Edge and Fuzzy Transform Based Image Compression Algorithm: edgeFuzzy

Deepak Gambhir and Navin Rajpal

Abstract Since edges contain symbolically important image information, their detection can be exploited for the development of an efficient image compression algorithm. This paper proposes an edge based image compression algorithm in fuzzy transform (F-transform) domain. Input image blocks are classified either as low intensity blocks, medium intensity blocks or a high intensity blocks depending on the edge image obtained using the Canny edge detection algorithm. Based on the intensity values, these blocks are compressed using F-transform. Huffman coding is then performed on compressed blocks to achieve reduced bit rate. Subjective and objective evaluations of the proposed algorithm have been made in comparisons with RFVQ, FTR, FEQ and JPEG. Results show that the proposed algorithm is an efficient image compression algorithm and also possesses low time complexity.

1 Introduction

Problem of storage and demand of exchanging images over mobiles and internet have developed large interest of researchers in image compression algorithms. Especially high quality images with high compression ratio i.e. low bit rate is gaining advantage in various applications such as interactive TV, video conferencing, medical imaging, remote sensing etc. The main aim of image compression algorithm is to reduce the amount of data required to represent a digital image without any significant loss of visual information. This can be achieved by removing as much redundant and/or irrelevant information as possible from the image without degrading its visual quality. A number of image compression methods exists in literature. Joint photographic experts group (JPEG), JPEG2000, fuzzy based, neural networks based,

D. Gambhir (✉) · N. Rajpal
School of Information and Communication Technology, Guru Gobind Singh Inderprastha University, Dwarka, New Delhi, India
e-mail: gambhir.deepak@gmail.com

N. Rajpal
e-mail: navin_rajpal@yahoo.com

H. Lu and Y. Li (eds.), *Artificial Intelligence and Computer Vision*,
Studies in Computational Intelligence 672, DOI 10.1007/978-3-319-46245-5_8

optimization techniques are the commonly used image compression techniques [1–7]. However, fuzzy logic's ability to provide smooth approximate descriptions have attracted researchers towards fuzzy based image compression methods. Fuzzy transform, motivated from fuzzy logic and system modeling, introduced by Perfilieva, possesses an important property of preserving monotonicity [8] that can be utilized significantly to improve the quality of compressed image. F-transform transforms an original function into a finite number of real numbers (called F-transform components) using fuzzy sets in such a way that universal convergence holds true.
Motivation: With the ever increasing demand of low bandwidth applications in accessing internet, images are generally exchanged at low bit rates. JPEG based on DCT is the most popularly used image compression standard. But at low bit rates, JPEG produces compressed images that often suffer from significant degradation and artifacts. Martino et al. [9] proposed an image compression method based on F-transform (FTR) that performed better than fuzzy relation equations (FEQ) based image compression and similar to JPEG for small compression rate. Later Petrosino et al. [10] proposed rough fuzzy set vector quantization (RFVQ) method of image compression that performed better than JPEG and FTR. Since F-transform has an advantage of producing a simple and unique representation of original function that makes computations faster and also has an advantage of preserving monotonicity that results in an improved quality of reconstructed compressed image, hence this paper proposes edge based image compression algorithm in F-transform domain named edgeFuzzy. The encoding of the proposed algorithm consists of following three steps:

1. **Edge detection using Canny algorithm**: In this step, each input image block is classified into either a low intensity (LI), a medium intensity (MI) or a high intensity (HI) block using canny edge detection algorithm.
2. **Intensity based compression using the fuzzy transform (F-transform)**: The blocks classified into LI, MI and HI blocks are compressed using the F-transform algorithm.
3. **Huffman coding**: The intensity based F-transform compressed image data is further encoded using Huffman coding technique to achieve low bit rate.

Contribution: It is well known that edges provide meaningful information present in an image. Thus, an image compression algorithm that exploits edge information is proposed. Input image blocks based on the number of edge pixels detected using canny edge detection algorithm are classified as either as LI, MI or HI blocks. Since LI blocks carry less information (because it contains less number of edge pixels) hence they can be compressed more as compared to MI and HI blocks using F-transform. Huffman coding is then performed on the compressed image, that further helps in reducing the achieved bit rate.The proposed algorithm produces a better visual quality of compressed image with well preserved edges. There is a significant improvement in the visual quality of compressed images obtained using the proposed algorithm as measured by PSNR over other state of art techniques as shown in Figs. 4, 5, 6, 7 and 8. The proposed algorithm also possess low time complexity as observed from Tables 1, 2, 3, 4, 5, 6, 7 and 8.

The rest of chapter is organized as follows: Literature review about the recent fuzzy based image compression algorithm is given in Sect. 2. F-transform based image compression is discussed in Sect. 3, and the proposed method is presented in Sect. 4. Results and discussions are provided in Sect. 5 and finally the conclusions are drawn in Sect. 6.

2 Literature Review

The process of image compression deals with the reduction of redundant and irrelevant information present in an image thereby reducing the storage space and time needed for its transmission over mobile and internet. Image compression techniques may either be lossless (reversible) or lossy (irreversible) [11]. In lossless compression techniques, compression is achieved by removing the information theoretic redundancies present in an image such that the compressed image is exactly identical to the original image without any loss of information. Run-length coding, entropy coding and dictionary coders are widely used methods for achieving lossless compression. Graphics interchange format (GIF), ZIP and JPEG-LS (based on predictive coding) are the standard lossless file formats. These techniques are widely used in medical imaging, computer aided design, video containing text etc. However, in lossy compression techniques, compression is achieved by permanently removing the psycho-visual redundancies contained in image such that the compressed image is not identical but only an approximation of the original image. Video conferencing and mobile applications are various applications using lossy image compression techniques. JPEG (based on DCT coding) is the most popularly used lossy image compression standard file format. Only lossy image compression techniques can lead to higher value of compression ratio.

Apart from providing semantically important image information, edges play an important role in image processing and computer vision. Edges contain meaningful data, hence their detection can be exploited for image compression. The main aim of edge detection algorithms is to significantly reduce the amount of data needed to represent an image, while simultaneously preserving the important structural properties of object boundaries. Du [12] proposed two algorithms for edge base image compression. First algorithm is based on transmission of SPIHT bit stream at encoder and detection of edge pixels in the reconstructed image whereas second algorithm is based on detection of edges at the encoder and their extraction followed by combination at the decoder. The clarity of edges is further improved by using edge enhancement algorithm. Desai et al. [13] proposed an edge based compression scheme by extracting edge and mean information for very low bit rate applications. Mertins [14] proposed an image compression method based on edge based wavelet transform. Edges are detected and coded as secondary information. Wavelet transform is performed in such a way that the previously detected edges are not filtered out and hence are successfully preserved in reconstructed image. Avramovic [15] presents a lossless image compression algorithm based on edge detection and local gradient.

The algorithm combines the important features of median edge detector and gradient adjusted predictor. In past few years, many edge detection image compression algorithm using fuzzy logic have been developed that are more robust and flexible compared to the classical approaches. Gambhir et al. [16] proposed adaptive quantization coding based image compression algorithm. The algorithm uses fuzzy edge detector for based on entropy optimization for detecting edge pixels and modified adaptive quantization coding for compression and decompression. Petrosino et al. [10] proposed a new image compression algorithm named as rough fuzzy vector quantization (RFVQ). This method is based on the characteristics of rough fuzzy sets. Encoding is performed by exploiting the quantization capabilities of fuzzy vectors and decoding uses specific properties of these sets to reconstruct the original image blocks. Amarunnishad et al. [17] proposed Yager Involutive Fuzzy Complement Edge Operator (YIFCEO) based block truncation coding algorithm for image compression. The method uses fuzzy LBB (Logical Bit Block) for encoding the input image along with statistical parameters like mean and the standard deviation. Gambhir et al. [18] proposed an image compression algorithm based on fuzzy edge classifier and fuzzy transform. Fuzzy classifier first classifies input image blocks into either a smooth or an edge block. These blocks are further compressed and decompressed using fuzzy transform (F-transform). The algorithm relies on automatic detection of edges in images to be compressed using fuzzy classifier. Here edge detection parameters once set for an image is assumed to be working with other images. Further, encoding a block to single mean value results in loss of information. Gambhir et al. [19] also proposed an algorithm named pairFuzzy that classifies blocks using competitive fuzzy edge detection algorithm and also reduces artifact using fuzzy switched median filter.

3 Fuzzy Transform Based Image Compression

Perfilieva proposed F-transform based image compression algorithm in [8, 20, 21] and compared its performance with the performance of JPEG and fuzzy relation equations (FEQ). Fuzzy transform converts a discrete function on the closed interval $[A, B]$ to a set of M finite real numbers called components of F-transform, using basis functions that forms the fuzzy partition of $[A, B]$. An inverse F-transform assigns a discrete function to these components, that approximates the original function up to a small quantity ϵ.

Fuzzy partition of the Universe:

Consider M $(M \geq 2)$ number of fixed nodes, $x_1 \leq x_2 \leq x_3 \cdots \leq x_M$, in a closed interval $[A, B]$ such that $x_1 = A$ and $x_M = B$. The fuzzy sets $[A_1, A_2, \ldots A_M]$ identified with their membership functions $[A_1(x), A_2(x), \ldots A_M(x)]$ defined on $[A, B]$ forms the fuzzy partition of the universe, if the following conditions hold true for $k = 1, 2, 3 \ldots M$.

1. $A_k(x)$ is a continuous function over the interval $[A, B]$.
2. $A_k(x_k) = 1$ and $A_k(x) = 0$ if $x \notin (x_{k-1}, x_{k+1})$.
3. $A_k : [A, B] \rightarrow [0, 1]$ and $\sum_{k=1}^{M} A_k(x) = 1$ for all x.
4. $A_k(x)$ increases monotonically on $[x_{k-1}, x_k]$ and decreases monotonically on $[x_k, x_{k+1}]$.

