DOUBLE-SEARCH-WINDOW BLOCK MATCHING USING THE FAST FOURIER TRANSFORM

Zhen LI and Hitoshi KIYA

Dept. of Information and Communication Systems, Tokyo Metropolitan University
6–6 Asahigaoka Hino-shi, 191–0065, Japan

ABSTRACT

We propose double-search-window block matching and describe an FFT-based fast algorithm for it. The double-search-window defines two different-sized search windows, an inside-search-window and an outside-search-window, for block-matching. The proposed method achieves the same accuracy as a direct SSD full search in the inside-search-window. It also offers a pseudo-full-search result in the outside-search-window. By evaluating the results comprehensively, the new approach could achieve more precise motion vectors as well as generate the predicted pictures with fewer matching errors, but with almost the same computational load as in the conventional single-search-window block matching.

Index Terms— FFT, block matching, motion estimation, motion vector, double-search-window

1. INTRODUCTION

Block matching is widely used in many fields, including pattern recognition, computer vision and video coding. We could generally enlarge the search window to improve accuracy, but the computational load would simultaneously become heavier. Therefore, many fast block matching algorithms (BMAs) have been developed to achieve better accuracy under lighter computational load. Their basic approaches can be generally divided into three types.

The first uses an approximative search window instead of a full-search window. While the computational load is greatly decreased, the accuracy is less than that of a full-search BMA, and the initial value greatly affects the results. For example, the three-step search [1] and diamond search [2] are based on this approach. The second type has the same performance as a full-search BMA in terms of accuracy, but imposes a lighter computational load. The successive elimination algorithm (SEA) [3, 4] is a representative of this type. However, the degree to which the computational load can be reduced depends on the input signal.

These first two types operate in the spatial domain. The third shifts the spatial domain problem into the frequency domain by using the Fast Fourier Transform(FFT) to calculate the cross-correlation. Compared to the full-search BMA, the third type [5, 6, 7, 8] achieves the same accuracy and greatly decreases the computational load as it does not depend on the input signal.

We propose an FFT-based block matching algorithm with a double-search-window (DSW) based on the third type. The DSW defines two different-sized search windows, an inside-search-window and an outside-search-window. The inside-search-window is exactly the same as the conventional search window. The outside-search-window is filled with assumed signals, e.g., the periodic part of the inside-search-window if we use the FFT-based algorithm. Therefore, it achieves at least the same result as a direct full search in the inside-search-window and improves this result by offering a pseudo-full-search in the outside-search-window. The proposed DSW BMA can simultaneously search both windows and the computational load is almost the same as that with conventional single-search-window methods.

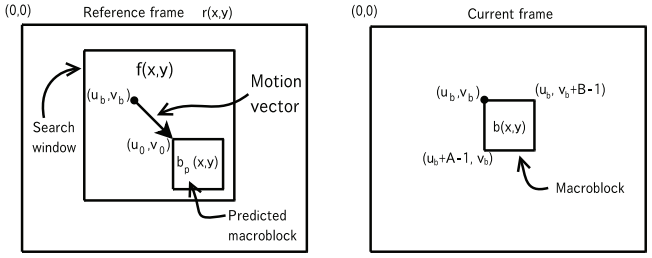


Fig. 1. Block matching with single-search-window.

2. PREPARATION

Let us begin by defining block matching and then providing a simple explanation of the FFT-based full-search BMA. In this paper, we have only focused on integral pixel block matching. Below, let $\mathbf{Z}$ denote the set of integer numbers.

2.1. Block matching

As shown in Fig. 1, let 2-D signal $b(x, y)$ in the current frame be a macroblock whose size is $A \times B$, and let 2-D signal $f(x, y)$ in the reference frame, $r(x, y)$, be a search window whose size is $M \times N$. (u_b, v_b) is the coordinate of the top-left point of $b(x, y)$ in $r(x, y)$. Below, let $A < M$, $B < N$, and $A, B, M, N \in \mathbf{Z}$.

