A Study on Block Matching Algorithms for Motion Estimation in Video Coding

L.C.Manikandan, Dr.R.K.Selvakumar, IEEE Senior Member

Abstract— Block-based motion estimation methods are the most popular and widely used methods in video coding systems. Motion estimation reduces temporal redundancies by exploiting inter picture correlation. This paper is a study of the existing block matching algorithms used for motion estimation in video coding. The algorithms that are revealed in this paper are widely used in implementing various standards ranging from MPEG1 / H.261 to MPEG4 / H.263. This paper also presents the computational complexity of the searching points of different block matching algorithms.

Index Terms— Block Matching, Motion Estimation, Video Coding, Absolute Mean Difference, MPEG, H.264

1 Introduction

VIDEO coding is the process of compacting or condensing a digital video sequence into a smaller number of bits. Video coding involves a complementary pair of systems, "encoder & decoder". The encoder converts the source data into a compressed form prior to transmission or storage and the decoder converts the compressed form back into a representation of the original video data. A video sequence typically contains temporal redundancy that is two successive pictures are often very similar except for changes induced by object movement, illumination, camera movement, and so on. Motion estimation and compensation are used to reduce this type of redundancy in moving pictures. The block-matching algorithm (BMA) for motion estimation has proved to be very efficient in terms of quality and bit rate. Therefore, it has been adopted by many standard video encoders.

Through this study we reviewed the Block Matching Algorithms(BMA), Full Search Motion Estimation[3], Cross-Search Algorithm[1], Three Step Search Algorithm[4], New Three-Step Search Algorithm[5], Four-Step Search Algorithm[8], Diamond Search Algorithm[9], Cross Diamond Search Algorithm[10], Hexagonal Search[11], Adaptive Rood Pattern Search[12].

2 Block Matching Algorithms

Block matching technique is the most popular and practical motion estimation method in video coding. Fig.1 shows how the block matching motion estimation technique works. Each frame of size M x N is divided into square blocks B(i, j) of size (b x b) with i = 1…., M/b and j = 1…..N/b. For each block Bm in the current frame, a search is performed on the reference frame to find a matching based on a block distortion measure (BDM). The motion vector (MV) is the displacement from the current block to the best matched block in the reference frame. MV consists of is a pair (x, y) of horizontal and vertical displacement values. Usually, a search window is defined to confine the search. The same motion vector is assigned to all pixels within block.

Suppose a block has size b x b pixels and the maximum al-
lov ital
an late
blo ock
ma

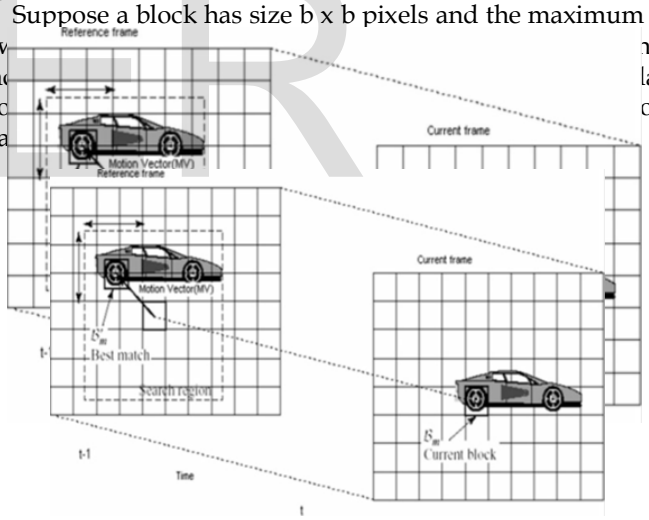
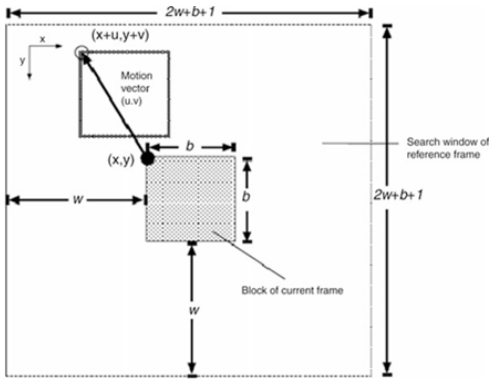


Fig.1 Block matching motion estimation



•*L.C.Manikandan, Asst. Professor, School of CSE, Mar Ephraem College of Engineering & Technology, Marthandam – 629 171, Tamilnadu, India E-mail: lcmanikandan@gmail.com*

•*Dr.R.K.Selvakumar, Professor, Dept. of IT, Agni College of Technology, Chennai – 600 103, Tamilnadu, India. E-mail: rkselvam@rediffmail.com*

Fig.2 Block matching method

A matching between the current block and one of the candidate blocks is referred to as a point being searched in the search window. If all the points in a search window are searched, the finding of a global minimum point is guaranteed.

