Embedded Julius: Continuous Speech Recognition Software for Microprocessor

Hiroaki Kokubo*, Nobuo Hataoka*, Akinobu Lee†, Tatsuya Kawahara‡ and Kiyohiro Shikano§
*Central Research Laboratory, Hitachi Ltd, Kokubunji, Tokyo, Japan.
Email:kokubo,hataoka@crl.hitachi.co.jp
†Nagoya Institute of Technology, Nagoya, Japan. Email: ri@nitech.ac.jp
‡Kyoto University, Kyoto, Japan. Email: kawahara@i.kyoto-u.ac.jp
§Nara Institute of Science and Technology, Nara, Japan. Email: sikano@is.naist.jp

***Abstract*—To expand CSR (continuous speech recognition) software to the mobile environmental use, we have developed embedded version of "Julius". Julius is open source CSR software, and has been used by many researchers and developers in Japan as a standard decoder on PCs. Julius works as a real time decoder on a PC. However further computational reduction is necessary to use Julius on a microprocessor. Further cost reduction is needed. For reducing cost of calculating pdfs (probability density function), Julius adopts a GMS (Gaussian Mixture Selection) method. In this paper, we modify the GMS method to realize a continuous speech recognizer on microprocessors. This approach does not change the structure of acoustic models in consistency with that used by conventional Julius, and enables developers to use acoustic models developed by popular modeling tools. On simulation, the proposed method has archived 20% reduction of computational costs compared to conventional GMS, 40% reduction compared to no GMS. Finally, the embedded version of Julius was tested on a developmental hardware platform named "T-engine". The proposed method showed 2.23 of RTF (Real Time Factor) resulting 79% of that of no GMS without any degradation of recognition performance.**

I. Introduction

Recently, the CSR (continuous speech recognition) software has been commercially available. These software packages are used to various applications such as dictation software and transcription of news announcement reports. However, these software packages are working on PCs (Personal Computers), which require huge computing resources. Introducing speech interfaces to mobile equipments such as car navigation systems and mobile phones, embedded CSR software running on microprocessors is needed[1]. We believe speech interfaces are popularized in various mobile products, if the embedded CSR software is readily available.

Our goal is to work CSR software named Julius on a microprocessor. Julius[2] is open source CSR software developed on PCs. It has been used by many researchers and developers in Japan as a standard decoder. The largest advantage of using Julius is standard formats of language/acoustic models are adopted. Developers are free to use acoustic/language models created by popular modeling tools (such as HTK[3], CMU-Cambridge Toolkit[4]).

We have developed the embedded Julius on a developmental hardware platform named "T-engine"[5]. The T-engine has a SH-4[6] microprocessor. The SH-4 is a 32-bit RISC processor developed by Renesas Technology Corp. A microprocessor has fewer computational power rather than a PC. In case of SH-4, its operating frequency is 240MHz. Problem is reducing computational costs.

A major part of computational cost for CSR is pdf (probability density function) calculations. To realize CSR software with low calculation cost, computational reduction for pdf calculations is needed. Popular approaches for reducing cost of pdf calculations are reduction in the number of Gaussians on a modeling phase and/or pruning Gaussians on a decoding phase. A typical methods of former approaches are tied-mixture HMMs and a tree-structured probability density method[7]. Typical method of the latter approaches is (GS) Gaussian Selection[8][9][10][11], which calculates only Gaussians near an input vector, instead of evaluating all Gaussians. Julius already introduces a PTM (Phonetic Tied-Mixture) modeling[12] and GMS (Gaussian Mixture Selection) algorithm[13], which reduce the cost of pdf calculations. Julius runs real time decoding on PCs. However further cost reduction is necessary on microprocessor.

In this paper, we propose an expansion of the GMS method in purpose of further reducing the cost of pdf calculations. This approach does not change structure of acoustic models in consistency with that used by Julius on PCs. This enables developers to apply speech applications developed on PCs for mobile applications.

II. Proposed method

In this section, we explain the conventional GMS method briefly, and then describe details of its modifications.

