ROBUST EMBEDDED VERSION OF CONTINUOUS SPEECH RECOGNITION SOFTWARE ON MICROPROCESSOR

Nobuo Hataoka[1], *Hiroaki Kokubo*[1], *and Yasunari Obuchi*[1],
[1]Central Research Laboratory, Hitachi Ltd.
Kokubunji, Tokyo 185-8601, JAPAN
Phone number: +81-42-323-1111
nobuo.hataoka.ss@hitachi.com,
hiroaki.kokubo.dz@hitachi.com, yasunari.obuchi.jx@hitachi.com

Akinobu Lee[2], *Tatsuya Kawahara*[3], *and Kiyohiro Shikano*[4]
[2]Nagoya Institute of Technology, Nagoya, JAPAN
[3]Kyoto University, Kyoto, JAPAN
[4]Nara Institute of Science and Technology, Nara, JAPAN

ABSTRACT

In this paper, we describe new mobile consumer services based on speech technologies to support a new digital/mobile era of ubiquitous communication. First, we report evaluation results of recognition modules for the noise robustness such as speech segmentation, microphone array techniques, and feature normalization modules. We used free Continuous Speech Recognition (CSR) software Julius/Julian as a speech decoder. Second, we have developed an embedded version of the Julius continuous speech recognition software on general-purpose microprocessors. We used T-Engine™ as a hardware platform. The technical problems to make Julius for embedded one are computing/process and memory reductions of Julius. We could realize about 2.00 of RTF(Real Time Factor) of speech recognition processing on the condition of 5000-word vocabulary.

Categories and Subject Descriptors

I.2.7 [**Natural Language Processing**]: Speech Recognition and Synthesis, H1.2 [**User/Machine Systems**] Human information processing, C.3 [**Special-Purpose and Application Based Systems**]: Microprocessor/micro-computer applications, Real-time and embedded systems

General Terms

Experimentation, Human Factors, Languages, Theory.

Keywords

Automatic Speech Recognition (ASR), Continuous Speech Recognition (CSR), Microphone Array, Feature Normalization, Julius/Julian: Free CSR Software, Embedded Julius, Hardware Platform: T-Engine, *SuperH* Microprocessor, GMS (Gaussian Mixture Selection), HMMs (Hidden Markov Models), HMIs (Human Machine Interfaces)

1. INTRODUCTION

Recently, the continuous speech recognition software [1] has been available and these software packages are used to various applications such as dictation software and transcription of news announcement reports. However, these software packages are running on PCs (Personal Computers) which have huge computing resources, both computing power and memories.

Our goal is to develop embedded continuous speech recognition software which runs on small computing power and with small memories to extend ASR software to mobile environmental use. We envision mobile application environments, e.g. mobile information service systems such as car navigation systems and cellular phones where an embedded speech recognizer [2] is running on and which are connected to remote servers that support a variety of information-seeking tasks.

Due to improvements in microprocessor performance, it has been possible to implement multimedia processing technologies using software on microprocessors and/or DSP (Digital Signal Processing) chips. This software, called *middleware*, is a kind of code library that connects hardware and end-user applications. Middleware enables developers and users to use various media processing technologies in different mobile applications, such as car navigation systems and hand-held PCs, using a single microprocessor.

In this paper, we first report the evaluation results of recognition modules for the noise robustness such as speech segmentation, microphone array techniques, and feature normalization modules [3]. We used Continuous Speech Recognition (CSR) software called Julius/Julian as a speech decoder [1]. Second, we report implementation results of the CSR software Julius on T-Engine consisting of a *SuperH*™ microprocessor [4]. We called this embedded CSR software Julius the embedded Julius. The *SuperH* microprocessor has a very limited process power

and therefore huge computing process and memory reductions were needed to realize the embedded Julius on the T-Engine board. For the implementation issues, we report a binary acoustic models and GMS (Gaussian Mixture Selection) computing reductions to realize real time processing. Finally, the evaluation experimental results are reported showing successful implementation of Julius on T-Engine.

