The Third AFSS(1998). 750－755

# A NOVEL FUZZY SEARCH ALGORITHM FOR BLOCK MOTION ESTIMATION

Pei-Yin CHEN, Jer Min JOU, Jian-Ming SUN

Department of Electrical Engineering, National Cheng Kung University
1 University Road, Tainan, Taiwan 70101, Republic of China
Tel: +886-6-2387092 Fax: +886-6-2345482 E-mail:jou@cad2.ee.ncku.edu.tw

**Abstract**

Due to the temporal and spatial correlation of the image sequence, the motion vector of a block is highly related to the motion vectors of its adjacent blocks in the same image frame. If we can obtain useful and enough information from the adjacent motion vectors, the total number of search points used to find the motion vector of the block may be reduced significantly. Using that idea, an efficient fuzzy prediction search (FPS) algorithm for block motion estimation is proposed in this paper. Based on the fuzzy inference process, the FPS can determine the motion vectors of image blocks quickly and correctly.

**Keywords:** Fuzzy inference, block motion estimation, motion vector

## 1. Introduction

In general, the motion estimation/com-pensation techniques are efficient methods to reduce the required bitrate by eliminating temporal redundancy of the image sequence. The most popular technique for motion estimation is the blocking matching algorithm (BMA) [1-5], which finds the best match for a block in the current frame within a search area in the previous frame.

In a typical BMA, a frame is divided into blocks of $M \times N$ pels or, more favorably, square blocks of $N^2$ pels. Then for a maximum motion displacement of $w$ pels per frame, the current block of pels is matched against a corresponding block at the same coordinate but in the previous frame, within the square window of width $N+2w$. A straightforward BMA is the full search algorithm (FS), which exhaustively searches for the best-matched block within a search area in the previous frame to get the optimal motion vector. However, the computational requirement of FS is often too high. Thus, efficient algorithms such as the three step search algorithm (3SS) [1], the cross search algorithm (CS) [2], the four step search algorithm (4SS) [4], and the simple and efficient search algorithm (SES) [5] have been developed to reduce the computational complexity significantly.

Due to the temporal and spatial correlation of sequence image, the motion vector of the current block is highly related to the motion vectors of its adjacent blocks. However, the real relation among the motion vectors of the blocks is unknown and hard to extract, so fuzzy reasoning [6][7] is employed to solve the problem. In this paper, we propose a novel fuzzy prediction search algorithm (FPS) for block motion estimation. Based on the fuzzy reasoning, the motion vectors of the adjacent blocks in the current frame are used to find the predicted motion vector of the current block. Using the predicted vector, the FPS can determine the initial center of the search area for the current block. Then, the FPS searches the whole search area with a $3 \times 3$ movable search window until the local minimum point lies in the center of the present search window, or the iteration times of search loop reaches the given maximum iteration times. Since the FPS starts searching from the better initial search center, it can determine the motion vectors of image blocks more correctly and quickly

According to the experimental results, the proposed method has the smallest average MSE, the largest average PSNR, the smallest average prediction errors, the smallest entropy of prediction errors, the least percentage of unpredictable pels (pels with absolute prediction errors larger than three, over a range of 255, are classified as unpredictable pels), and the least average search times than other search methods such as, 3SS, CS, 4SS and SES for some image sequences.

