Hash Based Block Matching For Motion Estimation

Ku. Dipti V. Jilhare, Dr. S. Y Amdani

Abstract— **This paper propose a new block matching algorithm for motion estimation. Two different block matching algorithms using for motion estimation are evaluated where no of computation to find out best match and peak signal to noise ratio (PSNR) are used to find out most optimal algorithm. Two different block matching algorithm are implemented in MATLAB. Each algorithm is evaluated using PSNR and compression time. The result shows that among the two algorithms evaluated proposed Hash based block matching algorithm has the best based on PSNR and compression time.**

In the proposed scheme, the blocks sharing the same hash values with the current block are selected as prediction candidates. Then the hash-based block selection is employed to select the best candidates. Achieve best coding efficiency. The proposed scheme, bit rate saving and time reduction.

Index Terms— **Motion estimation, Hash based block matching, Motion compensation, HEVC.**

I. Introduction

Motion estimation is the technique that helps to reduces temporal correlation of successive video frames to make compression of video efficient. Motion estimation estimates the pair of motion vectors which map the previous frame to the current frame of a video sequence. Advantage of Motion estimation is instead of transmit whole frame every time only motion vectors need to be transmit. So compression can be achieved by coding the motion vectors. Availability of limited channel bandwidth and storage medium for real time video processing require an efficient coding scheme. So fast and efficient Motion estimation techniques are required for making encoding process is an efficient one. Hence considerable works are demanded in this field. Motion estimation has mainly two subcategory first one is block matching algorithm methods and second one is pixel recursion method. Block motion estimation seems to be the efficient and regular method. The search method, search range and block matching criteria are the some major factors that affect the performance of Block matching Algorithm[1]. Full search method (FSBM) [5] and fast block matching algorithms are two methods are used to find out best match in motion estimation technique. In FBMAs the set of motion vector is computed by some fixed set of search patterns like, Three Step Search(TSS)[5], Four Step Search[6], Block Based Gradient Decent(BBGDS) and DS[7]. The bottleneck of motion estimation process is large number of computation. By reducing number of the search points are to be evaluated, computations in a Block matching algorithm such as addition, subtraction and absolute operations can be reduce. A fast motion estimation algorithm is needed consequently. In this paper a new algorithm for finding best match and motion vector is proposed. Hash based block matching algorithm is a fast block matching algorithm. Hash based algorithm use hash function to find out the best match[3]. It provides improved performance compared to previous methods in term of Peak Signal Noise Ratio (PSNR) and number of search points.

Ku. Dipti V. Jilhare, Computer Department, BNCOE, Pusad, INDIA
Dr. S. Y Amdani, Computer Department, BNCOE, Pusad, INDIA

II. LITERATURE SURVEY:

In past, many techniques have been developed to accelerate the block matching process. We classify these techniques into three categories. The techniques in the first category save the computations by reducing the number of the positions searched. Therefore, the obtained minimum of the matching error may only be a local minimum within the search set, S. The techniques in the second category, try to reduce the computational cost of the matching error for each search position. Whether this kind of techniques can obtain the global minimum or not depends on how we compute and compare the matching errors. The techniques in the first and second categories can be combined to further improve the efficiency and this kind of hybrid methods are classified as the third category[2].

Block based Motion Estimation has been adopted [2] to reduce the temporal redundancy between frames.

Full search involves the computation of SAD at each location in the search window. For a search window of size +/-P pixels, the number of search locations is $(2P+1)^2$. For a search window of 32x32 and a block size of 16x16, a total of 289 locations is searched to find the best match with the minimum SAD value. This results in significant computational complexity.

Many algorithms have been proposed to reduce the computational complexity of full search motion estimation. Some of the popular ones are the Three Step Search (TSS [4]), the New Three Step Search (NTSS [5]), the Four Step Search (4SS [6]), the Diamond Search (DS [7]) and the Adaptive Rood Pattern Search (ARPS [8]). These algorithms try to do Small Square (TSS, NTSS, 4SS) or diamond shaped (DS) search around a search center, and refine the search around the best matching block. Early termination techniques based on the SAD threshold values are used to reduce the computation cost. Algorithms like ARPS employ sophisticated search center prediction as the start point. Though these algorithms address computational cost well, but the performance in terms of PSNR is close to FS algorithm.

