An Unsymmetrical Diamond Search Algorithm for H.264/AVC Motion Estimation

Jun Luo and Jiaxin Peng

Key Laboratory of Optoelectronic Technology & System of the Ministry of Education, Chongqing University, Chongqing 400030, China
luojun@cqu.edu.cn

Abstract. The performance of motion estimation is of great importance for H.264 advanced video coding. It is estimated that motion estimation consumes about 70% of the encoding time. Lots of motion estimation algorithms are proposed to improve the encoding speed. Unlike the assumption of most motion estimation algorithms, the horizontal motion vectors are much larger than the vertical in most cases. With the unsymmetrical characteristic, this paper presents a new diamond search based motion estimation algorithm to improve the efficiency. The unsymmetrical diamond search shorten the vertical step to lower down the computation complexity. The search points of the big template and the small template are reduced to 5 and 3 respectively. The simulation results show that, the unsymmetrical diamond search can achieve much more significant speedup ratio than other motion estimation algorithms with relatively high probability.

Keywords: motion estimation, H.264, diamond search, block matching.

1 Introduction

H.264/MPEG4 Part 10 Advanced Video Coding (AVC) is one of the most popular international standards for video coding, which is developed by the joint video team of the Video Coding Experts Group (VCEG) of the International Telecommunication Union Telecommunication Standardization Sector (ITU-T) and the Moving Picture Experts Group (MPEG) of International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) [1-3]. This standard is designed for a broad application range that covers video conferencing, network streaming video, digital storage media, digital communication, television broadcasting, et al.

Compared with previous standards, H.264 can provide better video quality at substantially half or less of bitrates. The decrease in bitrates requires more accurate motion estimation, such that the computation complexity increases. It is estimated that motion estimation constitutes roughly 70% of the encoding time for an H.264 encoder. Thus, it is important and necessary to reduce the computation complexity of motion estimation.

In this paper, we propose an unsymmetrical diamond search (UDS) algorithm for motion estimation. The objective of the algorithm is to achieve faster motion

T. Tan et al. (Eds.): IGTA 2013, CCIS 363, pp. 54–65, 2013.

estimation with good quality. UDS uses unsymmetrical diamond search patterns which are derived from diamond search patterns. With unsymmetrical diamond search patterns, search points can be greatly reduced. The main contributions of this paper are: (1) incorporate unsymmetricity into diamond search to improve the performance; (2) reduce the computation complexity to speed up motion estimation.

The remainder of this paper is organized as follows. Section 2 reviews the principle of motion estimation for video coding. Block distortion measure and fast search algorithms are presented in details. Section 3 introduces the classic diamond search algorithm and its variants, while the unsymmetrical diamond search algorithm is discussed in section 4. Therefore, we compare the performance of UDS with the existing mainstream algorithms in section 5, which is followed by some concluding remarks in section 6. Finally, some acknowledgements are listed in section 7.

2 Motion Estimation

In the H.264/AVC standard, motion estimation is based on block matching algorithms (BMA). BMA divides a frame into small macroblocks, of which the partition sizes may be 4×4, 4×8, 8×8, 8×4, 8×16, 16×8 or 16×16. To do motion estimation for a macroblock of the current frame, BMA searches the minimum block distortion measure (BDM) macroblock in the search window of the reference frame. Therefore, the minimum BDM reference macroblock is used to predict the current macroblock, and the displacement of the two macroblocks is motion vector (MV). After motion estimation, the residual macroblock, the difference of the two macroblocks, contains far less information. It is obvious that more accurate motion estimation will produce less residual and lower bitrates. Motion estimation mainly composes of BDM and BMA.

