Enhancing sparse representation of color images by cross channel transformation

Laura Rebollo-Neira
Mathematics Department
Aston University
B4 7ET, Birmingham, UK

Aurelien Inacio
ENSIIE, 1 Rue de la Résistance,
91000 Évry-Courcouronnes, France

April 12, 2022

Abstract

Transformations for enhancing sparsity in the approximation of color images by 2D atomic decomposition are discussed. The sparsity is firstly considered with respect to the most significant coefficients in the wavelet decomposition of the color image. The discrete cosine transform is singled out as an effective transformation for this purpose. The enhanced feature is further exploited by approximating the transformed arrays using an effective greedy strategy with a separable highly redundant dictionary. The relevance of the achieved sparsity is illustrated by a simple encoding procedure. On a set of typical test images the compression at high quality recovery is shown to significantly improve upon JPEG, H.265, and WebP formats.

1 Introduction

In the signal processing field sparse representation usually refers to the approximation of a signal as superposition of elementary components, called atoms, which are members of a large redundant set, called a dictionary [1]. The superposition, termed atomic decomposition, aims at approximating the signal involving as few atoms as possible [1–4]. Sparsity is also relevant to data collection. Within the emerging theory of sampling known as *compressive sensing* (CS) this property is key for reconstructing signals from a reduced number of measures [5–7]. In particular, distributed compressive sensing (DCS) algorithms exploit inter signal correlation structure for multiple signal recovery [8, 9].

Sparse image representation using redundant dictionaries has been considered in numerous works e.g. [10–12] and in the context of applications such as image restoration [13–15], feature extraction [16–19] and super resolution [20–23].

The sparsity property of some types of 3D images benefits from 3D processing [24–27]. In particular most RGB (Red-Green-Blue) images admit a sparser atomic decomposition if

approximated by 3D atoms [28]. Within the 3D framework the gain in sparsity comes at expenses of computational cost though.

The purpose of this work is to show that the application of a transformation across the direction of the colors improves sparsity in the wavelet domain representation of the 2D color channels. The relevance of this feature is demonstrated by a simple encoding procedure rendering good compression results in comparison to commonly used image compression standards.

The contributions of the paper are listed below.

- Highlighting of a distinctive feature concerning sparse representation of RGB images. Namely, that transformations in the direction of the color channels significantly improve the sparsity of the 2D atomic decomposition of the image.
- Demonstration that the discrete cosine transform (dct) can improve sparsity with respect to the optimal decorrelation transform, i.e. that given by the eigenvectors of the correlation matrix, which is thereby image dependent.
- Showcasing of the relevance of sparsity to image compression with high quality recovery. On typical test images the achieved compression is shown to significantly improve upon JPEG, WebP, and H.265 formats.

The work is organised as follows: Sec. 2 introduces the mathematical notation. Sec. 3 compares several cross color transformations enhancing sparsity. A numerical example on a large data set is used to illustrate the suitability of the dct for that purpose. Sec. 4 demonstrates the gain in sparsity obtained by the atomic decomposition of color images when the dct is applied across the RGB channels. Sec. 5 illustrates the relevance of the approach to image compression with high quality recovery. The conclusions are summarised in Sec. 6.

2 Mathematical Notation

Throughout the paper we use the following notational convention. $\mathbb{R}$ represents the set of real numbers. Boldface letters indicate Euclidean vectors, 2D and 3D arrays. Standard mathematical fonts indicate components, e.g., $\mathbf{d} \in \mathbb{R}^N$ is a vector of components $d(i) \in \mathbb{R}$, $i = 1, \ldots, N$. The elements of a 2D array $\mathbf{I} \in \mathbb{R}^{L_x \times L_y}$ are indicated as $I(i, j)$, $i = 1, \ldots, L_x$, $j = 1, \ldots, L_y$ and the color channels of $\mathbf{I} \in \mathbb{R}^{L_x \times L_y \times 3}$ as $I(:, :, z)$, $z = 1, 2, 3$. The transpose of a matrix, $\mathbf{G}$ say, is indicated as $\mathbf{G}^\top$. A set of, say M, color images is represented by the arrays $\mathbf{I}\{m\} \in \mathbb{R}^{L_x \times L_y \times 3}, m = 1, \ldots, M$.

The inner product between arrays in $\mathbb{R}^{L_x \times L_y}$ is given by the Frobenius inner product $\langle \cdot, \cdot \rangle_{\mathrm{F}}$ as

$$\langle \mathbf{G}, \mathbf{I} \rangle_{\mathrm{F}} = \sum_{i=1}^{L_x} \sum_{j=1}^{L_y} G(i, j) I(i, j).$$

Consequently, the Frobenius norm $\| \cdot \|_{\mathrm{F}}$ is calculated as

$$\|\mathbf{G}\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{L_x} \sum_{j=1}^{L_y} G(i, j)^2}.$$

The inner produce between arrays in $\mathbb{R}^N$ is given by the Euclidean inner product $\langle \cdot, \cdot \rangle$ as

$$\langle \mathbf{d}, \mathbf{g} \rangle = \sum_{i=1}^{N} d(i)g(i).$$

3 Cross color transformations

Given a color image $I(i, j, k), i = 1, \ldots, L_x, j = 1, \ldots, L_y, k = 1, 2, 3$ the processing of the 3 channels can be realised either in the pixel/intensity or in the wavelet domain. Since the representation of most images is sparser in the wavelet domain [28–31] we approximate in that domain and reconstruct the approximated image by the inverse wavelet transform. Thus, using a 3×3 matrix $\mathbf{T}$, we construct the transformed arrays $\mathbf{U} \in \mathbb{R}^{L_x \times L_y \times 3}$ and $\mathbf{W} \in \mathbb{R}^{L_x \times L_y \times 3}$ as follows

$$U(:, :, z) = \sum_{l=1}^{3} I(:, :, l)T(l, z), \quad z = 1, 2, 3. \tag{1}$$

$$W(:, :, z) = \mathbf{dwt}(U(:, :, z)), \quad z = 1, 2, 3, \tag{2}$$

where **dwt** indicates the 2D wavelet transform. For the transformation $\mathbf{T}$ we consider the following cases

(i) The dct.

(ii) The reversible YCbCr color space transform.

iii) The principal components (PC) transform.

iv) A transformation learned from an independent set of images.

The dct is given by the matrix

$$\begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{\sqrt{2}}{\sqrt{3}}\cos(\frac{\pi}{6}) & \frac{\sqrt{2}}{\sqrt{3}}\cos(\frac{\pi}{3}) \\ \frac{1}{\sqrt{3}} & 0 & \frac{\sqrt{2}}{\sqrt{3}}\cos(\pi) \\ \frac{1}{\sqrt{3}} & \frac{\sqrt{2}}{\sqrt{3}}\cos(\frac{5\pi}{6}) & \frac{\sqrt{2}}{\sqrt{3}}\cos(\frac{5\pi}{3}) \end{pmatrix}.$$

The YCbCr color space transform is given by the matrix below [32]

$$\begin{pmatrix} 0.299 & -0.169 & 0.5 \\ 0.587 & -0.331 & -0.419 \\ 0.114 & 0.5 & -0.0813 \end{pmatrix}.$$

The columns of the principal components transform are the normalised eigenvectors of covariance matrix of the RGB pixels. Thus, the transformation is image dependent and optimal with respect to decorrelating the color channels.

