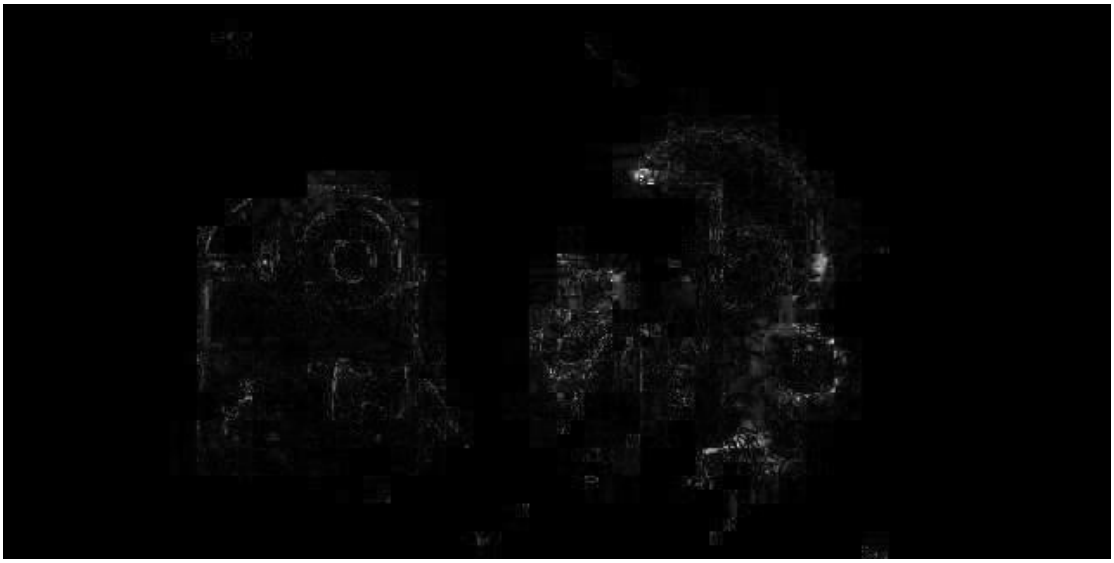
Introduction to Multimedia Homework #3

103062318 蔡尚倫

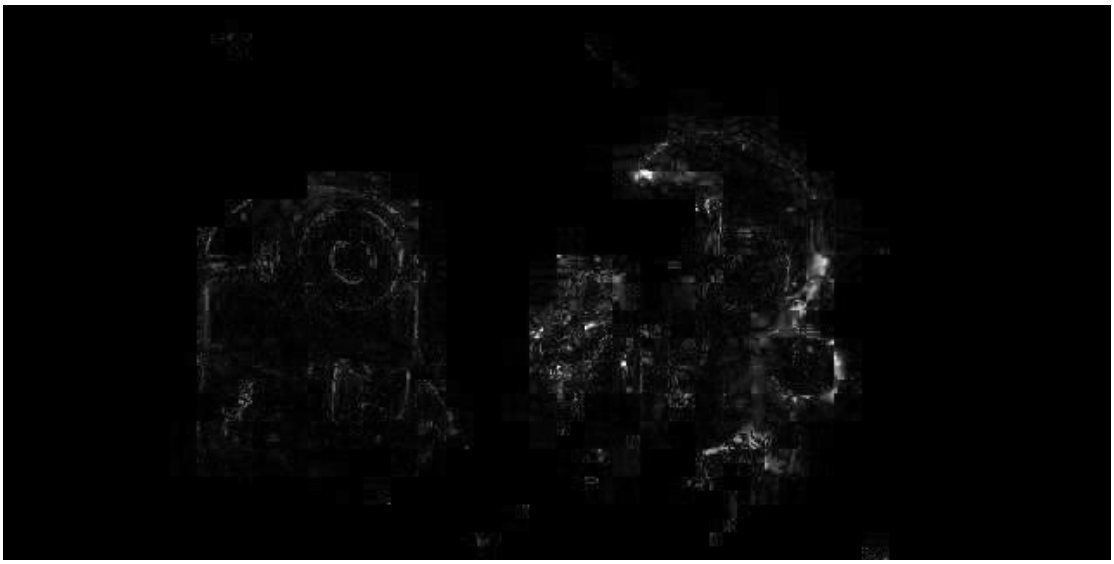
Q1.(2D logarithmic search method 簡寫成 2D--SEARCH)

a. Show the residual images when the target image is frame05073.jpg for all the above combinations. (8 images)文字下方代表該圖片 B=Block Size p=Search Range

