The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Standards

SPECTRAL PREPROCESSING TRANSFORM FOR MULTISPECTRAL AND HYPERSPECTRAL IMAGE COMPRESSION

DRAFT RECOMMENDED STANDARD

CCSDS 122.1-R-1

RED BOOK
March 2017

The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Standards

SPECTRAL PREPROCESSING TRANSFORM FOR MULTISPECTRAL AND HYPERSPECTRAL IMAGE COMPRESSION

DRAFT RECOMMENDED STANDARD

CCSDS 122.1-R-1

RED BOOK

March 2017

AUTHORITY

Issue:	Red Book, Issue 1
Date:	March 2017
Location:	Not Applicable

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF AUTHORITY:)

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorization of CCSDS documents is detailed in *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4), and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the e-mail address below.

This document is published and maintained by:

CCSDS Secretariat
National Aeronautics and Space Administration
Washington, DC, USA
E-mail: secretariat@mailman.ccsds.org

STATEMENT OF INTENT

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF INTENT:)

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of its members. The Committee meets periodically to address data systems problems that are common to all participants, and to formulate sound technical solutions to these problems. Inasmuch as participation in the CCSDS is completely voluntary, the results of Committee actions are termed **Recommended Standards** and are not considered binding on any Agency.

This **Recommended Standard** is issued by, and represents the consensus of, the CCSDS members. Endorsement of this **Recommendation** is entirely voluntary. Endorsement, however, indicates the following understandings:

- Whenever a member establishes a CCSDS-related **standard**, this **standard** will be in accord with the relevant **Recommended Standard**. Establishing such a **standard** does not preclude other provisions which a member may develop.
- Whenever a member establishes a CCSDS-related **standard**, that member will provide other CCSDS members with the following information:
  - The **standard** itself.
  - The anticipated date of initial operational capability.
  - The anticipated duration of operational service.
- Specific service arrangements shall be made via memoranda of agreement. Neither this **Recommended Standard** nor any ensuing **standard** is a substitute for a memorandum of agreement.

No later than five years from its date of issuance, this **Recommended Standard** will be reviewed by the CCSDS to determine whether it should: (1) remain in effect without change; (2) be changed to reflect the impact of new technologies, new requirements, or new directions; or (3) be retired or canceled.

In those instances when a new version of a **Recommended Standard** is issued, existing CCSDS-related member standards and implementations are not negated or deemed to be non-CCSDS compatible. It is the responsibility of each member to determine when such standards or implementations are to be modified. Each member is, however, strongly encouraged to direct planning for its new standards and implementations towards the later version of the Recommended Standard.

FOREWORD

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommended Standard is therefore subject to CCSDS document management and change control procedures, which are defined in the *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4). 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 sent to the CCSDS Secretariat at the e-mail 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.
- Canadian Space Agency (CSA)/Canada.
- Centre National d'Etudes Spatiales (CNES)/France.
- China National Space Administration (CNSA)/People's Republic of China.
- Deutsches Zentrum für Luft- und Raumfahrt (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.
- UK Space Agency/United Kingdom.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Belgian Federal Science Policy Office (BFSPO)/Belgium.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- China Satellite Launch and Tracking Control General, Beijing Institute of Tracking and Telecommunications Technology (CLTC/BITTT)/China.
- 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.
- Departamento de Ciência e Tecnologia Aeroespacial (DCTA)/Brazil.
- Electronics and Telecommunications Research Institute (ETRI)/Korea.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Geo-Informatics and Space Technology Development Agency (GISTDA)/Thailand.
- 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.
- Ministry of Communications (MOC)/Israel.
- National Institute of Information and Communications Technology (NICT)/Japan.
- National Oceanic and Atmospheric Administration (NOAA)/USA.
- National Space Agency of the Republic of Kazakhstan (NSARK)/Kazakhstan.
- National Space Organization (NSPO)/Chinese Taipei.
- Naval Center for Space Technology (NCST)/USA.
- Scientific and Technological Research Council of Turkey (TUBITAK)/Turkey.
- South African National Space Agency (SANSA)/Republic of South Africa.
- Space and Upper Atmosphere Research Commission (SUPARCO)/Pakistan.
- Swedish Space Corporation (SSC)/Sweden.
- Swiss Space Office (SSO)/Switzerland.
- United States Geological Survey (USGS)/USA.

PREFACE

This document is a draft CCSDS Recommended Standard. Its 'Red Book' status indicates that the CCSDS believes the document to be technically mature and has released it for formal review by appropriate technical organizations. As such, its technical contents are not stable, and several iterations of it may occur in response to comments received during the review process.

Implementers are cautioned **not** to fabricate any final equipment in accordance with this document's technical content.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 122.1-R-1	Spectral Preprocessing Transform for Multispectral and Hyperspectral Image Compression, Draft Recommended Standard, Issue 1	March 2017	Current draft

CONTENTS

CONTENTS (continued)

Section Page

Figure

CONTENTS (continued)

Figure Page

Table

1 INTRODUCTION

1.1 PURPOSE

The purpose of this document is to establish a Recommended Standard for a data compression algorithm applied to digital three-dimensional image data from payload instruments, such as multispectral and hyperspectral imagers, and to specify the compressed data format. Data compression is used to reduce the volume of digital data to achieve benefits in areas including, but not limited to,

a) reduction of transmission channel bandwidth;

b) reduction of the buffering and storage requirement;

c) reduction of data-transmission time at a given rate.

This document describes several preprocessing transforms that exploit similarities among the bands of multispectral and hyperspectral images. The result of any of these preprocessing transforms can then be efficiently compressed by a two-dimensional (2D) image encoder.

1.2 SCOPE

The characteristics of instrument data are specified only to the extent necessary to ensure multi-mission support capabilities. The specification does not attempt to quantify the relative bandwidth reduction, the merits of the approaches discussed, or the design requirements for encoders and associated decoders. Some performance information is included in reference [C1].

This Recommended Standard addresses compression of three-dimensional data from imaging spectrometers or atmospheric sounders; such data are often referred to as multi-, hyper-, or ultraspectral images. The standard is applicable to a wide range of space-borne digital data, where the requirement is for a scalable data reduction, including the option to use lossy compression, which allows some loss of fidelity in the process of data compression and decompression.

This Recommended Standard is formulated as a supplemental document that extends an existing Recommended Standard for two-dimensional image compression (reference [1]) to include support for three-dimensional image compression.

1.3 APPLICABILITY

This Recommended Standard applies to data compression applications of space missions anticipating packetized telemetry cross support. In addition, it serves as a guideline for the development of compatible CCSDS Agency standards in this field, based on good engineering practice. This Recommended Standard is only applicable as an extension to the Image Data Compression Recommended Standard (reference [1]).

1.4 RATIONALE

The concept and rationale for the lossy multispectral and hyperspectral data compression algorithm described herein may be found in reference [C1].

1.5 DOCUMENT STRUCTURE

This document is organized as follows:

a) Section 1 provides the purpose, scope, applicability, and rationale of this Recommended Standard and identifies the conventions and references used throughout the document. This section also describes how this document is organized. A brief description is provided for each section and annex so that the reader will have an idea of where information can be found in the document.

b) Section 2 provides an overview of the data compressor.

c) Section 3 specifies the upshift and downshift stages.

d) Section 4 specifies the spectral transforms that may be used in the compressor.

e) Section 5 specifies the 2D encoding stage and the format of a compressed image.

f) Annex A provides the Protocol Implementation Conformance Statement (PICS) Requirements List (RL) for implementations of this Recommended Standard.

g) Annex B discusses security, Space Assigned Numbers Authority (SANA), and patent considerations.

h) Annex C lists informative references.

i) Annex D provides tables of symbols used in this document.

j) Annex E expands abbreviations and acronyms used in this document.

k) Annex F provides transform weights necessary for rate allocation.

l) Annex G provides reference implementations for some of the mathematical operations described in this document.

1.6 NOMENCLATURE

1.6.1 NORMATIVE TEXT

The following conventions apply for the normative specifications in this Recommended Standard:

a) the words ‘shall’ and ‘must’ imply a binding and verifiable specification;

b) the word ‘should’ implies an optional, but desirable, specification;

c) the word ‘may’ implies an optional specification;

d) the words ‘is’, ‘are’, and ‘will’ imply statements of fact.

NOTE – These conventions do not imply constraints on diction in text that is clearly informative in nature.

1.6.2 INFORMATIVE TEXT

In the normative sections of this document (sections 3-5), informative text is set off from the normative specifications either in notes or under one of the following subsection headings:

– Overview;

– Background;

– Rationale;

– Discussion.

1.7 NOTATIONS, DEFINITIONS, AND CONVENTIONS

1.7.1 MATHEMATICAL NOTATION AND DEFINITIONS

1.7.1.1 Overview

The following definitions apply throughout this document.

1.7.1.2 Sequence

1.7.1.2.1 A sequence is an ordered list of quantities denoted by a comma separated list of values optionally delimited by curly brackets.

$$a, b, c \equiv \{a, b, c\} \tag{1}$$

Ellipsis points, ‘…’, may be used rather than explicitly listing each element in a sequence.

1.7.1.2.2 Given two integers a and b, the notation

$$a,\ldots,b \tag{2}$$

denotes the sequence, in ascending order, of the integer values from a to b, both included. When a is larger than b or a is equal to b, the sequence denotes respectively the empty sequence {} and the one-element sequence $\{a\}$. Thus, e.g., $\{0,\ldots,0\}$ is equal to $\{0\}$, and $\{0,\ldots,-1\}$ is equal to {}.

1.7.1.2.3 Given two integers a and b, the notation

$$A_a,\ldots,A_b \tag{3}$$

denotes a sequence of values A_i where i successively takes on each value in $a,\ldots,b$.

1.7.1.2.4 Given two integers a and b, and two constants c_1 and c_2, the notation

$$A_{c_1,a,c_2},\ldots,A_{c_1,b,c_2} \tag{4}$$

denotes a sequence of values A_{c_1,i,c_2} where i successively takes on each value in $a,\ldots,b$.

1.7.1.3 Rounding

For any real number x, the largest integer n such that $n \leq x$ is denoted by

$$n = \lfloor x \rfloor. \tag{5}$$

Similarly, for any real number x, the smallest integer n such that $n \geq x$ is denoted by

$$n = \lceil x \rceil. \tag{6}$$

Given a real number x, rounding to a nearest integer is denoted $\lceil x \rfloor$ and defined by

$$\lceil x \rfloor = \begin{cases} \lceil x - 0.5 \rceil & \text{if} \quad x > 0, \\ \lfloor x + 0.5 \rfloor & \text{otherwise.} \end{cases} \tag{7}$$

Thus, when x is equally close to two nearest integers, $\lceil x \rfloor$ selects the integer closest to zero.

1.7.1.4 Modulo

The modulus of an integer m with respect to a positive integer divisor n, denoted m mod n, is defined to be

$$m \bmod n = m - n\lfloor m / n \rfloor. \tag{8}$$

When it is stated that a value m is encoded modulo n, this means the number m mod n is encoded instead of m.

1.7.1.5 Logical Negation

The logical negation of a Boolean a is denoted $\neg a$. I.e., $\neg a$ is 'true' when a is 'false', and $\neg a$ is 'false' when a is 'true'.

1.7.2 IMAGE NOTATION AND DEFINITIONS

1.7.2.1 Overview

This subsection defines parameters and notation pertaining to an image and establishes limits on input images (e.g., size, bit depth, ...) that are supported by this Recommended Standard.

Quantities defined in this subsection are summarized in table D-1 of annex D.

1.7.2.2 Images in This Document

Four different images are defined in the encoding process:

- an *input image* to be encoded, denoted by the uppercase letter I;
- an *upshifted input image*, denoted by $I^{\uparrow}$, resulting from the application of the upshift stage to the input image (see 3.2.2);
- a *transformed image*, denoted by the uppercase letter T, resulting from the application of a spectral transform to the upshifted input image (see section 4);
- and a *downshifted transformed image*, denoted by $T^{\downarrow}$, resulting from the application of the downshift stage to the transformed image (see 3.2.3).

The properties defined in 1.7.2.3 apply to all four images (I, $I^{\uparrow}$, T, and $T^{\downarrow}$). For notational convenience, A is used to denote a generic image which could represent any of these four images.

1.7.2.3 Image Properties

1.7.2.3.1 Dimensions

1.7.2.3.1.1 An image A is a three-dimensional array of integer samples $A_{x,y,z}$, where the index z indicates the spectral band, and x and y are indices in the spatial dimensions.

NOTES

1 The term ‘band’ refers to a data slice along the z dimension of an image; however, it is only for the input image and upshifted input image where such a ‘band’ actually corresponds to a distinct spectral band, unless the identity transform (see 4.3) is selected for the spectral transform.

2 The spectral bands of the input image need not be arranged in order of increasing or decreasing wavelength. Rearranging the order of spectral bands can affect compression performance. This Recommended Standard does not address the tradeoffs associated with such a band reordering.

1.7.2.3.1.2 For I and $I^{\uparrow}$, indices x, y, and z take on integer values in the ranges $0 \le x < N_X$, $0 \le y < N_Y$, $0 \le z < N_Z$, where N_X, N_Y, and N_Z are the number of samples along each image dimension. For T and $T^{\downarrow}$, indices x and y take also on integer values in the ranges $0 \le x < N_X$, $0 \le y < N_Y$, while index z takes on values $0 \le z < N_{TZ}$.

NOTE – N_{TZ} may be less than N_Z.

1.7.2.3.1.3 Image sizes N_X and N_Y shall have a value of at least 17. Image size N_Z shall have a value of at least 1. Image sizes N_X and N_Z shall have a value of at most 2^{16}.

NOTE – There is no maximum value of N_Y.

1.7.2.3.1.4 Given an image A, the image corresponding to only band z of A is denoted A_z.

1.7.2.3.2 Bit Depth

1.7.2.3.2.1 The *bit depth* of an image A is the number of bits, n, used to represent each image sample.

1.7.2.3.2.2 An input image shall have a bit depth of at least 1 bit and at most 16 bits.

1.7.2.3.2.3 An image A may be *unsigned*, in which case all image samples satisfy

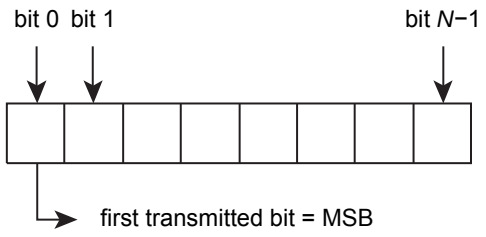
$$0 \le A_{x,y,z} \le 2^n - 1; \tag{9}$$

otherwise, A is *signed*, and all image samples satisfy

$$-2^{n-1} \le A_{x,y,z} \le 2^{n-1} - 1. \tag{10}$$

1.7.3 CONVENTIONS

In this document, the following convention is used to identify each bit in an N-bit word. The first bit in the word to be transmitted (shown in figures as the leftmost justified) is defined to be ‘bit 0’, the following bit is defined to be ‘bit 1’, and so on up to ‘bit $N-1$’. When the word is used to express an unsigned binary value (such as a counter), the Most Significant Bit (MSB), bit 0, shall correspond to the highest power of two, i.e., 2^{N-1}.



In accordance with modern data communications practice, spacecraft data words are often grouped into 8-bit 'words' that conform to the above convention. Throughout this Recommended Standard, the following nomenclature is used to describe this grouping:

8-bit word = 'Byte'.

1.8 REFERENCE

The following publication contains provisions that, through reference in this text, constitute provisions of this document. At the time of publication, the edition indicated was valid. All publications are subject to revision, and users of this document are encouraged to investigate the possibility of applying the most recent edition of the publication indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS publications.

[1] *Image Data Compression*. Issue 1.0. Proposed Draft Recommendation for Space Data System Standards (Proposed Pink Book), CCSDS 122.0-P-1.0. Washington, D.C.: CCSDS, July 2016.

2 OVERVIEW

2.1 GENERAL

This Recommended Standard defines a payload data compressor that has applicability to multispectral and hyperspectral imagers and sounders. This Recommended Standard does not attempt to explain the theory underlying the compression algorithm; that theory is partially addressed in references [C1] and [C2].

This standard extends the (two-dimensional) CCSDS Image Data Compression standard (reference [1]) by providing an effective method of encoding three-dimensional image data.

The input to the compressor is a three-dimensional image that has signed or unsigned integer sample values, as specified in 1.7.2. The compressed image output from the compressor is an encoded bitstream from which an exact or approximate reconstruction of the input image can be recovered (see subsection 3.4 of reference [C2]).

The compressor consists of two main functional parts, depicted in figure 2-1: a spectral transform, and a set of 2D encoders.

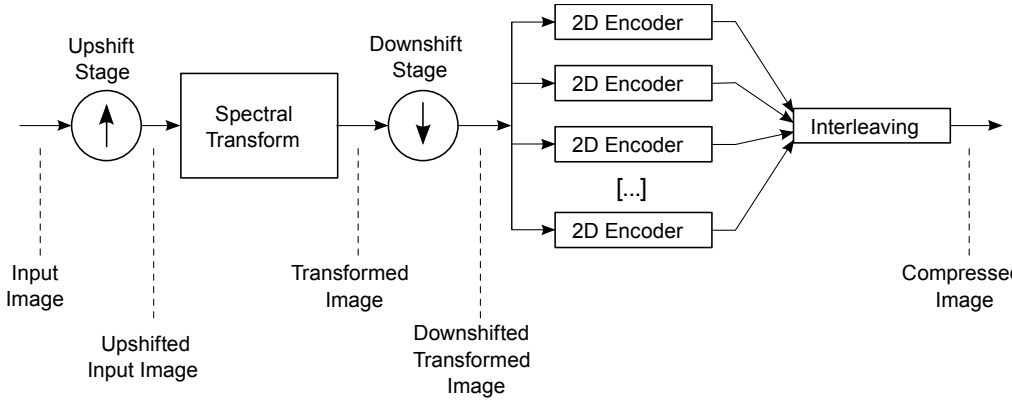


Figure 2-1: Compressor Schematic

The purpose of the spectral transform is to exploit the similarities between the spectral bands of an image, creating a transformed image that it is more efficiently compressed by the 2D encoders. Users may choose, according to their needs, to use one of several spectral transforms defined in section 4. Each transformed band is independently compressed by a 2D encoder; in practice, an implementation of the 2D encoder may be reused multiple times to accomplish this task.

Two additional minor functional parts are also included, with the purpose of adapting the bit depth before each of the two main functional parts. These two stages, named *upshift* stage and *downshift* stage, are defined in section 3. In addition to the two main functional parts and two minor functional parts, this standard also defines headers containing all the necessary

information to decode an image, and specifies the format for interleaving the various pieces of information that compose the compressed image.

While the purpose of this document is to specify the image *compression* process, non-trivial parts of the *decompression* process are also described following the corresponding parts of the compression process.

2.2 LOSSLESS AND LOSSY DATA COMPRESSION

There are two classes of image data compression methods: lossless and lossy. Under lossless compression, the original image can be reproduced exactly, while under lossy compression, quantization, or other approximations used in the compression process result in a reconstructed image that does not exactly match the original. The present Recommended Standard is capable of providing both lossless and lossy compression.

For a given image, the additional information required to achieve lossless compression will generally result in a larger compressed data volume than under lossy compression. Furthermore, variations in source image data content will lead to variations in compressed data volume. That is, the losslessly compressed image is of variable length.

Under the compression method described herein, compression parameters can be adjusted so that the distortion introduced in the reconstructed image by lossy compression is mostly homogeneous over the whole image. Moreover, lossy compression settings can be adjusted to provide either an approximate target reconstructed image fidelity with a compressed image of variable length, or a compressed image having a given fixed length with a variable image fidelity. I.e., the compression method can be used to meet either a target compressed data volume or an approximate target image quality.

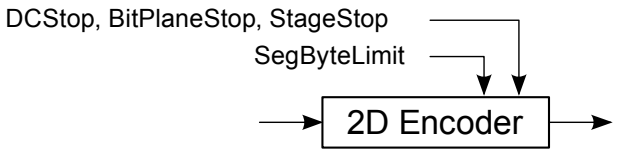


Figure 2-2: Compressor Quality and Rate Control

Under lossy compression for a given image, the tradeoff between overall reconstructed image fidelity and compressed data volume is controlled by adjusting the settings of each 2D encoder that determine the fidelity (via the DCStop, BitPlaneStop, and StageStop parameters defined in reference [1]) or compressed size (via the SegByteLimit parameter in reference [1]) of each transformed band. Simply using the same parameter settings for each 2D encoder, (e.g., using the SegByteLimit parameter to produce the same compressed data rate for each transformed band) generally yields poor performance in the tradeoff between image fidelity and data volume. Instead, it is better to use a *rate-allocation method* to assign a different compressed data rate to each 2D encoder. Further information is provided in annex F.

2.3 INPUT IMAGE QUALITY AND LOSSY DATA COMPRESSION

The compressor is designed to faithfully reproduce the original input image, i.e., minimize the error between the original and reconstructed images, while reducing compressed data volume. If the input image contains significant noise and/or artifacts, then the compressor will naturally attempt to faithfully reproduce such noise and artifacts.

For this reason, it is important that input images are relatively free from significant artifacts. For example, raw images from pushbroom hyperspectral imagers containing severe streaking artifacts may be poor candidates for compression if such artifacts are not removed first. Images with few artifacts or low noise, relative to the image signal, are generally good candidates for compression.

Also, other image artifacts, such as misregistration between spectral bands, are likely to increase the data rate required to reach a given image quality, as these artifacts hinder the ability of the encoder to exploit similarities between bands.

2.4 DATA TRANSMISSION

The effects of a single bit error can propagate to corrupt reconstructed data to the end of a compressed image (see reference [C1]). Therefore, measures should be taken to minimize the number of potential bit errors on the transmission link.

A user may choose to partition the output of an imaging instrument into smaller images that are separately compressed, e.g., to limit the impact of data loss or corruption on the communications channel, or to limit the maximum possible size of a compressed image. This Recommended Standard does not address such partitioning or the tradeoffs associated with selecting the size of images produced under such partitioning.

In addition, this Recommended Standard does not incorporate synchronization markers or other mechanisms to flag the headers of an image; it is assumed that the transport mechanism used for the delivery of the encoded bitstream will provide the ability to locate a subsequent image header in the event of a bit error.

In case the encoded bitstream is to be transmitted over a CCSDS space link, several protocols can be used to transfer a compressed image, including:

- Space Packet Protocol (reference [C3]);
- CCSDS File Delivery Protocol (CFDP) (reference [C4]);
- packet service or bitstream service as provided by the AOS Space Data Link Protocol (reference [C5]).

Limits on the maximum size data unit that can be transmitted may be imposed by the protocol used or by other practical implementation considerations. The user is expected to take such limits into account when using this Recommended Standard.

3 UPSHIFT AND DOWNSHIFT

3.1 OVERVIEW

This section defines the upshift stage applied before the spectral transform, and the downshift stage applied after the spectral transform and before the 2D encoders. The upshift stage scales an image by a power of two, effectively increasing the bit depth of the image by appending additional least-significant bits to its integer samples. The downshift stage does the opposite, effectively reducing bit depth by removing least-significant bits from the samples of an image.

The purpose of the upshift stage is to increase the bit depth of an input image up to the maximum bit depth supported by a given implementation of the transform stage. This increased bit depth generally provides higher lossy coding performance at high-fidelity compression.

The purpose of the downshift stage is to reduce the bit depth of the transformed image to match the bit depth supported by the 2D encoders. Note that lossless compression is generally not possible when the downshift stage removes any least significant bit.

Quantities defined in this section are summarized in table D-1 of annex D.

3.2 OPERATIONS

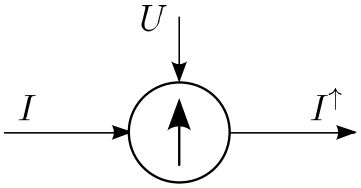


Figure 3-1: Block Diagram of the Upshift Stage

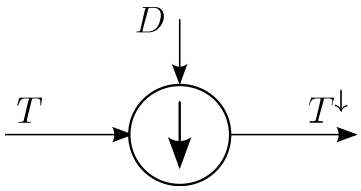


Figure 3-2: Block Diagram of the Downshift Stage

3.2.1 The upshift and downshift operations are controlled by user-specified parameters U and D, which shall be integers in the range from 0 to 15, with the difference $U - D \leq 5$.

3.2.2 Given input image I, the upshifted input image $I^{\uparrow}$ shall be formed by multiplying each sample of I by 2^U:

$$I^{\uparrow}_{x,y,z} = 2^U \cdot I_{x,y,z}\,. \tag{11}$$

NOTES

1 The upshift operation effectively appends U additional least-significant bits to each input image sample.

2 The upshift operation can be inverted by

$$I_{x,y,z} = \left\lfloor 2^{-U} \cdot I^{\uparrow}{}_{x,y,z} \right\rfloor. \tag{12}$$

3.2.3 Given a transformed image T, the downshifted transformed image $T^{\downarrow}$ shall be formed by multiplying each sample of T by 2^{-D} and rounding the resulting value to the nearest integer:

$$T^{\downarrow}_{x,y,z} = \left\lfloor 2^{-D} \cdot T_{x,y,z} \right\rfloor. \tag{13}$$

NOTES

1 The downshift operation is not equivalent to discarding the D least-significant bits of each sample of T.

2 The downshift operation effectively decreases the bit depth of each transformed image sample by at least $D - 1$ bits. In addition, for the particular case of $D = 1$, the bit depth is decreased by one bit.

3 When $D > 0$ the downshift operation is not invertible due to the loss of information in the rounding operation. In this case, the downshift operation can be *approximately* inverted by

$$T_{x,y,z} \approx 2^D \cdot T^{\downarrow}_{x,y,z}\,. \tag{14}$$

4 SPECTRAL TRANSFORM

4.1 OVERVIEW

This section specifies the four different spectral transforms supported by this Recommended Standard for the transform stage. Each transform produces a transformed image of integer samples, and may produce additional side information.

Computational resources required vary depending on the transform selected as well as the image dimensions. For example, under limited processing capabilities, computing a given transform might be feasible for a multispectral imager with a few bands but not for an ultraspectral imager with hundreds of bands.

Some of the transform definitions make use of intermediate quantities that are not integer valued. In general, replacing one of these intermediate quantities with a floating-point approximation may yield a transform output that is not accurate and thus not compliant with this Recommended Standard. Spectral transforms computed by two different compressors compliant with this Recommended Standard shall produce identical results, bit by bit, regardless of their respective underlying implementations. Users are advised to refer to annex G, which provides reference implementations of some of the mathematical formulations provided in this section.

Quantities defined in this section are summarized in table D-1 of annex D.

4.2 GENERAL

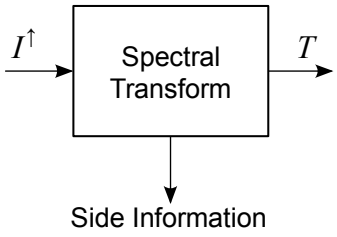


Figure 4-1: Block Diagram of a Spectral Transform

NOTE – A spectral transform converts an upshifted input image $I^{\uparrow}$ into a transformed image T (figure 4-1).

4.2.1 In addition to a transformed image T, a spectral transform can produce side information, which shall be stored in the compressed image headers as specified in 5.4.4.

4.2.2 For the spectral transform stage, users shall select one of the four transforms specified in 4.3, 4.4, 4.5, and 4.6.

4.3 IDENTITY TRANSFORM

4.3.1 OVERVIEW

The identity transform performs no operation on the upshifted input image. This transform is defined for the sake of providing a compressed data structure to encode a 3D image without requiring the implementation of a more complex transform stage.

4.3.2 DIMENSIONS

Under the identity transform, the transformed image shall have the same number of bands as the input image:

$$N_{TZ} = N_Z. \tag{15}$$

4.3.3 TRANSFORM APPLICATION

Under the identity transform, the transformed image T shall be equal to the upshifted input image $I^{\uparrow}$:

$$T = I^{\uparrow}. \tag{16}$$

4.3.4 DISCUSSION—SIDE INFORMATION

The identity transform does not produce any side information.

4.4 INTEGER WAVELET TRANSFORM

4.4.1 OVERVIEW

This subsection specifies the Integer Wavelet Transform (IWT), which has low computational complexity and provides a noticeable improvement in coding performance over the identity transform. The transform herein specified is known in the literature as the CDF 5/3 (reference [C6]) and is the same as the one used in the JPEG2000 standard (reference [C7]). While the number of levels of transform decomposition is variable in JPEG2000, in this standard it is fixed at five.

The IWT is a reversible transform that exactly recovers the original upshifted input image when inverted, provided that the transformed image does not suffer from any loss of information (either by the downshift stage or lossy compression by a 2D encoder). Subsection 4.4.3 defines the single-level IWT decomposition and its inverse; 4.4.4 describes how the five-level IWT decomposition is produced by successive applications of the single-level decomposition, and how to invert the five-level decomposition; and finally 4.4.5 describes how this one-dimensional five-level decomposition is applied to produce the transformed image.

4.4.2 DIMENSIONS

Under the IWT, the transformed image shall have the same number of bands as the input image:

$$N_{TZ} = N_Z. \tag{17}$$

4.4.3 SINGLE-LEVEL IWT DECOMPOSITION

4.4.3.1 Forward Calculation

4.4.3.1.1 The single-level IWT decomposition shall map an input sequence

$$X = \{x_0, x_1, \ldots, x_{N-1}\} \tag{18}$$

of size $N \geq 1$ to two output sequences, L and H, having length $p = \left\lceil \frac{N}{2} \right\rceil$ and $q = \left\lfloor \frac{N}{2} \right\rfloor$, respectively:

$$L = \{l_0, l_1, \ldots, l_{p-1}\}, \tag{19}$$

$$H = \{h_0, h_1, \ldots, h_{q-1}\}. \tag{20}$$

NOTE – L and H are the low-frequency and high-frequency *wavelet coefficients*, respectively, defined in 4.4.3.1.2 and 4.4.3.1.3.

4.4.3.1.2 If $N = 1$, then L shall contain the input value and H shall be an empty sequence, i.e.,

$$L = \{x_0\}, H = \{\}. \tag{21}$$

4.4.3.1.3 Otherwise (i.e., when $N > 1$), H is defined as

$$h_k = \begin{cases} x_{2k+1} - x_{2k}, & k = q-1 \text{ and } N \text{ is even} \\ x_{2k+1} - \left\lfloor \frac{1}{2}\left(x_{2k} + x_{2k+2}\right) \right\rfloor, & \text{otherwise.} \end{cases} \tag{22}$$

for $k = 0, \ldots, q-1$, and L is defined as

$$l_k = \begin{cases} x_0 + \left\lfloor \frac{1}{2}\left(h_0 + 1\right) \right\rfloor, & k = 0 \\ x_{2k} + \left\lfloor \frac{1}{2}\left(h_{k-1} + 1\right) \right\rfloor, & k = p-1 \text{ and } N \text{ is odd} \\ x_{2k} + \left\lfloor \frac{1}{4}\left(h_{k-1} + h_k + 2\right) \right\rfloor, & \text{otherwise.} \end{cases} \tag{23}$$

for $k = 0, \ldots, p-1$.

4.4.3.2 Inverse Calculation

4.4.3.2.1 Given $p = \lceil \frac{N}{2} \rceil$ low-frequency coefficients, L, and $q = \lfloor \frac{N}{2} \rfloor$ high-frequency coefficients, H, the inverse single-level IWT shall compute the corresponding original sequence $X = \{x_0, x_1, \ldots, x_{N-1}\}$ as follows.

4.4.3.2.2 If $N = 1$, the original value shall be

$$x_0 = l_0. \tag{24}$$

4.4.3.2.3 Otherwise (i.e., when $N > 1$), elements of X with even indices shall be computed as

$$x_{2k} = \begin{cases} l_0 - \lfloor \frac{1}{2}(h_0 + 1) \rfloor, & k = 0 \\ l_k - \lfloor \frac{1}{2}(h_{k-1} + 1) \rfloor, & k = p - 1 \text{ and } N \text{ is odd} \\ l_k - \lfloor \frac{1}{4}(h_{k-1} + h_k + 2) \rfloor, & \text{otherwise} \end{cases} \tag{25}$$

for $k = 0, \ldots, p - 1$, and the odd-indexed elements of X shall be computed as

$$x_{2k+1} = \begin{cases} h_k + x_{2k}, & k = q - 1 \text{ and } N \text{ is even} \\ h_k + \lfloor \frac{1}{2}(x_{2k} + x_{2k+2}) \rfloor, & \text{otherwise} \end{cases} \tag{26}$$

for $k = 0, \ldots, q - 1$.

4.4.4 FIVE-LEVEL IWT DECOMPOSITION

4.4.4.1 Forward Calculation

The five-level IWT decomposition shall be produced by five successive applications of the single-level IWT decomposition as follows:

a) Given an input sequence $X = \{x_0, x_1, \ldots, x_{N-1}\}$, L_1, H_1 denote the low- and high-frequency wavelet coefficients, respectively, obtained by applying the single-level decomposition to X. For $n > 1$, L_n, H_n denote the output of the single-level IWT decomposition when applied to input L_{n-1}.

b) The output of the five-level IWT decomposition, denoted $\text{IWT}_5(X)$, is defined as the concatenation of the following sequences:

$$L_5, H_5, H_4, H_3, H_2, H_1. \tag{27}$$

NOTE – Figure 4-2 illustrates the calculation of the five-level IWT decomposition via successive applications of the single-level IWT decomposition.

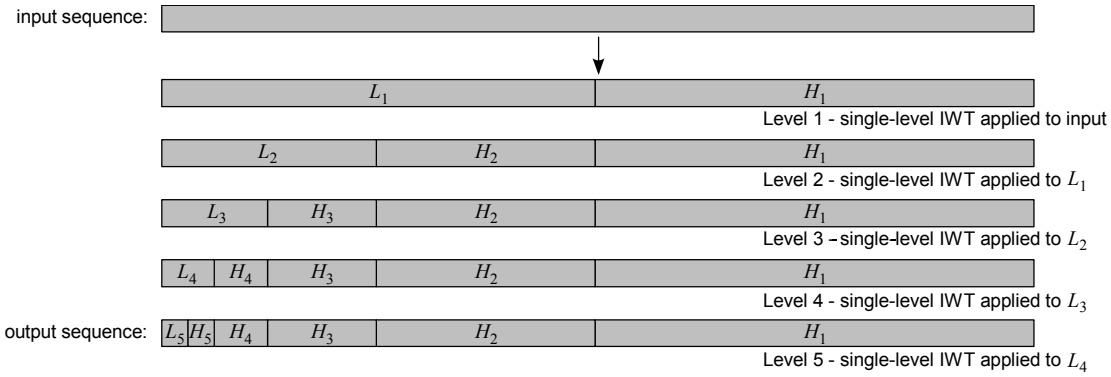


Figure 4-2: Five Levels of IWT

NOTE – For each $n = 2,\ldots,5$, the length of L_n is roughly half the length of L_{n-1}. For input sequences of length 16 or less, latter applications of the single-level IWT have no effect due to equation (21).

4.4.4.2 Inverse Calculation

Given five-level IWT decomposition output $Y = \mathrm{IWT}_5(X)$, the five-level IWT inverse, denoted $\mathrm{IWT}_5^{-1}(Y)$, shall be computed by inverting each successive single-level IWT decomposition in the reverse order in which it was applied.

4.4.5 TRANSFORM APPLICATION

The IWT shall produce a transformed image, T, by multiple independent applications of the five-level IWT decomposition to sequences of samples from the upshifted input image $I^{\uparrow}$:

$$\left\{T_{x,y,0},\ldots,T_{x,y,N_Z-1}\right\} = \mathrm{IWT}_5\left(\left\{I^{\uparrow}_{x,y,0},\ldots,I^{\uparrow}_{x,y,N_Z-1}\right\}\right) \tag{28}$$

for each $x = 0,\ldots,N_X - 1$ and each $y = 0,\ldots,N_Y - 1$.

NOTE – The calculation of the IWT for an upshifted input image is illustrated in figure 4-3.

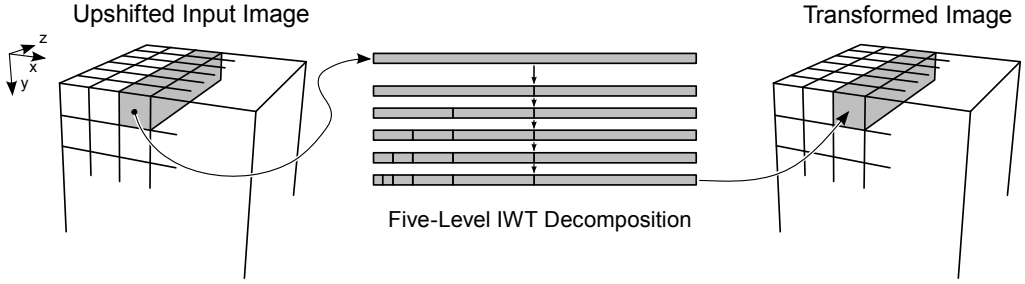


Figure 4-3: Application of the IWT

4.4.6 DISCUSSION—SIDE INFORMATION

The IWT does not produce any side information.

4.4.7 INVERSE TRANSFORM

The inverse IWT shall recover an upshifted input image, $I^{\uparrow}$, by multiple independent applications of the inverse five-level IWT decomposition to sequences of samples from a transformed image T:

$$\left\{I^{\uparrow}_{x,y,0},\ldots,I^{\uparrow}_{x,y,N_Z-1}\right\} = \mathrm{IWT}_5^{-1}\left(\left\{T_{x,y,0},\ldots,T_{x,y,N_Z-1}\right\}\right) \tag{29}$$

for each $x = 0,\ldots,N_X - 1$ and each $y = 0,\ldots,N_Y - 1$.

NOTE – This inverse is shown in figure 4-4.

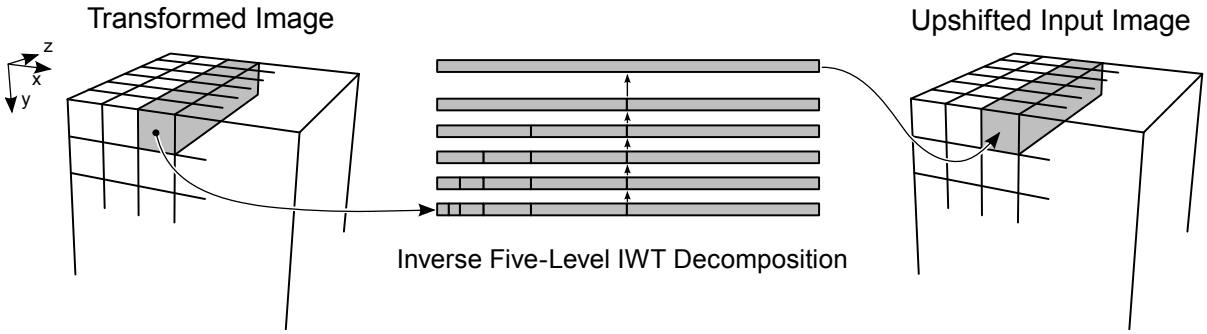


Figure 4-4: Inverse Application of the IWT

4.5 PAIRWISE ORTHOGONAL TRANSFORM

4.5.1 OVERVIEW

The Pairwise Orthogonal Transform (POT) is an approximation of the transform that provides perfect decorrelation—the Karhunen Loève Transform (KLT)—at a fraction of its computational cost. In general, the POT provides better coding performance than the IWT, but requires more computational resources and has a more complex implementation. The transform defined in this section is based on the POT as described in references [C8] and [C9].

The POT is composed of component operations, called mean-subtraction operations and pairwise operations, described in 4.5.3 and 4.5.4 respectively. These operations are combined into a *multilevel structure*, specified in 4.5.5, which is employed to produce the transformed image, as described in 4.5.6.

The POT is an exactly reversible transform that recovers the original upshifted input image when inverted, provided that the transformed image does not suffer from any loss of information (either by the downshift stage or lossy compression by a 2D encoder).

4.5.2 DIMENSIONS

Under the POT, the transformed image shall have the same number of bands as the input image:

$$N_{TZ} = N_Z. \tag{30}$$

4.5.3 MEAN-SUBTRACTION OPERATION

4.5.3.1 Overview

The intended purpose of the mean-subtraction operation is to ensure that the mean value of a sequence is approximately zero. The mean-subtraction operation calculates the nominal mean, m, of an input sequence, X, to produce an output sequence, Y, by subtracting m from each element of X. The mean-subtraction operation is depicted in figure 4-5.

