Copyright ⓒ 2018 KSII

JPEG-based Variable Block-Size Image Compression using CIE *La*b** Color Space

Samruddhi Y. Kahu[1] and Kishor M. Bhurchandi[1]
[1] Department of Electronics and Communication Engg., Visvesvaraya National Institute of Technology
Nagpur, Maharashtra 440010 - India
[e-mail: samruddhikahu@gmail.com]
*Corresponding author: Samruddhi Y. Kahu

Received January 12, 2018; revised March 26, 2018; accepted April 16, 2018;
published October 31, 2018

Abstract

In this work we propose a compression technique that makes use of linear and perceptually uniform CIE *La*b** color space in the JPEG image compression framework to improve its performance at lower bitrates. To generate quantization matrices suitable for the linear and perceptually uniform CIE *La*b** color space, a novel linear Contrast Sensitivity Function (CSF) is used. The compression performance in terms of Compression Ratio (CR) and Peak Signal to Noise Ratio (PSNR), is further improved by utilizing image dependent, variable and non-uniform image sub-blocks generated using a proposed histogram-based merging technique. Experimental results indicate that the proposed linear CSF based quantization technique yields, on an average, 8% increase in CR for the same reconstructed image quality in terms of PSNR as compared to the conventional *YCbCr* color space. The proposed scheme also outperforms JPEG in terms of CR by an average of 45.01% for the same reconstructed image quality.

Keywords: JPEG, Image Compression, CIE *La*b** Color Space, Contrast sensitivity function, Quantization.

1. Introduction

Since its inception in 1993, JPEG has become a widely used image compression format in internet and multimedia applications [1-3]. As a lot of applications and work-flows have already been developed around it, its widespread usage remains unabated [4] despite the emergence of image compression standards such as JPEG 2000, and HEIC (HEVC still image compression standard). The basic block diagram of JPEG is shown in **Fig. 1**. As noted in **Fig. 1**, plenty of research effort has been made on improving the performance of almost all the major blocks of JPEG. In [5], trapezoid and triangular blocks are used instead of the regular 8×8 blocks in JPEG. Shape Adaptive DCT (SA-DCT) has also been used in conventional image/video coding algorithms since the 1990s. In [6], image is segmented using splitting and merging technique and SA-DCT is used to transform the segmented arbitrary shaped image regions. SA-DCT along with overlapped transform is also used in [7] where smooth regions of the image are downsampled and then transformed. Superpixel Driven Graph Transform (SDGT) is used in [8] instead of SA-DCT. In [8], image is segmented using the graph technique and each segment is transformed using SDGT. Authors in [9] have proposed the use of fuzzy logic for approximating the 2D-DCT used for transforming image sub-blocks in JPEG. Discrete Hartley Transform (DHT) is proposed in [10] as an alternative to DCT for image sub-block transformation. As DHT is used, another efficient quantization approach and scanning order (in lieu of zigzag scanning order used in JPEG) is proposed in [10]. To enhance the quantization performance, Just Noticeable Difference (JND) based quantization techniques for JPEG have been proposed in [11-13]. A lot of research has been invested to find efficient JND profiles as Human Visual System (HVS) characteristics can be accurately and efficiently modeled using JND. As a result, many DCT based JND models have been proposed in the past [14 – 21]. A new quantization table is generated for JPEG based on a generic psychovisual error threshold in [22]. In [23], the existing transform and quantization operations of JPEG are modified so that only bit-shifts are required for quantization operation. This leads to lower power consumption while encoding/decoding images. In [24], local prediction based adaptive scanning is used instead of conventional zigzag scanning in DCT based compression techniques such as JPEG and H.264/AVC-intra. The Run Length Coding (RLC) technique of JPEG is further optimized in [25] so that the redundancies in the JPEG's RLC technique are further reduced. In [26], four modifications are proposed to improve the efficiency of JPEG's binary arithmetic QM coding. Adaptive Golomb Coding is used as entropy coding in [27] to improve the compression efficiency of JPEG. A saliency based approach to improve the performance of JPEG is proposed in [28] and a block-based image quality metric is used to improve the rate-quality performance of JPEG in [29]. Apart from these techniques, many other optimization techniques have been proposed for the performance improvement of JPEG. Interested reader is referred to [4] for more details.

On the other hand, to the best of the authors' knowledge, not a lot of research is focused on the color space conversion technique used in JPEG. Color space conversion plays an important role in any compression scheme as it represents the color image in a color space to minimize the psycho-visual redundancies. *YCbCr* has been conventionally used for coding of images and videos as it is a luminance-chrominance color space.

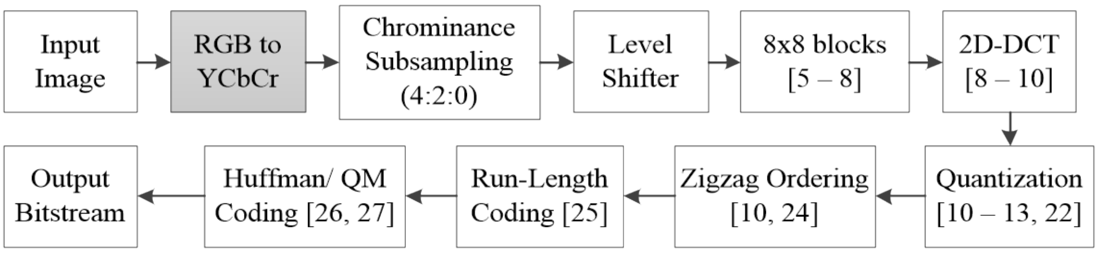


Fig. 1. Basic JPEG block diagram with relevant publications in the corresponding area of research

In this paper, we use a perceptually uniform and linear CIE *La*b** color space for compression of images. Though CIE *La*b** color space has been used in past for compression of images/videos, it has been majorly used for pre-processing while the actual encoding/decoding is done in *YCbCr* color space [34, 35]. As linear and perceptually uniform CIE *La*b** color space is used, we use a linear CSF for generation of the quantization matrices. Linear CSF used in this work is derived from the CSF proposed by Klein et al. [36]. Besides, a novel histogram based approach of dividing the image into variable block sizes is also proposed. Variable block size algorithms have come up in the past years and have been incorporated in [30 – 33]. They are mostly top-down quad-tree decomposition based approaches. These algorithms start from a large, uniform and fixed block called Macro Block (MB) typically of size 16×16 or 32×32. Each MB is then divided into smaller non-uniform blocks according to the image structure. In this process, though the size of image sub-blocks is variable, their positions are fixed. Our algorithm is more compatible to the image structure as both size and position of the image blocks are variable.

To describe the proposed techniques in detail, rest of the paper is organized as follows. In Section 2, we discuss the important techniques and concepts leading to the proposed techniques. Section 3 presents the proposed techniques while experimental results are presented and discussed in Section 4. We conclude this paper in Section 5.

2. Fundamental Methods and Materials

In this section, we review some of the fundamental techniques that are utilized in the proposed work.

2.1 CIE *La*b** Color Space

Images are captured and/or displayed by most of the digital devices in RGB color space since it is an orthogonal color space and hence is hardware friendly. RGB color space being highly correlated is not suitable for compression. Generally, *YCbCr* color space is used by most of the standard image and video compression systems as it is a highly decorrelated luminance-chrominance color space. However, in this paper, we have used CIE *La*b** color space instead of *YCbCr* due to its inherent advantages which are discussed as follows:

1) Like *YCbCr*, *La*b** is also a luminance-chrominance color space. Therefore, it gives high decorrelation and hence chrominance subsampling [37] can be effectively implemented.
2) *La*b** color space is a color-opponent color space with dimensions *L* for lightness and *a** and *b** for the color-opponent dimensions, based on nonlinearly-compressed CIE *XYZ* color space coordinates. The three planes *L*, *a** and *b** have pixel intensity ranges; *L* [0, 100], *a** [-100, 100] and *b** [-100, 100]. Hence, *L* plane can be represented using 7 bits/pixel as against the 8 bits/pixel required by *Y* plane of *YCbCr*. This itself contributes to

a saving of 1 bit/pixel. The values that a pixel can take in *a** and *b** plane are centered on 0. Hence, there is no need of level shifting for *a** and *b** planes unlike *YCbCr* where level shifting is required for all the three planes.

3) CIE *La*b** is a linear and perceptually uniform color space [35]. Hence quantization can be implemented effectively without perceptual loss of visual quality.
4) Another advantage of CIE *La*b** is that it is a device independent color space [38].

Since authentic direct color conversion formulae for converting from RGB to CIE *La*b** color space are not available, we first convert to CIE *XYZ* and then to *La*b**. For conversion to CIE *XYZ* color space, gamma-corrected RGB values are converted to linear RGB values before applying the following conversion formula [39];

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{1}$$

where *R*, *G*, *B* are the linear RGB values.

Conversion from CIE *XYZ* to CIE *La*b** [34, 40, 41] is accomplished using (2), (3) and (4) as discussed below:

$$L = 116 \times f\left(\frac{Y}{Y_n}\right) - 16 \tag{2}$$

$$a^* = 500 \times \left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right) \right] \tag{3}$$

$$b^* = 200 \times \left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right) \right] \tag{4}$$

where X, Y, Z are calculated using (1) above and X_n, Y_n, Z_n are the tri-stimulus values of the adapting white [34, 40, 41]. The function *f* used in (2), (3) and (4) is given as;

$$f(x) = \begin{cases} x^{1/3} & x > 0.008856 \\ 7.787x + \dfrac{16}{116} & x \le 0.008856 \end{cases} \tag{5}$$

2.2 Exponential Golomb Coding

Exponential Golomb codes were first proposed for the representation of non-negative integers with exponentially decaying probability distribution [42]. Exponential Golomb codes offer a host of advantages over other DCT based entropy coding techniques. Firstly, they don't require look-up tables to provide supplemental information to the encoder. And secondly, theoretically they can be used to encode a data source with an infinite number of possible symbols unlike Huffman coding [27]. Given a non-negative integer n, the zeroth order exponential Golomb code of n is represented as $G^0_{\exp}(n)$ and can be computed using following steps:

Step 1: Calculate prefix *m* from the symbol to be coded *n* using,

$$m = \lfloor \log_2(n+1) \rfloor \tag{6}$$

and form the unary code of *m* (unary code of a number *x* is the number of *x* zeros followed by 1).