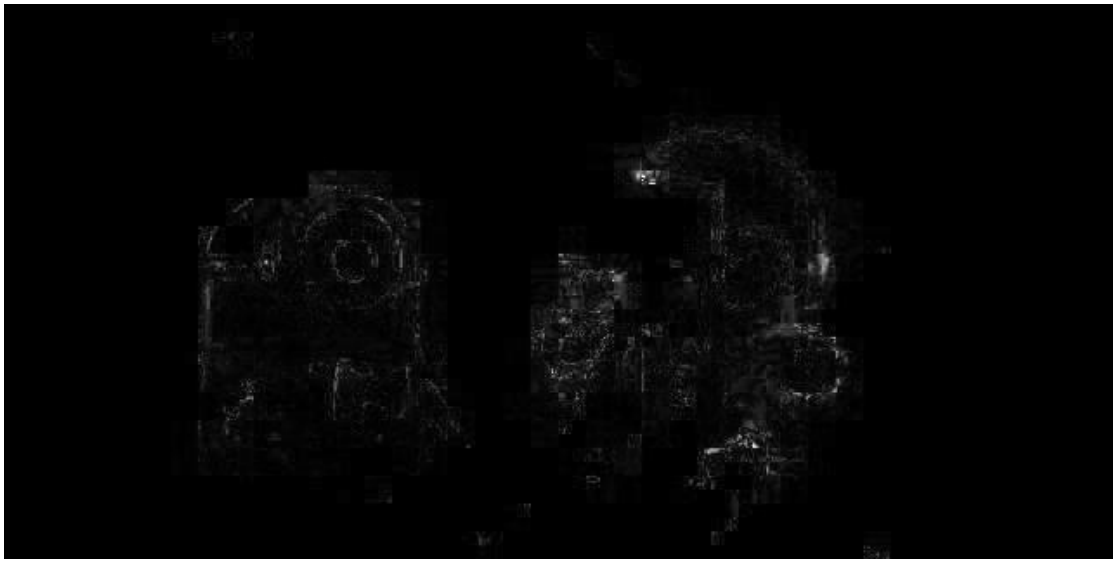
FULL SEARCH B=8 p=8



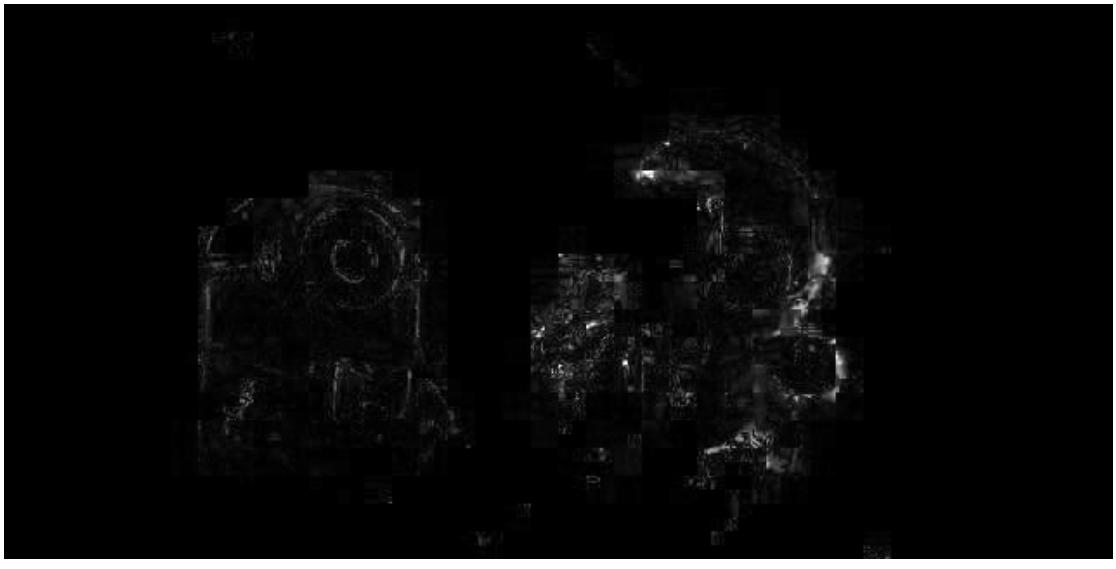
FULL SEARCH B=16 p=8



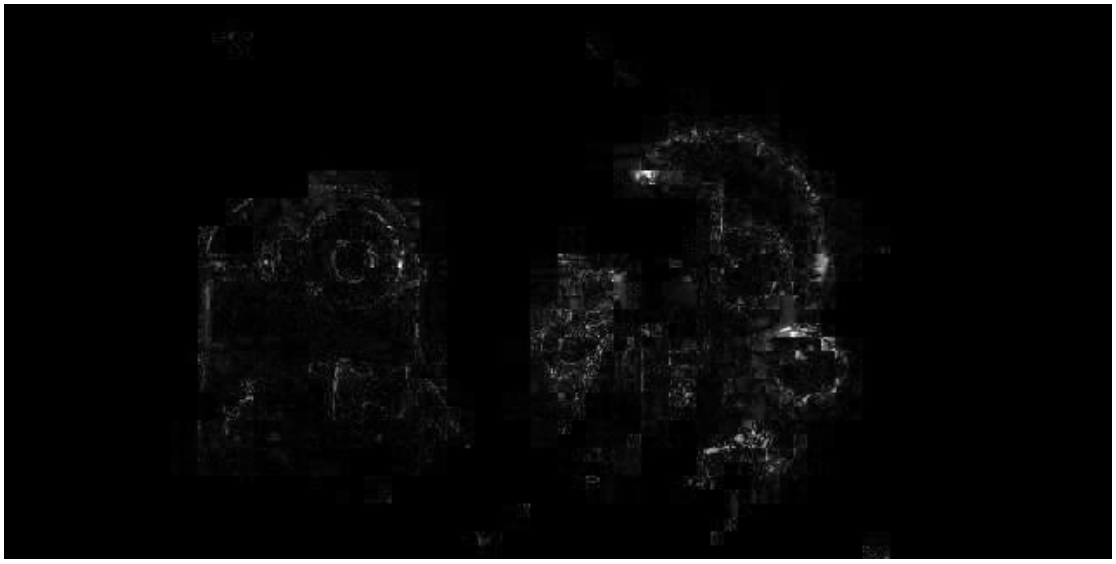
FULL SEARCH B=8 p=16



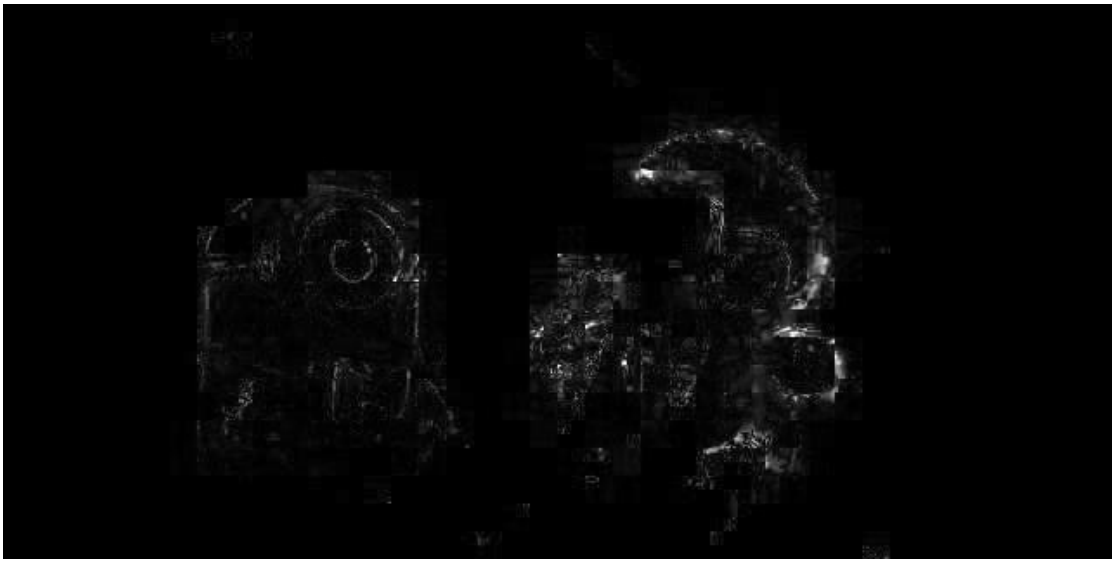
FULL SEARCH B=16 p=16



2D--SEARCH B=8 p=8

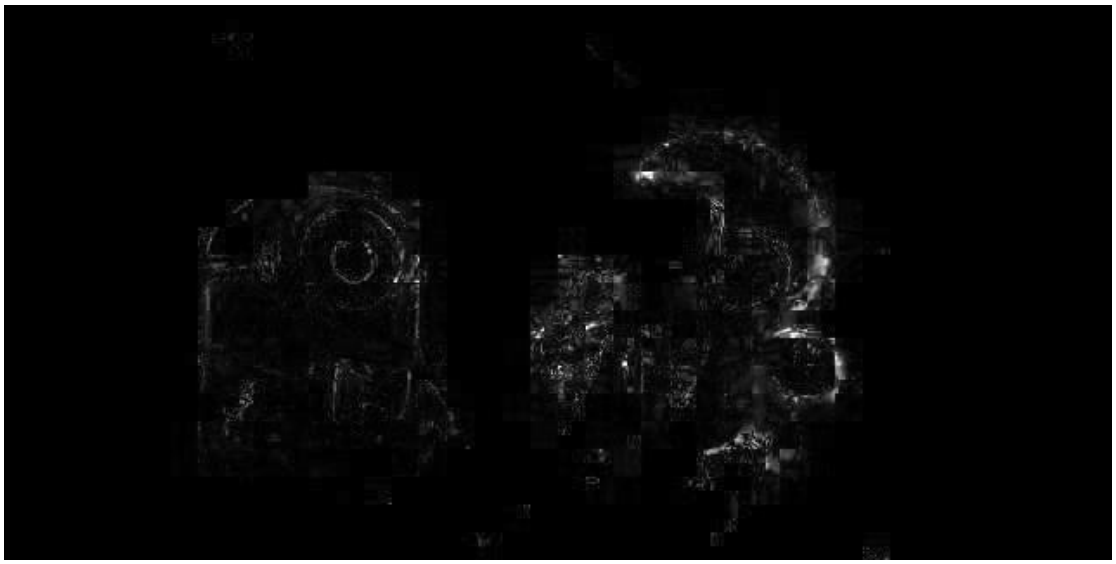