As a first test, the approximation of the transformed channels is realised by keeping a fixed number of the largest absolute value entries, and setting the others equal to zero. In relation to this, for an image of size $L_x \times L_y \times 3$ we define the Sparsity Ratio (SR) as follows

$$\text{SR} = \frac{L_x \cdot L_y \cdot 3}{\text{Number of nonzero entries in the three channels.}} \tag{3}$$

The quality of a reconstructed image $\tilde{\mathbf{I}}$, with respect to the original 8-bit image $\mathbf{I}$, is compared using the Peak Signal-to-Noise Ratio (PSNR),

$$\text{PSNR} = 10 \log_{10} \left(\frac{255^2}{\text{MSE}} \right), \tag{4}$$

where

$$\text{MSE} = \frac{1}{L_x \cdot L_y \cdot 3} \sum_{i,j,z=1}^{L_x,L_y,3} (I(i,j,z) - \tilde{I}(i,j,z))^2.$$

For the numerical examples below the transformation corresponding to case (iv) is learned from a set of images $\mathbf{I}\{m\}$, $m = 1, \ldots, M$ all of the same size. Starting from an invertible 3×3 matrix $\mathbf{T}^k$, with $k = 1$, the learning algorithm proceeds through the following instructions.

1) Use $\mathbf{T}^k$ to transform the arrays $\mathbf{I}\{m\} \to \mathbf{U}^k\{m\} \to \mathbf{W}^k\{m\}$ as in (1) and (2).

2) Approximate each transformed array $\mathbf{W}^k\{m\}$ to obtain $\tilde{\mathbf{W}}^k\{m\}$ by keeping the largest K absolute value entries.

3) Apply the inverse 2D wavelet transform **idwt** to reconstruct the approximated arrays $\tilde{\mathbf{U}}^k\{m\}$, $m = 1, \ldots, M$ as

$$\tilde{U}^k\{m\}(:,:,z) = \mathbf{idwt}(\tilde{W}^k\{m\}(:,:,z)), \quad z = 1, 2, 3.$$

4) Use the original images $\mathbf{I}\{m\}$, $m = 1, \ldots, M$ to find $\mathbf{G} = \mathbf{T}^{-1}$ by least squares fitting, i.e

$$\mathbf{G}^* = \underset{\substack{G(z,l) \\ z,l=1,2,3}}{\arg\min}\, \mathcal{E}(\mathbf{G}),$$

where

$$\mathcal{E}(\mathbf{G}) = \sum_{m=1}^{M} \sum_{i,j,l=1}^{L_x,L_y,3} (I\{m\}(i,j,l) - \tilde{I}\{m\}(i,j,l))^2$$

with

$$\tilde{I}\{m\}(i,j,l)) = \sum_{z=1}^{3} \tilde{U}^k\{m\}(i,j,z) G(z,l).$$

5) While $\mathcal{E}$ decreases, or the maximum number of allowed iterations has not been reached, set $k \to k+1$, $\mathbf{T}^k = (\mathbf{G}^*)^{-1}$ and repeat steps 1) – 5).

Given the arrays $\tilde{\mathbf{U}}^k\{m\}$, $m = 1, \ldots, M$ the least squares problem for determining the transformation $\mathbf{T}^k$ has a unique solution. However, the joint optimisation with respect to the arrays $\tilde{\mathbf{U}}^k\{m\}, m = 1, \ldots, M$ *and* the transformation $\mathbf{T}^k$ is not convex. Hence, the algorithm's outcome depends on the initial value.

The transformation (iv) used in the numerical examples of Secs. 3.1 and 4.1 has been learned from $M = 100$ images, all of size $384 \times 512 \times 3$, from the UCID data set [33], which contains images of buildings, places and cars. The learning curves for two random orthonormal initialisations is shown in Fig. 1.

It is worth mentioning that, as shown in Fig. 1, learning is richer when starting comparatively distant from a local minimum (left graph in Fig. 1). However, since the convergence is to a local minimum other random initialisations, even if generating less learning, may produce better results (right graph in Fig. 1).

The aim of the next numerical test is to demonstrate the effect on the SR (c.f. (3)) produced by the above (i) - (iv) transformations across the color channels.

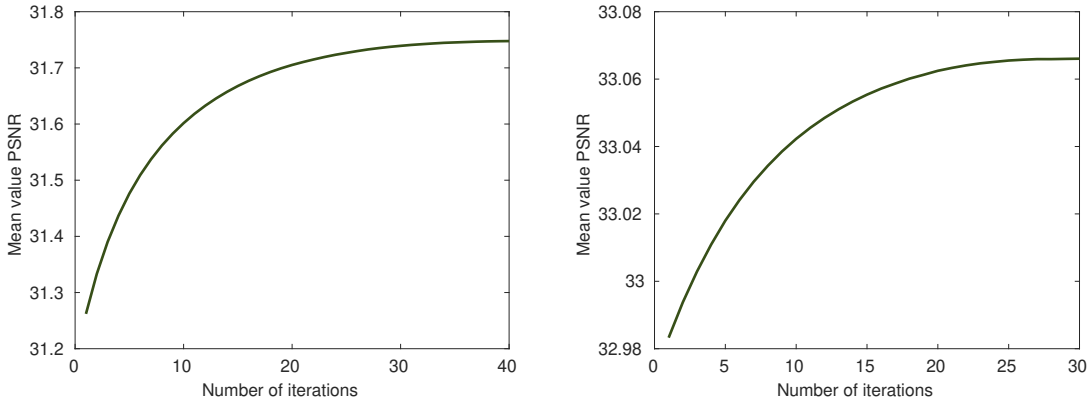

Figure 1: Learning curves for the transformation (iv) corresponding to two different random orthogonal transforms initialising the process. The mean value PSNR with respect to the 100 images in the training set corresponds to SR=15 for all the images.

3.1 Numerical Test I

Using the whole Berkeley data set [34], consisting of 300 images all of size $321 \times 481 \times 3$ we proceed with each image as in (1) and (2). The **dwt** corresponds to the Cohen-Daubechies-Feauveau 9/7 (CDF97) wavelet family. Fig. 3 shows the transformed channels of the image in Fig.2, including the dct transformation across channels (right graph) and without $\mathbf{T}$ transformation (left graph). As noticeable in the figure, the effect of the dct is to re-distribute the intensity in the color channels by transferring values between channels. In particular, after applying dct each point in the transformed channel 1 is proportional to the addition of the intensity of the 3 channels, i.e. for $z = 1$ (1) becomes

$$U(:,:,1) = \frac{1}{\sqrt{3}}(I(:,:,1) + I(:,:,2) + I(:,:,3)),$$

while for $z = 2$ the contribution of the intensity of channel 2 is null. Since

$$U(:,:,2) = \frac{\sqrt{2}}{2}(I(:,:,1) - I(:,:,3)),$$

pixels of similar intensity in channels 1 and 3 produce pixels corresponding to small absolute value in the transformed channel 2. Moreover, since

$$U(:,:,3) = (I(:,:,1) + I(:,:,3))\frac{\sqrt{3}}{2\sqrt{2}} - \frac{\sqrt{3}}{\sqrt{2}}I(:,:,2),$$

pixels of similar intensity in the 3 color channels would barely be noticed in the transformed channel 3.

