A COMPLEXITY COMPARISON BETWEEN MULTILAYER PERCEPTRONS APPLIED TO ON-SENSOR IMAGE COMPRESSION

*José Gabriel R. C. Gomes, Sanjit K. Mitra**

University of California,
Santa Barbara, CA 93106 USA
E-mail: (gabriel,mitra)@ece.ucsb.edu

Rui J. P. de Figueiredo

University of California,
Irvine, CA 92697 USA
E-mail: rui@ece.uci.edu

ABSTRACT

A multilayer perceptron (MLP) can be used to implement a vector quantizer (VQ) under severe constraints in the computational complexity allowed. Such constraints are typical in applications such as focal-plane image compression, in which we are interested in eliminating the analog-to-digital (A/D) converters and mapping the analog data directly into a compressed bit stream, to save energy and silicon area. We compare a nonlinear MLP called Kernel Lattice Vector Quantizer (KLVQ) and a clustering MLP known as Cluster-Detection-and-Labeling (CDL) network, with regard to their hardware requirements. We show that for similar rate-distortion performances, the KLVQ has complexity smaller than that of the CDL network.

1. INTRODUCTION

Multilayer perceptrons (MLPs) can be used in image-sensing applications, to implement low-complexity compression schemes that map vectors of analog samples directly into binary codewords. This direct mapping saves hardware by completely eliminating the analog-to-digital (A/D) converters that are traditionally used for encoding each pixel value before digital image compression is done. Self-Organizing Maps (SOMs) [1] have been already implemented for analog-hardware data compression, because of their similarity with full-search vector quantization (VQ). However, the lack of structure in their codebooks results in a high complexity that prevents the use of SOMs for focal-plane image compression. Linear MLPs previously described [2] would be able to achieve compression performance similar to that of full-search VQ, but with less than half the computational complexity. We also showed in [3] how to exploit this large computational gap, by using *tanh(x)* activation functions and a direct design based on a *data lattice* and Kernel principal component analysis (Kernel PCA). This approach was called Kernel lattice vector quantization (KLVQ).

Another class of MLPs, called Cluster-Detection-and-Labeling (CDL) networks, has been proposed by Eltoft and de Figueiredo [4], to solve clustering problems very efficiently. If the number of clusters is made large enough, the CDL networks can also be applied to signal compression. In this paper, we study simple 2-D examples of KLVQ and CDL networks for a low-complexity image compression application, comparing them in terms of physical device (hardware) count and compression performance.

*This work was supported in part by CAPES/Brazil, by a University of California MICRO grant with matching support from Philips Research Laboratory, and in part by Microsoft Corporation.

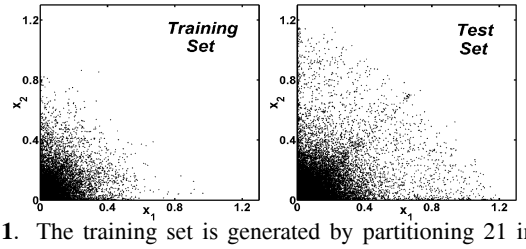

Fig. 1. The training set is generated by partitioning 21 images into 4×4 pixel blocks. The DC level of each block is encoded with 4-bit DPCM. An integer-value transform [5] is applied to the residual blocks, and only the first two components are kept. The test set, from different images, is 50% larger than the training set.

2. KLVQ REVIEW

Kernel PCA follows most of the ideas of PCA, but includes a mapping into a feature space of very high or infinite dimension, where the data covariance matrix is computed, and where separations of the data are supposed to be done with simpler surfaces [6], [7]. The problem is that the number of principal components in the feature space is as large as the number of data vectors available for covariance computation, so that a low-complexity hardware description of every eigenvector becomes impossible.

