REAL-TIME IMPLEMENTATION OF HMM-BASED MMSE ALGORITHM FOR SPEECH ENHANCEMENT IN HEARING AID APPLICATIONS

H. Sheikhzadeh[1], *R. L. Brennan*[2], *and H. Sameti*[1]

[1]Dept. of Elec. & Comp. Eng., University of Waterloo, Waterloo, Ont., Canada N2L 3G1,
[2]Unitron Industries Ltd., 20 Beasley Drive, P.O.Box 9017, Kitchener, Ont., Canada N2G 4X1

ABSTRACT

In this paper we describe our recent work on real-time implementation of a state-of-the-art HMM-based MMSE speech enhancement algorithm, where our earlier published algorithm [2] has been approximated, optimized, and simplified. The key innovations enabling the enhancement system to run in real-time are: 1) algorithm for automatic selection of the noise model, 2) pruning the MMSE forward calculations, 3) new pause detection method operative for the SNR down to 0 dB, and 4) task partitioning of the entire system, all developed from our more recent work. A preliminary version of the real-time enhancement system is simulated on IBM-PC 486DX2/66 and at the same time implemented on a DSP platform based on dual TMS320C30 DSP chips using single precision floating point arithmetic.

1. INTRODUCTION

Under noisy conditions, hearing impaired persons typically have greater difficulty understanding speech than those with normal hearing. This disadvantage translates to the requirement of an additional 2.5 and 12 dB SNR improvement for speech discrimination scores similar to those of normal hearing. As the capabilities of digital signal processing migrate to smaller devices, it is natural to consider its use as a front-end speech enhancement technique for future-generation hearing aids. As a first step, an off-line simulation of a conventional Wiener filter enhancement system was evaluated with encouraging clinical results [1]. This system was later adapted and run in real-time on a dual TMS320C30 DSP system.

To solve some of the short-comings of conventional Wiener-filtering, an off-line simulation of the minimum mean square error (MMSE) enhancement system based on hidden Markov modeling (HMM) of the speech signal was developed recently by our research group. Our extensive subjective and objective evaluations (reported in [2] and [3]), have shown that this system is superior to the HMM-based MAP and the conventional Wiener filtering systems.

As a result of the success of our previous work, we concluded that the MMSE-based system be adapted for real-time implementation and be further improved and refined to suit a DSP platform. The main improvements of the system achieved to date are two-fold: 1) Approximation and optimization of the operations involved, and 2) Introducing a robust pause detection algorithm to be used for noise model selection under various types and levels of noise. Further, we have devised proper partitioning of the various tasks involved in the enhancement process that enables the system to take full advantage of the capabilities of the dual DSP chips in our hardware platform.

2. EXTENSIONS TO THE MMSE METHOD

An improved HMM-based MMSE enhancement system, based on the algorithm first described in [4], was developed. As a basis for our system, we adopted the enhancement system detailed in [3], which incorporates a multiple state and mixture noise model to accommodate noise nonstationarity more effectively.

There are various types of noise in the environment with very different statistical spectral characteristics. It is always an advantage for the enhancement system to have *a priori* knowledge about the nature of the noise. Enhancement methods which make assumptions about the noise type are deficient in terms of functionality under various corrupting noise types. The HMM-based enhancement systems are inherently relying on the type of training data for noise. Expectedly, such a system can handle only the type of noise which has been used for training the noise HMM. Therefore, data from various noise types should be used for training the noise model. This creates the problem of a large model size for the noise HMM, where the search space expands linearly with the number of noise types and the computation cost grows accordingly. In addition to the growth of computation, the unwanted large search space deteriorates the system performance by introducing more sources of error in the MMSE forward algorithm. For the real-time implementation, we developed a novel noise adaptation algorithm. The method is devised to simultaneously 1) enable the system to handle arbitrary types of corrupting noise, and 2) avoid up-growth in computation complexity and preserve the real-time implementation capability of the model.

With a real-time implementable system as an objective, our extended MMSE enhancement system is illustrated in the block diagram of Figure 1.

Following directly the mathematical operations involved in MMSE speech enhancement would incur a very high computation cost. To solve this problem for the purpose of real-time implementation, we devised two major optimization attempts, to be described in the next two sections, by which the computer program's execution time dropped considerably.

0-7803-2431-5/95 $4.00 © 1995 IEEE

3. DOUBLE PRUNING OF THE MMSE FORWARD CALCULATION

The computation of the MMSE forward probability and filter weights is the most time consuming part of the system. For speech and noise HMMs of sizes $M \times L$ and $N \times P$, respectively, these operations call for calculation of $M \times L \times N \times P$ filter weights and the same number of noisy pdf values for each time frame t. In the forward algorithm, we prune the forward trellis search through two approaches as described below.

