© 2014 IJIRT | Volume 1 Issue 7 | ISSN: 2349-6002

Survey on Block Matching Algorithms for Motion Estimtion

Pooja Chaudhri[1], Ompriya Kale[2]
[1]*Computer Department, L.J. Institute of Engineering and Technology, Ahmedabad, India*
[2]*Computer Dept., L.J. Institute of Engineering and Technology, Ahmedabad, India*

***Abstract*— Motion estimation is basic problem in video processing. Block matching algorithms are very popular to find motion estimation. This paper is review of different Block Matching Algorithms used for video compression. It compares different methods. Algorithms described in this paper are widely used in different video compression standards like Mpeg and H.26X**

***Index Terms*—video compression; motion vectors; motion estimation; block matching**

I. INTRODUCTION

Video has become essential source of entertainment and part of our life. For the transmission of video and efficient storage of video in less memory, video compression is essential. For example if you are transmitting video over the internet, it is nearly impossible to transmit video in uncompressed format because of limited amount of available bandwidth. [1]

Video compression is needed in Internet Video streaming and TV broadcast where multimedia signals are transmitted over a fixed amount of limited bandwidth channels. Video compression is also useful when multimedia data is stored in memory devices like video-CDs or hard disk, which has limited storage capability. In case video is stored in CD without compression, it will barely store 5 to 8 minutes of video. In another example of, PAL system with standard TV resolution 720x576 pixels, and 8 bit RGB color scheme, the frame rate is 30 frames/second, and the data rate is 720x576x30x8x3=284 Mb/s. For HDTV with 1920x1080 pixels/frame, the data rate is 2.78 Gb/s. Such bandwidth and storage are not possible even with current powerful computer systems[1]

Basic flow of video compression is shown in fig 1. A motion compensated image for current frame is estimated. The motion vectors for block used for motion estimation are transmitted. The encoding side estimates motion vectors for current frame with respect to reference frame. A motion compensated image is then created for the current frame. The motion vectors calculated earlier and the difference of motion compensated image and current frame is JPEG encoded and transmitted. The encoded image is decoded by decoder and full frame is created.[2]

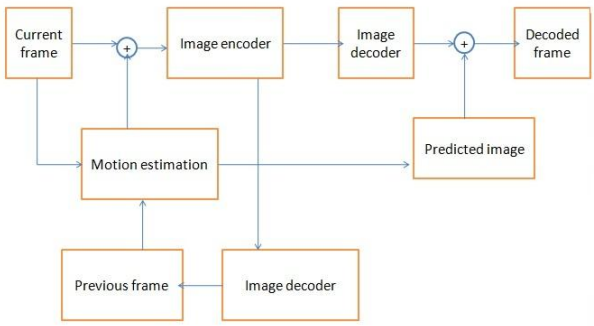

Fig 1: Block diagram of video compression scheme[1]

The most computationally expensive and resource hungry operation in the entire compression process is motion estimation. Hence, this field has seen the highest activity and research interest in the past two decades. This paper implements and evaluates the fundamental block matching algorithms. The algorithms that have been implemented are Exhaustive Search (ES), Three Step Search (TSS), New Three Step Search (NTSS) and Diamond Search (DS). Section II explains block matching in general and then the above algorithms in detail. Section III compares them and presents some simulation results. Section IV is summary followed by references.

II. BLOCK MATCHING ALGORITHM

Objects in the frame of video sequence move within frame to form corresponding objects in new subsequence. Generally, background of image remains the same. To find matching block, each Block of current frame is compared with past or future frame within a search area. [3][4]

Firstly, current frame is divided into smaller macro blocks, and then they are compared to corresponding macro blocks and its adjacent neighbors in previous or future frame. PSNR is calculated at each position in search area; and macro block that results in least cost is the one that matches the closest to current block. A vector is found which indicates the movement of current block with respect to matching block in future or past frame. This vector is called motion vector. Motion vectors for each and every frame are calculated.

