manuscript received: 5 Feb 2021
revised: 10 April 2021
accepted: 15 May 2021

Analysis of Motion Estimation Algorithms in Video

Dadvar Hosseini[1], Mahdi Noshyar[2], Saeed Barghandan[3*], Majid Ghandchi[4]
[1,3,4]Department of Electrical Engineering, Ahar Branch, Islamic Azad University, Ahar, Iran
[2]Associate professor, Department of Electrical Engineering, University of Mohaghegh Ardabili, Ardabil, Iran
[1]Email: dadvarhosseini@gmail.com ,[2]Email: nooshyar@uma.ac.ir
[3]Email: Saeed_Barghandan@yahoo.com (Corresponding Author)
[4] Email: majid.ghandchi@iau.ac.ir

Abstract
One of the most time-consuming parts of video encoding is motion estimation. Motion estimation is an important part of video encoding that improves the amount and accuracy of compression by exploiting the temporal similarity between successive frames. This action is the main cause of maximum compression in the video, which imposes the minimum quality loss on the video. In order to be able to compensate the movement, it is necessary to first find the value of the movement direction, which is called movement estimation.
By doing motion estimation, instead of sending repeated information at consecutive times, the difference between consecutive frames is encoded and sent in different ways. Generally, a motion estimator in video encoding includes fractional pixel motion estimation (FME) and integer pixel motion estimation (IME). The FME method provides better performance with the cost of computational complexity and estimation time on the encryption process.

Keywords: motion estimation, block matching, sequence, frame, pixel

Introduction

Motion estimation and compensation are the foundation of most video compression algorithms. To compensate for the motion, they assume that the current image is an image with a slight change compared to the previous image. This assumption makes it possible to use estimation and interpolation. When one frame is used as a reference, the set of frames in its sequence have a slight difference with each other, which is the result of the movement of objects or the movement of the camera. To make it easier to compare frames, a frame is not encrypted as a whole, but is divided into blocks, and the blocks are encrypted independently. For each block in the frame to be encoded (current frame), the best matching block in the reference frame is searched among the selected number of blocks. For each block, a motion vector is generated that represents the difference between the location of that block and the best corresponding block in the reference frame.

Full Search Algorithm (FSA)

One of the primary algorithms used for block-based motion estimation is the exhaustive search algorithm, which exhaustively tries all positions in the search region. The search algorithm is perfect in the sense that it is desirable if the search scope is carefully defined and ensures a better position matching determination. However, if the search domain is in both W directions with a step size of 1 pixel, and assuming that the search domain is square, a total of $(2w + 1)^2$ displacement steps are required to obtain a motion vector in each block. A large amount of computation is required, especially for a large search window. The high calculation of full search is unacceptable for use in real-time video applications, therefore fast and efficient search methods have been investigated [1].

Fast block Matching Algorithm

Many algorithms have been proposed to reduce the computational complexity of the motion estimation of the full search algorithm while maintaining the quality of the estimation. In all of them, a uniform quadrilateral model is used and it is assumed that the value of the distortion function increases with increasing distance from the point of least distortion. Therefore, by moving away from the block with the least distortion better, the surrounding blocks with high distortion are not good candidates for further matching. But the value of the distortion function in a block position is desirable. Therefore, the uniform quadrilateral assumption is a special case of the position principle. The uniform quadrangle assumption for developing the algorithm only allows checking some candidate blocks in the search area. In addition, the algorithm uses the value of the distortion function to guide the search towards a better match. But usually the compromise between the quality of matching and the number of evaluation points is the best matching criterion.

This method is considered to be better than outdated algorithms due to the time-consuming and complex calculations, despite the fact that it is better suited [2].

Motion Estimation with Fast Method

Jain and Jain were the first to propose the two-dimensional logarithmic (TDL) method to determine the movement path according to the minimum error function [3]. In their method, first, the error value is calculated for 5 locations, one in the center and the other four in the coordinates $(\pm\frac{w}{2}, \pm\frac{w}{2})$. In the next step, three other positions with a size of $\pm\frac{w}{2}$ in the direction the previous minimum move is performed. This is repeated several times, and each time the search size is halved, until the last time it reaches ±1 pixel. Finally, all the surrounding positions with a distance of one pixel are searched. For w=5, 21 positions should be tested, while in the all-position search method, the number of operations is $((2 \times 5) + 1)^2 = 121$. The three-stage search method (TSS) was first proposed by Koga et al. Bar suggested 6 pixels in each frame for the maximum amount of displacement [4]. In this method, first 21 other positions around the coordinate center are searched at a distance of$\frac{w}{2}$. The place that has the minimum difference is selected as the new coordinate center and 8 other positions around this center are searched with a distance of $\frac{w}{4}$. This operation is repeated until the distance from the center of the best match is one pixel. In this method, for example, for the maximum displacement of 6 pixels in the frame, 25 positions should be examined. R gave It is noteworthy that if w > 8, then instead of three steps, more steps are required. For example, for w=15, four search steps are required. This method is proposed as the best motion estimation for H.261 software encoders for video telephony applications. In the Modified Movement Estimation (MMEA) method proposed by Kappagantula and Rayo, before halving the size of the steps, two other positions are also checked [5]. In this method, 19 positions must be tested for w=7. In another method called conjugate direction search (CDS) proposed by Sirinivasan and Rayo, in each position, its two sides are searched with a distance of one pixel, and this process continues iteratively until the minimum MAE is found in the comparisons. This method is almost linear and for w=5, 13 operations are needed [6].

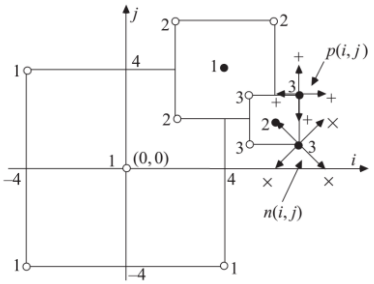

Fig.1. An example of CSA search with w=8 pixels per frame [7]

The CSA cross search method presented by Professor Ghanbari is the same logarithmic method that is used in other methods, with the difference that, in each step, only four positions that are at the end of a cross (x) are tested, but in the last step, not from X, the four positions at the end of + are also tested. With this method, the number of $5 + 4\log_2(w)$ operation is required to move the maximum image w pixels in the frame. Figure (1) shows the steps of this method [7].

The fastest motion estimation method is orthogonal search (OSA) presented by Puri et al. In this logarithmic method, four positions are tested in each step, two positions are in the horizontal direction and two positions are in the vertical direction.

Then the coordinate center is transferred to the location of the least error and this operation is repeated after halving the step. Therefore, the number of $1 + 4\log_2 w$ operations is required for the maximum displacement w. The number of operations required for each of the above methods in the displacement mode of 4 to 16 pixels in the frame is listed in Table (1) [7].

Table 1: Complexity of calculations in the reviewed algorithms [7]

Algorithm	Maximum number of search points	*w*		
		4	8	16
FSM	$(2w+1)^2$	81	289	1089
TDL	$2 + 7\log_2 w$	16	23	30
TSS	$1 + 8\log_2 w$	17	25	33
MMEA	$1 + 6\log_2 w$	13	19	25
CDS	$3 + 2w$	11	19	35
OSA	$1 + 4\log_2 w$	9	13	17
CSA	$5 + 4\log_2 w$	13	17	21

Motion Estimation by Pyramid Method

In all mentioned methods, it is assumed that the brightness of the images changes uniformly and by testing several positions, the final location can be determined. The truth is that in some cases, especially if the range of motion changes from one frame to another is increased, this method may not be very accurate and the value of the range of motion changes will be greater and the measurement accuracy will be lower. One way to solve this problem is to subsample the image, which reduces the speed of the transfer compared to this sampling. Subsampling can be done in multiple steps with a ratio of 2 to 1. In this method, smaller images are formed in the form of a pyramid, as shown in Figure (2) [7].

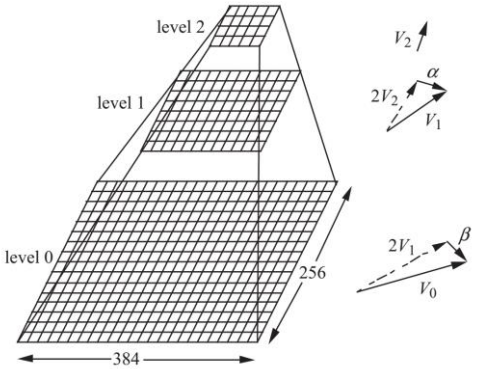

Fig.2. The image of the three-level pyramid [7]

Hence, this method is called BMA pyramid. An image pyramid with 3 levels in (Figure 2) shows that each pixel in the upper level is the average of 4 pixels in a lower level. This structure itself is a kind of low-pass filter. At the highest level of the pyramid (level 2 in Figure (2)), first, motion estimation is done through full or fast search, in 16 x 16pixel blocks. Then, the value of the motion vector is doubled, and at a lower level (level 1 in the above figure), matching search is done around the doubled vector to the size of ±1 pixel. This process continues and ends at the lowest level (zero level). Therefore, if the pyramid has n levels and the maximum movement w, at the highest level of the pyramid, the maximum amount of movement for searching will be equal to $\frac{w}{2^{n-1}}$ [7].

For example, if the maximum initial movement value is 32 pixels per frame, with a 3-level pyramid, the maximum movement at the top level will be 8 pixels per frame, which is a reasonable amount even for determining movement with the full search method or the fast search method. It is worth noting that this method itself represents a

fast way to search for the amount of motion, and the quality of motion compensation can be very close to the quality of motion estimation by full search. Regardless of the speed of movement, the computational complexity can be very close to the fast logarithmic method [7].

Sub-Pixel Motion Estimation

The sub-pixel motion estimation algorithm performs search and motion estimation in two ways (FME) and (IME).

In this method, a large area of pixels is searched as an integer. If the movement is such that it does not reach the position under investigation, it has no effect on selecting or rejecting the desired point. Therefore, this method leaves a lot of distortion and causes a sharp decrease in image quality. In the IME algorithm, first 9 positions in the macroblock are searched and the best matching is done with one of the mathematical functions, then the motion vector is drawn at the winning point.

Starting the search by IME greatly reduces the computational load and selects the best match between macroblock pixels, which is the basis of most motion estimation algorithms. Then, the macroblock is prepared for the next step by interpolation, and after that, the final motion vector is drawn.

Motion estimation in different types of images, in addition to using segmentation and blocks of different sizes, uses different algorithms based on the type of image structure (complexity of content and time dependencies). The most time-consuming part is the variable size of motion estimation blocks. The IME method searches macroblocks with only one pixel precision. Search speed and high distortion and low PSNR (about 25.6dB in the Furman frame) are prominent features of this algorithm.

FME search has better performance and higher computational complexity than IME. Although the search complexity is reduced by a large number of FME algorithms, the sum of the calculations required by FME is still significant.

The half-pixel fast search algorithm performs the search with half-pixel accuracy. This algorithm is widely used in H.262 and MPEG-2 standards. The result of this method creates images with relatively good resolution (about 27.2dB in the Furman frame).

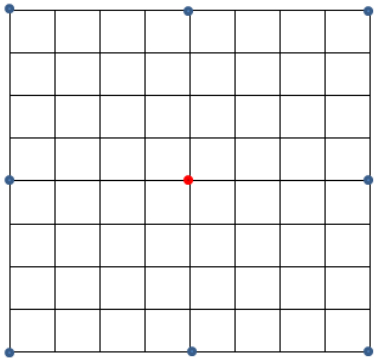

Fig.3. The specified pixel with the least distortion compared to the reference block

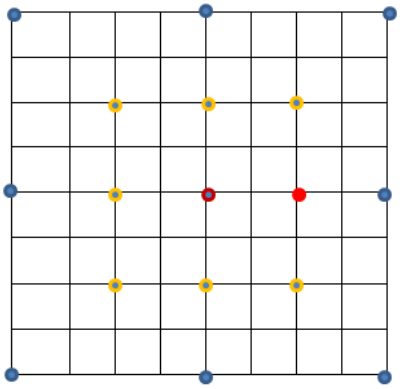

Fig.4. The point obtained by half-pixel search

The quality of the image has been improved with the advancement of estimation algorithms, the volume of compression and its accuracy up to a quarter of a pixel. In an experiment, it was shown that if only 48 points are searched by the quarter pixel algorithm, it achieves the same quality ratio as the full search Figure (5). But practically, this method examines at least 16 points. More search points do not show reasonable performance compared to the obtained quality ratio. Quad-pixel fast search algorithm with high search accuracy is used for motion estimation in HDTV and H.264 (about 29.3dB in Furman frame).

Fast estimation algorithms use the current frame with non-overlapping segmentation to try to find a block of the reference frame that best matches. The block in the reference frame moves around the position of the current frame block within the search window. Best block matching provides minimum distortion in the search window. Therefore, this algorithm increases the search speed significantly, without creating distortion.

Many types of efficient computing have been developed in this regard. The most common ones are three-steep search (TSS), new three-steep search (NTSS), 4-steep search (4SS), gradient-based search (BBGDS) and rhombus search (DS).

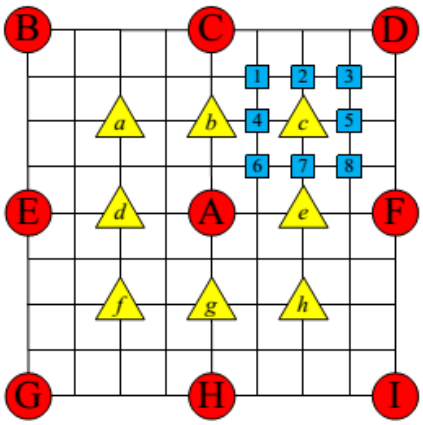

Fig.5.quarter Pixel search in full search [8]

Discussion

In order to evaluate the performance of the fast hierarchical search algorithm, the fast hierarchical search algorithm has been implemented in the JM11.0 framework related to H.264/avc. A PC computer with Intel Core2 core, 3.1-GHz CPU and 4-G RAM is used in this experiment. Six standard video sequences (container qcif, silentqcif, carphone qcif, foreman qcif, mobile qcif and Stephen qcif) have been investigated in this experiment. The IPPP encryption structure uses one frame as an intra-frame and four frames as a follower inter-frame, with a sequence search interval of [-15+15]. The optimal distortion rate and number of encoded frames are selected for each video sequence.

