FAST BLOCK MATCHING USING MINKOWSKI'S INEQUALITY

Jim Zong-Chang Lai
*Department of Computer Science and Engineering, National Taiwan Ocean University, Keelung, Taiwan, R.O.C*

Yi-Ching Liaw
*Department of Computer Science and Information Engineering, Nanhua University, Chiayi, Taiwan, R.O.C., ycliw@mail.nhu.edu.tw*

Shih-An Fong
*Information and Communications Research Labs, Industrial Technology Research Institute, Hsinchu, Taiwan, R.O.C.*

Follow this and additional works at: [https://jmstt.ntou.edu.tw/journal](https://jmstt.ntou.edu.tw/journal)

Part of the [Electrical and Computer Engineering Commons](https://jmstt.ntou.edu.tw/journal)

**Recommended Citation**
Lai, Jim Zong-Chang; Liaw, Yi-Ching; and Fong, Shih-An (2010) "FAST BLOCK MATCHING USING MINKOWSKI'S INEQUALITY," *Journal of Marine Science and Technology*: Vol. 18 : Iss. 1 , Article 11. 
DOI: 10.51400/2709-6998.1869
Available at: [https://jmstt.ntou.edu.tw/journal/vol18/iss1/11](https://jmstt.ntou.edu.tw/journal/vol18/iss1/11)

This Research Article is brought to you for free and open access by Journal of Marine Science and Technology. It has been accepted for inclusion in Journal of Marine Science and Technology by an authorized editor of Journal of Marine Science and Technology.
FAST BLOCK MATCHING USING MINKOWSKI’S INEQUALITY

Jim Zong-Chang Lai*, Yi-Ching Liaw**, and Shih-An Fong***

Key words: motion estimation, block matching, sum pyramid.

ABSTRACT

In this paper, we present a fast block matching algorithm by making use of the Minkowski’s inequality. Combined with other fast block matching algorithms, our method can further reduce the computational complexity significantly with little PSNR degradation. Compared to the full search block matching algorithm, our approach can reduce the computing time by a factor of 6.4 to 21.5. Compared to MSEQ, which is a well-known block matching algorithm preserving global optimality, our method can further reduce the corresponding computing time in average by 32.4%. The combination of our algorithm and the predictive search area approach for block motion estimation can achieve the more reduction of computing time with little PSNR degradation.

I. INTRODUCTION

Motion estimation using block matching has been widely used for video coding, such as H.264 and MPEG standards [2, 22] and for video applications [8, 18]. In a block matching algorithm (BMA), the current frame of an image sequence is divided into nonoverlapping blocks of \( N \times N \) pixels. For each template block in the current frame, the best matched block within a search window of size \((2W + 1) \times (2W + 1)\) in the previous frame is determined, where \( W \) is the maximum allowed displacement. The position difference between a template block in the current frame and the best matched block in the previous frame is called the motion vector.

The criterion most commonly used by block matching algorithms is the sum of absolute difference (SAD), which is defined below, between a template block at position \((x, y)\) in the current frame \(I_t\) and the candidate block at position \((x + \hat{u}, y + \hat{v})\) in the previous frame \(I_{t-1}\).

\[
SAD(\hat{u}, \hat{v}) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |g_t(x+i, y+j) - g_{t-1}(x+\hat{u}+i, y+\hat{v}+j)|
\]

where \(g_t(\ldots)\) is the gray value of a pixel in the current frame and \(g_{t-1}(\ldots)\) is the gray level of a pixel in the previous frame. In this paper, the motion vector \((u, v)\) is defined as follows:

\[
(u, v) = \arg \min_{(\hat{u}, \hat{v}) \in S} SAD(\hat{u}, \hat{v})
\]

where \(S = \{(\hat{u}, \hat{v}) \mid -W \leq \hat{u}, \hat{v} \leq W \text{ and } (x + \hat{u}, y + \hat{v}) \text{ is a valid pixel position in } I_{t-1}\}\).

