High Throughput JPEG 2000 (HTJ2K) and the JPH file format: a primer

Editors: Pierre-Anthony Lemieux (Sandflow Consulting LLC, USA), Reji Mathew (UNSW Sydney, Australia), Aous Naman (UNSW Sydney and Kakadu GPU, Australia), Shigetaka Ogawa (ICT-Link, Japan), Michael Smith (Wavelet Consulting LLC, USA), David Taubman (UNSW Sydney and Kakadu Software, Australia), Osamu Watanabe (Takushoku University, Japan)

Executive Summary

High Throughput JPEG 2000 (HTJ2K) is a new addition to the JPEG 2000 family of International Standards developed by JPEG Committee (ISO/IEC JTC 1/SC 29/WG 1).

HTJ2K brings an order of magnitude increase in throughput to JPEG 2000: approximately 10x for moderate to higher compressed bit rates and more than 30x for lossless coding. HTJ2K software and GPU throughputs become comparable to, if not higher than, that achievable with the original, much less functional, JPEG algorithm. Low-cost, high throughput hardware implementations can also be realized.

For example, full 4K 4:4:4 36-bit/channel video can be encoded to 2 bits/pixel at more than 90 fps and decoded at more than 140 fps on a 3.4GHz 4-core Skylake desktop processor, while a mid-range GTX1080 GPU can decode the same content at around 500 fps.

HTJ2K achieves these improvements by introducing a new HT block coder, which is a drop-in replacement for the original JPEG 2000 Part 1 block coder (J2K-1) and allows truly reversible transcoding to/from J2K-1.

HTJ2K preserves all features of JPEG 2000 Part 1 apart from quality scalability. It is compatible with the extensions defined by JPEG 2000 Part 2, the interactive communication protocols defined by JPEG 2000 Part 9, and other parts of the JPEG 2000 family. It fully supports resolution scalability, efficient spatial random access, multi-spectral and hyper-spectral content, and high throughput non-iterative precise rate control. It retains high coding efficiency from lossless all the way down to bit rates on the order of 0.5 bits/pixel.

Unlike J2K-1, the HT coder is not fully embedded and hence quality scalability is largely sacrificed. HTJ2K codestreams can however preserve quality layer boundaries enabling truly reversible transcoding between HT and J2K-1.

HTJ2K addresses the needs of professional video capture, editing, streaming and contribution markets, including high throughput lossless coding of high precision and half-float content. HTJ2K simultaneously addresses a broad range of applications from point-to-point streaming to extremely efficient image capture, preview and browsing. At the same time, high energy efficiency makes it an excellent candidate for mobile and satellite imaging applications.

Finally, the accompanying JPH file format updates JPEG 2000 formats with modern colour space support, supporting HDR content with almost unlimited precision, while also allowing for the representation of raw colour sensor data with custom colour filter array patterns.

Like JPEG 2000 Part 1, the HTJ2K standard is intended to be royalty free[1].

Table of Contents

Introduction....2
Usage scenarios....3
Overall architecture....4
Concurrency features....5
File formats and implementations....6
Throughput and coding efficiency....6
Conclusion....10

Introduction

JPEG 2000 provides a rich set of features that find application in many diverse fields. Some of the most important features, as found in JPEG 2000 Part 1 are:

- *Compression efficiency*
- *Quality scalability* – ability to extract almost any reduced quality representation from the codestream while retaining full coding efficiency
- *Resolution scalability* – ability to extract almost any power-of-2 related resolution from the codestream while retaining the full coding efficiency
- *Region-of-interest accessibility* – ability to reconstruct or communicate an arbitrary spatial region, while retaining high coding efficiency
- *Parallelism* – ability to decode and encode in parallel across many CPU cores, GPU threads or in hardware
- *Non-iterative optimal rate control* – ability to achieve a target compressed size without iterative encoding

Most of these features derive from the use of the EBCOT algorithm (Embedded Block Coding with Optimized Truncation), while use of the hierarchical Discrete Wavelet Transform (DWT) also plays an important role. In addition to these core features, the JPEG 2000 family of standards support the following applications:

- Efficient and responsive remote interactive browsing of imagery (including video and animations), via JPEG 2000 Part 9, also known as the JPIP standard.
- High dynamic range (HDR) compression, including support for non-linear tone curves as defined in JPEG 2000 Part 2.
- Rich metadata annotation, as defined in JPEG 2000 Part 2.
- Efficient compression of hyper-spectral and volumetric content, via JPEG 2000 Part 2.

