International Journal on Emerging Technologies **2**(2): 96-101(2011)

ISSN No. (Print) : 0975-8364
ISSN No. (Online) : 2249-3255

A Review on Comparative Analysis of Block Matching Algorithm

A.J. Tankariya, Jaikarn Singh and Mukesh Tiwari

Department of Electronics and Communication Engineering, SSSIST, Sehore, (MP)

(Received 11 October, 2011, Accepted 14 November, 2011)

ABSTRACT : This paper is a review of the block matching algorithms used for motion estimation in video compression. It implements and compares 7 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, ranging from MPEG1 / H.261 to MPEG4 / H.263. The paper also presents a very brief introduction to the entire flow of video compression.

Keywords: Block matching, motion estimation, video compression, MPEG.

I. INTRODUCTION

Video has huge redundant information which must be exploited to be stored and transmitted efficiently. The common technique to achieve this goal is known as motion estimation. In this technique, the current frame is predicted from a previous frame known as reference frame by using motion vectors. With the increasing demand of multimedia applications, considerable efforts are needed for efficient video compressing and encoding algorithms. Motion estimation has proven to be an effective technique for exploiting the temporal redundancy in video sequences and is therefore an essential part of MPEG and H.263 compression standards. Since motion estimation is the most computationally intensive portion of video encoding, efficient fast motion estimation algorithms are highly desired for video compressors subject to diverse requirement on bit rate, video sequence characteristics and delay. Knowledge of the motion is not available from a video data and must be deduced using computationally intensive algorithms. For efficient handling of motions with variety of contents, the need for adaptive motion estimation methods is inevitable. Digital video is widely used and plays an import role in modern society, such as digital entertainments, video conference, and video surveillance. However, the huge size of media data is the major obstacle for efficient video storage and communication. n order to compress the video signals, many international standards are developed, namely the ISO/IEC MPEG-x series [1], and the ITU-T H.26x series.

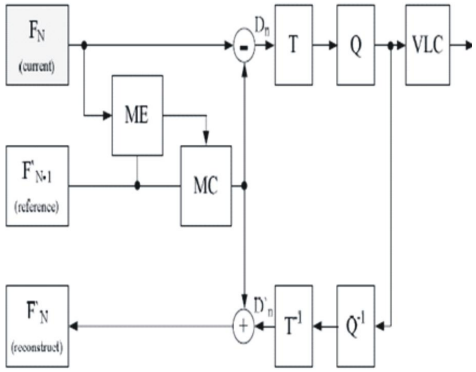

Fig. 1. Video compression process flow.

The basic flow of the entire compression-decompression process is depicted in Fig. 1. This includes the motion estimation (ME) and motion compensation (MC). The hybrid video encoder can be divided into the forward and backward paths. In the forward path, estimates the motion in the current frame with respect to a previous frame. A motion compensated image for the current frame is then created that is built of blocks of image from the previous frame. Here for each macro block (MB), motion estimation (ME) is firstly conducted to find the best matching position. Then the motion compensation (MC) is scheduled to

generate the matching reference block. Thirdly, the original MB is subtracted by the reference block to get the residuals Dn. Fourthly, the obtained residuals are transformed (T) and quantized (Q), and the calculated coefficients are encoded by the variable length code (VLC) and ready for transmission. In the backward path, the generated coefficients are firstly inverse quantized (Q-1) and inverse transformed (T-1) to get the reconstructed residuals D`n, which is then added with the reference block obtained by MC to generate the reconstruct MB, which will be used for future ME process.

The whole idea behind motion estimation based video compression is to save on bits by sending JPEG encoded difference images which inherently have less energy and can be highly compressed as compared to sending a full frame that is JPEG encoded. Motion JPEG, Where all frames are JPEG encoded, achieves anything between 10 : 1 to 15 : 1 compression ratio, where as MPEG can achieve a compression ratio of 30:1 and is also useful at 100 : 1 ratio [1] [2] [3]. It should be noted that the first frame is always sent full, and so are some other frames that might occur at some regular interval (like every 6th frame). The standards do not specify this and it might change with every video being sent based on the dynamics of the video.

