mathematics

MDPI

Article

John von Neumann's Space-Frequency Orthogonal Transforms

Dan Stefanoiu [1,2] and Janetta Culita [1,*]

[1] Faculty of Automatic Control and Computers, National University of Science and Technology POLITEHNICA Bucharest, 313 Splaiul Independentei, 060042 Bucharest, Romania; dan.stefanoiu@upb.ro
[2] Academy of Romanian Scientists, Ilfov Str. No. 3, 050044 Bucharest, Romania
* Correspondence: janetta.culita@upb.ro

Abstract: Among the invertible orthogonal transforms employed to perform the analysis and synthesis of 2D signals (especially images), the ones defined by means of John von Neumann's cardinal sinus are extremely interesting. Their definitions rely on transforms similar to those employed to process time-varying 1D signals. This article deals with the extension of John von Neumann's transforms from 1D to 2D. The approach follows the manner in which the 2D Discrete Fourier Transform was obtained and has the great advantage of preserving the orthogonality property as well as the invertibility. As an important consequence, the numerical procedures to compute the direct and inverse John von Neumann's 2D transforms can be designed to be efficient thanks to 1D corresponding algorithms. After describing the two numerical procedures, this article focuses on the analysis of their performance after running them on some real-life images. One black and white and one colored image were selected to prove the transforms' effectiveness. The results show that the 2D John von Neumann's Transforms are good competitors for other orthogonal transforms in terms of compression intrinsic capacity and image recovery.

Keywords: orthogonal transforms; time/space-frequency dictionary; windowed Fourier Transforms; analysis and synthesis of images

MSC: 47N70; 40B99; 40C99; 40D99; 40E99

Citation: Stefanoiu, D.; Culita, J. John von Neumann's Space-Frequency Orthogonal Transforms. *Mathematics* **2024**, *12*, 767. https://doi.org/10.3390/math12050767

Academic Editor: Ivo Petráš

Received: 4 December 2023
Revised: 19 February 2024
Accepted: 26 February 2024
Published: 4 March 2024

1. Introduction

Bidimensional (2D) Signal Processing is an important and large subfield of *Signal Processing* (SP) [1,2] that encompasses a variety of methods and algorithms. Mathematically, the 2D signals are represented by means or matrices. Typical 2D signals are the still (photographic) images and the complex-valued signals stored in matrices (video streams are rather considered of the 3D or even of multi-dimensional type, where one dimension is always associated with time). Therefore, (still) image processing is a special branch of 2D SP. This article aims to introduce *John von Neumann's Transform* (JvNT) devoted to the analysis and synthesis of such signals. The construction of 2D JvNT relies on two other transforms: the (well known) *Fourier Transform* (FT) and the 1D JvNT, described at length in [3].

Like in the case of 1D signals, the framework of 2D signals constitutes Lebesgue spaces for which both the FT and the scalar product can be defined, as mentioned in [3]. Usually, the bases of 2D signals are generated either by using naturally born 2D waveforms or by coupling two sets of 1D waveforms (one dealing with rows and the other one approaching the columns of a matrix). The orthogonality constraint generally is difficult to verify in the case of 2D waveforms. Therefore, the second approach is often preferred, as the orthogonality of 1D waveforms can easily be preserved.

The extension of the ideal Karhunen–Loève Transform from 1D to 2D is carried out using the first approach, though (see [2]). Instead of working with eigenvalues and eigenvectors of the autocorrelation matrix, like in the case of 1D signals, for 2D signals,

some scientists employ singular value decomposition to extract the principal components. Other scientists preserve the 1D approach, but the autocorrelation matrix is computed through multiplication of the signal matrix by the transposed matrix, after extracting the average.

In the case of transforms based on orthogonal polynomials (as mentioned in [3]), the second approach is adopted to jump into the 2D signal spaces. Nevertheless, the complexity of such transforms can tremendously increase, as shown, for example, in [4] and, very recently, in [5].

Beside the ideal and the polynomial-based transforms, the world of 2D orthogonal transforms can be structured into four classes, like in the case of 1D transforms [3], by replacing the time domain with the space domain: harmonic, space-frequency, space-scale, space-frequency-scale. Basically, any 1D transform can be extended to a 2D transform, under the same name.

The harmonic class is naturally generated by the FT. The extension from 1D to 2D is performed using the "two 1D sets" approach, which will be shortly reviewed within the next section. Despite its age, this transform continues to be employed in applications (although some approaches reported nowadays in the literature are rather naïve). For example, in [6], the 2D FT symmetry properties are analyzed. Symmetry is an important and desirable property of transforms, and can help reduce the computational burden. (Unfortunately, not all transforms are symmetric or conjugate-symmetric like FT. Fortunately, both 1D and 2D JvNTs are verified to possess such properties.) Besides 2D FT, two other versions have been defined: *Windowed Fourier Transform* (WFT, which is quite old, too) [7] and *fractional Fourier Transform* (frFT, which was quite recently introduced) [8,9]. In many applications, and especially in modern signal compression technology, 2D FT was replaced by the family of 2D Discrete Cosine Transforms, for which the analysis coefficients are real-valued (and not complex-valued, like in the case of FT). Note, however, that the cosine coefficients are not necessarily symmetric. Recent publications, like [10–13], prove that the employment of these transforms still is quite intense.

Space-frequency transforms also are obtained by windowing the signals. Sometimes, windowed transforms are referred to as *smooth wavelet transforms*. JvNTs are actually such transforms, where the working window is the *John von Neumann* (JvN) cardinal sinus. Besides JvNTs, some other remarkable transforms are reported in the literature. For example, the old Morlet wavelet (well known as the *Mexican hat*), which initially was devoted to geological prospections, is employed nowadays in image compression [14]. Another example is given by the Gauss–Gabor wavelet, which was extended to 2D signals in [15] using an interesting approach.

The class of space-scale transforms is quite large and, usually, is split into two groups: empirical transforms and generic transforms. Fractal 2D signals are mainly processed with such transforms. In fact, the fractal wavelets constitute the foundation of this class. The old Haar wavelet (also known as the *lazy wavelet*), as a typical empirical transform, works very well with images, as recently reported in [16,17]. The more sophisticated Walsh–Hadamard Transform (still empirical, though) can also be adapted to process images—see, for example, [18,19] (where it works in combination with Haar wavelet). Another empirical transform with increasing impact in image processing is the (less known) Slant Transform [20]. Although the extension of this transform to 2D signals is not easy, it was employed to denoise images, like in [21], and to process video streams (among other transforms), as recently reported in [22].

Without any doubt, nowadays, generic (fractal) orthogonal and biorthogonal wavelets are the most employed tools in image processing. Since their inception by Meyer, Mallat and Daubechies (more than 35 years ago), the literature reporting how these wavelets work in conjunction with images has become so vast that any attempt to encompass all the sound (and sometimes amazing) results is very likely doomed to fail. One can only cite a few interesting recent works, such as [23] (with application to image filtering), [24] (in which principal component analysis is developed by means of wavelets, aiming to achieve image

denoising), [25] (where image fusion is performed) and [26] (which deals with the efficient implementation of 2D wavelet transforms).

Transforms of the space-frequency-scale class exhibit great complexity and usually are non-orthogonal. Basically, any space-frequency or space-scale transform can evolve towards this class (including JvNT). The basis of the Lebesgue space is generated by means of all three operators: time-shifting, frequency modulation and scaling. They are applied (in this order) to a mother waveform (or wavelet) and generate a whole dictionary of space-frequency-scale atoms. Analysis of orthogonality and invertibility of the transform based on such a dictionary is a difficult task (and an open problem). Working with non-orthogonal bases is, however, necessary in many applications where image interpretation is required (like in medicine, geography, and underwater or space exploration). In this case, in opposition with image compression or denoising, keeping a high level of image redundancy is crucial.

By determining the difference from 1D transforms, the collection of 2D transforms was enriched with hybrid transforms, born through a combination of two or more already known transforms. For example, one can find a joint Cosine and Karhunen–Loève Transform in [27], a joint Cosine and Hadamard Transform in [28], and a joint Walsh and fractional Fourier Transform in [29].

The exploration of the scientific literature performed within this introduction is far from complete. But the goal was not to review the entire 2D SP subfield. This study aimed to correctly place JvNT in the wide world of orthogonal transforms and to show that, to the best of our knowledge, no other article in the literature has reported a similar approach and similar results to this article.

The following sections are presented next. In Section 2, after a short overview of 1D JvNT (already introduced in [3]), the extension to space signals is described. We prove that working with two 1D dictionaries is equivalent to working with a naturally born 2D dictionary. Section 3 is devoted to the design of the analysis and synthesis of numerical algorithms that allow for the efficient implementation of 2D JvNT. The simulation results and a discussion of the obtained results can be found in Section 4. The algorithms of Section 3 were implemented and tested on two types of images: black and white, and color. Some concluding remarks, an acronym list and a reference list complete the article.

2. Theoretical Background

2.1. Short Overview of JvNT for Discrete Time 1D Signals

The use of JvNT for 1D signals was described at length in [3]. Within this section, only a short overview of main notations and results from [3] is presented. The starting point is the cardinal sinus below (which the great scientist JvN defined long time ago [30]):

$$\nu(t) = \mathrm{sinc}(\pi t) = \frac{\sin(\pi t)}{\pi t},\ \forall t \in \mathrm{R}. \tag{1}$$

A dictionary of *time-frequency atoms* (tfas) can be built by applying time-shifting, frequency modulation and discretization to Function (1). More specifically, the generic tfas can be expressed as follows:

$$\nu_{T_s}^{[p,k]}[n] = \mathrm{e}^{2\pi k n T_s \mathrm{j}} \mathrm{sinc}[\pi(nT_s - p)],\ \forall n, p \in \mathrm{Z},\ \forall k \in \mathrm{N}, \tag{2}$$

where $T_s \in (0, 1]$ is the sampling period, n is the discrete (normalized) time, p is the time-shifting index and k is the frequency modulation index. In [3], it is proven that if $T_s = 1/K$, where $K \in \mathrm{N}^*$ is a sampling frequency (with integer values measured in Hz), then the reduced dictionary $\mathbf{V}_K = \left\{ \nu_{T_s}^{[p,k]} \right\}_{p \in \mathrm{Z}, k \in \overline{0,K-1}}$ is an orthogonal basis of the Lebesgue–Hilbert

space $\mathbf{l}^2$ (of discrete time signals). In this case, the notation $\nu_{T_s}^{[p,k]}$ changes to $\nu_K^{[p,k]}$, and Equation (2) becomes

$$\nu_K^{[p,k]}[n] = \mathrm{e}^{\frac{2nk\pi}{K}\mathrm{j}}\mathrm{sinc}\Big[\pi\Big(\frac{n}{K}-p\Big)\Big],\ \forall n,p\in\mathrm{Z},\ \forall k\in\overline{0,K-1}. \tag{3}$$

Consequently, the discrete JvNT is obtained by using the orthogonal basis $\mathbf{V}_K$. Thus, the JvN coefficients of the signal $x\in\mathbf{l}^2$ can be computed by projecting it on each tfa:

$$\mathbf{N}_K(x)[p,k] = X_K[p,k] = \Big\langle x,\nu_K^{[p,k]}\Big\rangle = \sum_{n\in\mathrm{Z}} x[n]\overline{\nu_K^{[p,k]}[n]} = \sum_{n\in\mathrm{Z}} x[n]\mathrm{sinc}\Big[\pi\Big(\frac{n}{K}-p\Big)\Big]\mathrm{e}^{-\frac{2kn\pi}{K}\mathrm{j}},\forall p\in\mathrm{Z},\ \forall k\in\overline{0,K-1}. \tag{4}$$

To recover the signal, expansion on the basis of $\mathbf{V}_K$ with coefficient (4) is employed:

$$x[n] = \frac{1}{K}\sum_{p\in\mathrm{Z}}\sum_{k=0}^{K-1} X_K[p,k]\nu_K^{[p,k]}[n] = \frac{1}{K}\sum_{p\in\mathrm{Z}}\sum_{k=0}^{K-1} X_K[p,k]\mathrm{sinc}\Big[\pi\Big(\frac{n}{K}-p\Big)\Big]\mathrm{e}^{\frac{2kn\pi}{K}\mathrm{j}},\ \forall n\in\mathrm{Z}. \tag{5}$$

If the signal $x\in\mathbf{l}^2$ is real-valued, the following remarkable property is verified:

$$X_K[p,K-k] = \overline{X_K[p,k]},\ \forall p\in\mathrm{Z},\ \forall k\in\overline{0,K-1}. \tag{6}$$

This means the JvN coefficients are conjugate-symmetric, which allows for a reduction in the computational burden by only evaluating approximately half of them. Moreover, the synthesis Equation (5) can benefit from symmetry in implementation.

In Definition (4) and Equation (5), signals from $\mathbf{l}^2$ either have infinite or finite support. In the second case, signals not only have finite energy, but are also stable (absolutely summable), i.e., they also belong to $\mathbf{l}^1\subset\mathbf{l}^2$. Consequently, *Fourier Transform* (FT), and especially *Discrete Fourier Transform* (DFT), can be computed for such signals.

Also, for finite support signals, the JvNT yields when working with a finite number of JvN coefficients, as shown in [3]. More specifically, depending on the signal length N and accuracy threshold $\varepsilon > 0$ (which allows for truncation of the signal (1) at the user's convenience), the lower and upper bounds of time-shifting index p can be derived: $p\in\overline{P_{\min},P_{\max}}$. The upper limit of the frequency modulation index, K, depends on N, as well, and can be set according to the Gabor–Heisenberg Uncertainty Principle [2], such that the time and frequency resolutions of the JvN spectrogram be approximately equal. For example, this requirement is fulfilled if $K=\Big\lfloor\sqrt{2N}+0.5\Big\rfloor$.

Although the synthesis of a finite-length signal with truncated tfas is not exact (except for the normalized instants located at integer multiples of K), the reconstruction accuracy can be controlled by the user through the parameters ε and K.

The extension of JvNT to spatial signals (images) strongly relies on the properties above of JvNT devoted to 1D signals.

2.2. JvNT for Discrete-Space 2D Signals

Although it is possible to develop a theory of continuous-space JvNT for 2D signals, the practical importance of such a transform is rather small. Nowadays, 2D signals (especially images) are digitally represented, processed and transmitted. Therefore, the following development only concerns digital signals, seen as (still) images.

Let $\mathbf{A}\in\mathrm{R}^{NR\times NC}$ be a matrix playing the role of a 2D discrete spatial signal. The matrix has NR rows and NC columns. If $\mathbf{A}$ is a digital representation of some (still) *black and white* (bw) image, then its elements are referred to as *pixels*. Hereafter, by convention, the term of *pixel* will be associated with the current element $a_{nr,nc}$ of matrix $\mathbf{A}$, located at coordinates (nr,nc) (for $nr\in\overline{1,NR}$ and $nc\in\overline{1,NC}$). In SP, such a matrix can be transformed in two ways: by means of a dictionary with 2D genuinely born atoms or by means of two dictionaries with 1D atoms.

In the first approach, usually, the atoms of the dictionary are obtained by spatially rotating a 1D *mother waveform* (mw). For example, in the case of JvN mw (1) and the matrix **A**, one (non-unique) rotated version is as follows:

$$\nu(x,y) = \operatorname{sinc}\left(\pi\sqrt{\frac{2NR}{NR+NC}x^2 + \frac{2NC}{NR+NC}y^2}\right),\ \forall x,y \in \mathrm{R}. \tag{7}$$

If **A** is a square matrix ($NR = NC$), then the mw (7) becomes

$$\nu(x,y) = \operatorname{sinc}\left(\pi\sqrt{x^2+y^2}\right),\ \forall x,y \in \mathrm{R}. \tag{8}$$

In the top image in Figure 1, the shape of mw (8) (for $NR = NC$) is drawn, while at the bottom, 2D FT is represented (spectrogram and phase surface).

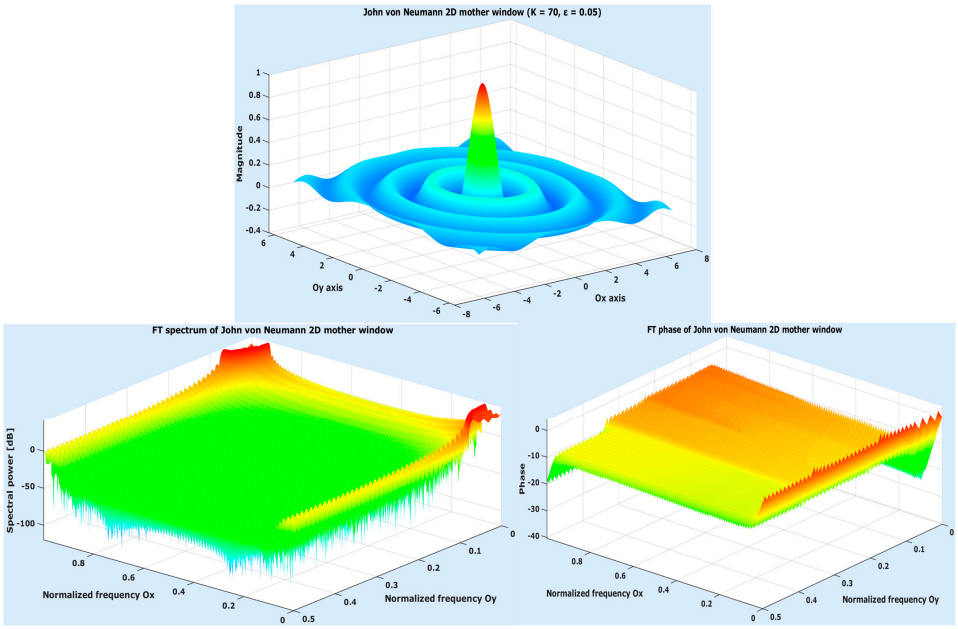

Figure 1. JvN 2D rotation window on (**top**) together with its FT spectrogram at (**bottom**, **left side**) and phase at (**bottom**, **right side**).

