Subblock sum matching algorithm for block-based interframe coding

S.S. Kim, J.S. Kim, H.W. Park*

Department of Information and Communication Engineering, Korea Advanced Institute of Science and Technology, 207-43, Cheongryangri, Dongdaemun-gu, Seoul 130-012, South Korea

Received 28 August 1997

Abstract

A fast block-matching algorithm for motion estimation is described for interframe image coding. Utilizing the fact that the motion is searched by a macroblock basis while the compression is performed by a block basis in many video compression standards, the proposed algorithm produces high PSNR and fast computation while keeping the bit-rate as low as those of conventional fast algorithms. The proposed algorithm uses the subblock sum as a matching parameter and the reversed square sum as a matching criterion. The new matching parameter and criterion reduce the total number of computations for block matching and also keep the quality of the decompressed image. The computational complexity and the compression performance of the proposed algorithm are compared with those of other block-matching algorithms. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Block-matching algorithm; Subblock sum (SBS); Bit-rate; Macroblock; Reverse squared sum (RSS)

1. Introduction

There has been large demand for digital video in recent years, for example, multimedia and high definition television (HDTV). Due to the huge amount of digital video data, however, compression algorithms are critical to the viability of digital video services in terms of storage as well as communication. The goal of the video-coding algorithms is to remove redundant information and to reduce the data rate. Interframe coding is one of the most powerful video compression techniques and can reduce the temporal redundancy in natural video sequences [3]. In the interframe coding, motion estimation is used to reduce temporal redundancy. The block-matching algorithm for motion estimation is widely used in many video compression standards because of its simplicity [7]. However, most of the encoding time is consumed in calculating the block matching. Therefore, many algorithms have been proposed to reduce the computation time for block matching [1–6].

In many video compression standards, the motion vector is searched with a macroblock basis (typically 16 pixels × 16 pixels), while the compression of the motion-compensated macroblock is performed by using a discrete cosine transform (DCT)
with a block basis (typically 8 pixels × 8 pixels). Most of the well-known block-matching algorithms do not take into account the fact of motion estimation with macroblock basis and DCT with block basis. The motion vector, which is obtained in a macroblock sense, does not necessarily have high compression performance in block-based DCT coding. The objective of this paper is to propose a new fast block-matching algorithm which utilizes this fact to attain a reduced resulting bit-rate and a fast computation. After a brief summary of a few block-matching algorithms in Section 2, the proposed algorithm and its simulation result are described in Sections 3 and 4, respectively.

2. Block-matching motion estimation

Block-matching algorithm for motion estimation of interframe coding segments an image into the fixed size of a small square block and it is assumed that each of these blocks is undergoing an independent translation. If these blocks are small enough, movement of larger objects can be closely approximated by piecewise translation of these small blocks. A motion vector is defined as the translation in 2-D image, which gives the smallest difference between the current macroblock and the reference macroblock as shown in Fig. 1. Various measures, including the mean-square error (MSE) and the mean absolute difference (MAD), can be used in the matching criterion.

Extensive computations are required to evaluate the MSE or the MAD for the best match. In order to reduce the computation requirements, it is possible to bound the maximum displacement so that the matching process can be performed within the bounded area which is called the search window. If a maximum displacement of ± $w$ pixels is allowed for both the $x$ and the $y$ directions, the search window includes $(2w + 1)^2$ locations. The number of operations, which is required to calculate the measures for this limited search area, however, is still too high to be realized in real time. Several algorithms have been proposed to reduce the computation time further [1–6]. Some of these algorithms are described in this section.

2.1. Partial area search algorithm

Because it takes a very long time to calculate the block-matching measures for all the vectors in the search window, several algorithms, which do not severely degrade the image quality, have been proposed to estimate the measures for vectors partially selected in the search window [2,3,5]. These techniques assume a monotonically increasing MAD around the location of the optimal vector. The MAD, however, often has several local minima in which these algorithms can be trapped. In these techniques, one of these local minima can be selected as a motion vector, which degrades image quality. Typical partial area search algorithms are logarithmic search, three-step search, and cross search.

2.2. Partial pixel comparison algorithm

Instead of limiting the number of vectors in the search window, the number of pixels with which the matching process is performed can be reduced by subsampling every other pixel in the horizontal and the vertical directions for fast computation [6]. To increase the accuracy of the motion vector, several subsampling patterns, which have the same subsampling period, but different pixel locations, can be used. For each subsampling pattern, a motion vector is obtained that minimizes the MAD over the locations where the subsampling pattern is used. Then a vector which has minimum MAD...
is selected as the optimum motion vector for the
block from the several motion vectors correspond-
ing to the subsampling patterns.

