Accelerometer-Based Hand Gesture Recognition System for Interaction in Digital TV

J. Ducloux[1,2,3], P. Colla[2], P. Petrashin[1], W. Lancioni[1], L. Toledo[1]

[1]Facultad de Ingeniería, Universidad Católica de Córdoba

[2]Especialización en Sistemas Embebidos, Instituto Universitario Aeronáutico

[3]Consorcio Córdoba TDT

Córdoba, Argentina

***Abstract*—This paper presents the design and implementation of a system of accelerometer-based hand gesture recognition. This system will be embedded within a modern remote control to improve human-machine interaction in the context of digital TV of Argentina. As the recognition of hand gestures is a pattern classification problem, two techniques based on artificial neural networks are explored: multilayer perceptron and support vector machine. This is performed in order to compare results and select the tool that best fits the problem. Jointly, signal digital processing techniques are used for preprocessing and adapting of the input signals to pattern recognition models. A gestural vocabulary of 8 types of gestures was used, which was also used by other similar works in order to compare results. An appropriate trade-off between the classifier recognition precision and resource utilization of the hardware platform is required in order to implement the solution within an embedded system. The obtained results of precision and utilization of resources are excellent.**

Keywords—accelerometer, artificial neural networks (ANNs), digital TV, embedded systems, hand gesture recognition, multilayer perceptron (MLP), remote control, support vector machine (SVM)

I. INTRODUCTION

Gesture recognition is the process of understanding and classifying the significant changes performed by the hands, arms, face and sometimes the heads of people. This has become a very attractive area of research for the design of man-machine interfaces equipped with artificial intelligence for many applications, such as sign language, disability, home automation, virtual reality, etc. [1].

Gesture recognition is a research area that is booming for both vision-based method and inertial sensors-based method of micro-electromechanical systems (MEMS) technology. The use of MEMS-technology is quite attractive since it includes low-cost sensors that do not suffer from major influences as vision-based recognition systems, such as ambient light levels and background type. Most of the available literature for MEMS-based hand gesture recognition only uses 3-axis accelerometer, with good results in the precision obtained.

Most of the techniques and algorithms used in systems of accelerometers-based hand gesture recognition on current literature [2]-[6] have been implemented in systems of large resources, such as notebooks or desktop computers, with fast processors and enough memories. The lightest algorithms were implemented in smartphones, but code optimizations must be carried out in order to reduce the computational load and resource utilization [7].

In this work, the hand movements in free space describing some previously defined shape by manipulating a device interaction are referred as gestures. In this case, the interaction device is a remote control. The gestures are represented by vectors containing the variations in the levels of acceleration versus time in three-dimensional space. Time series as acceleration signals are equally valid terms in order to describe gestures.

Within the context of digital TV, television systems have experienced huge improvements in recent times. The emergence of terrestrial digital TV and smart TVs allows to incorporate the concept of interactivity which together with the Internet services are producing an innovative impact and a consequent change in the user experience. The control of these devices is in most cases even solved by the traditional infrared remote control, which has become a limiting factor in the user interaction with the TV. Consequently, different types of interfaces and new control methods should be included in order to improve the user experience immersed in this technological evolution. Some of the renowned manufacturers of TV and entertainment devices began to incorporate novel user interfaces in their top-end products, such as voice control and image control, although the vast majority of remote controls used in TV is still offering basic functionality.

The gestures, particularly the hand gestures, have two aspects in their signal characteristics that make them difficult for recognition. Firstly, they present segmentation ambiguity. Secondly, they have temporal and spatial variability even for the same gestures and the same people.

Models such as the multilayer perceptron (MLP) [8] and the support vector machine (SVM) [9] are types of artificial neural networks (ANNs) that attempt to reproduce the problem-solving process of the brain [10]. As humans apply knowledge gained from experience to new problems or situations, a neural network takes solved problems in order to build a system that makes decisions and performs classifications. These techniques are widely used in robotics, medicine, speech recognition and data mining, to cite some

This paper is presented as a preliminary report for FSTICS Project N° 4, funded by the National Agency for Promotion of Science and Technology through FONARSEC, which has as Beneficiary Institution the Catholic University of Córdoba, and as the Adopter the TDT Córdoba Consortium also located in the same city.

areas of application. Problems suitable for neural solution are those that do not have an accurate computational solution, or that require very extensive algorithms in order to implement the solution. Thus, these techniques are suitable for applications in hand gesture recognition.

In this paper, a system of hand gesture recognition using ANNs is designed and implemented, in order to be embedded within a modern remote control and be used in the context of digital TV in Argentina. The proposed system is able to recognize isolated gestures, be independent of the user, work with a predefined gestural vocabulary of 8 classes, and exhibit excellent recognition rate. Recognized gestures are translated into control commands that will execute various actions on home digital TV systems. The gesture recognition system will be implemented in a microcontroller-based embedded system. The hardware platform is selected in order to obtain reduced execution times of the classifier algorithms for a real-time response, and to achieve a suitable recognition rate for a better user-experience. A comparative analysis of two types of ANNs is carried out in order to find the model that provides the better relationship between precision versus execution time. To train, validate and evaluate the models, an existing database is used.

The document is diagrammed as follows. In Section II are commented the composition and analysis of the used database, and the methodology in order to train, validate and evaluate ANNs models. In Section III are established the design strategy and the comparison parameters of ANNs models. The design, implementation and testing of the classification system are presented in Section IV. The results of precision and execution times of the algorithms are shown and discussed in Section V. Finally, the conclusions and future work are commented at the end of the article.

II. Database for Training, Validation and Evaluation of ANNs Models

The database employed here is the database of the uWave project [11], [12]. This project is pioneer in accelerometer-based hand gesture recognition using dynamic time warping (DTW). Gestural vocabulary of database includes 8 types of hand gestures, as shown in Fig. 1.

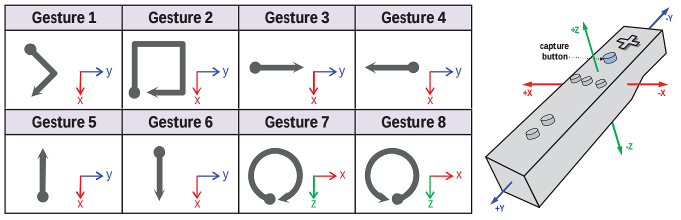

Fig. 1. Gesture vocabulary.

The database includes 4480 gestures, performed by 8 users, for 7 days and 10 repetitions per day. Thus, the database contains the 8 types of gestures in proportionate quantities.

The pattern recognition techniques used in this work are two types of ANNs: MLP and SVM. These models require training data in order to learn and store knowledge which will be used for performing the classification task. To do this is needed a database for training, validation and evaluation in order to select the best model with optimal behavior to resolve the problem. The database was analyzed and processed through the R language [13] using the RStudio IDE [14].

A. Analysis of the Database

Useful information for designing was extracted from the database. Analysis of missing data and outliers treatment were performed. The duration of the gestures in number of samples was obtained. This information is used to set the minimum and maximum duration of the gestures and to determine the size of the input buffers. Minimum and maximum values of acceleration levels of the gestures are useful to select the acceleration sensor according to the dynamic range of operation and to normalize the input. The discrete Fourier transform was applied for obtaining the bandwidth of the acceleration signals. A value of 7 Hz was obtained. This information is useful for selecting the sampling frequency of embedded system and cutoff frequency of the input filter.

B. Partition of the Database

The database was randomly divided into two representative groups of all observations. Thus, the percentage of each class was conserved. The 80% of the data was used for training and validation of the models. The remaining 20% was used for evaluation of the models. The leave one-out k-fold technique of cross-validation was implemented. The training and validation partition was divided into k=20 groups of data, using k-1=19 groups for training and 1 group for validating the model. This process was repeated k=20 times, using all data for validating the model in question. Thus, 20 validated models were obtained. The evaluation data were used for evaluating the best validated model in order to obtain the final recognition precision. The process steps can be seen in Fig. 2.

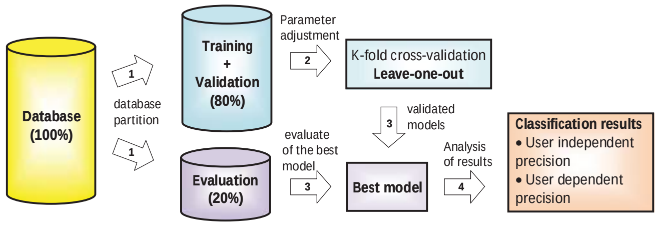

Fig. 2. Process for training, validation and evaluation of models.

