ERROR CORRECTION SCHEME FOR UNCOMPRESSED HD VIDEO OVER WIRELESS

M. Manohara, R. Mudumbai, J. Gibson and U. Madhow
Department of Electrical & Computer Engineering
University of California Santa Barbara CA 93106

ABSTRACT

Digital transmission of uncompressed high-definition video is challenging because of its high data rate and its extreme sensitivity to bit errors. In this paper we propose a simple error correction scheme to reduce the bit error effects in the video at the receiver end of a wireless channel. Our scheme uses the large amount of spatial redundancy already present in uncompressed HD video data to provide an extra layer of protection in addition to that provided by channel coding. Thus, our method requires no change to the video signal being transmitted and is compatible with any existing solution for HD video transmission. Using simulations over a range of byte error rates, we show that our scheme effectively reduces the number of visible artifacts because of uncorrected channel errors, and provides approximately 7 dB improvement in peak signal to noise ratio.

Index Terms - Uncompressed HD video, Wireless HDTV, Byte Error Rate

1. INTRODUCTION

We consider the problem of transmitting an uncompressed high definition (HD) video stream over a wireless connection. This problem is motivated by the application scenario illustrated in Figure 1, where a HD video source such as a Blu-ray disc is decrypted and decoded in a device that then transmits the uncompressed video stream wirelessly to a HD display. This is attractive because it offers a way to minimize wiring in a home theater setting and also provides a single universal and standard interface to any HD video display, thus simplifying installation of new devices [1].

HDTV (High Definition Television) has become popular because it has higher resolution than traditional television systems and an aspect ratio of 16:9 influenced by widescreen cinema. Presently, the data rate of uncompressed HD video can be as high as 3.0 Gbps (1080p60, RGB444 pixel format at 8 bits per pixel). In the future, the data rate is expected to rise with increases in resolution and color depth.

Since HD video sources require such large data rates, it may seem reasonable to compress the video stream before transmitting. However, compression introduces delay, reduces video quality and increases complexity at the transmitter (video source) and receiver (HDTV). Also, the video display will have to support multiple codecs to maintain inter-operability with the various video sources. Furthermore, the HDMI connector interface for HD video sources and displays supports uncompressed HD transmission. Motivation to transmit uncompressed HD video and more information on disadvantages related to compression can be found in [2-5].

Random bit errors (especially in the most significant bits of a pixel) in an uncompressed HD frame are typically visible to the naked eye as "sparkles." Given the high user expectations for HDTV, it is critical that decoding failures in the digital communication scheme used to convey these bits be masked from the user. This is especially important in applications where retransmission-based error recovery is not possible or desirable (e.g., when streaming to multiple screens). In this paper, we examine whether we can use the large amounts of redundancy in the uncompressed video to obtain low-complexity error resilience techniques to complement the error correction provided by channel coding. For the example system considered, the proposed scheme reduces the number of errors in the most significant bits by an order of magnitude, and achieves about 7 dB gain in PSNR.

While our approach is of general applicability, we are motivated by the recently developed Wireless HD (WiHD) standard, which supports the transmission of uncompressed HD content in the unlicensed 60 GHz frequency band [1], as shown in Figure 1.

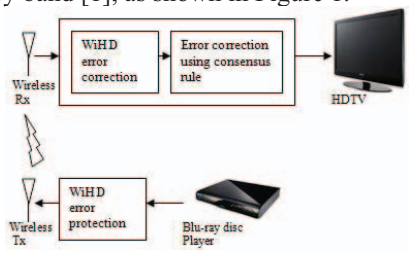


Figure 1: HD video over wireless. Uncompressed HD (RGB 444) videos are transmitted wirelessly from transmitter to receiver.

While the 7 GHz of spectrum in this band makes it an attractive choice for bandwidth-hungry applications, a key challenge is the susceptibility to blockage by humans

and furniture at these small carrier wavelengths. One approach to this problem is to use electronic beamsteering to steer around obstacles, using reflections from walls or the ceiling. This introduces a significant amount of variability in the signal-to-noise ratio, making it important to employ error resilience at the source coding layer to minimize quality fluctuations.

The remainder of this paper is organized as follows. In Section 2, we describe the details of the wireless setup. Section 3 motivates and describes our error correction algorithm, and Section 4 presents results to illustrate the effectiveness of this technique. We conclude in Section 5 with a brief discussion of future work.

