Engineering and Technology Journal
Journal homepage: engtechjournal.org

Motion Estimation for Gray Level Videos Using Different Block Matching Algorithms

Tareq Z. Hammood [a]*, **Matheel E. Abdulmunim** [b]

[a] Unit of Remote Sensing, College of Science, University of Baghdad, Baghdad, Iraq, tarik_z29@yahoo.com.

[b] Department of Computer Science, University of Technology, Baghdad, Iraq, Dr.MatheelDargazli@uotechnology.edu.iq.

*Corresponding author.

Submitted: 17/04/2020 Accepted: 07/08/2020 Published: 25/03/2021

K E Y W O R D S

Block matching, diamond search, hexagonal search, motion estimation.

A B S T R A C T

Motion Estimation (ME) is a very important operation in video coding. In order to reduce complexity of computations involved in ME and to increase quality of this process, many Block Matching Motion Estimation (BMME) Algorithms are proposed. The aim of this paper is to compare between these algorithms and find the best one. Seven BMME algorithms are used in this paper. The performance of each algorithm is evaluated for different types of motion to determine the best one of these algorithms. The evaluation is based on search points, and Peak Signal to Noise Ratio (PSNR). The simulation shows that Hexagonal Search is faster than all other Block Matching (BM) algorithms used in this paper regardless the type of video because it requires less number of search points to evaluate motion vectors for the video sequence. It requires 11.2424 average search point (SP) for small motions and 13.9708 for fast motions. It also gives a good quality that is close enough to the quality given by Full Search.

How to cite this article: T. Z. Hammod, and M. E. Abdulmunim, "Motion Estimation for Gray Level Videos Using Different Block Matching Algorithms," Engineering and Technology Journal, Vol. 39, Part B, No. 01, pp. 53-66, 2021.
DOI: https://doi.org/10.30684/etj.v39i1B.1684

1. INTRODUCTION

Now a day's videos are used in many applications. Most of these applications are in real time. Digital video compression methods must be fast and give high compassion ratio without degradation the quality of the output videos. The speed of these methods is necessary for many reasons such as large digital video's size, limited transmission channel's bandwidth, and limited device's memory. Each digital video has a sequential number frames/images and each of these frames has a finite number of pixels [1].

Block-matching ME algorithms have been widely used in various video coding standards such as MPEG series and H.26x because of their efficiency, and simplicity to code them. In block matching,

the current frame is divided into equally sized non-overlapped blocks called macro blocks (MBs) of size LxL. Each MB is compared with the MB at the same position and its adjacent neighbors in the reference frame within a search area of size (L+2w x L+2w). Where w is maximum motion displacement allowed. The displacement between the current block in the current frame and the best matched block in the reference frame is called the Motion Vector (MV) of that current block in the current frame. The MV, calculated for all the macro blocks included in a frame, represents the motion estimated in the current frame. A cost function based on a Block Distortion Measure (BDM) is used to determine the best match. Some of cost functions used are Mean Absolute Differences (MAD), Mean Square Error (MSE), and Sum of Absolute Differences (SAD). The basic idea of block matching is shown in figure 1 [2].

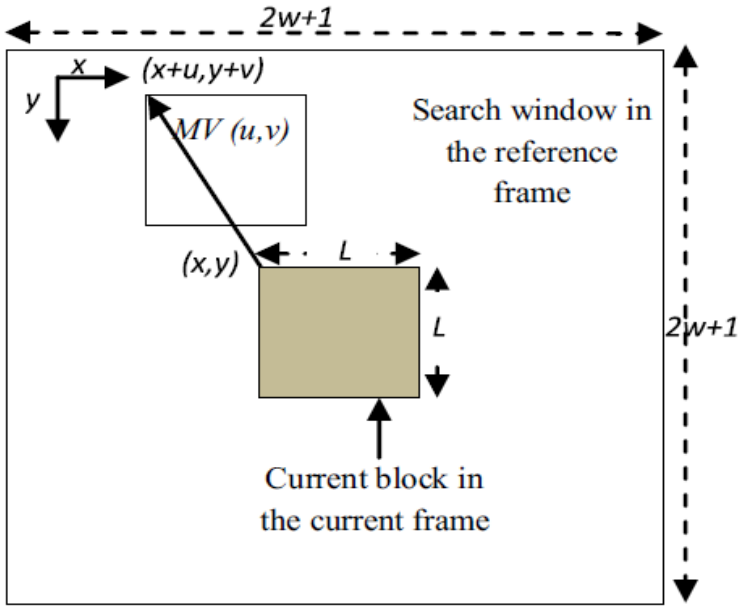


Figure 1: Block matching algorithm

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 results of applying algorithms described in section three 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. J.R. Jain and A.K. Jain proposed **Two Dimensional Logarithmic Search (TDLG)** in 1981. Four Search Points (SPs) in the cross (+) pattern around the center are checked at each step. The step size (p) is set to w/4, where w is the maximum displacement. p is halved when the minimum BDM point is at the center or at the boundary of the search window. When p = 1, eight points surrounding the center are checked. It has good results in the case of large search window [3].

The **One at a Time Search (OTS)** algorithm was introduced by R. Srinvasan and K.R. Rao in 1985. In this algorithm, search is performed first in the horizontal direction using three adjacent searching points. The search direction changes to vertical if the minimum BDM value is found at the central point and the search continues until the minimum BDM value is at the center of the search pattern. Only one checking point is added at each step until finding the solution and this is its strength [4].

A new algorithm called **Orthogonal Search (OS)** was introduced by A. Puri in 1987. It combines both Two Dimensional Logarithmic Search (TDLG) and Three Step Search (TSS) algorithms. It performs vertical search followed by a horizontal search to find optimal block. In the first step, three searching points in horizontal direction, one at the center and two in w/2 distance where w is the maximum displacement are checked. In the next step, two searching points around the minimum BDM point in the vertical direction are checked. The step size is divided by 2 and the procedure is repeated until the step size equals 1 [5].

