https://doi.org/10.2352/issn.2168-3204.2023.20.1.34

Benchmarking Lossless Still Image Codecs: Perspectives on Selected Compression Standards From 1992 Through 2022

Michael J. Bennett; University of Connecticut Library; Storrs, CT/US

Abstract

As complementary technologies evolve, data compression continues to be a foundational aspect of growing digital collections. In this study, selected lossless still image codecs from 1992 through 2022 were benchmarked across a variety of efficiency and performance measures using reference images from cultural heritage. Additionally, entropy estimates were calculated by source image to assist in characterizing image information and evaluating encoder efficiency against assessed feasible compression limits. Encoder designs and compression techniques were also examined in the context of the study's measured results.

Background

Still image compression has witnessed continued advances in new codecs and standards during the past three decades. This progress has come in response to marked improvements in sensor technologies and pixel densities along with new computational requirements of 3D, multispectral, and high-dynamic-range imaging [1]. In addition, state-of-the-industry display devices and software are today both able to more fully render and exploit such rich source data for a growing assortment of end user tasks.

Modern image data acquisition and post-processing workflows, as a result, all add to the needed capacities of storage and transmission. Notably lossless compression techniques serve an important role within imaging applications such as medical, aerial, satellite, deep space exploration and cultural heritage archiving where it is often critical that original rendered pixel data can be comprehensively recovered upon decompression [2]–[4].

The final years of the 2010's witnessed several new standards that each promised improvements in comparison to past alternatives [5]–[7]. This study aims to benchmark a selection of both the latest lossless still image codecs and older examples that have been released through time to better assess overall trends in compression development.

Methods Overview

The following selected still image codecs (Figure 1), each configured for mathematically lossless encoding, were evaluated across a range of reference test images.

Acronym	Name	ISO Standard	Initial Release
JPEG-1	JPEG Original Lossless Compression	ISO/IEC 10918	1992
JPEG-LS	Improved JPEG Lossless Compression	ISO/IEC 14495-1	1999
JPEG 2000	JPEG 2000 Part 1, Core Coding, Lossless Compression	ISO/IEC 15444-1	2000
JPEG XR	JPEG XR image coding system — Part 2: Image coding specification	ISO/IEC 29199-2	2009
FLIF	Free Lossless Image Format	none	2015
PIK	PIK	none	2017
JPEG XL	JPEG XL Image Coding System	ISO/IEC 18181-1	2022

Figure 1. *Selected lossless still image standards released from 1992-2022*

Compression efficiency and benchmark testing were conducted on a Dell Precision Tower 3620 workstation, running 64-bit Windows 10 Enterprise 10.0.14393-18362, with 32 GB RAM, 8GB AMD Radeon Pro WX 7100 GPU and Intel(R) Xeon(R) CPU E3-1285 v6 @ 4.10GHz, 4104 Mhz, 4 Cores, 8 Logical Processors. Attached to the workstation was a 27-inch Dell PQ2715Q, 3,840 x 2,160 pixels color-calibrated display.

Selected codecs were individually run from the command line in tandem with the 64-bit Command Line Process Profiling Tool v1.5.1 [8], [9]. This benchmarking application was configured to record CPU time, real (clock) time, and both RAM and video memory use during the encoding of each compressed file. For every codec tested, sets of three consecutive encodings were done per reference image. The resulting individual benchmark metrics were then averaged for each compressed test image and recorded as final values to get a more accurate sense of a given encoder's execution across possible fluctuations in overall workstation load and speed [10].

The lone exception to this test configuration were evaluations conducted on JPEG-1 original lossless compression. JPEG-1 (ISO/IEC 10918) reference software contains one of the few extant lossless encoders for the original 1992 specification [11]. Precompiled binaries of the C++ source code are difficult to locate and build for Windows OS. In turn, the software was instead compiled and installed from source on an Apple MacBook Pro (OS X 10.14.6, 16GB RAM, 3.1 GHz Intel Core i7 processor, 1TB SSD) with 13-inch 2,560 x 1,600 pixels color-calibrated display. Beyond compression savings calculations, benchmark testing was not conducted on the JPEG-1 encoder.

Reference images (Figure 2) were all 8 bit, sRGB color TIFF, PNG or PPM files based upon the input file format requirements of the encoder being tested. These images all have cultural heritage collections origins and were chosen to suitably represent the common variety of spatial and color information found in such collections [12].

