Complexity Reduction in Block Mode Determination for H.264

Peng Liu\(^1\), Tomonori Aikawa\(^1\), Satoshi Miyaji\(^2\), Liang Zhao\(^1\) and Hideo Yamamoto\(^1\)

\(^1\)Faculty of Engineering, Utsunomiya University,
7-1-2 Yoto, Utsunomiya-shi, Tochigi, 321-8585, Japan
E-mail: bblp@degas.is.utsunomiya-u.ac.jp
\(^2\)KDDI R&D Laboratories,
2-1-15 Ohara, Kamifukuoka-shi, Saitama, 356-8502, Japan
E-mail: miyaji@kddilabs.jp

Abstract: This paper considers the “variable block size” feature in H.264. Based on experimental observations, we give a simple algorithm to reduce the coding complexity. To get the coding efficiency and reduce the complexity at the same time, it is applied only to blocks that have small estimated cost value. We show that the proposed algorithm can reduce the coding time by up to 40% with the same picture quality and only 10% increase of the bitrate.

1. Introduction

H.264, also known as MPEG-4 part 10 (Advanced Video Coding), is a new video codec standard released in 2003 by the Joint Video Team (JVT) of ITU-T and ISO. To obtain a high coding efficiency, a number of new features are adopted in H.264 [1]. Among all of them, variable block size motion compensation (MC) is considered one of the most effective one, which can significantly increase the efficiency than fixed block size adopted in former codecs.

On the other hand, however, variable block size MC is also time expensive. Actually, in our test with the reference software JM 7.3 [2], it averagely takes 10 seconds to code one CIF-size frame (or 25 days for a two hours video) with a PC equipped with a 2.4GHz Pentium 4 processor and 256MB memory. This is because that the JM tries to find the best mode from all alternatives by exhausted computations.

In this paper, we consider the “variable block size” feature. Based on experimental observations, we show an algorithm to reduce the coding complexity in block mode determination. Our simple scheme is to apply the same mode as used by a reference block. To get the coding efficiency and reduce the complexity at the same time, we employ a threshold \(T\) and apply the scheme only to blocks that have costs (with respect to a pre-defined function) \(T\) or less. We show that it can reduce the coding time by up to 40% with the same picture quality and only 10% increase of the bitrate.

The rest of this paper is organized as follows. In Section 2, we discuss the block mode and its relationship with the rate-distortion (R-D) cost function used in JM 7.3. In Section 3, we present the proposed algorithm and show experiment results in Section 4. Finally we conclude in Section 5.

2. Block Mode and R-D Cost

When considering a P-frame, there are two modes, intra and inter, to code a 16x16 macroblock in H.264 basically. In intra mode, there are two sub-modes: 116x16 that is to code the whole macroblock in intra mode, and 14x4 that is to divide the macroblock equally into sixteen 4x4 blocks and code them individually by intra prediction. Similarly, in inter mode, there is a so-called copy mode, which is to copy the whole macroblock from the reference frame without MC.

On the other hand, sub-(intra)modes with MC in H.264 can be performed by using various block sizes and shapes, which are illustrated in Figure 1. Hence there are a total of 262 modes for coding a 16x16 macroblock: 2 intra modes, 1 copy mode, 3 inter modes with size larger than 8x8, and 256 inter modes with size 8x8 or smaller (since each 8x8 block can be coded by one of the four modes). They are called the block modes (also called as macroblock partition and sub-partition in [3]).

![Variable block size in H.264.](image)

Hence a smaller block size can result a better-matched block, so it can reduce the prediction residual. On the other hand, it also increases the number of motion vectors (MVs). To get the best coding efficiency, the H.264 reference software JM tries to find a best block mode from all modes by calculating their R-D costs. In H.264, bitrate is denoted by rate (R), while picture quality is evaluated by distortion (D) [4]. Thus the smaller R and D are the better. A function (called the rate-distortion function, or R-D function simply) showing the relation between R and D is used in JM, and the R-D optimization is performed to find a best mode.

In coding a P frame, the R-D cost is first calculated to determine the reference frame, then is calculated to choose the best block mode. For choosing the best block mode, the cost is defined as \(D + \lambda \Delta R\), where \(\lambda\) is defined by

\[
\lambda = 0.85 \times 2^{\frac{QP}{285}}.
\]

where \(QP\) is the quantization parameter.

3. The Proposed Algorithm

From experiment results, we observe that, usually there is a lot of blocks use the same modes as in the previous frame (see Section 4). This is the basic idea of proposed algorithm.
3.1. Algorithm in a glance

The flow of our algorithm is shown in Figure 2. Speaking simply, it is to apply the same mode (called the reference mode in this paper) used in the block (called the reference block) that is at the same position in the previous frame, if the estimated cost (will be explained in the next subsection) is less than a pre-defined threshold.

Notice that, by applying the reference mode, the video quality does not change significantly since we use the same QP (unless the mode changes between inter and intra).

The bitrate may increase since the reference mode may not be the best for the current block. To find whether the previous mode is good enough, we first calculate a modified R-D cost by coding the current block in the reference mode, and then compare it with the cost of the reference block. If the absolute difference is less than a threshold $T$, we say that the reference mode is good. Thus the larger $T$ is, the less complexity and the worse efficiency. In this way, we can get a tradeoff between coding complexity and efficiency.

