INTERNATIONAL STANDARD

ISO/IEC 14492

Second edition
2019-03

Information technology — Lossy/lossless coding of bi-level images

Technologies de l'information — Codage avec ou sans perte des images au trait

Reference number
ISO/IEC 14492:2019(E)

© ISO/IEC 2019



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2019

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see http://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

This document was prepared by ITU-T as ITU-T Rec. T.88 (2018) and drafted in accordance with its editorial rules. It was adopted by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This second edition cancels and replaces the first edition (ISO/IEC 14492:2001), which has been technically revised. It also incorporates ISO/IEC 14492:2001/Amd.1:2004, ISO/IEC 14492:2001/Amd.2:2003 and ISO/IEC 14492:2001/Amd.3:2012.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

© ISO/IEC 2019 – All rights reserved

INTERNATIONAL STANDARD ISO/IEC 14492
RECOMMENDATION ITU-T T.88

Information technology – Lossy/lossless coding of bi-level images

Summary

Rec. ITU-T T.88 | ISO/IEC 14492, commonly known as JBIG2, defines a coding method for bi-level images (e.g., black and white printed matter). These are images consisting of a single rectangular bit plane, with each pixel taking on one of just two possible colours. This Recommendation | International Standard has been explicitly prepared for a lossy, lossless, and lossy-to-lossless image compression.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T T.88	2000-02-10	8	11.1002/1000/4845
1.1	ITU-T T.88 (2000) Amd. 1	2003-06-29	16	11.1002/1000/6388
1.2	ITU-T T.88 (2000) Amd. 2	2003-06-29	16	11.1002/1000/6389
1.3	ITU-T T.88 (2000) Amd. 3	2011-05-14	16	11.1002/1000/11312
2.0	ITU-T T.88	2018-08-29	16	11.1002/1000/13688

Keywords

Bi-level images; image compression; lossy image compression; lossless image compression; lossy-to-lossless image compression; JBIG.

* To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

© ISO/IEC 2019 – All rights reserved

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at http://www.itu.int/ITU-T/ipr/.

© ITU 2019

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

© ISO/IEC 2019 – All rights reserved

Table of Contents

Page

0 Introduction v
0.1 Interpretation and use of the requirements v
0.2 Lossy coding ix
1 Scope 1
2 Normative references 1
3 Definitions 1
4 Symbols and abbreviations 3
4.1 Abbreviations 3
4.2 Symbol definitions 4
4.3 Operator definitions 11
5 Conventions 11
5.1 Typographic conventions 11
5.2 Binary notation 11
5.3 Hexadecimal notation 11
5.4 Integer value syntax 11
5.5 Array notation and conventions 12
5.6 Image and bitmap conventions 12
6 Decoding Procedures 13
6.1 Introduction to decoding procedures 13
6.2 Generic region decoding procedure 14
6.3 Generic Refinement Region Decoding Procedure 21
6.4 Text Region Decoding Procedure 25
6.5 Symbol Dictionary Decoding Procedure 33
6.6 Halftone Region Decoding Procedure 41
6.7 Pattern Dictionary Decoding Procedure 44
6.8 Colour palette decoding procedure 46
7 Control Decoding Procedure 47
7.1 General description 47
7.2 Segment header syntax 48
7.3 Segment t ypes 52
7.4 Segment syntaxes 54
8 Page Make-up 82
8.1 Decoder model 82
8.2 Page image composition 83
9 Encoding procedures (informative) 85
10 Control encoding procedures (informative) 85
11 Page break-up (informative) 85
11.1 Page break-up architecture 86
11.2 Page image decomposition 86
11.3 Multi-page document composition 88
Annex A – Arithmetic integer decoding procedure 89
A.1 General description 89
A.2 Procedure for decoding values (except IAID) 89
A.3 The IAID decoding procedure 91
Annex B – Huffman table decoding procedure 93
B.1 General description 93
B.2 Code table structure 93
B.3 Assigning the prefix codes 94
B.4 Using a Huffman table 95
B.5 Standard Huffman tables 96