2.1 Block Distortion Measure

In theory, mean square error (MSE) has been the most widely used BDM to describe the similarity for low computational complexity. MSE is based on Gauss probability distribution assumption. However, MSE is not the best criterion for motion estimation because the images of the video do not obey Gauss probability distribution assumption in most of cases. For simplicity, MSE is employed for motion estimation approximately. In mathematics, $\forall(\Delta x, \Delta y) \in SR$, MSE can be represented by

$$MSE = \min \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \Big[f(x,y) - r(x+\Delta x, y+\Delta y) \Big]^2, \tag{1}$$

where f is the current frame, r is the reference frame, x and y are the co-ordinates of a frame, M and N are the width and the height of a frame respectively, SR is the search range, and $(\Delta x, \Delta y)$ is the motion vector (MV).

However, to remove division and square in the BDM, H.264 uses sum of absolute differences (SAD) instead of MSE. SAD is represented by

$$SAD = \min \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left| f(x,y) - r(x+\Delta x, y+\Delta y) \right| \quad \forall (\Delta x, \Delta y) \in SR. \qquad (2)$$

2.2 Fast Search Algorithm

BMA is an important part for motion estimation. Regardless of computation complexity, exhaustive search (ES) or full search (FS) is considered to be the best BMA. ES searches every point in the search window, and is sure to find the minimum BDM macroblock. However, the computation complexity is too high to allow realtime implementations. Huang, et al. [4] estimated the computation of ES. Realtime motion estimation for a CIF (352×288) 30 fps video with search range $[-16, 15]$ requires 9.3 giga-operations per second (GOPS) while 127 GOPS is required for a D1 (704×576) 30 fps video with search range $[-32, 31]$. Therefore, many faster BMAs have been proposed to improve the performance of motion estimation. As opposed to ES, these BMAs are called Fast Search Algorithms (FSA).

Sequentially, three step search (TSS) [5], cross search (CS) [6], new three step search (NTSS) [7], four step search (FSS) [8], diamond search (DS) [9], hexagon-based search (HEXBS) [10], cross-diamond search (CDS) [11], hybrid unsymmetrical-cross multi-hexagon-grid Search (UMHexagonS) [12], directional diamond search (DDS) [13], prediction-based directional asymmetric search (PBDAS) [14], directional gradient descent search (DGDS) [15], et al. have been proposed in the past years. Among these FSAs, DS is one of the most excellent for the balance between speed and accuracy. In the next section, we will give a brief introduction about DS and its variants.

3 Traditional Diamond Search

As illustrated in Fig.1, DS uses two types of search patterns: large diamond search pattern (LDSP) and small diamond search pattern (SDSP). DS repeatedly use LDSP to search the minimum BDM in the nine points until the minimum BDM point is at the center of LDSP. Then SDSP is used to find the more accurate position. The details of DS are summarized as follows.

Step 1. The initial LDSP is centered at the origin of the search window. Compare the SADs of the nine points of LDSP. If the point with minimum BDM is at the center of LDSP, go to **Step 3.** Otherwise, go to **Step 2.**

Step 2. The previous minimum BDM point is considered as the center of new LDSP. When out of search range, go to **Step 3.** Compute the SADs. If the minimum BDM point is at the center, go to **Step 3.** Otherwise, repeat **Step 2.**

Step 3. Replace LDSP with SDSP. Locate the center of SDSP at the minimum BDM point in the previous search. Compute the SADs of the five points of SDSP. The minimum BDM point is the final solution.

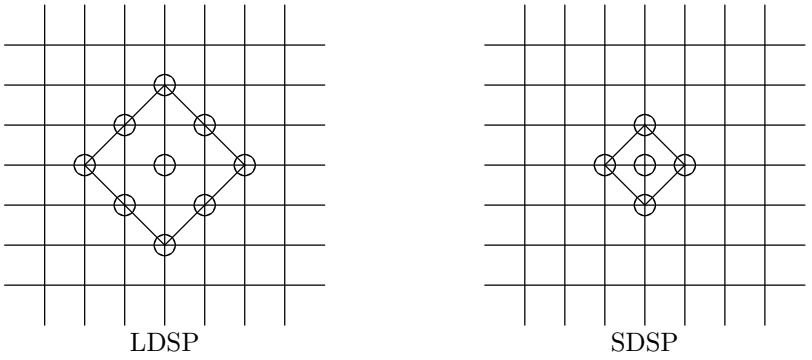

Fig. 1. LDSP and SDSP used by DS

A variant of DS is cross-diamond search (CDS) algorithm. As illustrated in Fig.2, CDS combines DS with CS. Firstly, it searches the minimum BDM point with CSP. When the minimum BDM point is at the center of CSP, search terminates. Otherwise, start half-diamond search by repositioning the center of LDSP at the previous minimum BDM point. If the new minimum BDM point is at the center of LDSP, stop the current search. Otherwise, begin diamond search till termination.