2.3. Hierarchical search algorithm

The basic idea of the hierarchical methods is to
predict an approximated motion vector in an image
with coarse resolution and to refine, in successive
searching processes, the predicted vector to obtain
the final vector [1]. The number of displacement
candidates decreases, and the approximated
matching criteria becomes accurate with successive
refinement. At the first layer, for example, a full
search is performed using the matching criterion of
the absolute difference of the mean value in a block
of \( N \times N \) for all the candidates to be searched. Since
the matching criterion is too coarse to estimate the
displacement vector, several candidates within
a certain range of the matching error are selected
to avoid the ill-posed problem. At the next layer,
the search is performed only for the candidate
positions selected at the previous layer and uses
a more refined matching criterion such as the
absolute difference of the mean value in a block
of \( N/2 \times N/2 \). By repeating this procedure, the
candidates are refined with better accuracy,
and finally a single position is selected at the last
layer.

2.4. Feature-based algorithm

This algorithm uses the line integral projections
as an image feature and reduces the computation
time for block matching [4]. The line integral pro-
jections are defined as the sums of the gray levels
along a given direction in the image. The one-
dimensional quantities obtained by summing the
gray levels of an image along the vertical and the
horizontal directions are called the vertical and
the horizontal projections, respectively. The hori-
zontal projection \( H(n) \) and vertical projection \( V(m) \)
are defined as follows:

\[
H(n) = \sum_{m \in B} f(n, m), \quad V(m) = \sum_{n \in B} f(n, m),
\]

where \( f(n, m) \) is a 2-D image and \( B \) is a block for
motion estimation. These horizontal and vertical
projections of a block are used for motion estima-
tion so that total number of computations can be
reduced.

3. Proposed algorithm

3.1. Background

Most block-matching algorithms for motion es-
timation are focused on finding optimum motion
vectors as fast as possible with less degradation of
the image quality. In many video compression stan-
dards, the motion vector is searched by a macro-
block (16 \( \times \) 16 pixels) basis whereas the prediction
error is compressed by a block (8 \( \times \) 8 pixels) based
DCT. Most motion estimation algorithms do not
take into account this fact. The proposed algorithm
improves the compression performance and the
computation simplicity by utilizing the above
fact.

For example, let us assume that the mean abso-
lute difference (MAD) between the current macro-
block \( M \) and the previous 16 \( \times \) 16 blocks \( M_1 \) and
\( M_2 \), which correspond to the motion vectors \( m_1 \) and
\( m_2 \), respectively, are shown in Fig. 2. In both
cases, the MADs for the macroblock of the motion
vectors \( m_1 \) and \( m_2 \) are the same and have a value
of 12. However, it is assumed that \( M_1 \) is a better
estimate of macroblock \( M \) than \( M_2 \) in the sense of
the compression rate because block 1 in case 1 re-
quires no coding whereas block 1 in case 2 does.
Even though the MAD sums of other blocks in
case 1 are a little larger than those in case 2,
the compressed bit-rates of case 1 can be less
than those of case 2 without any cost in image
quality.

The above matching property can be applied not
only to the case when the MAD sum of one of the
four blocks is near zero, but also to the case when
the MAD sum of one of the four blocks is far less
than that of the other blocks. The block whose
MAD sum is near zero is not coded. The block with
far less MAD may require the error term to be
coded, but with far fewer bits. Therefore, if we are
able to select the motion vector for which some of
the blocks in the macroblock have far less MADs than the blocks, it is possible to reduce the bit-rate.

In conjunction with the above idea, a fast algorithm is proposed for finding a motion vector with which a higher compression rate is obtained without degrading the image quality. The following is a detailed description of the proposed fast algorithm.

3.2. Proposed algorithm

In the proposed algorithm, a macroblock is divided into four blocks. Each of which is divided into four subblocks again as shown in Fig. 3. A ‘Block’ has the same meaning as in the MPEG standard, and its size is 8 × 8 pixels. The subblock has 4 × 4 pixels, which is the basic element for block matching in the proposed fast algorithm.

