A NEW ON-LINE ROBUST APPROACH TO DESIGN NOISE-IMMUNE SPEECH RECOGNITION SYSTEMS

Fabian Vargas, Rubem D. R. Fagundes, Daniel Barros Jr.

Electrical Engineering Dept.
Catholic University – PUCRS
Av. Ipiranga, 6681. 90619-900 Porto Alegre – Brazil
vargas@computer.org

Abstract

Hereafter, we present a new approach[1] dealing to couple with the harmful effects of noise on speech recognition systems (SRS). This approach is oriented to hardware redundancy and it is essentially a modification of the classic software-based recovery blocks scheme. When compared to conventional approaches using Fast Fourier Transform (FFT) and Hamming Code, the primary benefit of such a technique is to improve system performance when operating in real (i.e., noisy) environments. The second advantage is related to the considerably low complexity and reduced area overhead required for implementation. We implemented a fully version of the proposed algorithm using the C language. The goal of this implementation is twofold: first, it is used in this paper to illustrate the effectiveness of the proposed ideas and enrich further discussions. Second, it will be used in a near future as a system single-source code specification that will be partitioned and implemented according to a HW-SW codesign methodology we have proposed in previous works.

Keywords: Digital Signal Processing (DSP); Speech-Recognition Systems (SRS); Noise Immunity; On-Line Testing; Software-Based Recovery Blocks (SBRB); Area overhead; Performance Degradation.

1. Introduction

It is of common agreement the large increase of the number of applications requiring digital signal processing (DSP) components (implemented either in HW or SW parts). For instance, the combination of intelligent, Internet-enabled handheld devices and wireless technology has become popular very quickly. However, users express two consistent, significant complaints: it is difficult to use the keyboard-, menu-, or stylus-based interface to input any but the simplest functions; and the small screens are inadequate for meaningful output.

The obvious solution to both problems is to use speech-recognition systems (SRS) technology. But even simple voice applications are mistake-prone in noisy environments and require processing and other resources beyond those available in many cellular phones and personal digital assistants (PDAs). As said by Dr. Ken Hyers[2] to the IEEE Computer Magazine [1]: "*In a handset, an SRS becomes too error prone because the devices do not have enough processing power to handle noise filtering*". Devices will eventually have more capabilities as SRS' accuracy and efficiency improve. SRS sometimes deliver more than 90 percent accuracy in laboratory conditions, but the rate can drop to 50 percent or less in noisy environments [1].

If we consider applications like real-time robot decision-making, Internet interactive multimedia, portable telephones or aircraft on-board main computers, we can have a large spectrum of dedicated functions that can be implemented by means of voice processing, recognition algorithms, and powerful data communication protocols. Thus a high-throughput and reliable system architecture is mandatory since these systems are expected to be used in real-time critical applications.

Companies such as IBM, Lucent Technologies, and Motorola continue to work on improving and integrating SRS in small-footprint devices. For example, IBM has developed a speech-enabled coprocessor prototype for the Palm III that lets users execute several hundred commands and also offloads work from the main processor. In addition, Philips Electronics recently announced that it has ported its SRS software to a digital signal processor (DSP) developed by Lucent Technologies for hands-free devices [1].

The challenge of getting sufficient processing power and memory to handle many complex commands (in particular those dedicated for noise filtering) in a package small and inexpensive enough to work with handheld devices has delayed increased implementation of SRS in cellular phones and PDAs.

Therefore, hereafter we propose a new approach to couple with the high complexity of algorithms dedicated to noise filtering. Compared to conventional approaches based on Fast Fourier Transform (FFT) and Hamming Code, the proposed technique improves the degree of success of SRS in terms of recognizing the correct words in real noisy environments. The algorithm used is based on *Hidden Markov Models* (HMM) and performs on-line testing by means of the classic *Software-Based Recovery Block* (SBRB) technique [15]. The use of HMM and SBRB yields to the proposed approach a *low complexity* and *reduced area overhead* when implemented in hardware.