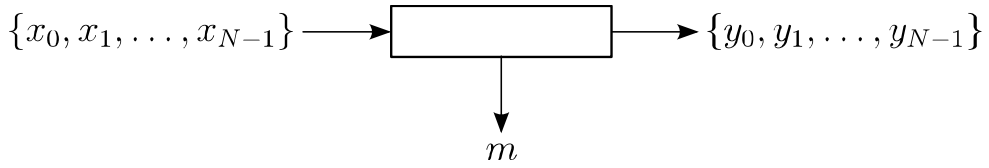


Figure 4-5: Mean-Subtraction Operation

4.5.3.2 Forward Calculation

4.5.3.2.1 Given an input sequence

$$X = \{x_0,\ldots,x_{N-1}\} \tag{31}$$

of length $N \geq 1$, the mean-subtraction operation shall compute an integer *nominal mean* value, m, and an output sequence

$$Y = \{y_0,\ldots,y_{N-1}\}. \tag{32}$$

4.5.3.2.2 The output sequence Y shall be obtained by subtracting m from each element of X:

$$y_i = x_i - m \qquad \text{for } i = 0,\ldots,N-1. \tag{33}$$

NOTE – The notation $Y = \mathcal{M}(X)$ is used to denote this calculation.

4.5.3.3 Nominal Mean

4.5.3.3.1 The nominal mean value m shall be an integer having the same bit depth as the input image I increased by U bits, where U is the user-specified upshift parameter defined in 3.2.1.

4.5.3.3.2 For a given input sequence, the value m should be an approximation of the arithmetic mean of X, and should be obtained as

$$m = \left\lfloor \frac{1}{N} \sum_{i=0}^{N-1} x_i \right\rfloor. \tag{34}$$

NOTE – As indicated by the use of the word 'should' above, selecting a different value of m is permissible, e.g., if an alternative calculation simplifies implementation complexity.

4.5.3.4 Inverse Calculation

The mean-subtraction operation may be reversed as

$$x_i = y_i + m \qquad \text{for } i = 0,\ldots,N-1. \tag{35}$$

NOTE – The value of m is stored as side information, as described in 4.5.8 and 5.4.4.3, and thus is available when the operation is reversed.

4.5.4 PAIRWISE OPERATION

4.5.4.1 Overview

The pairwise operation maps two length-N integer input sequences onto two length-N integer output sequences from which the input sequences can be recovered. Given input sequences that are approximately zero-mean, the pairwise operation is intended to produce output sequences that are approximately uncorrelated and zero-mean. As part of this operation, training parameters B and C are computed from the input sequences and output as side information that is needed to invert the operation. An additional input, *flip*, is used to optionally cause the operation to change the sign of all values in both output sequences. The pairwise operation is depicted in figure 4-6.

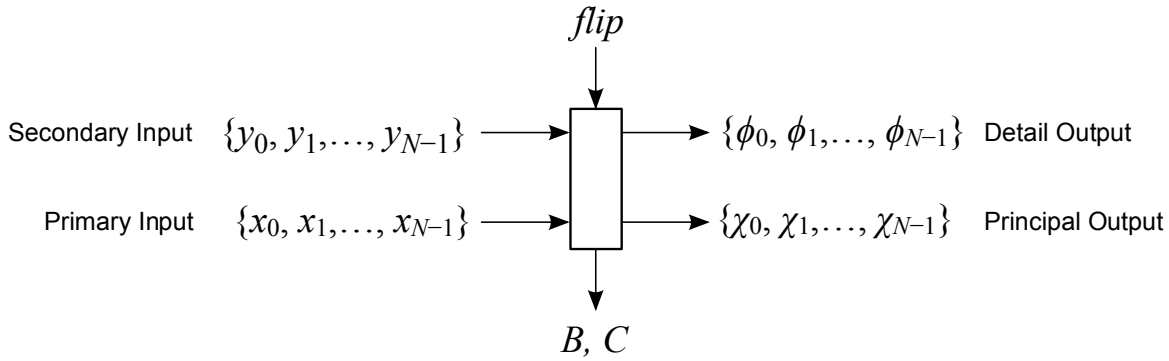


Figure 4-6: The Pairwise Operation

Subsection 4.5.4.2 enumerates the inputs and outputs of the pairwise operation. To perform the pairwise operation, first, training parameters B, C are computed from the input sequences, as described in 4.5.4.3. These training parameters are used to calculate weights $\tilde{w}_1$, $\tilde{w}_2$, and $\tilde{w}_3$ as described in 4.5.4.4. Finally, these weights are used to compute each pair of output values from each pair of input values, as described in 4.5.4.5. Subsection 4.5.4.6 provides the inverse of this calculation.

There are two types of pairwise operations: *balanced* and *unbalanced*. The two types differ in the way that the training parameters should be computed from the input, and in the way that the weights are calculated from the training parameters.

4.5.4.2 Inputs and Outputs

4.5.4.2.1 Inputs to the pairwise operation shall consist of two integer sequences, a *primary* input sequence $\{x_0,\ldots,x_{N-1}\}$ and a *secondary* input sequence $\{y_0,\ldots,y_{N-1}\}$, each of length $N > 0$, along with a Boolean *flip*, which can take a value of 'true' or 'false'.

4.5.4.2.2 The arithmetic precision of the pairwise operation is controlled by the user-specified parameter Ω, which shall be an integer in the range $9 \leq \Omega \leq 16$.

4.5.4.2.3 The value of Ω shall be fixed for a given image.

4.5.4.2.4 The output of the pairwise operation shall comprise two length-N integer output sequences, a *principal* output sequence $\{\chi_0,\ldots,\chi_{N-1}\}$ and a *detail* output sequence $\{\phi_0,\ldots,\phi_{N-1}\}$, calculated according to 4.5.4.5, along with a pair of integer *training parameters* B, C, described in 4.5.4.3.

4.5.4.3 Training Parameters

4.5.4.3.1 Given the pair of input sequences, scaled variances $\widetilde{\sigma_x^2}$, $\widetilde{\sigma_y^2}$ and covariance $\widetilde{\sigma_{x,y}}$ are defined as:

$$\begin{aligned}\widetilde{\sigma_x^2} &= N \cdot \sum_{i=0}^{N-1} {x_i}^2 - \left(\sum_{i=0}^{N-1} x_i \right)^2, \\ \widetilde{\sigma_y^2} &= N \cdot \sum_{i=0}^{N-1} {y_i}^2 - \left(\sum_{i=0}^{N-1} y_i \right)^2, \\ \widetilde{\sigma_{x,y}} &= N \cdot \sum_{i=0}^{N-1} \left(x_i \cdot y_i \right) - \left(\sum_{i=0}^{N-1} x_i \right) \cdot \left(\sum_{i=0}^{N-1} y_i \right).\end{aligned} \tag{36}$$

4.5.4.3.2 Training parameter B shall be equal to −1 or 1. Training parameter C shall be an integer in the range $-2^{\Omega-2} \le C \le 2^{\Omega-2} - 1$.

4.5.4.3.3 The training parameters B and C should be calculated from the values of $\widetilde{\sigma_x^2}$, $\widetilde{\sigma_y^2}$, $\widetilde{\sigma_{x,y}}$ as follows:

a) Training parameter B should be calculated as

$$B = \begin{cases} 1, & \text{if } \widetilde{\sigma_{x,y}} > 0, \\ -1, & \text{if } \widetilde{\sigma_{x,y}} \le 0. \end{cases} \tag{37}$$

b) For a balanced pairwise operation, training parameter C should be calculated as

$$C = \begin{cases} 2^{\Omega-2} - 1, & \text{if } \widetilde{\sigma_{\mathrm{x,y}}} = 0 \text{ and } \widetilde{\sigma_{\mathrm{x}}^2} \ge \widetilde{\sigma_{\mathrm{y}}^2} \\ -2^{\Omega-2}, & \text{if } \widetilde{\sigma_{\mathrm{x,y}}} = 0 \text{ and } \widetilde{\sigma_{\mathrm{x}}^2} < \widetilde{\sigma_{\mathrm{y}}^2} \\ \min\left(\max\left(\left\lfloor \frac{2^{\Omega-5} \cdot (\widetilde{\sigma_x^2} - \widetilde{\sigma_y^2})}{|\widetilde{\sigma_{x,y}}|} \right\rfloor, -2^{\Omega-2} \right), 2^{\Omega-2} - 1 \right), & \text{otherwise.} \end{cases} \tag{38}$$

c) and for an unbalanced pairwise operation, training parameter C should be calculated as

$$C = \begin{cases} 2^{\Omega-2} - 1, & \text{if } \widetilde{\sigma_{\mathrm{x,y}}} = 0 \text{ and } 2\widetilde{\sigma_{\mathrm{x}}^2} \ge \widetilde{\sigma_{\mathrm{y}}^2} \\ -2^{\Omega-2}, & \text{if } \widetilde{\sigma_{\mathrm{x,y}}} = 0 \text{ and } 2\widetilde{\sigma_{\mathrm{x}}^2} < \widetilde{\sigma_{\mathrm{y}}^2} \\ \min\left(\max\left(\left\lfloor \frac{2^{\Omega-4} \cdot (2\widetilde{\sigma_x^2} - \widetilde{\sigma_y^2})}{\left\lfloor 181 \cdot |\widetilde{\sigma_{x,y}}| / 2^7 \right\rfloor} \right\rfloor, -2^{\Omega-2} \right), 2^{\Omega-2} - 1 \right), & \text{otherwise.} \end{cases} \tag{39}$$

NOTE – Given a pair of approximately zero-mean input sequences, calculation of training parameters B and C via the equations above serves to produce a pair of approximately zero-mean and approximately decorrelated output sequences, which is the intended purpose of the pairwise operation. However, as indicated by the use of the word 'should' above, alternative methods of selecting the values of training parameters B and C are permissible, e.g., if an alternative calculation simplifies implementation complexity.

4.5.4.4 Weights

4.5.4.4.1 Calculation of the pairwise operation shall make use of integer weights $\tilde{w}_1$, $\tilde{w}_2$, and $\tilde{w}_3$, and an integer s, which depend on the values of B and C as follows.

4.5.4.4.2 The value of s shall be calculated as:

$$s = \begin{cases} 1, & \text{if } ((C \geq 0 \text{ or } B = 1) \text{ and } \textit{flip} \text{ is 'false'}) \text{ or } (C < 0 \text{ and } B = -1 \text{ and } \textit{flip} \text{ is 'true'}), \\ -1, & \text{otherwise.} \end{cases} \tag{40}$$

4.5.4.4.3 To calculate the weights $\tilde{w}_1$, $\tilde{w}_2$, and $\tilde{w}_3$, first intermediate quantities $\tilde{t}$, $\tilde{p}$, and $\widetilde{\sqrt{2}}$ shall be obtained as follows:

$$\tilde{t} = \begin{cases} 0, & \text{if } C = 2^{\Omega-2} - 1, \\ B \cdot 2^{\Omega-1}, & \text{if } C = -2^{\Omega-2}, \\ B \cdot \left\lfloor \sqrt{2^{2\Omega-3} - \left\lfloor \dfrac{2^{2\Omega+1} \cdot C}{\left\lfloor \sqrt{2^8 \cdot C^2 + 2^{2\Omega}} + 0.5 \right\rfloor} + 0.5 \right\rfloor} + 0.5 \right\rfloor, & \text{otherwise;} \end{cases} \tag{41}$$

$$\tilde{p} = \left\lfloor \sqrt{2^{2\Omega+2} - 16 \cdot \tilde{t}^2} + 0.5 \right\rfloor; \tag{42}$$

and

$$\widetilde{\sqrt{2}} = \left\lfloor \sqrt{2^{2\Omega+5}} + 0.5 \right\rfloor. \tag{43}$$

NOTE – The operation $\lfloor \sqrt{\cdot} + 0.5 \rfloor$ produces a well-defined integer quantity, and the use of floating-point approximations in an attempt to perform this operation may yield an incorrect, and thus noncompliant, result. See annex G for reference implementations.

4.5.4.4.4 For a balanced pairwise operation, weights $\tilde{w}_1$, $\tilde{w}_2$, and $\tilde{w}_3$ shall be obtained as follows:

$$\tilde{w}_1 = \begin{cases} \left\lfloor \dfrac{2^{\Omega-4}\cdot(2\tilde{p}-\widetilde{\sqrt{2}})}{\tilde{t}} + 0.5 \right\rfloor, & \text{if } C<0, \\ \left\lfloor \dfrac{2^{\Omega-3}\cdot(\widetilde{\sqrt{2}}-16\tilde{t})}{\tilde{p}} + 0.5 \right\rfloor, & \text{if } C\geq 0, \end{cases} \tag{44}$$

$$\tilde{w}_2 = \begin{cases} \left\lfloor \dfrac{\widetilde{\sqrt{2}}\cdot\tilde{t}}{2^{\Omega+3}} + 0.5 \right\rfloor, & \text{if } C<0, \\ \left\lfloor -\dfrac{\widetilde{\sqrt{2}}\cdot\tilde{p}}{2^{\Omega+4}} + 0.5 \right\rfloor, & \text{if } C\geq 0, \end{cases} \tag{45}$$

$$\tilde{w}_3 = \begin{cases} \left\lfloor \dfrac{2^{\Omega-4}\cdot(4\tilde{p}-\widetilde{\sqrt{2}})}{\tilde{t}} + 0.5 \right\rfloor, & \text{if } C<0, \\ \left\lfloor \dfrac{2^{\Omega-3}\cdot(\widetilde{\sqrt{2}}-8\tilde{t})}{\tilde{p}} + 0.5 \right\rfloor, & \text{if } C\geq 0. \end{cases} \tag{46}$$

4.5.4.4.5 For an unbalanced pairwise operation, these weights shall be obtained as follows:

$$\tilde{w}_1 = \begin{cases} \left\lfloor \dfrac{2^{\Omega-3}\cdot\left\lfloor\dfrac{\widetilde{\sqrt{2}}\cdot\tilde{p}}{2^{\Omega+2}}+0.5\right\rfloor - 2^{2\Omega-1}}{\tilde{t}} + 0.5 \right\rfloor, & \text{if } C<0, \\ \left\lfloor \dfrac{2^{\Omega-1}\cdot\left\lfloor\dfrac{\widetilde{\sqrt{2}}(2^{\Omega-2}-\tilde{t})}{2^{\Omega}}+0.5\right\rfloor}{\tilde{p}} + 0.5 \right\rfloor, & \text{if } C\geq 0, \end{cases} \tag{47}$$

$$\tilde{w}_2 = \begin{cases} \left\lfloor \dfrac{\tilde{t}}{2} + 0.5 \right\rfloor, & \text{if } C<0, \\ \left\lfloor -\dfrac{\widetilde{\sqrt{2}}\cdot\tilde{p}}{2^{\Omega+4}} + 0.5 \right\rfloor, & \text{if } C\geq 0, \end{cases} \tag{48}$$

$$\tilde{w}_3 = \begin{cases} \left\lfloor \dfrac{2^{\Omega-5} \cdot \left\lfloor \dfrac{\widetilde{\sqrt{2}} \cdot \tilde{p}}{2^{\Omega-1}} + 0.5 \right\rfloor - 2^{2\Omega-1}}{\tilde{t}} + 0.5 \right\rfloor, & \text{if } C < 0, \\ \left\lfloor \dfrac{2^{\Omega-1} \cdot \left\lfloor \dfrac{\widetilde{\sqrt{2}}(2^{\Omega} - \tilde{t})}{2^{\Omega+2}} + 0.5 \right\rfloor}{\tilde{p}} + 0.5 \right\rfloor, & \text{if } C \geq 0. \end{cases} \tag{49}$$

NOTE – A look-up table, using B and C as indices, can be used to efficiently obtain $\tilde{w}_1$, $\tilde{w}_2$, $\tilde{w}_3$, and s.

4.5.4.5 Forward Calculation

4.5.4.5.1 Given integers a, b, and $\tilde{w}$, the function Λ shall satisfy

$$\Lambda(a, \tilde{w}, b) = a + \left\lfloor \frac{\tilde{w} \cdot b + 2^{\Omega-2}}{2^{\Omega-1}} \right\rfloor. \tag{50}$$

4.5.4.5.2 For each $i = 0, \ldots, N-1$ given inputs x_i and y_i, outputs χ_i and ϕ_i shall be obtained as follows

$$\chi_i = \begin{cases} s \cdot k_{2,i}, & \text{if } C < 0, \\ s \cdot \Lambda(k_{1,i}, \tilde{w}_3, k_{2,i}), & \text{if } C \geq 0, \end{cases} \tag{51}$$

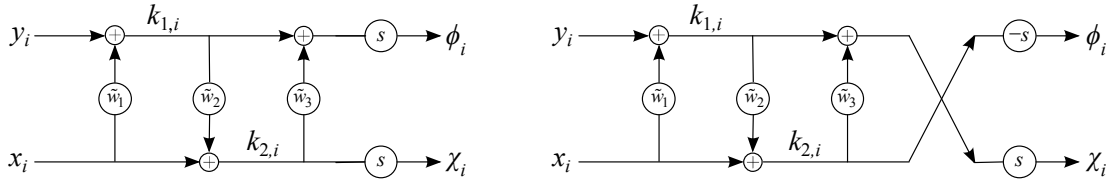
$$\phi_i = \begin{cases} s \cdot \Lambda(k_{1,i}, \tilde{w}_3, k_{2,i}), & \text{if } C < 0, \\ -s \cdot k_{2,i}, & \text{if } C \geq 0, \end{cases} \tag{52}$$

where

$$k_{1,i} = \Lambda(y_i, \tilde{w}_1, x_i), \text{ and} \tag{53}$$

$$k_{2,i} = \Lambda(x_i, \tilde{w}_2, k_{1,i}). \tag{54}$$

NOTE – Figure 4-7 depicts the calculation of the pairwise operation output (the values χ_i and ϕ_i) using a lifting network. The input values x_i and y_i are transformed by three lifting steps (having weights $\tilde{w}_1$, $\tilde{w}_2$, $\tilde{w}_3$), the result may then be permuted (when $C \geq 0$), and inverted (depending on the value of s).



(a) When C < 0 (b) When C ≥ 0

Figure 4-7: Lifting Network of the Pairwise Operation

4.5.4.6 Inverse Calculation

4.5.4.6.1 Given integers b, c, and $\tilde{w}$, where $c = \Lambda(a, \tilde{w}, b)$ for some integer a, equation (50) may be inverted to recover the value of a:

$$\Lambda^{-1}(c, \tilde{w}, b) = c - \left\lfloor \frac{\tilde{w} \cdot b + 2^{\Omega-2}}{2^{\Omega-1}} \right\rfloor = a \, . \tag{55}$$

4.5.4.6.2 For each $i = 0, \ldots, N-1$, inputs x_i and y_i and may be recovered from outputs χ_i and ϕ_i as

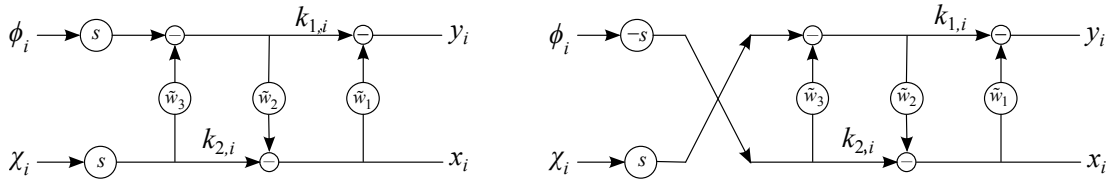
$$\begin{aligned} x_i &= \Lambda^{-1}(k_{2,i}, \tilde{w}_2, k_{1,i}), \\ y_i &= \Lambda^{-1}(k_{1,i}, \tilde{w}_1, x_i), \end{aligned} \tag{56}$$

where the quantities $k_{1,i}$, $k_{2,i}$ defined in equations (53) and (54) can be reconstructed as

$$k_{2,i} = \begin{cases} s \cdot \chi_i, & \text{if } C < 0, \\ -s \cdot \phi_i, & \text{if } C \geq 0, \end{cases} \tag{57}$$

$$k_{1,i} = \begin{cases} \Lambda^{-1}(s \cdot \phi_i, \tilde{w}_3, k_{2,i}), & \text{if } C < 0, \\ \Lambda^{-1}(s \cdot \chi_i, \tilde{w}_3, k_{2,i}), & \text{if } C \geq 0. \end{cases} \tag{58}$$

NOTE – Figure 4-8 depicts the calculation of the inverse pairwise operation using an inverse lifting network.



(a) C < 0 (b) C ≥ 0

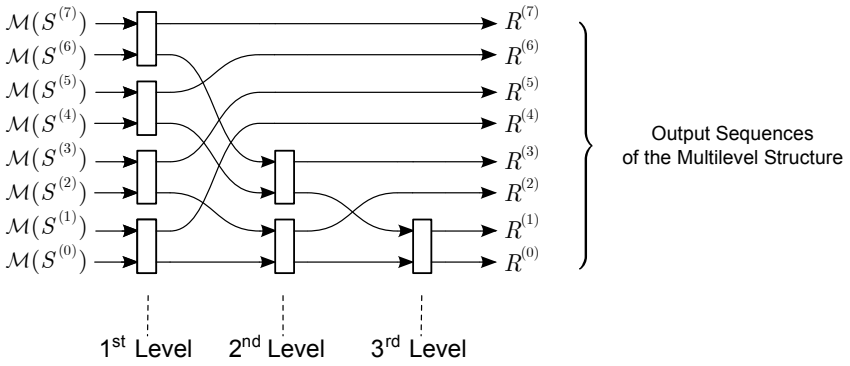
Figure 4-8: Inverse Lifting Network of the Pairwise Operation

4.5.5 MULTILEVEL STRUCTURE

4.5.5.1 Overview

A pairwise operation exploits correlation between two input sequences. To exploit correlation between multiple input sequences, mean-subtraction operations and pairwise operations are combined in a multilevel structure.

First, a mean-subtraction operation is applied to each input sequence, $S^{(i)}$, and then multiple levels of pairwise operations are applied. In the first level, a balanced pairwise operation is applied to approximately decorrelate pairs of mean-subtracted input sequences. At subsequent levels, pairwise operations are applied to the principal output sequences from pairwise operations from the immediately preceding level, concluding with a single pairwise operation applied at the last level. Figure 4-9 illustrates this multilevel structure for eight input sequences.



NOTE – Each box represents a balanced pairwise operation.

Figure 4-9: The Multilevel Structure for Eight Input Sequences

As sequences are treated in pairs, a single *unpaired* sequence (that is not modified by a pairwise operation) will arise at the first level when the number of input sequences is odd, and at higher levels when there are an odd number of principal outputs from the preceding level. Figure 4-10 shows unpaired sequences arising at the first and second level when there are five input sequences. An unpaired sequence at a given level can only serve as the secondary input of an unbalanced pairwise operation at the next level.

At the first level, when the number of input sequences is odd, the last one is unpaired. At subsequent levels, when the number of principal outputs from the preceding level is odd, the first one is unpaired at even-indexed levels and the last one is unpaired at odd-indexed levels. Alternating in this way ensures that an unpaired sequence is always paired at the next level.

Figure 4-10 illustrates the multilevel structure for five input sequences; unpaired sequences arise at the first and second levels in this case. Of the four pairwise operations employed, two of them are unbalanced because they receive an unpaired sequence. The unbalanced pairwise

operation at the third level has its inputs permuted so that the unpaired sequence is its secondary input.

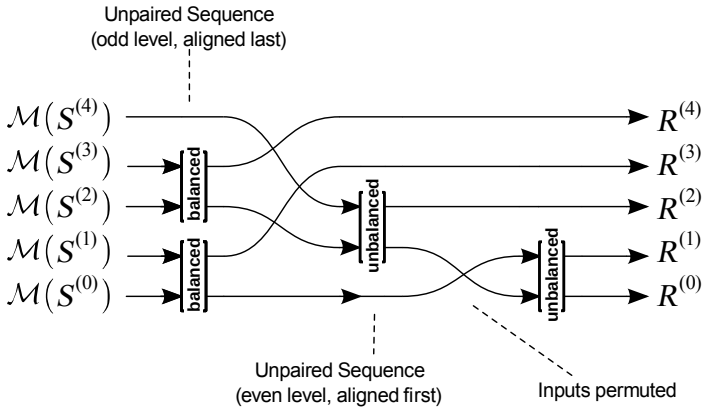


Figure 4-10: Example Alternating Alignments for the Case of Five Input Sequences and Three Levels

The output sequences, $R^{(i)}$, resulting from a multilevel structure are the detail outputs of each pairwise operation, as well as the principal output of the pairwise operation of the last level.

Subsection 4.5.5.3 specifies the structure of an individual level, and 4.5.5.4 specifies how the multilevel structure is formed by mapping outputs from one level to the inputs of the next level. Finally, 4.5.5.5 defines how the multilevel output sequences comprise output sequences from the individual levels.

4.5.5.2 Multilevel Inputs, Outputs, and Levels

4.5.5.2.1 A multilevel structure shall map N_Z integer input sequences

$$S^{(0)}, S^{(1)}, \ldots, S^{(N_Z-1)} \tag{59}$$

onto N_Z integer output sequences

$$R^{(0)}, R^{(1)}, \ldots, R^{(N_Z-1)} \tag{60}$$

of the same length.

NOTE – Each element of the previous coma-separated enumerations in equations (59) and (61) is itself a sequence of integer values by itself.

4.5.5.2.2 When $N_Z = 1$, the single input sequence $S^{(0)}$ shall produce a single output sequence $R^{(0)}$, defined as

$$R^{(0)} = \mathcal{M}(S^{(0)}). \tag{62}$$

4.5.5.2.3 When $N_Z > 1$, the multilevel structure shall comprise $\mathcal{L} = \lceil \log_2 N_Z \rceil$ individual levels, indexed from $\ell = 1$ to $\mathcal{L}$.

4.5.5.3 Individual Level

4.5.5.3.1 The input sequences at level ℓ shall be

$$I_\ell^{(0)}, \ldots, I_\ell^{(M_\ell - 1)} \tag{63}$$

where the number of input sequences is

$$M_\ell = \begin{cases} N_Z, & \ell = 1 \\ \left\lceil \frac{M_{\ell-1}}{2} \right\rceil, & \ell > 1 \end{cases}. \tag{64}$$

4.5.5.3.2 The level shall use pairwise operations to map these M_ℓ input sequences to M_ℓ integer output sequences.

4.5.5.3.3 An additional input, Boolean value $b_\ell^{(i)}$, shall be associated with each input sequence $I_\ell^{(i)}$.

NOTE – This Boolean value is used to determine the type of each pairwise operation (balanced or unbalanced).

4.5.5.3.4 The number of pairwise operations applied at level ℓ shall be

$$p_\ell = \left\lfloor \frac{M_\ell}{2} \right\rfloor. \tag{65}$$

4.5.5.3.5 For each pairwise operation, indexed from $i = 0, \ldots, p_\ell - 1$, output sequences $P_\ell^{(i)}$, $D_\ell^{(i)}$ denote, respectively, the principal and detail outputs. The input sequences to this pairwise operation shall be the pair of consecutively indexed input sequences $I_\ell^{(j)}$, $I_\ell^{(j+1)}$, where

$$j = \begin{cases} 2i + 1, & \text{if } \ell \text{ is even and } M_\ell \text{ is odd} \\ 2i, & \text{otherwise.} \end{cases} \tag{66}$$

4.5.5.3.6 If $b_\ell^{(j)}$ is 'true', then $I_\ell^{(j)}$ shall be the secondary input and $I_\ell^{(j+1)}$ shall be the primary input to the pairwise operation. Otherwise, $I_\ell^{(j)}$, $I_\ell^{(j+1)}$ shall be the primary and secondary input sequences, respectively, to the pairwise operation.

4.5.5.3.7 The pairwise operation shall be unbalanced if either of $b_\ell^{(j)}$ or $b_\ell^{(j+1)}$ is 'true'; otherwise the pairwise operation shall be balanced.

4.5.5.3.8 The value of the *flip* input for each pairwise operation shall be determined according to 4.5.7.

4.5.5.3.9 The output sequences at level ℓ shall consist of

$$\begin{aligned} &P_\ell^{(0)},\ldots,P_\ell^{(p\ell-1)}, \\ &D_\ell^{(0)},\ldots,D_\ell^{(p\ell-1)}, \end{aligned} \tag{67}$$

and, if M_ℓ is odd, one additional output sequence

$$U_\ell = \begin{cases} I_\ell^{(0)}, & \text{if } \ell \text{ is even,} \\ I_\ell^{(M_\ell-1)}, & \text{if } \ell \text{ is odd.} \end{cases} \tag{68}$$

4.5.5.4 Combining Individual Levels to Form the Multilevel Structure

4.5.5.4.1 The input sequences to the first level ($\ell = 1$) shall be produced by applying a mean-subtraction operation to each input sequence of the multilevel structure $S^{(i)}$ and assigning each associated Boolean value $b_1^{(i)}$ to 'false':

$$I_1^{(i)} = \mathcal{M}(S^{(i)}) \quad \text{for } i = 0,\ldots,N_Z - 1 \tag{69}$$

$$b_1^{(i)} = \text{'false'} \quad \text{for } i = 0,\ldots,N_Z - 1\,. \tag{70}$$

4.5.5.4.2 For subsequent levels ($\ell > 1$), for each $i = 0,\ldots,M_\ell - 1$, input sequence $I_\ell^{(i)}$ is equal to an output sequence from stage $\ell - 1$, and shall be assigned according to the following rule

$$I_\ell^{(i)} = \begin{cases} U_{\ell-1}, & \text{if } M_{\ell-1} \text{ is odd and } ((l \text{ is even and } i = M_\ell - 1) \text{ or } (l \text{ is odd and } i = 0)) \\ P_{\ell-1}^{(i-1)}, & \text{if } M_{\ell-1} \text{ is odd and } l \text{ is odd and } i > 0 \\ P_{\ell-1}^{(i)}, & \text{otherwise} \end{cases} \tag{71}$$

and the associated Boolean value for the sequence shall be assigned according to

$$b_\ell^{(i)} = \begin{cases} \text{'true'}, & \text{if } M_{\ell-1} \text{ is odd and } ((l \text{ is even and } i = M_\ell - 1) \text{ or } (l \text{ is odd and } i = 0)) \\ \text{'false'}, & \text{otherwise.} \end{cases} \tag{72}$$

4.5.5.5 Multilevel Output Sequences

4.5.5.5.1 The first output sequence $R^{(0)}$ of the multilevel structure shall equal the principal output from the final level:

$$R^{(0)} = P_{\mathcal{L}}^{(0)}. \tag{73}$$

4.5.5.5.2 The remaining output sequences $R^{(1)},\ldots,R^{(N_Z-1)}$ shall equal, respectively, the following detail outputs from the individual stages:

$$D_{\mathcal{L}}^{(0)}, D_{\mathcal{L}-1}^{(0)},\ldots,D_{\mathcal{L}-1}^{(p_{\mathcal{L}-1}-1)},\ldots,D_1^{(0)},\ldots,D_1^{(p_1-1)}, \tag{74}$$

i.e., all of the detail outputs from each level, in descending level order, and within each level arranged in ascending order of pairwise operation index.

4.5.6 TRANSFORM APPLICATION

4.5.6.1 Overview

The POT produces a transformed image, T, which is the result of applying the multilevel structure defined in 4.5.5 to the upshifted input image $I^{\uparrow}$.

4.5.6.2 Regions

4.5.6.2.1 The upshifted input image shall be partitioned along the y dimension into regions.

4.5.6.2.2 The height of each region shall be determined by a user-specified *region height* parameter F, which shall take one of the following values: 1, 2, 4, 8, 16, and 32. For $r = 0,\ldots,\left\lceil \frac{N_Y}{F} \right\rceil - 1$, region r has height

$$f_r = \min\{F, N_Y - r \cdot F\}. \tag{75}$$

NOTE – Thus each region has height of F samples, except possibly the last region which can have height less than F.

4.5.6.2.3 Each region shall then be processed by the POT, producing a corresponding region in the transformed image.

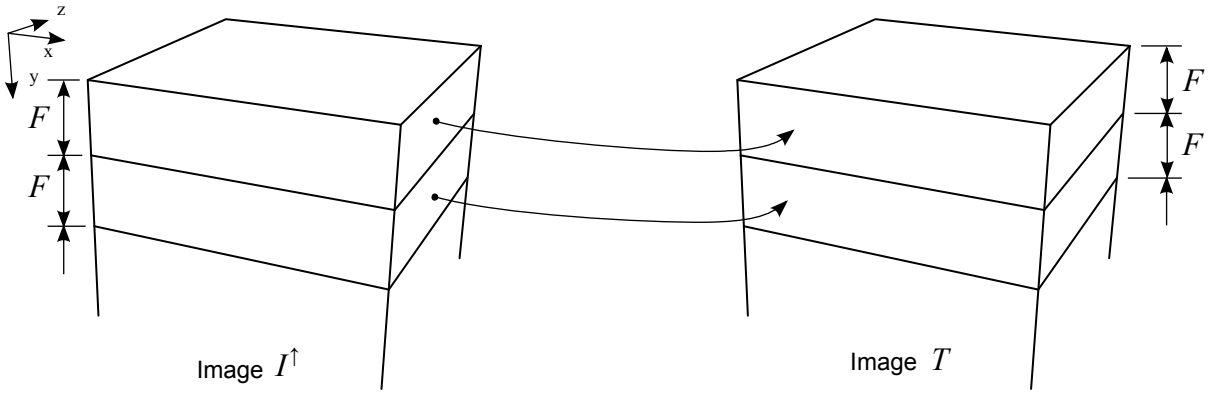


Figure 4-11: Region Partitioning

NOTE – The use of values 16 and 32 for F is possible only for certain configurations of the 2D encoders, as specified in 5.2.2.3.

4.5.6.3 Raster Scan Order

For $r = 0,\ldots,\left\lceil \frac{N_{\mathrm{Y}}}{F} \right\rceil - 1$, and $z = 0,\ldots, N-1$, the one-dimensional sequence $\Theta_{z,r}^{(A)}$ shall consist of all samples from band z of the rth region of the image A taken in raster scan order; i.e.,

$$\Theta_{z,r}^{(A)} = \left\{ \theta_0^{(A)}, \ldots, \theta_{f_r \cdot N_{\mathrm{X}} - 1}^{(A)} \right\}, \tag{76}$$

where

$$\theta_i^{(A)} = A_{(i \bmod N_{\mathrm{X}}), \left(\left\lfloor \frac{i}{N_{\mathrm{X}}} \right\rfloor + r \cdot F\right), z}. \tag{77}$$

4.5.6.4 Region Mapping

4.5.6.4.1 For each region in $I^{\uparrow}$, a multilevel structure with N_{Z} inputs shall be used to produce an equivalent region in T as follows.

4.5.6.4.2 The inputs of the multilevel structure employed in region r shall be

$$S^{(z)} = \Theta_{z,r}^{(I^{\uparrow})} \quad \text{for } z = 0,\ldots, N_{\mathrm{Z}} - 1. \tag{78}$$

4.5.6.4.3 The outputs of the multilevel structure employed in region r shall be

$$\Theta_{z,r}^{(T)} = R^{(z)} \quad \text{for } z = 0,\ldots, N_{\mathrm{Z}} - 1. \tag{79}$$

4.5.6.4.4 For $r = 0,\ldots,\left\lceil \frac{N_Y}{F} \right\rceil - 1$, $z = 0,\ldots,N_Z - 1$, and $i = 0,\ldots,f_r \cdot N_X - 1$, each element $\theta_i^{(T)}$ of one-dimensional sequence $\Theta_{z,r}^{(T)}$ shall be mapped into an element of the *z*th band of the *r*th region of the transformed image as follows:

$$T_{(i \bmod N_X),\left(\left\lfloor \frac{i}{N_X} \right\rfloor + r \bullet F\right),z} = \theta_i^{(T)}. \tag{80}$$

NOTE – The transform application process is depicted in figure 4-12.

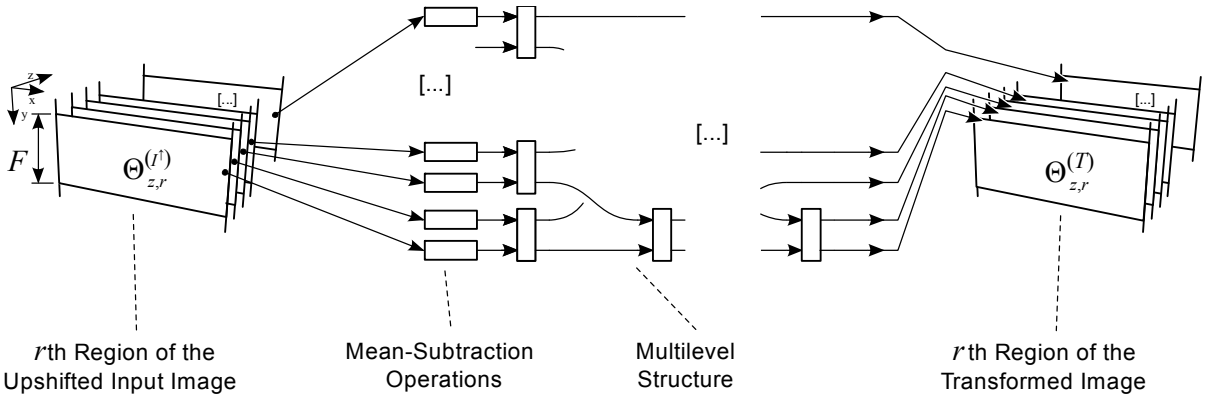


Figure 4-12: Application of the POT

4.5.7 *FLIP*

4.5.7.1 Overview

This subsection specifies the two allowed methods, called *bypass* mode and *stable* mode, for assigning the *flip* inputs in the multilevel structure used to compute the POT.

In bypass mode, each *flip* input is 'false'. This approach can sometimes lead to discontinuities between regions in the transformed image. The POT can still be inverted when such discontinuities occur, and thus lossless compression can still be achieved, but compression effectiveness is reduced, and distortion increases near such a discontinuity when compression is lossy.

Under stable mode, all *flip* inputs are set to 'false' for the first region, but for subsequent regions the discontinuities are mitigated by altering the values of the *flip* inputs from one region to the next. In figure 4-13, pairwise operation *A* has *flip* input $flip_A$ and training parameters B_A, C_A, and pairwise operation *B* (with *flip* input $flip_B$ and training parameters B_B, C_B) is the corresponding pairwise operation, i.e., having the same level and index as *A*, in the next region. The value of $flip_B$ depends on both the value of $flip_A$ as well as the values of training parameters B_A, C_A, B_B, and C_B under stable mode, as specified in 4.5.7.5.

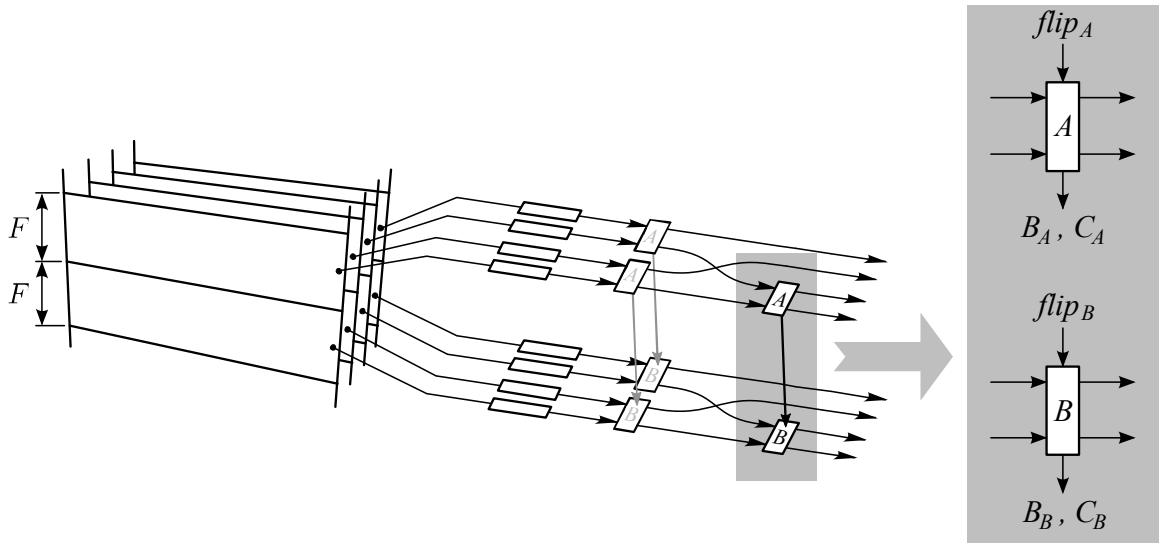


Figure 4-13: The Pairwise Operation Dependencies in Stable Mode

The output of a pairwise operation depends on the value of the *flip* input. Hence, within a region, it is necessary to determine the value of *flip* for each pairwise operation within a level, before the values of training parameters *B* and *C* can be computed for pairwise operations in subsequent levels.

