

**FAST MOTION ESTIMATION ALGORITHMS FOR HEVC VIDEO COMMUNICATION**

K.MOHAN KUMAR

17BEC7034

**URE001201902**

Supervised by

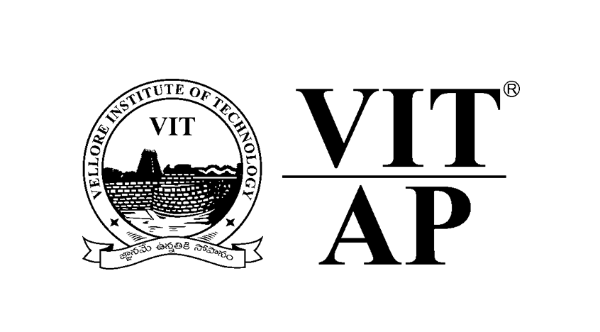
Dr. Purnachand Nalluri

School of Electronics Engineering

Vellore Institute of Technology –AP

School of Electronics and Communication Engineering (SENSE)

Inavolu, Amravati



Date: March 7, 2020

**TO WHOMESOEVER IT MAY CONCERN**

This is to certify that **Mr. K. Mohan Kumar** (Reg. No: 17BEC7034, URE: URE001201902) has completed his Undergraduate Research Experience-001 project work on the topic “**FAST MOTION ESTIMATION ALGORITHMS FOR HECV VIDEO COMMUNICATION**” under the supervision of me in VIT-AP University, Amaravati, Andhra Pradesh, India.

**Dr. Purnachand Nalluri**

**Associate Professor**

**School of Electronics Engineering**

**VIT-AP University, AP, India.**

**Abstract**

Over last few years, the demand of digital video is increasing and used in many applications like storage, surveillance, web streaming, broadcasting, video communications and conferencing. It is more complicated and challenging to process as the amount of data required for video is huge. So, Video Compression is necessary to reduce the size of video file for efficient and fast transmission.

Video compression techniques like Block based video encoders typically use Block Matching Algorithms (BMAs) to perform Motion Estimation. If the Motion Estimation algorithm searches each and every block in the Search Window then it is a Full Search algorithm. Searching every block in the entire Search Window increases the complexity of the encoder. Hence video encoders employ fast Motion Estimation algorithms which skip most of the blocks that are unlikely to be the optimum Motion Vector. But by using fast search algorithms there may be some degradation and also decrease in the output video quality as the estimated optimum Motion Vectors may not be accurate enough. Hence a good fast ME algorithm is necessary to decrease the Motion Estimation complexity but with negligible loss in compression ratio and output video quality.

This work mainly focuses on implementing the Fast Motion Estimation algorithms like Full search, Search point reduction and Cross Hexagon Search, Compensation of motion vectors to reconstruct the frame and also analyze the results of algorithms such as time taken to estimate the motion vectors, average number of computations per macro-block and Peak-to-Signal Ratio (PSNR) for each algorithm

**Acknowledgement**

I hereby thank Vellore Institute of technology –AP University for Undergraduate Research Experience (URE) – 001 and providing me the great opportunity to work on this project. Many thanks to my guide, Dr. Purnachand Nalluri, who has been a great source of inspiration, his timely support, creative ideas and guidance has helped in the successful completion of the same. Also, thanks to the committee members Dr. Jagadish Chandra Mudiganti, Dr. Achintya Kumar Sarkar, Dr. Ravindra Dhuli , who offered guidance and support.

I also thank my parents and friends who offered lot of support and encouragement to complete the work.

**Contents**

Abstract……………………………………………………………... (3)

Acknowledgement………………………………………………...… (4)

List of Abbreviations………………………………………………... (6)

List of Figures/tables………………………………………………... (7)

Introduction……………………………………………………….… (8)

-Need for video compression…………………………………..… (8)

-Block-based video encoder system…………………………...… (9)

-Motion Estimation………………………………………………. (10)

ME Algorithms…………….……………………………………….. (12)

-Classification of algorithms………………………………….… (12)

-Full Search Algorithm…………………………….…………… (12)

-Fast Search Algorithms… …………………………………….. (13)

-Cross Hexagon Search Algorithm…………………………….. (20)

Implementation…………………………………………………….. (22)

Results……………………………………………………………… (26)

Conclusion………………………………………………………….. (29)

Future Scope…………………………………………………..……. (29)

References……………………………………….............................. (30)

# **List of Abbreviations**

ME Motion Estimation

MV Motion Vector

BMA Block Matching Algorithm

SAD Sum of Absolute Difference

PSNR Peak-Signal-to-Noise Ratio

SW Search Window

SEA Successive Elimination Algorithm

ADS Absolute Difference between Sums

TSS Three Step Search

4SS Four Step Search

DS Diamond Search

HS Hexagonal Search

LDSP Large Diamond Search pattern

SDSP Small Diamond Search pattern

CS Cross Search

TZS Test Zonal Search

**List of Figures**

Figure 1: Black Diagram of Block-based video encoder

Figure 2: Motion Estimation

Figure 3: Forward and Backward Prediction

Figure 4: Illustration of TSS Algorithm

Figure 5: Illustration of DS Algorithm

Figure 6: Illustration of Four Step Search Algorithm

Figure 7: Illustration of Logarithmic Search Algorithm

Figure 8: Illustration of Cross Search Algorithm

Figure 9: Illustration of Hexagonal Search Algorithm

Figure 10: Illustration of Test Zone Search Algorithm

Figure 11: TZ Search Algorithm flow

Figure 12: Sequence frames

Figure 13: Illustrartion of Search Window

Figure 14: MATLAB Environment

**List of tables**

Table 1: Time, Computations and PSNR results of Foreman Sequence

Table 2: Time, Computations and PSNR results of Bus Sequence

Table 3: Time, Computations and PSNR results of Stefan Sequence

Table 4: Time, Computations and PSNR results of Coastguard Sequence

Table 5: Time, Computations and PSNR results of Hall-monitor Sequence

**Introduction**

The revolution of digital technology has created a huge impact on almost all types of industries in the areas of arts, entertainment, communications, marketing, media etc. and has marked the beginning of information age. With more and more advances in computational speed of digital computing devices, there is an ever growing increase in generation and demand for digital information including data, audio and video. Amongst all these multimedia data, the digital video is more complicated and challenging to process (which is the used in many applications including storage, surveillance, web streaming, broadcasting, video communications and conferencing) as the amount of data required for video is huge compared to data for audio information.

