JKAU: Comp. IT. Sci., **Vol. 6** No. 1-2, pp: 45 – 57 (1438 A.H./ 2017 A.D.)

Doi: 10.4197/Comp. 6-1.4

Color Image Compression Using Partial Pixels Elimination and Hybrid Pattern Recognition

Saleh A. Alshehri

Department of Commuter Science and Engineering, Jubail University College, Jubail Industrial City, Saudi Arabia
shehri@ucj.edu.sa

Abstract. The increased usage of media in our daily life has forced the need for efficient image storage and transmission. It has been shown previously that eliminating some image columns and rows of pixels can be recovered by the use of neural network. In this article, the method is further investigated for color images. Many of existing methods consume relatively large execution time for image compression. The computation time using this method is negligible compared to other method. Additionally, it takes less than half of the execution time than SPIHT to perform image compression at a compression quality greater than 0.75.

Keywords: Image compression, Pattern recognitio, Neural networks, Image pixels elimination, Color components replacement.

1. Introduction

Image compression plays an important role in many applications. For example, almost all social media services extensively deal with images and video clips storing and communicating. In such cases, the image size can affect the application's performance [1]. Since these social media mostly run on smartphones or tablets, the storage is also a concern. Also, smartphones are equipped with powerful cameras that can replace regular cameras. On the other hand, communication of images and video clips is constantly in increased demand [1]. Because of the demand for high image quality, image size is becoming larger with the help of advances in the technology. Image size and image processing algorithm computation performance directly affect the performance of real-time applications. This results in need for more efficient image compression methods.

Lossy and lossless image compressions are the two main categories. The original and the decompressed images are exactly the same in lossless compressions [2]. The compression is called lossy compression if the two images differ in any image pixels [2]. In certain applications where preserving the image quality is needed, such as medical and cosmological applications, lossless compression is a must. In other applications, lossy compression can safely be used. Personal device applications are examples of these applications.

Many color image compression methods have been proposed [1,3-6]. Image compression is based on two properties. The first property is that in many cases there are some parts of the image that do not contribute much to image appearance [7-9]. The second property is the image information redundancy [1,3]. These two image properties encourage many researchers to come up with many compression algorithms [1,4,6]. In particular, lossless compression is based either on statistical coding or run length encoding. In these two cases, repeated symbols are represented with a smaller number of codes [1,7]. In lossy compression, the unnecessary pixels are removed during compression stage and recovered during the decompression stage using various algorithms [1,7].

Pattern recognition methods can be used as lossy compression methods where the removed image pixels can be recovered intelligently. Many pattern recognition methods, such as artificial neural networks or simply (NN), Support Vector Machine (SVM), Self-Organizing Map (SOM), Principle Component Analysis (PCA) and others have been proposed [10-13]. Each of which has advantages and disadvantages. In many studies, removing some pixels is done based on finding some correlation between image color components [14-17]. This correlation can be measured in different color spaces such as RGB, YUV, and YCbCr [14]. The correlation between luminance and chrominance was investigated in [15]. Expressing one color component as polynomial approximation of the third component, which is called the dominant color, was presented [16].

In this research, we use two steps for image compression. First, image pixels are removed evenly from each color component. This will not generate blocking artifacts when pixels are recovered during decompression. The Hybrid pattern recognition (HPR) method is used to predict these removed pixels. Second, lower-level image depth binary matrices are completely removed from two color components. The third color component, together with a correction matrix factor, is used to recover the removed matrices. In this method, the computation time is negligible [18]. This is because the main task in the compression stage is just removing image pixels from predefined locations without the need to do any computation. Almost all other methods, including JPEG standards, consume considerable execution time [19].

2. Methodology

Achieving computationally efficient compression, together with a good compression ratio, is the main target of this research. It has been shown that NN can recover image pixels that have been removed from the original image [18]. The pixels' removal is done without any computation effort. It is done simply by removing even rows and even columns from image depth binary matrices. The method was carried out for gray images. To apply the same method on color images, it is advised to follow the same steps for each RGB color component. Each color component is decomposed into its 8 binary depth bits matrices based on:

$$A = \sum_{k=0}^{7} A_k \tag{1}$$

$$A_k = 2^k \times C_k \tag{2}$$

where C_k is N×M binary matrix.

Matrix A_7 has the most significant bits, while A0 has the least significant bits of image A. The color image is decomposed into R_k, G_k and B_k matrices, which correspond to binary matrices of red, green, and blue color components respectively. Even columns and rows are removed from R_0-R_7, G_0-G_7 and B_0-B_7 to constitute the compressed image [18]. Figure 1 shows original image and some of its depth binary matrices.

NN was constructed to predict the removed pixels bits [18]. The NN feature vector is constructed from the image depth matrices as follows. As shown in Fig. 2, bits $R_k(1,2)$, $R_k(2,1)$ and $R_k(2,2)$ are the bits to be estimated at each detection step. These bits for R_6 need R_7 (1,1), R_7 (1,2) , R_7 (1,3) , R_7 (2,1) , R_7 (2,2) , R_7 (2,3) , R_7 (3,1) , R_7 (3,2) and R_7 (3,3) together with $R_6(1,1)$, $R_6(1,3)$, $R_6(3,1)$ and $R_6(3,3)$ to be used as feature vector elements. The next set of bits to be estimated is $R_6(1,4)$, $R_6(2,3)$, $R_6(2,4)$. This can be done by sliding 2 bits to the right and so on [18]. The same process is applied to G and B color components. All R_7 bits are used since they are the most significant bits.