Step 2: Determine the binary representation of

$$(n+1)-2^m \tag{7}$$

truncated to *m* least significant bits.

Step 3: Concatenated binary representation of results of steps 1 and 2 is the exponential Golomb code of *n*.

2.3 Binary Arithmetic (QM) Coding

Binary arithmetic coding is used as an entropy coding technique in many image/video compression algorithms. QM-coder is a version of binary arithmetic coding used with some implementations of the JPEG standard [3, 43]. The process of binary arithmetic QM coding is presented here in brief.

In binary arithmetic QM coding, instead of coding symbols 0 and 1 directly, they are mapped to most probable symbol (MPS) or least probable symbol (LPS). The basic idea behind this mapping is that when a completely black image (black denoted by 0) has a small white patch (white denoted by 1) in it, 0 is mapped to MPS while 1 is mapped to LPS. Whenever a symbol is input, QM coder first decides whether it is MPS or LPS based on the symbols input in the past with the help of probability estimation tables and then encodes it.

Suppose we have an interval A and the LPS probability estimate is Qe, MPS probability estimate will obviously be 1 – Qe since there are only two possible input symbols. The interval A is then divided into two parts by the QM coder such that LPS sub-interval is A×Qe and the MPS sub-interval is A×(1 – Qe). The positioning and sizes of the sub-intervals according to convention is depicted in **Fig. 2**. The input symbol sequence of MPS and LPS is encoded into a code stream (pointer) C by QM coder. Ideally, C can point anywhere in the current interval (either MPS or LPS) but for ease of operation, QM coder points C at the bottom of the current interval.

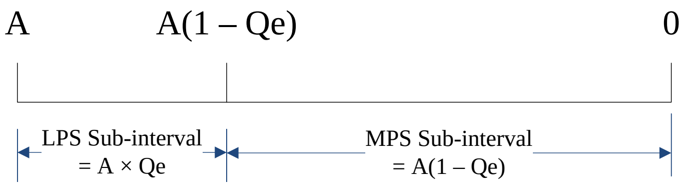


Fig. 2. Interval Subdivision in QM Coder

3. Proposed Work

Flowchart of the proposed JPEG-based image compression scheme is shown in **Fig. 3**. The blocks that are modified as compared to the basic JPEG block diagram in **Fig. 1** are highlighted. The structure of **Fig. 3** is deliberately kept similar to that in **Fig. 1** in order to highlight the modifications and bring out the differences.

In this scheme, classification and subsequent grouping of adjacent 8×8 blocks based on their similarity in terms of mean and variance is incorporated in CIE *La*b** color space. This results in the formation of blocks of 10 different sizes. Most of the HVS based quantization techniques for DCT are proposed for perceptually non-uniform and non-linear *YCbCr* color space and fixed block sizes. We propose modifications to make them suitable for the perceptually linear and uniform CIE *La*b** color space and for the image-dependent non-uniform image sub-blocks, as it is more pragmatic. Important and highlighted (modified) blocks/steps of the algorithm presented in **Fig. 3** are explained below in detail.

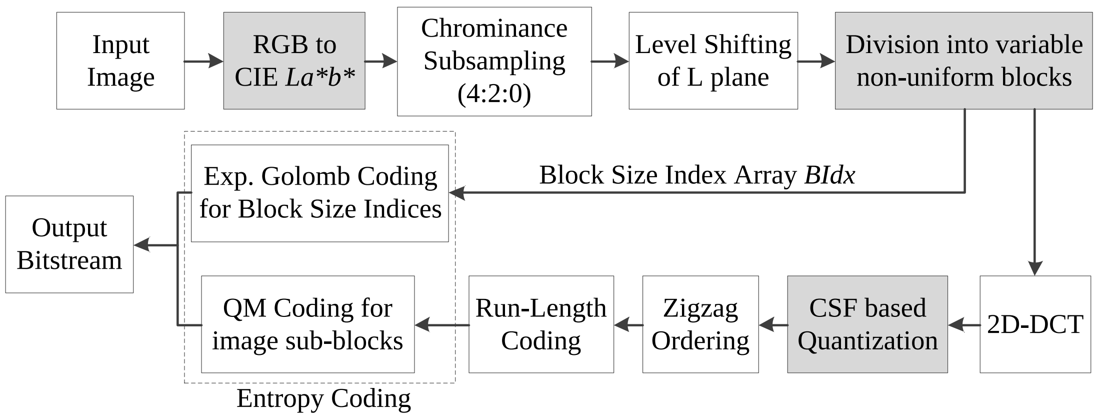

Input Image
RGB to CIE La*b*
Chrominance Subsampling (4:2:0)
Level Shifting of L plane
Division into variable non-uniform blocks
Block Size Index Array BIdx
Exp. Golomb Coding for Block Size Indices
Output Bitstream
QM Coding for image sub-blocks
Run-Length Coding
Zigzag Ordering
CSF based Quantization
2D-DCT
Entropy Coding

Fig. 3. Flowchart of the encoder of the proposed scheme

3.1 Conversion to CIE *La*b** Color Space

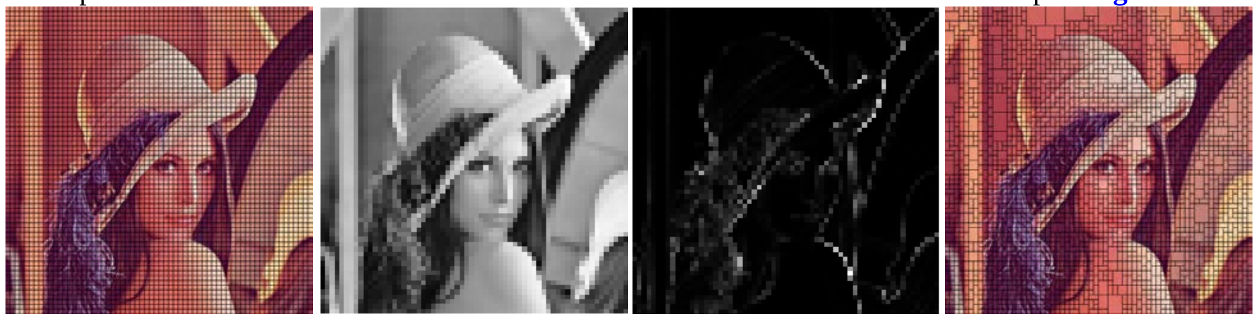
Input RGB image is first converted to CIE *La*b** using (1) to (5) given in section 2.1.

Since human eye is less sensitive to chrominance information than luminance, chrominance subsampling is applied on *a** and *b** planes. Subsampling ratio of 4:2:0 is used [37]. In other words, *a** and *b** planes are subsampled by a factor of 2 both horizontally and vertically.

As mentioned in section 2.1, since the intensity values of *a** and *b** color planes are already centered on zero, there is no need of level shifting for *a** and *b** planes. However, a value of 50 is subtracted from each pixel intensity value of the *L* plane in order to center the mean of L plane on zero, in the proposed algorithm. Rest of the steps of the proposed algorithm are same for level shifted *L* plane and subsampled chrominance planes *a** and *b**.

3.2 Division into variable non-uniform blocks

The proposed algorithm uses blocks of different rectangular shapes and sizes as against the uniform 8×8 sized blocks used by most of the image compression algorithms. Sixteen different block sizes are used in the proposed work, smallest being 8×8 and the largest, 32×32. The process of division into these non-uniform blocks is described below with the help of **Fig. 4**.



(a) Division into 8×8 blocks (b) Mean plane (normalized) (b) Variance plane (normalized) (d) Division into non-uniform blocks

Fig. 4. Process of histogram based merging on Lena

3.2.1 8×8 blocks

Initially, the input image is divided into the smallest size 8×8 non-overlapping uniform blocks just like in JPEG and as shown in **Fig. 4** (a).

3.2.2 Indexed histogram based merging

Mean and variance of each 8×8 block is calculated. Only first two statistical moments i.e. mean and variance are used to describe the characteristics of the 8×8 blocks. Thus, for each sub-plane (*L*, *a** or *b**) of an input image I of size P×Q, we have two planes each of size P/8×Q/8; namely mean plane containing all mean values of 8×8 blocks of the image shown in **Fig. 4** (b) and variance plane having all variances of 8×8 blocks of the image shown in **Fig. 4** (c). Mean and variance values of 8×8 image sub-blocks are displayed in **Fig. 4** (b) and **Fig. 4** (c). These values are normalized to the range 0 to 255 only for display purpose so that the information contained in them is clearly visible. However, for the indexed histogram formation and subsequent grouping, original mean and variance values are used. This data is stored in a 3 dimensional matrix (A) with two P/8×Q/8 planes. Thus, A(1:P/8,1:Q/8,1) denotes the mean plane and A(1:P/8,1:Q/8,2) denotes the variance plane.