## 2. The proposed algorithm

2.1 Fuzzy prediction for motion vector

Since the relation among the adjacent motion vectors in a image frame is not unknown, the fuzzy inderence is applied to predict the motion vectors of blocks. The fuzzy inference process, performed in the design, is based on the concepts of fuzzy implication and the compositional rules of inference for approximate reasoning [10]. Figure 1 shows the position-relation of the current block and the reference blocks used for fuzzy prediction in which each block is of size 16×16. The current block, denoted as $B_c$, is located at $(m, n)$, where $m$ $(n)$ is the column (row) number. Its four neighboring blocks, at location $(m\text{-}32, n)$, $(m\text{-}16, n)$, $(m, n\text{-}32)$, and $(m, n\text{-}16)$, are denoted as $B_i$ for $i \in \{1, 2, 3, 4\}$. Here, we use the neighboring motion vectors as the inputs to infer the motion vector of the current block. The corresponding membership functions and the 16 fuzzy control rules used for the prediction problem are shown in Fig. 2. Let the motion vector of $B_i$ be represented by $V_i = [\Delta c_i, \Delta r_i]$, and $\hat{P} = [\Delta\hat{c}_p, \Delta\hat{r}_p]$ mean the predicted motion vector of $B_c$. Then, $\Delta\hat{c}_p$ is calculated by using the following fuzzy prediction steps.

**Step 1:** Use $\Delta c_1$ and $\Delta c_2$ as the fuzzy input $a$ and input $b$ respectively, and then infer the output $\Delta c_H$ with the 16 fuzzy control rules shown in Fig. 2.

**Step 2:** Similarly, use $\Delta c_3$ and $\Delta c_4$ as the input $a$ and input $b$ respectively, and then infer the output $\Delta c_L$ with the 16 fuzzy control rules.

**Step 3:** Then determine $\Delta\hat{c}_p$ as follows:

$$\Delta\hat{c}_p = \frac{(\Delta c_H + \Delta c_L)}{2}.$$

**Step 4:** Similarly, use $\Delta r_1$ and $\Delta r_2$ as the inputs, and then infer the output $\Delta r_H$ with the 16 fuzzy control rules.

**Step 5:** Use $\Delta r_3$ and $\Delta r_4$ as the inputs, and then infer the output $\Delta r_L$ with the 16 fuzzy control rules.

**Step 6:** Then determine $\Delta\hat{r}_p$ as follows:

$$\Delta\hat{r}_p = \frac{(\Delta r_H + \Delta r_L)}{2}.$$

Thus, the predicted motion vector $\hat{P}$ can be determined.

2.2 Fuzzy prediction search algorithm

Suppose that the current block is located at the coordinate $(m, n)$. Let *Count* represent the maximum iteration times of the search loop set by users and BDM mean the block distortion measure used for motion estimation. The FPS algorithm is summarized as follows:

**Step 1:** Utilize the fuzzy inference procedure described in subsection 2.1 to obtain the predicted vector $\hat{P} = [\Delta\hat{c}_p, \Delta\hat{r}_p]$. Then set the initial center of the search area at the coordinate $((m + \Delta\hat{c}_p), (n + \Delta\hat{r}_p))$, and the loop counter $(S)$ to 1.

**Step 2:** Find the minimum BDM point among the nine checking points on a 3×3 movable search window .

**Step 3:** If $S >$ *Count* or the minimum BDM point is located at the center of the present search window, go to **Step 4**; otherwise increase $S$ by 1, perform the search process with the following two search patterns, and then repeat this step.

(a) If the minimum BDM point is located at the middle of the boundary column (row) of the present search window, the center of the next search window is shifted to the minimum BDM point, and 3 additional vertical (horizontal) adjacent checking points are used.

(b) If the minimum BDM point is located at the corner of the present search window, the center of the next search window is shifted to the minimum BDM point, and 3 additional checking points are used.

**Step 4:** Stop the search and calculate the overall motion vector.

Even we use a 3×3 movable search window, only part of the window (not all the nine search points) is searched. Thus, some unnecessary search points can be omitted. With the halfway-stop technique, the FPS algorithm stops the search process as soon as the minimum BDM point is found at the center of the present search window. Two examples of FPS are shown in Fig. 3 with different search paths as

*Count*=8; for the worst case, the total number of search points required is $(9+3\times(8-1))=30$. However, the minimum BDM point will be found at the center of the present search window at the first or second search iteration if we can always make good prediction. Thus, the average search points can be reduced significantly. That can be proven according to the experimental results described in the following section.

