ABSTRACT
A merging procedure for variable-size block motion estimation is proposed to reduce the computation for block-size decision. A smaller block-size is initially used for motion estimation. An adaptive threshold, which depends on the information obtained from the motion estimation, quantization parameter, and rate distortion cost function is used to determine if the motion vectors of the neighboring blocks should be merged or not. We simulate the proposed merging procedure using an H.264 video encoder to determine if 16x16, 16x8, 8x16, or 8x8 block-size should be used for each macroblock. Based on the simulations, performance of the proposed method is close to that of H.264 JM2.1 when 16x16, 16x8, 8x16 and 8x8 block modes are enabled and the exhaustive search method is used to determine the block-sizes. Computational complexity can be significantly reduced by the proposed algorithm since it only performs the motion search for one block-size.

1. INTRODUCTION
Motion estimation (ME) and compensation are critical components for digital video coding systems. Fixed-size block-based ME has been widely adopted by international standards such as MPEG-1, 2, 4 and H.261/H.263. Recently, variable-sizes block-matching (VSBM) based ME has been adopted by ITU-T H.264 [1]. Although VSBM can achieve significant video quality improvement, it requires high computational complexity for determining the best partition of a macroblock for ME. The exhaustive-search method needs to perform ME with all the block-sizes defined in the standard and choose the best one which minimizes a cost function. To reduce the complexity of VSBM, [2] proposed a “top-down” approach in which initially large blocks are matched. If the SAD (Sum of Absolute Difference) of the best match is above a predefined threshold, then that block is split into four smaller blocks. Rhee in [3] proposed a “bottom-up” VSBM algorithm which relies more on local motion information. A set of “candidate” motion vectors of each fixed-size small block is first obtained by full-search, whose matching error is less than a prescribed threshold. Neighboring blocks are then merged in a quadtree manner if they have at least one vector in common. In our work, we also use the motion vectors of smaller blocks and then compare them to decide if the macroblock should be merged or not. However, quantization parameter and Lagrange cost function are taken into consideration in determining the threshold for comparing the distance between the motion vectors of the two blocks. To illustrate the performance of the algorithm, we perform simulations with the 8x8 block-size as the initial block-size for the motion estimation in H.264, since when only the 8x8 mode is enabled, the performance is acceptable, while when only the 4x4 mode is enabled, the performance degradation is much more significant. The information of the motion-search results is used in the merge procedure (merged as 16x16, 16x8, 8x16, or 8x8 blocks). Simulations results show that it can achieve performance close to that of the H.264 JM2.1 video coding system when the 16x16, 16x8, 8x16, and 8x8 four block-modes are enabled and the exhaustive search is used. Furthermore, based on the 8x8 search results, split procedures can be applied to areas with complex motion to further improve the performance when smaller block-sizes (e.g. 4x4, 4x8, 8x4) are allowed.

2. VSBME USING MERGING PROCEDURE
In the “top-down” algorithm, if one larger block needs to split, one should perform motion estimation for those split smaller blocks to find best motion vectors of them to achieve higher performance. Therefore, there is trade off between complexity and performance. One can determine a smaller threshold to achieve better performance at the sacrifice that more larger blocks need to split to smaller blocks, and the computational complexity will be higher. On the other hand, in the “bottom-up” VSBM algorithm as in [3], if the threshold is
smaller, reserved candidate motion vectors in the set for each 4x4 block may be very close to each other around the global minimum and they may not represent true motion of that block. On the contrary, if the threshold is larger, many candidate motion vectors are reserved and overhead of comparison of each set of each small block should be considered. Therefore, we use constraint (1) below to decide if the two small blocks (block a and block b) should be merged into a large one

\[ \text{Dist}(x_a, y_b) \leq TH, \]  

(1)

where \( \text{Dist}(x_a, y_b) \) is the distance between motion vectors \( y_a \) and \( y_b \) of the blocks a and b.

In an H.264 video encoder, although the smallest block-size supported is 4x4, we still chose the 8x8 block for motion estimation initially, because from Fig. 2., it can be seen that the performance of this fixed-size block motion estimation of H.264 is acceptable. Furthermore, though best motion vectors of 4x4 blocks can result in small matching error, they are not suitable since they may not represent true motions. In our scheme, the full-search algorithm is first used to find the motion vectors with the best costs of the four 8x8 blocks in one 16x16 macroblock. Then merging process is applied. When the difference between the motion vectors of two neighboring 8x8 blocks a and b is not larger than \( TH \), blocks a and b are merged to a larger one (i.e., if \( \text{Dist}(x_a, y_b) \leq TH \), Merge(a, b) = 1).

