H-0199 October 22, 2003
Engineering and Technology

IBM Research Report

The ETSI Extended Distributed Speech Recognition (DSR) Standards: Client Side Processing and Tonal Language Recognition Evaluation

Alexander Sorin[1], Tenkasi Ramabadran[2], Dan Chazan[1], Ron Hoory[1], Michael McLaughlin[2], David Pearce[3], Fan C. R. Wang[4], Yaxin Zhang[5]

[1]IBM Research Division
Haifa Research Laboratory
Haifa 31905, Israel

[2]Motorola Labs.
Schaumburg, USA

[3]Motorola Labs.
Basingstoke, UK

[4]IBM Research Division
China Research Laboratory
Beijing, China

[5]Motorola Labs.
Shanghai, China

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

THE ETSI EXTENDED DISTRIBUTED SPEECH RECOGNITION (DSR) STANDARDS: CLIENT SIDE PROCESSING AND TONAL LANGUAGE RECOGNITION EVALUATION

Alexander Sorin[1], *Tenkasi Ramabadran*[2], *Dan Chazan*[1], *Ron Hoory*[1], *Michael McLaughlin*[2], *David Pearce*[3], *Fan CR Wang*[4] *and Yaxin Zhang*[5]

[1] IBM Research Labs, Haifa, Israel; [2] Motorola Labs, Schaumburg, USA; [3] Motorola Labs, Basingstoke, UK; [4] IBM Research Labs, Beijing, China; [5] Motorola Labs, Shanghai, China

ABSTRACT

In this paper we present work that has been carried out in developing the ETSI Extended DSR standards ES 202 211 and ES 202 212 [1][2]. These standards extend the previous ETSI DSR standards: basic front-end ES 201 108 and advanced (noise robust) front-end ES 202 050 respectively. The extensions enable enhanced tonal language recognition as well as server-side speech reconstruction capability. This paper discusses the client-side estimation of pitch and voicing class parameters whereas a companion paper discusses the server-side speech reconstruction. Experimental results show enhancement of tonal language recognition rates of proprietary recognition engines, when the standard extensions are used.

1. INTRODUCTION

The European Telecommunication Standards Institute (ETSI) STQ Aurora group has published two distributed speech recognition (DSR) standards in the years 2000-2002 [3]. The basic front-end, as well as the noise robust advanced front-end define feature extraction and compression on a mobile terminal. The compressed features are transmitted to a server for recognition back-end processing.

The front-end standardization process included recognition tests performed in several European languages, as well as American English. It is well known, however, that for some Asian languages such as Mandarin, Cantonese and Thai, recognition accuracy can be enhanced by introducing tonal information in addition to the spectral information [4], [8]. To promote the use of the ETSI DSR standards in Asia, the Aurora group has decided to extend the existing DSR standards to include extraction and compression of tonal information. The two extended standards (extended basic and advanced front-ends) also enable server-side speech reconstruction using the Mel-Cepstral features (MFCC) extended by the tonal information. The reconstruction algorithm is an integral part of the standards, and is discussed by a companion paper. The development of the extended standards has been carried out during the years 2002-2003, jointly by IBM and Motorola.

This paper deals with the client side processing, server-side post-processing of the tonal information and using the standard tonal parameter in tonal language recognition evaluation.

2. FRONT-END PROCESSING AT THE CLIENT

The overall complexity and memory requirements of the extended front-ends do not exceed those of the GSM-AMR speech encoder; the bit rate addition is 800 bps, resulting in a total bit-rate 5600 bps and the delay is identical to the delay of the basic and advanced front-ends.

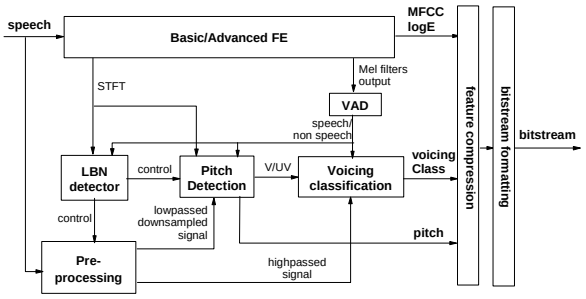

Figure 1: Front-end processing block diagram

In figure 1, the basic/advanced FE block represents the feature extraction component of the non-extended front-ends, producing MFCC and log energy information. In addition, two intermediate data structures are obtained from the basic/advanced feature extraction – the short time Fourier transform (STFT) and the outputs of the Mel filters. These are reused for the computation of the extension data.

