axioms

MDPI

Article

A Multilevel Fuzzy Transform Method for High Resolution Image Compression

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

[2] Centro Interdipartimentale Alberto CalzaBini, Università degli Studi di Napoli Federico II, Via Toledo 402, 80134 Napoli, Italy

* Correspondence: fdimarti@unina.it; Tel.: +39-0812538908; Fax: +39-081238905; Mobile: +39-3334529362

Abstract: The Multilevel Fuzzy Transform technique (MF-tr) is a hierarchical image compression method based on Fuzzy Transform, which is successfully used to compress images and manage the information loss of the reconstructed image. Unlike other lossy image compression methods, it ensures that the quality of the reconstructed image is not lower than a prefixed threshold. However, this method is not suitable for compressing massive images due to the high processing times and memory usage. In this paper, we propose a variation of MF-tr for the compression of massive images. The image is divided into tiles, each of which is individually compressed using MF-tr; thereafter, the image is reconstructed by merging the decompressed tiles. Comparative tests performed on remote sensing images show that the proposed method provides better performance than MF-tr in terms of compression rate and CPU time. Moreover, comparison tests show that our method reconstructs the image with CPU times that are at least two times less than those obtained using the MF-tr algorithm.

Keywords: F-transform; MF-tr; MIMMF-tr; compression rate; PSNR; PSNR threshold

Citation: Di Martino, F.; Sessa, S. A Multilevel Fuzzy Transform Method for High Resolution Image Compression. *Axioms* **2022**, *11*, 551. https://doi.org/10.3390/axioms11100551

Academic Editor: Hari Mohan Srivastava

Received: 22 September 2022
Accepted: 11 October 2022
Published: 13 October 2022

1. Introduction

The bi-dimensional Fuzzy Transform (F-transform) has been initially proposed in [1] as a lossy image compression technique. The direct F-transform is applied to construct the compressed image and the image is reconstructed by computing the inverse F-transform. In [2,3], the image is partitioned in blocks and the F-transform technique is applied to compress each block. In [2], the authors show that the F-transform image compression technique applied to blocks of the image achieves the best performance with respect to the images in which fuzzy relation equations are applied. In [3,4], the authors compare the F-transform and JPEG image compression technique, showing that the two methods provide similar results in terms of decompressed image quality, but F-transform has shorter execution times than JPEG.

In the last decade, the bi-dimensional F-transform has been applied in various image processing problems, such as image fusion [5–7], image autofocus [8,9], edge detection [10,11], image segmentation [12,13], image watermarking [14], and noise reduction [15]. An overview of the main image processing techniques that use the two-dimensional F-transform is provided in [16].

To control the loss of information due to the compression, the authors of [17] proposed a variation of the F-transform method called Multilevel F-transform (MF-tr), in which the image is compressed hierarchically in a multilevel structure similar to the ones used in Laplace Pyramid [18] and Wavelet-based [19,20] image compression. Initially, the image is compressed using the two-dimensional block-based F-transform method. It is subsequently decompressed, and the quality of the decompressed image compared to the original image is measured, computing the Peak Signal to Noise Ratio index (PSNR). If the quality of the decompressed image is lower than a predetermined threshold, the "Error" image is constructed, which is provided by the difference between the original image and the decompressed one, and the two-dimensional F-Transform is applied to compress the Error image. The reconstruction image will consist of the previously decompressed image in addition to the decompressed image of the Error. The PSNR of the reconstructed image is computed with respect to the original image; if the PSNR is below the threshold, the new Error image is built and the process is iterated into the next level. The process ends when the PSNR of the reconstructed image with respect to the original image is greater than or equal to the threshold.

The final reconstructed image is given by the sum of all reconstructed images computed in each level. This method represents a trade-off between the reconstructed image quality and the compression rate. If it is necessary to obtain a high quality of the reconstructed image, with a minimum loss of information compared to the original image, then a high value of the PSNR threshold is set and the execution times will be longer.

In [21], a variation of MF-tr, called Fast Multilevel Fuzzy Transform (Fast M-ftr), is proposed to accelerate execution times. In Fast M-ftr, a preprocessing phase is performed to find the optimal value of the compression rate, which guarantees to minimize the number of iterations.

The bi-dimensional F-transform can be applied to compress high resolution images. The authors of [22] compared four satellite lossy image compression methods using the wavelet domain, which are the image compression algorithm proposed by the Consultative Committee for Space Data Systems (CCSDS) [23], Wavelet lossy image compression [24], Bandelet [25], and JPEG 2000 [26]. The authors show that the CCSDS has the highest resolution image compression.

In [27], the authors use pseudo-exponential functions as basic functions. They compare the results with the ones obtained by applying the F-transform with cosinusoidal basic functions and wavelet image compression methods, showing that the decompressed images obtained with their method have high quality, but also high time consumption.

A hybrid massive image compression method is proposed in [28]. The bi-dimensional F-transform is used to compress the original image. Then, adjoint pixels in the compressed image with the same gray level are binned by merging them in a bin.

This method is very effective in the lossy compression of massive images, where it is not necessary to limit or reduce the information loss. In addition, the binning of images compressed via the bi-dimensional F-transform is a good trade-off between the level of compression and the quality of the reconstructed image.

In some cases, as in high level medical and remote sensing images, it is necessary to check that the quality of the decompressed image remains above a prefixed threshold. The F-transform massive image compression [28] is unsuitable for checking that the quality of the decompressed image is always greater than the threshold, unlike the algorithms based on the bi-dimensional MF-tr [17] and [21].

In this paper, we present a new approach based on the MF-tr to massive images. We call this method Massive Image Multilevel F-transform compression method (MIMF-tr). Our idea is to split the image into tiles of equal size. Splitting the image in tiles is necessary to compress massive images; each tile is compressed separately by executing the MF-tr compression algorithm. Then, all of the compressed tiles are merged to compose the compressed image. The decompressed image is obtained by executing the MF-tr image reconstruction method to each compressed tile and, finally, merging the reconstructed tiles.

MIMF-tr, similar to MF-tr, allows for controlling the quality of the compressed image. Furthermore, unlike MF-tr, MIMF-tr is suitable for compressing massive images, adopting a technique of partitioning the image in tiles and compressing the tiles separately.

