The Consultative Committee for Space Data Systems

Report Concerning Space Data System Standards

IMAGE DATA COMPRESSION

INFORMATIONAL REPORT

CCSDS 120.1-G-1

GREEN BOOK

June 2007

AUTHORITY

Issue:	Green Book, Issue 1
Date:	June 2007
Location:	Washington, DC, USA

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and reflects the consensus of technical working group experts from CCSDS Member Agencies. The procedure for review and authorization of CCSDS Reports is detailed in the *Procedures Manual for the Consultative Committee for Space Data Systems*.

This document is published and maintained by:

CCSDS Secretariat
Office of Space Communication (Code M-3)
National Aeronautics and Space Administration
Washington, DC 20546, USA

FOREWORD

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Report is therefore subject to CCSDS document management and change control procedures, which are defined in the *Procedures Manual for the Consultative Committee for Space Data Systems*. Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

Questions relating to the contents or status of this document should be addressed to the CCSDS Secretariat at the address indicated on page i.

At time of publication, the active Member and Observer Agencies of the CCSDS were:

Member Agencies

- Agenzia Spaziale Italiana (ASI)/Italy.
- British National Space Centre (BNSC)/United Kingdom.
- Canadian Space Agency (CSA)/Canada.
- Centre National d'Etudes Spatiales (CNES)/France.
- Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Federal Space Agency (FSA)/Russian Federation.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- Japan Aerospace Exploration Agency (JAXA)/Japan.
- National Aeronautics and Space Administration (NASA)/USA.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Belgian Federal Science Policy Office (BFSPO)/Belgium.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- Centro Tecnico Aeroespacial (CTA)/Brazil.
- Chinese Academy of Sciences (CAS)/China.
- Chinese Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Danish National Space Center (DNSC)/Denmark.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- Korea Aerospace Research Institute (KARI)/Korea.
- MIKOMTEK: CSIR (CSIR)/Republic of South Africa.
- Ministry of Communications (MOC)/Israel.
- National Institute of Information and Communications Technology (NICT)/Japan.
- National Oceanic and Atmospheric Administration (NOAA)/USA.
- National Space Organization (NSPO)/Taiwan.
- Naval Center for Space Technology (NCST)/USA.
- Space and Upper Atmosphere Research Commission (SUPARCO)/Pakistan.
- Swedish Space Corporation (SSC)/Sweden.
- United States Geological Survey (USGS)/USA.

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 120.1-G-1	Image Data Compression, Informational Report, Issue 1	June 2007	Current issue
EC 1	Editorial Corrrection	July 2007	Corrects erroneous values in table 5-1.

CONTENTS

CONTENTS (continued)

CONTENTS (continued)

Figure Page

CONTENTS (continued)

1 INTRODUCTION

1.1 PURPOSE

This report presents a summary of the key operational concepts and rationale which underlie the requirements for the CCSDS Recommended Standard, *Image Data Compression* (reference [1]). Supporting performance information along with illustrations are also included. This report provides a broad tutorial overview of the CCSDS Image Data Compression algorithm and is aimed at helping first-time readers to understand the Recommended Standard.

1.2 SCOPE

This document provides supporting and descriptive material only: **it is not part of the Recommended Standard**. In the event of any conflict between the *Image Data Compression* Recommended Standard and the material presented herein, the Recommended Standard shall prevail.

1.3 ORGANIZATION OF THE REPORT

This document is organized as follows.

Section 2 gives an overview of the compression algorithm and describes some of its features. This section is intended to provide high-level information for a user considering using the standard, whether the user intends to produce his own implementation or obtain one from a third party.

Section 3 describes compression options and parameters that may be adjusted by a user or an implementer of the Recommended Standard. Examples are presented to provide an indication of how compression may be affected by parameter selections.

Section 4 describes issues that would be of concern for a user attempting to produce a software or hardware implementation of the standard.

Section 5 provides results describing the lossy and lossless compression performance of the standard on test images, along with comparisons to performance results for other compression algorithms.

Section 6 documents some of the considerations and motivations that influenced the selection of the Recommendation and its features.

1.4 NOMENCLATURE

The following convention applies throughout this document:

– The capitalized word 'Recommendation' refers to the *Image Data Compression* recommendation described in reference [1].

1.5 DEFINITIONS

In this document, for any real number x, the largest integer n such that $n \leq x$ is denoted by

$$n = \lfloor x \rfloor,$$

and correspondingly, the smallest integer n such that $n \geq x$ by

$$n = \lceil x \rceil.$$

1.6 TEST IMAGES AND SOFTWARE IMPLEMENTATIONS

Results and examples in this document make use of the set of test images described in annex B. An available software implementation of the Recommendation and a set of verification test data are described in annex B.

1.7 REFERENCES

[1] *Image Data Compression*. Recommendation for Space Data System Standards, CCSDS 122.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, November 2005.

[2] *Space Packet Protocol*. Recommendation for Space Data System Standards, CCSDS 133.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, September 2003.

[3] M.J. Weinberger, G. Seroussi, and G. Sapiro. "The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS." *IEEE Transactions on Image Processing* 9, no. 8 (August 2000): 1309-1324.

[4] *Lossless Data Compression*. Recommendation for Space Data System Standards, CCSDS 121.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 1997.

[5] *Information Technology—JPEG 2000 Image Coding System: Core Coding System*. International Standard, ISO/IEC 15444-1:2004. 2nd ed. Geneva: ISO, 2004.

[6] N. Aranki, W. Jiang, and A. Ortega. "FPGA-Based Parallel Implementation for the Lifting Discrete Wavelet Transform." In *Parallel and Distributed Methods for Image Processing IV*, edited by Hongchi Shi, Patrick C. Coffield, and Divyendu Sinha, 96-107. Proceedings of SPIE Vol. 4118. Bellingham, Washington, USA: SPIE, October 2000.

[7] C. Chrysafis and A. Ortega. "Line-Based, Reduced Memory, Wavelet Image Compression." *IEEE Transactions on Image Processing* 9, no. 3 (March 2000): 378-389.

[8] R.F. Rice. *Some Practical Universal Noiseless Coding Techniques*. JPL-PUB-79-22; NASA-CR-158515. Pasadena, California: JPL, 1979.

[9] S. Golomb. "Run-Length Encodings (Corresp.)." *IEEE Transactions on Information Theory* 12, no. 3 (July 1966): 399- 401.

[10] A. Kiely. "Selecting the Golomb Parameter in Rice Coding." *The Interplanetary Network Progress Report* 42, no. 159 (November 15, 2004).

[11] Majeed M. Hayat, et al. "Statistical Algorithm for Nonuniformity Correction in Focal-Plane Arrays." *Applied Optics* 38, no. 5 (1999): 772-780.

[12] David Taubman and Michael Marcellin. *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Kluwer International Series in Engineering and Computer Science. Norwell, Massachusetts: Kluwer, November 2001.

[13] M.D. Adams and F. Kossentini. "JasPer: A Software-Based JPEG-2000 Codec Implementation." In *Proceedings of 2000 International Conference on Image Processing (Vancouver, BC, Canada)*, 2:53-56. Piscataway, NJ: IEEE, 2000.

[14] D. Le Gall and A. Tabatabai. "Sub-Band Coding of Digital Images Using Symmetric Short Kernel Filters and Arithmetic Coding Techniques." In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing, 1988 (New York, NY, USA)*, 2:761-764. Piscataway, NJ: IEEE, April 1988.

[15] M.J. Weinberger, G. Seroussi, and G. Sapiro. "LOCO-I: A Low Complexity, Context-Based, Lossless Image Compression Algorithm." In *Proceedings of Data Compression Conference, 1996 (Snowbird, UT, USA)*, 140-149. Piscataway, NJ: IEEE, 1996.

[16] A. Said and W.A. Pearlman. "A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees." *IEEE Transactions on Circuits and Systems for Video Technology* 6, no. 3 (June 1996): 243-250.

[17] A. Said and W.A. Pearlman. "Reversible Image Compression via Multiresolution Representation and Predictive Coding." In *Visual Communications and Image Processing '93*, edited by Barry G. Haskell and Hsueh-Ming Hang, 664-674. Proceedings of SPIE Vol. 2094. Bellingham, Washington, USA: SPIE, October 1993.

[18] A. Kiely and M. Klimesh. "The ICER Progressive Wavelet Image Compressor." *The Interplanetary Network Progress Report* 42, no. 155 (November 15, 2003).

[19] J.D. Villasenor, B. Belzer, and J. Liao. "Wavelet Filter Evaluation for Image Compression." *IEEE Transactions on Image Processing* 4, no. 8 (August 1995): 1053-1060.

[20] M.D. Adams and F. Kossentni. "Reversible Integer-to-Integer Wavelet Transforms for Image Compression: Performance Evaluation and Analysis." *IEEE Transactions on Image Processing* 9, no. 6 (June 2000): 1010-1024.

[21] Wim Sweldens. "The Lifting Scheme: A Construction of Second Generation Wavelets." *SIAM Journal on Mathematical Analysis* 29, no. 2 (March 1998): 511-546.

2 OVERVIEW

2.1 INTRODUCTION

The CCSDS Recommended Standard for Image Data Compression (reference [1]) defines a particular algorithm for compression of grayscale images. This general-purpose image compression algorithm has widespread applicability to many types of imaging instruments.

The algorithm is intended to be suitable for use on-board spacecraft; in particular, the algorithm complexity is designed to be sufficiently low to make high-speed hardware implementation feasible. In addition, the algorithm permits a memory-efficient implementation which does not require large intermediate frames for buffering (see 4.3). Consequently, the compressor is appropriate for frame-based image formats (two dimensions acquired simultaneously) produced, for example, by CCD arrays (called image frames) as well as strip-based input formats (i.e., images acquired one line at a time) produced by push-broom type sensors.

The Recommendation supports grayscale (two-dimensional) images with integer-valued pixels having a maximum dynamic range (bit depth) of 16 bits.

The Recommendation can provide both *lossy* and *lossless* compression. Under lossless compression, the original image data can be reproduced exactly, while under lossy compression, quantization and/or other approximations used in the compression process result in the inability to reproduce the original data set without some distortion. The perfect fidelity required by lossless compression results in a lower compression ratio (i.e., higher volume of compressed data) for a given source image.

The compressor relies on a Discrete Wavelet Transform (DWT). The Recommendation supports two choices of DWT: an *integer* and a *floating point* DWT (see 3.3). The integer DWT requires only integer arithmetic, is capable of providing lossless compression, and has lower implementation complexity. The floating point DWT provides improved compression effectiveness at low bit rates, but requires floating point calculations and cannot provide lossless compression.

To limit the effects of data loss that may occur on the communications channel, the wavelet-transformed image data are partitioned into *segments*, each loosely corresponding to a different region of the image (see 2.4.3.2). Each segment is compressed independently, so that the effects of data loss or corruption are limited to the affected segment. (But note that segment boundaries are not sharply defined in the image domain; see 2.4.3.3.) Partitioning the wavelet-transformed image data into segments also has the benefit of limiting the memory required for some implementations. The segment size can be adjusted to trade the degree of data protection for compression effectiveness; smaller segments provide increased protection against data loss, but tend to reduce the overall compression ratio (see 3.5).

Each segment begins with a segment header that provides information about compression options and compressed segment data (see 2.5.2). The encoded segments do not include synchronization markers or other scheme intended to facilitate the automatic identification of

segment boundaries. Consequently, users of the Recommendation must employ a suitable packetization scheme (see reference [2]) or other means of identifying segment boundaries.

Within each compressed image segment, data are arranged so that earlier portions of the compressed data segment tend to make a larger contribution in overall reconstructed segment fidelity than later portions. This *embedded* data structure allows a user to meet a constraint on compressed segment data volume by truncating the compressed bitstream for a segment at the appropriate point.

The tradeoff between reconstructed image quality and compressed data volume for each segment is controlled by specifying the maximum number of bytes in each compressed segment along with a 'quality' limit (see 3.4). The quality limit constrains the amount of wavelet-transformed image information to be encoded for each segment. For each segment, compressed data is produced until the byte limit or quality limit is reached, whichever comes first. Lossless compression is achieved when the integer DWT is used and the number of bytes required for losslessly encoding each segment does not exceed the segment byte limit.

2.2 QUANTIFYING COMPRESSION EFFECTIVENESS

For a compressed image, the *bit rate* achieved by the compressor, measured in bits/pixel, is defined as the number of bits used in the compressed representation of the image divided by the number of pixels in the image.

When image compression is lossy, one of several distortion or quality metrics can be applied to quantify the degree to which the reconstructed image matches the original. Examples of distortion or quality metrics include:

– Mean Squared Error (MSE):

$$\frac{1}{w \cdot h} \sum_i \sum_j (x_{i,j} - \hat{x}_{i,j})^2$$

– Peak Signal to Noise Ratio[1] (PSNR), measured in dB:

$$20 \log_{10} \frac{2^B - 1}{\sqrt{\text{MSE}}} \quad \text{(dB)}$$

– Maximum Absolute Error (MAE)

$$\max_{i,j} \left| x_{i,j} - \hat{x}_{i,j} \right|$$

[1] Note that other definitions of PSNR have been used in the literature, and so one should be cautious when comparing PSNR values from two different sources.

Here w and h denote the image width and height, respectively; $x_{i,j}$ and $\hat{x}_{i,j}$ denote the original and reconstructed pixel value in the i^{th} row and j^{th} column of the image; and B denotes the dynamic range (in bits) of the original image.

Technically speaking, PSNR is a quality metric (because increasing PSNR values indicates increasing reconstructed image fidelity) while MSE and MAE are distortion metrics.

Note that when compression is lossy, the reconstructed image (and therefore, image distortion) depends on the choice of scheme used by the decompressor to reconstruct the image given the information encoded in the compressed image segments. Subsection 4.4 describes the particular scheme used to produce all results presented in this Report.

For a given compressor with a fixed set of parameters, the term *rate-distortion performance* or *compression effectiveness* is used to refer to the image quality achieved as a function of bit rate. Section 3 includes some examples of rate-distortion curves. In the case of lossless compression, compression effectiveness is simply measured by the bit rate achieved. Note that compression effectiveness does not take into account any measure of implementation complexity (speed, memory requirements, etc.).

2.3 OTHER COMPRESSION APPROACHES

An alternative lossless image compression technique is the JPEG-LS standard (reference [3]). Compared to the CCSDS Image Data Compression Recommended Standard, JPEG-LS has lower complexity and achieves similar compression effectiveness, but it is designed only for lossless and near-lossless compression. Another lossless compression technique, which has lower complexity but is not specifically tailored for imagery, has been previously adopted by CCSDS in 1997 (see reference [4]).

The JPEG2000 standard (reference [5]) is another wavelet-based image compressor that is capable of providing effective lossless and lossy compression. The present Recommendation differs from the JPEG2000 standard in several respects:

a) it specifically targets use aboard spacecraft;

b) a careful trade-off has been performed between compression performance and complexity;

c) being less complex, it can be more easily implemented in either hardware or software;

d) it has a limited set of options, supporting its successful application without in-depth algorithm knowledge.

Compression effectiveness comparisons with other compressors are provided in section 5.

2.4 ALGORITHM OPERATION

2.4.1 GENERAL

The compressor consists of two functional parts, depicted in figure 2-1: a DWT module which performs the DWT decomposition of image data, as described in 2.4.3 of this document and section 3 of reference [1], and a Bit-Plane Encoder (BPE) which encodes the transformed data, as described in 2.4.4 of this document and section 4 of reference [1]. In the remainder of this document it is assumed that the reader is familiar with the contents of reference [1] including terminology defined there.

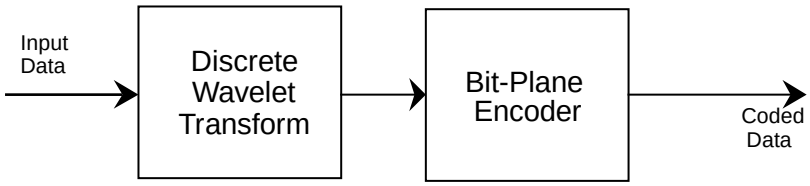


Figure 2-1: General Schematic of the Coder

2.4.2 INPUT IMAGE

For the purposes of the Recommendation, an *image* refers to a two-dimensional rectangular array of integer *pixel* values. The Recommendation supports compression of two-dimensional (grayscale) images with pixel dynamic range (bit depth) up to 16 bits per pixel. Pixels may be signed or unsigned integer quantities.

The Recommendation supports image widths as small as 17 columns and as large as 2^{20} columns. The minimum image height is 17 rows. The maximum image height is unbounded because it is not explicitly encoded as part of header information, but rather image height is inferred indirectly. Specifically, Part 1 of the segment header flags the last segment in an image, which determines the image height to within eight rows. The height can then be completely determined because the value of the height modulo 8 is encoded (in the PadRows field) in Part 1 of the segment header for the last segment (see subsection 4.2 of reference [1]). Because of this indirect method of indicating height, in a push-broom imaging application one could produce and transmit many compressed image segments before the height of the image is known.

Values of basic quantities describing the input image (PixelBitDepth, SignedPixels, ImageWidth) are encoded in the optional Part 4 of the segment header (see 2.5.2). This header part also indicates whether the image should be transposed (TransposeImg) following decompression.

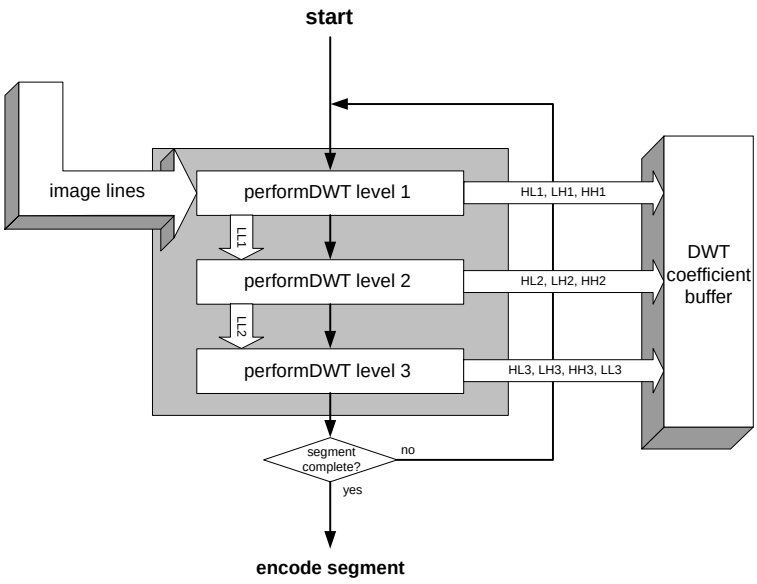
2.4.3 DISCRETE WAVELET TRANSFORM

2.4.3.1 Architecture

A program and data flow diagram of the transform (DWT) stage of the compressor is shown in figure 2-2. Image data input is shown at left in the diagram. The DWT coefficient buffer at right in the diagram stores wavelet coefficients computed by the DWT stage. The program flow in the diagram produces DWT coefficients for a single segment. Once the coefficients corresponding to a segment have been computed and placed in the buffer, the BPE stage can begin encoding that segment. The BPE stage relies on all of the coefficients in a segment being available simultaneously.

The different levels of DWT decomposition in figure 2-2 may operate at different clock rates: the clock rate requirement decreases as the level number increases because of the reduction of resolution at which higher levels of the DWT process the image.

The calculation of a DWT coefficient depends on image pixels in a limited neighborhood. Consequently, when images are encoded using more than one segment, it is often possible to implement the DWT stage so that DWT coefficients for a segment are produced without requiring input of a complete image frame. For example, in a push-broom imaging application, DWT coefficients may be produced in a pipeline fashion as new image scanlines are produced by the imager (references [6] and [7]). A pipeline implementation of the DWT, as discussed in 4.3, leads to DWT-internal storage requirements that are significantly lower than for a conventional frame-based implementation. Subsection 4.3 provides more detail on the program flow within each level and on the size requirements of the DWT-internal buffers (not illustrated in figure 2-2).



NOTE – Line-type arrows indicate program flow; block-type arrows indicate data flow.

Figure 2-2: Program and Data Flow of DWT Module

2.4.3.2 DWT Coefficient Structure: Blocks and Segments

The DWT stage performs three levels of two-dimensional (2-d) wavelet decomposition, producing 10 subbands, as illustrated in figure 2-3.

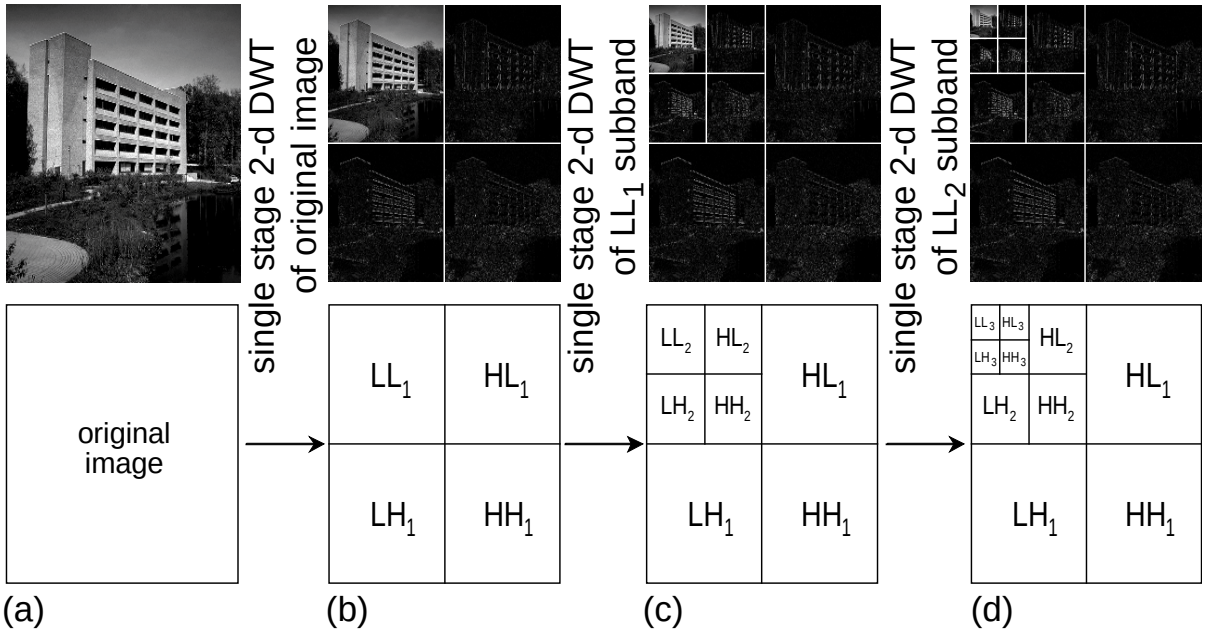


Figure 2-3: Three-Level 2-d DWT Decomposition of an Image

The BPE processes wavelet coefficients in groups of 64 coefficients referred to as *blocks*. A block loosely corresponds to a localized region in the original image. A block consists of a single coefficient from the lowest spatial frequency subband, referred to as the *DC coefficient*, and 63 *AC coefficients*, as illustrated in figure 2-4. This structure is used for jointly encoding information pertaining to groups of coefficients within the block because they exhibit significant statistical correlation.

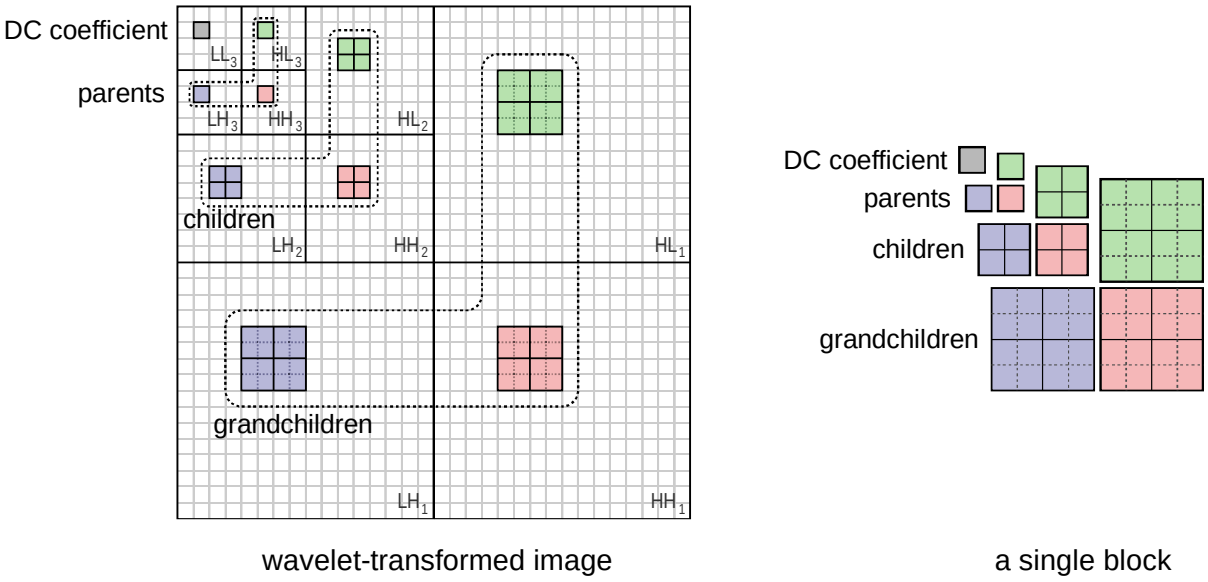


Figure 2-4: Schematic of a Wavelet-Transformed Image with the 64 Shaded Pixels Composing a Single Block

Blocks are processed by the BPE in raster scan order; i.e., rows of blocks are processed from top to bottom, and proceeding from left to right horizontally within a row. A *segment* is defined as a group of S consecutive blocks. Coding of DWT coefficients proceeds segment-by-segment and each segment is coded independently of the others. The value of S may change with each segment.

Figure 2-5 illustrates the partitioning of the DWT coefficients depicted in figure 2-4 into segments, each shaded a different color, when the segment size is S=5 for all but the last segment, which has S=1. (Note that this figure is purely for illustrative purposes; in fact, the minimum value of S is 16 for all but the last segment in an image.)

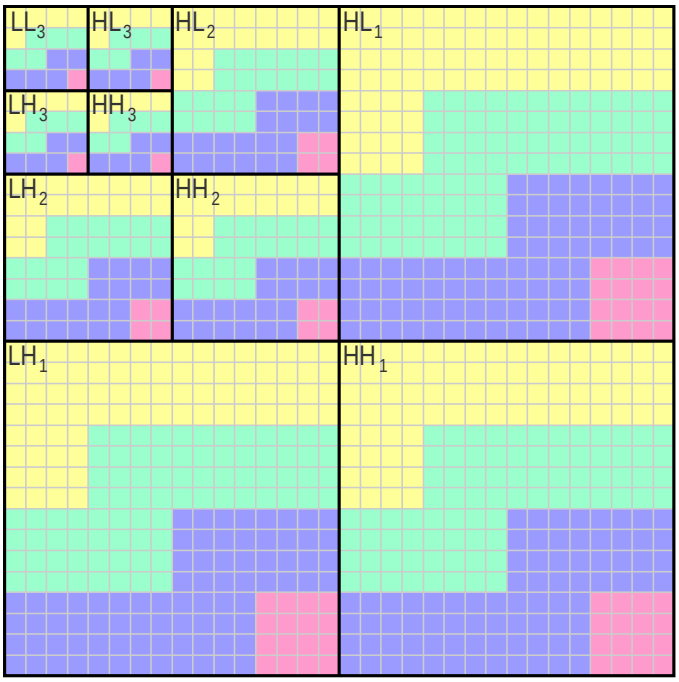
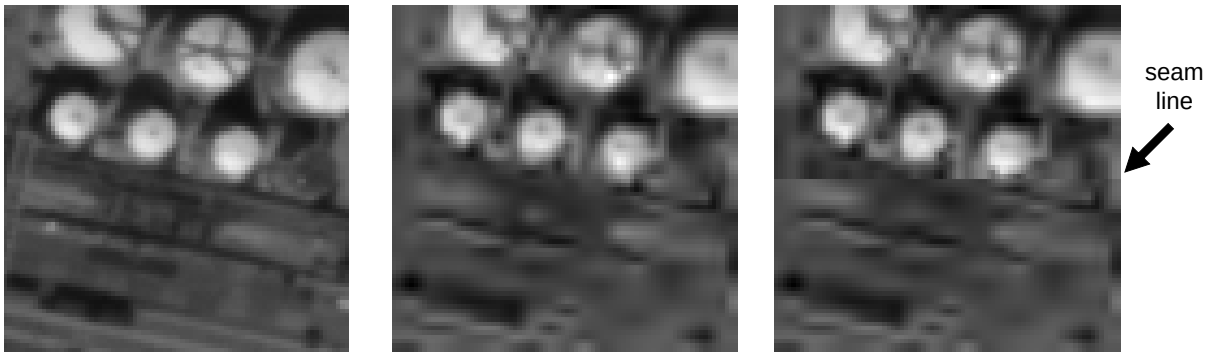


Figure 2-5: Schematic Illustrating the Partitioning of DWT Subbands into Segments Tinted with Different Colors

The use of smaller segments (i.e., a smaller value of *S*) can provide somewhat increased protection against data loss or corruption, and lower memory requirements in some implementations. However, smaller segments tend to reduce compression effectiveness. A discussion of segment size is provided in 3.5.

2.4.3.3 Effect of Block Errors

Partitioning the set of blocks into segments that are independently compressed allows for more efficient memory use and provides robustness to data loss or errors. These benefits could also be achieved by simply partitioning the original image into separate smaller images that are compressed independently. However, such an image-domain partitioning strategy can lead to noticeable boundaries between adjacent segments when lossy compression is used, even when adjacent segments are compressed to the same quality level and no data loss or corruption occurs. Figure 2-6 illustrates this effect. In figure 2-6, reconstruction (b) was produced using segments defined in the DWT domain, as in the present Recommendation. Reconstruction (c), produced by partitioning the original image into two smaller images that were separately compressed, has a noticeable horizontal seam between the upper and lower halves. All segments were compressed to the same quality level.



(a) Original, (b) Partitioning in Wavelet Domain (adopted by the Recommendation) and (c) Partitioning in Image Domain, illustrated using test image Pleiades_portdebouc_pan.

Figure 2-6: Effect of Image Domain Partitioning

Because segments are defined in the transform domain, segment boundaries are not sharply defined in the reconstructed image. Each block consists of 64 DWT coefficients, but the values of these coefficients can affect the reconstructed values of pixels in a localized spatial region that is much larger than 64 pixels. Thus, when a compressed segment is missing or corrupted by errors, it can affect a reconstructed image region that includes more pixels than the number of DWT coefficients in the segment. This region can be bounded by combining the regions of the reconstructed image that may be affected by each block.

A block is identified by the coordinates (r,c) of the DC coefficient (at row r and column c) within the LL_3 subband, with the upper left DWT coefficient in the subband having coordinates $(0,0)$. In an image with width w and height h, the pixels that may be affected by the values of DWT coefficients in block (r,c) are confined to a rectangular region of the reconstructed image with upper left corner $(\max\{8r-21,0\},\max\{8c-21,0\})$ and lower right corner $(\min\{8r+29,h-1\},\min\{8c+29,w-1\})$. For example, figure 2-7 illustrates the set of pixels that may be affected by corruptions to block (3,3). In figure 2-7, the shaded square bounds the region of pixels that may be affected by the values of DWT coefficients in the (3,3) block.

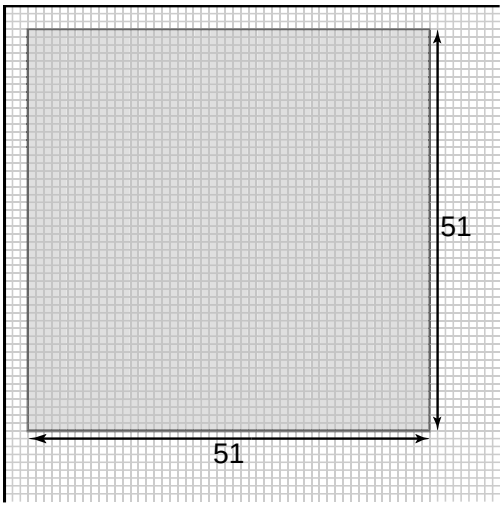
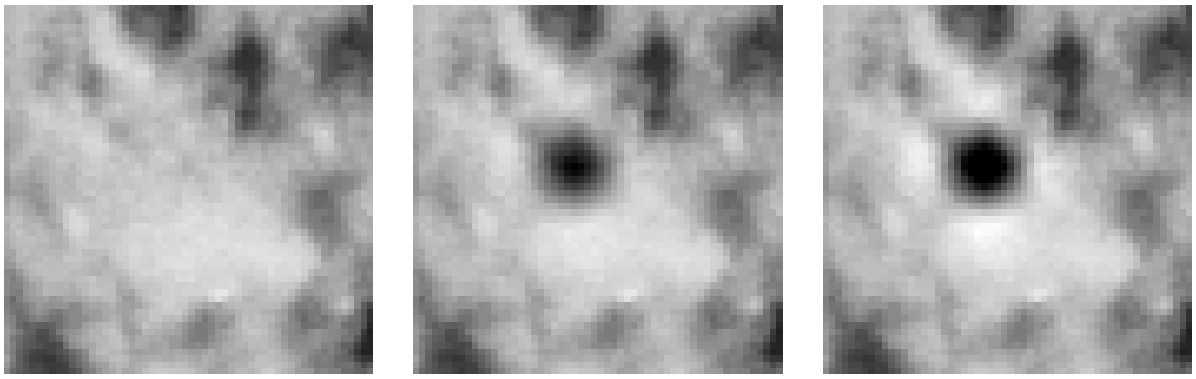


Figure 2-7: Region of Pixels Affected by the Values of DWT Coefficients in the (3,3) Block

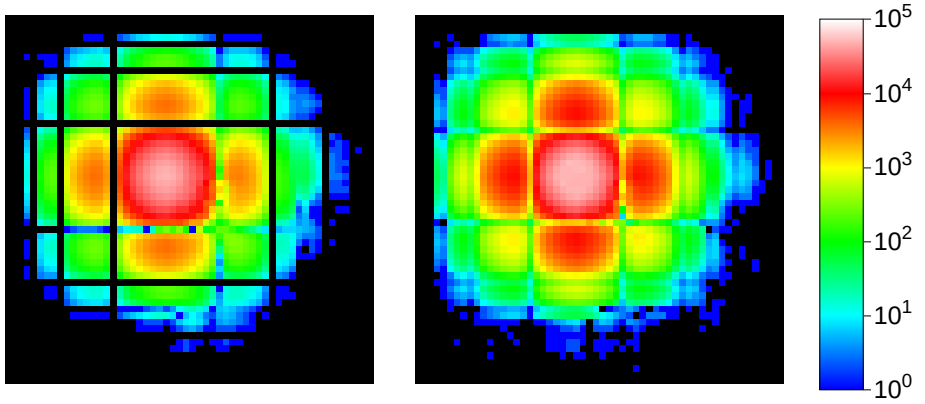
Figure 2-8 illustrates a small image and its reconstructed versions when all coefficients in block (3,3) are set to zero under the integer and float DWTs. Figure 2-9 illustrates the error magnitudes for this example.



(a) (b) (c)

NOTE – The image is a 56x56 pixel region from the p160_b_f test image.

Figure 2-8: Original Image (a) and Reconstructions under (b) Integer DWT and (c) Float DWT with All Coefficients in Block (3,3) Set to Zero



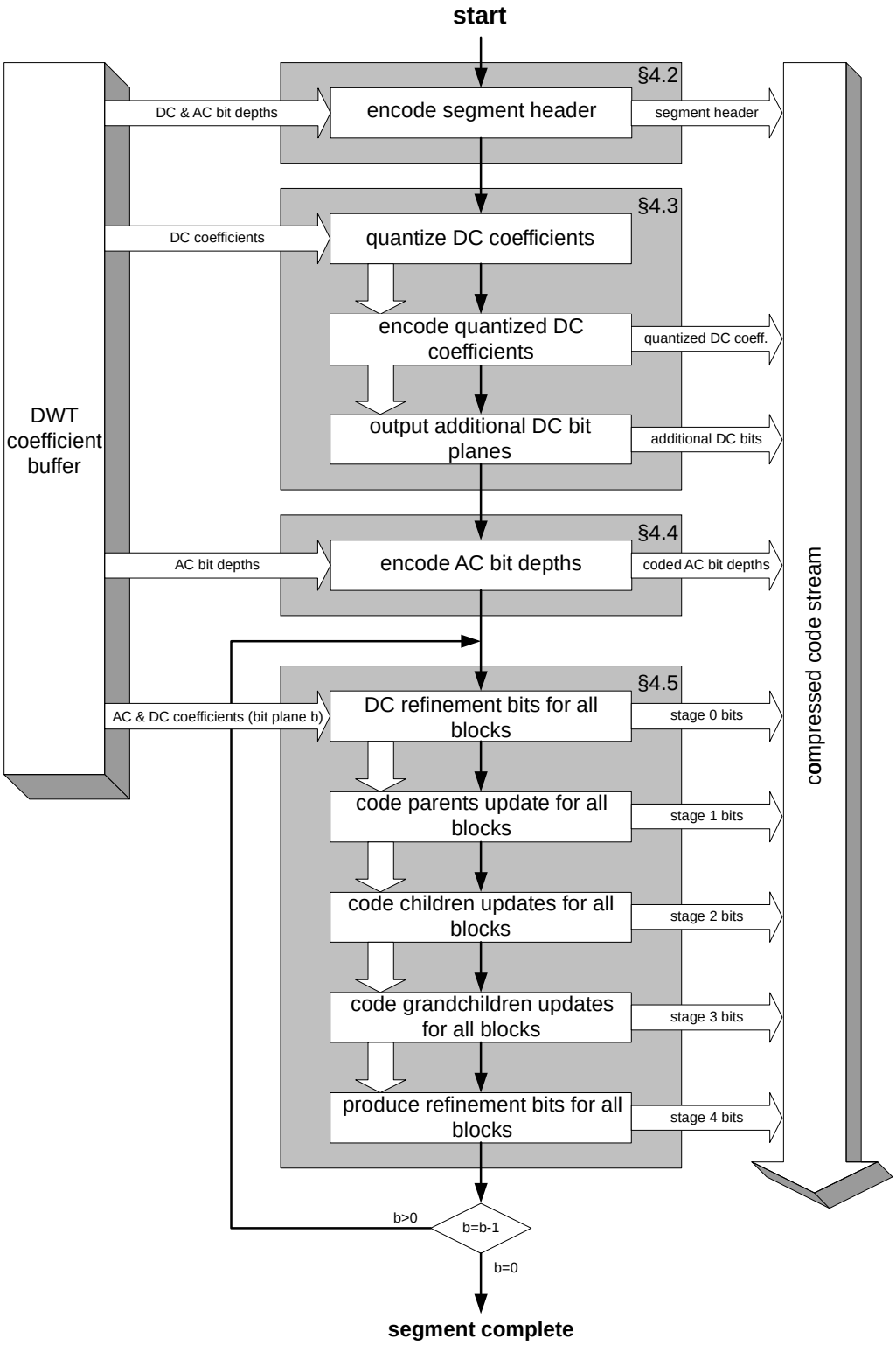
(a) (b)

NOTE – Pixels shown in black were unaffected by setting the block to zero. Colored pixels correspond to the magnitude scale at right.

Figure 2-9: Error Magnitudes for the Example of Figure 2-8 under (a) the Integer and (b) the Float DWTs

2.4.4 BIT-PLANE ENCODER

A program and data flow diagram of the BPE stage of the compressor is shown in figure 2-10. The BPE takes DWT coefficient data from the DWT coefficient buffer, encodes coefficient data, and places the encoded output in the compressed data stream. Coding of a segment is done in four steps, each corresponding to a shaded box in the figure. The steps encode the segment header (see subsection 4.2 of reference [1]), a quantized representation of DC coefficient information (see subsection 4.3 of reference [1]), the bit depths of AC coefficient blocks (see subsection 4.4 of reference [1]), and bit planes of AC coefficients (see subsection 4.5 of reference [1]). These coding steps are summarized below.



NOTE – Line-type arrows indicate program flow, block-type arrows indicate data flow. Subsection numbers refer to subsections of reference [1].

Figure 2-10: Program and Data Flow of BPE

The program flow in figure 2-10 corresponds to the encoding of DWT coefficient data for a single segment, including complete encoding of all bit planes. In fact, encoding of a segment can terminate earlier: coding of a segment stops when the prescribed compressed segment data volume limit has been reached, or when the prescribed segment quality level has been reached, whichever comes first.

The maximum number of bytes in each compressed segment is controlled via the SegByteLimit parameter, and image 'quality' (more specifically, the amount of DWT coefficient information to be encoded) is constrained via the DCStop, BitPlaneStop, and StageStop parameters (see 3.4). These parameters control the tradeoff between reconstructed segment quality and compressed data volume for each segment.

The encoded bitstream for a segment can be further truncated (or, equivalently, coding can be terminated early) at any point to further reduce the data rate, at the price of reduced image quality for the corresponding segment (cf. 3.4).

2.5 STRUCTURE OF AN ENCODED SEGMENT

2.5.1 GENERAL

Figure 2-11 illustrates the structure of an encoded segment. The encoding of each portion of the segment is summarized below.

Segment Header (§4.2)
Initial coding of DC coefficients (§4.3)
Coded AC coefficient bit depths (§4.4)
Coded bit plane b = BitDepthAC-1 (§4.5)
Coded bit plane b = BitDepthAC-2 (§4.5)
. . .
Coded bit plane b = 0 (§4.5)

Figure 2-11: Structure of an Encoded Segment

2.5.2 SEGMENT HEADER

Each encoded image segment begins with a segment header that provides information about compression options and compressed segment data. Some information encoded in the segment header (e.g., BitDepthDC, the maximum bit depth of all DC coefficients of the segment) is calculated based on data in the DWT coefficient buffer, and other values encoded in the segment header indicate user-selected compression options and parameters described

in section 3. The segment header structure is defined in subsection 4.2 of reference [1]. Table 2-1 illustrates the header structure and fields encoded in the header. In the table, fields shaded in green indicate quantities related to basic image parameters (e.g., image size and pixel type) and are described in 2.4.2. Unshaded fields indicate user-selectable parameters or options and are discussed in section 3. Fields shaded in gray indicate quantities that are computed based on input image and are not discussed further in this Report.

Table 2-1: Summary of Header Fields

Part	Field	Length (bits)	Refer to
Part 1A	StartImgFlag	1	
	EndImgFlag	1	
	SegmentCount	8	
	BitDepthDC	5	
	BitDepthAC	5	
	Reserved	1	
	Part2Flag	1	§3.1
	Part3Flag	1	
	Part4Flag	1	
1B	PadRows	3	
	Reserved	5	
Part 2	SegByteLimit	27	§3.3
	DCStop	1	
	BitPlaneStop	5	
	StageStop	2	
	UseFill	1	
	Reserved	4	
Part 3	*S*	20	§3.4
	OptDCSelect	1	§3.5
	OptACSelect	1	
	Reserved	2	
Part 4	DWTtype	1	§3.2
	Reserved	2	
	SignedPixels	1	
	PixelBitDepth	4	
	ImageWidth	20	
	TransposeImg	1	
	CodeWordLength	2	§3.3
	Reserved	1	
	CustomWtFlag	1	§3.6
	custom subband weight values	20	
	Reserved	11	

The segment header has four parts. The first part is mandatory and the remaining three parts are optional. The motivation for this variable-length header structure is to allow users to select only the header parts that are needed for a particular application. The size of a segment header can vary between 3 and 20 bytes depending on which optional header parts are included, and whether the segment is the last segment for an image.

The mandatory first part of the header flags the first and last segments of an image, indicates which optional header parts are included in the segment header, and encodes values of segment information that typically change from segment-to-segment. The first header part is three bytes long except for the last segment in an image, in which case the first header part is four bytes long. This extra byte is used to encode the image height, modulo 8.

The optional five-byte second part of the header specifies limits on the number of compressed bytes in a segment and the limits on the fidelity with which DWT coefficients are encoded. This part might be included at the start of an image or application session, or at the beginning of each segment when these parameters are adjusted from segment to segment, e.g., for variable output rate control.

The optional three-byte third part of the header specifies information that is typically fixed for each image or application session (e.g., *S*, the number of blocks in the segment), but is allowed to change with each segment. In a typical application, this part might be included at the beginning of each image, but not included for each segment.

The optional eight-byte fourth part of the header specifies parameters that must be fixed for an entire image, such as the choice of DWT (Float or Integer), image width, and the bit depth of pixels in the original image.

2.5.3 INITIAL CODING OF DC COEFFICIENTS

Following the segment header, the BPE encodes a quantized version of the DC coefficients in the segment (see subsection 4.3 of reference [1]). This initial step of coding DC coefficient information is done separately from the coding of AC coefficients so that a simple differential coding method can be used to exploit inter-block correlation among DC coefficients.

The amount of quantization performed in this coding step is intended to allow a significant amount of correlation in the DC coefficients to be exploited in the compression of these quantized coefficients, while ensuring that the BPE does not spend a large number of bits coding the DC coefficients to very high resolution before encoding any AC coefficient information.

Encoding of quantized DC coefficients is accomplished by applying a version of the Rice coding algorithm (reference [8]) to differences between successive quantized coefficients.

Depending on the relationship between the maximum AC and DC coefficient magnitudes, coding of additional bit planes of DC coefficients following the differentially coded

quantized values may be performed at this step (see subsection 4.3.3 of reference [1]). Bits providing further DC coefficient resolution are included as part of the subsequent bit-plane coding process, specifically as 'stage 0' of the coded bit plane.

2.5.4 AC COEFFICIENT BIT DEPTHS

Next, the BPE encodes the sequence of BitDepthAC_Block$_m$ values for the segment. The value of BitDepthAC_Block$_m$ indicates the number of bits needed to represent the largest magnitude AC coefficient in the m^{th} block. The sequence of values is differentially encoded using the same Rice coding method as for the quantized DC coefficients.

2.5.5 BIT PLANES OF AC COEFFICIENTS

The last step of the BPE stage is bit-plane coding of the AC coefficients (see subsection 4.5 of reference [1]).

Each wavelet coefficient is represented in binary using one sign bit and *R-1* bits to specify the magnitude. Here, *R* represents the maximum number of bits that may be needed to represent a DWT coefficient, and thus *R* is not a parameter that can be arbitrarily set by the user, but rather the value of *R* is determined by the image pixel bit depth and choice of DWT employed; this is described in more detail in 4.2. The number of bits used to represent AC coefficients in a block is typically much less than *R*, and this is the motivation for separately coding the sequence of BitDepthAC_Block$_m$ values for the segment.

A bit plane of a segment consists of all coefficient magnitude bits corresponding to the same bit position in each *R*-bit coefficient word. The BPE successively encodes bit planes in a segment, proceeding from the most-significant to the least-significant bit plane, inserting AC coefficient sign bit values at appropriate points in the encoded data stream. The resulting encoded bitstream constitutes an embedded data format that provides progressive transmission within a segment; DWT coefficient resolution effectively improves by a factor of two as encoding proceeds from one bit plane to the next.

Within a bit plane, AC coding is performed in stages numbered zero through four. Figure 2-12 shows the structure of an encoded bit plane.

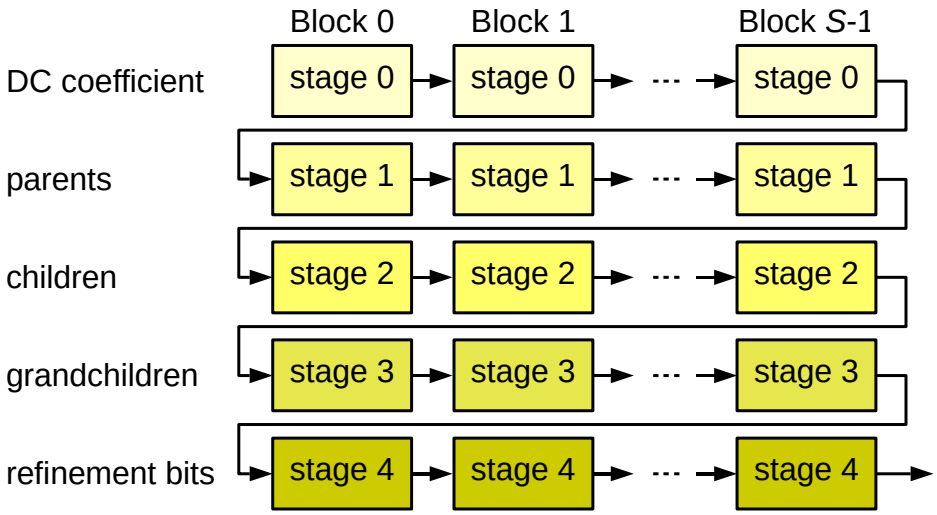


Figure 2-12: Structure of an Encoded Bit Plane

An encoded bit plane of a segment amounts to a series of 'updates' to the values of the coefficients in the segment. The updates to AC coefficients that are not yet significant are described by a series of binary words using a structure that is intended to exploit dependency between coefficients in a block (see subsection 4.5 of reference [1]). These binary words are not all equally likely, and to exploit this fact the words are encoded using one of a handful of variable-length binary codes; the specific code is selected adaptively. The entropy coded data are arranged so that all parent coefficients in the segment are updated first, followed by children, and then grandchildren coefficients, thereby supporting the desired embedded data format (see figures 2-10 and 2-12). Finally, the segment includes (uncompressed) refinement bits for the AC coefficients in the segment for which more significant magnitude bits are not all zero.

3 COMPRESSION OPTIONS AND PARAMETERS

3.1 GENERAL

The selection of compression options and parameters affects compression effectiveness and implementation complexity. For example, figure 3-1 illustrates that two different sets of compression parameters can yield dramatically different rate-distortion performance on the same test image, e.g., more than a factor of two difference in bit rate required to achieve an MSE distortion value of 10.

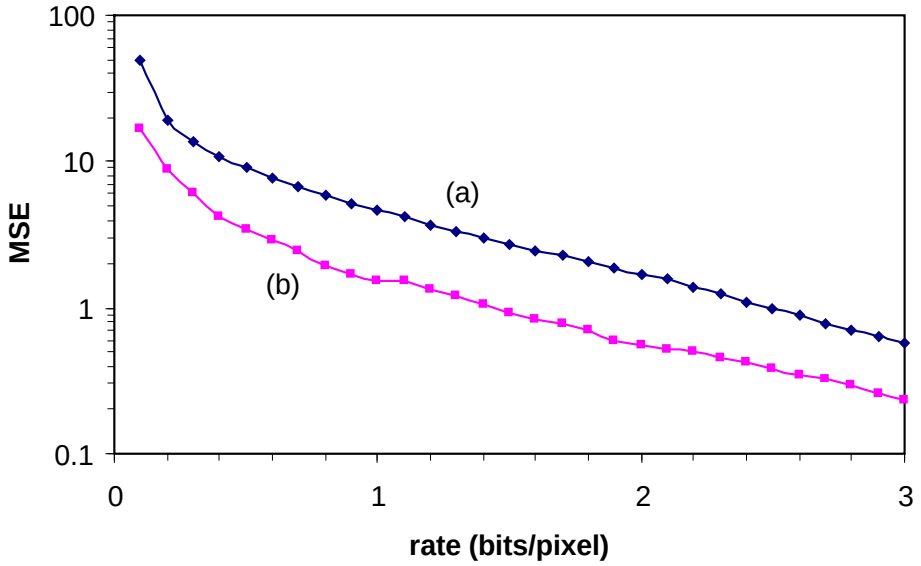


Figure 3-1: Rate Distortion Performance on Transposed Version of coastal_b7 Test Image Using (a) Integer DWT, Fixed-Rate Compression (with UseFill=1), *S*=16, and (b) Float DWT, Full-Frame (*S*=16384) Compression

For a particular application, users and implementers of the Recommendation should determine appropriate selections for compression options and parameters described in this section, including:

- selecting which optional segment headers to include (3.2);
- choosing between the integer and float DWT (3.3);
- adjusting the values of parameters that limit the compressed data volume and reconstructed image fidelity (3.4);
- selecting the number of blocks per segment (3.5);
- deciding between two methods of selecting the Golomb code parameter used in Rice coding (3.6);
- deciding whether to use custom subband weight factors (3.7).

Subsection 3.8 provides examples of uses of the Recommendation.

3.2 SEGMENT HEADERS

The segment header structure is defined in subsection 4.2 of reference [1] and summarized in 2.5.2. Each segment header may have up to four parts; the first part is mandatory, and the other parts are optional. Users must therefore select which of the optional segment header parts to include with each coded segment.

Values encoded in Parts 2 and 3 of the segment header can change from segment to segment, but in many applications one might expect them to be fixed for an image. Values encoded in Part 4 cannot change within an image. Thus, in a typical application, one might include all four header parts for the first segment of an image, and only Part 1 for subsequent segments. In some applications, (e.g., if compression parameters are fixed for an entire mission), one might never include any header parts except for Part 1.

Including optional header parts results in increased bit rate, which may become particularly significant when segments are small and low bit-rate compression is desired. For example, when S=16, including all optional header parts with each segment would contribute an additional 0.125 bits/pixel to the compressed bit rate.

Except where indicated otherwise, compression results in this section assume that all optional headers are included with the first segment, and none of the optional headers are included in subsequent segments.

3.3 CHOICE OF DWT

A user or implementer elects whether to use the float or integer DWT as defined in section 3 of reference [1]. This choice is indicated via the DWTtype field in the optional Part 4 of the segment header.

The float DWT cannot provide lossless compression, and so in an application where perfect image reconstruction is required, the integer DWT must be used. The integer DWT requires only integer arithmetic, while the float DWT requires floating-point calculations. Thus, the integer DWT may be preferable in some applications for complexity reasons.

The float DWT may be preferable in some lossy image compression applications because it often provides better compression effectiveness than the integer DWT at low bit rates. As an example, figure 3-2 illustrates the rate-distortion performance between the float and integer DWTs on a test image. It should be noted that this figure simply presents an illustrative example, and the performance difference between the integer and float DWT will vary depending on the source image.

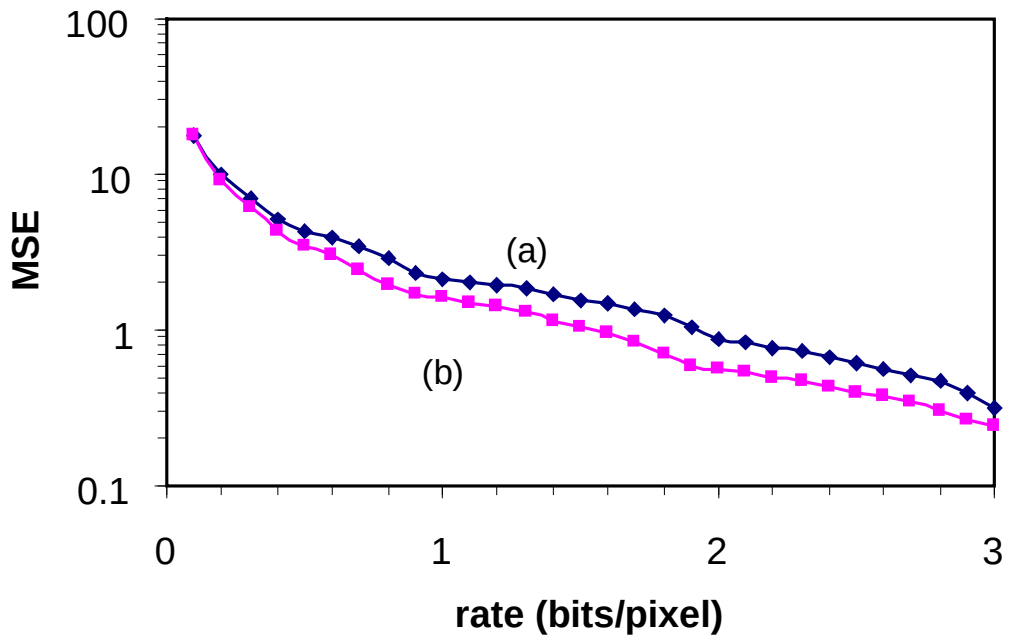


Figure 3-2: Rate-Distortion Performance of (a) Integer and (b) Float DWTs on the coastal_b7 Test Image Using Full-Frame Compression (*S*=16384)

3.4 CONTROLLING COMPRESSED DATA VOLUME AND IMAGE QUALITY

3.4.1 GENERAL

Because rate-distortion performance depends on the source image, it can be challenging to tune the parameters of an image compression algorithm to ensure that images are acceptable in terms of both compressed data volume and reconstructed quality.

To control the tradeoff between bit rate and image distortion, the Recommendation provides methods for specifying limits on compressed data volume and reconstructed image quality. For each segment, compressed data is produced until the segment's data volume limit or quality limit is reached, whichever comes first. These limits are specified using the parameters SegByteLimit, DCStop, BitPlaneStop, and StageStop. The values of these parameters are encoded in the optional Part 2 of the header.

The SegByteLimit parameter provides a direct limit on the number of compressed bytes (including headers) in a segment. Note that SegByteLimit must be an integer multiple of the CodeWordLength parameter described in 3.4.4.

The parameters DCStop, BitPlaneStop, and StageStop limit the amount of DWT coefficient information encoded in a compressed segment, and thus indirectly limit the reconstructed image quality. For this reason, loosely speaking, these three parameters can be said to specify a 'quality' limit. If one can determine (e.g., via experiments) values of these quality parameters that produce reconstructed images of acceptable quality for an application, then

the quality parameters can be set accordingly so that compressed data volume is no larger than needed to obtain the desired image quality.

Figure 3-3 indicates how the values of the DCStop, BitPlaneStop, and StageStop parameters limit the compressed data included in a segment. When DCStop=1, no data is included in a compressed segment beyond the initial coding of DC coefficients described in subsection 4.3 of reference [1]; this tends to permit only a very low fidelity reconstructed image. When DCStop=0, the limit on coded information included in the segment is determined by the BitPlaneStop and StageStop parameters: the value of BitPlaneStop indicates the bit-plane index corresponding to the quality limit, and the value of StageStop determines the last coding stage to be included within the given bit plane.

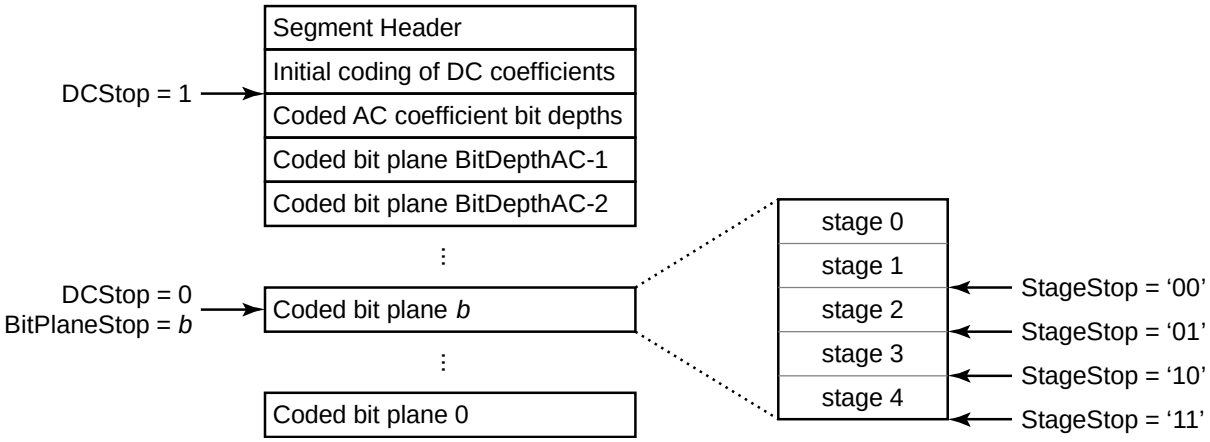
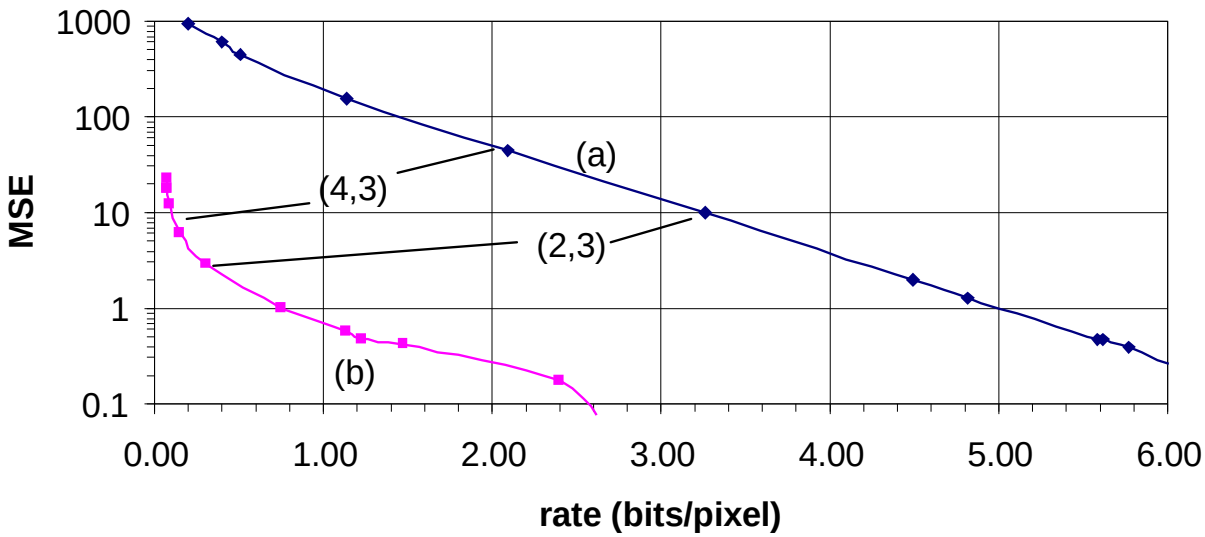


Figure 3-3: Relationship between Stopping Point within a Compressed Image Segment and DCStop, BitPlaneStop, and StageStop Parameters

Note that bit-plane index zero is the last (least significant) bit-plane encoded; i.e., smaller values of BitPlaneStop correspond to higher image quality. For a given value of BitPlaneStop, higher values of StageStop yield higher reconstructed image quality.

Figure 3-4 illustrates rate-distortion points corresponding to different quality limits for two test images. The labeled points in the figure demonstrate that there is not a simple correspondence between the values of the quality-limit parameters and MSE distortion.



NOTE – On each curve, two points are labeled with the corresponding values of the parameters (BitPlaneStop, StageStop).

Figure 3-4: Example of Rate-Distortion Points Corresponding to Different Quality Limits for Two Test Images: (a) europa, (b) coastal_b4

The combination of the byte-limit and quality-limit parameters provide flexibility in controlling compressed data volume and image quality.

For example, one can effectively eliminate the quality limit (by setting DCStop=0, BitPlaneStop=0, StageStop=3) so that the amount of compressed data produced for a segment is simply limited by the value of SegByteLimit.[2] This scenario is referred to as *rate-limited* compression; the rate limit is assumed to be applied uniformly to every segment of the image. If, in addition, the UseFill parameter is set to 1 (see 3.4.4) so that fill bits are added as needed to ensure that each compressed segment consists of exactly SegByteLimit bytes, then that compression is said to be *fixed-rate*.

At the other extreme, if one specifies a sufficiently large value for SegByteLimit, then compression is limited by the quality limit determined by the values of the DCStop, BitPlaneStop, and StageStop parameters. The compression in this case is referred to as *quality-limited*.[3]

[2] In this case, if the integer DWT is used, lossless compression will be achieved whenever the losslessly encoded DWT coefficient information has lower data volume than SegByteLimit for each image segment (see 3.4.2).

[3] Note that the terms *fixed-rate*, *rate-limited*, *quality-limited*, etc. are informal terms used for convenience of explanation in this Report. These terms are not formally defined as part of the Recommendation.

Between the quality-limited and rate-limited extremes, a user can set the byte and quality limits to reflect a desired image quality subject to a segment data volume constraint, without knowing in advance which limit will be reached for a given segment. (An example of this appears in 3.8.)

When an image is compressed using more than one segment, compression effectiveness is lower when the size of compressed segments is limited by a rate limit, than when segment size is controlled by a quality limit. For example, figure 3-5 shows the rate-distortion performance in the rate-limited and quality-limited cases for an image compressed using the float DWT with *S*=512. This difference in compression effectiveness arises because of variations in scene content over an image. Loosely speaking, a portion of an image corresponding to a smooth region will tend to be more amenable to compression than one corresponding to a highly detailed region. Figure 3-6 shows the image used to produce the rate-distortion results shown in figure 3-5. For this image, earlier DWT segments (corresponding to land) are less amenable to compression than later DWT segments (corresponding to water). When each segment is compressed to the same number of bytes, different segments will be reconstructed to different quality levels and overall rate-distortion performance for the image is worse than when compression is controlled by applying a quality limit.