The reminder of the paper is divided as follows: Section 2 presents the basic concepts involving speech

[1] *This work is partially supported by CNPq and FAPERGS.*

[2] *Dr. Ken Hyers is senior analyst for mobile commerce with Cahners In-Stat, a market research firm [1].*

recognition systems. The control and data flows, by means of general block diagrams, are briefly introduced to readers not familiar with such type of DSP systems. Section 3 details the proposed on-line testing approach by explaining how it "reconstructs" noisy signals. It is also addressed a discussion about the low complexity and the reduced area overhead required to implement the proposed algorithm in hardware. Section 4 is devoted to experimental results. A computation example is presented to illustrate the effectiveness of the proposed technique. To conclude, Section 5 presents the final considerations and future work.

2. Preliminary Considerations on the General Structure of Speech Recognition Systems

A speech recognition system (SRS) is basically a pattern recognition system dedicated to detect speech. In other words, to identify language words into a sound signal achieved as input from the environment. Fig. 1 shows the main steps performed by a front-end speech recognition system [2-5].

In the *Signal Analysis* step, a speech sampling will be made with an A/D converter. Those samples are processed in order to extract some relevant features from speech signal input. This step is responsible for signal handling, by converting the analog signal sampling, into a digital representation. The last task performed in this step is the vector quantization, when the speech signal is then replaced by a proper sequence of label-codes (this is the input for the next step of the SRS system, which is responsible for pattern matching).

The *Pattern Matching & the Decision Logic* steps are the "identification" steps, where the words spelled in the speech signal are recognized by generating a sequence of text words. The observation sequence is evaluated using *Hidden Markov Models* (HMM), which as the acoustic reference pattern, plays the main role in the recognition process [10,11].

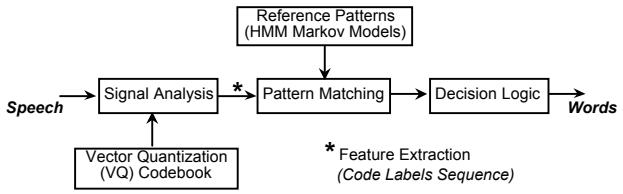


Fig. 1. General block diagram of speech recognition systems [2-5,16].

The main tasks performed in the signal analysis step are depicted in fig. 2a, and are described as follows:

Sampling: the SRS converts speech sound from the outside world into digital representation. Essentially, this task will include a sample and hold device, and an analog-to-digital (A/D) converter.

Low Pass Filter: cuts those high frequencies found on the signal due to sampling. Usually this filter is adjusted by sampling rate [4,12]. Typical cutoff frequency values in this type of application have ranged from 4KHz to 8 KHz.

Pre-Emphasis Filter: adjusts the high variations on spectrum frequencies due to glottal pulse and lips radiation found in the speech signal behavior [5,6].

Windowing: cuts the speech signal into blocks of 10 ms signal frame each. A Hamming window adjusts those frame samples in order to avoid the "spectral leakage" due to signal framing segmentation [7,13].

LPC/Cepstral Analysis: DSP algorithms process each frame by generating a vector of cepstral coefficients, which are the filter-model's parameters, and describe the spectral behavior for that frame [5,8].

VQ – Vector Quantization[3]: each *cepstral coefficient* is evaluated by distance measure by using as a map a codebook with reference vectors in the acoustic space. The final output is a sequence of code labels[4] (usually called *observations sequence*) that will be evaluated by the Pattern Matching Process [4,6,9].

To do so, the *Pattern Matching & Decision Logic* Block input data (i.e., the observation sequences from the *Signal Analysis* Block) are actually an "index" to access the local cache memories associated with each HMM block, as seen in fig. 2b. As response to these accesses, the local memories output the respective "probability of changing state from one node to another", in the Markov Model). Then, these probability values are added to the previous values stored in the pre-accumulators of the *Pattern Matching & Logic Decision* Block. (See fig. 2b.)