In Figure 1, the MIMF-tr image compression and reconstruction methods are schematized.

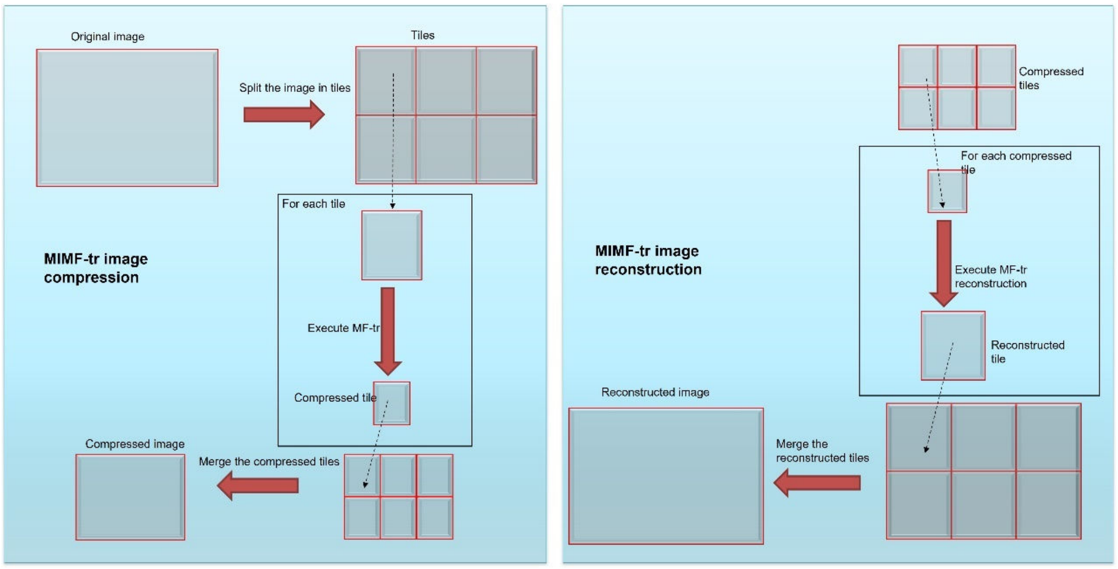


Figure 1. Schema of the MIMF-tr compression and reconstruction algorithms.

The Peak Signal to Noise Ratio (PSNR) measure is used to measure the quality of the compression. Each tile is partitioned in blocks and each block is compressed using the MF-tr compression method, in which the iteration process with the execution of the subsequent level continues until the PSNR of the decompressed block is greater than or equal to a fixed threshold.

In Section 2, we briefly describe the base concepts of direct and inverse F-transform and discuss the MF-tr image compression and reconstruction algorithms. In Section 3, we present the MIMF-tr image compression and reconstruction algorithms. The results of comparison tests executed on massive images are shown in Section 4. Conclusions are included in Section 5.

2. Preliminaries

2.1. Discrete Direct and Inverse F-Transform for Coding/Decoding Images

The bi-dimensional discrete F-transform was applied in [1–3] for coding/decoding a gray level image by approximating a continuous function f: $X \rightarrow Y$ in a closed interval [a,b] based on the knowledge of the value of f in a discrete set of points.

In [1], the concept of basic functions is introduced, which is given by the fuzzy set of a fuzzy partition of a closed interval real domain [a,b], as described below.

Let $\{x_1, x_2, \ldots, x_n\}$ be a set of $n \geq 2$ points of [a,b], called nodes, in order that $x_1 = a < x_2 < \ldots < x_n = b$. A fuzzy partition of [a,b] is given by a family of fuzzy sets $A_1, \ldots, A_n$: $[a,b] \rightarrow [0,1]$, where $A_i(x)$, $i = 1,2,\ldots,n$, are continuous functions on [a,b] if the following conditions hold:

- $A_i(x_i) = 1$, $\forall\, i \in \{1,2,\ldots,n\}$;
- $A_i(x) = 0$ $\forall\, x \notin (x_{i-1}, x_{i+1})$, where we assume $x_0 = x_1 = a$ and $x_{n+1} = x_n = b$;
- $A_i(x)$ strictly increases on $[x_{i-1}, x_i]$, $\forall\, x \in [a,b]$, and $\forall\, i \in \{1,2,\ldots,n\}$;
- $A_i(x)$ strictly decreases on $[x_i, x_{i+1}]$, $\forall\, x \in [a,b]$, and $\forall\, i \in \{1,2,\ldots,n-1\}$;
- $\sum_{i=1}^{n} A_i(x) = 1,\ x \in [a,b]$.

Let f be a continuous function defined in [a,b]. Furthermore, let $P = \{ p_1,\ldots,p_N \}$ be a set of N points in [a,b], where the values assumed by the function f are known. The set P is called a set of points, which is sufficiently dense with respect to the fuzzy partition $\{A_1, A_2,\ldots, A_n\}$ if for each basic function A_i, $i = 1,\ldots,n$, exists in at least a point p_j, $j = 1,\ldots,m$, in order that $A_i(p_j) > 0$.

If P is sufficiently dense with respect to the fuzzy partition $\{A_1, A_2,\ldots, A_n\}$, we can define the n-dimensional vector $\mathbf{F} = [F_1, \ldots,F_n]$ with components:

$$F_i = \frac{\sum_{j=1}^{N} f(p_j)A_i(p_j)}{\sum_{j=1}^{N} A_i(p_j)} \quad i = ,2,\dots,N \tag{1}$$

called discrete direct F-transform of *f* with respect to the fuzzy partition {A$_1$, A$_2$,…, A$_n$}.

The value of *f* in a point x ∈ [a,b] can be approximated using the discrete inverse F-transform of *f* with respect to {A$_1$, A$_2$, …, A$_n$}, given by the formula:

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

Now, let I be an N×M image. The previous concept of direct and inverse F-transform of a function defined on a real closed interval [a,b], can be extended to two-variable functions defined in the closed interval [a,b] × [c,d]. In this case, we can consider the image I as a bi-dimensional function defined on the closed interval [1,N] × [1,M], known in all the points with coordinates (i,j) of the pixels, where i ∈ [1,N] and j ∈ [1,M].

