高等影像處理

作業#2

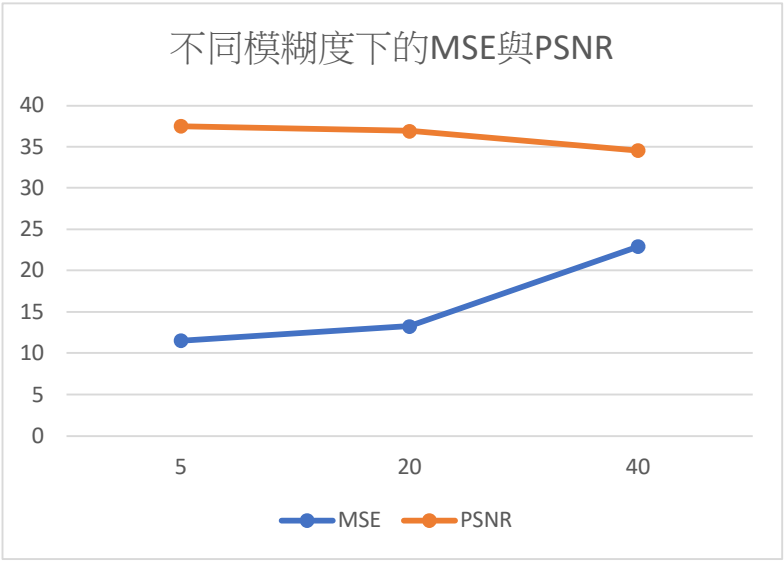
姓名： 顏郁芩

學號： 110368155

指導老師： 高立人 教授

1_(2)

Disgussion　1_(2)比較模糊過的影像和原本影像的 MSE 與 PSNR



模糊度			
	5	20	40
MSE	11.53	13.25	22.9138
PSNR	37.51	36.91	34.5298

Discussion 1_(2)分析與說明

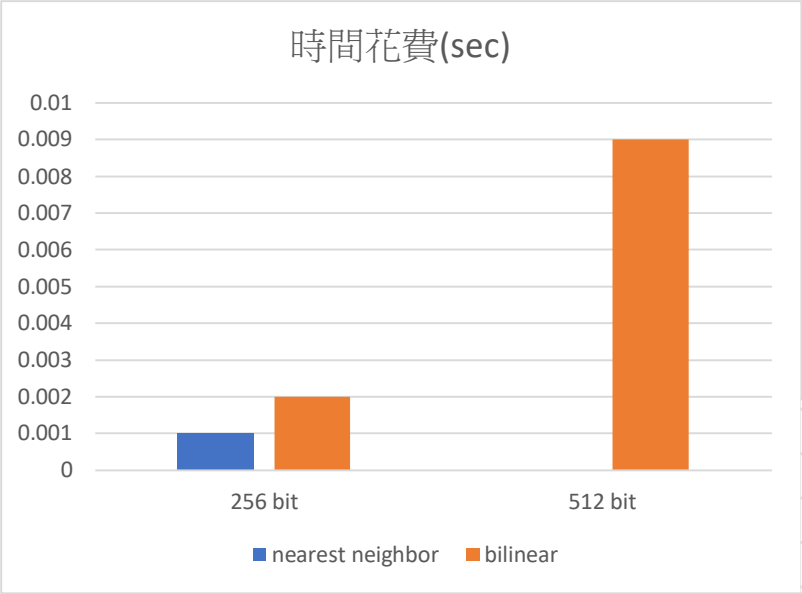
計算原影像與模糊後影像的 MSE 均方誤差(mean-square error)和 PSNR 峰值訊噪比（Peak signal-to-noise ratio），MSE 越小越好而 PSNR 多半會介在 30dB 到 50dB 之間且越高越好，大於 30dB 人眼會難以察覺和原始影像差異。(來源：維基百科)