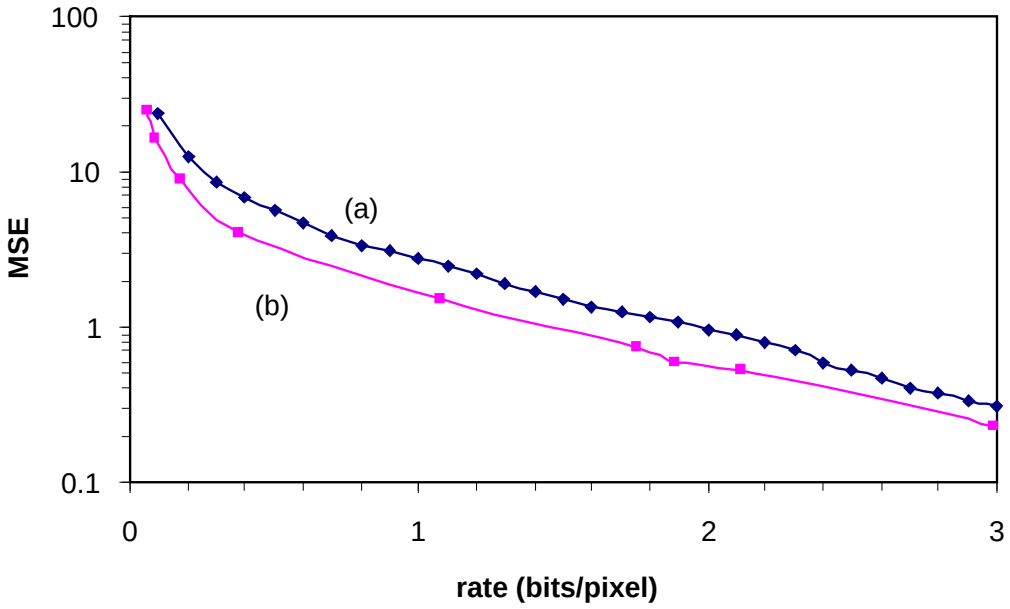


Figure 3-5: Rate-Distortion Performance for Transposed Version of coastal_b7 image Using the Float DWT in (a) Rate-Limited and (b) Quality-Limited Cases when *S*=512

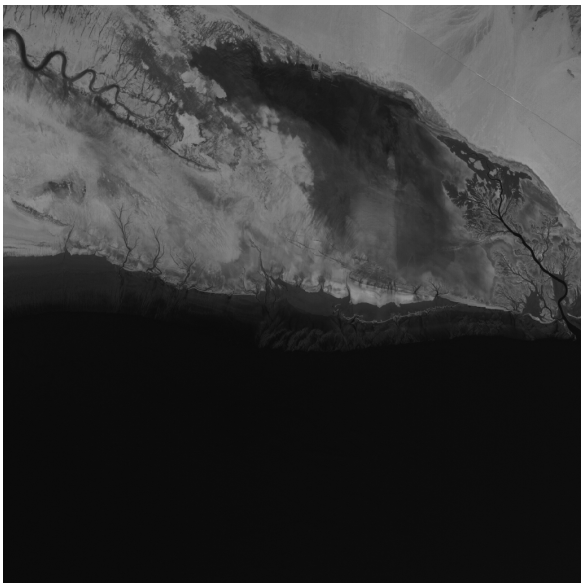


Figure 3-6: Transposed coastal_b7 image Used to Produce Rate-Distortion Curves in Figures 3-1 and 3-5

Under rate-limited compression, reconstructed segments generally do not all have the same fidelity, and overall rate-distortion performance is not generally optimized by allocating the same number of compressed bytes to each segment. Consequently, overall rate-distortion performance tends to be noticeably worse under rate-limited compression compared to quality-limited compression, and this effect tends to be more pronounced when segments are smaller.

Under quality-limited compression, compressed data volume can be difficult to control and predict a priori. But if image segments are all reconstructed to the same quality, overall rate-distortion performance tends to be better than under rate-limited compression, and compression effectiveness tends to be relatively insensitive to segment size.

When multiple segments are desired and one wants to optimize overall image quality subject to an overall (rather than segment-wise) rate constraint, one could employ a strategy to optimize the allocation of compressed bytes to different segments. For example, one might selectively truncate segments (see 3.4.3) in a way that optimizes overall reconstructed image quality subject to a rate constraint. Such an implementation, however, might have high complexity.

The selection of rate and quality parameters might also be influenced by implementation considerations. Specifically, implementation memory requirements tend to increase with the value of SegByteLimit. In addition, compression time may increase with data volume, since a compressor would typically not need to spend time producing compressed bit planes that are not to be included in the compressed segment anyway.

3.4.2 LOSSLESS COMPRESSION

As described above, to achieve lossless compression it is necessary to:

a) use the integer DWT;

b) set DCStop to 0, BitPlaneStop to 0, and StageStop to 3;

c) for each segment, set the value of SegByteLimit sufficiently large to accommodate the compressed data volume needed for losslessly encoding that segment.

Lossless compression is achieved whenever these conditions are satisfied. There is no separate 'lossless compression mode' of the compressor.

As described in item c) above, to guarantee lossless compression, the value of SegByteLimit must be large enough to accommodate the number of compressed bytes produced for the segment. But compressed data volume depends on the input image, and so it is difficult to provide a generic formula for a useful upper bound on SegByteLimit to ensure lossless compression. In fact, one can define pathological images (e.g., produced via random noise processes) that lead to data expansion rather than compression; useful bounds on compressed segment size for such scenarios are not known. In practice, one would typically expect that the value of SegByteLimit should be set to the amount of memory reserved to store the compressed segment: one cannot write a 'blank check' for data storage. Thus practical memory constraints may determine the appropriate value to use for SegByteLimit.

Segment quality and bit rate limits are allowed to change with each segment. A natural case where it would be appropriate to change the value of SegByteLimit within an image is the case where fixed-rate compression is desired and the last segment of an image corresponds to fewer image pixels than the other image segments. An example of this is illustrated in 3.8.

3.4.3 TRUNCATING A COMPRESSED SEGMENT

In addition to the segment byte-limit and quality-limit parameters, one can also control compressed segment data volume by truncating a compressed segment to meet a constraint on data volume. For example, one could obtain a low-fidelity image preview by transmitting a relatively small portion of the compressed data from each segment of an image.

Because of the hierarchical structure of an encoded segment, truncating a segment by increasing amounts tends to yield gradual overall degradation in reconstructed data quality for that segment.

One would not expect the values of the rate and quality limits encoded in Part 2 of the segment header to reflect accurately the rate or quality ultimately achieved by the truncated segment.

3.4.4 CODEWORDLENGTH AND USEFILL PARAMETERS

The value of the CodeWordLength parameter (encoded in the optional segment header Part 4) indicates the unit in which the compressor produces output. CodeWordLength may indicate 8-bit, 16-bit or 32-bit words.

If the number of compressed bits produced for a segment (including headers) is not a multiple of CodeWordLength, then fill bits ('0's) are appended as necessary to make the compressed length equal to an integer multiple of CodeWordLength. For example, a compression implementation that produces one 32-bit word at a time might include as many as 31 fill bits in the last word of the compressed segment.

In certain applications, (e.g., if fixed-rate compression is required) it may be important for the compressed segment length to be exactly equal to SegByteLimit. In this case, the UseFill flag (included in the optional segment header Part 2) should be set. When UseFill is set, and the number of bytes used to meet the segment quality limit is less than the value of SegByteLimit, fill bits ('0's) are appended to the compressed segment data so that the compressed segment length is exactly equal to SegByteLimit.

3.5 NUMBER OF BLOCKS PER SEGMENT

Users or implementers of the Recommendation select the number of blocks per segment, *S*, as described in 2.4.3.2. Each segment is compressed independently. The choice of *S* affects memory requirements, robustness to data errors or losses, and compression effectiveness.

An image with width w and height h generates $\lceil w/8 \rceil \cdot \lceil h/8 \rceil$ DWT coefficient blocks; the blocks can be thought of as an array with width $\lceil w/8 \rceil$ and height $\lceil h/8 \rceil$. When $S = \lceil w/8 \rceil \cdot \lceil h/8 \rceil$, the entire image is compressed as a single segment and that compression is said to be *full-frame*. When $S=\lceil w/8 \rceil$, each image segment loosely corresponds to a thin horizontal strip of the image, and it is said that *strip* compression is being performed. Strip compression can lead to a relatively memory-efficient implementation and may be convenient, e.g., for imagers produced by push-broom type sensors. As discussed below, however, in some circumstances compression effectiveness may be significantly improved when larger values of *S* can be used.

The minimum value of *S* is 16, except for the last segment of an image, which may consist of as few as a single block (S=1). This limitation prevents users from using very small segments, which would tend to degrade compression effectiveness. The maximum value of *S* is constrained by the number of blocks produced for an image and the 20-bit field used to encode the value of *S* in the optional Part 3 of the segment header. Thus the maximum value of *S* is $\min\left\{\left\lceil \frac{w}{8} \right\rceil \cdot \left\lceil \frac{h}{8} \right\rceil, \quad 2^{20}\right\}$.

Larger values of *S* generally lead to higher memory requirements, because the BPE coding process requires the availability of a complete segment.

Because segments are compressed independently, a bit error or data loss affecting one segment will not affect the ability to decompress other segments. Thus, smaller values of *S* tend to provide increased robustness to data loss or errors. However, when selecting segment size, one should keep in mind the data loss or corruption mechanisms likely to be encountered in practice. For example, if compressed segments are transmitted using large packets, and the dominant loss event is the occasional loss of a packet, it may not be appropriate to select a segment size that results in compressed segments that are very small compared to the packet length, since a single packet loss will lead to the loss of multiple segments anyway.

The relative cost of optional header parts increases when smaller segments are used. For example, when all optional headers are included with each segment, lossless compression of the coastal_b7 test image requires 470381 bytes (3.59 bits/pixel) when *S*=16, but only 452314 bytes (3.45 bits/pixel) when *S*=512; the difference is almost entirely due to the cost of optional headers. The relative difference becomes more pronounced at low bit rates: figure 3-7 shows the noticeably different rate-distortion performance when *S*=16 and *S*=512 for this same image when all optional header parts are included and compression is quality-limited. When quality-limited compression is used and optional headers are used sparingly, segment size tends to have little impact on compression effectiveness.

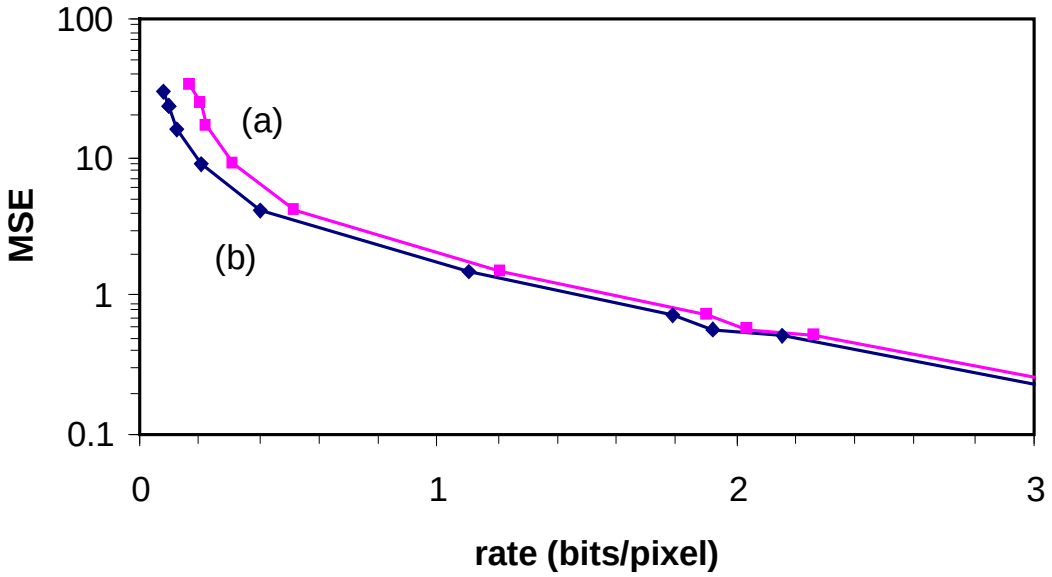


Figure 3-7: Rate-Distortion Performance for the coastal_b7 Test Image Using the Float DWT when (a) *S*=16 and (b) *S*=512 when Compression is Quality-Limited and all Optional Headers are Included

More significantly, the difference in compression effectiveness between quality-limited and rate-limited compression (previously illustrated in figure 3-5) tends to become more pronounced when segments are smaller; in fact, quality-limited and rate-limited compression have equal compression effectiveness when full-frame compression is used.

3.6 RICE CODE PARAMETER SELECTION METHOD

In an implementation of the Recommendation, one can choose between two different adaptive code parameter selection methods that can be used during segment encoding, as described in subsection 4.3.2.11 of reference [1]. The coding parameter selection methods used for these two integer sequences are indicated in the values of OptDCSelect, OptACSelect bits in the optional Part 3 of the segment header.

As part of the segment encoding process, a simple differential coding procedure is used for losslessly encoding two sequences of integers. In the first instance, the integers represent quantized DC coefficient values (see 2.5.3), and in the second instance the integers indicate the number of bits needed to represent the largest AC coefficient magnitude in each block of the segment (see 2.5.4).

To encode either of the integer sequences, differences between consecutive values in the sequence are computed. The sequence of differences is partitioned into smaller sequences referred to as 'gaggles' (see subsection 4.3.2 of reference [1]). For each gaggle, one of several Golomb codes (see reference [9]) is used to encode the difference values in the gaggle; the particular code is selected adaptively based on the difference values in the gaggle and the code selected (indicated by the value of the parameter k; see subsection 4.3.2 of reference [1]) is explicitly encoded in the compressed datastream.

The procedure of partitioning a sequences of samples, or sample differences, into smaller blocks, and adaptively selecting one of several Golomb codes for losslessly encoding each partition is referred to as Rice coding (see reference [8]). Rice coding is the basis for an earlier CCSDS data compression recommendation (reference [4]).

There are two adaptive methods that may be used to select the Golomb code (i.e., the value of the parameter k) to use for each gaggle. For each gaggle, the value of k can be selected optimally (i.e., selecting the code option that minimizes the encoded length for the gaggle), as is normally done in the Rice coding by exhaustively trying each code option. Alternatively, the Recommendation allows the code parameter for each gaggle to be selected by applying a computationally simpler but sometimes suboptimal parameter selection procedure developed in reference [10]. In either case, the parameter value selected is encoded in the compressed data stream, so the decompressor can decode the data without knowing which procedure was used.

As indicated in reference [10], the difference in compression effectiveness between the optimal and heuristic parameter selection methods is likely to be negligible. For example, on the test image set, the average effect of using the heuristic parameter selection for DC and AC coding is an increase of 0.02 bits/pixel (0.5%) in bit rate to achieve lossless compression when full-frame compression is used. Selecting whether to use optimal or heuristic parameter selection is likely to be a matter of implementation complexity.

3.7 CUSTOM SUBBAND WEIGHTS

When the integer DWT is used, each DWT coefficient is scaled by a weighting factor, as described in subsection 3.9 of reference [1]. That is, each DWT coefficient in a subband is multiplied by the same constant weight factor, which can take on values 1, 2, 4, or 8. The allowed weight factors are powers of two so that the multiplications used in the weighting process can be performed by bit shift operations.

Subband weight factors may be assigned by using the default weight factors specified in table 3-3 of reference [1], or a user may select custom weight factors. Subband weighting is not used in combination with the float DWT.

The subband weight factors determine the relative order in which bit planes from different subbands are encoded, which in turn affects compression effectiveness.

The default set of weights were chosen in an effort to minimize the MSE image distortion obtained at a given rate. The selection of weight factors is based primarily on empirical results described in 6.4.4.

A custom set of weights might be appropriate, for example, to optimize a different image quality metric, and may be based on experiments with images from a particular instrument of interest. Methods for selecting a custom set of weights is beyond the scope of this Report.

When Part 4 of the segment header is included, it flags the use of custom weight values and indicates the values of those weights. Note, however, that Part 4 of the segment header is optional even when custom weights are used. For example, if the same set of custom weights were used for an entire mission, one might elect not to encode the weight values in compressed images.

3.8 EXAMPLES

This subsection provides image compression examples illustrating the use of compression parameters to control the compressed data output and the representation of these parameters in segment headers. For testing and verification purposes, all of the encoded files produced in these examples can be obtained from http://cwe.ccsds.org/sls/docs/sls-dc/. Files of DWT coefficients are also available for the examples.

Six different compression examples are presented in the following subsections. Table 3-1 summarizes the input images used in the examples, the total number of blocks in the input image, and the number of blocks per segment (S) selected. In each case, the source image is one of the reference test images described in annex B.

Table 3-1: Example Source Images and Parameters

	Example					
	1	**2**	**3**	**4**	**5**	**6**
source image	small region from marstest	foc	marstest	europa	India_2kb4	spot-la_b3
dimensions (w x h)	32x32	1024x512	512x512	557x600	2048x2048	500x500
bit depth (bits/pixel)	8	12	8	8	10	8
total blocks	16	8192	4096	5250	65536	3969
blocks per segment (*S*)	16	128	64	350	1024	63

The values in the header for the first segment of each example are shown in table 3-2. In each of the examples, election is made to include all optional segment header parts (Parts 2, 3, and 4) with the first segment, and none of the optional headers in subsequent segments. In the table, header field values shaded in gray are not user adjustable parameters: StartImgFlag and EndImgFlag identify the first and last segments in an image; SegmentCount provides an index for each segment; and the values of BitDepthAC and BitDepthDC are determined during compression based on image data encountered in the segment. Fields shaded in green indicate basic input image parameters that can be computed from the values given in table 3-1. Refer to subsection 4.2 of reference [1] for proper binary representation of each value encoded in the header; refer to annex A for details.

Table 3-2: Values of Fields in the First Segment Header for Each Compression Example

Part	Field Name	Field Value					
		Ex. 1	**Ex. 2**	**Ex. 3**	**Ex. 4**	**Ex. 5**	**Ex. 6**
Part 1A	StartImgFlag	1	1	1	1	1	1
	EndImgFlag	1	0	0	0	0	0
	SegmentCount	0	0	0	0	0	0
	BitDepthDC	12	4	12	12	14	12
	BitDepthAC	10	4	10	10	10	9
	Part2Flag	1	1	1	1	1	1
	Part3Flag	1	1	1	1	1	1
	Part4Flag	1	1	1	1	1	1
1B	PadRows*	0	0	0	0	0	4
Part 2	SegByteLimit**	2^{27}	2^{27}	2^{27}	18381	5718	3000
	DCStop	0	0	*varies*	0	0	0
	BitPlaneStop	0	0	*varies*	0	4	0
	StageStop	3	3	*varies*	3	3	3
	UseFill	0	0	0	0	0	1
Part 3	*S*	16	128	64	350	1024	63
	OptDCSelect	1	1	1	1	1	1
	OptACSelect	1	1	1	1	1	1
Part 4	DWTtype	1	1	1	1	0	1
	SignedPixels	0	0	0	0	0	0
	PixelBitDepth	8	12	8	8	10	16
	ImageWidth	32	1024	512	557	2048	500
	TransposeImg	0	0	0	0	0	0
	CodeWordLength	8	8	8	8	8	8
	CustomWtFlag	0	0	0	0	0	0
	custom subband weight values	0	0	0	0	0	0

*Note that Header Part 1B is included only for the last segment of an image.

** Coded modulo 2^{27}.

In all of the examples, optimal code parameter selection is used in coding the quantized DC coefficients and the BitDepthAC values, i.e., OptDCSelect and OptACSelect are set to 1 in each example (see 3.6).

Example 1: Lossless Full-Frame Compression of a Small Image

In Example 1 the input image consists of a 32x32 pixel region extracted from the marstest image (specifically, the first 32 rows of the first 32 columns of the image).

The entire image consists of 16 blocks, which is the minimum value of the parameter *S*, and thus *S* is set to 16. The entire image is compressed using a single segment, and consequently the segment header includes Part 1B.

In this example, the image is compressed losslessly. As described in 3.4.2, to achieve lossless compression, the integer DWT is used, and set the values of the quality-limit parameters DCStop to 0, BitPlaneStop to 0, and StageStop to 3.

A total of 677 compressed bytes are produced. Note that the number of compressed bytes included in the segment must be an integer multiple of the word size indicated by the value of CodeWordLength. Thus, e.g., if CodeWordLength indicates 16-bit (2-byte) words, then it would be necessary to use 678 bytes to achieve lossless compression (the last byte would consist entirely of fill bits); when CodeWordLength indicates 8-bit (1-byte) words, 677 bytes would be needed. Thus, if it is assumed that single-byte words are used, a value of 677 or larger for SegByteLimit will yield lossless compression in this example. For convenience, SegByteLimit can be set to the maximum allowed value of 2^{27}, which is encoded mod 2^{27} in the header as 'zero'.

Example 2: Lossless Strip Compression

Example 2 applies 'strip' compression (refer to 3.5) to the 1024x512 foc image. The image width corresponds to 128 blocks, and so *S* is set to 128 for strip compression.

Example 3: Quality-Limited Lossy Strip Compression

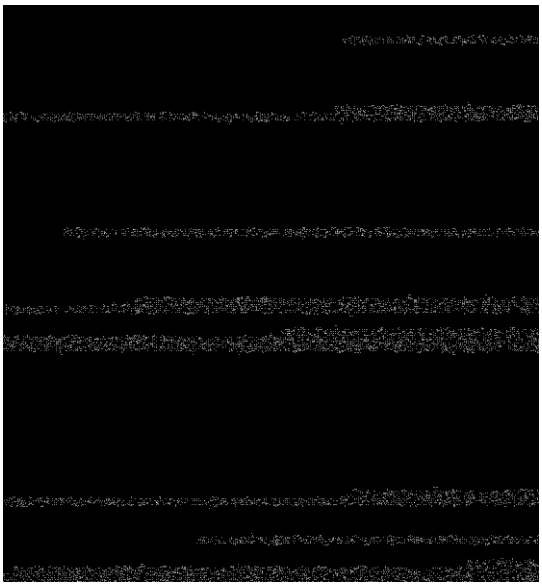
Example 3 illustrates quality-limited lossy compression on the marstest image when the integer DWT is used. In this example strip compression is used by setting *S* to 64. SegByteLimit is set to the maximum allowed value of 2^{27}, which is encoded in the header as zero. This ensures that compression is quality-limited.

The quality-limit parameters (DCStop, BitPlaneStop, StageStop) are specified in segment header Part 2. (In the example these parameters are varied to produce compressed images at several different quality levels.)

Example 4: Fixed-Rate Lossy Strip Compression

A rate-limited case is illustrated in Example 4 by setting the SegByteLimit to a large value which will cause some segments to be coded losslessly at less than SegByteLimit bytes while others are encoded at the fixed rate. This is achievable by setting UseFill to 0 and using integer DWT.

The europa image (557x600) is used with the integer DWT. The segment size is set at *S*=350, producing 15 total segments. Under lossless compression, the compressed image requires 275757 bytes. When SegByteLimit=18381 is used, the compressed image has 274357 bytes and compression is not lossless. However, as illustrated in the difference image shown in figure 3-8, significant portions of the image are error-free because some segments are completely encoded within the segment byte limit.



NOTE – Black regions of the image correspond to distortion-free reconstruction.

Figure 3-8: Magnitude Difference Image for Example 4

Example 5: Lossy Compression Including Both Rate-Limited and Quality-Limited Segments

Example 5 illustrates lossy compression in a case where compressed segment size is rate-limited for some segments and quality-limited for other segments. The test image India_2kb4, which has 2048x2048 pixels, is used. By setting *S* = 1024, each segment consists of four strips. UseFill is set to 0, SegByteLimit to 5718 (about 0.7 bits/pixel), and the quality parameters are set to BitPlaneStop=4, StageStop=3. With this choice of compression parameters, the compressed image has size 301239 bytes (about 0.65 bits/pixel) which is less than the allowed byte limit for the entire image, which is 365952 bytes. This indicates that not all of the segments are limited by the value of SegByteLimit; i.e., at least some of them must be limited by the quality parameters.

To show that not all segments are limited by the quality parameters, SegByteLimit can be increased to 6500 bytes without changing the other parameters. This change causes the compressed file size to increase to 323819 bytes. The fact that the compressed data volume increases shows that the smaller value of SegByteLimit was a limiting factor in the compressed size for at least some of the segments.

Example 6: Calculating SegByteLimit for Fixed-Rate Lossy Compression

Example 6 applies fixed-rate compression to the 500x500 spot-la_b3.raw image using the integer DWT. S is set to $\lceil 500/8 \rceil = 63$ for strip compression. The quality-limit parameters are set to ensure that compression is rate-limited and set UseFill to 1 to ensure fixed-rate compression.

Next, the appropriate value of SegByteLimit to achieve compression to six bits/pixel is calculated. Note that in the calculation of the wavelet transform of the image, four additional rows and columns are first appended to the image so that the padded image has width and height that are both multiples of eight (refer to reference [1], subsection 3.2). Consequently, every segment corresponds to 500x8 pixels, except for the last, which corresponds to half as many pixels, because of the padded rows. Thus, to achieve six bits/pixel compression, 3000 bytes are required for all but the last segment, for which 1500 bytes are required.

During encoding, the different value of SegByteLimit for the last segment can be simply included in the Part 2 header for the last segment. Alternatively, the encoding module can choose to stop encoding after 1500 bytes are reached, without including the Part 2 header in the coded bit stream. During decoding, the process stops when all 1500 bytes are exhausted for the last segment.

4 IMPLEMENTATION ISSUES

4.1 INTRODUCTION

This section presents information that may be useful when implementing the Recommendation in hardware or software. Subsection 4.2 discusses the number of bits needed to store each DWT coefficient in memory. Subsection 4.3 describes memory-efficient calculation of the 2-d DWT. Subsection 4.4 discusses the problem of reconstructing a compressed image given a set of quantized DWT coefficients. Subsection 4.5 briefly discusses pixel readout correction.

Another concern for hardware implementations is memory bandwidth, which is not addressed in this Report. Memory bandwidth is discussed in reference [12].

4.2 DYNAMIC RANGE EXPANSION

4.2.1 GENERAL

The dynamic range of output from a DWT can be larger than the dynamic range of input data. Such an increase is referred to as *dynamic range expansion*. When implementing the Recommendation, consideration of dynamic range expansion is necessary to determine word sizes needed to store DWT coefficients. For example, when the Float DWT is applied to an input image having 16-bit unsigned integer pixels, representation of some DWT coefficients may require 21 bits.

In 4.2.2 the amount of dynamic range expansion under certain scenarios is tabulated. Subsection 4.2.3 describes how to produce a test image that can be used to determine the amount of dynamic range expansion given a constraint on the dynamic range of the pixels in the input image.

This discussion of dynamic range expansion considers only the size of the binary words used to store final integer output of the DWT calculation,[4] but not the size of the registers needed to perform any of the intermediate calculations. In addition, it does not take into account the scaling factors applied to DWT coefficients when the integer DWT is used (see subsection 3.9 of reference [1]) because (a) in practice, to conserve memory, one could simply keep track of the number of bit planes to be shifted for each subband (rather than using larger words to accommodate the scaling), and (b) the number of bit planes shifted can change when a user uses custom subband weights.

[4] In this context, the round-off operations used to convert Float DWT output to integer values, as described in subsection 3.1 of reference [1], are treated as part of the DWT calculation process.

4.2.2 OVERVIEW

Table 4-1 indicates the maximum DWT coefficient word size for the different possible combinations of input image bit depth and DWT type. Table 4-1 indicates, e.g., that 21-bit words are adequate to store DWT coefficients when the input is a 16-bit image, but smaller words may be used in an application exclusively for 8-bit images. When the Integer DWT is used, the maximum DWT coefficient word size does not depend on whether the input image pixels are signed or unsigned.

Table 4-1: Word Size Needed to Store DWT Coefficients As a Function of Input Image Bit Depth and DWT

Input image dynamic range (bits/pixel)	Maximum DWT coefficient word size (bits)		
	Integer DWT	Float DWT	
		unsigned image	signed image
1	4	5	5
2	5	7	6
3	6	8	7
4	7	9	8
5	8	10	9
6	10	11	10
7	11	12	11
8	12	13	12
9	13	14	13
10	14	15	14
11	15	16	15
12	16	17	16
13	17	18	17
14	18	19	18
15	19	20	19
16	20	21	20

Not all subbands are susceptible to the same amount of dynamic range expansion. Table 4-1 indicates the worst-case expansion over all subbands, but in principle one could conserve memory by using different word sizes to store coefficients from different subbands. For the integer DWT, worst-case expansion occurs in the HH_3 subband; for the float DWT, worst-case dynamic range expansion occurs in the LL_3 subband. Table 4-2 indicates the amount of dynamic range expansion in each subband when the input image is a 16-bit unsigned image. It can be seen, e.g., that when the Integer DWT is applied to a 16-bit image, 20-bit words are needed only for the HH_3 subband. For other image bit depths, results analogous to those in table 4-2 can be computed using the dynamic range test image described in 4.2.3.

Table 4-2: Dynamic Range Expansion for 16-Bit Unsigned Input Images

	Integer DWT		Float DWT	
Subband	**Output Range**	**Maximum output bit depth**	**Output Range**	**Maximum output bit depth**
LL_3	[-50564, 116098]	18	[-185483, 709763]	21
HL_3	[-149512, 149510]	19	[-436993, 436993]	20
LH_3	[-149513, 149512]	19	[-436993, 436993]	20
HH_3	[-268252, 268251]	20	[-426616, 426616]	20
LL_2	[-48134, 113668]	18	[-101755, 363895]	20
HL_2	[-141188, 141187]	19	[-229302, 229302]	19
LH_2	[-141188, 141186]	19	[-229302, 229302]	19
HH_2	[-246398, 246398]	19	[-225832, 225832]	19
LL_1	[-40960, 106493]	18	[-59333, 190403]	19
HL_1	[-110591, 110590]	18	[-117385, 117385]	18
LH_1	[-110590, 110590]	18	[-117385, 117385]	18
HH_1	[-165886, 165886]	19	[-110351, 110351]	18

At each stage of decomposition, when the Float DWT is used, the HL and LH subbands have the same symmetric range of possible output values. This is not true when the Integer DWT is used because of internal round-off operations performed in the calculation.

Subbands LL_1 and LL_2 are not encoded by the BPE, but are simply used as intermediate results to calculate DWT coefficients at subsequent decomposition stages. When applying the Float DWT, it would not be necessary for coefficients in these subbands to be rounded to integer values, and so presumably the binary word size is irrelevant for these subbands. Table 4-2 includes entries pertaining to these subbands for completeness.

4.2.3 DYNAMIC RANGE TEST IMAGE

A template to produce a *dynamic range test image* has been created. Given a constraint on the range of pixel values in the input image (i.e., given a range of possible pixel values [p_{min}, p_{max}]), the template can be used to create a test image that, when decomposed using either the Float or Integer DWT, produces DWT coefficients that obtain the maximum and minimum possible value in each subband. Such a test image is intended primarily for two purposes:

a) When developing an implementation of the compression Recommendation, such a test image can be used to determine the word sizes necessary to accommodate the full range of DWT coefficient values that could be produced in each subband. That is, results analogous to those in 4.2.2 can be produced for a particular application scenario.

b) Given an implementation of the compression Recommendation, such a test image can be used to verify that the implementation has subband word sizes that are adequate to accommodate the full range of DWT coefficient values that could be produced.

The template consists of several two-dimensional patterns; each pattern minimizes or maximizes the value of a particular DWT coefficient in a particular subband.[5] The template is illustrated schematically in figure 4-1; red and green regions indicate pixels that should be set to maximum (p_{max}) and minimum (p_{min}) pixel values, respectively, to produce a dynamic range test image for a particular application. All other pixels can be set to any value in the range [p_{min}, p_{max}].

The superimposed labels in figure 4-1 indicate the function of each pattern. For example, the label 'HL_2 f^+,i^-' indicates that the associated pattern produces a maximum (+) value DWT coefficient in the HL_2 subband when the Float DWT (f) is used, and a minimum (-) value coefficient in the HL_2 subband when the Integer DWT (i) is used.

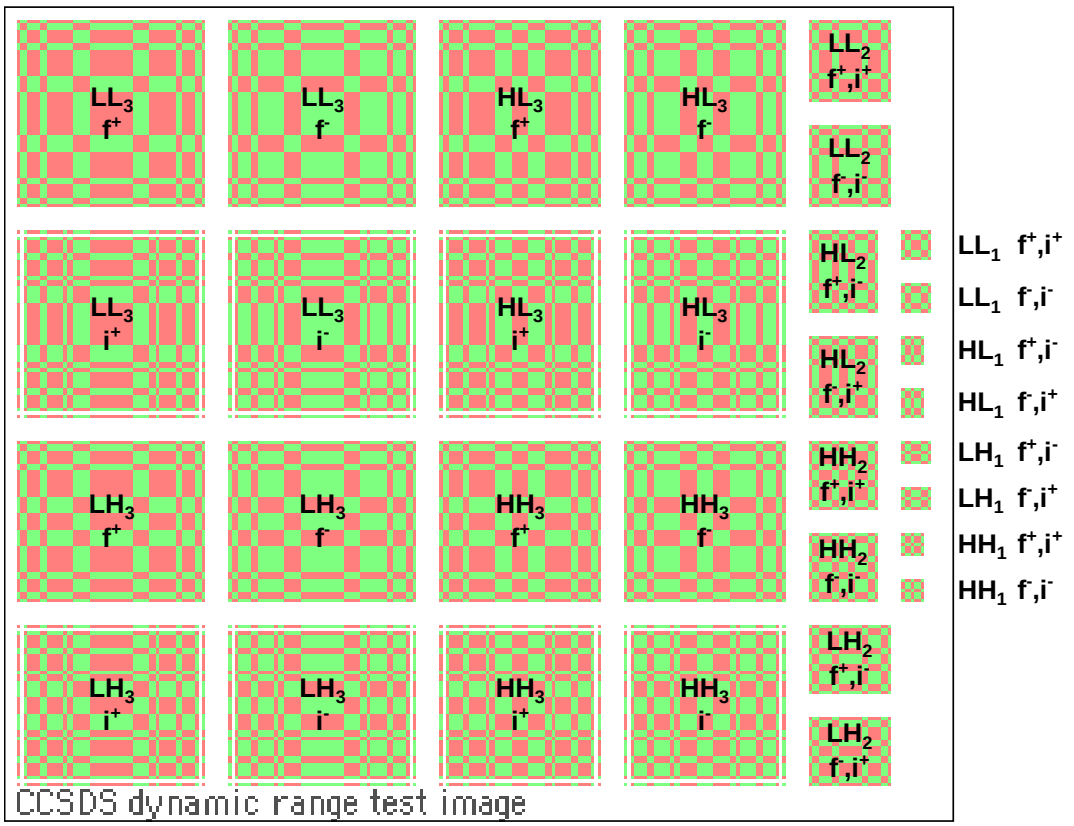


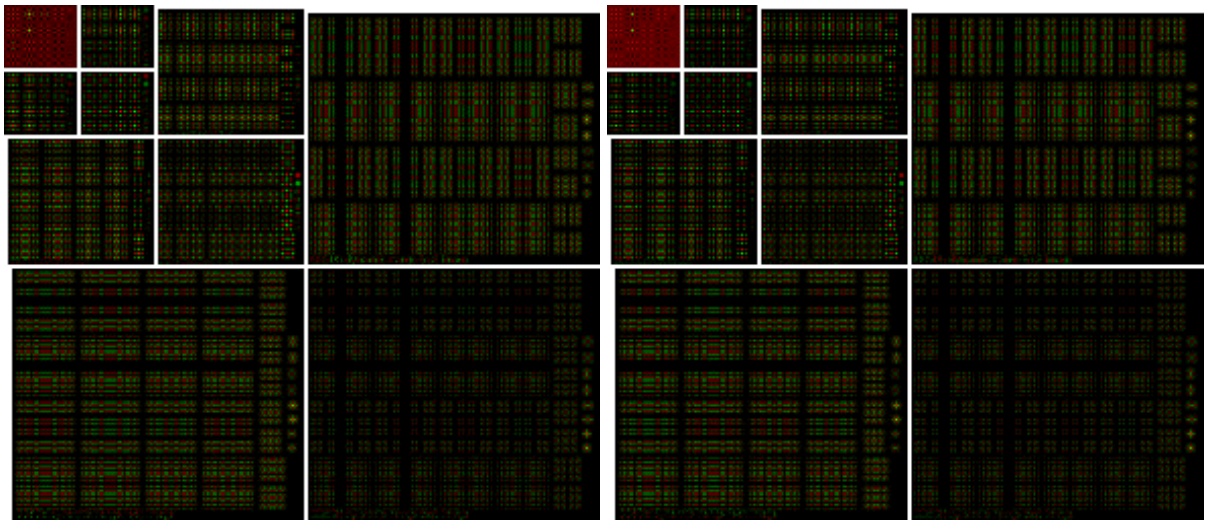
Figure 4-1: Schematic Diagram of Dynamic Range Test Image Template

[5]For completeness, the test images generated using the template produce maximum dynamic range expansion in all the subbands, including LL1 and LL2, even though those subbands are never encoded in the compressed data stream, but only produced as intermediate results.

The dynamic range test image template can be obtained from http://cwe.ccsds.org/sls/docs/sls-dc/. The template is a 288 x 248 image using one byte per pixel. A byte value of 255 indicates a pixel that should be set to p_{max} (shown in red in figure 4-1), and a byte value of zero indicates a pixel that should be set to p_{min} (shown in green in figure 4-1).

Figure 4-2 illustrates a colorized version of the DWT output (in the usual subband arrangement) when Integer and Float transforms are applied to a test image produced from the template. Positive and negative DWT coefficients are shaded red and green, respectively, with a brightness that is proportional to the magnitude of the coefficient.[6] Thus, in each subband depicted in figure 4-2, the brightest red and the brightest green points indicate the location within the subband of the maximum and minimum DWT coefficient values, respectively. Coordinates within each subband of the minimum and maximum valued DWT coefficient are also listed in table 4-3.

In figure 4-1, the patterns labeled 'LL_1 f^+,i^+' and 'L_1 f^-,i^-' are contained in the patterns labeled 'HH_2 f^+,i^+' and 'HH_2 f^-, i^-', respectively. This is reflected in table 4-3, which indicates that a maximum (and minimum) DWT coefficient is obtained twice in subband LL_1.



NOTE – Positive and negative coefficients are shaded red and green, respectively. Each subband is scaled independently.

Figure 4-2: Illustration of Integer (Left) and Float (Right) DWT Coefficients Produced by the Test Image

[6] The brightness scales in figure 4-2 are determined for each subband independently.

Table 4-3: Position (Row, Column) within Each Subband of Minimum and Maximum DWT Coefficient Generated by Test Image

Subband	Integer DWT		Float DWT	
	Max Pos.	Min Pos.	Max Pos.	Min Pos.
LL_3	(12,4)	(12,12)	(4,4)	(4,12)
HL_3	(12,19)	(12,26)	(4,26)	(4,19)
LH_3	(26,4)	(26,12)	(19,12)	(19,4)
HH_3	(26,19)	(26,26)	(19,19)	(19,26)
LL_2	(4,64)	(12,64)	(4,64)	(12,64)
HL_2	(28,63)	(20,63)	(20,63)	(28,63)
LH_2	(56,64)	(49,64)	(49,64)	(56,64)
HH_2	(35,63)	(42,63)	(35,63)	(42,63)
LL_1	(36,138) (71,127)	(44,138) (85,127)	(36,138), (71,127)	(44,138), (85,127)
HL_1	(60,137)	(52,137)	(52,137)	(60,137)
LH_1	(74,138)	(67,138)	(67,138)	(74,138)
HH_1	(81,137)	(88,137)	(81,137)	(88,137)

4.3 MEMORY-EFFICIENT DWT CALCULATION

4.3.1 GENERAL

A single DWT coefficient depends only on a relatively small cluster of image pixels. An important consequence of this fact is that when full-frame compression is not being performed, calculation of DWT coefficients in a segment requires only a portion of the pixels in the original image. Thus, *it is generally not necessary to store a complete frame of image or DWT data when full-frame compression is not being performed*. This fact becomes especially important when trying to conserve implementation memory for large frame or push-broom imager applications.

This subsection describes a memory-efficient approach to calculating DWT coefficients corresponding to a 'strip' of blocks (i.e., a single horizontal row of blocks; see 3.5) for an image with width w. For example, in a compression application for a push-broom imager, this approach might be used to produce a strip of blocks as soon as sufficient image rows have been produced from the imager.

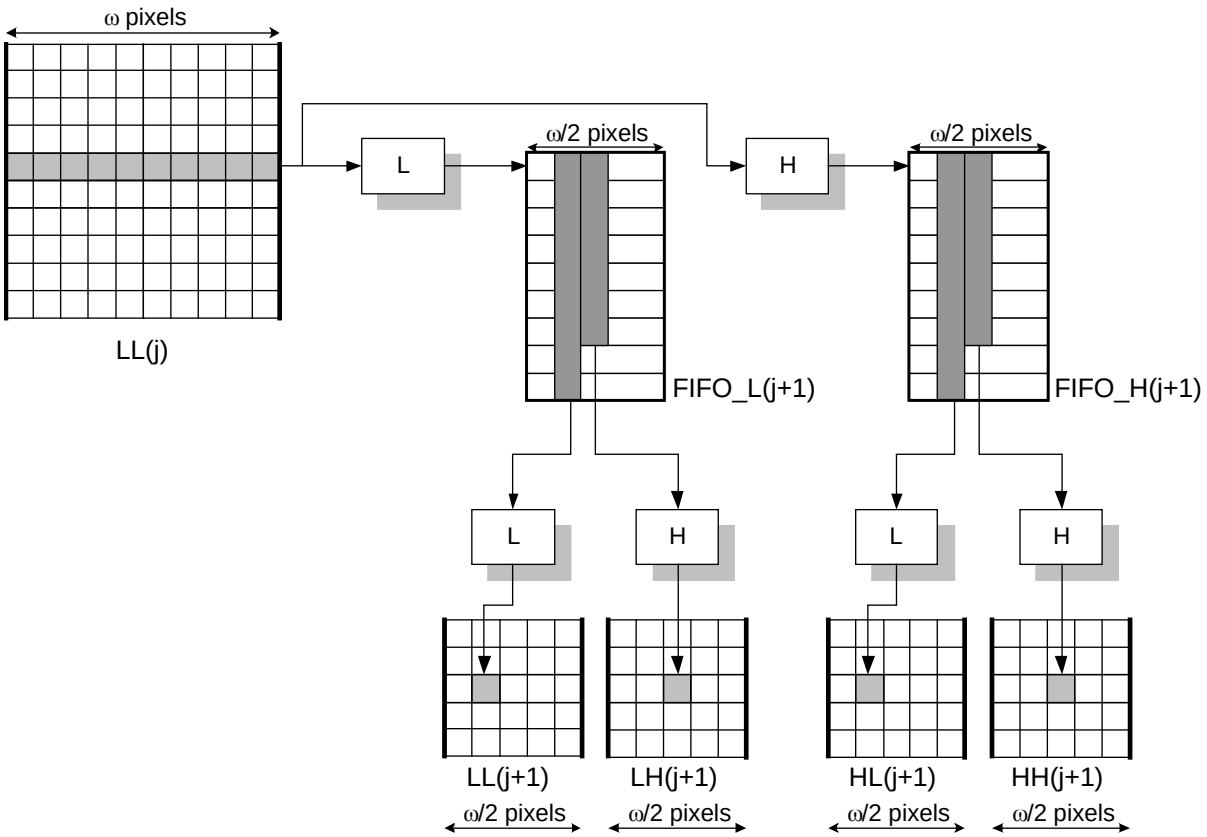


Figure 4-3: Buffering Mechanism for Single Level, 2-d DWT with 9/7 Filters

The following paragraphs first describe how a single level of 2-d DWT is implemented, and then proceed with the data flow and functional block diagrams for the entire transform.

4.3.2 SINGLE LEVEL DWT

The input to the $(j+1)^{st}$ level of transform are data rows from the j^{th} LL-subband, LL_j. A row from this subband has width $\omega = \frac{w}{2^j}$ pixels. The output are data rows of four subbands at level j+1: LL_{j+1}, LH_{j+1}, HL_{j+1}, HH_{j+1} (cf. figure 4-3). The considerations for this generic level of transform apply to any level index j=0, 1, 2 if the convention is adopted that LL_0 denotes the original image.

Whenever a new line from LL_j has arrived, it is subjected to the horizontal low- and high-pass filtering procedures of the 1-d DWT, represented in figure 4-3 by the respective functional blocks L and H. The resulting two data lines are pushed on two respective First-In-First-Out (FIFO) queues: the low-frequency coefficients on FIFO_L(j+1) and the high-frequency coefficients on FIFO_H(j+1). Each FIFO is nine rows deep, i.e. has a fixed capacity of $9\times\omega/2$ samples, in order to accommodate vertical low- and high-pass filters with nine and seven taps, respectively. The FIFOs work like a queue: when a new data row is pushed on a FIFO at the top, an older row drops out at the bottom and is deleted.

Every time *two* new lines have arrived from LL_j, each of the ω /2 columns of each FIFO is subjected to the 1-d low- and high-pass filtering operations L and H. The resulting samples constitute the next rows of the four subbands LL_{j+1}, LH_{j+1}, HL_{j+1}, HH_{j+1}. This means the clock rate at which rows are taken from LL_j is twice the clock rate at which rows are fed to each of the subbands LL_{j+1}, LH_{j+1}, HL_{j+1}, HH_{j+1}. In summary, every new couple of rows from the LL subband of some level produces one row (of half the original width) of each of the four subbands of the next level. The total FIFO capacity of one level is $2\times9\times\omega/2 = 9\times\omega$ samples.

4.3.3 THREE-LEVEL DWT

Figure 4-4 illustrates the concatenation of all three levels of the recommended DWT. The figure shows functions (boxes) and buffers (drums). The buffers P, Q, R, S, T, U are FIFO buffers with the following sizes:

$$\text{dimension}(P) = \text{dimension}(Q) = 9\frac{w}{2} \text{ samples}$$

$$\text{dimension}(R) = \text{dimension}(S) = 9\frac{w}{4} \text{ samples}$$

$$\text{dimension}(T) = \text{dimension}(U) = 9\frac{w}{8} \text{ samples}$$

The wavelet coefficient buffers $HH_1,\ldots,LL_3$ store the respective subband data. The complete set of 10 buffers constitute the Segment Block Buffer. The subband data is accumulated continuously, at different line rates. Once a segment of wavelet coefficient blocks has been encoded by the BPE, all corresponding subband data is discarded.

The filling state of the Segment Block Buffer changes with time. Because of the pipeline nature of the implementation, the maximum size requirement of the Segment Block Buffer is larger than the size needed simply to store all of the *S* blocks for a segment. The Segment Block Buffer size requirement is determined in 4.3.4.

There are four different clock domains in figure 4-4, controlled by the counters n, n', n'', and n'''.

- functions L1 and H1 produce lines at the line rate of the imaging sensor, counted by n;
- functions H2, L2, H3, L3, H4, L4 produce lines at half the line rate of the imaging sensor, counted by n';
- functions H5, L5, H6, L6, H7, L7 produce lines at one quarter the line rate of the imaging sensor, counted by n'';
- functions H8, L8, H9, L9 produce lines at one eighth the line rate of the imaging sensor, counted by n'''.

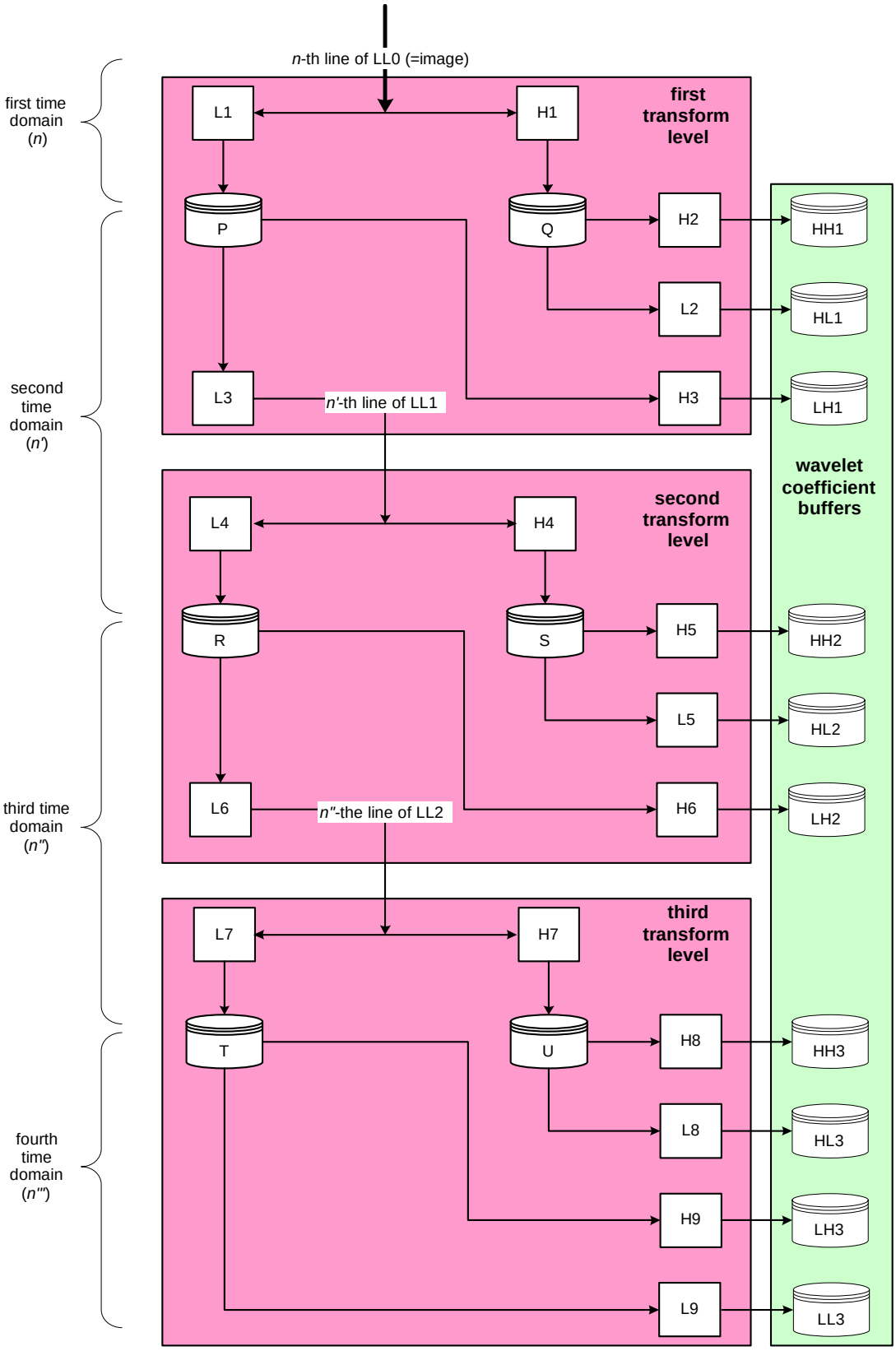


Figure 4-4: DWT Data Flow

The functional blocks L1 and H1 perform horizontal low- and high-pass filtering of the image lines. The corresponding output data are stored in the respective buffers P=FIFO_L(1) and Q=FIFO_H(1). Let p_{rc} and q_{rc} denote the samples at row r, column c of buffers P and Q, respectively, for $r = 0,1,...,8$, $c = 0,1,...,\frac{w}{2}-1$.

L3 and H3 perform vertical low- and high-pass 1-d filtering of the data stored in P. The data line $\{s_0,...,s_{w/2-1}\}$ which is produced by L3 for every new count of n' is given by

$$s_c = \sum_{-4}^{4} h_n p_{n+4,c} \tag{4-1}$$

and the data line $\{t_0,...,t_{w/2-1}\}$ produced by H3 for every new count of n' is given by

$$t_c = \sum_{-3}^{3} g_n p_{n+3,c} \tag{4-2}$$

In these equations, h_n and g_n are the wavelet coefficients specified in subsection 3.3 of reference [1].

L2 and H2 perform vertical low- and high-pass 1-d filtering of the data stored in Q. The data line $\{u_0,...,u_{w/2-1}\}$ produced by L2 is given by

$$u_c = \sum_{-4}^{4} h_n q_{n+4,c} \tag{4-3}$$

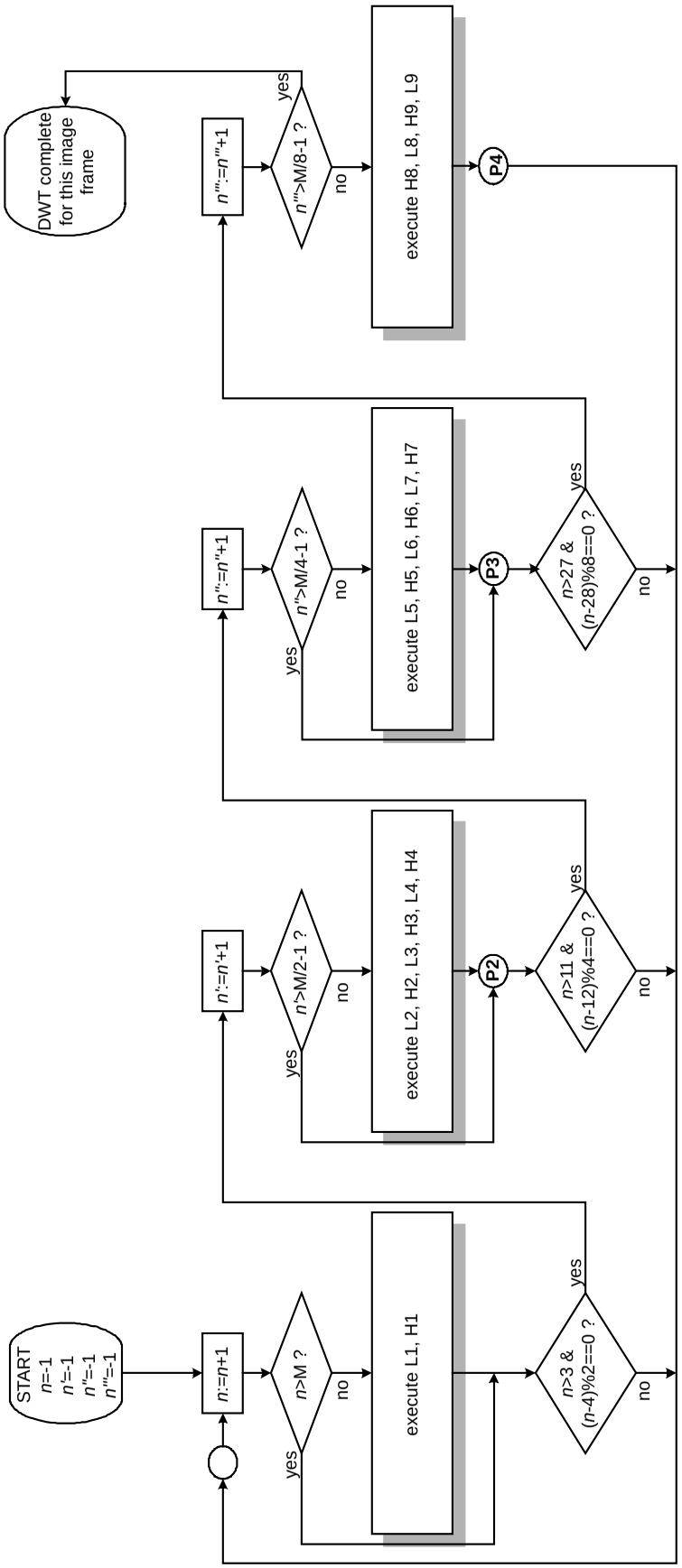
and the data line $\{v_0,...,v_{w/2-1}\}$ produced by H2 is given by

$$v_c = \sum_{-3}^{3} g_n q_{n+3,c} \tag{4-4}$$

Equations (4-1) through (4-4) apply to the Float DWT; the Integer DWT is implemented similarly. The filter definitions change at the beginning and ending of image frames; the details are omitted.

Equivalent definitions apply for the second and third transform levels.

Figure 4-5 exhibits the program flow of the proposed implementation, explicitly exhibiting the role of time/clocking of the proposed implementation. Each of the four columns in figure 4-5 corresponds to a different clock domain. Moving from left to right in the figure, each domain is clocked at half the rate of the preceding domain.



NOTE – The notation a%b denotes a mod b. The & symbol denotes the logical AND operation.

Figure 4-5: DWT Program Flow

The following formulas define the value of the first (master) clock n when the counters of the other clock domains are updated to assume their new values n', n'', n''':

$$\begin{aligned} P2&: \; n = 4 + 2n' \\ P3&: \; n = 12 + 4n'' \\ P4&: \; n = 28 + 8n''' \end{aligned} \tag{4-5}$$

The formulas (4-5) are true in the precise respective instances when the DWT program flow crosses points P2, P3, and P4 shown in figure 4-5.

Conversely, if the master clock domain displays the counter value n *as* the program flow crosses point P1, the counter values of the other clock domains are given by equation (4-6):

$$P1: \begin{cases} n' = \left\lfloor \dfrac{n-4}{2} \right\rfloor \\ n'' = \left\lfloor \dfrac{n-12}{4} \right\rfloor \\ n''' = \left\lfloor \dfrac{n-28}{8} \right\rfloor \end{cases} \tag{4-6}$$

4.3.4 BUFFER SIZE REQUIREMENTS

For an image width of w pixels, the FIFO memory requirements are given by

$$9w + 9\frac{w}{2} + 9\frac{w}{4} = 63\frac{w}{4} \text{ samples}$$

Aside from some negligible local memory required by the filter functions, the only other memory required is the Segment Block Buffer which has 'baseline' size $64 \times S$ samples since it must store S blocks. The baseline size is not the maximum size because there is some delay between the initial storage of subband data and the arrival of the associated DC coefficients, required to complete the 8×8 coefficient blocks. This delay is due to the pipeline structure of the implementation.

For simplicity in computing Block Segment Buffer size requirements, it is assumed in a first step that a segment corresponds to one strip, i.e. $S = w/8$. At any time when the program flow has reached point P4, the time counters n,n',n'',n''' have certain values, and a corresponding number of subband data lines has been produced since the start of DWT processing, and has been stored in the Block Segment Buffer. On the other hand, whenever point P4 is reached, a new segment's worth of DWT coefficients becomes available for BPE-coding. It is assumed that this and the segment before need to be stored simultaneously: the current one because it has just been in the process of being filled, and the one before because it has just been in the process of being encoded. All segments produced earlier have already been discarded and therefore do not contribute to Block Segment Buffer memory requirements.

Consider any pass through P4 where the DWT processing pipeline is filled, i.e., where n''' is large enough. For example, suppose $n''' = 10$. Since P4 and P1 coincide, the corresponding value of n is

$$n = 28 + 8 \cdot n''' = 108$$

The next time the program flow will reach P2, n will have increased by two and according to equation (4-6) n' will assume the value $\left\lfloor \frac{110-4}{2} \right\rfloor = 52$. This means the current value of n' is $n' = 51$. By a similar argument, the future value of n'' will be $\left\lfloor \frac{112-12}{4} \right\rfloor = 25$, and the current value of n'' is 24. Thus 52 rows of HH_1, LH_1, HL_1 data have been generated as well as 25 rows of HH_2, LH_2, HL_2 data and 11 rows of HH_3, LH_3, HL_3, LL_3 data, yielding a filling state of the Segment Block Buffer of

$$3 \cdot 52 \cdot \frac{w}{2} + 3 \cdot 25 \cdot \frac{w}{4} + 4 \cdot 11 \cdot \frac{w}{8} - 9 \cdot 64 \cdot \frac{w}{8} = 242 \cdot \frac{w}{8} \text{ samples}$$

The subtracted quantity is the result of nine instances of earlier data deletion for $n'''=0,\ldots,8$, after segments had been completely encoded (i.e., sent to BPE). Segment data for $n'''=9$ is now (i.e., that $n'''=10$) due for deletion but has been kept available as current input for the BPE. The total memory requirements (from all FIFOs and the Segment Block Buffer) are therefore given by

$$(126+242) \times w/8 = 46 \times w \text{ samples.}$$

For example, if three bytes are used to store each DWT coefficient for an image of width w=1024, then the memory requirement (for strip compression) is

$$46 \times 1024 \times 3 = 138 \text{ kilobytes.}$$

For comparison, if a full-frame image consists of 1024 rows, then storage of a full frame of DWT coefficients would require three megabytes.

More generally, let a segment consist of $z \geq 1$ strips. While the FIFO memory requirements remain as before ($63\frac{w}{4}$ samples), the Block Segment Buffer memory requirements are to be upscaled by the factor z; i.e., the memory requirement is $242 \cdot \frac{z \cdot w}{8}$ samples. In summary, for w pixels per row and z strips per segment, the total memory requirement for a three-byte-per-sample implementation is given by

$$w \cdot \left(\frac{189}{4} + z \cdot \frac{363}{4} \right) \text{ bytes}$$

4.4 DWT RECONSTRUCTION

4.4.1 GENERAL

The image information encoded in a compressed segment consists of bit planes of DWT coefficient data. After decoding this information, the job of the decompressor is to select a reconstructed value for each DWT coefficient and then perform the two-dimensional inverse wavelet transform to produce a reconstructed image. This subsection focuses on the problem of selecting a reconstructed value for each DWT coefficient given the information about that coefficient available to the decompressor.

Lossy compression effectiveness depends on the decompressor's scheme for selecting reconstructed DWT coefficient values. The Recommendation does not mandate any particular DWT coefficient reconstruction strategy, and so users are free to select a strategy that provides good results for their application. This subsection describes a simple and reasonably effective 'baseline' reconstruction scheme that was used to produce all compression results presented in this Report. The baseline scheme can serve as a reference for comparison with other approaches that might provide improved compression effectiveness in some applications.

Note that when the float DWT is used, DWT coefficients are real-valued and then rounded to the nearest integer before bit-plane encoding. Consequently, even when all bit-plane information is available to the decoder, improved reconstructed image quality might be possible by careful selection of DWT coefficient reconstructed values.

4.4.2 DC COEFFICIENTS

DC coefficients are represented in two's complement representation. After decoding the compressed segment data, for a given DC coefficient some number, b, of the least significant bits have unknown values while the values of the remaining more significant bits are known. Note that this b value for DC is determined by comparing the DC quantization factor, q, in subsection 4.3.2 of reference [1], to the stopping point in either subsection 4.3.3 or subsection 4.5 of reference [1]. The smaller value from the comparison is assigned to b.

Let c denote the true value of the DC coefficient. Note that c is real-valued when the float DWT is used and integer-valued when the integer DWT is used. Let $\tilde{c}$ denote the value obtained for the DC coefficient if the b unknown bits are set to 'zero'.

When the float DWT is used, the range of possible values for c is the interval

$$\left\lfloor \tilde{c} - \frac{1}{2}, \quad \tilde{c} + 2^b - \frac{1}{2} \right\rfloor$$

and under the baseline strategy the DC coefficient's reconstructed value, $\hat{c}$, is

$$\hat{c} = \tilde{c} + 2^{b-1} - \frac{1}{2}$$

which corresponds to the midpoint of the interval of possible values for c.

When the integer DWT is used, the possible values for c are the integers

$$\{\tilde{c}, \quad \tilde{c}+1, \quad \ldots, \quad \tilde{c}+2^{b}-1\}.$$

When $b > 0$ this is an even number of consecutive integers, and thus the average of these values is not an integer. For the baseline strategy the reconstructed value $\hat{c}$ is chosen to be the smaller of the two integers closest to the mean:

$$\hat{c} = \begin{cases} \tilde{c} + 2^{b-1}, & \text{if } b > 0 \\ \tilde{c}, & \text{if } b = 0 \end{cases}.$$

Example

Suppose BitDepthDC=10 for a segment, so each DC coefficient is represented as a 10-bit binary number using two's complement representation. For some DC coefficient, suppose the first four bits have been encoded in the compressed bitstream, i.e., $b = 6$, and the two's complement binary representation the DC coefficient is

1011XXXXXX

where each 'X' represents an unknown bit value. If the six unknown bit values are zero, the coefficient would have value -320; i.e., $\tilde{c} = -320$ in this example.

If the integer DWT was used, the possible values for the DC coefficient are {-320, -319, …, -257}. The reconstructed value for the DC coefficient is $\hat{c} = \tilde{c} + 2^{b-1} = -320 + 2^5 = -288$.

If the float DWT was used, the range of possible values for the DC coefficient is [-320.5, -256.5] and the reconstructed value for the DC coefficient is the midpoint of this range:

$$\hat{c} = \tilde{c} + 2^{b-1} - \frac{1}{2} = -320 + 2^5 - \frac{1}{2} = -288.5.$$

4.4.3 AC COEFFICIENTS

AC coefficients are represented using a sign bit along with several magnitude bits. After decoding the compressed segment data, for a given AC coefficient some number, b, of the least significant magnitude bits have unknown values, while the values of the remaining (more significant magnitude) bits are known.

Let a denote the true value of the AC coefficient. Note that a is real-valued when the float DWT is used and integer-valued when the integer DWT is used. Let $\tilde{a}$ denote the value obtained for the AC coefficient magnitude if the b unknown bits are set to 'zero'.

There are two possible situations. Either the sign of a is unknown and all of the known magnitude bits are zero, or the sign of a is known and not all of the magnitude bits are zero.

If the sign of the AC coefficient is unknown, then the set of possible values for a is symmetric about zero, and the selection for the baseline strategy is

$$\hat{a} = 0$$

as the reconstructed value under both the integer and float DWT.

If the sign of a is known and the float DWT is used, then the range of possible values for the magnitude $|a|$ is

$$\left[\tilde{a} - \frac{1}{2}, \quad \tilde{a} + 2^b - \frac{1}{2}\right]$$

and the midpoint of this interval can be used as the reconstructed value for the AC coefficient magnitude. I.e., under the baseline approach the AC coefficient reconstructed value, $\hat{a}$, is

$$\hat{a} = \left(\tilde{a} + 2^{b-1} - \frac{1}{2}\right) \cdot \text{sign}(a).$$

If the sign of a is known and the integer DWT is used, to reconstruct the AC coefficient, the subband weight applied to the AC coefficient must be taken into account, as described in subsection 3.9 of the Recommendation. Each subband weight is a power of two, and thus at the reconstruction stage, inverting the effect of the subband weight amounts to performing an appropriate number of right bit-shift operations. Let b^* denote the number of unknown bits in the coefficient after taking into account the subband weights and let $\tilde{a}^*$ denote the value obtained for the coefficient magnitude if the unknown bits are set to 'zero' and the weight factor is taken into account. Then the set of possible values for the magnitude $|a|$ is

$$\{\tilde{a}^*, \quad \tilde{a}^* + 1, \quad \ldots, \quad \tilde{a}^* + 2^{b^*} - 1\}.$$

This is an even number of consecutive integers, and thus when $b^* > 0$ the average of these values is not an integer, and so for the baseline strategy the smaller magnitude of the two integers closest to the mean value to determine the reconstructed value is selected:

$$\hat{a} = \begin{cases} \left(\tilde{a}^* + 2^{b^*-1}\right) \cdot \text{sign}(a), & \text{if } b^* > 0 \\ \tilde{a}^* \cdot \text{sign}(a), & \text{if } b^* = 0 \end{cases}.$$

Example

Suppose BitDepthAC=10 for a segment, so each AC coefficient is represented using a single sign bit along with 10 magnitude bits. For some AC coefficient, suppose that the sign bit indicates that the coefficient is negative and the first four magnitude bits have been decoded; i.e., $b = 6$, and the binary representation the AC coefficient magnitude is

1011XXXXXX

where each 'X' represents an unknown bit value.

If the float DWT was used and the six unknown bit values in the coefficient are zero, the coefficient magnitude would have value 704; i.e., $\tilde{a} = 704$ in this example. The range of possible values for the AC coefficient magnitude is [703.5, 767.5] and the reconstructed value for the AC coefficient is

$$\hat{a} = \left(\tilde{a} + 2^{b-1} - \frac{1}{2}\right) \cdot \text{sign}(a) = \left(704 + 2^5 - \frac{1}{2}\right) \cdot (-1) = -735.5$$

If the integer DWT was used, then the possible values of the AC coefficient depend on the subband to which the coefficient belonged because of the subband weights applied. Suppose that the AC coefficient of interest is from the HH_3 subband. In this case, the AC coefficient was scaled by a factor of 2^2 (see table 3-3 of the Recommendation) and consequently the two least significant bits must be zero, thus $b^* = 4$. The binary representation of the coefficient magnitude can be thought of as

1011XXXX

where each 'X' represents an unknown bit value. If these four unknown bits are set to zero, the coefficient would have magnitude 176; i.e., $\tilde{a}^* = 176$. Under the baseline strategy, the reconstructed value for this coefficient is $\hat{a} = \left(\tilde{a}^* + 2^{b^*-1}\right) \cdot \text{sign}(a) = \left(176 + 2^3\right) \cdot (-1) = -184$.

4.5 PIXEL READOUT CORRECTION

Two-dimensional images are typically produced from multiple sensing elements arranged in a one-dimensional or two-dimensional array. These sensing elements seldom have identical responses to a given stimulus, even when they are manufactured under precisely controlled processes. The result is a slight variation in signal response for each pixel given a completely flat input stimulus field. In addition, many imaging sensors have defects such as 'hot pixels' or 'dead pixels', corresponding to sensing elements that always give near-saturation or very low response, respectively.

These defects effectively act as noise in the source image, and thus adversely affect compression effectiveness. Compression effectiveness is improved when pixel readout correction can be performed onboard, prior to compression. Pixel readout correction algorithms vary depending on the sensor response, noise level, and applications with the aim to produce a uniform image when pixels receive the same light input (reference [11]).

5 PERFORMANCE RESULTS

5.1 GENERAL

This section provides sample compression results using the CCSDS reference image set and comparisons with other compression algorithms. Lossless compression results are provided in 5.2 for the CCSDS compressor along with performance results from the JPEG2000, JPEG-LS, SPIHT and CCSDS/Rice compressors. Lossy compression results are provided in 5.3 for the CCSDS compressor as well as for the JPEG2000 and SPIHT compressors.

Because the CCSDS image compression Recommendation is intended for high-speed space-born application, test results relevant for limited-memory applications are presented. In particular, for the present Recommendation, strip-based compression (see 4.3) is of primary interest. For comparison purposes, JPEG2000 compression performance under similar constraints is described in further detail below. Therefore, the JPEG2000 algorithm is exercised in both default 'frame-based' compression, as well as in the 'scan-based' (also called strip-based) compression. The SPIHT compressor only allows 'frame-based' compression. It is good to keep in mind that frame-based compression increases implementation memory requirements.

Limitations and simplifications applied to the JPEG2000 encoder (e.g., limitation in tile size, simplification of rate-distortion optimization procedure) imposed by hardware implementation constraints may result in significantly lower performance than results presented in this section, since the latter were obtained without such constraints.

5.2 LOSSLESS COMPRESSION

5.2.1 GENERAL

This subsection provides lossless compression results for the test images.

5.2.2 ALGORITHMS

5.2.2.1 CCSDS

Lossless compression results for the Recommendation are obtained using the integer DWT, with OptACSelect and OptDCSelect set to 1 while parameter *S* is adjusted to achieve both strip- and frame-based compression. All optional header parts are included in the first coded segment while no optional headers are included for subsequent coded segments.

5.2.2.2 JPEG2000

JPEG2000 results were produced using Verification Model (VM) 9.0,[7] which integrates the scan-based mode. Other well-known JPEG2000 compressors (references [12] and [13]) do not provide this mode because it is an optional mode defined in Part 2 of the JPEG2000 recommendation. Scan-based compression is of primary interest, because it imposes memory constraints comparable to strip compression in the CCSDS Recommendation. To simulate memory-limited implementation in JPEG2000, the DWT and the global rate allocation procedure are performed on a portion of the image with limited height. For completeness, frame-based JPEG2000 compression is also evaluated.

The following options are used to obtain lossless JPEG2000 compression results:[7]

- 'frame-based' and 'scan-based' compression options:
  - DWT: integer 5/3 filter (see reference [14]),
  - Number of resolution level: 3,
  - Code-block dimension in entropy coding module: 8 x 512,
  - Precision of the quantization step: set at step = 1/(2input_dynamic - 1);
- scan-based mode options:
  - Packet Progression Order in the codestream (*-Corder*) : PC (Position - Component),
  - Packet/Precinct Partition dimension within each resolution level (*-Cpp*): 32x2048, 16x2048, 8x2048 (32 lines maximum),
  - Scan buffer elements (*-Cscan_buffer_elements*): value of 1 is set for this parameter.

5.2.2.3 JPEG-LS

The software used is the JPEG-LS Reference Encoder - V.1.00. This software module was originally developed by Hewlett-Packard Company in the course of development of the ITU-T Rec. T.87|ISO/IEC 14495-1 standard. The names of the executables in the package derive from the acronym 'LOCO' (LOw COmplexity LOssless COmpression), as the core of the standard is based on the LOCO-I algorithm (LOCO for Images) developed at Hewlett-Packard Laboratories (reference [15]). The default parameter values are used to obtain performance results.

[7] ISO/IEC JTC1/SC29/WG1 N2021, Information JPEG2000 Verification Model 9.0 (available only to committee members).

JPEG-LS performs predictive-based lossless image compression. Pixels are compressed in raster-scan order using a single pass. Consequently, it is easily implementable as a scan-based compressor without any loss of compression effectiveness. JPEG-LS has very low complexity and provides highly effective compression, but can only be used to provide lossless or near-lossless compression.

5.2.2.4 CCSDS/Rice

The CCSDS/Rice compressor refers to the CCSDS recommendation in reference [4]. In performance simulation, the block length *J* is set at 16. The reference interval is set equal to the image width. Compression is performed using the simple differential predictive coding included in the lossless recommendation; i.e., two-dimensional correlations are not exploited in the simulation. Unlike the other compression approaches considered in this section, the CCSDS/Rice compressor is not specifically tailored to imagery.

5.2.2.5 SPIHT

The Set Partitioning In Hierarchical Trees (SPIHT) algorithm is a low-complexity progressive image compressor described in reference [16]. This enhanced implementation of a zerotree algorithm efficiently encodes zerotrees with a relatively modest level of complexity and produces an embedded bitstream. The algorithm uses bit-plane encoding of DWT coefficients to produce an embedded bitstream. For the results presented in table 5-1, the optional arithmetic coding is used.

The lossless mode in SPIHT is achieved by using the S+P multiresolution representation presented in reference [17].

SPIHT performances results are produced using the *progcode* and *progdecd* programs in the C++ software version 8.01, 8/16/96. After a certain point (corresponding to visually indistinguishable difference between original and recovered image) the coding process changes to a second compression technique which is less effective but faster and more memory-efficient. The software was not able to reach lossless compression for three of the test images, and this problem is indicated with dashes in table 5-1.

5.2.2.6 ICER

ICER is another progressive wavelet-based image compressor capable of providing lossless and lossy compression (reference [18]). ICER offers wavelet-domain image segmentation for error-containment purposes. ICER offers slightly improved lossless and lossy compression effectiveness compared to the CCSDS image compression Recommendation, but ICER has slightly higher complexity and has only been implemented as a frame-based compressor. A software implementation of ICER is used by the Mars Exploration Rover (MER) missions for onboard lossy image compression.

ICER lossless compression results given in table 5-1 are produced using four stages of wavelet decomposition, a 2/6 integer DWT, and a single error-containment segment.

5.2.3 RESULTS

Table 5-1 shows the compressed bit rate in bits/pixel achieved by each of the lossless compressors on the test set.

Table 5-1: Lossless Compression Performance

Bit depth	Image	Compressed bit rate (bits/pixel)							
		Strip-based compression				Frame-based compression			
		CCSDS	CCSDS/ Rice	JPEG-LS	JPEG 2000	CCSDS	JPEG 2000	SPIHT	ICER
8	coastal_b1	3.36	3.56	3.09	3.18	3.36	3.13	3.09	3.07
	coastal_b2	3.22	3.32	2.90	3.03	3.22	2.97	2.94	2.92
	coastal_b3	3.48	3.68	3.22	3.30	3.48	3.23	3.21	3.20
	coastal_b4	2.81	2.91	2.41	2.59	2.81	2.53	2.57	2.55
	coastal_b5	3.16	3.30	2.81	3.01	3.17	2.94	2.91	2.89
	coastal_b6h	3.02	2.75	2.50	2.68	3.02	2.60	2.71	2.54
	coastal_b6l	2.35	2.03	1.76	2.03	2.35	1.96	2.02	1.87
	coastal_b7	3.45	3.66	3.17	3.28	3.45	3.22	3.17	3.15
	coastal_b8	3.66	3.93	3.42	3.42	3.67	3.40	3.35	3.31
	europa3	6.61	7.48	6.64	6.56	6.60	6.52	6.46	6.30
	marstest	4.78	5.39	4.69	4.79	4.77	4.74	4.64	4.63
	lunar	4.58	5.23	4.35	4.56	4.58	4.49	4.43	4.40
	spot-la_b3	4.80	5.20	4.53	4.74	4.79	4.69	4.70	4.56
	spot-la_panchr	4.27	4.87	4.00	4.16	4.26	4.13	4.11	4.03
	average of 8-bit images	**3.82**	**4.09**	**3.54**	**3.67**	**3.82**	**3.61**	**3.59**	**3.53**
10	ice_2kb1	4.78	5.44	4.74	4.77	4.78	4.73	4.61	4.64
	ice_2kb4	3.37	3.86	3.23	3.28	3.37	3.25	3.17	3.18
	india_2kb1	4.77	5.25	4.63	4.76	4.77	4.72	4.63	4.63
	india_2kb4	4.06	4.70	3.97	4.05	4.07	4.01	3.93	3.94
	landesV_G7_10b	5.04	6.30	4.42	4.64	5.04	4.56	4.88	4.42
	marseille_G6_10b	6.74	7.56	6.57	6.77	6.72	6.72	6.60	6.49
	ocean_2kb1	4.94	5.32	4.61	4.91	4.94	4.88	4.79	4.75
	ocean_2kb4	3.81	4.41	3.60	3.76	3.81	3.73	3.67	3.64
	average of 10-bit images	**4.69**	**5.36**	**4.47**	**4.62**	**4.69**	**4.57**	**4.54**	**4.46**
12	foc	3.43	3.35	3.28	3.21	3.45	3.20	3.12	3.07
	pleiades_portdebouc_b3	7.87	8.63	7.79	8.01	7.87	7.97	7.71	7.73
	pleiades_portdebouc_pan	7.18	7.92	7.10	7.31	7.18	7.28	--	7.01
	solar	6.21	7.12	6.05	6.06	6.21	6.01	5.96	5.88
	sun_spot	5.79	6.63	5.67	5.71	5.78	5.66	5.66	5.49
	wfpc	3.82	4.04	3.61	3.49	3.84	3.47	3.43	3.32
	average of 12-bit images	**5.72**	**6.28**	**5.58**	**5.63**	**5.72**	**5.60**	**--**	**5.42**
16	P160_B_F	12.23	12.62	12.22	12.50	12.22	12.47	--	12.03
	sar16bit	9.92	10.31	9.90	10.07	9.92	10.02	--	9.68
	average of 16-bit images	**11.07**	**11.47**	**11.06**	**11.29**	**11.07**	**11.25**		**10.86**
--: value unavailable because of software problem during execution									

Of the algorithms considered, ICER delivers the best average lossless compression performance on the test images. However, JPEG-LS is about equally effective on the 8-bit and 10-bit images and has significantly lower complexity than the wavelet-based compressors. However, JPEG-LS provides only lossless and near-lossless compression, unlike the wavelet-based image compressors. Comparing the average compressed bit rate over all images at a given bit depth, the present Recommendation has lower performance than frame-based SPIHT, ICER, and JPEG2000 compressors on 8-bit, 10-bit and 12-bit test images. For 16-bit test data, the Recommendation performs better than JPEG2000. In both strip-based/scan-based and the frame-based options, performances of the JPEG2000 and the Recommendation are very close. The Recommendation algorithm provides similar performances in both strip-based compression and frame-based compression. The CCSDS/Rice compressor exhibits the lowest performance, which is to be expected since only one-dimensional correlation was explored in the test.

5.3 LOSSY COMPRESSION

5.3.1 OVERVIEW

To evaluate lossy compression performance, the PSNR and MAE metrics are calculated (see 2.2) at several different compressed bit rates for each of the test images.

5.3.2 ALGORITHMS

5.3.2.1 General

Three algorithms are used in the evaluation: the Recommendation (using a software implementation developed at NASA GSFC), the JPEG2000 standard (VM v9.0) in frame-based and scan-based modes and the SPIHT algorithm (reference [16]).

5.3.2.2 CCSDS Recommendation

The parameters used for lossy compression are the following:

- Float DWT;
- SegByteLimit is adjusted to achieve rate-controlled compression (see 3.4) at the desired bit rate;
- *S* is adjusted to achieve strip compression and also full-frame compression (see 3.5);
- all optional header parts are included in the first coded segment but no optional headers are included for subsequent coded segments;
- CodeWordLength is set to 0, corresponding to single-byte output words.

5.3.2.3 JPEG2000

The JPEG2000 implementation was described in 5.2.2.2. Both scan-based and frame-based modes are computed with the options specified in this previous part. Only the choice of DWT differs; for lossy compression the 9/7 Float DWT is used. Moreover, in lossy domain, to reach the targeted output rate, it is necessary to use a rate-allocation optimization: an optimal rate allocation procedure using Lagrangian multipliers is used (*-Flra* option under the VM9.0 software).

5.3.2.4 SPIHT

The SPIHT algorithm and software was discussed in 5.2.2.5. Lossy compression makes use of the 9/7 Float DWT. Arithmetic coding is performed.

The number of levels of DWT decomposition is not controlled by the user, but rather is set automatically by the software depending on the dimensions of the image. The number of decomposition levels for each test image is summarized in table 5-2.

Table 5-2: Number of Decomposition Levels Used by the SPIHT Software on Test Images

images	number of decomposition levels
landesV_G7_10b(padded to 464x2384), marseille_G6_10b, spot-la_panchr (padded to 1008x1008), europa3 (padded 560x608)	4
wfpc, pleiades_portdebouc_b3	5
b6l, b6h (coastal), marstest, munar, spot-la_b3 (padded 512x512) , sar_16bit, sun_spot, foc	6
coastal_b1, coastal_b2, coastal_b3, coastal_b4, coastal_b5, coastal_b7, solar, pleiades_portdebouc_pan(padded to 1408x5504)	7
coastal_b8, ice_2kb1, ice_2kb4, india_2kb1, india_2kb4, ocean_2kb1, ocean_2kb4, P_160_B_F	8

5.3.3 RESULTS

To assess lossy compression effectiveness, PSNR and MAE are measured on reconstructed images produced using each of the algorithms considered at bit rates of 0.25, 0.5, 1.0, and 2.0 bits/pixel for every image in the test set.

Complete results from these evaluations are tabulated in annex C. In this subsection these results are summarized by providing average PSNR as a function of bit rate over all images of a given dynamic range. These results are plotted in figures 5-1 to 5-4.

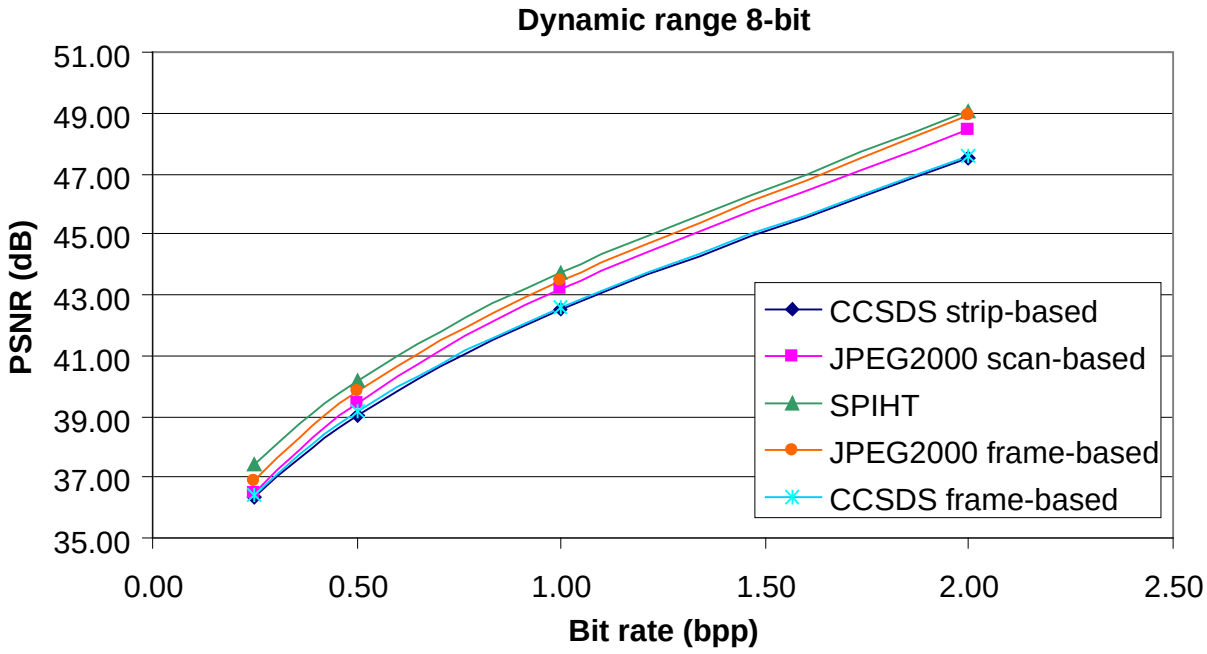


Figure 5-1: Mean PSNR Curve for Each Algorithm on 8-Bit Test Images

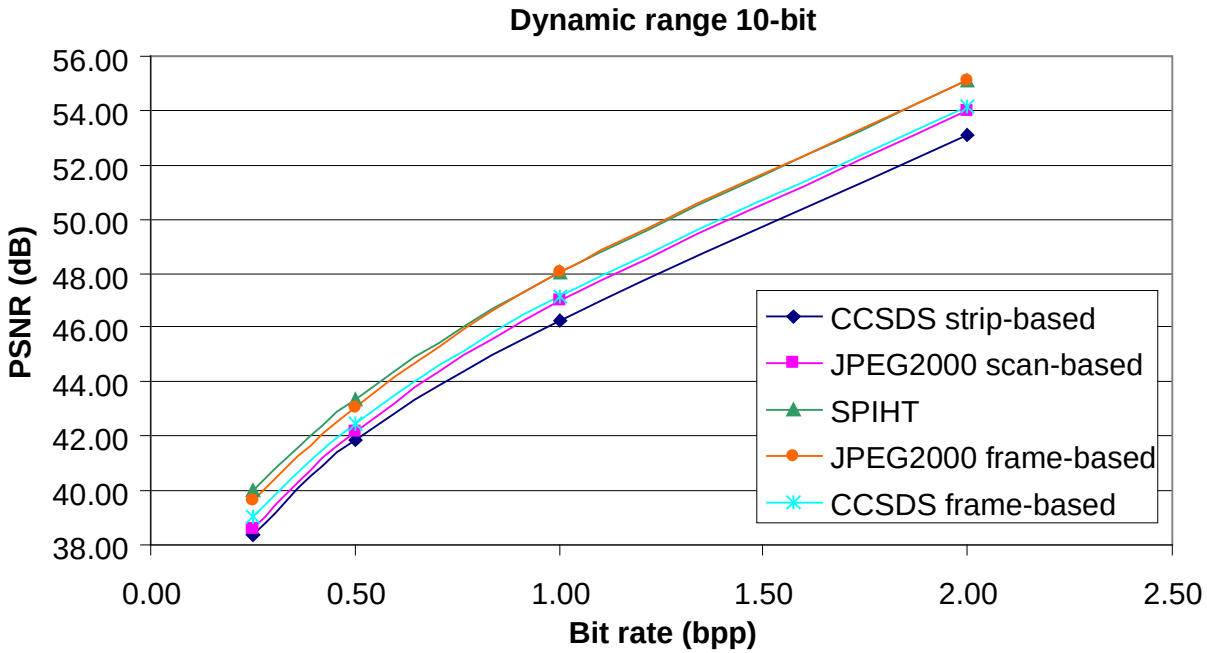


Figure 5-2: Mean PSNR Values for Each Algorithm on 10-Bit Test Images

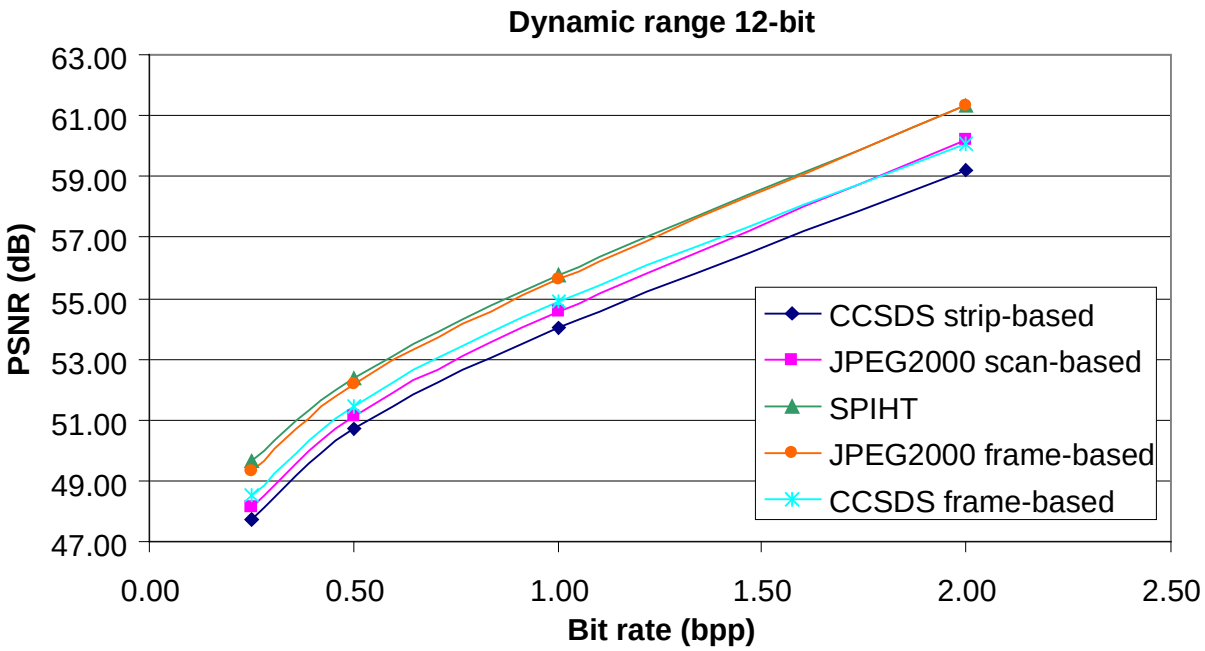


Figure 5-3: Mean PSNR Values for Each Algorithm on the 12-Bit Test Images

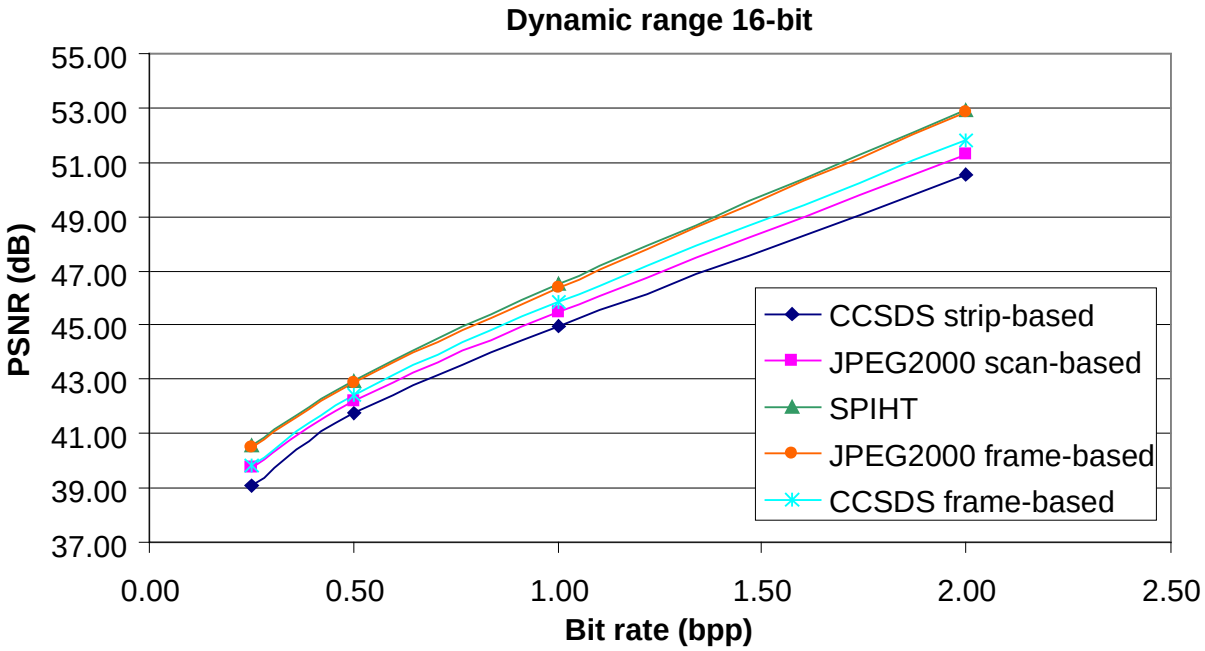


Figure 5-4: Mean PSNR Values for Each Algorithm on the 16-Bit Test Images

5.3.4 CONCLUSIONS

CCSDS strip-based compression offers compression effectiveness close to that obtained by JPEG2000 scan-based compression for all images both for PSNR and MAE metrics, with the JPEG2000 compressor providing somewhat better performance. The SPIHT and frame-based JPEG2000 compressors offer noticeably more effective compression, but at the expense of increased memory requirements. Preliminary visual comparison between the strip-based CCSDS and the scan-based JPEG2000 yielded no systematic conclusion.

For lossy compression, the CCSDS algorithm's frame-based results are better than its strip-based results by roughly one dB except for the eight-bit data, on which the results are very similar. Overall, the trade-off between implementation complexity and performance is observed for either strip-based or frame-based results: the CCSDS algorithm is slightly lower than JPEG2000. However, this trade-off in performance is within one dB, and this difference may be smaller than the performance penalty obtained when similar complexity constraints are imposed on JPEG2000 in a practical hardware implementation.

6 ALGORITHM SELECTION

6.1 OVERVIEW

This section describes some of the decisions made in defining the Recommendation along with some of the motivation behind those decisions. Subsection 6.2 provides an overview of the algorithm definition process. Subsection 6.3 describes differences between the Recommendation and the JPEG2000 standard. Subsection 6.4 documents the motivation and analysis that led to the selection of the particular DWTs and default subband weight factors included in the Recommendation.

6.2 SELECTION PROCESS

In 1998, the CCSDS Data Compression Working Group began an effort to establish an image compression recommendation suitable for space-borne applications. The working group agreed that a suitable compressor must meet the requirements listed in table 6-1; these requirements reflect the envisioned application of real-time hardware compression onboard a spacecraft.

Table 6-1: Image Compression Requirements

1	Process both frame and non-frame (push-broom) data
2	Offer adjustable coded data rate or image quality (up to lossless)
3	Accommodate from 4-bit to 16-bit input pixels
4	Provide real-time processing with space qualified electronics (≥ 20 Msamples/sec, ≤ 1 watt/Msamples/sec, based on year 2000 space electronics technology)
5	Require minimal ground operation
6	Limit the effects of a packet loss due to bit errors in transmission channel to a small region of the image

Apart from the requirements listed in table 6-1, perhaps the biggest consideration in the algorithm selection process was compression effectiveness. The ability to perform progressive compression was viewed as highly desirable but not mandatory. It was the hope of the working group that if any patents were included in the Recommendation, a royalty-free license could be offered to all CCSDS Member Agencies.

The working group also assembled a diverse set of 30 test images covering a variety of space-borne imaging applications, including solar, stellar, planetary, Earth observations, optical, radar, and meteorological applications. The image test set is described in annex B.

Algorithms considered as candidates for a CCSDS image compression recommendation included JPEG2000 as well as algorithms proposed by ESA, NASA, and CNES. Candidate algorithms were proposed and performance evaluations were conducted based on both quantitative evaluations of compression effectiveness as well as subjective assessments of image quality. In addition, implementation architecture studies were performed to assess the real-time processing capabilities of the proposed algorithms. Implementation complexity played a significant role in the final algorithm selection. For spacecraft applications, this could have a significant impact on the achievable processing rate.

A consensus was reached in 2003 when a wavelet-based algorithm followed by a limited complexity tree-based BPE was selected. The resulting image compression Recommendation combines elements from different algorithms that were initially proposed, along with modifications to reduce complexity.

6.3 COMPARISON WITH JPEG2000

6.3.1 GENERAL

Because JPEG2000 is a reference standard, this subsection describes some of the major differences and similarities between JPEG2000 and the CCSDS Recommendation as well as the motivation for adopting a different lower-complexity compression algorithm.

Both algorithms rely on progressive bit-plane encoding of wavelet-transformed image data, and both provide a choice of integer and a floating-point DWTs so that effective lossy and lossless compression can be achieved. JPEG2000 offers better compression effectiveness, but has significantly higher implementation complexity.

6.3.2 DWT

The JPEG2000-recommended floating-point DWT is the same as the one included in the CCSDS Recommendation, but the recommended integer DWT is a 5/3 DWT for JPEG2000 and a 9/7 DWT for CCSDS. Subsection 6.4 describes in detail the considerations for selection of the DWTs included in the CCSDS Recommendation. In tests, the 9/7 integer DWT offered better rate-distortion performance than the 5/3 integer DWT when other parts of the compression algorithm are fixed. For lossless performance, the 9/7 integer also performs slightly better on the test set as shown in 6.4.

The motivation for including a choice of integer or floating-point DWT, described in 3.3, applies equally to the CCSDS and JPEG2000 compressors.

The number of levels of wavelet decomposition may be as large as five in JPEG2000, but is fixed at three in the CCSDS Recommendation to maintain low complexity.