2. COMMUNICATION MODEL

We use a simple model for the communication system which is described as follows: uncompressed HD video is encoded using an inner code such as convolutional or turbo-like code, with an outer Reed-Solomon (RS) code to clean up residual errors and to detect large error bursts. We assume that interleavers are used after both the inner and the outer code, so that any errors in the detected codewords will get dispersed as random bit errors uniformly distributed throughout the video frame. This is similar to the error correction code specified in the WiHD standard [1].

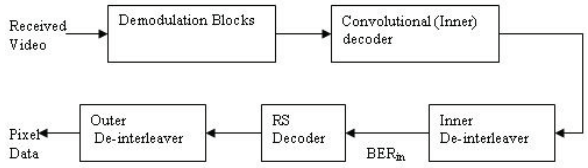


Figure 2: Model for error correction code for HD video. We use BER_{in} to indicate the byte error rate/ symbol error rate.

Rather than simulate in detail a particular inner code, we parameterize its performance by the symbol error probability, BER_{in}, seen by the outer RS code. We use the same outer code used in the WiHD specification i.e. the RS (224,216) code that is capable of correcting up to 4 byte errors. When there are more than 4 byte errors in a block, the RS decoder is unable to correctly decode the codeword. In that case, there are two possibilities: the RS decoder may either declare a decoding failure, or it may decode to a wrong codeword. The latter possibility is extremely rare in practice and we neglect it, i.e. we assume that whenever there are more than 4 byte errors, it always leads to a decoding failure. Further, we assume that the RS code is systematic; that is, the information symbols are visible in the codeword. Thus, upon decoding failure, the decoder simply outputs the information symbols as received, flagging them as suspect; the overall effect of channel errors is to produce random residual bit errors after the RS decoder all of which are flagged as possible bit errors by the decoder.

Our goal is to use the redundancy in the uncompressed HD video to correct the flagged bits that are in error, using their correlation with their neighbors. Given that the raw channel bit error rate is quite small in a well designed system, even flagged bits have a high probability of being right; this is because an RS decoder failure usually involves only 5 or 6 byte errors, however the decoder flags all 216 bytes as suspect. Our proposed error resilience technique is therefore conservative, treating a marked bit as correct unless established to be incorrect beyond reasonable doubt. This is in contrast to a related work [2], where all bits corresponding to a failed codeword are replaced using information from adjacent pixels; such an aggressive use of spatial redundancy can result in wrongly flipping correct bits.

3. THE ERROR CORRECTION TECHNIQUE

We consider 1080p25 HD videos with frame size of 1920x1080 pixels encoded as RGB444 with 8 bits per pixel. In an 8-bit pixel, the four most significant bits (MSBs) are more important to protect than the four least significant bits (LSBs), since errors in the four MSBs give rise to visible artifacts.

Our error correction technique is based on the fact that most of the pixels in a HD video frame are very similar to the spatially adjacent pixels. We analyzed some HD video frames to quantify the amount of spatial redundancy. We found that around 95% of the pixels in a frame match the pixels to their north, south, east and west in the first MSB. Note that when the MSB in the pixels do not match, it does not make sense to look for matches among less significant bits. Among the pixels where the first MSB matches its neighbors, about 91% match in the second MSB as well. Similarly, among the pixels where the first two MSBs match, around 85% match in the third MSB, and around 70% match in the fourth MSB given that the first three MSBs match. The match varies between 10-50% for the four LSBs, which is very low when the total number of pixels in the frame is considered. Hence, we attempt to correct as many bit errors as possible in the four MSBs, and, in keeping with our conservative approach, do not attempt to correct the LSBs. We now use this redundancy to design a simple error correction scheme.

Rule for comparison	Surrounding pixels do not match in 4 MSBs	Surrounding pixels match and is equal to the center pixel in 4 MSBs	Surrounding pixels match and is not equal to the center pixel in 4 MSBs
All	45.91%	53.21%	0.86%
Majority	5.48%	83.68%	10.83%

Table 1: Comparison of 'ALL in consensus' rule and 'MAJORITY in consensus' rule.

Table 1 compares two candidate methods: the 'All in consensus' and 'Majority in consensus' rules. Even though 'Majority in consensus' appears to work well because it has almost twice the amount of 'surrounding pixels match and equal' to the center pixel in four MSBs compared to 'All in consensus', it has almost ten times 'surrounding pixels match and not equal' to the center pixel in the four MSBs. In fact, 'Majority in consensus' rule actually results in increased number of bit errors after correction. Since our application requires a highly conservative rule for bit flipping, we choose the 'All in consensus' rule for our algorithm. The algorithm works as follows:

1. For every video frame, failure of the RS decoder indicates the bits that are marked as possibly in error.
2. We check if the previous significant bit positions have equal values in all the four surrounding pixels and in the current pixel.
3. If the previous bits match in all the five pixels under consideration, we check the current marked bit position, in the four surrounding pixels.
4. If the values of the bit position checked in the four surrounding pixels are ALL equal, we say that these bits are in consensus.
5. If the value of the consensus bits is different from the value of the marked bit, we flip the marked bit.
6. If the value of the consensus bits is same as the value of the marked bit, we do not change the value of the marked bit.
7. We do not try to correct marked bits in any of the four LSBs.

