ANALYSIS OF BLOCK MATCHING MOTION ESTIMATION ALGORITHMS

P. Muralidhar C.B.Rama Rao
pmurali@nitw.ac.in, cbrr@nitw.ac.in
Departments of Electronics and Communication Engineering
National Institute of Technology, Warangal

Abstract— **Motion estimation is one of the most important tasks in the current standards of video compression. In this paper seven different block matching motion estimation algorithms are analysed for fast motion estimation in video coding. The simulation results of the algorithms are compared based on PSNR, average time taken per frame, total number of search points per macro block and time taken per macro block.**

Index Terms— **Block matching, SAD, motion estimation, H.264/AVC**

I. INTRODUCTION

Video compression [1] is a technology that can efficiently reduce the bandwidth required to transmit video signals. Most video compression is lossy compression; in the other words it needs to retain original video quality under a few of constraints, such as, storage constraints, encoding time, hardware and software computation, and power consumption. Video compression takes advantage of data redundancy between successive video images to reduce the storage requirement by applying computational resources. In order to design a video compression system, it always encounters with trade off between video qualities, speed, and storage sizes.

H.264/MPEG-4 AVC [2] [3] inherited the characteristic of the block-based matching video coding standards and further presented lots of new features that can efficiently improve coding performance. For video coding systems, motion estimation (ME) can remove most of temporal redundancy, so a high compression ratio can be achieved. Among various ME algorithms, a full-search block matching algorithm (FSBMA) is usually adopted because of its high PSNR. Although FSBMA provides the best quality among various ME algorithms, it consumes the largest computation power.

For real time systems, we employ fast block matching motion estimation algorithms, as they have reduced computation. For these fast block matching algorithms, we try to attain PSNR closer to Full Search. The block matching algorithms that have been implemented are Full Search, Three Step Search (TSS), New Three Step Search (NTSS), Simple and Efficient TSS (SES), Four Step Search (4SS), Diamond Search (DS) and proposed algorithm on C. then a comparison is made among different algorithms.

In this paper we studied and analysed the existing motion estimation algorithms for fast block matching motion estimation in H.264 AVC.

Section II explains different block matching algorithm in general. In Section III we propose and describe our algorithm in detail. In Section IV we compare different algorithms for the Foreman sequence based on PSNR, number of search points and the time taken along with simulation results followed by summary and references.

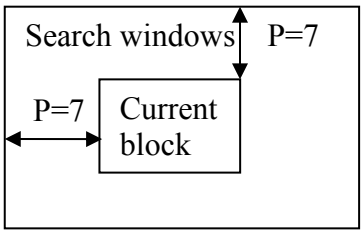

Fig. 1. Macro block of 16 pixels and a search parameter of 7

II. BLOCK MATCHING ALGORITHMS

Block matching can only be implemented for the frame having a single object moving within that frame to form corresponding objects in the subsequent frame. To implement block matching current frame is to divided into a matrix of 'macro blocks' that are then compared with corresponding block in the previous frame to create a vector that stipulates the movement of a macro block from one location to another in the previous frame. These motion vectors along with previous frame data are used to reconstruct the frame at the decoder. The search area is defined around a macro block for a search parameter of **p** (which is usually taken to be 7 pixels on all four sides of the corresponding macro block in the previous frame but can vary as per the movement in the frames) shown in Fig. 1. The larger the motions, the larger are search parameter **p**.

For each MB in the current frame (current MB), one reference block that is the most similar to current MB is sought in the searching range of size [-P, P] in the reference frame. There are many cost function to compare candidate blocks like Mean Square Error (MSE), Mean Absolute Difference (MAD), Sum of Absolute Difference (SAD). SAD is defined as

$$SAD(x,y) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |a(i,j) - b(i+x, j+y)| \qquad 1$$

Where a (i, j) is the pixel data in the current block of a size N x N, $b(i+x, j+y)$ is the pixel data within the search area of previous frame and (x, y) represents the candidate displacement vector. (x ,y) ranges from –p to p. After all searching candidates are examined, the candidate block that has the smallest SAD is selected as the motion vector of the current MB.

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

$$PSNR = 10 \log_{10}\left[\frac{(peak\ to\ peak\ value\ of\ original\ data)^2}{MSE}\right] \quad 2$$

A. Exhaustive Search (ES)

This algorithm also called Full search, calculates cost function at each possible location in the search window. ES is the most computationally expensive block matching algorithm of all. It finds the best possible match and gives the highest PSNR amongst any block matching algorithm. The obvious disadvantage to ES is that the larger the search window gets the more computations it requires.

B. Three Step Search (TSS)

It starts with the search location at center and sets the 'step size' S=4, for a usual search parameter value of 7. It then searches at eight locations +/-S pixels around location (0, 0). One point with the lowest cost is selected as new center and search is performed around this point with S=S/2 and repeats similar search for two more iterations until S=1. At this point the location with the least cost function is found. TSS procedure is shown in fig. 2. It gives a flat reduction in computation by a factor of 9. It has the disadvantage of getting stuck in local minima.

C. New Three Step Search (NTSS)

NTSS [4] is a center biased searching scheme and has provisions for half way stop to reduce computational cost. NTSS process is illustrated graphically in Fig. 3. In the first step 16 points are checked in addition to the search origin. 8 locations are a distance of S = 4 away and the other 8 are at S = 1 away from the search origin. If the lowest cost is at the origin then the search is stopped right here and the motion vector is set as (0, 0). If the lowest weight is at any one of the 8 locations at S = 1, then the origin of the search is changed to that point and check for weights adjacent to it. Depending on which point it is we might end up checking 5 or 3 points.

If the minimum cost from the first step is found for S=4 location then normal TSS is followed. Hence in worst case this has to check 33 points and 17 points is the minimum point checked for each macro block.

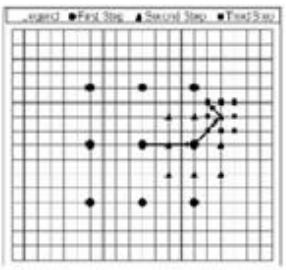

Fig. 2. Three step search

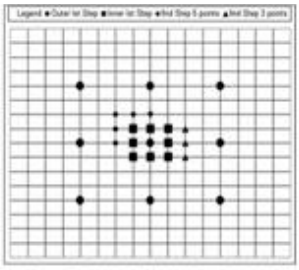

Fig. 3. New three step search

D. Simple and Efficient Search (SES)

SES [5] exploits the assumption of unimodal error surface. The main idea behind the algorithm is that for a unimodal surface there cannot be two minimums in opposite directions and hence the 8 point fixed pattern search of TSS can be changed to incorporate this and save on computations.

The algorithm still has three steps, each step has further two phases. The search area is divided into four quadrants and the algorithm checks three locations A, B, C as shown in Fig. 4. A is at the origin and B and C are S = 4 locations away from A in orthogonal directions. Depending on certain weight distribution amongst the three, the second phase selects few additional points Fig. 4. The rules for determining a search quadrant for seconds phase are as:

If SAD (A) ≥ SAD (B) and SAD (A) ≥ SAD (C), select (b);
If SAD (A) ≥ SAD (B) and SAD (A) ≤ SAD (C), select (c);
If SAD (A) < SAD (B) and SAD (A) < SAD (C), select (d);
If SAD (A) < SAD (B) and SAD (A) ≥ SAD (C), select (e);

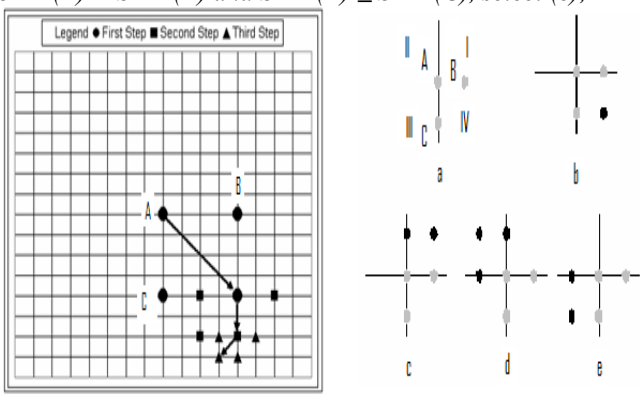

Fig . 4. SES

Although it saves a lot of computational time, but the error surface is not strictly unimodular and hence gives poor PSNR value.