The full search block matching algorithm (FSBMA), which determines the best matched block through calculating the SAD for all positions in the search window, is the simplest among the available block matching algorithms. Although FSBMA can obtain the global optimal result, however it has very intensive computations. To overcome this problem, many fast algorithms are developed [3, 5, 7, 8, 9, 11, 12, 13, 14, 16, 23]. Some well known algorithms are the three-step search (TSS) [14], new three-step search (NTSS) [16], one-dimensional search [3, 11], cross search [7], one-at-a-time search [12, 23], hierarchical search [9, 10], block-based gradient descent search [22], fast search with predictive searching area [5], and multi-resolution block matching algorithm (MRBMA) [13]. Another approach for motion estimation is called the partial-matching error technique, which can obtain the same SAD as the FSBMA [1, 4, 6, 15, 17, 19, 21]. Compared to FSBMA, this kind of approach may reduce the computational complexity significantly. This kind of technique first calculates the complete matching error at the predicted position and uses it as the initial minimum matching error (the initial minimum SAD). For each other search position, some forms of partial matching errors are calculated. The calculation of the complete SAD can be avoided, if the partial error is larger than the minimum SAD. Otherwise, the complete matching error (the complete SAD) is calculated and the minimum SAD is updated if the newly calculated matching error is smaller than the original one.

In this paper, a fast block matching algorithm will be developed. This algorithm uses Minkowski’s inequality [3, 15, 21] to eliminate those impossible candidate blocks, which...
cannot be rejected using available rejecting criteria, and obtain the same SAD as the full search block matching algorithm. Combined with other fast block matching algorithms, such as fast search with predictive searching area [5], our method will reduce the computing time significantly with little PSNR degradation. Our proposed method is different from the method presented in references [15] and [21], which also used the pyramid structure and the Minkowski’s inequality to reject impossible blocks. The major difference is that we use a different rejection criterion to reject impossible blocks and a different way of choosing the initial block. Moreover, the proposed method is also extended to speed up the block searching speed by combining with other fast block matching algorithms.

This paper is organized as follows. We present our concept and algorithm in section II. In section III, the combination of our algorithm and a predictive search area approach for block motion estimation (PSAFBME) [5] is presented. Experimental results are given in section IV and conclusions are presented in section V.

II. BLOCK MATCHING ALGORITHM USING ELIMINATION CRITERIA

In this section, we will present our block matching algorithm and the corresponding elimination criteria. Our method first selects an initial candidate block to calculate its SAD and uses this SAD as the minimum one. A modified version of pyramidal lower bound [15] is developed and is used to reject unlikely candidate blocks.

1. Principles of the Algorithm

Assume that the current frame I, is divided into nonoverlapping blocks of N× N pixels with N = 2^n. For a block X, the pyramid of X can be defined as a set of blocks {X_0, X_1, ..., X_n} with X_0 being a reduced resolution version of X, where m = 1, 2, ..., n. It is noted that X_m has 2^m×2^m pixels. The value of a pixel X_m(k, l) on the (m-1) level can be obtained by 4 pixels X_m+1(k, l), X_m+1(k, l+1), X_m+1(2k+1, 2l), and X_m+1(2k+1, 2l+1) on the m level. That is:

\[
X_{m}^{m+1}(k, l) = \frac{X_{m}^{m+1}(2k, 2l) + X_{m}^{m+1}(2k, 2l + 1)}{2} + \frac{X_{m}^{m+1}(2k + 1, 2l) + X_{m}^{m+1}(2k + 1, 2l + 1)}{2}
\]

The SAD of X_m and Y_m is denoted as SAD_m, which can be calculated by the following equation:

\[
SAD_{X,Y} = \sum_{k=0}^{M-1} \sum_{l=0}^{M-1} |X_{m}(k, l) - Y_{m}(k, l)|
\]