Furthermore, let A$_1$, …, A$_n$: [1,N] → [0,1] be a fuzzy partition of [1,N] and B$_1$, …, B$_m$: [1,M] → [0,1] be a fuzzy partition of [1,M]. Clearly, the sets of points P={1,2, …, N} and Q={1,2, …, M} are sufficiently dense with respect to the partitions {A$_1$, A$_2$, …, A$_n$} and {B$_1$, …, B$_m$}, respectively. Therefore, we can define the bi-dimensional discrete direct F-transform of *f* given by the matrix **F** with components:

$$F_{kl} = \frac{\sum_{j=1}^{M}\sum_{i=1}^{N} f(p_i,q_j)A_k(p_i)B_l(q_j)}{\sum_{j=1}^{M}\sum_{i=1}^{N} A_k(p_i)B_l(q_j)},\ i = 1,2,\dots,N,\ j = 1,2,\dots,M \tag{3}$$

The compressed n × m image is given by the direct F-transform **F**. The compression rate (the inverse of the compression ratio parameter) is $\rho = \frac{n\times m}{N\times M}$.

The original image is reconstructed by computing the bi-dimensional discrete inverse F-transform with respect to {A$_1$, A$_2$, …, A$_n$} and {B$_1$, B$_2$,…,B$_m$} given by:

$$I_{nm}^{F}(i,j) = \sum_{k=1}^{n}\sum_{l=1}^{m} F_{kl}\, A_k(i)\, B_l(j) \tag{4}$$

In [2–4], the bi-dimensional discrete direct F-transform is applied to the compressed images.

2.2. Multilevel F-Transform for Massive Image Compression

The MF-transform image compression method is an F-transform based method that allows you to control the loss of information in compression phase.

Initially (Level 1), the source image is compressed using the bi-dimensional F-transform. Then, it is decompressed and the PSNR is measured and compared with the original image.

Formally, let I$_0$ be the N × M source image. It is compressed using the direct bi-dimensional F-transform in an n × m image I_1^C and it is decompressed using the inverse bi-dimensional F-transform in an N × M image I_1^F. Call I_1^R the image reconstructed at Level 1. At this level, we have $I_1^R = I_1^F$, namely, the reconstructed image at the first level corresponds to the decompressed image.

The PSNR index is computed to measure the quality of the reconstructed image at each level. If the PSNR is greater than or equal to a prefixed threshold PSNR*th* the algorithm ends. Otherwise, a further image is calculated, called Error, whose pixels are given by the difference in absolute value between the corresponding pixels of the original image and those of the decompressed image.

The error image constitutes the input image to the next level (Level 2) I_2. The previous process is repeated, obtaining the compressed image I_2^C and decompressed image I_2^F. The reconstructed image at this level is given by $I_2^R = I_2^F + I_1^R$. If the PSNR of the reconstructed image I_2^R measured with respect to the source image I$_0$ is greater than or equal to PSNR*th*, then the process ends. Otherwise, the Level 2 error is calculated, which is given by the formula $I_2 = \left|I_2^F - I_1\right|$, which represents the absolute difference between the source image at Level 2 and its decompressed image. The image I_2 is the source image in the input to the next level, Level 3.

This process continues iteratively until the PSNR of the reconstructed image at the level L, such as I_L^R, measured with respect to the source image I_0 is greater than or equal to PSNR*th*.

The PSNR of the image reconstructed at Level k is obtained by calculating the Mean Square Error (MSE) with respect to the source image I_0, given by:

$$MSE = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}(I_k^R(i,j) - I_0(i,j))^2}{N \times M} \tag{5}$$

The PSNR index is given by the formula:

$$PSNR = 20 \log_{10} \frac{GrLev - 1}{MSE} \tag{6}$$

where the parameter *GrLev* is the number of gray levels of the source image.

The final reconstructed image is given by:

$$I_L^R = I_{L-1}^R + I_L^F = \sum_{k=1}^{L} I_k^F \tag{7}$$

The pseudocode of the MF-tr image compression algorithm is shown below (Algorithm 1). It is executed by assigning the N × M source image I_0 as an input parameter, which is the compression rate ϱ and the PSNR threshold PSNR*th*.

Algorithm 1 MF-tr image compression.

```
Input:     N × M source image I0
           Compression rate ϱ
           Threshold similarity PSNRth
Output:    Compressed n × m images obtained at each level
1.   I:=$I_0$,
2.   k:=1
3.   $I^R$:= 0        //  $I^R$ is initialized to the Null N × M image
4.   stopIter:= FALSE
5.   PSNRold: =PSNRth
6.   While (stopIter=FALSE)
7.       $I^C$[k]: =DirectFTR(I, ρ)    // Compression via direct F-transform
8.       $I^F$: =InversetFTR($I^C$[k], ρ) //Decompression via inverse F-transform
9.       $I^R$:=$I^R$ + $I^F$//Decompression via inverse F-transform
10.      Calculate the PSNR index (6)
11.      If (PSNR ≥ PSNRth) Then
12.          stopIter:= TRUE
13.      Else
14.          k: =k+1
15.          I := $|I^F - I|$
16.    End If
17.  End While
18.  Return $I^C[1], I^C[2], \ldots, I^C[k]$
```

The reconstructed image is obtained by (7). In Algorithm 2, the image reconstruction algorithm is shown.

Algorithm 2 MF-tr image reconstruction.

```
Input:      n × m compressed images $\mathbf{I^C[1], I^C[2], \ldots, I^C[L]}$
            Compression rate ρ
Output:     Reconstructed N × M image $I^R$
1.    $I^R$: =0        // $I^R$ is initialized to the Null N x M image
2.    For k = 1 to L
3.        $I^F$: =InversetFTR($I^C$[k], ρ)      //Decompression via inverse F-transform
4.        $I^R$: =$I^R + I^F$
5.    Next k
6.    Return $I^R$
```

The compressed image will be constituted by L matrices with dimension n × m. The final compression rate will be given by:

$$\rho^R = \frac{L \times n \times m}{N \times M} = L\rho \tag{8}$$

3. The Massive Images Multilevel F-Transform Image Compression Method

The MF-tr technique is not suitable for handling massive images as it would require high CPU times and memory capacity. On the other hand, techniques for reducing the complexity of the image, such as the one proposed in [28], are inapplicable if it is necessary to guarantee a minimum loss of information in the reconstructed image.

We propose a variation of the MF-tr technique in which the image is split in T tiles with an identical size. Each tile is separately compressed using the MF-tr method, with the constraint that the PSNR index of the reconstructed tile with respect to the original tile is not less than a prefixed threshold PSNR*th*.

Let I_0 be a $N_0 \times M_0$ massive image (for example, a remote sensing high resolution image in a band). The image I_0 is split in T tiles with size N × M, where $N = N_0/T$ and $M = M_0/T$. Each tile is separately compressed by executing the MF-tr compression method.

If I_{0t} is the t*th* N × M tile, it will be compressed in a set of L_t n × m matrices where L_t is the number of levels needed to compress the tile I_{0t}.

If $\rho = \frac{n \times m}{N \times M}$ is the compression rate used to compress the N × M matrix at each level, the final compression rate is given by:

$$\rho_t{}^R = \frac{L_t \times n \times m}{N \times M} = L_t\rho \tag{9}$$

The mean final compression rate is

$$\rho^R = \frac{1}{T}\sum_{t=1}^{T} \rho_t{}^R = \frac{\rho}{T}\sum_{t=1}^{T} L_t \tag{10}$$

The MF-tr image compression algorithm (Algorithm 1) is used to reconstruct the original image. The correspondent function MF-trCompression() is called the MIMF-tr image reconstruction algorithm (Algorithm 3).

Algorithm 3 MIMF-tr image compression.

```
Input:    N0 × M0 source image I0
          Compression rate ϱ
          Threshold similarity PSNRth
Output:   Compressed n × m images obtained at each level
1.  Split the image $I_0$ in T tiles $I_{01}, I_{02}, \dots, I_{0T}$
2.  For t = 1 to T
3.  // Execute MF-trCompression() to compress the $t^{th}$ tile
4.      $I_t^C[1], I_t^C[2], \dots, I_t^C[L_t]$: =MF-trCompression $(I_{0t}, \rho, PSNR_{th})$
5.  Next t
6.  Return $I_1^C[1], I_1^C[2], \dots, I_1^C[L_1],\ I_2^C[1], I_2^C[2], \dots, I_2^C[L_2], \dots,\ I_T^C[1], I_T^C[2], \dots, I_T^C[L_t]$
```

In Figure 2, the flow diagram of the MIMF-tr image compression algorithm is schematized.

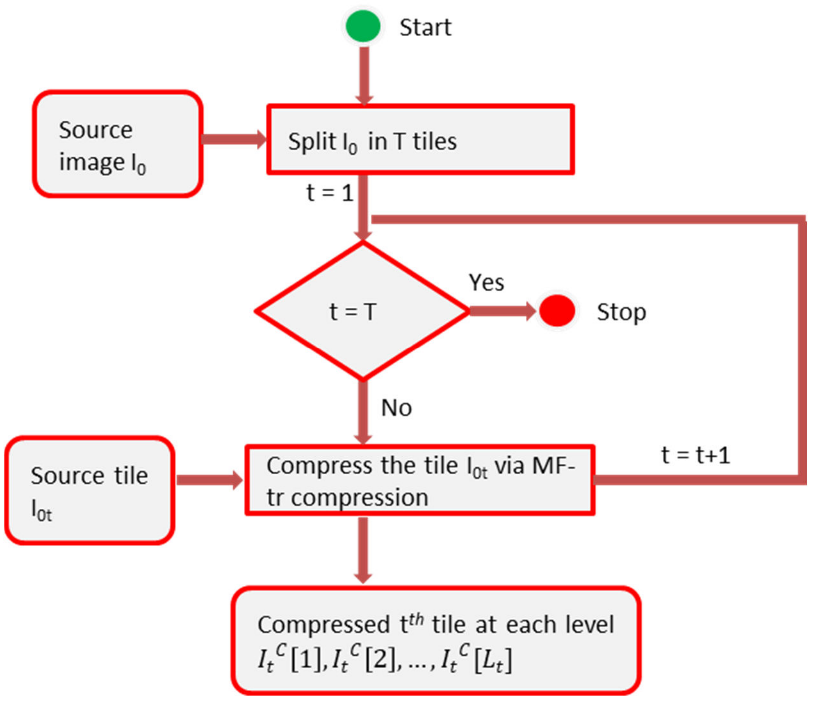

Start
Source image I_0
Split I_0 in T tiles
t = 1
t = T
Yes
Stop
No
Source tile I_{0t}
Compress the tile I_{0t} via MF-tr compression
t = t+1
Compressed t^{th} tile at each level $I_t^C[1], I_t^C[2], \dots, I_t^C[L_t]$

Figure 2. Flow diagram of the MIMF-tr image compression algorithm.

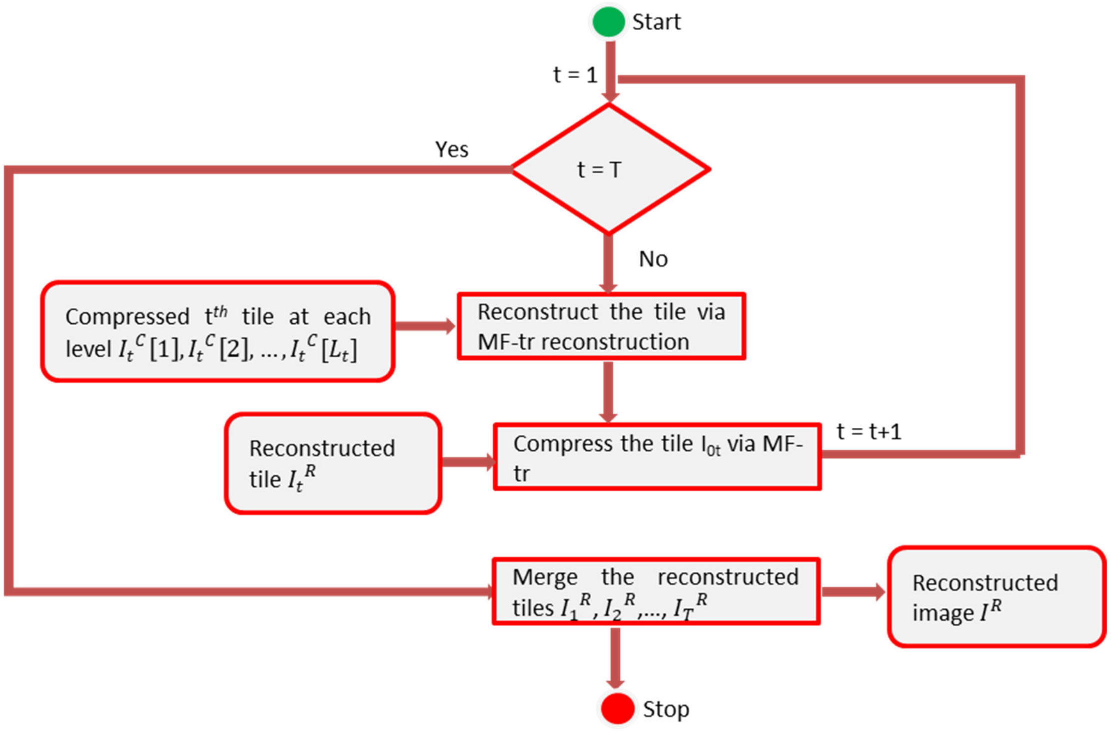
The MF-tr image reconstruction algorithm (Algorithm 2) is used to reconstruct the original image. The correspondent function MF-trReconstruction() is called in the MIMF-tr image reconstruction algorithm (Algorithm 4).

Algorithm 4 MIMF-tr image reconstruction.

```
Input:    n × m compressed images $\mathbf{I^C[1], I^C[2], \dots, I^C[L]}$
          Compression rate ϱ
Output:   Reconstructed N × M image $I^R$
1.  For t= 1 to T
2.      // Execute MF-trReconstruction() to reconstruct the tth tile
3.      $I^R_t$ =MF-trReconstruction$(I_t^C[1], I_t^C[2], \dots, I_t^C[L_t], \rho)$
4.  Next t
5.  $I^R$: =merge$(I^R_1, I^R_2, \dots, I^R_T)$
```

6. ***Return*** I^R

In Figure 3, the flow diagram of the MIMF-tr image reconstruction algorithm is schematized.



Start
t = 1
Yes
t = T
No
Compressed tth tile at each level $I_t^C[1], I_t^C[2], \ldots, I_t^C[L_t]$
Reconstruct the tile via MF-tr reconstruction
Reconstructed tile I_t^R
Compress the tile I_{0t} via MF-tr
t = t+1
Merge the reconstructed tiles $I_1^R, I_2^R, \ldots, I_T^R$
Reconstructed image I^R
Stop

Figure 3. Flow diagram of the MIMF-tr image reconstruction algorithm.

The final PSNR index of the reconstructed image with respect to the original one is given by formula (6). The final compression rate, calculated by formula (10), varies according to the PSNR threshold and the complexity of the image; the higher the quality of the reconstructed image (i.e., intended to be guaranteed), the higher the average number of levels necessary in the compression, and the higher the final compression rate ρ^R, the lower the compression of the image.

Since each tile is treated separately, the number of levels necessary in the compression can be different from tile to tile. If a tile contains more uniform or homogeneous image areas, the compression process will require the generation of fewer layers, compared to other tiles. A good choice of the number of tiles can be made in a way that some tiles can cover uniform image areas.

We test the MIMF-tr algorithm on a sample of remote sensing massive images by varying the number of tiles. To evaluate the best number of tiles in which we can split the image, we measure the mean number of levels necessary to compress the tiles given by:

$$\overline{L} = \frac{1}{\tau}\sum_{t=1}^{T} L_t \tag{91}$$

We perform comparisons of our method with the MF-tr method measuring the final PSNR, the final compression rate, and the number of levels obtained by executing the two methods. In next section, we show and discuss the results of our tests.

4. Test Results

We test our method on a set of 200 HLS Landsat surface remote sensing images extracted from the NASA Earth Data website 3660 × 3660 pixels [29]. The MIMF-tr algorithm was implemented in a Python platform; the tests were executed on an Intel Core i7-59360X processor with a clock frequency of 3 GHz. All the images are compressed with a compression rate ϱ = 0.067. The threshold PSNR*th* is set to 36.

We compare the performances of MIMF-tr with the ones of M-FTR in terms of number of levels necessary to compress the source images, final compression rate, and CPU time, using different values for the number of tiles in MIMF-tr. Briefly, we show in detail the results obtained for three images in the dataset.

In Figure 4a, the original image is shown, while in Figure 4b, the reconstructed image obtained by executing MF-tr is shown. Figure 4c–f shows the reconstructed images by splitting the original image in 6, 10, 20, and 30 tiles, respectively.

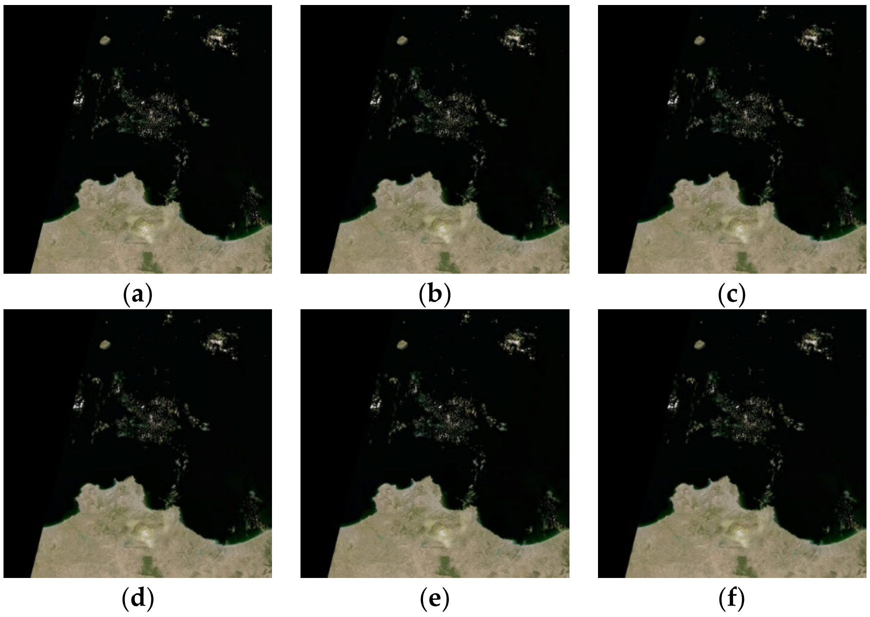


Figure 4. Source image and reconstructed images. (**a**) Source image; (**b**) reconstructed using MFTR; (**c**) reconstructed image (T = 6); (**d**) reconstructed image (T = 6); (**e**) reconstructed image (T = 20); (**f**) reconstructed image (T = 30).

