

多媒體技術概論 AS3

105060016 謝承儒

Q1.

實作 Function

1. [predictIMG, MV_row, MV_col, SAD] = PredictImage(referenceIMG, targetIMG, searchRange, blockSize, searchMethod)

如果 searchMethod == FullSearch :

- (1) 將 targetIMG 依照 blockSize 的大小，把整張圖片的 block 切割出來
- (2) 以 target block 為中心，其座標為(x,y)，向四周延伸，從座標(x- searchRange, y- searchRange) 到(x+ searchRange, y+searchRange)一個個得到 reference block，共有 searchRange^2 個
- (3) 將每個 reference block 都和 target block 做比較
=> Compare(referenceBlock, targetBlock)，便會得到一個 dissimilarity
- (4) 若新得到的 dissimilarity 比先前的還小，就更新 tmp_d、tmpBlock、tmp_row、tmp_col，等到每個 reference block 都做完後，tmp_d、tmpBlock、tmp_row、tmp_col 即是 predict block(和 target block 最相近的)
- (5) 將 SAD 往上加 tmp_d、tmpBlock 放入在 predictIMG 相對應的位置、新增 MV_row、MV_col 的 element

$$\text{MV}_{\text{row}(i,j)} = \text{tmp}_{\text{row}} - \text{targetBlock}_{\text{row}}$$

$$\text{MV}_{\text{col}(i,j)} = \text{tmp}_{\text{col}} - \text{targetBlock}_{\text{col}}$$

更新完後，即找下一個 target block 重複步驟(2)~(5)

- (6) 全部的 target block 做完後，即可得到 predictIMG，而 MV_row、MV_col 則用來繪製之後的箭頭圖

如果 searchMethod == threeStepSearch :

- (1) 將 targetIMG 依照 blockSize 的大小，把整張圖片的 block 切割出來
- (2) 計算需要做幾步

$$\text{stepNum} = \log_2 \text{searchRange}$$

- (3) 計算每一步的 Range，i 為做到第幾步

$$\text{stepRange} = 2^{\text{stepNum}-i}, i = 1, 2, \dots, \text{stepNum}$$

- (4) 選擇 target block，mid block 初始為 target block
- (5) 以 mid block 為中心，其座標為(x,y)，向四周延伸，得到 9 個 reference block，其座標分為(x- stepRange, y- stepRange)、(x, y- stepRange)、(x+ stepRange, y- stepRange)、(x- stepRange, y)、(x, y)、(x+ stepRange, y)、(x- stepRange, y+ stepRange)、(x, y+ stepRange)、(x+ stepRange, y+ stepRange)
- (6) 將每個 reference block 都和 target block 做比較
=> Compare(referenceBlock, targetBlock)，便會得到一個 dissimilarity
- (7) 若新得到的 dissimilarity 比先前的還小，就更新 tmp_d、tmpBlock、tmp_row、tmp_col，等到每個 reference block 都做完後，tmp_d、tmpBlock、tmp_row、tmp_col 即是 mid block
- (8) 以新的 mid block 重複做步驟(3)~(7)，記得更新 stepRange，直到做滿 stepNum。即可得到 predict block(和 target block 最相近的)，也將 SAD 往上加 tmp_d、新增 MV_row、MV_col 的 element

- (9) 選擇下一個 target block，重複做步驟(4)~(8)
- (10) 做完所有 target block 即可得到 predictIMG，而 MV_row、MV_col 則用來繪製之後的箭頭圖
2. [dissimilarity] = Compare(referenceBlock, targetBlock)
- (1) 將 referenceBlock 和 targetBlock 做相減
 - (2) 對相減後的所有 element 取絕對值
 - (3) 接著把全部 element 相加，即是 dissimilarity
- 註. 使用 double 型別做運算
3. [residualIMG] = ResidualImage(predictIMG, targetIMG)
- (1) 分成 R、G、B 做
 - (2) 以 R 為例，將 predictIMG 的 R 和 targetIMG 的 R 相減
 - (3) 對相減後的所有 element 取絕對值，即可得到 residualIMG_R
 - (4) G、B 也做步驟(2)、(3)，即可得到 residualIMG_G、residualIMG_B
 - (5) 把 residualIMG_R、residualIMG_G、residualIMG_B 疊加在一起，即是 residualIMG
4. psnr = computePSNR(input1_s, input2_s)
- (1) 求出 input1_s 的最大值，當作 MAX_I
 - (2) 利用 immse(input1_s, input2_s)，得到 MSE
 - (3) 將 MAX_I 、MSE 代入以下公式，求得 PSNR