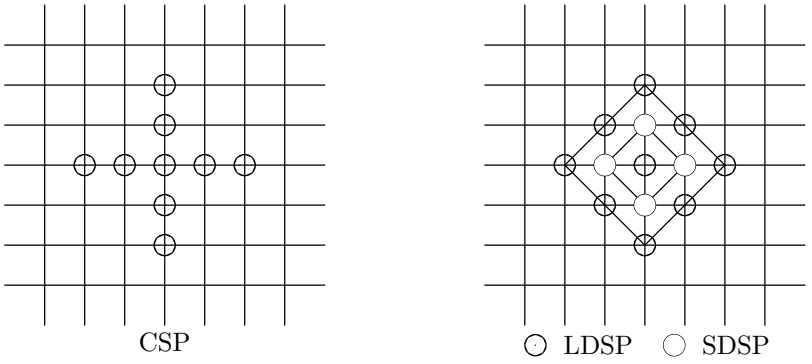

Fig. 2. Search patterns used by CDS

Another variant of DS is directional diamond search (DDS) algorithm. As illustrated in Fig.3, DDS uses a horizontal diamond search pattern (HDSP) and a vertical diamond search pattern (VDSP). Each pattern have five checking points, an origin and four vertices. The two points near the origin are called near points (NPT) while the other two points are called far points (FPT). Similarly with DS, DDS uses HDSP at the beginning of the search. If the minimum BDM point is at the center of HDSP, search with VSDSP and stop the search. Otherwise, if

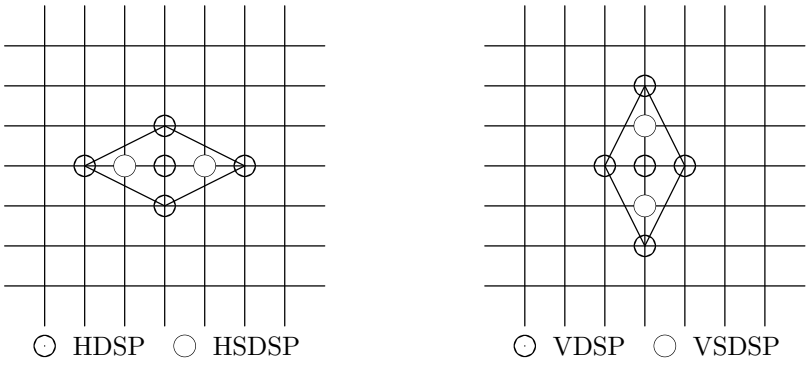

Fig. 3. Search patterns used by DDS

the minimum BDM point is an FPT, continue search with HDSP. additionally, if the minimum BDM point is an NPT, change the current search pattern to the other search pattern. Repeat the above steps till termination.

Other DS variants are also very excellent, but are omitted here due to the limitation of paper length. The above three algorithms are typical DS algorithm or its variants. To investigate the following algorithm, the three algorithms are picked as the reference.

4 Unsymmetrical Diamond Search

The search patterns of many FSAs are symmetrical. That is, they search the minimum BDM point both in horizontal and vertical with the same scale. However, in most of cases, the horizontal motion vector (MV) is much larger than the vertical. Thus, the horizontal motion and the vertical motion are said to be unsymmetrical. With the unsymmetrical characteristic, the horizontal search should be much rougher than the vertical search to speed up the motion estimation. In the above section, DS and CDS search in horizontal and vertical symmetrically. Hence, there is something to be improved. Additionally, DDS uses two unsymmetrical patterns, and switches the current pattern to the other when the minimum BDM point is an NPT. However, there are two evidences to prove that the VDSP of DDS is not reasonable. Firstly, as discussed above, the vertical motion is often very slightly despite of some irregular cases. Secondly, all FSAs is based on the assumption that MVs are always monotonically distributed in the adjacent area of the search window center. Therefore, the minimum BDM point is much more likely to be an NPT than an FPT. In this case, the HDSP should be used but the VDSP.

