FULL-CUSTOM SOFTWARE FOR START/END POINT DETECTION OF ISOLATED-SPOKEN WORDS

Mihai SIMA, Dragoş BURILEANU, Corneliu BURILEANU, Victor CROITORU

«Politehnica» University of Bucharest, Faculty of Electronics and Telecommunications,
Blvd. Iuliu Maniu 1-3, sect.6, Bucharest 77202, Romania,
phone: (+40.1) 410.64.45, fax: (+40.1) 411.11.87,
e-mail: {sima,croitoru}@ADComm.pub.ro, {bdragos,cburileanu}@mESsnet.pub.ro.

Abstract

An important problem in speech recognition is to detect the presence of speech in a background of noise. This problem is often referred to as the start/end point location problem.

The algorithm proposed is based on two measures of speech: short-time energy and zero-crossing rate either/or low-pass filtered version of these. The present implementation of our algorithm makes use of a true multi-tasking operating system (Linux), to account for the asynchronous events of speech acquisition. The algorithm forms, along with a noise compensation algorithm, a front-end processor of a speech recognizer system.

Firstly, the paper motivates the need of speech detection for speech recognition tasks. Next, we describe the philosophy of start/end point detector and show some software implementation details. The paper ends with main results and conclusions.

Keywords: start/end point detection, isolated-spoken words recognition, energy, zero-crossing rate, multi-tasking operating system (Linux).

1. Introduction

An important problem in speech recognition is to detect the presence of speech in a background of noise. This problem is often referred to as the start/end point location problem [5].

The algorithm proposed for start/end point detection of isolated-spoken words is rather a modified version of the one proposed in [1]. The main difference is the original one performs an end point detection on a fixed-size recording including one word only, while the present one extracts the isolated-spoken words from a continuous recording.

Our algorithm is based on two measures of speech: short-time energy and zero-crossing rate either/or low-pass filtered version of these.

The algorithm forms, along with a noise compensation algorithm, a front-end processor of a speech recognizer system. It should be noted here that speech detection is necessary for speech recognition since word boundaries must be approximately known to trigger the recognizer correctly.

Moreover, proper location of regions of speech not only substantially reduces the amount of processing, but also increases the recognition rate. For example, two utterances are to be compared by a dynamic time warping (DTW) procedure, each of them including about one third of useful speech information (fig. 1). The regions marked as *noise* are susceptible to produce errors, as utterances may be considered as "equals" during these regions. We believe this is one of the major difficulties in classifying among digits "2" and "8", both of them being of short length [6].

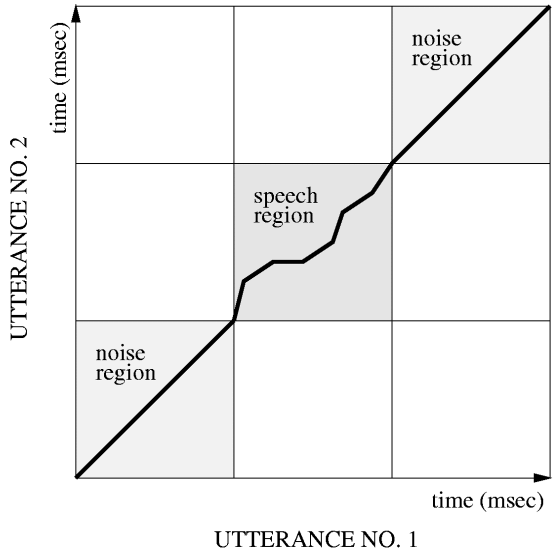

Fig. 1. DTW alignment for two utterances including about one third of useful speech

2. Algorithm for start/end point detection of isolated-spoken words

A reliable discrimination between voice and silence constitutes a difficult classification problem. It is known that zero-crossing rate generally complements the energy, i.e. during a word, when energy is low, zero-crossing rate is usually high and vice-versa [1][5]. Therefore, the algorithm rely on these two measures of speech, one with meaning of magnitude and the other with meaning of frequency [1]:

$$\hat{E}(n) = \sum_{m=0}^{N-1} \left| w(m)x(n-m) \right| \qquad (2.1)$$

$$ZCR = \sum_{n=0}^{N-1} \frac{1 - \mathrm{sgn}[x(n+1)]\,\mathrm{sgn}[x(n)]}{2} \qquad (2.2)$$

where $\hat{E}(n)$ is the pseudo-energy of the input signal x(n), w(m) is the weighting window, N is the window size, and ZCR means zero-crossing rate.

