HIGH PERFORMANCE SPEAKER AND VOCABULARY INDEPENDENT ASR TECHNOLOGY FOR MOBILE PHONES

Sergey Astrov, Josef G. Bauer

Siemens AG, Corporate Technology
81730 Munich, Germany
{sergey.astrov, josef.bauer}@mchp.siemens.de

Sorel Stan

Siemens AG, ICM Mobile Phones
81675 Munich, Germany
sorel.stan@mch.siemens.de

ABSTRACT

This paper presents the Siemens speech recognizer for mobile phones, VSR. VSR employs HMM technology and uses general-purpose phoneme-based acoustic models which make it speaker and vocabulary independent. The system can be easily reconfigured to work with arbitrary vocabularies. This provides full flexibility for the design of the user interface which contrasts with the capabilities of other low-resource recognizers.

The system requirements of VSR are very low. The emission probability calculation and the Viterbi search with a vocabulary of 30 words need only 16 MHz for real-time operation on an ARM microcontroller. The HMM acoustic models take up about 12 kilobytes of permanent storage. The most significant algorithmic improvement is the newly developed 3-D stream-based coding of the HMMs.

Despite low requirements in terms of system resources VSR achieves an outstanding recognition performance. The word error rate (WER) for a recognition task with 62 German isolated words including highly confusable digits is 7.0%.

1. INTRODUCTION

Mobile phones are one of the most attractive areas of application of speech recognition technology. Increasing demand for hands-free operation of mobile phones in cars and device miniaturization make speech recognition a natural choice for the man-machine interface. While dictation is beyond the capabilities of a terminal-based speech recognizer command-and-control functionality and voice dialing are very attractive applications. Currently several manufacturers offer mobile phones with voice interface — mainly for speaker dependent voice dialing.

The limited system resources of the mobile phones place strict restrictions on the memory and processing requirements of the speech recognizers. In the last two years several recognizers with low system requirements have been developed. Most of the implementations are not HMM-based being speaker-dependent only, i.e. the acoustic models must be trained by the user. Recently, an HMM-based recognizer requiring 50 MIPS and 1 MB of memory was described in [1]. The system can handle a medium-size 500-word vocabulary, but the memory requirements are too high for mobile phones. In [2] a recognizer with an 18-word vocabulary was presented which is able to run on a 1.28 MHz system achieving a WER=0.5% in a quiet office environment. The recognizer is based on whole-word HMM models which implies that the modification of the vocabulary requires a dedicated speech database and re-training of the HMMs.

We present the Siemens VSR recognizer that has very low demands on system resources and allows flexible modification of the vocabulary without the necessity to re-train the HMMs. The key algorithmic improvements which lead to the reduction of memory consumption and processing power are efficient triphone clustering that gives a small number of modeling units, and compression of the resulting HMMs using 3-D stream-based coding.

The structure of the paper is as follows. Section 2 introduces the Siemens VSR recognizer. The system resources of mobile phones are described and the restrictions on the implementation of a speech recognizer are discussed. Section 3 gives details on the context dependent Hidden Markov phoneme Models and presents the advantages over the use of context independent HMMs. In Section 4 the memory and computational effective 3-D streams approach used in VSR is described. Recognition results and real-time performance are presented in Section 5. The paper ends with a brief summary and the conclusions.

2. VSR RECOGNIZER

The Siemens VSR is a low-resource high-performance recognizer adapted to the system and operating constraints of mobile phones. Modern mobile phones are powered by a digital signal processor (DSP) and a microcontroller unit (MCU). The DSP runs the GSM base-band algorithms and performs acoustic processing functions (e.g. echo cancellation and noise reduction). The MCU runs the user interface software and other applications such as address book management, games, or speech recognition. Permanent storage of data is typically provided by a flash memory which is shared by all applications, therefore the VSR acoustic models cannot claim too much for themselves.

The VSR architecture consists of three main modules for feature extraction, emission probabilities computation, and Viterbi search for the best matching vocabulary entry. Although the speech recognizer can be implemented entirely on the MCU, a balanced work split between the limited computational resources of a mobile phone justifies running the feature extraction on the DSP.

