Cite as: Pearson, Glenn and Michael Gill, "An Evaluation of Motion JPEG 2000 for Video Archiving", Proc. Archiving 2005 (April 26-29, Washington, D.C.), IS & T (www.imaging.org), pp. 237-243. Keywords: Motion JPEG 2000, Video Archiving

An Evaluation of Motion JPEG 2000 for Video Archiving

Glenn Pearson and Michael Gill
Lister Hill National Center for Biomedical Communications
National Library of Medicine, NIH/HHS, Bethesda, MD

Abstract

Motion JPEG 2000 (MJ2) is one potential format for long-term video preservation. The format is attractive as an open standard with a truly lossless compression mode.

Currently, three software-only MJ2 implementations are readily available, from the Open JPEG 2000 project, from the Kakadu project, and (incorporating Kakadu) from vendor Morgan Multimedia. These are given a snapshot evaluation here. Among the findings: on a modern desktop machine, the Kakadu-based implementations can decode and deliver quarter-screen or smaller lossless-MJ2-encoded videos without frame drops. The newer Open JPEG 2000, while improving, is not yet competitive. All the implementations have practical limitations on acceptable input formats, and inadequate or missing audio support.

At higher image resolutions, playback without frame drops or reversion to lossy mode currently suggests hardware-based implementations. A practical impediment is limited availability of off-the-shelf board-level products.

Competing candidate file formats for video-editing, archiving, and delivery currently offer better-defined storage of metadata. Some formats, such as MPEG4/AVC, achieve better compression at the expense of some lossiness.

Introduction

Archiving Losslessly

Video archivists are keenly interested in techniques for long-term digital preservation on disk. In particular, consider the common case where the source material is not in digital form, but instead on film (to be scanned) or high-quality analog videotape (e.g., BetaCam SP). There is then a choice of destination digital format. A standardized format that reduces the storage costs of uncompressed video, but remains lossless, is attractive for preservation.

Motion JPEG 2000 for Video Archiving

Motion JPEG 2000 (MJ2), a video stream and file format, was standardized in 2002 as part of ISO/IEC's JPEG 2000 (JP2) standard[1,2,3,4], with subsequent refinements. This standard has been promoted by digital still camera manufacturers for its unified treatment of still and video compression. For stills, it is clearly of superior quality to its predecessor, JPEG, at any given compression[5]. MJ2 applies JP2 compression to each frame independently.

MJ2 is potentially attractive to video archivists not only because it is an open, international standard, but because it has a reversible, mathematically-lossless mode, not just the "virtually lossless" mode of certain other codecs.

These are early days for MJ2 implementations. Effort has concentrated on the MJ2 "Simple Profile", which has:

- a single video track, up to 30 frames/second (fps);
- an optional uncompressed mono/stereo audio track, interleaved with video;
- an optional still image;
- no references to media outside the file (i.e., self-contained);
- media data in temporal order.

Choosing MJ2 Encoder Settings for Archiving

When encoding, a number of parameters must be specified. The size, frame rate, and color encoding simply reflect the source material or encoder limitations. Other parameters are more open:

Number of Levels. The number of transform levels is one less than the number of resolutions in the hierarchy of wavelet decomposition. Table 1 shows suggested levels for various decoder "compliance points". Table 2 presents a proposed refinement by the Digital Cinema Initiative (DCI) to the JP2 codestream, which could be considered an extension and specialization of lossy MJ2. As we shall see, more levels give asymptotically better compression (and presumably scalability), but take longer to process.

Table 1. Aspects of Suggested Compliance Points ("Cpoints") for MJ2 Decoders [6]. A Cpoint-3 decoder is the most capable and ideally best performing. "Levels" is the minimum number of transform levels a compliant decoder will guarantee to process, so one might consider this a maximum when encoding. "Depth" is per color-space component, of which 3 is typical. Not shown: the limit for "Layers" at all compliance points is 15.