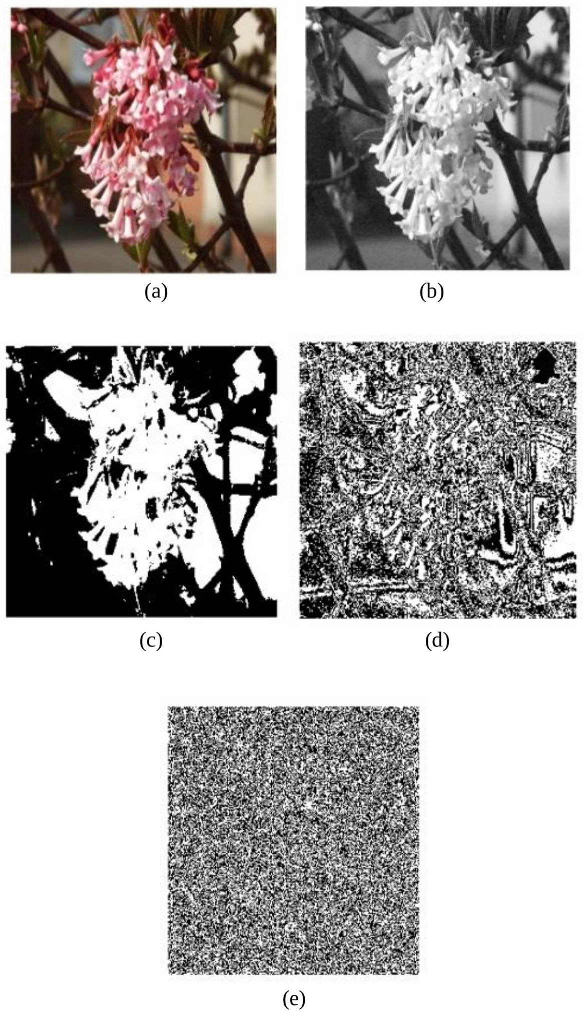


Fig. 1. (a) Original image (b) Red component (c) 8th depth matrix (d) 5th depth matrix (e) 1st depth matrix.

$R_7(1,1)$	$R_7(1,2)$	$R_7(1,3)$	...
$R_7(2,1)$	$R_7(2,2)$	$R_7(2,3)$	...
$R_7(3,1)$	$R_7(3,2)$	$R_7(3,3)$	...
...	...	...	...

$R_6(1,1)$	**$R_6(1,2)$**	$R_6(1,3)$	...
$R_6(2,1)$	**$R_6(2,2)$**	$R_6(2,3)$	...
$R_6(3,1)$	$R_6(3,2)$	$R_6(3,3)$	...
...	...	...	...

Fig. 2. One example of depth 7 and 6 matrices pixels used to construct feature vector and targets.

2.1 Lower-Depth Binary Matrices Elimination

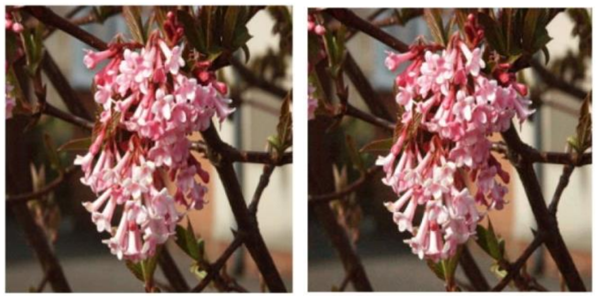
The lower-depth binary matrix of an image is the least significant bit matrix A_0. It shows very little important information to the color component values in terms of appearance. In fact, it looks like random bits, as shown in Fig. 1e. The information capacity is doubled at each higher bit matrix. Based on how much a bit contributes to the maximum pixel value, the sum of information capacity (IC) of binary matrix A0 through A3 together can be calculated as

$$IC_{3-0} = \frac{\sum_{i=0}^{3} 2^i}{2^8} = 5.9\% \tag{3}$$

Thus, bit matrices $A_{3\text{-}0}$ can be removed, while the maximum appearance effect will not exceed 5.9% of its image information. Uncompressed Color Image Dataset (UCID) contains 1338 uncompressed TIFF images each of which is 512 × 384 pixels on various topics including natural and unnatural objects [20]. When applying the calculation to the images in this database, it is found that the average IC are 2.59 %, 2.59% and 2.88% for R, G, and B color components respectively. However, it is recognized that there is some correlation between corresponding bits position in R, G, and B color components that can save us from removing all color components. For example, if $R_{3\text{-}0}$ are replaced by $B_{3\text{-}0}$ (written as $R_{3\text{-}0}$ <- $B_{3\text{-}0}$) the original and modified images are very close in appearance as shown in Fig. 3.

Measuring the correlation between the color components of all images in the UCID database produces correlation values between (R and G) equals 0.92, (R and B) equals 0.84 and (G and B) equals 0.95. The correlation between (R_{3-0} and G_{3-0}), (R_{3-0} and B_{3-0}) and (G_{3-0} and B_{3-0}) is 0.059, 0.049 and 0.064 respectively. It shows that R and B can be replaced by G with minimum possible error. This finding suggests that the substitution can be performed as in Eq. 4.

$$substitution_{i,j} = \arg\max corr\,(i,j)$$
$$where\ i\ \&\ j\ \in \{R,G,B\}\ and\ i \neq\ j \qquad (4)$$



(a) (b)

Fig. 3. Effect of eliminating low depth matrices (a) Original image (b) Image after matrices elimination.