## 3. Experimental results and performance comparisons

Since the fuzzy inference procedure requires complex computations, the table-look-up approach is applied to reduce the implementation complexity and to prompt the inference speed. A table with $2^{4\times2}$ entries must be constructed, in which each entry needs 4 bits to store a predicted vector value between -7 and 7. Thus, it need 128 bytes of memory.

In our experiments, the BDM is defined to be the mean squared error (MSE) and the block size is fixed at $16\times16$. The first 100 frames of the " Football," "Mobile," "Windmill," "Flower," "Tennis," "Salesman," and "Miss America" sequences (in CIF format) are used to test the proposed algorithm. It is noted that these sequences contain different combinations of still, slow, and fast moving objects. The comparisons are made by using six search algorithms: 1) FS, 2) 3SS, 3) CS, 4) 4SS, 5) SES and 6) FPS in terms of six different measures. These measures include 1) average MSE per pixel as shown in Table I, 2) average PSNR as shown in Table II, 3) average prediction errors per pixel as shown in Table III, 4) average entropy of prediction errors as shown in Table IV, 5) average percentage of unpredictable pels per frame (pels with absolute prediction errors larger than three, over a range of 255, are classified as unpredictable pels [3]) as shown in Table V, and 6) average search points per frame as shown in Table VI. In the tables, FPS(*Count*) is the proposed algorithm in which *Count* means the maximum iteration times. While the result of FS is set to 100%, the average percentages of results for other search algorithms are shown in the last "Percentage" column of the tables. According to the experimental results, the proposed FPS performs better than such algorithms as 3SS, CS, 4SS, and SES in terms of the above six measures.

## 4. Conclusions

An efficient fuzzy prediction search algorithm for block motion estimation is presented in this paper. Using the fuzzy reasoning, the algorithm can determine the motion vectors of image blocks quickly and correctly. To reduce the implementation complexity and to prompt the prediction speed, the fuzzy prediction procedure is implemented with the table-look-up approach. The experimental results show that the proposed algorithm works better than other search algorithms. Now, the VLSI architecture for the FPS algorithm is under developing.

## 5. Acknowledgement

This research was supported in part by the National Science Council, Republic of China, under the Grant NSC-87-2213-E-006-031.

## References

[1] T. Koga, K. Iinuma, A. Iijima and T. Ishiguro, "Motion-compensated interframe coding for video conferencing," in *Proc. NTC81*, pp. C9.6.1-9.6.5, New Orleans, L. A., 1981.

[2] M. Ghanbari, "The cross-search algorithm for motion estimation," *IEEE Trans. Communications*, vol. 38, no. 7, pp. 950-953, 1990.

[3] L.-G. Chen, W.-T. Chen, Y.-S. Jehng and T.-D. Chiueh, "An efficient parallel motion estimation algorithm for digital image processing," *IEEE Trans. Circuits and System for Video Tech.*, vol. 1, no. 4, pp. 378-385, 1991.

[4] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits and System for Video Tech.*, vol. 6, no. 3, pp. 313-317, 1996.

[5] J. Lu and M. L. Liou, "A simple and efficient search algorithm for block-matching motion estimation," *IEEE Trans. Circuits and System for Video Tech.*, vol. 7, no. 2, pp. 429-433, 1997.

[6] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 6, pp. 338-353, 1965.

[7] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller-Part I & Part II," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 2, pp. 404-435, 1990.

