Interactive Error Recovery in Facsimile without Retransmission

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

Abstract

This paper develops and studies interactive techniques for recovery from transmission errors that occur in facsimile images compressed with Group 3, Extended 2 Dimensional MMR coding scheme. When an error occurs in an MMR coded bitstream, the bitstream cannot be decoded after the error point. To prevent losing all information after an error, we develop standalone and interactive error recovery systems that utilize syntactical structure information of the coded bitstream to recover nearly all the data. The standalone recovery works well on text documents or regular graphics, but cannot handle more complex document structures. Interactive recovery, using feedback from a user, can recover from errors in a wide range of document types.

Keywords: Error Detection, Error Recovery, MMR Coding, Facsimile Image, Resynchronization

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 [1,2,3,5,7,8]. 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 [6,9]. In [9], 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. However, their error recovery is not for the coding scheme (MMR) used in most current facsimile machines, whereas our research addresses MMR. Thus, this research 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 symbol errors (multi-byte errors) of unknown patterns.

The next section provides a brief overview of MMR coding. Error detection, resynchronization conditions, and our error recovery approach are presented in section 3. Section 4 provides the performance of the approach, and section 5 provides conclusions.

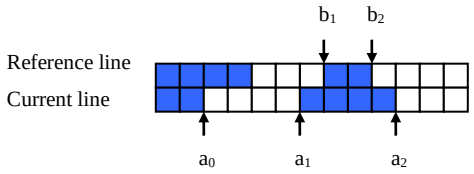
2. Extended 2 dimensional MMR code

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 [4].

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 MMR coding scheme does not provide any resynchronization point. 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. The overall approach to error recovery

The main idea is to detect (and localize) the error, and then find an early resynchronization point (after the error) from which to resume decoding. Since decoding a scan line often requires the previous (i.e., Reference) line, the resynchronization entails finding a location where the Reference line can be "guessed". Alternatively, since some (through not many) lines can be decoded without a reference to the previous line, the resynchronization can be done by locating such a line in the coded bitstream. These steps are summarized as follows:

1. Detect an error through constraint violation detection.
2. Search for a synchronization point.
3. Determine a Reference line (if needed).
4. Resume decoding from the synchronization point.

The major steps of error detection and resynchronization are addressed in the next subsections.

3.1 Error detection

To detect an error in the bitstream, we use the constraints implied in MMR shown in Figure 1. Specifically, 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.
- 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 $b_2 < a_1$.
- $V_R(i)$, $i = 1$, 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 elements are such that $b_1 - a_0 - i \le 0$.

3.2 Resynchronization

Since MMR does not provide any synchronization codeword, we must find a portion of the bitstream that can be used to resume decoding. There are two possible conditions that a portion should satisfy in order to be a synchronization point:

(1) A portion has a guessable Reference line, or
(2) A portion does not need any Reference line for decoding.

Satisfying either condition is enough for a portion to be a synchronization point.

In some cases, resynchronization points can be found algorithmically, while in more complex cases a small amount of feedback information from a user is needed. To illustrate the algorithmic method, we show next how two types of scan lines that can be used for synchronization point: a *white line* (blank line) and a line consisting of H modes only (*all-H line*).

A white line, which satisfies the first condition, can be used as a Reference line for the next line. According to MMR, a white line following a non-white line is coded with one or more P modes followed by one *V(0)* mode: *P ... P V(0)*, represented by the regular expression $P^+ V(0)$. Each white line that follows the first white line is represented by *V(0)*. The first non-white line after the white lines is coded by one or more H modes followed by one of any V modes, which is represented by a regular expression $H^+ V_D(i)$ where *D* is either *L* or *R* and *i* is *0*, or *1*, or *2*, or *3*. In summary, the first type of synchronization point, called *white line resynchronization*, is accomplished by searching in the bitstream for a regular expression of the form $P^+ (V(0))^+ H^+ V_D(i)$.

An all-H line does not refer to the previous line since the H mode explicitly codes the run-lengths of white runs and black runs. This type of line is represented by *H ... H* (that is, H^+). But the pattern is extremely rare since most of the binary images have right and left margins that are represented by V modes in MMR code. Consequently, the expression can be modified by adding two V modes at the beginning, so $V_D(i)\ V_D(j)\ H^+$ will be reflecting the margins, where *D* is *L* or *R* and *i, j = 0, 1, 2*, or *3*. The first V mode represents the right margin of the previous non-all-H line and the second V mode represents the left margin of the current all-H line.

