Bitlis Eren Üniversitesi Fen Bilimleri Dergisi
BİTLİS EREN UNIVERSITY JOURNAL OF SCIENCE
ISSN: 2147-3129/e-ISSN: 2147-3188
VOLUME: 12 NO: 1 PAGE: 215-225 YEAR: 2023
DOI:10.17798/bitlisfen.1225312

An Image Compression Method Based on Subspace and Downsampling

Serkan KESER[1*]

[1]*Kirsehir Ahi Evran University, Department of Electrical and Electronics Engineering, Kirsehir 40100, Turkey*

(ORCID: 0000-0001-8435-0507)

Keywords: Image compression, Downsampling, Transform coefficient matrix, KLT

Abstract

This study proposes a new Karhunen-Loeve transform-based algorithm with acceptable computational complexity for lossy image compression. The proposed study includes a simple algorithm using downsampling and KLT. This algorithm is based on an autocorrelation matrix found by clustering the highly correlated image rows obtained by applying downsampling to an image. The KLT is applied to the blocks of the downsampled image using the eigenvector matrix of the autocorrelation matrix, and the transform coefficient matrix is obtained. One of the important features of the proposed method (PM) is sufficient for a test image to have one transform matrix with low dimensional. The PM was compared with JPEG, BPG, and JPEG2000 compression methods for the Peak signal-to-noise ratio- Human Visual System (PSNR-HVS) and the Structural Similarity Index Measure (SSIM) metrics. The mean PSNR-HVS values of the PM, JPEG, JPEG2000, and BPG were 37.44, 37.16, 37.45, and 39.14, respectively, and their mean SSIM values were 0.95, 0.93, 0.952, and 0.962, respectively. It has been observed that the PM generally gives better results than other methods for images containing low-frequency components at high compression ratios. As a result, PM can be used to compress images, especially those containing low-frequency components.

1. Introduction

Transform encoding is often used to code an image. The purpose of transform coding is to represent the image with fewer data without degrading the quality of the image too much. [1]. Furthermore, transform coding-based image compression applications require low calculation costs. Transform coding is a popular approach to signal compression by compacting the signal's energy to fewer coefficients, enabling the signal's representation with as few bits as possible [2-4]. Various transforms have been applied to signal and image compression, including classical transforms like Discrete Fourier Transform (DFT), Walsh-Hadamard Transform (WHT), Discrete Cosine Transform (DCT), Wavelet Transform (WT), and Karhunen Loeve Transform (KLT). Some of these studies are used in the health areas [5-7]. Thus, large-scale data can be compressed and converted into smaller-sized data in these areas, and faster analysis can be performed. A wavelet-based algorithm is proposed for satellite image compression in [8]. Digital image compression using the Walsh Hadamard transform is performed in [9]. An image compression algorithm for the predictor of JPEG-LS has been used in [10]. In another paper [11], the author explains the JPEG2000 working structure comprehensively with examples, and the authors analyze wavelet transform and give a detailed introduction to Wavelet Toolbox software [12].

Usually, transform coding-based compression methods are preferred for lossy compression cases. DCT- and Wavelet Transform-based methods are widely used for image compression. DCT is the most popular transform in JPEG-based methods [13,14], and WT also became

* Corresponding author:skeser@ahievran@edu.tr
Received: 27.12.2022, Accepted: 02.03.2023

popular with JPEG2000 [15-17]. The Better Portable Graphics (BPG) is a state-of-the-art image compression format for coding digital images, and it is a new image format. Its main advantages are high compression rate, compression much more than JPEG for similar quality, and a subset of the High-Efficiency Video Coding (HEVC) open video compression standard [18-20]. The HEVC standard is a newly designed video compressor [21]. It has also been developed in a new format, such as AV1, that performs effective video compression [22]. When the image compression methods, including downsampling and upsampling in the literature, are examined, it is seen that downsampling is performed on the encoder side and upsampling on the decoder side. These studies have shown that images can be compressed using image coding standards (JPEG, JPEG2000) at low bit rates [23-29].

Bruckstein et al. [23] show how downsampling an image at low bit rates is applied, and the overall PSNR performance using the JPEG standard is improved. In [24], the authors first filtered the image with an all-phase DCT filter and then compressed it using JPEG. Another similar method is designed as an interpolation-dependent image downsampling method [25]. In another study, researchers have presented an image coding method based on random downsampling and compressive sensing image recovery method [26]. In [27], a novel perceptual image coding scheme is proposed via adaptive block-based super-resolution directed downsampling. However, the algorithms mentioned above are only applied to low-bit-rate image compression since their performance would degrade with the increasing coding bit rate [27]. In [28], an adaptive downsampling method is developed to solve this problem. This method is developed to be compared with standard compression methods at all bit rates, unlike these studies in [25] and [26]. The same idea improves video compression performance at low bit rates [29]. If the signal is well-correlated, most of the signal's energy can be compressed into a few transform coefficients by applying KLT. Although KLT (or PCA) is the most efficient orthogonal transform in energy compression and decorrelation, it was not used for real compression applications due to its signal-dependent nature [30]. However, with the increasing computational power of communication systems and computers, signal-specific methods (such as KLT-based compression algorithms) can be applied for real image compression applications using intelligent techniques [30-34].

In [30], their method first spectrally decorrelates the image using Vector Quantization and Principal Component Analysis (PCA) and then applies the JPEG2000 algorithm to the Principal Components (PCs) exploiting spatial correlations for compression. A method proposed for image compression with PCA blocks of images is trained using a Generalized Hebbian Algorithm (GHA), and then eigenvectors are estimated [31]. Another study selected Eigen images with maximum energy for face image compression with PCA [32]. PCA and Wavelet Difference Reduction (WDR) coding-based techniques have been proposed in another method, which combines PCA and Wavelet Difference Reduction (WDR)-based approaches to achieve a high compression ratio while maintaining the perceptual quality of the image [33]. This study obtained better results than JPEG2000 at high compression rates with the WDR and PCA methods. Another paper proposes an adaptive image transformation (AIT) approach for a group of image blocks that derives transformation kernels from KLT [34]. This study emphasizes that the obtained results are better than JPEG, especially at low compression ratios. In addition, a lossy image compression architecture utilizes the advantages of a convolutional autoencoder (CAE) to achieve a high coding efficiency [35]. In this paper, to generate a more energy-compact representation, they use the principal components analysis (PCA) to rotate the feature maps produced by the CAE and then apply the quantization and entropy coder to generate the codes.