For the numerical test the approximations are realised fixing SR=20 and SR=10. The reconstructed images are obtained for the approximated arrays $\tilde{W}$ as

$$\tilde{U}(:,:,z) = \mathbf{idwt}(\tilde{W}(:,:,z)), \qquad z = 1,2,3 \tag{5}$$

$$\tilde{I}(:,:,z) = \sum_{k=1}^{3} \tilde{U}(:,:,k)(T^{-1}(z,k)), \ z = 1,2,3 \tag{6}$$

Figure 2: One of the RGB images in the Berkeley's data set

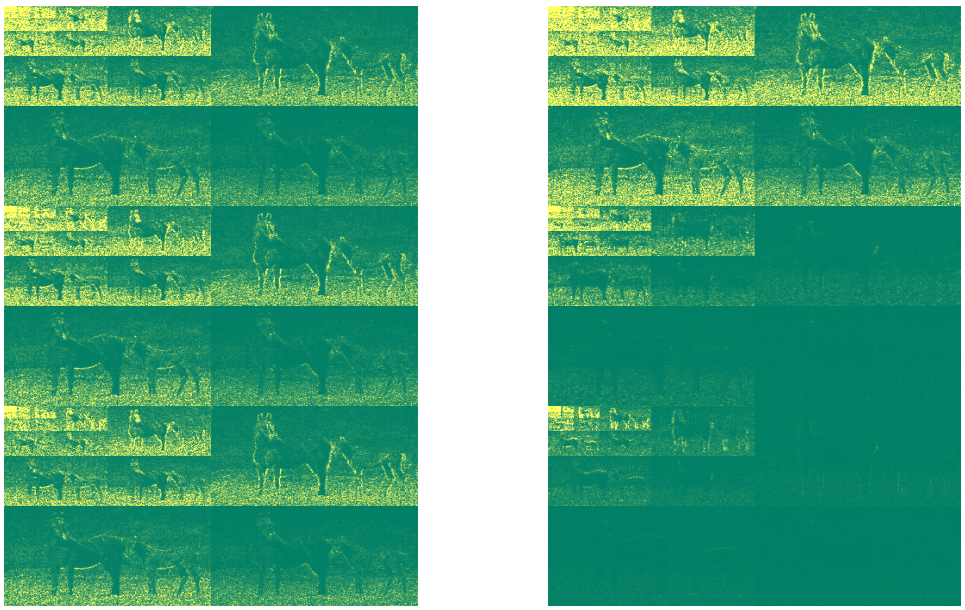

Figure 3: Magnitude of the entries in the array $\mathbf{W}$ constructed as in (2) from the image of Fig. 2, with $\mathbf{T}$ the dct (right graph) and without $\mathbf{T}$ (left graph).

where $\mathbf{T}^{-1}$ is the inverse of $\mathbf{T}$. When no transformation $\mathbf{T}$ is considered the image is reconstructed directly from (5).

The 2nd and 4th columns of Table 1 show the mean value PSNR ($\overline{\text{PSNR}}$), with respect to the whole data set, for SR=20 and SR=10 respectively, corresponding to the transformations

Transf.	$\overline{\text{PSNR}}$	std	$\overline{\text{PSNR}}$	std
(i) dct	33.9	5.0	38.9	5.1
(ii) YCbCr	33.7	5.0	38.7	5.1
(iii) PC	33.7	4.9	38.4	5.0
(iv) Learned	33.7	4.9	38.7	5.0
(v) No transf.	29.7	4.8	32.9	5.2

Table 1: Mean value PSNR, with respect to the 300 images in the Berkeley data set. The approximation of each image is realised by setting the least significant entries in the arrays $\mathbf{W}\{m\} = 1, \ldots, 300$ equal zero, in order to obtain SR=20 for all the images (2nd column) and SR=10 for all the images (4th column).

T listed in the 1st column of the table. The 3rd and 5th columns give the standard deviations (std).

While Table 1 shows that all (i) - (iv) transformations render equivalent superior results, with respect to not applying a transformation (case (v)), the dct slightly exceeds the others. Case (iv) refers to the best result achieved when initialising the learning algorithm with 500 different random orthonormal transformations. The resulting matrix is given below

$$\begin{pmatrix} -0.620 & -0.038 & -0.919 \\ -0.586 & -0.848 & 1.017 \\ -0.518 & 0.933 & -0.038 \end{pmatrix}.$$

When initialising the algorithm with transformations (i) and (ii) it appears that each of these transformations is close to a local minimiser of the method. This stems from the fact that such initialisations do not generate significant learning.

The common feature of most of the 300 images in the data set used in this numerical example is the correlation property of the three color channels. This property is assessed by the correlation coefficients

$$\begin{aligned} r(z) &= \frac{\sum_{i=1}^{L_x}\sum_{j=1}^{L_y}\Gamma(i,j,z)\Gamma(i,j,z+1)}{\sigma(z)\sigma(z+1)}, \quad z = 1, 2, \\ r(3) &= \frac{\sum_{i=1}^{L_x}\sum_{j=1}^{L_y}\Gamma(i,j,1)\Gamma(i,j,3)}{\sigma(1)\sigma(3)}, \end{aligned}$$

where

$$\Gamma(i,j,z) = (I(i,j,z) - \overline{I(:,:,z)}), \tag{7}$$

$$\sigma(z) = \sqrt{\sum_{i=1}^{L_x}\sum_{j=1}^{L_y}(I(i,j,z) - \overline{I(:,:,z)})^2}, \tag{8}$$

and $\overline{I(:,:,z)}$ indicates the mean value of channel z.

As seen in the 3 histograms of Fig. 4 for most images in the data set the correlation of the color channels is high. It is surprising then than the PC transformation (iii), which completely decorrelates the channels, does not overperform the other transformations, on the contrary. This feature has also been noticed in the context of bit allocation for subband color image compression [35].