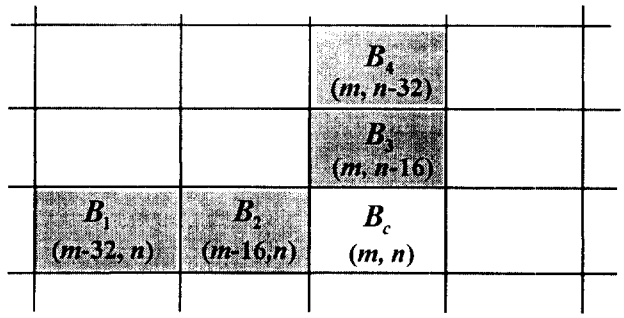


**Fig. 1. The position-relation of the current block and the reference blocks.**

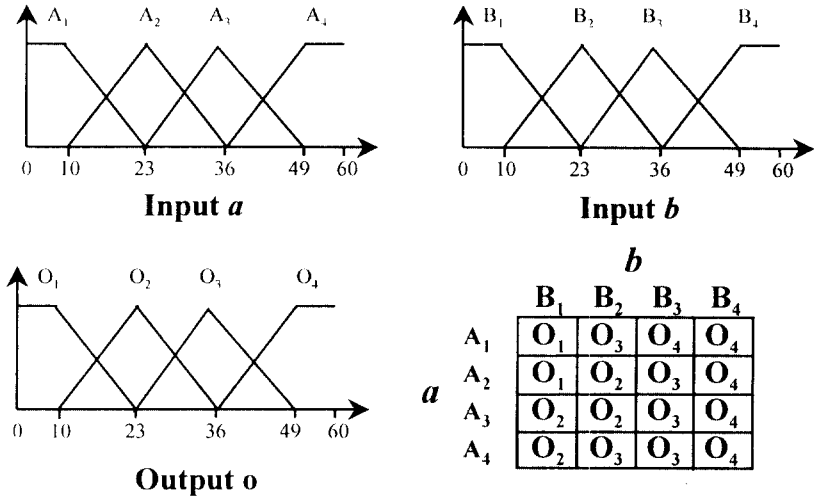


**Fig. 2. The corresponding membership functions and fuzzy control rules.**

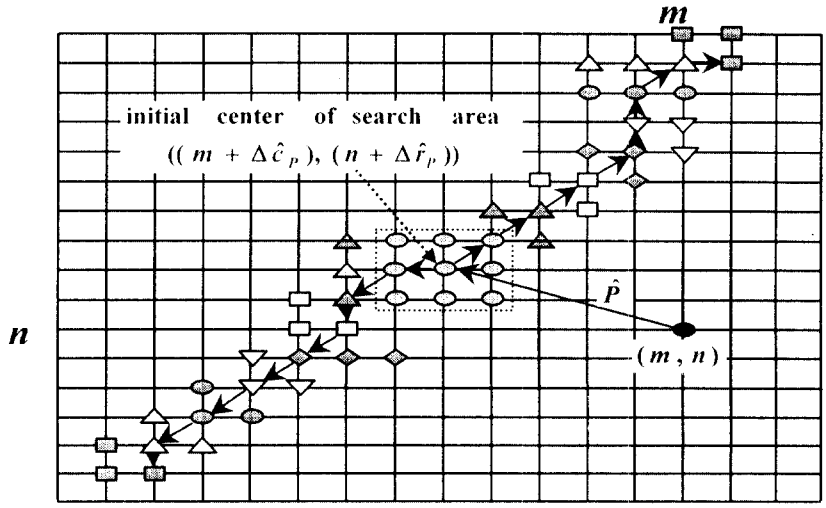


**Fig. 3. Two different search paths of the FPS.**

**TABLE I**

**Comparison of MSE of the search algorithms**

| | Football | Mobile | Windmill | Flower | Tennis | Salesman | Miss_A | Average | Percentage |
|---|---|---|---|---|---|---|---|---|---|
| FS | 360.93 | 357.59 | 350.35 | 268.85 | 135.14 | 28.71 | 16.31 | 216.84 | 100% |
| 3SS | 406.83 | 371.84 | 362.49 | 332.30 | 175.55 | 29.30 | 17.57 | 242.27 | 111.7% |
| CS | 458.62 | 387.51 | 597.43 | 742.30 | 219.65 | 29.34 | 18.70 | 350.51 | 161.6% |
| 4SS | 406.26 | 363.78 | 358.68 | 298.03 | 156.53 | 29.30 | 17.57 | 232.88 | 107.4% |
| SES | 463.02 | 371.42 | 362.91 | 331.40 | 196.29 | 29.32 | 17.72 | 253.15 | 116.7% |
| FPS(8) | 386.00 | 359.06 | 357.92 | 278.67 | 151.90 | 29.28 | 17.54 | 225.77 | 104.1% |
| FPS(10) | 381.74 | 359.06 | 357.80 | 278.66 | 149.48 | 29.28 | 17.54 | 224.79 | 103.7% |
| FPS(12) | 379.37 | 359.06 | 357.76 | 278.66 | 148.58 | 29.28 | 17.54 | 224.32 | 103.4% |