從實驗結果的圖可以發現，模糊度越高，其 MSE 會越大(和原圖差異越大)，而 PSNR 也會隨著模糊度增加而變小(較能辨明差異)。

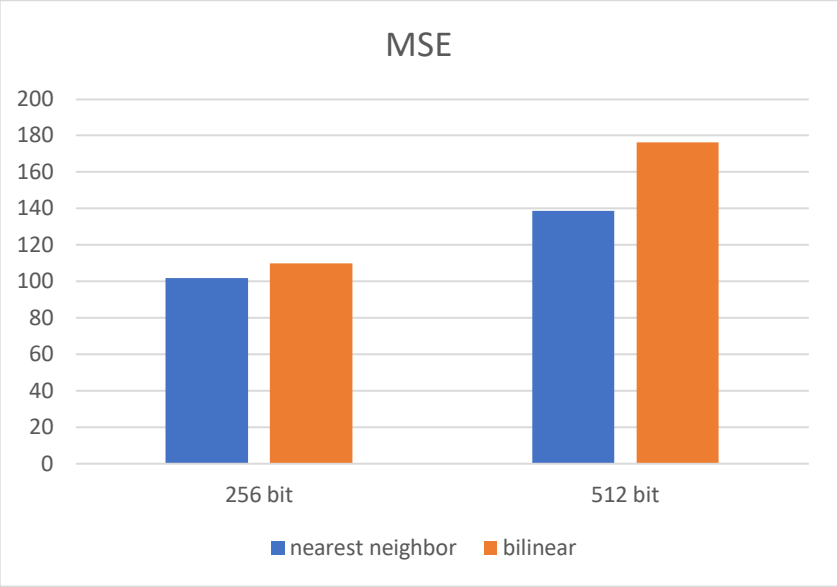
2_(1)
Figure
Nearest neighbor

bilinear

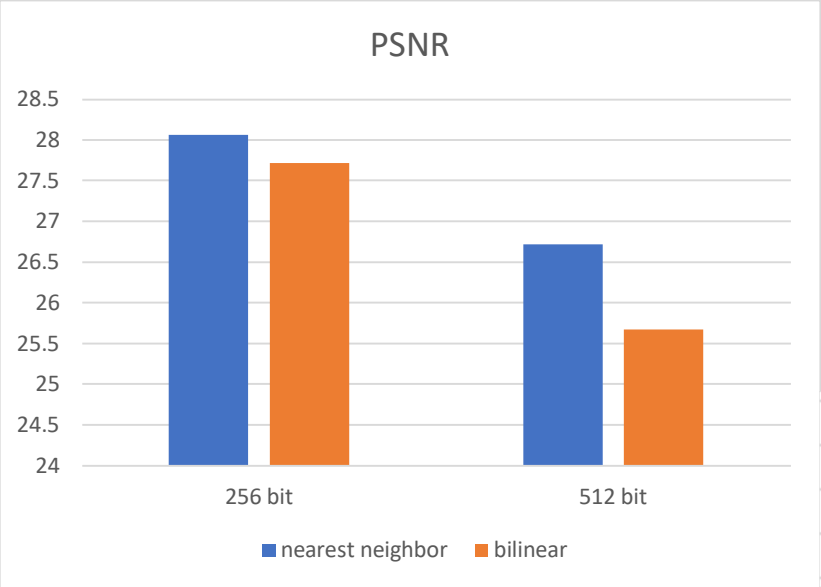
Discussion 2_(1) calculate the MSE and PSNR, and discuss the execution time



時間花費(sec)		
	256 bit	512 bit
nearest neighbor	0.001	0
bilinear	0.002	0.009



MSE		
	nearest neighbor	bilinear
256 bit	101.6489	109.9568
512 bit	138.5012	176.2669



PSNR		
	nearest neighbor	bilinear
256 bit	28.0597	27.7185
512 bit	26.7162	25.669

Discussion 2_(1)分析與說明
1.肉眼可見圖像結果：用 Nearest neighbor 的結果會比用 Bilinear 放大的棋盤化更嚴重 2.時間消耗：從實驗結果可見，用 Bilinear 放大所消耗的時間比用 Nearest neighbor 放大更多 3.MSE：從實驗結果可見，用 Bilinear 放大的 MSE 比用 Nearest neighbor 放大的 MSE 值更大 4.PSNR：從實驗結果可見，用 Bilinear 放大的 PSNR 比用 Nearest neighbor 放大的 PSNR 值更小