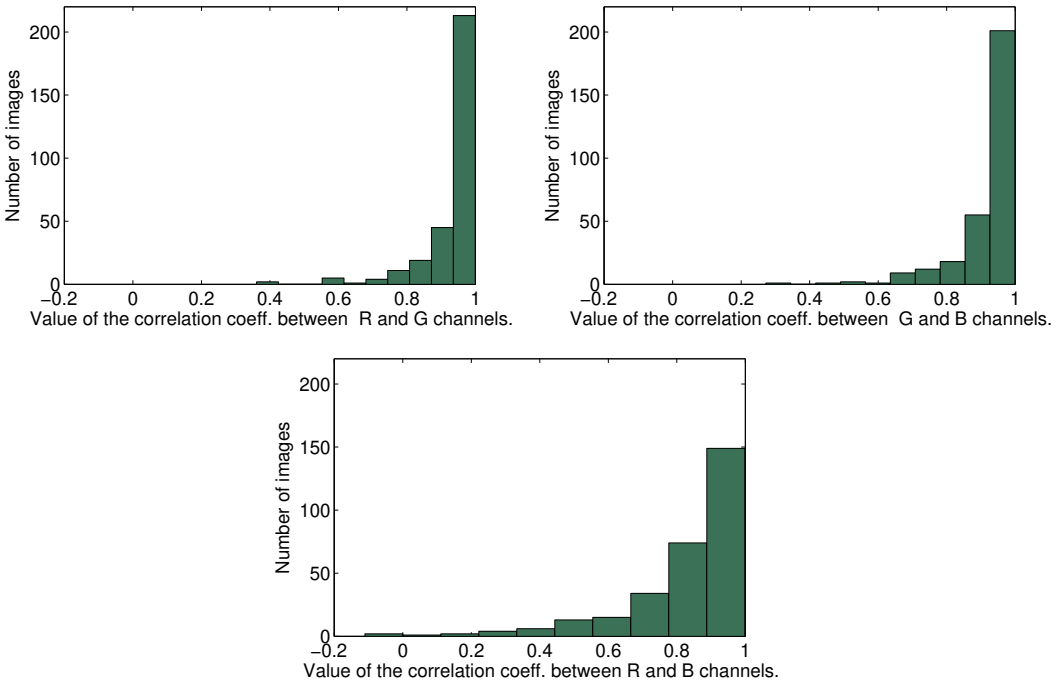

Figure 4: Histograms of the correlation coefficients between the RGB channels.

4 Approximations by atomic decomposition

We have seen (c.f. Table 1) that by transformation of channels it is possible to gain quality when reducing nonzero entries in the channels. Now we discuss how to improve quality further by approximating the 2D arrays (2) by an atomic decomposition, other than just by neglecting their less significant entries. For the approximation to be successful it is important to use an appropriate dictionary. To this end, one possibility could be to learn the dictionary from training data [36–43]. However as demonstrated in previous works [28, 30, 31] a separable dictionary, which is easy to construct, is well suited for the purposes of achieving sparsity and delivers a fast implementation of the approach. Since we use that dictionary in the numerical examples, below we describe the method for constructing the atomic decomposition of the array $\mathbf{W}$ considering specifically a separable dictionary.

Firstly we concatenate the 3 planes $W(:,:,z), z = 1, 2, 3$ into an extended 2D array $\mathbf{W}' \in \mathbb{R}^{3L_x \times L_y}$ and divide this array in small blocks $\mathbf{W}'_q$, $q = 1, \ldots, Q$. Without loss of generality the blocks are assumed to be square of size $N_b \times N_b$ say, and are approximated using separable dictionaries $\mathcal{D}^x = \{\mathbf{d}^x_n \in \mathbb{R}^{N_b},\ \|\mathbf{d}^x_n\|_2 = 1\}_{n=1}^{M_b}$ and $\mathcal{D}^y = \{\mathbf{d}^y_m \in \mathbb{R}^{N_b},\ \|\mathbf{d}^y_n\|_2 = 1\}_{m=1}^{M_b}$.

For $q = 1, \ldots, Q$ every element $\mathbf{W'}_q \in \mathbb{R}^{N_b \times N_b}$ is approximated by an *atomic decomposition* as below:

$$\mathbf{W'}_q^{k_q} = \sum_{n=1}^{k_q} c^{k_q,q}(n) \mathbf{d}^x_{\ell^{x,q}_n} (\mathbf{d}^y_{\ell^{y,q}_n})^T, \tag{9}$$

where $\ell^{y,q}_n$ is the index in the set $\{1, 2, \ldots, M_b\}$ corresponding to the label of the atom in the dictionary $\mathcal{D}^y$ contributing to the n-th term in the approximation of the q-th block. The index $\ell^{x,q}_n$ has the equivalent description. The assembling of the approximated blocks gives rise to the approximated array $\mathbf{W'}^a = \hat{\mathrm{J}}_{q=1}^{Q} \mathbf{W'}_q^{k_q}$, where $\hat{\mathrm{J}}$ represents the assembling operation, i.e. the

operation that retrieves the approximation $\mathbf{W}'^a \in \mathbb{R}^{3L_x \times L_y}$ of the array $\mathbf{W}' \in \mathbb{R}^{3L_x \times L_y}$ from the approximation of the blocks in the partition. The approximated array $\mathbf{W}'^a \in \mathbb{R}^{3L_x \times L_y}$ is reshaped back into 3 planes, $W^a(:,:,z) \in \mathbb{R}^{L_x \times L_y}$, $z = 1, \ldots, 3$, to be converted back to the approximated RGB intensity image as in (5) and (6).

The approximation of the partition $\mathbf{W'}_q \in \mathbb{R}^{N_b \times N_b}$, $q = 1, \ldots, Q$ is carried out iteratively as a two step process which selects i) the atoms in the atomic decomposition (9) and ii) the sequence in which the blocks in the partition are approximated. The procedure is called Hierarchized Block Wise (HBW) implementation of greedy pursuit strategies [29,44]. For the selection of the atoms we apply the Orthogonal Matching Pursuit (OMP) approach [45] dedicated to 2D with separable dictionaries (OMP2D) [46]. Thus, the whole algorithm is termed HBW-OMP2D [29]. The method iterates as described below.

On setting $k_q = 0$ and $\mathbf{R}_q^0 = \mathbf{W'}_q \in \mathbb{R}^{N_b \times N_b}$ at iteration $k_q + 1$ the algorithm selects the indices $\ell_{k_q+1}^{x,q}$ and $\ell_{k_q+1}^{y,q}$, as follows:

$$\ell_{k_q+1}^{x,q}, \ell_{k_q+1}^{y,q} = \underset{\substack{n=1,\ldots,M_b \\ m=1,\ldots,M_b}}{\arg\max} \left| \left\langle \mathbf{d}_n^x, \mathbf{R}_q^{k_q} \mathbf{d}_m^y \right\rangle_F \right|, \quad q = 1, \ldots, Q, \tag{10}$$

where $\mathbf{R}_q^{k_q} = \mathbf{W'}_q - \mathbf{W'}_q^{k_q}$. The calculation of $\mathbf{W'}_q^{k_q}$ is realised in order to minimise $\|\mathbf{R}_q^{k_q}\|_F$, which is equivalent to finding the orthogonal projection onto the subspace spanned by the selected atoms$\{\mathbf{A}_n = \mathbf{d}_{\ell_n^{x,q}}^x (\mathbf{d}_{\ell_n^{y,q}}^y)^T \in \mathbb{R}^{N_b \times N_b}\}_{n=1}^{k_q}$. In our implementation, the calculation of the coefficients $c^q(n)$, $n = 1, \ldots, k_q$ in (9) is realised as

$$c^{k_q,q}(n) = \left\langle \mathbf{B}_n^k, \mathbf{W'}_q^{k_q} \right\rangle_{\mathrm{F}}, \quad n = 1, \ldots, k_q, \tag{11}$$

