Transcoding of H.264 bitstream to MPEG-2 bitstream.

Sreejana Sharma, Student Member IEEE and Dr. K. R. Rao, IEEE fellow

Abstract—This paper presents a method for transcoding from H.264 to MPEG-2. This process is divided into two parts: intra frame transcoding and inter frame transcoding. For intra frame transcoding the brute force method of encoding and decoding in H.264 and re-encoding in MPEG-2 is applied. In inter frame transcoding the motion vector information from the H.264 decoding stage is reused in the MPEG-2 encoding stage leading to significant savings in the transcoding time with very less degradation in terms of PSNR when compared to an MPEG-2 encoded decoded sequence.

Index Terms—H.264, MPEG-2, transcoding.

1. Introduction

In this fast growing world of multimedia and telecommunications there is a great demand for efficient usage of the available bandwidth. With the growth of technology there is an increase in the number of networks, types of devices and different content representation formats as a result of which interoperability between different systems and networks is gaining in importance. Transcoding of video content is one such effort in this direction. This paper focuses on transcoding video from H.264/AVC [4][10][13] bitstream to MPEG-2 [7][11] bitstream. The main goals of an efficient transcoder are to maintain the quality of the transcoded bitstream to the one obtained by direct decoding and re-encoding of the input stream, to reuse the information available in the bitstream as much as possible so as to avoid multigenerational deterioration and last but not the least the process should be efficient, low in complexity and achieve the highest quality possible.

H.264/MPEG-4 part 10 fulfills significant coding efficiency, simple syntax specifications and seamless integration of video coding into all current protocols and multiplex architectures. It provides MPEG-2 comparable video quality at an average of half the bandwidth. However many legacy systems including all Digital TVs and home receivers use MPEG-2. This leads to the need for transcoding between H.264 and MPEG-2.

2. Considerations for H.264 to MPEG-2 transcoding.

For efficient transcoding it is essential to exploit the similarities present in the two standards and also keep in mind the differences between the two standards [6]. In this paper, transcoding from the baseline profile at level 3 of H.264/AVC to the simple profile at the main level of MPEG-2 is proposed, only I and P frames are involved.

<table>
<thead>
<tr>
<th>Feature Standard</th>
<th>MPEG-2</th>
<th>MPEG-4 part 10</th>
<th>H.264</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macroblock size</td>
<td>16x16 (frame mode)</td>
<td>16x8 (field mode)</td>
<td>16x16</td>
</tr>
<tr>
<td>Block size</td>
<td>8x8</td>
<td>16x16, 16x8, 8x8, 4x8, 8x8, 4x8</td>
<td></td>
</tr>
<tr>
<td>Intra prediction</td>
<td>No</td>
<td>Spatial Domain</td>
<td></td>
</tr>
<tr>
<td>Transform</td>
<td>8x8 DCT</td>
<td>8x8, 4x8 integer DCT 4x4, 2x2 Hadamard</td>
<td></td>
</tr>
<tr>
<td>Quantization</td>
<td>Scalar quantization with step size of constant increment</td>
<td>Scalar quantization with step size of increase at the rate of 12.5%</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Comparison of H.264 and MPEG-2

3. Intra frame transcoding

Transcoding of I frames is done by using the brute force method that is decoding the H.264 bitstream and then encoding it in the MPEG-2 format. Since the MPEG-2 standard does not have mode decision based intra coding, the input macroblock after being decoded by the H.264 decoder is passed through the MPEG-2 encoder where it is divided into four 8x8 blocks and a 2D 8x8 DCT is applied across it followed by quantization and VLC coding.

Table 2. Results of I frame transcoding
As can be seen from Table 2, the PSNR values obtained after transcoding are comparable to those obtained by encoding and decoding the output stream.

