TECHNICAL REPORT

ISO/IEC TR 29199-1

First edition
2011-07-15

Information technology — JPEG XR image coding system —

Part 1: System architecture

Technologies de l'information — Système de codage d'image JPEG XR —

Partie 1: Architecture du système

Reference number
ISO/IEC TR 29199-1:2011(E)

Contents

Page

Foreword v
1 Scope 1
2 Terms and definitions 1
3 Abbreviations 6
4 The JPEG XR image coding system 7
5 General overview of technical design 7
5.1 Basic technology structure 7
5.2 Supported image format types 8
5.3 Decoded image structure and interpretation 9
5.4 Data processing hierarchy and structures 10
5.5 The JPEG XR transform structure and hierarchy 11
5.6 Handling of image and tile boundaries 13
5.7 Quantization and lossless representation 13
5.7.1 Overall quantization design concepts 13
5.7.2 Quantization control on a spatial region basis 14
5.7.3 Quantization control on a frequency band basis 14
5.7.4 Quantization control on a colour plane component basis 14
5.7.5 Quantization control type combinations 14
5.8 Prediction of transform coefficients and coded block patterns 14
5.9 Adaptive ordering of coefficient scanning pattern 15
5.10 Entropy coding of transform coefficients 15
5.11 Codestream structure 16
6 JPEG XR design in relation to baseline JPEG and JPEG 2000 17
6.1 General 17
6.2 Image area partitions 18
6.3 Image fidelity refinement 18
7 High dynamic range (HDR) image coding 18
7.1 HDR formats supported in JPEG XR 18
7.2 HDR signal processing design in JPEG XR 19
7.3 Examples of HDR applications for JPEG XR 19
8 JPEG XR profiles and levels 19
8.1 Overview of profiles and levels 19
8.2 Sub-Baseline profile 20
8.3 Baseline profile 20
8.4 Main profile 20
8.5 Advanced profile 20
8.6 Levels 21
9 JPEG XR encoding practices 21
9.1 General encoding guidelines 21
9.2 Encoding for random access 21
9.3 Guidelines for tile size selection 22
10 The JPEG XR decoding process functionality 23
10.1 JPEG XR decoding process structure 23
10.2 Output colour conversion 24
10.3 Resolution scalability at decoder 24
10.3.1 General 24
10.3.2 DC-only image decoding 24

10.3.3 DC plus LP image decoding 24
10.4 Quality scalability at decoder 25
10.5 Spatial random access at decoder 25

11 JPEG XR codestream compressed-domain manipulation 25
11.1 General 25
11.2 Flexbits trimming 26
11.3 Flexbits and HP band elimination 26
11.4 Flexbits and HP and LP band elimination 26
11.5 Spatial versus frequency codestream mode switching 26
11.6 Rotation and flip 26
11.7 Compressed-domain region of interest extraction 26
11.8 Switching between interleaved and planar alpha planes 27
11.9 Compressed-domain retiling 27

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

In exceptional circumstances, when the joint technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example), it may decide to publish a Technical Report. A Technical Report is entirely informative in nature and shall be subject to review every five years in the same manner as an International Standard.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC TR 29199-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC TR 29199 consists of the following parts, under the general title *Information technology — JPEG XR image coding system*:

— *Part 1: System architecture* [Technical Report]

— *Part 2: Image coding specification*

— *Part 3: Motion JPEG XR*

— *Part 4: Conformance testing*

— *Part 5: Reference software*

Information technology — JPEG XR image coding system —

Part 1: System architecture

1 Scope

This document is a non-normative Supplement | Technical Report providing a technical overview and encoding and decoding practice guidelines for the JPEG XR image coding system as normatively specified in ITU-T Rec. T.832 | ISO/IEC 29199-2, ITU-T Rec. T.833 | ISO/IEC 29199-3, ITU-T Rec. T.834 | ISO/IEC 29199-4, and ITU-T Rec. T.835 | ISO/IEC 29199-5.

2 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

2.1
adaptive coefficient normalization
parsing sub-process where **transform coefficients** are dynamically partitioned into a **VLC**-coded part and a **fixed-length coded** part, in a manner designed to control (i.e., "normalize") bits used to represent the **VLC**-coded part

NOTE The **fixed-length coded** part of **DC coefficients** and **low-pass coefficients** is called **FLC refinement** and the **fixed-length coded** part of **high-pass coefficients** is called **flexbits**.

2.2
adaptive inverse scanning
parsing sub-process where the **zigzag scan order** associated with a set of **transform coefficients** is dynamically modified, based on the statistics of previously-parsed **transform coefficients**

2.3
adaptive VLC
parsing sub-process where the code table associated with **VLC** parsing of a particular **syntax element** is switched, among a finite set of fixed tables, based on the statistics of previously-parsed instances of this syntax element

2.4
alpha image plane
optional secondary image plane associated with an image of the same dimensions as the luma component of the primary image plane

NOTE The alpha image plane has one component, a luma component.

2.5
block
m×n array of **samples**, or an m×n array of **transform coefficients**

2.6
block index
integer in the range 0 to 15 identifying, by its position in **raster scan order**, a particular 4×4 **block** within a partition of a 16×16 **block** into 16 4×4 **blocks**

2.7
byte
sequence of 8 bits

2.8
chroma
component of the **primary image plane** with non-zero index, or the **transform coefficients** and sample values associated with this **component**

2.9
codestream
sequence of bits contained in a sequence of **bytes** from which syntax elements are parsed, such that the most significant bit of the first **byte** is the first bit of the **codestream**, the next most significant bit of the first **byte** is the second bit of the **codestream**, and so on, to the least significant bit of the first **byte** (which is the eighth bit of the **codestream**), followed by the most significant bit of the second **byte** (which is the ninth bit of the **codestream**), and so on, up to and including the least significant bit of the last **byte** of the sequence of **bytes** (which is the last bit of the **codestream**)

2.10
component
array of samples associated with an **image plane**

2.11
context
possible value of a specific instance of a **context variable**

2.12
context variable
variable used in the **parsing process** to select which data structure is to be used for the **adaptive VLC** parsing of a given syntax element

2.13
DC coefficient
first subset when the **transform coefficients**, that are contained in a specific **macroblock** and a specific **component,** are partitioned into 3 subsets

2.14
DC-LP array
array of all DC and low-pass **transform coefficients**, for all **macroblocks** associated with a specific **component**

2.15
decoder
embodiment of a **parsing process** and **decoding process**

2.16
decoding process
process of computing output sample values from the parsed syntax elements of the **codestream**

2.17
dequantization
process of rescaling the quantized **transform coefficients** after their value has been parsed from the **codestream** and before they are presented to the **inverse transform process**

2.18
encoder
embodiment of an **encoding process**

2.19
encoding process
process of converting source sample values into a **codestream**

2.20
file
finite-length sequence of **bytes** that is accessible to a **decoder** in a manner such that the **decoder** can obtain access to the data at specified positions within the sequence of **bytes** (e.g. by storing the entire sequence of **bytes** in random access memory or by performing "position seek" operations to specified positions within the sequence of **bytes**)

2.21
file format
specified structure for the content of a **file**

2.22
fixed-length code (FLC)
code which assigns a finite set of allowable bit patterns to a specific set of values, where each bit pattern has the same length

2.23
FLC refinement
fixed-length coded part of a **DC coefficient** or **low-pass coefficient** that is parsed using adaptive **fixed-length codes**

2.24
flexbits
fixed-length coded part of the **high-pass coefficient** information which is parsed using adaptive **fixed-length codes**

2.25
frequency band
collective term for one of the following three subsets of the **transform coefficients** for an **image**, which are separately parsed: **DC coefficients**, **low-pass coefficients**, and **high-pass coefficients**

2.26
frequency mode
codestream structure mode where the DC, low-pass, high-pass and **flexbits frequency bands** for each **tile** are grouped separately

2.27
hard tiles
codestream structure mode where the overlap operators are not applied across tile boundaries; instead, boundary overlap operators are applied at tile boundaries

