Available online at www.sciencedirect.com
ScienceDirect
International Journal of Approximate Reasoning
48 (2008) 110–131
INTERNATIONAL JOURNAL OF APPROXIMATE REASONING
www.elsevier.com/locate/ijar

An image coding/decoding method based on direct and inverse fuzzy transforms

Ferdinando Di Martino [a,b], Vincenzo Loia [b], Irina Perfilieva [c], Salvatore Sessa [a,*]

[a] *Università degli Studi di Napoli "Federico II", Dipartimento di Costruzioni e Metodi Matematici in Architettura, Via Monteoliveto 3, 80134 Napoli, Italy*

[b] *Università degli Studi di Salerno, Dipartimento di Matematica e Informatica, Via Ponte Don Melillo, 84084 Fisciano (Salerno), Italy*

[c] *University of Ostrava, Institute for Research and Applications of Fuzzy Modeling, 70103 Ostrava 1, Czech Republic*

Received 25 August 2006; received in revised form 17 June 2007; accepted 25 June 2007
Available online 10 July 2007

Abstract

With some modifications, we adopt the coding/decoding method of image processing based on the direct and inverse fuzzy transforms defined in previous papers. By normalizing the values of its pixels, any image can be considered as a fuzzy matrix (relation) which is subdivided in submatrices (possibly square) called blocks. Each block is compressed with the formula of the discrete fuzzy transform of a function in two variables and successively it is decompressed via the related inverse fuzzy transform. The decompressed blocks are recomposed for the reconstruction of the image, whose quality is evaluated by calculating the PSNR (Peak Signal to Noise Ratio) with respect to the original image. A comparison with the coding/decoding method of image processing based on the fuzzy relation equations with the Lukasiewicz triangular norm and the DCT method are also presented. By using the same compression rate in the three methods, the results show that the PSNR obtained with the usage of direct and inverse fuzzy transforms is higher than the PSNR determined either with fuzzy relation equations method or in the DCT one and it is close to the PSNR determined in JPEG method for small values of the compression rate.

Keywords: Fuzzy relation; Fuzzy relation equation; Fuzzy transform; Image coding/decoding; PSNR; RMSE; DCT; JPEG

1. Introduction

The concept of *Fuzzy transform* (shortly, *F-transform*) of a function, introduced in [18–20,22], establishes a correspondence between the set of continuous functions on the interval $[a, b]$ and the set of n-dimensional vectors. Conversely, the concept of *inverse F-transform* converts an n-dimensional vector into a continuous function which approximates the original function up to a small quantity ε. In many problems involving complex

* Corresponding author.
E-mail addresses: fdimarti@unina.it (F. Di Martino), loia@unisa.it (V. Loia), irina.perfilieva@osu.cz (I. Perfilieva), sessa@unina.it (S. Sessa).

0888-613X/$ - see front matter
doi:10.1016/j.ijar.2007.06.008

computations, then it is possible to operate with an image of the original function obtained by using the F-transform and hence by translating the functional problem into the respective problem of linear algebra, which is more convenient to manipulate since one deals with vectors. After that the computations are made, the result (which is an n-dimensional vector) is converted, via the inverse F-transform, to a continuous function which approximates the original function.

The same ideas concern also functions assuming assigned values in determined points of $[a,b]$ by using the concepts of *discrete F-transform and inverse discrete F-transform*. Indeed, these last concepts are applied to a coding/decoding method of image processing already mentioned in [19], here slightly modified in accordance to the papers [1,2,11,12,16]. In literature the usage of based fuzzy logic methods in image processes is widely known (see, e.g., [6,7,9,10,13,21,23]).

By normalizing the values of its pixels with respect to the length of the gray scale used, any image can be considered as a fuzzy matrix (relation). We subdivide this matrix in submatrices (possibly square) called blocks. Each block is compressed with the formula of the discrete F-transform of a function in two variables and successively it is decompressed via the related discrete inverse F-transform. We recompose the decompressed blocks by obtaining a new image which is very similar to the original image and the quality of this reconstructed image is evaluated by calculating the PSNR (Peak Signal to Noise Ratio) with respect to the original one. A comparison with the coding/decoding method of image processing based on the fuzzy relation equations with the Lukasiewicz triangular norm [1,2] (for short, FEQ) and with the Discrete Cousin Transform (for short, DCT) method are also presented. By using approximately the same compression rate in the three methods, the results show that the PSNR obtained with the usage of direct and inverse F-transforms (briefly, FTR) is higher than the PSNR determined with FEQ and DCT methods. Further, it assumes values close to the PSNR calculated in the JPEG method for low values of the compression rate. This paper is organized as follows: in Section 2 we give the concepts and theorems concerning the F-transforms of a continuous function in one variable and in Section 3, we extend these concepts to the F-transforms of continuous functions in two variables. In Section 4, we show how the techniques based on the discrete F-transform and its inverse are used for coding/decoding processes of images and in Section 5, we present our simulation results based on our proposed algorithm which is firstly compared with the FEQ method, afterwards with DCT and finally, with JPEG, by using several compression rates. In order to have an exhaustive picture of the experiments, we have considered 100 images of sizes 256×256 (pixels) extracted from Image Database of the University of Southern California (http://sipi.usc.edu/database/), but we show the results only for four wellknown images "Bridge", "Camera", "Lena" and "House" for brevity of presentation. Section 6 contains the conclusions.

2. *F*-transforms in one variable

Following the definitions and notations of [19], let $[a,b]$ be a closed interval, $n \geqslant 2$ and $x_1, x_2, \ldots, x_n$ be points of $[a,b]$, called *nodes*, such that $x_1 = a < x_2 < \cdots < x_n = b$. We say that an assigned family of fuzzy sets $A_1, \ldots, A_n : [a,b] \to [1,0]$ is a *fuzzy partition* of $[a,b]$ if the following conditions hold:

(1) $A_i(x_i) = 1$ for every $i = 1, 2, \ldots, n$;
(2) $A_i(x) = 0$ if $x \notin (x_{i-1}, x_{i+1})$, where we assume $x_0 = x_1 = a$ and $x_{n+1} = x_n = b$ by comodity of presentation;
(3) $A_i(x)$ is a continuous function on $[a,b]$;
(4) $A_i(x)$ strictly increases on $[x_{i-1}, x_i]$ for $i = 2, \ldots, n$ and strictly decreases on $[x_i, x_{i+1}]$ for $i = 1, \ldots, n-1$;
(5) $\forall x \in [a,b], \sum_1^n A_i(x) = 1$
The fuzzy sets $\{A_1(x), \ldots, A_n(x)\}$ are called *basic functions*. Moreover, we say that they form an *uniform fuzzy partition* if
(6) $n \geqslant 3$ and $x_i = a + h \cdot (i-1)$, where $h = (b-a)/(n-1)$ and $i = 1, 2, \ldots, n$ (that is the nodes are equidistant);
(7) $A_i(x_i - x) = A_i(x_i + x)$ for every $x \in [0,h]$ and $i = 2, \ldots, n-1$;
(8) $A_{i+1}(x) = A_i(x - h)$ for every $x \in [x_i, x_{i+1}]$ and $i = 1, 2, \ldots, n-1$.

Let $\{A_1, A_2, \ldots, A_n\}$ be a fuzzy partition of $[a,b]$ and $f(x)$ be a continuous function on $[a,b]$. Thus we can consider the following real numbers for $i = 1, \ldots, n$:

$$F_i = \frac{\int_a^b f(x)A_i(x)\mathrm{d}x}{\int_a^b A_i(x)\mathrm{d}x}. \tag{1}$$

Then we can define the n-tuple $[F_1, F_2, \ldots, F_n]$ as the *fuzzy transform* of f with respect to $\{A_1, A_2, \ldots, A_n\}$. The F_i's are called *components* of the F-transform and if the fuzzy partition is uniform, then the components (1) are given (cf. [19, Lemma 1]) by

$$F_i = \begin{cases} \frac{2}{h}\int_{x_1}^{x_2} f(x)A_1(x)\mathrm{d}x & \text{if } i = 1, \\ \frac{1}{h}\int_{x_{i-1}}^{x_i} f(x)A_i(x)\mathrm{d}x & \text{if } i = 2, \ldots, n-1, \\ \frac{2}{h}\int_{x_{n-1}}^{x_n} f(x)A_n(x)\mathrm{d}x & \text{if } i = n. \end{cases} \tag{2}$$

On the basis of knowledge of the components, now we can define the following function on $[a,b]$:

$$f_{F,n}(x) = \sum_{i=1}^{n} F_i A_i(x), \tag{3}$$

where $x \in [a,b]$. It is called *inverse F-transform* of f with respect to $\{A_1, A_2, \ldots, A_n\}$ and it approximates a given continuous function f on $[a,b]$ with arbitrary precision in the sense of the following theorem (cf. [19, Theorem 2]):

Theorem 1. *Let $f(x)$ be a continuous function on $[a,b]$. For every $\varepsilon > 0$, then there exist an integer $n(\varepsilon)$ and a related fuzzy partition $\{A_1, A_2, \ldots, A_{n(\varepsilon)}\}$ of $[a,b]$ such that for all $x \in [a,b]$:*

$$|f(x) - f_{F,n(\varepsilon)}(x)| < \varepsilon,$$

where $f_{F,n(\varepsilon)}(x)$ is the inverse F-transform of f with respect to $\{A_1, A_2, \ldots, A_{n(\varepsilon)}\}$.

