Journal of Zhejiang University SCIENCE A
ISSN 1673-565X (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

A novel adjustable multiple cross-hexagonal search algorithm for fast block motion estimation

XIE Chun-lai[†1], CHEUNG Chun-ho[†2], LIU Wei-zhong[1]

([1]*Department of Electronic Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*)

([2]*Hong Kong Institute of Technology, Hong Kong, China*)

[†]E-mail: hust_xcl@126.com; terencecheung@hkit.edu.hk

Received Sept. 7, 2006; revision accepted Feb. 7, 2007

Abstract: In this paper, we propose a novel adjustable multiple cross-hexagonal search (AMCHS) algorithm for fast block motion estimation. It employs adjustable multiple cross search patterns (AMCSP) in the first step and then uses half-way-skip and half-way-stop technique to determine whether to employ two hexagonal search patterns (HSPs) subsequently. The AMCSP can be used to find small motion vectors efficiently while the HSPs can be used to find large ones accurately to ensure prediction quality. Simulation results showed that our proposed AMCHS achieves faster search speed, and provides better distortion performance than other popular fast search algorithms, such as CDS and CDHS.

Key words: Motion estimation, Fast search algorithm, Adjustable search patterns, Threshold strategy, Hexagonal search pattern
doi:10.1631/jzus.2007.A1304 **Document code:** A **CLC number:** TP202

INTRODUCTION

Block motion estimation is widely used in many motion-compensated video coding standards, such as ISO MPEG-1/2/4, H.263 and H.264, to exploit high temporal redundancy in the successive frames. Video frames are first divided into equally sized blocks. When coding the current block, a matched block in the previous frames is used for prediction and then only the difference between these two blocks is coded. Generally speaking, the less the difference, the fewer the bits needed. Hence, finding the best matched block is very important for improving the compression efficiency. The full search (FS) algorithm is the most accurate one to find the motion vectors by evaluating all the possible candidate points, but it costs too much computational resource. In order to replace FS, many low-complexity and fast-search algorithms are proposed, such as the three-step search (TSS), the new three-step search (NTSS), the four-step search (4SS) (Po and Ma, 1996), the diamond search (DS) (Tham *et al*., 1998; Zhu and Ma, 2000), the hexagon based search (HEXBS) (Zhu *et al*., 2002; 2004), the cross-diamond search (CDS) (Cheung and Po, 2002), and the cross-diamond-hexagonal search (CDHS) (Cheung and Po, 2005) algorithms.

Many fast algorithms such as NTSS, DS, and HEXBS employ one or more large search patterns for larger scope search before smaller enhanced search. As the global center-biased motion vector distribution characteristic, most motion vectors are located near the zero vector (Cheung and Po, 2002). Using large search patterns will spend computation for more points, especially for low-motion sequences. A popular approach to this problem is to change the order of the large and small search patterns in the search process and employing a half-way-stop technique to determine whether to use the larger ones, such as CDS and CDHS. Another possible solution is to use motion vector prediction to determine the size of the search patterns.

Among the fast search algorithms mentioned above, NTSS, 4SS, DS, HEXBS and CDS get good

distortion performance but they use more search points. CDHS gets the fastest search speed among all the block-matching algorithms (BMAs), but its distortion performance decreases especially for sequences that have large motion contents. It seems to be difficult to combine both distortion and speed performances. In this paper, we proposed a novel adjustable multiple cross-hexagonal search (AMCHS) algorithm, which uses an adjustable multiple cross and two hexagonal search patterns in the search process. Simulation results showed that the speed of AMCHS is comparable to that of CDHS and its distortion performance is similar to that of DS.

The rest of the paper is organized as follows. Section 2 indicates the limitation of the small cross search pattern (SCSP). Section 3 introduces the adjustable multiple cross search pattern and the hexagonal search patterns used in AMCHS. Section 4 describes the use of Adaptation Model C_L. Section 5 describes the AMCHS algorithm. Section 6 reports some significant simulation results on AMCHS and conclusions are given in Section 7.

