Image transform based on an alpha-beta convolution model

Antonio Alarcón-Paredes, Elías Ventura-Molina, Oleksiy Pogrebnyak, and Amadeo Argüelles-Cruz

Center for Computing Research, National Polytechnic Institute. México, D.F.
aparedesb07@sagitario.cic.ipn.mx, eventura_a12@sagitario.cic.ipn.mx, olek@cic.ipn.mx, jamadeo@cic.ipn.mx

Abstract. In this paper, it is presented a novel method based on an alpha-beta convolution model, to be used at transformation stage in an image compression system. This method takes the alpha-beta's associative memories theory and is applied to a set of images in grayscale. Since these associative memories are used for data with binary inputs and outputs, it is also presented a modification to the original alpha and beta operators in order to be applied directly to the the pixel values of an image. The proposed method is applied as a traditional convolution, with the difference that instead of making sums of products, there are performed maximum or minimum operations of the alphas and betas. The Shannon entropy is used to measure the amount of bits of information contained in the images. The traditional images transform usually do not provide any kind of compression and they also use complex operations. Therefore, this new method represents an advantage by offering a lower amount of entropy in the transformed image that in the original image by making use of simple operations such as addition, subtraction, minimums, and maximums.

1 Introduction

The imminent growth in the amount of the existing information has given the guideline to think on mathematical methods that help us to represent the information in a compact manner, reducing the number of bits used for its representation. For this reason, data compression systems and, in a particular case, image compression systems have been created. There exists lossless image compression systems, whose stages are: transformation and coding; as well as image compression systems, whose stages are: transformation, quantization and coding [5], [13]. Although there are a large number of image transforms, the most common one is the DCT (Discrete Cosine Transform). The DCT was proposed by Ahmed, Nataraj and Rao in 1974 [1]. It is a transform that is applied to blocks of pixels of an image, each block is usually constituted by 8 x 8 or 16 x 16 pixels, and it consists in a bijective function that maps one to one the image values allowing DCT to be reversible, thus there is no loss of information, but it does not compress the image neither [10], [17]. From the early 80s, the CCITT (Consultative

Advances in Computer Science and Engineering
Research in Computing Science 58, 2012 pp. 351–362
Paper Recived 28-09-2012 and Acepted 07-11-2012

Committee for International Telegraphy and Telephony) and ISO (International Organization for Standardization) began to work together in order to develop an international standard for image compression, which was achieved in 1992 with the acronym JPEG (Joint Photographic Experts Group) [9], [20] that uses the DCT in its transformation stage. From this date, diverse modifications have been developed to the DCT for its fast implementation, such as Chen [4], who takes advantage of the symmetry of the cosine function to reduce the needed operations to implement the transform. Subsequently, Arai [3] develops the DCT only taking into consideration the real part of a DFT (Discrete Fourier transform) and using the FFT (Fast Fourier Transform) algorithm that was proposed by Winogard in [22]. In addition there have been new developments for the DCT which can be consulted in [12], [15], [16] and [24]. In Shannon information theory [18], the quantity of information is defined as a probabilistic process, taking the image as the information source; denoted by S with n elements $s_1, s_2, \ldots, s_n$ and $i = 1, 2, \ldots n$, its entropy is defined as:

$$H(S) = -\sum_i P(s_i) \log_2 (P(s_i)) \tag{1}$$

for this reason, it is used the Shannon entropy as a measure to know the amount of bits which represent each image in this paper.

This paper is organized as follows: in section 2 it is shown the theoretical framework for the development of the new method, section 3 describes the proposed method. The results and conclusions are shown in section 4 and section 5, respectively.

2 Theoretical Support

This paper presents a new method based on a modification of the original algorithm of the alpha-beta associative memories [23], and focuses in the image transformation stage. This section shows the theory which is the base for the proposed method.