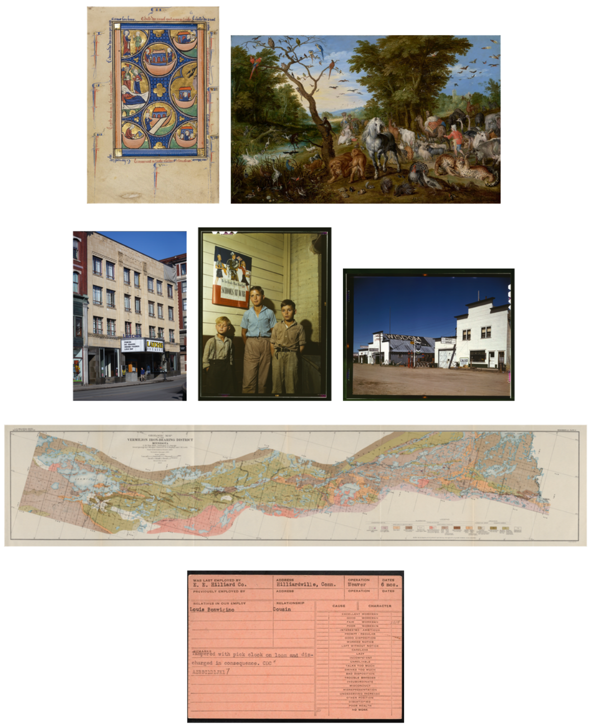

Figure 2. *Reference 8-bit, sRGB images (left-right, top-bottom) "painting_00353901," "painting_00103401," "born_digital," "medium_format_color_01," "medium_format_color_02," "map," "card"*

To verify that the resulting compressed files were mathematically lossless and reversible, pixel data of both input and output files were compared through RGB PSNR analysis using the following FFmpeg v3.4.1 filter and null muxer command [13]:

```
ffmpeg -i reference.tif -i output.tif -filter_complex psnr -f null –
```

PSNR values of infinity indicate that there is no difference between two given input signals:

```
PSNR r:inf g:inf b:inf average:inf min:inf max:inf
```

Lossless Encoding Software Commands

JPEG-1 (ISO/IEC 10918) files were encoded through the standard's reference software release 1.56 where -p and -c switches control predictive lossless integer encoding, and default Huffman coding were employed [11]. The encoder, however, was only able to work with PPM input files and not the study's main reference TIFF files. Therefore, a parallel set of PPM files were losslessly created in Adobe Photoshop from the reference TIFF files for use by the JPEG-1 encoder:

```
jpeg -p -c reference.ppm output.jpg
```

JPEG-LS (ISO/IEC 14495-1) files were created using IrfanView's [14] CharLS plugin through the /convert= instruction. CharLS supports baseline JPEG-LS Part 1 exclusively and in turn uses Huffman coding [15]:

```
i_view64.exe reference.tif /convert=output.jls
```

JPEG 2000 (ISO/IEC 15444-1) files were encoded through the following OpenJPEG v2.3.0 [16] command though which the software defaults to lossless compression:

```
opj_compress.exe -i input.tif -o output.jp2
```

JPEG XR (ISO/IEC 29199-2) files were encoded using the 64-bit NConvert v7.25 utility [17] with the following parameters... -npcd 2 = default, -c 0 = no compression, -q 100 =quality value, -ctype rgb = Channel Type, -keep_icc = Keep ICC Profile from original file and -corder inter = Interleaved Channel Order:

```
nconvert.exe -npcd 2 -c 0 -q 100 -ctype rgb -keep_icc -corder inter
-out jxr -o %.jxr input.tif
```

FLIF files were created using the FLIF v0.2.0 encoder with "effort" tuned to -E20 [18]. This version of the encoder was not able to use TIFF input files. Therefore, a parallel set of compatible lossless PNG files was created with Photoshop from the original TIFF reference images for use with the FLIF software. The effort setting, which essentially is an encoding speed regulator, was determined through a combination of Pareto optimal estimates and tested encode times that centered on possible feasible use in a production environment:

```
Flif.exe input.png -E20 output.flif
```

PIK files were encoded using the software's b4866ff Github commit compiled for Windows and the command line instruction below [5], [19]. This version of the encoder was also not able to use TIFF input files. Therefore, the same parallel set of lossless PNG files previously created from the original TIFF reference images and used in FLIF testing was also used with the PIK encoder:

```
Pik_c.exe --lossless input.png output.pik
```

JPEG XL (ISO/IEC 18181-1&2) files were created with the standard's reference software's v0.7.0 36ece478 GitLab commit compiled for Windows [20], [21]. The following specifiers were used in encoding --quality=100 = Mathematically Lossless, --effort=3 = Encoder effort setting. Like FLIF testing, the effort setting [22] was determined through a combination of Pareto optimal estimates and tested encode times that centered on possible feasible use in a production environment:

```
cjpegxl.exe --quality=100 --effort=3 input.png output.jxl
```

Note, all tested encoders, excluding JPEG-1, were benchmarked using the Process Profiling Tool's benchmarking template "-tb" switch which outputs results to the console [8]. Example of use with OpenJPEG lossless encoder:

```
ProcProfile64.exe -tb opj_compress.exe -i input.tif -o output.jp2
```

In addition, delentropy values [23] were also recorded for each reference file using the Simple Image Processing Pipeline tool calculator installed and run on Mac OS X 10.14.6 [24]:

```
sipp -in=input.tif -csv=true input.tiff,"delentropy value"
```

Finally, entropy percentages were also estimated for individual reference files by deploying the experimental paq8px v.207 lossless data encoder [25], [26] using the encoder's maximum processing memory switch "12":

```
paq8px_v207.exe -12 input.tif output directory
```

Results and Discussion

Entropy and Compression Efficiency Measures

The aim of compression is to reduce redundancy in stored data. This reduction increases effective data density through encoding information using fewer bits than the original representation [27], [28]. Most lossless compression methods do this by first generating a statistical model for the input data. The model is then used to map the input to bit sequences in such a way that "probable" (i.e. frequently encountered) data will produce shorter output than "improbable" data. In this way, lossless encoders ignore random data which has no predictive value and is itself not compressible [29], [30].