A histogram table is formed using these values and statistical bin size thresholds are chosen as discussed further. A sample indexed histogram table structure is shown in **Fig. 5**.

Index No.	Mean	Variance	Population
1	15.64	0.79	50
2	13.72	0.35	217
3	19.42	1.31	41
. . .	. . .	. . .	. . .

Fig. 5. Indexed Histogram Table H with arbitrary data.

The histogram formation process is described as below;

1) Initialize an indexed histogram table structure H with k = 1 row and four columns, namely index, mean, variance and population. The number of rows k goes on increasing if the next block [mean, variance] is different than the earlier registered one [mean, variance] using thresholds Th_{mn} and Th_{var}.
2) Initialize the first entry in the table by the mean and variance of the first 8×8 image block that is the top left corner block in the image and population and index to 1.
   H(1,2:3) = A(1,1,1:2), H(1,4) = 1 and H(1,1) = 1.
3) Scan the matrix A along the rows and read the next mean and variance values stored in A. Compare every new pair of mean and variance stored in matrix A with all the earlier recorded mean and variance pairs in H and increment the population of that row of H which satisfies the condition given in (8), else record mean and variance values at that location of A in the next new row of H and initialize its population to 1.

$$\left(\left|A(m,n,1)-H(k,2)\right|<Th_{mn}\right)AND\left(\left|A(m,n,2)-H(k,3)\right|<Th_{\mathrm{var}}\right) \quad (8)$$

The thresholds Th_{mn} and Th_{var} are obtained by calculating standard deviation of the mean and variance values in the mean plane and variance plane, respectively.

Note that, here, Th_{mn} is used to threshold the difference between mean values and similarly, Th_{var} is used to threshold the difference between variance values. Hence, standard deviation of the respective values is used as the threshold. In addition to the formation of the histogram table H, a new index matrix *Idx* of size P/8×Q/8 is formed simultaneously. *Idx* contains the indices of all the block locations in matrix A, as indicated by histogram table H. These indices in *Idx* are further used for grouping of the adjacent blocks. If indices of the immediate neighbouring blocks (N_4) are same, then blocks are merged to form bigger blocks.

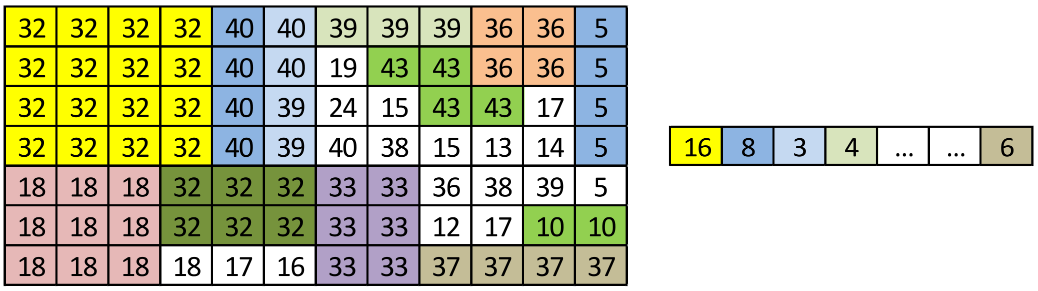
3.2.3 Merging of adjacent blocks

Adjacent blocks are grouped based on the index matrix *Idx* to form blocks of sizes 32×32, 32×24, 24×32, 24×24, 32×16, 16×32, and so on, till 8×8. These block sizes are allotted indices from 15 to 0, respectively. Note that all possible combinations of block sizes starting from 32×32 till 8×8 in steps of 8 are used.

The highest block size chosen is 32×32. This is because sharpness of human vision is highest in the fovea region covering approximately 2° of visual angle. This corresponds to a radius of approximately 40 pixels for $\Lambda = 1.5$ min/pixel for normal viewing distance. Thus a 32×32 image sub-block is faithfully covered by the visual angle [21]. Smallest block size considered is 8×8 in accordance with the JPEG standard. In the above procedure, vertical block sizes such as blocks 32×24, 24×16, 32×16, etc. are considered before horizontal blocks 24×32 or 16×32 as natural images contain more vertical structures than horizontal ones [44]. Experimental analysis shows that natural images contain very less number of block sizes greater than 32×32. Thus, considering block sizes greater than 32×32 will further increase the blocking overhead and the compression performance may deteriorate.

Further, a block size index array, *BIdx*, is constructed using *Idx* and the block indices 0 to 15. *BIdx* indicates the spatial position and its block index in the image and is used, further, for forming the compressed bitstream. The same *BIdx* will be used at the decoder to decode the image. Section of a sample index matrix *Idx* is shown in **Fig. 6** (a). Adjacent blocks having same indices are merged to form bigger non-uniform blocks which are shown using different colors. **Fig. 6** (b) shows a sample block size index array *BIdx* formed using the sample index matrix of **Fig. 6** (a). The algorithm used for merging is diagrammatically explained using **Fig. 7** and summarized in the form of a pseudo code in **Algorithm 1**.

It is clearly evident from **Algorithm 1** that priority is given for the formation of bigger block sizes such as 32×32, 32×24, 24×24 etc. Thus, the order in which blocks are checked for merging based on *Idx* is: 32×32, 32×24, 24×24, 32×16, 16×32, 24×16, 16×24, 16×16, 32×8, 8×32, 24×8, 8×24, 16×8, 8×16 and 8×8. In this way, the complete image is encoded and a block index array named *BIdx* is formed. This block index array is compressed using exponential Golomb coding and sent to the decoder as overhead information along with the compressed image blocks.



32	32	32	32	40	40	39	39	39	36	36	5
32	32	32	32	40	40	19	43	43	36	36	5
32	32	32	32	40	39	24	15	43	43	17	5
32	32	32	32	40	39	40	38	15	13	14	5
18	18	18	32	32	32	33	33	36	38	39	5
18	18	18	32	32	32	33	33	12	17	10	10
18	18	18	18	17	16	33	33	37	37	37	37

(a) Section of a sample index matrix *Idx*

16	8	3	4	...	...	6

(b) Corresponding block size index array *BIdx*

Fig. 6. Sample *Idx* and *BIdx* matrices

Algorithm 1: Grouping of adjacent 8×8 image sub-blocks

```
Input:   Index matrix Idx, Input image plane (L, a* or b*) of size P×Q
Output:  Block size index array BIdx, compressed codestream
begin
    1.  Initially set index variables to zero, i = 0, j = 0 and p = 1.
    Following steps are executed until all the 8×8 blocks of the image are encoded:
    2.  If all values of matrix Idx(i:i+3,j:j+3) are same
        Code 32×32 block of the image plane;
        BIdx(p) = 16;
        j = j + 3;
    3.  Else check all values of matrix Idx(i:i+3,j:j+2), if same
        Code 32×24 block of the image plane;
        BIdx(p) = 15;
        j = j + 2;
    4.  Else check values of matrix Idx(i:i+2,j:j+3), if same
        Code 24×32 block of the image plane;
        BIdx(p) = 14;
        j = j + 3;
    5.  Else check values of matrix Idx(i:i+2,j:j+2), if same
        Code 24×24 block of the image plane;
        BIdx(p) = 13;
        j = j + 2;
    6.  …
    7.  …
        …
    10. Code 8×8 block of the image plane;
        BIdx(p) = 0;
        j = j + 1;
    11. p = p + 1; Increment i appropriately;
end
```

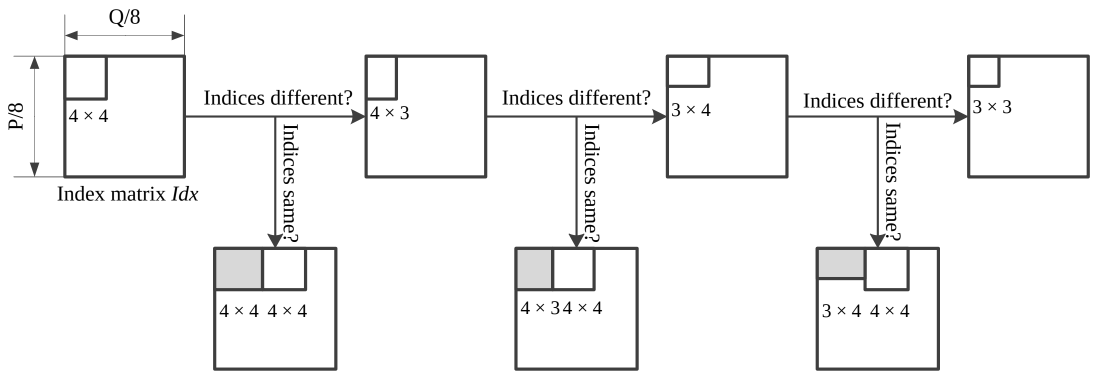
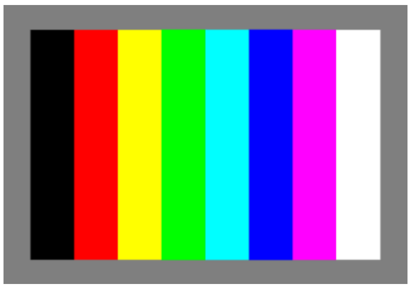
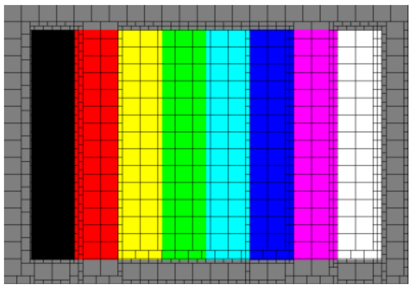


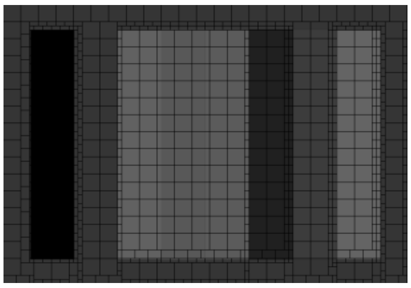
Fig. 7. Example Grouping of adjacent 8×8 image sub-blocks using index matrix *Idx*

Thus, after similarity based merging of adjacent 8×8 blocks, the input image gets divided into non-uniform blocks as seen in **Fig. 4** (d). The above splitting and merging algorithm is implemented on the level shifted *L* plane and subsampled *a** and *b** planes separately as all the three planes are compressed separately. **Fig. 4** (d) shows the results of division of *L* plane only. Colored image is shown instead of the gray scale luminance image for better understanding. From **Fig. 4** (d), it can be clearly seen that smooth regions have larger blocks whereas edges and textured regions get divided into smaller blocks, more specifically, 8×8 blocks. Horizontal and vertical edges get divided into preferably 8×32, 8×24, 8×16 and 32×8, 24×8, 16×8 sized blocks, respectively.

