INTERNATIONAL STANDARD



First edition 2001-12-15

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

Technologies de l'information — Codage avec/sans perte d'images à deux niveaux

iTeh STANDARD PREVIEW (standards.iteh.ai)

<u>ISO/IEC 14492:2001</u> https://standards.iteh.ai/catalog/standards/sist/3d30457f-5349-492c-90b6-659e7ef15d76/iso-iec-14492-2001



Reference number ISO/IEC 14492:2001(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

iTeh STANDARD PREVIEW (standards.iteh.ai)

<u>ISO/IEC 14492:2001</u> https://standards.iteh.ai/catalog/standards/sist/3d30457f-5349-492c-90b6-659e7ef15d76/iso-iec-14492-2001

© ISO/IEC 2001

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 either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office Case postale 56 • CH-1211 Geneva 20 Tel. + 41 22 749 01 11 Fax + 41 22 749 09 47 E-mail copyright@iso.ch Web www.iso.ch

Printed in Switzerland

CONTENTS

0	Introd	Introduction				
	0.1 Interpretation and use of the requirements					
		0.1.1	Subject matter for JBIG2 coding	viii		
		0.1.2	Relationship between segments and documents	ix		
		0.1.3	Structure and use of segments	ix		
		0.1.4	Internal representations	ix		
		0.1.5	Decoding results	xi		
		0.1.6	Decoding procedures	xi		
	0.2	Lossy coding				
		0.2.1	Symbol coding	xii		
		0.2.2	Generic coding	xii		
		0.2.3	Halftone coding	xiii		
		0.2.4	Consequences of inadequate segmentation	xiii		
1	Scope			1		
2	Norm	native References				
3	Terms	s and Defin	nitions	1		
4	Symb	Symbols and Abbreviationsch STANDARD PREVIEW				
	4.1	Abbrevi	iations	3		
	4.2	Symbol	definitions	4		
	4.3	Operator definitions				
5	Conv	- F	<u>ISO/IEC 14492;2001</u> https://standarda.itah.ai/astala.a/standarda/sist/2d20457f 5240_402a_00h6	10		
3		nventions				
	5.1	Typographic conventions059e7e115d70/80-lec-14492-2001				
	5.2	Binary notation				
	5.3	Hexade	cimal notation	11		
	5.4	Integer	value syntax	11		
		5.4.1	Bit packing	11		
		5.4.2	Multi-byte values	11		
		5.4.3	Bit numbering	11		
		5.4.4	Signedness	11		
	5.5	Array n	otation and conventions	11		
	5.6	Image and bitmap conventions				
6	Deco	Decoding Procedures				
	6.1	Introduction to decoding procedures				
	6.2	5.2 Generic region decoding procedure				
	0.2	6.2.1	General description			
		622	Input parameters	13		
		623	Return value	13		
		6.2.4	Variables used in decoding	14		
		6.2.5	Decoding using a template and arithmetic coding.	14		
		6.2.6	Decoding using MMR coding	18		
	6.3	Generic	c Refinement Region Decoding Procedure	19		
		6.3.1	General description	19		
		6.3.2	Input parameters	19		
		6.3.3	Return value	19		
		6.3.4	Variables used in decoding	20		
		6.3.5	Decoding using a template and arithmetic coding	20		

6.4	Text Re	gion Decoding Procedure
	6.4.1	General description
	6.4.2	Input parameters
	6.4.3	Return value
	6.4.4	Variables used in decoding
	6.4.5	Decoding the text region.
	6.4.6	Strip delta T
	6.4.7	First symbol instance S coordinate
	6.4.8	Subsequent symbol instance S coordinate
	6.4.9	Symbol instance T coordinate
	6.4.10	Symbol instance symbol ID
	6.4.11	Symbol instance bitmap
5.5	Symbol	Dictionary Decoding Procedure
	6.5.1	General description
	6.5.2	Input parameters
	6.5.3	Return value
	654	Variables used in decoding
	655	Decoding the symbol dictionary
	656	Height class delta height
	657	Delta width
	658	Symbol bitman
	659	Height class collective hitman
	6510	Exported symbols
<i>c c</i>	U.J.10	Pagion Deceding Precedure
5.0	Hainon	Concerned description
	0.0.1	
	0.0.2	Determined by
	0.0.3	Keturn value.
	0.0.4	Deceding the helftone project
	0.0.3	Decoding the natione region
5.7	Pattern	Dictionary Decoding Procedure F.C. 14492:2001
	6.7.1	General description ai catalog/standards/sist/3d30457f-5349-492c-90b6
	6.7.2	Input parameters659e7ef15d76/iso-ico-14492-2001
	6.7.3	Return value
	6.7.4	Variables used in decoding
	6.7.5	Decoding the pattern dictionary
Contro	ol Decodin	g Procedure
7.1	General	description
7.2	Segmen	t header syntax
	7.2.1	Segment header fields
	7.2.2	Segment number
	7.2.3	Segment header flags
	7.2.4	Referred-to segment count and retention flags
	7.2.5	Referred-to segment numbers
	726	Segment page association
	727	Segment data length
	728	Segment header example
12	, .2.0 Soomer	
.5		Dulas for sogmant references
	1.3.1	Nules for page associations
	1.5.2	kules for page associations
.4	Segmen	t syntaxes
	7.4.1	Region segment information field
	7.4.2	Symbol dictionary segment syntax
	7.4.3	Text region segment syntax
	7.4.4	Pattern dictionary segment syntax
	7.4.5	Halftone region segment syntax
	7.4.6	Generic region segment syntax
	7 4 7	Generic refinement region syntax
	1.4.1	Generic reminiment region syntax

7

				Page
		7.4.9	End of page segment syntax	76
		7.4.10	End of stripe segment syntax	76
		7.4.11	End of file segment syntax	76
		7.4.12	Profiles segment syntax	76
		7.4.13	Code table segment syntax	77
		7.4.14	Extension segment syntax	77
		7.4.15	Defined extension types	77
8	Page N	/lake-up		78
	8.1	Decoder	model	78
	8.2	Page image	age composition	78
Anne	хА_	Arithmetic	Integer Decoding Procedure	80
1 11110	A 1	General	description	83
	A.1	Drocodu	re for decoding values (except IAID)	02
	A.2		D decoding values (except IAID)	04
	A.3	Ine IAI	D decoding procedure	84
Anne	x B – 1	Huffman T	Cable Decoding Procedure	86
	B.1	General	description	86
	B.2	Code tab	ple structure	86
		B.2.1	Code table flags	87
		B.2.2	Code table lowest value	87
		B.2.3	Code table highest value	87
	B.3	Assignin	ng the prefix codes	87
	B.4	Using a	Huffman table	88
	B.5	Standard	1 Huffman tables STANDARD PREVIEW	89
Anne	x C –	Grav-scale	e Image Decoding Procedure 1 1	97
7 11110	C_1	General	description (Stanuarus.Iten.al)	07
	C.1	Input po	romatara	07
	C.2	mput pa	<u>ISO/IEC 14492:2001</u>	97
	C.3	Return v	^{/alufaps} ://standards.iteh.ai/catalog/standards/sist/3d30457f-5349-492c-90b6	97
	C.4	Variable	s used in decoding65967.ef15d76/iso-icc-14492-2001.	9
	C.5	Decodin	g the gray-scale image	98
Anne	x D – 1	File Forma	ats	99
	D.1	Sequenti	ial organisation	99
	D.2	Random	-access organisation	99
	D.3	Embedd	ed organisation	100
	D4	File head	der syntax	100
	2.1	D.4.1	D string	100
		D.4.2	File header flags	100
		D.4.3	Number of pages	100
Anne	x E –	Arithmetic	Coding	101
		Rinary	ncoding	101
	L. I	F 1 1	Recursive interval subdivision	101
		E.1.1 E.1.2	Coding conventions and approximations	101
	БЭ	Descript	ion of the arithmetic encoder	101
	L .2	E 2 1	Encoder code register conventions	102
		E.2.1 F 2 2	Encoding a decision (ENCODE)	102
		E.2.2	Encoding a 1 or 0 (CODE1 and CODE0)	103
		E.2.5 E.2.4	Encoding an MPS or LPS (CODEMPS and CODELS)	10.
		E.2.5	Probability estimation	104
		E.2.6	Renormalisation in the encoder (RENORME)	10
		E.2.7	Compressed data output (BYTEOUT)	10
		E.2.8	Initialisation of the encoder (INITENC)	10
		E.2.9	Termination of encoding (FLUSH)	10
		E.2.10	Minimisation of the compressed data	10