This study developed a new KLT-based image lossy compression algorithm with a good compression performance and reasonable computational complexity. In addition, one other difference between the proposed study and the studies above is that PCA and JPEG2000, WDR, CAE, and AIT are used as hybrids in those studies. Unlike the studies mentioned above, the PM presents a simple algorithm that includes only the downsampling of images and the application of KLT. One of the biggest problems of KLT-based image compression is sending the calculated transform matrix to the decoder side. If the size of this matrix is big, then more computations and bits need to be sent. In this study, an 8x8-dimensional transform matrix is used for a test image; thus, the computational complexity and the number of bits sent have been significantly reduced. First, the image ($N \times N$) is used for two subsampling in the horizontal and vertical directions, and four sub-images ($N / 2 \times N / 2$) are obtained to increase the correlation between the rows of a test image. The rows of these four sub-images are combined, and a new image of $2N \times N/2$ size is obtained. Then, the KLT is applied to all image blocks using eigenvectors of the autocorrelation matrix

obtained from this new image's blocks. Transform (KL) coefficients are obtained using eigenvectors corresponding to the largest M eigenvalues. Next, an algorithm based on Huffman coding [36] has been applied to transform coefficient blocks. This algorithm is applied to the differences among consecutive column coefficients. Thus, if the similarity of the successive transform coefficients increases, higher compression ratios can be achieved using the proposed algorithm. In addition, the proposed algorithms can be used for the image's sizes of N×P or P×N. This paper used the PSNR-HVS and SSIM methods to find better the image quality perceived by humans [37, 38]. In experimental studies, it has been seen that better results can be obtained by using the proposed simple compression algorithm, especially for images containing low-frequency components and highly compressed images. The study is organized as follows. First, the proposed KLT-based method is introduced in section 2. In this section, the encoder, difference matrices, Huffman coding method, decoder, and the complexity of the proposed algorithm are explained, respectively. Then, experimental results and discussions are given in Section 3, and the conclusion is given in section 4.

2. Material and Method

2.1. Materials

Forty test images have been used for the experimental research of this study. The test images are grayscale with a resolution of 8 bits ranging from sizes 256×256, 300×300, 512×512, and 640×640. The images with different structures from different databases were used to see the performance of the proposed method. While some of these images are commonly used in the literature as Barbara, Cameraman, Boat, and Baboon, others consist of the Brodatz texture [41], Salzburg Texture Image Database (Stex-512) [42], and Caltech-101 image databases [43].

2.2. Methods

If the signal is well-correlated, most of the signal's energy can be compressed into a few transform coefficients by applying KLT. Firstly, KLT is applied to all image blocks using eigenvectors of the autocorrelation matrix obtained from the blocks of the new image of 2N×N/2 size. In this way, a transform matrix of only 8×8 is sent to the decoder side for a test picture. In the study, Huffman coding has been applied to the difference matrix of the transform coefficient. In experimental studies, the proposed method's PSNR-HVS and SSIM values have been compared to JPEG, JPEG2000, and BFG.

2.2.1. The Proposed Subspace Method

An 8×8-dimensional eigenvector matrix has been used in the study. M-dimensional transform coefficients are obtained using the eigenvectors corresponding to the largest M eigenvalues (M<8). Next, the difference matrix ($\boldsymbol{C_{diff}}$) is found by taking the successive differences of the transform coefficient vectors to make more compression. The proposed compression method contains the following essential parts:

(i) The difference matrices are found on the encoder side using preprocessing the image and applying KLT to the blocks. Then, obtained transform matrices are coded by using Huffman coding. Finally, the bits of the difference matrices, the eigenvector matrix ($\boldsymbol{\Phi}$), and the obtained average block vectors ($\mathbf{V}$) by using DCT are sent to the decoder side.

(ii) The bits of these matrices are obtained to reconstruct the test image on the receiving side. First, all blocks are reconstructed using the generated eigenvector matrices and transform coefficients. Next, these blocks are combined, and an image of size 2N×N/2 is performed. Finally, the image (2N×N/2) is upsampled to obtain a reconstructed image of size N×N.

2.2.2. Preprocessing of the Test Images

On the encoder side, the size (N×N) of the original image is converted to a 2N×N/2-dimensional image by downsampling. Then, the image is divided into 8×N/2-dimensional blocks, and an autocorrelation matrix is found using all of them. Firstly, the image is downsampled using the mathematical expressions below, and sub-images are obtained

$$\boldsymbol{I_s} = \boldsymbol{I}(k_a, k_b) \; a=1,2 \text{ and } b=1,2, \qquad (1)$$

where $\boldsymbol{I}$ is shown test images, k_1 ($k_1 = 1,3,5,\dots,N-1$) and k_2 ($k_2 = 2,4,6,\dots,N$) represent the pixel indices of the rows and columns of images, and $\mathbf{I}_s$ represents the sub-images. Four sub-images are found as follows,

$$\boldsymbol{I_1} = \boldsymbol{I}(k_a, k_b), \qquad a=1 \text{ and } b=1,$$

$$\boldsymbol{I_2} = \boldsymbol{I}(k_a, k_b)\,, \quad \text{a=1 and b=2,}$$
$$\boldsymbol{I_3} = \boldsymbol{I}(k_a, k_b)\,, \quad \text{a=2 and b=1,}$$
$$\boldsymbol{I_4} = \boldsymbol{I}(k_a, k_b)\,, \quad \text{a=2 and b=2,}$$

i-th row of each sub-image is concatenated, and a 2N×N/2 -dimensional matrix is obtained. The 2N×N/2-dimensional image is shown in Figure 1 below.