However, division into non-uniform blocks may not be properly justified from **Fig. 4** (d) as regions appearing uniform are shown to be divided into smaller blocks. Hence, a simple color image shown in **Fig. 8** (a) is used to illustrate the result of division into non-uniform blocks. **Fig. 8** (b) shows the results of splitting and merging algorithm implemented on level shifted L plane of **Fig. 8** (a) justifying the division into non-uniform blocks. However, there are still some discrepancies as red and pink colored stripes are getting merged with gray colored boundary region in single 32×32 blocks. Similarly, yellow, green and aqua colors are also getting merged. This happens as luminance values of red, pink and gray colors and that of yellow, green and aqua colors are similar as seen from **Fig. 8** (c). This is taken care of by implementation of this algorithm on subsampled chrominance planes. Result of division of chrominance *b** plane into non-uniform blocks is shown in **Fig. 8** (d). In **Fig. 8** (d), all the color components are getting divided faithfully. This also justifies the need of implementation of the splitting and merging algorithm on the three image planes separately. As compared to the hierarchical variable block size algorithm of H.264 or HEVC, our algorithm is more compatible to the image structure as both size and position of the image blocks are variable.


(a) Original Image


(b) Division of *L* plane into non-uniform blocks (color image shown for illustrative purpose only)


(c) Grey scale image of *L* plane divided into non-uniform blocks

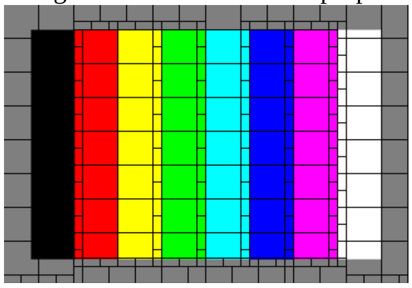

(d) Chrominance *b** plane divided into non-uniform blocks

Fig. 8. Process of histogram based merging on color bars image

The *M*×*N* image sub-blocks thus formed, are transformed into *M*×*N* spectral domain coefficients using two-dimensional Discrete Cosine Transform (2D-DCT).

3.4 CSF based Quantization

Since non-uniform block sizes are used, we need quantization matrices of sizes suitable for the 16 different block sizes. CSF based quantization method proposed in [36] is modified and used for the generation of quantization matrices. Though [36] describes quantization matrix generation for 8×8 block size only, this method is heuristically extended for the generation of quantization matrices for all other non-uniform block sizes. As CIE *La*b** space is used in the proposed scheme, a linear CSF is used for the generation of quantization matrices. As the range of intensity values is different for *L* plane and *a** and *b** planes, different set of quantization matrices are generated for *L* plane and *a** and *b** planes. *CSF* for quantization matrix generation as given in [36] is

$$CSF(f) = 100\sqrt{f}\,\exp(-0.13f) \tag{8}$$

where *f* is spatial frequency in cycles/degrees, given by

$$f(u,v) = 30\frac{\sqrt{u^2+v^2}}{8\times\Lambda} \tag{9}$$

where *u*,*v* are co-ordinates of the DCT block. Pixel size Λ is assumed to be 1.5 min/pixel [36]. Above equation gives frequency matrix for 8×8 block size only. Hence, the factor of 8 in the denominator of this equation. This equation can be extended to find frequency for any *M*×*N* size block using (10);

$$f(u,v) = 30\frac{\sqrt{u^2+v^2}}{Nn\times\Lambda} \tag{10}$$

where *Nn* is given by (11) as follows

$$Nn = \sqrt{M\times N} \tag{11}$$

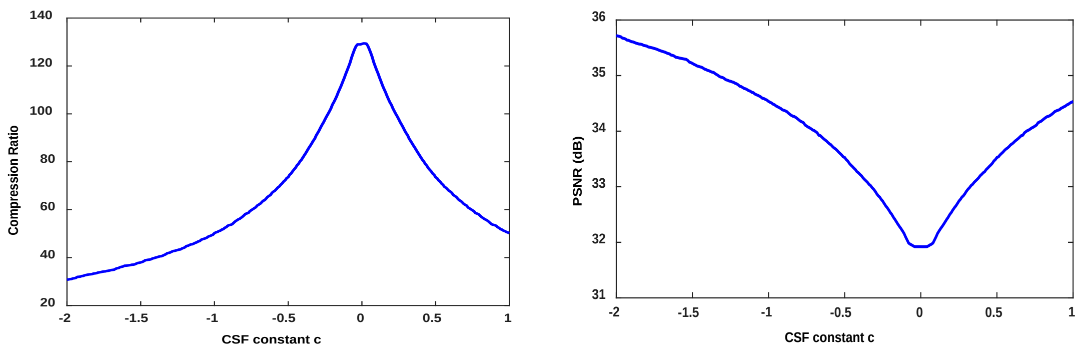
Since we are using CIE *La*b** color space for compression, a linear CSF given by (12) below is used in this paper for quantization matrix generation.

$$CSF(f) = c(f - f_{\max}) \tag{12}$$

where *f* is given by (10) and f_{max} is the maximum frequency in an *M*×*N* image sub-block and is calculated using the formula shown below

$$f_{\max} = 30\frac{\sqrt{M^2+N^2}}{Nn\times\Lambda} \tag{13}$$

In (12), *c* is a constant which governs the performance and quality of reconstructed image in terms of CR and PSNR as shown in **Fig. 9**.



(a) CSF constant c vs. CR (b) CSF constant c vs. PSNR

Fig. 9. CSF constant *c* versus CR, PSNR graphs for Lena image

As seen from **Fig. 9**, CR is highest for *c* values around zero, though PSNR is lowest for these values. An appropriate value of *c* can be chosen so that the tradeoff between CR and

PSNR is optimum. Linear CSF given in (12) gives good results as we are using *CIE La*b** space which is linear and perceptually uniform. This is evident from the results shown in **Table 1**.

Table 1 shows the CR and bpp (bits per pixel) achieved by linear CSF as well as non-linear CSF for fixed PSNR values. It can be clearly seen that for the same PSNR, CR achieved using the linear CSF proposed by us is better than the CR achieved by non-linear CSF proposed in [36]. Though the gain in CR is marginal in case of some images like Baboon, Yacht, Barbara, etc., compression performance of linear CSF based quantization is never poorer than that of non-linear CSF based quantization.

Table 1. Effect of linear and non-linear CSF based quantization on compression performance

Image Name	PSNR (dB)	Non-linear CSF [40]		Linear CSF	
		CR	bpp	CR	bpp
Baboon	30.2	13.95	0.57	14.41	0.55
Goldhill	34.52	34.21	0.23	36.72	0.22
Jupiter-moon	35.07	47.32	0.17	54.23	0.15
Lena	35.13	39.46	0.20	42.12	0.19
Woman-baby	41.4	107.98	0.07	119.63	0.07
Yacht	35.27	26.49	0.30	27.53	0.29
Barbara	34.42	28.84	0.28	30.42	0.26
Average	**35.14**	**42.61**	**0.19**	**46.44**	**0.17**

The role of orientation tuning is also considered by using the Orientation Tuning Function (OTF) as given by (14);

$$OTF(u,v) = \begin{cases} \exp\left(-9.5\left(\frac{u}{v}\right)^2\right) & \text{for } u < v \\ \exp\left(-9.5\left(\frac{v}{u}\right)^2\right) & \text{otherwise} \end{cases} \tag{14}$$

If the value of *OTF* drops below 0.5, *OTF* value is considered to be 0.5.

Using (9) to (14), the thresholds for DCT basis functions in an M×N matrix are given as;

$$T(u,v) = \begin{cases} \dfrac{1}{\text{Norm}(u,v)\times CSF(f)} & \text{for } u = 0 \text{ or } v = 0 \\ \dfrac{1}{\text{Norm}(u,v)\times CSF(f)\times OTF(u,v)} & \text{for } u > 0 \text{ and } v > 0 \end{cases} \tag{15}$$

Norm(*u*,*v*) are the normalization functions used in 2D-DCT. One thing to note from the above equation is that value of the threshold for $u = 0$ and $v = 0$ is ∞ as value of *CSF* is zero for $u = 0$ and $v = 0$. Thus, quantization values for the DC coefficient are decided (or calculated) separately.

Finally, quantization matrix is given by

$$Quant(u,v) = \min(T(u,v)\times range, coeff\_\max(u,v)) \tag{16}$$

where *coeff_max* is the matrix containing maximum values that DCT coefficients can take for a given range of spatial frequencies. Stimulus required to generate the matrix [36] is;

$$x_{\max}(x,y)=\frac{range}{2}\times\left(1+\operatorname{sgn}\left(\cos\left(\frac{(2x+1)u\pi}{2M}\right)\cos\left(\frac{(2y+1)v\pi}{2N}\right)\right)\right) \tag{17}$$

where sgn = +1 or -1 depending on the argument.

Above equation can be modified if the range of intensity values is from –X to +X, as in the case of *a** and *b** planes. The modified equation is

$$x_{\max}(x,y)=\frac{range}{2}\times\operatorname{sgn}\left(\cos\left(\frac{(2x+1)u\pi}{2M}\right)\cos\left(\frac{(2y+1)v\pi}{2N}\right)\right) \tag{18}$$

Quantization value for the DC coefficient is chosen depending upon the maximum value attained by the DC coefficient of the DCT sub-block that is calculated using (17) and (18). As quoted in [45], a human being can distinguish at the most 100 distinct luminance values (L_{max} = 100). Hence, we quantize the maximum DC coefficient value for *L* plane of each block size in 100 levels. For example, for an 8×8 *L* block, maximum DC coefficient value is 800. If this value is quantized in 100 levels, we end up at a DC quantization value of 8. For 8×8 chrominance *a** and *b** blocks also, the DC coefficient quantization value is taken as 8. However, the range of *a** and *b** planes is -100 to +100 which is double than that of *L* plane. Hence, the same DC coefficient quantization value (i.e. 8 in case of 8×8 blocks) results in 200 quantization levels for *a** and *b** planes. Similarly, DC quantization steps for some of the 16 block sizes are calculated and presented in **Table 2**.

Table 2. DC coefficient quantization levels for CSF based quantization

Block Size	DC coefficient quantization levels
8×8	8
8×16	11
16×8	11
16×16	16
16×24	20
24×16	20
24×24	24
24×32	28
32×24	28
32×32	32

Thus, 32 quantization matrices are generated; 16 for *L* plane and 16 for *a** and *b** planes, using the approach discussed above. However, the quantization matrices for *a** and *b** planes are double of the quantization matrices for *L* plane. Therefore, only 16 quantization matrices for *L* plane need to be stored.

3.5 Entropy Coding

Non-zero discrete cosine transformed quantized coefficients of each image sub-block are zigzag ordered before entropy coding. In this paper, we have used modified binary arithmetic QM coding as entropy coding technique with our proposed algorithm. Modification of QM coding used with JPEG to encode variable block size DCT coefficient matrix is straightforward. Exponential Golomb coding is used for encoding the non-uniform block size index array, *BIdx*.

The decoding process is essentially opposite and exact reverse of the complete encoding process shown in **Fig. 3**.

4. Experimental Results and Discussion

Performance of the proposed algorithm is experimentally evaluated using 25 standard test images of different types and sizes available online at the referenced websites [46, 47]. Peak Signal to Noise Ratio (PSNR) is used to evaluate the reconstructed image quality whereas compression performance is tested using bits per pixel (bpp) and CR. If length of the compressed bitstream is denoted by *no_bits*, bits per pixel denoted by *r* is given as

$$r = \frac{no\_bits}{p \times q \times 3} bpp \tag{19}$$

where *p* and *q* are the number of rows and columns of the color image, respectively. Value of CR can be obtained from *r* using the relation in (20);

$$CR = \frac{8}{r} \tag{20}$$

All experiments are done on a desktop computer with an Intel Core i7-4770 processor running at 3.40 GHz with 32 GB DDR4 RAM. MATLAB programming environment is used for Windows 8 OS.

As discussed earlier, 16 quantization matrices are generated and can be stored at the encoder and the decoder for the CSF based quantization approach. Alternatively, with some increase in the computational complexity, quantization matrices can be generated using the CSF based quantization approach or by interpolation from a single 8×8 quantization matrix for a given value of CSF constant *c*. Each value of CSF constant *c* results in a unique set of quantization matrices. This fact is used to control the compression quality i.e. the achieved bpp, CR and PSNR values in the proposed work. A lower absolute value of *c* will result in higher compression (CR) and lower reconstructed image quality (PSNR) whereas a higher absolute value of *c* yields lower CR and higher PSNR. A value of *c* = -0.5 is chosen arbitrarily for comparison and benchmarking so that the JPEG, JPEG-XR and the proposed algorithm yield exactly same PSNR on the set of test images. However, the final choice of c depends on the end-user or the application for which the proposed compression scheme is being used. In JPEG and JPEG-XR, compression quality is controlled by quantization factor or quality factor (QF). QF is a constant by which the standard JPEG quantization matrices are multiplied in order to achieve required CR and PSNR. QF can be varied from 1 to 100 in JPEG whereas it can take values from 1 to 255 in JPEG-XR. Higher value of QF results in high compression and lower reconstructed image quality and vice versa in JPEG [42]. However, higher value of QF yields higher reconstructed image quality and hence, less compression in JPEG-XR.

Representative original images from the set of 25 standard test images are shown in **Fig. 10** (a) – (e). Images reconstructed using the proposed technique along with bpp and PSNR are shown in **Fig. 11** (a) – (e). It is clear that images with high neighboring interpixel intensity variations like Baboon yield poor bpp and PSNR. On the other hand, images like Woman-baby, which contain low neighboring interpixel intensity variations, yield high bpp and PSNR.

(a)Lena (b)Peppers (c)Baboon (d)Yacht (e)Woman-baby

Fig. 10. Original images from the set of 25 standard test images

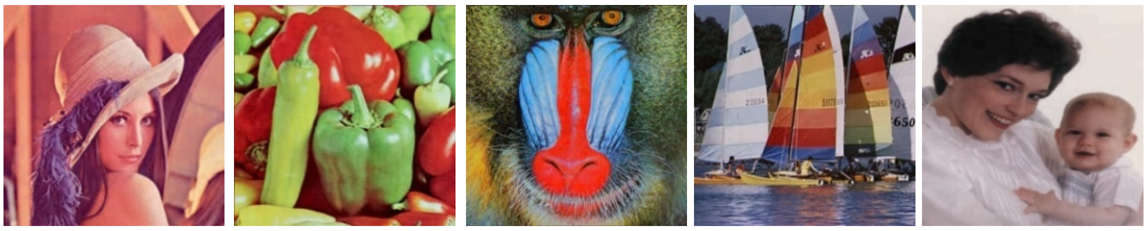

(a)Lena (*r*=0.09bpp, PSNR=33.6dB) (b)Peppers (*r*=0.12bpp, PSNR=32.6dB) (c)Baboon (*r*=0.22bpp, PSNR=29.6dB) (d)Yacht (*r*=0.14bpp, PSNR=32.9dB) (e)Woman-baby (*r*=0.05bpp, PSNR=39.3dB)

Fig. 11. Images compressed and reconstructed using the proposed algorithm

Results obtained using the proposed scheme are compared with JPEG using 25 standard test images and also with other related published research works [48-50].

4.1 Comparison with JPEG and JPEG-XR using standard test images

We compare the results obtained using the proposed scheme with JPEG and JPEG-XR [51]. For comparison with JPEG, we use MATLAB's built-in implementation of the JPEG encoder. Different parameters need to be set for the JPEG encoder in order to obtain optimum performance on different images in different situations. For JPEG, compression mode is set to 'lossy' and bit depth is set at its default value of 8. For comparison with JPEG-XR, we use an open source implementation by [52]. Since JPEG and the proposed scheme use non-overlapping blocks, we turn overlapping off in JPEG-XR for fair comparison of results. For the same reason, chrominance sub-sampling is also set to 4:2:0 in JPEG-XR. Image quality or quantization parameter is varied, in both JPEG and JPEG-XR, to get PSNR equal to that achieved by the proposed scheme for different images.

The benchmarking of the 25 standard test images along with the average results is shown in **Table 3**. We vary the quality setting parameter from 11 – 54 for JPEG and from 1 – 85 for JPEG-XR so as to get same PSNR values for the two algorithms.

Table 3. Comparison with image compression standards using 25 standard test images

Image Name*	PSNR (dB)	JPEG		JPEG-XR		Proposed Scheme	
		r (bpp)	CR	*r* (bpp)	CR	*r* (bpp)	CR
Baboon	29.6	0.26	30.24	0.26	**30.35**	0.22	**36.6**
Boats	33.9	0.13	59.9	0.1	77.9	0.11	69.7
Cable-car	32.9	0.15	51.9	0.12	65.5	0.13	59.5
Caster-stand	33.3	0.13	60.6	0.11	71.7	0.12	64.8
Color-bars	43.6	0.13	63.7	0.11	**71.2**	0.03	**310.4**
RGB full color cube	37.9	0.09	81.1	0.06	**144**	0.04	**186.9**
Cornfield	32.3	0.17	46.4	0.17	**48.1**	0.16	**48.6**
Chalk	34.9	0.12	68.2	0.06	137.8	0.08	104.5
Flower	34.4	0.12	69.3	0.09	90.1	0.10	76.2
Flowers	30.96	0.25	31.8	0.13	59.4	0.19	42.9
Goldhill	33.0	0.13	60.6	0.11	76	0.11	75.6
Jupiter-moon close-up	33.4	0.13	63.9	0.11	75.9	0.11	73.9
Jupiter-moon	34.5	0.11	75.9	0.09	**85.4**	0.07	**115.6**
Lena	33.6	0.12	64.02	0.09	85.4	0.09	81.3
Monarch	33.7	0.13	60.4	0.11	72.1	0.12	67.1
Peppers	32.6	0.12	65.4	0.09	85.4	0.12	65.6

