A New Fuzzy Based Motion Estimation Algorithm in Video Compression

S. MohammadReza Soroushmehr
Department of Electrical and Computer Engineering, Islamic Azad University, Khomeinishahr Branch, Isfahan, Iran
Email: soroushmehr@iaukhsh.ac.ir

Received: October 2009 Revised: January 2010 Accepted: February 2010

ABSTRACT:
Motion estimation (ME) is a major part of modern video codecs, which requires a huge amount of computations. To overcome this drawback, various techniques have been proposed. In this paper with the aid of fuzzy inference, an efficient algorithm is devised. The proposed algorithm exploits spatial correlation as well as temporal correlation among motion vectors. This algorithm uses fuzzy rules to determine the initial motion vector. After that, a local search around initial vector is carried out. In order to decrease the complexity of the algorithm, a look-up table is used. In this table, defuzzified values are stored. Also, to further reduction of complexity, few computations are performed in the proposed algorithm for stationary and quasi stationary blocks. To determine which block can be regarded as stationary or quasi stationary block, a simple comparison with a predefined threshold is done. The experimental results show that the proposed algorithm performs better than other fast block matching algorithms in terms of picture quality and computational complexity.

KEYWORDS: Block Motion Eestimation, Fuzzy reasoning, Spatial Correlation, Temporal Correlation.

1. INTRODUCTION

A video consists of a sequence of images called frames. Storage and transmission of video image sequences (frames) requires time and memory space as well as wide bandwidth for transmission. Nowadays, video ranks first, in terms of volume, among all types of produced data [1]. Broadcasting only one channel of ITU-R 601, in uncompressed digital form, requires a transmission bit rate of 216*Mbps*. A 4.7 Gigabyte DVD could store only 87 second of this uncompressed video. In this video format, only one camera is used. It is obvious that the amount of data is increased when the number of video channel is increased. Compression of video image sequences could be effective in reducing the required space.

During the past three decades, numerous compression standards such as H.261, H.263, H.264, and MPEG1, MPEG2 and MPEG4 for different video applications have been introduced. All of these standards have a section dedicated to reduction of temporal redundancy [2]. One of the techniques used to reduce temporal redundancy is motion estimation which results in better compression of video sequences. Various motion estimation techniques have been proposed yet. These techniques can be classified into transform domain and spatial domain. The latter is used frequently due to its simpler operations. Among spatial domain ME techniques, block-matching algorithms (BMA) and affine based transform are used and BMA is used more in video coding standards [3]. In the BMA a frame is partitioned into non-overlapped blocks. The displacement of a block in the current frame is measured with respect to a matched block in the reference frame. To find a matched block a search area is considered for every block. To form the search area *W* pixels extensions are considered on the edges of a corresponding block in the reference frame. In order to find the matched block a criterion function is defined. There are some criteria such as *sum absolute difference* (SAD), *sum square difference* (SSD), *mean absolute difference* (MAD), *mean square difference* (MSD), *pel difference classification* (PDC) [4], *minmax* [5] and *normalized cross correlation* (NCC) for finding the best matched block. Due to simple implementation and efficient performance, SAD, defined in Equation (1) is usually used as a criterion function.

$$SAD(x,y)=\sum_{i=1}^{N}\sum_{j=1}^{N}\left|\begin{matrix}I_{cur}(m+i,n+j)-\\ I_{ref}(m+x+i,n+y+j)\end{matrix}\right| \qquad (1)$$

In this equation I_{cur} and I_{ref} are the pixels in the current and reference frames respectively. Also, it is assumed that the size of each block is N×N. Moreover,

(m, n) is the left-top coordinates of the current block in the current frame. After partitioning the current frame into non-overlapping blocks, for each block a search area is designated in the reference frame. The size of this area is (2W+1) × (2W+1) pixel. Within the search area, any block which produces minimum distortion is the matched block. The vector which measures the displacement is called motion vector (MV). In the Equation (1) the motion vector is equal to (x, y). Therefore SAD is computed for the block with left-top coordinates equal to (m, n) and the reference block with left-top coordinates equal to (m + x, n + y).

In Figure (1) motion vector, search area, reference and current frames, reference and current blocks and best matching block are all shown.