The energy is mainly used to discriminate voiced/unvoiced regions of speech. We used pseudo-energy instead of energy as it is less sensitive to high signal levels.

Zero-crossing rate is also a very useful parameter for speech detection. A zero-crossing may be mathematically described by:

$$\mathrm{sgn}[x(n+1)] \neq \mathrm{sgn}[x(n)] \qquad (2.3)$$

where

$$\mathrm{sgn}[x(n)] = \begin{cases} +1 & \text{if } x(n) \geq 0 \\ -1 & \text{if } x(n) < 0 \end{cases} \qquad (2.4)$$

The proposed algorithm can choose the parameters corresponding with table 1.

We have to note that all the options but FILTERING can be *either/or* employed; FILTERING option can be *or* employed.

The figure 2 shows the principles used for start/end point detection.

Feature class	Options
SAMPLING RATE	8/16 kHz
QUANTIZATION	8/16 bit
WINDOW LENGTH	16/32/64 msec
FRAME LENGTH	8/16/32 msec
WINDOW FUNCTION	Dirichlet/Hamming
SPEECH PRE-AMPLIFICATION	yes/no
SPEECH PRE-EMPHASIZE	yes/no
ENERGY	energy/pseudo-energy/2nd autocorrelation coefficient
ENERGY LOG-COMPRESSION	yes/no
FILTERING	ENERGY low-pass filtering/ ZCR low-pass filtering
ZERO-CROSSING RATE GAP	1/3/5
THRESHOLDS	any float numbers
read from stdin	non-blocking/blocking