The primary drawback of JPEG 2000 is computational complexity, which is particularly burdensome for ultra-high definition video, and power/energy conscious applications.

The HTJ2K standard introduces the HT block coder, which is a drop-in replacement for the J2K-1 block coding algorithm defined in JPEG 2000 Part 1 and achieves an order of magnitude throughput improvement. This white paper provides a primer to HTJ2K applications, technology, capabilities and performance.

Usage scenarios

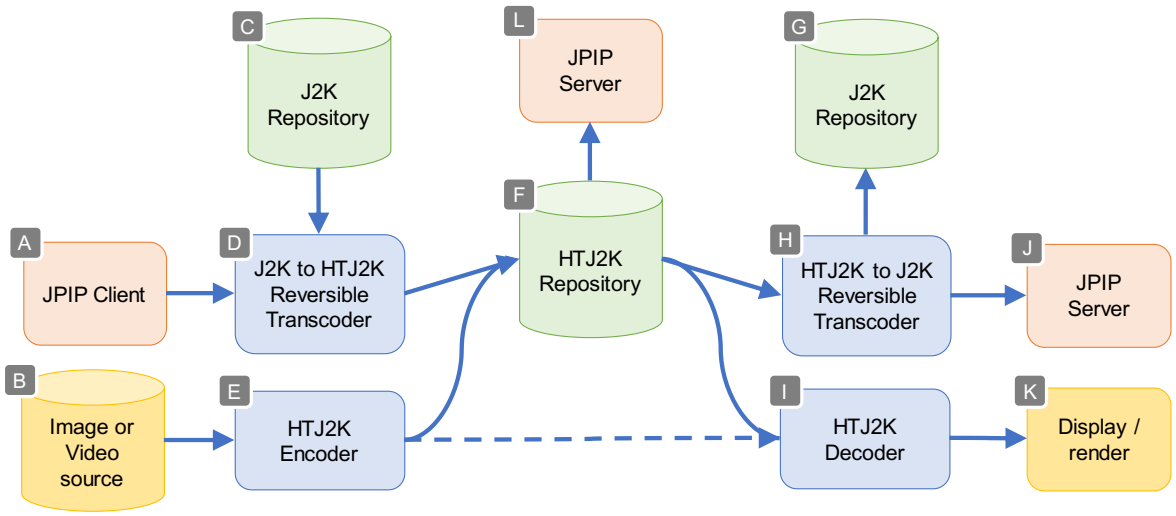

Figure 1: HTJ2K systems diagram

Distribution, archival and management of motion picture content

JPEG 2000 is widely used as an intermediate distribution format for digital cinema and video assets. This is facilitated by the Interoperable Master Format (IMF) specifications defined by the Society for Motion Picture Television Engineers (SMPTE). Such applications can benefit considerably from the use of HTJ2K. Following path **C-D-F-I-K** in the HTJ2K systems diagram of Figure 1, existing archived JPEG 2000 media can be transcoded, stored, distributed and rendered with greatly reduced computational complexity. Path **F-H-G** allows the original JPEG 2000 representation to be recovered losslessly, if required.

Energy efficient image capture and preview

HTJ2K and its associated JPH file format have many desirable properties as an image capture format for digital cameras, mobile phones and more advanced multi-sensor imaging devices. Direct encoding to HTJ2K requires very low energy in software, GPU or hardware. HTJ2K supports very high sample precisions (e.g., 12-, 16- or even 24 bits per sample), allowing high dynamic range content to be encoded with non-linear transfer functions like HLG and PQ, and linear and log-like representations – even losslessly. Even more important are the resolution scalability and region accessibility features of HTJ2K, which allow a JPH file to be interactively previewed with extremely low energy – vastly lower than that required to interact with a JPEG file. This usage scenario exercises path **B-E-F-I-K** in the HTJ2K systems diagram of Figure 1. Additionally, HTJ2K works seamlessly with the JPIP standard for interactive remote browsing of JPEG 2000 imagery. This provides a highly efficient mechanism to support remote preview of captured content from devices in an ad-hoc network, exercising path **B-E-F-L**. The most efficient and responsive possible browsing of remote content is enabled by transcoding HT code-blocks on demand, to the fully embedded J2K-1 representation, exercising path **B-E-F-H-J** in the HTJ2K systems diagram.