The imperfections in the displayed spectrogram are due to the truncation of mw. In fact, the two dominant corners of the spectral surface are cylindrical. This property sensibly complicates the analysis of orthogonality. Such an analysis, although interesting, is not developed within this article.

The second approach is more promising due to the strong similarity with 2D FT. Recall that the 2D DFT of matrix **A** is defined as below [1,2]:

$$\mathbf{F}(\mathbf{A})(\omega_x,\omega_y) = \sum_{nr=0}^{NR-1}\sum_{nc=0}^{NC-1} a_{nr+1,nc+1}\mathrm{e}^{-\mathrm{j}(nr\omega_x+nc\omega_y)} = \sum_{nr=0}^{NR-1}\sum_{nc=0}^{NC-1} a_{nr+1,nc+1}\mathrm{e}^{-\mathrm{j}nr\omega_x}\mathrm{e}^{-\mathrm{j}nc\omega_y},\ \forall \omega_x,\omega_y \in [-\pi,+\pi]. \tag{9}$$

The last expression in definition (9) allows us to write the following matrix form:

$$\mathbf{F}(\mathbf{A})(\omega_x,\omega_y) = \overline{\mathbf{e}_{NR}^T(\omega_x)}\mathbf{A}\overline{\mathbf{e}_{NC}(\omega_y)},\ \forall \omega_x,\omega_y \in [-\pi,+\pi], \tag{10}$$

where, by definition:

$$\mathbf{e}_N^T(\omega) = \left[1\ \ \mathrm{e}^{\mathrm{j}\omega}\ \ \mathrm{e}^{2\mathrm{j}\omega}\ \ \ldots\ \ \mathrm{e}^{\mathrm{j}(N-1)\omega}\right],\ \forall \omega \in [-\pi,+\pi]. \tag{11}$$

Equation (10) reveals that two harmonic vectors can be employed to compute the FT: one operating on matrix columns and another one operating on matrix rows. This property suggests that two dictionaries of orthogonal 1D tfas can be built (3): one associated with matrix columns and another one associated with matrix rows. Each dictionary is configured by some sampling rate, namely KR and KC, respectively. It is not necessary that the parameters KR and/or KC be directly correlated with the matrix sizes NR and NC. Nevertheless, according to the previous subsection, one can set $KR = \left\lfloor \sqrt{2NR} + 0.5 \right\rfloor$ and $KC = \left\lfloor \sqrt{2NC} + 0.5 \right\rfloor$. Also, the time-shifting index p of the previous section becomes, in this context, a *space-shifting index*. Since the matrix has finite sizes, the bounds of the space-shifting indices can be evaluated for rows, as well as for columns, and they are not necessarily the same. They are denoted as $PR_{\min}$ and $PR_{\max}$ for columns (of length NR) and by $PC_{\min}$ and $PC_{\max}$ for rows (of length NC). Usually, the accuracy threshold ε is unique (for example, set to 1%).

Definition 1. *The practical JvNT of 2D signals is defined as follows:*

$$\mathbf{N}_{KR,KC}(\mathbf{A})[pr,kr,pc,kc] = \left(\overline{\nu_{KR}^{[pr,kr]}}\right)^T \mathbf{A}\overline{\nu_{KC}^{[pc,kc]}}, \forall pr \in \overline{PR_{\min}, PR_{\max}},\ \forall kr \in \overline{0, KR-1},\ \forall pc \in \overline{PC_{\min}, PC_{\max}},\ \forall kc \in \overline{0, KC-1}, \quad (12)$$

where

$$\begin{cases} \nu_{KR}^{[pr,kr]} = \left[\ \nu_{KR}^{[pr,kr]}[0]\ \nu_{KR}^{[pr,kr]}[1]\ \nu_{KR}^{[pr,kr]}[2]\ \cdots\ \nu_{KR}^{[pr,kr]}[NR-1]\ \right]^T \\ \nu_{KC}^{[pc,kc]} = \left[\ \nu_{KC}^{[pc,kc]}[0]\ \nu_{KC}^{[pc,kc]}[1]\ \nu_{KC}^{[pc,kc]}[2]\ \cdots\ \nu_{KC}^{[pc,kc]}[NC-1]\ \right]^T \end{cases}, \quad (13)$$

$\forall pr \in \overline{PR_{\min}, PR_{\max}},\ \forall kr \in \overline{0, KR-1},\ \forall pc \in \overline{PC_{\min}, PC_{\max}},\ \forall kc \in \overline{0, KC-1}.$

Although not obvious, the definition above is equivalent to working in 2D signal space with a 2D JvN function:

$$\nu_{2D}(x,y) = \operatorname{sinc}(\pi x)\operatorname{sinc}(\pi y) = \nu(x)\nu(y),\ \forall x,y \in \mathrm{R}. \quad (14)$$

Recall that Function (1) is also the impulse response of an ideal *Low-Pass Filter* (LPF). This property is inherited by the discrete signal $\nu_{T_s}^{[0,0]}$, regardless of the value of the sampling period $T_s \in (0,1]$. More specifically, assume the following ideal digital LPF has a frequency response expressed as

$$H_1\left(\mathrm{e}^{\mathrm{j}\omega}\right) = \begin{cases} 1\,, \omega \in [-\omega_c, \omega_c] \\ 0\,, \omega \in [-\pi, +\pi] \backslash [-\omega_c, \omega_c] \end{cases} = \chi_{[-\omega_c,\omega_c]}(\omega),\ \forall \omega \in [-\pi, +\pi], \quad (15)$$

where $\chi_{[a,b]}$ is the index function of interval $[a,b]$ and $\omega_c = \pi T_s \in (0,\pi)$ is the cut-off normalized pulsation. The impulse response of the filter is then

$$h_1[n] = \frac{\omega_c}{\pi}\nu_{T_s}^{[0,0]}\left(\frac{\omega_c}{\pi}n\right),\ \forall n \in \mathrm{Z}. \quad (16)$$

In the 2D space of discrete signals, Equations (15) and (16) become

$$H_2\left(\mathrm{e}^{\mathrm{j}\omega_x}, \mathrm{e}^{\mathrm{j}\omega_y}\right) = \begin{cases} 1\,, (\omega_x, \omega_y) \in [-\omega_{cx}, \omega_{cx}] \times [-\omega_{cy}, \omega_{cy}] \\ 0\,, \text{otherwise} \end{cases} = \chi_{[-\omega_{cx},\omega_{cx}]}(\omega_x)\chi_{[-\omega_{cy},\omega_{cy}]}(\omega_y) = \\ = H_1\left(\mathrm{e}^{\mathrm{j}\omega_x}\right)H_1\left(\mathrm{e}^{\mathrm{j}\omega_y}\right), \forall \omega_x, \omega_y \in [-\pi, +\pi]; \quad (17)$$

$$h_2[nx, ny] = \frac{\omega_{cx}\omega_{cy}}{\pi^2}\nu_{2D}\left(\frac{\omega_{cx}}{\pi}nx, \frac{\omega_{cy}}{\pi}ny\right) = \frac{\omega_{cx}\omega_{cy}}{\pi^2}\nu\left(\frac{\omega_{cx}}{\pi}nx\right)\nu\left(\frac{\omega_{cy}}{\pi}ny\right),\ \forall nx, ny \in \mathrm{Z}. \quad (18)$$

In Equations (17) and (18), $\omega_{cx} = \pi X_s$ and $\omega_{cy} = \pi Y_s$, where $X_s \in (0,1]$ and $Y_s \in (0,1]$ are sampling distances along the Ox and Oy axes, respectively. Thus, in fact, Equation (18) is equivalent to:

$$h_2[nx, ny] = \frac{\omega_{cx}\omega_{cy}}{\pi^2}\nu_{X_s}^{[0,0]}\left(\frac{\omega_{cx}}{\pi}n\right)\nu_{Y_s}^{[0,0]}\left(\frac{\omega_{cy}}{\pi}n\right) = h_1[nx]h_1[ny],\ \forall nx, ny \in \mathrm{Z}. \tag{19}$$

While in the space of 1D signals, the LPF frequency characteristic is a rectangle over the $[-\omega_c, \omega_c]$ band, and in the space of 2D signals, it is a parallelepiped over the $[-\omega_{cx}, \omega_{cx}] \times [-\omega_{cy}, \omega_{cy}]$ band (rectangular). Orthogonality can be analyzed by shifting the parallelepiped along the pulsation axes such that the rectangular bands are almost disjunct (i.e., they intersect most along a vertical plane or a vertical line).

The only problem is that of defining appropriate scalar products in discrete space and frequency. In general, defining scalar products between 2D signals (i.e., between matrices) is a difficult task. Fortunately, as Equations (17) and (19) clearly reveal, both the impulse response and the frequency response can be expressed in factorial form with the help of 1D signals, one for each space or frequency axes. Hence, one can write

$$h_2 \equiv h_1^{\mathrm{Ox}} h_1^{\mathrm{Oy}},\ H_2 \equiv H_1^{\mathrm{Ox}} H_1^{\mathrm{Oy}}, \tag{20}$$

with natural notations. Identity (20) allows us to define special forms of scalar products in the context of 2D signals with the help of scalar products already defined in the context of 1D signals:

$$\begin{cases} \langle h_2, g_2 \rangle = \left\langle h_1^{\mathrm{Ox}} h_1^{\mathrm{Oy}}, g_1^{\mathrm{Ox}} g_1^{\mathrm{Oy}} \right\rangle = \langle h_1^{\mathrm{Ox}}, g_1^{\mathrm{Ox}} \rangle \left\langle h_1^{\mathrm{Oy}}, g_1^{\mathrm{Oy}} \right\rangle \\ \langle H_2, G_2 \rangle = \left\langle H_1^{\mathrm{Ox}} H_1^{\mathrm{Oy}}, G_1^{\mathrm{Ox}} G_1^{\mathrm{Oy}} \right\rangle = \langle H_1^{\mathrm{Ox}}, G_1^{\mathrm{Ox}} \rangle \left\langle H_1^{\mathrm{Oy}}, G_1^{\mathrm{Oy}} \right\rangle \end{cases}. \tag{21}$$

Definition (21), together with Definition (17) and Equation (19), enable the analysis of the orthogonality of 2D tfas by means of the orthogonality of corresponding 1D tfas. All the orthogonality and invertibility results proven in [3] for the 1D time-frequency dictionaries of JvN atoms can straightforwardly be transferred to 2D space-frequency dictionaries. The generic *space-frequency atom* (sfa) of the JvN 2D dictionary is as follows:

$$\begin{gathered} \nu_{KR,KC}^{[pr,pc,kr,kc]}[nr, nc] = \nu_{KR}^{[pr,kr]}[nr]\nu_{KC}^{[pr,kr]}[nc], \\ \forall pr \in \overline{PR_{\min}, PR_{\max}},\ \forall kr \in \overline{0, KR-1},\ \forall pc \in \overline{PC_{\min}, PC_{\max}},\ \forall kc \in \overline{0, KC-1}. \end{gathered} \tag{22}$$

Thanks to the factorization exhibited in Equation (22), the projection of matrix $\mathbf{A}$ on any sfa of the dictionary can be defined as follows:

$$\begin{gathered} \left\langle \mathbf{A}, \nu_{KR,KC}^{[pr,pc,kr,kc]}[nr, nc] \right\rangle = \sum_{nr=0}^{NR-1}\sum_{nc=0}^{NC-1} a_{nr+1,nc+1}\nu_{KR}^{[pr,kr]}[nr]\nu_{KC}^{[pr,kr]}[nc], \\ \forall pr \in \overline{PR_{\min}, PR_{\max}},\ \forall kr \in \overline{0, KR-1},\ \forall pc \in \overline{PC_{\min}, PC_{\max}},\ \forall kc \in \overline{0, KC-1}. \end{gathered} \tag{23}$$

At the same time, Definition (12) can be expressed at length:

$$\begin{gathered} \mathbf{N}_{KR,KC}(\mathbf{A})[pr, kr, pc, kc] = \sum_{nr=0}^{NR-1}\sum_{nc=0}^{NC-1} a_{nr+1,nc+1}\nu_{KR}^{[pr,kr]}[nr]\nu_{KC}^{[pr,kr]}[nc] = \left\langle \mathbf{A}, \nu_{KR,KC}^{[pr,pc,kr,kc]}[nr, nc] \right\rangle, \\ \forall pr \in \overline{PR_{\min}, PR_{\max}},\ \forall kr \in \overline{0, KR-1},\ \forall pc \in \overline{PC_{\min}, PC_{\max}},\ \forall kc \in \overline{0, KC-1}. \end{gathered} \tag{24}$$

From Equation (24), it results that the analysis of 2D signal $\mathbf{A}$ is equivalently performed by using a couple of 1D dictionaries (according to Definition (12)) or with the help of sfas taken from a 2D dictionary (according to Definition (23)).

Furthermore, Equation (24) can be expressed in implementation form below:

$$\begin{gathered} \mathbf{N}_{KR,KC}(\mathbf{A})[pr, kr, pc, kc] = \sum_{nr=0}^{NR-1}\sum_{nc=0}^{NC-1} a_{nr+1,nc+1}\mathrm{sinc}\left[\pi\left(\tfrac{nr}{KR} - pr\right)\right]\mathrm{sinc}\left[\pi\left(\tfrac{nc}{KC} - pc\right)\right]\mathrm{e}^{-2\pi\left(\frac{krnr}{KR} + \frac{kcnc}{KC}\right)\mathrm{j}}, \\ \forall pr \in \overline{PR_{\min}, PR_{\max}},\ \forall kr \in \overline{0, KR-1},\ \forall pc \in \overline{PC_{\min}, PC_{\max}},\ \forall kc \in \overline{0, KC-1}. \end{gathered} \tag{25}$$

The JvN coefficients (25) can be grouped into a 4D array and cannot be represented visually. To compute them, the previously mentioned symmetry property can be exploited for each harmonic index, provided that the 2D signal is real-valued.

As already mentioned, the orthogonality of the 2D dictionary involves the orthogonality of the transform (12). Consequently, the practical inverse of JvNT can be computed like in [3]. More specifically, the extension to spatial signals is expressed as follows:

a. if $nr = mr{\cdot}KR \in KR\mathbb{N} \cap \overline{0, NR-1}$ and $nc = mc{\cdot}KC \in KC\mathbb{N} \cap \overline{0, NC-1}$:

$$\widetilde{a}[mrKR+1, mcKC+1] = \frac{1}{KR{\cdot}KC} \sum_{kr=0}^{KR-1} \sum_{kc=0}^{KC-1} \mathbf{N}_{KR,KC}(\mathbf{A})[mr, kr, mc, kc]; \tag{26}$$

b. if $nr = mr{\cdot}KR \in KR\mathbb{N} \cap \overline{0, NR-1}$ and $nc \in \overline{0, NC-1} \backslash KC\mathbb{N}$:

$$\widetilde{a}[mrKR+1, nc+1] = \frac{\sum\limits_{pc=PC_{\min}}^{PC_{\max}} \sum\limits_{kc=0}^{KC-1} \left(\sum\limits_{kr=0}^{KR-1} \mathbf{N}_{KR,KC}(\mathbf{A})[mr, kr, pc, kc] \right) \nu_{KC}^{[pc,kc]}[nc]}{KR{\cdot}KC \sum\limits_{pc=PC_{\min}}^{PC_{\max}} \left(\nu_{KC}^{[pc,0]}[nc] \right)^2}; \tag{27}$$

c. if $nr \in \overline{0, NR-1} \backslash KR\mathbb{N}$ and $nc = mc{\cdot}KC \in KC\mathbb{N} \cap \overline{0, NC-1}$:

$$\widetilde{a}[nr+1, mcKC+1] = \frac{\sum\limits_{pr=PR_{\min}}^{PR_{\max}} \sum\limits_{kr=0}^{KR-1} \left(\sum\limits_{kc=0}^{KC-1} \mathbf{N}_{KR,KC}(\mathbf{A})[pr, kr, mc, kc] \right) \nu_{KC}^{[pc,kc]}[nc]}{KR{\cdot}KC \sum\limits_{pr=PR_{\min}}^{PR_{\max}} \left(\nu_{KR}^{[pr,0]}[nr] \right)^2}; \tag{28}$$

d. if $nr \in \overline{0, NR-1} \backslash KR\mathbb{N}$ and $nc \in \overline{0, NC-1} \backslash KC\mathbb{N}$:

$$\widetilde{a}[nr+1, nc+1] = \frac{\sum\limits_{pr=PR_{\min}}^{PR_{\max}} \sum\limits_{kr=0}^{KR-1} \sum\limits_{pc=PC_{\min}}^{PC_{\max}} \sum\limits_{kc=0}^{KC-1} \mathbf{N}_{KR,KC}(\mathbf{A})[pr, kr, pc, kc] \nu_{KR}^{[pr,kr]}[nr] \nu_{KC}^{[pc,kc]}[nc]}{KR{\cdot}KC \left[\sum\limits_{pr=PR_{\min}}^{PR_{\max}} \left(\nu_{KR}^{[pr,0]}[nr] \right)^2 \right] \left[\sum\limits_{pc=PC_{\min}}^{PC_{\max}} \left(\nu_{KC}^{[pc,0]}[nc] \right)^2 \right]}. \tag{29}$$

(The notations $KR\mathbb{N}$ and $KC\mathbb{N}$ stand for the sets of all non-negative integer multiples of numbers KR and KC, respectively).