**Need for Video Compression**

Video is a group of frames displayed per second i.e. frames per second. Consider 30 frames per second video; we know that Image is a matrix of pixels. The size of each pixel is 8 bit. Consider resolution of each video frame is 1280 X 720. The total size of frame would be 1280\*720\*3\*8 bits which is approximately 22 MB. Thus, total size of video for one second will be around 663 MB which is huge. Thus Image processing and video compression is required to reduce the size of video.

Video compression is the process of encoding a video file in such a way that it occupies less space than the original file. The size of video file is reduced by eliminating redundant and non-functional data from the original video file. A raw video contains immense amount of data. It contains a lot of redundant information which contains duplicate and irrelevant data. Communication and storage capabilities are limited and expensive [3]. Video compression can reduce the storage space by removing the unnecessary information, which reduces our investment on storing the video.

Video compression is necessary to achieve a good quality video with minimum size. Compression of video plays a crucial part in online video streaming, broadcasting and other storage applications. The higher the compression of video, the lesser is the memory needed to store it.

**Block-based video encoder system**

Block based video coding is one of the widely used technique in video compression. In block based video coding, each video frame is split in to coding blocks. The ME block predicts and estimates motion between frames and generate Motion Vectors (MVs) using ME algorithms.

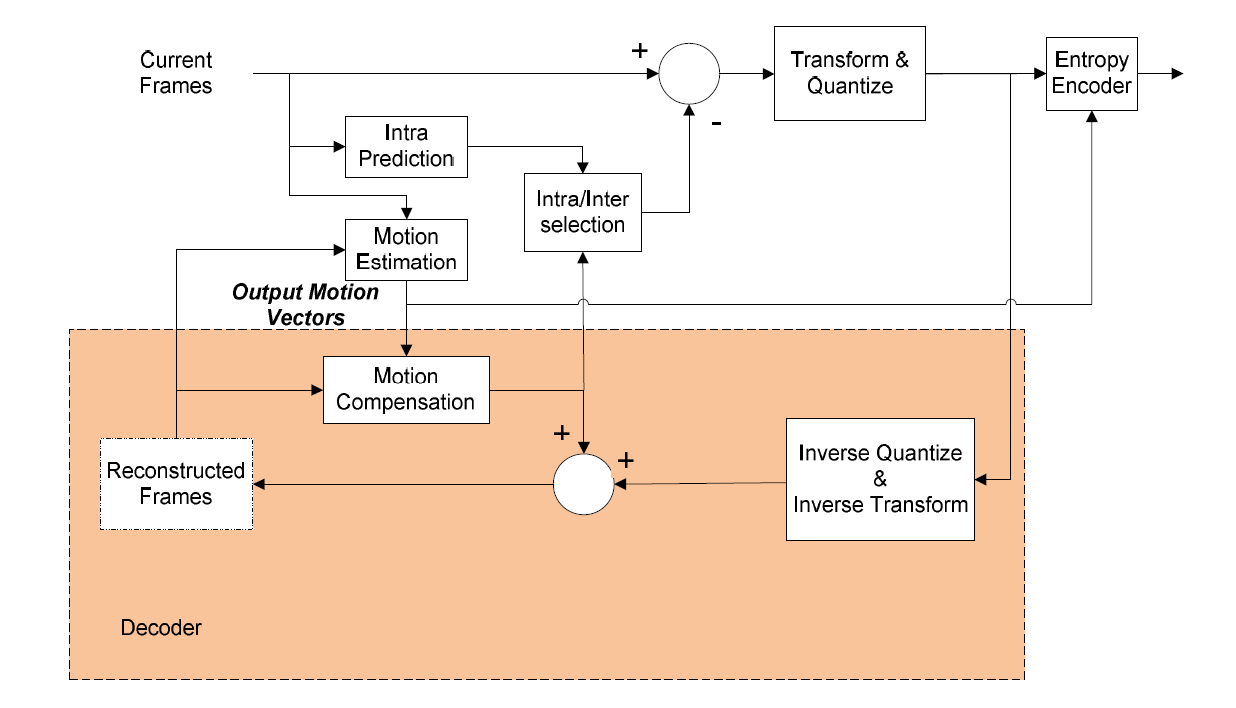


Fig. 1 Black Diagram of Block-based video encoder

Each coding block is predicted, transformed, quantized and entropy encoded. Each frame is split into various coding blocks and then the blocks of first frame in a video frame sequence is intra predicted and encoded (prediction of image blocks within the frame) and the rest of the frame’s blocks are either intra predicted or inter predicted (prediction between frames using Motion Estimation (ME) and motion compensation blocks). Nevertheless, in interprediction, in every frame the first block of slice (group of blocks) can be intra-coded and depends on the mode-decision algorithm. The ME block predicts and estimates motion between frames and generate the Motion Vectors (MVs). The MVs are entropy encoded and also sent to motion compensation block. The motion compensation block uses these MVs to generate motion compensated frames. These motion compensated frames are subtracted from the original frames (current frames) to generate residual frame blocks. This residual information is transformed, quantized and then entropy encoded. To generate identical predicted information in the decoder side, the encoder typically includes a decoding loop to reconstruct the original frames using predicted information. The decoding blocks are shown in dark color in Fig 2. To do this, the decoder takes the quantized information and passes it through inverse quantization and inverse transformation blocks. Then, the obtained data is added to the predicted data to reconstruct the subsequent video frames.