Live performances, broadcasting, view finding and telepresence systems

This use case exercises path **B-E-I-K** in the HTJ2K systems diagram of Figure 1, taking advantage of HTJ2K's extremely high encoding and decoding throughput, its high coding efficiency even at lower bit rates, and its non-iterative rate control capability, with low (sub-frame) end-to-end latency. These properties allow high quality 4K content to be streamed over conventional IP or local wireless links. Heterogeneous systems, involving low cost hardware, software or GPU platforms are fully supported with tunable latency, due to the flexible block-based structure of JPEG 2000.

Cache management for interactive rendering

Large images are often viewed interactively, by resolution or region of interest, so that successive rendered views contain many code-blocks in common. Each time a region of interest is rendered at some resolution, the relevant code-blocks are normally decoded on demand, being accessed dynamically, either from a local file or a client cache. The introduction of HTJ2K provides much more energy efficient ways to handle such applications. If the source content already employs the HT block coder, repeated decoding of common code-blocks during interactive rendering becomes vastly cheaper than with the J2K-1 block coding algorithm. If original media uses the J2K-1 block coder, code-blocks that are being repeatedly accessed can be transcoded to an equivalent HT representation and stored within an intelligent content cache, from which re-rendering becomes much less expensive in the future. The HT and J2K-1 representations consume similar amounts of cache memory and all information is preserved. Applications of this type exercise some or all of path **A-D-F-I-K** in the HTJ2K systems diagram of Figure 1.

Overall architecture

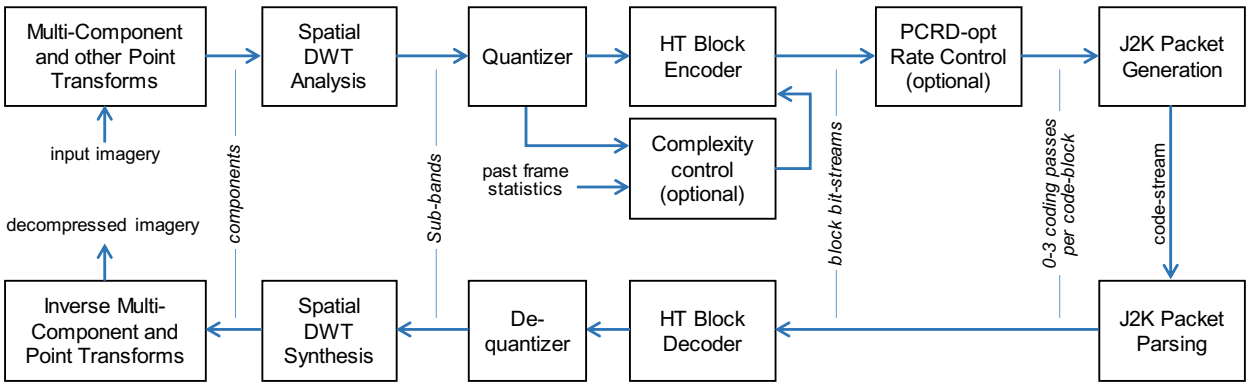

Figure 2: HTJ2K compression and decompression systems.

Figure 2 illustrates the elements involved in HTJ2K compression and decompression. HTJ2K substantially preserves the existing architecture and codestream syntax of JPEG 2000. Imagery is first subjected to any required multi-component transforms and/or non-linear point transforms, as defined in Part 1 or Part 2 of JPEG 2000, after which transformed image components are processed by a reversible or irreversible Discrete Wavelet Transform (DWT), producing a hierarchy of detail sub-bands and one base (LL) sub-band.

All sub-bands are partitioned into blocks with no more than 4096 samples, typical dimensions being 64x64 or 32x32; very wide and short blocks such as 1024x4 are also important for low latency applications. Each block is individually quantized (if irreversible) and coded, producing a block bit-stream comprising zero or more coding passes.

In the encoder, an optional Post-Compression Rate-Distortion optimization (PCRD-opt) phase is used to discard generated coding passes so as to achieve a rate or distortion target, which may be global (whole codestream) or local (small window of code-blocks). Finally, the bits belonging to the selected coding passes from each code-block are assembled into JPEG 2000 packets to form the final codestream.

Like J2K-1, the HT block coding algorithm also adopts a coding pass structure, with Cleanup, SigProp and MagRef coding passes, defined with respect to bit-planes p. However, the Cleanup pass associated with each bit-plane p fully encodes the magnitude and sign information encoded within all previous (larger p) coding passes, so that there is no point in emitting all of them to the codestream. The HT refinement passes, SigProp and MagRef, encode the same information as their J2K-1 counterparts, which allows a J2K-1 codestream to be transcoded to the HTJ2K format without altering its quantized sample representation in any way. To fully recover this information, an HT block decoder must process at most one Cleanup, one SigProp and one MagRef pass, whereas a J2K-1 block decoder may need to process many passes.

In both the J2K-1 and HT block coders, an encoder may drop any number of trailing coding passes from the information included in the final codestream. Indeed, the encoder need not generate such coding passes at all, if it can reasonably anticipate that they will be dropped.

With the HT block coder, both leading and trailing coding passes may be dropped (or never generated) by an encoder, so long as the first emitted coding pass is a Cleanup pass. This is the role of the Complexity Control element in Figure 2. As it turns out, it is usually sufficient for an HT encoder to generate just 6 coding passes, corresponding to two consecutive HT Sets. Later, the PCRD-opt stage selects at most 3 passes of the generated coding passes from each code-block for inclusion in the final codestream, where the selected passes belong to a single HT Set. One example of an effective complexity control strategy is the Cplex-EST algorithm offered by the Kakadu implementation[5]. It allows both still images and video content to be encoded with a fixed (typically 6) maximum number of coding passes per block while achieving precise rate control.

In some cases, there is no need for an encoder to generate more than a single HT Cleanup pass. This is certainly true for lossless compression, or where compression is driven simply by quantization step sizes. To parameterize these quantization step sizes, an HT Quality Factor with a similar meaning to that commonly used with JPEG, is under investigation[2].

Concurrency features

All JPEG 2000 codestreams are comprised of code-blocks with at most 4096 samples each, all of which can be processed in parallel. One of the things that makes the HT block coder so fast is that each HT block bit-stream is comprised of 3 to 5 byte streams, which are only loosely coupled so that they can also be processed concurrently. Beyond this, the individual processing steps of the HT block coder are almost all vectorizable, capable of concurrent processing in very wide vectors with 32 or even 64 samples each.

File formats and implementations

The HTJ2K standard defines a wrapping file format known as JPH, which extends the JP2 file format with support for modern parameterizable colour spaces, images without any colour space, and custom colour mappings suitable for raw sensor compression.

SMPTE has added support for HTJ2K in MXF files in a recent revision of its SMPTE ST 422 standard[3]. This should make it easy to adapt HTJ2K to D-Cinema, IMF and other MXF-based applications. An open source implementation is available[4].

JPEG 2000 Part 16 specifies the carriage of JPEG 2000 codestreams in ISO/IEC 23008-12, commonly referred to as HEIF. A revision is underway to support wrapping of HTJ2K codestreams.

The Kakadu SDK and demonstration executables[5] all support HTJ2K and the JPH file format, in conjunction with all supported features from JPEG 2000 Part 1, Part 2, Part 3 and Part 9.

OpenJPH[6] is an open source implementation of the HTJ2K standard. It includes an encoder and decoder implemented in C++ and a lightweight JavaScript decoder for web applications.

ICT-Link[7] has independently developed a reference implementation of the HTJ2K standard known as “TT,” written in C++, which forms the basis for the HTJ2K reference software in JPEG 2000 Part 5.

A Matlab-based HTJ2K encoder and decoder implementation[8] has been developed by Osamu Watanabe (Takushoku University, Japan).