2.28
high-pass coefficient
third subset, when the **transform coefficients** that are contained in a specific **macroblock** and a specific **component** are partitioned into 3 subsets

2.29
image
result of the **decoding process**, consisting of a **primary image plane** and an optional **alpha image plane**

2.30
image plane
collective term for a grouping of the **components** of the **image**

2.31
internal colour format
colour format associated with the spatial-domain samples obtained through the **inverse transform process** and the **sample reconstruction process**, and distinguished from the **output colour format** associated with the **output formatting process**

2.32
inverse core transform (ICT)
two steps of the **inverse transform process** that involve processing of **transform coefficients** associated with each **macroblock** independently, with no **overlap filtering**

2.33
inverse transform process
part of the **decoding process** by which a set of **dequantized transform coefficients** are converted into spatial-domain values

2.34
inverse scanning
process of reordering an ordered set of parsed **syntax elements** from the **codestream** to form an array of **transform coefficients** associated with a specific **component** and **macroblock**

2.35
low-pass coefficient
second subset, when the **transform coefficients** that are contained in a specific **macroblock** and a specific **component** are partitioned into 3 subsets

2.36
luma
component of an **image plane** with index zero, and the **transform coefficients** and sample values associated with this **component**

NOTE Although this term is commonly associated with a signal that conveys perceptual brightness information, as used in this International Standard the term is primarily an identifier of a particular array of samples or **transform coefficients** for an **image**.

2.37
macroblock
collection of **transform coefficients** or samples, across all **components**, that have the same indices *i* and *j* with respect to a **macroblock partition**

2.38
macroblock partition
partitioning of each **component**, into 16×16, 8×8, or 16×8 **blocks**, depending on the **internal colour format**

2.39
output bit depth
representation, including the number of bits and the interpretation of the bit pattern, used for the sample values of the output **image** that are the result of the **decoding process**

2.40
output colour format
colour format associated with the output **image** that is the result of the **decoding process**

2.41
output formatting process
process of converting the arrays of samples (that are the result of the **sample reconstruction process**) into the output samples that constitute the output of the **decoding process**

NOTE This specifies a conversion (if necessary) into the appropriate **output colour format** and **output bit depth**.

2.42
overlap filtering
steps of the **inverse transform process** that involve processing of **transform coefficients** across adjacent **blocks** and **macroblocks**

NOTE When **overlap filtering** is applied, it is applied across **macroblock** boundaries as well as **block** boundaries. When the **codestream** uses **soft tiles**, the **overlap filtering** is also applied across **tile** boundaries. Otherwise, **overlap filtering** does not occur across **tile** boundaries.

2.43
parsing process
process of extracting bit sequences from the **codestream**, converting these bit sequences to syntax element values, and setting the values of global variables for use in the **decoding process**

2.44
prediction
process of computing an estimate of the sample value or data element that is currently being decoded

2.45
primary image plane
image plane that consists of all **image components** that are not a part of the **alpha image plane**

2.46
quantization parameter (QP)
value used to compute the scaling factor for the **dequantization** of a **transform coefficient**, before the **inverse transform process** is applied

2.47
raster scan order
scan order in which a two-dimensional array of values is scanned row-wise from left to right, and the rows are scanned from the top row to the bottom

2.48
refinement
process of modifying a predicted or partially-computed **transform coefficient**

2.49
run
number of zero valued coefficient levels that precede a non-zero valued coefficient level in the **zigzag scan order** during the **inverse scanning** process

2.50
sample reconstruction process
process of converting dequantized **transform coefficients** into samples of the **image**

2.51
soft tiles
codestream structure mode where the overlap operators are applied across tile boundaries

2.52
spatial mode
codestream structure mode where the **DC**, **low-pass**, **high-pass** and **flexbits frequency bands** for each specific **macroblock** are grouped together

2.53
spatial transformation
element in the **codestream** indicating the preferred final displayed orientation of the decoded **image**, as specified in ISO/IEC 29199-2

NOTE The **spatial transformation** is only a suggestion, and **decoder** conformance is checked only for the decoded **image** prior to the application of this transformation (i.e. for orientation 0).

2.54
start code
bit pattern that specifies the beginning of a **tile packet** or other distinguished, contiguous set of syntax elements in the **codestream**

2.55
tile
collection of **macroblocks** that have the same indices *i* and *j* with respect to a **tile partition**

NOTE Each **tile** corresponds to the **macroblocks** for a rectangular region of the **image**.

2.56
tile packet
contiguous subset of the **codestream**, which contains the coded **syntax elements** associated with a specific **tile**

2.57
tile partition
partition of the **image** into rectangular arrays of **macroblocks**, as specified in ISO/IEC 29199-2

2.58
transform coefficients
values, associated with each specific **macroblock** and specific **component**, that — after **dequantization** — form the input arrays into the **inverse transform process**

2.59
variable-length code (VLC)
code which assigns a finite set of allowable bit patterns to a specific set of values, where each bit pattern is potentially of a different length

2.60
zigzag scan order
adaptive ordering for the **inverse scanning** process, which assigns array indices to each subsequent **transform coefficient** parsed from the **codestream**

3 Abbreviations

For the purposes of this document, the following abbreviations apply.

CBP	Coded Block Pattern
CIE	Commission Internationale de l'Éclairage (International Commission on Illumination)
DCT	Discrete Cosine Transform
FLC	Fixed-Length Code
HDR	High Dynamic Range
HP	High-Pass
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
JPEG	Joint Photographic Experts Group
LBT	Lapped Bi-orthogonal Transform
LP	Low-Pass
QP	Quantization Parameter
ROI	Region of Interest
VLC	Variable-Length Code
XR	eXtended Range

4 The JPEG XR image coding system

The JPEG XR image coding system enables the compressed representation of imagery for a broad range of applications, including support for an extended range of capabilities (e.g., relative to that of the baseline sequential JPEG encoding specified in ITU-T Rec. T.81 | ISO/IEC 10918-1) while minimizing computational resources and memory storage requirements.

The design includes support for a wide range of image representation formats, rapid local region access, and various scalability features including multi-resolution frequency scalability, a quality scalability enhancement layer at the highest resolution, and embedded codestream support for both lossy and lossless image representations using the same algorithmic processing elements. In particular, the JPEG XR design architecture includes support for requirements specific to high dynamic range imagery applications.

The design application focus for JPEG XR includes digital photography and associated workflows. However, the actual intended range of applications for the technology is broad. JPEG XR also has core codestream features that can be used to support usage scenarios such as interactive image usage in networked system environments.

The JPEG XR image coding system consists of the Specifications listed in Table 1.

Table 1 – Specifications of the JPEG XR image coding system

Subtitle	ITU-T Specification	ISO/IEC Specification	Normative? (Y/N)	Summary of content
System architecture	T.Sup2	29199-1	N	An overview of JPEG XR and its usage (this document)
Image coding specification	Rec. T.832	29199-2	Y	Core image coding specification including the codestream syntax, normative specified decoding process, informative example encoding process, and (optional) tag-based file format.
Motion JPEG XR	Rec. T.833	29199-2	Y	Specification of file storage format and decoding process for timed sequences of images encoded using JPEG XR encoding
Conformance testing	Rec. T.834	29199-4	Y	Methods and test suite for conformance testing of JPEG XR encoders and decoders
Reference software	Rec. T.835	29199-5	Y	Example encoder and reference decoder software in C source code form

5 General overview of technical design

5.1 Basic technology structure

JPEG XR is a block transform based image coding technology. It shares many of the same basic processing elements as are found in typical prior image coding designs, including the following:

– colour conversion

– rectangular region segmentation

– frequency transformation of block-shaped spatial regions

– sequential scanning of block transform coefficients

– scalar quantization of transform coefficient values, and

– variable-length coding.

