ELSEVIER

Signal Processing: *Image Communication* 13 (1998) 45–64

SIGNAL PROCESSING: IMAGE COMMUNICATION

Transmission error detection and concealment in JPEG images

Mourad Abdat, Ziad Al Kachouh, Maurice G. Bellanger*

Laboratoire d'Electronique, CNAM, 292 rue Saint-Martin, 75141 Paris Cedex 03, France

Received 20 September 1996

Abstract

The robustness to transmission errors of JPEG coded images is investigated and techniques are proposed to reduce their effects. After an analysis of the JPEG transfer format, three main classes of transfer format constituents are distinguished, and a JPEG compatible approach is proposed to stop error propagation in the entropy coded data, with encoder and decoder reset after fixed coding interval lengths.

With the use of restart intervals, the propagation of errors is stopped but no correction has taken place. Therefore, a concealment procedure is defined and investigated. It consists of two steps. First, error detection must be performed and three different techniques are assessed and compared. Then, block error concealment is achieved.

Simulation results are reported. Depending on the entropy coding and on the neighborhood templates used for detection and concealment, prediction based or interpolation based, improvements between 3 and 10 dB in peak-to-peak signal-to-noise ratios (PSNR) are provided by the robust decoder with respect to conventional JPEG decoders, for bit error rates around 10^{-4}.

Keywords: Image compression; JPEG; Huffman coding; Arithmetic coding; Transmission errors; Error detection; Error concealment

1. Introduction

An international standard for continuous-tone still image compression has been approved by International Standard Organization (ISO) under the denomination of International Standard-10918 (IS-10918) or, more commonly Joint Photographic image Experts Group (JPEG) [6, 24]. It has been designed for the storage and retrieval of continuous-tone still images on numerical devices. Since this kind of environment is virtually error-free, the robustness aspects have been, to a large extent, overlooked. With the generalization of digital techniques in all parts of the transmission networks, compressed images are likely to be transmitted on error-prone media, like radiocommunication links. In those conditions, it becomes important to have available detailed results concerning the robustness of the compressed images to transmission errors, as well as schemes to improve it.

* Corresponding author. Tel.: 33 1 40 27 20 82; fax: 33 1 40 27 27 79; e-mail: bellang@cnam.fr.

0923-5965/98/$19.00
PII S0923-5965(97)00046-5

Some efforts have already been devoted to transmission error detection and error concealment in still images without modification of the encoder. In [15], a transmission error detection method and an error concealment method are proposed for a DCT-based coding scheme with bit allocation. The error detection is based on the comparison of the current block boundaries with those of its four horizontal and vertical neighboring blocks and the computations are done in the frequency domain. This technique can detect accurately the damaged DCT coefficient in the block, but the detection of several errors in one block is virtually impossible. Furthermore, the method cannot be adapted to a JPEG-like coding scheme because of error propagation due to the entropy coding and the DC DPCM coding. The associated concealment method operates in the frequency domain, and it does not use the neighborhood blocks in the diagonal directions, thus, edges in those directions can be poorly restored. In [2, 22], an error resilient entropy code is proposed. It relocates the bits of the variable length code in order to get a fixed length code which is less sensitive to error propagation. It has the disadvantage of introducing modifications to the entropy coder and slightly increasing the bit-rate.

As concerns error concealment specifically, two approaches can be followed. The spatial domain approach restores the block boundaries properly but it can be weak on block edges and patterns. In fact, it is better suited to direct pixel coding schemes like DPCM than to block-based coding schemes. Alternatively, the frequency-domain approach restores the damaged blocks by prediction or interpolation. For example, in [18] frequency-domain concealment techniques are derived from the equalization of spatial block boundaries. In [3], the focus is on the restoration of damaged block patterns. The DC coefficient is concealed through interpolation from the DCs of neighboring blocks, and the coefficients AC(0, 1) and AC(1, 0) are interpolated according to the pattern type of the top–down and left–right blocks. The simplicity is appealing, but the performance can be rather limited in the case of high bit error rates (BER around 10^{-4}) and it is not suited to situations where several consecutive blocks are damaged. A fuzzy logic approach is presented in [13], and the results shown are obtained for isolated errors or limited error propagation (two consecutive damaged blocks), which is not the case of JPEG-coded images.

As concerns moving picture transmission, the efforts are focussed on Moving Picture Experts Group (MPEG) image transmission over ATM networks, and most of the proposed schemes deal with encoder modifications (prioritization, error control codes, etc.) [4, 7, 8, 17, 21].

More interestingly, the concept of extended synchronization codewords is presented in [10, 11], specifically for JPEG. A synchronization codeword is transmitted at the end of each block row, which is similar in effect to the restart intervals defined below, but less flexible and limited to isolated block split or merge errors.

The present paper, dedicated to JPEG, offers a complete solution, from header protection to error detection and concealment methods. It is organized as follows. In Section 2, an overview of the JPEG standard and the transfer format organization is given. Section 3 presents an analysis of the transfer format constituents and their classification according to their sensitivity to transmission errors, whereas Section 4 focusses on error propagation in the entropy-coded data and its limitation by the use of restart intervals. The impact on compression ratios for Huffman and arithmetic coding are presented in the same section. Subsequent error detection and concealment techniques are described in Sections 5 and 6. The whole decoder block architecture is given at the end of Section 6. The objective and subjective results obtained in the case of the two entropy-coding schemes with different neighborhoods are presented in Section 7. The conclusion offers some global assessment and points to track for further works.

2. JPEG and transfer format overview

The JPEG standard specifies the encoder, the decoder and the transfer format for continuous-tone still image compression. In order to have a large application area, the standard defines several coding schemes. In this paper, we are interested in sequential DCT-based processes, namely, the baseline process with Huffman coding and the extended process with an arithmetic entropy coder.

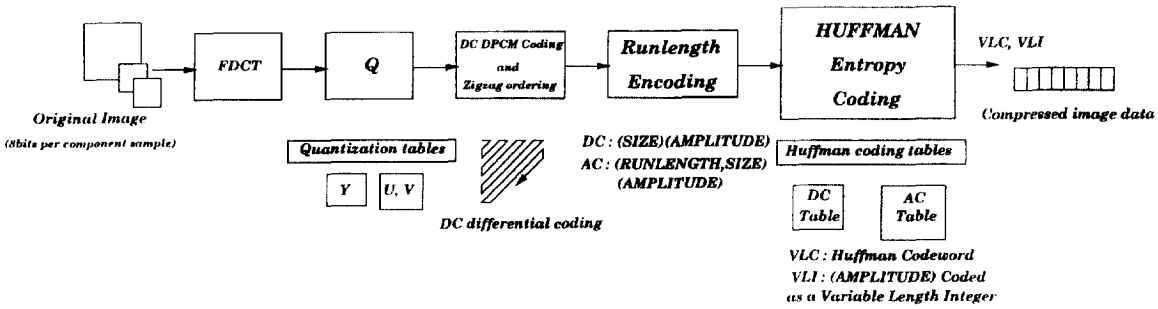


Fig. 1. Baseline JPEG coding process.

Fig. 1 shows the baseline process. The original image is scanned sequentially in 8×8 pixel blocks which are discrete cosine transform (DCT) transformed. The DCT coefficients are then quantized using provided quantization tables, the DC quantized coefficient is differentially coded and the AC coefficients are rearranged in a zigzag order. The run-length encoding encodes each non-null AC coefficient into two symbols (RUN, SIZE) and (AMPLITUDE). The former contains two four-bit parts: (RUN) encodes the number of consecutive zeros preceding the current coefficient in the block, and (SIZE) encodes the number of bits to code (AMPLITUDE). The second symbol (AMPLITUDE) represents the coefficient value coded in SIZE bits. Since RUN is coded in four bits, only zero runs which are less than 16 can be coded by one RUN symbol. In order to be able to code longer runs of zeros, two special values are used for the first symbol. The first special value is (15,0) which indicates the extension of the zero run, thus it encodes 16 consecutive zeros and it shall be followed by another (RUN, SIZE) symbol. The second special value is (0,0) which encodes an end of block (EOB): all the remaining coefficients in the block are zeros. After the run-length encoding, the first symbol (RUN, SIZE) is encoded by a Huffman coder as a variable length code (VLC). The second symbol (AMPLITUDE) is transmitted as a variable length integer (VLI) on SIZE bits, thus, the output of the entropy coder is an interleaved bit stream of VLCs and VLIs. The DC difference is coded into two symbols (SIZE) and (AMPLITUDE) without a RUN part in the first symbol.