**TABLE II**

**Comparison of PSNR of the search algorithms**

| | Football | Mobile | Windmill | Flower | Tennis | Salesman | Miss_A | Average | Percentage |
|---|---|---|---|---|---|---|---|---|---|
| FS | 22.61 | 22.70 | 22.40 | 23.92 | 29.24 | 33.92 | 36.07 | 27.27 | 100% |
| 3SS | 22.09 | 22.51 | 22.65 | 22.98 | 28.07 | 33.86 | 35.80 | 26.85 | 98.5% |
| CS | 21.55 | 22.34 | 20.44 | 19.44 | 27.24 | 33.85 | 35.59 | 25.78 | 94.5% |
| 4SS | 22.09 | 22.62 | 22.70 | 23.46 | 28.55 | 33.86 | 35.80 | 27.01 | 99.1% |
| SES | 21.52 | 22.51 | 22.64 | 23.00 | 27.64 | 33.85 | 35.76 | 26.70 | 97.9% |
| FPS(8) | 22.31 | 22.68 | 22.71 | 23.76 | 28.73 | 33.86 | 35.80 | 27.12 | 99.4% |
| FPS(10) | 22.35 | 22.68 | 22.71 | 23.76 | 28.76 | 33.86 | 35.80 | 27.13 | 99.5% |
| FPS(12) | 22.38 | 22.68 | 22.71 | 23.76 | 28.78 | 33.86 | 35.80 | 27.14 | 99.5% |

**TABLE III**

**Comparison of prediction errors of the search algorithms**

| | Football | Mobile | Windmill | Flower | Tennis | Salesman | Miss_A | Average | Percentage |
|---|---|---|---|---|---|---|---|---|---|
| FS | 10.40 | 10.01 | 9.46 | 8.66 | 5.00 | 3.31 | 2.66 | 7.07 | 100% |
| 3SS | 11.00 | 10.19 | 9.64 | 9.87 | 5.77 | 3.31 | 2.69 | 7.50 | 106.1% |
| CS | 11.58 | 10.37 | 12.66 | 15.19 | 6.33 | 3.31 | 2.73 | 8.88 | 125.6% |
| 4SS | 10.93 | 10.09 | 9.57 | 9.32 | 5.35 | 3.31 | 2.69 | 7.32 | 103.5% |
| SES | 11.64 | 10.18 | 9.65 | 9.87 | 6.06 | 3.32 | 2.70 | 7.63 | 107.9% |
| FPS(8) | 10.61 | 10.01 | 9.55 | 8.77 | 5.21 | 3.31 | 2.69 | 7.16 | 101.3% |
| FPS(10) | 10.55 | 10.01 | 9.55 | 8.77 | 5.19 | 3.31 | 2.69 | 7.15 | 101.1% |
| FPS(12) | 10.52 | 10.01 | 9.55 | 8.77 | 5.17 | 3.31 | 2.69 | 7.15 | 101.1% |

