資工系 100062236 林修安

多媒體技術概論 Assignment1 Report

1. Photo Enhancement

Method Description

先將圖片裡的 R、G、B 值取出來，並把 R、G、B 重組成一個 3x623392 的矩陣。

```
RGB = imread('ChangKungLake.jpg');
[w, l, d] = size(RGB);
RGB = double(RGB);
R = RGB(:, :, 1);
G = RGB(:, :, 2);
B = RGB(:, :, 3);
RGB = cat(1, R(:)', G(:)', B(:)');
```

$$RGB = \begin{bmatrix} R_1 & R_2 \dots\dots R_{623392} \\ G_1 & G_2 \dots\dots G_{623392} \\ B_1 & B_2 \dots\dots B_{623392} \end{bmatrix}$$

接著定義 YIQ 的轉換矩陣 θ，然後乘上 RGB 得到 YIQ，同時將 Y、I、Q channel 分別取出。

$$\theta = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \qquad YIQ = \begin{bmatrix} Y_1 & Y_2 \dots\dots Y_{623392} \\ I_1 & I_2 \dots\dots I_{623392} \\ Q_1 & Q_2 \dots\dots Q_{623392} \end{bmatrix} = \theta \begin{bmatrix} R_1 & R_2 \dots\dots R_{623392} \\ G_1 & G_2 \dots\dots G_{623392} \\ B_1 & B_2 \dots\dots B_{623392} \end{bmatrix}$$

```
theta = [0.299 0.587 0.114;0.596 -0.275 -0.321;0.212 -0.523 0.311];
YIQ = theta*RGB;
Y = YIQ(1, :);
I = YIQ(2, :);
Q = YIQ(3, :);
```

為了要顯示最初的 YIQ 轉換結果，要再將此三個陣列重組成 644x968x3 的矩陣。

$$YIQ = \begin{bmatrix} Y_1 & \dots\dots Y_{622749} \\ Y_2 & \dots\dots Y_{622750} \\ \dots & \dots\dots \\ \dots & \dots\dots \\ Y_{644} & \dots\dots Y_{623392} \end{bmatrix} \begin{bmatrix} I_1 & \dots\dots I_{622749} \\ I_2 & \dots\dots I_{622750} \\ \dots & \dots\dots \\ \dots & \dots\dots \\ I_{644} & \dots\dots I_{623392} \end{bmatrix} \begin{bmatrix} Q_1 & \dots\dots Q_{622749} \\ Q_2 & \dots\dots Q_{622750} \\ \dots & \dots\dots \\ \dots & \dots\dots \\ Q_{644} & \dots\dots Q_{623392} \end{bmatrix}$$

```
YIQ = cat(3, reshape(Y, w, l), reshape(I, w, l), reshape(Q, w, l));
```

因為 Gamma 轉換的輸出要在 0~1 之間，所以將 Y channel 的每個值先除以 255，再進行 Gamma 轉換，最後再乘以 255 以顯示經過 Gamma 轉換後的 YIQ 影像。

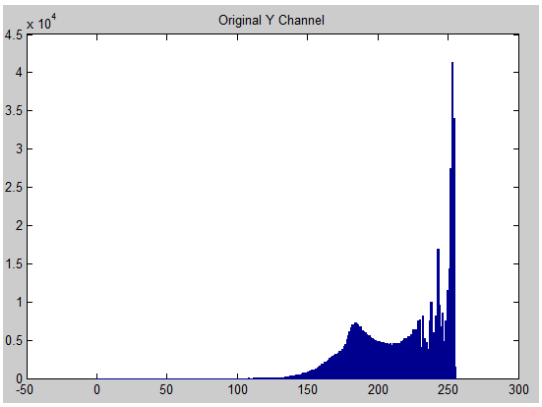
```
r = 4;
Y = ((Y./255).^r).*255;
YIQ = cat(3, reshape(Y, w, l), reshape(I, w, l), reshape(Q, w, l));
```

