HW3

這次作業希望我們實作出 motion estimation 的兩個 methods，並利用計算 SAD(sum of absolute differences)和 PSNR 的方式來分析自己的實作結果。

實作方法：

首先，建立名為 MotionEstimation 的 function，裡面會根據餵入參數決定要呼叫 FullSearch(實作 full search) 或者 TwodLogSearch(實作 2D logarithmic search)的 function，這兩個 function 是實作兩種不同 methods 的地方。

在 MotionEstimation 中，首先我們會利用一個 vector 存入 target frame 的 size，利用雙層迴圈跑過 i = 1:p:size(1)、j = 1:p:size(2)的方式，切割 frame 並給定每次進行 full-search 2D logarithmic search 的 position (i,j)，將此 position 當作參數傳入每次 call 相對應的 method 中，就可以把這個 position 當作 method 搜尋的起始點，並得到回傳的 image 以及 motion vector，完成 MotionEstimation 的目的。

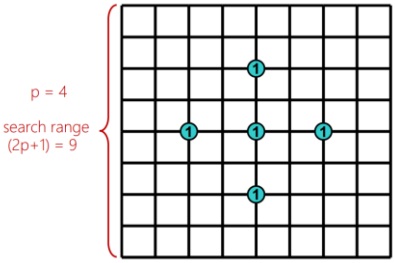
Full search 其實簡單來說，就是暴力的在給定的 range 中，利用 for 迴圈的方式，一次把 reference frame 中的一個 pixel 當作該個 block 的左上角，開始進行 SAD 的計算(與 target frame 比較)，在這裡，我利用講義給的公式來進行 SAD 的計算，其中公式中的 N 代表的是 size of macroblock window。要特別注意的是，在完成這樣的運算後，不要忘記把結果再加上一層 sum()，因為我們現在有 3 個 channel。

$$SAD(\boldsymbol{i},\boldsymbol{j}) = \sum_{k=0}^{N-1}\sum_{l=0}^{N-1}|T(x+k,y+l) - R(x+k+\boldsymbol{i},y+l+\boldsymbol{j})|$$

最後，將我們計算出的 SAD 值與現存的 SAD_min 做比較，當 SAD < SAD_min 時，要將 SAD_min 值更新，並透過找到的點與原本位置相減，在計算 SAD 時順便儲存 vector，計算出 motion vector。

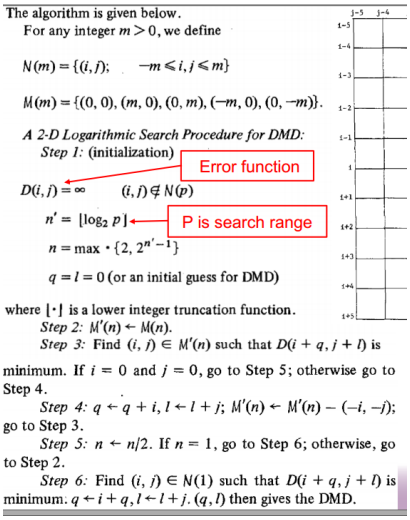
```
choosed_i = x + i;
choosed_j = y + j;
MV(ceil(x/macroblockSize), ceil(y/macroblockSize), 1) = choosed_i - x;
MV(ceil(x/macroblockSize), ceil(y/macroblockSize), 2) = choosed_j - y;
```

2D logarithmic search 號稱是 fast motion estimation algorithm，依據概念，我們首先利用 log(p)決定出一個距離 m，並透過這樣的方式，找出一個包含了自己 position 以及在自己的上、下、左、右距離 m 的五個點，如圖所示。



接下來，根據講義中給的 algorithm，我們就可以完成 2D logarithmic search 的實作。

利用剛剛找出的五個點，與 target 計算 SAD，找出其中的 min SAD 與其相對應的 block，利用 while 迴圈不斷重複，若這個 block 是中心點時，那我們就將 m = m/2；若這個 block 不是中心點時，以此 block 為下一次搜尋的中心點，繼續走相同步長，找出上下左右新的五個 block，重複剛剛的步驟直到 m = 1 為止。