At the end of this process, the *Decision Logic* Block compares the final probability values from the HMM blocks against a reference (threshold) value and selects the one with higher score (i.e., higher probability) of being the searched word. Then, recognizing the word. The *Pattern Matching & Decision Logic* steps have been implemented by the Viterbi algorithm [2,6,11,14] to perform the evaluation of the code-label sequence against the HMM structures. Fig. 2b illustrates this situation for the pattern matching process [2,3].

3. The Proposed Methodology

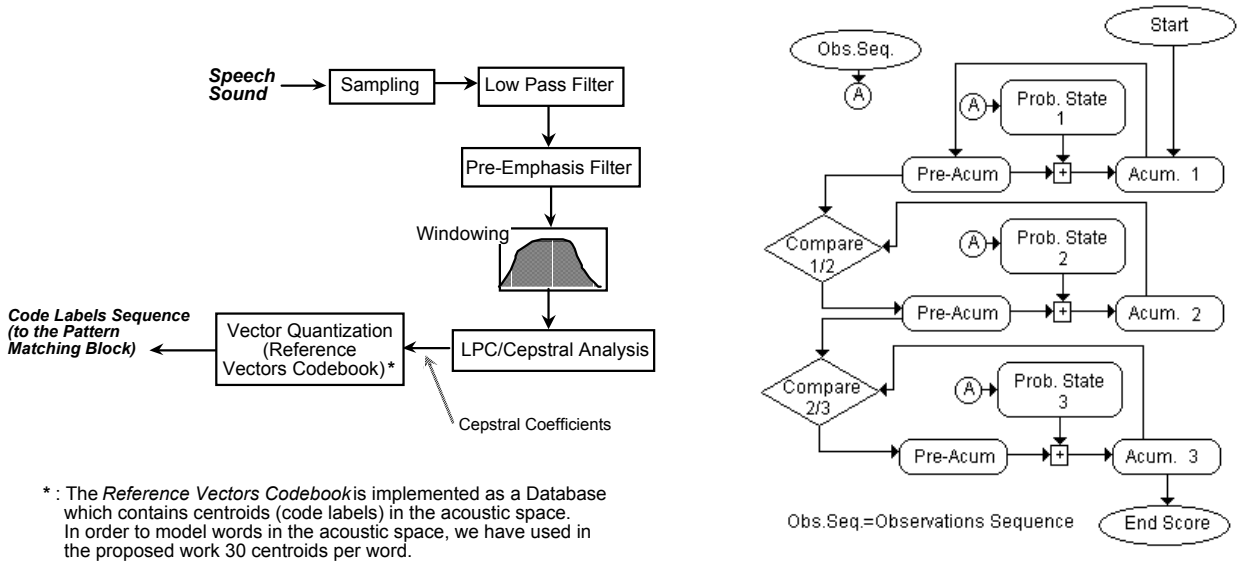
Although convolutional codes, first introduced by Elias [14], have been applied over the past decades to increase the efficiency of numerous communication systems, where they invariably improve the quality of the received information, there remains to date a lack of reliability when such an information is used to represent speech. In other words, voice-oriented systems used in bank-transaction or security applications are frequently struggled by noise like electromagnetic interference, or just background noise. As consequence, the information (i.e., the speech) is partially (or even totally) buried by noise. Thus, reducing the reliability of the incoming signal.

To overcome this problem, we are proposing a new approach to minimize (and if possible, eliminate) the noise associated with voice signal. Thus, allowing a more reliable speech recognition process by the system. The first idea behind the approach is to perform the reconstruction of the incoming signal every time the code labels sequence on the process of recognition does not respect a fundamental rule of quality. The algorithm used is based on *Hidden Markov Models* (HMM) and performs on-line testing by means of a modified version of the

[3] *LPC: "Linear Predictive Coding ". Cepstral is a homomorphic analysis usually employed in speech processing [2,5,8].*

[4] *After VQ, each code label in the sequence output (i.e., a centroid in the acoustic space) is called "observation".*

classic *Software-Based Recovery Block* (SBRB) technique [15]. Fig. 3 details the main blocks of the proposed approach.