Football	31.8	0.18	43.8	0.17	48.1	0.17	46.8
Strawberries-coffee	34.6	0.12	66.03	0.07	120.4	0.08	95.9
Stream	34.5	0.09	81.8	0.07	106.7	0.09	91.6
Woman-baby	39.3	0.08	99.5	0.04	190.4	0.05	168.6
Yacht	32.9	0.15	53.5	0.12	65.5	0.14	57.9
Airplane	33.6	0.13	62.9	0.09	85.4	0.11	73.6
Sailboat	31.7	0.17	46.9	0.14	59.2	0.15	53.7
Tiffany	33.4	0.11	71.6	0.06	128.2	0.09	86.5
Barbara	32.4	0.18	45.4	0.16	**50.7**	0.14	**57.3**
Average of 25 std. test images	**33.95**	**0.14**	**60.99**	**0.11**	**85.23**	**0.11**	**88.44**

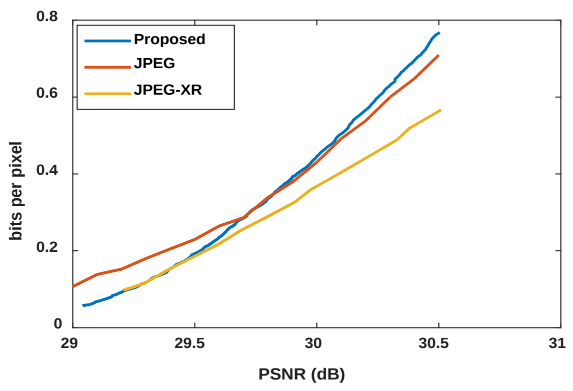
*The images are taken from Image processing place [46] and Classic Image processing library [47].

As seen from **Table 3**, proposed scheme surpasses the bpp and CR achieved by JPEG for all the standard test images. It can be deduced that the proposed scheme outperforms JPEG by a large margin in case of images having large smooth regions i.e. low variance images such as Color bars, Chalk, Jupiter-moon, etc. Due to the use of variable block sizes, low variance images get divided into larger blocks instead of the conventional 8 × 8 blocks in JPEG. The proposed scheme performs only marginally better than JPEG for images having high variance (i.e. more texture) such as Peppers, Football, Cornfield, etc. This is because high variance images do not benefit from the use of larger block sizes as proposed in the variable block size algorithm. Due to absence of smooth (low variance) regions, these images get divided into smaller block sizes only. Proposed scheme achieves, on an average, 45.01% more CR than JPEG.

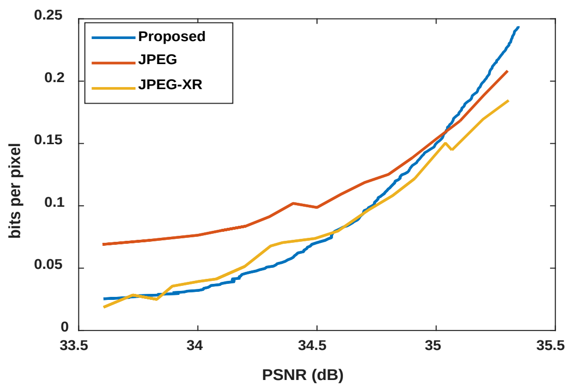
Though the proposed scheme achieves 3.63% better CR than JPEG-XR, it performs better than JPEG-XR only for a few images such as Baboon, Color-bars, RGB full color cube, Cornfield, Jupiter-moon and Barbara. In case of images such as Football, Goldhill, Jupiter-moon close-up, etc. JPEG-XR performs marginally better than the proposed scheme. JPEG-XR outperforms the proposed scheme for rest of the images as evident from **Table 3**.

Performance of the proposed scheme is also compared with JPEG and JPEG-XR using rate-distortion curves shown in **Fig. 12**. Bits per pixel versus PSNR graphs are shown for eight standard test images. Graphs are plotted by keeping the PSNR achieved using JPEG and JPEG-XR same as that achieved by the proposed algorithm. Thus, the algorithm having lower bpp value at the same PSNR can be considered to have better compression performance. As seen from **Fig. 12**, performance of the proposed scheme is better than JPEG for all the images at lower bitrates (i.e. lower bpp values). For the Woman-baby image, performance of the proposed scheme is better than JPEG at all the bitrates. The proposed scheme performs only marginally better than JPEG-XR at lower bit rates for images such as Baboon and Jupiter-moon. Performance of JPEG-XR is marginally better than the proposed scheme for images such as Woman-baby, Cornfield and Yacht. In general, it can be said that performance of JPEG-XR is comparable to the proposed scheme at lower bit rates.

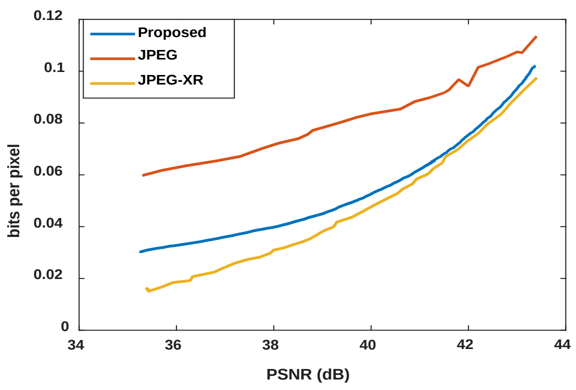
It is well-known that JPEG2000 achieves approximately 50% more CR than JPEG [53] while being 3 to 5 times more complex [54, 55]. Therefore, it is expected that the proposed scheme may not out-perform JPEG2000. Furthermore, JPEG2000 is a wavelet based compression scheme unlike the proposed scheme and JPEG-XR, which uses a transform similar to DCT and a block-based coding structure.



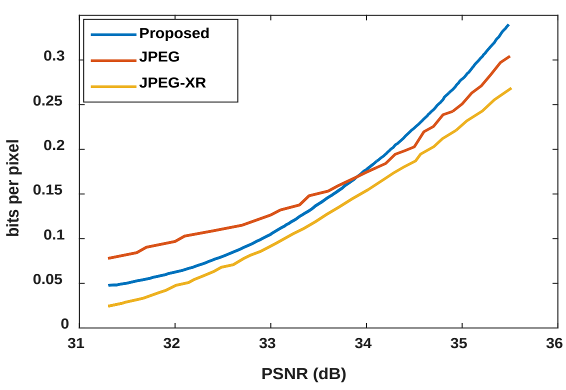
(a) Baboon



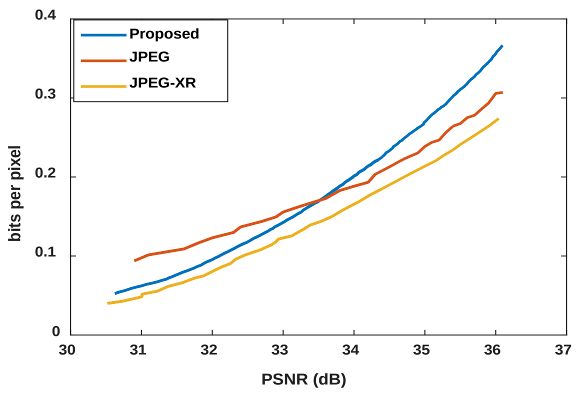
(b) Jupiter-moon



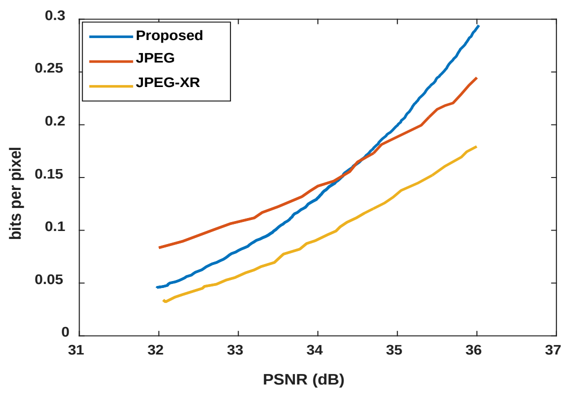
(c) Woman-baby



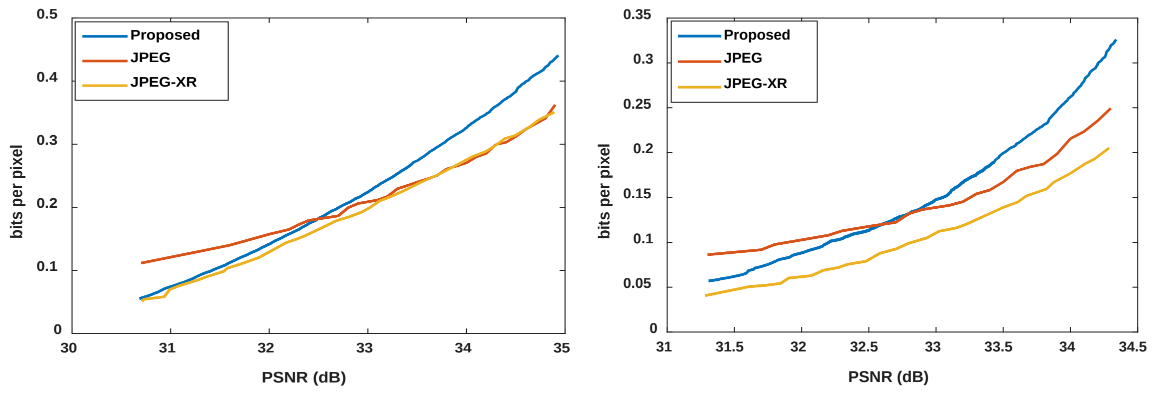
(d) Goldhill



(e) Yacht



(f) Airplane



(g) Cornfield (h) Peppers

Fig. 12. Bits per pixel versus PSNR plots for standard test images

Due to the use of variable non-uniform blocks, the proposed scheme requires more computations than JPEG. The histogram formation step in the proposed algorithm is the most computationally intensive step. An image plane of size P×Q is first divided into 8×8 non-overlapping blocks and then a histogram is formed using thresholds Th_{mn} and Th_{var}. In the worst case scenario, if none of the histogram rows get merged with another; formation of histogram itself may require $\left(\frac{P\times Q}{32}\right)\times\left(\frac{P\times Q}{32}+1\right)$comparisons. However, this scenario is not expected to occur as we are using image dependent thresholds Th_{mn} and Th_{var}. After experimentations, it has been observed that the number of histogram rows varies between 2 and 1167 for the standard test images and synthetic images used for benchmarking. Considering the number of histogram rows as 1167, the proposed algorithm requires approximately 1.5 times more computations as compared to JPEG.