For equidistant set of M points, $[A_1, A_2, \ldots A_M]$ forms a uniform fuzzy partition if the following additional conditions are satisfied for all x and k=2, … M−1(M≥2):

a. $A_k(x_k - x) = A_k(x_k + x)$,
b. $A_{k+1}(x) = A_k(x - \delta)$ where $\delta = (x_M - x_1)/(M - 1)$.

3.1 Discrete Fuzzy Transform for Two Variables

Consider $(M + N)$ fixed nodes (where $M, N \geq 2$), $x_1, x_2, x_3, \ldots x_M$ and $y_1, y_2, y_3, \ldots y_N$ of a two dimensional function, $f(x, y)$ on closed interval $[A, B] \times [C, D]$ such that $x_1 = A$, $x_M = B$, $y_1 = C$ and $y_N = D$. Let $[A_1, A_2, A_3, \ldots A_M]$ be the fuzzy partition of $[A, B]$ identified with their membership functions $[A_1(x), A_2(x), \ldots A_M(x)]$ such that $A_i(x) > 0$ for $[i = 1, 2, 3, \ldots M]$ and $[B_1, B_2, B_3, \ldots B_N]$ be the fuzzy partition of $[C, D]$ identified with their membership functions $[B_1(y), B_2(y), \ldots B_N(y)]$ such that $B_j(y) > 0$ for $[j = 1, 2, 3 \ldots N]$. The discrete fuzzy-transform of the function $f(x, y)$ is then defined as:

$$F_{k,l} = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} f(x_i, y_j) A_k(x_i) B_l(y_j)}{\sum_{i=1}^{M} \sum_{j=1}^{N} A_k(x_i) B_l(y_j)} \tag{1}$$

for $k = 1, 2, 3, \ldots m$ and $l = 1, 2, 3, \ldots n$.

And the inverse discrete fuzzy transform of F with respect to $\{A_1, A_2, \ldots A_M\}$ and $\{B_1, B_2, \ldots B_N\}$ is defined as:

$$f_{FN}(i, j) = \sum_{k=1}^{m} \sum_{l=1}^{n} F_{k,l} A_k(x_i) B_l(y_j) \tag{2}$$

for $i = 1, 2, 3 \ldots M$ and $j = 1, 2, 3, \ldots N$.

Let $f(x, y)$ be an image with M rows and N columns. The discrete F-transform compresses this image $f(x, y)$ into F-components $F_{k,l}$ using the Eq. (1) for $k = 1, 2, \ldots m$ and $l = 1, 2, \ldots n$. The compressed image $f_{FN}(i, j)$ can be reconstructed using related inverse F-transform using Eq. (2) for $i = 1, 2, \ldots M$ and $j = 1, 2, \ldots N$. Perfilieva and Martino [9, 20] proposed a method of lossy image compression and its reconstruction based on discrete F-transform.

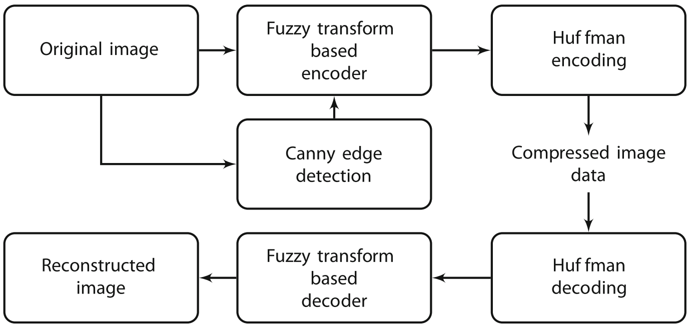


Fig. 1 Proposed method

4 Proposed Method

The proposed image compression method follows three steps:

1. Edge detection using Canny algorithm
2. Intensity based compression and decompression using F-transform
3. Huffman coding and decoding.

Figure 1 shows the block diagram for the proposed algorithm. The next subsections give the details of each step.

4.1 Edge Detection Using Canny Algorithm

Edge detection is a method of determining sharp discontinuities contained in an image. These discontinuities are sudden changes in pixel intensity which characterize boundaries of objects in an image. Canny edge detection [22], proposed by John F. Canny in 1986, is one of the most popular method for detecting edges. The performance of Canny detector depends upon three parameters: the width of the Gaussian filter used for smoothening the image and the two thresholds used for hysteresis threshold. Large width of the Gaussian function decreases its sensitivity to noise but at the cost of increased localization error and also some loss of detail information present in an image. The upper threshold should be set too high and the lower threshold should be set too low. Setting too low value of upper threshold increases the number of undesirable and spurious edge fragments in the final edge image and setting too high value of lower threshold results in break up of noisy edges. In MATLAB, lower threshold is taken to be 40 % of the upper threshold, if only the value of upper threshold is specified.

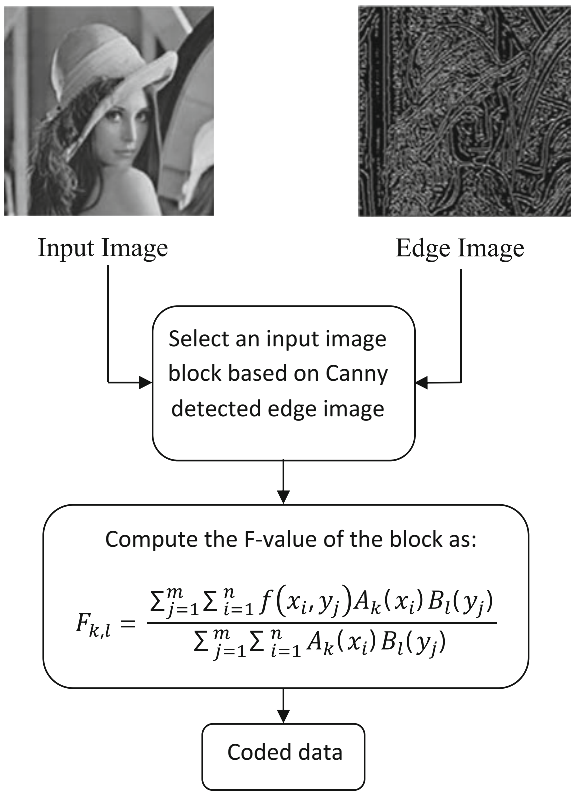

Input Image
Edge Image
Select an input image block based on Canny detected edge image
Compute the F-value of the block as:
$F_{k,l} = \frac{\sum_{j=1}^{m}\sum_{i=1}^{n} f(x_i, y_j) A_k(x_i) B_l(y_j)}{\sum_{j=1}^{m}\sum_{i=1}^{n} A_k(x_i) B_l(y_j)}$
Coded data

Fig. 2 Proposed fuzzy transform based encoder

4.2 Intensity Based Image Compression and Decompression Using F-Transform

Monotonicity of a function is an important property preserved by F-transform that helps in improving the quality of compressed (reconstructed) image. Input image is first divided into blocks of size $n \times n$. Based on the edge image obtained from Canny edge detection algorithm, the input image blocks are classified into LI blocks, MI blocks and HI blocks. A block with small number of edge pixels (less than T1) is classified as LI block, with high number of edge pixels (greater than T2) as HI blocks and rest (with edge pixels between T1 and T2) as MI blocks. These blocks are further compressed using F-transform into different size blocks. Since LI blocks contain less information (as it contains less edge pixels) hence can be compressed more as compared to HI blocks. For example: a LI $n \times n$ block is compressed to 3×3 block, a MI $n \times n$ block is compressed to 5×5 block and a HI $n \times n$ block is compressed to 7×7 block. These compressed blocks are further encoded using lossless Huffman encoding to achieve lower bit rate. Figure 2 shows the proposed encoder.

4.3 Introduction of Huffman Coding and Decoding

The compressed image is further encoded using Huffman coding scheme to achieve more compression. Huffman code is a popular method used for lossless data compression introduced by Huffman [23], is optimum in sense that this method of encoding results in shortest average length. This coding technique is also fast, easy to implement and conceptually simple.
Summary: The proposed algorithm is summarized as: The proposed algorithm edgeFuzzy, creates an edge image using Canny edge detection algorithm and classifies input image blocks as either LI, MI or HI blocks based on this edge image. Based on the intensity, these blocks are then compressed into different size blocks using the F-transform. These compressed blocks are encoded using Huffman coding that further reduces the bit rate.The proposed algorithm can produce different bit rates depending on the number of edge pixels detected by the Canny algorithm. Too many edge pixels detected by the Canny algorithm will result in low compression and hence high bit rate. Thus the bit rate achieved by the algorithm is sensitive to the edge detection algorithm. Since Huffman coding is a lossless compression technique, therefore its utilization can further reduce the bit rate without any loss of visual information at the cost of minutely increased time complexity.