The JPEG-extended coding process with arithmetic coding differs from the baseline process mainly by the entropy coder. There is no explicit run-length encoder, the DC difference and reordered AC coefficients are directly coded by an adaptive binary arithmetic coder. Arithmetic coding is a lossless source coding based on two recursive operations used for the subdivision of the unit interval. An adaptive statistical model is used to estimate the probability of the next decision (symbol) [1, 6, 19]. The encoder produces successive binary decisions relative to the quantized DCT coefficient values, zero runs and end-of-block decisions. For example, the first decision in a block is relative to the DC difference. If the difference is zero, a symbol 1 is generated, otherwise a 0 is generated by the binary arithmetic coder. For each correlated set of decisions, a statistical area is used to estimate the probabilities needed by the arithmetic coder. The coding contexts and their update are based on the preceding decisions in previous blocks or in the same block. The output of the binary arithmetic coder is a succession of 0's and 1's which are shifted out from the interval base register (C) used for the interval subdivision operations. More details about the binary arithmetic coder can be found in [6, 14, 19, 20].

For all the JPEG coding processes a compressed data format produced by the encoder and read by the decoder is specified. The compressed data is an ordered collection of markers, parameters and entropy-coded segments. The markers are fixed values coded on 16 bits. Each marker begins with an $0 \times FF$ byte followed by another byte which makes the marker value. These markers identify the different structural parts of the transfer format, and in most of the cases they are headers of a group of parameters. The parameters have

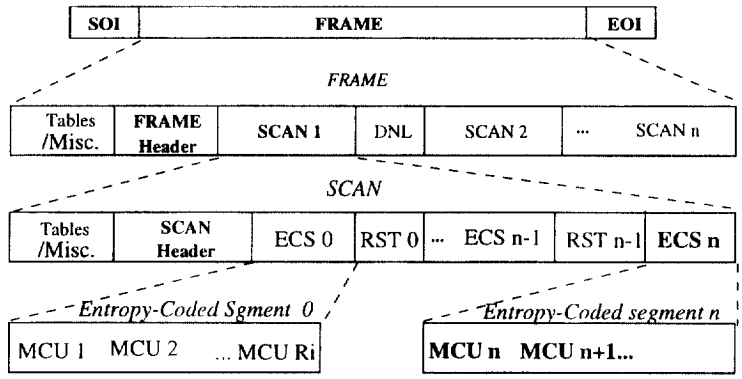


Fig. 2. High-level syntax of the JPEG transfer format.

specific values according to the source image characteristics, the coding process and other application-specific information. They code critical information for image decoding and reconstruction at the decoder side. The entropy-coded segments represent the output of the entropy encoder.

The high-level syntax and organization of the JPEG transfer format for non-hierarchical modes is presented in Fig. 2 [6]. The compressed data shall begin with a start of image (SOI) marker, contain a frame and end with an end of image (EOI) marker. A frame shall begin with a frame header and shall contain one or more scans. The header may be preceded by one or more table specifications or miscellaneous marker segments. The constituents of the frame header contain information relative to the coding process, image dimensions, data precision, number of components in the image and the component parameters. The third level of Fig. 2 specifies that a scan shall begin with a header followed by one or more entropy-coded data segments. The scan header may be preceded by one or more table specifications or miscellaneous marker segments. The header shall contain markers and parameters relative to the number of components in the scan, the parameters relative to each component contained in the scan and other parameters used in some coding processes. The ReSTart number i (RSTi) markers define the reset points for the entropy coder. At these points the entropy encoder shall be reset and the entropy decoder shall do the same to continue the decoding process. The entropy-coded segments are a succession of minimum coded units (MCU). The number of coded blocks contained within these units depends on the scan scheme and the coding process [6].

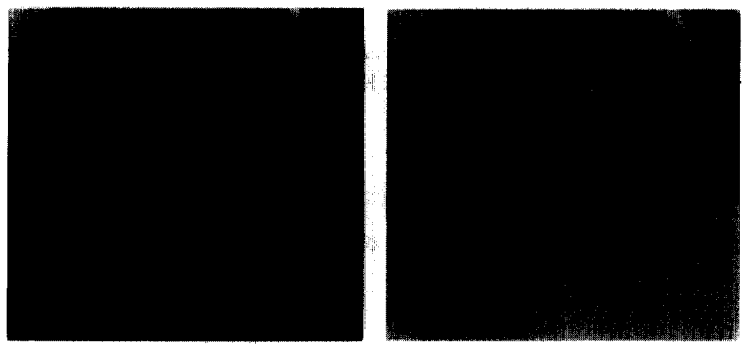
The transfer format, with its organization, is an important part of the JPEG standard, since it represents the entity which is transmitted from the encoder to the decoder, and it is also the entity which is interchanged between applications. In order to take suitable protection measures against transmission errors, the sensitivity of the constituents of the transfer format to transmission errors must be analyzed.

3. Transfer format robustness to transmission errors

The approach consists of introducing errors in each of the constituents, decoding the erroneous compressed image and comparing with the error-free decoded image. Let us begin with the frame header marker and segments. They are essential for the decoding process and, in most cases, the decoder cannot decode the image when one of the header parameters contains errors. Concerning the scan header markers and parameters, when the image is coded in one scan, the degradations affect the whole image. When it is coded in several scans, the degradations concern the components involved in the erroneous scan only, thus, depending on the importance of those components, different results may be obtained for the whole image. In most cases, errors

(a) (b)

Fig. 3. Lena decoded image with transmission errors on table definitions: (a) decoded image with errors on DC quantization; (b) decoded image with errors in Huffman tables.



(a) (b)

Fig. 4. Decoded images with one bit error in the entropy data: (a) baseline process; (b) extended process with arithmetic coding.

in scan headers give catastrophic results, since the information contained in those headers is critical. The same applies to transmission errors in table definitions. Quantization and Huffman tables can be lost. For the quantization tables, if the errors occur on the DC or low AC quantization steps, the decoded image quality is very poor, while a slight degradation is observed when the errors affect high-frequency coefficient quantization. Now, Huffman tables are lost in most cases if they contain transmission errors, and the decoding process is interrupted. An illustrative example is given in Fig. 3.

Concerning the entropy-coded data segments, with the Huffman coder, the entropy data form an interleaved bit stream of VLCs and VLIs. Transmission errors desynchronize the decoder, error propagation takes place and the decoded image quality depends on the location of the errors in the data. The reconstructed image quality is poor when the whole image is coded into a single entropy segment.

In the case of arithmetic coding, even with the presence of only one bit error in the data, the results are catastrophic. The error propagates till the end of the image due to the recursive operations in the binary arithmetic coder and the adaptivity of the statistical model. An illustration is given in Fig. 4.

Concerning the optional and application-specific transfer format parameters, since the image does not depend on them, their alteration has no impact on the reconstructed image.

In moving picture coding for transmission applications, the coding is generally layered, and the critical constituents of the bit stream are highly protected [4, 7, 8, 17, 21]. In a similar manner, we propose a classification of the transfer format in three classes, according to their sensitivity to transmission errors. The first class contains the start of image and end of image markers, the frame and scan headers, the quantization tables, the entropy coding tables (Huffman tables or arithmetic decision tables). The size of this class is relatively small, about a few hundred bytes, and it can be protected with the MPEG Reed–Solomon code (204, 188) which can correct 8 byte errors in 204 bytes. The overhead introduced that way does not affect significantly the compression ratio, a loss of under 0.3% for the whole compressed image with JPEG quantization tables.

