Fast Binary Block Matching Motion Estimation using Efficient One-Bit Transform

Ye-Kui Wang and Guo-Fang Tu

***Abstract*—Binary block matching algorithm employs one-bit transform (1BT) to transform video sequences from full resolution representation (8-bit/pixel in general) to 1-bit/pixel bit-plane, then performs block matching motion estimation in bit-plane to save computations. Unfortunately, current 1BT algorithms either have bad output motion vectors or are computational intensive. In this paper, a novel fast and efficient 1BT algorithm called *the overlap-windowed thresholding algorithm (OWT)*, and a new *fast binary block matching algorithm (FBBMA)* based on OWT are proposed. Besides OWT, there are three other novelties in FBBMA, namely *fast binary block matching*, *adaptive central-search-point prediction* and *center-biased search order*, in which significant modifications are contained. Four more existing techniques are employed in FBBMA to further improve performance. Experimental results show the superiorities of all the proposed algorithms and modifications. With similar quality of predicted sequence, the proposed OWT algorithm performs 37 times faster than the filter thresholding method, which gives the best quality among current 1BTs. FBBMA significantly outperforms over several other fast motion estimation algorithms, including the diamond search algorithm that was recently adopted in MPEG-4, in terms of both speed up ratio and quality of predicted sequence. In addition, the output motion vectors of FBBMA need fewer bits to be encoded than most if not all other ME algorithms.**

***Index Terms*—Binary block matching motion estimation, fast motion estimation, video compression, one-bit transform.**

I. Introduction

Motion estimation (ME) and compensation is efficient in eliminating temporal redundancy in video sources, and has become a central component of most video coding standards such as ISO MPEG-1/2/4 [1]-[3] and ITU-T H.261/263/26L [4]-[6]. Among the existing ME algorithms, the block matching algorithm (BMA) [7] is most widely used for its simplicity and high performance. The full search (FS) BMA provides an optimal solution in the sense of minimizing mean absolute difference (MAD) or sum of absolute difference (SAD) when MAD or SAD is used as the block distortion measure (BDM). However, FS is computational too costly for practical real-time applications and cannot produce in general the optimal bit rate, especially for very low-bit-rate video coding, which uses a significant portion of bits for motion vector (MV) encoding [8]. To resolve these problems, many fast search algorithms [9]-[15] and rate-distortion optimized algorithms [8], [16]-[18] were developed.

The fast search algorithms reduce computational complexity either by reducing the number of test positions [9]-[14] or by reducing the computational cost of BDM for each position [15], [19]-[31]. The binary block matching

algorithms (BBMAs) [20]-[21] based on one-bit transform (1BT) belong to the latter. BBMA first uses 1BT to transform the luminance component of video sequences from full resolution representation (8-bit/pixel in general) to 1-bit/pixel bit-plane. Then a different BDM, called the number of not matching points (NNMP), is used instead of SAD in the matching process. The SAD and NNMP of position *(m,n)* in blocks of size *NxN* are given as:

$$SAD(x,y) = \sum_{m,n=0}^{N-1} \left| I_t(m,n) - I_{t-i}(m+x, n+y) \right| \tag{1}$$

$$NNMP(x,y) = \sum_{m,n=0}^{N-1} B_t(m,n) \otimes B_{t-i}(m+x, n+y) \tag{2}$$

where $I_t(m,n)$ and $B_t(m,n)$ are the luminance intensity of the 8-bit/pixel representation and the binary value of one-bit transformed bit-plane in the current frame, $I_{t-i}(m+x,n+y)$ and $B_{t-i}(m+x,n+y)$ are those values in a previous frame, and $\otimes$ denotes the exclusive-or operation. By changing the subtraction and absolute operations to exclusive-or operations in calculating BDM for each test position, BBMA can achieve great computation reduction. 1BT plays a key role in BBMA. However, current 1BT algorithms either result in bad motion vectors with low quality of predicted sequence, or are computational intensive. Thus calls upon new fast and efficient 1BT algorithm.

Rate-distortion optimized ME algorithms try to obtain better rate-distortion performance in video coding by considering simultaneously block distortion and the number of bits for MV encoding. Usually they combine the block distortion and the bit-rate by a Lagrangian multiplier as the BDM in motion search, thus results in intensive computation.

In this paper, a thorough study of BBMA and 1BT is first presented. The drawbacks of existing 1BT methods are analyzed. To overcome the drawbacks, a novel fast and efficient 1BT algorithm called *the overlap-windowed thresholding algorithm (OWT)* is then proposed. To further improve the search speed and the prediction performance of the BBMA based on OWT, seven other techniques are employed. Thus the novel fast ME algorithm named *fast binary block matching algorithm (FBBMA)* is obtained. These improving techniques are: 1) *fast binary block matching*; 2) *adaptive central-search-point prediction*; 3) *center-biased search order*; 4) *still macroblock detecting*; 5) *halfway stop in block distortion computation*; 6) *variable search distance*; and 7) *multiple candidates selection*. The first three of them contain significant modifications made by the authors, and are proposed literarily here in the first time. The later four are existing techniques.

Among the seven techniques, the 1st, 4th, 5th and 6th can improve the search speed, the 7th can improve the quality of predicted sequence, and the 2nd can improve both. The 3rd, *center-biased search order*, can improve not only both of the search speed and the prediction performance, but also the rate-distortion performance. It produces

more correlative MVs that need fewer bits to be encoded, therefore more bits can be used for the predicted error and better rate-distortion performance can be acquired. What's more, the computational complexity of *center-biased search order* is much lower than the conventional rate-distortion optimized ME algorithms.

Our experimental results show the superiorities of all the proposed algorithms and modifications, including the modifications in the improving techniques.

The structure of this paper is organized as follows: Section II presents the overview of BBMA and 1BT. Then OWT is described in Section III. In Section IV, the FBBMA scheme and the seven improving techniques are presented. Experimental results of different 1BT methods, the modifications in the improving techniques, and various ME algorithms are provided in Section V. Finally, Section VI draws the conclusion.

II. Overview Of BBMA And 1BT

The concept of binary bit-plane matching is first proposed by Lee [22]. He exploits the bit-plane combined with the block/sub-block means as the matching criterion in a hierarchical motion search scheme. Later, he and his colleagues expanded the work extensively [23]-[27]. In [20] and [28], Feng et al exploit bit-plane matching as a preprocessing step to sort out the improbable locations where NNMPs are larger than a predefined threshold, and then the conventional FS is applied to the remained locations. For each block, the 8-bit luminance pixel data $\boldsymbol{I}$ (of either the current or the reference frame) is transformed into bit-plane $\boldsymbol{B}$ as

$$B(i,j)=\begin{cases}1, & if\ I(i,j)\geq bm\\ 0, & otherwise.\end{cases} \tag{3}$$

where *bm* is the block mean. We called this straightforward transforming strategy as the block mean thresholding (BMT) technique.

In [29], Mizuki et al use the binary edge maps as the bit-plane. The bit-plane is given by

$$B(i,j)=\begin{cases}1, & if\ I(i,j)\ is\ an\ edge\ pixel\\ 0, & otherwise.\end{cases} \tag{4}$$

Although this edge-detection based (EDB) 1BT can sometimes improve the estimating accuracy of moving object boundaries [30], it cannot be applied to the blocks where the number of edge pixels is small [29]-[30]. In addition, in the following case, EDB will give a wrong matching result. Assume there are two blocks of *4x4* as shown in Fig. 1(a) and (b). If EDB is used to transform them into bit-planes, the transforming results will be identical, as shown in Fig. 1(c). Therefor NNMP between the two blocks is 0 and will result in a wrong match. In the scenario of BBMA, the ideal one-bit transformed results for Fig. 1(a) and (b) should be as shown in Fig. 1 (d) and (e), respectively. The objective is to protect not only the edge information, but also the smoothness of smooth areas. Fortunately, BMT [20],

FT [21] and the proposed OWT can fit this requirement.

In FT, each 8-bit frame $\boldsymbol{I}$ is filtered with a *17x17* kernel $\boldsymbol{K}$ as given in Eqn. 5, to obtain the filtered frame I'. The bit-plane is then given by Eqn. 6.

$$K(i,j)=\begin{cases}1/25, & if\ i,j\in[0,4,8,12,16]\\ 0, & otherwise.\end{cases} \tag{5}$$

$$B(i,j)=\begin{cases}1, & if\ I(i,j)\geq I'(i,j)\\ 0, & otherwise.\end{cases} \tag{6}$$

From the frequency response of $\boldsymbol{K}$ (as shown in Fig. 2), we find that $\boldsymbol{K}$ is a multi-band-pass filter rather than a band-pass filter as stated in [21] and [31]. The low-frequency signal component is also reserved besides the edge information; thus the smoothness of smooth areas is also protected.

It should be pointed out that there is a wrong opinion in the field of BBMA on [32]. In [21] and [27], [32] is cited and said that binary matching is applied there. As a matter of fact, the strategy used in [32] is as follows: the pixels in current block are compared with the corresponding pixels in the shifted block of the reference frame, and classified as matching or mismatching pixels according to the difference between the two luminance values; then the position with the largest number of matching pixels is taken as the best match. We can see that it does not use bit-plane to represent multi-bit data, and no binary matching between bit-planes is carried out.