2.1 Associative memories

The AM (Associative memories) [8] are pattern recognition's algorithms, whose purpose is to recover full patterns from input patterns that could be altered. Input patterns are represented by column-vectors $\mathbf{x}$ and output patterns by column-vector $\mathbf{y}$. For each input pattern $\mathbf{x}$, there is one and only one output pattern $\mathbf{y}$, forming an association as an ordered pair: $(\mathbf{x}, \mathbf{y})$. The set of the p pattern associations is named fundamental association set or simply fundamental set, with $\mu = 1, 2, \ldots, p$, and it is represented as:

$$\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \,|\mu = 1, 2, \ldots p\} \tag{2}$$

The operation of an AM is divided into two phases: the learning phase, where input patterns $\mathbf{x}$ are associated with their corresponding output patterns $\mathbf{y}$ to

generate the associative memory; and the recovery phase, in which we introduce a pattern $\mathbf{x}$ as the input to the memory, and as result we expect to receive a corresponding pattern $\mathbf{y}$ in the outcome. There are two types of AM which are classified according to the nature of their pattern: auto-associative memories and hetero-associative memories. The memory is auto-associative if it fulfills that $\forall \mu, \mathbf{x}^{\mu} = \mathbf{y}^{\mu}$, *i.e.* each input pattern is equal to its corresponding output pattern; on the other hand, the memory will be hetero-associative if it is true that $\exists \mu \in \{1, 2, \ldots, p\} \, | \mathbf{x}^{\mu} \neq \mathbf{y}^{\mu}$, *i.e.* if there exists at least an input pattern different to its corresponding output pattern.

Traditional models of AM, such as Lernmatrix [19], Correlograph [21], as well as Linear Associator [2], [11] operate within the theory of artificial neuron model of McCulloch and Pitts [14]. That is,its operation is based on sums of products. In addition there are the morphological AM [6] and the alpha-beta AM [23] which are the only ones that are handled outside of this theory and instead of sums of products, they use maximums (or minimums) values of the sums, in the case of the morphological; and maximum (or minimum) of alphas and betas, in the case of the alpha-beta.

2.2 Convolution

On the other hand, in the theory of image processing, convolution is widely used. The convolution of an image is an operation in the spatial domain, *i.e.*, a method which operates directly on the value of the pixels in the image. To perform a convolution is required to define a mask, also called window, whose choice of values should be carefully taken; the mask is usually $3px \times 3px$. Convolution can be expressed as follows:

$$g\left(x, y\right) = T\left[f\left(x, y\right)\right] \tag{3}$$

where $f(x, y)$ is the input image, $g(x; y)$ is the output image and T is an operator in f that is defined on specific neighbors points to $(x; y)$; we will place the mask on each of these points to make the necessary operations. These operations consist of multiplying the pixels in the neighborhood (x, y) with the coefficients of the mask, adding the results to obtain the response in the pixel (x, y) of the resulting image [7].

Note that in both the classical AM models, and the convolution of images there are used sums of products; in the case of the alpha-beta AM, to its operation the sum of products is changed to maximums (or minimums) of alpha and betas. For the development of this work, it was thought to exchange the traditional function of convolution for a new method that is implemented as a convolution based on maximum or minimum of alfas and betas.

2.3 Alpha and beta operations

The alpha-beta AM [23] uses maximums and minimums and two binary operations proposed specifically: alpha (α) and beta (β), that establish the name

of alpha-beta AM. In order to make the definition of the binary operations alpha and beta, there should be specified the set A and the set B beforehand, as following:

$$A = \{0, 1\} \text{ and } B = \{0, 1, 2\} \tag{4}$$

Binary operation $\alpha : A \times A \to B$ is defined as:

Table 1 $\alpha : A \times A \to B$

x	y	$\alpha(x, y)$
0	0	1
0	1	0
1	0	2
1	1	1

Binary operation $\beta : B \times A \to A$ is defined as:

Table 2 $\beta : B \times A \to A$

x	y	$\beta(x, y)$
0	0	0
0	1	0
1	0	0
1	1	1
2	0	1
2	1	1

3 Proposed method

As can be shown, in *Table* 1 and *Table* 2, the alpha and beta operators could only handle binary inputs and outputs, hence it was required to extend them in order to handle real-valued numbers and thus operate directly over the pixel value of an image.

