We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

9

Design and Applications of Embedded Systems for Speech Processing

Jhing-Fa Wang[1], Po-Chun Lin[2] and Bo-Wei Chen[1]
[1]*National Cheng Kung University*
[2]*Tung Fang Design University*
Taiwan

1. Introduction

This chapter focuses on speech processing techniques, which involve speech feature extraction, sound localization, speaker identification/verification, and interactive retrieval of spoken documents. Several hardware design issues are discussed in each section. Speech processing applications frequently involve extensive mathematical computation, making resource and power consumption management important. Therefore, this chapter presents not only algorithms but also their corresponding improved solutions to embedded systems, such as fixed-point arithmetic design, field-programmable gate array (FPGA) verification, ARM-based system-on-a-programmable-chip (SoPC) architecture, and other single-chip designs.

The rest of this chapter is organized as follows. Section 2 introduces the feature extraction method that is used in speech processing. Section 3 then describes details of the sound localization technique. Next, Sections 4 and 5 elucidate speaker identification/verification and interactive retrieval of spoken documents. Conclusions are finally drawn in Section 6, along with recommendations for future research.

2. Embedded system design for speech feature extraction

Speech feature extraction is critical in speech processing applications. This section describes in detail frequently used speech features and the design of chips for extracting them. The computational complexity and memory requirement of the associated algorithms are also analyzed in detail to ensure favorable performance. Furthermore, a hybrid approach for fixed-point arithmetic and hardware design is developed to ensure low computational complexity. Finally, a single FPGA development board is considered as a case study to realize the design.

2.1 Methodology

2.1.1 Algorithm for calculating mel-frequency cepstral coefficients

The complete step-by-step process for calculating coefficients is described as follows (Vergin et al., 1996; Wang et al., 2003).

Step 1. Short-time fast Fourier transform (FFT)

$$\begin{cases} Y(m) = \frac{1}{F}\sum_{n=0}^{F-1} z(n)w(n)W_F^{nm} \\ w(n) = \beta(0.5 - 0.5cos\frac{2\pi n}{F-1}) \end{cases} \quad (1)$$

Step 2. Find the energy spectrum, $X(m) = |Y(m)|^2$.

Step 3. Calculate the energy in each channel:

$$\begin{cases} S[k] = \sum_{j=0}^{F/2-1} W_k(j)X(j) \\ \sum_{j=0}^{F/2-1} W_k(j) = 1 \end{cases} \quad (2)$$

Step 4. Take the logarithm and perform the cosine transform to obtain the Mel-frequency cepstral coefficients (MFCCs),

$$C[n] = \sum_{k=0}^{M-1} \log(S[k])\cos[n(k+0.5)\frac{\pi}{M}] \quad (3)$$

2.1.2 Improved algorithm for calculating mel-frequency cepstral coefficients

Generally, the required computational power and ROM in each frame can be determined clearly according to Table 1 (Wang et al., 2000; Wang et al., 2003). As shown in the table, the total required computational power is quite high due to the redundant operations and memory that stores the required constants. Accordingly, some modifications must be made to reduce the computational load.

The weighted energy spectrum in the Mel-window, $E_{k+1}(j)$, can be obtained by subtracting the weighted energy spectrum $E_k(j)$ from energy spectrum $X(j)$. All of the multiplications in (4) can be replaced by subtraction operations. Therefore, the memory required to store the weight constants for (4) becomes redundant and can be eliminated.

$$E_{k+1}(j) = X(j)\frac{d-(L+D)}{(L+2D)-(L+D)} \quad (4)$$

Additionally, applying the symmetric property of the cosine function to (3) flattens all of the operations and enables related items to be combined in a new formula, given below.

$$C[n] = \sum_{k=0}^{M/2-1} \{log(S[k]) + (-1)^n log(S[M-k-1])\}cos[n(k+0.5)\frac{\pi}{M}] \quad (5)$$

Therefore, the computational complexity of $C[n]$ operations can be re-estimated, and the result is given in Table 2 (Wang et al., 2000; Wang et al., 2003).

	Number of operations and required memory		***K*=256, *M*=20, *L*=12**		
	Computational power		Actual computational power		
	C[*n*]	*S*[*k*]	*C*[*n*]	*S*[*k*]	Total
Addition/subtraction	*L*(*M-1*)	*M*(*F*/2-1)	12x19=228	20x127=2540	2768
Multiplication	*LM*	*MF*/2	12x20=240	20x128=2560	2800
Logarithm	*M*	0	20	0	20
ROM size (words)	*LM*	*MF*/2	12x20=240	20x128=2560	2800

Table 1. Number of operations and the required memory estimated using the original MFCC algorithm.

	Improvement of *C*[*n*] calculation		**Total improvement**			
Operation	Computational power	Improvements (%)	*C*[*n*]	*S*[*k*]	Total	Improvement (%)
Addition/ subtraction	*L*(M/2+M/2-1)	0	12x19=228	113	341	87.6
Multiplication	*LM*/2	50	12x10=120	128	248	91.1
Logarithm	*M*	0	20	0	20	0
ROM size (words)	*LM*/2	50	12x10=120	128	228	91.8

Table 2. Improvement of *C*[*n*] calculation by rescheduling the original MFCC algorithm and the total improvement provided by the proposed method

The modified procedure in the MFCC algorithm is based mainly on the improved *S*[*k*] calculation, as discussed below and shown in Fig. 1 (Wang et al., 2000). Every Mel-window is divided into two blocks with equal bandwidth on the Mel-scale. Because the Mel-windows overlap each other, every block except for the first and last belongs to two Mel-windows.

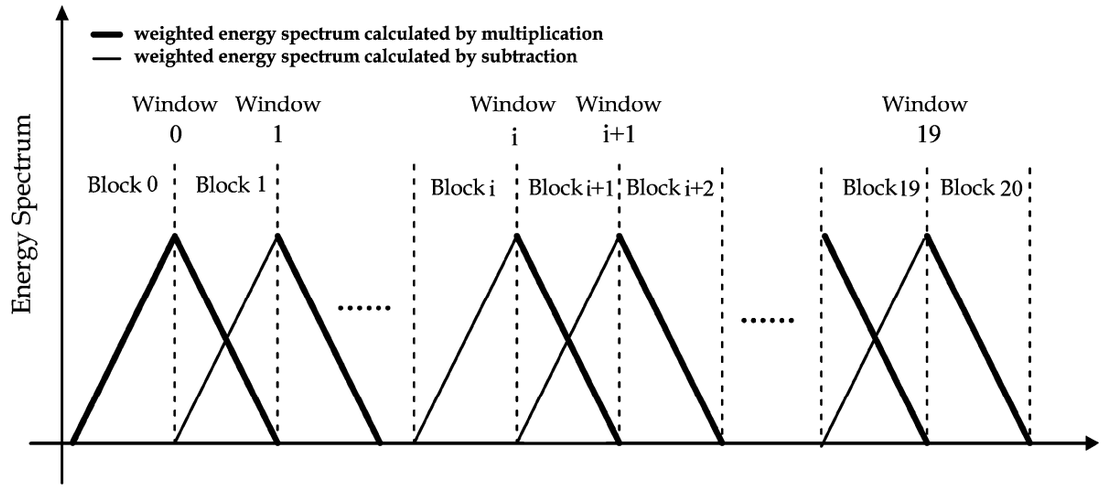


Fig. 1. Modified procedure for calculating *S*[*k*].