Concerning the VLSI implementation, most of these fast algorithms, e.g., the three-step search (TSS) [4] TSS has a fixed number of search points per each block; this gives a speedup of 9 over the FS. The TSS algorithm is one of the simple and efficient methods used for ME. TSS saving factor is 100 times greater when compared with Full search block matching Algorithm (FSBMA)., (TSS) have the drawbacks of irregular data flow and high control overhead, while the full-search BMA has the advantages of regular data flow and low control overhead [2]. Recently, a number of algorithms with regard to the pattern matching problems make use of integral projections to simplify the computational complexity of the pattern matching operation[19-21].

III. FULL SEARCH METHOD

In this letter, full-search (FS) algorithm method which exhaustively computes all possible candidate block cost function within search window to find the optimum value. To find out motion vector for a given candidate block, candidate block that gives the minimum block distortion measure is chosen. Full search method has not been a popular choice due to the large computational cost involved in it. However, Full search method gives the best match, so it produces the highest PSNR compared to other BMA methods. There are total (2P+1) × (2P+1) number of positions investigated in the Full search algorithm. Because of the large computation time, it is not suitable for real-time video coding. a fast full-search BMA (FFBMA), which is also based on the uses of integral projections, is presented to provide much faster motion estimation than that using the traditional FBMA, while preserving the optimality of estimate accuracy[9].

IV. MATCHING CRITERIA

The matching is to determine the comparison between the frames or portions of frames of video. The similarity measurement or correlation measurement is a very important element in the matching process. Instead finding the maximum similarity or correlation, finding the minimum dissimilarity or matching error is the best way in block matching. There are several matching criteria or cost functions have already been proposed, of which the most popular and less computation expensive is Sum of Absolute Difference (SAD). Another matching function is Mean Absolute Differences (MAD) and Peak Signal to Noise Ratio (PSNR) which are represented in Equations given below

$$\mathrm{SAD} = \sum_{i=0}^{N-1}\sum_{i=0}^{N-1}|Cij - Rij| \qquad (1)$$

$$\mathrm{MAD} = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{i=0}^{N-1}|Cij - Rij| \qquad (2)$$

In the Sum of Absolute Difference correspondingly pixels from each block are compared and their differences are summed, as in the equation (1). In the equation N is the size of the macro block, Cij is pixel value of current macro block and Rij is the pixels value of reference macro block, those values are compared. This SAD function is for two blocks C and R of size NxN. Cij is the value of pixel in the i^{th} row and j^{th} column of block C. The lower the SAD the better is the match and so the candidate block with minimum SAD should be chosen.

$$\mathrm{PSNR} = 10\log_{10}\left[\frac{255^2}{\mathrm{MSE}}\right] \qquad (3)$$

Peak-Signal-to-Noise-Ratio (PSNR) showing the motion compensated image its calculation is by motion vectors and macro blocks from the reference frame[1].

V. PROPOSED ALGORITHM

A. FRAMEWORK OF HASH-BASED BLOCK MATCHING

The basic idea of the hash-based block matching scheme is to find the prediction block for the current block by comparing the hash values of the current block with the hash values of the blocks in the reconstructed regions or reference frames. As shown in Fig.1, The first-level and second-level hash values of the current block are first generated. Then the hash values of the blocks in the reconstructed regions are compared with those of the current block. The first-level hash value is first used to find the candidates approximated to the current block[14].

If more than one block is found, the best prediction block is chosen from the candidates by comparing the second-level hash values. The current block is then encoded and reconstructed using the selected block as prediction. After that, there constructed block is updated to the reconstruction buffer. Then all newly available blocks in the reconstruction regions are hashed and updated to the hash dictionary. The hash dictionary constructed during the encoding process of the current frame will be used for inter motion estimation.

B. HASH-BASED BLOCK SELECTION

Let s={B_0,$B_{1,n}$,B_N} be the hashed blocks in the reconstructed regions of the current frame or the reference frames, where B_i is the i^{th} block and B_N is the number of hashed blocks. Let L_0, $L_{1,n}$, L_M be all possible first-level hash values. The hashed blocks are organized using a hash dictionary based on their first-level and second-level hash values.