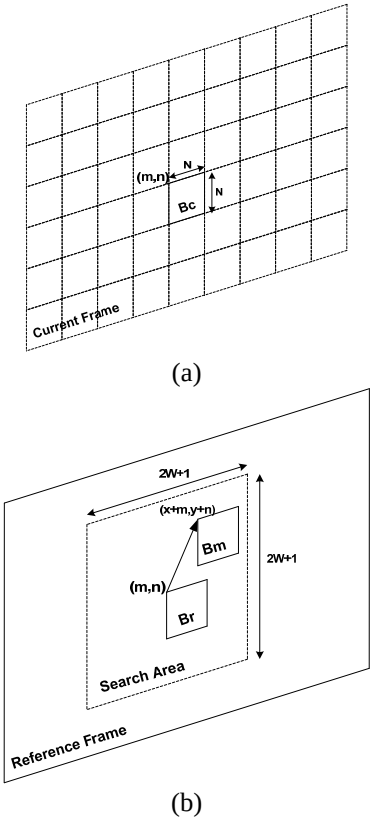


Fig. 1. Illustration of current frame (a), reference frame (a), search area and motion vector (b).

In this figure, the current block, B_c, is the block that its motion vector must be found. Also B_r is the corresponding block to B_c, in the reference frame. The best matching block, B_m, is a block in the search area that has the most similarity to B_c. The best matching block is the one that minimizes the SAD criterion defined in Equation 1.

Since an object usually occupies more than one block of an image, motions of neighboring blocks are similar to each other. This is known as spatial correlation of motion vectors. Also, due to inertia in the movement of objects, there is correlation among motion vectors of blocks of consecutive frames. This means that if a block in a frame has been displaced with a motion vector of (x, y) then there is a high probability that its corresponding block in the reference frame has a motion vector of or close to (x, y). This is known as temporal correlation of motion vectors. To compress a video sequence, spatial correlation as well as temporal correlation is exploited. Also, human visual system characteristics are used to achieve better compressions.

There are too many algorithms for finding the best matched block. *Full search algorithm* (FSA) is the simplest that finds the block in the reference frame with minimum SAD. The FSA is a straightforward routine that compares a block with all of the blocks in the search area to find the best matched block. Though it seems simple, this algorithm requires complex computations which prevent it from being a real time scheme [6]. To alleviate this shortcoming numerous algorithms have been devised for fast search and real time implementation. Some of these algorithms only inspect selected number of blocks from the search area to find the matched block. Some of these algorithms are logarithmic search [7], three step search (3SS) [8], four step search (4SS) [9], diamond search (DS) [10], and octagon search [11] algorithms. In all of these algorithms unimodal error surface assumption (UESA) is considered [7], but it is not always true [5]. Hence, in some algorithms initially the MV is predicted and then the search is performed around the predicted vector. For example in the EFS algorithm based on the spatial correlation among MVs, using fuzzy logic, prediction vector is extracted[12]. In *co directionality assisted predicted search* (CAPS) [13] and *adaptive predicted direction search* (APDS) [3] algorithms it is assumed that the prediction vector follows the same direction as the neighboring blocks. In the *fast adaptive motion estimation* (FAME) algorithm [14], a modified version of *predictive field adaptive fast search technique* (PMVFAST) [15] recommended by MPEG4 standard, spatial and temporal predicted MVs are used.

In this paper, based on fuzzy logic, we propose an algorithm that finds the best-matched block more accurately than other fast algorithms. To reduce the implementation complexity and to increase prediction speed, a fuzzy reasoning procedure is implemented with the table-look-up approach.

The organization of the paper is according to the following manner. Section 2 gives a brief review of fuzzy logic and presents our algorithms. In section 3, the proposed algorithm is compared with other fast ME algorithms through simulations performed on standard video sequences. Concluding remarks appear in section 4 of the paper.

2. FUZZY INFERENCE MOTION ESTIMATION ALGORITHM

Since Zadeh proposed the idea of fuzzy sets, a variety of applications of fuzzy logic have been implemented in various fields, ranging from industrial control to financial management [11]. The behavior of systems which are based on fuzzy reasoning can be represented with if-then rules. For example, in a fuzzy system with two inputs, X and Y and one output, Z, the following rules can be used.

Rule i : if X is Ai and Y is Bi then Z is Ci (2)

Where (Ai, Bi) and Ci are the antecedent membership functions associated with the fuzzified input and output variables respectively. For example, in motion estimation the following fuzzy rule can be used:

If displacements of two neighboring blocks in horizontal direction are respectively three and four pixels then the displacement of current block in horizontal direction is about three pixels.