2_(2)

Figure

Nearest neighbor 原圖縮小

lena_S_out_128_nearest.raw

lena_S_out_256_nearest.raw

Bilinear 原圖縮小

lena_S_out_128_bilinear.raw

lena_S_out_256_bilinear.raw

Nearest neighbor 模糊 20 之後縮小

lena_S_out_128_nearest_20.raw lena_S_out_256_nearest_20.raw

Bilinear 模糊 20 之後縮小

lena_S_out_128_bilinear_20.raw lena_S_out_256_bilinear_20.raw

Nearest neighbor 模糊 40 之後縮小

lena_S_out_128_nearest_40.raw

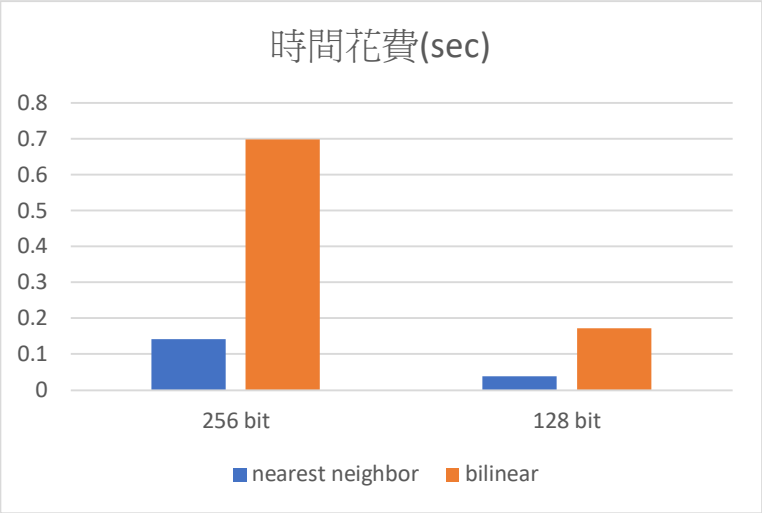
lena_S_out_256_nearest_40.raw

