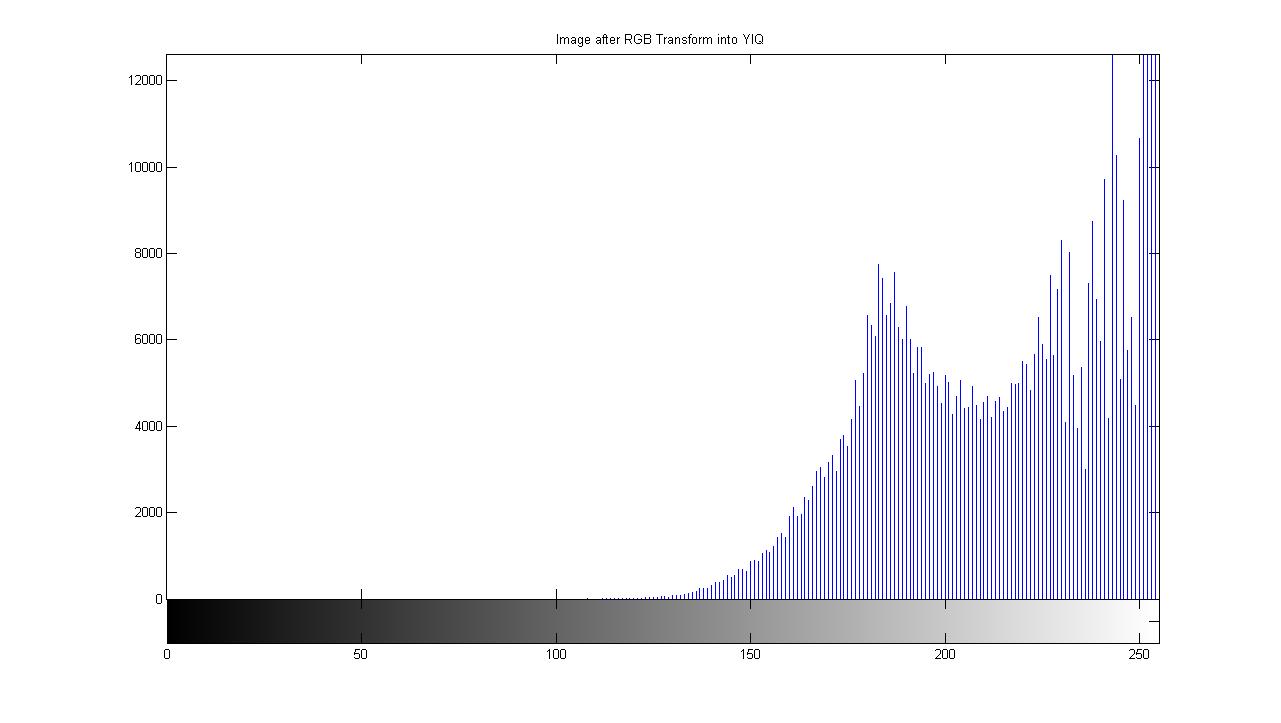
Hw1 Report

100070023 楊雅琪

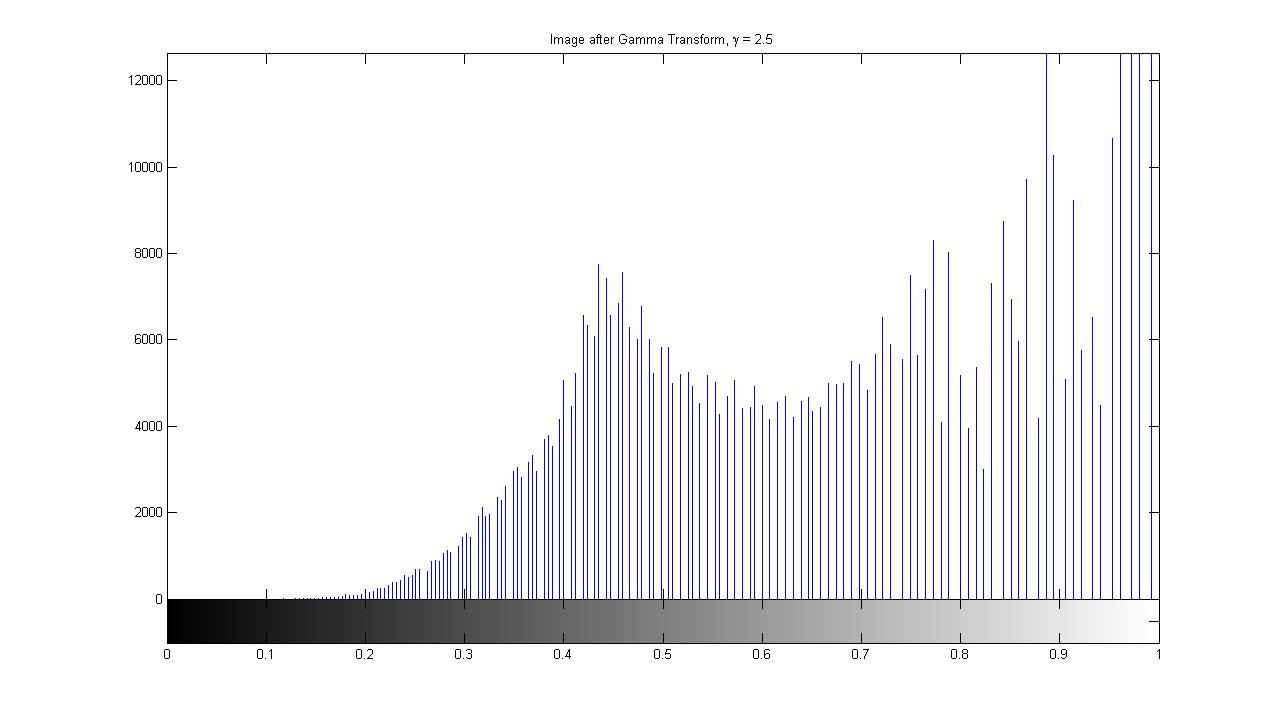
1. RGB to YIQ YIQ to RGB



1. RGB to YIQ Histogram



1. Apply gamma transform to Y channel (choose gamma=2.5)