2D--SEARCH B=16 p=8



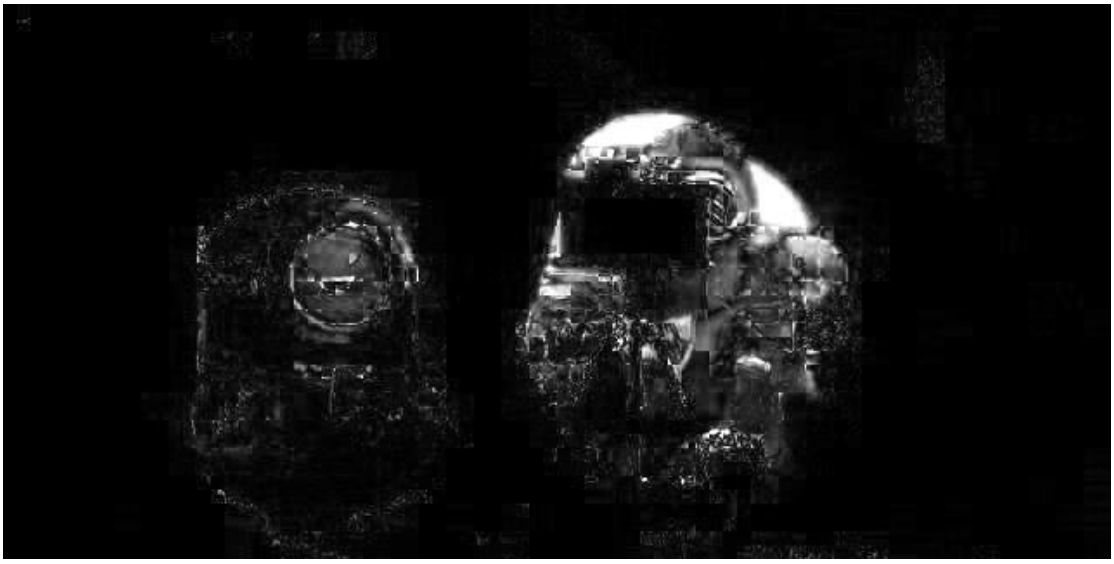
2D--SEARCH B=8 p=16

2D--SEARCH B=16 p=16

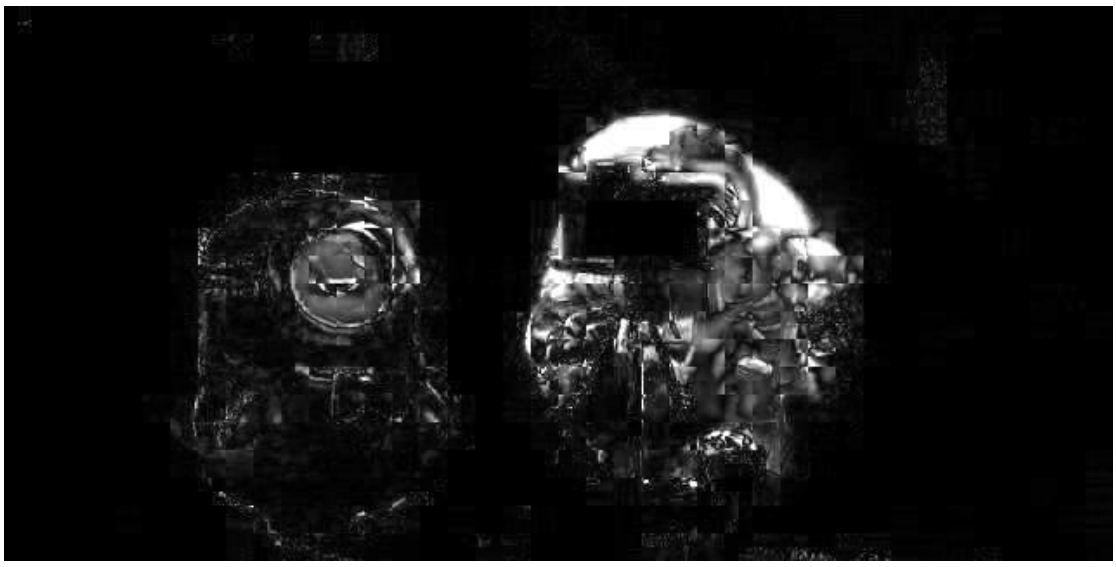


b. Show the residual images when the target image is frame05081.jpg for all the above combinations. (8 images) 文字下方代表該圖片 B=Block Size p=Search Range

