論文97-34S-5-9

Windows95 환경에서의 음성 인터페이스 구현

(Implementation of Speech Interface for Windows95)

韓 永 元 *, 裵 建 星 *

(Young Won Han and Keun Sung Bae)

요 약

본 논문에서는 멀티미디어 기능을 갖춘 컴퓨터에 음성인식 기술을 적용하여 Windows95 환경에서의 음성 입력에 의한 사용자 인터페이스를 구현하였다. 음성 인터페이스는 크게 세부분으로 나누어 지는데, 실시간 입력 음성을 검출하는 부분, 입력된 음성을 분석하고 DTW 패턴 정합 방법을 사용하여 입력된 명령어를 알아내는 인식 부분, 그리고 해당되는 명령을 실제로 수행하는 응용 부분이 있다. 음성 인터페이스를 위한 사운드카드 제어 부분은 low-level audio API를 재 구성한 클래스를 만들어 사용하였고, 개발 환경은 한글 Windows95 운영체제하에 Microsoft Visual C++ 4.0 컴파일러를 사용하였다.

Abstract

With recent development of speech recognition technology and multimedia computer systems, more potential applications of voice will become a reality. In this paper, we implement speech interface on the Windows95 environment for practical use of multimedia computers with voice. Speech interface is made up of three modules, that is, speech input and detection module, speech recognition module, and application module. The speech input and detection module handles the low-level audio service of Win32 API to input speech data on real time. The recognition module processes the incoming speech data, and then recognizes the spoken command. DTW pattern matching method is used for speech recognition. The application module executes the voice command properly on PC. Each module of the speech interface is designed and examined on Windows95 environments. Implemented speech interface and experimental results are explained and discussed.

I. 서 론

80년대 초반 개인용 컴퓨터가 등장한 이후 PC는 오늘날까지 그 성능에 있어서 엄청난 발전을 거듭하였다. 그 발전에는 CPU의 속도, 대용량의 메모리와 같은 하드웨어적인 것 뿐만 아니라 사용자의 욕구를 충족시킬 수 있는 여러가지 형태의 시도 또한 포함된다고 할 수 있다. 다양한 애플리케이션, 보다 편리하고도 빠른 사용자 환경, 효율적이고 확장이 용이한 자원관리 방법등, 보다 편리하고자 하는 인간의 욕구는 컴퓨터를 통해서도 여지없이 표현되었다. 특히 기계인 컴퓨터와 사용자인 인간 사이의 차이점을 극복하고자 하는 기계-사람간의 인터페이스 방법에 있어서도 많은 변화가 있었다. 초기의 컴퓨터는 키보드를 통해서 컴퓨터에게 명령을 하고 모니터 화면을 통해서 출력을 하는 다소 기계적인 형태의 인터페이스가 대부분이었다. 키보드를 다루

* 正會員, 慶北大學校 電子·電氣工學部

(School of Electronic and Electrical Engineering, Kyungpook National University)

接受日字: 1997年2月12日, 수정완료일: 1997年4月18日

는 일과 화면에 텍스트로 출력되는 내용들을 살펴보는 일은 초보자가 익숙해지기까지 많은 시간이 걸리는 번거러운 일이었으며, 이러한 결점을 극복해 보고자 진보된 하드웨어를 바탕으로 GUI (Graphic User Interface) 개념이 도입되었다. 이것은 사람에게 보다 익숙하고 인지하기 쉬운 그래픽 형태의 화면을 바탕으로 원하는 부분을 마우스나 키보드의 이동으로 선택할 수 있는 새로운 형태의 인터페이싱 방법으로 초보자라도 금방 이 환경에 익숙해져서 컴퓨터를 사용하게끔 할 수 있는 획기적인 기술로 오늘날의 컴퓨터에는 이미 보편화되기에 이르렀다. 많은 사용자들을 가진 마이크로소프트사의 Windows는 개인용 컴퓨터에 이러한 GUI를 포함시킨 대표적인 운영체제이다. Windows는 버전 3.1 이후 많은 변화를 거듭하여 마침내 32비트 운영체제인 Windows95로 탈바꿈했다. 이전의 버전에서는 텍스트 형식의 MS-DOS를 기반으로 하면서 GUI를 지원하는 형식이었으나, Windows95 부터는 완전한 GUI 형태의 운영체제로 바뀌었다. 이는 보다 편리한 인터페이스를 원하는 사용자들의 요구를 받아들인 것이라고도 할 수 있으며, 앞으로도 이러한 편리한 인터페이스에 대한 요구는 보다 다양한 형태로 끊임없이 계속되리라 생각된다.

