- Data Compression Final Project -

Novel Fast Block Motion Estimation Algorithms

指導老師 ： 戴顯權 教授

學生:

組員一 ： 黃琨懿

學號 ： N26931815

Email : hki95@beethoven.ee.ncku.edu.tw

組員二 ： 邱政斌

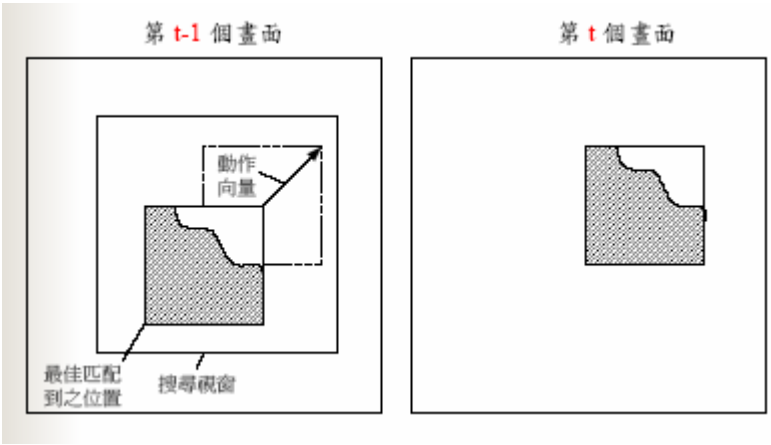
學號 ： N26944614

Email : ccp96@beethoven.ee.ncku.edu.tw

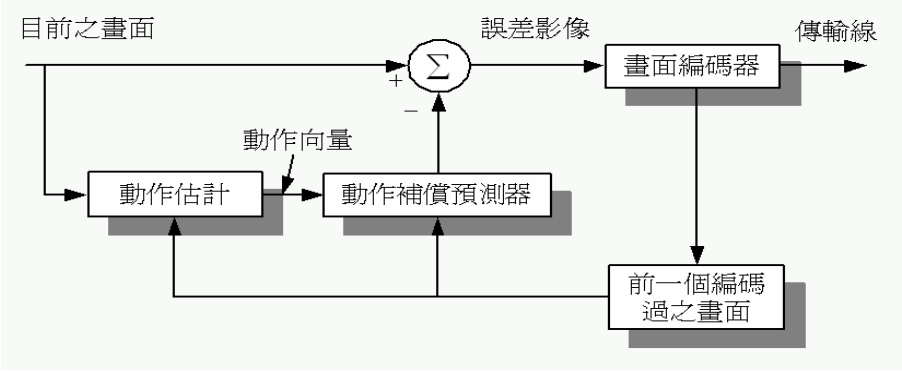
系所: 電機所 VLSI/CAD組

I. Introduction

近年來，一些視訊壓縮的標準如:ISO/IEC MPEG-1/2/4以及ITU-T H.261/263/264都採取使用方塊動作估計(block motion estimation)來降低畫面間累贅(temporal redundancy)。由於畫面間有其累贅性也就是在時間為T的這一個畫面與T-1或是T-2…等有其相關性(圖一)，我們可以利用方塊比對(Block-Matching)來找出T時間的這個畫面使用T-1組成最好的結果，之後我們只需要記錄下來每個方塊比對的結果也就是運動的結果。這些運動軌跡被稱做動作向量(motion vector)，找出動作向量的過程就是所謂的動作估計，之後將動作向量編碼傳送出去，完整的編碼流程於(圖二)。



圖一



圖二

而使用動作估計對於資料壓縮的好處在於如果將一個完整的方塊傳送出去需要8*8*8(Gray level)，總共需要512 bits，但如果在於搜尋範圍為+-8，動作估計所需要花費的編碼是4*2(u、v) ，總共需要8 bits即可完成編碼一個方塊。

方塊比對演算法(block-matching algorithm)可說是目前主流的動作估計演算法，有一些比較好的方法也已被採用為標準。最基本的搜尋法是利用Full Search(FS)演算法來實現，在參考畫面中一已定義的搜尋範圍(search range)內一筆一筆的找尋與目標方塊之SAD(sum of absolute difference)最小的方塊，並決定其動作向量。雖然其正確率較高，但搜尋時間與運算複雜度也跟著提升。為了減少運算複雜度，有很多快速的搜尋法被提出，包括了三步驟搜尋(3-step search, 3SS)，四步驟搜尋(4-step search, 4SS)，BBGDS(Block-Based Gradient Descent Search)，DS(Diamond Search)，HEXBS (Hexagon-Based Search)。

在2005年二月的IEEE Transactions on Multimedia中提出了兩個混合型(Cocktail)演算法，混合了十字、菱形以及六角形演算法，稱為Novel Cross-Diamond-Hexagonal Search(CDHS)，他的運算速度比FS快了將近30倍，然而誤差在0.01。

在2002也有提出類似的想法混合了十字與菱形演算法，另外也有加強加快六角形演算法，分別是Novel Cross-Diamond Search(CDS)以及Enhanced Hexagonal Search(EHEX)。

本報告中總共實做了CDHS兩種演算法:CDHS-T與CDHS-F，另外類似想法的CDS與其比較的有Cross Search、Diamond Search與Hexagonal Search，另外改良版本的Enhanced Hexagonal Search，最後最重要的當然是比較的基準Full Search。

接下來介紹的Background包括Full Search、Cross Search、Diamond

Search、Hexagonal Search，緊接著介紹Novel Search Algorithm包括 Enhanced Hexagonal Search 、 Cross-Diamond Search 、 Cross-Diamond-Hexagonal-T Search 與 Cross-Diamond-Hexagonal-F Search。

II. Background

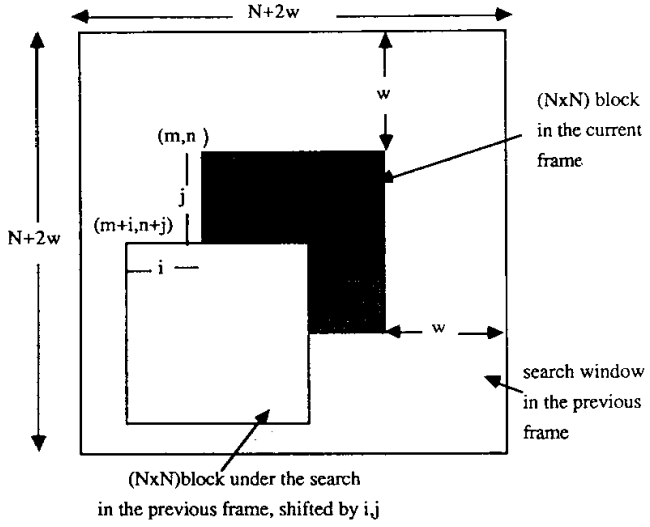
(1) Full Search (FS)

在處理移動位移估計的演算法之中，以全區域搜尋演算法(Full Search Block Match)最能夠有效地減少畫面間多餘的資訊量，顧名思義，在搜尋視窗內的每一點皆須逐點計算相似性，故需要大量的運算量。

如圖三所示，current frame為N * N方塊f(m,n)與previous frame中所有在搜尋視窗內之N * N方塊g(m+i,n+j)做比對，

$$SAD = \sum |f(m,n) - g(m+i,n+j)|$$

選出使得SAD最小的向量，V=(i,j) ,即是我們要的動作向量。搜尋視窗以f(m,n)為中心，水平和垂直方向分別向左右和上下延伸w，大小是(N+2*w)*(N+2*w)。

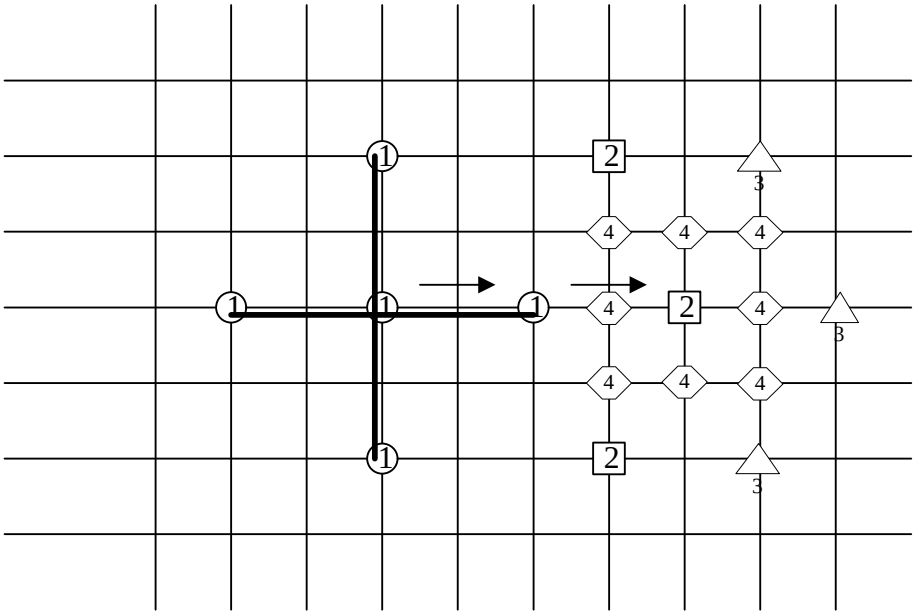


圖三

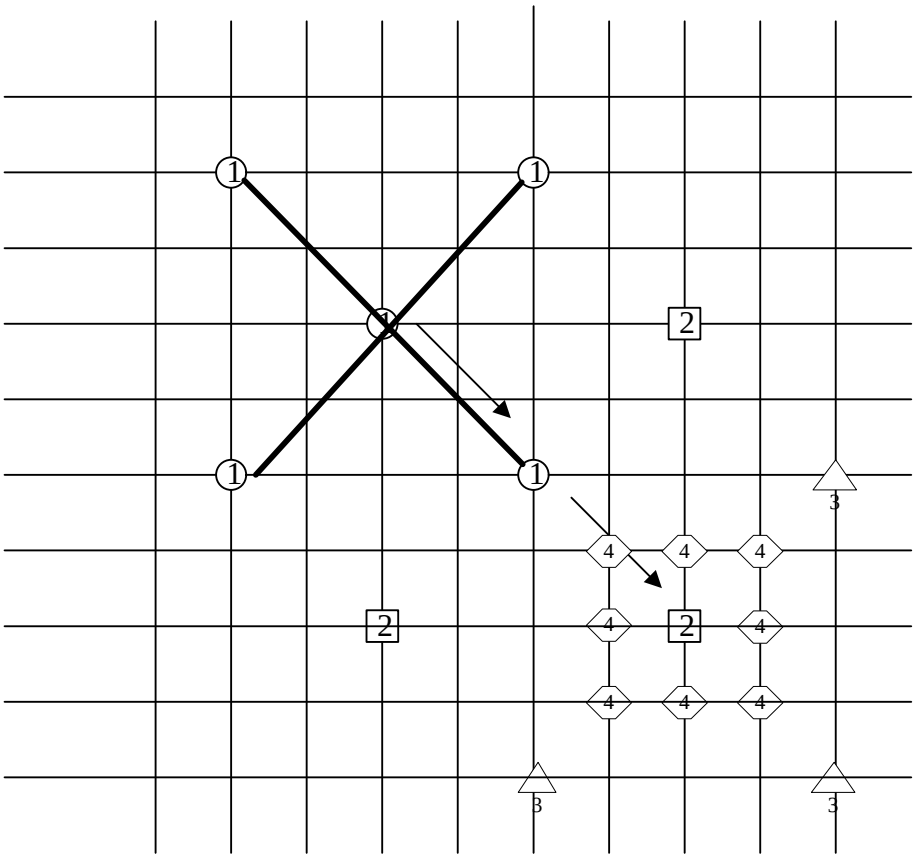
(2) Cross Search (CS)

Cross Search 分別有兩種的形式:一種是正十字(圖四)另一種斜十字(圖五)。從中央開始，找大交叉十字型的五個點來比較差距。如果

最小的點不是在中央分別在找尋附近三個點，如果在中央則依中央點的周圍八點做比較，直到找到最相近的點為止。而整個演算法的搜尋點數為 5+3*n+8，此次我們實做的為斜十字之 Cross Search。



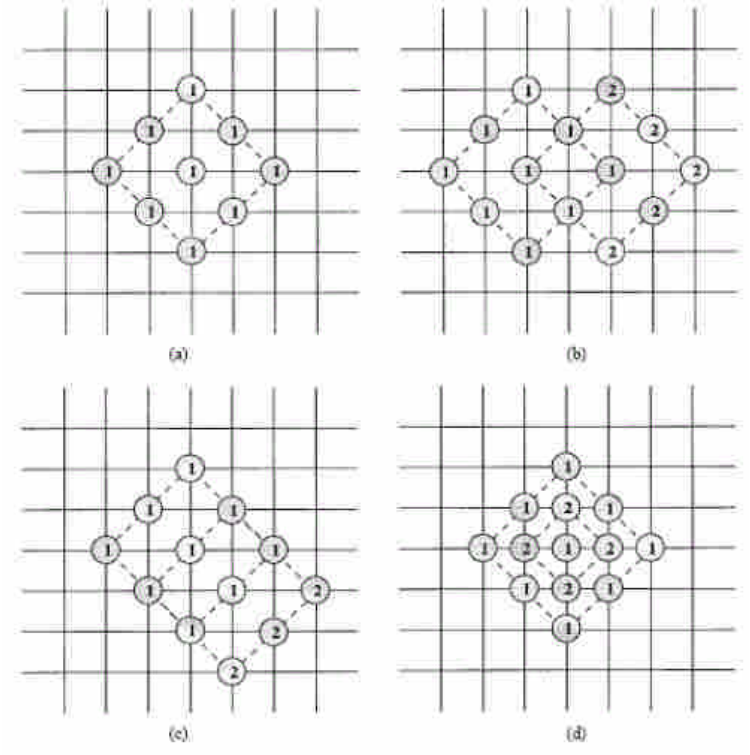
圖四



圖五

(3) Diamond Search (DS)

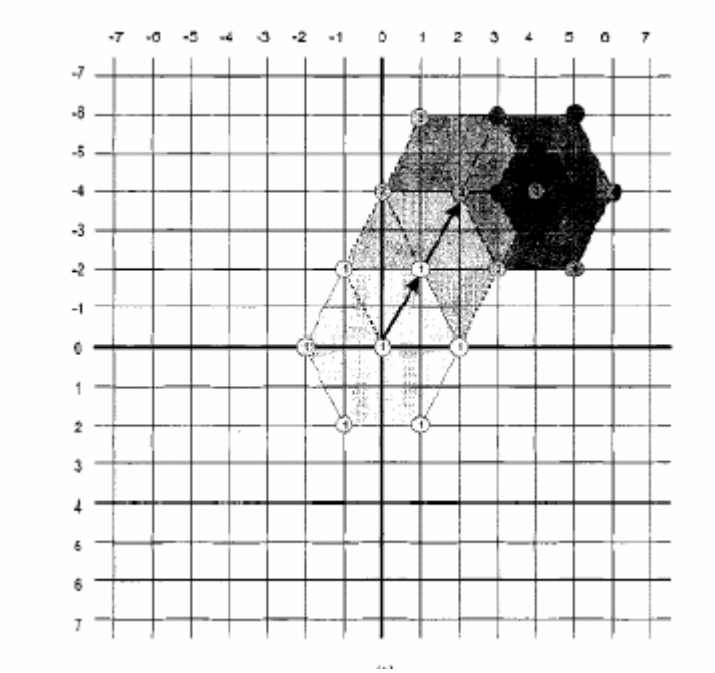
從中央開始，找周圍九個點來比較差距。找到最相近的點以後再把菱形中心移到那一點。若最相近的點為中央點，則把中央點相鄰的四個點繼續做比較。Diamond search 可以達到平均 13 個搜尋點。做法如圖六，整個演算法的搜尋點數為 9+(3,5)*n+4。



圖六

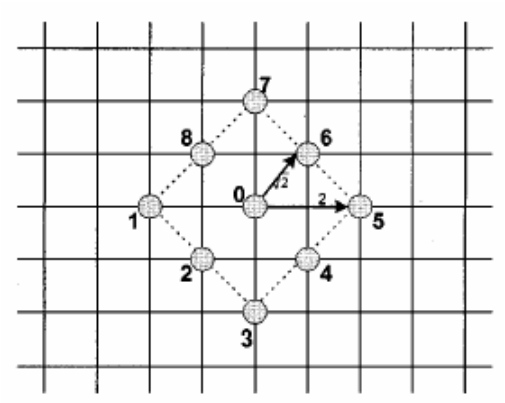
(4) Hexagon-Based Search (HEXBS)

從中央開始，找周圍六個點來比較差距。找到最相近的點以後再把六邊形中心移到那一點。如果最相近的點為中央點，則把六邊形縮小找周圍四個點來比較。做法如圖七，而整個演算法的搜尋點數為 7+(3)*n+4。



圖七

HEXBS 進一步改進了 diamond search 的搜尋點數分布狀態，因為在 DS 中，每一個 diamond 周圍的八個點與中心的距離不盡相同，有 2 及$\sqrt{2}$兩種(如圖八)，以 HEXBS 而言，周圍六個點的距離有$\sqrt{5}$及 2 兩種(圖九)，因此，不但搜尋的點比 DS 少，也比 DS 更接近圓形。



圖八

六角形搜尋法一開始是利用較大的六角形進行搜尋(如圖九)此大六角形(Large hexagonal search pattern, LHSP)，總共六個點，然而當最小誤差值出現在中心的時候便改使用小六角形(Small hexagonal

search pattern, SHSP) 進行搜尋(如圖十)，總共四個搜尋點。

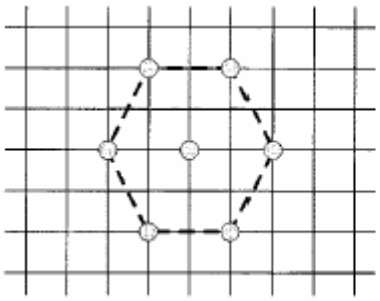
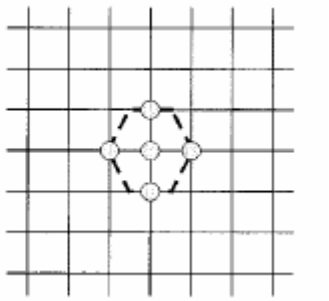


圖 九

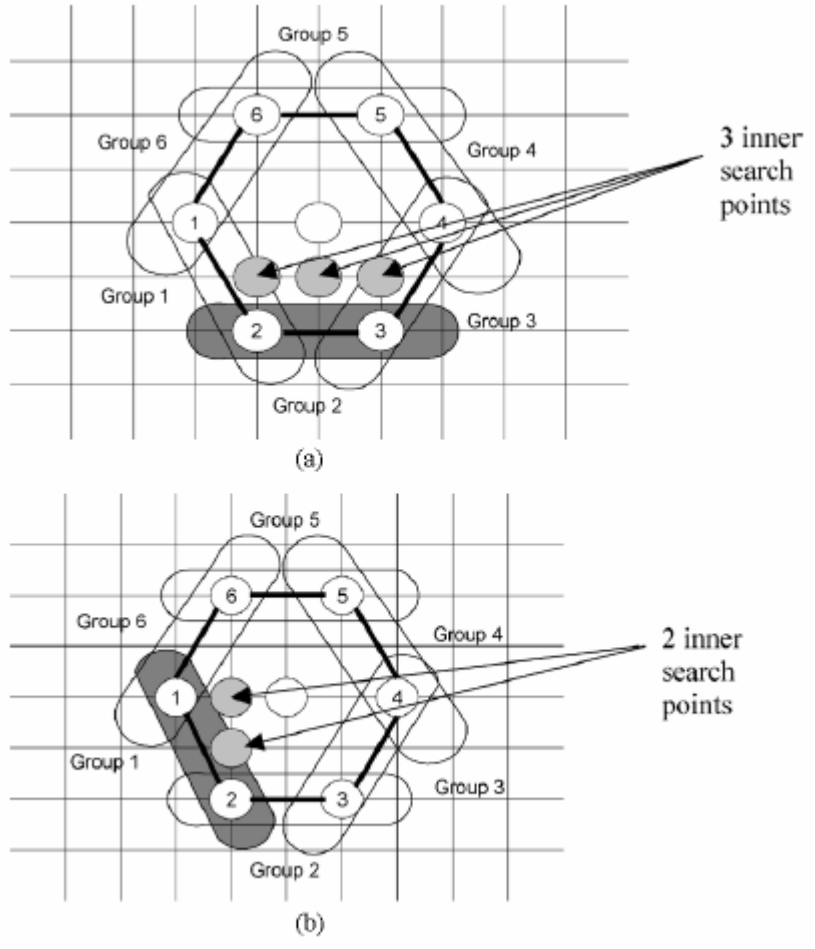


圖十

III. Novel Motion Estimation Methods

(1) Enhanced Hexagon-Based Search (EHEXBS)

在 HEXBS 中，第二階段的 inner search 須搜尋最小值中心點周圍四個點，而在 EHEXBS 中則利用了在 global minimum 周圍的單調失真特性(monotonic distortion characteristic)，使得 inner search 只需搜尋一個特定方向的 inner points，如圖十一所示，EHEXBS 額外計算了周圍六個 group 內對應的兩個方塊的 SAD 和，如(a)所示，假如最小 SAD 和是 group2，只須搜尋額外 3 個點，如(b)所示，如果是 group1，則只須搜尋額外 2 個點。因此，在搜尋速度和失真的表現上皆比 HEXBS 好。

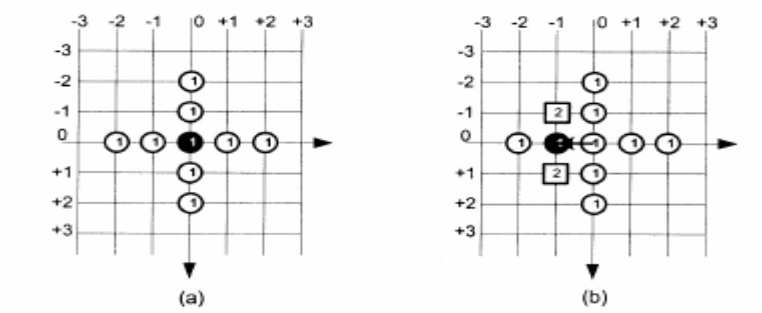


圖十一

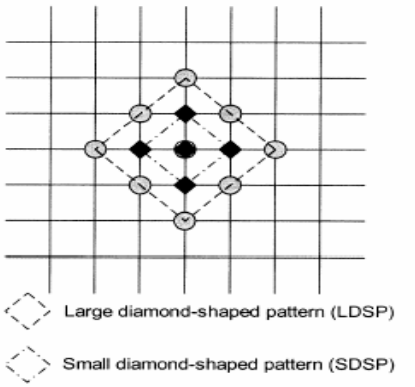
由於六角形搜尋法最後一步是搜尋中心點的邊緣四個點，而 Enhanced 六角形搜尋法根據 group 分類分別減少 1 或兩個搜尋點，所以大約減少 2*(4/6)+1(2/6)=1.67 個搜尋點，因為是每個方塊都減少了大約 1.67 個搜尋點，所以一個 frame 就減少了相當多的搜尋點，自然其速度比六角形搜尋法快很多。而整個搜尋法的搜尋點數為 7+3*n+(2,3)。

(2) Cross-Diamond Search (CDS)

在 CDS 中，第一階段為 Cross Search 方式，第三階段則利用形成的菱形做 Diamond Search，而在第一階段與第二階段 Cross Search 時，即可以少計算到中央小十字的五個點，之後計算方式則為 Diamond Search。第一、二階段如圖十二所示。第三、四階段如圖十三所示。

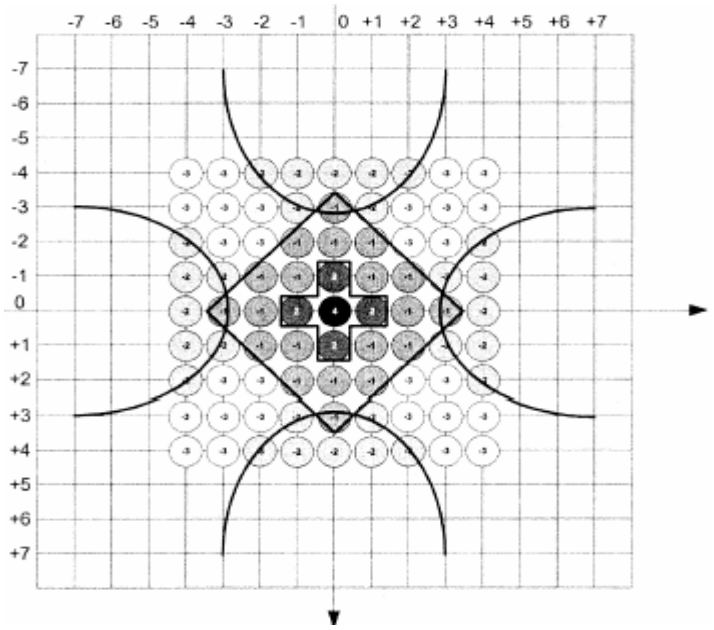


圖十二

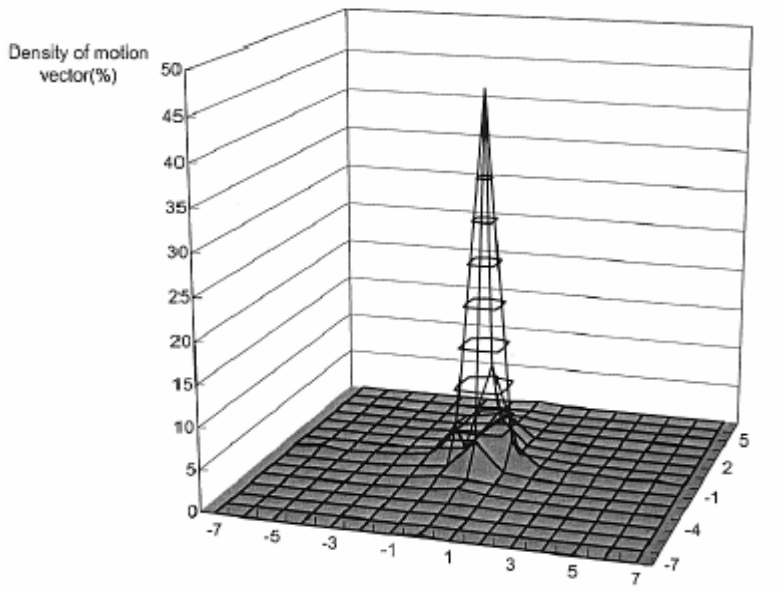


圖十三

Cross-Diamond 的優點便是在於最佳匹配的方塊在(0,0)附近時可以比 Diamond 減少 4 或 2 個搜尋點(如圖十四)，但其他地方卻會比菱形搜尋法多搜尋一些點，但是根據統計動態向量幾乎都是集中在於(0,0)附近(如圖十五)，所以整體來說可以有效的減少搜尋點數。

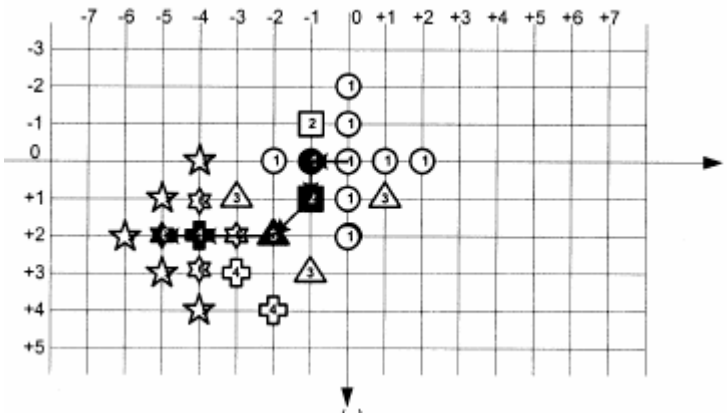


圖十四



圖十五

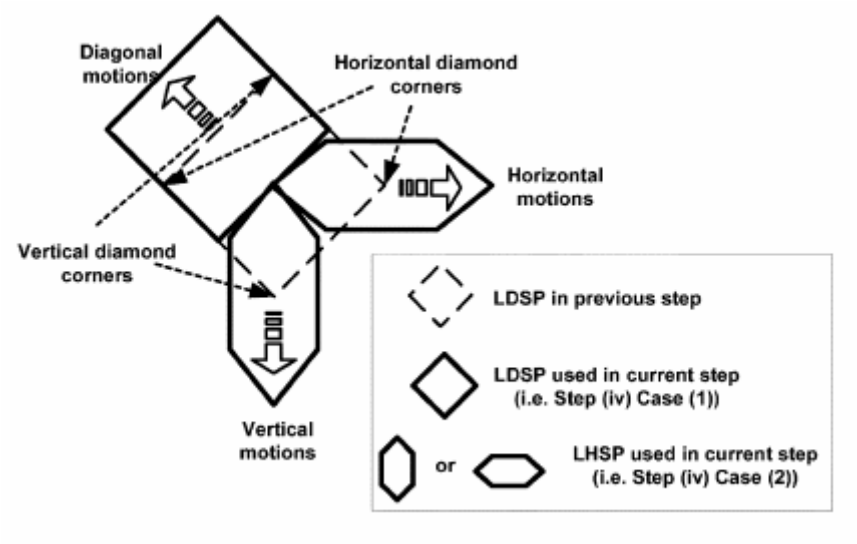
(圖十六)是一個 Cross-Diamond 的例子，第一步為九點十字，而最小點的位置為(-1,0)，此時再多搜尋(-1,-1)、(-1,1)兩點，這為半菱形，第三步發現最小的位置為菱形邊緣的位置，所以向外多搜尋三點，而第四步相同多搜尋三點，但第五步為菱形的角(Corner)，所以向外多搜尋五點，發現最小點在正中間，所以搜尋附近四點，結束搜尋。



圖十六

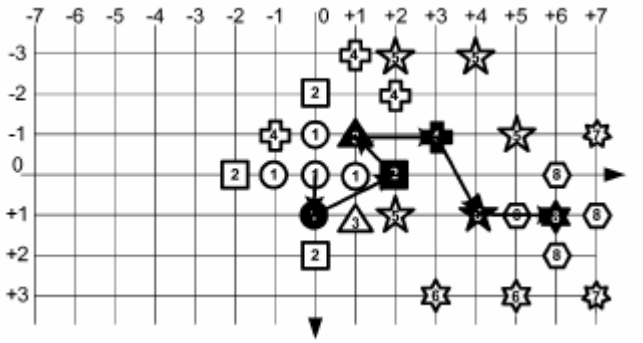
(3) Cross-Diamond- Hexagonal Search (CDHS)

在 CDHS 中，第一階段為 Cross Search 方式，第三階段若最相似的點為菱形的邊緣，則利用形成的菱形做 Diamond Search，若最相似的點為菱形的角，則做 Hexagon Search，一但進入 Hexagon Search 則一直做 Hexagon Search 直到找到最相似的點。而在第一階段與第二階段 Cross Search 時，即可以少計算到中央小十字的五個點，之後計算方式則為 Diamond Search 與 Hexagon Search，而 Hexagon Search 又分為垂直的 Hexagon 與水平的 Hexagon。方向性如圖十七所示。



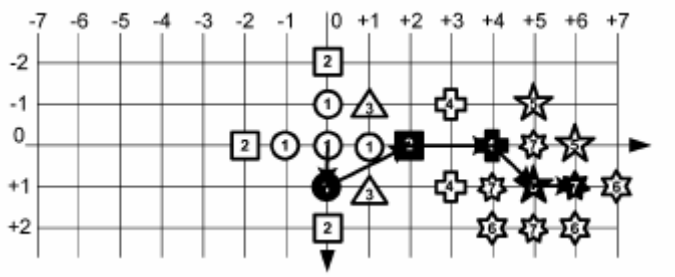
圖十七

由於 CDHS 中分成 T 以及 F 兩種搜尋法所以分別舉例說明。(圖十八)是一個 CDHS-T 的例子，第一步為十字搜尋法的五點，如果最小點在正中心(0,0)便停止搜尋，若沒有則再多搜尋四點如同十字搜尋法的九點，接著第三步如同 CDS 一樣多搜尋兩點，這是半菱形，若最小點在半菱形的邊緣(Side)則維持菱形搜尋法的向外多搜尋三點，但是若在角落(Corner)則變成六角形搜尋法，此後便一直使用六角形搜尋法，而此處使用的是 T 的六角形，可發現第五、六、七步都是只用六角形搜尋法，但唯一不同的是第五步需要搜尋五點，但六、七兩步只需要搜尋三點，如同六角形一樣當最小點為中心點時，則搜尋中心附近四點，而後結束搜尋。



圖十八

(圖十九)是一個 CDHS-F 的例子，第一步為十字搜尋法的五點，如果最小點在正中心(0,0)便停止搜尋，若沒有則再多搜尋四點如同十字搜尋法的九點，接著第三步如同 CDS 一樣多搜尋兩點，這是半菱形，若最小點在半菱形的邊緣(Side)則維持菱形搜尋法的向外多搜尋三點，但是若在角落(Corner)則變成六角形搜尋法，此後便一直使用六角形搜尋法，而此處使用的是 F 的六角形，可發現第五、六步為 F 形的六角形，第七步為搜尋中心點附近的四點，而後結束搜尋。

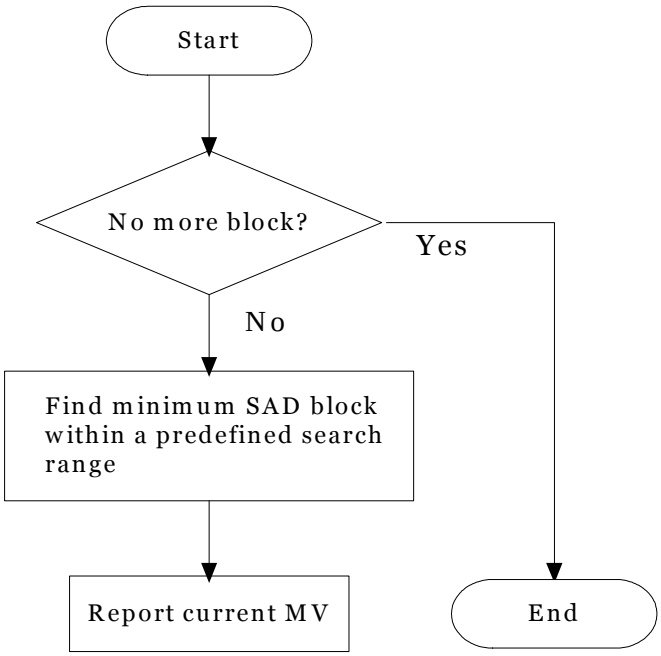


圖十九

IV. Implementation

(1) Full Search (FS)

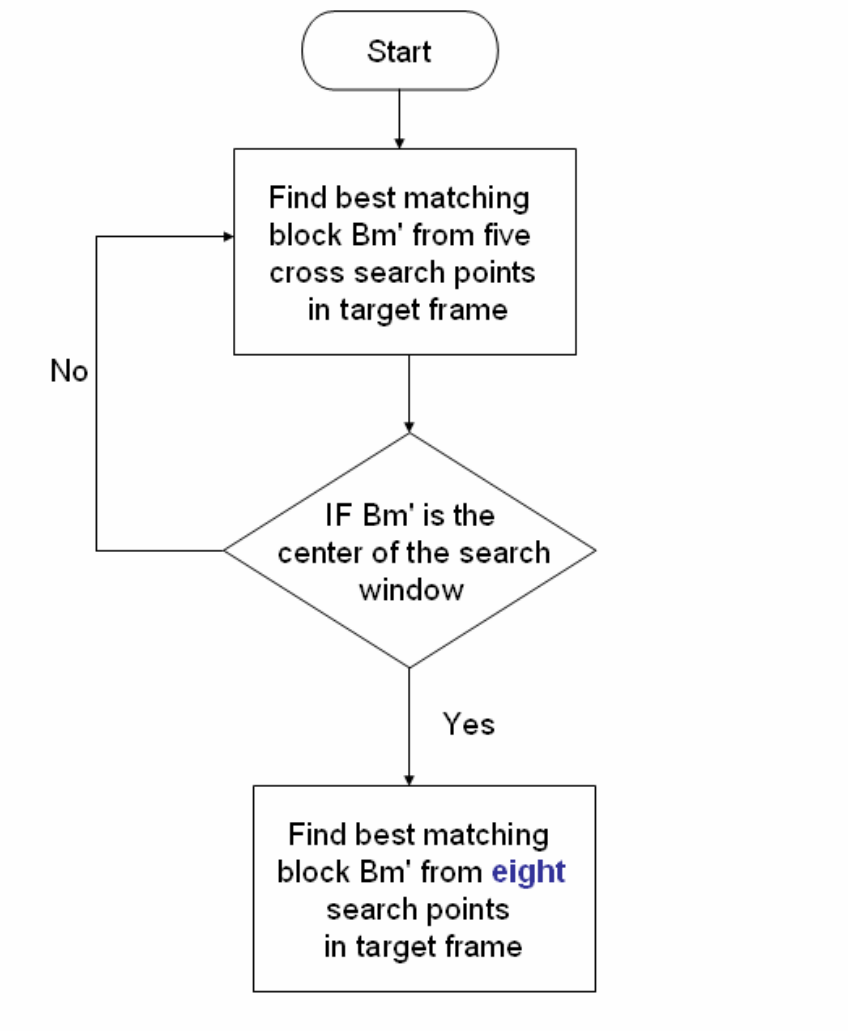
對初始的搜尋中心點置於搜尋視窗的中心，一次位移一個像素，比對搜尋視窗中每個相對應方塊的 SAD 值，最後以 SAD 值最小的方塊為最終搜尋結果。圖二十為其流程圖。



圖二十

(2) Cross Search (CS)

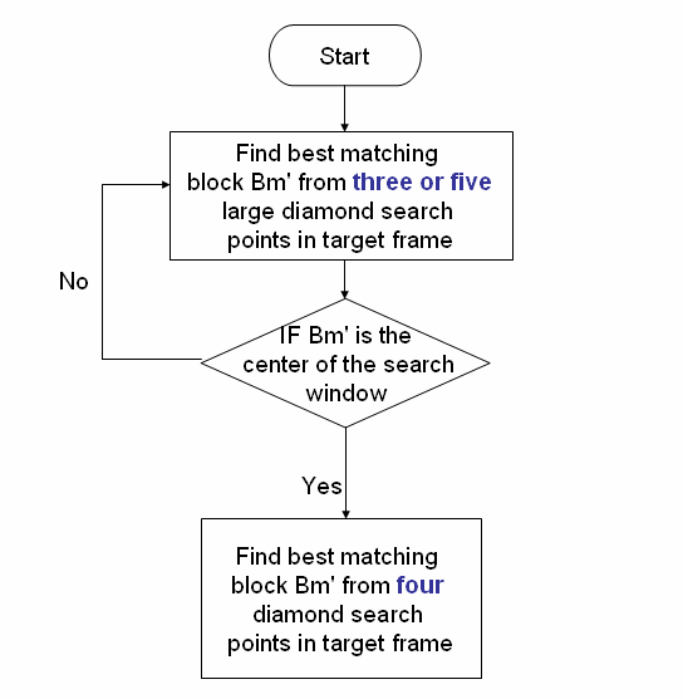
對初始的搜尋中心點置於搜尋視窗的中心，以大交叉十字型的五個點來比對搜尋視窗中每個相對應方塊的 SAD 值。找到最相近的中央點以後，再依中央點的周圍八點做比較，最後以 SAD 值最小的方塊為最終搜尋結果。圖二十一為其流程圖。



圖二十一

(3) Diamond Search (DS)

對初始的搜尋中心點置於搜尋視窗的中心，以菱形的形狀以中心點周圍的九個點來比對搜尋視窗中每個相對應方塊的 SAD 值。找到最相近的中央點以後，再依中央點的周圍小菱形的四點做比較，最後以 SAD 值最小的方塊為最終搜尋結果。圖二十二為其流程圖。

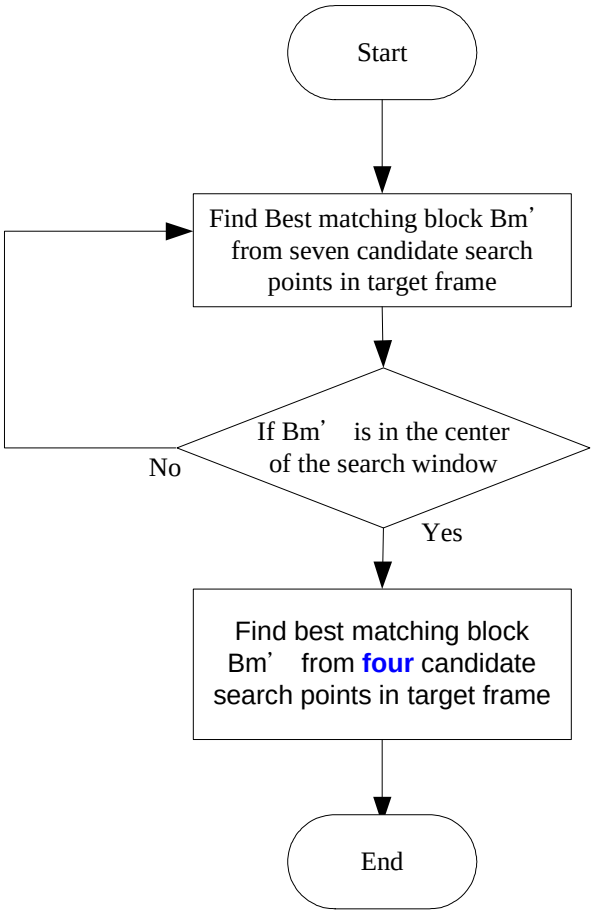


圖二十二

(4) Hexagon-Based Search (HEXBS)

Step1: 每一次 Hexagon 的移動需比較對應的七個點的 SAD 值大小,當最小值出現在中心點時，跳至 step2；否則,將新的中心點以最小 SAD 的點取代,並重複其相對應七個點的比對步驟，直到最小值出現在中心點，並跳至 step2。

Step2: 對此中心點及其周圍四個點作比對，SAD 最小值的點即為搜尋結果。如圖二十三所示為其流程圖。



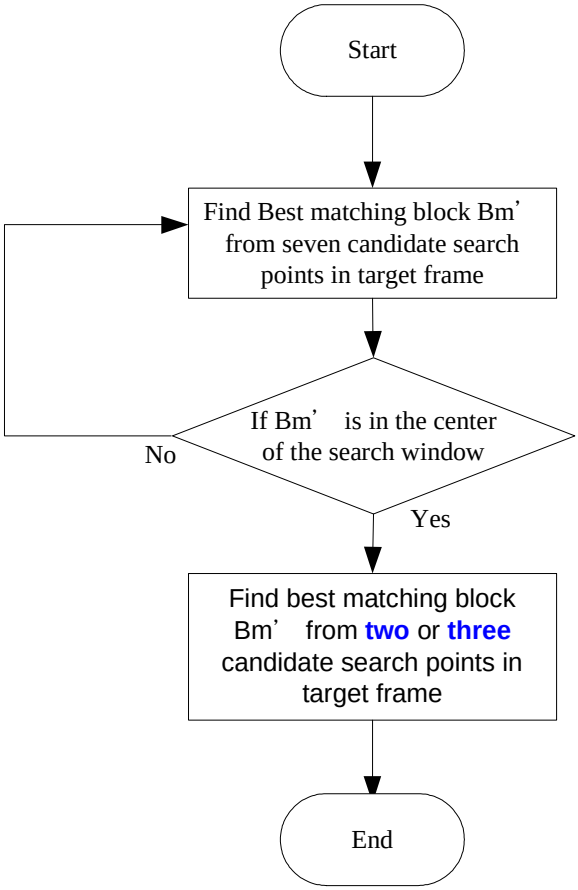
圖二十三

(5) Enhanced Hexagon-Based Search (EHEXBS)

Step1: 每一次 Hexagon 的移動需比較對應的七個點的 SAD 值大小，並計算相對應六個 group 內兩個點的 SAD 總和，當最小值出現在中心點時，跳至 step2；否則；將新的中心點以最小 SAD 的點取代;並重複其相對應七個點的比對步驟及相對應六個 group 內兩個點 SAD 總和的計算,直到最小值出現在中心點,並跳至 step2。

Step2: 根據六個 groups 中最小值出現的位置決定與中心點作比對的 inner points，比較之後所得到 SAD 最小值的點即為搜尋結果。

如圖二十四所示為其流程圖。



圖二十四

(6) Cross-Diamond Search (CDS)

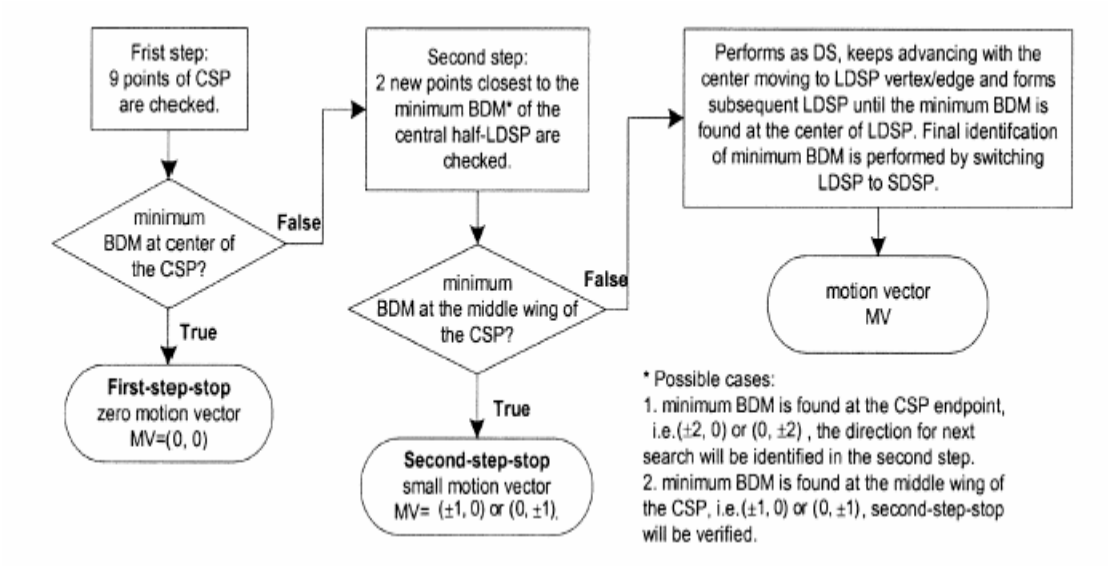
Step1：對初始的搜尋中心點置於搜尋視窗的中心，並做 Cross-shaped pattern (CSP)，找出最小的 BDM point，若中心點為最小點，則 search 結束否則跳到 Step2。

Step2：根據上下左右四個方向決定要增加哪兩點，若最小的 BDM point 在 $(\pm 1,0)$ 或 $(0,\pm 1)$，則 search 結束否則跳到 Step3。

Step3：做 Large diamond search pattern (LDSP)，比較對應點的 SAD 值大小,當最小值出現在中心點時，跳至 step4；否則，將新的中心點以最小 SAD 的點取代，並重複其相對應 LDSP 點的比對步驟，

直到最小值出現在中心點，並跳至 step4。

Step4: 做 Small diamond search pattern (SDSP)，對此中心點及其周圍四個點作比對，SAD 最小值的點即為搜尋結果。如圖二十五所示為其流程圖。



圖二十五

(7) Cross-Diamond- Hexagonal Search with large hexagonal search pattern (CDHS-T)

Step1：對初始的搜尋中心點置於搜尋視窗的中心，並做 Small cross-shaped pattern (SCSP)，找出最小的 BDM point，若中心點為最小點，則 search 結束否則跳到 Step2。

Step2：再做 Large cross-shaped pattern (LCSP)，找出最小的 BDM point 為上下左右四個方向中的哪個方向，跳到 Step3。

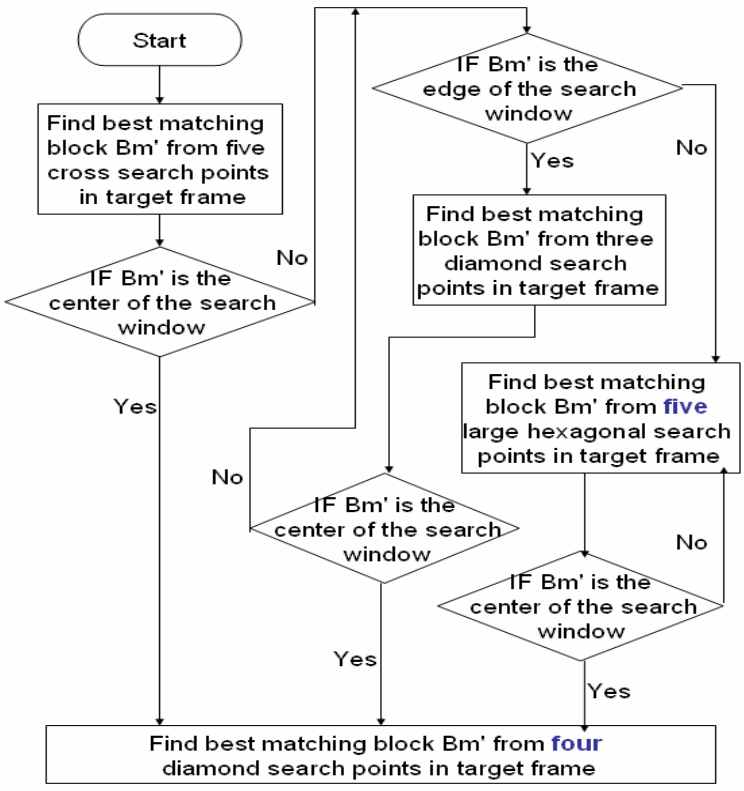
Step3：根據上下左右四個方向決定要增加哪兩點，若最小的 BDM point 在$(\pm 1,0)$或$(0,\pm 1)$，則 search 結束，若最小的 BDM point 為菱形的邊時則跳到 Step4，若最小的 BDM point 為菱形的角時則

跳到 Step5。

Step4：做 Half-diamond search，比較對應點的 SAD 值大小,當最小值出現在中心點時，跳至 step6；若最小的 BDM point 為菱形的角時則跳到 Step5，若最小的 BDM point 為菱形的邊時則重複 Step4 步驟，將新的中心點以最小 SAD 的點取代，直到最小值出現在中心點，並跳至 step6。

Step5：做 Large hexagon search pattern(LHSP)，比較對應點的 SAD 值大小,當最小值出現在中心點時，跳至 step6；否則，將新的中心點以最小 SAD 的點取代，並重複其相對應 LHSP 點的比對步驟，直到最小值出現在中心點，並跳至 step6。

Step6: 做 Small diamond search pattern (SDSP)，對此中心點及其周圍四個點作比對，SAD 最小值的點即為搜尋結果。如圖二十六所示為其流程圖。



圖二十六

(8) Cross-Diamond- Hexagonal Search with small hexagonal search pattern (CDHS-F)

Step1：對初始的搜尋中心點置於搜尋視窗的中心，並做 Small cross-shaped pattern (SCSP)，找出最小的 BDM point，若中心點為最小點，則 search 結束否則跳到 Step2。

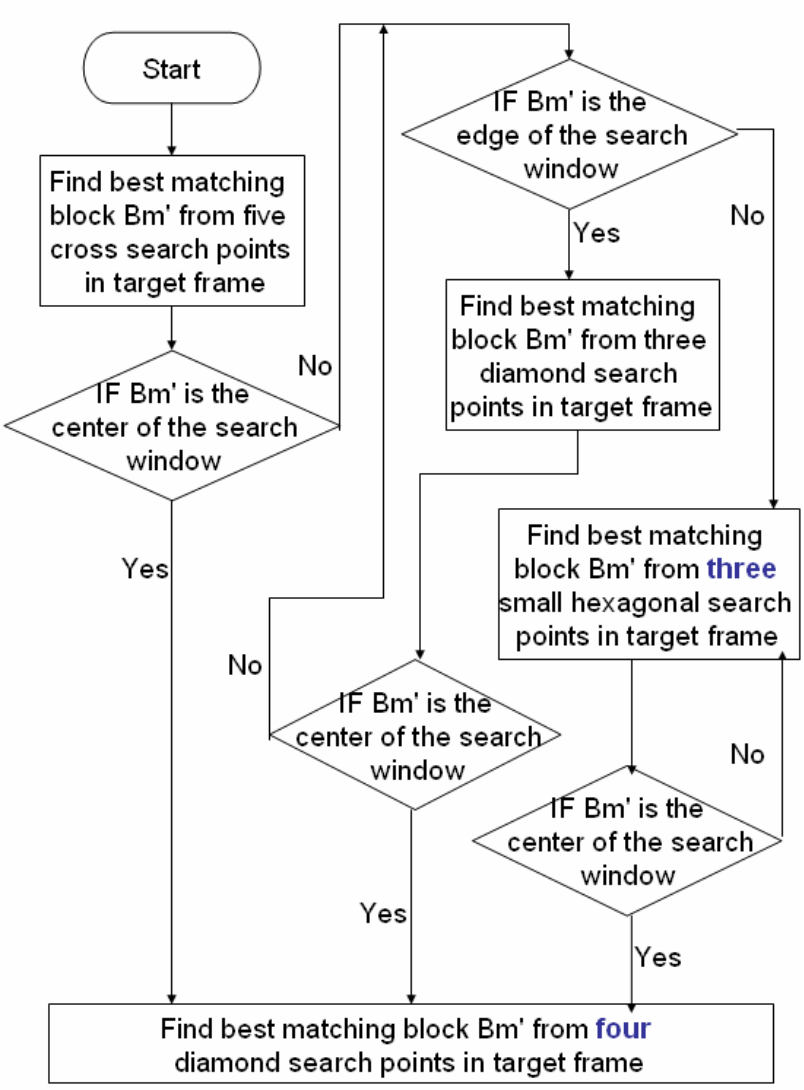
Step2：再做 Large cross-shaped pattern (LCSP)，找出最小的 BDM point 為上下左右四個方向中的哪個方向，跳到 Step3。

Step3：根據上下左右四個方向決定要增加哪兩點，若最小的 BDM point 在$(\pm 1,0)$或$(0,\pm 1)$，則 search 結束，若最小的 BDM point 為菱形的邊時則跳到 Step4，若最小的 BDM point 為菱形的角時則跳到 Step5。

Step4：做 Half-diamond search，比較對應點的 SAD 值大小,當最小值出現在中心點時，跳至 step6；若最小的 BDM point 為菱形的角時則跳到 Step5，若最小的 BDM point 為菱形的邊時則重複 Step4 步驟，將新的中心點以最小 SAD 的點取代，直到最小值出現在中心點，並跳至 step6。

Step5：做 Small hexagon search pattern(SHSP)，比較對應點的 SAD 值大小,當最小值出現在中心點時，跳至 step6；否則，將新的中心點以最小 SAD 的點取代，並重複其相對應 SHSP 點的比對步驟，直到最小值出現在中心點，並跳至 step6。

Step6: 做 Small diamond search pattern (SDSP)，對此中心點及其周圍四個點作比對，SAD 最小值的點即為搜尋結果。如圖二十七所示為其流程圖。


Start
Find best matching block Bm' from five cross search points in target frame
IF Bm' is the center of the search window
No
Yes
IF Bm' is the edge of the search window
No
Yes
Find best matching block Bm' from three diamond search points in target frame
Find best matching block Bm' from three small hexagonal search points in target frame
No
IF Bm' is the center of the search window
Yes
No
IF Bm' is the center of the search window
Yes
Find best matching block Bm' from four diamond search points in target frame

圖二十七

V. Experimental result

這裡我們實作了八個方法，使用了兩個 Sequence，分別為 Weather 以及 Stefan，所使用的 Block Size 皆為 8，所使用的序列為 I frame 後面連續接 9 個 P frame (IPPPPPPPPP)。

使用平台—

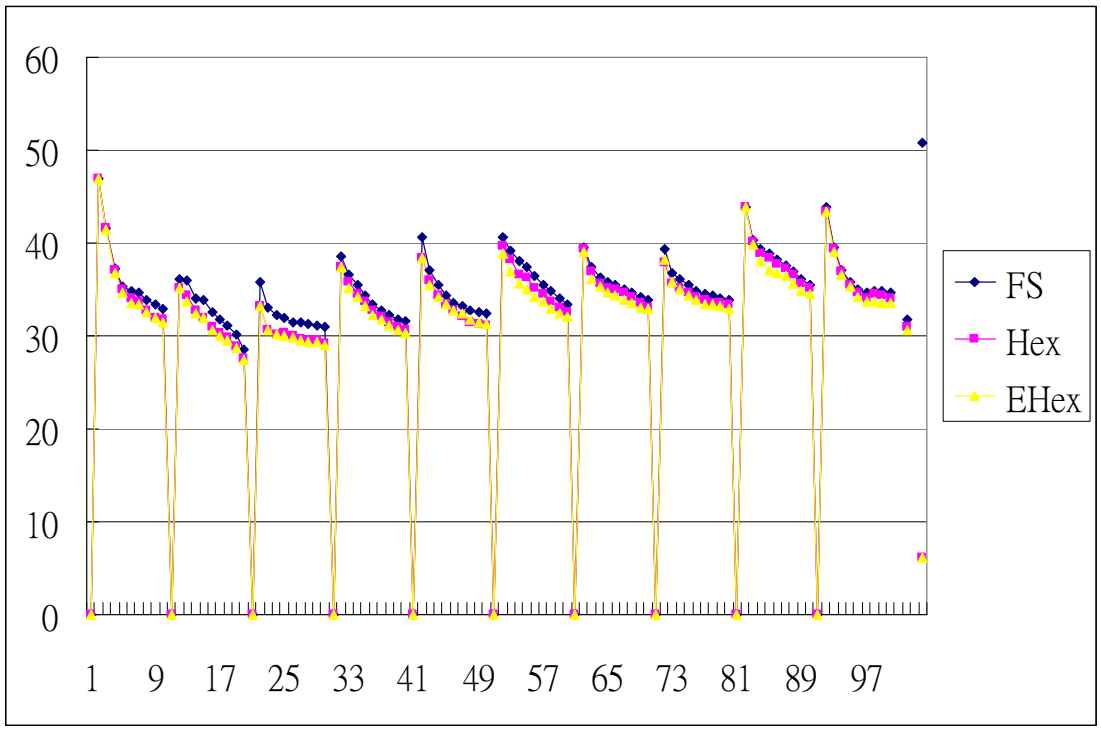
CPU：Intel Pentium 4 3.0 GHz

OS：Window XP

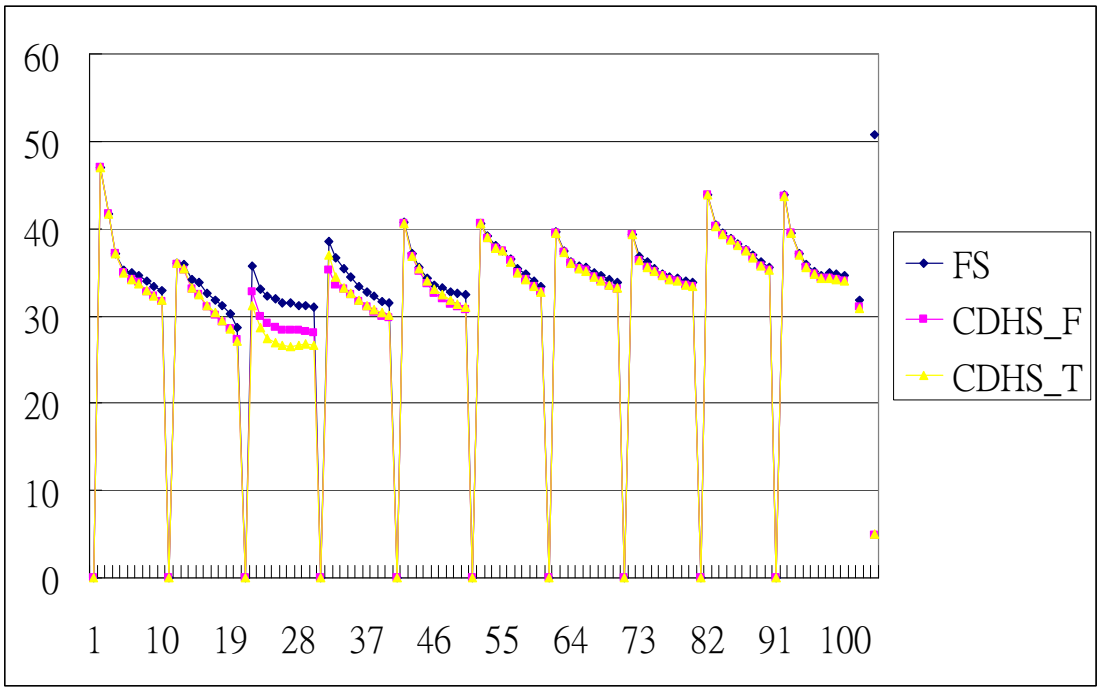
Compiler：VC++ 6.0

RAM：1.5 GB

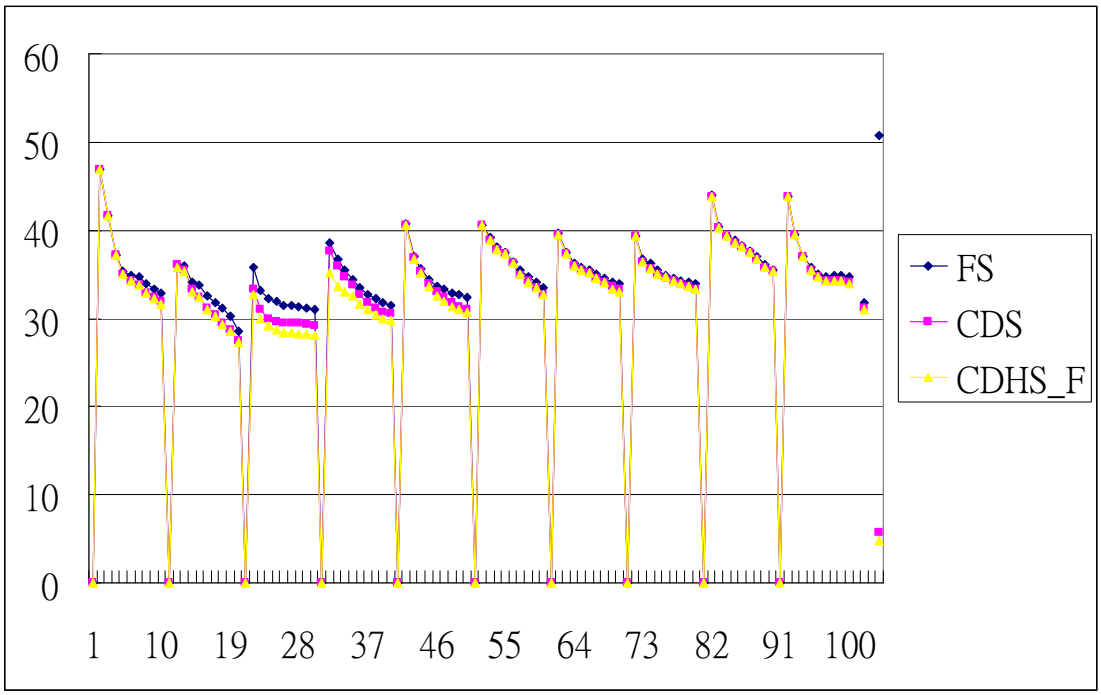
A. Sequence “Weather”



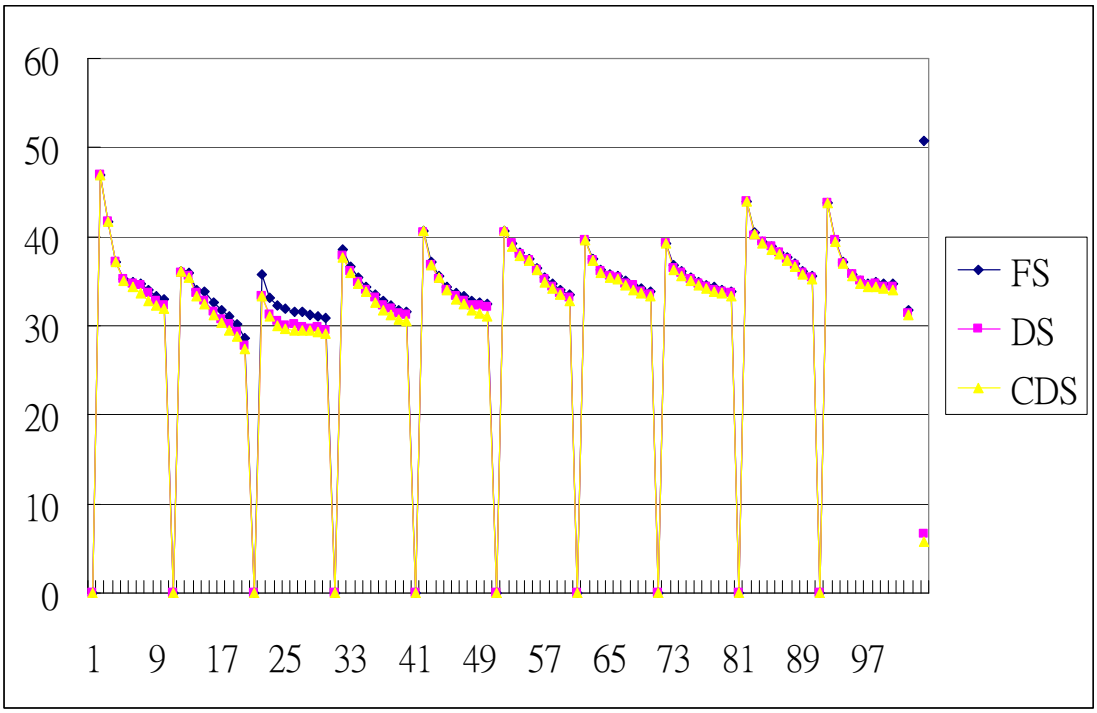
圖(1). Hexagon 與 Enhanced Hexagon PSNR 之比較圖
(搜尋視窗大小為±8)



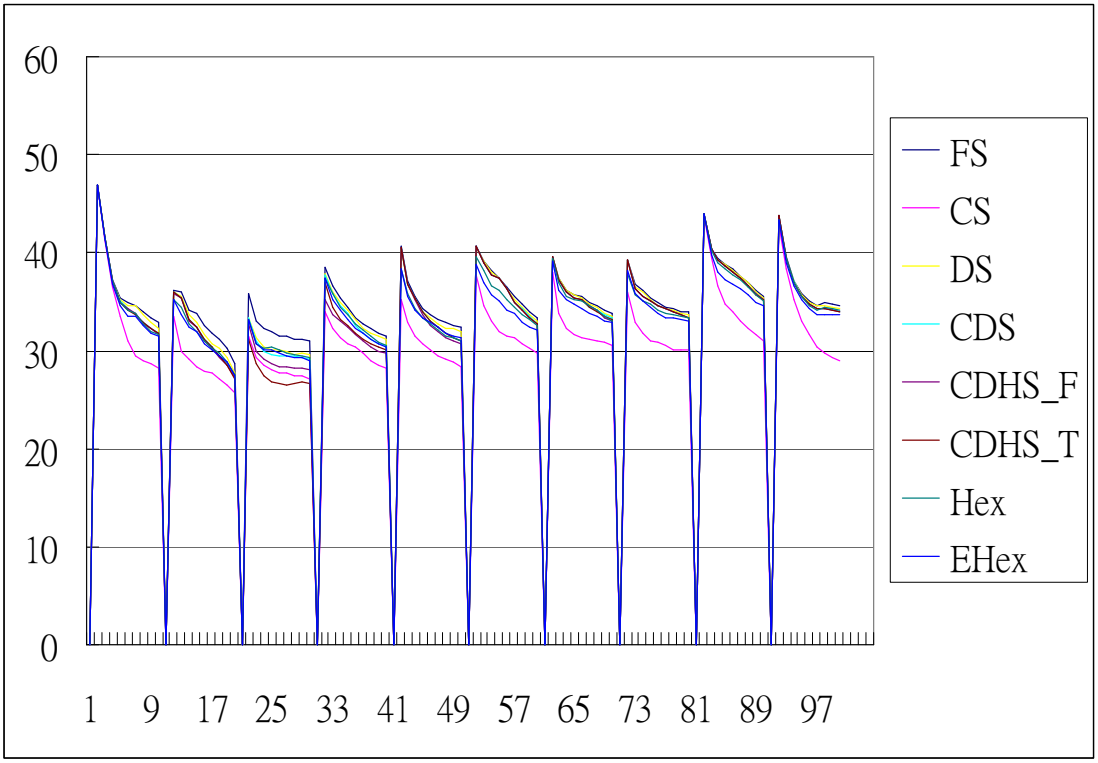
圖(2). CDHS_F 與 CDHS_T PSNR 之比較圖
(搜尋視窗大小為±8)



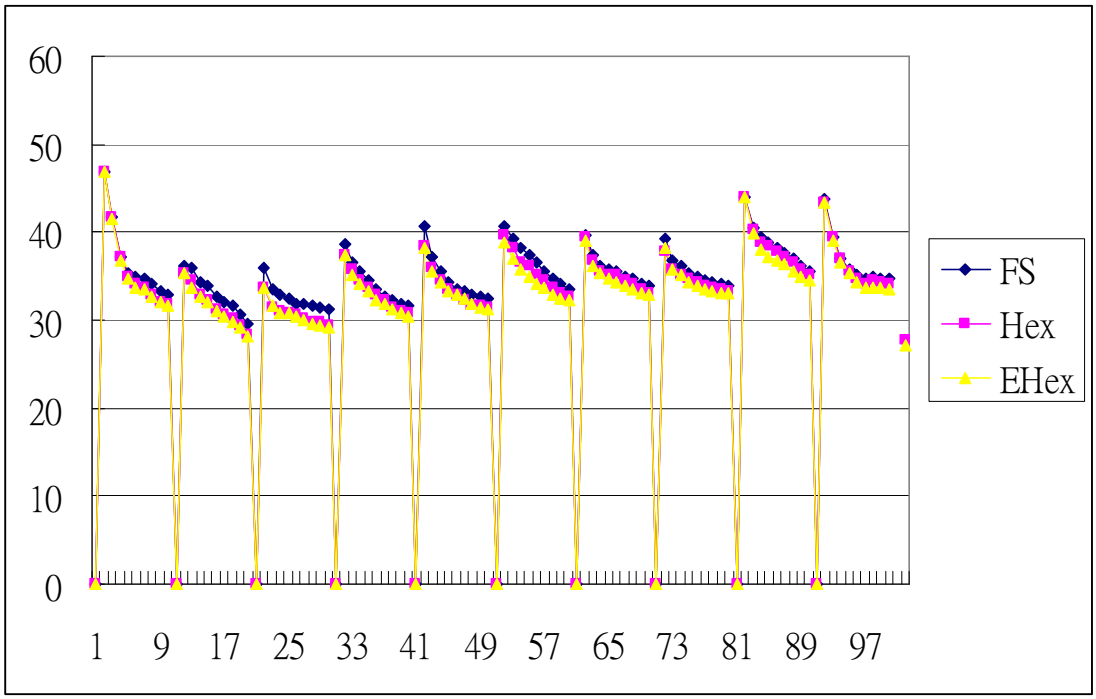
圖(3). CDS 與 CDHS_F PSNR 之比較圖
(搜尋視窗大小為±8)



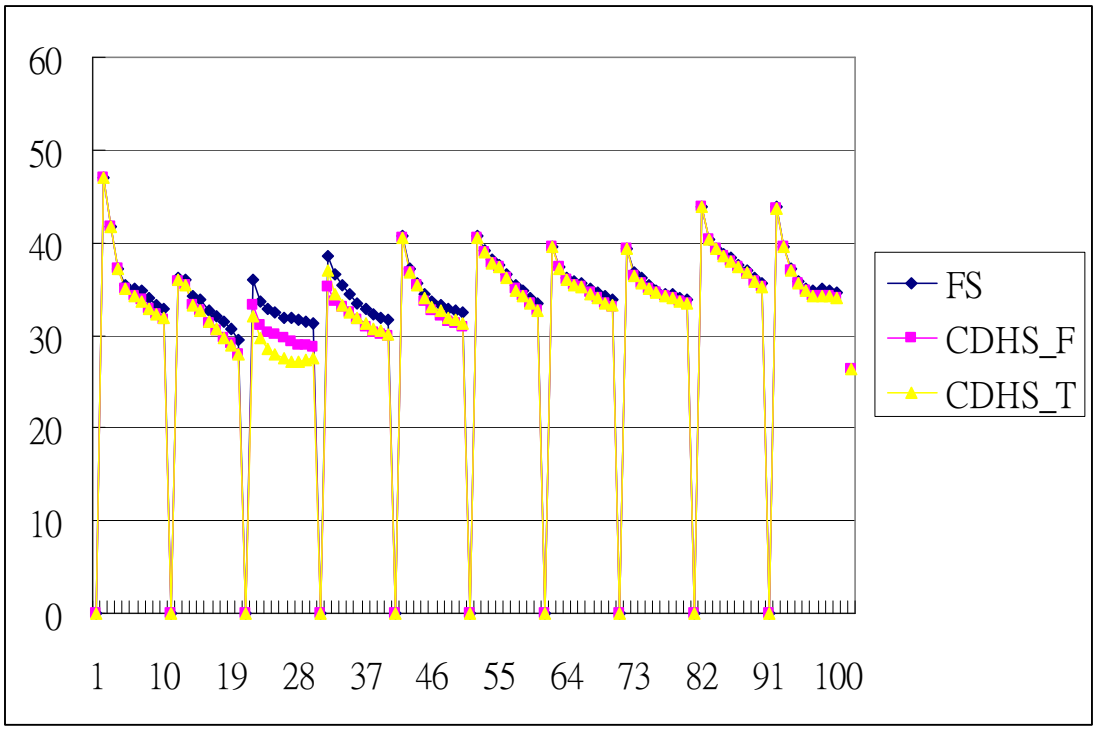
圖(4). CS 與 CDS PSNR 之比較圖
(搜尋視窗大小為±8)



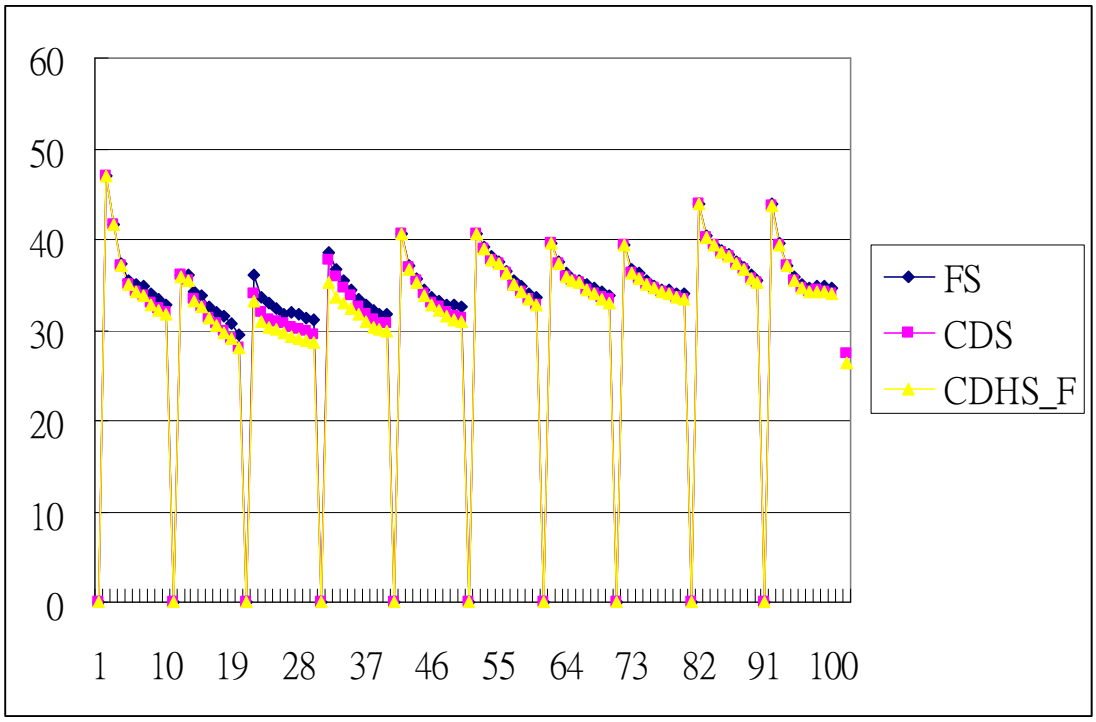
圖(5). 所有方法之比較圖 PSNR 之比較圖 (搜尋視窗大小為±8)



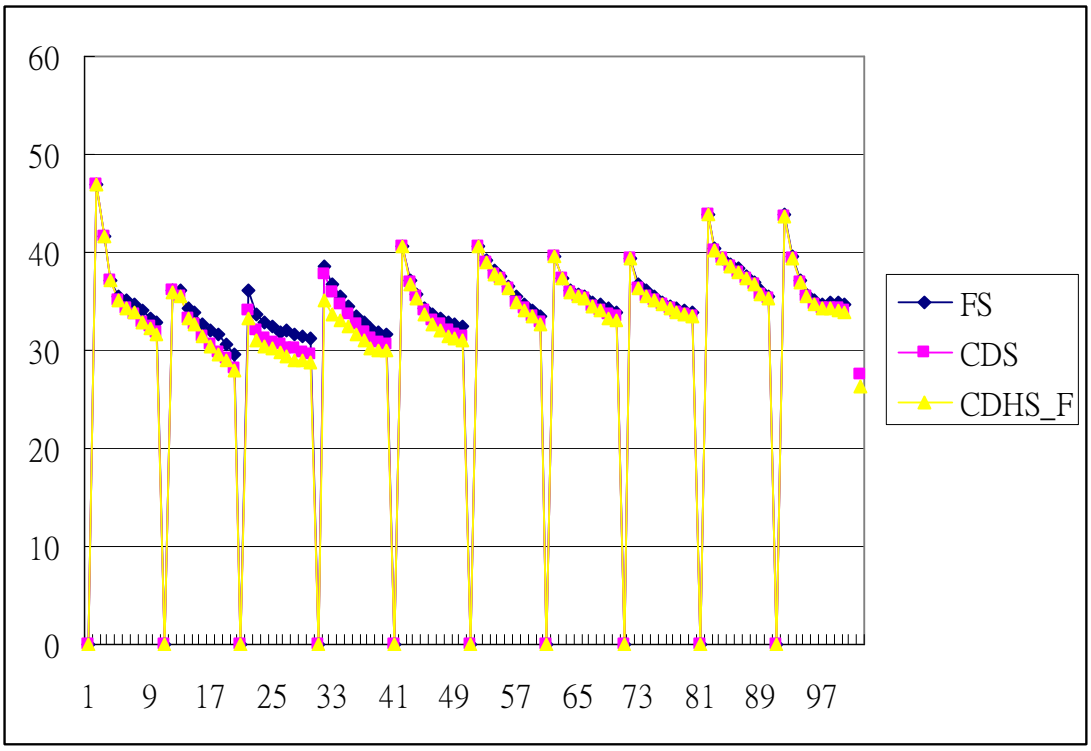
圖(6). Hexagon 與 Enhanced Hexagon PSNR 之比較圖
(搜尋視窗大小為±16)



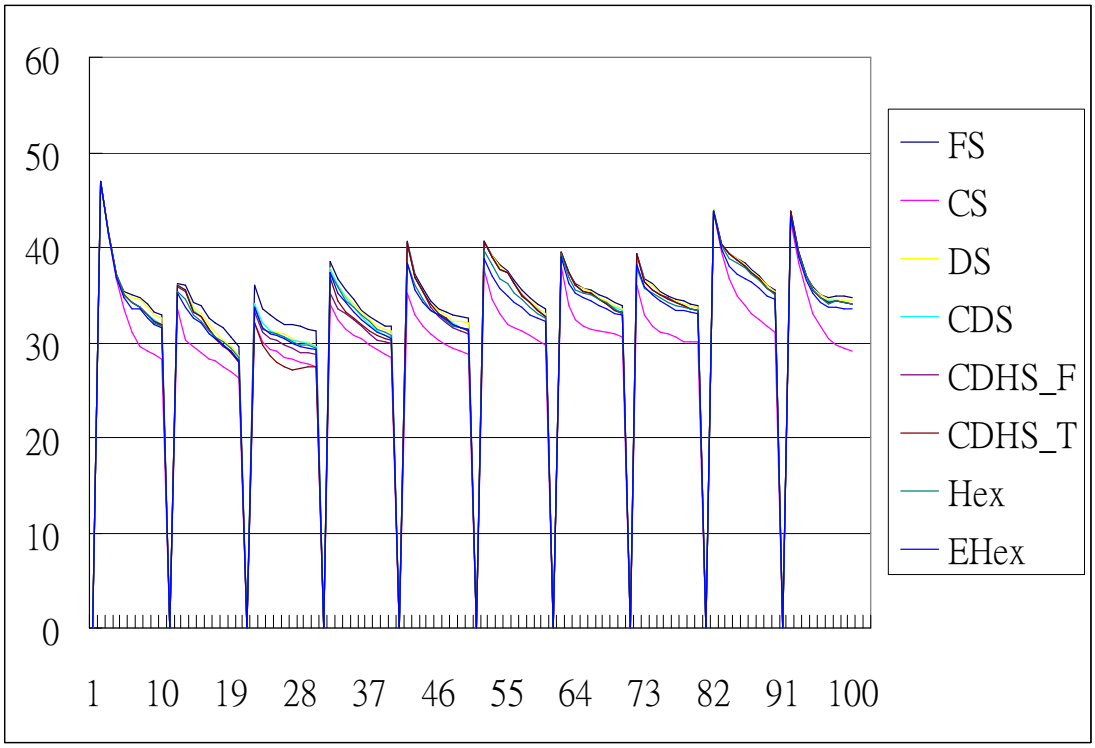
圖(7). CDHS_F 與 CDHS_T PSNR 之比較圖
(搜尋視窗大小為±16)



圖(8). CDS 與 CDHS_F PSNR 之比較圖
(搜尋視窗大小為±16)

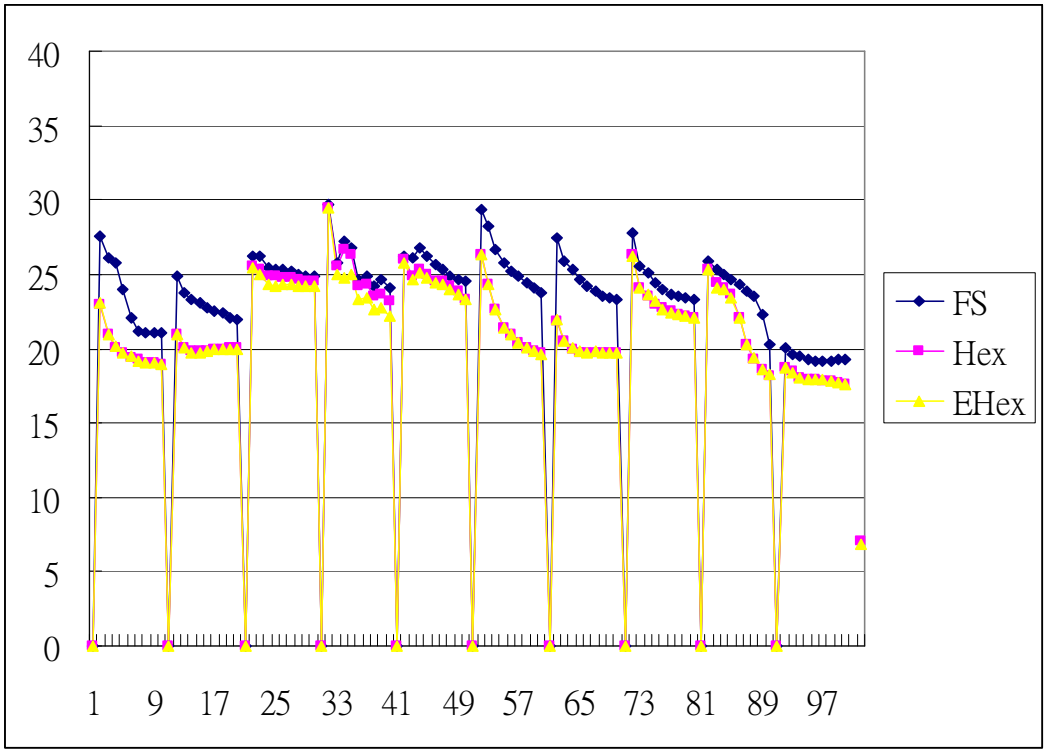


圖(9). CS 與 CDS PSNR 之比較圖
(搜尋視窗大小為±16)

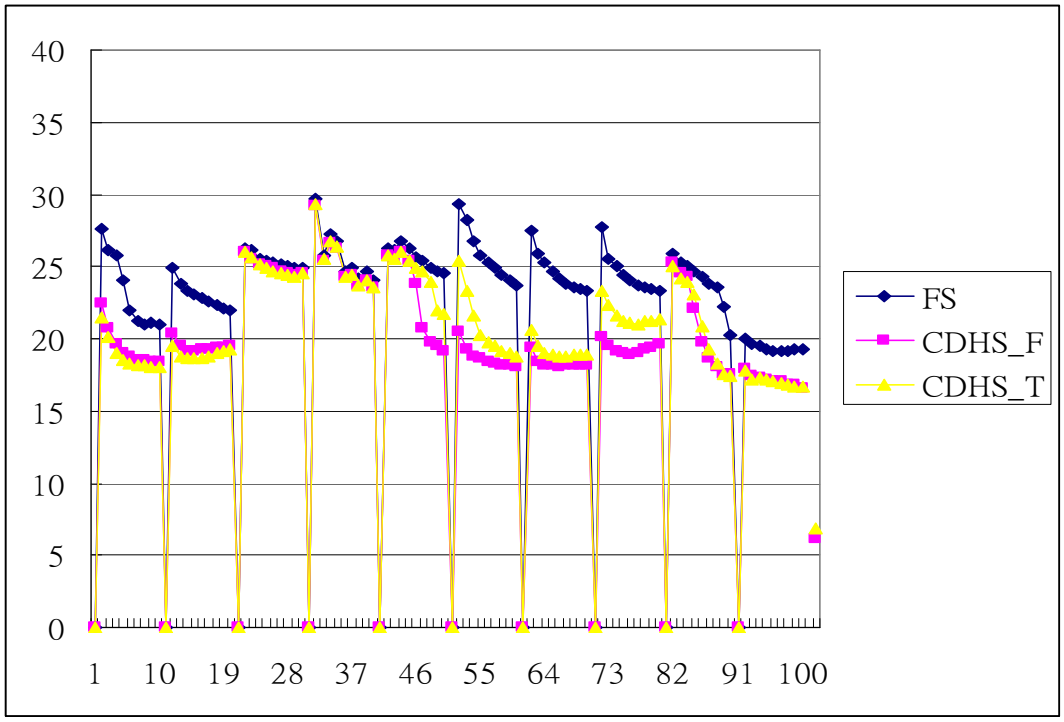


圖(10). 所有方法之比較圖 PSNR 之比較圖 (搜尋視窗大小為±16)

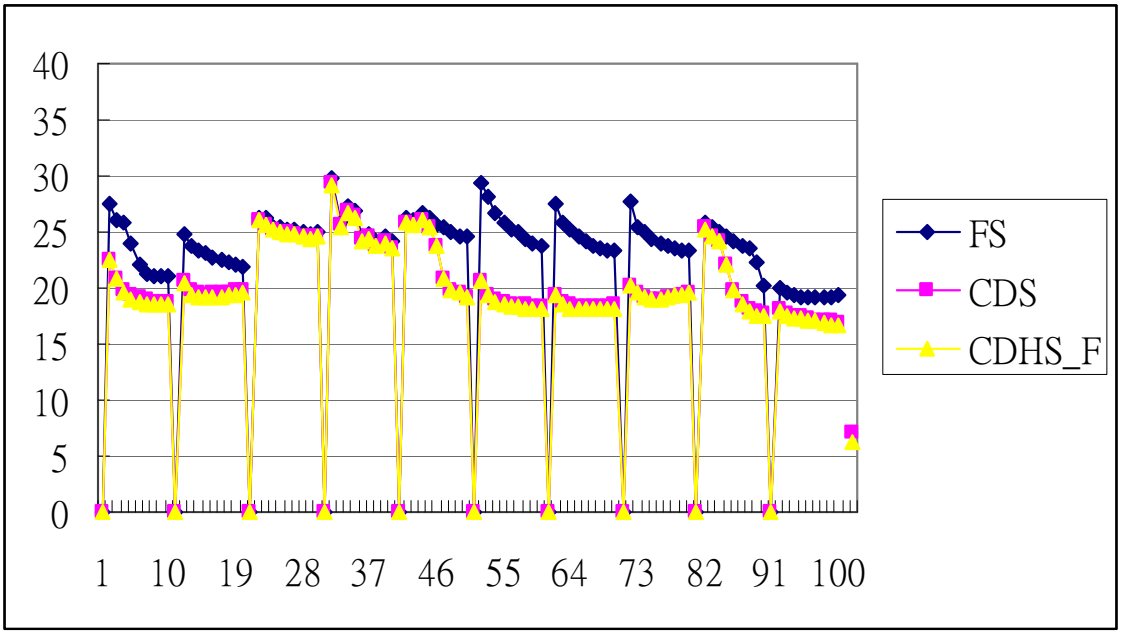
B. Sequence “Stefan”



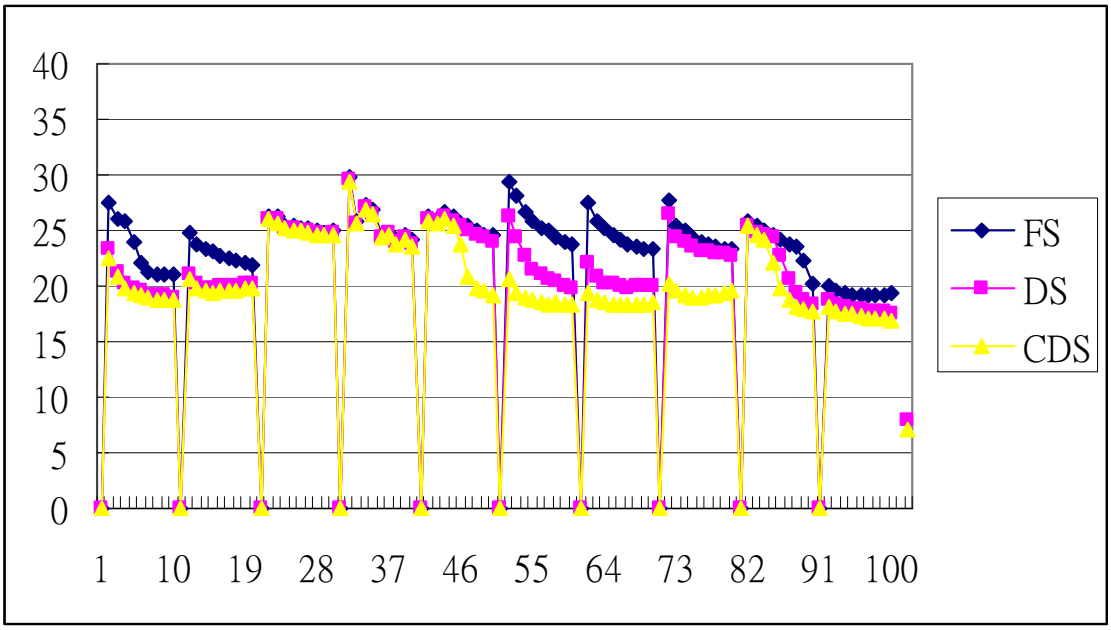
圖(11). Hexagon 與 Enhanced Hexagon PSNR 之比較圖
(搜尋視窗大小為±8)



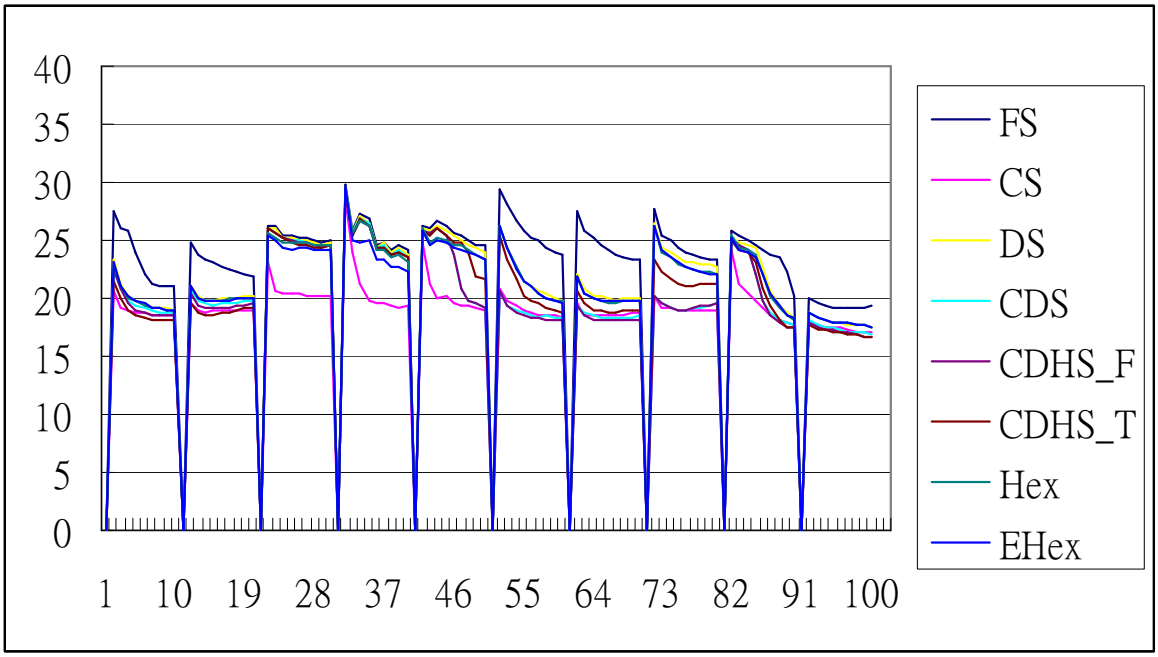
圖(12). CDHS_F 與 CDHS_T PSNR 之比較圖
(搜尋視窗大小為±8)



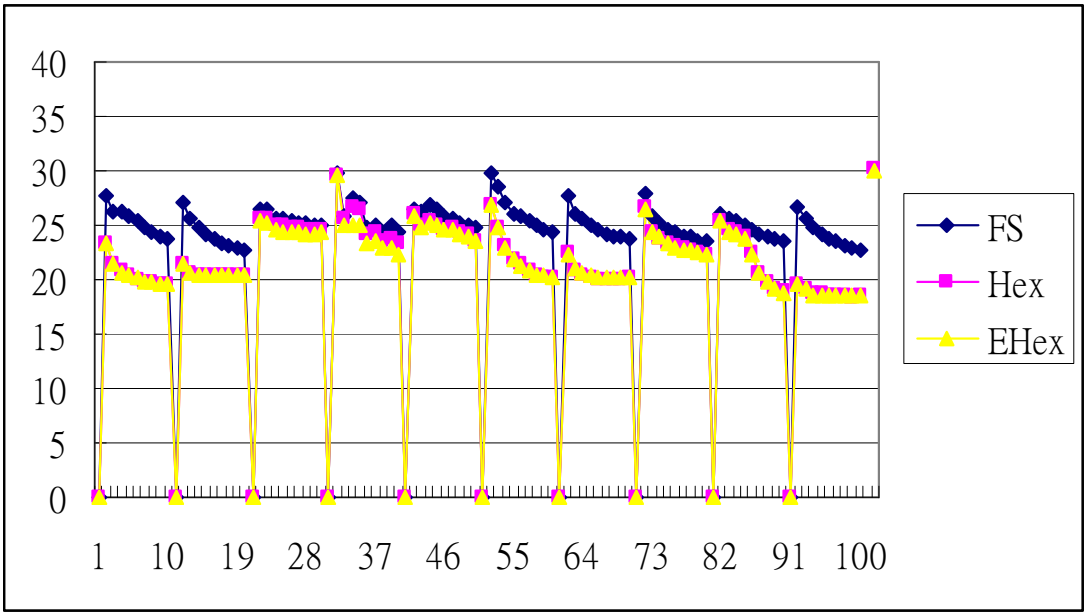
圖(13). CDS 與 CDHS_F PSNR 之比較圖
(搜尋視窗大小為±8)



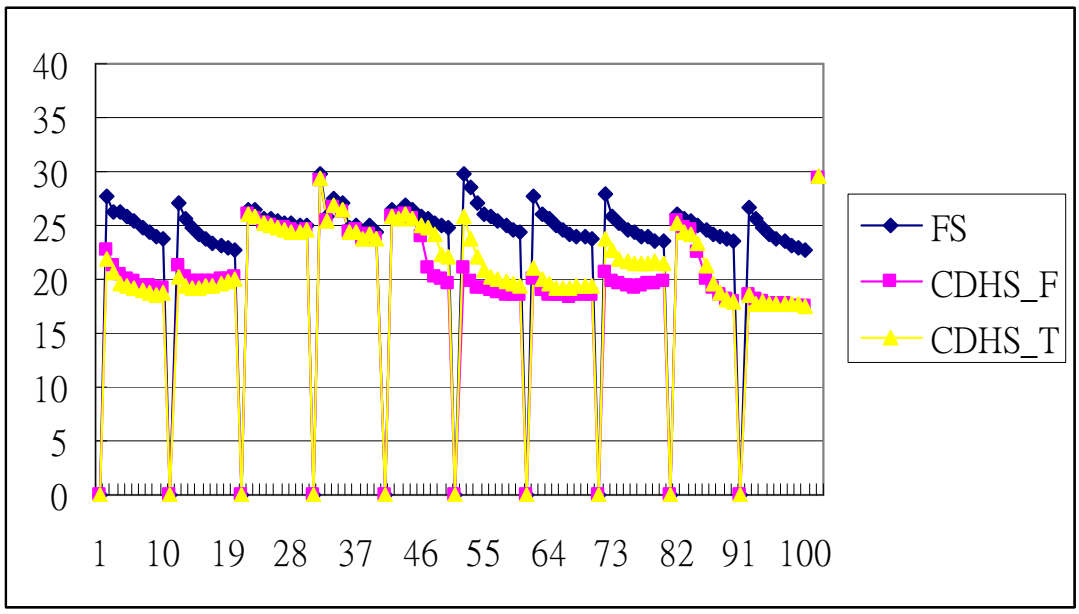
圖(14). CS 與 CDS PSNR 之比較圖
(搜尋視窗大小為±8)



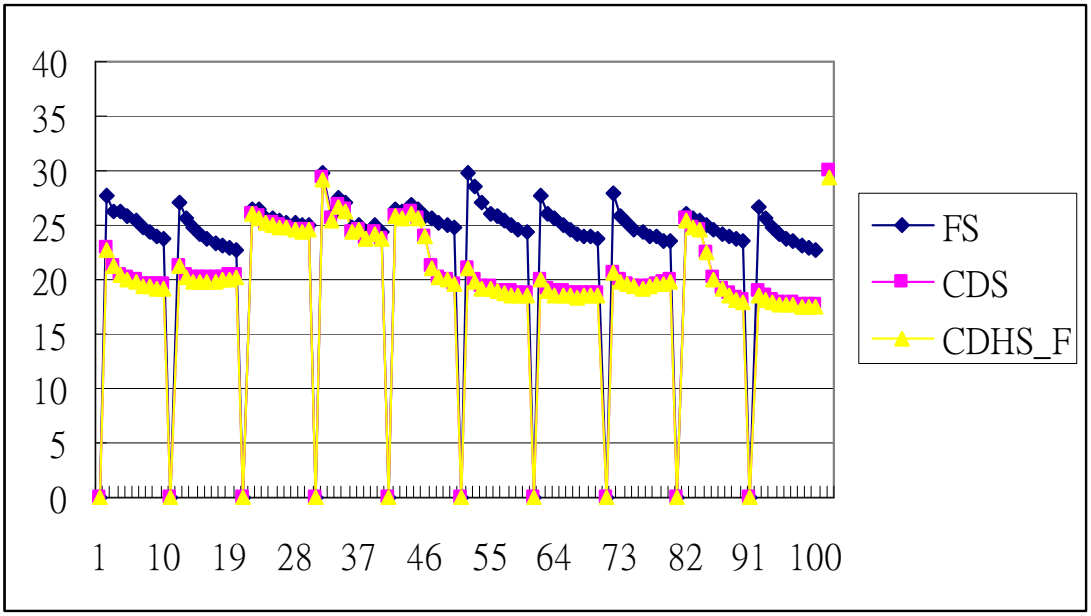
圖(15). 所有方法之比較圖 PSNR 之比較圖 (搜尋視窗大小為±8)



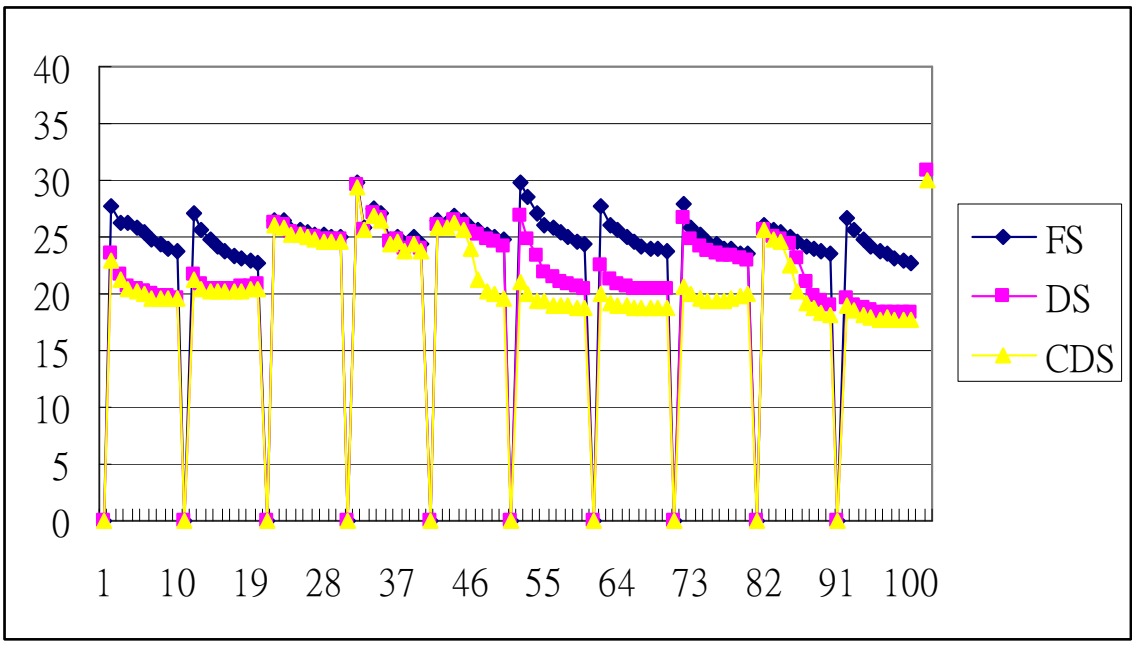
圖(16). Hexagon 與 Enhanced Hexagon PSNR 之比較圖
(搜尋視窗大小為±16)



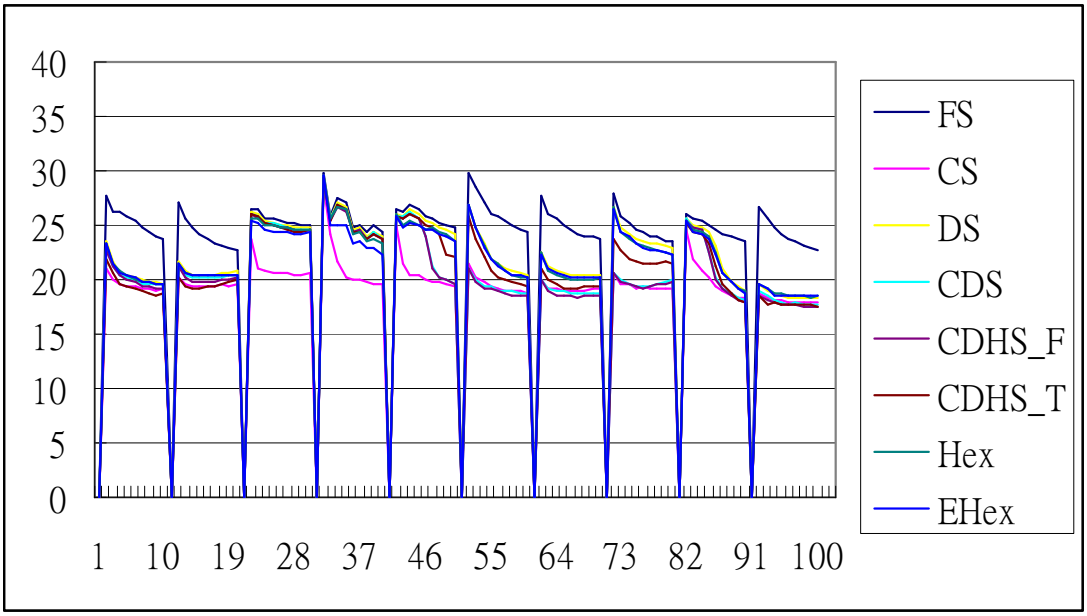
圖(17). CDHS_F 與 CDHS_T PSNR 之比較圖
(搜尋視窗大小為±16)



圖(18). CDS 與 CDHS_F PSNR 之比較圖
(搜尋視窗大小為±16)



圖(19). CS 與 CDS PSNR 之比較圖
(搜尋視窗大小為±16)



圖(20). 所有方法之比較圖 PSNR 之比較圖 (搜尋視窗大小為±16)

執行時間

	FS	CS	DS	CDS	CDHS_T	CDHS_F	Hex	EHex
Weather	50.766	6.594	6.562	5.782	4.781	4.969	6.125	6.156

表(1). 程式執行時間(sec)之比較 (搜尋視窗大小±8)

	FS	CS	DS	CDS	CDHS_T	CDHS_F	Hex	EHex
Weather	191.078	27.875	27.812	27.516	26.344	26.438	27.688	27.219

表(2). 程式執行時間(sec)之比較 (搜尋視窗大小±16)

	FS	CS	DS	CDS	CDHS_T	CDHS_F	Hex	EHex
Stefan	51.782	7.235	7.985	7.141	6.172	6.828	7.094	6.875

表(3). 程式執行時間(sec)之比較 (搜尋視窗大小±8)

	FS	CS	DS	CDS	CDHS_T	CDHS_F	Hex	EHex
Stefan	192.391	30.438	30.922	30.078	29.407	29.687	30.25	30.063

表(4). 程式執行時間(sec)之比較 (搜尋視窗大小±16)

平均 PSNR

	FS	CS	DS	CDS	CDHS_T	CDHS_F	Hex	EHex
Weather	35.3524	31.63311	34.93088	34.66835	34.37829	34.29954	34.43259	34.06385

表(5). 平均 PSNR 之比較 (搜尋視窗大小±8)

	FS	CS	DS	CDS	CDHS_T	CDHS_F	Hex	EHex
Weather	35.4414	31.78244	35.0297	34.80243	34.52223	34.4356	34.53392	34.15724

表(6). 平均 PSNR 之比較 (搜尋視窗大小±16)

	FS	CS	DS	CDS	CDHS_T	CDHS_F	Hex	EHex
Stefan	24.12667	19.40945	22.21025	20.78826	20.6278	21.21114	21.94738	21.78688

表(7). 平均 PSNR 之比較 (搜尋視窗大小±8)

	FS	CS	DS	CDS	CDHS_T	CDHS_F	Hex	EHex
Stefan	22.52643	17.7317	20.19005	18.95386	18.81056	19.32651	19.96612	19.82815

表(8). 平均 PSNR 之比較 (搜尋視窗大小±16)

VI. Conclusion

此次我們總共實做了八的方法，由最後的實驗數據可發現這八個方法其平均的 PSNR 比較起來，當然是 Full Search 最高，然而除了 Cross Search 外其他幾個改進的方法其平均的 PSNR 與 Full Search 相差並不多，實驗中我們使用了兩個 Sequence: Weather 以及 Stefan，Weather 是個蠻靜態的 Sequence，而 Stefan 是個動作差異很大的 Sequence，當 Stefan 只用 Full Search 時其 PSNR 也只剩下 24.12667。由實驗數據觀察到下列幾點:

- Weather 的 Sequence 當 Search Range 由 8 提升到 16，其平均的 PSNR 提高，但 Stefan 卻是相反，當 Search Range 由 8 提升到 16，其平均的 PSNR 卻下降。
- Diamond Search 其平均 PSNR 優於其他各個演算法，除了 Full Search 外。但其執行時間屬中等，然而六角形演算法其執行速度較快，而 PSNR 較 Diamond 低一點，所以兩種演算法普遍被各標準採用。
- 由於 Stefan 的動態較多，所以其執行時間稍比 Weather 多了一點。
- 因為動態估計都是根據前一個畫面來預測，所以當使用的 P-frame 距離 I-frame 較遠時，其 PSNR 逐漸下降。然後當 Sequence 的動態較大時，其 PSNR 下降更快

由於我們選定了一個 I-frame 搭配九個 P-frame 的緣故，所以每當到後面幾個 P-frame 其畫質都變很差，所以如何選擇 I、P-frame 的數目是很重要的，因為 I-frame 其壓縮比 P-frame 差，如何在效能與品質達成平衡，相同地動態估計就是在其 PSNR 與其執行複雜度取決達成平衡。

VII. Reference

[1] 戴顯權, “資料壓縮,” 第二版

[2] M. Ghanbari, “The Cross-Search Algorithm for Motion Estimation,” IEEE Trans. Communications, VOL. 38, NO. 1, JULY 1990

[3] Shan Zhu, and Kai-Kuang Ma, “A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation,” IEEE Trans. Image Process, VOL. 9, NO. 2, FEB 2000

[4] Ce Zhu, Xiao Lin, and Lap-Pui Chau, “Hexagon-Based Search Pattern for Fast Block Motion Estimation,” IEEE Trans. Circuits and Systems for Video Technology, VOL.12, NO.5, MAY 2002

[5] Chun-Ho Cheung, and Lai-Man Po, “A Novel Cross-Diamond Search Algorithm for Fast Block Motion Estimation,” IEEE Trans. Circuits and Systems for Video Technology, VOL.12, NO.12, DEC 2002

[6] Ce Zhu, Xiao Lin, and Lap-Pui Chau, Lai-Man Po, “Enhanced Hexagonal Search for Fast Block Motion Estimation,” IEEE Trans. Circuits and Systems for Video Technology, VOL.14, NO.10, OCTOBER 2004

[7] Chun-Ho Cheung, and Lai-Man Po, “Novel Cross-Diamond-Hexagonal Search Algorithms for Fast Block Motion Estimation,” IEEE Trans. Multimedia, VOL. 7, NO. 1, FEB 2005

VIII. Appendix

Demo 檔案列表:

(1) Test sequence: ”weather.y”

- ✓ Format :
  - Size: 352x288 pixels
  - Frame number: 100

Demo File List: (±8 search window)

BMA type	Demo file name	Differential file name
FS	out_weather_FS8.y	abs_diff_weather_FS8.y
CS	out_weather_CS8.y	abs_diff_weather_CS8.y
DS	out_weather_DS8.y	abs_diff_weather_DS8.y
CDS	out_weather_CDS8.y	abs_diff_weather_CDS8.y
CDHS_T	out_weather_CDHS_T8.y	abs_diff_weather_CDHS_T8.y
CDHS_F	out_weather_CDHS_F8.y	abs_diff_weather_CDHS_F8.y
HEXBS	out_weather_HEX8.y	abs_diff_weather_HEX8.y
EHEXBS	out_weather_EHEX8.y	abs_diff_weather_FS8.y

Demo File List: (±16 search window)

BMA type	Demo file name	Differential file name
FS	out_weather_FS16.y	abs_diff_weather_FS16.y
CS	out_weather_CS16.y	abs_diff_weather_CS16.y
DS	out_weather_DS16.y	abs_diff_weather_DS16.y
CDS	out_weather_CDS16.y	abs_diff_weather_CDS16.y
CDHS_T	out_weather_CDHS_T16.y	abs_diff_weather_CDHS_T16.y
CDHS_F	out_weather_CDHS_F16.y	abs_diff_weather_CDHS_F16.y
HEXBS	out_weather_HEX16.y	abs_diff_weather_HEX16.y
EHEXBS	out_weather_EHEX16.y	abs_diff_weather_FS16.y

(2) Test sequence: ”stefan.y”

- ✓ Format :
  - Size: 352x288 pixels
  - Frame number: 100

Demo File List: (± 8 search window)

BMA type	Demo file name	Differential file name
FS	out_stefan_FS8.y	abs_diff_stefan_FS8.y
CS	out_stefan_CS8.y	abs_diff_stefan_CS8.y
DS	out_stefan_DS8.y	abs_diff_stefan_DS8.y
CDS	out_stefan_CDS8.y	abs_diff_stefan_CDS8.y
CDHS_T	out_stefan_CDHS_T8.y	abs_diff_stefan_CDHS_T8.y
CDHS_F	out_stefan_CDHS_F8.y	abs_diff_stefan_CDHS_F8.y
HEXBS	out_stefan_HEX8.y	abs_diff_stefan_HEX8.y
EHEXBS	out_stefan_EHEX8.y	abs_diff_stefan_FS8.y

Demo File List: (± 16 search window)

BMA type	Demo file name	Differential file name
FS	out_stefan_FS16.y	abs_diff_stefan_FS16.y
CS	out_stefan_CS16.y	abs_diff_stefan_CS16.y
DS	out_stefan_DS16.y	abs_diff_stefan_DS16.y
CDS	out_stefan_CDS16.y	abs_diff_stefan_CDS16.y
CDHS_T	out_stefan_CDHS_T16.y	abs_diff_stefan_CDHS_T16.y
CDHS_F	out_stefan_CDHS_F16.y	abs_diff_stefan_CDHS_F16.y
HEXBS	out_stefan_HEX16.y	abs_diff_stefan_HEX16.y
EHEXBS	out_stefan_EHEX16.y	abs_diff_stefan_FS16.y

執行檔說明與列表:

- 執行目錄下需準備 (1) *.exe 執行檔。
  (2) Original test sequence。

(a) Full search algorithm

Exe file	Input file	Output file	Description
FS_S8.exe	stefan.y	out_stefan_FS8.y	With ±8 search range
		abs_diff_stefan_FS8.y	
FS_W8.exe	weather.y	out_weather_FS8.y	
		abs_diff_weather_FS8.y	
FS_S16.exe	stefan.y	out_stefan_FS16.y	With ±16 search range
		abs_diff_stefan_FS16.y	
FS_W16.exe	weather.y	out_weather_FS16.y	
		abs_diff_weather_FS16.y	

(b) Cross Search Algorithm

Exe file	Input file	Output file	Description
CS_S8.exe	stefan.y	out_stefan_CF.y	With ±8 search range
		abs_diff_stefan_CS8.y	
CS_W8.exe	weather.y	out_weather_CS8.y	
		abs_diff_weather_CS8.y	
CS_S16.exe	stefan.y	out_stefan_CS16.y	With ±16 search range
		abs_diff_stefan_CS16.y	
CS_W16.exe	weather.y	out_weather_CS16.y	
		abs_diff_weather_CS16.y	

(c) Diamond Search Algorithm

Exe file	Input file	Output file	Description
DS_S8.exe	stefan.y	out_stefan_DS8.y	With ±8 search range
		abs_diff_stefan_DS8.y	
DS_W8.exe	weather.y	out_weather_DS8.y	
		abs_diff_weather_DS8.y	
DS_S16.exe	stefan.y	out_stefan_DS16.y	With ±16 search range
		abs_diff_stefan_DS16.y	
DS_W16.exe	weather.y	out_weather_DS16.y	
		abs_diff_weather_DS16.y	

(d)Cross-Diamond Search Algorithm

Exe file	Input file	Output file	Description
CDS_S8.exe	stefan.y	out_stefan_CDS8.y	With ±8 search range
		abs_diff_stefan_CDS8.y	
CDS_W8.exe	weather.y	out_weather_CDS8.y	
		abs_diff_weather_CDS8.y	
CDS_S16.exe	stefan.y	out_stefan_CDS16.y	With ±16 search range
		abs_diff_stefan_CDS16.y	
CDS_W16.exe	weather.y	out_weather_CDS16.y	
		abs_diff_weather_CDS16.y	

(e)Cross-Diamond-Hexagonal Search with large hexagonal search pattern (CDHS-T) Algorithm

Exe file	Input file	Output file	Description
CDHS_T _S8.exe	stefan.y	out_stefan_CDHS_T8.y	With ±8 search range
		abs_diff_stefan_CDHS_T8.y	
CDHS_ T_W8.exe	weather.y	out_weather_CDHS_T8.y	
		abs_diff_weather_CDHS_T8.y	
CDHS_ T_S16.exe	stefan.y	out_stefan_CDHS_T16.y	With ±16 search range
		abs_diff_stefan_CDHS_T16.y	
CDHS_ T_W16.exe	weather.y	out_weather_CDHS_T16.y	
		abs_diff_weather_CDHS_T16.y	

(f)Cross-Diamond-Hexagonal Search with small hexagonal search pattern (CDHS-F) Algorithm

Exe file	Input file	Output file	Description
CDHS_F _S8.exe	stefan.y	out_stefan_CDHS_F8.y	With ±8 search range
		abs_diff_stefan_CDHS_F8.y	
CDHS_ F_W8.exe	weather.y	out_weather_CDHS_F8.y	
		abs_diff_weather_CDHS_F8.y	
CDHS_ F_S16.exe	stefan.y	out_stefan_CDHS_F16.y	With ±16 search range
		abs_diff_stefan_CDHS_T16.y	
CDHS_ F_W16.exe	weather.y	out_weather_CDHS_F16.y	
		abs_diff_weather_CDHS_F16.y	

(g)Hexagon-Based Search Algorithm

Exe file	Input file	Output file	Description
HEX_S8.exe	stefan.y	out_stefan_HEX8.y	With ±8 search range
		abs_diff_stefan_HEX8.y	
HEX_W8.exe	weather.y	out_weather_HEX8.y	
		abs_diff_weather_HEX8.y	
HEX_S16.exe	stefan.y	out_stefan_HEX16.y	With ±16 search range
		abs_diff_stefan_HEX16.y	
HEX_W16.exe	weather.y	out_weather_HEX16.y	
		abs_diff_weather_HEX16.y	

(h) Enhanced Hexagon-Based Search Algorithm

Exe file	Input file	Output file	Description
EHEX_S8.exe	stefan.y	out_stefan_EHEX8.y	With ±8 search range
		abs_diff_stefan_EHEX8.y	
EHEX_W8.exe	weather.y	out_weather_EHEX8.y	
		abs_diff_weather_EHEX8.y	
EHEX_S16.exe	stefan.y	out_stefan_EHEX16.y	With ±16 search range
		abs_diff_stefan_EHEX16.y	
EHEX_W16.exe	weather.y	out_weather_EHEX16.y	
		abs_diff_weather_EHEX16.y	