Note that such a fuzzy partition $\{A_1, A_2, \ldots, A_{n(\varepsilon)}\}$ of $[a,b]$ is not necessarily uniform. Theorem 1 concerns the continuous case, but now we deal the discrete case, that is we only know that the function f assumes determined values in some points $p_1, \ldots, p_m$ of $[a,b]$. We assume that the set P of these nodes is *sufficiently dense with respect to the fixed partition*, i.e. for each $i = 1, \ldots, n$ there exists an index $j \in \{1, \ldots, m\}$ such that $A_i(p_j) > 0$. Then we can define the n-tuple $[F_1, F_2, \ldots, F_n]$ as the *discrete F-transform* of f with respect to $\{A_1, A_2, \ldots, A_n\}$, where each F_i is given by

$$F_i = \frac{\sum_{j=1}^{m} f(p_j)A_i(p_j)}{\sum_{j=1}^{m} A_i(p_j)} \tag{4}$$

for $i = 1, \ldots, n$. Similarly as in (3), we call the *discrete inverse F-transform* of f with respect to $\{A_1, A_2, \ldots, A_n\}$ to be the following function defined in the same points $p_1, \ldots, p_m$ of $[a,b]$:

$$f_{F,n}(p_j) = \sum_{i=1}^{n} F_i A_i(p_j). \tag{5}$$

Analogously to Theorem 1, we have the following approximation theorem (cf. [19, Theorem 5]):

Theorem 2. *Let $f(x)$ be a function assigned on a set P of points $p_1, \ldots, p_m$ of $[a,b]$. Then for every $\varepsilon > 0$, there exist an integer $n(\varepsilon)$ and a related fuzzy partition $\{A_1, A_2, \ldots, A_{n(\varepsilon)}\}$ of $[a,b]$ such that P is sufficiently dense with respect to $\{A_1, A_2, \ldots, A_{n(\varepsilon)}\}$ and for every $p_j \in [a,b]$, $j = 1, \ldots, m$*

$$|f(p_j) - f_{F,n(\varepsilon)}(p_j)| < \varepsilon$$

holds true.

3. *F*-transforms in two variables

We can extend the above concepts to functions in two variables. Assume that our universe of discourse is the rectangle $[a,b]\times[c,d]$ and let $n,m \geqslant 2$, $x_1,x_2,\ldots,x_n \in [a,b]$ and $y_1,y_2,\ldots,y_m \in [c,d]$ be $n+m$ assigned points, called *nodes*, such that $x_1 = a < x_2 < \cdots < x_n = b$ and $y_1 = c < \cdots < y_m = d$. Furthermore, let $A_1,\ldots,A_n : [a,b] \to [0,1]$ be a *fuzzy partition* of $[a,b]$, $B_1,\ldots,B_m : [a,b] \to [0,1]$ be a *fuzzy partition* of $[c,d]$ and $f(x,y)$ be a continuous function on $[a,b]\times[c,d]$. Then we can define the $n \times m$ matrix $[F_{kl}]$ as the *F-transform* of f with respect to $\{A_1,\ldots,A_n\}$ and $\{B_1,\ldots,B_m\}$ if we have for each $k = 1,\ldots,n$ and $l = 1,\ldots,m$, $x \in [a,b]$ and $y \in [c,d]$ (cf. [19, Definition 49]):

$$F_{kl} = \frac{\int_c^d \int_a^b f(x,y)A_k(x)B_l(y)\mathrm{d}x\,\mathrm{d}y}{\int_c^d \int_a^b A_k(x)B_l(y)\mathrm{d}x\,\mathrm{d}y}. \tag{6}$$

Similarly as the formula (3), we define the *inverse F-transform* of f with respect to $\{A_1,A_2,\ldots,A_n\}$ and $\{B_1,\ldots,B_m\}$ to be the following function on $[a,b]\times[c,d]$ (cf. [19, Definition 51]):

$$f^F_{nm}(x,y) = \sum_{k=1}^{n}\sum_{l=1}^{m} F_{kl}A_k(x)B_l(y). \tag{7}$$

Of course, an approximation theorem, similar to Theorem 1, holds also in the case of two variables (cf. [19, Theorem 14]). In the discrete case, we assume that the function f assumes determined values in some points $(p_j, q_j) \in [a,b]\times[c,d]$, where $i = 1,\ldots,N$ and $j = 1,\ldots,M$. Moreover, the sets $P = \{p_1,\ldots,p_N\}$ and $Q = \{q_1,\ldots,q_M\}$ of these nodes are *sufficiently dense with respect to the chosen partitions*, i.e. for each $i = 1,\ldots,N$ there exists an index $k \in \{1,\ldots,n\}$ such that $A_i(p_k) > 0$ and for each $j = 1,\ldots,M$ there exists an index $l \in \{1,\ldots,m\}$ such that $B_j(q_l) > 0$.

In this case we define the matrix $[F_{kl}]$ to be the discrete F-transform, extension of (4), of f with respect to $\{A_1,\ldots,A_n\}$ and $\{B_1,\ldots B_m\}$ if we have for each $k = 1,\ldots,n$ and $l = 1,\ldots,m$:

$$F_{kl} = \frac{\sum_{j=1}^{M}\sum_{i=1}^{N} f(p_i,q_j)A_k(p_i)B_l(q_j)}{\sum_{j=1}^{M}\sum_{i=1}^{N} A_k(p_i)B_l(q_j)}. \tag{8}$$

By extending (5) to the case of two variables, we give the *discrete inverse F-transform* of f with respect to $\{A_1,A_2,\ldots,A_n\}$ and $\{B_1,\ldots,B_m\}$ to be the following function defined in the same points $(p_j,q_j) \in [a,b]\times[c,d]$, with $i \in \{1,\ldots,N\}$ and $j \in \{1,\ldots,M\}$, as

$$f^F_{nm}(p_i,q_j) = \sum_{k=1}^{n}\sum_{l=1}^{m} F_{kl}A_k(p_i)B_l(q_j). \tag{9}$$

The following generalization of Theorem 2 holds:

Theorem 3. *Let $f(x,y)$ be a function assigned on the points $(p_j,q_j) \in [a,b] \times [c,d]$, with $i \in \{1,\ldots,N\}$ and $j \in \{1,\ldots,M\}$. Then for every $\varepsilon > 0$, there exist two integers $n(\varepsilon)$, $m(\varepsilon)$ and related fuzzy partitions $\{A_1,A_2,\ldots,A_{n(\varepsilon)}\}$ of $[a,b]$ and $\{B_1,B_2,\ldots,B_{m(\varepsilon)}\}$ of $[c,d]$ such that the sets of points $P = \{p_1,\ldots,p_N\}$ and $Q = \{q_1,\ldots,q_M\}$ are sufficiently dense with respect to $\{A_1,A_2,\ldots,A_{n(\varepsilon)}\}$ and $\{B_1,B_2,\ldots,B_{m(\varepsilon)}\}$ and for every $(p_j,q_j) \in [a,b] \times [c,d]$, $i \in \{1,\ldots,N\}$ and $j \in \{1,\ldots,M\}$*

$$|f(p_i,q_j) - f^F_{n(\varepsilon)m(\varepsilon)}(p_i,q_j)| < \varepsilon$$

holds true.

The proof is omitted since it follows the same lines of the analogous Theorem 5 in [19] for one variable.

4. By coding/decoding images

In [19], a method of compression/decompression of images based on the FTR method is mentioned, but here we modify it slightly.

Let R be a grey image divided in $N \times M$ pixels. It is seen as a fuzzy relation $R:(i,j) \in \{1,\ldots,N\} \times \{1,\ldots,M\} \to [0,1]$, $R(i,j)$ being the normalized value of the pixel $P(i,j)$, that is $R(i,j) = P(i,j)/255$ if the length of the grey scale, for instance, has 256 levels. In [19], the image R is compressed by using a discrete F-transform in two variables $[F_{kl}]$ (cf. formula (8)) defined for each $k = 1,\ldots,n$ and $l = 1,\ldots,m$, as

$$F_{kl} = \frac{\sum_{j=1}^{M}\sum_{i=1}^{N} R(i,j)A_k(i)B_l(j)}{\sum_{j=1}^{M}\sum_{i=1}^{N} A_k(i)B_l(j)}, \tag{10}$$

where, by simplicity of notation, we have assumed $p_i = i$ and $q_j = j$ (consequently, $a = c = 1$, $b = N$, $d = M$), $A_1,\ldots,A_n$ and $B_1,\ldots,B_m$, with $n \ll N$ and $m \ll M$, are basic functions forming a fuzzy partition of the real intervals $[1,N]$ and $[1,M]$, respectively. The compressed image can be decoded by using the following inverse discrete F-transform (cf. formula (9)) for every $(i,j) \in \{1,\ldots,N\} \times \{1,\ldots,M\}$:

$$R^F_{nm}(i,j) = \sum_{k=1}^{n}\sum_{l=1}^{m} F_{kl}A_k(i)B_l(j). \tag{11}$$

We have subdivided the image R of sizes $N \times M$ (pixels) in submatrices R_B of sizes $N(B) \times M(B)$ (pixels), called blocks (cf. [1,2]), each compressed to a block F_B of sizes $n(B) \times m(B)(3 \leqslant n(B) < N(B),\ 3 \leqslant m(B) < M(B))$ via the discrete F-transform $F_{n(B)m(B)}[R_B] = (F^B_{kl})$ (cf. formula (10)) whose components, for each $k = 1,\ldots,n(B)$ and $l = 1,\ldots,m(B)$, are given by

$$F^B_{kl} = \frac{\sum_{j=1}^{M(B)}\sum_{i=1}^{N(B)} R_B(i,j)A_k(i)B_l(j)}{\sum_{j=1}^{M(B)}\sum_{i=1}^{N(B)} A_k(i)B_l(j)}. \tag{12}$$

