**Sborník vědeckých prací Vysoké školy báňské - Technické univerzity Ostrava**
číslo 2, rok 2005, ročník LI, řada strojní
článek č. 1478

**Marek PLUTA***

# VOICE-COMMAND CONTROL SYSTEM

# SYSTÉM ŘÍZENÍ OVLÁDANÝ HLASEM

**Abstract**

This paper describes an attempt to create a computer based voice-command control system, using digital sound processing, and artificial neural networks. First part covers the main purpose of the system, second one describes system's working principle, third explains methods and algorithms. Next, tools used in the creation of system are presented. Fifth part describes a current state of the system, and the last one shows its possible applications.

**Abstrakt**

Příspěvek popisuje pokus o vytvoření počítačem podporovaného systému řízení ovládaného hlasem, využívajícího číslicové zpracování zvuku a vytvořenou neuronovou síť. První část definuje hlavní úkoly systému, druhá popisuje principy práce systému, třetí vysvětluje použité metody a algoritmy. Následně je prezentován nástroj použitý pro vytvoření systému. Pátá část ukazuje současný stav systému a poslední část další možnosti vývoje.

## 1 INTRODUCTION

So far, personal computers (PC) are not powerful enough to understand spoken language, because of its complexity and variability. More powerful systems with sophisticated software, based on vocabulary and grammar rules databases, can perform it to some degree. However, there are many areas, where only a limited command set is required. This reduces a problem from two step: pattern recognition – linguistic analysis, to just a recognition problem, making use of common PCs possible.

An attempt was made to create a system flexible enough to be used in various areas. Its primary goal was to recognize a set of 104 commands for keys of extended AT standard PC keyboard. This goal included only engine, that can be connected to, or included in desired software or hardware project. Bearing in mind system's planned usage (as an input device) speed was considered a priority, at a cost of larger amount of memory needed. Second system's goal was to be flexible enough to allow easy modifications, especially in choosing various command sets.

## 2 STRUCTURE AND WORKING PRINCIPLE OF THE SYSTEM

Artificial neural network, which is the main part of the system, needs preparation to become ready to use. Because of that, the system comprises two main parts: the first one prepares it to work, and the second tests it. In real-life usage second part would be replaced with a desired program, based on testing part's algorithms, and enhanced with a desired functionality: keyboard events handling, I/O ports communication routines, or anything else. A system constructed as an engine must be flexible enough to allow its easy implementation in various projects. A good way to achieve it is to make it modular. Diversification into separate modules allows end user to choose, replace or modify desired parts according to his needs. Created system consists of four main modules:

* MSc, Department of Mechanics and Vibroacoustics, Faculty of Mechanical Engineering and Robotics, AGH University of Science and Technology, Al. Mickiewicza 30, Krakow, Poland, e-mail korowiow@biuro.net.pl

- the first module gathers an information (waveforms) and stores it on the hard drive,
- the second module performs a preprocessing on stored waveforms,
- the third module includes learning algorithm – main part of the preparation process,
- the fourth module can be used for system's testing, or as a basis for system's usage.

The preparation process needs sample voice-commands. The first module gathers them by recording separate commands via microphone. For every command there must be several (currently 16) waveforms recorded. Best recognition performance can be attained if commands are recorded by various speakers. It permits system to recognize commands spoken differently, and even enables people that didn't record sample commands to be recognized. Default time for a single waveform is approximately 1500 ms. Recordings are done with 16-bit precision, and 22050 Hz sample rate. First module also performs waveforms normalization, which enables recognition of commands recorded under various conditions – signal level will remain the same.

The learning process is not possible for raw waveforms. They have to be preprocessed and it is accomplished in the second module. If a recognition speed is a priority, preprocessing must be simple, as it will be performed for every command recorded: in the learning stage, as well as during the recognition. Therefore only necessary processing is performed here. This module takes recordings gathered by the first module, and converts them into learning patterns, writing them into a file.

The third module is most time consuming, but for one set of commands it is run only once, in the preparation stage. Moving most processing from part which is run every time a recognition is performed, to one that is run only once, shortens recognition latency. This module takes patterns generated by the second module and learns to recognize them. After a few hours of training the system is ready to use with the artificial neural network programmed to recognize voice-commands similar to those recorded in the first module.

The fourth module is a sample how to use prepared system. It records a voice-command from a microphone, normalizes, and preprocesses it, like in the previous modules. Then uses obtained data as an input pattern for the trained artificial neural network. The network performs pattern recognition. An answer is taken from its outputs, and presented to the user.