4.5.7.2 Modes

For a given image, a user shall have the ability to choose either bypass mode or stable mode to determine the value of each *flip* input in the POT calculation.

4.5.7.3 Discussion—Notation

For each $\ell = 1,\ldots,\mathcal{L}$ and $i = 0,\ldots, p_\ell - 1$ and $r = 0,\ldots,\left\lceil \frac{N_{\mathrm{Y}}}{F} \right\rceil - 1$, $flip_{i,\ell}^{(r)}$ denotes the *flip* input to the *i*th pairwise operation of the ℓth level of the multilevel structure applied to region *r*, and $B_{i,\ell}^{(r)}$, $C_{i,\ell}^{(r)}$ denotes the training parameters output from this pairwise operation.

4.5.7.4 Bypass Mode

In bypass mode, all *flip* inputs in each multilevel structure shall be set to 'false':

$$flip_{i,\ell}^{(r)} = \text{'false'}, \tag{81}$$

for each $\ell = 1,\ldots,\mathcal{L}$ and $i = 0,\ldots, p_\ell - 1$ and $r = 0,\ldots,\left\lceil \frac{N_{\mathrm{Y}}}{F} \right\rceil - 1$.

4.5.7.5 Stable Mode

In stable mode, the value of each *flip* input shall be computed as follows:

$$flip_{i,\ell}^{(r)} = \begin{cases} \text{'false'}, & \text{if } r = 0, \\ \neg\, flip_{i,\ell}^{(r-1)}, & \text{if } r > 0 \text{ and } B_{i,\ell}^{(r)} = B_{i,\ell}^{(r-1)} = -1 \\ & \quad \text{and} \left(\left(C_{i,\ell}^{(r)} \geq 0 \text{ and } C_{i,\ell}^{(r-1)} < 0 \right) \text{or} \left(C_{i,\ell}^{(r)} < 0 \text{ and } C_{i,\ell}^{(r-1)} \geq 0 \right) \right), \\ flip_{i,\ell}^{(r-1)}, & \text{otherwise,} \end{cases} \tag{82}$$

for each $\ell = 1,\ldots,\mathcal{L}$ and $i = 0,\ldots, p_\ell - 1$ and $r = 0,\ldots,\left\lceil \frac{N_{\mathrm{Y}}}{F} \right\rceil - 1$.

4.5.8 SIDE INFORMATION

The following values, which make up the side information associated with the POT, shall be encoded in a compressed image header in the format specified by 5.4.4.3:

- the value F;
- the value Ω;
- the mode (bypass or stable) used to calculate the *flip* values;
- for each region index $r = 0\ldots,\left\lceil \frac{N_{\mathrm{Y}}}{F} \right\rceil - 1$:
  - the N_{Z} nominal mean values m used in the mean-subtraction operations,
  - the $(N_{\mathrm{Z}} - 1)$ pairs of training parameters $B_{i,\ell}^{(r)}$ and $C_{i,\ell}^{(r)}$ from the pairwise operations;
- when in stable mode, the $(N_{\mathrm{Z}} - 1)$ values of $flip_{i,\ell}^{(r)}$ for some of the regions, as described in 5.4.4.3.

NOTE – While the values of $flip_{i,\ell}^{(r)}$ can be obtained from pairs of training parameters $B_{i,\ell}^{(r)}$ and $C_{i,\ell}^{(r)}$, encoding some of the $flip_{i,\ell}^{(r)}$ values in a compressed image header is necessary to resume decompression in the event of data loss within a compressed image.

4.5.9 INVERSE TRANSFORM APPLICATION

4.5.9.1 The Inverse POT shall recover an upshifted input image, $I^{\uparrow}$ (or, in the case of lossy compression, an approximate version of $I^{\uparrow}$), by inverting the multilevel structures that produced the transformed image T. This process is depicted in figure 4-14.

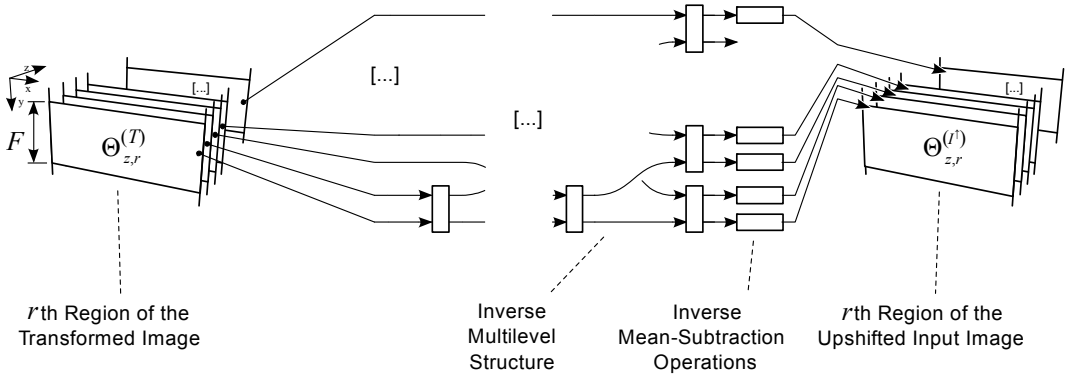


Figure 4-14: Inverse Application of the POT

4.5.9.2 To invert the POT using the stored side information, each of the pairwise operations shall be inverted, as described in 4.5.4.6, in the opposite order in which they were applied. Then, the mean-subtraction operations shall be inverted as described in 4.5.3.4.

4.6 ARBITRARY AFFINE TRANSFORM

4.6.1 OVERVIEW

The Arbitrary Affine Transform (AAT) maps an upshifted input image to a transformed image via one or more user-specified one-dimensional affine operations. An affine operation, defined in 4.6.3, is independently applied to vectors in the upshifted input image; each such vector consists of all samples sharing the same position for the x and y dimensions. The user-specified parameters that determine the affine operation consist of a matrix and a vector, as described in 4.6.3.2.

The upshifted input image is partitioned into regions, as described in 4.6.4, and a different affine operation may be used for each such segment. The AAT could thus be used in a way that adapts to the input image on a region-by-region basis, or a fixed transform could be used for the complete image.

While the AAT allows the use of a KLT computed for each image segment, the KLT has significant computational cost, and so prospective users of this approach may want to employ techniques to mitigate this cost, such as covariance subsampling (reference [C10]), divide-and-conquer approximations (reference [C11]), or exogenous KLTs (references [C12] and [C13]).

Unlike the IWT and the POT, the AAT is not guaranteed to be exactly reversible, i.e., lossless compression cannot be guaranteed when the AAT is used.

4.6.2 DIMENSIONS

Under the AAT, the number of bands in the transformed image, N_{TZ}, is a user-specified parameter, which shall be an integer in the range $1 \leq N_{TZ} \leq N_Z$.

NOTE – N_{TZ} can be less than N_Z.

4.6.3 AFFINE OPERATION

4.6.3.1 Inputs and Outputs

Affine operations are used to produce the AAT. Each affine operation shall map a length-N_Z input sequence

$$X = \{x_0, x_1, \ldots, x_{N_Z-1}\} \tag{83}$$

to a length-N_{TZ} output sequence

$$Y = \{y_0, y_1, \ldots, y_{N_{TZ}-1}\}. \tag{84}$$

NOTE – The notation $\mathcal{A}$ is used to denote this mapping:

$$Y = \mathcal{A}(X). \tag{85}$$

4.6.3.2 Parameters

4.6.3.2.1 The arithmetic precision of each affine transform calculation is controlled by the user-specified parameter Ψ, which shall be an integer in the range $0 \leq \Psi \leq 32$, and shall be fixed for a given image.

4.6.3.2.2 A user-specified $N_{TZ} \times N_Z$ integer matrix Q and a user-specified length-N_Z integer vector V shall determine each affine operation used to produce the AAT:

$$Q = \begin{pmatrix} q_{0,0} & q_{0,1} & \cdots & q_{0,N_Z-1} \\ q_{1,0} & q_{1,1} & \cdots & q_{1,N_Z-1} \\ \vdots & \vdots & \ddots & \vdots \\ q_{N_{TZ}-1,0} & q_{N_{TZ}-1,1} & \cdots & q_{N_{TZ}-1,N_Z-1} \end{pmatrix}, \quad V = \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{N_Z-1} \end{pmatrix}. \tag{86}$$

4.6.3.2.3 All elements of Q and V shall be in the range from -2^{31} to $2^{31} - 1$.

4.6.3.3 Forward Calculation

Each value of the output sequence Y shall be obtained as follows:

$$y_j = \sum_{i=0}^{N_Z-1} [\frac{q_{j,i}}{2^\Psi} \cdot (x_i + v_i)], \quad \text{for } j = 0, \ldots, N_{TZ} - 1. \tag{87}$$

NOTE – The calculation specified here is an integer approximation to the affine transformation $Y = \frac{1}{2^\Psi} Q(X + V)$.

4.6.3.4 Discussion—Inverse Calculation

The inverse of an affine operation, $\mathcal{A}^{-1}$, is user-defined.

NOTE – Since an affine operation is an integer approximation to the affine transformation $Y = \frac{1}{2^\Psi} Q(X + V)$, when matrix Q has an inverse Q^{-1}, users may wish to invert the component transform via an integer approximation to

$$\mathcal{A}^{-1}(Y) = 2^\Psi Q^{-1} Y - V. \tag{88}$$

4.6.4 TRANSFORM APPLICATION

4.6.4.1 Regions shall be defined in the upshifted input image by partitioning it along the y dimension (see figure 4-11).

4.6.4.2 The height of each region shall be determined by a user-specified *region height* parameter F, which shall take one of the following values: 1, 2, 4, 8, 16, and 32. For $r = 0, \ldots, \left\lceil \frac{N_Y}{F} \right\rceil - 1$, region r has height

$$f_r = \min\{F, N_Y - r \cdot F\}. \tag{89}$$

NOTES

1 Thus each region has height of F samples, except the last region which may have height less than F.

2 The use of values 16 and 32 for F is only possible for certain configurations of the 2D encoders, as specified in 5.2.2.3.

4.6.4.3 For each $r = 0, \ldots, \left\lceil \frac{N_Y}{F} \right\rceil - 1$, an affine operation $\mathcal{A}_r$ shall be defined for region r by specifying matrix Q_r and vector V_r that satisfy the constraints given in 4.6.3.2.2 and 4.6.3.2.3.

4.6.4.4 For each $x = 0,\ldots, N_X - 1$ and $y = 0,\ldots, N_y - 1$, the sequence of transformed image samples $\left\{T_{x,y,0},\ldots,T_{x,y,N_{TZ}-1}\right\}$ shall be computed from the sequence of upshifted image samples $\left\{I^{\uparrow}_{x,y,0},\ldots,I^{\uparrow}_{x,y,N_Z-1}\right\}$ using the rth affine operation $\mathcal{A}_r$:

$$\left\{T_{x,y,0},\ldots,T_{x,y,N_{TZ}-1}\right\} = \mathcal{A}_r\left(\left\{I^{\uparrow}_{x,y,0},\ldots,I^{\uparrow}_{x,y,N_Z-1}\right\}\right), \tag{90}$$

where r is the index of the region of the upshifted input image that contains these samples:

$$r = \left\lfloor \frac{y}{F} \right\rfloor. \tag{91}$$

NOTE – This application is illustrated in figure 4-15.

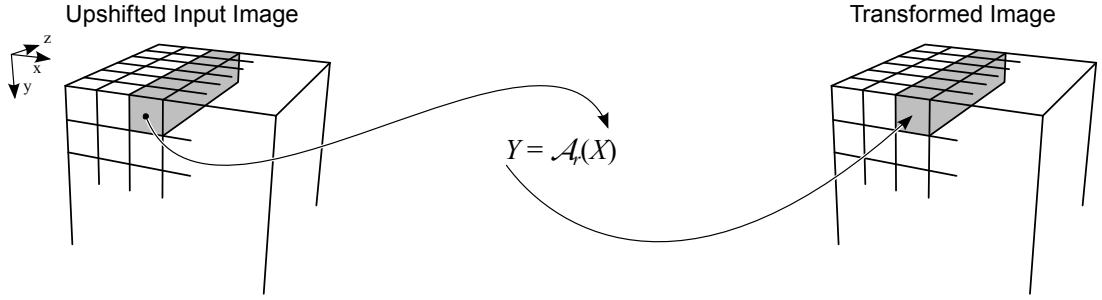


Figure 4-15: Application of the AAT

4.6.5 SIDE INFORMATION

The following values, which make up the side information associated with the AAT, shall be encoded in a compressed image header in the format specified in 5.4.4.4:

- the value F;
- the value Ψ;
- the value N_{TZ};
- for each region index $r = 0\ldots, \left\lceil \frac{N_Y}{F} \right\rceil - 1$, the matrix Q_r and the vector V_r.

4.6.6 DISCUSSION—INVERSE TRANSFORM APPLICATION

The inverse of the AAT is user-defined.

NOTE – To optimize coding performance, users may wish to consider the inversion of the AAT jointly with the inversion of the upshift and downshift stages, and possibly with the inversion of the 2D encoders.

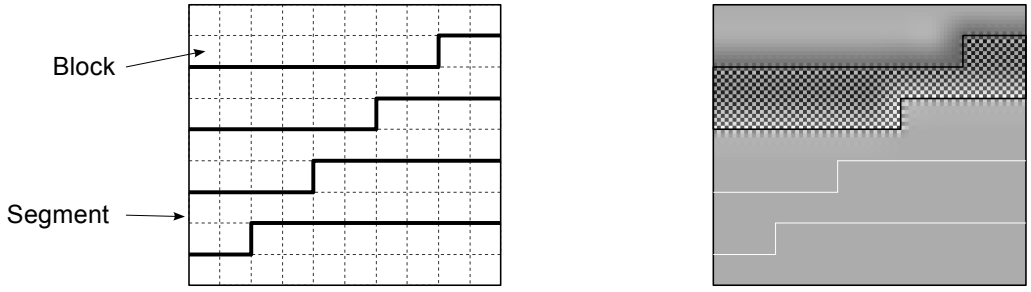
5 ENCODER

5.1 OVERVIEW

This section specifies the encoding stage of the compressor, which takes a downshifted transformed image and produces a compressed image by employing multiple instances of the 2D encoder defined in reference [1].

Each 2D encoder applies a 2D discrete wavelet transform (DWT) to a band of the downshifted transformed image. As specified in subsection 4.1 of reference [1], DWT coefficients output from this transform are arranged in 8 × 8 groups called *blocks*. Blocks can be further arranged in a rectangular array having $\lceil N_X / 8 \rceil$ columns and $\lceil N_Y / 8 \rceil$ rows of blocks. Taking successive blocks, in raster-scan order, from this array yields a *segment*, which is encoded to produce a *coded segment*. Blocks and segments are depicted in figure 5-1a.

Segment boundaries are defined in the DWT domain (following the application of the 2D DWT), and each segment loosely corresponds to a spatial portion of the 2D input image. This loose correspondence is illustrated in figure 5-1b, which is produced by reconstructing only the second segment shown in figure 5-1a (setting all other DWT coefficients to zero) for an input image consisting of a 'checkerboard' pattern. The recovered portion of the checkerboard input image does not have sharply-defined boundaries in the spatial domain because the DWT creates spatial-domain data dependencies between adjacent segments. Some informative portions of this Recommended Standard refer to a segment as corresponding to a spatial area, but readers should keep in mind that this correspondence is approximate.



(a) DWT domain (b) Spatial domain

Figure 5-1: Blocks, Segments, and Wavelet Data Dependencies

The coded segments produced by the 2D encoders are arranged in groups, and for each group a variable-length header is used to encode information describing the spectral preprocessing transform; the group and its associated header are called a *collection*. The collections form the compressed image, as illustrated in figure 5-2. A collection's header, specified in 5.4, encodes mandatory metadata and may also include setup metadata. Both mandatory metadata and setup metadata are necessary to decompress the image, but the latter might be constant for an entire mission and not necessarily produced on board, and thus is optional. The header

also includes any relevant portions of side information produced by the spectral transform stage.

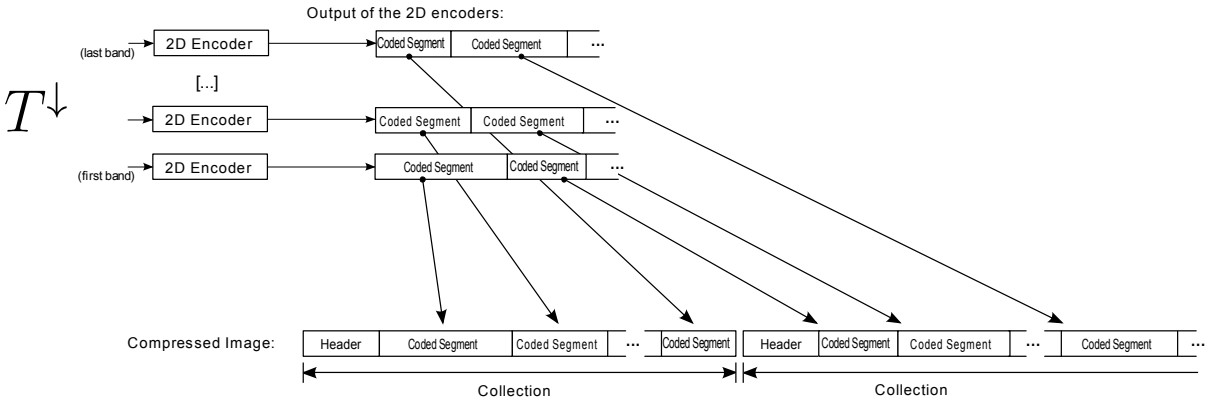
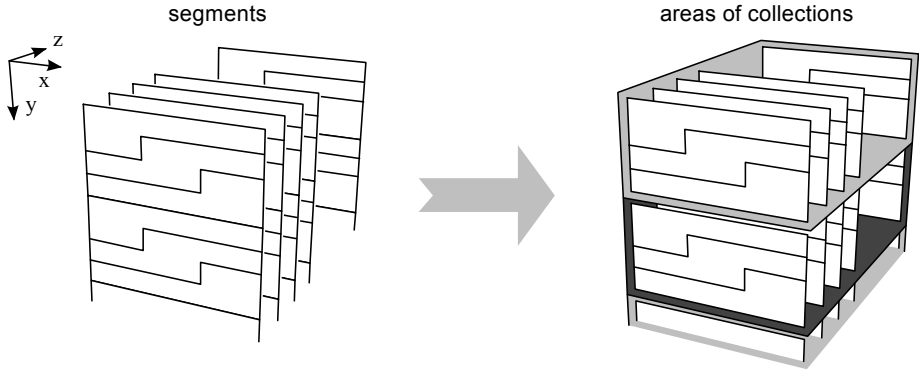


Figure 5-2: Example of the Composition of the Compressed File

User-specified parameters S' and R control the structure of a segment and a collection. The value of S' determines the number of blocks per segment, and the value of R determines the number of segments per band in each collection (see 5.2.2). These parameters thus determine the spatial area that (loosely) corresponds to each collection; this is simply referred to as the 'area' of a collection in informative portions of the text.

Together R and S' control the structure of the compressed image, but have only an indirect impact on its compressed size, which depends primarily on the fidelity with which each segment is encoded. Thus, a segment with only a few blocks encoded at high fidelity could yield a larger coded segment than one comprising many blocks but encoded at low fidelity.



NOTE – In this example R is 3.

