# FPGA Implementation of a Support Vector Machine for Classification and Regression

Marta Ruiz-Llata, Guillermo Guarnizo and Mar Yébenes-Calvino

*Abstract*— **We present a successful design for a high-performance, low-resource-consuming hardware for Support Vector Classification and Support Vector Regression. The system has been implemented on a low cost FPGA device and exploits the advantages of parallel processing to compute the feed forward phase in support vector machines. In this paper we show that the same hardware can be used for classification problems and regression problems, and we show satisfactory results on an image recognition problem by SV multiclass classification and on a function estimation problem by SV regression.**

## I. Introduction

Pervasive computing systems are promoting emerging application in health care, entertainment, sensor networks, environmental monitoring, etc.. These applications require low size, low power and hence constrained resource processors, and, in most cases, real time operation.

Ubiquitous applications face decision making problems based on information from multiple sensors. In most cases a good analytical description of the system does not exist so it is necessary to model it from a set of empirical data. One of the challenges is to provide computing platforms that deal with both needs: constrained resources hardware and the ability of learning from examples.

This paper focuses on the development of a high efficient resource-consuming special purpose FPGA-based digital system for complex classification and regression problems. Neural processors for vision [1] and more general applications [2] have been implemented on FPGAs. These devices provide many advantages such as solid development tools, easy reprogrammability and fast development time without losing performance with respect full-custom systems design. Additionally, real parallel processing can be achieved, which is an advantage over other embedded platforms, as microcontrollers [3] and DSPs.

The hardware we propose has been designed based on Support Vector Machines (SVM) learning paradigm, which have a solid theoretical background and more clear formulation compared to neural networks [4][5]. Most significant contributions to this field, report FPGA implementation of SVMs for specific target applications [6][7][8] and problems dealing with relatively simple dataset and binary classification problems. However there are a few works focused into generic applications: In [9] ongoing research into a generic and versatile architecture for SVM classification is described. A co-processor that support both training and testing phases is presented in [10]. In [11] a detailed study of the effects of the reduction in precision of the SVM parameters and the use of fixed point arithmetic is reported.

The authors are with the Department of Electronic Technology, Universidad Carlos III de Madrid, Calle Butarque 15, 28911, Leganés, Madrid, Spain (e-mail: marta.ruiz-llata@ing.uc3m.es).

In this paper we propose a hardware based on FPGAs that supports both Support Vector (SV) Classification and Support Vector Regression. The structure of the paper is as follows. In the next section we describe the basics of SV classification and SV regression. In section 3 we justify the use of common hardware architectures for the feed forward phase of SV classification problems and SV regression problems. In section 4 we present its results for a multiclass image classification problem using the COIL database. In section 5 we present the firsts results for a real-valued function estimation problem by regression. Finally we present conclusions and further research in section 6.

## II. SV Classification and Regression

### *A. Classification*

SVM were originally developed for binary classification [4]. For a binary problem, given a set of $l$ data elements $\mathbf{x}_i$ and their corresponding class $y_i$: $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_l, y_l)\}$ where $\mathbf{x}_i \in R^n$ and $y_i = \pm 1$, the training step consists of resolving the following quadratic programming problem with linear restrictions [5].

$$\begin{aligned} &\underset{\alpha \in R^l}{\text{minimize}} \quad \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{l} \alpha_i \\ &\text{subject to} \quad 0 \le \alpha_i \le C \qquad \text{for } i = 1, \ldots, l \\ &\qquad\qquad\quad \sum_{i=1}^{l} \alpha_i y_i = 0 \end{aligned} \tag{1}$$

Where $K(x_i, x_j)$ is a kernel function and $C$ is a regularization constant that works as a constraint for the value of the Lagrange multipliers $\alpha_i$.

The feed-forward classification function of a new, non-learned, vector **x** is:

$$y(\mathbf{x}) = \operatorname{sgn}\left( \sum_{i=1}^{N_{SV}} y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b \right) \tag{2}$$

where parameters $\alpha_i$ and $b$ are given in the learning phase, most resulting equal to 0. Those training patterns whose corresponding $\alpha_i$ parameters are not equal to 0 are the *support vectors* (SV), and thus the summation in equation (2) extends only to the number of support vectors ($N_{SV}$). The learning process matches the model capacity to the data complexity providing high generalization ability and ensuring good performance on the future, previously unseen data. Scarcity is also an advantage for the design of an adaptive constrained resources specific hardware.