(a) **(b)**

Fig. 2. SRS general block diagram: **(a)** *Signal Analysis* Block; **(b)** *Pattern Matching & Decision Logic* Block [2,3,16].

Basically, the approach works as follows: the incoming signal is analyzed by the Viterbi Algorithm which was implemented by means of the Hidden Markov Models (HMM), exactly as described in Section 2 (Fig. 2b). Note that the input signal (indicated by "*" in fig. 3) is the observations sequence, i.e., digital code labels representing speech. Therefore, in the occurrence of an incoming signal, Subsystem-I begins the process of recognition of the arriving observation codes. Note that the SRS must be previously trained to do this work, and that the number of words that can be recognized is equal to the number of HMM Blocks inside Subsystems I or II. Each one of these HMM Blocks is used to model a single word.

The whole approach is controlled by the *Viterbi Algorithm Controller* ("*VAC* Block", in fig. 3) which supervises the processes running on *Subsystem-I* and on the *Code Labels Sequence Simulator (CLSS)*. While the goal of the "HMM" Blocks in *Subsystem-I* is to compute the HMM probability scores for each of the incoming observations (*), the goal of the *VAC* Block is to check if the code labels sequence respects a fundamental rule (to be defined later) and if it is not the case, the *VAC* Block switches from *Subsystem-I* to *CLSS*. The *VAC* Block remains switched on the *CLSS* during a predefined period of time, and then, bounces back to the original incoming signal, i.e., to *Subsystem-I*. This process is dynamically performed every time the code labels sequence forming the original noisy speech signal does not respect a fundamental rule of quality.

For comparison purpose, the proposed *VAC* Block performs the *Acceptance Tests* that are commonly carried out in *software-based recovery blocks* schemes [15]. Having this comparison in mind, Fig. 3a shows the general block diagram of the proposed approach, which executes the following algorithm:

Ensure **T**
By **P**
Else **Q**
Else **Error**

Where:

T: Word recognition process
P: Primary Block (STEP-I)
Q: Alternate Block (STEP-II)
Error: Word recognition failure (insufficient signal-to-noise ratio, preventing the word to be correctly recognized).

The *CLSS* Block operates synchronously with the "HMM" Blocks of *Subsystem-I* under a unique clock signal generated by the *VAC* Block. The CLSS performs a pseudorandom generation [19] of code labels sequence segments. These pseudorandom-generated segments are used by the *VAC* Block to replace those segments of the original incoming information that are corrupted by noise. The time duration of each of these segments is 10ms. The whole process is controlled by the *VAC* Block, which is in charge of executing the *Acceptance Test* at the outputs of *Subsystem-I*. The Acceptance Test is, in summary, *the computation of the distance between two consecutive code labels forming the original incoming speech signal and the comparison of this value with a predefined one*. Fig. 4 summarizes this process.

