PAPER

Simple Bitplane Coding and Its Application to Multi-Functional Image Compression

Hisakazu KIKUCHI[†a)], *Fellow*, **Ryosuke ABE**[†], *Nonmember*, *and* **Shogo MURAMATSU**[†], *Member*

SUMMARY A simple image compression scheme is presented for various types of images, which include color/grayscale images, color-quantized images, and bilevel images such as document and digital halftone images. It is a bitplane coding composed of a new context modeling and adaptive binary arithmetic coding. A target bit to be encoded is conditioned by the estimates of the neighboring pixels including non-causal locations. Several functionalities are also integrated. They are arbitrary shaped ROI transmission, selective tile partitioning, accuracy scalability, and others. The proposed bitplane codec is competitive with JPEG-LS in lossless compression of 8-bit grayscale and 24-bit color images. The proposed codec is close to JBIG2 in bilevel image compression. It outperforms the existing standards in compression of 8-bit color-quantized images.

key words: *image compression, image coding, context model*

1. Introduction

Image and video transmission is the most traffic over communication channels. Imaging modalities and resolutions continue to grow. High-reality image communications are also emerging. Multiple components [1], high resolution [2]–[5], high dynamic range [6], and high speed [7] are keys for improving the reality of imaging and rendering. They increase the data rates of digital images. To cope with the increased information, data-partitioning of the image content and partial access to it are now recognized as a desirable functionality for image compression systems.

Although the rendering quality of received images is desired as high as possible, a user may want different quality levels in decoding even in a single download transaction. Progressive transmission and decoding is one of the solutions. These backgrounds are driving forces for a lossy-to-lossless codec that works for various types of images.

The purpose of this paper is to present a simple prototype codec that tries to answer the relevant issues. An immediate benchmark performance is assumed as follows.

1) The complexity is lower than JPEG2000 [8], [9].
2) It performs as well as JPEG-LS [10], [11] for lossless compression of color/grayscale images.
3) It performs better than GIF for lossless compression of color-quantized images.
4) It performs as well as JBIG/JBIG2 [12], [13] and is easier to use than JBIG2 for lossless compression of bilevel images.
5) Some functionalities are equipped with it as many as JPEG2000. For example, multiple components are admissible, bit depth is variable and a region-of-interest (ROI) is accurate.

The rest of the paper is organized as follows. The method of the proposed bitplane coding is presented in Sect. 2, where the context model is created by a scheme of bit modeling by pixel values. Brief notes for compression of various types of images are given in Sect. 3. Experimental results are given in Sect. 4 in comparison with some standard systems on compression performances and demonstrations on enhanced functionality. Conclusions follow in Sect. 5.

2. Simple Bitplane Coding

2.1 Underlying Guidelines

The compression performance is a significant concern on various types of image contents. On the other hand, the simplicity of a codec system is another criterion both for low-power dissipation and for simple developments of various functionalities on the core of the codec system. It is reasonable to choose adaptive arithmetic coding [14]–[17] for compatibility between high performance and generic applicability.

Bitplane coding is natural for a simple way to data partitioning. It is advantageous with respect to a fast estimation of probabilities and the immunity against context dilutions [18]–[21]. Assume that an 8-bit image is scanned on adaptive arithmetic encoding. Only two symbols of zero and one appear in bitplane coding. It is enough to estimate either one of their probabilities. In contrast, 256 symbols are involved in pixel-value coding. When the same number of pixels are visited in both types of coding, there are much more chances in updating the frequency counts of individual symbols in the bitplane coding than in the pixel-value coding. As the number of visited pixels increases, a faster probability estimation is possible in the bitplane coding.

The small alphabet size is also advantageous to prevent from excessively degrading the coding efficiency, which can be caused by an insufficient number of samples. It is valid, even if the original image may be partitioned into multiple pieces. It is also hopeful to make a locally adaptive estimation of the sample distribution in an image. In addition, the alphabet size of bitplane coding is the smallest. As far

---

Manuscript received September 27, 2011.
Manuscript revised January 6, 2012.

[†] The authors are with the Department of Electrical and Electronic Engineering, Niigata University, Niigata-shi, 950-2181 Japan.
a) E-mail: kikuchi@eng.niigata-u.ac.jp

DOI: 10.1587/transfun.E95.A.938

as context-based coding is concerned, the average number of samples in individual models is inversely proportional to a product of the number of context models and the alphabet size. Hence, if the alphabet size is small, the context dilution is likely to be suppressed.

The above mentioned two elements, that is, adaptive arithmetic coding and bitplane coding, are combined into a system as a context-based adaptive binary arithmetic coding of separate bitplanes. Additional functionalities are also desirable to meet the requirements on data-partitioning and partial access including tiling, ROI and progressive transmission. The profiles of the proposed codec are summarized in Table A· 1 in Appendix A in comparison with some standard codecs.

2.2 Bit Modeling by the Pixel Value Estimates

The pixel value at location (i, j) in a given image to be encoded is written by $x(i, j)$. Its decoded value is written by $y(i, j)$. Since the location indexes are trivial, they are omitted in the later description. The raster scanning order is used for visiting pixels.

As the process of the bitplane coding proceeds to lower bitplanes, the decoded value, y, of the target pixel approaches the true pixel value, $x = (x_8\, x_7\, x_6 \cdots x_1)$, where x_n denotes the nth bit of x in the case of an 8-bit grayscale image†.

In the proposed bitplane coding, every bit to be encoded is modeled by two sorts of contexts. The contexts are built by the estimates of partially-decoded pixels.

2.2.1 Neighborhood Context

The pixel location of a target bit, x_n where $n \in \{1, 2, \cdots, 8\}$, to be encoded is labeled by 0 on the 9-pixel template shown in Fig. 1. The neighborhood context bit is generated as follows.

$$q_i = \begin{cases} 1, & \text{for } y_i > y \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $i \in \{1, 2, \cdots, 9\}$ denotes the spatial location on the template. y_i and y represent the latest estimates of the neighboring pixels and the target pixel, respectively.

There exist two basic sorts of context modeling in image coding and are listed in the top two rows in Table 1. The most significant feature of the proposed bitplane coding is found in the context modeling, which is explained as follows.

In typical predictive coding [10], [11], [18]–[25], pixel values or prediction errors are used for generating contexts, because pixel values are strongly correlated. It is referred to as *pixel modeling by pixel values*. However, since the alphabet size is large, the context modeling with a large template suffers from the context dilution problem.

On the other hand, the neighboring bit patterns on the present bitplane used to be applied to the context modeling in many sorts of bitplane coding [8], [9], [12], [13], [26]–[29]. It is a scheme of *bit modeling by bit patterns*. However, since the correlation among neighboring bits on a bitplane is weak in natural grayscale images, it is hopeless to use a large template.

		8		
	5	2	6	9
7	1	0	3	
		4		

Fig. 1 Nine-pixel template for the neighborhood context.

Table 1 Approaches in context modeling.

Purpose	Method
pixel-encoding	pixel modeling by pixel values
bit-encoding	bit modeling by bit patterns
bit-encoding	bit modeling by pixel values
bit-encoding	bit modeling by bitplane connectivity

In the proposed bitplane coding, the inter-bit correlation on a bitplane is not used. Instead, for modeling a target bit, we use the pixel value estimates of which more significant bits have been already available at the decoder. The method is referred to as *bit modeling by pixel values*. The pixel value estimates are used instead of unknown true values at the decoder. Since the decoded pixel values are spatially correlated to each other as significantly high as true pixel values, it makes sense to apply a large template. At the same time, since the alphabet size is only two, it is highly probable to avoid the context dilution. Furthermore, the 9-pixel template in Fig. 1 covers a few non-causal locations with respect to the scanning order of pixels. Since context bits are defined by using the estimates of pixel values, it is possible to make a reference to the pixels at non-causal locations. This type of reference is impossible in any scheme of bit modeling by bit patterns or pixel modeling by pixel values.

2.2.2 Related Works

A similar but earlier study by Yoo et al. [30], [31] listed in Table 1 was informed by a reviewer of this paper. In order to make the differences clear, a brief explanation is developed below.

At first, their objective is to encode a *simple image* rather than a natural image. According to their definition, a simple image means the image which is represented by a sparse subset of the available intensity values.

Secondly, their context is made of *bitplane connectivity* for the pixel pairs on the 8-connective neighborhood. On encoding the n-th bitplane, the connectivity value is 1, if a pair of pixels of which values are truncated at the nth significant bit have the same value, and 0 if different. This point is an important difference between their modeling and the proposed one. In their method, the probability of the coincidence in the comparison is expected to be quite low. Hence their modeling is effective for *simple images* rather

---

†Note that it does not matter how high the bit depth is.

than natural images. In contrast, our comparison whether it is larger or not directly depends on the local correlation among neighboring pixel values.