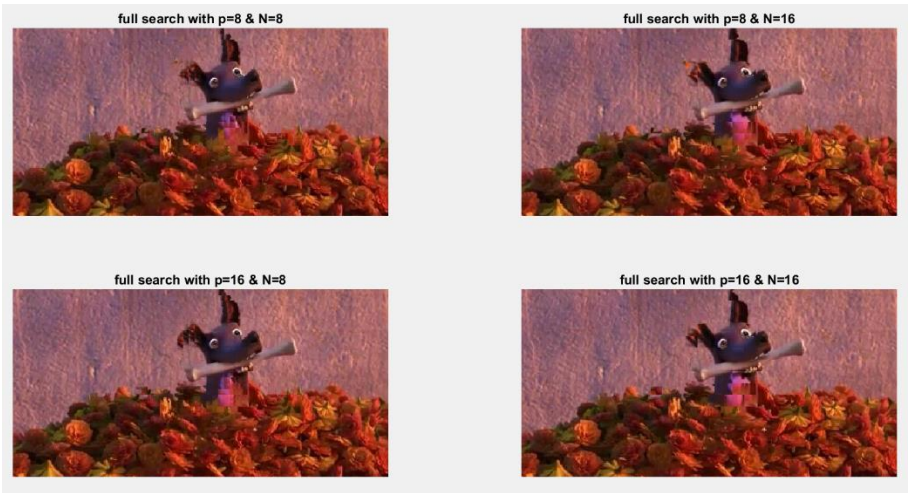


在迴圈中，透過 q = q + I 、 l = l + j 的方式，我們會得到最終要的 q、l 兩個值。接下來的步驟就跟 full search 很像，我們透過雙層迴圈，開始進行 SAD 的計算，並如同 algorithm 中所說，將最後的結果 image 定義成 D(i+q, j+l)，並計算出 motion vector。

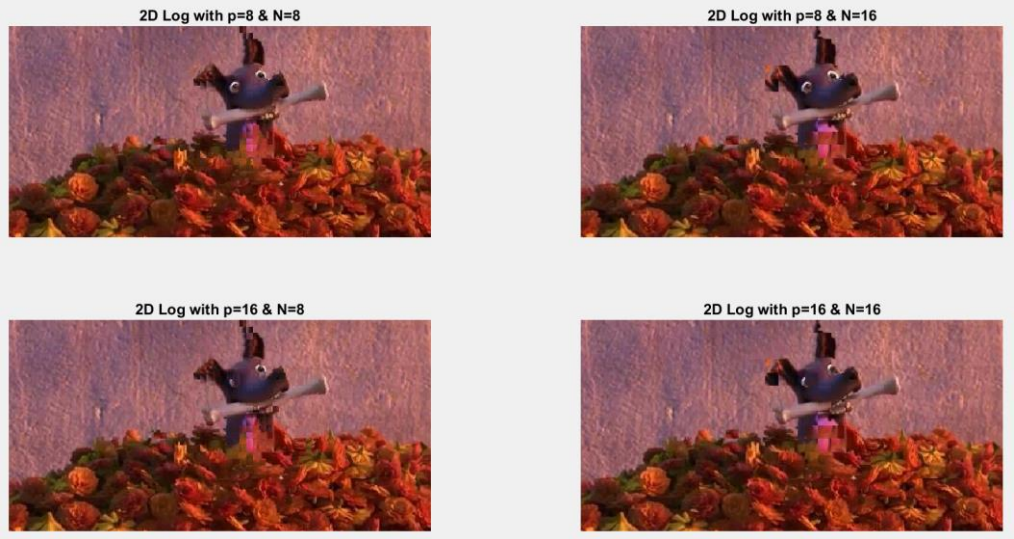
```
x3 = x + q;
x4 = x3 + macroblockSize - 1;
y3 = y + l;
y4 = y3 + + macroblockSize - 1;

img = reference( x3:x4, y3: y4,:);
MV(ceil(x/macroblockSize), ceil(y/macroblockSize), 1) = x3 - x;
MV(ceil(x/macroblockSize), ceil(y/macroblockSize), 2) = y3 - y;
```