The new operation $\alpha_{\mathbb{R}} : \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$ is defined as

$$\alpha_{\mathbb{R}}(x, y) = x - y + 1 \tag{5}$$

The alpha-beta AM can be max type and min type; besides, the method proposed in this paper can operate both types: max or min. Thus, the operation $\beta_{\mathbb{R}} : \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$ has an operation for each type of recovery, max ($\beta^{\vee}_{\mathbb{R}}$) or min ($\beta^{\wedge}_{\mathbb{R}}$), as follows:

$$\text{If } x = y \longrightarrow \beta_{\mathbb{R}}(x, y) = 1 \tag{6}$$

$$\beta^{\vee}_{\mathbb{R}}(x, y) = y - |x| - 1 \tag{7}$$

$$\beta^{\wedge}_{\mathbb{R}}(x, y) = x - |y| - 1 \tag{8}$$

Definition 1. *Let* $\mathbf{A} = [a_{ij}]$ *be a matrix of size* $m \times n$ *representing an image, and let* $\mathbf{sb} = [sb_{ij}]$ *be a matrix* $d \times d$ *-dimensional and represents an* ***image sub-block*** *of* $\mathbf{A}$*, such as:*

$$sb_{ij} = a_{r_i t_j} \tag{9}$$

where $i, j = 1, 2, \ldots, d$*,* $r = 1, 2, \ldots, m$*,* $t = 1, 2, \ldots, n$*, and* $sb_{ij} = a_{r_i t_j}$ *represents the pixel value given by coordinates* $(r + i, t + j)$*, where* (r, t) *and* $(r + d, t + d)$ *are the beginning and the end of the sub-block, respectively.*

Definition 2. *Let* $\mathbf{A} = [a_{ij}]$ *be a matrix of size* $m \times n$ *that represents an image. The value denoted by* ε *is a value greater than the maximum value that could assume a pixel of an image, thus:*

$$\varepsilon > \bigvee_{\forall i,j} a_{ij} \tag{10}$$

Definition 3. *Let* $\mathbf{sb} = [sb_{ij}]$ *be a matrix* $d \times d$ *-dimensional and represents an image sub-block, and let* $\varepsilon > \bigvee_{\forall i,j} sb_{ij}$ *be a value greater than the maximum value that could assume a pixel of the sub-block* $\mathbf{sb}$*. The* ***transformation mask*** *of* max *type, denoted as* $\mathbf{mt}^{\vee} = \left[mt_{ij}^{\vee}\right]_{h \times h}$*, is initialized to* 0*, except for its central pixel that assumes the value* ε*, when the* max *type* $(\beta_{\mathbb{R}}^{\vee})$ *is used:*

$$mt_{ij}^{\vee} = \begin{cases} \varepsilon & \textit{central pixel} \\ 0 & \textit{other case} \end{cases} \tag{11}$$

Definition 4. *Let* $\mathbf{sb} = [sb_{ij}]$ *be a matrix* $d \times d$ *-dimensional and represents an image sub-block, and let* $\varepsilon > \bigvee_{\forall i,j} sb_{ij}$ *be a value greater than the maximum value that could assume a pixel of the sub-block* $\mathbf{sb}$*. The* ***transformation mask*** *of* min *type, denoted as* $\mathbf{mt}^{\wedge} = \left[mt_{ij}^{\wedge}\right]_{h \times h}$*, is initialized to* 0*, except for its central pixel that assumes the value* $-\varepsilon$*, when the* min *type* $(\beta_{\mathbb{R}}^{\wedge})$ *is used:*

$$mt_{ij}^{\wedge} = \begin{cases} -\varepsilon & \textit{central pixel} \\ 0 & \textit{other case} \end{cases} \tag{12}$$

By means of simplicity, in the following definitions, the notation of the sub-block $\mathbf{sb} = [sb_{ij}]$ and the transformation mask (either if is max or min) $\mathbf{mt} = [mt_{ij}]$ as coordinates, thus, $[sb_{ij}] = \mathbf{sb}(i, j)$ y $[mt_{ij}] = \mathbf{mt}\,(i, j)$.

