ICON BASED ERROR CONCEALMENT FOR JPEG AND JPEG 2000 IMAGES

Michael Gormish

Ricoh Innovations, Inc., Menlo Park, California, gormish@rii.ricoh.com

ABSTRACT

This paper describes methods to recover the useful data in JPEG and JPEG 2000 compressed images and to estimate data for those portions of the image where correct data cannot be recovered. These techniques are designed to handle the loss of hundreds of bytes in the file. No use is made of restart markers or other optional error detection features of JPEG and JPEG 2000, but an uncorrupted low resolution version of the image, such as an icon, is assumed to be available. These icons are typically present in Exif or JFIF format JPEG files.

1. INTRODUCTION

1.1. Error detection and correction

Error prevention, detection, correction, and concealment has been extensively studied and reported on in the literature. The closest work to that described here come from video error concealment rather than from still systems. An excellent bibliography for these methods appears in [3].

Because small errors can have devastating effects on compressed files (in the worst case a single bit error could render the entire image unreadable), image compression standards have often included error detection and recovery techniques. JPEG provides for restart markers which cause the encoding and decoding process to be reset after some amount of coded data. Distortion due to any errors is thus limited to the region between restart markers. JPEG 2000 already encodes image portions independently to allow for random access, but several options were included to improve the ability to detect and recover from errors including: "segmentation symbols," "termination on every coding pass," and SOP and EPH markers.

However, error handling features of the compression standards are often not used. Use of additional redundancy to allow error detection or correction tends to increase the bitrate, something compression people are loath to do. Further, most transport protocols effectively provide an error free channel by detecting errors and requesting retransmission. The error protection is thus wasted.

1.2. Errors due to USB transfer

Digital cameras are typically assumed to operate in an error free environment and thus tend not to employ error correction features. Never the less, Figure 1 shows several images taken with the author's digital camera and transferred to a computer while the camera had a low battery. The images appeared to have been received correctly by the computer, so the compact flash card was erased. However, many applications refused to open these images, and even those applications that would open them would display completely ruined images. Examination of the compressed file revealed that several hundred bytes of data had been replaced with 0 bytes. The images initially appeared to have been transferred correctly only because the icon was transferred correctly.

Figure 1 - JPEG images with errors

Amazingly, even though hundreds of bytes were lost, Figure 1 reveals that a persistent Huffman encoder does eventually resynchronize. It is obvious that there is useful data available for the lower portion of the image, although the position and color is wrong. The loss of position occurs because the bytes with errors decoded to a different number of 8x8 blocks than the correct bytes would have. The color is incorrect in the lower portion of the images because JPEG codes the DC coefficient of each 8x8 block as a difference value with the previous coefficient. Because of the large number of lost differences the correct DC values (for all color plans are unknown.

Figure 2 shows schematically an image with two large errors in the codestream. The first region before any errors have occurred in the codestream appears correctly. Then the "Error Region" begins which may be a solid color band,

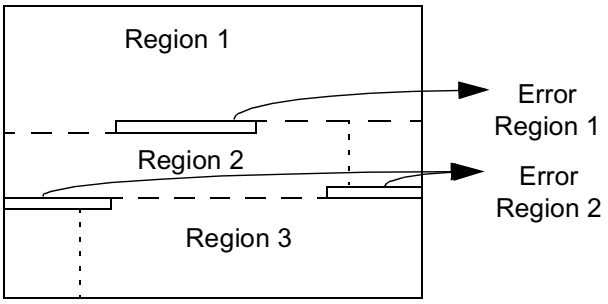

Figure 2 - Regions in an image with multiple errors

or an area with high amounts of visible noise. Eventually a good decoder resynchronizes with the codestream and produces recognizable image data in Region 2. Although the decoder resynchronized the beginning of blocks in the codestream the position in the decoded image is wrong. This leads to a large horizontal line where the second region has been shifted from the first region. It also leads to a vertical line artifact where the right edge of the picture is. Because Region 2 is shifted the left and right edges of the correct image are displayed next to each other in the visible portion of the image. In addition to the artifacts caused by the shifting of the image, the color of the image is typically wrong in Region 2. Images with more than one large error simply have an additional error region, and an additional visible horizontal and vertical line and intensity shift. The second error region in Figure 2 wraps around the edge of the displayed image

1.3. Common JPEG file formats

Examination of the files produced by the author's camera (and indeed all Ricoh, Canon, and Olympus cameras examined), revealed that no restart markers had been included in the produced codestreams. Thus the errors typically lead to the lost of the rest of the image.

However, all of the cameras included an icon in the beginning of the file. Indeed the two most common JPEG file formats: JFIF and Exif [1] both allow for storage on an icon, and other information such as color space. Further more, the Design Rule for Camera File System (DCF), is a standard that requires JPEG files to be in Exif format, and *requires* a 160 by 120 icon[2]. Therefore this paper examines methods to detect, correct, and conceal very larger errors occurring in JPEG and JPEG 2000 files.