After trying all possible substitutions, it has been found that when R_{3-0} and B_{3-0} are both replaced by G_{3-0}, the result produces the best performance in terms of appearance and compression ratio. This means both R_{3-0} and B_{3-0} can be totally removed from the image depth matrices at the compression stage and G_{3-0} will be used to recover them in the decompression stage. Discussion of the result can be found in the next section. Since there is some correlation between different color components, then it is worthwhile to find this correlation. There have been many studies reporting some findings regarding color components correlation [14-17]. We decided to find the correlation statistically. Once it is found, the correction parameter can be used to enhance the substitution as in the following correction equations

$$R_i = G_i \oplus \xi_{Ri} \qquad (5)$$

$$B_i = G_i \oplus \xi_{Bi} \qquad (6)$$

where:

$i \in \{0\ldots3\}$ represents matrix depth
ξ_R & ξ_B are correction matrices for R and B color components respectively
$\oplus$ is exclusive- OR

$$\xi_{Ri}(i,j) = \begin{cases} 1, & if\ p_{G=R}(i,j) < 0.5, \\ 0, & otherwise \end{cases} \quad (7)$$

$$\xi_{Bi}(i,j) = \begin{cases} 1, & if\ p_{G=B}(i,j) < 0.5, \\ 0, & otherwise \end{cases} \quad (8)$$

where:
$p_{G=R}$ is the probability of G and R pixels value are equal
$p_{G=B}$ is the probability of G and B pixels value are equal

After calculating ξ_B, the result did not show replacement improvement. However, some improvement was observed for ξ_R. This is because G and B have the highest correlation. The PSNR values difference when using G replacement only and when using G replacement with correction matric ξ_R was calculated statistically for the UCID database. Table 1 shows the result. The highest value is 69.5% when using only ξ_{R0}. So it is recommended to use this correction matrix only.

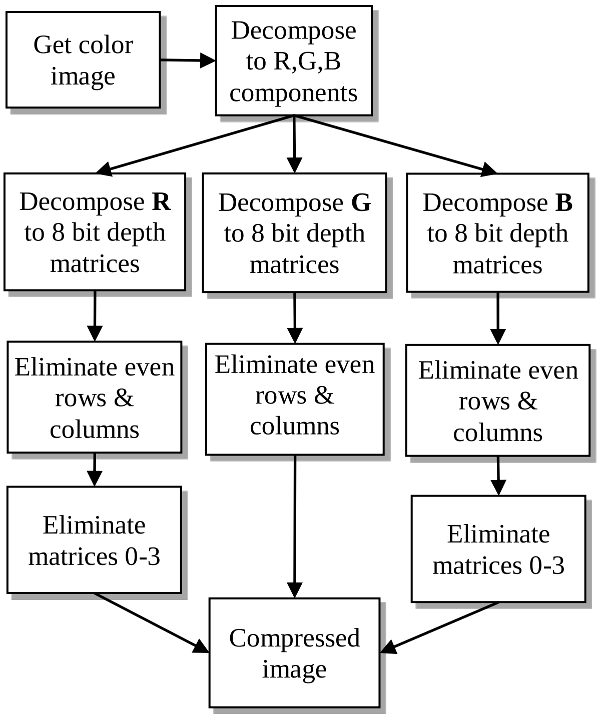
The result in Table 1 shows that almost 70% of the images in the database experience improvement when using G_0 replacement with ξ_{R0} matrix correction. The percentages were lower for higher matrices replacements.

2.2 Various PR Methods Comparison

While NN shows good performance in image compression literature, we decided to investigate the performance of other pattern recognition methods as well. Figure 4 shows the flow chart of the image compression stage. Figure 5 shows the steps to decompress the compressed image.

Table 1. Performance of G replacement with correction matrices.

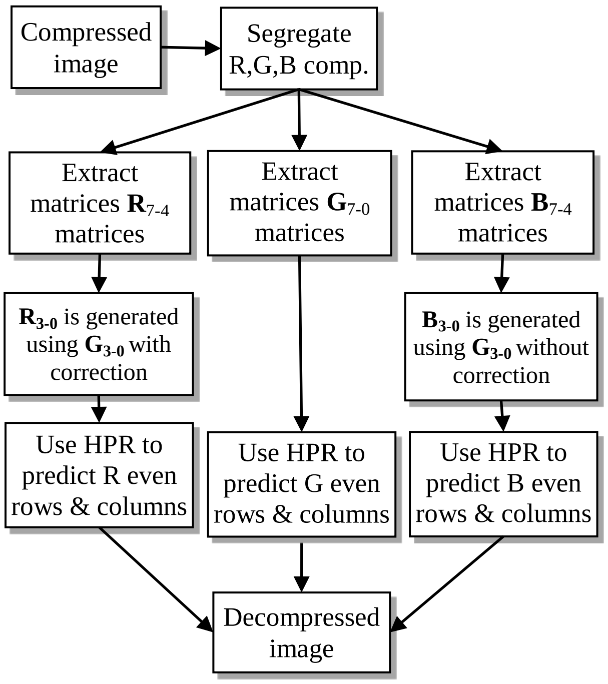
Correction matrices	Database images improvement
ξ_{R0}	69.5%
$\xi_{R0,1}$	67.3%
ξ_{R1}	64.4%
$\xi_{R1,2}$	61.7%
$\xi_{R0,1,2}$	61.7%
$\xi_{R0,2}$	60.1%
ξ_{R2}	59.7%
$\xi_{R0,1,2,3}$	58.7%
$\xi_{R1,2,3}$	58.3%
$\xi_{R0,2,3}$	57.9%
$\xi_{R2,3}$	57.5%
$\xi_{R0,1,3}$	54.8%
$\xi_{R1,3}$	54.2%
$\xi_{R0,3}$	54.1%
ξ_{R3}	53.7%



Get color image
Decompose to R,G,B components
Decompose **R** to 8 bit depth matrices
Decompose **G** to 8 bit depth matrices
Decompose **B** to 8 bit depth matrices
Eliminate even rows & columns
Eliminate even rows & columns
Eliminate even rows & columns
Eliminate matrices 0-3
Eliminate matrices 0-3
Compressed image

Fig. 4. Image compression steps flow chart.

For its popularity and implementation simplicity, the focus was on SVM; decision tree (DT); and classification discriminant (CD), which is Gaussian-based discriminator. MatLab tools were used to construct all these pattern recognition methods. The training data were formed randomly from 50 images of the UCID database. The same training data samples were used for all these pattern recognition methods.