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) = 20 \cdot \log_{10} \left(\frac{MAX_I}{\sqrt{MSE}} \right)$$

註. 使用 uint8 型別做運算

結果圖片



圖 1 Reference Image—frame 437



圖 2 Target Image—frame 439

Full Search, $p=8$, block size=8x8



圖 3 Predict Image



圖 4 Motion Vector



圖 5 Residual Image

Full Search, $p=8$, block size=16x16



圖 6 Predict Image



圖 7 Motion Vector



圖 8 Residual Image

Full Search, $p=16$, block size=8x8



圖 9 Predict Image



圖 10 Motion Vector



圖 11 Residual Image

Full Search, $p=16$, block size=16x16








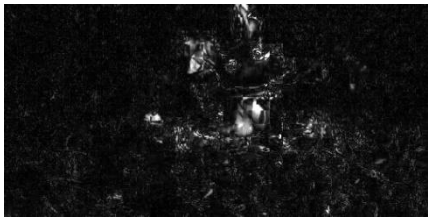

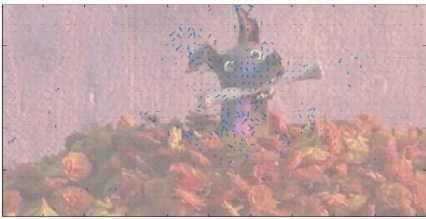




圖 12 Predict Image



圖 13 Motion Vector



圖 14 Residual Image

3-Step Search, $p=8$, block size= 8×8		
		
圖 15 Predict Image	圖 16 Motion Vector	圖 17 Residual Image
3-Step Search, $p=8$, block size= 16×16		
		
圖 18 Predict Image	圖 19 Motion Vector	圖 20 Residual Image
3-Step Search, $p=16$, block size= 8×8		
		
圖 21 Predict Image	圖 22 Motion Vector	圖 23 Residual Image
3-Step Search, $p=16$, block size= 16×16		
		
圖 24 Predict Image	圖 25 Motion Vector	圖 26 Residual Image

PSNR				
	p=8 block size=8x8	p=8 block size=16x16	p=16 block size=8x8	p=16 block size=16x16
Full Search	28.7054	27.1772	29.5829	27.6819
3-Step Search	28.1258	26.8857	28.3763	27.1451