LIMITATION OF SCSP

SCSP is the simplest search pattern in the discrete search grid and has been employed by many fast BMAs. However, it has limitation when being used to find large motion vectors. Here we will compare SCSP (Fig.1a) with the large cross search pattern (LCSP) (Fig.1b) in terms of finding motion vectors in one step only. Four typical examples by using these two cross search patterns, which are obtained from simulation with "Football" sequence, are shown in Fig.2. It is clear that the search process depicted in Fig.2a has dropped into local optima while the one depicted in Fig.2b gets higher matching quality. Similar situations may often happen if the motion vector appears to be large, though mostly we will get the same motion by using these two cross search patterns, as shown in Fig.2c and Fig.2d, especially for the low-motion sequences because of the center-based distributions (Tham *et al.*, 1998; Cheung and Po, 2002). In conclusion, SCSP cannot be used to find motion vectors accurately, especially for high-motion content.

Since SCSP has limitation of low accuracy accompanied with the advantage of simplicity, can we improve it to get a more effective search pattern? Based on SCSP, we propose an adjustable multiple cross search pattern (AMCSP) in this paper.

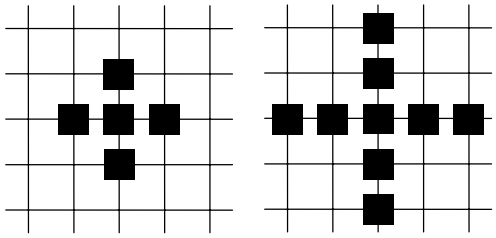


Fig.1 (a) Small cross search pattern; (b) Large cross search pattern

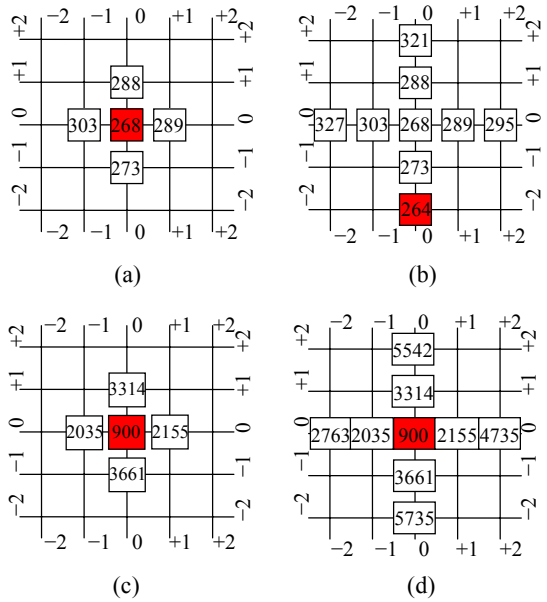


Fig.2 Four typical search processes by using small (a, c) and large (b, d) cross search patterns. Search points are labelled with the values of block distortion measure

SEARCH PATTERNS USED IN AMCHS

Adjustable multiple cross search pattern

As mentioned above, low accuracy of the motion vectors using SCSP leads to bad distortion performance. If we can find a way to judge whether the motion vector is accurate, the problem can be solved effectively. With reference to Fig.2c, only the block distortion measure (BDM) at point (0, 0), 900, is the smallest. In this case, even if we add 4 ending points

of LCSP to the search process, as shown in Fig.2d, the current minimum point (CMP) remains at the same place, i.e. (0, 0). Thus, we can think that the CMP derived by using SCSP in Fig.2c is accurate. Then let us turn to the search process depicted in Fig.2a. Though point (0, 0) wins minimum BDM, 268, points (0, −1) and (0, 1) get similar BDMs, 273 and 288. If we use LCSP to add some search points, we may get better result, as shown in Fig.2b. It appears that the accuracy of the motion vector found by using SCSP is related to the BDMs of the searched points. If the outer searched points have much larger BDMs than the CMP's BDM (CMBDM), the current motion vector (CMV) is accurate enough; otherwise, it is not reliable and more search points should be evaluated to enhance the search intensity.