TABLE IV

**Comparison of entropy of prediction errors of the search Algorithms**

| | Football | Mobile | Windmill | Flower | Tennis | Salesman | Miss_A | Average | Percentage |
|---|---|---|---|---|---|---|---|---|---|
| FS | 5.66 | 5.50 | 5.39 | 5.37 | 4.40 | 4.13 | 3.83 | 4.90 | 100% |
| 3SS | 5.73 | 5.53 | 5.42 | 5.57 | 4.57 | 4.13 | 3.84 | 4.97 | 101.4% |
| CS | 5.79 | 5.55 | 5.74 | 6.15 | 4.65 | 4.13 | 3.85 | 5.12 | 104.5% |
| 4SS | 5.71 | 5.51 | 5.40 | 5.49 | 4.47 | 4.13 | 3.84 | 4.94 | 100.8% |
| SES | 5.79 | 5.52 | 5.42 | 5.57 | 4.61 | 4.13 | 3.84 | 4.98 | 101.6% |
| FPS(8) | 5.68 | 5.50 | 5.40 | 5.38 | 4.44 | 4.13 | 3.84 | 4.91 | 100.2% |
| FPS(10) | 5.67 | 5.50 | 5.40 | 5.38 | 4.44 | 4.13 | 3.84 | 4.91 | 100.2% |
| FPS(12) | 5.67 | 5.50 | 5.40 | 5.38 | 4.43 | 4.13 | 3.84 | 4.91 | 100.2% |

TABLE V

**Comparison of percentage of unpredictable pels of the search algorithms**

| | Football | Mobile | Windmill | Flower | Tennis | Salesman | Miss_A | Average | Percentage |
|---|---|---|---|---|---|---|---|---|---|
| FS | 59.5% | 50.4% | 48.5% | 51.2% | 34.7% | 33.9% | 25.0% | 43.3% | 100% |
| 3SS | 60.4% | 50.8% | 49.1% | 55.0% | 37.9% | 33.9% | 25.1% | 44.6% | 103.0% |
| CS | 61.1% | 51.2% | 52.0% | 62.0% | 39.4% | 33.9% | 25.3% | 46.4% | 107.2% |
| 4SS | 59.8% | 50.6% | 48.6% | 53.7% | 36.0% | 33.9% | 25.1% | 44.0% | 101.6% |
| SES | 61.0% | 50.8% | 49.1% | 55.0% | 38.8% | 33.9% | 25.2% | 44.8% | 103.5% |
| FPS(8) | 59.6% | 50.4% | 48.5% | 51.3% | 35.2% | 33.9% | 25.1% | 43.4% | 100.2% |
| FPS(10) | 59.5% | 50.4% | 48.5% | 51.3% | 35.1% | 33.9% | 25.1% | 43.4% | 100.2% |
| FPS(12) | 59.5% | 50.4% | 48.5% | 51.3% | 35.1% | 33.9% | 25.1% | 43.3% | 100.1% |

TABLE VI

**Comparison of search points of the search algorithms**

| | Football | Mobile | Windmill | Flower | Tennis | Salesman | Miss_A | Average | Percentage |
|---|---|---|---|---|---|---|---|---|---|
| FS | 66676 | 66676 | 66676 | 66676 | 66676 | 66676 | 66676 | 66676 | 100% |
| 3SS | 7632 | 7599 | 7646 | 7662 | 7627 | 7596 | 7608 | 7624 | 11.4% |
| CS | 5097 | 5090 | 5135 | 5113 | 5111 | 5104 | 5106 | 5108 | 7.7% |
| 4SS | 6037 | 5227 | 5701 | 6265 | 5873 | 5178 | 5340 | 5660 | 8.5% |
| SES | 5357 | 5572 | 5369 | 5238 | 5360 | 5594 | 5501 | 5427 | 8.1% |
| FPS(8) | 4558 | 3049 | 3695 | 3969 | 3766 | 2798 | 3167 | 3572 | 5.4% |
| FPS(10) | 4631 | 3049 | 3697 | 3970 | 3801 | 2798 | 3167 | 3587 | 5.4% |
| FPS(12) | 4674 | 3049 | 3698 | 3971 | 3808 | 2798 | 3167 | 3595 | 5.4% |