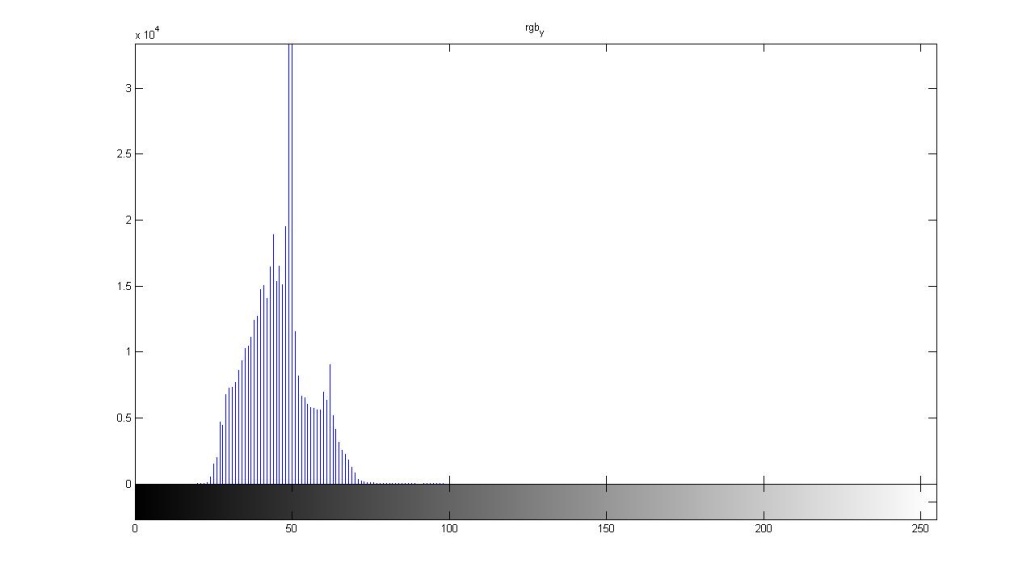


先將image乘上gamma值，再轉為double型態，並開始gamma transform

img2 = double(YIQMatrix()).^(2.5);

img2 = (img2 - min(img2(:)))./ max(img2(:) - min(img2(:)));