To utilize the fuzzy reasoning, first, the fuzzification process must be performed. After fuzzification, the fuzzified inputs are broadcasted to all the If-Then rules. Then the matching degree between the inputs (Xi, Yi) and the antecedent member functions (Ai, Bi) are determined. The output Z must be defuzzified to make use of it. In the proposed algorithm, we use *center of gravity* (COG) algorithm for defuzzification. And also triangle fuzzification is adopted. To simplify the implementation of the defuzzification computations the look-up table approach is utilized.

As discussed earlier, there is spatial and temporal correlation among motion vectors. In Figure 2 a number of blocks of a frame are shown.

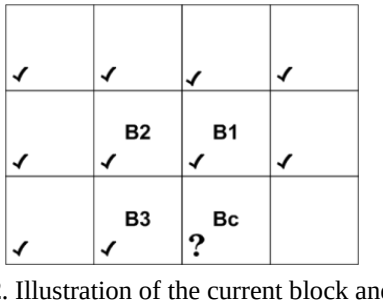
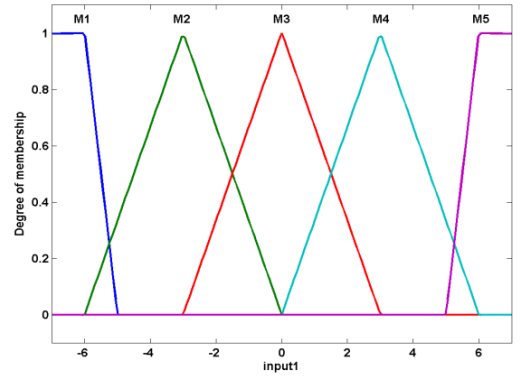


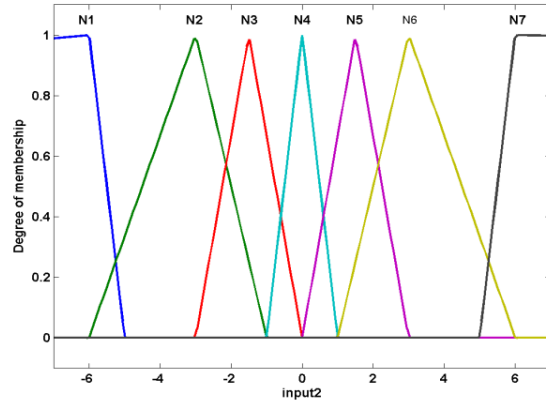
Fig. 2. Illustration of the current block and its neighboring blocks.

In a motion estimation algorithm the motion vectors of blocks are usually found in a row-by-row manner starting from top left corner. Therefore, in Figure 2 when we get to the block which is marked Bc, that all blocks have a check mark,✓, have known MVs. In this work, we use the MVs of blocks B_1, B_2 and B_3 because of their strong spatial correlation with Bc. Also, to exploit temporal correlation, we use the MV of the block corresponding to B_c in the reference frame. We refer to this block as B_4. We show MVs of blocks B_i as (MVX_i, MVY_i) where $1 \le i \le 4$. Here, MVX_i and MVY_i are the displacement of block Bi in the horizontal and vertical directions respectively with respect to the original block ($1 \le i \le 4$).

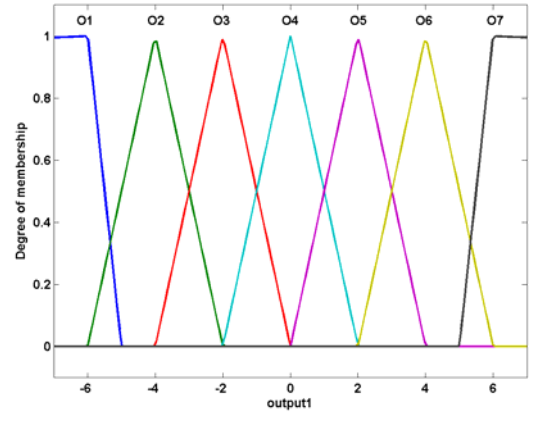
According to our simulation results we use the membership functions shown in Figure 3.



(a)



(b)



(c)

Fig. 3. Membership functions of Inputs (a), (b) and output (c).

