Just One Bit in a Million: On the Effects of Data Corruption in Files

Volker Heydegger

Universität zu Köln, Historisch-Kulturwissenschaftliche Informationsverarbeitung (HKI),
Albertus-Magnus-Platz, 50968 Köln, Germany
volker.heydegger@uni-koeln.de

Abstract. So far little attention has been paid to file format robustness, i.e., a file formats capability for keeping its information as safe as possible in spite of data corruption. The paper on hand reports on the first comprehensive research on this topic. The research work is based on a study on the status quo of file format robustness for various file formats from the image domain. A controlled test corpus was built which comprises files with different format characteristics. The files are the basis for data corruption experiments which are reported on and discussed.

Keywords: digital preservation, file format, file format robustness, data integrity, data corruption, bit error, error resilience.

1 Introduction

Long-term preservation of digital information is by now and will be even more so in the future one of the most important challenges for digital libraries. It is a task for which a multitude of factors have to be taken into account. These factors are hardly predictable, simply due to the fact that digital preservation is something targeting at the unknown future: Are we still able to maintain the preservation infrastructure in the future? Is there still enough money to do so? Is there still enough proper skilled manpower available? Can we rely on the current legal foundation in the future as well? How long is the technology we use at the moment sufficient for the preservation task? Are there major changes in technologies which affect the access to our digital assets? If so, do we have adequate strategies and means to cope with possible changes? and so on. These are just a few of the general questions decision-makers have to carefully weigh.

Although many of the factors are fairly unpredictable, there are still some constants involved in the process of digital preservation which are likely to persist in time: these are the fundamental concepts which draw the link from information to data.

On the lowest level this is the storage of information as *binary-coded data*. No matter if information was or is or will be stored on a punched card, floppy disk, hard drive or holographic memory, this concept is likely to remain. In contrast, the favourite storage technology for digital data has changed many times in the past.

M. Agosti et al. (Eds.): ECDL 2009, LNCS 5714, pp. 315–326, 2009.

Moreover, binary data must be set into a specific context, it needs to be formatted. The meaning of data is primarily determined through a specific model of information. For example, a raster image is typically described as a set of pixels with some other additional elements like the arrangement of the pixel values in a two-dimensional array or a statement on its chromaticity. Besides these descriptive characteristics, which make up the appearance of the specific model 'raster image', digital data is meant to be processed by software. Therefore formatting of data also includes several technical characteristics which have a strong relation to the technical environment.

Such a *data format* for a specific model of information is another essential concept that links between information and data. Hence, it will always be a core element in the digital preservation task.

Given this, it is obvious that file formats[1] are spotlighted when it comes to discussing the strategies for long-term preservation of digital information. The focus to date was namely on their implications for reducing storage space on storage systems, openness of specification for the public, level of adoption in the academic and commercial communities and several functional issues (e.g., searchable text, metadata support) [4, 5].

So far little attention has been paid to the factor *robustness*, i.e., *a file formats capability to keeping its information as safe as possible in spite of data corruption*. A reason for that may be the perception of data corruption as a phenomenon that is mainly related to the physical storage media since bit errors actually happen there and can be detected and corrected on this level. Another reason could be the circumstance that we usually talk about 'damaged files' rather than 'damaged file formats' although a file is basically nothing but an individual physical realisation of the file format in question. Last but not least, there is a lack of systematic surveys on the impact of physical data corruption to file integrity both under lab conditions and in the real world. However, recently published studies on the reliability of disks and storage systems in the real world [2, 10, 11] which verified the existence of disk failures over time are an indication for the latent risk of file corruption as well since files are manifest to storage media.[2]

The paper on hand reports on the first comprehensive research on the topic of file format robustness. It is based on the assumption that there is indeed a correlation between 1) the way information is encoded in a file with respect to the definitions of the underlying file format and 2) information consistency. If this is so, it should be possible to sustain the robustness of a file format. This would result in a significant reduction of information loss for the case of corrupted data in files.

Theoretical groundwork including preliminary results of experimental tests have already been undertaken and reported by the author [6]. Both have been extended to an in-depth study on the effects of data corruption in files of various file formats from the raster image domain.

[1] In the following, the term 'file format' is synonymous with 'data format'. Both terms refer to the same basic concept, whereas 'file format' and its instance, a file, additionally relate to the technical environment.