E. Four Step Search (4SS)

FSS [4] employs center biased searching and has a halfway stop provision. 4SS sets a fixed pattern size of S=2 for the first step. Thus it looks at 9 locations in a 5x5 window. If the least weight is found at the center of search window the search jumps to fourth step. If the least weight is at one of the eight locations except the center, then we make it the search origin and move to the second step. Depending on where the least weight location was, we might end up checking weights at 3 or 5 locations. The patterns are shown in Fig. **5**. Once again if the least weight location is at the center of the 5x5 search window we jump to fourth step or else we move on to third step. The third is exactly the same as the second step. In the fourth step the window size is dropped to 3x3, i.e. S = 1. The location with the least weight is the best matching macro block and the motion vector is set to point of that location. This search algorithm has the best case of 17 checking points and worst case of 27 checking points.

F. Diamond Search (DS)

DS [7] uses two different types of fixed patterns, one is Large Diamond Search Pattern (LDSP) and the other is Small Diamond Search Pattern (SDSP). DS procedure are illustrated in Fig. **6.** Just like in FSS, the first step uses LDSP and if the least weight is at the center location we jump to fourth step. The consequent steps, except the last step, are also similar and use LDSP, but the number of points where cost function is checked are either 3 or 5 and are illustrated in second and third steps of procedure. The last step uses SDSP around the new search origin and the location with the least weight is the best match.

DS has no limit on the number of steps that the algorithm can take but the search should remain inside the defines search range. The end result should see a PSNR close to that of ES while computational expense should be significantly less.

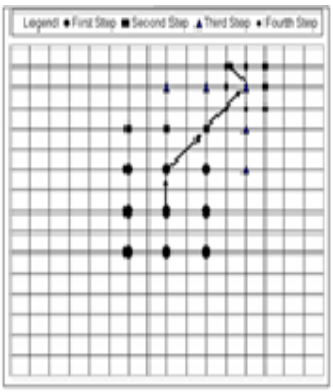

Fig. 5. Four Step search

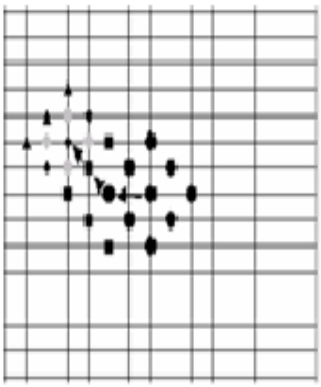

Fig. 6. Diamond search

III. Prediction Based Early Termination DS(PBETDS)

This algorithm makes use of the fact that the general motion in a frame is usually coherent, i.e. if the macro blocks around the current macro block moved in a particular direction then there is a high probability that the current macro block will also have a similar motion vector. There are many prediction strategies of which we use the mean of blocks on left and above. In the fig. 7. for the current macro block E the predicted motion vector is taken as the mean of the motion vectors of macro blocks A and B.

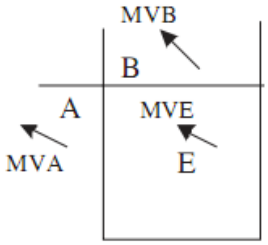

Fig.7 . Prediction vector of macro block from motion vectors of block A and block B

$$\overrightarrow{MV}_{pred(E)} = \text{Mean}(\overrightarrow{MV}_A, \overrightarrow{MV}_B) \quad 3$$

Length of prediction is defined as

$$L_{pred} = \lambda x \sqrt{Xpred^2 + Ypred^2} \quad 4$$

Where λ is defined as weight for optimization of L_{pred}.

Considering the direction of prediction, nine modes are defined as shown in fig. 8.