In information sources, redundancy is related to information content which Shannon and Weaver [31] describe in measures of entropy. The greater the entropy, the less predictable the data, and the more information content. Entropy, in turn, has been thought to provide a numeric correlation to the amount of statistical redundancy within a given source and ultimately the expected limit to which it can be losslessly compressed [1], [30]. The higher the entropy, the less redundancy, therefore the lower the expected compression limit.

Since redundancies in images are directly related to image content, it follows that compression is inherently scene dependent. Entropy calculations can, in turn, effectively characterize image scenes and be good predictors of compression potential for given images. However, image file entropy is inherently difficult to interpret and is contingent on how a given calculator models the input. This modeling can involve predictions on several different file elements such as bits, bytes, words, groups of pixels, etc. [32], [33].

Conventional mathematical analysis tools like MathWorks' MATLAB and Wolfram Mathematica normally calculate entropy through one dimensional intensity-based histograms that do not take into consideration the spatial distribution of pixel intensities throughout an image. This additional variable can also have a direct influence on resulting compression density. In turn, Larkin's novel delentropy measure [23] which attempts to also capture underlying spatial image structure and pixel co-occurrence was calculated for all reference images to potentially better understand correlation between an input's information content (and redundancy) and the resulting compression efficiencies of each encoder. For 8-bit grayscale source images, delentropy can range between a minimum of 0 to a maximum of 8. Since it was assumed that higher delentropy values would associate with lower compression potential, inverse delentropy percentages were calculated and plotted against lossless compression savings percentages for each codec for each test image (Figure 3).

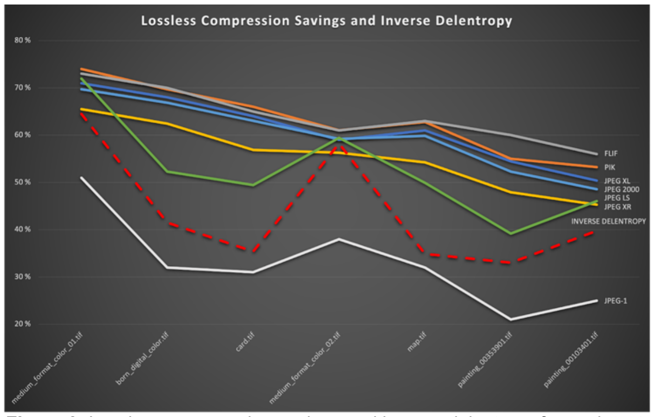

Figure 3. *Lossless compression savings and inverse delentropy for each test image*

Another technique that can be used to estimate the level of entropy in a given image file is to losslessly compress the data to its smallest feasible size using the experimental paq8px codec [26]. It works by compressing the input file bit by bit using context mixing i.e. mixing predictions from many models (at the expense of speed and memory usage). Though no application can describe all possible files and their data optimally, paq8px tops rankings on several benchmarks measuring compression ratio. As a result, it is an apt tool for estimating the lower bound of entropy for a given file. Such estimates can be accomplished by comparing the losslessly compressed size relative to the original input source.

Entropy estimates were thus made by calculating the percent decrease in file size of the paq8px compressed files compared to the study's reference TIFF files. Since it was assumed that higher entropy values would associate with lower compression potential, inverse entropy percentages were plotted against the lossless compression savings percentages of each codec for each test image (Figure 4).

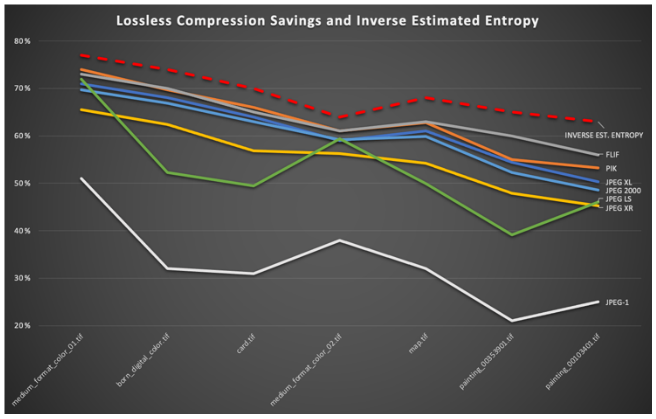

Figure 4. *Lossless compression savings and inverse estimated entropy as measured by paq8px for each test image*