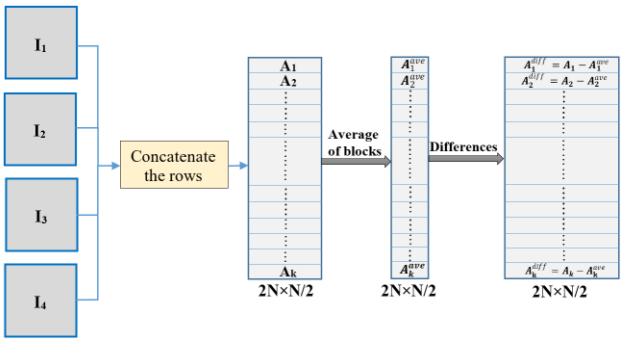


Figure 1. Obtaining a new image of size 2NxN/2 by downsampling

$$\boldsymbol{A_k} = \boldsymbol{A}(i,j), \tag{2}$$

where $\boldsymbol{A_k} \in \boldsymbol{R}^{\mathbf{8}\times\frac{N}{2}}$, *i*=1,2,…,8, *j*=1,2,…, *N*/2, *k*=1,2,…, *N*/4, indices *i* and *j* are the row and column of **A**, respectively. The total number of matrices of size *8×N/2* is *N/4*, and an average vector of 8×1 length is obtained as follows for each block ($\boldsymbol{A_k}$).

$$\boldsymbol{A_k^{ave}} = \frac{1}{\left(\frac{N}{2}\right)} \sum_{m=1}^{\frac{N}{2}} \boldsymbol{A_k}(m), \tag{3}$$

where $\boldsymbol{A_k^{ave}} \in \boldsymbol{R}^{\mathbf{8\times 1}}$, *k*=1,2,…, N/4, and *m* is the column index of $\boldsymbol{A_k}$. Then, the *k-th* mean vector ($\boldsymbol{A_k^{ave}}$) is subtracted from the *m-th* column of $\mathbf{A_k}$, and the *k-th* difference matrix $\boldsymbol{A_k^{diff}}$ is found as follows,

$$\boldsymbol{A_k^{diff}} = \boldsymbol{A_k}(m) - \boldsymbol{A_k^{ave}}, \tag{4}$$

where $\boldsymbol{A_k^{diff}} \in \mathbf{R}^{8\times\frac{N}{2}}$ and *m*=1,2,…,N/2. The autocorrelation matrices are formed by using these matrices ($\mathbf{A}^{\text{diff}}$). The autocorrelation matrix for the difference blocks is obtained as follows,

$$\mathbf{U} = \sum_{r=1}^{k} (\boldsymbol{A_r^{diff}})(\boldsymbol{A_r^{diff}})^T, \tag{5}$$

where $\mathbf{U} \in \mathbf{R}^{\mathbf{8\times 8}}$. Eigenvectors are calculated for the autocorrelation matrix ($\boldsymbol{U}$). When the eigenvalues of the autocorrelation matrix are sorted in descending order $\{\lambda_1 > \lambda_2 > \ldots > \lambda_8\}$, an eigenvector matrix, Φ, is formed by stacking the eigenvectors corresponding to **U**'s largest M eigenvalues, as shown in Equation 2.

$$\boldsymbol{\Phi} = \{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_M\}, \tag{6}$$

where $\boldsymbol{\phi}$'s are 8-dimensional eigenvectors and $\boldsymbol{\Phi} \in \mathbf{R}^{\mathbf{8\times M}}$. KLT is applied to the blocks using eigenvector and autocorrelation matrices, and transform coefficients are obtained. The *k-th* coefficient matrix ($\mathbf{Y}_k$), which is the corresponding *k-th* block, can be written as

$$\mathbf{Y}_k = \boldsymbol{\Phi}^T \boldsymbol{A}_k^{diff}, \tag{7}$$

where $\mathbf{Y}_k \in \mathbf{R}^{\mathbf{M\times N/2}}$. Then, quantization is achieved by sample–wise dividing each column in the *k-th* transform matrix ($\mathbf{Y}_k$) with the corresponding column in the quantization vector (**q**), then rounding to the nearest integer value.

$$\mathbf{C_k^{(u,v)}} = round\left(\frac{\mathbf{Y_k^{(u,v)}}}{\boldsymbol{\alpha}.\mathbf{q^{(u,v)}}}\right) \tag{8}$$

where the quantization vector (***q***) is weighted using a scalar quantization (quality) constant (***α***). This way, the compression ratio is controlled using the scalar coefficient (α) and the selected coefficient (M) size. All the transform matrices ($\mathbf{C_k}$) obtained in Equation 8. are combined, and the matrix **C** of size $M\times N^2/8$ is obtained.

2.2.3. Transform Difference Matrix (TDM)

Subsequently, for the columns of the combined transform matrix $\boldsymbol{C}$, ($\boldsymbol{C} \in \boldsymbol{R}^{\boldsymbol{M}\times\frac{\boldsymbol{N^2}}{\mathbf{8}}}$), the *t*-th column of the $\boldsymbol{C}$ is subtracted from its *(t+1)*-th column. Thus, only the difference values are compressed and sent. This process allows fewer bits to be assigned to the elements of consecutive columns. The pseudo-code of the algorithm is as follows.

Algorithm 1: Calculating the transform difference matrix

Input: Transform matrix ($\boldsymbol{C}$)
Output: Difference matrix ($\boldsymbol{C_{diff}}$)
for t=1 **to** $N^2/8$-1 **do**
Calculate *t*-th difference column,
$\boldsymbol{C_{diff}(t) = C(t) - C(t+1)}$
end
Equalize the first column of $\boldsymbol{C_{diff}}$ to the first column of $\boldsymbol{C}$,
$\boldsymbol{C_{diff}(1) = C(1)}$

Algorithms can be developed to assign a bit less for the difference matrix $\boldsymbol{C_{diff}}$ according to matrix **C,** and thus the image can be better compressed. In experimental studies, it is seen that column vectors are generally similar to each other, especially for consecutive similar columns. The most common ones are given in Figure 2 below.

$$\begin{bmatrix} 0 & 1 & -1 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 2. Some vectors commonly encountered in difference matrices

The algorithm developed focused on similar columns mentioned above.

2.2.4. Class Bits and Average DCT Coefficients

Many bits will be assigned to each vector element. However, similar columns can be clustered, and Huffman encoding can be applied depending on the possibilities generated by the column number of each cluster. When the obtained TDM was examined, most consecutive columns had similar values. These columns are combined, and Huffman coding is applied as a vector. The same columns of the TDM are stacked using vector quantization, and clusters are obtained. The clusters with the most number of columns are sorted in descending order, $C_s=\{\boldsymbol{C_1}, \boldsymbol{C_2}, \boldsymbol{C_3}, \dots, \boldsymbol{C_m}\}$, where *m* is the number of clusters and $\boldsymbol{C_s} \in \boldsymbol{R^{M\times(N^2/8)}}$. Class A is created by taking the ones corresponding to the first *t* ($\boldsymbol{C_1}, \boldsymbol{C_2}, \boldsymbol{C_t}$). The dictionary-A is obtained by applying Huffman coding according to the number of columns in the clusters of class A. Class B is performed with the remaining clusters ($\boldsymbol{C_{t+1}}, \boldsymbol{C_{t+2}}, \dots, \boldsymbol{C_m}$). The cluster's columns of class B are combined and create the vector **V**. Finally, the dictionary-B is obtained by applying Huffman coding to the vector **V** [36]. If the *l*-th column of the TDM of a test image belongs to class A, the bits are assigned according to dictionary-A. Otherwise, the bits are set to each element in the column using dictionary-B. An additional bit assignment is used for each column vector at the encoder side. This bit represents the class of columns. If the column belongs to class B, zero bit '0' is used; otherwise, one bit '1' is used. These bits are called 'class bits'. Besides, 1D Discrete cosine transform (DCT-II) is applied for each *k-th* average vector ($\boldsymbol{A_k^{ave}}$) in the study. The size of the DCT coefficient vector is eight. This vector is given for each mean vector as follows,

$$\boldsymbol{D_k}[s] = \text{Round}\left(\sum_{n=0}^{7} 2(\boldsymbol{A_k^{ave}}[n]) \cos\left(\frac{\pi}{16} s(2n+1)\right)\right), 0 \le s < 8 \quad (9)$$