The reconstructed matrix $\widetilde{\mathbf{A}} = [\widetilde{a}_{nr,nc}]_{\substack{nr \in \overline{1,NR} \\ nc \in \overline{1,NC}}}$ approximates the original matrix $\mathbf{A}$. Since the threshold ε is constant, the accuracy of approximation $\widetilde{\mathbf{A}}$ only depends on the two sampling rates KR and KC, which are set according to the Uncertainty Principle. To evaluate the accuracy, the following cost function can be employed:

$$\mathbf{A}(KR, KC, \varepsilon) = \frac{100}{1 + 10 \frac{\|\mathbf{A} - \widetilde{\mathbf{A}}\|_2}{\|\mathbf{A}\|_2}} \ [\%], \ \forall KR, KC \in \mathbb{N}^*, \ \forall \varepsilon > 0, \tag{30}$$

where $\|\mathbf{A}\|_2$ is the 2-norm of matrix $\mathbf{A}$ (the largest singular value). Definition (30) employs the function $1/(1+x)$, which compresses the interval $[0, +\infty)$ into the interval $(0, 1]$. The cost Function (30) is often referred to as *fitness*.

Equations (26)–(29) can involve significant computational burden, depending on the size of the original matrix. To reduce the runtime of the corresponding numerical algorithm, the symmetry properties should be accounted for at the expense of higher programming effort. Working with arrays with more than two dimensions is time-consuming within almost all programing environments. Therefore, using any property that can reduce the computational burden is worthwhile.

3. Numerical Algorithms to Implement 2D JvNT

Two algorithms were designed and implemented based on the previous section. Only the case of real-valued matrices was considered for direct 2D JvNT, because complex-valued

matrices are unusual in real-life applications. Nevertheless, an algorithm for complex-valued matrices can be designed according to general definition (4). However, another procedure was designed to implement the inverse 2D JvNT. Although the genuine matrix to achieve approximately recovery is real-valued, the 4D array of JvN coefficients includes complex-valued values. In this case, Equations (26)–(29) were employed to design the corresponding algorithm.

Both algorithms were designed to allow implementation within the MATLAB™ programming environment (version 2018), although other high-level languages can be employed, as well (such as C++ or Python).

3.1. Direct JvNT Algorithm for Real-Valued 2D Signals

The analysis algorithm requires some preliminary preparations. The aim is to design a procedure that invokes Algorithm 1 from [3]. In this way, most of the computations are performed just like in the case of 1D signals. Thus, the symmetry property can fully be exploited (the direct implementation of Equation (4) is also possible, even when considering the symmetry).

The computations in Equation (4) can be organized by using the matrix formalism. Thus, the equivalent expression of JvN analysis coefficients is as follows:

$$\mathbf{N}_{KR,KC}(\mathbf{A})[pr,kr,pc,kc]=\underbrace{\sum_{nr=0}^{NR-1}\operatorname{sinc}\Big[\pi\Big(\frac{nr}{KR}-pr\Big)\Big]\mathrm{e}^{-\frac{2\pi krnr}{KR}\mathrm{j}}\underbrace{\Bigg(\sum_{nc=0}^{NC-1}a_{nr+1,nc+1}\operatorname{sinc}\Big[\pi\Big(\frac{nc}{KC}-pc\Big)\Big]\mathrm{e}^{-\frac{2\pi kcnc}{KC}\mathrm{j}}\Bigg)}_{\text{TvN1D }-\text{R}}}_{\text{TvN1D }-\mathbb{C}}\tag{31}$$
$$\forall pr\in\overline{PR_{\min},PR_{\max}},\ \forall kr\in\overline{0,KR-1},\ \forall pc\in\overline{PC_{\min},PC_{\max}},\ \forall kc\in\overline{0,KC-1}.$$

In Equation (31), two JvN 1D Transforms are emphasized: one that operates with real-valued signals and another one that deals with complex-valued signals. Focus first on the inner JvNT. Practically every row of the real-valued matrix $\mathbf{A}$ produces a matrix with complex-valued coefficients:

$$c_{KC}^{nr+1}[pc-PC_{\min}+1,kc+1]=\mathbf{N}_{KC}(\operatorname{row}_{nr+1}(\mathbf{A}))[pc,kc]=\sum_{nc=0}^{NC-1}a_{nr+1,nc+1}\operatorname{sinc}\big[\pi\big(\tfrac{nc}{KC}-pc\big)\big]\mathrm{e}^{-\frac{2\pi kcnc}{KC}\mathrm{j}},\tag{32}$$
$$\forall nr\in\overline{0,NR-1},\ \forall pc\in\overline{PC_{\min},PC_{\max}},\ \forall kc\in\overline{0,KC-1}.$$

All matrices from Equation (32), $\left\{\mathbf{C}_{KC}^{nr+1}\right\}_{nr\in\overline{0,NR-1}}$, can be packed into a 3D array, $\mathbf{C}_{KC}$, where each matrix is a layer. The sizes of the array $\mathbf{C}_{KC}$ are as follows: $PC=PC_{\max}-PC_{\min}+1$ rows, KC columns and NR layers. Unfortunately, the second JvNT does not act on the columns of each layer, but on vectors extracted from the array $\mathbf{C}_{KC}$ along the layers. This is because the outer JvNT in Equation (31) is configured to work with NR-length signals. Thus, the 3D array should be reshaped into a large matrix, say $\mathbf{D}_{KC}$, with NR rows and $M=PC{\cdot}KC$ columns. The matrix $\mathbf{D}_{KC}$ can be obtained by linearizing each matrix $\mathbf{C}_{KC}^{nr+1}$ along its rows and stacking the resulting long rows from top to bottom. Linearization can be implemented by using the well-known *Theorem of Division with Remainder* (TDR) between integers. Hence, the current element of the matrix $\mathbf{D}_{KC}$ is:

$$d_{KC}[nr+1,m+1]=c_{KC}^{nr+1}\Big[\Big\lfloor\frac{m}{KC}\Big\rfloor+1,m\%KC+1\Big],\ \forall nr\in\overline{0,NR-1},\ \forall m\in\overline{0,M-1},\tag{33}$$

where $\lfloor a\rfloor$ is the (lower) integer part of number $a\in\mathrm{R}$, while $n\%N$ stands for the reminder of division n/N (with $n\in\mathrm{Z}$ and $N\in\mathrm{N}^{*}$).

Now, the 1D JvNT can be applied to each column of the complex-valued matrix $\mathbf{D}_{KC}$ (for which the symmetry property cannot be exploited):

$$c_{KR,KC}^{m+1}[pr-PR_{\min}+1,kr+1]=\mathbf{N}_{KR}(\operatorname{col}_{m+1}(\mathbf{D}_{KC}))[pr,kr]=\sum_{nr=0}^{NR-1}d_{KC}[nr+1,m+1]\operatorname{sinc}\big[\pi\big(\tfrac{nr}{KR}-pr\big)\big]\mathrm{e}^{-\frac{2\pi krnr}{KR}\mathrm{j}},\tag{34}$$
$$\forall m\in\overline{0,M-1},\ \forall pr\in\overline{PR_{\min},PR_{\max}},\ \forall kr\in\overline{0,KR-1}.$$

The resulting matrix, $\mathbf{C}_{KR,KC}^{m+1}$, can be stored in a 4D array in a cartesian position indicated by the column index m after applying the TDR again. More specifically, if $\mathbf{C}$ stands for the 4D array of analysis coefficients, then the matrix $\mathbf{C}_{KR,KC}^{m+1}$ is placed in $\mathbf{C}$ as follows:

$$\mathbf{C}\left[\bullet,\bullet,\left\lfloor\frac{m}{KC}\right\rfloor+1,m\%KC+1\right]=\mathbf{C}_{KR,KC}^{m+1},\ \forall m\in\overline{0,M-1}, \tag{35}$$

where "•" stands for "all elements" of the corresponding dimension. Thus, the current element of the 4D block is

$$\begin{gathered}c[pr-PR_{\min}+1,kr+1,pc-PC_{\min}+1,kc+1]=c_{KR,KC}^{(pc-PC_{\min}+1)\cdot KC+kc+1}[pr-PR_{\min}+1,kr+1],\\ \forall pr\in\overline{PR_{\min},PR_{\max}},\ \forall kr\in\overline{0,KR-1},\ \forall pc\in\overline{PC_{\min},PC_{\max}},\ \forall kc\in\overline{0,KC-1}.\end{gathered} \tag{36}$$

As an alternative, packing the coefficients in a 4D array can be avoided by storing all matrices $\mathbf{C}_{KR,KC}^{m+1}$ in a 3D array as layers:

$$\mathbf{C}[\bullet,\bullet,m+1]=\mathbf{C}_{KR,KC}^{m+1},\ \forall m\in\overline{0,M-1}. \tag{37}$$

Algorithm 1. *Direct JvNT for 2D real-valued signals*

- **Inputs:**
  - The signal to analyze: $\mathbf{A}\in\mathbb{R}^{NR\times NC}$, ($NR$-by-$NC$ real-valued matrix);
  - The dictionary configuring parameters: mw sampling rate along rows KR, mw sampling rate along columns KC and accuracy threshold ε (by default: $KR=\left\lfloor\sqrt{2NR}+0.5\right\rfloor$, $KC=\left\lfloor\sqrt{2NC}+0.5\right\rfloor$ and $\varepsilon=0.01$).
- **Initialization:**
  1. Compute the mw truncation limits NR_{ε} and NC_{ε} like in [3].
  2. Determine the space-shifting bounds $PR_{\min}$, $PR_{\max}$, $PC_{\min}$ and $PC_{\max}$ like in [3], as well.
  3. Set the number of rows in coefficient matrix of first JvNT: $PC=PC_{\max}-PC_{\min}+1$.
  4. Set the long vector length: $M=PC\cdot KC$.
- **Main loop:**
  1. **For** $nr=1:NR$
     - 1.1. Run **Algorithm 1** from [3] for $\{\mathrm{row}_{nr}(\mathbf{A}),\mathbb{R},KC,\varepsilon\}$ to produce the coefficient matrix $\mathbf{C}_{KC}^{nr}\in\mathbb{C}^{PC\times KC}$. (The symmetry property is considered, in order to reduce the computational burden.)
     - 1.2. Vectorize the matrix $\mathbf{C}_{KC}^{nr}$ along rows by means of TDR:
     $$\mathbf{d}_{KC}^{nr}=\left[c_{KC}^{nr}\left[\left\lfloor\frac{m-1}{KC}\right\rfloor+1,(m-1)\%KC+1\right]\right]_{m\in\overline{1,M}}\ \text{(row vector)}.$$
     - 1.3. Stack the vector $\mathbf{d}_{KC}^{nr}$ in the large matrix $\mathbf{D}_{KC}$ from top to bottom: $\mathbf{D}_{KC}[nr,\bullet]=\mathbf{d}_{KC}^{nr}$.

     End
  2. **For** $m=1:M$
     - 2.1. Run Algorithm 1 again from [3] for $\{\mathrm{col}_{m}(\mathbf{D}_{KC}),\mathbb{C},KR,\varepsilon\}$ to produce the coefficient matrix $\mathbf{C}_{KR,KC}^{m}\in\mathbb{C}^{PR\times KR}$ (where $PR=PR_{\max}-PR_{\min}+1$). (No symmetry is considered here.)
     - 2.2. Store the matrix $\mathbf{C}_{KR,KC}^{m}$ in the 4D array $\mathbf{C}$ with the help of TDR:
     $$\mathbf{C}\left[\bullet,\bullet,\left\lfloor\frac{m-1}{KC}\right\rfloor+1,(m-1)\%KC+1\right]=\mathbf{C}_{KR,KC}^{m}$$
     or, alternatively, store the matrix $\mathbf{C}_{KR,KC}^{m}$ in the 3D array $\mathbf{C}$ as the current layer: $\mathbf{C}[\bullet,\bullet,m]=\mathbf{C}_{KR,KC}^{m}$.

     End
- **Outputs:**
  - The 4D or 3D array of JvN analysis coefficients: $\mathbf{C}$ (complex-valued);
  - The space-shifting bounds of dictionaries: $PR_{\min}$, $PR_{\max}$, $PC_{\min}$ and $PC_{\max}$;
  - The signal sizes: NR (number of rows) and NC (number of columns).

3.2. Inverse JvNT Algorithm for Real-Valued 2D Signals

For the synthesis procedure, it is suitable to invoke Algorithm 2 from [3], such that the symmetry property can fully be exploited. Therefore, the previous matrix-oriented approach is useful in the design of this procedure, as well.

Algorithm 2. *Inverse JvNT for real-valued 2D signals*

➢ **Inputs:**
- The signal sizes: NR (number of rows) and NC (number of columns);
- The 4D or 3D array of JvN analysis coefficients: $\mathbf{C}$ (complex-valued);
- The space-shifting bounds of dictionaries: $PR_{\min}$, $PR_{\max}$, $PC_{\min}$ and $PC_{\max}$.

Initialization:
1. Set $PC = PC_{\max} - PC_{\min} + 1$ and $PR = PR_{\max} - PR_{\min} + 1$.
2. Set the sampling rates KR, KC and the size parameter M:
   2.1. If $\mathbf{C}$ is a 4D array, then set KR as the size along the second dimension and KC as the size along the fourth dimension. Set $M = PC \cdot KC$.
   2.2. If $\mathbf{C}$ is a 3D array, then KR is the size along the second dimension, while M is the size along the third dimension. Set $KC = M / PC$.

Main loop:
1. **For** $m = 1 : M$
   1.1. Extract the matrix $\mathbf{C}_{KR,KC}^{m} \in \mathbb{C}^{PR \times KR}$ from the array $\mathbf{C}$:
      1.1.1. If $\mathbf{C}$ is a 4D array, then: $\mathbf{C}_{KR,KC}^{m} = \mathbf{C}\left[\bullet,\bullet,\left\lfloor \frac{m-1}{KC} \right\rfloor + 1, (m-1)\%KC + 1\right]$.
      1.1.2. If $\mathbf{C}$ is a 3D array, then: $\mathbf{C}_{KR,KC}^{m} = \mathbf{C}[\bullet,\bullet,m]$.
   1.2. Run Algorithm 2 from [3] for $\left\{NR, \mathbf{C}_{KR,KC}^{m}, PR_{\min}, PR_{\max}, \mathbb{C}\right\}$ to produce the auxiliary column vector $\mathbf{d}_{KR}^{m}$ of length NR. (No symmetry is considered here.)
   1.3. Store the column vector $\mathbf{d}_{KR}^{m}$ in the matrix $\mathbf{D}_{KC}$ from left to right: $\mathbf{D}_{KC}[\bullet, m] = \mathbf{d}_{KR}^{m}$.

   End
2. **For** $nr = 1 : NR$
   2.1. Reshape $\mathbf{d}_{KC}^{nr} = \text{row}_{nr}\left(\mathbf{D}_{KC}\right)$ into matrix $\mathbf{C}_{KC}^{nr} \in \mathbb{C}^{PC \times KC}$ by means of TDR:
   $$c_{KC}^{nr}[p,k] = d_{KC}^{nr}[(p-1)K + k], \ \forall p \in \overline{1,PC}, \ \forall k \in \overline{1,KC}.$$
   2.2. Run Algorithm 2 from [3] for $\left\{NC, \mathbf{C}_{KC}^{nr}, PC_{\min}, PC_{\max}, \mathbb{R}\right\}$ to produce the auxiliary column vector $\tilde{\mathbf{a}}_{nr}$ of length NC. (Symmetry is considered to reduce the computational burden.)
   2.3. Transpose the vector $\tilde{\mathbf{a}}_{nr}$ and stack it in the matrix $\tilde{\mathbf{A}}$ from top to bottom: $\tilde{\mathbf{A}}[nr,\bullet] = \tilde{\mathbf{a}}_{nr}^{T}$.

   End

➢ **Output:** the synthesized signal: $\tilde{\mathbf{A}}$ (NR-by-NC matrix).

4. Simulation Results and Discussion

After implementing the algorithms from Section 3 within the MATLAB™ programming environment, several tests were performed on bw and colored images. All of them are represented by real-valued matrices (bw images) or 3D arrays (colored images). The simulation results for a couple of 2D signals are exhibited and discussed in this section. The JvN dictionaries were configured with an accuracy threshold of $\varepsilon = 0.01$.

Two types of tests were performed: a reconstruction test and a redundancy reduction test. The first one targets the image reconstruction accuracy (i.e., the transform invertibility accuracy), depending on the JvNT parameters and the dictionary size. The second one was preferred (among other tests) because there is a direct link between the orthogonality property and the capacity of an orthogonal transform to reduce the redundancy of a signal. Usually, the redundancy is assessed by means of the compression ratio/factor, obtained after selecting the most important transform coefficients, to represent the signal (instead of signals samples). For the 2D JvNT (like in the case of 1D JvNT in [3]), only the theoretical

compression capacity was estimated, as the whole compression algorithm depends on various other techniques to be applied on JvNT coefficients afterwards. Thus, the higher the theoretical compression capacity, the stronger the redundancy reduction.

4.1. Black and White Image

The image on top in Figure 2 is a picture of Daisy—a fashionable young singer in the sixties (the photo was downloaded from a public site [31]). In the MATLAB™ programming environment, the photo can be uploaded and displayed as a bw image—see the picture on the left side, on the bottom in Figure 2. Digital bw images like Daisy are represented in gray scale, which means the corresponding matrix includes pixels with integer values varying from 0 (pure black) to 255 (pure white). Each pixel takes one unsigned byte (8 bites) of binary representation. Thus, the visual effect of gray tones is achieved.