최근에는 음성을 통해 컴퓨터와 인터페이싱하는 방법에 대한 연구가 많이 진행되고 있다. 이미 마이크로소프트사에서도 다음 버전의 운영체제에는 사람의 음성을 듣고 그 명령에 따라 컴퓨터가 동작할 수 있도록 하는 방법, 즉 음성인식 기술을 포함할 것이라고 발표한 바가 있다. 국내에서도 음성인식 및 TTS (Text to Speech) 기술을 상용화한 애플리케이션이 발표된 바 있다. 음성은 사람들 간에 통신하는 가장 보편적이고 편리한 방법이다. 하지만 이것을 컴퓨터 환경에 적용한 경우는 그리 흔치 않다. 이는 지금까지의 음성처리기술이 일반 사용자들이 쉽게 사용하기에는 부족한 점이 많았기 때문이다. 하지만 최근의 음성인식기술의 상당한 진보와 이를 수용할 수 있는 컴퓨터 발전으로 인해 음성을 개인용 컴퓨터에 응용할 수 있는 잠재력이 서서히 실현되고 있다. 그래픽적인 환경이 컴퓨터가 사람에게 정보를 전달하는 좋은 방법이라면 음성은 사람이 컴퓨터에게 정보를 전달하는 아주 좋은 방법이 될 수 있다. 왜냐하면 음성은 인간의 기본적이고 자연스러운 의사 소통 수단이므로 특별한 훈련이 필요없이 기계에 적용하여 보다 편리하고 자연스러운 형태의 정보 입력이 가능하기 때문이다. 또한, 음성을 통한 정보전달 과정 중에도 동시에 손, 눈의 사용이 가능하므로 병렬적인 작업이 가능하며, 음성은 글로 쓰는 것보다는 8~10배, 키보드에 비해 3~4배 정도 빠르므로 보다 신속한 입력이 가능하다[1].

본 연구에서는 이러한 음성의 특성을 개인용 컴퓨터에서 활용하는 한 형태로 최근의 음성처리기술, 특히, 음성인식기술을 이용하여 음성을 통한 컴퓨터와의 인터페이스를 구현하고자 하였다. 사용자의 간단한 명령을 컴퓨터가 인지하여 그에 따른 명령을 수행하는 형태로 실제 사용환경에서 응용이 가능하도록 하였으며, 사용자의 편리성을 최대한 고려하였다. 음성인식기술은 화자 종속적인 환경에서 우수한 성능을 보이는 패턴정합 방식의 DTW (Dynamic Time Warping) 알고리즘을 사용한 고립단어 인식기술을 사용하였고, 향후 확장을 고려하여 객체지향적 프로그래밍을 지원하는 MFC (Microsoft Foundation Class)를 사용하여 모든 루틴을 클래스로 구성하였다. 특히 사운드 카드 제어 부분은 보다 세밀한 제어와 호환성을 고려하여 SDK (Software Delepment Kit)의 low-level audio API (Applications Programming Interface)를 재구성하여 클래스로 만들어 사용하였다. 개발 환경은 한글 Windows95 운영체제 하에 MFC를 지원하는 Microsoft visual C++ 4.0 컴파일러를 사용하였다.

II. 음성 인터페이스 시스템의 구성

음성 인터페이스 시스템은 크게 두 부분으로 나누어져 있다. 하나는 사용자의 음성을 훈련시킬 수 있도록 구성한 훈련기이고, 또 하나는 음성을 입력으로 하여 그 의미에 따라 명령을 수행하도록 하는 인식기이다. 이렇게 훈련기와 인식기를 분리하여 구성함으로써 음성 인터페이스 시스템 프로그램의 크기와 수행 부담을 줄일 수 있다. 프로그램의 확장과 수정을 용이하게 하기 위하여 각 기능별로 객체(object)화 하였으며 그 구성은 다음과 같다.