Here we propose a new search pattern, AMCSP, which is expected to be used for finding motion vectors more accurately than SCSP. Its shape can be adjusted to fit the accuracy of the CMV. At first, we set a threshold BDM (TBDM), which is used to judge the accuracy of CMV. Adaptive threshold strategy aiming to get a fixed search performance can be found in (Sorwar *et al*., 2003; 2005). The TBDM used in our strategy is also adaptive, which is linear to CMBDM. The TBDM is defined as

$$TBDM = CMBDM \times C_{\mathrm{L}}, \tag{1}$$

where C_{L} is the control parameter, which is a fixed value to every frame. In the search process, if one of the outer searched points has BDM less than TBDM, AMCSP will adjust its shape at the extending point (EP). Here we first assume C_{L} as 1.1. Fig.3 shows two typical shapes of AMCSP. Fig.3a has EP while Fig.3b does not. We can find the minimum EP is the one with BDM 288 as depicted in Fig.3a among the four outer points with BDMs 288, 289, 273 and 303. Note that EP may be CMP if CMP is one of the outer searched points (this happens when the position of CMP changes).

Finding EP is the key to adjust the shape of AMCSP. However it seems difficult to get it. Do we need to record all the searched points and choose EP from them? In practice we need record only several searched points that have the least BDMs. EP is the outer minimum point, which usually belongs to these recorded points (RPs) that have the least BDMs.

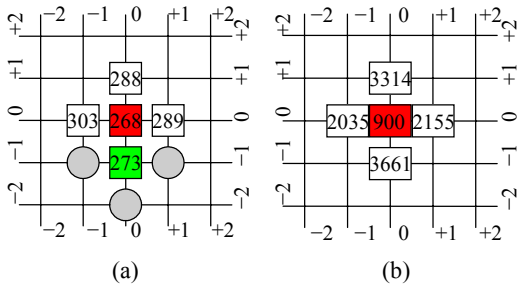


Fig.3 Two typical shapes of adjustable multiple cross search patterns when C_{L}=1.1. (a) *TBDM*=299, *EBDM*=273; (b) *TBDM*=990

In our algorithm, we record three RPs. Our experiments showed that the increase of distortion performance can be ignored while updating more RPs (more than three) costs too much computation. The minimum among these three RPs will be selected as the EP. It must satisfy two conditions: (1) its BDM is smaller than TBDM; (2) it is an outer but not a fully surrounded searched point. The choosing process is realized as follows:

```
for (i=1; i≤3; i++)
 if (RP[i].BDM<TBDM) && (!RP[i].State)
 { RP[i].State=1; EP=RP[i];
   Adjust the Shape and go to the Search;
   break;
 }
```

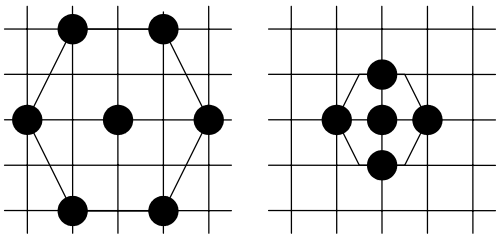
where binary *RP*[*i*].State indicates whether *RP*[*i*] has been used as EP. "0" indicates not and "1" indicates it has been used as EP and is fully surrounded by other searched points. Before the search process, the *RP*.States are all initialized as 0. Note that the RPs are sorted in ascending order of BDM.

If we use AMCSP depicted in Fig.3a and Fig.3b to replace LCSP depicted in Fig.2b and Fig.2d respectively, we can get the same or better CMP with fewer search points [The points (−1, −1) and (1, −1) in Fig.3a will be evaluated. They may have smaller BDMs than 264 in Fig.2b]. Note that AMCSP has many shapes, not only the two depicted in Fig.3.