For user-independent case, the entire data database is used. For the user-dependent case, the data of a particular user are used for model training, validation and evaluation.

III. Parameters for Design and Comparison of ANNs Models

The recognition rate is used as a parameter for comparison between MLP and SVM with linear kernel models, previously setting similar execution times for both classification algorithms. The execution times depend on the number of arithmetic operations as multiplications and sums involved in the algorithms. In both models, the number of inputs and outputs are fixed. Thus, for the MLP case, the execution time can be varied according to the number of hidden layer neurons. The number of neurons in the output layer is equal to 8, corresponding to the classes that represent gestural vocabulary. In the case of the SVM with linear kernel, 28 binary classifiers should be implemented in order to work with multiclass classification based on one-versus-one strategy [15]. An SVM with polynomial kernel of degree equal to 2 and one-versus-one multiclass strategy is implemented only for comparing recognition rate results.

The resource usage for the implementation of an SVM with linear kernel is considered as a design reference, which depends directly on the number of inputs and outputs of the system. Accordingly, the MLP architecture is obtained equaling the resource usage of the SVM with linear kernel, adjusting the number of neurons in the hidden layer.

IV. Design, Implementation and Test of the Classification System

To carry out this work, software engineering practices were used since it involves the creation of software as a fundamental part of the system. Analysis, definition and review of requirements were performed. The validated requirements are used later in order to verify the correct system operation. The analysis stage included the extraction of information from the database in order to learn details of the problem. During the design, architecture and interfaces of the system were defined. In addition, a plan for integration tests of the different modules that compose the classifier was established. In the implementation phase, the different software modules were coded and tested unitarily. Finally, the tests of integration and system that were planned in the previous phases are performed. The adoption of the above described methodology allowed decreasing the injection of errors into the software, reducing rework and obtaining a system with a certain degree of quality.

Firstly, the type of hardware platform in order to implement the system was defined. The microcontroller-based platforms were selected, due to a combination of features that are presented as advantages over other platforms for this particular application. Some features are low-cost, low energy consumption, high operating frequency, reuse and portability of code, peripherals such as I2C, SPI, UART, USB, incorporating hardware modules for digital signal processing and floating-point arithmetic. In addition, this platform provides different ways in order to update the firmware.

A. Design

Different configurations for the classifier were analyzed. The best results were obtained by the system whose architecture is shown in Fig. 3.

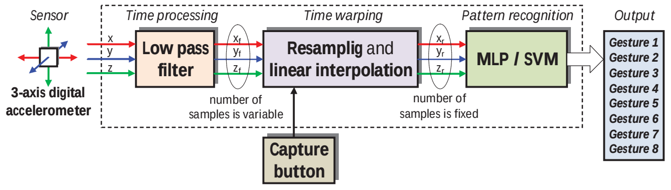

Fig. 3. Blocks diagram of the system of hand gesture recognition.

The system of hand gesture recognition has as inputs the acceleration signals and the capture button. The acquisition of the acceleration vectors for a given gesture starts when the capture button on the remote control is pressed and ends when the button is released. In this way, the samples of the 3 axes that make a gesture are obtained. The system has a single output corresponding to the recognized gesture from the captured samples, which will be converted into a control command for digital TV systems.

Before applying the classification algorithms of MLP and SVM, the input data must be preprocessed. The data provided by the accelerometer are processed by a finite impulse response (FIR) low-pass digital filter of linear phase, optimal and equiripple. This filter is used to attenuate any high frequency component as the associated with hand vibrations and noise. The filter was designed using the Parks-McClellan algorithm, in order to have a cutoff frequency of 7 Hz and -40 dB attenuation in the rejection band. The obtained filter length is equal to 9.

The stage of re-sampling and linear interpolation is responsible for normalizing the input of ANNs models. The gestures are variable in duration. They are composed of a variable number of acceleration samples. Thus, the stage of re-sampling and linear interpolation obtains a fixed number of acceleration samples in the 3 axes. The samples of fixed number are the inputs for MLP and SVM models. During the validation of ANNs models, the optimal number of inputs was obtained, it being equal to 10 inputs per axis.

The models of MLP and SVM with linear kernel perform pattern recognition over preprocessed input data. In the proposed configurations for MLP and SVM, time is implicitly represented since the temporal structure of the input signal is embedded in the spatial structure of the ANNs [16].

