




Multi-level fuzzy transforms image compression

Ferdinando Di Martino^{1,2}  · Salvatore Sessa^{1,2}

Received: 25 September 2017 / Accepted: 7 August 2018 / Published online: 17 August 2018
© Springer-Verlag GmbH Germany, part of Springer Nature 2018

Abstract

We present a new multi-level image compression method based on fuzzy transforms in which the image is decomposed in levels and afterwards each level-image is compressed as well. Unlike the traditional fuzzy transform image compression method, the proposed algorithm allows to check the quality of the reconstructed image at every level. Unlike the classical image compression F-transform algorithm, our method allows to control the quality of the reconstructed image, to be used for applications in which a high quality of the decoded image is necessary. We compare our method with the single level fuzzy transform, DCT, DWT, JPEG, JPEG2K algorithms in terms of quality of the reconstructed image and CPU coding/decoding time. The results show that the CPU time obtained in our method are comparable (resp., better) with the ones obtained via DCT, JPEG, JPEG2K (resp., DWT) algorithm.

Keywords Multi-level image compression · Fuzzy partition · Fuzzy transform · PSNR

1 Introduction

Usually a lossy image compression method is used for coding/decoding images, like, for example, JPEG. These methods are generally used for inserting images in WEB pages and for capturing images by digital cameras. The advantage in terms of sizes of the image obtained with a strong compression rate is balanced by a lower quality of the decompressed image and a great loss of information with respect to the original image.

At present time, in many applications, like cloud storage private data protection management in Wang et al. (2016) or video surveillance systems using drones in Uchida et al. (2017), it is necessary to control the quality of the decoded image: generally speaking, using a strong compression rate, high-frequency components are cut in the decoded image. In addition, by setting a priori only the compression rate,

we cannot control the quality of the resulting image: for instance, the quality of two decoded images can be very different if two diverse original images are coded under the same compression rate.

We show a new lossy multi-level image compression based on the fuzzy transforms (F-tr), called as MF-transforms (MF-tr). Our aim is to control the quality of the decoded images and to optimize the trade off between the image sizes and quality. In few words, the image is decomposed into more levels disposed as hierarchical structure: each level retains a particular information's content of the original image which decreases at the successive level. This happens also in other methods like the pyramid compression method (e.g., Toet 1989; Paris et al. 2015; Boiangiu et al. 2016; Ispas and Boiangiu 2017) and the wavelet transform (e.g., Walker and Nguyen 2001; Song 2006; Mallat 2009; Chowdhury and Khathum 2012; Qureshi and Deriche 2016; Khan et al. 2017; Ahanonu et al. 2018; Karthikeyan and Palanisamy 2018).

Furthermore, we intend to improve the performances obtained with the F-tr lossy image compression method proposed in Perfilieva (2006), Di Martino and Sessa (2007), Di Martino et al. (2008) in terms of quality of the decoded image, mainly in Di Martino et al. (2008) the authors show that the quality of the images is better than the one obtained by using the fuzzy relation equation (FEQ) and Discrete Cosine Transform (DCT) algorithms, used in JPEG

✉ Ferdinando Di Martino
fdimarti@unina.it

Salvatore Sessa
sessa@unina.it

¹ Dipartimento di Architettura, Università degli Studi di Napoli Federico II, Via Toledo 402, 80134 Napoli, Italy

² Centro Interdipartimentale di ricerca A. Calza Bini, Università degli Studi di Napoli Federico II, Via Toledo 402, 80134 Napoli, Italy

technique. Here we show that the MF-tr algorithm produces better quality of the decoded images with respect to the F-tr algorithm and Discrete Wavelet Transform (DWT), used in JPEG2K and Embedded Zerotree Wavelet (EZW) techniques.

Lossy image compression methods based on the concept of direct and inverse bi-dimensional F-tr (Di Martino and Sessa 2007; Perfilieva 2007; Di Martino et al. 2008; Perfilieva and Dankova 2008; Perfilieva and de Baets 2010) has been used in many others domains such as image fusion in Di Martino and Sessa (2017), Hodakova et al. (2011), Perfilieva (2007), Perfilieva and Dankova (2008), in image segmentation in Di Martino et al. (2010a), in image reduction in Di Martino et al. (2014), in image watermarking in Di Martino et al. (2012), in video compression in Di Martino et al. (2010b). In Di Martino and Sessa (2007) and Di Martino et al. (2008) the authors show that compression/decompression of images based on F-tr method gives the best results with respect to the ones based on fuzzy relation equations in terms of quality of the image and CPU time. Furthermore, the quality of the images with the F-tr method is comparable with that one obtained using the JPEG method for low values of the compression rate.

The MF-tr is based on a multi-level decomposition of the error, like in image fusion (see.), Perfilieva (2007), Perfilieva and Dankova (2008), Hodakova et al. (2011), Di Martino and Sessa (2017). In this method the error obtained with respect to the source image is measured at every level. The process is iterated until the error is less than or equal to a

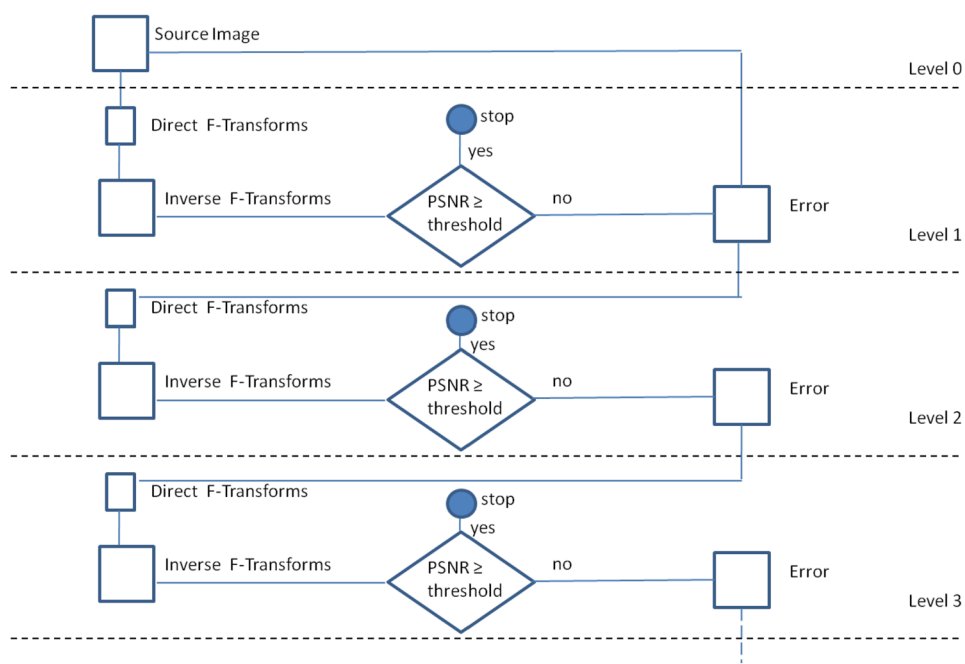
pre-fixed threshold. In Fig. 1 we show the schema of our MF-tr method.

