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。

```
image_size = size(T);
for x = 1 : b : image_size(1);
    for y = 1 : b : image_size(2);
```

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 `[SAD, full_a(x:x+b1-1, y:y+b1-1, :)] = myFullSearch(T, R, b1, p1, x, y);`，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 `sum(abs(full_a-T), 3)`，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 `total_full_a = total_full_a + SAD;`，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 `psnr = myPSNR(T, full_a);`。

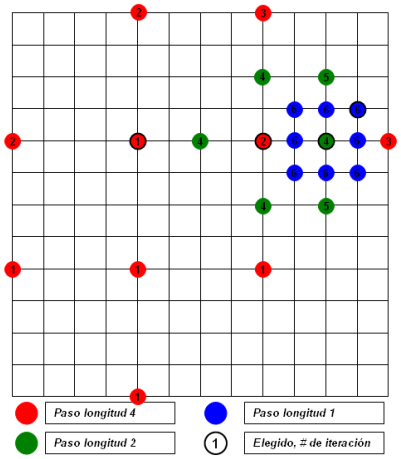
2. myFullSearch.m:

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 `SAD_now = sum(sum(sum(abs(T2-R2))));`，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。

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

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 `M = {[0 0], [N 0], [0 N], [-N 0], [0 -N]};`，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 `M{k}(1)` 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]

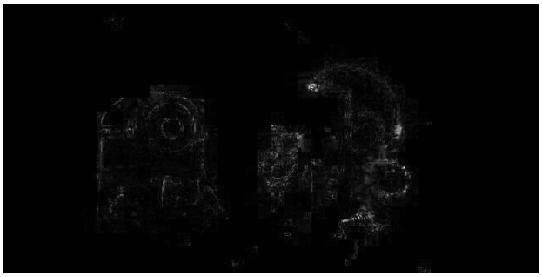
1. (a)

