

INTERNATIONAL
STANDARD

ISO/IEC
11544

First edition
1993-12-15

**Information technology — Coded
representation of picture and audio
information — Progressive bi-level image
compression**

*Technologies de l'information — Représentation codée de l'image et du
son — Compression de l'image progressive à deux niveaux*



Reference number
ISO/IEC 11544-1993(E)

AVA Documents
Document Sharing Hub

Contents	Page
1 Scope.....	1
2 Normative references.....	1
3 Definitions.....	1
4 Symbols and abbreviations	2
4.1 Acronyms.....	2
4.2 Symbolic constants	2
4.3 Mathematical symbols, operators, and indicators	3
4.4 Variables with mnemonic names.....	3
5 Conventions.....	3
5.1 Flow diagram conventions and symbols	3
5.2 Template graphics.....	3
5.3 Spatial phase.....	4
5.4 Data structure graphics.....	4
6 Requirements.....	7
6.1 General rules.....	7
6.2 Data organization.....	7
6.3 Resolution reduction	12
6.4 Differential-layer typical prediction	13
6.5 Lowest-resolution-layer typical prediction	16
6.6 Deterministic prediction (DP)	19
6.7 Model templates and adaptive templates	23
6.8 Arithmetic coding	26
7 Test methods and datastream examples	43
7.1 Arithmetic coding	44
7.2 Parameterized algorithm	51
7.3 Datastream examples	55

© ISO/IEC 1993
All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.
ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland
Printed in Switzerland

Annexes

A	Suggested minimum support for free parameters	56
B	Design of the resolution reduction table	57
	B.1 Filtering	57
	B.2 Exceptions	58
C	Adaptive template changes	60
	C.1 General	60
	C.2 Differential layers	60
	C.3 Lowest resolution layer	60
D	Design of the probability-estimation table	63
	D.1 Bayesian estimation	63
	D.2 Multiple contexts	63
	D.3 MPS/LPS parameterization	63
	D.4 Rapid tracking	64
	D.5 Reducing computational burden	64
E	Patents	68
	E.1 Introductory remarks	68
	E.2 List of patents	68
	E.3 Contact addresses for patent information	69
F	Bibliography	71

Foreword

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

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 11544 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*, in collaboration with CCITT. The identical text is published as CCITT Recommendation T.82.

Annexes A, B, C, D, E and F of this International Standard are for information only.

Patents

During the preparation of this International Standard, information was gathered on patents upon which application of the standard might depend. Relevant patents were identified as belonging to the patent holders listed in annex E. However, ISO/IEC cannot give authoritative or comprehensive information about evidence, validity or scope of patent and like rights. The patent holders have stated that licences will be granted under reasonable terms. Communications on this subject should be addressed to the patent holders (see annex E).

Introduction

This Recommendation | International Standard was prepared by the Joint Bi-level Image experts Group (JBIG) of ISO/IEC JTC1/SC29/WG9 and CCITT SGVIII. The JBIG experts group was formed in 1988 to establish a standard for the progressive encoding of bi-level images.

A progressive encoding system transmits a compressed image by first sending the compressed data for a reduced-resolution version of the image and then enhancing it as needed by transmitting additional compressed data, which builds on that already transmitted. This Recommendation | International Standard defines a coding method having progressive, progressive-compatible sequential, and single-progression sequential modes and suggests a method to obtain any needed low-resolution renditions. It has been found possible to effectively use the defined coding and resolution-reduction algorithms for the lossless coding of greyscale and color images as well as bi-level images.

0.1 General characteristics

This Specification defines a method for lossless compression encoding of a bi-level image (that is, an image that, like a black-and-white image, has only two colors). The defined method can also be used for coding greyscale and color images. Being adaptive to image characteristics, it is robust over image type. On scanned images of printed characters, observed compression ratios have been from 1,1 to 1,5 times as great as those achieved by the MMR encoding algorithm (which is less complex) described in Recommendations T.4 (G3) and T.6 (G4). On computer generated images of printed characters, observed compression ratios have been as much as 5 times as great. On images with greyscale rendered by halftoning or dithering, observed compression ratios have been from 2 to 30 times as great.

The method is bit-preserving, which means that it, like Recommendations T.4 and T.6, is distortionless and that the final decoded image is identical to the original.

The method also has “progressive” capability. When decoding a progressively coded image, a low-resolution rendition of the original image is made available first with subsequent doublings of resolution as more data is decoded. Note that resolution reduction is performed from the higher to lower resolution layers, while decoding is performed from the lower to higher resolution layers. The lowest resolution image sent in a progressive sequence is a sequentially coded image. In a single-progression sequential coding application, this is the only image sent.

Progressive encodings have two distinct benefits. One is that with them it is possible to design an application with one common database that can efficiently serve output devices with widely different resolution capabilities. Only that portion of the compressed image file required for reconstruction to the resolution capability of the particular output device has to be sent and decoded. Also, if additional resolution enhancement is desired, for say, a paper copy of an image already on a CRT screen, only the needed resolution-enhancing information has to be sent.

The other benefit of progressive encodings is that they can provide subjectively superior image browsing (on a CRT) for an application using low-rate and medium-rate communication links. A low-resolution rendition is transmitted and displayed rapidly, and then followed by as much resolution enhancement as desired. Each stage of resolution enhancement builds on the image already available. Progressive encoding can make it easier for a user to quickly recognize the image as it is being built up, which in turn allows the user to interrupt the transmission of the image.

Let D denote the number of doublings in resolution (called differential layers) provided by the progressive coding. Let I_D denote the highest resolution image and let its horizontal and vertical dimensions in pixels be X_D and Y_D . Let R_D denote the sampling resolution of the image I_D .

This Specification imposes almost no restrictions on the parameters R_D , X_D , Y_D , or D . Choices such as 400 or 200 dpi (dots-per-inch) for the resolution R_D of the highest resolution layer result in a hierarchy of resolutions commensurate with current facsimile standards. Choosing R_D as 600 or 300 dpi gives a progressive hierarchy more compatible with the laser printer resolutions available as of the writing of this Specification.

It is anticipated that D will typically be chosen so that the lowest resolution is roughly 10 to 25 dpi. Typical bi-level images when reduced to such a resolution are not legible, but nonetheless such low-resolution renditions are still quite useful and function as automatically generated icons. Page layout is usually apparent and recognition of particular pages that have been seen before at higher resolution is often possible.

As mentioned above, this Specification does not restrict the number D of resolution doublings. It can be set to 0 if progressive coding is of no utility, as is the case, for example, in hardcopy facsimile. Doing so retains JBIG's compression advantage over MMR (and in fact usually increases it somewhat), while eliminating the need for any buffering and simplifying the algorithm. Single-progression sequential JBIG coding has potential applications identical to those of MMR coding. Images compressed by a single-progression sequential encoder will be readable by decoders capable of progressive decoding, although only the lowest resolution version of a progressively encoded image will be decodable by a single-progression sequential decoder.

It is possible to use this Specification for the lossless coding of greyscale and color images by coding bit-planes independently as though each were itself a bi-level image. This approach to the coding of greyscale and color images can be used as an alternative to the photographic encoding specification CCITT Rec. T.81/ISO/IEC 10918-1 (JPEG) in its lossless mode. Preliminary experimental results have shown that JBIG has a compression advantage over JPEG in its lossless mode for greyscale images up to 6 bits-per-pixel. For 6 to 8 bits-per-pixel the compression results have been similar for both JBIG and JPEG. This Specification makes provision for images with more than one bit plane, but makes no recommendation on how to map greyscale or color intensities to bit-planes. Experimentally, it has been found that for greyscale images a mapping via Gray-coding of intensity is superior to a mapping via simple weighted-binary coding of intensity.

0.2 Stripes and data ordering

When it is necessary to distinguish progressive coding from the more traditional form of image coding in which the image is coded at full resolution from left to right and top to bottom, this older form of coding will be referred to as "sequential". The advantage of sequential coding over progressive coding is that no page (frame) buffer is required. Progressive coding does require a page buffer at the next-to-highest resolution because lower resolution images are used in coding higher resolution images.

It is possible to create a JBIG datastream with only a lowest resolution layer and this can be named single-progression sequential coding. In such coding, a full-resolution image is coded without reference to any differential resolution layers. The parameters D (mentioned in 0.1) is set equal to zero. It should be noted that in a progressive encoding of an image, the lowest resolution layer is actually encoded in single-progression sequential coding. If a full-resolution image is encoded using single-progression sequential coding, it will not be possible to decode the image progressively.