4.2 Comparison with contemporary techniques

Proposed scheme is also compared with other contemporary techniques such as those proposed in [48-50] respectively, as shown in **Table 4**. [49] has achieved compression using machine learning techniques. Image is compressed using selected colors during encoding and missing colors are predicted during decoding. Better compression quality is achieved by minimizing the prediction error during decoding. Data-hiding and compression tasks are incorporated seamlessly by [48]. Blocks are embedded with secret data and based on the current bit being embedded, the block is encoded using either SMVQ or image inpainting. [50] incorporates HVS based models for thresholding and quantization of coefficients in a wavelet based compression scheme.

For fair comparison of results, PSNR of the proposed scheme is kept equal to the highest PSNR achieved by one of the above-mentioned schemes. However, in case of Baboon image, PSNR of 25.2 dB cannot be achieved using the proposed method. Hence, minimum PSNR achievable for Baboon i.e. 29.05 dB, is used. Similarly for images; Peppers, Airplane, Tiffany and Sailboat, minimum PSNR achievable by the proposed scheme is used. As seen from **Table 4**, our method performs better than almost all the techniques proposed in [48-50]. Note that the bpp values quoted in [50] are converted to CR using the relation in (26). For comparison with results in [50], two images, namely, Parrots and Statue, from the LIVE database [56] are used.

Table 4. Comparison with other proposed compression schemes

Image Name	Proposed Scheme		Zhang et al.'s scheme		Qin et al.'s scheme		Sreelekha et al.'s scheme			
							PLC-1		PLC-2	
	CR	PSNR (dB)	CR	PSNR (dB)	CR	PSNR (dB)	CR	PSNR (dB)	CR	PSNR (dB)
Lena	105.8	33.0	23.7	33.0	20.66	29.85	-	-	-	-
Peppers	137.2	31.37	22.6	31.0	21.23	30.35	-	-	-	-
Baboon	136.9	29.05	11.4	25.2	-	-	-	-	-	-
Airplane	174.7	31.98	-	-	21.37	29.31	-	-	-	-
Tiffany	157.94	32.5	-	-	21.74	30.54	-	-	-	-
Sailboat	141.8	30.76	-	-	20.12	28.42	-	-	-	-
Monarch	63.81	33.86	-	-	-	-	40	33.23	40	33.86
Parrots	77.19	36.51					57.14	33.75	57.14	36.51
Statue	89.62	33.67					36.36	32.80	36.36	33.66

5. Conclusion

In this paper, images are compressed using a linear and perceptually uniform CIE *La*b** color space in the JPEG image compression framework instead of the conventional *YCbCr* color space. A linear CSF suitable for the linear and perceptually uniform CIE *La*b** color space is proposed. Image dependent variable size sub-blocks generated using the proposed novel histogram based merging technique are also used instead of the conventional 8×8 image sub-blocks in JPEG. In other words, images are compressed in the proposed scheme using the adaptively separated non-uniform blocks in the CIE *La*b** color space using the JPEG image compression framework but our own quantization matrices generated using the proposed linear CSF function.

The proposed scheme achieves, on an average, 45.01% more CR than JPEG for the same reconstructed image quality in terms of PSNR. It outperforms JPEG in terms of reconstructed image quality at lower bitrates. The proposed scheme also yields reconstructed image quality better than JPEG at all the bitrates for simple, low variance images. The proposed scheme achieves on an average, 3.63% better CR than JPEG-XR. Performance of JPEG-XR is comparable to the proposed scheme at lower bit rates only. It is also demonstrated that the use of linear CSF for the perceptually uniform and linear CIE *La*b** color space yields on an average 8% more compression than the non-linear CSF used with *YCbCr* color space.

References

[1] J. M. Pascual, H. M. Mora, A. F. Guilló and J. A. López, "Adjustable compression method for still JPEG images," *Signal Processing: Image Communication*, vol. 32, pp. 16-32, March 2015. Article (CrossRef Link)

[2] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consumer Electronics*, vol. 38, no. 1, pp. xviii—xxxiv, February 1992. Article (CrossRef Link)

[3] J. Mitchell, "Digital compression and coding of continuous-tone still images: Requirements and guidelines," *ITU-T Recommendation T.* 81, 1992.

[4] T. Richter, "JPEG on STEROIDS: Common optimization techniques for JPEG image compression," in *Proc. of IEEE Intl. Conf. Image Processing (ICIP)*, pp. 61 – 65, September 25 – 28, 2016. Article (CrossRef Link)

[5] J.-J. Ding, Y.-W. Huang, P.-Y. Lin, S.-C. Pei, H.-H. Chen and Y.-H. Wang, "Two-dimensional orthogonal DCT expansion in trapezoid and triangular blocks and modified JPEG image compression," *IEEE Trans. Image Process*, vol. 22, no. 9, pp. 3664-3675, September 2013.

Article (CrossRef Link)
[6] S. Makrogiannis, P. Schelkens, S. Folopoulos and J. Cornelis, "Region-oriented compression of color images using fuzzy inference and shape adaptive DCT," in *Proc. of IEEE Intl. Conf. Image Processing (ICIP)*, pp. 478 – 481, October 7 – 10, 2001. Article (CrossRef Link)
[7] J. Wu, Y. Xing, G. Shi and L. Jiao, "Image Compression with downsampled and overlapped transform at low bit rates," in *Proc. of IEEE Intl. Conf. Image Processing (ICIP)*, pp. 29 – 32, November 7 – 10, 2009. Article (CrossRef Link)
[8] G. Fracastoro, F. Verdoja, M. Grangetto and E. Magli, "Superpixel-driven graph transform for image compression," in *Proc. of IEEE Intl. Conf. Image Process. (ICIP)*, pp. 2631 – 2635, September 27 – 30, 2015. Article (CrossRef Link)
[9] M. Gordan, S. Meza, M. Cislariu, B. Orza, A. Vlaicu, D. Capatina and I. Stoian, "A fuzzy logic approach for the fast approximate computation of image transforms from block JPEG DCT coefficients," in *Proc. of IEEE Intl. Conf. Automation, Quality and Testing, Robotics (AQTR)*, pp. 1-6, May 19 – 21, 2016. Article (CrossRef Link)
[10] S. K. Pattanaik and K. K. Mahapatra, "DHT based JPEG image compression using a novel energy quantization method," in *Proc. of IEEE Intl. Conf. Industrial Technol. (ICIT)*, pp. 2827-2832, December 15 – 17, 2006. Article (CrossRef Link)
[11] Z. Wang, S. Simon, Y. Baroud and S. M. Najmabadi, "Visually lossless image compression extension for JPEG based on just-noticeable distortion evaluation," in *Proc. of Intl. Conf. Systems, Signals Image Process. (IWSSIP)*, pp. 237-240, September 10 – 12, 2015. Article (CrossRef Link)
[12] Y. Jiang and M. S. Pattichis, "JPEG image compression using quantization table optimization based on perceptual image quality assessment," in *Proc. of 45th Asilomar Conf. Signals, Systems Computers (ASILOMAR)*, pp. 225-229, November 6 – 9, 2011. Article (CrossRef Link)
[13] X. Zhang, S. Wang, K. Gu, W. Lin, S. Ma and W. Gao, "Just-Noticeable Difference-Based Perceptual Optimization for JPEG Compression," *IEEE Signal Process. Lett.*, vol. 24, no. 1, pp. 96-100, January 2017. Article (CrossRef Link)
[14] A. J. Ahumada and H. A. Peterson, "Luminance-model-based DCT quantization for color image compression," in *Proc. of SPIE, Human Vision, Vis. Process. Digital Display III*, pp. 365 – 374, August 1992. Article (CrossRef Link)
[15] A. B. Watson, "DCTune: A technique for visual optimization of DCT quantization matrices for individual images," in *Proc. of 24th Soc. Info., Display Digital Technol. Papers*, pp. 946 – 949, 1993. Article (CrossRef Link)
[16] X. Zhang, W. Lin and P. Xue, "Improved estimation for just-noticeable visual distortion," *Signal Process.*, vol. 85, no. 4, pp. 795 – 808, January 2005. Article (CrossRef Link)
[17] Z. Wei and K. Ngan, "Spatio-temporal just noticeable distortion profile for grey scale image/video in DCT domain," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 3, pp. 337 – 346, February 2009. Article (CrossRef Link)
[18] L. Ma, K. Ngan, F. Zhang and S. Li, "Adaptive block-size transform based just-noticeable difference model for images/videos," *Signal Process.:Image Comm.*, vol. 26, no. 3, pp. 162 – 174, March 2011. Article (CrossRef Link)
[19] I. Hontsch and L. Karam, "Adaptive image coding with perceptual distortion control," *IEEE Trans. Image Process.*, vol. 11, no. 3, pp. 213 – 222, August 2002. Article (CrossRef Link)
[20] S.-H. Bae and M. Kim, "A novel DCT-based JND model for luminance adaptation effect in DCT frequency," *IEEE Signal Process. Lett.*, vol. 20, no. 9, pp. 893 – 896, September 2013. Article (CrossRef Link)
[21] S.-H. Bae and M. Kim, "A novel generalized DCT-based JND profile based on an elaborate CM-JND model for variable block-sized transforms in monochrome images," *IEEE Trans. Image Process.*, vol. 23, no. 8, pp. 3227 – 3240, August 2014. Article (CrossRef Link)
[22] N. A. Abu, F. Ernawan, N. Suryana, "A Generic Psychovisual Error Threshold for the Quantization Table Generation on JPEG Image Compression," in *Proc. of 9th Intl. Coll. Signal*

