**CS3570 Introduction to Multimedia**

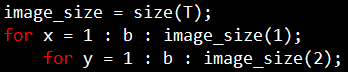
Homework #3 Report

103062234 張克齊

**[實作方法]**

1. 執行檔:

在三個執行檔中，架構是類似的；首先讀入圖片並轉成double的形式，接著利用兩層for迴圈代表每個Target block最左上角x, y值，並傳入到我們要用的Search方法中，並且這邊是直接將整個T和R傳入，在函數中才進行切割，如下圖所示，其中image\_size是一個vector存Target frame的size。

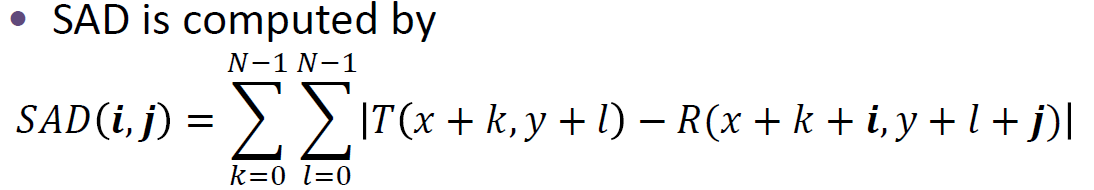


接著就能依據題目的要求，組合出8種可能並回傳每個SAD minimum和結果圖，其中每層for跑出是其中一個block的結果，因此呼叫函數的方法要像是，就能得到最後拼接的結果圖。但題目要求是要顯示他們的difference，並且討論區也有提到要將3個channel相加得到黑白圖，所以會是，這樣就能存我們的結果圖。

最後，求total SAD的方法很簡單，就在每層for累加得到的SAD minimum ，跑完就會是total SAD；並沿用Lab1寫的myPSNR.m來計算結果與Target frame的PSNR值。

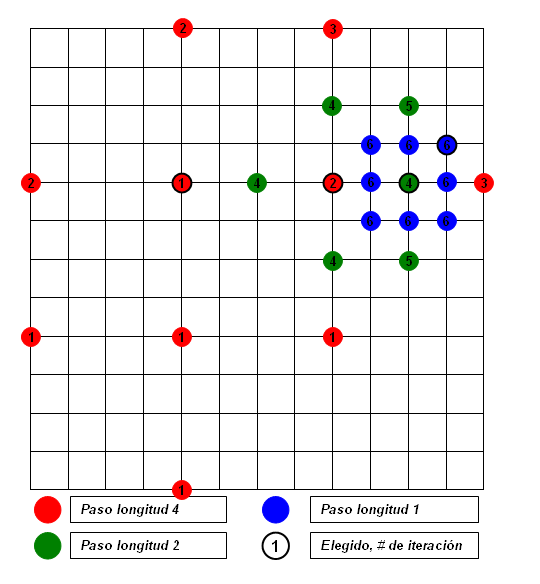
2. myFullSearch.m:

Full Search的方法很暴力，就是在給定的range中，在Reference frame一次動一個pixel為該block的左上角，其中要注意邊界的問題(要圍出完整的block)；並將圍出的block與target的block做SAD的運算，我是依照第四章講義44頁的公式來做(下圖)，code會像是，其中T2, R2為兩個block size \* block size大小的matrices，但因為有3個channel所以最外圈還要多一個sum做相加。最後，利用SAD\_min(一開始設成realmax代表很大的正實數)儲存每一層算出SAD最小值及SAD\_x, SAD\_y對應的(x, y)，所有迴圈跑完時輸出就是以該點為Reference frame的左上角圍出的block，最後的SAD\_min也會是這個block的最小值。



3. my2DLogSearch.m:

主要是參照第四章講義49頁中的pseudo code來實作，可參考下面的附圖來解釋作法。概念為依據一開始給的p取log決定步長，並找到包含自己以及上下左右移動步長共計五個block，分別與Target frame算SAD找出最小值對應的block，若不是中心的block下次就會從該對應block為中心繼續走相同步長找新的block，而若是中心點為SAD最小那會讓步長除2；若步長不等於1會繼續找，等於1時找法會不同，會變成要找包含自己及周圍八個共九個block，並找到SAD最小的就是我們要的最後的block，而回傳的形式會跟full search一樣。



在實作的code中，是利用while迴圈做每一次的找點，而其中有用到一個講義用的方法，N代表現在的步長，就是將每次的移動的方法存成map的型式，而在呼叫時就可以利用像是 代表第k個matrix的第1項，這樣就可以利用一個k的for迴圈簡單的找到對應的五個點並算SAD值，而當步長等於1時，也可以利用兩個從-1 ~ 1的for迴圈找出九個點，其他的部分就是if else的判斷，判斷方法就如同上面的解釋。

[結果圖]