Compressed image
Segregate R,G,B comp.
Extract matrices $\mathbf{R}_{7\text{-}4}$ matrices
Extract matrices $\mathbf{G}_{7\text{-}0}$ matrices
Extract matrices $\mathbf{B}_{7\text{-}4}$ matrices
$\mathbf{R}_{3\text{-}0}$ is generated using $\mathbf{G}_{3\text{-}0}$ with correction
$\mathbf{B}_{3\text{-}0}$ is generated using $\mathbf{G}_{3\text{-}0}$ without correction
Use HPR to predict R even rows & columns
Use HPR to predict G even rows & columns
Use HPR to predict B even rows & columns
Decompressed image

Fig. 5. Image decompression steps flow chart.

3. Results

3.1 Image Quality and CR for Depth Binary Matrices Elimination

The compressed image quality can be measured using the equation:

$$PSNR = 10 \times \log_{10}\left(\frac{255^2}{Mean\ Square\ Error}\right) \quad (9)$$

where

$$MSE = \frac{1}{R \times C}\sum_{i=0}^{R-1}\sum_{j=0}^{C-1}\left(OI(i,j) - CI(i,j)\right)^2 \quad (10)$$

where
MSE: mean square error
R: row size
C: column size
OI: the original image
CI: the compressed image

$$CR = 100 \times \left(1 - \frac{size\ of\ compressed\ image}{size\ of\ original\ image}\right) \quad (11)$$

As mentioned earlier, removing all $R_{3\text{-}0}$, $G_{3\text{-}0}$ and $B_{3\text{-}0}$ will produce a maximum possible MSE of 225 and minimum PSNR of 24.6. If only the LSB matrix is removed, MSE will be 1 and PSNR is 48.2. These were the limits of MSE and PSNR. To find the actual values, the analysis is done on the UCID image. All possible individual bit matrices color components replacements were investigated. Two color components are replaced by the third component. For example, the replacement (R_4 <- G_4, B_4 <- G_4, R_3 <- G_3, B_3 <- G_3, G_2 <- R_2, B_2 <- R_2 and R_1 <- B_1, G_1 <- B_1) is carried out as one trial. All other possible arrangements were tried. It was found that the minimum MSE and maximum PSNR are when $R_{3\text{-}0}$ and $B_{3\text{-}0}$ are replaced by $G_{3\text{-}0}$. In case of one color component replacement, the most repeated replacement with the minimum error is when $G_{3\text{-}0}$ is replaced by $B_{3\text{-}0}$ where 42.7% of the database images conform to this replacement. However, the compression ratio is 70%, which is lower than when using two color components replacements. Since PSNR is a logarithmic function, it may produce infinity if the MSE is equal to zero. Therefore, MSE will be used henceforth. Table 2 shows the above results.

3.2 Performance of Various PR Methods

When using the UCID database, the performances of the pattern recognition methods were comparable to each other with the exception that NN and CD drop slightly for the least three bit matrices. Table 3 shows the four PR methods training performance for the R component. Training results for other components are closely comparable.

Based on the method presented in [18], when investigating each predicted bit (bit_{11}, bit_{13} and bit_{14}) for each bit matrix level, various results were found as shown in the Table 3. For example, the training performance of SVM for bit_{11} in matrix level (R_7) is 92.42% and for bit_{14} it is 91.10%, which outperforms all other methods.

Training and test performances are measured according to Eq. 12.

$$Performance = \left(\frac{sum(abs(target - output))}{size\ of\ original\ image}\right)\% \quad (12)$$

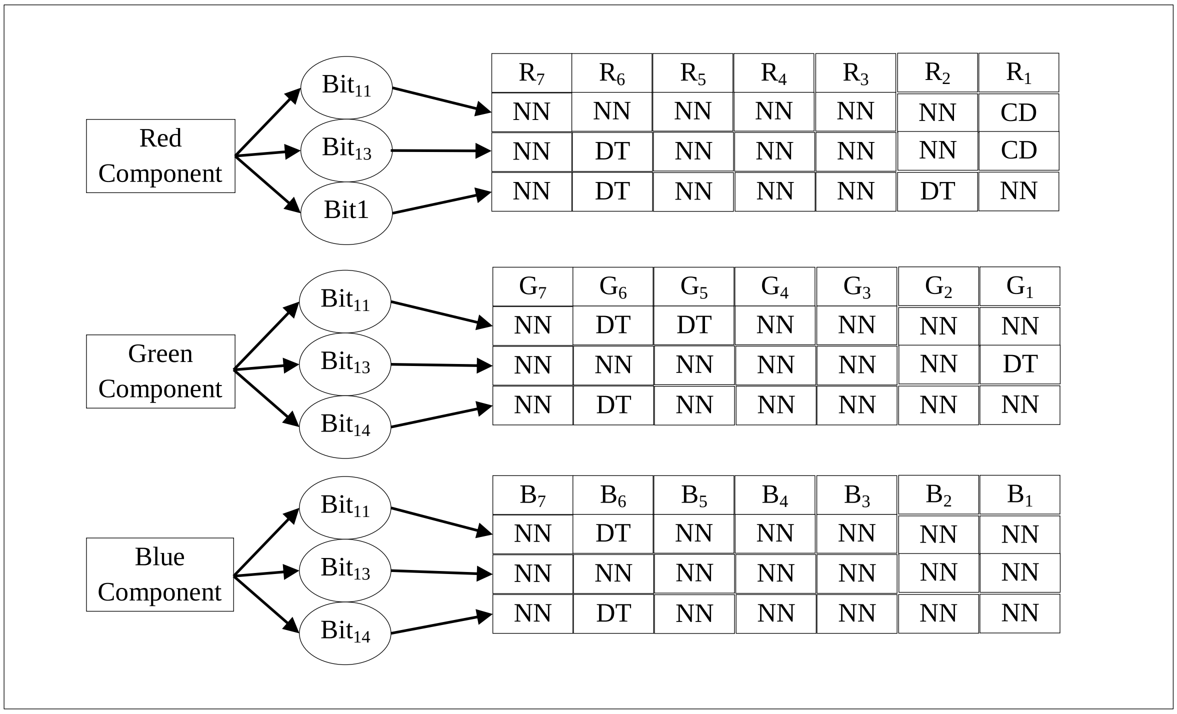
When applying these PR methods to the 1,338 images of the UCID database, some slight differences were found. This is because of the different nature of the various images in the database. The difference is also due to the less general nature of SVM than, for example, NN. Tables 4, 5, and 6 show the testing results. The performance result shown in the Tables 4, 5, and 6 is when, for example, bit_{11} only is predicted while bit_{13} and bit_{14} are the bits in the input image. For the entry "All Bits" in Table 4, all three bits (bit_{11}, bit_{13} and bit_{14}) are predicted and evaluated together which lower the results.