The following basic functions $\{A_1,\ldots,A_{n(B)}\}$ and $\{B_1,\ldots,B_{m(B)}\}$, used in (12), form an uniform fuzzy partition of $[1,N(B)]$ and $[1,M(B)]$, respectively:

$$\begin{aligned}
A_1(x) &= \begin{cases} 0.5\left(1+\cos\frac{\pi}{h}(x-x_1)\right) & \text{if } x \in [x_1,x_2],\\ 0 & \text{otherwise,}\end{cases}\\
A_k(x) &= \begin{cases} 0.5\left(1+\cos\frac{\pi}{h}(x-x_k)\right) & \text{if } x \in [x_{k-1},x_{k+1}],\\ 0 & \text{otherwise,}\end{cases}\\
A_n(x) &= \begin{cases} 0.5\left(1+\cos\frac{\pi}{h}(x-x_n)\right) & \text{if } x \in [x_{n-1},x_n],\\ 0 & \text{otherwise,}\end{cases}
\end{aligned} \tag{13}$$

where $n = n(B)$, $k = 2,\ldots,n$, $h = (N(B)-1)/(n-1)$, $x_k = 1 + h\cdot(k-1)$ and

$$\begin{aligned}
B_1(y) &= \begin{cases} 0.5\left(1+\cos\frac{\pi}{s}(y-y_1)\right) & \text{if } y \in [y_1,y_2],\\ 0 & \text{otherwise,}\end{cases}\\
B_t(y) &= \begin{cases} 0.5\left(1+\cos\frac{\pi}{s}(y-y_t)\right) & \text{if } y \in [y_{t-1},y_{t+1}],\\ 0 & \text{otherwise,}\end{cases}\\
B_m(y) &= \begin{cases} 0.5\left(1+\cos\frac{\pi}{s}(y-y_m)\right) & \text{if } y \in [y_{m-1},y_m],\\ 0 & \text{otherwise,}\end{cases}
\end{aligned} \tag{14}$$

where $m = m(B)$, $t = 2,\ldots,m$, $s = (M(B)-1)/(m-1)$, $y_t = 1 + s\cdot(t-1)$.

The compressed block F_B is decoded to a block $R^F_{n(B)m(B)}$ of sizes $N(B) \times M(B)$ by using the inverse discrete F-transform (cf. formula (11)) defined for every $(i,j) \in \{1,\ldots,N_B\} \times \{1,\ldots,M_B\}$ as

$$R^F_{n(B)m(B)}(i,j) = \sum_{k=1}^{n(B)} \sum_{l=1}^{m(B)} F^B_{kl} A_k(i) B_l(j), \tag{15}$$

which approximates the original block R_B with arbitrary precision in the sense of Theorem 3. For every block R_B and ε, this theorem guarantees the existence of integers $n(B) = n(B,\varepsilon)$ and $m(B) = m(B,\varepsilon)$ such that, by taking in account formula (15), the following inequality:

$$|R_B(i,j) - R^F_{n(B)m(B)}(i,j)| < \varepsilon$$

holds true. Unfortunately Theorem 3 does not give a practical method for building such integers $n(B,\varepsilon)$ and $m(B,\varepsilon)$ for an arbitrary ε. Thus we assume several known values of $n(B)$ and $m(B)$ with $n(B) < N(B)$ and $m(B) < M(B)$ and further, for every block R_B we consider different compression rates $\rho(B)$ given by $\rho(B) = (n(B) \cdot m(B))/(N(B) \cdot M(B))$. By simplicity, we have considered $N = M$ and $N(B) = M(B)$, that is square matrices subdivided in square blocks, in turn compressed to square blocks F_B with size $n(B) = m(B)$ and decoded to blocks $R^F_{n(B)m(B)}$ with sizes $N(B) = M(B)$. For each compression rate, we evaluate the quality of the reconstructed image via the Peak Signal to Noise Ratio (shortly, PSNR) given by

$$\text{PSNR} = 20\log_{10} \frac{255}{\text{RMSE}}, \tag{16}$$

where RMSE (Root Mean Square Error) is given by

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} \sum_{j=1}^{M} (R(i,j) - R^F_{NM}(i,j))^2}{N \times M}}. \tag{17}$$

We note that R^F_{NM} in formula (17) represents the reconstructed image obtained from the recomposition of the blocks $R^F_{n(B)m(B)}$. In order to give a precise idea we give a suitable example of a original block R_B with $N(B) = M(B) = 8$, firstly compressed to a block F_B with sizes $n(B) = m(B) = 3$ and after decoded to a block $R^F_{n(B)m(B)}$ with $N(B) = M(B) = 8$, in which the corresponding values of the normalized pixels vary within a grey scale of length equal to 255.

Example. Consider the following fuzzy relation R of sizes 8×8 with value pixels between 0 and 255 corresponding to the image of Fig. 1, which we normalize by obtaining the fuzzy relation R_B:

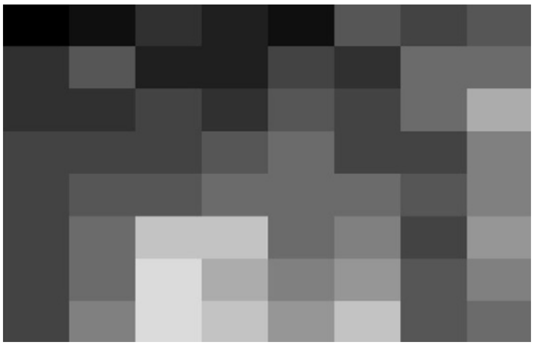

Fig. 1. The original image 8×8.

$$R = \begin{bmatrix} 21 & 27 & 65 & 44 & 36 & 98 & 87 & 112 \\ 63 & 98 & 46 & 58 & 80 & 75 & 117 & 120 \\ 75 & 71 & 78 & 75 & 102 & 88 & 122 & 176 \\ 79 & 85 & 86 & 103 & 123 & 96 & 91 & 145 \\ 83 & 113 & 101 & 118 & 131 & 121 & 98 & 134 \\ 85 & 131 & 201 & 203 & 127 & 148 & 87 & 165 \\ 88 & 124 & 212 & 189 & 139 & 162 & 106 & 139 \\ 91 & 141 & 224 & 197 & 154 & 197 & 111 & 133 \end{bmatrix},$$

$$R_B = \begin{bmatrix} 0.082 & 0.105 & 0.253 & 0171 & 0.140 & 0.382 & 0.339 & 0.437 \\ 0.246 & 0.382 & 0.179 & 0.226 & 0.312 & 0.292 & 0.457 & 0.468 \\ 0.292 & 0.277 & 0.304 & 0.292 & 0.398 & 0.343 & 0.476 & 0.687 \\ 0.308 & 0.332 & 0.335 & 0.402 & 0.480 & 0.375 & 0.355 & 0.566 \\ 0.324 & 0.411 & 0.394 & 0.460 & 0.511 & 0.472 & 0.382 & 0.523 \\ 0.332 & 0.511 & 0.785 & 0.792 & 0.496 & 0.578 & 0.339 & 0.644 \\ 0.343 & 0.484 & 0.828 & 0.738 & 0.542 & 0.632 & 0.414 & 0.542 \\ 0.355 & 0.550 & 0.875 & 0.769 & 0.601 & 0.769 & 0.433 & 0.519 \end{bmatrix}.$$

The original block R_B is firstly compressed to a block F_B of sizes 3×3 (hence $\rho = 0{,}14063 = (3 \times 3)/(8 \times 8)$) by using formula (12), in which the basic functions A_1, A_2, A_3 and B_1, B_2, B_3 are defined by formulas (13) and (14), respectively, and they form an uniform fuzzy partition of the interval [1,8]. The obtained block F_B is represented with the following fuzzy relation:

$$F_B = \begin{bmatrix} 0.213 & 0.264 & 0.431 \\ 0.368 & 0.453 & 0.486 \\ 0.501 & 0.671 & 0.521 \end{bmatrix},$$

which we can denormalize by deducing the following matrix:

$$\begin{bmatrix} 54 & 67 & 110 \\ 94 & 116 & 124 \\ 128 & 171 & 133 \end{bmatrix}.$$

The block F_B is decompressed to a block $R^F_{n(B)m(B)}$ of sizes 8×8 via formula (15) by obtaining the following fuzzy relation:

$$R^F_{n(B)m(B)} = \begin{bmatrix} 0.213 & 0.223 & 0.244 & 0.261 & 0.272 & 0.329 & 0.399 & 0.431 \\ 0.242 & 0.253 & 0.277 & 0.297 & 0.307 & 0.355 & 0.415 & 0.441 \\ 0.308 & 0.321 & 0.352 & 0.376 & 0.384 & 0.413 & 0.449 & 0.464 \\ 0.360 & 0.376 & 0.411 & 0.440 & 0.446 & 0.459 & 0.476 & 0.483 \\ 0.374 & 0.391 & 0.429 & 0.460 & 0.465 & 0.473 & 0.483 & 0.488 \\ 0.419 & 0.442 & 0.492 & 0.532 & 0.536 & 0.523 & 0.507 & 0.500 \\ 0.476 & 0.505 & 0.570 & 0.622 & 0.624 & 0.585 & 0.536 & 0.515 \\ 0.501 & 0.533 & 0.605 & 0.662 & 0.663 & 0.613 & 0.549 & 0.521 \end{bmatrix},$$

whose denormalization gives the following relation:

$$\begin{bmatrix} 54 & 57 & 62 & 67 & 69 & 84 & 102 & 110 \\ 62 & 64 & 71 & 76 & 78 & 90 & 106 & 113 \\ 78 & 82 & 90 & 96 & 98 & 105 & 114 & 119 \\ 92 & 96 & 105 & 112 & 114 & 117 & 121 & 123 \\ 95 & 100 & 110 & 117 & 119 & 121 & 123 & 124 \\ 107 & 113 & 126 & 136 & 137 & 134 & 129 & 128 \\ 121 & 129 & 146 & 159 & 159 & 149 & 137 & 131 \\ 128 & 136 & 154 & 169 & 169 & 156 & 140 & 133 \end{bmatrix},$$

which corresponds to the successive image of Fig. 2.