Definition 5. *Let* $\mathbf{sb} = [sb_{ij}]$ *be a matrix of size* $d \times d$ *representing an image sub-block, let* $\mathbf{mt}^{\vee} = [mt_{rt}^{\vee}]_{h \times h}$ *be a transformation mask* max *type, and let* $\mathbf{t} = [t_{ij}]$ *be the transformed sub-block; the* ***alpha*** max ***convolution*** *operation* $\left(*\alpha_{\max}\left(\mathbf{sb}, \mathbf{mt}^{\vee}\right)\right)$*, is expressed as:*

$$\mathbf{t}\,(i, j) = *\alpha_{\max}\left(\mathbf{sb}, \mathbf{mt}^{\vee}\right) = \bigvee_{i=-a}^{a} \bigvee_{j=-b}^{b} \alpha_{\mathbb{R}}\left(\mathbf{sb}\,(i + r, j + t)\,, \mathbf{mt}^{\vee}\,(r, t)\right) \tag{13}$$

where $a = \frac{h-1}{2}$ *and* $b = \frac{h-1}{2}$*.*

Definition 6. *Let* $\mathbf{sb} = [sb_{ij}]$ *be a matrix of size* $d \times d$ *representing an image sub-block, let* $\mathbf{mt}^{\vee} = [mt^{\vee}_{rt}]_{h\times h}$ *be a transformation mask* max *type, and let* $\mathbf{t} = [t_{ij}]$ *be the transformed sub-block; the* ***beta*** min ***convolution*** *operation* $\left(*\beta_{\min}\left(\mathbf{sb}, \mathbf{mt}^{\vee}\right)\right)$*, is expressed as:*

$$\mathbf{t}\left(i,j\right) = *\beta_{\min}\left(\mathbf{sb}, \mathbf{mt}^{\vee}\right) = \bigwedge_{i=-a}^{a}\bigwedge_{j=-b}^{b} \beta^{\wedge}_{\mathbb{R}}\left(\mathbf{sb}\left(i+r, j+t\right), \mathbf{mt}^{\vee}\left(r,t\right)\right) \tag{14}$$

where $a = \frac{h-1}{2}$ *and* $b = \frac{h-1}{2}$*.*

Definition 7. *Let* $\mathbf{sb} = [sb_{ij}]$ *be a matrix of size* $d \times d$ *representing an image sub-block, let* $\mathbf{mt}^{\vee} = [mt^{\vee}_{rt}]_{h\times h}$ *be a transformation mask* max *type, and let* $\mathbf{t} = [t_{ij}]$ *be the transformed sub-block; the* ***alpha*** min ***convolution*** *operation* $\left(*\alpha_{\min}\left(\mathbf{sb}, \mathbf{mt}^{\wedge}\right)\right)$*, is expressed as:*

$$\mathbf{t}\left(i,j\right) = *\alpha_{\min}\left(\mathbf{sb}, \mathbf{mt}^{\wedge}\right) = \bigwedge_{i=-a}^{a}\bigwedge_{j=-b}^{b} \alpha_{\mathbb{R}}\left(\mathbf{sb}\left(i+r, j+t\right), \mathbf{mt}^{\wedge}\left(r,t\right)\right) \tag{15}$$

where $a = \frac{h-1}{2}$ *and* $b = \frac{h-1}{2}$*.*

Definition 8. *Let* $\mathbf{sb} = [sb_{ij}]$ *be a matrix of size* $d \times d$ *representing an image sub-block, let* $\mathbf{mt}^{\vee} = [mt^{\vee}_{rt}]_{h\times h}$ *be a transformation mask* max *type, and let* $\mathbf{t} = [t_{ij}]$ *be the transformed sub-block; the* ***beta*** max ***convolution*** *operation* $\left(*\beta_{\max}\left(\mathbf{sb}, \mathbf{mt}^{\wedge}\right)\right)$*, is expressed as:*

$$\mathbf{t}\left(i,j\right) = *\beta_{\max}\left(\mathbf{sb}, \mathbf{mt}^{\wedge}\right) = \bigvee_{i=-a}^{a}\bigvee_{j=-b}^{b} \beta^{\vee}_{\mathbb{R}}\left(\mathbf{sb}\left(i+r, j+t\right), \mathbf{mt}^{\wedge}\left(r,t\right)\right) \tag{16}$$

