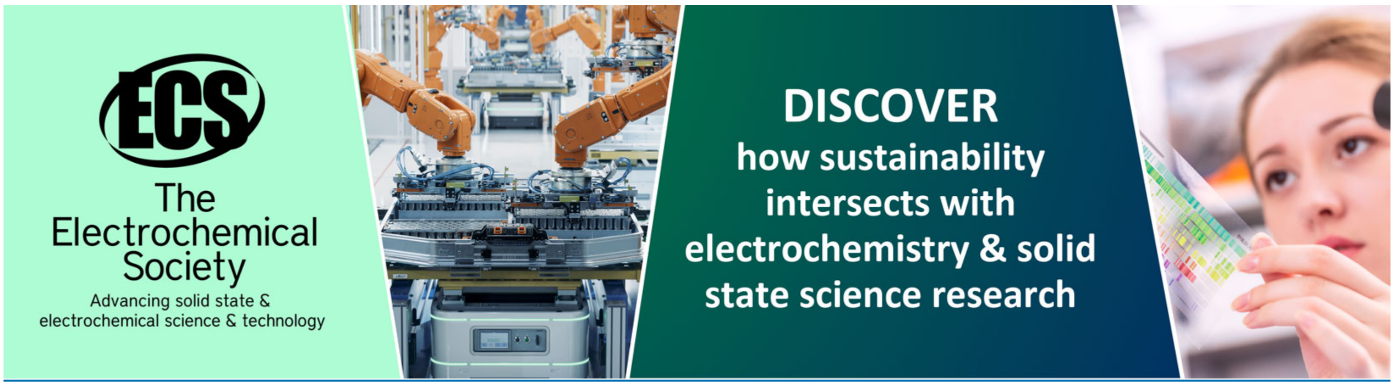
PAPER • OPEN ACCESS

A Novel Kite Cross Hexagonal Search Algorithm for Fast Block Motion Estimation

To cite this article: Tareq Zaid Hamood and Matheel Emadulden Abdulmunem 2021 *J. Phys.: Conf. Ser.* **1804** 012026

View the article online for updates and enhancements.


ECS
The Electrochemical Society
Advancing solid state & electrochemical science & technology
DISCOVER how sustainability intersects with electrochemistry & solid state science research

A Novel Kite Cross Hexagonal Search Algorithm for Fast Block Motion Estimation

Tareq Zaid Hamood[1] and Matheel Emaduldeen Abdulmunem[2]

[1] Unit of Remote Sensing, College of Science, University of Baghdad, Baghdad, Iraq

[2] Department of Computer Science, University of Technology, Baghdad, Iraq

tarik_z29@yahoo.com

Abstract. The performance quality and searching speed of Block Matching (BM) algorithm are affected by shapes and sizes of the search patterns used in the algorithm. In this paper, Kite Cross Hexagonal Search (KCHS) is proposed. This algorithm uses different search patterns (kite, cross, and hexagonal) to search for the best Motion Vector (MV). In first step, KCHS uses cross search pattern. In second step, it uses one of kite search patterns (up, down, left, or right depending on the first step). In subsequent steps, it uses large/small Hexagonal Search (HS) patterns. This new algorithm is compared with several known fast block matching algorithms. Comparisons are based on search points and Peak Signal to Noise Ratio (PSNR). According to results obtained in this paper, KCHS needs less search time than others algorithms and gives very acceptable performance quality.

1. Introduction

Efficient coding of video sequences is an important process and becomes essential in many multimedia and communication applications. These applications require a very high compression ratio because of limited channel bandwidth of real video playback [1].

Block Matching (BM) algorithms have been widely used in various video coding standards such as MPEG series and H.26x due to their efficiency and implementation simplicity in both software and hardware. In block matching, the current frame is divided into Macro Blocks (MBs) of equal size. Each of these MBs is compared with the block at the same position and its adjacent blocks in the previous frame (reference frame). The motion vector represents the movement of a macro block from one location to another in the reference frame. The MVs of all blocks in the frame are considered as the estimated motion of that frame. The search area for a good macro block match is constrained up to p pixels on all four sides of the corresponding macro block in the previous frame. This ‘p’ is called as the **search parameter**. Larger search parameter is required for larger motions which lead to more computationally expensive motion estimation process. The output of a cost function is used to match one macro block with another. The cost function is based on a **Block Distortion Measure (BDM)**. The block with minimum cost is considered as the closest to current block. Many cost functions are used. **Mean Absolute Difference (MAD)** is the most popular and less computationally expensive. MAD is given by equation (1). **Mean Squared Error (MSE)** is another cost function and is given by equation (2).

(1) $MAD = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|C_{ij} - R_{ij}\right|$

(2) $MSE = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left(C_{ij} - R_{ij}\right)^2$

Where N is the side of the macro bock, C_{ij} and R_{ij} are the pixels being compared in current macro block and reference macro block, respectively [2]. The basic idea of block matching is shown in figure

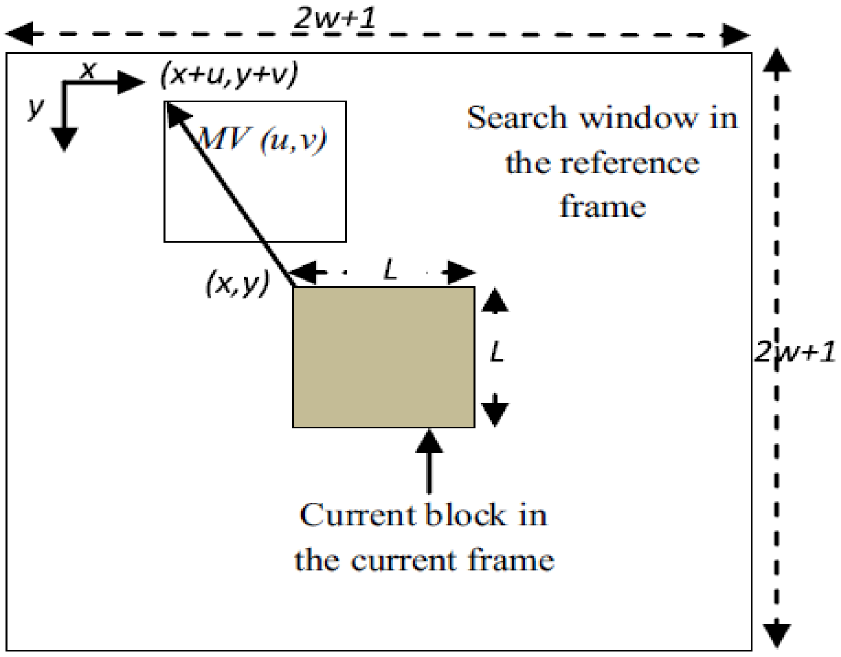

2w+1
x
y
(x+u,y+v)
MV (u,v)
Search window in the reference frame
(x,y)
L
L
2w+1
Current block in the current frame

Figure 1. Block matching algorithm.

