Fast Integer Motion Estimation for H.264 Video Coding Standard

Andy Chang*, Peter H. W. Wong**, Y.M. Yeung†, Oscar C. Au††
Department of Electrical and Electronic Engineering
The Hong Kong University of Science and Technology
Email: eecax@ust.hk*, peterwong@ieee.org**, eeyym@ust.hk†, eeau@ust.hk††

Abstract

Multiple Block Size Motion Estimation is adopted in the latest JVT/H.264 video coding standard to achieve higher coding efficiency. However, full exhaustive search of all block sizes is computational intensive with motion estimation complexity increase linearly with the number of block size allowed. In this paper, we propose a fast integer motion estimation method for multiple block size motion estimation (FMBME) which reduces computation significantly. Experimental result shows that, compared with full search, the proposed method can have a speed up factor of four with bit-rate increases within one percent.

1. Introduction

In video coding standard such as ITU H.261/263, ISO MPEG-1, 2 and 4, block-based motion estimation and compensation (ME/MC) technique is widely used to remove the temporal redundancy within the video sequence to achieve higher compression efficiency. In block-based motion estimation, the current frame will be divided into disjoint macroblocks. For each macroblock in the current frame, the encoder will search for a best match in the previous frame(s) as a predictor of the macroblock in the current frame. The predicted macroblock will be subtracted from the current macroblock to form the residual block. The residual block and the motion vector information from the motion estimation process will be encoded for transmission or storage.

In previous coding standard such as MPEG-1/2, the macroblock size in the ME process is usually 16x16 pixels and one motion vector is used to describe the relative displacement of a macroblock. In order to make a better prediction, the latest H.264 allows dividing the 16x16 macroblock into smaller block size for ME. This can reduce the residue energy with a trade-off of using more motion vectors to describe the motion of different regions within the macroblock. In H.264, the encoder support seven various block size

Figure 1 – Different Mode in dividing a macroblock in H.264

ME with a tree-structured hierarchical macroblock partitions as shown in Figure 1. The processing time increases linearly with the number of block type used. This is because motion estimation needs to be performed for each block type. This full searching process (the examination of all seven block modes) provides the best coding result but the increase in computation is very high. Simulation results show that using seven different block sizes can save more than 15% of bit rate when compared to using 16x16 block size only.

In this paper, we propose a fast multi-block integer motion estimation focusing on 16x16 (mode 1), 16x8 (mode 2), 8x16 (mode 3) and 8x8 (mode 4) which can efficiently reduce the computational cost while achieving similar visual quality and bit-rate. It is a bottom-up approach meaning that motion estimation will be processed on mode 4 (8x8) first then proceeds to mode 1 and finally mode 2 and mode 3.

The paper will be organized as follows. A review of multi-block motion estimation will be given in Section 2. Section 3 describes the observation on fast multi-block motion estimation algorithm follow by a description of the proposed method in Section 4. Experimental result will be presented in Section 5. Conclusion will be given in Section 6.

2. Review of multi-block motion estimation

Multiple block sizes ME/MC is adopted in H.264 to reduce the residue energy between the original image and the predicted image by increasing the accuracy of
prediction. In a macroblock, it is possible to contain more than one object and these objects may not move in the same direction. With this, one motion vector is not good enough to describe the motion of all objects in this macroblock. With only one motion vector, only part of the macroblock will be well described and the resulting residue energy can still be large due to the mismatch in the remaining part of the macroblock.