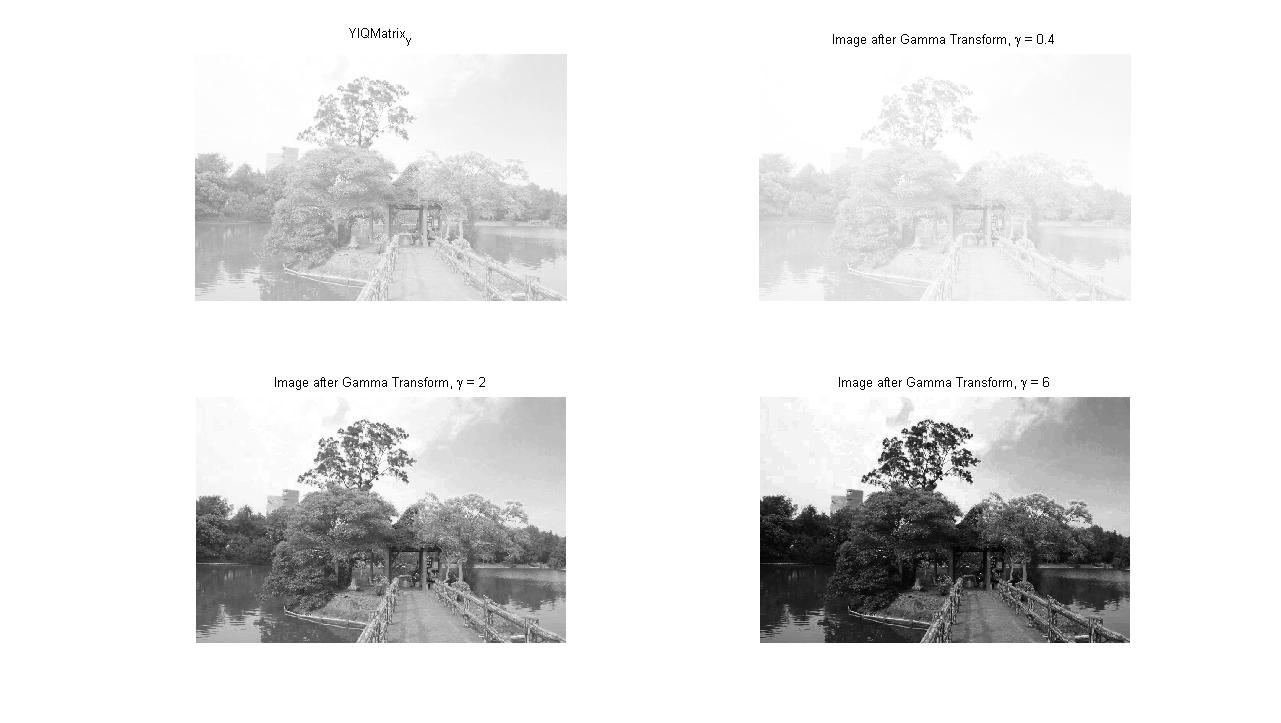
1. YIQ to RGB image & histogram of y channel