where $a = \frac{h-1}{2}$ *and* $b = \frac{h-1}{2}$*.*

Alpha-beta convolution transform algorithm

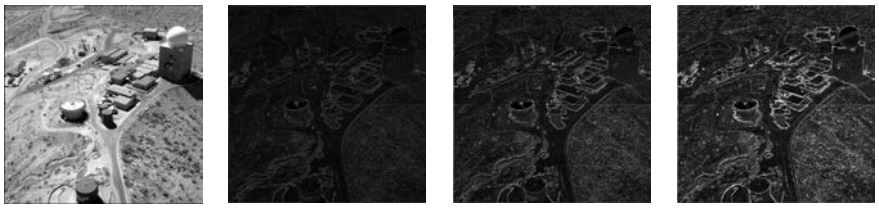
1. As usual on traditional image transforming methods, the alpha-beta convolution method proposed in here, is applied individually to an image sub-blocks of size $d \times d$. The image denoted as $\mathbf{A} = [a_{ij}]_{m\times n}$ is divided into $\kappa = (m/d) \cdot (n/d)$ sub-blocks $\mathbf{sb}^{\omega} | \omega = 1, 2, \ldots, \kappa$, where m and n are the image height and width respectively, a_{ij} is the $ij-th$ pixel of image with $a \in \{0, 1, 2, \ldots, L-1\}$ being L the number of bits that represents the value of a pixel.
2. Initialize to 0 all values in the resulting image $\mathbf{T} = [t_{ij}]_{m\times n}$.
3. Create the transforming mask **mt** depending on the desired usage: max or min, according to the *Definition 2*, *Definition 3* and *Definition 4*.
4. Apply the *alpha* max (or min) *convolution* to each sub-block according to the *Definition 5* and *Definition 7* and place the outcome on resulting image **T**. □

Inverse alpha-beta convolution transform algorithm

1. The inverse alpha-beta convolution method, is applied individually to the sub-blocks of transformed image **T**.
2. Initialize to 0 all values in the resulting recovered image $\mathbf{A}' = \left[a'_{ij}\right]_{m\times n}$.
3. Locate each sub-block of the transformed image **T** and apply the *beta* min (or max) *convolution* according to the *Definition 8* and *Definition 6*, using **the same transforming mask** used in the alpha-beta convolution transform algorithm and place the outcome on recovered image $\mathbf{A}'$. □

Since each time the **mt** is created, the maximum value of the current sub-block is taken, it is required to store every maximum in a vector.

4 Results



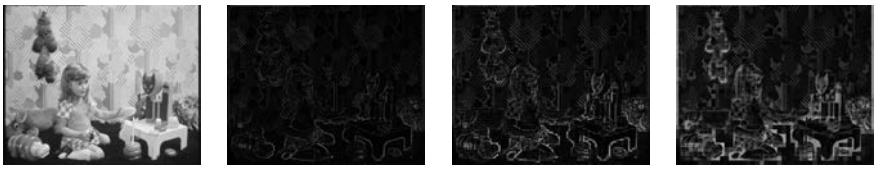
a) b) c) d)

Fig. 1 *Aerial* image (a), with sub-block different sizes: 4×4 (b), 8×8 (c), 16×16 (d).

In order to measure the proposed transform, a comparison between the original and the transformed image was performed using the Shannon entropy [18]. The method presented in this paper was applied to a set of 20 images widely used in many other papers, and was applied varying the size of **sb** (4×4, 8×8, 16×16) for each image, obtaining 60 transformed images. The set of testing images are grayscale, and 8 bits/pixel, and have different sizes.

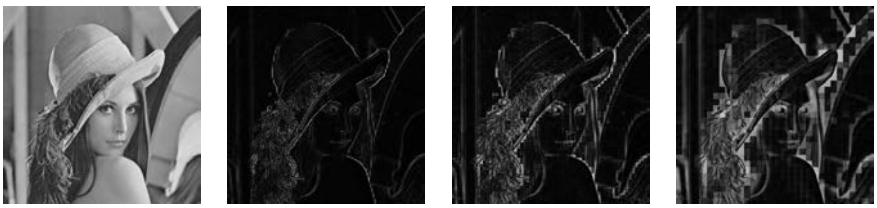
a) b) c) d)