The source image, considered as level 0, is compressed with the direct F-tr and decompressed with the inverse F-tr. The difference between the source and the decompressed image is given by the error at level (1) This process is iterated: at the next iteration the error obtained at level 1 represents the input image, then it is compressed and decompressed and the difference with the decompressed image is given by the error at level (2) The iteration stops if the quality of the reconstructed image, obtained as the sum of the inverse F-tr in every level, is greater or equal than a pre-defined value. We can set a threshold for the quality image by measuring at each decomposition level the Peak Signal to Noise Ratio (PSNR) obtained with the comparison of the reconstructed and the source images.

We adopt the F-tr compression process used in Di Martino and Sessa (2007), Di Martino et al. (2008), in which the original image is divided in blocks. As shown in Di Martino et al. (2010a, b, 2012, 2014), by dividing the image in blocks, we obtain final images with the best quality with respect to those ones obtained using the classical F-tr where the original image is not divided in blocks.

In Sect. 2 we recall the concept of F-tr, in Sect. 3 we present the F-tr image compression method, in Sect. 4 we present our method, in Sect. 5 we show the results of our tests and Sect. 6 is conclusive.

Fig. 1 MF-tr image compression schema



2 The fuzzy transform method

2.1 Fuzzy transforms in one variable

Following the definitions and notations of Perfilieva (2006), let $[a, b]$ be a closed interval, $n \geq 2$ and x_1, x_2, \dots, x_n be points of $[a, b]$, called nodes, such that $x_1 = a < x_2 < \dots < x_n = b$. We say that an assigned family of fuzzy sets $A_1, \dots, A_n: [a, b] \rightarrow [0, 1]$ is a fuzzy partition of $[a, b]$ if the following conditions hold:

- $A_i(x_i) = 1$ for every $i = 1, 2, \dots, n$;
- $A_i(x) = 0$ if x is not in (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;
- $A_i(x)$ is a continuous function on $[a, b]$;
- $A_i(x)$ strictly increases on $[x_{i-1}, x_i]$ for $i = 2, \dots, n$ and strictly decreases on $[x_i, x_{i+1}]$ for $i = 1, \dots, n - 1$;
- $\sum_{i=1}^n A_i(x) = 1$ for every $x \in [a, b]$.

The fuzzy sets $\{A_1, \dots, A_n\}$ are called basic functions. Moreover, we say that they form an uniform fuzzy partition if

- $n \geq 3$ and $x_i = a + h \cdot (i - 1)$, where $h = (b - a) / (n - 1)$ and $i = 1, 2, \dots, n$ (that is, the nodes are equidistant);
- $A_i(x_i - x) = A_i(x_i + x)$ for every $x \in [0, h]$ and $i = 2, \dots, n - 1$;
- $A_{i+1}(x) = A_i(x - h)$ for every $x \in [x_i, x_{i+1}]$ and $i = 1, 2, \dots, n - 1$.

Now we only deal with the discrete case, that is we know that the function f assumes determined values in some points p_1, \dots, p_m of $[a, b]$. We assume that the set P of these points is sufficiently dense with respect to the fixed partition, that is for each $i = 1, \dots, n$ there exists an index $j \in \{1, \dots, m\}$ such that $A_i(p_j) > 0$. Then we can define the n -tuple $\{F_1, \dots, F_n\}$ as the discrete F-tr of the function f with respect to $\{A_1, A_2, \dots, A_n\}$, where each F_i is given by:

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

for $i = 1, \dots, n$. We define the discrete inverse F-tr of the function f with respect to $\{A_1, A_2, \dots, A_n\}$ to be the following function defined in the same points p_1, \dots, p_m of $[a, b]$:

$$f_{F,n}(p_j) = \sum_{i=1}^n F_i A_i(p_j) \tag{2}$$

We have the following approximation theorem Perfilieva (2006):

Theorem 1 Let $f(x)$ be assigned on a set P of points p_1, \dots, p_m of $[a, b]$. Then for every $\epsilon > 0$, there exist an integer $n(\epsilon)$ and a related fuzzy partition $\{A_1, A_2, \dots, A_{n(\epsilon)}\}$ of $[a, b]$ such that P is sufficiently dense with respect to $\{A_1, A_2, \dots, A_{n(\epsilon)}\}$ and the inequality $|f(p_j) - f_{F,n(\epsilon)}(p_j)| < \epsilon$ holds true for every $p_j \in [a, b]$, $j = 1, \dots, m$.

2.2 Fuzzy 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 $n, m \geq 2$, $x_1, x_2, \dots, x_n \in [a, b]$ and $y_1, y_2, \dots, y_m \in [c, d]$ be $n + m$ assigned points, called nodes, such that $x_1 = a < x_2 < \dots < x_n = b$ and $y_1 = c < \dots < y_m = d$. Furthermore, let $A_1, \dots, A_n: [a, b] \rightarrow [0, 1]$ be a fuzzy partition of $[a, b]$ and $B_1, \dots, B_m: [c, d] \rightarrow [0, 1]$ be a fuzzy partition of $[c, d]$. In the discrete case, we assume that the function f assumes determined values in some points $(p_i, q_j) \in [a, b] \times [c, d]$, where $i = 1, \dots, N$ and $j = 1, \dots, M$. Moreover, the sets $P = \{p_1, \dots, p_N\}$ and $Q = \{q_1, \dots, q_M\}$ of these points are sufficiently dense with respect to the chosen partitions, that is, for each $i = 1, \dots, N$ there exists an index $k \in \{1, \dots, n\}$ such that $A_i(p_k) > 0$ and for each $j = 1, \dots, M$ there exists an index $l \in \{1, \dots, m\}$ such that $B_j(q_l) > 0$. Then we define the matrix $[F_{kl}]$ to be the discrete F-tr of f with respect to $\{A_1, \dots, A_n\}$ and $\{B_1, \dots, B_m\}$ if we have for each $k = 1, \dots, n$ and $l = 1, \dots, m$:

$$F_{kl} = \frac{\sum_{j=1}^M \sum_{i=1}^N f(p_i, q_j) A_k(p_i) B_l(q_j)}{\sum_{j=1}^M \sum_{i=1}^N A_k(p_i) B_l(q_j)} \tag{3}$$

By extending (2) to the case of two variables, we define the discrete inverse F-tr of f with respect to $\{A_1, A_2, \dots, A_n\}$ and $\{B_1, \dots, B_m\}$ to be the following function defined in the same points (p_i, q_j) in $[a, b] \times [c, d]$, with $i \in \{1, \dots, N\}$ and $j \in \{1, \dots, M\}$, as:

$$f_{nm}^F(p_i, q_j) = \sum_{k=1}^n \sum_{l=1}^m F_{kl} A_k(p_i) B_l(q_j) \tag{4}$$

It is possible to show that the following generalization of Theorem 1:

Theorem 2 Let $f(x, y)$ be known on $(p_i, q_j) \in [a, b] \times [c, d]$, $i \in \{1, \dots, N\}$, $j \in \{1, \dots, M\}$. Then for every $\epsilon > 0$, there exist two integers $n(\epsilon)$, $m(\epsilon)$ and related fuzzy partitions $\{A_1, A_2,$