If multi-block motion estimation is allowed, the macroblock will be segmented into smaller blocks. Each of them will get a motion vector pointing to the best matched block in the preceding pictures. In H.264, seven types of block size with different shapes are supported (Figure 1). The best match is found by minimizing the cost function:

\[
J(m, \lambda_{\text{MOTION}}) = \text{SAD}(s,c(m)) + \lambda_{\text{MOTION}} \cdot R(m - p) \tag{1}
\]

with \( m = (m_x, m_y) \) denotes the motion vector, \( p = (p_x, p_y) \) denotes the prediction for the motion vector and \( \lambda_{\text{MOTION}} \) denotes the Lagrange multiplier. The term \( R(m - p) \) represents the bits used to encode the motion information and are obtained by table-lookup. The SAD (Sum of Absolute Differences) is computed as:

\[
\text{SAD}(s,c(m)) = \sum_{x=0}^{B-1} \sum_{y=0}^{B-1} |s[x,y] - c[x-m_x, y-m_y]| \tag{2}
\]

with \( B = 16, 8 \) or 4, \( s \) and \( c \) are the current frame and the decoded previous frame respectively.

3. Observation on FMBME

In the 8x8 ME process (mode 4), for each sub-block shown in figure 1 we can obtain the optimal motion vector and SAD value. We found that there exists a high correlation between 8x8 ME information and the optimal motion vector for larger block sizes, i.e. 16x16, 8x16 and 16x8 block sizes.

Table 1 shows the hit-rate when the integer motion vector as well as the sub-pel motion vector of 8x8 sub-block 0, 1, 2 and 3 and the 16x16 optimal integer and sub-pel motion vector are exactly the same. We can see that the hit-rate is very high which indicate that 8x8 motion vectors are very good predictor for 16x16 ME.

<table>
<thead>
<tr>
<th></th>
<th>Integer MV</th>
<th>Sub-pel MV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman QCIF</td>
<td>93%</td>
<td>76.6%</td>
</tr>
<tr>
<td>Stefan QCIF</td>
<td>90%</td>
<td>82.6%</td>
</tr>
</tbody>
</table>

Table 1 – Hit-rate of 8x8 integer and sub-pel motion vector equals 16x16 optimal integer and sub-pel motion vector

Figure 2a shows the distribution of the motion vector difference between the best 8x8 sub-block 0 integer motion vector obtained from 8x8 ME process and the optimal integer motion vector obtained from 16x16 ME process. The testing sequence “Foreman” with QCIF format is used in the experiments. The percentage of 16x16 is not shown in this figure in the ease to view the surrounding distribution. We can see that the distance between 8x8 and 16x16 motion vector are very close to each other, if a local search is added to the 8x8 motion vector it is very likely that we can obtain the optimal motion vector for 16x16 block size. Figure 2b shows the distribution of the difference between 8x8 sub-block 1 integer motion vector and 16x16 optimal motion vector which have a similar distribution as shown in figure 2a.

We observed that the relationship between optimal vectors of mode 3 (8x16) and 8x8 motion vectors is similar to mode 1. However, for mode 2(16x8) there is
some problem in directly using 8x8 motion vectors as the predictor for the left or right sub-block in mode 2.

\[
\begin{array}{cc}
UP & UR \\
\hline
a & b \\
\hline
c & d \\
\end{array}
\]

Figure 3 – Example of motion vector prediction in H.264

In H.264, for each sub-block in different modes a predicted motion vector is calculated based on the surrounding motion vector information. This motion vector predictor will act as the search center of the current sub-block. The optimal motion vector obtained after ME will be subtracted from this motion vector predictor to get the motion vector difference. The motion vector difference will be encoded and sent to the decoder. In H.264, the predictor for 8x8 ME is done using median prediction. In figure 3, predictors for 8x8 motion vectors \( \text{MV}_a, \text{MV}_b, \text{MV}_c \) and \( \text{MV}_d \) are obtained like this:

\[
\begin{align*}
\text{pMV}_a &= \text{median} (\text{MV}_{UP}, \text{MV}_{UR}, \text{MV}_{LF}) \\
\text{pMV}_b &= \text{median} (\text{MV}_{UP}, \text{MV}_{UR}, \text{MV}_a) \\
\text{pMV}_c &= \text{median} (\text{MV}_a, \text{MV}_b, \text{MV}_{LF}) \\
\text{pMV}_d &= \text{median} (\text{MV}_a, \text{MV}_b, \text{MV}_c)
\end{align*}
\]

However, the predictor for mode 2 (16x8) and mode 3 (8x16) are obtained in a different way. Instead of using median prediction, H.264 makes use of the directional segmentation prediction to get the motion vector predictor for the current sub-block. For example, in figure 4a, the left-block in mode 3 will use \( \text{MV}_{LF} \) as the predictor and the right-block will use \( \text{MV}_{UR} \). Similarly the top-block in mode 2 (figure 4b) will use \( \text{MV}_{UP} \) as the predictor and the bottom-block will use \( \text{MV}_{UR} \).

In the situation where the current macroblock should be segmented horizontally, that means the upper portion and lower portion of the macroblock may move in a different direction, the accuracy of 8x8 \( \text{MV}_c \) and \( \text{MV}_d \) may be decreased. When we look at the predictors for \( \text{MV}_c \) and \( \text{MV}_d \) we can see that \( \text{pMV}_c \) and \( \text{pMV}_d \) is dominated by \( \text{MV}_a \) and \( \text{MV}_b \) using median prediction. Since \( \text{MV}_a \) and \( \text{MV}_b \) are referring to a different object, the accuracy for \( \text{pMV}_c \) and \( \text{pMV}_d \) will be affected especially when the motion difference between upper and lower part of the macroblock is very large. As a result, the accuracy of \( \text{MV}_c \) and \( \text{MV}_d \) may be affected and they are not suitable to predict the motion vector for the lower quarter of macroblock in mode 2. We found that this situation can be helped by including the predictor for mode 2 in our algorithm.

4. Algorithm

In the proposed fast integer motion estimation, full search is first performed for 8x8 block sizes. Each 8x8 motion vector (in quarter pixel accuracy) will be rounded to integer motion vector and used as the initial search point for mode 1, 2 and 3.

For mode 1, the SAD value for integer motion vector \( \text{MV}_a, \text{MV}_b, \text{MV}_c \) and the default median predictor will be computed in the first step. The motion estimation will be performed around the eight neighboring locations of the motion vector obtained in first step with minimum SAD value. The searching step of mode 2 and mode 3 are similar to mode 1 except that the top sub-block of mode 2 will make use of \( \text{MV}_a, \text{MV}_b \) and median predictor whereas the bottom sub-block will use \( \text{MV}_c, \text{MV}_d \) and median predictor. Similarly, motion vector of the left sub-block in mode 3 will be predicted use of \( \text{MV}_a, \text{MV}_c \) and median predictor whereas the right sub-block will use \( \text{MV}_b, \text{MV}_d \) and median predictor.

5. Simulations and Results

The proposed fast motion estimation algorithm was implemented in the reference JVT software version 7.3 [2]. We have tested the proposed method over a series of testing sequence with different resolution. The results of four QCIF (176x144) testing sequences “Akiyo”, “Coastguard”, “Foreman”, and “Stefan” are shown in this paper. In the simulation, the sequences are encoded at 30fps with different \( q_p \), the \( q_p \) ranges from 10 to 36 with step size of two. The PSNR, total
bitrate for P-frames and complexity comparison between FMBME and full search is shown in Table 2. The complexity comparison is based on the time used for integer motion estimation. The proposed FMBME can reduce computational cost by 69.7% on average (equivalent complexity of performing motion estimation on 1.2 block types instead of 4 block types) with negligibly small PSNR degradation (0.005dB) and slight increase in bit rate (0.04%).

6. Conclusion

In this paper, a fast integer motion estimation for multiple block size motion estimation is proposed for H.264 video coding. Experiment results show that the proposed method can reduce the computation cost significantly with negligible change to PSNR and bit rate.

7. Acknowledgement

This work was supported by Research Grants Council of Hong Kong SAR (HKUST6203/02E).

