Akshat Agrawal al. International Journal of Recent Research Aspects ISSN: 2349-7688, Vol. 1, Issue 3, December 2014, pp. 137-141

Comparative Analysis of Block Motion Estimation Algorithms

Akshat Agrawal

Assistant Professor Amity University Haryana

***Abstract*— The motion estimation and compensation algorithms form the essential components of all video compression techniques, but due to lot of computations involved real time motion compensation is not so easy. It is a critical yet computationally intensive task for video encoding. There are many computational effective block motion estimation algorithms but with trade-off between the algorithm accuracy and algorithm speed. This paper is a review of the block matching algorithms used for motion estimation in video compression. It implements and compares 6 different types of block matching algorithms that range from the very basic Exhaustive Search to the recent fast adaptive algorithms like Adaptive Rood Pattern Search. The algorithms that are evaluated in this paper are widely accepted by the video compressing community and have been used in implementing various standards [1], ranging from MPEG1 / H.261 to MPEG4 / H.263.**

***Keywords*: Block matching, motion estimation, video compression, MPEG, H.261, H.263**

I. INTRODUCTION

Intra-frame & Inter-frame compression

Video compression is needed for efficient storage of video, for efficient data transfer among various components of a video system, to reduce computational resources used in video processor hence reducing overall cost. There are two types of video compression techniques, lossless compression & lossy compression. In lossless compression, there is no information loss. The image reconstructed exactly the same as the original. Whereas in lossy compression some information loss is tolerable.

Spatial and Temporal redundancy are the two types of redundancies in the video sequences, Redundancy among neighboring pixels in an image is called as spatial redundancy & the coding technique which reduces this type of redundancy in an image are called the intra-frame coding. Whereas, redundancy between adjacent frames in a sequence of image is called as temporal redundancy & the coding technique which reduces the temporal redundancies are called inter-frame coding.

Block Motion Estimation

The sequence of frames in most of video sequences consist of high level of redundancy between consecutive frames because he change between consecutive frames is very minimal .The idea of temporal redundancy reduction is to encode first a reference frame and for the consecutive frames encode only the difference between the reference frame and current frame This difference is seen as the error or residual. Motion estimation is defined as searching the best motion vector, which is the placement of the coordinate of the best similar block in previous frame for the block in current frame. Block-based matching algorithms find the optimal motion vectors which minimize the difference between reference block and candidate blocks. The idea behind block matching is to divide the current frame into a matrix of macro blocks that are then compared with corresponding block and its adjacent neighbors 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. This movement calculated for all the macro blocks comprising a frame, constitutes the motion estimated in the current frame. The search area for a good macro block match is constrained up to p pixels on all fours sides of the corresponding macro block in previous frame [2][3]. This p is called as the search parameter.

Larger motions require a larger p and the larger the search parameter the more computationally expensive the process of motion estimation becomes. Usually the macro block is taken as a square of side 16 pixels, and the search parameter p is 7 pixels. The matching of one macro block with another is based on the output of a cost function. The macro block that results in the least cost is the one that matches the closest to current block. There are various cost functions [3], of which the most popular and less computationally expensive is Mean Absolute Difference (MAD) given by equation (i). Another is Mean Squared Error (MSE) given by equation (ii).

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

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

where N is the side of the macro bock, Cij and Rij are the pixels being compared in current macro block and reference macro block, respectively. Peak-Signal-to-Noise-Ratio (PSNR) given by equation (iii) characterizes the motion compensated image that is created by using motion vectors and macro clocks from the reference frame.

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

II. MOTION ESTIMATION ALGORITHMS

A Exhaustive Search (ES)

This algorithm, also known as Full Search, is the most computationally expensive block matching algorithm of all. This algorithm calculates the cost function at each possible location in the search window. As a result of which it finds

the best possible match and gives the highest PSNR amongst any block matching algorithm. Fast block matching algorithms try to achieve the same PSNR doing as little computation as possible. The obvious disadvantage to ES is that the larger the search window gets the more computations it requires.

III. THREE STEP SEARCH (TSS)

The general idea of TSS[4] is represented in Figure 1. It starts with the search location at the 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). From these nine locations searched so far it picks the one giving least cost and makes it the new search origin. It then sets the new step size S = S/2, and repeats similar search for two more iterations until S = 1. At that point it finds the location with the least cost function and the macro block at that location is the best match. The calculated motion vector is then saved for transmission. It gives a flat reduction in computation by a factor of 9. So that for p = 7, ES will compute cost for 225 macro blocks whereas TSS computes cost for 25 macro blocks. The idea behind TSS is that the error surface due to motion in every macro block is unimodal. A unimodal surface is a bowl shaped surface such that the weights generated by the cost function increase monotonically from the global minimum.