2. JPEG ERROR CONCEALMENT

While in the author's case the error region was easily identified by the string of zero bytes, it is not known how many 8x 8 blocks the error corresponds to in the image. In general, the location of the error may not be trivial to find. Thus this section provides algorithms for determining the location of a large error (and the mostly correct region after the error), determining the correct DC values to use in the regions beyond the first region, and estimating values to use in the error regions. This process is shown pictorially in Figure 3 and explained in detail in the following sections.

2.1. Align compressed data correctly

Every 8x8 block in the error codestream begins with a DC value which has been encoded by subtracting the previous block's DC value and then Huffman coding the difference. This list of these "differential DC" values may have substantially more or substantially fewer DC values than the number of 8x8 blocks in the image because the errors may have lead to insertion or deletion of codeblocks.

The icon image can be scaled to the size of the full image by standard interpolation methods e.g. pixel replication or bilinear interpolation. The scaled icon should be compressed using the same quantization table as was used to compress the error codestream. The compressed scaled icon can be decoded in the same manner as the codestream with errors yielding a list of differential DC (DDC) values for the icon.

Let I(b) be the differential DC value of the bth block of the scaled icon image. Let E(b) be the differential DC value of the error image. Let A(s) be the average absolute error between the signals when the error signal is shifted by s. That is:

$$M(s) = \frac{1}{N}\sum |I(b) - E(b-s)| \qquad (1)$$

where the sum is over all samples which overlap after shifting, and N is the number of sample which overlap. A sample is the value of a "pixel" in a single color component, e.g. the luminance value at a particular location in the image.

A typical graph of M(s) has several clear local minimums. The local minimums in M(s) indicate the amount of shift caused by errors in the compressed data. Typically there is a minimum at s=0 because the error is usually not at the very beginning of the codestream, and thus some of the codestream lines up correctly. There are also typically repeated local minimums corresponding to the number of 8x8 blocks in the width of the image. This occurs because the DC values are typically similar to the ones above and below. If there is only one error region there should be only two primary local minimums one at s=0 and one at the shift which aligns the rest of the image. Each additional error region will lead to a new local minimum.

Compute the how well the different shifts of the full image match at each location of the icon image:

$M1(b) = (I(b) - E(b-s1))^2$

$M2(b) = (I(b) - E(b-s2))^2$

M1 and M2 indicate how well different shifts of the image match the scaled icon. Since s1 is assumed to be the shift for the first portion of the image and s2 for the second

Figure 3 - Image in various stages of correction: aligned (top left), DC corrected (bottom left), error region filled in (top right)

Figure 5 - JPEG 2000 image: with LL subband set to midpoint (top), with LL replaced from icon (bottom)

portion, we determine the cumulative error caused by using the error image with shift s1 up to position t, and then the error image with shift s2 beyond position t.

$$C(t) = \sum_{i=0}^{t} M1(i) + \sum_{i=t}^{\text{end}} M2(i) \qquad (2)$$

The value of t which minimizes C(t) is taken as an estimate of the location of the error region and the 8x8 block to change from using one shift to another. This estimate is sufficient for computing the correct DC offset to use after the error.

2.2. Adjust DC value of full image to match the icon's DC value

Each region of the corrected image receives pixel data from a different place. The first region can be decoded normally, until the first error. The region after the error is filled in by decoding and shifting the blocks spatially by the amount determine previously. The absolute DC values for all the color components (typically Y, Cb, and Cr) must be determined. This can be done by determining the overall average DC value for blocks in the region in both the error image and the icon image.

The average Luminance DC value in the error image assuming the previous absolute DC value was 0 is:

$$O_y = \frac{1}{t2 - t1} \sum_{b=t1}^{t2} E(b - s2) \qquad (3)$$

identical equations apply for Cb and Cr components by summing those differential DC values.

The average Luminance DC value in the icon image is

$$A_y = \frac{1}{t2 - t1} \sum_{b=t1}^{t2} DC(b - s2) \qquad (4)$$

this value is not a differential value. Thus the corrected image should obtain the same average DC value in this region. To do this, the corrected image should simply use Ay - Oy as the previous DC values, and apply the differences decoded from the shifted codestream as usual.

2.3. Fill in blocks declared as errors from the scaled icon

There are many methods to fill in the blocks declared to be in error. Some are discussed in [3]. In the case where the icon is available, the simplest method is to copy the error region from the scaled version of the icon. This will yield an accurate replacement for the DC values, but the filled in region will not contain any high frequency content. Thus other techniques for filling in the error region can be used in combination with copying. Techniques which determine the value to fill in the frequency domain can easily be combined by using only the non DC values in the 8x8 blocks.

