Bit Error Recovery in MMR Coded Bitstreams Using Error Detection Points

Hyunju Kim and Abdou Youssef
Department of Computer Science
The George Washington University
Washington, DC, USA
Email: {hkim, ayoussef}@gwu.edu

Abstract

This paper investigates the relationships between single-bit errors and their detection points, which occur in bitstreams that have been compressed with Group 3, Extended 2 Dimensional MMR coding scheme, and develops a bit error recovery algorithm using the detection points. When single-bit errors occur in an MMR coded bitstream, the bitstream cannot, without some recovery, be correctly decoded after the errors. Our bit error recovery algorithm applies bit inversions and re-decoding to a specific set of candidate error bits within the error region, which have been determined by the algorithm. The testing results show that around 70% of multiple single-bit errors are corrected with the algorithm.

Keywords: Error detection, Bit error recovery, MMR coding, Facsimile image

1. Introduction

The amount of data traffic over communication lines is vast and growing daily, in part because images consist of massive volumes of data. As a result, images are often compressed. Since nearly all communication media are noisy and incur error, the impact of noise is very serious on compressed data and the errors are hard to detect and recover from. Yet, without error recovery, compressed data cannot be decoded.

Most receivers solve this problem by requesting a retransmission of the corrupted data. But retransmission adds more traffic to the networks as well as requires more time and resources. Even worse, retransmission is not possible in surveillance applications. In case retransmission request is not desirable or possible, the decoder should try to recover as much lost data as possible with received data only.

Most of the research on error recovery has focused on error concealment methods for lossy block-based DCT compression as in JPEG or MPEG [1,2,7]. These methods assume that a bitstream has been fully decoded and the position of the erroneous block is known. For error recovery in losslessly compressed bitstream, which is relevant to this paper, relatively little research has been done [6,8]. In [8], a generic error recovery scheme is presented where the error patterns are assumed to be known, which is the case in the older modems. In the more recent modems, the error patterns are unknown. In our research, the error patterns are assumed to be unknown. In [6], a facsimile error recovery is presented without assuming any knowledge of error patterns, but the image in their research is not compressed in MMR, which is used in most current facsimile machines. Our research is the first address error recovery in MMR coded bitstreams. In two other papers [4,5], we developed symbol error recovery in MMR fax over modems. In this paper, we address multiple single-bit error recovery in MMR. We define a single-bit error as an isolated wrongly flipped bit, which usually occurs in fax over IP. Thus, this paper assumes the following error model:

- The decoder cannot request retransmission of corrupted data.
- There is no error resiliency tool or code provided by the encoder or network channel.
- Bitstreams are coded with MMR code.
- The errors are single-bit errors.

The next section provides a brief overview of MMR coding. Section 3 provides behaviors of error detection points in MMR coded bitstreams, and section 4 presents an error recovery algorithm. Section 5 provides the performance of the approach, and section 6 provides conclusions.

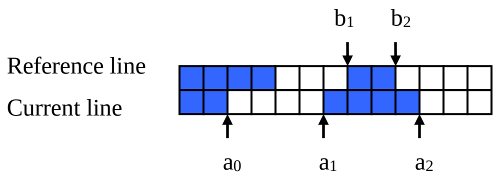
2. Extended 2 dimensional MMR code

The majority of facsimile machines in the world apply Extended 2 Dimensional MMR (Modified Modified READ) coding to compress facsimile images. This scheme is used for Group 3 and Group 4 facsimile images and defined at ITU (formerly CCITT) T.4 and T.6 recommendations [3].

MMR makes use of vertical relationships between black runs and white runs in two adjacent scan lines. It codes an image one scan line (1728 pixels) at a time and uses the previous line to code the current scan line being coded. Figure 1 shows the elements of MMR coding.

In MMR, when coding the topmost line, the Reference line is assumed to be an imaginary blank line above the page. Also, when coding a Current line, the initial value of the first changing element is taken to be an imaginary blank pixel left of the line. The position of a_0 changes according to the rules in Figure 1, and the position of a_1, a_2, b_1, and b_2 change in a way that is consistent with their definitions relative to a_0.

When an error occurs in an MMR coded bitstream, the error propagates through the bitstream because <u>MMR coding scheme does not provide any resynchronization point</u>. As a result, the decoder produces a corrupted image beyond the error position.



a_0: The position of the first changing element on the current line.
a_1: The next changing element to the right of a_0 on the current line.
a_2: The next changing element to the right of a_1 on the current line.
b_1: The first changing element in the reference line to the right of a_0 and of opposite color of a_0.
b_2: The next changing element to the right of b_1 on the reference line.