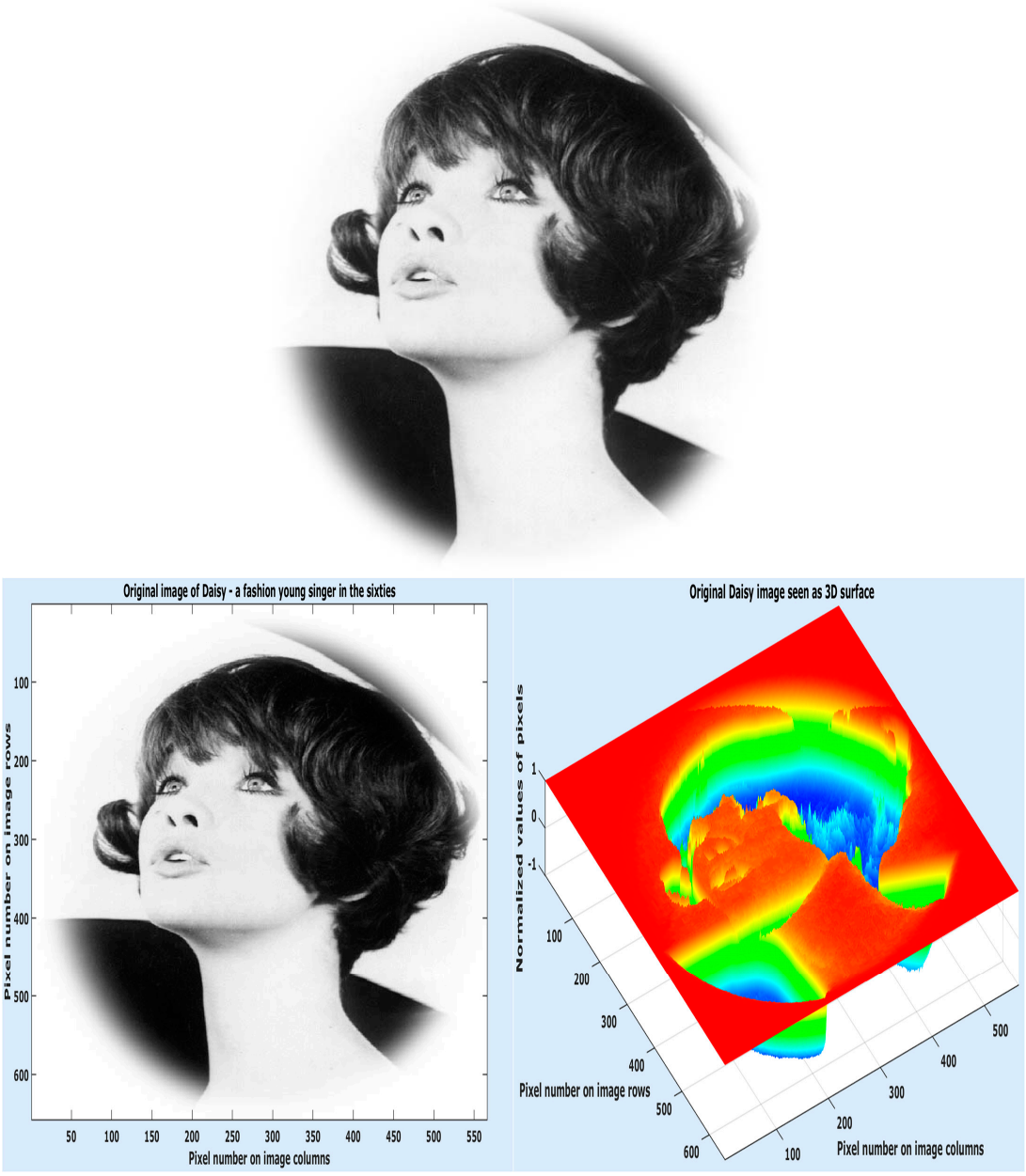

Figure 2. Images of Daisy—a fashionable young singer in the sixties. On (**top**): original photo uploaded from [31]. At bottom: original image displayed by MATLAB™ (**left side**) and original image displayed as 3D surface, after pixel normalization (**right side**).

Applying JvNT directly on bw image is unwise, as the resulting coefficients are not integers anymore and large errors can be obtained when trying to synthesize the image from them. (In general, for 2D signals with integer values, integer-to-integer transforms are more suitable to employ—see, for example [32,33].) Therefore, the integer pixels were normalized in the range of $[-1, +1]$, with *floating point* (fp) values. The resulting matrix can be represented as a 3D surface, like that on the right side on the bottom in Figure 2. The image sizes are $NR = 658$ and $NC = 566$ (with a total of 372,428 pixels).

The 2D FT applied to the normalized image led to the spectrum and phase of Figure 3.

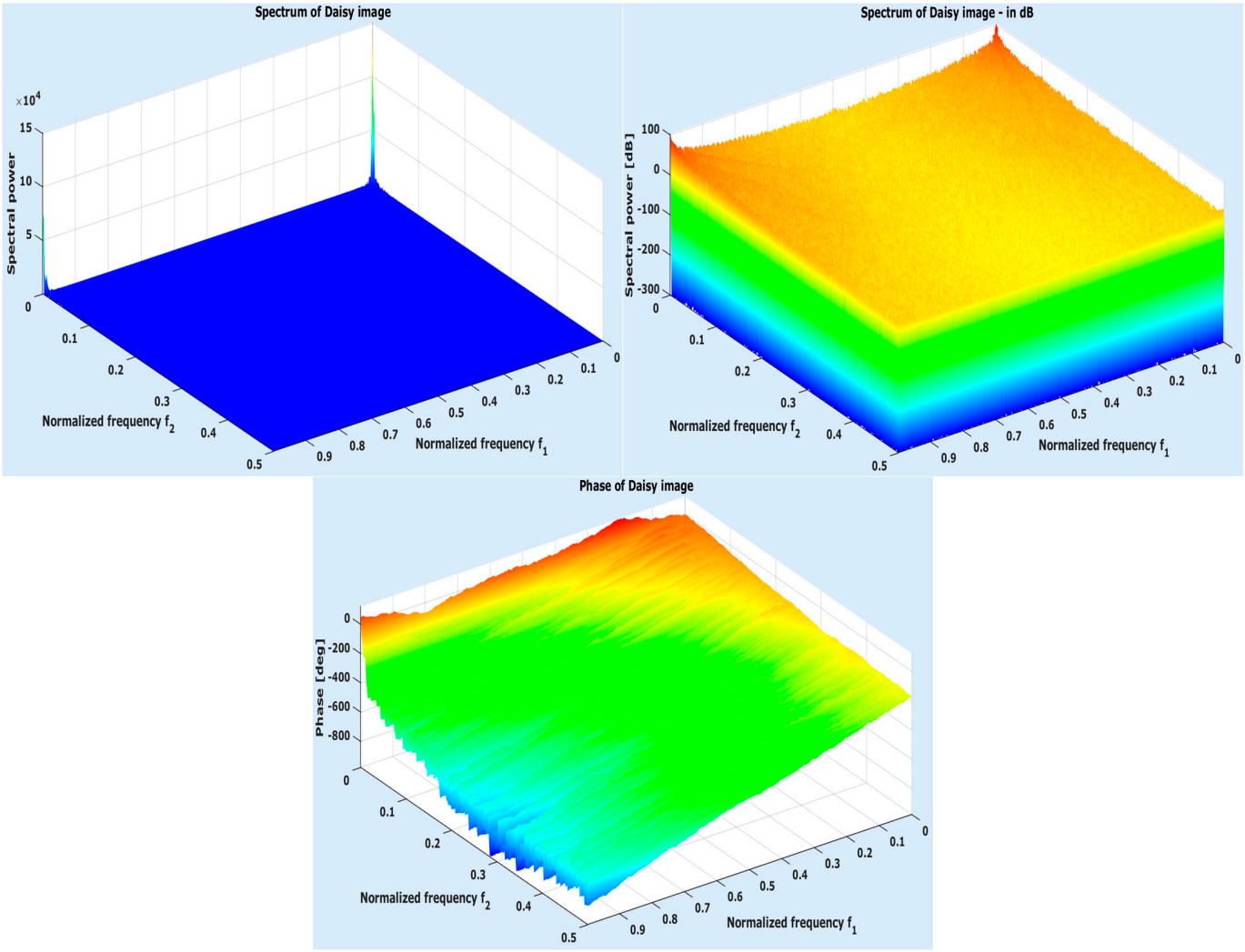

Figure 3. Frequency representation of Daisy image. On (**top**): spectrum on a linear scale (**left side**); spectrum in dB (**right side**). At (**bottom**): phase in deg.

The information of the Daisy image is concentrated in two corners, according to the first window of the figure (on top), where the spectrum on a linear scale is depicted. To the right, the spectrum in dB can be seen. The dominant frequencies are not so obvious under this surface. The phase surface is depicted at the bottom of the figure and reveals almost linear variations along the first frequency axis.

JvN analysis can be performed through Algorithm 1 in two cases:

- for $[KR|KC] = \left\lfloor \sqrt{[NR|NC]} + 0.5 \right\rfloor = [26|24]$;
- for $[KR|KC] = \left\lfloor \sqrt{2[NR|NC]} + 0.5 \right\rfloor = [36|34]$.

The resulting coefficients cannot be displayed because they belong to 3D or 4D arrays. Note that the total number of coefficients is 4,831,632 for $[KR|KC] = [26, 24]$ and 8,029,440 for $[KR|KC] = [36|34]$. Both numbers are very large.

To estimate the theoretical compression capacity of JvNT, the approach is similar to the one in [3]. Thus, the linearized arrays of JvN coefficients were first sorted in descending order of their magnitude. Assume that the total amount of energy accumulated by all coefficients is **E** (the sum of squared magnitudes) and set a threshold $\eta \in [0, 1]$ of relative energy. Then, the first coefficients that accumulate an amount of energy at least equal to η**E** are counted. Denote their number as $N_\eta \in \mathrm{N}$. In Figure 4, the variations in N_η (depending on the threshold η) are drawn in both cases. From the figures, one can notice that, if $\eta = 0.95$, then $N_{0.95} = 689$ or $N_{0.95} = 571$, which represents no more than $[0.014|0.007]\%$ of the total coefficient number, respectively. When comparing to the total number of pixels in the image (i.e., 372,428), both numbers $N_{0.95}$ are very small.

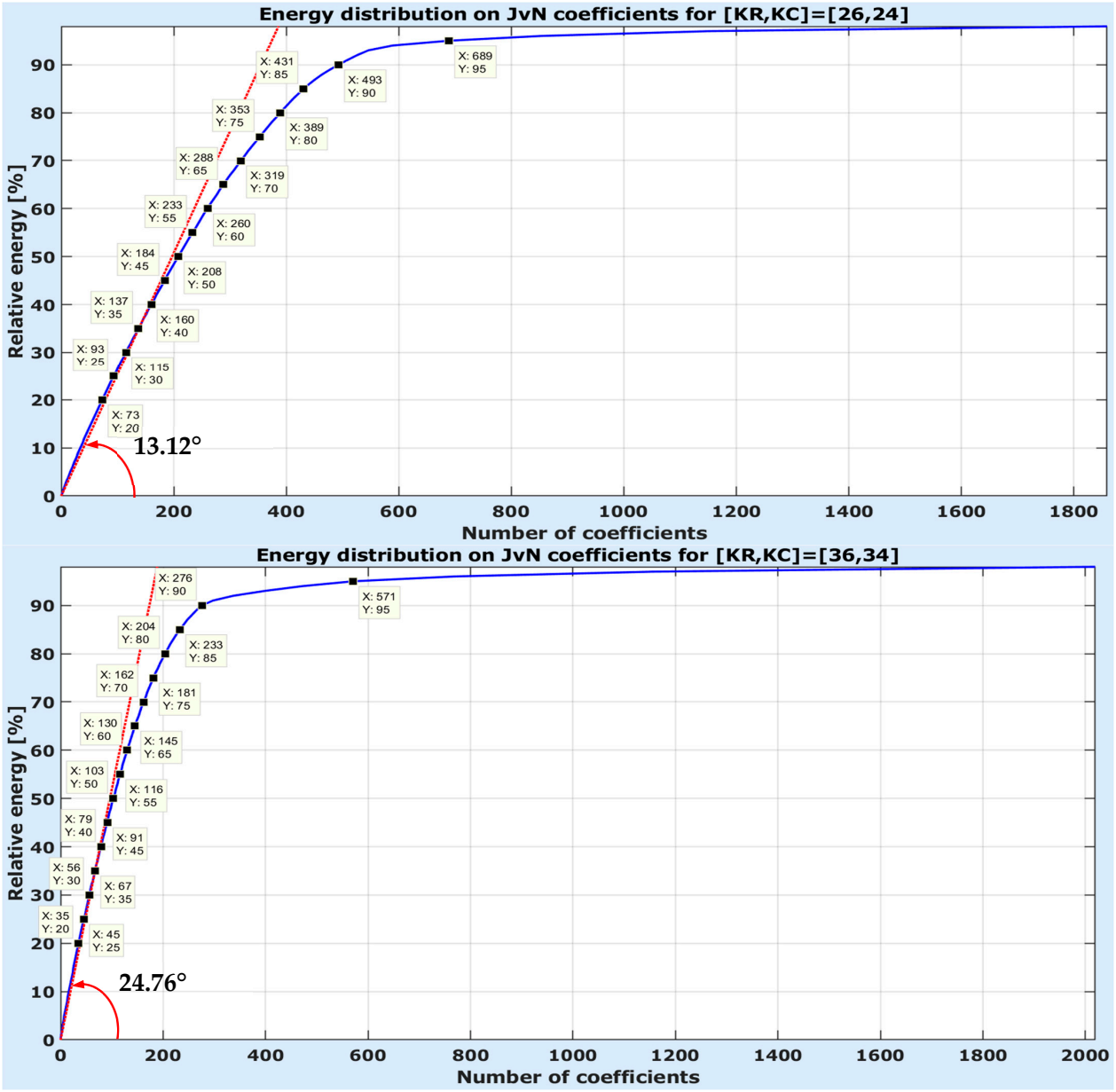

Figure 4. Theoretical compression capacity of JvNT applied to Daisy image for $[KR|KC] = [26|24]$ on (**top**) and $[KR|KC] = [36|34]$ at (**bottom**).

Nevertheless, to be fair, it should be outlined that, based on the difference from the binary representation of pixels, the JvN coefficients are complex-valued and represented in fp double precision, which means that $N_{0.95}$ must be multiplied first by 2, and then, by the number of bytes allocated to double precision, e.g., 8, on a 64-bit computer. In this case, the number of selected bytes is $[11,024|9136]$, which means approximately $[2.96|2.45]\%$ of the total number of bytes in the image. If the symmetry properties are considered, the number of bytes can be almost halved. Thus, $[1.62|1.35]\%$ of the total number of bytes in the image are sufficient to recover the Daisy picture with fair accuracy.

The two depicted angles standing for the theoretical compression capacity were computed as explained in [3]. The interval $[0,1]$ of η was discretized with a step of 0.01, which resulted in 101 energy thresholds (including the null one). For each couple $\{\eta, N_\eta\}$, the angle of the first derivative in the origin is estimated as follows:

$$\alpha_\eta = \arctan\frac{100\eta}{N_\eta}. \tag{38}$$

Then, the final angle is obtained by averaging all 100 angles (38) (except for the first one). The higher the average angle, the better the theoretical compression capacity. Figure 4 reveals that the second JvNT, for $[KR|KC] = [36|34]$, is superior to the first one, for $[KR|KC] = [26|24]$, in terms of theoretical compression capacity, since the corresponding average angle is 24.76°, compared to 13.12°.

In Table 1, the variations in Figure 4 are sampled.

Table 1. Relative energy $\eta[\%]$ versus JvN coefficient number N_η for Daisy image.

$[KR\,\vert\,KC]\downarrow$	$\eta\rightarrow$	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**	**65**	**70**	**75**	**80**	**85**	**90**	**95**	**Angle**
[26|24]	$N_\eta \rightarrow$	73	93	115	137	160	184	208	233	260	288	319	353	389	431	473	689	13.12°
[36|34]	$N_\eta \rightarrow$	35	45	56	67	79	91	103	116	130	145	162	181	204	233	276	571	24.76°

After the completion of the synthesis stage (through Algorithm 2), the obtained results are depicted in Figure 5 (for $[KR|KC] = [26|24]$) and Figure 6 (for $[KR|KC] = [36|34]$).

In both figures, the first synthesized image is obtained in fp, which means the pixels' normalized values must vary within a $[-1, +1]$ interval. The outliers are saturated, as they are produced by numerical errors. In the upper left corner, one can see the recovered 3D images. Apparently, both images are identical to the original one in Figure 2.

Nevertheless, the recovery errors are non-null, as the windows located in the left corner at bottom of each figure clearly reveals. Although not obvious, the first error (in Figure 5) is slighter smaller, as proven by the performance parameters below. The same behavior is proven by the next column in both figures. This time, the recovered images and errors work with binary (integer) pixels. To obtain the bw image, it suffices to map back the interval $[-1, +1]$ to the integers set at $\overline{0.255}$ by using rounding.

The white and gray spots in the binary error images of figures (see the right-side corners at the bottom) point to the affected pixels (recall that black means a null error). Nevertheless, the recovered bw images (in the upper right corner) are visually identical to the original image on top in Figure 2.