To avoid these problems, we propose a new DS variant, unsymmetrical diamond search (UDS). As shown in Fig.4, UDS uses two search patterns: large diamond search pattern (LDSP) and small diamond search pattern (SDSP). Unlike DS, the LDSP is the same as the HDSP of DDS. There are five points in

the LDSP: the center point (CPT), the up point (UPT), the down point (DPT), the left point (LPT) and the right point (RPT). The search scale of LDSP is 2 pixels in horizontal while 1 pixel in vertical. The SDSP have 3 points in horizontal with search scale of 1 pixel. Moreover, UDS refers to another four search patterns, which are illustrated by Fig.5. There are four half diamond search pattern (HDSP): up half diamond search pattern (UHDSP), down half diamond search pattern (DHDSP), left half diamond search pattern (LHDSP), right half diamond search pattern (RHDSP). For the four HDSPs, the centers of the diamonds formed by the dashed lines are the CPTs.

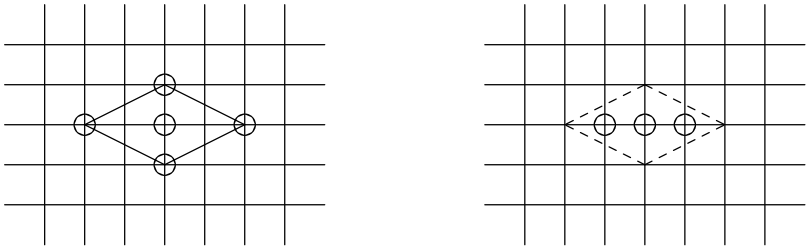

Fig. 4. Large and small search patterns used by UDS

UDS differs from DS in: (1) the search scales in vertical and horizontal are certain in the current search. (2) four HDSPs are employed to reduce computation. The procedure of UDS can be summarized as below.

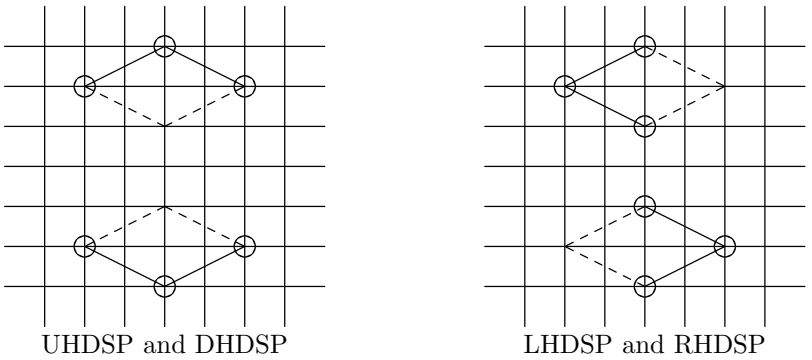

Fig. 5. Half diamond search patterns used by UDS

Step 1: The initial LDSP is placed at the center of the search window. Compare the SAD of CPT with other four points. If CPT is the minimum BDM point , stop the current search. Otherwise, go to **Step 2**.

Step 2: If the minimum BDM point is UPT in **Step 1**, switch search pattern to UHDSP. If DPT, switch search pattern to DHDSP. If LPT, switch search pattern to LHDSP, while if RPT, switch search pattern to RHDSP. Reposition the CPT of the chosen HDSP at the previous minimum BDM point. Compute the SADs of the four points. Go to **Step 3** when out of search range or the minimum BDM point is CPT. Otherwise, repeat this step till termination.

Step 3: Switch LDSP or HDSP to SDSP. Compute the SADs of the three points. The position of the minimum BDM point is the final solution.

With the procedure above, UDS only checks five points at the start. After that, it repeatedly search with HDSPs. There are only three points to be computed because the other two points have been computed in the former step. Consequently, 40% of computation in **Step 2** is saved. Finally, UDS only checks three points while DS checks five points. Fig.6 presents three examples of UDS.

5 Simulation Results

In this section, UDS is implemented by using the GNU Pascal language on a Pentium E 2160 1.8GHz personal computer running Ubuntu 10.04 operating system with Linux 2.6.32-44-generic kernel. The capacity of the RAM is 2GB, and the compiler is GPC 20070904. Several mainstream FSAs, TSS, HEXBS, DS, CDS and DDS, are implemented to investigate the performance of UDS. Here, search efficiency is defined to describe the performance of these FSAs. Search efficency is the comprehensive specification of searching speed and accuracy for macroblock matching. As the H.264/AVC standard suggests, SAD is considered to be the BDM.

In addition, three video sequences, *flower* (CIF, 250 frames), *foreman* (CIF, 300 frames) and *football* (CIF, 260 frames), are employed for simulations. *flower* is a horizontal-motion sequence, *foreman* is a vertical-motion sequence while *football* is a large-motion sequence mixed with horizontal and vertical motions.

To study the search efficiency, UDS is compared to the other FSAs in four aspects: (1)the average minimum SADs (MSAD) per pixel; (2) the average number of search points (NSP) per macroblock; (3) average distance from the true motion vectors per macroblock; (4) average probability of finding the true motion vectors per macroblock. The MVs searched by ES are regarded as the true MVs.

Moreover, the macroblock size is 16×16, the search range is $[-7, +7]$, and the reference frame is the last one of the current frame. The seven algorithms are simulated using the luminance of the three standard video sequences. Table 1-3 present the simulation results for *flower*, *foreman* and *football* respectively.

In Table 1, with the speedup ratio up to 20.531, UDS can also reach the high probability of 99.23%. As compared to DS, UDS saves about 6.77 search points per macroblock. When compared to CDS and DDS, UDS saves about 3.57 and 1.40 search points respectively. For sequence *flower*, UDS is 68.08% faster than DS, 35.88% faster than CDS, and 14.05% faster than DDS. The MVs in sequence *flower* are mainly horizontal. Unlike other FSAs, UDS can greatly reduce several

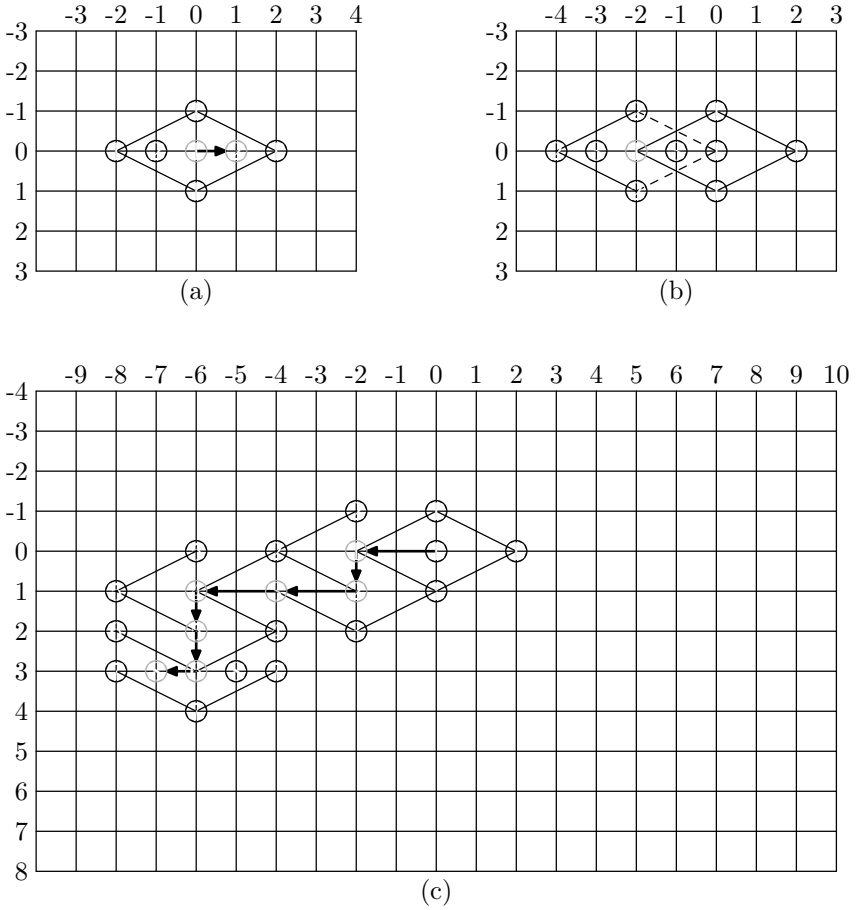

Fig. 6. Examples of UDS: each candidate point is marked with the corresponding step number. In each step, there is only one minimum BDM point. (a) A search path of two steps for UDS with MV (+1, 0). (b) A search path of three steps with MV (-2, 0). (c) A search path of eight steps for UDS with MV (-7, 3).

Table 1. Search efficiency comparisons (video sequence: *flower*)

FSA	MSAD	NSP	Speedup	Distance	Probability
ES	8.907	204.148	1.000	0.000	100.00%
TSS	8.930	23.347	8.744	0.641	74.48%
HEXBS	11.699	10.314	19.793	0.344	72.22%
DS	8.912	16.713	12.215	0.021	99.41%
CDS	8.917	13.512	15.109	0.033	98.73%
DDS	10.535	11.341	18.001	0.255	75.72%
UDS	8.913	9.943	20.531	0.026	99.23%

Table 2. Search efficiency comparisons (video sequence: *foreman*)

FSA	MSAD	NSP	Speedup	Distance	Probability
ES	7.360	204.035	1.000	0.000	100.00%
TSS	7.423	23.280	8.764	0.583	88.95%
HEXBS	7.927	10.331	19.749	0.756	82.21%
DS	7.476	14.844	13.745	0.352	95.05%
CDS	7.703	12.721	16.039	0.540	88.65%
DDS	7.567	8.929	22.850	0.465	88.08%
UDS	7.568	8.475	24.075	0.450	90.89%

search points in vertical direction. In this case, DDS may incorrectly use VDSP instead of HDSP. Therefore, the probability is very low and the MSAD value is very large.

In Table 2, UDS reaches the speedup ratio of 24.075 with the high probability of 90.89%. As compared to DS, CDS and DDS, UDS saves about 6.37, 4.25 and 0.45 search points respectively. For sequence *foreman*, UDS is 75.15% faster than DS, 50.10% faster than CDS, and 5.36% faster than DDS. The MVs in this sequence are mainly vertical. Thus, There is a reduction in probability for UDS. When searching vertical MVs, DDS achieves relatively high probability for the use of VDSP. Because of symmetrical characteristics, other FSAs are not sensitive to the directions of motions.

Table 3. Search efficiency comparisons (video sequence: *football*)

FSA	MSAD	NSP	Speedup	Distance	Probability
ES	15.069	204.122	1.000	0.000	100.00%
TSS	15.453	23.407	8.720	2.207	67.63%
HEXBS	16.288	11.086	18.413	2.256	55.81%
DS	15.555	18.933	10.781	1.439	80.18%
CDS	15.970	12.797	15.950	1.877	64.81%
DDS	15.679	11.595	17.604	1.691	67.62%
UDS	15.836	10.048	20.315	1.837	68.43%

In Table 3, UDS reaches the speedup ratio of 20.315, and the probability is the second highest one, regardless of ES. As compared to DS, CDS and DDS, UDS saves about 8.89, 2.75 and 1.55 search points respectively. For sequence *football*, UDS is 88.43% faster than DS, 27.36% faster than CDS, and 15.39% faster than DDS. The motions in sequence *football* is very complex, which test the comprehensive performance of these FSAs. Additionally, Fig.7-8 present the average NSPs and the average probability per macroblock for each frame of sequence *football*.