© ISO/IEC 2019 – All rights reserved

Page

Annex C – Gray-scale image decoding procedure ... 103
C.1 General description ... 103
C.2 Input parameters ... 103
C.3 Return value ... 103
C.4 Variables used in decoding ... 103
C.5 Decoding the gray-scale image ... 103
Annex D – File formats ... 105
D.1 Sequential organization ... 105
D.2 Random-access organization ... 105
D.3 Embedded organization ... 106
D.4 File header syntax ... 106
Annex E – Arithmetic coding ... 108
E.1 Binary encoding ... 108
E.2 Description of the arithmetic encoder ... 109
E.3 Arithmetic decoding procedure ... 116
Annex F – Profiles ... 124
Annex G – Arithmetic decoding procedure (software conventions) ... 127
Annex H – Datastream example and test sequence ... 129
H.1 Datastream example ... 129
H.2 Test sequence for arithmetic coder ... 150
Annex I – Patents ... 156
I.1 List of patents ... 156
I.2 Contact addresses for patent information ... 157
Annex J – Compliant example encoding methods ... 158
J.1 List of JBIG2 encoding components and corresponding algorithms ... 158
J.2 Method references ... 159
Annex K – Electronic conformance data and sample software ... 161
K.1 Attached electronic data (informative) ... 161
K.2 Working environments of the released sample software (informative) ... 162
K.3 How to use the sample software (informative) ... 162
Bibliography ... 165

Electronic attachments:

1. Reference source code implementation
2. Test bitstreams

© ISO/IEC 2019 – All rights reserved

0 Introduction

This Recommendation | International Standard, informally called JBIG2, defines a coding method for bi-level images (e.g., black and white printed matter). These are images consisting of a single rectangular bit plane, with each pixel taking on one of just two possible colours. Multiple colours are to be handled using an appropriate higher level standard such as Recommendation ITU-T T.44. It is being drafted by the Joint Bi-level Image Experts Group (JBIG), a "Collaborative Team", established in 1988, that reports both to ISO/IEC JTC 1/SC29/WG1 and to ITU-T.

Compression of this type of image is also addressed by existing facsimile standards, for example by the compression algorithms in Recommendations ITU-T T.4 (MH, MR), T.6 (MMR), T.82 (JBIG1), and T.85 (Application profile of JBIG1 for facsimile). Besides the obvious facsimile application, JBIG2 will be useful for document storage and archiving, coding images on the World Wide Web, wireless data transmission, print spooling, and even teleconferencing.

As the result of a process that ended in 1993, JBIG produced a first coding standard formally designated Recommendation ITU-T T.82 | International Standard ISO/IEC 11544, which is informally known as JBIG or JBIG1. JBIG1 is intended to behave as lossless and progressive (lossy-to-lossless) coding. Though it has the capability of lossy coding, the lossy images produced by JBIG1 have significantly lower quality than the original images because the number of pixels in the lossy image cannot exceed one quarter of those in the original image.

On the contrary, JBIG2 was explicitly prepared for lossy, lossless, and lossy-to-lossless image compression. The design goal for JBIG2 was to allow for lossless compression performance better than that of the existing standards, and to allow for lossy compression at much higher compression ratios than the lossless ratios of the existing standards, with almost no visible degradation of quality. In addition, JBIG2 allows both quality-progressive coding, with the progression going from lower to higher (or lossless) quality, and content-progressive coding, successively adding different types of image data (for example, first text, then halftones). A typical JBIG2 encoder decomposes the input bi-level image into several regions and codes each of the regions separately using a different coding method. Such content-based decomposition is very desirable especially in interactive multimedia applications. JBIG2 can also handle a set of images (multiple page document) in an explicit manner.

As is typical with image compression standards, JBIG2 explicitly defines the requirements of a compliant bitstream, and thus defines decoder behaviour. JBIG2 does not explicitly define a standard encoder, but instead is flexible enough to allow sophisticated encoder design. In fact, encoder design will be a major differentiator among competing JBIG2 implementations.

Although this Recommendation | International Standard is phrased in terms of actions to be taken by decoders to interpret a bitstream, any decoder that produces the correct result (as defined by those actions) is compliant, regardless of the actions it actually takes.

Annexes A, B, C, D, E and F are normative, and thus form an integral part of this Recommendation | International Standard. Annexes G, H, I, J and K are informative, and thus do not form an integral part of this Recommendation | International Standard.

0.1 Interpretation and use of the requirements

This section is informative and designed to aid in interpreting the requirements of this Recommendation | International Standard. The requirements are written to be as general as possible to allow a large amount of implementation flexibility. Hence the language of the requirements is not specific about applications or implementations. In this section a correspondence is drawn between the general wording of the requirements and the intended use of this Recommendation | International Standard in typical applications.