The proposed algorithm defines a subblock sum (SBS) of the subblock \( S_m \) for a matching parameter, and the SBS is defined as follows:

\[
SBS(m) = \sum_{(i,j) \in S_m} f(i,j),
\]

where \( f(i,j) \) is the gray level of the pixel located at \( (i,j) \). The \( SBS(m) \) is the sum of the gray levels of all the pixels in the subblock \( S_m \). The SBSs of the subblocks in the considered macroblock of the current frame are compared with those of the reference frame. The sum of the differences of the SBSs in a block is named the subblock sum difference (SBSD). The SBSD of block \( B_n \) is defined as follows:

\[
SBSD_{(x,y)}(n) = \sum_{S_m \in B_n} |SBS_{S_m}(m) - SBS_{r,(x,y)}(m)|,
\]

where \( SBS_{S_m}(m) \) is the SBS of subblock \( S_m \) in block \( B_n \) of the current frame, and \( SBS_{r,(x,y)}(m) \) is the SBS of subblock \( S_m \) in block \( B_n \) of the reference frame. In Eq. (3), \( (x, y) \) is a 2-D coordinate in the search window, and corresponds to a motion vector. To find the optimum motion vector which matches with current macroblock from the viewpoint of the compression rate, such as case 1 in Fig. 2, the following matching criterion, which is the reversed square sum (RSS), is defined:

\[
RSS(x,y) = \sum_{n=1}^{4} (SBSD_{\text{max}} - SBSD_{(x,y)}(n))^2,
\]

where \( SBSD_{\text{max}} \) is the maximum possible SBSD and \( SBSD_{(x,y)}(n) \) is the SBSD of block \( B_n \) in the macroblock. If each pixel has 8-bit resolution and the number of subblocks in a block is 16, the \( SBSD_{\text{max}} \) becomes 4080 \(( = 255 \times 16 \) ). In Eq. (4), the optimum motion vector is the vector \( (x, y) \) for which the matching criterion \( RSS(x, y) \) is largest in the search window.

Even though \( RSS(0,0) \) is not the largest \( RSS \) in the search window, the motion vector \( (0,0) \) will still
be the best one for bit reduction when variable length coding of the motion vector is used, provided $\text{RSS}(0, 0)$ is larger than some threshold value because the least number of bits is allocated to the motion vector $(0, 0)$ in variable length coding. Utilizing this fact and the above functions, the proposed fast algorithm searches for the motion vectors using the following procedure:

**Step 1.** Determine the subblock sums $\text{SBS}_m(m)$ ($m = 1, \ldots, 16$) for macroblock in the current frame.

**Step 2.** Determine the subblock sums $\text{SBS}_{m+4}(0,0)(m)$ ($m = 1, \ldots, 16$) for the macroblock corresponding to the motion vector $(0, 0)$ in the reference frame.

**Step 3.** Using the SBSs obtained in steps 1 and 2, calculate $\text{SBSD}_{0,0}(n)$ ($n = 1, \ldots, 4$) by using Eq. (3).

**Step 4.** Calculate the matching criterion $\text{RSS}(0, 0)$ using the $\text{SBSD}_{0,0}(n)$ obtained in step 3. If it is larger than the predetermined threshold, the vector $(0, 0)$ will be selected as the motion vector for this macroblock, and the motion estimation process will be finished for the macroblock. Otherwise, go to step 5.

**Step 5.** With a motion vector $(x, y)$ in the search window, steps 2, 3 and 4 are sequentially processed except that $\text{RSS}(x, y)$ is compared to the predetermined threshold. This step is repeated for all vectors $(x, y)$ in the search window.

**Step 6.** The vector $(x, y)$ that has the maximum $\text{RSS}$ will be selected as the motion vector for the macroblock.

### 3.3. Computational complexity

To calculate the SBS of a subblock, 15 additions are required. Since there are 16 subblocks in a macroblock, we need $240 (= 15 \times 16)$ additions to calculate the SBSs of the macroblock. Therefore, $240 \times (2w + 1)^2$ additions are totally required for the calculation of the SBSs of the macroblocks within the search window which has $(2w + 1)^2$ pixels. However, there are several subblocks which are also included in different macroblocks as shown in Fig. 4. For example, the 5th subblock of the macroblock $m$ is the same as the 1st subblock of macroblock $m + 4$, the 9th subblock of macroblock $m$ is the same as the 5th subblock of macroblock $m + 4$, and so on. Therefore, total number of additions can be reduced. Since the total number of subblocks within a search window is $(2w + 13)^2$ as shown in Fig. 5, only $15 \times (2w + 13)^2$ additions are necessary for calculating the SBSs in a search window.

