Bit error recovery in Internet facsimile without retransmission

Hyunju Kim and Abdou Youssef

Department of Computer Science
The George Washington University
Washington, DC 20052
Email: {hkim, youssef}@seas.gwu.edu

ABSTRACT

This paper proposes a bit error recovery method for Internet 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 correctly decoded after the error point. To prevent losing valid information after an error, we developed an error recovery system that detects bit errors and applies bit-inversion algorithms to correct the errors. In case the bit-inversion cannot correct the error, the system applies new algorithms that utilize syntactical structure information of the coded bitstream. Testing results show that around 95% of bit errors are corrected with our bit-inversion algorithms. The error recovery system also recovers nearly all the data for the remaining 5% of the images.

Keywords: Bit error recovery, MMR coding, Resynchronization, 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 retransmission. But this 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.

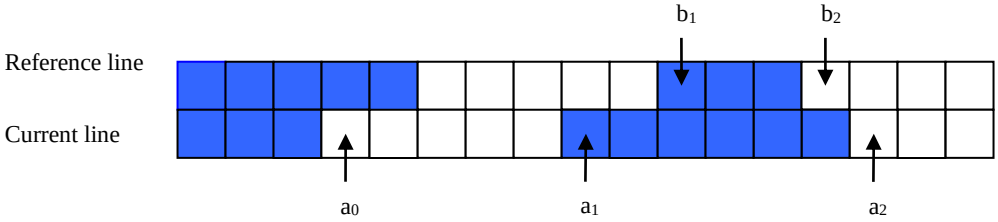
Most research on error recovery has focused on error concealment methods for lossy block-based DCT compression as in JPEG or MPEG.[1,2,8] These methods assume that a bitstream has been fully decoded and the position of the erroneous block is known. For error recovery in losslessly compressed bitstream, which is relevant to this paper, relatively little research has been done.[7,9] In Youssef and Ratner's research, 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 Shyu and Leou's research, an error recovery method is presented for facsimile image 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 coded bitstreams. In two other papers,[5,6] we developed symbol error recovery in MMR fax over modem. In this paper, we address single-bit error recovery in MMR. We define a single-bit error as an isolated wrongly flipped bit, which usually occurs in a data unit of the IP datagram. Thus, our research has 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. Our error recovery approach is presented in section 3, and the performance of the approach is illustrated in section 4. 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 in 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. Figure 1 shows the elements of MMR coding.



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 to a_0.
b_2 : The next changing element to the right of b_1 on the reference line.

The position of a_0 changes according to the rules in (b) and the positions 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 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 1st value of a_0 is taken to be an imaginary blank pixel left of the line.

Changing picture elements
(a)

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

MMR coding and update of a_0
(b)

Figure 1. MMR coding

When an error occurs in an MMR coded bitstream, the error propagates through the bitstream because MMR coding scheme does not provide any synchronization points. As a result, the decoder produces a corrupted image beyond the error position.

3. THE OVERALL APPROACH TO BIT ERROR RECOVERY

For the MMR coded bitstream, our approach to bit error recovery is to (1) detect the error, (2) determine the region of error, and then (3) apply bit-inversion and re-decoding to bits within the region. If the third step fails to recover from the error, our system tries to find a resynchronization point in the bitstream to resume decoding.

3.1 Error detection

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 $b_1 - a_0 - i \leq 0$.
- The Horizontal mode occurs and the run-length a_0a_1 is decoded, but the changing elements are either $| a_1 - b_1 | \leq 3$ or $b_2 < a_1$.
- 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.

These constraints are also used to verify any tentative bit-inversions in our error recovery system. After detecting single-bit error(s), our system tries to algorithmically correct the bit error(s) with bit-inversions. If any of the constraints is violated during the following re-decoding steps, the bit-inversions are assumed to be invalid.

3.2 Error region determination

An error detection point, denoted by ED_i, is usually located away from the error position, denoted by E_i. Then, the region of error becomes from E_i to ED_i ($E_i < ED_i$) in the bitstream. However, the exact position of the error, E_i, is not known when the error is detected. Thus, we need to find an earlier start point, denoted by ES_i, of the error region ($ES_i < E_i < ED_i$). The region of error ranges from ES_i to ED_i in the bitstream.