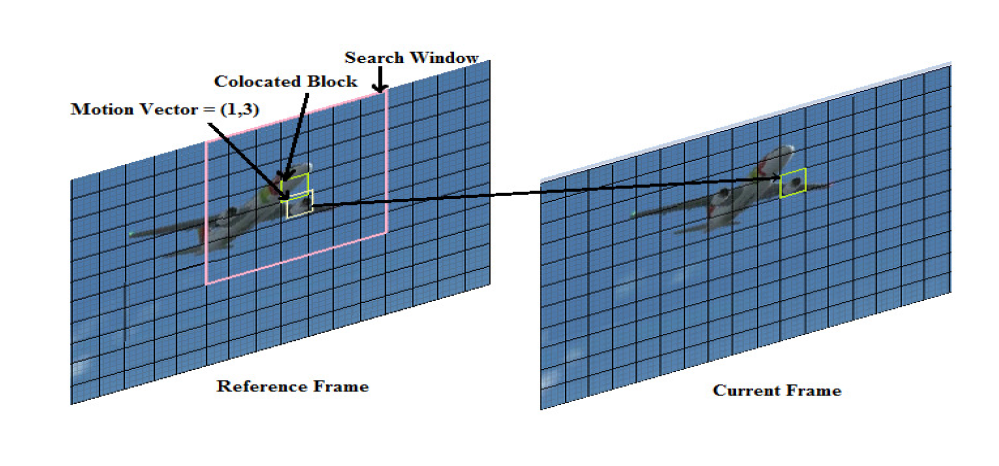


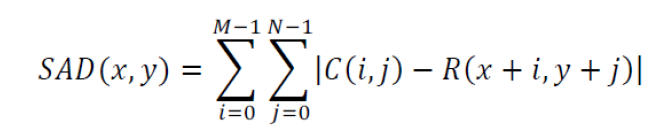
Fig. 2 Motion Estimation

**Motion Estimation**

Motion Estimation (ME) is the essential task in block based video encoders. It contributes to reduce the overall bitrate of a video signal by predicting and estimating the Motion Vectors (MVs) for each block in every frame. A good estimate of motion in a frame generates less entropy information (for residual frame blocks) and fewer bits to encode it and hence the compression ratio will be increased. So, for each block in every frame, the main task of ME is to estimate the motion content by finding the best matched block in the previously encoded frame (reference frame) region of interest (also called search window).

For every block of the current frame, a new Search Window (SW) is defined and the ME algorithm searches for the best matched block using a predefined cost function i.e. Sum of Absolute Difference (SAD). The final output of the ME is the coordinates of the optimal MV and its cost.

The minimum value of SAD can be used to determine the Motion Vector.



Where C represents the pixel values of Current frame and R represents the pixel values of Reference frame. The pixel values ranges from 0-255.

The motion estimation and prediction can be done using both past and/or future reference frames. Prediction using past reference frames is called *Forward prediction*, while the one using future reference frames is called *backward prediction*. This is illustrated in Fig. 3. Both the forward or backward prediction isdone by storing the reference frames (past or future frames) in reference frame buffers.

Using bi-directional ME, the compression efficiency can be increased at the cost of increase in complexity.

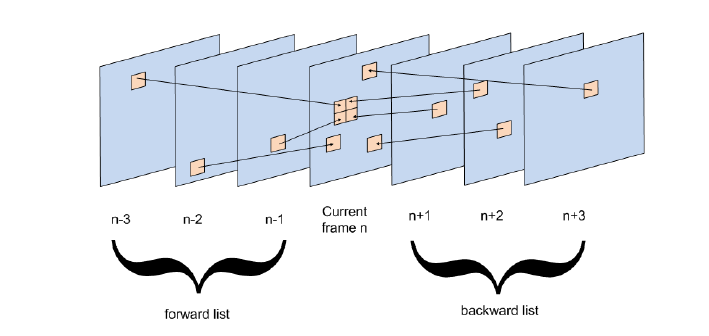


Fig. 3 Forward and Backward Prediction

**ME Algorithms**

The motion estimation unit together with motion compensation unit performs inter-picture prediction by converting the frames into motion vectors and motion predicted blocks. The motion estimation unit estimates the motion vectors of each block in a frame (except in intra frames) while the motion compensation unit uses these motion vectors and generates motion compensated (predicted) frames.

These motion compensated frames are then subtracted from the original video frames to get the residual frames and processed further. Typically, the motion estimation block uses block matching algorithms to find the motion vector of each block in a frame. The motion compensation unit performs interpolation (using functions like weighted-prediction) on reference picture to form motion compensated frame for every current frame.

**Classification of algorithms**

Motion estimation Algorithms are classified in to two categories.

**Full Search Algorithm**: Search each point in the entire search window and estimate the motion vectors.

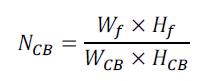
It gives the best video quality and the lowest bitrate (highest compression ratio). Full search method is very time consuming and complicated in terms of coding.

**Fast Search Algorithms**: Instead of searching all the blocks in the search window, the motion estimation algorithms can be designed to choose a certain fixed or varying pattern of blocks which gives the closest matching block.

Fast search algorithms are meant to reduce the search complexity by skipping the blocks which are less likely to estimate the Motion Vector.

**Full search Algorithm**

The full search method searches every possible location in the entire search window. As a result the algorithm finds the best matching block for every block in all the frames of video and gives the highest PSNR. But the computational time is very high. Let R be the search range set for the ME algorithm. Then the maximum possible number of search points for the SW are (2R+1)\* (2R+1). Let WCB x HCB be the width and height of the current block and let Wf x Hf be the frame width and height. Then there will be NCB coding blocks for each frame will be area of total frame/area of each block so it can be calculated using equation:



Let *Np* be the number of sub-partitions for each current block CTU. Let Nfbe the number of reference frames for each block that the ME has to be performed. Then the total number of search points can be calculated using equation:



The most computationally expensive block matching algorithm of all is the Full Search/Exhaustive Search since it calculates the cost function at each possible location in the search window. As a result it finds the best possible match and gives the highest PSNR amongst any block matching algorithm. The obvious disadvantage of Full Search is that the larger the search window gets the more computations it requires and consumes more time for estimation the motion vectors

**Fast Search Algorithms**

Fast search algorithms are meant to reduce the search complexity by skipping the blocks which are less likely to the Motion Vector. Fast ME algorithms achieve gain in speed with almost same PSNR (video quality) and bitrate (compression ratio) compared to Full Search algorithm.

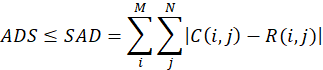
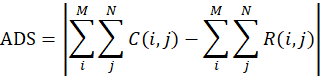
The fast ME algorithms can be classified into many types. Based on the loss of output video quality, the algorithms can be classified into lossless fast ME algorithms and lossy fast ME algorithms. The Successive Elimination Algorithm (SEA), MLSEA (Multi level Successive Elimination Algorithm), PDE (Partial Distortion Elimination) etc. are some examples of lossless fast ME algorithms.