Fig. 1 shows the partition of a 16x16 macroblock. The merging process can be performed as follows:

1. if (Merge(a, b) == 1 && Merge(c, d) == 1)  
   16x8 mode is used
2. if (Merge(a, c) == 1 && Merge(b, d) == 1)  
   8x16 mode is used.
3. if (1) and (2) are true,  
   16x16 mode is used.

When 8x8 blocks are merged into a 16x16 block, the motion vector \( MV_m \) for this merged block is obtained from the original motion vectors of the four 8x8 blocks. When the quantization parameter is large, we use the motion vector of the 8x8 block which results in the minimum motion vector rate. When the quantization parameter is small, we use the motion vector of the 8x8 block which results in minimum prediction error to represent the motion vector \( MV_m \). \( MV_m \) can be refined by searching the surrounding positions in sub-pel accuracies (quarter-pel accuracy). If the total cost after the motion estimation and the merging procedure is smaller than the intra prediction mode, the final motion vector differences and the quantized transform coefficients of the motion-compensated residuals are sent to the UVLC entropy coder to produce the H.264 bitstream.

Performance of our proposed scheme depends on an accurate and robust threshold. After performing the merging procedure, the number of motion vectors which affects the number of bits for representing the motion vectors will be reduced. However, the prediction errors of these blocks (or, the overall prediction errors of the merged block) will increase so as the bit-rate of residuals after quantization and entropy coding. Therefore, when determining \( TH \), the change of the total cost (distortion and bit-rate) should be taken into account in order to achieve improved performance. Generally speaking, for a larger homogeneous area, larger block-sizes may give better overall performance. On the other hand, for an area with detailed textures or edges, small blocks may be more suitable. Moreover, the larger the quantization parameter (QP), the more important the bit-rate of motion vectors, while the prediction error is dominant when a smaller QP is applied. In the next section, we describe an adaptive threshold with above considerations.

3. DETERMINATION OF THE THRESHOLD

It is intuitive that for an area with uniform motion or texture, it is better to use a larger block in the motion search. On the other hand, for an area with complex motion or texture, keeping the search results of smaller blocks will be better. Also, when determining the threshold \( TH \), the rate-distortion (R-D) cost function \( J \) should be taken into account in order to get better results.

The cost function \( J \) of a block \( i \) after motion estimation can be expressed as

\[ J_i = SAD_i + \lambda_{\text{motion}} \cdot R(MVD_i), \]  

(2)

where \( \lambda_{\text{motion}} \) is the Lagrange multiplier for motion search and \( R(MVD_i) \) is the number of bits of coding the motion vector difference between the motion vector \( MV_i \) and the predicted motion vector. Let \( J_a \) and \( J_b \) are the R-D cost functions of blocks a and b, respectively. The overall cost function \( J_m \) after the merging procedure should satisfy the constraint (3) blow to assure performance improvement:

\[ J_m \leq J_a + J_b. \]  

(3)

Since

\[ J_m = SAD_m + \lambda_{\text{motion}} \cdot R(MVD_m), \]  

(4)

where \( SAD_m \) is the matching error of the merged block, from (2) and (3), the prediction error of the merged block, \( SAD_m \), should satisfy

\[ SAD_m \leq J_a + J_b - \lambda_{\text{motion}} \cdot R(MVD_m) \]  

\[ = SAD_a + SAD_b + \lambda_{\text{motion}} \cdot \Delta R(MVD) \]  

(5)
where
\[
\Delta R(MVD) = R(MVD_a) + R(MVD_b) - R(MVD_m)
\]  
\tag{6}

is the decrement of the number of bits for the motion vectors after the merging procedure is performed. If small blocks are used for motion estimation and then merging procedure is applied, the increment of SAD and the decrease of motion vector rate \(\Delta R(MVD)\) are needed to check to see condition (5) is satisfied or not. We approximate the function of the prediction error \(S\) at different search positions around the best matching location as
\[
S(d) = a_2d^2 + a_1d + a_0,
\]  
\tag{7}

where \(d\) is the distance between a search position and the best position, to obtain the approximate increment of SAD. Since motion estimation in the R-D sense is performed in the H.264 video encoder, coefficients \(a_i\)’s are affected by different quantization parameters. We use the curve fitting method to find these \(a_i\)’s for experiments. Then the increment of SAD\(_a\) or SAD\(_b\) can be obtained in a small range around their best positions, and the decrement of the motion vector bits in (5) can be used to determine the threshold \(TH\), which can be expressed as
\[
TH = Th_0 + C,
\]  
\tag{8}

where \(C\) is a constant and
\[
Th_j = \max(Th_a, Th_b),
\]  
\tag{9}

\[
Th_j = S^{-1}(\max(Z(QP), SAD_m) - SAD_j),
\]  
\tag{10}

\(j = a, b\)

Here, a value \(Z(QP)\) is introduced to indicate that if the prediction error is smaller than \(Z(QP)\), the bit-rate of the quantized coefficients of the residuals is very low or even zero, and
\[
Z(QP) = \frac{(5/6) \cdot 2^{15+QP/6}}{K \cdot Q(QP\%6,0,0)},
\]  
\tag{11}

depends on the quantization process described in the H.264 document, \(K\) is a constant, and \(Q\) is the quantization matrix defined in the H.264 document. With the discussions above and the merging procedure described in section 2, some experimental results are shown in the next section.