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

Method: **Full Search**

total SAD: `3.0279e+03`

PSNR: `36.3029`

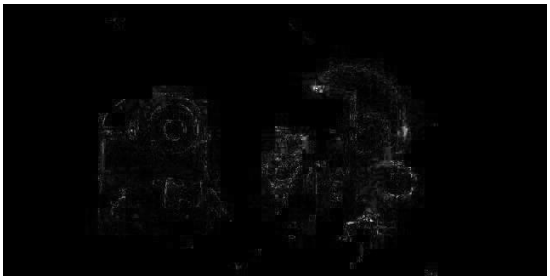


macroblock size = **8*8**, p = **16**

Method: **Full Search**

total SAD: `2.9728e+03`

PSNR: `36.5127`

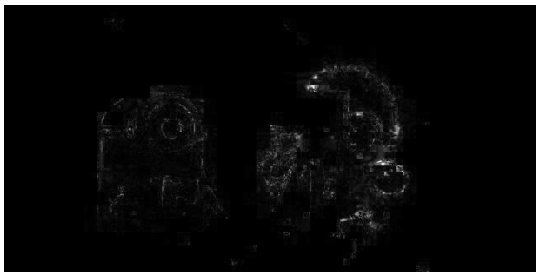


macroblock = **8*8**, p = **8**

Method: **2D Log Search**

total SAD: `3.3017e+03`

PSNR: `35.1268`

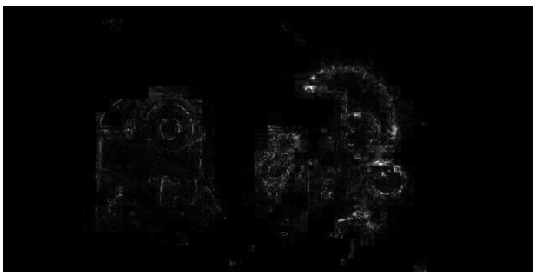


macroblock size = **8*8**, p = **16**

Method: **2D Log Search**

total SAD: `3.3017e+03`

PSNR: `35.1268`

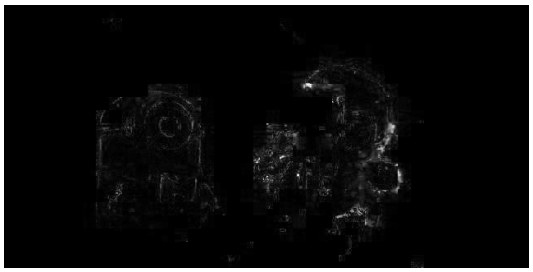


macroblock = **16*16**, p = **8**

Method: **Full Search**

total SAD: `3.5170e+03`

PSNR: `34.5329`

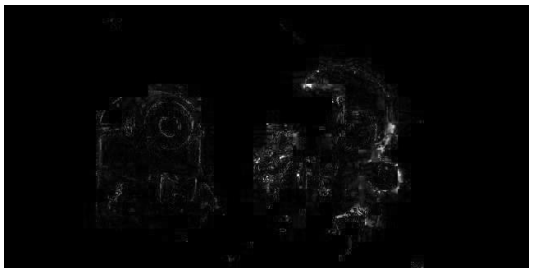


macroblock size = **16*16**, p = **16**

Method: **Full Search**

total SAD: `3.4977e+03`

PSNR: `34.5697`



macroblock = **16*16**, p = **8**

Method: **2D Log Search**

total SAD: `3.6547e+03`

PSNR: `34.1279`

macroblock size = **16*16**, p = **16**

Method: **2D Log Search**

total SAD: `3.6547e+03`

PSNR: `34.1279`

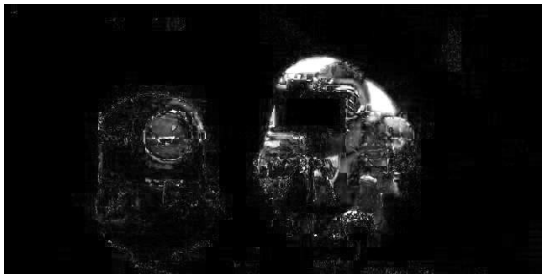
1. (b)

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

Method: **Full Search**

total SAD: 1.1694e+04

PSNR: 24.5113



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

Method: **Full Search**

total SAD: 9.0419e+03

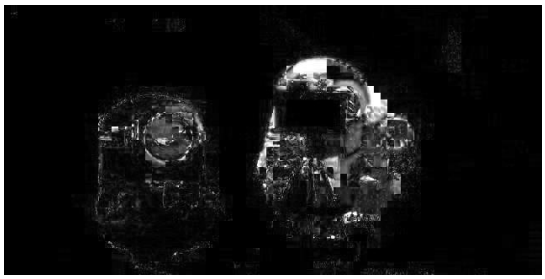
PSNR: 27.3147

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

Method: **2D Log Search**

total SAD: 1.1503e+04

PSNR: 24.8130

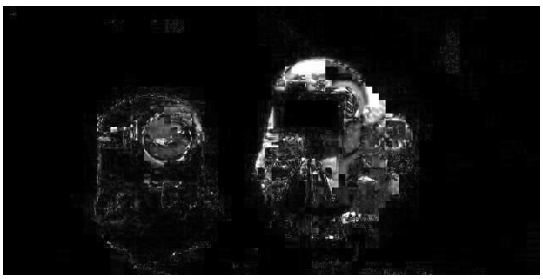


macroblock size = **8*8**, p = **16**

Method: **2D Log Search**

total SAD: 1.1503e+04

PSNR: 24.8130

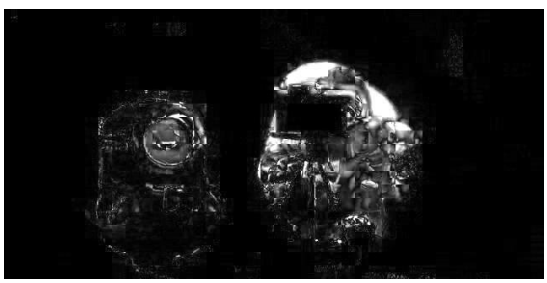


macroblock = **16*16**, p = **8**

Method: **Full Search**

total SAD: 1.3904e+04

PSNR: 23.2883

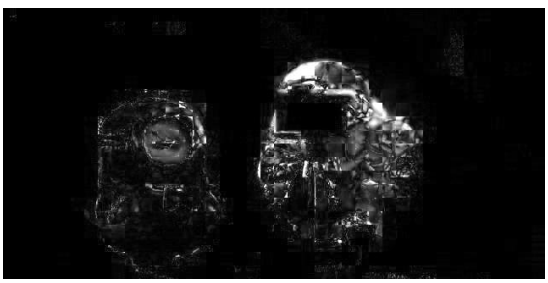


macroblock size = **16*16**, p = **16**

Method: **Full Search**

total SAD: 1.1608e+04

PSNR: 25.2046

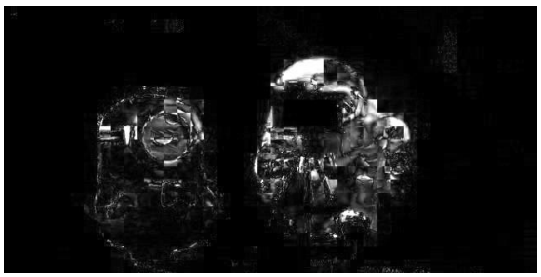


macroblock = **16*16**, p = **8**

Method: **2D Log Search**

total SAD: `1.2926e+04`

PSNR: `24.0388`

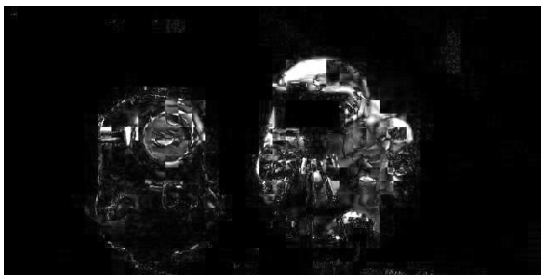


macroblock size = **16*16**, p = **16**

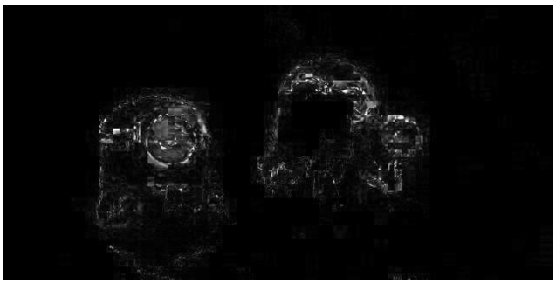
Method: **2D Log Search**

total SAD: `1.2926e+04`

PSNR: `24.0388`



2.



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

Method: **2D Log Search**

total SAD: `5.7284e+03`

PSNR: `30.4543`

[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 -> `Elapsed time is 3.364738 seconds.`

Log -> `Elapsed time is 0.282508 seconds.`

(ii) p = 16: Full -> `Elapsed time is 12.130325 seconds.`

Log -> `Elapsed time is 0.255403 seconds.`

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。

```
[SAD, log_R1(x:x+b-1, y:y+b-1, :)] = my2DLogSearch(T, R1, b, p, x, y);
[SAD2, log_R2(x:x+b-1, y:y+b-1, :)] = my2DLogSearch(T, R2, b, p, x, y);
if SAD < SAD2
    result(x:x+b-1, y:y+b-1, :) = log_R1(x:x+b-1, y:y+b-1, :);
    total = total + SAD;
else
    result(x:x+b-1, y:y+b-1, :) = log_R2(x:x+b-1, y:y+b-1, :);
    total = total + SAD2;
end
```