Table (2) compares FHFPS with HFPS and PMFPS. It used average PSNR (dB) and bit rate (kbits/s) to evaluate video quality. Computational complexity is measured in motion estimation time (s).

Table 2: Comparison between fast algorithms for fractional pixels [10]

	Container (176×144)			*Silent (176×144)*		
	PSNR	ME time	kbits/s	PSNR	ME time	kbits/s
HFPS	36.00	168.94	27.46	35.79	172.77	54.45
PMFPS	35.86	69.24	28.74	35.76	74.19	56.48
FHFPS	35.98	58.31	28.88	35.76	60.35	56.73
	Carphone (176×144)			*Foreman (176×144)*		
	PSNR	ME time	kbits/s	PSNR	ME time	kbits/s
HFPS	37.23	190.53	69.70	36.53	180.61	98.05
PMFPS	37.14	82.07	70.93	36.42	89.65	99.87
FHFPS	37.21	68.49	71.36	36.52	62.58	99.94
	Mobile (176×144)			*Stefan (352×288)*		
	PSNR	ME time	kbits/s	PSNR	ME time	kbits/s
HFPS	33.31	175.26	233.9	35.48	314.82	663.7
PMFPS	33.21	76.32	235.1	35.36	128.1	671.1
FHFPS	33.28	61.47	236.6	35.47	113.72	674.1

From Table (2-4), it can be seen that FHFPS is faster than HFPS and PMFPS. On average, a computation reduction of 64.5% can be achieved compared to HFPS. Also, FHFPS performance is almost similar to HFPS performance in terms of PSNR and bit rate. The results show that a slight increase in bit rate is proposed by FHFPS and the average increase in bit rate is 2.56%.

It can also be understood from this table that in FHFPS the quality improvement in terms of PSNR is higher than PMFPS. For example, the PSNR of FHFPS is 0.12 dB greater than that of PMFPS for containerized video sequences. For the video sequence, FHFPS has a PSNR increase of 0.1dB compared to PMFPS. For testing other video sequences, a slight PSNR improvement is observed over PMFPS. The number of FHFPS search points is limited compared to PMFPS search points. This means that the complexity of calculations related to FHFPS is similar to PMFPS [9].

References

[1] International Organization for Standardization. ISO/IEC 15938-5:2003: Information Technology—Multimedia Content Description Interface—Part 5: Multimedia Description Schemes, 1st edn. Geneva, Switzerland. (2003).

[2] Y. Sung Ho. Advanced Video Coding for Next-Generation Multimedia Services. Published by InTech Janeza Trdine 9, (2012), Ch. 2, pp. 121-164.

[3] J.R. Jain and A.K. Jain. Displacement measurement and its application in interframe image coding. IEEE Trans. 29(12): 1799–1808. (1981).

[4] Koga, K. "Motion compensated interframe coding for video conferencing," in *Proc. Nat. Telecommun. Conf., New Orleans, LA*, , pp. 5.3.1–5.3.5, (1981).

[5] S. Kappagantula and K.R. RAO, Motion compensated predictive coding, Proceedings of International Technical Symposium, SPIE, San Diego. United States, pp. 1-7, (1983).

[6] R. Srinivasan and RAO, K.R. Predictive coding based on efficient motion estimation, International Conference on Communications Amsterdam, Nederland, pp. 521–526, (1984).

[7] Mohammed Ghanbari. Standard Codecs Image compression to advanced video coding. 3rd Edition. (2011).

[8] F. Wang, Y. Li, H. Jia, X. Xie and W. Gao. National engineering laboratory for video, An efficient fractional motion estimation architecture for avs real-time full HD video encoder, IEEE International Conference on Imaging Systems and Techniques Proceedings, Manchester, UK, pp. 1-6, (2012).

[9] Sh. Metkar and S. Talbar, Motion Estimation Techniques for Digital Video Coding, Applied Science and Technology, 2013, Ch. 2, pp. 25-50.

[10] Z. Wei and Z. Khin, A fast hierarchical 1/4-pel fractional pixel motion estimation algorithm of H.264/AVC video coding, 8th Conference on Industrial Electronics and Applications, Chicago, 2013, pp. 891-895.