Finally, for the calculation of the matching criterion of Eq. (4), four SBSDs are obtained for each macroblock by using Eq. (3); this requires a total of $4 \times 4$ subtractions, $4 \times 4$ absolute calculations, and $4 \times 3$ additions per macroblock. These SBSDs are then subtracted from $\text{SBSD}_{\text{max}}$, and the subtracted result is squared and accumulated to obtain the matching criterion $\text{RSS}$ as given in Eq. (4), which requires a total of $4$ subtractions, $4$ squares, and $3$ additions per macroblock.

The above analysis of the computation requirement is summarized in Table 1. Note that this number of computations can decrease when the RSS at vector $(0, 0)$ is larger than the predetermined threshold. The reduction of the number of computations depends on the characteristics of the video sequence. If the video sequence has a large background still area, it is more likely that the calculation time can be reduced significantly.

Table 2 shows the number of computations required by several algorithms to determine the motion vector of a macroblock when the search window is $15 \times 15$, i.e., $w = 7$. In this case, the...
computation time of the proposed algorithm is about six times less than that of the full-search algorithm and one and half times less than that of the feature-based algorithm. Table 3 shows the number of computations required by several algorithms to determine the motion vector of a macroblock when the search window is $31 \times 31$, i.e., $w = 15$. In this case, the computation time of the proposed algorithm is about eight times less than that of the full-search algorithm and almost one and half times less than that of the feature-based algorithm, but about three times more than that of the three-step search algorithm. In Tables 2 and 3, the number of computations for the proposed algorithm decreases as the threshold value decreases. The relation between the number of computations and the threshold value is described in the following section.

4. Simulation results

4.1. Performance measure

For the performance measure of the proposed algorithm, the peak signal-to-noise ratio (PSNR), which is defined as follows, is used:

$$\text{PSNR} = -10 \log_{10} \frac{\sum_{n=0}^{N-1} \sum_{m=0}^{M-1} (f(n, m) - f_c(n, m))^2}{(N \times M) / 255^2},$$

(5)

Table 1
Computational complexity (number of computations) of the proposed algorithm for limited displacement $w$

<table>
<thead>
<tr>
<th>Calculation of SBSs of the macroblock in consideration</th>
<th>Additions $240$</th>
<th>Absolutes $15 \times (2w + 13)^2$</th>
<th>Squares $35 \times (2w + 1)^2 + 240$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculation of SBSs in the search area</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Calculation of the cost function</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2
Comparison of the number of computations for motion vector estimation of a macroblock by using several algorithms when $w = 7$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Additions</th>
<th>Absolutes</th>
<th>Squares</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full search</td>
<td>114 975</td>
<td>57 600</td>
<td></td>
</tr>
<tr>
<td>Three-step search</td>
<td>13 797</td>
<td>6 912</td>
<td></td>
</tr>
<tr>
<td>Feature-based search</td>
<td>28 155</td>
<td>7 200</td>
<td></td>
</tr>
<tr>
<td>The proposed</td>
<td>19 050</td>
<td>3 600</td>
<td>900</td>
</tr>
</tbody>
</table>

Table 3
Comparison of the number of computations for motion vector estimation of a macroblock by using several algorithms when $w = 15$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Additions</th>
<th>Absolutes</th>
<th>Squares</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full search</td>
<td>491 071</td>
<td>246 016</td>
<td></td>
</tr>
<tr>
<td>Three-step search</td>
<td>18 396</td>
<td>9 216</td>
<td></td>
</tr>
<tr>
<td>Feature-based search</td>
<td>92 763</td>
<td>30 752</td>
<td></td>
</tr>
<tr>
<td>The proposed</td>
<td>61 610</td>
<td>15 376</td>
<td>3844</td>
</tr>
</tbody>
</table>
where the size of the image is $N \times M$, $f(n, m)$ is the original image, and $f_3(n, m)$ is the reconstructed image using the proposed motion estimation. In Eq. (5), pixel depth is 8 bits, i.e., the maximum gray level is 255. Of course, the PSNR may not give an exact measure of the image quality from a subjective viewpoint. A lower PSNR does not necessarily mean a low-quality image in the sense of human visual perception. However, the PSNR is usually used as a typical image-quality measure in image compression since it can give a quantitative criterion.

4.2. Determination of the threshold value

Performance of the proposed algorithm depends somewhat on the threshold value for the zero vector decision in step 4 of the proposed algorithm. If the threshold value decreases, the number of zero motion vector increases so that the computing time can also be reduced, but the reconstructed image can be degraded. Therefore, the threshold value should be determined in consideration of both image quality and the number of computations.