For every 10 msec speech frame, the extension data, comprising the pitch and the voicing class, is computed and encoded together with the MFCC and logE data. We shall now describe the extension blocks presented in figure 1.

2.1. Voice Activity Detection (VAD)

Voice Activity information is used to enhance the pitch estimation robustness under noisy conditions. It is also incorporated as part of the voicing class information and can be used for segmentation (or end-point detection) of the speech data for improved recognition performance. The voice activity detector is based on the outputs from the mel-filter bank, which are computed in the MFCC feature extraction process.

2.2. Efficient Robust Pitch Detection

Pitch is estimated for frames classified by the VAD as speech. The algorithm consists of three major stages: 1) selection of pitch candidates with corresponding scores by frequency-domain analysis; 2) computation of time-domain correlation scores; 3) selection of the final pitch estimate among the candidates.

Usually, pitch estimation is sensitive to low frequency band noise (LBN) with significant spectral components inside the frequency diapason where the pitch frequency (F_0) search is performed. LBN is often present in the passenger compartment of a moving or idling automobile, thus severely limiting the applicability of pitch estimation methods in mobile environments. In order to overcome this problem LBN detection is performed at each frame preceding the pitch estimation. The (binary) decision of the LBN detector is passed to the pitch estimator in order to appropriately modify some of its parameters on the fly as described below.

2.2.1. Low-frequency Band Noise Detection

The LBN detection is performed by power spectrum analysis for non-speech frames. A ratio R_{curr} is computed of the maximal spectrum value in the diapason [*0, 380Hz*] to the maximal spectrum value in the complementary high frequency diapason. An integrative LBN contribution measure R is updated as $R = 0.99*R + 0.01*R_{curr}$. LBN is detected if $R>1.9$. The integrative measure R is initialized to 1.9.

2.2.2. Generation of Pitch Candidates

The algorithm is based on spectral peaks analysis and inherits the main ideas described in [5] with some modifications aiming to limit the amount of computations being performed in the worst case. The STFT computed by the DSR front-end is reused as an input for this stage.

The STFT frequency resolution is doubled by Dirichlet interpolation, and the absolute values are computed. The N highest spectral peaks are determined by finding the highest local maxima on the discrete spectrum, where $N \le 20$. Only peaks associated with frequency greater than 300Hz are considered if LBN has been detected. Then the peak amplitudes are normalized so that their sum is equal to 1. The set of N peaks inspires a utility function $U(F_0)$ of all possible pitch values within a search range $F_{0min} \le F_0 \le F_{0max}$:

$$U(F_0) = \sum_{i=1}^{N} A_i \cdot I(f_i / F_0) \qquad (1)$$

where A_i and f_i are respectively the amplitudes and frequencies of the spectral peaks, and I(r) is a periodic piecewise constant influence function with period R=1 defined as follows:

$$I(r) = \begin{cases} 1, & |r| \le 65/512 \\ 0.5, & 65/512 < |r| \le 100/512 \\ 0, & 100/512 < |r| < 0.5 \end{cases} \qquad (2)$$

Each additive component of the sum (1) has local maxima in the vicinity of the integer dividers of the corresponding peak frequency f_i. The pitch candidates are selected among the local maxima of the utility function.

An approximate algorithm with guaranteed worst case complexity is employed for finding all local maxima of $U(F_0)$. The M highest peaks (M=7) are selected among the N peaks and sorted in a descending order of their amplitudes. For every peak f_i all the break-points of the piecewise constant *individual utility function* $A_i \cdot I(f_i / F_0)$ are determined. *The* process continues until either all M peaks are processed or the number of break points (and therefore the amount of computations) exceeds a predefined limit. Then the individual utility functions built for $K \le M$ peaks are merged together representing a *preliminary utility function*. The four highest local maxima of the preliminary utility function are selected. Their utility values are recomputed directly in accordance with (1), using all N peaks. Then, two pitch candidates are selected among the four maxima. The selection process gives preference to candidates with high utility value, high F_0 and F_0 close to the one found at the previous frame. Each candidate comprises an F_0 value along with a corresponding spectral score (the utility value).