## 3 METHODS AND ALGORITHMS USED

A multi-layer feed-forward neural network is especially well fitted for pattern recognition. It needs a process of learning, after which it can recognize patterns similar to those learned. It consists of several layers of processing units called neurons. Neurons have a number of weighted inputs and one output. Their output is calculated according to the following rule:

$$y = f\left(\sum_n w_n x_n\right), \tag{1}$$

where:

$y$ – neuron output,

$x_n$ – one of neuron's inputs,

$w_n$ – particular input's weight,

$f$ – activation function.

There are many kinds of activation functions, but for this system's purpose a sigmoid one:

$$f(x) = \frac{1}{1+e^{-x}}, \tag{2}$$

was chosen. In a feed-forward multi-layer networks data flows from input units, through following layers, to output units. Every neuron from the particular layer has its output connected to one input of every neuron in the next layer, sending the same output value to all of them.

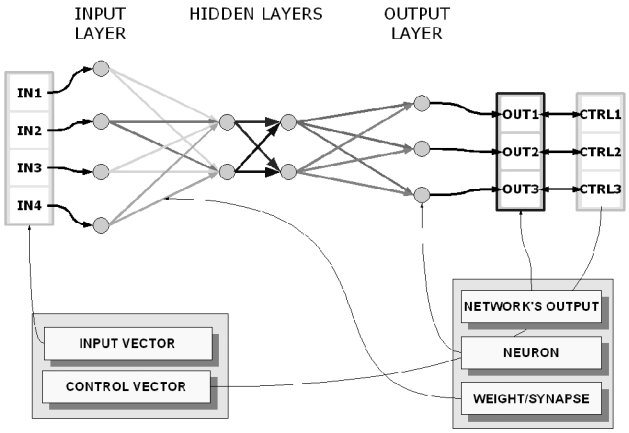


**Fig. 1** Model of feed-forward multi-layer neural network

At the beginning, weights of all neurons' connections are randomised. Then, they are set in a process of training. A supervised learning rule was chosen. It consists of many iterations where various patterns are provided to network inputs. For each learning pattern desired output is known. The first propagation of a pattern data will probably give values completely different from desired ones, but after each wrong answer connections weights are corrected a bit, according to a rule:

$$\begin{aligned} \delta_k &= y_k(1-y_k)(t_k-y_k) \\ \Delta w_{i,k} &= \eta \delta_k x_{i,k} \end{aligned}, \tag{3}$$

where:

$\delta_k$ – error of particular network output,

$y_k$ – particular network output,

$t_k$ – particular value from control vector (desired output),

$\Delta w_{i,k}$ – value of weight correction,

$\eta$ – learning parameter,

$x_{i,k}$ – output value sent from neuron $i$ in previous layer to neuron $k$ in output layer.

Error values are known only for the output layer. For previous layers they must be calculated using a back-propagation routine, which (for the sigmoid activation function) can be described as:

$$\delta_h = y_h(1-y_h)\sum_k w_{h,k}\delta_k \ , \tag{4}$$

where:

$\delta_h$ – error of a neuron in a hidden layer,

$\delta_k$ – error of a particular neuron in next layer,

$y_h$ – hidden layer neuron output,

$w_{h,k}$ – value of weight between neuron $h$ in current layer, and $k$ in next layer.

Patterns are presented in a random order. Each presentation is repeated until network's answer is near to the desired one, and then next random selected pattern is presented. This procedure is repeated up to the point, where all patterns cause an adequate answer immediately, without a need of weights' correction. In this point, the artificial neural network is ready to use.

Voice commands in fourth module of the system are recorded and preprocessed in the same way as in the training process. Network inputs are provided with values of input vector. Data is propagated through following layers. Processing in every neuron is simple – even considering many neurons the propagation is fast. Finally, answer of the whole network is read from its outputs.

An artificial neural network with circumscribed topology is a good choice for pattern recognition tasks, as it can solve hard to define problems, where an exact algorithm cannot be given – it needs only a set of sample patterns to learn on, and gives answers even with partially corrupted input data. Moreover, it treats similar objects as one – a necessity in a voice-command recognition, where every recording of the same command will differ to some degree.