..., $A_{n(\epsilon)}$ of $[a, b]$ and $\{B_1, B_2, \dots, B_{m(\epsilon)}\}$ of $[c, d]$ such that the sets of points $P = \{p_1, \dots, p_N\}$ and $Q = \{q_1, \dots, q_M\}$ are sufficiently dense with respect to $\{A_1, A_2, \dots, A_{n(\epsilon)}\}$ and $\{B_1, B_2, \dots, B_{m(\epsilon)}\}$ and the following holds true for every $(p_i, q_j) \in [a, b] \times [c, d]$, $i \in \{1, \dots, N\}$ and $j \in \{1, \dots, M\}$:

$$|f(p_i, q_j) - f_{n(\epsilon)m(\epsilon)}^F(p_i, q_j)| < \epsilon \tag{5}$$

3 F-transforms in two variables for image compression

Let I be a grey image of $M \times N$ sizes and L_t be the scale of grey levels, with $I(i, j) = P(i, j)/L_t$, seen as $I: (i, j) \in \{1, \dots, M\} \times \{1, \dots, N\} \rightarrow [0, 1]$, $I(i, j)$ being the normalized value of the pixel $P(i, j)$. For brevity, we put $p_i = i$, $q_j = j$, $a = c = 1$, $b = N$, $d = M$. We suppose that $A_1, \dots, A_m: [1, M] \rightarrow [0, 1]$ (resp., $B_1, \dots, B_n: [1, N] \rightarrow [0, 1]$) with $m < M$ (resp., $n < N$), form a fuzzy partition of $[1, M]$ (resp., $[1, N]$). Then I is divided in sub-matrices I_C of $M(C) \times N(C)$ sizes ($I_C: (i, j) \in \{1, \dots, M(C)\} \times \{1, \dots, N(C)\} \rightarrow [0, 1]$), defined as blocks compressed to blocks C of sizes $m(C) \times n(C)$ (with $m(C) < M(C)$, $n(C) < N(C)$) via the discrete F-tr $[F_{kl}^C]$ given by:

$$F_{kl}^C = \frac{\sum_{j=1}^{N(C)} \sum_{i=1}^{M(C)} I_C(i, j) A_k(i) B_l(j)}{\sum_{j=1}^{N(C)} \sum_{i=1}^{M(C)} A_k(i) B_l(j)} \tag{6}$$

for each $k = 1, \dots, m(C)$ and $l = 1, \dots, n(C)$. As above, naturally we do in such a way that the set $\{1, \dots, M(C)$ (resp., $\{1, \dots, N(C)\})$ is sufficiently dense to the fuzzy partition $\{A_1, \dots, A_{m(C)}\}$ (resp., $\{B_1, \dots, B_{n(C)}\})$ defined in $[1, M(C)]$ (resp., $[1, N(C)]$). Then we decode the blocks with the inverse F-tr $I_{m(C)n(C)}^F: \{1, \dots, M(C)\} \times \{1, \dots, N(C)\} \rightarrow [0, 1]$ defined as:

$$I_{m(C)n(C)}^F(i, j) = \sum_{l=1}^{n(C)} \sum_{k=1}^{m(C)} F_{kl}^C A_k(i) B_l(j) \tag{7}$$

which approximates I_C with arbitrary precision in the sense of Theorem 2, that is there exist certainly two integers $n(C) = n(C, \epsilon)$, $m(C) = m(C, \epsilon)$ and $\epsilon > 0$ for every block C such that the inequality

$$|I_C(i, j) - I_{m(C)n(C)}^F(i, j)| < \epsilon \tag{8}$$

holds true for every $(i, j) \in \{1, \dots, M(C)\} \times \{1, \dots, N(C)\}$. Practically speaking, we assign several values to $n(C)$ and $m(C)$ with $m(C) < M(C)$, $n(C) < N(C)$ and hence to the compression rate $\rho(C) = (m(C) \cdot n(C))/(M(C) \cdot N(C))$. Here we use (cfr., [6, 7, 8]) the fuzzy sets $A_1, \dots, A_{m(C)}: [1, M(C)] \diamond [0, 1]$ and $B_1, \dots, B_{n(C)}: [1, N(C)] \diamond [0, 1]$ defined as

$$A_1(i) = \begin{cases} 0.5 \left(\cos \frac{\pi}{h} (i - 1) + 1 \right) & \text{if } i \in [1, x_2] \\ 0 & \text{otherwise} \end{cases}$$

$$A_k(i) = \begin{cases} 0.5 \left(\cos \frac{\pi}{h} (i - x_k) + 1 \right) & \text{if } i \in [x_{k-1}, x_{k+1}] \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$A_{m(C)}(i) = \begin{cases} 0.5 \left(\cos \frac{\pi}{h} (i - x_{m(C)-1}) + 1 \right) & \text{if } i \in [x_{m(C)-1}, M(C)] \\ 0 & \text{otherwise} \end{cases}$$

where $k = 2, \dots, m(C)$, $h = (M(C) - 1)/(m(C) - 1)$, $x_k = 1 + h \cdot (k - 1)$ and

$$B_1(j) = \begin{cases} 0.5 \left(\cos \frac{\pi}{s} (j - 1) + 1 \right) & \text{if } j \in [1, y_2] \\ 0 & \text{otherwise} \end{cases}$$

$$B_t(j) = \begin{cases} 0.5 \left(\cos \frac{\pi}{s} (j - y_t) + 1 \right) & \text{if } j \in [y_{t-1}, y_{t+1}] \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

$$B_{n(C)}(j) = \begin{cases} 0.5 \left(\cos \frac{\pi}{s} (j - y_{n(C)-1}) + 1 \right) & \text{if } j \in [y_{n(C)-1}, N(C)] \\ 0 & \text{otherwise} \end{cases}$$

where $t = 2, \dots, n(C)$, $s = (N(C) - 1)/(n(C) - 1)$, $y_t = 1 + s \cdot (t - 1)$.

4 The MF-transform image process

We consider a grey-level image I of sizes $N \times M$ at level 0. In order to control the quality of the reconstructed image with respect to the original one, we put a threshold value for the PSNR index defined as

$$PSNR = 20 \log_{10} \frac{Lt}{MSE} \tag{11}$$

where MSE stands for the Mean Square Error defined as

$$MSE = \frac{\sum_{i=1}^N \sum_{j=1}^M (I^{(h)}(i,j) - I_0(i,j))^2}{N \times M} \tag{12}$$

Lt can assume at most the value 255 in an 8-bit grey pixel depth image. If $I = I^{(0)}$ is the original image, $I^{(h)}$ is the decoded image, obtained at the h th level, given as:

$$I^{(h)} = \begin{cases} I^{(1)} & \text{if } h = 1 \\ \sum_{r=1}^{h-1} I_F^{(r)} = I^{(h-1)} + I_F^{(h)} & \text{if } h > 1 \end{cases} \tag{13}$$

where $I_F^{(h)}$ is the inverse F-tr calculated at the h th level. We obtain $I_F^{(h-1)} = I^{(h-1)} - I^{(h-2)}$ if $h \geq 2$ and $I_F^{(1)} = I^{(1)} - I^{(0)}$. Figure 2 schematizes the reconstruction process.