The feature vectors are computed as follows: the speech signal sampled at 8 kHz is cut into overlapping frames of 32 ms with a frame shift of 15 ms. For each frame a set of 12 MFCC values are computed which together with the frame energy and the corresponding delta and acceleration coefficients form a 39-D vector. Applying a linear discriminant analysis (LDA) based transformation to the supervector obtained from two consecutive frames, a 24-D feature vector is finally obtained.

VSR uses the following acoustic models for mobile phones ap-

plications. Each word is modeled using a set of context dependent phonemes, each phoneme is modeled by a 6-state HMM. Observation densities are 24-D Gaussians with diagonal covariance matrix and only one global variance parameter. Gaussians are coded by eight 3-D streams. The stream approach is discussed in Section 4.

3. IMPROVED PHONETIC MODELING

Earlier versions of the VSR used context independent phoneme models. To achieve a high recognition accuracy a total number of 4000 Gaussian densities was necessary. With a total number of modeling units of about 100 the number of Gaussians per modeling unit was very high at about 40.

The latest versions of the VSR use context dependent phoneme models. For the new modeling scheme with much more modeling units a total number of 1200 Gaussians was found to perform as good as context independent models with 4000 Gaussians. Table 1 gives a comparison of the two different approaches for 6 languages. Training and recognition are performed on the Speech-

Language	word error rate [%]	
	context indep. phoneme models, 4000 Gaussians	context dependent phoneme models, 1200 Gaussians
German	6.9	6.8
Spanish	0.8	1.2
French	4.9	5.5
Italian	1.9	1.3
Dutch	5.9	5.9
US-Engl.	6.0	5.7

Table 1. Comparison of two different approaches for acoustic modeling, isolated command word recognition.

Dat II, the Polyphone, and the Macrophone speech databases.

The investigated recognition task is isolated application word recognition with about 50 words in the vocabulary. The test results which are based on more than 11000 utterances in total show that the much smaller context dependent models perform as well as the context independent models (the differences are not statistically significant).

With the number of modeling units being now about 600 the average number of Gaussians per unit is as small as 2. The actual number of Gaussians per unit is not equally distributed. Units with a higher frequency in the training material have a higher number of Gaussians while the maximum number of Gaussians per unit is limited. For a limited vocabulary a small number of Gaussians per unit has a positive effect on the complexity of the emission computation. Using context independent phoneme models (small amount of units with many Gaussians) even small vocabularies can result in a need for a big amount of Gaussians for decoding. With the context dependent phoneme models (many units with only few Gaussians each) a small vocabulary normally requires only a small amount of Gaussians for decoding. This can result in an even higher reduction of the computational complexity.

The modeling units are strongly tied triphone states. Data driven decision trees with additional a-priori tying rules as described in [3] are applied for tying. The a-priori rules result in an important property of the tying scheme: as the theoretically maximum number of units is much smaller than for untied triphones the tying scheme can be stored in the form of a simple table that uses as little memory as the decision tree. While the algorithm for decoding a decision tree is rather complex the tying table can be handled with an ultra-compact algorithm.

Through the use of strongly tied context dependent phoneme models memory consumption as well as processing power requirements could be reduced by about 70% without degradation of recognition performance in comparison to context independent models.

4. STREAM DISTRIBUTION CLUSTERING HMMS (SDCHMM)

The feature vectors $\vec{x}$ of the VSR recognizer have 24 components with each component being stored as a 1-byte value.

The emission probability of $\vec{x}$ for state s $b_s(\vec{x})$ is a sum of M Gaussians with diagonal covariance matrix and equal variances σ. As the sum is typically dominated by one term only the following approximation is made $\sum_{m=1}^{M}(\cdot) \approx \max_{m=1}^{M}(\cdot)$.

VSR uses the stream (subspace) distribution clustering HMMs (SDCHMM) introduced in [4] in order to reduce the memory consumption. The theory of SDCHMM with shared codebook and 3-D streams is briefly explained below. The P Gaussians from all mixtures form a set of 24-D vectors which is broken down into streams. Several stream structures were designed in [4]. In the case of VSR, the components of the stream vectors are almost uncorrelated because of the LDA transformation from the feature extraction module; that is why the simple structure is used [5].