Inside the search window, there are $(N - B) \times (M - A)$ different macroblocks, which have the same size as $b(x, y)$. All these different macroblocks are compared with $b(x, y)$ to find the one most similar to it (that with minimum matching error). This procedure can be defined as "full-search block matching." One of the matching criteria is the sum of squared differences (SSD) as

$$\mathrm{SSD}_{b,f}(u, v) = \sum_{x=0}^{A-1} \sum_{y=0}^{B-1} \{f(x + u, y + v) - b(x, y)\}^2, \quad (1)$$

$$(u, v) \in R_{IN}, \quad u, v \in \mathbf{Z} \quad \text{and}$$

$$R_{IN} = \{(u, v) | 0 \leqslant u \leqslant M - A \text{ and } 0 \leqslant v \leqslant N - B\}. \quad (2)$$

The u and v are the shift amounts whose range is R_{IN}. The purpose of block matching is to find (u_0, v_0) that yields the minimum matching

criterion:

$$\mathrm{SSD}_{b,f}(u_0, v_0) = \min_{(u,v)\in R_{IN}} \{\mathrm{SSD}_{b,f}(u, v)\}. \quad (3)$$

In order to compare it with the proposed method we refer to it as single-search-window block matching.

2.2. Motion Vector and Predicted Macroblock

The two main purposes of block matching are to determine the motion vector and the predicted macroblock. As shown in Fig. 1, if (u_0, v_0) is the result of block matching, the motion vector, **MV**, is defined as

$$\mathbf{MV} = \begin{pmatrix} v_0 - v_b \\ u_b - u_0 \end{pmatrix}. \quad (4)$$

At the same time, the predicted macroblock, $b_p(x, y)$, is defined as

$$b_p(x, y) = f(x + u_0, y + v_0), \quad x = 0, 1, \ldots, A-1, \quad y = 0, 1, \ldots, B-1. \quad (5)$$

2.3. FFT-Based Block Matching Algorithm

First, by zero-padding $b(x, y)$, we can have signal $g_b(x, y)$ which has the same size as $f(x, y)$. Then, we can rewrite Eqs. (1) and (3) as

$$\min_{(u,v)\in R_{IN}} \{\mathrm{SSD}_{b,f}(u, v)\} = C_{g_b} - \max_{(u,v)\in R_{IN}} \{2cor_{g_b,f}(u, v) - S_{f^2}(u, v)\}, \quad (6)$$

where

$$C_{g_b} = \sum_{y=0}^{B-1}\sum_{x=0}^{A-1}\{b(x, y)\}^2, \quad (7)$$

$$cor_{g_b,f}(u, v) = \sum_{y=0}^{N-1}\sum_{x=0}^{M-1} g_b(x, y) f(x + u, y + v), \quad \text{and} \quad (u, v) \in R_{IN} \quad (8)$$

$$S_{f^2}(u, v) = \sum_{y=0}^{B-1}\sum_{x=0}^{A-1}\{f(x + u, y + v)\}^2. \quad (9)$$

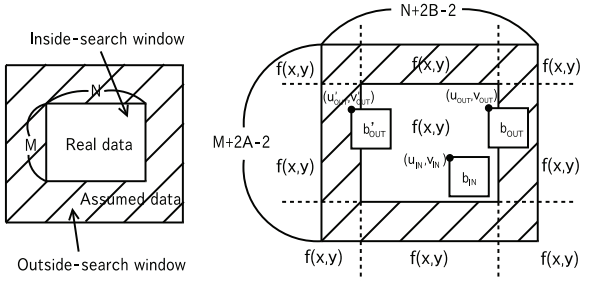
The first term, C_{g_b}, on the right-hand side of Eq. (6), is independent of the shift amounts (u, v), which means it would not affect the result of block matching. The second term is the circular cross-correlation between $g_b(x, y)$ and $f(x, y)$, which can easily be calculated by using the FFT. The third term, $S_{f^2}(u, v)$, can also be efficiently calculated as the recursive summation. In addition, all the FFT-based fast BMAs [5, 6, 7, 8] are available for a double-search-window.

