A TEMPORAL ERROR CONCEALMENT ALGORITHM FOR H.264 BASED ON PLANE ESTIMATION

Jinghong Zheng, Lap-Pui Chau

School of Electrical and Electronic Engineering
Nanyang Technological University, Nanyang Avenue, Singapore (689574)

Abstract

In this paper, we propose a new temporal error concealment algorithm for the forthcoming video coding standard H.264, which uses a first-order plane to estimate motion vector. In H.264, a 16×16 inter macroblock can be divided into variant block shape for motion prediction. The blocks within a small area are likely to move in the same direction. By using the motion vectors that next to the vertices of lost macroblock, we can constitute a first-order plane that indicate the movement tendency in this small area, and estimate the motion vector of vertices. Then the motion vectors of vertices are used to interpolate motion vector of each pixel separately. To reduce the computation cost, we select a simple equation to implement the interpolation. The simulation results show that our method can efficiently improve the video quality over different macroblock lost rate.

1. Introduction

H.264 is a forthcoming coding standard which adopts some new coding schemes [1]. The primary difference between H.264 and previous coding standards is that a 16×16 macroblock can be divided into different block shapes for motion prediction. Figure 1 shows the seven different block division modes and the order of encoding that are adopted in H.264 [2]. Each block has a corresponding motion vector. The number of motion vector in one macroblock is changed from 1 set to 16 sets. In H.264, an inter macroblock will contain more motion vectors, and a motion vector covers smaller region than previous coding standards. Our error concealment algorithm makes use of this property to recover the lost macroblock. The motion vectors that cover neighboring 4×4 blocks are used to estimate the motion vectors of the four vertices that locate at the corners of lost macroblock. By using the four estimated motion vectors, the motion vector of each pixel that is within the lost macroblock can be interpolated.

Until now many efficient temporal error concealment methods have been proposed. Chen et al proposed an overlapped motion compensation algorithm to recover the lost blocks [3]. Lee et al presents a method that uses multi-frame boundary matching algorithm to compare each candidate block, then select the best matching block to mask corrupted block [4]. Wang et al use best neighborhood matching theory to recover lost block [5]. Teskeridou et al proposed an approach that makes use of neighboring blocks. They compare each candidate block with the neighboring block of corrupted block. After finding the best matching block, they use the block at corresponding location to mask the corrupted block [6]. Atzori et al proposed an algorithm that combines boundary matching algorithm and mesh-based warping [7]. The main drawback of these methods is that incorrect estimation of the concealment displacement can lead to poor recovery result of the whole or most part of corrupted block. To avoid this problem, some motion field interpolation methods had been presented. Mualla et al present a motion field interpolation method that uses a bilinear motion field interpolation equation to calculate the motion vector of each pixel [8]. Jong et al proposed an algorithm that is based on NURBS interpolation [9]. In these interpolation methods, the motion vector for each pixel is estimated individually. And incorrect estimation of motion vector only effect partial pixels, but not the whole block.

Figure 1. The macroblock division modes used in H.264.

In this paper, we propose a new error concealment method that makes use of the coding property of H.264. First, we use the motion vectors of neighboring 4×4 blocks to constitute a first-order plane. Because the motion vector in H.264 covers smaller region than previous coding standards, the correlation between neighboring motion vectors increases. We can use the first-order plane to estimate the motion vector of vertex that is at corner of lost macroblock. Then we select a typical linear interpolation function to calculate the motion vector for each pixel [10]. This interpolation function not only makes the motion change smoothly within the recovered macroblock, but also
makes the recovered macroblock smoothly connect with its neighboring macro-blocks.

2. Algorithm of Linear Interpolation

In this section we proposed a linear interpolation error concealment algorithm for H.264. In H.264, the motion vector covers smaller region than previous coding standard, hence the correlation of motion vectors between neighboring blocks increases. We make use of this property of the macroblock mode to develop our new algorithm in the following.

Our approach can divide into two steps. First, we estimate the motion vectors of four vertices of lost macroblock. We assume that the location of lost macroblock has been detected, and all the neighboring macroblocks are correctly received and decoded. The corresponding positions of lost macroblock and its neighboring macroblocks are shown in figure 2, i and j denote the spatial location of macroblocks. \( M_{v1}, M_{v2}, M_{v3}, \) and \( M_{v4} \) indicate the motion vectors of four vertices in lost macroblock. The motion vectors that cover neighboring \( 4 \times 4 \) blocks are used to estimate the motion vectors of four vertices. For simplicity, we only present how to calculate \( M_{v1} \). \( v_1, v_2 \) and \( v_3 \) represent the motion vectors of neighboring \( 4 \times 4 \) blocks respectively, as shown in Figure 2. They are used to estimate the value of \( M_{v1} \).

\[
\begin{align*}
F_{i+1,j-1} & F_{i+1,j} \quad F_{i+1,j+1} \\
F_{i,j-1} & F_{i,j} \quad F_{i,j+1} \\
F_{i+1,j-1} & F_{i+1,j} \quad F_{i+1,j+1}
\end{align*}
\]

Figure 2: The estimation of the vertices motion vectors.