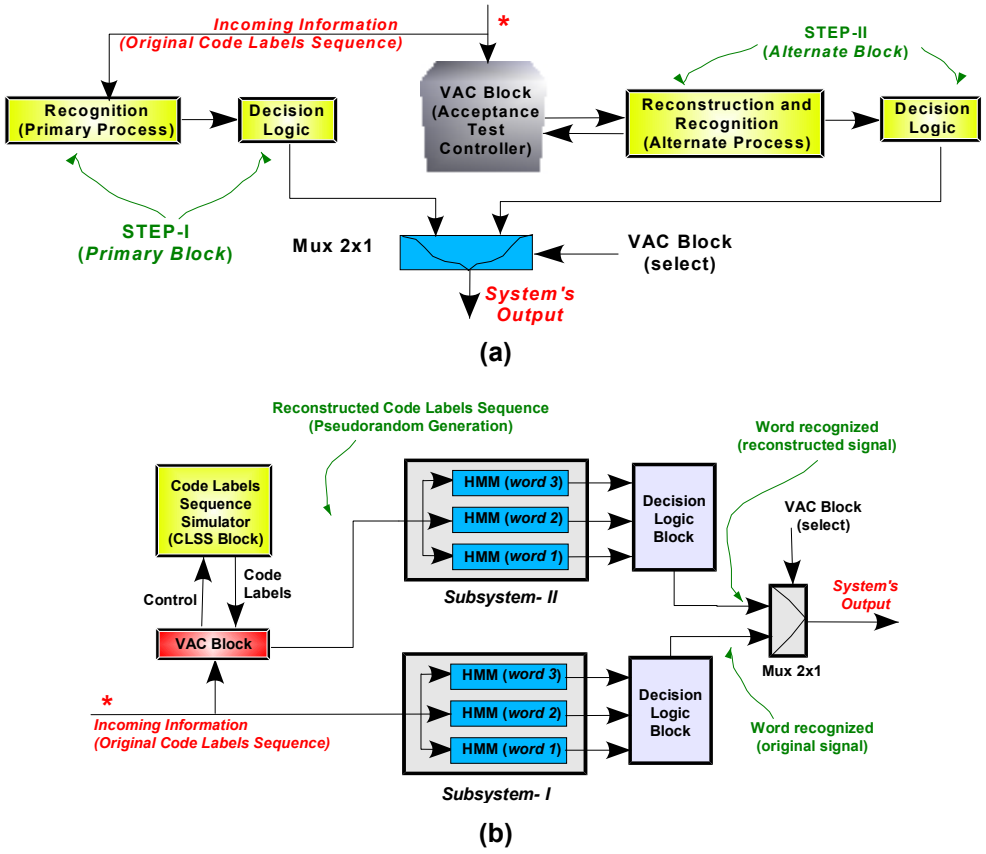


Fig. 3. The proposed approach: **a)** General Block Diagram; **b)** More details for a 3-word recognition example.

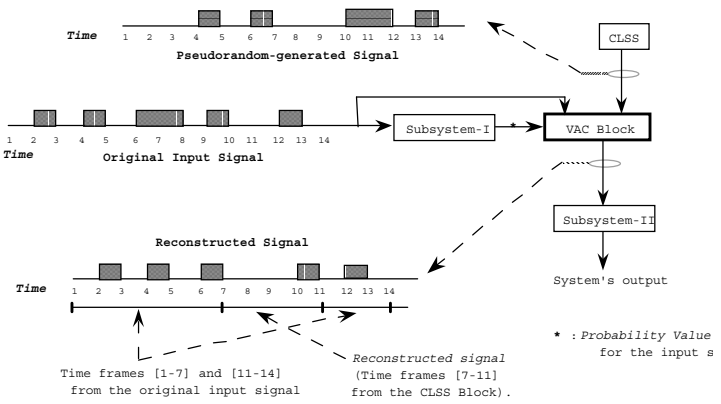


Fig. 4. Details of the switching process for voice signal reconstruction. The pseudorandomly-generated observations [19] replace the incoming noisy signal segments during the time period in which the *VAC* Block detects that the distance between two consecutive code labels forming the original incoming speech signal is larger than a predefined reference value.

3.1 The Reconstruction Technique and the Pseudorandom Generation of Code Labels

The reconstruction technique of the noisy original signal and the pseudorandom generation process of observation sequence segments are described as follows:

I) The observation sequence (or *code labels* sequence) segments are stored in the Reference Vectors Codebook (RVC), in the Vector Quantization Step (see fig. 2a). The RVC can be implemented in hardware (resp. software) in the form of a local cache memory (resp. large amount of space allocated in the system's memory) containing data generated during the training procedure of the SRS. Therefore, each address of the RVC stores a single number normalized in the interval [0,1], which represents a *reference point in a two-dimensional map of the acoustic space*.

II) For the above code labels sequence, the HMM Blocks of *Subsystem-I* compute the score for the respective word they were trained for. Concurrently to this process, the *VAC* Block monitors whether the distance between every two consecutive code labels of the incoming original sequence increases by an *approximately constant step*.

III) If the *VAC* Block detects large distance differences (larger than a predefined threshold value) between two subsequent code labels, then probably noise above expected values is struggling the original voice signal. As consequence, the word in the recognition process will not be correctly identified. To avoid this problem, the *VAC* Block discards the last two consecutive code labels and switches to the *CLSS* Block. In other words, the *VAC* Block switches from the "Primary" to the "Alternate" Block after the execution of an *acceptance test* that detects the violation of the following rule:

> **Fundamental Rule**: "*the distance between two consecutive code labels must always be positive and approximately constant*".

IV) Once the recognition process is switched to the *CLSS* Block, code labels are pseudorandomly generated. At the same time, the fundamental rule described in (III) is checked and if the distance between two consecutive pseudorandom generated code labels does not obey this rule, they are *discarded* by the *VAC* Block. In this case, *another code label is generated and the rule checked again*. If the fundamental rule is *respected*, then the code label is *appended to the original code labels sequence*. By doing so, the original incoming signal is then, "reconstructed". This process is repeated consecutively until the *VAC* Block switches back to the *Primary* Block (*Subsystem-1*), which happens at every 10ms.

Once the reconstructed code labels sequence reaches *Subsystem-II*, the goal of the "HMM" Blocks therein is to compute, exactly as it was done previously by the "HMM" Blocks in *Subsystem-I*, the probability scores of changing from one state to another in the Markov chain. However, note that in *Subsystem-II* **this probability is computed for the reconstructed signal** (in *Subsystem-I*, these probabilities were computed for the original incoming signal, before the reconstruction process).

Finally, the next step is the *Decision Logic* Block, which chooses among the words on the process of recognition, the one with the higher score, i.e., the one with the higher probability of occurrence (note that the higher score must also be higher than a predefined threshold value in order to be considered a "valid" word).

Discussions:

Note that as described above, due to the fact the code labels generated by the *CLSS* Block have been generated randomly, it is expected that there will exist noise in the "reconstructed" signal as well. This is true because it is quite probable that the random-generated observations sequence is different from the one generated from the noise-free original signal. On the other hand, it is also true that the noise present in the "reconstructed" signal due to the random-generation process is lower than the noise embedded in the incoming original signal. As conclusion, in a given range ($min_{noise\text{-}signal}$, $max_{noise\text{-}signal}$), it is expected to have a positive improvement in the quality of the voice signal.

In these considerations, "$min_{noise\text{-}signal}$" is the minimum noise affecting the original incoming signal, below which the random-generated code labels in the *CLSS* Block have no improvement in the final signal quality (in this case, the SRS operates correctly with or without the use of the *CLSS* Block). Similarly, the $max_{noise\text{-}signal}$ is the maximum noise affecting the voice signal, above which the SRS is no more able to operate correctly (even if the *CLSS* Block is selected).

4. Experimental Results

This section presents a computation example that we have developed to illustrate the proposed approach. With this purpose, we implemented an SRS to recognize 4 words. The system was fully described in the C language[5]. To train the system, we used the database *TIMIT*, developed in a joint cooperation between *Texas Instruments Corp*. and the *Massachusetts Institute of Technology* (MIT) [17,18]. Also, we have selected 100 people to pronounce each of the 4 words, which resulted in a database of 400 reference words.

Table 1 presents the 4 words we have implemented and trained the SRS, and summarizes experiment results for the system operating in the ideal noise-free environment of a laboratory. Table 2 presents results for the SRS behavior with respect to different levels of noise affecting the input speech signal.