Thirdly, their neighborhood template is smaller than ours especially toward the northern and western directions. A related but minor issue is on the southeast and southwest pixels. They use them, but we do not. According to our investigations, the context bits generated by those pixels were less helpful in our modeling, and therefore those locations were omitted from the template.

The fourth difference is their complex and special modeling for encoding the MSB. The detail is omitted here, since the method is complicated and is combined with an additional device referred to as a *matching indicator*.

Besides the above mentioned points, they introduces *scalable bitplane reduction* which is a sort of *histogram compaction* to improve the performance up to a competitive level to JPEG-LS.

In summary, both the method by Yoo et al. and the proposed one exploit the more significant bits. The context is generated by the coincidence between the MSBs in their method, whereas, in this work, it is generated by the value comparison between pixel values rather than a bit pattern along the bit depth.

2.2.3 Estimation of Pixel Values

Suppose that the nth bitplane is being encoded at present. For every pixel, the higher bits up to the $(n+1)$th bitplane have been known at the decoder. The other lower n bits are unknown. The value of the unknown part distributes over the interval of $[0,\ 2^n-1]$. We assume that zero and one occur with the equal probability in the unknown less significant n bits. Under this assumption, the pixel value estimate of the target pixel is expressed by

$$y^{(n)} = \left\lfloor \frac{y}{2^n} \right\rfloor 2^n + 2^{n-1} - 1 \tag{2}$$

at the nth bit-plane encoding and decoding, where y is the latest decoded value and $\lfloor \cdot \rfloor$ denotes truncation. The first term in the right-hand side of Eq. (2) represents the decoded bits that are exactly equal to the significant bits in x. The other part is the expectation value of the unknown part under the assumption of an equal distribution of unknown binary symbols. Although the expectation value for the unknown LSBs affects nothing for making the context bits, the pixel value representation by Eq. (2) is advantageous in developing several functionalities. The binary representation of the pixel value estimate is given in Fig. 2 in the case of $n = 4$. When the initial value, $y^{(8)}$, is computed by Eq. (2), y can be arbitrary, as long as it is an 8-bit integer. Actually the value of y affects nothing for setting the initial value.

After the target bit, x_n, has been encoded/decoded, the target pixel value estimate, y, is immediately updated by a simple bit operation: the nth and $(n-1)$th bits of y are replaced with x_n and 0, respectively. It is expressed by the following substitution.

$$y^{(4)} = \begin{array}{|c|c|c|c|c|c|c|c|}\hline x_8 & x_7 & x_6 & x_5 & 0 & 1 & 1 & 1 \\ \hline \end{array}$$

Fig. 2 Binary representation of a pixel value estimate in the case of $n = 4$.

$$y \leftarrow y + x_n 2^{n-1} - \lfloor 2^{n-2} \rfloor. \tag{3}$$

The pixel value estimate is used in a coming chance of reference and will be the decoded pixel value, when decoding is stopped.

2.2.4 Control of the Template Size

The non-causal pixels are useless in encoding the most significant bitplane. They are skipped in such a case.

The inter-pixel correlation decreases at lower bitplanes in most natural images where the bit depth is usually eight. The number of template pixels are hence gradually decreased as 8, 7, 6, and 5 for encoding four less significant bitplanes, respectively. If the numbers on the template in Fig. 1 are not larger than the present number of template pixels, the numbered locations are active for context modeling.

2.2.5 Self Context

Highlight areas and shadow areas in an image are probable to represent different objects, respectively. According to this observation, a target bit being encoded is conditioned by the magnitude of the estimate y of the target pixel. When K-bit conditioning is used, the excursion of the tone scale is divided into 2^K successive intervals. The K-bit self context for D-bit grayscale images is hence defined by

$$q_0 = i, \quad \text{for } 2^{D-K} i \le y < 2^{D-K}(i+1), \tag{4}$$

where $i \in \{0, 1, \cdots, 2^K - 1\}$. The self context is empirically helpful in precise estimation of the statistical bias in the distribution of pixel values.

2.2.6 Context Model

The context model of a target bit is defined by a sequence of the neighborhood context bits followed by the self context bits. The context model is expressed by a binary number, which is a simple concatenation of the context bits as follows.

$$q = (q_1, \cdots, q_J, q_{01}, \cdots, q_{0K}), \tag{5}$$

where q_i and q_{0k} stand for the neighborhood context bits and the self context bits, respectively. J denotes the number of reference pixels on the template for the neighborhood context.

$J = 9$ and $K = 3$ are default values for the pixel bit depth of $D = 8$. As a result, at most $2^{J+K} = 4096$ context models are created for conditioning a target bit in binary arithmetic coding. One may wonder if the number of contexts is extraordinarily many. However, it is a fact as will be demonstrated in Sub-sect. 2.4 and it is one of the features of

```
$n \leftarrow$ the bit depth
Set the pixel value estimates by Eq. (2)
While $n > 0$ {
    // Input the code bit-stream
    Initialize the frequency counts
    for all pixels {
        Neighborhood context by Eq. (1)
        Self context by Eq. (4)
        Define the model $q$ by Eq. (5)
        Binary arithmetic encoding of $x_n$ by $q$
        Update the pixel value estimate by Eq. (3)
    }
    Output the code bit-stream
    $n \leftarrow n - 1$
}
```

Fig. 3 Pseudo code of the encoding algorithm. A double-slush implies a comment and no operation is executed. To get the pseudo code of decoding, activate it, instead, comment out the *output* line and change *encoding* to *decoding*.

Table 2 Effects of bit modeling onto bit rate reduction.

Test image	Bit rates in bits per pixel			
	context-free	self context only	modeling by bits	modeling by values
airplane	7.010	6.459	4.835	3.896
baboon	7.938	7.268	6.616	6.028
balloon	7.657	7.102	4.136	2.984
bank	7.721	7.505	5.470	4.645
barb	7.851	7.443	5.707	4.795
barb2	7.879	7.351	5.615	4.735
camera	7.218	6.782	5.164	4.282
couple	6.373	6.352	4.651	3.798
goldhill	7.440	7.061	5.480	4.536
lena	7.803	7.509	5.639	4.645
lennagrey	7.958	7.370	5.345	4.283
noisesquare	5.779	5.608	5.126	5.110
peppers	7.923	7.463	5.465	4.522
shapes	7.076	6.653	1.725	1.239
us021	6.933	6.615	5.751	5.379
us092	6.450	6.018	5.270	4.744
Average	7.313	6.910	5.125	4.351

the proposed bitplane coding.

2.3 Encoding and Decoding Procedures

The most significant bitplane is selected for encoding at first. Every bit on a bitplane is modeled by the previously described contexts. It is sent to an adaptive binary arithmetic encoder as well as its model of Eq. (5).

The frequency counts are initialized in the adaptive binary arithmetic encoder[†], when the bitplane changes. All the initial probabilities are set to be equal. The arithmetic encoder encodes an input bit, and updates the cumulative frequency counts. Once the present bitplane has been encoded, the encoder outputs a sequence of code bits. The encoding bitplane is switched to the next lower bitplane, and the procedure is repeated. The pseudo code of the encoding algorithm is given in Fig. 3. One will see that it is simple.

Decoding shares the same algorithm with encoding. As noted in the legend of Fig. 3, the *input* operation is activated instead of the *output* operation, and *encoding* is replaced with *decoding*.

2.4 Experimental Evidence That Supports the Proposed Context Modeling

In the proposed context-based bitplane coding, every bit being encoded is modeled by the pixel value estimates of which pixels are located at its immediate neighborhood. This is a key in the combination of bitplane coding and adaptive binary arithmetic coding [33], [34].

In this subsection, a supplementary experiment is conducted to demonstrate the superiority over the conventional scheme of *bit modeling by bit patterns.* In Table 2, three extreme cases are compared with the proposed scheme of *bit modeling by pixel values* (in short, modeling by values in the table), which is listed in the rightmost column. The first of them is a scheme of context-free model where no context modeling is applied. The second is the case where only the self context modeling is applied and the neighborhood context is completely suppressed in the algorithm in Fig. 3. The third is a combination of the self context and *bit modeling by bit patterns*, where the neighborhood context is developed by the causal bits on the present encoding bitplane in use of the template in Fig. 1 because the non-causal bits are unavailable in this method.

At first as seen in Table 2, the self context evidently works well. As for the neighborhood context, it is verified that the context modeling is very effective in the combination of adaptive binary arithmetic coding and bitplane coding. In the experiment of actual compression of grayscale images, two types of context modeling resulted in bit rate saving of 1.785 and 2.559 bpp on average over the neighborhood context-free compression. Their difference is 0.774 bpp and is significantly large. It is, hence, evident that the scheme of *bit modeling by pixel values* is superior to that of *bit modeling by bit patterns* on the bitplanes.

2.5 Distortion Estimates in Decoding

In the proposed bitplane coding, encoding and decoding can be ceased at any bitplane. The average distortion of pixels in a decoded image is known before decoding or encoding. The distortion estimate of a decoded 8-bit grayscale image is given by

$$d \simeq 6\ell + 9 \tag{6}$$

in terms of of PSNR in dB, where ℓ is the number of decoded bitplanes. It gives a rough lower bound of PSNR of a decoded image. The proof is given in Appendix B.