We use 3-D streams of which the definition can be described as follows: the first 3 dimensions of Gaussians mean vectors (μ_1, μ_2, μ_3) are placed in the first stream, the second 3 dimensions (μ_4, μ_5, μ_6) form the second stream, etc. Thus the set of 24-dimensional Gaussians is broken onto eight 3-D streams.

The mixture emission probability of the state s is represented then as

$$b_s(\vec{x}) = \max_{m=1}^{M} \left(c_{s,m} \cdot \prod_{k=1}^{K} N_k(\vec{x}, \mu_{s,m}, \sigma) \right)$$

where $N_k(\vec{x}, \mu_{s,m}, \sigma)$ is a Gaussian PDF for stream k; $c_{s,m}$ is the weight of the Gaussian m in the state s; m is the Gaussian number within state s; k is the stream index; K is the number of streams (in our case $K = 8$).

All of the stream vectors are quantized using k-means algorithm. The parameters of the SDCHMM are represented by indices which point to values in one shared codebook. This shared codebook is used for all streams together. The shared codebook has 256 entries as in this case the indices are of size of one byte and they are easy to handle. With the 256-entry codebook 3-D streams are used as they have good memory efficiency and do not lead to a degradation of the recognition performance.

SDCHMMs also allow to reduce the complexity of the emission computation since stream vectors can have only 256 possible vector values. Then it is possible to precompute the partial log likelihoods for every stream and every codebook entry. These partial log likelihoods have to be computed only once for each frame. Then during emissions computation only additions are performed. For 3-D streams the reduction of total number of operations is about 66%. The emission computation algorithm is shown below.

1. Precomputation: Partial log likelihoods $H(\mu_{s,m}, k)$ (called in [6] *atom*) for the first three components of feature vector are

computed for all of 256 codebook entries and stored in the memory. This step is repeated for the next three components, etc.

2. Emission computation: The emission log likelihood $B_{s,m}(\vec{x})$ is computed using the following equation:

$$B_{s,m}(\vec{x}) = C_{s,m} + \sum_{k=1}^{K} H(\hat{\mu}_{s,m}, k)$$

where $C_{s,m}$ denotes the penalty (weight) of Gaussian m within state s; $\hat{\mu}_{s,m}$ denotes the code that points to the codebook; $H(\hat{\mu}_{s,m}, k)$ denotes the precalculated partial log likelihood for codebook entry m and stream k.

The Gaussian penalties $C_{s,m}$ are 16-bit integers. In order to reduce further the model storage requirements the penalties are coded using scalar quantization with a 256-values codebook. In our tests such coding does not increase the WER.

SDCHMMs with penalties coding are memory effective as it is only necessary to store the indices and the codebooks. The memory requirements of SDCHMM with 3-D streams are analyzed below. The shared codebook occupies 768 bytes, the whole set of Gaussian indices occupies 9600 bytes, the set of penalty indices occupies 1200 bytes and the penalty codebook occupies 512 bytes. Thus 12080 bytes of memory are required to store the quantized stream-based models. Note that the CDHMM with the same number of Gaussians takes up 31200 bytes. Therefore the relative memory saving done by SDCHMM with 3-D streams is 61%.

5. TEST RESULTS

5.1. Recognition Results

The comparison of the recognition performance of CDHMMs with that of 3-D stream SDCHMMs is shown in Table 2. We have tested CDHMMs and SDCHMMs on several tasks in German language that shows to be difficult from an ASR point of view (see Table 1). The results of the experiments for Spanish language are similar as for German language [5].

CDHMMs and 3-D stream SDCHMMs were tested with the following recognition sets of German language: VM62 (VoiceMail) and SDII-mbl-apl are isolated command word tasks with the vocabulary of size of 62 and 83 words respectively. SDII-c_d, SieTill-c_d, SDII-mbl-c_d and SDC-c_d are continuous digits (c_d) recognition tasks. The test sequences are of size of 500–13600 sentences. The utterances were recorded in different noise environments (telephone and mobile phone networks, moving cars). The databases VoiceMail, SpeechDat II / II mobile / Car and SieTill are available from ELRA ([7]).

The continuous digits recognition tasks SDC-c_d (recorded in moving cars) and SDII-mbl-c_d (recorded via mobile phone network and partially in moving cars) have the highest WERs. Note that all shown results are based on general purpose phoneme based acoustic models. Whole word models trained on digit material can of course result in lower error rates but are not investigated here.

From the recognition results of SDCHMMs and CDHMMs it can be seen that the recognition performance does not degrade. Thus, SDCHMMs have the similar recognition accuracy as CDHMMs but require less memory and allow to reduce the recognition time.

5.2. Performance Analysis

Portable devices, such as mobile phones and organizers, support dynamic clock rate configuration per software. Thus the battery life can be extended by lowering the clock rate in idle it also depends on the computational requirements of the applications. Since the power consumption and performance can be optimized for each task knowing the processor load factor of an application becomes very important.

Given that a program i consumes C_i clock cycles to run on a processor with the clock rate f its execution time E_i can be computed as follows

$$E_i = C_i \times \frac{1}{f}. \tag{1}$$

Most programs written for embedded systems must run under real-time constraints. They are typically implemented as periodic tasks with a constant interval between requests, D_i, called deadline. Each task must be completed before the next request for it occurs, so the constraint on the execution time is $E_i \leq D_i$. We can ask what is the minimum clock frequency f_i for the deadline to be met, and obtain

$$f \geq \frac{C_i}{D_i} \implies f_i = \frac{C_i}{D_i}. \tag{2}$$

For our speech recognizer every 15 ms a new feature vector is extracted from the digitized speech signal. Then the corresponding emission probabilities are computed and the Viterbi search scores for all vocabulary entries are updated. The deadline D is set by the frame shift of 15 ms. As mentioned in Section 2 the feature vectors could be calculated by the DSP, while the emission probability computation and the Viterbi search could run on the MCU. As the optimizations presented in this paper are not related to the calculation of feature vectors, we focus our performance analysis solely on the behavior of the code running on the MCU.

A critical performance measure for embedded systems is their throughput, i.e. the number of tasks that can use the processor in time-sharing without violating their respective deadlines.

For instance, for a voice-controlled MP3-player application it would be of interest to know if the speech recognition and the MP3 audio decoder could run simultaneously. Generally, let us assume that there are n different tasks which have execution times $E_1, \ldots, E_n$ and the same deadline D.

Since the processor must finish executing all the tasks before the deadline D, the following relation must be satisfied

$$E_1 + \ldots + E_n \leq D \implies \frac{E_1}{D} + \ldots + \frac{E_n}{D} \leq 1.$$

If the n deadlines are different, then we obtain the constraint

$$\frac{E_1}{D_1} + \frac{E_2}{D_2} + \ldots + \frac{E_n}{D_n} \leq 1. \tag{3}$$

Substituting in the above equation the clock frequencies and cycle counts from Eqs. 1 and 2, we arrive to

$$f_1 + f_2 + \ldots + f_n \leq f. \tag{4}$$

The ratio $\rho_i = E_i/D_i = f_i/f$ is called the processor load factor of the task i.

Eq. 3 (or equivalently Eq. 4) gives the necessary and sufficient condition for the n tasks to be schedulable.

We chose to benchmark our recognizer on the ARM RISC (Reduced Instruction Set Computer) microcontroller which is

HMM	word error rate [%]					
	VM62	SDII-c_d	SieTill-c_d	SDII-mbl-c_d	SDII-mbl-apl	SDC-c_d
CDHMM	6.6	7.6	9.9	11.6	6.6	13.4
SDCHMM 3-D	7.0	8.4	10.3	12.4	7.5	13.6

Table 2. Comparison of 3-D stream SDCHMM and CDHMM recognition performances.