(4)

where M = 2^m, and X_m(k, l) and Y_m(k, l) represent the values of the (k, l)th pixels of X_m and Y_m, respectively. Let

\[
PSAD_{X,Y} (k, l) = |X_{m}^{m+1}(2k, 2l) - Y_{m}^{m+1}(2k, 2l)| + |X_{m}^{m+1}(2k + 1, 2l) - Y_{m}^{m+1}(2k + 1, 2l)| + |X_{m}^{m+1}(2k, 2l + 1) - Y_{m}^{m+1}(2k, 2l + 1)| + |X_{m}^{m+1}(2k + 1, 2l + 1) - Y_{m}^{m+1}(2k + 1, 2l + 1)|
\]

(5)

It is noted that PSAD_m(k, l) represents the sum of absolute difference for 8 pixels on the (m+1)th level, which represents the higher resolution of X_m(k, l) and Y_m(k, l). Let

\[
TPSAD_{X,Y} (kk) = \sum_{l=0}^{M-1} \sum_{l=0}^{M-1} \sum_{l=0}^{M-1} \sum_{l=0}^{M-1} |X_{m}(k, l) - Y_{m}(k, l)|
\]

(6)

and

\[
SAD_{X,Y} = \min_{k, l} \{PSAD_{X,Y} (k, l)\}
\]

(7)

Let MSAD_{X,Y}(kk) be defined by

\[
MSAD_{X,Y}(kk) = TPSAD_{X,Y}(kk) - \sum_{l=0}^{M-1} \sum_{l=0}^{M-1} |X_{m}(k, l) - Y_{m}(k, l)|
\]

(8)

where M = 2^m, k1 is the quotient of kk/2^m, l1 is the remainder of kk/2^m, and 0 ≤ kk < 2^m×2^m. From Minkowski’s inequality, we can easily conclude that

\[
PSAD_{X,Y} (k, l) ≥ |X_{m}(k, l) - Y_{m}(k, l)|
\]

(9)

Let MSAD_{X,Y}(kk) be defined by

\[
MSAD_{X,Y}(kk) = TPSAD_{X,Y}(kk)
\]

(10a)
The above expression can also be represented as

\[ \text{MSAD}^{i}_{X,Y}(i) \geq \text{MSAD}^{i-1}_{X,Y}(i) \]

\[ \geq \text{MSAD}^{i-1}_{X,Y}(i-1), \]

\[ 0 \leq i \leq n, 0 \leq i < 2^{l} \times 2^{l}, \text{and } 0 \leq j < 2^{l-1} \times 2^{l-1} \]  (10b)

With the above results in hand, we can describe our algorithm as follows. Our algorithm first constructs the sum pyramid of the previous frame using the method present in reference [16]. For an image frame of \( W \times H \) pixels, the computational complexity of constructing the sum pyramid is \( 4(W-1)(H-1) \) for \( N = 16 \), where \( N \) is the block size in \( I_{1} \).

For a template block \( X \) in \( I_{1} \), an initial motion vector \((u, v)\) is estimated using the spatial correlation between two frames. The estimation of \((u, v)\) will be discussed in the next subsection. The SAD between block \( X \) and the block with displacement \((u, v)\) in frame \( I_{1} \) is calculated and this value is stored as the minimum SAD, \( \text{SAD}_{\text{min}} \). For other candidate block \( Y \) in frame \( I_{1} \), \( \text{SAD}^{i}_{X,Y}(i) \) is determined first. If \( \text{SAD}^{i}_{X,Y} \) is larger than \( \text{SAD}_{\text{min}} \), the candidate block \( Y \) is rejected; otherwise \( \text{SAD}^{i}_{X,Y} \) is computed. If \( \text{SAD}^{i}_{X,Y} \) is greater than \( \text{SAD}_{\text{min}} \), block \( Y \) is eliminated. Otherwise, \( \text{SAD}^{i}_{X,Y}(0) \) is calculated and check whether it is larger than \( \text{SAD}_{\text{min}} \). If \( \text{SAD}^{i}_{X,Y}(0) \) is larger than \( \text{SAD}_{\text{min}} \), then block \( Y \) is rejected; otherwise, \( \text{MSAD}^{i}_{X,Y}(i) \) is calculated. This process is repeated until block \( Y \) is rejected or the bottom level of the sum pyramid of \( Y \) is reached and \( \text{SAD}_{X,Y} \) is calculated. If \( \text{SAD}_{X,Y} \) is greater than \( \text{SAD}_{\text{min}} \), then block \( Y \) is rejected; otherwise \( \text{SAD}_{\text{min}} \) is replaced by \( \text{SAD}_{X,Y} \). This technique can eliminate many candidate blocks without calculating their SADs.

2. Initial Motion Vector Estimation

The initial motion vector (MV) can be estimated through using spatial and temporal correlations of motion vectors. The idea is to select a set of candidate motion vectors from spatially and/or temporally neighbors and choose the best one as the initial motion vector.

Let \( TB_{i,j} \) be a template block of \( N \times N \) pixels at the \((i, j)\)th position of the current frame \( I_{1} \) and \( SB_{k,l} \) be a candidate block at the \((k, l)\)th position of the previous frame \( I_{1} \). Denote the motion vector of block \( TB_{i,j} \) as \((u, v)_{i,j} \). Let the sum pyramids of \( TB_{i,j} \) and \( SB_{k,l} \) be \( \{TB^{0}_{i,j}, TB^{1}_{i,j}, \ldots, TB^{N}_{i,j}\} \) and \( \{SB^{0}_{k,l}, SB^{1}_{k,l}, \ldots, SB^{N}_{k,l}\} \), respectively, where \( n = \log_{2} N \). The SAD between \( TB^{m}_{i,j} \) and \( SB^{n}_{k,l} \) is denoted as \( \text{SAD}^{m}_{u,v} \), where \( m = 1, 2, \ldots, n \), \( u = k - i \), and \( v = l - j \). Denote the four neighboring blocks of \( TB_{i,j} \) as \( TB_{i,j,m-1} \) \((p = 1 \text{ to } 4) \) as shown in Fig. 1. Let the motion vector of \( TB_{p} \) be \((u_{p}, v_{p})\). Similarly, denote the SAD between \( TB^{m}_{i,j} \) and \( SB^{n}_{k,l} \) as \( \text{MSAD}^{m}_{u,v} \). The estimated motion vector of \( TB_{i,j} \) is represented by \((\hat{u}, \hat{v})_{i,j} \), which is determined as

\[ (\hat{u}, \hat{v})_{i,j} = \arg \min_{(u,v) \in S} \text{SAD}^{i}_{u,v} \]  (11)

where \( S = \{(u_{1}, v_{1}), (u_{2}, v_{2}), (u_{3}, v_{3}), \text{and } (u_{4}, v_{4})\} \).

At this stage, we would like to present our fast block matching algorithm using Minkowski’s inequality (FBMAUMI) as follows.

1. Input a template block \( TB \) in frame \( I_{1} \) and use equation (11) to determine its estimated motion vector \((\hat{u}, \hat{v})\). Set the motion vector \((u, v) = (\hat{u}, \hat{v})\).

2. Use Eq. (1) to calculate \( \text{SAD}(\hat{u}, \hat{v}) \) and let \( \text{SAD}_{\text{min}} = \text{SAD}(\hat{u}, \hat{v}) \).

3. For each block \( SB \) in the search window of size \( (2W + 1) \times (2W + 1) \), use block rejection algorithm to update \( \text{SAD}_{\text{min}} \) and the motion vector \((u, v)\) of \( TB \).

The block rejection algorithm is presented below.

1. Let the motion vector of \( SB \) with respect to \( TB \) be \((u^{*}, v^{*})\) and set \( m = 0 \).

2. Calculate \( \text{SAD}^{m}_{TB,SB} \). If \( \text{SAD}^{m}_{TB,SB} \geq \text{SAD}_{\text{min}} \), rejected the candidate block \( SB \) and leave \((u, v)\) and \( \text{SAD}_{\text{min}} \) unchanged; otherwise set \( \text{MSAD}^{m}_{TB,SB}(-1) = \text{SAD}^{m}_{TB,SB} \).

3. If \( m < n \), where \( n = \log_{2} N \), do the following procedures:

   a. For each \( i = 0 \) to \( (2^{m-1} \times 2^{m-1} - 1) \), calculate \( \text{MSAD}^{m}_{TB,SB}(i) \) using Eq. (9).
   b. If \( \text{MSAD}^{m}_{TB,SB}(i) \geq \text{SAD}_{\text{min}} \), rejected the candidate block \( SB \) and leave \((u, v)\) and \( \text{SAD}_{\text{min}} \) unchanged.

4. Let \((u, v) = (u^{*}, v^{*})\) and \( \text{SAD}_{\text{min}} = \text{MSAD}^{m}_{TB,SB}(2^{m-1} \times 2^{m-1} - 1) \).

III. OMBINATION OF OUR APPROACH WITH PSAFBME

The proposed fast motion block matching algorithm can be combined with other fast algorithms to achieve further speedup. However, the guarantee of global minimum solution may be sacrificed. In this paper, we use the algorithm of predicting search area for block motion estimation [5] as an example. The predictive search area approach for block area motion estimation (PSAFBME) generates a search area \( SA \) for a template block \( TB_{i,j} \) in the current frame \( I_{1} \). It is noted that \( SA \) in general is much smaller than the full search window of size \( (2W + 1) \times (2W + 1) \). \( SA \) can be represented by

\[ SA = SA_{1} \cup SA_{2} \cup SA_{3} \cup SA_{4} \]  (12)
Where \( S_{Ak} \) \((k = 1 - 4)\) denotes the subsearch area of size \((2D + 1) \times (2D + 1)\) with center at \((i + u_k, j + v_k)\), where \((u_k, v_k)\) is the motion vector of the block \(TB_k \) \((k = 1 - 4)\) as shown in Fig. 1. It is noted that \(TB_k\) is the \(k\)th neighboring block of \(TB_{ij}\). \(D = 2\) or \(3\) may be used to obtain little image degradation. The combination of our approach with PSAFBME is referred to as MPSAFBME in this paper. Compared to PSAFBME, MPSAFBME has the much less computational complexity. Now, we would like to present the modified predictive search area for block motion estimation (MPSAFBME) as follows.

1. Input a template block \(TB\) in frame \(I\) and use Eq. (11) to determine its estimated motion vector \((\hat{u}, \hat{v})\). Set the motion vector \((u, v) = (\hat{u}, \hat{v})\).
2. Use Eq. (1) to calculate \(SAD(\hat{u}, \hat{v})\) and let \(SAD_{\text{min}} = SAD(\hat{u}, \hat{v})\).
3. For each block \(SB\) in the search area \(SA\), use block rejection algorithm to update \(SAD_{\text{min}}\) and the motion vector \((u, v)\).
4. Output \(SAD_{\text{min}}\) and the motion vector \((u, v)\) of \(TB\).

IV. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed algorithm, six sets of video sequences (as shown in Fig. 2) are used and several motion block estimation algorithms are implemented. The first 50 frames in six video sequences are used in the simulation. Each image of the above mentioned video sequences is of size \(352 \times 240\).

All computing is performed on an Intel Core 2 Duo 6600 2.4 GHz PC with 1 GB of memory. The full search block matching algorithm (FSBMA), the partial distortion elimination (PDE) [1], the successive elimination algorithm (SEA) [17], the multilevel successive elimination algorithm (MSEA) with three levels [6], the winner-update algorithm with the lower bound derived from Minkowski’s inequality (WinUpMI) [4], and our proposed method (FBMAUMI) provide the global optimality. It is noted that MSEA with three levels obtains the same results as references [15] and [21] in our implementation. MPSAFBME (presented in section III), PSAFBME [5], TSS [14], and BBGDS (block-based gradient descent search) [20] sacrifice motion estimation accuracy to gain the reduction of computational complexity. The global optimality is not guaranteed by these four algorithms.

The PSNR (peak signal-to-noise ratio) and computing time are used to evaluate the performances of block matching algorithms. In this paper, PSNR is defined by

\[
\text{PSNR} = 10 \log_{10} \left( \frac{255^2}{\text{MSE}} \right) \tag{13}
\]

where MSE is the mean square error between the original and estimated image frames. In this paper, all the template and candidate blocks are of size \(16 \times 16\). The size of search window is \(33 \times 33\). Tables 1 and 2 show the PSNRs and computing time of six video sequences for the FSBMA, PDE, SEA, MSEA, WinUpMI, and FBMAUMI methods. From Tables 1 and 2, we can find that all algorithms obtain the same PSNR for each video sequence and our proposed method FBMAUMI performs better than the other methods. Comparing to FSBMA, FBMAUMI can reduce the computing time by a factor of 6.4 to 21.5. Compared with MSEA, which is the second best, our proposed method FBMAUMI can reduce the computing time in average by 32.4%.

Tables 3 and 4 give the PSNRs and computing time for MPSAFBME with \(D = 2\), PSAFBME with \(D = 2\), TSS, and BBGDS. From Tables 2 and 4, we can find that any one of these four algorithms obtains the lower PSNR than that of FSBMA or FBMAUMI, since these four methods sacrifice motion estimation accuracy to gain the reduction of computational complexity. From Table 3, we can find that MPSAFBME has the highest PSNR. Compared with BBGDS, MPSAFBME with \(D = 2\) gives the higher PSNR with about the same computing time. From Tables 1 to 4, we may conclude that compared to FBMAUMI, MPSAFBME with \(D = 2\) can reduce the computing time by a factor of about 2.86 with little PSNR degradation. Compared with PSAFBME, our proposed method can further reduce the computing time by 55.0% to 69.5%.

V. CONCLUSIONS

In this paper, we develop a fast block matching algorithm using Minkowski’s inequality (FBMAUMI) to speed up motion estimation. Combining PSAFBME with our method, we...
**Table 1. PSNR comparison for six methods preserving global optimality.**

<table>
<thead>
<tr>
<th>Video Sequence</th>
<th>Method</th>
<th>FSBMA</th>
<th>PDE</th>
<th>SEA</th>
<th>MSEA</th>
<th>WinUpMI</th>
<th>FBSMAUMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile</td>
<td>FSBMA</td>
<td>23.34</td>
<td>23.34</td>
<td>23.34</td>
<td>23.34</td>
<td>23.34</td>
<td>23.34</td>
</tr>
<tr>
<td>Football</td>
<td>FSBMA</td>
<td>22.79</td>
<td>22.79</td>
<td>22.79</td>
<td>22.79</td>
<td>22.79</td>
<td>22.79</td>
</tr>
<tr>
<td>Garden</td>
<td>FSBMA</td>
<td>23.78</td>
<td>23.78</td>
<td>23.78</td>
<td>23.78</td>
<td>23.78</td>
<td>23.78</td>
</tr>
<tr>
<td>Salesman</td>
<td>FSBMA</td>
<td>35.40</td>
<td>35.40</td>
<td>35.40</td>
<td>35.40</td>
<td>35.40</td>
<td>35.40</td>
</tr>
<tr>
<td>Susie</td>
<td>FSBMA</td>
<td>35.96</td>
<td>35.96</td>
<td>35.96</td>
<td>35.96</td>
<td>35.96</td>
<td>35.96</td>
</tr>
<tr>
<td>Tennis</td>
<td>FSBMA</td>
<td>29.07</td>
<td>29.07</td>
<td>29.07</td>
<td>29.07</td>
<td>29.07</td>
<td>29.07</td>
</tr>
</tbody>
</table>

Table 2. Computing time (in seconds) comparison for six methods preserving global optimality.

<table>
<thead>
<tr>
<th>Video Sequence</th>
<th>Method</th>
<th>FSBMA</th>
<th>PDE</th>
<th>SEA</th>
<th>MSEA</th>
<th>WinUpMI</th>
<th>FBSMAUMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile</td>
<td>FSBMA</td>
<td>10.67</td>
<td>6.641</td>
<td>4.281</td>
<td>0.703</td>
<td>0.811</td>
<td>0.564</td>
</tr>
<tr>
<td>Football</td>
<td>FSBMA</td>
<td>10.827</td>
<td>7.297</td>
<td>6.455</td>
<td>1.735</td>
<td>1.438</td>
<td>0.969</td>
</tr>
<tr>
<td>Garden</td>
<td>FSBMA</td>
<td>10.660</td>
<td>6.652</td>
<td>4.421</td>
<td>0.922</td>
<td>0.952</td>
<td>0.581</td>
</tr>
<tr>
<td>Salesman</td>
<td>FSBMA</td>
<td>10.718</td>
<td>5.812</td>
<td>2.779</td>
<td>0.466</td>
<td>0.516</td>
<td>0.499</td>
</tr>
<tr>
<td>Susie</td>
<td>FSBMA</td>
<td>10.703</td>
<td>6.110</td>
<td>3.096</td>
<td>1.014</td>
<td>1.156</td>
<td>0.669</td>
</tr>
<tr>
<td>Tennis</td>
<td>FSBMA</td>
<td>10.685</td>
<td>7.423</td>
<td>7.406</td>
<td>2.484</td>
<td>2.982</td>
<td>1.671</td>
</tr>
<tr>
<td>Average</td>
<td>FSBMA</td>
<td>10.713</td>
<td>6.641</td>
<td>4.740</td>
<td>1.221</td>
<td>1.309</td>
<td>0.826</td>
</tr>
</tbody>
</table>

Table 3. PSNR comparison for four methods not preserving global optimality.

<table>
<thead>
<tr>
<th>Video Sequence</th>
<th>Method</th>
<th>TSS</th>
<th>PSAFBME</th>
<th>BBGDS</th>
<th>MPSAFBME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile</td>
<td>FSBMA</td>
<td>23.05</td>
<td>23.33</td>
<td>23.33</td>
<td>23.33</td>
</tr>
<tr>
<td>Football</td>
<td>FSBMA</td>
<td>22.01</td>
<td>21.66</td>
<td>21.55</td>
<td>21.66</td>
</tr>
<tr>
<td>Garden</td>
<td>FSBMA</td>
<td>21.80</td>
<td>23.66</td>
<td>23.42</td>
<td>23.66</td>
</tr>
<tr>
<td>Salesman</td>
<td>FSBMA</td>
<td>35.14</td>
<td>35.37</td>
<td>35.37</td>
<td>35.37</td>
</tr>
<tr>
<td>Susie</td>
<td>FSBMA</td>
<td>34.90</td>
<td>35.89</td>
<td>35.78</td>
<td>35.89</td>
</tr>
<tr>
<td>Tennis</td>
<td>FSBMA</td>
<td>27.58</td>
<td>27.87</td>
<td>27.73</td>
<td>27.87</td>
</tr>
<tr>
<td>Average PSNR</td>
<td>FSBMA</td>
<td>27.41</td>
<td>27.96</td>
<td>27.87</td>
<td>27.96</td>
</tr>
</tbody>
</table>

can reduce computing time significantly with little PSNR degradation. Compared to FSBMA, our proposed method (FBMAUMI) can reduce the computing time by a factor of 6.4 to 21.5. Compared to the well-know multilevel successive elimination algorithm preserving global optimality (MSEA), our algorithm can further reduce the computing time in average by 32.4%. MPSFBME with $D = 2$ reduces the computing time of FSBMA by 96.7% to 98.0% with the average PSNR degradation of 0.43 dB.

**REFERENCES**