B. Implementation

The filtering, preprocessing, MLP and SVM with linear kernel algorithms were implemented in C language and numerical formats of floating-point and fixed-point. The Q5.10 numeric format is a fixed-point representation that uses integer numbers for representing real numbers of finite precision. This representation uses fewer hardware resources that the floating-point one in devices without floating-point unit (FPU) for performing arithmetic operations. The floating-point format provides more precise computations and a greater dynamic range. The algorithms were implemented as software modules in C language. An independent and portable application was developed and implemented in different test hardware platforms. In the future, the functionality of the hand gesture recognition will be implemented as a task within the context of a real-time operating system (RTOS).

A gesture is represented by three vectors $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$, each one of length N, containing the samples of the amplitude variations of acceleration versus time in three dimensional space. The raw data of input are limited and normalized in order to avoid possible overflows in arithmetic operations. The low-pass digital filter is continuously applied to the input data. For example, for the x-axis, the FIR filter of length $L=9$ with input $\boldsymbol{x}$ and output $\boldsymbol{x_f}$ is described by the difference equation

$$x_f(n) = \sum_{k=0}^{L-1} b_k \, x(n-k) \tag{1}$$

where b_k is the set of filter coefficients. Equation (1) is used in order to implement the filter algorithm.

The gestures are variable in duration depending on the user hand movement speed. When the capture button is pressed, the input samples for each axis are stored in buffers until the button is released. Before applying the algorithm for re-sampling and linear interpolation, the duration of the gestures is validated. To convert from the original sampling frequency $F_s=1/T_s$ to the desired sampling frequency $F_r=1/T_r$, the level of the input signal is obtained at the time instants $t=m*T_r$, where m is the sample index of fixed number with values 0-9 for each axis. For the x-axis, the sampling frequency conversion and linear interpolation formula is

$$x_r(m) = \left(D * m - n_m + 1\right) x_f(n_m) + \left(D * m - n_m\right) x_f(n_m - 1) \quad (2)$$

where *M* is the fixed number of samples of re-sampling output, *D=(N-1)/(M-1)* is the decimation factor, and n_m is the sample index of $\boldsymbol{x_f}$ which is located at or just above the value of *D*m*. The algorithm of re-sampling and linear interpolation involves comparisons and multiplications whose number depends on the duration of each gesture. Thus, the execution time of this algorithm depends on the number of samples of each gesture.

Fig.4 shows how the low-pass filter acts on x-axis raw acceleration samples, and how the output samples of filter are processed by the re-sampling and linear interpolation algorithm in order to maintain the fixed number of samples.

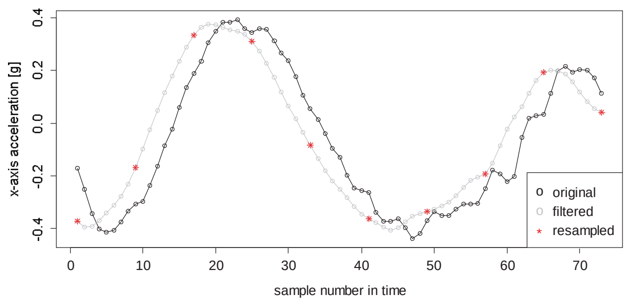

Fig. 4. Original, filtered and resampled acceleration signals for x-axis.

Re-sampled vectors $\boldsymbol{x_r}$, $\boldsymbol{y_r}$, $\boldsymbol{z_r}$, of length *M=10* correspond to 30 inputs ***e*** of the classification algorithms of MLP and SVM with linear kernel.

MLPs are feedforward ANNs. Typically, the network consists of a set of sensory units that constitute the input layer, one or more hidden layers of computation nodes, and the output layer of computation nodes. The MLP is trained by backpropagation algorithm. The forces of inter-neuron connections, known as synaptic weights, are used in order to store the acquired knowledge. The output of the *j*th neuron of the hidden layer is given by

$$V_j = g\left(\sum_{k=0}^{K} w_{jk}\, e_k\right) \quad (3)$$

where e_k represents the *k*th input signal, *K* is the number of neurons of the input layer, w_{jk} corresponds to the synaptic weight for connecting the *k*th neuron of the input layer with the *j*th neuron of the hidden layer, and $g()$ is the activation function. For *k=0*, the input corresponds to the bias value, which is a fixed input in $e_0=-1$ and weight $w_{j0}=u_j$. The output of the *i*th neuron of the output layer is given by

$$O_i = g\left(\sum_{j=0}^{J} W_{ij}\, V_j\right) \quad (4)$$

where *J* is the number of neurons of the hidden layer, W_{ij} corresponds to the synaptic weight connecting the *j*th neuron of the hidden layer with the *i*th neuron of the output layer, and $g()$ is the activation function. For *j=0*, the input corresponds to the bias value, which is a fixed input in $V_0=-1$ and weight $W_{i0}=u_i$. In both cases, the activation function of neurons used in this work is the sigmoid function via an approximation with linear segments [17]. Using (3) and (4), the MLP classification algorithm may be implemented.

The SVM is another category of feedforward ANNs also can be used for pattern classification. The SVM with linear kernel is implemented using the optimal hyperplane equation representing a linear decision surface in the multidimensional input space defined by

$$\boldsymbol{w}_0^T \boldsymbol{e} + b_0 = 0 \quad (5)$$

where ***e*** is the input vector, $\boldsymbol{w_0}$ is a weight vector and b_0 is the bias value. The training aim is to find the parameter vector $\boldsymbol{w_0}$ and the bias value b_0 for defining the optimal hyperplane. Since an SVM operates as a binary classifier, a one-versus-one multiclass strategy is used in order to solve the problem of classification of *C=8* classes. Thus, *[C*(C-1)]/2=28* binary classifiers are constructed corresponding to all possible combinations of pairs of classes. The output class is obtained by a voting scheme. Using (5), the 28 binary classifiers of the SVM with linear kernel can be implemented.

The ANNs that are implemented in this work were trained through a supervised learning process, using the libraries provided by R: nnet [18] and e1071 [19]. To achieve convergence of SVM and MLP models, regularization parameters were modified during training and validation. Fitted values for the training data, maximum number of iterations, initial random weights, etc. were adjusted in the MLP. The training was conducted in order to avoid local minimums and overtraining. MLP models have been constructed to have a correct generalization capability. In SVM with linear kernel case, the regularization parameter is cost. The cost was varied in order to obtain optimal performance of the model. According to the design criteria of Section III, the resulting MLP has architecture of 30 neurons in the input layer, 20 neurons in the hidden layer, and 8 neurons in the output layer with a total of 788 synaptic weights. Each of the 28 trained SVMs has 30 input parameters and 1 value bias, for a total of 840 input parameters and 28 values of bias.

In addition to the C language implementation, the algorithms were coded in a software application for the Android operating system, using the paradigm of object-oriented programming. The software application runs on a smartphone in order to perform real testing of the classifier algorithms. The software application uses 3-axis accelerometer of the smartphone and runs classification algorithms when the gesture is captured. The application also allowed creating a database for training the MLP and SVM models.

C. Test

During each stage of the project life cycle, test plans were defined in order to guide the corresponding validation stage. Test cases were created in each of these plans for the unit, integration and system levels.

The unit tests were performed during the codification process of each software module. The I2C driver for configuration and data capture from the accelerometer was tested. The low-pass filter was tested with a function generator and oscilloscope, verifying the cutoff frequency, gain in the pass-band and attenuation in the rejection band. The module of re-sampling and linear interpolation was tested by introducing samples of variable number of known waveforms in order to verify a proper interpolation and validated number of output samples. The MLP and the SVM with linear kernel modules were tested with database data in order to verify the correct classified output in both numeric formats.

The integration tests were performed between pairs of adjacent modules, in order to verify that the interfaces between modules operate properly. The data types and formats were checked in order to ensure correct integration of the modules.

The system tests were performed according to the defined requirements. The performance requirements tests are discussed in detail in Section V. As with any digital implementation, the error due to the finite word length in the digital representation is present. This error depends on the used numeric format. The ANNs models were coded in C and implemented in numerical fixed-point and floating-point formats. The classification results of these implementations were contrasted with the classification results of the models in R (the IEEE-754 double precision floating-point numeric format is used). The classification results were exactly the same. As a system test of a similar final implementation, the algorithms were tested on an Android-powered smartphone.

V. Experimental Results

After carrying out the training and validation of the MLP and SVM models, the best trained model of each type is selected in order to test their performance with evaluation data.