The vector ($\boldsymbol{F}=\begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_k \end{bmatrix}$) containing the DCT coefficients is obtained by combining all the DCT vectors.

where $\mathbf{F} \in \mathbf{R^{2N\times 1}}$. Huffman coding is applied for the **F** vector. Then the bits of the dictionary and DCT coefficients are sent to the receiving side. In Figure 3, the encoder parts of the proposed method are shown for a test image.

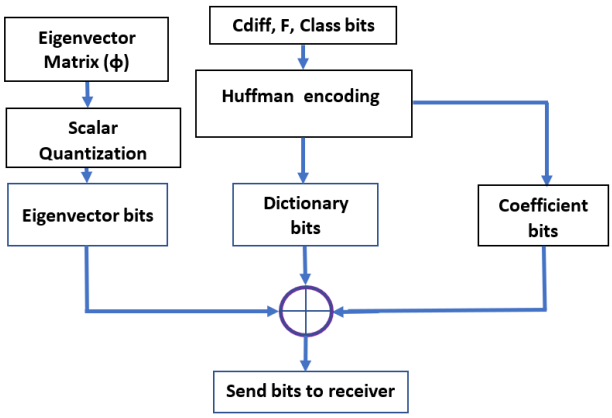


Figure 3. Block diagram of the encoder side

As can be seen from Figure 3, the transform difference matrix ($\boldsymbol{C_{diff}}$), **V,** and class bits are coded by Huffman coding. Thus, the dictionary and coefficient bits are obtained. The bits of eigenvectors, dictionaries, and coefficients are sent to the receiver side. In the encoder part, the bits of the eigenvectors' elements corresponding only to the largest *M* eigenvalues are sent to the decoder. Then, the 10-bit assignment is performed by scalar quantization for each element of the remaining eigenvectors. Eigenvectors corresponding to the largest M (M<8) eigenvalues of autocorrelation matrices are used in the study.

2.2.5. Reconstruction of the Test Images

On the decoder side, the received difference matrices, average vectors ($\widehat{\boldsymbol{A}}_k^{ave}$) dictionaries and the bits of the eigenvector matrix are decoded. Firstly, the difference matrix is obtained, then the consecutive

columns of the TDM are summed using the algorithm given below, and the transform matrix ($\boldsymbol{C}$) is found.

Algorithm 2: Calculating the transform matrix

Input: Transform matrix ($\boldsymbol{C}$)
Output: Difference matrix ($\boldsymbol{C_{diff}}$)
for t=1 **to** N/2-1 **do**
Calculate *t*-th column,
$C(t) = C_{diff}(t) + C_{diff}(t+1)$
end

The *k-th* DCT coefficients $(\widehat{\boldsymbol{D}}_k)$ are obtained by decoding the bits on the receiving side. Then, with inverse DCT, approximate *k-th* average vectors ($\widehat{\boldsymbol{A}}_k^{ave}$) are found. The *k-th* dequantized transform matrix for the image blocks becomes:

$$\widehat{\boldsymbol{Y}}_k = \boldsymbol{q}^{u,v}\boldsymbol{C}_{\boldsymbol{k}}^{\boldsymbol{u,v}},\ k=1,2,\ldots,N/4 \qquad (10)$$

where $\boldsymbol{C_k}, \widehat{\boldsymbol{Y}}_k \in \boldsymbol{R}^{\boldsymbol{M}\times\frac{\boldsymbol{N}}{\boldsymbol{2}}}$. Then, the blocks of the image are reconstructed using the eigenvector matrix. The *k-th* decompressed image block ($\widehat{\boldsymbol{I}}_{\boldsymbol{k}} \in \boldsymbol{R}^{\boldsymbol{8}\times\frac{\boldsymbol{N}}{\boldsymbol{2}}}$) is obtained by using the eigenvector matrix and dequantized transform matrix as follows,

$$\widehat{\boldsymbol{I}}_{\boldsymbol{k}} = \boldsymbol{\Phi}\widehat{\boldsymbol{Y}}_{\boldsymbol{k}} + \widehat{\boldsymbol{A}}_{\boldsymbol{k}}^{\boldsymbol{ave}}, \qquad (11)$$

these blocks are combined, and the reconstructed image ($\widehat{\boldsymbol{I}}$) of size *2N×N/2* is obtained. Then, four sub-images are performed from this image ($\widehat{\boldsymbol{I}}$). Each sub-mage is upsampled by 2, and the reconstructed image (*N×N*) is found. All of these processes are shown in Figure 4. DaC indicates Dictionary and Coefficient in Figure 4

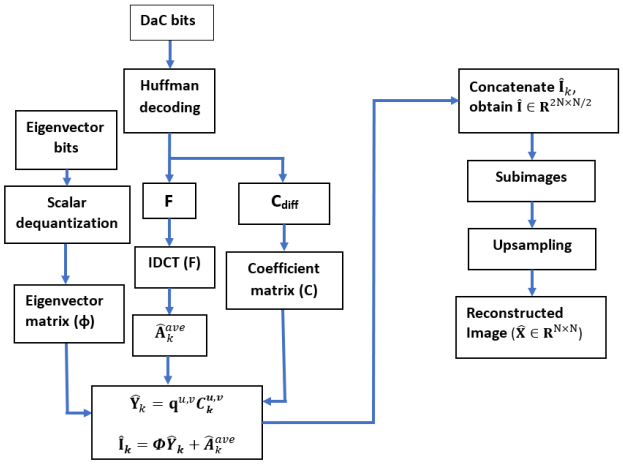


Figure 4. Block diagram of the decoder side

2.2.6. The Complexity of the Proposed Algorithm

Two critical factors in image coding are the algorithm's compression rate and execution complexity. For the proposed method's algorithm complexity, the sum of coding and decoding times per pixel of the image was measured in microseconds. It is known that JPEG has less computational cost than JPEG2000 [39]. Therefore, only the computational cost of the JPEG is compared with the proposed method in this study. Matlab codes in [40] were used for the algorithm of the JPEG standard. The proposed method (PM) was evaluated for three different compression rates (BPP) ranges. The average microseconds per pixel (ASPP) values for 40 test images are given in Table 1 below.