Throughput and coding efficiency

Throughput compared to J2K-1

We present throughput results for the 292 frame 12 bit/channel 4K 4:4:4 JPEG test video ARRI_AlexaDrums_3840x2160p_24_12b_P3_444[9].

Table 1 presents throughput results for a configuration involving 128x32 code-blocks and an irreversible version of the LeGall 5/3 wavelet transform, a feature that relies upon the coding extensions defined in JPEG 2000 Part 2. The same table includes throughput results for the more common choice of 32x32 code-blocks. All of these results involve non-iterative precise rate control, with typical visual weighting factors used to drive the distortion model behind the post-compression rate-distortion optimization phase shown in Figure 2.

Table 1: Throughput performance for visually weighted encoding with irreversible Le Gall 5/3 wavelet transforms.

	128x32 code-blocks				32x32 code-blocks			
	ENCODING (fps[10])		DECODING (fps)		ENCODING (fps)		DECODING (fps)	
bpp[11]	J2K-1	HTJ2K	J2K-1	HTJ2K	J2K-1	HTJ2K	J2K-1	HTJ2K
1	15.7	112 (7.1x)	33.6	187 (5.6x)	12.8	94 (7.3x)	32.5	175 (5.4x)
2	10.1	95 (9.4x)	17.6	156 (8.9x)	8.8	76 (8.6x)	17.1	143 (8.4x)
4	6.2	75 (12x)	9.2	117 (13x)	5.7	59 (10x)	8.9	105 (12x)

Table 2 presents throughput and PSNR results for a configuration involving 64x64 code-blocks with the irreversible CDF 9/7 wavelet transform defined in JPEG 2000 Part 1; in this case visual weighting is disabled so that the PSNR metric reported in the table is the objective actually targeted during encoding.

Table 2: Throughput and objective compression performance, measured via PSNR, for unweighted encoding with the irreversible CDF 9/7 wavelet transform with 64x64 code-blocks.

	ENCODING (fps)		DECODING (fps)		PSNR (dB)	
bpp	J2K-1	HTJ2K	J2K-1	HTJ2K	J2K-1	HTJ2K
1	13.2	84 (6.4x)	31.2	117 (3.8x)	41.41 dB	40.92 dB
2	8.5	76 (8.9x)	16.7	112 (6.7x)	45.54 dB	44.83 dB
4	5.3	64 (12x)	8.6	91 (11x)	51.18 dB	50.10 dB

Coding efficiency compared to J2K-1

We present results to indicate the coding efficiency difference between J2K-1 and the HT block coders. This is done by encoding 224 high resolution (36.4 Megapixel) images, converted from raw Sony A7R photos to 16-bit/channel TIFFs. The images are compressed first to JP2 files (J2K-1 block coder) with a fixed bit rate and visual optimization, then reversibly transcoded to JPH files (HT block coder). The histogram of the change of file size is presented in Figure 3 for three compressed bitrates. Evidently, the extremely high throughput of HTJ2K is obtained at the expense of a small loss in coding efficiency, on the order of about 6%, relative to the original JPEG 2000 algorithm.

Throughput compared to the original JPEG algorithm

Figure 4 provides some insight into the relative throughput of HTJ2K and the original JPEG algorithm; the comparison is with the heavily optimized libjpeg-turbo implementation[12], evaluated on the same i7 Skylake processor above, using TurboJPEG's tjbench tool, with just one thread. We note that HTJ2K scales pretty much linearly with thread count, while JPEG cannot (especially for decoding), HTJ2K coding efficiency is about 30% better for the same visual quality, and HTJ2K offers exact rate control without iterative encoding, region and resolution-of-interest accessibility, high sample precision and many other features that are unavailable with the original JPEG algorithm.