Coding in the progressive-compatible sequential mode is said to be "compatible" with coding in the progressive mode because the datastreams created (encoder) or read (decoder) in either mode carry exactly the same information. All that changes with a switch from progressive to progressive-compatible sequential encoding is the order in which parts of the compressed data are created by the encoder. All that changes with a switch from progressive to progressive-compatible sequential decoding is the order in which these parts are used by the decoder.

This compatibility is achieved by breaking an image into smaller parts before compression. These parts are created by dividing the image in each of its resolution "layers" into horizontal bands called "stripes." Progressive-compatible sequential coding does require a "stripe" buffer (much smaller than a page buffer) and additional individual "state" memory used for adaptive entropy coding of each resolution layer and bit plane.

Figure Intro. 1 shows such a decomposition when there are three resolution layers, three stripes per layer, and only one bit plane. Table Intro. 1 shows defined ways to sequence through the nine stripes.

Notice that in addition to the progressive-versus-sequential distinction that is carried by the **SEQ** bit, there is also a resolution-order distinction that is carried by the **HITOL** bit. Encoders work from high resolution downward and so most naturally encode the stripes in **HITOL** order. Decoders must build up the image from low resolution and so most naturally process stripes in the opposite order. When an application uses an encoder that sends progressively coded data directly to a decoder, one or the other must buffer to invert the order. When an application includes a database, the database (with appropriate set-up) can be used to buffer and invert the order (including setting **HITOL** correctly) thereby removing this requirement from the encoder and decoder.

A stripe has a vertical size that is typically much smaller than that of the entire image. The number L_0 of lines per stripe at the lowest layer is another free parameter. As an example, L_0 might be chosen so that a stripe is about 8 mm. If such a choice is made, the number S of stripes in an image of a business-letter-sized sheet of paper will be about 35.

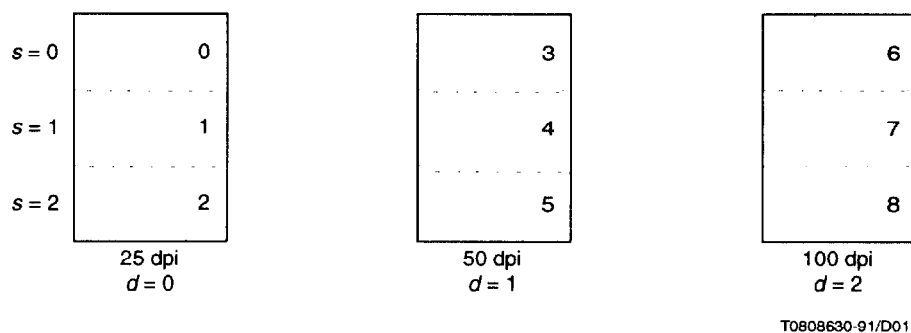


Figure 0.1 – Decomposition in the special case of 3 layers, 3 stripes, and 1 bit plane

Table 0.1 – Possible bi-level data orderings

HITOLO	SEQ	Example order
0	0	0, 1, 2 3, 4, 5 6, 7, 8
0	1	0, 3, 6 1, 4, 7 2, 5, 8
1	0	6, 7, 8 3, 4, 5 0, 1, 2
1	1	6, 3, 0 7, 4, 1 8, 5, 2

When there is more than one bit plane, as in Figure 0.2, there are twelve defined stripe orderings. Table 0.2 lists them. As before, the **HITOLO** bit carries the resolution-order distinction, and the **SEQ** bit carries the progressive-versus-sequential distinction. When the **ILEAVE** bit is 1, it indicates the interleaving of multiple bit planes. When the **SMID** bit is 1, it indicates s , the index over the stripe, is in the middle as shown more clearly in Table 11 of 6.2.4.

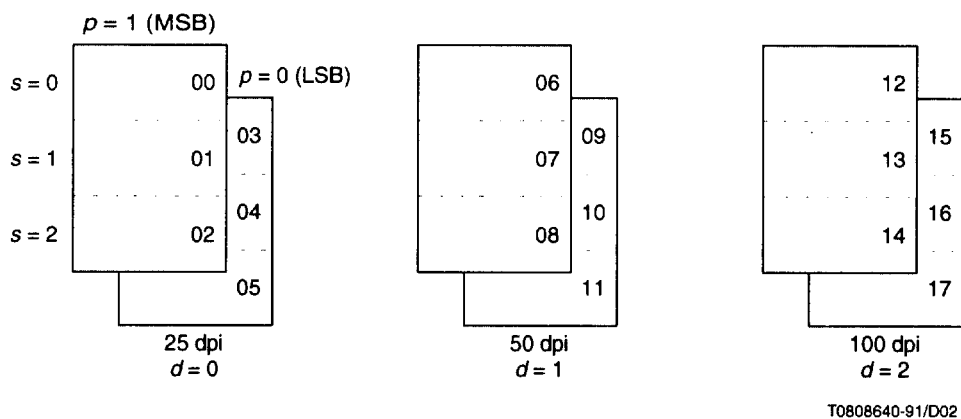


Figure 0.2 – Decomposition in the special case of 3 layers, 3 stripes, and 2 bit planes

Table 0.2 – Possible multi-plane data orderings

HITOLO	SEQ	ILEAVE	SMID	Example order
0	0	0	0	(00,01,02 06,07,08 12,13,14) (03,04,05 09,10,11 15,16,17)
0	0	1	0	(00,01,02 03,04,05) (06,07,08 09,10,11) (12,13,14 15,16,17)
0	0	1	1	(00,03 01,04 02,05) (06,09 07,10 08,11) (12,15 13,16 14,17)
0	1	0	0	(00,06,12 03,09,15) (01,07,13 04,10,16) (02,08,14 05,11,17)
0	1	0	1	(00,06,12 01,07,13 02,08,14) (03,09,15 04,10,16 05,11,17)
0	1	1	0	(00,03 06,09 12,15) (01,04 07,10 13,16) (02,05 08,11 14,17)
1	0	0	0	(12,13,14 06,07,08 00,01,02) (15,16,17 09,10,11 03,04,05)
1	0	1	0	(12,13,14 15,16,17) (06,07,08 09,10,11) (00,01,02 03,04,05)
1	0	1	1	(12,15 13,16 14,17) (06,09 07,10 08,11) (00,03 01,04 02,05)
1	1	0	0	(12,06,00 15,09,03) (13,07,01 16,10,04) (14,08,02 17,11,05)
1	1	0	1	(12,06,00 13,07,01 14,08,02) (15,09,03 16,10,04 17,11,05)
1	1	1	0	(12,15 06,09 00,03) (13,16 07,10 01,04) (14,17 08,11 02,05)

The two new variables **ILEAVE** and **SMID** plus the two earlier variables **HITOLO** and **SEQ** make it possible to index all twelve of these orders. The other four of the sixteen possible combinations for these four binary variables have no stripe ordering associated with them. If there is only one plane, stripe order is not dependent on **ILEAVE** and **SMID** and their values are inconsequential.

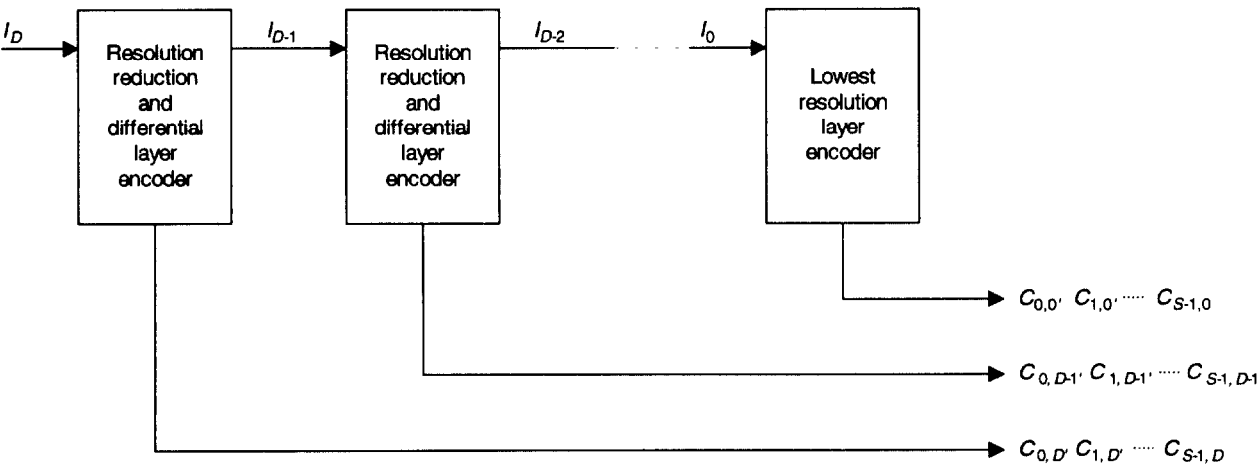
The compressed data $C_{s,d,p}$ for stripe s of resolution layer d of bit-plane p is independent of stripe ordering. All that changes as **HITOLO**, **SEQ**, **ILEAVE** and **SMID** vary is the order in which the data is concatenated onto a datastream. This is the compatibility feature noted earlier.

