Article

Fuzzy Transform Image Compression in the YUV Space

Barbara Cardone , Ferdinando Di Martino * , Salvatore Sessa *

Posted Date: 14 September 2023

doi: 10.20944/preprints202309.0964.v1

Keywords: F-transform; F1-transform; color image compression; RGB space; YUV space.

Article

Fuzzy Transform Image Compression in the YUV Space

Barbara Cardone [1], Ferdinando Di Martino [1,2,*] and Salvatore Sessa [1,2]

[1] Dipartimento di Architettura, Università degli Studi di Napoli Federico II, Via Toledo 402, 80134 Napoli, Italy; sessa@unina.it

[2] Centro Interdipartimentale di Ricerca "Alberto Calza Bini",Università degli Studi di Napoli Federico II, Via Toledo 402, 80134 Napoli, Italy

* Correspondence: fdimarti@unina.it

Abstract: The two-dimensional Fuzzy Transform was applied in image compression. The quality of the image compressed with this methodis better than that obtained using methods based on fuzzy relation equations and comparable with that of the JPEG method, with better compression execution times. In this paper we propose a variation of the method based on the two-dimensional Fuzzy Transform of image compression, in which the image is first partitioned into blocks and then each block is compressed with the higher compression rate. The advantage of this method consists into a greater compression of the image, guaranteeing a high quality of the reconstructed image. The results show that our method is better than the Fuzzy Transform method improving also the quality of the reconstructed image.

Keywords: F-transform; F[1]-transform; color image compression; RGB space; YUV space

1. Introduction

YUV is a color model used in the NTSC, PAL, and SECAM color encoding systems., describing the color space in terms of a brightness component (the Y band called *luma*) and the two chrominance components (the U and V bands called *chroma*).

The YUV model has been used to advantage in image processing; its main advantage is that, unlike the three Red, Green and Blue (RGB) bands, which are equally perceived by the human eye, in YUV space, most of the color image information is contained in the Y band, as opposed to the U and V bands. main applications of the YUV model in image processing, is then the lossy compression of images, which can be performed mainly in the U and V bands, with slight loss of information.

YUV is used in the JPEG color image compression method [1,2] where the Discrete Cousine Transform (DCT) algorithm is executed on the YUV space, sub-sampling and reducing in dynamic range the UV channels in order to balance the reduction of data and the feel of human eyes. In [3] the DCT algorithm is executed in the YUV space for wireless capsule endoscopy application; the results show that the quality of the reconstructed images is better than that obtained by applying the DCT image compression method in the RGB space.

Many authors proposed image compression and reconstruction algorithms applied on the YUV space in order to improve the quality of the reconstructed images.

In [4,5] an image compression algorithm based on fuzzy relation equations is applied in the YUV space to compress color images; the image is divided in blocks of equal sizes, coding the blocks in the UV channels more strongly than the blocks in the Y band. In [6] the Fuzzy Transform technique (for short, F-transform) [7,8] is applied to coding color images in the YUV space; the author show that the quality of color images coded and decoded via F-transform in the YUV space is better than the one obtained using the F-transform method in the RGB space and comparable with the one obtained using the JPEG method.

A fractal image compression technique applied on the YUV space is proposed in [9]; the authors show that the quality of color images coded/decoded using this approach is better than the one obtained applying the fractal image compression method in the RGB space.

 doi:10.20944/preprints202309.0964.v1

Furthermore, comparison tests between RGB and YUV perception-oriented properties performed in [10] show that compressed images in the YUV space provide better quality than images compressed in the RGB spaces in a human-computer interaction and machine vision applications.

In [11] is applied a technique using Tchebichef bit allocation to compress images in the YUV space; the results shown that this method improves the visual quality of color images compressed via JPEG by 42%. A color image compression method applying a subsampling process to the two chroma channels and a modification algorithm to the Y channel is applied to color images in [12] to improve the JPEG performances.

An image compression method with learning-base filter is applied in [13] on color images constructing the filter in YUV space instead of RGB space; author shows that the quality of the coded images is better than the one obtained using the filter in the RGB space. In [14] an image lossy compression algorithm in which quantization and subsampling are executed in the YUV space is applied for wireless capsule endoscopy; the quality of the coded images results better than the one obtained executing quantization and subsampling in the RGB space. In [15] a wavelet-based color image compression method using trained convolutional neural network are used in the lifting scheme is applied on the YUV executing the trained CNN on the Y, U, V channels separately; this method improves the quality of the coded images obtained applying traditional wavelet-based color image compression algorithms; however, execution times are much higher than those adopted by applying traditional color image compression algorithms.

In [16] an image reconstruction method performed on the YUV space is applied to prevent from corruption of data performed using adversarial perturbation of the image; the results show that the image can be recovered on the YUV space without distortions and with a high visual quality.

In this paper we propose a novel image compression algorithm in which the bidimensional First-Degree F-Transform algorithm [17,18] is applied for coding/decoding color images in the YUV space.

A generalization of the F-transform, called high order F^m-transform, has been proposed in [19] in order to reduce the approximation error of the original function approximated with the inverse F-transform. In the F^m -transform the components of the direct high order fuzzy transforms consist of polynomials of degree s, unlike the components of the direct F-transform (labelled as F^0-transform), where they were constant values. The greater the degree of the polynomial, the smaller the error of the approximation; however, as the degree of the polynomial increases, the computational complexity of the algorithm increases.

In [18] the bi-dimensional first-order degree F-transform (F^1-transform) is used to compress images; authors show that the quality of the coded/decoded images is better than the one obtained executing F-transform, with negligible augments in CPU time. The critical point of this method consists of the fact that, unlike the F-transform and JPEG methods, it requires not the compressed image to be saved in memory, but matrices of three coefficients of the same size as the compressed image; therefore, it needs a memory area three times greater than that necessary to archive the compressed image.

To solve this problem, we propose a new lossy color image compression algorithm in which is executed the F^1-transform algorithm to code/decode color images transformed in the YUV space. The transformed image in each of the three channels is partitioned into blocks and each block is compressed by the bi-dimensional direct F^1-transform, compressing the blocks of the chroma channels more. The image is subsequently reconstructed by decomposing the single blocks with the use of the bi-dimensional inverse F^1-transform.

The main benefits of this method are as follows:

- the use of the bi-dimensional F^1-transform represents a trade-off between the quality of the compressed image and the CPU times. It reduces the information loss obtained by compressing the image with the same compression rate using the F-transform algorithm with acceptable coding/decoding CPU time;
- the compression of the color images is carried out in the YUV space to guarantee a high visual quality of the color images and solve the criticality of the F^1-transform color image compression method on the RGB space [18] which needs a larger memory to allocate the information of the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 September 2023 doi:10.20944/preprints202309.0964.v1

compressed image. In fact, by performing a high compression of the two chrominance channels, the size of the matrices in which the information of the compressed image is contained, is reduced in these two channels, and this allows to reduce the memory allocation and CPU times.

We compare our color lossy image compression method with the JPEG method and with the image compression methods based on the bi-dimensional F-transform [7,8] and F^1-transform [19] on the RGB space and on the bi-dimensional F-transform on the YUV space [6].

In the next Section the concepts of F-transform and F^1-transform are briefly presented, and the F-transform lossy color image compression method applied in YUV space is shown as well. Our method is presented in Section 3. In Section 4 the comparative results obtained on datasets of color images are shown and discussed; concluding discussions are contained in Section 5.