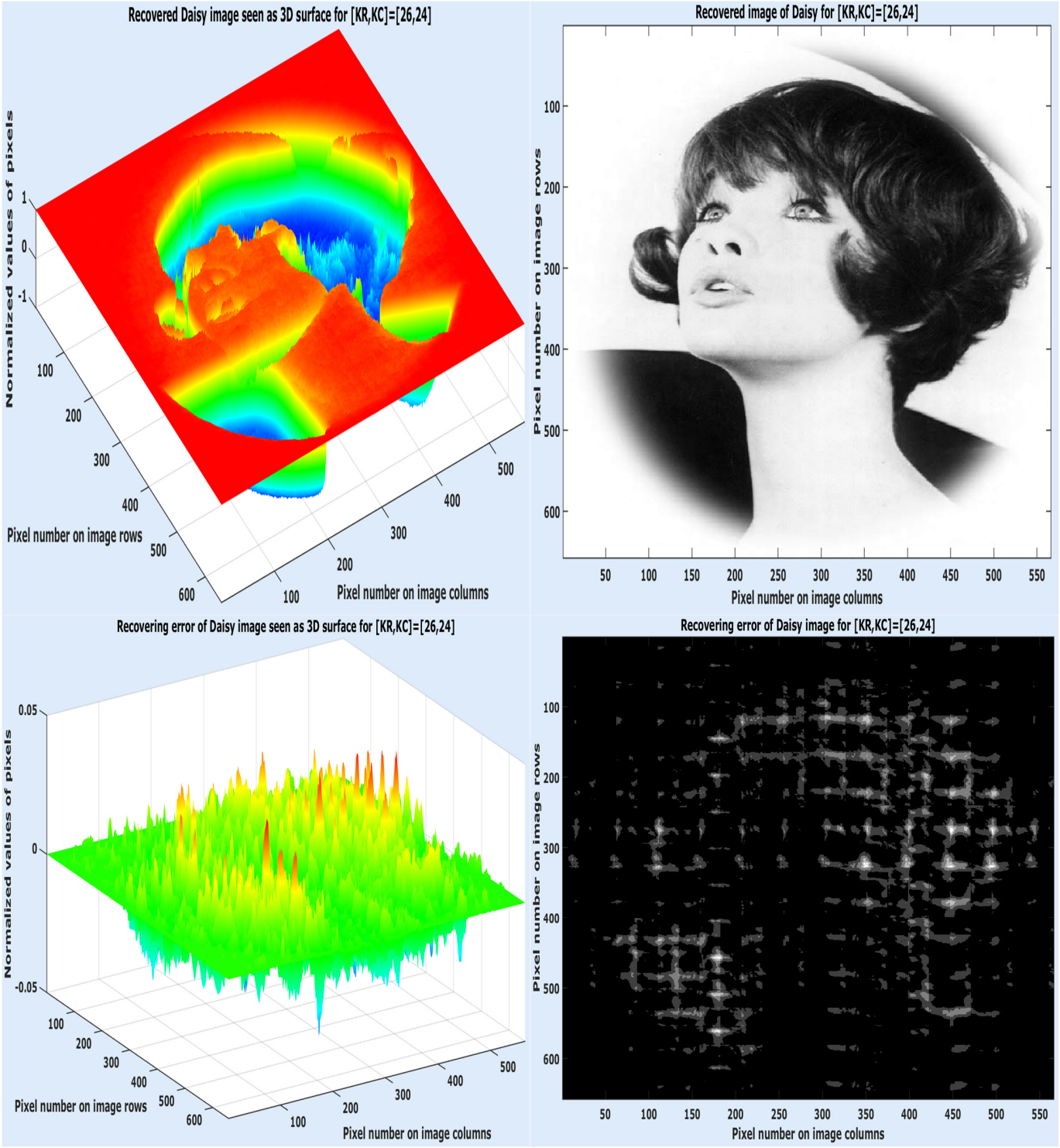

Figure 5. Results of JvN synthesis for Daisy image, in cases $[KR|KC] = [26|24]$. On (**top**): recovered 3D image (fp) (**left side**) and bw image (integer/binary) (**right side**). At (**bottom**): recovery error of 3D image (fp) (**left side**) and bw image (integer/binary) (**right side**).

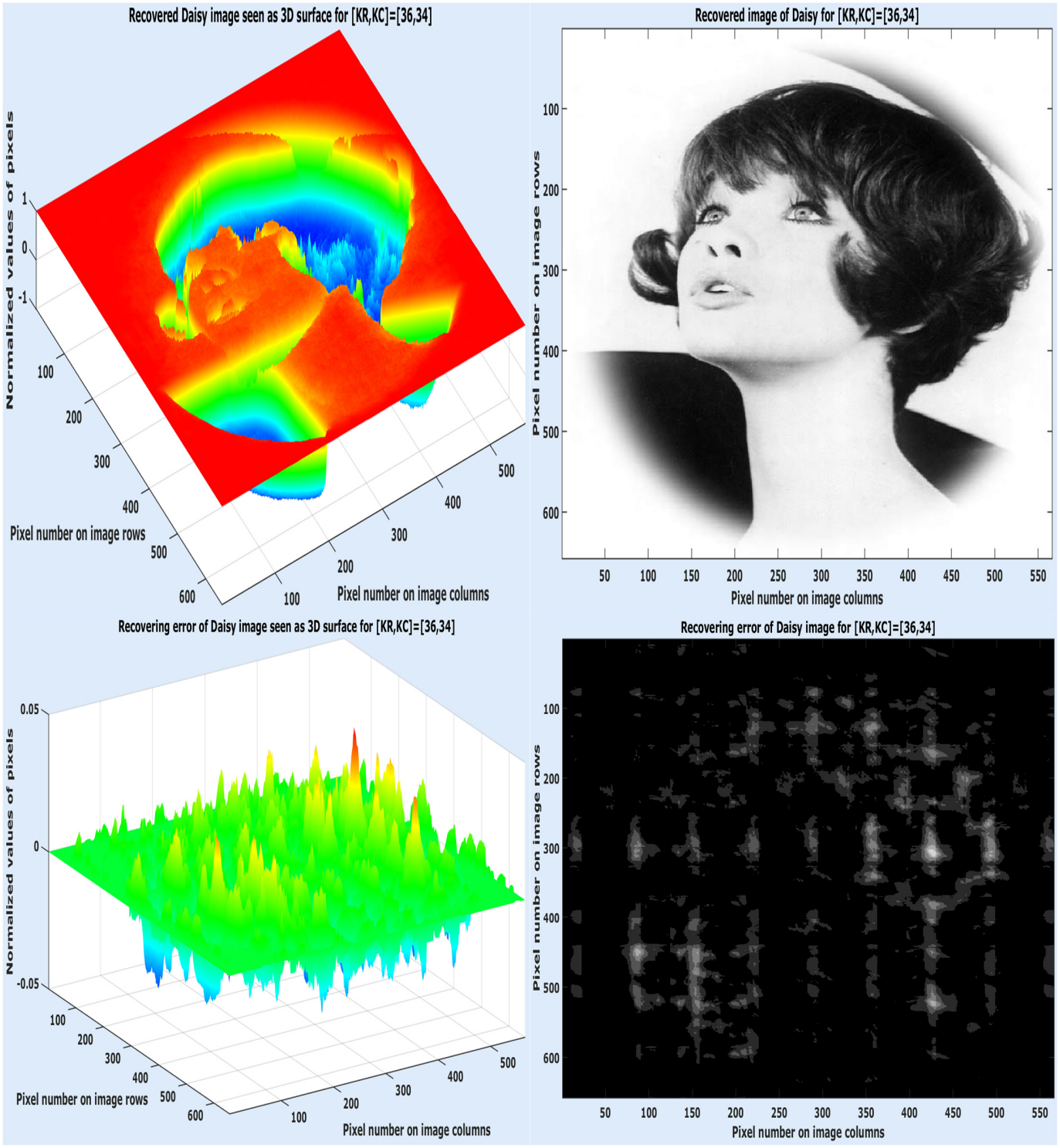

Figure 6. Results of JvN synthesis for Daisy image in cases $[KR|KC] = [36|34]$. Same window disposal as in Figure 5.

The performance parameters of 2D JvNT are listed in Table 2. The relative *standard deviation* (std) of the error is computed as follows:

$$\sigma_{\tilde{\mathbf{A}}} = 100 \frac{\left\| \mathbf{A} - \tilde{\mathbf{A}} \right\|_2}{\|\mathbf{A}\|_2} \ [\%], \tag{39}$$

where $\mathbf{A}$ is the genuine matrix before analysis, and $\widetilde{\mathbf{A}}$ is the synthesized matrix. The fitness (which actually evaluates the relative recovery accuracy) is computed by using definitions (19) and (28). More specifically,

$$\mathbf{A} = \frac{100}{1 + \frac{\sigma_{\widetilde{\mathbf{A}}}}{10}} \ [\%]. \tag{40}$$

Table 2. Performance parameters of JvNT in case of Daisy image.

Image Size $NR \times NC$	Sampling Rates $[KR \mid KC]$	Relative Std of Error [%]			Fitness (Accuracy) [%]			Analysis Runtime [s]	Synthesis Runtime [s]
		fp	Saturated fp	Integer	fp	Saturated fp	Integer		
658×566	$[26\vert 24]$	0.7744	0.643	0.1624	92.81	93.96	97.75	282.74	39.79
	$[36\vert 34]$	0.9574	0.8642	0.2065	91.26	92.05	97.01	455.18	80.84

As one can notice, the recovery accuracy improves while passing from fp representations of pixels to binary representation. The first JvNT is slightly better than the second one in terms of recovering accuracy. Nevertheless, the second JvNT seemingly is better than the first one in terms of compression capacity.

The synthesis runtime is sensibly smaller than the analysis runtime: seven times for $[KR|KC] = [26|24]$ and six times for $[KR|KC] = [36|34]$. The runtime can rapidly increase without using the symmetry properties. Also, in all implementations, the JvN coefficients were recorded in 3D arrays, which also speed up the run (compared to the 4D array solution).

Figure 7 displays the results of lossy synthesis for the Daisy image. In the window on top in the figure, fitness characteristics are drawn for the two JvNTs, depending on the *number of strongest coefficients* (nsc), N_η. If the nsc increases, fitness increases, as expected. Although in the beginning (for a small nsc), the second JvNT (for higher sampling rates) performs better in terms of fitness (see the drawing in red), when more and more coefficients are added, the performance of both transforms becomes approximately the same (the first JvNT tacks the lead with very small advancement).

Regarding the fitness characteristics, two points were selected for each couple of sampling rates. Thus, the synthesized images on the left side at the bottom of Figure 7 were obtained for $\eta = 0.95$, i.e., from the strongest $[689|571]$ coefficients (see Table 1 again). Both images are visibly distorted (blurred) because the number of selected coefficients is too small (despite them storing approximately 95% of image energy). This is confirmed either by the recovery errors with a relative std of $[4.61|4.51]\%$ or by the fitness values of $[68.45|68.93]\%$. When closely looking at the images, one can see that the second JvNT (at the bottom) offered slightly better accuracy (and slightly smaller error to the right) than the first JvNT (on top) (in fact, the resulting fitness is about 0.5% higher).

If we move now to the right on fitness characteristics, up to the points determined by $\eta = 0.99$, this time, the nscs are $[4328|4815]$. In the 64-bit representation, they take $[138,496\,|\,154,080]$ bytes, i.e., $[37.19\,|41.37]\%$ of the image's total number of bytes (372,428). With the symmetry property, these percentages are almost halved: $[19.5\,|21.8]\%$ of the image's total number of bytes. This yields sensibly, reducing the blurring effect in both synthesized images on the right side at the bottom of the figure. The recovery errors are smaller, as well, with a relative std of $[1.61|1.59]\%$. Consequently, fitness increases to $[86.1|86.27]\%$. In this case, too, the second JvNT performs slightly better in terms of fitness, but at the expense of a higher nsc employed in the synthesis.

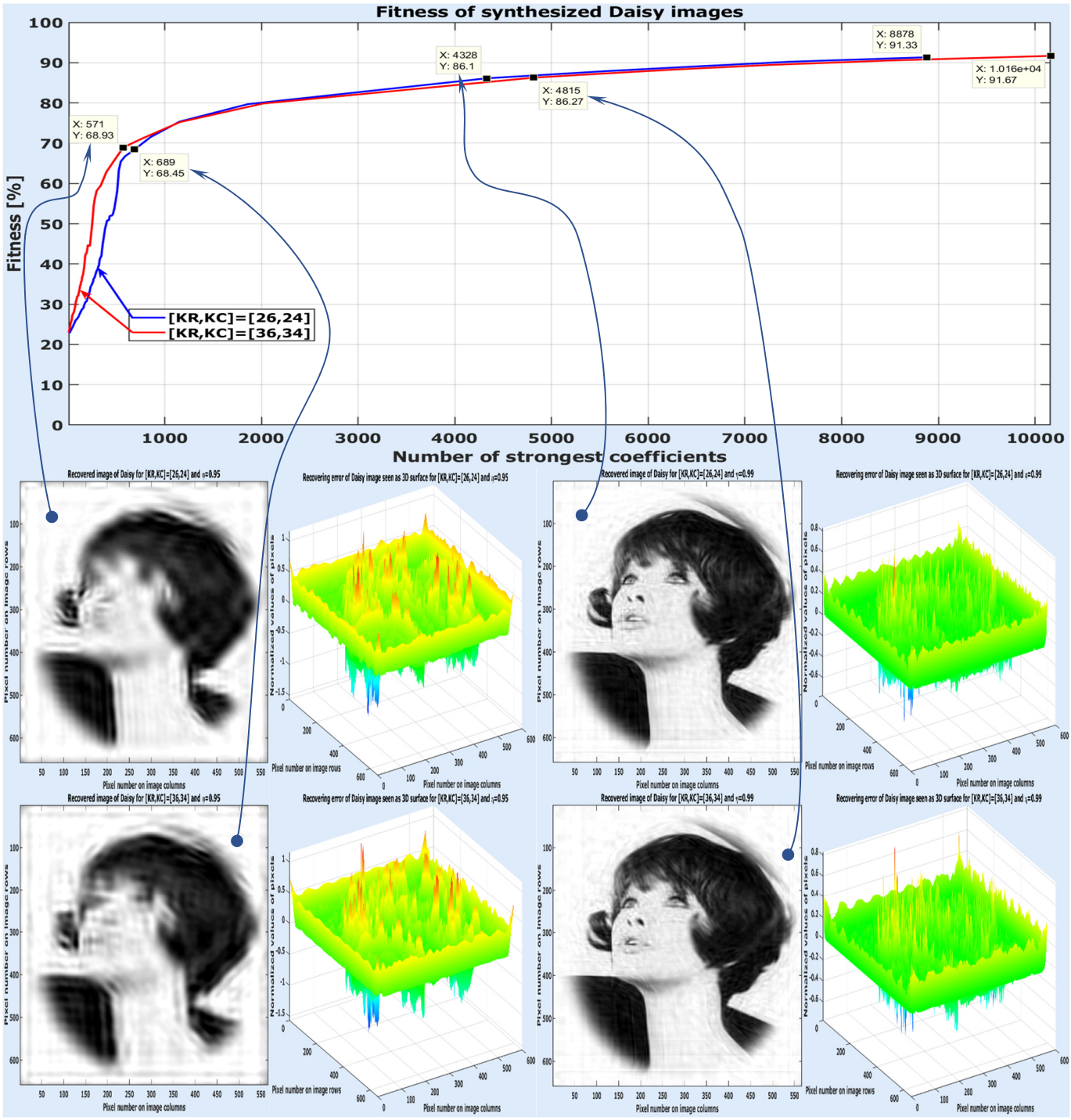

Figure 7. Results of lossy synthesis of Daisy image. Fitness variations in the middle. Synthesized images and recovery errors for $\eta = 0.95$ on the left side and for $\eta = 0.99$ on the right side. On both sides: for $[KR|KC] = [26|24]$ at the top and for $[KR|KC] = [36|34]$ at the bottom.

4.2. Colored Image

The data type required to operate with colored images using a computer is 3D array. If the image is represented in a *red–green–blue* (RGB) system (which is one of the most employed), then the array consists of three layers—one for each color component. On every layer, unsigned integers determine the proportion of each color that contributes to pixel definition, as suggested in Figure 8.

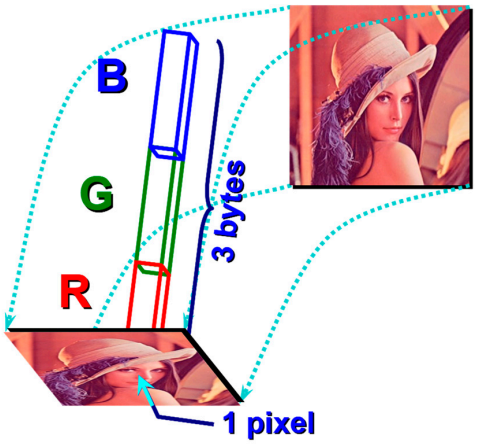

Figure 8. Digital representation of colored images in RGB system (demonstrated on a Lena photo).

So, practically, a colored image is represented by three matrices of bytes. The image under study in this article is depicted in Figure 9: a photo of the masterpiece painting *Ship of Dreams* (also known as *The Wind*) by Salvador Dali (the photo was taken from a private paintings album).