A. *Recognition Rate*

Confusion matrices are used for showing the recognition performance of the models. Columns represent the estimated or predicted output. Rows represent the real output. Main diagonal shows the number of classes correctly classified. The other cells display the classification errors. Table I shows the normalized confusion matrix of the MLP for the user-independent case, with an average recognition rate of 98.65%.

TABLE I. Normalized MLP confusion matrix for user-independent case.

MLP		estimated							
		↘	↵	→	←	↑	↓	↺	↻
real	↘	**0.982**	0	0	0.009	0	0.009	0	0
	↵	0	**0.973**	0	0	0	0.009	0.009	0.009
	→	0	0	**0.982**	0	0	0.009	0.009	0
	←	0	0	0.009	**0.991**	0	0	0	0
	↑	0	0	0	0.009	**0.991**	0	0	0
	↓	0.009	0	0	0.009	0.009	**0.973**	0	0
	↺	0	0	0	0	0	0	**1**	0
	↻	0	0	0	0	0	0	0	**1**

Table II shows the normalized confusion matrix of the SVM with linear kernel for the user-independent case, with an average recognition rate of 97.31%.

TABLE II. Normalized SVM with linear kernel confusion matrix for user-independent case.

SVM		estimated							
		↘	↵	→	←	↑	↓	↺	↻
real	↘	**0.982**	0	0	0.009	0	0.009	0	0
	↵	0.009	**0.973**	0	0	0	0	0.018	0
	→	0	0	**0.991**	0	0	0	0.009	0
	←	0	0	0.018	**0.973**	0	0	0	0.009
	↑	0	0	0.009	0	**0.991**	0	0	0
	↓	0	0.009	0.009	0.027	0.018	**0.938**	0	0
	↺	0	0.027	0.009	0	0	0	**0.964**	0
	↻	0	0.027	0	0	0	0	0	**0.973**

The average recognition rate achieved by the SVM with polynomial kernel of degree 2 was 99.21%.

In addition, tests varying the number of neurons in the hidden layer of MLP were performed, obtaining an average recognition rate of 96.12% with 10 neurons, with a decrease of 50% on the use of hardware resources.

In the user-dependent case, the variability is bounded to a particular user. An average recognition rate of 99.21% was obtained by MLP model, 99.21% by SVM with linear kernel, and 99.43% by SVM with polynomial kernel.

Table III shows a comparative table with others research works which used different pattern recognition techniques.

TABLE III. Precisions of other works.

Pattern recognition technique	Precision [%]		Number of gestures
	User-dependent	User-independent	
DTW with AP and CS [5]	99.79	96.00	18
FDSVM [6]	95.21	89.29	12
DTW [11]	93.50	75.40	8
DTW in 3-axes [20]	99.20	96.40	8
PCA and decision tree [21]	-	97.35	10
MLP proposed	**99.21**	**98.65**	**8**
SVM with linear kernel proposed	**99.21**	**97.31**	**8**

B. *Execution Times of Algorithms*

Table IV shows the execution times of the algorithms implemented in C language on different platforms with microcontroller-based hardware and numeric formats, for the gestures with minimum and maximum duration found in the database, indicated by the number of samples. Both MLP and linear SVM with kernel models exhibit similar execution times.

TABLE IV. Execution times of the algorithms.

Microcontroller	Numeric format	Re-sampling and linear interpolation [msec]		MLP [msec]		SVM with linear kernel [msec]		Clock [MHz]
		Gesture of 17 samples	Gesture of 315 samples	Gesture of 17 samples	Gesture of 315 samples	Gesture of 17 samples	Gesture of 315 samples	
PIC24FJ256GB110 (16 bits core)	IEEE-754 32-bits float	1.274	8.893	16.307	16.638	16.484	16.430	32
	Q5.10	0.887	8.498	3.117	3.068	3.027	3.021	
dsPIC33FJ64GP204 (16 bits core)	IEEE-754 32-bits float	0.509	3.557	6.523	6.655	6.594	6.573	80
	Q5.10	0.355	3.399	1.247	1.227	1.211	1.208	
PIC32MX250F128D (32 bits core)	IEEE-754 32-bits float	0.153	1.084	2.597	2.646	2.578	2.582	48
	Q5.10	0.105	1.021	0.685	0.685	0.704	0.704	
PIC32MX795F512L (32 bits core)	IEEE-754 32-bits float	0.089	0.633	1.546	1.576	1.567	1.569	80
	Q5.10	0.061	0.596	0.450	0.448	0.477	0.477	
STM32F407VGT6 with FPU disabled (32 bits core)	IEEE-754 32-bits float	0.032	0.174	0.440	0.450	0.440	0.440	168
	Q5.10	0.020	0.155	0.108	0.108	0.100	0.100	
STM32F407VGT6 with FPU enabled (32 bits core)	IEEE-754 32-bits float	0.006	0.028	0.142	0.146	0.122	0.122	168