The second class corresponds to the entropy-coded data which account for approximately 98% of the coder bit-rate. An error-correcting code alone is not sufficient for its protection for two reasons, first this may require a large overhead and second, if only one bit error escapes the error-correcting code, the reconstructed image can be catastrophic as shown in Fig. 4. An error-correcting code must be combined with other solutions which limit the error propagation. In our scheme, we have used no error-correcting code for this class, but we have adopted the JPEG restart interval to limit error propagation and we have used error detecting and concealment techniques to improve the reconstructed image quality.

The last class contains the optional and application-specific segments. The image reconstruction does not depend on these segments, and their protection is not needed.

As a specific example, an image with three components coded by the base-line process into one interleaved scan with the two JPEG quantization tables and four Huffman tables has given the following bit distribution for each of the proposed classes:

- Class 1: 3992 bits (1.2%),
- Class 2: 318912 bits (98.5%),
- Class 3: 992 bits (0.3%).

It is worth emphasizing that the size of the second class is closely related to the compression ratio, thus, mainly to the quantization tables.

4. Error propagation

Transmission errors in entropy-coded data propagate over several DCT coefficients and blocks. This propagation is mainly due to the following reasons:

- Entropy coding: In the baseline process, the error propagates because of the variable-length Huffman codes and the variable-length integers which are interleaved in the bit stream. In the extended process with arithmetic coding, the error propagation is due to the recursivity of the interval subdivisions in the binary arithmetic coder and to the adaptivity of the statistical model.
- Run-length encoding: In the baseline process, decoding erroneous runs of zeros modifies the block architecture and may result in block split and merge errors.
- Differential DC coding: An error on a decoded DC difference is propagated over all the remaining DC coefficients.
- Inverse DCT: An error on one coefficient in the DCT domain translates in a spatial erroneous block.

In the baseline process, the variable length codes propagate errors because their length is known only when they are completely decoded. This error propagation is magnified by the VLCs and VLIs interleaving. A bit error in a Huffman code word can transform the code word into a different code word or even several code words with different lengths. This leads to the desynchronization of the decoder, and, in order to limit the error propagation, autosynchronized Huffman codes must be used [10]. According to the definition of self-synchronized Huffman codes, the JPEG Huffman tables given in the standard contain several synchronization code words [6]. For example, the JPEG DC difference Huffman tables contain five synchronizing code words

over a total of 12 code words in the table. Even when synchronization is recovered, this is not sufficient to get a well-reconstructed image because the decoder must know where to put the decoded data.

An error on a (RUN, SIZE) symbol modifies the zero run when RUN is erroneous and desynchronizes the decoder when SIZE is erroneous. Errors at the block level may result from additional decoded EOBs or missing EOBs. If additional EOBs are decoded, some extra blocks are decoded, thus some blocks split into pairs of blocks. On the contrary, when there are missing EOBs, pairs of blocks are decoded as one block, it is a merge error. In the two cases many blocks are shifted in the image.

The DPCM coding of the DC coefficients propagates errors till the next synchronization marker. The use of PCM instead of DCPCM results in a loss of compression ratio of about 5–10%. Concerning the inverse discrete cosine transform, an error in a single coefficient produces errors on all the pixels of the reconstructed block. With the arithmetic coding, the error propagation is catastrophic, all the blocks following the error are lost.

In order to limit error propagation, a generic solution is to use independent synchronization points at regular interval lengths fixed by the user. At the beginning of each interval, the DPCM coder and the entropy coder are reset, and the same is done at the decoder side upon the reception of a synchronization marker. This confines the errors in intervals of specified lengths and the different intervals are independant, which can also be exploited for parallel decoding of the image.

In the case of the JPEG standard, the synchronization intervals are named restart intervals. They are also extended to the case of the MPEG standard under the name of slice [7]. The restart intervals are delimited by 16-bit markers. Eight markers (0 x FFD0–0 x FFD7) are periodically used to define the synchronization points in the entropy data, and a dedicated header segment define restart interval (DRI) defines the number of minimum coded units (MCU) per interval.

The scheme of restart intervals is more attractive than those proposed in [11,22], which split the image data into fixed segments, for several reasons. First, it is a JPEG compatible solution, since any JPEG decoder recognizes the synchronization points. Second, the length of the intervals is fixed by the user, and can be chosen according to the robustness/compression trade-off desired. Moreover, the concept of restart intervals is applicable to any coding scheme with any entropy coder.

Concerning the lengths of the intervals, they are expressed in MCUs. An MCU can contain different numbers of blocks depending on the number of image components in the scan and the subsampling factors of each component. The minimum length of an MCU is a one-coded block which corresponds to the case where

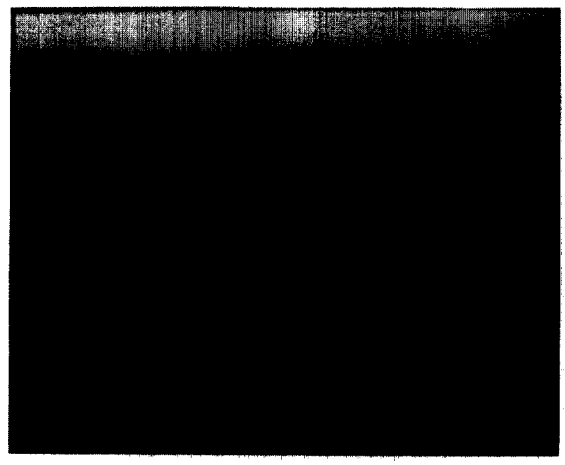
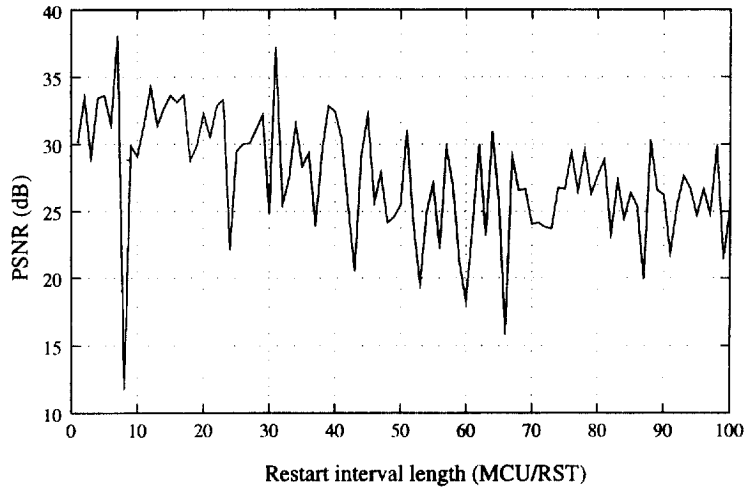
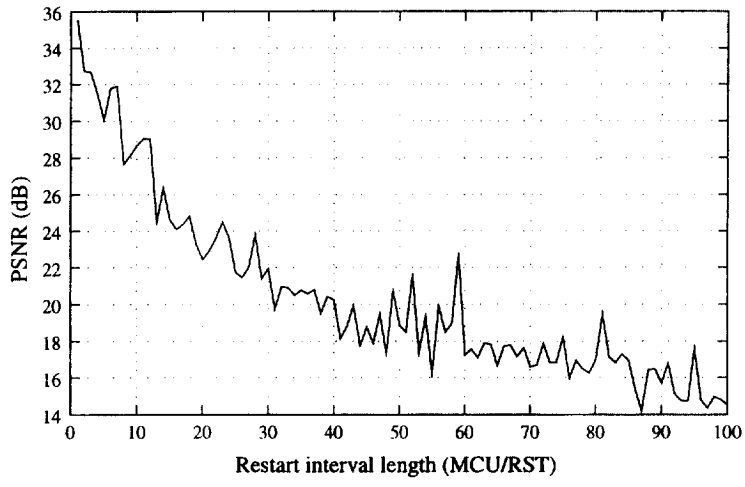


Fig. 5. Error-free decoded Gold image.



(a)



(b)

Fig. 6. PSNR obtained under BER of 10^{-4} for the image Gold: (a) baseline process; (b) extended process with arithmetic coding.

there is only one image component in the scan. For example, in the case of an interleaved scan with three components, a luminance component with horizontal and vertical sampling factors $H_l = 2, V_l = 2$, and two chrominance components with sampling factors $H_c = 1, V_c = 1$, an MCU contains six coded blocks with four luminance coded-blocks and one coded-block for each chrominance component. In the simulations, we have used gray-scale images with one luminance component coded into one scan, thus an MCU corresponds to a one-coded block. Fig. 5 is the error-free decoded Gold image.

For a bit error rate around 10^{-4}, the JPEG-coded image without restart intervals suffers important degradations. The peak-to-peak signal-to-noise ratio (PSNR) is less than 10 dB, the best values being obtained for Huffman coding. With restart intervals, the PSNR of the decoded images is significantly improved. The greatest improvement is obtained for Huffman coding, as illustrated in Fig. 6, where peak-to-peak signal-to-noise

Fig. 7. Decoded Gold images for different interval lengths under BER of 10^{-4} in the case of the baseline process: (a) without restart intervals, PSNR = 6.19 dB; (b) restart length of 30 MCU/interval, PSNR = 25.85 dB; (c) restart length of 10 MCU/interval, PSNR = 31.90 dB.

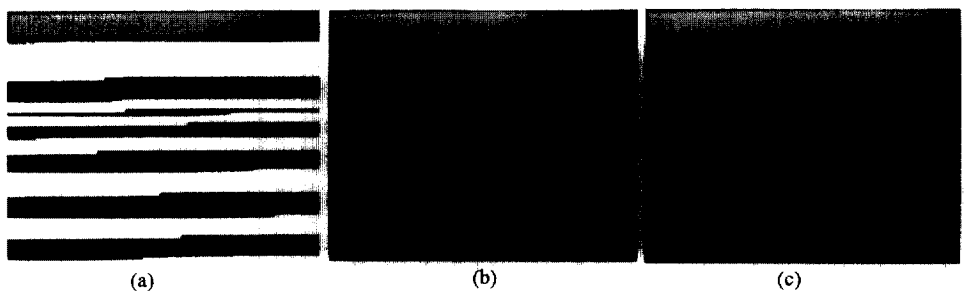


Fig. 8. Decoded Gold images for different interval lengths under BER of 10^{-4} in the case of the extended process with arithmetic coding: (a) without restart intervals, PSNR = 5.77 dB; (b) restart length of 30 MCU/interval PSNR = 21.96 dB; (c) restart length of 10 MCU/interval, PSNR = 29.82 dB.

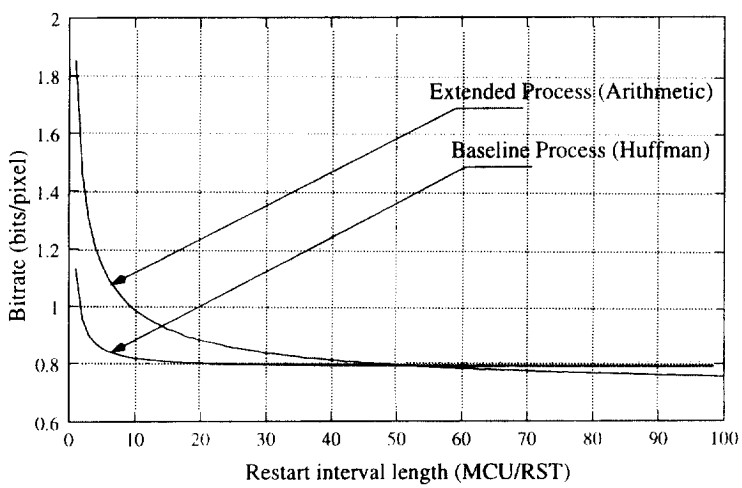


Fig. 9. Compression ratios of the Gold image coded with different restart interval lengths.

ratios versus the lengths of the restart intervals are shown, for a bit error rate of 10^{-4} and, respectively, for Huffman and arithmetic based codings. Figs. 7 and 8 present some visual results. In Fig. 9 the compression ratios versus the lengths of the restart intervals are shown. Because of the adaptivity of the arithmetic coding, the corresponding compression ratio is affected by the length of the restart intervals. For restart interval lengths smaller than 10 MCU/RST the compression ratio of the extended process is affected. Fixed JPEG tables are employed and the overhead is only due to the restart point markers.

5. Error detection

Three techniques are proposed to detect the different types of transmission errors.

5.1. Error detection by coherence tests

The first coherence level which is tested is relative to the transfer format markers and segments. An incoherence in the received transfer format structure is considered as an error. For example, in the case of a block split or merge, the count of decoded blocks in a restart interval is, respectively, smaller or greater than the number of coded blocks in a restart interval as indicated in the DRI segment. Thus, decoding a different number of blocks in a restart interval indicates a merge or split error in the interval. Furthermore, by taking into acount the information contained within the frame and scan headers, some errors can be detected such as receiving an incoherent marker or segment. The detection of this type of errors is very important for the whole decoding process, since it can avoid interruptions.

Concerning the coherence tests at the block and entropy-coding levels, they depend on the entropy-coding technique used. In the case of the base-line process with Huffman coding, three coherence levels are considered. The first one deals with the received Huffman code words. In some cases, the decoder reads words with length exceeding the maximal permitted length for Huffman code words (16 bits). In this case, it is clear that the received word is not recognized in the Huffman table and it is declared as an error. The decoder ignores the first bit and proceeds to another code-word identification from the second bit. This allows a quicker decoder synchronization than if all the bits are ignored. Once a code word is recognized, it is decoded as the current (RUN, SIZE) value. At the block level, all the AC coefficients following the erroneous coefficient are considered erroneous, and they are concealed later in the process.

The coherence tests at the run-length decoding level consist in detecting incoherent values of SIZE and incoherent successions of (RUN, SIZE) values. For example, the succession of two EOBs or four (15,0) symbols are incoherences in the decoding process. At the block level, with certain errors, the number of decoded coefficients in a block may be greater than 64, which corresponds to an error. In order to detect directly split and merge errors, a bit-rate test is done for each erroneous block. The decoder compares the average bit-rate of the decoded block D_b, with the average bit-rate D_m of the surrounding blocks. The decoder detects a block split error if $D_b < 0.2D_m$, and it detects a merge error if $D_b > 5D_m$, where the values 0.2 and 5 are fixed empirically.

Concerning the arithmetic coder, only coherence tests at the transfer format structure level are considered. The decoder cannot detect incoherences in the arithmetic entropy received bit stream. Although the coherence test detection techniques are reliable, they do not detect all the errors, and additional detection methods are needed.