where the set $\{\mathbf{B}_n^{k_q} \in \mathbb{R}^{N_b \times N_b}\}_{n=1}^{k_q}$ is biorthogonal to the set $\{\mathbf{A}_n \in \mathbb{R}^{N_b \times N_b}\}_{n=1}^{k_q}$ and needs to be upgraded and updated to account for each newly selected atom. Starting from $\mathbf{B}_1^1 = \mathbf{Q}_1 = \mathbf{A}_1 = \mathbf{d}_{\ell_1^x}^x (\mathbf{d}_{\ell_1^y}^y)^\top$ the updating and upgrading is realised through the recursive equations [46,47]:

$$\begin{aligned}
&\mathbf{B}_n^{k_q+1} = \mathbf{B}_n^{k_q} - \mathbf{B}_{k_q+1}^{k_q+1} \left\langle \mathbf{A}_{k_q+1}, \mathbf{B}_n^k \right\rangle_{\mathrm{F}}, \quad n = 1, \ldots, k_q \\
&\mathbf{B}_{k_q+1}^{k_q+1} = \mathbf{Q}_{k_q+1} / \|\mathbf{Q}_{k_q+1}\|_{\mathrm{F}}^2, \text{ where} \\
&\mathbf{Q}_{k_q+1} = \mathbf{A}_{k_q+1} - \sum_{n=1}^{k_q} \frac{\mathbf{Q}_n}{\|\mathbf{Q}_n\|_{\mathrm{F}}^2} \left\langle \mathbf{Q}_n, \mathbf{A}_{k_q+1} \right\rangle_{\mathrm{F}},
\end{aligned} \tag{12}$$

with the additional re-orthogonalisation step

$$\mathbf{Q}_{k_q+1} \leftarrow \mathbf{Q}_{k_q+1} - \sum_{n=1}^{k_q} \frac{\mathbf{Q}_n}{\|\mathbf{Q}_n\|_{\mathrm{F}}^2} \cdot \left\langle \mathbf{Q}_n, \mathbf{Q}_{k_q+1} \right\rangle_{\mathrm{F}}. \tag{13}$$

As discussed in [29, 44], for $\ell_{k_q+1}^{x,q}$ and $\ell_{k_q+1}^{y,q}$, $q = 1, \ldots, Q$ the indices resulting from (10), the block to be approximated in the next iteration corresponds to the value $q^\star$ such that

$$q^\star = \underset{q=1,\ldots,Q}{\arg\max} \left| \left\langle \mathbf{d}_{\ell_{k_q+1}^{x,q}}^x, \mathbf{R}_q^{k_q} \, \mathbf{d}_{\ell_{k_q+1}^{y,q}}^y \right\rangle \right|.$$

The algorithm stops when the required total number of $K = \sum_{q=1}^{Q} k_q$ atoms has been selected. This number can be fixed using the SR, which is now calculated as

$$\mathrm{SR} = \frac{L_x \cdot L_y \cdot 3}{K}. \tag{14}$$

Remark 1. *The above described implementation of HBW-OMP2D is very effective in terms of speed, but demanding in terms of memory (the partial outputs corresponding to all the blocks in the partition need to be stored at every iteration). An alternative implementation, termed HBW Self Projected Matching Pursuit (HBW-SPMP) [30, 48], would enable the application of the identical strategy to much larger images than the ones considered in this work.*

4.1 Numerical example II

For this and the next numerical example, we use a mixed dictionary consisting of two classes of sub-dictionaries of different nature:

I) The trigonometric dictionaries $\mathcal{D}_C^x$ and $\mathcal{D}_S^x$, defined below,

$$\mathcal{D}_C^x = \{w_c(n)\cos\frac{\pi(2i-1)(n-1)}{2M}, i = 1,\ldots,N_b\}_{n=1}^{M_x}$$
$$\mathcal{D}_S^x = \{w_s(n)\sin\frac{\pi(2i-1)(n)}{2M}, i = 1,\ldots,N_b\}_{n=1}^{M_x},$$

where $w_c(n)$ and $w_s(n)$, $n = 1,\ldots,M_x$ are normalisation factors.

II) The dictionary $\mathcal{D}_L^x$, which is constructed by translation of the prototype atoms in Fig. 5.

The mixed dictionary $\mathcal{D}^x$ is built as $\mathcal{D}^x = \mathcal{D}_C^x \cup \mathcal{D}_S^x \cup \mathcal{D}_L^x$ and $\mathcal{D}^y = \mathcal{D}^x$.

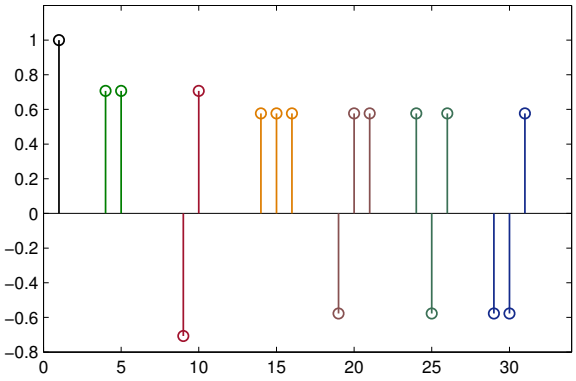

Figure 5: Prototypes (each in different color) that generate the dictionaries $\mathcal{D}_L^x$ by sequential translations of one point.

Table 2 shows the improvement in $\overline{\text{PSNR}}$ achieved by atomic decompositions using the mixed dictionary for SR= 20 and SR= 10.

Notice that while case (v), which does not include any **T** transformation, gives superior results than by disregarding entries (c.f. Table 1) when applying any of the transformations (i) – (iv) the results improve further. The PC transform, however, appears significantly less effective than the others. In addition to rendering the best results, the dct brings along the additional advantage of being orthonormal. Consequently, it does not magnify errors at the inversion step. Because of this, we single out the dct as the most convenient cross color transformation out of the four considered here.

Block size	8×8				16×16			
Transf.	$\overline{\text{PSNR}}$	std	$\overline{\text{PSNR}}$	std	$\overline{\text{PSNR}}$	std	$\overline{\text{PSNR}}$	std
(i) dct	40.5	5.0	48.1	4.4	40.8	4.9	48.6	4.2
(ii) YCbCr	40.3	5.0	47.8	4.4	40.6	4.9	48.3	4.2
(iii) PC	39.6	4.8	46.3	4.5	39.9	4.8	46.7	4.5
(iv) Learned	40.3	4.9	47.4	4.3	40.5	4.9	47.8	4.2
(v) No transf.	34.0	5.2	39.1	5.6	34.1	5.2	39.3	5.5

Table 2: Mean value PSNR, with respect to 300 images in the Berkeley data set, produced by 2D atomic decomposition of the arrays $\mathbf{W}\{m\} = 1, \ldots, 300$ in order to obtain SR=20 (2nd and 6th column) and SR=10 (4th and 8th column).

5 Application to image compression

In order achieve compression by filing an atomic decomposition we need to address two issues. Namely, the quantisation of the coefficients $c_q(n)\, n = 1, \ldots, k_q$, $q = 1, \ldots, Q$ in (9) and the storage of the indices $(\ell_n^{x,q}, \ell_n^{y,q}), n = 1, \ldots, k_q$, $q = 1, \ldots, Q$. We tackle the matters by simple but effective procedures [31].