As can be appreciated, a C language implementation could be carried out smoothly for a microcontroller in real time. The IEEE-754 floating-point format allows a greater dynamic range and a greater precision in the values of the parameters of the algorithms. This format increases processing time on the platforms without FPU. The Q5.10 fixed-point format allows a faster execution with less precision in the parameters of the algorithms due to truncation. The hardware platforms more attractive for implementation are the alternatives of 32 bits [22], [23], [24], since the system must also perform additional tasks associated with the functionality of a modern remote control within the context of an RTOS. The use of an FPU accelerates the execution of the algorithms using floating-point format. To compare results with other techniques of hand gesture recognition, in [11] an execution time of 300 msec on a 16-bit microcontroller was obtained. In [20], an FPGA-based development kit is used for implementing an acceleration unit with an algorithm execution time of 58.3 msec.

The algorithms were implemented and tested in a smartphone. A Samsung Galaxy S3 Mini smartphone was used, running the Android operating system (platform 4.03 and API level 15). Table V shows the algorithms execution times.

TABLE V. MINIMAL EXECUTION TIMES OF THE ALGORITHMS IN SMARTPHONE RUNNING ANDROID.

Numeric format	Re-sampling and linear interpolation [msec]		MLP [msec]		SVM with linear kernel [msec]	
	Gesture of 17 samples	Gesture of 315 samples	Gesture of 17 samples	Gesture of 315 samples	Gesture of 17 samples	Gesture of 315 samples
32-bits float	0.03	0.09	0.244	0.244	0.244	0.244

The precision obtained in the smartphone test for user-dependent case was 100%.

VI. CONCLUSIONS

The objective of designing and obtaining a system of accelerometer-based hand gesture recognition, with good recognition rate and reduced execution times was reached. The proposed classifier involves digital signal processing and pattern recognition techniques, whose algorithms will be embedded within the hardware. Tests conducted on different hardware platforms indicate that it is possible to implement the system with a low-cost 32-bit microcontroller. The PIC32MX250 microcontroller family is the selected platform. This platform is chosen because it executes quickly the algorithms and has enough memories of data and program. Also, this platform is the lower cost between the tested platforms. Others important factors for the choice of this platform are the local market availability and free-use development tools. In addition, both ANNs models were tested in a smartphone with good results.

Two types of ANNs were compared for finding an optimal solution between precision and hardware resources use, with suitable results in both MLP and SVM with linear kernel models. To similar processor uses, the MLP showed a slightly higher performance than the SVM with linear kernel. The MLP resource usage is controlled varying the number of hidden layer neurons. The MLP model was found to be the best and will finally be embedded into the remote control. The design and implementation of artificial intelligence models for embedded systems were presented in detail. Different microcontroller-based hardware platforms were used in order to perform and document the measurements of execution times of classification algorithms.

We have demonstrated that the implementation of a system of hand gesture recognition based on MEMS-accelerometers and low-cost resource-constrained devices is possible. We have achieved excellent pattern recognition rates, and execution times that allow a real-time operation. We have verified the appropriate use of ANNs-based models for the development of a more natural and friendly interface in order to improve the user experience in the context of the digital TV of Argentina.

VII. FUTURE WORK

A database will be collected with the final hardware for training the MLP model. This model will be embedded into the remote control. In addition, the number of gestural vocabulary classes will be increased. A Bayesian filter will be incorporated to reinforce the correct MLP output. Finally, the automatic delineation of the beginning and ending of gestures will be included. This is possible because the system could operate in continuous mode of hand gesture recognition due to the reduced execution times obtained.

REFERENCES

[1] S. Mitra and T. Acharya, "Gesture recognition: a survey," in IEEE Trans. on Systems, Man and Cybern., vol. 37, pp. 311-324, May. 2007.