훈련기에서는 사용자의 음성을 입력받아서 특징 파라미터를 추출하여 기준 패턴을 형성하는 부분이다. 기준패턴을 생성해야 하므로 입력 오류를 최소화할 수 있도록 입력 상태 확인 기능을 포함하여 그림 1과 같이 구성하였다.

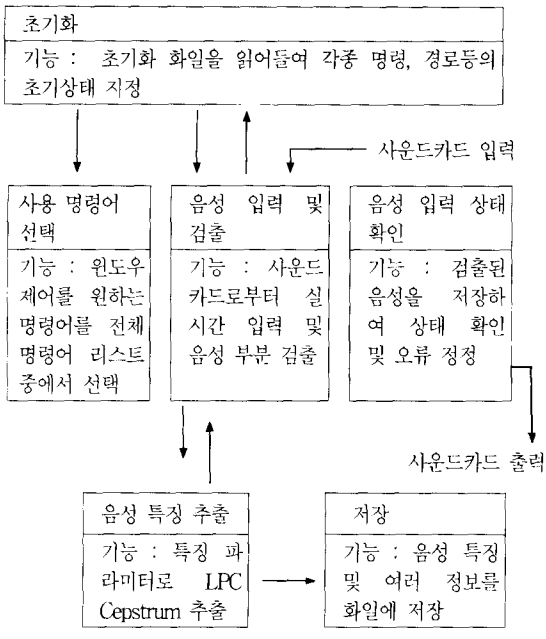


그림 1. 훈련기의 구성
Fig. 1. Structure of the training system.

인식기는 Windows 환경에서 계속 수행되면서 마이크 입력을 감지하여 음성을 입력 받아 인식하여 Windows의 여러 응용 프로그램의 동작을 제어하는 역할을 수행하는데 객체의 구성은 그림 2와 같다.

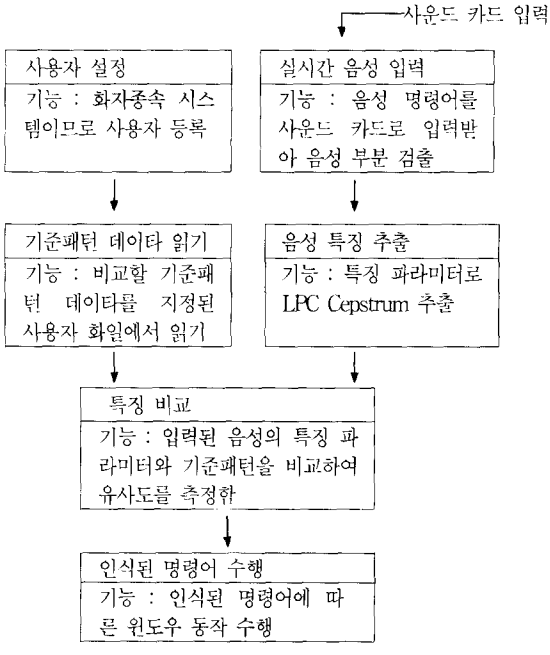


그림 2. 인식기의 구성
Fig. 2. Structure of the recognition system.

1. 음성 입력 및 검출

음성 입력은 마이크를 통해 들어오는 음성신호를 메모리 버퍼를 이용하여 저장하여 음성신호의 시작점과 끝점을 판별하는 기능을 수행하게 된다. 특히 입력 음성신호를 실시간적으로 처리할 수 있도록 고려되어야 하는데, 마이크를 통해 들어온 음성은 PC 사운드카드를 통해 A/D(Analog to Digital) 변환이 이루어지게 된다. Windows 환경에서 PC의 사운드카드 디바이스를 제어하기 위해서 SDK에서 지원하는 low-level audio API 를 사용한다. 사운드카드를 통해서 음성을 입력받기 위해서는 초기화, 입력 시작, 입력 종료 세 가지 과정을 거치게 된다. 초기화 과정에서는 입력용 디바이스 열기, 입력을 저장하기 위한 버퍼 준비, 디바이스에 버퍼 넘겨주기 등을 하게되며, 입력 시작은 이미 넘겨받은 버퍼에 입력 신호를 저장하고 현재 저장하고 있는 버퍼의 위치, 버퍼가 다 찼을 경우 메시지 발생하기 등을 하게 된다. 입력 종료는 사운드카드 디바이스를 리셋(reset)시키고 디바이스를 닫아 사용권을 해제하게 된다.