In order to have a more uniform distribution of the pitch candidates over the pitch search range, the above procedure is independently applied to 3 overlapping pitch search intervals: S_1=[52Hz,120Hz], S_2=[100Hz,210Hz], S_3=[200Hz,420Hz]. If the pitch estimates obtained at the previous 6 frames are close enough to each other, then S_1, S_2 and S_3 are intersected with a narrowed search range [$0.666*F_{0prev}$, $2.2*F_{0prev}$], where F_{0prev} is the pitch estimate at the previous frame. For each of the intersections W_1, W_2 and W_3, two pitch candidates are generated, provided it is non-empty.

For the low frequency sub-range W_1, representing long pitch periods, the window used for computing the STFT is too short for a reliable estimation. A new, approximated STFT is generated from the current and previous STFTs according to:

$$STFT(2\pi f) = STFT_{prev}(2\pi f) \cdot e^{j\Delta 2\pi f} + STFT_{curr}(2\pi f) \qquad (3)$$

where Δ is the frame shift.

The pitch search intervals are processed in the order W_3, W_2, W_1. For each pitch candidate a correlation score is computed as described below (2.2.3). If the candidate's spectral score and correlation score are close to 1 the candidate is declared as the pitch estimate and the process terminates. Otherwise all pitch candidates are passed to an estimate selection block where less restricted constraints are imposed on the candidate scores, as described in 2.2.4.

2.2.3. Correlation Scores Calculation

Correlation is computed for each pitch candidate using a low pass filtered and downsampled version of the speech signal. The filtering is done by a 6-th order IIR Butterworth filter with a cut-off frequency of 800Hz. If no LBN has been detected then the low pass filter is combined with a first order IIR low frequency emphasis filter. Downsampling factors of 4, 5 and 8 are used for sampling rates of 8kHz, 11kHz and 16kHz respectively. An F_0 value associated with a pitch candidate is transformed to a time lag divided by the downsampling factor. Long enough fragment of the downsampled signal is stored in order to enable processing of large lags. In general, the lag is fractional, and the method described in [6] is used for computing the correlation.

2.2.4. Final Decision Making

A heuristic logical scheme is used for selecting the final pitch estimate. The scheme has a decision tree form, and tests various conditions involving absolute values of the spectral and correlation scores, relative score values of the candidates, and pitch information from previous frames. If none of the conditions is satisfied the frame is declared as unvoiced.

2.3. Voicing Classification

There are four possible voicing classes: non-speech, unvoiced, mixed-voiced and fully-voiced. The mixed-voiced class was defined in order to improve the quality of the server-side speech reconstruction.

Classifying a frame as non-speech or unvoiced is carried out by the VAD and the pitch detector respectively. An additional classification process is required, in order to distinguish between the two voiced classes. This is done by checking the energy of the high frequency band as well as a zero-crossing measure.

2.3. Data Compression

For the pitch information compression, the frames are coupled into pairs. Pitch values are quantized with 12 bits per pair, using both absolute and differential quantization. Reference values for the differential quantization are chosen dynamically in a way that limits the spreading of channel errors effect. One additional bit per frame is required for the voicing class. The class index and pitch index jointly represent the actual voicing class, thus enabling 4 classes, according to table 1:

Voicing class	Pitch index	Class index
Non-speech	0	0
Unvoiced-speech	0	1
Mixed-voiced speech	> 0	0
Fully-voiced speech	> 0	1

Table 1: Class quantization

Two CRC bits are added per frame pair. Thus, for a frame pair (20 msec), a total of 16 bits is required and the bit-rate increase required for the extension is 800 bps.

3. PITCH TRACKING AT THE SERVER

While client side pitch detection is subject to strict requirements in terms of computation load and delay, this is not the case for the server side processing. Thus, it is possible to further refine the raw pitch contour, decoded at the server, by a pitch tracking mechanism that utilizes a look-ahead of several speech frames. The objective of the pitch tracking process is to produce a continuous pitch track and remove pitch and voicing errors. Pitch tracking is beneficial for both tonal language recognition and for speech reconstruction and is therefore an integral part of the standard.

The input to pitch tracking module is a set of successive pitch period values and corresponding voicing classifications and log energy values, available in both basic and advanced Front-End standards. The outputs are the corrected pitch values and corrected voicing classifications.

The pitch tracking process is carried out in two stages. In the first stage, gross pitch and voicing error are corrected and in the second the pitch contour is smoothed.

Gross pitch or voicing classification errors are corrected by examining the neighbors of a frame. For example if a frame was classified as voiced and its neighbors were classified as unvoiced, it is reclassified as unvoiced.

In other cases, the pitch F_0 of a voiced frame may be multiplied or divided by an integer. Such a decision is made by first determining a reference pitch value F_{0ref} for the voiced segment surrounding the voiced frame. The reference pitch represents the most energetic series of consecutive frames with similar pitch values within the segment. If F_{0ref} and F_0 are distant then F_0 is multiplied or divided by an integer that minimizes their distance.

The smoothing for a voiced frame is carried out by performing a weighted average of the pitch values of neighboring voiced frames. Before the weighted average operation unvoiced frames within the averaging interval are assigned the pitch value of the middle frame. For other voiced frames the pitch values is multiplied or divided by an integer, such as they become as close as possible to the pitch of the middle frame.

4. TONAL LANGUAGE RECOGNITION

4.1. Pitch Contour Preprocessing

In order to obtain the most effective and robust tonal language recognition the pitch contour (either proprietary or standard, after pitch tracking) is further processed. This is not a part of the standard and is usually specific to each system. Here, we shall briefly describe the proprietary preprocessing applied by the two systems examined.

In the IBM tonal language recognition system, the preprocessing consists of converting the pitch values to the log-frequency domain and dynamic smoothing.

Given a voicing classification and pitch frequency value $F_0(n)$, the computation of the tonal parameter $TP^{(IBM)}(n)$ is done as follows:

$$LP(n) = \begin{cases} \log F_0(n) - \overline{LP}(n) & voiced \\ \beta \cdot TP^{(IBM)}(n-1) + N(n) & unvoiced \end{cases} \quad (4)$$

$$TP^{(IBM)}(n) = (1-\gamma)LP(n) + \gamma \cdot TP^{(IBM)}(n-1) \quad (5)$$

where $\overline{LP}(n) = \alpha \cdot \overline{LP}(n-1) + (1-\alpha) \cdot \log F_0$,

$N(n)$ is a uniformly distributed random noise, and α, β and γ are constant parameters.

In the Motorola tonal language recognition system, the tonal parameter is computed as follows:

$$TP^{(Motorola)} = \frac{1}{2}\left[F_0(n) + F_0(n-1) - F_0(n-2) - F_0(n-3)\right] \quad (6)$$

4.2. Experimental Results

Both IBM and Motorola used CUDIGIT corpus as a part of the evaluation. CUDIGIT is a Cantonese continuous digit string database publicly available from the Chinese University of Hong Kong. It contains spoken Cantonese digit strings, recorded by 50 speakers with a sampling rate of 8 kHz. For each digit string, four signals were artificially generated: clean, car noise (10dB), street noise (15dB) and babble noise (15dB). The training corpus involves 38 speakers (total 21640 sentences). And the testing corpus has another 12 speakers (total 6835 sentences).

The IBM rank-based continuous speech recognition system [7] uses a phonetic representation for every word in the vocabulary. Each phone is modeled with a three-state, HMM

(Hidden Markov Model), and acoustically dissimilar variants of each state are identified using a decision tree structure. The IBM proprietary pitch contour is extracted via time-domain autocorrelation based pitch detection. The tonal parameter is computed as in (5). The acoustic vector combines 12 MFCCs and the tonal parameter, augmented with the first and second temporal derivatives of the first 13 coordinates.

In addition to the common CUDIGIT database, CNDIGITS - an IBM proprietary Mandarin database was used for the evaluation. CNDIGIT is a Mandarin telephony digit string database containing both landline and mobile recordings, all in 8 kHz. The training corpus has 3994 speakers (total 12742 sentences), while the testing corpus uses another 120 speakers (total 1729 sentences).