3. BLOCK MATCHING WITH DOUBLE-SEARCH-WINDOW

3.1. Double-Search-Window

The range of shift amounts (u, v) is $(M-A)\times(N-B)$ in conventional block matching algorithms with the single-search-window, $f(x, y)$. This is reflected in Eq. (8) as R_{IN}. Let us refer to this signal-search-window as the inside-search-window.

Apart from the inside-search-window, a double-search-window (DSW) defines a larger search area, which is called an outside-search-window (shown in Fig. 2 (a)). Different from the inside-search-window, the outside-search-window is formed by using assumed data not the real data from the reference frame, $r(x, y)$. For example in Fig. 2 (b), the FFT-based DSW uses the periodic part of inside-search-window $f(x, y)$ to form a $(M + 2A - 2) \times (N + 2B - 2)$ outside-search-window.



(a) Conceptual diagram for double-search window (b) Double-search-window using periodic data

Fig. 2. Double-search-window.

3.2. FFT-based DSW

The FFT-based BMA calculates the non-circular cross-correlation, $cor_{g_b,f}(u, v)$, in Eq. (8) by using part of $\widehat{cor}_{\hat{g}_b,\hat{f}}(u, v)$, which is the cross-correlation between two periodic signals, $\hat{g}_b(x, y)$ and $\hat{f}(x, y)$, as

$$\widehat{cor}_{\hat{g}_b,\hat{f}}(u, v) = \sum_{y=0}^{N-1}\sum_{x=0}^{M-1} \hat{g}_b(x, y)\hat{f}(x + u, y + v), \quad (u, v) \in R_{OUT}, \quad (10)$$

where

$$R_{OUT} = \{(u, v) | 0 \leqslant u \leqslant M - 1 \text{ and } 0 \leqslant v \leqslant N - 1\}. \quad (11)$$

The period of two signals, $\hat{g}_b(x, y)$ and $\hat{f}(x, y)$, is $M \times N$. The single-search-window block matching in Eq. (1) only uses part of $\widehat{cor}_{\hat{g}_b,\hat{f}}(u, v)$, as

$$cor_{g_b,f}(u, v) = \widehat{cor}_{\hat{g}_b,\hat{f}}(u, v), \quad (u, v) \in R_{IN}. \quad (12)$$

However, if we use all of it, we can achieve double-search-window block matching,

$$\mathrm{SSD}_{b,\hat{f}}(u, v) = \sum_{x=0}^{A-1}\sum_{y=0}^{B-1}\{\hat{f}(x + u, y + v) - b(x, y)\}^2, \quad (u, v) \in R_{OUT}. \quad (13)$$

There will be two block matching results for Eqs. (1) and (13), as shown in Fig. 2 (b). The first, as b_{IN}, is in the inside-search-window,

$$\mathrm{SSD}_{b,\hat{f}}(u_{IN}, v_{IN}) = \min_{(u,v)\in R_{IN}} \{\mathrm{SSD}_{b,\hat{f}}(u, v)\}. \quad (14)$$

The second, as b_{OUT}, is in the out-search-window,

$$\mathrm{SSD}_{b,\hat{f}}(u_{OUT}, v_{OUT}) = \min_{(u,v)\in R_{OUT}} \{\mathrm{SSD}_{b,\hat{f}}(u, v)\}. \quad (15)$$

Note that because of periodicity, block b_{out} and block b'_{out} in Fig. 2 have the same matching error,

$$\mathrm{SSD}_{b,\hat{f}}(u_{OUT}, v_{OUT}) = \mathrm{SSD}_{b,\hat{f}}(u'_{OUT}, v'_{OUT}), \quad (16)$$