<table>
<thead>
<tr>
<th>Clip Number</th>
<th>Sequence Type</th>
<th>Bit rate (Mbps)</th>
<th>Average PSNR of transcoded sequence (dB)</th>
<th>Average PSNR of MPEG-2 encoded decoded sequence (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Foreman</td>
<td>2</td>
<td>34.385</td>
<td>34.654</td>
</tr>
<tr>
<td>2</td>
<td>Akiyo</td>
<td>1.4</td>
<td>35.525</td>
<td>35.889</td>
</tr>
<tr>
<td>3</td>
<td>Coastguard</td>
<td>3.4</td>
<td>33.928</td>
<td>34.869</td>
</tr>
<tr>
<td>4</td>
<td>Hall Monitor</td>
<td>1.8</td>
<td>33.349</td>
<td>33.598</td>
</tr>
<tr>
<td>5</td>
<td>Container</td>
<td>2.6</td>
<td>33.705</td>
<td>33.783</td>
</tr>
</tbody>
</table>

As can be seen from Table 2, the PSNR values obtained after transcoding are comparable to those obtained by encoding and decoding the output stream.

4. Inter frame transcoding

The process of inter (P) frame transcoding is divided into four parts: 1) motion vector extraction from H.264 decoding stage 2) motion vector resampling 3) motion vector refinement and 4) motion vector reuse.

However, the simple re use of the H.264 motion vectors does not take into account the quantization errors and hence it will produce non optimal results. Besides this H.264 can have up to 16 motion vectors due to adaptive block size motion estimation. H.264 also has increased pixel accuracy as compared to MPEG-2. These advanced features of H.264 make direct reuse of H.264 motion vectors in MPEG-2 non optimal. Taking the average of all of the motion vectors to arrive at a single 16x16 vector becomes less reliable as the number of sub-block increases because of the possibility that a motion vector was found in a different direction.

Since an H.264 macroblock can have the following block sizes: 16x16, 16x8, 8x16 and 8x8. Also each 8x8 block can be sub-divided into 8x8, 8x4, 4x8 and 4x4 hence there can be up to 16 motion vectors for a single 16x16 macroblock. For this reason it is important to arrive at a single 16x16 motion vector for the entire macroblock so that it can be re used at the MPEG-2 encoding stage. Simply taking the average of all these motion vectors can produce large degradation in terms of PSNR. The proposed method extracts motion vectors from H.264 and scales them based on the macroblock size or sub-macroblock size. The scaled motion vectors are then summed together to obtain a single motion vector for the entire 16x16 macroblock. The algorithm for this process is described as follows.

5. Motion vector resampling

For the 16x8 block size a single motion vector is determined as follows:

MPEG-2mv_x = (H.264mv_ax + H.264mv_bx)/2
MPEG-2mv_y = (H.264mv_ay + H.264mv_by)

Where H.264mv_ax and H.264mv_bx are the horizontal components of the 16x8 block sized H.264 motion vectors, while H.264mv_ay and H.264mv_by are the vertical components of the 16x8 block sized H.264 motion vectors.

For the 8x16 block size a single motion vector is arrived by the calculation as shown below:

MPEG-2mv_x = (H.264mv_ax + H.264mv_bx)
MPEG-2mv_y = (H.264mv_ay + H.264mv_by)/2

In H.264 if the MB type is 3 or 4, the macroblock will be further divided into sub macroblocks. In this case to arrive at a single motion vector for the entire 16x16 block, first single motion vectors are derived for each 8x8 block and then the four motion vectors for each 8x8 block are summed together to get the final 16x16 motion vector for the entire macroblock.

Figure 2. Motion vectors in 16x16 macroblock of H.264
If the sub-macroblock type is 0 then the sub-macroblock size is 8x8. A single motion vector for H.264 for this block size is arrived as follows:

\[
\begin{align*}
H.264\text{mv\_ax} &= (H.264\text{mv\_ax1}) \\
H.264\text{mv\_ay} &= (H.264\text{mv\_ay1})
\end{align*}
\]

Now in case where the sub-macroblock type is 1, the 8x8 block is split into two 8x4 sub blocks then first a single motion vector is derived for the particular 8x8 block as follows:

\[
\begin{align*}
H.264\text{mv\_ax} &= (H.264\text{mv\_ax1} + H.264\text{mv\_bx1})/2 \\
H.264\text{mv\_ay} &= (H.264\text{mv\_ay1} + H.264\text{mv\_by1})/4
\end{align*}
\]

where \(H.264\text{mv\_ax1}\) and \(H.264\text{mv\_bx1}\) are the vertical components of the motion vectors of the sub-macroblocks, while \(H.264\text{mv\_ay1}\) and \(H.264\text{mv\_by1}\) are the horizontal components of the motion vectors of the sub-blocks.