Process. Appl., pp. 39 – 43, March 8 – 10, 2013. Article (CrossRef Link)

[23] P. A. M. Oliveira, R. S. Oliveira, R. J. Cintra, F. M. Bayer, A. Madanayake, "JPEG quantisation requires bit-shifts only," *Electronics Lett.*, vol. 53, no. 9, pp. 588-590, April 2017. Article (CrossRef Link)

[24] H.-H. Chen, Y.-W. Huang and J.-J. Ding, "Local prediction based adaptive scanning for JPEG and H. 264/AVC intra coding," in *Proc. of IEEE Intl. Conf. Image Process. (ICIP)*, pp. 1636-1640, September 15 – 18, 2013. Article (CrossRef Link)

[25] M. B. Akhtar, A. M. Qureshi, "Optimized run length coding for jpeg image compression used in space research program of IST," in *Proc. of Intl. Conf. Computer Networks Info. Technol. (ICCNIT)*, pp. 81-85, July 11 – 13, 2011. Article (CrossRef Link)

[26] G. Lakhani, "Modifying JPEG binary arithmetic codec for exploiting inter/intra-block and DCT coefficient sign redundancies," *IEEE Trans. Image Process.*, vol. 22, no. 4, pp. 1326-1339, April 2013. Article (CrossRef Link)

[27] J.-J. Ding, H.-H. Chen and W.-Y. Wei, "Adaptive Golomb code for joint geometrically distributed data and its application in image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 4, pp. 661-670, April 2013. Article (CrossRef Link)

[28] V. Hosu, F. Hahn, O. Wiedemann, S.-H. Jung and D. Saupe, "Saliency driven image coding improves overall percieved JPEG quality," in *Proc. IEEE Pic. Coding Symp. (PCS)*, December 4 – 7, 2016. Article (CrossRef Link)

[29] L. Jin, K. Egiazarian and C.-C. Jay Kuo, "JPEG based perceptual image coding with block based image quality metric," in *Proc. of IEEE Intl. Conf. Image Process. (ICIP)*, pp. 1053-1056, Sept. 30 – Oct. 3, 2012. Article (CrossRef Link)

[30] T. Nguyen, P. Helle, B. Winken, B. Bross, D. Marpe, H. Schwarz and T. Weigand, "Transform coding techniques in HEVC," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 6, pp. 978 – 989, December 2013. Article (CrossRef Link)

[31] M. Wein, "Variable block-size transforms for H.264," *IEEE Trans. Circuits and Syst. Video Technol.*, vol. 13, no. 7, pp. 604 – 613, July 2003. Article (CrossRef Link)

[32] T. Huang, S. Dong and Y. Tian, "Representing visual objects in HEVC coding loop," *IEEE J. Emerging Sel. Topics Circuits Syst.*, vol. 4, no. 1, pp. 5 -16, March 2014. Article (CrossRef Link)

[33] I. Rhee, G. R. Martin, S. Muthukrishnan and R. A. Packwood, "Quadtree-srtuctured variable-size block-matching motion estimation with minimal error," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 1, pp. 42 – 50, February 2000. Article (CrossRef Link)

[34] M. Q. Shaw, J. P. Allebach and E. J. Delp, "Color difference weighted adaptive residual preprocessing using perceptual modeling for video compression," *Signal Process.: Image Commun.*, vol. 39, pp. 355-368, November 2015. Article (CrossRef Link)

[35] C.-H. Chou and K.-C. Liu, "Colour image compression based on the measure of just noticeable colour difference," *IET Image Process.*, vol. 2, no. 6, pp. 304-322, December 2008. Article (CrossRef Link)

[36] S. A. Klein, A. D. Silverstein and T. Carney, "Relevance of human vision to JPEG-DCT compression," in *Proc. of SPIE-The Intl. Soc. Optical Engg.*, vol. 1666, pp. 200-216, August 1992. Article (CrossRef Link)

[37] E. Dumic, M. Mustra, S. Grgic and G. Gvozden, "Image quality of 4:2:2 and 4:2:0 chroma subsampling formats," in *Proc. of Intl. Symp. ELMAR*, pp. 19-24, September 28 – 30, 2009.

[38] P. Zeng and Z. Chen, "Perceptual quality measure using JND model of the human visual system," in *Proc. of Intl. Conf. Electric Info. Control Engg. (ICEICE)*, pp. 2454-2457, April 15 – 17, 2011. Article (CrossRef Link)

[39] A. Ford and A. Roberts, *Colour space conversions*, 1998. URL: Article (CrossRef Link) [Last accessed: 13 December 2007] 2011.

[40] M. D. Fairchild, *Color appearance models*, 2^{nd} Edition, John Wiley & Sons, New York, 2013. Article (CrossRef Link)

[41] G. Sharma and R. Bala, "Digital color imaging handbook," *CRC press*, New York, 2002. Article (CrossRef Link)

[42] R. C. Gonzalez and R. E. Woods, "Digital image processing," *3rd Edition*, *Prentice Hall*, 2002.

[43] W. B. Pennebaker and J. L. Mitchell, "JPEG: Still image data compression standard," *Springer Science & Business Media*, Massachusetts, 1992.

[44] G. Lakhani, "Improving DC coding models of JPEG arithmetic coder," *IEEE Signal Process. Lett.*, vol. 11, no. 5, pp. 505-508, May 2004. Article (CrossRef Link)

[45] C. Poynton, "Color in digital cinema," *C. Swartz (ed.) Understanding Digital Cinema: A Professional Handbook*, Elsevier, pp. 57-82, 2005. Article (CrossRef Link)

[46] *Image Databases - Image Processing Place.* Article(CrossRefLink)

[47] *Classic Image Processing Library*. Article (CrossRef Link).

[48] C. Qin, C.-C. Chang and Y.-P. Chiu, "A novel joint data-hiding and compression scheme based on SMVQ and image inpainting," *IEEE Trans. Image Process.*, vol. 23, no. 3, pp. 969-978, March 2014. Article (CrossRef Link)

[49] C. Zhang and X. He, "Image compression by learning to minimize the total error," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 4, pp. 565-576, April 2013. Article (CrossRef Link)

[50] G. Sreelekha and P. S. Sathidevi, "An HVS based adaptive quantization scheme for the compression of color images," *Digital Signal Process.*, vol. 20, no. 4, pp. 1129-1149, July 2010. Article (CrossRef Link)

[51] F. Dufaux, G. J. Sullivan and T. Ebrahimi, "The JPEG XR image coding standard," *IEEE Signal Process. Mag.*, vol. 26, no. 6, November 2009. Article (CrossRef Link)

[52] Article (CrossRef Link)

[53] A. Skodras, C. Christopoulos and T. Ebrahimi, "The JPEG 2000 still image compression standard," *IEEE Signal Process. Mag.*, vol. 18, no. 5, pp. 36 - 58, September 2001. Article (CrossRef Link)

[54] M. W. Marcellin, M. J. Gormish, A. Bilgin and M. P. Boliek, "An overview of JPEG 2000," In *Proc. of IEEE Data Compression Conf. (DCC)*, pp. 523 – 541, March 28 – 30, 2000. Article (CrossRef Link)

[55] D. Santa-Cruz and T. Ebrahimi, "A study of JPEG 2000 still image coding versus other standards," in *Proc. of 10th European Signal Process. Conf.*, pp. 1 – 4, September 4 – 8, 2000.

[56] H. R. Sheikh, M. F. Sabir and A. C. Bovik, "A statistical evaluation of recent full reference quality assessment algorithms," *IEEE Trans. Image Process.*, vol. 15, no. 11, pp. 3440 – 3451, November 2006. Article (CrossRef Link)

Samruddhi Y. Kahu received B.Eng. degree in Electronics and Communication Engineering from Nagpur University, Nagpur, India in 2011 and M.Tech. degree in Communication Systems Engineering from Visvsesvaraya National Institute of Technology, Nagpur, India in 2014 where she is currently pursuing her Ph.D. Her research interests include image and video compression and processing.

Kishor M. Bhurchandi received his B.Eng. and M.Eng. degrees in electronics engineering in 1990 and 1992. He further obtained his Ph.D. degree from Visvesvaraya Regional College of Engineering, Nagpur University, Nagpur, India in 2002, where he is currently working as a Professor. He is principal investigator of two funded major research projects in the field of signal processing and embedded systems. He has more than 45 publications to his credit. He has also worked on industrial problems on heavy and in motion weighing systems. He is the co-author of a popular book titled Advanced Microprocessors and Peripherals published by McGraw Hill, India. His research interests include color image processing and analysis, computer vision, digital signal processing and embedded systems.