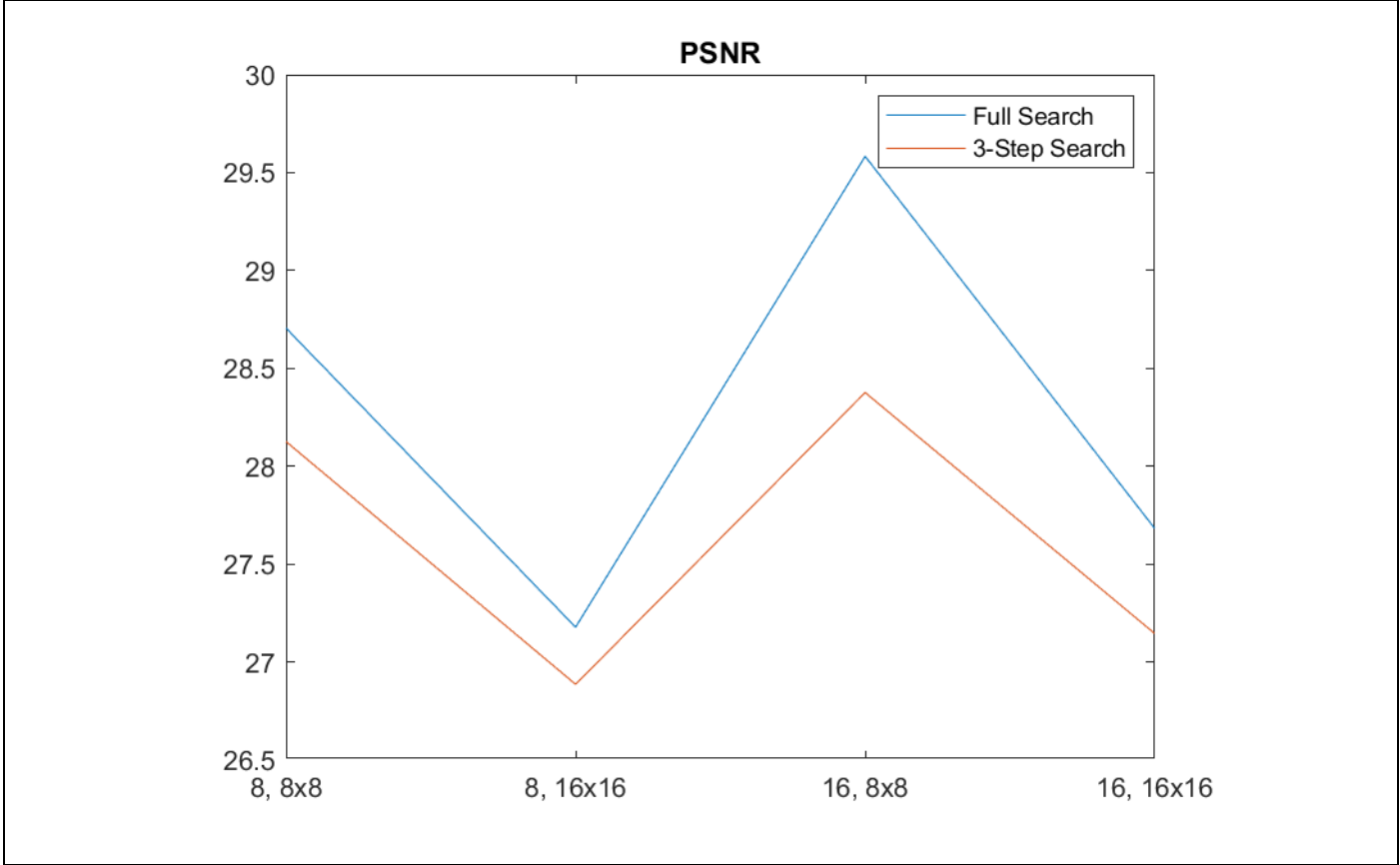


圖 27 各種狀況下的 PSNR

※註. PSNR 是用 uint8 格式計算

SAD				
	p=8 block size=8x8	p=8 block size=16x16	p=16 block size=8x8	p=16 block size=16x16
Full Search	8704.89	9944.32	8354.28	9677.73
3-Step Search	9069.93	10124.07	8943.86	9980.17

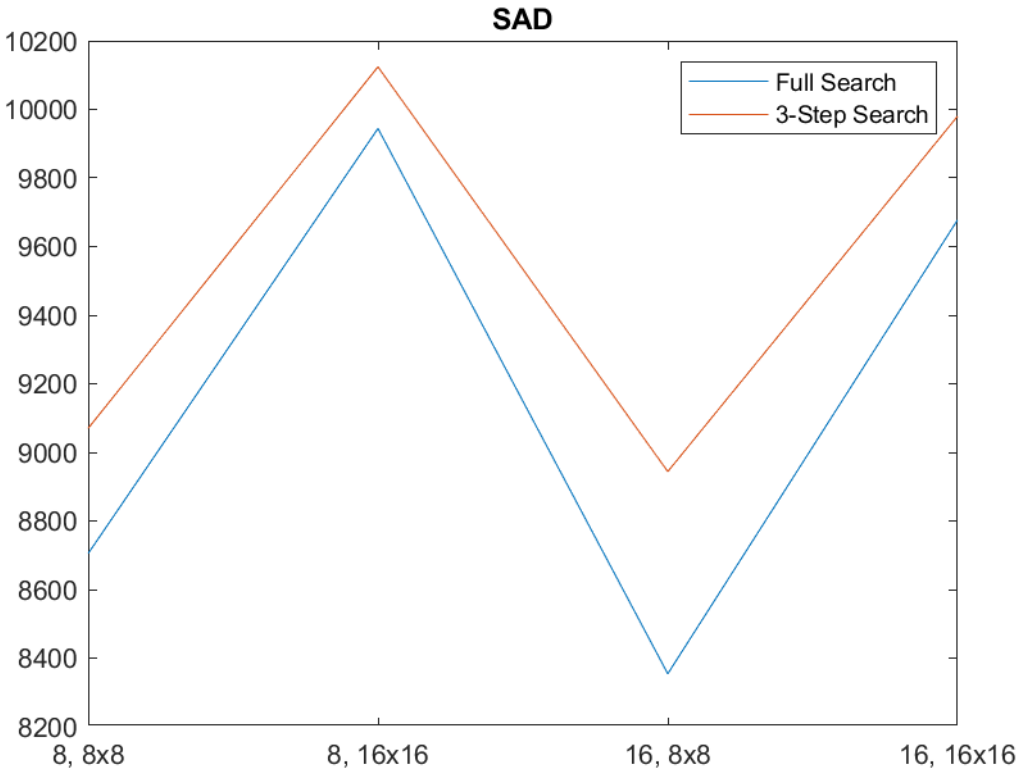


圖 28 各種狀況下的 SAD

※註. SAD 是用 double 格式計算

分析以及討論

1. Search Method 對 output results 的影響

Full Search 會搜尋 target block 範圍內的所有 reference block，和 3-Step Search 相比，有更多的 reference block，所以可以找到比 3-Step Search 更好的結果。

因此，Full Search 會有更高的 PSNR，更低的 SAD。

2. Search Range 對 output results 的影響

Search Range 越大就能得到更多的 reference block，也就可以得到更好的結果。

因此，p=16 會有更高的 PSNR，更低的 SAD。

3. Block Size 對 output results 的影響

Block Size 越小就代表每個 block 中累積的誤差會更低，也就可以得到更好的結果。

因此，b=8 會有更高的 PSNR，更低的 SAD。



4. Discuss the relation between SAD and PSNR




PSNR 是的值越高，就代表和原圖越相像，而 SAD 的值越高，就代表和原圖的誤差越多。

可以看出兩者的代表的意義剛好相反，因此 PSNR 和 SAD 會呈現負相關。

Q2. Change The reference image to be frame432.jpg, with Full Search, p=8, block size=8

結果圖片

	
圖 29 Reference Image—frame 432	圖 30 Target Image—frame 439

Full Search, p=8, block size=8x8		
		
圖 31 Predict Image	圖 32 Motion Vector	圖 33 Residual Image

	Reference Image—frame 437	Reference Image—frame 432
PSNR	28.7054	23.0774
SAD	8704.89	13824.43

※註. PSNR 是用 uint8 格式計算，SAD 是用 double 格式計算

分析以及討論

1. Compare and discuss the PSNR with the result

和 frame 437 相比，frame 432 比 target image(frame 439)相異更大，像是狗舌頭就很明顯在 frame 432 就沒有出現。因此 frame 439 的屬於舌頭的那些 target block，就很難找到相似的 reference block，導致 PSNR 下降。

Q3. Analyze the time complexity

- a. Measure the execution time required for the two search algorithms with the two different search range sizes ($p=8$ and $p=16$).

只考慮 Search 的時間，也就是做出 Predict Image 的時間，其餘像是 `imread()`、`imwrite()` 的讀寫動作不納入計算。

Search Method	Search Range	Block Size	Execution time(s)
Full Search	8	8x8	1.734217
Full Search	16	8x8	6.290722
3-Step Search	8	8x8	0.228661
3-Step Search	16	8x8	0.280484

- b. Compare and discuss the execution time with the theoretical time complexity.

先從對每一個 block 的所需計算量(in worst case)來看， p 是 Search Range， b 是 block size

Full Search :

(1) 每找到一個 reference block 需要對 $b \times b$ 個 element 去做運算 $\Rightarrow b^2$

(2) 要找 $p \times p$ 個 reference block $\Rightarrow p^2$

因此，運算量為 $b^2 p^2$ 。

按照上題表格，Full search 的情況下， p 變成 2 倍(從 8 變成 16)，時間相差 $\frac{6.290722}{1.734217} \approx 3.6$ 倍，和理論中的 4 倍相近。

3-Step Search :

(1) 對一個 block 要搜尋 $\log_2(p)$ 次

(2) 每一次搜尋都要找 9 個 reference block

(3) 每找到一個 reference block 需要對 $b \times b$ 個 element 去做運算 $\Rightarrow b^2$

因此，運算量為 $9b^2 \log_2(p)$

按照上題表格，3-Step search 的情況下，p 變成 2 倍(從 8 變成 16)，時間相差 $\frac{0.280484}{0.228661} \approx 1.22$ 倍，

和理論中的 $\frac{\log_2(16)}{\log_2(8)} = 1.33$ 倍相近。