Additional features of the design that may not be found in some older image coding systems include the following:

- tile region segmentation
- multi-resolution frequency band hierarchy
- reversible integer-based colour conversion
- reversible integer-based spatial frequency transformation
- overlapped block processing for spatial frequency transformation
- selectable degrees of overlap processing (or elimination of overlap processing) within tiles
- selectable use of either "soft" (overlapped) or "hard" (non-overlapped) tile boundary processing
- prediction of transform coefficient values
- prediction of transform coded block patterns
- integer processing of floating-point data representations
- fixed-length coded "flexbits" coefficient fidelity refinement data
- adaptive coefficient scanning order
- adaptive switching of variable-length code tables
- support of both lossless and lossy compression using the same signal processing steps, and
- exact specification of decoded image data values (for both lossless and lossy representations)

5.2 Supported image format types

JPEG XR supports the encoding of a variety of basic decoded output image formats as shown in Table 2. The design includes a distinction between the "internal colour format" (specified by the INTERNAL_CLR_FORMAT syntax element) that is used for the processing steps within the main part of the decoding process, and the intended decoded output format (specified by the OUTPUT_CLR_FORMAT and OUTPUT_BITDEPTH syntax elements) to which the decoded image is converted prior to final output. Six types of internal colour format are supported (corresponding to the enumeration values YONLY, YUV420, YUV422, YUV444, YUVK, and NCOMPONENT). In all cases, the naming of a format type should not be interpreted as necessarily implying a particular relationship to visible light interpretations in the sense of a CIE colour space – for further detail on this subject, the reader is referred to ITU-T Rec. T.832 | ISO/IEC 29199-2 Annex C.

Table 2 – Supported image formats

Basic format type	**Supported colour bit depths and representations**
Grayscale	1, 8 and16 bits per component unsigned integer 16 and 32 bits per component fixed point 16 and 32 bits per component floating point
RGB	8, 10, and 16 bits per component unsigned integer 16 and 32 bits per component fixed point 16 and 32 bits per component floating point 5-5-5- and 5-6-5 packed bits per component unsigned integer
RGB with Alpha channel	8 and 16 bits per component unsigned integer 16 and 32 bits per component fixed point 16 and 32 bits per component floating point
Shared-exponent RGBE	Four bytes: one for the red (R) mantissa, one for the green mantissa (G), one for the blue (B) mantissa, and one for a common exponent (E)
CMYK and CMYK with Alpha	8 and 16 bits per component unsigned integer
YUV 4:2:0 and YUV 4:2:0 with Alpha	8 bits per component unsigned integer
YUV 4:2:2 and YUV 4:2:2 with Alpha	8, 10 and 16 bits per component unsigned integer
YUV 4:4:4 and YUV 4:4:4 with Alpha	8, 10 and 16 bits per component unsigned integer 16 bits per component fixed point
n-Channel and n-Channel with Alpha	8 and 16 bits per component unsigned integer

5.3 Decoded image structure and interpretation

A decoded image may have multiple colour channels (also referred to as *components*). Each colour channel consists of a two-dimensional rectangular array of *sample* values. Each *sample* is a scalar-valued quantity which may represent either an integer or floating-point value.

NOTE 1 - Within the JPEG XR decoding process (or example encoding process), all processing is performed using integer arithmetic; however, the final result of the decoding process (or input to the example encoding process) actually represents a floating point value in some use cases.

NOTE 2 – The term used here is *sample,* rather than the term *pixel* that is sometimes used in such contexts. However, in graphics terminology, the term *pixel* is most typically used to refer to the entire set of scalar-valued quantities for a location in an image (e.g., the intensity of Red, Green, and Blue for a location in an image). In some scenarios, this multi-component concept of a pixel is difficult to apply (such as for the YUV 4:2:2 and YUV 4:2:0 sampling structures supported in JPEG XR, for which the sampling density is different for different colour components). In informal usage, the term pixel may sometimes alternatively refer to the scalar value for a single colour component. For clarity, the term *pixel* has been avoided here in favour of the term *sample* as an unambiguous reference to a scalar-valued quantity.

One channel is referred to as the *luma* channel, and the remaining channels are called the *chroma* channels and (when present) the *alpha* channel.

The luma channel can typically be interpreted as a monochrome representation of the image content. It is often denoted by the symbol Y. Chroma channels are sometimes referred to as U and V channels.

A *monochrome* image has only a luma channel. A YUV image has a luma channel and two chroma channels (and may also have an alpha channel).

NOTE 3 - The use of the term luma or the symbol Y should not be interpreted as necessarily implying that the channel represents true luminance in the light representation sense (e.g., as in CIE specifications). Similarly, the use of the term chroma or the symbols U or V should not be interpreted as implying that the channels represent chromaticity in the light representation sense or that any particular colour space representation is used. When feasible, colour interpretation metadata should be associated with the JPEG XR coded image to specify the actual interpretation of the decoded colour channels.

An image may also have an alpha channel, in which each sample indicates the degree of transparency of a location in the image. Alpha channel support is important to many applications such as gaming and animation.

The colour channels are grouped into *image planes*. When present, the alpha channel may either be encoded together with the other channels or may be encoded separately. When encoded separately, the alpha channel is the only channel of the *alpha image plane*. The set of all other colour channels is referred to as the *primary image plane.* When encoded together with the other channels, the alpha channel is part of the primary image plane.

The decoding process of an alpha image plane is the same as that of a monochrome image.

Ordinarily, each colour channel represents an evenly-spaced rectangular sample grid of samples in which each sample represents the intensity of an associated measure. Generally, the array for every colour channel represents the same spatial region. However, the chroma channels may in some cases be encoded with half of the resolution of the associated luma channel, either horizontally (in the case associated with the internal colour format enumeration value YUV422 in the decoding process, which corresponds to using the YUV 4:2:2 sampling structure) or both horizontally and vertically (in the case associated with the internal colour format enumeration value YUV420 in the decoding process, which corresponds to using the YUV 4:2:0 sampling structure). In such YUV image cases, the intended positioning of the chroma channel sampling grids relative to the luma channel sampling grid can be indicated by the encoder using the codestream syntax elements CHROMA_CENTERING_X and CHROMA_CENTERING_Y.

Although the use of 1, 3, or 4 colour channels is expected to be the most typical, the JPEG XR codestream syntax supports up to 4111 colour channels. When stored using the tag based file format of ITU-T Rec. T.832 | ISO/IEC 29199-2 Annex A, the maximum number of colour channels for a JPEG XR image is 9 (eight channels in the primary image plane plus an alpha image plane).

5.4 Data processing hierarchy and structures

There are five layers in the basic hierarchy of data structures used in the JPEG XR decoding process, as follows:

– sample or transform coefficient (a scalar valued quantity)
– block (a rectangular array of samples or transform coefficients)
– macroblock (the set of samples or transform coefficients for a 16×16 region of the luma component and any associated 16×16, 8×16 or 8×8 regions of other components)
– tile (the set of macroblocks corresponding to a particular separately-encoded rectangular region of an image plane)
– image or image plane

This hierarchy is shown in Figure 1.

Because the dimensions of the represented image may not be exactly divisible by 16, some cropping (e.g., at the top and right edges) of the decoded image planes may be performed to produce the decoded image from the decoded tiles, as illustrated in Figure 1. In addition to supporting cropping for the top and right edges, JPEG XR also supports cropping at the left and bottom edges of the image when desired, which can be important for enabling some use cases – such as the compressed-domain transformation operations discussed in Clause 11.