Table 1. The average ASPP values of PM and JPEG for three different compression ranges

BPP	PM (ASPP)	JPEG (ASPP)
(1-1.6)	29.7	10.8
(0.66-0.9)	10.2	10.6
(0.4-0.61)	5.7	10.5

As shown in Table 1, the PM has less ASPP than Jpeg, especially for BPP: (0.66-0.9). Especially at high compression rates (BPP: (0.4-0.61)), the ASPP of the PM is about half that of JPEG. However, when BPP: (1-1.6), the PM's ASPP value is approximately three times that of JPEG

3. Experimental Results and Discussion

The algorithm execution times have been obtained on a desktop PC with a 3 GHz (I5-7400) and 8 GB Ram under Windows 10. All images are individually compressed using PM, JPEG, JPEG2000, and BPG at different compression rates ranging from 0.4 bpp to 2 bpp. Some of the test images belonging to various databases are shown in Figure 5 below
The SSIM and PSNR-HVS values obtained for each image are given in Figure 6 below. Test image numbers and compression ratios of images are also shown in the columns to the right of Figure 6
Figures 6 (a) and 6 (b) show that the PSNR-HVS and SSIM values of BPG are higher than those of the PM, JPEG, and JPEG2000 for most test images. Besides, the PSNR-HVS and SSIM values of the PM are better than those of JPEG and JPEG2000 for some images. However, BPG has been the method that produces the best quality images in general for 40 test images. Images containing high-frequency components are generally low-compression images in figure 6. The proposed method for these images gave a lower performance, especially compared to JPEG2000 and

BPG. Nevertheless, the PM produced images of better or approximate quality compared to other methods for images containing low-frequency components. On the other hand, it is seen that PM gives lower PSNR-HVS values in some images with indices 6, 8, 25, 26, 27, and 35 than in other methods. When these images were examined, it was seen that they generally had high frequencies. Therefore, the average SSIM and PSNR-HVS values of four methods are given for these images in Table 2 below**.**

According to the results shown in Table 2, the BPG has the best scores in terms of average SSIM and PSNR-HVS values. Besides, the PM's SSIM and PSNR-HVS values are higher than JPEG. As a result of the study, the PM generates better scores than JPEG but worse than BPG and JPEG2000 in average SSIM and PSNR-HVS values. On the other hand, in some images, such as the test image in Figure 7, PM achieved much better results than other methods. When this image is examined, it can be seen that it has high and low frequency regions. This shows that PM can give good results in images with different frequencies. For example, a reconstructed test image found using the PM, JPEG, JPEG2000, and BPG is presented in Figure 7, respectively

Although the proposed method's average PSNR-HVS and SSIM values were lower than the BPG, its PSNR-HVS and SSIM values are significantly better than other methods for test images in Figure 7.

Some studies using PCA in the literature can compress images with better quality than JPEG2000 [30,33,35]. However, these studies used algorithms such as hybrid PCA+ JPEG2000 [30], PCA+Wavelet [33], and PCA+CAE [35]. Therefore, the computation time of these studies is longer than JPEG and JPEG2000. The proposed method, on the other hand, has less or close computation time compared to JPEG and JPEG2000, especially at high or medium compression ratios. Because it uses only a KLT-based simple compression algorithm. However, when the same images used in [33] and in this study were examined, it was seen that the study in [33] gave higher PSNR values than the PM.

3. Conclusion and Suggestions

This manuscript generates a new KLT-based method for performing lossy image compression. The main goal of this method is to provide a good compression performance at reasonable computational complexity. Unfortunately, KLT is a signal-dependent transform that disadvantages practical applications. However, this problem was overcome by using a transform (eigenvector) matrix, whose size is as small as possible. This study differs from other studies as it obtains highly correlated pixel blocks with downsampling to compress the image better. The study proposes a KLT-based image compression method applied to the merged blocks of sub-images found by downsampling. The differences among consecutive transform coefficient columns are compressed with this algorithm. It has been observed that the transform matrix used in this way gives satisfactory results. This study only uses an 8x8-dimensional transform matrix for a test image. First, the image is subsampled, and four sub-images are obtained. Next, the rows of these four sub-images are combined, and a new image of 2N×N/2 size is obtained. Then, the KLT is applied to all image blocks using eigenvectors of the autocorrelation matrix obtained from this new image's blocks. Next, transform coefficients are obtained using eigenvectors corresponding to the largest M eigenvalues. Next, Huffman coding is applied to the differences between consecutive column coefficients. In the study, test images are collected from different databases. Thus, the performance of the proposed method is examined for images with various frequency components. In the results found for the test images, the average PSNR-HVS and SSIM values of the proposed method were better than JPEG, very close to JPEG2000, and lower than BPG. The proposed method's mean PSNR-HVS and SSIM values were 37.44 and 0.950, respectively. The mean PSNR-HVS values for JPEG, JPEG2000, and BPG were 37.16, 37.45, and 39.14, respectively, and their mean SSIM values were 0.930, 0.952, and 0.962, respectively. It has been observed that the PM generally gives better results than those of the JPEG and JPEG2000 in images containing low-frequency components at high compression ratios. However, the compression performance of the PM was decreased in images containing high-frequency components at low compression ratios. The computation time of the PM is about two times less than JPEG at compression ratios between 0.4 and 0.6, and it was close to JPEG between 0.66 and 0.9. When these results are examined, it is seen that the PM gives better or close results compared to other methods, especially for images containing low-frequency components. Thus, PM can be used to compress such images