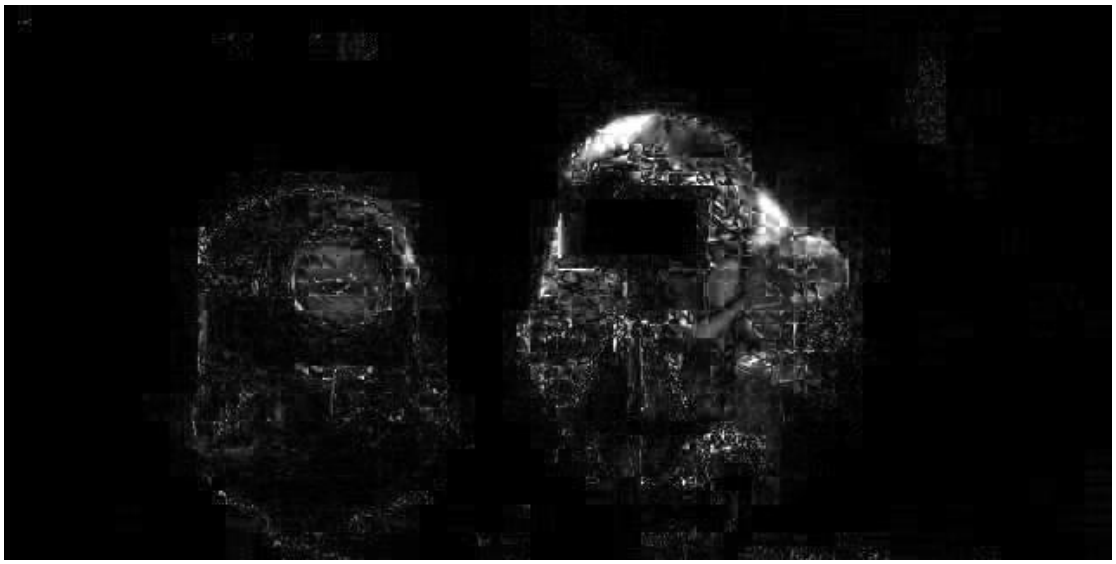
FULL SEARCH B=8 p=8



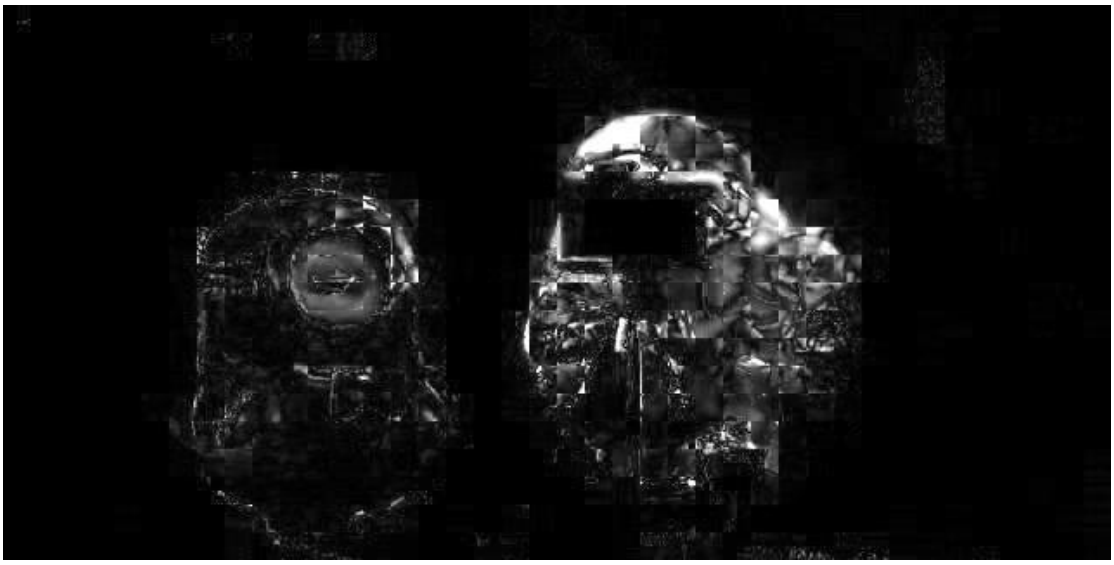
FULL SEARCH B=16 p=8



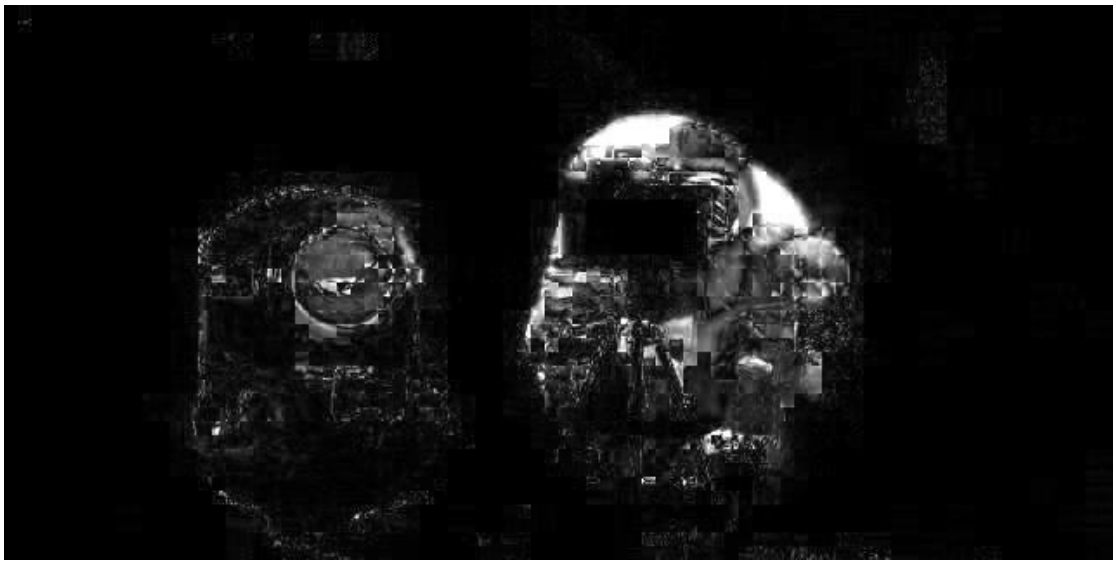
FULL SEARCH B=8 p=16



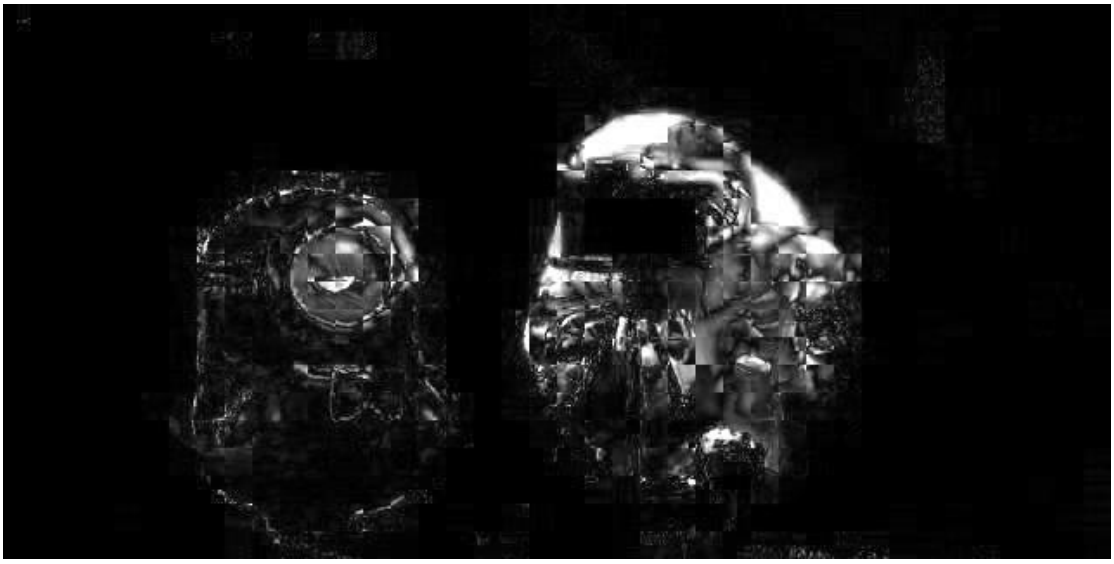
FULL SEARCH B=16 p=16



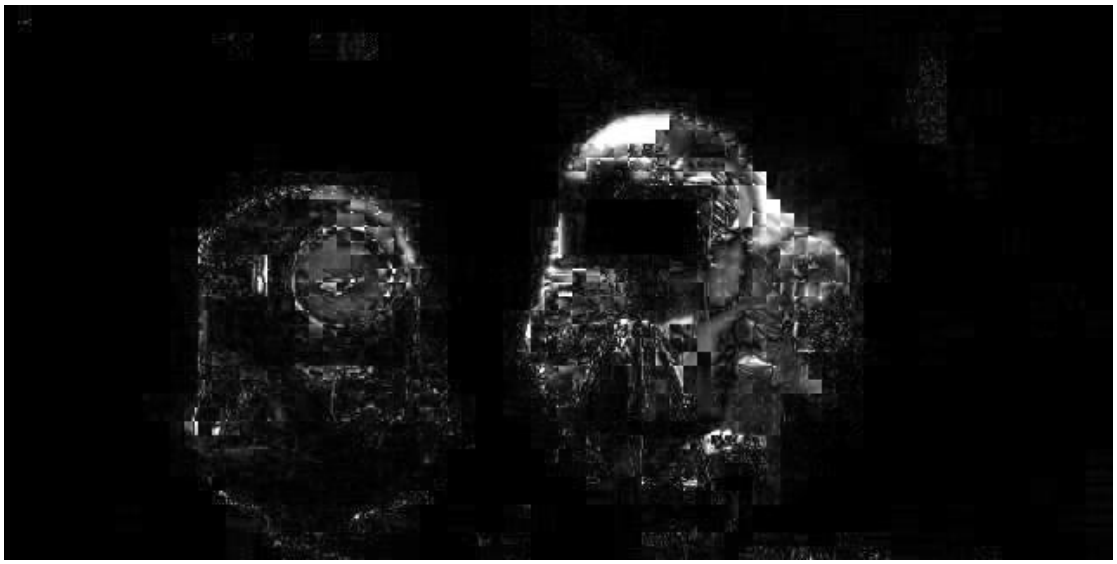
2D--SEARCH B=8 p=8



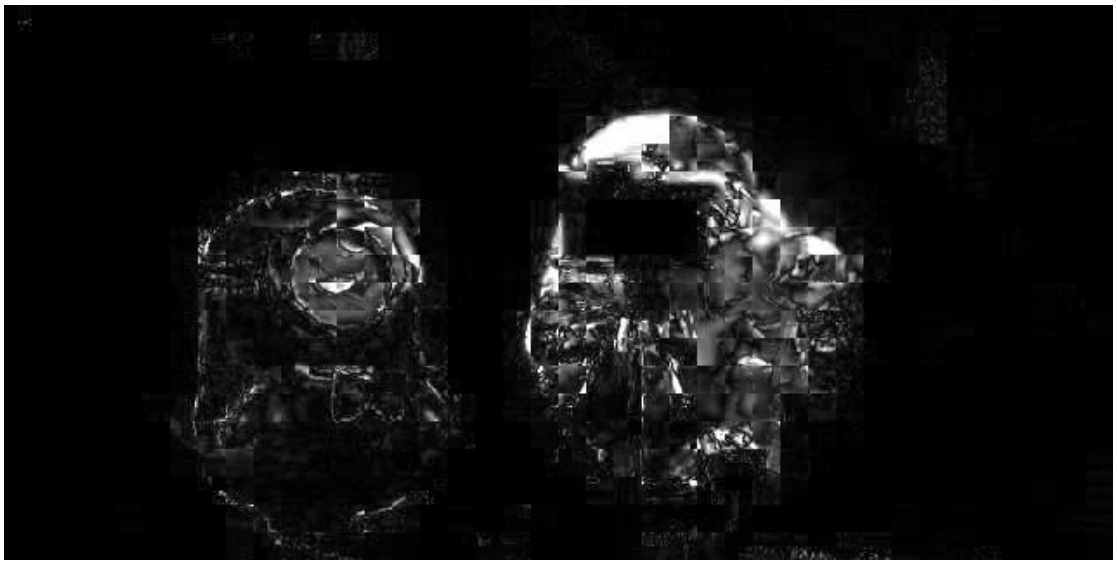
2D--SEARCH B=16 p=8



2D--SEARCH B=8 p=16



2D--SEARCH B=16 p=16



c. Show and compare the total SAD values for all the results in (a) and (b).

B=Block Size p=Search Range

Show

(a)

The reference image is frame05072.jpg