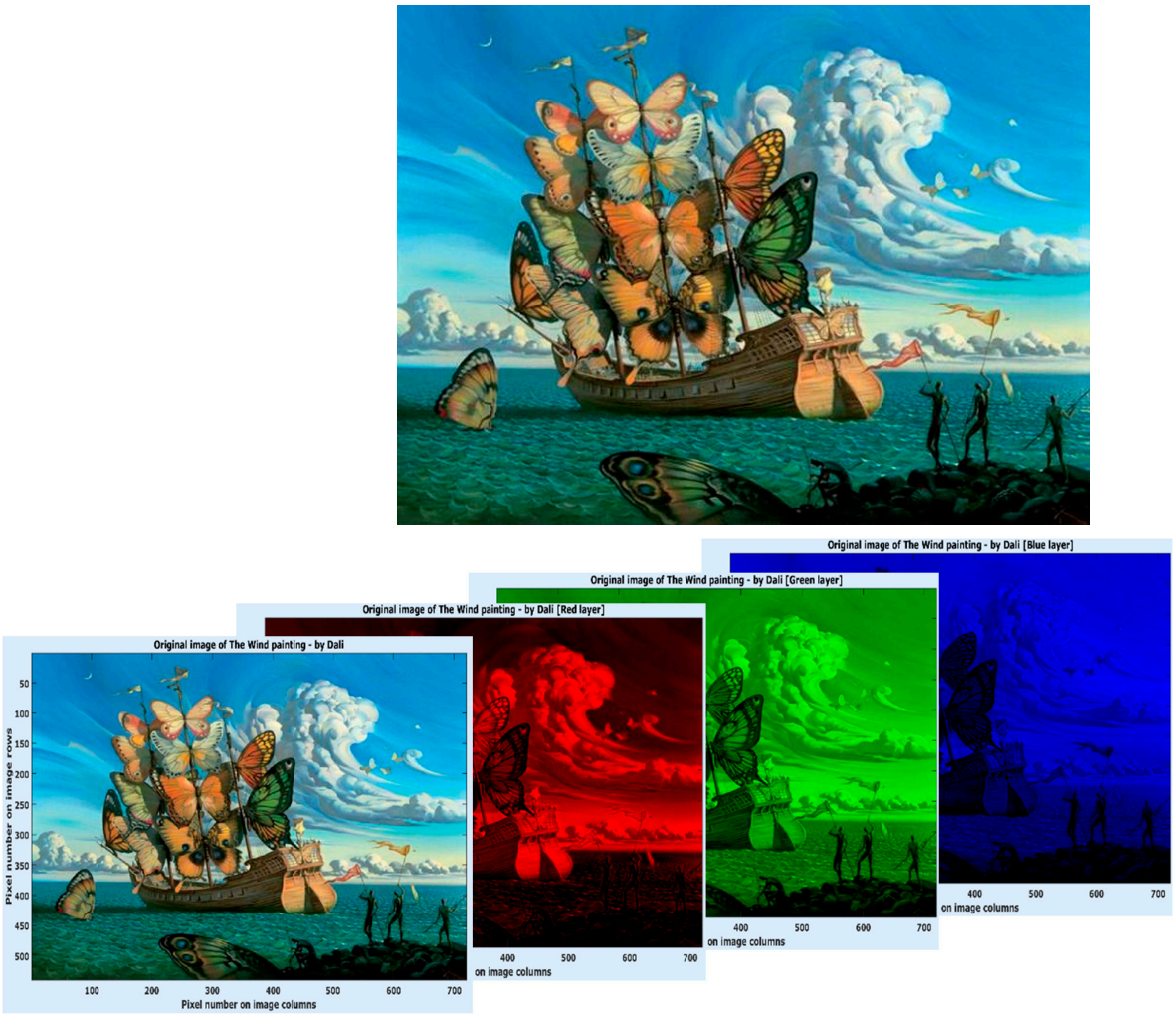

Figure 9. Images of *Ship of Dreams/The Wind* painting by Salvador Dali. On (**top**): original photo. At **bottom**: original image displayed by MATLAB™ (**left side**); towards the (**right side**): the Red, Green and Blue layers.

The image sizes are $NR = 540$ and $NC = 720$ (with a total of 388,800 pixels represented on three unsigned bytes each, i.e., 1,166,400 bytes for the whole 3D array). Apparently, beside the colormap, the layers are not so different from each other, as all three must follow the same pattern imposed by the painting itself. Nevertheless, the three matrices are different, as proven by their norms: 58,883.81 for the R layer; 83,152.92 for the G layer; 89,721.65 for the B layer. Also, the norms computed on the differences between them are quite big: 32,776.94 for R-G; 45,320.43 for R-B; 16,823.66 for G-B. Interestingly, the G and B layers are closer to each other than to the R layer. Moreover, their norms are bigger. These numerical results are visually confirmed by the whole image, where one can easily notice that the blue and green colors are dominant over the red color, which only fills small areas.

Two approaches can be considered when applying JvNT to colored images: extend definition (12) to 3D arrays (i.e., work with three instead of two dictionaries) or process every layer separately by means of 2D transform. The first approach is appropriate when the number of layers is large enough. In the case of colored images, with only three layers, the third dictionary would be devoted to processing 1D signals with a length of three, which is unnecessarily complex. Therefore, the second approach (process every layer as a 2D signal) is more suitable. This strategy is like the one employed for the bw Daisy image (first, normalize the unsigned integers of the layer; then, process the resulting matrix with the fp values; finally, come back to the unsigned integers for the recovered layer).

The 2D FT applied on each layer led to the results in Figure 10.

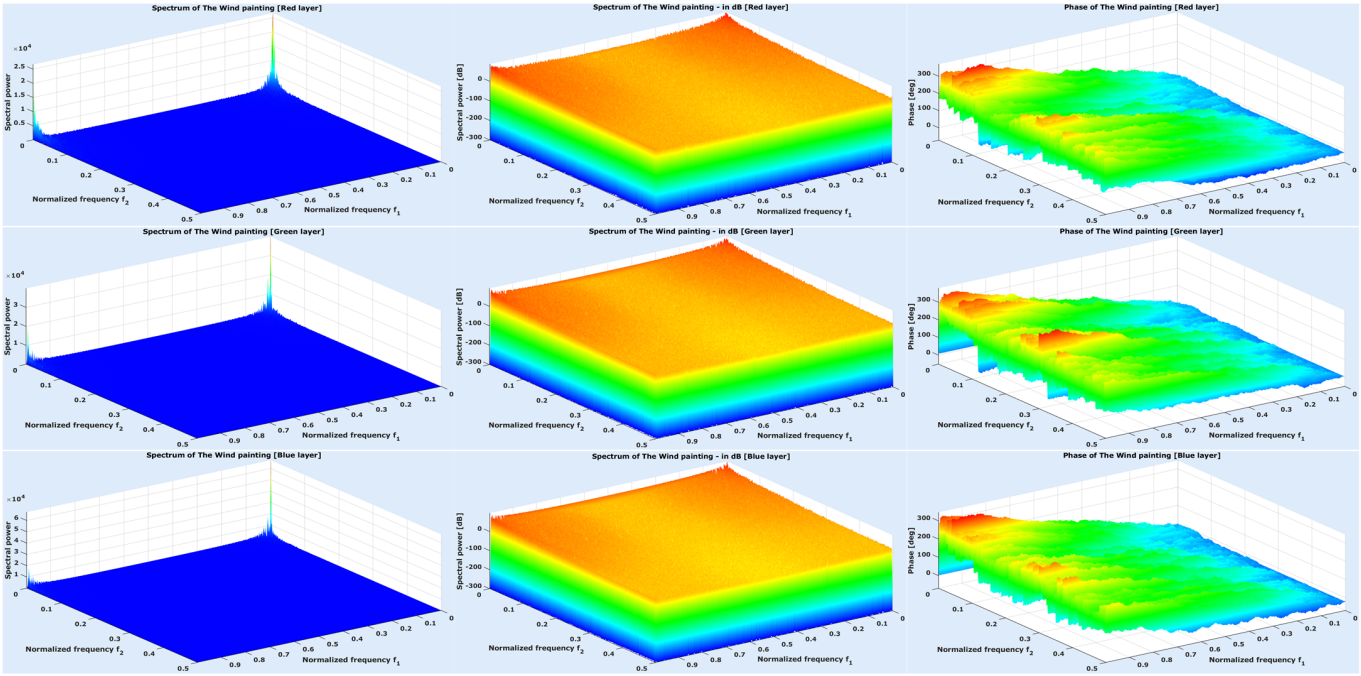

Figure 10. Spectrum in linear scale (**left side**), spectrum in dB (**middle**) and phase in deg (**right side**) of *Ship of Dreams/The Wind* painting. The RGB layers are stacked from top to bottom.

The spectral representation of all layers has the same main characteristics as for the bw image: the energy/information seems to be concentrated in the main two corners of the frequency plane, although there is energy dissipation on the remaining Fourier coefficients. Therefore, compression of the image by using FT could perform as well.

Algorithm 1 was run for each layer in the case of the sampling rates below:

- $[KR|KC] = \left\lfloor \sqrt{[NR|NC]} + 0.5 \right\rfloor = [23|27]$;
- $[KR|KC] = \left\lfloor \sqrt{2[NR|NC]} + 0.5 \right\rfloor = [33|38]$.

The total number of analysis coefficients is quite big: 4,862,430 per layer for $[KR|KC] = [23|27]$ and 8,226,240 per layer for $[KR|KC] = [33|38]$. Nevertheless, like in

case of Daisy photo, only a few of them encode essential information about the analyzed image. A similar analysis of compression capacity led to the results in Figures 11 and 12 and Table 3.

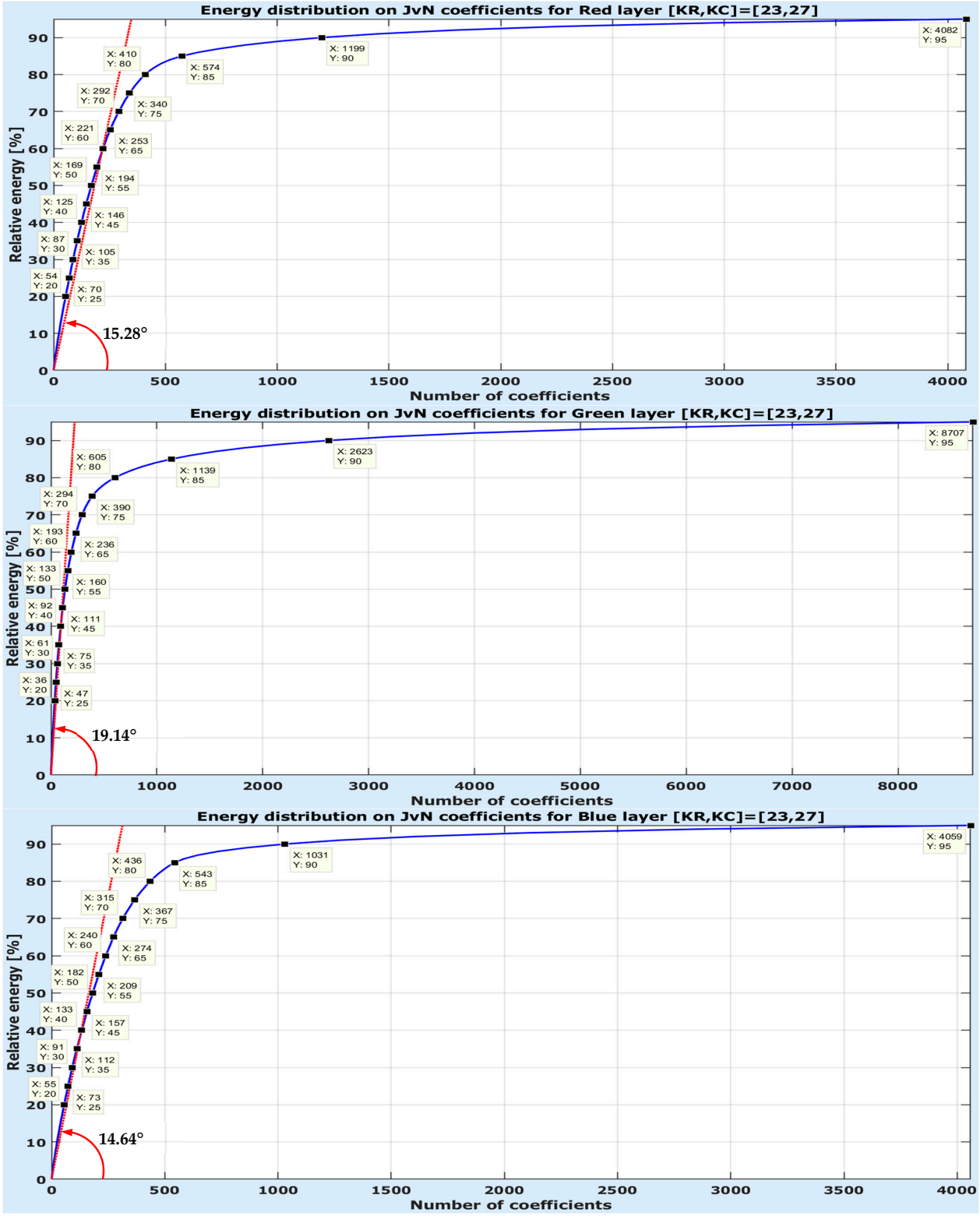

Figure 11. Theoretical compression capacity of JvNT applied to *Ship of Dreams/The Wind* painting for $[KR|KC] = [23|27]$ in layers: Red on (**top**), Green in the (**middle**) and Blue at (**bottom**).

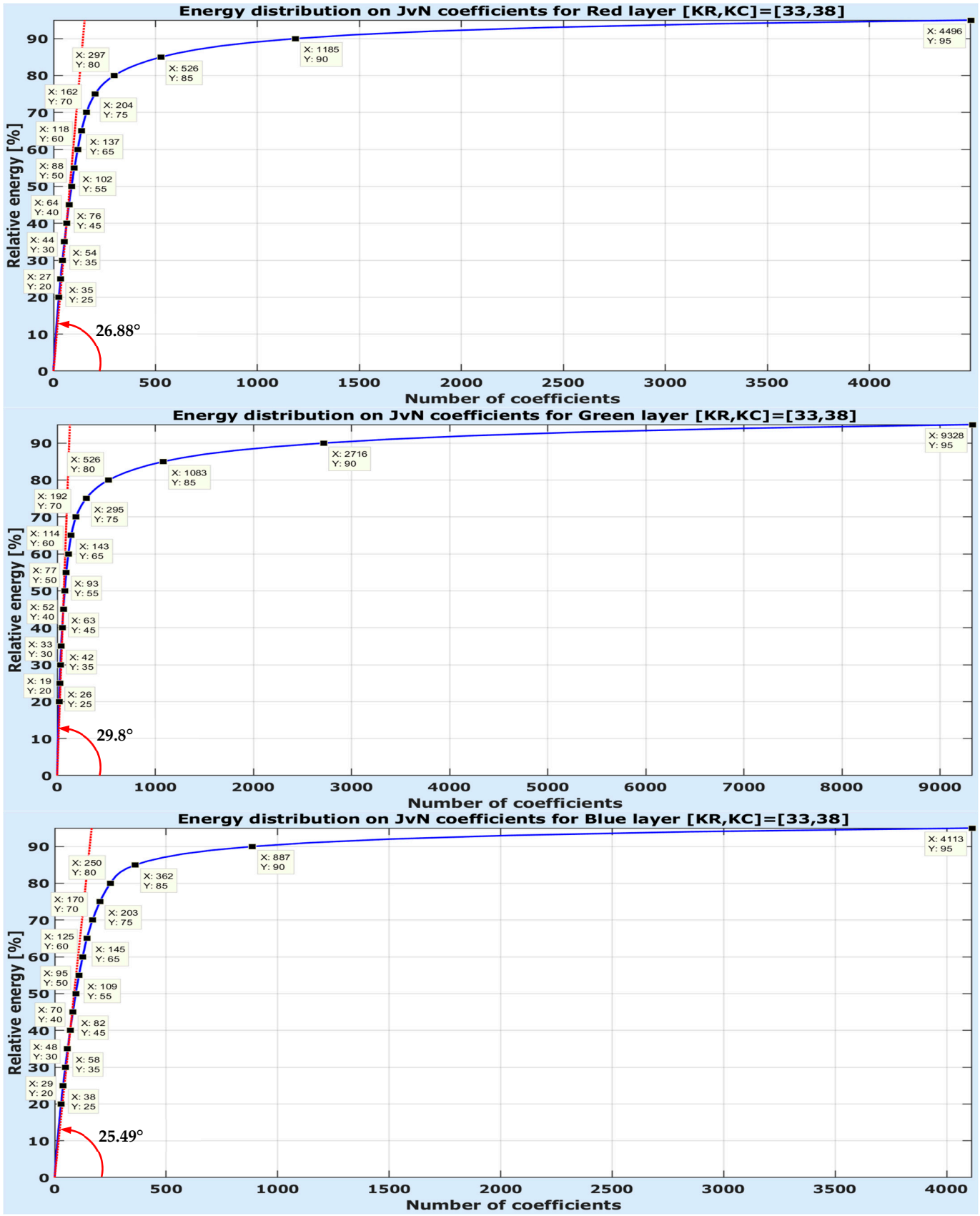

Figure 12. Theoretical compression capacity of JvNT applied to *Ship of Dreams/The Wind* painting for $[KR|KC] = [33|38]$ in layers: red on (**top**), green in the (**middle**) and blue at (**bottom**).

Table 3. Relative energy η[%] versus JvN coefficient number N_η for *Ship of Dreams/The Wind* painting layers.

$[KR\,\vert\,KC]\downarrow$	$\eta\rightarrow$	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**	**65**	**70**	**75**	**80**	**85**	**90**	**95**	**Angle**	**Layer**
[23|27]	$N_\eta \rightarrow$	54	70	87	105	125	146	169	194	221	253	292	340	410	574	1199	4082	15.28°	**R**
		36	47	61	75	92	111	133	160	193	236	294	390	605	1139	2623	8707	19.14°	**G**
		55	73	91	112	133	157	182	209	240	274	315	367	436	543	1031	4059	14.64°	**B**
[33|38]	$N_\eta \rightarrow$	27	35	44	54	64	76	88	102	118	137	162	204	297	526	1185	4496	26.88°	**R**
		19	26	33	42	52	63	77	93	114	143	192	295	526	1083	2716	9328	29.80°	**G**
		29	38	48	58	70	82	95	109	125	145	170	203	250	362	887	4113	25.49°	**B**

If $\eta = 0.95$, the total number of coefficients (on all three layers) is 16,848 for $[KR|KC] = [23|27]$ and 17,937 for $[KR|KC] = [33|38]$. This means that 95% of the image energy is concentrated in $[0.1155|0.0727]\%$ of the total coefficient number. In the case of 64-bit representation, the number of bytes that can be selected is 269,568 for $[KR|KC] = [23|27]$ and 286,992 for $[KR|KC] = [33|38]$, i.e., $[23.11|24.61]\%$ of the total number of bytes in the image (on all three layers).