To get the best results, selecting the best individual pattern recognition methods and grouping them in a single hybrid method is preferable. The numbers in bold in Tables 4, 5, and 6 are the highest. Figure 6 shows the resultant hybrid pattern recognition (HPR) method based on these results. These results show that the majority of the best performance comes from NN, with a few exceptions for DT and CD.

The structure in Fig. 6 produces the optimum results of predicting the removed bit_{11}, bit_{13} and bit_{14} from the compressed color images. For its slow execution time, SVM will not be evaluated. At the end it has low prediction performance in this particular application when applied to all images in the UCID database. Reapplying this HPR method to the UCID database yields the MSE and

PSNR results shown in Table 7. Figure 7 shows PSNR values.

Figure 8 shows the effect of proposed method on Peppers (256x256) image. The PSNR is better than the average of the UCID database. The PSNR value for Peppers photo when using G replacement with correction matrix was slightly better than when not using correction matrix as shown in Fig. 8c and Fig. 8d.



	R_7	R_6	R_5	R_4	R_3	R_2	R_1
Bit_{11}	NN	NN	NN	NN	NN	NN	CD
Bit_{13}	NN	DT	NN	NN	NN	NN	CD
Bit1	NN	DT	NN	NN	NN	DT	NN

	G_7	G_6	G_5	G_4	G_3	G_2	G_1
Bit_{11}	NN	DT	DT	NN	NN	NN	NN
Bit_{13}	NN	NN	NN	NN	NN	NN	DT
Bit_{14}	NN	DT	NN	NN	NN	NN	NN

	B_7	B_6	B_5	B_4	B_3	B_2	B_1
Bit_{11}	NN	DT	NN	NN	NN	NN	NN
Bit_{13}	NN	NN	NN	NN	NN	NN	NN
Bit_{14}	NN	DT	NN	NN	NN	NN	NN

Fig. 6. Optimum results of predicting the removed bit11, bit13 and bit14 using various pattern recognition methods.

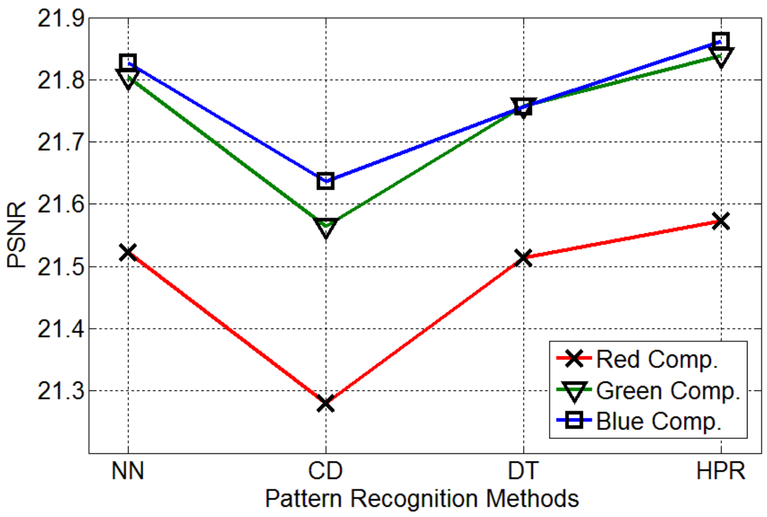


Fig. 7. PSNR results for various PR methods, including the proposed method.

Table 2. Color components matrices replacements performance.

Category	Replacements	(POME)[a] among categories	Average error	CR
One color component	R_{3-0} <- G_{3-0}	35.7%	13.5	70%
	R_{3-0} <- B_{3-0}	21.7%	13.7	70%
	G_{3-0} <- R_{3-0}	0%	13.5	70%
	G_{3-0} <- B_{3-0}	42.7%	13.4	70%
	B_{3-0} <- R_{3-0}	0%	13.7	70%
	B_{3-0} <- G_{3-0}	0%	13.4	70%
Two color components	R_{3-0} <- G_{3-0} B_{3-0} <- G_{3-0}	42.1%	26.8	74%
	R_{3-0} <- B_{3-0} G_{3-0}<- B_{3-0}	32.2%	27.0	74%
	G_{3-0} <- R_{3-0} B_{3-0} <- R_{3-0}	26.7%	27.2	74%
Remove the three color components	NA	NA	81.0%	78%

[a] Percentage of Occurrences as Minimum Error, which is the number of times this replacement occurred in all images divided by the total number of database images

Table 3. Various PR methods training error for RED component.

PR method	Predicted bit	R7	R6	R5	R4	R3	R2	R1
NN	Bit_{11}	91.35	84.03	75.89	65.75	59.99	53.80	51.64
	Bit_{13}	90.98	82.09	72.50	65.47	58.58	54.25	51.90
	Bit_{14}	89.96	80.09	71.18	62.94	57.33	54.10	52.15
SVM	Bit_{11}	92.42	89.98	86.84	83.17	80.17	77.68	76.94
	Bit_{13}	91.82	89.12	85.96	82.13	79.94	77.41	77.19
	Bit_{14}	91.10	87.76	84.30	81.09	79.02	77.58	76.94
CD	Bit_{11}	91.03	78.91	69.35	61.67	56.65	53.25	51.71
	Bit_{13}	90.59	78.35	68.03	61.04	56.07	52.80	52.04
	Bit_{14}	89.55	75.82	66.52	59.94	55.71	53.10	51.07
DT	Bit_{11}	92.09	87.47	82.37	77.69	74.08	71.38	70.65
	Bit_{13}	91.70	86.76	81.44	76.52	73.66	71.18	70.33
	Bit_{14}	90.78	85.06	79.55	75.43	72.98	71.37	70.19

Table 4. Testing error for RED component.