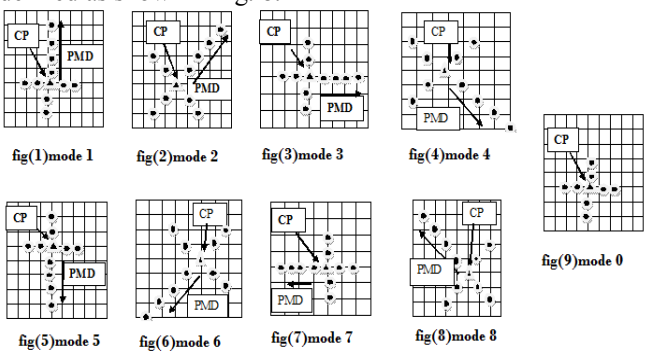

Fig. 8. Different modes of prediction (1) mode 1 (2) Mode 2 (3) Mode 3 (4) Mode 4 (5) Mode 5 (6) Mode 6 (7) Mode 7 (8) Mode 8 (9) Mode 0

*CP-Center Point & PMD-Predicted Motion Direction

Different modes are selected based on following equations.

If $X_{pred}=0$ and $Y_{pred}=0$; select mode 0; 5

If $|X_{pred}|<Y_{pred}/4$; select mode 1; 6

If $X_{pred}/4<Y_{pred}<4xX_{pred}$; select mode 2; 7

If $|Y_{pred}|<X_{pred}/4$; select mode 3; 8

If $-4xX_{pred}<Y_{pred}<-X_{pred}/4$; select mode 4; 9

If $|X_{pred}|<-Y_{pred}/4$; select mode 5; 10

If $4xX_{pred}<Y_{pred}<X_{pred}/4$; select mode 6; 11

If $|Y_{pred}|<-X_{pred}/4$; select mode 7; 12

If $-X_{pred}/4<Y_{pred}<-4xX_{pred}$; select mode 8; 13

For mode 0 shown in Fig. 8.(9) simple rood pattern is searched on 5 search locations.

For mode 1,3,5,7 shown in Fig. 8.(1), Fig. 8.(3), Fig. 8.(5) and Fig. 8.(7) respectively rood pattern is searched, in addition points are searched in the direction of prediction, where starting from the end of the rood, we search 2 more points than the prediction length.

Similarly for mode For mode 2,4,6,8 shown in Fig. 8.(2), Fig. 8.(4), Fig. 8.(6) and Fig. 8.(8) respectively cross pattern is searched, in addition points are searched in the direction of prediction, where starting from the end of the cross, we search 2 more points than the prediction length.

Minimum distortion block (MDB) among these search points is selected as center for the large diamond search pattern (LDSP). Now diamond search with the LDSP is performed repeatedly until MDB comes at the center of LDSP.

In last step 9 point search is performe where 9 points are searched around the MBD resulted from diamond search.

The MBD among these 9 candidate blocks gives the final motion vector.

- Early Elimination based SAD
  Instead of normal SAD calculation we use early elimination SAD which checks if the SAD of the present candidate block has exceeded the previously calculated SAD_{min}, when the number of pixels accessed are ½ , ¾ and 7/8 of the total number of pixel in the macro block. If the SAD comes out to be greater than SAD_{min} then the SAD calculation is stopped midway. This prevents a considerable amount of time as there is good probability that the present SAD would exceed SAD_{min} in the other half of the SAD calculation.

IV. SIMULATION RESULTS

In our simulation experiments, the block size is fixed at 16x16. To make a consistent comparison, block matching is conducted within a15x15 search window (i.e., 7 pels displacement in horizontal and vertical directions). Elimination based Sum of absolute distance rather than normal SAD is used to eliminate impossible candidates. The algorithms were implemented for Foreman sequence.

During the implementation phase all of the above 7 algorithm have been implemented on C (Visual Studio Platform) with the use of OpenCV functions. The implementation helps us to get more clear picture of the hardware aspect which will be a matter of consideration at the time of architecture designing.

As is shown by Fig. 10., ES, TSS, NTSS, SES, 4SS, DS and new algorithm have been plotted against the frame number (frame no. is the distance of the frame under consideration from the reference frame used for reconstruction).

It was found that NTSS and SES perform badly whereas our proposed algorithm gave PSNR close to the ES and better than DS as shown in TABLE I. While the ES takes on an average around 184 searches per MB whereas our proposed algorithm takes 30 searches per MB which is little greater than DS which is 28 searches per MB. However due

to better PSNR than DS thus our algorithm proves to be better than DS.

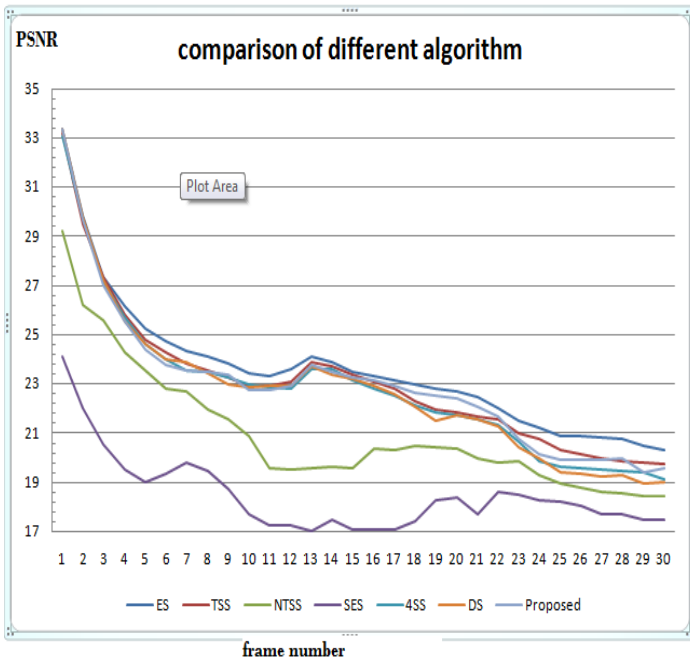

Fig. 10. PSNR vs Current Frame no. for foreman sequence

3SS has a higher probability of getting trapped in local minima whereas the prediction technique used in our proposed methodology gives the better approach local minima. The time taken by our algorithm is also lesser than ES and DS even if the search points are larger than DS as in TABLE I. This is because we have used early elimination based SAD which prevents lots of computational time.

The weight of optimisation λ has been found to give best results for values in the range of 1.45 to 1.65, thus we use it as a parameter to adjust video quality externally.

TABLE I
NUMBER OF SEARCH POINTS PER MACRO BLOCK, AVERAGE PSNR, AVERAGE TIME TAKEN FOR MOTION ESTIMATION PER FRAME FOR DIFFERENT ALGORITHMS ON FOREMAN SEQUENCE

S. No.	Name of Algorithm	Avg no of SP per MB	Avg PSNR	Avg time taken per frame (msec)
1.	ES	184	23.576	99.567
3.	NTSS	24	20.984	18.9
5.	4SS	20	22.836	16.233
6.	DS	28	22.792	22.233
7.	PETDS	30	23.018	20.167

*these time calculation have been done on intel core-2-duo processor (2 GHz).

REFERENCES

[1] Iain E. G. Richardson, Video Codec Design, West Sussex: John Wiley & Sons Ltd., 2002, Ch. 4, 5, & 6. S. M. Metev and V. P. Veiko,
[2] K. R. Raoand J. J. Hwang, techniques and Standards for Image, Video and Audio Coding. Englewood Cliffs, NJ: Prentice Hall, 1996.
[3] Coding of moving pictures and audio, ISO/IEC JTC1/SC29/WG11 N2932, Oct. 1999.
[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] Jianhua Lu, and Ming L. Liou, "A Simple and Efficent Search Algorithm for Block-Matching Motion Estimation", IEEE Trans. Circuits And Systems For Video Technology, vol 7, no. 2, pp. 429-433, April 1997.
[6] Lai-Man Po, and Wing-Chung Ma, "A Novel Four-Step Search Algorithm for Fast Block Motion Estimation", IEEE Trans. Circuits And Systems For Video Technology, vol 6, no. 3, pp. 313-317, June 1996.
[7] Shan Zhu, and Kai-Kuang Ma, " A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation", IEEE Trans. Image Processing, vol 9, no. 2, pp. 287-290, February 2000.
[8] J.-B. Xu, L.-M. Po, and C.-K. Cheng, "Adaptive motion tracking block matching algorithms for video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 97, pp. 1025–1029, Oct. 1999.