2.1 Full Search Motion Estimation (FS)

In order to get the best match block in the reference frame, it is necessary to compare the current block with all the candidate blocks of the reference frames. Full search motion estimation calculates the sum absolute difference (SAD) value at each possible location in the search window. Full search computed the all candidate blocks intensive for the large search window. Full search algorithm is illustrated in Fig.3.

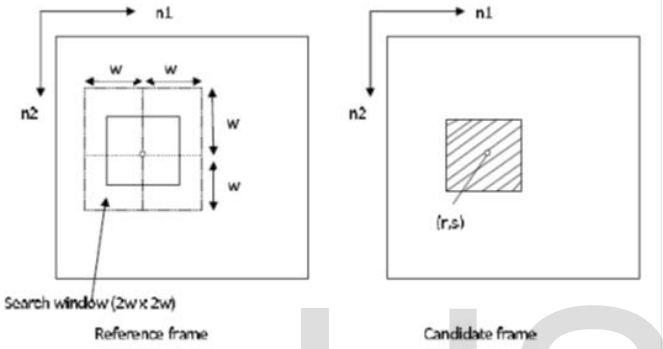


Fig.3 Full search motion estimation

2.2 Cross Search Algorithm

The cross search algorithm (CSA) has been proposed by Ghanbari in 1990. It is also a logarithmic step search algorithm using a (X) cross searching pattern in each step. The basic idea is still a logarithmic step search where in each search step only 4 locations are tested. The cross search algorithm can then be described as follows:

Step 1: The current block and the block at (0,0), are compared and if the value of the distortion function is less than a predefined threshold T then the current block is classified as a nonmoving block and the search stops. Otherwise go to Step 2.

Step 2: Initialize the minimum position (m, n) at m = 0, n = 0 and set the search step size p equal to half of the maximum motion displacement w, i.e., p = w/2.

Step 3: Move the coordinates (i, j) to the minimum position (m, n), that is i = m and j = n.

Step 4: Find the minimum position (m, n) of the coordinates (i, j) , (i - p , j - p) , (i - p , j + p) , (i + p , j - p) and (i + p , j +P).

Step 5: If p = 1 go to Step 6, otherwise halve the step size p, and then go to Step 3.

Step 6: If the final minimum position (m, n) is either (i, j), (i - 1, j - 1) or (i + 1, j + 1) go to Step 7, otherwise go to Step 8.

Step 7: Search for the minimum position at (m, n), (m - 1, n), (m, n - l), (m + 1, n) and (m, n + 1). Here the end points of a Greek cross (+) are searched.

Step 8: Search for the minimum position at (m, n), (m- 1, n - l), (m - 1, n + l), (m + 1, n - 1) and (rn + 1, n + 1). In this case the end points of a St. Andrew's cross (X) are searched.

CSA for a maximum motion displacement of w = 8 pels/frame is shown in Fig.4.

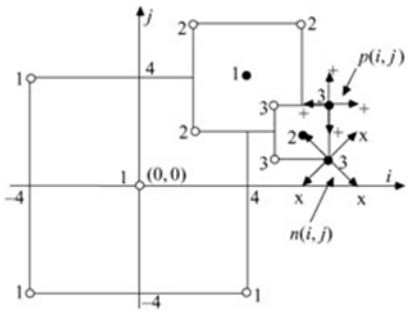


Fig.4 An example of the CSA search for w=8 pels/frame

In CSA almost 25% (one in every four) of the pels at the boundaries are not searched.

2.3 Three Step Search Algorithm (TSS)

TSS algorithm was proposed by Koga et al. [2], this is a fine-coarse search mechanism. The TSS algorithm can then be described as follows:

Step 1: It involves search based on 4-pixel/4-line resolution at nine locations i.e. 9x9 search window, with the center point corresponding to zero MV.

Step 2: It involves search based on 2-pixel/2-line resolution i.e. 5x5 search window around the location determined by the first step.