Bilinear 模糊 40 之後縮小

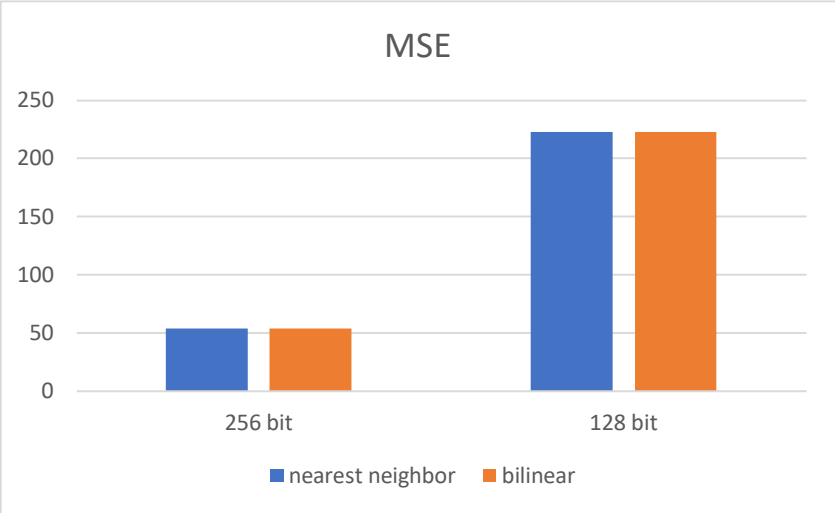
lena_S_out_128_bilinear_40.raw

lena_S_out_256_bilinear_40.raw

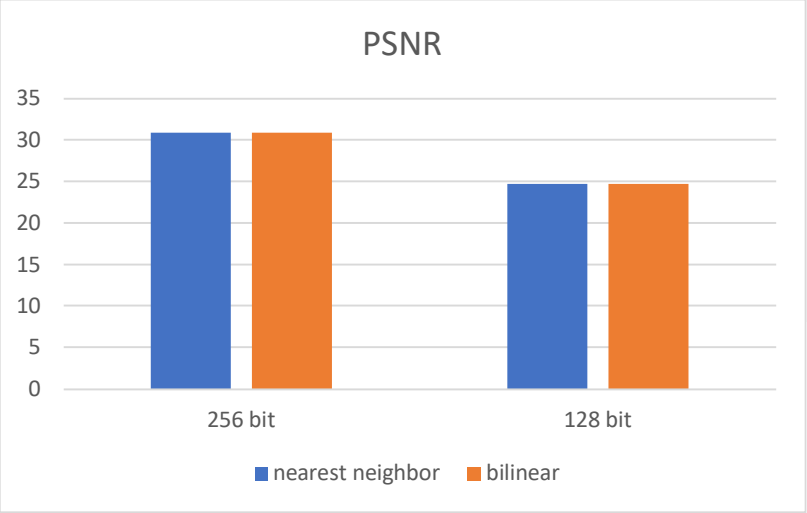
Discussion 2_(2) calculate the MSE and PSNR, and discuss the execution time



花費時間(sec)		
	256 bit	128 bit
nearest neighbor	0.141	0.038
bilinear	0.698	0.172

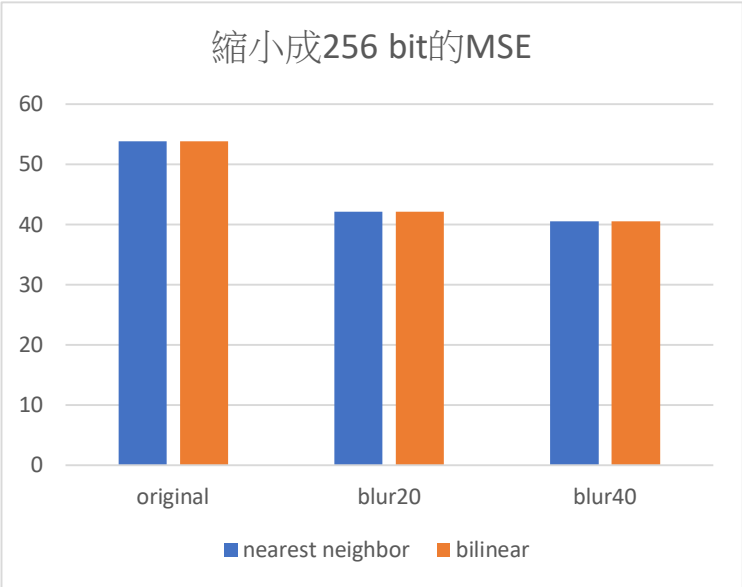


MSE		
	nearest neighbor	bilinear
256 bit	53.7909	53.7909
128 bit	222.5766	222.5766