For simplicity, the remainder of this introduction will assume there is only one bit plane and the subscript p denoting bit plane will be dropped from $C_{s,d,p}$.

0.3 Encoder functional blocks

An encoder can be decomposed as shown in Figure 0.3. (In single-progression sequential coding only the lowest-resolution-layer encoder would be used.)

Although conceptually there are D algorithmically identical differential-layer encoders as shown in Figure 0.3, some implementations may choose to recursively use only one physical differential-layer encoder.



T0806280-90/D03

Figure 0.3 – Decomposition of encoder

0.3.1 Resolution reduction and differential layer encoder

Each of the resolution-reduction-and-differential-layer-encoder blocks of Figure 0.3 is identical in function, hence only a description of the operation at one layer is needed. For such a description there are only two resolution layers involved. For simplicity in the remainder of this subclause, the incoming image will be referred to as the "high-resolution" image and the outgoing image, as the "low-resolution" image. Note though that the "high" and "low" resolution images of any particular resolution-reduction-and-differential-layer-encoder block in Figure 0.3 are not in general the highest and lowest resolution images of the entire system.

A resolution-reduction-and-differential-layer-encoder block of Figure 0.3 can itself be decomposed into sub-blocks as shown in Figure 0.4. Not all sub-blocks need be used in all systems. Refer to the tables in clause 4 for a definition of signal names.

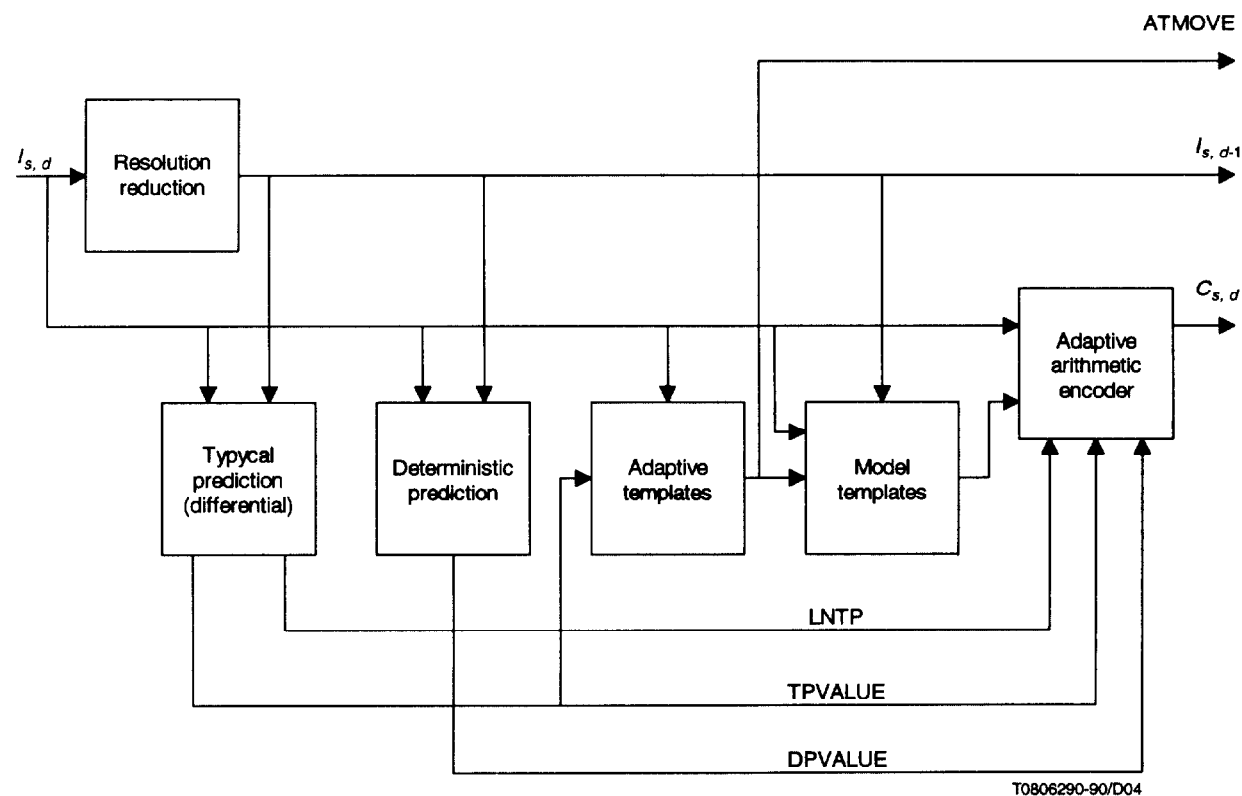


Figure 0.4 – Resolution reduction and differential layer encoder

Acronyms for the processing blocks of this figure and some others to be discussed in this introductory clause are given in Table 0.3.

Table 0.3 – Acronyms for processing blocks

Acronym	Meaning
AAD	Adaptive Arithmetic Decoder
AAE	Adaptive Arithmetic Encoder
AT	Adaptive Templates
DP	Deterministic Prediction
MT	Model Templates
RR	Resolution Reduction
TPB	Typical Prediction (Bottom)
TPD	Typical Prediction (Differential)

0.3.1.1 Resolution reduction

The resolution-reduction (RR) block performs resolution reduction. This block accepts a high-resolution image and creates a low-resolution image with, as nearly as possible, half as many rows and half as many columns as the original.

An obvious way to reduce the resolution of a given image by a factor of two in each dimension is to subsample it by taking every other row and every other column. Subsampling is simple, but creates images of poor subjective quality, especially when the input image is bi-level.

For bi-level images containing text and line drawings, subsampling is poor because it frequently deletes thin lines. For bi-level images that contain halftoning or ordered dithering to render greyscale, subsampling is poor because greyness is not well preserved, especially if the dithering period is a power of two, as is frequently the case.

This Specification suggests a resolution reduction method. This particular method has been carefully designed, extensively tested, and found to achieve excellent results for text, line art, dithered greyscale, halftoned greyscale, and error-diffused greyscale.

0.3.1.2 Differential layer typical prediction

The differential-layer typical prediction (TP) block provides some coding gain, but its primary purpose is to speed implementations. Differential-layer TP looks for regions of solid color and when it finds that a given current high-resolution pixel for coding is in such a region, none of the processing normally done in the DP, AT, MT, and AAE blocks is needed. On text or line-art images, differential-layer TP usually makes it possible to avoid coding over 95% of the pixels. On bi-level images rendering greyscale, processing savings are significantly smaller.

0.3.1.3 Deterministic prediction

The purpose of the deterministic-prediction (DP) block is to provide coding gain. On the set of test images used in the development of this Specification it provided a 7% gain, and such a gain is thought to be typical.

When images are reduced in resolution by a particular resolution reduction algorithm, it sometimes happens that the value of a particular current high-resolution pixel to be coded is inferable from the pixels already known to both the encoder and decoder, that is, all the pixels in the low-resolution image and those in the high-resolution image that are causally related (in a raster sense) to the current pixel. When this occurs, the current pixel is said to be deterministically predictable. The DP block flags any such pixels and inhibits their coding by the arithmetic coder.

DP is a table driven algorithm. The values of particular surrounding pixels in the low-resolution image and causal high-resolution image are used to index into a table to check for determinicity and, when it is present, obtain the deterministic prediction. DP tables are highly dependent on the particular resolution reduction method used. Provision is made for an encoder to download DP tables to a decoder if it is using a private resolution reduction algorithm. If an application requires default DP, decoders need to always have the default DP tables and no DP tables need be sent. Hence, if the suggested resolution reduction algorithm is used, no DP table need ever be sent.

0.3.1.4 Model templates

For each high-resolution pixel to be coded, the model-templates (MT) block provides the arithmetic coder with an integer called the context. This integer is determined by the colors (binary levels) of particular pixels in the causal high-resolution image, by particular pixels in the already available low-resolution image, and by the spatial phase of the pixel being coded. "Spatial phase" describes the orientation of the high-resolution pixel with respect to its corresponding low-resolution pixel.

The arithmetic coder maintains for each context an estimate of the conditional probability of the symbol given that context. The greatest coding gain is achieved when this probability estimate is both accurate and close to 0 or 1. Thus, good templates have good predictive value so that when the values of the pixels in the template are known, the value of the pixel to be coded is highly predictable.

0.3.1.5 Adaptive templates

The adaptive-templates (AT) block provides substantial coding gain (sometimes as much as 80%) on images rendering greyscale with halftoning. AT looks for periodicity in the image and on finding it changes the template so that the pixel preceding the current pixel by this periodicity is incorporated into the template. Such a pixel has excellent predictive value.

Such changes are infrequent, and when one occurs, a control sequence (indicated symbolically by **ATMOVE** in Figure 0.4) is added to the output datastream. Hence, decoders need not do any processing to search for the correct setting for AT.

0.3.1.6 Adaptive arithmetic encoder

The adaptive-arithmetic-encoder (AAE) block is an entropy coder. It notes the outputs of the TP and DP blocks to determine if it is even necessary to code a given pixel. Assuming it is, it then notes the context and uses its internal probability estimator to estimate the conditional probability that the current pixel will be a given color. Often the pixel is highly predictable from the context so that the conditional probability is very close to 0 or 1 and a large entropy coding gain can be realized.

Maintaining probability estimates for each of the contexts is a non-trivial statistical problem. A balance must be struck between obtaining extremely accurate estimates and the conflicting need of adapting quickly to changing underlying statistics.

0.3.2 Lowest resolution layer encoder

Figure 0.5 shows a lowest-resolution-layer encoder. It is conceptually simpler than the differential-layer encoder because the RR and DP blocks are not applicable and the PT, AT, and MT blocks are different since there is no lower resolution layer to be used as input. Refer to the tables in clause 4 for a definition of signal names. (Not all sub-blocks need to be used in all systems.)

Lowest-resolution-layer TP like differential-layer TP is primarily intended to speed processing. The algorithms used for the two versions of TP are quite different, however, and it is not possible to skip as high a percentage of pixels with lowest-resolution-layer TP as it is with differential-layer TP. On images with text and line art, lowest-resolution-layer TP allows skipping about 40% of the pixels.

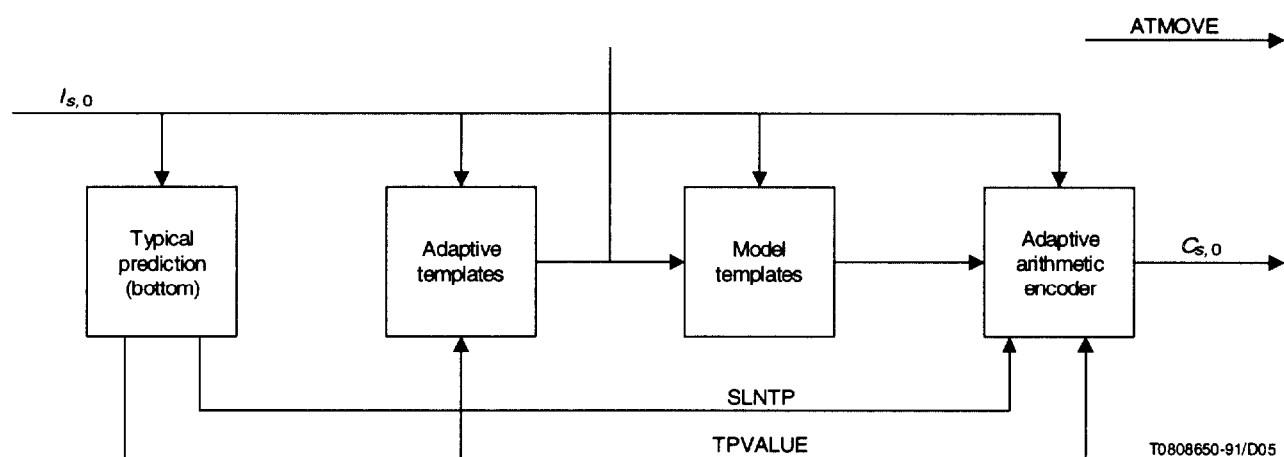


Figure 0.5 – Lowest-resolution-layer encoder

0.4 Decoder functional blocks

Figures 0.6, 0.7 and 0.8 are analogous to Figures 0.3, 0.4 and 0.5 but show decoding rather than encoding. Note that the RR and AT blocks do not appear in the decoder. Refer to the tables in clause 4 for a definition of signal names. In single-progression sequential coding only lowest-resolution-layer-decoder block of Figure 0.6 would be used. Not all sub-blocks in Figures 0.7 and 0.8 need be used in all systems.

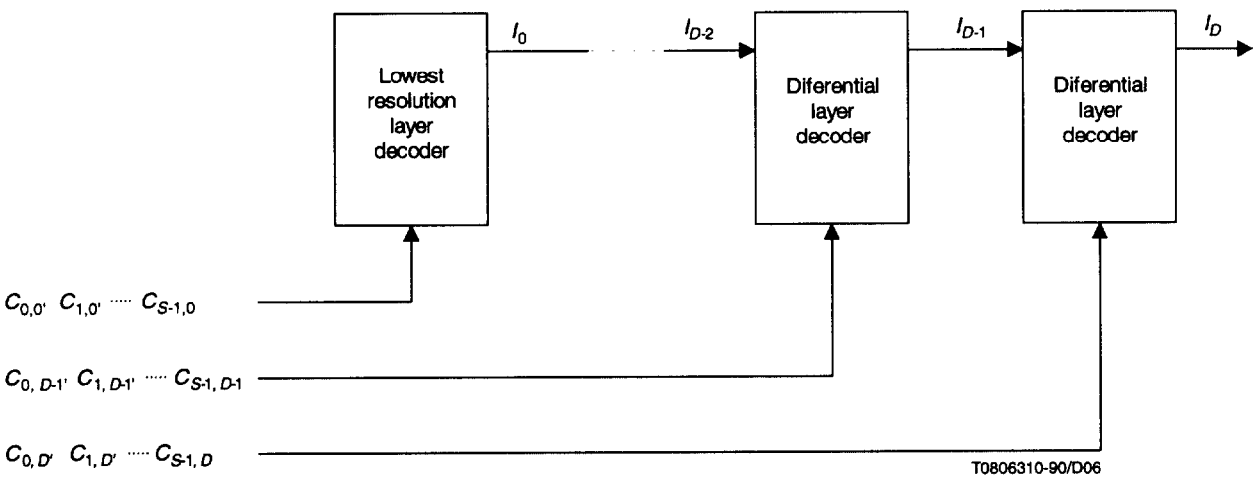


Figure 0.6 – Decomposition of decoder

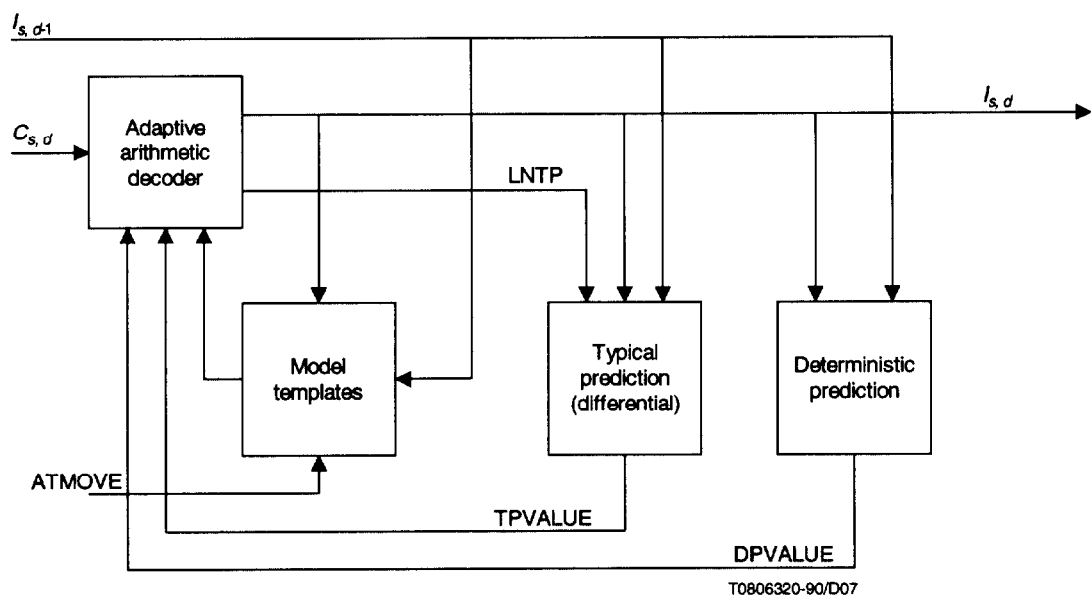


Figure 0.7 – Differential layer decoder

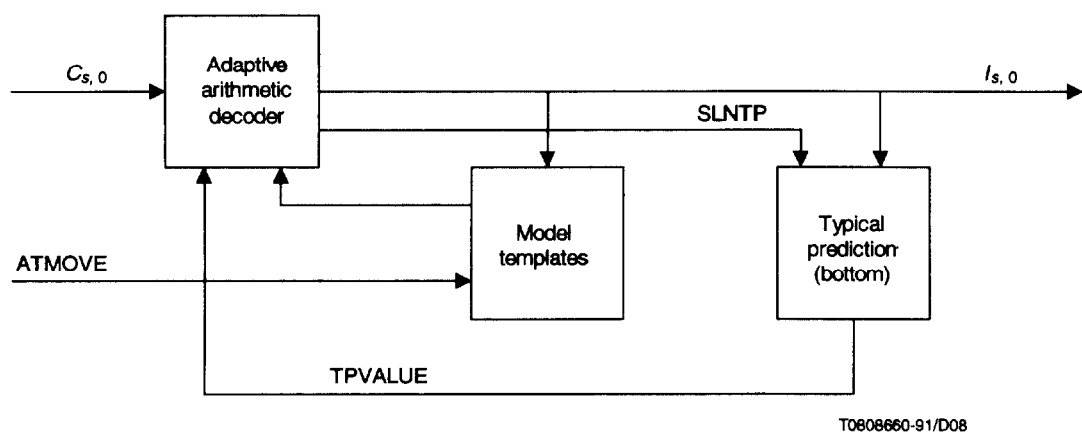


Figure 0.8 – Lowest-resolution-layer decoder

INTERNATIONAL STANDARD

CCITT RECOMMENDATION

INFORMATION TECHNOLOGY – CODED REPRESENTATION OF PICTURE AND AUDIO INFORMATION – PROGRESSIVE BI-LEVEL IMAGE COMPRESSION

1 Scope

This Recommendation | International Standard defines a bit-preserving (lossless) compression method for coding image bit-planes and is particularly suitable for bi-level (two-tone, including black-white) images.

2 Normative references

There are no normative references. Informative references to standards and to the technical literature are listed in Annex F.

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply.

- 3.1 adaptive arithmetic coder:** A mechanism for adaptively compressing or decompressing data by using observed data characteristics to predict and code future data symbols.
- 3.2 adaptive templates (AT):** Model templates which can be modified by moving an AT pixel during the processing of an image to take advantage of observed patterns in the image.
- 3.3 AT lag:** The distance in pixels between the pixel being encoded and the AT pixel.
- 3.4 AT pixel:** A special pixel in the model template that is allowed to adaptively change its location during the processing of an image.
- 3.5 bit-plane:** An array (or “plane”) of bi-level symbols constructed from an image by choosing a particular bit from each pixel.
- 3.6 bit-plane interleaving:** A method used for mixing together two or more bit-planes of data into a single stream.
- 3.7 byte:** Eight bits of data.
- 3.8 byte stuffing:** A mechanism for unambiguously distinguishing between predefined escape bytes indicating the start of a marker segment and bytes identical to the escape byte which naturally occur in a compressed datastream.
- 3.9 context:** An integer corresponding to the specific pattern of the template and spatial phase (if needed) that is used to identify the index of the state of the adaptive arithmetic coder to be used for coding the current pixel.
- 3.10 deterministic prediction (DP):** A method for exactly predicting (and therefore not coding) individual pixels in an image by using a lower resolution version of the same image along with very specific knowledge of the method of resolution reduction used.
- 3.11 differential layer coder:** A mechanism for encoding or decoding differential-layer images.
- 3.12 differential-layer image:** An image at a given resolution which is described by making reference to pixels in a lower-resolution image.
- 3.13 entropy coder:** Any lossless method for compressing or decompressing data.
- 3.14 escape byte:** A byte in a datastream signifying that information to follow has special marker-code meaning.
- 3.15 high-resolution pixel:** A pixel from the higher resolution image of the two resolution layers under discussion.
- 3.16 line not typical (LNTF):** A condition which occurs during typical prediction when one or more of the pixels associated with a given low-resolution line would be predicted incorrectly.
- 3.17 lowest-resolution-layer coder:** A mechanism for encoding or decoding lowest-resolution-layer images.

- 3.18 lowest-resolution-layer image:** An image at a given resolution which is described without reference to any lower resolution images.
- 3.19 low-resolution pixel:** A pixel from the lower resolution image of the two resolution layers under discussion.
- 3.20 marker:** A combination of an escape byte and a marker byte that introduces control information.
- 3.21 marker byte:** A byte immediately following an escape byte that defines the type of control information being introduced.
- 3.22 marker segment:** The combination of a marker and any additional bytes of associated control information.
- 3.23 model template (MT):** A geometric pattern describing the location of pixels relative to a pixel to be coded. It is used to model local image characteristics.
- 3.24 pixel:** One picture element of an image which is described by a rectangular array of such elements.
- 3.25 progressive behavior:** A coding technique shows progressive behavior if an image is first coded as a lowest resolution layer image and then is successively increased in resolution by means of differential layer images.
- 3.26 progressive coding:** A method of coding an image in which the image may be segmented into stripes, and then the entire image is first coded as a lowest resolution layer image and then is successively increased in resolution by means of differential layer images. This is compatible, by stripe/layer data reordering, with progressive-compatible sequential coding.
- 3.27 progressive-compatible sequential coding:** A method of coding an image in which the image may be segmented into stripes, the image stripes are coded in sequence, and within each stripe the image is coded to full resolution progressively. This is compatible, by stripe/layer data reordering with progressive coding.
- 3.28 protected stripe coded data (PSCD):** A compressed image datastream which has been modified by stuffing bytes to distinguish between predefined escape bytes which signal special marker segments (which are not a part of the compressed datastream) and bytes identical to the escape byte which occur naturally in the compressed datastream.
- 3.29 resolution reduction method (RR):** A method for transforming an image with a particular resolution into an image describing the same subject, but with a lower resolution.
- 3.30 sequential behavior:** A coding technique shows sequential behavior if portions of the image near the top are completely described before portions below have been described at all.
- 3.31 single-progression sequential coding:** A method of coding an image such that the image is fully coded in a single resolution layer, line by line, from left to right and top to bottom, without reference to any lower resolution images. This is compatible with progressive coding and progressive-compatible sequential coding if the number of differential layers is zero.
- 3.32 spatial phase:** An attribute of a pixel in a differential-layer image that describes its orientation with respect to the low-resolution pixel associated with it.
- 3.33 spatial resolution:** The number of pixels used to describe a region of an image of fixed spatial size.
- 3.34 stripe:** A fixed vertical size region of an image which encompasses the entire horizontal width of that image.
- 3.35 target pixel:** A pixel to be processed.
- 3.36 typical prediction (TP):** A method for exactly predicting (and therefore not coding) blocks of pixels in an image by exploiting large regions of solid color.

4 Symbols and abbreviations

4.1 Acronyms

See Table 1.

4.2 Symbolic constants

See Table 2.

Table 1 – Acronyms

Acronym	Meaning
AT	Adaptive templates
DP	Deterministic prediction
LPS	Less probable symbol
LSB	Least significant bit
MPS	More probable symbol
MSB	Most significant bit
MT	Model templates
RR	Resolution reduction
TP	Typical prediction

Table 2 – Symbolic constants

		Value	
Constant	Meaning	ISO	Hexadecimal
ABORT	Abort	00/04	0x04
ATMOVE	AT movement	00/06	0x06
COMMENT	Private comment	00/07	0x07
ESC	Escape	15/15	0xff
NEWLEN	New length	00/05	0x05
RESERVE	Reserve	00/01	0x01
SDNORM	Normal stripe data end	00/02	0x02
SDRST	Reset at stripe data end	00/03	0x03
STUFF	Stuff	00/00	0x00

4.3 Mathematical symbols, operators, and indicators

See Table 3.

4.4 Variables with mnemonic names

See Table 4.

5 Conventions

5.1 Flow diagram conventions and symbols

All flow diagrams are entered at the top and exited at the bottom. The symbol “<<” denotes a binary left shift with zero fill of low order bits and the symbol “>>” denotes a binary right shift with zero fill of high order bits. For both “<<” and “>>” the quantity on the left is the quantity being shifted and the quantity on the right is the shift amount. The binary logical AND of two numbers is indicated by “&”.