Step 3: This is repeated in the third step with 1-pixel/1-line resolution and a search window of 3x3. Step4: The last step yields the MV.

The search pattern of TSS is shown in Fig.5.

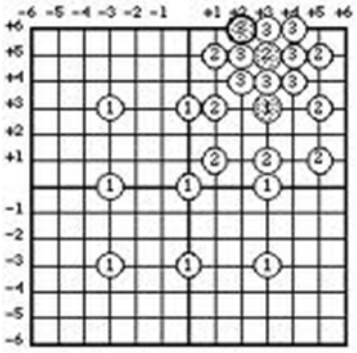


Fig.5 Search pattern of three step search algorithm

2.4 New Three-Step Search Algorithm (NTSS)

NTSS [4] improves on TSS results by providing a center biased searching scheme and having provisions for half way stop to reduce computational cost. It was one of the first widely accepted fast algorithms and frequently used for implementing earlier standards like MPEG 1 and H.261. The NTSS algorithm is described as follows.

Step 1: Totally points are checked including the central nine

points on the 3 x 3 grid and the eight neighboring points on the 9 x 9 grid. If the minimum BDM point is the search window center, the search will be terminated; otherwise go to Step 2.

Step 2: If one of the central eight neighboring points on the 3 x 3 grid is found to be the minimum in the first step, go to Step 3; otherwise go to Step 4.

Step 3: Move the small 3 x 3 search window so that the window center is the winning point found in Step1. Search additional five or three points according to the location of the previous winning point, then the search will stop.

Step 4: Reduce the large 9 x 9 search window size by half and move the center to the minimum BDM point in Step 1, follow the searching process of Step 2 and Step 3 in 3SS.

Fig.6 shows two different search paths for finding motion vector within 5 x 5 area.

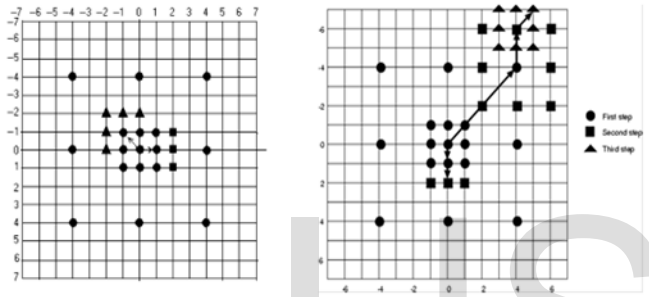


Fig.6 Two different search paths for finding MV within 5 x 5 area in N3SS.

2.5 Four-Step Search Algorithm (FSS)

The FSS algorithm is summarized as follows:

Step 1: A minimum BDM point is found from a nine-checking point's pattern on a 5 x 5 window located at the center of the 15 x 15 searching area as shown in Fig.7 (a). If the minimum BDM point is found at the center of the search window, go to Step 4; otherwise go to Step 2.

Step 2: The search window size is maintained in 5 x 5. However, the search pattern will depend on the position of the previous minimum BDM point.

a) If the previous minimum BDM point is located at the corner of the previous search window, five additional checking points as shown in Fig.7 (b) are used.

b) If the previous minimum BDM point is located at the middle of horizontal or vertical axis of the previous search window, three additional checking points as shown in Fig.7 (c) are used. If the minimum BDM point is found at the center of the search window, go to Step 4; otherwise go to Step 3.

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.7 (d) and the direction of the overall motion vector is considered as the minimum BDM point among these nine searching points.

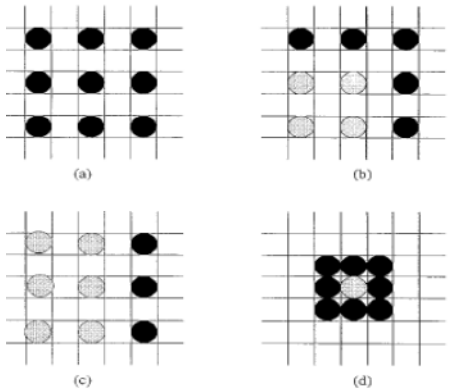


Fig.7 Search Patterns of 4SS. (a) First Step, (b) second/third step, (c) second/third step, (d) fourth step

Two examples of 4SS are shown in Fig. 8 with different search paths.