[2] S. Zhou, Q. Shan, F. Fei, J. Li, C. Kwong, P. Wu, B. Meng, C. Chan, J. Liou, "Gesture recognition for interactive controllers using MEMS motion sensors," in 2009 4th IEEE International Conference on Nano/Micro Engineered and Molecular Systems (NEMS 2009), vol. 2, pp. 935-940, Shenzhen, China, Jan. 2009.

[3] S.Cho, E. Choi, W. Bang, J. Yang, J. Sohn, D. Kim, Y. Lee, S. Kim, "Two stage recognition of raw acceleration signals for 3D gesture understanding cell phones," in 10th International Workshop on Frontiers in Handwriting Recognition, 2006.

[4] B. Lee Cosío, "ANN for gesture recognition," M.S. thesis, School of Eng., Panamerican Univ., Mexico, 2012.

[5] A. Akl and S. Valaee, "Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation, & compressive sensing," in IEEE Int. Conference on Acoustics Speech and Signal Processing (ICASSP), vol. 4, pp. 2270-2273, Dallas, Texas, USA, Mar. 2010.

[6] J. Wu, G. Pan, D. Zhang, G. Qi, S. Li, "Gesture recognition with a 3-D accelerometer," in 2009 Proceedings of the 6th International Conference on Ubiquitous Intelligence and Computing (UIC 2009), vol. 1, pp. 25-38, Brisbane, Australia, Jul. 2009.

[7] G. Niezen and G. Hancke, "Evaluating and optimising accelerometer-based gesture recognition techniques for mobile devices," in AFRICON 2009, vol. 1, pp. 424-429, Nairobi, Kenya, Sep. 2009.

[8] S. Haykin, "Multilayer perceptrons," in Neural Networks: A Comprehensive Foundation, 2th ed. New Jersey: Prentice Hall, 1999, ch. 4, sec. 1, pp. 178-180.

[9] V. Vapnik, "An overview of statistical learning theory," in IEEE Trans. on Neural Networks, vol. 10, no. 5, pp. 988-999, Sep. 1999.

[10] S. Haykin, "Introduction," in Neural Networks: A Comprehensive Foundation, 2th ed. New Jersey: Prentice Hall, 1999, ch. 1, sec. 1, pp. 23-27.

[11] L. Jiayang, W. Zhen, Z. Lin, "uWave: accelerometer-based personalized gesture recognition and its applications," in 2009 IEEE Int. Conf. on Pervasive Computing and Communications (PerCom 2009), vol. 1, pp. 113-121, Galveston, Texas, Mar. 2009.

[12] uWave project database [online]. Available: http://www.owlnet.rice.edu/~zw3/projects_uWave.html

[13] The R project for statistical computing: http://www.r-project.org

[14] RStudio IDE: http://www.rstudio.com

[15] S. Haykin, "Support Vector Machines," in Neural Networks: A Comprehensive Foundation, 2th ed. New Jersey: Prentice Hall, 1999, ch. 6, sec. 4, pp. 351-356.

[16] S. Haykin, "Temporal Processing using feedforward networks," in Neural Networks: A Comprehensive Foundation, 2th ed. New Jersey: Prentice Hall, 1999, ch. 13, sec. 1, pp. 657-658.

[17] M.T. Tommiska, "Efficient digital implementation of the sigmoid function for reprogrammable logic," in IEE Proceedings Computers and Digital Techniques, vol. 150, no. 6, pp. 403-411, Nov. 2003.

[18] Package nnet [Online]. Available: http://cran.r-project.org

[19] Package e1071 [Online]. Available: http://cran.r-project.org

[20] S. Hussain and A. Rashid, "User independent hand gesture recognition by accelerated DTW," in Int. Conf. on Informatics, Electronics & Vision (ICIEV 2012), vol. 2, pp. 1033-1037, Dhaka, Bangladesh, May. 2012.

[21] X. Dang, W. Wang, K. Wang, M. Dong, L. Yin, "A user-independent sensor gesture interface for embedded device," in 2011 IEEE SENSORS Proceedings, vol. 2, pp. 1465-1468, Limerick, Ireland, Oct. 2011.

[22] PIC32MX1xx/2xx datasheet [Online], Microchip, 2012. Available: http://www.microchip.com

[23] PIC32MX7xx datasheet [online], Microchip, 2010.

[24] STM32F407xx datasheet [online], STMicroelectronics, 2012.