실시간 처리를 위해서는 여러 가지 방법이 있을 수 있으나 멀티태스킹 환경에서 CPU의 수행부담을 줄일 수 있도록하는 방법으로 다중버퍼 할당 방법을 사용하였다. 초기화 과정에서 디바이스에 버퍼를 넘겨 줄 때 다중의 버퍼를 할당하며 디바이스는 각각의 버퍼가 찰 때마다 콜백(callback)메시지를 발생하도록 하였다. 애플리케이션에서는 이 콜백 메시지를 받아 입력된 음성을 실시간적으로 처리하게 된다. 메시지를 발생한 버퍼는 이미 입력 데이타로 차버린 버퍼로 처리과정을 거친 후 해제하고 계속적인 입력이 가능하도록 새로운 버퍼를 다시 할당하게 된다. 마치 큐(queue) 형식의 버퍼를 계속 할당함으로써 연속적인 입력이 가능해진다. 초기화 과정에서 다수의 버퍼를 할당하는 것은 한 개의 버퍼가 차고 나서 콜백 메시지 발생후 콜백 윈도우에서는 버퍼의 입력 데이터를 처리하고나서 새로운 버퍼를 다시 할당하게 되는데 이러한 콜백 윈도우에서 데이터를 처리하는 동안에도 버퍼에는 계속입력되어야 하므로 다수의 버퍼를 할당하여 동시에 입력과 데이터 처리를 가능하게 하기 위해서이다. 이러한 다중 버퍼에 의한 입력 방식은 버퍼에 데이타가 찰 때 Windows와는 독립적으로 DMA 과정으로 행해지고 버퍼가 다 찼을 때 메시지만으로 버퍼를 제어하게 되므로 처리의 부담을 줄이는 장점이 있다. 본 연구에서

는 8kHz 샘플링 주파수에 샘플당 16비트로 A/D 변환하고 초기 설정 버퍼는 800샘플(1600 bytes) 크기의 버퍼 12개를 할당하였다. 그림 3은 오디오 서비스 처리 과정을 나타낸 것이다.

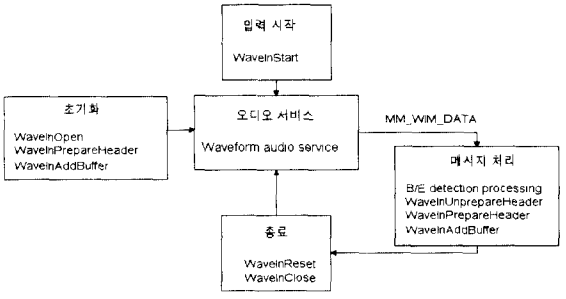


그림 3. 오디오 서비스 처리 과정
Fig. 3. Audio service procedure.

음성의 시작점과 끝점을 주변잡음과 분리하여 정확하게 찾아내는 일은 매우 중요하다. 본 연구에서는 잡음 환경하에서의 효율적인 끝점검출을 위하여 웨이브렛(wavelet) 변환을 이용하는 방법을 사용하였다[2]. 웨이브렛 변환으로 얻어지는 웨이브렛 계수가 잡음에 상당히 강인한 특성을 나타내므로 웨이브렛 영역에서의 스케일별 분포를 이용 잡음에 강인한 끝점 검출이 가능하다.

2. 음성 인식

음성인식 부분은 입력 부분에서 얻어진 음성 데이타로부터 인식에 필요한 특징 파라미터를 추출하여 훈련기에서는 기준패턴을 만들고, 인식기에서는 테스트 패턴을 만들어 기준패턴과의 비교를 통해 인식 결과를 얻게 되는데, 그 구성은 그림 4와 같다.

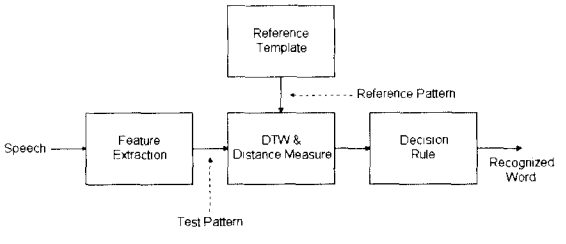


그림 4. 인식 시스템의 블럭도
Fig. 4. Block diagram of recogniton system.