0.1.1 Subject matter for JBIG2 coding

JBIG2 is used to code bi-level documents. A bi-level document contains one or more pages. A typical page contains some text data, that is, some characters of a small size arranged in horizontal or vertical rows. The characters in the text part of a page are called *symbols* in JBIG2. A page may also contain "halftone data", that is, gray-scale or colour multi-level images (e.g., photographs) that have been dithered to produce bi-level images. The periodic bitmap cells in the halftone part of the page are called *patterns* in JBIG2. In addition, a page may contain other data, such as line art and noise. Such non-text, non-halftone data is called *generic* data in JBIG2.

The JBIG2 image model treats text data and halftone data as special cases. It is expected that a JBIG2 encoder will divide the content of a page into a text region containing digitized text, a halftone region containing digitized halftones, and a generic region containing the remaining digitized image data, such as line-art. In some circumstances, it is better (in image quality or compressed data size) to consider text or halftones as generic data; conversely, in some circumstances it is better to consider generic data using one of the special cases.

© ISO/IEC 2019 – All rights reserved

An encoder is permitted to divide a single page into any number of regions, but often three regions will be sufficient, one for textual symbols, one for halftone patterns, and the third for the generic remainder. In some cases, not all types of data may be present, and the page may consist of fewer than three regions.

The various regions may overlap on the physical page. JBIG2 provides the means to specify how the overlapping regions are recombined to form the final page image.

A text region consists of a number of symbols placed at specified locations on a background. The symbols usually correspond to individual text characters. JBIG2 obtains much of its effectiveness by using individual symbols more than once. To reuse a symbol, an encoder or decoder must have a succinct way of referring to it. In JBIG2, the symbols are collected into one or more symbol dictionaries. A symbol dictionary is a set of bitmaps of text symbols, indexed so that a symbol bitmap may be referred to by an index number.

A halftone region consists of a number of patterns placed along a regular grid. The patterns usually correspond to gray-scale values. Indeed, the coding method of the pattern indices is designed as a gray-scale coder. Compression can be realized by representing the binary pixels of one grid cell by a single integer, the halftone index (which is usually a rendered gray-scale value). This many-to-one mapping (the pattern in a cell into a gray-scale value) may have the effect that edge information present in the original bitmap is lost by halftone coding. For this reason, lossless or near-lossless coding of halftones will often be better in image quality (though larger in size) if the halftone is coded with generic coding rather than halftone coding.

0.1.2 Relationship between segments and documents

A JBIG2 file contains the information needed to decode a bi-level document. A JBIG2 file is composed of *segments*. A typical page is coded using several segments. In a simple case, there will be a page information segment, a symbol dictionary segment, a text region segment, a pattern dictionary segment, a halftone region segment, and an end-of-page segment. The page information segment provides general information about the page, such as its size and resolution. The dictionary segments collect bitmaps referred to in the region segments. The region segments describe the appearance of the text and halftone regions by referencing bitmaps from a dictionary and specifying where they should appear on the page. The end-of-page segment marks the end of the page.

0.1.3 Structure and use of segments

Each segment contains a segment header, a data header, and data. The segment header is used to convey segment reference information and, in the case of multi-page documents, page association information. A data header gives information used for decoding the data in the segment. The data describes an image region or a dictionary, or provides other information.

Segments are numbered sequentially. A segment may refer to a lower-numbered, or *earlier*, segment. A region segment is always associated with one specific page of the document. A dictionary segment may be associated with one page of the document, or it may be associated with the document as a whole.

A region segment may refer to one or more earlier dictionary segments. The purpose of such a reference is to allow the decoder to identify symbols in a dictionary segment that are present into the image.

A region segment may refer to an earlier region segment. The purpose of such a reference is to combine the image described by the earlier segment with the current representation of the page.

A dictionary segment may refer to earlier dictionary segments. The symbols added to a dictionary segment may be described directly, or may be described as refinements of symbols described previously, either in the same dictionary segment or in earlier dictionary segments.

A JBIG2 file may be organized in two ways, sequential or random access. In the sequential organization, each segment's segment header immediately precedes that segment's data header and data, all in sequential order. In the random access organization, all the segment headers are collected together at the beginning of the file, followed by the data (including data headers) for all the segments, in the same order. This second organization permits a decoder to determine all segment dependencies without reading the entire file.