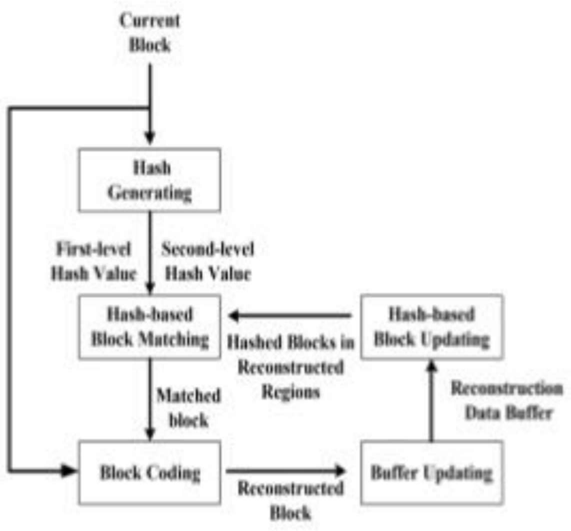


Fig 1. Framework of the proposed system

The blocks with the same first-level hash value will be stored to the same position of the hash dictionary. The blocks with the same first-level hash values and different second hash values will be stored as a list. Thus the blocks similar to the current block can be found in time 0(1). As shown in Fig. 2, given an input block with hash values (L_3, 234), the position of the blocks approximated to the current block can be determined by the first-level hash value . The selected blocks (B_{31},B_{32},B_{33}) are further checked by comparing these second-level hash values. The block with the matched second-level hash value (B_{33}) is selected as the best prediction block. After the current block is encoded, the hash-based block updating is employed to filter out identical blocks from all newly available candidates which need to be updated to the hash dictionary. Only the blocks with first-level and second-level hash values different from those in the hash dictionary will be included in the hash dictionary.

C. HASH FUNCTION DESIGN

Two hash functions are adopted in the proposed scheme to find the best prediction block. The first hash function used in the hash-based block matching is designed to find the prediction blocks approximated to the target block. However traditional hash functions like CRC [15] can only be used to find blocks identical to the input block. As a result, traditional hash functions do not help to find blocks which approximates to the input block.

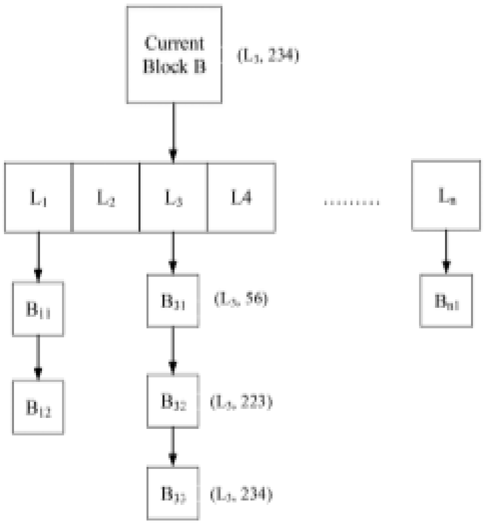


Fig 2. Hash based block matching

The proposed hash function for the first-level hash values consists of three 8-bit components , µ,r and c . The first 8-bit µ is generated from qthe average value of the target block. The second and third 8-bit components are generated by performing directional XOR operations on the target block. The second and third 8-bit components convey the structure information of the target block. Let [i , j] be the8*8 block with 8-bit pixel values, the first 8-bit µ is calculated as

$$\mu - \frac{\sum_{0\le i<8, 0\le j<8} B[i,j]}{64}. \qquad (1)$$

For the second and third 8-bit, a binary level map M is first derived using the average value μ of the target block

$$M[i,j] = \begin{cases} 1, & \text{when } B[i,j] > \mu \\ 0, & \text{otherwise} \end{cases}, 0 \le i < 8;\ 0 \le j < 8. \qquad (2)$$

The second 8-bit r is generated by performing XOR operations along rows of the level map M. The eight bits of r is deduced as follows:

$$r[j] = M[0,j] \oplus M[1,j] \oplus M[2,j] \oplus M[3,j] \oplus M[4,j] \oplus M[5,j] \oplus M[6,j] \oplus M[7,j] \quad 0 \le j < 8 \qquad (3)$$

where $\oplus$ denotes the XOR operation. The third 8-bit c is generated by performing XOR operations along columns of the level map M. The eight bits of c is deduced as follows:

$$c[i] - M[i,0] \oplus M[i,1] \oplus M[i,2] \oplus M[i,3] \oplus M[i,4] \oplus M[i,5] \oplus M[i,6] \oplus M[i,7] \quad 0 \le i < 8. \qquad (4)$$

Finally the three components µ,r,c are combined together to generate the hash value which can be calculated as follows h=(µ<<16)+(r<<8)+c.

where << denotes the left shift operator .