Hexagonal search patterns

Because AMCSP is not suitable to be used to find large motion vectors, we must find some other search pattern(s) to solve the problem. Among all the fast BMAs, HEXBS has shown robustness in finding large motion vectors efficiently. The hexagonal search patterns, which are depicted in Fig.4, can be

used to find all motion vectors without limitation of the search range and the shape. The large hexagonal search pattern (LHSP) has good symmetry and can cover a large search scope while the small hexagonal search pattern (SHSP) can be used to enhance the search. These two search patterns are also employed in our proposed AMCHS to work together with AMCSP.



(a) (b)

Fig.4 Two hexagonal search patterns. (a) Large hexagonal search pattern; (b) Small hexagonal search pattern

ADAPTATION MODEL OF C_L

TBDM has a significant impact on the search process and is a function of C_L; hence, choosing a suitable C_L is important for good search performance. An adaptation model of C_L based on normalized block least mean square (NBLMS) adaptive algorithm is proposed in (Sorwar *et al*., 2003; 2005) and shows good performance to get desired search speeds or distortion measures. Similar model is also adopted in this paper. It has three modules:

(1) Motion estimation. The input of this module is comprised of the video frame pair, $x^{[m]}$, and C_L, where m is the frame number. The output of this module is the average BDM per pixel. It can be expressed as

$$y^{[m]} = f_1(x^{[m]}, C_L), \tag{2}$$

where f_1 is a deceasing function of C_L (under stationary x).

(2) Performance conclusion. Because of the high correlation between the video frames, we use the first $m-1$ frames' outputs to predict $y^{[m]}$. The error signal is:

$$e^{[m]} = \frac{1}{m-1}\sum_{i=0}^{m-1} y^{[i]} - y^{[m]}. \tag{3}$$

The value of e must be minimized as the adaptation process progresses.

(3) Adaptation of C_L. This module updates the value of C_L for the next frame:

$$C_L^{[m+1]} = C_L^{[m]} - f_2(e^{[m]}, y^{[m]}), \tag{4}$$

where f_2 is a linear or nonlinear function.

The well-known NBLMS adaptive algorithm is used to define the function f_2. For a frame length of K, the input signal can be considered stable. The value of C_L is updated like this:

$$C_L^{[m+K]} = C_L^{[m]} - e^{[m]} \frac{1}{K}\sum_{i=0}^{K-1} y^{[m+i]} \Big/ \sum_{i=0}^{K-1} (y^{[m+i]})^2. \tag{5}$$

Here we bound C_L with [1.05, 1.30]. Our experiments showed that, when C_L is smaller than 1.05, the distortion performance is affected very much. When C_L is larger than 1.30, the decrease of the BDM values can be ignored while the number of needed search points still increases.

PROPOSED ADJUSTABLE MULTIPLE CROSS-HEXAGONAL SEARCH ALGORITHM

Motion estimation for one block

At first, we employ the simple search pattern AMCSP to fit the characteristic of center-biased motion vector distributions. AMCSP is designed to find small motion vectors, and we add a limitation to the adjustable multiple cross searching process. If CMP moves away from the search center by more than one pixel, the search advances to the next step. Then we adopt half-way-stop and half-way-skip techniques to determine whether to employ hexagonal search patterns. That is useful for saving lots of search points especially for quasi-stationary or stationary blocks.

The whole search process for one block is summarized as follows.

Step 1: Initialization

Place AMCSP at the search center, (0, 0). The initial shape of AMCSP is initialized like that of SCSP.

Step 2: Adjustable multiple cross searching

The candidate points on AMCSP are evaluated.

If CMP does not occur at (0, 0), (0, 1), (−1, 0), (0,

−1), or (1, 0), the search proceeds to Step 3.

If an EP is available, the shape of AMCSP is adjusted and this step is repeated; otherwise, the search stops (this is called half-way-stop as shown in Fig.5a).

