

# Homework #1 Report

102062111 林致民

## 1. Demosaicing

首先，題目要求要使用Nearest Neighbor Algorithm 去把圖像解析出來，那麼我的做法是，在當前點 $(i, j)$ ，考慮兩種情形：當前點是綠色 & 當前點不是綠色。

假設當前點是綠色，那麼我們就不用去計算當前點的綠色值，我們僅需要計算藍色直還有紅色值。紅色和藍色圍繞在綠色當前點最多只有兩個，然後又因為要馬分佈在『上下』，要馬分不在『左右』，所以當前點的藍色值和紅色會等於 $(上 + 下) / 2$  或者 $(左 + 右) / 2$ ，然後把結果填進去就可以了。

假設當前點是紅色或藍色，綠色一定是分佈在當前點的上下左右，所以當前點的綠色值 =  $(上 + 下 + 左 + 右) / 4$ ，不管當前點是藍色點還是紅色點。然後再從中考慮當前點到底是紅色還是藍色，假設是藍色就不用去計算藍色值，然後去計算周圍四個角落紅色的值 =  $(右上 + 左上 + 右下 + 左下) / 4$ ，同理，假設當前點是紅色就不用去計算紅色的值，然後去計算周圍四個角落藍色的值 =  $(右上 + 左上 + 右下 + 左下) / 4$ 。

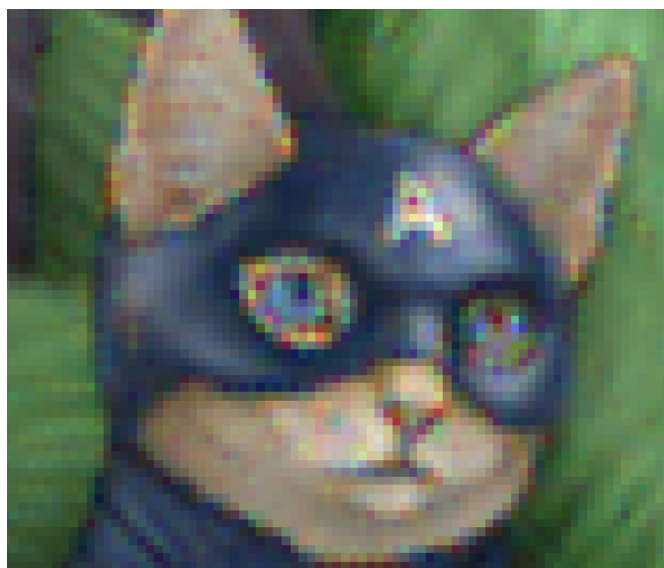
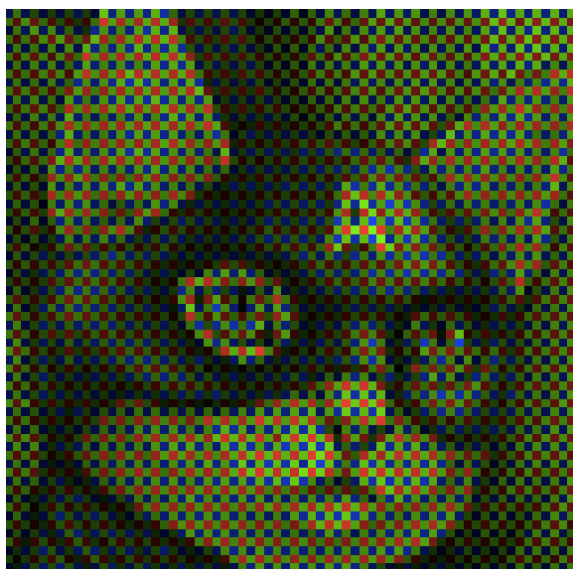


原始圖片

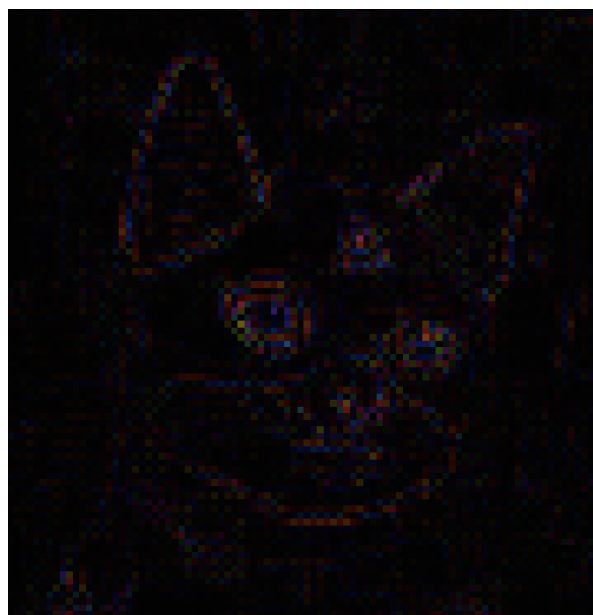
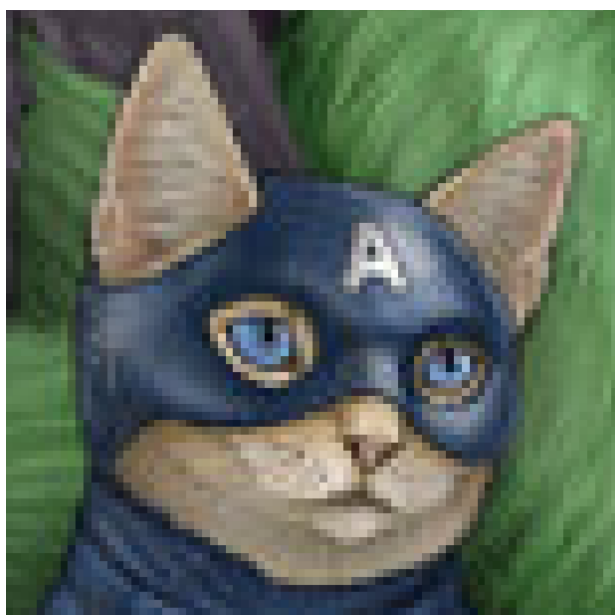


經過處理

原始圖片與處理後的圖片細部比較：



Absolute Difference :



未經過MASK的圖片

Absolute Difference

然後還有一些點發現到，如果把圖片存成JPG會比較接近原來的圖片，然後把圖片存成png會變成上圖（右）的那樣，覺得挺神奇的。

PSNR 大概接近30.8左右，圖像壓縮中典型的峰值信噪比值在 30 到 40dB 之間，愈高愈好。

## 2. Dithering

(a) Thresholding (我選擇取平均是因為這樣看起來會讓圖片相對漂亮一點)

首先，我取Threshold的方式僅僅是把所有的value加起來平均 (Average Dithering)，然後去檢查每一個pixel的value，假設當前value我所選的Threshold還小，那就讓它 = 0，反之，讓其value = 1。



助教給的圖片



我選的圖片

(b) Error Diffusion

首先，就是對每一點做以下這些事情：

1. 與當前 $(i, j)$ 點比較，其右邊的點  $img(i, j + 1) = img(i, j + 1) + e * (7/16)$
2. 其左下角的點  $img(i + 1, j - 1) = img(i + 1, j - 1) + e * (3/16)$
3. 下面的點  $img(i + 1, j) = img(i + 1, j) + e * (5/16)$
4. 右下角的點  $img(i + 1, j + 1) = img(i + 1, j + 1) + e * (5/16)$

對每個做完一次後pixel，然後去再掃描一次，假設當前點的value > 0.5 (以我計算的例子來說，我是以double的形式在處理image)，我就讓這個點的value = 1，反之value = 0。當然啦，做出來的結果和Thresholding來比，看起來順眼多了，也能夠比較能表達比較細部的顏色，如下兩張圖：





然後在比較一下我選擇的圖片和原始圖片的差別：



然後放大的情況下作比較（Thredsholding vs Error Diffusion）：



會發現到Error Diffusion 的顆粒比較小，比較細緻，當圖片縮小或，會以密集稀疏來表達顏色的深淺，這看起來是相當有趣的實踐。

### 3. Interpolation 對原始圖片放大四倍

(a)的做法就只是mapping到原始圖片的顏色。(b)的話是使用雙線性內插來處理圖片，他的精神其實在於離比較點越近，他對圖片顏色的影響力就越大。稍微比較一下兩者，其實會發現到，(a)的做法會有比較明顯的鋸齒狀，而(b)的做法處理出來的圖片相對於(a)比較平滑，不過兩者都還是有個共同點在於放大後圖片還是一樣有點模糊，效果沒有想像中的那麼好。