The rest of the paper is organized as follow. In section two, some related works are described. Section three explains some popular block matching algorithms used in this paper. Section four gives the details of the proposed algorithm. Section five gives results of applying algorithms described in section three and the proposed algorithm on some video sequences and in section five, the conclusions are given.

2. Related works

Many block matching algorithms have been proposed in last years. A new search algorithm called **Cross Diamond Search (CDS)** was introduced by C.H. Cheung and L.M. Po in 2002. In this algorithm, a cross shape pattern is used in the initial step. This shape consists of nine points (the center of the search window and 8 surrounding points) forming the shape (+). If the minimum BDM point is the center then the search stops. Otherwise, Large Diamond Search Pattern (LDSP) is repeatedly applied until the minimum BDM point becomes the center of LDSP. Then, a Small Diamond Search Pattern (SDSP) is applied to get the final motion vector [3].

C.W. Lam et al. proposed **Kite Cross Diamond Search (KCDS)** algorithm in 2004. A Small Cross Search Pattern (SCSP) is applied in the first step. Then, a Kite Search Pattern (KSP) is applied in the second step. In the following steps, LDSP is applied repeatedly until the minimum BDM point is at the center, after that, a SDSP is applied to find the final motion vector. Experimental results show that this KCDS algorithm gives results better than results given by Diamond Search (DS) and CDS. They show that KCDS is a very fast algorithm. This algorithm is used especially for video conferencing applications [4].

The **Cross Diamond Hexagonal Search (CDHS)** algorithm was introduced by C.H. Cheung and L.M. Po in 2005. This algorithm uses Small Cross Shape Patter (SCSP) in the first step and Large Cross Search Pattern (LCSP) in the second step. Then, it switches to diamond search patterns (LDSP and SDSP). To further reduce the checking points, two pairs of hexagonal search patterns (Small Hexagonal Search Pattern (SHSP) and Large Hexagonal Search Pattern (LHSP)) are proposed in conjunction with candidates found located at diamond corners [5].

L. Hao et al. introduced **Diamond Hexagonal Search (DHS)** algorithm in 2006. The algorithm uses three types of search patterns. A big diamond search pattern consisting of 5 points is used in the first step. The directional hexagon (vertical or horizontal) search pattern is applied repeatedly in the next steps to find a small region where the best motion vector is expected to locate. Finally, SDSP is used to find the final MV [6].

The **Cross Hexagonal Search (CHS)** algorithm was proposed by S. Zhu et al. in 2009. In this algorithm, two SCSP are applied in the first two steps. Each step has a halfway stop when the minimum BDM point is at the center. A LCSP is applied in the third step and this step also has halfway stop. Then, a LHSP is applied repeatedly until the minimum BDM point is at the center. The final step applies SHSP to find the final motion vector [7].

A new algorithm was proposed by R. Bali and V.K. Govindan in 2014. The proposed algorithm is obtained by combining **Three Step Search (TSS)** and **Simple and Efficient Search (SES)**. The search window in the reference frame is divided into four quadrants. The SES is used to find the best one of these quadrants, while TSS is used to find the final MV with in the selected quadrant. The experiments show that this algorithm gives better results than both TSS and SES [8].

S.KU. Chhotary et al. proposed a new hybrid algorithm which is a combination of TSS and DS in 2016. The first step of TSS is applied in the initial step. If the minimum BDM point is a) the center, then step 3 of TSS is applied and the search stops. b) One of the corners, then continue performing the same steps of TSS. c) One of the axis, then LDSP is applied followed by applying the last step of TSS [9].

A new algorithm called **Three Step Cross Search (TSCS)** was proposed by R. Bhandari and A. Vyas in 2016. This algorithm consists of three steps. In the first two steps, a cross shape pattern forming (+) is applied with step size (S) equal to 4 and 2 in the first and second steps respectively. In the last step, 8 points surrounding the center point found in the previous step are evaluated to get the final motion vector [10].

S.K. Sahu and D. Shukla proposed a new algorithm called **Star Diamond-Diamond Search (SDDS)** in 2018. In the first step, a star diamond search pattern is applied. This pattern consists of 9 points. In the following steps, a SDSP is applied repeatedly until the minimum BDM point is at the center. Star pattern is applied only once to detect the direction of the motion vector, if it is stationary, vertical, diagonal, or horizontal [11].

3. Block matching algorithms

A number of most popular and used block matching algorithms are described in this section. Brief explanations of these algorithms are given in subsections below.

3.1. Full Search (FS)

Full search (also called Exhaustive Search (ES)) calculates the cost function at each possible location in the search window. FS will check (2w + 1)2 search points to find the solution when the search window range in the reference frame is ±w in both directions. It gives the optimal solution and also gives the highest PSNR. But it is the most computationally expensive algorithm. Its drawback is that more numbers of computations are required when larger search window is used. So it is not good for coding real-time videos [12]. Therefore, many block matching algorithms with different search strategies and patters were proposed to reduce computations and give good results.

3.2. Three Step Search (TSS)

It is one of the oldest fast block matching algorithms. Koga et al. proposed this algorithm in 1981 [13]. It consists of three steps. In each step, it searches at eight points +/- S around the center point in addition to the center point only in the first step, where S is the step size. The point with the least cost is set as the center point for the next step. Step size is first set to four (i.e. S = 4 if search parameter is 8) and is divided by two after each step. After completing the three steps, the MV is set to the position of the minimum BDM point (i.e. the best match) [9].

TSS is based on the assumption of unimodal error surface. This means that when the search goes far away from the global minimum, the error increases monotonically. In fact, this assumption is not always true due to reasons such as the inconsistent block segmentation of moving object and background and the aperture problem, etc. [14] TSS procedure is shown in figure 2.

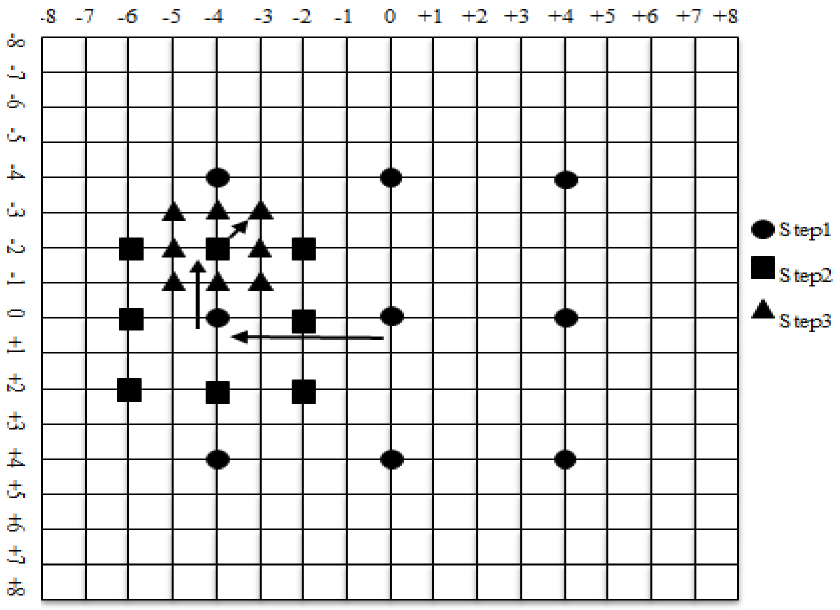


Figure 2. TSS example where the motion vector is (-3,-3).