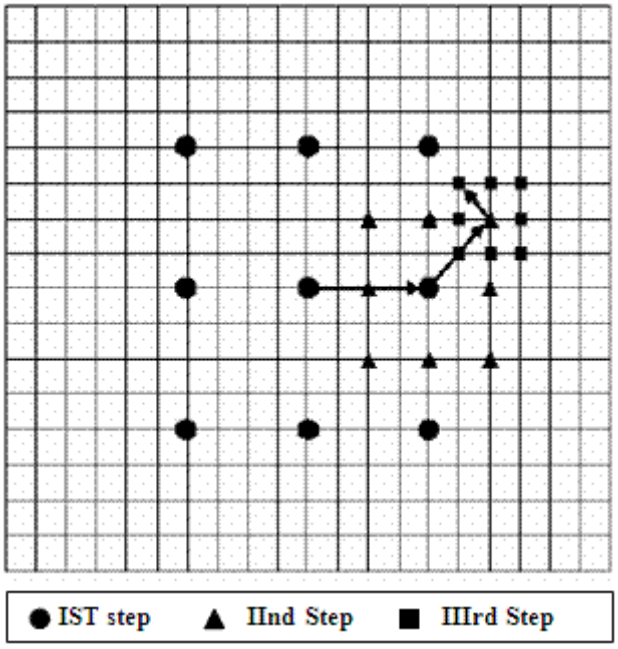

Figure 1. Three Step Search the motion vector is (5, -3).

Simple and Efficient Search (SES)

SES [5] is another extension to TSS and exploits the assumption of unimodel 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 like TSS, but the innovation is that each step has further two phases. The search area is divided into four quadrants and the algorithm checks three locations A, B and C as shown in Figure 2. 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 2). The rules for determining a search quadrant for seconds phase are as follows:

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

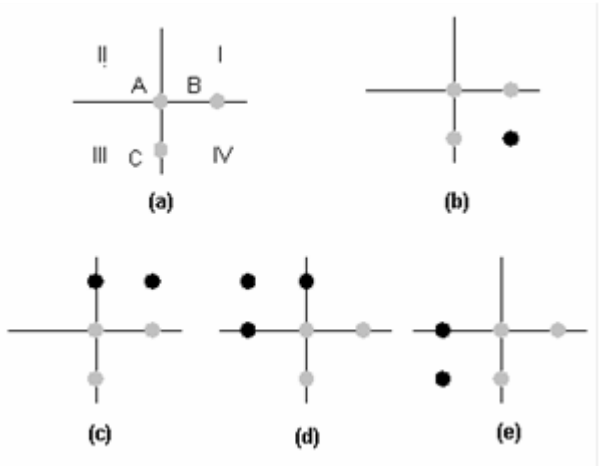

Figure. 2. Search patterns corresponding to each selected quadrant: (a) Shows all quadrants (b) quadrant I is selected (c) quadrant II is selected (d) quadrant III is selected (e) quadrant IV is selected

Once we have selected the points to check for in second phase, we find the location with the lowest weight and set it as the origin. We then change the step size similar to TSS and repeat the above SES procedure again until we reach S = 1. The location with the lowest weight is then noted down in terms of motion vectors and transmitted. An example process is illustrated in Fig 3. Although this algorithm saves a lot on computation as compared to TSS, it was not widely accepted for two reasons. Firstly, in reality the error surfaces are not strictly unimodal and hence the PSNR achieved is poor compared to TSS. Secondly, there was another algorithm, Four Step Search, that had been published a year before that presented low computational cost compared to TSS and gave significantly better PSNR.