For $q = 1, \ldots, Q$ the absolute value coefficients $|c_q(n)|$, $n = 1, \ldots, k_q$ are converted to integers through uniform quantisation as follows

$$c_q^{\Delta}(n) = \begin{cases} |c_q(n)| \geq \theta & \\ & 0 \quad \text{otherwise.} \end{cases} \tag{15}$$

The signs of the coefficient are encoded separately as a vector, $\mathbf{s}_q$, using a binary alphabet. Each pair of indices $(\ell_n^{x,q}, \ell_n^{y,q})$ corresponding to the atoms in the decompositions of the block $\mathbf{W}_q'$ is mapped into a single index $o_q(n)$. The set $o_q(1), \ldots, o_q(k_q)$ is sorted in ascending order $o_q(n) \to \tilde{o}_q(n)$, $n = 1, \ldots, k_q$ to take the differences $\delta_q(n) = \tilde{o}_q(n) - \tilde{o}_q(n-1)$, $n = 2, \ldots, k_q$ and construct the string of non-negative numbers $\tilde{o}_q(1), \delta_q(2), \ldots, \delta_q(k_q)$. The order of the set $\tilde{o}_q(n)$, $n = 1, \ldots, k_q$ induces order in the unsigned coefficients, $c_q^{\Delta}(n) \to \tilde{c}_q^{\Delta}(n)$, and in the corresponding signs $s_q(n) \to \tilde{s}_q(n)$.

For each q the number 0 is added at the end of the indices $\tilde{o}_q(n)$, $n = 1, \ldots, k_q$ before concatenation, to be able to separate strings corresponding to different blocks. Each sequence of strings corresponding to $q = 1, \ldots, Q$ is concatenated and encoded using the off-the-shelf MATLAB function `Huff06` [49], which implements Huffman coding.

The compression rate is given in bits-per-pixel (bpp) which is defined as

$$\text{bpp} = \frac{\text{Size of the file in bits}}{\text{Number of intensity pixels in a single channel}}.$$

At the reconstruction stage the indices $(\tilde{\ell}_n^{x,q}, \tilde{\ell}_n^{y,q})$, $n = 1 \ldots k_q$ are recovered from the string of differences $\delta_q(n))$, $n = 2, \ldots, k_q$. The signs of the coefficients are read from the binary string. The quantised unsigned coefficients are read and transformed into real numbers as:

$$|\tilde{c}_q^{\,\text{r}}(n)| = \Delta \cdot \tilde{c}_q^{\Delta}(n) + (\theta - \Delta/2) \quad n = 1 \ldots k_q.$$

The codec for reproducing the examples in the next sections has been made available on [50].

5.1 Numerical Example III

The relevance to image compression of the achieved sparsity by dct cross color transformation is illustrated in this section by comparison with results yielded by the compression standards JPEG, WebP and H.265, firstly on the 15 images in Table 3 and then on the Kodak data set.

The images in Table 3 are typical images, used for compression tests, available in ppm or png format. The first 9 images are classic test images taken from [51]. The last six images are portions of $1024 \times 1024 \times 3$ pixels shown in Fig. 6 from very large high resolution images available on [52].

All the results have been obtained in the MATLAB environment (version R2019a), using a machine with CPU Intel(R) Core(TM) i7-3520M RAM 8GB CPU @ 2.90GHz. For the image approximation the HBW-OMP2D method was implemented with a C++ MEX file. All the channels and all the images were partitioned into blocks of size 16×16. The approximation times to produce the results in Table 4 are displayed in the last column of Table 3.

For realising the comparison we proceed as follows: we set the required value of PSNR as that produced by JPEG at quality=95 and tune compression with the other methods to produce the same PSNR. In our codec the tuning is realised by approximating the image up to $\text{PSNR}_o = 1.025 \cdot \text{PSNR}$ (where PSNR is the targeted quality) and setting the quantisation parameter Δ so as to reproduce the targeted value of PSNR.

For compression with JPEG we use the MATLAB `imwrite` command. The compression with WebP was realised using the software for Ubuntu distributed on [53]. The compression with the H.265 encoder was implemented with the `ffmpeg` tool. All the approaches were tuned for producing the same value of PSNR as JPEG for quality 95. The Mean Structural SIMilarity (MSSIM) index [54] was then calculated with the approximation corresponding to those values of PSNR.

No	Image	Size	time (secs)
1	Lenna	$512 \times 512 \times 3$	1.8
2	Goldhill	$576 \times 720 \times 3$	3.8
3	Barbara	$576 \times 720 \times 3$	3.2
4	Baboon	$512 \times 512 \times 3$	2.7
5	Zelda	$576 \times 784 \times 3$	2.5
6	Sailboat	$512 \times 512 \times 3$	1.8
7	Boy	$512 \times 768 \times 3$	3.8
8	Jupiter	$1072 \times 1376 \times 3$	2.9
9	Saturn	$1200 \times 1488 \times 3$	2.6
10	Building	$1024 \times 1024 \times 3$	9.5
11	Cathedral	$1024 \times 1024 \times 3$	7.6
12	Flower	$1024 \times 1024 \times 3$	3.6
13	Spider-web	$1024 \times 1024 \times 3$	3.8
14	Bridge	$1024 \times 1024 \times 3$	8.5
15	Deer	$1024 \times 1024 \times 3$	5.8

Table 3: Test Images. The last column gives the approximation times to produce the results in Table 4.

Figure 6: Illustration of the $1024 \times 1024 \times 3$ panels from the six high resolution test images (No 10–15) listed in Table 3.

I	dB	ssjp	sswb	ssh65	sssr	bjp	bwb	bh65	bsr
1	35.9	0.98	0.98	0.97	0.97	3.28	2.21	1.90	**1.37**
2	36.6	0.98	0.98	0.98	0.98	3.63	2.47	3.07	**2.01**
3	37.2	0.98	0.99	0.99	0.99	3.67	2.80	2.96	**1.77**
4	28.8	0.97	0.97	0.96	0.96	5.80	4.66	3.41	**2.48**
5	39.3	0.98	0.98	0.98	0.95	2.61	1.81	1.81	**1.07**
6	30.9	0.98	0.98	0.95	0.95	4.49	3.21	1.87	**1.51**
7	32.6	0.97	0.98	0.96	0.97	4.34	3.07	3.52	**2.22**
8	48.2	0.99	0.99	0.99	0.99	0.60	0.51	0.37	**0.20**
9	49.0	0.99	0.99	0.99	0.99	0.34	0.36	0.24	**0.12**
10	37.4	0.99	0.99	0.99	0.99	3.41	2.35	2.73	**1.75**
11	38.5	0.99	0.99	0.99	0.99	2.84	1.83	1.54	**1.20**
12	41.5	0.99	0.99	0.99	0.99	1.78	1.08	**0.43**	0.53
13	45.0	0.99	0.99	0.99	0.99	1.55	1.10	0.82	**0.57**
14	34.5	0.99	0.99	0.99	0.99	3.57	2.48	1.74	**1.48**
15	30.9	0.97	0.97	0.96	0.96	3.76	2.59	1.05	**0.90**

Table 4: Compression rate (bpp) corresponding to JPEG (bjp), WebP (bwb), H.265 (bh65), and the proposed sparse representation (bsr), for the values of PSNR given in the 2nd column. The corresponding values of MSSIM are given in in the 3rd to 6th columns. ssjp, sswb, ssh65 and sssr indicate the MSSIM produced by JPEG, WebP, H.265, and the sparse representation codec respectively.