New Three Step Search (NTSS) is proposed by R. Li et al. in 1994. It was used frequently in oldest standards such as MPEG 1 and H.261. In the first step, 8 points at distance of S = 4 away (similar to TSS) from the center point and 8 points at distance of S = 1 away from the center point and the center point (17 points) are checked. If the minimum BDM point is:

- At the center point, the motion vector is set to (0, 0) and the search stops.
- At point at distance of S = 1, set this point as the center point and check points adjacent to it to find the minimum BDM point among them. In this step, there is overlapping (only three or five points are checked). The motion vector is the position of the minimum BDM point.
- At point at distance of S = 4, then TSS algorithm is applied.

NTSS requires 17 points at minimum and 33 points at maximum to check every macro block. So this algorithm gives good results when detecting small motion, but it is not suitable for large motion [6]. A new algorithm called **Binary Search (BS)** was introduced by T. Urabe et al. in 1994. In this algorithm, the search window is divided into a number of regions. Then, a full search is performed in only one region. In the first step, 9 pixels, one at the center, four at the corners of the search window, and four at the boundaries of the search window are checked. Dividing the search window into regions is based on these points. In the second step, the region containing the minimum BDM point is chosen and a full search is performed in this region. If minimum BDM point is located at [7]:

- Center of the search window (worst case), then BS will check 33 points.
- One of boundary points (average case), then BS will check 23 points.
- One of corners (best case), then BS will check only 17 points.

The **Spiral Search (SS)** algorithm was proposed by T. Zahariadis and D. Kalivas in 1995. It combines the ideas of the TSS and BS algorithms. It consists of three steps. During these steps, it searches for minimum BDM points using a spiral path. Advantages of SS are 1) it reduces searching points without leaving out zones of pixels where the matching criterion is not evaluated, and 2) it reduces computations [8].

L.K. Liu and E. Feig proposed **Block-Based Descent Search (BBGDS)**. It uses 3×3 search pattern at each step and the number of search steps is unrestricted. It checks points in the search pattern and finds the minimum BDM point. Then, it uses the gradient descent direction to determine the search direction and the position of new checking block [9].

The **Simple and Efficient Search (SES)** algorithm [10] was introduced by J. Lu and M.L. Liou in 1997. SES assumes unimodal error surface. In unimodal surface, it is impossible to have two minimums in opposite directions. Therefore, it reduces number of search points (8 points) used in the search pattern of TSS and as a result reducing computations. The algorithm consists of three steps and these steps are similar to the steps of TSS except that each step has further two phases. In the first phase, the search area is divided into four quadrants and the algorithm selects one of them depending on certain weight. In the second phase, the best location in that quadrant is set as origin. The step size is changed in away similar to TSS [11].

Y. Nie and K.K. Ma proposed **Adaptive Road Pattern Search (ARPS)** in 2002. It consists of two sequential search stages: 1) initial search and 2) refined local search. In the first stage, an **Adaptive Rood Pattern (ARP)** is applied to find a good starting point for the next stage. ARP size for each MB is determined dynamically based on the MVs of the neighboring MBs. This step will reduce searching points and the probability of being trapped into local minimum point. In the second stage, a **Unit-size Rood Pattern (URP)** is applied recursively until the final MV is found. **Zero-Motion Prejudgment (ZMP)** can be also used to make the search faster. ZMP is useful when the video sequence has small motion contents [12].

3. BLOCK MATCHING ALGORITHMS

In this section, a number of the most popular and used block matching algorithms are described. Details of these algorithms are given in the following subsections.

I. Exhaustive Search (ES)

Exhaustive search is also known as **Full Search (FS)** calculates the cost function at each possible location in the search window. FS will check $(2w + 1)^2$ search points to find the motion vector when the block size is N × N points and 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 [13].

II. Three Step Search (TSS)

It is one of the oldest algorithms used in block matching and was proposed by Koga et al. in 1981. TSS has been widely used in real time video applications due to its simplicity, significant computational reduction, and good performance [14]. The main steps of this algorithm are given as follows [15]:

1) **Step 1**: Set the center point to the origin of the search window. Nine points at distance of w/2 are checked first, where w is the maximum displacement. The minimum BDM point is set as the center point for the second step.
2) **Step 2**: The step size is halved again and eight points at distance of this new step size are checked. Again, the minimum BDM point is set as the center point for the third step.
3) **Step 3**: The step size is halved again and eight points at distance of this new step size are checked. The position of the minimum BDM point after this step is considered as the final MV.

TSS is based on the assumption that the error surface is unimodal. So it is not a good choice for videos containing small motions [16]. TSS procedure is shown in figure 2.

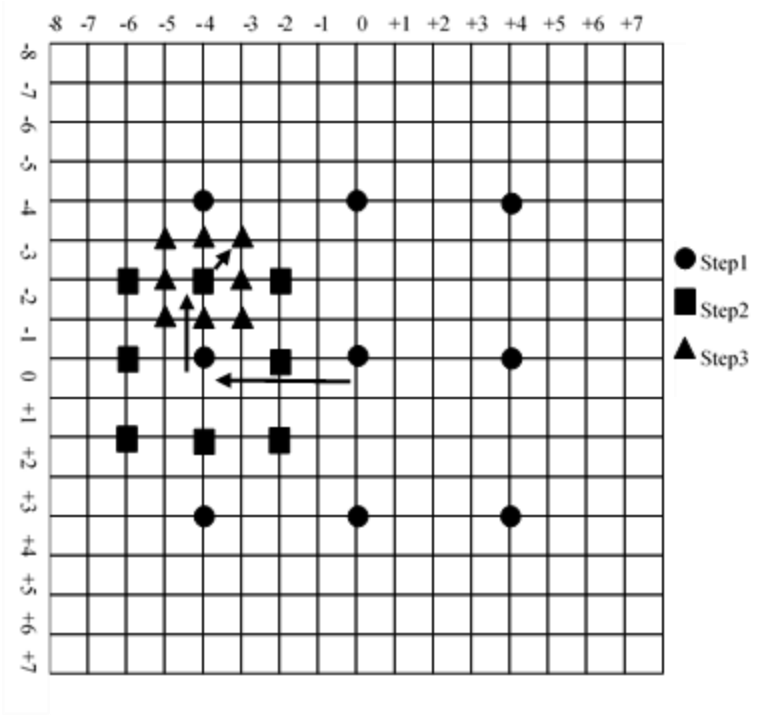


Figure 2: TSS example where the motion vector is (-3,-3).