The white line resynchronization works very well for text binary images because in most text documents, many successive blank lines intervene between non-blank lines. As long as a binary image has a blank line, the algorithm is able to resynchronize decoding by the line. The all-H line resynchronization can be used when a binary image has graphic contents.

Still, these two modes of resynchronization do not cover every type of document such as two-column text. To increase coverage, other modes are needed. However, these modes are hard to recognize algorithmically. For that reason, we developed an interactive error recovery system, which is able to receive characteristics information on a target image form the user and invoke different recovery approaches according to the information.

When our interactive error recovery system detects an error, the system presents a GUI to the user. The GUI displays an image, which has been decoded so far from the bitstream, and asks the user to check some of several mode options. Based on the user feedback, the system determines the error mode. Six different error modes have been defined:

- *Text mode*: It indicates a text image that has horizontal spaces (blank lines) between lines.
- *Graphics mode*: It indicates an image has graphics and an error occurs in the middle of them.
- *White-Space mode*: It indicates that an error occurs in a white space region.
- *Vertical-Line mode*: It indicates that an image has a vertical line in the middle of it (as in two-column documents). This line need not go to the bottom of the image.
- *Mixed mode*: It indicates that an image has some text on its left side and some graphics on the right side.
- *Repetitive mode*: It indicates that an image has graphics, and that the image has many repetitive, that is, identical or nearly identical, horizontal lines.

For each mode, we have identified a regular expression to be searched for to locate a resynchronization point, and created a corresponding Reference line or Current line. The following gives the regular expressions that the modes search for in the interactive system:

- Text mode: $P^+ (V(0))^+ H^+ V_D(i)$ (same as the white line resynchronization)
- Graphics mode: $V_D(i)\ V_D(j)\ H^+$ (same as the all-H line resynchronization)
- White-Space mode: $(V(0))^+ H^+ V_D(i)$
- Vertical-Line mode: $P\ V(0)\ (V_D(i))^+ H^+ V_D(i)$
- Mixed mode: $P\ V(0)\ (V_D(i))^+ H^+ (V_D(i))^+$
- Repetitive mode: $(V_D(i))^+ H/P$

Based on the corresponding regular expression, for each mode, an error recovery module has been developed. With the modules pulled together, an error recovery system has been implemented on a Pentium 4, 1.6 GHz using the C programming language and Java 2 SDK 1.4.0.

4. Performance evaluation

Testing has been done with 7 facsimile test images provided by ITU and 8, more selective, sample images. To evaluate the performance of the error recovery system objectively, the modified Error Sensitivity Factor (ESF) metric is defined by

$$\frac{E_p}{E_b \times S_b}$$

where E_p is the total number of erroneous pixels in the output image, E_b is the total number of transmission error bits in an encoded image, and S_b is the size of the bitstream in bits.

Testing results, presented in Table 1, show that our system recovers nearly all the data that would be lost without error recovery. In particular, if a text page has an error, our system recovers the whole page except for at most one text line. Figure 2, 3, and 4 show recovered images from symbol errors with the interactive error recovery system.

Table 1. A set of ESF values (with 32bit-symbol errors)

ITU test images	S_b	Recovery modes	E_p	ESF($\times 10^{-3}$)
No.2	149232	Repetitive	4672	0.9783
		No recovery	15321	3.2083
No.3	96352	Text	4594	1.4900
		No recovery	259972	84.3171
No.4	256048	Text	5146	0.6281
		No recovery	47346	5.7785
Rotated No.4	265360	Text	1703	0.2006
		No recovery	57704	6.7955
No.5	91632	Text	218	0.0743
		No recovery	15730	5.3645
No.6	297392	Text	1135	0.1193
		No recovery	72412	7.6091
No.7	151368	Vertical-Line	529	0.1092
		No recovery	57068	11.7817
No.8	60280	Graphics	306	0.1586
		No recovery	33343	17.2855

5. Conclusion

In this paper, algorithms for error recovery in Extended 2 Dimensional MMR coded bitstream have been proposed. Since MMR coding does not provide any resynchronization points, each algorithm attempts to find a portion in the bitstream that could be used for resynchronization. We developed an interactive system, which incorporates the algorithms. The testing shows that our system recovers well an image from error as long as the image has the characteristics that the algorithms work for.

6. 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] Ismaeil, I., Shirani, S., Kossentini, F., and Ward, R., "An Efficient, Similarity-Based Error Concealment Method for Block-Based Coded Images", *In Proceedings of Image Processing*, vol. 3, 2000, pp. 388-391.