A third way of encapsulating of JBIG2-encoded data is to embed it in a non-JBIG2 file – this is sometimes called the *embedded organization*. In this case a different file format carries JBIG2 segments. The segment header, data header, and data of each segment are stored together, but the embedding file format may store the segments in any order, at any set of locations within its own structure.

0.1.4 Internal representations

Decoded data must be stored before printing or display. While this Recommendation | International Standard does not specify how to store it, its decoding model presumes certain data structures, specifically buffers and dictionaries.

© ISO/IEC 2019 – All rights reserved

Figure 1 illustrates major decoder components and associated buffers. In this figure, decoding procedures are outlined in bold lines, and memory components are outlined in non-bold lines. Also, bold arrows indicate that one decoding procedure invokes another decoding procedure; for example, the symbol dictionary decoding procedure invokes the generic region decoding procedure to decode the bitmaps for the symbols that it defines. Non-bold arrows indicate flow of data: the text region decoding procedure reads symbols from the symbol memory and draws them into the page buffer or an auxiliary buffer. Although it is not shown in Figure 1, the encoded data stream flows to the decoding procedures, and the block labelled "Page and auxiliary buffers" produces the final decoded page images.

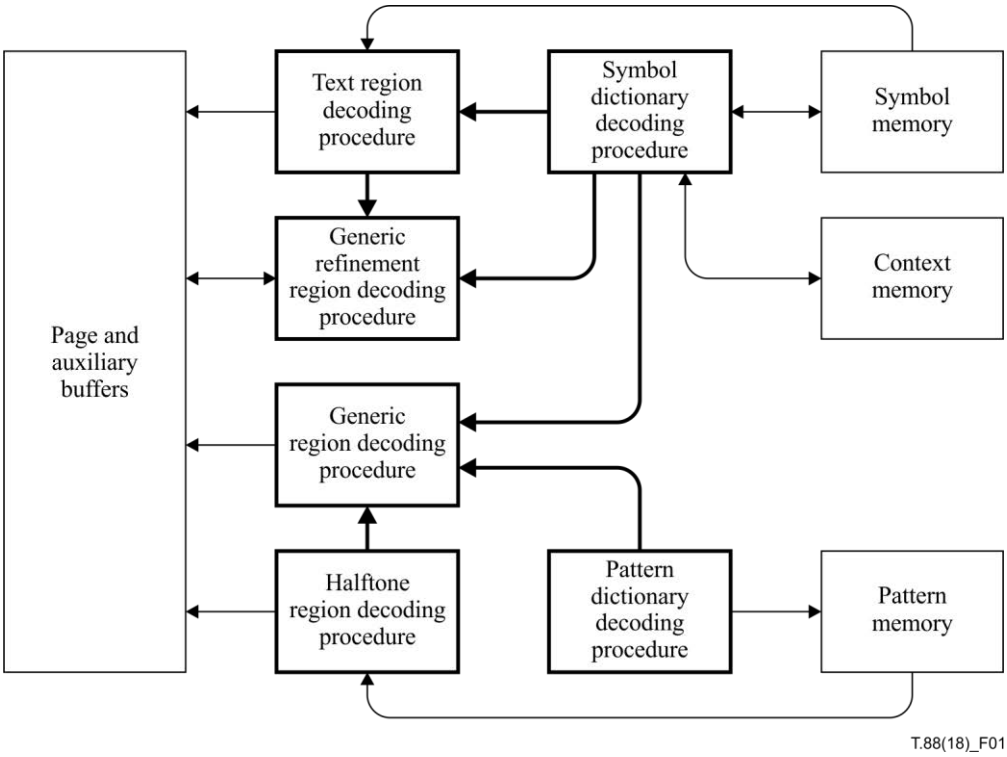


Figure 1 – Block diagram of major decoder components

The resources required to decode any given JBIG2 bitstream depend on the complexity of that bitstream. Some techniques such as striping can be used to reduce decoder memory requirements. It is estimated that a full-featured decoder may need two full-page buffers, plus about the same amount of dictionary memory, plus about 100 kilobytes of arithmetic coding context memory, to decode most bitstreams.

A buffer is a representation of a bitmap. A buffer is intended to hold a large amount of data, typically the size of a page. A buffer may contain the description of a region or of an entire page. Even if the buffer describes only a region, it has information associated with it that specifies its placement on the page. Decoding a region segment modifies the contents of a buffer.

There is one special buffer, the *page buffer*. It is intended that the decoder accumulate region data directly in the page buffer until the page has been completely decoded; then the data can be sent to an output device or file. Decoding an *immediate* region segment modifies the contents of the page buffer. The usual way of preparing a page is to decode one or more immediate region segments, each one modifying the page buffer. The decoder may output an incomplete page buffer, either as part of progressive transmission or in response to user input. Such output is optional, and its content is not specified by this Recommendation | International Standard.

All other buffers are auxiliary buffers. It is intended that the decoder fill an auxiliary buffer, then later use it to refine the page buffer. In an application, it will often be unnecessary to have any auxiliary buffers. Decoding an *intermediate* region segment modifies the contents of an auxiliary buffer. The decoder may use auxiliary buffers to output pages other than those found in a complete page buffer, either as part of progressive transmission or in response to user input. Such output is optional, and its content is not specified by this Recommendation | International Standard.

A symbol dictionary consists of an indexed set of bitmaps. The bitmaps in a dictionary are typically small, approximately the size of text characters. Unlike a buffer, a bitmap in a dictionary does not have page location information associated with it.

© ISO/IEC 2019 – All rights reserved

0.1.5 Decoding results

Decoding a segment involves invocation of one or more decoding procedures. The decoding procedures to be invoked are determined by the segment type.

The result of decoding a region segment is a bitmap stored in a buffer, possibly the page buffer. Decoding a region segment may fill a new buffer, or may modify an existing buffer. In typical applications, placing the data into a buffer involves changing pixels from the background colour to the foreground colour, but this Recommendation | International Standard specifies other permissible ways of changing a buffer's pixels.

A typical page will be described by one or more immediate region segments, each one resulting in modification of the page buffer.

Just as it is possible to specify a new symbol in a dictionary by refining a previously specified symbol, it is also possible to specify a new buffer by refining an existing buffer. However, a region may be refined only by the generic refinement decoding procedure. Such a refinement does not make use of the internal structure of the region in the buffer being refined. After a buffer has been refined, the original buffer is no longer available.

The result of decoding a dictionary segment is a new dictionary. The symbols in the dictionary may later be placed into a buffer by the text region decoding procedure.

0.1.6 Decoding procedures

The *generic region decoding procedure* fills or modifies a buffer directly, pixel-by-pixel if arithmetic coding is being used, or by runs of foreground and background pixels if MMR and Huffman coding are being used. In the arithmetic coding case, the prediction context contains only pixels determined by data already decoded within the current segment.

The *generic refinement region decoding procedure* modifies a buffer pixel-by-pixel using arithmetic coding. The prediction context uses pixels determined by data already decoded within the current segment as well as pixels already present either in the page buffer or in an auxiliary buffer.

The *text region decoding procedure* takes symbols from one or more symbol dictionaries and places them in a buffer. This procedure is invoked during the decoding of a text region segment. The text region segment contains the position and index information for each symbol to be placed in the buffer; the bitmaps of the symbols are taken from the symbol dictionaries.

The *symbol dictionary decoding procedure* creates a symbol dictionary, that is, an indexed set of symbol bitmaps. A bitmap in the dictionary may be coded directly; it may be coded as a refinement of a symbol already in a dictionary; or it may be coded as an aggregation of two or more symbols already in dictionaries. This decoding procedure is invoked during the decoding of a symbol dictionary segment.

The *halftone region decoding procedure* takes patterns from a pattern dictionary and places them in a buffer. This procedure is invoked during the decoding of a halftone region segment. The halftone region segment contains the position information for all the patterns to be placed in the buffer, as well as index information for the patterns themselves. The patterns, the fixed-size bitmaps of the halftone, are taken from the halftone dictionaries.

The *pattern dictionary decoding procedure* creates a dictionary, that is, an indexed set of fixed-size bitmaps (patterns). The bitmaps in the dictionary are coded directly and jointly. This decoding procedure is invoked during the decoding of a pattern dictionary segment.

The *control decoding procedure* decodes segment headers, which include segment type information. The segment type determines which decoding procedure must be invoked to decode the segment. The segment type also determines where the decoded output from the segment will be placed. The segment reference information, also present in the segment header and decoded by the control decoding procedure, determines which other segments must be used to decode the current segment. The control decoding procedure affects everything shown in Figure 1, and so is not shown there as a separate block.