The search area for a good macro block match is constrained up to *p* pixels on all fours sides of the corresponding macro

block in previous frame. This '*p*' is called as the search parameter. Larger motions require a larger *p*, and the larger the search parameter the more computationally expensive the process of motion estimation becomes[2]. Usually the macro block is taken as a square of side 16 pixels, and the search parameter *p* is 7 pixels. The idea is represented in Fig 2. The matching of one macro block with another is based on the output of a cost function. The macro block that results in the least cost is the one that matches the closest to current block. There are various cost functions, of which the most popular and less computationally expensive is Mean Absolute Difference (MAD) given by equation (i). Another cost function is Mean Squared Error (MSE) given by equation (ii)

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

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

Where, N= size of macro block

C_{ij} = pixel into current frame

R_{ij} = pixel into reference frame

Peak-Signal-to-Noise-Ratio (PSNR) given by equation (iii) characterizes the motion compensated image that is created by using motion vectors and macro clocks from the reference frame.

$$\text{PSNR}=10*\log_{10}\left(\frac{\text{peak to peak value of original data}}{MSE}\right)^2 \qquad \text{(iii)}$$

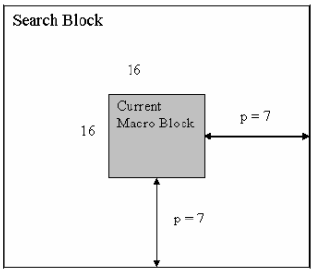

Fig 2: search parameter and search window

A] Exhaustive Search :

This algorithm is the most computationally expensive block matching algorithm[2] but it finds the best possible match and gives the highest PSNR in comparison with other methods. The algorithm measures the cost function at each possible location in the search area. The maximum similarity or minimum dissimilarity gives the best match. It also delivers good accuracy in searching for the best match. But because of a large amount of computation is involved, it is not suitable in real time video coding. The obvious drawback of this method is that the larger the search area gets the more computations it requires.[2][3][4]

B] Three Step Search :

TSS[2][3] became very popular for low bit-rate video applications because of its simplicity and also robust and near optimal performance. It searches for the best motion vectors in a course to fine search pattern. The algorithm may be described as:

Step 1: An initial step size is picked up as 4. Eight blocks at a distance of step size from the centre (around the centre block) are picked for comparison.
Step 2: The step size is reduced by the factor 2. The centre is moved to the point with the minimum distortion.
Step 3: The step size is now 1. The new center is the point with maximum PSNR from step 2.[4][5]

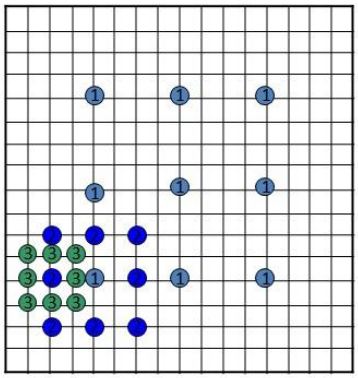

Fig 3 Three step search

C] Four Step Search :

This algorithm also exploits the center-biased property in its search. The computational complexity of the four-step search is less than that of the three step search, while the performance in terms of quality is as good. It is also more robust than the three step search and it maintains its performance for image sequences with complex movements like camera zooming and fast motion[5]. Hence it is a very attractive strategy for motion estimation. The search algorithm is illustrated in Fig.4.

Step 1: Calculate cost at center and 8 surrounding points. If the minimum cost point is found at the center of the search window, proceed to Step 4; otherwise move to Step 2.
Step 2: add search points as per the search pattern. However, the search pattern will depend on the position of the previous minimum cost point.
a) If the previous minimum cost point is located at the middle of horizontal or vertical axis of the previous search window, three additional checking points as shown in Fig. 4 are used.
b) If the previous minimum cost point is located at the corner of the previous search window, five additional checking points as shown in Fig. 4 are used.
If the minimum cost point is found at the center of the search window, go to Step 4; otherwise go to Step 3.