Fig. 2 *Baloon* image (a), with sub-block different sizes: 4×4 (b), 8×8 (c), 16×16 (d).



a) b) c) d)

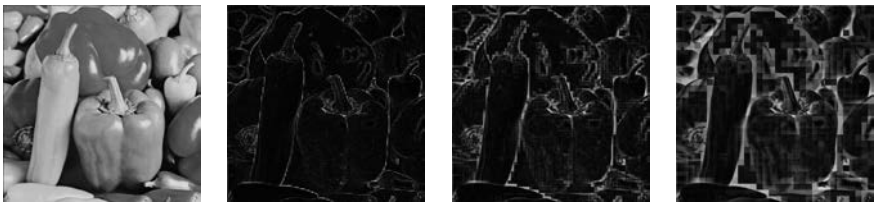
Fig. 3 *Girl* image (a), with sub-block different sizes: 4×4 (b), 8×8 (c), 16×16 (d).



a) b) c) d)

Fig. 4 *Lenna* image (a), with sub-block different sizes: 4×4 (b), 8×8 (c), 16×16 (d).

Figures 1 to 6 show the original image (a) and some transformed images using 4×4 sub-block size at (b), 8×8 sub-block size at (c), and 16×16 sub-block size at (d). The results that compare the entropy between the original image versus the transformed images are shown at *Table* 3.



a) b) c) d)

Fig. 5 *Peppers* image (a), with sub-block different sizes: 4×4 (b), 8×8 (c), 16×16 (d).

a) b) c) d)

Fig. 6 *Zelda* image (a), with sub-block different sizes: 4×4 (b), 8×8 (c), 16×16 (d).

Table 3. Entropy results with different **sb** sizes.

Image	Size	Entropy	Entropy 4×4	Entropy 8×8	Entropy 16×16
Aerial	2048×2048	7.1947	5.7846	6.4473	6.8813
Baboon	512×512	7.3577	6.4610	6.9275	7.1501
Baloon	720×576	7.3459	3.8442	4.7928	5.6843
Barbara	720×576	7.4838	5.7309	6.3724	6.8072
Bike	2048×2560	7.0219	5.3234	6.0152	6.5303
Board	720×576	6.8280	4.5892	5.3763	6.2891
Boats	512×512	7.1914	5.5520	6.2688	6.8104
Couple	512×512	7.2010	5.3092	6.0276	6.5776
Elaine	512×512	7.5060	5.1970	5.8841	6.5101
F-16	512×512	6.7043	4.8730	5.5588	6.0325
Girl	720×576	7.2878	5.0300	5.8262	6.4912
Goldhill	720×576	7.5300	5.3532	6.1223	6.6593
Hotel	720×576	7.5461	5.4036	6.2258	6.9116
Lenna	512×512	7.4474	5.1027	5.8879	6.5444
Man	512×512	7.1926	5.5016	6.3273	6.9066
Peppers	512×512	7.5943	5.1029	5.9029	6.662
Sailboat	512×512	7.4847	5.7920	6.5942	7.1399
Tiffany	512×512	6.6002	4.7339	5.3748	5.8841
Woman	2048×2560	7.2515	5.4033	6.0797	6.5413
Zelda	720×576	7.3335	4.5692	5.3577	6.1714

The data contained in the table refers to the image dimensions in pixels, entropy of the original image (Entropy), the entropy of the transformed image using a **mt** of 4×4 pixels (Entropy 4×4), the entropy of the transformed image using a **mt** of 8×8 pixels (Entropy 8×8), and the entropy of the transformed image using a **mt** of 16×16 pixels (Entropy 16×16). The transformation stage of an image compressor does not provide any information reduction, its main purpose is to make easier the compression at following steps (quantization and coding). Since the alpha-beta convolution transform provides information reduction, it is clear that represents an advantage over the traditional transforming methods, besides the method proposed in this paper uses very simple operations such as addition, substraction, minimums and maximums (comparisons).

5 Conclusions

The modified alpha and beta operators are capable to handle real valued inputs and may be used as well in image processing, and it is clear the next step could be the use of modified alpha-beta associative memories to perform the image transform.

By replacing the sums of products in traditional convolution with the maximums or minimums of alpha and beta, was possible to create an alternative for image transform in an image compression system, offering low computational cost by means of using simple operations.

The experimental results show that the alpha-beta convolution transform is reversible, that is, this new method has no information loss. Although 8 definitions were presented, it is needed the proposition and demonstration of some lemmas or theorems is required in order to formally prove this method is reversible.

As mentioned, the transformation stage in an image compression system does not provide any compression to images, so since the method proposed in this paper offers a smaller entropy on the transformed images, it also represents an advantage for the quantization and coding stages in an image compression system.

References

1. Ahmed, N., Natarajan, T., Rao, K. R.: Discrete cosine transform. IEEE Transactions on computers. **23** (1974) 90-93
2. Anderson, J. A.: A simple neural network generating an interactive memory, Mathematical Biosciences. **14** (1972) 197-220
3. Arai, Y., Agui, T., Nakajima, M.: A fast DCT-SQ scheme for image. Transactions of the IEICE. **71** (1988) 1095-1097
4. Chen, W., H., Smith, C., H., Fralick, S. C.: A fast computational algorithm for the discrete cosine transform. IEEE Transactions on Communications. **25** (1977) 1004-1009
5. Cosman, P., Gray, R., Olshen, R.: Handbook of medical imaging processing and analysis. Academic Press. USA. (2000) ISBN 0-12-077790-8
6. Díaz de León Santiago, J.L. & Yáñez Márquez, C.: Memorias Morfológicas Heteroasociativas. IT-57, Serie Verde, ISBN 970-18-6697-5, CIC-IPN, México (2001)
7. González, R. C., Woods, R. E. & Eddins, S. L.: Digital Image Processing using MATLAB. Prentice Hall. (2003). ISBN 0130085197
8. Hassoun, M. H.: Associative Neural Memories. Oxford University Press, New York. (1993)
9. ISO/IEC IS 10918-1 | CCITT T.81: Digital Compression and Coding of Continuous-Tone Still Image. (1992) ISO/IEC
10. Jain, A. K.: A sinusoidal family of unitary transforms. IEEE Transaction on Pattern Analysis and Machine Intelligence. **4** (1979) 356-365
11. Kohonen, T.: Correlation matrix memories, IEEE Transactions on Computers, C-21, **4** (1972) 353-359

12. Kok, C. W.: Fast algorithm for computing discrete cosine transform. IEEE Transactions on Signal Processing. **45** (1997) 757-760
13. Lakac, R., Plataniotis, K. N.: Color image processing methods and applications. CRC Press, Taylor & Francis. USA. (2007) ISBN 978-0-8493-9774-5
14. McCulloch, W. & Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics. **5** (1943) 115-133
15. Pan, W.: A fast 2-D DCT algorithm via distributed arithmetic optimization. IEEE Proceeding Image Processing. **3** (2000) 114-117
16. Ramírez, J., García, A., Frenadse, P. G., Parrilla, L., Lloris, A.: A new architecture to compute the discrete cosine transform using the quadratic residue number system. IEEE Int. Symposium on Circuits and Systems. (2000)
17. Rao, K. R., Yip, P., C.: The transform and data compression handbook. The Electrical Engineering and Signal Processing Series. (2001) ISBN 0-8493-3692-9
18. Shannon, C. E.: A mathematical theory of communication. Bell system technical journal. **27** (1948) 379-423 y 623-656
19. Steinbuch, K. V.: Die Lernmatrix. Kybernetik, **1**, **1** (1961) 36-45
20. Wallace, G. K.: The JPEG still picture compression standard. Proceedings of Communications of the ACM. **34** (1991) 30-44
21. Willshaw, D., Buneman, O. & Longuet-Higgins, H.: Non-holographic associative memory, Nature. **222** (1969) 960-962
22. Winograd, S.: On computing the discrete Fourier transform. Proceedings of the National Academy of Sciences of the United States of America. **73** (1976) 1005-1006
23. Yáñez, C.: Associative memories based on order relations and binary operators (in Spanish). Doctoral Theses. Center for Computing Research. (2002)
24. Yu, S., Swartzlander, E. E.: DCT implementation with distributed arithmetic. IEEE Transactions on Computers. **50** (2001) 985-991