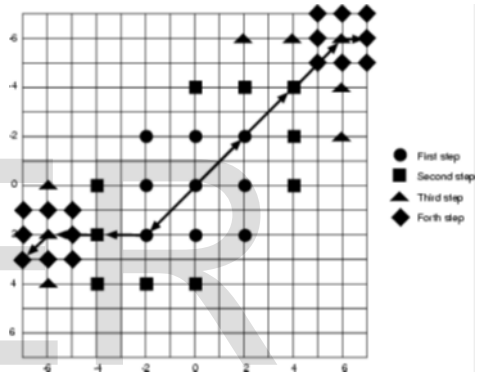


Fig.8 Two different search paths of 4SS

2.6 Diamond Search Algorithm (DS)

DS [9] algorithm employs two search patterns as illustrated in Fig.10, which are derived from the crosses (×) in Fig.9. The first pattern, called large diamond search pattern (LDSP), 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 smaller diamond shape, called small diamond search pattern (SDSP).

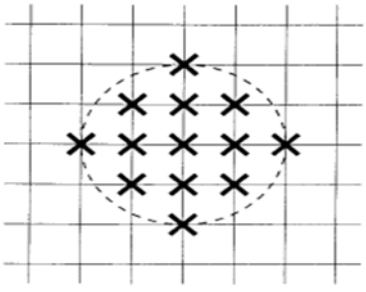


Fig.9 An appropriate search pattern support—circular area with a radium of 2 pels.

The 13 crosses show all possible checking points within the

circle.

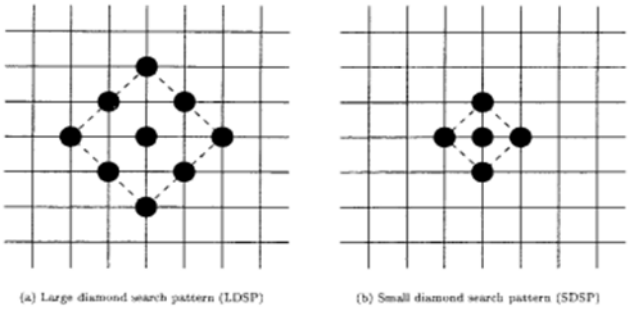


Fig.10 Two search patterns derived from Fig.9

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 repositioned 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.

2.7 Cross Diamond Search Algorithm (CDS)

The DS algorithm uses a large diamond-shaped pattern (LDSP) and small diamond-shaped pattern (SDSP), as depicted in Fig.11.

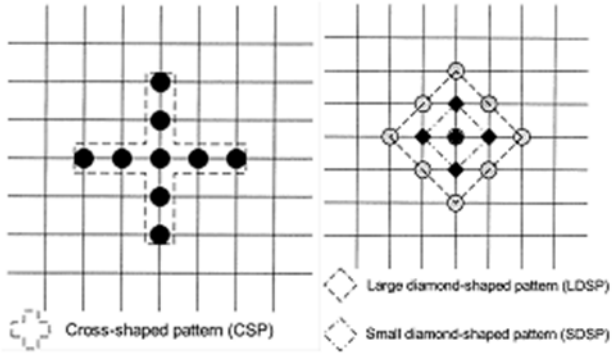


Fig.11 Search patterns used in the CDS algorithm.
(a) CSP (b) LDSP and SDSP.

Below summarizes the CDS algorithm:

Step 1: Starting: A minimum BDM is found from the nine search points of the CSP located at the center of search window. If the minimum BDM point occurs at the center of the CSP, the search stops. Otherwise, go to Step (2).

Step 2: Half-diamond Searching: Two additional search points of the central LDSP closest to the current minimum of the central CSP are checked, i.e., two of the four candidate points located at (±1, ±1). If the minimum BDM found in previous step located at the middle wing of the CSP, i.e., (±1, 0) or (0, ±1) and the new minimum BDM found in this step still coincides with this point, the search stops. Otherwise, go to Step (3).

Step 3: Searching: A new LDSP is formed by repositioning the minimum BDM found in previous step as the center of the LDSP. If the new minimum BDM point is still at the center of the newly formed LDSP, then go to Step (4) (Ending); otherwise, this step is repeated again.

Step 4: Ending: With the minimum BDM point in the previous step as the center, a new SDSP is formed. Identify the new minimum BDM point from the four new candidate points, which is the final solution for the motion vector.

2.8 Hexagonal Search (HEXBS)

The HEXBS algorithm is summarized as follows:

Step 1: The large hexagon with seven checking points is centered at, the center of a predefined search window in the motion field. If the MBD point is found to be at the center of the hexagon, proceed to Step (3) (Ending); otherwise, proceed to Step (2) (Searching).