The size of the transforming block or the filter kernel $\boldsymbol{K}$ in BMT or FT is an important parameter that imposes great influence upon BBMA performance. *17x17* is selected for $\boldsymbol{K}$ in [21] because the author[1] thought that a smaller kernel might be too sensitive to noise in the video frames whereas too large a kernel would give a poor estimate of the thresholds to be used in 1BT. The additional reason been said was that inter-correlation among pixels falls off rapidly and for a pixel distance greater than 8 the correlation was very poor, so using a neighborhood of *17x17* pretty much was the limit of the good correlation neighborhood. We have tested other $\boldsymbol{K}$ sizes ranging from *7x7* to 41*x*41, finding that *17x17* is almost the best for several standard test sequences. However, it is not true for BMT that 8 is the best distance regarding the pixel correlation, as can be seen from Fig. 3.

Fig. 3 gives the peak-peak signal-to-noise ratio (PSNR) results of different transforming block sizes ranging from 4*x*4 to 100*x*100 in BMT. The PSNR is averaged from the predicted frames of the sequence Miss America (CIF, 10 frame/sec). The original previous frame is taken as the reference frame. Full search BBMA is performed with the MV range [*-7, 7*]. The block size for ME is *16x16*. Three typical bit-planes of the first frame (only the central part) are shown in Fig.4 (a)-(c), where a binary value 1/0 is represented as a white/black point.

[1] The authors have discussed the question with Dr. Vasudev Bhaskaran, one of the authors of [21], through e-mails.

It is seen from Fig. 3 that PSNR of BMT improves rapidly when the block size increases from a small value. However, when the block size continues to increase after a moderate value (about 20), the performance is inclined to degrade. The explanation for this phenomenon can be partly if not all found from Fig. 4. If the transforming block is too small such as of *4x4*, the blocky effect (the many horizontal and vertical pseudo edges in Fig. 4(a)) is so serious that the overall contours are almost submerged by the artifacts. On the contrary, if the transforming block is too large such as of *352x288*, the blocky effect is removed, however at this time many important details are lost, as shown in Fig. 4(c). In both the above cases, BBMA is apt to be trapped into a non-optimal location. If a moderate block size is applied, there will be a compromise between the blocky effect and the detail information, thus results in a better performance. However, the blocky effect is still apparent, as can be seen in Fig. 4(b). A question arises: is there any 1BT method that can simultaneously preserve detail information and remove the blocky effect?

FT happens to be a good candidate. However, filtering operation is required in FT, which makes FT computational too intensive, especially in the software implementation. We have implemented FT in software using the library of Fastest Fourier Transform in the West (FFTW) [33], while the computational cost of FT is still 52 times higher than BMT. If other methods such as convolution method are used, the computational cost of FT will be higher. Therefore, a fast and efficient 1BT method becomes imperative for BBMA.

In the next section, we will provide a solution by presenting OWT, which can do as well as and sometimes even better than FT in terms of predicting efficiency, with much lower computational complexity.

III. Overlap-Windowed Thresholding 1bt

A deduction can be derived from Fig. 4(a)-(c) that: if enough detail information is to be preserved, BMT should be performed by a small block wise; if the blocky effect is to be removed, the transforming thresholds should be calculated based on a large block. Then, what if we calculate the threshold based on a large window while using a small transforming block? This is the idea where OWT comes from. Fig. 4(d) is the resulting bit-plane of OWT with the transforming block of *4x4* and the threshold window of *22x22*. It is seen that the blocky effect is well removed with plenty of detail information.

A. The Algorithm

The luminance frame is transformed block by block. For each $n \times n$ transforming block, we first calculate the average luminance value of all the pixels within the threshold window, which covers the transforming block, as shown in Fig. 5. The block locates in the center of the window, which is of $m \times m$ ($m=n+2w$) and moves following with the block. The bit-plane is given by

$$B(i,j)=\begin{cases}1, & if\ I(i,j)\geq wm\\ 0, & otherwise.\end{cases} \tag{7}$$

where *wm* is the mean of the threshold window.

B. The Values of Parameters w And *n*

Similar as the transforming block size that is important in BMT, *m* (or *w*) and *n* are important in OWT. For the binary ME purpose, there must exist a best combination of *m* and *n*. Here, both *w* and *n* are confined to be integer times of 4 for fast calculation of the window mean. The values are obtained is as follows:

First to find a proper value for *n*. Let *w* to be a moderate value such as 8, and *n* varies from 4 to 32 stepped by 4. For each *n*, the average PSNRs of the motion predicted frames are calculated for several standard test sequences. The best n is selected from the curve of the average PSNR versus *n*. The best *n* is 4.

Then to find a proper value for *w*. Let *n* to be 4 and *w* varied from 4 to 32 stepped by 4. For each *w*, the average PSNRs of the motion predicted frames are calculated for the same sequences. The best *w* is selected from the curve of the average PSNR versus *w*. The result for *w* is 16.

At last, repeat the first step with *w* set to be 16. The result keeps unchanged. It proves that the results are valid.

C. Fast Calculation of OWT

In some software implementations of video encoder such as H.263 TMN Version 2.0 [34], the mean value of each *4x4* block in the luminance frame is calculated before ME. These mean values are used to examine if there exists motion between a macroblock in current frame and the corresponding macroblock in the reference frame. In OWT, these mean values can be efficiently utilized to speed up the calculation of the transforming threshold. That is the reason why the parameters *w* and *n* are confined to be integer times of 4.

Assume that the luminance frame is ***F***. Let us define a new frame ***F4***, in which each pixel represents a *4x4* block of pixels in ***F***, and the pixel intensity is the mean of the corresponding block. As shown in Fig. 6, suppose that the current transforming block is pixel 1 in ***F4***. We denote the ***F4*** pixels in the nth transforming window corresponding to the transforming block *n* (such as 1, 2, 3 and 4 shown in Fig. 6) as *wn(i, j)*, where *i* and *j*, ranging from 0 to 8, represent the order of row and column, respectively. Therefore the threshold for block 1 is given by Eqn. 8, and for block 2, 3 and 4, the thresholds *thn* (*n=2, 3, 4*) are given by Eqn. 9 to 11, respectively.

$$th1=\sum_{i,j=0}^{8} w1(i,j) \tag{8}$$

$$th2 = \sum_{i,j=0}^{8} w2(i,j) = th1 - \sum_{i=0}^{8} w1(i,0) + \sum_{i=0}^{8} w2(i,8) \tag{9}$$

$$th3 = \sum_{i,j=0}^{8} w3(i,j) = th1 - \sum_{j=0}^{8} w1(0,j) + \sum_{j=0}^{8} w3(8,j) \tag{10}$$

$$th4 = \sum_{i,j=0}^{8} w4(i,j) = th3 - \sum_{i=0}^{8} w3(i,0) + \sum_{i=0}^{8} w4(i,8) \tag{11}$$

The window mean *wm* used in Eqn. 7 can be gained by dividing *thn (n=1 to 4)* with the number of blocks covered in the window. Computations of thresholds for other blocks are as similar. Using this method, much time can be saved in calculating the transforming thresholds.

From Fig.6, we can see that *w3(i,0)=w1(i+1,0)* and *w4(i,0)=w2(i+1,0)* for *i=0 to 7*. Therefore in Eqn. 9 and 11 there are many repetitive computations. These relationships are utilized to further speed up the computation.

Applying the ideas described above, the computation strategy of the transforming thresholds for OWT is taken as the following steps:

1) For each block in the left edge of a frame, i.e. *F4(i,0)* (*i=0* to *NoL-1*, and *NoL* is the number of lines in ***F4***), a summative value *Row_Sum(i)* is calculated as

$$Row\_Sum(i) = \sum_{j=0}^{4} F4(i,j) \tag{12}$$

2) For each block (including the ones in the left edge), a summative value *Col_Sum(i,j)* is calculated as Eqn. 13, where *F4(k,j)=0* if *k<0* or *k> NoL-1*.

$$Col\_Sum(i,j) = \sum_{k=i-4}^{i+4} F4(k,j) \tag{13}$$

3) Calculate the thresholds for the left-edge blocks. For the top-left block, the threshold *th(0,0)* is given by Eqn. 14. For the other left-edge blocks *F4(i,0), i=1* to *NoL-1*, the threshold *th(i,0)* is given by Eqn. 15, where *Row_Sum(k)=0* if *k<0* or *k> NoL-1*.

$$th(0,0) = \sum_{k=0}^{4} Row\_Sum(k) \tag{14}$$

$$\begin{aligned} th(i,0) &= \sum_{k=i-4}^{i+4} Row\_Sum(k) \\ &= th(i-1,0) - Row\_Sum(i-5) + Row\_Sum(i+4) \end{aligned} \tag{15}$$

4) Calculate the thresholds for the remaining blocks. The thresholds are given by Eqn. 16, where *Col_Sum(i,k)=0* if *k<0* or *k> NoC-1*, and *NoC* is the number of columns in ***F4***.

$$th(i,j) = \sum_{k=j-4}^{j+4} Col\_Sum(i,k) = th(i,j-1) - Col\_Sum(i,j-5) + Col\_Sum(i,j+4) \quad (16)$$

The window mean *wm* used in Eqn. 7 can be gained by dividing *th(i,j)* with the number of blocks covered in the window.