3.2. Modified R-D cost: vector comparison

We have seen that R and D are two main indexes to evaluate a coding block mode. In our algorithm, the “cost comparison” is critical to determine if the reference mode is good enough. Recall that in JM, a scalar value of $R + \lambda D$ is used as the cost. In the proposed algorithm, for more accuracy, we consider to use the vector $(R, D)$ comparison. In particular, we use the Euclidean distance

$$\sqrt{(R_c - R_r)^2 + (\lambda D_c - \lambda D_r)^2}$$

(2)

to compare the costs, where $R_c$ and $D_c$ denote the values of the current block, and $R_r$ and $D_r$ denote the values of the reference block, respectively.

3.3. Block mode determination in detail

Now we explain the block mode determination procedure of proposed algorithm in detail. In this paper, we only consider the P-frames. Suppose that there are $P_n$ ($n = 1, 2, \ldots$) P-frames.

$P_1$ is processed normally as the same as JM, which is shown in Figure 3, where we use $M_c$ denotes the block mode that is used to encode current block.

For the frame $P_n$ ($n = 2, 3, \ldots$), we first calculate the modified cost $e$ by (2) in reference mode. Next, we compare $e$ and the threshold $T$. If $e$ is smaller than $T$, the reference mode will be applied. See the details in Figure 4. On the other hand, of course, intra prediction and the MV are computed newly except the copy mode.

4. Experiments for Performance

4.1. Environment of experiments

In the experiments, we used the next ten widely-known video clips in two sizes of QCIF and CIF: akiyo, news, foreman, tempe, mobile & calendar (mobile), mother & daughter (mother), coastguard, container, hall, and silent. We used the H.264 reference software JM 7.3.

The test hardware is a PC equipped by a 1.8GHz Pentium 4 processor and 512MB memory. The OS is Windows® 2000 with the Visual Studio .NET 2003 development environment. Quantization parameter is fixed in the test. Main parameters for coding are shown in Table 1.

<table>
<thead>
<tr>
<th>Table 1. Main parameters used in experiments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>Number of the frames</td>
</tr>
<tr>
<td>Frame sequence</td>
</tr>
<tr>
<td>Video format</td>
</tr>
<tr>
<td>Number of I-frame</td>
</tr>
<tr>
<td>The QP of the I-frame</td>
</tr>
<tr>
<td>The QP of the inter frame</td>
</tr>
<tr>
<td>ME search area</td>
</tr>
<tr>
<td>Number of reference frames</td>
</tr>
</tbody>
</table>
4.2. Preliminary experiment

First we have to determine the threshold $T$. For this, we observe the block modes determined by the default JM algorithm, and then compare the two modes used for the blocks at the same location of two successive P frames.

The comparisons for two typical videos, akiyo, container and mobile, are shown in Figures 5-7 (other comparisons are omitted due to space limit). The ratios of blocks (called the hit block) that use the same mode in two successive P frames are 70.6% for akiyo (QCIF) and 7.1% for mobile (QCIF), respectively. On the other hand, given a threshold, bar graph shows the absolute number of hit blocks within the threshold interval, whereas line graph shows the cumulative ratio due to the threshold. Notice that the sampling intervals are different in two figures.

It can be observed that, videos of low motion usually have a high hit ratio even for a small threshold. For instance, 50 is good for akiyo. On the other hand, video with high motion, such as mobile, performs on the contrary. Though omitted here, the same conclusion can be observed for both CIF and QCIF formats.

By this observation, we decide the threshold to be 50, 100, 150 and 200 to investigate low-motion videos, and use 500, 1000, 1500 and 2000 to investigate high-motion videos.

4.3. Coding time reduction

Now we consider the performance of the proposed algorithm. Since there is no significant difference between the format (CIF or QCIF), we only show the results for QCIF-size videos in Figures 8-10. Two typical results of akiyo, container and mobile with QCIF-size are shown in Figures 11-13. The figures show (in percentage) the reduction of coding time and the increase of bitrate. We used the same parameters as the previous experiment.

From Figure 7, we can see that the proposed algorithm successfully reduces the coding time. It means that, although we have to do some additional processes (such as the comparison of costs), the overhead is small. For videos with low motion, the reduction of coding time is remarkable even for a small threshold, at a maximum ratio of about 40% (for akiyo). On the other hand, for videos with high motion, the reduction of coding time increases slowly at a maximum ratio of about 10%.

The increased bitrates caused by the proposed algorithm are shown in Figure 8. We can see that the increase of bitrate for a high-motion video is small, whereas for low motion videos such as silent and mother, the bitrate increases rapidly.
In this paper, we have proposed a simple algorithm to reduce the coding complexity in block mode determination for H.264. Our idea is to use the same mode as used in a reference block. Experiment results show that our algorithm can reduce the coding time by up to 40% with the same picture quality and only 10% increase of the bitrate. We note that the coding time for high-motion videos can also be reduced from 10% to 30% with little increase of the bitrate.

For the efficiency, the proposed algorithm depends on the video’s character. Actually we need to determine the threshold manually. In order to choose a good threshold automatically in the program, we need more detailed studies in the future.

References


5. Conclusion