5.2 Template graphics

It will frequently be necessary to show graphically the relationship of pixels in a high-resolution image to pixels in a low-resolution image. Figure 1 is a three-dimensional graphic showing such a relationship. In the text of this Specification two dimensional drawings like that in Figure 2 are used instead since they are more compact than their three-dimensional equivalent. Note that in the two-dimensional graphic low-resolution pixels are shown as circles and that the corresponding high-resolution pixels are shown as squares, partially hidden by the low-resolution circles.

Table 3 – Mathematical symbols, operators, and indicators

Notation	Meaning
$C_{s,d}$	Coded data for stripe s and layer d
D	Number for final resolution layer in BIE
D_L	Number for initial resolution layer in BIE
h	High-resolution pixel
I_d	Image at layer d
l	Low-resolution pixel
L_c	Length in bytes of private comment
L_d	Lines per stripe at layer d
M_X	Maximum horizontal offset allowed for AT pixel
M_Y	Maximum vertical offset allowed for AT pixel
p	Probability
P	Number of bit-planes
R_d	Resolution at layer d
S	Number of stripes
X_d	Horizontal image size at layer d
y_{AT}	Line in which an AT switch is to be made
Y_d	Vertical image size at layer d
τ_X	Horizontal offset of the AT pixel
τ_Y	Vertical offset of the AT pixel
$>>$	Binary right shift
$<<$	Binary left shift
$\&$	Bitwise AND
$!$	Logical NOT
$\oplus$	Exclusive OR
$\lceil \cdot \rceil$	Ceiling function (smallest integer $\geq$ argument)
0x	Hexadecimal indicator

5.3 Spatial phase

A “spatial phase” can be associated with pixels in any resolution layer other than the lowest. This “phase” describes the orientation of the pixel with respect to the lower resolution pixel associated with it. If it is the upper left hand pixel of the four pixels associated with a single low-resolution pixel, it shall be said to have “phase 0”. Similarly, the upper right pixel shall have “phase 1,” the lower left pixel shall have “phase 2,” and the lower right pixel shall have “phase 3” (see Figure 3).

5.4 Data structure graphics

Tables 5 to 16 contain graphics illustrating the decomposition of fields into sub-fields. Typographically leftmost sub-fields shall be transmitted earlier. The last row of each of these graphics gives field sizes in bytes. An entry of “1/8” denotes a single bit. An entry of “varies” is used when the field size is variable and depends upon options chosen, parameters chosen, or the particular image being coded.

Table 4 – Variables with mnemonic names

Variable	Meaning
A	Interval size register
BID	Bi-level image data
BIE	Bi-level image entity
BIH	Bi-level image header
BUFFER	Buffer
C	Code register
CE	Conditional exchange
CHIGH	Code register, high two bytes
CLOW	Code register, low two bytes
CT	Bit counter
CX	Context
DPLAST	DP last
DPON	DP enabled
DPPRIV	DP private
DPTABLE	DP table
DPVALUE	DP value
HITOLO	High to low
ILEAVE	Interleave multiple bit-planes
LNTP	Line not typical
LPIX	Low-resolution pixel
LRLTWO	Lowest-resolution-layer two-line template
LSZ	LPS size on coding interval
MPS	More probable symbol
NLPS	Next if LPS
NMPS	Next if MPS
PIX	Pixel
PSCD	Protected stripe coded data
SC	Stack (of 0xff bytes) counter
SCD	Stripe coded data
SDE	Stripe data entity
SEQ	Sequential
SLNTP	Same LNTP
SMID	Index over stripe is in middle
SWTCH	Switch
ST	State
TPBON	Lowest-resolution-layer TP enabled
TPDON	Differential-layer TP enabled
TPVALUE	TP value
VLENGTH	Variable length

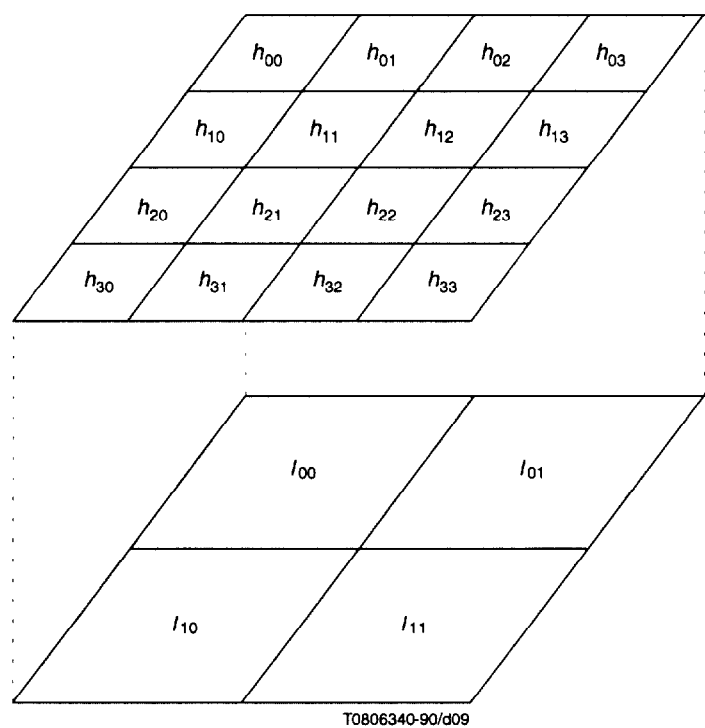


Figure 1 – High- and low-resolution pixels in three-dimensional graphic

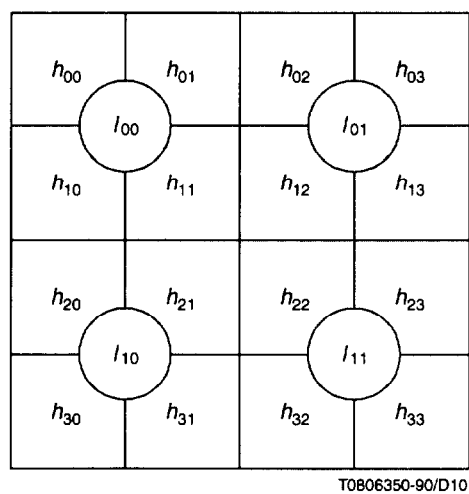


Figure 2 – High- and low-resolution pixels in two-dimensional graphic

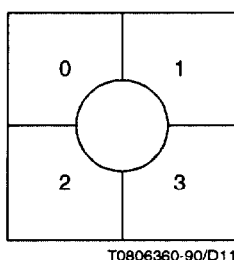
6 Requirements

6.1 General rules

6.1.1 Color assignment

Each bit of each pixel plane is either 0 or 1. When the image is bi-level, a 1 bit shall indicate the foreground color and a 0 bit shall indicate the background color. If there is more than one bit plane, the mapping of intensity and color to bit-planes is not defined by this Specification.

NOTE – Whether 1 or 0 represents the foreground color is inconsequential for all aspects of this Specification except the described resolution reduction method. This resolution reduction method has a slight asymmetry between foreground and background colors.



T0806360-90/D11

Figure 3 – Four possible phases for high-resolution pixels

6.1.2 Edge conventions

The resolution reduction, typical prediction, deterministic prediction, and coding algorithms all iterate through the image in the usual raster scan order, that is, from left to right and top to bottom. The processing for a current target pixel will reference the colors of some pixels in fixed spatial relationship to that target pixel. At image edges, these neighbor references may not lie in the actual image. For both high and low-resolution images, the rules to satisfy off-image references shall be as follows:

- a background colored (0) border shall be assumed to lie to the top, left, and right of the actual image;
- the bottom of the image shall be extended downward as far as necessary by pixel replicating the actual last line of the image.

Furthermore, in referencing pixels across stripe boundaries, the following rules shall be used:

- A pixel reference in a stripe above the current one shall return the actual value of the pixel, unless the pixel is above the image, in which case the background-border-rule for the image top shall be applied.
- A pixel reference in a stripe below the current one shall be satisfied by pixel replicating the last line of the current stripe. In particular, actual values shall not be used even if the reference is still within the image.

NOTE – This latter rule is only of consequence for the low-resolution image, as for decodability there can never be any references to high-resolution pixels in the line below. Also, the described resolution reduction algorithm happens to never reference even low-resolution pixels in the line below.

6.1.3 Byte alignment

NOTE – Because of the header and marker conventions to be described in 6.2, marker segments are always byte aligned in a datastream.