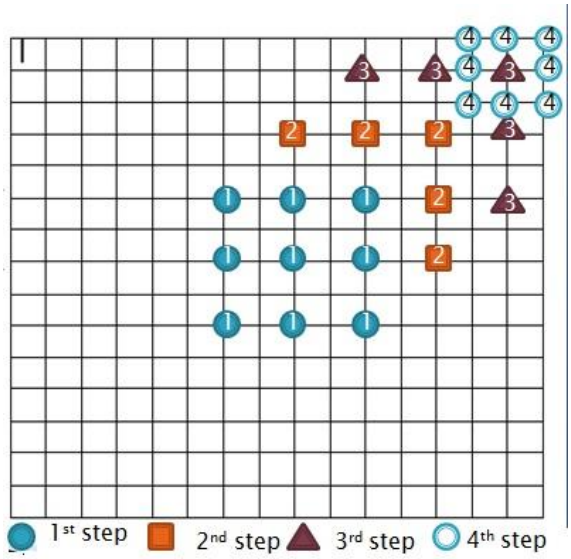

Fig. 4 4SS procedure

Step 3: The searching pattern strategy is the same as Step 2, but finally it will go to Step 4.

Step 4: The search window is reduced to 3 x 3 as shown in Fig. 4 and the direction of the overall motion vector is considered as the minimum cost point among these nine Searching points[5].

Search points needed (considering the displacement parameter p in [-7,7] range)-

Minimum: 9+8 = 17

Worst case: 9+5+5+8= 27

Thus computational complexity is just two block matches more compared to TSS and lesser compared to 33 block matches in NTSS. Also its performance is comparable to that of NTSS.[2][3]

D] Diamond Search :

The DS algorithm [2][3][7] employs two search patterns. The first pattern, called large diamond search pattern (LDSP) shown in Figure 5(a), comprises nine checking points from which eight points surround the center one to compose a diamond shape. The second pattern consisting of five checking points forms a small diamond shape, called small diamond search pattern (SDSP) shown in Figure 5(b). In the searching procedure of the DS algorithm, LDSP is repeatedly used until the minimum block distortion (MBD) occurs at the center point. DS algorithm consistently performs well for the image sequence with wide range of motion content.[7]

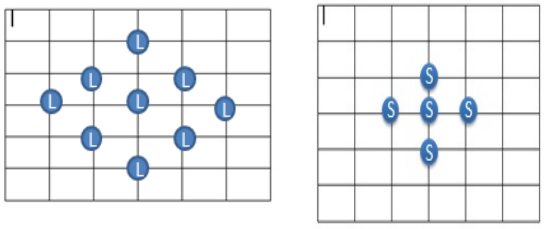

Fig 5(a) large pattern (b) small pattern

In the searching procedure of the DS algorithm, LDSP is repeatedly used until the step in which the minimum block distortion (MBD) occurs at the center point. The search pattern is then switched from LDSP to SDSP as reaching to the final search stage. Among the five checking points in SDSP, the position yielding the MBD provides the motion vector of the best matching block.

The DS algorithm is summarized as follows.

Step (1)The initial LDSP is centered at the origin of the search window, and the 9 checking points of LDSP are tested. If the MBD point calculated is located at the center position, go to **Step 3**; otherwise, go to **Step 2**.

Step (2) The MBD point found in the previous search step is re-positioned as the center point to form a new LDSP. If the new MBD point obtained is located at the center position, go to **Step 3**; otherwise, recursively repeat this step.

Step (3) Switch the search pattern from LDSP to SDSP. The MBD point found in this step is the final solution of the motion vector which points to the best matching block.[6]

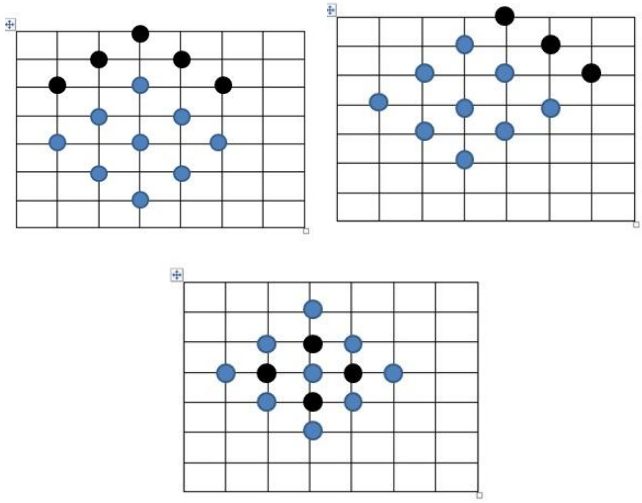

Fig 6(a) 1st case 6(b) 2nd case
6(c) 3rd case

Three cases of checking-point overlapping in LDSP when the MBD point found in the previous search step (shaded dots) is located at (a) one of the corner points, (b) one of the edge points, and (c) the center point. The solid black dots are the new checking points where the computation of block-distortion measurement is required for the current search step.

In the searching procedure of the DS algorithm, LDSP is repeatedly used until the step in which the minimum block distortion (MBD) occurs at the center point. The search pattern is then switched from LDSP to SDSP as reaching to the final search stage as in fig 5(b).

Among the five checking points in SDSP, the position yielding the MBD provides the motion vector of the best matching block. [3][6]

E] Hexagonal Search :

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, 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 depicted in Fig. 4.11.

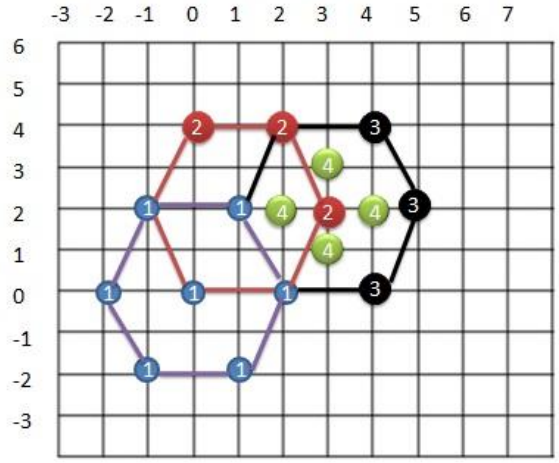

Fig. 7 hexagon search method[8]

Step 1: The large hexagon with seven checking points is formed at the center of a predefined search window. Calculate the cost at each of the 7 points.
Step 2: If minimum cost is found to be at the center of the hexagon, move to step 5 otherwise proceed to step 3.
Step 3: With the minimum cost point in the previous search step as the center, a new hexagon is formed. Three new candidate points are checked, and the minimum cost point is again identified. That is the new center.
Step 4: Repeat step 3 until you get the same center in two successive iterations. Then proceed to step 5.
Step 5: Switch the search pattern from large to small size of diamond. Add 4 points around the new center and calculate the cost. The new minimum cost point is the final solution of the motion vector [8].
Consider the example shown in fig. 7. In the first step we have 1 center and 6 others check points. In the first iteration point (2,1) is winner so we added 3 more points around the winner node. In the 2nd iteration point (2,3) is winner so we added 3 new points around it. In the 3rd iteration again the same point (2,3) is a winner so we switched from hexagon to SDSP and added 4 check points. And finally the point (1,3) is the winner. So motion vector for this macro block is.(1,3).

III. SIMULATION RESULTS

In this dissertation work, we have implemented 5 block matching algorithm methods. We have implemented exhaustive search, three step search, four step search, diamond search and Hexagonal search. We have calculated PSNR and number of computations required for each method for each frame.
Fig 8 show number of computations required for all the mehods, while figure 9 shows number of computations required for all the methods except Exhaustive search, so we can compare methods properly. Figure 9 shows the PSNR achieved in all the methods. These results are for the test video sequence garden.
It can be seen from the results that number of computations required for the exhaustive search is higher as compared to all the other methods.

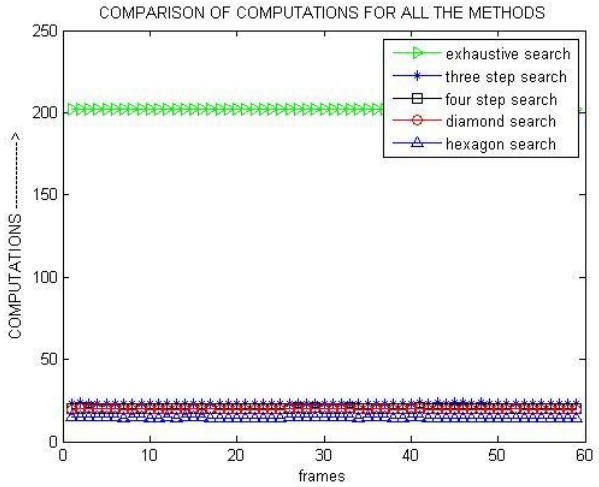

Fig. 8 comparison of all methods for computations

Now consider figure 9. It can be seen clearly from the result that Diamond Search method requires less number of computations and Hexagonal Search requires least number of computations to obtain motion vector.
In figure 10 PSNR achieved in all the methods is shown. From the result it is clear that Exhaustive Search method has best PSNR while Diamond Search and Hexagonal Search has significant PSNR as compared to Exhaustive Search.

