An Analysis of Contemporary JPEG2000 Codecs for Image Format Migration

William Palmer
British Library
96 Euston Road
London, United Kingdom
william.palmer@bl.uk

Peter May
British Library
96 Euston Road
London, United Kingdom
peter.may@bl.uk

Peter Cliff
British Library
96 Euston Road
London, United Kingdom
peter.cliff@bl.uk

ABSTRACT

This paper presents results of an analysis of different implementations of the JPEG2000 standard, specifically part 1: JP2, an image format that is currently popular within the digital preservation community. In particular we are interested in the effect different JPEG2000 codecs (encoders and decoders) have on image quality in response to lossy compression. We focus on three main codec libraries for analysis - Kakadu, JasPer and OpenJPEG - migrating 932 TIFF newspaper images to lossy JPEG2000 files using 2:1 and 4:1 compression ratios, and monitor image quality using PSNR. We look at the combination of encoder/decoder pairs and find that using OpenJPEG for both gives the best quality results, albeit with the slowest execution time. We also find that in some circumstances, particularly when a JasPer encoder is used, in order to retain image quality of the decoded image, the best choice of decoder may not be the same codec used to create the JPEG2000; based on these results, the encoding library is therefore recommended technical preservation metadata to retain.

Keywords

JPEG2000, TIFF, migration, codec, PSNR, image quality, generational loss

1. INTRODUCTION

The British Library, as a memory institution, holds large quantities of digital content, including over 2 million files produced from the JISC funded British Newspaper digitisation projects [1]. The number of files within our digital collections is ever increasing and these need to be cost-effectively preserved to ensure long-term access to these images.

Storing this collection as TIFF files would require a significant amount of storage just to preserve the masters alone. JPEG2000 presents an alternative file format for digital images, that has a number of advantages for preservation, such as reduced storage costs compared with the traditional TIFF master files and the ability to contain both master and lower-resolution access copies within a single file (compared with TIFF masters which typically require a separate access copy - PNG or JPEG - to also be kept) [2].

JPEG2000 files can be compressed in either of two ways; losslessly, where a bit-identical copy of the data is maintained and can be retrieved, and lossy, where an exact copy of the data is not maintained and some fidelity is lost. In general, lossy compression results in smaller image files, but will suffer from information loss resulting in image artefacts such as blurring, as is especially the case with high compression ratios.

Putting aside the merits of preserving image files using lossy compression, what is unclear is whether all JPEG2000 codecs perform to the same quality. Do all codecs produce the same quality of file given the same settings, or are there consistent variations between them? Ebrahimi et. al. [3] considered the effects of compression ratios on perceived image quality using three JPEG2000 codecs (JasPer, Kakadu and IrfanView). Their results suggested JasPer performed the better of the three (it is unclear if this result is statistically significant), however this operated on a small sample (29) of small files (768x512 pixels) from the LIVE Image Quality database [4], using a large range of compression ratios (from 2:1 - 300:1), and focussed primarily on the metric for determining image quality rather than tool performance itself (quality of result, speed, etc.). In contrast, a sample of the JPEG2000 files from our newspaper collection were images of sizes 4672x5944 pixels and 8320x9568 pixels, with applied compression ratios at the low end of the spectrum, around 2:1 (although, as supported by JPEG2000, a number of compression levels have been defined to provide numerous quality-level images within the one file, for example, for access).

If we consider the preservation process for migrating an image to JPEG2000 and then accessing it, even for quality assurance purposes, then two distinct uses of the codec are required. The original image must first be encoded using a JPEG2000 codec to create a JP2 file, then accessed through a decoding step with a JPEG2000 codec. However the codec used for these two steps does not have to be the same.

From a preservation perspective, it would therefore be useful to understand the effects of various codecs, combinations of encoding/decoding codecs, and codec settings on the resulting files, thus providing evidence to enable appropriate tool selection within a migration workflow. This paper presents our initial analysis results of such codec use for a single migration, before looking at an extension considering the effects of lossy migration on subsequent migrations.