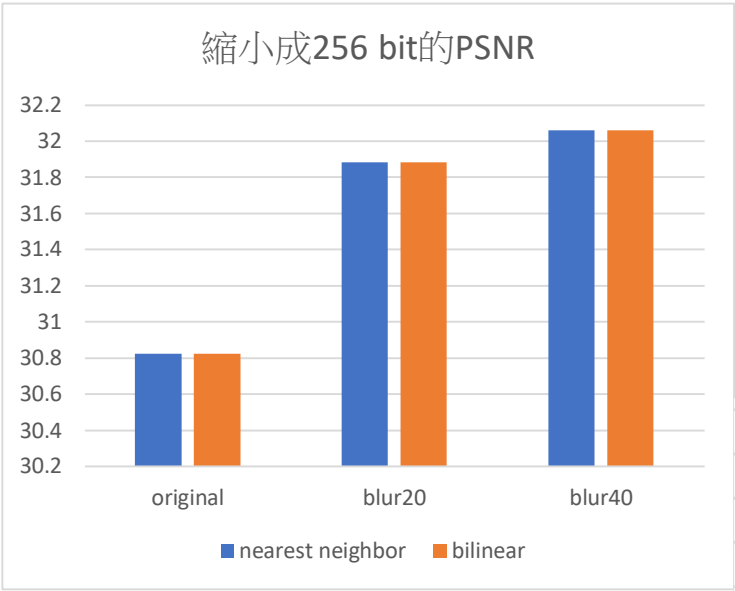


PSNR		
	nearest neighbor	bilinear
256 bit	30.8237	30.8237
128 bit	24.656	24.656

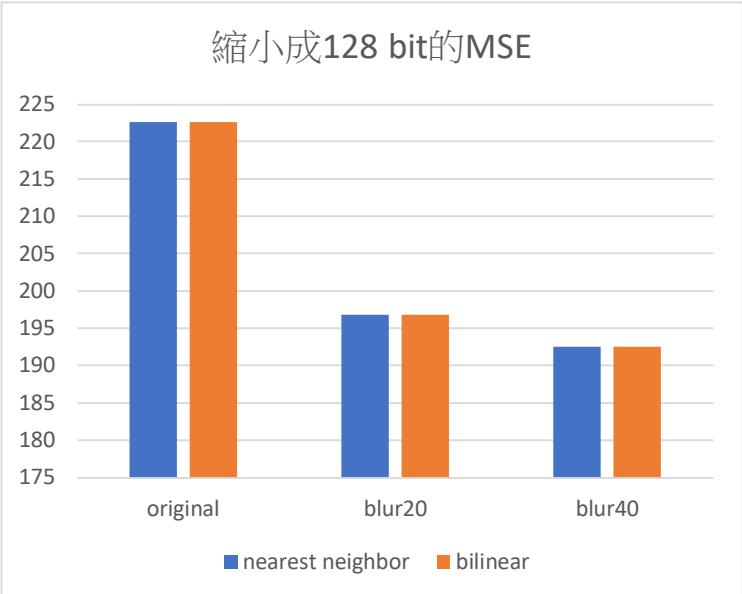
Discussion 2_(2) check the result with or without before resizing



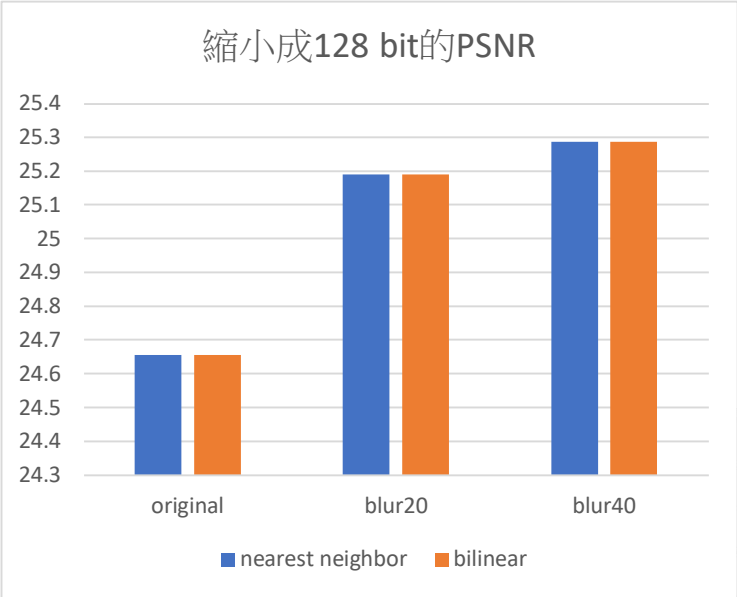
縮小成256 bit的MSE			
	original	blur20	blur40
nearest neighbor	53.79	42.13	40.4582
bilinear	53.79	42.14	40.4582



縮小成256 bit的PSNR			
	original	blur20	blur40
nearest neighbor	30.8237	31.884	32.0607
bilinear	30.8237	31.884	32.0607