PR method	Predicted bit	R7	R6	R5	R4	R3	R2	R1
NN	Bit_{11}	**98.01**	**95.29**	**93.11**	**91.06**	**89.34**	**88.19**	87.74
	Bit_{13}	**98.06**	94.98	**92.99**	**90.98**	**89.29**	**88.11**	87.75
	Bit_{14}	**97.66**	94.27	**92.26**	**90.46**	**89.02**	88.06	**87.74**
	All Bits	93.73	84.54	78.36	72.50	67.65	64.36	63.23
SVM	Bit_{11}	97.72	95.05	92.91	90.94	89.21	88.12	87.73
	Bit_{13}	97.76	95.08	92.78	90.71	89.11	88.10	87.72
	Bit_{14}	97.29	94.32	92.19	90.34	88.96	88.04	87.70
	All Bits	92.77	84.45	77.88	71.99	67.29	64.26	63.16
CD	Bit_{11}	97.92	94.61	92.79	90.91	89.26	88.14	**87.77**
	Bit_{13}	98.04	94.57	92.80	90.87	89.22	88.07	**87.76**
	Bit_{14}	97.54	93.79	92.02	90.30	88.92	88.04	87.66
	All Bits	93.50	82.97	77.62	72.08	67.40	64.25	63.19
DT	Bit_{11}	97.96	95.22	93.00	90.99	89.26	88.12	87.74
	Bit_{13}	98.06	**95.31**	92.86	90.84	89.14	88.11	87.74
	Bit_{14}	97.54	**94.42**	92.20	90.38	89.00	**88.06**	87.71
	All Bits	93.57	84.95	78.05	72.20	67.40	64.29	63.20

Table 5. Testing error for GREEN component.

PR method	Predicted bit	G7	G6	G5	G4	G3	G2	G1
NN	Bit$_{11}$	**98.19**	95.51	93.36	**91.65**	**90.01**	**88.70**	**87.93**
	Bit$_{13}$	**98.22**	**95.67**	**93.40**	**91.61**	**89.91**	**88.66**	87.84
	Bit$_{14}$	**97.78**	94.33	**92.60**	**90.93**	**89.58**	**88.50**	**87.82**
	All Bits	94.19	85.50	79.36	74.19	69.51	65.86	63.60
SVM	Bit$_{11}$	97.87	95.39	93.27	91.38	89.78	88.53	87.87
	Bit$_{13}$	97.89	95.42	93.29	91.42	89.69	88.54	87.85
	Bit$_{14}$	97.45	94.56	92.49	90.87	89.50	88.44	87.78
	All Bits	93.21	85.37	79.05	73.67	68.97	65.51	63.50
CD	Bit$_{11}$	98.09	94.87	93.26	91.50	89.71	88.66	87.88
	Bit$_{13}$	98.18	94.92	93.22	91.44	89.73	88.60	87.84
	Bit$_{14}$	97.70	94.12	92.52	90.82	89.44	88.47	87.77
	All Bits	93.97	83.91	79.00	73.76	68.89	65.73	63.49
DT	Bit$_{11}$	98.07	**95.53**	**93.42**	91.48	89.90	88.59	87.87
	Bit$_{13}$	98.18	95.60	93.39	91.46	89.77	88.56	**87.87**
	Bit$_{14}$	97.73	**94.64**	92.49	90.87	89.52	88.43	87.78
	All Bits	93.98	85.78	79.30	73.81	69.17	65.58	63.52

Table 6. Testing error for BLUE component.

PR method	Predicted bit	B7	B6	B5	B4	B3	B2	B1
NN	Bit$_{11}$	**98.12**	95.50	**93.45**	**91.40**	**89.68**	**88.43**	**87.87**
	Bit$_{13}$	**98.21**	**95.61**	**93.52**	**91.37**	**89.65**	**88.38**	**87.85**
	Bit$_{14}$	**97.80**	94.44	**92.71**	**90.80**	**89.31**	**88.30**	**87.82**
	All Bits	94.13	85.55	79.68	73.57	68.64	65.11	63.54
SVM	Bit$_{11}$	97.81	95.35	93.29	91.13	89.54	88.36	87.82
	Bit$_{13}$	97.86	95.33	93.33	91.17	89.46	88.31	87.82
	Bit$_{14}$	97.38	94.67	92.56	90.71	89.21	88.27	87.79
	All Bits	93.05	85.35	79.18	73.01	68.21	64.94	63.42
CD	Bit$_{11}$	98.09	94.94	93.26	91.26	89.42	88.23	87.81
	Bit$_{13}$	98.16	95.03	93.24	91.26	89.40	88.23	87.82
	Bit$_{14}$	97.73	94.29	92.58	90.67	88.95	88.10	87.79
	All Bits	93.98	84.26	79.08	73.19	67.77	64.56	63.42
DT	Bit$_{11}$	98.11	**95.56**	93.36	91.28	89.58	88.39	87.83
	Bit$_{13}$	98.11	95.57	93.41	91.25	89.52	88.33	87.82
	Bit$_{14}$	97.69	**94.72**	92.56	90.72	89.23	88.27	87.80
	All Bits	93.91	85.85	79.34	73.25	68.34	65.00	63.45

Table 7. Performance measures of various PR methods.