5 Results and Discussions

To reduce the storage space, bandwidth and time for uploading and downloading from internet and mobile, this paper proposes an edge based image compression algorithm in F-transform domain. The proposed algorithm exploits edge information extracted using Canny algorithm for compressing an image. Original images (row 1), Canny edge detected images with threshold T = 0.005 and width $\sigma = 1$ (row 2) and Canny edge detected images with threshold T = 0.2 and width $\sigma = 1$ is shown in Fig. 3. It is also observed that an increase in the value of threshold decreases the number of detected edge pixels. To evaluate the performance of the proposed algorithm, it has been tested on eight set of test images: Lenna, Bridge, House, Cameraman, Goldhill, peppers, Airplane and Lake of size 256×256 as well as on eight different set of test images: Tank, Straw, Aerial, Boat, Elaine, Lake, Pentagon and Wall of size 512×512. These set of images are downloaded from SIPI image database [24]. The process of compression is done on the PC with 4 GB RAM, Intel core i7 @ 2.50 GHZ with windows 8.1, 64 bit operating system using MATLAB 8.2, R2013b. The bit rate achieved using the proposed algorithm without lossless Huffman coding (i.e. bpp), using the proposed algorithm with lossless Huffman coding (i.e. bpp_H) and using the proposed algorithm with lossless arithmetic coding (i.e. bpp_A) in place of lossless Huffman coding is summarizes in Table 1 and Table 2 for images of size 256×256. These results are obtained by dividing input images of size 256×256 into blocks of size 16×16 and size 8×8 respectively and reducing

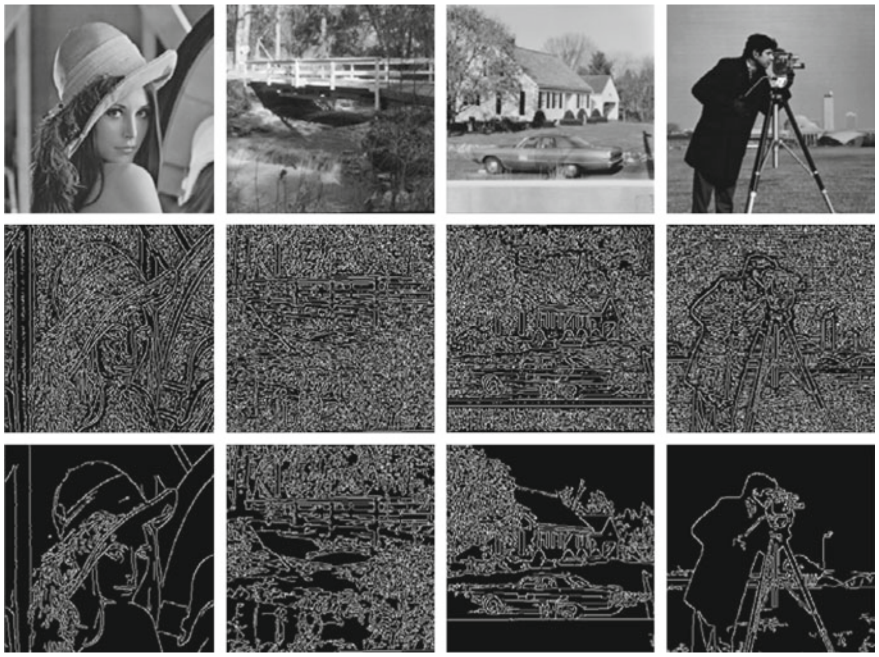


Fig. 3 *row 1*: Original images, Lenna, Bridge, House, Cameraman; Canny edge detected images at *row 2*: T = 0.005, *row 3*: T = 0.2 for the $\sigma = 0.5$

to 7×7 for HI block, 5×5 for MI block and 3×3 for LI block. The number of LI blocks, MI blocks and HI blocks for different images and different value of thresholds T1 and T2 is also shown in tables. It is also observed that small values of T1 and T2 increases the number of MI and HI blocks, this results in high bit-rate i.e. reduced compression. From these tables it is also observed that with the proposed algorithm different bit rate is achieved for different images at same values of T1 and T2. This is because compression using proposed method depends on the number of LI, MI and HI blocks and these number of blocks depends on the detected edge pixels. These number of detected edge pixels in turn depends on the parameters of Canny detection algorithm as well as on the nature of original image that is to be compressed. It is observed in Table 1 that the proposed algorithm achieved bit rate ranging from 0.032 bpp to 0.097 bpp approximately, while compressing original images of size 256×256 with the block size of 16×16. It is observed in Table 2 that the proposed algorithm achieved bit rate ranging from 0.118 bpp to 0.409 bpp approximately, while compressing original images of size 256×256 with the block size of 8×8. This increase in bit rate results in improvement of the visual image quality of the reconstructed image. Visual results of proposed algorithm for achieving compression of images of size 256×256 for blocks of size 16×16 (row 1–row 3) and for blocks of size 8×8 (row 4–row 6) is shown in Figs. 4 and 5 respectively.

Table 1 Proposed algorithm bit rate for T = 0.005 and $\sigma = 1$ for block size 16×16 (Image Size 256×256)

Images	T1	T2	LI	MI	HI	bpp	bpp_H	bpp_A
Lenna	0.40	0.50	256	000	000	0.035	0.032	0.028
	0.30	0.40	208	048	000	0.047	0.043	0.038
	0.20	0.35	070	181	005	0.082	0.075	0.069
Bridge	0.40	0.50	255	001	000	0.035	0.033	0.030
	0.30	0.40	144	111	001	0.063	0.058	0.053
	0.20	0.35	010	225	021	0.103	0.097	0.089
House	0.40	0.50	256	000	000	0.035	0.031	0.026
	0.30	0.40	183	073	000	0.053	0.047	0.042
	0.20	0.35	056	187	013	0.089	0.079	0.071
Cameraman	0.40	0.50	254	002	000	0.036	0.031	0.031
	0.30	0.40	136	118	002	0.065	0.056	0.054
	0.20	0.35	016	215	025	0.103	0.090	0.085
Goldhill	0.40	0.50	256	000	000	0.035	0.032	0.030
	0.30	0.40	169	087	000	0.056	0.050	0.046
	0.20	0.35	016	223	017	0.100	0.091	0.086
Peppers	0.40	0.50	256	000	000	0.035	0.033	0.029
	0.30	0.40	246	010	000	0.038	0.035	0.030
	0.20	0.35	098	157	001	0.074	0.070	0.062
Airplane	0.40	0.50	255	001	000	0.035	0.028	0.025
	0.30	0.40	195	060	001	0.050	0.039	0.035
	0.20	0.35	012	226	018	0.101	0.080	0.072
Lake	0.40	0.50	256	000	000	0.035	0.033	0.032
	0.30	0.40	221	035	000	0.044	0.040	0.039
	0.20	0.35	043	209	004	0.089	0.082	0.081

Alongwith the subjective evaluation, the proposed algorithm is also objectively evaluated using various quality measures such as: PSNR, RMSE and SAD. RMSE (root mean square error) [25] measures the square root of the cumulative squarer error between the original image and the compressed image. Mathematically

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{M}[I(i,j) - C(i.j)]^2}{M \times N}} \tag{3}$$

where $M \times N$ is total number of pixels in an image. $I(i,j)$ and $C(i,j)$ are the intensity values of original and compressed images at location (i,j) respectively.

PSNR (peak signal to noise ratio) calculates the peak signal-to-noise ratio, in dB between two images and is defined as

Table 2 Proposed algorithm bit rate for T = 0.005 and $\sigma = 1$ for block size 8×8 (Image Size 256×256)

Images	T1	T2	LI	MI	HI	bpp	bpp_H	bpp_A
Lenna	0.40	0.50	1003	021	000	0.146	0.134	0.121
	0.30	0.40	813	190	021	0.200	0.185	0.170
	0.20	0.35	305	639	080	0.345	0.320	0.281
Bridge	0.40	0.50	970	053	001	0.154	0.146	0.129
	0.30	0.40	587	383	054	0.267	0.251	0.220
	0.20	0.35	116	718	190	0.432	0.409	0.369
House	0.40	0.50	978	046	000	0.152	0.136	0.121
	0.30	0.40	699	279	046	0.237	0.213	0.191
	0.20	0.35	246	635	143	0.383	0.345	0.317
Cameraman	0.40	0.50	959	065	000	0.156	0.137	0.120
	0.30	0.40	593	366	065	0.270	0.235	0.207
	0.20	0.35	118	685	221	0.443	0.387	0.341
Goldhill	0.40	0.50	983	040	001	0.151	0.139	0.122
	0.30	0.40	648	335	041	0.247	0.226	0.198
	0.20	0.35	147	735	142	0.407	0.374	0.352
Peppers	0.40	0.50	1016	008	000	0.143	0.135	0.118
	0.30	0.40	907	109	008	0.172	0.163	0.144
	0.20	0.35	402	595	027	0.302	0.287	0.253
Airplane	0.40	0.50	993	031	000	0.148	0.118	0.105
	0.30	0.40	672	321	031	0.238	0.184	0.165
	0.20	0.35	132	737	155	0.415	0.326	0.276
Lake	0.40	0.50	1010	014	000	0.144	0.134	0.121
	0.30	0.40	791	219	014	0.203	0.189	0.166
	0.20	0.35	256	681	087	0.360	0.336	0.298

$$PSNR = 20\ log_{10}\left(\frac{L}{RMSE}\right) \tag{4}$$

where L is the maximum possible value of intensity (for 8 bit image, $L = 255$).

SAD (sum of absolute difference) is used to measure the similarity between two images and is obtained using

$$SAD = \sum_{i=1}^{N}\sum_{j=1}^{M} |I(i,j) - C(i.j)| \tag{5}$$

Low value of RMSE, high value of PSNR and low value of SAD are generally desirable. Though these measures are most commonly used measures for objective analysis but these measures does not agree with human visual perception and hence SSIM and FSIM are also used for performance evaluation.