III. Cross Search (CS)

Ghanbari proposed this algorithm in 1990. This algorithm uses a (X) cross searching pattern in each step. Only four points are checked in each step. Steps of the algorithm are explained as follows [17]:

1) **Step 1**: The current block and the block at the same location in the reference frame are compared. If the difference is less than a predefined threshold T, then the motion vector of the current block is set to (0,0). Otherwise go to Step 2.
2) **Step 2**: Set the step size equals to half the maximum allowable displacement w, of the maximum motion displacement w, i.e., p = w/2.
3) **Step 3**: Four points forming the shape (X) at distance of p from the center point and the center point are checked. Set the minimum BDM point as the new point center and halve the step size. This step is repeated until the step size is 1, i.e. p = 1.
4) **Step 4**: If the minimum BDM point is one of the points forming the shape (\) (three points), then go to Step 5, otherwise go to Step 6.

5) **Step 5**: The end points of a Greek cross (+) are searched considering the minimum BDM in the previous step as the center point.
6) **Step 6**: The end points of a St. Andrew's cross (X) are searched considering the minimum BDM in the previous step as the center point. CS procedure is shown in Figure 3.

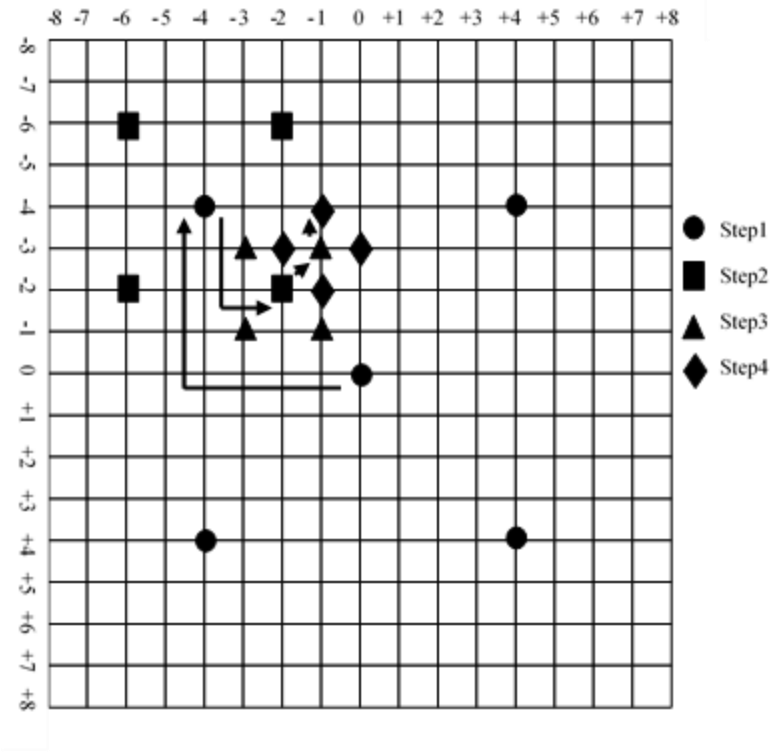


Figure 3: CS example where the motion vector is (-1,-4)

IV. Four Step Search (FSS)

L.M. Po and W.C. Ma introduced this algorithm in 1996. It uses a center-biased search pattern with nine checking points on a 5 x 5 window in the first three steps. Here in these steps, the search step p is fixed and set to 2 (i.e. p = 2) regardless the size of the search parameter (i.e. size of the search window in the reference frame). In Step 4 (final step), FSS uses a search window of size 3 x 3 (i.e. p = 1). FSS steps are summarized as follows [18]:

1) **Step 1**: Set the search step p to 2.
2) **Step 2**: Eight points at distance p from the center point and the center point (5 x 5 window search) are checked. If the minimum BDM point is the center point, go to Step 5; otherwise go to Step 3.
3) **Step 3**: Same as Step 2.
4) **Step 4**: Same as Step 2.
5) **Step 5**: Set p to 1. Eight points at distance p from the center point and the center point (3 x 3 window search) are checked. The MV is setting to the position of the minimum BDM point found in this step.

Overlapping exists in this algorithm. In steps 3 and 4, only three or five points are checked. FSS procedure is shown in figure 4.

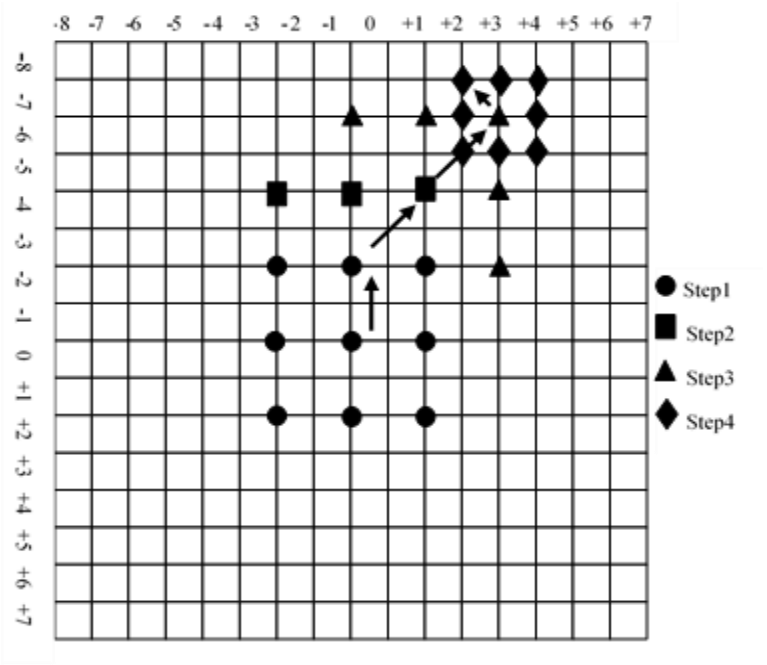


Figure 4: FSS example where the motion vector is (+3,-7)

