EMBEDDED JULIUS ON T-ENGINE PLATFORM

Nobuo Hataoka[1] and Hiroaki Kokubo[1],
Akinobu Lee[2], Tatsuya Kawahara[3], and Kiyohiro Shikano[4]
[1]: Central Research Laboratory, Hitachi Ltd, Kokubunji, Tokyo 185-8601, JAPAN,
{nobuo.hataoka.ss, hiroaki.kokubo.dz}@hitachi.com
[2]: Nagoya Institute of Technology, Nagoya, JAPAN, ri@nitech.ac.jp
[3]: Kyoto University, Kyoto, JAPAN, kawahara@i.kyoto-u.ac.jp
[4]: Nara Institute of Science and Technology (NAIST), Nara, JAPAN, shikano@is.naist.jp

ABSTRACT

In this paper, we report implemental results of an embedded version of Julius. We used T-Engine™ as a hardware platform which has a *SuperH* microprocessor. The Julius is free and open Continuous Speech Recognition (CSR) software running on Personal Computers (PCs) which have huge CPU power and storage memory size. The technical problems to make Julius for embedded version are computing/process and memory reductions of Julius software. We realized 2.23 of RTF (Real Time Factor) of embedded speech recognition processing on the condition of 5000-word vocabulary without any recognition accuracy degradation.

Keywords

Automatic Speech Recognition (ASR), Continuous Speech Recognition (CSR), Julius: Free CSR Software, Embedded Julius, Hardware Platform: T-Engine, *SuperH* Microprocessor, GMS (Gaussian Mixture Selection), HMMs (Hidden Markov Models)

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 memory 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 report implementation results of the CSR software Julius on T-Engine consisting of a *SuperH*™ microprocessor[3]. 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. SPEECH PROCESSING ENVIRONMENT

2.1 System Image for Applications

As information technology expands into the mobile environments to provide ubiquitous communication, an intelligent interface 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 and portable devices. Mobile terminals such as cellular phones, PDAs (Personal Digital Assistants) and Hand-held PCs are already connected to networks such as the Internet to access information from web servers. For effective use of mobile information access, speech processing and image

0-7803-9733-9/06/$20.00©2006 IEEE

processing will be key technologies for intelligent mobile terminals[4]. 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.

As the speech processing environments, there are hardware devices such as CPU and memory which the speech processing including speech recognition and speech synthesis runs on, and communication infrastructure to connect to application servers. Figure 1 shows a system image consisting of terminal/client, Internet, and center/server. The processing devices are hardware-related devices such as PCs, microprocessors, and memories. The communication infrastructure includes wired and wireless environments.

2.2 Hardware Needs for Media Processing

Figure 2 summarizes hardware needs for mobile terminals such as HPC (Hand-held PC) and PDA. In the case of multi-language

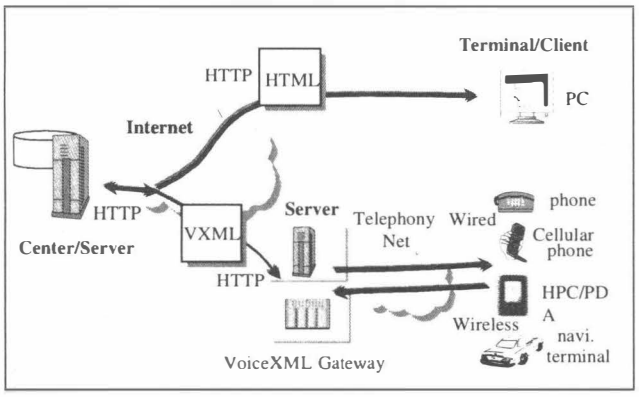
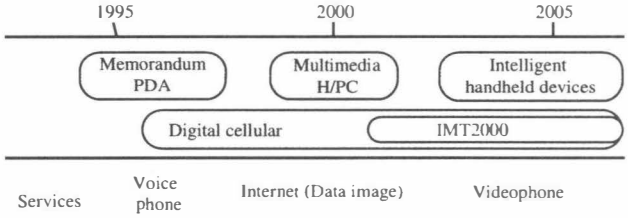


Figure 1: *System Image (Terminal, Network, and Centre)*



Services	Voice phone	Internet (Data image)	Videophone
		Voice Recognition	Speech Translation
Processor	50-100 MIPS (0.1-0.3 W)	300-500 MIPS (0.1-0.3 W)	1-2 GIPS (0.1-0.3 W)
Memory	< 1 MB	10-20 MB	> 100 MB

Figure 2: *Hardware Needs for HPC/PDA Applications*

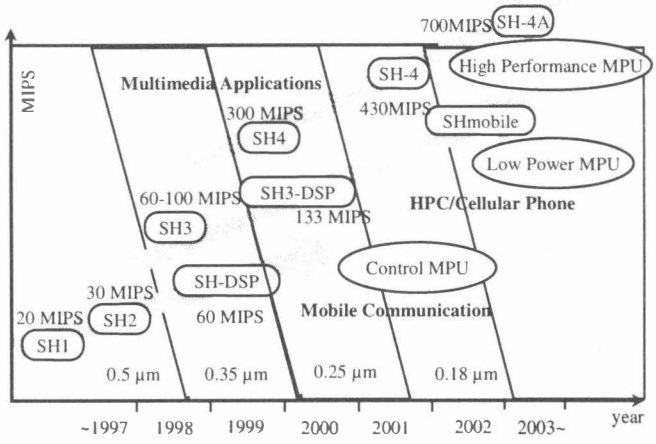


Figure 3: *Road Map for Microprocessors (SuperH Series)*

speech translation, CPU over 2 GIPS (Giga Instruction Per Second) and memory size over 100MByte will be needed. Due to improvements in microprocessor performance , various media processing technologies such as MPEG Codec and speech processing are possible to realize by software implementation. Figure 3 shows a road map for microprocessor such as *SuperH* series

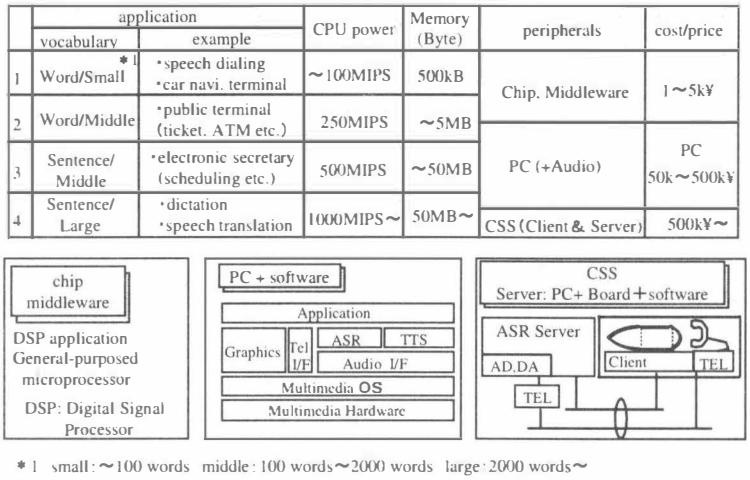
The *SuperH* microprocessors are products of Renesas and Hitachi Ltd. and there are 3 types depending on applications. The first type is for mobile communication using control MPU such as SH2. The second one is for HPC/cellular phone applications using SH3 series which are characterized by low power MPU. The third type is for multimedia applications using high performance MPU, SH4 series.

2.3 Implementation Varieties of ASR

Figure 4 shows implementation varieties of Automatic Speech Recognition (ASR) according to speech applications. Depending on the processing power and cost, there are three types of structures. The first one is chip and middleware on microprocessors. The second one is PCs and software implementation, and third one is a CSS (Client and Server System) structure.

To summarize, around 500MIPS CPU power and 50MBye memory size have been currently available by one microprocessor and these hardware environments will make it possible to implement continuous speech recognition software on a microprocessor.

In this paper, we have implemented the embedded version of Continuous Speech Recognition software called Julius using the T-engine platform consisting of SH-4 microprocessor.



*1 small: ~100 words middle: 100 words~2000 words large: 2000 words~

Figure 4: *Implementation Varieties of ASR*

3. EMBEDDED VERSION OF JULIUS

3.1 Free/Open CSR Software JULIUS

Julius is free and open CSR software which has been developed by Japanese Universities and delivered by WEB[1]. Julius can recognize large vocabulary over 20,000 words and running on Personal Computers (PCs) which have huge computing resources.

3.2 T-Engine with *SuperH* Microprocessor

T-Engine is a developmental hardware platform which has network security architecture and common Operating System (OS) called eTROM[5]. 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 1 shows T-Engine (MS7751RC01) specifications. We used Hitachi's *SuperH* microprocessor called SH-4 which has 240 MHz/430MIPS 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.

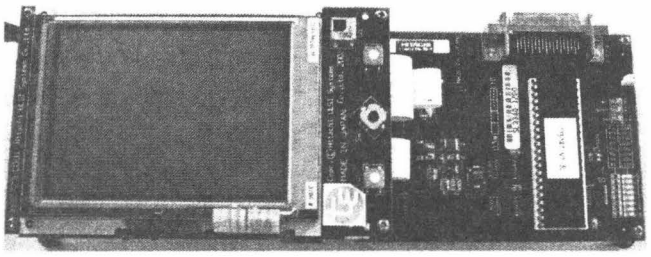


Figure 5: *T-Engine Board*

3.3 Embedded CSR Implementation

Julius is free/open continuous CSR software[6] which can recognize large vocabulary over 20,000 words, and running on PCs which have huge computing resources. Table 2 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.

Table 1: *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 2: *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%

3.4 Implemental 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, huge 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 Bytes memory size. Usually, 100M Bytes is a maximum memory size on embedded board environments. Therefore, huge memory reductions are needed for embedded use.

3.5 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.

3.6 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[7]. 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

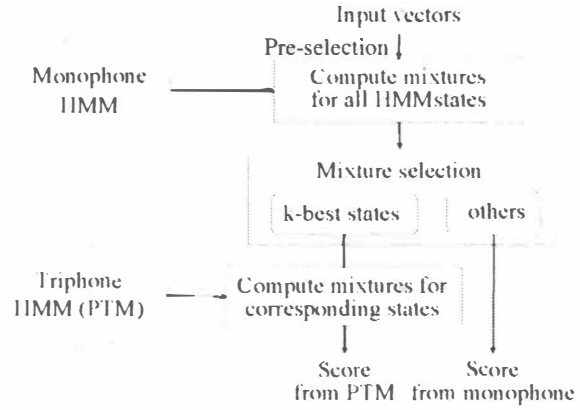


Figure 6: *GMS (Gaussian Mixture Selection)*

input vectors, Gaussian mixtures for all monophone HMM states are computed 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

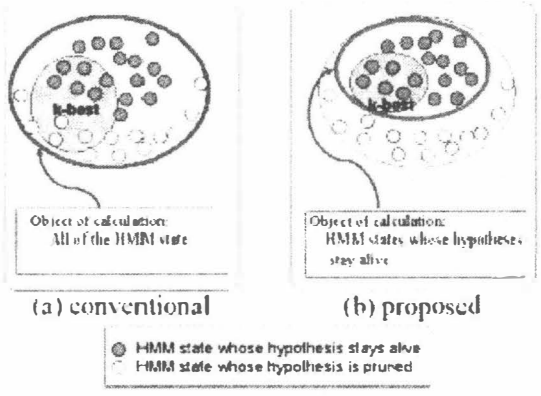


Figure 7: *Modification on Mixture Selection Stage*

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)[8] 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 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.

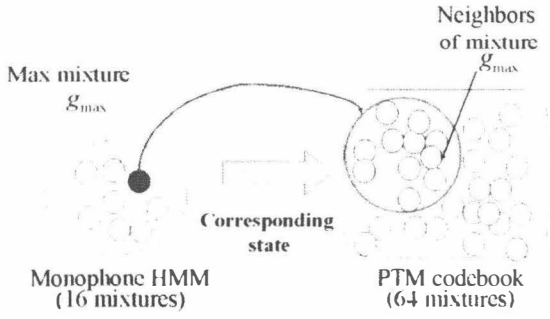


Figure 8: *GS (Gaussian Selection) in HMM state*

4. Evaluation Experiments

4.1 Experimental Setup

Table 3 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 3: *Experimental setup for Julius*

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

4.2 Evaluation Results

(1)Preliminary Computing Process Reduction

There is no approximated process 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 4 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 the top recognition rates. Two conditions, monophone and triphone are set in the evaluation. The recognition rates of the condition 2 with triphone were 89.1% for original and 85.9% for T-Engine showing 5% recognition accuracy distortion by the T-Engine board.

Table 4: *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 each one male and one female from Japanese 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 5 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.

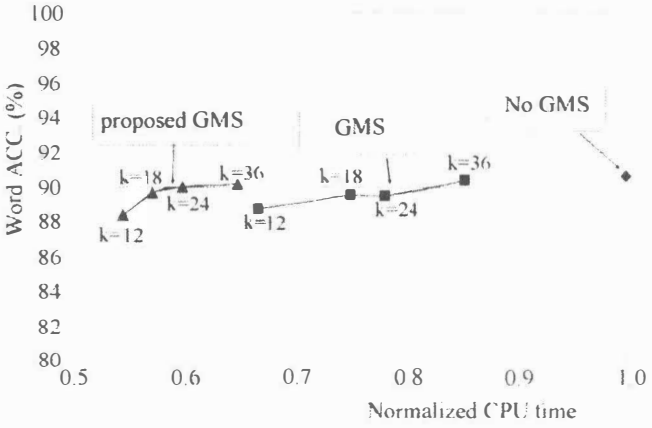


Figure 9: *Method Comparison: CPU time vs. Word ACC*

Table 5: *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

5. FUTURE WORK

We will investigate more compact and more noise robust embedded version of Julius which has 20,000-word vocabulary size. The new CPU processor SH-4A (400MHz/700MIPS) 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.

6. SUMMARY

This paper describes implementation issues of Embedded Julius. We have developed an embedded version of the Julius continuous speech recognition (CSR) software on general-purpose microprocessors. We used T-Engine™ (SH-4, 240MHz/430MIPS) as a hardware platform. We could realize 2.23 of RTF (Real Time Factor) of CSR processing on the condition of 5000-word vocabulary.

ACKNOWLEDGEMENT

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

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

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

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

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

[8] 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.