2.2 Hardware implementation

2.2.1 Fixed-point arithmetic design

The word recognition system is based on the hidden Markov model (HMM). To achieve area-efficiency, MFCC chips are designed using fixed-point arithmetic. The procedure for implementing the fixed-point program is as follows.

Step 1. Partition the algorithm into *n* modules; this involves calculations of the energy spectrum, channel energy, and MFCCs.
Step 2. Determine the lower bound and upper bound on each module.

The format of fixed-point variables is determined based on the dynamic range of the input variables in the first module. Once this module has been analyzed, the output is fed into the next module and analysis continued until all modules fit the fixed-point data format, as presented in Table 3 (Wang et al., 2003).

	Maximum	Minimum	Abs. minimum	Fixed-point format
Energy spectrum	3121190.0012	0.000311	0.000311	24.8
Energy in each channel	2172253.6092	0.124351	0.124351	24.8
MFCC	214.006766	-75.082199	1.567230	9.7
Logarithm value	5.336967	-0.905350	0.905350	5.11

Table 3. Analysis of dynamic range and determined fixed-point data format.

Step 3. Error measurement for each module

This step evaluates the quantization error by comparing their outputs with the output of the corresponding floating-point routines as shown in Table 4 (Wang et al., 2003).

	Energy spectrum (%)	Energy in each channel (%)	MFCC (%)
Word with vowel phonemes	0.020563623	0.022464977	0.474503142
Word with nasal phonemes	0.028652758	0.035190628	0.481810443
Word with fricative phonemes	0.031959564	0.039397264	0.698307547
Word with stop phonemes	0.052471004	0.057502398	0.454653425
Word with affricate phonemes	0.041492785	0.05067771	0.454107475

Table 4. Average error with the determined fixed-point data format.

Step 4. Performance measurement

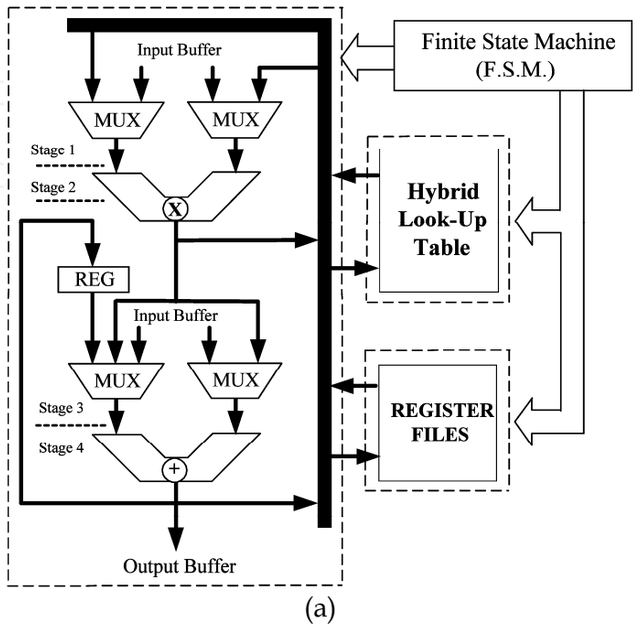
The impact of the recognition rate of the fixed-point MFCC algorithm is evaluated at this stage, as shown in Table 5 (Wang et al., 2003).

	User 1(%)	User 2(%)	User 3(%)	User 4(%)	User 5(%)	Average (%)
Floating-point	92.0	90.0	93.0	91.0	93.0	91.8
Fixed-point	91.0	88.0	91.0	88.0	90.0	89.6

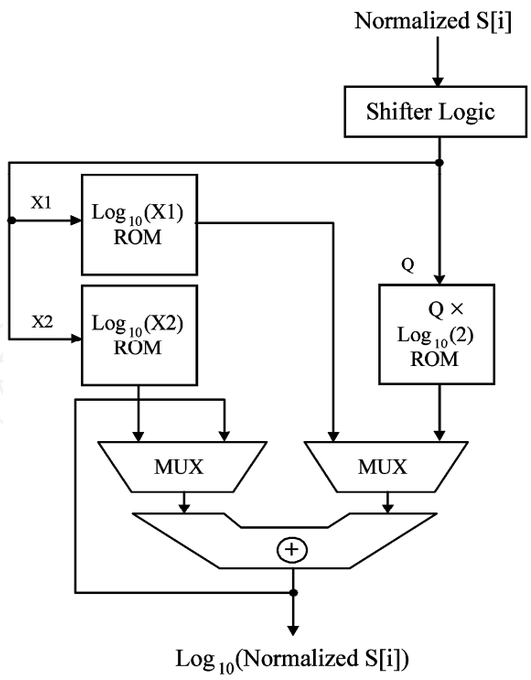
Table 5. Comparison of recognition rates achieved using floating-point and fixed-point structure.

2.2.2 Circuit design

The use of improved partitioned look-up tables is another commonly used method to perform such elementary functions as logarithm, square root, and trigonometric functions, for example. Figure 2(a) shows the proposed four-stage pipeline architecture.



(a)



(b)

Fig. 2. Circuit design of an embedded system. (a) Architecture of the proposed MFCC chip. (b) Architecture of the look-up table (Wang et al., 2003).

Based on such architecture, only one processing unit is used and all data are processed in pipeline fashion. Figure 2(b) displays the architecture of the improved partitioned table. The shifter logic is used to find the Q value of the minimum left shift and to output the least significant 16 bits, which are the addresses of the two subtables. Only one two-stage pipelined multiplier and adder, which is shared by both the main data path and the look-up table, is used.

At the verification stage, an FPGA board is utilized to implement the MFCC system prototype. First, synthesizable Verilog-HDL descriptions are coded. Synopsys FPGA Express (www.synopsys.com) generates the corresponding netlist files. The Xilinx Flow-Engine completes generating placement, routing, and bit-stream files. The design is implemented successfully in the XC4062XL FPGA chip.

3. Embedded system design for sound localization

This section introduces a sound localization system, which exploits the average magnitude difference function (AMDF), for finding the directions of environmental signal sources. To verify the accuracy of the algorithm, the entire system is implemented on a single FPGA development board using the Quartus II software tool. Then, the System-on-Chip (SoC) design, based on the FPGA code with the 0.18μm CMOS process, is implemented. The experimental results indicate that the proposed system can achieve higher accuracy with reduced complexity and area of the hardware.

3.1 Methodology

Figure 3 presents the overall architecture of the sound localization system, including a sound signal amplifier, an analogue-to-digital (AD) converter, a sound activity detector, and an AMDF module. External acoustic signals are received by a pair of microphones and magnified by an amplifier. The AD converter transforms analogue data to digital data. The sound activity detection block consists of threshold value detection, zero-crossing rate (ZCR), and end-point detection modules. Three methods are utilized to distinguish desired segments from silent periods. Finally, the AMDF module (Wang et al., 2008a; Wang et al., 2009) estimates the delay based on the desired signal segments, and converts the delay into angles. A brief workflow of the system is shown in Fig. 4

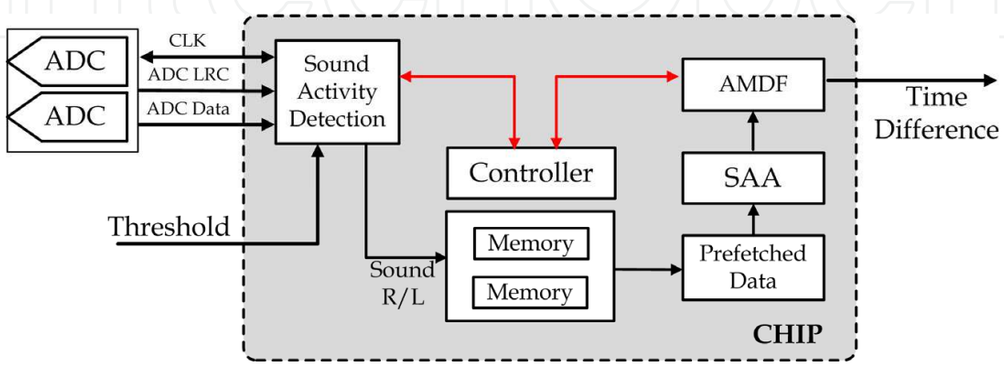


Fig. 3. Overview of the sound localization system (Wang et al., 2011).

