ELSEVIER

Signal Processing: *Image Communication* 17 (2002) 131–144

SIGNAL PROCESSING: IMAGE COMMUNICATION

www.elsevier.com/locate/image

File format technology in JPEG 2000 enables flexible use of still and motion sequences

J. Scott Houchin[a,*], David W. Singer[b]

[a] *Eastman Kodak Company, Rochester, NY, USA*
[b] *Apple Computer, Cupertino, CA, USA*

Abstract

While there exist many different image file formats, the JPEG committee concluded that none of those formats addressed a majority of the needs of tomorrow's complicated imaging applications. Many formats do not provide sufficient flexibility for the intelligent storage and maintenance of metadata. Others are very restrictive in terms of colorspace specification. Others provide flexibility, but with a very high cost because of complexity. The JPEG 2000 file format addresses these concerns by combining a simple binary container with flexible metadata architecture and a useful yet simple mechanism for encoding the colorspace of an image. This paper describes the binary format, metadata architecture, and colorspace encoding architecture of the JPEG 2000 file format. It also shows how this format can be used as the basis for more advanced applications, such as the upcoming motion JPEG 2000 standard.

Keywords: JPEG 2000; JP2; File format; Metadata; XML; Color; ICC; Moving pictures; Streaming; MPEG

1. Introduction

For several years, the JPEG committee (ISO/IEC JTC1/SC29/WG1) has been working on a next generation image compression algorithm called JPEG 2000 (ISO/IEC 15444-1).[1] The compression algorithm itself is wavelet based and promises to provide many new image access features to applications. To practically use those features, however, the codestream must be combined with all of the information describing that image; this requires the use of a file format in addition to the codestream syntax. Therefore, as part of the JPEG 2000 standard, the committee has defined a pair of tightly linked standard file formats. The minimal/simpler variation is called JP2 [6] and is targeted at applications that are limited to the storage of RGB and grayscale images. The JPX [6] format expands the capabilities of the standard to allow for the specification of other types of images (both photographic and non-photographic) and defines a set of standard metadata fields. The JPX file format also allows for the specification multiple images within a single file, and for the combination of those images through compositing and animation. While using these formats is optional for conformance, either applications are strongly encouraged to use the JP2 or JPX file formats where appropriate.

*Corresponding author. Tel.: +1-716-588-8495.
E-mail addresses: scott.houchin@kodak.com (J.S. Houchin), singer@apple.com (D.W. Singer).

[1] ISO/IEC 15444-1, Information technology—JPEG 2000 image coding system.

0923-5965/02/$ - see front matter
PII: S0923-5965(01)00023-6

There are three main aspects of the JP2 and JPX file formats:

- binary container,
- colorspace specification,
- metadata embedding.

As described in the following sections, the file formats provide a strong foundation and feature set for today's applications. In addition, it is well understood that new applications are created every day, and that these file format must continue to meet the needs of tomorrow's applications for many years. As such, these formats are built upon a strong foundation with well-defined means for extension. This extension mechanism will also be described.

Note that at the time of writing, neither the JPEG 2000 standard nor the Motion JPEG 2000 standard have been finalized; therefore, aspects may be changed as the standards are finalized. The final standards, not this paper, should be taken as the definitive references.

2. Why two file formats?

In a world that already has too many choices, it may come to the surprise of the developer that the JPEG 2000 standard defines not one, but two file formats for use in "single-image" type applications. The reason for this is very clear when one considers the state of the industry with respect to the largest group of potential users: consumers.

To provide a file format that is useful for a large number of target applications and industries, that file format would have to have a very large feature set. For example, commercial graphic arts applications require a format that can contain CMYK image data as well as the ICC profiles that describe that data. Similarly, remote sensing applications require support for many channels of image data. The problem is that it is not practical to implement a reader that can be used in every application area, and thus there will be situations in which a particular JPEG 2000 file reader cannot open a particular JPEG 2000 file. While many industries have learned to deal with such interoperability problems, this type of situation cannot be tolerated on the consumer desktop. Thus, it was important to define a compliance point at which all readers (including those targeted at the consumer desktop) could read all files written to that compliance point. This compliance point is the JP2 file format. All readers are required to read all JP2 files. Applications that require features greater than those defined as part of the JP2 must use the JPX file format.

It is important to note, however, that many aspects of the JPX file format are compatible with inclusion in JP2 file, and thus it is possible (and appropriate in many situations) to create JPX files that can be completely read by a simpler JP2 reader.

3. Binary container

At the highest level, the JP2 and JPX file formats are simply binary containers. They provide a mechanism to encapsulate both the compressed codestream and any metadata required to describe the decompressed data (such as the colorspace). However, the binary container also specifies how applications can gain access to the data embedded within the file. To truly meet the needs of today's and tomorrow's applications, that binary container must provide quick access to the individual elements, yet not overburden applications that desire to modify or add new elements to the file.

3.1. File structure

Conceptually, a JP2 file contains a sequence of boxes. Each box encapsulates one element of the file, such as the codestream or a group of metadata fields. Many of those boxes directly contain the data embedded in the file. Other boxes, called superboxes, just contain more boxes, providing only grouping and higher level meaning to other pieces of data. The binary structure of a box is as follows (Fig. 1).