Table 1. Algorithm options

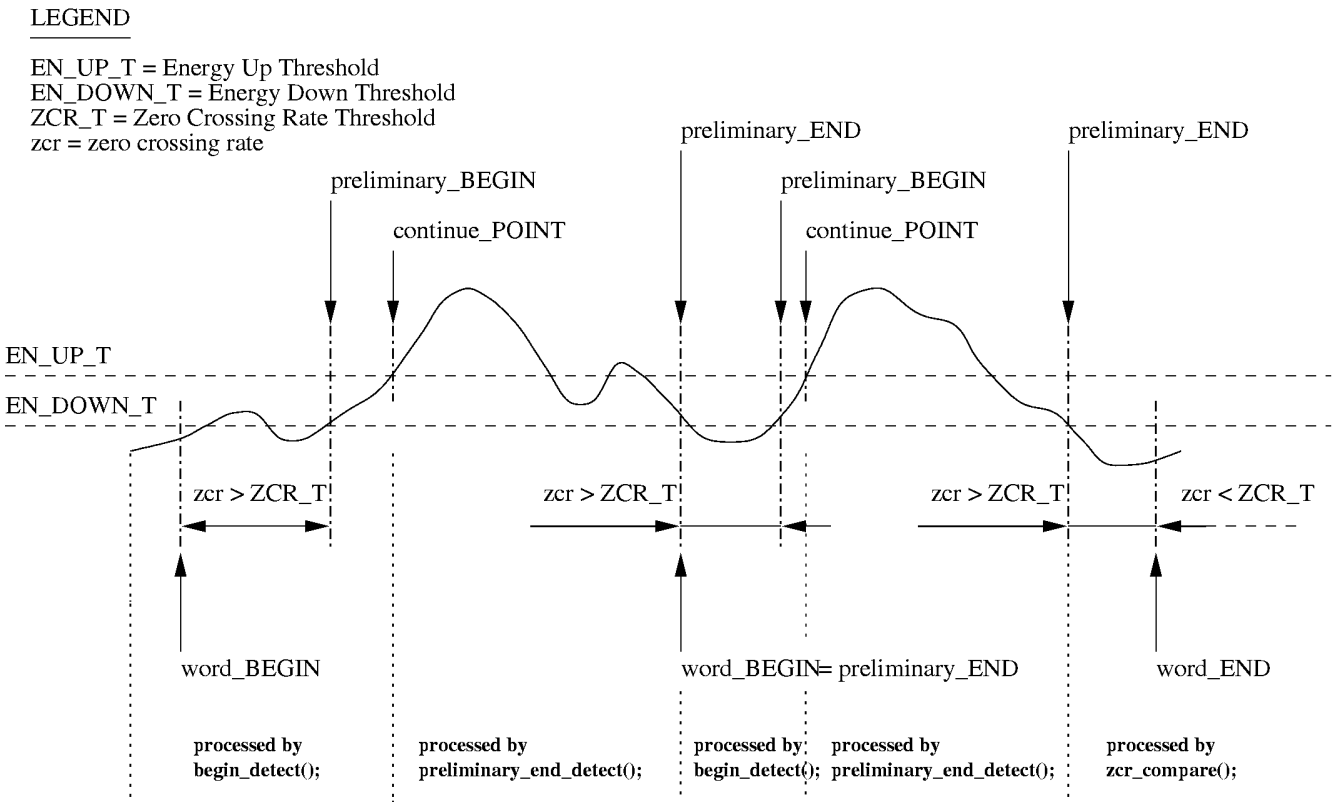

Fig. 2. The algorithm used for start/end point detection.

3. Software implementation details

The program we propose is actually a filter: it reads from standard input and writes to standard output. As a consequence, the program may be easily included in a shell script. The flowchart of the main subroutine is shown in figure 3.

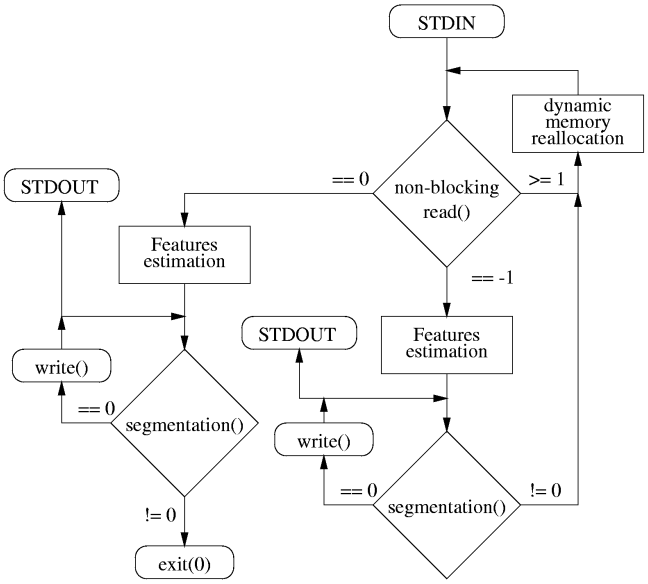

Fig. 3. The flowchart of the main subroutine.

It includes a non-blocking loop to read data from a pipe (stdin). To account for different OSes, we chose the buffer size argument of read() function to be equal to _POSIX_PIPE_BUF value [8]. When data is not present in the pipe, segmentation subroutine is called in order to classify between speech and noise. The segmentation loop exits when there are no words to extract any more. Then the program dynamic reallocs to update the storage requirements and goes back to the non-blocking read.

The flowchart of the segmentation subroutine is shown in figure 4. It includes callings of begin_detect() and end_detect() subroutines.

The main idea is if no word_BEGIN is found, the buffer storing the acoustic vectors is truncated in order to remove the noisy region. But if a word_BEGIN is found, a word_END detection is subsequently performed. If there are not enough data to allow word_END detection decision after a word_BEGIN was found, the switch is set up in order to bypass begin_detect() sequence on next calling of segmentation().

In order to facilitate the integration of the proposed software into native parallel data acquisition system such as a multi-microphone one, the routines were carefully written to provide synchronization between threads [3][4].

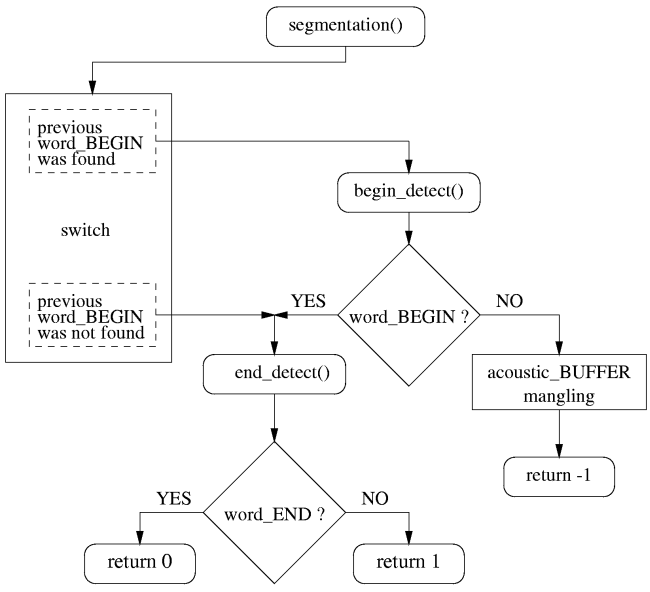

Fig. 4. The flowchart of the segmentation subroutine.

Two mutual exclusion locks (mutex) were defined at the level of memory dynamic reallocation (fig. 3) and begin_detect() and end_detect() subroutines (fig. 4). Therefore, the start/end point classification decisions and consequently memory reallocations are performed in a flexible way.

4. Results and conclusions

Our algorithm was implemented at the pre-processor level of two different systems: an isolated-word speaker-independent recognition system [2] and a voice dialing system in Romanian [7].

The word detection algorithm runs with no problems for isolated-spoken words. However, the problem of start/end point detection is by far more difficult for a speaker-independent voice dialer application. The algorithm should not split digits including, for example, unvoiced plosives (p, t) with large silence regions. Also, the algorithm should not link digits when it is hard to distinguish the silence period between words, from surrounding sounds. In figure 5 we show such a difficult combination.

Therefore, our word detection algorithm performs well for digits separated by silence no shorter than 300 msec.