	Quarter Screen	Std. Video	HD Video	Digital Cinema
"Cpoint-…"	0	1	2	3
Height up to	288 pix.	576	1080	3112
Width up to	360 pix.	720	1920	4096
Depth up to	8 bits	12	12	16
Levels	3	4	5	5

Table 2. DCI Digital Cinema Distribution Master (DCDM) Requirements [7]. This differentiates digital cinema into "2K" and "4K" profiles and their projector decoders. There's a single tile and single layer. The 4K code stream is specially structured, so that a 2K decoder easily gets a 2K image. A gamma-corrected CIE XYZ color space is used.

	DCDM 2K	**DCDM 4K**
Frame rate	24 fps (or 48)	24 fps
Height up to	1080 pixels	2160
Width up to	2048 pixels	4096
Depth, Color	12 bits, X'Y'Z'	12, X'Y'Z'
Max. Levels	5	6

Number and Type of Layers. A "layer" is a quality level, typically expressed at encode time by a quality value or a compression rate. The highest level specified for a file (lossless in our case) impacts the filesize: it determines the bits per pixel stored and thus the maximum quality decodable. Providing a lossless layer implies use of the reversible integer 5/3 transform[1].

Additional layers of lesser quality, necessarily lossy, can be requested at encode time. Each such layer can be thought of as gathering up resources from several appropriate adjacent levels to express the bits per pixel needed for the stated quality. In practice, these layers act as hints to a decoder during real-time playback of where to stop as decoding time runs out for each frame, as an alternative to frame drops. For our evaluation here, we start from the posture that, for archiving, frame drops can be tolerated and the single lossless layer is enough, but revisit the issue later.

Number of Tiles. Images can be subdivided into tiles to ease transient memory loading. Tiling accommodates extremely large images, or handheld devices with minimal memory. A single tile seems fine for our application here.

Evaluation of Available Software-Only Motion JPEG 2000 Implementations

To date, we have looked at the three most-available software-only MJ2 implementations, and associated tools:

Kakadu [8]

David Taubman's JP2 implementation provides free executables (that we restrict ourselves to here) and licensable source code; a non-commercial license costs a few hundred dollars. MJ2 offerings are command-line functions kdu_v_compress and kdu_v_expand. Conversion is from or to a "vix" file: a Kakadu-specific text header with raw file appended; additional parameters are passed on the command line. YC_bC_r (colloquially known as YUV) and RGB planar raw formats are supported, with or without chroma subsampling. The still image viewer kdu_show does not support video, but a desire in that direction has been expressed.

Morgan Multimedia's Codec (MM) [9]

This French company sells an inexpensive, proprietary codec for MJ2 encode/decode on the Windows platform. Built around Kakadu, but sped up and enhanced, it takes the usual form of DirectShow and Video for Windows "filters". As a DirectX-compliant codec, it permits playback with, e.g., Windows Media Player, of native or AVI-wrapped MJ2 files. A property-page GUI, invocable from the taskbar or from within compliant video editors, allows user adjustments of parameters. The typical result of an editor invoking MM compression is an AVI-wrapped MJ2 file - a file with .avi extension and internal "fourcc" code (i.e., subtype) of "MJ2C"; in which a MJ2 bytestream follows an AVI header. The encoder accepts 4:4:4 formats RGB32, RGB24, RGB555, RGB565, and chroma subsampled YUY2, UYVY, YV12, and IYUV (aka I420)[10]. The last two are planar.

Open JPEG 2000 (OJ2) [11]

From the Communications and Remote Sensing Lab, Université Catholique de Louvain, Belgium, OJ2 provides open-source C-language implementations of JP2 and MJ2 for Linux and Windows. The MJ2 offering consists of two command-line conversion programs, "frames_to_mj2" and "mj2_to_frames", that convert respectively from and to raw YUV files, the only supported video format. (Additional utilities work with sequences of JP2 image files.) As with Kakadu, the compressor boasts a large set of command line options, most related to per-frame JP2 settings. We worked with the distributed binaries, plus a build with VC7/XP.