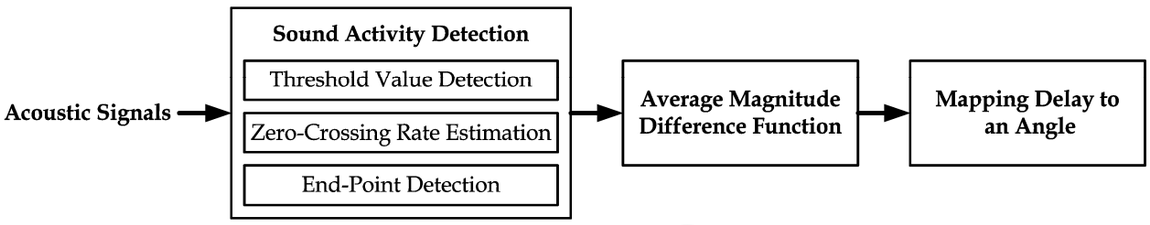


Fig. 4. Workflow of the sound localization system.

3.1.1 Sound activity detection

When the pair of microphones receives the sound signals, the system begins to determine whether the input signal needs to be handled. The detection of sound activity comprises three steps, which are as follows.

- Threshold Value Detection: Whether the amplitude of the input signal exceeds a threshold is determined by this step. If the amplitude exceeds the threshold, then the system begins to store the input signal data in memory.
- Zero-Crossing Rate Estimation: In acoustics, a sound wave has positive and negative values of displacement around the zero amplitude. Zero-crossing rates are calculated by counting the crossings of the baseline over time. The presence of an active ZCR signal can improve threshold value detection.
- End-Point Detection: An end-point beacon is generated when an ongoing input signal falls below the threshold for a preset period.

3.1.2 Direction-of-arrival estimation

Figure 5 displays a microphone array, where $x_1(t)$ and $x_2(t)$ represent the acoustic signals that are received by microphones 1 and 2 respectively; d denotes the distance between these two microphones; θ is the direction between the array and an unknown source, whose signal is represented as $s(t)$. The source is assumed to be far enough from the microphone array so that the acoustic wave-front that impinges upon the microphone array can be approximated as a plane wave. Let microphone 1 be the reference point; the relationship between the received signals and the source signal in the time domain is given by the equation,

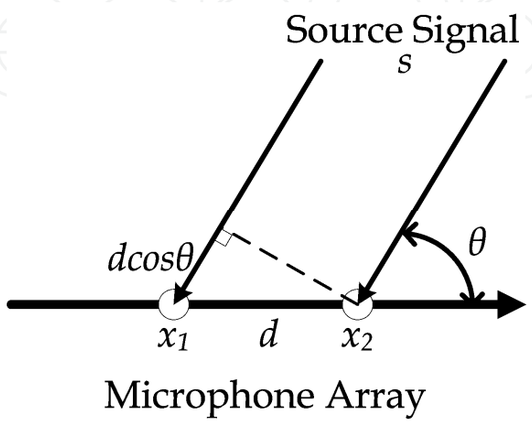


Fig. 5. Direction-of-arrival illustration.

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} s(t) \\ s(t-\tau) \end{bmatrix} \tag{6}$$

where τ is the propagation delay from the source to the microphone. As shown in the figure, after the wave-front impinges on microphone 2, the wave-front takes time "τ" to reach microphone 1. The distance between the wave-front and microphone 1 is $d\cos\theta$ (Chen et al., 2010). Therefore,

$$\tau = \frac{d\cos\theta}{c} \tag{7}$$

where c is the sound velocity. In real-life applications, noise and reverberation may distort wave shapes, potentially affecting the propagation delay. A feasible way to estimate the accurate time delay involves using the AMDF. The AMDF firstly fixes the signal at microphone 1, and then shifts the signal at microphone 2 to calculate the time delay. When both signals are the most similar, the difference between the waves will be minimized. In other words, the τ value is obtained when the correlation between the waveforms of the both microphone signals is maximal. Let N be the total number of windows and i represent the sliding window index. The AMDF can be expressed as

$$\hat{\tau} = \arg\min_{\tau} \frac{1}{N} \sum_{i=1}^{N} \left| x_1^i(t) - x_2^i(t-\tau) \right|. \tag{8}$$

3.2 FPGA implementation

The entire sound localization system (except for the microphone signal amplifiers) was implemented on a single Altera DE2-70 FPGA board. Software design was developed by using the Quartus II software tool. Firstly, on the FPGA board, the AD converter controller used the I²C protocol to control serial input and serial output data. The sound activity detection block was divided into three modules and implemented separately. At the time-delay estimation stage, the AMDF block used conventional basic operational logic elements, such as shift registers, subtraction, absolute value operands, and accumulation, to facilitate the entire design. All blocks implemented the pipeline technique to further accelerate computation. Finally, the output result is displayed on the DE2-70 board using a seven-segment display and LEDs. The system used a total of around 15,600 logic elements (around 188,000 logic gates).

3.3 SoC implementation

After the FPGA simulation and validation were complete, the sound localization system was ported to the chip level. In this system, after an input signal passed through the sound activity detection module, it was stored in the left and right SRAMs respectively. Next, the subtraction/absolute/accumulation (SAA) (Wang et al., 2011) module performed the major operations in the AMDF, including subtraction, taking absolute values, and accumulation. Hence, the AMDF block was able to estimate the time delay using the SAA module and convert it into a direction by accessing a predefined table in the ROM.

However, while running the AMDF, the system must perform the correlation analysis, $\Gamma = \sum_{i=1}^{N} \left| x_1^i - x_2^i \right|$, N times. The variable N is set herein to 64 for convenience of chip implementation. To reach a favorable trade-off between the chip area and performance, the system used a folding technique to realize the SAA architecture (Fig. 6 and (9)). A comparison with the unfolded SAA architecture revealed that the number of adders had been reduced from 127 to eight, and the number of units that performed the absolute value operation had been decreased from 64 to four. The length of the critical paths was effectively minimized, enhancing the clock rate.

$$\Gamma = \begin{cases} \sum_{j=1}^{4} \left| x_1^j(t) - x_2^j(t) \right| + \Gamma_{\text{accumulative}} & \text{when } t = 16j. \\ \sum_{j=1}^{4} \left| x_1^j(t) - x_2^j(t) \right| & \text{when } t = 16j+1,2,...,15. \end{cases} \tag{9}$$

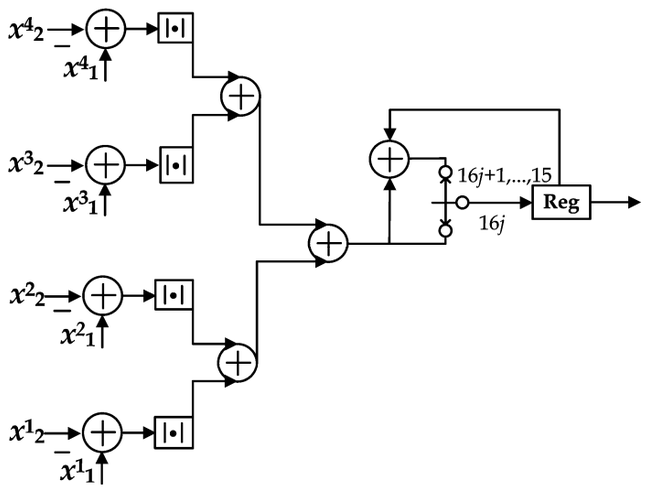


Fig. 6. Folded SAA architecture (Wang et al., 2011).