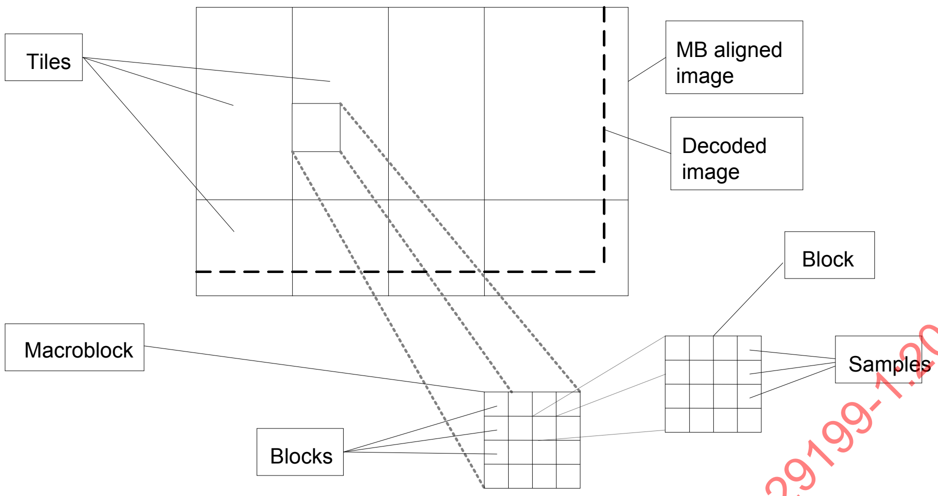

Figure 1 – Hierarchy of data structures used in the JPEG XR decoding process

An image may contain from 1 to 4096 columns of tiles spanning across the horizontal direction and from 1 to 4096 rows of tiles spanning the vertical direction. Image tiles are aligned in rows and columns, such that all tiles containing any subset of the image samples in a given horizontal row have same height and all tiles containing any subset of the image samples in a given vertical column have the same width. However, tiles containing samples of different horizontal rows may have different heights, and tiles containing samples of different vertical columns may have different widths.

5.5 The JPEG XR transform structure and hierarchy

The transform converts the spatial domain image data to frequency-domain information. JPEG XR uses a hierarchical two-stage lapped bi-orthogonal transform (LBT), with a low-complexity structure that is exactly invertible in integer arithmetic (also referred to as integer reversible). The transform is based on two basic operators: the core transform and the overlap filtering. The core transform is conceptually similar to the widely used discrete cosine transform (DCT), and can similarly exploit the spatial correlation within a block. The overlap filtering is designed to exploit the correlation across block boundaries and to mitigate blocking artifacts. Together the combined transform is equivalent to an LBT, and hence it offers state-of-the art coding performance, both objectively and visually, while minimizing computational complexity. JPEG XR further improves the performance of the transform by adopting a two-stage hierarchical construction. The resulting hierarchical two-stage LBT effectively uses longer filters for lower frequencies and shorter filters for higher frequency detail. Thus, the transform has a better coding gain as well as reduced ringing and blocking artifacts when compared to traditional block transforms.

The overlap filtering is functionally independent of the core transform, and it can be switched on or off, as chosen by the encoder. There are three options for overlap filtering: 1) disabled for both stages, 2) enabled for the first stage but disabled for the second stage, or 3) enabled for both stages. The overlap filtering option selected by the encoder is signalled to the decoder as part of the compressed codestream. The flexibility to enable or disable the overlap operators controls the effective filter length of the overall transform. Disabling the overlap filters at both levels can minimize ringing artifacts related to the use of long filters, as well as enable very low decoding complexity. Alternatively, applying the overlap filters at both levels can mitigate blocking artifacts at very low bit rates. However, the typical anticipated use is the enable the overlap for the first transform stage and disable it for the second, which provides a compromise setting with good compression performance over a broad range of bit rates, minimal blocking effects, and minimal ringing artifacts.

Each operation of the JPEG XR transform is designed to be exactly reversible to enable mathematically lossless encoding. JPEG XR implements reversible transforms using a lifting-based structure, which minimizes the dynamic range expansion of the input data, and thus reduces implementation complexity and maximizes lossless compression performance. As the transform lifting operations and all other operations of the decoding process use only integer arithmetic, the decoder output is bit-exact for any given compressed codestream.

The transform operates in a hierarchical manner as follows in the example encoding process for the luma component:

– Each 4x4 block within a component of a macroblock undergoes a first stage transform, yielding one DC coefficient and 15 first-stage AC coefficients for each of the 16 blocks in the 16×16 region corresponding to the macroblock.
– The 16 DC coefficients are then further collected into a single 4×4 block, and a second transform stage is applied to this block. This yields 16 new coefficients: one second-stage DC coefficient, and 15 second-stage AC coefficient for this block of first-stage DC coefficients. These coefficients are referred to, respectively, as the DC and lowpass (LP) coefficients of the original macroblock.
– The other 240 coefficients, i.e. the AC coefficients of the first-stage transform of the macroblock, are referred to as the highpass (HP) coefficients.

The transform coefficients are grouped into three sub-bands that are referred to using the above terminology – i.e., the DC band, LP band, and HP band.

The chroma components are processed similarly; however, in the case of the YUV422 and YUV420 internal colour formats, the chroma arrays for a macroblock are 8×16 and 8×8, respectively, so the processing performed in the second stage of transformation is adjusted to use a smaller block size. In the YUV420 and YUV422 cases, the chroma component for a macroblock has 60 and 120 HP coefficients, respectively.

The LP band of a macroblock is composed of all the AC coefficients (15 coefficients for the luma and full-resolution chroma cases, 7 for YUV422 chroma channels and 3 for YUV420 chroma channels) of the second stage transform.

Figure 2 illustrates this frequency hierarchy for a macroblock.

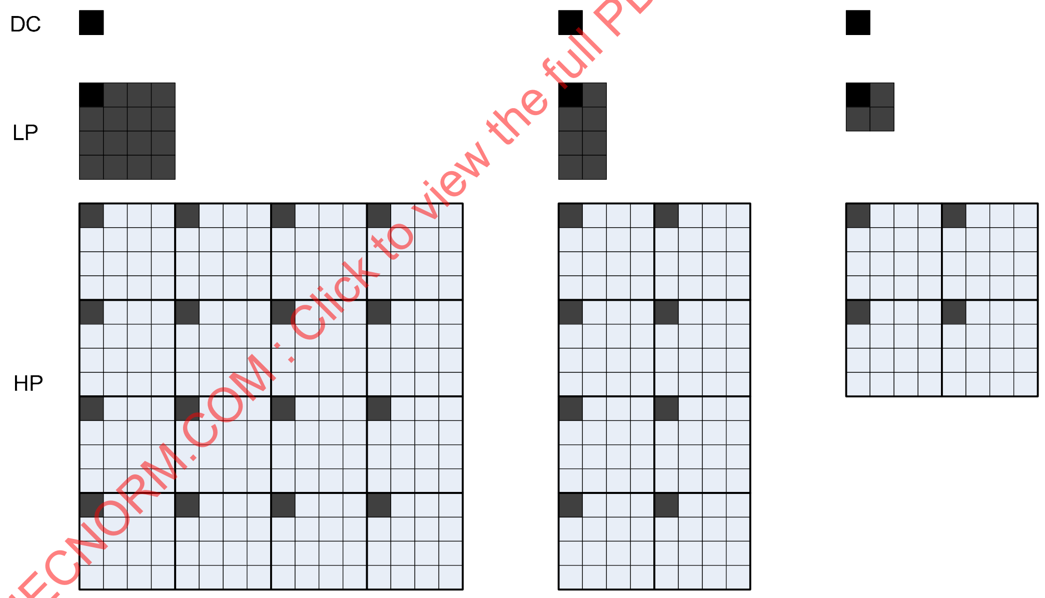

Figure 2 – Frequency hierarchy for a macroblock
(left: luma and full-resolution chroma, center: YUV422 chroma, right: YUV420 chroma)

For the decoding process, this sequence of operations is basically reversed to perform inverse transformation.

NOTE - The discussion has focused on the encoding process since the transform design tends to be conceptually easier to understand from that perspective. However, only the decoding process is normatively specified in the JPEG XR image coding standard.

5.6 Handling of image and tile boundaries

Adjustments are made to the transform processing around the edges of the image when overlapping is enabled, in order to match the DC gain of the processing that is applied in other regions, so that near-flat images do not produce substantial non-DC transform coefficient values and artifacts are minimized near the image boundaries.