The second hash function is used to locate the identical blocks. The CRC[15] with 8bits, which is very suitable for filtering out identical blocks from candidates, is used as the second-level hash function in the proposed scheme.

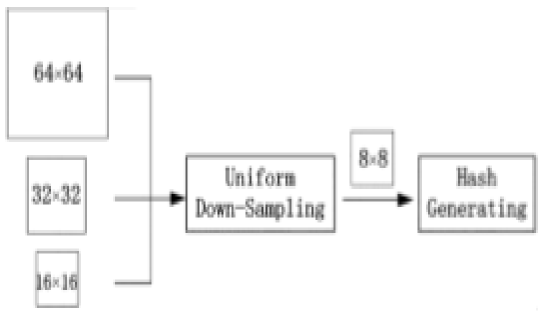


Fig 3. Uniform down-sampling process

D. Uniform Hash Generating and Temporal Hash Copying

The proposed hash functions can be extended for large blocks (64*64, 32*32, and16*16). However, the complexity of calculating hash values for large blocks is much higher than that for blocks. To reduce complexity of hash calculation, a uniform hash calculating scheme is proposed for large blocks. As shown in Fig. 3, 16*16, 32*32, 64*64 blocks are first down sampled to 8*8 blocks. Then the first-level and second-level hash values are generated using the hash function designed for blocks 8*8.

Partial frame changes are often observed from the successive frames in screen videos. It is redundant to recalculate the hash values of exactly matched regions in successive frames. We propose a temporal hash copying (THP) method to reduce the complexity of hash calculations. In the proposed scheme, the input frame is first split into square regions. Then the regions are classified into two categories based on their mean square difference (MSD) between the regions and their collocated regions in the previous frame. The regions with zero MSD are classified as zero-motion regions. Otherwise, they are classified as non-zero motion regions. For the blocks in zero-motion regions, their hash values are obtained by copying the hash values of their collocated blocks in the previous frame, whose hash values have been calculated when coding previous frames. Only the hash values of the blocks in non-zero motion regions need to be calculated.

Since most regions in screen videos are with zero-motions, THP can significantly reduce the computation time of hash calculation. The size of square regions is set to 64*64 in our implementation.

VI. EXPERIMENTAL RESULTS

The performance of the proposed motion estimation is evaluated with FS, but the search strategies and patterns vary between them. Experiments have been conducted to justify the performance of our Hash based algorithm. The peak-signal-to-noise-ratio (PSNR) was used as a performance measure.

A. COMPRESSION RATIO

The effectiveness of a compression scheme is indicated by its compression ratio, which is determined by dividing the amount of data before the compression by the amount of data after the compression. Through the removal of redundancies and sometimes at the expense of fidelity, a compression system reduces the entropy of the video data, thus reducing the bitrates required to store or transmit a bitstream.

The compression ratio can be found from a simple formula which is the size of the original data divided by the size of the compressed image as shown in equation below. This ratio shows the capability of different coding algorithms to compress images.

$$Compression\ Ratio = \frac{Size\ of\ Orginal\ image}{Size\ of\ Compressed\ image}$$

The compression ratio can be used for indicating the picture quality, since most of the compression techniques operate over a range of compression rate and decompression quality. Generally, the greater the compression ratio, the less the quality of the output images. The trade-off between compression ratio and the quality is an important factor to consider when compressing images.

The amount of data is measured in bits, which is the number of binary symbols required to represent the data. The following bitrates are commonly used to represent video data[21].

B. IMAGE QUALITY

Image quality is one of the significant measures for image and video compression systems. Normally, the compression and decompression process cause the degradation of the reconstructed image..