把經過 Gamma 轉換的 YIQ 換回 RGB，顯示結果。

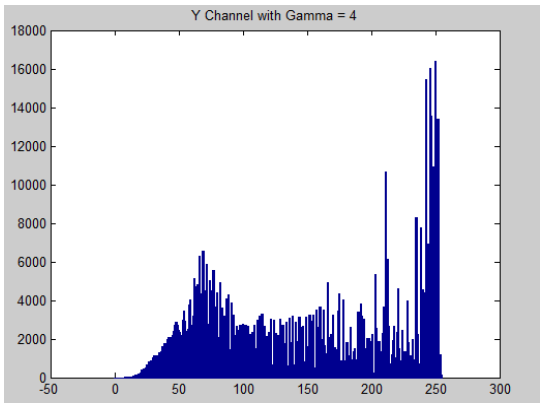
```
RGB = theta\[Y;I;Q];
RGB = cat(3, RGB(1, :), RGB(2, :), RGB(3, :));
RGB = reshape(RGB, w, l, 3);
```

Results

下圖為最初 YIQ 轉換後的結果，以及其 Y channel 的分布圖。



下圖為 YIQ 經過 Gamma(r=4)轉換後的結果，以及其 Y channel 的分布圖。



下圖為原圖以及經過上述強化步驟之後的結果。

Discussions

原圖太亮了，濃度也太淡了，比較不像是陽光太強或是有薄霧籠罩，而是接近浮水印的程度，轉成 YIQ 後發現原圖的 pixel 大部分都集中在 150 以上，程式碼裡有一行 gt150 = sum(Y>150); 即是在計算 Y channel 裡高於 150 的 pixel 數，總共是 615900 個 pixel，占了 98.8%，因此原圖看起來非常亮。

經過 Gamma 轉換(r=4)，Y channel 裡高於 150 的 pixel 數為 305423 個 pixel，降到了 48.99%，高亮度的 pixel 分布較為平均，降低了亮度，但因 I、Q channels 沒有改變，所以色彩不會受影響。

試了幾個 Gamma 值，大概 r=3~6 都是滿正常的，但 r=7 之後就有些太深了。

Execution

直接在 Command Window 輸入 hw1_1，並確定程式與圖檔在同個目錄下即可，執行完畢應出現四張影像及兩張長條圖，而高於 150 的 pixel 數及所占百分比會直接在 Command Window 輸出。

我將此題結果輸出成 JPG 檔並附在資料夾中：YIQ.jpg, YIQ_gamma.jpg, Final.jpg。

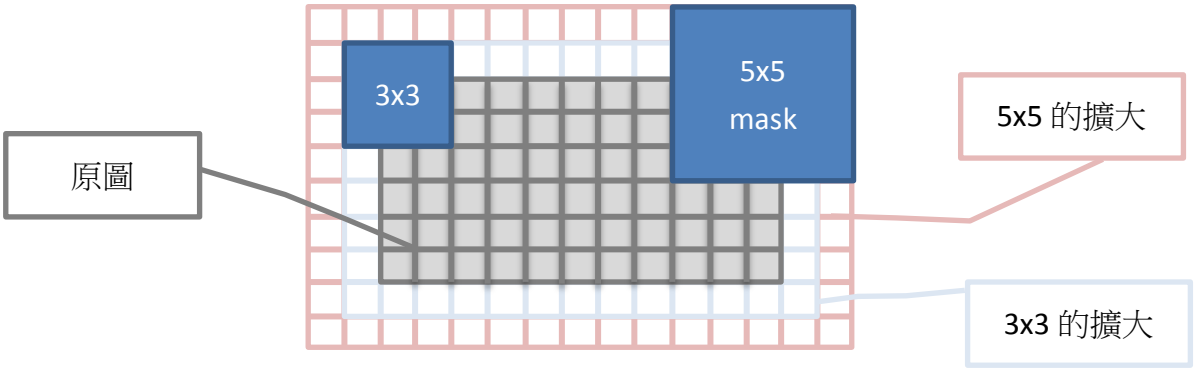
2. Image Filter and Convolution Masks

Method Description

設定所有需要的 masks，而 Gaussian Blur 3x3, 5x5 的 mask 先存分子，分母則在正式處理影像時再除。

```
%------define all masks-----%
GB3 = [1 2 1;2 4 2;1 2 1];% Gaussian Blur 3x3
GB5 = [1 4 6 4 1;4 16 24 16 4;6 24 36 24 6;4 16 24 16 4;1 4 6 4 1];% Gaussian Blur 5x5
EDx = [1 1 1;0 0 0;-1 -1 -1];% Edge Detection x Direction
EDy = [-1 0 1;-1 0 1;-1 0 1];% Edge Detection y Direction
USp = [0 -1 0;-1 5 -1;0 -1 0];% Unsharp
```