2. Preliminaries

2.1. The bi-dimensional F-Transform

Let [a,b] be a closed real interval and let $\{x_1, x_2, \ldots, x_n\}$ be a set of points of [a,b], called *nodes*, such that $x_1 = a < x_2 < \ldots < x_n = b$.

Let $\{A_1,\ldots,A_n\}$ be a family of fuzzy sets of X, where $A_i : [a,b] \rightarrow [0,1$: it forms a *fuzzy partition* of X if the following conditions hold:

(1) $A_i(x_i) = 1$ for every i =1,2,…,n;

(2) $A_i(x) = 0$ if $x \notin (x_{i-1}, x_{i+1})$, by setting $x_0 = x_1 = a$ and $x_{n+1} = x_n = b$;

(3) $A_i(x)$ is a continuous function over [a,b];

(4) $A_i(x)$ is strictly increasing over $[x_{i-1}, x_i]$ for each i = 2, …, n ;

(5) $A_i(x)$ is strictly decreasing over $[x_i, x_{i+1}]$ for each i = 1,…, n-1;

(6) $\sum_{i=1}^{n} A_i(x) = 1$ for every $x \in [a,b]$.

Let $h = \frac{b-a}{n-1}$. The fuzzy partition $\{A_1,\ldots, A_n\}$ is an *uniform fuzzy partition* if:

(7) $n \geq 3$;

(8) $x_i = a+h\cdot(i-1)$, for every i = 1, 2, …, n;

(9) $A_i(x_i - x) = A_i(x_i + x)$ for every $x \in [0,h]$ and i = 2,…, n-1;

(10) $A_{i+1}(x) = A_i(x - h)$ for every $x \in [x_i, x_{i+1}]$ and i = 1,2,…, n-1.

Let $f(x)$ be a continuous function over [a,b] and $\{A_1, A_2, \ldots, A_n\}$ be a fuzzy partition of [a,b]. The n-tuple $F = [F_1, F_2, \ldots, F_n]$ is called *unidimensional direct F-transform* of f with respect to $\{A_1, A_2, \ldots, A_n\}$ if the following hold:

$$F_i = \frac{\int_a^b f(x)A_i(x)dx}{\int_a^b A_i(x)dx} \qquad i=1,2,\ldots,n \tag{1}$$

The following function $f_{F,n}$ defined in [a,b]:

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

is called *uni-dimensional inverse F-transform* of the function f.

The following theorem holds (cfr. [7, Theorem 2]):

Theorem 1. *Let $f(x)$ be a continuous function over [a,b]. For every $\varepsilon > 0$ there exist an integer $n(\varepsilon)$ and a fuzzy partition $\{A_1, A_2, \ldots, A_{n(\varepsilon)}\}$ of [a,b] such that for all $x \in [a, b]$ $\left|f(x) - f_{F,n(\varepsilon)}(x) < \epsilon\right|$.*

Now consider the discrete case where the function f is known in a set of N points $P = \{p_1,\ldots,p_N\}$, where $p_j \in [a,b]$ j = 1,2,…,m. The set $\{p_1,\ldots,p_N\}$ is called *sufficiently dense* with respect to the fixed fuzzy partition $\{A_1, A_2, \ldots, A_n\}$ if for every i = 1,…,n, there exists at least an index $j \in \{1,\ldots,m\}$ such that $A_i(p_j) > 0$.

If the set P is sufficiently dense with respect to the fuzzy partition, we can define the *discrete direct F-transform* with components:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 September 2023 doi:10.20944/preprints202309.0964.v1

$$F_i = \frac{\sum_{j=1}^{N} f(p_j)A_i(p_j)}{\sum_{j=1}^{N} A_i(p_j)} \qquad i=1,2,\ldots,n \tag{3}$$

and the *discrete inverse F-transform*:

$$f_{F,n}(p_j) = \sum_{i=1}^{n} F_i\, A_i(p_j) \quad j=1,\ldots,N \tag{4}$$

The following theorem applied to the discrete inverse F-transform holds (cfr. [7, Theorem 5]):

Theorem 2. *Let f(x) be a continuous function over [a,b], known in a discrete set of points {p1,...,pm}. For every ε > 0 there exist an integer n(ε) and a fuzzy partition {A1, A2, …, An(ε)} of [a,b], with respect to which P is sufficiently dense, that is such that for every j = 1,…,N,* $|f(p_j) - f_{F,n(\varepsilon)}(p_j) < \epsilon|$.

By Theorem 2, the inverse fuzzy transform (4) can be used to approximate the function *f* in a point.

Now we consider functions in two variables. Let $x_1, x_2, \ldots, x_n$ be a set of n nodes in [a,b] , where $n > 2$ and $x_1 = a < x_2 < \ldots < x_n = b$, and let $y_1, y_2, \ldots, y_m$ be a set of n nodes in [c,d], where $m > 2$ and and $y_1 = c < y_2 < \ldots < y_m = d$. Moreover, let $A_1,\ldots,A_n : [a,b] \to [0,1]$ be a fuzzy partition of [a,b], $B_1,\ldots,B_m : [c,d] \to [0,1]$ be a fuzzy partition of [c,d] and let *f*(x,y) be a function defined in the closed set [a,b]×[c,d].

We suppose that *f* assumes known values in a set of points $(p_i,q_j) \in [a,b]\times[c,d]$, where $i = 1,\ldots,N$ and $j = 1,\ldots,M$, where the set $P=\{p_1, \ldots , p_N\}$ is sufficiently dense with respect to the fuzzy partition $\{A_1,\ldots, A_n\}$ and the set $Q=\{q_1, \ldots ,q_M\}$ is sufficiently dense with respect to the fuzzy partition $\{B_1,\ldots,B_m\}$.

In this case, we can define the *bi-dimensional discrete F-transform* of *f*, given by matrix $[F_{hk}]$ $h = 1,\ldots,n$ and $k = 1,\ldots,m$ with components:

$$F_{hk} = \frac{\sum_{j=1}^{M}\sum_{i=1}^{N} f(p_i,q_j)A_h(p_i)B_k(q_j)}{\sum_{j=1}^{M}\sum_{i=1}^{N} A_h(p_i)B_k(q_j)} \quad h=1,2,\ldots,n \quad k=1,2,\ldots,m \tag{5}$$

and the *bi-dimensional discrete inverse F-transform* of *f* with respect to $\{A_1, A_2, \ldots, A_n\}$ and $\{B_1,\ldots,B_m\}$ given by

$$f_{nm}^{F}(p_i, q_j) = \sum_{h=1}^{n}\sum_{k=1}^{m} F_{hk}\, A_h(p_i)\, B_k(q_j), \quad i=1,2,\ldots,N, \quad j=1,2,\ldots,M \tag{6}$$

2.2. The bi-dimensional F¹-Transform

This paragraph introduces the concept of higher degree fuzzy transform or F^r-transform. One-dimensional square-integrable functions will now be considered.