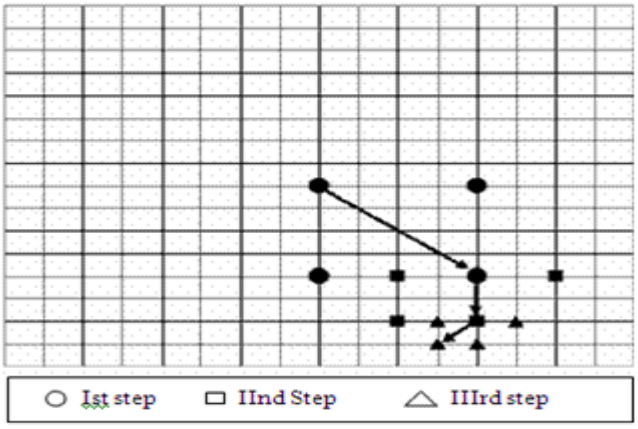

Fig. 3. The SES procedure. The motion vector is (3, 7) in this

example.

Four Step Search(4SS)

4SS [7] also employs center biased searching and has a halfway stop provision. 4SS sets a fixed pattern size of S = 2 for the first step, no matter what the search parameter p value is. 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. The search window is still maintained as 5x5 pixels wide. Depending on where the least weight location was, we might end up checking weights at 3 locations or 5 locations. The patterns are shown in Fig 3. 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 o that location. A sample procedure is shown in Fig 4. This search algorithm has the best case of 17 checking points and worst case of 27 checking points.

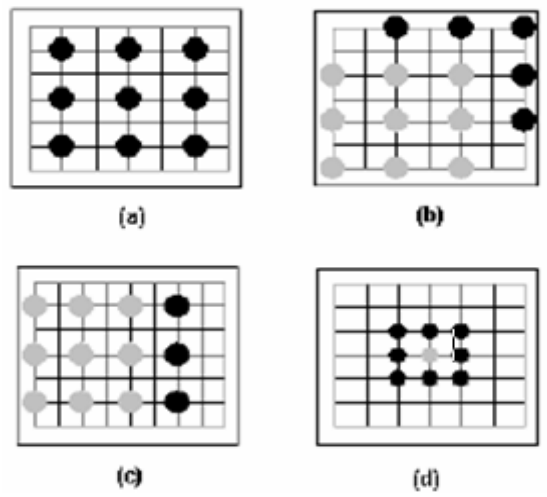

Fig. 4. Search patterns of the FSS. (a) First step (b) Second/Third step(c)Second/Third Step (d) Fourth Step

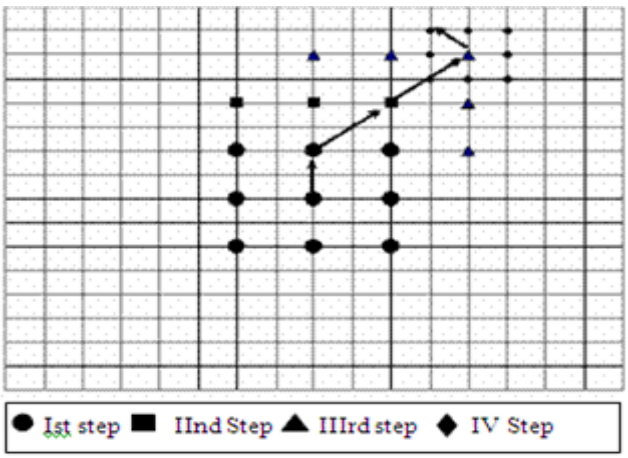

Fig 5. Four Step Search procedure. The motion vector is (3,7).

Diamond Search (DS)

DS [8] algorithm is exactly the same as 4SS, but the search point pattern is changed from a square to a diamond, and there is no limit on the number of steps that the algorithm can take.DS uses two different types of fixed patterns, one is Large Diamond Search Pattern (LDSP) and the other is Small Diamond Search Pattern (SDSP). These two patterns and the DS procedure are illustrated in Fig. 5. 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 shown in Fig.9. The last step uses SDSP around the new search origin and the location with the least weight is the best match. As the search pattern is neither too small nor too big and the fact that there is no limit to the number of steps, this algorithm can find global minimum very accurately. The end result should see a PSNR close to that of ES while computational expense should be significantly less. Fig. 5. Diamond Search procedure. This figure shows the large diamond search pattern and the small diamond search pattern. It also shows an example path to motion vector (-4, -2) in five search steps four times of LDSP and one time of SDSP.