The first approach stems from the observation that the majority of the noisy pdf values for a frame are negligibly small. Thus using all the pdf values would lead to a great amount of redundant calculations in the forward algorithm. For the noisy pdf calculation we use the log domain to avoid underflow. This at the same time makes the pdf calculation more efficient than directly calculating the pdf values, since the forward probabilities are also calculated in the log domain. As will be described in Section 4, the operations involved in finding the log pdf values can be greatly optimised. As a result, finding all the log pdf values for a frame is not very costly. Next, the maximum value of the log pdf for each frame was deducted from all log pdf values to normalise their range to $[0, -\infty)$. This does not affect the final results since the output pdfs appear both in the numerator and the denominator of the equation for the filter weight calculations [3]. In the MMSE forward algorithm, a low threshold set empirically is used to eliminate those nodes in the trellis having the log pdfs below the threshold. It should be noted that due to the presence of various levels and various types of noise, use of an absolute (rather than relative) lower threshold would not be practical. In our simulations on PC, we use a linked-list structures to save the log pdfs and the associated indices to the speech and noise models. However, for real-time implementation of the enhancement algorithm, this would not be acceptable since the computation would have to be data independent. Instead, we use an array of such structures with a hard limit on the maximal number of acceptable log pdf values for each frame.

The second approach is based on the observation that even after pruning the log pdfs, many of the forward probabilities are still negligible. In the forward algorithm, we calculate all the forward probabilities (after the log pdf pruning) for each frame. At the same time, for each frame, we find the total value of the forward probabilities for the filter weight calculations (see [4] and [3] for more details). We prune out the nodes with the forward probabilities below a pre-determined threshold relative to the total value. Again, no absolute threshold is used. Similar to the log pdf calculations, we use an array of structures to save the forward probabilities (and the indices). A hard limit is also put on the maximum number of preserved nodes per frame. The above pruning not only speeds up the forward algorithm, but also affects the Wiener filtering process. The Wiener filter weights are only calculated for the preserved nodes. This implies that only a small fraction of the Wiener filters are used to make the overall filter which is the weighted sum of the surviving individual filters.

As a result of the double pruning described above, the computation cost of the enhancement process becomes independent of the input data, and independent of the size of the speech and noise HMMs as the number of the saved filter weights are independent of the model sizes. Without this pruning, however, the computation cost would increase proportionally with the speech and noise model sizes.

4. APPROXIMATING PDF OF NOISY SPEECH

Calculation of the noisy Gaussian pdf (i.e. the pdf for noisy speech as input to the enhancement system) is computationally costly because of the need to inverse covariance matrices of size $K \times K$ ($K = 256$ in our system) and of the need of multiplication of the matrices with an equally large size. Since the summation of two AR processes is not necessarily an AR process, decomposition of the inverse of the covariance matrix of the noisy speech, Σ_{z_t}, into two Toeplitz matrices comprised of AR coefficients of noisy speech z_t is not generally applicable. In order to avoid the expensive calculation for the log pdfs of noisy speech due mainly to large matrix inversion and multiplication, an approximate method is devised.

To find the log pdfs, the autocorrelation coefficients of the clean speech and noise are calculated from their AR coefficients. Assuming additivity and independence of the noise and the clean speech signal, their autocorrelation coefficients are added to form the autocorrelation coefficients of the corresponding noisy speech. The Levinson-Durbin recursion is performed on the calculated autocorrelation coefficients to find the AR parameter set of the noisy speech, $a_p \triangleq \{a_p(0), a_p(1), \ldots, a_p(p)\}$, $a_p(0) = 1$ and gain σ^2. (The AR order for the noisy speech is chosen to be higher than that in either the speech or the noise model.) Given an observation vector size K, with $K \gg p$, the log pdf can be approximated by $b(z_t) = -\alpha/(2\sigma^2) - (K/2)log(2\pi\sigma^2)$, where $\alpha \triangleq r_t(0)R_p(0) + 2\sum_{m=1}^{p} r_t(m)R_p(m)$. $r_t(m)$ is simply the autocorrelation sequence of z_t and $R_p(m) \triangleq \sum_{n=0}^{p-m-1} a_p(n)a_p(n+m)$.

For real-time implementation, we calculate the autocorrelation sequence $R_p(m)$ for all the models off-line. This leaves the real-time computation burden only on the terms $r_t(m)$ and α. As a result, the computation cost for the pdf of noisy speech is reduced drastically from $O(K^3)$ to $O(K.p)$ (p being the AR order).

5. PAUSE DETECTION ALGORITHM

As discussed in Section 2, the noise model selection algorithm relies on accurate pause detection. Although we are looking only for pauses on the order of a few hundred milliseconds long, detecting non-speech activity is not straightforward due to the effects of noise. To be able to cope with different classes of noise, we adopted a hybrid approach to the problem. Central to our approach to the pause detection problem is an autocorrelation voicing detector algorithm proposed in [5]. Voicing detection was performed on the enhanced signal rather than on the noisy signal. This has the advantage of less noisy estimates of the voiced (V) segments but has the drawback of estimating more unvoiced (UV) segments than it should. We updated the average measure

of the energies of the voiced and unvoiced segments (as by-products of the autocorrelation analysis), respectively, for both noisy and enhanced signals. The ratio of the average unvoiced to voiced energies for the noisy signal (UE/VE) and that for the enhanced signal (UEE/VEE) were used to correct the errors in the voicing decision process. Specifically, for a sequence of unvoiced frames to be accepted as a pause, UE/VE and UEE/VEE had to be lower than an empirically determined threshold. In order to avoid artifacts, we rigorously tested our pause detection algorithm on four different (clean and noisy) sentences from the TIMIT database. Noisy sentences were generated by artificially adding simulated white noise, simulated helicopter noise and recorded multi-talker (babble) noise at SNR's ranging from 0 to 10 dB. The algorithm has been tested and fine tuned extensively until it produces consistent pause detection results for all different noises over all noise levels and for all different sentences.

6. HARDWARE DESCRIPTION AND TASK PARTITIONING

The target DSP platform is based on a dual TMS320C30 system. Each processor has two address spaces accessible through the primary and expansion buses allowing flexible off-chip data transfers. The processors access the majority of their memory through their primary (main) bus. These buses are linked by multiplexers and arbitration logic to allow each processor to access the memory of the other. In addition, the primary buses are also linked to a global memory area containing control registers, the boot program (in EEPROM), and another block of global RAM. Private memory, available on the secondary (expansion) bus, is efficiently used to store slowly changing coefficient information, freeing up the primary bus for high speed data transfers. Communication between the processors is accomplished through interconnected high-speed serial ports and the shared memory areas. Each processor is equipped with 256k words (4-byte wide) of external 0-wait state memory and 2k words of internal memory.

The TMS320C30 implements single precision (4-byte) floating point arithmetic. While a significant advantage over fixed point processors, the HMM algorithm (originally double precision) had to be adapted to work efficiently in this environment. Whenever possible, arithmetic was performed in the 'log' domain. Overflow and underflow had to be strictly prevented. Certain functions such as "log" and "exp" were approximated by Chebyshev polynomial series to avoid the overhead of the math libraries. Trigonometric functions were placed in a lookup table since only a finite number of values were necessary. Memory usage was significantly trimmed with an emphasis placed on the use of dynamic allocation to reduce the static allocation on the limited available stack. Generally, a deterministic design was pursued based on the worst case scenario since processing must always proceed.

The HMM algorithm code was developed in C which was advantageous to our DSP (spectral subtraction) system since the single bus (multi-access) architecture of the 'C30 lends itself well to the compiled C code. An ANSI compliant and reasonably efficient c compiler is available for this purpose. A design philosophy was adopted where C code would be developed with provision for time-critical (FFT routines for example) hand-assembled code. As a result, the C code formed the backbone of the project with sections of assembly code specially written to conform to the C-calling conventions.

Knowing that our C code for the HMM algorithm is portable, we proceeded to optimize its execution speed on a PC 486DX2/66 using the WATCOM C386 compiler v9.5. Of course, a few constraints had to be placed on the code for the DSP environment. One of the first issues on our earlier DSP system was where to place the stack (internal memory or external). During function calls, arguments are placed on the stack and it is advantageous for it to be in internal memory for fast access. Up to three bus accesses may be made per cycle of which only one may be from external memory. If the stack is internal, program instructions may be fetched from external memory with no conflict. Accordingly, a software stack was formed from 512 words of internal memory. Since static allocation (performed on the stack) would quickly exhaust its limited size, an extreme emphasis was placed on dynamic allocation (malloc).

At present, we are able to simulate a near real-time version of our system on a PC 486DX2/66. The simulations have resulted in an execution speed of about one-half real-time and a memory usage of less than 400 Kbytes. The results clearly indicated that the real-time implementation is feasible once proper task partitioning is applied to the enhancement algorithm.