縮小成128 bit的MSE			
	original	blur20	blur40
nearest neighbor	222.6	196.8	192.476
bilinear	222.6	196.8	192.476



縮小成128 bit的PSNR			
	original	blur20	blur40
nearest neighbor	24.656	25.1903	25.287
bilinear	24.656	25.1903	25.287

Discussion 2_(2)分析與說明

1.時間消耗：從實驗結果可見，用 Bilinear 縮小所消耗的時間比用 Nearest neighbor 縮小更多

2.MSE 和 PSNR：因為縮小整數倍的話用 Bilinear 和 Nearest neighbor 的結果會相同，從實驗結果也可見，兩者的數值是相同的。

3.模糊過後再縮小：利用了兩種模糊程度(20 和 40)來做比較，從實驗結果可見，MSE 會隨模糊程度變大而降低，PSNR 則是會隨著模糊程度變大而縮小，此數值可以證明先模糊再進行縮小的效果會比直接用原圖縮小來得好。

2_(3)

Figure

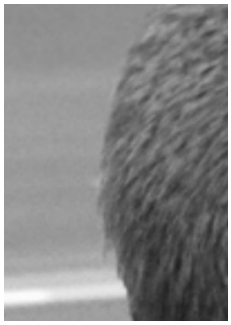
(原圖顯示縮小三倍)

duck_range.raw duck_biliner.raw duck_bicubic.raw



(原圖顯示縮小 6 倍)

duck_range.raw duck_biliner.raw duck_bicubic.raw


Discussion 2_(3) Discuss the difference from two algorithms

利用 biliner 和 bicubic 兩種方式來放大同一區塊的圖片，第一個 duck_range.raw 是需要放大的原區域圖，duck_biliner.raw 和 duck_bicubic.raw 是結果，為了方便判斷所以顯示用了兩種不同方式。
但檔案 duck_biliner.raw 和 duck_bicubic.raw 都是 duck_range.raw 放大 5 倍的結果。
從上面的圖片可看出，bicubic 方式比較容易看出細節，但是顯示放大超過一定程度的時候會造成像素點很明顯，而且在放大的時候需要注意 overflow 問題。Biliner 的放大結果較為模糊，細節也不明顯，但是顯示放大時像素點不會像 bicubic 放大那麼明顯。
在需要觀察細節的場合，用 bicubic 放大會比 biliner 放大來得好。