To determine ES_i, we conducted an extensive test and measured average *detection latency*, which is the length in bits from E_i to ED_i. The result shows that the average detection latency, denoted by DL_μ, is about 80 bits, and the standard deviation of the detection latency, denoted by DL_σ, is about 245 bits. Thus, the start of the error region, ES_i, has been practically set to $ED_i - 815$ ($DL_\mu + 3DL_\sigma = 815$ bits). If the correction of the error fails within the error region, a resynchronization point S_i ($ES_i < E_i < ED_i < S_i$) is used to resume decoding.

3.3 Resynchronization in MMR

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

Derived from the syntactical information in MMR, two types of scan lines can be used as a synchronization point: a *white line* (blank line) and a line consisting of the Horizontal modes only (*all-H line*). A white line, which satisfies the first condition, can be used as a Reference line for the next line since it consists of only white pixels. Then, decoding can be resumed from the following line. An all-H line does not refer to the previous line since the Horizontal mode explicitly codes the run-lengths of white runs and black runs, thus it satisfies the second condition.

For each type of the scan line to be used as a resynchronization point, we determined a regular expression (pattern) to search for in the bitstream to locate such a resynchronization point, as explained next.

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.

Figure 2 and 3 present pseudocode of the white line resynchronization and all-H line resynchronization. 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. These resynchronizations are used when the error recovery system fails to correct bit error(s) with the bit-inversion. The system calls the resynchronization modules to find a synchronization point so that it keeps decoding from the point.

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 with the Current line constructed by step 3.

Figure 3. All-H line resynchronization

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.

3.4 Bit inversion for multiple single-bit errors

When we determine an error region after detecting bit error(s), we do not know how many single-bit errors have occurred in the region. First, our system applies a simple bit-inversion and re-decoding to every bit in the region. It flips a bit in the region and checks for the violations of the constraints, which are presented in subsection 3.1, during the re-decoding step to see whether the bit-inversion is valid or not. If there are any violations, it reinstates the bit and moves to the next bit.

In case there is only one single-bit error within the region, the simple bit-inversion and re-decoding would correct the error. On the other hand, if there are multiple single-bit errors, the simple bit-inversion and re-decoding cannot correct the errors. A brute-force way to correct these multiple single-bit errors would try all 2^{815} bit-combinations (an error region contains 815 bits as defined in subsection 3.2.), which is infeasible. Fortunately, we found some relationships between bit-errors and their detection points, which could be used to reduce the number of trials to a realistic range.

In another paper,[4] the authors conducted an extensive test, which introduced randomly generated pairs of bit errors into the ITU test images and measured their detection points. 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 detection constraints. This means that when we detect single-bit error(s), there would be 83% chance to correct the error by using the sequential simple bit-inversion and re-decoding. For the remaining 17% of the test cases, the simple bit-inversion does not work because the error regions contain more than one single-bit errors. In this paper, we develop a recovery algorithm to take care of the remaining 17% of cases.

According to the test results, levels of error impacts affect the detection points; 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. Thus, in multiple single-bit errors, detection points are likely to be affected by the last bit error within the error region. It also verified that more bit errors within an error region produce shorter detection latencies in most cases. We developed an error recovery algorithm for multiple single-bit errors by utilizing these observations. Figure 4 shows pseudocode for the algorithm assuming that there are two bit errors in the error region. The algorithm can be modified for more than two bit errors by extending step 3.

1. Detect a bit error, set ED_{pair} to its error detection point (ED), and define its error region.
2. For each bit in the region
   2.1 Record its ED.
   2.2 If the ED is equal to or greater than ED_{pair}
      2.2.1 Create a candidate error bit, C_i (i, ED_i) where i is the bit address and ED_i is the ED from the bit, i.
3. For each C_i where its ED_i is equal to ED_{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 re-decoding it.
         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. Bit-inversion algorithm for multiple single-bit errors

As a result, our bit error recovery system has the several procedures, which are presented in Figure 5. After trying the simple bit-inversion and the bit-inversion for multiple single-bit errors, if the error still persists, the system assumes that the error occurs outside of the presumed error region. In this case, the resynchronization modules will be used to find synchronization points.

