HW2

1. How can you perform the conversion and make sure the new RGB images are visually identical to the original grayscale images?

(convert all these 8-bit grayscale images into 36-bits per pixel in RGB)

講義中轉換 rgb 到 grayscale 使用 L= R*0.299+G*0.587+ B*0.114，所以只要使 L=R=G=B 就可以使新的 36 bit RGB 存各種顏色(R,G,B)12bit 都存相同於 gray-scale(8 bit)的值。.

2. If you have a 6-bit gray-scale image in 720p resolution, what is the resolution of the dithered image?

   將 6 bit gray-scale 轉換成只用 1 bit 黑白，為了模擬出 6 bit 可能的所有顏色使用 8x8 矩陣可以概括每個 64 種顏色(6 bit)，所以原本一個 1 pixel 變 64 pixel ，720p->720*8=5760p，因為寬高都變 8 倍。

3. What text color you should use to maximize the readability of your text, so that your professor wouldn't be mad at you?

可讀性能從兩種對比來探討，一是亮度比，也就是計算 luminance。另一種是色度差從 RGB 每個像度值的差異來看。

從 12 色相環來看，黃色對比最高的顏色是深藍色，所以選擇深藍色能達到最佳可讀性。

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

For yellow = (255,255,0) background

For dark blue= (0,0,255) text

Color difference: |Red1-Red2|+|Green1-Green2|+|Blue1-Blue2|

4. transform RGB to XYZ

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.24 & -1.54 & -0.50 \\ -0.97 & 1.88 & 0.04 \\ 0.06 & -0.20 & 1.06 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

(a) How do we convert an RGB pixel back to an XYZ pixel

$$\begin{pmatrix} 3.24 & -1.54 & -0.5 \\ -0.97 & 1.88 & 0.04 \\ 0.06 & -0.2 & 1.06 \end{pmatrix}^{-1} = \begin{pmatrix} 0.41212\ldots & 0.35683\ldots & 0.18093\ldots \\ 0.21228\ldots & 0.71359\ldots & 0.07320\ldots \\ 0.01672\ldots & 0.11444\ldots & 0.94696\ldots \end{pmatrix}$$

做 inverse 得到反矩陣就可以使用 RGB 來轉換回 XYZ

(b) Is the RGB to XYZ conversion lossy or lossless? Why?

!ossless. 因為可以用矩陣乘法來從兩邊互相轉換，所以並不會 loss.

(c) Are the above equation applicable to all RGB color models?

不可以。根據不同 model 的定義應該該要去修改矩陣才可能得到正確的對應關係

5. convert the image from pixel domain into DCT domain using a 16 x 16 2D DCT

a) (0.5%) What is the resolution of your output thumbnail image?

1280×720

720/16=45

變成 45p 因為每個 16*16 block 便 1 個 pixel

b) (0.5%) How many pixels in the original image were used to create your output thumbnail image

因為要看過整張圖來轉成較小的所以

1280*720=921600 pixels

6. create a new JPEG algorithm for the Taiwanese Coast Guard Dog Squad (TCGDS), in order to maximize the image quality. Which step of the JPEG algorithm do you want to change?

Sampling: RGB to YCrCb 轉換時的 a:b:c 為 4:4:4 保存所有的數值不少去，可以完整保存原本的資訊，當然壓縮就相對少。

Quantization step: 如果讓 DCT 的係數保存完整不去 quantize 和 round off，可以保存原圖較完整轉換回去。

7. Please propose a simple progressive JPEG algorithm that generates 6 scans in total. Please explain how individual DCT coefficients get assigned to the scans.