In lossy fast ME algorithms, there will be huge gain in speed but slight decrease in PSNR and bitrate compared to Full Search algorithm. Some of the examples are search area sampling techniques, pixel decimation techniques, hybrid algorithms (include more than one fast ME algorithms) etc.

Examples of Fast Motion Estimation Algorithms

**Successive Elimination Algorithm**

In Successive Elimination Algorithm(SEA), there are two stages. In the first stage it calculates the sum of intensities of all pixels between current and reference block i.e Absolute Difference between Sums (ADS) [3].In the second stage, the algorithm estimates the search point based on an inequality equation shown below .



For any two blocks of equal size MxN, the ADS is always less than or equal to their SAD. If the search blocks has larger ADS then the current minimum distorted search point, can be omitted from the search process. So that ME can be faster. There is another algorithm called Multilevel SEA which extends the concept of SEA in a multilevel hierarchical way which ultimately reduces the complexity.

**Hierarichal and Multiresolution Algorithms**

In hierarchical search algorithm, the frame or search window is down-sampled to lower resolution, usually multiresolution samples of a frame are formed using sampling filters like Low Pass Filter . The hierarchical ME algorithms are also used in a frame interpolation for frame ratte up conversion applications[6].

**Pixel Decimation Block Matching Algorithms**

These block matching algorithms uses only few pixels with pre-defined pattern in the blocks to estimate the Motion Vectors so that the computational time for ME can be reduced. There are many approaches like hexagonal pattern, spiral pattern etc. and N-Queen pattern is proposed because it improves coding effienciency with almost similar vedio quality compared to existing pixel-decimation algorithms[5][6].

**Search Point Reduction Algorithms**

This algorithms reduce the number of search points in the search window to reduce the complexity of ME process.

**A. Three Step Search (TSS) Algorithm**

The three step search starts with searching the centre point, which is the co-located point of current block in reference frame. Then,it starts searching the surrounding eight locations, with step size 4. The point which has the minimum error (minimum SAD or SSD), is taken as reference centre point to the next step. If the lowest cost is at the centre, then the motion search is stopped and the centre point is taken as the motion vector. Otherwise, the algorithm proceeds to second step[10].

In the second stage, the step size is reduced to 2, and the search is performed around the surrounding eight positions. The best matching point is again taken as reference, to the third step and the search is performed with step size one. The step is the final step, since there is no further search possible with step size less than one. The best matched point in the third step is the final best matching motion vector.

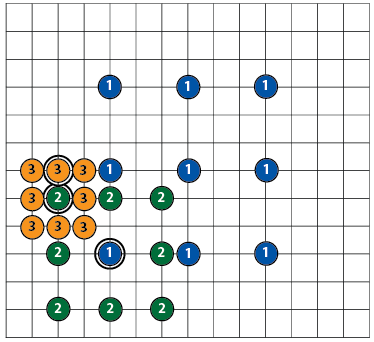


Fig.4 Illustration of TSS Algorithm

**B. Diamond Search (DS) Algoritrhm**

In DS algorithm , diamond shaped pattern is used with step size of one [8]. This algorithm gives a good PSNR. There are two types Diamond patterns, one is Large Diamond Search pattern(LDSP) and the other is Small Diamond Search Pattern(SDSP) The first step uses LDSP and the last step uses SDSP. The algorithm continues using LDSP, until the minimal point comes out to be at the origin of the LDSP

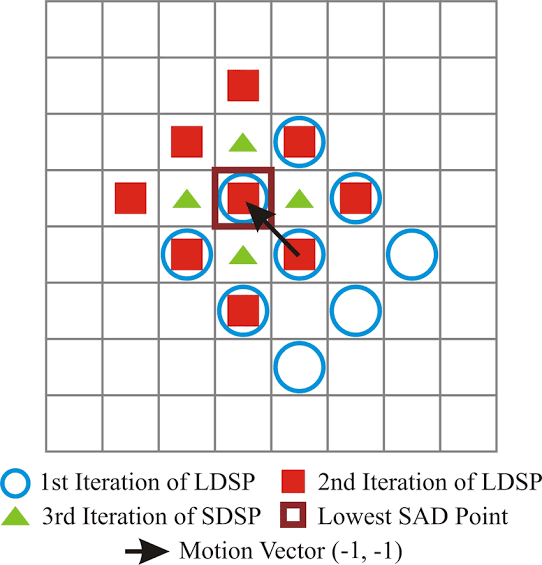


Fig. 5 Illustration of DS Algorithm

**C. Four Step Search (4SS)**

This algorithm could have 2, 3 or 4 search steps. 4SS performs better than the popular TSS in terms of motion compensation errors.4SS takes more time to estimate the motion vectors but produce better PSNR than TSS.

****

Fig. 6 Illustration of 4SS Algorithm

**D. 2D-Logarithmic Search Algorithm**

This algorithm is similar to TSS Algorithm. It starts searching with five locations along the centre of the edges of the search window, and the centre point. The minimal point is taken as the reference point for the next step and the search area is reduced by a factor of two. The algorithm continues to search until the search area is reduced to 3x3, and in the last step the search is performed in the entire nine locations.

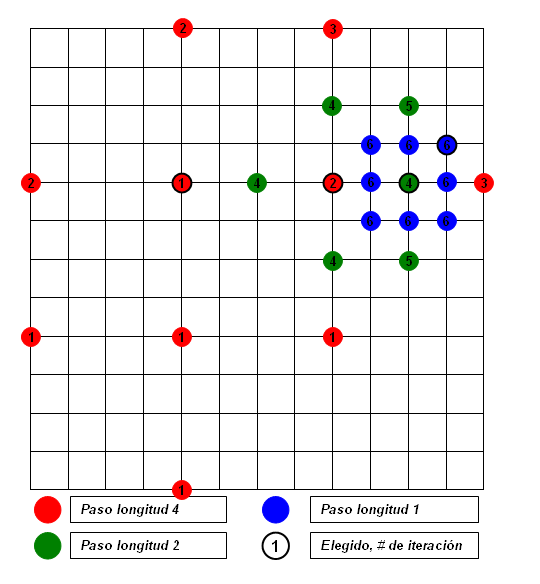


Fig.7 Illustration of Logarithmic Algorithm

**E. Cross Search (CS) Algorithm**

The cross search algorithm is similar to Logarithmic search, except that the search is performed in cross (‘X’) pattern instead of diamond (plus ‘+’) pattern.