[2] In contrary, there are a considerable number of studies on the reliability of storage media in case of physical degradation which were examined under lab conditions, e.g. [7].

2 Approach

This research work is based on a study on the status quo of file format robustness. A controlled test corpus was built which comprises files of various file formats with different format characteristics. The files are the basis for data corruption experiments. A software tool which is able to damage files by flipping single bits (or bytes) was developed for that purpose. The damaged files are compared against the undamaged reference file, measuring the quantitative difference (i.e., the differences based on the pixel data) between both with metrics from statistical data analysis.

Conclusions are finally drawn on the measured values. A summary report on the study results is given in chapter 2.2.

2.1 Study Outline

2.1.1 Selection of File Formats and Test Corpus

The decision on which file formats to choose for the study was influenced by several criteria. On the one hand, those formats which have a factual relevance as an archival format for digital libraries and archives were considered, as well as including current trends (e.g., JPEG2000 as archival format). Second, the decision was also influenced by the (estimated) active usage of file formats in the real world, also including the commercial communities. Third, technical considerations were taken into account. There had to be sufficient support of the file format by software tools to successfully run the experiments for the study. Openness of the file format specification is also desirable in order to allow for an in-depth analysis of format features and structures. And fourth, the choice of file format was subject to theoretical considerations which are very strongly linked to content-related aspects of the research topic (e.g., strategies for improvement of robustness).

In consideration of all these criteria, the following file formats were selected: [3] Tagged Image File Format, version 6.0 (TIFF), JPEG File Interchange Format (JPG), JPEG 2000 File Format (JP2), Portable Network Graphics, version 1.2 (PNG), Graphics Interchange Format, version 89a (GIF), Windows Bitmap (BMP).

Based on these six formats, a set of test files was arranged in the form of a 'controlled' test corpus. It offers the advantage to precisely choose such format characteristics which are of special interest for robustness and hence are the best for analytical research such as this. [4] This controlled test corpus is also assumed to be representative, since the files' characteristics are in most cases common ones.[5]

An evaluation of the file formats shows that there are many aspects which are basically important for robustness, yet their actual significance is often marginal if looked at in isolation. For example: TIFF defines the characteristic 'Photometric Interpretation'. For monochrome and grey-scaled images it enables an application to interpret the pixel data in either a white-is-zero (i.e., bit value 0 has to be interpreted as white) or black-is-zero format. If the data which carries this information is corrupted, an

[3] The abbreviations in parentheses will be used in the following sections of this paper.

[4] In contrast, a random sample of files would be targeted to descriptive research but this is not the primary objective here.

[5] Compare Table 1 in which the files are described.

application may either not be able to process the file at all or may process the data in the contrary sense (given that the application has not implemented a default behaviour for such cases). In both cases the effect of data corruption concerning this characteristic is fatal. However, viewed from a strict probabilistic point, the probability that such an error occurs at all is (for low corruption rates) fairly small since this information is encoded using only 12 bytes which is infinitesimal in comparison to possible file sizes of a million bytes or more.

One of the factors which is suspected to have a great influence on robustness is compression of data. The danger of losing data during transmission over noisy channels is a well-known fact[6]. Especially in conjunction with data being compressed the consequences of bad data can potentially become extremely drastic. Therefore it is also assumed that this is similar for data serialized on some storage medium. Thus, file formats which heavily build on data compression are suspected of being less robust than those that do not. Although this may not really be a new or even surprising insight, it is not always beyond all question: JPEG2000 codec not only supports simple error detection but also, in a quite advanced way, correction of errors via several features defined in the JPEG2000 compression algorithm.[7] There should be at least gradual variations in the robustness for files with different compression algorithms. The compression feature was included in those files of the test corpus which underlying file format allows for usage of a specific compression algorithm.

Table 1 in the appendix gives an overview of those corpus test files which are discussed in more detail in chapter 2.2 (report on the results).

2.1.2 Experimental Design and Technical Environment

The study is based on experiments in which the files of the test corpus are processed along a chain of different software components. First they are damaged ('Corrupter'), then converted to an analysis format ('Converter') and finally analysed ('Analyser') by applying metrics which basically compare the pixel data from the undamaged and corrupted file. 'Damaged' means: Bits (or bytes) are not totally dropped but changed, i.e. the bits are flipped.[8]

The Corrupter is equipped with different operation modes. The main one is a general corruption procedure on the whole file where randomly[9] chosen bits are flipped according to an arbitrary percentage.[10] By doing so, the introduced errors are fairly equally distributed. This pattern of error distribution is most suitable for this research task, since it is the most neutral pattern that allows for universally valid conclusions regarding robustness. In reality, there is no error pattern that can be verified as the

[6] Some file formats, especially those which are designed for application in the World Wide Web, provide control mechanisms. PNG for example has defined checksums for each of its chunks to enable error control for the processing software.

[7] A description would be beyond the scope of this paper; please see [8] for details on the JPEG 2000 compression.

[8] Dropping data was not considered for the experiments since pre-tests revealed strong evidence that this is a knock-out criterion in most cases even for very small corruption rates.

[9] This is done by a random number generator which implements the Mersenne Twister Algorithm [9].

[10] This is the operation mode for the results reported here.

general pattern. The error pattern depends on a large number of factors like the nature of storage media, logic of storage, source responsible for damage and many others.[11]

The Converter is mainly a wrapper for third-party file format converting tools. The Windows bitmap format (BMP) was chosen as the analysis format for it fits the requirements of such an analysis format best. It is a simple structured format, easy to process, widely supported by conversion tools and still capable of describing the characteristics (basically the pixel data) we need for analysis of file format robustness.

Conversion to an analysis format is not the only option one could apply at this stage of the experiment. Either way, we need some form of re-processing of the corrupted files since the analysis builds on a comparison of the pixel data. Thus, e.g., in case of compressed data, decompressing the data is absolutely essential in order to be able to compare the pixel data properly.[12] So in this second stage of the experiment, every corrupted file is converted to uncompressed BMP.

In the last stage, the files are compared against the reference file which was also converted to BMP. The Analyser has implemented a number of metrics (see below) which work on the pixel data. The results of measurement are finally written to a file for further analysis.

Several pre-tests were conducted in order to optimize the experimental design. One of the critical steps is the conversion of the corrupted files to the chosen analysis format. It was supposed by experience that the third party software used for that may perform differently. Three open source conversion tools were chosen for conversion: ImageMagick and IrfanView for all file formats and additionally Kakadu (JP2 only).[13] A complete test cycle (applying two different corruption rates with a smaller trial) was run on every test file using the three tools successively. Based on the analysis of the metric results, the tool with the best performance was finally chosen for the experiment.

Another decision that had to be made was related to the choice of corruption rates. This choice had to be made individually since the file sizes of the test files differ and therefore some very low rates could not be performed on small files. Moreover, the rates were 'calibrated' according to the results of pre-tests since it would not be sensible to run large tests on successive corruption rates where the differences in the results are marginal (this was true for some of the files in either the very low or very high rates). As a result, the span of corruption rates applied on the files varies from exactly one bit[14] corrupted (this was done for all files) up to 0.1% (of the individual file size). Furthermore, in some cases the introduced errors caused serious problems (e.g., sudden crashes, buffer overruns, refused termination), especially for very high

[11] Just because of this broad variation in the factors possibly determining an error pattern, the random pattern introduced here is most suitable for general conclusions on robustness. It is the intention of this work to draw general conclusions on robustness, independently from determinants which may vary by and by.

[12] During the process of format conversion, data is read into memory and re-encoded to the chosen target format. As a consequence, the data is also completely reinterpreted by the application, including the corrupted data as well.

[13] See http://www.imagemagick.org,
http://www.irfanview.com/, and http://www.kakadusoftware.com/.

[14] This is equivalent to a 1-byte corruption in most cases, since most file formats use at least one byte for storage of information units.

corruption rates. In such cases the experiments for the given rate could not performed to its end as well.

Finally, the size of the samples had to be determined. This is a rule of thumb estimate which can be figured out by pre-tests. These led to the decision to set the size of the samples to 1000 per run, i.e. for each corruption rate the test files were corrupted, converted, and compared to the undamaged reference file a thousand times.. All files, the corrupted ones as well as the converted analysis files, were saved on hard-disk to enable secondary analysis.

2.1.3 Metrics

The Robustness Indicator (RI) is a simple metric that counts the number of different pixels of two image data sets and relates it to the total number of pixels:

$$\mathrm{RI} = \mathrm{D} / \mathrm{n} \tag{1}$$

where

D is the number of different pixels,

n is the total number of pixels.

Interpretation: The smaller the RI, the better the robustness. Range is from 0 to 1. If multiplied by 100, it shows the percentage of different pixel values.

For the overall comparison of file format robustness, the RI values for each single file-to-file comparison are summed up and finally averaged over the sample size:

$$\mathrm{RImean} = \sum RI\,/m \tag{2}$$

where

$\sum RI$ is the sum of all RI,

m is the sample size (usually 1000).

A very familiar image quality metric is the root mean squared error (RMSE). This was implemented as well. It is among a group of related metrics (e.g., peak-signal-to-noise ratio, mean squared error) which basically tell us the same facts in slightly different form.[15] RMSE is defined as:

$$\mathrm{RMSE} = \sqrt{\frac{\sum\left(Xi - Yi\right)^2}{n}} \tag{3}$$

where

Xi is a pixel-constituting value of the reference image data

Yi is a pixel-constituting value of the corrupted image data

n is the total number of pixel constituting values

Interpretation: The higher the RMSE, the worse the robustness. Range is from 0 to RMSEmax.[16]

[15] Therefore these metrics were not computed.

[16] RMSEmax is one of the auxiliary metrics we defined. It is the highest (worst) possible value for RMSE for the given image data set of the reference file.

Just as for RI, the mean over all single RMSE (RMSEmean) of a sample was computed.[17]

Besides these two core metrics for measurement of robustness, some further metrics which aim to provide cross-checks on the validity of the measured values for RImean and RMSEmean were included. These are: RI and RMSE based on the actual successful comparisons (called RImean- and RMSEmean-), median, standard deviation, skewness and confidence intervals (based on either the median or the mean, depending on the skewness of the distribution of the single values for RImean and RMSEmean).[18]

There are many different metrics which could be applied to measure robustness. For reasons of paper space, they can not be discussed in detail.[19] Nevertheless, there is a motivation for the choice of the primary metrics for this study. Besides the fact that the implementation of these metrics is easy and effective[20], it was mainly influenced by two further points:

First, it was intended to make a statement of 'hard facts': How many of the pixels have changed after data corruption, based on the data that encode its information? RI is perfectly suited for this. It actually tells us if the data that keeps the pixel's information is still undistorted or not.

RMSE has been introduced since it is also aimed at drawing conclusions on the quality of what is left after data corruption. In contrast to RI, which is based on boolean comparison, it is a metric of gradual changes within the pixels. This turns it into a metric more suitable for quality measurement. Nevertheless, this is only true by tendency. RMSE is, as all the other measurements of distortion, more or less suited for quality measurement [1] since its significance highly depends on the type of distortion produced by the corrupted data.

2.2 Results

Tables 2 and 3 in the appendix contain the results for RMSEmean and RImean. Results for RMSEmean- and RImean- as well as the other auxiliary metrics are not included in here since they did not have an impact on the validity of the core observations we make in the next section.

2.2.1 Observations

In general, the results for the robustness metrics differ for various file formats in a wide range, depending on the specific format characteristic introduced to the test file. Many of our test files responded to data corruption in a very sensitive way, especially those with compressed data in it. A single bit can be extremely significant for robustness: the corruption of just one single bit out of the entirety of a million bits proves to be destructive for information consistency.

[17] The formula is very similar to the RI modification (additionally RMSE is divided by m), therefore a representation of it is not included at this place.

[18] Since these metrics are auxiliary metrics for RI and RMSE and also very common in statistics, they are not described in detail.

[19] See [1] for a compilation of measures.

[20] This is especially important for a study like this one, where we deal with a huge amount of data.

Observation 1: Three of the files we described in table 1 do not make use of any data compression (bmp_A, tiff_A1, tiff_A2). Among these three, the robustness for bmp_1 is the best for both RImean and RMSEmean. Most notably for the levels of higher data corruption (starting from 0.05 onwards, not shown in the table), bmp_A outperforms the two TIFF files.

Observation 2: The two TIFF files we described in table 1 are almost identical in all of their characteristics, except in the way they arrange the pixel data within the file. File tiff_A1 groups the image data in one continuous byte sequence. The position of this sequence within the file is stored in exactly one small sequence of bytes (the offset) in the so-called TIFF header. If this essential sequence is corrupted, the entire file can not processed any more, since the application is not able to find the pixel data anymore. In contrary to tiff_A1, tiff_A2 contains 144 of these image data stripes, thus also 144 offsets which point to the stripes. This means at the same time, that file tiff_B contains 143 essential sequences of bytes more than file tiff_A1. Following the rules of probability, the lower robustness of tiff_A2 shown by our metrics is fairly evident.

Observation 3: All of the ten files containing compressed data show higher values for RImean (indicating less robustness) than those which do not comprise compressed data, over all levels of data corruption. For RMSEmean, these are eight out of ten. Only the JP2 files with fully enabled error resilience functionality (JP2_B, JP2_C) could achieve roughly the same results as the worst performing file (tiff_A2) out of the three files with uncompressed data.

Observation 4: The evaluation of the metrics results over all files which make use of compression must be regarded differently for RImean and RMSEmean. Regarding RMSEmean, the JP2 files are by far the best performing files whereas the data for RImean can not verify this. However, a secondary (visual) analysis of the corrupted files clearly show much better image quality levels than for those files with the other compression methods. Therefore the reflections on the applicability of distortion metrics in section 2.1.3 prove true.

Observation 5: Regarding the TIFF files: Uncompressed TIFFs (tiff_A1, tiff_A2) always perform better than TIFFs which make use of compression (tiff_B, tiff_C). For both RImean and RMSEmean, usage of JPEG compression appears to be better than ZIP compression (followed by LZW compression; this file is not included in table 1).

Observation 6: Regarding the JPEG files: JPEG compression appears to react quite differently to data corruption. Although jpeg_B is of higher compression (0.081) than jpeg_A (0.327) it shows much better robustness (based on the data for both RImean and RMSEmean). Robustness can additionally be improved if the JPEG compression feature 'progressive' is used.

Observation 7: Regarding the JP2 files: The file with lossless data compression (jp2_B) does not prove to be more robust than jp2_C which contains lossy compressed data. For RImean it performs even worse. Concerning error resilience: Especially the data for RMSEmean show the advantages of the error resilience features introduced in JPEG 2000 part 2. The files in which we included error resilience features (jp2_B, jp2_C) performed significantly better than the ones that do not include these features, especially for higher corruption levels.

Further observations: We also included JP2 test files[21] to the corpus in which the different resilience options are varied with each other and did several tests on them. One of the main findings is that the usage of SOP and EPH[22] markers alone showed only moderate improvement of robustness compared to files not using any of the error resilience features. Best robustness improvements can be achieved if all of the error resilience options are enabled. One further finding is that additional usage of smaller so called precincts, which in [3] is also considered to improve robustness, only results in marginal improvement of robustness.

Pre-tests related to the tools we used in the Converter module brought to light that there is perhaps an obstacle which needs to be overcome for acceptance of JPEG 2000 compression in the preservation context for the future: From the three different tools we tested for handling the JP2 files, only the one with which the files were also created (Kakadu) was able to deal with the error resilience features in such a way that robustness was actually improved.

2.2.2 Conclusions

Our assumption concerning the negative relation of data compression and robustness could be verified. The result data reveal better performance of files with uncompressed data for almost all tested files. Nevertheless, this is not always as obvious as it is still assumed. The findings in observation 2 and observation 3 show that JP2 can be quite competitive to file formats not using data compression feature if, for the latter, other features which potentially have an influence on robustness are not considered.[23]

Data compression is a crucial feature regarding robustness. However, it is by far not the only one. Observation 1 and 2 clearly revealed this fact. In the first case, BMP uncompressed showed better robustness than TIFF uncompressed. The reason for this lies in the fact that a standard BMP file is fairly near to raw data. It contains only few essential data[24], i.e., data the application necessarily needs in order to process the file. Thus the probability of corruption of such essential data compared to files of other file formats is quite low. Observation 2 is a good example for the effects of structural determinations in file formats. In this case, the image data is split in many parts (strips), whereas each part is referenced by an individual offset. Unfortunately, every single offset to the image data is essential in order to process the referenced data sequence in a regular way. If an offset is corrupted, the effect of this corruption is not restricted to the offset's bytes alone, all of the bytes which carry the information of the referenced data sequence are affected as well. Given this, a single corruption of an offset is momentous as such - but even more if there is not only one single offset but many (as it is for tiff_A2), since the rules of probability have an increasing effect.

This study demonstrates that there are indeed factors, originating in file format design and functionality, which have an influence on the robustness of files. Taking

[21] Not included in table 1.

[22] For information on error resilience and other JPEG 2000 features, see [8].

[23] Given this, the findings of tests in [3] where JP2 is supposed to be even more robust than TIFF uncompressed could not be verified. The image of the distorted TIFF shown in this report (Figure 6b) strongly let us suppose that the TIFF files used for these tests are multiple striped TIFFs (corresponding to tiff_A2, see observation 2) which prove to be significantly less robust than 1-striped TIFFs.

[24] This refers to the categorization of file format data in [6].

these factors more carefully in consideration should lead to improvement of saving digital information beyond the existing strategies (such as error correction on hardware level or distributed and redundant storage of files).

3 Outlook

Although it is believed that research in this area is very fundamental for the overall goal of the preservation task of digital libraries and the maintenance of our (digital) cultural heritage, very little research has been undertaken on this topic so far [3, 6]. Further research by the author is currently in progress, concentrating on file formats which mainly deal with text information. This will be a fruitful contribution especially with regard to the latest trends in the discussion on the appropriate file format for text and hybrid content. It will also be the groundwork for suggestions on the improvement of robustness on file format level, which will be made in a final step.[25]

References

1. Avcıbas, I., Sankur, B., Sayood, K.: Statistical evaluation of image quality measures. Journal of Electronic Imaging 11(2), 206–223 (2002)
2. Bairavasundaram, L.N., et al.: An Analysis of Data Corruption in the Storage Stack. ACM Transactions on Storage 4(3) (2008)
3. Buonora, P., Liberati, F.: A Format for Digital Preservation of Images: A Study on JPEG 2000 File Robustness. D-Lib Magazine 7/8 (2008), http://www.dlib.org/dlib/july08/buonora/07buonora.html (accessed May 2009)
4. Chapman, S., et al.: Page Image Compression for Mass Digitization. In: Archiving 2007. Final program and proceedings, pp. 37–42 (2007)
5. Gilesse, R., Rog, J., Verheusen, A.: Life Beyond uncompressed TIFF: Alternative File Formats for the Storage of Master Image Files. In: Archiving 2008. Final program and proceedings, pp. 41–46 (2008)
6. Heydegger, V.: Analysing the Impact of File Formats on Data Integrity. In: Archiving 2008. Final program and proceedings, pp. 50–55 (2008)
7. Iraci, J.: The Relative Stabilities of Optical Disk Formats. Restaurator 26(2) (2005)
8. ISO/IEC 15444-5:2003. JPEG 2000 image coding system (2003)
9. Matsumoto, M., Nishimura, T.: Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. ACM Trans. on Modeling and Computer Simulation 8(1), 3–30 (1998)
10. Panzer-Steindel, B.: Data Integrity, internal CERN/IT study (2007), http://indico.cern.ch/getFile.py/access?contribId=3&sessionId=0&resId=1&materialId=paper&confId=13797 (accessed May 2009)
11. Schroeder, B., Gibson, G.A.: Disk failures in the real world: What does an mttf of 1,000,000 hours mean to you? In: Proceedings of the 5th USENIX Conference on File and Storage Technologies, FAST (2007)

[25] This study reported in here is part of a larger project. All existing and future results will be published in the end of it. Please contact the author for any further questions.

Appendix

Table 1. Selection[26] of test corpus files with brief description of core characteristics. All images are of the same dimensions (498 x 719). The compression rate indicates the ratio of: file size / file size of uncompressed file.

format	name	characteristics
TIFF	tiff_A1	file size (byte): 1075526; image type: coloured; bits per pixel: 24 (8,8,8); compression: none; number of stripes: 1;
	tiff_A2	file size (byte): 1075682; image type: coloured; bits per pixel: 24 (8,8,8); compression: none; number of stripes: 144;
	tiff_B	file size (byte): 107697; image type: coloured; bits per pixel: 24 (8,8,8); compression: jpeg; compression rate: 0.101; number of stripes: 1;
	tiff_C	file size (byte): 253494; image type: greyscale; bits per pixel: 8; compression: ZIP; compression rate: 0.856; number of stripes: 1;
JPEG	jpeg_A	file size (byte): 352162; image type: coloured; bits per pixel: 24 (8,8,8); compression: jpeg; compression rate: 0.327;
	jpeg_B	file size (byte): 87010; image type: coloured; bits per pixel: 24 (8,8,8); compression: jpeg; compression rate: 0.081
	jpeg_C	file size (byte): 83133; image type: coloured; bits per pixel: 24 (8,8,8); compression: jpeg; mode: progressive; compression rate: 0.078;
JP2	jp2_A	file size (byte): 103048; image type: coloured; bits per pixel: 24 (8,8,8); compression: jpeg 2000; compression rate: 0.096; error resilience: no error resilience features included;
	jp2_B	file size (byte): 689227; image type: coloured; bits per pixel: 24 (8,8,8); compression: jpeg 2000; compression rate: 0.641; features: all error resilience features included[27], lossless compression;
	jp2_C	file size (byte): 103048; image type: coloured; bits per pixel: 24 (8,8,8); compression: jpeg 2000; compression rate: 0.096; features: all error resilience features included, lossy compression
PNG	png_A	file size (byte): 700847; image type: coloured; bits per pixel: 24 (8,8,8); compression: ZIP; compression rate: 0.652
GIF	gif_A	file size (byte): 207.637; image type: coloured; bits per pixel: 8; compression: LZW; compression rate: 0.575
BMP	bmp_A	file size (byte): 1.075.678; dimensions: 498 x 719; image type: coloured; bits per pixel: 24 (8,8,8); compression: none

[26] The test corpus contains 43 files in total, for reason of space, the table only contains those files which are discussed in more detail.

[28] Please see [8] for a description of the error resilience features. We used all of the features which are provided by the creation software Kakadu.

Table 2. Result data for metric RImean

	1 bit	0.0005	0.0010	0.0025	0.0050	0.0075	0.0100
tiff_A1	0.0000	0.0021	0.0072	0.0148	0.0222	0.0397	0.0463
tiff_A2	0.0017	0.0134	0.0296	0.0734	0.1340	0.1741	0.2413
tiff_B	0.2123	0.5307	0.7170	0.8507	0.9127	0.9379	0.9533
tiff_C	0.1598	0.7127	0.8434	0.9384	0.9695	0.9784	0.9882
jpeg_A	0.1886	0.8106	0.8927	0.9567	0.9790	0.9861	0.9903
jpeg_B	0.1658	0.4147	0.6335	0.8079	0.8929	0.9280	0.9466
jpeg_C	0.1978	0.4422	0.6667	0.8135	0.8857	0.9185	0.9362
jp2_A	0.2245	0.6435	0.8529	0.9838	0.9973	0.9987	0.9989
jp2_B	0.0401	0.7242	0.8910	0.9778	0.9898	0.9932	0.9944
jp2_C	0.1595	0.4593	0.6884	0.9034	0.9608	0.9761	0.9823
png_A	0.1680	0.9033	0.9539	0.9779	0.9915	0.9949	0.9962
gif_A	0.1700	0.5367	0.6500	0.7549	0.8180	0.8407	0.8582
bmp_A	0.0000	0.0011	0.0022	0.0041	0.0062	0.0109	0.0164

Table 3. Result data for RMSEmean. Please note: By tendency, result values from 100.0 onwards also go along with very poor visual image quality.

	1 bit	0.0005	0.001	0.0025	0.005	0.0075	0.01
tiff_A1	0.0308	0.6099	1.6920	3.2719	4.7602	8.0671	9.4261
tiff_A2	0.2790	2.5138	5.1467	12.4700	22.3199	29.4321	39.4085
tiff_B	14.1767	40.2629	59.7920	82.3622	96.8800	102.9344	107.0307
tiff_C	19.9272	74.3268	87.2172	101.3937	109.1695	113.6602	116.4384
jpeg_A	19.7448	94.3614	102.8009	106.8254	107.1547	109.3841	113.1354
jpeg_B	13.6913	35.8373	63.1653	89.7461	102.0462	108.1198	111.3786
jpeg_C	7.4957	17.8646	32.4221	54.0923	81.5385	98.3300	108.5839
jp2_A	1.9187	6.8776	10.4855	19.7557	31.4201	40.9756	48.0054
jp2_B	0.2605	4.8834	7.6121	15.7449	21.4953	32.9461	38.0008
jp2_C	1.0732	4.5185	8.7373	12.2292	20.8362	29.7704	35.1526
png_A	31.1893	132.2457	137.7804	139.0441	142.4394	143.4338	144.0673
gif_A	32.6047	83.0106	93.2021	102.1034	108.3719	112.9389	116.7700
bmp_A	0.0300	0.5241	0.6677	1.2994	1.9721	2.7480	3.9217