(a) Changing picture elements

Coding Modes	Conditions	Codes	Next values of a_0
Pass Mode (P Mode)	$a_1 > b_2$	0001	$a_0 := b_2$
Horizontal Mode (H Mode)	$a_1 \le b_2$ & $\lvert a_1 - b_1 \rvert > 3$	001 + M*(a_0a_1) + M*(a_1a_2)	$a_0 := a_2$
Vertical Mode (V Mode)	$a_1 \le b_2$ & $\lvert a_1 - b_1 \rvert \le 3$		
V(0)	$a_1 = b_1$	1	$a_0 := a_1$
$V_L(1)$	$a_1 = b_1 - 1$	010	$a_0 := a_1$
$V_L(2)$	$a_1 = b_1 - 2$	000010	$a_0 := a_1$
$V_L(3)$	$a_1 = b_1 - 3$	0000010	$a_0 := a_1$
$V_R(1)$	$a_1 = b_1 + 1$	011	$a_0 := a_1$
$V_R(2)$	$a_1 = b_1 + 2$	000011	$a_0 := a_1$
$V_R(3)$	$a_1 = b_1 + 3$	0000011	$a_0 := a_1$

* M(a_0a_1) and M(a_1a_2) are Huffman codewords of the lengths ($a_1 - a_0$) and ($a_2 - a_1$). Two Huffman tables are specified in T.4/T.6, one for white runs and one for black runs.

(b) MMR coding and update of a_0

Figure 1. MMR coding

3. Bit error detection points

To detect an error, we use the constraints implied in MMR shown in Figure 1. Each MMR code mode has conditions. When our error recovery system finds any inconsistencies between a code mode and the corresponding conditions, it assumes that an error occurred in the bitstream. Specifically, our error detection is done by checking for the following violations of the constraints:

- No codeword in the code table matches the received bit pattern.
- The Pass mode occurs, but the changing element b_2 is off the right end of the scan line.
- $V_R(i)$, $i = 1$, or 2, or 3 occurs, but the changing element b_1 is off the right end of the scan line.
- $V_L(i)$, $i = 1$, or 2, or 3 occurs, but the changing element are such that $b_1 - a_0 - i \le 0$.
- The Horizontal mode occurs and the run-length a_0a_1 is decoded, but the changing elements are either $| a_1 - b_1 | \le 3$ or $b_2 < a_1$.
- The Horizontal mode occurs and the run-lengths a_0a_1 and a_1a_2 are decoded, but the a_1a_2 is off the right end of the scan line.
- The Horizontal mode occurs and the run-length a_0a_1 is decoded, but the corresponding a_1a_2 does not exist.
- The Horizontal mode occurs and the color of the run-length a_0a_1 is determined, but there is no codeword in the color runs table that matches the received bit pattern.

To investigate the relationships between bit errors and their detection points determined by our detection method, we conducted an extensive test, which introduced randomly generated pairs of bit errors into the ITU test images and measured their Detection Points (DP). The Inter-Error Distance (IED), which is the distance in bits between two adjacent bit errors, was distributed between 0 to 800 bits ($0 < \text{IED} < 800$).

The test results show that around 83% of the error pairs were detected before proceeding to the second bit error because the first bit error violates the constraints. These errors are called *one single-bit errors*. This means that when we detect single-bit error(s), there would be 83% chance to correct the error by using the sequential backward bit-inversion and re-decoding, which is explained next. Our system that implemented the bit-inversion and re-decoding method detects an error, determines the error region, which is practically defined 800 bits backward from the DP (the start of the region is defined by DP – 800 bits), and applies bit-inversion and re-decoding to every bit in the region. The system checks for the violations of the constraints during the re-decoding step to see whether the bit-inversion is valid or not by decoding the next 25 scan lines. If there are any violations, the system re-instates the bit and moves to the next bit. If there is no violation, the bit-inversion is assumed to be valid.

For the remaining 17% of the test cases (called *multiple single-bit errors*), the bit-inversion and re-decoding does not work since there are more than one single-bit errors. In these cases, a brute-force way to correct the error would try all 2^{800} (assuming there are 800 bits in the error region) bit-combinations, which is unfeasible. Fortunately, our testing results show that there are some relationships between IED and DP, which could be used to reduce the number of trials to a realistic range. Figure 2 shows a histogram of the one single-bit errors, which represents the relationship between IED and P(d) that is defined as follows:

$$P(d) = \frac{number\ of\ times\ that\ d \le IED < d+1}{total\ number\ of\ one\ single\text{-}bit\ errors}$$

The average IED is around 410 bits, and the standard deviation of the IEDs on the one single-bit errors is 225 bits. Figure 3 shows a histogram of the multiple single-bit errors. Its average IED is around 260 bits, and the standard deviation of the IEDs is 235 bits. As Figure 2 shows, there is no relationship between IED and P(d) in one single-bit errors. On the other hand, in multiple single-bit errors, there is a strong relationship between IED and P(d) compared to the one single-bit errors. This is because of levels of error impacts and persistence of the impacts to our error detection constraints. If the first bit error causes significant error impacts, it would be detected right after the occurrence with our detection method; if the first bit error causes insignificant error impacts, these impacts will last until the second bit error without causing any violations of the detection constraints. In this case, a small IED is likely to lead to a multiple single-bit error since the first bit error's insignificant error impacts will persist over the small IED.