The advantages of TSS are 1) it is simple. 2) it reduces computations significantly. 3) it has good performance. So, it was used frequently in most applications. The disadvantages of 3SS are 1) it is not suitable for small motions because of using uniform search pattern. 2) it uses large square search pattern in the first step (9x9), so there is high probability that it gets trapped to local minimum.

3.3. Cross Search (CS)

This algorithm was proposed by Ghanbari in 1990 [15]. It starts by using 5x5 search pattern forming a (X) shape and step size = 4 when the maximum motion displacement is 8. In each step, four locations are checked at the end of (X) pattern. The minimum BDM point is set as the center point of the next step and the step size is divided by 2. This procedure is repeated until the step size becomes equal to 1. At this step size, if the minimum BDM point of the previous step is at the center, upper left, or lower right; CS uses a (+) search pattern is used; otherwise, CS uses a (X) search pattern. The BDM point in this step will give the final motion vector [16]. CS procedure is shown in figure 3.

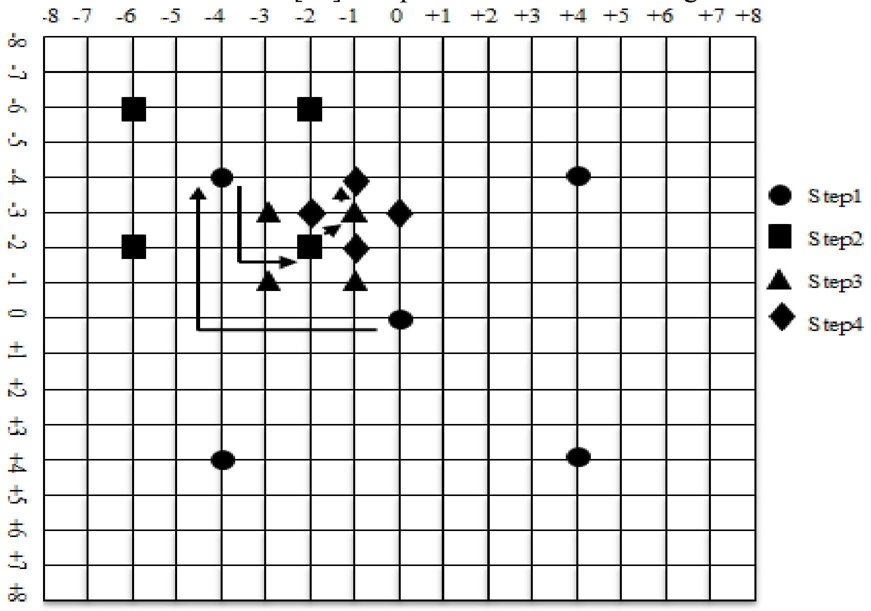


Figure 3. CS example where the motion vector is (-1,-4).

3.4. Four Step Search (FSS)

This algorithm was proposed by L.M. Po and W.C. Ma in 1996 [17]. FSS uses square search pattern of size 5 x 5 which consists of nine search points. FSS consists of four steps. The step size is not changed during the first three steps and is set to 2 (i.e. S = 2). In the fourth step the window size is reduced to 3x3, it means that, S = 1. The value of search parameter p is not important. In first step, FSS checks 8 points at distance +/-S around the center point in addition to the center point (i.e. 9 points). In second and third steps, three or five points are checked according to position of least weight point in the previous step (i.e. there is overlapping). In the fourth step, 8 points at distance +/-S around the center point are checked. In steps one and two, the algorithm skips third step and immediately executes fourth step, if the center point is the point with least weight and the final MV is found [12]. FSS procedure is shown in figure 4.

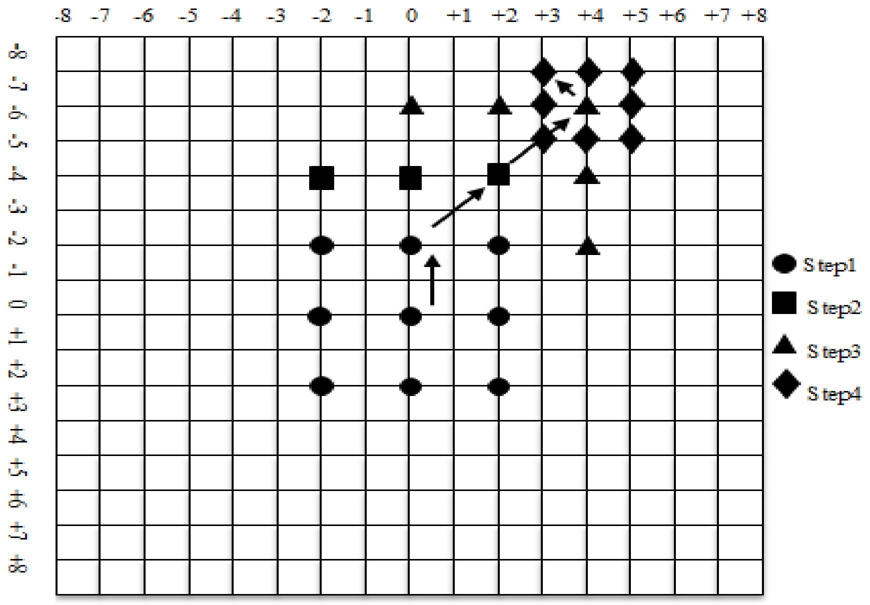


Figure 4. FSS example where the motion vector is (+3,-7).

3.5. Diamond Search (DS)

S. Zhu and K.K. Ma introduced this algorithm in 2000 [18]. DS is similar to FSS, but it uses diamond search pattern instead of square search pattern and the number of steps in the algorithm is not restricted. DS uses two different search patterns, LDSP consisting of 9 checking points as shown in figure 5a and SDSP consisting of 5 checking points as shown in figure 5b.