4. SIMULATION RESULTS AND DISCUSSIONS

H.264 reference software JM2.1 is used to implement our algorithm. The motion vectors with integer-pel accuracy of the four 8x8 blocks in a MB obtained by full-search in

the rate-distortion sense are used to perform the merging process. 16x16, 16x8 and 8x16 are the possible merged block-sizes. We use MOTHER AND DAUGHTER, FOREMAN and TABLE TENNIS in QCIF (176x144) as the test video sequences. The R-D performance of our merging process is shown in Fig. 3. TABLE I lists the Bit Rate/PSNR simulation results. It can be seen that the R-D curve of the proposed method is close to that of H.264 (when 16x16, 16x8, 8x16 and 8x8 modes are enabled) especially when the test sequence contains smaller motion such as the MOTHER AND DAUGHTER sequence. The performance of other sequences with more complex motion or texture such as FOREMAN or TABLE TENNIS degrades slightly. Although at lower bit-rates, when only the 8x8 mode is enabled, the performance is much lower than that when only the 16x16 mode is enabled, the proposed merging procedure can improve the performance close to the optima. The proposed scheme also achieves higher performance at high bit-rates than that of the original 8x8 mode. It also can be seen that the proposed method outperforms the bottom-up algorithm (as in [3], but merge from the 8x8 block-size to 8x16, 16x8, or 16x16 block-sizes instead), especially at lower bit-rates. For the computational complexity of our proposed approach, only one block-size (8x8) is used in the motion estimation, therefore, the computation load of the motion search for other block-sizes can be saved achieving significant computation reduction. The complexity of the merging procedure is very low compared to the motion estimation.

5. REFERENCES


II - 791

Fig. 1 Partition of a 16x16 block into four 8x8 blocks.
Fig. 2. Performance of H.264 (FOREMAN sequence).

(a) FOREMAN sequence

(b) MOTHER AND DAUGHTER sequence

Table 1: Bit Rate-PSNR Performance Comparison

<table>
<thead>
<tr>
<th>Seq</th>
<th>Proposed</th>
<th>Original*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sq.1</td>
<td>QP = 0</td>
<td>1429.81/48.04</td>
</tr>
<tr>
<td></td>
<td>QP = 6</td>
<td>674.12/43.38</td>
</tr>
<tr>
<td></td>
<td>QP = 12</td>
<td>292.89/38.73</td>
</tr>
<tr>
<td></td>
<td>QP = 18</td>
<td>131.33/34.56</td>
</tr>
<tr>
<td></td>
<td>QP = 24</td>
<td>64.48/30.80</td>
</tr>
<tr>
<td></td>
<td>QP = 31</td>
<td>34.05/26.93</td>
</tr>
<tr>
<td>Sq.2</td>
<td>QP = 0</td>
<td>637.49/48.84</td>
</tr>
<tr>
<td></td>
<td>QP = 6</td>
<td>248.97/44.58</td>
</tr>
<tr>
<td></td>
<td>QP = 12</td>
<td>91.08/40.11</td>
</tr>
<tr>
<td></td>
<td>QP = 18</td>
<td>37.14/36.43</td>
</tr>
<tr>
<td></td>
<td>QP = 24</td>
<td>17.37/33.31</td>
</tr>
<tr>
<td></td>
<td>QP = 31</td>
<td>9.05/30.58</td>
</tr>
<tr>
<td>Sq.3</td>
<td>QP = 0</td>
<td>1109.29/48.17</td>
</tr>
<tr>
<td></td>
<td>QP = 6</td>
<td>533.20/43.30</td>
</tr>
<tr>
<td></td>
<td>QP = 12</td>
<td>245.86/38.57</td>
</tr>
<tr>
<td></td>
<td>QP = 18</td>
<td>110.83/34.65</td>
</tr>
<tr>
<td></td>
<td>QP = 24</td>
<td>51.24/30.84</td>
</tr>
<tr>
<td></td>
<td>QP = 31</td>
<td>27.01/26.90</td>
</tr>
</tbody>
</table>

Sq.1: FOREMAN,  Sq. 3: TABLE TENNIS, Sq.2: MOTHER AND DAUGHTER

*Original H.264 with 16x16, 16x8, 8x16 and 8x8 modes enabled.