KLVQ [3] approaches this problem by reducing the amount of data vectors, in order to keep a minimal representation of the shape of the data density. Assuming the data to have Laplacian density (Fig. 1), we generate new data vectors that are regularly distributed over a pyramidal lattice. For example, the smallest lattice that can be used to represent a 2-D pyramid is the triangle with vertices $\mathbf{e}_0 = (0,0)$, $\mathbf{e}_1 = (1,0)$ and $\mathbf{e}_2 = (0,1)$. After warping with a Kernel $k(\mathbf{x}, \mathbf{e}_i) = tanh(g\mathbf{x}^T\mathbf{e}_i)$ (see Fig. 6), these vectors describe a subspace with 2 dimensions, embedded in a vector space of infinite dimension. The eigenvectors are described by a few (six) coefficients that may be represented with analog hardware [8]. The implementation can be summarized as follows. Given an input vector $\mathbf{x}$ from a data set (training or test) $\mathbf{X}$, we evaluate the vector of Kernels $k(\mathbf{x}, \mathbf{e}_i)$:

$$\mathbf{z} = tanh\left(g \begin{bmatrix} 0 \\ \mathbf{x} \end{bmatrix} + b\right) \tag{1}$$

Then we compute the projections over the principal components in the feature space, by means of a matrix-vector multiplication plus bias applied to $\mathbf{z}$:

$$\mathbf{y} = \mathbf{W}(g,b)\mathbf{z} + \mathbf{b}(g,b) \tag{2}$$

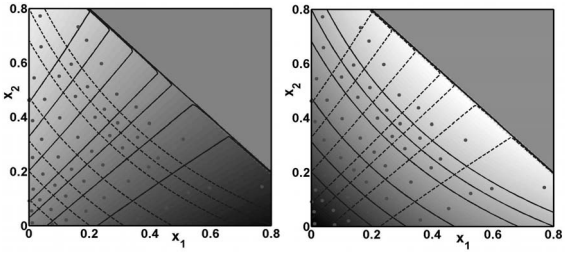

Fig. 2. A KLVQ designed using the training data set in Fig. 1. The thresholds assigned to y_1 are shown by solid contour lines in the left plot, and those assigned to y_2 are shown in the right plot. The reconstruction code-vectors are represented by the dark dots.

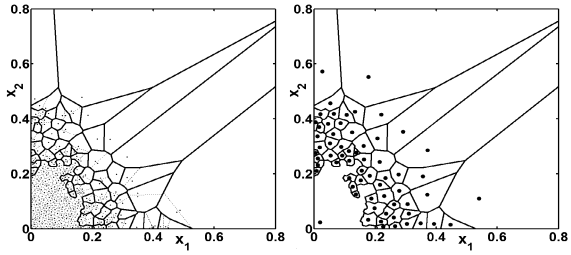

Fig. 3. A CDL network for VQ, designed using the training data set in Fig. 1. The prototype-cluster assignment is shown in the left plot. The right-hand side plot repeats the decision boundaries and shows the code-vectors at the gravity center of each partition.

The pair of parameters $(\mathbf{W}, \mathbf{b})$ is computed by Kernel PCA. All *input vectors* $\mathbf{x}$ are then given binary codes, obtained from scalar quantization of each component of $\mathbf{y}$. By scalar quantization of $\mathbf{y}$, we are actually performing VQ of $\mathbf{x}$. The thresholds of all scalar quantizers are stacked in a parameter vector $\mathbf{t}$. The centroids of the cells of vectors $\mathbf{x}$ with the same code can be computed and stored in a codebook $\mathbf{C}$, to be used as VQ reconstruction vectors $\mathbf{c}_j$. Given the training set $\mathbf{X}$, the pair (H, D) is a function only of $\mathbf{t}$, g and b, and these parameters can be chosen to minimize a cost function $J = D + \lambda H$. We choose values of gain and bias from a fine grid defined over a rectangle $g_1 < g < g_2$ and $b_1 < b < b_2$, and for each (g, b) the threshold vector is computed by a nonlinear unconstrained optimization routine (the Nelder-Mead simplex method, available in MATLAB as *fminsearch.m*), so that the mean-squared reconstruction error (MSE, equivalent to choosing $\lambda = 0$) is minimized. Plotting all $(H(g,b), D(g,b))$ points together and selecting only those on the lower convex hull is equivalent to performing optimization using different Lagrange multipliers. An example of 6-bit VQ designed from the training data in Fig. 1 is shown in Fig. 2. The dark dots indicate the vectors in the codebook.