The simplest measures of quality are the Mean-Square Error (MSE) and the Peak Signal to Noise Ratio (PSNR). The MSE between two images is given by

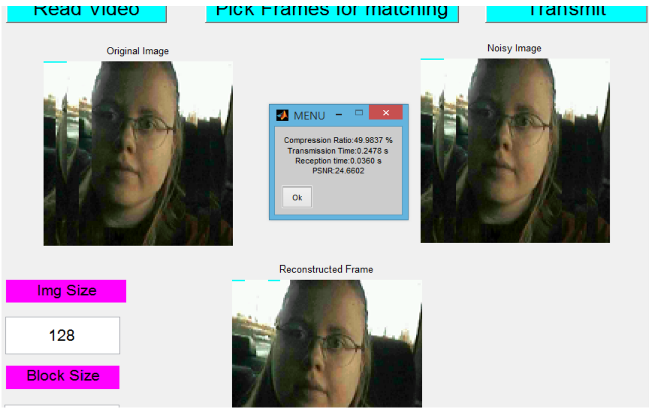
$$MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} (x_{ij} - \hat{x}_{ij})^2$$

where the images size is M×N, X_{ij} is the original image and X_{ij} is the reconstructed image. One problem with MSE is that it depends strongly on the image intensity scaling. In contrast, PSNR avoids this problem by scaling the MSE according to the image. It is determined as follows:

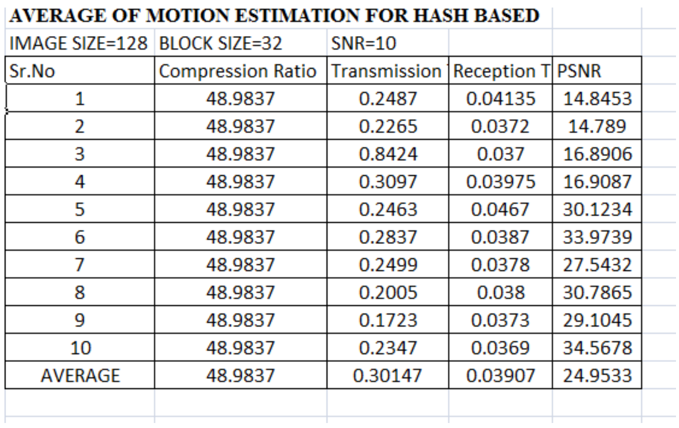
$$PSNR = 10\log_{10}(\frac{S^2}{MSE})$$

where, S is the maximum intensity value. PSNR is measured in decibels (dB). This measure (PSNR) is also not ideal, but it is commonly used. Its main failure is that the signal strength of the image is estimated as $(S)^2$ (value squared), rather than the actual signal strength of the image. However, the difference between PSNR for different compression methods or parameter settings is still a valid comparison, despite this drawback.

C. Screen shot Of Output screen of Hash Based with calculated Compression ratio, Transmission Time,Reception Time, PSNR



Compression Ratio:49.9837 %
Transmission Time:0.2478 s
Reception time:0.0360 s
PSNR:24.6602



AVERAGE OF MOTION ESTIMATION FOR HASH BASED

IMAGE SIZE=128 BLOCK SIZE=32 SNR=10

Sr.No	Compression Ratio	Transmission	Reception T	PSNR
1	48.9837	0.2487	0.04135	14.8453
2	48.9837	0.2265	0.0372	14.789
3	48.9837	0.8424	0.037	16.8906
4	48.9837	0.3097	0.03975	16.9087
5	48.9837	0.2463	0.0467	30.1234
6	48.9837	0.2837	0.0387	33.9739
7	48.9837	0.2499	0.0378	27.5432
8	48.9837	0.2005	0.038	30.7865
9	48.9837	0.1723	0.0373	29.1045
10	48.9837	0.2347	0.0369	34.5678
AVERAGE	48.9837	0.30147	0.03907	24.9533

Table no: 1

VII. CONCLUSION

Screen content has emerged as an important category of video source, the block matching received very much attention by researcher because of their simplicity and regularity.

In this paper, motion estimation, criteria for movement estimation performance. Then propose Hash-based algorithm which enables frame level block searching which enable block selection is employed to select the best candidates to achieve the best coding efficiency.

Full search algorithm works better in terms of PSNR values, reconstructed image quality, and average number of search points.

REFERENCES

[1] Bichu Vijay, Ganapathi Hegde, Sanju S, “Fast Block-Matching Motion Estimation Using Modified Diamond Search Algorithm” Amrita School of Engineering, Bangalore, India International Journal of Advanced Computer Engineering and Communication Technology (IJACECT), ISSN (Print): 2319-2526, Volume -3, Issue -1, 2014.

[2] Chung-Ming Kuo, Nai-Chung Yang, I-Chang Jou1 and Chaur-Heh Hsieh, “A Novel Prediction-Based Asymmetric Fast Search Algorithm for Video Compression”, Ming Chung University Gui-Shan, 333, Taoyuan, Taiwan, R.O.C.

[3] Weijia Zhu, Wenpeng Ding, Jizheng Xu, Yunhui Shi, and Baocai Yin “Hash-Based Block Matching for Screen Content Coding”, IEEE Transactions On Multimedia, Vol.17, No.7, July 2015 935.

[4] Koga, T, Iinuma, K, Hirano, A, Iijima, Y. & lshiguro, T. (1981). “Motion-compensated interframe coding for video conferencing”, in Proc. Nat. Telecommunication Conf. , New Orleans, LA, pp. G5.3.1-G5.3.5.

[5] R. Li, B. Zeng, and M. L. Liou, “A new three-step search algorithm for block motion estimation” , IEEE Trans. Circuits Syst. Video Technology, vol. 4, no. 4, (1994) August, pp. 438–442.
[6] L. M. Po and W. C. Ma, “A novel four-step search algorithm for fast block motion estimation”, IEEE Trans. Circuits Syst. Video Technol., vol. 6, no. 3, June, pp. 313–317.
[7] Zhu, S. & Ma, K. K. (2000). A new diamond search algorithm for fast block-matching motion estimation, IEEE Trans. Image Process. , Vol. 9, No. 2, pp. 287-290, ISSN: 1057-7149.
[8] S. Goel, Y. Ismail, and M. A. Bayoumi, “Adaptive search window size algorithm for fast motion estimation in H.264/AVC standard”, in Proc. Midwest Symp. Circuits Syst., (2005) August, pp. 1557–1560.
[9] Yih-Chuan Lin and Shen-Chuan Tai “Fast Full-Search Block-Matching Algorithm for Motion-Compensated Video Compression”, IEEE Transactions On Communications, Vol. 45, No. 5, May 1997 527.
[10] J. S. Kim and R. H. Park, “A fast feature-based block matching algorithm using integral projections”, IEEE J. Select. Areas Commun., vol. 10, pp. 968–971, June 1992.
[11] Y. H. Fok and O. C. Au, “An improved fast feature-based block motion estimation” ,in Proc. IEEE 1994 Int. Conf. Image Processing, 1994, pp. 741–745.
[12] H. B. Park and C. Wang, “Image compression by vector quantization of projection data”, IEICE Trans. Inform. Syst., vol. E75-D, pp. 148–155, Jan. 1992.
[13] K. H. Jung and C. Wang “Projective image representation and its application to image compression,” IEICE Trans. Inform. Syst., vol. E79-D, pp. 136–142, Feb. 1996.
[14] N. Purnachand, L .N. Alves, and A. Navzrro, “Fast motion estimation algorithm for HEVC”, in Proc. IEEE Int. Conf. Consum. Electron, Sep. 2012, pp. 34–37.
[15] C.-M. Kuo, Y.-H. Kuan, C.-H. Hsieh, and Y.-H. Lee, “A novel prediction-based directional asymmetric search algorithm for fast block matching motion estimation”, IEEE Trans. Circuits Syst. Video Technol., vol. 19, no. 6, pp. 893–898, Jun. 2009.
[16] W. Zhu, W. Ding, J. Xu, Y. Shi, and B. Yin, “2-D dictionary based video coding for screen contents”, in Proc. Data Compression Conf., 2014, pp. 43–52.
[17] A. D. Houghton, “Cyclic redundancy checking”,in The Engineer’s Error Coding Handbook. New York, NY, USA: Springer, 1996, pp. 12–24 IEEE Transactions On Multimedia, Vol.17, No.7, July 2015.
[18] Abdelrahman Abdelazim “Fast Motion Estimation Algorithms For Blockbased Video Coding Encoders”, Applied Digital Signal And Image Processing Research Centre, School Of Computing, Engineering And Physical Sciences, University Of Central Lancashire 2011.
[19] Prashant A.Bhalge, Salim.Y.Amdani “Categories for Fast Block Matching Algorithm” ,International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-4 Issue-7, December 2014.
[20]P. A. Bhalge, S. Y. Amdani, “Fast Block Based Motion Estimation using Various Search Patterns” , International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-4 Issue-7, December 2014.
[21] Mr. P. N. Sharma, P.A.Bhalge, “A Survey on Motion Estimation Using Adaptive Weighted Mean Filter”, IJSTE - International Journal of Science Technology & ngineering , Volume 2 , Issue 08 , February 2016.