Now let's consider the overall computations required for OWT. For a frame of *N* lines with *M* pixels per line, according to Eqn. 12-16, the computation amount is $\frac{5}{8}MN + N + 4$ additions. In addition, each *4x4* block needs a division and each pixel needs a compare operation. So the overall computation amount of OWT for an entire frame is $\frac{5}{8}MN + N + 4$ additions, $\frac{1}{16}MN$ divisions and MN compares. Note that the computations of calculating the block means is not taken into account because they are already available. From the simulation results we will see that even those computations are considered, the transforming speed of OWT is still 37 times faster than FT.

D. The Identity of OWT, BMT and FT

Although OWT, BMT and FT seem very different from each other, they have something identical. Let's consider the filter kernel $\boldsymbol{K}$ of *rxr* (*r* is an odd number) as

$$K_{i,j} = 1/r^2, \ for\ all\ i, j \quad (17)$$

In fact, this is an averaging filter, and the FT with this filter kernel is identical to the OWT with *n=1* and *w=(r-1)/2*. The filter in [21] is actually a subsampled version of the averaging filter. In addition, BMT is apparently a special case of OWT with *w=0*. The other identity among them is that all of their objectives are to identify the pixel attribute of brighter or darker compared to its surroundings. If a pixel is brighter than its surroundings, it will be denoted as 1; otherwise it will be denoted as 0.

According to the above analysis, OWT can be looked as a fast approximate implementation of FT. This might have general meaning in filter implementation.

IV. The Fast Binary Block Matching Algorithm

This section presents the proposed fast ME algorithm FBBMA, which is based on OWT and seven improving techniques. First the improving techniques are described one by one. Then the details of FBBMA will be given at the end of this section.

A. Still Macroblock Detection

Each MB contains 16 sub-blocks of *4x4*. If all the 16 absolute differences between each sub-block and the

corresponding sub-block in the reference frame are smaller than the predefined threshold *th1*, then the MB is looked as still.

If a MB cannot be declared as a still MB according to the sub-block means, then calculate SAD of position *(0,0)*, i.e., *SAD(0,0)*. If *SAD(0,0)* is smaller than the predefined threshold *th2*, the MB is looked as still; otherwise the MB is not a still MB.

B. Adaptive Central-Search-Point Prediction

The central-search-point prediction technique [35]-[38] utilizes the spatio-temporal correlation of MVs to estimate a central search point. The average distance from the predicted point to the optimal point is shorter than that from the zero point. Therefore the size of the search window can be reduced and the searching speed can be increased. Recently, this class of methods has gained a great deal of developments [39]-[43].

A typical case is that the central search point is predicted as Eqn. 18, which is employed in H.263 TMN Version 2.0 [34].

$$mv0 = mv0' + [(mv1 - mv1') + (mv2 - mv2')]/2 \tag{18}$$

where ***mv0*** is the predicted MV of current MB, ***mv1*** and ***mv2*** are the MVs of the left-hand-side and top-side neighboring MBs in current frame, the ones with prime signs are the MVs of the corresponding MBs in the reference frame, respectively, as shown in Fig. 7.

There are two modifications in the proposed *adaptive center-search-point prediction*. One is called *search center selection*; and the other is *search range limitation*.

Search Center Selection Sometimes ***mv0*** is not a good prediction compared to the zero point *(0,0)*. In our method, the central search point *(px,py)* is decided according to the SADs of ***mv0*** and *(0,0)*. If SAD of ***mv0*** is smaller than *SAD(0,0)*, then *(px,py)* is set to be ***mv0***, otherwise it is set to be *(0,0)*.

Search Range Limitation In video coding, the resulting MV is usually limited to a predefined range. In this case, when the predicted central search point *(px,py)* is not identical to *(0,0)*, the resulting MV might locate outside of the MV range even when a small search distance is used. As shown in Fig. 8, *(px,py)* is the central search point, the search distance is *w* and the MV range is [-*w*, *w*]. Assume that point A wins at last in the motion estimation process, then the actual result will be B, which is the truncated result of A. However, it is possible that there exists a point C, and the order of the SAD values of ABC is *SAD(A)* < *SAD(C)* < *SAD(B)*. Therefore B is not as good as C, and it is not the best candidate within the search range. In order to get rid of such cases, we can discard the outside-search-range points before calculating the SAD. By using this strategy, better MV results can be obtained and

much computational cost can be saved, as can be seen from the experimental results.

C. Variable Search Distance

Generally, the center-search-point prediction process can give good prediction therefore the predicted point *(px,py)* is close to the optimal point, and the search can be performed in a small range. However, sometimes the predicted results are not very good, then the search distance should be a larger value. In our method, the search distance *SD* is decided according to the SAD value of the predicted result *(px,py)*, as given by

$$SD = \begin{cases} w/4, & if\ SAD(px, py) < th3 \\ w/2, & if\ th3 \le SAD(px, py) < th4 \\ w, & if\ SAD(px, py) \ge th4 \end{cases} \tag{19}$$

where *w* is related to the predefined MV range, i.e., the MV range is [-*w*, *w*].

D. Fast Binary Block Matching

There are two modifications in *fast binary block matching*. One is to employ 32-bit exclusive-or operation to perform the binary bit-plane match; the other is to avoid the repetition in obtaining the 32-bit data from bit-plane.

In Eqn. 2, $B_t(m,n)$ and $B_{t-i}(m+x,n+y)$ are 1-bit binary values, and $\otimes$ denotes 1-bit exclusive-or operation. Therefore computing NNMP once needs 256 exclusive-or operations and 255 additions if the standard macroblock of *16x16* is used. However, if multi-bit values are used to perform the exclusive-or operation, the computation will be greatly reduced. Here, we use 32-bit values. Then Eqn. 2 is changed to

$$NNMP = \sum_{i=0}^{7} NumOf1s(u(i) \otimes v(i)) \tag{20}$$

where *u(i)* and *v(i)* are 32-bit unsigned values from the *16x16* blocks, $\otimes$ denotes bitwise-exclusive-or operation, and *NumOf1s(x)* returns the number of "1" bits in the unsigned 32-bit value *x*. In *u(i)* and *v(i)*, each bit represents one pixel, therefore one 32-bit value represent 2 lines of pixels and there are totally 8 32-bit unsigned values in a *16x16* block. According to Eqn. 20, computing NNMP once needs only 8 exclusive-or operations and 7 additions. To get the number of "1" bits, we use a look-up table with 256 entrances for an 8-bit value. Therefore, it needs 4 look-up-table operations and 3 additions to get the number of "1" bits in a 32-bit value.

Computation of NNMP can be further sped up by removing repetitive operations in obtaining the 32-bit data from bit-plane. Let's consider a special case as shown in Fig. 9. After 1BT, the bit-planes are stored in buffers, which are addressed byte by byte. Assume that the MB to be matched is the one that starts from *(0,0)*, therefore the first 32-bit value *u(0)* is formed by 4 bytes: the first 2 bytes in line 0 and the first 2 byte in line 1. When testing the position *(0,0)*

of the reference frame, the first 32-bit value *v(0)* is formed similarly as *u(0)*. However, testing the position *(3,4)* will be totally different. Now *v(0)* is formed by part of the first 3 bytes in line 4 and line 5. Let the addresses of line 4 and line 5 be *p4* and *p5*, respectively, then *v(0)* is given by

$$\begin{cases} v(0).byte0 = (p4(0) << 3) \,|\, (p4(1) >> 5); \\ v(0).byte1 = (p4(1) << 3) \,|\, (p4(2) >> 5); \\ v(0).byte2 = (p5(0) << 3) \,|\, (p5(1) >> 5); \\ v(0).byte3 = (p5(1) << 3) \,|\, (p5(2) >> 5). \end{cases} \tag{21}$$

where << and >> are bitwise left shift and right shift operators, and | denotes bitwise-inclusive-or operation. Other *u(i)* and *v(i)* are obtained as similar. From Eqn. 21 we can see that to get the 32-bit values needs complex operations. What's more, the operation might be repeatedly performed for certain positions. Now assume that the MB to be matched is the MB starting from *(16,0)*. The position *(3,4)* is within the search range when a search distance not smaller than 13 is used. Therefore the 32-bit values for position *(3,4)* are needed again. To remove the repetitive computations, we calculate the 32-bit values beforehand for each possible position and save them in a buffer. When a value is needed, the only operation is to bring it out from the buffer.

E. Halfway Stop in Block Distortion Computation

In a block matching ME scheme, after the BDM of each position is calculated, it is compared to the minimum BDM of the tested positions. The computation for current position might be stopped anytime when part of its BDM is already larger than the minimum BDM. A detailed description can be found in [44]. In Eqn. 20, there are 8 steps to add up the entire NNMP. We compare current part of NNMP to the minimum NNMP (MNNMP) of the tested positions after each step. If current part of NNMP is not smaller than MNNMP, then the testing position will be discarded immediately without further computations; otherwise continue to the next step.

F. Center-Biased Search Order

There are two significant modifications in our proposed *center-biased search order*: one is *unshaped zonal search (UZS)* and the other is *minimum BDM decrease*. UZS is enlightened by the zonal search algorithm [45] and its developments [46]-[47]. In the standards of MPEG1/2/4 and H.261/263, MVs are differentially encoded and the encoding pattern is as shown in Fig.10. The reason that the class of zonal search algorithms [45-47] became better and better is just because that they are more and more compliant with the encoding pattern. However, as can be seen from Fig. 10, there is no regular zonal partitioning according to bits required for encoding MVs. Therefore any regular zonal search such as circular zonal search [46] or diamond zonal search [47] is not the best search pattern.

Unshaped Zonal Search (UZS) To fully utilize the encoding pattern, we use an unshaped zonal partitioning strategy instead of any regular method in our scheme. Zones are partitioned purely according to the bits required for encoding MVs. That is, all positions with the MVs that require the same number of encoding bits are partitioned into one zone. Therefore there are totally 20 zones when the search distance is 15, as shown in Fig. 19. UZS is defined as follows:

1) Partition the all the candidate positions into different zones according to the method described above.
2) The search is performed zone by zone. The more the bits of a zone, the latter in the zone the search would be performed.
3) Within a zone, the search order of different positions is decided according to their distances from the search center. The closer the distance, the earlier the position is tested.

Minimum BDM Decrease To further benefit the central positions, a more strict comparing condition is employed: each time the search zone is changed, MNNMP is reduced by a small number.

Due to the center-biased characteristic of MVs, both of the two modifications in the proposed *center-biased search order* can bring forward averagely the time when the optimal point is found. Therefore in the early search stages MNNMP will be smaller, and the improving technique *halfway stop in block distortion computation* can perform more efficiently. Moreover, the output MVs will require fewer bits to be encoded. These can be seen from the experimental results.

G. Multiple Candidates Selection

In *multiple candidates selection*, not one but *th5* candidates with the minimum NNMPs are found in BBMA. Then the best is selected amongst the *th5* candidates and the predicted center *(px,py)*. The position with the minimum SAD will be the final winner. In order to benefit the central position, *SAD(px,py)* is decreased by 100 before compared to other SADs, similar as in H.263 TMN encoder 2.0.

H. FBBMA

The algorithm FBBMA is defined as the following steps:

1) Each frame is partitioned into non-overlapping *4x4* blocks, and the block mean is calculated for each block. Then OWT is employed to obtain the bit-plane. After that, the following steps are performed MB wisely, and they are not performed for the first frame.
2) For each MB, the technique *still macroblock detection* is first applied to detect whether current MB is a still MB. If the MB is still, then the motion vector ***mv****=(0,0)* and go to next MB; otherwise go to step 3.

3) Applying the technique *adaptive center-search-point prediction* to predict the central search point *(px,py)*, and calculate *SAD(px,py)* if *(px,py)* is not *(0,0)*. If *SAD(px,py)* is smaller than the threshold *th2*, then ***mv***=*(px,py)* and go to next MB; otherwise go to step 4.
4) Decide the search distance by employing the technique *variable search distance*.
5) According to the search order defined by the technique *center-biased search order*, applying the techniques *fast binary block matching* and *halfway stop in block distortion computation*, beginning with *(px,py)*, search *th5* candidates with the minimal NNMPs.
6) Applying the technique *multiple candidates selection* to find the final MV for current MB from the *th5* candidates, and then go to next MB.

The predicting performance and the searching speed can be compromised by adjusting the five parameters *th1-5*.

V. Experimental Results

The test sequences are Flower Garden, Football, Mobile & Calendar and Table Tennis, all with the size *352x240* and frame rate 30 frame/s. The frame number of each sequence is 115, 125, 140 and 112, respectively. Only the luminance component is considered. In ME, the original previous frame is taken as the reference frame, the MV range is [*-15, 15*], and the block size for ME is *16x16*. All the PSNR values in the following tables are calculated between the predicted sequence and the original sequence, the unit is decibel (dB). The values for the five parameters *th1-5* are respectively 3、768、3072、7680 and 4. They are relatively robust for different situations and video sequences.

A. Performance of OWT

To compare the performances of different 1BT algorithms, we combine them with the conventional FS and one of the fast ME algorithms, the three-step search (TSS) algorithm [10]. The transforming block size for BMT is 16×16. The results are shown in Table I, where PSNR-FS or PSNR-TSS is the output PSNR of 1BT method when FS or TSS is used as the ME search strategy; Speed Up is the speed up ratio versus FT. All the necessary overhead computation costs are taken into account. None of the seven improving techniques is applied in this experiment, and the exclusive-or operation is performed in 8-bit wise.

It is seen from Table I that, whatever under the full search or the fast search, the PSNR of OWT is much higher than BMT, and similar (slightly higher for Flower Garden and Football, and slightly lower for the other two sequences) as FT. With such a good predicting performance, OWT can perform 37 times faster than FT.

B. Performance of Fast Binary Block Matching

Table II shows the speed up ratios, with comparison to the 8-bit exclusive-or operation case, of OWT/FS with different binary block matching methods. OWT/FS denotes the ME algorithm that employing OWT as 1BT method and FS as the search strategy. None of the other six improving techniques is applied.

We can see that applying 32-bit instead of 8-bit exclusive-or operation makes the matching speed doubled. By avoiding the repetition operations in obtaining the 32-bit data from bit-plane, the speed up ratio can be increased to about 2.5 times.

C. Performance of Adaptive Central-Search-Point Prediction

Three cases are considered to show the performance of the proposed technique *adaptive central-search-point prediction*: No Prediction that the central search point is always (*0*,*0*); Original Prediction that the predicting method in Eqn. 18 is applied; and the proposed method. The results are shown in Table III, where the speed up ratio is obtained with comparison to the no-prediction case. Note that all of the other six improving techniques are employed in this experiment.

It can be seen that the proposed prediction method performs better than the original prediction method in terms of both PSNR and speed up ratio. By using the *adaptive central-search-point prediction*, the PSNR is improved by up to 0.14 dB, and the computational cost is reduced by 1.3 to 2.6 times for different sequences.

D. Performance of Center-Biased Search Order

In order to show the performance of the proposed technique *center-biased search order*, the natural scan order and the diamond zonal search (DZS) [47] are implemented for comparison. The results are shown in Table IV, where the parameter Bits4MV denotes the average bits required for encoding the MVs of each macroblock, the unit is bits/MB. Here the speed up ratio is obtained with comparison to the case of the scan order. Note that three of the improving techniques *fast binary block matching*, *adaptive central-search-point prediction*, and *halfway stop in block distortion computation* are applied in this experiment.

Due to the center-biased characteristic of the MV of real world sequences, the search orders (such as DZS and the proposed method) that benefit central positions give better PSNR performance. Because of the reason as described in Subsection IV.F, they can also improve the searching speed. In addition, the output MVs will be more natural and correlative thus need fewer bits to be encoded. It is seen from Table IV that, compared to the scan order: DZS can

improve PSNR from 0.01 to 0.06 dB for different sequences, the speed up ratio is 1.1 to 1.2 times, and the bits required for encoding MV are reduced by 0.1 to 0.3 bits/MB; the proposed method can improve PSNR from 0.05 to 0.16 dB for different sequences, the speed up ratio is 1.2 to 1.4 times, and the bits required for encoding MV are reduced by 0.3 to 1.0 bits/MB. The superiority of the proposed *center-biased search order* over DZS is obvious.

E. Performance of FBBMA

FBBMA is compared, in terms of PSNR, speed up ratio to FS, and the average bits required for encoding the MVs of each macroblock, versus the ME algorithms FS, TSS [10], MPC [32] and DS [14]. The results are shown in Table V. The value of the threshold needed by MPC is set to be 28, which provides the best PSNR performance among all possible values for the tested sequences. Note that all the necessary overhead computation costs are taken into account when calculating the speed up ratios.

It is obvious from the results that the proposed FBBMA is significantly faster than all these algorithms while achieving close and in most cases significantly better PSNR. On the average, for the sequences examined in this test, FBBMA is roughly 2.7, 77.2 and 1.4 times faster whereas its PSNR is approximately 1.10dB, 0.11dB and 0.24dB higher than TSS, MPC and DS, respectively. For the cases examined, FBBMA is about 69.5 times faster than FS, while having an average PSNR loss of only 0.23dB. In addition, the output MVs of FBBMA need the least bits to be encoded, even compared to FS.

In Figures 11 to 14, which show the per frame results, FBBMA is obviously significantly better than TSS, MPC and DS, and its PSNR is close to FS for almost all the frames of each sequence.

VI. CONCLUSIONS

In this paper, we propose a novel simple and efficient 1BT algorithm, called *the overlap-windowed thresholding algorithm (OWT)*, and a new fast block based motion estimation algorithm, called *the fast binary block matching algorithm (FBBMA)*. The proposed FBBMA is based on OWT and the seven improving techniques 1) *fast binary block matching*; 2) *adaptive central-search-point prediction*; 3) *center-biased search order*; 4) *still macroblock detecting*; 5) *halfway stop in block distortion computation*; 6) *variable search distance*; and 7) *multiple candidates selection*. The first three of them contain significant modifications done by the authors, and are proposed literarily here in the first time.

Our experimental results demonstrate the superiorities of all the proposed algorithms and modifications. With similar PSNR performance, OWT can perform 37 times faster than the 1BT algorithm that provides the best PSNR performance among the existing methods. *Fast binary block matching* can improve the matching speed by 2.5 times.

Both *adaptive central-search-point prediction* and *center-biased search order* can improve not only PSNR but also the searching speed. What's more for *center-biased search order* is that it also decreases at the same time the bits required for encoding the output MVs. For the examined sequences, the proposed FBBMA performs averagely 69.5 times faster than FS, with PSNR of the predicted sequences degraded by only 0.23 dB, while at the same time the bits for MV encoding is reduced. The algorithm performs significantly better in terms of all the three parameters than the classical fast algorithm TSS, the similar algorithm MPC, and the recent algorithm DS, which was recently adopted to the MPEG-4 verification model.

ACKNOWLEDGMENT

The authors wish to thank Dr. V. Bhaskaran, Epson Palo Alto Lab, for the helpful discussions on one-bit transform techniques, and the anonymous reviewers for their valuable comments.

REFERENCES

[1] ISO/IEC 11172-2, "Information technology - coding of moving picture and associated audio for digital storage media up to about 1.5 Mbit/s: Part 2 video," Aug. 1993.
[2] ISO/IEC 13818-2, "Information technology-generic coding of moving pictures and associated audio: Part 2 video," Nov. 1995.
[3] ISO/IEC JTC1/SC29/WG11 N2725, "Overview of the MPEG-4 Standard," Mar. 1999.
[4] ITU-T Recommendation H.261, "Video codec for audiovisual services at p x 64 kbits," Mar. 1993.
[5] ITU-T Recommendation H.263, "Video codec for low bitrate communication," May 1996.
[6] G. Bjontegaard (editor), "H.26L Test Model Long Term Number 7 (TML-7) draft 0", ITU-T Q.15/SG16 document VCEG-M81, May 2001, available from ftp://standard.pictel.com/video-site/0104_Aus/VCEG-M81d0.doc.
[7] C. Cafforio and F. Rocca, "Methods for measuring small displacements of television images," *IEEE Trans. Inform. Theory*, vol. IT-22, no. 5, pp. 573-579, Sept. 1976.
[8] J. C. -H. Ju, Y. -K. Chen, and S. Y. Kung, "A fast rate-optimized motion estimation algorithm for low-bit-rate video coding," *IEEE Trans. Circuits. Syst. Video Technol.*, vol. 9, no. 7, pp. 994-1002, Oct. 1999.
[9] J. R. Jian and A. K. Jian, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. COM-29, pp. 1799-1808, Dec. 1981.
[10] T. Koga, K. Iinuma, A. Iijima, and T.Ishiguro, "Motion-compensated interframe coding for video conferencing," in *Proc. NTC81*, New Orleans, LA, USA, 1981, pp. C9.9.1-9.6.5.
[11] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits. Syst. Video Technol.*, vol. 4, pp. 438-442, Aug. 1994.
[12] L. -M. Po and W. -C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 313-317, June 1996.
[13] F. -H. Cheng and S. -N. Sun, "New fast efficient two-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 7, pp. 977-983, Oct. 1999.
[14] S. Zhu and K. -K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Trans. Image Process.*, vol. 9, no. 2, pp. 287-290, Feb. 2000.
[15] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, pp. 148-157, Apr. 1993.
[16] H. Bi and W. -Y. Chan, "Rate-distortion optimization of hierarchical displacement fields," *IEEE Trans. Circuits Syst. Video Technol.*, vol.8, pp. 18-24, Feb. 1998.
[17] M. C. Chen and A. N. Willson, Jr., "Rate-distortion optimal motion estimation algorithms for motion-compensated transform video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 147-148, Apr. 1998.
[18] D. T. Hoang, P. M. Long, and J. S. Vitter, "Efficient cost measures for motion estimation at low bit rates," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 488-500, Aug. 1998.

[19] S. Lee and S. -I. Chae, “Motion estimation algorithm using low resolution quantisation,” *Electron. Lett.*, vol. 32, no. 7, pp. 647-648, Mar. 1996.
[20] J. Feng, K. -T. Lo, H. Mehrpour, and A. E. Karbowiak, “Adaptive block matching motion estimation algorithm using bit-plane matching,” *in Proc. ICIP-95*, Washington D.C., USA, 1995, pp. 496-499.
[21] B. Natarajan, V. Bhaskaran, and K. Konstantinides, “Low-complexity block-based motion estimation via one-bit transforms,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 4, pp. 702-706, Aug. 1997.
[22] X. Lee, “A fast feature matching algorithm of motion compensation for hierarchical video codec,” in *Proc. SPIE VCIP-92*, Boston, MA, USA, 1992, vol. 1818, pp. 1462-1474.
[23] Y. -Q. Zhang and X. Lee, “Feature-based motion estimation,” in *Proc. IEEE Int. Symp. Communications*, Taipei, Dec. 1995.
[24] X. Lee and Y. -Q. Zhang, “A fast hierarchical motion-compensation scheme for video coding using block feature matching,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 6, pp. 627-635, Dec. 1996.
[25] X. Song, Y. -Q. Zhang, and T. Chiang, “Hierarchical motion estimation algorithm using binary pyramid with 3-scale tilings,” in *Proc. SPIE VCIP-98*, Jan. 1998, pp. 80-87.
[26] X. Song, T. Chiang, and Y. -Q. Zhang, “A scalable hierarchical motion estimation algorithm for MPEG-2,” in *Proc. IEEE ISCAS-98*, June 1998.
[27] X. Song, T. Chiang, X. Lee, and Y. -Q. Zhang, “Fast binary pyramid motion estimation,” in *Proc. of 2000 5th Int. Conf. Signal Processing*, Beijing, China, Aug. 2000.
[28] J. Feng, K.-T. Lo, H. Mehrpour, and A.E. Karbowiak, “Adaptive block matching algorithm for video compression,” *IEE Proc. –Vis. Image Signal Processing*, vol. 145, no. 3, pp. 173-178, June 1998.
[29] M. M. Mizuki, U. Y. Desai, I. Masaki, and A. Chandrakasan, “A binary block matching architecture with reduced power consumption and silicon area requirement,” in Proc. *IEEE ICASSP-96*, Atlanta, USA, 1996, vol. 6, pp. 3248-3251.
[30] Y. -L. Chan and W. -C. Siu, “Edge oriented block motion estimation for video coding,” *IEE Proc. –Vis. Image Signal Processing*, vol. 144, no. 3, pp. 136-144, June 1997.
[31] P. H. W. Wong and O. C. Au, “Modified one-bit transform for motion estimation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 7, pp. 1020-1024, Oct. 1999.
[32] H. Gharavi and M. Mills, “Block matching motion estimation algorithms—new results,” *IEEE Trans. Circuits. Syst.*, vol. 37, no. 5, pp. 649-651, May 1990.
[33] M. Frigo and S. G. Johnson. (Sept. 2000). FFT library: The fastest Fourier transform in the west. [Online]. Available WWW: http://www.fftw.org.
[34] Telenor R&D. (June 1996). H.263 encoder version 2.0. [Online]. Available FTP: ftp://bonde.nta.no/pub/tmn/software/.
[35] A. Puri, H. -M. Hang, and D. L. Schilling, “An efficient block-matching algorithm for motion-compensated coding,” in *Proc. IEEE CASSP-87*, 1987, pp. 1063-1066.
[36] C. -H. Hsieh, P. -C. Lu, J. -S. Shyn, and E. -H. Lu, “Motion estimation algorithm using interblock correlation,” *Electron. Lett.*, vol. 26, no. 5, pp. 276-277, Mar. 1990.
[37] S. Zafar, Y. -Q. Zhang, and J. S. Baras, “Predictive block-matching motion estimation for TV coding – part I: inter-block prediction,” *IEEE Trans. Broadcasting*, vol. 37, no. 3, pp. 97-101, Sept. 1991.
[38] Y. -Q. Zhang and S. Zafar, “Predictive block-matching motion estimation for TV coding – part II: inter-frame prediction,” *IEEE Trans. Broadcasting*, vol. 37, no. 3, pp. 97-102-105, Sept. 1991.
[39] L. Luo, C. Zou, X. Gao, and Z. He, “A new prediction search algorithm for block motion estimation in video coding,” *IEEE Trans. Consumer Electron.*, vol. 43, no. 1, pp. 56-61, Feb. 1997.
[40] D. -W. Kim, J. -S. Choi and J. -T. Kim, “Adaptive motion estimation based on spatio-temporal correlation,” *Signal Processing: Image Commun.*, vol. 13, pp. 161-170, Aug. 1998.
[41] J. -C. Tsai, C. -H. Hsieh, S. -K. Weng, and M. -F. Lai, “Block-matching motoin estimation using correlation search algorithm,” *Signal Processing: Image Commun.*, vol. 13, pp. 119-133, Aug. 1998.
[42] J. M. Jou, P. -Y. Chen, and J. -M. Sun, “The gray prediction search algorithm for block motion estimation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 6, pp. 843-848, Sept. 1999.
[43] J. -B. Xu, L. -M. Po, and C. -K. Cheung, “Adaptive motion tracking block matching algorithms for video coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 7, pp. 1025-1029, Oct. 1999.
[44] A. Tabatabai, R. Jasinschi, and T. Naveen, “Motion estimation methods for video compression - a review,” *J. Franklin Inst.*, vol. 335, no. 8, pp. 1411-1441, Nov. 1998.
[45] Z. He and M. L. Liou, “A high performance fast search algorithm for block matching motion estimation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 5, pp. 826-828, Oct. 1997.
[46] A. M. Tourapis, O. C. Au, and M. L. Liou, “Fast motion estimation using circular zonal search,” in *Proc. SPIE VCIP-99*, San Jose, CA, USA, Jan. 1999, vol. 3653, pp. 1496-1504.
[47] A. M. Tourapis, O. C. Au, M.L. Liou, and G. Shen, “An advanced zonal block based algorithm for motion estimation", in *Proc. IEEE ICIP-99*, Kobe, Japan, October 1999, section 26PO3.1.

List of Figure and Table Captions

Figures and Tables

20	20	180	180	180	180	20	20
20	20	180	180	180	180	20	20
20	20	180	180	180	180	20	20
20	20	180	180	180	180	20	20

(a) (b)

0	1	1	0	0	0	1	1	1	1	0	0
0	1	1	0	0	0	1	1	1	1	0	0
0	1	1	0	0	0	1	1	1	1	0	0
0	1	1	0	0	0	1	1	1	1	0	0

(c) (d) (e)

Fig. 1. Special cases of 8-bit blocks and bit-planes.

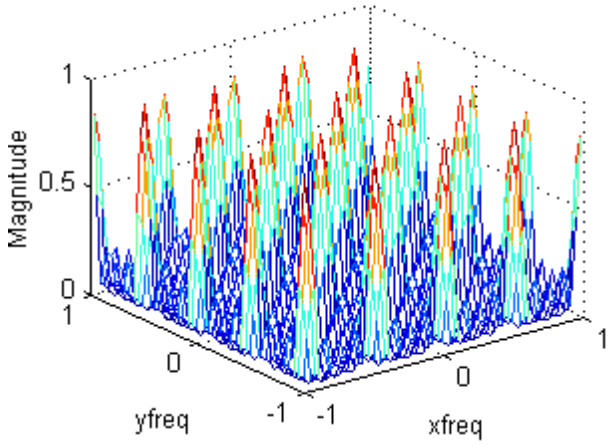

Fig. 2. Frequency response of the filter kernel ***K***.

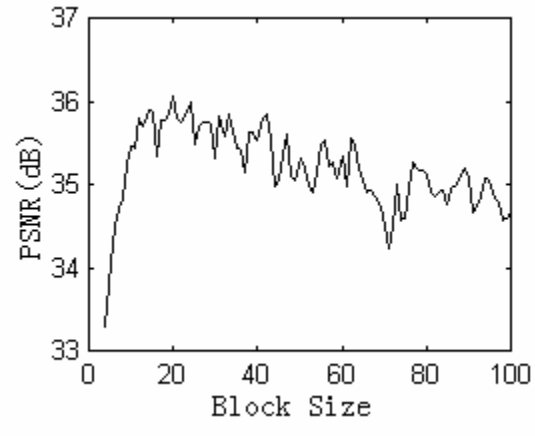

Fig. 3. PNSR curves of BMT with different transforming block sizes.

Fig. 4. 1BT transformed bit-planes using (a) BMT with block size *4x4*; (b) BMT with block size *20x20*; (c) BMT with block size *352x288*; (d) OWT with block size *4x4* and window size *22x22*.

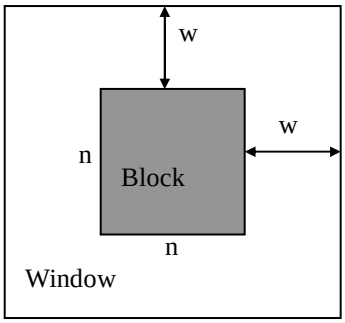

Fig. 5. The transforming block and the threshold window.

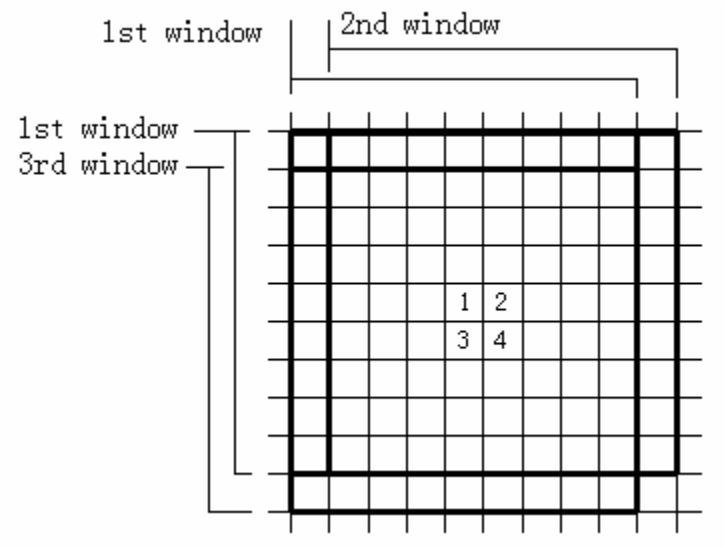

Fig. 6. Geometry for calculating the transforming thresholds of OWT.

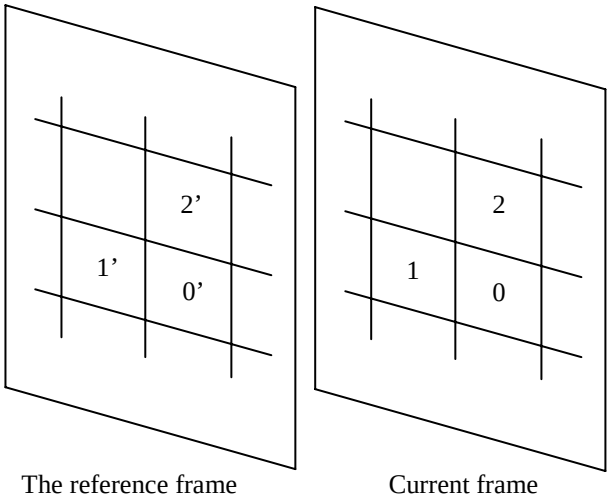

Fig. 7. Geometry of macroblocks for predicting the initial motion vector.

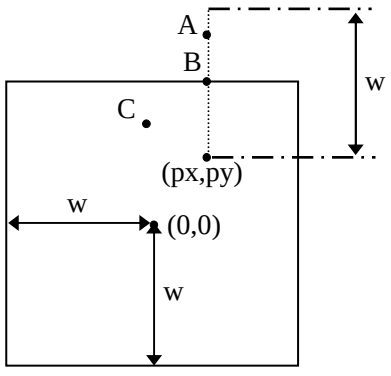

Fig. 8. Search range limitation.

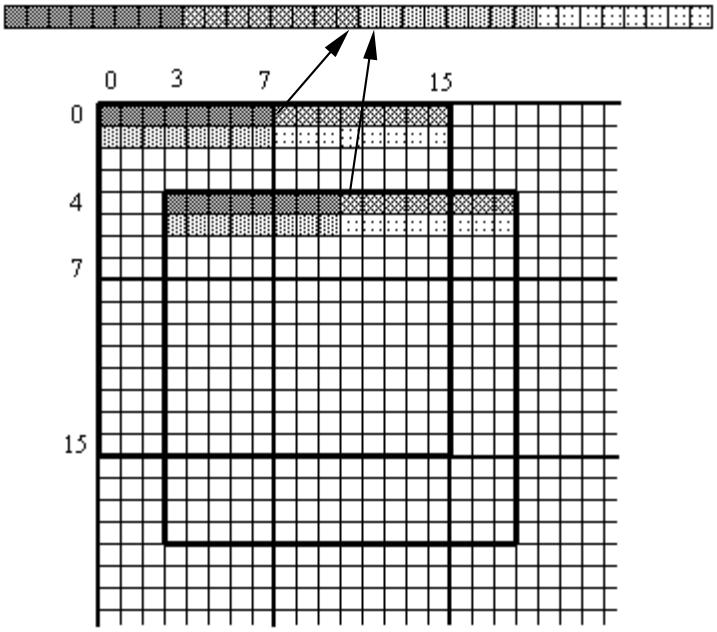

Fig. 9. Geometry of binary block matching.

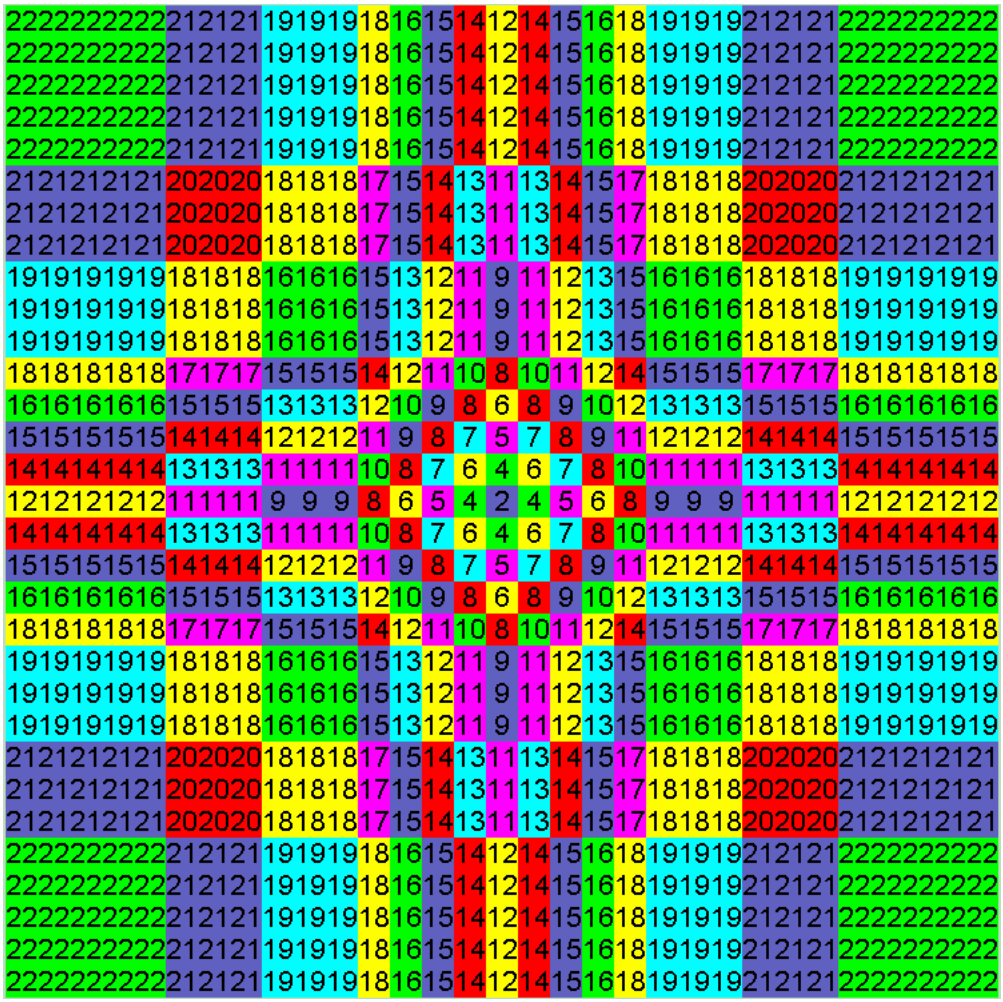

Fig. 10. Bits required for encoding MVs according to distance from the predicted central search point. Here the maximum distance is 15.

TABLE I
PERFORMANCE OF OWT WITH COMPARSIONS TO OTHER 1BT ALGORITHMS

1BT Algorithm	Performance Parameter	Sequences			
		Flower Garden	Football	Mobile & Calendar	Table Tennis
BMT	PSNR-FS	22.90	21.01	22.33	27.76
	PSNR-TSS	20.09	20.25	21.95	26.75
	Speed Up	52.5	54.2	55.4	52.7
FT	PSNR-FS	23.31	21.83	22.71	28.76
	PSNR-TSS	20.32	20.69	22.26	27.21
	Speed Up	1.0	1.0	1.0	1.0
OWT	PSNR-FS	23.41	21.93	22.69	28.59
	PSNR-TSS	20.48	20.89	22.21	27.32
	Speed Up	37.6	37.4	40.0	38.0

TABLE II
SPEED UP PERFORMANCE OF FAST BINARY BLOCK MATCHING

Binary Block Matching Method	Sequences			
	Flower Garden	Football	Mobile & Calendar	Table Tennis
8-bit XOR	1.0	1.0	1.0	1.0
32-bit XOR	2.0	2.0	2.0	2.0
32-bit XOR + No Repetition	2.5	2.6	2.6	2.5

TABLE III
PERFORMANCE OF ADAPTIVE CENTRAL-SEARCH-POINT PREDICTION

Predicting Method	Performance Parameter	Sequences			
		Flower Garden	Football	Mobile & Calendar	Table Tennis
No Prediction	PSNR	23.69	22.44	22.96	29.36
	Speed Up	1.0	1.0	1.0	1.0
Original Prediction	PSNR	23.67	22.41	22.95	29.31
	Speed Up	1.0	1.1	1.0	1.0
Proposed Prediction	PSNR	23.71	22.50	22.96	29.45
	Speed Up	2.6	1.3	1.5	1.3

TABLE IV
PERFORMANCE OF CENTER-BIASED SEARCH ORDER

Search Order	Performance Parameter	Sequences			
		Flower Garden	Football	Mobile & Calendar	Table Tennis
Scan Order	PSNR	23.40	21.93	22.81	28.61
	Speed Up	1.0	1.0	1.0	1.0
	Bits4MV	4.7	6.6	2.9	5.2
Diamond Zonal Search	PSNR	23.43	21.96	22.82	28.67
	Speed Up	1.2	1.1	1.1	1.1
	Bits4MV	4.4	6.5	2.8	5.1
Proposed Search Order	PSNR	23.49	22.03	22.86	28.77
	Speed Up	1.4	1.2	1.2	1.2
	Bits4MV	3.7	6.1	2.6	4.5

TABLE V
PERFORMANCE OF FBBMA WITH COMPARISONS TO OTHER ME ALGORITHMS

ME Algorithm	Performance Parameter	Sequences				Average
		Flower Garden	Football	Mobile & Calendar	Table Tennis	
FS	PSNR	23.79	22.87	22.99	29.86	24.88
	Speed Up	1.0	1.0	1.0	1.0	1.0
	Bits4MV	3.8	6.1	2.6	4.6	4.2
TSS	PSNR	21.48	21.76	22.59	28.38	23.55
	Speed Up	25.0	25.3	24.9	25.9	25.3
	Bits4MV	6.7	7.1	2.8	6.9	5.9
MPC	PSNR	23.53	22.43	22.78	29.42	24.54
	Speed Up	0.9	0.9	0.9	0.9	0.9
	Bits4MV	4.6	6.8	2.8	5.9	5.0
DS	PSNR	23.52	22.13	22.89	29.07	24.41
	Speed Up	42.8	44.9	54.7	50.3	48.2
	Bits4MV	3.9	5.8	2.6	4.2	4.1
FBBMA	PSNR	23.71	22.50	22.96	29.45	24.65
	Speed Up	57.5	49.2	61.4	109.8	69.5
	Bits4MV	3.6	5.8	2.6	4.2	4.0

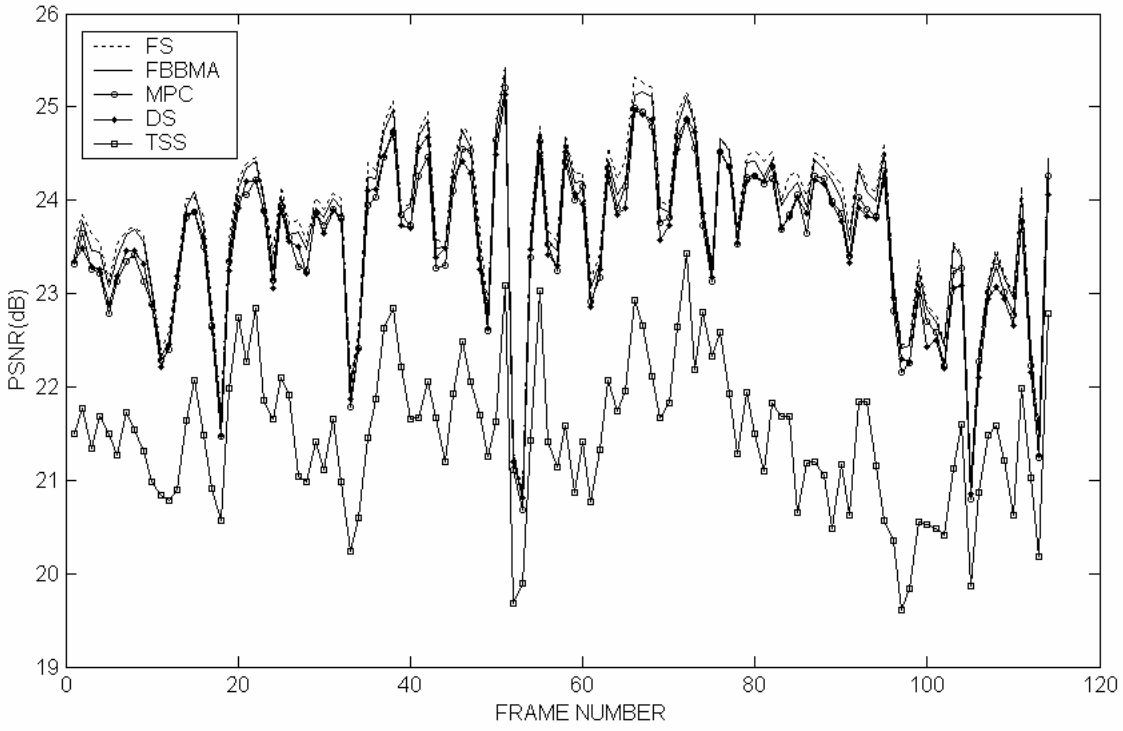

Fig. 11. Per frame PSNR of different ME algorithms for the sequence *Flower Garden*.