實作結果：

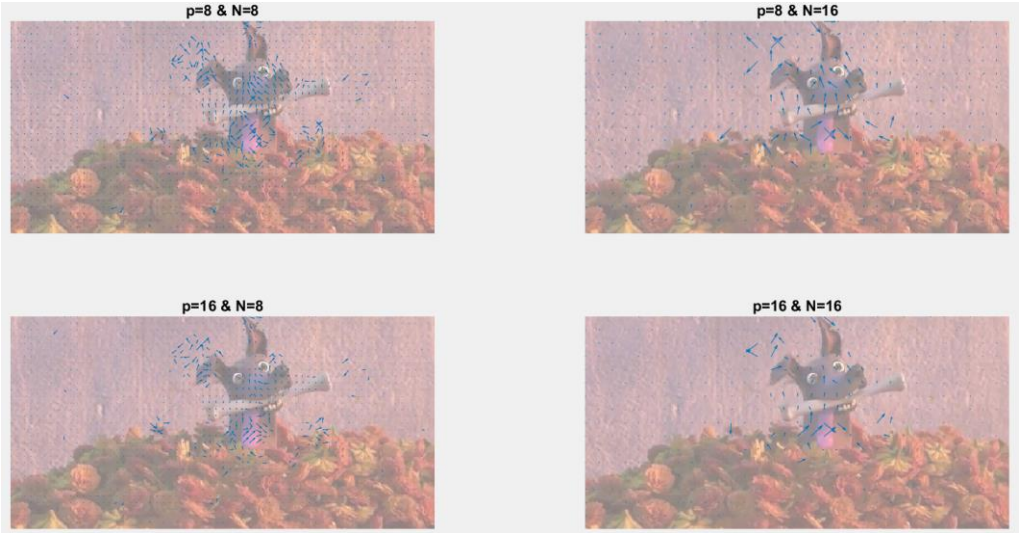
(a). predicted images for full search



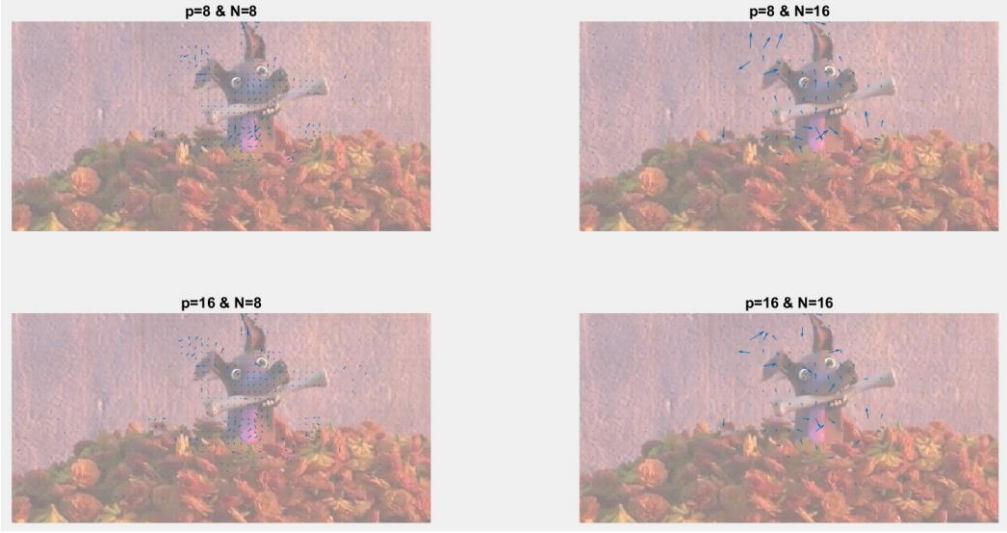
(a). predicted images for 2D logarithmic search