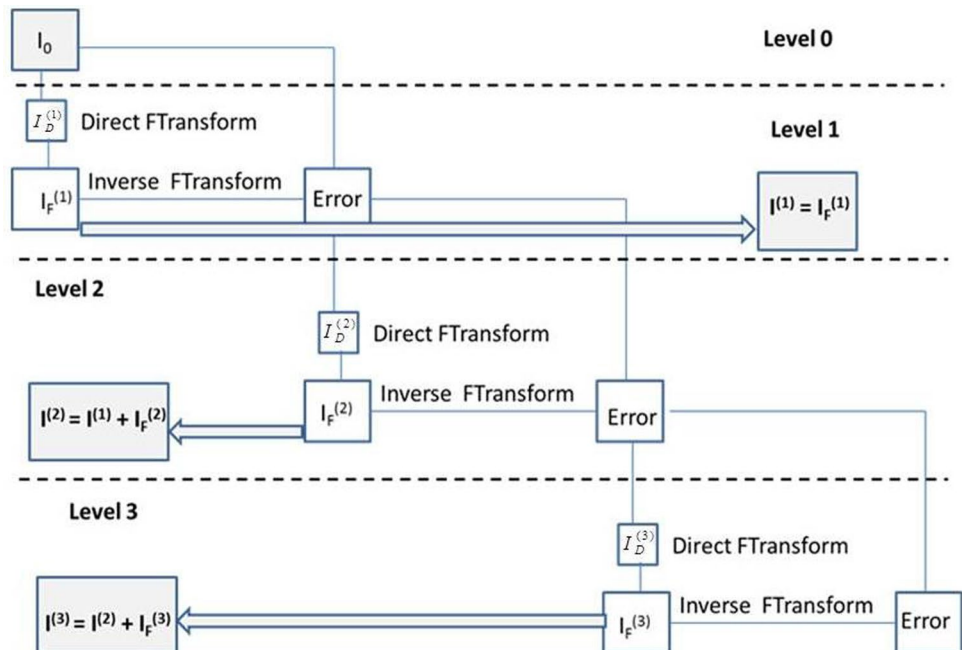
If the PSNR calculated at each level is greater or equal than the threshold, the process stops otherwise we

consider the next level. We use the F-tr compression process described in Sect. 2.2, dividing the image in blocks of dimensions $n \times m$. Each block is compressed via (6) and decompressed via (7). The inverse F-tr $I_F^{(h)}$ at the h th level is obtained by merging all the decompressed blocks. In our algorithm the process stops when at least one of the next three conditions is true:

- the PSNR value is greater than a threshold $PSNR_{th}$;
- the difference between the PSNR at the h th level and the PSNR at the $(h-1)$ th level is less than a difference threshold $DIFFPSNR_{th}$;
- the current level reaches a maximal level $hmax$.

We add the second condition because the $DIFFPSNR_{th}$ value decreases by increasing the h th level. Hence it is reasonable to stop the process when the image quality obtained at a certain level differs in minimum quantity if compared to that one obtained at the $(h-1)$ th level. The third condition is added in order to stop the process after $hmax$ iterations if the first two conditions are false. As schematized in the coding/decoding pseudocode below, the conditional statement in the step 15) is necessary because in the first iteration ($h = 1$) the value of the variable $DIFFPSNR$ must be set to a value greater than the difference threshold $DIFFPSNR_{th}$. We denote with $I_D^{(1)} \dots I_D^{(h)}$ the direct F-tr obtained at each level.

Fig. 2 Schema of the images reconstructed at each level



MF-transforms coding pseudocode

- 1) Set the thresholds $PSNR_{th}$, the $DIFFPSNR_{th}$ and h_{max}
- 2) $I = I_0$
- 3) $h = 1$
- 4) $R = I / Lt$ // normalization of I
- 5) Divide the $N \times M$ normalized image R in blocks C of dimension $n \times m$.
- 6) For each block C
 - a. compute the direct F-tr of I_C using the basic functions (9) and (10)
 - b. compute the inverse F-tr
- 7) Reconstruct the direct F-tr $I_D^{(h)}$ joining all the direct F-tr of all blocks C
- 8) Reconstruct $I_F^{(h)}$ joining all the inverse F-tr of all blocks C
- 9) $I_F^{(h)} =: It * I_F^{(h)}$ // de-normalization of $I_F^{(h)}$
- 10) Error = $I - I_F^{(h)}$
- 11) Compute $I^{(h)}$ using (13)
- 12) Compute PSNR using (11)
- 13) IF ($h=1$) $DIFFPSNR = DIFFPSNR_{th} + 1$
- 14) ELSE $DIFFPSNR = PSNR - PSNR_0$
- 15) IF ($PSNR < PSNR_{th}$) AND ($DIFFPSNR > DIFFPSNR_{th}$) AND ($h < h_{max}$)
 - a. $I = Error$
 - b. $h = h + 1$
 - c. $PSNR_0 = PSNR$
 - d. Go to 4
- 16) ELSE
 - RETURN $h, I_D^{(1)} \dots I_D^{(h)}$
- 17) ENDIF

MF-transforms decoding pseudocode

- 1) $I_F = 0$
- 2) FOR $s = 1$ to h
- 3) compute the inverse F-tr $I_F^{(s)}$ of $I_D^{(s)}$
- 4) $I_F^{(s)} =: It * I_F^{(s)}$ // de-normalization of $I_F^{(s)}$
- 5) $I_F = I_F + I_F^{(s)}$
- 6) NEXT s
- 7) RETURN I_F

5 Test results

For our tests we have considered a sample of grey images taken from the USC-SIPI Image Database (<http://sipi.usc.edu/database/>). We perform our tests by using a Intel Core i7-59360X processor with a clock frequency of 3 GHz.

For measuring the performances of the MF-tr algorithm, we compare it with the classical F-tr, FEQ, DCT, DWT, JPEG, JPEG2K algorithms. In these comparison we measure the PSNR and the CPU time obtained applying each algorithm. For brevity, we only give the results for three images of this sample. In the first experiment the grey image Lena of sizes 256×256 (Fig. 3a) is divided in blocks of sizes 4×4 compressed in blocks of sizes 2×2 ($\rho = 0.25$). At the 1st level the PSNR is 28.410 and we set $PSNR_{th} = 30$, $DIFFPSNR_{th} = 0.1$. Thus both thresholds are reached at the 5th level (Fig. 3b). In Fig. 3c, e, g, i (resp., d, f, h, l) we show

the decoded images (rep., corresponding errors) obtained at the 1st, 2nd, 3rd, 4th level.

In Table 1 we show the PSNR obtained for each level. The column $DIFFPSNR$ shows the difference between the PSNRs obtained at the h th and $(h-1)$ th levels.

In the next experiment the grey image Leopard of sizes 256×256 (Fig. 4a) is divided in blocks of sizes 4×4 compressed in blocks of sizes 2×2 ($\rho = 0.25$). At the 1st level the PSNR is 24.675 and we set $PSNR_{th} = 26$, $DIFFPSNR_{th} = 0.1$. Thus both thresholds are reached at the 3rd level (Fig. 4b). In Fig. 4a we show the original image. In Fig. 4b we show the resulting image obtained at the 2nd level. Fig. 4c, e (resp., d, f) show the decoded images (resp., corresponding errors) obtained at the 1st, 2nd level.

