A NOVEL HEXAGON-BASED SEARCH ALGORITHM FOR FAST BLOCK MOTION ESTIMATION

Ce Zhu, Xiao Lin, Lap-Pui Chau, Keng-Pang Lim, Hock-Ann Ang, Choo-Yin Ong

Centre for Signal Processing, School of Electrical & Electronic Engineering
Nanyang Technological University, Singapore 639798

ABSTRACT

In block motion estimation, search pattern with different shape or size has very important impact on search speed and distortion performance. In this paper, we propose a novel algorithm using hexagon-based search (HEXBS) pattern for fast block motion estimation. The proposed HEXBS algorithm may find any motion vector with fewer search points than the diamond search (DS) algorithm. The speedup gain of the HEXBS method over the DS algorithm is more striking for finding large motion vectors. Experimental results substantially justify the fastest performance of the HEXBS algorithm compared with several other popular fast algorithms.

1. INTRODUCTION

Block-matching motion estimation is vital to many motion-compensated video coding techniques/standards, which is aimed to exploit the strong temporal redundancy between successive frames. However, the motion estimation is quite computational intensive and can consume up to 80% of the computational power of the encoder if the full search (FS) is used by exhaustively evaluating all possible candidate blocks within a search window. Therefore, fast search algorithms are highly desired to significantly speed up the process without sacrificing the distortion seriously. Many computationally efficient variants were developed, typically among which are three-step search (TSS), new three-step search (NTSS) [1], four-step search (4SS) [2], block-based gradient descent search (BBGDS) [3] and diamond search (DS) [4] [5] algorithms.

In TSS, NTSS, 4SS and BBGDS algorithms, square-shaped search patterns of different sizes are employed. On the other hand, the DS algorithm adopts a diamond-shaped search pattern, which has demonstrated faster processing with similar distortion in comparison with NTSS and 4SS. This inspires us to investigate why diamond search pattern can yield speed improvement over some square-shaped search patterns and what is the mechanism behind. As a result, one may wonder whether there is any other pattern shape better than diamond for faster block motion estimation. In the following, we propose a hexagon-based search algorithm that can achieve substantial speed improvement over the diamond search algorithm with similar distortion performance.

2. HEXAGON BASED SEARCH ALGORITHM

2.1 Hexagonal Search Pattern

Ideally, a circle-shaped search pattern with a uniform distribution of a minimum number of search points is desirable to achieve the fastest search speed. Each search point can be equally utilized with maximum efficiency. Referring to the diamond search pattern [4] [5], we can see that the diamond shape is not approximate enough to a circle, which is just 90 degree rotation of a square. Consequently, a more circle-approximated search pattern is expected in which a minimum number of search points are distributed uniformly. A hexagon based search pattern is devised which is depicted in Fig. 1. There are two different sizes of hexagonal search patterns, of which the larger one consists of 7 check points marked as 1 with the center surrounded by 6 endpoints of the hexagon with the two edge points (up and down) being excluded. Like the shrunk diamond pattern, a smaller shrunk hexagonal pattern covering 4 check points marked as 2 is inside the large one, which is finally applied in the focused inner search. For the larger hexagonal search pattern, we can see the 6 endpoints are approximately uniformly distributed around the center with similar distances to the center. Note that the hexagonal search pattern also contains 2 fewer check points than the 9-point diamond search pattern. In the search process, the large hexagon-based search pattern keeps advancing with the center moving to any of the six endpoints. Whichever endpoint the center of the large search pattern moves to, there are always three new endpoints emerging and the other 3 endpoints being overlapped. Recall that in diamond search, 3 or 5 new points appear each time the diamond moves in different directions, in average 4 new points being evaluated for each move.

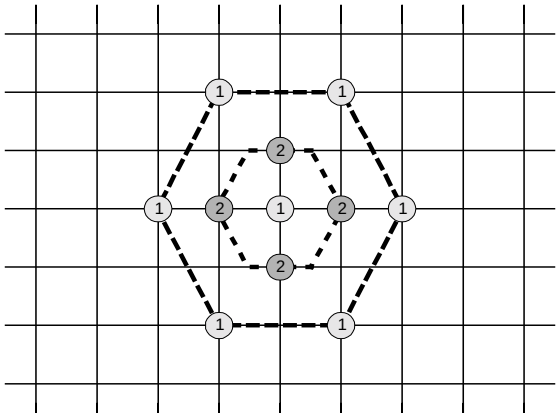

Fig. 1. Hexagon-based search (HEXBS): large (1) and small (2) hexagonal patterns.