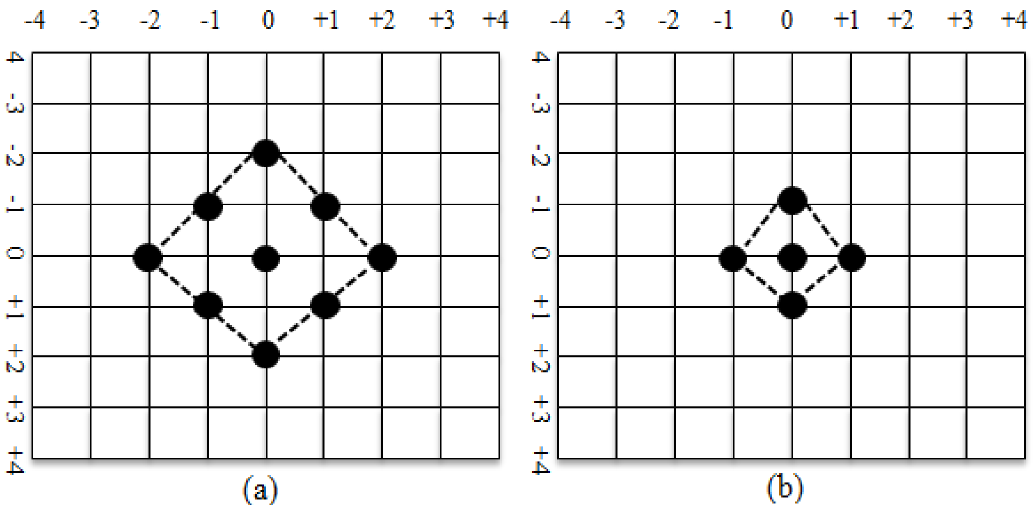


Figure 5. Diamond search patterns; (a) Large Diamond Search Pattern (LDSP), (b) Small Diamond Search Pattern (SDSP).

In first step, LDSP is applied. In second step, SDSP is applied if the center point is the minimum BDM point search stops; otherwise LDSP is applied recursively until the center point is the minimum BDM point. After that, SDSP is applied and the final MV is found. When applying LDSP repeatedly, there will be overlapped checking points (only 3or 5 new points are checked) [19]. DS procedure is shown in figure 6.

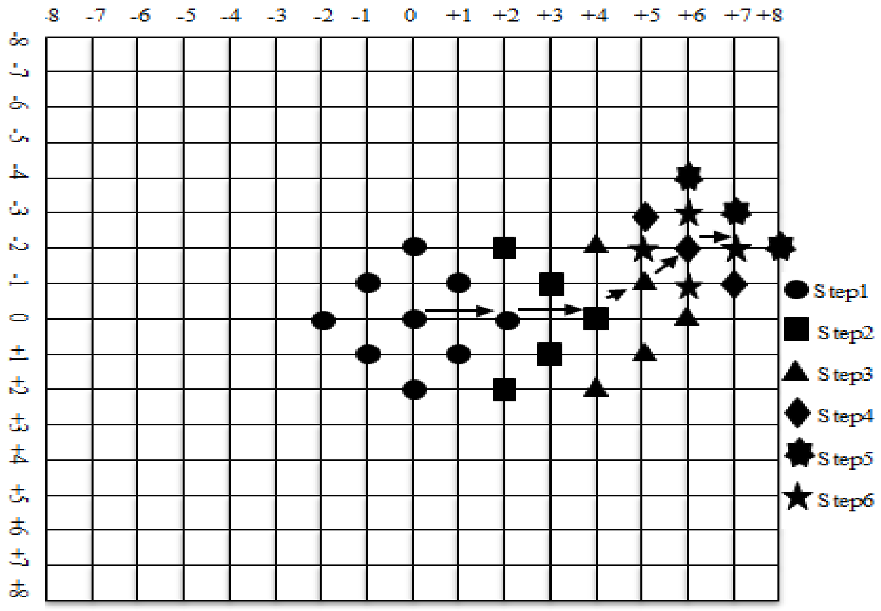


Figure 6. DS example where the motion vector is (+7,-2).

The advantages of DS are 1) it uses diamond search pattern, which is better than rectangular search pattern. The search pattern has moderate size. So, global minimum can be found accurately [12]. 2) the probability of being trapped in local minima is minimized because DS uses infinite number of steps until finding final solution.

3.6. Hexagonal Search (HS)

C. Zhu et al. proposed HS in 2002 [19]. HS uses two different hexagonal search patterns, The Large Hexagonal Search Pattern (LHSP) as shown in figure 7a (seven points), and The Small Hexagonal Search Pattern (SHSP) as shown in figure 7b (five points).

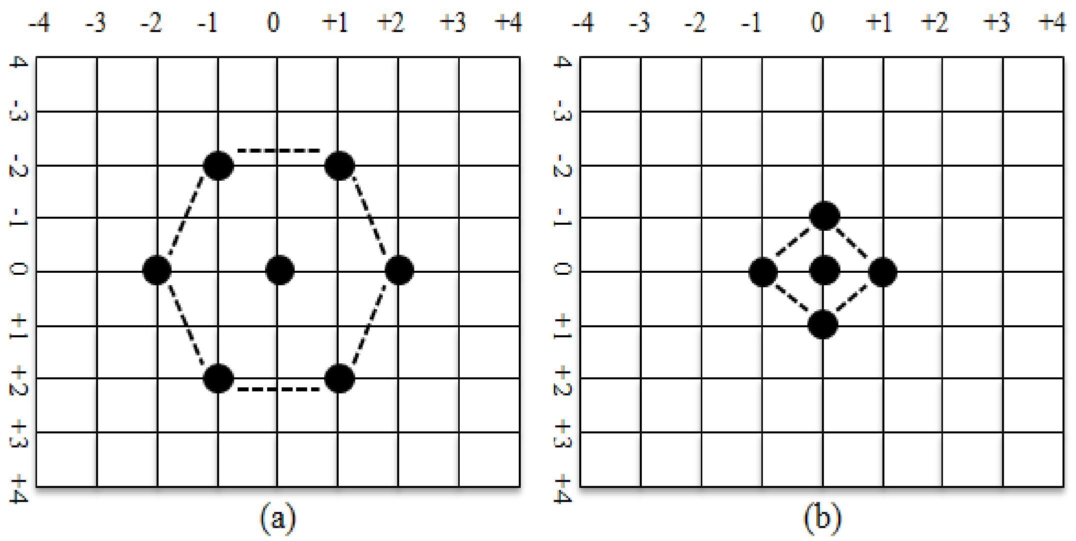


Figure 7. Hexagonal search patterns; (a) Large Hexagonal Search Pattern (LHSP), (b) Small Hexagonal Search Pattern (SHSP).

In first step, LHSP is applied. In the second step, SHSP is applied if the center point is the minimum BDM point and search stops; otherwise LHSP is applied recursively until the center point is the minimum BDM point. After that, SHSP is applied and the final MV is found. When applying LHSP repeatedly, there will be overlapped checking points (only 3 new points are checked) [19]. HS procedure is shown in figure 8.