[†] The adaptive binary arithmetic codec in our prototype is the range coder [32].

3. Compression of Various Images and Some Functionality Extensions

The simple bitplane coding presented in the previous section is applied to compression of various types of images. They are color images, color-quantized images, and bilevel images including digital halftone images by ordered dithering and error diffusion. Additional adaptations to those image contents are described in this section as well as the implementations of some functionality extensions.

3.1 Compression of Color Images

Color image compression is performed by applying the bitplane coding to individual color components after a color component transform [35]–[37]. It is defined by

$$\begin{pmatrix} E \\ M \\ N \end{pmatrix} = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 1 & -1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}, \tag{7}$$

where R, G, and B are red, green, and blue components, respectively. It is reversible in the strict sense of integer pixel values, if it is implemented in a lifting form as follows.

$$\left.\begin{aligned} &M \leftarrow R - G \\ &N \leftarrow B - G \\ &E \leftarrow G + \left\lfloor \frac{M+N}{3} \right\rfloor \end{aligned}\right\} \tag{8}$$

where a single fixed-point multiplication suffices. The chroma components, M and N, are represented in signed-magnitude integers. The sign bit information is encoded independently of the magnitude information.

Multi-component images comprising four or more primaries can be efficiently compressed by an application of an effective component transform. Once a specific multi-component color space is given, it is possible to find a qualified and reversible component transform in the form of lifting [37].

Even after the decorrelation by the component transform, there exist weak dependencies among color components. For example, a considerable level of noisy behavior of pixel values is often observed over shadow areas in an image. Similarly, the color tends to saturate or to converge to achromatic colors in highlight areas. In order to exploit these dependencies, an additional one-bit shade context is defined as follows.

$$q_s(c) = \begin{cases} 1, & \text{for } y(E) < s \ \text{ or } \ y(E) + s \geq 2^D \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where c denotes anyone of E, M and N in Eq. (8) and $y(E)$ denotes the estimate of the luma, respectively. s is a threshold value and is given by 2^{D-k}, where D and k are the bit depth of the luma and the number of the present self context bits, respectively. To implement the shade context, the luma component is encoded earlier than the chroma components.

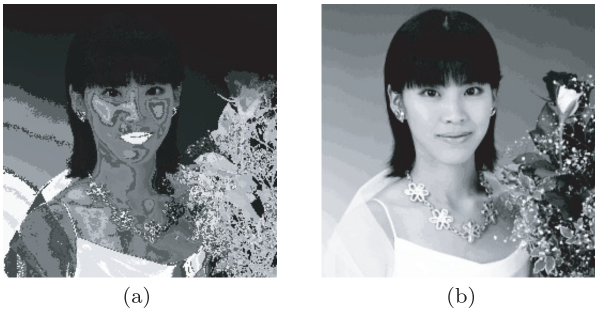

(a) (b)

Fig. 4 Palette index images. (a) before sorting, and (b) after sorting.

3.2 Compression of Color-Quantized Images

Eight-bit color-quantized images are one of categories of color images as used in the GIF format. A color table (color palette) is used to map true 24-bit color values onto 8-bit color indexes. Note that labeling methods for color indexes are independent of color value mapping. Consequently, when the color index values are used as substitute for tone values, a resulting image displayed in grayscale looks abnormal in appearance, as shown in Fig. 4(a).

A preprocessing is hence introduced so that a processed image may show visually natural appearance. From a color value triplet of (R, G, B) in a given color table, a wide-sense luma [38]

$$L = R + G + B \tag{10}$$

is defined, and the color table is re-indexed by sorting the 256 entries in ascending order of the luma. The proposed technique is a kind of palette reordering [39] and is the simplest among many methods[†].

As observed in part (b) of Fig. 4, the new indexes work as if the resultant image were a grayscale image so that a generic bitplane coding works well in compression.

3.3 Compression of Bilevel Images

Bilevel images are often produced in a variety of categories for their own purposes such as found in printing, facsimile transmission, segmentation and document archiving. Typical categories of bilevel images include error diffusion halftones, clustered-dot dither halftones, character-based documents, and line/drawing artworks.

As a consequence, neighboring pixel values are probable to correlate to each other. The statistical nature of bilevel images is different from that of less significant bitplanes of natural scene images. Since the bitplane to be encoded is single, both non-causal pixels and self contexts are helpless and are thus dropped. Instead, an extended 15-pixel template is defined for the neighborhood context as shown in Fig. 5.

[†] Much better methods are available, if the computational time is not significant [39], but a simpler system is the concern in this work.

			14			
	9	10	6	11	12	
	8	2	3	4	7	15
13	5	1	0			

Fig. 5 Fifteen-pixel template for bilevel image compression.

It is worth to mention that the proposed bitplane coding is option-free unlike JBIG2 and is thus easy-to-use for generic users. This point is significantly different from JBIG2, where there are many coding engines including JBIG [12], halftone [13], MMR[†], SPM[††] [29] and others[†††].

3.4 Functionality Extensions

Three major functionality extensions are described in this subsection.

3.4.1 Progressive Transmission

Since the data partitioning capability with respect to bitplanes and color components is inherently equipped with the proposed bitplane coding, the functionality of progressive transmission requires none of additional processing. It is implemented by transmitting the separate code streams of different bitplanes as individual data.

Progressive decoding is just a result of a successive decoding of separate bitplanes for rendering a decoded image at a desired level of bit depth. The pixel accuracy is incrementally refined bitplane by bitplane. The average distortion estimate over a decoded image is available in Eq. (6) prior to encoding. This benefits a wide range of users, because it is easy to use.

3.4.2 Arbitrarily Shaped Region-of-Interest

The shape of an ROI is arbitrary. It does not matter, even if the ROI may be a collection of disjointed dots and pieces of a jigsaw puzzle as illustrated in Fig. 6. Since none of spatial transforms are involved with the proposed coding, a blur-free ROI is decoded.

The ROI information is a binary map of one-bit depth. It is stacked on top of the most significant bitplane of an image being encoded. Hence the ROI information and the image bitplanes are concatenated along the bit depth, as if the bit depth of the image is extended. When an RGB color image is encoded, the binary ROI map is basically stacked on every component. If different ROI maps are applied to individual color components, they are stacked on respective components. In this way, the ROI information is embedded into the image content.

In addition to the ROI-embedded data form, a separate data form of an ROI and an image content is allowed as an option.

An ROI-embedded image is encoded in one of two ways. The first is to skip the non-ROI areas, as if the background of the ROI were occupied by a pixel value of a uniform intensity. The other is to encode the entire area including the non-ROI areas. In this case, the foreground and the background of an ROI are decodable in different accuracies, as long as the bit depth for the foreground is longer than or equal to that for the background. Progressive decoding of ROI areas is also possible.

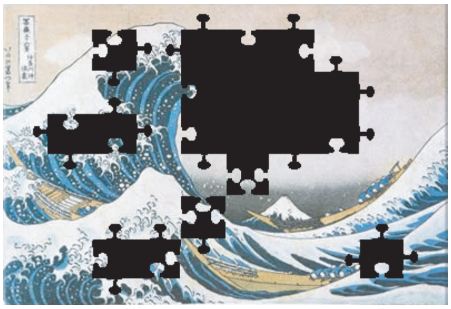


Fig. 6 An example of a decoded image by an arbitrarily shaped ROI.

3.4.3 Selective Tile Partitioning

Tile partitioning is implemented, if all ROIs are defined to cover the whole image without overlaps, where the shape of a unit tile is arbitrary. The tiling pattern is specified by a tile unit and its offset. The horizontal and vertical offset values are set to define the left-top position where the tiled region begins.

When tile partitioning is applied, tiled areas are separately and independently encoded. Tiles are fully or partially selectable for encoding and decoding.

Owing to the arbitrary shape of ROI and selective tiling, coding units can be defined object by object in an image so that the object scalability would be developed as an advanced functionality.

4. Experimental Results

The performances of the proposed bitplane coding in lossless compression of six categories of images are investigated in comparison with existing standard systems. Demonstrations on some functionality extensions are also presented.

4.1 Lossless Compression Performances

4.1.1 An Overview

A brief overview on the average bit rates in lossless compression of six categories of images are presented in Table 3 for comparing the proposed codec with existing standards.

[†] modified modified READ

[††] soft pattern matching

[†††] A user of JBIG2 is requested to specify various options to get the best fit to image contents to be encoded. Depending on the contents located at different areas in an image, qualified coding algorithms are applied to the different areas after their segmentations. If the default settings are chosen, the potential of JBIG2 is hardly exploited.

Table 3 Average bit rates in lossless compression.

Category of test images	Bit rates in bpp/c[1]		Reference codec
	Proposed	Reference	
24-bit color	**4.229**	4.352	JPEG-LS
8-bit grayscale	**4.351**	4.413	JPEG-LS
8-bit color (gif)[2]	**4.931**	5.430	JPEG-LS
bilevel halftone[3]	**0.470**	0.475	JBIG2
bilevel halftone[4]	0.309	**0.303**	JBIG2
bilevel document[5]	0.054	**0.043**	JBIG2

[1] Bit rates are in bits/pixel per component for 24-bit color and color bilevel halftone images.

[2] The average bit rate of GIF files is 5.899 bpp including header information which is equivalent to 0.025 bpp for the set of 14 test images.

[3] Floyd-Steinberg error diffusion

[4] Ordered dither by the 40-dot/18° CMY orthogonal screen set

[5] CCITT fax test set (200 dpi)

Detailed data for respective image categories will be presented after the overview. Note that the reference codec stands for the best performing codec among JPEG2000 [40], JPEG-LS [41], JBIG [42], and JBIG2[†] in respective image categories.

As for JBIG2, it should be noticed that any verification models are unavailable to the public and the parameter setting of options depends on image contents. Hence comparisons to JBIG2 were carefully conducted on the basis of literature, if the literature and test images are available.

Table 3 summarizes the lossless compression results for various types of images, while the separate data for individual image categories are presented in the following subsections. As seen in Table 3, the proposed codec is competitive to every standard codec that performs best for each category of images.

4.1.2 Grayscale Images

The lossless compression performance of the proposed method on grayscale images is listed in Table 4, where accompanied are the bit rates of JPEG-LS that is the best performing standard codec. Most of 16 test images are the popular USC set[††]. As seen in the table, the performances of the proposed method and JPEG-LS are competitive in lossless compression of grayscale images.

4.1.3 24-Bit Color Images

The lossless compression performance of the proposed method on 24-bit color images is listed in Table 5, where bit rates of JPEG-LS, which is the best performing standard codec, are also given. Twenty-two test images mainly comprise of the USC set, ISO/JIS-SCID [43] and Sony sRGB standard images [44]. Reversible color component transforms used are EMN of Eq. (8) for the proposed bitplane coding and GMN[†††] for JPEG-LS, respectively, since GMN is recommended in the standardization [11]. Again, the proposed method and JPEG-LS are competitive in lossless compression of color images.

Table 4 Bit rates in grayscale image compression.

Test image	Resolution	Bit rates in bpp	
		Proposed	JPEG-LS
airplane	512×512	3.896	3.817
baboon	512×512	6.028	6.036
balloon	720×576	2.984	2.904
bank	256×256	4.645	4.812
barb	720×576	4.795	4.691
barb2	720×576	4.735	4.686
camera	256×256	4.282	4.314
couple	256×256	3.798	3.699
goldhill	720×576	4.536	4.477
lena	512×512	4.645	4.607
lennagrey	512×512	4.283	4.238
noisesquare	256×256	5.110	5.683
peppers	512×512	4.522	4.513
shapes	512×512	1.239	1.214
us021	640×480	5.379	5.897
us092	640×480	4.744	5.024
Average	–	4.351	4.413

Table 5 Bit rates in color image compression.

Test image	Resolution	Bit rates in bpp/c[1]	
		Proposed in EMN	JPEG-LS in GMN
N1A, woman	1536×1920	4.288	4.424
N2A, cafe	1536×1920	4.978	5.173
N3A, fruits	1920×1536	4.362	4.490
N4A, wine	1920×1536	4.300	4.439
N5A, bike	1536×1920	4.302	4.347
N6A, orchid	1920×1536	3.806	3.971
N7A, musicians	1920×1536	5.460	5.690
N8A, candle	1920×1536	4.900	5.134
Sony party 4x	2048×1536	3.681	3.721
Sony picnic 4x	2048×1536	3.865	3.926
Sony portrait 4x	2048×1536	3.594	3.654
airplane	512×512	3.768	3.772
baboon	512×512	5.955	6.071
barbara	720×576	3.388	3.571
boats	720×576	2.955	3.184
couple	256×256	3.894	3.900
girl	256×256	4.510	4.470
goldhill	720×576	3.299	3.596
lena	512×512	4.466	4.583
peppers	512×512	4.810	4.925
Average	–	4.229	4.352

[1] Bit rates are in bits/pixel per component.

4.1.4 8-Bit Color-Quantized Images

Eight-bit color-quantized test images were generated by a combination of median cut and error diffusion-quantization in Jasc Paint Shop Pro. A majority of the 14 test images are

---

[†] Reference [42] for halftones and Refs. [29] and [47] for document images

[††] http://sipi.usc.edu/database

[†††] $$\begin{pmatrix}G\\M\\N\end{pmatrix} = \begin{pmatrix}0 & 1 & 0\\1 & -1 & 0\\0 & -1 & 1\end{pmatrix}\begin{pmatrix}R\\G\\B\end{pmatrix}$$

Table 6 Bit rates in bpp of 8-bit color-quantized image compression.

Test image (GIF)	GIF	Proposed		JPEG-LS
		sort-free	luma-sort	luma-sort
goldengate	6.214	6.257	5.693	6.251
milkdrop	5.013	4.558	4.477	5.122
tiffany	6.504	6.119	5.284	5.769
baboon	8.024	6.836	6.666	6.924
lena	6.782	6.181	5.083	5.485
airplane	5.531	5.684	4.829	5.401
peppers	6.464	6.311	5.082	5.616
boy	5.300	5.058	4.759	5.266
couple	7.181	6.375	5.432	5.844
girl	6.500	5.967	5.436	5.811
goldhill	6.288	6.090	5.161	5.511
portrait	5.477	5.132	4.566	5.040
screenshot	1.677	1.923	1.507	2.498
yacht	5.628	5.883	5.056	5.475
Average	5.899	5.598	4.931	5.430

Table 7 Bit rates of color bilevel error-diffusion halftones.

Test image	Resolution	Bit rates in bpp/c[1]		
		Proposed	JBIG	JBIG2
N1A, woman	1536×1920	0.487	0.514	0.483
N2A, cafe	1536×1920	0.557	0.582	0.561
N3A, fruits	1920×1536	0.432	0.467	0.437
N4A, wine	1920×1536	0.441	0.465	0.435
N5A, bike	1536×1920	0.429	0.461	0.433
N6A, orchid	1920×1536	0.375	0.431	0.389
N7A, musicians	1920×1536	0.520	0.562	0.533
N8A, candle	1920×1536	0.543	0.572	0.554
Sony party 4x	2048×1536	0.446	0.498	0.447
Average	–	0.470	0.506	0.475

[1] Bit rates are in bits/pixel per component.

Table 8 Bit rates of color bilevel ordered-dither halftones.

Test image	Resolution	Bit rates in bpp/c[1]		
		Proposed	JBIG	JBIG2
N1A, woman	1536×1920	0.299	0.372	0.293
N2A, cafe	1536×1920	0.406	0.440	0.409
N3A, fruits	1920×1536	0.279	0.315	0.261
N4A, wine	1920×1536	0.278	0.315	0.266
N5A, bike	1536×1920	0.301	0.330	0.293
N6A, orchid	1920×1536	0.229	0.278	0.218
N7A, musicians	1920×1536	0.344	0.398	0.349
N8A, candle	1920×1536	0.404	0.435	0.409
Sony party 4x	2048×1536	0.241	0.312	0.233
Average	–	0.309	0.355	0.303

[1] Bit rates are in bits/pixel per component.

Table 9 Compression ratios of bilevel document images.

Test image (bilevel)	Content attribute	Compression ratio		
		Proposed	JBIG	JBIG2
f01_200	textual	37.3	38.6	54.3
f02_200	line-art	61.8	61.1	59.9
f03_200	textual	24.6	25.0	31.6
f04_200	textual	10.5	10.3	17.1
f05_200	textual	21.3	21.5	29.5
f06_200	line-art	40.0	42.3	43.0
f07_200	textual	10.2	9.6	13.0
f08_200	line-art	36.7	37.4	36.5
f10_200	mixture	9.1	7.9	9.8[1]
Average	———	18.5	17.7	23.2

The data of JBIG2 except for *f10* owes to Ref. [29].

[1] The value is cited from Ref. [47].

the USC set. The lossless compression bit rates are shown in Table 6 where the GIF file sizes including header information are also shown for the purpose of reference. The proposed codec outperforms all of JPEG2000, JPEG-LS, JBIG, and JBIG2. Note that the bit rates for JPEG-LS were obtained by preprocessing of color table sort in Sect. 3.2. If plain GIF images are fed to JPEG-LS, the results are disappointing.