V. Diamond Search (DS)

This algorithm was proposed by S. Zhu and K.K. Ma in 2000. DS uses two different types of patterns, **Large Diamond Search Pattern (LDSP)** consisting of nine checking points as shown in figure 5a, and **Small Diamond Search Pattern (SDSP)** consisting of five checking points shown in figure 5b [19].

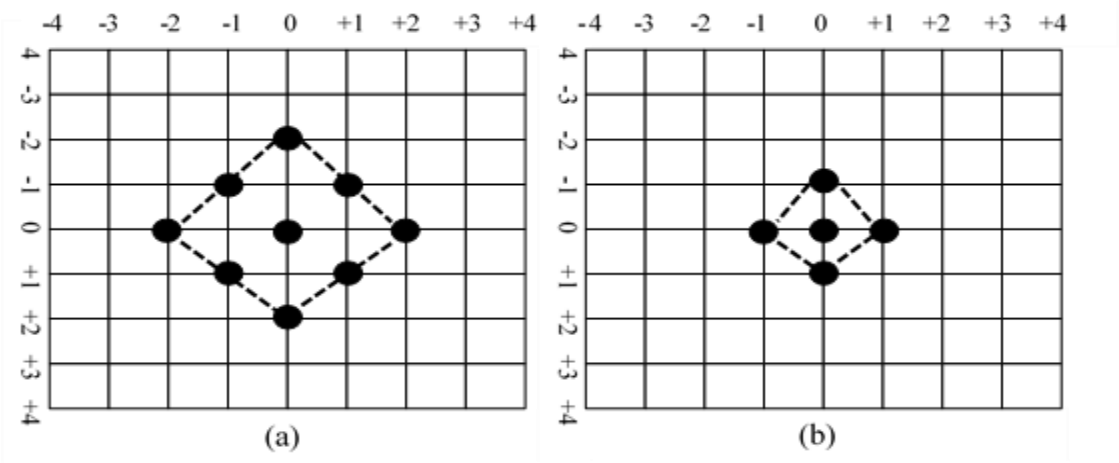


Figure 5: Diamond search patterns; (a) Large Diamond Search Pattern (LDSP), (b) Small Diamond Search Pattern (SDSP)

In DS algorithm, LDSP is repeatedly used until the minimum BDM point is at the center point. Then, SDSP is used as the final step. The MV is setting to the position of the minimum BDM point found using SDSP. The steps of DS are summarized as follows [19]:

1) **Step 1**: Apply LDSP centered at (0,0). Nine points will be checked. If the center point is the minimum BDM point (among these nine points), perform Step 3; otherwise, perform Step 2.
2) **Step 2**: Apply LDSP again considering the minimum BDM point found in the previous search as the center point of the new LDSP. If the minimum BDM point is at the center, go to Step 3; otherwise, repeat this step.
3) **Step 3**: Apply SDSP (four points are checked).The MV is setting to the minimum BDM point found in this step.

There are overlapped checking points when applying LDSP repeatedly (five checking points if the corner point is the minimum BDM point and three checking points if the edge point is the minimum BDM point). Also, there are overlapped points when switching from LDSP to SDSP (only four checking points). DS procedure is shown in figure 6 [19].

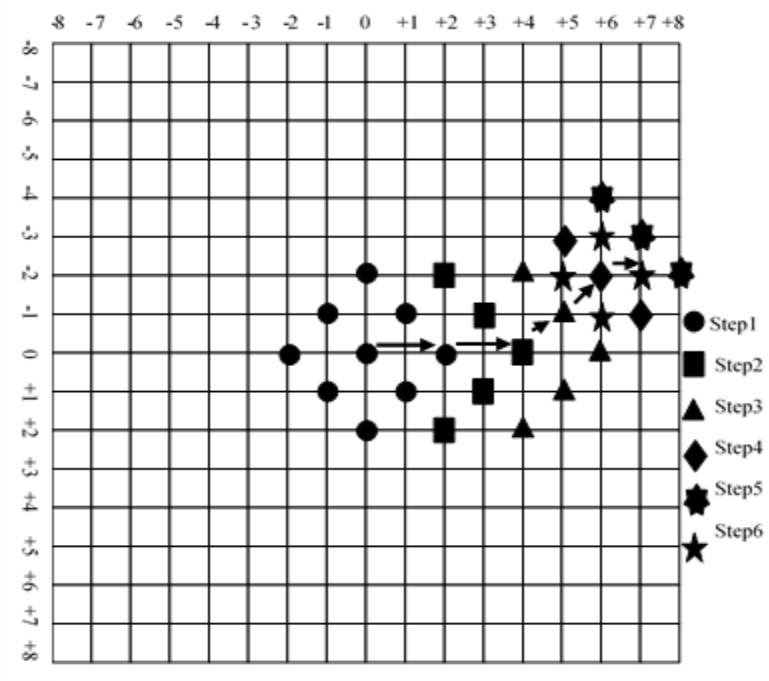


Figure 6: DS example where the motion vector is (+7,-2)

VI. Hexagonal Search (HS)

C. Zhu et al. proposed HS in 2002. HS uses two different hexagonal search patterns, The **Large Hexagonal Search Pattern (LHSP)** consisting of seven checking points as shown in figure 7a, and The **Small Hexagonal Search Pattern (SHSP)** consisting of five checking points as shown in figure 7b. The 6 endpoints in LHSP are approximately uniformly distributed around the center with similar distances to the center. The steps of HS are summarized as follows [20]:

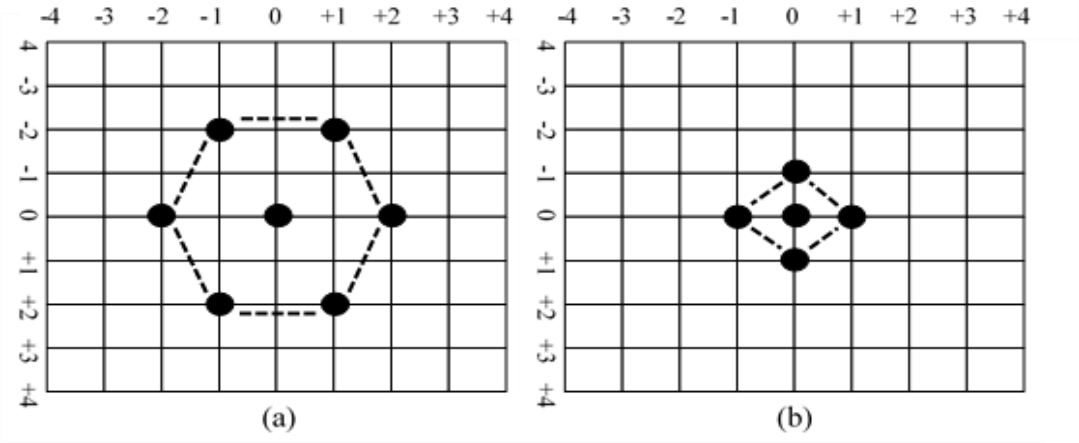


Figure 7: Hexagonal search patterns; (a) Large Hexagonal Search Pattern (LHSP), (b) Small Hexagonal Search Pattern (SHSP)

1) **Step 1 (Starting)**: Apply LHSP centered at (0,0). If the center point is the minimum BDM point, perform Step 3; otherwise, perform Step 2.
2) **Step 2 (Searching)**: Apply LHSP centered at the minimum BDM point of the previous step. Only three new points are checked (overlapping). If the center point is the minimum BDM point, perform Step 3; otherwise, continue repeating this step.
3) **Step 3 (Ending)**: Apply SHSP. The MV is set to the position of the minimum BDM point after checking these four points.

HS procedure is shown in figure 8.

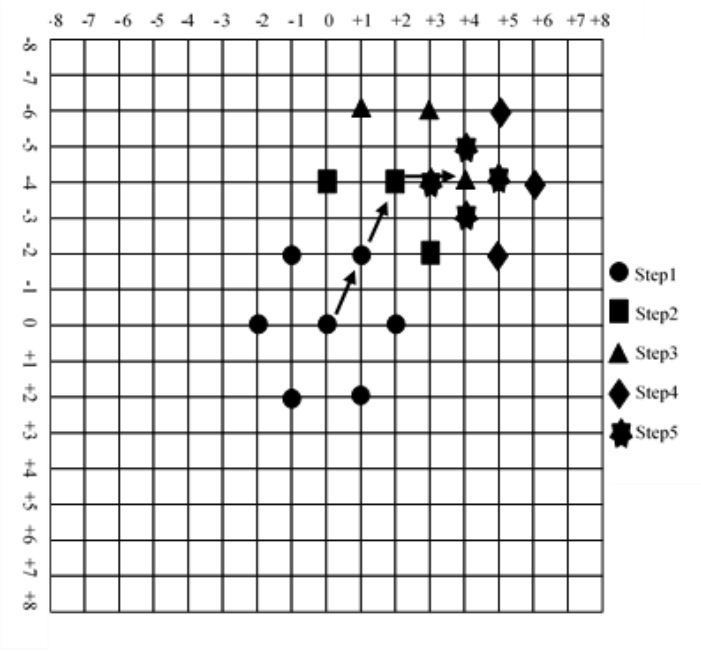


Figure 8: HS example where the motion vector is (+4,-4)

VII. Flat Hexagonal Search (FHS)

This algorithm was introduced by C.H. Chen and Y.F. Li in 2004. Motions in most digital videos are likely to be in the horizontal direction with fewer motions in the vertical direction. So, to find the MV very fast, it is better to enlarge the search pattern in the horizontal direction as possible as can [21].

The flatted-hexagon pattern is a diamond pattern with removing top and bottom checking points. The **Flat Hexagonal Search Pattern (FHSP)** consists of seven checking points as shown in figure 9. Steps of FHS can be explained as follows [21]:

1) **Step 1 (Starting)**: Apply FHSP centered at (0,0). If the center point is the minimum BDM point, perform Step 3; otherwise, perform Step 2.
2) **Step 2 (Searching)**: Apply FHSP centered at the minimum BDM point of the previous step. Only three new points are checked (overlapping). If the center point is the minimum BDM point, perform Step 3; otherwise, continue repeating this step.
3) **Step 3 (Ending)**: Apply a cross pattern centered at the minimum BDM point of the previous step. Four new points are checked and the point with the minimum BDM is set as the final MV.

When FHSP is repeatedly applied in Step 2, there are always only three new endpoints to be checked regardless the position of the minimum BDM point in the previous search step [21]. FHS procedure is shown in figure 10.