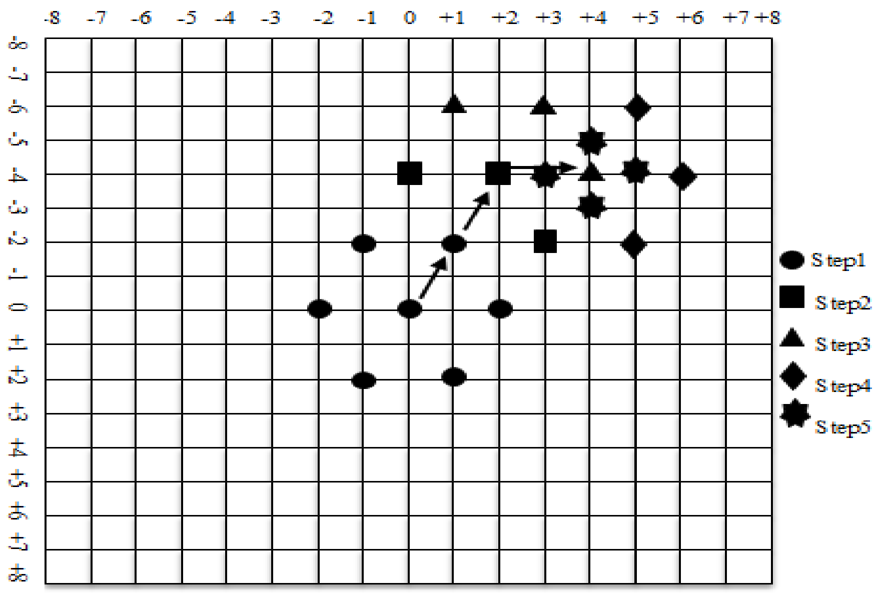


Figure 8. HS example where the motion vector is (+4,-4).

The advantages of HS are 1) it uses more compact pattern than diamond and square patterns, so all directions are considered to find the MV. 2) it checks only seven searching points in the first step which makes it faster. 3) in the second step and the following steps except last step, it checks only 3 points because the other points are overlapped with the previous step (faster than DS which checks 3 or 5 points in these steps). The disadvantage of HS is that using search pattern with few points and large size will make this algorithm sometimes trapped into local minimum point.

3.7. Flat Hexagonal Search (FHS)

This algorithm was introduced by C.H. Chen and Y.F. Li in 2004 [20]. The FHS uses Flatted Hexagonal Search Pattern (FHSP) as shown in figure 9. In first step, FHSP is applied. In second step, SHSP is applied if the center point is the minimum BDM point and the search stops; otherwise, FHSP is applied recursively until the center point is the minimum BDM point. Finally, SHSP is applied and the final MV is found. When applying FHSP repeatedly, there will be overlapped checking points [21]. FHS procedure is shown in figure 10.

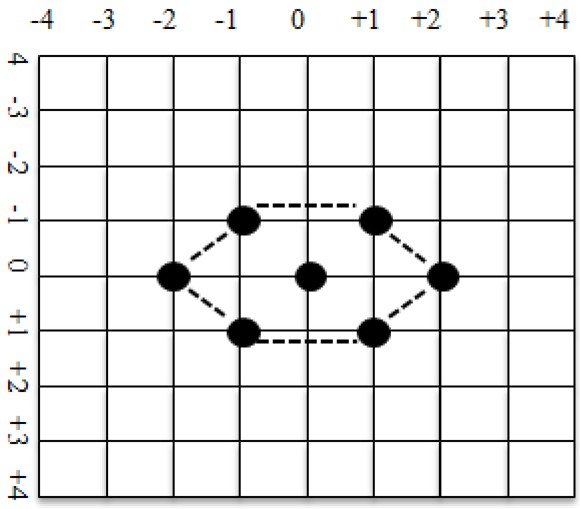


Figure 9. Flat Hexagonal Search Pattern (FHSP).

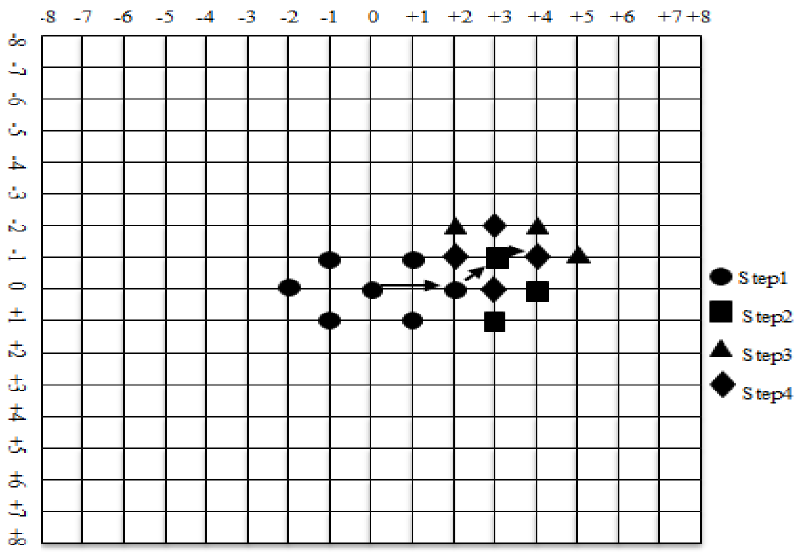


Figure 10. FHS example where the motion vector is (+4,-1).

4. Kite Cross Hexagonal Search (KCHS) algorithm

Three search patterns are used in KCHS algorithm: Small Cross Search Pattern (SCSP), Small/Large Hexagonal Search Patterns (SHSP/LHSP) as shown in figure 7 above, and Kite Search Patterns KSP (up, down, left, and right) as shown in figure 11.