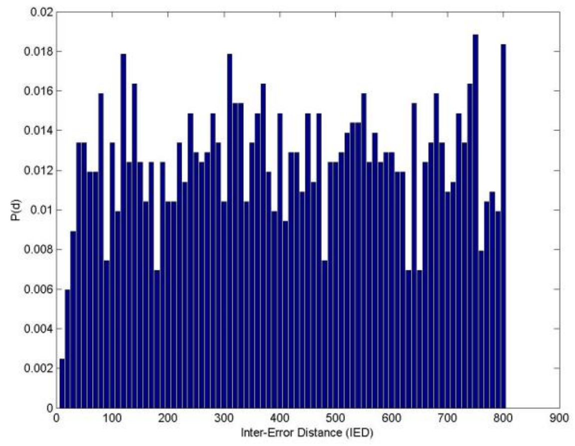


Figure 2. IEDs on one single-bit errors

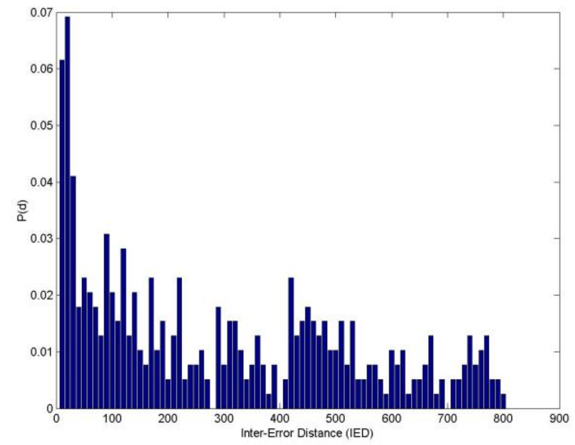


Figure 3. IEDs on multiple single-bit errors

We also measured Detection Latency (DL), which is the length in bits from the latest error bit to the DP, on these two types of single-bit errors. In one single-bit errors, average DL is around 40 bits and its standard deviation is around 75 bits; in multiple single-bit errors, average DL is around 600 bits and its standard deviation is around 4000 bits. The big differences over these values are also because of the levels of error impacts; one single-bit errors cause significant error effects, thus they are detected right after the occurrences; multiple single-bit errors cause insignificant error effects, thus they are not well detected and produce long DLs. In some extreme cases, multiple single-bit errors produce long DLs up to tens of thousands.

Since one single-bit errors are corrected with the bit-inversion and re-decoding, we need to develop a new algorithm to correct multiple single-bit errors. As the multiple single-bit errors have the unique characteristics presented above, we investigated the behaviors of DPs in the errors by measuring three DPs for each pair of multiple single-bit errors as follows:

- DP_{pair}: DP of a pair of bit errors
- DP_{first}: DP of the first bit error only (the second bit error is assumed to be corrected)
- DP_{second}: DP of the second bit error only (the first bit error is assumed to be corrected)

The results show that around 97% of the multiple single-bit errors produce DP_{pair} values such as $DP_{pair} \leq DP_{first}$ or $DP_{pair} \leq DP_{second}$, which indicates that more bit errors produce smaller DLs. We also found the following DP behaviors:

- DP_{pair} is equal to DP_{second} (75% of the multiple single-bit errors)
- DP_{pair} is less than DP_{first} (93% of the multiple single-bit errors)