For the next test we use the Kodak data set,consisting of 24 true color images, all of size $512 \times 768 \times 3$ (or $768 \times 512 \times 3$), available on [55]. In this case the approximation time is 4.2 secs per image. The comparison with other methods was conducted as with the previous data set. The results are shown in Table 5.

It is worth noticing that while for the first set of images the H.265 codec outperforms both JPEG and WebP methods for most images, in the Kodak data set it becomes the worst performer. In this case WebP standard yields the second best results.

6 Conclusions

The application of a cross color transformation for enhancing sparsity in the atomic decomposition of RGB images has been proposed. It was demonstrated that the effect of the transformation is to re-distribute the most significant values in the dwt of the 2D channels. As a result, when approximating the arrays by disregarding the less significant entries, the quality of the reconstructed image improves with respect to disabling the cross color transformation. Four transformations have been considered: (i) a 3 point dct, (ii) the reversible YCbCr color space transform, (iii) the PC transform, (iv) a transformation learned from an independent set of images.

The quality of the image approximation was improved further by approximating the transformed arrays by an atomic decomposition using a separable dictionary and the greedy pursuit strategy HBW-OMP2D. The dct was singled out as the most convenient cross color transformation for approximating RGB color images in the wavelet domain.

I	dB	ssjp	sswb	ssh65	sssr	bjp	bwb	bh65	bsr
1	40.5	0.99	0.99	0.99	0.99	4.4	**3.1**	6.9	**3.2**
2	40.2	0.99	0.99	0.99	0.99	3.0	1.9	3.1	**1.6**
3	42.2	0.99	0.99	0.99	0.99	2.4	1.6	2.3	**1.2**
4	40.9	0.99	0.99	0.99	0.99	3.1	2.1	3.5	**1.9**
5	39.2	0.99	0.99	0.99	0.99	4.6	3.3	5.6	**3.1**
6	40.7	0.99	0.99	0.99	0.99	3.6	2.6	5.2	**2.4**
7	41.9	0.99	0.99	0.99	0.99	2.7	1.7	2.5	**1.5**
8	39.2	0.99	0.99	0.99	0.99	4.7	3.3	6.2	**3.2**
9	41.7	0.99	0.99	0.99	0.99	2.7	1.8	3.3	**1.4**
10	41.4	0.99	0.99	0.99	0.99	2.8	2.0	3.2	**1.6**
11	40.9	0.99	0.99	0.99	0.99	3.4	2.4	4.7	**2.3**
12	42.2	0.99	0.99	0.99	0.99	2.7	2.0	3.3	**1.5**
13	38.7	0.99	0.99	0.99	0.99	5.2	**3.8**	8.2	3.9
14	39.0	0.99	0.99	0.99	0.99	4.1	2.9	5.0	**2.5**
15	40.8	0.99	0.99	0.99	0.99	2.8	1.9	3.0	**1.6**
16	42.2	0.99	0.99	0.99	0.99	3.0	2.0	4.2	**1.8**
17	41.5	0.99	0.99	0.99	0.99	3.0	2.0	3.6	**1.7**
18	39.0	0.99	0.99	0.99	0.99	4.1	3.0	5.2	**2.7**
19	41.0	0.99	0.99	0.99	0.99	3.4	2.5	4.7	**2.1**
20	41.2	0.99	0.99	0.99	0.99	2.4	1.5	2.8	**1.4**
21	40.5	0.99	0.99	0.99	0.99	3.4	2.6	5.1	**2.2**
22	39.5	0.99	0.99	0.99	0.99	3.5	2.4	4.1	**2.2**
23	41.3	0.99	0.99	0.99	0.99	2.4	1.7	1.8	**0.9**
24	38.0	0.99	0.99	0.99	0.99	3.9	2.7	4.0	**2.4**

Table 5: Same description as in Table 4 but for the 24 images in Kodak data set.

The approximation approach has been shown to be relevant for image compression. By means of a simple coding strategy the achieved compression for typical test images considerably improves upon the most commonly used compression standards, namely JPEG and WebP. The results also overperfom those produced by the H.265 codec.

References

[1] S. Mallat and Z. Zhang, "Matching pursuit with time-frequency dictionaries," *IEEE Trans. Signal Process.*, Vol (41,12) 3397–3415 (1993).

[2] S. Chen, D. Donoho and M. Saunders "Atomic Decomposition by Basis Pursuit", *SIAM Review*, 129–159 (2001).

[3] S. Mallat, A Wavelet Tour of Signal Processing: The Sparse Way, Academic Press (2009).

[4] Y. Eldar, P. Kuppinger and H. Biölcskei, "Block-Sparse Signals: Uncertainty Relations and Efficient Recovery", *IEEE Trans. Signal Process.*, **58**, 3042–3054 (2010).

[5] E. Candès and M. Wakin, "An introduction to compressive sampling", *IEEE Signal Processing Magazine,* **25**, 21 – 30 (2008).

[6] J. Romberg, "Imaging via compressive sampling", *IEEE Signal Processing Magazine,* **25**, 14 – 20 (2008).

[7] R. Baraniuk, "More Is less: Signal processing and the data deluge", *Science* **331**, 717 – 719 (2011).

[8] M.F. Duarte, S. Sarvotham, D. Baron, M. B. Wakin, and R.G. Baraniuk, "Distributed compressed sensing of jointly sparse signals", Proc. Asilomar Conf. Signals, Syst. Comput. 1537–1541 (2005).

[9] R. Torkamani, H. Zayyani, and R. Sadeghzadeh, "Model-based decentralized Bayesian algorithm for distributed compressed sensing", *Signal Processing: Image Communication,* **95**, (2021), 116212.

[10] J. Wright, Yi Ma, J. Mairal, G. Sapiro, T.S. Huang, and S. Yan, "Sparse Representation for Computer Vision and Pattern Recognition", *Proc. of the IEEE,* **98**, 1031 – 1044 (2010).

[11] M. Elad, Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing, Springer (2010).

[12] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang, "A survey of sparse representation: algorithms and applications", *IEEE access,* (2015).

[13] J. Mairal, M. Eldar, and G. Sapiro, "Sparse Representation for Color Image Restoration", *IEEE Trans. Image Proces.,* **17**, 53 – 69 (2008).

[14] W. Dong, L. Zhang, G. Shi, and X. Li, "Nonlocally Centralized Sparse Representation for Image Restoration", *IEEE Trans. Image Proces.,* **22**, 1620 – 1630 (2013).