특징 파라미터 추출을 위한 음성신호의 분석 조건은 다음과 같다.

- 샘플링 주파수 : 8 kHz
- 양자화 : 16 bits
- 분석 프레임의 크기 : 160 샘플
- 프레임 이동 간격 : 80 샘플
- 특징 파라미터 : 12차 LPC Cepstrum
- LPC 분석 방법 : Autocorrelation 방법
- Preemphasis factor : 0.95
- 창함수 : 해밍 윈도우(Hamming Window)

시작점 및 끝점 판별이 된 음성신호는 160 샘플을 한 프레임으로 하여 그 절반인 80 샘플씩 이동하면서 매 프레임마다 해밍 창함수를 씌운 다음 $1-0.95\,z^{-1}$의 전처리과정(preemphasis)을 거친 후 자기상관계수를 이용하여 12차 LPC 계수를 구하고 이를 이용하여 12차 LPC 켑스트럼(cepstrum)을 구한다. 이 때 각 프레임의 대수 에너지(log energy)에 해당되는 영차 항은 제외하였다. 구해진 켑스트럼 계수의 높은 차수와 낮은 차수의 바람직하지 못한 변화를 제거하기 위하여 식(1)과 같은 밴드 패스 리프트(band pass lifer : BPL) 형태의 가중 함수를 사용하였다[3].

$$w_k = 1.+0.5\,L\,\sin(\pi\,k/L),\quad k=1,2,\ldots,P \qquad (1)$$

여기서, L은 샘플링 주파수를 고려하여 12 로 정하였고[3], P 는 켑스트럼 차수에 해당된다.

음성인식을 위해서는 입력으로 들어온 테스트 패턴을 인식하기 위하여 미리 저장해둔 기준패턴과 비교하는 패턴정합 작업을 수행한다. 사람이 발음하는 속도와 길이는 개인에 따라 차이가 있으며, 동일한 사람이 같은 발음을 하더라도 발음 속도와 길이는 일반적으로 다르다. 또한 이러한 음성의 신장과 수축은 음성 구간에 따라 그 정도가 다르다. 그러므로 음성의 유사도 측정은 이러한 음성의 특성을 보상할 수 있도록 특징 벡터들을 대응시켜야 하는데, 이러한 문제를 해결하기 위해 동적 프로그래밍 기법을 이용한 DTW 알고리즘을 사용하였다. 이때 유사도 측정을 위한 거리측도로는 켑스트럼의 유클리디안(Euclidean) 거리를 사용하였다.

입력된 음성신호의 테스트 패턴에 대한 인식과정은 다음과 같다. 기준패턴은 훈련기에서 각 단어당 2개씩 마련하도록 하였다. 첫번째 패턴을 I그룹, 두번째 패턴을 II그룹 이라고 하면, 먼저 입력된 테스트 패턴과 I그룹과의 유사도를 측정한다. 이때 가장 유사도가 높은 순서대로 후보 그룹을 설정하게 된다. 그런다음 II그룹

에서 후보 그룹에 해당되는 기준패턴과 테스트 패턴을 비교하여 유사도가 가장 높은, 즉, 거리값이 가장 작은 패턴을 인식된 단어로 결정하게 된다. 이렇게 함으로써 유사도가 높은 패턴 사이에서 잘못 판단되어 일어나는 오인식의 상당한 부분을 유사도가 높은 그룹을 지정해서 다시한번 패턴 비교를 하여 보다 정확한 결과를 얻을 수 있다. 그리고 II그룹에서는 후보 그룹 사이에서만 패턴 비교를 수행하므로 수행 시간을 줄일 수 있는 장점이 있다. 후보 그룹의 갯수는 경험적으로 명령어 수를 5로 나눈 정수 몫을 후보 그룹의 수로 정했으며 총 명령어 수가 10개 이하 일 경우에는 후보 그룹을 두개 정하여 II그룹에서 두개중 유사도가 높은 쪽을 인식된 단어로 결정하도록 하였다.