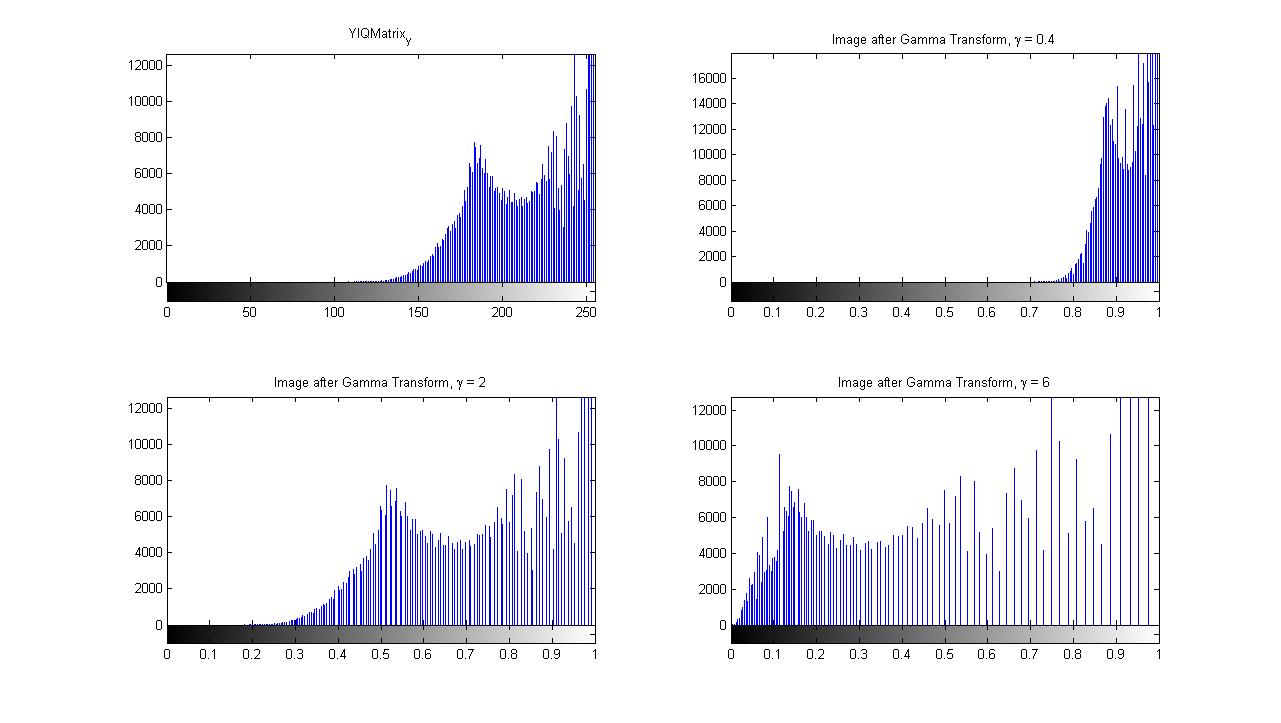




1. Compare & Discuss

從gamma value來看





* Gamma值越大，image明顯亮度越下降，而histogram的分布就越均勻
* 比較了RGB to YIQ 及YIQ to RGB 的histogram，很明顯的是分布區域並不同，這個問題讓我疑惑了許久，我實作這兩個轉換時，都是按照講義公式來解：

YIQMatrix(i,j,1)=0.299\*lake(i,j,1)+0.587\*lake(i,j,2)+0.114\*lake(i,j,3);

YIQMatrix(i,j,2)=0.596\*lake(i,j,1)-0.275\*lake(i,j,2)-0.321\*lake(i,j,3);

YIQMatrix(i,j,3)=0.212\*lake(i,j,1)-0.523\*lake(i,j,2)+0.311\*lake(i,j,3);