Adaptive Rood Pattern Search (ARPS)

ARPS [9] 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. This algorithm uses the motion vector of the macro block to its immediate left to predict its own motion vector. An example is shown in Fig. 6. The predicted motion vector points to (3, -2). In addition to checking the location pointed by the predicted motion vector, it also checks at a rood pattern distributed points, as shown in Fig 6, where they are at a step size of S = Max (|X|, |Y|). X and Y are the x-coordinate and y-coordinate of the predicted motion vector. This rood pattern search is always the first step. It directly puts the search in an area where there is a high probability of finding a good matching block. The point that has the least weight becomes the origin for subsequent search steps, and the search pattern is changed to SDSP. The procedure keeps on doing SDSP until least weighted point is found to be at the center of the SDSP. A further small improvement in the algorithm can be to check for Zero Motion Prejudgment [8], using which the search is stopped half way if the least weighted point is already at the center of the rood pattern. The main advantage of this algorithm over DS is if the predicted motion vector is (0, 0), it does not waste computational time in doing LDSP, it rather directly starts using SDSP. Furthermore, if the predicted motion vector is far away from the center, then again ARPS save on computations by directly jumping to that vicinity and using SDSP, whereas DS takes its time doing LDSP. Care has to be taken to not repeat the computations at points that were checked earlier. Care also needs to be taken when the predicted motion vector turns our to match one of the rood pattern location. We have to avoid double computation at that point. For macro blocks in the first column of the frame, rood pattern step size is fixed

at 2 pixels.

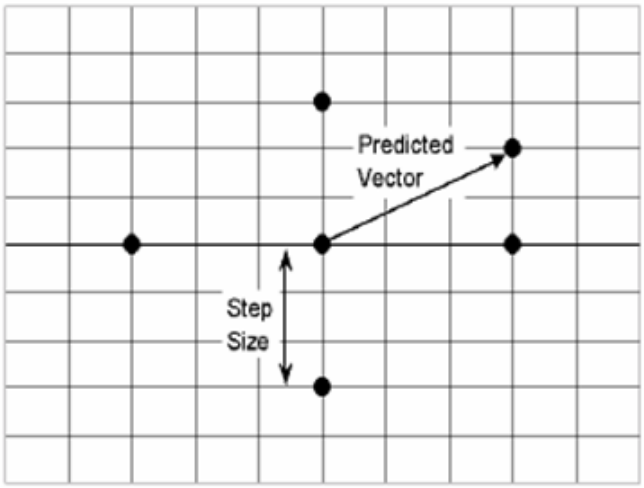

Fig. 6. Adaptive Root Pattern: The predicted motion vector is (3,-2) and the step size S = Max(|3|, |-2|) = 3.

IV. CONCLUSIONS AND SIMULATION RESULTS

During the course of this project all of the above 7 algorithms have been implemented. 'Caltrain' video sequence with a distance of 2 between current frame and reference frame was used to generate the frame-by-frame results of the algorithms. Fig. 7 shows a plot of the average number of searches required per macro block for the Caltrain sequence using the 6 fast block matching algorithms. The PSNR comparison of the compensated images generated using the algorithms is shown in Fig 8. The results are extremely similar to the results of [7] and [8]. As is shown by Fig. 11, 4SS, DS and ARPS come pretty close to the PSNR results of ES. While the ES takes on an average around ~205 searches per macro block, DS and 4SS drop that number by more then an order of magnitude. ARPS further drops by a factor of 2 compared to DS. NTSS and TSS although do not come close in PSNR performance to the results of ES, but even they drop down the number of computations required per macro block by almost an order of magnitude. SES takes up less number of search point computations amongst all but ARPS.

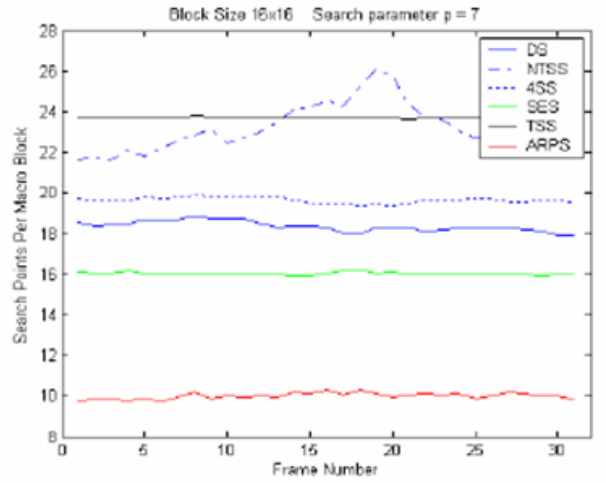

Fig. 7. Search points per macro block while computing the PSNR performance of Fast Block Matching Algorithms
It however also has the worst PSNR performance. Although PSNR performance of 4SS, DS, and ARPS is relatively the same, ARPS takes a factor of 2 less computations and hence is the best of the fast block matching algorithms studied in this paper. The results are similar to that of [8].