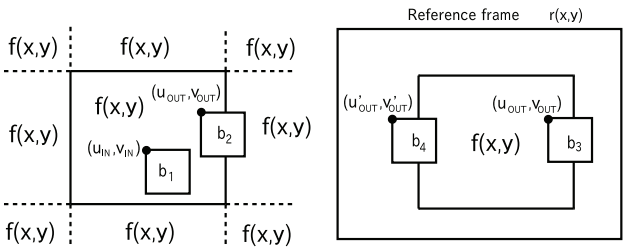


Fig. 3. Three coordinates and four candidates for predicted macroblock.

where

$$u_{OUT} = u'_{OUT} + k_1 M, \quad v_{OUT} = v'_{OUT} + k_2 N, \quad k_1, k_2 \in \mathbf{Z}. \tag{17}$$

3.3. Motion Vector and Predicted Macroblock in DSW

As shown in Fig. 3, there are three coordinates, (u_{IN}, v_{IN}), (u_{OUT}, v_{OUT}), and (u'_{OUT}, v'_{OUT}), to be found in the DSW block matching. There are also four candidates for predicted macroblock, which are b_1, b_2, b_3, and b_4.

Macroblock b_1 is the result of an inside-search-window full search. Part of macroblock b_2, b_3, or b_4 is beyond the inside-search-window. Also, b_2 is taken from the periodic part of inside-search-window $f(x, y)$. b_3 is taken from the real data of reference frame $r(x, y)$. b_4 is also taken from $r(x, y)$ but its coordinate is in the periodic position to b_3. We can then use these three coordinates and four candidates to determine the motion vector and the predicted macroblock.

A. IF $(u_{IN}, v_{IN}) = (u_{OUT}, v_{OUT})$

Then, the following equation holds

$$\mathrm{SSD}_D(u_0, v_0) = \mathrm{SSD}_{b,\hat{f}}(u_{IN}, v_{IN}) = \mathrm{SSD}_{b,\hat{f}}(u_{OUT}, v_{OUT}). \tag{18}$$

Where $\mathrm{SSD}_D(u_0, v_0)$ means the minimum SSD in the DSW, whose coordinate is (u_0, v_0). Therefore, we can determine the motion vector and the predicted macroblock from Eqs. (4) and (5). They are the same as those in single-search-window block matching.

B. IF $(u_{IN}, v_{IN}) \neq (u_{OUT}, v_{OUT})$

First, we choose one coordinate from b_1, b_3, and b_4, for the motion vector, which has the minimum matching error.

$$\mathrm{SSD}_D(u_0, v_0) = \min\{\mathrm{SSD}_{b,\hat{f}}(u_{IN}, v_{IN}), \mathrm{SSD}_{b,r}(u_{OUT}, v_{OUT}), \mathrm{SSD}_{b,r}(u'_{OUT}, v'_{OUT})\}, \tag{19}$$

where

$$\mathrm{SSD}_{b,r}(u_0, v_0) = \sum_{x=0}^{A-1} \sum_{y=0}^{B-1} \{r(x + u_0, y + v_0) - b(x, y)\}^2. \tag{20}$$

Then, according to Eq. (4), we can determine the motion vector using (u_0, v_0).

Table 1. Comparison of computational load for one block matching ($M = N$,$A = B = N/2$).

Direct SSD full search

Search window size	Addition	Multiplication
Inside-search-window	$N^4/8$	$N^4/16$
Outside-search-window	$9N^4/8$	$9N^4/16$

FFT-based BMA

Search window size	Addition	Multiplication
Inside-search-window	$(12\log_2 N + 9/4)N^2$	$(12\log_2 N - 7)N^2$
Proposed DSW	$(12\log_2 N + 5)N^2$	$(12\log_2 N - 7)N^2$
Outside-search-window	$(48\log_2 N + 80)N^2$	$(48\log_2 N + 56)N^2$

For the predicted macroblock on the other hand, one block from b_2, b_3, and b_4, which has the minimum matching error will be chosen.

