TMS320C32 DSP를 이용한 실시간 화자종속 음성인식 하드웨어 모듈(VR32) 구현

Real-Time Implementation of Speaker Dependent Speech Recognition Hardware Module Using the TMS320C32 DSP: VR32

정 익 주*, 정 훈*
(Ik Joo Chung*, Hoon Chung*)

요 약

본 연구에서는 Texas Instruments 사의 저가형 부동소수점 디지털 신호 처리기(Digital Signal Processor, DSP)인 TMS320C32를 이용하여 실시간 화자종속 음성인식 하드웨어 모듈(VR32)을 개발하였다.

하드웨어 모듈의 구성은 40MHz의 TMS320C32 DSP, 14bit 코덱인 TLC32044(또는 8bit μ-law PCM 코덱), EPROM과 SRAM 등의 메모리와 호스트 인터페이스를 위한 로직 회로로 이루어져있다. 뿐만 아니라 이 하드웨어 모듈을 PC 상에서 평가해보기 위한 PC 인터페이스용 보드 및 소프트웨어도 개발하였다. 음성인식 알고리즘의 구성은 에너지와 ZCR을 기반으로 한 끝점검출(Endpoint Detection) 및 10차 가중 LPC 켑스트럼(Weighted LPC Cepstrum) 분석이 실시간으로 이루어지며 이후 Dynamic Time Warping(DTW)를 통하여 최고 유사 단어를 결정하고 다시 검증과정을 거쳐 최종 인식을 수행한다. 끝점검출의 경우 적응 문턱값(Adaptive threshold)을 이용하여 잡음에 강인한 끝점검출이 가능하며 DTW 알고리즘의 경우 C 및 어셈블리를 이용한 최적화를 통하여 계산속도를 대폭 개선하였다. 현재 인식률은 일반 사무실 환경에서 통상 단축다이얼 용도로 사용할 수 있는 30 단어에 대하여 95% 이상으로 매우 높은 편이며, 특히 배경음악이나 자동차 소음과 같은 잡음환경에서도 잘 동작한다.

ABSTRACT

We have developed the speech recognition hardware module(VR32) based on speaker dependent speech recognition technique. The hardware module includes a 40MHz TMS320C32 DSP, a 14bit linear codec TLC32044 or an 8bit μ-law PCM combo codec, memory devices, and logic circuits for host interfacing. In addition, the evaluation board and software working on a PC have been developed for easy evaluation. The algorithm for recognition consists of a real-time endpoint detection based on energy parameters and zero-crossing rate parameters and the feature extraction using the 10^{th} order weighted LPC cepstrum analysis and finally pattern matching using the dynamic time warping(DTW) followed by verification. The algorithm for endpoint detection has environmental robustness by adopting adaptive thresholds and the computational burden of the DTW is reduced by code optimization using C and assembly languages. Currently, the recognition accuracy is more than 95% even in noisy environment.

I. 서 론

최근에 음성인식 기술의 실질적인 활용 및 상용화에 큰 관심이 고조되면서 음성인식 알고리즘을 실시간으로 구현하는 여러 가지 방법에 대한 연구 및 개발이 활발히 이루어지고 있다. 실시간 음성인식기를 구현하는 플랫폼으로는 우선 PC나 Workstation의 호스트 CPU를 이용하는 환경과 스탠드얼로운(Stand Alone) 응용을 위한 프로그래머블 DSP나 ASIC화된 음성인식 칩을 이용하는 두 가지로 분류된다. PC나 Workstation의 경우 많은 주메모리와 보조 기억장치 등, 이용할 수 있는 자원이 충분하므로 대용량 음성인식 시스템을 구현하기에 용이하다. 한편 최근에 주목을 받고 있는 음성인식 전화기 또는 음성인식을 이용한 가전기기 등의 조작과 같은 응용을 위해서는 저가의 소형 음성인식 하드웨어가 필요하게 된다. 이 경우 사용할 수 있는 자원의 제한으로 인하여 인식기의 성능 및 인식어휘의 수가 다소 영향을 받게 된다. 스탠

*강원대학교 전자공학과
접수일자 : 1998년 2월 3일

드얼로운 인식기의 경우 인식어휘의 수는 통상 수십 단어로 제한되며 인식 방식은 응용에 따라 화자독립 또는 화자종속이 모두 가능할 것이다. 저가의 프로그래머블 DSP 또는 전용 음성인식 칩은 이런 응용제품에 적합하다고 할 수 있다. 그러나 음성인식 응용의 성격상 융통성이 결여 되는 전용 음성인식 칩의 경우 그 응용이 극히 제한을 받게 된다. 따라서 DSP를 이용한 저가의 음성인식 모듈의 개발은 성능과 활용의 융통성 면에 있어서 매우 유리하다고 할 수 있다.

본 논문은 이러한 배경을 바탕으로 현재 발표된 부동소수점 DSP 가운데 가장 저가인 Texas Instuments사의 TMS320C32(40MHz)를 이용한 화자종속 음성인식 모듈(VR32)의 개발에 관한 것이다. VR32의 개발시 염두에 둔 응용으로는 음성인식 전화기와 같이 화자종속 기술이 매우 유용하게 쓰이면서도 저렴한 재료비를 요구하는 응용제품을 목표로 하였다. 한편 VR32 하드웨어의 구성은 범용성과 확장성이 있으므로 향후 소프트웨어의 변경만으로 화자독립 응용에도 활용이 가능하다.

논문의 구성은 2장에서는 VR32의 하드웨어 구성, 3장에서는 인식 알고리즘, 4장에서는 DSP 구현, 그리고 5장에 결론으로 이루어져 있다.

Ⅱ. VR32의 하드웨어 구성 및 특징

VR32의 하드웨어 구성에 대한 블록도가 그림 1에 나와 있다. 초기에 사용한 메모리의 용량은 20ns의 32K word (128Kbyte)의 SRAM을 사용하였으며 이 메모리에는 인식을 위한 데이타와 인식 종료 시에 인식 결과를 확인시켜주기 위해 필요한 음성합성용 파라메타가 저장된다. 또한 TMS320C32의 경우 별도의 스트로브 신호를 이용하여 추가의 메모리 뱅크를 설치할 수 있으므로 70ns의 저속 SRAM 64K word를 옵션으로 설치 할 수 있다. 이 메모리는 인식과는 관계가 없으며 다만 확장성을 위하여 추가의 메모리를 설치할 수 있도록 한 것이다. 예를 들어

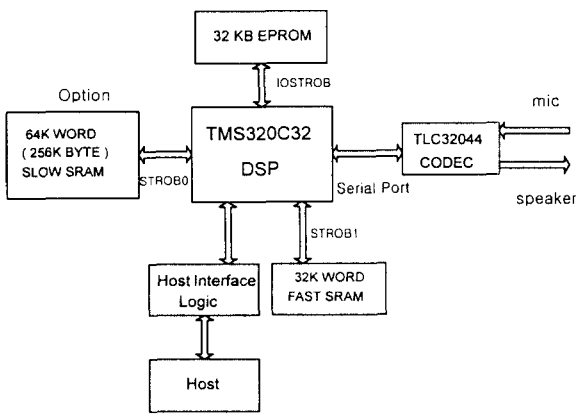

그림 1. VR32의 하드웨어 구성도

이 메모리는 음성압축을 이용한 메시지 저장등에 이용될 수 있다. 음성 입출력을 위한 코덱으로는 14bit 리니어 코덱인 TLC32044와 보다 저렴한 가격을 위한 일반 8bit μ-law PCM 콤보 코덱을 사용한 두 가지 종류를 개발하였다. 이 코덱들은 TMS320C32의 시리얼포트에 연결된다.

한편 음성인식 프로그램은 32Kbyte의 EPROM에 저장되어 있으며 추가의 프로그램을 위하여 64Kbyte의 EPROM도 사용할 수 있다. VR32의 특징은 다음과 같다.

①32K word의 메모리 사용 시 30 단어까지 인식.
②30단어 인식 시 인식에 소요되는 시간은 0.5초 미만
③인식이 완료된 후 인식확인을 위한 음성출력
④호스트 시스템과의 편리한 인터페이스 제공

한편 인식성능의 경우 현재 여러 가지 환경에서 테스트한 결과 잡음 환경하에서도 95% 이상의 인식률을 보였다.

Ⅲ. VR32의 인식 알고리즘

음성인식 알고리즘으로는 화자종속 고립단어 인식에 좋은 성능을 보이는 Dynamic Time Warping(DTW)을 사용하였다. 특히 이미 PC 환경에서 개발된 화자종속 고립단어 인식 소프트웨어인 VoiceAccess에서 사용되어 그 안정성이 확인된 끝점검출 및 분석, 그리고 패턴정합 관련 모듈들이 DSP 환경에 맞게 변형되어 사용되었다[6]. 다만 PC에서 구현할 때에는 11.025KHz의 표본화율을 사용하였는데 VR32에서는 8KHz의 표본화율을 이용하였다. 인식이 이루어지는 전체적인 흐름도가 그림 2에 나와 있다.

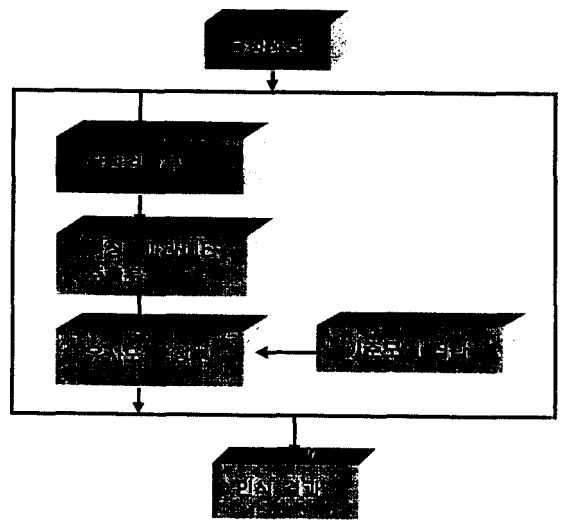

그림 2. 음성인식 시스템의 구성

우선 끝점검출과 동시에 음성 분석이 이루어지며, 분석된 결과는 인식하고자 하는 단어들의 기준패턴들과의

유사도 측정에 사용되고, 최소거리를 주는 단어를 찾은 후 검증과정을 거쳐 최종인식을 수행한다.

3.1 잡음에 강인한 실시간 끝점검출

실시간 음성 인식기를 구현함에 있어 실시간과 관련하여 가장 중요한 과정이 끝점 검출일 것이다. 왜냐하면 실질적으로 실시간으로 끝점을 검출하여 음성구간에 해당하는 음성파형을 저장할 수 있다면 그 이후로는 모든 것을 비실시간으로 처리할 수 있기 때문이다. 그러나 끝점 검출만은 반듯이 실시간으로 행하여져야 한다. 잡음환경하에서 고립단어 음성인식 시스템의 경우 오인식의 상당부분이 부정확한 음성의 검출에서 기인한다. 최근 고립단어 인식시스템의 평가에서 인식에러의 상당 부분이 부정확한 음성검출에 의해 발생한다고 밝히고 있다. 특히 비교패턴 간의 끝점 정보를 이용하여 시간 보상을 하는 DTW 기반의 음성인식 시스템의 경우 Hidden Markov Model(HMM) 기반의 인식 시스템보다 음성검출 부분이 전체 인식률 및 인식속도에 미치는 영향이 크다. VR32의 경우 DTW를 인식 알고리즘으로 사용하므로 보다 정교한 끝점검출 알고리즘을 사용하였다. 일반적으로 음성의 검출은 잡음이 있는 환경에서 수행되므로 정확한 음성검출을 위해서는 주변 잡음환경을 잘 모델링하여야 하며, 효과적인 음성검출 파라메타와 결정법칙을 가지고 있어야 한다. 그리고 시간에 따라 변화하는 주변환경에 적응할 수 있도록 설계되어야 한다[3][4]. 또한 실용적인 음성인식 시스템을 위해서는 실시간 음성검출에 적절한 계산량을 요구하여야 한다. 이미 잘 알려진 Rabiner & Sambur의 끝점검출 알고리즘을 포함한 대다수의 끝점검출 알고리즘은 오프라인 처리에 적합하며, 어느 구간 내에 음성이 포함되어있다는 가정하에서 음성의 끝점을 검출해낸다[5]. 경우에 따라서는 두 번 이상의 구간 검색을 하여 음성을 검출하므로 실시간 끝점검출에 적합하지 않은 경우도 있다. VR32에서는 위에서 제기한 잡음적응과 실시간 처리문제에 대해 신뢰성 있는 음성 검출결과를 보여주는 알고리즘을 사용하였다. 실시간 끝점검출을 위하여 사용한 파라메타로는 변형된 영교차율(zero crossing rate, ZCR)과 에너지를 이용하였다. 실시간으로 이루어져야하므로 모든 것이 100 샘플단위의 프레임 단위로 적용되어진다. 단, 음성 프레임인지 아닌지를 결정하는 단위가 100 샘플 단위이지만 이를 결정하는 파라메타들은 보다 나은 통계적 추정을 위하여 그림 3과 같이 200 샘플 단위로 파라메타를 추정하며 이들 파라메타를 통하여 음성 프레임으로 결정이 되면 그 200 샘플 중 가운데 100 샘플이 음성 프레임으로 저장이 된다.

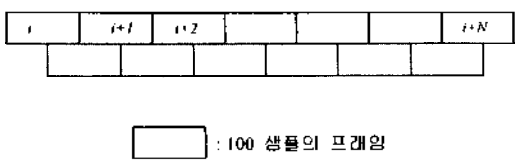

그림 3. 실시간 끝점검출 시 음성 프레임의 구조

음성프레임을 결정하기 위한 파라메타들의 정의가 식(1)부터 식(5)에 나와있다.

$$\begin{aligned} VoiceStartEnergy &= 10\times Energy + OffsetEnergy \\ VoiceEndEnergy &= 0.7\times VoiceStartEnergy \\ VowelEnergy &= 4.0\times VoiceStartEnergy \\ preVowelEnergy &= 2.0\times VoiceStartEnergy \\ UnvoicedZCR &= 5\times ZCR + OffsetZCR \end{aligned} \tag{1}$$

여기서 $OffsetEnergy$와 $OffsetZCR$은 각 시스템 마다 경험적 또는 실험적으로 결정되는 값이다. 위의 수식에서 $Energy$와 ZCR은 각각 다음과 같은 프레임 에너지와 프레임 영교차율을 기반으로 구해진다.

$$M = \sum_{k=0}^{99} (x_{i,k})^2$$

$$last\_energy_0 = 0$$

$$energy_i = \sum_{k=0}^{99} (x_{i,k} - M)^2 + last\_energy_{i-1}$$

$$last\ \ energy_i = \sum_{k=0}^{99} (x_{i,k} - M)^2 \tag{2}$$

$$Energy = 0.95\times Energy + 0.05\times energy_i \tag{3}$$

$$zcr_i = \sum_{k=0}^{99} count(x_{i,k}, x_{i,k+1}) + last\_zcr_{i-1}$$

$$last\_zcr_i = \sum_{k=0}^{99} count(x_{i,k}, x_{i,k+1}) \tag{4}$$

여기서 $count(x_{i,k}, x_{i,k+1})$는

$count(x_{i,k}, x_{i,k+1}) = 1$, if

$x_{i,k} - M < 1.5\times sqrt(Energy) < x_{i,k+1} - M$ or

$x_{i,k+1} - M < 1.5\times sqrt(Energy) < x_{i,k} - M$ or

$x_{i,k} - M < -1.5\times sqrt(Energy) < x_{i,k+1} - M$ or

$x_{i,k+1} - M < -1.5\times sqrt(Energy) < x_{i,k} - M$

otherwise, $count(x_{i,k}, x_{i,k+1}) = 0$ (5)

위의 수식들에서 x는 음성샘플이고 i는 프레임 인덱스이며, k는 한 프레임 안에서의 샘플 인덱스이다. 우선 초기 5 프레임이 연속해서 음성 프레임이라고 판정이 되면 일단 음성이 시작되었다고 가정한다. 이 경우 에너지 값이 $VoiceStartEnergy$보다 크거나 영교차율이 $UnvoicedZCR$보다 크면 음성프레임으로 판정한다. 또한 예외적으로 3 프레임이 연속해서 $preVowelEnergy$보다 크고 이후 7 프레임 내에서 다시 음성이 시작하면 음성이 시작한 것으로 가정한다. 이는 예를 들어 "끝"이나 "집"과 같이 모음을 포함하고 있으나 폐쇄음으로 끝나 순간 높은 에너지

값을 가지지만 지속시간이 짧아 이를 검출하지 못하는 경우가 발생하기 때문이다. 일단 음성이 시작 되었으면 이후 들어오는 프레임은 모두 음성으로 가정하고 동시에 끝점에 해당하는 프레임을 찾기 시작한다. 에너지가 *VoiceEndEnergy*보다 작고 그리고 영교차율이 *UnvoicedZCR*보다 작으면 끝점이 검출된 것으로 간주하고 그 이후 20 프레임을 일단 분석하여 버퍼에 저장함과 동시에 이 20 프레임 중 5 프레임이 음성으로 판정되지 않으면 음성검출을 완료한다. 만약 음성이 재개되면 이들을 계속해서 분석 저장하고 끝점검출을 반복한다. 이외에도 별도로 시작점과 끝점부근에서 주로 발생하는 저주파잡음인 숨결잡음 제거루틴이 추가된다. 끝점검출이 끝나면 사후처리를 거친다. 검출된 총 프레임 수가 10 프레임 이하거나 또는 전체 프레임 중 *VowelEnergy*보다 큰 프레임수가 4 이하면 이를 잡음으로 간주하고 다시 음성검출에 들어간다. 한편 LPC 계열의 합성을 위하여 실시간 피치 검출이 부가적으로 이루어지므로 이를 이용하여 안정된 피치의 유무 정보를 이용하여 *VowelEnergy*보다 큰 프레임 수가 4 이상이더라도 안정된 피치 구간이 없을 경우 이를 잡음으로 간주한다.

주변 잡음의 변화에 적응하기 위하여 음성구간 검출과정에서 그 프레임이 음성이 아니라고 판정이 되면 식(3)과 같이 관련 에너지 파라메타를 적응시킨다. 이를 통하여 여러 에너지 문턱값과 영교차율 문턱값이 변화된 주변잡음에 적응하도록한다. 그림 4는 위에서 설명한 알고리즘을 이용하여 검출된 "스페이스바"의 음성파형이다.

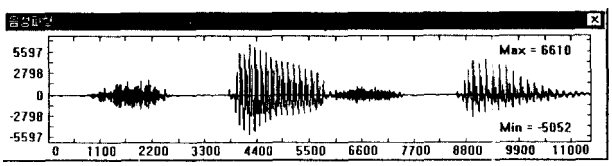

그림 4. 끝점검출된 "스페이스바"

3.2 실시간 음성분석 및 피치 추출

실시간 끝점검출이 프레임 단위로 이루어지고 영교차율과 에너지 만을 이용하므로 TMS320C32의 남는 계산능력으로 동시에 음성분석 및 피치 추출이 실시간으로 이루어진다. 음성분석은 200 샘플을 분석 구간으로 하고 100 샘플 마다 분석이 이루어진다. Preemphasis, Hamming window, 10차 가중 LPC 캡스트럼의 순으로 이루어진다. 이와 관련된 수학적인 수식이 식 (6)에서 식 (9)에 정리되어 있다.

$$\text{Preemphasis}: \hat{x}(n) = x(n) - 0.95x(n-1) \qquad (6)$$

$$\text{Hamming window}: \hat{x}(n) = \hat{x}(n)w(n), \quad 0 \le n \le N$$

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \qquad N = 200 \qquad (7)$$

다음으로 LPC 필터계수 a_m는 Durbin 알고리즘을 이용하여 구한다[2]. LPC 필터계수가 구해지면 이를 다음과 같이 캡스트럼 계수로 변환한다.

$$c_m = a_m + \sum_{k=1}^{m-1}\left(\frac{k}{m}\right)c_k a_{m-k}, \qquad 1 \le m \le 10 \qquad (8)$$

한편, 이렇게 구해진 캡스트럼 계수를 그대로 사용하지 않고 캡스트럼 계수의 민감도를 완화시키기 위해 다음과 같이 가중함수를 적용한 가중 캡스트럼 계수를 사용한다[7].

$$\hat{c}_m = \hat{w}_m c_m$$

$$\hat{w}_m = \left[1 + 5\sin\left(\frac{\pi m}{10}\right)\right], \qquad 1 \le m \le 10 \qquad (9)$$

결과적으로 한 프레임으로 부터 10개의 가중 캡스트럼 계수($\hat{c}_m$, $1 \le m \le 10$)의 특징 파라메터를 추출해내게 된다.

한편 인식 종료 시 인식확인을 위해 음성합성을 하게 되는데 이때에는 인식을 위하여 저장된 가중 LPC 캡스트럼(WLPCC) 계수와 잔여 에너지(Residual energy)를 이용하여 합성을 하게 된다. 합성 시에는 우선 WLPCC를 역가중화한 후 LPC 계수로 바꾸어 사용했다. 이때 피치와 유무성음에 대한 정보가 필요하므로 사용자가 단어를 훈련시킬 때 피치 및 유무성음을 분석하여 이 정보를 별도로 저장해 놓음으로서 후에 이를 이용하여 LPC 합성을 하게 된다. 한편 TMS320C32는 부동소수점 연산기이므로 분석된 WLPCC의 계수들은 각각 4byte의 부동소수점으로 저장되어질 수 있다. 그러나 VR32에서는 메모리 절약을 위하여 이를 2byte 즉 16 bit로 양자화하여 저장하였다. 물론 분석계수들을 보다 적은 bit로 양자화하는 방법이 음성부호화에서 많이 제안되었으나 인식의 성능을 고려하여 본격적인 양자화는 행하지 않았다. 사실 16bit는 분석계수를 저장하기에 충분한 해상도로서 실상은 32bit의 부동소수점으로 저장하는 것 자체가 다소 사치스러운 면이 있다고 할 수 있다. 실제 실험에 있어서도 16bit로 양자화하여 저장하였을 경우 32bit의 부동소수점으로 저장된 것과 비교하였을 때 인식률 및 합성음의 차이를 거의 느낄 수 없었다. 또한 합성음의 경우 기존의 LPC 합성 보다는 훨씬 좋은데 이는 정식의 LPC 부호화와는 달리 계수의 양자화가 거의 되지 않았다고 볼 수 있으며 또한 합성의 경우 인식이 종료 된 후 그 결과를 합성하여줌으로 합성을 위한 충분한 계산능력이 확보되므로 최근에 음성부호화기에서 많이 사용되는 포스트 필터링을 하였다. 하지만 이 음성출력의 목적이 인식된 결과를 확인시켜주는 것이므로 이러한 목적으로는 충분한 합성음질이 확보되었다고 가정하고 더 이상의 알고리즘은 추가하지 않았다. 한편 피치분석은 당연히 개루프 방식으로 이루어졌으며 자기상관계수를 이용하여 피치를 추출하였다. 후에 설명하겠지만, 실제 DSP에서의 구현 시

상당한 최적화가 이루어졌으므로 실시간 처리를 위한 충분한 계산능력을 확보할 수 있었다. 따라서 피치 추출을 위하여 AMDF 및 클리핑과 같은 계산량을 감소시키는 알고리즘을 사용하지 않았다[1]. 따라서 피치의 추출은 상당히 안정적이며 정확하다. 또한 합성 시, 혹시나 발생할 수 있는 피치 더블링과 같은 오류를 보안하기 위해 검출된 피치 컨투어를 5 포인트의 미디안 필터링을 하여 합성에 사용하였다. 한편 훈련은 2번 발음하고 난 후 이 두 개의 분석 패턴을 클러스터링하여 하나의 기준패턴으로 만든다. 또한 이 과정에서 동일 어휘를 두 번 발음하므로 그 어휘의 서로 다른 발음으로부터 유사도를 구하여 개별적으로 기억해 두었다가 모든 어휘의 훈련이 끝나면 일정한 통계과정을 거친 후 나중에 실제 인식 시, DTW가 끝난 후 최소거리로 얻어진 인식 결과가 등록된 어휘인지를 검증하는데 이용된다. 한편 이럴 경우 합성된 음질이 다소 무디지는 듯한 느낌을 받게 되는데 이는 특정 발음의 분석계수를 사용하지 않고 두번 발음된 명령어를 클러스터링하여 만들어진 분석계수를 사용하기 때문이다. 그러나 인식률이 우선이므로 이 방법을 채택하였다.

3.3 DTW를 이용한 패턴 매칭

고립단어 인식에 적용될 수 있는 음성인식 방법으로는 DTW와 HMM이 대표적이라고 할 수 있다. 현재 여러 논문 및 실제 테스트에서 DTW가 화자종속 고립단어 인식에서 HMM보다는 우수한 것으로 알려져 있다. 특히 음성의 발성시간을 보상하는 능력은 단순한 모델을 가지는 HMM에 비하여 탁월하다고 할 수 있다. 그러나 DTW의 단점은 위에서 언급한 바와 같이 끝점검출에 대단히 민감하여 오검출 시에 인식률에 큰 영향을 미친다는 점과 또 한가지는 계산량이 많다는 점이다[8][9]. 계산과 관련해서는 다음과 같은 Dynamic Programming 방법을 사용한다.

$$g(a_i, b_j) = d(a_i, b_j) + \min\{g(a_{i-1}, b_j), g(a_{i-1}, b_{j-1}), g(a_{i-1}, b_{j-2})\} \quad (10)$$

여기서 $g(a_i, b_j)$는 국부적 최적 누적거리이고 $d(a_i, b_j)$는 국부 거리이며 a_i, b_j는 특정 프레임의 동일차원 특징벡터이다.

VR32에서는 이 계산과 관련된 부분을 알고리즘 최적화와 코드 최적화 두 가지 측면에서 해결하였다. 우선 기본적으로 DTW 탐색을 위한 전역제한은 기준패턴 길이의 1/2~2배 까지로 하였으며 국부제한은 Itakura가 제안한 방식을 채택하였으며 테스트 패턴을 x축에 위치시켰다[10]. 또한 테스트패턴을 기준패턴들과 비교를 할 때마다 최소거리를 갱신하여 기억하고 있다가 특정 기준패턴과 비교 도중 그때까지의 누적거리가 기억하고 있는 최소거리 보다 크게 되면 비교를 중지하고 다른 기준패턴과의 비교를 시작한다. 이렇게 하므로서 인식성능에는 영향을 주지 않으면서도 DTW와 관련된 계산량을 기존의 DTW 계산 방식보다 최고 70%에서 최저 30% 정도 감소할 수 있었다. 또한 위의 방법은 인식 속도에 가중치를 둘 수 있다는 장점이 있다. 즉 빠른 인식속도를 요하는 어휘를 기준패턴 리스트의 앞부분에 놓아 테스트 패턴과 먼저 비교되도록 하므로, 만약 그 테스트 패턴이 빨리 인식되고자 하는 바로 그 어휘라면 전체 누적거리가 작을 것이고 그 이후의 기준 패턴과의 비교에서는 탐색 도중 그 누적거리 값이 기억되어 있는 최소거리를 넘어서게 되고, 다음 기준패턴과의 비교로 넘어가게 되어 실질적으로 많은 계산량의 이득 및 빠른 인식 결과를 얻게 된다.

Ⅳ. VR32의 DSP 구현

이 절에서는 위에서 설명한 알고리즘을 TMS320C32에서 구현하는 부분에 대하여 설명하고자 한다. 구현 환경 및 과정을 간략히 소개하면, C 언어로 작성된 알고리즘과 관련된 부분은 우선 PC 개발환경을 통하여 그 타당성을 검증한 후, 이를 DSP용 C 컴파일러와 일반 DSP 개발용 보드를 이용하여 실시간으로 동작시킨다. 컴파일러로는 Texas Instruments사의 버전 4.7을 사용하였고, 개발용 보드로는 Radisys사의 TMS320C30을 이용한 SPIRIT30을 이용하였다. 일단 기본 알고리즘이 개발용 보드에서 동작이 되면 계산속도와 밀접한 관련이 있는 부분은 어셈블리 언어를 이용하여 최적화 시킨 후, 이를 직접 제작한 하드웨어 모듈에 포팅하는 과정으로 개발을 하였다.

우선 DSP를 이용한 실시간 구현시 항상 우선 고려되는 부분이 음성의 입출력과 관련된 A/D 및 D/A 부분이다. 대부분의 경우 DSP들은 이를 위하여 시리얼포트를 제공하고 있는데 음성인식과 같이 연속해서 들어오는 음성데이타를 실시간에 처리해야 할 경우처럼 계산량이 많은 응용에서는 이 시리얼포트와 관련된 프로그램의 구조가 DSP의 자원을 효율적으로 사용하는데 큰 영향을 미치게 된다. 실시간 프로그램의 기본구조는 그림 5와 같이 더블버퍼링을 사용하는 것이다. 경우에 따라서는 보다 복잡한 구조를 가지는 순환버퍼를 사용하기도 하는데 DSP 응용환경처럼 기준 클럭이 정확한 경우 더블버퍼링으로도 충분하다.

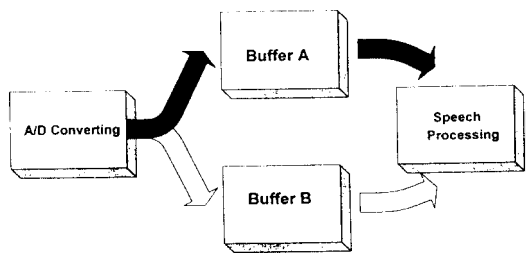

그림 5. 실시간 처리를 위한 더블버퍼링

더블버퍼링 처리에서 가장 흔히 사용되는 것이 시리얼 포트의 인터럽트를 처리하는 것이다. 즉 현재 A 버퍼가 데이터가 차 있는 상태이고 B 버퍼는 비어있다고 가정하자. 이런 상황에서 인터럽트를 이용하는 경우 DSP의 CPU는 A 버퍼의 데이터에 대하여 음성검출, 음성분석, 피치추출을 하는 동시에 시리얼포트로부터 들어오는 1/8000초마다와 인터럽트에 대하여 하던 일을 잠시 멈추고 시리얼포트에 들어온 데이터를 B 버퍼로 이동시킨 후 다시 하던 일을 계속하게 된다. 이런 구조는 명료하고 간단하다는 점에서 쉽게 구현할 수 있고 디버깅도 용이하다는 장점이 있는 반면, CPU가 인터럽트에 반응해야하는 부담을 가지고 있으므로 CPU 본연의 임무에 충실하기 어렵다. 특히 C 언어를 이용하는 경우 인터럽트 서비스루틴으로 갈 때마다 부가적인 여러 가지 스택과 관련된 일들이 추구되므로 상당히 비효율적이 된다. VR32에서는 인터럽트 방식을 이용할 경우 실시간처리가 불가능하였다. 이를 해결할 수 있는 방법으로는 DMA를 이용하는 것이다. DMA를 이용할 경우, CPU는 A 버퍼의 데이터에 대해 실시간으로 처리해야할 계산과 관련된 일만을 하며, 시리얼포트로부터 들어오는 음성데이타는 DMA에 의해 B 버퍼로 저장 된다. 특히 TMS320C32의 경우 DMA의 성능이 매우 우수한데 이는 DMA를 위한 전용의 데이터버스 및 어드레스 버스를 가지고 있기 때문이다. 그렇지 않은 경우에는 소위 버스 스틸링이란 기법을 이용하는데 이 경우에는 DMA를 사용하더라도 CPU의 성능에 다소 영향을 주게 된다. 위에서 설명한 사항은 인식의 경우 뿐만 아니라 합성 시에도 적용된다. 즉 CPU는 합성과 관련된 계산만을 담당하며 계산된 결과를 시리얼포트로 전송하는 것은 DMA가 하게 되는 것이다.

한편 DSP로 구현 시에는 DSP 마다 다소 차이가 있기는 하지만 대부분의 경우, 내부구조는 매우 효율적으로 되어 있는 반면 외부로 나갈 때에는 버스 다중화와 같은 과정을 통하여 비효율적인 구조를 갖게 된다. 예를 들어 내부 버스구조는 매우 세분화되어 있으나 외부로 나갈 때에는 다중화되게 된다. 또한 내부 클럭이 외부보다 빠른 경우에는 내장 데이타 또는 프로그램 메모리를 최대한 활용하는 것이 실시간 구현시 관건이 된다. 가령 TMS320C32의 경우에는 내부메모리를 한 인스트럭션 사이클 동안에 두 번 접근할 수 있다. 그러나 TMS320C32는 가격을 최소로 줄이기 위하여 이 중요한 내부메모리를 대폭 감소시켰다. 상위 버전인 TMS320C30이나 TMS320C31의 경우 2K word의 내부메모리를 가지고 있으나 TMS320C32의 경우에는 512 word로 축소되었다. 따라서 이 적은 내부메모리를 마치 금과옥조와도 같이 아껴 최대한 활용하여야 한다. 이를 위한 기본 지침은 우선 가장 많이 사용되는 데이터 배열을 내부 메모리에 할당하며 사용이 끝난 경우 이를 다음 루틴에서 재사용할 수 있게 하는 것이다. 또한 프로그램 중 가장 많이 사용되면서도 짧은 부분을 이 내부메모리에 적재하는 것이다. VR32의 경우 더블 데이터 버퍼와 DTW와 관련된 데이터 배열이 이 내부 메모리를 사용하며 프로그램의 경우 계산속도와 가장 밀접한 관계가 있는 국부 거리를 계산하는 루틴과 스택이 이 내부 메모리에 할당 되었다. 또한 PCM 코덱을 사용하는 경우에는 μ-law PCM 부호화, 복호화 관련 루틴이 역시 내부메모리에 할당되었다.

마지막으로 DSP 구현과 관련되어 언급할 부분은 어셈블리 언어를 이용한 최적화 부분이다. 사실 이 부분이 DSP를 통한 구현에서 가장 어려운 부분이다. 잘 알려진 바와 같이 DSP가 가진 장점을 최대한 이용하기 위해서는 어셈블리로 구현하여야 한다. 그러나 이는 너무 많은 개발 시간과 노력을 요구하므로 최근에는 DSP에서도 C 언어와 같은 고급언어를 사용하여 개발하는 추세로 변하고 있다. 특히 이런 추세는 고정소수점 DSP에서 두드러지게 나타나고 있는데, 얼마 전까지만 하여도 고정소수점 DSP는 당연히 어셈블리로 구현되어야 한다고 인식되어왔다. 그러나 최근에는 고정소수점 DSP 조차도 고급언어를 이용할 수 있는 구조로 변해가고 있다. 한편 부동소수점 DSP의 경우는 비교적 고급언어의 컴파일러를 사용하여 구현할 경우 어셈블리를 사용하여 구현하는 만큼은 아니지만 높은 효율을 얻을 수 있다. 그럼에도 불구하고 계산속도와 관련된 민감한 부분의 어셈블리 구현은 필수적이라고 하겠다. VR32의 경우 분석과 DTW 관련 부분 중 계산량과 밀접한 관련이 있는 부분은 어셈블리로 구현되었다. 인식 속도와 가장 관련이 있는 부분은 국부거리를 계산하는 부분인데 이 루틴의 가장 안쪽 루프의 계산을 줄이면 전체 계산량도 그와 비례하는 정도로 줄어들게 되므로 최대한 병렬 명령어를 사용하여 최적화하였다. 또한 매 샘플마다 이루어지는 μ-law PCM관련 부호화, 복호화 부분도 어셈블리로 구현되었다. 다만 TMS320C3x 계열의 DSP는 비록 DSP연산에서 가장 중요한 Multiply and Accumulation(MAC) 연산을 병렬명령을 통하여 지원하기는 하지만 구조 자체가 이를 용이하게 사용하기에는 다소 부적합하여 최적화하는데 많은 노력이 소요되었다.

한편 하드웨어와 관련하여는 VR32의 경우 음성인식 모듈이므로 호스트 인터페이스와 관련된 프로토콜을 지원해야 한다. 특히 화자종속의 경우 화자독립 인식기와는 달리 호스트와의 인터페이스가 상당히 복잡하다. 서로 주고 받아야 할 정보로는 인식된 결과 뿐만 아니라 훈련과 관련하여 어떤 어휘를 훈련시킬 것인가를 알려주는 인덱스 정보와 훈련 및 인식 과정의 결과 또는 상태를 알려주는 정보들이 있다. 이를 위하여 VR32는 음성인식에 대한 전문적인 지식이 없는 사람도 사용할 수 있도록 최대한 쉬운 인터페이스를 제공한다. 마치 VR32 모듈을 하나의 부품처럼 사용할 수 있도록 구성되어져 있다.

그림 6과 그림 7은 각각 TLC32044음성 코덱 및 μ-law PCM 코덱을 사용한 VR32 음성인식 모듈을 보여주고 있다. μ-law PCM 코덱을 사용한 VR32의 경우 전원 차단시 훈련 데이터 보존을 위한 배터리 backup 회로를 포함하

고 있다. 또한 그림 8은 이들을 PC에서 평가해볼 수 있도록 만든 평가 보드에 VR32 모듈을 얹은 것이다. 평가보드는 음성신호를 증폭하는 앰프와 PC와 인터페이스를 위한 로직 및 디버깅을 위한 7-세그먼트 LED 출력으로 구성되어 있다.

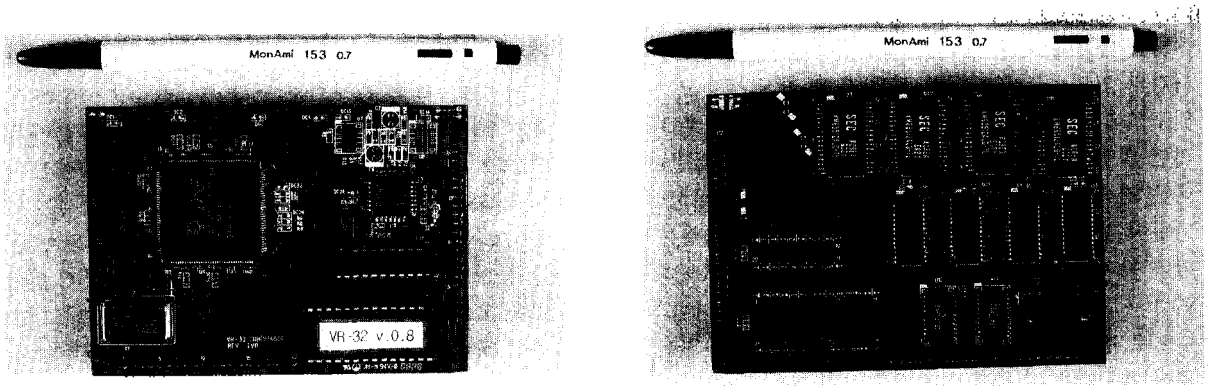

그림 6. TLC32044를 사용한 VR32 음성인식 모듈의 앞면과 뒷면 사진

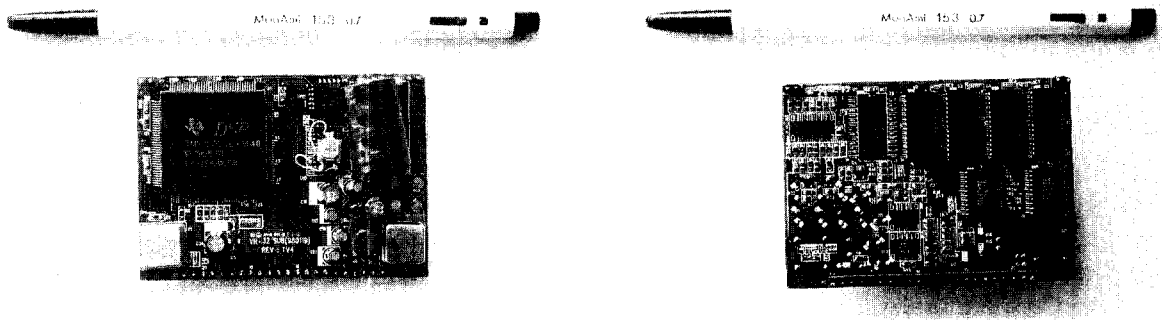

그림 7. μ-law PCM 코덱을 사용하고 음성데이터 보존을 위한 배터리 Backup 기능을 가진 VR32 음성인식 모듈의 앞면과 뒷면 사진

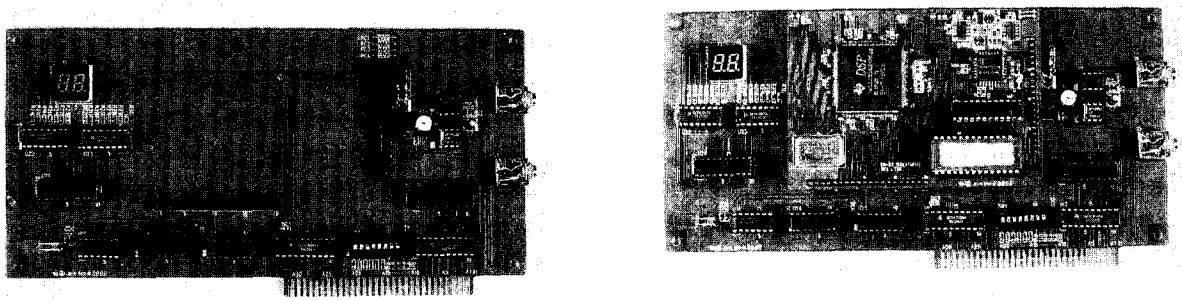

그림 8. VR32의 평가를 위한 PC 인터페이스를 포함한 평가보드와 VR32 모듈을 장착한 평가보드의 사진

인식된 결과를 PC 모니터를 통해 출력하도록하는 프로그램도 함께 만들었다. 그림 9는 PC용 프로그램을 보여주고 있다.

초기 실험실에서 테스트하기 위해 사용한 명령어들은 다음과 같은 것 들로 음성을 통하여 전화를 걸 경우 흔히 사용될 수 있는 단어들이다.

인식테스트는 실제 사무실 환경에서 실시간 인식테스트를 중심적으로 하였다. 전화기의 핸드셋을 이용하여 마이크와 가까운 거리에서 발성을 할 경우 위의 30개 단어에 대하여 97% 이상의 인식률을 보이며 스피커폰과 같은 환경에서의 인식률을 테스트하기 위하여 별도의 AGC 회로를 구성하고 마이크와 약 30cm 거리를 유

지하고 발성할 경우에도 95% 이상의 인식률을 보였다.

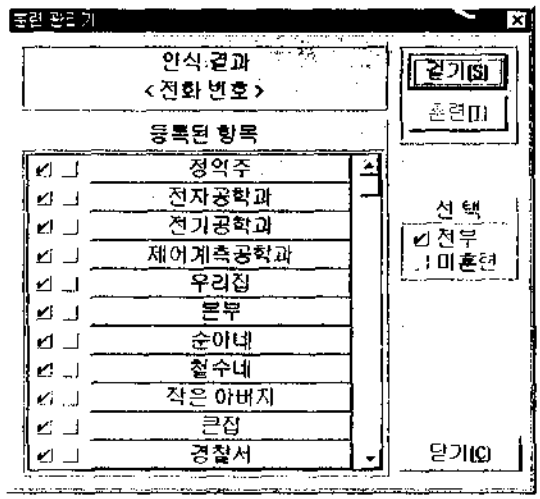

그림 9. VR32 평가를 위한 PC용 소프트웨어

정익주, 강원대학교, 한림대학교, 전자공학과, 전기공학과, 제어계측공학과, 경찰서, 소방서, 사무실, 동사무소, 본부, 우리집, 할아버지, 작은아버지, 철수네, 김수네, 순이네, 중국집, 청와대, 백악관, 삼성전자, 대우자동차, 현대중공업, 총무과, 인사과, 비서실, 응급실, 행정실

V. 결 론

본 논문에서는 TMS320C32를 이용한 화자종속 음성인식 모듈인 VR32의 개발 과정에서 사용된 알고리즘과 실시간 구현 시 고려해야 할 여러 가지 실질적인 내용들에 대하여 논하였다. 특히 실시간 처리에서 가장 중요한 잡음에 강인한 끝점검출 알고리즘을 제안하였으며 음성인식 시스템 구현 시 필요로 하는 최적화 기술 등, 알고리즘을 실제 동작하는 실시간 시스템으로 구현하는 구현 기술에 대해서도 개발 과정에서 얻어진 여러 결과들을 중심으로 소상히 기술하였다. 음성인식 시스템을 구현한다는 것은 음성인식 이론에 바탕을 둔 알고리즘과 더불어 이 알고리즘이 실제 상황에서 발생할 수 있는 여러가지 요소들을 고려할 수 있도록 변형 내지는 개선하는 것이 필수적이며 또한 구현과 관련된 여러 주변 기술들을 필요로 한다. 이번에 개발한 VR32는 저가의 DSP를 사용하여 최근에 급증하고 있는 음성인식 인터페이스에 대한 요구에 적합하도록 많은 요소들을 고려하였다. 실제 상황에서의 VR32의 인식률은 전화기 핸드셋을 이용하여 가까이 대고 발음할 경우 97%를 상회하며 스피커폰을 이용하여 30cm 정도의 거리에서 발음할 경우에도 95%의 인식률을 유지한다. 한편 VR32는 하드웨어의 구조가 확장가능하고 프로그래머블 DSP의 장점인 융통성을 최대한 이용하면 화자독립 인식기의 플랫폼으로도 활용이 가능하다. 화자독립 음성인식을 위해 본 연구실에서는 어휘종속 인식기인 AnyVoice와 어휘독립 인식기인 AnyVoca를 개발 중에 있다.

참 고 문 헌

1. L.R.Rabiner and R.W Scafer, "Digital Processing of Speech Signals", PrenticeHall, 1978.
2. L.R.Rabiner and B.H.Juang, "Fundamentals of Speech Recognition", PrenticeHall, 1993.
3. J.G.Wilpon and L.R.Rabiner, "An improved word-detecton for telephone quality speech incorporating both syntactic and semantic constraint", AT&T Tech. J., 63, No. 3, pp. 476-498, March, 1984.
4. B.S.Atal and L.R.Rabiner, "Pattern recognition approach to Voiced-Unvoiced Silence Classfication with Applications to speech Recognition," IEEE, Trans. Acoust., Speech, Signal Processing, Vol. ASSP-24, No. 3, pp. 201-211, 1976.
5. L.R.Rabiner and M.R.Sambur, "An algorithm for determining the endpoints of isolated utterances," Bell Syst. Tech. J., Vol. 54, pp. 297-315, Feb. 1975.
6. 정훈, 정익주 "Windows용 음성인식 Software 'VoiceAccess' 개발에 관한 연구", Vol. 7, pp. 299-303, 제 7회 신호처리 합동학술대회, 1994.
7. B.H Huang, L. R. Rabiner, and J. G Wilpon, "On the Use of Bandpass Liftering in Speech Recognition", IEEE Transactions of Acoustics, Speech and Signal Processing, Vol. 35, No. 7, pp. 947-954, July 1987.
8. H. Sakoe and S. Chiba "Dynamic Programming Algorithm Optimization for Spoken Word Recognition", IEEE Transactions of Acoustics, Speech and Signal Processing, Vol. ASSP-26, No. 1, pp. 43-49, Feburary, 1978.
9. Cory Myers, L. R. Rabiner, Arron E Rosenberg, "Performance Tradeoffs in Dynamic Time Warping Algorithms for Isolated Word Recognition", IEEE Transactions of Acoustics, Speech and Signal Processing, Vol. ASSP-26, No. 6, pp. 575-582, 1978.
10. F.Itakura, "Minimum prediction residual applied speech Recognition," IEEE Transactions of Acoustics, Speech and Signal Processing, Vol. ASSP-23, pp. 57-72, Feb. 1975.
11. "TMS320C3x User's Guide," Texas Instruments, 1994.
12. "TMS320 Floating-Point DSP Assembly Language Tools User's Guide," Texas Instruments, 1995.
13. "TMS320 Floating-Point DSP Optimizing C Compiler User's Guide," Texas Instruments, 1997

▲정 익 주(Ik Joo Chung)

1986년 2월 : 서울대학교 전자공학과 졸업(공학사)
1988년 2월 : 서울대학교 대학원 전자공학과 졸업(공학석사)
1992년 2월 : 서울대학교 대학원 전자공학과 졸업(공학박사)
1991년 4월~1992년 7월 : (주)두인전자 선임연구원
1992년 8월~현재 : 강원대학교 전자공학과 조교수

▲정 훈(Hoon Chung)
1994년 2월 : 강원대학교 전자공학과 졸업(공학사)
1996년 2월 : 강원대학교 대학원 전자공학과 졸업(공학석사)
1996년 3월~현재 : 강원대학교 대학원 전자공학과 박사과정