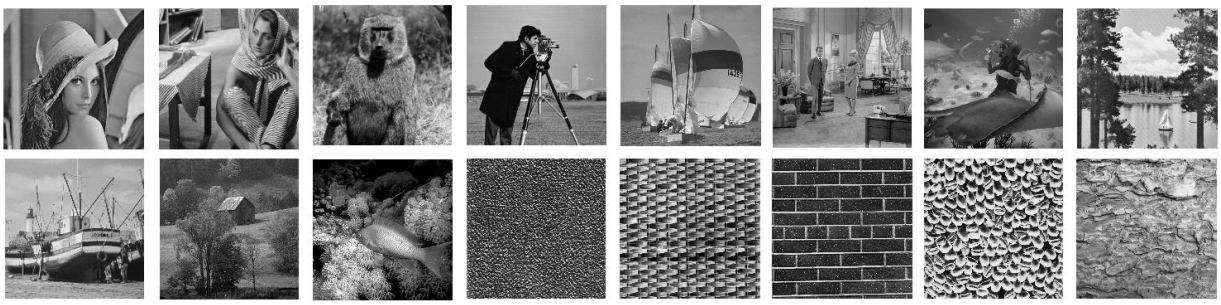
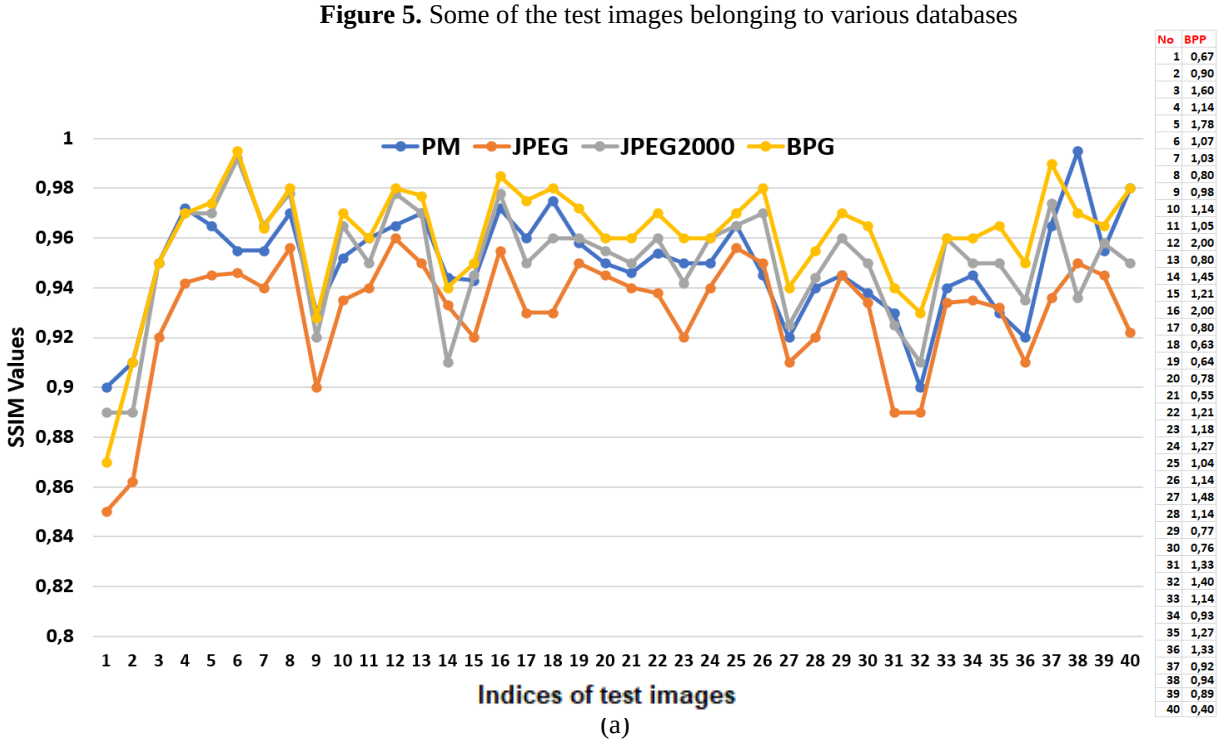
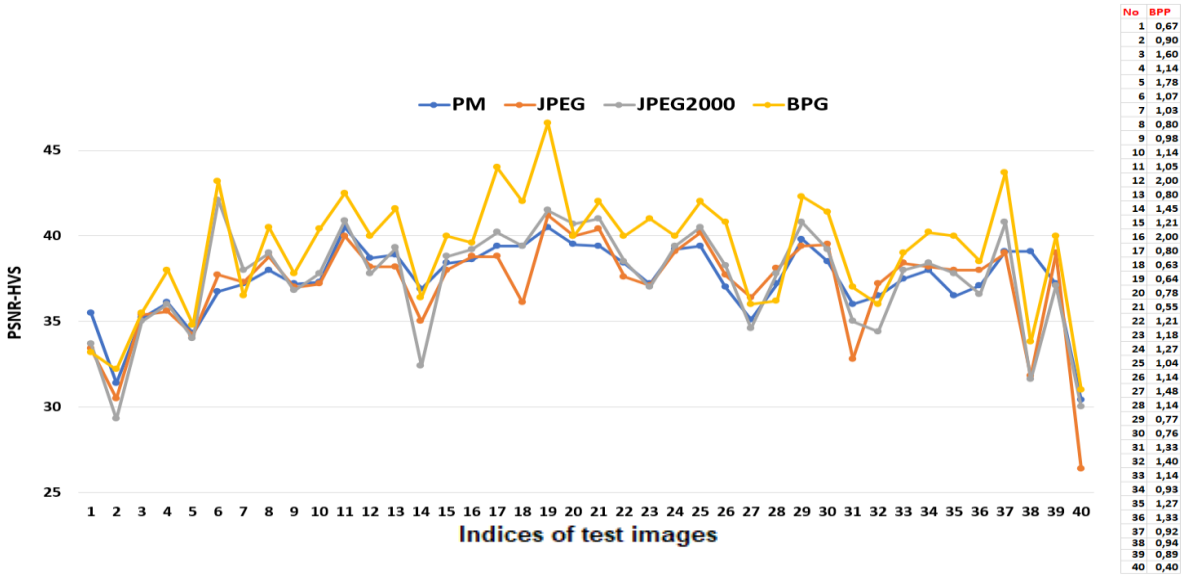


Figure 5. Some of the test images belonging to various databases



No	BPP
1	0,67
2	0,90
3	1,60
4	1,14
5	1,78
6	1,07
7	1,03
8	0,80
9	0,98
10	1,14
11	1,05
12	2,00
13	0,80
14	1,45
15	1,21
16	2,00
17	0,80
18	0,63
19	0,64
20	0,78
21	0,55
22	1,21
23	1,18
24	1,27
25	1,04
26	1,14
27	1,48
28	1,14
29	0,77
30	0,76
31	1,33
32	1,40
33	1,14
34	0,93
35	1,27
36	1,33
37	0,92
38	0,94
39	0,89
40	0,40

