Multimedia Processing
Term project
on

ERROR CONCEALMENT TECHNIQUES IN H.264 VIDEO TRANSMISSION OVER WIRELESS NETWORKS
Interim Report

Spring 2016
Under
Dr. K. R. Rao

by
Moiz Mustafa Zaveri (1001115920)
moiz.mustafazaveri@mavs.uta.edu
Contents

1. Problem Statement
2. Objective
3. The H.264 Standard
4. Sequence Characterization
5. Error Characteristics
6. Error Concealment Techniques
7. Quality Metrics
8. Generation of Errors
9. References
Acronyms

AVC  Advanced Video Coding
AVS  Audio Video Standard
BD   Bjontegaard Distortion
DSL  Digital Subscriber Line
HEVC High Efficiency Video Coding
IEC  International Electrotechnical Commission
ISO  International Organization for Standardization
ITU  International Telecommunication Union
JM   Joint Model
LAN  Local Area Network
MMS  Multimedia Messaging Service
MSU  Moscow State University
PSNR Peak signal to noise ratio
SAD  Sum of absolute differences
SI   Spatial Information
SSIM Structural similarity index metric
TI   Temporal Information
Problem Statement:

Video transmission errors are errors in the video sequence that the decoder cannot decode properly. In real-time applications, no retransmission can be used, therefore the missing parts of the video have to be concealed. To conceal these errors, spatial and temporal correlations of the video sequence can be utilized. As H.264 employs predictive coding, this kind of corruption spreads spatio-temporally to the current and consecutive frames.

Objective:

To implement both the spatial domain and temporal domain categories of error concealment techniques in H.264 [10] with the application of the Joint Model (JM) Reference software [10] and use metrics like the peak signal to noise ratio (PSNR), structural similarity index metric (SSIM) [9], BD bit rate [13] and BD PSNR [13] in order to compare and evaluate the quality of reconstruction.

The H.264 standard:

Figure 1: H.264 encoder block diagram [7]
H.264/AVC [10], is an open licensed standard, which was developed as a result of the collaboration between the ISO/IEC Moving Picture Experts Group and the ITU-T Video Coding Experts Group. It is one of the most efficient video compression techniques available today. Some of its major applications include video broadcasting, video on demand, MMS over various platforms like DSL, Ethernet, LAN, wireless and mobile networks, etc.

**Sequence characterization**

From one video sequence we can extract two types of information: spatial and temporal, depending on which characteristics we are looking at.

**Temporal information**

**Movement characteristic**

It is easier to conceal linear movements in one direction because we can predict pictures from previous frames (the scene is almost the same). If we movements in many directions or scene cuts, find a part of previous frame that is similar is going to be more difficult, or even impossible
e.g. in case of scene cuts. Here, a sequence of five frames are seen, with a step of three frames between every one, of three different sequences: a football match, a village panorama and a music video clip. In the music video sequence we have two scene cuts in the same amount of frames than the village sequence, where we have a smooth movement in one direction. Obviously, it will be easier to conceal the village sequence.

**Speed characteristic**

The slower is the movement of the camera, the easier will be to conceal an error. We can see an example of two different video speeds if we compare the village sequence with the football sequence.

![Figure 3: Movement and speed [2]](image)

**Spatial information**

**Smoothness of the neighborhood**

The smoothness of the neighborhood of the erroneous macroblock will determine the difficulty of the spatial concealment. Here, we see three cases. In the first one it is going to be easy to reconstruct the lost macroblock because the neighborhood is very uniform (smooth) with almost no difference between the neighboring macroblocks. In the second situation, it is going to be a
little bit more difficult; we have to look for the edges and then, recover the line. The third case is an example where the neighbors cannot help us to recover the macroblock because they do not give any information about the lost part (in this case, the eye).

![Figure 4: Smoothness of the neighborhood](image)

**Error Characteristics**

- Lost information
- Size and form of the lost region
- I/P frame

If the error is situated in the I frame, it is going to affect more critically the sequence because it will affect all the frames until the next I frame and I frames do not have any other reference but themselves. If the error is situated in a P frame it will affect the rest of the frames until the next I frame but we still have the previous I frame as a reference.

**Error Concealment Techniques:**

The main task of error concealment is to replace missing parts of the video content by previously decoded parts of the video sequence in order to eliminate or reduce the visual effects of bit stream error. Error concealment exploits the spatial and temporal correlations between the neighboring image parts within the same frame or from the past and future frames.
The various error concealment methods can be divided into two categories: error concealment methods in the spatial domain and error concealment methods in the time domain.