The rules used in this paper are shown in Table 1.

Table 1. 35 Fuzzy rules used in the proposed algorithm

	Y1	**Y2**	**Y3**	**Y4**	**Y5**	**Y6**	**Y7**
X1	Z1	Z1	Z2	Z3	Z3	Z4	Z5
X2	Z1	Z2	Z2	Z3	Z4	Z5	Z5
X3	Z2	Z2	Z3	Z3	Z4	Z5	Z6
X4	Z3	Z3	Z4	Z4	Z5	Z5	Z6
X5	Z3	Z4	Z4	Z5	Z6	Z6	Z7

One important characteristic of video sequences is that most of the blocks can be regarded as stationary or quasi stationary blocks [9]. To determine which block is stationary, SAD (0, 0) (in Equation 1) can be compared with a predefined threshold, *T*. It means that in the first step of the algorithm SAD (0, 0) can be compared with *T*. If SAD (0, 0) is less than *T*, (0, 0) is regarded as motion vector and further computations are eliminated. Hence it results in complexity reduction.

Using this characteristic of video sequences and spatial and temporal correlation among motion vectors and with the aid of fuzzy inference, proposed algorithm called *Fuzzy Inference Search Algorithm* (FISA) is described as below:

1- If the SAE (0, 0) value, computed with the Equation (1), is less than a predefined value ,*T*, then the mentioned block is determined as a stationary block else the following steps will run.

2- The value of (MVX_2+7) and (MVX_1+7) are respectively used as the column and row address of a lookup table with 15×15 entries. Each entry of the table is the result of the defuzzification processes. The obtained value of the table is stored in MVX_H variable.

3- The step 2 is repeated for the following values.

   a. $(MVX_3 + 7)$, $(MVX_2 + 7)$

   b. $(MVY_2 + 7)$, $(MVY_1 + 7)$

   c. $(MVY_3 + 7)$, $(MVY_2 + 7)$

The extracted values of the table are stored in MVX_V, MVY_H and MVY_V variables respectively.

4- The step 2 is repeated for $(\frac{MVX_V + MVX_H}{2} + 7)$ and $(MVX\,4 + 7)$ and also for $(\frac{MVY_V + MVY_H}{2} + 7)$ and $(MVY\,4 + 7)$ to extract values stored in *MVX* and *MVY* variables respectively.

5- (*MVX*, *MVY*) is set as the initial center of a search window. If the minimum distortion among these nine blocks belongs to the center of the search window step 6 is run. Otherwise the next search window is shifted to the minimum point and (*MVX*, *MVY*) is set to this minimum point and this step is repeated.

6- A window is located and the minimum point among the points of this window is computed and stored in (*MVX*, *MVY*). (*MVX* - *MVXc*, *MVY* - *MVYc*) is set as the MV where (*MVXc*, *MVYc*) is the top-left coordinate of the current block.

In Figure 4 the steps of the proposed algorithm are displayed as an example.

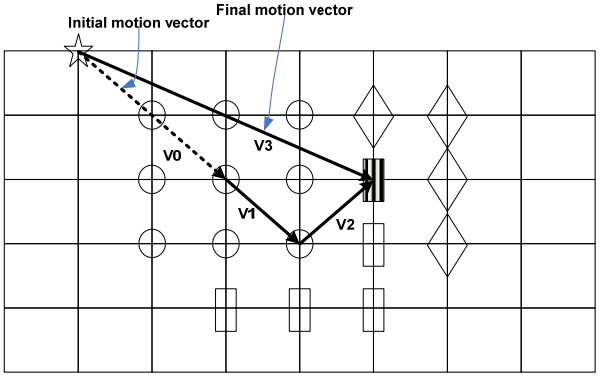


Fig. 4. Illustration of the proposed algorithm steps

It is assumed that the initial motion vector is (2, 2). This vector is shown as V_0. A search window (the points shown in circle) is centered on (2, 2). Suppose that the minimum point among these nine points is (3, 3). Therefore a new search window is centered on (3, 3). In other words, five new points (the points shown in rectangular) are added. Suppose that in this step of the algorithm, (4, 2) is the minimum one. The algorithm is repeated and four new points (the points shown in diamond) are added. Since the minimum point is the center of the window (i.e the point (4, 2) is the minimum) the search is stopped and the motion vector (4, 2) is regarded as the final motion vector in this example. This vector is shown as V_3 in Figure 4.