(a)

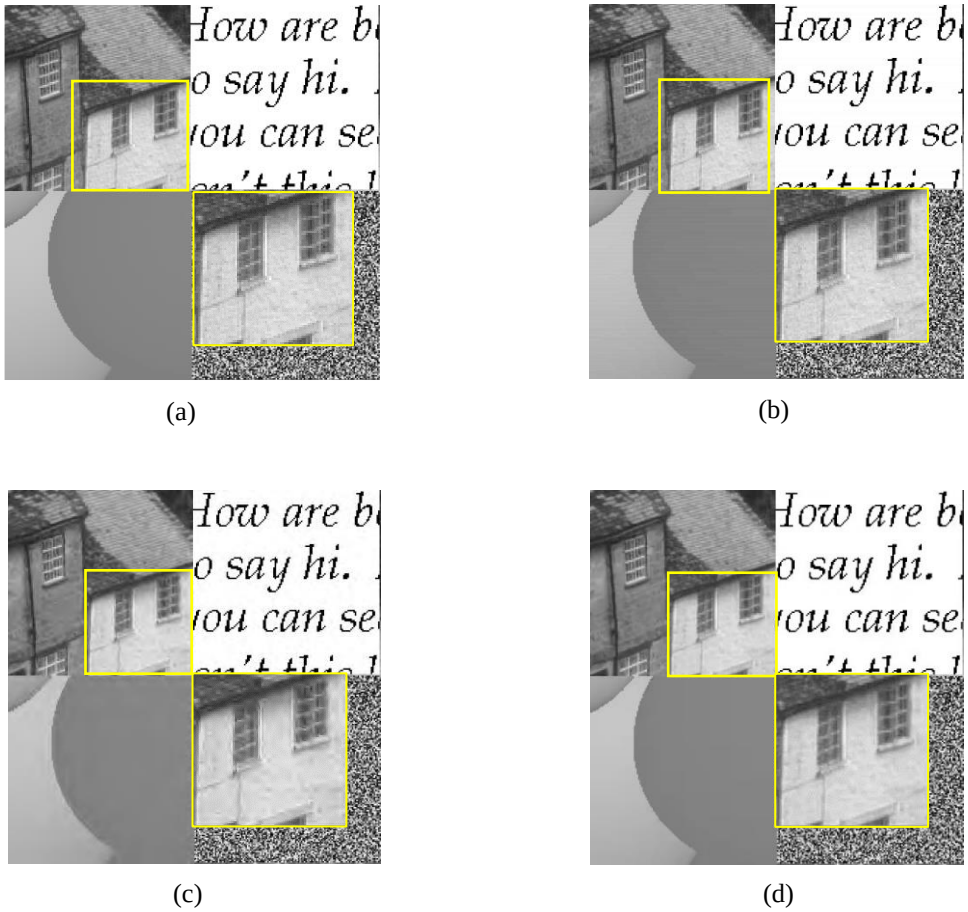


(b)

Figure 6. SSIM (a) and PSNR-HVS (b) values for PM, JPEG, JPEG2000, and BPG.

Table 2. The average SSIM and PSNR-HVS values of the methods

SSIM				PSNR-HVS			
PM	**JPEG**	**JPEG2K**	**BPG**	**PM**	**JPEG**	**JPEG2K**	**BPG**
0.950	0.930	0.952	0.962	37.44	37.16	37.45	39.14



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

Figure 7. The original image (a), PM (b) (SSIM: 0.99, PSNR:40, BPP: 0.94), JPEG2000 (c) (SSIM: 0.93, PSNR:29.2, BPP: 0.94), and BPG (d) (SSIM: 0.97, PSNR:33.8, BPP: 0.94)

Statement of Research and Publication Ethics

The study is complied with research and publication ethics

References

[1] Bhatt, A., Ashutosh, K. B., and Bhatt, K., "Image compression algorithms under JPEG with lapped orthogonal transform and discrete cosine transformation.", *International Journal of Engineering Research and Development, 7*(3): 6-10, 2013.

[2] Katharotiya, A., Patel, S., and Goyani, M., "Comparative Analysis between DCT & DWT Techniques of Image Compression.", *Journal of Information Engineering and Applications, 1*(2): 9-17, , 2011.

[3] Effros, M., and Chou, P. A., "Weighted universal transform coding: Universal image compression with the Karhunen-Loeve transform." In Proceedings., International Conference on Image Processing, Washington, USA, 61-64, 1995.

[4] Ahmed, N., Natarajan, T., and Rao, K. R., "Discrete cosine transform." *IEEE transactions on Computers*, 100(1): 90-93, 1974.

[5] Roy A.B., Dey D., Mohanty B. and Banerjee D., "Comparison of FFT, DCT, DWT, WHT compression techniques on electrocardiogram and photo plethysmography signals." *Special Issue of International Journal of Computer Applications* (Inte. Conf. on Computing, Communication and Sensor Network, CCSN'2012), 6-11, 2012.

[6] OH, T. H., and Besar, R., "Medical image compression using JPEG-2000 and JPEG: A comparison study". *Journal of Mechanics in Medicine and Biology*, 2(03n04), 313-328, 2002.

[7] Bharath, K. N., and Padmajadevi, G., "Hybrid compression using dwt-dct and huffman encoding techniques for biomedical image and video applications". *International Journal of Computer Science and Mobile Computing*, 2(5), 255-261, 2013.

[8] Muthukumaran, N., and Ravi, R., "The performances analysis of fast efficient lossless satellite image compression and decompression for wavelet based algorithm". *Wireless Personal Communications*, 81(2), 839-859, 2015.

[9] Ghrare, S.E., and Khobaiz, A.R. "Digital image compression using block truncation coding and Walsh Hadamard transform hybrid technique." *International Conference on Computer, Communications, and Control Technology* (I4CT), 477-480, 2014.

[10] Xie, Y., Jing, X., Sun, S., and Hong, L., "A fast and low complicated image compression algorithm for predictor of JPEG-LS.", *IEEE International Conference on Network Infrastructure and Digital Content*, Beijing, China, 353-356, 2009.

[11] Skodras, A., Christopoulos, C., Ebrahimi, T., "The JPEG 2000 still image compression standard.", *IEEE Signal processing magazine*, 18(5): 36-58, 2001.