Let A_h, $h = 1,\ldots,n$, be the h*th* fuzzy set of the fuzzy partition $\{A_1,\ldots,A_n\}$ defined on [a,b] and $L_2([x_{h-1},x_{h+1}])$ be the Hilbert space of square-integrable functions f,g: $[x_{h-1},x_{h+1}] \longrightarrow R$ with the inner product:

$$\langle f, g\rangle_h = \frac{\int_{x_{h-1}}^{x_{h+1}} f(x)g(x)A_h(x)dx}{\int_{x_{h-1}}^{x_{h+1}} A_h(x)dx} \tag{7}$$

Given a positive integer r, we denote with $L_2^r([x_{h-1},x_{h+1}])$ a linear subspace of the Hilbert space $L_2([x_{h-1},x_{h+1}])$ having as orthogonal basis the polynomials $\{P_h^0, P_h^1, \ldots, P_h^r\}$ constructed applying the Gram-Schmidt ortho-normalization defined as:

$$\begin{cases} P_h^0 = 1 \\ P_h^{s+1} = x^{s+1} - \sum_{j=1}^{s} \dfrac{\langle x^{s+1}, P_h^j\rangle}{\langle P_h^j, P_h^j\rangle} \end{cases} \quad s=1,\ldots,r-1 \tag{8}$$

The following Lemma holds (Cfr. 7, Lemma 1):

Lemma 1. *Let* F_k^r *be the orthogonal projection of the function f on* $L_2^r([x_{h-1},x_{h+1}])$. *Then:*

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 September 2023 doi:10.20944/preprints202309.0964.v1

$$F_h^r(x) = \sum_{s=1}^{r} c_{h,i} P_h^s(x) \tag{9}$$

where

$$c_{h,s} = \frac{\langle f, P_{kh}^s \rangle_k}{\langle P_h^s, P_h^s \rangle_h} = \frac{\int_{x_{k-1}}^{x_{k+1}} f(x)\, P_h^s(x) A_h(x) dx}{\int_{x_{k-1}}^{x_{k+1}} (P_h^s(x))^2 A_h(x) dx} \tag{10}$$

F_h^r it is the h*th* component of the direct F^r-transform of f. The inverse F^r-transform of f in a point x ∈ [a,b] is:

$$f_{F,n}^r(x) = \sum_{k=1}^{n} F_h^r A_k(x) \tag{11}$$

For r = 0, we have $P_h^0 = 1$ and the F^0-transform is given by the F-transform in one variable ($F_h^0(x) = c_{h,0}$).

For r = 1, we have $F_h^1(x) = (x - x_h)$ and the h*th* component of the F^1-transform is given by the formula:

$$F_h^1(x) = c_{h,0} + c_{h,1}(x - x_h) = F_h^0(x) + c_{h,1}(x - x_h) \tag{12}$$

If the function f is known in a set of N points P ={$p_1,\ldots p_N$}, $c_{h,0}$ and $c_{h,1}$ can be discretized in the form:

$$c_{h,0} = \frac{\sum_{i=1}^{n} f(p_i) A_h(p_i)}{\sum_{i=1}^{n} A_h(p_i)} \tag{13}$$

$$c_{h,1} = \frac{\sum_{i=1}^{n} f(p_i)(p_i - x_h) A_h(p_i)}{\sum_{i=1}^{n} A_h(p_i)(p_i - x_h)^2} \tag{14}$$

The F^1-tranform can be extended in a bi-dimensional space. Let L_2 ([x_{h-1}, x_{h+1}] × [y_{k-1}, y_{k+1}]) be the Hilbert space of square- integrable functions f: [x_{h-1}, x_{h+1}] × [y_{k-1},y_{k+1}]→ R with the weighted inner product:

$$\langle f, g \rangle_{hk} = \int_{x_{h-1}}^{x_{h+1}} \int_{y_{k-1}}^{y_{k+1}} f(x,y) g(x,y) A_h(x) B_k(y) dx \ldots dy \tag{15}$$

Two functions $f, g \in L_2$ ([x_{h-1}, x_{h+1}] × [y_{k-1}, y_{k+1}]) are orthogonal if $\langle f, g \rangle_{hk} = 0$.

Let f: X ⊆ R^2 → Y⊆ R be a continuous bi-dimensional function defined in a closed set [a,b] × [c,d]. Let {A_1, A_2,…, A_n} be a fuzzy partition of [a,b], and let {B_1, B_2,…, B_m} be a fuzzy partition of [c,d].

Moreover, let {(p_1,q_1),…, (p_N,q_{jN})} a set of N points in which is known the function f, where (p_i,q_j)∈ [a,b]×[c,d]. Let the set P={p_1, … , p_N} be sufficiently dense with respect to the fuzzy partition {A_1,…, A_n} and let the set Q={q_1, … ,q_M} be sufficiently dense with respect to the fuzzy partition {B_1,…,B_m}.

We can define the bi-dimensional direct F^1-transform of f, with components:

$$F_{hk}^1(x,y) = c_{hk}^{00} + c_{hk}^{10}(x - x_h) + c_{hk}^{01}(y - y_k) \tag{16}$$

where c_{hk}^{00} is the component F_{hk} of the bi-dimensional discrete direct F transform of f, given by (5). The three coefficients in (17) are given by:

$$c_{hk}^{00} = F_{hk} = \frac{\sum_{j=1}^{N} f(p_j, q_j) \cdot A_h(p_j) \cdot B_k(q_j)}{\sum_{j=1}^{N} A_h(p_j) \cdot B_k(q_j)} \tag{17}$$

$$c_{hk}^{10} = \frac{\sum_{j=1}^{N} f(p_j, q_j) \cdot (p_j - x_h) \cdot A_h(p_j) \cdot B_k(q_j)}{\sum_{j=1}^{N} (p_j - x_h)^2 \cdot A_h(p_j) \cdot B_k(q_j)} \tag{18}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 September 2023 doi:10.20944/preprints202309.0964.v1

$$c_{hk}^{01} = \frac{\sum_{j=1}^{N} f(p_j, q_j) \cdot (q_j - y_k) \cdot A_h(p_j) \cdot B_k(q_j)}{\sum_{j=1}^{N} (q_j - y_k)^2 \cdot A_h(p_j) \cdot B_k(q_j)} \quad (19)$$

The inverse F¹-transform of f in a point $(x,y) \in [a,b] \times [c,d]$ is:

$$f_{F,n}^{1}(x,y) = \sum_{h=1}^{n} \sum_{k=1}^{m} F_{h,k}^{1} A_h(x) B_k(y) \quad (20)$$

where $F_{h,k}^{1}(x,y)$ is the (h,k)*th* component of the bi-dimensional direct F¹-transform, given by the formula (16).

2.3. Coding/decoding images using the bi-dimensional F and F¹-Transforms

Let I be a grey N × M image. A pixel can be considered a data point with coordinates (i,j), where i = 1,2,…,N and j = 1,2,…M; the value of this data point is given by the pixel value I(i,j). In [8] the image is normalized in [0,1] according with the formula R(i,j) = I(i,j)/(L-1) where L is the number of grey levels, partitioned in blocks of equal sizes N(B)×M(B), coded to a block F_B of sizes n(B) × m(B) with n(B) << N(B) and m(B) << M(B), using the bi-dimensional direct F-transform.

Let $\{A_1,\ldots,A_{n(B)}\}$ be a fuzzy partition of [1,N(B)] and let $\{B_1,\ldots,B_{m(B)}\}$ be a fuzzy partition of [1,M(B))], each block is compressed by the bi-dimensional direct F-transform:

$$F_{hk}^{B} = \frac{\sum_{j=1}^{M(B)} \sum_{i=1}^{N(B)} R(i,j) A_h(i) B_k(j)}{\sum_{j=1}^{M(B)} \sum_{i=1}^{N(B)} A_h(i) B_k(j)} \quad (21)$$

The coded image is reconstructed by merging all compressed blocks.

Each block is decompressed by using the bi-dimensional inverse F-transform. The pixel value I(i,j) in the block is approximated by the value:

$$f_{n_B m_B}^{F^B}(i,j) = \sum_{h=1}^{n(B)} \sum_{k=1}^{m(B)} F_{hk}^{B} A_h(i) B_k(j) \quad (22)$$

The decoded image is reconstructed by merging the decompressed blocks.