### B. Regression

In this case the problem consists on estimating a real-valued function. Given a set of $l$ data samples $\mathbf{x}_i$ and their corresponding known outputs $y_i$: $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_l, y_l)\}$ where $\mathbf{x}_i \in R^n$ and $y_i \in R$, assuming a ε-insensitive loss function proposed by Vapnik [4], the training step consists of resolving the following quadratic programming problem with linear restrictions:

$$\begin{aligned} \underset{\alpha,\alpha^* \in R^l}{\text{minimize}} \quad & \frac{1}{2}\sum_{i,j=1}^{l} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) K(x_i, x_j) \\ & + \varepsilon \sum_{i=1}^{l} (\alpha_i^* + \alpha_i) - \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) y_i \end{aligned} \tag{3}$$

$$\begin{aligned} \text{subject to} \quad & 0 \le \alpha_i^*, \alpha_i \le C \qquad \text{for } i = 1, \ldots, l \\ & \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) = 0 \end{aligned}$$

where $K(x_i, x_j)$ is a kernel function, $C$ is the regularization parameter and $\varepsilon$ is a positive parameter which defines the called *insensitive zone* inside of which the training errors are ignored. $C$ and $\varepsilon$ are predefined constants.

The feed-forward estimation function of a new, non-learned vector **x** is:

$$y(\mathbf{x}) = \sum_{i=1}^{N_{SV}} (\alpha_i^* - \alpha_i) K(\mathbf{x}_i, \mathbf{x}) + b \tag{4}$$

where parameters $\alpha_i$, $\alpha^*_i$ and $b$ are given in the learning phase. As in the classification model, only those resulting support vectors are used in the feed-forward phase.

### C. Parameter selection for hardware implementation

Typical Kernels are polynomial functions and Gaussian functions. Other kernels have been proposed in the literature. Our system uses the hardware-friendly kernel function proposed in [13]. This kernel greatly simplifies the SVM feed-forward phase computation in constrained hardware while maintains good classification performance with respect to the conventional Gaussian kernel [12][13]. Table 1 shows the examples of kernel functions.

TABLE 1
KERNEL FUNCTIONS

| Kernel | Description |
|---|---|
| Polynomial of degree $d$ | $[(\mathbf{x}^T \mathbf{x}_i) + 1]^d$ |
| Gaussian | $e^{\left(-\lVert \mathbf{x}_i - \mathbf{x} \rVert^2 / 2\sigma^2\right)}$ |
| Hardware-friendly | $2^{-\gamma \lVert \mathbf{x}_i - \mathbf{x} \rVert_1}$ |

Using the Hardware-Friendly kernel, the parameters to be fixed prior the training step are the $\gamma$ parameter of the kernel, which is made $\gamma = 2^p$, the regularization parameter $C$ and, in the case of SV regression, the $\varepsilon$ parameter. All these parameters are selected taking in mind that (2) and (4) will be solved in hardware using fixed-point arithmetic and low resolution (8 bits or 16 bits word lengths). Input attributes $\mathbf{x}_i$ and associated outputs $y_i$ are normalized to the range [-1,+1]. Parameter $\gamma = 2^p$ depends on the dimension of the sample data $\mathbf{x}_i$. The goal is to constrain to the word length size the result of the product of the $\gamma$ parameter by the higher L1-norm of a pair of sample patterns, that is, the maximum value of the exponent of the hardware-friendly kernel function. For example, if $\mathbf{x}_i \in R^2$ then $\gamma = 2^{-1}$ [12] and if $\mathbf{x}_i \in R^{1024}$ then $\gamma = 2^{-8}$. To determine $C$, and the $\varepsilon$ parameter in regression, we use an iterative training strategy with the goal of minimizing errors while keeping a reduced number of support vectors.

## III. HARDWARE PLATFORM

The basic hardware architecture to perform (2) and (4) is represented in Figure 1. It is composed of as many parallel SV_blocks as the resulting number of support vectors ($N_{SV}$), a controller, which is a state machine, and a summation block.