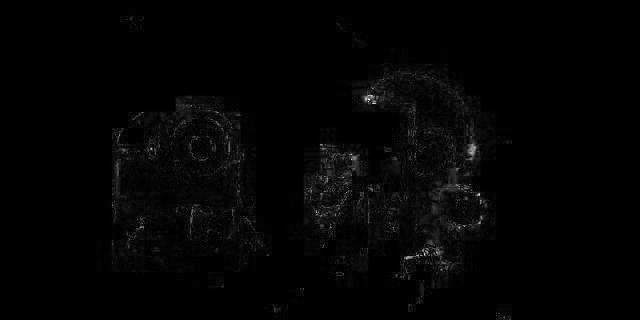
**1. (a)**

macroblock = **8\*8**, p = **8** macroblock size = **8\*8**, p = **16**

Method: **Full Search** Method: **Full Search**

total SAD:  total SAD: 

PSNR:  PSNR: 

macroblock = **8\*8**, p = **8** macroblock size = **8\*8**, p = **16**

Method: **2D Log Search** Method: **2D Log Search**

total SAD:  total SAD: 

PSNR:  PSNR: 

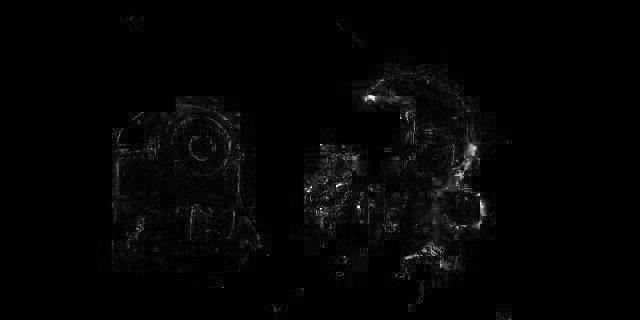
 

macroblock = **16\*16**, p = **8** macroblock size = **16\*16**, p = **16**

Method: **Full Search** Method: **Full Search**

total SAD:  total SAD: 

PSNR:  PSNR: 

macroblock = **16\*16**, p = **8** macroblock size = **16\*16**, p = **16**

Method: **2D Log Search** Method: **2D Log Search**

total SAD:  total SAD: 

PSNR:  PSNR: 

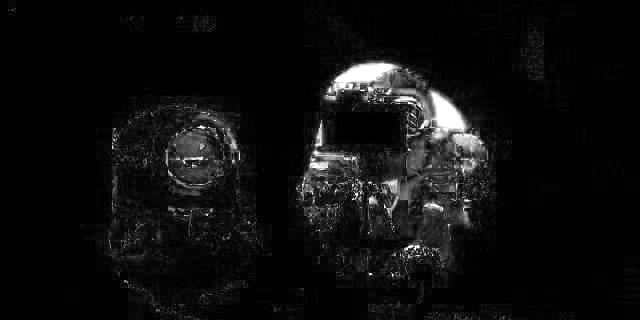
**1. (b)**

macroblock = **8\*8**, p = **8** macroblock size = **8\*8**, p = **16**

Method: **Full Search** Method: **Full Search**

total SAD:  total SAD: 

PSNR:  PSNR: 

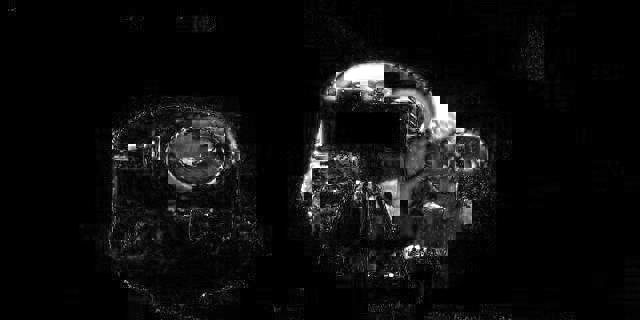
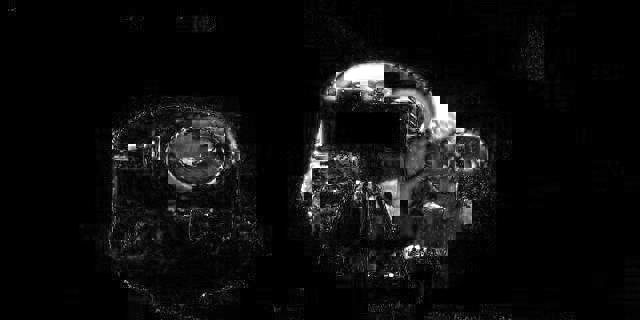
 

macroblock = **8\*8**, p = **8** macroblock size = **8\*8**, p = **16**

Method: **2D Log Search** Method: **2D Log Search**

total SAD:  total SAD: 

PSNR:  PSNR: 

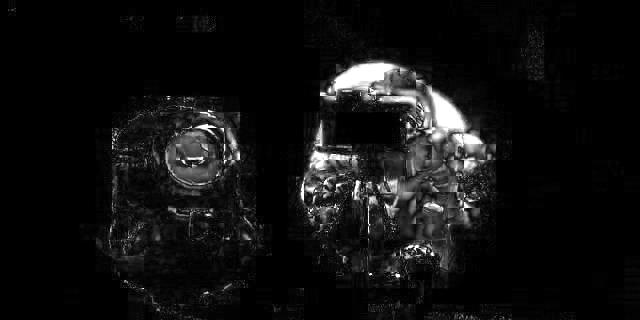
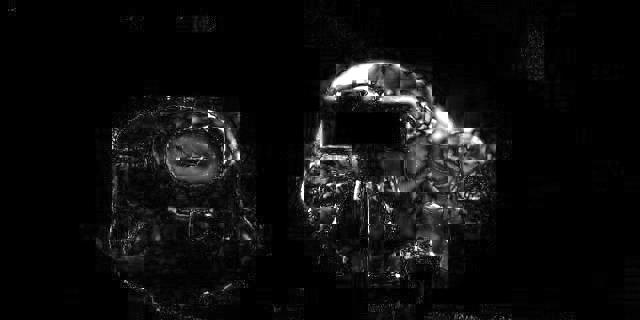
 

macroblock = **16\*16**, p = **8** macroblock size = **16\*16**, p = **16**

Method: **Full Search** Method: **Full Search**

total SAD:  total SAD: 

PSNR:  PSNR: 

macroblock = **16\*16**, p = **8** macroblock size = **16\*16**, p = **16**

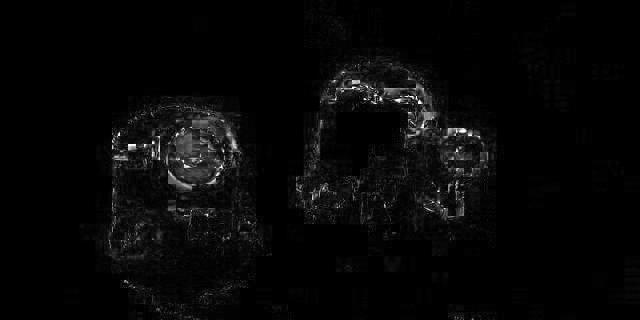
Method: **2D Log Search** Method: **2D Log Search**

total SAD:  total SAD: 

PSNR:  PSNR: 

**2.**



macroblock = **8\*8**, p = **8**

Method: **2D Log Search**

total SAD: 

PSNR: 

[時間複雜度分析]

a. 兩種Search方法的時間分析:

(i) Full Search:

因為作法是取出每個block並掃過(2p+1)^2的pixel range，而其中block數量可以轉化成T的image\_size來表示，也就是(T\_row/block\_size) \* (T\_column/block\_size)。

**Total = (T\_row/block\_size) \* (T\_column/block\_size) \* (2p+1)^2**

(ii) 2D Log Search:

想法是從取點下手，若取到的不是中心點，而是像取到上面，那我們可以知道下面有一半不會再取到了，而若是取到中心點也會因為range減半也少掉至少一半，因此會是log2((2p+1)^2)，其中log2((2p+1)^2)代表每次range的範圍會減半

**Total = (T\_row/block\_size) \* (T\_column/block\_size) \* log2((2p+1)^2)**

b. 不同p(range)時的執行時間差別:

我是運用tic; toc;得到程式執行時間，寫在timeCompare.m中。

(i) p = 8: Full -> 

Log -> 

(ii) p = 16: Full -> 

Log -> 

c. 比較與討論:

從b的結果我們可以發現2D Log Search時間明顯比Full Search短很多，這跟我們在a中分析的結果一樣，因為一個有取log，所以整個搜尋時間可以非常有效的降低。

[比較與討論]

1. 比較1(a)(b) total SAD:

Total SAD的意義代表Target frame和Reference frame差異的總和，愈高代表兩者差距愈大，從直接打開input圖片可以發現1(b)的Target frame與Reference frame差距較大，因此1(b)得到的total SAD值就會比1(a)還要高，實作結果也有得到預估的效果，1(a)約為3000多而1(b)為10000多。

2. 討論PSNR和SAD的關係:

我們這邊算的PSNR意義為Target frame和拼接對應Reference結果的失真程度，值愈大代表失真愈少，SAD則是兩者的差異程度；因此，SAD愈大時代表差異變大，圖片失真應該會增加，因此我們得到的PSNR值應該會變小，反之亦然。

3. 第二小題使用bi-directional方法的差別：

這邊我們是利用分別比較兩個Reference frame(丟兩次2D Log Search)並取兩者中最小SAD的對應block當作結果，方法如下圖所示，其中b是block size。透過兩個Reference frame結果跟1(b)相比可以發現其中total SAD變小，這是必然的結果因為我們是取兩者中較小的那個；而PSNR的部分則是變大了，因為我們參考的frame增加了，可以想像的是圖片的失真也會因此減少，所以最後的PSNR值會上升。