Fig.8 Illustration of CS Algorithm

**F. Hexagonal Search (HS) Algorithm**

HS Algorithm is similar to DS algorithm. This algorithm uses Hexagon pattern instead of square pattern[9]. Hexagonal Pattern search algorithm need to evaluate only 3 new checking points for each new search step compared with 3 or 5 in the Diamond Search. Hence, number of search points will be less than the Diamond search. DS was sensitive to motion vectors in different directions – since its eight checking points have different distances from the centre point the advancing speed for the DS per step is sqrt(2) pels horizontally and vertically but only 2 pels diagonally. Ideally a circle-shaped search pattern with a uniform distribution of a minimum number of search points was more desirable to achieve the fastest search speed uniformly. This search pattern should have a minimum number of search points distributed uniformly where each search point is used equally with maximum efficiency and where the redundancy among search points should be removed maximally. As a result, hexagon based search pattern (HEXBS) has a more circle-approximated pattern. The pattern consists of six endpoints with the two horizontal points being 2 pels from the centre and the remaining four points sqrt(5) pels from the centre - thus the six endpoints are approximately uniformly distributed. So, analysis showed a speed improvement rate of as high as over 80% for locating some motion vectors in certain scenarios.

It has three steps:

***a) Starting*:** Hexagonal pattern points with centre (7 points) are checked initially. If the minimal point is found to be at the centre, then the algorithm skips to final stage, the ending step.

***b)Searching:*** The minimal point (minimal distorted block), obtained in the starting step is taken as centre, and a hexagon is again formed by adding only three new search locations. With the obtained hexagonal pattern, the minimal point is again searched. If the minimal point is found to be at the centre, then the algorithm skips to the third step ‘ending’ otherwise this step is repeated continuously[9].

***c)Ending:*** In this final step, the search pattern is switched from large hexagon to small hexagon that needs to be covered only four points.The minimal point obtained in this small hexagon is taken as the final motion vector block[2][3].

 ******

Fig.9 Illustration of HS Algorithm

**G. Test Zone Search Algorithm**

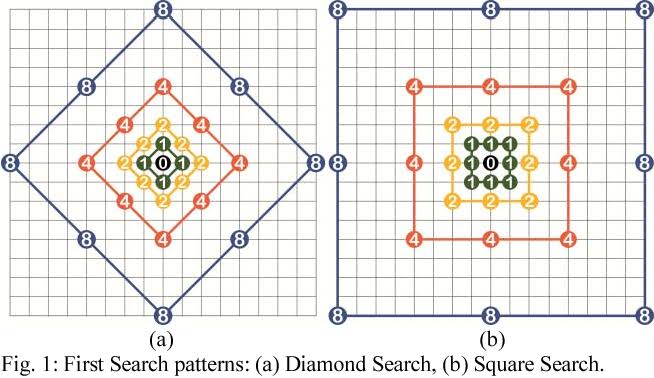
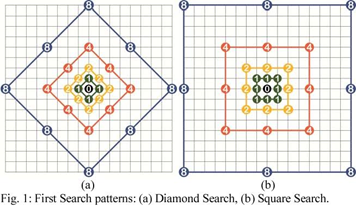
The TZ search algorithm is one of the most efficient ME algorithm . It has four stages.

*A.Motion Vector prediction:*

TZS algorithm employs four predictors. They are median predictor, left predictor, up predictor and upper right predictor[1][3]. The minimum of these predictors is selected as a starting location for further search steps.

*B.Initial Grid Search:*

In this step, the algorithm searches the search window in diamond or square pattern with different stride lenghts ranging from 1 through 64 in multiples of 2. Each grid pattern contains eight search points. The minimum MV is taken as the centre search point for further steps. The stride length for this minimum distortion point is stored in variable say ‘uiBestDistance’.

*Diamond Pattern Square pattern*

Fig.10 Illustration of TZS Algorithm

*C.Raster Search:*

In this step, a predefined value i.e 5 (stored in a variable say iRaster) is set before compilation of the code. This value is used as the sampling factor for seach window . The condition for performing this raster search is that uiBestDistancemust be greater than iRaster. If this condition is not satisfied, the algorithm will skip this step. If this step is processed, then uiBestDistance is changed to iRaster value[3].

*D.Raster/Star Refinement:*

This step is a fine refinement of the motion vectors obtained from the previous step. In general, either of Raster or Star refinement is enabled for fast computation. In both of these refinements, either 8-point square and diamond pattern is used. They differ in their search operation. The raster-refinement will search by down-scaling the uiBestDistance value (obtainedfrom raster search) by 2 in every step of the loop, till uiBestDistance equals to zero. The star refinement is similar to step 2 except for change in starting search location. The whole refinement process will only start if uiBestDistance is greater than zero. After every loop, the new stride length is stored in variable uiBestDistance. The search stops when uiBestDistance equals to zero, which means that the obtained search point is the optimal MV[3].

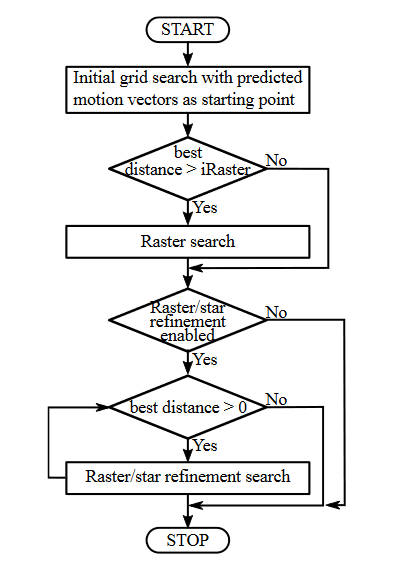
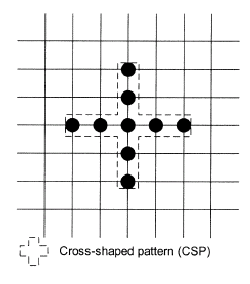


Fig.11 TZS Algorithm flow

**Cross Hexagonal Search(Proposed Algorithm)**

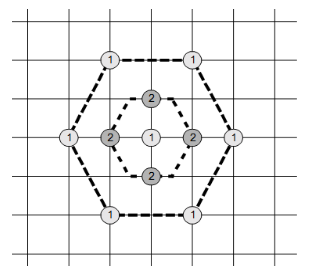
Step 1: (Starting) A minimum SAD is found from the five search points of the Cross Search Pattern (CSP) with step size of 1. If the minimum SAD point occurs at the center of the CSP, the search stops. Otherwise, go to Step 2.