假設是 8x8 的 DCT coefficient 產生出來，知道因為又分為 Y, Cr, Cb 三種 component，其中以 Y 最為主要，而從 8x8 中左上角 DC 低頻為主往右下高頻影響較小。

所以可根據不同 coefficient 的重要性排序設計 6 個 scan 來分次傳送這全部 coefficient 到 decode 端：

1. Y,Cr,Cb: DC 0
2. Y: AC 1-2
3. Y: AC 3-5
4. Cr: 1-63
5. Cb: 1-63
6. Y: 9-63

8. You need to implement your own: (1) PSNR, (2) DCT, and (3) inverse DCT functions

使用 python opencv 來實作，用到的 library 如圖上所示：

(1) 讀檔以及宣告一些參數

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

img = cv2.imread('bear.jpeg')
img = cv2.resize(img,(1400,880))
yuv = cv2.cvtColor(img,cv2.COLOR_BGR2YUV_I420)###轉 YUV
img1 = img.astype('float')
C_temp = np.zeros((8,8),float)
dst = np.zeros((8,8),float)

yuv_d_1 = np.zeros((yuv.shape[0],yuv.shape[1]), float)
yuv_r_1 = np.zeros((yuv.shape[0],yuv.shape[1]), float)
yuv_d_4 = np.zeros((yuv.shape[0],yuv.shape[1]), float)
```

```
yuv_r_4 = np.zeros((yuv.shape[0],yuv.shape[1]), float)
yuv_d_16 = np.zeros((yuv.shape[0],yuv.shape[1]), float)
yuv_r_16 = np.zeros((yuv.shape[0],yuv.shape[1]), float)
```

(2) Dct (底為 8*8) 使用矩陣運算的方式(原本為一個值一個用 for loop 但跑太久又難 debug)

$$G_{u,v} = \sum_{x=0}^{7}\sum_{y=0}^{7} \alpha(u)\alpha(v) g_{x,y} \cos\left[\frac{\pi}{8}\left(x+\frac{1}{2}\right)u\right] \cos\left[\frac{\pi}{8}\left(y+\frac{1}{2}\right)v\right]$$

```
m = 8
n = 8
N = n
C_temp[0, :] = 1 * np.sqrt(1/N)

for i in range(1, m):
     for j in range(n):
          C_temp[i, j] = np.cos(np.pi * i * (2*j+1) / (2 * N )) * np.sq
rt(2 / N )
```

計算出運算用的矩陣 C_temp

```
width_d =175
height_d =165
width = 1400
height = 880

for i in range(height_d):
    for j in range(width_d):

        dst = np.dot(C_temp , yuv[i*8:(i+1)*8,j*8:(j+1)*8])
        dst = np.dot(dst, np.transpose(C_temp))

        yuv_d_1[i*8,j*8] = dst[0,0]
        yuv_d_4[i*8:i*8+2,j*8:j*8+2] = dst[0:2,0:2]
        yuv_d_16[i*8:i*8+4,j*8:j*8+4] = dst[0:4,0:4]
```

到這裡完成 dct 且做 1, 4, 16 的運算

(3) Idct (底為 8*8)

$$f_{x,y} = \sum_{u=0}^{7}\sum_{v=0}^{7} \alpha(u)\alpha(v)F_{u,v}\cos\left[\frac{\pi}{8}\left(x+\frac{1}{2}\right)u\right]\cos\left[\frac{\pi}{8}\left(y+\frac{1}{2}\right)v\right]$$

```
        dst = np.dot(np.transpose(C_temp),yuv_d_1[i*8:(i+1)*8,j*8:(j+1)
*8])
        dst = np.dot(dst, C_temp )

        yuv_r_1[i*8:(i+1)*8,j*8:(j+1)*8] = dst[:,:]

        dst = np.dot(np.transpose(C_temp),yuv_d_4[i*8:(i+1)*8,j*8:(j+1)
*8])
        dst = np.dot(dst, C_temp )

        yuv_r_4[i*8:(i+1)*8,j*8:(j+1)*8] = dst[:,:]

        dst = np.dot(np.transpose(C_temp),yuv_d_16[i*8:(i+1)*8,j*8:(j+1
)*8])
        dst = np.dot(dst, C_temp )

        yuv_r_16[i*8:(i+1)*8,j*8:(j+1)*8] = dst[:,:]
```

完成 idct 的部分

(4) psnr

- Mean Square Error (MSE):

$$MSE = \frac{\sum_{M,N}[I_1(m,n) - I_2(m,n)]^2}{M * N}$$

  - M, N: Columns and rows, i.e., pixels

- PSNR:

$$PSNR = 10\log_{10}\left(\frac{R^2}{MSE}\right)$$

  - R: input value domain range, e.g., 255

在 code 中的最後部分，和把圖片從 yuv420 回到 rgb 一起處理：

```
###for 1
dst_i = yuv_r_1.astype('uint8')
dst_i = cv2.cvtColor(dst_i ,cv2.COLOR_YUV2BGR_I420)
cv2.imshow('1', dst_i)

mse_1=0
for i in range(880):
    for j in range(1400):
        mse_1 =mse_1 + (((yuv_r_1[i][j]-yuv[i][j]))**2)/(1400*880)

psnr_1= 10*np.log10((255**2)/mse_1)

###for 4
dst_i = yuv_r_4.astype('uint8')
dst_i = cv2.cvtColor(dst_i ,cv2.COLOR_YUV2BGR_I420)
cv2.imshow('4', dst_i)

mse_4=0
for i in range(880):
    for j in range(1400):
        mse_4 =mse_4 + (((yuv_r_4[i][j]-yuv[i][j]))**2)/(1400*880)

psnr_4= 10*np.log10((255**2)/mse_4)

###for 16
dst_i = yuv_r_16.astype('uint8')
dst_i = cv2.cvtColor(dst_i ,cv2.COLOR_YUV2BGR_I420)
cv2.imshow('16', dst_i)

mse_16=0
for i in range(880):
    for j in range(1400):
        mse_16 =mse_16 + ((yuv_r_16[i][j]-yuv[i][j])**2)/(1400*880)

psnr_16= 10*np.log10((255**2)/mse_16)

cv2.waitKey(0)
```

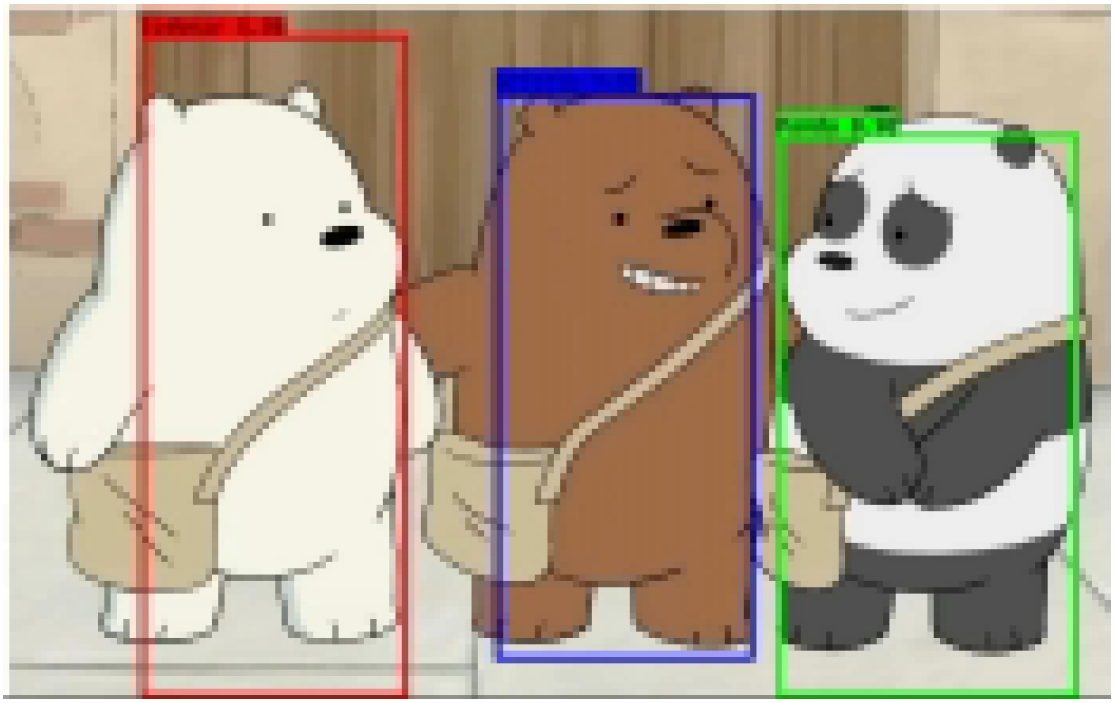
```
cv2.destroyAllWindows()

print("PSNR 1 = ",round(psnr_1,2))
print("PSNR 4 = ",round(psnr_4,2))
print("PSNR 16 = ",round(psnr_16,2))
```

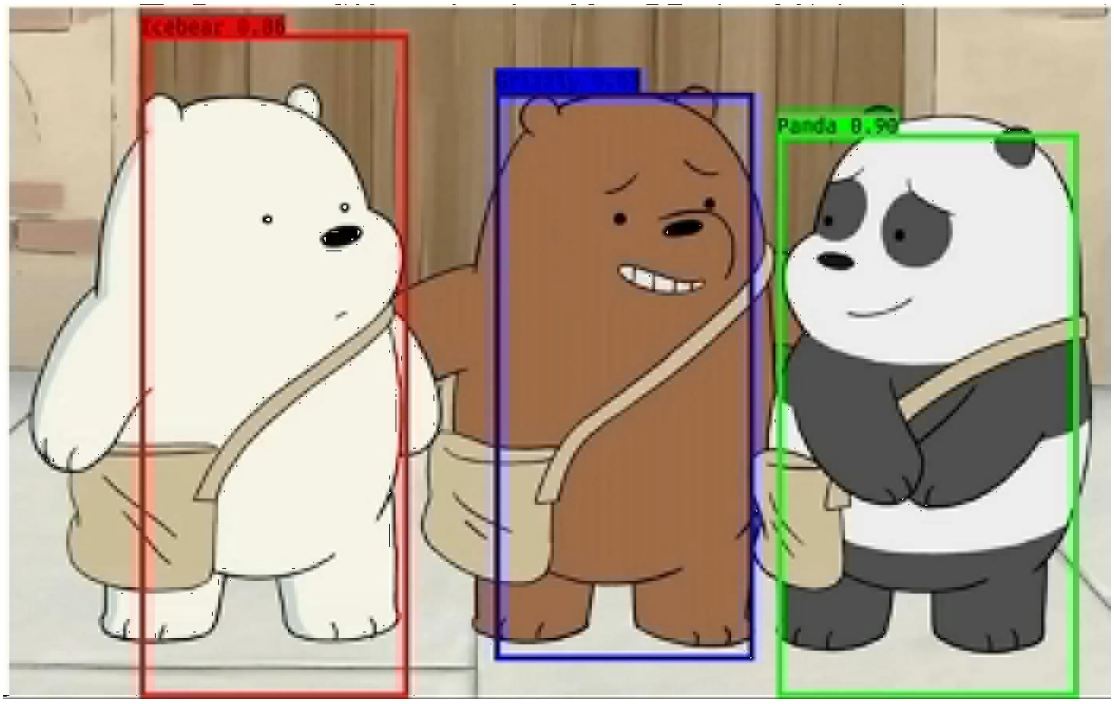
最後得到的 PSNR：

```
PSNR 1 =  22.19
PSNR 4 =  26.83
PSNR 16 =  35.47
```

圖片 1



圖片 4



圖片 16