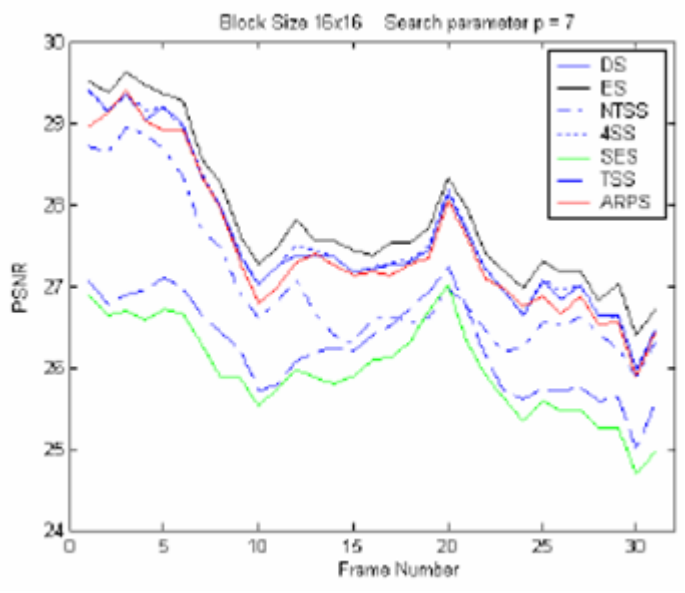

Fig. 8 PSNR performance of Fast Block Matching Algorithms. Caltrain Sequence was used with a frame distance of 2.

V. SUMMARY

Video compression plays an important role in archival of entertainment based video (CD/DVD) as well as real-time reconnaissance / video conferencing applications. While ISO MPEG sets the standard for the former types of application, ITU sets the standards for latter low bit rate applications. In the entire motion based video compression process motion estimation is the most computationally expensive and time-consuming process. The research in the past decade has focused on reducing both of these side effects of motion estimation. Block matching techniques are the most popular and efficient of the various motion estimation techniques. This paper first describes the motion compensation based video compression in brief. It then illustrates and simulates 7 of the most popular block matching algorithms, with their comparative study at the end. This paper first describes the motion compensation based video compression in brief. It then illustrates and simulates 6 of the most popular block matching algorithms, with their comparative study at the end.

VI. FUTURE WORK:

Since these algorithms are based on assumption that the Block Distortion measure increases as the checking point moves away from the global minimum point. However this assumption does not always hold in the real world video sequence. These algorithms may get trapped into local minimum points and thus produce large matching error compared with the ES algorithm. The algorithms can be

enhanced and made much less computationally intensive by the use of techniques like Normalized partial distortion [11]. These algorithms can decrease the computational complexity while maintaining its MSE performance equal to ES algorithms.

REFRENCES

[1]. R.Schafer and T. Sikora, "Digital Video Coding Standards and Their Role in Video Communications," Proceedinas ofthe LEEE Vol. 83, pp. 907-923, 1995.

[2]. Aroh barjatya, "Block matching algorithms for motion estimation,"Final project paper, DIP, at Utah state university, spring 2004.

[3]. Borko Furht, Joshua Greenberg, Raymond Westwater, Motion Estimation Algorithms For Video Compression. Massachusetts: Kluwer Academic Publishers, 1997. Ch. 2&3.

[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]. Motion Estimation developing package,[Online]. Avaliable http://www.ee.cityu.edu.hk/~impo/publications/

[7]. 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.

[8]. 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.

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

[10]. Chun-Ho Cheung, and Lai-Man Po, "A Novel Small Cross-Diamond Search Algorithm for Fast Video Coding and Video Conferencing Applications", Proc. IEEE ICIP, September 2002.

[11]. Chok-Kwan Cheung; Lai-Man Po; , "Normalized partial distortion search algorithm for block motion estimation," Circuits and Systems for Video Technology, IEEE Transactions on , vol.10, no.3, pp.417-422, Apr 2000 doi: 10.1109/76.836286

[12]. Block matching algorithms, http://www.mathworks.com/matlabcentral/fileexchange/ Conferencing Applications", Proc. IEEE ICIP, September 2002