The related PSNR, calculated with formula (16), is equal to 19.4789. Indeed, we see that the PSNR increases if the compression rate ρ increases as shown in Fig. 3 (in which, for sake of completeness, we have $\rho = 0.1463 = (3\times 3)/(8\times 8)$, $\rho = 0.25 = (4\times 4)/(8\times 8)$, $\rho = (5\times 5)/(8\times 8) = 0.39$, $\rho = 0.56 = (6\times 6)/(8\times 8)$, $\rho = 0.77 = (7\times 7)/(8\times 8)$).

We compare the results with those ones obtained by coding/decoding images with the FEQ method, that is with fuzzy relation equations under triangular norms [5,8]. In accordance to the papers [1,2], we use the Lukasiewicz triangular norm $L : [0,1]^2 \to [0,1]$ defined, for all $x, y \in [0,1]$, as

$$xLy = \max\{0, x + y - 1\}.$$

In [1,2] any image R is subdivided in blocks R_B of sizes $N(B)\times M(B)$ (pixels) as well. These blocks are compressed to blocks G_B of sizes $n(B)\times m(B)$ (pixels) via the following equation of "max-t" type (cf. [5]):

$$G_B(p,q) = \bigcup_{i=1}^{N(B)} \bigcup_{j=1}^{M(B)} [(A_p(i) L B_q(j)) L R_B(i,j)], \tag{18}$$

where the codebooks $A_p : i \in \{1,\ldots,N(B)\} \to A_p \in [0,1]$, $p = 1,\ldots,n(B)$, and $B_q : j \in \{1,\ldots,M(B)\} \to B_q \in [0,1]$, $q = 1,\ldots,m(B)$, are fuzzy sets with Gaussian membership functions given by

$$A_p(i) = \exp\left[-\alpha\left(p\frac{N_B}{n_B} - i\right)^2\right],$$

$$B_q(j) = \exp\left[-\alpha\left(q\frac{M_B}{m_B} - j\right)^2\right]$$

Fig. 2. The decompressed image 8×8.

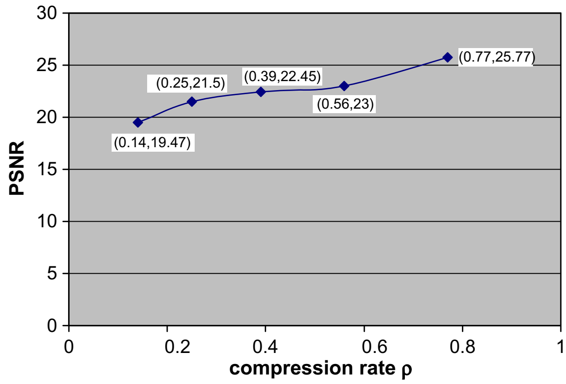

Fig. 3. Behaviour of PSNR w.r.t. the compression rate ρ for image of Fig. 1.

being the parameter α (its range is the set $\{0.1, 0.2, \ldots, 1\}$) optimized in such a way the RMSE is minimized on the decompression of each block. Indeed, each block G_B is decoded to a block D_B of sizes $N(B) \times M(B)$ via the following equation of "min-$\rightarrow_t$" type (cf. [5]):

$$D_B(i,j) = \bigcap_{p=1}^{n(B)} \bigcap_{q=1}^{m(B)} [(A_p(i) L B_q(j)) \rightarrow_L G_B(p,q)], \tag{19}$$

where "$\rightarrow_L$" is the residuum operator of the Lukasiewicz triangular norm given, for all $x, y \in [0,1]$, by

$$x \rightarrow_L y = \min\{1, 1 - x + y\}.$$

We evaluate the PSNR (16) for the final image D (obtained with the recomposition of all the blocks D_B), where the RMSE is given from

$$\sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{M}(R(i,j) - D(i,j))^2}{N \times M}} \tag{20}$$

and we compare it with the PSNR calculated with the FTR method and the DCT method by using compression rates whose values are close to those ones utilized in FEQ and FTR methods. For sake of completeness, we also evaluate PSNR and RMSE in the JPEG method and we take into account also the Sum of Absolute Differences (shortly, SAD) defined as

$$\sum_{i=1}^{N} \sum_{j=1}^{M} |R(i,j) - D(i,j)|, \tag{21}$$

which we compare with the same quantity calculated in the FTR and FEQ methods in correspondence of the (approximately) same compression rates.

5. Simulation results

For our tests we have considered 100 images extracted from Image Database of the University of Southern California (http://sipi.usc.edu/database/) with $N = M = 256$.

For brevity of presentation, here we present our results only for four gray level images "Bridge" (Fig. 4a), "Camera" (Fig. 5a), "Lena" (Fig. 6a) and "House" (see Fig. 7a).

The values of $M(B) = N(B)$ and $m(B) = n(B)$ used in each compression rate in the FTR and FEQ methods are scheduled in Table 1.

The corresponding values of the PSNR for "Bridge", "Camera", "Lena" and "House" are given in Tables 2–5, respectively.

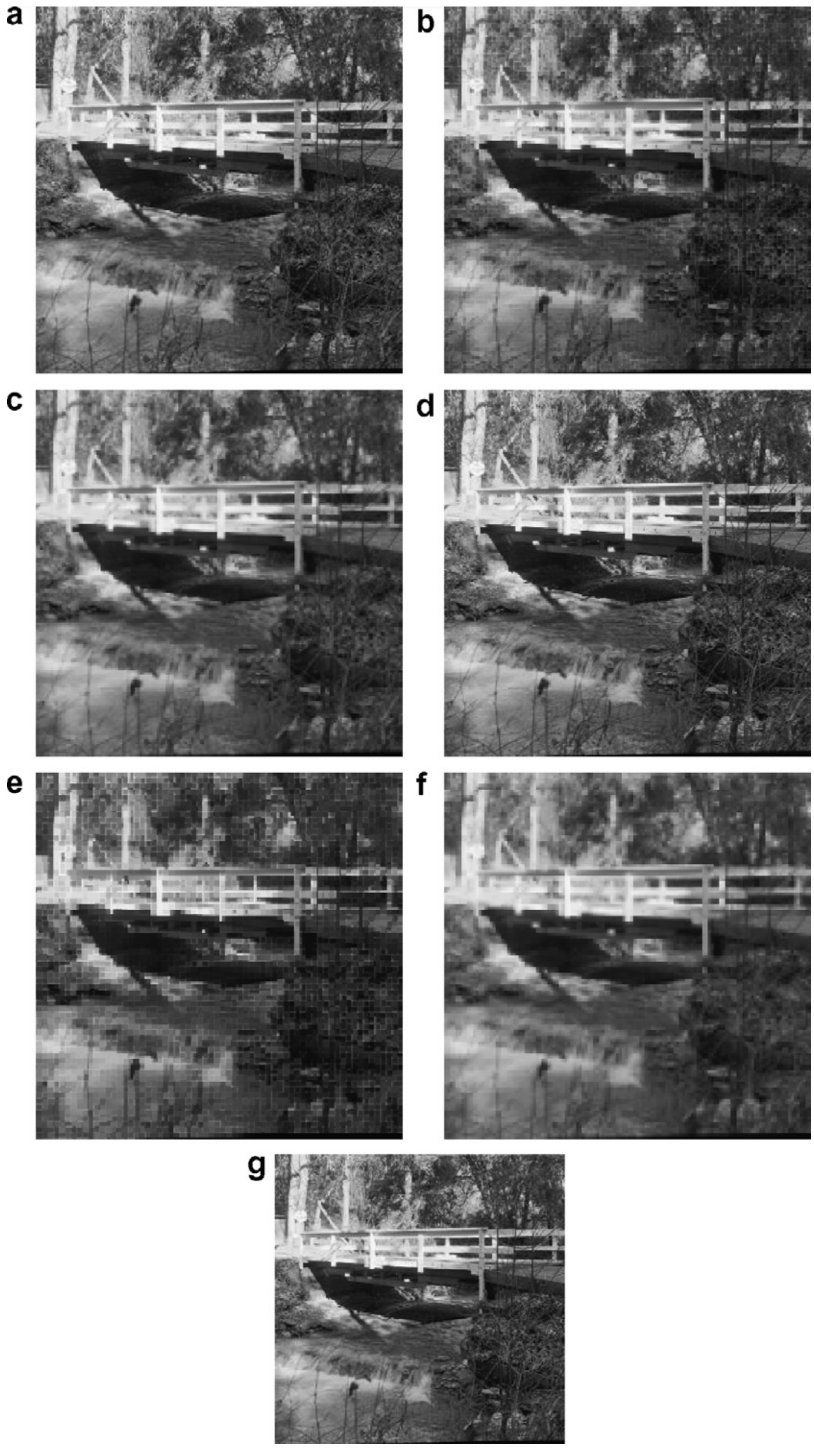

Fig. 4. (a) ‘‘Bridge’’; (b) FEQ, $\rho = 0.44444$; (c) FTR, $\rho = 0.44444$; (d) JPEG, $\rho = 0.430832$; (e) FEQ, $\rho = 0.25$; (f) FTR, $\rho = 0.25$; (g) JPEG, $\rho = 0.244705$.

As shown in Tables 2–5, the PSNR calculated in the FTR method is superior than PSNR evaluated in the FEQ and DCT methods. The successive Tables 6–9 give a precise idea about the coding and decoding times in

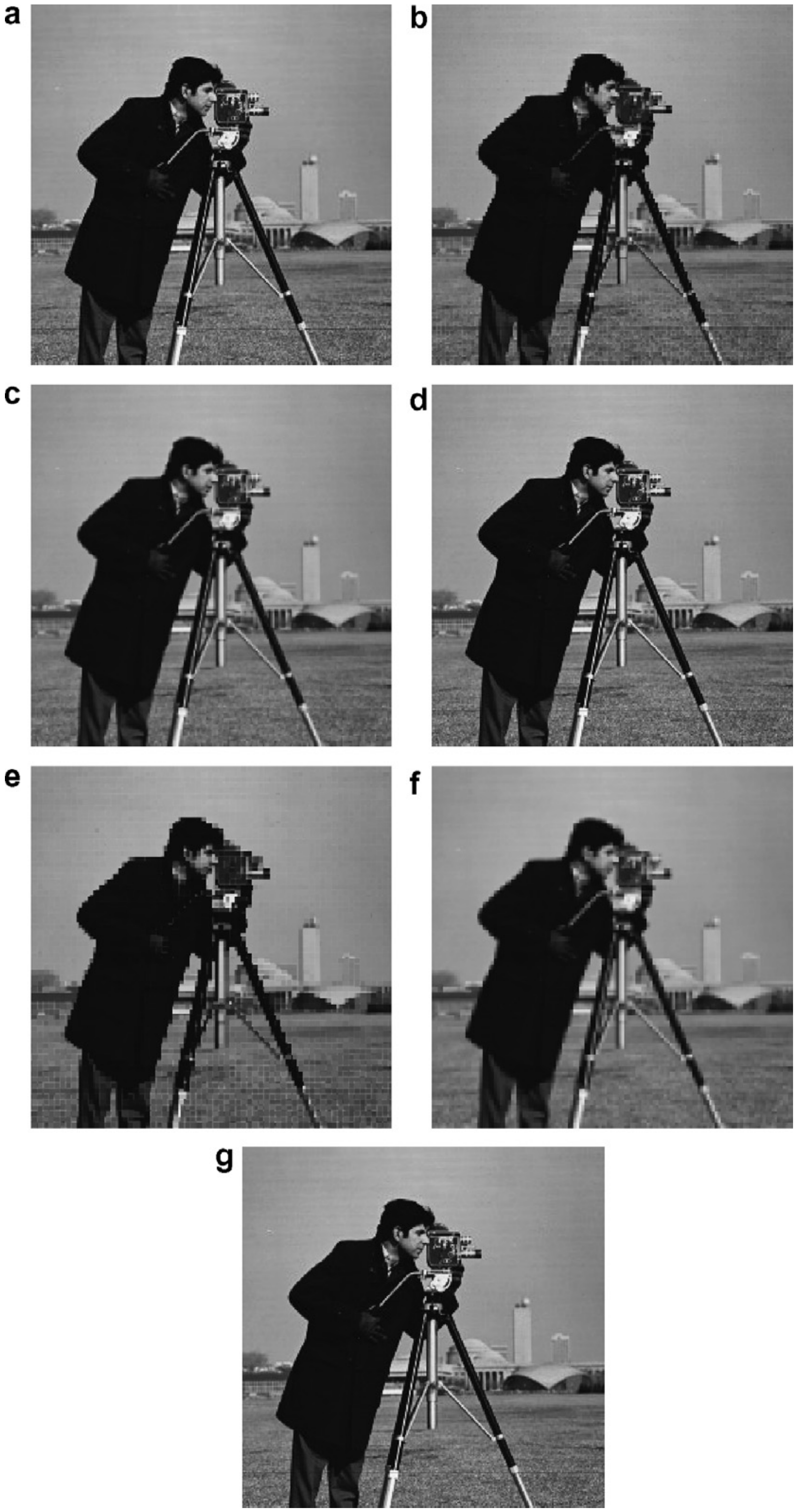

Fig. 5. (a) "Camera"; (b) FEQ, $\rho = 0.44444$; (c) FTR, $\rho = 0.44444$; (d) JPEG, $\rho = 0.436127$; (e) FEQ, $\rho = 0.25$; (f) FTR, $\rho = 0.25$; (g) JPEG, $\rho = 0.249496$.

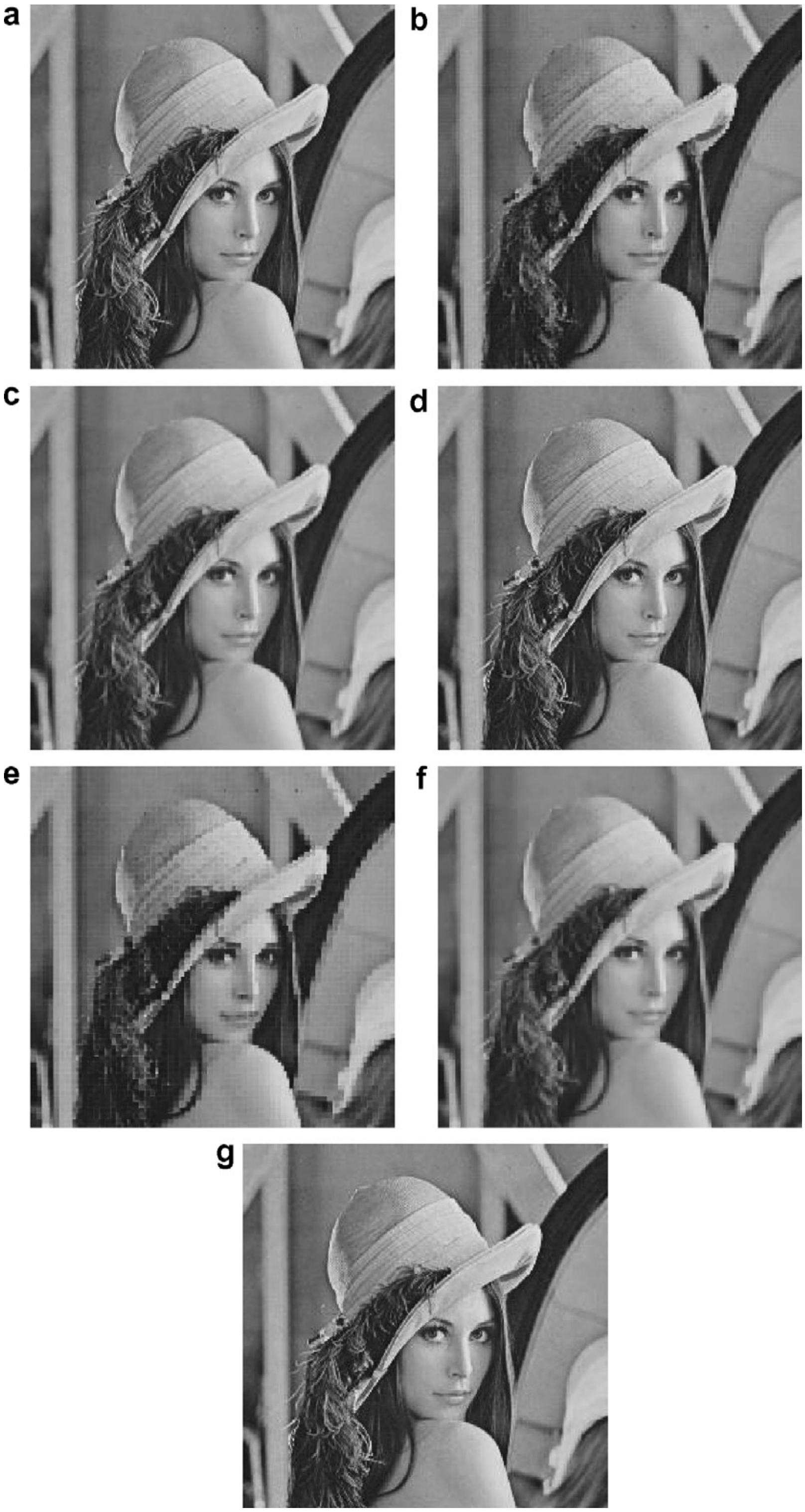

Fig. 6. (a) "Lena"; (b) FEQ, $\rho = 0.44444$; (c) FTR, $\rho = 0.44444$; (d) JPEG, $\rho = 0.439859$; (e) FEQ, $\rho = 0.25$; (f) FTR, $\rho = 0.25$; (g) JPEG, $\rho = 0.240500$.

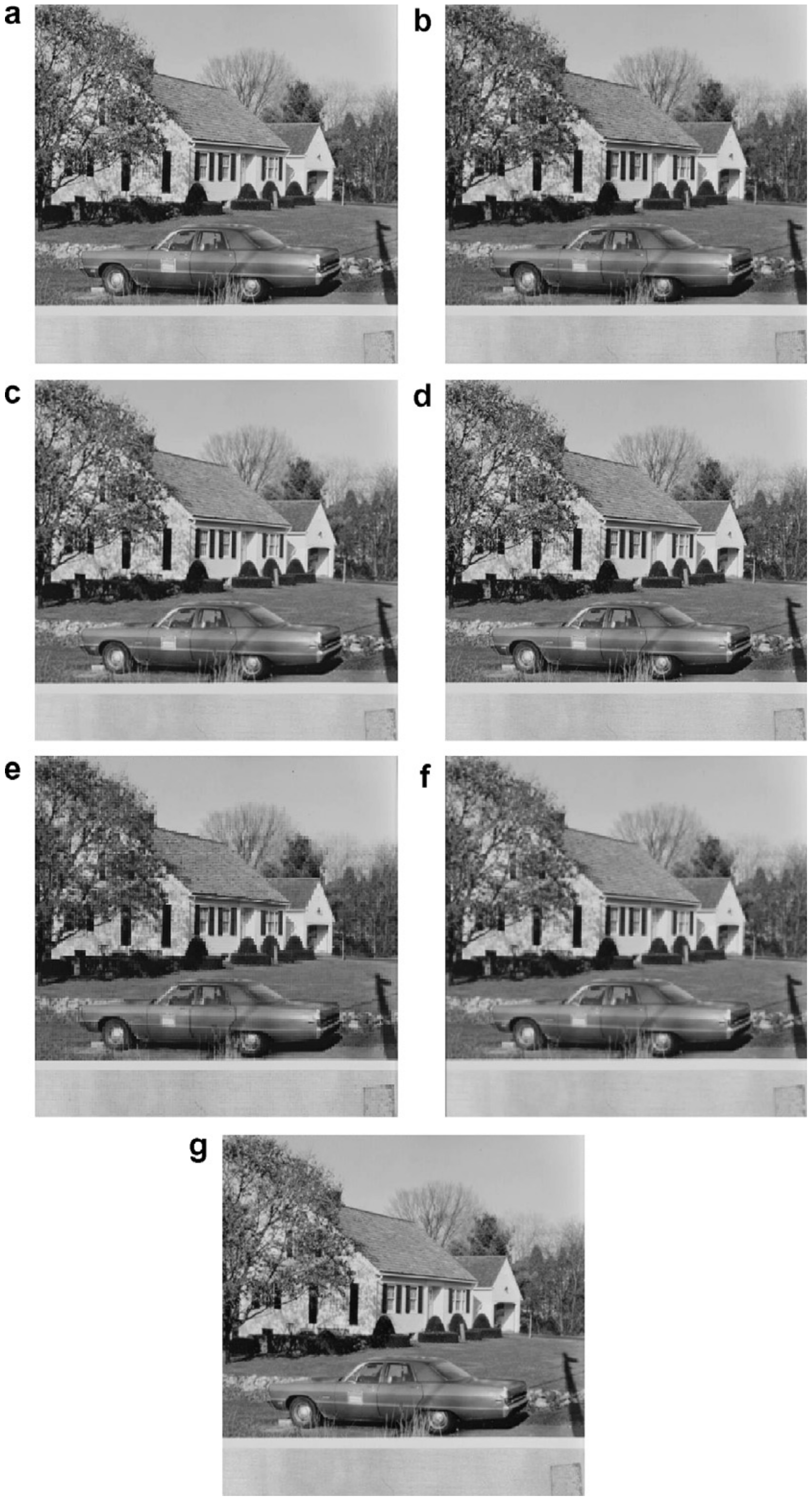

Fig. 7. (a) "House"; (b) FEQ, $\rho = 0.44444$; (c) FTR, $\rho = 0.44444$; (d) JPEG, $\rho = 0.434868$; (e) FEQ, $\rho = 0.25$; (f) FTR, $\rho = 0.25$; (g) JPEG, $\rho = 0.240472$.

Table 1
Compression rates used in the experiments

$\rho(B)$	$M(B)$	$N(B)$	$m(B)$	$n(B)$
0.035156	16	16	3	3
0.062500	8	8	2	2
0.140625	8	8	3	3
0.250000	8	8	4	4
0.444444	3	3	2	2

Table 2
Values of PSNR for "Bridge"

$\rho(B)$	PSNR in FTR	PSNR in FEQ	ρ in DCT	PSNR in DCT
0.035156	20.7262	11.0283	0.034668	18.6115
0.062500	21.4833	14.2812	0.058304	19.4849
0.140625	23.2101	16.4632	0.140305	20.8430
0.250000	24.6975	19.7759	0.244705	22.5470
0.444444	27.0960	23.7349	0.430832	26.1490

Table 3
Values of PSNR for "Camera"

$\rho(B)$	PSNR in FTR	PSNR in FEQ	ρ in DCT	PSNR in DCT
0.035156	20.6304	11.8273	0.034561	18.1489
0.062500	21.5427	15.4535	0.060745	19.4447
0.140625	23.5428	17.4869	0.139465	22.1506
0.250000	25.0676	20.5530	0.249496	24.0288
0.444444	27.4264	23.7706	0.436127	25.5431

Table 4
Values of PSNR for "Lena"

$\rho(B)$	PSNR in FTR	PSNR in FEQ	ρ in DCT	PSNR in DCT
0.035156	23.5685	12.6959	0.034810	21.9341
0.062500	24.5514	17.1275	0.061127	23.0445
0.140625	26.8100	19.7528	0.130330	24.8803
0.250000	28.4310	23.2983	0.240500	27.4874
0.444444	30.8003	26.9285	0.439859	29.7911

Table 5
Values of PSNR for "House"

$\rho(B)$	PSNR in FTR	PSNR in FEQ	ρ in DCT	PSNR in DCT
0.035156	22.9525	11.8965	0.034494	20.2155
0.062500	23.8517	16.5426	0.062360	21.2327
0.140625	26.4038	19.9876	0.137035	23.2612
0.250000	28.1763	23.8031	0.240472	26.5368
0.444444	31.5114	28.7464	0.434868	30.7693

three methods for "Bridge", "Camera", "Lena" and "House", respectively, under the above compression rates.

Tables 6–9 show that the coding (resp. decoding) times in the FTR (resp. DCT) method are lower than the analogous times in the FEQ and DCT (resp. FTR) methods under the above compression rates. For sake of completeness, we report in Tables 10–13 the values of the PSNR (16) evaluated in the JPEG method under the

Table 6
Coding and decoding times for "Bridge"

$\rho(B)$	FTR coding time	FTR decoding time	FEQ coding time	FEQ decoding time	ρ in DCT	DCT coding time	DCT decoding time
0.035156	2.13	5.53	45.68	5.15	0.034668	2.45	2.11
0.062500	1.19	4.19	20.21	2.82	0.058304	6.78	2.36
0.140625	3.59	5.59	54.98	4.52	0.140305	4.02	2.45
0.250000	2.32	4.32	20.36	3.62	0.244705	4.65	2.68
0.444444	3.39	4.40	19.07	4.50	0.430832	4.77	2.54

Table 7
Coding and decoding times for "Camera"

$\rho(B)$	FTR coding time	FTR decoding time	FEQ coding time	FEQ decoding time	ρ in DCT	DCT coding time	DCT decoding time
0.035156	2.16	5.59	23.50	4.15	0.034561	2.36	1.94
0.062500	1.20	4.25	10.87	2.59	0.060745	5.54	2.09
0.140625	3.41	5.49	27.96	4.24	0.139465	18.29	2.40
0.250000	2.11	4.31	10.83	3.18	0.249496	3.07	2.68
0.444444	3.34	5.09	10.80	3.78	0.436127	3.46	2.85

Table 8
Coding and decoding times for "Lena"

$\rho(B)$	FTR coding time	FTR decoding time	FEQ coding time	FEQ decoding time	ρ in DCT	DCT coding time	DCT decoding time
0.035156	4.10	14.82	97.74	15.90	0.034810	4.13	3.31
0.062500	4.38	11.55	57.09	9.54	0.061127	5.11	3.14
0.140625	5.46	14.93	146.94	14.26	0.130330	5.87	3.61
0.250000	6.09	11.34	58.11	11.15	0.240500	6.18	4.11
0.444444	6.21	11.72	55.25	13.66	0.439859	6.24	4.28

Table 9
Coding and decoding times for "House"

$\rho(B)$	FTR coding time	FTR decoding time	FEQ coding time	FEQ decoding time	ρ in DCT	DCT coding time	DCT decoding time
0.035156	4.10	14.82	97.74	15.90	0.034494	4.13	3.31
0.062500	4.38	11.55	57.09	9.54	0.062360	5.11	3.14
0.140625	5.46	14.93	146.94	14.26	0.137035	5.87	3.61
0.250000	6.09	11.34	58.11	11.15	0.240472	6.18	4.11
0.444444	6.21	11.72	55.25	13.66	0.434868	6.24	4.28

Table 10
PSNR and % gain parameters for "Bridge"

$\rho(B)$	ρ in JPEG	PSNR in JPEG	% Gain FTR over FEQ	% Gain JPEG over FTR
0.035156	0.034668	22.6985	87.9364	9.5159
0.062500	0.058304	24.7253	50.4306	15.0907
0.140625	0.140305	28.1149	65.5220	21.1321
0.250000	0.244705	31.2148	24.8868	26.3885
0.444444	0.430832	37.2367	14.1610	37.4250

same compression rates for "Bridge", "Camera", "Lena" and "House", respectively. We have also given in these tables the following parameters:

Table 11
PSNR and % gain parameters for "Camera"

$\rho(B)$	ρ in JPEG	PSNR in JPEG	% Gain FTR over FEQ	% Gain JPEG over FTR
0.035156	0.034561	25.5207	74.4303	23.7043
0.062500	0.060745	28.4293	39.4033	31.9672
0.140625	0.139465	33.4379	52.3460	42.0302
0.250000	0.249496	38.8007	21.9656	54.7842
0.444444	0.436127	45.5878	15.3795	66.2186

Table 12
PSNR and % gain parameters for "Lena"

$\rho(B)$	ρ in JPEG	PSNR in JPEG	% Gain FTR over FEQ	% Gain JPEG over FTR
0.035156	0.034810	29.8727	85.6387	26.74841
0.062500	0.061127	32.4369	43.3449	32.11833
0.140625	0.130330	35.7345	35.7275	33.28795
0.250000	0.240500	37.5461	22.0303	32.06043
0.444444	0.439859	38.4881	14.3780	24.96015

Table 13
PSNR and % gain parameters for "House"

$\rho(B)$	ρ in JPEG	PSNR in JPEG	% Gain FTR over FEQ	% Gain JPEG over FTR
0.035156	0.034494	30.0249	92.9349	30.8132
0.062500	0.062360	32.0180	44.1835	34.2378
0.140625	0.137035	34.2460	32.1009	29.7010
0.250000	0.240472	35.1001	18.3724	24.5731
0.444444	0.434868	35.7719	9.6185	13.5205

$$(\%\ \text{Gain FTR over FEQ}) = [(\text{PSNR in FTR}) - (\text{PSNR in FEQ})] \cdot 100/(\text{PSNR in FEQ}),$$
$$(\%\ \text{Gain JPEG over FTR}) = [(\text{PSNR in JPEG}) - (\text{PSNR in FTR})] \cdot 100/(\text{PSNR in FTR}),$$

in order to obtain the variation in percentage of the PSNR in the JPEG (resp. FTR) method with respect to the PSNR evaluated in the FTR (resp. FEQ) method.