The F-transform compression and decompression algorithms are shown in pseudocode in Algorithms 1a and 1b.

Algorithm 1a. F¹-transform image compression

Input: N×M Image I with L grey levels
Size of the blocks of the source image N(B)×M(B)
Size of the compressed blocks n(B)× m(B)

Output: n×m compressed image I_C

1. Normalize the source image I in [0,1]
2. Partition the source image in blocks of size N(B)×M(B)
3. **For** each block
4. **For** h = 1 to n(B)
5. **For** k = 1 to m(B)
6. Compute the (hk)*th* component of the bidimensional direct F-transform by (21)
7. **Next** k
8. **Next** h
9. **Next** block
10. Merge the compressed blocks
11. De-normalize the image
12. **Return** the compressed n×m image I_C

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 September 2023 doi:10.20944/preprints202309.0964.v1

Algorithm 1b. F-transform image decompression

Input: n×m compressed image I_c
Output: N×M decoded image I_D

1. Normalize the compressed image in [0,1]
2. Partition the compressed image I_c in blocks of size n(B)×m(B)
3. **For** each compressed block
4. **For** i = 1 to N(B)
5. **For** j = 1 to M(B)
6. Compute the (i,j)*th* pixel of the decoded block by the bidimensional inverse F-transform (22)
7. **Next** j
8. **Next** i
9. **Next** compressed block
10. Merge the decompressed blocks
11. De-normalize the decompressed image
12. **Return** the decompressed N×M image I_D

In [17] an improvement of the quality of the decompressed image is accomplished using the bi-dimensional F^1-transform.

The blocks are compressed by using the bi-dimensional direct F^1-transform:

$$F_{hk}^{1B} = c_{hk}^{00} + c_{hk}^{10}(i-h) + c_{hk}^{01}(j-k) \tag{23}$$

where:

$$c_{hk}^{00} = F_{hk}^{B} = \frac{\sum_{j=1}^{M(B)}\sum_{i=1}^{N(B)} R(i,j)A_h(i)B_k(j)}{\sum_{j=1}^{M(B)}\sum_{i=1}^{N(B)} A_h(i)B_k(j)} \tag{24}$$

$$c_{hk}^{10} == \frac{\sum_{j=1}^{M(B)}\sum_{i=1}^{N(B)} I(i,j)|i-h|A_h(i)B_k(j)}{\sum_{i=1}^{N(B)} A_h(i)(i-h)^2 \sum_{j=1}^{M(B)} B_k(j)} \tag{25}$$

$$c_{hk}^{01} == \frac{\sum_{j=1}^{M(B)}\sum_{i=1}^{N(B)} I(i,j)|j-k|A_h(i)B_k(j)}{\sum_{j=1}^{M(B)} B_k(j)(j-k)^2 \sum_{i=1}^{N(B)} A_h(i)} \tag{26}$$

The above three coefficients are constructed merging the coefficients of each block and finally stored, forming the output of coding process.

In the inverse process the image is reconstructed by decompressing the block with the bi-dimensional inverse F^1-transform:

$$f_{n_B m_B}^{1F^B}(i,j) = \sum_{h=1}^{n(B)}\sum_{k=1}^{m(B)} F_{hk}^{B} A_h(i)\, B_k(j) \tag{27}$$

where the bi-dimensional direct F^1-transform of the block F_{hk}^{B} is calculated by (23).

The decompressed blocks are merged to form the decompressed image.

The F^1-transform compression and decompression algorithms are shown in pseudocode in Algorithms 2a and 2b.

Algorithm 2a. F^1-transform image compression

Input: N×M Image I with L grey levels
Size of the blocks of the source image N(B)×M(B)
Size of the compressed blocks n(B)× m(B)
Output: n×m matrices of the direct F^1-transform coefficients c^{00}, c^{10} and c^{01}

1. Normalize the source image I in [0,1]
2. Partition the source image in blocks of size N(B)×M(B)
3. **For** each block
4. **For** h = 1 to n(B)
5. **For** k = 1 to m(B)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 September 2023 doi:10.20944/preprints202309.0964.v1

Compute the component c_{hk}^{00} by (24)
Compute the component c_{hk}^{10} by (25)
Compute the component c_{hk}^{01} by (26)
Compute the (hk)*th* component of the bidimensional direct F¹-transform by (26)
Next k
Next h
Next block
Merge the compressed blocks to obtain the n×m matrices of the coefficients c^{00}, c^{10} and c^{01}
Return the compressed n×m matrices of the coefficients c^{00}, c^{10} and c^{01}

Algorithm 2b. F¹-transform image decompression

Input: n×m matrices of the direct F¹-transform coefficients coefficients c^{00}, c^{10} and c^{01}
Size of the blocks of the decoded image N(B)×M(B)
Size of the blocks of the coded image n(B)×m(B)
Output: N×M decoded image I_D

Partition the F¹-transform coefficients c^{00}, c^{10} and c^{01} in blocks of size n(B)×m(B)
For each compressed block
For i = 1 to N(B)
For j = 1 to M(B)
Compute the (i,j)*th* pixel of the decoded block by the bidimensional inverse F¹-transform (27)
Next j
Next i
Next compressed block
Merge the decompressed blocks
De-normalize the decompressed image
Return the decompressed N×M image I_D

3. The YUV-based F¹-transform color image compression method

Let I be a N×M color image in L grey levels. All pixel values in the three bands R, G and B are normalized in [0,1].

Considering a 256 grey levels color image and the scaled and offset version of the YUV color space, the source image is transformed in the YUV space via the formula [20]:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.332 & 0.500 \\ 0.500 & -0.419 & -0.813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \quad (28)$$

Then, the F¹-transform image compression algorithm is executed separately to the three normalized images Y, U and V, using a strong compression for the chroma images U and V.

If N(B) and M(B) are the sizes of each block in the three channels, the blocks in the brightness channel are compressed with rate $\varrho_Y = \frac{n_Y(B)\times m_Y(B)}{N(B)\times M(B)}$ and the blocks in the two chroma channels are compressed with rate $\varrho_{UV} = \frac{n_{UV}(B)\times m_{UV}(B)}{N(B)\times M(B)}$, where $n_{UV}(B) << n_Y(B)$ and $m_{UV}(B) << m_Y(B)$, so that $\varrho_{UV} << \varrho_Y$.

The F¹-transform image compression algorithm will store, in output for each channel, the three matrixes of the coefficients of the bi-dimensional direct F¹-transform: c^{00}, c^{10} and c^{01}. The size of the three matrices in the brightness channel is $\varrho_Y(N\times M)$ and the size of the three matrices in each of the two chroma channels is $\varrho_{UV}(N\times M)$.

By suitably choosing the brightness and chroma compression rates, it is possible to reduce the memory capacity needed to store the direct F¹-transform coefficients in the RGB space.

For example, suppose we execute the F¹-transform image compression algorithm in the RGB space to compress a 256×256 color image, by partitioning the image into 16x16 blocks compressed into 4×4 blocks. The compression rate will be $\varrho_{RGB} = 0.0625$ and the size of the matrix of each coefficient is 64×64. Executing the F¹-transform algorithm in the YUV space and compressing the 16×16 blocks in the two chroma channels in 2×2 blocks ($\varrho_{UV} = 0.016$) and the 16×16 blocks in the brightness channel

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 September 2023 doi:10.20944/preprints202309.0964.v1

in 8×8 blocks (ϱ_{UV} = 0.25), the size of the matrix of each coefficient in the U and V channels will be 32×32, and the size of the matrix of each coefficient in the Y channel will be 128×128. So, by carrying out the compression of the source image in the YUV space in this way, an advantage is obtained both in terms of visual quality of the reconstructed image and in terms of available memory necessary to archive the coefficients of the direct F¹-transforms in the three channels.