4.1.5 Color Bilevel Halftone Images

As for color bilevel halftones, neither adequate test images nor reliable compression data are available in the literature. Hence, color bilevel halftone images have been generated by two popular methods: Floyd-Steinberg error diffusion [45] and clustered-dot ordered dithering by the 40-dot/18-degree Cyan-Magenta-Yellow orthogonal screen set [46]. Nine test images comprise of ISO/JIS-SCID and *party_4x* of Sony sRGB standard images. The JBIG and JBIG2 codecs for the experiments are Leadtools v.15 [42], where 3-line/13-pixel and 3-line/16-pixel templates have been applied to error diffusion halftones and clustered-dot ordered dither halftones, respectively, since those were the best among others with respect to bit rates.

The experimental results are listed in Table 7 for the error diffusion halftones and in Table 8 for the clustered-dot ordered dither halftones. As found in the tables, the proposed method outperforms JBIG in lossless compression of both types of bilevel halftones. It slightly outperforms JBIG2 in compression of error-diffusion halftones and is slightly inferior to JBIG2 in compression of ordered dither halftones.

4.1.6 Bilevel Document Images

Bilevel document images are finally tested. They are CCITT facsimile test images [47] at 200 dpi-resolution, among which *f02, f06* and *f08* are line-art images, *f10* is a mixture of texts and dither halftones, and the others are textual images.

The compression performances are compared in Table 9, where the compression ratios of JBIG were actually obtained by Leadtools v.15 [42]. Those of JBIG2 are borrowed from JBIG2 (MQ) [47] for *f10* and from JBIG2 (SPM) [29] for the others, respectively†.

As seen in the table, the proposed bitplane coding is competitive to JBIG in lossless compression of bilevel document images. It is inferior to JBIG2 by 20% but is free from content-dependent intensive computations. The excellence in JBIG2 is found in compression of textual images that are targeted by the SPM algorithm.

---

†The data for *f10* is unavailable in Ref. [29].

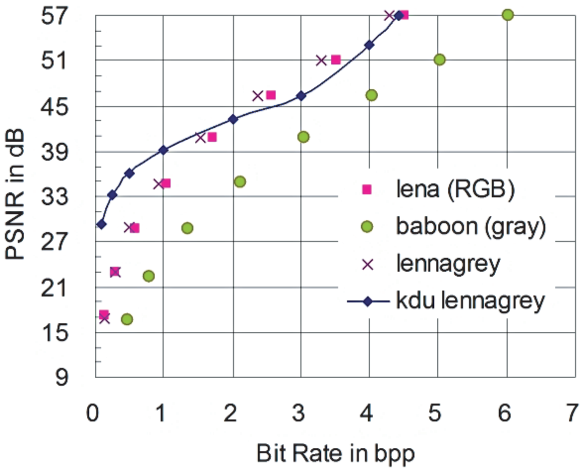


Fig. 7 Distortion-rate plot in the progressive transmission. kdu refers to a JPEG2000 codec, Kakadu version 2.2 in [9].

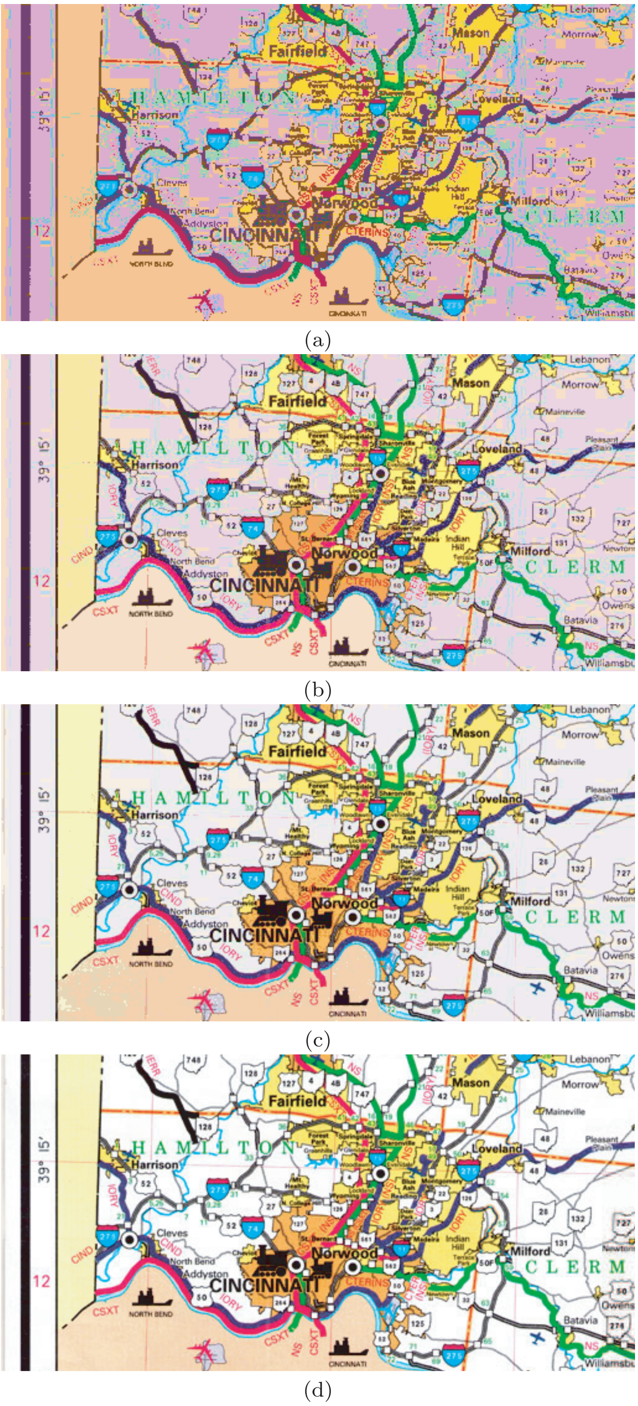
4.2 Progressive Transmission

Progressive transmission is one of the functionality extensions to lossless image compression by bitplane coding. Figure 7 shows the distortion plot against bit rates[†] in the case of progressive transmission. The progression level is specified by the number of decoded bitplanes. The plots at the top edge in the figure represent the bit rates at which lossless decoding is attained and hence the value of PSNR should be read as infinity. The horizontal lines with each 6dB spacing in PSNR represent the distortion estimates given by Eq. (6).

As observed in the figure, the distortion-rate plots form multiple clusters. Each of them distributes on a specific level of distortion, even if image contents are different. It has been demonstrated that the before-the-fact estimate of distortion agrees with the actual distortion after decoding. In addition, every plot locates at a level higher than its corresponding distortion estimate. This fact means that the distortion estimate before encoding guarantees the worst level of distortion in decoding of a progressively transmitted image.

While the proposed codec is inferior to JPEG2000 at low bit rates, it is satisfactory at high bit rates. The performance plots of the proposed coding and JPEG2000 cross around PSNR= 43dB. It is reasonable, since lossless compression and high-quality applications are assumed to be the major target of the proposed codec.

A few decoded samples of a part of a color railway map[††] image are shown in Fig. 8. As seen in the figure, decoded pictures are blur-free even at the earliest level of decoding, because no spatial filtering is involved with the proposed codec. The color balance has been considerably improved in the 2nd level of decoding as seen in part (b). The third level decoding presents a satisfactory rendition desired for map information in spite of quite low PSNR.



(a)

(b)

(c)

(d)

Fig. 8 Progressive transmission of a part of *railmap*. The progression levels, decoding bit rates, and decoding distortions in dB are as follows. (a) 1st level, 0.343 bpp, and 13.64 dB, (b) 2nd level, 0.597 bpp, and 20.05 dB, (c) 3rd level, 0.936 bpp, and 26.45 dB, and (d) 8th level (lossless decoding), 4.455 bpp, and ∞ dB.

[†] Bit rate of a 24-bit color image stands for bpp per component.

[††] The Ohio Department of Transportation, http://www2.dot.state.oh.us/map1/OhioRailMap/

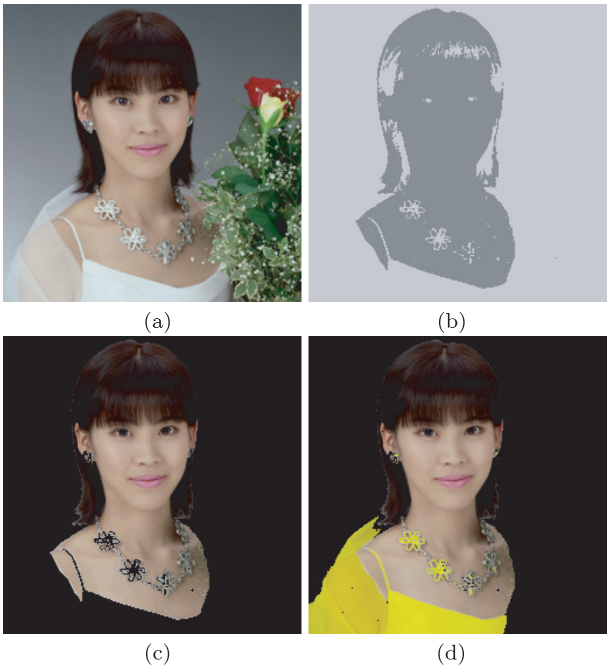


Fig. 9 Arbitrarily shaped ROI transmission. (a) original, (b) ROI, (c) ROI-decoded image, and (d) another ROI-decoded image where the ROI for blue component does not cover the dress region.