2. CAR TELEMATICS SERVICE

2.1 System Concept for Network Applications

As information technology expands into the mobile environment to provide ubiquitous communication, intelligent interfaces will be a key element to enable mobile access to networked information. For mobile information access, HMIs (Human Machine Interfaces) using speech might be the most important and essential application, as speech interfaces are more effective for small, portable devices. Mobile terminals such as cellular phones, PDAs (Personal Digital Assistants) and Hand-held PCs (Personal Computers) are already connected to networks such as the Internet to access information from web services. For effective use of mobile information access, speech processing and image processing will be key technologies for intelligent mobile terminals.

Especially, Car Telematics refers to a new service concept where mobile terminals (e.g. car navigation systems, cellular phones) are used to connect to networked information services. Figure 1 illustrates the total service system concept, which consists of three parts, e.g. Terminal/Client, Network, and Centre/Server. For the Terminals, sophisticated HMIs are required to handle various inquiries and to deliver information from the Centre using speech and image input/output. The Network

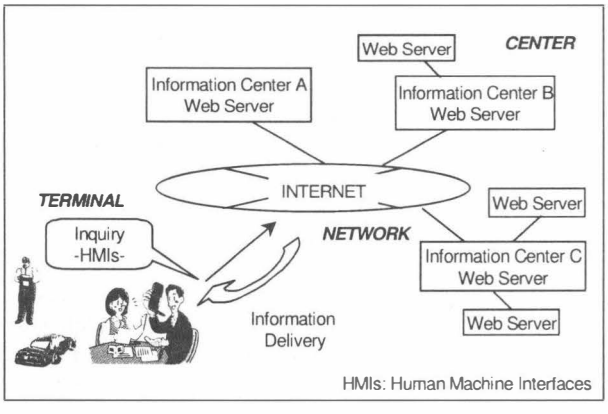

Figure 1: *Service System Image(Terminal, Network, and Centre)*

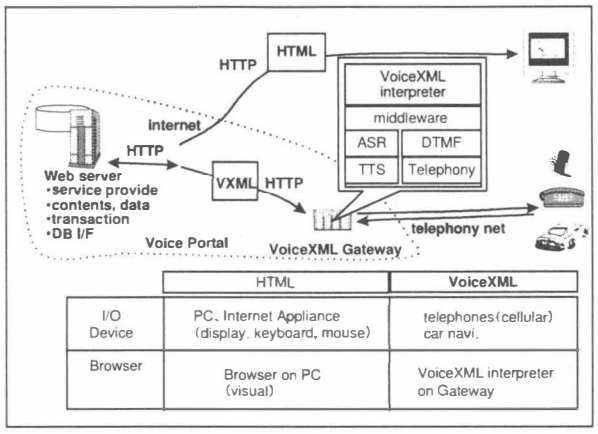

Figure 2: *Voice Portal Concept based on VoiceXML Gateway*

is typically the Internet; and via the Internet, the user's requests are transferred to related Web servers at the Centre, and required information will be provided from the Centre to users via Networks and Terminals [5].

2.2 Voice Portal Architecture

In the car, HMIs based on speech processing such as ASR and TTS are essential to provide a safe driving environment. Car Telematics using speech technology is thought of as one of the Voice Portal services. Figure 2 shows the Voice Portal service concept based on the VoiceXML gateway. In the VoiceXML gateway, VXIs are implemented for the WWW to be accessed by voice. The VoiceXML is a W3C (WWW Consortium) standard to provide dialog functions to voice systems. The VoiceXML Gateway also incorporates ASR/TTS engines; sometimes, terminals such as car navigation systems also incorporate ASR/TTS engines.

3. ASR MODULE EVALUATION

3.1 Free/Open CSR Software JULIUS/JULIAN

Julius/Julian is free/open CSR software which has been developed by Japanese Universities and delivered by WEB [1]. Julius/Julian can recognize large vocabulary over 20,000 words and running on Personal Computers (PCs) which have huge computing resources, meaning more than 1GIPS CPU power and 1GByte memory. Julius is an n-gram-based decoder and Julian is a network-based one. We evaluate many noise handling techniques (ASR modules) using a Julian decoder in automotive environments to make technical problems of speech processing modules clear [3].