We assume that the location of pixel that locates at the vertex of lost macroblock is \((0, 0)\) in two dimensional reference frame, and each pixel size is one unit. The coordinates of neighboring block centers are \((1.5, -2.5)\), \((-2.5, -2.5)\) and \((-2.5, 1.5)\) respectively, as shown in Figure 3(b). We use the coordinates of neighboring block centers and the corresponding motion vectors, \( v_1, v_2 \) and \( v_3 \), to constitute a first-order plane \( P \) as described in equation (1).

\[
\frac{x}{a} + \frac{y}{b} + \frac{z}{c} = 1
\]  

where \( x \) and \( y \) indicate the center coordinates of neighboring \( 4 \times 4 \) blocks in two dimensional reference frame, and \( z \) is the corresponding motion vector, as shown in Figure 3(a).

Then we can use \( v_1, v_2 \) and \( v_3 \) to compute the coefficients in equation (1), the result is represented in equation (2).

\[
\begin{align*}
a &= \frac{2.5v_1 - v_2 + 2.5v_3}{v_2 - v_1} \\
b &= \frac{2.5v_1 - v_2 + 2.5v_3}{v_2 - v_1} \\
c &= \frac{2.5v_1 - v_2 + 2.5v_3}{4}
\end{align*}
\]

Because the neighboring blocks are likely to move in same direction, we can assume that \( M_{v1} \) should locate on the first-order plane \( P \) in figure 3(a). Then \( M_{v1} \) can be estimated by \( v_1, v_2 \) and \( v_3 \), as represented in equation (3). \( M_{v2}, M_{v3}, \) and \( M_{v4} \) can be estimated by similar approach. If the left or right macroblock is corrupted too, we can select other neighboring motion vectors that are from the upper and below macroblocks to realize the concealment. But the recovery result will not as good as using the nearest motion vectors.

\[
M_{v1} = \frac{2.5v_1 - v_2 + 2.5v_3}{4}
\]  

(a)
Then we use the motion vectors of four vertices to interpolate the motion vector for each pixel within the lost macroblock. To reduce the computation cost, we select a simple linear equation (4) to interpolate the motion vector for each pixel.

\[
M_{v_x,y} = M_{v_1}(\tilde{y} - \tilde{x}) + M_{v_2}(\tilde{x} \tilde{y}) + M_{v_3}(1 - \tilde{y} + \tilde{x}) + M_{v_4}(\tilde{x} - \tilde{x} \tilde{y})
\]  

(4)

where \( \tilde{x} \) and \( \tilde{y} \) represent the normalized pixel coordinates that are calculated by equation (5). In equation (5), \( x_0, x_1, y_0 \) and \( y_1 \) represent the edge coordinates of lost macroblock in four different directions respectively, as shown in Figure 4.

\[
\tilde{x} = \frac{x - x_0}{x_1 - x_0}
\]

(5a)

\[
\tilde{y} = \frac{y - y_0}{y_1 - y_0}
\]

(5b)

By using this interpolation algorithm, motion discontinuity can be avoided. The motion vectors change smoothly over the image. And the recovered macroblock will connect smoothly with its neighboring macroblocks. For example, when \( \tilde{x} = 1 \), this means the pixel is on the right side of macroblock. In this case, only \( M_{v_2} \) and \( M_{v_4} \) contribute to \( M_{v_{\tilde{x},\tilde{y}}} \).

Through the interpolated motion vectors, we can find corresponding pixel intensities in the reference frame. And we can use these pixels to displace the lost pixel, as presented by equation (6).

\[
I_{\text{rec}}(x, y, k) = I_{\text{ref}}^c(x + M_{v_{\tilde{x},\tilde{y}}}^H, y + M_{v_{\tilde{x},\tilde{y}}}^V, k - 1)
\]

(6)

where \( I_{\text{rec}} \) and \( I_{\text{ref}}^c \) denote the pixel intensity in current frame and reference frame respectively. \( k \) indicates the frame number. \( M_{v_{\tilde{x},\tilde{y}}}^H \) and \( M_{v_{\tilde{x},\tilde{y}}}^V \) denote the horizontal and vertical part of \( M_{v_{\tilde{x},\tilde{y}}} \) respectively. By using these corresponding pixels, the lost macroblock can be reconstructed. The simulation results prove that our method can improve the video quality obviously.