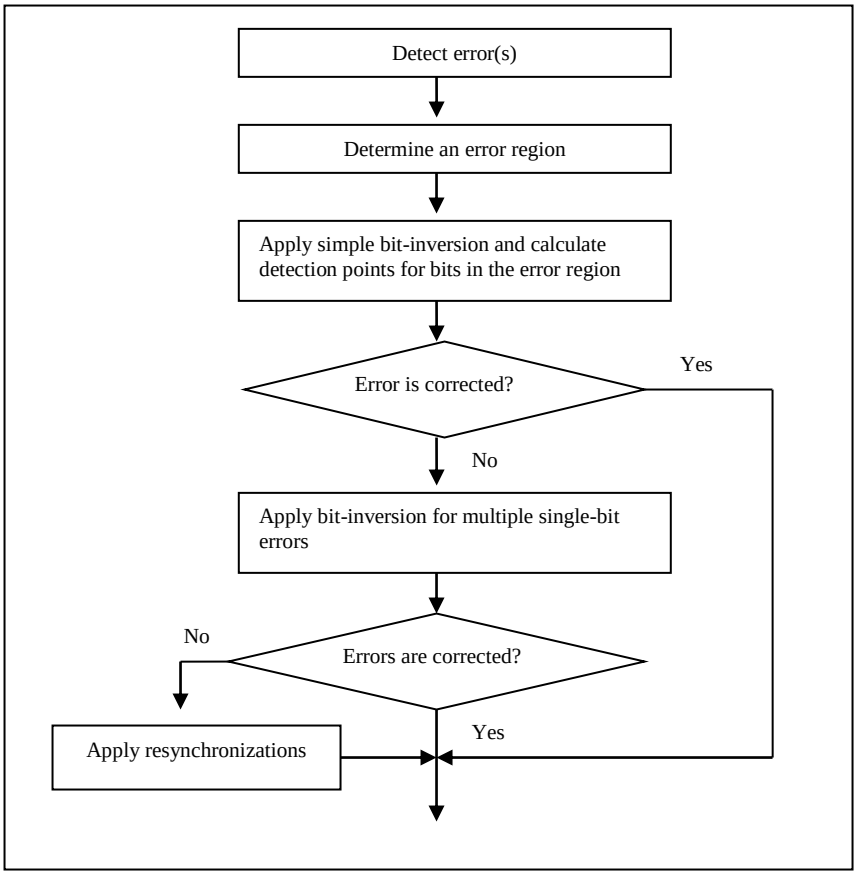


Figure 5. Bit error recovery procedures

4. 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 95% of single-bit errors are corrected with our bit error recovery system. 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}. On the other hand, assuming no error recovery for the same test cases, the RE increases to the range from 0.0014 to 0.2, and averages at 0.04.

For the remaining 5% of the test cases, which failed to recover from errors with the bit-inversion algorithms, the error recovery system invokes the white line resynchronization and the all-H line resynchronization modules. The system takes an earlier synchronization point from the two synchronization points provided by the resynchronization modules and resumes decoding from the point. Figure 6 and 7 present some recovered images with the resynchronization modules, which recover nearly all the data.

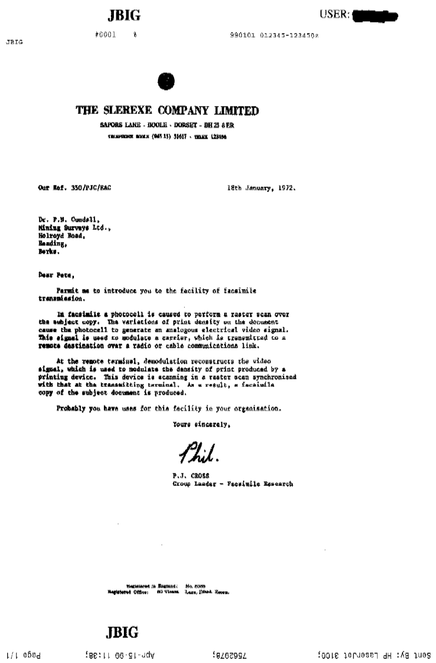
5. CONCLUSION

In this paper, a bit error recovery approach for Internet facsimile images is proposed. We developed an error recovery system, which detects single-bit errors, determines an error region, and applies bit-inversion and re-decoding to correct single-bit errors within the error region. After finding the error region, the system first applies a simple bit-inversion and re-decoding to correct one single-bit error. In case this step cannot correct the error, our system applies the proposed bit-inversion algorithm for multiple single-bit errors, which utilizes detection points that have been identified during the simple bit-inversion and re-decoding.

Our test results show that around 83% of the bit errors were corrected with the simple bit-inversion and re-decoding; around 12% of the errors were corrected with the bit-inversion for multiple single-bit errors; the remaining 5% of the test cases failed to recover from errors with the bit-inversions because the errors occurred outside of the error region. To recover the images from these errors, our system applies the resynchronization modules to skip erroneous regions and resume decoding.

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. ITU-T Recommendation T.6, Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus, 1988.
4. Kim, H., and Youssef, A., "Bit Error Recovery in MMR Coded Bitstreams Using Error Detection Points", accepted to appear in CISST2003, Jun. 23-26, Las Vegas.
5. Kim, H., and Youssef, A., "Interactive Error Recovery in Facsimile without Retransmission", accepted to appear in ITCC2003, Apr. 28-30, Las Vegas.
6. Kim, H., and Youssef, A., "Error Recovery in Facsimile without Retransmission", accepted to appear in CATA2003, Mar. 26-28, Honolulu.
7. 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.
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.



JBIG USER: [redacted]

#0001 8 990101 012345-1234567

JBIG

THE SLEREXE COMPANY LIMITED

SAPORS LANE - BOOLE - DORSET - BH 25 8 ER

TELEPHONE BOOLE (945 13) 51617 - TELEX 123456

Our Ref. 350/PJC/EAC 18th January, 1972.

Dr. P.N. Cundall,
Mining Surveys Ltd.,
Holroyd Road,
Reading,
Berks.

Dear Pete,

Permit me to introduce you to the facility of facsimile transmission.

In facsimile a photocell is caused to perform a raster scan over the subject copy. The variations of print density on the document cause the photocell to generate an analogous electrical video signal. This signal is used to modulate a carrier, which is transmitted to a remote destination over a radio or cable communications link.

At the remote terminal, demodulation reconstructs the video signal, which is used to modulate the density of print produced by a printing device. This device is scanning in a raster scan synchronised with that at the transmitting terminal. As a result, a facsimile copy of the subject document is produced.

Probably you have uses for this facility in your organisation.

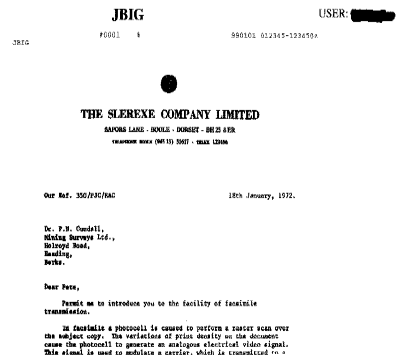
Yours sincerely,

Phil.

P.J. CROSS
Group Leader - Facsimile Research

JBIG

Figure 6(a) Original test image



JBIG USER: [redacted]

#0001 8 990101 012345-1234567

JBIG

THE SLEREXE COMPANY LIMITED

SAPORS LANE - BOOLE - DORSET - BH 25 8 ER

TELEPHONE BOOLE (945 13) 51617 - TELEX 123456

Our Ref. 350/PJC/EAC 18th January, 1972.

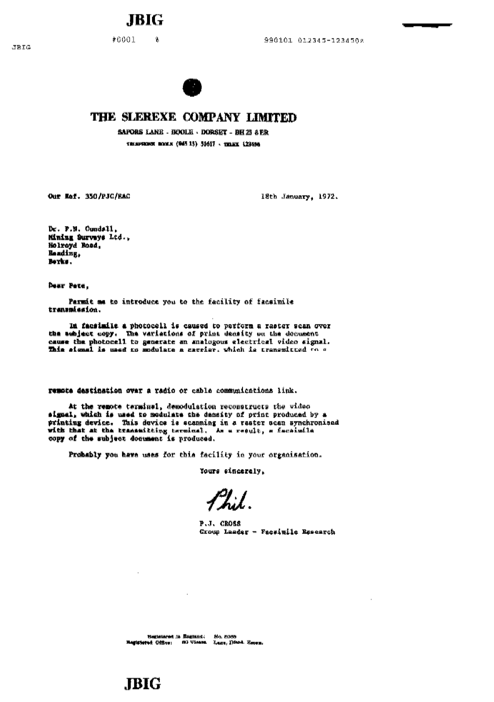
Dr. P.N. Cundall,
Mining Surveys Ltd.,
Holroyd Road,
Reading,
Berks.