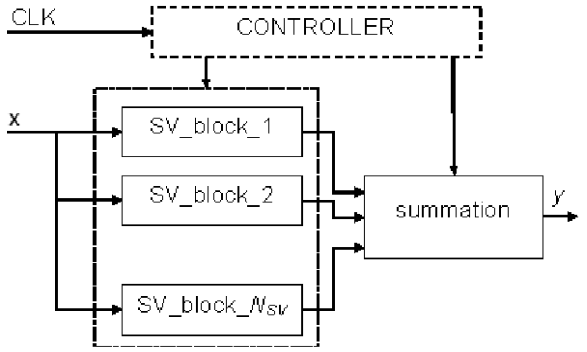

Fig. 1 Architecture for SV estimation function

As can be seen in Figure 1,each SV_block_i associated to support vector i, inputs vector **x** and outputs (5) in the case of SV classification or (6) in the case of SV regression. The summation block performs the addition of the results of all SV_blocks and outputs the estimated value of $y$. The highest bit is the sign bit which directly provides the class in the case of classification. In the case of regression, all bits are used to provide the normalized output.

$$y_i \alpha_i 2^{-\gamma \|\mathbf{x}_i - \mathbf{x}\|_1} \tag{5}$$

$$(\alpha_i^* - \alpha_i) 2^{-\gamma \|\mathbf{x}_i - \mathbf{x}\|_1} \tag{6}$$

In order to design SV_block_i, it can be easily shown that equations (5) and (6) can both be represented by (7).

$$sign_i \beta_i 2^{E_{1i}} \tag{7}$$

where $sign_i$ is a binary value that represents the class $y_i$ of the support vector $\mathbf{x}_i$ in the case of classification and the sign of $(\alpha_i^* - \alpha_i)$ in the case of regression. $\beta_i$ is the normalized value of $\alpha_i$ in the case of classification and $|\alpha_i^* - \alpha_i|$ in the case of regression, due to restrictions in (1) and (3) in both cases, the values of $\alpha_i$ and $|\alpha_i^* - \alpha_i|$ are in the range from 0 to $C$. Finally $E_{1i} = -\gamma \|\mathbf{x}_i - \mathbf{x}\|_1$.

The internal architecture of one SV_block_i for the calculation of (7) is represented in Figure 2. It is composed of: Memory blocks, identified as M(·), the E1_block, a CORDIC block and shift registers. Memory blocks store support vector $\mathbf{x}_i$, its corresponding normalized weight $\beta_i$ and its corresponding binary label $sign_i$.

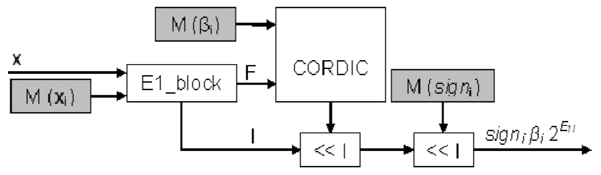

Fig. 2 Block diagram of the module that computes the contribution of each support vector

The E1_block computes the value $E_1 = -\gamma \|\mathbf{x}_i - \mathbf{x}\|_1$; it is designed taking into account the characteristics of the dataset. The input attributes of the input data are introduced sequentially, thus the final value of the L1-norm is obtained after a number of clock cycles equal to the number of input attributes, so if $\mathbf{x}_i \in R^2$ then the L1-norm is obtained in two clock cycles and if $\mathbf{x}_i \in R^{1024}$ then 1024 clock cycles are needed. E1_block provides two outputs, one associated with its fractional part (F) and the other with the integer part (I).

The CORDIC block computes $\alpha_i 2^F$ using a COordinate Rotation DIgital Computer (CORDIC) algorithm as described in reference [13]. CORDIC algorithms are a class of hardware-efficient algorithms that provide iterative solutions based on shifts and adds for the calculation of trigonometric and transcendental functions [14]. We use this algorithm to avoid the use of hardware multipliers in the computation of the feed forward phase of the SVM, and based on the use of this algorithm the computation is performed in as many clock cycles as the number of bits used to represent input parameters (8 or 16 in our implementations). The integer part of $E_1$ ($I$), is introduced in the final calculations by means of a shift register that shifts the result of the CORDIC block $I$ times as $\alpha_i 2^{E_1} = \alpha_i 2^{(F+I)} = \alpha_i 2^F 2^I$. Last block adds 1 sign bit according to the value of $sign_i$.

The implementation of the previous scheme represented in figure 2 was carried out on an Altera EP2C20 Cyclone II FPGA device. We have implemented two versions of the system, the first one uses 8 bits resolution for representing the following data: attributes $x_j$ of input vectors $\mathbf{x}=\{x_1, x_2,\ldots, x_n\}$ and support vectors $\mathbf{x}_i$, training $y_i$ and estimated outputs $y(\mathbf{x})$, associated weights $\beta$i and inputs and internal registers of the CORDIC block. All data and functional blocks of the system are stored in the FPGA device except inputs and support vectors. The clock rate of the system, limited by E1_block logic is 30 MHz. The 8-bit version of one SV_block_i (see figure 2) uses about 180 logic cells, which represent less than 1% of the selected low cost FPGA device. The second version, which uses 16-bit resolution, occupies less than 2% of the device. Taking into account the controller and the summation block (see figure 1) we can implement a Support Vector Machine for classification and regression that process up to 42 support vectors in parallel using 16-bit fixed point arithmetic or up to 86 using 8-bit arithmetic.

## IV. Hardware SV Classification

The performance of the system for a classification problem is tested on a simplified COIL database [15]. Our database is composed of 4 objects (4 classes). The samples of each object are 8 bits grey scale images with a resolution of 32×32 pixels. There are 72 different angular views of each object (see Figure 3). These images are used directly as inputs to the classifier, first converted to a vector of 32×32=1024 attributes as represented in figure 3.

In this section we face a multiclass classification problem. SVM were originally developed for binary classification [4]. Several methods have been proposed where a multiclass classifier is constructed by combining several binary classifiers. We have chosen a "one-against-one" training strategy, which means that assuming $q$ classes we need to train all possible pairs of classes, which results in $q(q\text{-}1)/2$ different SVMs to train. Compared to another typical strategy, such as "one-against-all", which trains $q$ binary classifiers, our approach provides a higher number of binary classifiers, but simpler, and thus fits better in constrained resources hardware in a modular way. The classification strategy in the feed-forward phase is as in a knockout tournament [15], hence *q*-1 classifiers are evaluated by the hardware SVM.

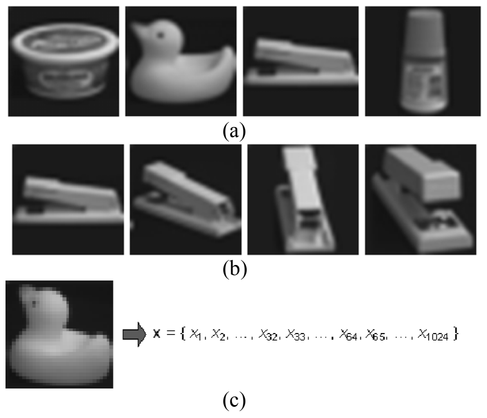

Fig. 3 Implementation of a classifier of images (a) One sample of each class (32×32 8-bit gray scale images). (b) Different samples of one class. (c) High dimensional vector representing one sample: first row of pixels are attributes 1 to 32, second row are attributes 33 to 64 and so on up to 1024 attributes.

As there are 4 classes (see Figure 3.a), 6 binary classifiers are trained using MATLAB. We used an iterative training strategy with the goal of minimizing classification errors while keeping a reduced number of support vectors. The training parameters are $\gamma = 2^{-8}$, $C = 1$ and the number of support vectors is limited to 10 per binary classifier.

In the hardware we used 8 bits per attribute to store each of the 10 resulting support vectors, 8 bits to store its corresponding normalized weight $\beta_i$ and an additional bit for its binary label $sign_i$. The 1024 input attributes of the input data are introduced sequentially, thus the final value of the L1-norm is obtained after 1024 clock cycles in an 18 bit accumulator register. Less significant 8 bits are discarded as $\gamma=2^{-8}$, the following 8 bits correspond to the fractional part of $E_1$ (F) and another two more significant bits of the accumulator register correspond to the integer part of $E_1$ (I).