JPEG XR supports two tile types of tile boundary handling which can be selected by the encoder:

– "soft" tiling, in which the transform overlapping stages cross over the tile boundaries; and

– "hard" tiling, in which the transform overlapping is applied only within each individual tile, and the boundaries of tiles are treated in the same manner as the extreme boundaries of the image.

When the overlap mode selected by the encoder is set to disable all overlap filtering within the tiles, all tile boundaries are naturally "hard". However, when overlap processing is enabled within tiles and "soft" tile boundary handling is selected, proper decoding of the samples in areas very close to the tile boundaries that are not image boundaries requires access to data from more than one tile. The selection of "hard" tile boundary handling by the encoder can eliminate this cross-tile decoding dependency, although it may induce some block artifacts at low bit rates in a manner similar to that of an ordinary non-overlapped block transform.

5.7 Quantization and lossless representation

5.7.1 Overall quantization design concepts

The purpose of quantization is to reduce the entropy of each transform coefficient value by (in actuality or as a conceptual analogy to the processing technique which may be more sophisticated) dividing its value by a scale factor and rounding the result to an integer value. The same scaling factor is then applied during the decoding process to amplify the quantized integer values in order to perform an approximate inversion of the encoder's quantization process. This scaling factor is referred to as the quantization step size (as the scaling factor governs the size of the increment between adjacent selectable decoded coefficient reconstruction values), and the process of applying the scaling factor is referred to as inverse quantization, dequantization, or value reconstruction (although, strictly speaking, quantization is not an invertible process). This type of quantization and inverse quantization processing is often referred to as scalar quantization, uniform scalar quantization, mid-tread scalar quantization, or uniform scalar quantization with a dead-zone, although, the use of such terms sometimes incorrectly implies certain restrictions on the way the encoder operates.

A key example encoding process for scalar quantization is to divide the true coefficient value by the step size and round the result to an integer value. The "mid-tread" and "dead-zone" terms refer to the region of coefficient values that are mapped to a zero-valued representation, and a biasing of the rounding operation may be used to make this region larger than the region of input coefficient values mapped to other reconstruction values – as a way to reduce the entropy of the result. When such a biasing is used, the "uniform scalar quantization" term is not completely accurate (because the expanded dead-zone indicates that the quantization steps are not uniform in size), and when more sophisticated techniques are applied in the encoder, the "scalar quantization" term may also not be completely accurate (because the quantization process may involve more than just a deterministic mapping of scalar input to a quantized output). The JPEG XR standard actually only normatively specifies the decoding process, while leaving encoder designers the freedom to design encoding algorithms that may be more sophisticated than ordinary scalar quantization.

The selection of the quantization step size provides the encoder with a mechanism to trade off the quality of the encoded image with the bit rate required to represent it in compressed form. For JPEG XR encoding, since the true transform coefficient values prior to quantization are integers (because of the lifting-based invertible design of the JPEG XR transform processing steps), dividing the coefficient values by a quantization step size equal to 1 is equivalent to skipping the quantization operation, and allows the coefficient values to be represented exactly without approximation error. In this manner, the JPEG XR design enables the encoded representation of the source image to either be completely mathematically lossless or to be lossy while keeping the decoding process the same in either case.

A parameter referred to as the quantization parameter (QP) is used to select the step size. When the QP value is small, a small change of QP results in a small change of the quantization step size; for larger values of QP the same incremental difference results in a larger difference in the quantization step size.

JPEG XR provides several ways of controlling quantization parameters on a spatial region, frequency band, and image component basis. This flexibility enables the encoders to deploy various bit allocation techniques to improve the quality of encoded image according to their desired criteria.

5.7.2 Quantization control on a spatial region basis

In the spatial dimension, JPEG XR allows the following types of QP control:

- A QP value can be selected in the image plane header.
- A QP value can be selected in the tile header, allowing different tiles within the image to use different QPs.
- Different macroblocks within a tile can use different QP values.

In the last case, JPEG XR allows the tile header to define up to 16 different QP values for the LP and HP bands of that tile. For each macroblock, an index is sent as part of the macroblock information that specifies which of these QP values is to be applied for decoding that macroblock. The index specifying the QP is variable-length coded to minimize the signaling overhead. (However, only the QP for the LP and HP bands can vary on a macroblock basis in this manner; the quantization parameter for the DC band for all macroblocks in the tile must be identical.)

5.7.3 Quantization control on a frequency band basis

JPEG XR allows the QP to depend on the frequency bands in the following ways:

- All frequency bands may share the same QP value.
- The coefficients in the DC and LP bands may use the same QP value, while the coefficients in the high pass band use a different QP.
- The coefficients in the LP and HP bands may use the same QP value, while the DC coefficients use a different QP, or
- Each frequency band can use a different QP value.

5.7.4 Quantization control on a colour plane component basis

The relationship between the QP of the different colour planes can be established in the following modes:

- In the uniform mode, the QP value for all the colour planes is the same.
- In the mixed mode, the QP value for the luma colour plane is set to one value, while the QP for all other colour planes is set to a different value.
- In the independent mode, the QP value for each colour plane can be specified separately.

5.7.5 Quantization control type combinations

JPEG XR also allows for a combination of the above flexibilities. For example, one tile could have different QP values for the different colour planes, but the same QP applied to different bands. Another tile could have different QP values for the different frequency bands of the luma component, but use the same QP for all bands of the chroma components. Thus, the quantization control can be tuned for a rate-distortion optimized bit allocation as well as to support features such as region of interest (ROI) emphasis. The quantity of "overhead" data used to signal the QP values for the most common application scenarios is minimized in various ways in the syntax design.

5.8 Prediction of transform coefficients and coded block patterns

JPEG XR uses inter-block prediction of the transform coefficients and coded block patterns (CBPs) to achieve additional coding efficiency. There are four types of inter-block prediction in JPEG XR as follows:

- Prediction of DC coefficient values across macroblocks within tiles
- Prediction of LP coefficient values across macroblocks within tiles
- Prediction of HP coefficient values across first-stage transform blocks within macroblocks
- Prediction of coded block patterns (CBPs) across first-stage transform blocks within macroblocks and across macroblocks within tiles

As the overlap transform already removes some amount of inter-block redundancy, prediction is used only when the inter-block correlation has a strong and dominant orientation. For any block of transform coefficients that is encoded using prediction, only the DC coefficient or one row or column of the transform coefficient values is encoded using prediction.

When an image contains more than one tile, the prediction processes operate only within each individual tile, to enable independent entropy decoding of the tiles.

5.9 Adaptive ordering of coefficient scanning pattern

Coefficient scanning is the process of converting each 2-D block of transform coefficients into a 1-D list of symbols for entropy coding. In some designs this process is referred to as zigzag scanning. In JPEG XR, the coefficient scan pattern is adapted dynamically based on the local statistics of the preceding coded coefficients in the tile. Adjacent members of the scan order may not be horizontally or vertically neighbouring in their 2-D block index values. The adaptation process adjusts the scan pattern so that coefficients with a higher probability of non-zero values are scanned earlier in the scanning order.

5.10 Entropy coding of transform coefficients

Coded transform coefficients typically account for a high percentage of the bit usage. Therefore, the efficient coding of transform coefficients is critical to the overall performance of an image codec.

A traditional approach is based on forming run-level symbols which each jointly represent a run of zeros between non-zero quantized transform coefficient values together with the value of the next non-zero coefficient, and then entropy coding the run-level symbols using a variable-length code (VLC) table. Often a single VLC table is used for each type of coefficient.

Another well-known entropy coding is that of context-adaptive binary arithmetic encoding, in which the symbols to be encoded are decomposed into binary symbol representations, each with an associated context selection based on neighbouring symbol values, and then the binary symbols are encoded with an arithmetic coding engine using a context-specific probability estimate, and the probability estimate is updated after encoding each binary symbol. Such a method can provide very good coding efficiency, but has substantial requirements for computational complexity and involves a high degree of serial computational dependencies.