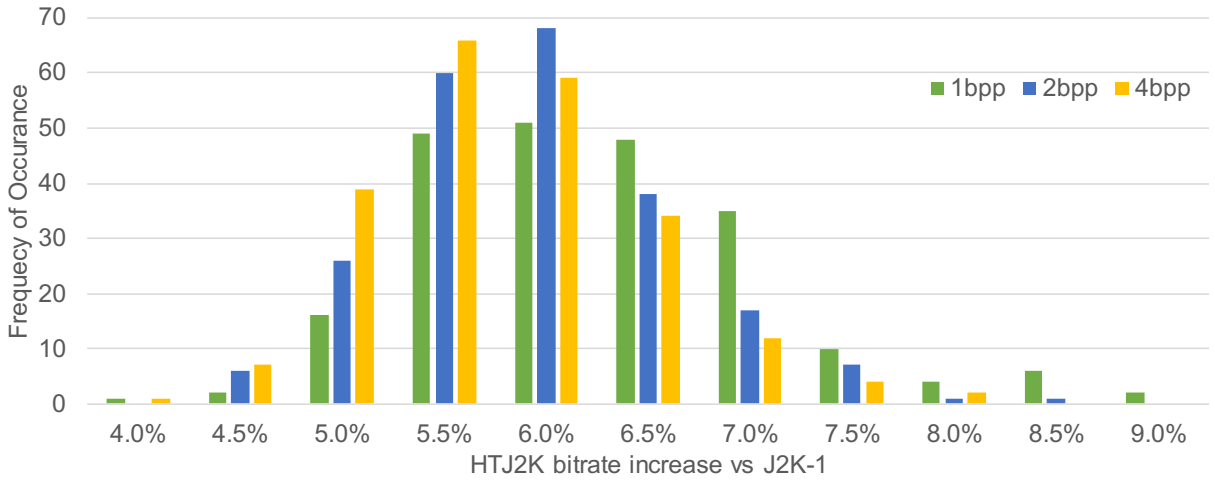

Figure 3: Histogram of the size increase of individual JPH files after reversible transcoding from JP2, for a set of 224 high resolution and high precision (16-bit/channel) RGB photographic images at three compressed bitrates.

JPEG 2000 was designed to support low latency compression. In fact, it is possible to achieve extremely low end-to-end latencies, down to a few tens of lines of an image or video frame. At very low latencies, however, the number of code-blocks available for concurrent processing necessarily reduces, which can make it hard or even impossible to achieve very high throughputs with the original JPEG 2000 block coding algorithm, even in hardware. HTJ2K provides the solution to this problem, by delivering a block coding algorithm with both much lower complexity and much higher levels of internal concurrency.

Latency

HTJ2K can be configured to achieve very low end-to-end latencies, using short and wide code-blocks. For example, with 2 levels of vertical 5x3 DWT, fundamental end-to-end latency can be 24 lines, while with 3 levels of vertical 9x7 DWT the fundamental end-to-end latency becomes 76 lines. Allowing for computation delay, practical latencies may be larger by up to 50% in a hardware implementation, so that the 24 line configuration corresponds to ~0.25ms latency for 2160p/60 content or ~0.5ms for 1080p/60 content. Figure 5 reveals the rate-distortion performance of HTJ2K as a function of end-to-end latency, compared with a full frame configuration[13].

GPU deployment

A first GPU implementation of HTJ2K was presented in 2019 at the IEEE International Conference on Image Processing[14] (ICIP'2019), including all elements of the decoding pipeline in Figure 2. On an NVIDIA GTX1080 GPU, the implementation was able to decode 4K 4:4:4 12bit/channel video at 560 fps, 440 fps and 402 fps, respectively, from 1 bpp, 4 bpp and losslessly compressed codestreams. To the best of our knowledge, this performance exceeds that achievable by the original JPEG algorithm, or indeed any other standardized image codec.