References


Table 2 – PSNR and Bitrate Comparison between the proposed FMBME and FS with QP = 14 to 36

(a) Akiyo QCIF (b) Coastguard QCIF (c) Stefan QCIF (d) Foreman QCIF

<table>
<thead>
<tr>
<th>QP</th>
<th>Foreman QCIF Full Search</th>
<th>QP</th>
<th>Foreman QCIF FMBME</th>
<th>QP</th>
<th>Akiyo QCIF Full Search</th>
<th>QP</th>
<th>Akiyo QCIF FMBME</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>46.5 9132152 15.14</td>
<td>46.5 9145768 43.98</td>
<td>0.01</td>
<td>0.15</td>
<td>-65.57</td>
<td>14</td>
<td>46.13 18805000 17.08</td>
</tr>
<tr>
<td>16</td>
<td>44.09 7228912 14.39</td>
<td>44.09 7230304 42.77</td>
<td>0.02</td>
<td>0.02</td>
<td>-66.16</td>
<td>16</td>
<td>44.47 1600832 17.07</td>
</tr>
<tr>
<td>18</td>
<td>43.11 5442992 14.78</td>
<td>43.1 5455400 43.99</td>
<td>0.01</td>
<td>0.23</td>
<td>-66.99</td>
<td>18</td>
<td>42.58 13145048 17.27</td>
</tr>
<tr>
<td>20</td>
<td>41.52 4115176 14.33</td>
<td>41.52 411928 43.46</td>
<td>0.25</td>
<td>0.25</td>
<td>-67.01</td>
<td>20</td>
<td>40.08 10862968 16.94</td>
</tr>
<tr>
<td>22</td>
<td>40.03 321168 14.91</td>
<td>40.03 3228888 45.14</td>
<td>0.23</td>
<td>-66.97</td>
<td>22</td>
<td>39.29 9499992 16.91</td>
<td>39.3 949942 52.435</td>
</tr>
<tr>
<td>24</td>
<td>38.33 2360040 14.72</td>
<td>38.33 2364376 45.44</td>
<td>0.18</td>
<td>-66.99</td>
<td>24</td>
<td>37.35 7002320 16.34</td>
<td>37.35 7002064 51.902</td>
</tr>
<tr>
<td>26</td>
<td>36.86 1764948 14.53</td>
<td>36.85 176440 46.04</td>
<td>0.01</td>
<td>-68.43</td>
<td>26</td>
<td>35.64 515693 16.18</td>
<td>35.65 515080 51.877</td>
</tr>
<tr>
<td>28</td>
<td>35.33 133302 14.42</td>
<td>35.51 133320 46.81</td>
<td>-0.01</td>
<td>-69.18</td>
<td>28</td>
<td>33.94 4275176 16.00</td>
<td>33.95 426352 52.003</td>
</tr>
<tr>
<td>30</td>
<td>34.01 1001072 13.84</td>
<td>34.01 999664 46.79</td>
<td>-0.14</td>
<td>0.37</td>
<td>-70.43</td>
<td>30</td>
<td>32.06 831920 15.70</td>
</tr>
<tr>
<td>32</td>
<td>32.58 763248 13.66</td>
<td>32.61 759040 47.54</td>
<td>0.03</td>
<td>-71.28</td>
<td>32</td>
<td>30.32 234624 15.19</td>
<td>30.35 2348624 52.558</td>
</tr>
<tr>
<td>34</td>
<td>31.33 599776 12.77</td>
<td>31.35 597168 46.99</td>
<td>-0.24</td>
<td>-72.81</td>
<td>34</td>
<td>28.78 1762120 15.58</td>
<td>28.8 1757304 52.736</td>
</tr>
<tr>
<td>36</td>
<td>29.93 460192 12.51</td>
<td>29.95 460928 47.25</td>
<td>-0.02</td>
<td>-73.33</td>
<td>36</td>
<td>27.15 1285888 15.03</td>
<td>27.15 1285888 54.274</td>
</tr>
</tbody>
</table>

Average: -0.01 | -0.04 | -68.79%

(c) (d)