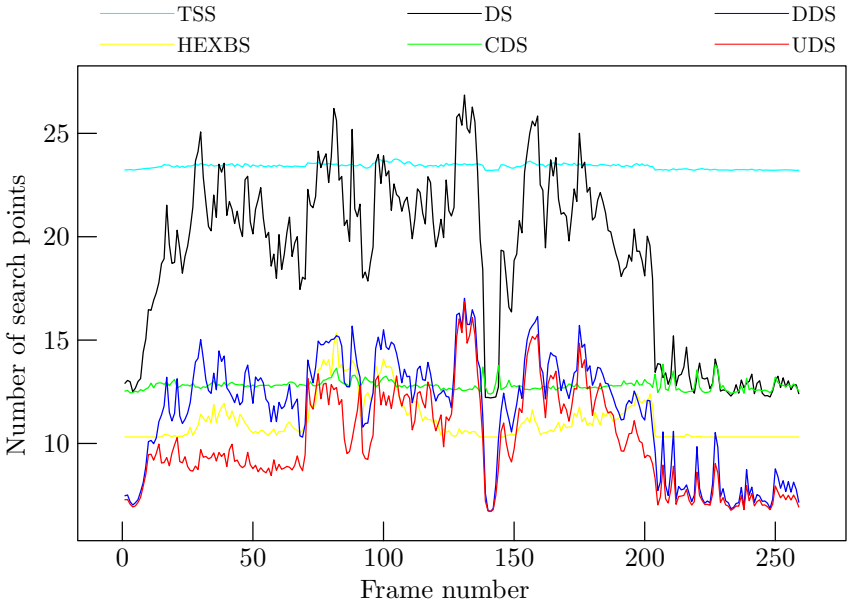

Fig. 7. Average NSP per macroblock in each frame of sequence *football*

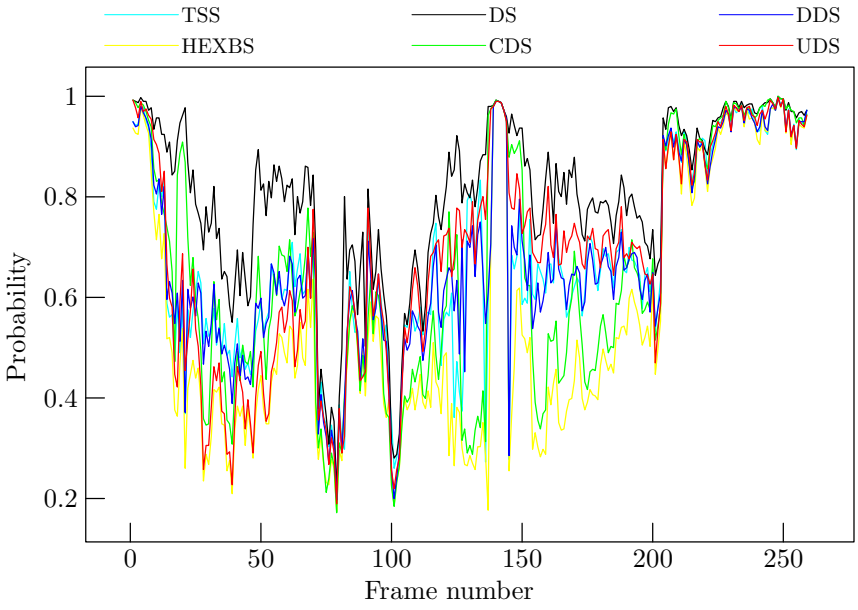

Fig. 8. Probability for finding the true MVs in each frame of sequence *football*

With the simulation results, UDS is much faster than other FSAs with relatively high probability. As well, TSS and HEXBS are widely used for ME. However, TSS has low speed and low probability for the wide search range. The speed of HEXBS is fast enough while the probability is very low. The MSAD value of HEXBS is also much larger than other FSAs. Thus, the comprehensive performance of TSS and HEXBS is not as good as other FSAs.

