High Throughput JPEG 2000 (HTJ2K) Performance on Laptop & Desktop PCs

Case study

JPEG 2000 is widely used in the media and entertainment industry for Digital Cinema distribution[1], Studio Video Masters via the Interoperable Master Format[2] and Broadcast Contribution links[3]. High Throughput JPEG 2000 (HTJ2K or JPEG 2000 Part 15) is an update to JPEG 2000 published in 2019[4,5]. HTJ2K provides an order of magnitude speed up over legacy JPEG 2000 Part 1.

This case study compares the *throughput and coding efficiency of HT2K, ProRes and JPEG 2000 Part 1 in a typical media and entertainment use case*: processing motion picture content with widely available commercial software tools running on an ordinary notebook computer. Apple ProRes is a proprietary codec that is also widely used within the industry, due to its low complexity that enables real time playback on common CPU platforms like notebook computers. The results show that HTJ2K achieves a throughput that is at least comparable to ProRes while outperforming ProRes in rate distortion efficiency.

CPU-based HTJ2K throughput

Speed testing was performed on a MacBook Pro (15-inch, 2018) with 2.9Ghz 6-core Intel i9-8950HK CPU with 16GB RAM using BlackMagic DaVinci Resolve Studio 17.4.4, Apple Compressor 4.5.4 and the Kakadu v8.2.1 applications “kdu_vcom_fast” and “kdu_vex_fast.” BlackMagic DaVinci Resolve[6] and Apple Compressor[7] are commonly used software applications for high quality editing, mastering and compression within the professional media industry. The Kakadu “kdu_vcom_fast” and “kdu_vex_fast” applications are included with the Kakadu SDK[8] that perform JPEG 2000 encoding and decoding respectively. The test clip spans 1 minute from the *Meridian*[9] short in 3840×2160 24p 4:2:2 10bit BT709 format. Encoding and decoding speed testing results are shown in Figure 1. These results show that HTJ2K has comparable encoding and decoding speed to ProRes HQ, while HTJ2K is much faster than J2K1.