$$\mathrm{SSD}_D(u_0, v_0) = \min\{\mathrm{SSD}_{b,\hat{f}}(u_{OUT}, v_{OUT}), \mathrm{SSD}_{b,r}(u_{OUT}, v_{OUT}), \mathrm{SSD}_{b,r}(u'_{OUT}, v'_{OUT})\}. \tag{21}$$

3.4. Computational Load of DSW

Table 1 lists the computational load of a direct SSD full search and the FFT-based BMA in different search windows. The size of the inside-search-window is $M \times N$, that of the macroblock is $A \times B$, and that of the outside-search-window is $(M+2A-2)\times(N+2B-2)$. For the simplicity, let $M = N$ and $A = B = N/2$. Let the ratio between complex addition and real addition be $2 : 1$ and the ratio between complex multiplication and real multiplication be $4 : 1$.

We can see that the computational load of FFT-based BMA is less than the direct SSD full search with the same search window size. The computational load of FFT-based BMA with DSW is almost the same as it with the inside-search-window.

4. SIMULATION

We simulated motion estimation by using 0–9 frames of the video sequence "caltrain" (size: 400×512) and 1–12 frames of the video sequence "garden" (size: 240×352). The macroblock size was fixed at 16×16. There were four search window sizes: 32×32 (±8 pixels around the macroblock), the proposed $32 - 62$ DSW (the inside-search-window was 32×32 and the outside-search-window was 62×62), 48×48 (±16 pixels around the macroblock), and 62×62 (±23 pixels around the macroblock). The experiments were carried out using Matlab 6.5 on a computer with an Intel Core2 2.4-GHz CPU and 2-GB RAM.

4.1. Average Motion Vector Distance per Macroblock

Fig. 4 shows the average motion vector distance per macroblock. For every macroblock b_i, there are three motion vectors $\mathbf{MV62}_i$, $\mathbf{MV32}_i$, and $\mathbf{MVD32}_i$ because three different BMAs are used, which correspond to a 32×32 full search, a $32-62$ DSW, and a 62×62 full search. Let $\mathbf{MV62}_i$ be the correct answer, and the distance between $\mathbf{MV32}_i$ or $\mathbf{MVD32}_i$ be the motion vector distance for the macroblock. That is,

$$D32_{b_i} = \| \mathbf{MV62}_i - \mathbf{MV32}_i \|_2 \quad \text{and} \tag{22}$$

$$DD32_{b_i} = \| \mathbf{MV62}_i - \mathbf{MVD32}_i \|_2 \, . \tag{23}$$

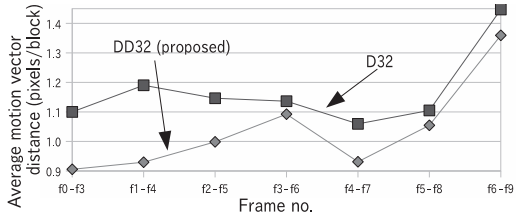


Fig. 4. Average motion vector distance per macroblock for one frame ("caltrain") ($f_i - f_j$: f_i is reference frame number and f_j is current frame number).

Table 2. Execution time for generating one predicted frame.

Direct SSD full search

	Execution time(s)	
Search window size(pixels)	"caltrain"	"garden"
32×32	4.52	1.54
48×48	16.94	5.71
62×62	34.18	11.47

FFT-based BMA

	Execution time(s)	
Search window size (pixels)	"caltrain"	"garden"
32×32	0.22	0.08
$32 - 62$DSW (proposed)	0.23	0.09
48×48	0.42	0.16
62×62	1.88	0.63

The average motion vector distance per macroblock for one frame is defined as

$$D32 = \{\sum_i D32_{b_i}\}/N_b \quad \text{and} \tag{24}$$

$$DD32 = \{\sum_i DD32_{b_i}\}/N_b, \tag{25}$$

where N_b means the total number of macroblocks per frame. We only use macroblocks that have the real data around them for ± 23 pixels. The same macroblocks were used in the execution time and PSNR estimation. We can see from the result that the proposed method was better than the 32×32 full search.