3.4 Experimental results

The sound localization system was tested with sources in different directions, ranging from 15° to 90° in steps of 15°, at five distances (1–5m). The experimental results indicated that the average accuracy was 80%–90%. The estimation error could be maintained in the range ±5°–±10°. With respect to chip performance, the number of logic gates was reduced to 32,616. Also, the core size and power consumption were minimized (see (Wang et al., 2008a; Wang et al., 2009; Wang et al., 2011) for details).

4. Embedded system design for speaker identification/verification

The field of speaker recognition has existed for five decades (Furui, 2004). Recently, speaker recognition systems have found many applications in the real world. It is highly flexible and convenient for a wide range of daily-life applications. Various approaches, involving neural networks (Clarkson et al., 2001), Gaussian mixture models (GMMs) (Burget et al., 2007), and

support vector machines (SVMs) (Cortes et al., 1995), have been adopted for recognizing speakers. Among them, SVM-based speaker recognition has recently attracted much attention.

Based on the idea of the working set, Platt et al. (1998) proposed the use of the sequential minimal optimization (SMO) algorithm, which is a widely used learning algorithm that involves decomposition, to solve the quadratic programming (QP) problem. Basically, the SMO algorithm performs the following two processes repeatedly: 1) selecting a fixed number of Lagrange multipliers, and 2) solving the QP problem of the multipliers until an optimal solution is found. Although the SMO algorithm makes SVM learning feasible when the number of training samples is very large, the number of required computational iterations still results in a heavy computational burden, which makes it unsuitable for use with stand-alone embedded devices.

The operation of the proposed system based on SMO involves a training phase and an identification phase. Since the SMO training algorithm has huge computational load, it is realized as a dedicated, very large-scale integration (VLSI) module, which is a hardware component. The rest processes of the system, such as speech preprocessing, speech feature extraction, and SVM-based voting, are implemented in software. The proposed system has 90% less training time than the embedded C-based ARM processor, and achieves an 89.9% accuracy with the 2010 speaker recognition database of the National Institute of Standards and Technology (NIST). The proposed system was tested and found to be fully functional on a Socle CDK prototype development board (www.socle-tech.com.tw) with an AMBA-based Xilinx FPGA board and an ARM926EJ processor.

4.1 Methodology

4.1.1 Support vector machine

Support vector classification (Cortes et al., 1995) is a computationally efficient means of finding hyperplanes in a high-dimensional feature space. Training an SVM is the equivalent to finding a hyperplane with the maximum margin.

The canonical representation of a decision hyperplane is (10),

$$y_i\left(w^T\phi\left(x_i\right)+b\right)\geq 1, \quad i=1,...,N \tag{10}$$

where w is the weights of training instances; b is a constant; y_i is the label of x_i. The optimization problem involves minimizing $\|w\|^2$. In imperfect separation, the optimal hyperplane is obtained by solving the following constrained optimization problem (11),

$$\begin{cases} \min\limits_{w,b,\xi}\dfrac{1}{2}w^Tw+C\left(\sum\limits_{i=1}^{N}\xi_i\right) \\ y_i(w\phi(x_i)+b)+\xi_i-1\geq 0, \quad \xi_i\geq 0, 1\leq i\leq N \end{cases} \tag{11}$$

where C is a real-valued cost parameter, and ξ_i is a penalty parameter (slack variable). If $\phi(x_i)=x_i$, the SVM finds a linear separating hyperplane with the maximal margin. An SVM

is called a nonlinear SVM when φ maps x_i into a higher-dimensional space. Equation (12) is the Lagrange function for imperfect separation.

$$\begin{cases} \arg\max\limits_{\alpha} \; L_D = \sum\limits_{i=1}^{N} \alpha_i - \frac{1}{2} \sum\limits_{i=1}^{N} \sum\limits_{j=1}^{N} y_i y_j x_i^T x_j \alpha_i \alpha_j \\ \sum\limits_{i=1}^{N} y_i \alpha_i = 0, \quad 0 \le \alpha_i \le C, 1 \le i \le N \end{cases} \tag{12}$$

Basically, (12) is a QP problem and can be solved using the SMO algorithm.

4.1.2 Sequential Minimal Optimization

The basic problem of the SMO algorithm is the need to find hyperplane parameters, w and b, by updating Lagrange parameter a. The SMO algorithm searches through the feasible region of the dual problem and maximizes the objective function by choosing two a terms and jointly optimizes them (with the values of the other a terms fixed) in each iteration. Then, the objective function can be written as (13).

$$\begin{aligned} L_D = {} & \alpha_1 + \alpha_2 + \text{Constant}_1 \\ & - \frac{1}{2} [y_1 y_1 x_1^T x_1 a_1^2 + y_2 y_2 x_2^T x_2 a_2^2 + 2 y_1 y_2 x_1^T x_2 a_1 a_2 \\ & + 2 (\sum_{i=3}^{N} \alpha_i y_i x_i^T)(y_1 x_1 \alpha_1 + y_2 x_2 \alpha_2) + \text{Constant}_2] \end{aligned} \tag{13}$$

Let $\frac{\partial L_D}{\partial \alpha_2} = 0$, yielding (14).

$$\begin{aligned} \alpha_2^{\text{new}} &= \alpha_2^{\text{old}} + \Delta \alpha_2 \\ &= \alpha_2^{\text{old}} + \frac{y_2 (E_2^{\text{old}} - E_1^{\text{old}})}{\eta} \end{aligned} \tag{14}$$

where E_1^{old} and E_2^{old} are prediction errors, and η is given by (15).

$$\begin{aligned} \eta &= 2K_{12} - K_{11} - K_{22} \\ &= 2x_1^T x_2 - x_1^T x_1 - x_2^T x_2 \\ &= -\left\| x_2 - x_1 \right\|^2 \end{aligned} \tag{15}$$

Let the minimum and maximum feasible values of a_2 be L and H, respectively. The unconstrained α_2^{new} must be checked to determine whether it is in the feasible range. Then, a clipping function, (16), is used to generate the new constrained $\alpha_2^{\text{new,clipped}}$.

$$\alpha_2^{\text{new,clipped}} = \begin{cases} H, & \text{if } H > \alpha_2^{\text{new}} \\ \alpha_2^{\text{new}}, & \text{if } L \le \alpha_2^{\text{new}} \le H \\ L, & \text{if } \alpha_2^{\text{new}} < L \end{cases} \tag{16}$$

Eventually, α_1^{new} can be obtained from (17).

$$\begin{aligned} \alpha_1^{\text{new}} &= \alpha_1^{\text{old}} + \Delta\alpha_1 \\ &= \alpha_1^{\text{old}} - y_1 y_2 \Delta\alpha_2 \end{aligned} \tag{17}$$

The terms Δa_1 and Δa_2 are used to update the hyperplane parameters w and b according to (18) and (19).