Step 2: With the MBD point in the previous search step as the center, a new large hexagon is formed. Three new candidate points are checked, and the MBD point is again identified. If the MBD point is still the center point of the newly formed hexagon, then go to Step (3) (Ending); otherwise, repeat this step continuously.

Step 3: Switch the search pattern from the large to the small size of the hexagon. The four points covered by the small hexagon are evaluated to compare with the current MBD point. The new MBD point is the final solution of the motion vector.

A hexagon-based search pattern is depicted in Fig.12.

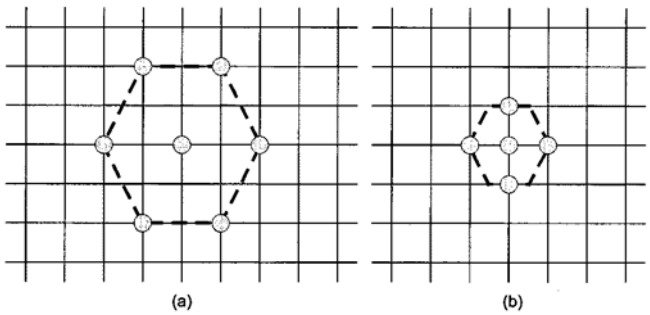


Fig.12 (a) Large Hexagonal Block Search (HEXBS)
(b) Small Hexagonal Block Search (HEXBS)

2.9 Adaptive Rood Pattern Search (ARPS)

The ARPS algorithm is summarized as follows:

Step 1: Compute the matching error (SAD centre) between the current block and the block at the same location in the reference frame.

Step 2: Align the center of ARP with the center point of the search window and check it's four search points plus the position of the predicted MV to find out the current min-

imum matching error (MME) point.

Step 3: Set the center point of the unit-size rood pattern (URP) at the MME point found in the previous step and check its points. If the new MME point is not incurred at the center of the current URP, repeat this step; otherwise, the MV is found, corresponding to the MME point identified in this step.

ARPS algorithm search pattern is as shown in Fig.13.

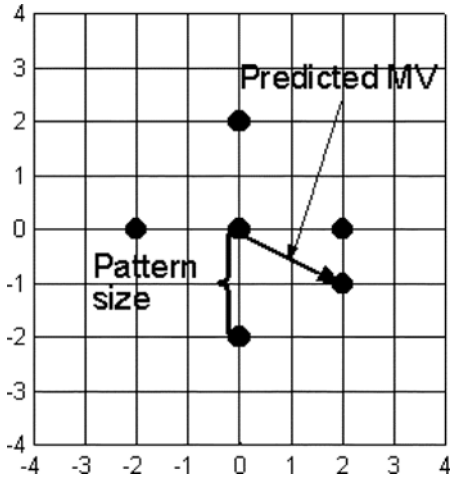


Fig.13 Adaptive Rood Pattern

3. Comparative Study

Here we are categorizing the algorithms into nonlinear method and linear method based on maximum searching points. Most of the algorithms follow nonlinear approach. In our survey CDS algorithm only follow linear search approach. From this comparative study, we have found that the full search (FS) takes larger number of search points. The Adaptive Rood Pattern Search (ARPS) algorithms take a minimum numbers of search points and gives better PSNR. Getting a better quality image by reducing number of search points remains a goal. The Table 1 shows the computational complexity of the searching points.

TABLE 1
Comparison Table – Maximum Searching Points

Method	Algorithm	Maximum Searching Time	Maximum number of search points – Window Size W = 4, 8, 16		
			4	8	16
Non Linear Model	FS	$(2w + 1)^2$	81	269	1089
	CS	5 + 4 log2w	13	17	21
	TSS	$1 + 8 \log_2 w$	17	25	33
	NTSS	$[1+8 \log_2 w] + 8$	25	33	41
	4SS	18 (log2(w/4)) + 9	81	289	1089
	DS	$9+msx\{5,4,3\}* \log_2 w$	19	24	29
	HEXBS	$7+3*\log_2 w+4$	17	20	23
	ARPS	$1+4*\log_2 w$	9	13	17
Linear Model	CDS	3 + 2w	11	19	35

The Table 2 shows the PSNR Performance evaluation of various algorithms in different video streams.