Spatial domain error concealment utilizes information from the spatial smoothness nature of the video image. Each missing pixel of the corrupted image part is interpolated from the intact surroundings pixels. Weighted averaging is an example of a spatial domain error concealment method.

Temporal domain error concealment utilizes the temporal smoothness between adjacent frames within the video sequence. The simplest implementation of this method is replacing the missing image part with the spatially corresponding part inside a previously decoded frame, which has maximum correlation with the affected frame. Examples of temporal domain error concealment methods include the copy-paste algorithm, the boundary matching algorithm and the block matching algorithm.

**Spatial Error Concealment:**
All error concealment methods in spatial domain are based on the same idea which says that the pixel values within the damaged macroblocks can be recovered by a specified combination of the pixels surrounding the damaged macroblocks. In this technique, the interpixel difference between adjacent pixels for an image is determined. The interpixel difference is defined as the average of the absolute difference between a pixel and its four surrounding pixels. This property is used to perform error concealment.

The first step in implementing spatial based error concealment is to interpolate the pixel values within the damaged macroblock from four next pixels in its four 1-pixel wide boundaries. This method is known as ‘weighted averaging’, because the missing pixel values can be recovered by calculating the average pixel values from the four pixels in the four 1-pixel wide boundaries of the damaged macroblock weighted by the distance between the missing pixel and the four macroblocks boundaries (upper, down, left and right).
The formula used for weighed averaging is as follows [2]:

$$mb(i, k) = \frac{1}{d_L + d_R + d_T + d_B} [d_R mb_L(i, 2N) + d_L mb_R(i, l) + d_B mb_T(2N, k) + d_T mb_B(l, k)]$$

where \(i, k = 1, 2, 3, \ldots, N\).

- \(d_L\) : distance between the interpolated pixel and the nearest pixel \(mb_L = mb(i, 0)\) in left boundary.
- \(d_R\) : distance between the interpolated pixel and the nearest pixel \(mb_R = mb(i, N + 1)\) in right boundary.
- \(d_T\) : distance between the interpolated pixel and the nearest pixel \(mb_T = mb(0, j)\) in top boundary.
- \(d_B\) : distance between the interpolated pixel and the nearest pixel \(mb_B = mb(N + 1, j)\) in bottom boundary.
- \(N \times N\) : Size of the block.

**Temporal Error Concealment:**

It is easier to conceal linear movements in one direction because pictures can be predicted from previous frames (the scene is almost the same). If there are movements in many directions or scene cuts, finding a part of previous frame that is similar is more difficult, or even impossible.
**Copy paste Algorithm:**

It replaces the missing image part with the spatially corresponding part inside a previously decoded frame, which has maximum correlation with the affected frame.

![Copy Paste Algorithm Diagram](image)

**Figure 6: Copy paste algorithm [1]**

**Boundary matching:**

Let B be the area corresponding to a one pixel wide boundary of a missing block in the nth frame $F_n$. Motion vectors of the missing block as well as those of its neighbors are unknown. The coordinates $[\hat{x}, \hat{y}]$ of the best match to B within the search area A in the previous frame $F_{n-1}$ have to be found. The equation used is as follows: [1]

$$[\hat{x}, \hat{y}] = \arg \min_{x,y \in A} \sum_{i,j \in B} |F_{n-1}(x+i,y+j) - B(i,j)|. \tag{2}$$

The sum of absolute differences (SAD) is chosen as a similarity metric for its low computational complexity. The size of B depends on the number of correctly received neighbors M, boundaries of which are used for matching.
Block matching:
Better results can be obtained by looking for the best match for the correctly received MB on top, bottom, left or right side of the missing MB. The equation used is as follows: [1]

$$[\hat{x}, \hat{y}]_D = \arg \min_{x,y \in A_D} \sum_{i,j \in MB_D} |F_{n-1}(x + i, y + j) - MB_D(i, j)|$$

(3)

where ‘A_D’ represents the search area for the best match of MB_D, with its center spatially corresponding to the start of the missing MB.