The entropy coding in JPEG XR differs from these other designs in the following respects:

– VLC-based coding is applied rather than arithmetic coding, in order to minimize computational complexity and serial computational dependencies. However, a more sophisticated and adaptive VLC encoding technique is applied to provide enhanced coding efficiency.

– Coefficient values are normalized so that only their most-significant bits are encoded using a VLC and so that no more than one quarter of these encoded bits is likely to be non-zero. The least significant bits dropped as a result of the normalization are coded using fixed-length codes.

– A joint symbol signals a non-zero transform coefficient value together with the run of zeros after the coefficient rather than before it. This approach, referred to as "3½D-2½D" coding, provides some improvement in coding efficiency over the older technique of "2D" run-level paired symbol coding .

– Multiple contexts are defined for the joint symbols to be encoded, based on the values of neighbouring previously-encoded data. The use of these contexts exploit non-stationary statistics of the coefficients so as to tune the encoding adaptively to the individualized local context.

– The choice of the employed VLC table among a set of predefined ones is selected in an adaptive manner based on local statistics of previously coded symbols for the same selected context.

– The VLC table size is minimized so as to reduce memory footprint.

This entropy coding design achieves enhanced compression capability relative to traditional designs while minimizing the computational complexity and memory footprint.

When the image sample values have a high dynamic range, the dynamic range of the transform coefficients is even higher. With lossless coding or small quantization step sizes, the range of quantized transform coefficients will then also be relatively high. If VLC entropy coding were directly applied to coefficient values with such a high dynamic range, it could lead to a substantial increase of computational and memory complexity both at the encoder and the decoder.

Adaptive coefficient normalization is a step which processes such transform coefficients so as to render them suitable for efficient entropy coding using smaller VLC tables. In adaptive coefficient normalization, a statistical measure of the variance of the coefficients is computed to group the coefficients into magnitude categories that each contain a number of possible coefficient values that is a power of two. Each magnitude category is referred to as a "bin". The coefficient is then located within its respective category by its in-bin address which is the fixed-length code that indicates which member of the bin category is indicated. Thus, the bin identifier and the in-bin address uniquely determine the encoded quantized coefficient value. Instead of coding the coefficient directly (e.g., using a large VLC table), its representation is decomposed into the bin identifier and in-bin address.

The bin identifier is compressed using an efficient entropy code constructed by context-adaptive selection of a table from a small set of VLC tables. The in-bin address is quite random, to the degree that entropy coding would not substantially help in compressing this address. Therefore, a fixed-length code (using the number of bits equal to the base-two logarithm of the number of values in the associated bin) is used to encode the in-bin address. For lossless compression of 8 bit per sample source images, the fixed-length code portion of the encoded codestream may account for more than 50% of the total bits, and the fraction may be even higher for source images with a wider dynamic range.

In the JPEG XR frequency-mode codestream structure (described in 5.11), the fixed-length in-bin address bits of the HP coefficients are separated from the entropy coded bin identifiers and are grouped together into a separate portion of the codestream. This portion of codestream is known as the "flexbits". These flexbits form a fidelity enhancement layer which can be used to improve the quality of the decoded image as produced without use of the flexbits. This layer may alternatively be omitted or truncated to reduce the size of the compressed image. Thus, JPEG XR enables the use of progressive layered decoding where a coarse reconstruction of the image may be obtained even if the flexbits are unavailable or if they are only partially available at the decoder. A special case of this approach is where an exact lossless reconstruction of the source image can be obtained by using the flexbits, and a lossy reconstruction can be formed without them.

5.11 Codestream structure

There are two fundamental modes of codestream structuring that are supported in JPEG XR – the spatial and frequency modes. In both modes, the codestream is laid out with an image header and image plane header followed by an index table. The index table indicates the location of the data for each tile. The two types of codestream structure are illustrated for an example image in Figure 3.

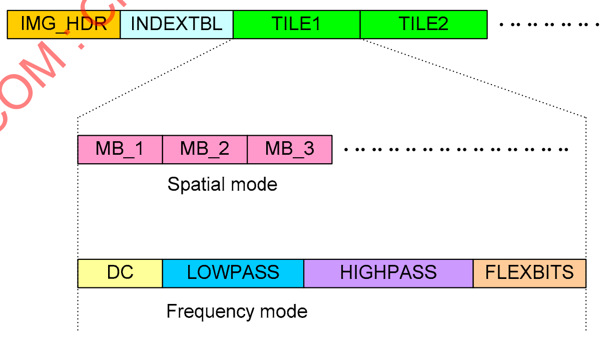

Figure 3 – JPEG XR codestream structure

Although in this illustration the data for different tiles are shown as being positioned contiguously and consecutively in tile order, the actual relative positioning of data for different tiles may be arbitrary within the codestream, and there may be gaps between the data for different tiles. The actual locations of the sections of data for each tile are indicated in the index table. Also, although in this illustration the data for each tile is shown as a contiguous section of codestream data, this is not necessarily the case in the frequency mode of operation.

In the spatial mode, the coded data for a tile is laid out in a single contiguous section of the codestream data, in macroblock order. The compressed bits pertinent to each macroblock are located together, and the data for different macroblocks of a tile are concatenated in raster scan order, i.e., scanning left to right and then from top to bottom within each tile.

In the frequency mode, the relevant codestream data is separated into four frequency bands. The encoded data for the DC coefficients of all macroblocks within the tile are collected together in raster scan order, the encoded data for the LP coefficients of all macroblocks within the tile are collected together in raster scan order, and the encoded data for the HP coefficients of all macroblocks within the tile are collected together in raster scan order, such that the data for each of these frequency bands is found at a separate position in the codestream. This forms three sub-band structured data groups. Tile sub-bands may be ordered arbitrarily, although may typically be desirable to serialize DC, LP and HP sub-bands in that order for ease of progressive access. In addition, a fourth "band" consisting of the flexbits, comprising information pertaining to the low order bits of the HP band coefficients is formed. The flexbits can account for a significant fraction of the overall codestream size in the case of lossless or high-fidelity encoding, while for large values of quantization step size, the flexbits may be absent.

6 JPEG XR design in relation to baseline JPEG and JPEG 2000

6.1 General

The baseline JPEG (ITU-T Rec. T.81 | ISO/IEC 10918-1), JPEG 2000 (ITU-T Rec. T.800 | ISO/IEC 15444-1), and JPEG XR (ITU-T Rec. T.832 | ISO/IEC 29199-2) core imaging coding technologies are all based on frequency-domain coding. Frequency-domain coding techniques combine frequency decomposition with some quantization and entropy coding scheme. The frequency decomposition stage organizes the spectral content of an image into distinct frequency coefficients. The coefficients are then quantized and entropy coded to form the compressed codestream.

The original JPEG standard employs a frequency decomposition that is confined within the borders of 8×8 blocks. Such a scheme is referred to as a block transform. Block transforms are relatively simple to implement in both software and hardware. However, the choice of block size in such a design requires a compromise between having sufficiently long frequency analysis basis functions to capture the essential low frequencies of an image when coding at low bit rates, and having sufficiently short basis functions to adapt to brief transitory high-frequency phenomena. Excessively large block sizes can tend to lead to blurriness and "ringing" artifacts near edges in visual scenes, while insufficiently large block sizes can fail to provide the necessary "theoretical coding gain" advantage of capturing very-low-frequency characteristics into very few transform coefficients. Additionally, in block-based coding schemes, the edges of the decoded blocks can become visible due to the different quantization errors at neighbouring blocks, thereby producing the well-known visually annoying phenomenon of "blocking artifacts". The original JPEG design did not support the separation of images into separately-encoded tile regions.