6 Conclusion

This paper presents an unsymmetrical diamond search algorithm for H.264/AVC motion estimation. It incorporates unsymmetricity into traditional diamond search, and shortens the search step in vertical directions. Therefore, several search points can be efficiently reduced to speedup the process of motion estimation. Half diamond search patterns are also employed to search no overlapping points. The simulation results show that UDS reaches great speedup ratio with relatively high enough probability in various cases.

Acknowledgement. The authors would like to thank Xiaohua Yang and Kai Meng for their helpful suggestions and detailed remarks. The authors also wish to thank GNU and the great contributors of GPC.

References

[1] Wiegand, T., Sullivan, G.J., Bjøntegaard, G., Luthra, A.: Overview of the H.264/AVC video coding standard. IEEE Transactions on Circuits and Systems for Video Technology 13(7), 560–576 (2003)

[2] Joint Video Team, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC (2003)

[3] Vani, R., Sangeetha, M.: Survey on H.264 Standard. In: Meghanathan, N., Chaki, N., Nagamalai, D. (eds.) CCSIT 2012, Part III. LNICST, vol. 86, pp. 397–410. Springer, Heidelberg (2012)

[4] Huang, Y.W., Chen, C.Y., Tsai, C.H., Shen, C.F., Chen, L.G.: Survey on block matching motion estimation algorithms and architectures with new results. Journal of VLSI Signal Processing 42, 297–320 (2006)

[5] Koga, T., Linuma, K., Hirano, A., Iijima, Y., Ishiguro, T.: Motion compensated interframe coding for video conferencing. In: Proc. Nat. Telecommum. Conf., pp. C9.6.1–C9.6.5 (1981)

[6] Ghanbari, M.: The cross search algorithm for motion estimation. IEEE Transactions on Communications 38(7), 950–953 (1990)

[7] Li, R., Zeng, B., Liou, M.L.: A new three-step algorithm for block motion estimation. IEEE Transactions on Circuits and Systems for Video Technology 4(4), 438–442 (1994)

[8] Po, L.M., Ma, W.C.: A novel four-step search algorithm for fast block motion estimation. IEEE Transactions on Circuits and Systems for Video Technology 6(3), 313–317 (1996)

[9] Zhu, S., Ma, K.K.: A new diamond search algorithm for fast block-matching motion estimation. IEEE Transactions on Image Processing 9(2), 287–290 (2000)
[10] Zhu, C., Lin, X., Chuan, L.P.: Hexagon-based search patterns for fast block motion estimation. IEEE Transactions on Circuits and Systems for Video Technology 12(5), 349–355 (2002)
[11] Cheung, C.H., Po, L.M.: A novel cross-diamond search algorithm for fast block motion estimation. IEEE Transactions on Circuits and Systems for Video Technology 12(12), 1168–1177 (2002)
[12] Jia, H., Zhang, L.: Directional diamond search pattern for fast block motion estimation. Electronics Letters 39(22), 1581–1583 (2003)
[13] Chen, Z., Xu, J., He, Y., Zheng, J.: Fast integer-pel and fractional-pel motion estimation for H.264/AVC. Journal of Visual Communication and Image Representation, Special Issue on Emerging H.264/AVC Video Coding Standard 17, 264–290 (2006)
[14] Kuo, C.M., Kuan, Y.H., Hsieh, C.H., Lee, Y.H.: A novel prediction-based directional asymmetric search algorithm for fast block-matching motion estimation. IEEE Transactions on Circuits and Systems for Video Technology 19(6), 893–899 (2009)
[15] Po, L.M., Ng, K.H., Cheung, K.W., Wong, K.M., Uddin, Y.M.S., Ting, C.W.: Novel directional gradient descent searches for fast block motion estimation. IEEE Transactions on Circuits and Systems for Video Technology 19(8), 1189–1195 (2009)