4.2. Execution time and PSNR

Table 2 lists the execution time for generating one predicted frame using different block matching methods. Figs. 5 and 6 plot the PSNR between the predicted frame and the current frame for different methods. We can see that the execution time increased as the search window enlarged. Compared to the direct SSD full search, the FFT-based BMA reduced the execution time to $1/18 - 1/40$ according to the size of the search windows. The execution time of the proposed $32 - 62$ DSW was almost the same as that of 32×32 full search.

From Figs. 5 and 6, the PSNR of the proposed $32 - 62$ DSW was absolutely better than that of the 32×32 full search and it was near that of the 48×48 full search.

5. CONCLUSIONS

We proposed double-search-window block matching and described an FFT-based fast algorithm for it. The computational load of DSW

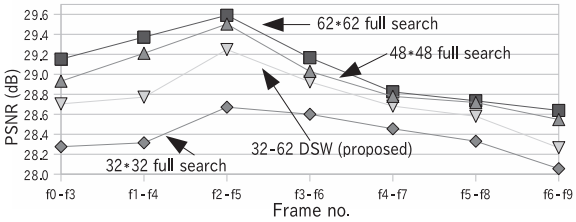


Fig. 5. PSNR of predicted frame ("caltrain").

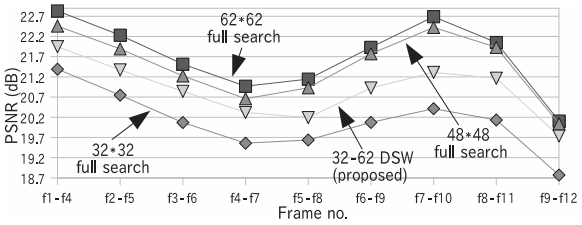


Fig. 6. PSNR of predicted frame ("garden").

was almost the same as that for an inside-search-window full search but DSW offered a better block matching result. The simulation results confirmed that the proposed method used less execution time and achieved a better PSNR and more precise motion vectors. Therefore, DSW is considered useful and effective for motion estimation, motion vector detection, and many other block matching applications.

6. REFERENCES

[1] T. Koga, K. Iinuma, A. Hirano, and Y. Iijima, "Motion compensated interframe coding for video conferencing," in *Proc. IEEE NTC Conf.*, pp. G5.3.1–G5.3.5, 1981.

[2] S. Zhu and K. Ma, "A new diamond search algorithm for fast block motion estimation," *IEEE Trans. Image Process.*, vol.9, no.2, pp.287–290, 2000.

[3] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Trans. Image Proc.,* vol.4, no.1, pp. 105–107, Jan. 1995.

[4] Y. Noguchi, J. Furukawa, and H. Kiya, "A fast full search block matching algorithm for MPEG-4 video," in *Proc. IEEE Int. Conf. Image Proc.*, vol.1, pp.61–65, 1999.

[5] Steven L. Kilthau, Mark S. Drew, and Torsten Moller, "Full Search Content Independent Block Matching Based On The Fast Fourier Transform," *IEEE Int. Conf. Image Process.,* vol.1, pp.669–672, New York, Sept. 2002.

[6] A. Fitch, A. Kadyrov, W. Christmas, and J. Kittler, "Fast robust correlation," *IEEE Trans. Image Process.*, vol.14, no.8, pp.1063–1073, 2005.

[7] F. Essannouni, R. Oulad Haj Thami, D. Aboutajdine, and A. Salam, "Simple noncircular correlation method for exhaustive sum square difference matching," *Optical Engineering,* vol.46, no.10, 107004, Oct. 2007.

[8] Z. Li, A. Uemura, and H. Kiya,"An FFT-based full-search block-matching algorithm with SSD criterion," *Int. Conf. APSIPA ASC*, pp.457–460, Oct. 2009.