4.3 Arbitrarily Shaped ROI Transmission

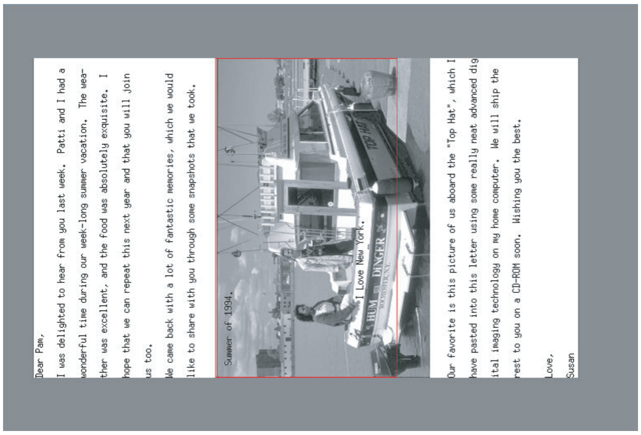
An example of lossless transmission of an arbitrarily shaped ROI is given in Fig. 9, where parts (a), (b), and (c) show the original color image, ROI, and the ROI-decoded image decoded up to upper 5 bitplanes, respectively. The bit rates of the original image itself (a), the ROI-embedded image in progressive transmission, and the progressively decoded image (c) are 3.766, 3.899, and 0.569 bpp per component, respectively.

The ROI mask in part (b) has been embedded into three color components of the original image, and the bit rate has increased by 0.133 bpp/component. It can be reduced to 0.026 bpp/component, if the ROI mask is separately encoded as a bilevel image and is shared with three components.

Additional regions of interest are acceptable to be combined. Part (d) is such an example, where two color components of red and green of the dress region have been added as the second ROI and combined with the first ROI. As a result, the dress region has been decoded in yellowish color. The ROI-embedded image with different ROIs on three components has been transmitted/decoded up to five more significant bitplanes, which is shown in part (d). The bit rate is 0.625 bpp per component in total.

4.4 Partitioning into Tiles

As an example, tiling is applied to *cmpnd.pgm*, a 512 × 768-pixel image. The tile unit is a 384×218-pixel block of which



Dear Pam,
I was delighted to hear from you last week. Patti and I had a
wonderful time during our week-long summer vacation. The wea-
ther was excellent, and the food was absolutely exquisite. I
hope that we can repeat this next year and that you will join
us too.
We came back with a lot of fantastic memories, which we would
like to share with you through some snapshots that we took.

Summer of 1994.
I Love New York.

Our favorite is this picture of us aboard the "Top Hat", which I
have pasted into this letter using some really neat advanced dig
ital imaging technology on my home computer. We will ship the
rest to you on a CD-ROM soon. Wishing you the best.

Love,
Susan

Fig. 10 A sample of tile partitioning divided into three pieces. The image has been rotated to save space. The second tile is marked by a red rectangle to ease the identification of three tiles.

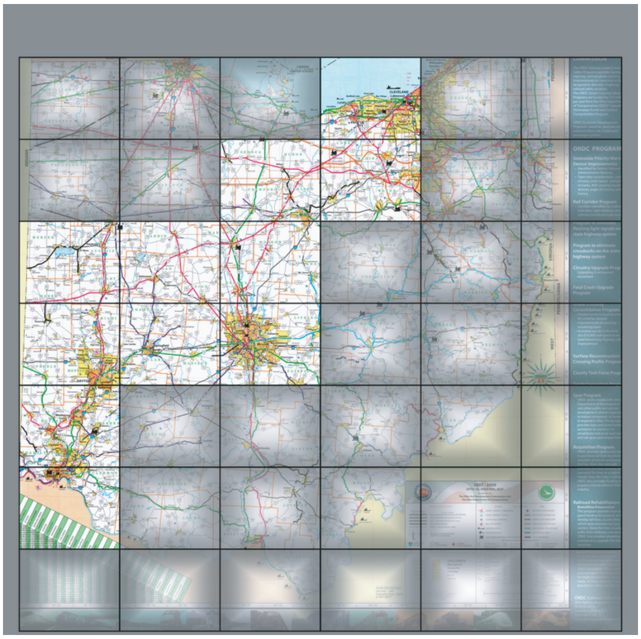


Fig. 11 A sample of selective tile partitioning divided into 6 columns and 7 rows. Eleven unshaded tiles on the way to Cincinnati from Cleveland have been selected for encoding and/or decoding.

horizontal and vertical offsets are 64 and 36 pixels, respectively. The tile-partitioned and encoded areas are the central part of the original image as shown in Fig. 10. They have been encoded in 1.889 bpp on average over 3 pieces.

If the whole area covered by three tiles is encoded at once, it is encoded into 1.935 bpp. As visually expected from the image, the middle area is completely different from the other letter areas. Since separate encoding of those three tiles allows the encoder to learn the statistically different trends in those tiles, tile-partitioned encoding has a chance of doing efficient compression.

The selective tiling functionality is demonstrated in Fig. 11. The encoded image is a railway map of 2225×2216-

Table 10 Near-lossless encoding and decoding cycle effects. JPEG-LS was operated in near-lossless mode with the maximum allowable error of 2, excepting the fifth data row. The test image is *lennagrey*.

Codec	# of cycles	Max. Abs. Error	PSNR in dB	Bit rate in bpp
JPEG-LS	1	2	45.14	2.095
	2	4	42.37	2.147
	3	6	40.79	2.202
	4	8	39.64	2.258
JPEG-LS	1	4	40.12	1.533
Proposed	—	2	46.37	2.361
	—	4	40.74	1.537
	—	8	34.67	0.935

pixel resolution. The tiling offset is (55, 186) and the tile unit is a 352 × 288-pixel CIF-sized block. Eleven unshaded tiles on the route of Cleveland to Cincinnati have been selected for encoding and/or decoding out of 42 tiles in total. The progression level for decoding is upper 5 bitplanes. The average bit rate over 11 selected tiles is 1.985 bpp/component. The decoded data of the selected tiling amounts to 12.8% of that for lossless compression of the whole image. The distortion by the progressive decoding of the selected areas is 38.33dB in terms of PSNR, and fine visual quality is presented. This type of functionality is useful for the navigation of traffic information and pathological images.

4.5 Repetition-Resilient Near-Lossless Compression

Simplified near-lossless compression is possible in the bitplane coding. The maximum distortion at pixels is simply described by insignificant bitplanes. It is worth to note that no degradation will appear any more, even if multiple applications of near-lossless encoding and decoding may be repeated with the same or finer accuracy. This is a contrast to JPEG-LS, where the repetition of near-lossless encoding/decoding causes an increase in distortion.

For example, near-lossless compression of JPEG-LS is applied to *lennagrey.pgm*. It is compressed into 2.095 bpp under the maximum allowable error of 2, and the distortion is 45.14dB in PSNR, as shown in Table 10. When the decompressed image is once again compressed by the same allowable error, the bit rate, the distortion, and the maximum error are degraded to 2.147 bpp, 42.37dB in PSNR, and 4, respectively. As a cycle of encoding and decoding is repeated, the pixel error increases by the allowable error limit.

Needless to mention, the first-time compression is fine in near-lossless JPEG-LS. For reference, the case of maximum allowable error of 4 is also given in the fifth row of data in the table. Also, remember that it is impossible for JPEG-LS to get efficient lossless compression of the data that has been once compressed in the near-lossless mode.

In the proposed bitplane coding within the error limit of 2, the image is encoded into 2.361 bpp and the distortion is 46.37dB. Although the bit rate is higher than that of JPEG-LS, no further distortion is generated, even if encoding and decoding may be repeated.

5. Conclusions

A simple method of context-based bitplane coding has been presented for lossless image compression as well as for functionality extensions. Conditioning contexts for encoding and decoding are formed by the estimates of neighboring pixels instead of bit patterns on relevant bitplanes. While the algorithm is simple, the proposed method not only results in good compression performances with respect to various types of images, but also allows the codec system to gain a variety of data-partitioning functionality including bitplane scalability, arbitrary shaped ROI transmission, progressive transmission, selectable tiling, and others. As for lossless compression of grayscale and 24-bit color images, presently the proposed codec is competitive with JPEG-LS. Also, in compression of color-quantized images, it outperforms the existing standards.

An improvement in the compression performance and implementations of more functionality extensions such as object scalability are promising issues. The applications to HDR images, SHD images, multi-component images, high-quality video, and 3-dimensional voxel-based image data will be hopeful.

Acknowledgments

The first author thanks Professor Masayuki Tanimoto of Nagoya University for his comment that none of transforms would be the best to exploit correlations. It was a hidden driving force to seek for such a simple method. The authors thank Mr. Kunio Funahashi, Meisei Electric Co. Ltd., who was involved with the study as a graduate student. They would like to thank the reviewers for their comments that were valuable for improving the quality of presentation of the work.

References

[1] M. Yamaguchi, H. Haneishi, and N. Ohyama, "Beyond red-green-blue (RGB): Spectrum-based color imaging technology," J. Imaging Sci. and Tech., vol.52, no.1, pp.010201-1–15, Jan./Feb. 2008.

[2] S. Ono and N. Ohta, "Super high definition image communications — A platform for media integration," IEICE Trans. Commun., vol.E76-B, no.6, pp.599–608, June 1993.

[3] S. Ono, N. Ohta, and T. Aoyama, Super-High-Definition Images: Beyond HDTV, Artech House, Boston, 1995.

[4] Digital Cinema Initiatives (DCI), http://www.dcimovies.com

[5] U. Catalyurek, M.D. Beynon, C. Chang, T. Kurc, A. Sussman, and J. Saltz, "The virtual microscope," IEEE Trans. Inf. Tech. Biomed., vol.7, no.4, pp.230–248, Dec. 2003.

[6] E. Reinhard, G. Ward, S. Pattanaik, and P. Debevec, High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting, Morgan Kaufmann, San Francisco, 2006.

[7] http://www.sony.net/Products/SC-HP/cx_news/vol47/featuring.html

[8] D. Taubman, "High performance scalable image compression with EBCOT," IEEE Trans. Image Process., vol.9, no.7, pp.1158–1170, July 2000.

[9] D.S. Taubman and M.W. Marcellin, JPEG2000, Kluwer Academic, Boston, 2002.
[10] M.J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," IEEE Trans. Image Process., vol.9, no.8, pp.1309–1324, Aug. 2000.
[11] ISO/IEC 14495-1, "Information technology — Lossless and near-lossless compression of continuous-tone still images: Baseline," Dec. 1999.
[12] ISO/IEC 11544, Information technology — Coded representation of picture and audio information — Progressive bi-level image compression, March 1993.
[13] ISO/IEC 14492, Information technology — Coded representation of picture and audio information — Lossy/lossless coding of bi-level images, July 1999.
[14] J. Rissanen, "A universal data compression system," IEEE Trans. Inf. Theory, vol.29, no.5, pp.656–664, Sept. 1983.
[15] G. Martin, "Range encoding: An algorithm for removing redundancy from a digitised message," Proc. Video and Data Recording Conf., pp.24–27, Southampton, March 1979.
[16] I.H. Witten, R.M. Neal, and J.G. Cleary, "Arithmetic coding for data compression," Commun. ACM, vol.30, no.6, pp.520–540, 1987.
[17] A. Moffat, R.M. Neal, and I.H. Witten, "Arithmetic coding revisited," ACM Trans. Inf. Syst., vol.16, no.3, pp.256–294, July 1998.
[18] X. Wu and N. Memon, "Context-based, adaptive, lossless image coding," IEEE Trans. Commun., vol.45, no.4, pp.437–444, April 1997.
[19] X. Wu, "Lossless compression of continuous-tone images via context selection, quantization, and modeling," IEEE Trans. Image Process., vol.6, no.5, pp.656–664, May 1997.
[20] N. Memon and X. Wu, "Recent developments in context-based predictive techniques for lossless image compression," The Computer J., vol.40, no.2/3, pp.127–136, 1997.
[21] X. Wu and N. Memon, "Context-based lossless interband compression-extending CALIC," IEEE Trans. Image Process., vol.9, no.6, pp.994–1001, June 2000.
[22] B. Aiazzi, L. Alparone, and S. Baronti, "Context modeling for near-lossless image coding," IEEE Signal Process. Lett., vol.9, no.3, pp.77–80, March 2002.
[23] B. Aiazzi, L. Alparone, and S. Baronti, "Near-lossless image compression by relaxation-labelled prediction," Signal Process., vol.82, no.10, pp.1619–1631, Oct. 2002.
[24] S. Forchhammer, X. Wu, and J.D. Andersen, "Optimal context quantization in lossless compression of image data sequences," IEEE Trans. Image Process., vol.13, no.4, pp.509–517, April 2004.
[25] I. Matsuda, Y. Umezu, N. Ozaki, J. Maeda, and S. Itoh, "A lossless coding scheme using adaptive predictors and arithmetic code optimized for each image," IEICE Trans. Inf. & Syst. (Japanese Edition), vol.J88-DII, no.9, pp.1798–1807, Sept. 2005.
[26] A. Said and W.A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," IEEE Trans. Circuits Syst. Video Technol., vol.6, no.3, pp.243–250, June 1996.
[27] K. Shinoda, H. Kikuchi, and S. Muramatsu, "A lossless-by-lossy approach to lossless image compression," ICIP 2006, pp.2265–2268, Atlanta, Oct. 2006.
[28] K. Shinoda, H. Kikuchi, and S. Muramatsu, "Lossless-by-lossy coding for scalable lossless image compression," IEICE Trans. Fundamentals, vol.E91-A, no.11, pp.3356–3364, Nov. 2008.
[29] P.G. Howard, "Text image compression using soft pattern matching," The Computer J., vol.40, no.2/3, pp.146–156, 1997.
[30] Y. Yoo, Y. Kwon, and A. Ortega, "Embedded image-domain adaptive compression of simple images," Proc. 32nd Asilomar Conf. on Signals, Syst. & Computers, pp.1256–1260, 1998,
[31] Y. Yoo, Y. Kwon, and A. Ortega, "Embedded image-domain compression using context models," Proc. ICIP 1999, pp.477–481, Kobe, Oct. 1999.
[32] M. Lundqvist, http://www.geocities.com/mikaellq/
[33] K. Funahashi, H. Kikuchi, and S. Muramatsu, "Progressive bitplane coding for lossless image compression," IEICE Technical Report, SIP2008-39, June 2008.
[34] K. Funahashi, H. Kikuchi, and S. Muramatsu, "Bitplane coding conditioned by decode pixel values for lossy-to-lossless image compression," 23rd Picture Coding Symp. Japan, Izu, pp.17–18, Oct. 2008.
[35] N. Liu, Fractal Imaging, Academic Press, New York, 1997.
[36] H. Kikuchi, J. Hwang, S. Muramatsu, and J. Shin, "Reversible component transforms by the LU factorization," 28th Picture Coding Symp. (PCS 2010), pp.238–241, Nagoya, Dec. 2010.
[37] J. Hwang, H. Kikuchi, S. Muramatsu, K. Shinoda, and J. Shin, "Reversible implementations of irreversible component transforms and their comparisons in image compression," IEICE Trans. Fundamentals, vol.E95-A, no.4, pp.824–828, April 2012.
[38] H. Kikuchi, K. Funahashi, and S. Muramatsu, "Generic bitplane coding based on bit modeling by the decode expectation values of pixels — Lossless compression of color-quantized and bilevel images," 23rd Signal Proc. Symp., pp.409–414, Kanazawa, Nov. 2008.
[39] A.J. Pinho and A.J.R. Neves, "A survey on palette reordering methods for improving the compression of color-indexed images," IEEE Trans. Image Process., vol.13, no.11, pp.1411–1418, Nov. 2004.
[40] http://jj2000.epfl.ch/
[41] http://www.hpl.hp.com/loco/
[42] http://www.leadtools.com/
[43] Graphic Technology — Prepress Digital Data Exchange — Standard Colour Image Data, ISO/JIS-SCID, JISX9201-1995, Japanese Standards Association, Tokyo, 1995.
[44] Sony sRGB Standard Images. http://www.colour.org/tc8-04/test_images/Sony/
[45] R. Floyd and L. Steinberg, "An adaptive algorithm for spatial grey scale," Proc. Soc. Inf. Display, vol.17, p.75, 1976.
[46] H.R. Kang, Digital Color Halftoning, SPIE Press and IEEE Press, 1999.
[47] http://www.imagepower.com/technology/

Appendix A: Profiles of the Proposed Codec

The profiles of the proposed prototype codec are viewed in Table A· 1. The following description is a supplement to the table and the main body of the paper.

The bitplane scalability is a simple consequence of bitplane coding where every bitplane is encoded and transmitted separately and independently. The receiver decodes a sequence of separate code bit-streams of individual bitplanes to improve the accuracy of decoded pixels.

Separate coding of bitplanes will result in a sort of near-lossless functionality where errors in decoded pixels are bounded by the powers of two that correspond to the decoded bit depth. The pixel error is created by the substitution of insignificant bitplanes by a uniform expectation value. As a consequence, cyclic repetitions of encoding and decoding will change nothing, once the image has been encoded. In contrast, the near-lossless compression in JPEG-LS is subjected to individual pixel values, and the repetition resilience is lost.

To implement the resolution scalability, data-partitioning corresponding to a specified resolution is necessary. Since none of spatial transforms are involved with the proposed bitplane coding, any choice of resolution scalability is optionally possible. For example, the octave scalability is possible to be introduced by a reversible wavelet trans-

Table A· 1 Profiles of the proposed bitplane codec compared to existing codecs.

Functionality Extension	Proposed	JPEG2000	JPEG-LS	JBIG2
Complexity in encoding/decoding	✪✪✪	✪✪	✪✪✪✪	✪
Compression of grayscale images	✪✪✪	✪✪✪	✪✪✪	✪
Compression of color images	✪✪✪	✪✪✪	✪✪✪	✪
Compression of color-quantized images	✪✪✪✪	✪	✪✪	✪✪
Compression of bilevel documents	✪✪✪	✪	✪✪	✪✪✪✪
Compression of bilevel halftones	✪✪✪	✪	✪	✪✪✪✪
Near-lossless compression (NLC)	Yes	N.A.	Yes	N.A.
NLC repetition resilience	Yes	N.A.	No	N.A.
Tile partitioning	selectable	whole	N.A.	N.A.
Resolution scalability	optional	Yes	N.A.	Yes
Pixel accuracy scalability	bit-plane	SNR	N.A.	pseudo-gray
Arbitrary shape of ROI	crispy	blurry	N.A.	N.A.
ROI transmission	bit-depth	Max-Shift[1]	N.A.	N.A.
Progressive transmission	coarse	fine	N.A.	coarse
Distortion in progressive decoding	known	unknown	N.A.	unknown

[1] The scaling option is also available at the expense of strong blurring effect.

✪✪✪✪ excellent, ✪✪✪ good, ✪✪ fair, ✪ poor.

form. The FFT and DCT are simple alternatives for the resolution scalability with arbitrary scales. The scale of resolution scalability depends on applications and image contents. Even if an extra cost in bit rates may be charged by an optional resolution scalability, it is inexpensive as long as the scale is around a few octaves. It is the case for high-resolution images such as SHD images including digital cinema and virtual slides.

As for the computational complexity in encoding and decoding, it is higher than that of representative predictive coding such as JPEG-LS, because pixels are scanned bitplane by bitplane. The computational complexity required for scanning bitplanes is considered to be tolerable at the expense of the enhanced functionalities. Note that the bitplane scanning in EBCOT [8] is a core of efficient scalable coding equipped with a variety of functionalities. Nevertheless, the proposed bitplane coding is of lower complexity than JPEG2000, since none of spatial transforms are involved.

Appendix B: Proof of Eq. (6)

The pixel distortion in decoding the information of upper bitplanes is equivalent to the error that is generated by truncating lower bits of a pixel value. As the truncation level decreases by one bit, PSNR increases by 6 dB. Hence the average distortion in terms of PSNR is expressed by $d = 6\ell + c$, where ℓ denotes the number of decoded bitplanes and c is a constant.

Assume that decoding is at the level of $\ell = 7$. It is equivalent to decode all bitplanes excepting the LSB. The pixel error in the decoded image is either 0 or 1, and a uniform distribution of the error is assumed in the discrete sample space. The mean squared error is thus found to be $0^2 \times 0.5 + 1^2 \times 0.5 = 0.5$. It means that $d = 10 \log \frac{255^2}{0.5} \simeq 10 \log 2^{17} \simeq 51$. Hence one finds that $c = 9$ and obtains Eq. (6).

Appendix C: Test Images

The thumbnail images of the test images used in this work are shown for traceability. Those of grayscale images, 24-bit color images, 8-bit color-quantized images, color bilevel halftones, and bilevel document images are shown in Figs. A· 1–A· 5, respectively.

Fig. A· 1 Grayscale test images.

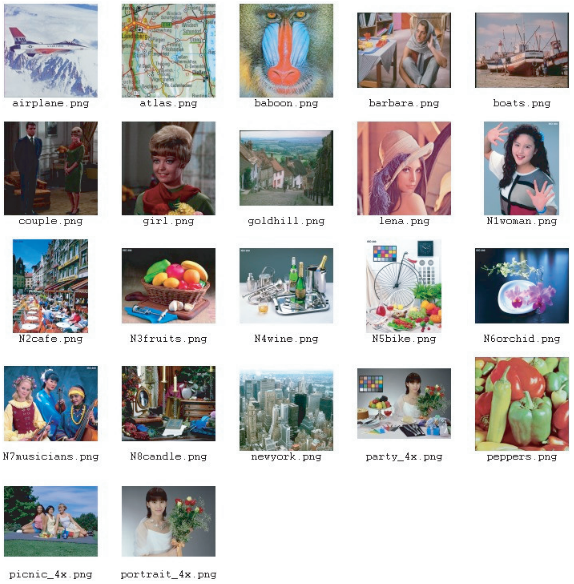


Fig. A· 2 24-bit color test images.

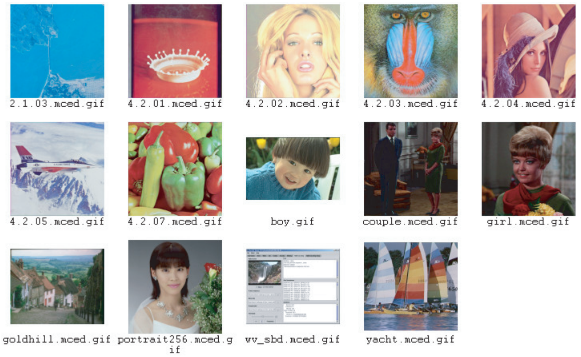


Fig. A· 3 8-bit color-quantized test images (GIF).

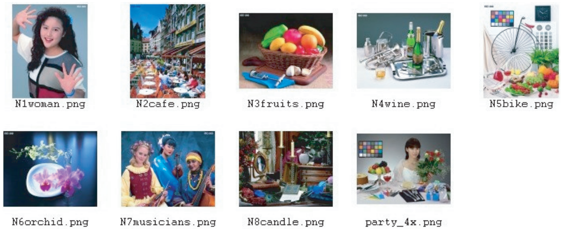


Fig. A· 4 Color bilevel halftone test images.

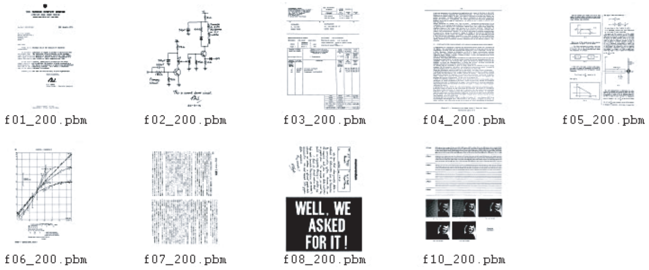


Fig. A· 5 Bilevel document test images.

Hisakazu Kikuchi was born in Niigata, Japan, in 1952. He received B.E. and M.E. degrees from Niigata University, Niigata, in 1974 and 1976, respectively, and Dr. Eng. degree in electrical and electronic engineering from Tokyo Institute of Technology, Tokyo, in 1988. From 1976 to 1979 he worked at Information Processing Systems Laboratory, Fujitsu Ltd., Tokyo. Since 1979, he has been with Niigata University, where he is a Professor in the Department of Electrical and Electronic Engineering. During the 1992 academic year, he was a visiting scholar in the Electrical Engineering Department, University of California, Los Angeles, USA. He holds a visiting professorship at Chongqing University of Posts and Telecommunications and Nanjing University of Information Science and Technology, both in China, since 2002 and 2005, respectively. His research interests include digital signal processing and image processing. He is a Member of ITE (Institute of Image Information and Television Engineers of Japan), IIEEJ (Institute of Image Electronics Engineers of Japan), and IEEE. He served the general co-chair of ITC-CSCC 2011 in Korea, the chair of Circuits and Systems Group, IEICE, in 2000 and the general chair of Digital Signal Processing Symposium, IEICE, in 1998 and Karuizawa Workshop on Circuits and Systems, IEICE, in 1996, respectively.

Ryosuke Abe was born in Tokamachi, Japan, in 1988. He received B.E. degree in electrical and electronic engineering from Niigata University, Niigata, in 2010. He studies to obtain M.E. degree. His research interests are in the field of image/video processing and compression.

Shogo Muramatsu was born in Tokyo, Japan in 1970. He received B.E., M.E. and Dr. Eng. degrees in electrical engineering from Tokyo Metropolitan University, Tokyo, in 1993, 1995 and 1998, respectively. In 1997, He joined Tokyo Metropolitan University. In 1999, he joined Niigata University, where he is currently an Associate Professor in the Department of Electrical and Electronic Engineering. He was a Visiting Researcher at MICC & VIP Laboratory, University of Florence, Italy, during the 2003 academic year. His research interests are in digital signal processing, multirate systems, image processing and VLSI architecture. He is a Member of ITE (Institute of Image Information and Television Engineers), IIEEJ (Institute of Image Electronics Engineers of Japan), IPSJ (Information Processing Society of Japan), and IEEE.