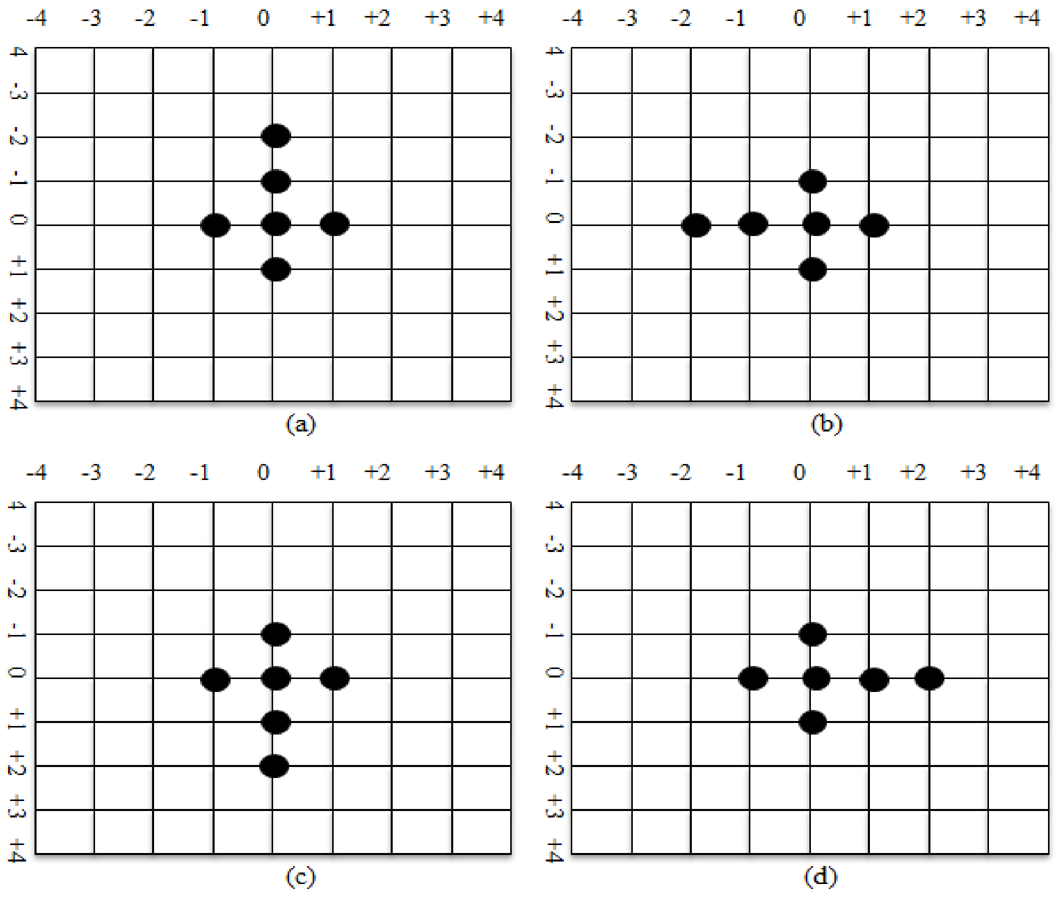


Figure 11. Kite search patterns; (a) Up-kite, (b) Left-kite, (c) Down-kite, (d) Right-kite.

The details of KCHS algorithm are given in Algorithm 1.

Algorithm 1: Kite Cross Hexagonal Search (KCHS).

Input	Reference frame, Current frame.
Output	Motion vector of current frame.

Begin

Step 1 (SCSP): SCSP is applied at the center of the search window (5 points). If the minimum BDM point is the center point, then the search stops; otherwise, go to Step 2.

Step 2 (KSP): One of the four types of KSP is applied depending on the position of the minimum BDM point found in Step 1 which will be considered as the new center point for KSP. If the minimum BDM point is the center point, the search stops; otherwise go to Step 3.

Step 3 (Hexagonal Searching): A new LHSP is applied considering the minimum BDM of Step 2 as the center point. If the new minimum BDM point is the center point, then go to Step4; otherwise repeat this step.

Step 4 (Ending – Converging step): SHSP is applied and the minimum BDM point is found. This point is considered as the motion estimation solution.

End

KCHS requires 5 search points for stationary block and 9 search points for quasi-stationary block, whereas the DS requires 13 search points, and HS requires 11 search points for both stationary and quasi-stationary blocks. KCHS procedure is shown in figure 12.

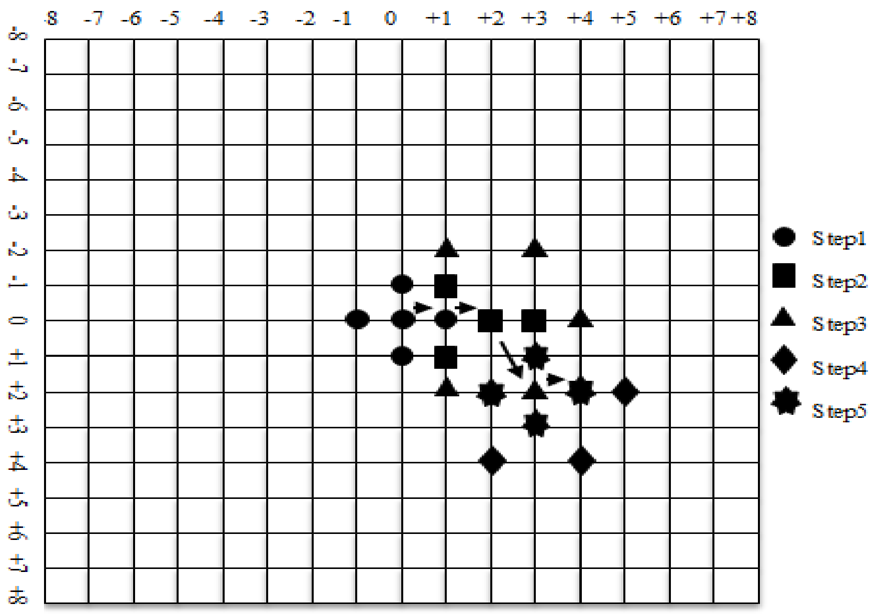


Figure 12. KCHS example where the motion vector is (+4,+2).

5. Simulations and results

In this paper, experiments are executed using MATLAB 2018a program. The computer used here has the following specifications: Intel® Core™ i3 CPU 3217U @ 1.8G processor, 3 MB Smart Cache, 1 GB video card memory, and 64bit Windows®7 ultimate.

Parameters used in this simulation are given in table 1. The first 30 frames of each video sequence are used in simulation. The comparison between BM algorithms is done based on the value of search points required to find the motion vector for each frame, and the value of PSNR between original frames and reconstructed frames. Each video sequence is processed by eight fast BM algorithms: FS, FSS, TSS, HS, DS, CS, FHS, and the new proposed Kite Cross Hexagonal Search (KCHS). The simulation results are shown in Tables 2 and 3 and Figures 13-17.

Table 1. Parameters Used in Simulation.

Block Size	**8 x 8**			
Search Parameter	8			
Cost Function	Mean Absolute Difference (MAD)			
Video sequence	**Format**	**Frame size**	**Motion type**	**Total frames**
Akiyo	Cif	288 x 352	Small	288
Football	Cif	240 x 352	Large	348

Figure 13. Video sequences applied in simulation; (a) Akiyo, (b) Football.

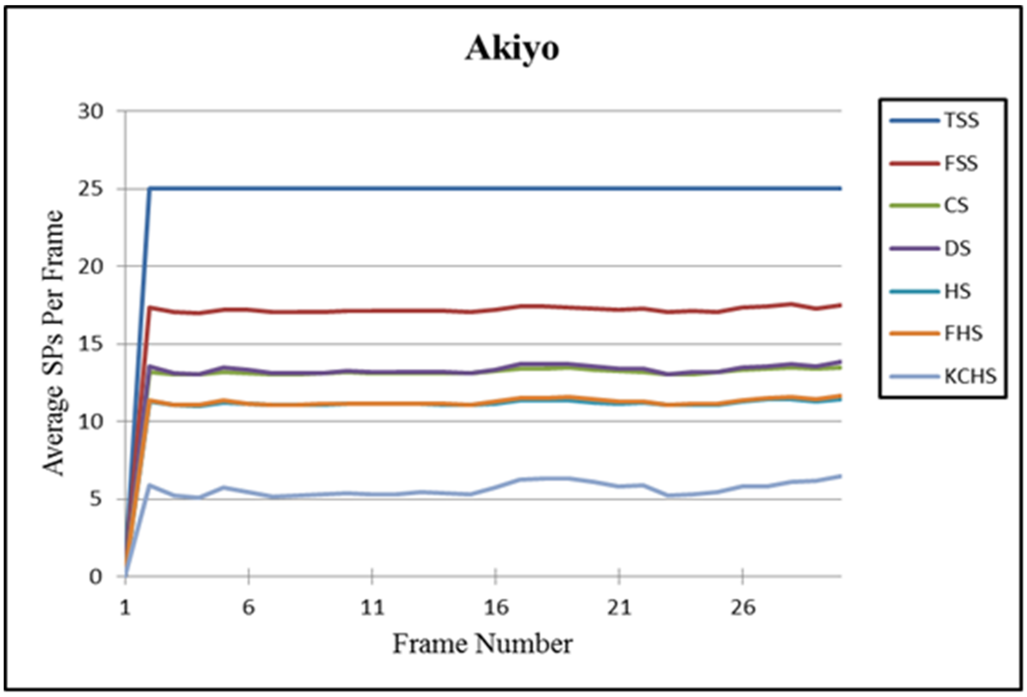


Figure 14. comparison of SP/FR (search point/frame) for different BM algorithms on Akiyo video sequence.