Below we show in pseudocode the YUV F¹-transform color image compression algorithm (Algorithm 3a).

Algorithm 3a. YUV F¹-transform color image compression

Input: N×M color image I with L grey levels
Size of the blocks of the source image N(B)×M(B)
Size of the compressed blocks in the Y channel n_Y(B)× m_Y(B)
Size of the compressed blocks in the U and V channels n_{UV}(B)×m_{UV}(B)

Output: n×m matrices of thedirect F¹-transform coefficients c^{00}, c^{10} and c^{01} in the Y, U and V channels

1. Extract the single band images I_R, I_G and I_B
2. Transform the RGB images I_R, I_G and I_B in the YUV images I_Y, I_U and I_V by (28)
3. **Execute** F¹-transform image compression (I_Y, N(B), M(B), n_Y(B), m_Y(B)) //compress I_Y
4. **Execute** F¹-transform image compression (I_U, N(B), M(B), n_{UV}(B), m_{UV}(B)) //compress I_U
5. **Execute** F¹-transform image compression (I_V, N(B), M(B), n_{UV}(B), m_{UV}(B)) //compress I_V
6. **Return** the compressed matrices of the coefficients c^{00}, c^{10} and c^{01} in the bands Y, U and V

The decompression process is performed by executing the F¹-transform image decompression algorithm in the brightness and chroma channels. The F¹-transform image decompression algorithm is executed separately for each of the channels Y, U and V, by assigning as input both the three coefficient matrices of the direct F¹-transform and the dimensions of the original and compressed blocks.

Then the three decoded images I_{DY}, I_{DU}, and I_{DV} are transformed in the RGB space, by the formula [20]:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.164 & 0 & 1.596 \\ 1.164 & -0.813 & -0.392 \\ 1.164 & 2.017 & 0 \end{bmatrix} \begin{bmatrix} Y - 16 \\ U - 128 \\ V - 128 \end{bmatrix} \quad (29)$$

Finally, the decoded image in the RGB band (I_{DR}, I_{DG}, I_{DB}) is returned.

Below is shown in pseudocode the YUV F¹-transform color image decompression algorithm (Algorithm 3b).

Algorithm 3b. YUV F¹-transform image decompression

Input: n×m matrices of the direct F¹-transform coefficients coefficients c^{00}, c^{10} and c^{01} in the Y, U and V channels
Size of the blocks of the decoded image N(B)×M(B)
Size of the compressed blocks in the Y channel n_Y(B)× m_Y(B)
Size of the compressed blocks in the U and V channels n_{UV}(B)×m_{UV}(B)

Output: N×M decoded image I_D

1. c^{00}, c^{10} and c^{01} in blocks of size n_Y(B)×m_Y(B)
2. I_{DY} = F¹-transform image decompression (c_Y^{00}, c_Y^{10}, c_Y^{01}, N(B), M(B), n_Y(B), m_Y(B)) //Y ch. decomp.
3. I_{DU} = F¹-transform image decompression (c_U^{00}, c_U^{10}, c_U^{01}, N(B), M(B), n_{UV}(B), m_{UV}(B)) //U ch. decomp.
4. I_{DV} = F¹-transform image decompression (c_V^{00}, c_V^{10}, c_V^{01}, N(B), M(B), n_{UV}(B), m_{UV}(B)) //V ch. decomp.
5. Transform the YUV images I_{DY}, I_{DU} and I_{DV} in the RGB images I_{DR}, I_{DG} and I_{DB} by (29)
6. **Return** the decompressed N×M color image in the RGB space (I_{DR}, I_{DG} and I_{DB})

We compare our lossy color image compression report with the JPEG algorithms and the color image compression methods based on F-transform in the RGB space [8], and F-transform on the YUV space [6] and F¹-transform on the RGB space [17].

The Peak Signal to Noise index (PSNR) used to measure the quality of the decoded images. To measure the gain obtained executing the YUV F¹-transform algorithm with respect to another color

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 September 2023 doi:10.20944/preprints202309.0964.v1

image compression method, we measure the PSNR gain, expressed in percentage and given by the formula

$$\text{Gain}(\text{YUV } F^1 - \text{transform}) = \frac{[(\text{PSNR YUV } F^1 - \text{transform}) - (\text{PSNR other method})] \cdot 100}{(\text{PSNR other method})} \quad (30)$$

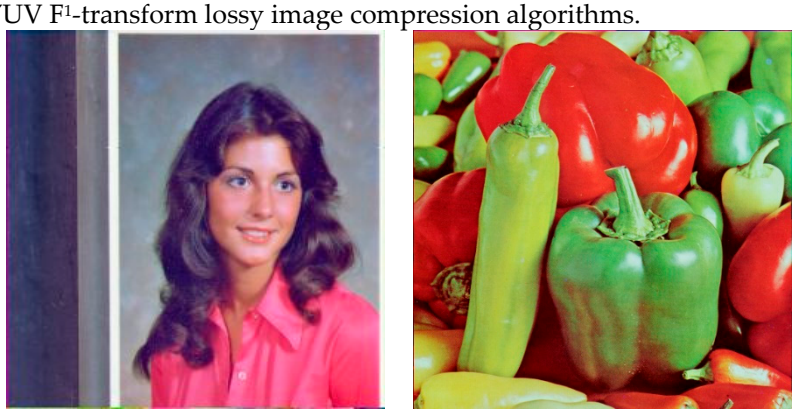
In next Section the results applied to color image dataset are shown and discussed.

4. Results

We test the YUV F1-transform lossy color image compression algorithm on the color image dataset provided by the University of Southern California Signal and Image Processing Institute (USC SIPI) and published on the website http://sipi.usc.edu/database.

The dataset is made up of over 50 color images of different sizes. For brevity, we show in detail the results obtained for the 256x256 source images 4.1.04 and the 412x512 source image 4.2.07; they are shown in Figure 1.

Each image was compressed and decompressed by performing JPEG, YUV F-transform, F¹-transform and YUV F¹-transform lossy image compression algorithms.



(**a**) (**b**)

Figure 1. Source images: (a) 256x256 image 4.1.04; (b): 512x512 image 4.2.07.

We compare the four image compression methods measuring the quality of the reconstructed image as the compression rate change. The compression rate used by executing YUV F-transform and YUV F¹-transform is the mean compression rate set for each channel Y, U and V.

In Figure 2 we show, for the original image 4.1.04, the decoded images obtained by executing the four algorithms with a compression rate $\varrho \approx 0.10$.