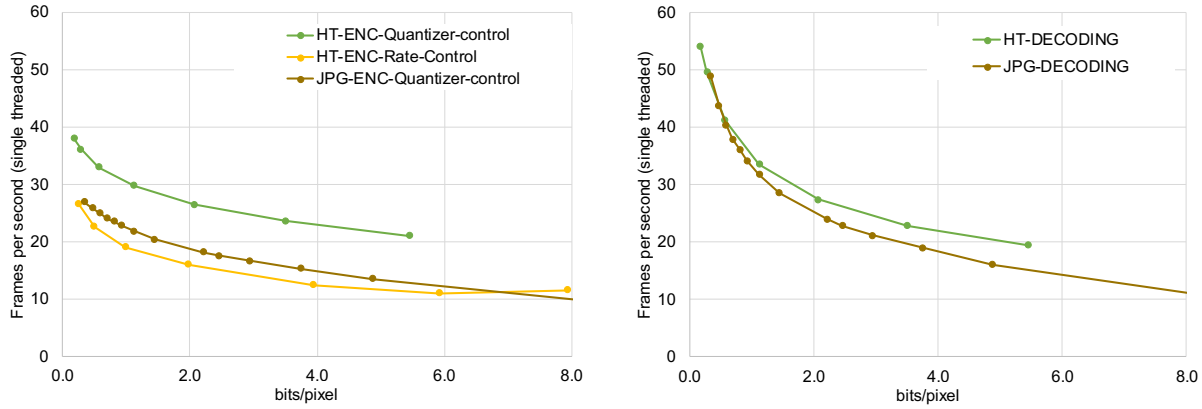

Figure 4: Single-threaded encode (ENC) and decode (DECODING) throughput comparison between HTJ2K and the original JPEG algorithm (JPG), for 4K 4:4:4 8-bit/channel content, based on the Kakadu implementation of HTJ2K and the libjpeg-turbo implementation of JPEG, showing both rate-controlled and quantizer-based (quality-factor) encoding methods for HTJ2K. Note that JPEG doesn't offer rate control, and bit rates for HTJ2K are about 30% smaller than for JPEG at similar quality.

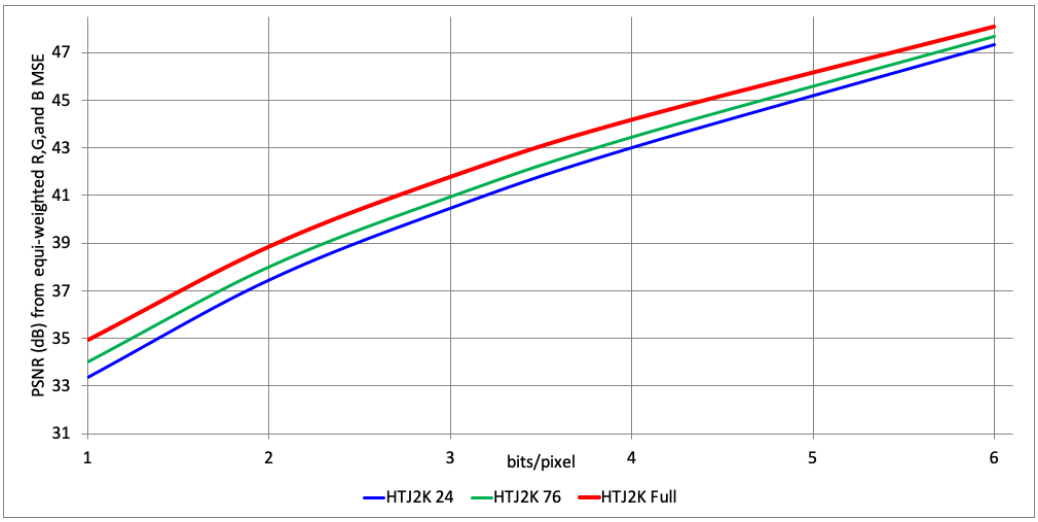

Figure 5: Rate-distortion performance of HTJ2K over 10 RGB 4:4:4 test images in low latency configurations with fundamental end-to-end latencies of 24 lines (2 levels of vertical 5x3 DWT) and 76 lines (3 levels of vertical 9x7 DWT), compared with full HTJ2K (5 levels of 9x7 DWT). Including computation, practical latencies may be larger by up to 50% in a hardware implementation.

In a separate paper[15] submitted to ICIP'2020, a GPU encoding implementation is described that is capable of reaching throughputs of 455 fps and 435 fps, respectively, for 1 bpp and lossless compression of the same 4K 4:4:4 content, again on the GTX1080 GPU. These results are expected to improve with further optimization, but already suggest that HTJ2K may be able to achieve real-time encoding and decoding of 8K 4:4:4 content at 120 fps on common GPU platforms.

FPGA deployment

It is possible already to provide evidence for the benefits of HTJ2K in hardware. ICT-Link[7] has developed a prototype FPGA-based encoder[16] for HTJ2K, whose HT block encoder module is capable of processing 4 samples per clock cycle, at clock rates of 120MHz on an Intel Stratix EP1 FPGA, using approximately 3000 logic

elements. The complete HTJ2K encoder includes support for 9x7 and 5x3 wavelet transforms, bit-depths from 8 to 16 bits/sample, with 1 or 3 colour components, and decorrelating colour transform, with all processing performed on-chip. This level of performance is well matched to the data streams produced by modern high-performance CMOS image sensors, being readily scaled to match the number of available readout channels offered by the sensor.

Conclusion

With a relatively minor addition, HTJ2K brings sweeping improvements to the JPEG 2000 family of standards.

HTJ2K provides higher coding efficiency, resolution scalability, region-of-interest accessibility and much more parallelism compared to the original JPEG algorithm, while achieving similar if not higher throughput compared to a single threaded and heavily optimized implementation.

HTJ2K preserves almost all the rich feature set of JPEG 2000, except for quality scalability, offering an order of magnitude increase in throughput (i.e., vastly lower computational complexity) at the cost of a 5-10% reduction in coding efficiency.

HTJ2K and the original J2K-1 representations are fully interchangeable, allowing reversible transcoding to be incorporated at any point within a capture, distribution, archiving, caching or rendering system, to obtain the best features of both algorithms without sacrificing the integrity of the data. HTJ2K can even preserve all quality layering, profiles and other aspects of a non-HTJ2K codestream during transcoding.

---

[1] The primary technology contributor to HTJ2K (Kakadu R&D Pty Ltd) has made royalty-free declarations to the ITU and ISO.

[2] Ayyoub Ahar; Saeed Mahmoudpour; Osamu Watanabe; David Taubman; Peter Schelkens, "Parameterization of the quality factor for the high throughput JPEG 2000, “Proceedings Volume 11353, Optics, Photonics and Digital Technologies for Imaging Applications VI; 113530V (2020) (doi: 10.1117/12.2557414).

[3] https://doi.org/10.5594/SMPTE.ST422.2019

[4] https://github.com/sandflow/jid

[5] https://kakadusoftware.com

[6] https://github.com/aous72/OpenJPH

[7] Contact: Shigetaka Ogawa, s_ogawa@mug.biglobe.ne.jp

[8] https://bitbucket.org/osamu620/mathtj2k/

[9] All throughputs are obtained using an HP EliteDesk 800 G2 Small Form Factor PC (L1G76AV) with 2x8GB of DDR4 2133 MHz RAM, an Intel i7-6700 4-core processor with 3.4GHz base frequency and 4.0GHz max turbo frequency. Encoding is achieved using the Kakadu application kdu_v_compress, using the “Cplex-EST” algorithm for the complexity control module shown in Figure 2, to ensure that no more than two HT-Sets are produced for any code-block. Decoding is achieved using the Kakadu application kdu_vex_fast.

[10] fps = frames per second

[11] bpp = bits per pixel

[12] https://github.com/libjpeg-turbo/libjpeg-turbo

[13] The results presented here are obtained using the entirely intra-frame low-latency version of the Cplex-EST algorithm in Kakadu’s HTJ2K implementation, encoding at most two HT Sets per code-block to ensure high throughput. PSNR results are averaged over a diverse set of ten RGB test images from the JPEG test set: Drums; Helicopter View; Pendulus Wide; Crowd Run; Park Joy; Screen Content; Basket Ball; Sintel-2; Bike; and Woman.

[14] A. Naman and D. Taubman, “Decoding high-throughput JPEG 2000 (HTJ2K) images on a GPU,” IEEE International Conference on Image Processing, 2019.

[15] A. Naman and D. Taubman, “Encoding high-throughput JPEG 2000 (HTJ2K) images on a GPU,” submitted to IEEE International Conference on Image Processing, 2020.

[16] Shigetaka Ogawa, “JPEG 2000 HT Encode Hardware,” ISO/IEC JTC1/SC29/WG1 input document M87040, 17 April 2020.