# ERROR RECOVERY IN FACSIMILE WITHOUT RETRANSMISSION

**Hyunju Kim**
**Department of Computer Science**
**The George Washington University**
**Washington, DC 20052 USA**
**hkim@gwu.edu**

**Abdou Youssef**
**Department of Computer Science**
**The George Washington University**
**Washington, DC 20052 USA**
**youssef@seas.gwu.edu**

**Abstract**

This paper develops and studies techniques for recovery from transmission errors that occur in facsimile images compressed with the Group 3, Extended 2 Dimensional MMR coding. When an error occurs in an MMR coded bitstream, the bitstream cannot be decoded after the error point, and the decoded image will have nothing after the error position. To prevent losing all information after an error, we develop techniques that utilize syntactical structure information of the coded bitstream to recover nearly all the data.

**Keywords**: Error Recovery, Resynchronization, 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.

Highly efficient compression methods remove various redundancies within the original data so as to produce a compact representation of the data. When transmission errors occur, the compact representation causes serious problems to a decoder: the decoder cannot reproduce the original data after the errors because of the propagation of the error effect.

Most receivers solve this problem by requesting retransmission. But retransmission adds more network traffic, requires more time, and incurs additional costs. 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 [1,2,4,6]. 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 [5,7]. In [7], 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 [5], a facsimile error recovery is presented without assuming any knowledge of error patterns, but the image in their research is not compressed with MMR, which is used in most current facsimile machines. Our research is the first to address error recovery in MMR, and we assume the following error model:

- There is no retransmission.
- 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.

This paper is organized as follows. Section 2 provides a brief overview of MMR coding. Error detection and resynchronization conditions are presented in section 3, and our error recovery approach is presented in section 4. Section 5 illustrates 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. 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 does not provide any resynchronization point. As a result, the decoder produces a corrupted image beyond the error position.

## 3 CONDITIONS OF ERROR DETECTION AND RESYNCHRONIZATION IN MMR

### 3.1 Error detection

To detect an error in the bitstream, we use the constraints implied in MMR shown in Figure 1. Error detection is done by checking for the following violations of the constraints:

- No codeword in the code table matches the received bit pattern.
- P mode occurs, but the changing element $b_2$ is off the right end of the scan line.
- H mode occurs and the run-length $a_0a_1$ is decoded, but the changing elements are either $|a_1 - b_1| \leq 3$ 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 $b_1 - a_0 - i \leq 0$.

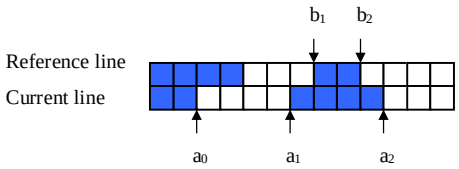
### 3.2 Conditions of resynchronization in MMR

Since MMR does not provide any synchronization codeword, we must find a portion of the bitstream that can be used somehow as a synchronization point. 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.

Derived from the syntactical information in MMR, two types of scan lines 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. An all-H line does not refer to the previous line since the H mode explicitly codes the white runs and black runs, thus it satisfies the second condition. More generally, a line consisting of one V mode and followed by one or more H modes can also be a synchronization point by guessing the length of the V mode that is likely to represent the length of the left margin in the line. As a result, our approach for error recovery consists of the following steps:




$a_0$: 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) | $a_1 \leq b_2$ & $\|a_1 - b_1\| > 3$ | 001 + M*($a_0a_1$) + M*($a_1a_2$) | $a_0 := a_2$ |
| Vertical Mode (V Mode) | $a_1 \leq b_2$ & $\|a_1 - b_1\| \leq 3$ | | |
| 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

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

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.

## 4 RESYNCHRONIZATION ALGORITHMS IN MMR CODED BITSTREAM

### 4.1 Algorithms to search for synchronization point

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 a white line is coded as *V(0)*. The first non-white line after the

white lines is coded by one or more H modes followed by one V mode, which is represented by a regular expression $H^+ V_D(i)$ where $D$ is either $L$ or $R$ and $i = 0, 1, 2,$ 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 $P^+ (V(0))^+ H^+ V_D(i)$. Figure 2 presents a pseudocode of resynchronization using a white line synchronization point. The validation checking is accomplished by checking if decoding the next 25 scan lines proceeds without any error; if so, the point is assumed to be valid.

For the second type of resynchronization (*All-H line resynchronization*), an all-H line should be found, which is represented by $H \ldots H$ (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.

Once this pattern is found in the bitstream, a Current line is constructed with the pixels that are decoded from the H modes, and decoding resumes from the first bit after the last H mode in the pattern with the newly constructed Reference line. Figure 3 shows a pseudocode of resynchronization using the all-H line synchronization.

1. Detect error using the error condition of subsection 3.1.
2. Search for the next white line synchronization expression, $P^+ (V(0))^+ H^+ V_D(i)$.
3. Check if the synchronization point is valid or not by checking for error-free decoding of the next 25 lines.
   3.1 If it is not valid, go to step 2.
4. Set a Reference line with 1728 white pixels.
5. Decoding the bitstream from the first H mode in the expression.