6.2 Data organization

6.2.1 Image decomposition

The highest level data structure described in this Specification is known as a bi-level image entity (**BIE**). A given **BIE** may contain data for one or more resolution layers and bit-planes. The data describing a given image in all its available resolutions and bit-planes may, but need not necessarily be, contained in more than one **BIE**.

NOTE – A multiple **BIE** description of an image is needed when images are first made available at low or intermediate resolution or bit-plane precision and there may or may not be a request for enhancement to higher resolution or precision.

6.2.2 Bi-level image entity and header (**BIE** and **BIH**) decomposition

As shown in Table 5, a bi-level image entity (**BIE**) shall comprise a bi-level image header (**BIH**) and bi-level image data (**BID**).

Table 5 – Decomposition of **BIE**

BIE	
BIH	BID
Varies	Varies

The bi-level image header shall comprise the fields shown in Tables 6, 7 and 8.

Table 6 – Decomposition of **BIH**

BIH											
<i>D_L</i>	<i>D</i>	<i>P</i>	–	<i>X_D</i>	<i>Y_D</i>	<i>L₀</i>	<i>M_X</i>	<i>M_Y</i>	Order	Options	DPTABLE
1	1	1	1	4	4	4	1	1	1	1	0 or 1728

Table 7 – Decomposition of order byte

Order							
MSB	...						LSB
–	–	–	–	HITOLO	SEQ	ILEAVE	SMID
1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8

Table 8 – Decomposition of options byte

Options							
MSB	...						LSB
–	LRLTWO	VLENGTH	TPDON	TPBON	DPON	DPPRIV	DPLAST
1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8

The first byte of the **BIH** shall specify *D_L*, the initial resolution layer to be specified in this **BIE**. Most frequently, this number will be zero in which case the data transmitted will allow building up the image without any prior knowledge about it. It will be non-zero if a previous **BIE** has already defined the image to some intermediate layer and only incremental information is to be specified. The second byte specifies *D*, the final resolution layer described in this **BIE**. Note that with such multiple **BIE**'s, when *D_L* is zero than *D* equals a number of differential layers, but not the total number of differential layers.

The third byte shall specify *P*, the number of bit-planes. If the image is bi-level, *P* shall be 1.

The fourth byte is fill. It shall always be written as 0.

The three subsequent four-byte fields specify X_D , Y_D and L_0 , which are respectively, the horizontal dimension at highest resolution, the vertical dimension at highest resolution, and the number of lines per stripe at lowest resolution. These three integers are coded most significant byte first. In other words, X_D is the sum of 256^3 times the fifth byte in **BIH**, 256^2 times the sixth byte, 256 times the seventh byte, and the eighth byte.

The seventeenth and eighteenth bytes shall specify M_X and M_Y , the maximum horizontal and vertical offsets allowed for the AT pixel. These parameters are discussed further in 6.7.3.

The nineteenth byte of the **BIH** shall carry the binary parameters **HITOLO**, **SEQ**, **ILEAVE** and **SMID**, which together specify the order in which stripe data is concatenated to form **BID**. More detail is provided in 6.2.4. The four most significant bits of this byte are fill and shall always be written as 0.

The twentieth byte of the **BIH** shall specify options. Its most significant bit is fill and shall always be written as 0. The template used for coding the lowest resolution layer shall have 2 or 3 lines as **LRLTWO** is 1 or 0 (see 6.7.1). If the **VLENGTH** bit is 0, there shall be no **NEWLEN** marker segments (see 6.2.6.2). If **VLENGTH** is 1, there may or may not be **NEWLEN** marker segments present. The **TPDON**, **TPBON** and **DPON** bits are 1 when it is desired to enable respectively, differential-layer TP, lowest-resolution-layer TP, and DP. The **DPPRIV** and **DPLAST** bits are only meaningful if **DPON** equals 1. If **DPON** is 1 and **DPPRIV** is 1, a private DP table is to be used. If **DPLAST** is 0, the private DP table (1728 bytes) is to be loaded. Otherwise the DP table last used is to be reused.

The **DPTABLE** field of **BIH** shall only be present if **DPON** equals 1, **DPPRIV** equals 1, and **DPLAST** equals 0. Its size and interpretation are defined in 6.6.

The variables D_L , D , P , X_D , Y_D , L_0 , M_X , M_Y , **HITOLO**, **SEQ**, **ILEAVE**, **SMID**, **LRLTWO**, **VLENGTH**, **TPDON**, **TPBON**, **DPON**, **DPPRIV** and **DPLAST** are free parameters. Some applications standards may restrict the choices for some or all of them. Table 9 shows the limits on these parameters as set by either their implicit natures or the field sizes allowed for them in a **BIH**.

A JBIG datastream is any datastream created as described in the normative portions of this Specification with parameters in the range of Table 9. In the interests of creating as large as possible a community of applications for which it is possible to share hardware and exchange decodable data, Annex A suggests minimum support for these free parameters. Different applications are encouraged to choose parameter values within the suggested ranges of minimum support whenever it is possible to do so. Implementations desiring to be compatible with a broad range of applications may wish to support all free parameter choices within the suggested ranges.

Table 9 – Absolute limits on free parameters

Parameter	Minimum	Maximum
D_L	0	D
D	D_L	255
P	1	255
X_D	1	4 294 967 295
Y_D	1	4 294 967 295
L_0	1	4 294 967 295
M_X	0	127
M_Y	0	255
HITOLO	0	1
SEQ	0	1
ILEAVE	0	1
SMID	0	1
LRLTWO	0	1
VLENGTH	0	1
TPDON	0	1
TPBON	0	1
DPON	0	1
DPPRIV	0	1
DPLAST	0	1

6.2.3 Iteration for parameters dependent on resolution layer

The image dimensions at lower layers (indexed by d) shall be defined recursively for $D \geq d \geq 1$ by the iterations.

$$X_{d-1} = \lceil X_d / 2 \rceil \quad (1)$$

$$Y_{d-1} = \lceil Y_d / 2 \rceil \quad (2)$$

where $\lceil \cdot \rceil$ denotes the ceiling function, or, in other words, the smallest integer greater than or equal to the argument.

For $1 \leq d \leq D$ the number of lines per stripe at layer d shall be defined by

$$L_d = 2 \times L_{d-1} \quad (3)$$

At all layers there will necessarily be

$$S = \lceil Y_0 / L_0 \rceil \quad (4)$$

stripes. In many cases, the last stripe in any layer d will have fewer than L_d lines.

6.2.4 Bi-level image data (BID) decomposition

The coded data $C_{s,d,p}$ defining a given stripe s at resolution d and bit plane p shall be contained in a stripe data entity or **SDE**. The **BID** shall consist of a concatenation of **SDE**'s and floating marker segments as shown in Table 10.

Table 10 – Decomposition of BID

BID						
Floating marker segment(s)	SDE _{s, d, p}	Floating marker segments(s)	SDE _{s, d, p}	...	Floating marker segment(s)	SDE _{s, d, p}
Varies	Varies	Varies	Varies	...	Varies	Varies

The ordering of the **SDE**'s depends on **HITOLO**, **SEQ**, **ILEAVE** and **SMID**. Index nesting shall be as defined by Table 11. Only the six combinations of the three variables **SEQ**, **ILEAVE** and **SMID** shown in Table 11 are allowed. The remaining two combinations shall never occur. The loops over the dummy variables s and p are respectively from 0 to $S-1$ (top to bottom) and $P-1$ to 0 (MSB to LSB). If **HITOLO** is 0, the dummy variable d shall range from D_L to D . Otherwise it shall range from D to D_L .

For a tutorial example see Table Intro. 2.

Table 11 – Ordering of stripe encodings in BID

			Loops		
SEQ	ILEAVE	SMID	Outer	Middle	Inner
0	0	0	p	d	s
0	1	0	d	p	s
0	1	1	d	s	p
1	0	0	s	p	d
1	0	1	p	s	d
1	1	0	s	d	p

6.2.5 Stripe data entity (SDE) decomposition

As shown in Table 12, each **SDE** shall be terminated by an **ESC** byte and either an **SDNORM** byte or an **SDRST** byte.

Table 12 – Structure of the **SDE**

SDE		
PSCD	ESC	SDNORM or SDRST
Varies	1	1

Normally the terminating byte will be **SDNORM**, which implies that all “state” information is saved. If instead the terminating byte is **SDRST**, the “state” for that particular bit-plane and that particular resolution layer shall be reset prior to encoding or decoding the next stripe of that plane and layer. Resetting the state with **SDRST** requires initializing the adaptive probability estimators as at the top of the image, resetting (if necessary) the AT pixel to its default location, and initializing $LNTF_{y-1}$ to 1 when in the lowest resolution layer. It also requires that all functions including resolution reduction, deterministic prediction, typical prediction, and model templates start the next stripe as they do when at the top of the image, that is, as defined in 6.1.2.