Mostly, motion in frames is along the horizontal or vertical direction. So, Cross Search Pattern (+) is considered.

Step 2: Similar to Step1 with step size of 2. If the minimum SAD point occurs at the center of the CSP, the search stops. Otherwise, go to Step 3.

Step-3: A minimum SAD is found from the Nine search points of the Hexagonal Pattern with step size of 2



Step 4: (Ending) With the minimum SAD point in the previous step as the center, a new SDSP is formed. The location of the minimum SAD point found for this step is the motion vector.

Mostly, Motion in frames of a sequence or video is along the horizontal or vertical direction. So, Cross Search Pattern (+) can estimate the motion vectors with less computation time. Hexagonal Search also less computations to estimates the MV’s So, Cross Hexagonal Search algorithm saves the computation time and also requires less number of search points for motion estimation. This algorithm also produces desirable PSNR.

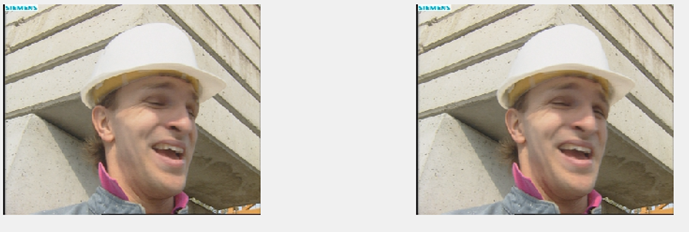
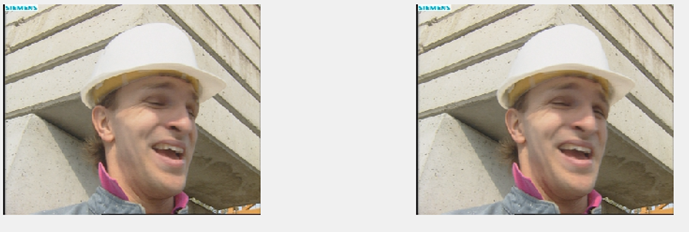
**Implementation**

The following Motion estimation algorithms have been implemented in MATLAB: Full search,Three Step Search,Diamond Search,Hexagonal Search and Cross Hexagonal Search

ME algorithms estimates the motion between frames and produce motion vectors which are used for compensation. These motion vectors are added to the reference frame to result a new frame called compensated frame or reconstructed frame. The difference between compensated frame and current frame is calculated to estimate the Peak-Signal-to-Noise ratio(PSNR).

**Video Sequences used for analysis**

1. Foreman sequence (352 x 288)

Reference frame Current frame

Fig. 12 Sequence frames

2. Bus sequence (352 x 288)



3. Stefan sequence (352 x 288)



4. Coastguard sequence (352 x 288)



5. Hall-monitor sequence (352 x 288)



Thirty frames of above sequences are considered for motion estimation

Block size is 4 x 4 and Search window size is 16 x 16.

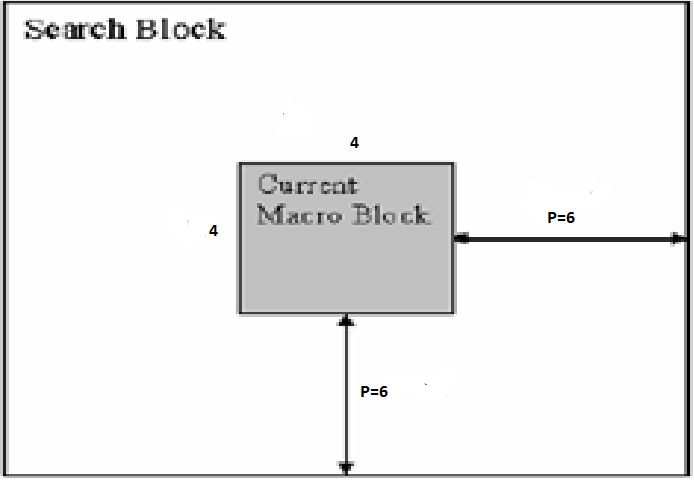


Fig.13 Illustration of Search Window

**Algorithms Execution**

motionsEstAnalysis.m – the main script to execute all algorithms is run in the MATLAB command window. Initially 2 frames of a particular video sequence are loaded into the workspace – the first is the reference frame and the second is the frame to be predicted (encoded). The first block match algorithm is called the block distortion measure (BDM) used is the mean absolute difference (MAD). The macroblock size is set at 16 pixels x 16 pixels and the maximum displacement in the search area is ±7 pixels in both the horizontal and the vertical directions. A frame difference of 2 was used in calculating predicted frames.

The algorithm function called returns the motion vector matrix for the predicted frame – one motion vector for every macroblock in the frame. The average number of points searched to calculate each motion vector within the predicted frame is also returned.

The motion vector matrix is then input into the motionComp.m function which creates the motion compensated image from each motion vector and its corresponding macroblock in the reference frame.

The PSNR of the motion compensated image with respect to the original frame is then calculated and recorded by calling the imgPSNR.m function – one value for each predicted frame.

The main script also contains code to produce tic toc function to estimate the time taken by the algorithms to estimate the motion vectors and also PSNR.