The JPEG 2000 standard addresses these shortcomings of the original JPEG design by employing a wavelet transform decomposition within each of its tile regions. In such a wavelet decomposition, the basis functions of the lower frequencies are made longer than those of the higher frequencies, such that the number of resulting transform coefficients that represent the low frequencies becomes correspondingly smaller than the number of coefficients representing high frequencies. Such a scheme additionally avoids the production of blocking artifacts since hard-edged segmentation boundaries are avoided. Region segmentation independence is supported by optionally segmenting pictures into rectangular tiles and applying the wavelet transform only within the boundaries of each tile (with the choice of tile size being selected by the encoder). Tile boundary handling in JPEG 2000 always uses "hard" tile boundaries, such that all samples corresponding to the area for each tile can be decoded without cross-tile dependencies (and tile boundary block edge effects may become evident at low bit rates).

As discussed in 5.5, JPEG XR image coding uses a frequency decomposition scheme that is a hybrid between the JPEG and JPEG 2000 approaches. Like the original JPEG standard, its frequency decomposition uses block-based transformations, which eases the implementation. However, an overlapping processing stage is employed to lengthen the basis functions and thus enhance theoretical coding gain beyond what would be provided by an ordinary block transform of the same basic block size. Additionally, the block transform is re-applied in a hierarchical fashion, which gives its sub-band decomposition some substantial similarities to that of a wavelet transform. Specifically, a high degree of localized precision is enabled by use of a relatively short basic transform block size (4×4, with extension to an 8×8 region of support by the use of overlapping operators when such operators are applied by the encoder) while repeated application of the overlapping and core transform stages extends the lowest-frequency basis function to a length of 16×16, 20×20, or as much as 36×36, depending on the degree of overlapping selected by the encoder. The number of coefficients for each frequency is disseminated in a hierarchical fashion as in wavelet decomposition, such that the number of frequency coefficients for the lowest and middle frequency bands are related to those of the highest frequencies by factors of 256 and 16, respectively, with the highest-frequency coefficients having relatively short basis functions. Moreover, a tiling segmentation feature, conceptually similar to the one found in JPEG 2000, is also supported in JPEG XR.

Thus all three of these core coding technologies apply two basic signal processing building blocks: image area partitioning (whether block or tile based) and frequency decomposition. The following Subclauses will discuss these two schemes and how they can be viewed in a more general manner.

6.2 Image area partitions

Both JPEG 2000 and JPEG XR share a core concept called "tiling" which allows a specification of rectangular regions that subdivide the image. JPEG XR can partition an image into a regular or irregular grid (in rows and columns) of tiles. Each tile is independently encoded. This subdivision into tiles is utilized in a number of ways; to specify a context for image coding; or to allow areas of the image called "regions of interest" to be manipulated by the system-level architecture. The flexibility of the tiling segmentation feature is further enhanced in JPEG XR by support of both the "hard" and "soft" modes of tile boundary handling.

6.3 Image fidelity refinement

JPEG 2000 and JPEG XR also use frequency sub-band representation of an image. Frequency sub-band decomposition is useful for partitioning the spectral content of an image in a manner that allows for scenarios that benefit from progressive resolution refinement of the image. JPEG 2000 support an arbitrary number of dyadic decompositions of the image during the encoding process, while in JPEG XR, the input image is decomposed by two levels of factor-of-four transform decomposition in each image dimension and therefore JPEG XR natively provides three levels of spatial frequency resolution in each dimension (ratios of 1/16, 1/4, and 1 in each dimension, or 1/256, 1/16, and 1 in two-dimensional sampling ratios).

JPEG 2000 uses bit-plane coding of transform coefficients. Therefore, its codestream can be progressively decoded in bit-plane order and can provide a continuous degree of quality refinement. In JPEG XR, the quality refinement at the decoding time can be provided at two distinct levels: decoding the codestream without the flexbits, and decoding the codestream with the flexbits.

7 High dynamic range (HDR) image coding

7.1 HDR formats supported in JPEG XR

The support of high dynamic range (HDR) imagery is an important feature of the JPEG XR design. In this discussion, we refer to sample dynamic ranges beyond 8-10 bits per sample as high dynamic range imagery.

JPEG XR can support compression of various HDR image formats, as illustrated below in Table 3. Further, JPEG XR can enable lossless compression of many of these HDR formats, including such formats as 16-bit signed and unsigned integer and fixed-point representations, and 16-bit floating-point representation. The fixed-point and floating-point representations can be useful for the encoding of values beyond the nominal range from reference black to reference white, which can assist in performing image post-processing such as exposure and white balance adjustments.

Table 3 – High dynamic range imagery types supported in JPEG XR

Supported HDR format	Remarks
16-bit unsigned integer and signed fixed point	Fixed point encoding can assist in encoding values beyond the nominal range from reference black to reference white. Mathematically lossless encoding is supported for these formats.
32-bit signed fixed point	Fixed point encoding can assist in encoding values beyond the nominal range from reference black to reference white
16-bit floating point	Conceptually similar to IEC 60559 (IEEE 754) floating point 1 sign bit, 5 exponent bits and 10 mantissa bits Mathematically lossless encoding is supported for this format.
32-bit floating point	IEC 60559 (IEEE 754) 32-bit single-precision floating point 1 sign bit, 8 exponent bits and 23 mantissa bits
32-bit shared-exponent RGBE	32-bit RGBE includes four bytes: one for the red (R) mantissa, one for the green (G) mantissa, one for the blue (B) mantissa, and one for a shared exponent (E)

7.2 HDR signal processing design in JPEG XR

JPEG XR uses signal processing operations that are performed entirely in the integer domain for the specified decoding process and the anticipated encoding process. However, the design actually supports the coding of floating-point image content by use of a conversion process which enables the manipulation of floating-point image data as integer values within the encoding and decoding processes.

JPEG XR handles the HDR image formats consistently using only integer processing during the encoding and decoding processes. Lossless and lossy coding is possible when the data ranges allow for this to happen with a 32-bit signal processing wordlength. Further, the amount of loss is in line with expectations – with a probabilistically bounded loss for a certain quantization parameter, and avoiding major artifacts such as rollover (where a dark sample "rolls over" to a bright value or vice versa). The representation of the data in a JPEG XR compressed file and the core signal processing steps of the decoding process are consistent across the different supported formats, in the sense that these aspects are essentially independent of the represented source format. At the end of the decoding process, there are specified (simple) conversion operations that take place to convert the decoded image data to its final form for image interpretation by the application.

7.3 Examples of HDR applications for JPEG XR

Some examples of HDR applications for JPEG XR image coding are as follows:

- HDR picture capture, storage, transmission, printing, and display
- HDR picture editing with compression at intermediate stages of workflow
- Interactive network access to HDR photo archives
- Converting uncompressed raw images to JPEG XR to reduce storage size requirements or increase transfer speed

8 JPEG XR profiles and levels

8.1 Overview of profiles and levels

In order achieve interoperability between different encoder and decoder implementations in various applications, coding standards such as JPEG XR define a set of predefined constraints on their syntax and values using definitions of "profiles" and "levels". Such sets of constraints limit the required resources needed for a decoder implementation and therefore enable implementation of decoders with limited resources, while ensuring interoperability for any codestream that conforms to the profile(s) and level(s) that are relevant to the requirements of the application.

The 2010 edition of the JPEG XR image coding specification defines four profiles as shown in Table 4: the Sub-Baseline, Baseline, Main, and Advanced. Profiles define the constraints on the bit depth and types of image coding features that are supported and therefore on the computational complexity derived from these aspects of image decoding. Levels, on the other hand, define the memory requirement of the decoder by limiting parameters such as image size and the maximum number of tiles.

Table 4 – JPEG XR profiles

Profile	Supported Features	Superset of
Sub-Baseline	8-bit Grayscale Up to 10-bit per channel RGB	N/A
Baseline	Up to 16-bit integer and fixed point Grayscale Up to 16-bit integer and fixed point per colour RGB	Sub-Baseline
Main	Grayscale, RGB, CMYK, n-Channel Alpha channel up to 32-bit integer and fixed point 16-bit and 32-bit floating point	Sub-Baseline and Baseline
Advanced	All supported image formats in JPEG XR	Sub-Baseline, Baseline, and Main

8.2 Sub-Baseline profile

This profile defines the constraints for a very basic JPEG XR decoder of the lowest complexity in both computational and memory requirements. This profile is limited to support decoding of grayscale images with bit depth up to 8 bits unsigned integer and decoding RGB images with bit depth up to 10 bits unsigned integer per colour component.