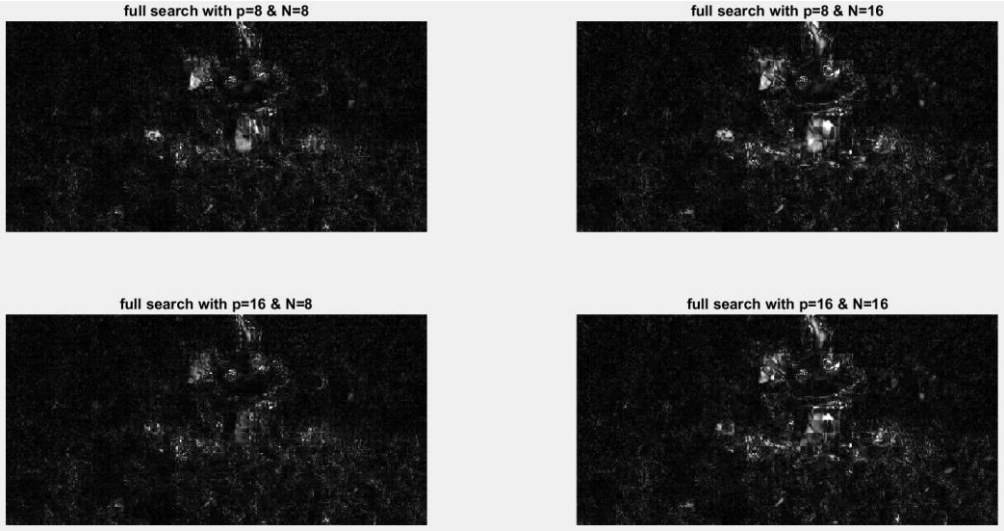
(b). motion vectors images for full search



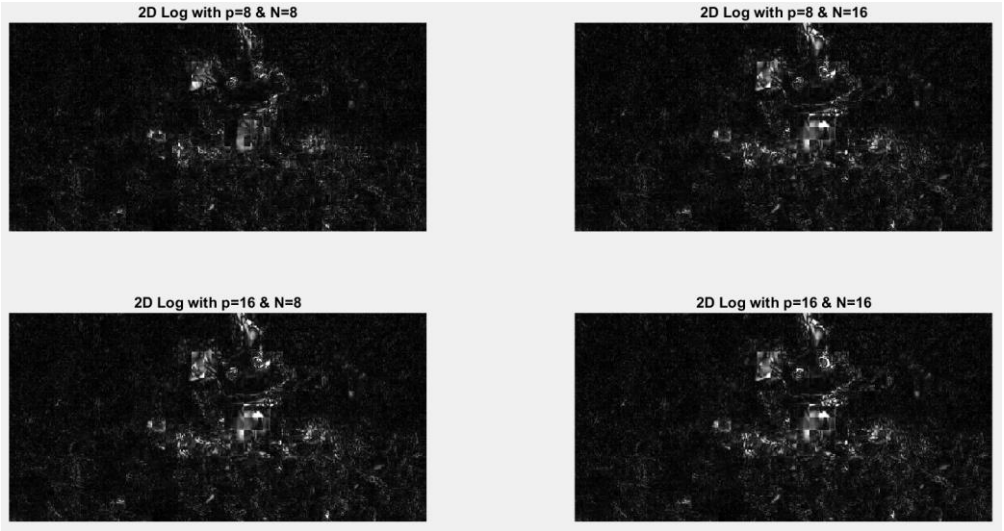
(b). motion vectors images for 2D logarithmic search