CUDIGIT					CNDIGIT
clean	car	street	babble	entire set	
-16.15	+5.65	+17.50	+14.14	+9.52	+1.30

Table 2: %Improvement in recognition accuracy when using standard ETSI pitch instead of IBM pitch

Table 2 shows the improvement in recognition accuracy (measured by word error rate) when replacing the IBM pitch by the ETSI pitch. For both cases, acoustic model multi-style training was performed in advance with the same features as in the testing. Overall, it can be seen that the ETSI pitch has an advantage over the IBM pitch in noisy situations.

Motorola used its HMM-based recognition engine with the MLite++ search engine configuration as the evaluation test bench. A proprietary, frequency domain, autocorrelation based, pitch detection module [8] is used, and the tonal parameter is computed as in (6). The acoustic vector combines 12 MFCCs and the tonal parameter augmented with the first temporal derivative of the first 13 coordinates.

In addition to the CUDIGIT database, another proprietary Mandarin digit database was used for the evaluation. The database contains 8 kHz close-talking microphone digit strings recording, with 182 speakers (total 8699 sentences) in the training set and 170 speakers (total 7509 sentences), recorded in seven different environments, in the test set. Tables 3 and 4 summarize the evaluation results:

clean	Car	street	babble	entire set
+13.30	+18.98	+20.55	+9.39	+15.97

Table 3: %Improvement in recognition accuracy when using ETSI pitch instead of Motorola pitch (CUDIGIT)

office	airport	Car	park	mall	station	street	entire set
+2.00	+2.86	+2.47	+1.45	+2.16	-2.87	+1.59	+1.31

Table 4: %Improvement in recognition accuracy when using ETSI pitch instead of Motorola pitch (Mandarin digits)

Both table 3 and table 4 show the improvement in recognition accuracy (measured by word error rate) when replacing the Motorola pitch by the ETSI pitch. In all cases acoustic model multi-style training was performed in advance, with the same features as in the testing. Overall, it can be seen that the ETSI pitch outperforms the Motorola pitch in both clean and noisy situations.

5. CONCLUSIONS

With a minimal bit-rate increase, enhanced tonal language recognition can be obtained in the DSR environment by using the extension of the standard ETSI DSR front-ends. The tonal parameter extracted using the extended DSR features is more noise robust than proprietary tonal parameters.

In addition to tonal language recognition, the pitch information available at the server can be used for speech reconstruction (also part of the standards). Another possible usage of the pitch information is for performing emotion detection.

6. REFERENCES

[1] ETSI ES 202 211, “Speech Processing, Transmission and Quality aspects (STQ); Distributed speech recognition; Extended front-end feature extraction algorithm; Compression algorithms; Back-end speech reconstruction algorithm,” 2003.

[2] ETSI ES 202 212, “Speech Processing, Transmission and Quality aspects (STQ); Distributed speech recognition; Extended advanced front-end feature extraction algorithm; Compression algorithms; Back-end speech reconstruction algorithm,” 2003.

[3] Pearce D., "Enabling new speech driven services for mobile devices: An overview of the ETSI standards activities for distributed speech recognition front-ends", in proc. American Voice I/O Society (AVIOS), San-Jose, 2000

[4] Chen C.J., Li H., Shen L. and Fu G., "Recognize tone languages using pitch information on the main vowels of each syllable", in Proc. . ICASSP, Salt Lake City, 2001

[5] Chazan, D., Zibulski M., Hoory, R. and Cohen, G., "Efficient periodicity extraction based on sine-wave representation and its application to pitch determination of speech signals", in Proc. Eurospeech, Aalborg, 2001

[6] Y. Medan, E. Yair and D. Chazan, "Super resolution pitch determination of speech signals", IEEE Trans. Acouts., Speech and Signal Processing, vol. 39, pp.40-48, Jan. 1991

[7] Bahl L.R., de Souza P.V., Gopalakrishnan P.S., Nahamoo, D. and Picheny M.A., "Robust methods for using context-dependent features and models in a continuous speech recognizer", in Proc. ICASSP, Adelaide, 1994

[8] Tan H. and Zhang Y., "Mandarin tone recognition in embedded speech recognition systems", in Proc. of the Thirteenth Australian Language and Speech Conference (ALSC), Sydney, 2001