The whole system, including 6 binary image classifiers, fits in 75% of the logic elements of the FPGA EP2C20 Cyclone II selected device and a 64Kbytes external memory to store all support vectors. The clock speed of the system is 30 MHz and the classification speed of the system, limited by the reading of the external memory, is 2 ms. Figure 4 shows the error rate of each of the individual binary classifiers obtained by HW simulation. The error rate is 0% for three of them and below 3.3% for the other three cases. The error rate of the 4-classes classifier, following the strategy previously described is 4%. Lower error rate can be achieved increasing the number of support vectors and increasing resolution. 0% error rate is obtained when using floating point arithmetic and up to 30-40 support vectors per binary classifier [15].

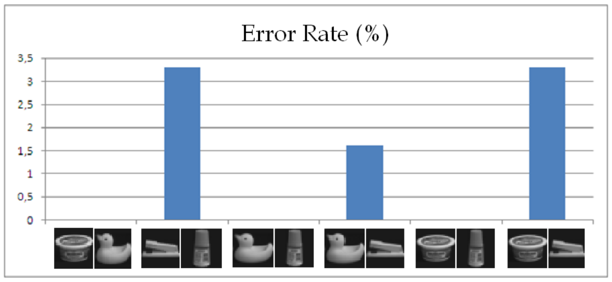

Fig. 4 Error rate of each one-against-one hardware classifier.

## V. Hardware SV Regression

In this section we present preliminary results of the hardware system for its use in SV regression problems. In this case the goal is to estimate the *sinc* function for two dimensions:

$$y = \frac{\sin\sqrt{x_1^2 + x_2^2}}{\sqrt{x_1^2 + x_2^2}} \qquad (8)$$

The samples database (figure 5) is generated by randomly choosing 400 input $(x_1, x_2)$ values, being $x_i \in (-10,10)$, and calculating its corresponding output value *y*. This is a valid example for testing the function estimation problem [5].

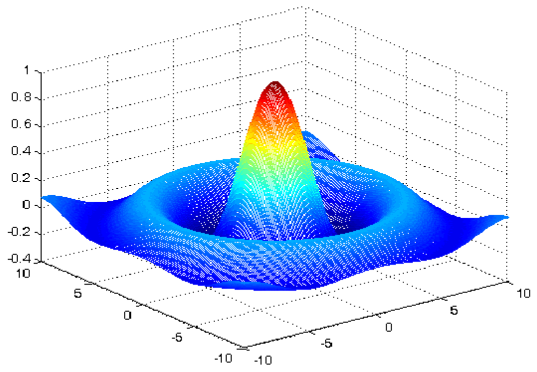

Fig. 5. Sinc Function.

The process of training and subsequent testing of data was performed using MATLAB [16]. The first step is to find, for our training data, the optimum values for the regularization parameter *C*, the parameter which defines the *insensitive zone ε* and the parameter for hardware friendly kernel γ. After this training, we can verify the effectiveness of the machine by testing it with new unknown data and comparing these results with the real values for *sinc* function. The testing data are 100 values of $(x_1, x_2)$ also randomly chosen in the same range as sample data.

Figure 6 shows the firsts results obtained using the hardware friendly kernel function described in Table 1 as the kernel function [13] with a value of $\gamma = 2^{-2}$. The other training parameters are fixed to $C$=1 and $\varepsilon$=0.01. Figure 6 shows the support vectors that are obtained after the training as marked by circles. The number of support vectors is 283. This figure also shows the estimated outputs for the test database. The average error between the estimated output value and the real value is 0.02, which means that the SVM performs quite well when using the hardware friendly kernel.

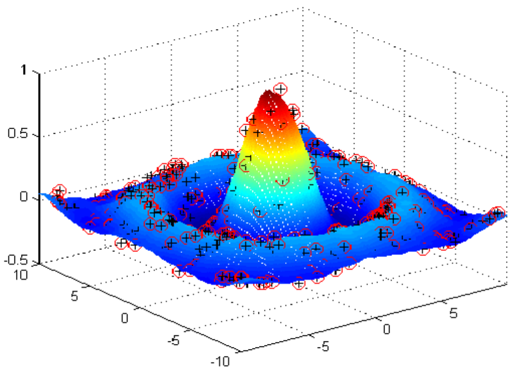

Fig. 6 Results of training with the Hardware-Friendly Kernel with $\gamma$ = 0.25, $C$ = 1 and $\varepsilon$ = 0.01. The number of support vectors obtained is 283/400.