If the symmetry properties are considered, the percentages are almost twice smaller: $[12.66|13.51]\%$. The average on the three layers of the compression angles depicted in Figure 11 is 16.35° (for $[KR|KC] = [23|27]$), while this average increases to 27.39° for the angles in Figure 12 (where $[KR|KC] = [33|38]$). This result shows that, similarly to the analyzed bw image, the second JvNT (for higher sampling rates) seemingly can better compress the color image than the JvNT for smaller sampling rates.

Each layer was recovered from the corresponding JvN coefficients by using Algorithm 2. The synthesis results are displayed in Figures 13 and 14 for $[KR|KC] = [23|27]$ and $[KR|KC] = [33|38]$, respectively. In both figures, the layers are shown in rows (red on top, green in the middle and blue at the bottom). The synthesized paintings are recomposed from RGB layers like in Figure 15.

When looking at all the previous pictures, one can straightforwardly conclude that the synthesized images are very close to the original ones. This subjective observation is confirmed by the performance parameters in Table 4.

Table 4. Performance parameters of JvNT in case of *Ship of Dreams/The Wind* painting.

Image Size $NR \times NC$	**Sampling Rates** $[KR\,\vert\,KC]$	**Layer**	**Relative Std of Error [%]**			**Accuracy [%]**			**Analysis Runtime [s]**	**Synthesis Runtime [s]**
			fp	**Saturated fp**	**Integer**	**fp**	**Saturated fp**	**Integer**		
540 × 720	[23|27]	R	0.8122	0.8032	0.5187	92.49	92.57	95.07	263.05	36.14
		G	1.1138	1.1121	0.3266	89.98	89.99	96.84	268.28	37.24
		B	0.9174	0.9024	0.3335	91.60	91.72	96.77	269.43	36.21
		P	**0.9478**	**0.9393**	**0.3929**	**91.36**	**91.43**	**96.23**	**800.67**	**109.59**
	[33|38]	R	0.9030	0.8953	0.5795	91.72	91.78	94.52	425.91	73.15
		G	1.0567	1.0552	0.3103	90.44	90.46	96.99	436.25	80.64
		B	0.8057	0.7900	0.2907	92.54	92.68	97.18	433.21	80.96
		P	**0.9218**	**0.9135**	**0.3935**	**91.57**	**91.64**	**96.23**	**1295.37**	**234.75**

The notation "**P**" in the table above stands for the whole picture, as composed by all three layers. On each row "**P**", the averages of the errors relative the std and fitness values are listed. Also, the runtimes are cumulated on row "**P**", as if the layers were sequentially processed. One can see that the two JvNTs are similar in terms of recovery accuracy, although the second one runs longer (however, recall that this JvNT seems to perform better in terms of compression capacity). The synthesis runtime is 6 to 8 times smaller than the analysis runtime, especially thanks to the symmetry properties.

The analysis of lossy synthesis was performed like in the case of the Daisy image. Nevertheless, for Dali's painting, each layer was processed separately. The obtained results are illustrated in Figure 16 (for $[KR|KC] = [23|27]$) and Figure 17 (for $[KR|KC] = [33|38]$).

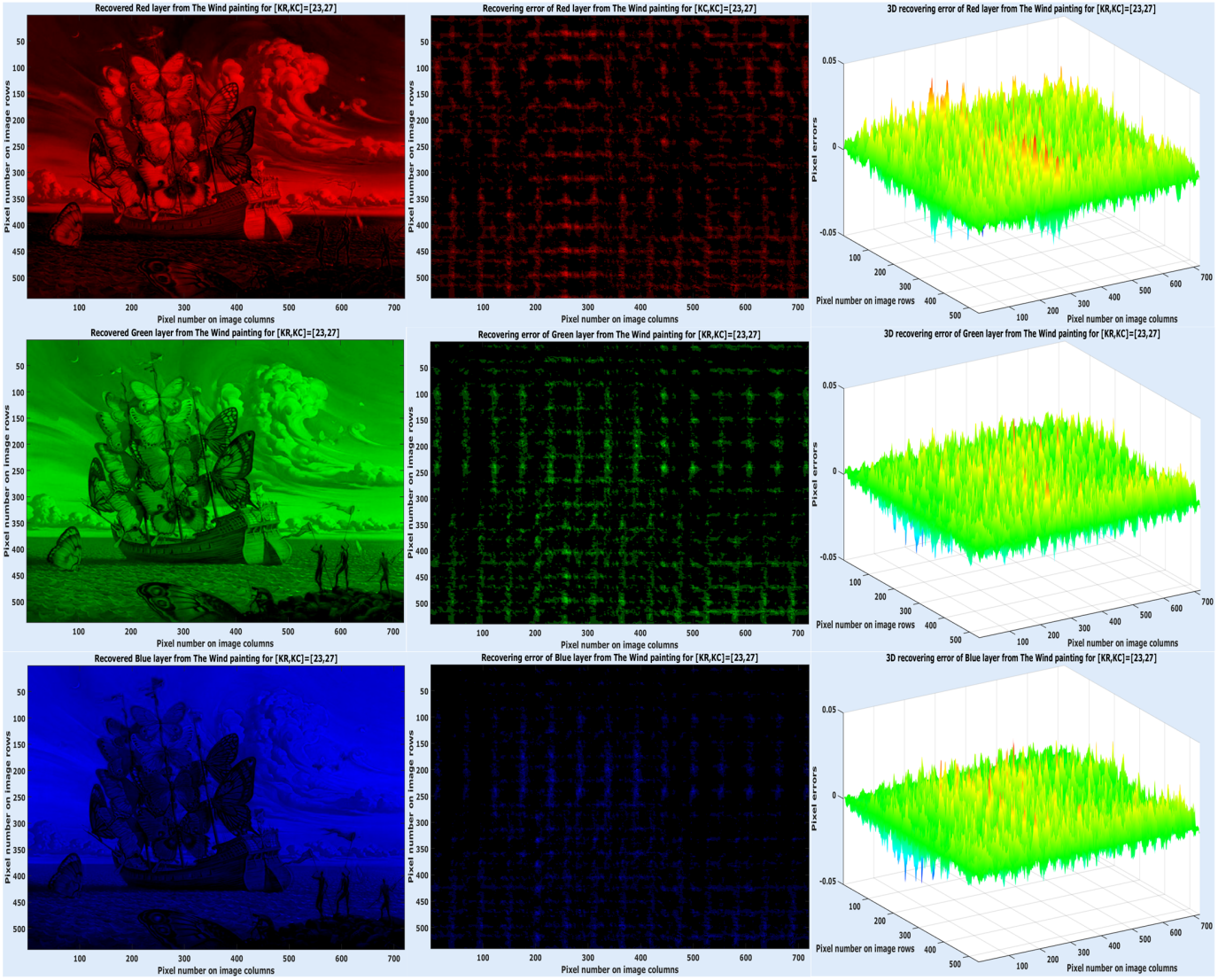

Figure 13. Results of JvN synthesis for *Ship of Dreams/The Wind* painting layers for $[KR|KC] = [23|27]$: R layer on (**top**), G layer in the (**middle**), B layer at (**bottom**). For each layer, one can see (from (**left**) to (**right**)) the recovered binary image, the recovery error as a digital image and the recovery fp error as 3D surface.

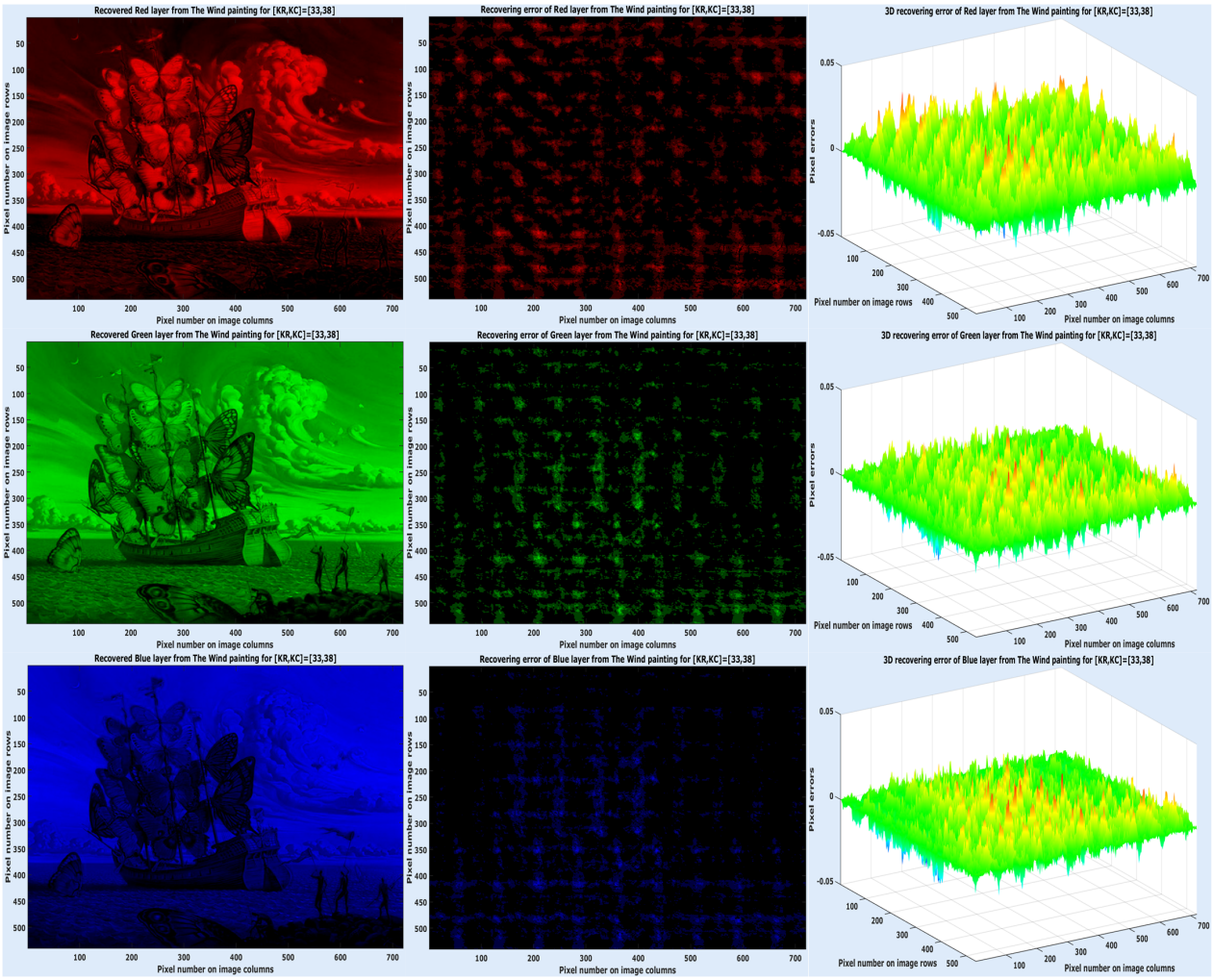

Figure 14. Results of JvN synthesis for *Ship of Dreams/The Wind* painting layers for $[KR|KC] = [33|38]$. Same window disposal as in Figure 13.

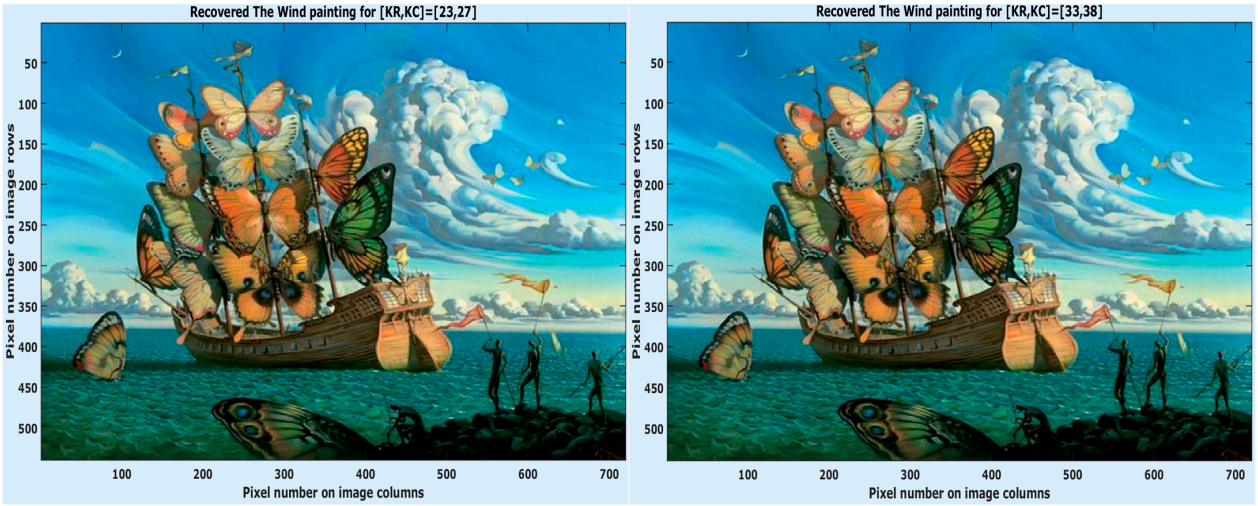

Figure 15. Synthesized images of *Ship of Dreams/The Wind* painting for $[KR|KC] = [23|27]$ on the left side and $[KR|KC] = [33|38]$ on the right side.

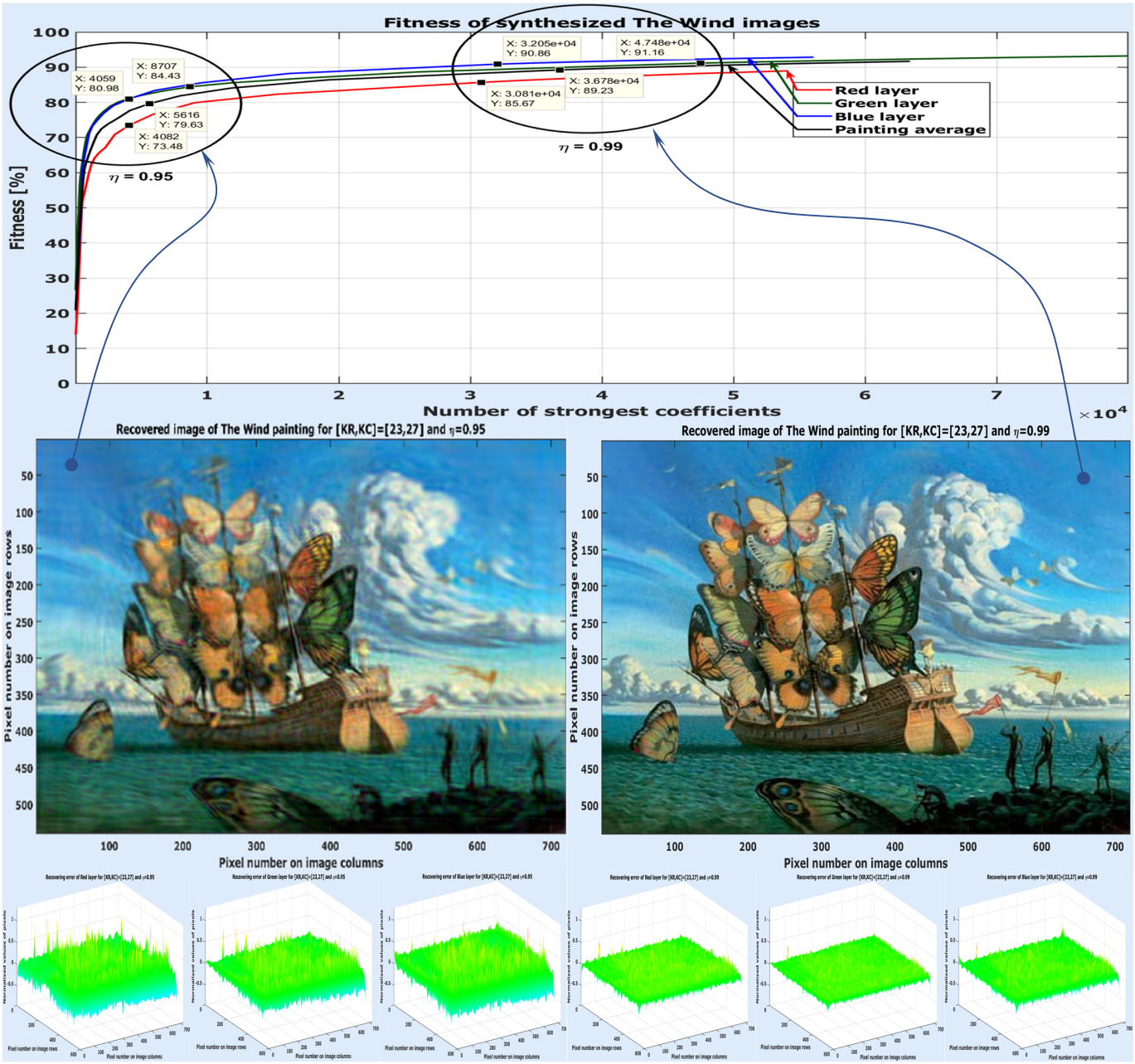

Figure 16. Results of lossy synthesis of *Ship of Dreams/The Wind* painting for $[KR|KC] = [23|27]$. Fitness variations on (**top**). Synthesized images and recovery errors for $\eta = 0.95$ on the (**left**) side and for $\eta = 0.99$ on the (**right**) side.