3

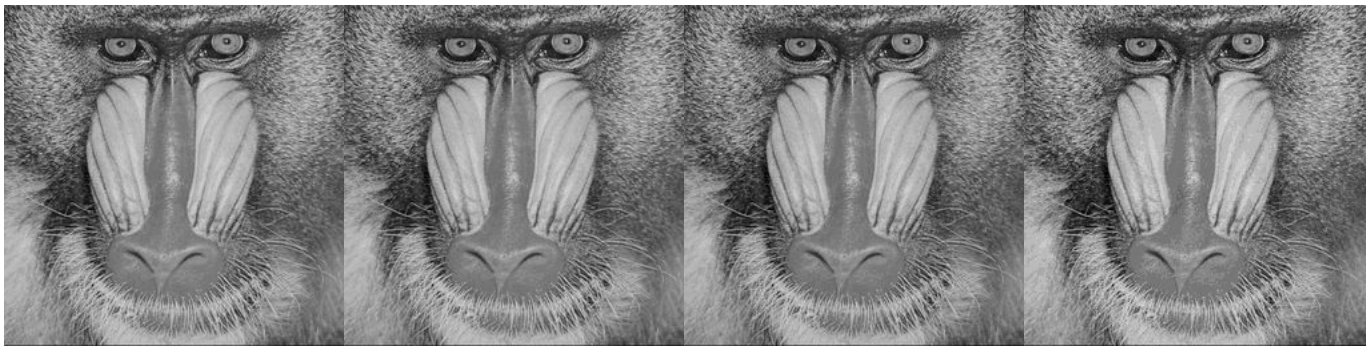
Figure lena256

lena256_7.raw lena256_6.raw lena256_5.raw lena256_4.raw

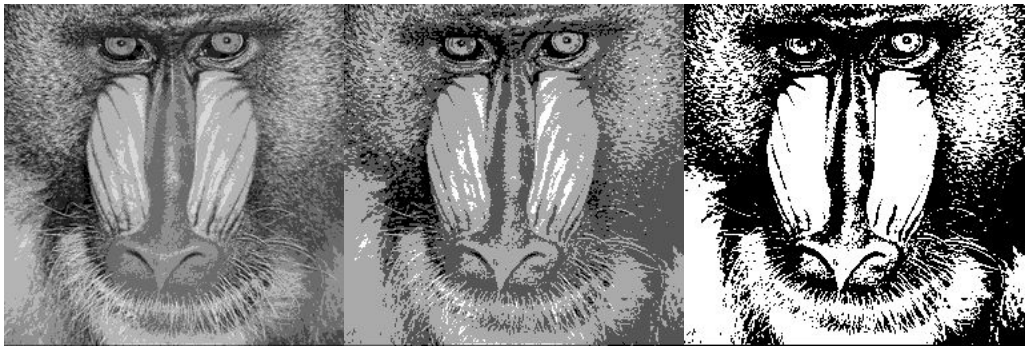
lena256_3.raw lena256_2.raw lena256_1.raw

Figure baboon256

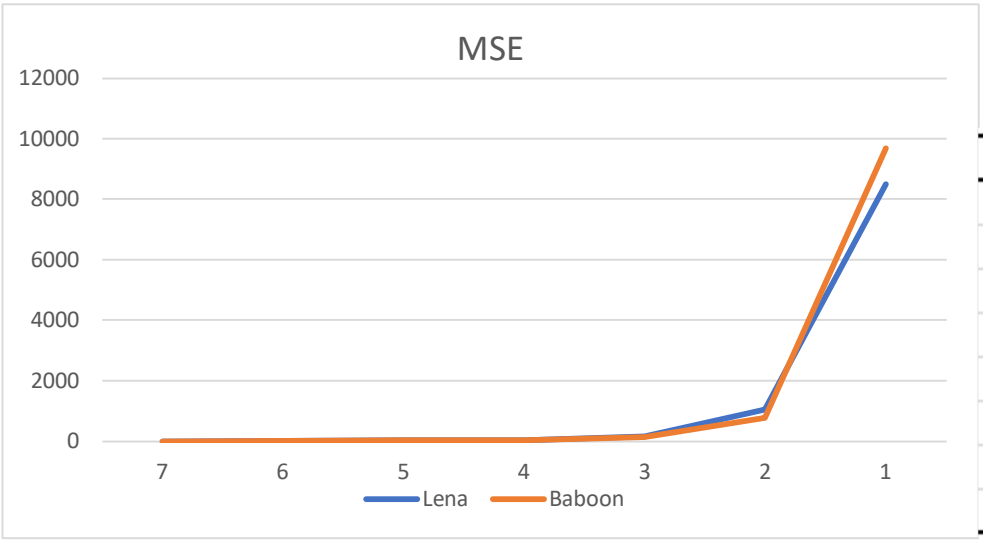
baboon256_7.raw baboon256_6.raw baboon256_5.raw baboon256_4.raw