2.2 Algorithm Development

With the designed hexagonal pattern, we develop the search procedure as follows. In the first step, the large hexagonal pattern with 7 check points is used for search. If the optimum is found at the center, we switch to use the shrunk hexagonal pattern including 4 check points for the focused inner search. Otherwise, the search continues around the point with minimum block distortion (MBD) using the same large hexagonal pattern. Note that while the large hexagonal pattern moves along the direction of decreasing distortion, only 3 new non-overlapped check points will be evaluated as candidates each time. Fig. 2 shows an example of the search path strategy leading to the motion vector (+4, -4), where 20 (=7+3+3+3+4) search points are evaluated in 5 steps sequentially. The proposed hexagon-based search (HEXBS) algorithm can be summarized in the following detailed steps.

Step i) **Starting**: The large hexagon with 7 check points is centered at (0,0), the center of a predefined search window in the motion field. If the MBD point is found to be at the center of the hexagon, proceed to Step iii) (Ending); otherwise, proceed to Step ii) (Searching).

Step ii) **Searching**: With the MBD point in the previous search step as the center, a new large hexagon is formed. Three new candidate points are checked, and the MBD point is again identified. If the MBD point is still the center point of the newly formed hexagon, then go to Step iii) (Ending); otherwise, repeat this step continuously.

Step iii) **Ending**: Switch the search pattern from the large size of hexagon to the small size of hexagon. The four points covered by the small hexagon are evaluated to compare with the current MBD point. The new MBD point is the final solution of motion vector.

From the above procedure, it can be easily derived that the total number of search points per block is

$$N_{HEXBS}(m_x, m_y) = 7+3\times n+4 \quad (1)$$

where (m_x, m_y) is the final motion vector found, and n is the number of execution of Step ii).

2.3 Analysis of the HEXBS Algorithm

In this subsection, we will examine the proposed HEXBS algorithm as compared with the diamond search (DS) algorithm in term of number of points evaluated to find same motion vectors. For block motion estimation, computational complexity can be measured by number of search points required for each motion vector estimation. In contrast with the HEXBS algorithm by which the number of search points used is indicated in Equation (1), the diamond search method requires the following number of search points per block

$$N_{DS}(m_x, m_y) = 9+M\times \tilde{n} +4 \quad (2)$$

where M is either 5 or 3 depending on the search direction and $\tilde{n}$ depends on the search distance. The $\tilde{n}$ in Equation (2) is always greater than or equal to the n in Equation (1) for finding a same motion vector (m_x, m_y). Especially for locating motion vectors in the direction of $[\tan^{-1}(\pm 2)] \pm k\pi$, k=0, 1, it can be derived that the speed improvement rate (*SIR*) of HEXBS over DS can be as high as over 80% in some scenarios, where the *SIR* for locating a motion vector can be obtained by

$$SIR = \frac{(9+M\times\tilde{n}+4)-(7+3\times n+4)}{7+3\times n+4}\times 100\% \quad (3)$$

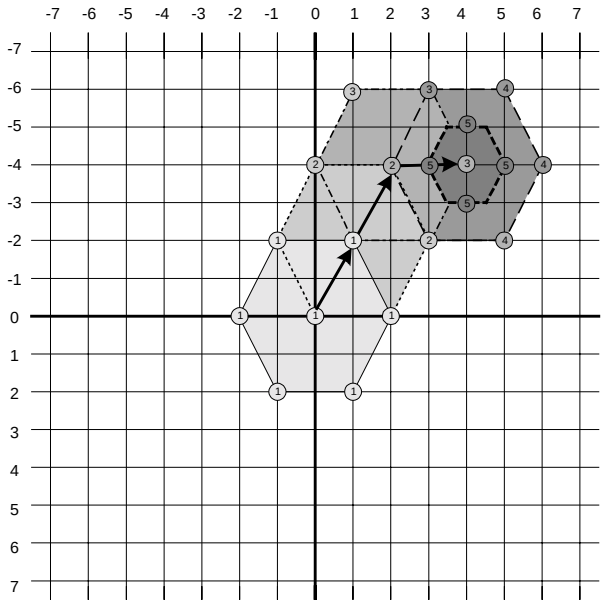

Fig. 2. Search path example locating the motion vector (+4, -4) by HEXBS. Note that a small HEXBS pattern is applied in the final step after the best candidate search point at step 3 remains the best at step 4. Totally 20 (=7+3+3+3+4) search points are evaluated in 5 steps.

By analyzing the minimum possible number of search points of HEXBS and DS, we show in Fig. 3 the number of search points saved by HEXBS as compared with DS for each motion vector location within ±4 region. From the figure, we can see that for the stationary and quasi-stationary motion vectors the HEXBS algorithm has 2 fewer search points, while more search points can be saved for locating medium to large motion vectors beyond the ±1 region. For example, 7 search points (block matches) can be saved by HEXBS for locating motion vectors (±1, ±3), (±4, ±2), (±2, ±4) and (±2, ±5). In short, the new HEXBS scheme can find any motion vector in motion field with fewer search points than the DS algorithm. Generally speaking, the larger the motion vector, the more search points the HEXBS algorithm saves, which is justified by the following experimental results.