In both figures, the synthesis results are depicted at the bottom, for $\eta = 0.95$ to the left and for $\eta = 0.99$ to the right. In both cases, the blurring effect is obvious for the smaller threshold. In turn, when the threshold increases, the synthesized images are clearer. Below the images, the recovery errors are shown for each layer (red, green and blue from left to right) after passing to integer pixels. The same representation scale was employed for both thresholds. One can see that the magnitude of errors for $\eta = 0.99$ is smaller, as expected. The fitness variation is drawn above recovered images for each layer. A fourth variation is added, standing for the whole picture, by averaging the layer variations. Unlike for the bw image, in this case, the threshold η led to different nscs, depending on each layer. In Table 5, the number of such coefficients, together with the corresponding fitness values, are listed for the two outlined groups of points in each figure.

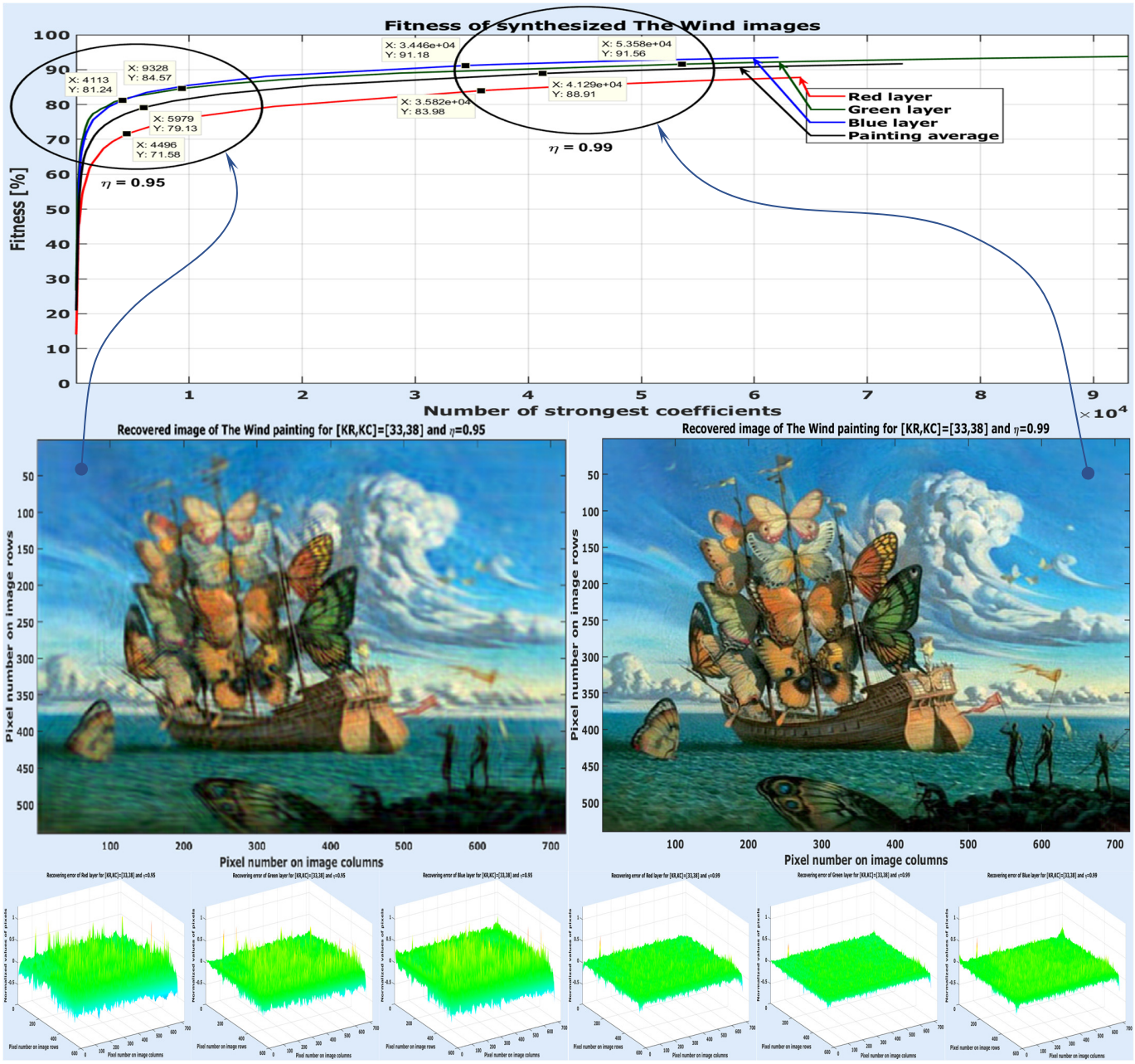

Figure 17. Results of lossy synthesis of *Ship of Dreams/The Wind* painting for $[KR|KC] = [33|38]$. Same window disposal as in Figure 16.

Table 5. Number of strongest coefficients and fitness (accuracy) values in case of *Ship of Dreams/The Wind* painting.

Sampling Rates $[KR\,|\,KC]$	Layer	η=0.95		η=0.99	
		$N_{0.95}$	Accuracy [%]	$N_{0.99}$	Accuracy [%]
[23|27]	R	4082	73.48	30,814	85.67
	G	8707	84.43	47,479	91.16
	B	4059	80.98	32,046	90.86
	P	**16,848**	**79.63**	**110,339**	**89.23**
[33|38]	R	4496	71.58	35,823	83.98
	G	9328	84.57	53,582	91.56
	B	4113	81.24	34,458	91.18
	P	**17,937**	**79.13**	**123,863**	**88.91**

The Green layer seems to be the most accurately recovered, but at the expense of a greater number of employed coefficients (almost twice bigger than for the other layers). This means the Green layer is less auto-correlated than the other two layers. For the whole picture, the nscs were cumulated, while the accuracy was averaged. Although the nsc is quite small for $\eta = 0.95$ (as analyzed above), the synthesized images have modest visual quality. In the case of $\eta = 0.99$, the visual quality is improved. This time, though, the nscs are 6–7 times bigger. More specifically, the nscs are $[110,339|123,863]$. They take $[0.7564|0.5019]\%$ of the total coefficient number. In 64-bit representation, the numbers of bytes that correspond to $[110,339|123,863]$ are $[1,765,424|1,981,808]$, respectively (recall that JvN coefficients are complex-valued). Obviously, both numbers are bigger than the number of bytes of the whole image (1,166,400). Employing the symmetry property becomes crucial for reducing the total number of bytes. In this case, the image can be recovered with good visual quality from approximately $[80.68|89.95]\%$ of the total number of bytes in the image (on all three layers).

The explanation for the last result compared to the previous one, coming from the Daisy image (where the theoretical compression rate was much better), very likely resides in the fact that Dali's painting reveals a much more complex fractal structure. JvNT is a smooth transform with limitations in analyzing and/or the compression of highly fractal images. Normally, the analysis instrument should have the same nature as the analyzed object. In this case, fractal transform (for example, built by using Daubechies' wavelets [34]) could perform better. Dali's painting was selected on purpose to outline, on one hand, that symmetry becomes crucial for transform effectiveness and, on the other hand, that, like any other transform, JvNT has limitations, too, in this case, imposed by its non-fractal nature.

The analysis of lossy synthesis is just the starting point for achieving a sounder analysis concerning the lossy compression of images. When computing the compression ratio, the side information must be accounted for, as well. This means either encoding and sending the positions of the strongest coefficients or sending all coefficients, after setting to null (or even to smaller values) the weakest coefficients. In both cases, the compression performance is reduced, as the total number of bytes necessary to recover the image increases. Usually, after some orthogonal transform is applied to the image, a series of compression algorithms are employed to encode the resulting coefficients. This is a sophisticated endeavor, beyond the goal of the current article.

5. Concluding Remarks

The extension of John von Neumann's orthogonal transform from time-varying (1D) signals to space-varying (2D) signals was the main goal of this article. The requirement to meet was to preserve both the orthogonality and the invertibility properties of resulted 2D transform. Two approaches were investigated in this study: a rotational one and a cartesian one. Because, in the first approach, the orthogonality is very difficult to preserve, the second approach was adopted. Thus, the natural link with 1D transform was exploited, on one hand, to keep intact the orthogonality property and, on the other hand, to design efficient analysis–synthesis algorithms for image processing. Moreover, it has been shown that using a couple of 1D transforms for 2D signals is equivalent to the analysis and synthesis of such signals with 2D transform, thanks to a fortunate factorization of the cardinal sinus. After running the corresponding numerical algorithms on black and white as well as on colored images, the results were analyzed in two main respects: the theoretical compression capacity and the quality of the reconstructed image. According to the obtained results, one can conclude that John von Neumann's 2D transform is a promising instrument to be applied to images prior to other compression methods. Beside compression, 2D transform can be employed in the analysis of any 2D signal (not necessarily an image) and, furthermore, the extension of its definition to multi-dimension signals is straightforward, as, for each new dimension, one new 1D transform can be defined and employed.

Author Contributions: D.S. and J.C. equally contributed to the conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing—original draft preparation, writing—review and editing, visualization, supervision, project administration and funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data employed while testing the numerical algorithms can be sent to readers on request.

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

Acronyms

{1,2,3,4}D	{one,two,three,four} dimension(s)	bw	black and white (image)
DFT(s)	Discrete Fourier Transform(s)	dB	decibels (logarithmic scale)
FT(s)	Fourier Transform(s)	deg	degrees (for angles)
JvN	John von Neumann	fp	floating point (representation)
JvNT(s)	John von Neumann Transform(s)	frFT	fractional Fourier Transform(s)
LPF	Low-Pass Filter	mw	mother waveform/window
RGB	Red-green-blue (image digital system)	nsc(s)	number(s) of strongest coefficients
SP	Signal Processing	sfa(s)	space-frequency atom(s)
TDR	Theorem of Division with Remainder	std	standard deviation
WFT(s)	Windowed Fourier Transform(s)	tfa(s)	time-frequency atom(s)

References

1. Oppenheim, A.V.; Schafer, R. *Digital Signal Processing*; Prentice Hall International Ltd.: London, UK, 1985; ISBN 0-13-214107-8.
2. Proakis, J.G.; Manolakis, D.G. *Digital Signal Processing. Principles, Algorithms and Applications*; Prentice Hall Inc.: Upper Saddle River, NJ, USA, 1996; ISBN 0-13-394338-9.
3. Stefanoiu, D.; Culita, J. John von Neumann's Time-Frequency Orthogonal Transforms. *Mathematics* **2023**, *11*, 2607. [CrossRef]
4. Abdulhussain, S.H.; Ramli, A.; Jassim, W.A. Shot Boundary Detection Based on Orthogonal Polynomial. *Multimed. Tools Appl.* **2019**, *78*, 20361–20382. [CrossRef]
5. Sabbane, F.; Tairi, H. Watermarking Approach Based on Hermite Transform and a Sliding Window Algorithm. *Multimed. Tools Appl.* **2023**, *82*, 47635–47667. [CrossRef] [PubMed]
6. Shao, Z.H.; Tang, Y.D.; Liang, M.; Shang, Y.; Wang, F.; Wang, Y. Double Image Encryption Based on Symmetry of 2D-DFT and Equal Modulus Decomposition. *Multimed. Tools Appl.* **2021**, *80*, 8973–8998. [CrossRef]
7. Cohen, L. *Time-Frequency Analysis*; Prentice Hall: Hoboken, NJ, USA, 1995; ISBN 978-0135945322.
8. Yan, L.; Piao, S.C.; Xu, F. Orthogonal Waveform Separation in Multiple-Input and Multiple-Output Imaging Sonar with Fractional Fourier Filtering. *IET Radar Sonar Navig.* **2021**, *15*, 471–484. [CrossRef]
9. Ranjan, R.; Thakur, A. Image Encryption Using Discrete Orthogonal Stockwell Transform with Fractional Fourier Transform. *Multimed. Tools Appl.* **2023**, *82*, 18517–18527. [CrossRef]
10. Ince, I.F.; Bulut, F.; Kilic, I.; Yildirim, M.E.; Ince, O.F. Low Dynamic Range Discrete Cosine Transform (LDR-DCT) for High-Performance JPEG Image Compression. *Vis. Comput.* **2022**, *38*, 1845–1870. [CrossRef]
11. Da Silveira, T.L.T.; Canterle, D.R.; Cintra, R.J. A Class of Low-Complexity DCT-Like Transforms for Image and Video Coding. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *32*, 4364–4375. [CrossRef]
12. Yao, M.H.; Zheng, S.J.; Hu, Y.; Zhang, Z.; Peng, J.; Zhong, J. Single-Pixel Moving Object Classification with Differential Measuring in Transform Domain and Deep Learning. *Photonics* **2022**, *9*, 202. [CrossRef]
13. Dur-e-Jabeen Khan, T.; Siddique, I.A.A.; Asghar, S. An Algorithm to Reduce Compression Ratio in Multimedia Applications. *CMC-Comput. Mater. Contin.* **2023**, *74*, 539–557. [CrossRef]
14. Kumar, S.S.; Mangalam, H. Quantization Based Wavelet Transformation Technique for Digital Image Compression with Removal of Multiple Artifacts and Noises. *Cluster Comput.—J. Netw. Softw. Tools Appl.* **2019**, *22*, 11271–11284. [CrossRef]
15. Fischer, S.; Sroubek, F. Self-Invertible 2D Log-Gabor Wavelets. *Int. J. Comput. Vis.* **2007**, *75*, 231–246. [CrossRef]
16. Ito, I.; Egiazarian, K. Two-Dimensional Orthonormal Tree-Structured Haar Transform for Fast Block Matching. *J. Imaging* **2018**, *4*, 131. [CrossRef]
17. Kazaryan, M.; Shahramanian, M.; Zabunov, S. Investigation of The Similarity Algorithm of The Satellite Images Storage System for Stability on the Basis of Haar Wavelets According to Tikhonov. *Aerosp. Res. Bulg.* **2019**, *31*, 71–90. [CrossRef]
18. Armah, G.K.; Ahene, E. Application of Residue Number System (RNS) to Image Processing Using Orthogonal Transformation. In Proceedings of the IEEE International Conference on Communication Software and Networks (ICCSN), Chengdu, China, 6–7 June 2015; pp. 322–328.

19. Lu, L.; Shi, B.C. Fast Algorithm of (k, k−1) Type Discrete Walsh-Haar Transformation and Application in Image Edge Detection. In *Remote Sensing Image Processing, Geographic Information Systems, and Other Applications, Proceedings of the 7th Symposium on Multispectral Image Processing and Pattern Recognition (MIPPR), Guilin, China, 4–6 November 2011*; SPIE: Bellingham, WA, USA, 2011; Volume 8006, p. 80062N.
20. Pratt, W.K.; Chen, W.H.; Welch, L.R. Slant Transform Image Coding. *IEEE Trans. Commun.* **1974**, *22*, 1075–1093. [CrossRef]
21. Lian, Q.F.; Shen, L.X.; Xu, Y.; Yang, L. Filters of wavelets on invariant sets for image denoising. *Appl. Anal.* **2011**, *90*, 1299–1322. [CrossRef]
22. Tonge, A.; Thepade, S.D. Creating Video Visual Storyboard with Static Video Summarization using Fractional Energy of Orthogonal Transforms. *Int. J. Adv. Comput. Sci. Appl.* **2022**, *13*, 265–273. [CrossRef]
23. Duris, V.; Chumarov, S.G.; Semenov, V.I. The Orthogonal Wavelets in the Frequency Domain Used for the Images Filtering. *IEEE Access* **2020**, *9*, 211125–211134. [CrossRef]
24. Shin, Y.H.; Park, M.J.; Kim, J.O. Deep Orthogonal Transform Feature for Image Denoising. *IEEE Access* **2020**, *8*, 66898–66909. [CrossRef]
25. Enesi, I.; Harizaj, M.; Cico, B. Implementing Fusion Technique Using Biorthogonal DWT to Increase the Number of Minutiae in Fingerprint Images. *J. Sens.* **2022**, *2022*, 3502463. [CrossRef]
26. Puchala, D. Effective Lattice Structures for Separable Two-Dimensional Orthogonal Wavelet Transforms. *Bull. Pol. Acad. Sci.—Tech. Sci.* **2022**, *70*, e141005. [CrossRef]
27. Forczmanski, P.; Kukharev, G.; Shchegoleva, N. Simple and Robust Facial Portraits Recognition under Variable Lighting Conditions Based on Two-Dimensional Orthogonal Transformations. In Proceedings of the 17th International Conference on Image Analysis and Processing (ICIAP), PT 1 8156, Naples, Italy, 9–13 September 2013; pp. 602–611.
28. Zhu, H.Q.; Gui, Z.G.; Chen, Z.H. Discrete Fractional COSHAD Transform and Its Application. *Math. Probl. Eng.* **2014**, *2014*, 567414. [CrossRef]
29. Rakheja, P.; Singh, P.; Vig, R. An Asymmetric Image Encryption Mechanism Using QR Decomposition in Hybrid Multi-Resolution Wavelet Domain. *Opt. Lasers Eng.* **2020**, *134*, 106177. [CrossRef]
30. Taub, A.H. *John von Neumann Collected Works (6 Volumes), Volume II: Operators, Ergodic Theory and Almost Periodic Functions in a Group*; Pergamon Press Ltd.: Oxford, UK, 1961; ISBN 9780080095660.
31. Daisy on Flikr. Available online: https://www.flickr.com/photos/str-le-ro/6139434734 (accessed on 22 November 2023).
32. Adams, M.D.; Kossentini, F. Reversible Integer-to-Integer Wavelet Transforms for Image Compression: Performance, Evaluation and Analysis. *IEEE Trans. Image Process.* **2000**, *9*, 1010–1024. [CrossRef]
33. Dalal, T.; Yadav, J. Large-Scale Orthogonal Integer Wavelet Transform Features-Based Active Support Vector Machine for Multi-Class Face Recognition. *Int. J. Comput. Appl. Technol.* **2023**, *72*, 108–124. [CrossRef]
34. Daubechies, I. *Ten Lectures on Wavelets*; SIAM—Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 1992; ISBN 9780898712742.