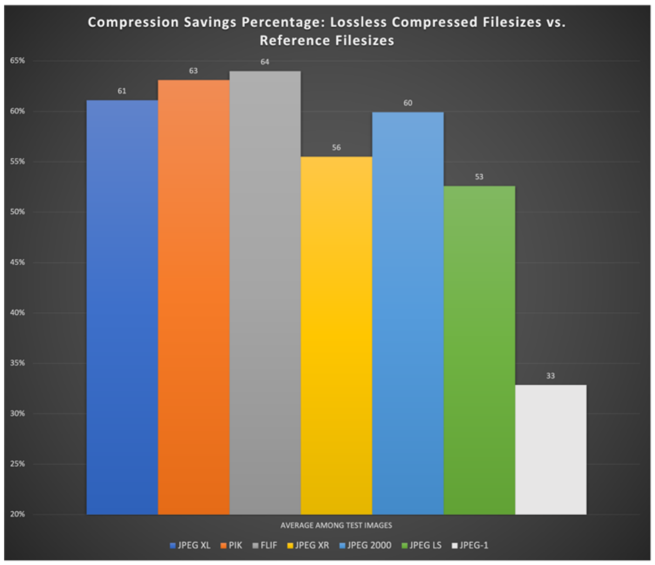

Figure 5. *Lossless compression savings average among all test images*

Data compression strategies involve compromises among resulting data density, speed, and memory use. Computer science describes these associations as space-time or time-memory tradeoffs [30], [34]. As a result, benchmarking across these individual metrics can give additional insights into the art and design of a compression software's context modeling, coding, and overall feasibility towards potential real use.