The most computationally expensive and resource hungry operation in the entire compression process is motion estimation. Hence, this field has seen the highest activity and research interest in the past two decades. This paper implements and evaluates the fundamental block matching algorithms from the mid-1980s up to the recent fast block matching algorithms. It also presents a literature review of few papers from the last 3 years.

II. MOTION ESTIMATION ALGORITHMS

This section comprehensively presents a block-based algorithm for motion estimation. The below supposition behind motion estimation is that the patterns corresponding to objects and background in a frame of video sequence move within the frame to form corresponding objects on the subsequent frame.

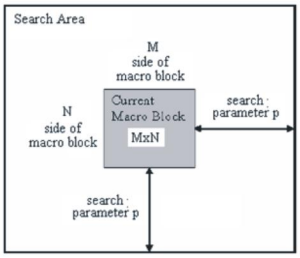

Fig. 2. Block matching a macro block of side 16 pixels and a search parameter *p* of size 7 pixels.

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. 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 idea is represented in Fig 2.

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, of which the most popular and less computationally expensive is Mean Absolute Difference (MAD) given by equation (*i*). Another cost function is Mean Squared Error (MSE) given by equation (*ii*).

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

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

where N is the side of the macro bock, C_{ij} and R_{ij} are the pixels being compared in current macro block and reference macro block, respectively.

Other than above the *SAD* criterion is often used as the criterion for choosing the best-matching block in the reference frame due to its simplicity and good performance given as below.

$$SAD(k,l;u,v) = \sum_{k=0}^{M-1}\sum_{i=0}^{N-1} | B_{ij}(k,l) - B_{i-u,\,j-v}(k,l) |$$

where, $B_{i,j}$ (k, l), represents the $(k, l)^{th}$ pixel of a 16×16 macro block from the current picture at the spatial location (i, j). And next $B_{i-u,\,j-v}(k, l)$ represents the $(k, l)^{th}$ pixel of a candidate macro block from a reference picture at the spatial location (i, j) displaced by the vector (u, v).

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.

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

A. Full Search (FS)

This algorithm, also known as Exhaustive Search, it is one of the classical algorithms for block-based motion compensation. It 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. Its plus point is high accuracy for block motion estimation and small number of index MVs for represent the

reconstruction image (only one MV represent for the whole block). Its negetive point is high computational time (slow) and a lot of image detail may be lost because it represent by block.

B. Three Step Search (TSS)

This is one of the earliest attempts at fast block matching algorithms and dates back to mid 1980s. It uses a three step search procedure to determine MV estimation. The algorithm is known as TSS algorithm and it is used for displacement computation up to 7 pels/frame.

The search procedure for this algorithm is described as follows: where in the first step, 9 checking points centered at the origin of the search window are first searched with a step size of. The check point with the minimum cost function value is selected as the centre point for the second step. In the second step, Another 8 checking points surrounding the new centre with half the previous step size are checked. Again, the check point with the minimum cost function value is selected as the centre point for the third step. In this last third step of the procedure, the step size is again halved. 8 more checking points surrounding the new centre is checked. The MV is given by the position of the point that gives the minimum cost function value in this stage.

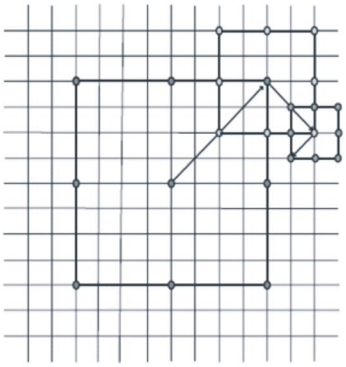

Fig. 3. Three Step Search procedure.

C. New Three Step Search (*NTSS*)

Rather than using the uniform distribution as being used in TSS algorithm, which becomes inefficient for small motion estimation, NTSS algorithm emphasis on the use of centre-biased MV distribution, which is one of real world image sequence's characteristics. The search procedure for NTSS [4] differs from TSS by firstly, employing a centre-biased checking point pattern in its first step and secondly, incorporating a halfway-stop technique for stationary or quasi-stationary blocks. It was one of the first widely accepted fast algorithms and frequently used for implementing earlier standards like MPEG 1 and H.261.