8.3 Baseline profile

This profile defines the constraints for a basic JPEG XR decoder of relatively low complexity in both computational and memory requirements. This profile is limited to support decoding of grayscale images with bit depth up to 16 bits with either unsigned integer or fixed point signed integer, and decoding RGB images with bit depth up to 16 bits with either unsigned integer or fixed point signed integer per colour component. It is primarily intended for applications in which images are captured and/or processed with moderate extended colour range support. Examples of such applications are embedded devices with image capture capability such as mobile phone and light compact digital cameras.

8.4 Main profile

This profile supports many of the image formats that are supported in the JPEG XR image coding specification. This profile supports Grayscale images, RGB, CMYK, and n-Channel, each with or without an alpha channel. This profile supports bit depth up to 32 bits with unsigned integer, fixed point signed integer, float, or half formats. Mainstream software based decoders are good application examples of such profile.

8.5 Advanced profile

This profile supports all image formats that are supported in the 2010 edition of the JPEG XR image coding specification. It is intended for decoder implementations can decode all possible image formats within the JPEG XR design. Software based decoders can be an example of such a decoder. For instance, an image viewer or editing software on a personal computer may support the Advanced profile. The specification of this profile ensures that there is some defined profile that can be identified as supporting any arbitrary expressable syntax for a JPEG XR image codestream.

8.6 Levels

The JPEG XR image coding specification defines seven levels for each profile. Levels constrain the image width and height, the number of tiles horizontally and vertically, the size of tiles horizontally and vertically, and the number of bytes needed to store the decoded image.

9 JPEG XR encoding practices

9.1 General encoding guidelines

The following practices are suggested as general encoding guidelines:

– **Overlap filter mode**: It is recommended to switch off overlap filters for both stages of the transform hierarchy when images are encoded losslessly or at high bit rates. Turning off overlap filters also tends to provide the best results for encoding HDR images with floating-point output formats. For encoding images at medium bit rates, applying the overlap filter only for the first transform stage tends to provide the best results in most cases. Finally, it is sometimes beneficial to switch on the overlap filters of both stages for images that are encoded at very low bit rates.

– **Spatial versus frequency mode codestream format**: The JPEG XR codestream supports two basic representation modes: the spatial and frequency modes. In the spatial mode, the bits representing different sub-bands of each tile are packed together in one single data packet that is parsed macroblock-by-macroblock. In the frequency mode, the codestream bits for each tile are packed into separate sections – one for each sub-band. In the frequency mode, each sub-band can be decoded without needing to access the data for higher frequency sub-bands. The selected codestream format should be chosen by the encoder according to whether the spatial scalability features of the design are expected to be used by the decoder and according to the data buffering capabilities of the encoder.

– **Tile size selection**: Guidelines for tile size selection are provided in 9.3.

– **Internal colour format**: For typical three-component images, it is recommended to use YUV444 internal colour format to obtain the best quality at medium and high bit rates. The YUV422 or YUV420 internal colour formats may be more suitable at low bit rates.

9.2 Encoding for random access

A JPEG XR encoder can partition an image horizontally and vertically into rows and columns of tiles. Each tile is entropy decoded independently, although when "soft" (transform-overlapped) tile boundary operation is selected, proper reconstruction of the image areas near the edges of the tile may require some information from neighbouring tiles. Tiling is the main mechanism provided in JPEG XR for fast local region access.

A smaller tile size provides a finer granularity of fast local access. However, using small tile sizes has some undesired consequences. In particular, the compression efficiency may be reduced when choosing a smaller tile size. Several factors contribute to this issue, as reviewed below:

– JPEG XR entropy coding is adaptive. The entropy coding engine is reset to an initial state at the beginning of a tile. Generally the initial state is not optimal. If the tile is small, the coding engine cannot adapt well to the statistics of the encoded data, resulting in a higher percentage of macroblocks that are not coded optimally, which hurts compression efficiency.

– Each macroblock needs some information from its top and left neighbours in the same tile to predict DC/LP coefficients and coded block patterns (CBPs). Those neighbours are unavailable for the top left macroblock, and only one neighbour is available for any other macroblock on the top or left boundary of the tile. This reduces the coding efficiency. For smaller tile sizes, the penalty is higher.

– The encoder has to spend some bytes on tile header information (such as start codes and quantization information) for each tile. The relative cost of the header is higher for the smaller tiles.

– Each tile requires some bytes on index table offsets to identify where the data packets (one packet for the spatial mode codestream layout and up to four packets for the frequency mode codestream layout) of this tile are in the codestream. Again, the relative cost of these addresses is higher for the small tiles. Also the frequency mode codestream layout has even higher cost in this regard.

Usually, the first factor contributes the most to the coding efficiency loss. The second factor dominates the efficiency loss for very small tile sizes. At low bit rates, the third and forth factors may also be significant.

9.3 Guidelines for tile size selection

Table 5 illustrates the general trend of bit rate increases as tile size is varied. Each entry provides the average percentage bit rate increase for a given QP and tile size combination as measured experimentally for a set of sample images. Each image was encoded with 64 different QP and tile size combinations. For each image and each QP value, the bit rate for the single tiled case is used as the reference for calculating the bit rate increase percentage for each other tile size selection.

Table 5 – Example average bit rate increase percentage as a function of tile size

Tile size	QP=1	QP=8	QP=16	QP=24	QP=32	QP=48	QP=64	QP=80
1088×896	0.06%	0.01%	−0.13%	−0.22%	−0.33%	−0.22%	−0.08%	0.31%
496×528	0.12%	−0.03%	−0.20%	−0.31%	−0.41%	−0.17%	0.31%	1.46%
256×256	0.32%	0.08%	−0.07%	−0.09%	−0.12%	0.62%	2.26%	6.06%
128×128	1.18%	0.62%	0.77%	1.12%	1.49%	4.18%	10.45%	25.34%
64×64	4.45%	2.67%	3.91%	5.60%	7.49%	17.15%	40.31%	96.52%
32×32	16.71%	10.21%	15.05%	21.45%	28.77%	63.83%	149.69%	363.72%
16×16	52.26%	37.00%	49.65%	66.80%	86.51%	178.40%	399.48%	941.25%

The following observations are evident in these experiment results:

– Using small tile sizes incurs more coding efficiency penalty for lower bit rates. The main reason for this is that for a lower bit rate, more coefficients are quantized to zero, and thus fewer non-zero samples are entropy coded. This results in less of an opportunity for the entropy coding process to adapt to the image data statistics. Another reason is the cost for tile header and index table offsets become more burdensome for low bit rates.

– For large tile sizes, reducing the tile size often slightly improves the compression efficiency. This is because different regions of an image may have different statistics. Partitioning the image into tiles so that each tile is more statistically homogeneous improves the entropy coding efficiency.

– Tile sizes of 512×512 or larger rarely have any significant compression efficiency loss (bit rate increase greater than 1%).

– Tile sizes of 256×256 or larger typically have a negligible compression efficiency impact (bit rate increase less than 1%), except at low bit rates (less than 0.2 bits per sample).

– Tile sizes of 128×128 or larger typically have a negligible impact (bit rate increase less than 1%) for higher bit rates (greater than 0.6 bits per sample) and are ordinarily acceptable (bit rate increase less than 5%) for intermediate bit rates (0.2 – 0.6 bits per sample).

– Tile sizes of 64×64 or larger are typically acceptable (bit rate increase less than 5%) for higher bit rates (above 1 bit per sample).

– Tile sizes smaller than 64×64 can cause substantial bit rate increases.

Although the above experiments were performed for a relatively few 8 bit per sample RGB images of a certain size, these conclusions should also hold approximately true for other image formats and sizes.