In addition to introducing the steps of the algorithm, its flowchart is presented in Figure 5.

In stage 1 of the flowchart, the distortion of zero motion vector (i.e. SAD (0, 0)) is compared with a predefined threshold, T. This stage is equivalent to step one of the algorithm.

The stages from 3 to 8, use a lookup table called LUT. For example, in the 3rd stage of the flowchart, (MVX_2+7) and (MVX_1+7) are respectively used as the column and row address of LUT. Entries of LUT are the result of the defuzzification process. The obtained value of LUT is stored in MVX_H variable. Similar operations are performed from 4th stage to 8th stage.

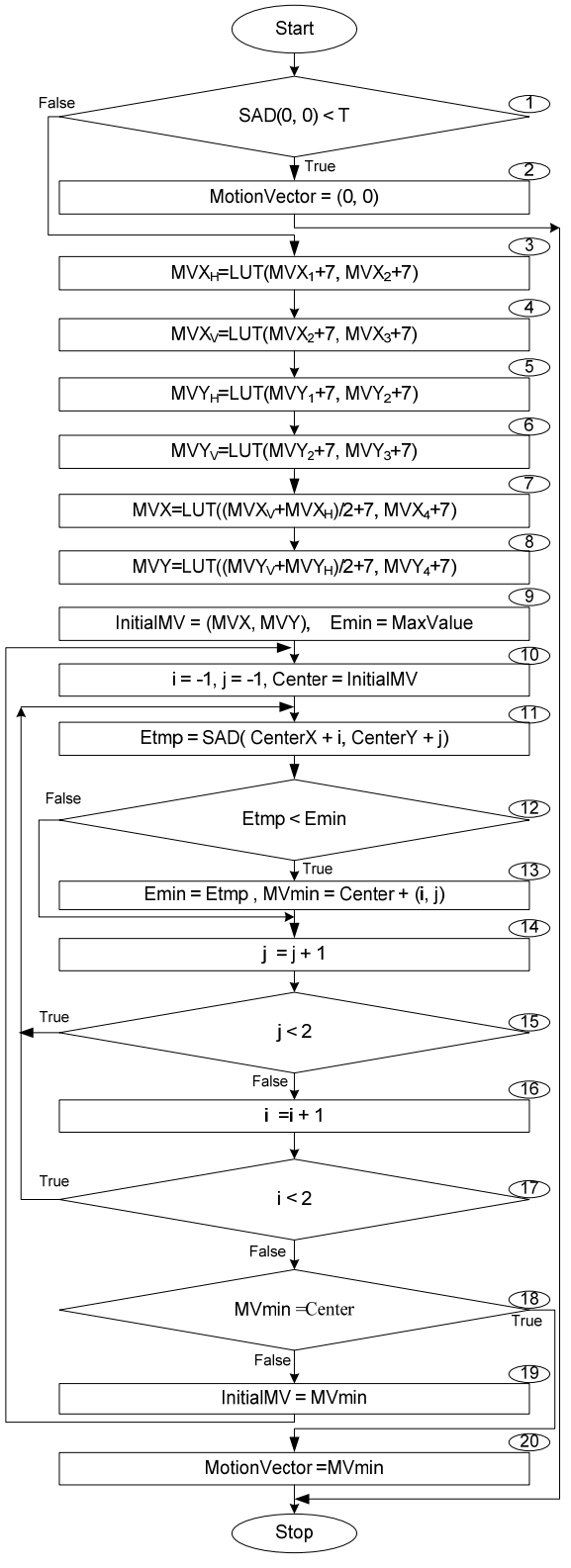


Fig. 5. Flowchart of the proposed algorithm

After performing these stages, (MVX, MVY) is used as an initial motion vector, shown as *InitialMV*. After finding the initial motion vector, a 3×3 window is centered on the end of this vector. Hence, 9 blocks are examined. These include the block with coordinates same as Center and eight other blocks with coordinates that are one unit different in horizontal and or vertical directions. To get the coordinates, variables "*i*" and "*j*" are varied by -1 to +1. This is shown in steps 11 to 17 of the flowchart. *Etmp* used as a temporary variable to store value of SAD. The minimum value among nine SADs is stored in *Emin* variable. Therefore the initial value of this variable is set to a large number. Suppose that the size of a block is 16×16 pixel and each pixel needs one byte. Therefore the maximum value of SAD will be 65280. Hence, *Emin* is set to a value larger than 65280 such as 65281. If minimum error is produced by a block which is located at the center of the search window (MVmin=Center) then step 19 of the flowchart is performed and search is repeated and minimum point among nine blocks will be the new search center (stage 19 of the flowchart). Otherwise step 20 will be carried out and search is stopped. In the final stage (i.e. 20^{th} stage) MVmin is set as final motion vector of the block.