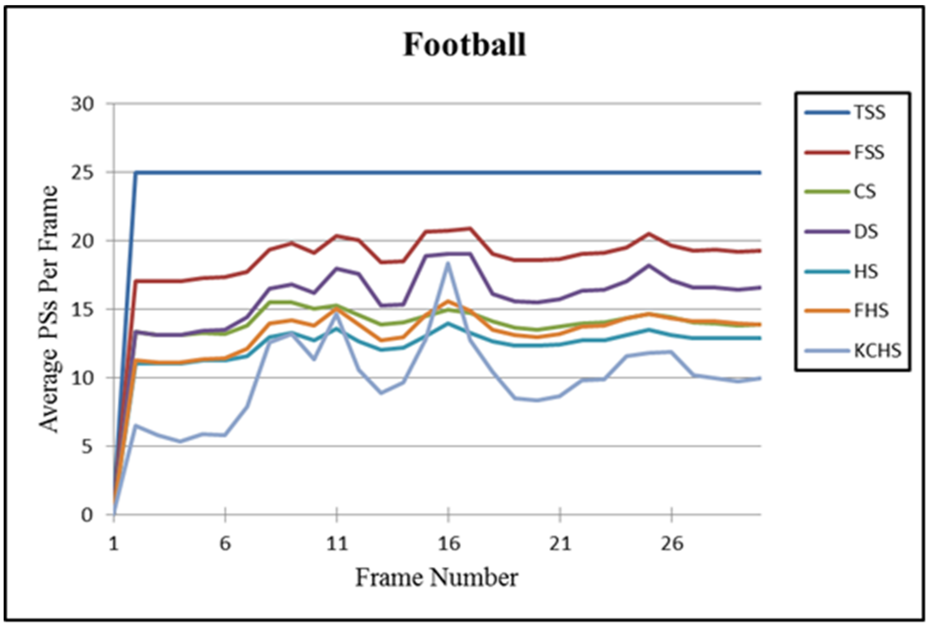


Figure 15. Comparison of SP/FR (search point/frame) for different BM algorithms on Football video sequence.

Table 2. Average SP of Different MB Algorithms Applied on Different Video Sequences.

BM algorithm	**Average SP**	
	Akiyo	Football
FS	289	289
TSS	25	25
FSS	17.2771	20.7094
CS	13.2947	14.6182
DS	13.4385	18.8094
HS	11.2424	13.9708
FHS	11.3337	15.0712
KCHS	5.8715	13.354

FS is not included in figures 14 and 15 since it always requires 289 search points per every frame regardless the type of video. Clearly from figures 14 and 15 and table 2, the proposed KCHS has best average SP (i.e. approximately 6 points for video with small motion and approximately 13 points for video with large motion), while TSS needs the highest number of points to be search (i.e. exactly 25 points for both videos).

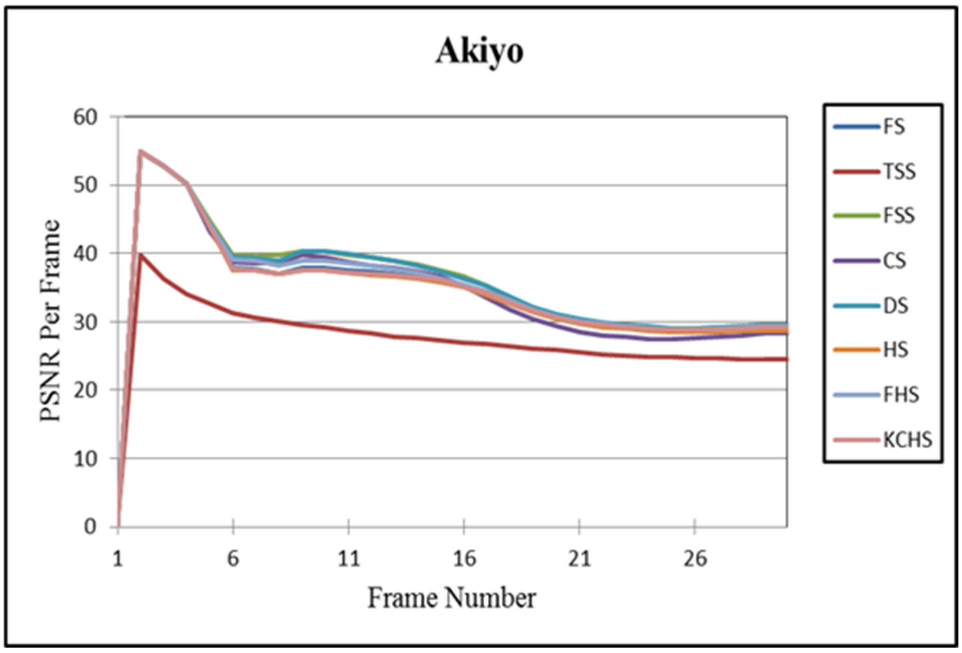


Figure 16. Comparison of PSNR/FR (PSNR/frame) for different BM algorithms on Akiyo video sequence.

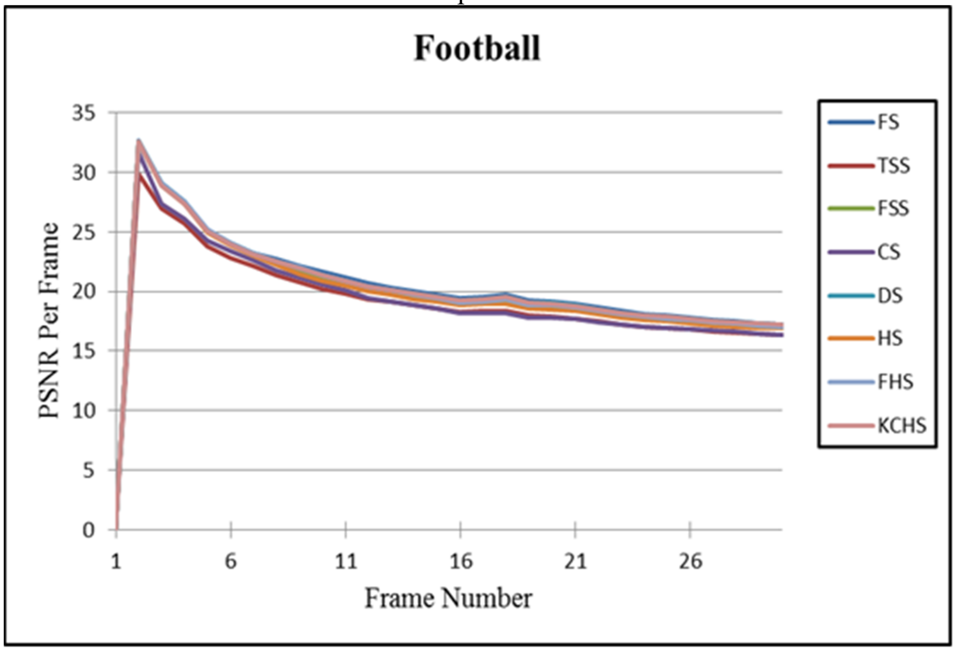


Figure 17. Comparison of PSNR/FR (PSNR/frame) for different BM algorithms on Football video sequence.