The dual processor DSP platform proves to be well suited to the algorithm implementation. Processor 1 handles the forward FFT and all interrupt driven signal I/O, and pre-processes the signal into the autocorrelation domain of order 16. The autocorrelation coefficients and frequency domain information are then passed to processor 2 in the shared memory space. Processor 2 calculates the noisy pdf, and the likelihood of each Wiener filter prototype (corresponding to the speech and noise models in the pre-trained HMMs) expressed as a series of weights. The frequency domain Wiener filter is then calculated as a weighted sum of these prototype Wiener filters. After multiplication by this filter, the enhanced signal spectrum is transformed back the the time domain by an inverse FFT on Processor 2. This signal is passed back to Processor 1 in the shared memory space. During speech pauses, Processor 2 adapts the prototype Wiener filter according the the new noise characteristics.

7. SUMMARY

In this paper we describe our recent progress of research and development towards real-time implementation of a state-of-the-art HMM-based MMSE speech enhancement system, where the original algorithm has been approximated, optimized and simplified. The key innovations in making the system real-time include the use of a noise-model selection algorithm, pruning of the MMSE forward calculations, a pause detection method operating over the SNR down to 0 dB for white, helicopter and multi-talker noises, and proper task partitioning of the entire enhancement system. A preliminary version of the real-time system is simulated on IBM-PC 486DX2/66 first and then implemented on a DSP

platform based on dual TMS320C30 DSP chips. The experimental results demonstrate that the real-time system simulated on IBM-PC is able to produce speech enhancement performance (in terms of the SNR measure) essentially the same as that implemented on the DSP platform, both are approaching the performance of the system we developed earlier on Unix-based platform [2].

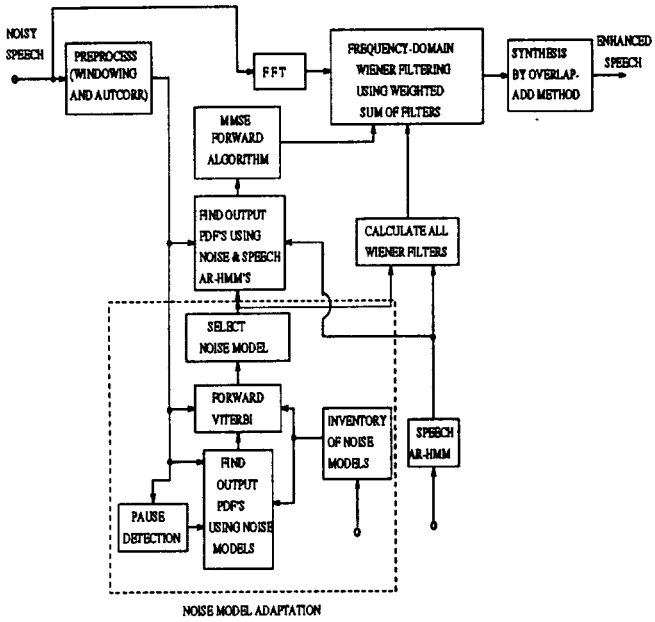

Figure 1. Extended MMSE enhancement system block diagram

REFERENCES

[1] D. G. Jamieson and R. L. Brennan, "Evaluation of Speech Enhancement Strategies for Normal and Hearing-Impaired Listeners", in *Proceedings of the ESCA Workshop on Speech Processing in Adverse Conditions*, (Cannes-Mandelieu, France), pp. 155–158,163, Nov. 1992.

[2] H. Sheikhzadeh, H. Sameti, L. Deng, and R. L. Brennan, "Comparative Performance of Spectral Subtraction and HMM-Based Speech Enhancement Strategies with Application to Hearing Aid Design", in *Proceedings of the ICASSP*, vol. 1, pp. 13–16, Apr. 1994.

[3] H. Sameti, H. Sheikhzadeh, L. Deng, and R. L. Brennan, "HMM-Based Strategies for Enhancement of Speech Embeded in Non-Stationary Noise", *IEEE Transactions on Speech and Audio Processing*, Submitted 1994.

[4] Y. Ephraim, "A Minimum Mean Square Error Approach for Speech Enhancement", *Proceedings of the ICASSP*, pp. 829–832, 1990.

[5] D. A. Krubsack and R. J. Niederjohn, "An Autocorrelation Pitch Detector and Voicing Decision with Confidence Measures Developed for Noise-Corrupted Speech", *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 39, pp. 319–328, Feb. 1991.