Step 3: Half-hexagonal searching

If CMP locates at (x, 0), three points (x±2, 0) ("±" is the same with the sign of x. If x>0, "±" is "+"; otherwise, "±" is "−"), (x, 2) and (x, −2) are evaluated (Step 3 in Fig.5c).

If CMP locates at (0, y), three points (2, y), (−2, y) and (0, y±2) are evaluated.

If CMP locates at (x, y), three points (x±2, y), (x±2, y±2), (x, y±2) are evaluated (Step 3 in Fig.5b).

If CMP still locates at the same place, the search goes to Step 5 (this is called half-way-skip as shown in Fig.5b).

Step 4: Large hexagonal searching

An LHSP is formed by positioning the CMP found in the previous step as the LHSP center. If CMP takes the same place, the search proceeds to Step 5; otherwise, this step is repeated.

Step 5: Ending

With CMP found in the previous step as the center, an SHSP is formed. If CMP still locates at the center, the search stops; otherwise, this step is repeated.

Three typical search paths by using AMCHS are shown in Fig.5.

Proposed AMCHS algorithm

The NBLMS algorithm is initialized by AMCHS to adapt the control parameter C_L. For every K frames of the video sequence, the same C_L is used for motion estimation. C_L is initialized to 1.05 for the first $2K$ frames and then updated for the next K frames by Eq.(5). The complete AMCHS algorithm is summarized as follows:

Precondition: Input video sequences of N frames. Set the "frame length" to K.

Initialization:

$C_L^{[1]}=C_L^{[2]}=\ldots=C_L^{[2K]}=1.05$.

Body:

for m=1, 1+K, …, 1+[(N−1)/K]×K

(1) K-frame motion estimations:

S=V=0;

for i=0, 1, …, min(K−1, N−m−1)

Calculate motion vectors for frame m+i using the algorithm described in Section 4 and let $y^{[m+i]}$ be the BDM.

$$S+=y^{[m+i]};\quad V+=(y^{[m+i]})^2.$$

(2) Performance measurement:

$$e^{[m]}=\frac{1}{m-1}\sum_{i=0}^{m-1}y^{[i]}-\frac{S}{K}.$$

(3) Adaptation of C_L:

$$C_L^{[m+K]}=C_L^{[m+K+1]}=\ldots=C_L^{[m+K+\min(K-1,\,N-m-1)]}=C_L^{[m]}+e^{[m]}S/(KV).$$

Post conditions: Motion vectors.

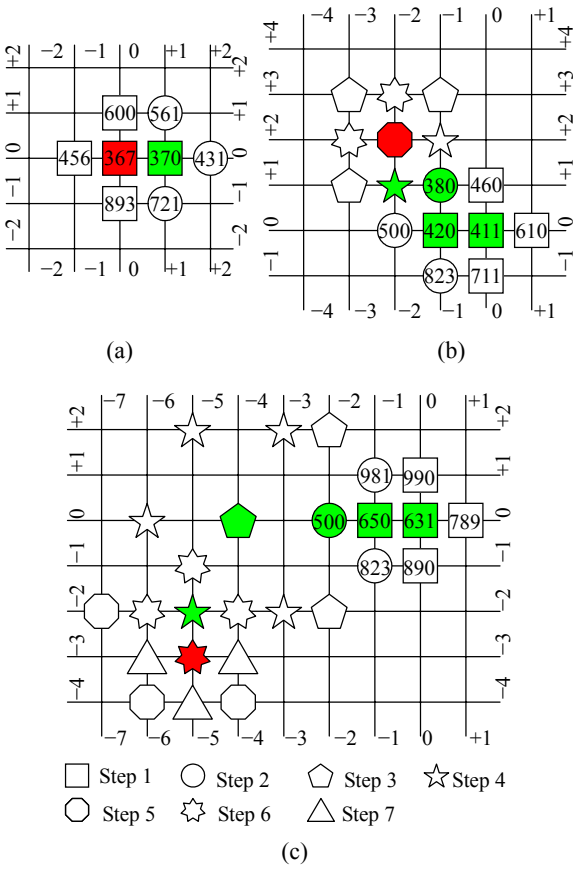


Fig.5 Three typical examples of the search path. Candidate points evaluated by using AMCSP (C_L=1.1) are labelled with their BDMs and the rest are labelled with the step numbers. (a) MV: (0, 0); (b) MV: (–2, 2); (c) MV: (–5, –3)