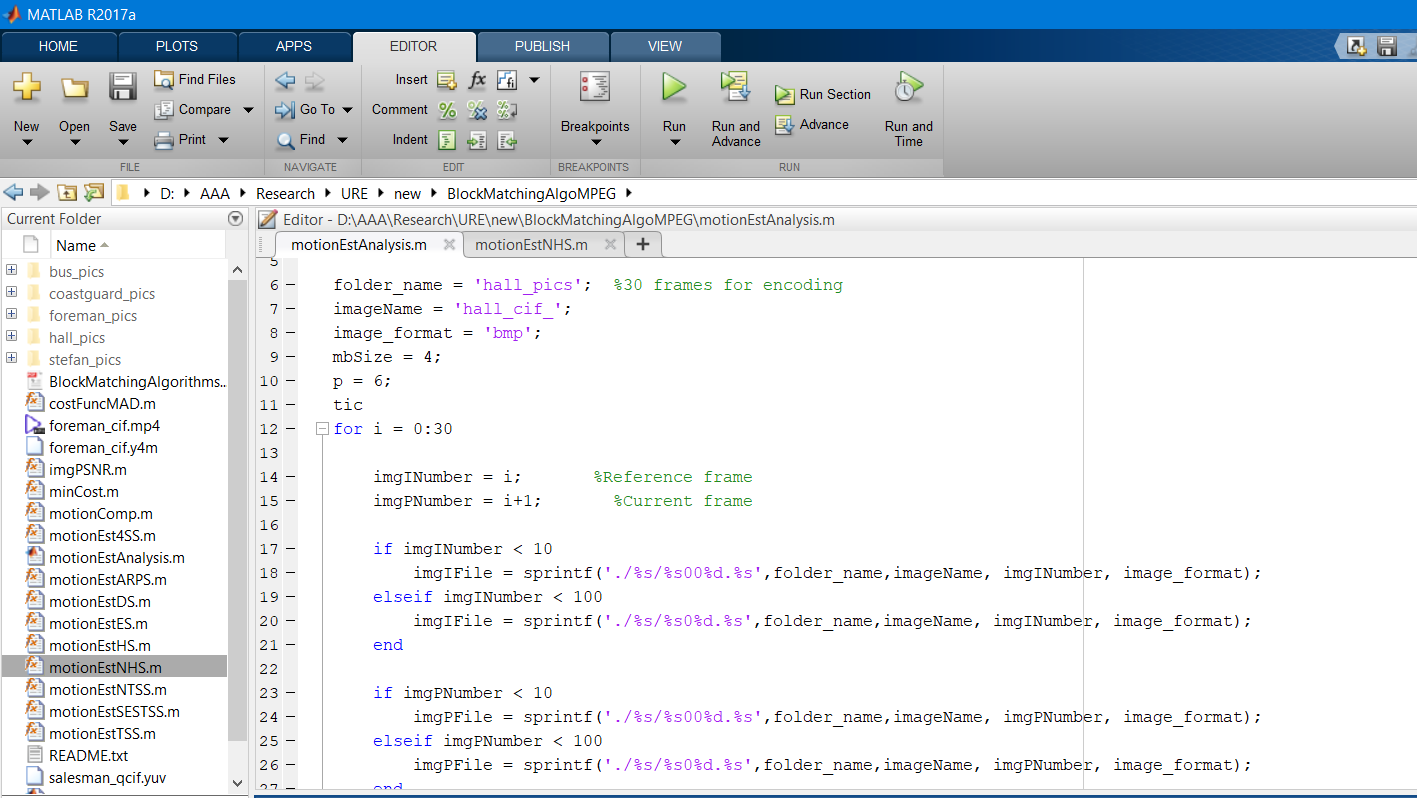
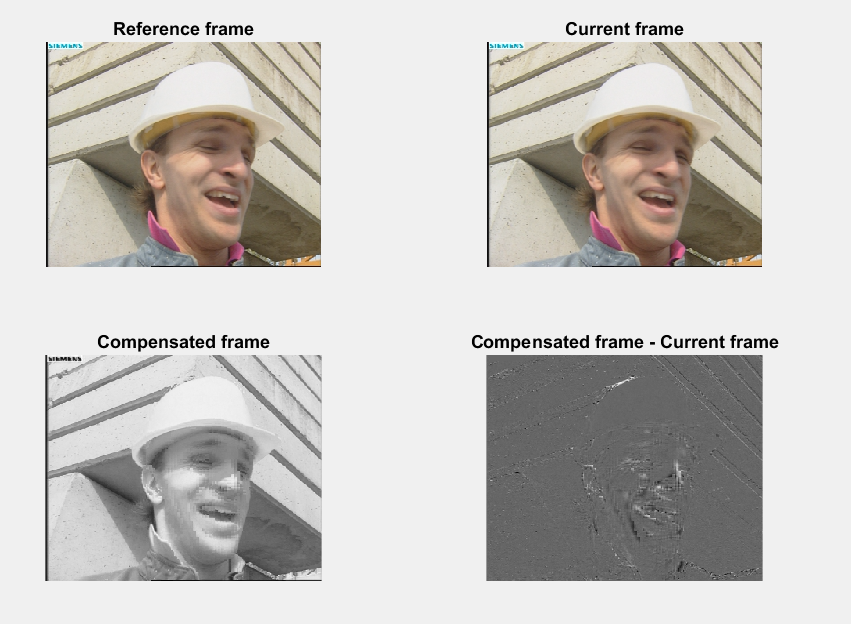


Fig.14 MATLAB environment

**Results**

**Processor Specifications:** Intel i5, 7th Generation, RAM: 8 GB, GPU: Intel HD620

Foreman Sequence (352 x 288)



Foreman Sequence (352 x 288) and 30 frames sequence

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Time taken for Motion Estimation (sec)** | **Average number of Search Points for a macro-block** | **Average**  **PSNR(dB)** |
| **Full**  **Search** | 84.05 | 169 | 37.20 |
| **Three Step**  **Search** | 10.83 | 17 | 35.02 |
| **Diamond**  **Search** | 10.7 | 18 | 35.88 |
| **Hexagonal**  **Search** | 8.94 | 14 | 35.35 |
| **Cross Hexagonal**  **Search** | 8.61 | 12 | 35.85 |

Bus sequence (352 x 288) and 30 frames sequence

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Time taken for Motion Estimation (sec)** | **Average number of Search Points for a macro-block** | **Average**  **PSNR(dB)** |
| **Full**  **Search** | 70.32 | 169 | 26.24 |
| **Three Step**  **Search** | 9.62 | 17 | 21.38 |
| **Diamond**  **Search** | 12.11 | 21 | 22.5 |
| **Hexagonal**  **Search** | 9.71 | 15 | 21.3 |
| **Cross Hexagonal**  **Search** | 12.12 | 19 | 22.03 |