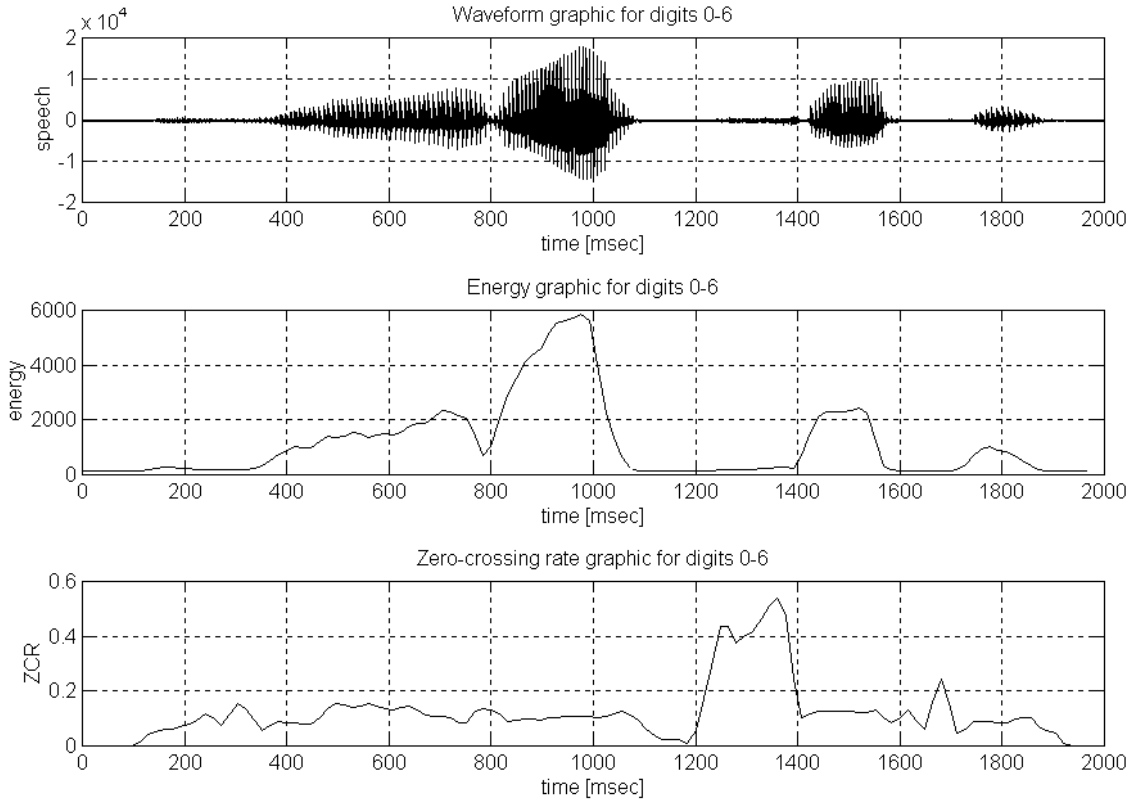

Fig. 5. Waveform, energy and ZCR graphics for “zero-six”

An important effort was directed to identify the optimum choice at different stages of software development; for example, window and frame durations, filtering parameters, integer arithmetic instead of floating point one, use of POSIX and ANSI C standards.

The performances of the start/end point detection algorithm described in this paper allowed for a robust, speaker independent, speech recognition system.

References

[1] Burileanu, C., "Contribuţii la realizarea sistemelor logice programate cu aplicaţii în electronica funcţională (Recunoaşterea automată a vorbirii)", *PhD Dissertation*, Polytechnical Institute of Bucharest, Bucharest, 1986.

[2] Burileanu, D., M. Sima, C. Burileanu, V. Croitoru, "A Neural Network-Based Speaker-Independent System for Word Recognition in Romanian Language", *Proceedings of the "Text, Speech, Dialog" Conference*, Brno, Czech Republic, pp. 177-182, 1998.

[3] McCarthy, M., "What is Multi-Threading ?", *Linux Journal*, Issue 34, pp. 31-40, 1997.

[4] McCarthy, M., "Thread-Specific Data and Signal Handling in Multi-Threaded Applications", *Linux Journal*, Issue 36, pp. 45-52, 1997.

[5] Rabiner, L. R., M. R. Sambur, "An Algorithm for Determining the Endpoints of Isolated Utterances", *The Bell System Technical Journal*, Vol. 54, No. 2, pp. 297-315, 1975.

[6] Sima, M., V. Croitoru, D. Burileanu, "Performance Analysis on Speech Recognition using Neural Networks", *Proceedings of the International Conference on Development and Application Systems*, Suceava, Romania, pp. 259-266, 1998.

[7] Sima, M., D. Burileanu, V. Croitoru, C. Burileanu, "The Application of Neural Network Paradigms in Speech Recognition for a Romanian Voice Dialing System", *Proceedings of Communications-98*, Military Technical Academy, Bucharest, pp. 233-238, 1998.

[8] Stevens, W. R., *Advanced Programming in the UNIX® Environment*, Addison-Wesley, Reading, Massachusetts, 1992.