The Analysis

A brief quantitative analysis is made of each implementation's encode and decode performance, as well as degree of compression, and the effect of the number of levels on each of these. In addition, a qualitative look is taken at implementation shortfalls (e.g., audio, metadata), and interoperability.

Each analysis starts with a short headerless YUV video file. For convenience we began with a CIF[12]-sized file (288h x 352w), "Foreman" [13], a deinterlaced, 300 frame long, 30 fps, 4:2:0 subsampled clip often used in video evaluations. We also report early results with a 480h x 720w but otherwise technically similar clip, "Claps", a sequence of head and shoulder shots of individuals clapping. This was recorded at NLM on a 3 CCD miniDV camera, edited in Adobe Premier Pro 1.5, output as uncompressed AVI, then passed though the "avitoyuv" conversion utility. YUV file viewing (and repackaging of Foreman as an AVI file for MM testing) was done with the "Emily" [14] viewer.

Findings

All performance times were measured (n=1) on a single-CPU 3.19 GHz Pentium 4 Dell OptiPlex GX 270, with 512MB RAM and 1 GB pagesize, running Windows 2000 Pro. Default software parameters were used except as mentioned.

Performance

"Overall" times for OJ2 0.96 and Kakadu 4.3.2 derive from externally-measured process times, divided by total frames. Other times are based on reports by internal timers.

As its crisp performance indicates (Chart 1), Kakadu has been speed-optimized. Reported transform tuning for specific processors and instruction sets include Pentium/MMX, PowerPC/Altivec, and UltraSparc/VIS. The quarter-screen decode times are well below 33.3 ms/frame needed for 30 fps video without loss.

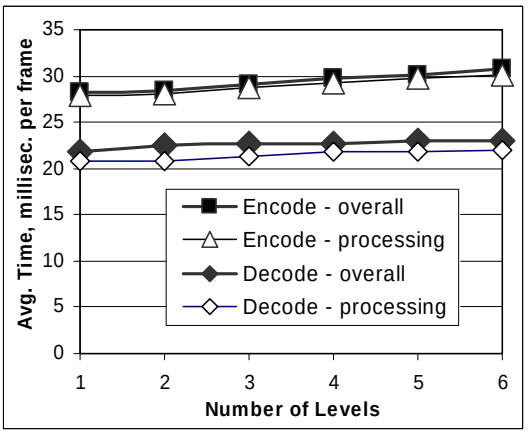

Chart 1. Kakadu Speed for Foreman Clip. "Processing" times for encode exclude input file reads, and for decode exclude output file writes. The latter were measured separately (not shown) and essentially account for the difference from Overall shown.

OJ2's code is recently produced, and clearly has not yet been tuned much for performance (Chart 2), although it is roughly 75% faster than the previous 0.95 release.

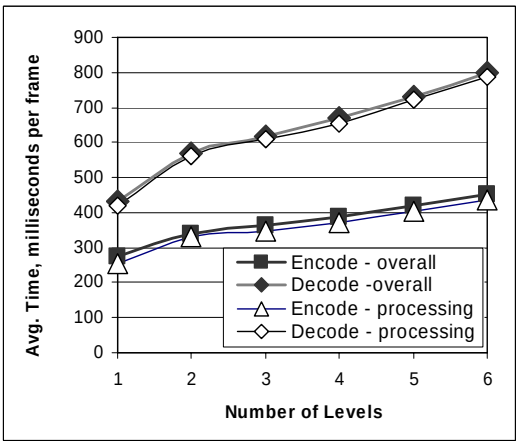

Chart 2. OJ2 Speed for Foreman. "Processing" times exclude file I/O. Shown here are tests with v 0.96 Windows binaries as distributed. (Source code was also compiled under VC7 and run.)

Next, we applied Kakadu to Claps, with 3.41 times the pixels of Foreman. A decode performance of around 73-79 ms/frame is what would be expected from proportionality. It's slightly better than that (Chart 3), perhaps because Claps compresses better. The performance is independent of level, except for a hint of a very shallow "U" relationship. The larger file size causes file I/O to consume a larger fraction of overall time, particularly for encoding.