The target image is frame05073.jpg

FULL SEARCH B=8 p=8	TOTAL SAD:3027.929412
FULL SEARCH B=16 p=8	TOTAL SAD:3516.972549
FULL SEARCH B=8 p=16	TOTAL SAD:2972.760784
FULL SEARCH B=16 p=16	TOTAL SAD:3497.654902
2D--SEARCH B=8 p=8	TOTAL SAD:3282.972549
2D--SEARCH B=16 p=8	TOTAL SAD:3674.929412
2D--SEARCH B=8 p=16	TOTAL SAD:3306.611765
2D--SEARCH B=16 p=16	TOTAL SAD:3678.400000

(b)

The reference image is frame05072.jpg

The target image is frame05081.jpg

FULL SEARCH B=8 p=8	TOTAL SAD:11693.690196
FULL SEARCH B=16 p=8	TOTAL SAD:13903.788235
FULL SEARCH B=8 p=16	TOTAL SAD:9041.898039
FULL SEARCH B=16 p=16	TOTAL SAD:11607.898039
2D--SEARCH B=8 p=8	TOTAL SAD:12128.490196
2D--SEARCH B=16 p=8	TOTAL SAD:13824.560784
2D--SEARCH B=8 p=16	TOTAL SAD:10623.835294
2D--SEARCH B=16 p=16	TOTAL SAD:12483.407843

Compare

同演算法不同 Block size 與不同 Search range，由小到大排序 SAD，大致是 B=8 p=16 < B=8 p=8 < B=16 p=16 < B=16 p=8

原因應該是：

Block size 如果取 16 的話，切割整個圖片後，如果是從一堆非常不像的 Block 中選 SAD 最小的，由於一次性選擇一個 Block 的關係，造成填補 16*16 個都很不像的 pixel，所以會與原本 target image 差很多，所以 B=8 填補效果會好

於 B=16。

Search range 則是很直覺的，先考慮圖片前後的相關性，某個 Block 在 frame_i 移動到 frame_i+1(這裡先不考慮切換場景的情況，因為測試圖片沒有這種情況)，如果你搜尋範圍比較大，那麼你就有越大的機會找到該 Block 在下一個 frame 的位置，所以 p=16 填補效果會好於 p=8。