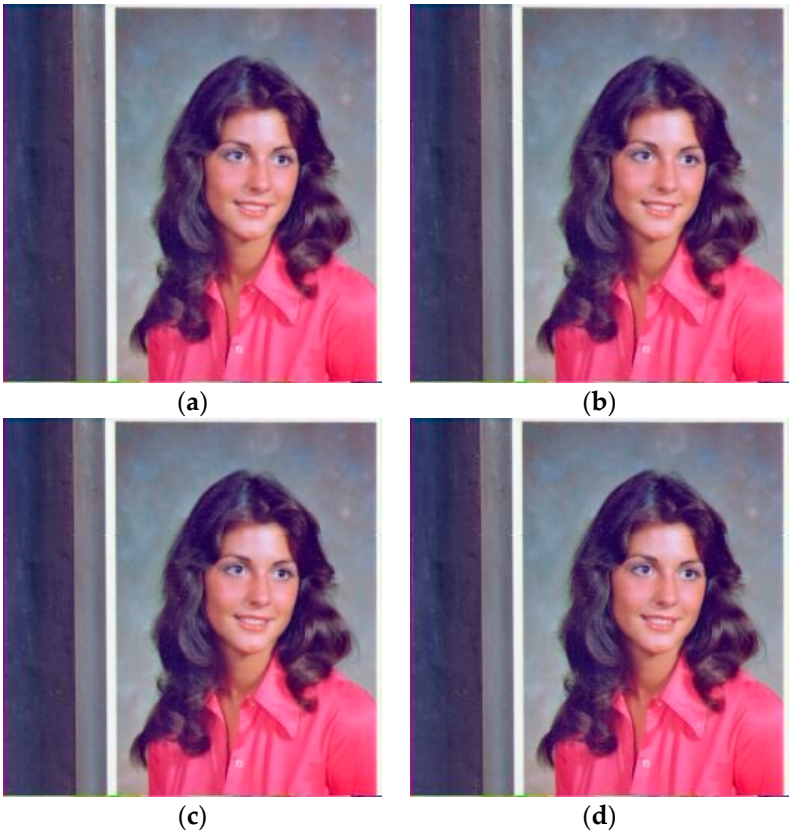
(**a**) (**b**)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 September 2023 doi:10.20944/preprints202309.0964.v1

(c) (d)

Figure 2. Decoded image 4.1.04, $\varrho \approx 0.10$, obtained via: (a) JPEG; (b) F^1-transform; (c):YUV F-transform; (d) YUV F^1-transform.

Figure 3 shows, for the original image 4.1.04, the decoded images obtained by executing the four algorithms setting a compression rate $\varrho \approx 0.25$.



(a) (b)

(c) (d)

Figure 3. Decoded image 4.1.04, $\varrho \approx 0.25$, obtained via: (a) JPEG; (b) F^1-transform; (c):YUV F-transform; (d) YUV F^1-transform.

Figure 4 shows the trend of the PSNR index obtained varying the compression rate. The trends obtained by executing JPEG and F^1-transform are similar. However, for strong compressions ($\varrho < 0.1$), the PSNR value calculated by executing JPEG decreases exponentially as the compression increases: this result shows that the quality of the decoded image obtained by JPEG drops quickly for very high compressions. The highest PSNR values are obtained by performing YUV F-transform and YUV F^1-transform. In particular, the PSNR values obtained with the two methods are similar for $\varrho < 0.2$, while, for lower compressions, YUV F^1-transform provides decompressed images of better quality than those obtained with YUV F-transform.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 September 2023 doi:10.20944/preprints202309.0964.v1

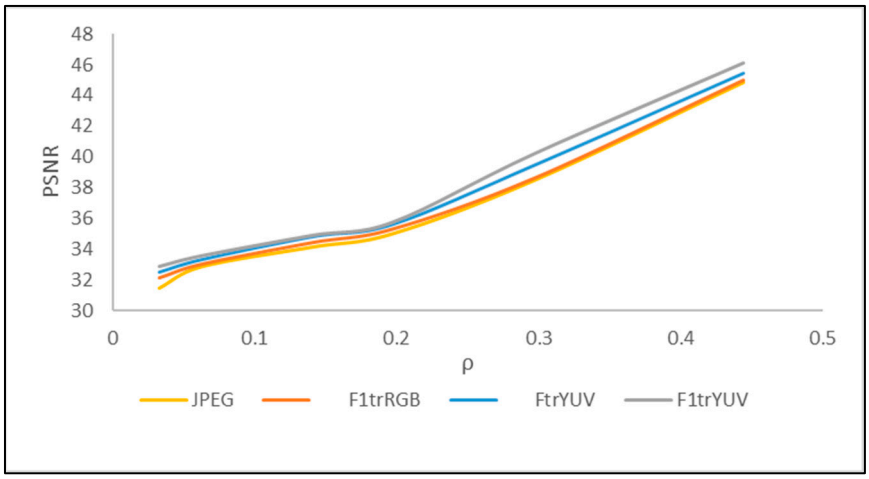


Figure 4. PSNR trend for the color image 4.1.04 obtained by executing the four color image compressions algorithms.

Now we show the results obtained for the color image 4.2.07. In Figure 5 we give the decoded images obtained by executing the four algorithms with compression rate $\varrho \approx 0.10$.

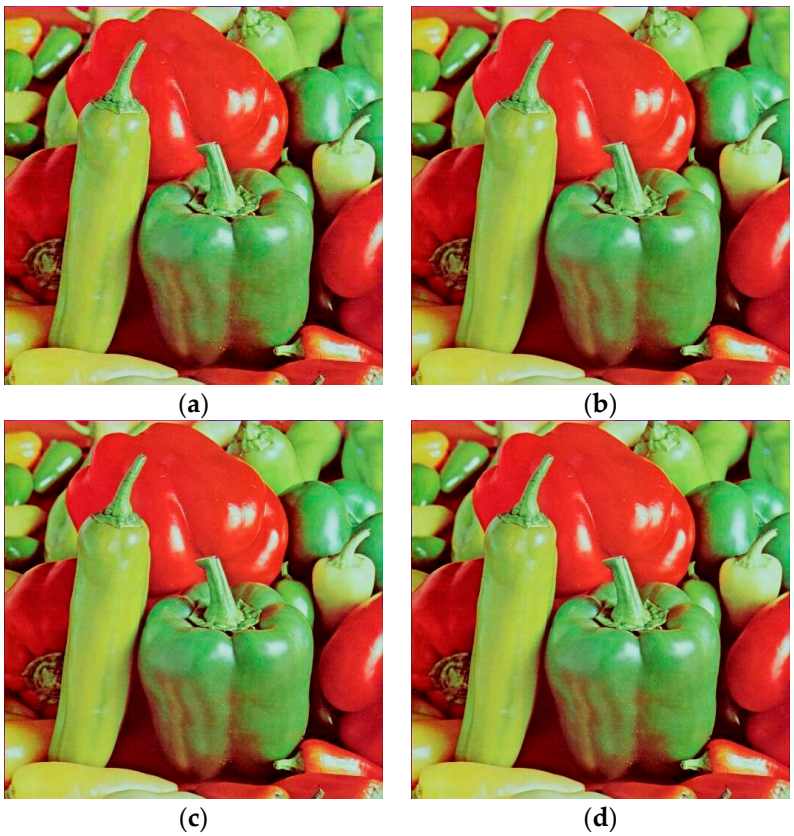


Figure 5. Decoded image 4.2.07, $\varrho \approx 0.10$, obtained via: (a) JPEG; (b) F^1-transform; (c):YUV F-transform; (d) YUV F^1-transform.