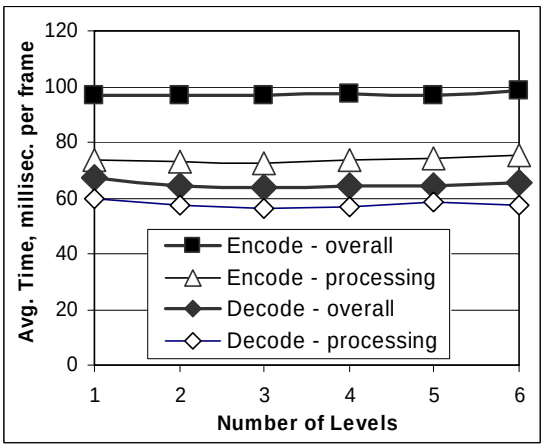

Chart 3. Kakadu Speed for Claps Clip. .

The performance of MM was assessed informally. Unlike Kakadu and OJ2, MM has a real-time requirement, including encoding within a video capture chain. To test encoding generally, Foreman was exported from Emily as an uncompressed RGB AVI file. A chain of A/V filters was built in GraphEdit[15] to read the file, split off any audio, convert the color space, then apply MM's encoding and file output. With 3 MJ2 levels, this process at full-speed (clockless) took about 9 ¼ s., as seen in a record of CPU and disk utilization captured with Windows PerfMon. (Future MM/GraphEdit tests might instead rely on an achieved-frame-rate field in the filters' property sheet.) This is roughly 31 ms per frame, consistent with Kakadu's performance of 29 ms in Chart 1. (The vendor claims that highly-compressed lossy operations, particularly encoding, are now tuned to be much faster than Kakadu or prior MM.). As for real-time decoding, if necessary MM (given no quick-lossy-layer alternative) drops frames. MM can't report drops, but subjectively, Foreman didn't show them. With Claps-size videos, Kakadu's 58 ms/frame in Chart 3, versus 33 ms/frame at 30 fps playback, implies dropping at least every other frame.

Degree of Compression

For a CIF-sized file (Chart 4), there are no compression benefits beyond 3 levels, and 2 is also acceptable for slightly faster decode time. Similarly, for a full-screen video, there are no benefits beyond 4 levels, and 3 are also good. Generalizing, one can recommend the number of levels given in Tables 1 and 2, or one less.

Kakadu creates smaller MJ2 files than OJ2. Speculatively, differences in default settings, amount of metadata stored, or spaced reserved before need is determined, might be contributing factors.

Number of layers has minimal effect on filesize. A separate OJ2 Foreman test where a half-dozen layers (including lossless) were encoded increased filesize 0.09% (with 3-level) and 0.15% (with 6-level) above 1-layer size.

Beyond these specific results, broad, uniform swatches of color compress much better than busy detail. Furthermore, as discussed later, spurious "detail" can be introduced by noise, film grain, or rapid interlaced motion.

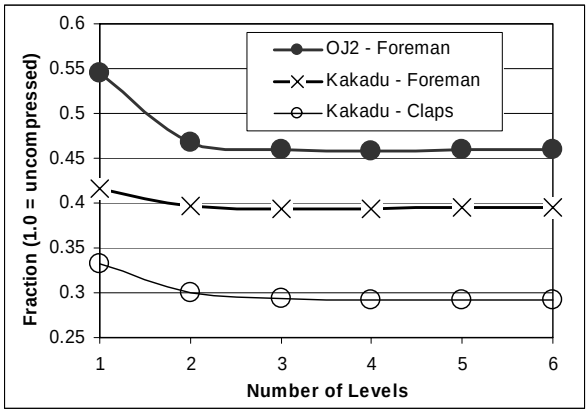

Chart 4. Kakadu and OJ2 Compression.