Figure 5-3: Portions of an Image Associated with Segments and Segments Grouped in Collections

To ensure regularity in the definition of collections, constraints are imposed on the relationship between R and S', and on the allowed values of region height F (when a transform stage that defines a region height is used), so that:

- all segments encode the same number of blocks, except possibly the last one of each band, which may encode fewer (5.2.2 and 5.3.2);
- all collections include matching sets of segments from each transformed spectral bands (5.3.3);
- all collections are associated with a rectangularly shaped areas, which span the full width of the image and all its bands (5.2.2.3); and
- all regions, as defined in 4.5.6.2 and 4.6.4.1, are constrained to fit completely within the area covered by a single collection; i.e., no region can cross the boundary from one collection to another (5.2.2.3).

Figure 5-4 is a comprehensive depiction of all the dimensions related to regions, segments, and collections.

Decoding of collections can be performed either with a single 2D decoder or with multiple 2D decoders in parallel. Following the 2D decoder(s), inverse spectral transform needs to be employed to reconstruct the 3D images. Should parallel decoding of segments be desirable, in addition to the location of collection headers, the location of coded segment headers needs to be known in advance. This information may be provided by the transport mechanism (see 2.4), such as when a packet protocol is employed to transfer collection headers and coded segments separately.

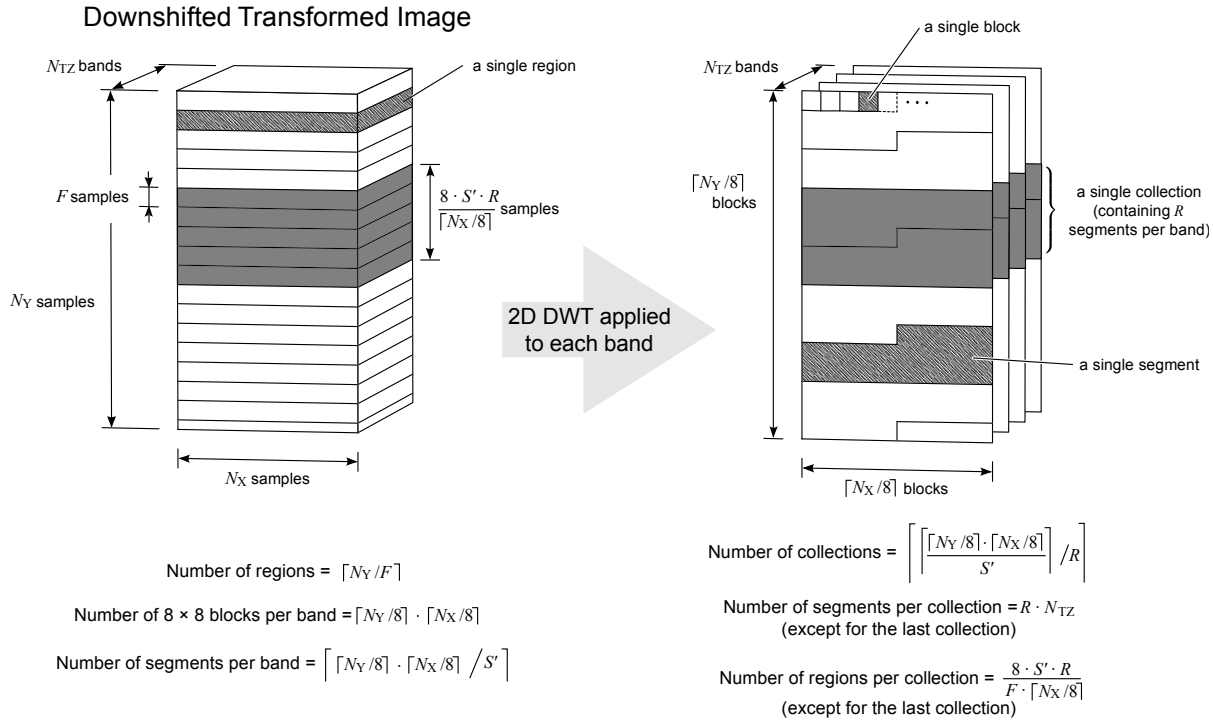


Figure 5-4: Dimensions Related to Regions, Segments, and Collections

5.2 SPECIFICATION

5.2.1 DEFINITIONS

5.2.1.1 A compressed image shall consist of a succession of one or more *collections*, defined in 5.2.1.2.

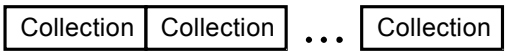


Figure 5-5: Compressed Image Structure

5.2.1.2 A collection shall consist of a *header*, defined in 5.4, followed by *coded segments*, defined in 5.3.

Figure 5-6: Overview of the Structure of a Collection

5.2.2 USER-SPECIFIED PARAMETERS

5.2.2.1 General

The following four user-specified parameters shall control the arrangement of the elements of a compressed image:

a) S', which determines the number of 8 × 8 *blocks per segment* (of DWT coefficients from the 2D encoder)

b) R, which determines the number of *coded segments per transformed band in each collection*;

c) when the POT or AAT is used, the user-specified parameter F, which controls *region height*; and

d) W, which controls the *output word size*, measured in bytes.

5.2.2.2 Coded Segments per Collection

Each collection shall contain R coded segments from each band, except possibly the last collection, which may contain fewer. Specifically, the last collection shall contain R' coded segments from each band, where R' is defined as

$$R' = \left(\left\lceil \frac{\lceil N_X / 8 \rceil \cdot \lceil N_Y / 8 \rceil}{S'} - 1 \right\rceil \bmod R \right) + 1 . \tag{92}$$

NOTE – R' is an integer in the range $1 \le R' \le R$.

5.2.2.3 Constraints

The values of the parameters defined in this section shall meet the following constraints.

a) S' shall be an integer in the range $16 \leq S' \leq 2^{20}$.

b) R shall be an integer in the range $1 \leq R \leq 2^{16}$.

c) W shall be an integer in the range $1 \leq W \leq 8$.

d) The values of R and S' shall furthermore be selected so that

$$\left(S' \cdot R\right) \bmod \left\lceil \frac{N_{\mathrm{X}}}{8} \right\rceil = 0 . \tag{93}$$

e) F shall only take the value of 16 or 32 provided that

$$\frac{8RS'}{\lceil N_{\mathrm{X}} / 8 \rceil} \bmod F = 0 . \tag{94}$$

NOTES

1 Constraint d) ensures that the area corresponding to a collection is a rectangularly shaped area covering the entire width of the image.

2 While F may take various values, as defined either in 4.5.6.2 or 4.6.4.1, constraint e) ensures that regions do not cross collection boundaries.

5.3 SEGMENTS

5.3.1 GENERAL

5.3.1.1 Each band of the downshifted transformed image $T_z^{\downarrow}$ shall be coded, as an independent 2D image, using the 2D image compressor specified in reference [1].

5.3.1.2 The encoding shall produce a succession of *coded segments* defined in subsection 4.1 of reference [1].

NOTES

1 There are N_{TZ} 'successions of coded segments'; one succession for each band of $T^{\downarrow}$.

2 As defined in reference [1], at least one coded segment is always produced for each band.

5.3.2 2D ENCODER PARAMETERS

The parameters used in each instance of the 2D compressor shall meet the following constraints.

a) The parameter S, defined in subsection 4.1 of reference [1], shall be equal to S' for all segments, except for the last one of each band, for which it shall be equal to:

$$\left(\left(\left\lceil \frac{N_X}{8} \right\rceil \cdot \left\lceil \frac{N_Y}{8} \right\rceil - 1\right) \bmod S'\right) + 1. \tag{95}$$

b) The value of *OptDCSelect*, defined in subsection 4.2.4 of reference [1], shall be the same for all bands.

c) The value of *OptACSelect*, defined in subsection 4.2.4 of reference [1], shall be the same for all bands.

d) Either the Float DWT or Integer DWT may be used, as defined in section 3 of reference [1] and signaled by the parameter *DWTtype* defined in in subsection 4.2.5 of reference [1], but the same DWT shall be used for all bands.

e) The parameter *SignedPixels*, defined in subsection 4.2.5 of reference [1], shall be '0' if the identity transform is employed and the input image is unsigned; otherwise it shall be '1'.

   1) The pixel bit depth parameter (the value of R in subsection 3.2.1 of reference [1]) shall be at least as large as the bit depth of the band $T_z^{\downarrow}$ being encoded, but shall be no larger than the maximum allowed value indicated in subsection 3.2.1 of reference [1].

   2) The pixel bit depth should be equal to the maximum allowed value indicated in subsection 3.2.1 of reference [1].

   3) For the purposes of the bit depth calculation, $T_z^{\downarrow}$ shall be considered signed, except when *SignedPixels* is set to '0', in which case $T_z^{\downarrow}$ shall be considered unsigned.

f) The parameter ImageWidth, defined in subsection 4.2.5 of reference [1], shall be N_X.

g) The parameter TransposeImg, defined in subsection 4.2.5 of reference [1], shall be '0'.

h) The parameter CodeWordLength, defined in subsection 4.2.5 of reference [1], shall be W.

i) The same set of wavelet sub-band weights, defined in subsection 3.9 of reference [1], shall be used for all bands. Thus, either all bands shall employ standard weights, or all bands shall employ the same set of custom weights.

NOTES

1 When a maximum compressed image size is desired, users are expected to select an appropriate value for the parameter *SegByteLimit*, defined in subsection 4.2.3 of reference [1], for each segment. Optionally, users can employ the parameter *UseFill*, defined in subsection 4.2.3 of reference [1], to ensure the compressed image is padded up to the specified fixed length. Reference [C1] provides strategies to set these values.

2 When a 'quality' limit is desired, users can select appropriate values for parameters *DCStop*, *BitPlaneStop*, and *StageStop*, defined in subsection 4.2.3 of reference [1]. Reference [C1] provides strategies to set these values.

3 Parameters *PixelBitDepth* and *ExtendedPixelBitDepthFlag* are not necessary to decompress a compressed image, and any valid value provides the same result. The bit depth of the input image is provided in the Bit Depth field of the Setup Metadata part defined in 5.3.3.

5.3.3 SEGMENT ORDER

5.3.3.1 The coded segments produced by coding the downshifted transformed image band $T_i^{\downarrow}$ shall be partitioned into groups and assigned to collections in the following manner: the first collection comprises the first R coded segments of each band, the second collection comprises the next R coded segments of each band, and so on until all coded segments are grouped into collections.

5.3.3.2 Within a collection, all coded segments from one band shall be arranged in order of increasing *absolute segment count* before any of the coded segments from the next (higher-indexed) band, where the absolute segment count shall equal zero for the first segment of a 2D image and shall increment by one for each subsequent segment of that image.

NOTES

1 The arrangement of coded segment data within a collection is analogous to band-sequential order.

2 The absolute segment count is equivalent to the *SegmentCount* (defined in 4.2.2.2.3 of reference [1]), except for not wrapping back to zero after reaching a value of 255.

5.4 HEADER

5.4.1 GENERAL

5.4.1.1 The header of a collection shall consist of the following parts in the following order, as depicted in figure 5-7:

a) Mandatory Metadata—4 bytes, mandatory (see 5.4.2);

b) Setup Metadata—5 bytes, optional (see 5.4.3);

c) Side Information—variable length, conditional (see 5.4.4);

d) Padding—variable length, conditional (see 5.4.5).

5.4.1.2 The Mandatory Metadata header part shall be present in all headers.

5.4.1.3 The Setup Metadata part shall be present only when indicated in the Mandatory Metadata part.

5.4.1.4 The Side Information part shall be included whenever a transform that produces side information (see 5.4.4) is used.

5.4.1.5 The Padding part shall be used whenever padding bits are needed so that the total header length is a multiple of the word size W (see 5.4.5).

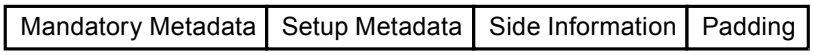


Figure 5-7: Overview of the Header Structure when All Header Parts Are Included

5.4.2 MANDATORY METADATA

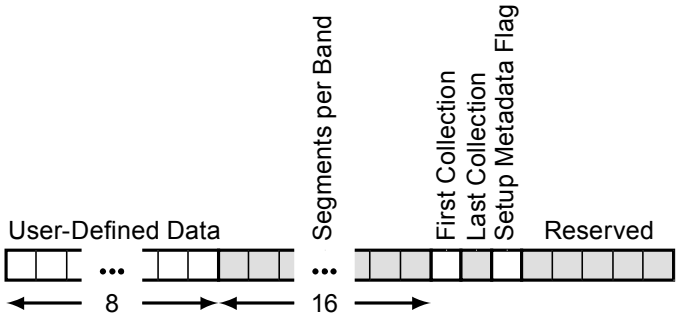


Figure 5-8: Overview of Mandatory Metadata Structure

The Mandatory Metadata header part, depicted in figure 5-8, shall consist of the fields specified in table 5-1, arranged in the order listed.

Table 5-1: Mandatory Metadata

Field	Width (bits)	Description	Reference
User-Defined Data	8	The user may assign the value of this field arbitrarily, e.g., to indicate the value of a user-defined index of the image within a set of images.	
Segments per Band	16	For all but the last collection, the value of R encoded modulo 2^{16} as a 16-bit unsigned binary integer. For the last collection, the value of R' encoded modulo 2^{16} as a 16-bit unsigned binary integer.	5.2.2
First Collection	1	'0': this is not the first collection in the compressed image. '1': this is the first collection.	
Last Collection	1	'0': this is not the last collection in the compressed image. '1': this is the last collection.	
Setup Metadata Flag	1	'0': no Setup Metadata part is present in this header. '1': a Setup Metadata part is present in this header.	5.4.3
Reserved	5	This field shall contain all 'zeros'.	

5.4.3 SETUP METADATA

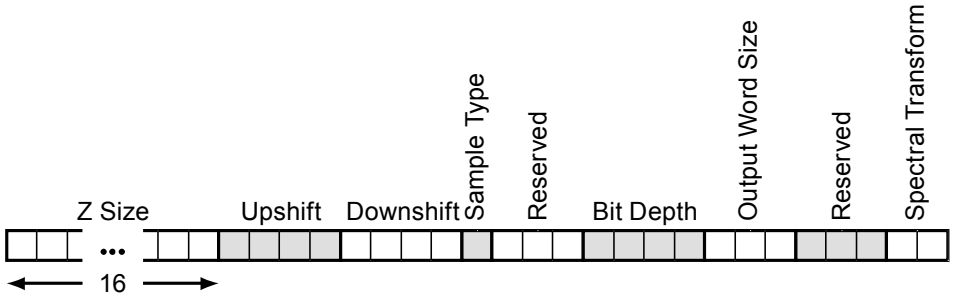


Figure 5-9: Overview of Setup Metadata Structure

The Setup Metadata header part, depicted in figure 5-9, shall consist of the fields specified in table 5-2, arranged in the order listed. The Setup Metadata part shall be present only if the *Setup Metadata Flag* of the Mandatory Metadata part is set to '1'.

Table 5-2: Setup Metadata

Field	Width (bits)	Description	Reference
Transformed Z Size	16	The value N_{TZ} encoded mod 2^{16} as a 16-bit unsigned binary integer.	4.3.2, 4.4.2, 4.5.2, 4.6.2
Upshift	4	The value U encoded as a 4-bit unsigned binary integer.	3.2
Downshift	4	The value D encoded as a 4-bit unsigned binary integer.	3.2
Sample Type	1	'0': input image sample values are unsigned integers. '1': input image sample values are signed integers.	1.7.2.3.2.3
Reserved	3	This field shall contain all 'zeros'.	
Bit Depth	4	The bit depth of the input image encoded mod 2^4 as a 4-bit unsigned binary integer.	1.7.2.3.2.1
Output Word Size	3	The value W encoded mod 2^3 as a 3-bit unsigned binary integer.	5.2.2
Reserved	3	This field shall contain all 'zeros'.	
Spectral Transform	2	This field indicates that the input image has been transformed by one of the following spectral transforms: '00' for the Identity Transform, '01' for the Integer Wavelet Transform, '10' for the Pairwise Orthogonal Transform, '11' for the Arbitrary Affine Transform.	section 4

5.4.4 SIDE INFORMATION

5.4.4.1 General

5.4.4.1.1 When present, the side information header part shall be of variable length and shall encode side information needed to reconstruct the collection as specified in the following subsections.

5.4.4.1.2 The side information part of the header shall be used when the POT or AAT is employed and omitted otherwise.

5.4.4.2 Regions Associated with Coded Collections

5.4.4.2.1 The side information header part is specified in 5.4.4.3 and 5.4.4.4 for the POT and AAT respectively. In either case, side information shall be partitioned into collections by associating regions, as defined in 4.5.6.2 and 4.6.4.1, with collections.

5.4.4.2.2 Each coded collection shall be associated with K regions, except for the last one, which shall be associated with K' regions, where

$$K = \frac{8RS'}{F \cdot \lceil N_X / 8 \rceil}, \text{ and} \tag{96}$$

$$K' = \left(\left\lceil \frac{N_Y}{F} - 1 \right\rceil \bmod K \right) + 1. \tag{97}$$

5.4.4.2.3 The rth region shall be associated with the ith collection of a compressed image, where

$$i = \left\lfloor \frac{r}{K} \right\rfloor, \text{ and} \tag{98}$$

the first collection is indexed by $i = 0$.

NOTES

1 The index r for a region is defined in 4.5.6.2 and 4.6.4.1 for the POT and AAT respectively.

2 The first K regions of an image belong to the first collection, the next K regions of an image belong to the second collection, and so on, with the last collection of an image taking K' regions.

3 A transform region is associated with a collection if the image area covered by the region is part of the image area covered by the collection.

5.4.4.3 POT Side Information

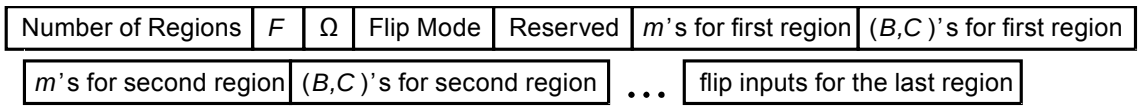


Figure 5-10: Overview of the Side Information Part of Each Collection for the POT

5.4.4.3.1 When the POT is employed, the side information part for each collection shall consist of either two or three consecutive pieces.

5.4.4.3.2 The first piece shall be composed of the fields specified in table 5-3, arranged in the order listed.

Table 5-3: First Piece of the Side Information Part of Each Collection for the POT

Field	Width (bits)	Description	Reference
Number of Regions	16	For all but the last collection, the value of K encoded modulo 2^{16} as a 16-bit unsigned binary integer. For the last collection, the value of K' encoded modulo 2^{16} as a 16-bit unsigned binary integer.	5.4.4.2
Region Height	3	The value $\log_2(F)$ encoded as a 3-bit unsigned binary integer.	4.5.6.2
Arithmetic Precision	3	The value of Ω encoded as the 3-bit binary representation of the value $\Omega - 9$.	4.5.4.2.2
Flip Mode	1	The *flip* mode, encoded using a single bit with value '0' for bypass mode and '1' for stable mode.	4.5.7
Reserved	1	This field shall contain all 'zeros'.	

5.4.4.3.3 The second piece shall be composed of multiple instances of the fields specified in table 5-4, arranged in the order listed.

5.4.4.3.4 There shall be one instance for each region associated with the current collection (as defined in 5.4.4.2), arranged in order of increasing region index r (i.e., from top to bottom as shown in figure 4-11).

Table 5-4: Fields of the Second Piece of the Side Information Part of Each Collection for the POT

Field	Width (bits)	Description	Reference
Nominal Means	(variable)	The sequence of N_Z nominal mean values m associated with mean-subtraction operations, arranged in increasing order of band index z. Each such nominal mean value is encoded as an integer in two's complement representation when the input image is signed, and as an unsigned integer otherwise. In either case, values of m are encoded with a bit depth equal to the bit depth of the input image, I, incremented by U bits, where U is the user-specified parameter defined in 3.2.1.	4.5.3.3
Pairs	(variable)	The $(N_Z - 1)$ pairs of values B and C associated with pairwise operations, arranged in the order associated with their outputs described in 4.5.5.5 (i.e., first the B and C values produced by the pairwise operation that has $R^{(1)}$ as detail output, then the ones associated with $R^{(2)}$, then the ones associated with $R^{(3)}$, etc.). Each pair B, C is jointly encoded by writing the value of $E = 2C + (1-B)/2$, as a two's complement integer using Ω bits. I.e., for each pairwise operation, the $\Omega - 1$ bits of C concatenated with the sign bit of B.	4.5.4.3

5.4.4.3.5 The third piece shall be present only if stable mode is employed and shall be composed of the fields specified in table 5-5, arranged in the order listed.