Figure 6 shows the decoded images of 4.2.07 obtained by executing the four algorithms with a compression rate $\varrho \approx 0.25$.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 September 2023 doi:10.20944/preprints202309.0964.v1

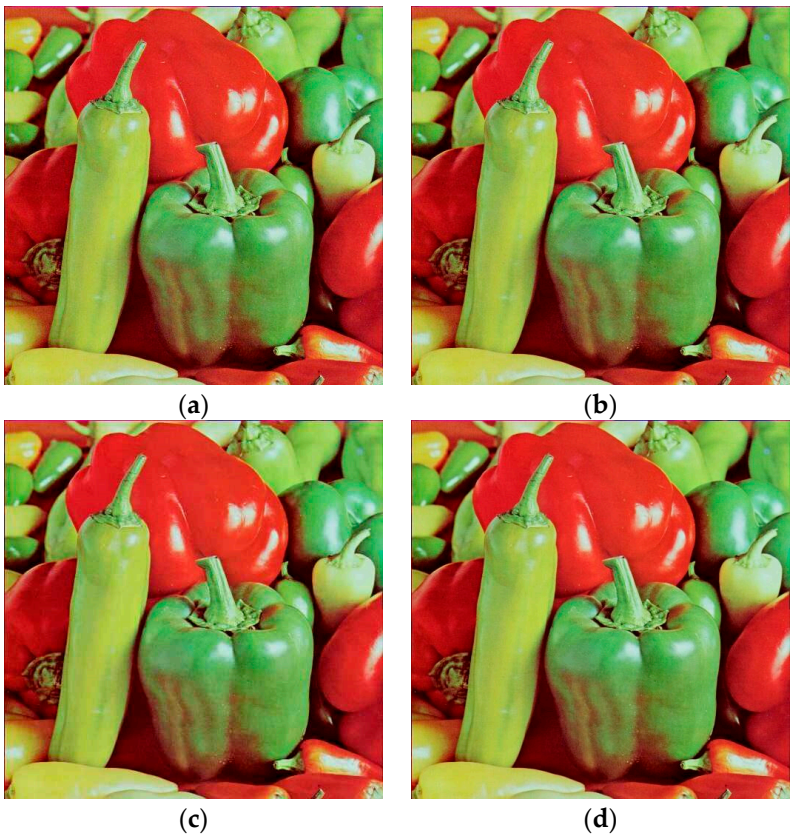


Figure 6. Decoded image 4.2.07, $\varrho \approx 0.25$, obtained via: (a) JPEG; (b) F^1-transform; (c):YUV F-transform; (d) YUV F^1-transform.

Figure 7 shows the trend of the PSNR index obtained by varying the compression rate. The best values of PSNR are obtained by executing YUV F^1-transform. The trend of the PSNR obtained by executing YUV F-transform is better than the one obtained by executing F-transform and JPEG. As the results obtained for the color image 4.1.04 show, the trend of PSNR obtained by executing JPEG for the image 4.2.07 decays rapidly as compression increases ($\varrho < 0.1$).

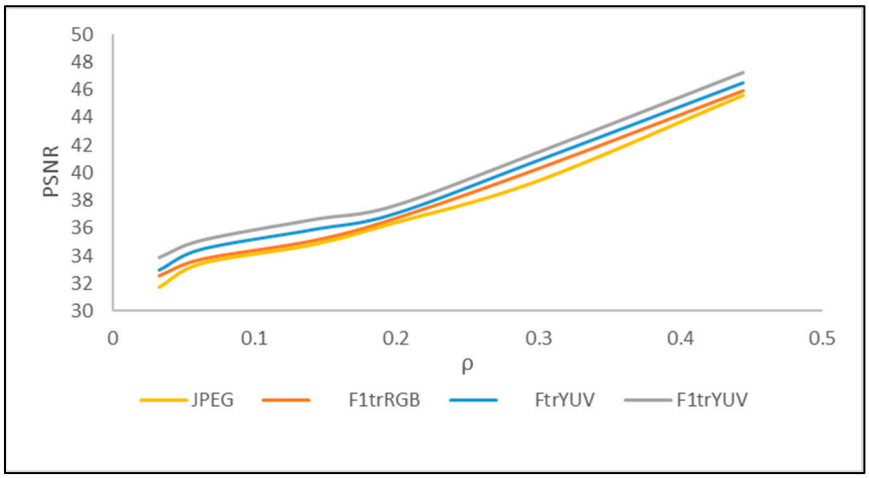


Figure 7. PSNR trend for the color image 4.2.07 obtained by executing the four color image compressions algorithms.

In Figure 8 we show the trends of the gain of the YUV F^1-transform algorithm with respect to the other three color image compression algorithms, where the gain index is calculated by (30) and is averaged for all the images of the dataset used in the comparative tests. The gain of the proposed method with respect to YUV F-transform is approximately equal to 2% regardless of the compression rate. The gain of YUV F^1-transform with respect to F^1-transform varies from 3%, for small

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 September 2023 doi:10.20944/preprints202309.0964.v1

compressions, to 4% for high compressions ($\varrho < 0.2$). The gain of YUV F¹-transform with respect to JPEG varies from 3%, for small compressions, to 5% for medium-high compressions ($0.1 < \varrho < 0.2$); for compression rates lower than 0.1 the gain index increases quickly as the compression increases until it reaches 7%.

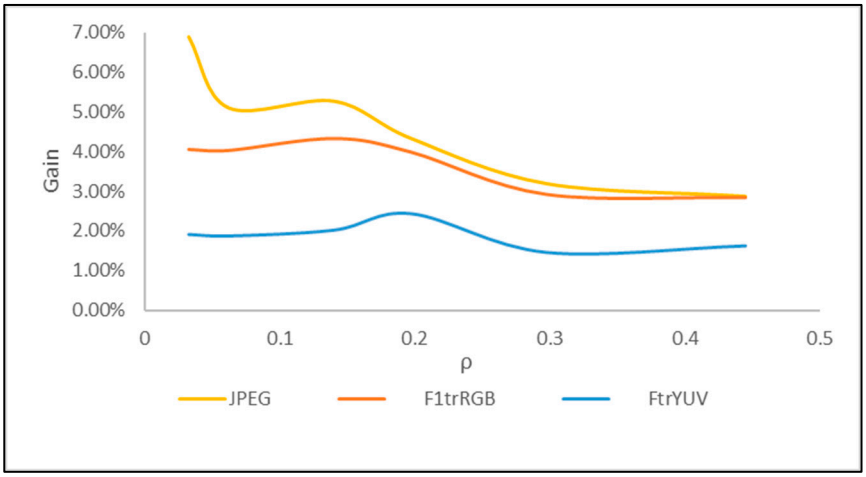


Figure 8. Trend of the Gain of YUV-F¹transform with respect to the other three color image compression methods.

These results shown that the quality of the images coded/decoded by YUV-F¹transform is higher than that obtained using YUV-Ftransform, F¹transform and JPEG, regardless of the compression level.

Finally, in Table 1 we show the mean coding and decoding CPU time obtained by executing the four color image compression algorithms: the average values refer to the CPU times measured for all images of the same size and for all compression rates: we see that both the coding and decoding CPU times measured by executing YUV F1-transform are comparable with those obtainedwith the other four image compression methods. Therefore, YUV-F¹-transform improves the quality of the reconstructed images obtained by executing JPEG, F¹-transform and YUV F-transform and provides CPU times similar to those obtained by executing the other image compression three methods at same time.

Table 1. Mean coding and decoding CPU time obtained for the 256x256 and 512x512 images by executing the four image compression algorithms.