In Table 2 we show the PSNR obtained for each level. The column $DIFFPSNR$ shows the difference between the PSNRs obtained at the h th and $(h-1)$ th levels.

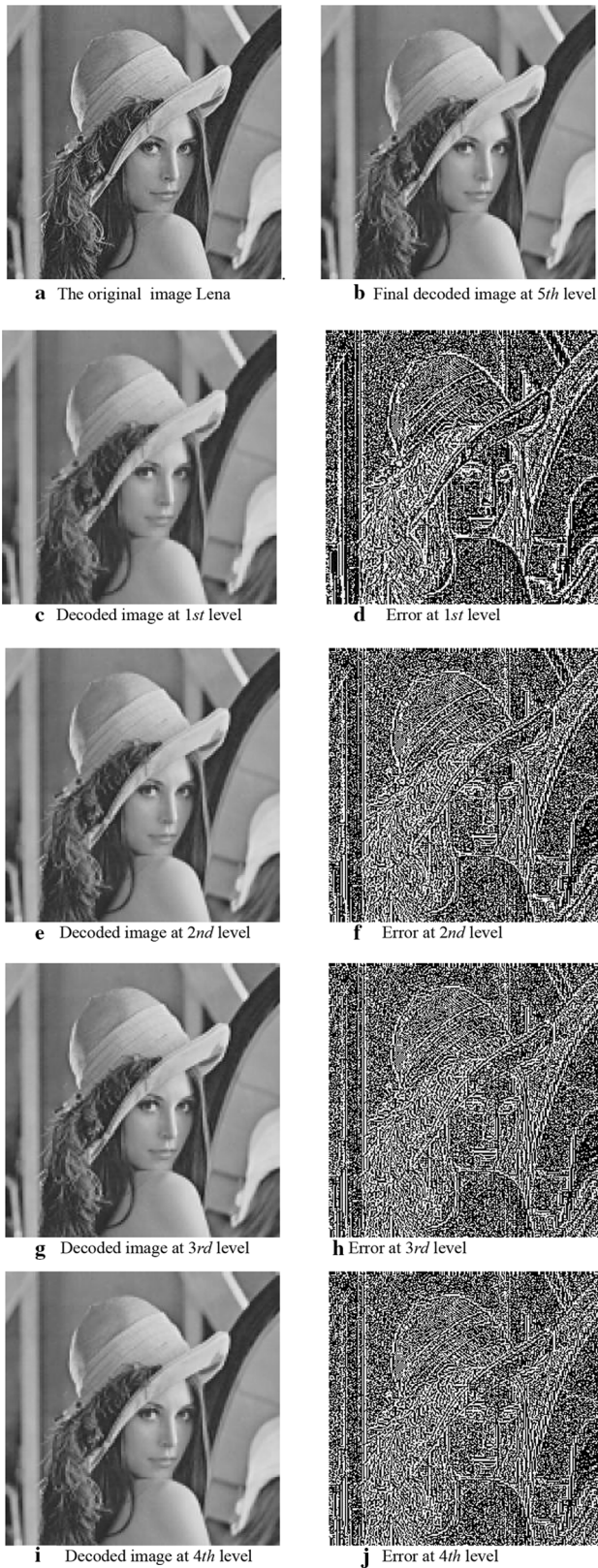


Fig. 3 **a** The original image Lena, **b**. Final decoded image at 5th level. **c**. Decoded image at 1st level. **d**. Error at 1st level. **e**. Decoded image at 2nd level. **f**. Error at 2nd level. **g**. Decoded image at 3rd level. **h**. Error at 3rd level. **i**. Decoded image at 4th level. **j**. Error at 4th level

Table 1 PSNR obtained at each level for the image Lena ($\rho=0.25$)

Level	PSNR	DIFFPSNR
1	28.410	–
2	29.238	0.828
3	29.681	0.443
4	29.935	0.254
5	30.024	0.089

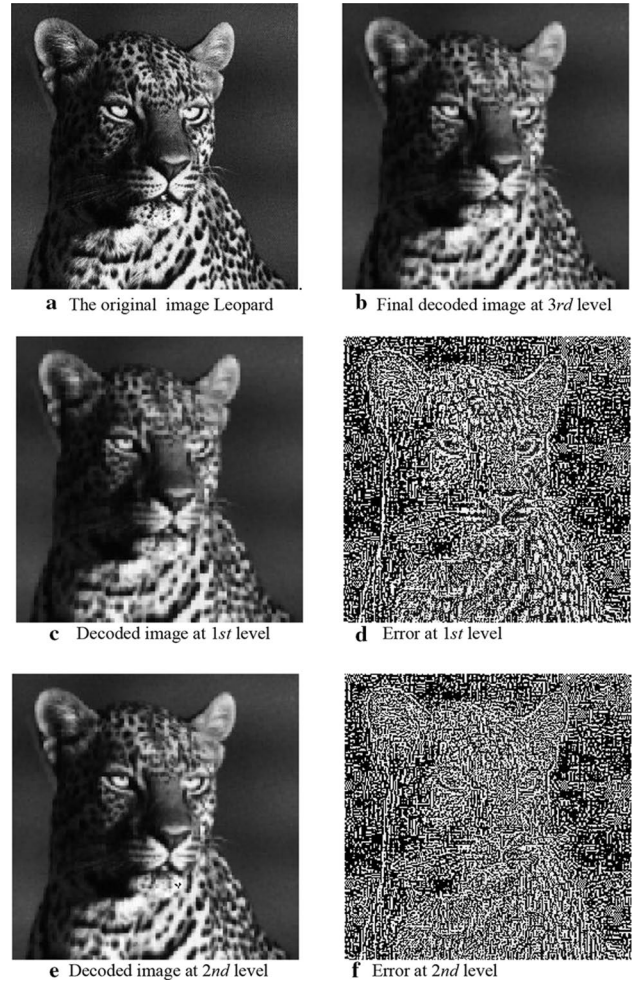


Fig. 4 **a** The original image Leopard. **b** Final decoded image at 3rd level. **c** Decoded image at 1st level. **d** Error at 1st level. **e** Decoded image at 2nd level. **f** Error at 2nd level

Table 2 PSNR obtained at each level for the image Leopard ($\rho=0.25$)

