(nxn)最左上pixel的值來內插至放大後新增的範圍，Shrink:縮小時,取值方法為原圖中每組(nxn)最左上角的pixel值,如圖:

:

```
QImage img2(img.cols*position, img.rows*position, QImage::Format_RGB888);

for(i=0 ; i<img.rows*position ; i++)
{
    for(j=0 ; j<img.cols*position ; j++)
    {

        int k = floor(i/position);
        int l = floor(j/position);
        int r = img.at<Vec3b>(k,l)[2];
        int g = img.at<Vec3b>(k,l)[1];
        int b = img.at<Vec3b>(k,l)[0];
        img2.setPixel(j,i,qRgb(r,g,b));

    }
}
ui->label->setPixmap(QPixmap::fromImage(img2));
```

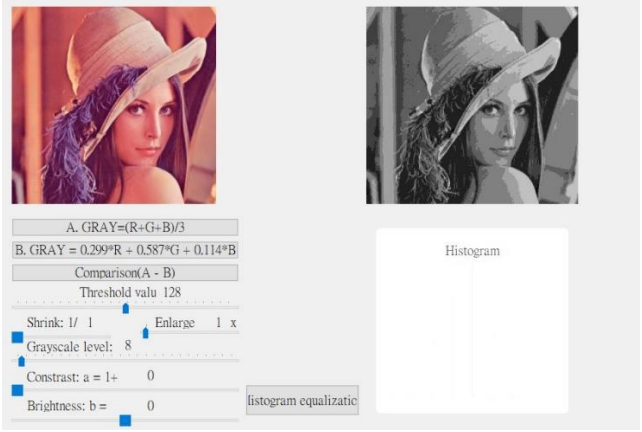


左:2x,右1/3x



Grayscale levels:

```
m = floor(m*position/256)*(256/position);
```

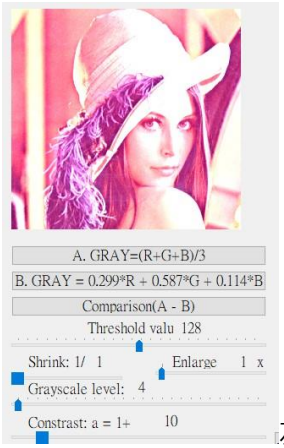


如上圖, Grayscale level為8時

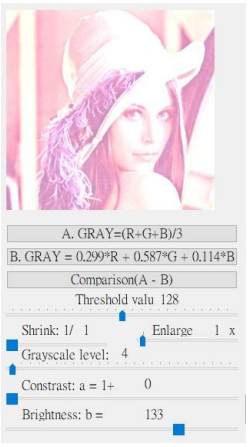
6.Implement a function to adjust the brightness and constrast of image.

公式為 alpha(x,y)+beta ,gamma設為1,主要目標是看對比與明暗調節

```
int r = floor(img.at<Vec3b>(i,j)[2]*alpha0+beta0);
int g = floor(img.at<Vec3b>(i,j)[1]*alpha0+beta0);
int b = floor(img.at<Vec3b>(i,j)[0]*alpha0+beta0);
r = qBound(0,r,255);
g = qBound(0,g,255);
b = qBound(0,b,255);
img2.setPixel(j,i,qRgb(r,g,b));
```

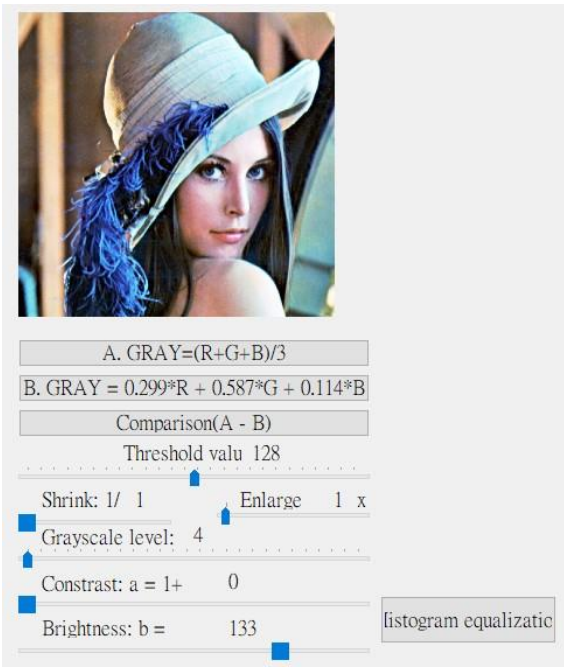


左圖為對比alpha調整,右圖明暗beta調整



7.Implement a histogram equalization function.

先計算出整張圖片各channel的cdf,然後對所有pixel進行均值化,如圖:




```
        rhistogram[ img.at<Vec3b>(i,j)[2] ]++;
        ghistogram[ img.at<Vec3b>(i,j)[1] ]++;
        bhistogram[ img.at<Vec3b>(i,j)[0] ]++;
    }
}
rcdf[0] = rhistogram[0];
gcdf[0] = ghistogram[0];
bcdf[0] = bhistogram[0];
for (i = 1; i < 256; i ++)
{
    rcdf[i] = rcdf[i - 1] + rhistogram[i];
    gcdf[i] = gcdf[i - 1] + ghistogram[i];
    bcdf[i] = bcdf[i - 1] + bhistogram[i];
}

int max = img.rows*img.cols;

for(i=0 ; i<img.rows ; i++)
{
    for(j=0 ; j<img.cols ; j++)
    {

        float x = rcdf[img.at<Vec3b>(i,j)[2]];
        float y = gcdf[img.at<Vec3b>(i,j)[1]];
        float z = bcdf[img.at<Vec3b>(i,j)[0]];
        float rf = (x - rcdf[0]) / (max-rcdf[0]);
        float gf = (y - gcdf[0]) / (max-gcdf[0]);
        float bf = (z - bcdf[0]) / (max-bcdf[0]);
        int r = floor(rf*255);
        int g = floor(gf*255);
        int b = floor(bf*255);
        img2.setPixel(j,i,qRgb(r,g,b));
```