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

[5] Park, J., Kim, J., and Lee, S., "DCT Coefficients Recovery-Based Error Concealment Technique and Its Application to the MPEG-2 Bit Stream Error", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 7, no. 6, Dec. 1997, pp. 845-854.

[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] Sun, H., and Kwok, W., "Concealment of Damaged Block Transform Coded Images Using Projections onto Convex Sets", *IEEE Trans. on Image Processing*, vol. 4, no. 4, Apr. 1995, pp. 470-477.

[8] 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.

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

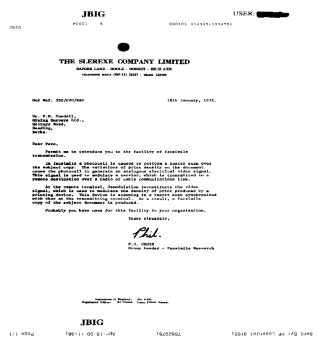


Figure 2(a) Original test image

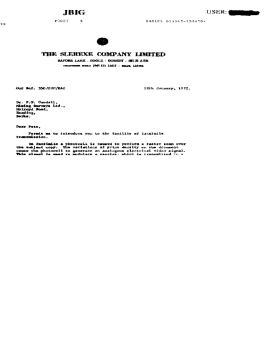


Figure 2(b) Decoded image without error recovery (no data after the error point)

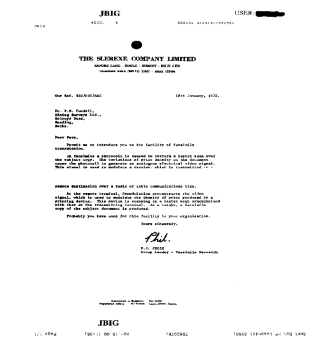


Figure 2(c) Recovered image with the Text Mode

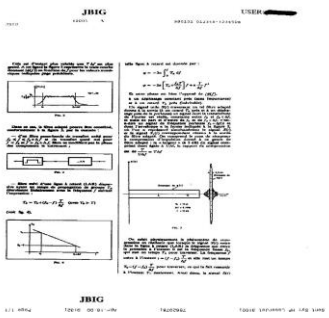


Figure 3(a) Original test image

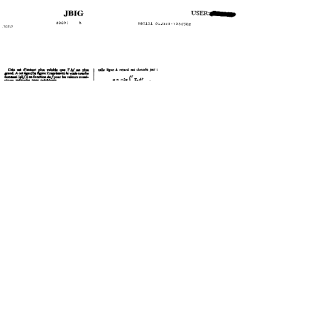


Figure 3(b) Decoded image without error recovery

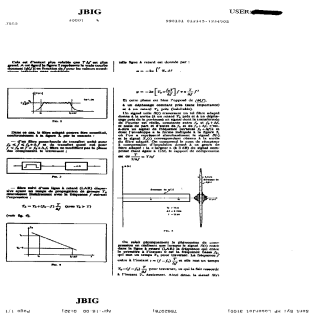


Figure 3(c) Recovered image with the Vertical-Line Mode

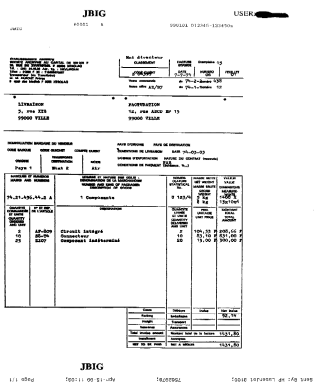


Figure 4(a) Original test image

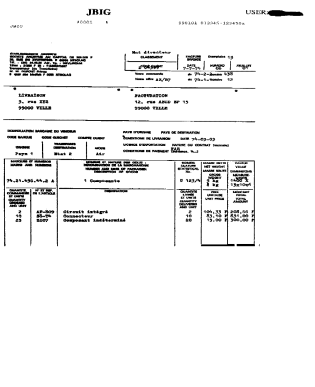


Figure 4(b) Decoded image without error recovery

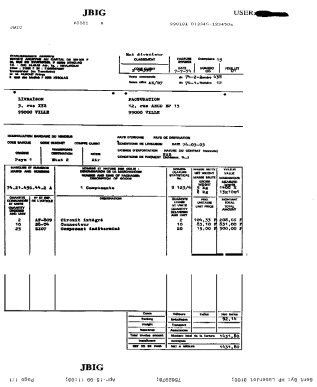


Figure 4(c) Recovered image with the Repetitive Mode