Level	PSNR	DIFFPSNR
1	24.675	–
2	25.709	1.034
3	26.181	0.472

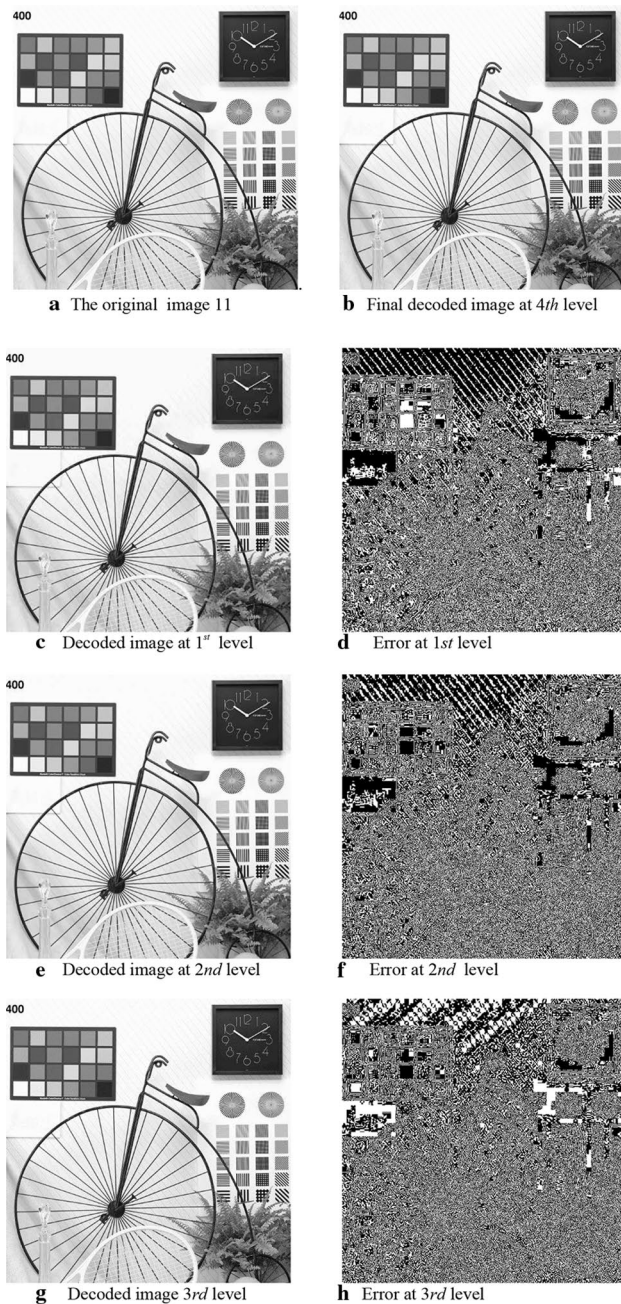


Fig. 5 **a** The original image 11. **b** Final decoded image at 4th level. **c** Decoded image at 1st level. **d** Error at 1st level. **e** Decoded image at 2nd level. **f** Error at 2nd level. **g** Decoded image 3rd level. **h** Error at 3rd level

In the next experiment the grey image “11” of sizes 512×512 (Fig. 5a) is divided in blocks of sizes 4×4 compressed in blocks of sizes 2×2 ($\rho = 0.25$). At the 1st level the PSNR is 30.412. We set $PSNR_{th} = 32$ and $DIFFPSNR_{th} = 0.1$. These thresholds are reached at the 3rd level. In Fig. 5a we show the original image. In Fig. 5b we show the decoded image obtained at the

Table 3 PSNR obtained at each level for the image “11” ($\rho = 0.25$)

Level	PSNR	Diff PSNR
1	28.615	–
2	29.779	1.164
3	30.244	0.465
4	30.341	0.097

Table 4 CPU time in ms obtained for the images Lena, Leopard and “11”

Image	Dimensions of the image	CPU time in F-tr	Final level in MF-tr	CPU time in MF-tr
Lena	256×256	17.43	5	48.00
Leopard	256×256	17.83	3	34.18
11	512×512	14.61	4	54.75

Table 5 Mean CPU time varying the final level for images of sizes 256×256

Final level (MF-tr)	Mean CPU time (MF-tr)	Average CPU time MF-tr/number of levels
2	25.87	12.94
3	32.89	10.96
4	40.79	10.20
5	48.00	9.60
6	55.64	9.27
7	64.77	9.25
8	73.71	9.21

Table 6 Mean CPU time varying the final level for images of sizes 512×512

Final level (MF-tr)	Mean CPU time (MF-tr)	Average CPU time MF-tr/number of levels
2	35.06	17.53
3	46.91	15.64
4	60.18	15.05
5	73.95	14.79
6	87.31	14.55
7	101.47	14.50
8	115.33	14.42

4th level. Fig. 5c, e, g (resp., d, f, h) show the decoded images (resp., corresponding errors) obtained at the 1st, 2nd, 3rd level.

In Table 3 we show the PSNR obtained for each level. The column DIFFPSNR shows the difference between the PSNRs obtained at the h th and $(h-1)$ th levels. Table 4

Table 7 Mean CPU time varying the final level for images of sizes 1024×1024

Final level (MF-tr)	Mean CPU time (MF-tr)	Average CPU time MF-tr/number of levels
2	45.17	22.59
3	61.02	20.34
4	79.58	19.90
5	98.32	19.66
6	114.38	19.06
7	131.65	18.81
8	150.01	18.75

shows the CPU time required with the F-tr and MF-tr methods for decoding the original images.

In Table 5 (resp., 6, 7) we show the mean CPU time obtained for all 256×256 (resp., 512×512 , 1024×1024) grey images of the above database by varying the final level. In the 3rd column we show the ratio between the total CPU time and the final level.

Tables 5, 6 and 7 show that this ratio decreases by increasing the final level. Indeed the results show the CPU mean time is 0.5 s (resp., 1 s) for images of sizes 256×256

(resp., 512×512 and 1024×1024) by performing a decomposition up to the eighth level. In other words, the MF-tr method can be considered a good compromise between the quality of the reconstructed image and the time necessary for its reconstruction.

By sake of completeness, comparison experiments are made with other image compression methods for measuring the performances of the MF-tr algorithm in terms of quality of the reconstructed image by means of the PSNR. Indeed we consider the grey levels of the error images at each level and apply the Huffman encoding to maximize the compression. For brevity, we show the PSNR obtained for the 256×256 grey image Lena in Table 8. The second column shows the dimension in bytes of the direct F-tr at any level, the third column shows the dimension in bytes of the sum of all the direct F-tr until to the l th level. The compression rate ρ is given by the ratio between the number of bytes of the sum of the directed F-tr and the number of bytes necessary to store the original images. The 4th, 5th, 6th columns show the PSNR in the MF-tr, JPEG and JPEG2K methods, respectively and in the 8th (resp., 9th) the percentage gain PSNR of the JPEG (resp., JPEG2K) with respect to the MF-tr, respectively defined as:

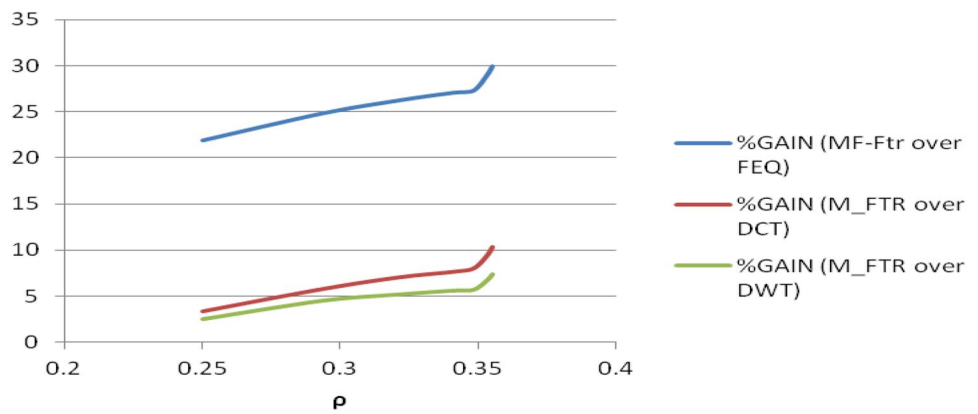
Table 8 PSNR comparisons for the image Lena (MF-tr, JPEG and JPEG2K)