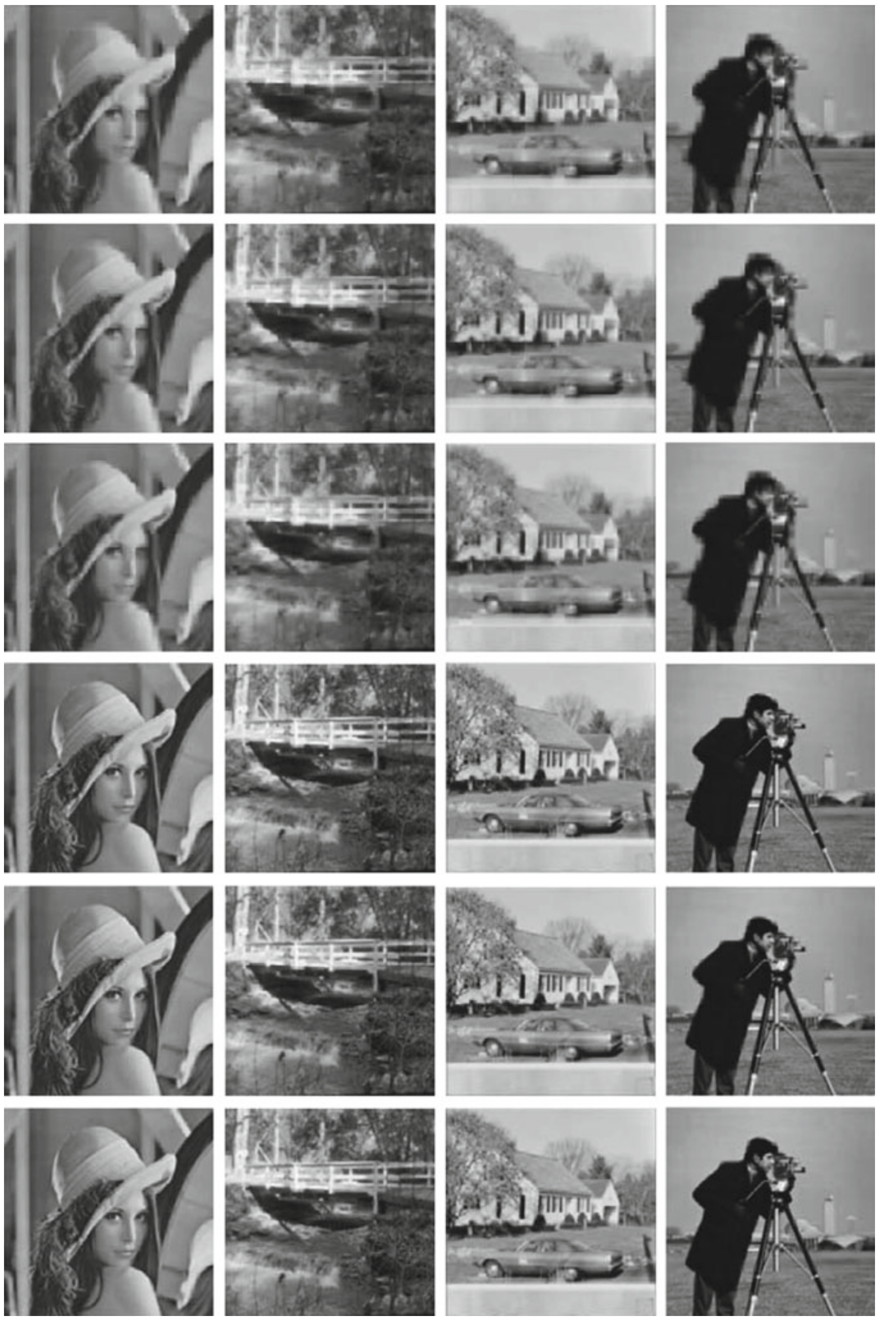


Fig. 4 Compressed images obtained using proposed algorithm (i) row 1 to row 3 for block size 16 × 16 and thresholds (0.4 and 0.5), (0.3 and 0.4) and (0.2 and 0.35) respectively and (ii) row 4 to row 6 for block size 8 × 8 and thresholds (0.4 and 0.5), (0.3 and 0.4) and (0.2 and 0.35) respectively for the test images of size 256 × 256

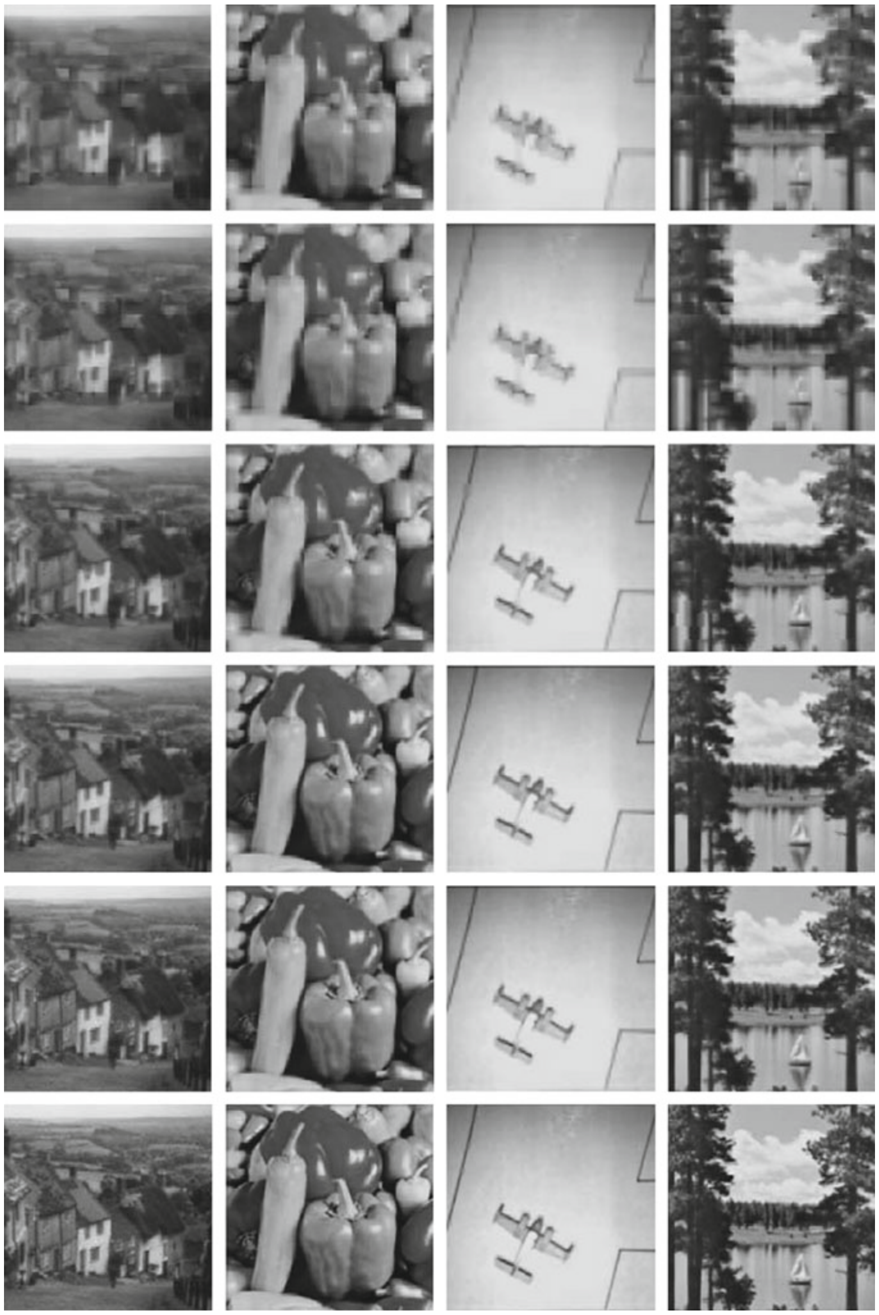


Fig. 5 Compressed images obtained using proposed algorithm (i) row 1 to row 3 for block size 16×16 and thresholds (0.4 and 0.5), (0.3 and 0.4) and (0.2 and 0.35) respectively and (ii) row 4 to row 6 for block size 8×8 and thresholds (0.4 and 0.5), (0.3 and 0.4) and (0.2 and 0.35) respectively for the test images of size 256×256

Table 3 Quality parameters obtained from the Proposed algorithm

Images	Code rate	PSNR (dB)	RMSE	SAD	MSSIM	FSIM
Lenna	0.032	24.40	15.36	728244	0.651	0.761
	0.043	24.78	14.70	695899	0.679	0.780
	0.075	25.91	12.91	576150	0.747	0.819
	0.134	27.87	10.31	425458	0.828	0.875
	0.185	28.82	9.24	378924	0.864	0.894
	0.320	30.26	7.82	281447	0.921	0.933
Bridge	0.033	21.46	21.55	1121689	0.390	0.614
	0.058	22.63	18.84	1053150	0.455	0.663
	0.097	23.57	16.90	885433	0.562	0.732
	0.146	24.63	14.96	790560	0.641	0.777
	0.251	25.71	13.33	676875	0.744	0.824
	0.409	26.96	11.44	516928	0.844	0890
House	0.031	22.49	19.14	931507	0.568	0.647
	0.047	24.82	14.64	880885	0.609	0.685
	0.079	25.11	14.15	756344	0.686	0.741
	0.136	25.92	12.89	611565	0.768	0.794
	0.213	27.43	10.84	532639	0.829	0.832
	0.345	29.71	8.33	403139	0.897	0.896
Cameraman	0.031	21.39	21.83	740882	0.667	0.701
	0.056	21.56	21.29	713780	0.690	0.719
	0.090	23.43	17.18	555940	0.767	0.779
	0.137	24.93	14.42	453574	0.830	0.857
	0.235	25.91	12.91	396270	0.870	0.857
	0.387	28.77	9.28	278423	0.929	0.914
Goldhill	0.032	23.10	17.84	835045	0.460	0.676
	0.050	23.52	16.91	787305	0.506	0.712
	0.091	24.89	14.53	662135	0.604	0.781
	0.139	25.84	13.03	588327	0.673	0.821
	0.226	26.76	11.70	521006	0.745	0.858
	0.374	28.47	9.61	416031	0.830	0.912
Peppers	0.033	22.05	20.13	821800	0.677	0.766
	0.035	22.09	20.04	817509	0.681	0.768
	0.070	23.44	17.16	668149	0.762	0.812
	0.135	26.25	12.41	450748	0.857	0.874
	0.163	26.68	11.81	427206	0.871	0.881
	0.287	28.65	9.42	323339	0.922	0.921