不同的演算法，由小到大排序 SAD，大致是 FULL SEARCH < 2D—SEARCH
原因應該是：

2D—SEARCH 是為了加速尋找速度的演算法，所以無法像 FULL SEARCH 一樣，將範圍內的部分搜尋一遍，會有找不夠好的情況發生，就是你固定選取的參考 Block 不是最佳填補 Block，所以填補效果 FULL SEARCH 會好於 2D—SEARCH。

不同的 target image，有小到大排序 SAD，大致是 frame05073<frame05081
原因應該是：

如果 reference image 與 target image 相差太多個 frame，一是你的 Search range 不太可能找到相似的 Block(Object 可能已經移動很多，像是測試圖片中右邊小小兵手拿的相機)，除非你 Search 整張圖片才有機會，二是可能經過這麼多個 frame 可能早就已經切換場景了，就不太可能填補出圖片，所以填補效果 frame05073 會好於 frame05081。

d. Show the PSNR of all the results in (a) and (b), and discuss the relation with the SAD. B=Block Size p=Search Range

Show

(a)
The reference image is frame05072.jpg
The target image is frame05073.jpg

FULL SEARCH B=8 p=8	PSNR:36.303438
FULL SEARCH B=16 p=8	PSNR:34.532875
FULL SEARCH B=8 p=16	PSNR:36.513541
FULL SEARCH B=16 p=16	PSNR:34.569724
2D--SEARCH B=8 p=8	PSNR:35.258490
2D--SEARCH B=16 p=8	PSNR:34.051036
2D--SEARCH B=8 p=16	PSNR:35.201534
2D--SEARCH B=16 p=16	PSNR:34.041279

(b)

The reference image is frame05072.jpg

The target image is frame05081.jpg

FULL SEARCH B=8 p=8	PSNR:24.511646
FULL SEARCH B=16 p=8	PSNR:23.288301
FULL SEARCH B=8 p=16	PSNR:27.315794
FULL SEARCH B=16 p=16	PSNR:25.205202
2D--SEARCH B=8 p=8	PSNR:24.324987
2D--SEARCH B=16 p=8	PSNR:23.461490
2D--SEARCH B=8 p=16	PSNR:25.737785
2D--SEARCH B=16 p=16	PSNR:24.476057

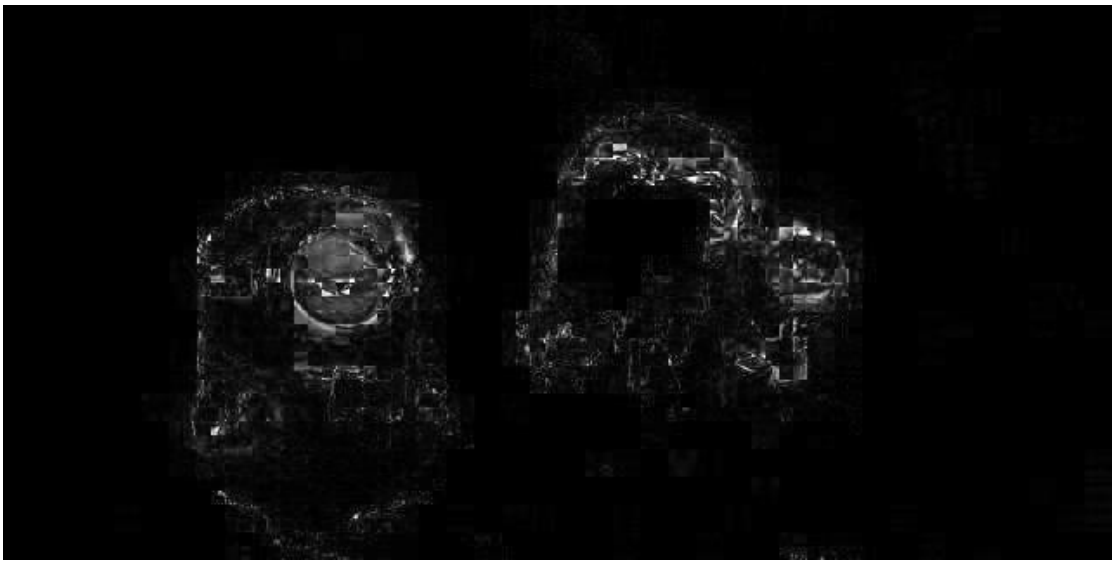
Diucuss

PSNR 的值越大，代表兩張圖片的相似度越高，而 SAD 的值越大，則代表兩張圖片的相似度越低，所以 PSNR 與 SAD 是負相關的兩種數值，代表的意義都是在描述兩張圖片之間的相似程度，只是數值大小代表的意義截然不同。

Q2. (2D logarithmic search method 簡寫成 2D--SEARCH)

a. Show the residual images and total SAD value when the target image is frame05081.jpg in search range p=8, 8x8 macroblock size and 2D logarithmic search method. (1 image) 文字下方代表該圖片 B=Block Size p=Search Range

Bi2D--SEARCH B=8 p=8 TOTAL SAD:5755.235294 PSNR:30.565988



b. Discuss the results from (a) with the problem 1 of the same settings (p=8, 8x8 macroblock size, 2D logarithmic search method).