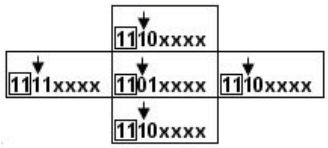


Figure 4: 'ALL in consensus rule' is used for error correction. The bit position 5 (third MSB) of the center pixel is marked.

In the example shown in Figure 4, the 3rd significant bit in the center pixel (0 in 1101xxxx) is marked as a potential bit error. Checking the first 2 MSBs in the four surrounding pixels, we find that they match the current pixel ('11' as shown in the example). The current bit (marked by the arrow) is now checked against the surrounding pixels. Since they are all equal (1 in this case), we have a consensus. However, the consensus differs from the value of the marked bit (0 in this case), so we flip this bit. Hence, the value of this pixel after correction is 1111xxxx.

4. RESULTS

We use five frames of the standard 1080p25 HD videos, with YUV420 pixel format: Pedestrian, Riverbed, and Rush Hour for simulations [6]. First, we convert these YUV 420 videos to RGB 444 videos. Then, we simulate the effect of the channel by introducing random bit errors in each frame, the number of bit errors being determined by BER_{in} as follows:

1. We divide the video frame into blocks of 216 bytes to which 8 parity bytes are added by the RS encoder, to create a "channel block" of 224 bytes each. The 216 bytes are mapped to random pixels in the video frame to account for the interleaving.
2. Given BER_{in}, the byte error rate at the input of the RS code, we generate a random number, b of byte errors in a block of 224 incoming bytes. If b>=5, we have a RS decoder failure, and we flag every bit in all 216 payload bytes as possible errors. In addition, we randomly choose b bytes out of the block, and randomize every bit in these bytes to model the uncorrected errors.
3. We repeat Step 2 for every channel block in the frame.

We then apply the 'ALL in consensus' rule to correct the bit errors in the video frame. While we do not present numerical results due to lack of space, we note that the proposed scheme significantly reduces the number of errors in the four MSBs (relative to the number after RS decoding or before correction). For the first MSB, the probability that a randomly placed bit error is left uncorrected is roughly equal to the probability that its four neighbors do not agree, which we have previously observed to be about 5%. Thus, we expect the number of bit errors in the first MSB to be reduced by an order of magnitude after applying the all-in-consensus rule, and this is what we find in our simulations. The improvements in the second, third and fourth MSBs are progressively smaller, corresponding to the smaller proportion of matches between all four neighbors.

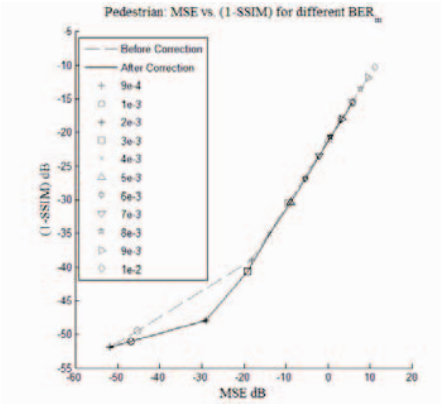
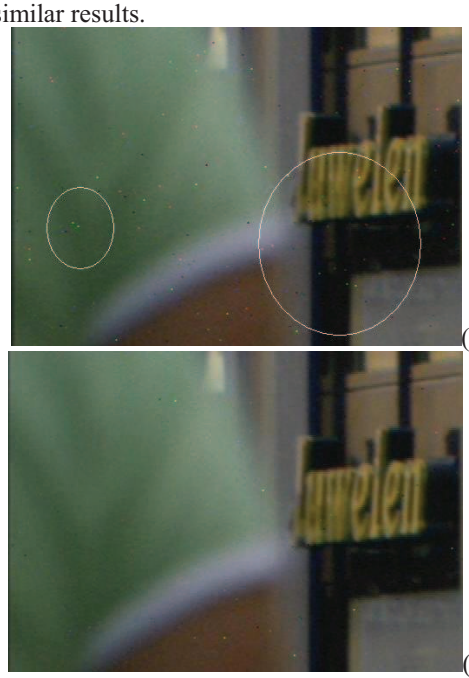


Figure 5: MSE vs. (1-SSIM) dB, before and after correction for $BER_{in} = 9x10^{-4}$ to $1x10^{-2}$, exhibit a linear relationship.

We now present some results (both visual and quantitative) to illustrate the improvement in video quality resulting from the reductions in MSB errors. We convert the

RGB 444 videos back to YUV 420 videos to calculate the various error measures. A popular quantitative indicator is the peak signal-to-noise ratio (PSNR). Another error measure that is closer to the Human Visual System (HVS) is the Structural Similarity Index (SSIM) that quantifies the subjective scores [7]. While it is well-known that PSNR fails to capture certain types of distortions, it is adequate for our application because we only have randomly distributed bit errors. To see this, we use the MSU video quality measurement tool [8] to plot log MSE vs. log (1-SSIM) for the video frame Pedestrian with random bit errors; see Figure 5. In this tool, if the error between the frames being compared is zero, then the PSNR is normalized to 100 dB or MSE = 6.5025×10^{-6} (approximately zero) and the SSIM is equal to 1. It can be seen that the MSE and SSIM are strongly correlated over a range of BER_{in} before and after correction. Two other videos, Riverbed and Rush Hour, yield similar results.



(a)

(b)

Figure 6: (a) An enlarged portion of Pedestrian frame before correction. (c) An enlarged portion of Pedestrian frame after correction.

We now present some visual results. A snapshot of Pedestrian is shown in Figure 6(a). We simulated the channel at byte error rate of 7×10^{-3}. Figures 6(b) and 6(c) show the enlarged top left corner of the received frame before and after correction; notice the reduction in sparkles due to the correction.

Next, we simulated the channel at various byte error rates ranging from 9×10^{-4} to 1×10^{-2}. The PSNR plot at various byte error rates is shown in Figure 7: the improvement after correction is about 7 dB at byte error rates that are high enough for a significant number of bit errors at the output of the RS decoder.

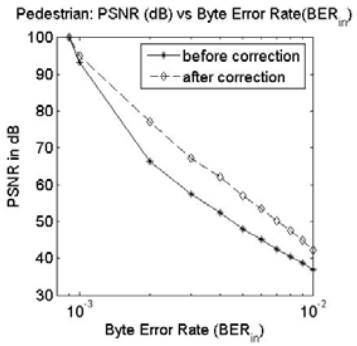


Figure 7: PSNR (dB) vs. Byte Error Rate (BER_{in}) before and after correction (around 7dB improvement on average)

5. FUTURE WORK

The proposed error correction using consensus rule takes advantage of the spatial redundancy in the video frames and improves PSNR by approximately 7dB. It is of interest to explore techniques that also use temporal redundancy, combination of consensus and averaging approaches to correct or smooth LSBs. Finally, it is important to explore a broader set of options for HD over wireless, including the design of "lightweight" compression schemes that retain the advantages of uncompressed HD while utilizing bandwidth more efficiently.

6. REFERENCES

[1] WirelessHD, "WirelessHD Specification Version 1.0 Overview," *WirelessHD*, October 2007. [Online]. Available: http://www.wirelesshd.org.

[2] H. Singh, et al., "MAC 23-2 – Supporting Uncompressed HD Video Streaming Without Retransmissions Over 60GHz Wireless Networks," in *Wireless Communications and Networking Conference*, 2008, pp. 1939-1944.

[3] H. Singh, X. Qin, H. Shao, C. Ngo, C. Kwon, S.S. Kim, "Support of Uncompressed Video Streaming Over 60GHz Wireless Networks," in *Consumer Communications and Networking Conference*, 2008, pp. 243-248.

[4] H. Shao, et al., "Adaptive Multi-beam transmission of Uncompressed Video over 60GHz Wireless Systems," in Future Generation Communication and Networking, 2007, pp. 430-435.

[5] N.Geri, "Wireless HDTV–Compressed or Uncompressed? That is the question," AMIMON Ltd, November 2006.

[6] M. Alvarez, "HD - VideoBench. A Benchmark for Evaluating High Definition Digital Video Applications," June 2007. [Online]. Available: http://personals.ac.upc.edu/alvarez/hdvideobench/install.html

[7] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol.13, no. 4, pp. 600-612, April 2004.

[8] MSU Graphics and Media Lab (Video Group), *MSU Video Quality Measurement Tool*. [Online]. Available: http://compression.ru/video/quality_measure/video_measurement_tool_en.html