$$\Delta w = \Delta\alpha_1 y_1 x_1 + \Delta\alpha_2 y_2 x_2 \tag{18}$$

$$\begin{aligned} \Delta b &= \frac{1}{2}(b_1 + b_2) \\ &= \frac{1}{2}\left[\left(E_1 + y_1 \Delta\alpha_1 x_1^T x_1\right) + \left(E_2 + y_2 \Delta\alpha_2 x_2^T x_2\right)\right] \end{aligned} \tag{19}$$

4.2 Hardware implementation

The proposed system can perform both speaker training and identification. Based on the complexity analysis in Fig. 7, the SMO training, which takes 90.89% of the training time, is the computational bottleneck. Hence, the SMO is realized in hardware and the rest processes, including preprocessing, feature extraction and voting analysis, are implemented in software. As shown in Fig. 8, the proposed design comprises four blocks, which are the software-based extraction block (SEB), hardware-based training block (HTB), and software-based voting block (SVB). The SEB mainly performs speech preprocessing and speech feature extraction. The HTB executes the SMO algorithm, and the SVB is designed to find the target speaker based on a multiclass SVM. This design can be applied to a fast-trainable system in a stand-alone embedded environment (see (Kuan et al., 2010) for details).

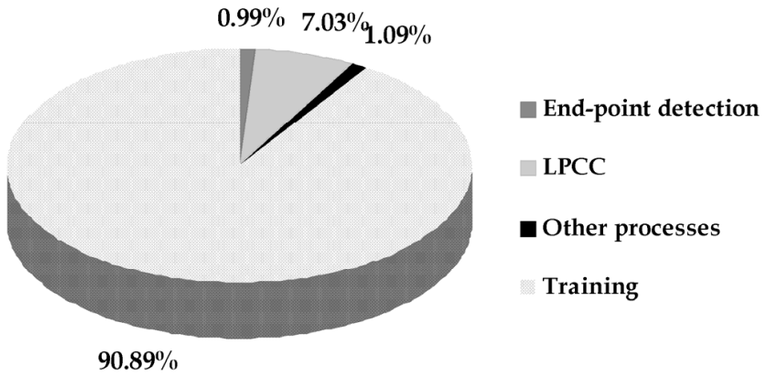


Fig. 7. Complexity analysis for speaker identification.

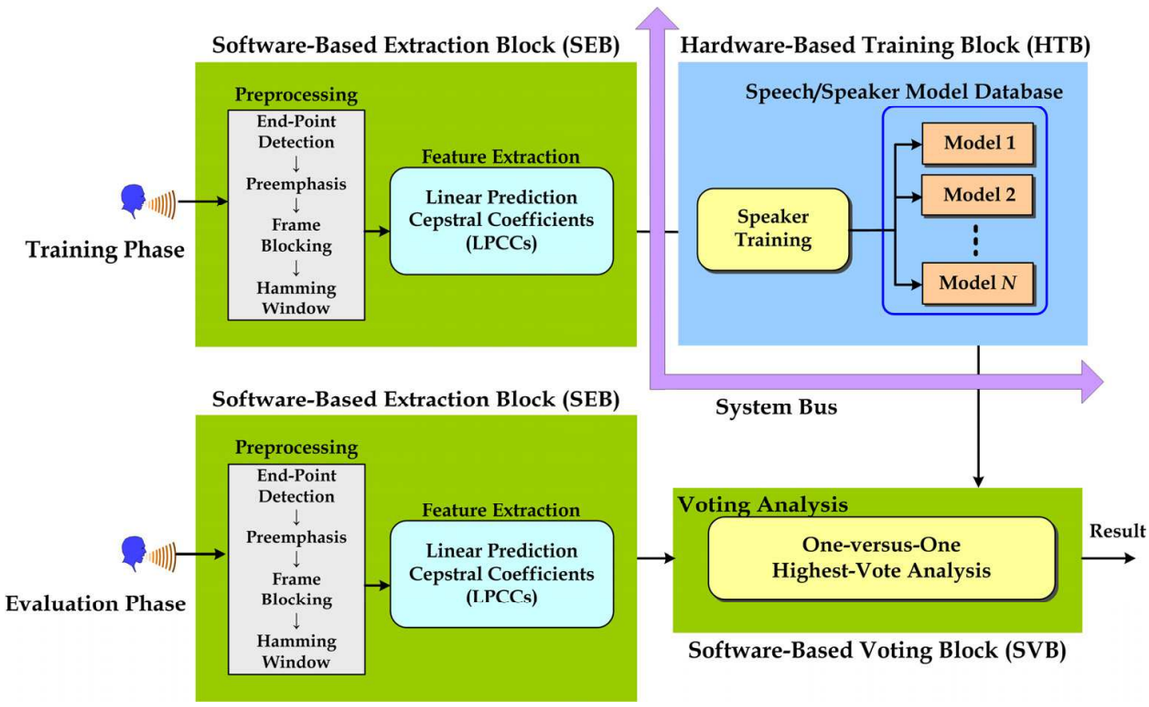


Fig. 8. Proposed hardware/software co-design system for speaker identification

4.3 Experimental results

The NIST 2010 speaker recognition evaluation (SRE10) speech corpus (by nine speakers) was adopted to evaluate the proposed hardware/software co-design framework. Six datasets, including nine speakers' files in SRE10, were used to evaluate a speaker identification system for an entrance security application. The training utterance of each speaker was 10s long. The duration of the testing utterances was 2–6s. The order of the linear predictive cepstral coefficients (LPCCs) was 18.

Figure 9 presents a time-cost comparison between the proposed hardware/software system and the embedded C code system (ARM-ported system). The proposed design had a 90% lower time-cost than the embedded C code one in the case of interest. Details of the evaluation can be found in (Wang et al., 2008b).

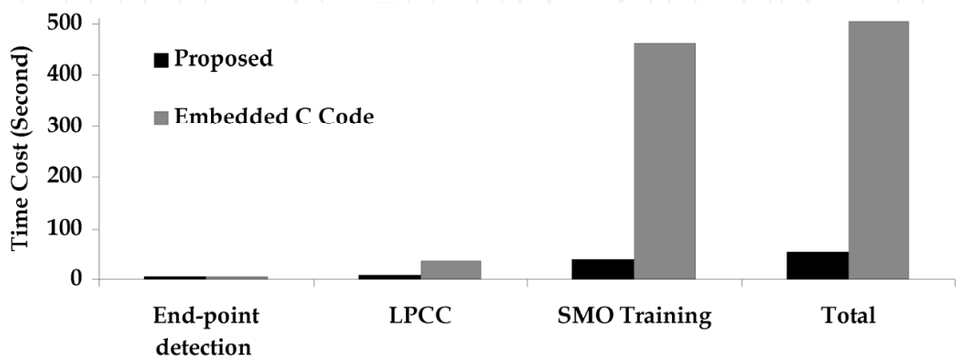


Fig. 9. Performance evaluation based on time cost.

5. Embedded system design for interactive retrieval of spoken documents

Owing to the increasingly widespread use of personal portable devices, an efficient method for retrieving spoken data with limited resources is required. This section proposes an efficient feature-based sentence-matching algorithm for speaker-dependent personal spoken sentence retrieval. Such a system can efficiently retrieve database sentences only partially matched to query sentence inputs. A whole matching plane-based accumulation (WMPB) scheme is then designed to determine the global similarity score. The proposed algorithms are based on the feature-level comparison and do not require acoustical and language models.