3.2 Data Collection in Automotive Environments

The recording was done in downtown Tokyo, where the car was forced to drive slowly with frequent stops due to the traffic jam. Therefore, a large part of the background noise was from the surrounding environment such as other cars, constructions, etc. The speaker was sitting on the passenger seat, and there was a linear microphone array on the dashboard in front of the speaker. The array consists of 7(seven) microphones, which are located at the interval of 10cm, 5cm, 5cm, 5cm, 5cm, and 10cm. The array microphones were labeled as #1 to #7 from the driver seat side to the window side, so the central microphone was #4. The speaker has a push button (PB) to activate the recognition program. The PB was used as a PTT (Press To Talk) button. After the PB was pushed, a prompt message and a beep sound were given to the speaker. The setup of data collection using automotive environments is shown in Figure 3.

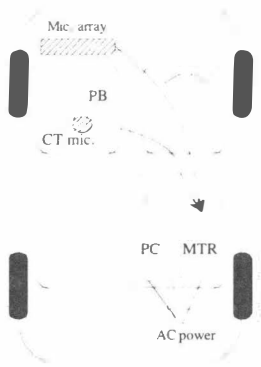

Figure 3: *Data Collection Setup in Car*

We have collected the speech data from 18 speakers (11 male and 7 females, all in their early twenties). 3,620 utterances for POIs (Point of Interests) were collected in total, and they were roughly segmented using a fixed time period from the beep. After segmentation, the length of the data was approximately 7 hours in total. These utterances were then labeled by the POI names. Some utterances include wrong pronunciation and hesitation, but they are all labeled as long as the intention of the speaker can be inferred. In this process, we found that 28 (0.8%) utterances which could not be labeled as any POIs, and were categorized as OOV (out of vocabulary). The number of utterances per speaker ranged from 134 to 326, and the number of utterances per POI ranged from 10 to 48. These numbers are supporting the argument that the utterances were made spontaneously.

Table 1: *Estimated SNR of Each Microphone Data*

mic. ID	SNR (full band) dB	SNR (400-5500Hz) dB
1	-0.5	9.3
2	-2.8	12.1
3	-3.4	8.6
4	-3.0	9.2
5	-2.7	11.7
6	-3.8	8.5
7	-2.9	10.5
close-talk	56.7	83.2

Next, we extracted the endpoint information using the close talking data and the POI label using Viterbi alignment, They were used as the "oracle" endpoint information for the noisy data. We then estimated the signal-to-noise ratio (SNR) by comparing the power of the speech and non-speech segments. Table 1 shows the estimated SNR for each microphone. Since the noise spectrum has a strong peak in the low-frequency range, we also calculated the SNR after applying a bandpass filter with the passing band of 400Hz to 5500Hz. It is interesting that the estimated SNR does not have any correlation with the distance between speakers and microphones, although the speech recognition has a correlation with the distance as mentioned later in this paper.

3.3 ASR Module Evaluation Results

(1) Baseline Experiments

After collection and analyzing the data, we carried out evaluation experiments of 152 POI isolated word recognition using Julian decoder and our original decoder. For Julian conditions, the sample acoustic model with PTM triphones, which is distributed with the source code, was used. Among various variations of Julian, the Julian-v3.4.2 grammar driven decoder with 12 MFCC and log power, plus their first-order time derivatives is used. The path of the silence models only was allowed, which is interpreted as rejection. All the data were originally sampled by 44.1kHz, but down-sampled to 16kHz prior to the experiments. In the baseline experiments, fixed length segments are used without any sophisticated endpointing, and cepstral mean normalization (CMN) was applied.

The results of the baseline experiments showed the recognition rate of distant-talk was 86.0% and that of close-talk was 93.4%. The rate of distant-talk was that of the central microphone, #4. In the experiments, the individual recognition rates ranged from about 60% to 97%.

(2) Evaluation of Various Modifications

We evaluated modifications in various modules of the speech recognition system. All results are summarized in Figure 4.

First, the importance of rejection and endpointing are evaluated using oracle information. When we used the oracle information about OOV (by adding lexicons), the 28 OOV utterances were automatically recognized correctly. Without oracle information about OOV, Julian recognized halves of the 28 OOV utterances correctly, and then the improvement by oracle information was 0.4% absolute. When we used the oracle information of the speech period, the recognition rate increased to 90.2% indicating very large improvements.