Table 5-5: Third Piece of the Side Information Part of Each Collection for the POT

Field	Width (bits)	Description	Reference
Last *Flip* Inputs	(variable)	The sequence of $(N_Z - 1)$ values of the *flip* input associated with the pairwise operations of the last region associated with the collection (the region associated with the largest region index r) using the same order used for B and C: first the *flip* value for the pairwise operation that has $R^{(1)}$ as detail output, then the *flip* value for the pairwise operation that has $R^{(2)}$ as detail output, etc. Each value is coded as a 1-bit field, which takes a value of '0' for a *flip* value of 'false' and a value of '1' for a *flip* value of 'true'.	4.5.7.5

5.4.4.4 AAT Side Information

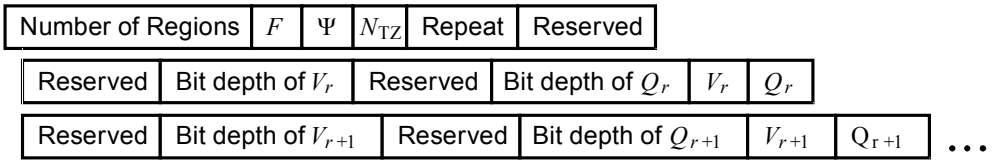


Figure 5-11: Overview of the Side Information Part of Each Collection for the AAT

5.4.4.4.1 When the AAT is employed, the side information part for each collection shall consist of two consecutive pieces.

5.4.4.4.2 The first piece shall be composed by the fields specified in table 5-6, arranged in the order listed.

Table 5-6: First Piece of the Side Information Part of Each Collection for the AAT

Field	Width (bits)	Description	Reference
Number of Regions	16	For all but the last collection, the value of K encoded modulo 2^{16} as a 16-bit unsigned binary integer. For the last collection, the value of K' encoded modulo 2^{16} as a 16-bit unsigned binary integer.	5.4.4.2
Region Height	3	The value $\log_2(F)$ encoded as a 3-bit unsigned binary integer.	4.6.4.1
Scaling Factor	5	The value Ψ as a 5-bit unsigned binary integer.	4.6.3.2.1
Input Z Size	16	The value N_Z encoded mod 2^{16} as a 16-bit unsigned binary integer.	1.7.2.3.1.2
Repeat	1	This field shall be set to '1' when all regions associated with the current collection have the same (Q_r, V_r) values, and shall be set to '0' otherwise.	5.4.4.4.5
Reserved	7	This field shall contain all 'zeros'.	

5.4.4.4.3 When the *repeat* field in the first piece of side information of a collection is set to '0', the second piece shall be composed of multiple instances of the fields specified in table 5-7, arranged in the order listed.

5.4.4.4.4 There shall be one instance for each region associated with the current collection (as defined in 5.4.4.2), arranged in order of increasing region index *r*.

Table 5-7: Fields of the Second Piece of the Side Information Part of Each Collection for the AAT

Field	Width (bits)	Description	Reference
Reserved	3	This field shall contain all 'zeros'.	
Vector Bit Depth	5	The number of bits required to represent one value of the vector V_r as a two's complement integer, encoded mod 2^5 as a 5-bit unsigned binary integer.	
Reserved	3	This field shall contain all 'zeros'.	
Matrix Bit Depth	5	The number of bits required to represent one value of the matrix Q_r as a two's complement integer, encoded mod 2^5 as a 5-bit unsigned binary integer.	
Vector Values	(variable)	The elements $v_0,\ldots, v_{N_Z-1}$ of vector V_r, with each element encoded as an integer in two's complement representation using the bit depth previously stated.	4.6.3.2
Matrix Values	(variable)	The elements of matrix Q_r, each encoded as an integer in two's complement representation using the bit depth previously stated, encoded in raster-scan order, i.e., $q_{(0,0)},q_{(0,1)},\ldots,q_{(0,N_Z-1)},$ $q_{(1,0)},q_{(1,1)},\ldots,q_{(1,N_Z-1)},$ … $q_{(N_{TZ}-1,0)},q_{(N_{TZ}-1,1)},\ldots,q_{(N_{TZ}-1,N_Z-1)}.$	4.6.3.2

5.4.4.4.5 When the *repeat* field in the first piece of side information of a collection is set to '1', the second piece shall be composed of a single instance of the fields specified in table 5-7, which encodes the (Q_r, V_r) pair associated with the first region in the collection.

5.4.5 PADDING

5.4.5.1 The padding header part shall be used if and only if the combined length of all other header parts present is not an integer multiple of the word size W.

5.4.5.2 The length of the padding header part shall be the smallest number of bits needed to produce a complete header with length that is an integer multiple of the word size W.

NOTE – This part has a minimum length of 1 bit and a maximum length of $8 \cdot W - 1$ bits.

5.4.5.3 All bits in the padding header part shall have value 'zero'.

ANNEX A

IMPLEMENTATION CONFORMANCE STATEMENT PROFORMA

(NORMATIVE)

A1 INTRODUCTION

A1.1 OVERVIEW

This annex provides the Protocol Implementation Conformance Statement (PICS) Requirements List (RL) for an implementation of a compressor or decompressor for *Spectral Preprocessing Transform for Multispectral and Hyperspectral Image Compression*, CCSDS 122.1-R-1, March 2017. The PICS for an implementation is generated by completing the RL in accordance with the instructions below. An implementation shall satisfy the mandatory conformance requirements referenced in the RL.

The RL in this annex is blank. An implementation's completed RL is called the PICS. The PICS states which capabilities and options have been implemented. The following can use the PICS:

- the implementer of a compressor or decompressor, as a checklist to reduce the risk of failure to conform to the standard through oversight;
- the supplier and acquirer or potential acquirer of a compressor or decompressor implementation, as a detailed indication of the capabilities of the implementation, stated relative to the common basis for understanding provided by the standard PICS proforma;
- the user or potential user of a compressor or decompressor implementation, as a basis for initially checking the possibility of interoperability between compressor and decompressor implementations;
- a compressor or decompressor implementation tester, as the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation.

A1.2 ABBREVIATIONS AND CONVENTIONS

The RL consists of information in tabular form. The status of features is indicated using the abbreviations and conventions described below.

Item Column

The number in the item column identifies the item in the table.

Description Column

The description column contains a brief description of the item. It implicitly means 'is <item description> supported by the implementation?'

Reference Column

The reference column indicates the relevant subsection of *Spectral Preprocessing Transform for Multispectral and Hyperspectral Image Compression*, CCSDS 122.1-R-1 (this document).

Status Column

The status column uses the following notations:

M mandatory.

O optional.

N/A not applicable.

O.*i* qualified optional—for a group of related optional items labeled by the same numeral *i*, the logic of their selection is defined immediately following the table.

C.*j* conditional—the requirement on the capability ('M', 'O', or 'N/A') depends on the support of another optional item. The numeral *j* identifies a unique conditional status expression defined immediately following the table.

Values Allowed Column

The values allowed column contains the list or range of values allowed. The following notations are used:

range of values:	<min value> .. <max value>
example:	2 .. 16
list of values:	<value1>, <value2>, …, <valueN>
example:	3, 6, 9, …, 21
N/A	not applicable

Item Support or Values Supported Column

In the item support or values supported column, the support of every item as claimed by the implementer shall be stated by entering the appropriate answer ('Y', 'N', or 'N/A') or the values supported:

Y yes, item supported by the implementation.

N no, item not supported by the implementation.

range or list of values supported.

N/A not applicable.

References to Items

The support of an item in the PICS proforma can be referred to by indicating the table and item number separated by a solidus character '/'. For example, 'A-2/8' refers to the support for the 8th item in table A-2.

Prerequisite Line

A prerequisite line takes the form: Prerequisite: <predicate>. A prerequisite line at the top of a table indicates that the table need not be completed if the predicate is 'false'.

A1.3 INSTRUCTIONS FOR COMPLETING THE RL

An implementer shows the extent of compliance to the Recommended Standard by completing the RL; that is, the state of compliance with all mandatory requirements and the options supported are shown. The resulting completed RL is called a PICS. The implementer shall complete the RL by entering appropriate responses in the support or values supported column, using the notation described in A1.2. If a conditional requirement is inapplicable, N/A should be used. If a mandatory requirement is not satisfied, exception information must be supplied by entering a reference X*i*, where *i* is a unique identifier, to an accompanying rationale for the noncompliance.

A2 PICS PROFORMA FOR SPECTRAL PREPROCESSING TRANSFORM FOR MULTISPECTRAL AND HYPERSPECTRAL IMAGE COMPRESSION

A2.1 GENERAL INFORMATION

A2.1.1 Identification of PICS

Date of Statement (DD/MM/YYYY)	
PICS serial number	
System Conformance statement cross-reference	

A2.1.2 Identification of Implementation Under Test (IUT)

Implementation name	
Implementation version	
Function implemented	Compression _____ Decompression _____
Special configuration	
Other information	

A2.1.3 Identification

Supplier	
Contact Point for Queries	
Implementation Name(s) and Versions	
Other information necessary for full identification, e.g., name(s) and version(s) for machines and/or operating systems;	
System Name(s)	

A2.1.4 Document Version Summary

CCSDS 122.1 Document Version	
Addenda Implemented	
Amendments Implemented	
Have any exceptions been required? NOTE – A YES answer means that the implementation does not conform to the Recommended Standard. Non-supported mandatory capabilities are to be identified in the PICS, with an explanation of why the implementation is non-conforming.	Yes____ No____

A2.2 CAPABILITIES

A2.2.1 Image

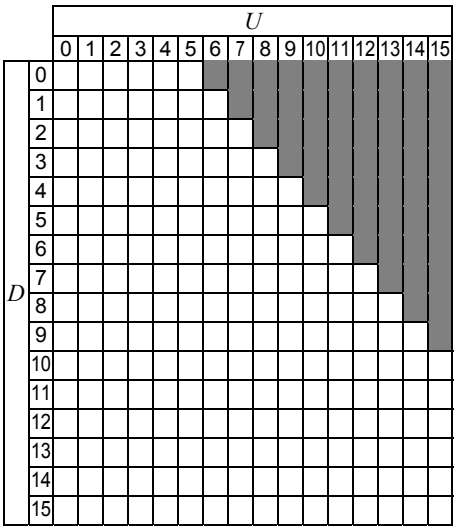
Table A-1: Image Properties

Item	Description	Reference	Status	Values Allowed	Item Support or Values Supported
1	Signed Samples	1.7.2.3.2.3	O.1	N/A	
2	Unsigned Samples	1.7.2.3.2.3	O.1	N/A	
3	N_X	1.7.2.3.1.2, 1.7.2.3.1.3	M	17..2^{16}	
4	N_Y	1.7.2.3.1.2, 1.7.2.3.1.3	M	17.. unlimited	
5	N_Z	1.7.2.3.1.2, 1.7.2.3.1.3	M	1..2^{16}	
6	Bitdepth	1.7.2.3.2.1, 1.7.2.3.2.2	M	1..16	

O.1: It is mandatory to support at least one of these items.

A2.2.2 Upshift and Downshift

Table A-2: Upshift and Downshift Stages

Item	Description	Reference	Status	Values Allowed	Item Support or Values Supported
1	User-specified parameters U and D	3.2.1	M	U, D such that $0 \le U \le 15$, $0 \le D \le 15$, and $U - D \le 5$	
2	Upshift stage	3.2.2	M	N/A	
3	Downshift stage	3.2.3	M	N/A	

Mark supported combinations of user-specified parameters U and D with a cross on the appropriate cell of the provided table.

A2.2.3 Spectral Transform

Table A-3: Supported Spectral Transforms

Item	Description	Reference	Status	Values Allowed	Item Support or Values Supported
1	Identity Transform	4.3	O.2	N/A	
2	Integer Wavelet Transform	4.4	O.2	N/A	
3	Pairwise Orthogonal Transform	4.5	O.2	N/A	
4	Arbitrary Affine Transform	4.6	O.2	N/A	

O.2: It is mandatory to support at least one of these items.

Table A-4: Pairwise Orthogonal Transform Features

Prerequisite: A-3/3 Pairwise Orthogonal Transform					
Item	**Description**	**Reference**	**Status**	**Values Allowed**	**Item Support or Values Supported**
1	Suggested Nominal Mean Calculation	4.5.3.3.2	O	N/A	
2	Arithmetic Precision, Ω	4.5.4.2.2	M	9..16	
3	Suggested Training Parameters Calculation	4.5.4.3.3	O	N/A	
4	Region Size, F	4.5.6.2	M	1, 2, 4, 8, 16, 32	
5	Bypass *Flip* Mode	4.5.7.4	O.3	N/A	
6	Stable *Flip* Mode	4.5.7.5	O.3	N/A	

O.3: It is mandatory to support at least one of these items.

Table A-5: Arbitrary Affine Transform Features

Prerequisite: A-3/4 Arbitrary Affine Transform					
Item	**Description**	**Reference**	**Status**	**Values Allowed**	**Item Support or Values Supported**
1	Number of Transformed Image Bands, N_{TZ}	4.6.2	M	$1..2^{16}$	
2	Range of user-specified parameters Q	4.6.3.2.2	M	$-2^{31} .. 2^{31}-1$	
3	Range of user-specified parameters V	4.6.3.2.2	M	$-2^{31} .. 2^{31}-1$	
4	Arithmetic Precision, Ψ	4.6.3.2.1	M	0..31	
5	Region Size, F	4.6.4.1	M	1, 2, 4, 8, 16, 32	

A2.2.4 Encoder

Table A-6: Encoder Features

Item	Description	Reference	Status	Values Allowed	Item Support or Values Supported
1	Compressed Image	5.2.1.1	M	N/A	
2	Segments per Transformed Band, R	5.2.2	M	$1..2^{16}$	
3	Output Word Size, W	5.2.2	M	1..8	
4	Blocks per Segment, S'	5.2.2	M	$16..2^{20}$	
5	Valid combinations of S' and R	5.2.2.3	M	N/A	
6	Valid values of F given S' and R	5.2.2.3	C.1	N/A	
7	Parameter Mapping	5.3.2	M	N/A	
8	Segment Order	5.3.3	M	N/A	
9	Header	5.4	M	N/A	
10	Mandatory Metadata Part	5.4.2	M	N/A	
11	Setup Metadata Part	5.4.3	O	N/A	
12	POT Side Information Part	5.4.4.3	C.2	N/A	
13	AAT Side Information Part	5.4.4.4	C.3	N/A	
14	Padding Part	5.4.5	M	N/A	

C.1: IF A-3/3 OR A-3/4 — If POT or AAT supported
THEN M — then mandatory
ELSE N/A

C.2: IF A-3/3 — If POT supported
THEN M — then mandatory
ELSE N/A

C.3: IF A-3/4 — If AAT supported
THEN M — then mandatory
ELSE N/A

ANNEX B

SECURITY, SANA, AND PATENT CONSIDERATIONS

(INFORMATIVE)

B1 SECURITY CONSIDERATIONS

B1.1 SECURITY BACKGROUND

It is assumed that security is provided by encryption, authentication methods, and access control to be performed at the application and/or transport layers. Mission and service providers are expected to select from recommended security methods suitable to the specific application profile. Specification of these security methods and other security provisions is outside the scope of this Recommended Standard.

B1.2 SECURITY CONCERNS

Security concerns in the areas of data privacy, integrity, authentication, access control, availability of resources, and auditing are to be addressed in the appropriate layers and are not related to this Recommended Standard. The use of image data compression does not affect the proper functioning of methods used to achieve such protection. The use of image data compression slightly improves data integrity because the alteration of even a single bit of compressed data is likely to cause conspicuous and easily detectible corruption of the reconstructed data, thus making it more likely that malicious data alteration will be detected.

B1.3 POTENTIAL THREATS AND ATTACK SCENARIOS

An eavesdropper will not be able to decompress compressed data if proper encryption is performed at a lower layer.

B1.4 CONSEQUENCES OF NOT APPLYING SECURITY

There are no specific security measures prescribed for compressed data. Therefore consequences of not applying security are only imputable to the lack of proper security measures in other layers.

B2 SANA CONSIDERATIONS

The recommendations of this document do not require any action from SANA.

B3 PATENT CONSIDERATIONS

At time of publication, the specifications of this Recommended Standard are not known to be the subject of patent rights.

ANNEX C

REFERENCES

(INFORMATIVE)

[C1] *Spectral Preprocessing Transform for Multispectral & Hyperspectral Image Compression*. Report Concerning Space Data System Standards (Green Book), CCSDS 120.3-G. Forthcoming.

[C2] *Image Data Compression*. Issue 2. Report Concerning Space Data System Standards (Green Book), CCSDS 120.1-G-2. Washington, D.C.: CCSDS, February 2015.

[C3] *Space Packet Protocol*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 133.0-B-1. Washington, D.C.: CCSDS, September 2003.

[C4] *CCSDS File Delivery Protocol (CFDP)*. Issue 4. Recommendation for Space Data System Standards (Blue Book), CCSDS 727.0-B-4. Washington, D.C.: CCSDS, January 2007.

[C5] *AOS Space Data Link Protocol*. Issue 3. Recommendation for Space Data System Standards (Blue Book), CCSDS 732.0-B-3. Washington, D.C.: CCSDS, September 2015.

[C6] A. Cohen, I. Daubechies, and J.-C. Feauveau. "Biorthogonal Bases of Compactly Supported Wavelets." *Communications on Pure and Applied Mathematics* 45, no. 5 (June 1992): 485–560.

[C7] *Information Technology—JPEG 2000 Image Coding System: Core Coding System*. 2nd ed. International Standard, ISO/IEC 15444-1:2004. Geneva: ISO, 2004.

[C8] Ian Blanes and Joan Serra-Sagristà. "Pairwise Orthogonal Transform for Spectral Image Coding." *IEEE Transactions on Geoscience and Remote Sensing* 49, no. 3 (2011): 961–972.

[C9] Ian Blanes, et al. "Isorange Pairwise Orthogonal Transform." *IEEE Transactions on Geoscience and Remote Sensing* 53, no. 6 (2015): 3361–3372.

[C10] Barbara Penna, et al. "Transform Coding Techniques for Lossy Hyperspectral Data Compression." *IEEE Transactions on Geoscience and Remote Sensing* 45, no. 5 (2007): 1408–1421.

[C11] Ian Blanes, et al. "Divide-and-Conquer Strategies for Hyperspectral Image Processing: A Review of Their Benefits and Advantages." *IEEE Signal Processing Magazine* 29, no. 3 (2012): 71–81.

[C12] Carole Thiebaut, et al. "CNES Studies of On-Board Compression for Multispectral and Hyperspectral Images." In *Satellite Data Compression, Communication, and Processing III*, 668305-1–668305-15. Edited by Roger W. Heymann, Bormin Huang, and Irina Gladkova. SPIE Proceedings vol. 6683. Bellingham, Washington: SPIE, 19 September 2007.

[C13] Michel Barret, et al. "Lossy Hyperspectral Images Coding with Exogenous Quasi Optimal Transforms." In *Proceedings of the 2009 Data Compression Conference*, 411–419. Piscataway, New Jersey: IEEE Conference Publications, 2009.

[C14] *Information Technology—Programming Languages—C*. 3rd ed. International Standard, ISO/IEC 9899. Geneva: ISO, 2011.

[C15] D. Taubman. "High Performance Scalable Image Compression with EBCOT." *IEEE Transactions on Image Processing* 9, no. 7 (2000): 1158–1170.

[C16] F. Auli-Llinas, et al. "Efficient Rate Control for JPEG2000 Coder and Decoder." In *Proceeding of the 2006 Data Compression Conference*, 282–291. Piscataway, New Jersey: IEEE Conference Publications, 2006.

[C17] F. Falzon and S. Mallat. "Low Bit Rate Image Coding over Bases." In *Proceedings of the Fourteenth International Conference on Pattern Recognition*, Vol. 2: 1260–1263. Piscataway, New Jersey: IEEE Conference Publications, 1998.

[C18] S. Mallat and F. Falzon. "Analysis of Low Bit Rate Image Transform Coding." *IEEE Transactions on Signal Processing* 46, no. 4 (1998): 1027–1042.

[C19] O. M. Kosheleva, et al. "Rate Distortion Optimal Bit Allocation Methods for Volumetric Data Using JPEG 2000." *IEEE Transactions on Image Processing* 15, no. 8 (2006): 2106–2112.

[C20] David Taubman and Michael Marcellin. *JPEG2000 Image Compression Fundamentals, Standards and Practice*. The Springer International Series in Engineering and Computer Science, Vol. 642. New York: Springer, 2002.

ANNEX D

TABLES OF SYMBOLS USED

(INFORMATIVE)

This annex tabulates symbols used in this Recommended Standard.

Table D-1: Symbols

Symbol	Meaning	Reference
Images		
I	The input image.	1.7.2.2
$I^{\uparrow}$	The upshifted input image.	1.7.2.2
T	The transformed image.	1.7.2.2
$T^{\downarrow}$	The downshifted transformed image.	1.7.2.2
A	An image that is either I, $I^{\uparrow}$, T, or $T^{\downarrow}$.	1.7.2.2
Image Properties		
N_X, N_Y	The size of an image in the x or y indices.	1.7.2.3.1
N_Z	The size of I and $I^{\uparrow}$ in the z index.	1.7.2.3.1
N_{TZ}	The size of T and $T^{\downarrow}$ in the z index.	1.7.2.3.1
Sequences		
X, Y	Generic sequences of values.	1.7.1.2
x_i, y_i	Elements of the sequences X and Y.	1.7.1.2
L, H	The sequences of low frequency and high frequency coefficients produced by the single-level IWT operation.	4.4.3
l_i, h_i	Elements of the sequences L and H.	4.4.3
χ_i, ϕ_i	Elements of the sequences produced by the principal and detail outputs of a pairwise operation.	4.5.4.2
$I_{\ell}^{(i)}$	Input sequences of an individual-level of a POT multilevel structure.	4.5.5.3
$P_{\ell}^{(i)}$, $D_{\ell}^{(i)}$, U_{ℓ}	Output sequences of an individual-level of a POT multilevel structure.	4.5.5.3
$S^{(i)}$, $R^{(i)}$	Input sequences ($S^{(i)}$) and output sequences ($R^{(i)}$) of a POT multilevel structure.	4.5.5.2
$\Theta_{z,r}^{(A)}$	Sequence of elements from band z of the rth region of image A.	4.5.6.3
$\theta_i^{(A)}$	Elements of the sequence $\Theta_{z,r}^{(A)}$.	4.5.6.3
Pairwise Orthogonal Transform		
m	Nominal mean produced by a mean-subtraction operation.	4.5.3.3

Symbol	Meaning	Reference
Ω	An integer representing the precision of the POT.	4.5.4.2
B	Training parameter of a pairwise operation.	4.5.4.3
C	Training parameter of a pairwise operation.	4.5.4.3
$\tilde{t}$	Intermediate value derived from B and C.	4.5.4.3
$\tilde{p}$	Intermediate value derived from $\tilde{t}$.	4.5.4.3
$\tilde{\sigma}_x^2$, $\tilde{\sigma}_y^2$, $\widetilde{\sigma_{x,y}}$	Scaled variance of the elements x_i, y_i, and scaled covariance between them.	4.5.4.3
$\tilde{w}_1$, $\tilde{w}_2$, $\tilde{w}_3$	The weights employed in the lifting network.	4.5.4.4
s	The conditional sign change in the lifting network.	4.5.4.4
$\Lambda(\cdot)$	Lifting step operation.	4.5.4.5
$\mathcal{L}$	Number of levels in a multilevel structure.	4.5.5.2
Pairwise Orthogonal Transform (continued)		
M_ℓ	Number of input sequences for individual level ℓ.	4.5.5.3
p_ℓ	The number of pairwise operations applied at level ℓ.	4.5.5.3
$b_\ell^{(i)}$	Boolean value associated with sequence $I_\ell^{(i)}$.	4.5.5.3
r	Region index.	4.5.6.2
F	User-specified parameter controlling region size.	4.5.6.2
f_i	Size of the *i*th region along the y index.	4.5.6.2
$flip_{i,\ell}^{(r)}$	*Flip* input to the *i*th pairwise operation of the ℓth level of the multilevel structure applied to region r.	4.5.7
Arbitrary Affine Transform		
r	Region index.	4.6.4.1
Q_r	The *r*th transform matrix for the AAT.	4.6.3.2
V_r	The *r*th transform vector for the AAT.	4.6.3.2
Ψ	Scaling factor that controls the precision of the AAT.	4.6.3.2
F	User-specified parameter controlling region size.	4.6.4.1
f_r	Size of the *r*th region along the y index.	4.6.4.1
Collections		
R, R'	Number of segments from the same band in a collection.	5.2.2
S'	Segment size (in blocks per segment).	5.2.2
K, K'	Number of regions associated with a collection.	5.4.4.2
Other		
U	User-specified upshift parameter.	3.2.2
D	User-specified downshift parameter.	3.2.3
W	The output word size.	5.2.2

ANNEX E

ABBREVIATIONS AND ACRONYMS

(INFORMATIVE)

AAT	Arbitrary Affine Transform
CFDP	CCSDS File Delivery Protocol
IWT	Integer Wavelet Transform
KLT	Karhunen Loève Transform
MSB	Most Significant Bit
MSE	Mean Square Error
PICS	Protocol Implementation Conformance Statement
POT	Pairwise Orthogonal Transform
RL	Requirements List
SANA	Space Assigned Numbers Authority

ANNEX F

REFERENCE RATE-ALLOCATION WEIGHTS

(INFORMATIVE)

F1 GENERAL

Given a limit on compressed data volume for an image, a *rate-allocation* method attempts to allocate compressed data rate within the different compressed data elements of that image, thus encoding some at higher rate than others, with the aim of optimizing overall reconstructed image quality subject to the overall constraint on compressed data volume.

In particular, the problem here is to allocate rate among the transformed spectral bands; compressing all of the transformed spectral bands allocating rate homogeneously generally yields very poor reconstructed image quality.

For rate-allocation algorithms that attempt to minimize the mean square error (MSE) between original image I and reconstructed image $\hat{I}$, which we denote MSE(I, $\hat{I}$), it is common to employ a band-dependent constant multiplicative weight factor that accounts for the specific spectral transform employed.

Such rate-allocation weights can be used to describe an approximate relation between MSE(I, $\hat{I}$) in terms of the MSE between original and reconstructed bands of the transformed image as follows:

$$\mathrm{MSE}(I,\hat{I}) \simeq \frac{1}{N_{\mathrm{TZ}}} \sum_{b} \left(\mathrm{weight}_b \cdot \mathrm{MSE}(T_b, \widehat{T_b}) \right). \tag{99}$$

A straightforward modification of this relationship takes into account the application of the upshift and downshift stages described in section 3:

$$\mathrm{MSE}(I,\hat{I}) \simeq \frac{1}{N_{\mathrm{TZ}}} \sum_{b} \left(\mathrm{weight}_b \cdot \mathrm{MSE}(T^{\downarrow}{}_b, \widehat{T^{\downarrow}{}_b}) \cdot 4^{D-U} \right), \tag{100}$$

provided that D is small enough so that $T^{\downarrow}$ sufficiently resembles T.

Hence, rate allocation methods can operate directly on downshifted transformed bands by employing the SegByteLimit, DCStop, BitPlaneStop, or StageStop fields described in subsection 4.2.3 of reference [1], and by taking into account how quality differences in transformed bands translate to the decoded image.

Similarly, rate-allocation methods based on the variance can be employed as well over the bands of transformed images, by employing

$$\text{weight}_b \cdot \text{variance}\left(T^{\downarrow}{}_b\right) \cdot 4^{D-U}, \tag{101}$$

instead of

$$\text{variance}\left(T^{\downarrow}{}_b\right) \cdot 4^{D-U}, \tag{102}$$

when allocating rate for transformed band $T^{\downarrow}{}_b$.

In the following subsections of this annex reference rate-allocation weights for the IWT and the POT are given. In addition, a few references to rate-allocation methods are provided. There are no reference weights for the AAT, as these weights are specific to each user-defined transform.

Further discussion on rate-allocation methods and details on the weights derivation is provided in reference [C1].

F2 IWT

For an image transformed with the IWT, the following weights should be employed, given as a function written in the C programming language (reference [C14]).

```
#include <stdint.h>

void iwtWeights(int32_t ntz, double weights[]) {
      double L[] = {1.0, 1.5, 2.75, 5.375, 10.6875, 21.34375};
      double H[] = {0.71875, 0.921875, 1.5859375,
            3.04296875, 6.021484375};

      int32_t N = ntz;

      // Fill weights for high frequencies in each level
      int currentLevel;

      for (currentLevel = 0; currentLevel < 5; currentLevel++) {
            int q = N / 2; // High-frequency coefficients in this level

            if (q == 0) break;

            N -= q;

            for (int i = 0; i < q; i++) {
                  weights[N + i] = H[currentLevel];
            }
      }

      // Fill weights for the low-frequency coefficients
      for (int i = 0; i < N; i++) {
            weights[i] = L[currentLevel];
      }
}
```

When employing the previous function, weights obtained for $N_{TZ} \leq 16$ are given in the following table.

Table F-1: IWT Weights

Z Size	Weights
1	1
2	1.5, 0.71875
3	2.75, 0.921875, 0.71875
4	2.75, 0.921875, 0.71875, 0.71875
5	5.375, 1.5859375, 0.921875, 0.71875, 0.71875
6	5.375, 1.5859375, 0.921875, 0.71875, 0.71875, 0.71875
7	5.375, 1.5859375, 0.921875, 0.921875, 0.71875, 0.71875, 0.71875
8	5.375, 1.5859375, 0.921875, 0.921875, 0.71875, 0.71875, 0.71875, 0.71875
9	10.6875, 3.04296875, 1.5859375, 0.921875, 0.921875, 0.71875, 0.71875, 0.71875, 0.71875
10	10.6875, 3.04296875, 1.5859375, 0.921875, 0.921875, 0.71875, 0.71875, 0.71875, 0.71875, 0.71875
11	10.6875, 3.04296875, 1.5859375, 0.921875, 0.921875, 0.921875, 0.71875, 0.71875, 0.71875, 0.71875, 0.71875
12	10.6875, 3.04296875, 1.5859375, 0.921875, 0.921875, 0.921875, 0.71875, 0.71875, 0.71875, 0.71875, 0.71875, 0.71875
13	10.6875, 3.04296875, 1.5859375, 1.5859375, 0.921875, 0.921875, 0.921875, 0.71875, 0.71875, 0.71875, 0.71875, 0.71875, 0.71875
14	10.6875, 3.04296875, 1.5859375, 1.5859375, 0.921875, 0.921875, 0.921875, 0.71875, 0.71875, 0.71875, 0.71875, 0.71875, 0.71875, 0.71875
15	10.6875, 3.04296875, 1.5859375, 1.5859375, 0.921875, 0.921875, 0.921875, 0.921875, 0.71875, 0.71875, 0.71875, 0.71875, 0.71875, 0.71875, 0.71875
16	10.6875, 3.04296875, 1.5859375, 1.5859375, 0.921875, 0.921875, 0.921875, 0.921875, 0.71875, 0.71875, 0.71875, 0.71875, 0.71875, 0.71875, 0.71875, 0.71875

NOTE – IWT weights have finite binary representations.

F3 POT

For an image transformed with the POT, the following weights should be employed, given as a function written in the C programming language (reference [C14]).

```
#include <stdint.h>
#include <cmath.h>

void potWeights(int32_t ntz, double weights[]) {
   int32_t index = ntz - 1;

   // Calculate detail outputs for each level
   int16_t currentLevel = 0;
   int32_t bandsInThisLevel = ntz;
   int32_t bandsInThePreviousLevel = 0;

   // For each level with more than one band...
   while (bandsInThisLevel > 1) {
      int32_t operationsInThisLevel = bandsInThisLevel / 2;

      for (int32_t j = 0;  j < operationsInThisLevel;  j++) {

        // If there was an unpaired band and this is the
        // unbalanced operation that deals with it...
        if (bandsInThePreviousLevel % 2 == 1
          && ((j == 0 && currentLevel % 2 == 1)
          || (j == operationsInThisLevel - 1 && currentLevel % 2 === 0))) {

        // This is an unbalanced pairwise operation
        weights[index--] = pow(2, currentLevel - 2);
      } else {

        // This is a regular pairwise operation
        weights[index--] = pow(2, currentLevel - 1);
      }
   }

     // Set up next level
     bandsInThePreviousLevel = bandsInThisLevel;
     bandsInThisLevel = bandsInThisLevel / 2 + bandsInThisLevel % 2;
     currentLevel++;
   }

   // The principal output of the topmost operation
   weights[0] = pow(2, currentLevel);
}
```

When employing the previous function, weights obtained for $N_{TZ} \leq 16$ are given in the following table.

Table F-2: POT Weights

Z Size	Weights
1	1
2	2, 0.5
3	4, 0.5, 0.5
4	4, 1, 0.5, 0.5
5	8, 1, 0.5, 0.5, 0.5
6	8, 1, 1, 0.5, 0.5, 0.5
7	8, 2, 1, 0.5, 0.5, 0.5, 0.5
8	8, 2, 1, 1, 0.5, 0.5, 0.5, 0.5
9	16, 2, 1, 1, 0.5, 0.5, 0.5, 0.5, 0.5
10	16, 2, 1, 1, 1, 0.5, 0.5, 0.5, 0.5, 0.5
11	16, 2, 2, 1, 1, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5
12	16, 2, 2, 1, 1, 1, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5
13	16, 4, 1, 2, 1, 1, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5
14	16, 4, 1, 2, 1, 1, 1, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5
15	16, 4, 2, 2, 1, 1, 1, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5
16	16, 4, 2, 2, 1, 1, 1, 1, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5

NOTE – When multiplied by two, the weights for the POT are integer values.

F4 RATE ALLOCATION

F4.1 GENERAL

A number of rate optimization algorithms that vary in complexity and performance are cited below. These algorithms may be employed or modified to allocate rate among spectral transformed images, coded segments or subsets of coded segments. When and how often rate allocation algorithms are applied is also application dependent.

F4.2 POST-COMPRESSION RATE ALLOCATION

With a Post-Compression Rate allocation method, an image is initially compressed without any rate restriction and rate allocation is subsequently applied by discarding parts of the original compressed file (reference [C15]). This approach obtains a quasi-optimal rate allocation, but requires that the encoder provides for each segment a set of truncation points and distortion measures at these points. A convex hull algorithm and a Lagrange multiplier are employed to select the quasi-optimal rate for each segment. If distortion measures are not available, truncation points marking the end of each coding stage may be employed to simulate the interleaving of coding passes as in reference [C16].

F4.3 MALLAT-FALZON MODEL

Mallat and Falzon described an accurate relation between rate and distortion that can be employed for rate allocation (references [C17] and [C18]). The relation varies for each image and requires non-trivial training. An application to three-dimensional data is provided in reference [C19].

F4.4 REVERSE WATER FILLING

The Reverse Water Filling method provides a model between rate and distortion based on the variance of the original data. It is a simple model that obtains modest results. The description of an elementary reverse water fill method follows.

Under the reverse waterfilling method, given *TotalRate* bits per pixel per band to split among *N* transformed coded segments (discounting headers), for each coded segment *i* from 1 to *N*, rate is allocated according to:

$$\text{SegmentRate}_i = \text{TotalRate} + \frac{1}{2}\log_2\left(\sigma_i^2 \Big/ \sqrt[N]{\prod_j \sigma_j^2}\right), \tag{103}$$

where σ_i is a variance measure associated with segment *i*, such as the variance of the wavelet coefficients that compose a segment, or the variance of the equivalent region prior to the application of the spatial wavelet transform. Note that SegmentRate_i and TotalRate are given in bits per pixel per band.

A comprehensive justification of this formula is provided in reference [C20].

NOTES

1 Variances can be measured before or after the bi-dimensional wavelet transform defined in section 3 of reference [1] is applied. Measuring variances after it is applied may provide better performance.

2 Variances need to be properly scaled by using the appropriate weights described in the previous sections of this annex. For example by

$$\sigma_i = \sigma_i' \cdot \text{weights}_b \cdot 4^{D-U}, \tag{104}$$

where σ_i' is the measured variance.

3 Under common circumstances, the Reverse Water Filling method may produce negative rates for segments where no rate should be allocated, virtually increasing the total amount of rate allocated. It is often a sufficient solution to proportionally reduce the rate of segments with positive rates.

ANNEX G

REFERENCE IMPLEMENTATION OF MATHEMATICAL OPERATIONS

(INFORMATIVE)

This annex includes reference implementations in the C programming language (reference [C14]) of some of the mathematical operations described in this document.

– An example follows for $\left\lfloor \frac{a}{2^b} \right\rfloor$, with a and b integers, and $b \geq 0$. This example is suitable for equations (22), (23), (25), (26), (39), (45), (47), (48), (49), (50), (55), and (65).

```
#include <stdint.h>

int64_t iDivPowTwoFloor(int64_t a, int b) {
    return a >> b;
}
```

– Two examples follow for $\left\lfloor \frac{a}{b} \right\rfloor$, with a and b integers. These examples are suitable for equations (34), (36), (38), and (39).

Example 1

This example performs an integer division and ensures negative results are properly rounded towards negative infinity.

```
#include <stdint.h>

int64_t iDivFloor1(int64_t a, int64_t b) {
    int64_t q = a / b;
    int64_t r = a % b;

    if (r != 0 && (r < 0) != (b < 0))
        q--;

    return q;
}
```

Example 2

This example takes advantage of the guarantees on the precision of IEEE 754 operations by setting the floating point rounding mode. This example and the previous one provide identical results.

```
#include <stdint.h>
#include <math.h>
#include <fenv.h>

int64_t iDivFloor2(int64_t a, int64_t b) {
    int oldRoundingMode = fegetround();

    if (fesetround(FE_DOWNWARD))
        abort();

    int64_t r = floor(a / (double)b);
    fesetround(oldRoundingMode);

    return r;
}
```

– Two examples follow for $\left\lfloor \frac{a}{b} + \frac{1}{2} \right\rfloor$, with a and b integers. These examples are suitable for equations (41), (44), (46), (47), and (49).

Example 1

This example calculates the integer division with one extra bit of precision and takes into account the $+\frac{1}{2}$ before the floor operation. Proper rounding is ensured as in `iDivFloor1`.

```
#include <stdint.h>

int64_t iDivHalfFloor1(int64_t a, int64_t b) {
    int64_t e = a << 1;

    int64_t q = e / b;
    int64_t r = e % b;

    if (r != 0 && (r < 0) != (b < 0))
        q--;

    return (q + 1) >> 1;
}
```

Example 2

In a similar approach as in `iDivFloor2`, this example takes advantage of the guarantees on the precision of IEEE 754 operations by setting the floating point rounding mode. This example and the previous one provide identical results.

```
#include <stdint.h>
#include <math.h>
#include <fenv.h>

int64_t iDivHalfFloor2(int64_t a, int64_t b) {
    int oldRoundingMode = fegetround();

    if (fesetround(FE_DOWNWARD))
        abort();

    int64_t r = floor(a / (double)b + 0.5f);

    fesetround(oldRoundingMode);

    return r;
}
```

– Two examples follow for $\left\lfloor \sqrt{a} + \frac{1}{2} \right\rfloor$, with a an integer. These examples are suitable for equations (41), (42), and (43).

Example 1

This example calculates the integer square root with one extra bit of precision and then properly rounds the result.

```
#include <stdint.h>

uint64_t iSqrtHalfFloor1(uint64_t a) {
    uint64_t b = a << 2;

    uint64_t root = 0;
    int i;

    for (i = 31; i >= 0; i--) {
        uint64_t t = (root + (1 << i)) << i;

        if (b >= t) {
            b -= t;
            root |= 2 << i;
        }
    }

    return (root + 2) >> 2;
}
```

Example 2

As in previous examples, this example takes advantage of the guarantees on the precision of IEEE 754 operations by setting the floating point rounding mode. This example and the previous one provide identical results.

```
#include <stdint.h>
#include <math.h>
#include <fenv.h>

uint64_t iSqrtHalfFloor2(uint64_t a) {
    int oldRoundingMode = fegetround();

    if (fesetround(FE_DOWNWARD))
        abort();

    double sqrtResultFloat = floor(sqrt(a) + 0.5);

    fesetround(oldRoundingMode);

    return (uint64_t) sqrtResultFloat;
}
```