3. CDL NETWORK REVIEW

The CDL network is a two-layer MLP proposed in [4] for clustering applications. The first layer has a complete description of a set of prototypes and thresholds that are used to decide on the membership of a given input $\mathbf{x}$ to a corresponding cluster. Each cluster c_i, $i = 1, ..., N$, is represented by a set of prototypes $\mathbf{p}_{ij}$, $j = 1, ..., P_{ij}$. We decide whether $\mathbf{x}$ is sufficiently close to one of the prototypes simply by comparing the Euclidean distance between them to a threshold:

$$d(\mathbf{x}, \mathbf{p}_{ij}) = (\mathbf{x} - \mathbf{p}_{ij})^T (\mathbf{x} - \mathbf{p}_{ij}) > \xi \qquad (3)$$

What makes the implementation possible in MLP format is the fact that this radial-basis computations can be implemented as inner-products for each neuron and comparison with variable thresholds:

$$t_{ij} = (\mathbf{p}_{ij}^T \mathbf{p}_{ij} - \xi + \mathbf{x}^T \mathbf{x})/2 \longrightarrow \mathbf{x}^T \mathbf{p}_{ij} < t_{ij} \qquad (4)$$

This similarity criterion can be physically implemented by a zero-level hard-limiting unit without a variable threshold ($f(x) = sgn(x)$), if we append $\mathbf{x}^T \mathbf{x}$ to the ij-th neuron inputs:

$$b_{ij} = (\xi - \mathbf{p}_{ij}^T \mathbf{p}_{ij})/2 \longrightarrow [\mathbf{x}^T \ \ \mathbf{x}^T \mathbf{x}] \begin{bmatrix} \mathbf{p}_{ij} \\ -0.5 \end{bmatrix} + b_{ij} < 0 \quad (5)$$

The second layer can operate in two different modes: training and classification (test). In training mode, we start with a neural network with only one prototype (which is the first input of the data set), and follow the three-stage training method explained in [4] (*CDL-NoMerging*, *CDL-Merging* and *Cluster Evaluation*). Depending on $\mathbf{p}$ and ξ, some inputs $\mathbf{x}$ may not be encoded during the training. The design parameters are shown in Table 1.

After the design is complete, we implement the output of the first layer using a winner-takes-all (WTA) operation, which has the same function of the MAXNET [7] and the similarity thresholds mentioned in the equations above are not used for evaluation. In Fig. 3, we show an example of a VQ that was designed following this method, together with tentative reconstruction code-vectors computed by averaging the CDL network partition of the training set $\mathbf{X}$. For the hardware implementation of the second layer in test mode, we will need a network that maps the minimum-distance winner decision into a binary codeword that describes the winner. It is illustrated in Fig. 4 for the case of 8 prototypes, each of them representing one cluster.

4. COMPLEXITY ESTIMATION EXAMPLE

We seek an expression for computational complexity in terms of the number of hardware units required for a mixed-signal implementation, and so we use a *CMOS transistor* as the complexity unit in this work, instead of counting multiplications, additions and comparisons per pixel. In order to make the transistor count, we make approximations based on neural network implementation details from [9] and [10]. We explain the method by means of an example showing the complexity evaluation of the systems in Figs. 2 and 3. The complexity offset due to color conversion (RGB to luminance), DPCM and linear transform coefficient computation is also included. It is the same for both systems. We do not count the transistors for optical sensing, since they are also present in conventional sensors that use A/D converters.

Table 1. Parameters for CDL Network design.

n_{min}	$n_{limit}^{(**)}$	ξ_L	α
1	4500	$2\xi_0^{(*)}$	0.9

(*) $\xi_0 = K \times d_{av}$, where $d_{av} = 0.083$ is the average distance between the input vectors $\mathbf{x}$ in Fig. 1. To design vector quantizers with 2, 4, 8, 16, 32, 64, and 128 clusters, we chose K = 1, 1, 2, 3, 8, 23, and 102.
(**) The training set has approximately 4800 vectors (taken from the database of 480000 pixel blocks in Fig. 1, decimated by 100), and at least 4500 should be classified, in training mode, before the training stops.

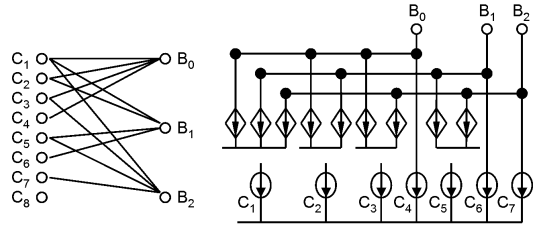

Fig. 4. The second layer of a CDL network maps a vector $\mathbf{e}_j$ (all zeros, and a "1" in the position j of the winning prototype) into a binary code for position j. The controlled current sources just copy the values of the current sources at their bottom, buffering the values so that currents can be appropriately added to generate the binary code. Some prototypes need more buffering than others.

