Introduction of Multimedia HW1

103062318 蔡尚倫

1. DCT image compression

包含 DCT_image.m　dct2d.m　decode.m　encode.m　idct2d.m

PSNR.m(這裡用大寫避開 build in function conflict)

RGB：(n=2 ,4,8)

DCT_image.m：imread　image，並且把它轉成 double 計算，call

encode.m　->　decode.m　-> imwrite　image ->PSNR(original,image)
show　PSNR

encode.m：利用雙層 for loop 每 8 個為間距取得 block 的起始值，call dct2d.m，矩陣可以用(,)來取得的 slice，這邊我分成 R、G、B 三種分開實作(後面發現其實可以更快的寫法)，接著創造出一個 8X8 mask matrix，結著依照 n 傳入的參數決定要保留的數量，與上方一樣使用雙層 for loop，分別對每個 block 做點乘(.*)，用意是會一一對應，如果我 mask 上的值是 1 的話會保留 0 的話捨棄

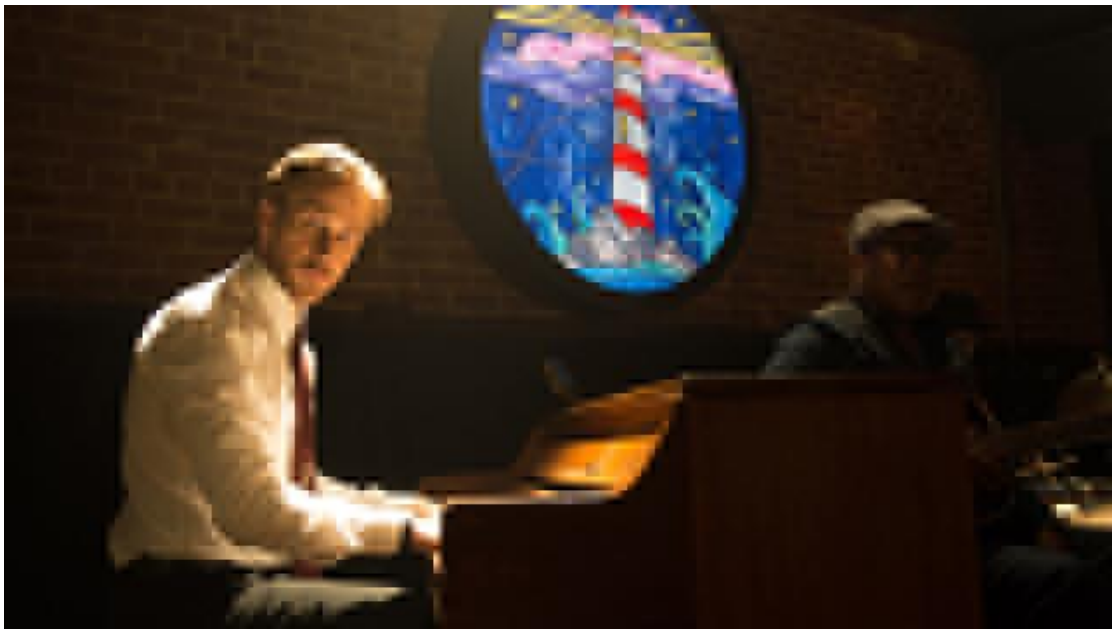
dct2d.m：做 DCT 轉換再 matlab 因為可以做矩陣的相乘，所以描述公式就比較簡單，由原來的公式可以轉為 A*input*A’ ，A 可用一維的 DCT 來表示，如此一來就能簡化公式的繁雜，要注意 n 值得填寫，因為這次是以 8X8 為一個 block，所以 n=8。

decode.m ：與 encode 相反，呼叫 idct2d.m，也不用在做 mask

idct2d.m：將原本公式相反即可還原 A’*input*A

PSNR.m：把公式套入即可，因為會重複用到寫成 function

n=2 a2.png PSNR = 26.888657



n=4 a4.png PSNR = 33.372222

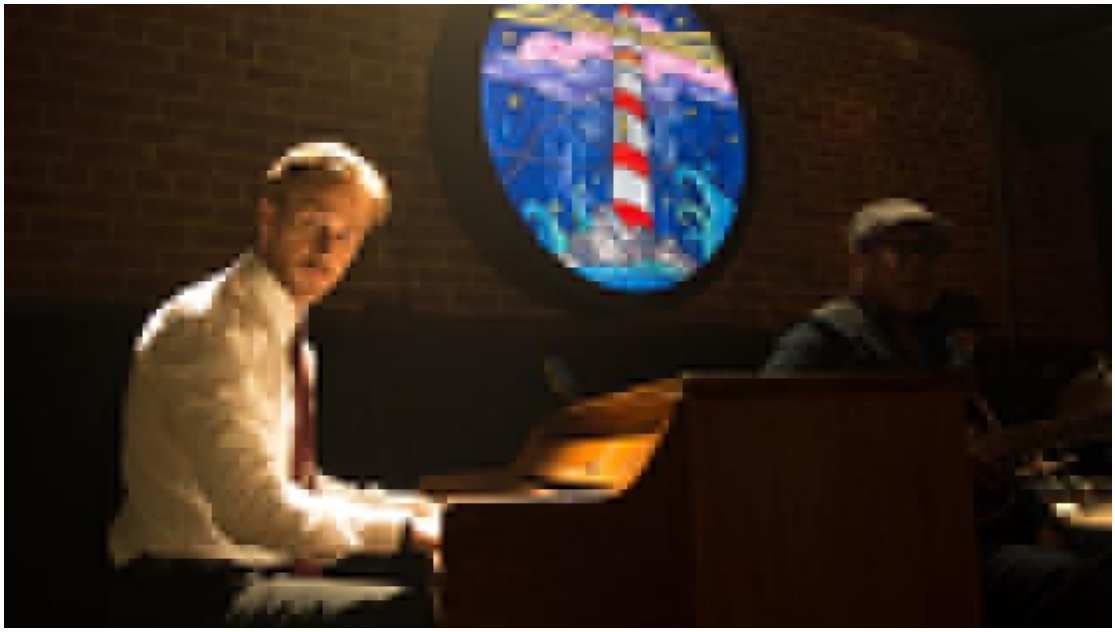
n=8　a8.png　PSNR = Inf

(a) PSNR 大小是 n=8 > n=4 > n=2 ，PSNR 代表的就是與原本圖片的差距，因為 DCT 是將圖片轉成頻率，而題目要求取左上角是要保留低頻的資訊，而一張圖片通常有 locality 的特性，就是在該點附近的顏色”通常”會跟自己很像，而高頻代表的就是圖片上物體與物體的邊界，會是顏色變化最巨大的部分，所以濾掉高頻會使的圖片邊界模糊，如同 n=2 的圖片所展現的，但有時為了壓縮，其實 n=4 的圖片還原度已經表現的不錯。

YIQ(n=2 4 8)

主要與 RGB 做法相同，不同的是在 DCT 處理以前要從 RGB domain 轉成 YIQ domain，最後做完 encode decode 之後要記得再轉回 RGB domain

n=2 b2.png PSNR = 26.888434



n=4 b4.png PSNR = 33.372743

n=8　b8.png　PSNR = Inf

(b) PSNR 大小是 n=8 > n=4 >n=2，理由跟(a)小題一樣，PSNR 代表的就是與原本圖片的差距，也可以稱做還原程度，保留越多資訊當然 PSNR 會越大，當兩者沒有差距時，則 PSNR 為 INF

(c) 兩者出來的結果 PSNR 是相同的，理由我想是說你在不同的 domain 上丟棄資訊，一樣是丟棄等量的資訊量，所以在做還原圖片時，兩者出來的結果會非常類似，兩者 domain 轉換是線性的，所以在哪個 domain 做 DCT，結果是相同的，這裡注意到的是可能會因為你轉換矩陣小數點的準確度，而造成 PSNR 有些微差距，大概 0.00X 左右，如果結果不是差太多，應該都是能接受的範圍

2. Image filtering

包含：imgfilter.m

Gaussian filter：(n=3　segma=0.3) (n=9　segma=1)

imgfilter.m：先用 fspecial 創出 Gaussian mask，這裡只要上網去查 document 填入參數即可，接著創造一個大小是(rows+2*(n-1)/2, cols+2*(n-1)/2) zeros matrix，目的用來填入原本 image，方便 filter 的移動，將原本的圖片填到正中央的位置，這樣 filter 最中心的點剛好就對到圖片的第一個 pixel，接著跑兩層 for loop，將 mask 與對應到的 block 做.*(點乘)，並將該矩陣的數值全部加起來，assign 到輸出的 image，跑出 loop，再 image　imwrite

n=3　segma=0.3 (此圖片有縮放過，原圖請參考資料夾)

ta_gaus3X3.jpg

n=9　segma=1 (此圖片有縮放過，原圖請參考資料夾)

ta_gaus9X9.jpg

(a)由濾鏡(n)大小可得知，如果 n 很大的話，就會參考到周圍大部分點，而 Gaussian filter 可以看作是一種權重的分配，是由 Gaussian 函數產生，是一種常態分布，以自己這個 pixel 當作是常態分佈的最高點，segma 大小代表你這個點的重要性，越低的話資料越集中，波峰也就越高，越大的話資料就開始往兩側分散，波峰相較而言就比較低。所以第一張圖片採用 n=3　segma=0.3 基本上幾乎就是採用自己的那個點當作輸出結果，所以圖片會覺得跟原本很像，然而 n=9　segma=1 的則是濾鏡較大，參考到的周圍值就越多，自己的權重也沒有那麼高

了，所以圖片會將原本的 noise 消除不少。

Median filter：(n=3) (n=9)

主要的操作跟 Gaussian 差不多，最主要是在 filter 的使用不一樣，這次是將濾鏡選取的範圍內，找出這些所有值的中位數，而 3*3=9 與 9*9=81 剛好都是基數，所以中位數很好找，用這個中位數來代表這個 pixel 的輸出

n=3　(此圖片有縮放過，原圖請參考資料夾)

ta_med3X3.jpg

n=9　(此圖片有縮放過，原圖請參考資料夾)

ta_med9X9.jpg

(b) n=3 的圖片很成功的消除了 noise，而且圖片還是保持的清晰，然而 n=9 的圖片雖然消除了 noise，但卻讓圖片過於模糊，讓人感覺好像丟了許多的資訊。兩者的差距在於濾鏡的大小，濾鏡太大會造成 sample 到的值太多，進而有很大的機率被周圍 pixel 取代。Gaussian 與 median 兩者 median 表現較佳，原因我想在於 noise 顏色是白色，白色的值恰巧是最高的，所以使用 median 時，白色自然不太可能成

為中位數，反觀 Gaussian 雖然利用自己 pixel 權重較大的特性，可是還是會被 noise 的高數值影響，造成圖片沒辦法有效的過濾掉 noise。

3. Interpolation

包含：interpolation.m　PSNR.m

Nearest neighbor：

interpolation.m：創一個長與寬都是原本圖片四倍的 zeros matrix，接著用兩層 for loop 跑 4*original rows 和 4*original cols，接著用 sampling 的方式來取得該 pixel 應該要對應到原來 original image 的 pixel，只要將 i/4 and j/4 取 round 即可，因為原本就是要找靠近的點，所以四捨五入是很好的方法，也是講義上寫的方式，如此一來就能取得放大後的圖片，最後 image imwrite，由於下方圖片縮放後會影響效果，所以換頁展示。

nearest neighbor　a_4X.png　PSNR = 26.236858

Bilinear：

interpolation.m：前置作業與 nearest neighbor 一樣，最核心的演算是內插法來決定權重，與課本相同，這裡不分 RGB 三種來做，使的 code 較為簡潔，但會跑出 warning，不過不影響結果。Sample 完的 x 與 y 分別取 floor 與 ceil，產生四個變數 f_x　c_x　f_y　c_y，比例的取得方式用 y_rat =rem(j/4,1)，這樣的方法可以讓我們將整數部分消除，

用 mod 也可以(因為確保兩個值會是正數)，接著分別取出四個 pixel(RGB)取出，方法是例如 img(f_x,f_y,:)，用冒號就能代表取得該維度的 vector，這裡代表的是三原色 RGB，如此一來不用像之前一樣分開來做，再用 cat 合併。接下來套用內插公式，分別先將左上與右上執行內插(x 軸上的比例，記得要乘另外一邊的比例，這部分看 code 比較能理解)，再來將左下與右下座內插，得到的兩個新矩陣，再做一次內插，但這次上下合併，不要用錯比例。如此一來就能完成內插出來的 RGB vector，將其 assign 給輸出圖片的 pixel，這裡會出現前面所說的 warning，簡單來說 compiler 說有可能會改變矩陣大小，可是你在寫 code 時確保不會發生即可。最後 image imwrite，由於下方圖片縮放後會影響效果，所以換頁展示。

bilinear　aa_4X.png　PSNR = 26.101265

(a)(b)已在圖片上方顯示

(c)這兩張圖片的差距在於 nearest neighbor 因為是選用其中一個原本的 pixel，所以正常顆粒感會很重，另外一個 bilinear 因為用內插法將周圍四個點做了權重的乘積，所以會發現該 pixel 不會出現在 original image 上，產生近視的模糊感，兩者的 PSNR 值相近。我想這裡的 PSNR 就不在是還原度的概念，因為從原本資訊少的 original image 要產生

資訊多的放大圖片，本來就是會有些技巧去混淆觀看者，以為兩者是接近的，所以我覺得這裡 PSNR 的意義是放大效果的表現，越高代表與高解析度圖片的”pixel 顏色”相似程度越高，畢竟他解析度高，擁有的資訊就是比較多，不太可能完全猜測使用到的顏色，只能藉由周圍 pixel 顏色的權重分配做出類似的顏色，自然 PSNR 就不會太高。