For completeness, we limit ourselves to show only "Bridge" in Fig. 1b–g, "Camera" in Fig. 2b–g, "Lena" in Fig. 3b–g, "House" in Fig. 4b–g reconstructed under the three methods with the compression rates $\rho = 0.444444, 0.25$ for FTR and FEQ with Eqs. (18) and (19) and with the same (approximately equal) values of ρ for JPEG.

The successive Tables 14–17 give the values of the RMSE (17) and of the SAD (21) in three methods for the same images under the above compression rates.

In Table 18 (resp. Table 19) we report the values of the coding and decoding time, with approximately equal compression rates, for "Bridge" and "Camera" (resp. "Lena" and "House") in the JPEG method to be compared with Tables 6–9 which contain the analogous values in the FEQ and FTR methods for the same images.

Table 14
RMSE and SAD for "Bridge"

$\rho(B)$	RMSE in FEQ	RMSE in FTR	SAD in FEQ	SAD in FTR	ρ in JPEG	RMSE in JPEG	SAD in JPEG
0.035156	71.6349	23.4548	3883022	1145018	0.034668	18.6903	944097
0.062500	49.2584	21.4968	2455725	1034990	0.058304	14.8005	737776
0.140625	38.3160	17.6211	1574219	835993	0.140305	10.0183	494429
0.250000	26.1665	14.8479	1076978	689872	0.244705	7.0113	348847
0.444444	16.5880	11.2652	567366	511329	0.430832	3.5051	177114

Table 15
RMSE and SAD for "Camera"

$\rho(B)$	RMSE in FEQ	RMSE in FTR	SAD in FEQ	SAD in FTR	ρ in JPEG	RMSE in JPEG	SAD in JPEG
0.035156	65.3394	23.7148	2560925	816695	0.034561	13.5054	561273
0.062500	43.0393	21.3504	1415428	703207	0.060745	9.6621	388303
0.140625	34.0561	16.9590	1134652	544050	0.139465	5.4281	227029
0.250000	23.9271	14.2285	632856	443017	0.249496	2.9275	133504
0.444444	16.5200	10.8448	358033	325692	0.436127	1.3401	63760

Table 16
RMSE and SAD for "Lena"

$\rho(B)$	RMSE in FEQ	RMSE in FTR	SAD in FEQ	SAD in FTR	ρ in JPEG	RMSE in JPEG	SAD in JPEG
0.035156	59.1218	16.9090	2860780	696303	0.034810	8.1829	394796
0.062500	35.4950	15.0998	1503574	593984	0.061127	6.0911	298826
0.140625	26.2361	11.6423	610129	441547	0.130330	4.1669	205918
0.250000	17.4431	9.6603	603278	354249	0.240500	3.3825	156163
0.444444	11.4845	7.354	329571	258131	0.439859	3.0348	124182

Table 17
RMSE and SAD for "House"

$\rho(B)$	RMSE in FEQ	RMSE in FTR	SAD in FEQ	SAD in FTR	ρ in JPEG	RMSE in JPEG	SAD in JPEG
0.035156	64.8209	18.1515	11963262	2936833	0.034494	8.0407	1460981
0.062500	37.9673	16.3665	6259591	2581496	0.062360	6.3921	1091999
0.140625	25.5364	12.1997	3568132	1873294	0.137035	4.9459	736445
0.250000	16.4583	9.9478	2287157	1451945	0.240472	4.4827	574576
0.444444	9.3158	6.7760	1092637	932965	0.434868	4.1490	502499

Table 18
Coding and decoding times for "Bridge" and "Camera" in JPEG

ρ (Bridge)	Coding time bridge	Decoding time bridge	ρ (Camera)	Coding time camera	Decoding time camera
0.034668	3.94	2.48	0.034561	3.91	2.46
0.058304	16.67	2.79	0.060745	13.38	2.61
0.140305	5.32	3.10	0.139465	42.05	2.99
0.244705	5.72	3.25	0.249496	5.74	3.33
0.430832	7.05	3.57	0.436127	6.61	3.55

Table 19
Coding and decoding times for "Lena" and "House" in JPEG

ρ (Lena)	Coding time lena	Decoding time lena	ρ (House)	Coding time house	Decoding time house
0.034810	6.26	3.99	0.034494	16.98	9.25
0.061127	6.87	4.25	0.062360	19.07	10.00
0.130330	8.16	4.58	0.137035	23.39	11.34
0.240500	9.6	5.15	0.240472	28.64	13.09
0.439859	10.6	5.26	0.434868	33.34	15.15

Then our tests show that the PSNR in the FTR method is less than the PSNR obtained by using the JPEG method. The gain of PSNR obtained with FTR method respect to the PSNR obtained with FEQ method is more evident in images with low compression rate. For these low values of the compression rate, the PSNR obtained with the FTR method has a value close to the PSNR obtained in the JPEG method. Fig. 8 (resp. Figs. 9–11) shows the behaviour of the PSNR obtained in the methods FTR, FEQ and JPEG with respect to the compression rate for "Bridge" (resp. "Camera", "Lena", "House").

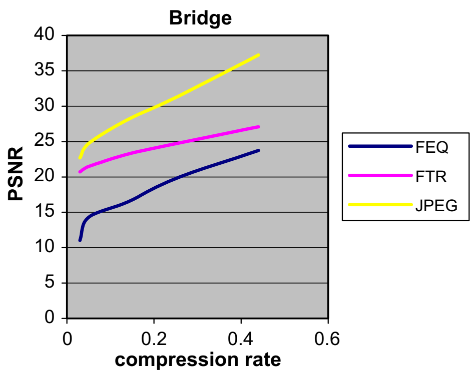

Fig. 8. PSNR in the FTR, FEQ, JPEG methods for “Bridge”.

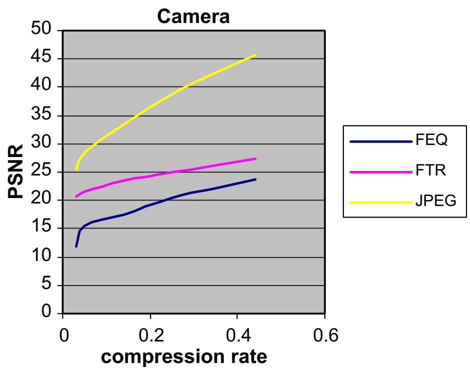

Fig. 9. PSNR in the FTR, FEQ, JPEG methods for “Camera”.

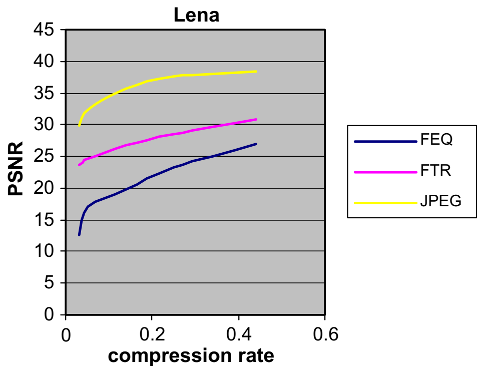

Fig. 10. PSNR in the FTR, FEQ, JPEG methods for “Lena”.

In Fig. 12 (resp. Figs. 13–15) we represent, with respect to the compression rate, the decreasing curve of the gain in percentage of the PSNR obtained by using the FTR method over the PSNR calculated in the FEQ method based on Eqs. (18) and (19) for “Bridge” (resp. “Camera”, “Lena”, “House”) as shown in Table 10 (resp. Tables 11–13). There is analogous representation, with an increasing curve, of the gain in percentage of the PSNR obtained by using the JPEG method over the PSNR calculated in the FTR method.

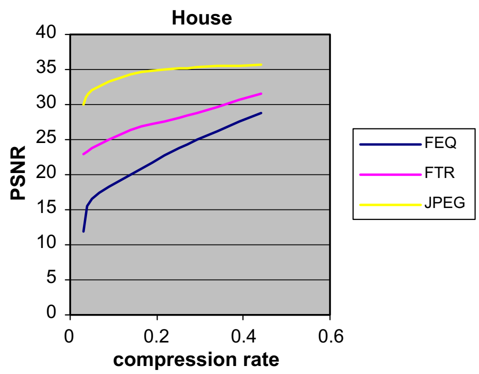

Fig. 11. PSNR in the FTR, FEQ, JPEG methods for "House".

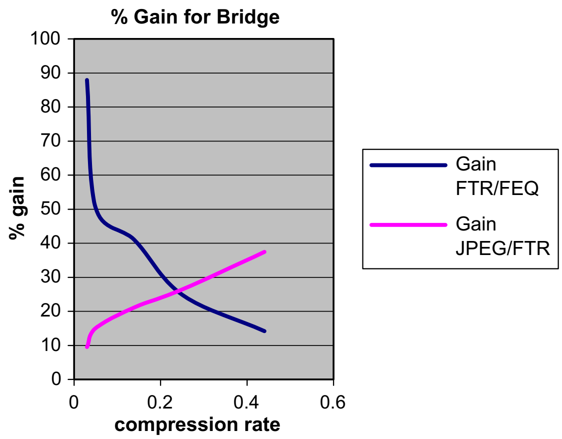

Fig. 12. % Gain of FTR over FEQ and JPEG over FTR for "Bridge".