**Figure 2** White line resynchronization

1. Detect error using the error condition of subsection 3.1.
2. Search for the all-H synchronization expression, $V_D(i)\ V_D(j)\ H_1 \ldots H_n$.
3. For ($cnt$ = 1; $cnt$ < n+1; $cnt$++)
   3.1 Decode $H_{cnt}$, add the runs into the end of the Current line's right side, and fill the remaining left side with a white run.
   3.2 Check if the Current line is valid or not (by decoding the next 25 lines).
   3.2.1 If it is valid, exit the For-loop and go to step 4.
4. Check if step 3 found a valid synchronization point.
   4.1 If the point is not valid, go to step 2.
5. Decode the bitstream from the first bit after the last H mode, $H_n$, with the Current line constructed by step 3.

**Figure 3** All-H line resynchronization

### 4.2 An error recovery system

We developed an error recovery system, which incorporates the previous two resynchronization modules. After detecting an error, the system calls the white line resynchronization module to find a white line since most facsimile images are likely to have white lines. But the white line resynchronization module cannot detect all-H lines that are possibly located between the error detection point and the white line synchronization point. Thus, the system calls the all-H line resynchronization module to search for any all-H line located before the white line. If the system finds an all-H line during the searching step, the line will be the next synchronization point. If it fails, the system resumes decoding from the white line. Therefore, our error recovery system has the following steps:

1. Detect an error.
2. Call the white line resynchronization module and set *Rw* to the synchronization bit address from the module.
3. Set *bit address* to the error detection point.
4. While *bit address* is less than *Rw*
   4.1 Call the all-H line resynchronization module.
   4.1.1 If it found a valid synchronization point, set *Rh* to the point and exit the loop.
   4.1.2 If it fails to find a point, increase *bit address* by one.
5. Resume decoding from the synchronization point, either *Rw* or *Rh*.

## 5 ILLUSTRATIONS OF THE ERROR RECOVERY PERFORMANCE

The error recovery system has been implemented on a Pentium 4, 1.6 GHz using the C programming language. Testing has been done with 7 ITU facsimile test images and 8, more selective, sample images. Figure 4 shows an image that has been recovered from two errors with the white line resynchronization module. Figure 5 shows an image recovered with the all-H line resynchronization module.

## 6 CONCLUSION

In this paper, algorithms for error recovery in MMR coded bitstreams 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 as a resynchronization point. The testing shows that our system recovers well an image from error(s) as long as the image has the characteristics that the algorithms work for. We have also developed an interactive version of the error recovery system, described elsewhere, which solicits characteristic information on an image from the user and invokes different recovery approaches according to the information.

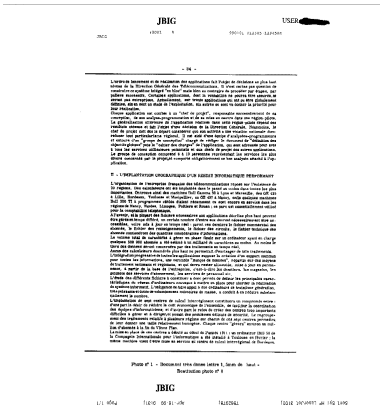


**Figure 4(a)** Original ITU test image

**Figure 4(b)** Decoded image without error recovery (no data after the error position)

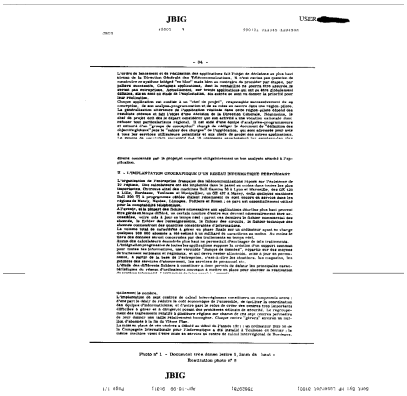


**Figure 4(c)** Recovered image with the white line resynchronization module

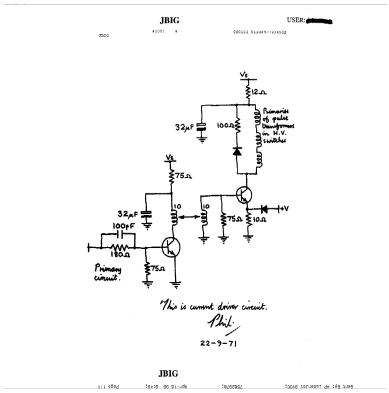


**Figure 5(a)** Original ITU test image

**Figure 5(b)** Decoded image without error recovery

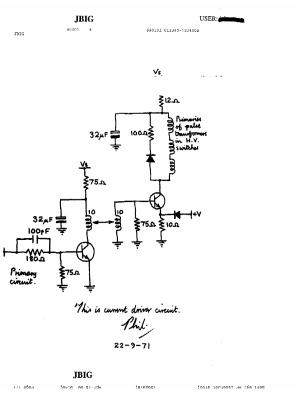


**Figure 5(c)** Recovered image with the all-H line resynchronization module

## 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] Kwok, W., and Sun, H., "Multi-Directional Interpolation for Spatial Error Concealment", *IEEE Trans. on Consumer Electronics*, vol. 39, no. 3, Aug. 1993, pp. 455-460.

[5] 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.

[6] 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.

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