6.3.3 BIT-PLANE ENCODING

For bit-plane encoding of DWT coefficient information, JPEG2000 uses a context-based embedded block coder that adapts to local subband statistics. The subband coefficients are quantized and collected into rectangular arrays called code-blocks. Each code-block is independently entropy coded using an effective bit-plane coder. For each of three passes, a context and a bit-stream are generated and provided to the adaptive arithmetic coder (see figure 6-1). This approach provides several benefits, including random image data access, better rate control, and flexibility in arranging progression order. Moreover, the processing of code-blocks is readily amenable to parallelization because the coding procedure is completely the same and independent from one code-block to another, a separate bit-stream is generated, and no information from other code-blocks is necessary.

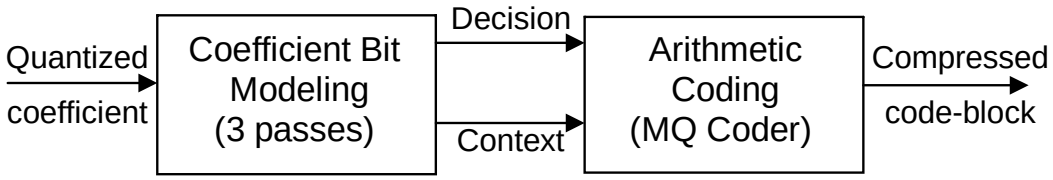


Figure 6-1: JPEG2000 Bit-Plane Encoding Scheme for a Code-Block with Quantized Coefficients

JPEG2000 includes features that are not available under the CCSDS Recommendation. However, in addition to added complexity, these features also come at the price of higher overhead (in terms of header information) that may not be negligible for small images. For example, for a single-component image and one tile, the required marker size is 92 bytes (reference [5]). By contrast, under the CCSDS Recommendation, headers may be as small as three bytes per segment when optional headers are omitted.

The JPEG2000 standard includes error resilient bit-stream syntax and tools to improve performance of transmitting compressed images over noisy channels. The proposed tools include data partitioning and resynchronization, error detection and concealment, and Quality of Service transmission based on priority. Error resilience is achieved at the entropy coding level by re-initializing context models at the beginning of each code-block, termination of arithmetic coder for each pass, reset of contexts after each coding pass, selective arithmetic coding bypass with the lazy mode and the use of segmentation symbols. In addition, protection is implemented at the packet level by using short packet format or packet with resynchronization marker placed in front of every packet.

The CCSDS image compression Recommendation provides error containment at the segment level as described in 2.1.

In JPEG2000, for each coding-pass and each code-block the entropy coder calculates the resulting reduction in MSE distortion. Once the entire image has been compressed, a post-processing operation passes over all the compressed code-blocks and determines the extent to

which each code-block's embedded bit-stream should be truncated to achieve a target bit-rate or quality metric.

This rate control approach presents two significant concerns. First, unlike the CCSDS compressor, the compressed data volume produced by the JPEG2000 rate control procedure may not exactly meet the data volume target, which implies that users must contend with a variable output rate. More significantly, this optional rate-distortion optimization is based on an iterative algorithm that relies on Lagrangian multipliers and has a high implementation complexity.

Without the rate/distortion optimal control, the JPEG2000 algorithm may not reach the targeted rate on some images (e.g., for image b6l, the bit rate of 2.0 bits/pixel cannot be reached without this rate control; instead a bit rate of 1.4 bits/pixel was obtained).

6.3.4 LIMITED MEMORY COMPRESSION

Memory-efficient compression can be particularly important for space-borne applications. As described in 3.5 and 4.3, implementation memory for the CCSDS compressor can be reduced by careful selection of the number of blocks per segment, *S*, e.g., by performing strip compression. However, increased memory efficiency may come at the price of reduced compression effectiveness, as illustrated in 3.5.

Analogous to strip compression with the CCSDS compressor (see 3.5), a scan-based mode is included in the optional Part 2 of the JPEG2000 standard. This mode is available in the Verification Model of JPEG2000 but is not implemented in the well known JPEG2000 decoders such as Kakadu (reference [12]) and JasPer (reference [13]), and is not included in all commercial decoders.

To make a fair comparison between the CCSDS and JPEG2000 compressors under a limited-memory scenario, scan-based JPEG2000 should be compared to strip compression in CCSDS. Such comparisons, along with frame-based comparisons, are detailed in section 5. In frame-based compression, the JPEG2000 compressor can offer several dB improvement in PSNR image quality compared to the CCSDS Recommendation. However, this benefit largely evaporates (to less than one dB) when comparing scan-based JPEG2000 to CCSDS strip compression. In such a memory-limited scenario, the improvement in compression effectiveness offered by JPEG2000 is likely to be outweighed by the higher implementation complexity.

6.3.5 SUMMARY

JPEG2000 includes capabilities that the Recommendation lacks. It can be used to provide Region-Of-Interest (ROI) coding; i.e., certain regions of an image can be encoded with higher fidelity than remaining portions of the image. JPEG2000 allows compressed image data to be arranged so that image *resolution* (rather than overall image distortion) progressively improves as more compressed data are received. It also includes definitions to

handle multi-component images (e.g., RGB color images), while the Recommendation addresses only grayscale images.

JPEG2000 code-blocks are independently encoded, and thus compression is amenable to parallelization at the code-block level. By contrast, the tree-based intra-subband coding performed under CCSDS limits parallelization to the segment level or higher. Independent coding of JPEG2000 code-blocks and resynchronization markers also offer improved error-containment compared to CCSDS.

In JPEG2000, the use of context modeling combined with arithmetic coding and Lagrangian rate-control leads to high compression effectiveness. In experiments on test images (see section 5), the JPEG2000 compressor provides significantly better lossy compression performance in frame-based compression, but only slightly better compression performance (within one dB PSNR) when implementation memory is significantly constrained.

However, some components of JPEG2000 that help to provide high compression performance (context modeling, arithmetic coding, Lagrangian rate-distortion optimization) also have high implementation complexity. This limits the suitability of JPEG2000 for space-borne missions with high data-throughput rates and limited capacity of acquisition, storage and transmission.

This higher complexity is primarily due to the block-based entropy coding approach and the rate-allocation optimization algorithm.

6.4 DWT SELECTION

6.4.1 OVERVIEW

Reference [19] has shown that the recommended 9/7 biorthogonal float DWT possesses several desirable mathematical properties for decorrelation and exhibits the best compression performance among wavelet bases of similar length. Because of its excellent performance in terms of both rate/distortion and visual representation, it had already been selected for the JPEG2000 recommendation. Its consistently superior performance has been confirmed in compression trials. The 9/7 biorthogonal float DWT was therefore selected for this Recommendation.

As discussed in 3.3, a floating-point DWT does not provide lossless compression and requires floating-point calculations, and for these reasons an integer DWT was also selected. In the remainder of this section, the considerations and motivation that led to the selection of the particular integer DWT included in the Recommendation are discussed. Evaluation of candidate integer DWTs was based on results in reference [20] and on compression trials conducted by the working group.

6.4.2 DWT BASIS

6.4.2.1 General

The 5/3 integer DWT recommended as part of JPEG2000 has low complexity and provides excellent lossless compression effectiveness. However, for lossy compression even at high bit rates the performance rapidly decreases compared to 9/7 integer DWTs. This motivated the consideration of two 9/7 integer DWTs, referred to as '9/7F' and '9/7M', following the terminology of reference [20].

6.4.2.2 Complexity

The computational complexity of several wavelet filters using the lifting scheme is tabulated in reference [20]. Table 6-2 shows the number of additions, bit shifts, and multiplications required by the 5/3, the 9/7M, and the 9/7F filters when symmetries in the filter coefficients and the complexity reduction offered by lifting schemes are exploited. The table provides two sets of numbers. The first set indicates the number of operations in the case of a straightforward implementation of the transform. The second set, shown in parentheses, gives the number of operations required if multiplication operations are replaced by combinations of bit shift and addition operations. This latter metric is of particular interest either for hardware implementations or for software implementations on architectures where integer multiplications are more costly than addition and bit shift operations. In the case where both numbers are the same, only one figure is given.

Table 6-2: Computational Complexity of the 5/3, 9/7M and 9/7F Filters (from reference [20])

Transform	Additions	Shifts	Multiplications	Total Operations
5/3	5	2	0	7
9/7M	8 (9)	2 (3)	1 (0)	11 (12)
9/7F	12(26)	4 (18)	4 (0)	20 (44)

Table 6-2 shows that the 5/3 filter has the lowest complexity. The 9/7M filter is close to the 5/3, but the 9/7F has significantly higher complexity.

6.4.2.3 Performance

Performance has been evaluated after extensive compression trials. Some of the results are documented in annex E.

For lossless compression, table E-6 provides comparison between the 9/7M and the 5/3 integer filters on a set of twenty test images, using the same bit-plane encoder in reference [1]. It is clear that the 9/7M DWT performs slightly better on most of the test images. The 5/3 transform also fares reasonably well considering its very low computational complexity.

For lossy compression at low bit rates, the 9/7F transform performs best for objective lossy compression. As the bit rate increases, lossy compression performance becomes increasingly influenced by the lossless performance characteristics of the transforms. When subjective performance is considered, the 9/7F, the 5/3, and the 9/7M perform about equally well (reference [20]).

Compression effectiveness of reversible integer-to-integer wavelet transforms is influenced by several factors:

- **Number of lifting steps:** The integer DWTs analyzed can be implemented efficiently using lifting schemes (reference [21]), described in annex D. Since each additional lifting step tends to increase the approximation error, transforms with fewer lifting steps generally perform better, everything else being equal.
- **Rounding function:** The function used to round results to integer values affects the difference in performance apparent between reversible integer-to-integer transforms and their conventional counterparts. In calculating the results, a biased floor function is used: $Q(x) = \left\lfloor x + \frac{1}{2} \right\rfloor$. It is conceivable to remove the bias of one-half, however. Such a change would reduce computational complexity at the expense of introducing increased mean error.
- **Image dynamic range:** As the dynamic range of the signal grows, the relative errors introduced by rounding become smaller. For images with eight-bit pixels, there is a clear difference in PSNR performance between the reversible integer-to-integer and conventional versions of a transform, especially at higher bit rates. For images with 12-bit pixels, however, there is little difference in performance.
- **Bit rate:** As bit rate decreases, the difference in compression performance between reversible integer-to-integer and conventional versions of a transform diminishes. At a sufficiently low bit rate, the quantization of transform coefficients becomes so coarse that errors introduced by rounding intermediate results tend to be masked by quantization of the coefficients themselves.

6.4.2.4 Summary

Comparing the 9/7M and 9/7F filters, the 9/7M filter provides better compression performance, especially for lossless compression and at high bit rates. At low bit rates, i.e., high compression ratios, the 9/7F tends to perform somewhat better, but this does not justify a preference over the 9/7M, especially since users who are primarily interested in low rate compression might be inclined to use the float DWT.

The 5/3 filter has the lowest computational complexity. However, the 9/7M has moderately higher complexity and provides significantly better lossy compression effectiveness.

For these reasons, the 9/7M was considered the best compromise in terms of complexity and performance and was therefore selected as the integer DWT for the Recommendation. The formulation of the recommended 9/7M filter described in reference [1] has a slight difference in the rounding operations than what is in reference [20]. The formulation in reference [1] reduces mean-error bias during reconstruction.

6.4.3 NUMBER OF DECOMPOSITION LEVELS

A single-level two-dimensional wavelet decomposition produces four subbands of wavelet coefficients, called LL-, HL-, LH-, and HH-subbands, each having half the width and half the height of the input image. Of these four subbands, generally only the LL-subband retains significant intra-band correlation of its coefficients. Indeed, the LL-band visually appears as a smoothed, low-resolution version of the original image. A multi-level 2-d DWT attempts to decorrelate the data further by iteratively applying a single-level 2-d DWT to the LL-subband produced by the previous stage. This is illustrated in figure 2-3 for a three-stage decomposition.

An increased number of levels also allows increased exploitation of inter-band correlation. Increasing the number of stages of wavelet decomposition thus can further decorrelate the image data and thus provide increased compression performance. However, the fraction of image data affected by subsequent decomposition stages decreases exponentially, and after a point the gains provided by further decomposition stages become negligible as shown in figure 6-2 on representative images, i.e., 8-bit coastal_b6h, 10-bit ice_2kb4, 12-bit solar and 16-bit p160_b_f, within the CCSDS test data set processed by the JPEG2000 in frame-based mode. In practice, the number of decomposition stages used by wavelet-based image compressors is often not larger than five.

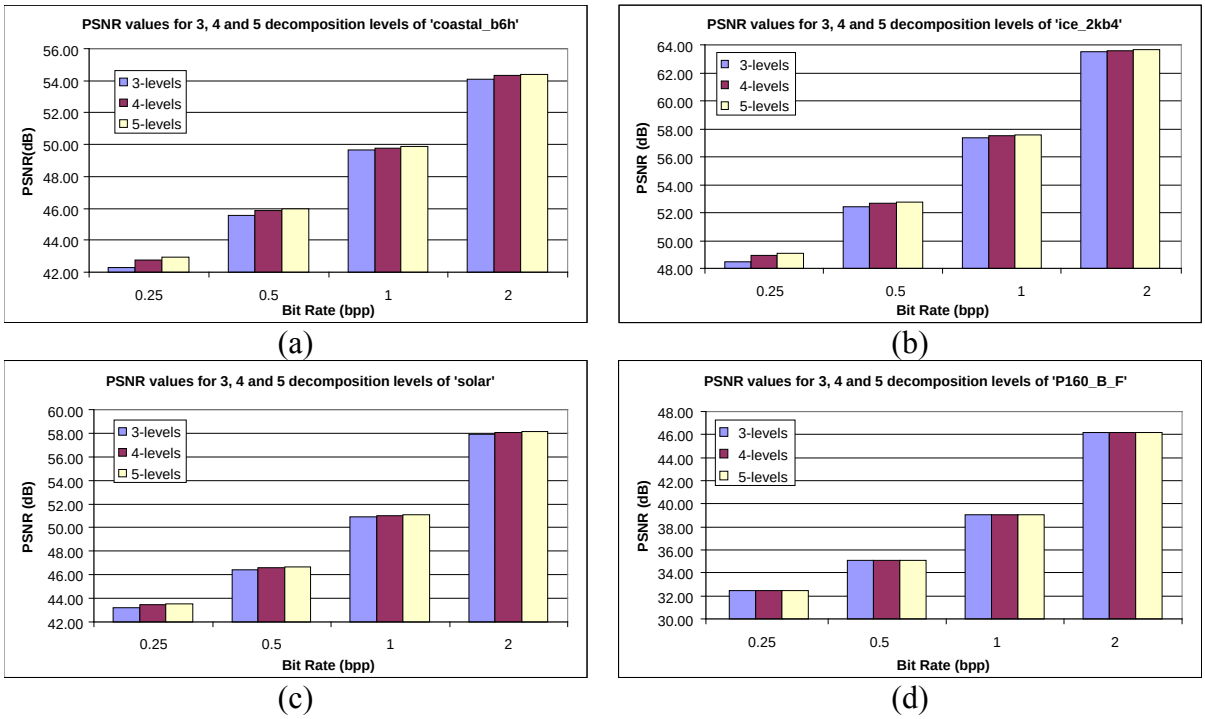


Figure 6-2: PSNR Comparison for DWT Levels at 3, 4 and 5 on (a) 8-Bit, (b) 10-Bit, (c) 12-Bit and (d) 16-Bit Images Using JPEG2000 Coder in Frame-Based Mode

It is seen from this experiment that more decomposition levels do not improve the PSNR values of more than 0.5 dB from three to four levels and not more than 0.63 dB from three to five levels. These PSNR gains are relatively small considering the much higher complexity associated with implementing a larger number of decomposition levels.

Using a smaller number of decomposition levels reduces implementation complexity because fewer operations are required to compute the transformed image. In the case of the CCSDS image compressor, a smaller number of decompositions has the effect of also reducing the size of a block of coefficients. At three decomposition stages, a block consists of 64 DWT coefficients, and the LL subband (i.e., the DC coefficient) represents $1/64 \approx 1.6\%$ of the DWT coefficients. The differential coding scheme applied to the DC coefficients (refer to 2.5.2) exploits correlation remaining among these coefficients.

Using a smaller number of decomposition stages also has the effect of slightly reducing the impact of loss or corruption of encoded data. As described in 2.4.2.2 and 2.4.2.3, because segments are defined in the DWT domain rather than in the image domain, when data for a segment is lost, a few pixels near the borders of that segment may appear blurred. Using a smaller number of decomposition stages reduces the number of pixels affected by such a loss.

6.4.4 SUBBAND WEIGHTS

For effective operation, the BPE requires that, on average, encoding the same bit plane from two different subbands will result in roughly the same reduction in image distortion per bit. This is achieved when the float DWT is used (because the float DWT is nearly isometric), but not when the integer DWT is used. Thus, when the integer DWT is used, the subbands must be scaled: all subband coefficients are multiplied by their respective weight factors before encoding. This subsection describes how the default subband weights were determined.

Annex E presents an analytical derivation of subband weight factors under certain assumptions (e.g., a linear DWT, and uniform distribution of quantization error, as might be expected at high bit rates). The analytically derived weight sets presented in annex E4 serves as a reference point for comparison.

In fact, the assumptions under which the weights were derived are not strictly true (the integer DWT is not linear), and test results demonstrate that a different set of weights (incorporated as the default weighting for the Recommendation) offers improved compression effectiveness on images in the test set.

As part of these tests, the different choices of subband weights were also evaluated for the 9/7F DWT, and the results (demonstrating better compression effectiveness of the 9/7M DWT) motivated the selection of the 9/7M DWT as part of the Recommendation.

The average image distortion obtained at four rates (0.25, 0.5, 1.0, and 2.0 bits/pixel) on the test images under several different combinations of DWT and weight factors are compared:

- Scenario 3: Integer 9/7M filter with analytically derived weights given in annex E4;
- Scenario 4: Integer 9/7M filter with the weights ultimately adopted as the default weights for the compression Recommendation;
- Scenario 5: Integer 5/3 DWT with analytically derived weights given in annex E4;
- Scenario 6: Integer 9/7F filter with analytically derived weights given in annex E4;
- Scenario 7: Integer 9/7F filter with all subbands weighted equally;
- Scenario 8: 9/7 float DWT (for reference purposes).

Figure 6-3 shows the performance results averaged over an earlier set of test images and over four bit rates. Annex E details the results of the underlying individual trials. (At the time these results were produced, the compression Recommendation had not been finalized; the Recommendation ultimately adopted in fact yields slightly higher values of PSNR.)

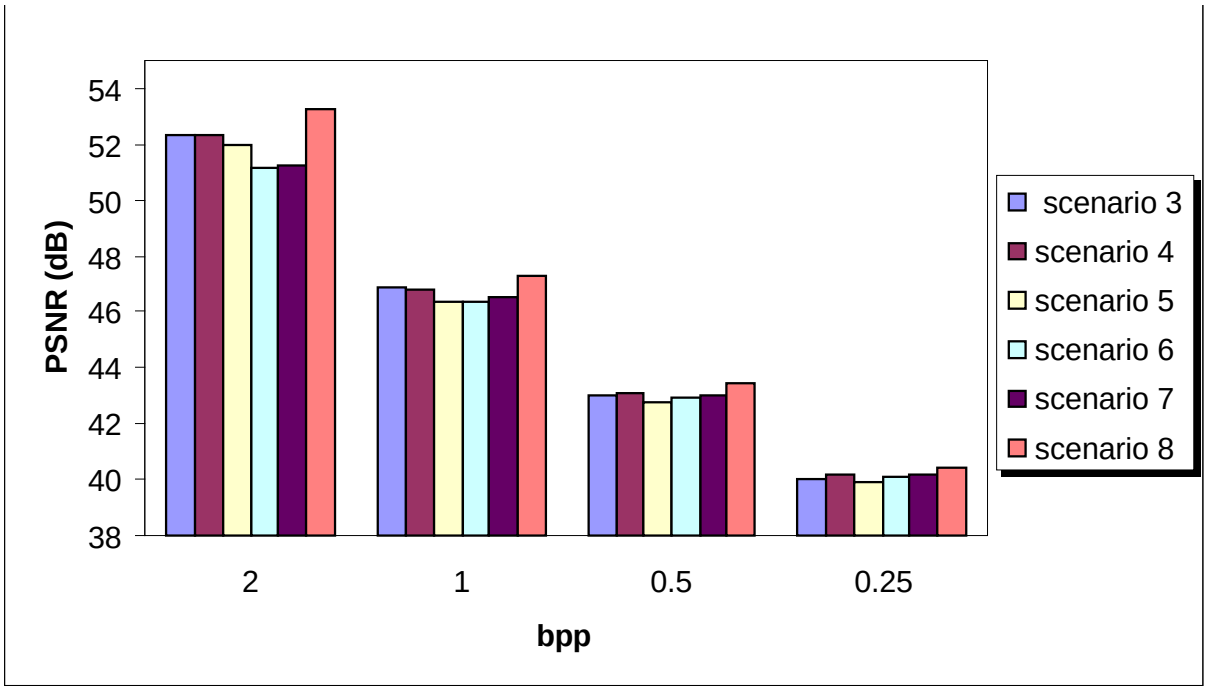


Figure 6-3: Performance Averages for an Earlier CCSDS Data Set

It follows from figure 6-3 that the selected weight factors provide roughly 0.1 dB improvement over the analytically determined weights. The results also demonstrate that, for the subband weights considered, the 9/7M DWT tends to offer improved compression effectiveness compared to the 9/7F DWT.

ANNEX A

EXAMPLES OF HEADER BITS

The binary form of the header bits for the six examples in 3.8 is presented in table A-1.

Table A-1: Header Bits for the First Segment Expressed in Hexadecimal Format Unless Otherwise Indicated

Header	Example 1	Example 2	Example 3	Example 4	Example 5	Example 6
Part 1A	C018a7	800847	8018a7	8018a7	801ca7	801897
Part 1B	00	n/a	n/a	n/a	n/a	80 (last segment)
Part 2	0000000060	0000000060	0000001060(DCStop='1') 0000000080(StageStop='00') 00000000a0(StageStop='01') 00000000c0(StageStop='10') 00000000e0(StageStop='11')	0008f9a060	0002cac260	0001770070 (First 62 segments) 0000bb8070 (Last segment if Part 2 included)
Part 3	00010c	00080c	00040c	0015ec	00400c	0003fc
Part 4	8800020000000000	8c00400000000000	8800200000000000	880022d000000000	0a00800000000000	88001f4000000000

ANNEX B

AVAILABLE SOFTWARE AND TEST DATA

B1 AVAILABLE SOFTWARE

Users of the Recommendation can test the compression performance by acquiring open-source software via links provided at http://cwe.ccsds.org/sls/docs/sls-dc/. The CCSDS organization and its participating agencies are not liable for any damages to the user data or processing system as a result of using these software implementations. These implementations were developed only to demonstrate compression performance, no optimization in speed or programming was guaranteed. These available software implementations have passed the verification test data set described in B2.

B2 IMPLEMENTATION VERIFICATION TEST

A set of implementation verification tests compiled based on the Recommendation was produced by an independent software development effort and cross-verified with the results from the software aforementioned in B1. This set of tests exercises both the integer and the float DWTs, as well as rate-limited and quality-limited compression. The test set uses only the test image set described in B3 as well as figure 4-1 of this Report; both are available at the location given in B1.

The verification test set includes a variety of cases to be encountered in missions projected for both push-broom as well as frame sensors, However, it is not intended fully to exercise every compression option or the full range of parameter values specified in the Recommendation, some of which are simply beyond the current sensor implementation technology.

B3 IMAGE SET

The CCSDS reference test image set includes a variety of space imaging instrument data such as solar, stellar, planetary, Earth observations, optical and radar, and meteorological.

The image set includes dynamic ranges from 8 to 16 bits/pixel. It should be noted, however, that the nominal dynamic range may be somewhat misleading. For example, the foc image has a nominal dynamic range of 12 bits/pixel (because this is the dynamic range of the camera that produced the image), but in fact this particular image can be represented using 8-bit pixels (i.e., the four most significant bits of each pixel are unused). Similarly, for the SAR image (sar_16bit), 99.7% of the pixels use no more than a 12-bit dynamic range, and for the wfpc image, 99.8% of the pixels need only a 9-bit dynamic range. Users of the test images should keep this in mind; compression results (especially lossless) may be misleading.

The reference images are listed in table B-1 and illustrated in B4.

Table B-1: CCSDS Reference Image Set

Image Name	Source - Copyright	Size (width × height)	Dynamic range (bits/pixel)
coastal_b1	Landsat - NASA	1024 × 1024	8
coastal_b2	Landsat - NASA	1024 × 1024	8
coastal_b3	Landsat - NASA	1024 × 1024	8
coastal_b4	Landsat - NASA	1024 × 1024	8
coastal_b5	Landsat - NASA	1024 × 1024	8
coastal_b6l	Landsat - NASA	512 × 512	8
coastal_b6h	Landsat - NASA	512 × 512	8
coastal_b7	Landsat - NASA	1024 × 1024	8
coastal_b8	Landsat - NASA	2048 × 2048	8
europa3	Galileo Image from Europa - NASA	557 × 600	8
marstest	Mars Pathfinder - NASA	512 × 512	8
lunar	Galileo – NASA	512 × 512	8
spot-la_b3	SPOT 3 Imaging - CNES	500 × 500	8
spot-la_panchr	SPOT 3 Imaging - CNES	1000 × 1000	8
ice_2kb1	NOAA Polar Orbiter (AVHRR) - NOAA	2048 × 2048	10
ice_2kb4	NOAA Polar Orbiter (AVHRR) - NOAA	2048 × 2048	10
india_2kb1	NOAA Polar Orbiter (AVHRR) - NOAA	2048 × 2048	10
india_2kb4	NOAA Polar Orbiter (AVHRR) - NOAA	2048 × 2048	10
ocean_2kb1	NOAA Polar Orbiter (AVHRR) - NOAA	2048 × 2048	10
ocean_2kb4	NOAA Polar Orbiter (AVHRR) - NOAA	2048 × 2048	10
landesV_G7_10b	SPOT 5 Imaging - CNES	454 × 2381	10
marseille_G6_10b	SPOT 5 Imaging - CNES	528 × 1856	10
pleiades_portdebouc_b3	Simulated PLEIADES - CNES	1376 × 320	12
pleiades_portdebouc_pan	Simulated PLEIADES - CNES	1400 × 5504	12
solar	Big Bear Solar Observatory - NASA	1024 × 1024	12
sun_spot	Big Bear Solar Observatory - NASA	512 × 512	12
wfpc	Hubble Space Telescope- NASA	800 × 800	12
foc	Hubble Space Telescope - NASA	1024 × 512	12
sar_16bit	ERS-1 - ESA	512 × 512	16
P_160_B_F	Picard Imager (IAS) - CNRS	2048 × 2048	16

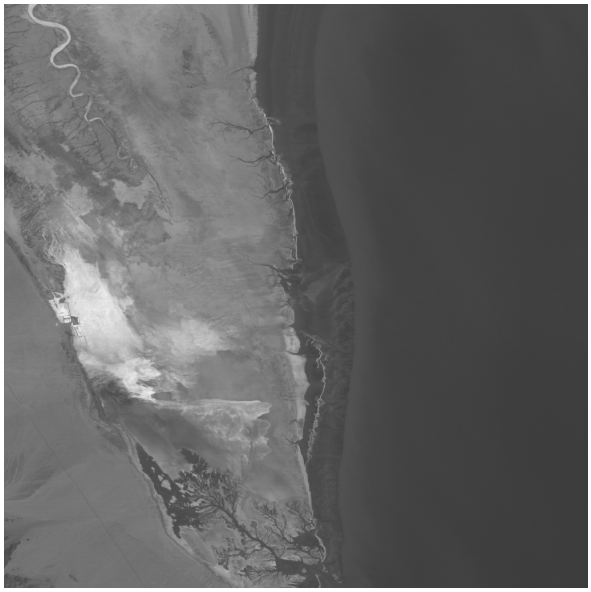
The images are provided for testing purposes only by the CCSDS Data Compression Working Group. They are stored as binary data without header or trailer (raw data). Images with eight-bit pixel depths are stored in byte format, while those with pixel depths above eight bits are stored in unsigned 2×integer format in raw data form and in PGM format.

Copyright of the images is acknowledged as follows:

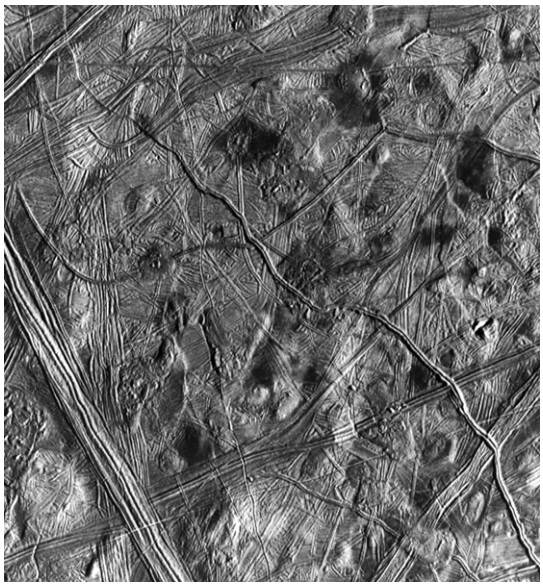
'Copyright *xxx*.[8] All rights reserved. This image may be used and distributed without restrictions provided that this copyright statement is retained and that any derivative work acknowledges the origin of the information.'

Test images are available at http://cwe.ccsds.org/sls/docs/sls-dc/.

B4 IMAGE PREVIEWS



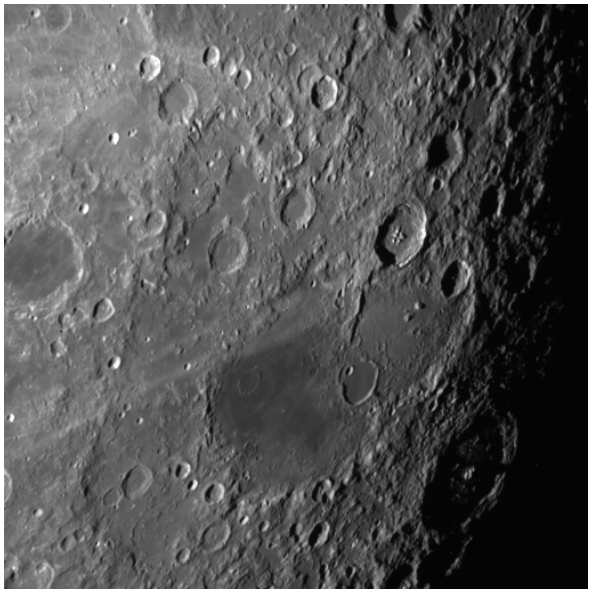
coastal_b2



europa3

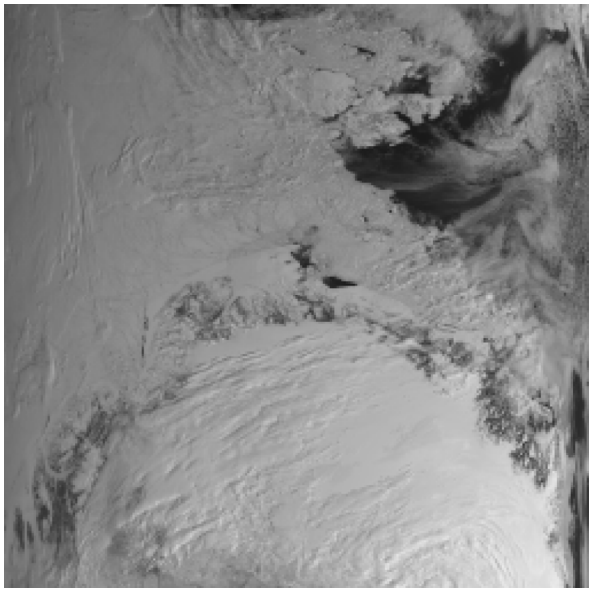
[8] *xxx*=ESA, NASA, CNES, NOAA, or CNRS.

marstest

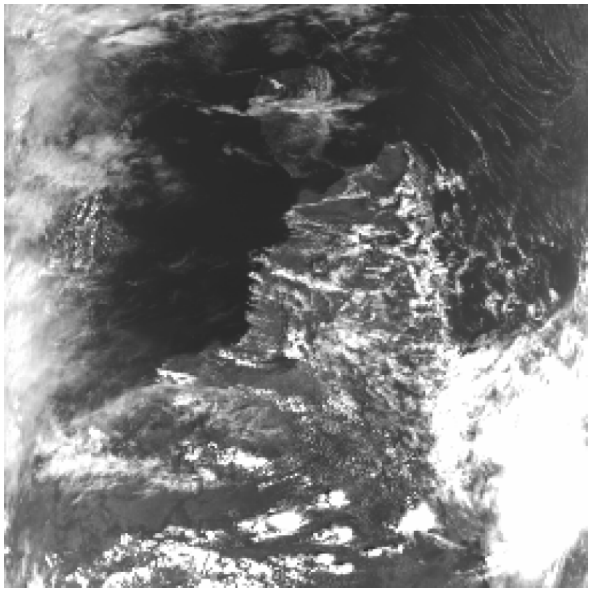


lunar

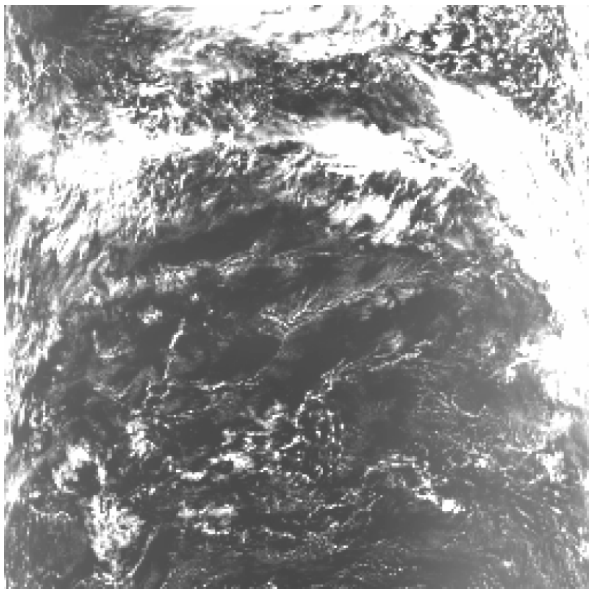
spot-la_panchr



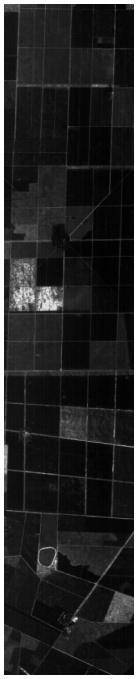
ice_2kb1



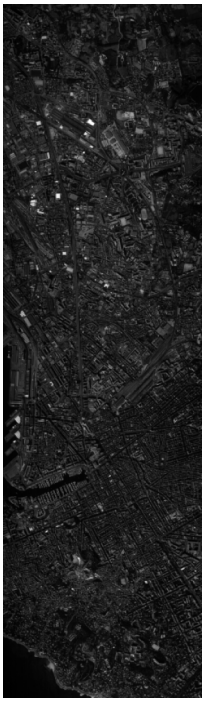
india_2kb1



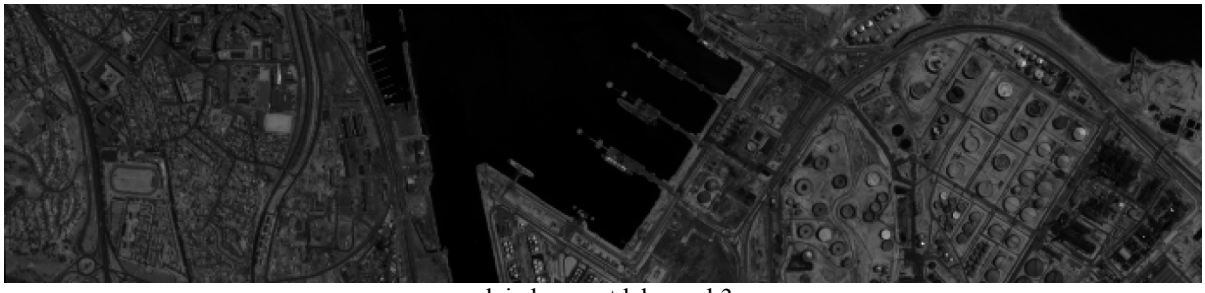
ocean_2kb1



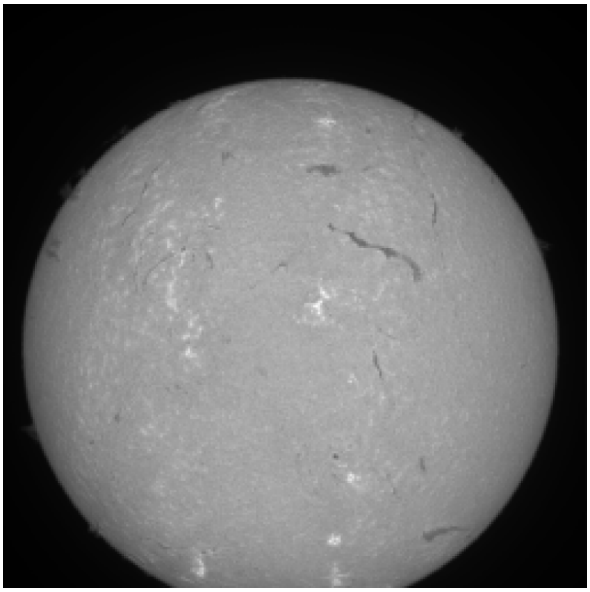
landesV_G7_10b



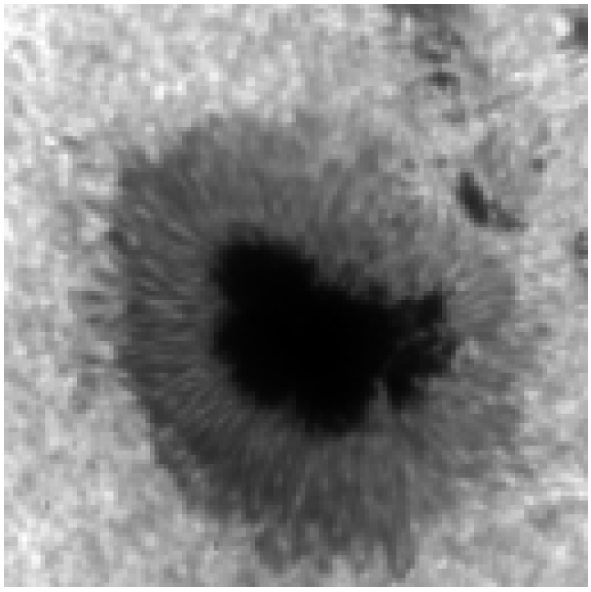
marseille_G6_10b



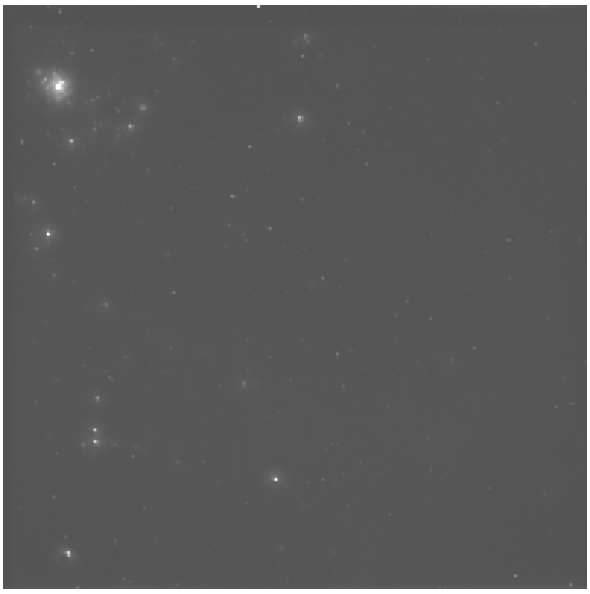
pleiades_portdebouc_b3



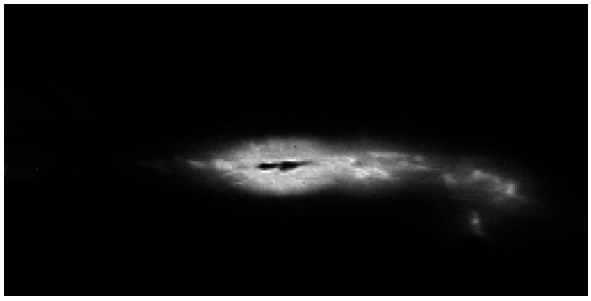
solar



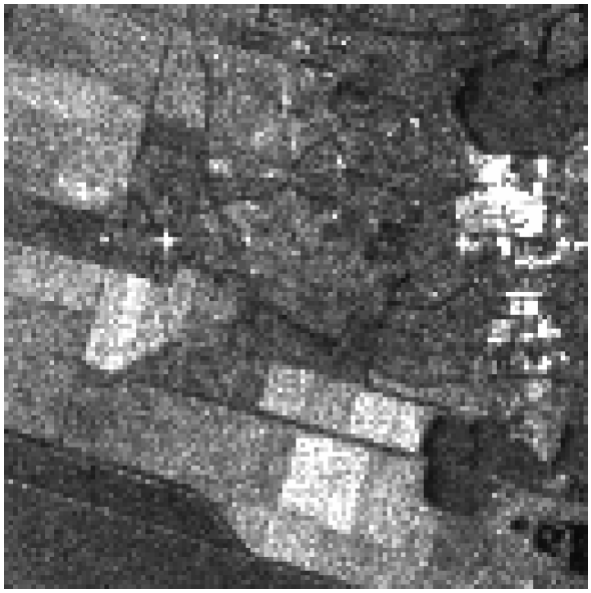
sun_spot



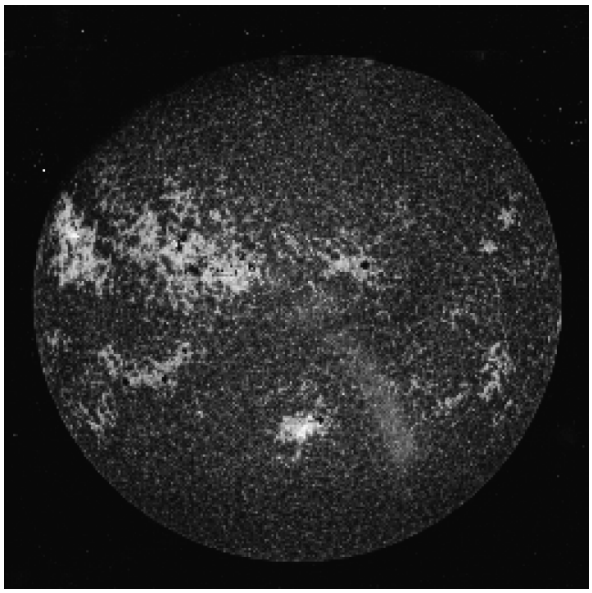
wfpc



foc



sar



p160_b_f

ANNEX C

LOSSY COMPRESSION RESULTS

The lossy image-compression results on the test images, using each of the compression algorithms identified in section 5, are detailed in table C-1 at four coding rates: 0.25, 0.5, 1.0, and 2.0 bits/pixel. Under each technique, the left column is the PSNR value, while the right column is the maximum absolute error value. The CCSDS strip-processing results were obtained using the software implementation by NASA's Goddard Space Flight Center (GSFC).

Table C-1: Detailed Performance Data of Different Algorithms on CCSDS Reference Image

Image File Name	Dynamic Range	Bit rate (bpp)	CCSDS Strip-Based		JPEG2000 Scan-Based (with Lagrangian)		SPIHT		JPEG2000 Frame-Based (with Lagrangian)		CCSDS Frame-Based	
costal_b1	8	0.25	40.90	27	40.77	31	42.20	16	41.27	26	40.95	27
		0.5	43.79	15	43.85	16	44.82	9	44.15	13	43.80	15
		1	46.51	7	46.96	7	47.79	5	47.21	6	46.40	7
		2	50.66	4	50.98	3	51.81	3	51.45	3	50.64	3
costal_b2		0.25	40.84	26	40.70	31	42.21	15	41.22	23	40.86	24
		0.5	44.03	15	44.07	15	45.14	9	44.45	13	44.05	15
		1	47.18	7	47.82	7	48.62	5	48.08	6	47.07	7
		2	51.42	3	52.27	3	52.91	2	52.42	3	51.35	3
costal_b3		0.25	39.13	35	38.95	38	40.45	23	39.54	31	39.24	27
		0.5	42.49	20	42.41	20	43.49	12	42.83	15	42.59	15
		1	46.00	7	46.17	8	46.96	6	46.40	7	45.95	7
		2	50.29	4	50.59	4	51.43	3	50.95	3	50.48	4
costal_b4		0.25	42.13	24	42.31	25	43.66	15	42.84	25	42.17	23
		0.5	45.72	15	45.89	13	46.93	8	46.28	11	45.78	15
		1	49.56	7	50.03	5	50.75	4	50.34	5	49.59	7
		2	54.14	3	55.09	2	55.99	2	55.37	2	53.66	3
costal_b5		0.25	39.79	36	40.00	31	41.35	21	40.63	27	39.90	28
		0.5	43.31	20	43.47	19	44.68	12	44.08	14	43.49	14
		1	47.37	7	47.69	8	48.73	5	48.15	7	47.35	7
		2	52.19	3	53.00	3	53.58	2	53.09	2	52.00	3
costal_b6h		0.25	41.56	16	41.85	17	42.38	12	42.29	15	41.69	13
		0.5	43.83	10	45.26	9	45.62	7	45.56	9	43.61	12
		1	47.28	6	49.43	7	49.48	4	49.64	5	47.37	7
		2	52.27	3	53.83	2	54.02	2	54.13	2	52.24	3

Image File Name	Dynamic Range	Bit rate (bpp)	CCSDS Strip-Based		JPEG2000 Scan-Based (with Lagrangian)		SPIHT		JPEG2000 Frame-Based (with Lagrangian)		CCSDS Frame-Based	
costal_b6l	8	0.25	47.14	8	47.25	9	47.85	5	47.57	9	47.16	8
		0.5	48.60	6	50.03	5	50.46	3	50.31	5	48.49	6
		1	52.01	3	53.48	3	53.40	2	53.72	3	51.97	3
		2	55.91	2	60.40	1	60.80	1	60.98	1	55.90	3
costal_b7		0.25	39.32	37	39.51	39	40.74	22	40.07	25	39.45	27
		0.5	42.43	24	42.61	20	43.71	13	43.15	14	42.71	14
		1	46.00	11	46.32	9	47.26	7	46.72	7	46.08	7
		2	50.43	5	50.96	4	51.79	3	51.32	3	50.67	3
costal_b8		0.25	41.34	26	41.31	23	42.07	15	41.65	20	41.52	15
		0.5	42.56	14	43.10	13	43.38	9	43.23	13	42.49	14
		1	44.57	7	45.07	7	45.42	7	45.16	8	44.52	7
		2	48.34	6	49.29	4	49.70	4	49.81	4	48.19	7
europa3		0.25	18.77	152	19.26	149	19.56	145	19.52	147	18.98	149
		0.5	21.24	120	21.61	115	21.98	101	21.92	107	21.53	112
		1	24.89	87	25.59	86	25.76	64	25.78	79	24.95	84
		2	30.95	52	31.63	35	31.50	34	31.89	32	31.05	39
marstest		0.25	27.16	80	27.37	84	28.07	74	27.89	74	27.35	80
		0.5	30.27	46	30.47	60	31.03	39	30.96	50	30.45	46
		1	34.41	35	34.83	36	35.30	23	35.31	29	34.67	26
		2	40.80	14	41.50	15	42.21	10	42.23	10	41.19	16
lunar		0.25	27.71	82	28.32	104	29.00	71	28.65	80	27.86	82
		0.5	30.92	49	31.33	47	32.06	51	31.78	47	31.07	51
		1	35.41	26	36.00	28	36.51	27	36.44	26	35.54	26
		2	41.78	14	42.69	13	43.05	9	42.93	9	41.92	13
spot-la_b3		0.25	30.54	80	30.48	81	31.17	54	30.83	68	30.70	55
		0.5	32.82	48	32.90	46	33.55	37	33.29	42	33.03	44
		1	35.91	27	36.14	28	36.87	26	36.57	25	36.15	27
		2	41.12	13	41.70	12	42.72	9	42.48	9	41.68	14
spot-la_panchr		0.25	32.22	62	32.40	78	32.86	62	32.77	66	32.31	62
		0.5	34.92	50	35.03	40	35.57	37	35.40	36	35.11	30
		1	38.51	26	38.86	25	39.39	16	39.32	18	39.07	17
		2	44.42	10	44.74	11	45.52	8	45.57	6	45.08	7

Image File Name	Dynamic Range	Bit rate (bpp)	CCSDS Strip-Based		JPEG2000 Scan-Based (with Lagrangian)		SPIHT		JPEG2000 Frame-Based (with Lagrangian)		CCSDS Frame-Based	
ice_2kb1	10	0.25	38.93	135	39.12	160	39.98	95	39.56	140	39.03	121
		0.5	42.17	84	42.36	98	43.10	61	42.70	73	42.31	65
		1	46.48	51	46.81	49	47.56	33	47.21	40	46.75	35
		2	53.02	17	53.56	22	54.54	10	54.29	11	53.44	17
ice_2kb4		0.25	47.78	48	47.48	116	49.32	34	48.48	47	48.08	51
		0.5	51.67	33	51.74	35	53.00	17	52.39	26	51.91	30
		1	56.44	15	56.81	14	57.67	8	57.39	11	56.92	15
		2	62.64	7	63.13	5	63.70	4	63.52	3	63.22	3
india_2kb1		0.25	34.84	271	35.21	322	36.74	145	36.27	251	35.71	234
		0.5	38.46	186	38.71	189	40.50	107	40.04	129	39.42	135
		1	43.41	100	43.80	98	45.87	38	45.51	56	45.11	60
		2	51.12	32	51.68	32	53.91	16	53.60	13	52.76	16
india_2kb4		0.25	37.90	421	38.03	288	40.35	117	39.73	202	39.15	117
		0.5	41.90	113	42.14	267	44.59	61	44.18	61	43.56	64
		1	47.14	64	47.65	65	50.46	28	50.18	26	49.18	31
		2	54.87	26	55.58	20	58.56	8	58.48	8	58.00	7
landesV_ G7_10b		0.25	40.80	107	41.23	91	42.05	51	42.04	64	41.49	70
		0.5	42.90	66	43.80	57	43.93	41	44.31	43	43.05	48
		1	45.91	40	48.57	38	47.35	21	49.26	22	46.02	34
		2	52.31	18	54.74	12	54.00	9	55.12	9	52.44	14
marseille_G 6_10b		0.25	28.41	333	28.88	334	29.57	233	29.43	299	28.57	333
		0.5	31.65	206	31.98	237	32.50	211	32.37	205	31.85	208
		1	35.43	114	35.95	106	36.29	116	36.27	103	35.53	138
		2	41.47	63	42.54	56	42.73	40	42.93	39	41.79	63
ocean_ 2kb1		0.25	36.26	238	36.68	251	37.54	142	37.29	178	36.69	238
		0.5	39.65	128	40.02	168	40.96	75	40.64	105	40.07	133
		1	44.18	64	44.72	75	45.71	39	45.49	49	45.04	63
		2	51.24	30	51.89	24	52.93	17	52.85	16	52.08	17
ocean_ 2kb4		0.25	42.32	114	42.24	223	44.47	81	44.00	98	43.41	105
		0.5	46.41	55	46.62	74	48.30	33	48.00	36	47.62	35
		1	51.29	29	51.73	30	53.30	16	53.08	17	52.71	16
		2	58.25	15	58.96	11	60.46	6	60.36	6	59.41	8

Image File Name	Dynamic Range	Bit rate (bpp)	CCSDS Strip-Based		JPEG2000 Scan-Based (with Lagrangian)		SPIHT		JPEG2000 Frame-Based (with Lagrangian)		CCSDS Frame-Based	
pleiades_portdebouc_b3	12	0.25	31.72	1529	32.52	1372	33.03	733	32.94	1057	31.92	774
		0.5	35.43	687	35.85	978	36.29	394	36.07	448	35.60	444
		1	39.75	384	40.23	290	40.61	229	40.50	262	39.95	258
		2	46.28	130	47.06	124	47.47	119	47.40	92	46.40	128
pleiades_portdebouc_pan		0.25	36.21	1216	36.97	998	38.23	543	38.00	723	37.25	1100
		0.5	40.65	447	41.14	565	42.13	272	41.96	337	41.44	303
		1	45.31	215	45.89	235	46.76	135	46.69	146	46.07	139
		2	51.22	130	51.88	91	52.80	66	52.82	50	51.62	68
solar		0.25	42.16	335	42.30	300	43.65	184	43.22	270	42.85	210
		0.5	45.27	184	45.41	184	46.70	125	46.42	143	46.06	161
		1	48.88	109	49.31	107	51.01	71	50.90	81	50.34	79
		2	54.78	65	55.62	55	57.97	30	57.88	31	56.87	34
sun_spot		0.25	48.61	200	48.35	363	50.03	117	49.62	219	49.02	192
		0.5	52.40	94	52.71	146	53.24	65	53.06	91	52.46	64
		1	55.47	54	55.90	47	56.27	32	56.10	32	55.53	46
		2	59.83	30	60.93	21	61.22	16	61.23	16	60.03	26
wfpc		0.25	64.79	47	65.36	39	65.94	19	65.73	17	64.89	29
		0.5	66.04	21	66.66	15	66.82	13	66.80	13	66.04	14
		1	67.69	11	68.34	8	68.58	7	68.53	7	67.66	10
		2	71.10	7	72.83	4	73.15	4	73.26	4	70.99	7
foc		0.25	62.93	39	63.32	37	66.87	14	66.50	21	65.27	25
		0.5	64.44	27	64.91	29	68.89	8	68.75	10	67.08	14
		1	67.05	23	67.91	19	71.21	8	71.02	8	70.03	7
		2	72.12	12	72.99	9	75.42	3	75.50	3	74.68	3
sar16bit	16	0.25	47.83	2913	48.21	2094	48.71	1438	48.48	1725	48.35	1462
		0.5	50.10	1445	50.30	1352	50.80	911	50.62	1155	50.40	1435
		1	53.17	887	53.50	799	53.94	707	53.74	799	53.27	838
		2	58.61	476	59.24	379	59.59	350	59.50	318	58.64	481
P160_B_F		0.25	30.37	18119	31.33	16577	32.50	14029	32.47	16211	31.34	13761
		0.5	33.42	12695	34.11	12421	35.14	9425	35.08	11602	34.48	8872
		1	36.73	8339	37.46	8353	39.12	5149	39.08	6817	38.46	5107
		2	42.46	4301	43.37	4001	46.20	2209	46.16	2059	44.96	2024

For summary purpose, the results in table C-1 are averaged at each bit rate for each dynamic range, as presented in table C-2.

Table C-2: Summary of Performance for (a) CCSDS (Strip-Based) (b) JPEG2000 (Scan-Based) (c) SPIHT (d) JPEG2000 (Frame-Based) (e) CCSDS (Frame-Based)

(a)

CCSDS Strip-Based				
	PSNR Mean Value for each dynamic range			
	8-bit	10-bit	12-bit	16-bit
0.25	36.33	38.41	47.74	39.10
0.50	39.07	41.85	50.71	41.76
1.00	42.54	46.29	54.03	44.95
2.00	47.48	53.12	59.22	50.54
	MAE Mean Value for each dynamic range			
0.25	49.36	208.38	561.00	10516.00
0.50	32.29	108.88	243.33	7070.00
1.00	18.79	59.63	132.67	4613.00
2.00	9.71	26.00	62.33	2388.50

(b)

JPEG2000 Scan-Based				
	PSNR Mean Value for each dynamic range			
	8-bit	10-bit	12-bit	16-bit
0.25	36.46	38.61	48.13	39.77
0.50	39.43	42.17	51.11	42.20
1.00	43.17	47.01	54.60	45.48
2.00	48.48	54.01	60.22	51.30
	MAE Mean Value for each dynamic range			
0.25	52.86	223.13	518.17	9335.50
0.50	31.29	140.63	319.50	6886.50
1.00	18.86	59.38	117.67	4576.00
2.00	8.00	22.75	50.67	2190.00

(c)

SPIHT				
	PSNR Mean Value for each dynamic range			
	8-bit	10-bit	12-bit	16-bit
0.25	37.40	40.00	49.63	40.61
0.50	40.17	43.36	52.35	42.97
1.00	43.73	48.03	55.74	46.53
2.00	49.07	55.10	61.34	52.90
	MAE Mean Value for each dynamic range			
0.25	39.29	112.25	268.33	7733.50
0.50	24.79	75.75	146.17	5168.00
1.00	14.36	37.38	80.33	2928.00
2.00	6.57	13.75	39.67	1279.50

(d)

JPEG200 Frame-Based				
	PSNR Mean Value for each dynamic range			
	8-bit	10-bit	12-bit	16-bit
0.25	36.91	39.60	49.33	40.47
0.50	39.81	43.08	52.18	42.85
1.00	43.49	48.05	55.62	46.41
2.00	48.90	55.14	61.35	52.83
	MAE Mean Value for each dynamic range			
0.25	45.43	159.88	384.50	8968.00
0.50	27.79	84.75	173.67	6378.50
1.00	16.50	40.50	89.33	3808.00
2.00	6.36	13.13	32.67	1188.50

(e)

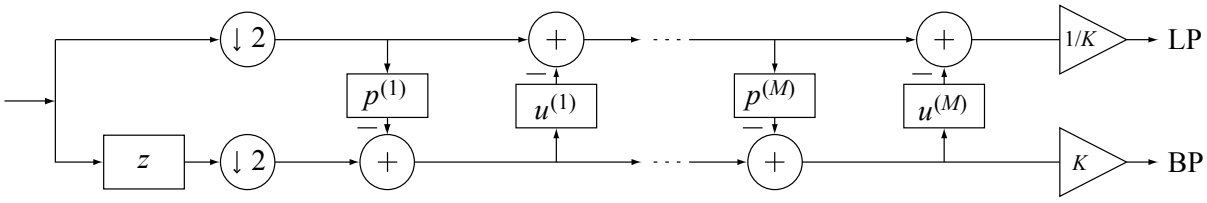
CCSDS Frame-Based				
	PSNR Mean Value for each dynamic range			
	8-bit	10-bit	12-bit	16-bit
0.25	36.44	39.02	48.53	39.85
0.50	39.16	42.47	51.45	42.44
1.00	42.62	47.16	54.93	45.87
2.00	47.58	54.14	60.10	51.80
	MAE Mean Value for each dynamic range			
0.25	44.29	158.63	388.33	7611.50
0.50	28.79	89.75	166.67	5153.50
1.00	17.07	49.00	89.83	2972.50
2.00	8.64	18.13	44.33	1252.50

ANNEX D

DWT LIFTING SCHEME

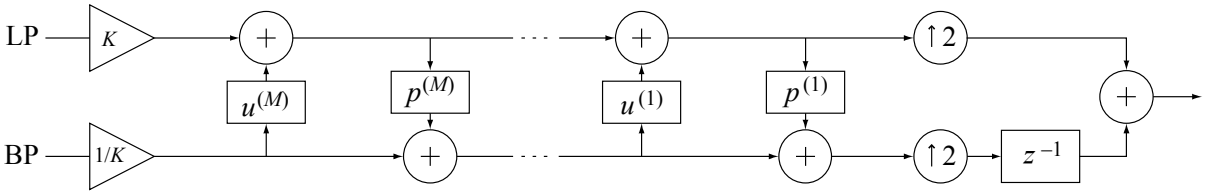
D1 GENERAL DESCRIPTION

In this annex, a general description of the lifting scheme (reference [21]) is given. There are two ways of looking at lifting, either from the basis-function point of view or from the filter point of view. The filter point of view is considered here.



NOTE – The transform proceeds first with the Lazy wavelet, then alternating dual lifting and lifting steps, and finally a scaling. LP=low pass filtered signal, BP=band pass filtered signal.

Figure D-1: The Forward Wavelet Transform Using Lifting



NOTE – The inverse transform proceeds first with a scaling, then alternating lifting and dual lifting steps, and finally the inverse Lazy transform. The inverse transform can immediately be derived from the forward by running the scheme backwards and flipping the signs.

Figure D-2: The Inverse Wavelet Transform Using Lifting

Computing the wavelet transform using lifting steps consists of several stages. The idea is first to compute a trivial wavelet transform (the Lazy wavelet or polyphase transform) and then improve its properties by alternating lifting steps, called prediction and update steps, respectively (see figure D-1). The Lazy wavelet only splits the signal into its even and odd indexed samples:

$$s_l^{(0)} = x_{2l}$$

$$d_l^{(0)} = x_{2l+1}$$

A prediction step consists of applying a filter to the even samples and subtracting the result from the odd ones:

$$d_l^{(i)} = d_l^{(i-1)} - \sum_k p_k^{(i)} s_{l-k}^{(i-1)}$$

An update step does the opposite, applying a filter to the odd samples and subtracting the result from the even samples:

$$s_l^{(i)} = s_l^{(i-1)} - \sum_k u_k^{(i)} d_{l-k}^{(i)}$$

Eventually, after, say, *M* pairs of prediction and update steps, the even samples become the low pass coefficients while the odd samples become the high pass coefficients, up to a scaling factor *K*:

$$y_{2l} = s_l^{(M)} / K \text{ and } y_{2l+1} = K\, d_l^{(M)}$$

As always, the inverse transform can be found by reversing the operations and flipping the signs (see figure D-2). The first computation is thus

$$s_l^{(M)} = K\, s_l \text{ and } d_l^{(M)} = d_l / K$$

Then undo the *M* alternating update and prediction steps:

$$s_l^{(i-1)} = s_l^{(i)} + \sum_k u_k^{(i)} d_{l-k}^{(i)}$$

and

$$d_l^{(i-1)} = d_l^{(i)} + \sum_k p_k^{(i)} s_{l-k}^{(i-1)}$$

Finally retrieve the even and odd samples as

$$x_{2l} = s_l^{(0)}$$

$$x_{2l+1} = d_l^{(0)}$$

The following theorem holds:

Every wavelet or subband transform with finite filters can be obtained as the Lazy wavelet followed by a finite number of prediction and update steps plus one scaling operation.

D2 LIFTING AND INTEGER WAVELET TRANSFORMS

Since it is possible to write every wavelet transform using lifting, it follows that an integer version of *every* wavelet transform can be built. To achieve this, in each lifting step the result of the filtering operation can be rounded right before the adding or subtracting operation. An integer prediction step thus becomes:

$$d_l^{(i)} = d_l^{(i-1)} - \left\lfloor \sum_k p_k^{(i)} s_{l-k}^{(i-1)} + 1/2 \right\rfloor,$$

while an integer update step is given by

$$s_l^{(i)} = s_l^{(i-1)} - \left\lfloor \sum_k u_k^{(i)} d_{l-k}^{(i)} + 1/2 \right\rfloor.$$

This obviously results in an integer-to-integer transform. Because it is written using lifting steps, it is invertible, and the inverse again immediately follows by flipping the signs and reversing the operations.

This leads to the following pseudo-code implementation of an invertible integer wavelet transform using lifting:

$$\begin{aligned}
&s_l^{(0)} = s_{2l} \\
&d_l^{(0)} = s_{2l+1} \\
&\textit{for } i = 1:(1):M \\
&\quad \forall l : d_l^{(i)} = d_l^{(i-1)} - \left\lfloor \sum_k p_k^{(i)} s_{l-k}^{(i-1)} + 1/2 \right\rfloor \\
&\quad \forall l : s_l^{(i)} = s_l^{(i-1)} - \left\lfloor \sum_k u_k^{(i)} d_{l-k}^{(i)} + 1/2 \right\rfloor \\
&\textit{end}
\end{aligned} \tag{D-1}$$

The inverse transform is given by:

$$\begin{aligned}
&\textit{for } i = M:(-1):1 \\
&\quad \forall l : s_l^{(i-1)} = s_l^{(i)} + \left\lfloor \sum_k u_k^{(i)} d_{l-k}^{(i)} + 1/2 \right\rfloor \\
&\quad \forall l : d_l^{(i-1)} = d_l^{(i)} + \left\lfloor \sum_k p_k^{(i)} s_{l-k}^{(i-1)} + 1/2 \right\rfloor \\
&\textit{end} \\
&s_{2l+1} = d_l^{(0)} \\
&s_{2l} = s_l^{(0)}
\end{aligned} \tag{D-2}$$

D3 EXAMPLE: 5/3 INTEGER WAVELET

Though not part of the Recommendation, the 5/3 Integer DWT is useful to demonstrate the Lifting Scheme.

The analysis low- and high-pass 5/3 wavelet filters are described in reference [12] as:

$$\{h_{-2}, h_{-1}, h_0, h_1, h_2\} = \left\{-\frac{1}{8}, \frac{1}{4}, \frac{3}{4}, \frac{1}{4}, -\frac{1}{8}\right\};$$
$$\{g_{-1}, g_0, g_1\} = \left\{-\frac{1}{4}, \frac{1}{2}, -\frac{1}{4}\right\}$$

These filters are normalized to have unit gain at $\omega = 0$ and $\omega = \pi$ in the z-domain. Factorization of this filter bank gives the lifting steps for this filter:

$$\{p^1_{-1}, p^1_1\} = \left\{-\frac{1}{2}, -\frac{1}{2}\right\}$$
$$\{u^1_{-1}, u^1_1\} = \left\{-\frac{1}{4}, -\frac{1}{4}\right\}$$

with scaling factors $K_0 = 1$ and $K_1 = \frac{1}{2}$, for the low pass and high pass value, respectively. The Lifting Scheme for the forward direction is formally given by

$$s^{(0)}_l = s_{2l}$$
$$d^{(0)}_l = s_{2l+1}$$
$$\forall l : d^{(1)}_l = d^{(0)}_l - \left\lfloor \frac{1}{2}\left(s^{(0)}_{l-0} + s^{(0)}_{l+1}\right) + 1/2 \right\rfloor$$
$$\forall l : s^{(1)}_l = s^{(0)}_l - \left\lfloor \frac{-1}{4}\left(d^{(1)}_{l-1} + d^{(1)}_{l-0}\right) + 1/2 \right\rfloor$$

The following figures illustrate the Lifting Scheme for the 5/3 Integer DWT in an application to a one-dimensional signal.

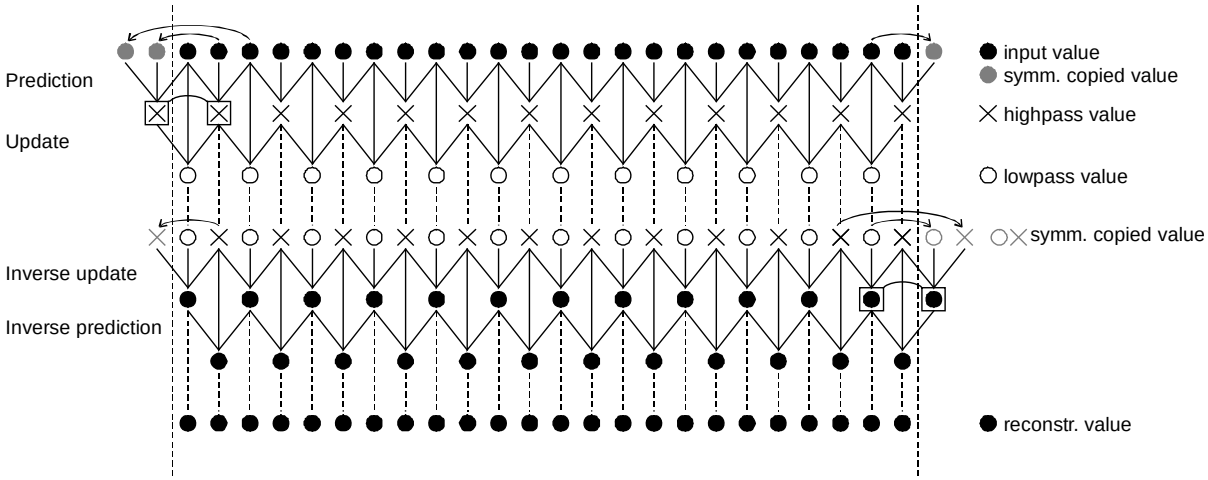


Figure D-3: Schematic Layout of the 5/3 Lifting Scheme, Showing the Prediction, Update, and Symmetrical Copy Operations

Forward transform

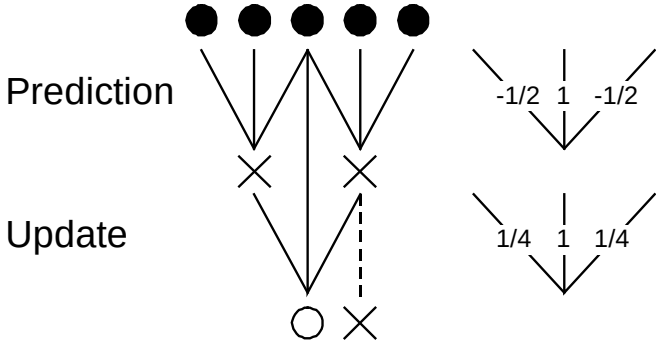


Figure D-4: Filter Coefficients for the Forward Lifting Process Using the 5/3 Filter

Backward transform

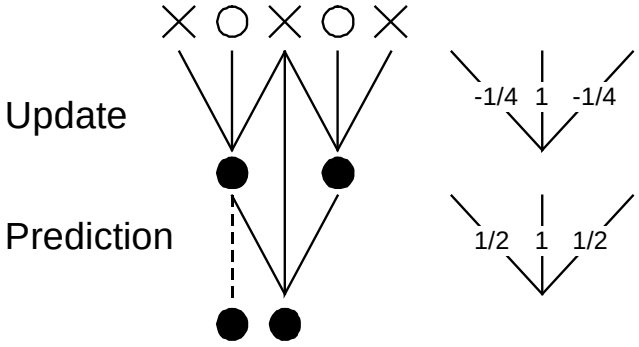


Figure D-5: Filter Coefficients for the Inverse Lifting Process Using the 5/3 Filter

The Lifting Scheme for the backward direction is formally given by

$$\forall l : s_l^{(0)} = s_l^{(1)} + \left\lfloor \frac{-1}{4}\left(d_{l-1}^{(1)} + d_{l-0}^{(1)}\right) + 1/2 \right\rfloor$$

$$\forall l : d_l^{(0)} = d_l^{(1)} + \left\lfloor \frac{1}{2}\left(s_{l-0}^{(0)} + s_{l+1}^{(0)}\right) + 1/2 \right\rfloor$$

$$s_{2l+1} = d_l^{(0)}$$

$$s_{2l} = s_l^{(0)}$$

D4 EXAMPLE: 9/7M WAVELET FILTER

The 9/7M wavelet filter requires one lifting step; i.e., there is only one prediction-update pair.

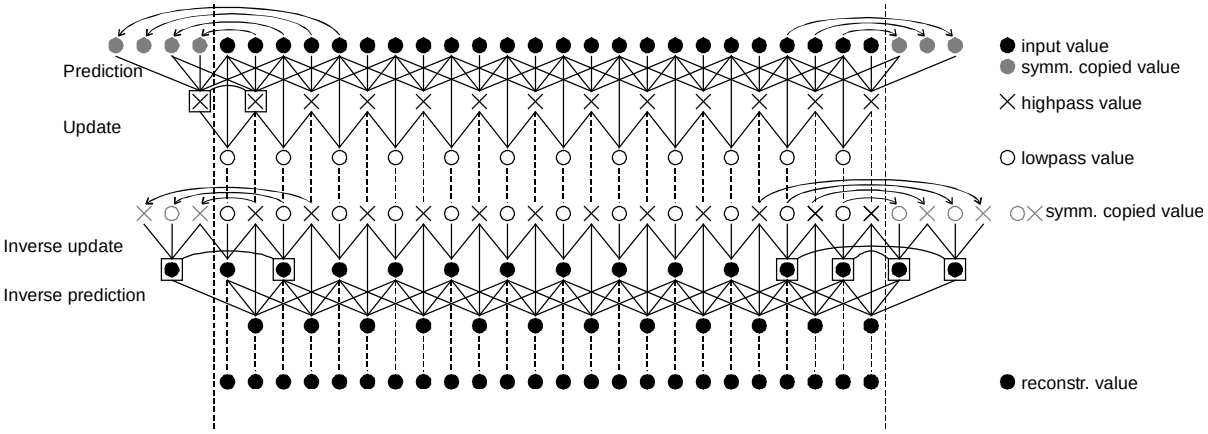


Figure D-6: Schematic Layout of the 9/7 Lifting Scheme, Showing the Prediction, Update, and Symmetrical Copy Operations

Forward transform

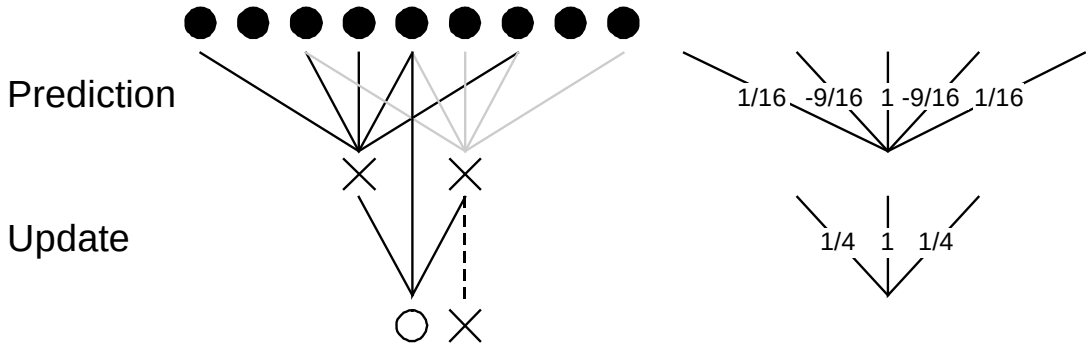


Figure D-7: Filter Coefficients for the Forward Lifting Process Using the 9/7M Filter

The following formulas are the consecutive filter operations to perform the forward 9/7M wavelet transform using the lifting scheme. They are obtained by substituting the correct filter coefficients in equation (D-1):

$$s_l^{(0)} = s_{2l}$$
$$d_l^{(0)} = s_{2l+1}$$
$$\forall l : d_l^{(1)} = d_l^{(0)} - \left\lfloor \frac{-1}{16}\left(s_{l-1}^{(0)} + s_{l+2}^{(0)}\right) + \frac{9}{16}\left(s_{l-0}^{(0)} + s_{l+1}^{(0)}\right) + 1/2 \right\rfloor$$
$$\forall l : s_{1,l}^{(1)} = s_l^{(0)} - \left\lfloor \frac{-1}{4}\left(d_{l-1}^{(1)} + d_l^{(1)}\right) + 1/2 \right\rfloor$$

Backward transform

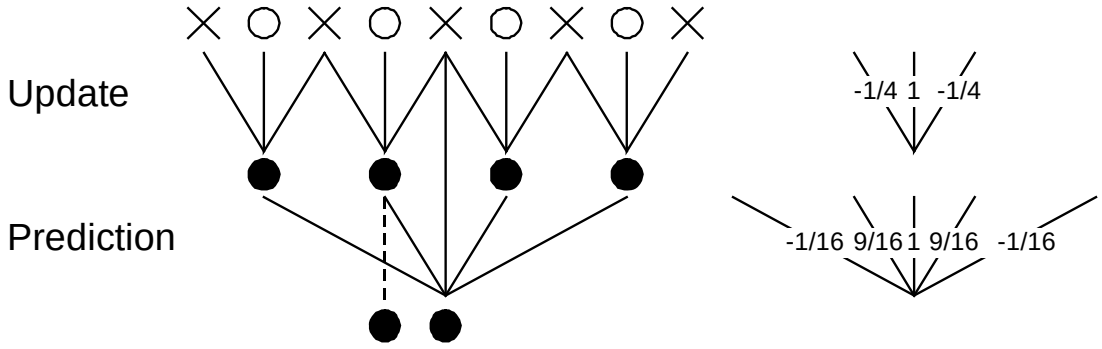


Figure D-8: Filter Coefficients for the Inverse Lifting Process Using the 9/7M Filter

The following formulas are the consecutive filter operations to perform the backward 9/7M wavelet transform using the lifting scheme. They are obtained by substituting the correct filter coefficients in equation (D-2):

$$\forall l : s_l^{(0)} = s_l^{(1)} + \left\lfloor \frac{-1}{4}\left(d_{l-1}^{(1)} + d_l^{(1)}\right) + 1/2 \right\rfloor$$
$$\forall l : d_l^{(0)} = d_l^{(1)} + \left\lfloor \frac{-1}{16}\left(s_{l-1}^{(0)} + s_{l+2}^{(0)}\right) + \frac{9}{16}\left(s_l^{(0)} + s_{l+1}^{(0)}\right) + 1/2 \right\rfloor$$
$$s_{2l+1} = d_l^{(0)}$$
$$s_{2l} = s_l^{(0)}$$

ANNEX E

DWT WEIGHT ANALYSIS

E1 INTRODUCTION

This annex derives subband weight factors intended to optimize rate vs. MSE distortion performance for linear DWTs at high bit rates. The argument is primarily based on equation (E-12) below which gives an approximate expression for image distortion in terms of subband weight factors. Under the high bit-rate assumption, subband weight factors that optimize rate-distortion effectiveness result in each wavelet coefficient's making the same contribution to expected MSE distortion. The weighting rule (E-13, E-14) is therefore derived from (E-12) by simply analyzing the individual contributions.

The analysis assumes a linear DWT, but in fact the integer DWT is only approximately linear because of the round-off operations involved. Consequently, the default subband weights for the integer DWT have been obtained empirically, and the Recommendation supports optional user-defined weights.

E2 DEPENDENCE OF IMAGE DISTORTION ON QUANTIZATION FACTORS

Let $\{x_i\}$ denote an image signal and $\{y_k\}$ denote the DWT coefficients obtained by applying a 2-d, multi-level DWT to the image. For clarity, image pixels $\{x_i\}$ and wavelet coefficients $\{y_k\}$ use a single index. The DWT is called isometric if for any such signal pair

$$\sum_i x_i^2 = \sum_k y_k^2 . \tag{E-1}$$

When the DWT is not isometric, weighting of wavelet coefficients is necessary before applying the BPE.

Let

$$x_i = \sum_k T_{ik} y_k \tag{E-2}$$

denote the *inverse* of the DWT. For each k, let q_k denote the quantization step size applied to the coefficient y_k in the course of encoding the DWT coefficient data. For bit-plane encoding, this quantization factor depends only on the current bit plane and is independent of any subband or coefficient index k, e.g., $q_k = 2^b$ for the b^{th} bit plane, for all k. Despite the

present scope of using only uniform quantization factors $q_k = q$, for reasons of clarity and generality the index k is retained.[9]

Let $\{\hat{y}_k\}$ denote the reconstructed wavelet coefficients (see 4.4) after quantization:

$$\hat{y}_k = \left(\left\lfloor \frac{y_k}{q_k} \right\rfloor + \frac{1}{2} \right) \cdot q_k \tag{E-3}$$

and let $\{\hat{x}_i\}$ denote the corresponding reconstructed image:

$$\hat{x}_i = \sum_k T_{ik} \hat{y}_k \tag{E-4}$$

The following argument is a statistical one, i.e., is valid on the average for an ensemble of images.

First, the variance of the *quantization error* of any particular coefficient y_k may be expressed in terms of the applied quantization step size q_k as

$$\left\langle (y_k - \hat{y}_k)^2 \right\rangle = \left\langle q_k^2 \right\rangle = \frac{1}{q_k} \int_{-\frac{q_k}{2}}^{\frac{q_k}{2}} \rho^2 d\rho = \frac{q_k^2}{12} \tag{E-5}$$

assuming a uniform distribution of the quantization error. This assumption is reasonable if the standard deviation of the quantized data is significantly larger than the quantization step size. This is the case for $q_k \to 0$, i.e., at high bit rates. The analysis continues under this assumption.

Next, the expected MSE image distortion

$$\left\langle E^2 \right\rangle = \sum_i \left\langle (x_i - \hat{x}_i)^2 \right\rangle \tag{E-6}$$

of the de-compressed image is quantified in terms of the quantization factors used in the wavelet domain. By definition (E-2),

$$\left\langle E^2 \right\rangle = \sum_i \left\langle \left[\sum_k \left[T_{ik} (y_k - \hat{y}_k) \right] \right]^2 \right\rangle . \tag{E-7}$$

[9] The bit-plane coding employed in the Recommendation effectively produces a dead-zone quantizer for each DWT coefficient. For simplicity of analysis, however, for each *k*, DWT coefficient y_k is modeled as being uniformly quantized by a quantizer with step size q_k.

It is a reasonable approximation in this context to assume that the quantization errors of the individual wavelet coefficients are decorrelated. The expected distortion is therefore given by

$$\left\langle E^2 \right\rangle = \sum_{i,k} T_{ik}^2 \left\langle \left(y_k - \hat{y}_k\right)^2 \right\rangle = \sum_k \left(\sum_i T_{ik}^2 \right) \left\langle \left(y_k - \hat{y}_k\right)^2 \right\rangle . \tag{E-8}$$

Introducing factors

$$\alpha_k = \sqrt{\sum_i T_{ik}^2} \tag{E-9}$$

equation (E-8) becomes

$$\left\langle E^2 \right\rangle = \sum_k \alpha_k^2 \left\langle \left(y_k - \hat{y}_k\right)^2 \right\rangle = \frac{1}{12} \sum_k \alpha_k^2 \cdot q_k^2 \ . \tag{E-10}$$

Equation (E-10) shows that, under the above assumptions, MSE image distortion depends only on the transform and wavelet domain quantization and not on the data itself.

E3 DISTORTION WITH SUBBAND WEIGHT FACTORS

Let z_k denote the weighted version of DWT coefficient y_k when the weight factor is ω_k:

$$y_k \mapsto \omega_k y_k = z_k \ (\text{for all } k) \tag{E-11}$$

then coding is applied to the weighted wavelet coefficients $\{z_k\}$. The image distortion now resulting from quantizing the weighted coefficients $\{z_k\}$ instead of the original ones $\{y_k\}$, again using quantization factors q_k, is given by

$$\left\langle E^2 \right\rangle = \sum_k \alpha_k^2 \left\langle \left(y_k - \hat{y}_k\right)^2 \right\rangle = \sum_k \frac{\alpha_k^2}{\omega_k^2} \left\langle \left(z_k - \hat{z}_k\right)^2 \right\rangle = \frac{1}{12} \sum_k \left(\frac{\alpha_k}{\omega_k} \right)^2 q_k^2 \ . \tag{E-12}$$

In equation (E-12), each factor $\left(\frac{\alpha_k}{\omega_k}\right)^2$ weights how the size of the quantization factor of the k^{th} DWT coefficient contributes to the expected MSE distortion. If any particular coefficient contributes significantly more to image distortion than the others, it intuitively suggests that a smaller quantization factor should be selected for this coefficient (i.e., some extra bit rate should be allotted to that coefficient) in order to mitigate this effect; to balance this, a coefficient which contributes less could be quantized more coarsely (i.e., the extra bit rate should be taken from this coefficient). Indeed it can be shown analytically under certain assumptions that if all coefficients contribute uniformly, the entropy of the so-quantized wavelet data is minimized. The subband weighting rule for all k is therefore given by

$$\left(\frac{\alpha_k}{\omega_k}\right) = 1, \tag{E-13}$$

or

$$\omega_k = \alpha_k = \sqrt{\sum_i T_{ik}^2}\,. \tag{E-14}$$

If the DWT is isometric, then $\alpha_k = 1$ for all k and all subbands are weighted equally. Otherwise, subband weights may be computed using equation (E-14).

In this case, image distortion in terms of quantization factors applied is given by

$$\left\langle E^2 \right\rangle = \frac{1}{12}\sum_k q_k^2\,. \tag{E-15}$$

Equation (E-15) means that each coefficient, if truncated at the same bit-plane level, contributes equally to the overall MSE distortion on average: weighting in accordance with (E-15) means that all coefficients are equally significant with respect to coding. This property is implicitly assumed in the BPE coding scheme.

E4 COMPUTATION OF ANALYTICAL SUBBAND WEIGHTS

Weights have been computed presently for some transforms following the analytical procedure described above. As shown in E5, for the non-linear integer transforms, these analytically derived weights may be outperformed by other sets of subband weight factors.

Table E-1: Analytically Derived Weighting Factors for 9/7 Float DWT

subband	HL_1	LH_1	HH_1	HL_2	LH_2	HH_2	HL_3	LH_3	HH_3	LL_3
α_k (theoretical values)	0.97	0.97	1	0.96	0.96	0.93	1.01	1.01	1.00	1.01
β_k (closest power of 2)	1	1	1	1	1	1	1	1	1	1

Table E-2: Analytically Derived Weights for 9/7M Integer DWT

subband	HL_1	LH_1	HH_1	HL_2	LH_2	HH_2	HL_3	LH_3	HH_3	LL_3
α_k (theoretical values)	1.56	1.56	1	2.63	2.63	1.45	5.15	5.15	2.79	9.52
β_k (closest power of 2)	2	2	1	2	2	1	4	4	2	8

Table E-3: Analytically Derived Weights for 9/7F Integer DWT

subband	HL_1	LH_1	HH_1	HL_2	LH_2	HH_2	HL_3	LH_3	HH_3	LL_3
α_k (theoretical values)	1.28	1.28	1	1.68	1.68	1.23	2.32	2.32	1.74	3.08
β_k (closest power of 2)	1	1	1	2	2	1	2	2	2	4

Table E-4: Analytically Derived Weights for 5/3 Integer DWT

subband	HL_1	LH_1	HH_1	HL_2	LH_2	HH_2	HL_3	LH_3	HH_3	LL_3
α_k (theoretical values)	1.44	1.44	1	2.22	2.22	1.28	4.06	4.06	2.21	7.48
β_k (closest power of 2)	1	1	1	2	2	1	4	4	2	8

E5 EMPIRICAL RESULTS

This subsection presents empirical results of MSE distortion as a function of bit rates under different choices of DWT and subband weight factors. The results serve two purposes, first, to compare two different choices of subband weight, and second to compare the 9/7F and 9/7M DWTs.

Figure 6-3 shows the performance results averaged over an earlier set of test images for four compression rates (0.25, 0.5, 1.0, and 2.0 bits/pixel). Table E-5 exhibits the results of the underlying individual trials, taking into consideration four different scenarios:

Scenario 3: 9/7M filter with weights 221 221 442 8 (analytically derived weights)

Scenario 4: 9/7M filter with weights 221 442 884 8

Scenario 6: 9/7F filter with weights 111 221 222 4 (analytically derived weights)

Scenario 7: 9/7F filter with weights 111 111 111 1 (equal weights)

The results demonstrate that the choice of the weight factors affects compression effectiveness.

Scenario 4 gives the best average performance results for the 9/7M filter, and scenario 7, for the 9/7F filter. The set of possible weight factors has not been exhaustively evaluated, and there may exist other choices of weight factors that offer better performance.

The maximum absolute difference in PSNR occurring in a single trial with the 9/7M is 0.96 dB, in favor of scenario 4, which also performs best on average over all images. The maximum absolute difference occurring in a single trial with the 9/7F is 0.83 dB, in favor of scenario 6, which on the other hand performs worst on average over all images. This indicates a significant dependence on image content; no single set of weights will perform optimally for all possible images. For instance, scenario 6 outperforms scenario 7 at all bit rates (0.25, 0.5, 1.0, and 2.0 bits/pixel) for the *foc* and *wfpc* images, even though scenario 7 outperformed scenario 6 on the average.

No choice of DWT and weight factors was found to be better in all cases. However, the maximum difference in favor of the 9/7M is 4.52 dB, and the maximum difference in favor of the 9/7F DWT is 0.37dB. This suggests that even when the 9/7F DWT performs better than the 9/7M DWT, the differences tend to be small. On the average, the 9/7M performs 0.36dB better than the 9/7F DWT (45.61dB and 45.25dB, respectively).

Based on these conclusions, the 9/7M filter was selected, as discussed in 6.4, in combination with the set of weights:

221 442 884 8

Table E-5: PSNR Obtained Using Different Configuration on an Earlier CCSDS Data Set

	97M 221 221 442 8	9/7M 221 442 884 8	97F 111 221 222 4	97F 111 111 111 1
scenario	3	4	6	7
average	45.54	45.61	45.14	45.25
	PSNR	PSNR	PSNR	PSNR
marstest.raw (2 bits/pixel)	39.80	39.76	38.72	39.00
marstest.raw (1 bits/pixel)	34.01	33.88	33.57	33.89
marstest.raw (0.5 bits/pixel)	29.93	30.00	29.96	29.99
marstest.raw (0.25 bits/pixel)	26.69	26.92	26.81	26.98
spot-la_b1.raw	39.26	39.29	38.68	38.68
spot-la_b1.raw	34.38	34.52	34.44	34.42
spot-la_b1.raw	31.31	31.52	31.59	31.43
spot-la_b1.raw	29.21	29.25	29.38	29.47
spot-la_b2.raw	39.31	39.33	38.70	38.68
spot-la_b2.raw	34.41	34.56	34.46	34.45
spot-la_b2.raw	31.40	31.56	31.61	31.47
spot-la_b2.raw	29.20	29.24	29.38	29.50
spot-la_b3.raw	40.20	40.14	39.24	39.28
spot-la_b3.raw	35.34	35.42	35.23	35.25
spot-la_b3.raw	32.36	32.47	32.46	32.42
spot-la_b3.raw	30.07	30.19	30.22	30.31
spot-panchromatic	43.02	43.08	41.06	41.30
spot-panchromatic	37.87	37.75	37.13	37.36
spot-panchromatic	34.38	34.46	34.23	34.37
spot-panchromatic	31.80	31.92	31.85	31.86
forest_2kb1.dat	55.32	55.32	53.71	53.79
forest_2kb1.dat	48.47	48.04	47.54	48.06
forest_2kb1.dat	42.92	42.74	42.62	42.88
forest_2kb1.dat	38.77	38.85	38.91	39.01
forest_2kb4.dat	56.68	57.08	54.91	54.83
forest_2kb4.dat	50.85	50.45	49.66	50.23
forest_2kb4.dat	45.48	45.30	45.03	45.42
forest_2kb4.dat	41.04	41.09	40.95	41.13
ice_2kb1.dat	51.93	51.75	50.95	51.22
ice_2kb1.dat	45.94	45.83	45.56	45.92
ice_2kb1.dat	41.74	41.79	41.68	41.88
ice_2kb1.dat	38.53	38.71	38.67	38.76
ice_2kb4.dat	59.87	60.83	57.13	56.30
ice_2kb4.dat	55.25	55.08	53.60	53.76
ice_2kb4.dat	51.10	51.01	50.43	50.65
ice_2kb4.dat	47.46	47.53	47.29	47.35

	97M 221 221 442 8	9/7M 221 442 884 8	97F 111 221 222 4	97F 111 111 111 1
india_2kb1.dat	50.24	49.89	49.34	49.65
india_2kb1.dat	42.97	42.84	42.66	43.01
india_2kb1.dat	38.06	38.09	37.95	38.23
india_2kb1.dat	34.30	34.53	34.51	34.72
india_2kb4.dat	53.79	53.64	52.36	52.59
india_2kb4.dat	46.78	46.56	46.14	46.53
india_2kb4.dat	41.59	41.58	41.38	41.67
india_2kb4.dat	37.51	37.66	37.55	37.77
north-atlantic_1kb1.raw	52.05	52.06	51.22	51.45
north-atlantic_1kb1.raw	46.43	46.30	45.84	46.32
north-atlantic_1kb1.raw	42.21	42.30	42.18	42.38
north-atlantic_1kb1.raw	39.05	39.36	39.20	39.35
north-atlantic_1kb4.raw	46.46	46.45	46.40	46.69
north-atlantic_1kb4.raw	41.09	41.18	41.00	41.33
north-atlantic_1kb4.raw	37.58	37.79	37.85	37.87
north-atlantic_1kb4.raw	35.15	35.32	35.45	35.53
ocean_2kb1.dat	50.42	50.11	49.43	49.73
ocean_2kb1.dat	43.77	43.65	43.53	43.76
ocean_2kb1.dat	39.26	39.31	39.29	39.42
ocean_2kb1.dat	35.80	35.95	35.97	36.09
ocean_2kb4.dat	56.71	56.76	54.58	54.38
ocean_2kb4.dat	50.73	50.63	49.96	50.18
ocean_2kb4.dat	46.13	46.11	45.79	46.01
ocean_2kb4.dat	42.06	42.16	41.99	42.15
foc.dat	70.10	70.42	68.79	68.38
foc.dat	65.96	66.07	65.29	65.23
foc.dat	63.61	63.86	63.47	63.32
foc.dat	62.37	62.56	62.33	62.09
solar.dat	54.35	53.98	53.75	54.07
solar.dat	48.49	48.51	48.41	48.58
solar.dat	44.65	44.87	45.01	45.04
solar.dat	41.73	41.76	42.05	42.13
sun_spot.dat	59.24	59.26	58.67	58.88
sun_spot.dat	54.89	55.37	55.13	54.91
sun_spot.dat	51.78	52.49	52.24	52.04
sun_spot.dat	48.33	48.64	48.39	48.59
wfpc.dat	69.19	69.91	67.65	67.06
wfpc.dat	66.55	66.91	65.46	65.18
wfpc.dat	65.03	65.45	64.62	64.38
wfpc.dat	63.99	64.18	63.56	63.32

	97M 221 221 442 8	9/7M 221 442 884 8	97F 111 221 222 4	97F 111 111 111 1
sar16bit.raw	58.01	57.72	57.64	57.91
sar16bit.raw	52.57	52.65	52.74	52.89
sar16bit.raw	49.47	49.64	49.83	49.88
sar16bit.raw	47.34	47.41	47.53	47.77

In comparing the analytically derived and empirically derived weights, the superior result is highlighted in green for the 9/7M DWT and in orange for the 9/7F DWT. In comparing the 9/7M and 9/7F DWTs, using the empirically derived weights, superior results are indicated in red text.

Lossless performance of the 9/7M and 5/3 filters followed by BPE is presented in table E-6 along with the CCSDS 121.0-B-1 lossless compression scheme using 1-d and 2-d predictors.

Table E-6: Lossless Compression Results in Bits/Pixel

Image name	CCSDS 121.0-B.1	CCSDS 121.0-B.1	9/7M filter	5/3 filter
	1-d prediction	2-d Prediction	Analytical Weights	Optimal Weights
marstest.raw	5.23	5.27	4.82	4.97
spot-la_b1.raw	5.35	5.08	4.95	4.98
spot-la_b2.raw	5.31	5.08	4.93	4.97
spot-la_b3.raw	5.14	4.92	4.76	4.80
spot-panchromatic	4.85	4.47	4.31	4.36
forest_2kb1.dat	4.65	4.4	4.20	4.24
forest_2kb4.dat	4.62	4.31	3.96	4.06
ice_2kb1.dat	5.44	5.19	4.82	4.91
ice_2kb4.dat	3.87	3.72	3.39	3.46
india_2kb1.dat	5.26	5.05	4.81	4.87
india_2kb4.dat	4.71	4.5	4.10	4.21
north-atlantic_1kb1.raw	5.21	4.98	4.77	4.86
north-atlantic_1kb4.raw	6.06	5.74	5.73	5.77
ocean_2kb1.dat	5.32	5.12	4.98	5.02
ocean_2kb4.dat	4.42	4.16	3.84	3.94
foc.dat	3.36	3.2	3.44	3.42
solar.dat	7.12	6.54	6.24	6.29
sun_spot.dat	6.63	6.2	5.80	5.90
wfpc.dat	4.05	3.79	3.80	3.79
sar16bit.raw	10.31	10.1	9.97	10.04

ANNEX F

GLOSSARY

The following terms are used to help explain operation concepts in this book. They are not part of the Recommendation.

BPE	Bit-Plane Encoder. The recommended processing algorithm used to encode DWT coefficient data.
DWT	Discrete Wavelet Transform. The recommended processing algorithm to transform image data to wavelet coefficient data.
9/7 DWT	A DWT using a nine-tap filter to obtain low-pass wavelet coefficients and a seven-tap filter to obtain high-pass wavelet coefficients. Two different specific 9/7 Discrete Wavelet Transforms are recommended: 9/7 Float DWT for lossy compression and 9/7 Integer DWT for lossless compression.
9/7 Float DWT	Three-level 9/7 DWT whose filter coefficients are specified real numbers. Implemented using floating-point arithmetic.
9/7 Integer DWT	Three-level 9/7 DWT whose filter coefficients are specified rational numbers. Implemented using integer arithmetic.
embedded	A descriptive property of compressed data that is structured so that earlier portions of the coded bit stream tend to make a larger improvement in overall reconstruction fidelity than later portions. This property allows a coded bit stream of lower bit rate to be obtained by simply truncating a compressed bit stream at higher bit rate.
rate-limited	Limited by the value of SegByteLimit, eliminating the quality limit (by setting DCStop=0, BitPlaneStop=0, and StageStop=3).
fixed-rate	Rate-limited compression with UseFill set to 1 so that padding bits are used when needed to obtain exactly SegByteLimits for each segment.
quality-limited	Limited by the values of DCStop, BitPlaneStop, and StageStop. Compression is said to be quality-limited when each segment has SegByteLimit set sufficiently high, and UseFill set to 0, so that the amount of compressed data in each segment is determined by the quality limit (i.e., the values of DCStop, BitPlaneStop, and StageStop).
full-frame	A full image frame is included in one single segment during compression, i.e., $S = \lceil w/8 \rceil \cdot \lceil h/8 \rceil$, where w and h are the width and height of the image, respectively.
segment	Bitstream of compressed code consisting of a data field headed by a segment header. The segment header is defined in subsection 4.2 of reference [1]. The data field contains the encoded bits from S consecutive blocks. S is a user-selected parameter such that $16 \leq S \leq 2^{20}$.
strip compression	Compression performed on a linear array of transformed pixels. When $S = \lceil w/8 \rceil$, each image segment loosely corresponds to a thin horizontal strip of the image, then *strip* compression is performed.