Discuss

Result from(a)

Bi2D--SEARCH B=8 p=8 TOTAL SAD:5755.235294 PSNR:30.565988

Problem 1 of the same settings

2D--SEARCH B=8 p=8 TOTAL SAD:12128.490196 PSNR:24.324987

由數據很明顯看出，無論是在 SAD 與 PSNR 的表現上，bi-directional prediction 較優，最基本的原因可能是，能參考的填補 Block 更多，那麼尋找到最佳填補 Block 只會越好，最差也只會與單方向的 Problem 1 of the same settings 一樣(如果全部取用 frame72 的話)，那麼再從 frame 的變化來分析，在 frame72 的中找不到的最佳填補 Block，可能會在 frame85 中找到最佳填補 Block，我拿這次的測試圖片來講解，左邊的小小兵主要取到的 Block 來自 frame72，而如果只用 frame72 預測右邊小小兵舉起相機的動作的話，就沒辦法比 frame85 右邊已經拿起相機的小小兵來的好，所以使用這種 bi-directional 能從前後尋找到較好的填補 Block。

Q3. (2D logarithmic search method 簡寫成 2D--SEARCH)

a. Analyze the theoretical time complexity for the two search algorithms.

rows： 圖片行的數目/Block Size

cols： 圖片列的數目/Block Size

range： 搜尋範圍

Time complexity of FULL SEARCH

rows*cols*(2*range+1)^2 = O(rows*cols*ranges^2)

Time complexity of 2D—SEARCH

rows*cols*(constant $*$ $\log_2(2 * \text{range} + 1)$) = O(rows*cols*($\log_2$ range))

分析按照實際的 CODE，觀察最主要的迴圈，其餘的部分可以當作時間複雜度的常數，從 FULL SEARCH 開始，觀察最大的主迴圈有四層，所以就能計算出 FULL SEARCH 的時間複雜度，而 2D—SEARCH 則是改善 range^2 的時間複雜度，改善方式為，移動半徑為 p/2，上下左右加自己五個點去比較 SAD，最多總共會

找((2p)/(p/2)+1)^2 個點，會有 log2(range)層，因為每次選中間那個點就會以二分之一縮小範圍，所以可以加提升 range^2/log2(range)效能

b. Measure the execution time required for the two search algorithms with the two different search range sizes (p=8 and p=16).以 Q1.a 測試 B=Block Size p=Search Range

FULL SEARCH B=8 p=8	Elapsed time is 6.790916 seconds.
FULL SEARCH B=8 p=16	Elapsed time is 24.874815 seconds.
2D--SEARCH B=8 p=8	Elapsed time is 0.583278 seconds.
2D--SEARCH B=8 p=16	Elapsed time is 0.720939 seconds.

c. Compare and discuss the execution time with the theoretical time complexity. B=Block Size p=Search Range

在相同的 B、p、range=8 條件下，由數據明顯的看出 2D—SEARCH 跑得比 FULL SEARCH 快。相同演算法下，Search range 較大的需要花較久的時間。由剛剛理論的效能提升為 range^2/log2(range)，計算大概為 8^2/3 = 21.33 倍，而由時間觀察到的效能提升為 6.790916/0.583278，計算大概為 11.64 倍，我想理論比較大，是因為沒考慮到下方 log2(range)中的常數項，因為我們並沒有把 Constant 很仔細地計算出來，所以常常理論會高於現實在這裡看來相當直覺。FULL SEARCH 無論如何勢必會跑過四次 Loop，2D—SEARCH 則可能更快找到相似的點(如果每次都剛好中心點的 SAD 最小，就能將 range/2)，由此也可由程式角度推斷 2D—SEARCH 會比 FULL SEARCH 快很多。

Q4. (2D logarithmic search method 簡寫成 2D--SEARCH)

how you implement the methods

FULL SEARCH：(fullsearch.m) 起始點用左上角的座標代表整個 Block

讀進 target image 與 reference image(再傳入前已經轉成 double 型態)，取得圖片(任一皆可)的 rows cols heights，開始跑四層 for-loop 去尋找填補的 Block 起始點(Block 左上角的座標)。前面兩層 loop 是 i=0:block sizes:rows-blocks sizes 和 j=0:block sizes:rows-blocks，這裡從 i=0 和 j=0 開始的原因是方便取得 Block，只需要(i+1,i+block sizes,:)，式子就不會有減有加的，另外每 sizes 個數跑迴圈，就能準確到達想要填補的 Block 起始點，最後到 rows-blocks sizes，如果到 rows 的話會多做一次。

接著將上下左右的邊界找好，我這裡就開始判斷有沒有超過原本圖片的邊界，有的話就做修正，假如小於 0，就修正為 0，假如大於 rows 或 cols，就修

成為 rows-block sizes 或 cols-block sizes(用上方(i+1,i+block sizes,:)這種方式取的時候，就保證部會超過 rows 或 cols)。

最後跑兩層 loop 跑上下 range，將 range 中所有的 Block 個別算出 SAD，記住最小的 SAD，那麼這個 Block 就是這 Block 最佳的填補 Block。

算完後得到 result_img，與原本的 target_img 相減以及取絕對值，並將三維度的值對應相加，得到 residual_img，這裡我將 result_img 與 residual_img 都傳出 function。

2D—SEARCH：(Mysearch2D.m)　　起始點用左上角的座標代表整個 Block

前面兩層 loop 與 FULL SEARCH 相同就不贅述。

接著開始跑一個 while loop，直到搜尋半徑砍半成 1 為止，找的方式跟助教一樣，找自己一個 Block 起始點與搜尋半徑上下左右四個 Block 的起始點，這裡當然也要判斷邊界，亦即不要超過原本圖片的邊界與 Serach range 的邊界，將五個 Block 的 SAD 算完後比較最小的，如果 SAD 最小的還是在自己這個 Block 的起始點的話，搜尋半徑就砍半，直到砍到剩下 1 就跳出 loop

最後在該 Block 最近的 8 個 Block 比較，擁有最小 SAD 者，就是這個 Block 就是這 Block 最佳的填補 Block。

算完後得到 result_img，與原本的 target_img 相減以及取絕對值，並將三維度的值對應相加，得到 residual_img，這裡我將 result_img 與 residual_img 都傳出 function。