Implementation of fast motion estimation algorithms and comparison with full search method in H.264

A.Ahmadi[†] *, M.M.Azadfar* [††]

[†] *Department of Planning & Engineering , Mobile Communications Company of Iran (MCCI) , Tehran , Iran*
[††]*Multimedia Systems Group , IT Faculty , Iran Telecommunication Research Center (ITRC) , Tehran , Iran*

Summary
In this paper, Three motion estimation algorithms for H.264 standard have been implemented and performance and some features of these three have been tested. These algorithms are full search and two fast search method. Finally some parameters such as bit rate, speed up and PSNR for different value of QP have been compared. Implementation has been performed in Matlab software.

Key words:
ITU-H.264 standard,motion estimation,full search,fast search.

1. Introduction

Motion Estimation (ME) is an important part of any video compression system, since it can achieve significant compression by exploiting the temporal redundancy existing in a video sequence.

Unfortunately it is also the most computationally intensive function of the entire encoding process. In motion estimation the current image is divided into Macro Blocks (MB) [1].

Most of algorithms have been proposed for motion estimation use from BMA_based (Block Matching Algorithms) methods. In this methods, motion estimation is performed for a N×M blocks of current frame, It is done with checking entire N×M blocks from search area situated in the reference frame(s) and calculating the difference between the current block and other reference blocks and finally choosing the block that has the most similarity (minimum SSD) to the early block in current frame. Then, difference of two blocks as residual (motion compensated residual) and the distance of them as motion vector, is coded and transmitted. Search area in the reference frame, generally is a range of [-15…,15]×[-15,…,15] pixels relative to the location of the block to be searched.

$$SSD_{A\ (F,G)} = \sum_{s \in A} \left| F_{(s)} - G_{(s)} \right|^2 \qquad (1)$$

$MVi,j=(x,y)$ *@ min SSD*

In H.264, unlike other standard, M and N is variable and are selected based on complexity value of motion. The seven possible modes of a block in H.264, are 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4. Fitting selection of the block sizes leads to acceptable quality in low bit rates. The problem with method of choosing the best mode of block size is the time, because of the process complexity, we cannot find the best in real time.

The motion estimation process accounts for more than 50% of total encoding time in case of one reference frame, however, as the number of reference frames increases, the relative computational portion of motion compensation increases gradually, and at last it is about 70% in case of four reference pictures.[2]

In fast search algorithms, the motion estimation process instead of full search, follows special pattern that checks less point number, Such as diamond pattern (figure 1) and hexagonal[1] pattern.

smaller motion compensation block sizes can produce better motion compensation results. However, a smaller block size leads to increased complexity (more search operations must be carried out) and an increase in the number of motion vectors that need to be transmitted. Sending each motion vector requires bits to be sent and the extra overhead for vectors may outweigh the benefit of reduced residual energy. An effective compromise is to adapt the block size to the picture characteristics, for example choosing a large block size in flat, homogeneous regions of a frame and choosing a small block size around areas of high detail and complex motion.[3]

The most important criterion for this selection is rate-distortion (RD) and minimization the j_m value that achieves from equation (2).

$$j_m = SSD_m + \lambda_{Mode} \times r_m \qquad (2)$$

Where SSD is sum of square error and λ_{Mode} is equal to:

$$\lambda_{Mode} = 0.85 \times 2^{(QP-12)/3} \qquad (3)$$

Value of QP is between 0 and 51 and r represents the number of bits that needs for motion vector coding.

The informative RD optimization technology provides a bitrate reduction of up to approximately 10% and a PSNR improvement of up to 0.35 dB, while it increases the computational complexity of the H.264 encoder.[4]

$$PSNR_{dB} = 10\log_{10}\frac{(2^n-1)^2}{MSE} \quad (4)$$

2. MB mode determination and motion estimation methods

The seven possible modes of a block, as mentioned earlier, are 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4. Fitting selection of the block sizes leads to acceptable quality in low bit rates. Very different methods may be used in this section that some of them in the following

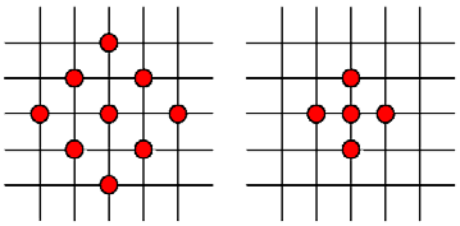
2.1. Full Search method (recommendation of standard)