			Page
E.3	Arithmetic decoding procedure		109
	E.3.1	Decoder code register conventions	111
	E.3.2	Decoding a decision (DECODE)	111
	E.3.3	Renormalisation in the decoder (RENORMD)	111
	E.3.4	Compressed data input (BYTEIN)	111
	E.3.5	Initialisation of the decoder (INITDEC)	114
	E.3.6	Resynchronisation of the decoder	114
	E.3.7	Resetting arithmetic coding statistics	115
	E.3.8	Saving arithmetic coding statistics	115
Annex F – Pro	ofiles		116
Annex G – Ar	ithmetic 1	Decoding Procedure (Software Conventions)	119
Annex H – Da	tastream	Example and Test Sequence	121
H.1	Datastre	eam example	121
H.2	Test seq	uence for arithmetic coder	142
Annex I – Pat	ents		147
Bibliography			149

iTeh STANDARD PREVIEW (standards.iteh.ai)

<u>ISO/IEC 14492:2001</u> https://standards.iteh.ai/catalog/standards/sist/3d30457f-5349-492c-90b6-659e7ef15d76/iso-iec-14492-2001

Foreword

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

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

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

International Standard ISO/IEC 14492 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 ITU-T. The identical text is published as ITU-T Recommandation T.88.

Annexes A, B, C, D, E and F form a normative part of ISO/IEC 14492. Annexes G, H and I are for information only.

(standards.iteh.ai)

<u>ISO/IEC 14492:2001</u> https://standards.iteh.ai/catalog/standards/sist/3d30457f-5349-492c-90b6-659e7ef15d76/iso-iec-14492-2001

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 ITU-T Recommendation 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 ITU-T Recommendations 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 ITU-T Recommendation 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.

iTeh STANDARD PREVIEW

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.

ISO/IEC 14492:2001

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 This Recommendation | International Standard. Annexes G and H 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 digitised text, a halftone region containing digitised halftones, and a generic region containing the remaining digitised 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.

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 grayscale values. Indeed, the coding method of the pattern indices is designed as a gray-scale coder. Compression can be realised 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 (standards.iteh.ai)

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 organised in two ways, sequential or random access. In the sequential organisation, each segment's segment header immediately precedes that segment's data header and data, all in sequential order. In the random access organisation, 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 organisation 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 organisation*. 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.

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 labeled "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.

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.

https://standards.iteh.ai/catalog/standards/sist/3d30457f-5349-492c-90b6-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 summarises the types of data being decoded, which decoding procedure is responsible for decoding them, and what the final representations of the decoded data are.

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	Pattern

Table $I = Entities$ in the decoding proc	ess
---	-----

(standards.iteh.ai)

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.

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. **CS.Iten.al**

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: 22:2001 https://standards.iteh.ai/catalog/standards/sist/3d30457f-5349-492c-90b6-

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.

iTeh STANDARD PREVIEW (standards.iteh.ai)

<u>ISO/IEC 14492:2001</u> https://standards.iteh.ai/catalog/standards/sist/3d30457f-5349-492c-90b6-659e7ef15d76/iso-iec-14492-2001

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – LOSSY/LOSSLESS CODING OF BI-LEVEL IMAGES

1 Scope

This Recommendation | International Standard 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 Anternational 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.

- CCITT Recommendation T.6 (1988), Facsimile coding schemes and coding control functions for Group 4 facsimile apparatus.
- ISO/IEC 8859-1:1998, Information technology 8-bit single-byte coded graphic character sets Part 1: Latin alphabet No. 1.
- ISO/IEC 10646-1:2000, Information technology Universal Multiple-Octet Coded Character Set (UCS) Part 1: Architecture and Basic Multilingual Plane.

3 Terms and Definitions

For the purposes of this Recommendation | International Standard, the following terms and 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.