Table 2
PSNR Performance evaluation

BMA's	Video Sequences	
	Football	Foreman
FS	26.78	37.98
CS	24.82	32.35
TSS	26.30	31.92
NTSS	25.97	37.33
4SS	23.51	32.49
DS	26.49	37.67
CDS	25.78	34.26
HEXBS	24.66	31.42
ARPS	26.48	37.42

4. Conclusion

Block matching techniques are the most popular and efficient of the various motion estimation techniques. In this paper, an overview of some Block matching motion estimation algorithms range from the very basic Full Search to the recent fast adaptive algorithms like Pattern Based Search in H.264 CODEC has been discussed with their computational complexity. As a consequence, the computation of video coding is greatly reduced with ARPS.

References

[1] M. Ghanbari, "The Cross-Search Algorithm for Motion Estimation", Vol. 38, No. I, pp 950-953, IEEE Transactions on Communications, July 1990.

[2] S. Immanuel Alex Pandian et al., "A Study on Block Matching Algorithms for Motion Estimation", Vol.3, No.1, pp 34-44, International Journal on Computer Science and Engineering, Jan 2011.

[3] T. Koga, K. Iinuma, A. Hirano, Y. Iijima and T. Ishiguro, "Motion compensated interframe coding for video conferencing," pp. G5.3.1-5.3.5, Pro. Nat. Telecommun. Conf., New Orleans, Nov. 1981.

[4] Renxiang Li, Bing Zeng, and Ming L. Liou, "A New Three-Step Search Algorithm for Block Motion Estimation", IEEE Trans. Circuits And Systems For Video Technology, Vol 4., No. 4, pp. 438-442, August 1994.

[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] Deepak Turaga, Mohamed Alkanhal, "Search Algorithms for Block-Matching in Motion Estimation" Spring, 1998.

[7] Lai-Man Po Wing-Chung Ma, "A Novel Four-Step Search Algorithm for Fast Block Motion Estimation", vol. 6, No. 3, pp. 313-317, IEEE Transactions on Circuits Syst. Video Technol., June 1996.

[8] Shan Zhu and Kai-Kuang Ma, "A New Diamond Search Algorithm for Fast

Block-Matching Motion Estimation", Vol. 9, No. 2, pp 287-290, I EEE Transactions on Image Processing, FEBRUARY 2000.

[9] Chun-Ho Cheung and Lai-Man Po, "A Novel Cross-Diamond Search Algorithm for Fast Block Motion Estimation", Vol. 12, No. 12, pp 1168-1177, IEEE Transactions on Circuits and Systems for Video Technology, December 2002.

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

[11] Yao Nie, and Kai-Kuang Ma, "Adaptive Rood Pattern Search for Fast Block Matching Motion Estimation", Vol 11, no. 12, pp. 1442-1448, IEEE Transactions on Image Processing, December 2002.

[12] Aroh Barjatya, "Block Matching Algorithms For Motion Estimation", Student Member, IEEE, DIP 6620 Spring 2004.

[13] Chandra Sekhar. CH, J.V.K. Ratnam, "Comparison of Fast Block Matching Algorithms for Motion Estimation", pp 1609-1618, International Journal of Electronics and Computer Science Engineering.

[14] M. Ezhilarasan and P. Thambidurai, "Simplified Block Matching Algorithm for Fast Motion Estimation in Video Compression " vol. 4, pp. 282-289, Journal of Computer Science, 2008.

[15] A. Barjatya, "Block Matching Algorithms for Motion Estimation," DIP 6620, Spring 2004.

[16] Kwon Moon Nam, Joon-Seek Kim, Rae-Hong Park "A Fast Hierarchical Motion Vector Estimation Algorithm Using Mean Pyramid" Vol.5, No.4, pp 344-351, IEEE Transactions on Circuits and Systems for Video technology, August 1995.

[17] MPEG-4 Video Verification Model (Version 14.0), ISO/IEC JTC1/SC29/WG11 N2932, Oct. 1999.

[18] M. J. Chen, L. G. Chen, and T. D. Chiueh, "One-dimensional full search motion estimation algorithm for video coding," Vol. 4, pp. 504–509, IEEE Trans. Circuits Syst. Video Technol., Oct. 1994.

[19] S. Metkar and S. Talbar, "Motion Estimation Techniques for Digital Video Coding", V0l 12, pp 64, SpringerBriefs in Computational Intelligence, 2013.

[20] B.G. Kim, S.T. Kim, S.K. Song and P.S. Mah, "Fast-adaptive rood pattern search for block motion estimation", Vol.41, No.6, ELECTRONICS LETTERS, August 2005.