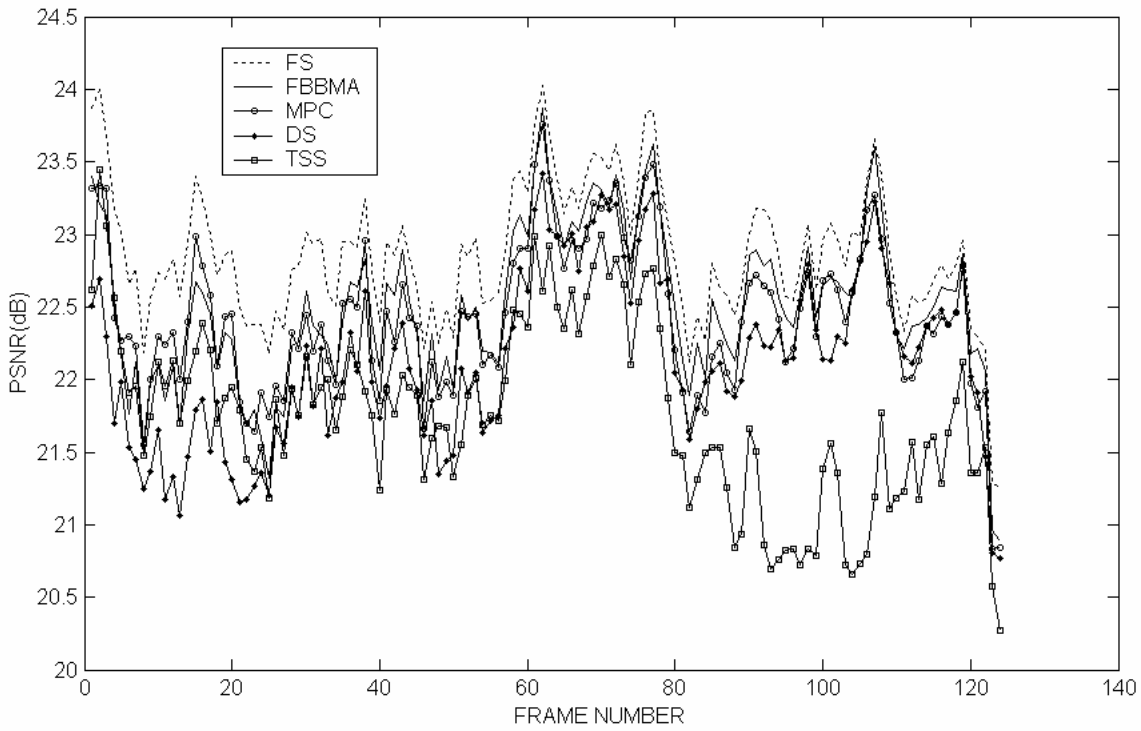

Fig. 12. Per frame PSNR of different ME algorithms for the sequence *Football*.

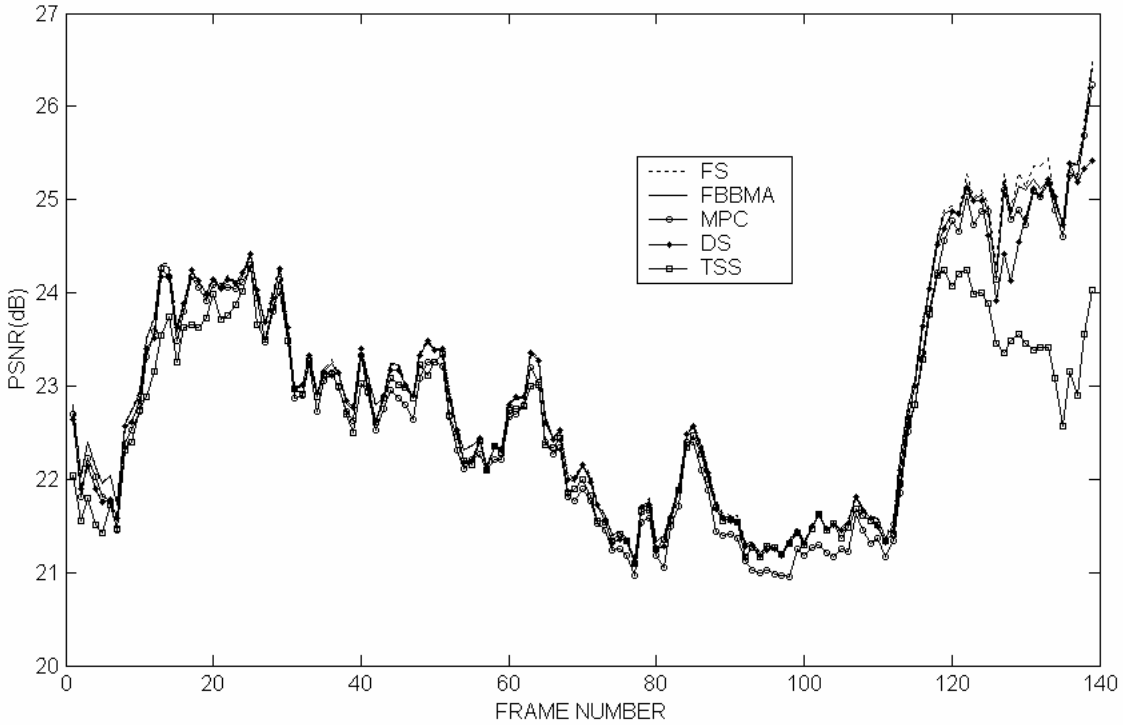

Fig. 13. Per frame PSNR of different ME algorithms for the sequence *Mobile & Calendar*.

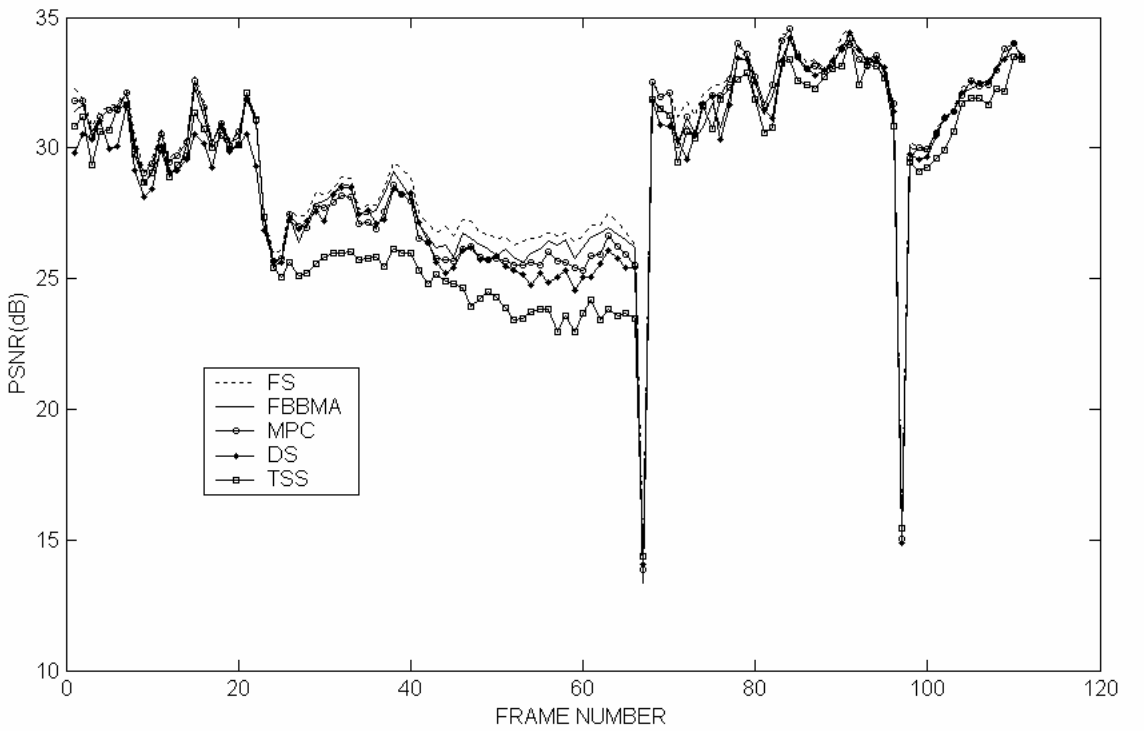

Fig. 14. Per frame PSNR of different ME algorithms for the sequence *Table Tennis*.

Footnotes

1. Footnote of the first page

Manuscript received_____; revised_____.

The authors were with the Department of Electrical Engineering, the Graduate School of Academia Sinica, Beijing 100039, P. R. China.

Ye-Kui Wang is now with Tampere International Center for Signal Processing (TICSP), Tampere University of Technology, FIN-33101, Tampere, Finland.

2. Footnote 1

We have discussed this question with Dr. Vasudev Bhaskaran, one of the authors of [19], through e-mails.