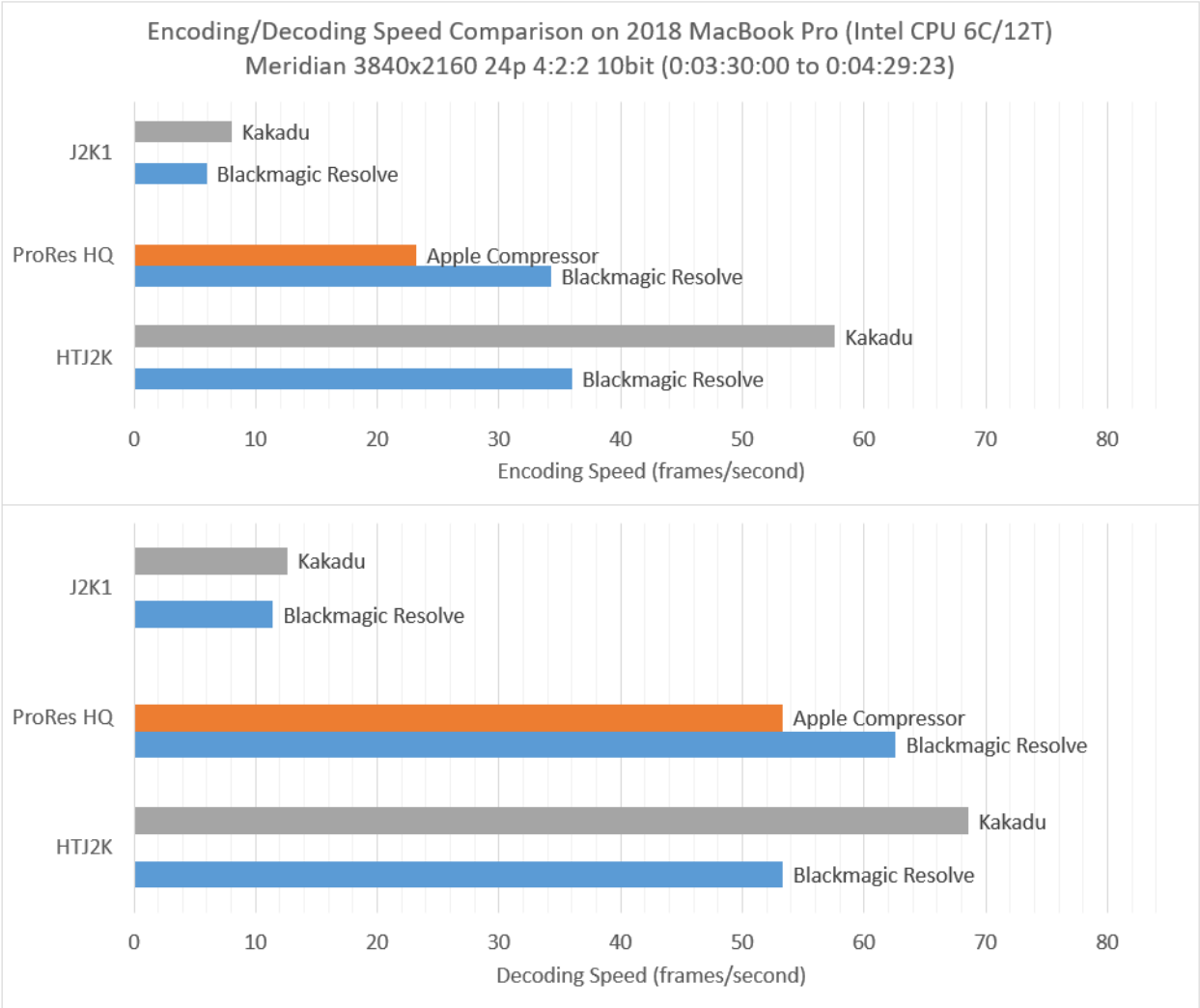

Figure 1: Encoding and decoding throughput test results

When performing high-performance speed testing, the time spent reading and writing uncompressed and compressed data to/from disk can itself become a bottleneck. Typical real-time capture applications store uncompressed imagery in memory that can be accessed directly by the encoder, while real-time playback applications store decoded data in memory not on disk. The Blackmagic Resolve and Apple Compressor speed results included reading/writing a QuickTime V210 10bit 4:2:2 uncompressed file (~32 GB), while the Kakadu speed results included reading/writing a YUV 16bit 4:2:2 uncompressed file (~48GB). With HTJ2K decoding at 68.6 frames per second, Kakadu's "kdu_vex_fast" results in decoded data being written to disk at 68.6 frames/second × (3840×2160 pixels/frame) × 2 samples/pixel × 2 Bytes/sample = 2,275,983,360 Bytes/second ~ 2.12 GB/second, which is close to the upper limits of the sustained write-rate of the NVM PCIe 3.0 1.0TB SSD included within the 2018 MacBook Pro testing platform. If the output file argument (-o) is not used with the

Kakadu “kdu_vex_fast” application, the software decodes the input file to memory instead of disk. Decoding the same HTJ2K file in this manner takes 11.1 seconds, which corresponds to 129.7 frames per second and is therefore almost twice as fast as decoding when writing the decoded image data to disk.

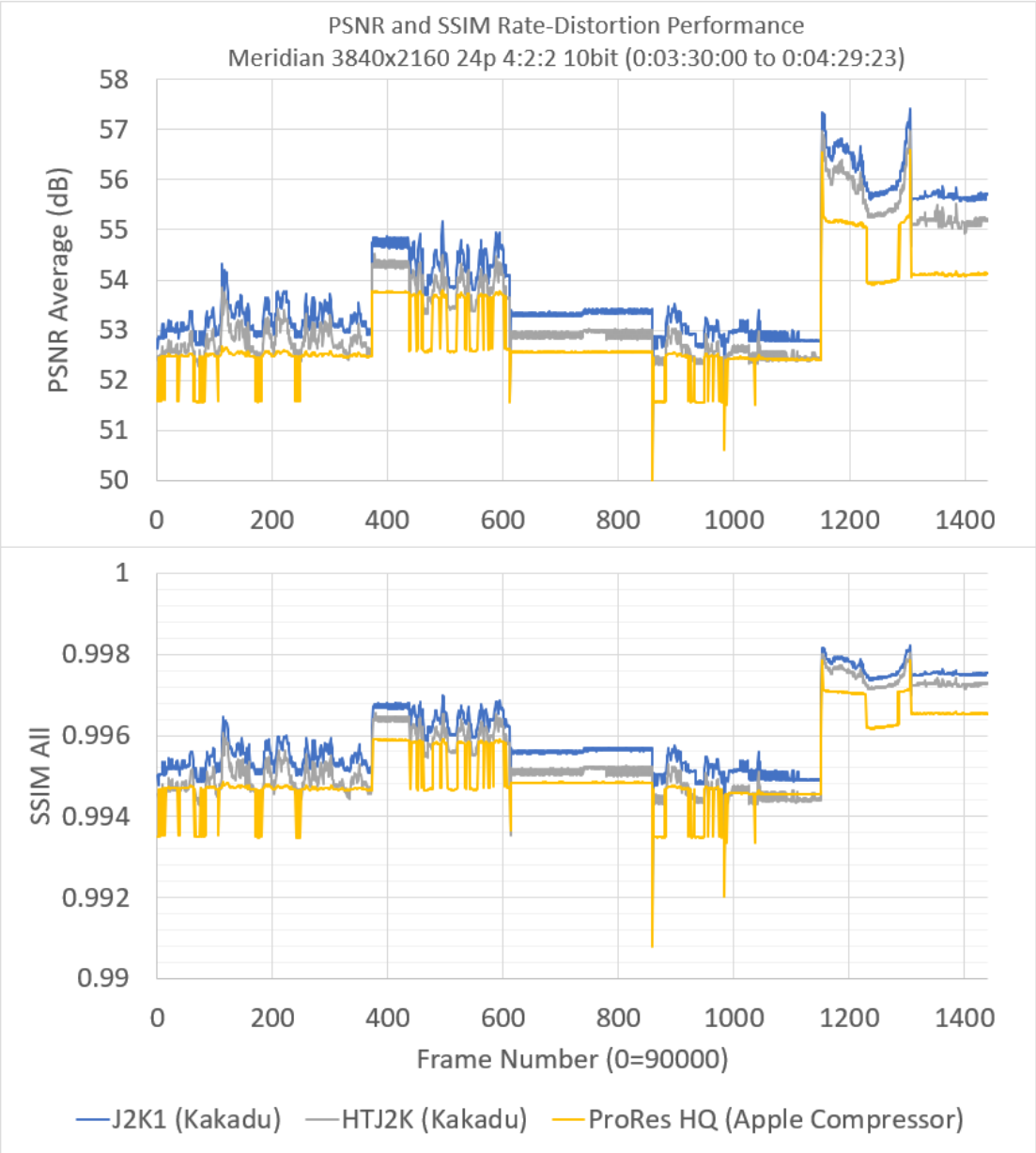

Figure 2: PSNR and SSIM performance

Rate-distortion performance of HTJ2K

Rate Distortion performance of HTJ2K, J2K1 and ProRes HQ was compared by first compressing 1 minute of Meridian with ProRes HQ using Apple Compressor 4.5.4, and then computing the actual data

rate of the output ProRes HQ file, which was determined to be 758Mbs. HTJ2K and J2K1 streams with the same bit-rate were then generated using Kakadu v8.2.1 "kdu_v_compress," with a constant coded length constraint for each frame[10]. The PSNR and SSIM results are shown in Figure 2. These results show that HTJ2K outperforms ProRes HQ by a similar margin to that by which J2K1 outperforms HTJ2K.

FFMPEG 5.0 was used to calculate these PSNR and SSIM results, which includes the $PSNR_{average}$ and $SSIM_{all}$ metrics that are shown in Figure 10. For 4:2:2 content, $PSNR_{average}$ is computed as

$$PSNR_{average} = 10\ log_{10}\left(\frac{1023^2}{MSE_Y/2 + MSE_{Cb}/4 + MSE_{Cr}/4}\right)$$

and MSE_Y, MSE_{Cb}, MSE_{Cr} are the mean square error of the luma and chroma components at their native sampling resolution. $SSIM_{all}$ is computed with a weighted average in a similar way, as:

$$SSIM_{all} = SSIM_Y/2 + SSIM_{Cb}/4 + SSIM_{Cr}/4$$

Conclusion

HTJ2K achieves a throughput at least comparable to ProRes while outperforming ProRes in rate distortion efficiency. In fact, in tests performed here, the HTJ2K throughput was limited by the throughput of the storage system used by the notebook computer.

By building on JPEG 2000, HTJ2K leverages existing code, features, infrastructure, and expertise. As evidence, there are now multiple independent implementations of HTJ2K, the majority of which are open-source. Examples include: OpenJPH[11]; OpenHTJS[12]; OpenHTJ2K[13]; MatHTJ2K[14]; Grok[15]; OpenJPEG[16]; the Kakadu SDK used for experimental results in this paper; and the formal HTJ2K reference software[17]. HTJ2K also supports many features not found in ProRes such as lossless coding and resolution scalability.

---

[1] https://dcimovies.com/

[2] https://doi.org/10.5594/SMPTE.ST2067-21.2020

[3] https://vsf.tv/download/technical_recommendations/VSF_TR-01_2018-06-05.pdf

[4] https://www.itu.int/rec/T-REC-T.814/en

[5] https://www.iso.org/standard/76621.html

[6] https://www.blackmagicdesign.com/products/davinciresolve/

[7] https://www.apple.com/final-cut-pro/compressor/

[8] https://kakadusoftware.com/

[9] Experiments were performed using a short, called *Meridian*, produced using a modern ACES production workflow with high-quality visual effects and post-production techniques was released by Netflix under the "Creative Commons" license and is available from https://opencontent.netflix.com/.

[10] HTJ2K encoding parameters "-rate 3.8109 Qstep=0.0009765625 Cmodes=HT Cblk={32,128} Cplex={6,EST,0.25,-1} -double_buffering 64 -no_weights" J2K encoding parameters "-rate 3.8109 Qstep=0.0009765625 Sbroadcast={5,single,irrev} Cblk={32,128} -double_buffering 64 -no_weights"

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

[12] https://www.npmjs.com/package/openht

[13] https://github.com/osamu620/OpenHTJ2K

[14] https://github.com/osamu620/MatHTJ2K

[15] https://github.com/GrokImageCompression/grok

[16] https://github.com/uclouvain/openjpeg

[17] https://gitlab.com/wg1/htj2k-rs