To put the extension into context, we expect to provide access to our files for many hundreds of years into the future. Although Gollins [5] promotes a parsimonious "rule-of-thumb" approach to preservation, "using only the minimum necessary intervention to secure our digital heritage for the next generation", no guarantees can be made about the absence of future migrations. Of concern when migrating files between lossy formats is digital generational loss - the increasing loss of information with each migration. Although it is fairly logical to conclude that lossy compression, having resulted in information loss, will cause ever increasing degradation in subsequent lossy migrations, to what extent will this degradation be? How quickly will it degrade? And is the amount of degradation affected by codec choice?

This paper starts by looking at the JPEG2000 codec libraries available, mentioning details of profiles, specifically with respect to compression rates, and indicating choices made for the experimental work carried out. Section 3 details the methodology of our single-migration codec analysis, as well as the generational loss extension. Results are presented in Section 4, with general conclusions and ideas for future work presented in Section 5.

2. USE OF JPEG2000

Whilst the British Library and other memory institutions [2] now utilise JPEG2000 files for preservation, their utilisation outside of these organisations would appear low. As an indication of the mainstream use of JP2 files, a recent search for "image/jp2" content type over the UK Web Domain Dataset Format Profile (1996-2010)[6] found only 53 files with some identification as "image/jp2". To put this further into context, a similar search for "image/jpeg" returned over 153 million files.

This apparently low uptake in the wider world may, in itself, present a preservation risk through lack of "high-quality" tool support [7]. Irrespective of whether this is an issue or not, there are several commercial and open-source libraries currently available; the question this paper starts to address is how do these compare?

2.1 JPEG2000 Libraries

There are several JPEG2000 codec libraries currently available, including:

- Kakadu[1], last updated January 2013. Commercial.
- OpenJPEG[2], last updated November 2012. BSD 3-Clause license.
- JasPer[3][8], last updated January 2007. License based on MIT license.
- JJ2000[4], last modified date is November 2009 (note: the actual last modification is possibly much before that). License unclear.
- FFMPEG[5], last updated December 2012. (L)GPL license.
- Other commercial codecs: Aware, LuraTech, LeadTools & J2K Codec

[1]www.kakadusoftware.com

[2]www.openjpeg.org

[3]www.ece.uvic.ca/˜frodo/jasper/

[4]code.google.com/p/jj2000/

[5]git.videolan.org/?p=ffmpeg.git;a=blob;f=libavcodec/j2kenc.c

For this analysis we chose to focus on the first three tools: Kakadu because it appears to be the codec of choice for large institutions; OpenJPEG as it is an open source codec that is actively maintained; and JasPer as it is the JPEG2000 codec widely used by other open source projects.

Conversely, we decided not to use JJ2000 due to its lack of recent development activity and mainstream use; FFMPEG for similar reasons; nor other commercial tools as we had no readily-available access. These codecs are in consideration for future research (see Section 5.1).

2.2 Profiles

A profile specifies desired image properties, such as compression (reversible or irreversible) and quality layers (with associated compression rates). From this, appropriate control settings to be used by a codec to create a JPEG2000 image can be derived. Significant investment seems to have been placed in trying to identify an appropriate level of compression whilst maintaining an acceptable quality of archival and production masters. Techniques mentioned in [2] formed around using either objective judgements from human observers to determine a visually lossless compression ratio, or by taking a minimal loss approach through compression with a maximum bit rate[6], i.e. all data is retained but there is minimal loss through rounding errors introduced by floating point transforms and from quantization. These approaches have resulted in similar findings regarding appropriate levels of compression, however it should be noted that the necessary compression settings needed to achieve visually lossless images is dependent on the images themselves [2].

The National Digital Newspaper Program (NDPD), for example, decided on an 8:1 compression ratio for JPEG 2000 production masters[7] as a compromise between file size and image quality, although 4:1 and 6:1 were judged to be visually lossless [9].

The Wellcome Trust's digital library use an iterative approach to determining compression rates across a collection (increasing compression until artefacts are observed, then stepping back), but have found a 10:1 compression ratio works well for books and 8:1 for archive collection material [2].

The British Library's recommended JPEG2000 profile for use in mass digitisation projects, in particular for our newspaper digitisation, is detailed in [10]. This specifies 12 quality layers, with compression levels starting at a minimally lossy rate. Whilst Kakadu and OpenJPEG can encode images according to this profile, JasPer cannot as it cannot use different precinct sizes. Additionally, there seems to be a bug in OpenJPEG v2.0.0 when coder bypass is used[8].

To make a comparative analysis of the codecs we chose a profile that could be encoded by all three chosen coders based on the British Library's recommended profile [10]. This is shown in Table 1.

2.3 Automated Codec Comparison

Much research has been done on Image Quality Algorithms (IQA) for measuring visible image quality [3, 11], however as Buckley [2] notes, currently "there is simply no

[6]Compressed bit rate is the ratio of compressed image data size to the image width and height[2]

[7]NDPD use uncompressed TIFF files for preservation masters.

[8]code.google.com/p/openjpeg/issues/detail?id=209

Table 1: **Test JPEG2000 profile**

File format	JP2
Transformation	9-7 irreversible (lossy)
Progression order	RPCL
Tiling	none
Levels	6
Precinct size	all 256x256
Quality layers	one
Code block size	64x64
Coder bypass	no

substitute for a human observer". Despite this, for the large collections held by us, human observation over the entire collection is simply not practical; more automated means are required.

One such IQA is the peak signal-to-noise ratio (PSNR) which gives a numerical value of the errors introduced by a lossy image encode, on a logarithmic scale, measured in decibels. A higher value indicates a better ratio of signal-to-noise, and provides some indication as to the quality of the image.

As a metric, PSNR is considered not to match well to perceived visual quality [12]. As part of our investigation we also tested use of a tool that calculated SSIM (Structural Similarity) as a metric for image file format migration quality. We found that SSIM was less sensitive to changes in the image than PSNR and was good for correlating *similar* images, such as resized images or those with added noise. However, the runtime of that tool was longer than ImageMagick's PSNR comparison. From our analysis the better metric of the two for image *identicalness* (as would be expected from a migration) was PSNR as it was faster and more sensitive to small changes in an image, thus giving greater assurance of success. We therefore opted to use PSNR as metric of image quality for this work.

3. METHODOLOGY

Migrating a TIFF to JPEG2000 and then viewing (or performing image analysis on) the resulting file requires both an encode and decode step. An assumption is typically made that the same codec should be used for both, but that does not need to be the case. The base experiment compares the 9 different combinations of encoder-decoder pairings possible with three libraries (Kakadu 7.1, OpenJPEG 2.0 and JasPer 1.900.1-13), see Table 4. This work is then extended to consider the effects of generational loss, i.e. how multiple migrations (encode-decode cycles) affect image quality.

3.1 Dataset

The input dataset used was 932 greyscale TIFF original masters from the British Library's JISC1 newspaper collection, totalling 26GB. Images from this sample averaged 51.0 megapixels, with a minimum of 21.1 megapixels and maximum of 93.4 megapixels.

3.2 Data Preparation

JasPer cannot take TIFF as an input format, therefore to make the migration experiments fairer, the TIFF input files are first converted to PNM (Portable Any Map) files using ImageMagick's *tifftopnm*, version 6.6.9.7-5ubuntu3.2.

3.3 Comparison Approach

A program was written to generate shell scripts that performed the following steps on a PNM file, for each encoder-decoder pairing. For each encoder, the appropriate command line listed in Section 3.4 was used to obtain JPEG2000 images meeting the desired profile. The steps are:

1. **Migrate:** Use the specified encoder and decoder to convert the PNM to a JP2 and then back to a PNM file (this is repeated as necessary for generational loss test);
2. **Validate Profile:** Extract information from the JP2 file using Jpylyzer, for later validation against the desired profile (specified in Section 2.2);
3. **Calculate PSNR:** Use ImageMagick to calculate the PSNR of the original TIFF file versus the migrated output PNM file (this comparison with the original is repeated after each migration for generational loss analysis), storing results in a CSV file;
4. **Consolidate Results:** Create a zip file that collects outputs from the above

This program was wrapped in a Hadoop MapReduce program so that, for each input TIFF file, firstly the tool is executed to generate the necessary shell scripts, and then these generated scripts are executed. A separate program was produced that extracted information from all the run outputs and produced aggregated CSV files.

Using a compression ratio of 2:1, runs were made consisting of ten generations of encode-decodes for each encoder-decoder pair, with, as per the methodology above, PSNR calculated between the original file and the output PNM after each generation. For these runs, it was found JasPer did not use all the space available to it - its compressed files were consistently more compressed than the requested compression ratio. Consequently, to enable all three encoders to produce output files of the same size and compression ratio, a further run was made using a 4:1 compression ratio. Nearly all files for each coder were within a few bytes of each other - see Table 3 - however, for some files JasPer still over-compressed them.

3.4 Encoder Command Line Parameters

The command line parameters which generate images conforming to the profile specified in Section 2.2, for each codec library, are shown in Table 2.

Table 2: **JPEG2000 command lines for each library**

Kakadu	Creversible=no Corder=RPCL Clevels=6 Cprecincts={256,256} Cblk={64,64}
OpenJPEG	-I -p RPCL -n 7 -c [256,256],[256,256],[256,256],[256,256], [256,256],[256,256],[256,256] -b 64,64
JasPer	-T jp2 -O mode=real -O prg=rpcl -O numrlvls=7 -O prcwidth=256 -O prcheight=256 -O cblkwidth=64 -O cblkheight=64

Note that the number of levels requested differs between Kakadu and the other tools, however, analysis of the outputs using Jpylyzer[13] shows these results to be equivalent. This discrepancy could be due to codec authors' different interpretations of the specification.

4. RESULTS

4.1 Exact Re-generation of Compressed Files

On each test run the JPEG2000 files were recreated (encoded) three times by each encoder. Each set of encodes produced identical files according to their MD5 checksum, suggesting that there is no variation produced by each library during encoding.

4.2 Compression Ratio

Kakadu and OpenJPEG's encoders routinely meet the compression ratio asked of them. There was a difference with the JasPer encoder, in that if it did not need all the space afforded by the specified compression ratio, it encoded at a higher compression ratio, producing a smaller file. This was also expected of the other codecs at lower compression ratios but was not apparent, from our results, at 2:1 compression.

4.3 File size

At 2:1 compression, the mean file size difference between the Kakadu and OpenJPEG encoders was 0.04% ±0.03% (7467 bytes ±5938 bytes). As already noted, JasPer encoded smaller files than the requested compression ratio would entail.

At 4:1 JasPer compressed files that were closer to the other encoders, see Table 3. However, as can be seen in Figure 2, at a requested 4:1 compression, JasPer was not always able to fully utilise the compression ratio.

Table 3: Mean difference between compressed file sizes at 4:1 compression for encoder-decoder pairs

Kakadu - OpenJPEG	OpenJPEG - JasPer	Kakadu - JasPer
186 ±106 bytes (0.002% ±0.001%)	747093 ±1102190 bytes (5.4% ±8.3%)	747277 ±1102201 bytes (5.4% ±8.3%)

4.4 Single Migration Image Quality

The results from the first encode-decode cycle for each encoder-decoder pairing are shown in Figure 1 and Figure 2, with the corresponding mean average PSNR (and standard deviation) shown in Table 4.

Table 4: Mean average PSNR for each encoder-decoder pair at 2:1 and 4:1 Compression rates

Encoder-Decoder	Mean Average PSNR (2d.p)	
	2:1 rate	4:1 rate
jasper-jasper	47.81 ±0.50dB	46.96 ±1.54dB
jasper-opj20	49.56 ±0.32dB	47.78 ±2.12dB
jasper-kakadu	49.45 ±0.32dB	47.69 ±2.08dB
kakadu-jasper	50.34 ±0.35dB	47.20 ±2.03dB
kakadu-opj20	54.17 ±0.41dB	48.12 ±2.44dB
kakadu-kakadu	54.22 ±0.43dB	48.13 ±2.45dB
opj20-jasper	52.62 ±0.37dB	48.23 ±2.45dB
opj20-opj20	**55.02 ±0.52dB**	**48.30 ±2.51dB**
opj20-kakadu	54.58 ±0.48dB	48.23 ±2.48dB

They show the OpenJPEG encoder with OpenJPEG decoder to produce the highest average PSNR for both 2:1 and 4:1 compression rates, at 55.02dB (2d.p.) and 48.30dB (2d.p.) respectively. The next highest, with a slight drop in PSNR, is the OpenJPEG-Kakadu pairing (54.58dB and 48.23dB for 2:1 and 4:1 respectively), followed by the Kakadu encoder with either using OpenJPEG or Kakadu for decoding (both showing 54.2dB and 48.1dB to 1d.p. for 2:1 and 4:1 respectively).

Our results indicate that files encoded with JasPer and decoded using any of the tools tend to result in a lower PSNR ($< 50dB$ for 2:1 and $< 48dB$ for 4:1) than when using other libraries for encoding. Using OpenJPEG or Kakadu as the decoder for such encoded files gives slightly better average PSNR results than using JasPer as the decoder (approx 49.5dB as opposed to 47.8dB for 2:1 compression; approx 47.7dB as opposed to 47dB for 4:1 compression).

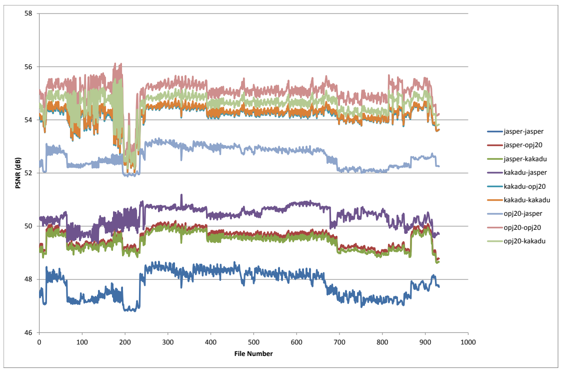

Figure 1: PSNR for first encode-decode for each encoder-decoder pair at 2:1 compression

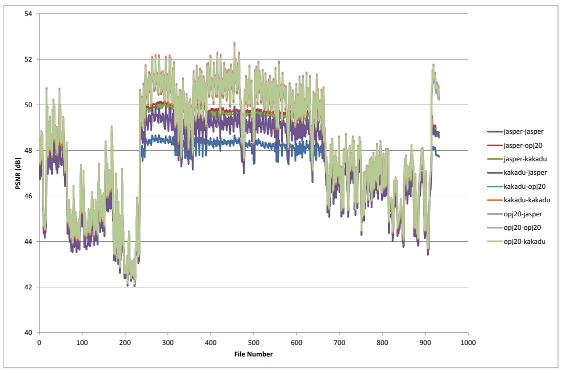

Figure 2: PSNR for first encode-decode for each encoder-decoder pair at 4:1 compression

A statistical analysis of the mean average PSNRs for each encoder-decoder pair (at 2:1 compression) showed that there is a statistically significant difference in the mean average PSNR, at 0.05 level, between all combinations of encoder-decoder pairs, apart from between Kakadu-OpenJPEG and Kakadu-Kakadu (the orange and hidden blue line 3rd and 4th from the top in Figure 1). This is congruent with the results in Table 4, which show the mean average PSNRs for these two pairings to be almost identical.