The TSS uses a uniformly allocated checking pattern for motion detection and is prone to missing small motions. The NTSS process is illustrated graphically in Fig 4. In the first step 16 points are checked in addition to the search origin for lowest weight using a cost function. Of these additional search locations, 8 are a distance of $S = 4$ away (similar to TSS) 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 we change the origin of the search to that point and check for weights adjacent to it. Depending on which point it is we might end up checking 5 points or 3 points. The location that gives the lowest weight is the closest match and motion vector is set to that location. On the other hand if the lowest weight after the first step was one of the 8 locations at $S = 4$, then we follow the normal TSS procedure.

Hence although this process might need a minimum of 17 points to check every macro block, it also has the worst-case scenario of 33 locations to check. Its plus point is less computational time on block position than FS and static pattern of searching point (not complexity). Its negetive point is, Allocation of the check point at the first stage leaves several gaps which becomes inefficient small motion estimation. It can be reduced by using smaller block size but may increase searching time and space for motion index.

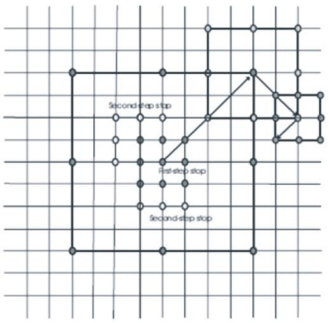

Fig. 4. New Three Step Search block matching.

D. Simple and Efficient Search (*SES*)

SES [5] is another extension to TSS and 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 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 Fig. 5. *A* is at the origin and *B* and *C* are $S = 4$ locations away from *A* in orthogonal directions.

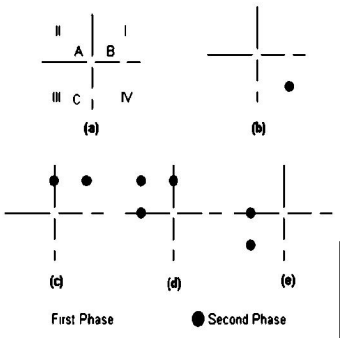

Fig. 5. Search patterns corresponding to each selected quadrant: (*a*) Shows all quadrants (*b*) quadrant I is selected (*c*) quadrant II is selected (*d*) quadrant III is selected (*e*) quadrant IV is selected.

Depending on certain weight distribution amongst the three the second phase selects few additional points (Fig

5). The rules for determining a search quadrant for seconds phase are as follows:

Select (*b*): If MAD(*A*) $\geq$ MAD(*B*) and MAD(*A*) $\geq$ MAD(*C*)

Select (*c*): If MAD(*A*) $\geq$ MAD(*B*) and MAD(*A*) $\leq$ MAD(*C*)

Select (*d*): If MAD(*A*) < MAD(*B*) and MAD(*A*) < MAD(*C*)

Select (*e*): If MAD(*A*) < MAD(*B*) and MAD(*A*) $\geq$ MAD(*C*)

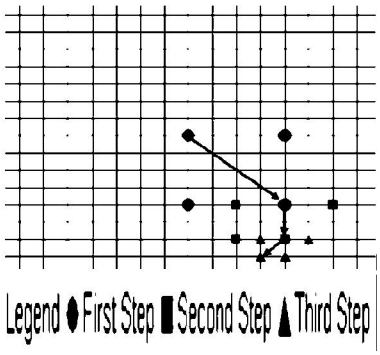

Fig. 6. The SES procedure.

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. 6.

E. Four Step Search (4SS)

Similar to NTSS, 4SS [6] 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 5×5 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 5×5 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. 7. Once again if the least weight location is at the center of the 5×5 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 3×3, *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. 7. This search algorithm has the best case of 17 checking points and worst case of 27 checking points.

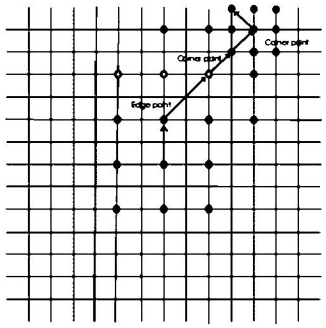

Fig. 7. Four Step Search procedure.

F. Diamond Search (DS)

DS [7] 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. 8.

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.8. 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.