A priority is to attain imperceptible recognition latencies, so only necessary computations are performed outside the neural network. A complex preprocessing could select data inserted into the network more precisely making the training process faster, but would slow down recognition process. Raw waveforms, however, proved to be too hard for network to learn on, so a few transformations had to be added. Voice-commands can be recorded under various conditions, causing signal level to vary from waveform to waveform. A normalisation prevents this. In every waveform a sample with the largest value is taken, to compute an amplify ratio, applied then to the whole waveform.

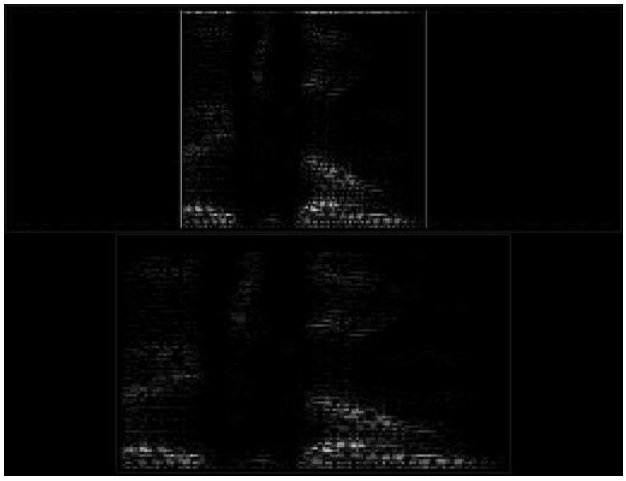

**Fig. 2** Effect of silence removal and time scaling algorithm: before – top, after – bottom

Human speech carries an information in both time and frequency domain. In waveforms only a time domain information is easily accessible. To make frequency domain information accessible as well the FFT algorithm is used. To preserve a time domain information, whole waveform is divided into smaller windows using a rectangular window function – the simplest, but the fastest one (it permits usage of fast memory copy functions, instead of weighting every sample). Its drawbacks are most significant near the window's borders, and don't affect the system quality much, because only a selected band from every spectrum is used. Windows of 512-sample size are taken every 128 samples – the step is smaller than their size. In this way, spectra of nearby windows are not independent, but better time-domain resolution is attained. Described transformation gives a two-dimensional data representation, with the time based and the frequency based information easily accessible. Considering 22050 Hz sampling frequency, 128 samples in the time-domain gives resolution of 5,8 ms, and 512-sample window gives 43 Hz resolution in the frequency-domain.

Frequency band ranging from 0 to 11025 Hz is not only unnecessary, but also disadvantageous, containing too much obsolete information. Upper limit of 4085 Hz proved enough. The removal of low frequencies was also necessary, as the mains frequency (50 Hz) and its harmonics (especially 150 Hz) were masking any other signals in this band. Finally, a band of 215 – 4085 Hz was chosen and proved enough to distinguish between various voice-commands.

Periods of silence at the beginning and in the end of recording prevent the neural network's training process from being convergent. A silence-removal algorithm is used to correct it. Finally, the time scaling is performed: in the time frequency domain number of spectra is set to 160 for every pattern. After the silence removal, patterns have various numbers of spectra. Their deficiency or redundancy is corrected with the linear interpolation.

Finally, a topology of the neural network had to be chosen: a number of layers and a number of neurons in each layer. Two values were fixed: 14400 input units, equal to the size of input pattern, and 104 output units, equal to the number of recognized commands. For a neural network to solve a non-linear problem a minimum of 2 hidden layers is required. A layer is called hidden if it isn't network's input or output. There are no precise clues how many units each layer should contain, as it depends on a processed data. There was one limitation: an amount of memory used by the working neural network. An attempt was made not to exceed 100 MB, to allow system's usage even on computers with 128 MB of memory. Number of neurons in first and second hidden layers was set equally to 600, resulting in a following topology: 14400 – 600 – 600 – 104.

## 4 TOOLS USED TO CREATE THE SYSTEM

Every part of the system is a computer program written in C language using GNU C Compiler. C is much faster than an object-based C++, and has useful libraries, including routines for sound recording. It allows porting created software to different architectures. Programs are written in a way that allows them to be compiled under either Linux or Windows. There is a possibility to run separate modules on different machines. Especially the third one is recommended to run on a fast machine.