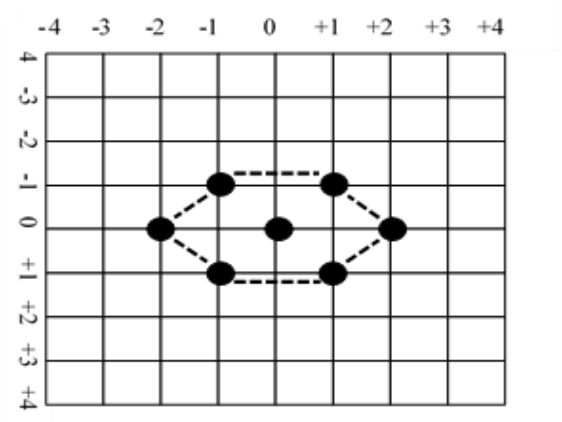


Figure 9: Flat Hexagonal Search Pattern (FHSP)

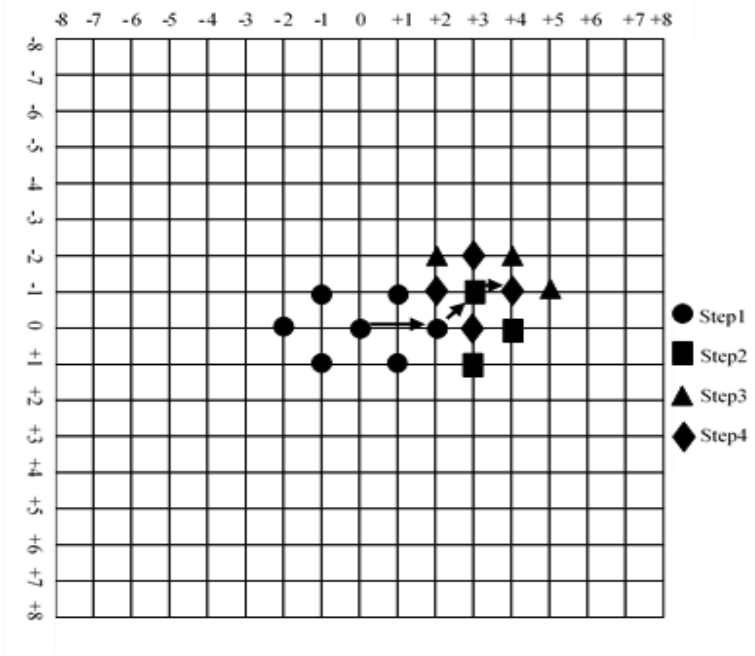


Figure 10: FHS example where the motion vector is (+4,-1)

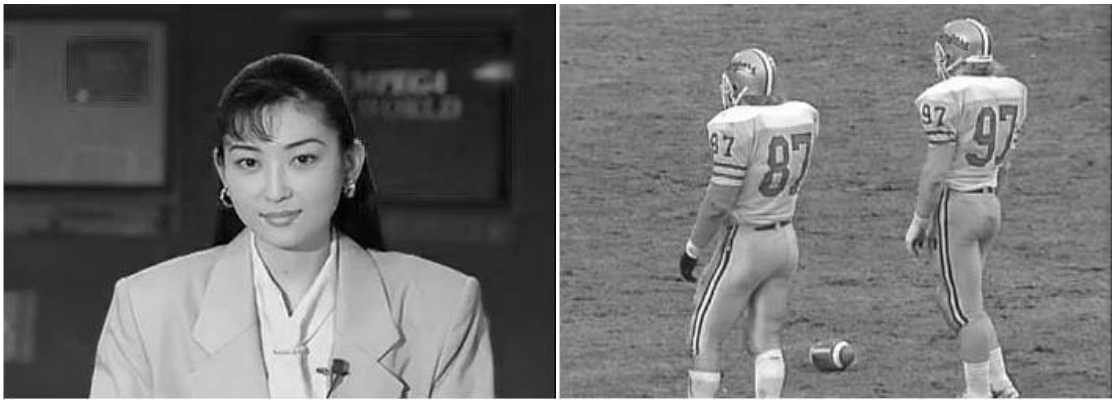
4. SIMULATIONS AND RESULTS

Experimental results are obtained using MATLAB R2018a software and a computer system with the following properties: Intel® Core™ i3 CPU 3217U @ 1.8G processor, 1 MB L2 Cache, 2 GB DDR3 RAM, video card of 512 MB memory and 64bit Windows®7 ultimate operating system.

Parameters used in this simulation are given in Table I. 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 using seven fast BM algorithms: Full Search (FS), Four Step Search (FSS), Three Step Search (TSS), Cross Search (CS), Hexagonal Search (HS), Diamond Search (DS), and Flat Hexagonal Search (FHS). The simulation results are shown in Tables 2 and 3 and Figures 11-13.