		NN	CD	DT	HPR
MSE	R	494.90	521.91	497.73	489.52
	G	462.24	487.96	468.70	458.65
	B	464.09	483.89	471.56	460.51
	Avg.	473.74	497.92	479.33	469.56
Avg. PSNR		21.72	21.49	21.68	21.76

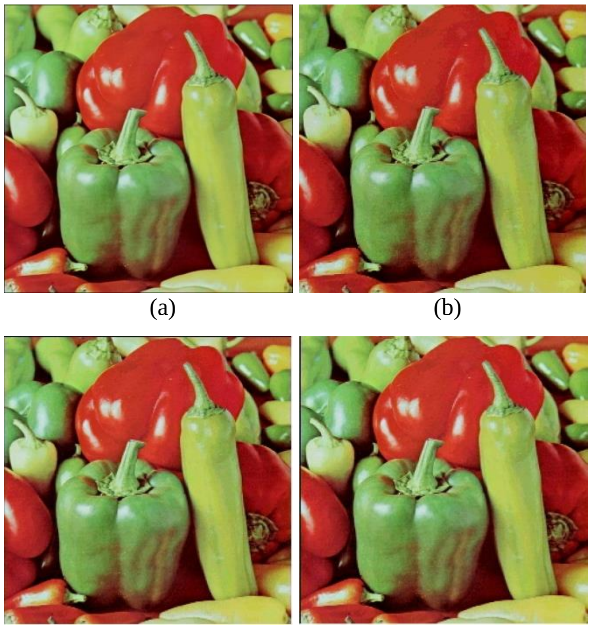

(a) (b) (c) (d)

Fig. 8. Peppers photo (a) Original (b) Decompressed using PR method only (PSNR = 29.69) (c) G replacement without correction (PSNR = 31.99) (d) G replacement with correction (PSNR = 32.00).

There are many studies that showed positive results that can be compared with the present ones. Since the study in [7] involves comparison with the standard JPEG compression method, it would be convenient to compare our proposed method's results with it. Even though PSNR is a standard objective image quality measure, it does not precisely reflect the visual perception of image quality. So in addition to PSNR, Structural Similarity Index Measurement (SSIM) and Universal Image Quality Measurement (UIQI) are also reported [7,19]. SSIM measures the similarity between two images, while the UIQI model measures the image distortion as a combination of loss of correlation, luminance, and contrast distortion. The results reported in that research were shown for various JPEG quality factors. We decided to show the comparison with the least and the most quality factors. Table 8 shows the findings when applying the method to a standard Peppers image.

The PSNR, SSIM, and UIQI values in Table 8 are the average vales computed for R, G, and B components. The proposed HPR method shows higher PSNR than JPEG with the least quality factor while it is much less than the best PSNR reported in [7]. For SSIM, the proposed method is almost equal to the best of JPEG but less than the best reported result. However, our proposed method shows a superior UIQI result than JPEG and is comparable to the reported method. This explains the good appearance of images shown in Figure 8.

Table 8. Performance measures compared to method in [7].

Measuring metrics	JPEG		Rawat and Meher		Proposed HPR
	Min	Max	Min	Max	
PSNR	22.23	30.12	25.99	30.86	22.95
SSIM	0.69	0.86	0.78	0.98	0.85
UIQI	0.29	0.57	0.44	0.80	0.72

For the purpose of evaluating the proposed method in terms of CR and execution time, SPIHT algorithm is used. SPIHT is very effective in terms of compression ratio. Level index is used to choose the preferred CR [21]. For higher CR, higher level should be used. It results also in better quality of decompressed images. However, this is on the account of the processing time. Figure 9 shows the average execution time vs. compression quality (using UIQI) for various SPIHT levels applied on UCID image database. Matlab was run on Intel® Core™ i7 machine with Windows 8 to execute the proposed and SPIHT algorithms. The proposed algorithm is not comparable to SPIHT algorithm in terms of CR. Using level 12 for example; SPIHT needs only 13.5% of the original image size while the proposed algorithm requires 66.7%. At this level, the decompressed images quality and execution time of both algorithms are closed to each other. For higher levels, the SPIHT quality gets higher but the execution time increases while the proposed method has fixed execution time. It is worth mentioning that the proposed method's execution time is for decompression stage only. The compression stage execution time can be considered negligible while it is not in case of SPIHT.

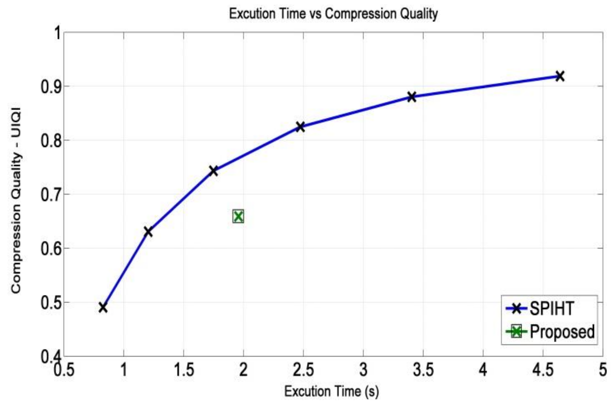


Fig. 9. Execution time vs compression quality for the SPIHT with various levels and the proposed algorithms.

4. Discussion

The proposed method showed the visibility of using the gray image compressing method that is based on pixels elimination for color images. Further elimination applied to lower depth matrices of red and blue components were possible. The green component was used to estimate the other two components with some correction matrices during the decompressing stage. This method reached a 74% compression ratio. The hybrid pattern recognition method composed of NN, CD, and DT showed better performance than individual methods. Even though the difference is insignificant as shown in Table 6, it can be noticed that MSE for HPR is always lower than other methods for the three color components. Since the computation expense of using NN, CD, and DT is comparable, HPR (which is a combination of them) can be used to produce slightly better MSE. The proposed method shows SSIM and UIQI values that are comparable to other studies. This indicates that while PSNR is a little lower than the compared study, the visual quality of the decompressed image using the proposed method is better.