Level	Bytes directed F-tr	Bytes sum of directed F-tr	ρ	PSNR MF-tr	PSNR JPEG	PSNR JPEG2K	%GAIN JPEG over MF-tr	%GAIN JPEG2K over MF-tr
1	16,384	16,384	0.250	28.410	36.371	37.388	28.022	31.602
2	2826	19,210	0.293	29.238	36.380	37.421	24.427	27.988
3	1843	21,053	0.321	29.681	36.412	37.462	22.678	26.215
4	1290	22,344	0.341	29.935	36.477	37.488	21.854	25.231
5	492	22,835	0.348	30.024	36.503	37.517	21.579	24.957
6	328	23,163	0.353	30.453	36.565	37.530	20.070	23.239
7	123	23,286	0.355	30.703	36.569	37.536	19.128	22.278

Table 9 PSNR comparison and MF-tr gain with respect FEQ, DCT and DWT for image Lena

Level	ρ	PSNR(MF-transforms)	PSNR (FEQ)	PSNR (DCT)	PSNR (DWT)	%GAIN (MF-Ftr over FEQ)	%GAIN (M_FTR over DCT)	%GAIN (M_FTR over DWT)
1	0.250	28.410	23.298	27.487	27.868	21.940	3.358	2.312
2	0.293	29.238	23.429	27.643	28.001	24.794	5.770	4.418
3	0.321	29.681	23.511	27.726	28.212	26.243	7.051	5.207
4	0.341	29.935	23.578	27.798	28.455	26.962	7.688	5.684
5	0.348	30.024	23.603	27.815	28.516	27.204	7.942	5.703
6	0.353	30.453	23.617	27.827	29.001	28.945	9.437	6.789
7	0.355	30.703	23.628	27.836	29.302	29.943	10.300	7.436

Fig. 6 Trend of the gain percentages of MF-tr with respect to FEQ, DCT, DWT



$\%Gain (JPEG \text{ over MF-tr}) = (PSNR \text{ JPEG} - PSNR \text{ MF-tr}) / PSNR \text{ MF-tr}$.

$\%Gain (JPEG2K \text{ over MF-tr}) = (PSNR \text{ JPEG2K} - PSNR \text{ MF-tr}) / PSNR \text{ MF-tr}$.

These results show that the PSNR gain percentage obtained by using the JPEG and JPEG2K algorithms with respect to the MF-tr algorithm decreases by increasing the levels. This trend is shown also for the other grey images in the dataset. The results in Table 9 show that the gain percentage of PSNR, obtained by using the MF-tr algorithm with respect to FEQ, DCT, DWT algorithms, increases by increasing the compression rate, that is increasing the levels considered in the MF-Ftr algorithm. This trend is shown also for the other grey images in the dataset. In the FEQ compression method we used the Lukasiewicz t-norm and the related residuum operators. In the DCT algorithm the image is partitioned in 8×8 size blocks. In the DWT algorithm at each level the HH, LH and HL coefficients are discarded, whereas the LL coefficients are transformed into the successive level. The best results are obtained by using the MF-tr algorithm. The DWT algorithm gives better results with respect to the FEQ and DCT algorithms. The gain percentage of PSNR for the image Lena are defined as.

$\%Gain (MF-tr \text{ over FEQ}) = (PSNR \text{ MF-tr} - PSNR \text{ FEQ}) / PSNR \text{ FEQ}$.

$\%Gain (MF-tr \text{ over DCT}) = (PSNR \text{ MF-tr} - PSNR \text{ DCT}) / PSNR \text{ DCT}$.

$\%Gain (MF-tr \text{ over DWT}) = (PSNR \text{ MF-tr} - PSNR \text{ DWT}) / PSNR \text{ DWT}$.

In Fig. 6 we show the mean trends of the three gain percentages by varying the compression rate.

In Tables 10 and 11 we show the mean coding and decoding CPU times, respectively, obtained for the 256×256 grey images in the dataset applying the MF-tr, F-tr, FEQ, DCT and DWT algorithms.

These results show that the CPU time obtained by using the MF-tr algorithm are averagely acceptable. Both the mean coding and decoding CPU times calculated by using

the MF-tr algorithm are better than the corresponding ones in the DWT algorithm, independently from the compression rate.

Figures 7 and 8 show the trend of the coding and the decoding CPU time, respectively, calculated in seconds by using the MF-tr and DWT algorithm. These trends highlight the benefits in terms of CPU time in the coding and decoding processes of the MF-tr algorithm with respect to the DWT algorithm, regardless of the compression rate.

Table 10 Mean coding CPU time for 256×256 grey images in several methods

rho	CODING CPU TIME (ms)				
	MF-tr	F-tr	FEQ	DCT	DWT
0.250000	6.09	6.09	58.11	6.21	11.45
0.293125	11.42	6.10	58.03	6.24	23.02
0.321250	16.88	6.12	57.73	6.25	37.30
0.340938	22.31	6.13	57.39	6.27	49.27
0.348438	27.69	6.16	56.98	6.30	63.31
0.353438	33.98	6.17	56.77	6.38	73.54
0.355313	42.37	6.20	56.34	6.42	85.71

Table 11 Mean decoding CPU time for 256×256 grey images in several methods

rho	DECODING CPU TIME (ms)				
	MF-tr	F-tr	FEQ	DCT	DWT
0.250000	11.34	11.34	11.15	5.19	26.15
0.293125	14.45	11.41	11.36	5.25	26.44
0.321250	16.01	11.45	11.78	5.38	26.92
0.340938	18.48	11.50	12.35	5.47	27.34
0.348438	20.31	11.56	12.61	5.53	27.76
0.353438	21.66	11.62	12.78	5.60	28.01
0.355313	22.40	11.67	12.89	5.68	28.62

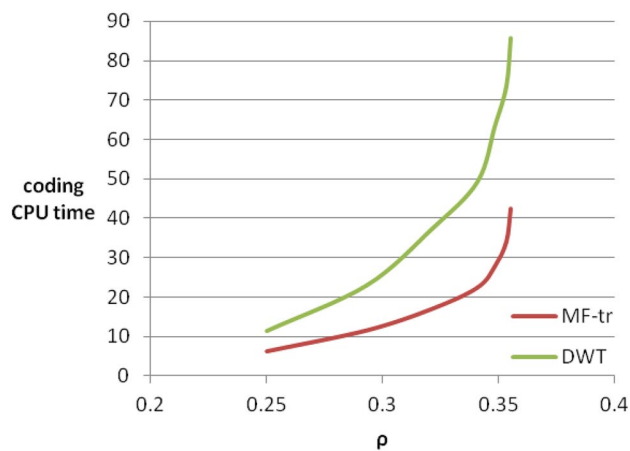


Fig. 7 Trend of the coding CPU time of MF-tr (in red) with respect to DWT (in green)