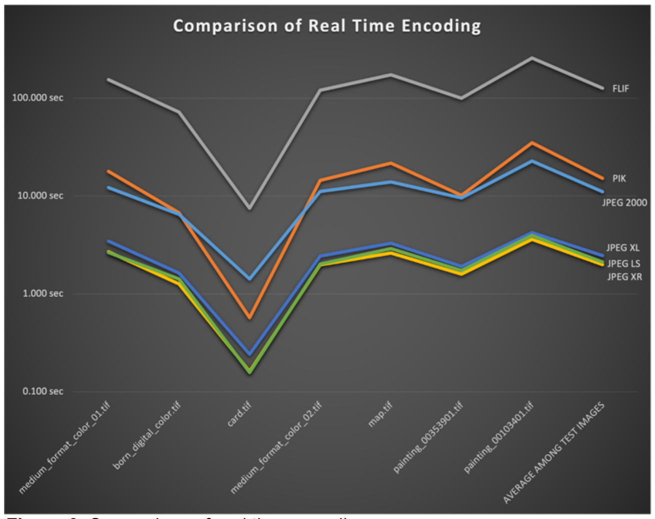

Figure 6. *Comparison of real time encoding*

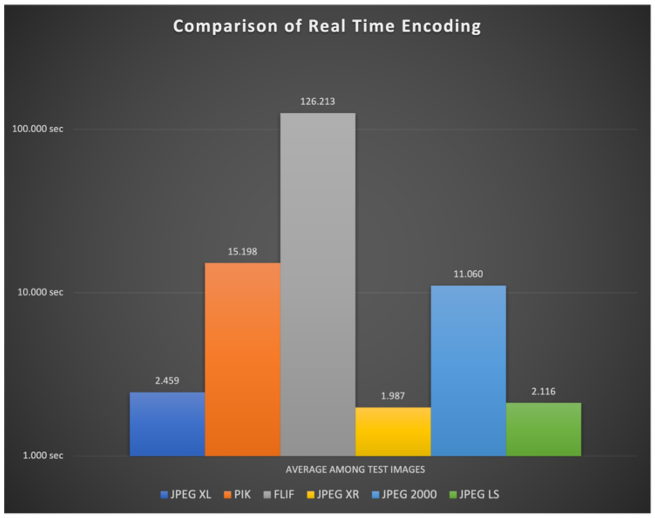

Figure 7. *Comparison of real time encoding, average among all test images*

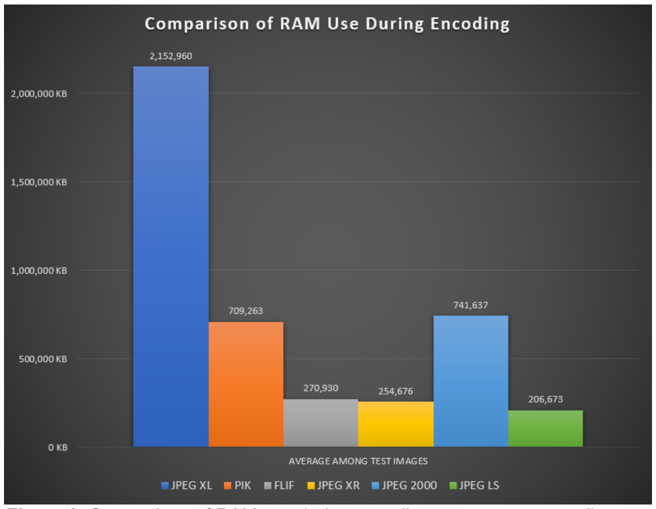

Figure 8. *Comparison of RAM use during encoding, average among all test images*

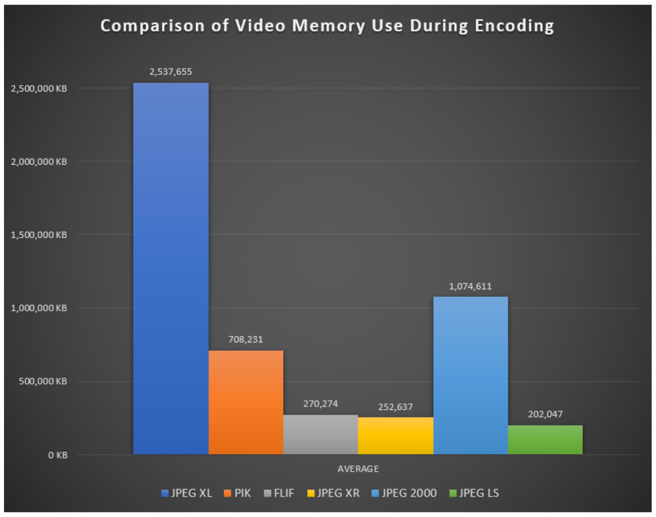

Figure 9. *Comparison of video memory use during encoding, average among all test images*

JPEG-1 and JPEG-LS

Basic lossless compression schemes primarily do two things in sequence with source images. First, they statistically predict and decorrelate the input data to identify and reduce spatial redundancy. In addition, color transformations from an RGB to a luminance-chrominance color space may also be performed during this initial preprocessing stage. Second, the results of preprocessing are then entropy coded into bit sequences through algorithms such as Huffman coding which reduces preprocessing statistical redundancies, or arithmetic coding that deals with both spatial and statistical redundancies [1].

Though differing in overall compression efficiency, JPEG-1 and baseline JPEG-LS followed similar trends in file-by-file results (Figure 3). Figure 3 also suggests that inverse delentropy calculations can be a good predictor of the compression potential of these two codecs based upon the nature of a given input file's data. Where this tool may lose accuracy, however, is when analyzing images with text and/or thin, high contrast edges like the "card" and "map" test images (and when attempting to predict the compression efficiencies of more modern, post-millennial codecs that were soon to arrive).

The gap in compression savings percentages by file between JPEG-1 and JPEG-LS is most likely due to JPEG-LS's ability to achieve more advanced and complete decorrelation through its use of the LOCO-I algorithm for prediction which also allows for residual modeling and context-based coding. JPEG-1 on the other hand employs the simpler and less effective differential pulse-code modulation (DPCM) predictive coding that is unable to attain total decorrelation of input data and involves no subsequent context modeling [29], [35], [36]. Both codecs employed Huffman entropy coding during testing. Finally, unlike specifications that would arrive in the ensuing decade, early lossless JPEG variants do not define a specific color transform during encoding [37].

JPEG 2000 and JPEG XR

Lossless JPEG 2000 and JPEG XR also share commonalities. Both employ either true wavelet (JPEG 2000) or wavelet-like (JPEG XR) biorthagonal transforms which use reversible integer arithmetic. Additionally, each uses unique internal color transforms aimed at better decorrelating the original RGB channels and improving compression efficiency and lifting schemes that significantly reduce computational complexity and increase performance. Both codecs also offer optional image tiling which can increase speed and enhance error resiliency [38]–[40].

Lossless JPEG 2000 utilizes a Discrete Wavelet Transform (DWT) in the form of an integer-to-integer, reversible 5/3 filter bank [41]. Components of the filter bank are separate high-pass and low-pass spatial filters collectively known as the analysis filter bank. The low-pass filter preserves an input signal's low frequencies and removes or attenuates high frequencies, while the high-pass filter preserves high frequencies such as edges and detail while removing or attenuating low frequencies. These filters inform the codec's iterative prediction steps during implementation. From there JPEG 2000 employs computationally expensive EBCOT context modeling followed by an MQ coder which is a modified version of one of the earliest practical applications of adaptive binary arithmetic coding [39]. Taken together, JPEG 2000 remains an efficient if complex and somewhat slow lossless compression codec to this day (Figures 4-7).

Though JPEG XR employs a unique two-stage lapped biorthogonal transform (LBT), it too selectively uses high and low frequency filters in a predictive way [40]. In turn, the codec followed a similar file-by-file pattern to JPEG 2000 of file size percent savings (Figure 4) but was 4% less efficient overall on average as JPEG 2000 (Figure 5) during testing. However, JPEG XR was the fastest encoder of all the codecs that were evaluated by 9 seconds on average and used 487MB less RAM and 822MB less video memory on average than JPEG 2000 (Figures 6-9). These findings may be partially attributed to JPEG XR's use of a form of adaptive Huffman entropy coding [42], [43]. Though often suboptimal in terms of compression, Huffman coders are less complex and faster than arithmetic coders [39], [44].

FLIF, PIK, and JPEG XL

FLIF and PIK may be viewed as experimental precursors to lossless JPEG XL. Through the codec's effort setting [22], JPEG XL uses a highly flexible modular encoding mode that employs innovative components of both of its predecessors. These elements include a weighted self-correcting predictor plus simpler predictors which are adjustable per context, and effective meta-adaptive (MA) context modeling. Finally, JPEG XL uses a "range" variant of the recently introduced Asymmetric Numeral Systems (ANS) entropy coding family. rANS can achieve compression ratios similar to arithmetic coding, while being significantly faster in performance [45]–[50].

Though FLIF, PIK, JPEG XL, and JPEG 2000 compressed images with similar efficiency on average (Figure 5), JPEG XL was much faster than these codecs (Figures 6, 7). However, it should be remarked that this speed may be partially attributed to JPEG XL's more concerted use of available memory resources during encoding (Figure 8, 9).

Conclusion

Lossless still image codecs have gone through a period of intriguing developments in the thirty years since the original JPEG-1 specification was published. The findings of this study shed broader light on how this evolution has taken shape. Through a comparative analysis of technical designs, benchmark metrics, and entropy estimates, the past and current state of lossless codecs can thus be more clearly characterized.

For instance, lossless JPEG-1 and JPEG-LS can be seen as similar variants of early generation still image compression based upon their documented schemes, benchmark results, and common image-by-image efficiency trends that closely follow inverse delentropy estimates. Of note, JPEG-LS is a fast, low complexity Huffman-based encoder (Figure 7) that uses minimal memory (Figures 8, 9) but is highly variable in compression efficiency based upon source image information (Figure 3).

In contrast, among the more modern codecs examined in this review, all were able to encode images to similar efficiency levels on average and on an image-by-image basis. That these levels also closely followed the pattern of paq8px's image specific inverse entropy estimates (Figure 4) suggest that the paq8px tool is a good predictor of the compression potential of such post-millennial codecs. Moreover, as these codecs all comparably approached paq8px's approximations for optimum lossless compression, this may be an indication that there is little practical room left for future lossless standards to make further efficiency gains through new designs.

Still, it should be remembered that JPEG XL's novel predictors, adaptive, image-specific context modelers, and its use of rANS have yet to be fully exploited as the standard is new, and more mature encoders await development around the specification. Among the lossless bitstreams examined, it is uniquely expressive and poised for future refinement through the adoption of techniques like AI heuristics. Notably, JPEG XL's adjustable effort feature, if pareto optimized for given image content, holds the promise of a highly efficient, best-in-class lossless compression scheme that can effectively leverage available hardware resources to work fast and at scale within production environments.

References

[1] E. Allen, "Manual of Photography and Digital Imaging Chapter 29 Image Compression," in Manual of Photography and Digital Imaging, 10th ed., Burlington, MA: Focal Press, 2012. [Online]. Available: https://learning.oreilly.com/library/view/the-manual-of/9780240520377/xhtml/37_Chap29.xhtml

[2] V. Derks, "Home · team-charls/charls Wiki · GitHub," Dec. 29, 2019. https://github.com/team-charls/charls/wiki (accessed Dec. 24, 2022).

[3] I. M. Pu, Fundamental Data Compression. Burlington, MA: Butterworth-Heinemann, 2006. [Online]. Available: https://doi.org/10.1016/B978-0-7506-6310-6.X5000-4

[4] K. Sayood, Introduction to Data Compression, 5th ed. Cambridge, MA: Morgan Kaufmann, 2018. [Online]. Available: https://doi.org/10.1016/C2015-0-06248-7

[5] Google, "GitHub - google/pik: A new lossy/lossless image format for photos and the internet," Jul. 2017. https://github.com/google/pik (accessed Jan. 31, 2020).

[6] A. Rhatushnyak et al., "Committee Draft of JPEG XL Image Coding System." Aug. 13, 2019. [Online]. Available: https://arxiv.org/abs/1908.03565

[7] J. Sneyers, "FLIF - Free Lossless Image Format," Oct. 2015. https://flif.info/ (accessed Feb. 24, 2019).

[8] D. Catt, "Command Line Process Profiling Tool," Dec. 30, 2013. https://encode.su/threads/1838-Command-Line-Process-Profiling-Tool (accessed Mar. 04, 2019).

[9] D. Catt, "Command Line Process Profiling Tool v1.5.1." Dec. 30, 2013. Accessed: Mar. 04, 2019. [Online]. Available: https://encode.su/attachment.php?attachmentid=2657&d=1388416311

[10] M. J. Bennett, "Dataset to Benchmarking Lossless Still Image Codecs: Perspectives on Selected Compression Standards From 1992 Through 2022," 2023. [Online]. Available: https://opencommons.uconn.edu/libr_pubs/71/

[11] T. Richter, "libjpeg Release 1.56: A complete implementation of 10918-1 (JPEG) comming from jpeg.org (the ISO group) with extensions for HDR currently discussed for standardization.," Aug. 2019. https://github.com/thorfdbg/libjpeg (accessed Jan. 24, 2020).