TABLE I: 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



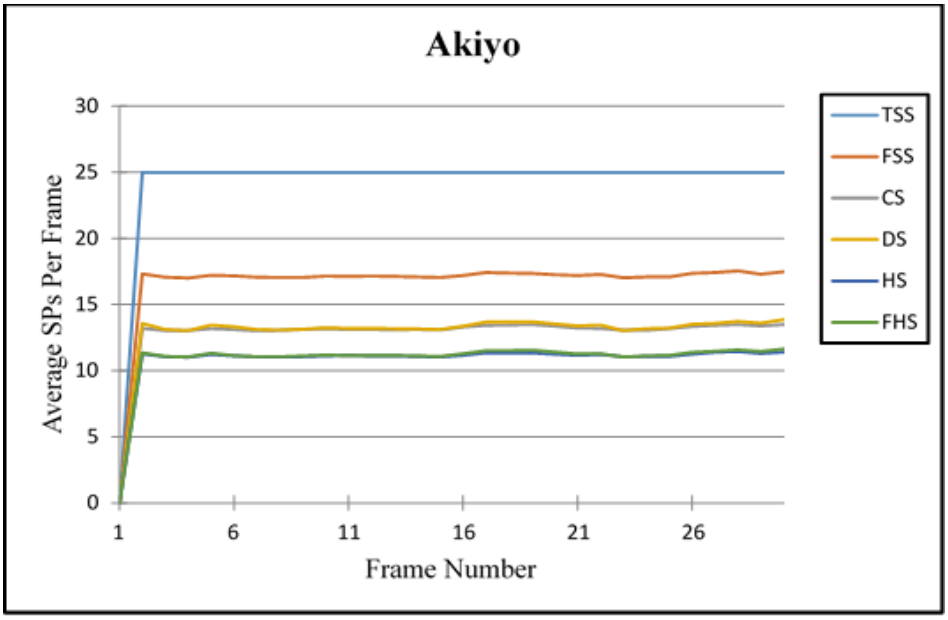
(a) **(b)**

Figure 11: Video sequences used in the simulation; (a) Akiyo, (b) Football

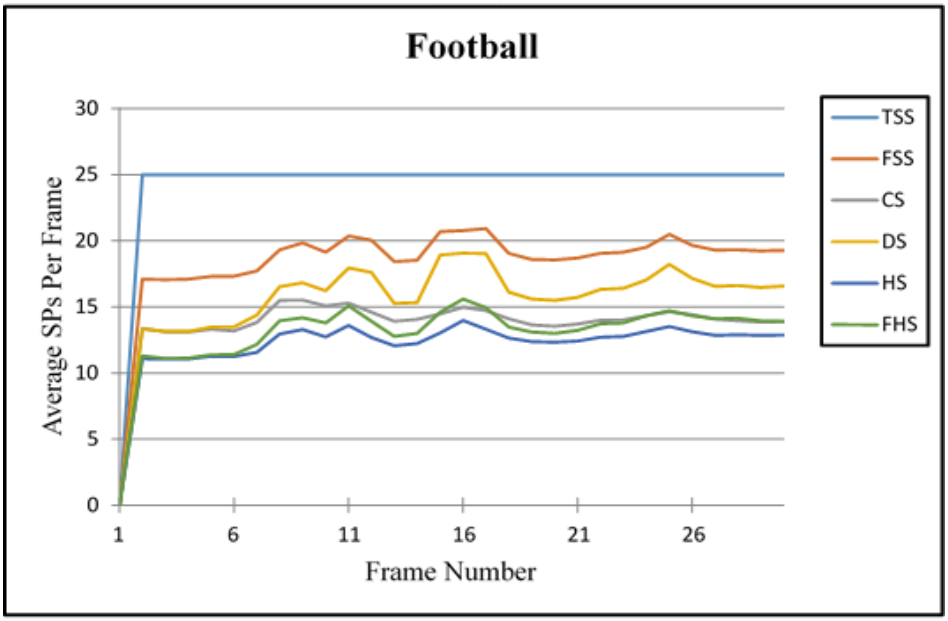
Increasing block size leads to decreasing number of blocks processed in each frame and causing to reduce complexity. On the other hand, when increasing the block size, it is difficult to find the optimal motion vector and hence decreasing the Peak Signal to Noise Ratio (PSNR).

TABLE II: Average SP of different BM 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



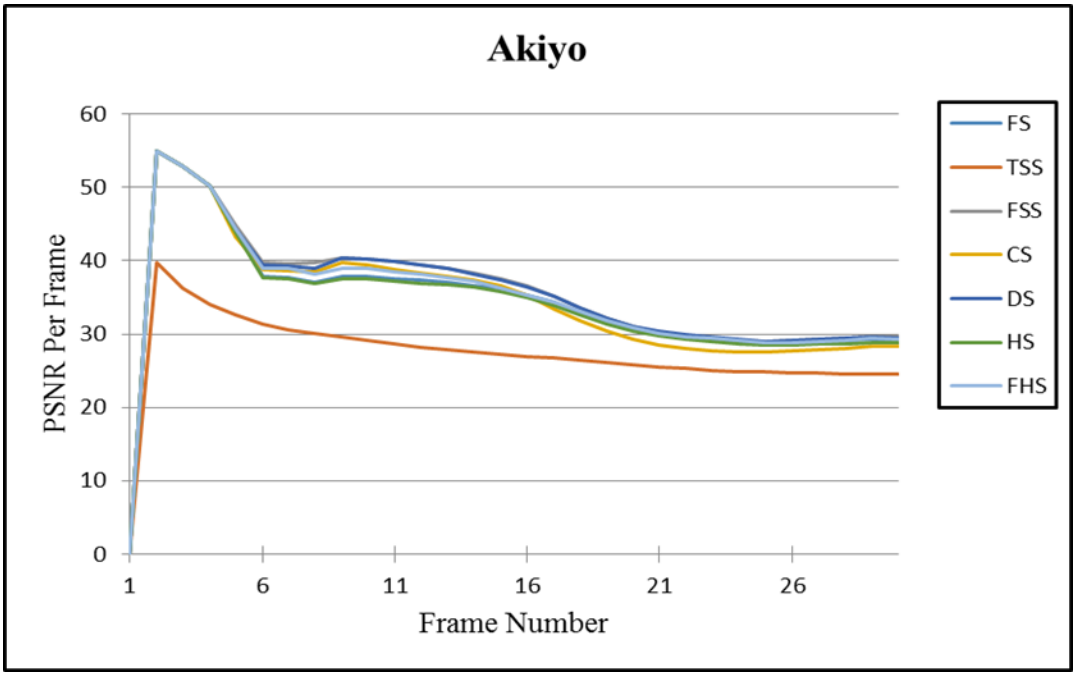
(a)