In Table 1, the results obtained by compressing the satellite source image with MF-tr and MIMF-tr are shown in Figure 4a. The mean level, PSNR, final compression rate, and CPU time are averaged on the three bands: Red, Green, and Blue.

Table 1. Mean levels, PSNR, mean compression rate, and CPU times for the source image in Figure 2a.

Algorithm	Tile	Mean Level	PSNR	ρ^R	CPU Time (s)
MF-tr	-	5.00	36.44	0.33	94.65
MIMF-tr	6	4.00	36.40	0.27	78.28
MIMF-tr	10	3.40	36.38	0.23	77.49
MIMF-tr	20	3.35	36.22	0.22	77.28
MIMF-tr	30	3.40	36.18	0.23	77.54

The best results are obtained by MIMF-tr using 20 tiles. Furthermore, independently from the number of tiles set, the performances of MIMF-tr in terms of mean level, final compression rate, and CPU times obtained using MIMF-tr are better than the ones obtained using MF-tr. The final CPU time taken by MIMF-tr to compress the image is reduced by more than 23% compared to the CPU time taken by MF-tr. In addition, the mean number of levels achieved using MIMF-tr to obtain a reconstructed image with PSNR greater than 36 is always less than the one achieved using MF-tr. The final compression rate obtained executing MIMF-tr is always less than the final compression rate obtained by running MF-tr.

Figure 5a shows another HLS Landsat surface remote sensing image. Figure 5b shows the reconstructed image obtained by executing MF-tr. Figure 5c–f shows the reconstructed images by splitting the original image in 6, 10, 20, and 30 tiles, respectively.

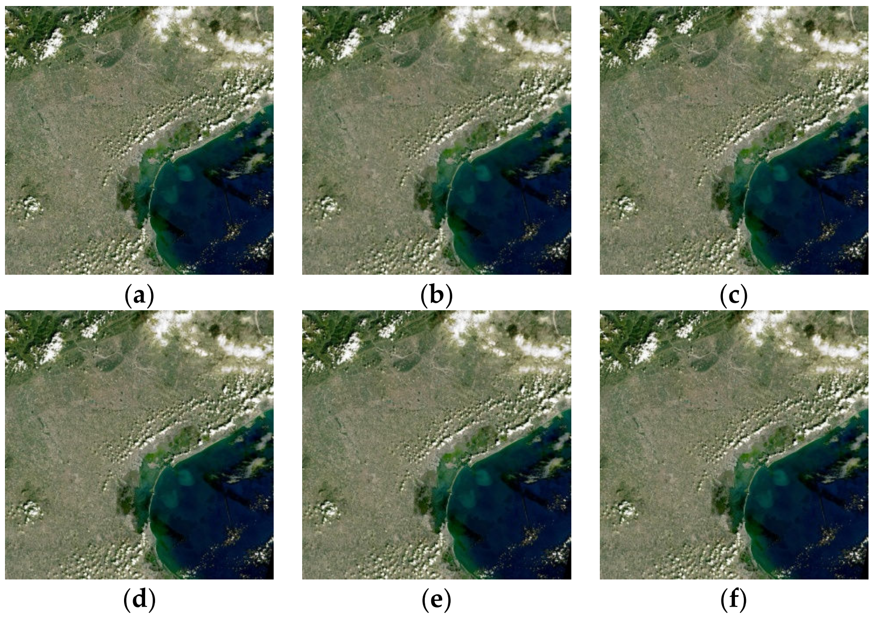


Figure 5. Source image and reconstructed images. (**a**) Source image; (**b**) reconstructed using MFTR; (**c**) reconstructed image (T = 6); (**d**) reconstructed image (T = 6); (**e**) reconstructed image (T = 20); (**f**) reconstructed image (T = 30).

In Table 2, the results obtained by compressing the satellite source image in Figure 5a with MF-tr and MIMF-tr are shown. The mean level, PSNR, and CPU time are averaged on the three bands: Red, Green, and Blue.

Table 2. Mean levels, PSNR, mean compression rate, and CPU times for the source image in Figure 5a.

Algorithm	Tiles	Mean Level	PSNR	ρ^R	CPU Time (s)
MF-tr	-	5.67	36.44	0.38	98.12
MIMF-tr	6	4.28	36.40	0.27	78.23
MIMF-tr	10	4.20	36.38	0.25	77.78
MIMF-tr	20	4.23	36.22	0.26	77.91
MIMF-tr	30	4.26	36.18	0.27	78.05

The best results are obtained by MIMF-tr using 10 tiles. As in the case of the previous image, the performances of MIMF-tr are better than the ones of MF-tr, independently from the number of tiles set. The CPU time taken by MIMF-tr to compress the image is reduced by more than 22% compared to the CPU time taken by MF-tr and the final compression rate and the mean number of levels are always less than the one measured using MF-tr.

Figure 6 shows another HLS Landsat surface remote sensing image. Figure 5b shows the reconstructed image obtained by executing MF-tr. Figure 5c–f shows the reconstructed images by splitting the original image in 6, 10, 20, and 30 tiles, respectively.

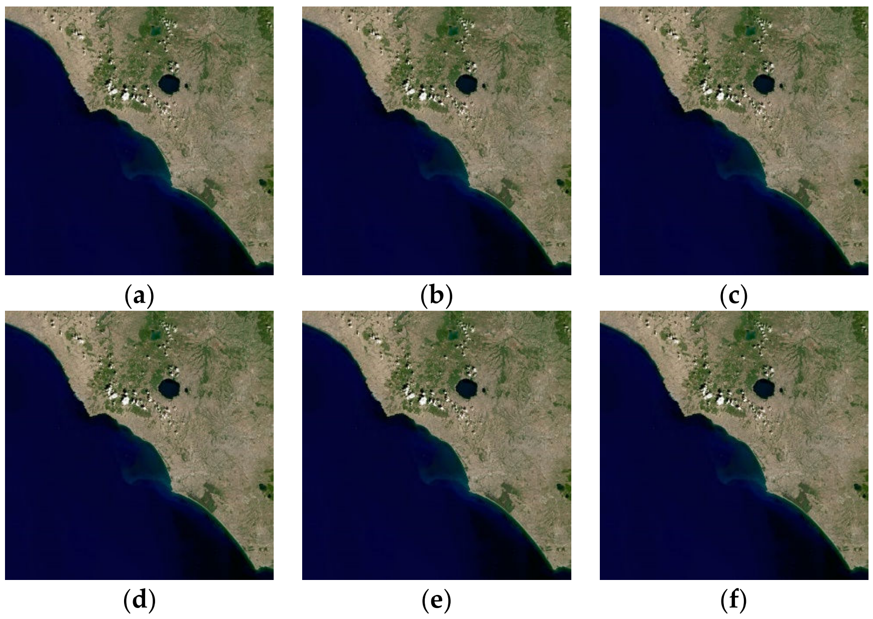


Figure 6. Source image and reconstructed images. (**a**) Source image; (**b**) reconstructed using MFTR; (**c**) reconstructed image (T = 6); (**d**) reconstructed image (T = 6); (**e**) reconstructed image (T = 20); (**f**) reconstructed image (T = 30).

In Table 3, the results obtained compressing the satellite source image in Figure 4a with MF-tr and MIMF-tr are shown. The mean levels, PSNR, and CPU time are averaged on the three bands: Red, Green, and Blue.

Table 3. Mean levels, PSNR, mean compression rate, and CPU times for the source image in Figure 4a.

Algorithm	Tiles	Mean Level	PSNR	ρ^R	CPU Time (s)
MF-tr	-	4.67	36.44	0.36	94.21
MIMF-tr	6	3.23	36.40	0.22	76.34
MIMF-tr	10	3.17	36.38	0.21	76.09
MIMF-tr	20	3.09	36.22	0.20	75.94
MIMF-tr	30	3.16	36.18	0.21	76.02

The best results are obtained by MIMF-tr using 20 tiles. As in the previous examples, the performances of MIMF-tr are better than the ones of MF-tr, independently from the number of tiles set. The CPU time taken by MIMF-tr to compress the image is reduced by more than 25% compared to the CPU time taken by MF-tr. Moreover, in this case, the mean level and final compression rate measured executing MIMF-tr are less than the one obtained executing MF-tr.

The graphs in Figure 7a–c show the trend of the mean level, final compression rate, and CPU time obtained, respectively, by executing MIMF-tr with respect to MF-tr for all the satellite images in the dataset. The measurements of the three parameters obtained by executing MIMF-tr are averages obtained by executing MIMF-tr and decomposing the source image into 6, 10, 20, and 30 tiles, respectively.

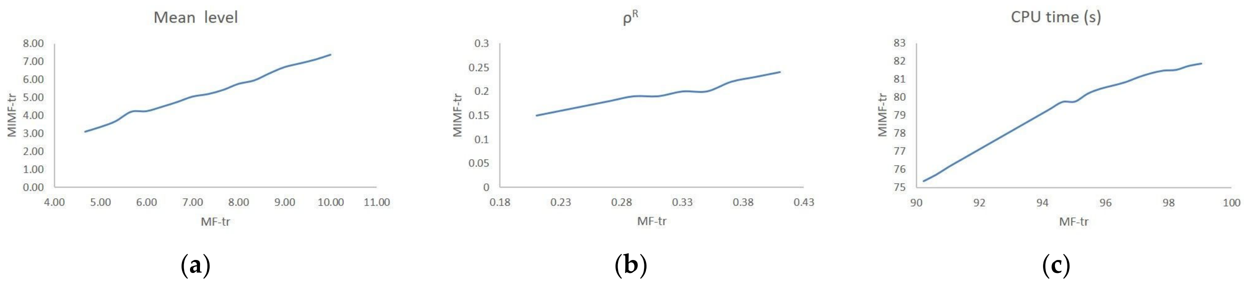


Figure 7. Trend of: (**a**) Mean level; (**b**) final compressed rate; (**c**) CPU time.

These results show that for all images, MIMF-tr has better performance than MF-tr, both in terms of the number of levels required for compression and final compression rate and CPU time, regardless of the number of tiles used in MIMF-tr.

Furthermore, the three trends in Figure 7a–c show that these performances increase as the complexity of the image increases, i.e., the higher the number of levels required for compression, the higher the increase in the final compression rate and CPU time.

5. Conclusions

In this research, a variation of the Multilevel Fuzzy Transform image compression algorithm is proposed to optimize the compression of massive images. The image is decomposed into tiles; each tile is compressed separately by running MF-tr and the compressed tiles are subsequently merged to reconstruct the compressed image.

Our algorithm was tested on an extended sample of large satellite color images. The comparative results showed that, regardless of the number of tiles used, our method performed better than MF-tr both in terms of the number of levels required for compression, as well as the final compression rate and CPU time. Performance is better the more complex the image is, i.e., the greater the number of layers necessary to compress it.

These results highlight, in particular, two significant aspects:

- MIMF-tr is better suited than MF-tr for the compression of massive images; the strategy employed by the algorithm of partitioning the image into tiles and separating the compression of single tiles allow for the optimization of processing times and memory consumption;
- MIMF-tr provides better performance than MF-tr, reducing the CPU time by more than 20% and providing higher image compression than MF-tr, while ensuring the same reconstructed image quality.

In the future, we intend to carry out further comparative tests on a larger sample of massive images, to verify the performance and robustness of M-ftr in the presence of various types of noise in the image.