Table 1 summarizes the types of data being decoded, which decoding procedure is responsible for decoding them, and what the final representations of the decoded data are.

© ISO/IEC 2019 – All rights reserved

Table 1 – Entities in the decoding process

Concept	JBIG2 bitstream entity	JBIG2 decoding entity	Physical representation
Document	JBIG2 file	JBIG2 decoder	Output medium or device
Page	Collection of segments	Implicit in control decoding procedure	Page buffer
Region	Region segment	Region decoding procedure	Page buffer or auxiliary buffer
Dictionary	Dictionary segment	Dictionary decoding procedure	List of symbols
Character	Field within a symbol dictionary segment	Symbol dictionary decoding procedure	Symbol bitmap
Gray-scale value	Field within a halftone dictionary segment	Pattern dictionary decoding procedure	Pattern

0.2 Lossy coding

This Recommendation | International Standard does not define how to control lossy coding of bi-level images. Rather it defines how to perform perfect reconstruction of a bitmap that the encoder has chosen to encode. If the encoder chooses to encode a bitmap that is different than the original, the entire process becomes one of lossy coding. The different coding methods allow for different methods of introducing loss in a profitable way.

0.2.1 Symbol coding

Lossy symbol coding provides a natural way of doing lossy coding of text regions. The idea is to allow small differences between the original symbol bitmap and the one indexed in the symbol dictionary. Compression gain is effected by not having to code a large dictionary and, afterwards, by having a cheap symbol index coding as a consequence of the smaller dictionary. It is up to the encoder to decide when two bitmaps are essentially the same or essentially different. This technique was first described in [1].

The hazard of lossy symbol coding is to have *substitution errors*, that is, to have the encoder replace a bitmap corresponding to one character by a bitmap depicting a different character, so that a human reader misreads the character. The risk of substitution errors can be reduced by using intricate measures of difference between bitmaps and/or by making sure that the critical pixels of the indexed bitmap are correct. One way to control this, described in [5], is to index the possibly wrong symbol and then to apply refinement coding to that symbol bitmap. The idea is to encode the basic character shape at little cost, then correct pixels that the encoder believes alter the meaning of the character.

The process of beneficially introducing loss in textual regions may also take simpler forms such as removing flyspecks from documents or regularizing edges of letters. Most likely such changes will lower the code length of the region without affecting the general appearance of the region – possibly even improving the appearance.

A number of examples of performing this sort of lossy symbol coding with JBIG2 can be found in [7].

NOTE – Although the term "text region" is used for regions of the page coded using symbol coding, other possible uses of symbol coding include coding line-art and other non-textual data.

0.2.2 Generic coding

To effect near-lossless coding using generic coding, the encoder applies a preprocess to an original image and encodes the changed image losslessly. The difficulties are to ensure that the changes result in a lower code length and that the quality of the changed image does not suffer badly from the changes. Two possible preprocesses are given in [11]. These preprocesses flip pixels that, when flipped, significantly lower the total code length of the region, but can be flipped without seriously impairing the visual quality. The preprocesses provide for effective near-lossless coding of periodic halftones and for a moderate gain in compression for other data types. The preprocesses are not well-suited for error diffused images and images dithered with blue noise as perceptually lossless compression will not be achieved at a significantly lower rate than the lossless rate.

0.2.3 Halftone coding

Halftone coding is the natural way to obtain very high compression for *periodic* halftones, such as clustered-dot ordered dithered images. In contrast to lossy generic coding as described above, halftone coding does not intend to preserve the original bitmap, although this is possible in special cases. Loss can also be introduced for additional compression by not putting all the patterns of the original image into the dictionary, thereby reducing both the number of halftone patterns and the number of bits required to specify which pattern is used in which location.

© ISO/IEC 2019 – All rights reserved

For lossy coding of error diffused images and images dithered with blue noise, it is advisable to use halftone coding with a small grid size. A reconstructed image will lack fine details and may display blockiness but will be clearly recognizable. The blockiness may be reduced on the decoder side in a postprocess; for instance, by using other reconstruction patterns than those that appear in the dictionary. Error diffused images may also be coded losslessly, or with controlled loss as described above, using generic coding.

More details on performing this halftone coding can be found in [12].