A. Gaussian mixture selection

The GMS method[13] is a procedure to select Gaussian distributions by the HMM states using a hierarchical relationship between monophone models and triphone models. Fig. 1 shows a GMS procedure. For each frame, all Gaussians of monophone HMM states are computed and then Gaussians of triphone models corresponding to k-best states of monophone HMMs are selected for computations.

This method is effective for reducing computational cost of pdf calculations, but almost half of the total process time is

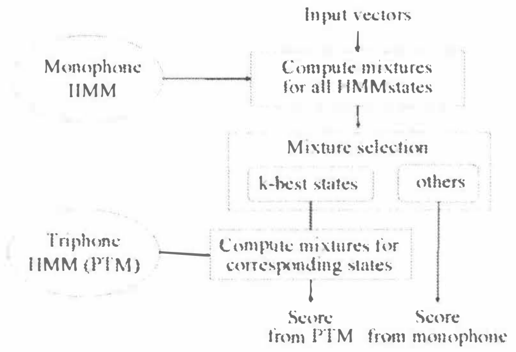

Fig. 1. Conventional Gaussian Mixture Selection (GMS)

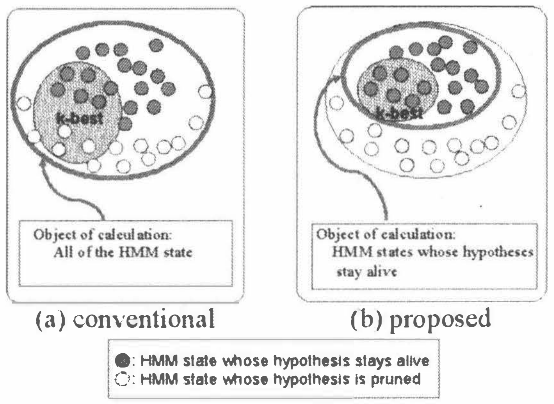
k-best
Object of calculation: All of the HMM state
k-best
Object of calculation: HMM states whose hypotheses stay alive
(a) conventional
(b) proposed
●: HMM state whose hypothesis stays alive
○: HMM state whose hypothesis is pruned

Fig. 2. Modification on Mixture Selection Stage

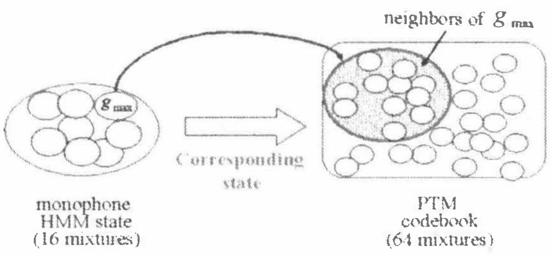

Fig. 3. Gaussian Selection within HMM State

still taken by pdf calculations. Additional process reduction is needed.

B. Modifications on GMS

We made two modifications on the GMS method as follows;

1) Computational reduction on mixture selection stage: Since low-scored hypotheses are pruned by a beam search strategy, hypotheses in all HMM states do not necessarily stay alive. Hypotheses survived at time $t-1$ will be transfer to limited HMM states through transition arcs. And some HMM states will not be visited by any hypothesis at time t. It is waste to calculate pdfs in HMM states which hypotheses will not visit.

In the conventional GMS method, all HMM states in monophone models are calculated for the mixture selection, since information of active states is not available at that time. Knowing where hypotheses stay alive at time $t-1$, only HMM states which these hypotheses may be visited are calculated for the mixture selection stage at time t.

Fig. 2 shows an image of the modified strategy. Filled circles are designated HMM states which hypotheses at time $t-1$ will visit, and unfilled circles are designated HMM states which no hypotheses will visit at time t. Fig. 2 (a) is monophone models for the conventional GMS. Pdf scores of all states are calculated and k-best states among them are selected (meshed area). Possibly no hypothesis is transferred to some state of k-best states at time t. It is useless to calculate a pdf score of this state. Fig. 2 (b) is monophone models for the proposed and modified GMS. Target HMM states for pdf calculations are restricted to states where hypotheses at time $t-1$ will transfer (within a bold circle). To know target HMM states, we use information about HMM states having hypotheses survived at time $t-1$ and transition arcs.

Applying this modification, computational cost is reduced for the mixture selection compared to the conventional GMS method, which calculates all states in monophone models. Furthermore, since there is no fear of selecting the useless states whose pdf score is not needed to calculate, a small number of k-best states could be specified without any degradation of recognition performance.

2) Gaussian selection within HMM state: Gaussian selection (GS)[8] is based on the idea "only Gaussians close to an input vector have dominant effect on a pdf score of an HMM state". On the other hand, the conventional GMS method calculates all Gaussians within an HMM state, even though a pdf score derived from Gaussian distant from the input vector is negligible. For reducing pdf calculations on HMM states, the calculation strategy is changed: calculating only Gaussians neighboring the input vector, instead of calculating all Gaussians. Fig. 3 shows details of the modification. g_{max} is the Gaussian whose score is maximum within a monophone HMM state. It is plausible idea that Gaussians close to g_{max} are also close to the input vector. Based on this idea, only Gaussians close to g_{max} are calculated for pdf score of a PTM state. The others, which are far from g_{max}, are ignored. Concrete procedure is as follows: Distances between $g_{mono}(i)$ and $g_{PTM}(j)$ with all combinations of i and j are calculated in advance, where $g_{mono}(i)$ and $g_{PTM}(j)$ represent i-th Gaussians in a monophone HMM state and j-th Gaussian in a corresponding PTM codebook respectively. A selection of $g_{PTM}(k)$ close to g_{max} is carried out using table lookup. By this procedure the computation cost is much more reduced.

III. Experimental results and discussions

A. Experimental conditions

Table I shows experimental conditions. The vocabulary size was 5,000 words. Julius is a two-pass decoder using a bigram language model for the 1st-pass and a trigram language model

TABLE I
EXPERIMENTAL CONDITIONS

Vocabulary size	5,000 words
Language models	Bigram (1st-pass) Trigram(2nd-pass)
Triphone models (PTM)	3,000 state,64 mixtures
Monophone models for GMS	129 state,16 mixtures
Platform	1.PC (Pentium 4 2.8GHz, Linux OS) 2.T-engine (SH-4 240MHz)

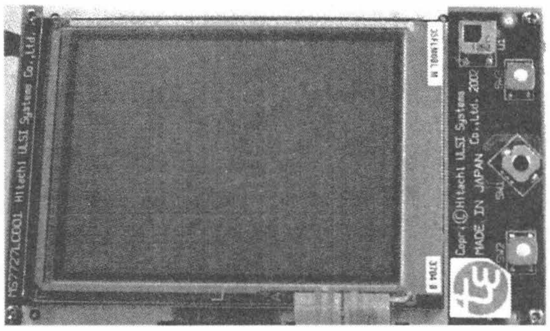

Fig. 4. T-engine Board

for the 2nd-pass. Acoustic models were PTMs having 3,000 states with 64 mixtures. Monophone models with 129 states and 16 mixtures/states were used for the GMS. Test-set speech was selected 100 sentences from JNAS (Japanese Newspaper Article Sentences) corpus[14].

Experiments were carried out in two platforms: simulation on a PC and evaluation on a T-engine board. The T-engine board is a developmental hardware platform which has common operating system (OS) called eTRON[5]. Fig. 4 and Fig. 5 show a photo and architecture of the T-engine board respectively. Table II shows specifications of the T-engine board.

The T-engine platform consists of a CPU board, an LCD board. The CPU board has one 32-bit MPU called SH-4[6] developed by Renesas Technology Corp. The SH-4 is a RISC processor with operating frequency of 240MHz. Work memory has 64 Mbytes. Program, dictionary, Language models and Acoustic models are stored in a CF (Compact Flash) card.