Other Limitations

The most troubling aspect was with audio. OJ2, oriented towards video research, has no native-within-MJ2 audio support. There is no evidence that Kakadu has, either, in spite of sufficient decode performance to make real-time playback plausible. Kakadu presumably supposes that some wrapper will be supplied if audio is desired. With MM, that wrapper is AVI. An AVI file can enclose audio (raw or compressed) and video streams independently, either abutted or interleaved or both, but synchronization can be an issue. MM is also said to support Simple Profile raw audio in native MJ2 files. We hope to probe this further, using Claps and digital samples captured from NLM's biomedical collection on BetaCam SP and older forms of analog tape.

Another problem to be alert for is a filesize limit. If a program loads its input file entirely into virtual memory at the outset (as OJ2 did prior to v 0.96), this typically prohibits a file greater than 2 GB, under current desktop Windows. Certain such filesize limits can sometimes be circumvented with a third-party "frame server", or by dealing with sequences of image files (a new OJ2 option).

The archivist, when digitizing video, should be aware of what raw input formats these three MJ2 implementations accept. While Kakadu accepts RGB and YUV color spaces, OJ2 is limited to YUV, and neither handles non-planar, per-pixel "packed" formats. MM takes in certain planar and packed formats, within an AVI container. All three products read and write encoded MJ2 files, but only MM does AVI.

Finally, a word about support. These offerings are academic or small-business products, backed by a small number of individuals. Kakadu and OJ2 both provide well-organized, substantial free documentation. Kakadu offers additional reference material with a paid license, and has the most active developers' forum. MM has a complex product line of similarly-named codecs (MJ2, JP2, Motion JPEG, LSI-MJPEG), with little specific information about MJ2.

Interoperability

Ideally, a file encoded to MJ2 with one of these products is decodable in another. Furthermore, a Windows codec like MM should allow playback in a media player, and encoding within a video editor (often an important component of digitalization workflow). We mention here some problems detected.

When we first attempted direct OJ2 file playback using Morgan codec version 1.40, a significant "fog" effect was seen (Figure 1). Version 2.00 of November, 2004, with further performance tuning, no longer exhibits this flaw. However, the first frame of the video is sometimes inverted, possibly due to a Microsoft player refresh bug.

Figure 1. Interoperability Problems Being Overcome. Foreman as he should appear (left) and did appear (right) until recent fix.

As for encoding AVI files with MM, while successful using GraphEdit, it was not using Adobe Premier Pro 1.5: MM surprisingly did not appear among other DirectShow codecs for movie exports. The vendor posits a color subsampling mismatch, and is actively addressing the issue.

Further Discussion

Playback Performance and the Role of Layers

Computer technology continues to advance, as traditionally expressed by Moore's law. Within a decade, this advance, even without further speedups to best-of-breed MJ2 implementations, is likely to allow real-time full-screen lossless MJ2 decoding without frame dropping on typical desktop machines. In the meantime, where the chief goal of encoding is not delivery, but rather long-term archiving, performance is a secondary concern, and frame-dropping during playback may be tolerable.

However, adding lossy quality layers can permit a smoother, more attractive playback with current-generation equipment. As indicated by results above, the filesize penalty for this is trivial. We hope to look further into what are the optimal number and spacing of such layers, given some projected distribution of playback environments. Note that low-quality levels, motivated by narrow bandwidth channels, are likely not of interest (unless an extractive server like Blitz described next is used.) This is because it makes little sense to transmit a large lossless file of which only a small fraction is used. Instead, one should separately encode and transmit a highly compressed file (not necessarily MJ2).

Alternative Hardware Approaches

At higher resolutions that digital TV, problems due to a lengthy decoding time might be solved by a hardware-based MJ2 system. For example, theater-based digital cinema[16] can have specialized hardware. Another example is an MJ2-server, such as the Sony "Blitz" system[17] recently shown at NLM, that streams media to remote clients, accommodating their bandwidth and processing limitations.

Behind any hardware solution is a JP2 chip, a number of which are available for volume incorporation into cameras. Not all hardware systems claim the throughput needed for real-time TV-resolution MJ2 video. Two that do are DSPWorx's chip pair, "Cheetah" and "Leopard"[18], and Amphion's circuit designs for "functional cores" within a system-on-chip, specifically a "CS6510" JP2 core paired with an on-chip embedded processor[19].

There is a paucity of off-the-shelf JP2/MJ2 PC boards. Analog Devices has evaluation boards (ADV202-SD, -HD) for its JP2 chip derived from Kakadu, but these are limited to "quantity one"[20]. Consequently, ambitious creators of high-level systems, like SAMMA[21], have had to prototype their own boards. The OJ2 project is moving towards letting its MJ2 software "wrapper" work with JP2 chips.

Retaining Metadata

Archivists seek to preserve a video's metadata. This may be video stream data such as 608/708B closed captioning[22]. Or it may be user-defined metadata. Video-editing file formats (e.g., OMF, GXF, MXF, AAF)[23] provide places within the file for user-defined metadata. The MJ2 standard also allows emplacing metadata. It provides great flexibility in such placement, but little guidance as to what to include and where. Further definitional work is needed at the standards level, for metadata interoperability among MJ2 implementations as well as metadata transfer to and from other file formats. Meanwhile, storing metadata such as captions outside the MJ2 file would seem prudent.

For JP2 digital still cameras, the situation is better: a recent ANSI standard[24] defines required and optional metadata about the camera, capture time and settings, image statistics, and GPS location. Text annotations (plausibly added with editing software) and audio are also supported.

Compression Improvement

For interlaced video, the MJ2 standard defines a choice of per-frame or per-field encoding. Per-field compresses better during rapid movement; otherwise per-frame is preferable. The per-field choice is beginning to appear in products, e.g., Morgan v 2.00. This early support applies that choice to the whole movie. Perhaps a smart encoder will evolve to make the best choice for each frame – a feature MPEG4/Advanced Video Codec (AVC)[25] offers as "Picture-adaptive frame/field coding" (PAFF). With ITU-R 601 video clips, PAFF compressed 15-20% better than AVC's per-frame-only mode[26]. (AVC has further fine-tuning for interleaving beyond MJ2, with slightly different compression algorithms for fields and frames. And a fourth option, MBAFF, picks the best choice of frame or field coding within fixed rectangles of each frame[27], to be ~15% better than PAFF[26].)

As mentioned, MJ2 can be mathematically lossless, avoiding any generational loss, unlike the "virtually lossless" modes of codecs like AVC. (In fairness, AVC does allow individual macroblocks to be passed unaltered and uncompressed, though this is not greatly desirable. Moreover, new AVC extensions include a "H444P" profile, only for unsubsampled 4:4:4 video. It has a lossless mode that skips the transform, but retains prediction and entropy coding. The result is said to be "fairly efficient" overall, combining "not the best" intraframe compression with the advantages of interframe prediction[28].)

Lossless MJ2 gives less compression than AVC's virtually-lossless quality level[29]. Much of this difference (beyond the lossiness itself) is likely due to AVC's interframe comparisons. To date, there has been an effort towards "product differentiation" between the work of the two ISO/IEC JTC 1/SC29 subcommittees, WG1's JP2/MJ2 and WG11's MPEG4, by restraining MJ2 to intraframe-only encoding. Perhaps this restraint should be lifted. While interframing is certainly harder to implement in cameras, editors, and players, it yields the long-term benefits of more efficient compression.

It is instructive to consider lossy AVC with and without interframing. One study[30] found that interframing generally achieved higher compression at a given quality level, except when the source was a high-resolution film scan (e.g., 4K horizontal, 35mm film); the film grain suppressed any interframing benefit. The benefit would emerge if the images were preprocessed to a much lower resolution by pixel-averaging. It seems likely that these findings would apply to lossless MJ2 with interframing. (The same study, comparing lossy MJ2 and I-frame-only AVC, found them similar when lightly compressed, with AVC-I sometimes having a slight edge.) Other experiments[31] with JP2-like wavelet codecs with interframing saw similar results.

More quantitatively, Imaizumi et al[32] built an experimental JP2-based software framework for lossless interframe comparisons, using JP2 to compress the difference frames (between actual and predicted image), and supplemented with motion-estimation vectors. Best settings delivered a 10-12.5% filesize reduction on 720 x 576 clips.

Conclusion

Lossless MJ2 has promise as an archival format, but more time is needed for implementations, such as those evaluated here, to be fully practical and convenient. Kakadu and MM have achieved real-time performance. Will archiving of analog video develop as a third application area for MJ2, beyond still camera video capture and digital cinema distribution? It may well, although there is a counter-current of activity from modern MPEG formats, which, while lossy, are high-quality. A lossless format in effect pays a cost in disk space, to avoid the labor costs of a Hollywood-style compressionist (with high-end software) to optimize a lossy format. This can be a reasonable trade off for a library.

Acknowledgements

Within NLM, Karen Steely, Leif Neve, Nancy Dosch, Jim Main, and Mike Detweiler helped with video material and processing. We appreciate correspondents at vendor sites, particularly OJ2's François-Olivier Devaux and MM's Guillaume de Bailliencourt. Thanks as well to Branch Chief George Thoma and others for their comments and support.

References

1. ISO/IEC Intl. Std. 15444, Information technology – JPEG 2000 image coding system, particularly Part 3: Motion JPEG 2000 (Sept. 2002, with subsequent amendments).
2. Michael D. Adams, The JPEG-2000 Still Image Compression Standard, N2412, ISO/IEC JTC 1/SC 29/WG 1. (Dec. 2002).
3. Jin Li, Image Compression: The Mathematics of JPEG 2000, Modern Signal Processing 46, pp. 185-221. (2003).
4. David Taubman, Michael Marcellin, JPEG2000: Std. for Interac. Imaging, Proc IEEE 90 (8), pp. 1336-57. (Aug 2002).
5. Diego Santa-Cruz, Touradj Ebrahimi, A Study of JPEG 2000 Still Image Coding versus Other Standards, Proc. X Euro. Signal Proc. Conf., Tampere, Finland, pp. 673+. (Sept. 5-8, 2000). http://jj2000.epfl.ch/jj_publications/papers/004.pdf
6. Part 3 draft Amendment 3, Definition of compliance classes and testing for Motion JPEG 2000. (Nov 2002). www.jpeg.org/public/15444-3fpdam3.doc .
7. ISO/IEC JTC1 SC29, Proposed draft amend. 1 to 15444-1: Profiles for Digital Cinema Applications, WG1 N3471. (Nov. 2004).www.itscj.ipsj.or .jp/sc29/open/29view/29n6379t.doc .
8. David Taubman/Univ. of New South Wale's Kakadu commericial C++ implementation of JP2000 Part 1; www.kakadusoftware.com. Copyright Unisearch Ltd.
9. Morgan Multimedia, Montpellier, France. www.morgan-multimedia.com. Some specs, personal communication.
10. See www.fourcc.org for AVI fourcc format descriptions.
11. The Open JPEG Project, U. Catholique de Louvain, Belgium, www.tele.ucl.ac.be/ PROJECTS/OPENJPEG.
12. "Common Intermediate Format", a ¼-screen NTSC- and PAL-friendly format from ITU H.261 videoconferencing std.
13. Short quarter-screen YUV video files were from http:// meru.cecs.missouri.edu/free_download/videos/
14. Nick Young, Emily 2004 YUV Viewer, http:// dmsun4.bath.ac.uk/resource/emily/emily.htm.
15. GraphEdit is distributed with Microsoft's DirectX SDK. Also useful: the GSpot AVI diagnostic (gspot@headbands.com).
16. Eric Edwards, Siegfried Foessel, JPEG 2000 for Dig. Cinema Appls. (Apr, 2001). www.jpeg.org/public/ DCINEMA-v2.pdf
17. Eisaburo Itakura, Hiryasu Furuse, Akifumi Mishima, Eric Edwards, A Single Source SNR/Resolution Scalable Video Delivery Sys., IS&T 2004 Archiving Conf., pg. 259. (2004).
18. DSPWorx, Cheetah chip, www.dspworx.com/cheetah.htm; www.dspworx.com/downloads/dsw2000s_pb. pdf.
19. Amphion, press release about JP2000 and other cores: www.amphion.com/news/news-100902.htm. See also: www.edtneurope.com/story/tech/ OEG20020717S0005-R.
20. Analog Devices, Inc., chip: www.analog.com/en/prod/0, 2877,ADV202,00.html. See also ADV202 evaluation boards.
21. System for Automated Migration of Media Archives (SAMMA), Media Matters, Inc., www.media-matters.net.
22. Caption stds. include EIA-608 (NTSC) & -708B (DTVCC). See Delivering Captions in DTV. (Oct. 2002). www. broadcastpapers.com/data/NCAMDTVCaptions-print.htm
23. OMF (Avid) and GXF (Grass Valley Group) are vendor-invented interchange formats. Vendor-neutral standards are MXF (MPEG-Pro, SMPTE) and AAF (AAF Assoc.).
24. ANSI/13A IT10.2000-2004, Digital Still Cameras – JPEG 2000 DSC Profile.
25. Gary J. Sullivan, Pankaj Topiwala, Ajay Luthra, The H.264/AVC Adv. Vid. Coding Std.: Overview & Intro. to the Fidelity Range Extensions, SPIE Conf. on Appl. of Dig. Im Proc XXVII.(Aug 2004). [FRExt includes High 4:4:4 Profile]
26. Gary Sullivan, Thomas Wiegand, Video Compr. - from Concepts to H.264/AVC Std, Proc IEEE, pp.1-13.(Dec 2004).
27. MPEG4 is a multipart ISO standard, promoted by MPEG Indus. Forum (www.m4if.org/resources.php). Of note: Pt. 10, AVC, aka ITU H.264. See also Pt. 15, AVC File Format.
28. D. Marpe, V. George, H.L. Cycon, K.U. Barthel, Perf. Eval. of Motion-JPEG2000 in comparison with H.264/AVC…., SPIE's Intl. Symp. on Photonics Tech. for Robotics, Automation, and Manuf.; Wavelet Appl. in Industrial Processing. (Oct. 2003).
29. Til Halback, Mathias Wien, Concepts & Perf. of Next-Gen. Video Compr. Standardization. Proc. Nordic Signal Proc. Symp. (NORSIG), aboard Hurtigruten. Norway. (Oct. 2002).
30. Michael Smith, John Villasenor (ICSL/UCLA), Intra-frame JPEG2000 vs. Inter-frame Compression Comparison, SMPTE Tech. Conf., Pasedena, CA (Oct 2004). www.smpte.org/ conferences/146sescomp.cfm; www.conferencemediagroup. com/detail.asp?product_id=SM-04-02-02 for audio.
31. Jens-Rainer Ohm, Mihaela van der Schnaar, John W. Woods, Interframe Wavelet Coding – Motion Picture Representation for Universal Scalability, EURASIP Signal Proc.: Image Comm., Special issue on Digital Cinema. (2004). (www.ece. ucdavis.edu/~mihaela/IC_DCspecial_InterframeWavelet.pdf
32. Shoko Imaizumi, Ayuko Takagi, Hitoshi Kiya, Lossless Inter-frame Vid. Coding using Ext. JPEG2000, Proc 2002 Int. Tech Conf Circ/Sys, Comp & Comm, Phuket,Thai. (Jul 2002). www.kmutt.ac.th/itc2002/CD/pdf/19_07_45/ FA1_PJ/7.pdf.

Biographies

Glenn Pearson has a Ph.D. in Computer Science from the University of Maryland, College Park. Since 1997, he has been with MSD, Inc., Vienna, Virginia, developing imaging workflow and presentation software for NLM, part of NIH within the Department of Health and Human Services. His most recent project involves video streaming and archiving.

Michael J. Gill is an electronics engineer with the Communications Engineering Branch of LHNCBC, a research and development division of NLM. His research interests include image transmission, communications systems, and performance measurement. He is a Senior Member of the IEEE and received his BS EE from the University of Maryland College Park.