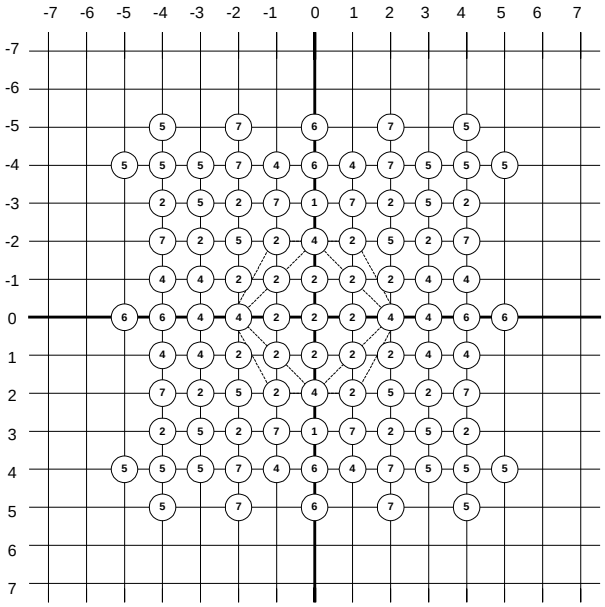

Fig. 3. Number of search points saved by HEXBS as compared to DS for each motion vector location.

3. EXPERIMENTAL RESULTS

The experimental setup is as follows: the distortion measurement of mean absolute difference (MAD) used, block size of 16×16, and search window size of ±7. Five standard video sequences "Salesman" (352×288, 448 frames), "Coastguard" (352×240, 300 frames), "Tennis" (720×480, 39 frames), "Garden" (720×480, 98 frames) and "Football" (720×480, 59 frames) were used, which vary in motion content as well as frame size. We also produced a sequence "DanceWolf" (720×480, 299 frames) from a film "Dance With The Wolf" which contains large motion with horse racing, for testing purpose.

Average MAD values and the search point numbers are summarized in Table I and Table II for different algorithms including FS, NTSS, 4SS, DS and our proposed HEXBS, respectively. (Note that the search point number per block for the FS is fixed as 255 for all video sequences.) We can see that the proposed HEXBS algorithm consumes the smallest number of search points with marginal increase in MAD compared with other fast algorithms. The number of search points used by the HEXBS method is substantially smaller than that by NTSS, 4SS or DS, nearly half the number of NTSS. Here we mainly compare the DS with the proposed HEXBS algorithm in terms of number of search point as well as MAD. Table III tabulates the average speed improvement rate (*SIR*) and average MAD increase in percentage of the proposed HEXBS over DS. For "Salesman" sequence with motion vectors limited within a small region around (0, 0), our proposed HEXBS algorithm achieves 21.41% speed improvement over DS. For "Coastguard" sequence with medium motion, the SIR of HEXBS over DS is 27.58%. For "Football" and "DanceWolf" which contain large motion, as predicted in theory our HEXBS algorithm has obtained higher speed improvement over DS, here more than 36%. The larger the motion in a video sequence, the larger the speed improvement rate of HEXBS over DS or the other fast algorithms will be. On the other hand, the degradation in MAD of HEXBS compared with DS is trivial, less than 1.7% or smaller of MAD increase for all the video sequences in our experiment. Fig. 4 plots a frame-by-frame comparison of MAD and search point number per block respectively for the different algorithms applied to "Garden" sequence. This figure shows the similar MAD performance for all the methods tested, while it clearly manifests the substantial superiority of the proposed HEXBS algorithm to the other methods in terms of number of search points used.

4. CONCLUSIONS

We have developed a novel fast algorithm using hexagon-based search pattern in block motion estimation, which demonstrates significant speedup gain over the diamond-based search and other fast search methods while maintaining similar distortion performance. The proposed HEXBS consistently has a faster search performance than DS regardless of no-, small-, medium- or large-motion. In other words, the new hexagon-based search scheme may find any motion vector in motion field with fewer search points than the DS algorithm. Generally speaking, the larger the motion vector, the more significant the speedup gain for the new method will be. The experimental results have verified the statement and convincingly demonstrated the superiority of the proposed HEXBS to the other fast methods in terms of using the smallest number of search points with marginal increase in distortion.

REFERENCES

[1] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," IEEE Transactions on Circuits & Systems for Video Technology, vol. 4, pp. 438-442, Aug. 1994.