B. PC Simulation

Before evaluating on the T-engine platform, recognition performance was evaluated on PC simulation. Program source for T-engine was recompiled on a PC. Evaluation was done on three conditions; (1)no GMS, (2)conventional GMS, (3)proposed GMS. Performance was measured by word accuracy and CPU time. The number of k-best state for the GMS was varied from 12 to 36 (k=12, 18, 24, 36). Fig. 6 shows experimental results. The horizontal axis of CPU time was normalized by the time of the condition (1) (no GMS). In the case of using GMS, we found the less value of k-best showed the less

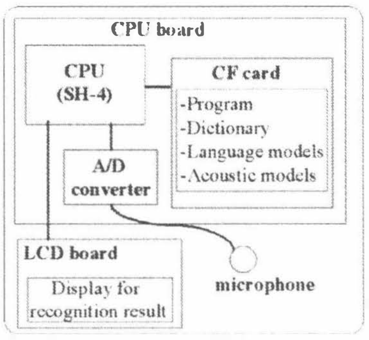

Fig. 5. Architecture of T-engine Platform

TABLE II
SPECIFICATION OF T-ENGINE BOARD

CPU	SH7751R (SH-4) (430MIPS / 1.7GFLOPS)
Operating Freq.	Interal:240MHz, External:60MHz
Flash ROM	8MByte
User RAM	64MByte
OS	T-Kernel
A/D	16MHz,16mit
LDC	TFT color, 240 x 320
Power	DC 5.6V
Size	120 mm x 75 mm

processing time and the less word accuracy. The proposed GMS method showed significant computational time reduction by 40% with accuracy loss of under 1% (k=24).

Fig. 7 shows a profile of computational cost measured on a PC. The vertical axis is a normalized CPU time calculated by the Linux command "gprof". In no GMS case, 73% of total computational cost was taken by pdf calculations. Adopting the conventional GMS method, the total computational cost was reduced to 74% of no GMS in spite of additional costs for calculating monophone models. Modifying the GMS method (the proposed GMS), the normalized CPU time for GMS was reduced 26% to 20%, the normalized CPU time of pdf calculations was reduced 24% to 16%. As a result, the total computational cost was reduced to 60% of no GMS.

C. Evaluation on T-engine platform

We evaluated system performance of the recognizer on the T-engine platform. Experimental conditions were the same as the PC simulation. The number of test speech was 60 utterances (30 males, 30 females). According to results on the PC simulation, the number of k-best state for the GMS was set to 24.

Experimental results are shown in Table III. Memory size consumed on Julius is less than 50MBytes. Adopting the GMS algorithm, additional memory space was needed for monophone models used on the mixture selection. Word accuracies of 3 conditions were almost the same. We define the RTF (Real

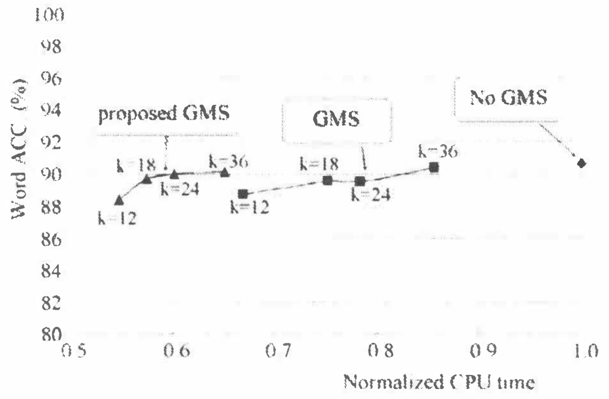

Fig. 6. Comparison of Methods

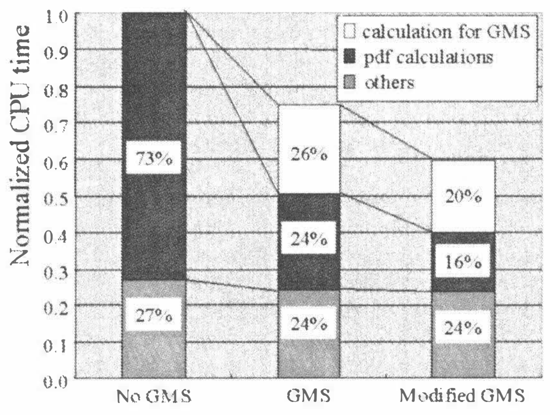

Fig. 7. Profile of Computational Cost Measured on PC (k=24)

Time Factor) was processing time normalize by speech length. The proposed GMS showed 2.23 of RTF resulting 79% of that of no GMS. According to the PC simulation, the process reduction by the proposed GMS was 40%. The gap of process reduction rate between PC and T-engine may be caused by differences in CPU architecture and cache memory size. The cache of T-engine was only 32 Mbytes.