The difference in the PSNR means between the JasPer-OpenJPEG and JasPer-Kakadu pairs (green and red lines 2nd and 3rd from the bottom in Figure 1) was only just statistically significant. Again, this is reflected in the closeness of mean PSNRs seen in Table 4.

For the 4:1 compression ratio, shown in Figure 2, the statistical analysis of the difference in average PSNR between each encoder-decoder pair showed that there is a statistical significance, at 0.05 level, between JasPer-JasPer (blue line) and all other encoder-decoder pairs, apart from Kakadu-JasPer (purple line). This corresponds to a 0.24dB difference in mean average PSNR, and so the lack of statistical significance is unsurprising. Similarly, there is a statistical significance in the mean PSNR between Kakadu-JasPer (purple line) and all other encoder-decoder pairs, apart from JasPer-JasPer.

For the remaining differences between pairs, the JasPer-OpenJPEG or JasPer-Kakadu pairings compared against any other combination typically showed low levels of significance (some showed no significance, for example JasPer-OpenJPEG vs Kakadu -Kakadu), reflective of the small differences in mean PSNR showed in Table 4. All other combinations have even lower differences in mean PSNRs which are not statistically significant.

4.5 Generational loss

Ten encode-decode cycles were run for each file with each encoder-decoder pair. This was to test how the encoder-decoder pairs coped with generational loss through repeated migrations of lossy-encoded images.

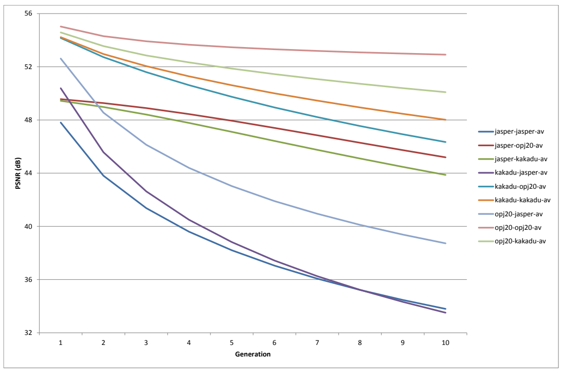

Figure 3: PSNR showing ten generations of 2:1 encode-decode for each encoder-decoder pair

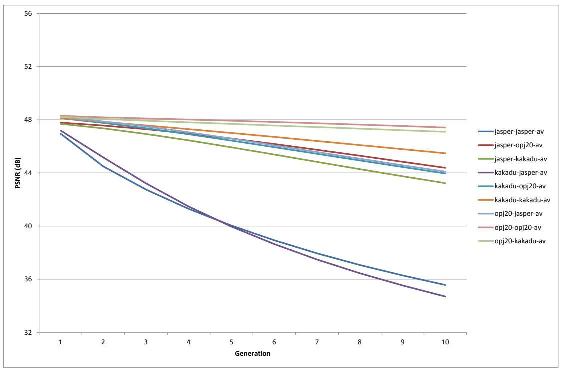

Figure 4: PSNR showing ten generations of 4:1 encode-decode for each encoder-decoder pair

Figures 3 and 4 show the mean average PSNRs after each generation, for 10 encode-decode cycles. The results are the average across all 932 files.

For both 2:1 and 4:1 compression ratios, all libraries show some signs of PSNR degradation, however some codecs suffer less than others. The OpenJPEG-OpenJPEG pairing is consistently the best (for both compression ratios) with only an average drop of 2.11dB (2d.p.) in PSNR for 2:1 compression and 0.88dB (2d.p.) for 4:1, over 10 generations.

At 2:1 compression, using JasPer as a decoder in combination with any other encoder results in the largest drops in PSNR - at least 13.89dB (2d.p.; OpenJPEG-JasPer). In particular though, the Kakadu-JasPer and JasPer-JasPer pairings both show the largest drops in PSNR for both compression rates.

4.6 Pixel differences vs original

As another means to gauge image quality, we used ImageMagick to count the number of pixels changed between the original image and the encode-decoded images. At 2:1 compression the maximum number of pixels that were more than 1% different to the original pixels, for Kakadu and OpenJPEG encoded files using Kakadu and OpenJPEG decoders, was 549. The average total number of pixels with an absolute difference to the original was between 20-24% of image pixels. JasPer figures are not quoted because, as previously described, at 2:1 compression JasPer tends to overcompress.

At 4:1 compression the percentage differences between encoder-decoder pairs are close. The overall mean of the average differences for each codec pair suggests approximately 2% of pixels are greater than 1% different to those in the original image. The same calculation for absolute differences in pixels gives an average of 55% of pixels being different.

4.7 Execution speed

A record was kept of execution speed for the generational loss encode-decode process, measuring the initial conversion from TIFF to PNM, using ImageMagick's *tifftopnm*, followed by ten encode-decode cycles. Files were decoded by the decoder directly to PNM, ready for the next encode.

Table 5 shows the average speed per encode-decode cycle for each encoder-decoder pair. Using Kakadu to encode and decode images is the quickest, using OpenJPEG is the slowest. Interestingly, using OpenJPEG as the encoder and/or decoder tends to result in longer encode-decode cycles.

Table 5: Mean execution speed of encoder-decoder pairs, per encode-decode cycle

Encoder-Decoder	Speed (s) at 2:1	Speed (s) at 4:1
Kakadu-Kakadu	16	12
Kakadu-JasPer	24	23
JasPer-Kakadu	25	24
JasPer-JasPer	32	32
Kakadu-OpenJPEG	44	34
JasPer-OpenJPEG	48	46
OpenJPEG-Kakadu	75	68
OpenJPEG-JasPer	78	78
OpenJPEG-OpenJPEG	103	90

5. CONCLUSION

1. For our test images and codec settings, the best quality encoder-decoder pair was OpenJPEG-OpenJPEG. The next best, where a different encoder was used, was the Kakadu-Kakadu pair, at approximately 1dB lower than OpenJPEG-OpenJPEG at 2:1 compression. It is worth noting that the execution speed of those pairs is both the fastest (Kakadu-Kakadu) and slowest (OpenJPEG-OpenJPEG). The file size difference between these encoders at 2:1 and 4:1 was low.

2. It appears that when a specific compression ratio is requested of the JasPer encoder, it compresses up to that ratio. If the encoder is unable to use all the space afforded to it by the compression ratio, the resulting file will have a higher compression ratio. This is readily apparent in the 2:1 compression test, and apparent in the 4:1 compression test.
3. From the results of the first encode-decode cycle for the codec pairs, we see that the JasPer decoder does not produce as high quality output as other decoders, given the same input file. Our results show the JasPer decoder was up to approximately 4dB worse quality than other decoders (e.g. Kakadu-Kakadu vs Kakadu-JasPer at 2:1 compression).
4. The first encode-decode at a higher compression ratio of 4:1, where the JasPer encoder was able to compress to the requested ratio, produced files much closer to the requested ratio than at 2:1. Results for these cases showed that only the JasPer-JasPer and Kakadu-JasPer pairs performed notably worse than other pairs. The average PSNR values for those two pairs were statistically significant from the other encoder-decoder pairs.
5. The lower quality output of the JasPer decoder became apparent in the generational loss tests, where use of a JasPer decoder, especially in conjunction with a JasPer or Kakadu encoder, produced notably larger drops in PSNR. When Kakadu or OpenJPEG decoders were used, the PSNR drop was less severe; they produced better quality decoded images from the same compressed JP2.
6. The JasPer decoder is unable to decode any of the test JP2 files to the same quality as the other decoders, including JPEG2000 files encoded by its own encoder. If quality of decoded files is important it may be advisable to decode JPEG2000 images with a decoder known to be good for the encoder rather than using an unknown/lower quality built-in decoder. For digital preservation, this indicates an importance in understanding the library used to create the JPEG2000 file.

5.1 Future work

There are many potential avenues to extend this work, but some notable areas we feel warrant further work include:

- Repeating the testing with different collections, for example; photographs
- Using different codecs and settings, for example; using Kakadu's *-precise* flag to increase precision
- Using different encoding profiles; tiles vs precincts, compression ratios, compression layers
- Using different quality metric(s)

6. ACKNOWLEDGEMENTS

This work was partially supported by the SCAPE project (www.scape-project.eu). The SCAPE project is co-funded by the European Union under FP7 ICT-2009.4.1 (Grant Agreement number 270137).

7. REFERENCES

[1] *The JISC Digitisation Programme: Overview of Projects.* URL: `http://www.jisc.ac.uk/media/documents/programmes/digitisation/digitisation_v2_overview_final.pdf` (Last accessed: 04/26/2013).

[2] Robert Buckley. *Using Lossy JPEG 2000 Compression for Archival Master Files.* Library of Congress, Mar. 12, 2013. URL: `http://www.digitizationguidelines.gov/still-image/documents/JP2LossyCompression.pdf` (Last accessed: 04/25/2013).

[3] Farzad Ebrahimi, Matthieu Chamik, and Stefan Winkler. "JPEG vs. JPEG 2000: an objective comparison of image encoding quality". In: *Proceedings of SPIE.* Vol. 5558. 2004, pp. 300–308. URL: `http://stefan.winklerbros.net/Publications/adip2004.pdf` (Last accessed: 04/26/2013).

[4] H. R. Sheikh et al. *LIVE image quality assessment database (2003).* 2003. URL: `http://live.ece.utexas.edu/research/quality/` (Last accessed: 04/26/2013).

[5] Tim Gollins. "Parsimonious preservation: preventing pointless processes!" In: *Online Information.* 2009, pp. 75–78. URL: `http://www.nationalarchives.gov.uk/documents/parsimonious-preservation.pdf` (Last accessed: 04/26/2013).

[6] *Format Profile JISC UK Web Domain Dataset (1996-2010).* DOI: `10.5259/ukwa.ds.2/fmt/1`.

[7] *Is JPEG-2000 a Preservation Risk?* Jan. 28, 2013. URL: `http://blogs.loc.gov/digitalpreservation/2013/01/is-jpeg-2000-a-preservation-risk/` (Last accessed: 04/24/2013).

[8] M.D. Adams and F. Kossentini. "JasPer: a software-based JPEG-2000 codec implementation". In: *Image Processing, 2000. Proceedings. 2000 International Conference on.* Vol. 2. 2000, 53–56 vol.2. DOI: `10.1109/ICIP.2000.899223`.

[9] Robert Buckley and Roger Sam. *JPEG 2000 Profile for the National Digital Newspaper Program.* Apr. 27, 2006. URL: `http://www.loc.gov/ndnp/guidelines/docs/NDNP_JP2HistNewsProfile.pdf` (Last accessed: 04/25/2013).

[10] *The British Library JPEG 2000 Profile for Bulk Digitisation.* URL: `http://www.digitizationguidelines.gov/still-image/documents/Martin.pdf` (Last accessed: 04/26/2013).

[11] Thien Phan, Phong Vu, and Damon M. Chandler. "On the Use of Image Quality Estimators for Improved JPEG2000 coding". In: Asilomar Conference on Signals, Systems, and Computers (2012). 2012. URL: `http://vision.okstate.edu/pubs/asilomar_2012.pdf` (Last accessed: 04/26/2013).

[12] Zhou Wang et al. "Image Quality Assessment: From Error Visibility to Structural Similarity". In: *IEEE Transactions on Image Processing* 13.4 (Apr. 2004). URL: `https://ece.uwaterloo.ca/~z70wang/publications/ssim.pdf` (Last accessed: 04/26/2013).

[13] David Tarrant and Johan Van Der Knijff. "Jpylyzer: Analysing JP2000 files with a community supported tool". 2012. URL: `http://eprints.soton.ac.uk/341992/`.