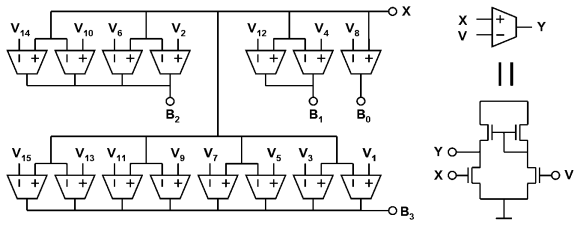

Fig. 5. Implementation of a B-bit scalar quantizer using $2^B - 1$ comparators. V_1 to V_{15} are voltage thresholds in ascending order.

4.1. Complexity Offset

The first step is color conversion from R, G and B sensors into luminance – we do not consider the compression of C_b and C_r data in this work. That can be thought of as an inner product $d_Y = \mathbf{v}\mathbf{s}^T$, with $\mathbf{v} = [0.299\ \ 0.587\ \ 0.114]$ and $\mathbf{s} = [s_R\ \ s_G\ \ s_B]$. An inner product of N-component vectors in which one of them is constant is assumed to take $N + 2$ transistors[1] in analog implementations of very low complexity and good precision [10]. So the operation above takes 5 transistors for each pixel, which means $\theta_1 = 80$ transistors over a block of 4×4 pixels. Computing $d_Y = \mathbf{v}\mathbf{s}^T$ for every pixel in the block results in a 16-component vector of luminance data ($\mathbf{d}_Y$), that we denote as $\mathbf{d}$ for simplicity. The next step is to find the difference with respect to the estimated mean (DC level) for DPCM: $\Delta\mu = \sum d_i/16 - \hat{\mu}$, which is an inner product of size 17 ($\theta_2 = 19$), and to quantize $\Delta\mu$ with 4 bits. A simple implementation of a 4-bit *flash scalar quantizer* requires 15 hard-limiters, as shown in Fig. 5, and a total of $\theta_3 = 60$ transistors. The quantized value $Q(\Delta\mu)$ can be obtained from the 15 outputs of the scalar quantizer by means of an inner product plus a bias, which is an inner product of size 16 ($\theta_4 = 18$). The updated estimate of the mean is obtained by scalar accumulation, i.e. $\hat{\mu}_{new} = \hat{\mu} + Q(\Delta\mu)$, which is an inner product of size 2, and requires at least the gate capacitance of one transistor for the delay memory, taking $\theta_5 = 5$ transistors. Finally, we compute a linear transform [5] of the mean-removed vector $\mathbf{v} = \mathbf{d} - \hat{\mu}\mathbf{u}$, where $\mathbf{u}$ is the column-vector whose components are all equal to one, and keep only the first two coefficients:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{H}\mathbf{v} = \mathbf{H}\mathbf{d} - \mu\mathbf{H}\mathbf{u} \qquad (6)$$

This corresponds to two inner products of size 16 plus bias (or two size-17 inner products). The complexity is therefore $\theta_6 = 38$ transistors. Adding all the estimates θ_j, $j = 1, ..., 6$, we obtain the total $\theta_{offset} = 220$ transistors, or 13.8 transistors/pixel.

4.2. Complexity of KLVQ

The computation of Eq. (1) requires 4 transistors per non-zero input operation (see Fig. 6), which takes $\tau_1 = 8$ transistors. The computation of $\mathbf{y} = W\mathbf{z} + \mathbf{b}$ (Eq. (2)) requires two inner products of vector of size 2, plus a bias (i.e. two inner products of size 3), so that $\tau_2 = 10$ transistors are required (as shown in Sec. 4.1). Most of the computational complexity of the KLVQ encoder lies on the scalar quantization of each component of $\mathbf{y}$, in this case y_1 and y_2. If we use three bits for representing y_1 using a flash A/D converter, that requires $4+2+1 = 7$ comparators, and the same for y_2, as shown in Fig. 5, with a total of 14 comparators. So the total amount of transistors for SQ of $\mathbf{y}$ is $\tau_3 = 56$. Adding $\tau_1 + \tau_2 + \tau_3$, we obtain 74 transistors, or 4.6 transistors/pixel. Including the 13.8 transistors/pixel offset (θ_{offset}), we obtain the overall complexity of this KLVQ example: $T_{KLVQ} = 18.4$ transistors/pixel. In Figs. 8 and 9, the KVLQ from Fig. 2 appears as the rightmost of the 3 points with complexity 18.4.