[12] M. J. Bennett, "Dataset to Assessing the Potential Use of High Efficiency Video Coding (HEVC) and High Efficiency Image File Format (HEIF) in Archival Still Images." 2018. Accessed: Apr. 30, 2018. [Online]. Available: https://opencommons.uconn.edu/libr_pubs/62

[13] "FFmpeg v3.4.1," Dec. 11, 2017. https://www.ffmpeg.org/ (accessed Jan. 21, 2018).

[14] "IrfanView." 2019. Accessed: Feb. 09, 2020. [Online]. Available: https://www.irfanview.com/

[15] K. Kanryu and V. Derks, "GitHub - team-charls/charls: CharLS, a C++ JPEG-LS library implementation," Dec. 29, 2019. https://github.com/team-charls/charls (accessed Jan. 24, 2020).

[16] "OpenJPEG 2.3.0 released," Oct. 04, 2017. http://www.openjpeg.org/2017/10/04/OpenJPEG-2.3.0-released (accessed Dec. 29, 2017).

[17] "NConvert v7.25," Jan. 16, 2019. https://www.xnview.com/en/nconvert/ (accessed Jan. 15, 2015).

[18] J. Sneyers, "Release FLIF v0.2 (FLIF16) · FLIF-hub/FLIF · GitHub." Sep. 22, 2016. Accessed: Feb. 09, 2020. [Online]. Available: https://github.com/FLIF-hub/FLIF/releases/tag/v0.2

[19] Shelwein, "PIK image format, compiled Windows binary of b4866ff Github commit.," May 29, 2019. https://encode.ru/threads/2793-PIK-image-format (accessed Jul. 15, 2019).

[20] "(36ece478) · Commits · JPEG / JPEG XL Reference Software · GitLab," GitLab, Jan. 20, 2022. https://gitlab.com/wg1/jpeg-xl/-/commit/36ece4788d24f8875fb5b2d924cf6c7fd210e2b6 (accessed Nov. 08, 2022).

[21] Scope, "JXL version 0.3 released - Page 2," Jan. 21, 2022. https://encode.su/threads/3564-JXL-version-0-3-released?p=72756&viewfull=1#post72756 (accessed Nov. 08, 2022).

[22] J. Sneyers, "libjxl/encode_effort.md at main · libjxl/libjxl · GitHub," Jan. 15, 2023. https://github.com/libjxl/libjxl/blob/main/doc/encode_effort.md (accessed Jan. 24, 2023).

[23] K. G. Larkin, "Reflections on Shannon Information: In search of a natural information-entropy for images," Putney, Australia, Sep. 2016. [Online]. Available: https://arxiv.org/abs/1609.01117

[24] R. Vera, "Causticity/sipp." Causticity, 2015. Accessed: Feb. 21, 2020. [Online]. Available: https://github.com/Causticity/sipp

[25] Z. Gotthardt, "paq8px - Page 85 paq8px_v207.zip," Encode, Jul. 09, 2022. https://encode.su/threads/342-paq8px?p=75128&viewfull=1#post75128 (accessed Nov. 04, 2022).

[26] M. Mahoney et al., "PAQ8PX compression archiver." Oct. 16, 2022. Accessed: Nov. 02, 2022. [Online]. Available: https://github.com/hxim/paq8px

[27] "Data compression," Wikipedia. Feb. 14, 2020. Accessed: Feb. 17, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Data_compression&oldid=940727795

[28] D. A. Lelewer and D. S. Hirschberg, "Data Compression," ACM Comput. Surv., vol. 19, no. 3, Sep. 1987, Accessed: Aug. 23, 2019. [Online]. Available: https://www.ics.uci.edu/~dan/pubs/DataCompression.html

[29] "Lossless compression," Wikipedia. Nov. 16, 2022. Accessed: Nov. 21, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Lossless_compression&oldid=1122254601

[30] M. Mahoney, "Data Compression Explained," Apr. 15, 2013. http://mattmahoney.net/dc/dce.html (accessed Jul. 17, 2019).

[31] C. Shannon and W. Weaver, The Mathematical Theory of Information. Urbana: University of Illinois Press, 1949. [Online]. Available: https://pure.mpg.de/pubman/item/item_2383164_3/component/file_2383163/Shannon_Weaver_1949_Mathematical.pdf

[32] Z. Gotthardt, "Entropy of a file," Encode, May 01, 2022. https://encode.su/threads/3860-Entropy-of-a-file?p=74069&viewfull=1#post74069 (accessed Nov. 06, 2022).