Dear Pete,

Permit me to introduce you to the facility of facsimile transmission.

In facsimile a photocell is caused to perform a raster scan over the subject copy. The variations of print density on the document cause the photocell to generate an analogous electrical video signal. This signal is used to modulate a carrier, which is transmitted to a

Figure 6(b) Decoded image without error recovery (no data after the error point)



JBIG [redacted]

#0001 8 990101 012345-1234567

JBIG

THE SLEREXE COMPANY LIMITED

SAPORS LANE - BOOLE - DORSET - BH 25 8 ER

TELEPHONE BOOLE (945 13) 51617 - TELEX 123456

Our Ref. 350/PJC/EAC 18th January, 1972.

Dr. P.N. Cundall,
Mining Surveys Ltd.,
Holroyd Road,
Reading,
Berks.

Dear Pete,

Permit me to introduce you to the facility of facsimile transmission.

In facsimile a photocell is caused to perform a raster scan over the subject copy. The variations of print density on the document cause the photocell to generate an analogous electrical video signal. This signal is used to modulate a carrier, which is transmitted to a

remote destination over a radio or cable communications link.

At the remote terminal, demodulation reconstructs the video signal, which is used to modulate the density of print produced by a printing device. This device is scanning in a raster scan synchronised with that at the transmitting terminal. As a result, a facsimile copy of the subject document is produced.

Probably you have uses for this facility in your organisation.

Yours sincerely,

Phil.

P.J. CROSS
Group Leader - Facsimile Research

JBIG

Figure 6(c) Recovered image

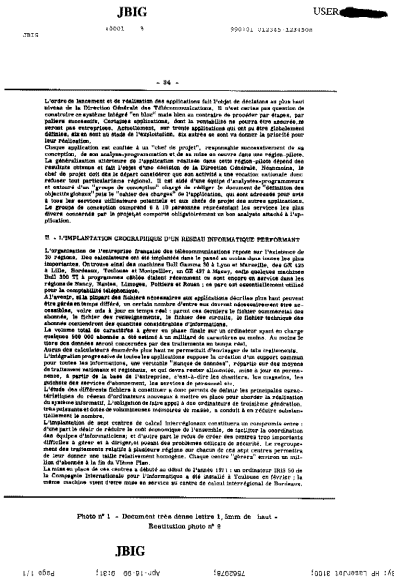


Figure 7(a) Original ITU test image

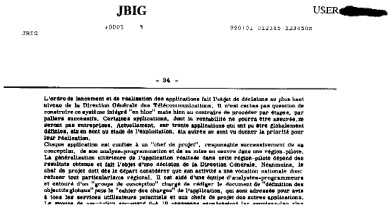


Figure 7(b) Decoded image without error recovery (no data after the error position)

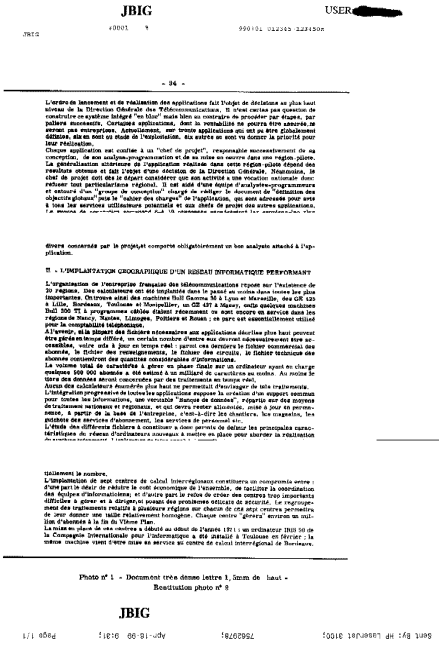


Figure 7(c) Recovered image from two errors