4.3. Complexity of CDL Network

The CDL network shown in the left part of Fig. 3 has 281 prototypes, but a computation of Euclidean distances is required only for the 164 prototypes that are closest to the decision boundaries. According to Eq. (5), 164 inner products of size 3 have to be implemented, so $\tau_1 = 820$ transistors. The computation of the norm-square $x_1^2 + x_2^2$ requires two transistors (quadratic distortion), so $\tau_2 = 2$. After all Euclidean distances are computed, the WTA operation can be implemented at a cost of 2 transistors per prototype, as shown in Fig. 7, so $\tau_3 = 328$ transistors are required at that stage.

The second layer has to do binary coding of the WTA results. As shown in Fig. 4, WTA outputs that have only one output connection do not require buffering, and WTA outputs that are associated with the [000000] codewords are not even connected to the output. Assuming 1 transistor per *current-mirror* buffer and 1 prototype per cluster, the total number of transistors in the second layer is

$$N \times C_{N,N} + (N-1) \times C_{N,N-1} + ... + 0 \times C_{N,1} + 0 \times C_{N,0} \quad (7)$$

where $C_{n,p} = n!/(p!(n-p)!)$. In the case of the 6-bit CDL network of Fig. 3, we have a summation of 64 terms represented as $\mathbf{c} = [6\ 5\ 5\ 5\ 5\ 5\ 5\ 4\ ...0\ 0\ 0]$, weighted by the number of prototypes per cluster, which is a vector with components sorted in

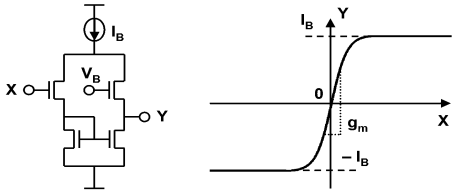

Fig. 6. Implementation of $f(x) = tanh(gx+b)$ with 4 transistors.

[1] An inner product of N-component analog vectors, where one of the vectors is constant, requires $N + M$ transistors [10]. M is the number of transistors in the current conveyor where the N transistor outputs are added in current mode. For our computational complexity analysis, we have assumed $M = 2$ as in [9].

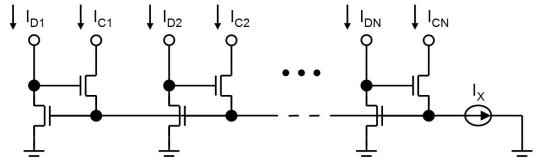

Fig. 7. WTA evaluation, using two transistors per prototype [9].

ascending order ($\mathbf{n} = [1\ 1\ 1\ 1\ ...5\ 5\ 7\ 155]$), so that more complicated clusters can be represented with less hardware. The weighted summation $\mathbf{cn}^T$ is $\tau_4 = 292$. Adding $\tau_1 + \tau_2 + \tau_3 + \tau_4$, we obtain 1442 transistors, or 90.1 transistors/pixel. Including the 13.8 transistors/pixel offset (θ_{offset}), we obtain the overall complexity of this CDL network example: $T_{CDL} = 103.9$ transistors/pixel. In Figs. 8 and 9, the CDL network from Fig. 3 appears with the 103.9 complexity.