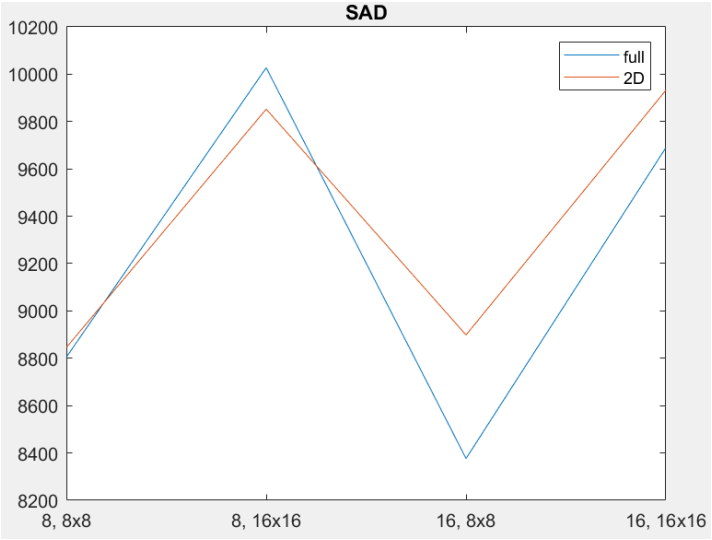
(c). residual images for full search



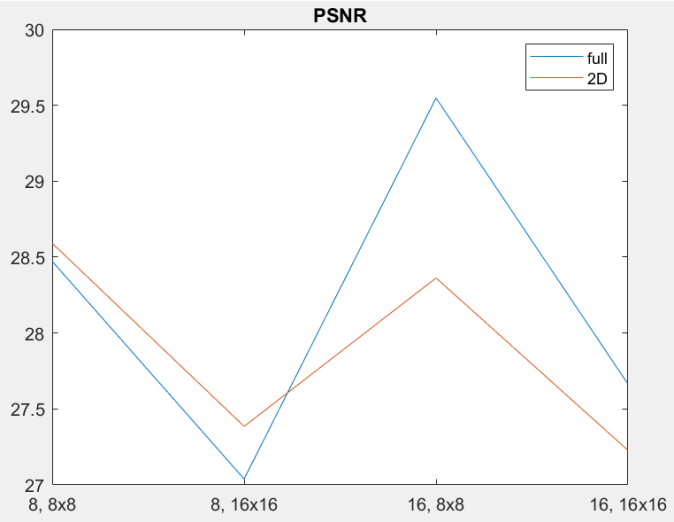
(c). residual images for 2D logarithmic search



(d). SAD



(d). PSNR



Q2:

PSNR for Q1: 28.4729

PSNR for Q2: 22.8481

比較:

SAD 全名 sum of absolute differences，在我們的計算中代表著跟原圖(Target)的差異，因此值愈大代表跟原圖差異較大，是比較不好的結果。PSNR 則跟 SAD 相反，當值愈大時，代表畫質愈好，因此 SAD 與 PSNR 應該是呈現負相關的關係。

比較 SAD 與 PSNR 折線圖可以發現，當我們切的 size 值愈小時，因為可以將圖分成更多的 macroblock，因此預測與計算的準測程度就能提升，SAD 較小、PSNR 較大；至於 range 方面，當 range 愈大時，我們越找到更多資料來還原圖片，因此在 range = 16 但相同 size 下，會得到較低的 SAD 與較高的 PSNR，得到的效果為佳。總和以上的結論，取 size 小與 range 大的切割方法，也就是當 searchRange =16 、 macroblockSize =8 時，會得到最接近 target 的結果。

比較 full search 與 2D logarithmic search，因為 2D logarithmic search 是比較快速的作法，在時間複雜度與運行時間上都會比 full search 好，因此也會犧牲掉一些計算的情況，在某些條件下的效果上，會比 full search 的全盤考慮差一點。

比較 Q2 計算出的 PSNR，其值會比 Q1 在相同 size 以及 range 時較差，這是因為我們的 target 圖與 Q1 的 reference 較為相近，在還原上可以增加準確程度，取相近的地方來進行還原，可靠度會比較高，圖片的失真也會因此減少，因此會有這樣的結果。

時間複雜度：
依據迴圈的運行範圍，就可以定義出兩種 methods 的時間複雜度。
Full search: O (p^2 * N^2)
2D logarithmic search: O (log(p) * N^2)

時間結果：
(1). P = 8, size = 8

Full search: Elapsed time is 2.207450 seconds.

2D logarithmic search: Elapsed time is 0.469946 seconds.

(2). P = 16, size = 8

Full search: Elapsed time is 7.554371 seconds.

2D logarithmic search: Elapsed time is 0.292968 seconds.

從時間複雜度我們可以發現，兩者的時間差異主要是與 p 相關。因為 full search 中複雜度會與 p^2 成正比，而 2D logarithmic search 則與 log(p)成正比，在這樣的差異下，2D logarithmic search 會比 Full search 明顯節省很多時間，搜尋時間可以有效降低。從實際得到的時間結果來看，相同 p、N 下，full search 需要的時間明顯較多，與我們的預測相同。比較 full search 在相同 size 下，p = 8、16 的差別，因為此時 p 是 1:2，平方後兩者時間差應該要為 1:4，由我們得到的時間結果來看，兩者差別也大概是四倍，與理論上差異不大。