## VI. CONCLUSION

In this paper we describe a hardware implementation for support vector classification and regression of data. The architecture has been designed to be implemented in FPGAs in order to achieve parallel processing. It has been conceived as a general purpose architecture for embedded applications, where the number of support vectors and the resolution of the parameters can be configured. Depending on the problem there will be different tradeoffs between classification or regression estimation accuracy, processing speed and target FPGA density, concerning cost or power consumption. Additionally, there is not a limit to the dimension of the input vectors (attributes) and the number of support vectors that can be processed in parallel is only limited by the size of the FPGA.

In most embedded applications, especially those requiring a large number of high dimensional SVs (as the example described in section IV), it is necessary to use external memory because the internal memory of the device is not large enough to store weights, labels and support vectors. We assume this is a typical situation so memory management is one of our main concerns for future research. We are investigating techniques to improve the throughput of the memory interface and methods to improve the capacity of the memory, especially in the case of multiclass classification where different binary classifiers share some support vectors.

## REFERENCES

[1] F. Yang, M. Paindavoine. "Implementation of an RBF Neural Network on Embedded Systems: Real-Time Face Tracking and Identity Verification". IEEE Transactions on Neural Networks 14(5), 1162-1175 (2003).

[2] P. Lee, E. Costa, S. McBader, L. Clementel, and A. Sartori. "LogTOTEM: A Logarithmic Neural Processor and its Implementation on an FPGA Fabric", IJCNN 2007, Orlando, FL.

[3] A. Boni, F. Pianegiani, and D. Petri "Low-power and low-cost implementation of SVMs for smart sensors" IEEE Transactions on Instrumentation and Measurement, vol. 56, no. 1, pp.39-44, Feb. 2007.

[4] V. N.Vapnik : *The Nature of Statistical Learning Theory*, 2nd ed. New York: Springer-Verlag, 1995, ch. 5-6.

[5] B. Schölkopf and A. Smola, *Learning With Kernels*. Cambridge, MA: MIT Press, 2002.

[6] D. Anguita, A. Boni, and S. Ridella, "A digital architecture for support vector machines: Theory, algorithm and FPGA implementation", IEEE Trans. Neural Networks, vol. 14, no. 5, pp. 993-1009, Sep. 2003.

[7] W-Y. Choi, D. Ahn, S.B. Pan, K. Il Chung, Y. Chung, S-H. Chung. "SVM-Based Speaker Verification System for Match-on-Card and Its Hardware Implementation". ETRI Journal, Volume 28, Number 3, June 2006.

[8] J. Manikandan, B. Venkataramani, V. Avanthi, "FPGA Implementation of Support Vector Machine Based Isolated Digit Recognition System,", 2009 22nd International Conference on VLSI Design, pp.347-352, 2009.

[9] C. Kyrkou, T. Theocharides. "SCoPE: Towards a Systolic Array for SVM Object Detection". IEEE Embedded Sytems Letters, Vol. 1, pp. 46-49, 2009

[10] S. Cadambi, I. Durvanovic, V. Jakkula, M. Sankaradass, E. Cosatto, S. Chakradhar, H.P. Graf. 'A Massively Parallel FPGA-based Coprocessor for Support Vector Machines'. Proc. Of 17th IEEE Symposium on Field Programmable Custom Computing Machines, 2009, pp. 115-122

[11] D. Anguita, A. Ghio, S. Pischiutta and S. Ridella "A support vector machine with integer parameters". Neurocomputing, 2008, Vol. 72, pp. 480-489.

[12] M. Ruiz-Llata, M. Yébenes-Calvino."FPGA Implementation of Support Vector Machines for 3D Object Identification", in ICANN 2009, Part I, LNCS 5768, pp. 467–474, 2009.

[13] D. Anguita, S. Pischiutta, S. Ridella and D. Sterpi, "Feed-Forward Support Vector Machine Without Multipliers," IEEE Trans. Neural Netw.,vol. 17, no.5, pp. 1328-1331, Sep. 2006.

[14] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," *in Proc. ACM/SIGDA 6th Int. Symp.Field Programmable Gate Array,* Monterey, CA, 1998, pp. 191-200.

[15] M. Pontil, A. Verri. "Support Vector Machines for 3D Object Recognition". IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (6), 637-646 (1998).

[16] S. R. Gunn. *Support Vector Machines for Classification and Regression. Technical Report.* Image Speech and Intelligent Systems Research Group, University of Southampton, 1997.