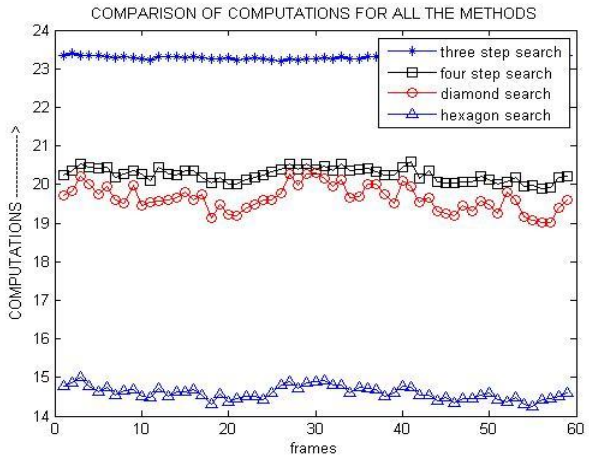

Fig. 9 comparison of computations for TSS,FSS,DS and HS

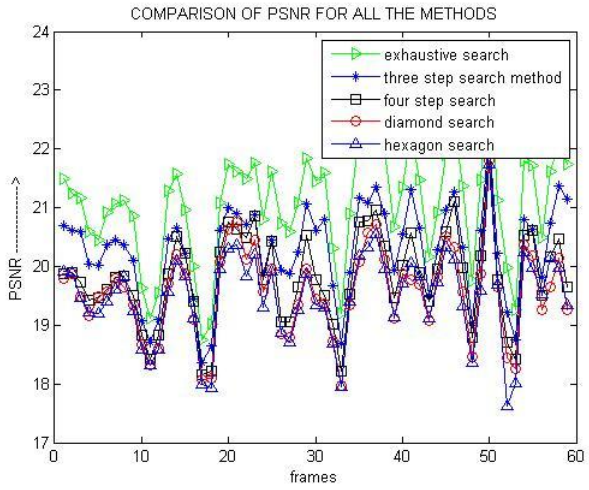

Fig. 10 Comparison of PSNR

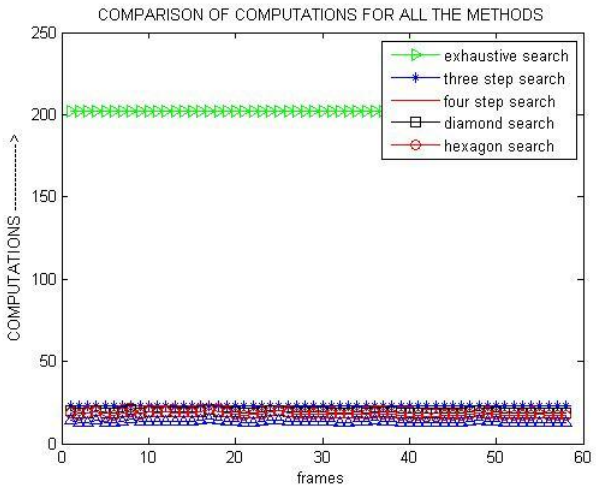

Fig.11 comparison of computations for all methods.

Fig 11 show number of computations required for all the mehods, while figure 12 shows number of computations required for all the methods except Exhaustive search, so we can compare methods properly. Figure 13 shows the PSNR achieved in all the methods. These results are for the test video sequence football.

It can be seen from the results that number of computations required for the exhaustive search is higher as compared to all the other methods.

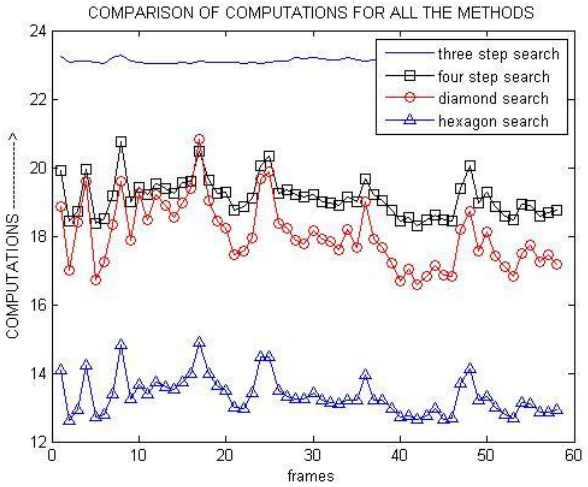

Fig.12 Comparison of computations for TSS, FSS, Hs and DS

For football

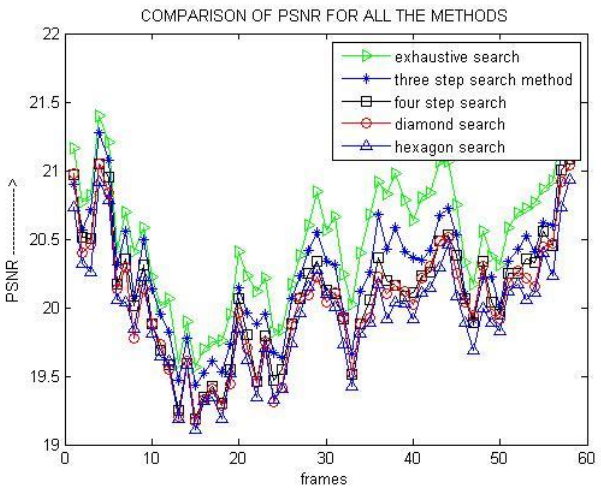

Fig.13 Comparison of PSNR

IV SUMMARY

In this paper we have discussed existing Block Matching Algorithms like ES, TSS, FSS, DS and HS. All these algorithms can be summarized as under:

- Exhaustive search has the best PSNR but on the other hand it requires maximum number of computations to obtain motion vector.
- Three Step Search, New Three Step Search and Four Step Search require less number of computations but PSNR is very low in these methods.
- DS and HS similar PSNR to exhaustive search with fewer number of computations.
- From the results it can be seen clearly that hexagonal search require least number of computations with significant visual quality.