[2] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," IEEE Transactions on Circuits & Systems for Video Technology, vol. 6, pp. 313-317, June 1996.

[3] L. K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," IEEE Transactions on Circuits & Systems for Video Technology, vol. 6, pp. 419-423, Aug. 1996.

[4] Shan Zhu, Kai-Kuang Ma, "A new diamond search algorithm for fast block-matching motion estimation," IEEE Transactions on Image Processing, vol. 9, no. 2, pp. 287-290, Feb. 2000.

[5] Jo Yew Tham, Surendra Ranganath, Maitreya Ranganath, and Ashraf Ali Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," IEEE Transactions on Circuits & Systems for Video Technology, vol. 8, no. 4, pp. 369-377, Aug. 1998.

TABLE I. Average MAD per pixel for different methods and different video sequence

	Salesman	Coastguard	Tennis	Garden	Football	DanceWolf
FS	2.772	5.040	7.152	6.162	8.092	2.848
NTSS	2.776	5.081	7.587	6.510	8.350	2.887
4SS	2.807	5.132	7.540	6.702	8.500	2.910
DS	2.802	5.095	7.629	6.732	8.480	2.901
HEXBS	**2.821**	**5.168**	**7.745**	**6.845**	**8.599**	**2.944**

Table II. Average number of search points per block for different methods and different video sequence

	Salesman	Coastguard	Tennis	Garden	Football	DanceWolf
NTSS	18.084	21.358	22.713	27.481	23.813	23.960
4SS	17.355	19.687	19.855	21.661	20.559	20.586
DS	13.065	16.365	16.909	20.750	19.020	18.855
HEXBS	**10.761**	**12.827**	**13.005**	**15.375**	**13.910**	**13.816**

Table III. Average speed improvement rate (*SIR*) and average MAD increase in percentage of our HEXBS over DS

	Salesman	Coastguard	Tennis	Garden	Football	DanceWolf
Avg. *SIR* (%)	21.41	27.58	30.02	34.96	36.74	36.47
Avg. MAD increase (%)	0.68	1.43	1.52	1.68	1.40	1.48

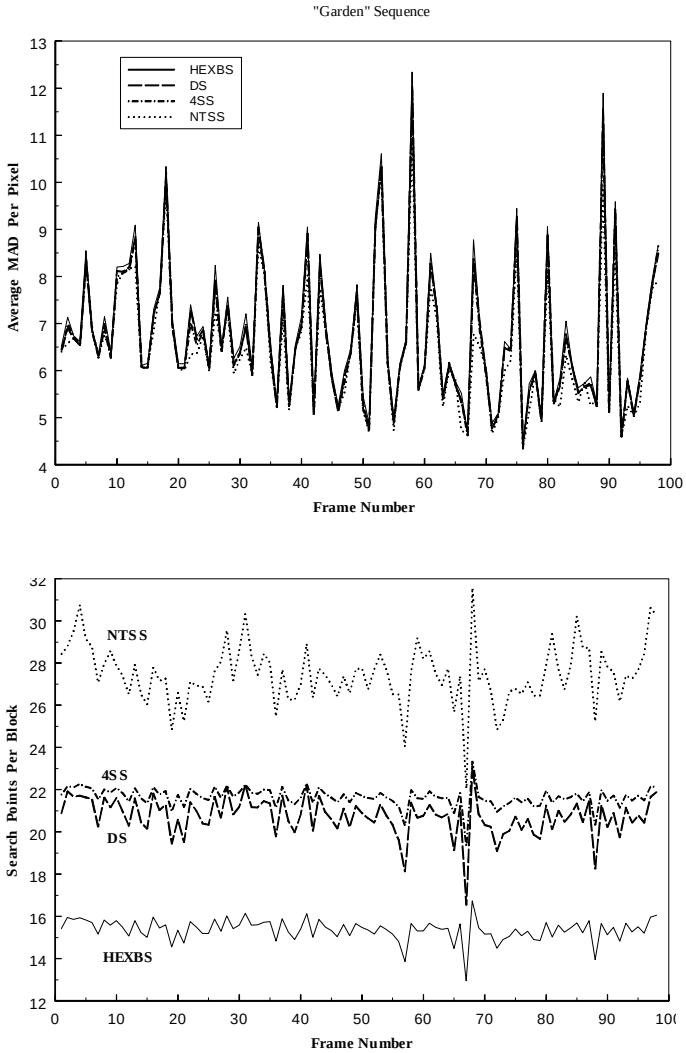

Fig. 4. Performance comparison of NTSS, 4SS, DS and the proposed HEXBS for "Garden" sequence in terms of average MAD per pixel and the average number of search points per block, respectively.