The system was trained to recognize 4 words, each of them represented by a different number of 10ms-duration code labels. More precisely, we observed the minimum number equal to 41.26 and the maximum equal to 82.12 code labels per word, as seen in Table 1. Fig. 5 shows the architecture of the system implemented. In this case, the input signal indicated by "*" in Figs. 3 and 5 is represented by an average sequence of 2328ms duration for 232,8 code labels: 4 words x 58.2 code labels per word in average x 10ms = 2328ms overall speech time duration.

Words *	Average Number of Code Labels per word	System confidence (%) [frequency of which words are recognized correctly]
She	41.36	98.00
Greasy	82.12	100.00
Dark	68.06	100.00
All	41.26	97.00
Average	58.2	98.70

***** *Sequence size: 58,2 code labels in average per word. Each code label represents 10ms of continuous speech sound.*

Table 1. Words used to implement and train the system, and the respective system degree of success to recognize the correct word. (*Results for an ideal noise-free environment.*)

Percentage of the voice signal corrupted by noise * (%)	System confidence (%) [frequency of which words are recognized correctly]
60	81.75
50	87.50
40	92.00
30	95.00
20	95.75
10	98.50

Table 2. Preliminary results showing the SRS performance in terms of recognizing the correct word for different noise levels affecting the input speech signal.

[5] *Around 3000 lines of code: 2000 lines dedicated for the recognition and training procedure; 400 lines for the signal A/D sampling and filtering procedure; and 600 lines for interfacing the system with the database TIMIT.*

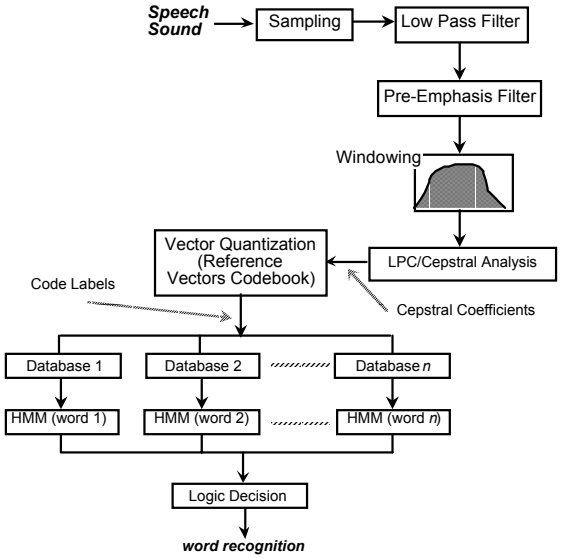


Fig. 5. General block diagram of the SRS implemented and trained to recognize 4 words. (Note that for 4 words implemented, *n* is equal to 4.)

5. Final Discussions & Future Work

We have presented a new approach to minimize (and if possible eliminate) the noise struggling voice signal in speech recognition systems (SRS). The basic idea behind the proposed approach is formalized around the traditional *software-based recovery blocks* (SBRB) scheme found in the literature. However, in our case we have modified the SBRB scheme towards a hardware implementation and to the specific needs of speech recognition systems.

Aiming to improve input voice signal quality, the proposed approach is split in two steps: in STEP-I, the algorithm performs in the "Primary" Block a first try to recognize the words. If it succeeds, there is no need to move to the second step (in the "Alternate" Block) since noise is under acceptable values and the words were recognized with a good confidence level.

If the system fails to recognize the word in STEP-I (a fundamental rule of quality defined in *Section 3.1* was not respected), the algorithm moves to STEP-II. In this case, the proposed approach rules that for each of the 10ms time frames of the input voice signal whose rule of quality was not attained, there will be a replacement by a 10ms time frame generated randomly by a *Code Labels Sequence Simulator* (CLSS). At this moment, we have a "reconstructed" voice signal whose recognition score is recomputed at the end of STEP-II. Then, one of the three possible situations may happen: the system succeeds to recognize the good word, the system fails by recognizing the wrong word, or the system aborts because the minimum recognition score was not attained.