In a default form – with a set of 104 commands – the system was tested on PCs with processors ranging from AMD K6-2 500 MHz to AMD Athlon64 3400+. Amount of memory varied from 128 to 512 MB. Several sound cards were used: Creative Sound Blaster 16, Sound Blaster Live, and integrated sound solutions. The only important limitation is amount of memory needed, as the artificial neural network takes 95 MB. A processor clock speed is important especially in system's preparation stage: it takes up to 10 hours to pass the third module for a 1400 MHz Athlon processor. Recognition latency of prepared system is imperceptible even on 850 MHz Duron processor.

## 5 CURRENT STATE

An advantage of particular methods in voice-command recognition cannot be predicted without a large number of thorough tests. Single test consisting of the preprocessing and the training lasts about 10 hours. It was difficult to test many sets of parameters using a single PC. Only in last development stage there emerged a possibility to use 26-node Linux-based cluster. It was used to test various neural network topologies. A full test should include estimation of the recognition efficiency for chosen parameters. It should be tested with commands spoken by a real person. However, an amount of parameters' combinations is too big to accomplish this for every set. Because of that, a shorter testing procedure was undertaken. Training process, although not as reliable as the real system efficiency test, can give information on system's efficiency. It can be completed automatically, as the third module was equipped with time measuring procedures. Generally speaking, the smaller number of propagations and corrections, the better system's efficiency was attained.

This method isn't enough to fine-tune system's parameters. Parameters space is multi-dimensional, so that it's two- or three-dimensional subspaces must be considered at a time. Currently, the most thoroughly tested parameters are numbers of neurons in hidden layers. Further tests of preprocessing parameters are important now, as they have a great impact on the overall efficiency.

The rate of correctly recognized commands is approximately 80% for a not prepared speaker. After some work with the system, user should become used to it, and the efficiency should raise. Attained efficiency is probably not high enough to use the system in most real-life application in current stage, but as it is constantly raising, it is only a matter of time that the system becomes reliable enough.

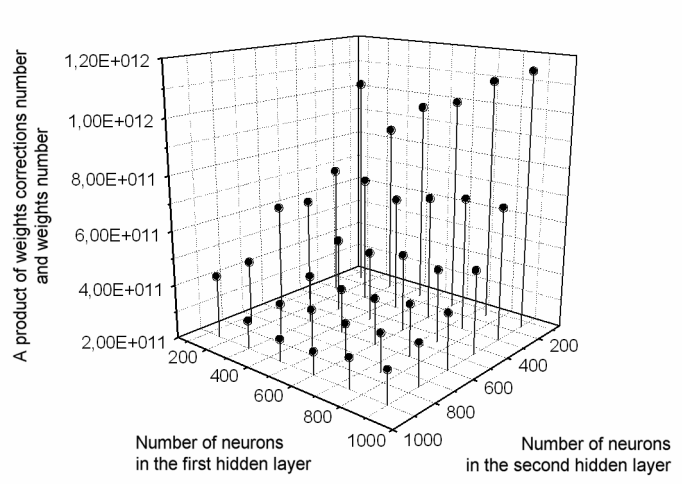


**Fig. 3** Training process efficiency related to number of neurons in hidden layers: the lower values, the faster process

## 6 POSSIBILITIES OF SYSTEM UTILISATION

A main goal was to create a basis for voice-command control system starting from a keyboard emulation. Even though efficiency attained so far can be insufficient for some purposes, it would fit others. It can be useful for disabled people, or in situations where user's hands are occupied operating another device (in laboratories or industry). The system is flexible. With knowledge of C it can be highly customized, having various command sets for various devices. The system can be run on one machine, as this machine's input device, or its parts can be run on separate machines: some to record commands, one for recognition, controlling other devices. A modular structure gives a possibility to use the system as a pattern recognition engine for nearly any kind of patterns. Considering sound, selecting another part of frequency band could help distinguish not commands, but, for example, speakers. After changes in first and second module it can be freely used in other ways: another time based process recognition, visual recognition, searching for patterns in texts, and more.

### REFERENCES

[1] OZIMEK, E. *Dzwiek i jego percepcja*. Warszawa – Poznan : Wydawnictwo Naukowe PWN, 2002. ISBN 83-01-13772-X.

[2] PRESS, W. H., TEUKOLSKY S. A., VETTERLING, W. T., FLANNERY, B. P. *Numerical Recipes in C*. Cambridge University Press, 1997. ISBN 0-521-43108-5.

[3] KRÖSE, B., VAN DER SMAGT, P. *An Introduction to Neural Networks*. The University of Amsterdam, 1996.

**Reviewer:** doc. Ing. Radim Farana, CSc., VŠB-Technical University of Ostrava