0.2.4 Consequences of inadequate segmentation

In order to obtain optimum coding, both in terms of quality and file size, the correct form of encoding should be used for the appropriate regions of the document pages. This subclause briefly describes the consequences of errors in this segmentation.

Using lossy symbol coding for a document containing both text and halftone data will result in poor compression. Depending on the encoder, the quality of the halftone data may be good or bad. Using the form of lossy symbol coding described in [5], the visual quality will probably not suffer.

Using lossy generic coding (using the preprocesses given in [11]) for a document containing both symbol and halftone data usually results in good quality and moderate compression.

Line art and regions of handwritten text may be coded efficiently using generic coding, but depending on the encoder, these types of regions can also be very effectively coded with symbol coding.

© ISO/IEC 2019 – All rights reserved

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION

Information technology – Lossy/lossless coding of bi-level images

1 Scope

This Recommendation | International Standard[1] defines methods for coding bi-level images and sets of images (documents consisting of multiple pages). It is particularly suitable for bi-level images consisting of text and dithered (halftone) data.

The methods defined permit lossless (bit-preserving) coding, lossy coding, and progressive coding. In progressive coding, the first image is lossy; subsequent images may be lossy or lossless.

This Recommendation | International Standard also defines file formats to enclose the coded bi-level image data.

2 Normative references

The following Recommendations and International Standards contain provisions which, through references in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of ITU maintains a list of the currently valid ITU-T Recommendations.

- Recommendation ITU-T T.6 (1988), *Facsimile coding schemes and coding control functions for Group 4 facsimile apparatus*.
- Recommendation ITU-T T.45 (2000), *Run-length colour encoding*.
- ISO/IEC 8859-1:1998, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1*.
- ISO/IEC 10646:2017, *Information technology – Universal Coded Character Set (UCS)*.

3 Definitions

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

3.1 **adaptive template pixel(s)**: Special pixel(s), in a template, whose location is not fixed.

3.2 **aggregation**: Joining or merging of several individual symbols into a new symbol.

3.3 **bi-level image**: Rectangular array of bits.

3.4 **bit**: Binary digit, representing the value **`0`** or **`1`**.

3.5 **bitmap**: Bi-level image.

3.6 **buffer**: Storage area used to hold a bitmap.

3.7 **byte**: Eight bits of data.

3.8 **combination operator**: Operator used to combine the prior contents of a bitmap with new values being drawn into that bitmap.

3.9 **coordinate system**: Numbering system for two-dimensional locations where locations are labelled by two numbers, the first one increasing from left to right and the second one increasing from top to bottom.

3.10 **delta S**: Difference in the S coordinates between two successive symbol instances in a non-empty strip.

3.11 **delta T**: Difference in the T coordinates between two successive non-empty strips.

---

[1] This Recommendation | International Standard includes an electronic attachment with a reference source code implementation and test bitstreams.

© ISO/IEC 2019 – All rights reserved

3.12 **decoding procedure**: Component of a decoder that decodes a certain type of data.

3.12.1 **integer decoding procedure**: Decoding procedure whose output on each invocation is a single value.

3.12.2 **arithmetic integer decoding procedure**: Integer decoding procedure that uses arithmetic entropy decoding.

3.12.3 **region decoding procedure**: Decoding procedure whose output is a bitmap.

3.12.4 **generic region decoding procedure**: Region decoding procedure that operates by decoding pixels individually or in runs.

3.12.5 **generic refinement region decoding procedure**: Region decoding procedure that operates by modifying a reference bitmap to produce an output bitmap.

3.12.6 **gray-scale decoding procedure**: Decoding procedure whose output is a gray-scale image.

3.12.7 **pattern dictionary decoding procedure**: Decoding procedure whose output is a list of patterns.

3.12.8 **halftone region decoding procedure**: Region decoding procedure that operates by drawing a set of patterns into a bitmap, placing the patterns according to a halftone grid.

3.12.9 **Huffman table decoding procedure**: Decoding procedure whose output is a Huffman table.

3.12.10 **text region decoding procedure**: Region decoding procedure that operates by drawing a set of symbol instances into a bitmap.

3.12.11 **symbol dictionary decoding procedure**: Decoding procedure whose output is a list of symbols.

3.13 **decoder**: Entity capable of decoding a bitstream in conformance with this Recommendation | International Standard.

3.14 **dictionary**: List of bitmaps.