Table 3. Average SP of Different MB Algorithms Applied on Different Video Sequences.

BM algorithm	**Average PSNR**	
	Akiyo	Football
FS	25.7137	18.5017
TSS	22.7353	18.3949

FSS	25.6562	18.2435
CS	24.4959	17.9308
DS	25.4761	18.2863
HS	25.4697	18.2744
FHS	25.422	18.17
KCHS	25.4096	18.3662

From figures 16 and 17 and table 3, it is clear that KCHS gives PSNRs closely to the PSNRs given by FS (i.e. 25.4096 for video with small motion, and 18.3662 for video with large motion).

6. Conclusions

In this paper, eight BM algorithms are executed and compared based on two performance measuring parameters: average search points per frame, and PSNR per frame.

This paper shows that the kite Cross Hexagonal Search algorithm (KCHS), which combines kite, cross, and hexagonal patterns is faster than all other BM algorithms used in this paper (requires less search points) regardless the type of video. At the same, it gives results with accuracy similar to the accuracy of results given by other algorithms. It is especially used for video conferencing.

References

[1] R.A. Manap, S.S.S. Ranjit, A.A. Basari, and B.H. Ahmad, 2010. Performance Analysis of Hexagon-Diamond Search Algorithm for Motion Estimation, *IEEE Int. Conf. of Computer Engineering and Technology (ICCET)*, Vol. **3**, pp. 155-159.

[2] A. Barjatya, 2004. Block Matching Algorithms For Motion Estimation, *DIP 6620 Final Project Paper.*

[3] C.H. Cheung, and L.M. Po, 2002. A Novel Cross-Diamond Search Algorithm for Fast Block Motion Estimation, *IEEE Trans. Circuits and Systems For Video Technology*, Vol. **12**, No. 12, pp. 1168-1177.

[4] C.W. Lam, L.M. Po, and C.H. Cheung, 2004. A Novel Kite-Cross-Diamond Search Algorithm for Fast Block Matching Motion Estimation, *IEEE International Symposium on Circuits and Systems*, Vol. **3**, pp. 729–732.

[5] C.H. Cheung, and L.M. Po, 2005. Novel Cross-Diamond-Hexagonal Search Algorithms for Fast Block Motion Estimation, *IEEE Transactions on Multimedia*, Vol. **7**, No. 1, pp. 16-22.

[6] L. Hao, Z. Hang, W. Jun, and C. Jun, 2004. A fast block-matching algorithm based on variable shape search, *Journal of Zhejiang University Science A*, pp.194-198.

[7] S. Zhu, J. Tian, X. Shena, and K. Belloulata, 2009. A Novel Cross-Hexagon Search Algorithm Based on Motion Vector Field Prediction, *IEEE International Symposium on Industrial Electronics (ISIE)*, pp. 1870-1874.

[8] R. Bali, and V.K. Govindan, 2014. An Efficient Block Matching Algorithm for Fast Motion Estimation Using Combined Simple and Efficient Search (SES) and Three Step Search (TSS) Algorithm, *International Journal of Engineering Research & Technology (IJERT)*, Vol. **3**, No. 5, pp. 328-332.

[9] S.KU. Chhotray, D. Kannoujia, and S. KU. Jha, 2016. An Efficient Block Matching Algorithm For Fast Motion Estimation Using Combined Three Step Search And Diamond Search Algorithm, *International Journal of Computer & Communication Technology*, Vol. **5**, No. 3, pp. 83-86.

[10] B. Rahul, and V. Ashutosh, 2016. New Technique of Three Step Search Algorithm used for Motion Estimation in Video Compression, *International Research Journal of Engineering and Technology (IRJET)*, Vol. **3**, No. 5, pp. 760-763.

[11] K.S Satish, and S. Dolly, 2018. Star Diamond-Diamond Search Block Matching Motion Estimation Algorithm for H.264/AVC Video Codec, *International Journal of Innovative*

Research in Computer and Communication Engineering, Vol. **6**, No. 1.

[12] H. Surrah, and M. Haque, 2014. A Comparative Approach for Block Matching Algorithms used for Motion Estimation, *IJCSI*, Vol. **11**, No. 3, pp. 562-568.

[13] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, 1981. Motion compensated inter frame coding for video conferencing, in *Proc. NTC 81*, pp. 961-965.

[14] J. Lu, and M.L. Liou, 1997. A Simple and Efficient Search Algorithm for Block-Matching Motion Estimation, *IEEE Trans. Circuits And Systems For Video Technology*, Vol. **7**, No. 2, pp. 429-433.

[15] M. Ghanbari, 1990. The cross search algorithm for motion estimation, *IEEE Trans. Commun.*, Vol. **COM-38**, pp. 950- 953.

[16] N. Al-Najdawi, M.N. Al-Najdawi, and S. Tedmori, 2014. Employing a Novel Cross-Diamond Search in a Modified Hierarchical Search Motion Estimation Algorithm for Video Compression. *Information Sciences 268*, pp. 425–435.

[17] L.M. Po, and W.C. Ma, 1996. A novel four-step search algorithm for fast block motion estimation, *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. **6**, No. 3, pp. 313-317.

[18] S. Zhu, and K.K. Ma, 2000. A new diamond search algorithm for fast block matching motion estimation, *IEEE Trans. Image Processing*, Vol. **9**, No. 2, pp. 287-290.

[19] C. Zhu, X. Lin , L.P. Chau , K.P. Lim , H.A. Ang, and C.Y. Ong, 2001. A novel hexagon-based search algorithm for fast block motion estimation, *IEEE International Conference Acoustics, Speech, and Signal Processing Proceedings (ICASSP)*, Vol. **3**, pp. 1593-1596.

[20] T.H. Chen, Y.F. Li, 2004. A novel flatted hexagon search pattern for fast block motion estimation, *International Conference on Image Processing (ICIP)*, Vol. **3**, No. 2, pp.1477-1480.

[21] N.M. Khokhar, and W. Harasani, 2015. Performance Analysis of Fast Block Matching Motion Estimation Algorithms, *International Journal of Computer Applications*, Vol. **122**, No. 6.