3. SIMULATION RESULTS

In this section, the simulation results of the proposed algorithm (FISA), 3SS, 4SS and EFS algorithms by using standard video sequences are presented. Thirty frames of "*Susie*", "*Garden*", "*Trevor*" and "*Tennis*" standard video sequences, with resolution 352×240, are used. In addition, 30 frames of "*Foreman*", with 176×144 resolution, and 30 frames of "*Caltrain*", with 512×400 resolution, are used. The maximum displacement (*W*), the block size and the *T* values are set to 7, 16×16, and 512 respectively.

Comparisons are performed in terms of CG (Computing Gain) and QG (Quality Gain) [16]. In table 2, the mentioned algorithms, in terms of QG parameter, are compared for 30 frames of standard sequences. QG, defined in equation 3, is the difference between the peak *signal to noise ratio* (PSNR) of FISA and PSNR of other algorithms.

$$QG = PSNR(FISA) - PSNR(X) \quad (3)$$

In Equation (3), X is replaced with one of FSA, 3SS, 4SS, EFS algorithms, The higher the QG, the better the quality of reconstructed frame of FISA. In comparison with 3SS, 4SS, and EFS, FISA has positive QG except for Garden sequence and in comparison with EFS. It means that in most cases, FISA has superior performance. For example according to Table 2, for *Caltrain* sequence and in comparison with 3SS, QG is equal to 0.98dB. In other words, the PSNR of FISA algorithm is 0.98dB greater than that of 3SS algorithm. According to the table 2, the PSNR values of the EFS, FISA and FSA are very close to each other. Only in "Garden" sequences, the PSNR value of the

EFS is slightly higher than those of the FISA.

Table 2. QG of FISA as compared with FS, 3SS, 4SS, EFS algorithms

	FS	3SS	4SS	EFS
Foreman	-0.23	0.18	0.12	0.19
Caltrain	-0.11	0.98	0.31	0.01
Susie	-0.29	1.1	0.76	0.09
Garden	-0.15	0.71	0.36	-0.01
Trevor	-0.05	0.42	0.31	0.03
Tennis	-0.25	1	0.36	0.15
Average	-0.18	0.73	0.37	0.08

In Table 3, the mentioned algorithms are compared with the parameter named CG. This parameter is computed according to Equation (4).

$$CG = \frac{NSP(X)}{NSP(FISA)} - 1 \quad (4)$$

In this equation, NSP stands for *number of search points*. Higher value in CG results in lesser computation in FISA. For example, in the FSA and for Garden sequence, Equation (1) is executed for about 202 points, while in FISA, this number is about 11. It means that FISA is about 18 times faster than FSA. Also, for *Susie* sequence, FISA is about 23 times faster than FSA.

Table 3. CG of FISA as compared with FS, 3SS, 4SS, EFS algorithms

	FS	3SS	4SS	EFS
Foreman	11.63	0.50	0.27	0.07
Caltrain	15.82	0.91	0.53	0.05
Susie	20.80	1.50	0.95	0.38
Garden	17.19	1.09	0.72	0.10
Trevor	22.14	1.61	0.89	0.28
Tennis	17.25	1.09	0.62	0.06
Average	17.47	1.12	0.66	0.16

According to Table 3, in all circumstances the CG is positive, which means FISA saves more computations. For example, in "Caltrain" sequence, FISA is 5% faster than EFS. Or on average, FISA is 112% faster than 3SS and 16% faster than EFS algorithms. In "Trevor" sequence that has small motions, the QG of FISA is higher than those of 3SS, 4SS, EFS. In this sequence, the FS algorithm has only 0.05dB higher QG. Also, in this sequence FISA is about 23 times faster than the FS algorithm.