The file is a sequence of boxes. Each box then contains header and contents fields. The header of the box starts with a 4-byte length followed immediately by a 4-byte type.

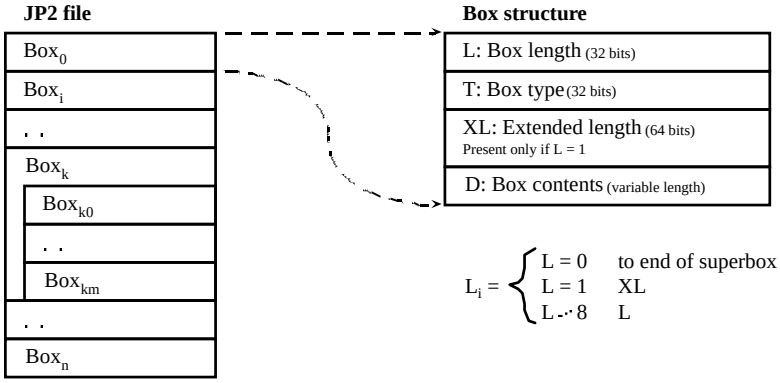

Fig. 1. File and box structure.

The length field of the box, L, specifies the amount of data in the box, in bytes, including all box header fields. Because the length of the box includes the length and type fields, the smallest legal box size is 8 bytes. Values of the L field smaller than 8 are used to indicate special cases. If the value of L is 0, then the box contains all data up to the end of the superbox, or to the end of the file for top-level boxes (boxes not contained within any other superbox). For example, if the value of L in Box_n in Fig. 1 were 0, then Box_n would go to the end of the file. If the value of L in Box_{km} were 0, then Box_{km} would go to the end of Box_k.

If the value of L is 1, the box may be longer than can be represented by a 4-byte field. In this case, the extended length field immediately follows the box-type field. This 8-byte field specifies the actual length of the box.

The type field specifies the kind of data found within the contents of the box. Generally, box types are represented by four ASCII[2] characters rather than integer codes.

Once the length of the box is determined, the sequence of boxes in the file can be parsed, even if the format of the contents of the individual boxes is unknown. Fig. 2 shows how the box lengths can be used to jump from box to box.

As shown in Fig. 2, the length of each box includes any boxes contained within that box. For example, the length of Box 1 includes the length of Boxes 2–4, in addition to the header fields for Box 1 itself. In this case, if the type of

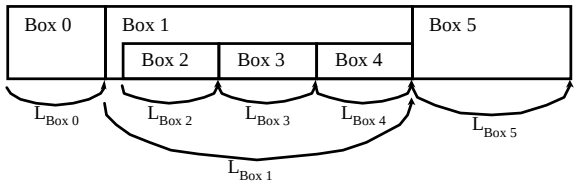

Fig. 2. Jumping from box to box.

Box 1 was not understood by a reader, it would not recognize the existence of Boxes 2–4 because they would be completely skipped by jumping the length of Box 1 from the beginning of Box 1.

This box structure is what provides for the extension of the simple JP2 file format into more complex formats, such as the JPX file format. For example, there are two additional file formats planned at this time, each building upon the frameworks defined as part of the family:

- JPM enables the storage of compound images using multiple compression types across a single scene.
- Motion JPEG 2000 enables the storage of sequences of individual image frames (such as burst sequences from digital cameras or digital cinema).

3.2. Boxes defined in the file format

The JP2 file format defines several standard boxes. These boxes encapsulate the standard and

mandatory information needed by most applications to properly parse the file:

- The signature and application profile boxes allow this file to be recognized as being part of a JPEG 2000 family of files. In addition, the application profile box allows each file to specify what readers can properly interpret from the file. For example, a JPX file may indicate that it can be correctly interpreted by a simpler JP2 reader (to the extent defined within the JP2 standard).
- The header box contains the standard image header information, such as image size, resolution, bit depth, and colorspace.
- The contiguous codestream box contains the JPEG 2000 compressed image data.
- Metadata boxes allow vendor-specific and non-image data to be embedded within the file.

Many of these boxes will be discussed in the following sections. However, for the definitive description of the boxes in a file, see the JPEG 2000 standard (see footnote 1) itself.

3.3. Building on the binary container

While the file format does define a minimal set of features, the flexibility of the binary container allows a great deal of extensibility without affecting compatibility with a JP2 reader (or any of the other JPEG 2000 family readers). Because each box contains a type value, applications can easily recognize important boxes and skip unknown boxes. In addition, there are very few requirements on the order of defined boxes. In JP2 files, the Signature and File Type boxes shall be at the beginning of the file, the header box will be found at some point after those boxes, and the codestream box will be found somewhere after the header box. Putting data between those boxes is permitted and encouraged as required by a particular application. The JPX file format reduces those restrictions for the header and codestream boxes.

In the past, most digital image file formats allow a writer to create an optimized file only for a small number of applications. While those formats are often used in other applications, they may be used non-optimally due to restrictions within the image file format. It is clear that new applications are created every day, and that it is not possible to imagine today how images will be used in five years. Thus, it is necessary that flexibility be built into the baseline system. By allowing a great deal of flexibility, the standard allows for the creation of files optimized for a much wider range of applications.