Based on these observations, we can conclude that (1) correcting the first error bit of a pair of bit errors is likely to produce a DP that is the same as DP_{pair} and (2) correcting the second error bit of a pair of bit errors is likely to produce a DP that is equal to or greater than DP_{pair}. As a result, we can use DP values when we define candidate error bits since correcting error bit(s) produce a DP that is equal to or greater than the original DP in most cases. Our sequential bit-inversion and re-decoding would flip each bit in an error region and produce a set of DP values from the re-decoding. Then, by selecting bits produced DP values, which are equal to or greater than the original DP, we can make up a set of candidate error bits. The next section explains an algorithm that utilizes these DP values.

```
1. Detect a bit error, set $DP_{pair}$ to its DP value, and define its error region.
2. For each bit in the region
        2.1 Flip the bit.
        2.2 Check if the bit-inversion is valid or not by re-decoding the line that
            contains the bit.
                2.2.1 If it is valid, exit the error recovery module and keep decoding.
                2.2.2 If it is not valid, record its DP.
        2.3 If the DP is equal to or greater than $DP_{pair}$
                2.3.1 Create a candidate error bit, $C_i$ ($i$, $DP_i$) where $i$ is
                      the bit address and $DP_i$ is the DP from the flipped bit, $i$.
        2.4 Re-flip the bit
3. For each $C_i$ where its $DP_i$ is equal to $DP_{pair}$
        3.1 Flip the bit, $i$.
        3.2 For each $C_j$ where $j > i$
                3.2.1 Flip the bit, $j$.
                3.2.2 Check if the bit-inversion is valid or not by the re-decoding step.
                        3.2.2.1 If it is valid, go to step 4.
                        3.2.2.2 If it is not valid, re-flip the bit, $j$.
        3.3 Re-flip the bit, $i$.
4. Check if the step 3 corrected the error.
        4.1 If it did not correct the error, call the resynchronization modules.
```

Figure 4. Single-bit error recovery algorithm

4. Bit error recovery algorithm

Our approach to bit error recovery is to (1) detect the error, (2) determine the region of error, and (3) apply bit-inversion and re-decoding to bits within the region. If the third step fails to recover an image from errors, our system tries to find a portion of the bitstream that can be used somehow as a synchronization point. We developed *white-line resynchronization* and *all-H line resynchronization* modules to search for the portions, as described in our paper [4]. These modules skip erroneous regions and resume decoding from the potions that can be used as synchronization points.

In the first step, bit errors will be detected by checking any violations of the detection constraints. As already mentioned, an error region is practically set to 800 bits backward from the DP. Figure 4 shows an algorithm that would be used in the third step assuming that there are at most two single-bit errors in the error region. The algorithm can be modified for more than two bit errors by extending step 3 in Figure 4.

5. Performance evaluation

The error recovery system has been implemented on a Pentium 4, 1.6 GHz using the C programming language. We evaluated the system with 7 facsimile test images provided by

ITU, using what we call the Relative Error (RE) metric, defined as

$$RE = \frac{E_o / S_o}{E_b}$$

where E_o is the total number of erroneous binary pixels in the output image, S_o is the size of the uncompressed image, and E_b is the number of error bits in the bitstream. RE ranges from 0 (best) to 1 (worst).

Test results show that around 70% of multiple single-bit errors are corrected with our algorithm presented in Figure 4. About 90% of the corrected test cases produced 0 values of RE. The remaining 10% of the test cases did not correct errors, but passed the re-decoding checking without any violation detection and produced decoded images where the error is barely noticeable in the output. In these cases, the RE ranges from 2×10^{-6} to 7×10^{-4}, and averages at 10^{-4}.

6. Conclusion

In this paper, behaviors of error detection points are presented, and a single-bit error recovery algorithm using the detection points for MMR coded bitstreams is proposed. We developed an error recovery system, which detects an error and applies the algorithm to correct errors within the error region. Around 70% of multiple single-bit errors are corrected with our algorithm.

If the error cannot be corrected with the algorithm, the system calls the resynchronization modules to find a resynchronization point from which it resumes decoding. The test results show that our algorithm successfully corrects most of single-bit errors.

7. References

[1] Alkachouh, Z., and Bellanger, M., "Fast DCT-Based Spatial Domain Interpolation of Blocks in Images", *IEEE Trans. on Image Processing*, vol. 9, no. 4, Apr. 2000, pp. 729-732.

[2] Hemami, S., and Meng, T., "Transform Coded Image Reconstruction Exploring Interblock Correlation", *IEEE Trans. on Image Processing*, vol. 4, no. 7, Jul. 1995, pp. 1023-1027.

[3] ITU-T Recommendation T.6, Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus, 1988.

[4] Kim, H., and Youssef, A., "Error Recovery in Facsimile without Retransmission", accepted to appear in CATA2003, Mar. 26-28, Honolulu.

[5] Kim, H., and Youssef, A., "Interactive Error Recovery in Facsimile without Retransmission", accepted to appear in ITCC2003, Apr. 28-30, Las Vegas.

[6] Shyu, W., and Leou, J., "Detection and Correction of Transmission Errors in Facsimile Images", *IEEE Trans. on Communications*, vol. 44, no. 8, Aug. 1996, pp. 938-948.

[7] Wang, Y., Zhu, Q., and Shaw, L., "Maximally Smooth Image Recovery in Transform Coding", *IEEE Trans. on Communications*, vol. 41, no. 10, Oct. 1993, pp.1544-1551.

[8] Youssef, A., and Ratner, A., "Error Correction in HDLC without Retransmission", In Proceedings of CISST, vol. 2, 2001, pp. 526-532.