5.1 Methodology

5.1.1 Sentence matching for retrieving spoken sentences

Sentence matching is performed to determine the similarity between two sentences. Consider two spoken sentences *A* and *B*: Assume that $A = \{a_1 a_2 ... a_m\}$ is an *m*-word spoken sentence and $B = \{b_1 b_2 ... b_n\}$ is an *n*-word spoken sentence. The similarity between *A* and *B* can then be directly determined from the number of matched words (common words) in these two sentences. For example, if spoken sentences *A* and *B* are "I have a meeting in London tomorrow" and "Where is my meeting tomorrow?" respectively, then "meeting" and "tomorrow" are the matched words. Since only subsets of words in sentences are matched, sentence matching is a form of partial matching. This partial sentence-matching concept can be applied to spoken sentence retrieval.

Because this similarity is defined semantically, using a speech recognition system with acoustical and language models to transcribe spoken sentences into semantic texts is intuitive. To develop a language-independent retrieval system with a small required memory and favorable performance for a medium-sized sentence database, feature-level partial matching algorithms that do not use acoustic and language models are proposed herein.

5.1.2 Spoken sentence retrieval based on feature-level partial matching

This subsection presents a new partial matching system that is applied to the feature level. Figure 10 shows the proposed feature-level partial matching. First, the features of the spoken sentence are extracted frame by frame. The feature sequence is then segmented into equally sized matching units that are called feature pattern units (FPUs). Given a query sentence *Q* with *l* FPUs and a database sentence *D* with *k* FPUs, the sentences *Q* and *D* are denoted by $Q = \{q^F_{sub1} q^F_{sub2} ... q^F_{subl}\}$ and $D = \{d^F_{sub1} d^F_{sub2} ... d^F_{subk}\}$. These equally sized FPUs of the query and database sentences form a matching plane, shown in Fig. 11. Each matching block in the matching plane is associated with an FPU in the query sentence and the database sentence.

Let Ψ be the feature-level similarity function. The global similarity score for *Q* and *D* in the feature-level is calculated

$$\Psi\left[D^F, Q^F\right] = \Psi\left[d^F_{sub1} d^F_{sub2} \ldots d^F_{subk}, q^F_{sub1} q^F_{sub2} \ldots q^F_{subl}\right] = \left\{\sum_{i=1}^{l}\sum_{j=1}^{k} \Psi[d^F_{subj}, q^F_{subi}]\right\} \cdot M\left(D^F\right) \quad (20)$$

where $\Psi[d^F_{subj}, q^F_{subi}]$ is the local similarity score, which quantifies the similarity between FPUs q^F_{subi} and d^F_{subj}; $M(D^F) = 1/k$ represents the normalization factor for different database sentences.

Clearly, $\Psi[d^F_{subj}, q^F_{subi}]$ depends on the feature distances between every pair of FPUs, q^F_{subi} and d^F_{subj}. Although $\Psi[d^F_{subj}, q^F_{subi}]$ can also be implemented using $\Phi[d^S_{subj}, q^S_{subi}]$, however, a distance threshold is required (Itoh, 2001; Itoh & Tanaka 2002). Further, this threshold is difficult to define owing to variation in speech. Without a threshold comparison, an attempt is made herein to find a better similarity score function based on only the feature distances.

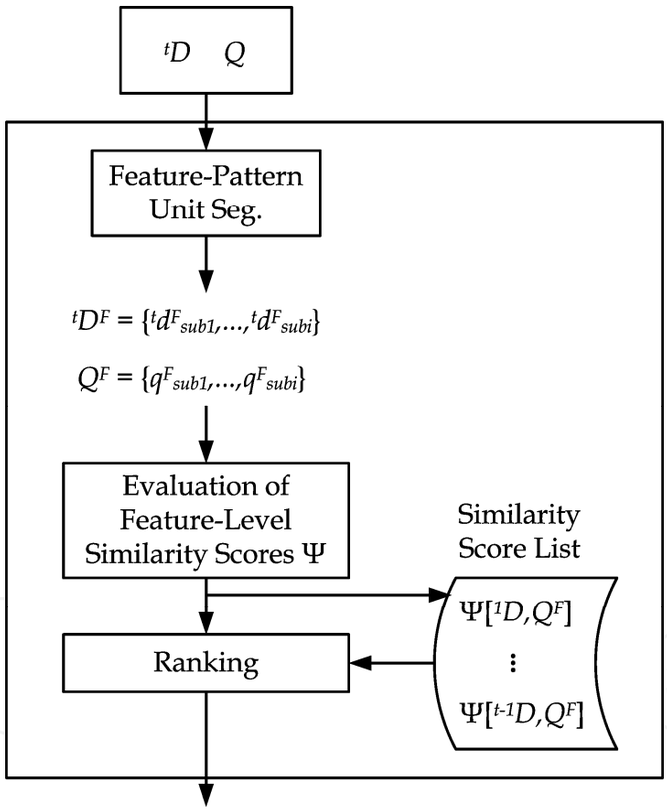


Fig. 10. Proposed feature-level partial matching algorithm.

To test which weighting function performs well, experiments on inverse exponential weighting (IEW(X) = $1/e^X$) and inverse distance weighting (IDW(X) = $1/X^p$, where p is an integer weighting power) techniques for summing local similarity scores were conducted (see the previous work (Lin & Wang, 2007) for details). Based on this experiment, the IDW function outperformed the IEW function; therefore, IDW was used to evaluate the similarity score. The global similarity score function Ψ is defined as,

$$\Psi\left(D^F, Q^F\right) \equiv \sum_{i=1}^{l}\sum_{j=1}^{k} \mathrm{IDW}\left[\mathrm{distance}\left(d_{subj}^F, q_{subi}^F\right)\right] \cdot M\left(D^F\right) \tag{21}$$

where q_{subi}^F denotes the i-th FPU of the query sentence, and d_{subj}^F represents the j-th FPU of the database sentence. The IDW method provides a measure of estimating uncertainty of variables. Moreover, this approach is sufficiently flexible to model the variables in a trend curve (Tomczak, 1998).

5.2 WMPB algorithm

Based on the above description, the proposed spoken sentence retrieval is summarized as follows.

Step 1. Sentence segmentation and feature extraction

Assume that the FPU size is n frames. The length of the overlapping between successive FPUs is $n/2$ frames (Ng & Victor, 2000). A spoken query sentence and a spoken sentence from the database are segmented based on the FPU size, with $n/2$ overlapping frames. The FPU overlap of $n/2$ frames is taken from another work (Itoh, 2001). Moreover, such a setting covers each frame in the query and database sentences; this scheme of redundancy is thought to be advantageous for partial matching. According to Fig. 11, this query sentence has l FPUs and the database sentence has k FPUs.

Step 2. Determination of matching plane

For a query sentence with l FPUs and a database sentence with k FPUs, a 2-D matching plane that contains $l \times k$ matching blocks is created. T matching planes are created if the database contains T sentences. Figure 11 illustrates the creation of the matching planes.

Step 3. Calculation of the similarity score of each matching block

For each matching block, dynamic programming is utilized to calculate the feature distance of the two FPUs. These feature distances are then used to determine local similarity scores using the IDW function.

Step 4. Accumulation of similarity scores

Over the whole matching plane, the similarity scores associated with all of the matching blocks are accumulated to yield a global similarity score.

Step 5. Iterative checking sentences from other databases

Repeat steps 1 to 4 for the other database sentences until all of their global similarity scores are obtained.