Similarly when the sub-macroblock type is 2, the 8x8 block is divided into two 4x8 sub-blocks, a single motion vector for the 8x8 block is arrived as follows:

\[
\begin{align*}
H.264\text{mv\_ax} &= (H.264\text{mv\_ax1} + H.264\text{mv\_bx1}) \\
H.264\text{mv\_ay} &= (H.264\text{mv\_ay1} + H.264\text{mv\_by1})/2
\end{align*}
\]

Again if the 8x8 sub-macroblock is divided into four 4x4 blocks then a single motion vector is arrived at for the 8x8 block as follows:

\[
\begin{align*}
H.264\text{mv\_ax} &= (H.264\text{mv\_ax1} + H.264\text{mv\_bx1} + H.264\text{mv\_cx1} + H.264\text{mv\_dx1})/4 \\
H.264\text{mv\_ay} &= (H.264\text{mv\_ay1} + H.264\text{mv\_by1} + H.264\text{mv\_cy1} + H.264\text{mv\_dy1})/4
\end{align*}
\]

Finally when all the motion vectors for each of the 8x8 sub-blocks have been obtained then, a single motion vector for the entire 16x16 block is arrived by adding all the motion vectors of each of the 8x8 sub-block as follows:

\[
\begin{align*}
\text{MPEG}\text{-2mv\_x} &= 1/4(H.264\text{mv\_x0} + H.264\text{mv\_x1} + H.264\text{mv\_x2} + H.264\text{mv\_x3}) \\
\text{MPEG}\text{-2mv\_y} &= 1/4(H.264\text{mv\_y0} + H.264\text{mv\_y1} + H.264\text{mv\_y2} + H.264\text{mv\_y3})
\end{align*}
\]

where 0, 1, 2 and 3 are the four different 8x8 blocks in a single 16x16 macroblock.

6. Motion vector refinement

In most cases of transcoding, the effect of quantization errors is negligible and hence performing a new motion estimation will give the same result as the incoming motion vector (i.e. the incoming motion vector is optimal) [5]. However since in general there is no guarantee that the effect is negligible all the time, there are non-zero probabilities that the quantization errors may cause the incoming motion vector to be non-optimal. Although the optimized motion vector can be obtained by new motion estimation, it is not desirable because of its high computational complexity. The reuse of the incoming motion vector [5] has been widely accepted because it is generally thought to be as good as performing new full-scale motion estimation and has been assumed in various transcoding architectures.

The optimal motion vector can be obtained by refining the incoming motion vector within a small search range as opposed to performing full-scale motion estimation.

A small search range of -2 pels and +2 pixels around the incoming motion vectors is used to search for the motion vector with the minimum SAD and use it as the optimum motion vector. This will not only give an optimal motion vector but will also reduce the complexity involved in performing new full scale motion estimation. So in order to overcome this I have performed some motion vector refinement using the HAVS (horizontal and vertical search) method [5]. According to this scheme, instead of searching all checking points within the search window, the HAVS scheme searches first for a minimum SAD point in the horizontal line and then over the vertical line.

Figure 3. Flow chart depicting the algorithm for motion vector refinement
7. Results and overview of P frame transcoding

The video clips used for this experiment are standard test clips [16]. The objective of the proposed scheme for P frame transcoding is to reduce the complexity of the transcoding process without significant changes in the PSNR. It can be observed from table 3 that the PSNR obtained by complete decoding and re-encoding of the H.264 bit stream is comparable to that obtained by the proposed method. The PSNR also depends on other factors such as the bit rate and the motion search window size. For these experiments the bit rate was maintained constant at 1Mbps and the motion search window for the full re-encoding process was maintained at -15 pels to +15 pels both in horizontal and vertical directions.