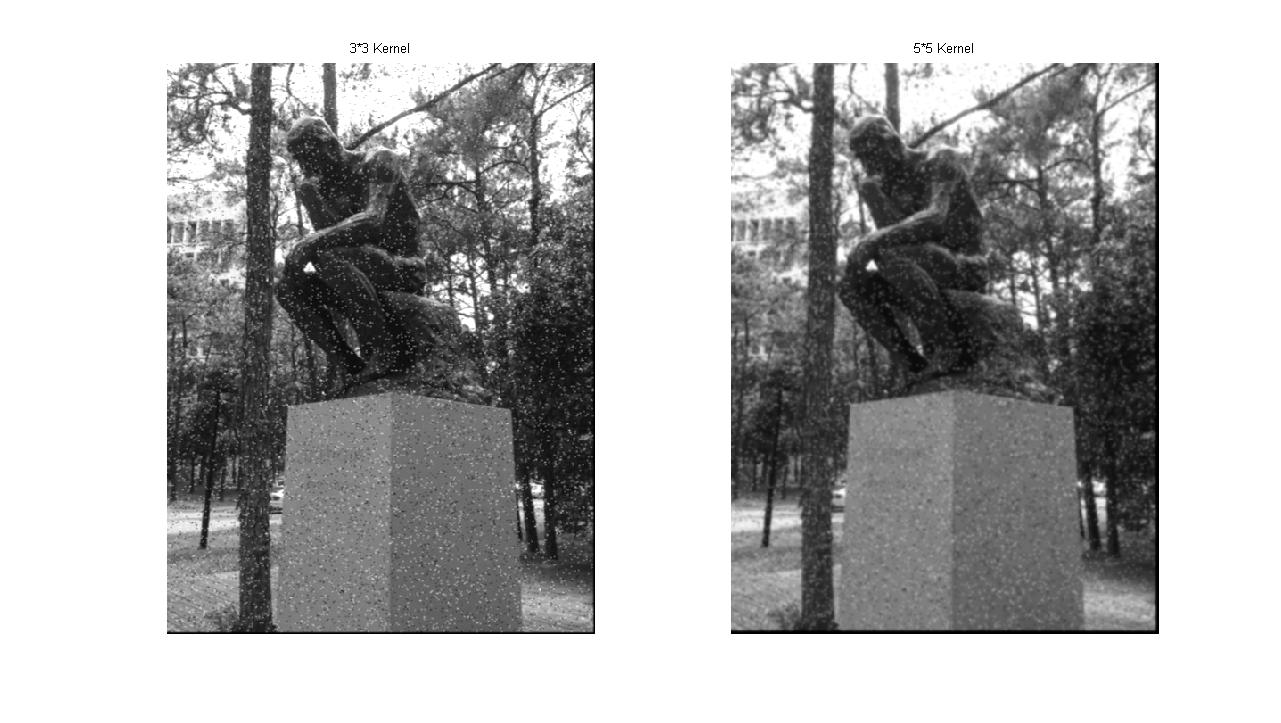
RGB(i,j,1)=img2(i,j,1)+0.956\*YIQMatrix(i,j,2)+0.621\*YIQMatrix(i,j,3);

RGB(i,j,2)=img2(i,j,1)-0.272\*YIQMatrix(i,j,2)-0.647\*YIQMatrix(i,j,3);

RGB(i,j,3)=img2(i,j,1)-1.106\*YIQMatrix(i,j,2)+1.703\*YIQMatrix(i,j,3);

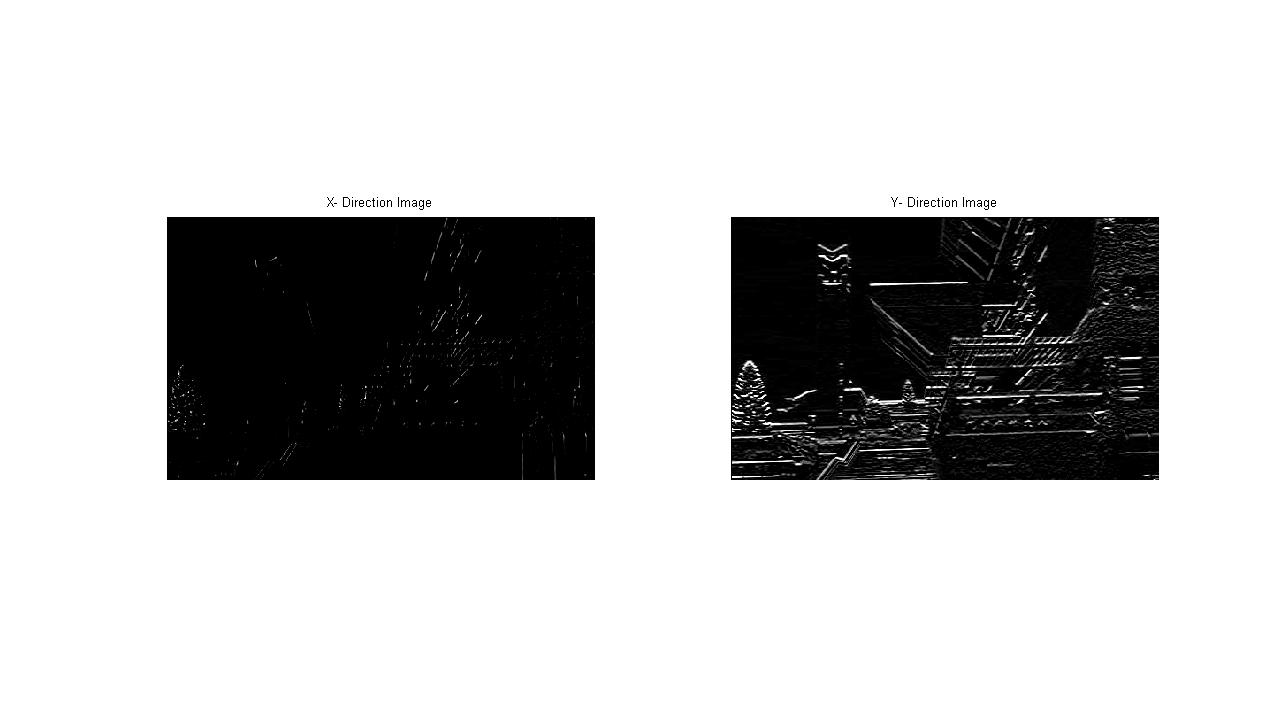
但轉換出來的顏色與histogram與原先相差甚遠，轉回來的顏色大部分為紫藍，少部分為紅，而我也實際用一些內建function跑過，並不會有這些問題，這可能要再找時間與助教討論。但可以知道的是，從RGB轉為YIQ時，會將大部份的能量集在y平面，而其餘彩度只占少部分，這也有可能是再轉回來RGB可能會失真的關係。