![PSNR analysis of the proposed algorithms for different threshold values. The numbers in every point are PSNR. The threshold in the horizontal axis is given in percentage from the maximum RSS.](image1)

![The number of computations of the proposed algorithms for different threshold values. The vertical axis shows the number of computations in terms of percentage of the required computations in comparison with 100% threshold as shown in Table 1.](image2)
Fig. 8. PSNR comparison of the various block-matching algorithms, such as the full search, the three-step search, the feature-based and the proposed algorithm: (a) Mother and Daughter sequence with QCIF (Average bit-rate = 1890 per frame); (b) Akiyo sequence with QCIF (Average bit-rate = 830 per frame); (c) News sequence with CIF (Average bit-rate = 9400 per frame); (d) Foreman sequence with CIF (Average bit-rate = 30000 per frame).
To determine the optimum threshold value, the number of computations and the PSNR of the compressed sequences were analyzed using several threshold values. The simulation results are shown in Figs. 6 and 7. Four video sequences were used for this simulation, each with 300 frames and QCIF (176 x 144 pixels/frame) or CIF (352 x 288 pixels/frame). These were well-known sequences for video compression, such as Akiyo (QCIF), News (CIF), Foreman (CIF), and Mother & Daughter (QCIF). In these figures, the threshold values were defined as a percentage from the maximum RSS, which is \(4 \times (255 \times 16 \times 4)^2\) in this simulation. Fig. 6 shows the average PSNR of four sequences with respect to different threshold values. According to this simulation, 99% of the threshold gives optimum PSNR as shown in Fig. 6. In Fig. 7, the number of computations for different threshold values are described in terms of percentage of the required computations in comparison with the 100% of the threshold value as shown in Table 1.

### 4.3. Comparison to other algorithms

The average bit-rate and PSNR of the compressed sequences were analyzed using four different block-matching algorithms: the full-search algorithm, the feature-based algorithm, the three-step search algorithm, and the proposed algorithm. Four sequences were used as test sequences: Akiyo (QCIF), Mother & Daughter (QCIF), News (CIF), and Foreman (CIF), each with 300 frames. In the simulation, the compression method of ITU H.263 was employed. The frame rate was 10 Hz, i.e., 100 frames were selected at every third frames from the 300-frame video sequences. Only the first frame was I-frame, and the following frames are all P-frames which were motion-estimated from the previous frames.

Fig. 8 shows the PSNR comparisons with the same average bit-rates, for each frame of News, Mother and Daughter, Foreman, and Akiyo. In Fig. 8, PSNRs of 50 frames are displayed, which were samples of every other frames from the simulation results of 100 frames. Fig. 9 shows the average bit-rate versus the average PSNR for each sequence. The threshold value for the proposed algorithm was set to 99% since it provides the best PSNR as shown in Fig. 6, and less computation requirements than 100% threshold as shown in Fig. 7.

### 5. Conclusion

In this paper, a new fast block-matching algorithm for a higher compression rate was proposed. The idea of the new algorithm is based on the fact that, in many video compression standards such as MPEG1, MPEG2 and H.26x, motion estimation is performed with a 16 x 16 macroblock basis whereas
Fig. 9. PSNR analysis for various bit-rates per frame; (a) Mother and Daughter sequence with QCIF; (b) Akiyo sequence with QCIF; (c) News sequence with CIF; (d) Foreman sequence with CIF.

an 8 × 8 block-based DCT is performed for compression of the motion-compensated macroblock. The subblock sum (SBS) is introduced to find the motion vector as fast as possible without any severe degradation of the reconstructed image quality. The reversed squared sum (RSS) is also introduced as a matching criterion to reduce the compressed bit-rate and computation complexity.

The PSNR of the proposed algorithm was better than that of the feature-based algorithm, far better than that of the three-step search algorithm, though it was a little worse than that of the full-search method. The computation time of the proposed algorithm was about six times less than that of full-search method and one and half times less than that of the feature-based algorithm, though one and half times more than that of the three-step search algorithm when the search window $w$ is 7.

A disadvantage of the proposed algorithm is its weakness in finding motion vectors in images with busy texture patterns. Because it uses the 4 × 4 subblock sums as a basic element of the matching parameter, it is inherently difficult to find exact motion in a texture area where the motion vector is more likely to have similar subblock sums even though the patterns are not similar. This kind of weakness also exists in other fast algorithms where partial characteristics of the macroblock are measured for block matching. However, most of the natural images have high spatial correlation so that the 4 × 4 subblock sums can fully represent the characteristics of the macroblocks.
References