5. SIMULATION RESULTS AND CONCLUSIONS

We designed KLVQs with bit allocations varying from $[0; 1]$ to $[4; 3]$. For design and test, we follow the description provided in Sec. 2 and choose values of gain varying from 0.005 to 10 and bias varying from -1.5 to 10. For each (g, b), distortion minimization was performed for all bit allocations and the (H, D) values of the best encoders were applied to a program for computation of the lower convex hull points. To design CDL networks with 1 to 7 bits of resolution, we followed the procedure mentioned in Sec. 3 and kept increasing K (Table 1) until the desired number of clusters was achieved. The KLVQs and CDL networks were tested for compression performance and the computational complexity was computed according to Sec. 4. Figures 8 and 9 compare the performance of KLVQ and CDL networks with respect to hardware requirements for compression applications only. It should be noted that CDL networks achieve a lower distortion at low entropy. However, KLVQ has lower complexity over all the range of entropies used in the simulations. In the CDL network, many prototypes are used to represent a few clusters (as shown in the left part of Fig. 3), which causes the first layer of the network to be large, a property that applies to the description of clusters with complicated shapes in classification problems.

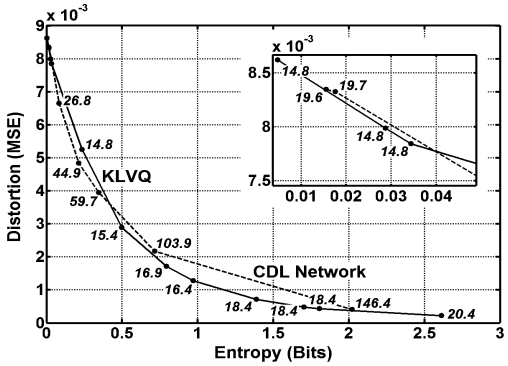

Fig. 8. Compression performance and complexity (training set).

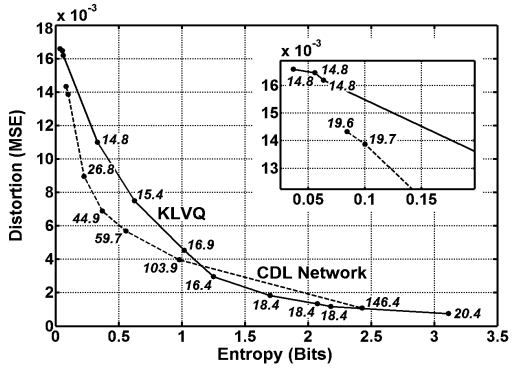

Fig. 9. Compression performance and complexity (test set).

6. REFERENCES

[1] T. Kohonen, *Self-Organizing Maps*, Springer-Verlag, Berlin; Heidelberg; New York.

[2] J. G. R. C. Gomes and S. K. Mitra, "Analog multilayer perceptron implementation of low complexity VQ for image compression," in *Proc. IEEE Int. Conf. Image Processing*, Barcelona, Spain, September 2003, pp. II.279–II.282.

[3] J. G. R. C. Gomes and S. K. Mitra, "Kernel PCA for quantization of analog vectors on a pyramid," in *Proc. IEEE Int. Workshop on Neural Networks for Signal Processing*, Toulouse, France, September 2003, pp. 579–606.

[4] T. Eltoft and R. J. P. deFigueiredo, "A new neural network for cluster-detection-and-labeling," *IEEE Trans. Neural Networks*, vol. 9, no. 5, pp. 1021–1035, September 1998.

[5] H. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization with 16-bit arithmetic for H.26L," in *Proc. IEEE Int. Conf. on Image Processing*, Rochester, NY, USA, Sep 2002, pp. II.489–II.492.

[6] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[7] R. J. P. de Figueiredo, "A new nonlinear functional analytic framework for modeling artificial neural networks," in *IEEE Int. Symp. on Circ. and Syst.*, New Orleans, LA, May 1990, pp. 723–726.

[8] G. Linán, A. Rodríguez-Vázquez, S. Espejo, and R. Domínguez-Castro, "ACE16K: A 128×128 focal plane analog processor with digital I/O," in *Proc. 7th IEEE Int. Workshop on Cellular Neural Networks and Their Applications*, 2002, pp. 132–139.

[9] A. G. Andreou, K. A. Boahen, P. O. Pouliquen, A. Pavasović, R. E. Jenkins, and K. Strohbehn, "Current-mode subthreshold MOS circuits for analog and VLSI neural systems," *IEEE Trans. Neural Networks*, vol. 2, no. 2, pp. 205–213, March 1991.

[10] G. Linan, *Diseno de Chips Programables de Senal Mixta con Bajo Consumo de Potencia para Sistemas de Vision en Tiempo Real*, Ph.D. Dissertation, University of Seville, Spain, June 2002.