Step 6. Ranking of database sentences

Rank the database sentences in accordance with global similarity scores. Because the local similarity scores of all the matching blocks in the matching plane are accumulated to yield a global similarity score, the proposed spoken sentence retrieval method is called the whole matching plane-based (WMPB) algorithm.

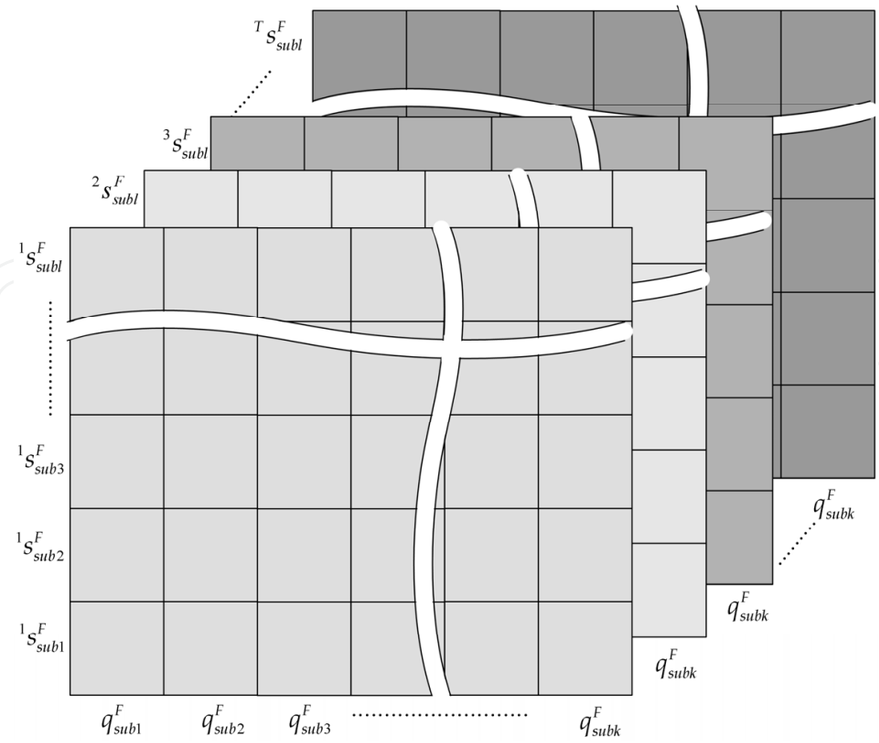


Fig. 11. Example of creation of matching plane.

5.3 Embedded system implementation

The proposed spoken sentence retrieval system was realized in a Pocket PC (HP iPAQ H5550) with a 128 MB RAM and 48 MB flash memory. The Pocket PC uses an Intel PXA255 processor (an XScale micro-architecture based on the ARM V5TE), which is a dedicated portable chip and suitable for handheld devices (www.intel.com). A 16-bit integrated audio codec (AC'97 2.0) was adopted for concurrent real-time speech input/output. The average memory size of one sentence was 142.3 kB with a sampling rate of 8 kHz. The Microsoft embedded complier based on Visual C++ 4.0 was used for the OS of the Pocket PC. Since the PXA255 processor does not support floating-point computation, a fixed-point conversion strategy was conducted to tackle the problem (see (Lin & Wang, 2007)). After the conversion method transformed the partial-matching program into a fixed-point format, the program was burned into the onboard flash memory. The system showed that the program occupied only 140 kB memory, which is appropriate for portable devices.

5.4 Experimental results

The experiments are divided into two phases - the parameter setting phase and the evaluation phase. In the parameter setting phase, experiments are conducted to find the best parameters of the IDW function and the FPU size for the proposed algorithm. Table 6 lists the characteristics of the experimental environment. Some experiments were to evaluate the retrieval performance of the proposed partial matching algorithm. Sentences were spoken naturally by one person without controlling the duration of the words or speaking at a

deliberately chosen rate. The query sentences partially matched their related database sentences. Here, matching keywords are defined as the terms that are common to queries and their related database sentences. Table 7 lists the overall statistics concerning the experimental database. The database sentences were ranked by their global similarity scores. The retrieval performance was assessed using the most commonly used measurement, which is non-interpolated mean average precision (mAP) (Baeza-Yates & Ribeiro-Neto, 1999; Lo et al., 2002). The mAP is defined as,

$$\text{mAP} = \frac{1}{L}\sum_{i=1}^{L}\left\{\frac{1}{M}\sum_{j=1}^{M_i}\left\{\frac{1}{N_j}\sum_{k=1}^{N_j}\text{precision}_{N_{Q_j}}(k)\right\}\right\} \tag{22}$$

where N_j denotes the total number of relevant sentences for query j; M_i represents the total number of queries in batch i; L is the total number of query batches, and $\text{precision}_{N_{Q_j}}(k)$ is the precision of Q_j when k sentences are retrieved. Finally, Table 8 summarizes the overall statistics for the entire experimental database.

Input	**Spoken query sentence**
Output	Ranking of spoken database sentences
Acoustical environment	In-door environment
Sampling rate	8 kHz
Quantization	16 bits
Frame size	256 samples (32 ms)
Frame overlapping size	64 samples (8 ms)
Speech feature	10-order LPCCs
DTW local path constraint	Type 1
FPU size	22 frames
IDW	$1/X^8$

Table 6. Characteristics of experimental environments.

Data set	**Data set A**			**Data set B**			**Data set C**		
Phase	Parameter setting phase			Evaluation phase					
Number of database sentence	50 Mandarin			50 Mandarin			50 Mandarin + 50 English		
Number of query sentence	15			15			30		
Percentage of common words among queries and their relevant database sentences	46.2			42.9			51.3		
Statistics type	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
Frame number of database sentence length	68	120	90.2	63	114	88	50	192	92
Frame number of query sentence length	65	89.1	73.8	62	85	71.6	66	208	76
Number of relevant database sentence per query	3	7	3.21	2	7	3.47	2	6	3.86

Table 7. Database statistics.

Platform	Data set	# database sentences	#query sentences	mAP	Response time for one query (sec.)
PC (Pentium 4 3.0GHz with 512Mb RAM)	B	50 Mandarin	15	0.887	<0.5
	C	50 Mandarin +50 English	30	0.763	<1.0
iPAQ H5550 PocketPC	B	50 Mandarin	15	0.799	<1.5
	C	50 Mandarin +50 English	30	0.675	<2.5

Table 8. Experimental results.

6. Conclusion

This chapter presented various speech processing approaches for use in embedded systems, involving speech feature extraction, sound localization, speaker identification/verification, and interactive retrieval of spoken documents. To facilitate implementation, related algorithms and methods of improving them are discussed with reference to FPGA and ARM-based architectures. Experiments were also conducted using testing datasets; the results showed that proper hardware design can improve the performance of the approaches, and the efficacy of the improved algorithms was subsequently demonstrated.

7. Acknowledgment

This work was supported in part by the National Science Council of the Republic of China under Grant No. 100-2218-E-006-033. The authors would like to thank Po-Yi Shih and Jr-Siang Peng for their dedication to this chapter. Furthermore, each section is written by the following authors: Bo-Wei Chen (Sections 1, 3, and 6), Po-Yi Shih (Section 2), Jr-Siang Peng (Section 4), and Po-Chun Lin (Section 5).

8. References

Baeza-Yates, R. & Ribeiro-Neto, B. (1999). *Modern Information Retrieval* (1st edition), Addison Wesley/ACM Press, ISBN 978-020-1398-29-8, New York, NY.