[12] Sun, C., Li, Q., and Liu, J., "The study of Digital Image Compression based on wavelets", *International Conference on Audio, Language and Image Processing*, Shanghai, China, 312-316, 2010.

[13] M. A. Engin and B. Cavusoglu, "New Approach in Image Compression: 3D Spiral JPEG," in *IEEE Communications Letters*, vol. 15, no. 11, pp. 1234-1236, 2011.

[14] Ince, I. F., Bulut, F., Kilic, I., Yildirim, M. E., & Ince, O. F. "Low dynamic range discrete cosine transform (LDR-DCT) for high-performance JPEG image compression". *The Visual Computer*, *38*(5), 1845-1870, 2022.

[15] Shinde, A. A., and Kanjalkar, P., "The comparison of different transform based methods for ECG data compression.", *International Conference on Signal Processing, Communication, Computing and Networking Technologies*, 332-335, 2011.

[16] Telagarapu, P., Naveen, V. J., Prasanthi, A. L., and Santhi, G. V., "Image compression using DCT and wavelet transformations.", *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 4(3): 61-74, 2011.

[17] Hartung, F., and Girod, B., "Watermarking of uncompressed and compressed video". *Signal processing*, 66(3): 283-301, 1998.

[18] Hofbauer, H., Rathgeb, C., Wagner, J., Uhl, A., and Busch, C., "Investigation of Better Portable Graphics Compression for Iris Biometric Recognition," *2015 International Conference of the Biometrics Special Interest Group (BIOSIG)*, Darmstadt, Germany, 1-4, 2015.

[19] F. Bellard. BPG Image format. http://http://bellard.org/bpg . Access date: 10.06.2019.

[20] YEE, David, et al. "Medical image compression based on region of interest using better portable graphics (BPG).", *IEEE international conference on systems, man, and cybernetics (SMC)*, Banff, AB, Canada, 216-221, 2017.

[21] Sullivan, G. J., Ohm, J. R., Han, W. J., and Wiegand, T., "Overview of the high efficiency video coding (HEVC) standard". *IEEE Transactions on circuits and systems for video technology*, 22(12):1649-1668, 2012.

[22] Chen, Y., Murherjee, D., Han, J., Grange, A., Xu, Y., Liu, Z., and Chiang, C. H., "An overview of core coding tools in the AV1 video codec". *In Picture Coding Symposium (PCS), IEEE*, 41-45, 2018.

[23] A. M. Bruckstein, M. Elad, and R. Kimmel, "Down-scaling for better transform compression," *IEEE Trans. Image Process*., 12(9): 1132–1144,2003.

[24] C. Y. Wang et al., "JPEG-based image coding algorithm at low bit rates with down-sampling and interpolation," in *4th Int. Conf. On Wireless Communications, Networking and Mobile Computing (WiCOM'08)*, pp. 1–5, IEEE, Dalian, 1-5, 2008.

[25] Y. B. Zhang et al., "Interpolation-dependent image downsampling,"*IEEE Trans. Image Process*. , 20(11): 3291–3296, 2011.

[26] R. Pournaghi, X. L. Wu, and X. M. Liu, "Low bit-rate image coding via local random down-sampling," in Picture Coding Symp. (PCS), *IEEE, San Jose, California*, 329–332, 2013.
[27] Chen, H., He, X., Ma, M., Qing, L., and Teng, Q., "Low bit rates image compression via adaptive block downsampling and super resolution". *Journal of Electronic Imaging*, 25(1): 013004, 2016.
[28] W. S. Lin and L. Dong, "Adaptive downsampling to improve image compression at low bit rates," *IEEE Trans. Image Process*, 15(9): 2513–2521, 2006.
[29] V. A. Nguyen, Y. P. Tan, and W. S. Lin, "Adaptive downsampling/upsampling for better video compression at low bit rate," in *IEEE Int. Symp. On Circuits and Systems*, 1624–1627, 2008.
[30] Báscones, D., González, C., and Mozos, D., "Hyperspectral image compression using vector quantization, PCA and JPEG2000". *Remote sensing*, 10(6), 907, 2018.
[31] Wang, C. W., and Jeng, J. H., "Image compression using PCA with clustering". *In 2012 International Symposium on Intelligent Signal Processing and Communications Systems IEEE*, 458-462, 2012.
[32] Liu, Y. R., and Kau, L. J., "Scalable face image compression based on Principal Component Analysis and arithmetic Coding". *In 2017 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW), IEEE*, 265-266, 2017.
[33] Vaish, A., and Kumar, M., "WDR coding based image compression technique using PCA". *In 2015 International Conference on Signal Processing and Communication (ICSC), IEEE*,360-365, 2015.
[34] Rafael do Esprito Santo, "Principal Component Analysis applied to digital image compression", *Hospital Israelita Albert Einstein HIAE,* Sao Paulo (SP), Brazil, 2012.
[35] Cheng, Z., Sun, H., Takeuchi, M., and Katto, J., "Deep convolutional autoencoder-based lossy image compression". *In 2018 Picture Coding Symposium (PCS)*, pp. 253-257, IEEE, 2018.
[36] BULUT, F., "Huffman Algoritmasıyla Kayıpsız Hızlı Metin Sıkıştırma". *El-Cezeri Journal of Science and Engineering*, 3(2), 2016.
[37] Egiazarian, K., Astola, J., Ponomarenko, N., Lukin, V., Battisti, F., and Carli, M., "New full-reference quality metrics based on HVS." *In Proceedings of the Second International Workshop on Video Processing and Quality Metrics*, (4) , 2006.
[38] Hore, A., and Ziou, D., "Image quality metrics: PSNR vs. SSIM.", *20th international conference on pattern recognition*, Istanbul, Turkey, 2366-2369, 2010.
[39] Marcellin, M. W., Gormish, M. J., Bilgin, A., and Boliek, M. P. "An overview of JPEG-2000.", *In Proceedings DCC 2000. Data Compression Conference, Snowbird*, UT, USA, 523-541, 2000.
[40] Gonzalez, R. C. and Woods, R. E., *Digital Image Processing*, 3rd ed., Upper Saddle River, NJ, USA: Prentice-Hall,Inc., 2006.
[41] P. Brodatz, Textures: *A Photographic Album for Artists and Designers*. Dover Publications, 1966.
[42] STex, Salzburg texture image database (STex). Available online at <http://wavelab.at/sources/STex/>, 2009.
[43] Fei-fei, L., Fergus, R., and Perona, P., "One-shot learning of object categories". *IEEE Trans*. PAMI, 2006.