Stefan sequence (352 x 288) and 30 frames sequence

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Time taken for Motion Estimation (sec)** | **Average number of Search Points for a macro-block** | **Average**  **PSNR(dB)** |
| **Full**  **Search** | 71.84 | 169 | 28.64 |
| **Three Step**  **Search** | 9.27 | 17 | 25.63 |
| **Diamond**  **Search** | 10.02 | 18 | 26.31 |
| **Hexagonal**  **Search** | 8.52 | 17 | 25.63 |
| **Cross Hexagonal**  **Search** | 8.58 | 13 | 26.16 |

Coastguard Bus sequence (352 x 288) and 30 frames sequence

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Time taken for Motion Estimation (sec)** | **Average number of Search Points for a macro-block** | **Average**  **PSNR(dB)** |
| **Full**  **Search** | 71.45 | 169 | 33.01 |
| **Three Step**  **Search** | 9.23 | 17 | 32.28 |
| **Diamond**  **Search** | 10.04 | 18 | 32.4 |
| **Hexagonal**  **Search** | 7.57 | 12 | 30.12 |
| **Cross Hexagonal**  **Search** | 9.05 | 13 | 32.3 |

Hall-monitor sequence (352 x 288) and 30 frames sequence

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Time taken for Motion Estimation (sec)** | **Average number of Search Points for a macro-block** | **Average**  **PSNR(dB)** |
| **Full**  **Search** | 71.58 | 169 | 38.48 |
| **Three Step**  **Search** | 8.88 | 17 | 37.5 |
| **Diamond**  **Search** | 9.5 | 16 | 37.80 |
| **Hexagonal**  **Search** | 8.01 | 13 | 37.33 |
| **Cross Hexagonal**  **Search** | 7.01 | 10 | 37.57 |

**5. Conclusion**

Cross Hexagonal Search Algorithm estimated the motion vectors with less computation time than other algorithms. The minimum number of search points (computations) are also less compared to other algorithms with almost same PSNR.

Motion estimation is one of the most complex tasks in block based video encoders. Especially in the encoder of the latest video coding standard HEVC, the complexity of ME is further increased due to increase in the block size to 64x64 pixels. At the search window level, the search range also increased to 64 in HEVC and hence the number of search points also increases, where for each point the rate distortion cost has to be calculated for all block sizes. Hence to reduce the ME complexity, reducing the number of search points effectively is one of the main approaches used. Thus fast search algorithms are used to reduce the search points.

Full search algorithms consumes more time compared to fast search algorithm, however fast search algorithms also produce high PSNR with negligible loss.

**6. Future Scope**

The ME complexity can be further decreased by using an appropriate hardware architecture. By introducing parallel stages for computing the SAD cost function, the computational cost can be further reduced.

*Effective mode decision algorithm*: The ME complexity can be further reduced by using effective mode decision algorithms, which focus on eliminating some of the unnecessary modes of the current block. Hence for each selected mode, the proposed fast ME algorithm can be used and thus the overall performance of the video encoder can be increased. In the hardware architecture, the present work can be extended in order to incorporate mode decision and produce MVs for the blocks that are only necessary.

*Fractional ME architecture*: The ME is also performed at sub-pixel level. After performing the integer ME in hardware, the MVs can be used to get the half-pixel and quarter-pixel accurate MVs. A future line of work may focus on the implementation of the fractional ME hardware architecture.

*Motion Compensation unit*: The ME generates the MVs for each block in every frame. The motion compensation unit utilizes these MVs and generates motion compensated frames. These motion compensated frames are subtracted from the current frames to get the residual frames. To verify the effectiveness of the ME architecture, the motion compensation unit hardware architecture can be designed and integrated with ME architecture.

**7. References**

[1] Casey, J.: An Investigation of Block Searching Algorithms for Video Frame Codecs. Master's Dissertation. Dublin, Dublin Institute of Technology, 2008

[2] G. J. Sullivan, J. Ohm, W. J. Han, and T. Wiegand, “Overview of the high efficiency video coding (HEVC) standard,” Circuits and Systems for Video Technology, IEEE Transactions on, vol. 22, no. 12, pp. 1649–1668, 2012

[3] N.Purnachand, L.N. Alves,and A. Navarro,“Improvements to TZSearch motion estimation algorithm For multiview video coding.,” International Conference on Systems, Signals and Image Processing (IWSSIP), Austria

[4] A.C.Bovik ,The essential guide to video processing ,Academic Press -Elsevier,2009

[5] A. Saha, M. Jayanta and S. Shamik, "New pixel-decimation patterns for block matching in motion estimation," Image Communication Signal Processing, vol. 23, no. 10, pp. 725-738

[6] W. Chung-Neng, Taiwan, Y. Shin-Wei, L. Chi-Min and C. Tihao, "A hierarchical decimation lattice based on N-queen with an application for motion estimation," IEEE Signal Processing Letters, vol. 10, no. 8, pp. 228 - 231, Aug. 2003 [7] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," ITU-T VCEG Q6/16 Report, Austin, TX, USA, April 2001

[8] Z. Shan and M. Kai-Kuang, "A new diamond search algorithm for fast block matching motion estimation," IEEE Transactions on Image Processing, vol. 9, no. 2, pp. 287 - 290, Feb. 2000.

[9]Z. Ce, L. Xiao and L.-P. Chau, "Hexagon-based search pattern for fast block motion estimation," IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, no. 5, pp. 349 - 355, May 2002

[10]T. Koga, K. Iinuma, A. Hirano, Y. Iijima and T. Ishiguro (Nov-Dec 1981) Motion-Compensated Interframe Coding for Video Conferencing, Proceedings National Telecommunications Conference, New Orleans, '81 (IEEE), p G.5.3.1 - G.5.3.5.

[11] M. Ghanbari. (July 1990) The Cross-Search Algorithm for Motion Estimation, IEEE Transactions on Communications, Volume 38, No. 7, pp. 950–953.

**Annexure**

CADEC 2019 PUBLISHED PAPER

“Review on Fast Motion Estimation Algorithms for HEVC in Video Communication” recently published in “*Proceeding of the First Annual Conference on Computer-aided Developments in Electronics and Communication (CADEC-2019), 2-3 March 2019”*