(continued)

Table 3 (continued)

Images	Code rate	PSNR (dB)	RMSE	SAD	MSSIM	FSIM
Airplane	0.028	25.44	13.63	353654	0.848	0.802
	0.039	25.46	13.59	349291	0.852	0.805
	0.080	27.27	11.03	271371	0.897	0.860
	0.118	28.23	9.88	241199	0.914	0.883
	0.184	28.48	9.60	227639	0.924	0.891
	0.326	30.15	7.93	173804	0.956	0.937
Lake	0.033	19.96	25.63	1108335	0.540	0.680
	0.040	20.12	25.21	1080084	0.555	0.689
	0.082	22.07	20.08	825933	0.685	0.764
	0.134	23.89	16.29	645148	0.772	0.827
	0.189	24.64	14.94	581306	0.815	0.851
	0.336	27.16	11.18	424595	0.894	0.910

SSIM (Structural similarity index measure) [26] measures the structural similarity between the two images and is calculated using:

$$SSIM = \frac{1}{W}\sum_{i=1}^{W}\left(\frac{2\mu_{i_i}\mu_{c_i} + (K_1L)^2}{\mu_{i_i}^2 + \mu_{c_i}^2 + (K_1L)^2}\right) \times \left(\frac{2\sigma_{i_ic_i} + (K_2L)^2}{\sigma_{i_i}^2 + \sigma_{c_i}^2 + (K_2L)^2}\right) \tag{6}$$

where μ_i, μ_c and σ_i, σ_c are mean intensities and standard deviations respectively, K_1 and K_2 are constants as, $0 < K_1, K_2 < 1$ and W is the number of local windows of the image. A large value of SSIM indicate the ability of algorithm to retain the original image.

The FSIM (feature similarity index measure) [27] measures the similarity between two images by computing locally the combination of the phase congruency (PC) [28] and gradient magnitude (GM) information using

$$FSIM = \frac{\sum_i \sum_j S(i,j) max\left\{PC_i(i,j), PC_c(i,j)\right\}}{\sum_i \sum_j max\left\{PC_i(i,j)PC_c(i,j)\right\}} \tag{7}$$

where

$$S(i,j) = \left(\frac{2PC_i(i,j)PC_c(i,j) + K_{PC}}{PC_i^2(i,j) + PC_c^2(i,j) + K_{PC}}\right) \times \left(\frac{2G_i(i,j)G_c(i,j) + K_{GM}}{G_i^2(i,j) + G_c^2(i,j) + K_{GM}}\right) \tag{8}$$

Table 4 Coding and decoding timing for test images (seconds)

Images	Code rate	Proposed coding time	Proposed decoding time	Code rate	FTR coding time	FTR decoding time	Code rate	JPEG coding time	JPEG decoding time
Lenna	0.032	3.52	1.58	0.035	4.10	14.82	0.034	4.13	3.31
	0.043	3.62	2.15	0.062	4.38	11.55	0.061	5.11	3.14
	0.134	3.79	2.40	0.140	5.46	14.93	0.130	5.87	3.61
	0.185	4.34	3.01	0.250	6.09	11.34	0.240	6.18	4.11
	0.320	4.95	3.73	0.391	6.21	11.72	0.439	6.24	4.28
Bridge	0.032	3.53	1.58	0.035	2.13	5.53	0.034	2.45	2.11
	0.043	3.61	2.13	0.062	1.19	4.19	0.058	6.78	2.36
	0.134	3.83	2.45	0.140	3.59	5.59	0.140	4.02	2.45
	0.185	4.29	3.11	0.250	2.32	4.32	0.244	4.65	2.68
	0.320	4.86	3.79	0.391	3.39	4.40	0.430	4.77	2.54
House	0.031	2.33	1.02	0.035	4.10	14.82	0.034	4.13	3.31
	0.047	3.35	1.54	0.062	4.38	11.55	0.062	5.11	3.14
	0.136	4.65	2.49	0.140	5.46	14.93	0.137	5.87	3.61
	0.213	3.53	1.68	0.250	6.09	11.34	0.240	6.18	4.11
	0.345	5.60	2.61	0.391	6.21	11.72	0.434	6.24	4.28
Cameraman	0.031	3.88	1.76	0.035	2.16	5.59	0.034	2.36	1.94
	0.056	5.66	2.74	0.062	1.20	4.25	0.061	5.54	2.09
	0.137	3.39	1.28	0.140	3.41	5.49	0.139	18.29	2.40
	0.255	4.08	1.86	0.250	2.11	4.31	0.249	3.07	2.68
	0.387	6.23	2.89	0.391	3.34	5.09	0.436	3.46	2.85

(continued)

Table 4 (continued)

Images	Code rate	Proposed coding time	Proposed decoding time	Code rate	FTR coding time	FTR decoding time	Code rate	JPEG coding time	JPEG decoding time
Goldhill	0.032	2.25	1.23	0.035	4.11	5.32	0.034	2.38	1.96
	0.050	3.63	1.95	0.062	4.68	4.44	0.061	5.56	3.12
	0.139	2.32	1.22	0.140	3.84	5.46	0.139	5.31	3.11
	0.226	3.76	1.93	0.250	5.24	4.51	0.249	3.16	2.32
	0.374	6.06	3.07	0.391	6.84	5.16	0.436	4.11	2.43
Peppers	0.033	2.31	1.23	0.035	2.16	5.59	0.034	4.33	1.94
	0.035	2.44	1.33	0.062	1.20	4.25	0.061	5.31	2.11
	0.135	2.12	1.15	0.140	3.41	5.49	0.139	7.03	3.10
	0.163	2.58	1.37	0.250	2.11	4.31	0.249	3.45	3.16
	0.287	4.55	2.28	0.391	3.34	5.09	0.436	4.15	2.50
Airplane	0.028	3.46	1.84	0.035	3.55	5.62	0.034	3.25	1.81
	0.039	4.77	2.62	0.062	4.96	5.25	0.061	4.15	2.11
	0.118	3.20	1.72	0.140	4.87	5.43	0.139	6.24	2.35
	0.184	5.21	2.77	0.250	6.69	4.28	0.249	3.25	2.64
	0.326	5.51	2.74	0.391	6.25	5.19	0.436	3.72	2.71
Lake	0.033	1.67	0.91	0.035	4.04	10.11	0.034	2.36	4.11
	0.040	2.35	1.21	0.062	3.03	9.82	0.061	5.54	4.15
	0.134	2.29	1.02	0.140	3.23	11.12	0.139	18.29	3.71
	0.189	2.62	1.41	0.250	3.45	14.18	0.249	3.07	4.59
	0.336	4.87	2.38	0.391	5.27	10.21	0.436	3.46	5.85

The corresponding values of these measures (PSNR, RMSE, SAD, SSIM and FSIM) for various images of size 256 × 256 at different code rates is shown in Table 3. It is also observed from the table, that low bit rate results degrade in quality of the reconstructed image. An idea about the time needed during coding and decoding images of size 256 × 256 using proposed algorithm, FTR and JPEG for achieving almost similar compression rates for various images when run on the same environment is given by Table 4. The proposed algorithm is much faster than its counterparts is also observed in Table 4.

The bit rate achieved using proposed algorithm (i.e. bpp_H) by dividing input images of size 512 × 512 into blocks of size 16 × 16 and size 8 × 8 is summarizes in Table 5 and Table 6 respectively. The results obtained using intensity based

Table 5 Proposed algorithm bit rate for T = 0.005 and $\sigma = 1$ for block size 16 × 16 (Image Size 512 × 512)

Images	T1	T2	LI	MI	HI	bpp	bpp_H	bpp_A
Elaine	0.40	0.50	1018	0006	0000	0.036	0.033	0.029
	0.30	0.40	0672	0346	0006	0.057	0.053	0.051
	0.20	0.35	0152	0755	0117	0.099	0.092	0.085
Boat	0.40	0.50	1022	0002	0000	0.035	0.031	0.026
	0.30	0.40	0728	0294	0002	0.053	0.046	0.042
	0.20	0.35	0096	0876	0054	0.097	0.085	0.081
Lake	0.40	0.50	1018	0006	0000	0.036	0.033	0.031
	0.30	0.40	0775	0243	0006	0.051	0.047	0.042
	0.20	0.35	0104	0840	0080	0.099	0.091	0.087
Straw	0.40	0.50	1022	0022	0000	0.036	0.030	0.028
	0.30	0.40	0207	0795	0022	0.087	0.073	0.069
	0.20	0.35	0000	0751	0273	0.123	0.104	0.095
Tank	0.40	0.50	1023	0001	0000	0.035	0.028	0.026
	0.30	0.40	0534	0489	0001	0.065	0.051	0.048
	0.20	0.35	0007	0969	0048	0.102	0.081	0.076
Aerial	0.40	0.50	1020	0004	0001	0.035	0.030	0.027
	0.30	0.40	0774	0246	0004	0.051	0.042	0.038
	0.20	0.35	0055	0926	0043	0.098	0.084	0.077
Wall	0.40	0.50	0990	0034	000	0.037	0.019	0.018
	0.30	0.40	0247	0743	0034	0.086	0.046	0.042
	0.20	0.35	0000	0672	0352	0.130	0.072	0.071
Pentagon	0.40	0.50	1024	0000	0000	0.035	0.028	0.027
	0.30	0.40	0773	0251	0000	0.050	0.040	0.036
	0.20	0.35	0062	0943	0019	0.096	0.077	0.074