用一個 tmp 矩陣存原圖並擴大四邊的值來做 filtering，先將原圖的值放入 tmp 的中間，然後再將四邊分別複製，就能完成 3x3 mask 需要的擴大，另外 5x5 的則是用已經算好的 3x3 再擴大，即可省去判斷邊界 pixel 的問題。




```
%------replcate edge value for GB3-----%
tmp = zeros(ra+2, ca+2);
IMa3 = zeros(ra, ca);
for i=2:ra+1
   tmp(i, 2:ca+1) = IMa(i-1, :);
end
tmp(:, 1) = tmp(:, 2);
tmp(:, end) = tmp(:, end-1);
tmp(1, :) = tmp(2, :);
tmp(end, :) = tmp(end-1, :);
```

實作 filtering 的方法都差不多，用四個 for 迴圈，前兩個 for 迴圈選擇目前要處理的 pixel，後兩個 for 迴圈則將 mask 的值乘上原圖的 pixel 值並存入新的矩陣中。

實作 Gaussian Blur 時要在存入新矩陣時將新的值除以 16、256。

實作 Edge Detection 時要判斷如果新的 pixel 小於 0 則設成 0，大於 255 則設成 255。

資工系 100062236 林修安

```
%------implement GB3-----%
for i=2:ra+1
    for j=2:ca+1
        sum = 0;
        mi = 1;
        for ii=i-1:i+1
            mj = 1;
            for jj=j-1:j+1
                sum = sum + tmp(ii, jj)*GB3(mi, mj);
                mj = mj + 1;
            end
            mi = mi + 1;
        end
        IMa3(i-1, j-1) = sum/16;
    end
end
```

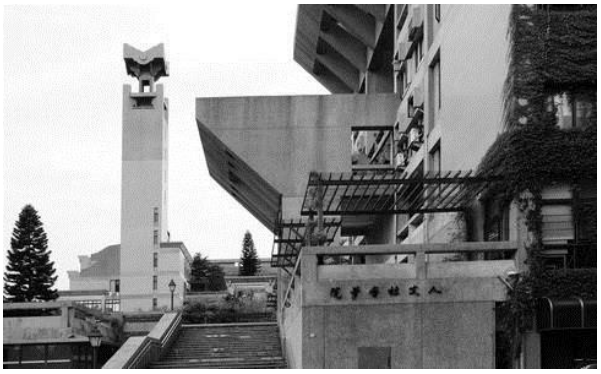
```
%------implement EDx-----%
for i=2:rb+1
    for j=2:cb+1
        sum = 0;
        mi = 1;
        for ii=i-1:i+1
            mj = 1;
            for jj=j-1:j+1
                sum = sum + tmp(ii, jj)*EDx(mi, mj);
                mj = mj + 1;
            end
            mi = mi + 1;
        end
        if(sum>255) sum=255;
        else if(sum<0) sum=0;
            end
        end
        IMb(i-1, j-1) =  sum;
    end
end
```

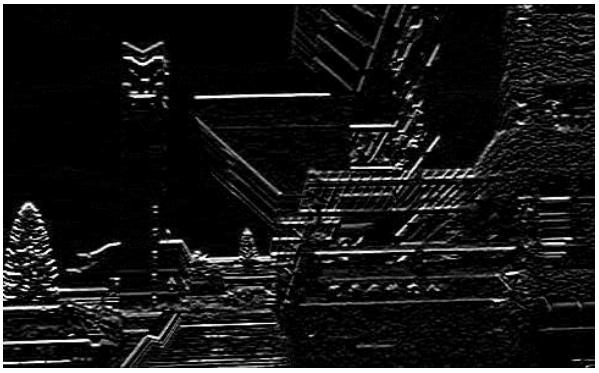
Results & Discussions

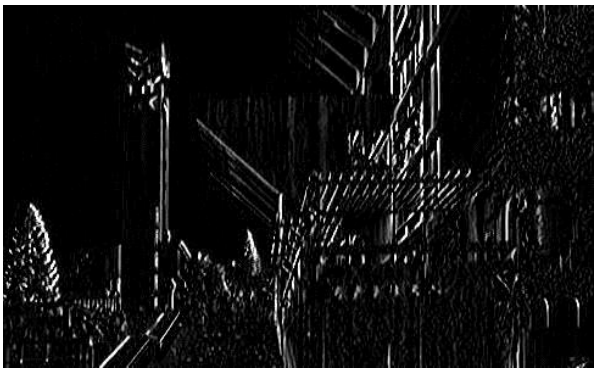
左圖為原圖，加上黑白雜訊的沉思者圖片，中間為經過 3x3 Gaussion Blur 處理後的結果，放大之後差別更明顯，處理過後的圖片中的雜訊圓潤多了，因為模糊的關係所以雜訊沒那麼銳利。右圖為經過 5x5 Gaussian Blur 處理後的結果，雜訊更融入背景了，有種被手抹平的質感，但也可以明顯地看出雕像底座的邊緣被模糊化了，也可以從後方建築物的窗戶看出來，跟原圖的銳利度有明顯差別。



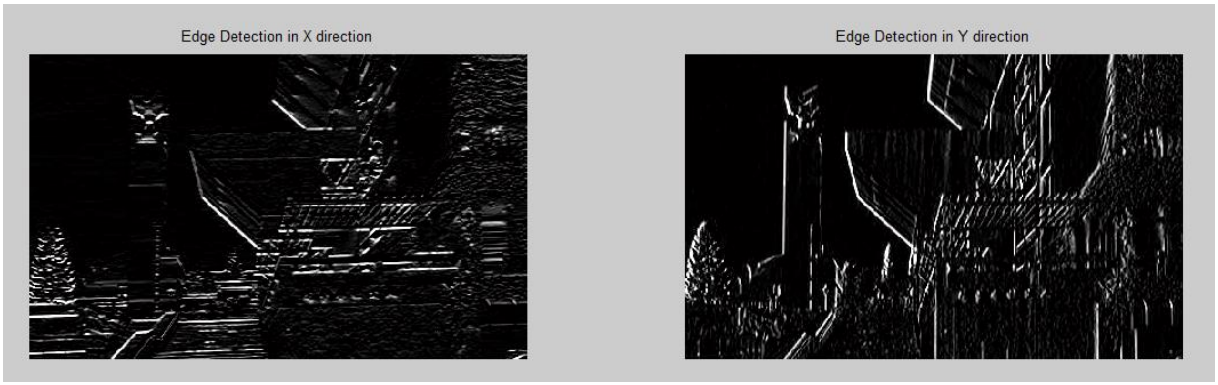
中間為人社鐘塔原圖，左下是經過 X 方向 Edege Detection 處理後的結果，將所有水平走向的線條都勾勒出來，大概就是下雨的時候會第一時間碰到雨水，向上的面。而右下圖是 Y 方向 Edge Detection 處理後的結果，垂直走向的線條也清晰可見，如果太陽在鐘塔及右邊凸出部分的中間，其勾勒出的線條就會是被太陽照亮的面，即鐘塔的右側與人社院的左側。



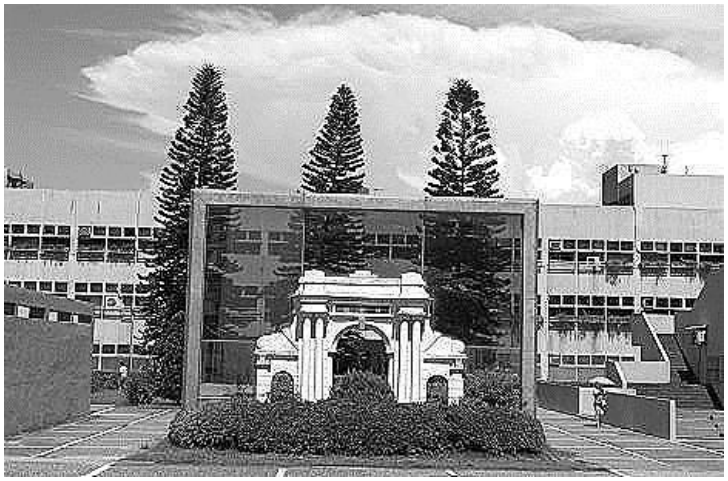




我把 mask 的 1 與-1 全部調換再處理一次，左圖則勾出下方的水平線條，而右圖勾出的垂直線條都是左面的線條。



左圖為原圖，右圖為 Unsharp 處理後的結果，圖中的樹木以及草叢很明顯地連葉片的邊緣都被強化了，原圖的樹枝與背景的界線是模糊的、連續的，但銳利化之後，樹枝與背景的界線好像被一分為二，樹枝周圍有一圈比背景淺色的邊線，使得顏色之間的分界更清楚。



Execution

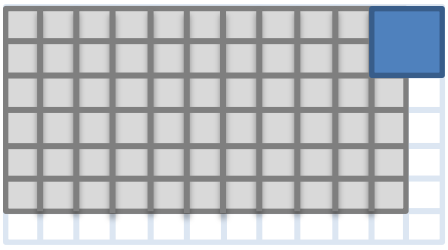
同上執行 hw1_2，但因為我一次處理並輸出八張圖，所以程式執行時間需要至少一分鐘，請耐心等待。

我將此題結果輸出成 JPG 檔並附在資料夾中：3x3.jpg, 5x5.jpg, edgex.jpg, edgey.jpg, unsharp.jpg。

3. Image Interpolation

Method Description

我選擇第一個 channel 來實作，並將其向下、向右擴大，以方便進行 Bilinear Interpolation，並宣告兩個長、寬皆四倍的矩陣來存結果。



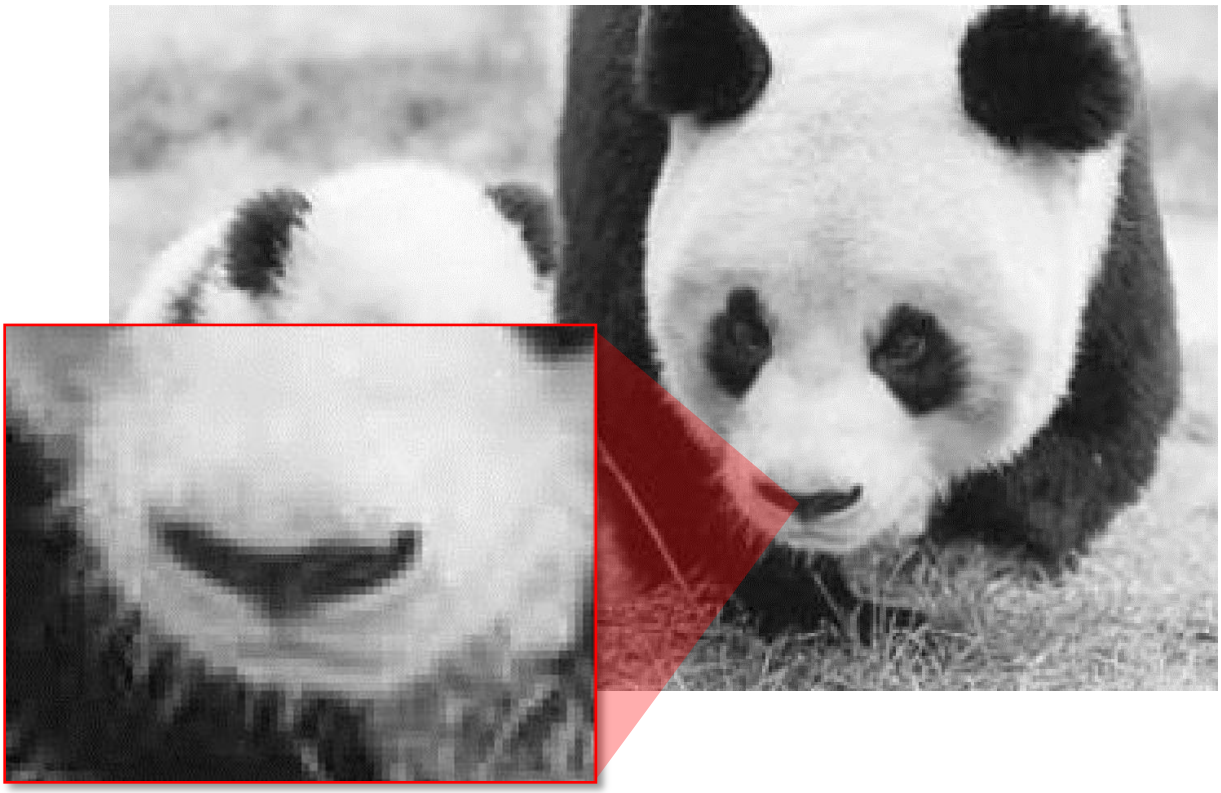
```
X = X(:, :, 1);
[w, l] = size(X);
X(w+1, :) = X(w, :);
X(:, l+1) = X(:, l);
W = 4*w; L = 4*l;
NN = zeros(W, L);
BI = zeros(W, L);
```