5. Cconclusion

A gray image compression method was extended to be applied to color images and presented in this article. The compression was based on removing rows and columns pixels from all image color components. Also, the low-level depth image matrices of the red and blue color components were removed. This constituted the compressed image. To decompress the compressed image, the removed rows and column were recovered by using hybrid pattern recognition. The eliminated low-depth matrices were recovered using the third color component matrices with correcting factors. The method was tested on a standard image database. It showed PSNR and compression ratios of 21.76dB and 74% respectively. The obtained results were comparable to some positive results reported in the literature. The main advantage of our proposed method is it offers negligible execution time while preserving good image appearance.

References

[1] Dhawan, S., A review of image compression and comparison of its algorithms. *Int. J. Electronics & Comm. Tech.*, 2: 22-26, 2011.

[2] Gonzalez, R., Digital image processing. Addison-Wesley. Third Edition., 2008.

[3] Vijayvargiya, G., Silakari, S. and Pandey, R., A survey: various techniques of image compression. *Int. J. Comput. Sci. Info. Secur.*, 11: 51-55, 2013. https://arxiv.org/ftp/arxiv/papers/1311/1311.6877.pdf, accessed on April 18, 2016.

[4] Koli, N., and Ali, M., A survey on fractal image compression key issue. *Info. Tech. J.*, 7: 1085-1095, 2008.

[5] Sekaran, K. and Kuppusamy, K.: Performance analysis of compression techniques using SVD, BTC, DCT and GP. *J. Comput. Eng.*, 17: 6-11, 2015.

[6] Gunjikar, N., Comparison of Different Image Compression Techniques. *Int. J. Comput. Appl.*, 70: 7-12, 2013.

[7] Rawat, C. and Meher, S., A hybrid image compression scheme using DCT and fractal image compression. *Int. Arab J. Info. Tech.*, 10: 553-562, 2013. http://ccis2k.org/iajit/PDF/vol.10,no.6/4378.pdf, accessed on April 18, 2016.

[8] Xiong, Z., Sun, X. and Wu, F., Block-based image compression with parameter-assistant inpainting. *IEEE Trans. Image Proc.,* 19: 1651-1657, 2010.

[9] Moan, S. and Farup, I., Exploiting Change Blindness for Image Compression. In: International Conference on Signal-Image Technology & Internet-Bases Systems. Nov. 23-27, Bangkok, pp: 89-95, 2015.

[10] Dangman, J., Complete discrete 2-D Gabor transforms by neural networks for image analysis and compression. *IEEE Trans. ASSP,* 36: 1169–1179, 1988.

[11] Amerijckx, C., Verleyse, M., Thissen, P., et al., Image compression by self-organized Kohonen maps. *IEEE Trans. Neural Net.,* 9: 503–507, 1998.

[12] Dutta, D., Choudhury, S., Hussain, M., et al., Digital image compression using neural networks. In: Processing of International Conference on Advances in Computing, Control, and Telecommunication Technologies. pp. 116–120, 2009.

[13] Kecman, V., Combining support vector machine learning with the discrete cosine transform in image compression. *IEEE Trans. Neural Net.,* 14: 950-958, 2003.

[14] Gershikov, E., Lavi-Burlak, E. and Porat, M., Correlation-based approach to color image compression. *Signal Proc.: Image Comm.,* 22: 719-733, 2007.

[15] San, X., Cai, H. and Li, J., Color image coding by using inter-color correlation. In: Proceedings of the IEEE International Conference on Image Processing. Oct. 8-11, pp.3117-3120, 2006.

[16] Gershikov, E. and Porat, M., Correlation vs. decorrelation of color components in image compression - which is preferred?, In: Proceedings of the European Signal Processing Conference. Sept. 3-7, Poznan, PP: 985 – 989, 2007. http://www.eurasip.org/Proceedings/Eusipco/Eusipco2007/Papers/b2l-f03.pdf, accessed on April 18, 2016.

[17] Santhi, M. and Banu, R., Enhancing the color set partitioning in hierarchical tree (SPIHT) algorithm using correlation theory. *J. comput. Sci.,* 7: 1204-2111, 2011.

[18] Alshehri, S., Neural network technique for image compression. *IET image proc.,* 10: 1-5, 2105.

[19] Thayammal, S. and Selvathi, D., Edge preserved image compression using extended shearlet transform. *J. Comput. Sci.,* 11: 82-88, 2015.

[20] Schaefer, G. and Stich, M., UCID – an uncompressed colour image database. In: Proceedings of Storage and Retrieval Methods and Applications for Multimedia. Jan. 18, San Jose, 5307: 472-480, 2004.

[21] Said, A., and William A., A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. on circuits and systems for video technology.,* 6, (3): 243-250, 1996.

ضغط الصور الملونة باستخدام تمييز البكسل الجزئي والتعرف على النمط المختلط

صالح الشهري

قسم علوم الحاسب الآلي، كلية الجبيل الجامعية، مدينة الجبيل الصناعية، المملكة العربية السعودية

shehri@ucj.edu.sa

المستخلص. أدى الاستخدام المتزايد لوسائل الإعلام في حياتنا اليومية إلى الحاجة لتخزين فعال للصور ونقلها. وقد تبين من قبل أن استبعاد بعض بيانات تمثيل الصور (البكسل: أعمدة وصفوف الصور) يمكن استعادته عن طريق استخدام الشبكة العصبية. في هذه المقالة، يتم إجراء مزيد من التحقق في طريقة الضغط للصور الملونة. تستهلك العديد من الطرق الموجودة وقتًا تنفيذيًا كبيرًا نسبياً لضغط الصور. ويعد زمن حساب الطريقة المستخدمة لا يذكر بالمقارنة مع الطرق الأخرى. بالإضافة إلى ذلك، يستغرق زمن الحساب وقتًا أقل من النصف، ويحتاج وقت تنفيذ أقل لضغط الصور في جودة أكبر من ٠.٧٥.

الكلمات المفتاحية: ضغط الصور، التعرف على الأنماط، الشبكات العصبية، إزالة بيكسل الصور، استبدال مكونات اللون.