[15] Zhenming Su, Simiao Zhu, Xin Lv, and Yi Wan, "Image restoration using structured sparse representation with a novel parametric data-adaptive transformation matrix", *Signal Processing: Image Communication,*, **52**, 151–172 (2017).

[16] J. Wright, A. Y. Yang, and A. Ganesh, "Robust Face Recognition via Sparse Representation", *IEEE Trans. Pattern Analysis and Machine Intelligence,* **31,** 210 – 227 (2009).

[17] XT. Yuan, X. Liu, and S. Yan, "Visual Classification With Multitask Joint Sparse Representation", *IEEE Trans. Image Proces.,* **21**, 4349 – 4360 (2012).

[18] D. Heinsohn, E. Villalobos, L. Prieto, and D. Mery "Face recognition in low-quality images using adaptive sparse representations", *Image and Vision Computing*, **85**, 46–58 (2019).

[19] K. Raja, R. Ramachandra, and C. Busch "Collaborative representation of blur invariant deep sparse features for periocular recognition from smartphones". *Image and Vision Computing*, **101**, 46–58 (2020).

[20] J. Yang, J. Wright, and T. Huang, "Image Super-Resolution via Sparse Representation," *IEEE Trans. Image Proces.,* **19**, 2861 – 2873 (2010).

[21] Y. Zhang, J. Liu, W. Yang, and Z. Guo, "Image Super-Resolution Based on Structure-Modulated Sparse Representation", *IEEE Trans. Image Proces.*, **9**, 2797 – 2810 (2015).

[22] Chaopeng Zhang, Weirong Liu, Jie Lui, Chaorong Liu, and Changhong Shi, "Sparse representation and adaptive mixed samples regression for single image super-resolution", *Signal Processing: Image Communication*, **67**, 79–89 (2018).

[23] Xuesong Li, Guo Cao, Youqiang Zhang, Ayesha Shafique, and Peng Fu, "Combining synthesis sparse with analysis sparse for single image super-resolution", *Signal Processing: Image Communication*, **83** 115805 (2020).

[24] C. F. Caiafa and A. Cichocki, "Computing sparse representations of multidimensional signals Using Kronecker Bases", *Neural computation*, **25,** 186 – 220 (2013).

[25] A. Cichocki, D. Mandic, L. De Lathauweri, G. Zhou, Q. Zhao, C. Caiafai, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis", *IEEE Signal Processing Magazine*, **32**, 145–163 (2015).

[26] Q. Dai, S. Yoo, and A. Kappeler "Sparse Representation-Based Multiple Frame Video Super-Resolution", *IEEE Trans. Image Proces.*, **26**, 2080 – 2095 (2017).

[27] H. S. Mousavi and V. Monga, "Sparsity-based Color Image Super Resolution via Exploiting Cross Channel Constraints", *IEEE Trans. Image Proces.,* **26**, 5094 – 5106 (2017).

[28] L. Rebollo-Neira and D. Whitehouse, "Sparse representation of 3D images for piecewise dimensionality reduction with high quality reconstruction", *Array*, **1**, `doi = 10.1016/j.array.2019.100001` (2019).

[29] L. Rebollo-Neira, R. Matiol, and S. Bibi, "Hierarchized block wise image approximation by greedy pursuit strategies," *IEEE Signal Process. Letters*, **20**, 1175–1178 (2013).

[30] L. Rebollo-Neira, " Effective sparse representation of X-Ray medical images", *International Journal for Numerical Methods in Biomedical Engineering*, in press (2017).

[31] L. Rebollo-Neira, "A competitive scheme for storing sparse representation of X-Ray medical images", *PLoS ONE*, 13(8): e0201455. `https://doi.org/10.1371/journal.pone.0201455` (2018).

[32] D. S. Taubman and M. W. Marcellin, JPEG2000 Image Compression, Fundamentals, Standards and Practice, Kluwer Academic Publishers (2002).

[33] G. Schaefer and M. Stich, "UCID: an uncompressed color image database", Proceedings Volume 5307, Storage and Retrieval Methods and Applications for Multimedia 2004; (2003) https://doi.org/10.1117/12.525375.

[34] `https:www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/` (Last access April 2022).

[35] E. Gershikov and M. Porat, "On color transforms and bit allocation for optimal subband image compression", *Signal Processing: Image Communication*, **22**, 1–18 (2007).

[36] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, Te-Won Lee, and T. J. Sejnowski, "Dictionary Learning Algorithms for Sparse Representation", *Neurocomputing*, **15**, 349–396 (2003).

[37] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation", *IEEE Trans. Signal Process.* **54**, 4311–4322 (2006).

[38] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double Sparsity: Learning Sparse Dictionaries for Sparse Signal Approximation", *IEEE Trans. Signal Process.* **58**, 1553–1564 (2010).

[39] I. Tošić and P. Frossard, "Dictionary Learning: What is the right representation for my signal?", *IEEE Signal Processesing Magazine*, **28**, 27–38 (2011).

[40] J. Zepeda, C. Guillemot, and E. Kijak, "Image Compression Using Sparse Representations and the Iteration-Tuned and Aligned Dictionary", *IEEE Journal of Selected Topics in Signal Processing,* **5**,1061–1073 (2011).

[41] S. Hawe, M. Seibert, and M. Kleinsteuber, "Separable dictionary learning", Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, 438–445 (2013).

[42] M. Srinivas, R. R. Naidu, C.S. Sastry, and C. Krishna Mohana, "Content based medical image retrieval using dictionary learning", *Neurocomputing*, **168**, 880–895 (2015).

[43] B. Wen, S. Ravishankar, and Y. Bresler, "'Structured Overcomplete Sparsifying Transform Learning with Convergence Guarantees and Applications", *International Journal of Computer Vision*, **114**, 137–167 (2015).

[44] L. Rebollo-Neira, "Cooperative greedy pursuit strategies for sparse signal representation by partitioning", *Signal Processing,* **125**, 365–375 (2016).

[45] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," *Proc. of the 27th ACSSC*,**1**,40–44 (1993).

[46] L. Rebollo-Neira, J. Bowley, A. Constantinides and A. Plastino, "Self contained encrypted image folding", *Physica A*, **391**, 5858–5870 (2012).

[47] L. Rebollo-Neira and D. Lowe, "Optimized orthogonal matching pursuit approach", *IEEE Signal Process. Letters*, **9**, 137–140 (2002).

[48] L. Rebollo-Neira, M. Rozložník, and P. Sasmal, "Analysis of the Self Projected Matching Pursuit Algorithm", *Journal of The Franklin Institute* 8980–8994 (2020).

[49] K. Skretting, `https://www.mathworks.com/matlabcentral/fileexchange/2818-huffman-coding-and-arithmetic-codin` (Last access April 2022).

[50] `http://www.nonlinear-approx.info/examples/node015.html` (Last access April 2022).

[51] `https://www.hlevkin.com/hlevkin/06testimages.htm` (Last access April 2022).

[52] `https://imagecompression.info/test_images` (Last access April 2022).

[53] `https://developers.google.com/speed/webp` (Last access April 2022).

[54] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, **13**, 600–612 (2004).

[55] R. Franzen, `http://r0k.us/graphics/kodak/` (Last access April 2022).