3. Simulation Results

We select three QCIF video sequences, i.e. Stefan, Foreman, and Coastguide, to conduct the experiment. The videos are encoded and decoded by H.264-PFGS program. The frame structure of encoded video is IPPPIPP…… By changing the quantization factor, the videos are encoded at different bitrates. The location of lost macroblock is selected randomly. We test the macroblock lost rate at 2%, 5% and 10% respectively. The PSNR of each scenario were presented in table 1. Because H.264 adopts different motion estimation scheme, most temporal error concealment methods that are designed for previous coding standards are not suitable for H.264. Then we only compare the simulation results of our algorithm with the most common error concealment method—temporal replacement, which replaces the lost motion vectors with \((0,0)\). PSNRc represents the PSNR of video that is recovered by our algorithm. PSNRf denotes the video recovered by temporal replacement. PSNRr denotes the PSNR of corrupted video with different macroblock lost rate. The simulation results show that the quality of corrupted video can be efficiently improved by the proposed method. For the video sequence Stefan that contains fast movement, our temporal error concealment method still can obtain a good recovery results.

Figure 5 shows the change tendency of PSNR along different bitrate. In figure 5(a), the macroblock lost rate is 2%, the PSNR of recovered video is close to the PSNR of original decoded video. In figure 5(b), 10% macroblocks are lost in each frame. Compared with the temporal replacement, the proposed method can archive better visual quality.
4. Conclusion

In this paper, we propose a temporal error concealment method for H.264. In H.264, an inter macroblock contains more motion vectors. This new coding scheme increases the correlation of neighboring motion vectors. By using the motion vectors of neighboring $4 \times 4$ blocks that are next to the vertex of lost macroblock, we can constitute a three-dimensional first-order plane, and use this plane to estimate the motion vector of vertex in lost macroblock. Then we select a linear interpolation equation to compute the motion vector for each pixel of the lost macroblock. The simulation result shows the quality of corrupted image can be improved obviously by our method.

5. Reference


### Table 1. The Simulation Results of Our Algorithm

<table>
<thead>
<tr>
<th>Video Sequence</th>
<th>Original PSNR (dB)</th>
<th>QP</th>
<th>Bitrate (Kbit/s)</th>
<th>PSNR (dB)</th>
<th>PSNR of Different Macroblock Lost Rate(dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2%</td>
</tr>
<tr>
<td>Foreman</td>
<td>33.86</td>
<td>20</td>
<td>191.26</td>
<td>PSNR&lt;sub&gt;L&lt;/sub&gt;</td>
<td>22.42</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PSNR&lt;sub&gt;T&lt;/sub&gt;</td>
<td>32.39</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PSNR&lt;sub&gt;C&lt;/sub&gt;</td>
<td>33.30</td>
</tr>
<tr>
<td></td>
<td>36.28</td>
<td>16</td>
<td>302.97</td>
<td>PSNR&lt;sub&gt;L&lt;/sub&gt;</td>
<td>22.57</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PSNR&lt;sub&gt;T&lt;/sub&gt;</td>
<td>33.55</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PSNR&lt;sub&gt;C&lt;/sub&gt;</td>
<td>35.39</td>
</tr>
<tr>
<td>Stefan</td>
<td>31.53</td>
<td>20</td>
<td>459.48</td>
<td>PSNR&lt;sub&gt;L&lt;/sub&gt;</td>
<td>21.53</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PSNR&lt;sub&gt;T&lt;/sub&gt;</td>
<td>28.97</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PSNR&lt;sub&gt;C&lt;/sub&gt;</td>
<td>30.21</td>
</tr>
<tr>
<td></td>
<td>34.72</td>
<td>16</td>
<td>736.32</td>
<td>PSNR&lt;sub&gt;L&lt;/sub&gt;</td>
<td>22.63</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PSNR&lt;sub&gt;T&lt;/sub&gt;</td>
<td>30.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PSNR&lt;sub&gt;C&lt;/sub&gt;</td>
<td>32.97</td>
</tr>
<tr>
<td>Coast-guide</td>
<td>31.83</td>
<td>20</td>
<td>191.43</td>
<td>PSNR&lt;sub&gt;L&lt;/sub&gt;</td>
<td>22.61</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PSNR&lt;sub&gt;T&lt;/sub&gt;</td>
<td>30.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PSNR&lt;sub&gt;C&lt;/sub&gt;</td>
<td>31.49</td>
</tr>
<tr>
<td></td>
<td>34.46</td>
<td>16</td>
<td>374.24</td>
<td>PSNR&lt;sub&gt;L&lt;/sub&gt;</td>
<td>23.76</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PSNR&lt;sub&gt;T&lt;/sub&gt;</td>
<td>33.06</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PSNR&lt;sub&gt;C&lt;/sub&gt;</td>
<td>33.37</td>
</tr>
</tbody>
</table>

PSNR<sub>L</sub>: The PSNR of corrupted video without any concealment.
PSNR<sub>T</sub>: The PSNR of video recovered by temporal replacement
PSNR<sub>C</sub>: The PSNR of video recovered by our proposed algorithm.