接著就照著兩種方法各自的規則處理影像，因為 i, j 除以 4 之後四捨五入或是取 floor 都有可能為 0，故如果為 0 則設為 1。

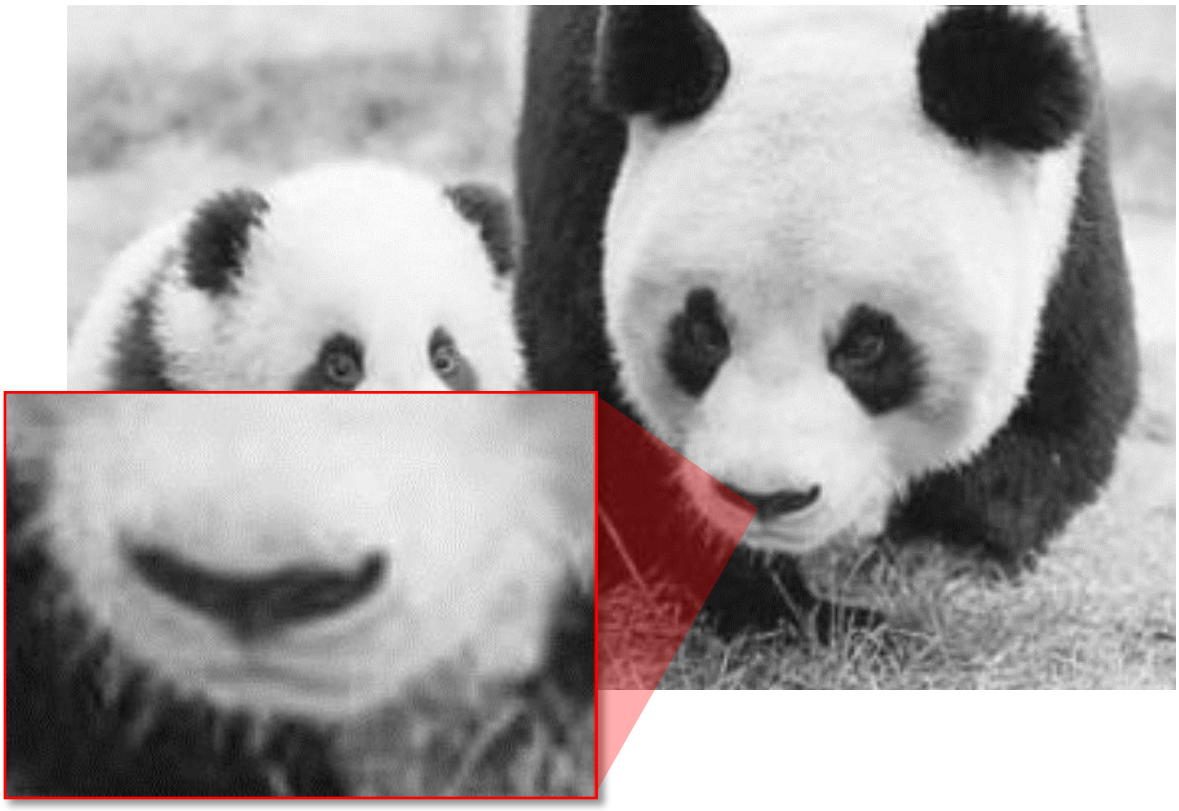
```
for i=1:W
    for j=1:L
        a = i/4;
        b = j/4;
        NNi = round(a); NNj = round(b);
        BIi = floor(a); BIj = floor(b);
        if(NNi==0) NNi = 1; end
        if(NNj==0) NNj = 1; end
        if(BIi==0) BIi = 1; end
        if(BIj==0) BIj = 1; end
        NN(i, j) = X(NNi, NNj);
        for ii=BIi:BIi+1
            for jj=BIj:BIj+1
                BI(i, j) = BI(i, j) + X(ii, jj)*abs((1-abs(a-ii))*(1-abs(b-jj)));
            end
        end
    end
end
```

Results & Discussions

下圖為 Nearest Neighbor 方法得出的結果，圖中熊貓的眼睛、鼻子、嘴巴周圍仍然有一格一格的馬賽克現象，放大之後更為明顯，但於一般大小之下不影響整體圖片的品質。



下圖為 Bilinear 處理後得出的結果，可以看出黑色區塊與白色區塊之間的邊界柔和了很多，放大後會顯得較模糊，但是沒有馬賽克的現象，保持了毛髮柔軟的質感。



Execution

同上執行 hw1_3。

我將此題結果輸出成 JPG 檔並附在資料夾中：NearestNei.jpg, Bilinear.jpg。