5.2. Error detection in the frequency domain

The magnitudes of the current block AC coefficients are compared with those of the neighboring blocks. Taking into account the AC coefficient distribution [16, 23], the comparison is based on the fact that frequency

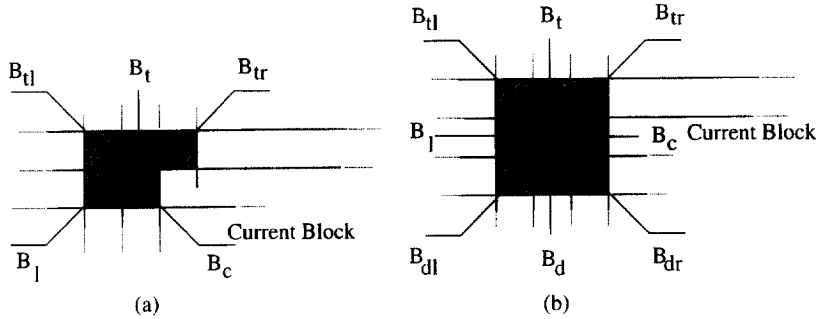


Fig. 10. Neighborhoods for error detection and concealment: (a) prediction neighborhood; (b) interpolation neighborhood.

active zones in the natural images are not isolated. It is rare to find an isolated AC coefficient with a large magnitude and there will be at least an adjacent block with a similar AC coefficient at the same rank in the zigzag order. As shown in Fig. 10, two types of block neighborhoods are used.

The prediction neighborhood uses four surrounding blocks. If k is the rank of the current AC coefficient in the current block, the coefficient is declared erroneous if one of the two conditions is fulfilled:

$$|B_c(k)| \geqslant \alpha(\max(|B_l(k)|, |B_t(k)|, |B_{tl}(k)|, |B_{tr}(k)|) + 1) \tag{1}$$

or

$$|B_c(k)| \leqslant \beta(\min(|B_l(k)|, |B_t(k)|, |B_{tl}(k)|, |B_{tr}(k)|)). \tag{2}$$

The functions min and max return, respectively, the minimum and maximum of their arguments. The coefficients α and β depend on the rank of the coefficient and we have $\alpha = 1/\beta = k + 1$ which is suited for the coefficient magnitude evolution in the block.

The interpolation neighborhood uses seven surrounding blocks. The block to the right is not used because the errors propagate in that direction, thus we have a high probability that the right block is erroneous if the current one is damaged. The AC magnitude tests are similar to those of the prediction neighborhood, and the following values are used:

$$c\_\mathrm{hor}(k) = B_l(k), \tag{3}$$

$$c\_\mathrm{vert}(k) = \frac{(B_t(k) + B_d(k))}{2}, \tag{4}$$

$$c\_\mathrm{diag1}(k) = \frac{(B_{tl}(k) + B_{dr}(k))}{2}, \tag{5}$$

$$c\_\mathrm{diag2}(k) = \frac{(B_{tr}(k) + B_{dl}(k))}{2}. \tag{6}$$

In Eqs. (1) and (2) we replace $B_l(k)$, $B_t(k)$, $B_{tl}(k)$ and $B_{tr}(k)$, respectively, by $c\_\mathrm{hor}(k)$, $c\_\mathrm{vert}(k)$, $c\_\mathrm{diag1}(k)$ and $c\_\mathrm{diag2}(k)$.

A variant of this method in the case of the interpolation neighborhood introduces weights on the coefficients. For example, weights can be greater than one for past blocks and less than one for future blocks. By doing so, the effects of the nondetected errors in the future blocks are attenuated.

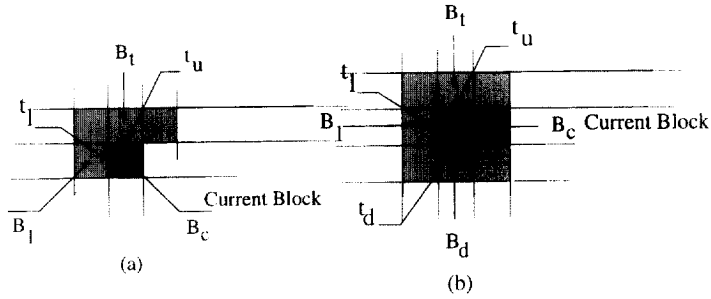


Fig. 11. Neighborhoods for spatial error detection: (a) prediction neighborhood (two boundaries); (b) interpolation neighborhood (three boundaries).

5.3. Error detection in the spatial domain

The aim is mainly the detection of a DC coefficient error and some large AC errors. Errors on DC coefficients correspond to discontinuities in adjacent block boundaries. Taking into account that the evolution of pixel values in natural images is smooth, a large variation in pixel value is considered as an error. The proposed technique compares the average value of the block boundaries weighted by the inverse of their variance with those of the neighborhood blocks. This concept was previously used for edge detection [9].

Let B_1 and B_2 be two blocks with a shared boundary and let m_1 and m_2 be, respectively, the average values of the pixels of B_1 and B_2 boundaries and σ_1^2, σ_2^2 their variances. The comparison of the borders of the two adjacent blocks is based on the value

$$t = 2\frac{|m_1 - m_2|}{\sigma_d}, \tag{7}$$

where

$$\sigma_d^2 = \frac{\sigma_1^2 + \sigma_2^2}{2}. \tag{8}$$

According to [9], the optimal value of t to detect natural edges with 8 pixels in both sides of the edge is: $t_{\text{opt}} = 2.92$. Thus, an edge is detected if $t > t_{\text{opt}}$. Here, transmission errors replace natural edges. In order to preserve natural edges but detect transmission errors, it has been observed that the threshold must be higher than t_{opt}. A dynamic threshold which takes into account the image context has been determined:

$$t_{\text{threshold}} = t_m(\text{last correct block}) + t_{\text{opt}}, \tag{9}$$

where t_m(last correct block) is the average of the t values of the boundaries of the last correctly decoded block. The initial value of the threshold corresponds to twice t_{opt}.

As shown in Fig. 11, the two neighborhoods correspond to two and three block boundaries. For the prediction neighborhood, an error is detected if $t_t > t_{\text{threshold}}$ and $t_l > t_{\text{threshold}}$. In the case of the interpolation neighborhood, a third boundary is tested, thus an error is detected if $t_t > t_{\text{threshold}}$ and $t_l > t_{\text{threshold}}$ and $t_d > t_{\text{threshold}}$. For high bit error rate (BER $= 10^{-3}$), the test of the lower boundary must be weighted because the lower block may be erroneous and not yet concealed.

It is worth mentioning that, for the frequency and spatial error detections, the blocks situated at the borders of the image are treated as special cases with the available neighborhood.

Table 1 illustrates the performance of the different methods. Most of the errors are detected by the spatial domain technique. In practice, an ordering is introduced: coherence tests are performed first, then

Table 1
Percentage of errors detected by the different methods

	Baseline decoder	Extended decoder
Error detection by coherence tests	35%	15%
Error detection in the frequency domain	10%	7%
Error detection in the spatial domain	70%	86%

frequency-domain detection and finally spatial domain detection. Once an error is detected, the subsequent tests are dropped.

6. Error concealment and error compensation

Once an error is detected, the decoder takes several actions relative to damaged block concealment and error propagation limitation. The error concealment is performed in the frequency domain, and, with a reduced set of computations, the edges and frequency activity of the block are restored.

In the case of the baseline process, when an error occurs, in most of the cases the decoder recovers synchronization after decoding a few blocks. The remaining blocks in the restart interval are not damaged. In order to preserve them, the detection technique returns the type of the detected error. Five types of errors have been distinguished:

- Merge error: the decoder conceals the current block, discards the entropy data until the next EOB and conceals directly the next block.
- Split error: the decoder conceals the current block, discards the data until the next EOB and removes the next block.
- Incoherence in the entropy data: if the error is detected within a DC coefficient or low rank AC coefficients (rank less than 6 in the zigzag order), the whole block is concealed. In the other cases only the current AC coefficient and the remaining ones in the current block are concealed.
- Spatial error: the whole block is concealed.
- Frequency-domain error: if the rank of the coefficient is less than 6, the whole block is concealed. Otherwise, only the current and remaining coefficients in the block are concealed.

The error propagation limitation in the case of the baseline process is achieved by shifting blocks in the case of a split or merge errors and propagating the DC concealment over the remaining blocks in the restart interval.

In the case of the extended process with arithmetic coding, all the errors are processed in the same manner. The current block and remaining ones in the restart interval are concealed.

The damaged block concealment is similar for the baseline and extended processes. In the case of the prediction neighborhood shown in Fig. 10, the DC coefficient is concealed by the average value of the surrounding DC values:

$$B_c(0) = \frac{(B_l(0) + B_t(0) + B_{tl}(0) + B_{tr}(0))}{4}, \tag{10}$$

while the AC coefficient of the rank k in the zigzag order is concealed by

$$B_c(k) = \frac{(\mathrm{maxabs}(B_l(k), B_t(k)) + \mathrm{maxabs}(B_{tl}(k), B_{tr}(k)))}{2}. \tag{11}$$

The function maxabs returns the coefficient which has the maximum absolute value. This concealment takes into account the horizontal, vertical and diagonal edges of the surrounding blocks. In the case of the prediction neighborhood, the typical deviation of the differences between the concealed block boundaries and those of the top and left blocks is subtracted from the concealed block pixel values.

In the case of the interpolation neighborhood shown in Fig. 10, the DC coefficient is concealed by the average of the surrounding blocks:

$$B_c(0) = \frac{(B_l(0) + B_t(0) + B_{tl}(0) + B_{tr}(0) + B_{dl}(0) + B_d(0) + B_{dr}(0))}{7}, \tag{12}$$

and the AC coefficients are concealed by the same equation as (11) by replacing $B_l(k)$, $B_t(k)$, $B_{tl}(k)$ and $B_{tr}(k)$, respectively, by $c\_\mathrm{hor}(k), c\_\mathrm{vert}(k), c\_\mathrm{diag1}(k)$ and $c\_\mathrm{diag2}(k)$ given in Eqs. (3)–(6).

In addition to damaged block concealment, the decoder has to recover synchronization under incoherences in the transfer format structure. A modified version of the synchronization procedure used in [12] has been employed. It is used when an incoherent marker is decoded or when there is a marker missing. The aim is to recover decoder synchronization, and avoid decoder interrupts and long error propagations. The algorithm is as follows:

```
Notice error for the concealment procedure
Find the next marker
if found_marker is not a JPEG marker
  go to action_2
else
  if found_marker is valid but different from (EOI, RST0-RST7)
    if found_marker is coherent
      go to action_1
    else
      go to action_2
  if found_marker is an RST marker
    if found_marker = = desired_marker + 1 (or +2)
      go to action_1
    else
      go to action_2
action_1:
    put the found_marker in the bitstream and decode empty segments
    /* after three successive empty decoded segments the found marker
    is accepted as an RST marker*/
    skip action_2
action_2:
    desired_marker = found_marker
    continue normal decoding
```

Fig. 12 shows the decoder arrangement and the location of the procedures. The dotted lines indicate whether an error is detected and, in the case of the baseline process, they indicate also the error type. First, the decoder proceeds to coherence tests, then frequency-domain error detection and finally spatial domain error detection. After each detection procedure, the concealment technique is invoked if an error is detected. In the case of the prediction neighborhood, the prediction and concealment techniques are performed on-line, and a damaged block is immediately concealed. In the case of the interpolation neighborhood, an additional row is

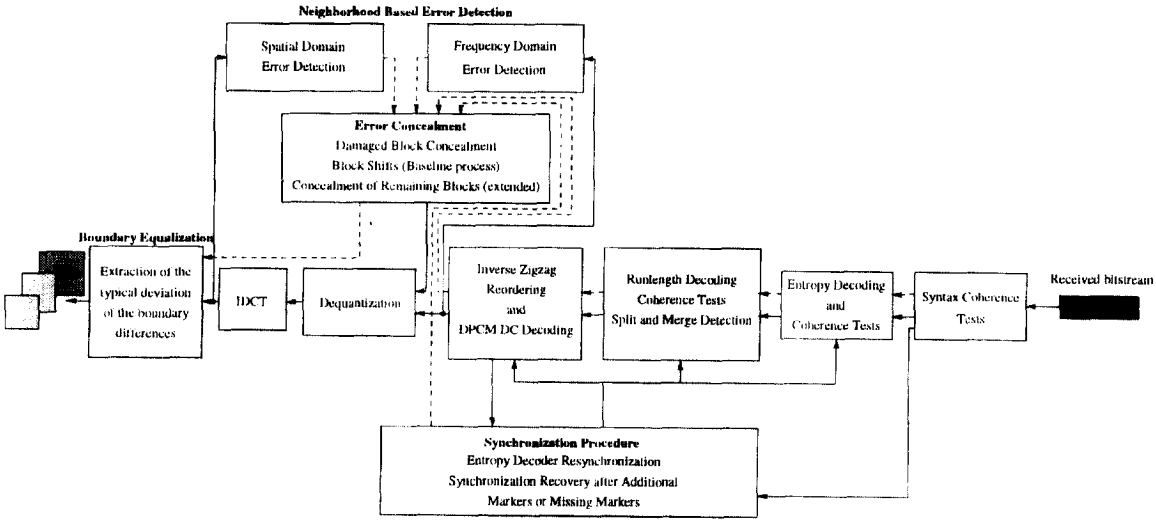


Fig. 12. Robust decoder architecture.

needed to get the complete neighborhood. The same synchronization procedure is employed for the baseline and extended processes.

7. Results

The results presented concern the image "Gold" which is a 720×576 one-component image. An MCU corresponds to one coded DCT block. The image is coded with 15MCU per restart interval, which corresponds to a loss in compression ratio of 1.94% for the baseline process and 18.60% for the extended process. Isolated bit errors at random position have been considered in these simulations. Fig. 13 presents the PSNRs obtained versus the bit error rate (BER) for the two coding processes with the prediction neighborhood. The improvements provided by robust decoders with respect to JPEG conventional decoders is significant. For bit error rates around 10^{-4}, the PSNR improvement is about 5 dB in the case of the base-line process. The robust decoder shows less interruptions than the conventional JPEG decoder. In the case of the extended process, the PSNRs are smaller and the improvement provided by the robust decoder with respect to the conventional JPEG decoder is about 3–4 dB for most bit error rates. The main problem with arithmetic coding is that the decoder diverges progressively after a transmission error, the detection techniques are less efficient, and more blocks have to be concealed.

The results of the interpolation neighborhood are shown in Fig. 14. More than 5 dB of improvement are provided by the robust decoder with respect to the conventional JPEG decoder for the base-line process. For the extended process the gain is about 5 dB.

One can observe that the curves associated with the conventional decoder in Figs. 13 and 14 are not exactly the same. This is due to the distributions of random errors, which are not the same in both simulations, for the same rates.

In order to complete the objective results, Fig. 15 presents visual results obtained for the "Gold" image under a bit error rate of 2×10^{-4} and a 15MCU per restart interval length. The quality of the images decoded by the robust decoders looks better than that of the images decoded by JPEG conventional decoders. The