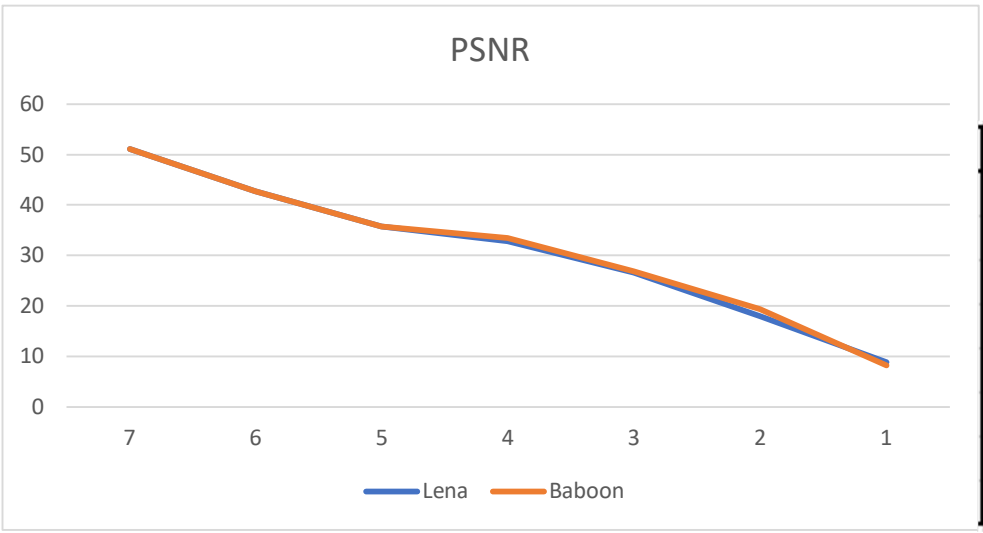
baboon256_3.raw baboon256_2.raw baboon256_1.raw



Discussion 3 calculate MSE and PSNR



MSE		
	Lena	Baboon
7 bit	0.5006	0.504
6 bit	3.494	3.521
5 bit	17.4598	17.49
4 bit	33.8709	29.47
3 bit	142.859	133.1
2 bit	1038.54	759.2
1 bit	8494.11	9679



PSNR		
	Lena	Baboon
7 bit	51.1359	51.11
6 bit	42.6976	42.66
5 bit	35.7104	35.7
4 bit	32.8325	33.44
3 bit	26.5817	26.89
2 bit	17.9665	19.33
1 bit	8.8396	8.273

Discussion 3 discuss the bit rate saving

Lena 和 Baboon 兩張圖的差異是細節程度不同。

從 MSE 的結果可以看出，Lena 的 MSE 在 bit rate 降低的過程中會比 Baboon 的 MSE 升高速度來得快，大概在 4bit 的時候就會明顯超過 Baboon。

而從 PSNR 的結果可以看出，雖然兩者數據相差不大但是 Baboon 的 PSNR 數據在 bit rate 降低的過程中會比 Lena 稍微高一點點。

由實驗數據可以證明 Baboon(細節比較多)的圖在縮小 bit rate 的時候結果會比 Lena(細節比較少)的要來得好一些，也就是說細節比較多的圖可以縮小比較多 bit rate 而不會影響到輸出結果太多。

4
Figure
Discussion 4 using D_4,D_8,D_m distance to find shortest path

(因為程式碼沒有完成所以只有把想法寫在這邊，程式部分標示未完成)

D_4,D_8,D_m distance 各有自己的判斷方式所以想法是以原本的 pixel 做為中心，往外畫九宮格然後每一格設置 flag 來表示可不可以走(可以選擇的路徑)。像是這個樣子

0	3	6
1	4	7
2	5	8

，4 是原本的 pixel。

如果圖長得像

0	3	6
1	4	7
2	5	8

這樣的話，D_4 判斷的時候 5 和 7 的 flag=1，D_8 判斷時 0、5、7 的 flag=1，D_m 判斷時 0、5、7 的 flag=1。

如果圖像是

0	3	6
1	4	7
2	5	8

這樣的話，D_4 判斷的時候 7 的 flag=1，D_8 判斷時 0、6、7 的 flag=1，D_m 判斷時 0、7 flag=1。

利用這個方式來判斷接下來可以選擇的路徑(要扣掉走過來的方向的那個 pixel(flag 要改成 0))，全部可能路徑都走過一次之後從有辦法走到終點的路徑中找路徑最短的當作答案。

想要用 linked list 建立樹但是沒有想出遇到岔路的時候比較好的解決辦法。程式碼裡面的 4.cpp 檔案停留在想利用重複呼叫函數的方式寫但是會造成路徑重複走過的狀態。