1. Image filter & convolution mask
2. Gaussian blur mask



根據公式，我寫了四個loop來跑每行每列的matrix，乘上kernel matrix後再累加，最後就能  
得出此兩張圖，我的發現是，kernel從3\*3－＞5\*5的實作中，圖片中的白點明顯被模糊化後，整張圖案起來較為清楚，雜訊較少，但其實主體(沉思的那個人)也會被模糊化，而滿明顯的是，兩張圖的右下邊都有黑邊，而5\*5的黑邊更是比3\*3的大塊，這大概是因為在mask的時候會造成一些error，而n\*n kernel matrix的n越大，error會更多，因此，要怎麼取捨，要多做比較來衡量，在這張圖中，我認為5\*5的kernel出來效果較好。

1. Edge Detection Mask



X\_Matrix = [1 0 -1; 1 0 -1; -1 0 -1];

Y\_Matrix = [1 1 1; 0 0 0; -1 -1 -1];

也是使用了4個loop，依照原image的行列跑，將轉為double的image的pixel的edge附近分別乘了x\_matrix(or y\_matrix)之後累加，最後放進imagex(or imagey)

for v = 2:484

for u = 2:297

sum\_x = 0;

sum\_y = 0;

for i=-1:1

for j=-1:1

sum\_x = sum\_x + doubleImage(u + i, v + j)\* X\_Matrix(i + 2,j + 2);

sum\_y = sum\_y + doubleImage(u + i, v + j)\* Y\_Matrix(i + 2,j + 2);

end

end

image\_x(u,v) = sum\_x;

image\_y(u,v) = sum\_y;

end;

end

比較之後，發現x方向的mask較不明顯，而y方向的mask較明顯。

1. Unsharp masking



做法是將讀入image\*2後減掉blur mask(此用Gaussian blur mask 3\*3 kernel matrix)，成為unsharp mask，也是4個loop實現(與前幾題相似)。

出現的圖會是銳利化的image。

1. Interpolation
2. NN



實作時，將image輸入後，將image存為[m,n,c]，  
再創一個4倍的I\_NN = uint8(zeros(M\*factor, N\*factor));

接下來再用3個loop依照m n c跑(從四個最接近整數位置像素的值求出新值)：

for i= 0:M\*factor-1

for j= 0:N\*factor-1

x = floor(j/factor);

y = floor(i/factor);

for k= 1:c

I\_NN(i+1, j+1, k) = I(y+1, x+1, k);

end

end

end

1. Bilinear



作法如講義所述，找出四點後再找half pixel position，連續三次的內插法後可得：

再創一個4倍的I\_4times = uint8(zeros(M\*factor, N\*factor));

比較：

NN執行時間較快，Bilinear相對較慢，而NN的效能較差，Bilinear就是效能較好，