For P frame transcoding the test sequence is fed into the H.264 encoder and the obtained H.264 bit stream is then decoded. Motion vectors are extracted from the H.264 decoding stage and then summed together using the proposed algorithm. The obtained motion vectors are then refined over -2 pixels to +2 pixels in the horizontal and vertical direction to get the optimum motion vectors. These obtained motion vectors are then fed into the MPEG-2 encoding stage to get the final MPEG-2 transcoded bitstream.

<table>
<thead>
<tr>
<th>Clip No.</th>
<th>Test Clip</th>
<th>PSNR (dB)</th>
<th>PSNR (dB)</th>
<th>Average Time (ms)</th>
<th>PSNR (dB)</th>
<th>Average Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Akiyo</td>
<td>42.23</td>
<td>41.38</td>
<td>6000</td>
<td>40.98</td>
<td>2031</td>
</tr>
<tr>
<td>2</td>
<td>Coastguard</td>
<td>34.85</td>
<td>34.12</td>
<td>8000</td>
<td>33.11</td>
<td>3224</td>
</tr>
<tr>
<td>3</td>
<td>Container</td>
<td>34.89</td>
<td>34.68</td>
<td>7000</td>
<td>33.58</td>
<td>3500</td>
</tr>
<tr>
<td>4</td>
<td>Foreman</td>
<td>37.45</td>
<td>37.11</td>
<td>8000</td>
<td>36.71</td>
<td>2000</td>
</tr>
<tr>
<td>5</td>
<td>Hall Monitor</td>
<td>36.51</td>
<td>36.21</td>
<td>8000</td>
<td>35.31</td>
<td>3040</td>
</tr>
</tbody>
</table>

It can be observed that the results obtained by the proposed method are very close to those obtained by H.264 decoding and MPEG-2 encoding and decoding of the clip. However, there are significant savings in the encoding time due to motion vector reuse from the H.264 decoding stage. The transcoder complexity is measured as the total time spent for the MPEG-2 encoding process (as this was the only variable).

Figure 5. Comparison of the subjective quality of the input P frame (left) and the transcoded output P frame (right) for the test sequence Foreman transcoded at 1Mbps.
9. Experimental setup

The test clips used are standard CIF (352x288) resolution test clips. For each clip 30 frames are transcoded. For I frame transcoding these clips are transcoded at different bit rates from H.264 to MPEG-2. The GOP size is set equal to 30 with I II II II II ....GOP structure. The PSNR of the transcoded sequence is then compared with the PSNR of the MPEG-2 encoded decoded sequence. For P frame transcoding, the clips are encoded into H.264 streams at a bit rate of 1Mbps and with a GOP size of 15 and the IPPPPP....GOP. These streams are then transcoded to 1Mbps MPEG-2 with the same GOP structure IPPPPP.... The PSNR of the transcoded sequence is then compared with the PSNR of the MPEG-2 encoded decoded sequence and the brute force method of H.264 to MPEG-2 transcoding. The PSNR values presented are the average PSNR values for all the frames in the test sequences used.

10. Conclusions

By cleverly converting motion vectors of the various block sizes (16x16,16x8,8x16,8x8,4x8,4x4) in H.264 for P frames to a single motion vector 16x16 macroblock followed by further refinement, significant savings in complexity inherent in the H.264 to MPEG-2 transcoding are obtained while maintaining negligible loss in video quality compared to complete H.264 decoding and MPEG-2 encoding. This is illustrated with various video test clips in CIF 4:2:0 format, using the proposed transcoding scheme.

11. Future work

The research presented in this paper is directed at low complexity, speed and comparable quality. Now that these targets have been achieved, the transcoder can be optimized for use in specific applications. This transcoder can be applied on PVR or multimedia server to play the delivered H.264 content in the existing MPEG-2 equipment.

Also this paper is based on transcoding of H.264 from baseline profile to MPEG-2 simple profile with no B frames. The same process could be extended to transcoding from the main profile of H.264 to main profile of MPEG-2 with the presence of B frames.

12. References