Table 6 Proposed algorithm bit rate for T = 0.005 and $\sigma = 1$ for block size 8×8 (Image Size 512×512)

Images	T1	T2	LI	MI	HI	bpp	bpp_H	bpp_A
Elaine	0.40	0.50	3922	0174	0000	0.151	0.142	0.128
	0.30	0.40	2613	1309	0174	0.247	0.230	0.207
	0.20	0.35	0737	2717	0642	0.404	0.378	0.341
Lake	0.40	0.50	3930	0165	0001	0.151	0.140	0.126
	0.30	0.40	2797	1133	0166	0.235	0.216	0.195
	0.20	0.35	0826	2760	0600	0.395	0.366	0.330
Boat	0.40	0.50	3979	0117	0000	0.148	0.132	0.120
	0.30	0.40	2710	1269	0117	0.236	0.208	0.190
	0.20	0.35	0607	2951	0538	0.403	0.357	0.325
Straw	0.40	0.50	3646	0446	0004	0.168	0.145	0.131
	0.30	0.40	1394	2252	0450	0.347	0.304	0.274
	0.20	0.35	0066	2588	1442	0.519	0.458	0.413
Tank	0.40	0.50	3955	0141	000	0.149	0.120	0.117
	0.30	0.40	2247	1708	0141	0.266	0.214	0.197
	0.20	0.35	0189	3200	0707	0.444	0.359	0.340
Aerial	0.40	0.50	3954	0141	0001	0.149	0.128	0.117
	0.30	0.40	2794	1160	0142	0.233	0.201	0.185
	0.20	0.35	0617	2976	0512	0.400	0.347	0.320
Wall	0.40	0.50	3414	0665	0017	0.184	0.104	0.096
	0.30	0.40	1490	1924	0682	0.362	0.209	0.196
	0.20	0.35	0083	2394	1619	0.534	0.319	0.294
Pentagon	0.40	0.50	4008	0088	0000	0.146	0.119	0.105
	0.30	0.40	2866	1142	0088	0.224	0.181	0.179
	0.20	0.35	0564	3084	0448	0.397	0.325	0.324

F-transform compression (i.e. bpp) with edge detection algorithm, intensity based F-transform compression with lossless arithmetic coding (i.e. bpp_A) is also given in the table. Although it is observed in results that the arithmetic coding provides better compression as compared to Huffman coding, but since the presented algorithm supports achieving faster compression at superior quality. Thus, the Huffman code is chosen over the arithmetic code. This result is in line with [29], where it is clearly proved that the Huffman code is having higher performance than arithmetic coding.

Results of the proposed algorithm for achieving compression of images of size 512×512 is shown in Figs. 6 and 7. The corresponding values of these measures (PSNR, RMSE, SAD, SSIM and FSIM) for various images of size 512×512 at different code rates is shown in Table 7. An idea about the time needed during coding

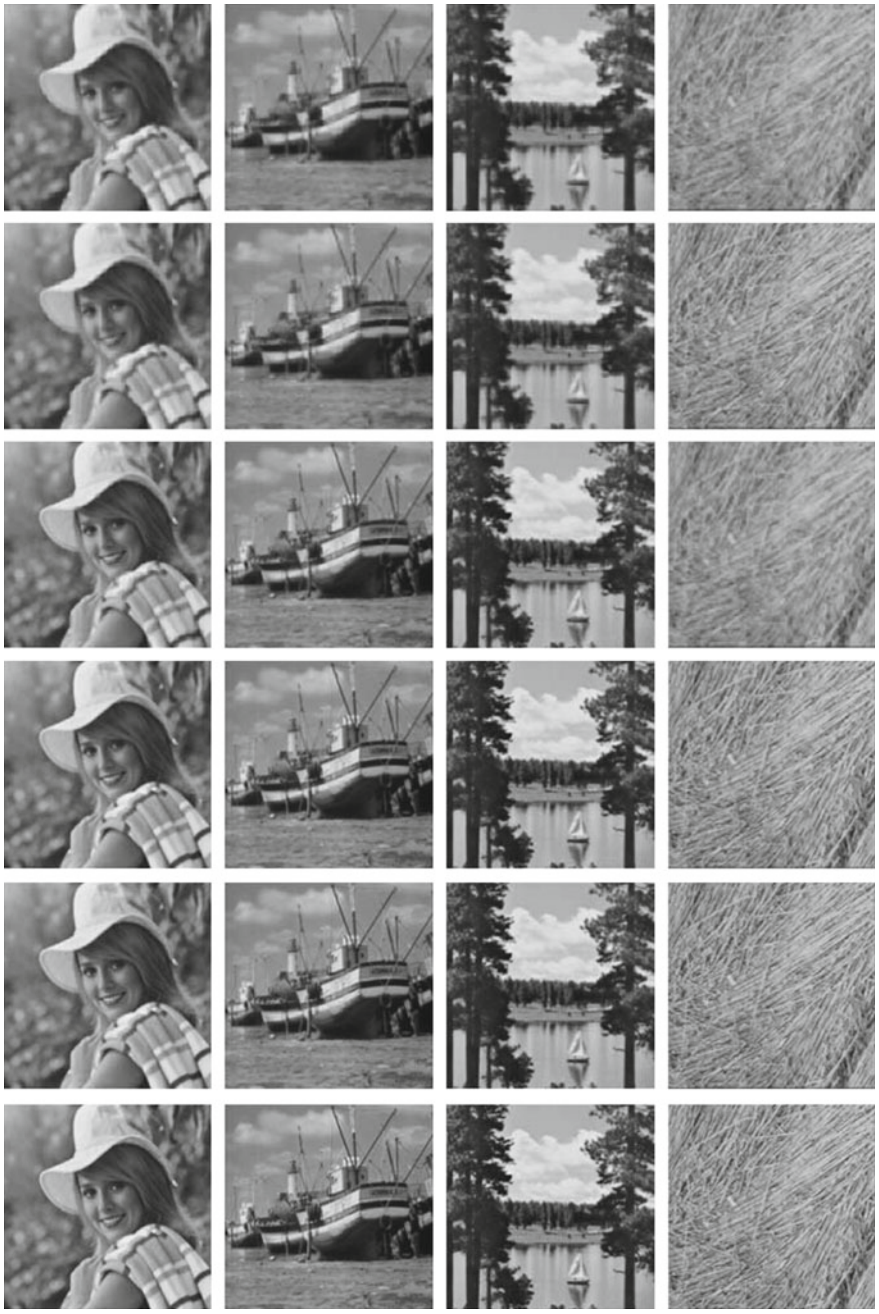


Fig. 6 Compressed images obtained using proposed algorithm (i) row 1 to row3 for block size 16×16 and thresholds (0.4 and 0.5), (0.3 and 0.4) and (0.2 and 0.35) respectively and (ii) row 4 to row 6 for block size 8×8 and thresholds (0.4 and 0.5), (0.3 and 0.4) and (0.2 and 0.35) respectively for the test images of size 512×512

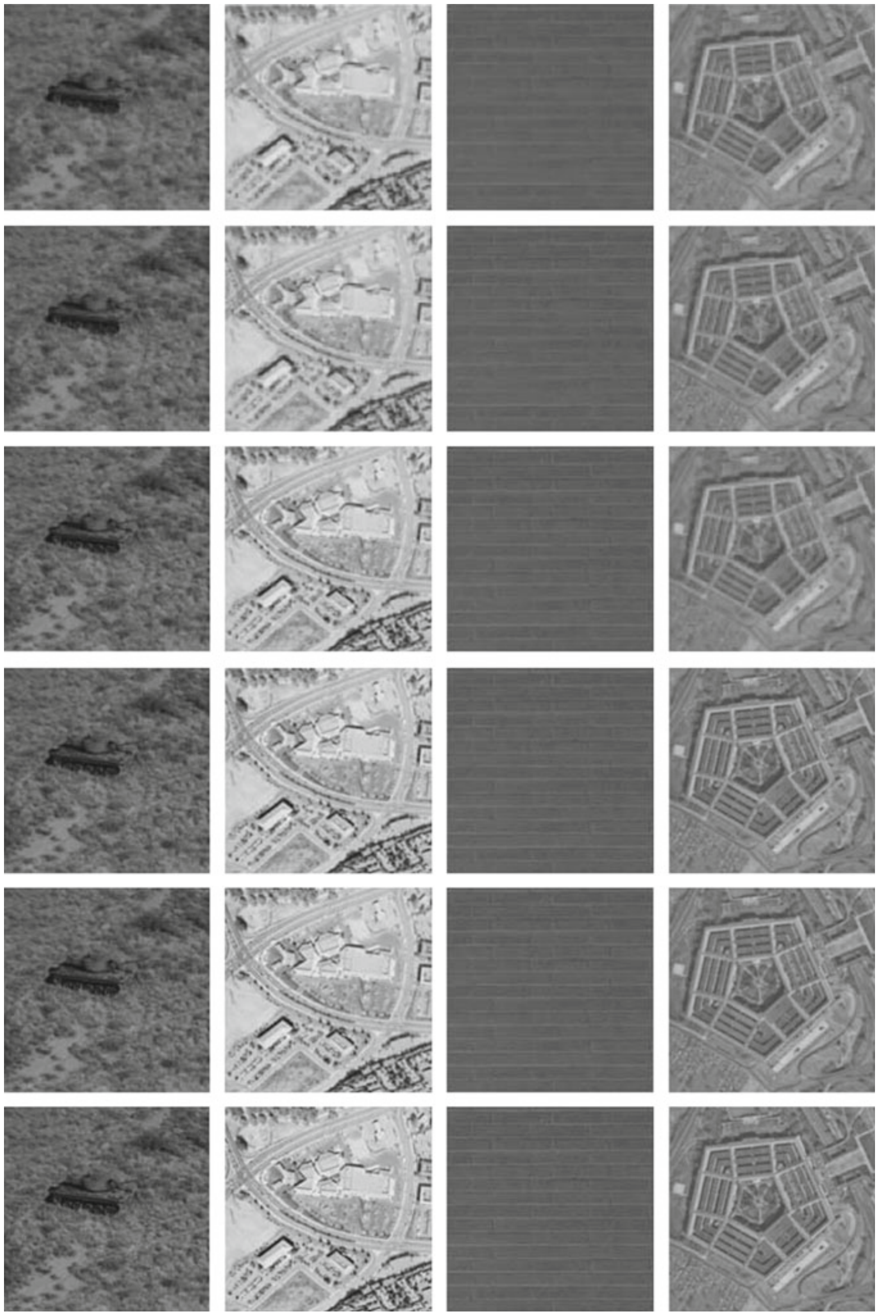


Fig. 7 Compressed images obtained using proposed algorithm (i) row 1 to row3 for block size 16×16 and thresholds (0.4 and 0.5), (0.3 and 0.4) and (0.2 and 0.35) respectively and (ii) row 4 to row 6 for block size 8×8 and thresholds (0.4 and 0.5), (0.3 and 0.4) and (0.2 and 0.35) respectively for the test images of size 512×512

Table 7 Quality parameters obtained from the Proposed algorithm

Images	Code rate	PSNR (dB)	RMSE	SAD	MSSIM	FSIM
Elaine	0.033	26.95	11.45	2010390	0.841	0.886
	0.053	27.10	11.25	1962169	0.860	0.894
	0.092	28.59	9.47	1664300	0.914	0.934
	0.142	30.58	7.53	1395212	0.947	0.965
	0.230	31.03	7.18	1312780	0.957	0.969
	0.378	32.54	6.01	1110832	0.974	0.982
Boat	0.031	23.32	17.39	2852722	0.711	0.806
	0.046	23.53	16.97	2761318	0.743	0.821
	0.085	25.22	13.97	2239399	0.847	0.892
	0.132	26.59	11.93	1909585	0.907	0.937
	0.208	27.51	10.75	1701478	0.931	0.948
	0.357	29.87	8.18	1292754	0.964	0.971
Lake	0.033	22.44	19.23	3152624	0.752	0.818
	0.047	22.59	18.91	3076984	0.773	0.828
	0.091	24.69	14.85	2382606	0.878	0.905
	0.140	26.36	12.26	1979480	0.927	0.945
	0.216	27.02	11.36	1806957	0.942	0.952
	0.366	29.33	8.70	1377865	0.969	0.972
Straw	0.030	18.32	30.94	6520098	0.378	0.656
	0.073	19.51	26.95	5614869	0.613	0.792
	0.104	20.48	24.11	5019019	0.723	0.859
	0.145	21.61	22.31	4609318	0.794	0.899
	0.304	23.28	17.47	3471536	0.896	0.940
	0.458	25.41	13.66	2646641	0.947	0.972
Tank	0.028	26.08	12.64	2528483	0.667	0.799
	0.051	26.76	11.70	2311961	0.749	0.849
	0.081	28.08	10.05	1983966	0.845	0.916
	0.120	28.80	9.25	1821468	0.892	0.944
	0.214	29.94	8.113	1572469	0.926	0.960
	0.359	31.96	6.43	1233007	0.962	0.982
Aerial	0.030	20.44	24.21	4210707	0.588	0.737
	0.042	20.73	23.44	4024164	0.639	0.762
	0.084	22.55	18.93	3166496	0.797	0.870
	0.128	23.73	16.55	2719732	0.870	0.919
	0.201	24.76	14.73	2343067	0.908	0.936
	0.347	27.30	16.99	1684364	0.956	0.968

(continued)

Table 7 (continued)

Images	Code rate	PSNR (dB)	RMSE	SAD	MSSIM	FSIM
Wall	0.019	30.87	7.29	1498038	0.763	0.791
	0.046	31.54	6.75	1385567	0.826	0.846
	0.072	32.31	6.61	1271507	0.881	0.910
	0.104	32.80	5.83	1202858	0.908	0.946
	0.209	33.87	5.16	1037257	0.936	0.956
	0.319	35.47	4.29	856799	0.965	0.982
Pentagon	0.028	24.49	15.19	2830518	0.645	0.779
	0.040	24.77	14.71	2719454	0.690	0.798
	0.077	26.43	12.44	2713948	0.822	0.889
	0.119	27.51	10.73	1891397	0.866	0.936
	0.181	28.38	9.70	1685452	0.914	0.944
	0.325	30.68	7.45	1259609	0.957	0.971

and decoding images of size 512×512 using proposed algorithm, FTR and JPEG for achieving almost similar compression rates for various images is given in Table 8. Comparison of PSNR for different compressed images, achieved using proposed method, RFVQ, FTR, FEQ and JPEG methods of compression with respect to code rate is shown in Fig. 8 for four images of size 256×256 and four images of size 512×512. The increasing curve of the proposed method over other methods shows the superiority of the proposed algorithm. At some bit rate, the RFVQ supersedes proposed edgeFuzzy algorithm but results in higher time complexity because of large number of clusters needed.

In comparison to authors' pairFuzzy [19] algorithm high compression ratio and high PSNR is achieved using proposed algorithm. The use of artifact reduction algorithm reduces the artifacts but at the cost of blurring the compressed image.

6 Conclusion

This chapter presents an edge based image compression algorithm in F-transform domain named edgeFuzzy. Input image blocks are first classified as LI, MI and HI blocks based on the edge image obtained using canny edge detection algorithm. Since LI blocks contain small number of edge pixels and hence less information, is therefore compressed more as compared to MI and HI blocks using F-transform. Huffman encoding is further performed on the compressed image to achieve low bit rate. Both subjective and objective evaluation shows that the proposed algorithm outperforms over other state of art image compression algorithms.

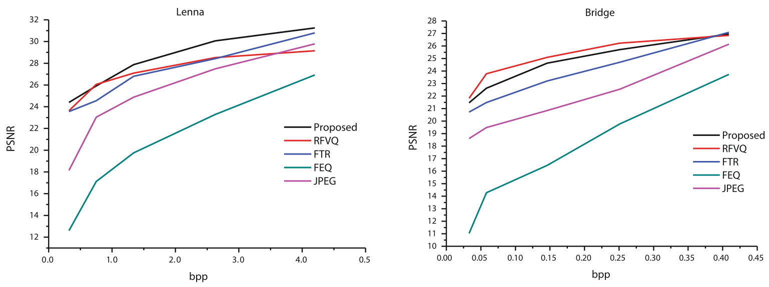
Table 8 Coding and decoding timing for test images (seconds)

Images	Code rate	Proposed coding time	Proposed decoding time	Code rate	FTR coding time	FTR decoding time	Code rate	JPEG coding time	JPEG decoding time
Elaine	0.033	5.45	3.46	0.035	5.71	5.15	0.034	5.63	5.55
	0.053	5.54	2.99	0.062	6.53	5.79	0.061	6.50	6.42
	0.092	11.04	6.11	0.097	7.59	6.02	0.086	6.70	6.43
	0.142	4.99	2.27	0.140	6.76	5.95	0.140	6.53	6.55
	0.230	5.40	3.55	0.250	6.35	6.20	0.240	6.50	5.95
	0.378	9.81	7.75	0.391	8.13	8.95	0.431	6.25	6.11
Boat	0.031	5.05	3.42	0.035	6.10	5.43	0.036	6.39	6.27
	0.046	5.08	3.21	0.062	6.05	5.25	0.063	5.46	5.48
	0.085	11.48	7.90	0.097	6.27	5.32	0.092	6.41	5.91
	0.132	5.25	3.49	0.140	6.79	6.35	0.140	7.01	6.95
	0.208	5.59	3.55	0.250	7.22	7.15	0.241	6.39	6.35
	0.357	14.03	6.99	0.391	7.29	6.44	0.432	6.43	6.25
Lake	0.033	5.84	3.46	0.035	7.53	6.59	0.036	6.47	5.48
	0.047	6.07	3.59	0.062	7.47	6.75	0.062	5.86	5.55
	0.091	8.50	5.13	0.097	6.25	5.49	0.089	5.79	5.65
	0.140	2.95	1.58	0.140	6.27	5.52	0.142	5.89	6.21
	0.216	4.24	1.59	0.250	6.99	6.58	0.239	6.01	6.00
	0.366	6.30	3.43	0.391	7.16	6.53	0.429	6.12	6.31
Straw	0.030	5.73	3.50	0.035	6.84	6.65	0.035	6.61	4.55
	0.073	5.68	3.26	0.062	7.31	6.60	0.062	6.22	6.21
	0.104	10.67	6.14	0.097	6.95	6.95	0.089	6.63	6.13
	0.145	3.23	1.48	0.140	7.62	7.11	0.139	5.89	5.95
	0.304	3.67	1.62	0.250	6.50	6.48	0.249	5.92	5.45
	0.458	11.78	6.38	0391	7.11	7.02	0.439	6.23	5.58

