THIN CLIENT FRONT-END PROCESSOR FOR DISTRIBUTED SPEECH RECOGNITION

K. F. Chow, S. C. Liew, K. T. Lua

School of Computing, National University of Singapore
3 Science Drive 2, Singapore 117543

ABSTRACT

We present a front-end feature processor for distributed speech recognition for an integer-based DSP, and we employ block floating point and range reduction for the computation of elementary functions. We show that by reducing the numerical accuracy of the block floating point and the elementary functions, we are able to reduce the operational requirements to 12.6 wMOPs, 2.4 kWords of RAM, 3.7 kWords of ROM. When used on a small vocabulary of 800 words 6.4 perplexity, and a large vocabulary of 20,200 words 102.5 perplexity, our optimized DSP front-end produces recognition accuracy comparable to an equivalent implementation on a floating point processor, without requiring a retrain of the recognition system with features produced by our DSP front-end.

1. INTRODUCTION

There have been recent interests in the research community related to distributed speech recognition (DSR). Of particular interest is the Speech, Transmission and Quality Aspects (STQ) group of the European Telecommunications Standards Institute (ETSI), whose work includes the definition of a standard for an advanced front-end processor for distributed speech recognition [1, 2]. They have defined the algorithm for compatibility between the mobile client and the recognition back-end. The algorithm may then be implemented on an integer-based digital signal processor (DSP) at the client.

In DSP programming, there is often a need to trade off between numerical accuracy and power consumption in the DSP. While there have been recent advances in DSP technology that offer hardware floating point units within the processor, integer-based DSPs remain popular for their affordability. An implementation of the front-end feature extraction on an integer DSP for distributed speech recognition should satisfy three conditions: (i) low computational complexity, (ii) low memory consumption, (iii) minimal degradation in recognition accuracy compared to a floating point equivalent implementation. In this paper, we present a front-end feature extractor for distributed speech recognition for an integer-based DSP.

2. IMPLEMENTATION

Figure 1 shows the block diagram of our DSP front-end implementation.

Our DSP front-end accepts a 16 kHz speech signal and divides into frames of 20 milliseconds, with a 10-millisecond overlap. The Fast Hartley Transform (FHT) is used, instead of the Fast Fourier Transform (FFT). Both FHT and FFT produce the same output but FHT is preferred for its reduced complexity [3]. Additive noise is handled with magnitude spectral subtraction as oppose to power spectral subtraction, as magnitude spectral subtraction has been shown [4] to be more robust. Root Cepstral Coefficients (RCC) is implemented using a root transformation of the filter-bank energies in place the traditional MFCC which uses a logarithmic transformation. The RCC is known for its improved robustness under noisy conditions [5, 6]. A discrete cosine transform converts the root-powered filter banks to cepstral features. Convolutional noise is then handled with a technique known as Blind Equalization [7] that is frame-synchronous and performs better than RASTA [8]. At the end of the processing, the 13 features (12 cepstral features, 1 log energy) are quantized and packed into a stream.

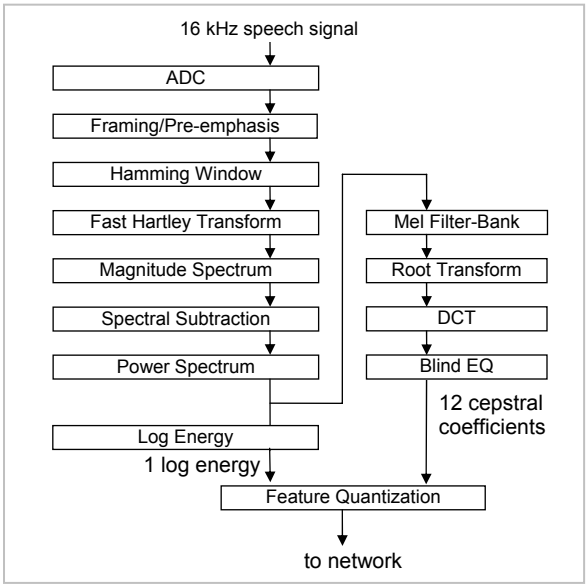

Figure 1: Block diagram of our front-end processor.

3. BLOCK FLOATING POINT

Block Floating Point [9] (BFP) is a technique used in integer processing to compromise speed and numerical accuracy. Clearly, in an integer DSP, we do not have the luxury of hardware-assisted floating point operations for maximum numerical accuracy. The use of fixed point arithmetic, however,

suffers from inaccuracy, due to its limited range. Floating point may be emulated in software in an integer-DSP processor, but assigning an exponent for every mantissa may be too computationally expensive. We may retain the benefits of floating point and keep the computational complexity low, by allowing several mantissas to share a single exponent.

If there are K values $X(0)$, $X(1)$, …, $X(K\text{-}1)$, they may be represented by their separate mantissa and exponent components such that

$$X(i) = m_{X(i)} \times 2^{e_{X(i)}},\ 1.0 \cdot 2^{24} \le m_{X(i)} < 2.0 \cdot 2^{24} \tag{1}$$

Then, in a BFP implementation,

$$\begin{aligned} X(i) &= (m_{X(i)} >> \eta_{X(i)}) \times 2^{e_{X,\sigma}} \\ &= m'_{X(i)} \times 2^{e_{X,\sigma}} \end{aligned} \tag{2}$$

such that the mantissas $m'_{X(\sigma\Lambda)}, m'_{X(\sigma\Lambda+1)}, \ldots, m'_{X(\sigma\Lambda+\Lambda-1)}$ share the same exponent $e_{X,\sigma}$ where

$$e_{X,\sigma} = \max(e_{X(\sigma\Lambda)}, e_{X(\sigma\Lambda+1)}, \ldots, e_{X(\sigma\Lambda+\Lambda-1)}), \quad \forall \sigma \in [0, K/\Lambda) \tag{3}$$

Λ is defined as the size of each block of exponents that share a single exponent and is currently set to 16. An increase in Λ increases the sharing, and hence decreases computational complexity at the cost of reduced numerical accuracy, and vice versa.

BFP is implemented in the stages where the power and magnitude spectrum of the frame of speech is used, where K=256. In particular, it is implemented in the Fast Hartley Transform, Magnitude Spectrum, Spectral Subtraction, Power Spectrum, Log Energy and the Mel Filter-Bank Analysis stages of the block diagram as shown in Figure 1.

4. RANGE REDUCTION FOR ELEMENTARY FUNCTIONS

The elementary functions of logarithm, square-root and the root-power are computed with a table-based approach known as range reduction [10, 11]. In general, the reduction may be defined as:

$$\begin{aligned} y &= f(x) \\ &= g(f(R_{N_f} \ldots R_2 R_1 x), f(R_1^{-1}), f(R_2^{-1}), \ldots, f(R_{N_f}^{-1})) \end{aligned} \tag{4}$$

where $f(ab)=g(f(a),f(b))$ and where the value of N_f indicates the number of reduction steps for elementary function $f(x)$. Depending on the property of the elementary function f, function g may take on the function of a sum or a product of $f(a)$ and $f(b)$. For example, the logarithm of the same form in equation (4) above may be expressed as:

$$\begin{aligned} y &= \ln(x) \\ &= \ln(m_x \cdot 2^{e_x}) \\ &= \ln(R_{N_{\log}} \ldots R_1 m_x) + \ln(R_1^{-1}) + \ldots + \ln(R_{N_{\log}}^{-1}) + e_x \ln 2 \end{aligned} \tag{5}$$

And the square-root and root-power are of the same form and may be expressed as:

$$\begin{aligned} y &= x^{\alpha} \\ &= (m_x \cdot 2^{e_x})^{\alpha} \\ &= (R_{N_{pow}} \ldots R_1 x)^{\alpha} \cdot R_1^{-\alpha} \cdot \ldots \cdot R_{N_{pow}}^{-\alpha} \cdot 2^{e_x \alpha} \end{aligned} \tag{6}$$

where the square-root is a special case where α=0.5. R_i is a reduction function that is a table of S_i number of pre-computed values, and the function may be defined as follows:

$$R_i = r_i(R_{i-1} \ldots R_2 R_1 x) \tag{7}$$

such that the following condition holds:

$$1.0 \le R_i \ldots R_2 R_1 x < 1.0 + 2^{-\sum_{n=1}^{i} S_n} \tag{8}$$

To achieve condition (8), the reduction function r_i may be defined as:

$$r_i(v) = \left[1.0 + \left\lfloor (v - 1.0) \times 2^{\sum_{n=1}^{i} S_n} \right\rfloor \times 2^{-\sum_{n=1}^{i} S_n} \right]^{-1} \tag{9}$$

where the size of each table or r_i is 2^{S_i}. Whether f is the elementary function of logarithm or a root-power, when $R_{N_f} \ldots R_2 R_1 x$ is approximated to 1.0, equation (4) may be re-expressed as

$$y = g(f(R_1^{-1}), f(R_2^{-1}), \ldots, f(R_{N_f}^{-1})) \tag{10}$$

The number of reduction steps have been set for the logarithm, square-root and root-power (for RCC) as N_{log}=2, N_{pow}=2, N_{sqrt}=3 respectively. We set S=8, (where $S=S_i, \forall i$) which indicates the precision of the reduction table r_i. A smaller S or N will result in the decrease in the computational complexity and ROM usage. It is likely, however, that it will result in the decrease the recognition accuracy due to the decrease in numerical accuracy.

5. FEATURE QUANTIZATION

To reduce network bandwidth, we may use the vector quantization techniques defined by ETSI [11] or by IBM [12]. But in favor of a technique that is both low in computational complexity and low in memory usage, we employ a scalar quantization technique that uses a μ-law-like companding function to achieve the quantization. If $C(v) \ge 0$, then the quantized index $C'(i)$ may be defined as follows:

$$C'(i) = \max(2^{L_i - 1} - 1, \frac{\ln\left(1 + \mu_i \frac{C(i) - \overline{C}_i}{C_i^+}\right)}{\ln(1 + \mu_i)} \times (2^{L_i - 1} - 1)) \tag{11}$$

If, however, $C(i) < 0$, then

$$C'(i) = -\max(2^{L_i - 1}, \frac{\ln\left(1 + \mu_i \frac{C(i) - \overline{C}_i}{C_i^-}\right)}{\ln(1 + \mu_i)} \times 2^{L_i - 1}) \tag{12}$$

where μ_i is the curve of the μ-law function, $\overline{C}_i$ is the mean, C_i^+ the positive maximum of $C(i) - \overline{C}_i$, C_i^- the negative maximum of $C(i) - \overline{C}_i$ of the i-th cepstral coefficient. L_i the number of bits that the i-th cepstral coefficient will be quantized with. It should be clear that with this technique, the bit-rate of the compression may be varied only by altering the value of L_i without incurring any penalty in computational complexity or memory usage.

6. EXPERIMENTS

Our experiments were performed with a Semi-Continuous Hidden Markov Model speech recognition system developed by our lab. The system uses agglomerated word-internal triphones. Our corpus, collected by our lab, is a speaker-dependent corpus related to a command and control application. It consists of 800 English words and its trigram language model has a perplexity of 6.4. The training set spans about 3 hours and consists of

1,500 clean spoken utterances, from which 784 triphones were trained. The test set spans about ¾-hour and consists of 500 clean spoken utterances. Three different noises—F16 cockpit, Volvo car interior, Factory—publicly obtained from the Signal Processing Information Base artificially added to each sample to simulate the desired condition.

The accuracy of the recognition is computed by the following expression,

$$ACC(\%) = \frac{N_{total} - N_{sub} - N_{ins} - N_{del}}{N_{total}} \times 100\% \qquad (13)$$

where N_{total} is the total number of words, N_{sub} the number of substitions, N_{ins} the number of insertions and N_{del} the number of deletions. All accuracy is presented as the averaged accuracy for all three noise conditions. The computational complexity is computed in weight million operations per second (wMOPs) as defined in [13].

Table 1 compares the recognition accuracy of our DSP front-end processor under different compression rates with the recognition accuracy of a floating point equivalent implementation of the front-end processor. Table 2 shows the operational requirements of the DSP front-end compared to the ETSI STQ's operational requirements for the Advanced Front-End processor [14].

	Average Word Accuracy (%)			
	FP	DSP	DSP	DSP
Bitrate (kbps)	41.6	41.6	7.0	4.8
Clean	95.3	95.3	95.1	95.0
30 dB	95.3	95.3	95.2	94.9
25 dB	95.2	95.2	95.1	94.7
20 dB	94.8	94.8	94.8	94.4
15 dB	92.6	92.6	92.5	91.4
10 dB	77.1	77.1	76.5	72.5
05 dB	38.2	38.2	38.0	36.9
00 dB	34.2	34.1	34.3	33.9

Table 1: Accuracy of our DSP front-end against an FP front end for a small vocabulary task.

	ETSI Requirements	Our Baseline Implementation
Complexity (wMOPs)	17.0	15.978
ROM (kWords)	15.0	8.080
RAM (kWords)	6.0	2.413

Table 2: Our DSP front-end processor against ETSI's specified operational requirements.

Clearly, the accuracy of our unquantized DSP-based front-end feature extraction is close, if not equivalent, to the floating point implementation, bearing in mind that the engine was trained with features extracted with the floating point front-end. It should be noted that while the quantization performs fairly well in 4.8 kbps under clean conditions, the recognition accuracy degrades significantly at low SNRs. In view of this, we have chosen 7.0 kbps as the quantization rate for the rest of the experiments. It can be seen that the front-end processor that we have developed runs within the operational requirements spelled out by ETSI STQ Aurora.

Optimization may be achieved by increasing the sharing of the exponent in the BFP technique. This may be accomplished by increasing the value for Λ. Using 7.0 kbps as the compression rate, Table 3 shows that the recognition accuracy suffers no degradation even when all 256 mantissas share 1 exponent (Λ=256). It also shows small decrease in computational complexity and RAM usage.

Further optimization may be achieved by reducing *S*. Numerical accuracy is expected to decrease. Table 4 shows that the decrease in recognition accuracy is minimal, if *S*=4. When *S*=0, however, the reduction is omitted. Consequently, the computational complexity decreases sharply. The recognition, however, degrades significantly as well.

	Average Word Accuracy (%)				
Λ	16	32	64	128	256
Clean	95.1	95.1	95.1	95.1	95.1
30dB	95.2	95.3	95.3	95.3	95.3
25dB	95.1	95.2	95.2	95.2	95.2
20dB	94.8	94.8	94.8	94.8	94.8
15dB	92.5	92.5	92.5	92.5	92.5
10dB	76.5	76.5	76.5	76.5	76.5
5dB	38.0	38.0	38.0	38.0	38.0
0dB	34.3	34.3	34.3	34.3	34.3
	Operational Requirements				
Complexity	**15.98**	**15.91**	**15.87**	**15.85**	**15.84**
RAM	**2.413**	**2.397**	**2.389**	**2.385**	**2.383**
ROM	8.080	8.080	8.080	8.080	8.080

Table 3: Effects of varying Λ.

	Average Word Accuracy (%)				
S	8	6	4	2	0
Clean	95.1	95.1	95.1	95.2	94.0
30dB	95.3	95.2	95.2	95.3	94.0
25dB	95.2	95.2	95.1	95.2	93.6
20dB	94.8	94.8	94.8	94.8	92.1
15dB	92.5	92.4	92.5	92.2	84.8
10dB	76.5	76.5	76.6	76.3	61.4
5dB	38.0	38.1	38.1	37.7	36.0
0dB	34.3	34.2	34.1	33.8	33.0
	Operational Requirements				
Complexity	15.84	15.84	15.84	15.84	**11.65**
RAM	2.383	2.383	2.383	2.383	2.383
ROM	**8.080**	**4.624**	**3.760**	**3.544**	**3.490**

Table 4: Effects of varying S.

	Average Word Accuracy (%)				
N_{log}, N_{pow}, N_{sqrt}	2,2,3	2,2,2	2,2,1	1,1,2	1,1,1
Clean	95.1	95.1	95.2	95.1	95.1
30dB	95.2	95.2	95.2	95.2	95.2
25dB	95.1	95.2	95.2	95.2	95.2
20dB	94.8	94.9	94.8	94.8	94.8
15dB	92.5	92.5	92.5	92.3	92.2
10dB	76.6	76.8	76.9	76.4	76.3
5dB	38.1	38.0	38.3	37.8	37.6
0dB	34.1	34.2	34.4	33.7	33.7
	Operational Requirements				
Complexity	15.84	14.26	12.62	14.06	12.42
RAM	2.383	2.383	2.383	2.383	2.383
ROM	3.760	3.696	3.664	3.632	3.568

Table 5: Effects of varying N_{log}, N_{pow}, N_{sqrt}.

One final optimization may be performed on the DSP front

end processor by reducing the number of reduction steps per elementary function. Table 5, however, shows that we are able to keep the degradation in recognition accuracy fairly stable when the number of reduction steps of the square-root, N_{sqrt}, is reduced to 1. We also observe a signification decrease in computational complexity and ROM usage.

7. LARGE VOCABULARY SPEECH RECOGNITION

Our large vocabulary, collected by our lab, has 20,200 words and its trigram language model has a perplexity of 102.5 obtained from a corpus of 15 million running words. The training set is a speaker-dependent 15-hour collection of 15,000 clean spoken utterances, from which a total of 1,690 triphones were trained. The test-set is a 1-hour dictation of 1,000 clean spoken utterances. The same three types of noise and the various SNR levels were simulated by amplifying and artificially adding them into the clean signals. Table 6 demonstrates that our DSP implementation of the front-end processor suffers insignificant degradation in recognition accuracy even when used on a large vocabulary corpus.

Implementation	FP	DSP	DSP
Bitrate (kbps)	41.6	41.6	7.0
Λ	N/A	16	256
S	N/A	8	4
N_{log}, N_{pow}, N_{sqrt}	N/A	2, 2, 3	2, 2, 1
	Average Word Accuracy (%)		
Clean	92.6	92.6	92.4
30 dB	92.2	92.2	92.0
25 dB	91.8	91.7	91.8
20 dB	89.1	89.1	89.2
15 dB	79.9	79.9	80.0
10 dB	55.3	55.1	54.9
05 dB	33.6	33.6	33.5
00 dB	31.7	31.6	31.7
	Operational Requirements		
Complexity	N/A	15.98	12.62
RAM	N/A	2.413	2.383
ROM	N/A	8.080	3.664

Table 6: Comparing our DSP front-end against an FP front-end for a large vocabulary task.

	ETSI Requirements	Our Optimized Implementation
Complexity (wMOPs)	17.0	12.62
ROM (kWords)	15.0	2.383
RAM (kWords)	6.0	3.664

Table 7: Our optimized DSP front-end against the ETSI specified operational requirements.

8. CONCLUSION

We have shown that the recognition accuracy produced by our DSP front-end processor is comparable if not equivalent to a floating point implementation of the front-end processor. This implies that we do not need to train the recognition system specifically with the features produced by the DSP front-end. And this is true for both small vocabulary and large vocabulary systems. We have shown that we may derive large savings in computational complexity, ROM and RAM usage without sacrificing the recognition accuracy. As a result, our DSP front-end processor, as shown in Table 7, runs within the limits for the operational requirements specified by ETSI requirements for the advanced front-end.

9. REFERENCES

[1] STQ Aurora Working Group, "ETSI ES 202 050 V1.1.1," ETSI, Jul. 2002.

[2] STQ Aurora Working Group, "ETSI ES 201 108 V1.1.2," ETSI, Apr. 2000.

[3] A. Ganapathiraju, J. Hamaker, A. Skjellum and J. Picone, "A Comparative Analysis of FFT Algorithms," available at http://www.isip.msstate.edu/publications/journals/index.html, Dec. 1997.

[4] A. Acero. and R. M. Stern, "Towards microphone-independent speech recognition," in *Proc. of the DARPA Speech and Natural Language Workshop* , 1990.

[5] U. Yapanel, J. H. L. Hansen, R. Sarikaya and B. Pellon, "Robust Digit Recognition in Noise: An Evaluation Using the AURORA Corpus," in *Eurospeech,* vol. 1, pp. 209-212, 2001.

[6] R. Sarikaya and J. H. L. Hansen, "Analysis of the Root Cepstrum for Acoustic Modeling and Fast Decoding in Speech Recognition," in *Eurospeech,* Sep. 2001.

[7] L. Mauuray, "Blind Equalization in the Cepstral Domain for Robust Speech Recognition," in *Proc. European Signal Processing Conference,* 1998.

[8] C. Kermovant, "A Comparison of Noise Reduction Techniques for Robust Speech Recognition," in *IDIAP Research Report,* Jul. 1999.

[9] A. V. Oppenheim, "Realization of Digital Filters using Block Floating-Point Arithmetic," in *IEEE Trans. on Audio and Electroacoustics,* vol. 18, pp. 130-136, Jun. 1970.

[10] M. Schulte and E. Swartzlander, "Exact Rounding of Certain Elementary Functions," in *Proc. of the 11th IEEE Symposium on Computer Arithmetic,* pp. 138-145, Jul. 1993.

[11] J. Hormigo and J. Villalba, "A Hardware Algorithm for Variable-Precision Logarithm," in *Proc. IEEE Conf. on Application-Specific Systems, Architectures and Processors,* pp. 215-224, Jul. 2000.

[12] G. N. Ramaswamy and P. S. Gopalakrishnan, "Compression of Acoustic Features for Speech Recognition in Network Environments," in *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing,* pp. 977-980, 1998.

[13] ITU-T, "ITU-T Software Library Tools 2000 User's Manual," in *ITU-T Users' Groups on Software Tools,* Geneva, Dec. 2000.

[14] STQ Aurora DSR Working Group, "Advanced DSR Front-End: Definition of Required Performance Characteristics," in *AU/371/01*, Oct. 2001.