D. Discussions

In this section, we evaluated the embedded Julius on the developmental hardware platform named T-engine. Experimental results indicated that memory requirement for the embedded Julius was less than 50 MBytes. It isn't big problem for mobile applications, because commercial PDAs have 64-128MBytes of work memory. Regarding processing speed, the proposed GMS showed 2.23 of RTF on the SH-4 (430MIPS). Renesas Technology Corp. released an upper version of the SH-4 named SH-4A (720MIPS) in 2004. It is 1.7 times faster than SH-4. Using SH-4A, the embedded Julius will achieve almost real time processing.

TABLE III
SYSTEM PERFORMANCE ON T-ENGINE (SH-4)

Conditions	Memory size	Word accuracy	RTF
no GMS	48.9MBytes	89.1%	2.81
Conventional GMS	49.8MBytes	89.3%	2.62
Proposed GMS	49.8MBytes	89.7%	2.23

IV. CONCLUSIONS AND FUTURE WORK

This paper describes an embedded Julius on T-engine. For computational reduction, we developed a modification of the GMS method. This approach does not change the structure of acoustic models in consistency with that used by conventional Julius, and enable developers to use acoustic models created by popular modeling tools. The experimental results show 2.23 of RTF resulting 79% of that of no GMS without distortion of word accuracy.

We will investigate more compact and more noise robust. For the noise robustness, we are developing a noise reduction front-end. We will also develop an application prototype using the embedded Julius on a mobile platform.

ACKNOWLEDGMENT

This research activity has been supported by the e-society project founded by Ministry of Education, Culture, Sports, Science and Technology, Japan.

REFERENCES

[1] N. Hataoka, K. Kokubo, Y. Obuchi, and A. Amano, " Development of robust speech recognition middleware on microprocessor," Proc. of IEEE ICASSP, pp.II837-II840, 1998.
[2] A.Lee, T.Kawahara, and S.Doshita, " An efficient two-pass search algorithm using word trellis index," Proc. of ICSLP, pp.1831-1834, 1998.
[3] S.Yang, G.Evermann, T.Hain, D.Kershaw, G.Moore, J.Odell, D.Ollanson, D.Povey, V.Valtchev, and P.Woodland, The HTK book(for HTK version 3.2.1), In Cambridge University Engineering Department, 2002.
[4] P.R.Clarkson and R.Rosenfeld, "Statistical language modeling using the CMU-Cambridge toolkit", Proc. Eurospeech, vol.5, pp.2707-2710, 1997.
[5] http://www.t-engine.org/index.html
[6] http://www.renesas.com/
[7] T.Watanabe, k.Shinoda, K.Takagi, and K.Iso, "High speed speech recognition using tree-structured probability density function," Proc. ICASSP, pp.556-559, 1995.
[8] E.Bocchieri, "Vector quantization for efficient computation of continuous density likelihoods," Proc of ICASSP, pp.692-695, 1993.
[9] K.Knill, M. Gales, and S.Young, "Using of Gaussian selection in large vocabulary continuous speech recognition using HMM's." Proc. of ICSLP, pp.470-473, 1996.
[10] M.Gale, K.Knill, and S.Young, "State-based Gaissoan selection in large vocabulary continuous speech recognition using HMM's," IEEE trans. Speech & audio processing, Vol.7, No.2, pp.152-161, 1999.
[11] D.Paul, "An investigation of Gaussian shortlists," Proc. of ASRU, pp.209-212, 1999.
[12] A.Lee, T.Kawahara, K.Takeda, and K.Shikano, " A new phonetic tied-mixture model for efficient decoding, " Proc. of ICASSP, pp.1269-1272, 2000.
[13] A.Lee, T.Kawahara, and K.Shikano, " Gaussian mixture selection using context-independent HMM, " Proc. of ICASSP, pp.69-72, 2001.
[14] K.Itoh, M.Yamamoto, K.Takezawa, T.Matsuoka, . K.Shikano, T.Kobayashi, and S.Itahashi, "The design of the newspaper-based Japanese large vocabulary continuous speech recognition corpus," Proc. of ICSLP, pp.3261-3264, 1998.