3. JPEG 2000 ERROR CONCEALMENT

Although there are not currently shipping digital cameras with JPEG 2000 we can anticipate their creation and examine correcting the same type of error. JPEG 2000 has very different properties than the DCT based original JPEG. In theory, icons are unneeded because a 1600x1200 image with 5 levels of wavelet transform has an 50x38 LL sub-

band, which could be used as an icon. Additional levels of wavelet coefficients could be decoded to provide 100x75 or 200x150 images. However, when legacy systems are updated to use JPEG 2000 they may still use Exif and DCF and thus provide a separately coded icon. Further, some systems may depend on an icon that is exactly 160x120 not 100x75 or 200x150. Operating systems or image browsing programs may create their own independently coded icons. Thus the situation with an uncorrupted icon, but a corrupt image may still occur. We find that using an independently coded icon provides an extremely simple way to conceal errors in the JPEG 2000 file.

3.1. Determine Error Location

For any given system e.g. a camera with a 1600x1200 image and a 160x120 icon and medium high compression, it should be possible to determine typical similarities between the wavelet coefficients from the original image and those from the icon. For the low frequency subbands this match should be very close, and for high frequency subbands (where the icon has no data) there is no match at all.

As shown in Figure 4, wavelet subbands from the presumed valid icon can be compared with corresponding subbands from the image. If the wavelet coefficients from a scaled icon differ significantly for any code block, an error can be declared for that code block. Further if there is an error in a packet header and synchronization is lost the whole subband (or precinct) can be declared in error.

In practice it is not necessary to expand the icon to the full size of the original image and compress. A smaller expansion could be done and only the lower frequencies of the wavelet transform compared. This would save computation time and memory.

3.2. Replace coded data with data from icon

In the event of an error, data can be replaced on a code block by codeblock basis with data from the scaled icon. If this is done on a compressed codestream, packet headers will have to be rewritten. For the case of loss of an entire packet, the packet from the low resolution image can be used. This process is shown in Figure 4.

3.3. Example

The upper image in Figure 5 was created by compressing an image with JPEG 2000, causing an error in the LL subband and replacing that subband by 0 (if the subband was left with the error the image was much worse). In this case there were enough levels of wavelet transform that the LL subband fit in a single codeblock, thus the errors effect the entire image. If there were fewer levels of wavelet transform, or a higher level subband contained the error, the effect would be more localized.

If the complete LL subband is replaced with the LL subband created by scaling an icon to full size and compressing it in the same way as the original image was compressed the lower image in Figure 5 results. This image even printed at full size, looks every bit as good as an original. (In terms of MSE the error is relatively high, but this is a perfect example of a case where MSE is a very poor measure of the human response to image quality).

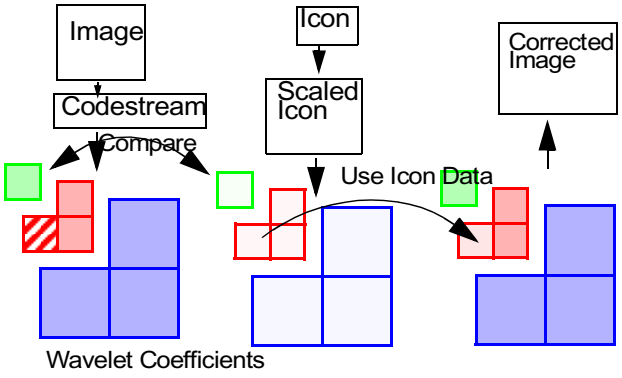

Figure 4 - JPEG 2000 Error Concealment

4. CONCLUSION

For the images ruined by the low power USB transfer, the techniques in the paper mean the difference between usable images and lost memories. No PSNR values are possible since the original image was lost. Furthermore, PSNR turns out to be relatively useless in simulations of similar errors. This is because small errors in the DC value are unnoticeable visually, but increase PSNR greatly. While the methods presented here are particularly useful for binary symmetric error channels, they may be useful for burst errors as long as there are only a few bursts within an image.

5. REFERENCES

1. Japan Electronic Industry Development Association (JEIDA), *Digital Still Camera Image File Format Standard (Exchangeable image file format for Digital Still Cameras: Exif)*, Version 2.1, June 12, 1998, available (in English) at http://www.pima.net/standards/it10/PIMA15740/exif.htm

2. Japan Electronic Industry Development Association (JEIDA), *Design Rule for Camera File System (DCF)*, adopted December 1998, available (in English) at http://www.pima.net/standards/it10/PIMA15740/dcf.htm

3. Yao Wang and Qin-Fan Zhu, "Error Control and Concealment for Video Communications: A Review," *Proceedings of the IEEE*, May 1998 Vol. 86, No. 5, pp. 974-997.

4. O. R. Mitchell and A. J. Tabatabai, "Channel error recovery for transform image coding," IEEE Trans. Commun., vol. COM-29, pp. 1754,1762, Dec. 1981.