3.14.1 **pattern dictionary**: List of patterns.

3.14.2 **symbol dictionary**: List of symbols.

3.15 **export flag**: Bit indicating that a symbol is on the export list of a symbol dictionary.

3.16 **export list**: List of the symbols in a symbol dictionary that may be used by referring to that symbol dictionary.

3.17 **gray-scale image**: Rectangular array of non-negative integer indices.

3.18 **gray-scale pixel**: Integer-valued element in a gray-scale image.

3.19 **halftone grid**: Rectilinear grid of locations specifying where patterns are to be drawn.

3.20 **height class**: Set of symbols in a symbol dictionary whose heights are all equal.

3.21 **height class delta height**: Difference in height between two height classes.

3.22 **height class delta width**: Difference in width between two symbols in a height class.

3.23 **Huffman table**: Collection of table lines specifying how values are encoded.

3.24 **lossless coding**: Method of encoding data so that the decoded data are identical to the original data.

3.25 **lossy coding**: Method of encoding data so that the decoded data differ, ideally only in insignificant ways, from the original data.

3.26 **ordinal**: Value used as a counter.

3.27 **out-of-band value**: Non-numeric value that may be produced in place of an integer.

3.28 **pattern**: Bitmap produced by a pattern dictionary decoding procedure.

3.29 **pixel**: Element with **`0`** or **`1`** as its value in a bitmap.

3.30 **prefix length**: Length of the Huffman code prefix in a table line.

3.31 **range length**: Number of additional code bits in a table line.

3.32 **reference bitmap**: Bitmap used as the reference plane during the refinement region decoding procedure.

3.33 **referred-to segment**: Other segment required in order to decode the current segment.

© ISO/IEC 2019 – All rights reserved

3.34 **region**: Bitmap produced by a region decoding procedure.

3.35 **segment**: Segment header and its segment data.

3.36 **strip**: Full-width or full-height portion of the coordinate system of a text region.

3.36.1 **empty strip**: Strip containing the reference corners of no symbol instances.

3.36.2 **non-empty strip**: Strip containing the reference corner of at least one symbol instance.

3.37 **strip size**: Extent in pixels of the non-full dimension of a strip.

3.38 **symbol**: Bitmap produced by a symbol dictionary decoding procedure.

3.39 **symbol ID**: Integer used to identify a symbol or to index into an array of symbols to retrieve the symbol.

3.40 **symbol instance**: Symbol drawn, possibly with refinement, at a particular location in a text region.

3.41 **symbol instance refinement delta height**: Difference in height between a symbol instance's reference bitmap and the bitmap produced by the generic refinement region decoding procedure.

3.42 **symbol instance refinement delta width**: Difference in width between a symbol instance's reference bitmap and the bitmap produced by the generic refinement region decoding procedure.

3.43 **symbol instance refinement delta X:** Difference between the X coordinates of the top left corners of a symbol instance's reference bitmap and the bitmap produced by the generic refinement region decoding procedure.

3.44 **symbol instance refinement delta Y**: Difference between the Y coordinates of the top left corners of a symbol instance's reference bitmap and the bitmap produced by the generic refinement region decoding procedure.

3.45 **table line**: Specification of the encoding of a single value or a range of values as a Huffman code prefix followed by a fixed number of additional code bits.

3.46 **typical prediction**: Method of signalling that an entire row of a generic region is identical to the preceding row.

3.47 **value**: Integer or out-of-band indicator that is decoded.

4 Symbols and abbreviations

NOTE – Due to ISO nomenclature requirements, within the context of clause 4, the term "symbol" is locally used to mean a variable name.

4.1 Abbreviations

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

AT Adaptive Template

EOFB End-of-Facsimile Block

ID Identifier

LPS Less Probable Symbol, i.e., less probable binary value

LSB Least Significant Bit

MMR Modified Modified READ

MPS More Probable Symbol, i.e., more probable binary value

MSB Most Significant Bit

OOB Out-of-Band

READ Relative Element Address Designate

TPGD Typical prediction for generic direct bitmap coding

TPGR Typical prediction for generic refinement bitmap coding

NOTE – The term "symbol" in the abbreviations LPS and MPS does not refer to the symbols (bitmaps) in this Recommendation | International Standard. The LPS and MPS abbreviations are used despite this because they are the generally-accepted terminology in arithmetic coding.

© ISO/IEC 2019 – All rights reserved