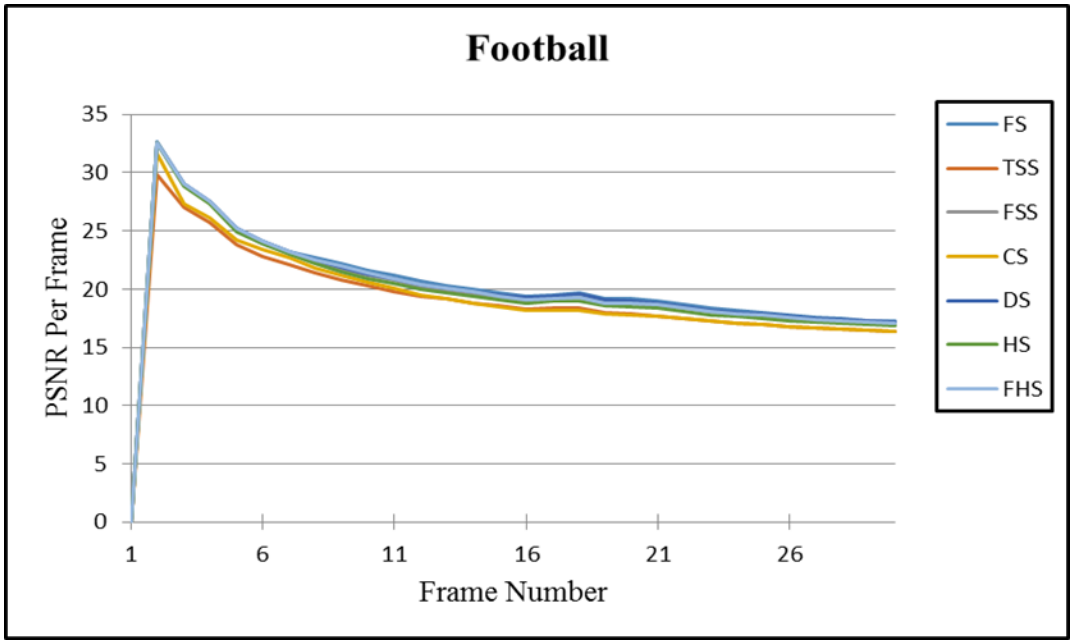
(b

Figure 12: Comparison of SP/FR (search point/frame) for different BM algorithms on (a) Akiyo video sequence, (b) Football video sequence

FS is not included in figure 12 since it requires a fixed number of search points per every frame (i.e. 289 search points) regardless the type of video. From figure 12 and Table II, it is clear that HS needs least number of points to be search (i.e. approximately 11 points for video with small motion and approximately 14 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).



(a)



(b)

Figure 13: Comparison of PSNR/FR (PSNR/frame) for different BM algorithms on (a) Akiyo video sequence, (b) Football video sequence

From figure 13 and Table III, it is clear that HS gives the highest PSNR regardless the type of video because it gives the optimal solution. FSS gives the second highest PSNR (i.e. 25.6562) and TSS gives the worst PSNR (i.e. 22.7353) when the motion in the video is small. TSS gives the second highest PSNR (i.e. 18.3949) and CS gives the worst PSNR (i.e. 17.9308) when the motion in the video is large.

TABLE III: Average PSNR of different BM 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

5. CONCLUSIONS

Experimental results of seven fast BM algorithms are compared in this paper based on two performance measuring parameters: **average search points per frame**, and **PSNR per frame**. These results show that **Hexagonal Search** is faster than all other BM algorithms used in this paper regardless the type of video because it requires less number of search points to evaluate motion vectors for the video sequence. It also gives a good quality that is close enough to the quality given by Full Search.

REFERENCES

[1] P.N. Korat, and D.R. Bhojani, "A New Hybrid Block Based Motion Estimation Algorithm for Video Compression", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (IJAREEIE), Vol. 3, No. 5, pp. 9586-9596, May 2014.

[2] N.K. Nakum, and A.M.Kothari, "A Review paper on Implementation & Comparative Analysis of Motion Estimation Algorithm in Video Compression", International Journal of Recent Technology and Engineering (IJRTE), Vol. 1, No. 5, pp. 57-60, November 2012.

[3] J. Iain, and A. Jain, "Displacement measurement and its application in inter frame image coding", ZEEE Trans. Commun., Vol. COM-29, pp. 1799-1808, December 1981.

[4] **[**R. Srinivasan, and K. Rao, "Predictive coding based on efficient motion estimation", IEEE Trans. Commun., Vol. COM-33, pp. 888-896, August 1985.

[5] A. Puri, H.M. Hang, and D.L. Schilling, "An efficient block matching algorithm for motion compensated coding", Proc. IEEE Int. Conf. Acoust., Speech, and Signal Proc., pp. 1063- 1066, 1987.

[6] R. Li, B. Zeng, and M.L. Liou, "A new three step search algorithm for block motion estimation", IEEE Trans. on Circuits and Systems for Video Technology, Vol. 4, No. 4, pp. 438-442, August 1994.

[7] T. Urabe, H. Afzal, G. Ho, P. Pancha, and M. El Zarki, "MPEGTooI: an X window-based MPEG encoder and statistics tool", Multimedia Systems, Vol. 1, pp. 220-229, 1994.

[8] T. Zahariadis and D. Kalivas, "A spiral search algorithm for fast estimation of block motion vectors", in Proc. EUSIPCO '96, vol. 2, pp. 1079-1082, Trieste, Italy, 1996.

[9] L.-K Liu, and E. Feig, "A block based gradient descent search algorithm for block motion estimation in video coding", IEEE Trans. on Circuits and Systems for Video Technology, Vol. 6, No. 4, pp. 419-422, August 1996.

[10] J. Lu, and M.L. Liou, "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, April 1997.

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

[12] Y. Nie, and K.K. Ma, "Adaptive Rood Pattern Search for Fast Block-Matching Motion Estimation", IEEE Trans. Image Processing, Vol. 11, No. 12, pp. 1442-1448, December 2002.

[13] A.P. Chauhan, R.R. Parmar, S.K. Parmar, and S.G. Chauhan, "Comparative analysis on the performance of block matching motion estimation algorithm", Journal of Information, Knowledge and research in Computer engineering, Vol. 2, No. 2, pp. 366-370, November 2012.

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

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

[16] S.KU. Chhotray, D. Kannoujia, and S. KU. Jha, "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, 2016.

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

[18] L.M. Po, and W.C. Ma, "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, June 1996.

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

[20] C. Zhu, X. Lin , L.P. Chau , K.P. Lim , H.A. Ang, and C.Y. Ong, "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, May 2001.

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