REFERANCES

[1] M. Manikandan, P. Vijayakumar and N. Ramadass**,** **"**Motion estimation method for video compression - an overview", Wireless and Optical Communications Networks, *IFIP Intl' Conference*, pp. 5-11, 13 April 2006.

[2] X. Jing and L. P. Pui., "An efficient three-step search algorithm for block motion estimation."*IEEE Transactions on Multimedia,* volume 6, 2004, pp. 435 – 438.

[3] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Transactions Circuits Syst. Video Technoolgy.*, volume 4, Aug. 1994, pp. 438–442.

[4] M.Ghanbari, "The Cross-Search Algorithm for Motion Estimation", *IEEE Transactions on Communications*, Vol. 38, No. *7*, July 1995.

[5] L M Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation", *IEEE Transactions Circuits System Video Technol*ogy, vol. 6, June 1996, pp. 313-317.

[6] Shan Zhu and 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, February 2000.

[7] Chun-Ho Cheung, and Lai-Man Po, "A Novel Cross-Diamond Search Algorithm for Fast Block Motion Estimation", *IEEE Transactions Circuits And Systems For Video Technology*, vol 12., no.12, December 2002, pp. 1168-1177.

[8] Ce Zhu, Xiao Lin, and Lap-Pui Chau. "Hexagon-Based Search Patten for Fast Block Motion Estimation," *IEEE Transactions on Circuits and Systems for Video Technology,* Vol.12, No.5, May 2002,pp.349-355.

[9] Ce Zhu, X. Lin, L. Chau, and L. M Po., "Enhance hexagonal search for fast block motion estimation", *IEEE Transactions on Circuits and Systems for Video Technology,* 2004, pp.1210 - 1214.

[10] L. Chau and Ce Zhu, "A Fast Octagon Based Search Algorithm for Motion Estimation", *Signal Processing*, Volume 83 Issue 3, March 2003, pp.671-675.

[11] Liang Yaling Liu Jing Du Minghui , , "A Cross Octagonal Search Algorithm For Fast Block Motion Estimation", *International Symposium on Intelligent Signal Processing and Communication Systems*, December 13-16, 2005 Hong Kong, pp.357-360.

[12] Zaynab Ahmed , Abir Jaafar Hussain and Dhiya Al-Jumeily, "Fast Computations of Full Search Block Matching Motion Estimation (FCFS)", ISBN: 978-1-902560-25-0 © PGNet(2011).

[13] S. Immanuel Alex Pandian, Dr.G. Josemin Bala and Becky Alma George, "A Study on Block Matching Algorithms for Motion Estimation", International Journal on Computer Science and Engineering, Jan 2011.

[14] Chittranjan Pradhan and Dipannita Adak, "Survey on Block Matching Algorithms using Motion Estimation", *International Journal of Computer Applications (0975 – 8887, Volume 46– No.16,* May 2012,pp.6-10.

[15] Borko Furhut, "A survey on multimedia compression technique and standards", *Department of computer science and engineering, Florida Atlantic university,*2005.

[16] Wei-Yi Wei, "Digital Video Compression, Fundamentals and standards " , *National Taiwan University, Taipei, Taiwan, ROC*

[17] Yung-Ming Chou and Hsueh-Ming Hang," A New Motion Estimation Method Using Frequency Components", *Journal of Visual Communication and Image Representation Vol. 8, No. 1*, March, 1997.

[18] Ming Fai Fu, Oscar Au, Wing Cheong Chan," Temporal Interpolation using Wavelet Domain Motion Estimation and Motion Compensation*", IEEE ICIP*, 2002.

[19] Min Li, Mainak Biswas, Sanjeev Kumar and Truong Nguyen, "DCT-Based Phase Correlation Motion Estimation", *UCSD, ECE Dept., La Jolla CA 92093.*

[20] M. Manikandan, P. Vijayakumar and N. Ramadass**,** **"**Motion estimation method for video compression - an overview", Wireless and Optical Communications Networks, *IFIP Intl' Conference*, pp. 5-11, 13 April 2006.