However, it is also vitally important that files can be shared between different applications, and thus, a high level of interoperability is necessary. By using the box framework, we allow application specific optimizations to be done in an interoperable way, with minimal barriers to adoption.

4. Specification of colorspace

One of the chief jobs of a digital image file format is to tell an application how to interpret the samples of the image, and one of the primary aspects of this is the specification of the colorspace. Given the importance of the specification of color, and the magnitude of the problem introduced by incorrect interpretation, it is unfortunate that most digital image formats do not allow an application to precisely specify the colorspace of an image. When past file formats have allowed for a more precise specification of color, the flexibility in selecting a colorspace for encoding of the image data was often very limited.

The JP2 and JPX file formats change this situation by adopting a thorough and precise architecture for colorspace specification that still allows for a high degree of flexibility.

This architecture is built around three major requirements:

- that the colorspace of an image shall be precisely and accurately specified within the digital image file,
- that the creator of an image shall be allowed to select the colorspace of the image such that the image file best meets the needs of the target application,

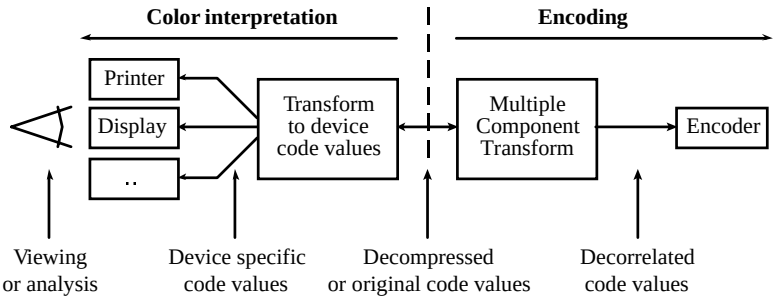

Fig. 3. Division of color architecture in JPEG 2000.

- that any reader of that file shall be able to properly interpret the colorspace of the image and correctly display that image.

Obviously, these are lofty goals, and practicality must be kept in mind when designing a system based on such requirements. However, the color architecture in the JP2 file format makes great strides in resolving these requirements into a system that is practically implementable in any system (as required by the overall interoperability goals of the JP2 file format). Application requirements that are not fully met by the JP2 file format will rely on the JPX file format. This prevents many interoperability problems by guaranteeing that all JP2 files can be read by all JP2 readers, while still providing a means by which the extended features can be used.

There are two major parts of the color specification architecture in the JPEG 2000 standard: specification of encoding and specification of interpretation, as shown in Fig. 3.

This architecture clearly differentiates the steps that were taken by the file writer in order to maximize compression efficiency (encoding) from those steps that must be taken by a file reader in order to properly use the image (interpretation).

4.1. Color encoding

The color encoding half of the architecture represents those steps that the file writer took to prepare the image data to be compressed using the JPEG 2000 algorithm. In general, this involves a transform that mixes the components to decorrelate the data. The decorrelated data is then compressed on a component-by-component basis.

4.1.1. Baseline encoding (JP2)

In JPEG 2000, encoding, through the multiple component transform, can be specified on a tile-by-tile basis, and thus is specified as part of the codestream syntax, not the file format. Part I of the standard defines two multiple component transforms.

The irreversible component transform (ICT) is better known as the YC_bC_r transformation from baseline DCT-based JPEG. It transforms data in an RGB colorspace to a luminance–chrominance form and is useful in lossy compression applications. Once the components have been transformed and compressed, the chrominance components can be truncated to improve compression with minimal loss of quality.

The reversible component transform (RCT) provides decorrelation of RGB data in situations where lossless compression is required. Once the components are decompressed, the reverse RCT can be applied to get back to the original code values. Obviously, if the chrominance components were truncated, there would still be loss associated with the image.

While these two transforms do not meet the needs of all applications, they are generally sufficient for digital photography and Internet imaging applications.

4.1.2. Extended encoding (JPX)

The JPX file format (along with extensions to the compressed codestream as defined in Part II of

the standard) extends the encoding architecture to allow for two new capabilities. The first new feature is the ability to use a custom multiple component transform. This would allow an encoder to generate an optimal multiple component transform for a particular image, thus improving quality or increasing compression. The second new feature is the ability to shape or gamma encode the image data as part of the encoding process. For example, linear encoded data is required in many high-end compositing applications; however, it compresses very poorly. The file reader would extract the inverse shaper or gamma value from the codestream and use that value to transform the image data back to its original form.

One possible image processing chain for extended encoding is shown in Fig. 4.

4.2. Color interpretation

The color interpretation half of the architecture represents those steps that a file reader must take to properly print, display or prepare the image for analysis. In general, this involves a transform that converts the decompressed (and recorrelated) code values into a colorspace designed for use with the target output device or analysis. The JP2 and JPX file formats define several methods by which the colorspace of the image can be specified. These methods balance the required flexibility with the barriers to adoption and interoperability between readers.