Author Contributions: Conceptualization, Ferdinando Di Martino (F.D.M.) and (Salvatore Sessa) S.S.; methodology, F.D.M. and S.S.; software, F.D.M. and S.S.; validation, F.D.M. and S.S.; formal analysis, F.D.M. and S.S.; investigation, F.D.M. and S.S.; resources, F.D.M. and S.S; data curation, F.D.M. and S.S.; writing—original draft preparation, F.D.M. and S.S.; writing—review and editing, F.D.M. and S.S.; visualization, F.D.M. and S.S.; supervision, F.D.M. and S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

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

References

1. Perfilieva, I. Fuzzy transforms. *Fuzzy Sets Syst.* **2006**, *157*, 993–1023.
2. Di Martino, F.; Sessa, S. Compression and decompression of images with discrete fuzzy transforms. *Inf. Sci.* **2007**, *17*, 2349–2362.
3. Di Martino, F.; Loia, V.; Perfilieva, I.; Sessa, S. An image coding/decoding method based on direct and inverse fuzzy transforms. *Int. J. Approx. Reason.* **2008**, *48*, 110–131.
4. Di Martino, F.; Loia, V.; Sessa, S. Fuzzy transforms for compression and decompression of colour videos. *Inf. Sci.* **2010**, *180*, 3914–3931.
5. Hodakova, P.; Perfilieva, I.; Dankova, M.; Vajgl, M. F-transform based image fusion. In *Image Fusion*, Ukimura, O., Ed.; IntechOpen: London, UK, 2011; 440 p, pp. 3–22, doi:10.5772/602.
6. Manchanda, M.; Sharma, R. An improved multimodal medical image fusion algorithm based on fuzzy transform. *J. Vis. Commun. Image Represent.* **2018**, *51*, 76–94.
7. Di Martino, F.; Sessa, S. Complete image fusion method based on fuzzy transforms. *Soft Comput.* **2019**, *23*, 2113–2123.
8. Roh, S.B.; Oh, S.K.; Pedrycz, W.; Seo, K. Development of autofocusing algorithm based on fuzzy transform. *Fuzzy Sets Syst.* **2019**, *288*, 129–144.
9. Di Martino, F.; Sessa, S. Passive image autofocus by using direct fuzzy transform. *Int. J. Comput. Sci. Eng.* **2019**, *20*, 240–254.
10. Daňková, M.; Hodáková P.; Perfilieva I.; Vajgl M. Edge detection using F-transform. In Proceedings of the 11th International Conference on Intelligent Systems Design and Applications, Córdoba, Spain, 22–24 November 2011; pp. 672–677, doi:10.1109/ISDA.2011.6121733.
11. Perfilieva, I.; Hodáková, P.; Hurtik, P. Differentiation by the F-transform and application for edge detection. *Fuzzy Sets Syst.* **2016**, *288*, 96–114.
12. Di Martino, F.; Loia, V.; Sessa, S. A segmentation method for images compressed by fuzzy transforms. *Fuzzy Sets Syst.* **2010**, *161*, 56–74.
13. Di Martino, F.; Sessa, S. PSO image thresholding on images compressed via fuzzy transforms. *Inf. Sci.* **2019**, *506*, 308–324.
14. Di Martino, F.; Sessa, S. Fragile watermarking tamper detection with images compressed by fuzzy transform. *Inf. Sci.* **2012**, *195*, 62–90.
15. Alibabaie, N.; Latif, A. Adaptive periodic noise reduction in digital images using fuzzy transform. *J. Math. Imaging Vis.* **2021**, *63*, 503–527.
16. Di Martino, F.; Sessa, S. Fuzzy transforms for image processing and data analysis. In *Fuzzy Transforms for Image Processing and Data Analysis: Core Concepts, Processes and Applications*; Springer: Cham, Germany, 2020; Volume 250, doi:10.1007/978-3-030-44613-0.
17. Di Martino F.; Sessa S. A Multi-level image compression method based on fuzzy transforms. *J. Ambient Intell. Humaniz. Comput.* **2019**, *10*, 2745–2756.
18. Boiangiu, C.A.; Cotofana, M.V.; Naiman, A.; Lambru, C. A generalized Laplacian Pyramid aimed at image compression. *J. Inf. Syst. Oper. Manag.* **2016**, *10*, 327–335.
19. K Khan, U.R.; Ahmed, S.; Nazeer, T. Wavelet based image compression techniques: Comparative analysis and performance evaluation. *Int. J. Emerg. Technol. Eng. Res.* **2017**, *5*, 9–13.
20. Karthikeyan, C.; Palanisamy, C. An efficient image compression method by using optimized discrete wavelet transform and Huffman encoder. *J. Comput. Theor. Nanosci.* **2018**, *15*, 289–298.
21. Di Martino, F.; Perfilieva, I.; Sessa, S. A fast multilevel fuzzy transform image compression method. *Axioms* **2019**, *8*, 135.
22. Indradjad, A.; Nasution, A.S.; Gunawan, H.; Widipaminto, A. A comparison of satellite image compression methods in the wavelet domain. In Proceedings of the IOP Conference Series Earth and Environmental Science, The 4th International Conference of Indonesian Society for Remote Sensing 30 October 2018, Makassar, Indonesia, 14–16 June 2019; Volume 280, p. 012031.
23. Hernández-Cabronero, M. The CCSDS 123.0-B-2 low-complexity lossless and near-lossless multispectral and hyperspectral image compression standard: A comprehensive review. *IEEE Geosci. Remote Sens. Mag.* **2021**, *9*, 102–119.
24. DeVore, R.A.; Jawerth, B.; Lucier, B.J. Image compression through wavelet transform coding. *IEEE Trans. Inf. Theory* **1992**, *38*, 719–746.
25. Mallat, S.; Peyré, G. Surface compression with geometric bandelet. ACM Transactions on Graphics. *Proc. SIGGRAPH'05* **2005**, *24*, 601–608.
26. Acharya, T.; Tsai Ping-Sing. *JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI architectures*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2004; 292 p, ISBN: 0-471-48422-9.
27. Monica, D.; Widipaminto, A. Fuzzy transform for high-resolution satellite images compression. *TELKOMNIKA Telecommun. Comput. Electron. Control* **2020**, *18*, 1130–1136.
28. Cardone, B.; Di Martino, F. Bit reduced FCM with block fuzzy transforms for massive image segmentation. *Information* **2020**, *11*, 351.
29. Earth Science Data Systems. NASA's Earth Science Data Systems (ESDS) Program Oversees the Life Cycle of NASA's Earth Science Data—From Acquisition through Processing and Distribution. Available online: https://search.earthdata.nasa.gov_(accessed on 01 July 2022).