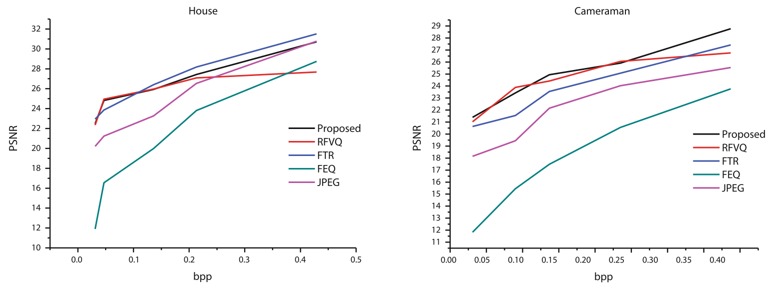
(continued)

Table 8 (continued)

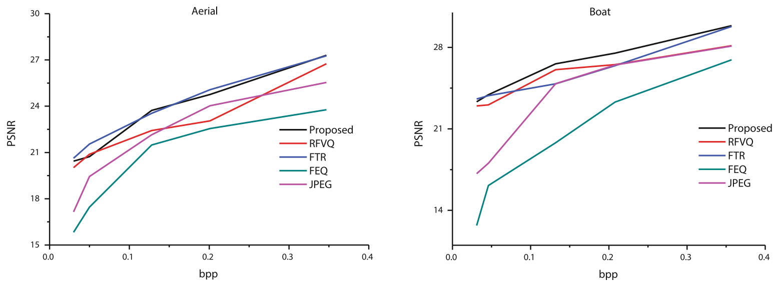
Images	Code rate	Proposed coding time	Proposed decoding time	Code rate	FTR coding time	FTR decoding time	Code rate	JPEG coding time	JPEG decoding time
Tank	0.028	5.60	3.13	0.035	8.31	7.35	0.034	6.33	6.21
	0.051	3.81	1.88	0.062	8.95	7.39	0.062	6.49	6.25
	0.081	7.56	4.53	0.097	9.04	8.24	0.093	6.59	6.14
	0.120	3.19	1.49	0.140	8.22	7.31	0.136	7.01	6.42
	0.214	3.66	1.54	0.250	7.08	7.00	0.241	7.23	6.59
	0.359	4.79	2.20	0.391	7.25	7.52	0.438	6.95	6.53
Aerial	0.030	2.95	1.43	0.035	4.28	4.52	0.035	6.17	4.28
	0.042	3.08	1.50	0.062	6.94	6.18	0.062	6.12	4.65
	0.084	5.60	2.18	0.097	4.51	4.75	0.094	5.92	5.32
	0.128	3.70	3.02	0.140	6.21	6.06	0.139	6.25	5.26
	0.201	4.14	3.18	0.250	4.95	5.47	0.242	5.93	5.23
	0.347	5.99	2.92	0.391	5.32	5.95	0.434	5.93	5.26
Wall	0.019	3.46	1.84	0.035	5.74	5.26	0.035	5.62	5.56
	0.046	4.77	2.62	0.062	6.06	5.94	0.062	5.96	6.12
	0.072	5.20	1.72	0.097	6.42	6.12	0.090	5.86	6.18
	0.104	5.21	2.77	0.140	5.98	5.69	0.140	6.11	6.54
	0.209	5.51	2.74	0.250	6.43	6.12	0.242	6.24	5.89
	0.319	6.94	2.95	0.391	6.92	5.29	0.426	6.13	5.99
Pentagon	0.028	1.67	0.91	0.035	3.95	1.78	0.044	4.98	5.09
	0.040	2.35	1.21	0.062	4.26	3.12	0.065	5.16	5.26
	0.077	2.29	1.02	0.097	4.78	3.26	0.098	5.23	5.32
	0.119	2.62	1.41	0.140	5.56	3.56	0.152	5.26	5.45
	0.181	4.87	2.38	0.250	5.53	3.93	0.301	5.84	5.82
	0.325	5.96	2.97	0.391	4.08	2.96	0.394	5.62	5.71



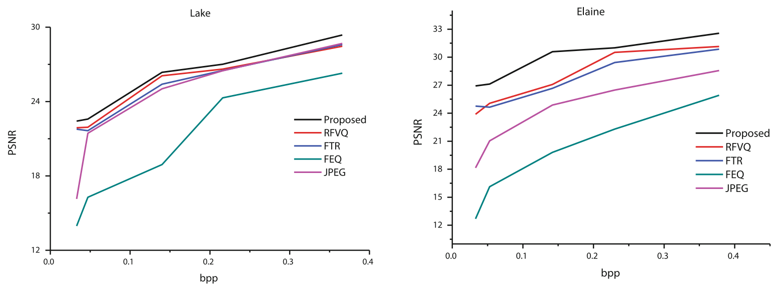
(a) Lenna and Bridge test images of sizes 256 × 256



(b) House and Cameraman test images of sizes 256 × 256



(c) Aerial and Boat test images of sizes 512 × 512



(d) Lake and Elaine test images of sizes 512 × 512

Fig. 8 PSNR comparison of Proposed, RFVQ, FTR, FEQ and JPEG methods

References

1. Lossy image compression, in *Encyclopedia of GIS* (Springer, US, 2008)
2. M. Biswas, S. Kumar, T.Q. Nguyen, N. Balram, Support vector machine (SVM) based compression artifact-reduction technique. J. Soc. Inf. Display **15**(8), 625–634 (2007)
3. S. Saha, Image compression: from DCT to wavelets: a review. ACM Crossroads **6**(3), 12–21 (2000)
4. S. Wang, T. Lin, United coding method for compound image compression. Multimedia Tools Appl. **71**(3), 1263–1282 (2014)
5. W. Xiaolin, X. Zhang, X. Wang, Low bit-rate image compression via adaptive down-sampling and constrained least squares upconversion. IEEE Trans. Image Process. **18**(3), 552–561 (2009)
6. L. Xi, L. Zhang, A study of fractal image compression based on an improved genetic algorithm. Int. J. Nonlinear Sci. **3**(2), 116–124 (2007)
7. Q. Xia, X. Li, L. Zhuo, K.M. Lam, A novel low-bit-rate image compression algorithm, in *Advances in Multimedia Information Processing-PCM 2010* (Springer, 2011), pp. 100–110
8. I. Perfilieva, B. De Baets, Fuzzy transforms of monotone functions with application to image compression. Inf. Sci. **180**(17), 3304–3315 (2010)
9. F. Di Martino, V. Loia, I. Perfilieva, S. Sessa, An image coding/decoding method based on direct and inverse fuzzy transforms. Int. J. Approx. Reason. **48**(1), 110–131 (2008)
10. A. Petrosino, A. Ferone, Rough fuzzy set-based image compression. Fuzzy Sets Syst. **160**(10), 1485–1506 (2009)
11. L. Wang, L. Jiao, W. Jiaji, G. Shi, Y. Gong, Lossy-to-lossless image compression based on multiplier-less reversible integer time domain lapped transform. Signal Process.: Image Commun. **25**(8), 622–632 (2010)
12. D. Ke, L. Peng, New algorithms for preserving edges in low-bit-rate wavelet-based image compression. IEEJ Trans. Electr. Electron. Eng. **7**(6), 539–545 (2012)
13. U. Desai, I. Masaki, A. Chandrakasan, B.K.P. Horn, Edge and mean based image compression, in *Proceedings of IEEE Acoustics, Speech, and Signal Processing, ICASP 1996*, vol. 49 (1996)
14. A. Mertins, Image compression via edge-based wavelet transform. Opt. Eng. **38**(6), 991–1000 (1999)
15. A. Avramovic, Lossless compression of medical images based on gradient edge detection, in *Proceedings of the 19th Telecommunications Forum (TELFOR), Belgrade* (2011), pp. 1199–1202
16. D. Gambhir, N. Rajpal, Fuzzy edge detector based adaptive quantization image coding: FuzzAQC, in *Proceedings of the Recent Advances in Information Technology (RAIT)* (2012), pp. 101–106
17. T.M. Amarunnishad, V.K. Govindan, A.T. Mathew, Improving BTC image compression using a fuzzy complement edge operator. Signal Process. **88**(12), 2989–2997 (2008)
18. D. Gambhir, N. Rajpal, Image coding using fuzzy edge classifier and fuzzy f-transform: dualFuzzy. Int. J. Fuzzy Comput. Modell. **1**(3), 235–251 (2015)
19. D. Gambhir, N. Rajpal, Improved fuzzy transform based image compression and fuzzy median filter based its artifact reduction: pairfuzzy. Int. J. Mach. Learn. Cybern. **6**(6), 935–952 (2015)
20. I. Perfilieva, Fuzzy transforms. Fuzzy Sets Syst. **15**(8), 993–1023 (2006)
21. I. Perfilieva, R. Valásek, Data compression on the basis of fuzzy transforms, in *EUSFLAT Conference* (Citeseer, 2005), pp. 663–668
22. J. Canny, A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **6**, 679–698 (1986)
23. D.A Huffman et al., A method for the construction of minimum redundancy codes, in *Proceedings of I.R.E* (1952), pp. 1099–1101
24. USC-SIPI image database. http://sipi.usc.edu/database/
25. Z. Wang, A.C. Bovik, Mean squared error: love it or leave it? a new look at signal fidelity measures. IEEE Signal Process. Mag. **26**(1), 98–117 (2009)
26. Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)

27. L. Zhang, D. Zhang, X. Mou, FSIM: a feature similarity index for image quality assessment. IEEE Trans. Image Process. **20**(8), 2378–2386 (2011)
28. S. Serikawa, H. Lu, L. Zhang, Maximum local energy: an effective approach for image fusion in beyond wavelet transform domain. Comput. Math. Appl. **64**(5), 996–1003 (2012)
29. A. Shahbahrami, R. Bahrampour, M. Sabbaghi Rostami, M.A. Mobarhan, Evaluation of huffman and arithmetic algorithms for multimedia compression standards. arXiv:1109.0216 (2011)