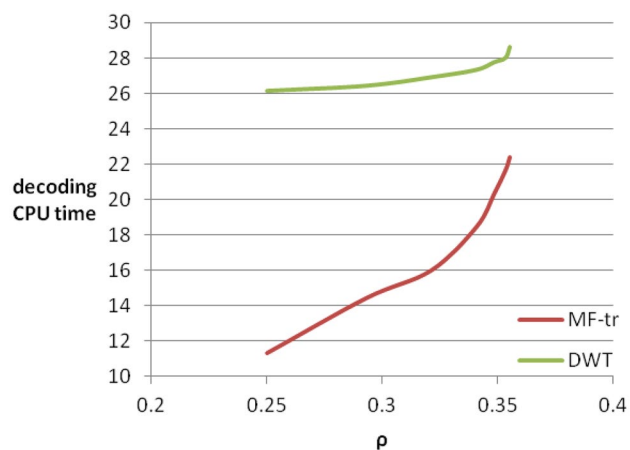


Fig. 8 Trend of the decoding CPU time of MF-tr (in red) with respect to DWT (in green)

6 Conclusions

Lossy image compression algorithms are used in many disciplines in which a loss of information in the reconstructed image is considered acceptable. The drawback of the F-tr method consists in the fact that they do not allow to set in advance the quality of the reconstructed image, but only its compression rate. The MF-tr algorithm is a multi-level compression image technique that uses the F-tr algorithm for coding and decoding the input image at each level. The user can set a PSNR threshold for controlling the quality of the reconstructed image. This characteristic makes the MF-tr method usable also in the cases where it is necessary to guarantee a high quality of the reconstructed image, such as, for example, in video surveillance or environmental control systems.

The results show that the CPU time (necessary for reconstructing the final image) is acceptable and the coding/decoding time decreases by increasing the number of the levels. Moreover, comparisons show that the PSNR gain percentage obtained by using the MF-tr algorithm is better than the corresponding ones obtained by using F-tr, FEQ, DCT, DWT algorithms. The quality of the decoded image is not comparable with the ones obtained by using the JPEG and JPEG2K techniques, but the PSNR gain percentage of the JPEG and JPEG2K techniques diminishes by increasing the number of levels. In future researches the quality of the decoded image will be improved by integrating the MF-tr algorithm with specific quantization and entropy encoding algorithms, moreover such method shall be dedicated to other topics like segmentation and image fusion. In the future we intend to build a fragile watermarking algorithm based on the MF-tr method in order to apply it in a cloud storage environment to protect the privacy of sensitive images. This fragile watermarking system will be based on the schema proposed in Di Martino et al. (2010a) in which we apply the MF-tr method controls the quality of the marked stored images.

Acknowledgements This work was written under the auspices of INDAM-GCNS (Italy).

References

- Ahanonu E, Marcellin M, Bilgin A (2018) Lossless image compression using reversible integer wavelet transforms and convolutional neural networks 2018 Data Compression Conference, Snowbird, pp. 395–395. <https://doi.org/10.1109/DCC.2018.00048>
- Boiangiu CA, Cotofana MV, Naiman A, Lamburu C (2016) A generalized Laplacian pyramid aimed at image compression. *J Inf Syst Oper Manag* 10(2):327–335
- Chowdhury MMH, Khatun A (2012) Image compression using discrete wavelet transform. *Int J Comput Sci Issues* 9(4):327–330
- Di Martino F, Sessa S (2007) Compression and decompression of images with discrete fuzzy transforms. *Inf Sci* 177:2349–2362
- Di Martino F, Sessa S (2017) Complete image fusion method based on fuzzy transforms. *Soft Comput.* <https://doi.org/10.1007/s00500-017-2929-4>.
- Di Martino F, Loia V, Perfilieva I, Sessa S (2008) An image coding/decoding method based on direct and inverse fuzzy transforms. *Int J Approx Reasoning* 48(1):110–131
- Di Martino F, Loia V, Sessa S (2010a) A segmentation method for images compressed by fuzzy transforms. *Fuzzy Sets Syst* 161:56–74
- Di Martino F, Loia V, Sessa S (2010b) Fuzzy transforms for compression and decompression of colour videos. *Inf Sci* 180:3914–3931
- Di Martino F, Loia V, Sessa S (2012) Fragile watermarking tamper detection with images compressed by fuzzy transform. *Inf Sci* 195:62–90
- Di Martino F, Hurtik P, Perfilieva I, Sessa S (2014) A color image reduction based on fuzzy transforms. *Inf Sci* 266:101–111
- Hodakova P, Perfilieva I, Dankova M, Vajgl M (2011) F-transform based image fusion. In: Ukimura O (ed) *Image fusion*. InTech, Rijeka, pp. 3–22

- Ispas C, Boiangiu CA (2017) An image compression scheme based on Laplacian Pyramid. *J Inf Syst Oper Manag* 11(2):350–358
- Karthikeyan C, Palanisamy C (2018) An efficient image compression method by using optimized discrete wavelet transform and Huffman encoder. *J Comput Theor Nanosci* 15(1):289–298
- Khan UR, Ahmed S, Nazeer T (2017) Wavelet based image compression techniques: comparative analysis and performance evaluation. *Int J Emerg Technol Eng Res* 5(9):9–13
- Mallat S (2009) *A wavelet tour of signal processing: the sparse way*, 3 Edn. Academic Press, Burlington
- Paris S, Hasinoff SV, Kautz J (2015) Local Laplacian filters: edge-aware image processing with a Laplacian pyramid. *Commun ACM CACM* 58(3):81–91
- Perfilevia I (2006) Fuzzy transforms. *Fuzzy Sets Syst* 157:993–1023
- Perfilevia I (2007) Fuzzy transform in image compression and fusion. *Acta Math Univ Ostrav* 15:27–37
- Perfilevia I, Dankova M (2008) Image fusion on the basis of fuzzy transforms. *Proceedings of the 8th International FLINS Conference on Computational Intelligence in Decision and Control, Madrid*, pp. 471–476
- Perfilevia I, De Baets B (2010) Fuzzy transforms of monotone functions with application to image compression. *Inf Sci* 180:3304–3315
- Qureshi MA, Deriche M (2016) A new wavelet based efficient image compression algorithm using compressive sensing. *Multimed Tools Appl* 75:6737–6754
- Song M-S (2006) Wavelet image compression. *Contemp Math* 414:41–73
- Toet A (1989) A morphological pyramidal image decomposition. *Pattern Recogn Lett* 9(4):255–261
- Uchida N, Okutake T, Yamamoto N (2017) Image recognitions of collaborative drones' security controls for FPV systems. *Int J Space Based Situated Comput* 7(3):129–135
- Walker JS, Nguyen TQ (2001) Wavelet-based image compression (Chap. 6). In: Rao KR et al (ed) *The transform and data compression handbook*. CRC Press LLC, Boca Raton
- Wang Y, Du y., Cheng X, Liu Z, Lin K (2016) Degradation and encryption for outsourced PNG images in cloud storage. *Int J Grid Utility Comput* 7(1):22–28

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.