Generally, in this method, all possible modes is checked. With performing the motion estimation for every blocks and calculating the R-D criterion for all of them, We can decided which block sizes should be used. First of all, motion estimation for macroblock (16×16 block) is performed and j_m is calculated. Then macroblock is divided into two 16×8 and then 8×16 blocks and for each of them, motion estimation and j_m calculation is performed. The sum of calculated j_m of blocks in each mode, is the j_m of that mode. From this four state, a mode that has minimized the j_m is choosed. If the selected mode is 8×8, breaking process of each blocks is continued like the previous. The smallest possible block sizes is 4×4 and afterward the breaking procedure is finished. The most important disadvantage of this algorithm versus it's high precision is, the over time expended to procedure and this,overshadow the feature of realtime coding.

2.2.The first method of fast search and MB mode determination (fast search 1) [2]

In this algorithm, fast motion estimation with modified diamond search for variable block sizes is performed. Motion vector field adaptive search technique (MVFAST) uses a different initial search point and search patterns with selective application of large diamond search (LDS) and small diamond search (SSD) according to the characteristics of motion activity assessed by the similarity of motion vector field among contiguous blocks.

In the LDS case, the LDSP (LDS pattern) shown in Fig.1 is centered at the search center, and all the checking points of LDSP are tested at the first step. If the minimum SAD occurs at the search center of LDSP, then the search pattern is switched from LDSP to SDSP, and the position having the minimum SAD in SDSP is decided as the motion vector. Otherwise, a new center of LDSP is placed at the point that yields the minimum SAD in the previous step, and all points on the new LDSP are tested again. This process is iteratively repeated until the minimum SAD falls on the search center.



(a) LDSP (b) SDSP

Fig. 1 Two types of diamond search pattern

In the proposed scheme, from the similarity of motion vector among hierarchical block size structure is utilized instead of neighboring block's motion field as in MVFAST for the motion field adaptive search. ME is performed from 4×4 to 16×16 block sizes. At first, sixteen basic motion vectors of all 4×4 blocks in one macroblock are found by MVFAST method. The search centers and patterns for larger blocks of 8×4, 4×8, and 8×8 are selected on the basis of the motion vector deviation of the 4×4 blocks constituting the given block.

The motion vector deviation D of a given block of 8×4, 4×8, 8×8 is computed as the average absolute distance between 4×4 block motion vector and their mean vector as:

$$D = \frac{1}{N}\sum_{i=1}^{N}\left|MV_{b(i)} - mean\_vector\right| \quad (5)$$

Where, N is the number of basic motion vectors of *MVb(i)* in the given block, and *MVb(i)* is the motion vector of i^{th} 4×4 block in a given block.

If all basic motion vectors of 4×4 block corresponding to the current block are the same, that is, if *D* equals to zero, the mean vector becomes the motion vector of current block, and ME is skipped. If *D* is smaller than or equal to T_1, only the first step of the small diamond search (SDS) is performed at the center position. That is, only five points as shown in Fig. 1(b) is tested. Otherwise, if *D* is smaller than or equal to T_2, the SDS is performed. If *D* is larger than T_2, LDS is performed. Therefore, as motion vector deviation increases, more complex and larger diamond search starting from the

mean vector is applied. The same process is also performed for 16×8, 8×16, 16×16 blocks with regard to 8×8 block motion vectors. In this experiment, the values 1 and 8 are used respectively for the thresholds T_1 and T_2. [2]

2.3. The second method of fast search and MB mode determination (fast search 2)[5]

This algorithm has been proposed to reduce the number of potential modes and to restrict the set of past coded reference picture for ME. The algorithm will eliminate ME for some block types and reference pictures and will differentiate Skip mode (SKIP mode refers to the 16×16 mode where no motion and residual information is encoded, So no motion search is required and it has the lowest complexity) from other block types and give it the highest priority. The algorithm is based on whether the error surface versus block size is monotonic, that is, whether the current macroblock has the same tendency of using smaller block size (sub-macroblock partition) or larger block size. The error surface is built by initial 3 modes (block sizes): 16×16, 8×8, and 4×4. Here 8×8 means that the entire macroblock is examined using only 8×8 partitions, and 4×4 means that the entire macroblock is examined using only 4×4 partitions. We call that the error surface is monotonic if :

$J_{mode}(16\_16) < J_{mode}(8\_8) < J_{mode}(4\_4)$ or versa.

If the error surface is not monotonic, all other modes need to be tested. If the error surface is monotonic, only modes (block sizes) between the best two modes are tested. For example, if the best two modes are 16×16 and 8×8, which implies that the macroblock tends to use larger block partitions, only 16×8 and 8×16 are further tested; if the best two modes are 8×8 and 4×4, this implies that the macroblock tends to use smaller block partitions (or sub-macroblock partitions), and only 8×4 and 4×8 modes are further tested.

- step1: check SKIP mode. if $J_{mode}(SKIP) < T1$, select SKIP as best mode, stop; otherwise go to step2;
- step2: check 16×16 and 8×8. if ($J_{mode}(SKIP) < J_{mode}(16\_16)$)&&($J_{mode}(SKIP) < J_{mode}(8\_8)$), go to step7; otherwise, go to step3;
- step3: check 4×4; if (($MinJ_{mode} = = J_{mode}(8\_8)$) || ($MaxJ_{mode} = = J_{mode}(8\_8)$)), go to step4; if $MaxJ_{mode} = = J_{mode}(4\_4)$, go to step5; if $MaxJ_{mode} = = J_{mode}(16\_16)$, go to step6;
- step4: check 16×8, 8×16, 8×4, 4×8; go to step7;
- step5: check 16×8, 8×16; go to step7;
- step6: check sub-macroblock partition; go to step7;
- step7: choose the best mode among all tested modes.

In step1, threshold $T_1 = N_{bits}.\lambda_{mode}$, where N_{bits} equals the minimum number of bits required for non SKIP inter modes. In step2, by comparing SKIP with 16×16 and 8×8, we assume that if the RD cost for SKIP is the minimal, then the probability for other modes to have cost less than SKIP will be very small, so no other modes need to be checked. We check the monotonic condition in step3, In step6, an additional decision is performed for each 8×8 partition to decide which type shall be used among the 4 sub-macroblock partitions. Only 8×4 and 4×8 need to be tested.

The search for motion estimation in this algorithm also is performed according to LDS pattern.

3. Implementation results

The experiments were carried out on a 1.6 MHz Intel Pentium machine with 256 MB RAM memory. The environment for comparing the performances is MATLAB software.

The algorithms is applied on two successive frams of the bitstream. The first frame as reference frame is used for coding the next frame.

For example, figure 2 shows two successive frames of silent bitstream and difference of them(residual) has been shown in figure 3.

In the residual frame, PSNR is 28.52 dB and j_m is equal to 9052. but with applying full search algorithm on the frames, motion compensated residual frame shown better results. PSNR and J_m are 40.18 dB and 1950 respectively. This process last about 54 seconds in test states. This frame is shown in figure 4.

After running The fast search(1) algorithm on the frames, following result (fig 5) will be achieved.

In this frame, PSNR is 38.36 dB and average of J_m criterion is 2561, while the Speed_up value is 4.9 respect to the full search method.

In the next state, with applying the fast search(2) algorithm, PSNR, J_m and Speed_up will be 38.35 dB, 4590 and 7.7 respectively. Residual frame has been shown in figure 6. The energy and bit_rate of this frame is larger than previous and these are the cost of better Speed_up. All of these tests is performed for QP=26.

Calculation of speed_up parameter is as following:

$$Speed\_up = \frac{Time_{Full\ search}}{Time_{Fast\ search}} \qquad (6)$$

Fig. 2 two successive frames of silent bitstream

Fig. 3 difference of two frames

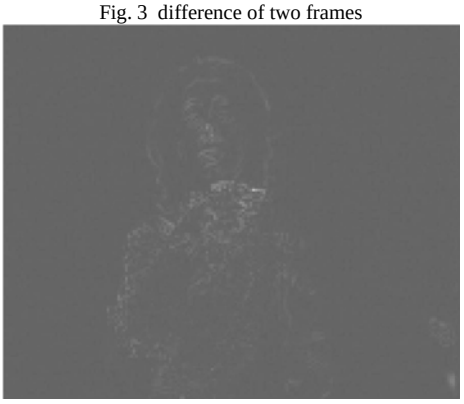


Fig . 4 residual frame after applying full search

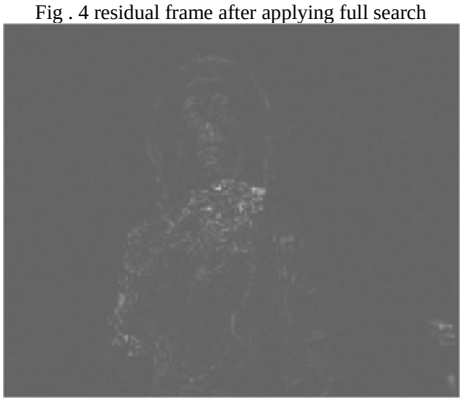


Fig . 5 residual frame after applying fastl search(1)

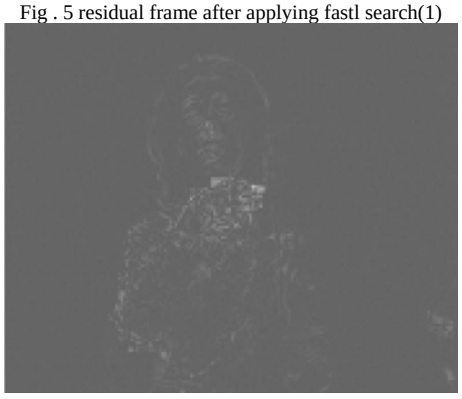


Fig . 6 residual frame after applying fastl search(2)

These successive two frames have been chosen from foreman, coastguard, hall, news and silent bitstreams and their parameters such as bit-rate, speed_up and PSNR have been compared. Both of bit-rate and speed_up parameters is respect to the Full search and hence one value has been assigned to them. The bit-rate calculation is performed based on RD cost value in the picture.

The comparison is presented for three separate values of QP parameter (consist of 5, 26, 50). the results is shown in annex tables(from table1 to table4). To obtain more precision results Table4 for 10 successive frames (F=10)(at *QP=26*) has been achieved.

Table 1: comparison at *QP=5 (F=1)*

QP=5	Full Search			Fast Search (1)			Fast Search (2)		
	Bitrate [%]	PSNR (dB)	SpeedUp [%]	Bitrate [%]	PSNR (dB)	SpeedUp [%]	Bitrate [%]	PSNR (dB)	SpeedUp [%]
foreman	1	35.78	1	1.1	35.37	7.6	2.55	34.9	6.6
coastguard	1	31.76	1	1	31.75	7.2	1.8	31.65	6.9
news	1	38.3	1	1.24	37.1	5.9	3.2	37.00	10.7
hall	1	36.84	1	1	36.7	7	1.7	35.43	8.9
silent	1	40.41	1	1.5	38.41	6.1	2.9	38.43	7.8

Table 2: comparison at *QP=26 (F=1)*

QP=26	Full Search			Fast Search (1)			Fast Search (2)		
	Bitrate [%]	PSNR (dB)	SpeedUp [%]	Bitrate [%]	PSNR (dB)	SpeedUp [%]	Bitrate [%]	PSNR (dB)	SpeedUp [%]
foreman	1	35.47	1	1	35.29	5.5	1.7	35.1	5.2
coastguard	1	31.66	1	1	31.77	5.1	1.5	31.6	6.1
news	1	38.25	1	1.2	37.13	5.1	2.6	36.99	10.8
hall	1	36.7	1	1	36.6	4.2	1.5	35.35	12.5
silent	1	40.18	1	1.3	38.36	4.9	2.4	38.35	7.7

Table 3: comparison at *QP=50 (F=1)*

QP=50	Full Search			Fast Search (1)			Fast Search (2)		
	Bitrate [%]	PSNR (dB)	SpeedUp [%]	Bitrate [%]	PSNR (dB)	SpeedUp [%]	Bitrate [%]	PSNR (dB)	SpeedUp [%]
foreman	1	32.03	1	1.14	31.03	3.7	1	30.01	11.3
coastguard	1	30.88	1	1	30.7	3.8	0.9	30.5	15.3
news	1	35.19	1	1.1	34.87	4.3	1	34.27	43
hall	1	34.8	1	1	34.7	3.8	0.9	34.6	39
silent	1	35.44	1	1.2	34.1	3.5	0.9	33.45	21

Table 4: comparison at *QP=26 (F=10)*

QP=26	Full Search			Fast Search (1)			Fast Search (2)		
	Bitrate [%]	PSNR (dB)	SpeedUp [%]	Bitrate [%]	PSNR (dB)	SpeedUp [%]	Bitrate [%]	PSNR (dB)	SpeedUp [%]
foreman	1	35.7	1	1	35.4	3.4	1.6	35.1	3.4
coastguard	1	30.3	1	1	30.4	7	1.5	30.2	6.15
news	1	37.9	1	1	37.8	6.9	1.9	37.56	10.4
hall	1	36.78	1	1	36.77	4.7	1.5	36.29	12.1
silent	1	39.9	1	1.4	38.11	7.4	2.4	38.07	7.8

Fig. 7 foreman bitstream

Fig. 8 Coastguard bitstream

Fig. 9 news bitstream

Fig. 10 hall bitstream

4. Conclusion

In this paper, three different algorithms for motion estimation are tested and compared. The frames are I picture type and motion estimation process is performed using one reference frame.Comparison between FS(1) and FS(2) : Whereas FS(1) operates with small block sizes, In small value of QP and frames with a monotonic motion (foreman), it is faster than FS(2) and produces lower bit-rate. But when the motion details or moving areas decrease, FS(1) has lower speed up than FS(2), because FS(2) first checks larger blocks and has the skip mode. For larger QP or less motility frames, it has faster operation and rather than others shows better bit-rate. When QP increases, The speed up value for FS(1) is decreased, but in FS(2), this is increased strongly. And it is due to the starting search procedure from small block sizes in FS(1) and from larger block sizes in FS(2). And for all QP values, Full search algorithm has the best PSNR, FS(2) has the most speed up and PSNR for FS(1) is better than FS(2). As it seems, when QP increases, the PSNR is reduced for all algorithms, because this leads to choosing larger block sizes and produces a significant amount of energy in motion compensation residual frame. For small QPs, The most amount of bit-rate is for FS(2) and for large QP the most is for FS(1).

References

[1] M. Ghanbari, Standard Codecs : Image Compression to Advanced Video Coding. IEE, 2003.

[2] Paolo De Pascalis, Luca Pezzoni, Gian Antonio Mian, and Daniele Bagni, "fast motion estimation with size-based predictors selection hexagonal search in H.264/AVC encoding", IEEE 2003

[3] Woong IL Choi; Byeungwoo Jeon; "fast motion estimation with modified diamond search for variable motion block sizes", International Conference on Image Processing, Volume 2, pp 371-4, Sept. 2003

[4] H.264 and MPEG-4 Video Compression, Iain E. G. Richardson, John Wiley Press, 2003.

[5] Ki-Hun HAN, Yung-Lyul, "Fast Macroblock Mode Determination to Reduce H.264 Complexity", IEICE TRANS. Fundamentals, Vol.E88–A, No.3 MARCH 2005

[6] Peng Yin; Tourapis, H.-Y.C.; Tourapis, A.M.; Boyce, J.; "Fast mode decision and motion estimation for JVT/H.264**",** International Conference on Image Processing, Volume 3, pp 853-6, Sept. 2003

[7] T. Shanableh and M. Ghanbari, "Heterogeneous video transcoding tolower spatio-temporal resolutions and different encoding formats," IEEETrans. on Multimedia, vol. 2, no. 2, pp. 101–110, Jun. 2000.

[8] R. Srinivasan and K. R. Rao, "Predictive coding based on efficientmotion estimation," IEEE Trans. on Comm., vol. 2, no. 2, pp. 888–896,1985.

[9] H. G. Musmann, P. Pirsch, and H. J. Grallert, "Advances in picture coding," Proc. IEEE, vol. 73, pp. 523–548, Apr. 1985.

[10] M. Kung, VLSI Array Processors. Englewood Cliffs NJ: Prentice Hall,1987.

[11] K. Tavassoli and W. Badawy, "A prototype for parallel motion estimation architecture using full-search block matching algorithm," in International Workshop on Digital and Computational Video (DCV'02), Nov. 2002.

[12] J.-C. Tuan and C.-W. Chang, T.-S. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching vlsi architecture," IEEE Trans. on Cicuits and Systems for Video Tech.,vol. 12,No.1,pp.61–72,Jan. 2002.

Amir Ahmadi received the B.E. degree in Communications, from Tabriz Univ. (Tabriz-IRAN) in 2003. He also received his M.S degree in Electronics from Iran University of Sience & Technology (Tehran-IRAN) in 2006. He has worked as a Planner in Dept. of Planning & Engineering at Mobile Communications Company of Iran (MCCI) since 2006. His research interest includes Digital Image Processing, Image and Video Coding.

M.Mahdi Azadfar received the B.E. degree in Electronics from Ferdowsi Univ. (Mashad-IRAN) in 1992. He also received his M.S degree in Electronics from Tarbiat Modaress Univ.(Tehran-IRAN) in 1995. He has worked as a researcher in Dept. of Multiplex,Network, Information Technology, at Iran Telecom Research Center (ITRC) since 1995. His research interest includes Digital Image Processing, Image and Video Coding, Analog and Digital Television, Multimedia systems, Video Broadcasting over Network. He is currently with the ITRC as a faculty member.