CPU time		JPEG	F¹trRGB	FtrYUV	F¹trYUV
Coding	256x256	2.76	2.78	2.41	3.09
	512x512	5.75	5.88	5.66	6.01
Decoding	256x256	5.82	5.86	5.04	5.73
	512x512	9.52	9.85	9.12	9.56

5. Conclusions

A lossy color image compression process employing the bi-dimensional F¹-transform in YUV space is proposed. The benefit of this approach is to improve the quality of the reconstructed image, with acceptable CPU coding/decoding times. In fact, the F1-transform method manages to retain more information of the original image than other image compression methods, but with memory to be allocated and execution times. The proposed method on the YUV space allows to obtain a high quality of the decompressed image, without increasing the allocated memory and the CPU times. The results show that this method improves the quality of the decompressed image compared to that obtained with the use of JPEG, F-transform applied in YUV space and F¹-transform applied in RGB space. Moreover the execution times are compatible with those obtained by executing the other three color image compression methods.

In the future we intend to adapt the YUV F^1-transform algorithm to the compression of large color images

Author Contributions: Conceptualization, B. C.. and S.S.; methodology, B. C., F.D.M. and S.S.; software, B. C., F.D.M. and S.S.; validation, B. C., F.D.M. and S.S.; formal analysis, B. C., F.D.M. and S.S.; investigation, B. C., F.D.M. and S.S.; resources, B. C., F.D.M. and S.S.; data curation, B. C., F.D.M. and S.S.; writing—original draft preparation, B. C., F.D.M. and S.S.; writing—review and editing, B. C., F.D.M. and S.S.; visualization, B. C., F.D.M. and S.S.; supervision, B. C., F.D.M. and S.S.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wallace, G. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics* **1992**, *38*(1), xviii–xxxiv, doi: 10.1109/30.125072..
2. Raid, A. M.; Khedr, W. M.; El-Dosuky, M. A.: Ahmed, W. Jpeg image compression using discrete cosine transform-A survey. *International Journal of Computer Science & Engineering Survey (IJCSES)* **2014,** *5*(2), 39-47, doi 10.5121/ijcses.2014.5204.
3. Mostafa, A.; Wahid, K.; Ko, S.B. An Efficient YUV-based Image Compression Algorithm for Wireless Capsule Endoscopy, IEEE CCECE 2011 2011 24th Canadian Conference on Electrical and Computer Engineering (CCECE) 8-11 May 2011, Niagara Falls, Ontario, Canada., pp 943-946, doi: 10.1109/ICCITechn.2011.6164787.
4. Nobuhara, H.; Pedrycz, W.; Hirota, K. Relational image compression: optimizations through the design of fuzzy coders and YUV color space, *Soft Computing* **2005**, *9*(6), 471–479, doi:10.1007/s00500-004-0366-7.
5. Nobuhara, H.; Hirota, K.; Di Martino, F.; Pedrycz, W.; Sessa S. Fuzzy relation equations for compression/decompression processes of colour images in the RGB and YUV colour spaces, *Fuzzy Optimization and Decision Making* **2005**, *4*(3), 235–246, doi: 10.1007/s10700-005-1892-1.
6. Di Martino, F.; Loia, V.; Sessa, S. Direct and Inverse Fuzzy Transforms for Coding/Decoding Color Images in YUV Space, *Journal of Uncertain Systems* **2009**, 3(1), 11-30.
7. Perfilieva, I. Fuzzy Transform: Theory and Application. *Fuzzy Sets and Systems* **2006**, *157*, 993-1023, doi: 10.1016/j.fss.2005.11.012.
8. Di Martino, F.; Loia, V.; Perfilieva, I.; Sessa, S. An Image Coding/Decoding Method Based on Direct and Inverse Fuzzy Transforms. *International Journal of Approximate Reasoning* **2008**, *48*, 110-131, doi:10.1016/j.ijar.2007.06.008.
9. Son, T. N.; Hoang, T. M.; Dzung, N. T.; Giang, N. H. Fast FPGA implementation of YUV-based fractal image compression, 2014 IEEE Fifth International Conference on Communications and Electronics (ICCE), Danang, Vietnam, 2014, pp. 440-445, doi: 10.1109/CCE.2014.6916745.
10. Podpora, M.; Korbas, G. P.; Kawala-Janik, A. YUV vs RGB – Choosing a Color Space for Human-Machine Interaction, 2014 Federated Conference on Computer Science and Information Systems, Warsaw, Poland, 7 - 10 September 2014, vol. 3 pp. 29–34, doi: 10.15439/2014F206.
11. Ernawan, F.; Kabir, N.; Zamli, K.Z., An efficient image compression technique using Tchebichef bit allocation. *Optik*, **2017**, *148*, 106-119.
12. Zhu, S.; Cui, C.; Xiong, R.; Guo, U.; Zeng, B. Efficient Chroma Sub-Sampling and Luma Modification for Color Image Compression, in IEEE Transactions on Circuits and Systems for Video Technology, vol. 29, no. 5, pp. 1559-1563, May 2019, doi: 10.1109/TCSVT.2019.2895840.
13. Sun, H.; Liu, C.; Katto, J.; Fan, Y. An Image Compression Framework with Learning-Based Filter, Proceedings of 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 14-19 June 2020, Seattle, WA, USA, pp. 152-153.
14. Malathkar, N.V.; Soni, S.K. High compression efficiency image compression algorithm based on subsampling for capsule endoscopy. *Multimed Tools Appl* **2021**, 80, 22163–22175, doi:10.1007/s11042-021-10808-0.
15. Ma, H.; Liu, D.; Yan, N.; Li, H.; Wu, F. End-to-End Optimized Versatile Image Compression With Wavelet-Like Transform *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2022**, *44*, (3),. 1247-1263, doi: 10.1109/TPAMI.2020.3026003.
16. Yin, Z.; Chen, L.;Lyu, W.;Luo, B. Reversible attack based on adversarial perturbation and reversible data hiding in YUV colorspace. *Pattern Recognition Letters* **2023**, *166*, 1-7, doi:10.1016/j.patrec.2022.12.018.
17. Di Martino, F.; Sessa, S.; Perfilieva, I. First Order Fuzzy Transform for Images Compression. *Journal of Signal Information Processing* **2017**, *8*, 178-94, doi:10.4236/jsip.2017.83012.

18. Di Martino, F.; Sessa, S. Fuzzy Transforms for Image Processing and Data Analysis - Core Concepts, Processes and Applications; Springer Nature: Cham, Switzerland, 2020, pp. 217, doi:10.1007/978-3-030-44613-0.
19. Perfilieva, I.; Dankova, M.; Bede, B. Towards a higher degree F-transform. *Fuzzy Sets and Systems* **2011**, *180*, 3–19, doi:10.1016/j.fss.2010.11.002.
20. Technical Committee ISO/IEC JTC 1/SC 29 Coding of audio, picture, multimedia and hypermedia information, ISO/IEC 10918-1:1994 - Information technology — Digital compression and coding of continuous-tone still images: Requirements and guidelines, 1994, 182 pp.
21. Wang, Y.; Tohidypour, H.R.; Pourazad, M.T.; Nasiopoulo, P.; Leung, V.C.M. Comparison of Modern Compression Standards on Medical Images for Telehealth Applications. In 2023 IEEE International Conference on Consumer Electronics (ICCE), 2023, pp. 1-6.
22. Prativadibhayankaram, S.; Richter, T.; Sparenberg, H.; Fößel, S. Color Learning for Image Compression. *arXiv preprint arXiv*, **2023**, 2306.17460.
23. Yin, Z.; Chen, L.;Lyu, W.;Luo, B. Reversible attack based on adversarial perturbation and reversible data hiding in YUV colorspace. *Pattern Recognition Letters* **2023**, *166*, 1-7, doi:10.1016/j.patrec.2022.12.018.