HMM	Emission Prob. Calculation [MHz]			Viterbi Search [MHz]		
	ARM9TDMI	ARM920T	ARM940T	ARM9TDMI	ARM920T	ARM940T
CDHMM	26.2	28.9	30.3	8.3	9.1	13.5
SDCHMM 3-D	7.7	8.2	12.7	8.3	9.1	13.5

Table 3. Minimum computational requirements in MHz for a 30-word vocabulary.

widely used in embedded systems [8]. For this task we employed the ARM Instruction Simulator (ARMulator) supplied with the ARM Developer Suite v1.2.

ARM cores come in two basic flavors, Von Neumann and Harvard, depending on the memory access architecture. Von Neumann cores (e.g. ARM7TDMI family) use a single bus for both data and instruction accesses, while the Harvard cores (e.g. ARM9TDMI family) have a separate data and instruction bus, thus allowing simultaneous data accesses and instruction fetches.

Harvard cores are not normally employed in their raw state, but typically a cached variant with a Harvard cache architecture and a Von Neumann memory interface is used. However, benchmarking raw Harvard cores using an ideal zero wait-states memory model can be useful as an indication of the maximum achievable performance for a cached variant assuming 100% cache efficiency.

To have a reference for the maximum performance, our benchmarking is first done on the ARM9TDMI uncached core using the default ARMulator's model of zero wait-states 32-bit memory. Then we repeat the benchmarking on ARM920T (16 KB data and 16KB instruction cache) and on ARM940T (4KB data and 4KB instruction caches), for a system configured with the processor clock rate of $f = 100$ MHz, the bus clock rate of $f_{\text{bus}} = 50$ MHz, and a 32-bit memory with 100 ns non-sequential acess time and 20 ns sequential access time. The memory introduces 4 wait states for non-sequential R/W and 1 wait state for sequential R/W access.

We used in our benchmark a 5 s long utterance and a vocabulary of 30 words. Table 3 shows the minimum computational requirements, expressed in MHz, for real-time operation of the emission probability calculation and the Viterbi search using the CDHMM and SDCHMM with 3-D streams. The dependency of the processor load factor on the system configuration, e.g. cache size and memory access times, are clearly visible.

6. CONCLUSIONS

In this paper we presented the Siemens VSR speech recognizer. The system features context dependent Continuous Densities Hidden Markov phoneme Models that allow excellent recognition rates with arbitrary vocabularies. CDHMM parameters are compressed using our recently proposed 3-D stream based coding. The HMM acoustic models take up only 12 kilobytes of flash storage. On an ARM microcontroller, VSR requires about 16 MHz for real-time computation of the emission probabilities and the Viterbi search with a vocabulary of 30 words.

The flexibility of defining vocabularies and the small memory footprint of the acoustic models combined with the low-power requirements and the competitive recognition rates in noisy environments qualify the Siemens VSR as a high-quality speech recognizer for mobile phones.

7. REFERENCES

[1] S. Deligne et. al, "Low-resource speech recognition of 500-word vocabulary," in *Proc. European Conference on Speech Communication and Technology (Eurospeech)*, 2002, pp. 1833–1836.

[2] E. Cornu et. al, "An ultra-low power, ultra miniature voice command system based on hidden markov models," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2002, pp. 3800–3803.

[3] U. Ziegenhain and J. G. Bauer, "Triphone tying techniques combining a-priori rules and data driven methods," in *Proc. European Conference on Speech Communication and Technology (Eurospeech)*, 2001, pp. 1413–1416.

[4] E. Bocchieri and B. K.-W. Mak, "Subspace distribution clustering hidden markov model," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 3, pp. 264–275, 2001.

[5] S. Astrov, "Memory space reduction for Hidden Markov Models in low-resource speech recognition systems," in *Proc. Int. Conf. on Spoken Language Processing (ICSLP)*, 2002, pp. 1585–1588.

[6] A. Aiyer, M. J. F. Gales, and M. A. Picheny, "Rapid likelihood calculation of subspace clustered gaussian components," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2000, pp. 1519–1523.

[7] "ELRA internet site," 2000, `http://www.icp.grenet.fr/ELRA`.

[8] Application Note 93, "Benchmarking with ARMulator," March 2002, ARM DAI 0093A, ©ARM Ltd.