[33] M. Rabbani and P. W. Jones, "4. Entropy Estimation and Lossless Compression," in Digital Image Compression Techniques, SPIE, 1991, p. 33. [Online]. Available: https://app.knovel.com/hotlink/pdf/id:kt00850TRP/digital-image-compression/entropy-estimation-lossless

[34] "Space–time tradeoff," Wikipedia. Feb. 11, 2020. Accessed: Mar. 01, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Space%E2%80%93time_tradeoff&oldid=940179567

[35] "JPEG Lossless Compression (ISO/IEC 14495)," Jun. 17, 2021. https://www.loc.gov/preservation/digital/formats/fdd/fdd000151.shtml (accessed Mar. 24, 2019).

[36] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," IEEE Trans. Image Process., vol. 9, no. 8, pp. 1309–1324, Aug. 2000, doi: 10.1109/83.855427.

[37] D. S. Taubman and M. W. Marcellin, "JPEG2000 Image Compression Fundamentals, Standards and Practice, Part IV Other Standards," in JPEG2000 Image Compression Fundamentals, Standards and Practice, Boston, MA: Springer US, 2002. doi: 10.1007/978-1-4615-0799-4.

[38] S. Lawson and J. Zhu, "Image compression using wavelets and JPEG2000: a tutorial," Electron. Commun. Eng. J., vol. 14, no. 3, pp. 112–121, Jun. 2002, doi: 10.1049/ecej:20020303.

[39] M. Rabbani and R. Joshi, "An overview of the JPEG 2000 still image compression standard," Signal Process. Image Commun., vol. 17, no. 1, pp. 3–48, Jan. 2002, doi: 10.1016/S0923-5965(01)00024-8.

[40] C. Tu, S. Srinivasan, G. J. Sullivan, S. Regunathan, and H. S. Malvar, "Low-complexity hierarchical lapped transform for lossy-to-lossless image coding in JPEG XR / HD Photo," presented at the Optical Engineering + Applications, San Diego, California, USA, Aug. 2008, p. 70730C. doi: 10.1117/12.797097.

[41] D. Le Gall and A. Tabatabai, "Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques," in ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing, New York, NY, USA, 1988, pp. 761–764. doi: 10.1109/ICASSP.1988.196696.

[42] S. S. Jadhav and S. K. Jadhav, "JPEG XR an Image Coding Standard," Int. J. Comput. Electr. Eng., pp. 137–140, 2012, doi: 10.7763/IJCEE.2012.V4.465.

[43] Sridhar Srinivasan, Chengjie Tu, Shankar L. Regunathan, and Gary J. Sullivan, "HD Photo: a new image coding technology for digital photography," presented at the Proc.SPIE, Sep. 2007, vol. 6696, p. 66960A. doi: 10.1117/12.767840.

[44] K. Tatwawadi, "What is Asymmetric Numeral Systems? Understanding the new entropy coder family.," 2019. https://kedartatwawadi.github.io/post--ANS/ (accessed Dec. 03, 2022).

[45] J. Alakuijala, J. Sneyers, L. Versari, and J. Wassenberg, "JPEG White Paper: JPEG XL Image Coding System," ISO/IEC JTC 1/SC 29/WG1 N90063, JPEG White Paper V. 1.4, Jan. 2021. Accessed: Nov. 23, 2022. [Online]. Available: https://ds.jpeg.org/whitepapers/jpeg-xl-whitepaper.pdf

[46] J. Duda, "Asymmetric numeral systems: entropy coding combining speed of Huffman coding with compression rate of arithmetic coding." arXiv, Jan. 06, 2014. Accessed: Dec. 03, 2022. [Online]. Available: http://arxiv.org/abs/1311.2540

[47] J. Duda, K. Tahboub, N. J. Gadgil, and E. J. Delp, "The use of asymmetric numeral systems as an accurate replacement for Huffman coding," in 2015 Picture Coding Symposium (PCS), Cairns, Australia, May 2015, pp. 65–69. doi: 10.1109/PCS.2015.7170048.

[48] JPEG XL: The Next Generation "Alien Technology From The Future" by Jon Sneyers [IMAGE READY], (Nov. 24, 2020). Accessed: Dec. 31, 2022. [Online Video]. Available: https://www.youtube.com/watch?v=t63DBrQCUWc

[49] J. Sneyers, "JPEG XL Reference Software, JPEG XL Format Overview, format_overview.md - GitLab," 2021. https://gitlab.com/wg1/jpeg-xl/-/blob/f88745497118727f861cb00887cadcb395d10f1c/doc/format_overview.md (accessed Dec. 02, 2022).

[50] J. Sneyers and P. Wuille, "FLIF: Free lossless image format based on MANIAC compression," in 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, Sep. 2016, pp. 66–70. doi: 10.1109/ICIP.2016.7532320.

Author Biography

Michael J. Bennett is Head of Digital Imaging and Conservation at the University of Connecticut. There he oversees the digital capture and conservation operations for the UConn Library. His research interests include technologies and techniques that focus on image acquisition, post-processing, and 2D and 3D data formats. He holds a BA from Connecticut College and an MLIS from the University of Rhode Island.