SIMULATION RESULTS

Our proposed AMCHS is simulated using the luminance of 20 popular video sequences, which are in QCIF/CIF/SIF/CCIR601 formats: Akiyo (250 frames), Foreman (90 frames), Silent (250 frames), Hall (250 frames), News (90 frames), Coastguard (250 frames), Mother-daughter (250 frames), Football

(97 frames), Mobile (95 frames) and Suzie (75 frames). Each of them has two different formats. Block size of 16×16, search window size of 15×15 and the sum of absolute difference (SAD) as the block distortion measure are used in all the simulations. Empirically the value of K=4 is chosen for the experiments. Because motion vector prediction is widely used in popular video compression standards, such as MPEG-4, H.264, median motion vector prediction technique (Tourapis *et al*., 2001) is also used in our experiments. With the median of the motion vectors of the left, top and top-right blocks of the original search position, we can evaluate fewer search points to get more accurate motion vectors for the current block.

The proposed AMCHS is compared against six traditional BMAs: FS, NTSS, DS, HEXBS, CDS, and CDHS. Table 1 shows the performance comparison on sequences "Coastguard" (QCIF). It is clear that AMCHS outperforms other fast BMAs in terms of speed, BDM, distance from and probability of the true motion vector.

Table 1 Performance comparison of different BMAs on sequence "Coastguard" (QCIF)

BMAs	*N*s	Speedup	BDM	Dis	Prob
FS	225	1.000	3.601	0.0000	100.000
NTSS	18.35	12.262	3.605	0.0033	99.878
DS	13.97	16.106	3.623	0.0244	99.375
HEXBS	13.02	17.281	3.940	0.3089	92.832
CDS	10.13	22.211	3.604	0.0031	99.927
CDHS	7.80	28.846	3.606	0.0042	99.870
AMCHS	**6.82**	**32.996**	**3.604**	**0.0023**	**99.956**

*N*s: the number of search points per block; BDM: Average SAD per pixel; Dis: the average distance from the true MV; Prob: the probability of finding the true MV, which is obtained by using FS

Table 2 shows average number of search points per block used by different BMAs with different sequences while Table 3 shows the average SAD per pixel according to Table 2. It is clear that our AMCHS gets the fastest speed among all the tested BMAs while its distortion performance is comparable with that of DS. Note that the AMCHS provides smaller BDMs than CDHS for nearly all the tested sequences, than CDS and HEXBS for most of the tested sequences, than DS for nearly half of the tested sequences and even smaller than NTSS for several sequences. Fig.6 plots the average number of search points used and the average SAD per pixel frame by frame using "Foreman" sequence. We can see that the AMCHS requires the fewest search points while it can maintain similar or better prediction quality as compared with other BMAs.

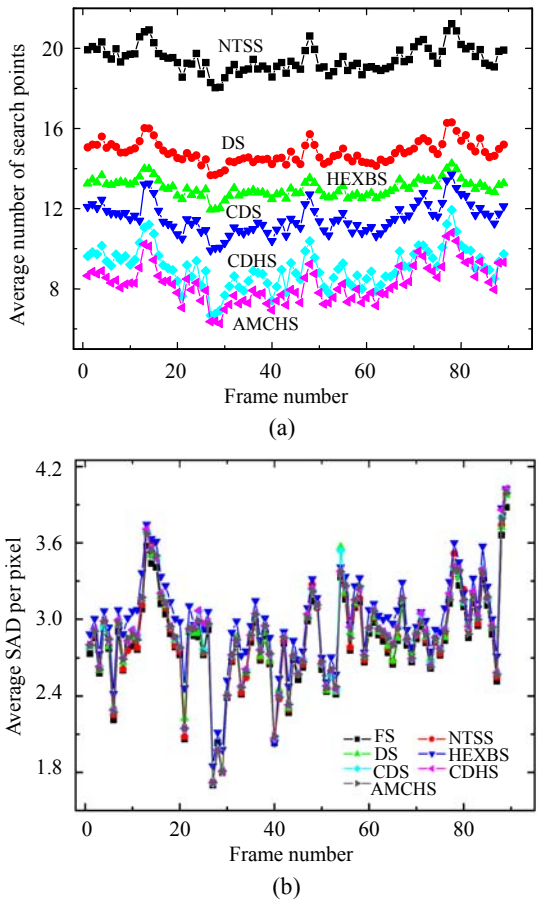


Fig.6 Frame-wise performance comparison between different BMAs with "Foreman" sequence (CIF) in terms of (a) average number of search points per block and (b) average SAD per pixel

CONCLUSION

In this paper, we proposed a novel adjustable multiple cross-hexagonal search algorithm. It employs an adjustable multiple cross search pattern, which can be used to find small motion vectors accurately. Two other hexagonal search patterns can be used to find large motion vectors efficiently. In the search process, half-way-stop and half-way-skip techniques are used to speed up the search. Simulation results showed that our proposed AMCHS is more robust, effective and efficient than other BMAs.

Table 2 Average numbers of search points per block used by different algorithms for different sequences

Sequences	BMAs						
	FS	NTSS	DS	HEXBS	CDS	CDHS	AMCHS
Akiyo[a]	225/225	17.62/17.37	13.31/13.17	11.93/11.54	9.46/9.27	6.28/5.75	**5.67**/**5.45**
Coastguard[a]	225/225	18.35/18.15	13.97/13.97	13.02/12.36	10.13/10.51	7.80/7.58	**6.82**/**7.30**
Foreman[a]	225/225	19.12/19.47	14.61/14.81	12.75/13.03	10.79/11.55	8.49/9.07	**7.48**/**8.27**
Hall[a]	225/225	17.78/18.49	13.56/13.87	11.83/10.05	9.75/10.34	6.63/**7.22**	**6.35**/8.26
Mother-daughter[a]	225/225	18.15/19.48	13.81/14.46	12.32/12.69	10.00/11.01	7.03/**8.22**	**6.72**/8.52
News[a]	225/225	17.78/17.69	13.31/13.45	11.78/11.79	9.54/9.65	6.44/6.34	**5.80**/**6.00**
Silent[a]	225/225	18.25/18.41	14.05/14.17	12.50/12.42	10.33/10.61	7.33/7.32	**6.76**/**7.08**
Football[b]	225/225	21.34/21.74	16.50/17.39	14.35/14.33	14.15/15.38	11.68/11.84	**10.76**/**11.33**
Mobile[b]	225/225	17.90/18.28	13.44/14.02	11.86/11.94	9.82/10.53	6.83/7.08	**6.43**/**6.86**
Suzie[b]	225/225	19.57/20.94	14.92/16.35	13.29/13.80	11.86/13.62	9.55/**10.71**	**9.39**/10.98

[a] QCIF (176×144)/CIF (352×288); [b] SIF (352×240)/CCIR601 (720×486)

Table 3 Average SAD per pixel with respect to different algorithms for different sequences

Sequences	BMAs						
	FS	NTSS	DS	HEXBS	CDS	CDHS	AMCHS
Akiyo[a]	**0.469/0.585**	0.470/0.586	0.479/0.590	0.478/0.593	0.470/0.587	0.470/0.587	0.470/0.587
Coastguard[a]	**3.601/4.736**	3.605/4.745	3.623/4.745	3.940/4.809	3.604/4.745	3.606/4.750	3.604/4.745
Foreman[a]	**3.134/2.819**	3.163/2.853	3.232/2.867	3.466/2.991	3.230/2.885	3.231/2.886	3.220/2.874
Hall[a]	**1.732/2.513**	1.734/2.519	1.752/2.525	1.743/2.535	1.734/2.521	1.734/2.524	1.733/2.521
Mother-daughter[a]	**1.416/1.521**	1.419/1.530	1.463/1.546	1.507/1.555	1.422/1.537	1.423/1.539	1.419/1.533
News[a]	**1.048/1.104**	1.049/1.109	1.057/1.110	1.057/1.111	1.049/1.110	1.049/1.112	1.049/1.111
Silent[a]	**1.695/1.861**	1.714/1.893	1.712/1.903	1.721/1.920	1.721/1.912	1.728/1.926	1.715/1.903
Football[b]	**10.109/8.297**	10.439/8.878	10.563/8.901	10.807/8.987	10.638/8.975	10.762/9.408	10.695/9.181
Mobile[b]	**9.783/6.468**	9.789/6.562	9.790/6.522	10.274/6.558	9.797/6.534	9.798/6.631	9.794/6.556
Suzie[b]	**2.598/2.641**	2.628/2.736	2.641/2.681	2.690/2.704	2.666/2.732	2.673/2.824	2.655/2.777

[a] QCIF (176×144)/CIF (352×288); [b] SIF (352×240)/CCIR601 (720×486)

References

Cheung, C.H., Po, L.M., 2002. A novel cross-diamond search algorithm for fast block motion estimation. *IEEE Trans. on Circuits Syst. Video Technol.*, **12**(12):1168-1177. [doi:10.1109/TCSVT.2002.806815]

Cheung, C.H., Po, L.M., 2005. Novel cross-diamond-hexagonal search algorithm for fast block motion estimation. *IEEE Trans. on Multimedia*, **7**(1):16-22. [doi:10.1109/TMM.2004.840609]

Po, L.M., Ma, W.C., 1996. A novel four-step search algorithm for fast block motion estimation. *IEEE Trans. on Circuits Syst. Video Technol.*, **6**(3):313-317. [doi:10.1109/76.499840]

Sorwar, G., Murshed, M., Dooley, L., 2003. A Fully Adaptive Performance-scalable Distance-dependent Thresholding Search Algorithm for Video Coding. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, p.649-652.

Sorwar, G., Murshed, M., Dooley, L., 2005. Fully Adaptive Performance Scalable Block-based Motion Estimation. Fifth Int. Conf. on Information, Communications and Signal Processing, p.1145-1149. [doi:10.1109/ICICS.2005.1689233]

Tham, J.Y., Ranganath, S., Ranganath, M., Kassim, A.A., 1998. A novel unrestricted center-biased diamond search algorithm for block motion estimation. *IEEE Trans. on Circuits Syst. Video Technol.*, **8**(4):369-377. [doi:10.1109/76.709403]

Tourapis, A.M., Au, O.C., Liou, M.L., 2001. New Results on Zonal Based Motion Estimation Algorithms—Advanced Predictive Diamond Zonal Search. IEEE International Symposium on Circuits and Systems, p.183-186.

Zhu, S., Ma, K.K., 2000. A new diamond search algorithm for fast block matching motion estimation. *IEEE Trans. on Image Processing*, **9**(2):287-290. [doi:10.1109/83.821744]

Zhu, C., Lin, X., Chau, L., Po, L.M., 2002. Hexagon-based search pattern for fast block motion estimation. *IEEE Trans. on Circuits Syst. Video Technol.*, **12**(5):349-355. [doi:10.1109/TCSVT.2002.1003474]

Zhu, C., Lin, X., Chau, L., Po, L.M., 2004. Enhanced hexagonal search for fast block motion estimation. *IEEE Trans. on Circuits Syst. Video Technol.*, **14**(10):1210-1214. [doi:10.1109/TCSVT.2004.833166]