The final position of the best match is given by an average over the positions of the best matches found for the neighboring blocks, computed as follows: [1]

$$\hat{x} = \frac{1}{M} \sum_{D} \hat{x}_D; \quad \hat{y} = \frac{1}{M} \sum_{D} \hat{y}_D.$$  

(4)

The MB sized area starting at the position [\hat{x}, \hat{y}] in F_{n-1} is used to conceal the damaged MB in F_n. To reduce the necessary number of operations, only parts of the neighboring MBs can be used for the MV search.
Quality Metrics:

An objective image quality metric can play a variety of roles in image processing applications. First, it can be used to dynamically monitor and adjust image quality. For example, a network digital video server can examine the quality of video being transmitted in order to control and allocate streaming resources. Second, it can be used to optimize algorithms and parameter settings of image processing systems. Third, it can be used to benchmark image processing systems and algorithms. In this project the following quality metrics are used.

i. Peak Signal to Noise Ratio (PSNR)
ii. Distortion Artifacts
iii. Spatial Information (SI) & Temporal Information (TI)
iv. Structural Similarity Index Metric (SSIM)
v. Bjontegaard Distortion – Bit Rate (BD-BR)
vi. Bjontegaard Distortion – PSNR (BD-PSNR)

Peak Signal to Noise ratio (PSNR)

In scientific literature it is common to evaluate the quality of reconstruction of a frame $F$ by analyzing its peak signal to noise ratio (PSNR). There are different ways of calculating PSNR. One is frame-by-frame and the other is the overall average.

The Joint Model reference software outputs PSNR for every component $c$ of the YUV color space for every frame $k$. The PSNR for an 8 bit PCM (0-255 levels) is calculated using: [1]
\[ PSNR^{(c)}_k = 10 \cdot \log_{10} \frac{255^2}{MSE^{(c)}_k} [dB] \] (5)

Where, \( PSNR_k \) is the PSNR for the \( k^{th} \) frame and \( MSE_k \) is the mean square error of the \( k^{th} \) frame, given by: [1]

\[ MSE^{(c)}_k = \frac{1}{M \cdot N} \sum_{i=1}^{N} \sum_{j=1}^{M} [F(i, j) - F_0(i, j)]^2 \] (6)

Where, \( N \times M \) is the size of the frame, \( F_0 \) is the original frame and \( F \) is the current frame. The average PSNR is calculated using: [1]

\[ PSNR^{(c)}_{av} = \frac{1}{N_{fr}} \sum_{k=1}^{N_{fr}} PSNR^{(c)}_k \] (7)

Where, \( N_{fr} \) is the number of frames and \( PSNR_k \) is the PSNR for the \( k^{th} \) frame.

**Distortion Artifacts**

Here measurement of distortion artifacts like blockiness and blurring is done. Blockiness is defined as the distortion of the image characterized by the appearance of an underlying block encoding structure [1]. This metric compares the power of blurring of two images. If the value of the metric for first picture is greater, than the value for the second picture, it means that second picture is more blurred, than first. On the other hand, blurriness is defined as a global distortion over the entire image, characterized by reduced sharpness of edges and spatial detail [1]. This metric was created to measure the visual effect of blocking. If the value of the metric for first picture is greater, than the value for the second picture, it means that first picture has more blockiness, than the second picture.
Spatial and temporal Information

Spatial and temporal information of video sequences play a crucial role in determining the amount of video compression that is possible, and consequently, the level of impairment that is suffered when the scene is transmitted over a fixed-rate digital transmission service channel. Spatial and temporal measures that can be used to classify the type of a sequence are presented in order to assure appropriate coverage of the spatial-temporal plane in subjective video quality.
tests. Spatial and temporal information of video sequences tell us the amount of video compression possible and the level of impairment that is suffered during transmission.

The **Spatial Information** (SI) is based on the Sobel filter [1]. The Sobel filter generates an image emphasizing the edges. Each video frame $F_n$ at time $n$ is first filtered with the Sobel filter ($Sobel (F_n)$). Next, the standard deviation over the pixels ($std_{space}$) in each Sobel-filtered frame is computed. This operation is repeated for each frame in the video sequence and results in a time series of spatial information of the scene. The mean value in the time series ($mean_{time}$) is chosen to represent the spatial information content of the scene. It can be measured using: [1]

$$SI = mean_{time}\{std_{space}[Sobel (F_n)]\}$$  \hspace{1cm} (8)

The Temporal Information (TI) is based upon the motion difference feature $M_n(i,j)$, which is the difference between the pixel values at the same location in space but at successive frames. It can be measured using: [1]

$$M_n(i,j) = F_n(i,j) - F_{n-1}(i,j)$$  \hspace{1cm} (9)

where $F_n(i,j)$ is the pixel at the $i^{th}$ row and $j^{th}$ column of the $n^{th}$ frame in time.

The measure of TI is computed as the mean time ($mean_{time}$) of the standard deviation over space ($std_{space}$) of $M_n(i,j)$ over all $i$ and $j$: [1]

$$TI = mean_{time}\{std_{space}[M_n(i,j)]\}$$  \hspace{1cm} (10)

More the motion in adjacent frames, higher the values of TI.

**Structural Similarity Index Metric (SSIM)**

The main function of the human visual system (HVS) is to extract structural information from the viewing field, and HVS is highly adapted for this purpose. Therefore, a measurement of structural information loss can provide a good approximation to perceived image distortion. SSIM compares local patterns of pixel intensities that have been normalized for luminance and contrast. The luminance of the surface of an object being observed is the product of the illumination and the reflectance, but the structures of the objects in the scene are independent of
the illumination. Consequently, to explore the structural information in an image, the influence of illumination must be separated.

Let \( x \) and \( y \) be two image patches extracted from the same spatial location of two images being compared. Let \( \mu_x \) and \( \mu_y \) be their means and \( \sigma_x^2 \) and \( \sigma_y^2 \) be their variances. Also, let \( \sigma_{xy}^2 \) be the variance of \( x \) and \( y \). The luminance, contrast and structure comparison are given by: [1]

\[
\begin{align*}
    l(x, y) &= \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \quad (11) \\
    s(x, y) &= \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \quad (12) \\
    c(x, y) &= \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \quad (13)
\end{align*}
\]

Where \( C_1 \), \( C_2 \) and \( C_3 \) are all constants given by: [1]

\[
C_1 = (K_1L)^2, \quad C_2 = (K_2L)^2 \quad \text{and} \quad C_3 = C_2/2 \quad (14)
\]
L is the dynamic range of the pixel values (L = 255 for 8 bits/pixel gray scale images), and $K_1 \ll 1$ and $K_2 \ll 1$ are scalar constants. The general SSIM can be calculated as follows: [1]

$$\text{SSIM}(x,y) = [l(x,y)]^\alpha \cdot [c(x,y)]^\beta \cdot [s(x,y)]^\gamma$$

(15)

Where $\alpha, \beta$ and $\gamma$ are parameters which define the relative importance of the three components.

**Generation of errors**

This is done by modifying the function “decode one slice” that is found in the “image.c file” of the decoder source code. The purpose of this function is, as its name says, decoding one slice. The operation is quite simple: it takes a slice, reads macroblocks successively from the bitstream and decodes them by calling the function “decode one macroblock”. When the flag ”end of slice” gets the value ”TRUE” we go out of the function until the next slice needs to be decoded.

The error is generated in the frames of the video sequence randomly with a uniform distribution. When a new slice is detected (every time the function “decode one slice” is called), a random threshold number from 0 to 99 is generated. Then, we compare this value with the error rate per slice we want to introduce (we took it from the “decoder.cfg” file as a percentage). If the generated value is lower than the error rate, the whole slice is treated as erroneous. Here, instead of calling the function “decode one macroblock”, the selected error concealment method will be used to conceal the slice.

**Error input by command line**

We have seen that, to introduce an error rate per slice, we have to write the required percentage in the “decoder.cfg” file from where it is compared with the random threshold generated. The problem is that C generates random numbers using pseudo-random sequences. There the sequence of random numbers will always be the same.
Figure 12: Akiyo without and with error [1]

Figure 13: Fussball without and with error [1]
Thus a number called ‘seed’ is used. If the seed is not modified, the sequence of random numbers would be the same and, therefore, we would have the errors in the same parts of the sequence. On one hand we are going to use the same seed every time we want to compare different error concealment methods because we want to be fair in the comparison. On the other hand, we want to simulate errors in different parts of the sequence, so we are going to conceal every sequence with different seeds.
Standard way of running the decoder is: [1]

```
1decd.exe <Configuration_file>
```

Example:
```
1decd.exe decoder.cfg
```

Modified way: [1]

```
1decd.exe -s <Seed> <Configuration_file>
```

Example:
```
1decd.exe -s 4 decoder.cfg
```
References:


