DECODER OPTIMIZATION OF LARGE VOCABULARY CONNECTED WORDS SPEECH RECOGNITION SYSTEM ON DSP

Seokyeong Jeong[1], Taeyoon Kim[2], Oh-il Kwon[3] and Hanseok Ko[1]

[1] Department of Visual Information Processing, Korea University, Seoul, Korea
[2] School of Electronics Engineering, Korea University, Seoul, Korea
[3] R&D Center, Hyundai Autonet Co., Icheon, Korea
Email: {syjeong,tykim}@ispl.korea.ac.kr, koi@haco.co.kr, hsko@korea.ac.kr

ABSTRACT

This paper presents an efficient decoder technique for use in large vocabulary connected words speech recognition on DSP platform, which is designed to operate in an intelligent vehicle system. Designing speech interface for a vehicular environment is challenging because the computing resource available to an embedded system is typically much limited than the general-purpose PC. It is important to consider optimization techniques from computational algorithms to hardware structures where ASR engine is to be ported. We introduce efficient implementation techniques for embedded platforms are presented - such as tied-mixture model, Gaussian selection, internal memory mapping in DSP L2 cache, static memory management, parameter table lookup, etc.- all for realizing a high-speed speech recognizer design suitable for DSP vehicular navigation system.

1. INTRODUCTION

Recently, speech recognition systems on embedded platforms have received a lot of attention. Particularly in the telematics devices wherein voice command navigation and voice activated email uploading/downloading functions can be deployed, speech interface capability is becoming attractive in the next generation automobiles. However, because the available resources in embedded systems are extremely limited and the computing power is significantly lower than general PC, speech recognition systems based on large vocabulary or continuous speech pose a challenge in implementation. To solve these problems, various optimization techniques for decoders have been investigated. Among the recently proposed ones, a tied-mixture HMM model has been proposed as a promising one [1][2]. This acoustic model has not only small size compatible with DSP system, but also high accuracy nearly like CHMM model. Hence our system uses tied-mixture model compact enough for embedded environment, so that it is anticipated to render memory efficiency and fast decoding. Moreover, we use the Gaussian selection method with pool evaluation to improve of recognition processing speed.

It is important to consider the optimization techniques from computational algorithm to hardware structures where ASR engine is to be ported. DSP core has 2-level cache and uses 2nd level (L2) cache as internal memory. Cache memory's speed is as fast as about 90 times more than SDRAM. Thus it achieves good performance if it is loaded into the internal memory of repeated processes or much exhausted routine by computational loads. Moreover dynamic memory allocation and free processing causes additional hardware execution time in DSP. Hence, static memory allocation by stack data structure is used to handle the memory. We discuss other optimization techniques in algorithms used in fast decoder development.

This paper is organized as follows. Section 2 briefly reviews the tied-mixture model for embedded system. Section 3 describes the hardware platform using a general purpose DSP where the proposed decoding algorithm and optimization techniques are implemented. In Section 4, representative experiments and their results are presented. Finally, conclusions are made in Section 5.

2. PROPOSED ALGORITHMS

2.1. Tied-mixture model

2.1.1. Acoustic model for embedded system

The acoustic model for a DSP system has two challenges. First, it cannot accommodate as much memory as that of a conventional CHMM model. Second, degradation of recognition performance must be minimized in spite of the resource limitation. To solve these problems, a Semi-Continuous HMM (SCHMM) modeling technique was introduced and implemented through the tied-mixture modeling method [1][2]. Also, a Context Dependent model (tri-phone) is employed to reflect maximum articulation effect by previous and next context. We also employ the state tying technique of a decision tree based to solve the unseen context problems and prevent data insufficiency problems at tri-phone training [4]. As such, the employed acoustic model consists of about 14% of CHMM model size while the recognition performance decreases less than 1%. The output pdf of SCHMM is as follows:

$$b_s(\vec{x}_t) = \sum_{l=1}^{L} b_s(i) N(\vec{x}_t; \vec{\mu}_l, \vec{\Sigma}_l) \qquad \textbf{(1)}$$

where L is codebook size, $b_s(l)$ is weight by discrete probability value of index l in state output pdf and $N(\bullet)$ denotes Gaussian distribution. where L is codebook size, $b_s(l)$ is weight by discrete probability value of index l in state output pdf and $N(\bullet)$ denotes Gaussian distribution.

2.1.2. Gaussians selection with pool evaluation

In Viterbi decoding, most of the computing time is expended in output probability computation. Output probability computation can be divided into two parts – the "Mahalanobis distance" part (a Euclidian distance computation that consider variance) and the "Log-Add" part of each Gaussian mixture. Particularly in the tied-mixture model, the state pdf arithmetic process shares all Gaussians in the pool and adds all the weight of each mixture. Thus, the log-add operation substantially occupies the most computation parts. To reduce the decoding time, it is important to select only a few representative Gaussians and employ them for log add computation. If a feature vector is input in the each frame, after the Mahalanobis distance calculation about all Gaussians in the pool, select these Gaussians in order of high scorers when sorted above the threshold value. Therefore, the computational load can be reduced using the selected Gaussians as they are used for "log-add" computation. The threshold value is obtained from experiments. The Gaussian selection expression based on tied-mixture is as follows:

$$b_s(\vec{x}_t) = \sum_{p_i > thresh} p_i\,, \qquad p_i = b_s(i) N(\vec{x}_t; \vec{\mu}_i, \vec{\Sigma}_i) \tag{2}$$

where p_i is *i*'th Gaussian likelihood.

2.2. Fast decoder design

It is important to consider the optimization techniques from a computational algorithm to hardware structures where an ASR engine is to be ported. Hereafter we will discuss some techniques used in fast decoder development.

2.2.1. Log-add look-up table

As described in the previous section, log addition operation takes the most computing time in output probability calculation. Generally, since log-add operation is difficult to implement directly, it is modified as follows theory [6];

$$\begin{aligned} A+B &= A\left(1+\frac{B}{A}\right) \\ \ln(A+B) &= \ln\left(A\left(1+\frac{B}{A}\right)\right) \\ \ln(A+B) &= \ln(A)+\ln\left(1+\frac{B}{A}\right) \\ \ln(A+B) &= \ln(A)+\ln\left(1+e^{(\ln(B)-\ln(A))}\right) \end{aligned} \tag{3}$$

Hence, we implement the log-add operation with log and exponential function in development tool as follows;

$$\log\_\operatorname{add}(x,y) = x + \log(1+\exp|x-y|) \tag{4}$$

where *x* and *y* are log domain parameters.

However, if a log addition expression is given as in Eq (5), it requires 2 additions, 1 subtraction, and 1 log and exponential computation in each log addition processing.

If we execute log(•) and exp(•) functions using the math library supported in an actual DSP compiler, the log(•) function takes 152cycles and the exp(•) function takes 229cycles in a 32bit float type's operation [7]. The log-add(•) operation consumes about 100 times compared to the add operation of 4cycles in the 32bit float type. As before, it is impossible to implement a real-time decoder based on the TM model where log-add operation is performed frequently. Therefore, a log-table created in program initializing to approximate the log addition operation is used instead of using log(•) and exp(•) functions. The error rate through approximation is acceptable and the computational load is replaced by one addition and subtraction, and multiplication and division just once in an actual operation.

$$\begin{aligned} &\log\_\text{table}[i] = \log(1+\exp(i * step))\,,\ (step = -5/256, i < 256) \\ &\log\_\operatorname{addT}(x,y) = x + \log\_\text{tabel}[|x-y| * (-256/5)] \end{aligned} \tag{5}$$

2.2.2. Static memory allocation using stack

In a Viterbi search, the structure named token that has each state's accumulated probability value gets created and disappears frequently [5]. Because the number of tokens changes frequently in search processing, it should be manageable dynamically and the structure is a profitably linked list data type. However, it causes additional hardware execution time in DSP during dynamic memory allocation and free processing. Thus, it is recommended to use static memory if possible. The stack data structure is designed to handle the token's memory. First, allocate the necessary token's whole number of memory space statically. When the token's creation is required in a Viterbi search, the index of already existing memory is passed to a new token. Because each activated token consists of a linked list data structure, it can be approached sequentially when the token is created or disappears. Also, a frequently required part of memory allocation is the token creation part in the entire recognition structure. Therefore, it can reduce the total recognition execution time by static memory assignment using stack.

2.2.3. Internal memory in DSP L2 cache

Cache memory acts as an intermediate bridge between SDRAM and CPU on DSP inside, and exists by L1 and L2 form. L1 memory is preserved, and L2 memory is programmable in that the developer can use it by data structure of the program code or cache, stack, heap etc… The size is smaller than SDRAM, but the speed is as fast as more than 90 times. Thus it gives good performance if it is loaded into the internal memory of repeated processes or a much exhausted routine by computational loads. To take account of the size given L2 memory, we load the functions such as pool evaluation, output pdf, memory free, etc. and data structure such as log-add table etc. into the internal memory.

2.2.4. State pdf caching

Most of the occupying computational load during the decoding process based on HMM is output pdf computation. In particular, because search space becomes huge for large vocabulary recognition, output pdf of many states is calculated for just one frame. However, it is not necessary to compute the state pdf repeatedly about equal states whose operation is already performed.

Therefore, store the state likelihood whose output pdf is already calculated into the cache so that the cache value is used when the same state's calculation is repeated. For fast processing, the program defines the cache size as the total number of states, and declares the flag array as much as cache size. Then if output pdf computation of any state is performed, activate the cache index of flag array as the state's index and store the likelihood into the cache array located in the same index. After this calculation, refer the relevant cache values to the state index.

2.3. Connected word search network design

Continuous speech recognition provides more convenience to device users than isolated word recognition. Because of the reduction in the number of responses required in dialogue process, users in continuous speech recognition are enabled to execute the various commands in only 1 step. Continuous speech recognition, however, needs more resources to perform decoding in huge search space and thus must deal with linguistic information. For the purpose to reduce the system resources and provide the convenience similar to that of continuous speech recognition, we implemented it with a connected word recognition system. This system is based on isolated word recognition platform, but the search space follows the connected search network. Performing the Viterbi algorithm, the token, which stores a partial accumulated probability and some information, saves its present word and traverses next word until it reaches to word end node [5]. When the utterance is finished, the maximum probable path is backtracked to obtain the recognized word sequence. In the case of navigation system, command menu is organized using level building structure like that of tree topology in general. Hence the user speaks just one sentence from top-down menu to command the order while user must speak every time in each level menu as in an isolated word recognition system.

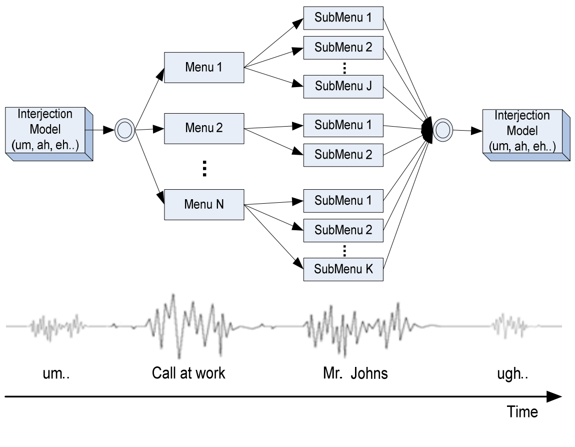

Fig. 2. Block diagram of connected word search network

3. HARDWARE OVERVIEW

The proposed decoder in this paper is developed to use in an embedded system to recognize main place-names of 1800 places in city with speech. The core processor in embedded system used has 150MHz (900MFLOPS) maximum frequency. Floating point computation is supported and the execution of 8 instructions is possible at the same time by pipeline using 8 arithmetic logic units. Delivering data through 32bit outside memory interface and outside SDRAM is 16MB, and FLASH Read Only Memory is 1MB. The ADC/DAC converter for speech input/output uses of 11KHz sample rate, quantization precision is 16bit.

Cache memory of L1, L2 exists inside the DSP core, L1 is reserved by data and program memory of each 4Kbyte allocate. L2 cache is 64Kbyte and can use data and programs freely. The system performance is increased as the core program code and data are loaded into L2 internal memory. A block diagram of the total system is as follows.

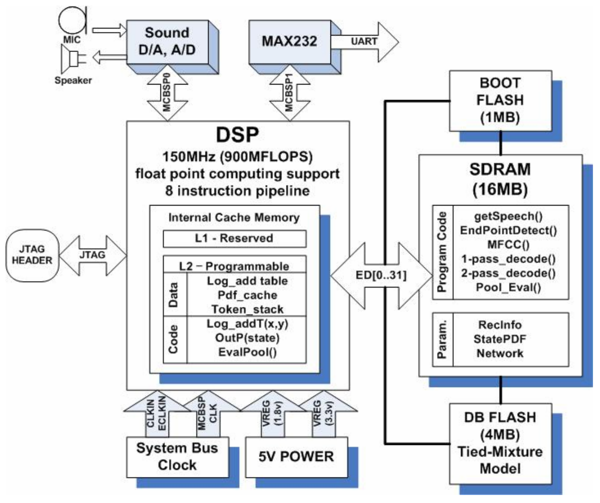

Fig. 3. Block diagram of DSP system

4. EXPERIMENTS

4.1. General setup

Twenty six (26) dimensions and 2 streams MFCC which consist of 12 MFCC's, 12 delta MFCC's, and 2 log Energy's were used for feature vector. For the test set, vocabulary selected is 1880 names of places in Seoul city. The test DB consists of 4,680 utterances from 19 men and 20 women between 20~40 years old. The speech DB sampling rate is 11 kHz with 16bit precision, and PCM encoded.

A tied-mixture acoustic model was trained. The number of Gaussians in pool is 512 (256 in each stream), and 1,649 triphone states were obtained via decision tree state clustering from 3,686 triphones.

4.2. Tied Mixture Model vs. CHMM

We compared the performance of the proposed tied-mixture model with a conventional CHMM acoustic model. The same Viterbi decoder, beam pruning value, and various parameters are used in tied-mixture model and CHMM. We compared the accuracy and size of models. Table 1 shows the result of the experiments.

Model	Accuracy (%)	Size (Byte)
CHMM	94.14	6,633,428
TM-HMM	93.31	933,472

Table 1. Comparison of performance between tied-mixture model and CHMM acoutic model

In this result, tied-mixture model accuracy is slightly lower than CHMM. However, we could obtain 7 times smaller size model without significant performance degradation.

4.3. With log-add look-up table

Table 2 shows total processing times when conventional log(•), exp(•) function and log additional tables are used. Log look-up table and conventional log_add(•) functions are implemented using equations derived in Section 2.2.1. A huge amount of processing time reduction is obtained. The total processing time of conventional log_add(•) function is 16.7 times more than the log-table version. Although the DSP supports float arithmetic, log(•) or exp(•) functions consume enormous computational time. Thus, it is important not to use logarithm or exponential functions if possible and instead convert with lookup tables.

Decoding	Conventional Log-add function	Log look-up table
Processing time(μsec)	16,810,377	1,805,876

Table 2. Comparison of performance using log-table and conventional log-add function

4.4. With static memory allocation using stack

Table 3 shows a comparison of total processing times of using dynamic memory allocation and static memory allocation. Static memory allocation uses stack data type and it consumes shorter processing time than dynamic memory allocation by about 0.86 times.

Decoding	Dynamic memory allocation	Static memory allocation
Processing time (μsec)	2,099,855	1,805,876

Table 3. Comparison of performance using dynamic memory allocation and static memory allocation

4.5. Connected word recognition

We computed the connected word performance of 902 Seoul city address book consisting of 2 words (section & village names) task. Word network consists of 2-level building structure like that of tree topology. Test used 4,665 of continuous sentence DB that 39 speakers each pronounced 120 sentences. In each sentence, 1st address and 2nd address of Seoul city address unit mixed is also included. Thus the total collected test number is 9,206 words. Recognition experiment result with word and sentence recognition rate is as follows.

	Correct (%)	Correct # of	Deletion error	Substitution error	Insertion error	# of Utter.
Sentence	93.09	4,326		321		4,647
Word	96.37	8,837	1	332	0	9,170

Table 4 Connected word recognition results

Table 4 shows high accuracy rate in sentence task on embedded platform. Total processing time is same with above results.

5. CONCLUSIONS

In this paper, we described various design techniques for fast decoding with 1,880 isolated words and 902 connected words recognition task on automobile navigation system. To reduce the memory space, the tied-mixture model was considered. In addition, some resource optimizing techniques for DSP architecture, such as using a log table, internal memory, pdf cache, and memory management are proposed. Moreover, connected word search network is applied on embedded system for users making utterances more convenient. The experimental results show that the tied-mixture model reduces the memory size by about 7 times than the conventional CHMM without significant performance degradation. Other optimizing methods for embedded platform improved the overall processing speed substantially.

7. REFERENCES

[1] J.R. Bellegarda, D. Nahamoo, "Tied mixture continuous parameter modeling for speech recognition", IEEE Transactions on SAP, Vol. 38, Issue: 12, Dec. 1990, pp. 2033 – 2045

[2] Junho Park and HANSEOK KO, "Achieving a Reliable Compact Acoustic Model for Embedded Speech Recognition System with High Confusion Frequency Model Handling", Speech Communication, Vol.48, Issue 6, pp 737-745, June, 2006.

[3] Gales, M., Knill, K., Young, S., 1999. "State-based Gaussian selection in large vocabulary continuous speech recognition using HMM's". IEEE Trans. Speech Audio Process. 7 (March), 152–161.

[4] S. J. Young, J. J. Odell, and P. Woodland, "Tree-based state-tying for high accuracy acoustic modeling," in Proc. ARPA Workshop Human Language Technol., 1994, pp. 286-291.

[5] S.J. Young, N.H. Russel, J.H. Thornton, "Token passing: a simple conceptual model for connected speech recognition systems" Technical Report F_INFENG/TR38, Cambridge University Engineering Department. 1989.

[6] SJ Melnikoff, SF Quigley "Implementing the Log-Add Algorithm in Hardware", *Electronic Letters,* 39, 12, 2003, 939-941.

[7] TMS320C67x FastRTS Library Programmer's Reference (SPRU100A), Texas Instruments, October 2002. [Online]. Available: http://focus.ti.com/lit/ug/spru100a/spru100a.pdf