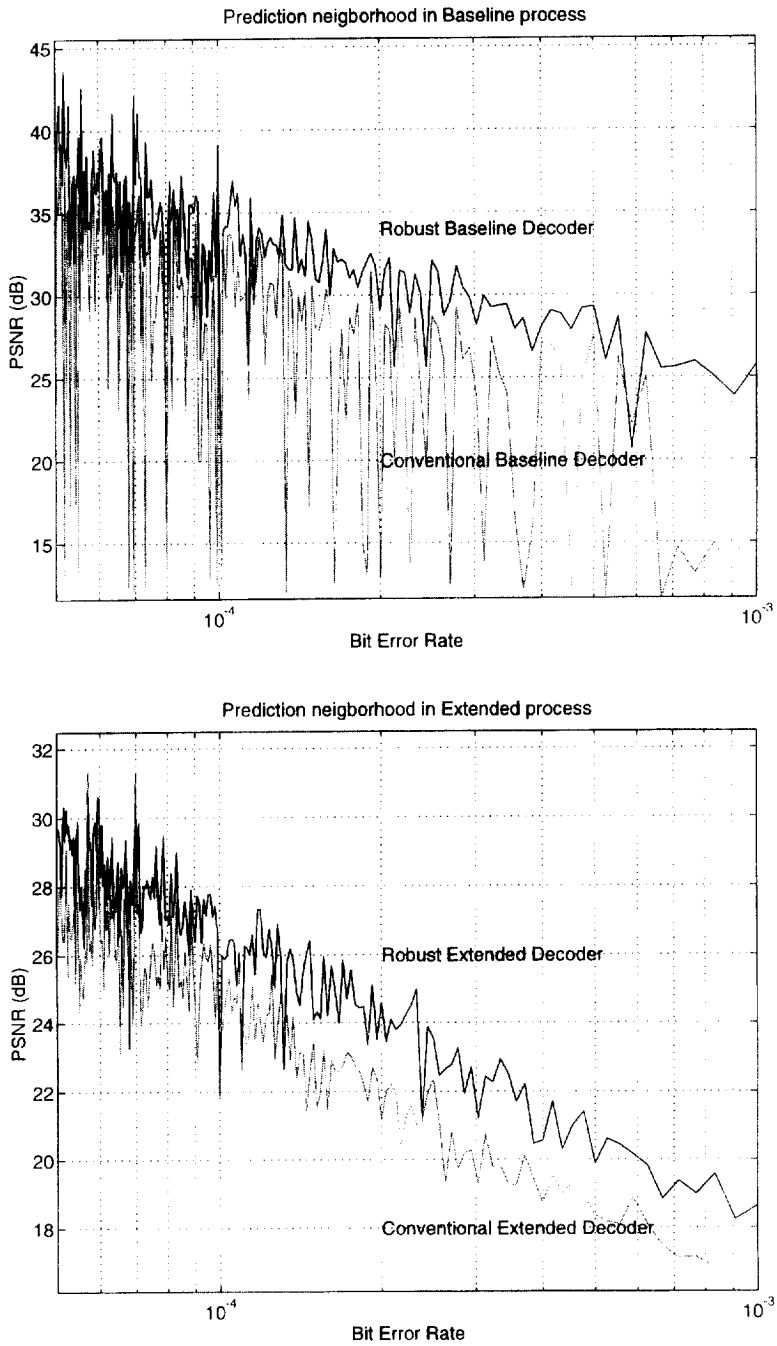


Fig. 13. PSNRs obtained by the robust and conventional JPEG decoders for the image "Gold" coded with 15MCU/RST in the case of the prediction neighborhood.

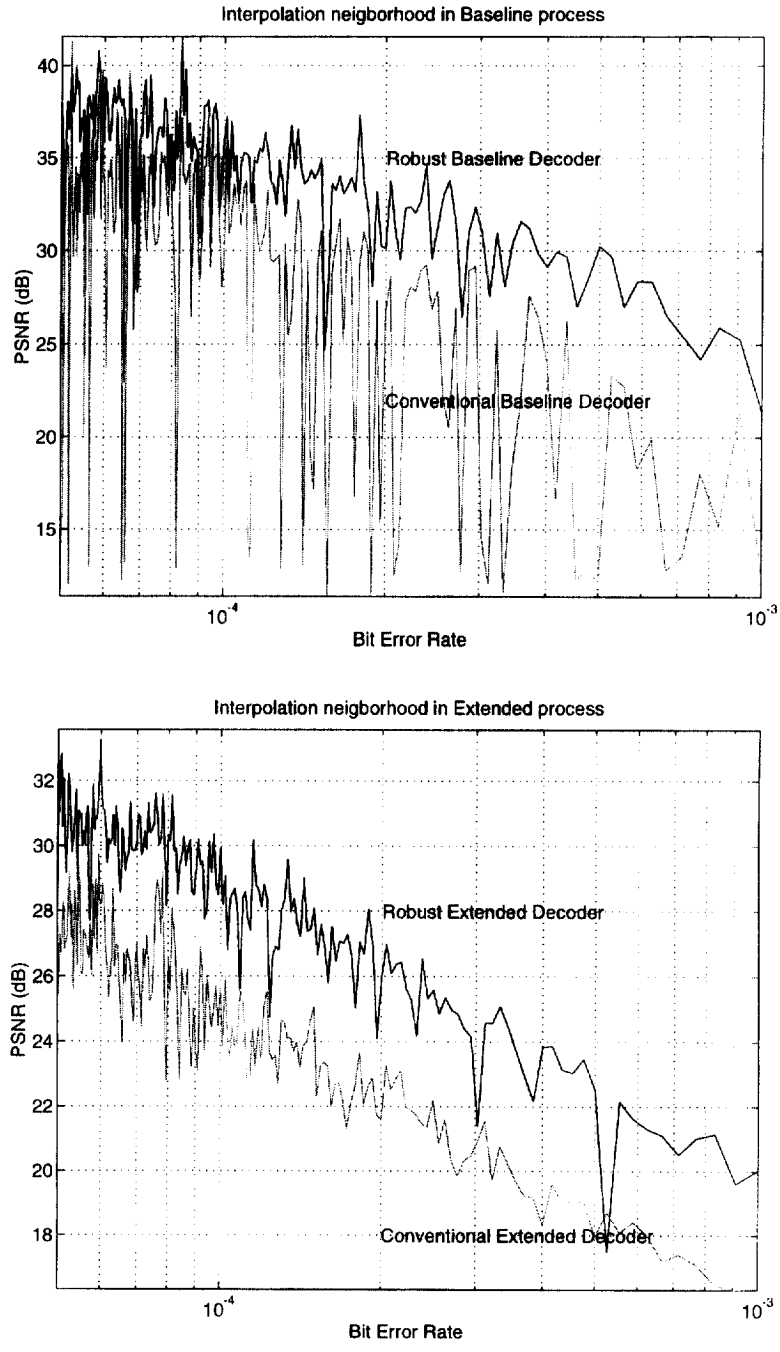


Fig. 14. PSNRs obtained by the robust and conventional JPEG decoders for the image "Gold" coded with 15MCU/RST in the case of the interpolation neighborhood.

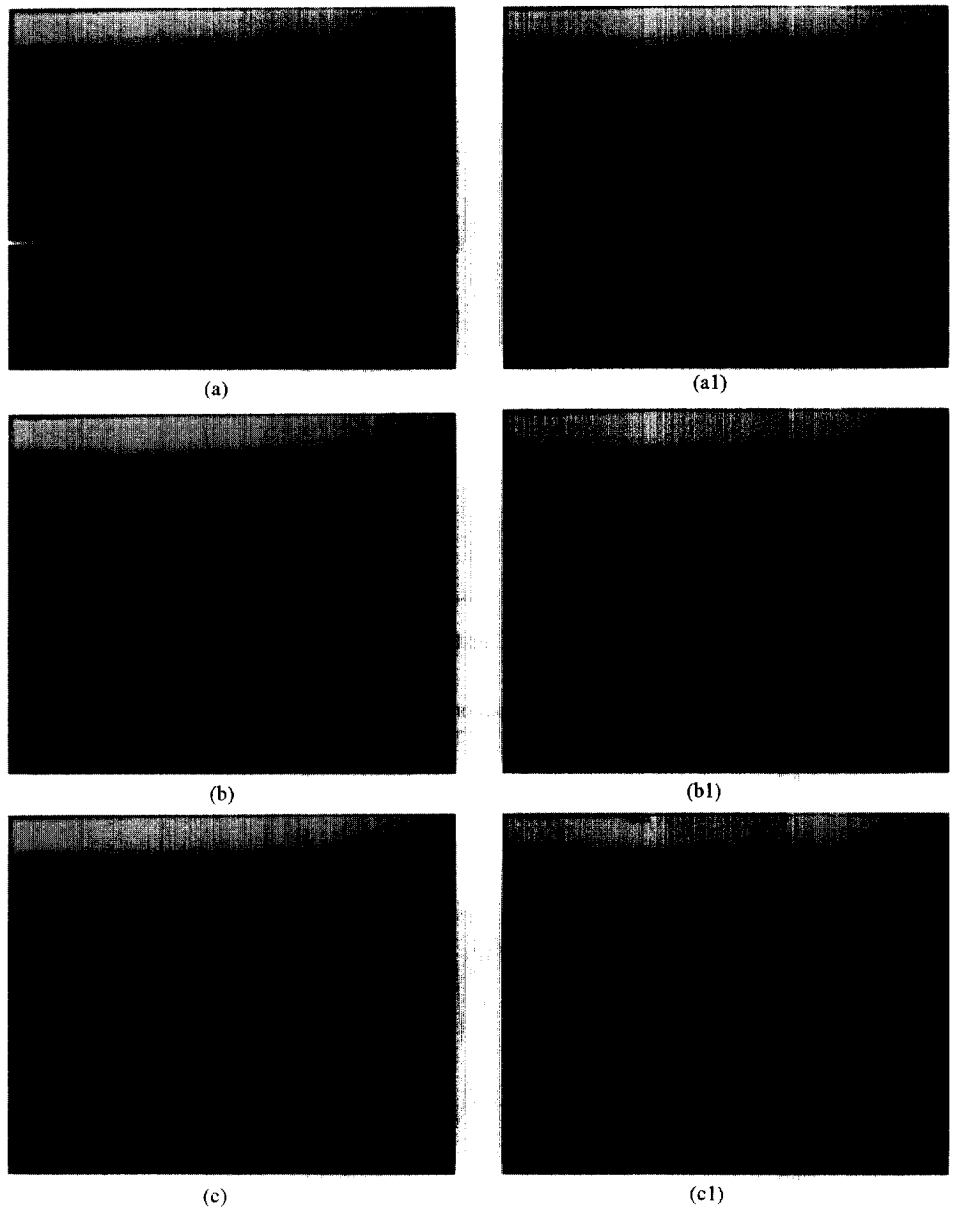


Fig. 15. "Gold" image decoded by the robust and conventional JPEG decoders with a 15MCU/RST interval length and under a BER of 2×10^{-4}: (a) conventional baseline process, PSNR = 25.50 dB; (b) robust baseline process with prediction neighborhood, PSNR = 29.94 dB; (c) robust baseline process with interpolation neighborhood, PSNR = 31.15 dB; (a1) conventional extended process, PSNR = 21.09 dB; (b1) robust extended process with prediction neighborhood, PSNR = 23.75 dB; (c1) robust extended process with interpolation neighborhood, PSNR = 26.29 dB.

interpolation neighborhood shows much better results than the prediction neighborhood. More detailed objective and subjective results can be found in [1].

8. Conclusions

A complete strategy has been described, to counter the effects of transmission errors in JPEG coded images. It covers both the baseline process and the extended process with arithmetic coding and it contains essentially two phases: the introduction of restart intervals to stop error propagation and a procedure to detect and conceal damaged blocks.

As a global assessement, it can be stated that the baseline process turns out to be more robust than the extended process. For the first phase, in practical applications, it is recommended to retain about 20MCU as the restart interval length. A greater length is likely to be needed for the extended process, depending on the trade-off between compression rate and robustness. Now, for the second phase and the detection and concealment of erroneous blocks, the approach based on prediction neighborhoods is simpler to implement. However, a better quality of the decoded images is achieved with the interpolation neighborhoods, which impose additional constraints in delay and storage capacity.

Only isolated bit errors have been considered in the simulations. Other types of errors can occur in transmission systems, like burst errors in radio channels or cell loss in ATM networks and the situation at the image decoder input is dependent on the techniques implemented to correct the channel errors. The extension of the results obtained above in this context is a matter for further work, as well as the adaptation of the proposed techniques to animated images, particularly in the context of the MPEG family of standards.

References

[1] M. Abdat, Etude des techniques de compression des images fixes et amélioration de la résistance aux erreurs de transmission, Thèse de Docteur du Conservatoire National des Arts et Métiers (CNAM), Paris, Décembre 1995.
[2] N.T. Cheng, N.G. Kingsbury, The ERPC: An efficient error resilient technique for coding positional information or sparse data, IEEE Trans. Comm. 40 (1) (1992) 140–148.
[3] L.T. Chia et al., On the use of transform domain information for concealment of errors in JPEG images, Proc. EUSIPCO'94, September 1994, pp. 616–619.
[4] M. Ghanbari, An adapted H.261 two-layer video codec for ATM networks, IEEE Trans. Comm. 40 (9) (1992) 1481–1490.
[5] R.G. Gonzalez, R.E. Woods, Digital Image Processing, Addison-Wesley, New York, 1992.
[6] ISO/IEC, IS 10918, ITU-T, Rec. T.81, Information technology – Digital compression and coding of continuous-tone still images, Part 1: Requirements and Guidelines, 1992.
[7] ISO/IEC DIS-13818-2, ITU-T H.262, Coding of moving pictures and associated audio, May 1994.
[8] K. Joseph, S. Ng, D. Raychaudhuri, R. Siracusa, J. Zdepski, R. Saint Girons and T. Savatier, MPEG++: A robust compression and transport system for digital HDTV, Signal Processing: Image Communication 4 (4–5) (1992) 307–323.
[9] M. Kunt, G. Granlund, M. Kocher, Traitement numérique des images, Presses Polytechniques et Universitaires Romandes, Traitement de l'Information, Vol. 2, Juillet 1993.
[10] W.M. Lam, A.R. Reibman, Self-synchronizing variable-length codes for image transmission, ICASSP'92, 1992, Vol. 3, pp. 477–480.
[11] W.M. Lam, A.R. Reibman, An error concealment algorithm for images subject to channel errors, IEEE Trans. Image Process. 4 (5) (1995) 533–542.
[12] T.G. Lane, The independent JPEG group's JPEG software, Release 3, March 1992.
[13] X. Lee, Y.Q. Zhang, A. Leon-Garcia, Information loss recovery for block-based image coding techniques – A fuzzy logic approach, IEEE Trans. Image Process. 4 (3) (1995) 259–273.
[14] G.G. Longdon Jr., An introduction to arithmetic coding, IBM J. Res. Dev. 28 (2) (1984) 135–149.
[15] O.R. Mitchell, A.J. Tabatabai, Channel error recovery for transform image coding, IEEE Trans. Comm. 38 (12) (1981) 1741–1754.
[16] F. Muller, Distribution shape of two-dimensional DCT coefficients of natural images, Electron. Lett. 29 (22) (1993) 1935–1936.
[17] P. Nasiopoulos, R.K. Ward, A hybrid coding method for digital HDTV signal, ICASSP'95, 1995, pp. 769–772.
[18] J.W. Park, D.S. Kim, S.U. Lee, On the error concealment technique for DCT based image coding, Proc. ICASSP'94, 1994, Vol. III, pp. 293–296.

[19] W.B. Pennebaker, J.L. Mitchell, JPEG: Still Image Data Compression Standard, Van Nostrand Reinhold, New York, 1993.
[20] W.B. Pennebaker et al., An overview of the basic principles of the Q-coder, IBM J. Res. Dev. 32 (6) (1988) 717–726.
[21] D. Raychaudhuri, H. Sun, R.S. Girons, ATM transport and cell-loss concealment techniques for MPEG video, ICASSP'93, 1993, Vol. I, pp. 117–120.
[22] D.W. Redmill, N.G. Kingsbury, Still image coding for noisy channels, Proc. IEEE Internat. Conf. Image Processing, ICIP 94, November 1994, pp. 95–99.
[23] R.C. Reininger, J.D. Gibson, Distribution of two dimensional DCT coefficients for images, IEEE Trans. Comm. COM-31 (6) (1983) 835–839.
[24] G.K. Wallace, The JPEG still picture compression standard, Comm. ACM 34 (4) (1991) 31–34.