Two more sets of oracle experiments were carried out to investigate how the recognition rate changes according to the SNR improvement and vocabulary size refinement. As for the SNR, the distant-talk data were mixed with the close-talk data linearly in the time domain with a varying weight. Two typical points are plotted and the SNR was calculated after applying the bandpass filter. Figure 4 shows 14dB and 26dB enhanced speech data results. The improvement from the baseline to 14dB is much larger than the improvement from 14dB to 26dB. The vocabulary size refinement was done simply by splitting the dictionary into two or four groups, and the one including the correct word was used for recognition. Steady improvements were observed every time the vocabulary was reduced.

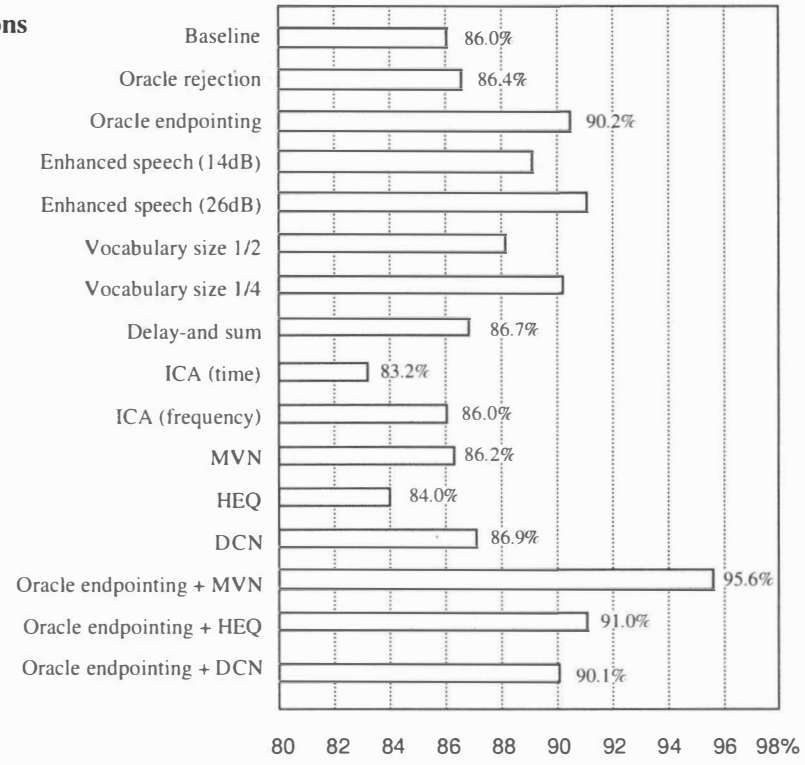

Figure 4: *Summary of Evaluation Experiments*

Next, two typical microphone array techniques were evaluated. The delay-and-sum beamformer [6] was implemented in a simple manner using fixed delay between microphones. The signals of seven microphones are first upsampled to 64kHz, then added delays, summed each other, and downsampled to 16kHz again. ICA [7] was tested in the time and frequency domains, using the microphone #3 and #5, next to the central microphone #4 on the both sides. The results show that the delay-and-sum beamformer provides small improvement (0.7% absolute), but ICA does not improve the recognition rate at all. We also tried ICA using seven microphones, but we had even larger degradation. These results indicated that the majority of the noise was non-directional, and some of them are correlated with each other due to reverberation.

In the feature domain (MFCC), we tried three normalization techniques, Mean and Variance Normalization (MVN), Histogram Equalization (HEQ) [8], and Delta-Cepstrum Normalization (DCN) [9]. These techniques were tested with our original decoder which had equivalent ability to Julian. The results of these feature normalization methods did not show any improvements. However, when we applied these methods with the oracle endpointing information, we got high recognition rates. In particular, MVN showed excellent and the highest recognition accuracy with the oracle endpointing, and the recognition rate was 95.6%, which was 4.5% absolute better than CMN with the oracle endpointing. It indicates that the effectiveness of MVN is highly dependent on good estimation of the speech segment. Contrastingly, HEQ and DCN could not improve the recognition rate even with oracle endpointing. Since these techniques have more parameters to estimate, short utterances in these experiments would not be sufficient for them.

4. EMBEDDED VERSION OF JULIUS

4.1 T-Engine with *SuperH* Microprocessor

T-Engine is a developmental hardware platform which has network security architecture and common Operating System (OS) called eTROM [10]. The T-Engine board consists of a CPU board, an LCD board, and a debugging board. Figure 5 shows a photo of T-Engine and Table 2 shows T-Engine (MS7751RC01) specifications. In this work, we used Hitachi's *SuperH* microprocessor called SH-4 which has 240 MHz CPU power on T-Engine.

The SH-4 is a RISC processor which has 32bit floating point calculation, and cache access commands. The work memory has 64MBytes, but only 55MBytes can be used for embedded software implementation. To implement Julius software on T-Engine, hardware modification was done for 16kHz sampling frequency and analog noise reduction.

4.2 Embedded CSR Implementation

Julius is free/open continuous CSR software [11] which can recognize large vocabulary over 20,000 words, and running on PCs which have huge computing resources. Table 3 shows specifications of the embedded version of Julius implemented on T-Engine. Two conditions were checked to realize a real time processing. The first condition was a monophone type for acoustic models, and the second condition was a triphone type. For the language model, both conditions had bigram and trigram. The word accuracy rates on PCs were 86.05% and 90.65% for 5,000-word vocabulary size, respectively. Pre-evaluation for implementation of this Julius PC software on T-Engine showed far beyond the real time processing. Especially the initialization of acoustic models and program calling process needed over 10 minutes for 5,000-word vocabulary recognition.

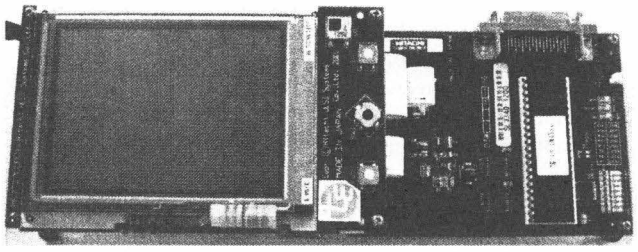

Figure 5: *T-Engine Board*

Table 2: *T-Engine Specifications*

CPU	*SuperH* SH-4 (240MHz/430MIPS)
Flash Memory	8 Mbyte
Work Memory	64 Mbyte
OS	T-Kernel
Input/Output I/F	USB(Host), PCMCIA card, Serial, Headphone output, Microphone Input, LCD I/F, Extended buss I/F, etc.
LCD board	TFT color monitor 240×320
Size	120 mm × 75 mm

Table 3: *Embedded Julius Specifications*

	Condition 1 (CND1)	Condition 2 (CND2)
Vocabulary size	5,000	←
Acoustic Models	Monophone	Triphone (PTM)
Language Models	bigram trigram	←
Beam width	400	←
Word accuracy (results on PC)	86.05%	90.65%

4.3 Developmental Issues

To realize the embedded version of Julius on T-Engine, the developmental issues are summarized as follows;

(1) CPU computing burden reduction:

The CPU power of T-Engine is restricted. Currently, the normal CPU power of PCs has been over 1.0GHz, however the T-Engine CPU power is around 200MHz. Therefore, CPU burden reductions are necessary to realize real time processing on T-Engine.

(2) Memory burden reduction:

Especially, limitation of memory capacity on T-Engine is a fatal issue comparing to PCs which have over 1G-Byte memory size. Usually, 100M-Byte size is a maximum memory size on embedded board environments. Therefore, memory reductions are needed for the embedded use.

4.4 Preliminary Computing Process Reduction

(1) Compact Acoustic Models

The Julius is using an HTK format. The HTK format for acoustic models is Ascii type resulting that the model memory size is 12MByte. The binary encoding from Ascii encoding of acoustic models could lead huge memory size reduction from 12MByte to 3MByte.

(2) Addlog Table Memory Reduction

The addlog calculation was done before recognition process using a logarithm table. By this table memory assignment modification, huge memory reduction was done from 2M Byte to 2k Byte and no recognition accuracy distortion occurred.

(3) MFCC Calculation Speed-up

Speech parameter extraction of MFCC (Mel Frequency Cepstrum Coefficient) is reduced using cos. and sin. tables.

4.5 GMS Process Reduction

(1) GMS Calculation Burden

The GMS (Gaussian Mixture Selection) is a method to select Gaussian distributions by the HMM states using a hierarchical relationships between monophone models and triphone models [12]. This GMS method is introduced to reduce acoustic likelihood calculation, however the more process reduction is needed because the GMS calculation is almost half of the total process time.

Figure 6 shows a GMS procedure. For each frame of input vectors, Gaussian mixtures for all monophone HMM states are computed (Pre-selection) and then Gaussian mixtures of triphone models are calculated for the only k-best states of monophone HMM models.

(2)Modifications on GMS

We made two modifications on the GMS method as follows;

(i) Computational Reduction on Pre-selection

The pruning process of a speech recognition decoder is done by the hypotheses that if scores are below a beam threshold, the calculations in pruned HMM states are not necessary. The only HMM states in active nodes need to be calculated.

In the conventional GMS, the scores of all monophone HMM states are calculated in a pre-selection stage. Knowing information of active nodes at the pre-selection stage, the only monophone HMM states linked to the active nodes can be calculated. Figure 7 shows an image of the modified strategy. Filled circles are designated HMM states whose hypotheses stay alive, and unfilled circles are designated HMM states whose hypotheses are pruned. Figure 7(a) shows the conventional GMS. The pdf scores of all states are calculated and k-best states from among them are selected (meshed area). Possibly some states of k-best states have no active hypothesis. It is useless to calculate a pdf score of pruned hypothesis. Figure 7(b) shows the monophone models for the proposed and modified GMS. The target of pdf calculation is restricted to HMM states whose hypotheses stay alive (within a bold circle). Applying the modification, the computational cost is reduced for the mixture selection of the GMS compared to the conventional GMS. Furthermore, since there is no fear of selecting the useless states whose hypothesis is pruned, a small number of k-best could be specified without any degradation of recognition accuracy.

(ii) Gaussian Selection within HMM State

The Gaussian Selection (GS) [13] is based on the idea that "Score calculated by a Gaussian neighboring an input vector is dominant on score of HMM state." On the other hand, all Gaussians within HMM state are calculated in the original GMS, even though scores derived from Gaussians distant from input vectors are negligible.

For reducing calculations of scores on HMM states, we change calculation strategy: calculating only neighbor

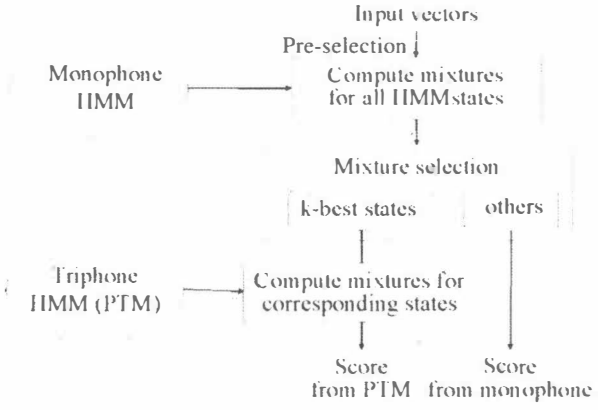

Figure 6: *GMS (Gaussian Mixture Selection)*

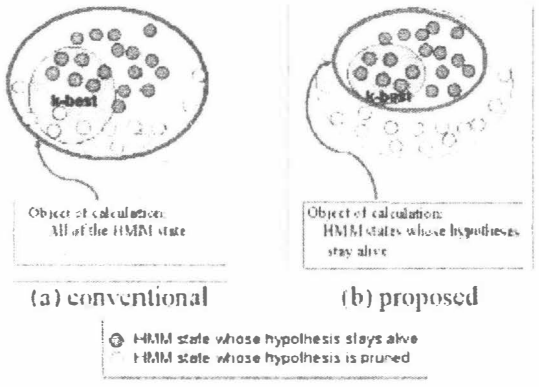

Figure 7: *Modification on Mixture Selection Stage*

Gaussians of input vector, instead of calculating all Gaussians. Figure 8 shows details of our strategy. The g_{max} is Gaussian maximum mixture score in an HMM state of monophone HMM. It is plausible idea that neighbor Gaussians of input vector in HMM state of triphone HMM are close to g_{max}. Based on this idea, only neighbors of g_{max} are calculated on HMM state, others, which are far from g_{max}, are omitted to calculate. By this procedure computation cost is much more reduced.

Distances between Gaussians can be calculated and stored in hash tables in advance.

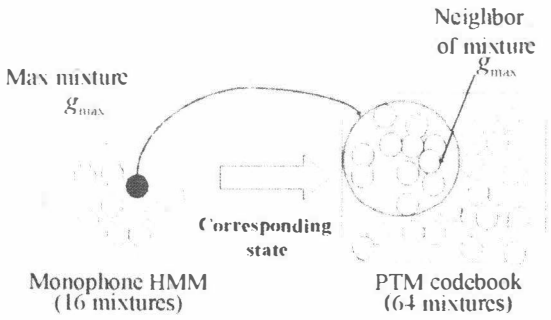

Figure 8: *GS (Gaussian Selection) in HMM state*

5. Evaluation Experiments

5.1 Experimental Setup

Table 4 shows experimental conditions for Julius software. The vocabulary size was 5,000 words and the triphone models had 3,000 states and 64 mixtures. The monophone models had 129 states and 16 mixtures.

Table 4: *Experimental setup for Julius*

vocabulary size	5,000 words
Triphone (PTM)	3,000 states, 64 mixture
Monophone	129state, 16 mixture

5.2 Evaluation Results

(1) Preliminary Computing Process Reduction on T-Engine

There is no approximated processing in this preliminary computing process reduction. This means no recognition rate distortion occurs by this process reduction. However, the distortion by the board noise may occur, so we tested T-Engine performance evaluation first. For the T-Engine performance evaluation, we used line input from PC file speech to avoid utterance varieties and environmental noise varieties.

In details, the following procedures are used for the T-Engine board evaluation. First, the input speech is input to T-Engine from PC using a line input, and then the speech input is stored to the flash memory attached to T-Engine. This speech file is incorporated by the T-Engine internal noise. Utterances by 30 males and 30 female were used for the evaluation. Table 5 shows board evaluation results. In the table, the original shows results of file speech input meaning digitalized data by PC. This means recognition results of original are top recognition rates. Two conditions, monophone and triphone are set in the evaluation. The recognition rates of the condition 2 were 89.1% for original and 85.9% for T-Engine showing 5% recognition accuracy distortion by the T-Engine board.

Table 5: *Evaluation Results(1): word accuracy(ACC)*

	CND1: monophone		CND2: triphone	
	Original	T-Engine	Original	T-Engine
Male 30	78.2%	72.7%	86.7%	83.7%
Female 30	84.5%	79.7%	91.9%	88.2%
Total	81.3%	76.2%	89.1%	85.9%

(2) GMS Process Reduction

First, the recognition performance of the proposed GMS process reduction method was evaluated by the PC (Linux: Pentium4 2.8GHz) simulation. Figure 9 shows the evaluation results. Evaluation data were 100 sentences for one male and one female from JNAS speech corpus. From the results, we found less k-neighbor value showed less word accuracy and the proposed GMS reduction method showed significant computing process time reduction (40% reduction) with small word accuracy loss (only 1%).

Next, the performance on the T-Engine (SH-4, 240MHz/430MIPS) platform was evaluated. The evaluation data were sentence utterances from 30 males and 30 females. The no. of k-neighbor was 24. Table 6 shows evaluation results on T-Engine. We found that requirement for the embedded Julius is less than 50MByte and that there was no big difference on word accuracy among no GMS, original GMS and the proposed GMS. The RTF (Real Time Factor) shows process length normalized by the utterance length. The proposed GMS showed 2.23 of RTF resulting 79% of that of no GMS. By the simulation, the process reduction by the proposed GMS was 40%. This difference may be occurred form T-Engine architecture and usage of cache memory.

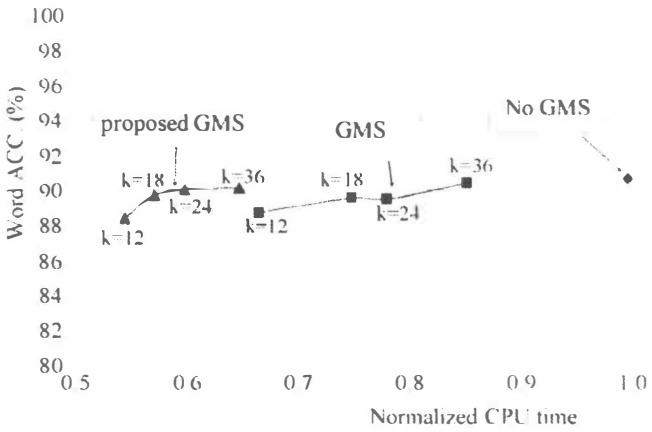

Figure 9: *Method Comparison: CPU time vs. Word ACC*

Table 6: *Evaluation Results(2) on T-Engine*

	Memory size	word ACC	RTF
no GMS	48.9MBytes	89.1%	2.81
original GMS	49.8MBytes	89.3%	2.62
proposed GMS	49.8MBytes	89.7%	2.23

6. FUTURE WORK

We will investigate more compact and more noise robust embedded version of Julius which has 20,000 word vocabulary sizes. The new CPU processor SH-4A (400MHz/700MIP) will be used to get fast processing time, For the noise robustness, we are developing a new noise reduction process module at the front-end.

7. SUMMARY

This paper describes two issues. First, we report huge evaluation results of recognition modules for the noise robustness. We found the Mean and Variance Normalization (MVN) showed excellent and the highest recognition accuracy with the oracle endpointing.

Second, we have developed an embedded version of the Julius continuous speech recognition (CSR) software on general-purpose microprocessors. We used T-Engine™ as a hardware platform. We could realize about 2.00 of RTF(Real Time Factor) of CSR processing on the condition of 5000-word vocabulary.

ACKNOWLEDGEMENT

The authors are thankful to Prof. Sadaoki Furui of Tokyo Institute of Technology and Prof. Tetsunori Kobayashi of Waseda University for their valuable comments. The work mentioned in the section 3 was supported by New Energy and Industrial Technology Development Organization (NEDO) in Japan. The work reported in sections 4 and 5 has been supported by the e-Society Project founded by Ministry of Education, Culture, Sports, Science and Technology in Japan.

REFERENCES

[1] Julius – an Open Source Large Vocabulary CSR Engine, http://julius.sourceforge.jp/en/julius.html/

[2] N. Hataoka, K. Kokubo, Y. Obuchi, and A. Amano, "*Development of Robust Speech Recognition Middleware on Microprocessor*," Proc. of IEEE ICASSP1998, pp.II837-II840, 1998.

[3] Y. Obuchi, et al. "*Development and Evaluation of Speech Database in Automotive Environments for Practical Speech Recognition Systems*," in appearing in Proc. of Interspeech2006, Pittsburgh, PA, USA, Sept. 2006.

[4] H. Kokubo, et al., "*Embedded Julius: Continuous Speech Recognition Software for Microprocessor*," in appearing in Proc. of MMSP2006, Canada, Oct., 2006

[5] N. Hataoka, et al., "*Robust Speech Dialog Interface for Car Telematics Service*," Proc of IEEE CCNC2004, Las Vegas, Jan., 2004

[6] W. Kellermann, "*A Self Steering Digital Microphone Array*," Proc. of ICASSP1991, Toronto, Canada, 1991.

[7] N. Murata, S. Ikeda, and A. Ziehe, "*An Approach to Blind Source Separation Based on Temporal Structure of Speech Signals*," BSIS Technical Report, 00-6, 2000.

[8] A. de. La Torre, et al., "*Non-linear Transformation of the Feature Space for Robust Speech Recognition*," Proc. of ICASSP2002, Orland, FL, USA, 2002.

[9] Y. Obuchi and R.M. Stern, "*Normalization of Time-derivative Parameter Using Histogram Equalization*," Proc. of EUROSPEECH, Geneva, Switzerland, 2003.

[10] T-Engine: http://www.t-engine.org/index.html

[11] A. Lee, T. Kawahara, S. Doshita, "*An Efficient Two-pass Search Algorithm using Word Trellis Index*", Proc. of ICSLP, pp.1831-1834, 1998.

[12] A. Lee and T. Kawahara and K. Shikano, "*Gaussian Mixture Selection using Context-Independent HMM*," Proc. of IEEE ICASSP2001-1-18, 2001.

[13] K. M. Knill, et al., "*Use of Gaussian Selection in Large Vocabulary Continuous Speech Recognition using HMMs*," in Proc. of ICSLP, vol. 1, pp. I-470-I-473, 1996.