Burget, L.; Matejka, P.; Schwarz, P.; Glembek, O. & Cernocky, J.-H. (2007). Analysis of Feature Extraction and Channel Compensation in a GMM Speaker Recognition System. *IEEE Transactions on Speech, Audio and Language Processing*, Vol.15, No.7, (September 2007), pp. 1979-1985.

Chen, B.-W.; Wang, J.-F. & Wang, J.-C. (2010). Improving Direction of Arrival Estimation Based on the Directivity Pattern Analysis and Adaptive Cascaded Classifiers. *Journal of the Chinese Institute of Engineers*, Vol.33, No.5, (July 2010), pp. 751-760.

Clarkson, T.-G.; Christodoulou, C.-C.; Guan, Y.; Gorse, D.; Romano-Critchley, D.-A. & Taylor, J.-G. (2001). Speaker Identification For Security Systems Using Reinforcement-Trained pRAM Neural Network Architectures. *IEEE Transactions on Systems, Man, and Cybernetics (SMC)—Part C: Applications and Reviews*, Vol.31, No.1, (February 2001), pp. 65-76.

Cortes, C. & Vapnik, V. (1995). Support Vector Networks. *Machine Learning*, Vol.20, (1995), pp. 273-297.

Furui, S. (2004). Fifty Years of Progress in Speech and Speaker Recognition. *Acoustical Society of America Journal*, Vol.116, No.4, (2004), pp. 2497-2498.

Itoh, Y. (2001). A Matching Algorithm between Arbitrary Sections of Two Speech Data Sets for Speech Retrieval, *Proceedings of ICASSP 2001 IEEE International Conference on Acoustics,*

Speech, and Signal Processing, pp. 593-596, Salt Lake City, Utah, USA, May 07-11, 2001.

Itoh, Y. & Tanaka, K. (2002). Speech Labeling and the Most Frequent Phrase Extraction Using Same Section in a Presentation Speech, *Proceedings of ICASSP 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. I737-I740, Orlando, Florida, USA, May 13-17, 2002.

Kuan, T.-W.; Wang, J.-F.; Wang, J.-C.; Lin, P.-C. & Gu, G.-H. (2010). VLSI Design of an SVM Learning Core on Sequential Minimal Optimization Algorithm. *IEEE Transactions on Very Large Scale Integration Systems*, Vol.PP, No.99, (December 2010), pp. 1-11.

Lin, P.-C. & Wang, J.-F. (2007). Speaker Change Detection and Spoken Sentence Retrieval for Automatic Minute Taking. Doctoral dissertation, National Cheng Kung University, Taiwan.

Lo, W. K., Meng, H. & Ching, P. C. (2002). Multi-Scale and Multi-Model Integration for Improved Performance in Chinese Spoken Document Retrieval, *Proceedings of ICSLP2002 7th International Conference on Spoken Language Processing,* pp. 1513-1516, Denver, Colorado, USA, September 16-20, 2002.

Ng, K., & Victor W. Z. (2000). Subword-Based Approaches for Spoken Document Retrieval. *Speech Communication*, Vol.32, No.3, (October 2000), pp. 157-186.

Platt, J.-C. (1998). Fast Training of Support Vector Machines Using Sequential Minimal Optimization, In: *Advances in Kernel Methods: Support Vector Machines*, Schölkopf, B.; Burges, C. & Smola, A., (Eds.), MIT Press, ISBN 978-026-2194-16-7, Cambridge, MA.

Tomczak, M. (1998). Spatial Interpolation and Its Uncertainty Using Automated Anisotropic Inverse Distance Weighting (IDW) Cross-Validation/Jackknife Approach. *Journal of Geographic Information and Decision Analysis*, Vol.2, No.2, (May 1998), pp. 18-30.

Vergin, R.; O'Shaughnessy, D. & Gupta, V. (1996). Compensated Mel Frequency Cepstrum Coefficients, *Proceedings of ICASSP 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 323–326, Atlanta, Georgia, USA, May 07-10, 1996.

Wang, J.-C.; Wang, J.-F. & Weng, Y.-S. (1998). The Chip Design of Mel Frequency Cepstrum Coefficient for HMM Speech Recognition, *Proceedings of the 9th VLSI Design/CAD Symposium*, pp. 439-442, Nangtou, Taiwan, August 20-22, 1998.

Wang, J.-C.; Wang, J.-F. & Weng, Y.-S. (2003). Chip Design of MFCC Extraction for Speech Recognition. *VLSI Journal on Integration*, Vol.32 No.1-3, (November 2002), pp. 111-131.

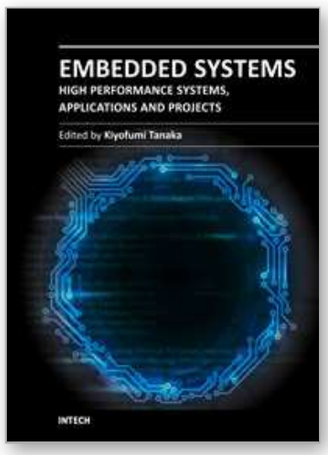
Wang, J.-F.; Chou, C.-H.; Huang, Y.-J.; Lin, P.-C. & Chen, B.-W. (2011). An Improvement of Chip Design for Auditory Source Localization Awareness, *Proceedings of 3rd International Conference on Awareness Science and Technology*, pp. 362-365, Dalian, China, September 27-30, 2011.

Wang, J.-F.; Jiang, Y.-C. & Sun, Z.-W. (2009). FPGA Implementation of a Novel Far-Field Sound Localization System, *Proceedings of TENCON 2000 IEEE Region 10 Conference*, pp. 1-4, Singapore, November 23-26, 2009.

Wang, J.-F.; Wang, J.-C. & Weng, Y.-S. (2000). Chip Design of Mel Frequency Cepstral Coefficients for Speech Recognition, *Proceedings of ICASSP 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 3658-3661, Istanbul, Turkey, June 05-09, 2000.

Wang, J.-F.; Wang, J.-C.; Chen, B.-W. & Sun, Z.-W. (2008a). A Long-Distance Time Domain Sound Localization, *Proceedings of UIC 2008 5th International Conference on Ubiquitous Intelligence and Computing*, pp. 616–625, Oslo, Norway, June 23-25, 2008.

Wang, J.-F.; Wang, J.-C.; Mo, M.-H.; Tu, C.-I; & Lin, S.-C. (2008b). The Design of a Speech Interactivity Embedded Module and Its Applications for Mobile Consumer Devices. *IEEE Transactions on Consumer Electronics*, Vol.54, No.2, (May 2008), pp. 870-876.



Embedded Systems - High Performance Systems, Applications and Projects

Edited by Dr. Kiyofumi Tanaka

ISBN 978-953-51-0350-9
Hard cover, 278 pages
Publisher InTech
Published online 16, March, 2012
Published in print edition March, 2012

Nowadays, embedded systems - computer systems that are embedded in various kinds of devices and play an important role of specific control functions, have permeated various scenes of industry. Therefore, we can hardly discuss our life or society from now onwards without referring to embedded systems. For wide-ranging embedded systems to continue their growth, a number of high-quality fundamental and applied researches are indispensable. This book contains 13 excellent chapters and addresses a wide spectrum of research topics of embedded systems, including parallel computing, communication architecture, application-specific systems, and embedded systems projects. Embedded systems can be made only after fusing miscellaneous technologies together. Various technologies condensed in this book as well as in the complementary book "Embedded Systems - Theory and Design Methodology", will be helpful to researchers and engineers around the world.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen