AN EFFICIENT SPATIAL DOMAIN ERROR CONCEALMENT METHOD FOR H.264 VIDEO

Jinghong Zheng, Lap-Pui Chau

School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

Abstract

This paper presents an efficient spatial domain error concealment method for the forthcoming video coding standard H.264. In H.264, a frame is divided into 4×4 blocks during encoding procedure. For natural image signal, the blocks are smoothly connected with each other. Based on this property, a linear smoothness constraint equation that describes the connection of lost block and its neighboring blocks can be constructed. By solving this equation, the coefficients of lost block can be recovered. Because the reconstructed high frequency coefficients may be affected by noise, the recovered center pixel may have obvious error. To eliminate the error, we use the recovered pixels that are on the boundaries of the lost block and average pixel difference to interpolate the center pixels. The implementation is simple, and it is suitable for real-time video application. Experimental results show our method has better recovery result than conventional approach.

1. Introduction

H.264 is a forthcoming video coding standard. To achieve better coding efficiency, this new coding standard adopts some new coding schemes [1]. Most existing video coding standards use the 8×8 Discrete Cosine Transform (DCT) to compress the block pixels. H.264 is similar to prior standards in its use of a block transform, but there still exist some significant differences between them. The primary difference is that H.264’s transform is based on a 4×4 block, instead of the DCT, an integer transform is used [2] with basically the same coding efficiency as a 4×4 DCT. Although there are some spatial error concealment methods appeared in literature for the existing video coding standard, these methods are optimized for the 8×8 block. So the development of error concealment method for H.264 video becomes more important. In this paper, we propose a spatial error concealment method for this new coding standard.

There are some error concealment methods that can recover the corrupted block, such as projection onto convex sets (POCS), temporal predictive concealment (TPC) and spatial predictive concealment (SPC). The POCS is an efficient approach to prevent error propagation [3,4]. It uses the overlap information from several frames and blocks to generate relations among pixels in the damaged block. The TPC exploits the information in the previous frames to recover the corrupted areas with small motion [5]. The SPC methods often make use of the spatial redundancy to recover the damaged blocks. Many algorithms of SPC that were proposed for MPEG video are based on the smooth property of image. They form an object function according to different constraints. By solving the object function, the lost coefficients can be recovered. Park et al [6] and Chung et al [7]’s methods imposed the smooth connection constraint only on the boundary of the lost block to recover the lost DCT coefficients. The approach presented by Wang et al [8] imposed the smooth constraint to each pixel of the corrupted blocks and used the correctly received coefficients as well as the boundary pixels of adjacent blocks to calculate the lost DCT coefficients. Their recovery result is better than the results presented in [6] and [7], but the computation complexity hence be increased.

In this paper, we make use of the concept of SPC method to reconstruct the H.264 video. The main difference between the SPC method used in MPEG video and our proposed SPC method for H.264 video is the acquisition of unique solution from the object function. In MPEG video, when the smooth connection constraint is imposed only on the boundary of the corrupted block, the object function is an uncertain equation [6,7], which cannot get unique solution to reconstruct all 64 coefficients, only the low frequency coefficients can be recovered. In H.264 video, since the block size reduced to 4×4, the constraint function we define has unique solution and the whole 16 coefficients of the damaged block can be reconstructed. Since the reconstructed high frequency coefficients may be infected by noise, the reconstructed center pixels may have error. Conventional approach that can reduce the error is imposing the smooth constraint to each pixels, this will increase the computational complexity. To avoid this drawback, we propose a simple interpolation method that makes use of the recovered boundary pixels to compute the center pixels. Then the average of pixel difference is taken into account to enhance the video quality.

2. Smooth Connection Recovery Algorithm

In this section, we propose an efficient coefficients recovery algorithm for H.264. We assume the location of the lost block is detected, and the four neighboring blocks are correctly decoded. The size of block is 4×4, and an image is composed of Q×P blocks. Let us recall the notation appeared in reference [6]. The lost block is notated...
as \( F_i \), the four neighboring blocks are represented as \( F_{i-}\), \( F_{i+}\), \( F_{i-}\), and \( F_{i+}\), respectively, as shown in Fig 1. The four boundary pixel vectors of the block \( F_i \) in four different directions are \( c_i, b_i, l_i, \) and \( r_i \) respectively. \( c_i, b_i, l_i, \) and \( r_i \) are the boundary pixel vectors of the four neighboring blocks as described in Fig. 1.

Thus, we have:

\[
\begin{align*}
 c_i(n) &= F_i(n,0), \quad n = 0,\ldots,3 \quad (1a) \\
 b_i(n) &= F_i(n,3), \quad n = 0,\ldots,3 \quad (1b) \\
 l_i(m) &= F_i(0,m), \quad m = 0,\ldots,3 \quad (1c) \\
 r_i(m) &= F_i(3,m), \quad m = 0,\ldots,3 \quad (1d)
\end{align*}
\]

where \( F_i(n,m) \) represent the horizontal \( n \)-th pixel of the \( m \)-th row in lost block.

![Fig. 1: The lost block and the four boundary pixel vectors](image)

Firstly, we use the boundary pixel vectors to define a function that describes the connection of the damaged block and its four neighboring blocks. Because of the smoothness property of image, we can assume that the luminance components change relatively slowly within a small area. This means the boundary pixel values of two neighboring blocks are close. In equation (2) we defined the smooth connection constraint function, the recovered boundary pixel vectors should minimize the functions \( \varphi \).

\[
\varphi = \| c_i - c_{i-}\| + \| b_i - b_{i+}\| + \| l_i - l_{i-}\| + \| r_i - r_{i+}\|
\]

(2)

Next, we use the coefficients to represent the boundary pixel vectors of the damaged block. The inverse transform defined in H.264 is:

\[
\begin{align*}
 a' &= 13A + 17B + 13C + 7D \quad (3a) \\
 b' &= 13A + 7B - 13C - 17D \quad (3b) \\
 c' &= 13A - 7B - 13C + 17D \quad (3c) \\
 d' &= 13A - 17B + 13C - 7D \quad (3d)
\end{align*}
\]

where \( a', b', c', \) and \( d' \) are inverse transformed pixel values, the relation between \( a' \) and the original pixel \( a \) is \( a' = 676a \), similarly \( b' = 676b \) and so on. \( A, B, C, \) and \( D \) are transformed coefficients of pixels. The inverse transform is computed by the row-column decomposition method [2]. From equation (3), we can define the one-dimensional inverse transform matrixes \( h(n) \) and \( h(m) \) in equation (4). \( n \) and \( m \) denote the horizontal and vertical locations in pixel domain.

\[
\begin{align*}
 h(0) &= [13 \quad 17 \quad 13 \quad 7] \quad (4a) \\
 h(1) &= [13 \quad 7 \quad -13 \quad -17] \quad (4b) \\
 h(2) &= [13 \quad -7 \quad -13 \quad 17] \quad (4c) \\
 h(3) &= [-13 \quad -17 \quad 13 \quad -7] \quad (4d)
\end{align*}
\]

Then, we can constitute a two-dimensional inverse transform matrix \( h_{n,m} \).

\[
h_{n,m} = h(n) \otimes h(m) / 676^2
\]

(5)

The inverse transform matrix \( h_{n,m} \) is a \( 1 \times 16 \) matrix. By using the inverse transform matrix in equation (4), each pixels of the lost block can be represented by coefficient vector \( X_i \).

\[
F_i(n,m) = h_{n,m}X_i
\]

(6)

In equation (6), \( X_i \) is a \( 16 \times 1 \) coefficient vector of the block \( F_i \). \( F_i(n,m) \) indicates a pixel, whose coordinate is \( (n,m) \). Then we can rewrite the boundary pixel vectors of \( F_i \) in the following:

\[
\begin{align*}
 c_i(n) &= h_{n,c}^iX_i, \quad 0 \leq n \leq 3 \quad (7a) \\
 b_i(n) &= h_{n,b}^iX_i, \quad 0 \leq n \leq 3 \quad (7b) \\
 l_i(m) &= h_{n,l}^iX_i, \quad 0 \leq m \leq 3 \quad (7c) \\
 r_i(m) &= h_{n,r}^iX_i, \quad 0 \leq m \leq 3 \quad (7d)
\end{align*}
\]

where \( h_{n,c}^i, h_{n,b}^i, h_{n,l}^i, \) and \( h_{n,r}^i \) are \( 4 \times 16 \) inverse transform matrixes given by equation (8) respectively, and \( c_i(m), b_i(m), l_i(n) \) and \( r_i(n) \) are \( 4 \times 1 \) pixel vectors.

\[
\begin{bmatrix}
 h_{0,0}^c \\
 h_{1,0}^c \\
 h_{2,0}^c \\
 h_{3,0}^c
\end{bmatrix}
\quad (8a),
\begin{bmatrix}
 h_{0,0}^g \\
 h_{1,0}^g \\
 h_{2,0}^g \\
 h_{3,0}^g
\end{bmatrix}
\quad (8b)
\]

\( h_{n,m}^c \) and \( h_{n,m}^g \) are consistent.
In order to reduce the artifact within the block, we use first-order linear interpolation along each direction to calculate the average difference between each pixel in vertical and horizontal directions respectively, and use this average difference to improve the estimation of center pixels as described in Fig. 2.

\[
P(-1,-1) \quad P(0,-1) \quad P(1,-1) \quad P(2,-1) \quad P(3,-1) \quad P(4,-1) \\
P(-1,0) \quad P(0,0) \quad P(1,0) \quad P(2,0) \quad P(3,0) \quad P(4,0) \\
P(-1,1) \quad P(0,1) \quad P(1,1) \quad P(2,1) \quad P(3,1) \quad P(4,1) \\
P(-1,2) \quad P(0,2) \quad P(1,2) \quad P(2,2) \quad P(3,2) \quad P(4,2) \\
P(-1,3) \quad P(0,3) \quad P(1,3) \quad P(2,3) \quad P(3,3) \quad P(4,3) \\
P(-1,4) \quad P(0,4) \quad P(1,4) \quad P(2,4) \quad P(3,4) \quad P(4,4) 
\]

Consider the \([P(0,0),...,P(3,3)]\) is the recovered pixel matrix of damaged block \(F_i\). The \(D_r\) and \(D_a\) are the average differences in vertical and horizontal directions. Then, we use them to set the center pixels as in equation (12).

\[
P'(1,1) = \frac{P(0,1) + P(1,0)}{2} + \frac{D_r + D_a}{2} \\
P'(2,1) = \frac{P(2,0) + P(0,1)}{2} + \frac{D_r + D_a}{2} \\
P'(1,2) = \frac{P(0,2) + P(1,3)}{2} + \frac{D_r + D_a}{2} \\
P'(2,2) = \frac{P(2,3) + P(3,2)}{2} + \frac{D_r + D_a}{2}
\]

Therefore, we use these modified center pixels to replace the prior recovered center pixels for a better result. Through our experiment, it has proven that the modified pixel matrix has a much better PSNR than the result without applying the averaging method.

3. Simulation Result

Since H.264 is an upcoming video coding standard, in the literature, there has not much work reported on the error concealment method for H.264 video. To show the performance of proposed method, we compare our method with the directly copying error concealment method, which is used in most standard decoders. We select three different video sequences, i.e. Akiyo, Foreman and Stefan, whose resolutions are 176 × 144 pixels, to conduct the experiments. Because our algorithm is designed for intrablock, each video was encoded in 150 I-frame by TML 8.4 software. By changing the quantization factor of the encoder, we encoded each sequence at different bitrates. In each frame, we randomly select the location of the lost blocks, and set the lost block rate to 1%, 5%, 10% and 15%, respectively.

The PSNR of each scenario were presented in the Table 1. Firstly, we use the whole sixteen recovered coefficients to reconstruct the pixels of the lost block, and calculate the PSNR of the reconstituted frame, which is notated as PSNRc. However this method cannot achieve satisfactory
visual performance, since the center pixels may have negative values. Then, we only selected seven low frequency coefficients to recover the lost pixels, and the rest nine high frequency coefficients are set to zero. The PSNR of the frame recovered by this method is notated as PSNR_p, the PSNR improve by about 2 dB. PSNR_p denotes the PSNR of video that is reconstructed by directly copying method. PSNR_c represents the PSNR of our proposed method presented in section two. The experiment result shows that our method can obtain better recovery result than the other two approaches.

4. Conclusion

This paper proposes a new algorithm of error concealment for the new video coding standard H.264. We use the boundary pixels of lost block and its four neighboring blocks to constitute a constraint equation that is based on first-order derivative-based smoothness measure. And the first-order linear interpolation is exploited in our method to further achieve higher visual quality. Our experiment has shown that our approach is efficient for different transmission bitrates in various video sequences over a wide range of block lost rate.

5. Reference

<table>
<thead>
<tr>
<th>YUV</th>
<th>Bitrate (Kbit/s)</th>
<th>PSNR (without loss)</th>
<th>Block Lost Rate</th>
<th>PSNR&lt;sub&gt;L&lt;/sub&gt;</th>
<th>PSNR&lt;sub&gt;r&lt;/sub&gt;</th>
<th>PSNR&lt;sub&gt;p&lt;/sub&gt;</th>
<th>PSNR&lt;sub&gt;c&lt;/sub&gt;</th>
<th>PSNR&lt;sub&gt;o&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>akiyo</td>
<td>321.84</td>
<td>37.39</td>
<td>1%</td>
<td>25.96</td>
<td>28.48</td>
<td>30.30</td>
<td>34.11</td>
<td>36.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5%</td>
<td>20.27</td>
<td>22.61</td>
<td>24.77</td>
<td>32.46</td>
<td>35.85</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10%</td>
<td>18.05</td>
<td>20.21</td>
<td>22.44</td>
<td>30.80</td>
<td>35.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15%</td>
<td>12.15</td>
<td>14.19</td>
<td>16.44</td>
<td>29.48</td>
<td>31.74</td>
</tr>
<tr>
<td></td>
<td>265.43</td>
<td>35.96</td>
<td>1%</td>
<td>25.63</td>
<td>28.03</td>
<td>29.79</td>
<td>33.41</td>
<td>35.61</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5%</td>
<td>20.25</td>
<td>22.57</td>
<td>24.66</td>
<td>31.95</td>
<td>34.79</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10%</td>
<td>18.04</td>
<td>20.16</td>
<td>22.35</td>
<td>30.43</td>
<td>34.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15%</td>
<td>12.27</td>
<td>14.44</td>
<td>16.69</td>
<td>29.15</td>
<td>31.43</td>
</tr>
<tr>
<td>foreman</td>
<td>545.58</td>
<td>34.07</td>
<td>1%</td>
<td>24.21</td>
<td>25.45</td>
<td>27.31</td>
<td>29.75</td>
<td>33.78</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5%</td>
<td>18.70</td>
<td>19.84</td>
<td>21.99</td>
<td>26.78</td>
<td>33.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10%</td>
<td>16.73</td>
<td>17.92</td>
<td>20.10</td>
<td>25.46</td>
<td>32.56</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15%</td>
<td>10.85</td>
<td>11.99</td>
<td>14.22</td>
<td>22.72</td>
<td>30.05</td>
</tr>
<tr>
<td></td>
<td>439.54</td>
<td>32.68</td>
<td>1%</td>
<td>24.04</td>
<td>25.12</td>
<td>26.87</td>
<td>29.16</td>
<td>32.47</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5%</td>
<td>18.61</td>
<td>19.87</td>
<td>21.98</td>
<td>26.40</td>
<td>31.97</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10%</td>
<td>16.70</td>
<td>17.91</td>
<td>20.07</td>
<td>25.05</td>
<td>31.58</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15%</td>
<td>10.77</td>
<td>12.00</td>
<td>14.23</td>
<td>22.63</td>
<td>29.52</td>
</tr>
<tr>
<td>stefan</td>
<td>1230.53</td>
<td>32.30</td>
<td>1%</td>
<td>25.14</td>
<td>26.27</td>
<td>27.89</td>
<td>29.72</td>
<td>31.71</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5%</td>
<td>20.06</td>
<td>21.27</td>
<td>23.43</td>
<td>25.38</td>
<td>30.52</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10%</td>
<td>17.92</td>
<td>19.13</td>
<td>21.43</td>
<td>22.99</td>
<td>29.66</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15%</td>
<td>12.14</td>
<td>13.41</td>
<td>15.82</td>
<td>21.12</td>
<td>26.05</td>
</tr>
<tr>
<td></td>
<td>1008.56</td>
<td>30.52</td>
<td>1%</td>
<td>24.65</td>
<td>25.54</td>
<td>27.06</td>
<td>28.75</td>
<td>30.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5%</td>
<td>19.75</td>
<td>20.99</td>
<td>23.08</td>
<td>24.57</td>
<td>29.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10%</td>
<td>17.86</td>
<td>19.11</td>
<td>21.32</td>
<td>22.56</td>
<td>28.64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15%</td>
<td>12.14</td>
<td>13.45</td>
<td>15.85</td>
<td>20.86</td>
<td>25.64</td>
</tr>
</tbody>
</table>

Note:  
PSNR<sub>L</sub>: without any error concealment.  
PSNR<sub>r</sub>: using the whole 16 recovered coefficients to calculate the missing pixels.  
PSNR<sub>p</sub>: using the 7 low frequency coefficients to recover the missing pixels.  
PSNR<sub>c</sub>: using motion compensated copying method to recover the missing pixels.  
PSNR<sub>o</sub>: our method.