The proposed speech recognition system was fully implemented in the C-language. The obtained results demonstrate this is a very promising technique and indicate that the next step of the research will be dedicated to a *real case-study*. This *case-study* will be partitioned into hardware and software parts and prototyped using programmable logic device technology and dedicated DSP processor.

References

[1] CLARK, D. Speech Recognition: The Wireless Interface Revolution. IEEE Computer Magazine, Vol. 34, No. 3, Mar. 2001, pp.16-18.

[2] RABINER, L. R., JUANG, B. H. Fundamentals of Speech Recognition. New Jersey , Prentice Hall, 1993.

[3] FAGUNDES, R. D. R.; VARGAS, F.; BARROS Jr., D. A Viterbi Algorithm Implementation Using Hardware/Software Co-Design in Speech Recognition Systems. The International Association of Science and Technology for Development Conference - IASTED'2000. Marbella, Spain, Sep. 19-22, 2000. (http://www.iasted.com).

[4] DELLER, J., PROAKIS, J.G., HANSEN, J. H. L. Discrete-Time Processing of Speech Signals. New York: Macmillan, 1993.

[5] GRAY Jr., A.H.; MARKEL, J. D. Linear Prediction of Speech. Communication and Cybernetics, 3 ed., Berlin. Springer, 1982.

[6] RABINER, L. R.; LEVINSON, S. E. A speaker- independent, syntax-directed, connected word recognition system based on hidden markov models and level building. IEEE Transactions on Acoustics, Speech, and Signal Processing, v.33, n.3, p.561-73, June 1985.

[7] OíSHAUGHNESSY, D. Speech Communication Human and Machine. Massachusetts: Addison-Wesley, 1987.

[8] MAKHOUL, J. Linear Prediction: a Tutorial Review. Proceedings of the IEEE, v.63, n.4, p. 561-580, Apr. 1975.

[9] LINDE, Y.; BUZO, A.; GRAY, R.M. An Algorithm for Vector Quantizer Design. IEEE Transactions on Communications, v.28, n.1, p.84-95, Jan. 1980.

[10] RABINER, L. R.; WILPON, J. G.; SOONG, F. K. High performance connected digit recognition using Hidden Markov Models. IEEE Transactions on Acoustics, Speech, and Signal Processing, v.37, n.8, p.1214-1225, Aug. 1989.

[11] VARGAS, F.; FAGUNDES, R. D. R.; BARROS Jr., D. Orienting Redundancy and HW/SW Codesign Techniques Towards Speech Recognition Systems. 2nd IEEE Latin American Test Workshop - LATW2001. Cancun, Mexico, Feb. 11-14, 2001. pp. 226-233.

[12] PROAKIS, J. G.; MANOLAKIS, D. G. Digital Signal Processing: Principles, Algorithms and Applications. Prentice Hall, 3rd Edition (Oct. 5, 1995), ISBN: 0133737624.

[13] OPPENHEIM, A. V.; SCHAFER, R. W.; BUCK, J. R. Discrete Time Signal Processing. Prentice Hall, 2nd Edition (Feb. 15, 1999), ISBN: 0137549202.

[14] VITERBI, A. J. Convolutional Codes and Their Performance in Communication Systems. IEEE Transactions on Communications Tech., Vol. Com-19, No. 5, Oct. 1971, pp.751-772.

[15] PRADHAN, D. K. Fault-Tolerant Computer System Design. Prentice-Hall, 1996. 544p.

[16] VARGAS, F.; FAGUNDES, R. D. R.; BARROS Jr., D. 26th IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP2001. Salt Lake City, Utah - USA, 2001. (http://www.icassp2001.org).

[17] Dekker, M. Digital Speech Processing, Synthesis, and Recognition. (1989), ISBN: 0824779657.

[18] http://www.ti.com.

[19] BARDELL, P. H.; MCANNEY, W. H.; SAVIR, J. Built-In Test for VLSI – Pseudorandom Tecniques. John Wiley & Sons Inc., New York, USA 1986.