NOTE – Resetting the state with **SDRST** should not be done unnecessarily as doing so degrades compression efficiency and may introduce artifacts at stripe borders in lower-resolution images.

Protected stripe coded data (**PSCD**) is defined as the bytes of **SDE** that remain after removing the two terminating bytes. A decoder shall create stripe coded data (**SCD**) from the **PSCD** by replacing all occurrences of an **ESC** byte followed by a **STUFF** byte with a single **ESC** byte. An encoder shall create **PSCD** from **SCD** by replacing all occurrences of an **ESC** byte with an **ESC** byte followed by a **STUFF** byte. The use of **SCD** is described in 6.8. **PSCD** is used in defining the **SDE** rather than **SCD** so that data for one stripe can be unambiguously located in the **BID**.

An **ESC** byte and **ABORT** byte may be used to prematurely terminate the **BID** as shown in Table 13.

NOTE – Without a mechanism like this an encoder encountering a problem would “hang” its associated decoder indefinitely. There would be no way to restart the decoder as it would not reset until after it had received the announced amount of data.

Table 13 – Marker code to prematurely terminate a **BID**

ESC	ABORT
1	1

6.2.6 Floating marker segments

Floating marker segments provide control information. They shall not occur within an **SDE**. They may occur between **SDE**'s or before the first **SDE**. There are three floating marker segments: **ATMOVE**, **NEWLEN**, and **COMMENT**. (See Note 2 in 6.2.6.2.)

6.2.6.1 Adaptive-template (AT) movement

The location of the AT pixel may be changed with the **ATMOVE** marker segment shown in Table 14.

Table 14 – Structure of the **ATMOVE** floating marker segment

ESC	ATMOVE	Y_{AT}	τ_X	τ_Y
1	1	4	1	1

The third, fourth, fifth, and sixth byte define y_{AT} , the line at which the template changes. The seventh and eighth bytes define τ_X and τ_Y , the horizontal and vertical offsets for the new AT pixel. The line y_{AT} shall be decoded as the sum of 256^3 times the third byte, 256^2 times the fourth byte, 256 times the fifth byte, and the sixth byte.

The probability estimator is not re-initialized following an **ATMOVE**. The resolution layer and bit plane for which a given **ATMOVE** marker segment is to be effective shall be that of the first **SDE** to follow the marker segment. The line numbering for y_{AT} restarts at 0 for each stripe so that if, for example, a change is to be effective on the initial line of a stripe, y_{AT} equals 0.

Further discussion of adaptive template pixels and the variables y_{AT} , τ_X , and τ_Y appears in 6.7.3.

6.2.6.2 Redefining image length

If **VLENGTH** is 1, it is permissible to change the length Y_D of the image with a new-length marker segment as shown in Table 15.

Table 15 – Marker segment to indicate a new vertical dimension

ESC	NEWLEN	Y_D
1	1	4

At most only one new length marker segment shall appear in any **BIE**. However, a marker segment could refer to a line in the immediately preceding stripe due to an unexpected termination of the image or the use of only one stripe. Such a marker segment is followed immediately by **ESC** + **SDNORM/SDRST**, and the new Y_D given by the newlength marker segment can be less than the line number at the end of the preceding stripe. The encoder shall not code more than the number of lines in each layer corresponding to the new Y_D . Within the new length marker segment, Y_D shall be packed into its four byte field exactly as it is packed into its four byte field in **BIH**. The new Y_D shall never be greater than the original.

NOTES

1 The new length marker has been defined so that it is possible to begin coding an image of unknown length. In this case the original dimension Y_D placed into the header serves as an indication of the maximum length that the image might possibly be.

2 Some applications might need the new length marker segment to immediately follow a PSCD just before the **ESC** + **SDNORM/SDRST** in order to prematurely terminate. This usage is currently under study.

6.2.6.3 Comment marker segment

An **ESC** byte followed by a **COMMENT** byte and a four-byte integer L_c shall begin a comment marker segment as shown in Table 16.

The number L_c shall be equal to the sum of 256^3 times the third byte, 256^2 times the fourth byte, 256 times the fifth byte, and the sixth byte. This number shall give the length of only the private comment portion of the comment marker segment. In other words, the total length of the comment marker segment shall be $L_c + 6$ bytes.

Table 16 – Comment marker segment

ESC	COMMENT	L_c
1	1	4

6.2.7 Reserved marker byte

An **ESC** byte followed by a **RESERVE** byte is a reserved marker. One possible use of this marker is described in 6.8.2.8. No future extensions of this Specification shall use this marker for any purpose. Encoders or decoders may employ it for any private purpose desired. The reserved marker shall never appear in a public datastream.

6.3 Resolution reduction

The default deterministic prediction table is matched to the suggested resolution reduction algorithm described in this subclause. It is permissible to use an alternative resolution reduction algorithm in place of the suggested resolution reduction algorithm, but in that case either deterministic prediction must be disabled or a matched deterministic prediction table must be downloaded to the decoder as part of the **BIH**.

The resolution reduction algorithm is identical for all resolution layers and all bit-planes. The processing to create the image at resolution layer $d - 1$ from an image at resolution layer d is described here.

If X_d or Y_d is not even, for purposes of resolution reduction a new layer d image shall be created by adding as necessary either a column of background color (0) at the right side or a pixel replication of the last line at the bottom. Hence for the remainder of this subclause, X_d and Y_d are assumed even.

The original image can be divided up into two by two blocks of pixels, and each of these two by two superpixels shall map to one low-resolution pixel in the reduced-resolution image. These low-resolution pixels shall be determined from left to right and top to bottom in the normal raster scan order. The suggested resolution reduction rule of this Specification is defined by Figure 4 and Table 17. The reasoning behind this particular mapping is explained in Annex B (informative).

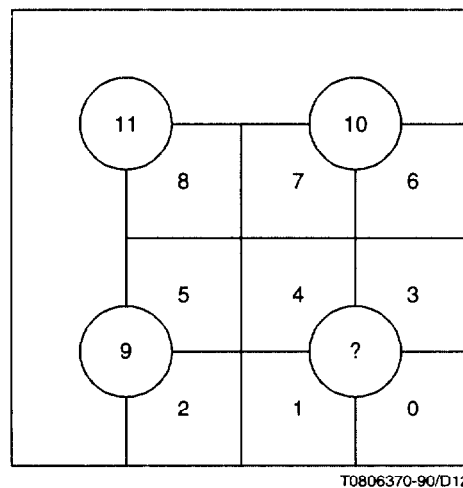


Figure 4 – Pixels used to determine the color of a low-resolution pixel

The circle with a “?” within it represents the low-resolution pixel the color of which is to be determined. The circles and squares with numbers within them correspond to pixels used in making this determination.

The colors of the numbered pixels define an index, with each numbered pixel defining one bit in the index. The pixel labeled “0” determines the least significant bit of the index and each additional numbered pixel determines the bit of the index corresponding to its number. When a pixel takes on the foreground color, its corresponding bit in the index shall take on the value “1.” Given this index, the color of the pixel labeled “?” shall be defined by Table 17, which is indexed from left to right. For example, the colors of the pixels corresponding to indices 0 through 7 are 0, 0, 0, 1, 0, 0, 0, 1, respectively.

At the edges of an image, some of the numbered pixels of Figure 4 may not be within the image. For the purpose of defining an index, the general edge rules of 6.1.2 shall be used.

When beginning resolution reduction, the upper-left most pixel of the high resolution image shall be aligned with pixel 4 in Figure 4.

6.4 Differential-layer typical prediction

Differential-layer TP shall be enabled or disabled with the **TPDON** bit in the options field of **BIH**. If it is disabled (**TPDON** = 0), the TPD block in both an encoder or decoder shall simply output **TPVALUE** = 2 for all pixels, to indicate to the arithmetic encoding or decoding block that no prediction is being made. Also, when differential-layer TP is disabled, the pseudo-pixel **LNTF** shall be neither encoded nor decoded by the arithmetic coder. The discussion in the remainder of this subclause and its subclauses assumes differential-layer TP is enabled (**TPDON** = 1).

Whenever reference is made to a pixel that because of edge effects is not actually in the current stripe, the value of this pixel shall be determined by the general edge rules of 6.1.2.