4. CONCLUSION

In this paper, a fast method was proposed for finding motion vectors in motion estimation part of a codec. In the proposed algorithm, correlation between motion vectors of neighboring blocks in one frame and sequence frames was used and by using fuzzy logic, the motion vector of the block was predicted. Because of the complex computations of defuzzification, in this paper, look-up table was considered. The proposed algorithm needs fewer searches for the images, which has very slow movements or background without decreasing in quality of reconstructed images. According to the simulation results, the proposed algorithm was about 23 times faster than the FS whereas there was no significant difference between PSNR of FS and that of the proposed algorithm.

REFERENCES

[1] Richardson; **Iain E.G. Video Codec Design**, England: John Wiley & Sons Ltd, (2002)

[2] Ghanbari, Mohammad. **Video Coding, an Introduction to Standard Codecs**, London: IEE Series, (1999)

[3] Nam, Jae Yeal, *et al*; **"New Fast Search Algorithm for Block Matching Motion Estimation Using Temporal and Spatial Correlation of Motion Vector",** *IEEE Trans. Consumer Electronics*, Vol. 46, No. 4, pp. 934-42, (2000):

[4] Gharavi, H., and Mills, Mike; **"Block Matching Motion Estimation-New Results",** *IEEE Trans.Circuits* Sys, 37, No. 5, pp. 649-51, (1990)

[5] Chen, Mei-Juan *et al*, **"A New Block-Matching Criterion for Motion Estimaton"**, *IEEE Trans. Circuits. Sys. Video Technol*, Vol. 5, No. 3, (1995): 231-6.

[6] Sorwar, Golam, Manzurand Murshed and Laurence S. Dooley; "**A Fully Adaptive Distance-Dependent Thresholding Search (FADTS) Algorithm for Performance Management Motion Estimation"**, *IEEE Trans. Circuits Sys. Video Technol*., Vol. 17, No. 4, pp. 429-40, (2007)

[7] Jain, J and A. Jain; **"Displacement Measurement and its Application in Interframe Image Coding"** *IEEE Trans. Commun., COMM-29*, pp. 1799-1808, (1981)

[8] T. Koga, *et al*.; **"Motion Compensated Interframe Coding for Video Conferencing"**, *Paper presented at National Telecommunication Conference, New Orleans*, LA, (November 1981)

[9] Po, Lai-Man and Wing-Chung Ma; **"A Novel Four-Step Search Algorithm for Fast Block Motion Estimation",** IEEE *Trans. Circuits Syst. Video Techno*., Vol. 6, No. 3, pp. 313-17, (1996)

[10] Tham, Jo Yew *et al*; **"A Novel Unrestricted Center-Biased Diamond Search Algorithm for Block Motion Estimation"**, *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 8, No. 4, pp.: 369-77, (1998)

[11] L.P. Chau, and C. Zhu, **"A Fast Octagon Based Search Algorithm for Motion Estimation"**, *Elsevier science, Signal Processing journal*, Vol. 83, No. 3, pp. 671-75, (2003)

[12] Chen, Pei-Yin and Jer Min Jou, **"An Efficient Blocking Matching Algorithm Based on Fuzzy Reasoning"** *IEEE Trans Syst, Man, Cybern.-Part B*, Vol. 31, No. 2, pp. 253-59, (2001)

[13] Soroushmehr, S.M.Reza, Shadrokh Samavi and Shahram Shirani, **"Block Matching Algorithm Based on Local Codirectionality of Blocks",** *Paper presented at the IEEE International Conference on Multimedia and Expo (ICME)*, New York, (June 2009)

[14] Ahmad, Ishfaq *et al*, **"A Fast Adaptive Motion Estimation Algorithm"**, *IEEE Trans. Circuits Sys. Video Technol.*, Vol. 16, No. 3, pp. 420-38, (2006)

[15] Tourapis, A. M., O. C. Au, and M. L. Liou, **"Fast Block-Matching Motion Estimation Using Predictive Motion Vector Field Adaptive Search Technique (PMVFAST)",** *ISO/IEC JTC1/SC29/WG11 MPEG2000/M5866, Noordwijkerhout, The Netherlands,* (March 2000)

[16] Tsai, Jang-Jer and Hsueh-Ming Hang, **"Modeling of Pattern-Based Block Motion Estimation and its Application",** *IEEE Trans. Circuits Sys. Video Technol.*, Vol. 19, No. 1, pp. 108-13, (2009)