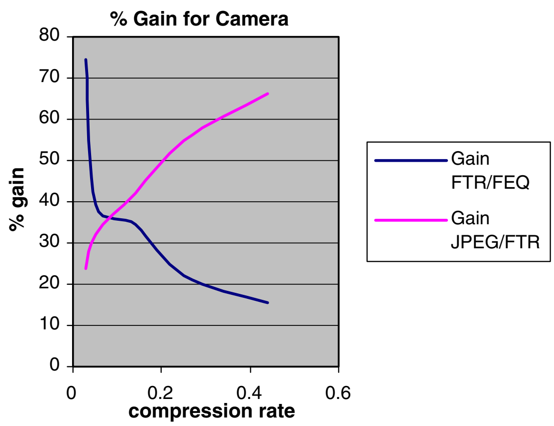

Fig. 13. % Gain of FTR over FEQ and JPEG over FTR for "Camera".

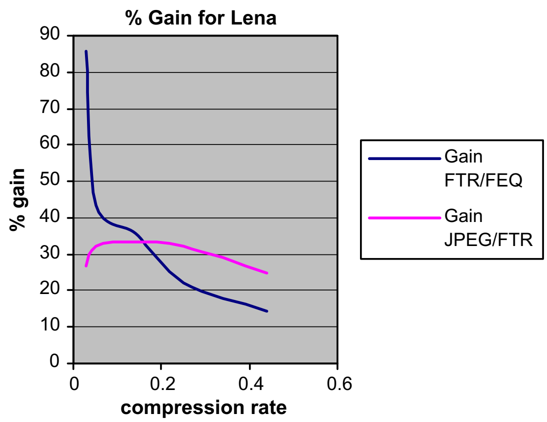

Fig. 14. % Gain of FTR over FEQ and FTR for "Lena".

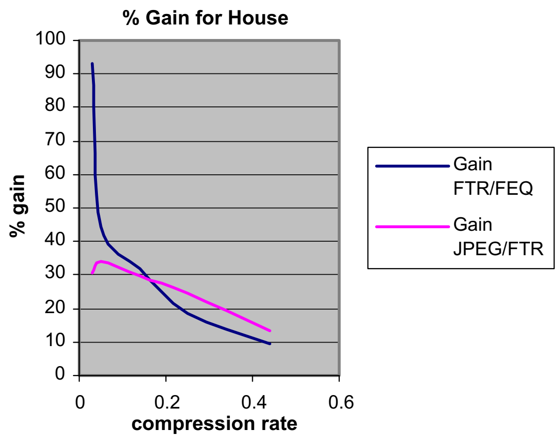

Fig. 15. % Gain of FTR over FEQ and JPEG over JPEG over FTR for "House".

6. Conclusions

The preponderant existence of computer networks used for video conference sessions and multimedia applications makes the research on fast efficient image compression algorithms an issue of vital importance. Thanks to the advent of the Internet, most of the communication tools based on visual interaction are widely exploited for professional and personal needs for which the goal in achieving efficiency is more crucial than precision and detail. In this case, the natural power of the human eye is somehow capable of recovering or integrating the missing information without affecting the overall perception phenomena. In addition to "native" Internet applications where video conferencing may be done by accepting a lost of information, we believe that interesting results can be obtained from our approach if we consider mobile video applications. Ten years ago, it was the internet that threw the telecom, media and marketing worlds into ferment . Nowadays we are observing a counter-tendency: many cable companies deal that will result in co-branded phones allowing consumers to download entertainment programs to their mobile phone and even remotely program their home digital video recorder. This trend opens new scenarios for marketers: if online video e-advertising is ramping up fast then mobile video advertising will follow it closely. For most mobile music and video applications the speed plays a key role; of course the technology will provide more bandwidth, but in the same way the media integration becomes more complex. Within this scenario, our compression/decompression approach can be useful considering the simplicity in implementing the operational framework.

We have shown that compression/decompression of images based on the FTR method gives best results with respect to the FEQ method based on Eqs. (18) and (19): indeed the quality of the image, measured by PSNR, reconstructed with the first method is quite superior with respect to that one of the image decoded with the second method. Further the PSNR of the image deduced with the FTR method is close to the PSNR value obtained with the JPEG method for low values of the compression rate. Moreover, the compression time in the FTR method is minor with respect to the FEQ method. Other studies are necessary: among others, a comparison with other types of fuzzy relation equations used for coding/decoding processes of images (see, e.g., [7,14,15,17]) and applications of the FTR method to other topics like digital watermarking [4], coding/decoding of videos [12], image information retrieval [3].

Acknowledgement

The research made by I. Perfilieva has been partially supported by the project 1M6798555601 of MŠMT (Čech Republic). We thank the referees whose suggestions have greatly improved the presentation.

References

[1] F. Di Martino, V. Loia, S. Sessa, A method for coding/decoding images by using fuzzy relation equations, in: T. Bilgic, B. De Baets, O. Kaynak (Eds.), Fuzzy Sets and Systems – IFSA 2003, Lecture Notes in Artificial Intelligence, vol. 2715, Springer, Berlin, 2003, pp. 436–441.
[2] F. Di Martino, V. Loia, S. Sessa, A method in the compression/decompression of images using fuzzy equations and fuzzy similarities, in: Proceedings of the 10th IFSA World Congress, Istanbul, Turkey, 2003, pp. 524–527.
[3] F. Di Martino, H. Nobuhara, S. Sessa, Eigen fuzzy sets and image information retrieval, in: Proceedings of the International Conference on Fuzzy Information Systems, Budapest, vol. 3, 2004, pp. 1385–1390.
[4] F. Di Martino, S. Sessa, Digital watermarking in coding/decoding processes with fuzzy relation equations, Soft Computing 10 (2006) 238–243.
[5] A. Di Nola, W. Pedrycz, E. Sanchez, S. Sessa, Fuzzy Relation Equations and Their Applications to Knowledge Engineering, Kluwer Academic Publishers, Dordrecht, 1989.
[6] H.M. Gronscurth, Fuzzy data compression for energy optimization models, Energy 23 (1) (1998) 1–9.
[7] K. Hirota, W. Pedrycz, Data compression with fuzzy relational equations, Fuzzy Sets and Systems 126 (3) (2002) 325–335.
[8] E.P. Klement, R. Mesiar, E. Pap, Triangular Norms, Kluwer Academic Publishers, Dordrecht, 2000.
[9] X. Kong, L. Wang, G. Li, Fuzzy clustering algorithms based on resolution and their application in image compression, Pattern Recognition 35 (2002) 2439–2444.
[10] K.F. Loe, W.G. Gu, H.K. Phua, Speed-up fractal image compression with a fuzzy classifier, Signal Processing: Image Communication 10 (4) (1997) 303–311.
[11] V. Loia, W. Pedrycz, S. Sessa, Fuzzy relation calculus in the compression and decompression of fuzzy relations, International Journal of Image and Graphics 2 (2002) 1–15.
[12] V. Loia, S. Sessa, Fuzzy relation equations for coding/decoding processes of images and videos, Information Sciences 171 (2005) 145–172.
[13] S. Makrogiannis, G. Economou, S.K. Fotopoulos, Region oriented compression of color images using fuzzy inference and fast merging, Pattern Recognition 35 (2002) 1807–1820.
[14] H. Nobuhara, W. Pedrycz, K. Hirota, Fast solving method of fuzzy relational equation and its application to lossy image compression, IEEE Transactions on Fuzzy Systems 8 (3) (2000) 325–334.
[15] H. Nobuhara, W. Pedrycz, K. Hirota, Relational image compression: optimizations through the design of fuzzy coders and yuv color space, Soft Computing 9 (6) (2005) 471–479.
[16] H. Nobuhara, K. Hirota, F. Di Martino, W. Pedrycz, S. Sessa, Fuzzy relation equations for compression/decompression processes of colour images in the RGB and YUV colour spaces, Fuzzy Optimization and Decision Making 4 (3) (2005) 235–246.
[17] H. Nobuhara, K. Hirota, W. Pedrycz, S. Sessa, A motion compression/reconstruction method based on max T-norm composite fuzzy relational equations, Information Sciences 176 (2006) 2526–2552.
[18] I. Perfilieva, Fuzzy transforms: application to reef growth problem, in: R.B. Demicco, G.J. Klir (Eds.), Fuzzy Logic in Geology, Academic Press, Amsterdam, 2003, pp. 275–300.
[19] I. Perfilieva, Fuzzy transforms, Fuzzy Sets and Systems 157 (8) (2006) 993–1023.
[20] I. Perfilieva, E. Chaldeeva, Fuzzy transformation, in: Proceedings of the Ninth IFSA World Congress and 20th NAFIPS International Conference, Vancouver, Canada, 2001, pp. 1946–1948.
[21] F. Russo, A nonlinear technique based on fuzzy models for the correction of quantization artifacts in image compression, Measurement 32 (4) (2002) 273–279.

[22] M. Stepnicka, R. Valàsek, Fuzzy transforms for functions with two variables, in: J. Ramik, V. Novak (Eds.), Methods for Decision Support in Environment with Uncertainty – Applications in Economics, Business and Engineering, IRAFM, Ostrava, Czech Republic, 2003, pp. 96–102.
[23] G.E. Tsekouras, A fuzzy vector quantization approach to image compression, Applied Mathematics and Computation 167 (5) (2005) 539–560.