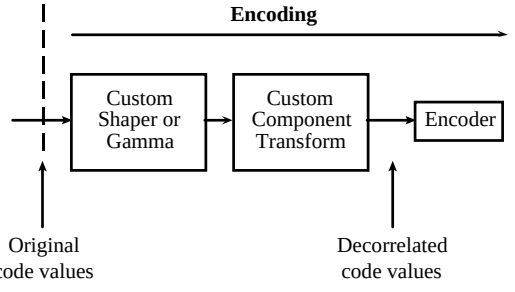

Fig. 4. Encoding a color image using extended encoding options.

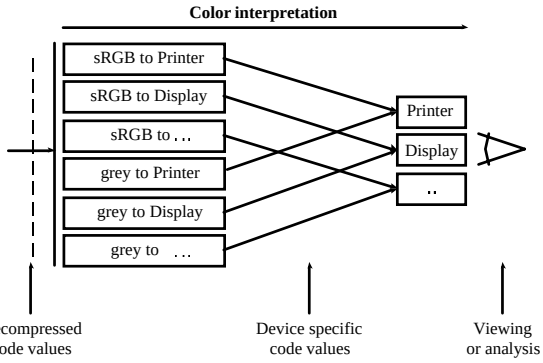

Fig. 5. Interpreting the colorspace of an image using the enumerated method.

4.2.1. The enumerated method

Like many other digital image file formats, the JP2 and JPX formats define a list of standard colorspaces and assigns integer codes to represent those spaces. For an image encoded in one of those standard spaces, the file would embed that integer code into the file (the enumerated method). While there are a large number of known and commonly used colorspaces, each enumerated colorspace must be natively understood by the reader, and thus the complexity of a reader implementation is somewhat proportional to the number of colorspaces that must be understood through enumeration. The image chain for the enumerated method is shown in Fig. 5. Note that each input to device color transform must be natively known to the application (or how to generate it must be known).

To ensure that the JP2 format is practically implementable in all applications, the set of enumerated spaces is restricted to sRGB [3] and a grayscale space related to sRGB [3] for JP2 files. Other spaces, such as the ITU CIELAB, are expected to be standardized as extensions in the JPX file format. The JPEG committee also intends to provide a mechanism by which vendors can register and use other standard colorspaces.

4.2.2. The any ICC method

Another common way to specify the colorspace of the image is to embed an ICC profile[2] in the file

[2] Coded character set—7 bit, American Standard Code for Information Interchange, ANSI X3.4–1986.

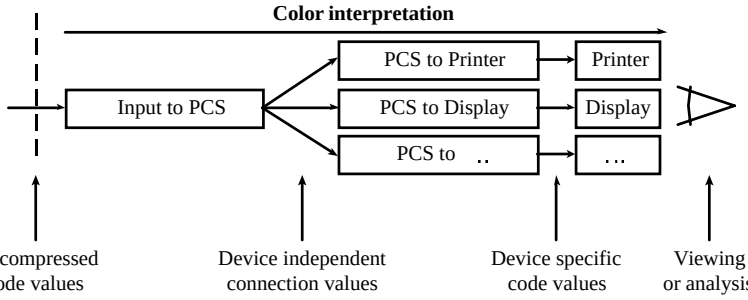

Fig. 6. Interpreting the colorspace of the image using the any ICC or restricted ICC method.

(the any ICC method). This profile (an input profile) specifies the transformation between the decompressed code values and the profile connection space (PCS). To convert to a device specific colorspace, the input profile is combined with an output profile (a profile that specifies the transformation from the PCS to the device specific colorspace). The decompressed code values are then processed through the combined profile by an ICC color management engine. The image chain for the any ICC method is shown in Fig. 6.

However, some ICC profiles are very complex, containing multiple 1D-LUTs, matrices and 3D-LUTs. In many applications, it is impractical or impossible to implement a fully compliant ICC color management engine. As such, the any ICC method is expected to be defined as part of JPX in Part II, not in JP2, because it is very important that all JP2 readers be able to read all JP2 files.

4.2.3. *The restricted ICC method*

The major problem with the enumerated method is lack of flexibility. While sRGB is appropriate for some applications, other colorspaces are needed in other applications. For example, RGB images targeted at wide-gamut output devices should be stored in a wide-gamut colorspace, such as ROMM RGB [7]. Other images, such as those captured by scanning a consumer 35 mm negative, require the storage of information whiter than a perfect diffuse white reflector (a consumer negative may contain information up to 30,000% white). For those images, a colorspace such as ERIMM RGB [7] would be appropriate.

However, as noted above, it is impractical to enumerate a large number of spaces, as the complexity of a JP2 reader increases with the number of required enumerated colorspaces. The generic use of ICC profiles does not provide the answer either, as a complete ICC color management engine would represent barriers to adoption in many applications.

However, the ICC profile format specification [2] does define two classes of profiles. These classes, three-component matrix-based input and monochrome input profiles, can be implemented very easily. Those transforms contain at maximum three 1D look-up tables and a 3×3 matrix. These two profile classes allow for the specification of a very large number of RGB and grayscale colorspaces.

The restricted ICC method for specifying the colorspace of an image in the JP2 format uses any profile conforming to one of these two classes of profiles. That profile is embedded into the JP2 file. Applications reading the JP2 file have two choices for interpreting the color of the image:

- Extract the profile and use any compliant ICC color management engine to transform the image into a colorspace appropriate for the desired output device.
- Extract the profile, extract the look-up tables and matrix from the profile, and transform the image using application specific code. If this choice is made, it is important to note that the transformation specified within the profile transforms the decompressed code values to the profile connection space. The application must combine that transformation with the appropriate transformation for generating values appropriate for the desired output

device. Additional information for performing such a transformation is contained within the JPEG 2000 standard itself.

Note that the image chain for the restricted ICC method is identical to that of the any ICC method as shown in Fig. 6. The difference is that the Input to PCS transform is guaranteed to be of low complexity.

4.2.4. The vendor color method

While the use of ICC profiles does provide a very high degree of flexibility, there are two drawbacks to those methods:

- It is a non-trivial process to match a profile against a set of known profiles.
- ICC profiles are generally limited to photographic applications.

Because of these drawbacks, the JPX format is expected to define a fourth method, called the vendor color method, to allow any application in any domain to define a "shortcut-code" to represent a particular colorspace definition. This code is stored as universally unique ID (UUID); a 16-byte number that when generated correctly, should be unique "throughout the universe". This can be done without involving either the JPEG committee or another third party.

4.2.5. Using multiple colorspace specification methods

It is important to note that there is often more than one way to specify the colorspace of an image. For example, a well-defined colorspace can be specified through enumeration or can be specified through an ICC profile (which technically specifies the default transformation from that colorspace to the PCS, not the colorspace itself). However, while these multiple methods often produce identical results, they are optimized for different applications. Each of the defined methods has pros and cons:

- The enumerated and vendor color methods allow for quick recognition, but the application must recognize the code and natively understand how to transform the image data to the relevant device colorspaces. In addition, colorspaces defined in Part II of the standard or by registration will not be understood by a conforming JP2 reader.
- The any ICC method allows almost any colorspace to be precisely and accurately specified in a generic way, but can be impractical or impossible to support in some applications.
- The restricted ICC method allows for many RGB and monochrome spaces to be specified in a generic way that is practically implementable in all applications, but approximations may need to be made in order to represent the transformation to the PCS within the profile restrictions. However, the performance of a system (in terms of decoding speed) using restricted ICC profiles will be higher than a system using the more complex profiles.

When creating optimal files for a particular application, the writer may be able to choose one best method for specifying the colorspace of the image. However, many files must also be interoperable outside that particular application, and performance issues may need to be considered. When considering target optimization, interoperability, and performance, a balance will often only be found by using multiple representations of the colorspace of the image.

The JP2 and JPX file formats allow for any number of different colorspace methods to be used within a single file, allowing the file writer to address these three issues, provided that all of those methods are "equivalent". In fact, it is expected that many JPX files will contain multiple methods by default.

For example, consider the registration of standard and vendor colorspaces in Part II. In most cases, it is desirable to create files that can be read by any reader (and thus meet the conformance requirements for a JP2 reader). This can be a problem, because the new colorspaces are not understood by older and simpler readers. If interoperability is indeed required, the file writer must use the restricted ICC method to specify the colorspace of that image. However, an application that has an optimized processing path for the new colorspace would need to compare the ICC profile byte for byte with a reference profile. In the best case, this step is justa nuisance. In the worst case,

such as with scene spaces such as ERIMM RGB [7], this step is not possible, as there may not be a specific reference profile, because the profile embedded within the file must specify the desired rendering (look) of the image in addition to the colorimetric transformation.

In these cases, it is very desirable to also embed the enumerated code for that new colorspace in the file (at the insignificant cost of around 15–27 bytes).

In another example, the "best" transformation from the decompressed code values to the PCS requires the use of 3D look-up tables, which unfortunately reduces the performance of the system. In this example, the application desires to provide a "quick-and-dirty" user-selectable setting, indicating that the application should approximate the colorspace transformation to improve performance, at the expense of quality. In this example, the file would contain both the best ICC profile, using the any ICC method, and a less complex profile, using the restricted ICC method.

It is expected that some applications will have requirements from both examples, and thus use all three colorspace methods within the same file, as shown in the image chain in Fig. 7.

5. Encapsulating metadata within a JP2 file

In many applications, much of the value of an image is derived from metadata. For example, the emotional value of a picture of your children is increased if you remember when and where the picture was taken. In other cases, metadata can be used to provide enhanced processing on the image. This may allow an application to improve an image such that it much more closely reflects the photographers' memory of the scene.

The JP2 file format provides two mechanisms for embedding metadata into a file:

- XML boxes allow XML [9] documents to be embedded within a JP2 file. The meaning of the data (and the definition of the XML elements) is specified through the XML document type definition (DTD).
- UUID boxes allow binary data to be embedded within the file. The contents of each UUID box starts with a UUID [4]. This number is generated by the application developer when the format of the data to be contained in the box is determined.

Both the XML and UUID methods allow vendors to add metadata to image files in such a way as to not impact readers who do not understand that data. Also, when properly written, there is very little chance that one vendor's data will be mistaken for another vendor's. Additionally, XML and UUID boxes may be placed almost anywhere in the file, allowing the writer to place the metadata where it is most appropriate for the target application.

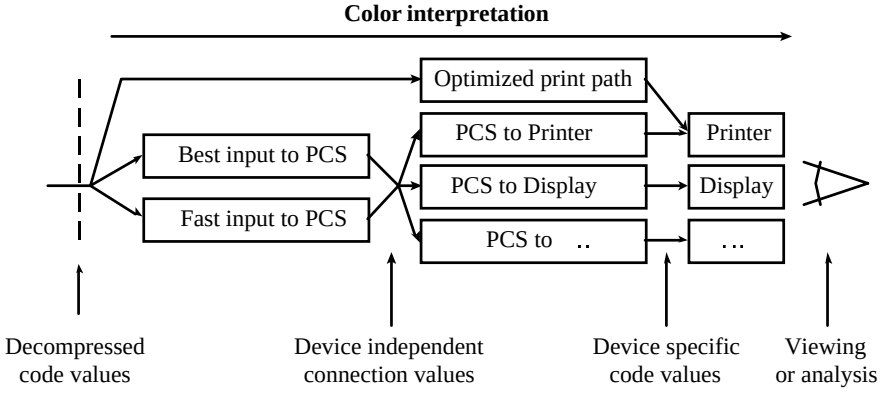

Fig. 7. Color interpretation image chain with multiple options.

6. Compositing and animation in JPX

The other novel new feature of the JPX file format is the ability to specify multiple images and the methods by which those images shall be combined through either compositing or animation. While this is similar to the capabilities of the motion JPEG 2000 format, it is targeted at simpler applications that do not include synchronized sound or real-time frame rates.

The building block of this architecture in JPX is the compositing layer. A compositing layer represents the combination of one or more JPEG 2000 compressed codestreams. This allows, for example, the RGB color channels of the layer to be stored separately from the opacity mask that will be used to combine the layer with the other layers (and thus facilitate separate editing of the color data from the opacity data).

Each compositing layer is then combined, in turn, as specified through a set of instructions. Those instructions allow for traditional Porter-Duff compositing, cropping, scaling, and animation. For example, the instructions are able to specify the animation of a single compositing layer over a static background.

7. Extending the file format architecture to new application domains

As shown in the above sections, the JP2 file format architecture provides a strong foundation for extension. In many applications, the format can be extended in ways that are transparent to a standard JP2 reader.

The existing JPEG standard has been widely used to handle motion sequences, particularly in two applications areas. The first is in editing applications, where coding systems that perform frame differencing can make frame-accurate editing difficult, yet the size (and bandwidth requirements) of uncompressed video make such interframe codecs impractical. The second area is in consumer applications, where a JPEG codec is already available. For example, several digital still cameras permit the recording of short-motion sequences.

Historically, there has not been a standard file format for the interchange of motion data. This has led to a variety of formats being used in the marketplace. Worse, there was not a standard for the use of the JPEG codec in motion sequences, which led to incompatible variants being used in the marketplace. A standard format to contain motion JPEG 2000 sequences will help to avoid these problems.

Finally, it is often desirable that a motion sequence has other multimedia information presented at the same time: an audio track with the video, or time-based metadata. To do this, a general multimedia container is needed.

The JPEG committee sought to adopt a file format, which would be upward compatible with the JP2 still-image format described above, and be a flexible container for multimedia data in general, including, but not limited to, motion JPEG 2000.

Another committee in ISO, the MPEG committee, when developing the MPEG-4 specification, had also recognized the advantages that would accrue from a standard format. They desired a format that could be used to support content creation (editing), local playback, and streaming from streaming servers using a variety of protocols. The MP4 format[3] that resulted from this work serves all these functions, and is able to contain a variety of multimedia data. The MP4 format uses the same object (box) structures as the JP2 format.

By taking the MPEG-4 file format, and defining how JPEG 2000 motion sequences may be stored within it, the MJP2[4] format inherits these benefits, as well as providing the industry with a common file format for multimedia data, including JPEG 2000, MPEG-4, and, it is expected, MPEG-7 metadata in the future.

This section details some of the key design aspects of the file format used for motion JPEG 2000, the MJP2 format. The MJP2 format is based on, and structurally compatible with, the MP4 format used in MPEG-4. It, in turn, derives much of its design and approach, and is largely

[3] The MP4 file format: ISO/IEC 14496-1 subpart 4.

[4] ISO/IEC 15444-3, Information technology—JPEG 2000 image coding system: motion JPEG 2000.

compatible with, the QuickTime file format [8] as used in QuickTime, including concepts introduced in QuickTime 4.

At the time of writing, the motion JPEG 2000 standard was not complete; therefore, aspects may have changed as the standard was developed. The final standard, not this paper, should of course be taken as the definitive reference.

7.1. Conceptual structure

The conceptual structure of a media file is that it is a movie composed of tracks, which are logically parallel in time. In each track, there is a sequence of samples (for example, a video frame is a sample). Each sample has associated with it: timing, size, and position in the file.

The physical structure (layout) of a media file is that it is a sequence of boxes, as in JP2. The data structures that describe the movie, tracks, and samples are kept in a single box, the movie box. The actual compressed video and audio (the sample data itself) is kept separately.

The movie box is able to refer to other files by URLs. The sample data can be stored within the same file as the movie box, or in these other files. Within the movie box, the temporal sequence of samples (e.g., video frames) is fully defined with, for each sample:

- its duration,
- its size,
- its position within its enclosing file.

These three pieces of information allow the temporal ordering (sequence) of the samples to be de-coupled from their storage sequence. This means that during editing, for example, each track in the presentation can be stored in a separate file, and the editing can proceed merely by updating the timing information in the movie box. This can be saved to disk in a file without samples in it quite rapidly. However, when the presentation is complete, a different save can be done: the movie box written first in a file, and then the samples of each track, in time-order, and interleaved. Such a file serves well for local playback and interchange. It can also be used for delivery over download protocols such as HTTP. Once the movie box has downloaded, and enough samples collected, playback may proceed as long as the protocol is delivering data as fast as it is being played. The file does not need to be downloaded completely before being played.

7.2. Representation of JPEG 2000 data

Motion sequences of JPEG 2000 data store complete, unaltered JPEG 2000 codestreams as the individual frames of video. This means that an existing encoder or decoder can be used for motion purposes. In addition, the JP2 header structures can be used to describe the images. Therefore, all the advantages and uses of that header—for colorspace management, for example—can be used for motion sequences.

This is done in a compact fashion. The designers of the format recognized that many frames of video would normally share a common description. The file format stores a set of descriptions; each frame of video is associated with one of the descriptions, by its index into the set. The description identifies the coding scheme used (and thus the decoder required)—for example, JPEG 2000, or MPEG-4 Visual—and provides any parameters needed by that decoder.

By re-using both the codestream syntax and the header syntax, the design of both the file format, and software to create and read it, is simplified.

7.3. Composition

In the motion JPEG 2000 file format, JPEG 2000 is presented as a "peer" coding system to MPEG-4 visual. Those aspects of the file format that are held in common have a common definition. Motion JPEG 2000 sequences can thus be represented as a simple timed sequence of images, without involving MPEG-4 technology beyond the file format.

However, it is optionally permissible to use aspects of MPEG-4 in motion JPEG 2000 files. One such area concerns powerful visual and audio composition.

The basic file format permits simple audio and visual composition. In audio, this consists simply of assigning a volume and left/right balance to

each audio track within the file, and requiring that the tracks be mixed for output.

In visual tracks, simple layered composition is provided. Each track has an assigned *layer*, transformation *matrix*, and graphics *composition mode*. The layer provides a composition ordering for the visual tracks, from the back to the front. The matrix allows simple 2D transformations of the visual information, as it is rendered.

The composition modes allow for simple copying, single-color transparency (''blue-screen''), and basic alpha composition, using an alpha plane within the compressed image data.

If more advanced composition is required, the MPEG-4 BIFS (binary information for scenes) system can be used. This provides both audio and visual composition and interaction, in both 2D and 3D spaces.

Temporal composition is also possible. All of MJP2, MP4, and QuickTime permit the use of ''edit lists'', which provide an explicit mapping of the track's timeline into the timeline of the movie. ''Blank'' segments of time can be inserted, and sections of the track can be moved in time, or reused. This is another area where the updating of small tables during editing can facilitate rapid work.

7.4. Other media

It is rare that simple motion video is all that is desired in a presentation. Still cameras may be able to record audio as an ''annotation'' to single pictures, or in parallel with motion sequences. As has been noted above, more complex time-based composition may be desired and time-based metadata may be needed, leveraging the emerging MPEG-7 standard.

All these are permitted in the MJP2 specification. At the simplest level, raw audio in either 8-bit or 16-bit (DV compatible) format is permitted. This allows for simple audio/visual sequences. As with composition, more complex support for audio may optionally be used from MPEG-4. MPEG-4 audio includes a number of tools: speech and general-purpose (''music'') codecs and synthetic audio from either text-to-speech or a structured audio system for making music.

There is simple support for metadata and extension information within the file format (an overall copyright notice, or vendor-specific boxes). MPEG-4 also provides simple content information streams. Perhaps the most promising developments in this area come from MPEG-7, which is a general scheme for defining and handling metadata associated with a media composition. The MPEG-7 standard is expected to work with MPEG-4 media and within MP4 files; thus, its use with motion JPEG 2000 should also be possible. This will permit time-based, grammar-defined metadata to be used in an open, extensible, and interoperable way.

7.5. Streaming support

A popular and important way to deliver media uses the concept of ''streaming''. In a file-based delivery, a presentation is downloaded to a client, which plays it from the file. Though the file may be played as it is downloaded (which can be done with MP4 and MJP2 files, if properly structured), this is not ''true'' streaming: the client system must still buffer the file, and a reliable transport be used. The rate of file transfer is decoupled from the natural rate of the media: the download may happen faster, or slower, than the rate needed to play the media. If the user requests a ''rewind'' and review of previous material, it is locally available and such seeking can be rapid.

True streaming involves sending the media to the client at the playback rate. The client receives the media incrementally, buffers only small quantity, and then decodes, displays, and discards it. If the user requests a ''rewind'' and review of previous material, it must be re-requested from the server.

There are a number of streaming protocols in use today. Standard ones include MPEG-2 transport, the protocol suite from the IETF that includes real-time protocol (RTP) [1], and real-time streaming protocol (RTSP) [5]. A number of vendors have proprietary protocols also.

All of MJP2, MP4, and QuickTime support both modes of delivery, and the support for streaming is designed to be protocol neutral. In

MJP2, support for file-based playback is required, and streaming support is optional.

Often a streaming protocol is supported by a streaming server, which uses a file format specific to that protocol. This is often a way to achieve the best performance, by careful tuning of the format, and the placement of media on disk systems. However, in the formats discussed here, protocol-independent support was desired.

Each streaming server is ideally decoupled from intimate knowledge of the media it is serving. Such intimate knowledge can be an impediment to efficient operation. For example, the RTP protocol requires that if a video frame from a macroblock-based codec (e.g., H.261) is split into several packets, those packets should start at a block boundary, and contain enough information for a decoder to decode that frame fragment even if the preceding fragment is lost. Scanning media streams at the time of serving can slow a media server down.

Both these problems are addressed by the concept of "hint" tracks. Hint tracks are extra tracks added to a file, which instruct the server in the process of transmitting media data for a specific protocol. Just as, for example, visual tracks can contain MPEG-4 visual, or JPEG 2000, so the design of hint tracks permits their definition for, for example, RTP, and MPEG-2 transport.

A sample in a hint track consists of packet-formation instructions for a media packet (protocol data unit). Instructions might include sending "immediate" data from the hint track (e.g., a packet header), or extracting data from an associated media track (e.g., a visual track containing motion JPEG 2000). By following these instructions, the server is able to packetize data correctly and efficiently without necessarily being aware of the details of the codec or its packetization scheme. Instead, an offline process is run that provides the "bridge" between media and server, which adds these hint tracks to the file.

In this way, authoring tools are media aware, but protocol and server unaware. They write the media data into the file in its "natural", unfragmented state. Servers, on the other hand, are essentially media unaware, but obviously protocol aware. This decoupling permits great flexibility in technology. As new protocols are developed, hint track formats for those protocols, and the accompanying hinter software can be developed, and existing media assets streamed over those protocols from the new servers supporting them.

The hint tracks, because they can reference data in the media tracks, need not greatly expand the size of the file (10% growth is common for RTP hint tracks). Indeed, by using the file references described above, the hint media file containing the hint tracks can be distinct from the media files containing the compressed media itself. Finally, because the media is retained in its original, protocol-unaware format, any media file can be locally played, or indeed re-edited and re-hinted. In this way, life-cycle management is possible.

7.6. *Multifunction files*

The discussion so far has focused on the use of MJP2 and MP4 functions. However, because of the common box (or atom) object structure of these files, it is possible to make multifunction files that contain, for example, both a still image and a motion sequence. The still image might be a "preview", or the image that will be printed, for a motion sequence.

Each of these standards defines that unrecognized objects should be ignored. Thus, a still image reader may be able to find its header and image within a file, which also contains a movie and image sequence.

However, when such a general tool is used, there is a danger: a file may have only a single "type" (e.g., file-name extension or mime-type), and as a result, readers may be uncertain as to whether they can read a file or not. For example, a still image reader may be able to read a motion file—if a still image is present.

To avoid scanning the entire file, the header of the file contains information, provided by the file writer, on which standards the file adheres to.

8. Conclusion

As shown, the JP2 file format provides a strong foundation and set of tools for describing images.

It is also extensible, allowing applications to tailor and optimize JP2 files for that particular application, while still preserving interoperability across a large number of application domains.

In addition, because of the tight integration with QuickTime and MPEG-4 binary structures, the system provides a platform for delivering both still and motion imagery, along with any associated data (such as sound or image descriptions) within one binary container. While not without risks, this integration maximizes interoperability between still and motion imaging and reduces the implementation burdens in systems that handle both types of media.

References

[1] S. Casner et al., RTP: A transport protocol for real-time applications, RFC 1889.

[2] International Color Consortium, ICC profile format specification. ICC.1:1998-09, <http://www.color.org/ICC-1_1998-09. PDF>.

[3] International Electrotechnical Commission, Colour management in multimedia systems: Part 2: Colour management, Part 2-1: Default RGB colour space—sRGB, IEC 61966-2-1 1998, 9 October 1998, <http://w3.hike.te.chiba-u.ac.jp/IEC/100/PT61966/parts/> or <http://www. sRGB.com/>.

[4] ISO/IEC 11578:1996 Information technology—open systems interconnection—remote procedure call, <http://www.iso.ch/cate/d2229.html>.

[5] A. Rao et al., Real-time streaming protocol (RTSP), RFC 2326.

[6] J.S. Houchin, SPIE 4115–48, JPEG 2000 file format provides flexible architecture for color encoding, Appl. Digital Image Process. (San Diego, CA) XXIII (3 August 2000), pp. 476–483.

[7] K.E. Spaulding, G.J. Woolfe, E.J. Giorgianni, Reference input/output medium metric RGB color encodings (RIMM/ROMM RGB), in: Proceedings of PICS 2000 Conference, Portland, OR, 26–29 March 2000.

[8] The QuickTime file format specification <http://www.apple. com/quicktime/resources/qtfileformat. pdf>.

[9] W3C, Extensible Markup Language (XML 1.0), Rec-xml-19980210, <http://www. w3. org/TR/REC-xml>.