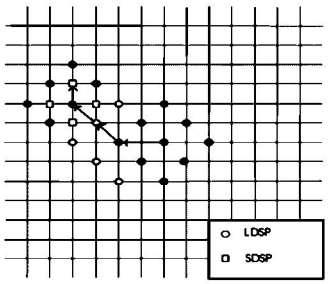

Fig. 8. Diamond Search procedure.

G. Adaptive Rood Pattern Search (ARPS)

ARPS [8] 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. 9. 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. 9, 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.

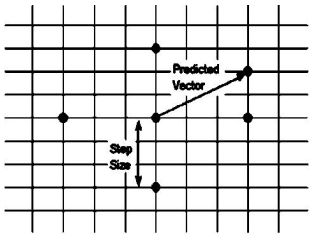

Fig 9. Adaptive Rood Pattern Search.

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.

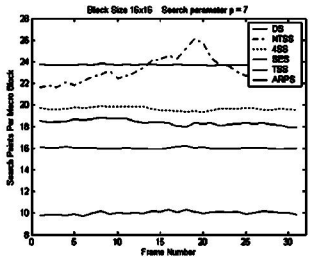

Fig. 10. Search points per macro block while computing the PSNR performance of Fast Block Matching Algorithms.

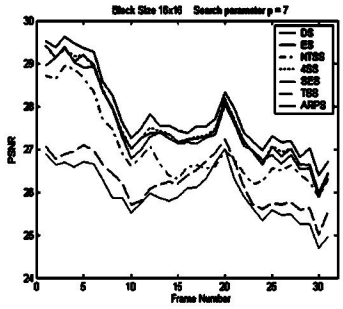

Fig. 11. PSNR performance of Fast Block Matching Algorithms. Caltrain Sequence was used with a frame distance of 2.

III. OTHER RECENT ALGORITHMS

DS proved to be best block matching algorithm of the last century. Every new algorithm in the new millennia is an improvement on the results of DS. Cross Diamond Search (CDS) [9], Small Cross Diamond Search (SCDS) [10] and New Cross Diamond Search (NCDS) [11], all improve on the performance of DS by modifying the starting search pattern from LDSP to cross search pattern (CSP). Amongst themselves these three algorithms differ with respect to the number of points being used out of the CSP. CDS uses all the 9 points whereas SCDS and NCDS use only the inner 5 points to start and then expand their search. However, analogous to the NTSS that eventually ends up doing similar calculations like TSS, these CSP based variants end up going the DS way. Another reason for their improvement over DS is the provision of multiple half-step stops. It should be mentioned that out of the three CSP based variants only NCDS comes closer to the performance of ARPS. The others although an improvement on DS, do not match the performance of ARPS.

IV. SUMMARY

The past two decades have seen the growth of wide acceptance of multimedia. 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. Three more, very recent, block matching algorithms are studied in the end as part of literature review. Of the various algorithms studied or simulated during the course of this final project ARPS turns out to be the best block matching algorithm.

REFERENCES

[1] S. Nomura, K. Yamanaka, O. Katai, H. Kawakami, and T. Shiose, "Anovel adaptive morphological approach for degraded character imagesegmentation," Pattern Recognition, pp. 1961-1975, (2005).

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

[2] M. Ghanbari, Video Coding, An Introduction to Standard Codecs, London: The Institute of Electrical Engineers, Ch. 2, 5, 6, 7 and 8, (1999).

[3] Iain E.G. Richardson, Video Codec Design, West Sussex: John Wiley & Sons Ltd., Ch. 4, 5 and 6, (2002).

[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 Efficient 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] 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).

[9] 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).

[10] Chun-Ho Cheung, and Lai-Man Po, "A Novel Cross-Diamond Search Algorithm for Fast Block Motion Estimation", *IEEE Trans. Circuits And Systems For Video Technology*, vol **12**, no. 12, pp. 1168-1177, December (2002).

[11] C.W. Lam, L.M. Po and C.H. Cheung, "A New Cross-Diamond Search Algorithm for Fast Block Matching Motion Estimation", Proceeding of (2003) *IEEE International Conference on Neural Networks and Signal Processing*, pp. 1262-1265, Dec. (2003), Nanjing, China.