3. 응용 부분

응용부분에서는 음성인식 부분에서 인식된 결과를 이용하여 Windows의 여러 환경을 제어하는 부분이다. 다른 응용 프로그램을 수행하고 윈도우 모양을 바꾸고 메뉴를 선택하거나 커서의 위치를 바꾸는 등의 음성 명령에 적절한 윈도우 메시지를 발생시켜 Windows 환경을 제어하게 된다.

윈도우 메시지 처리 방법이 16비트 Windows 환경과는 달리 32비트 Windows95에서는 많은 부분이 바뀌었다. 32비트 운영체제인 Windows95에서는 각각이 독립적인 주소 공간을 가지는 쓰레드(thread) 단위로 프로그램이 수행되며 16비트에서 하나의 큐를 사용하던 것과는 달리 각 쓰레드는 각각의 가상 입력 큐(virtualized input queue)와 메시지 큐(message queue)를 가진다. 한 응용 프로그램에서 다른 응용 프로그램에게 메시지를 보내고자 할 경우 메시지를 받을 프레임 윈도우 핸들(frame window handle)을 얻어 통신할 수 있는데, 음성 인터페이스 시스템은 일반적으로 프레임 윈도우 보다는 현재 포커스(focus)를 가진 윈도우에게 메시지를 보내어 제어해야 하므로 각 쓰레드가 독립적인 한 응용 프로그램에서 다른 응용 프로그램의 현재 포커스 윈도우를 알아낼 수 없다. 이러한 문제점을 극복하기 위한 방법으로 쓰레드의 가상 입력 큐를 공유하는 방법을 사용하였다. 이 방법은 16비트 Windows 환경에서 하나의 큐를 사용하던 것처럼 두개의 응용 프로그램의 가상 입력 큐를 공유하여 하나의 가상 입력 큐를 사용하도록 하는 방법이다. 이것을 지원하는 함수의 프로토타입(prototype)은 다음과 같다.

```
BOOL AttachThreadInput(
    DWORD idAttach,      // thread to attach
    DWORD idAttachTo,    // thread to attach to
    BOOL fAttach         // attach or detach
);
```

attach는 자신의 큐를 버리고 다른 응용 쓰레드의 큐에 연결하는 것을 의미하는데 첫 번째의 idAttach는 자신의 가상 입력 큐를 버리고 새로운 큐를 attach할 쓰레드 ID 그리고 두 번째의 idAttachTo는 자신의 가상 입력 큐를 공유하고자하는 쓰레드의 ID를 나타내며 마지막의 fAttach는 TRUE일 경우는 attach하고 FALSE일 경우는 공유를 해제하고 원래되로 복구할 것을 나타낸다. 인식기에서 attach 하기 전과 하고 난 후의 관계를 간략하게 그림 5 및 6에 나타내었다. 그림 5는 attach 하기 전의 인식기와 일반적인 응용 프로그램의 모습인데, 각각 독립적인 큐를 가지고 있음을 알 수 있다. 그림 6은 attach 하고 난 후 응용 프로그램의 가상 입력 큐를 버리고 인식기의 큐를 공유하고 있는 모습이다. attach된 후에는 다른 쓰레드에 메시지를 보내고, 포커스 핸들을 얻는 등의 일을 공통의 큐를 통해 할 수 있게 되는 것이다. 즉 인식기와 응용 프로그램이 하나로 묶여져서 인식기에서 자유롭게 응용 프로그램을 제어할 수 있게 되는 것이다.

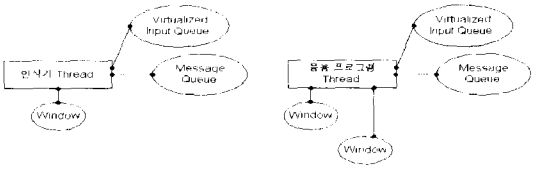


그림 5. Attach하기 전의 인식기와 응용 프로그램의 상태

Fig. 5. State of the recognition system and application before attaching.

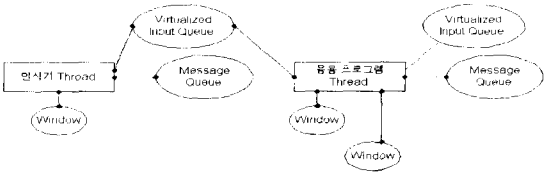


그림 6. Attach하고 난 후의 인식기와 응용 프로그램 상태

Fig. 6. State of the recognition system and application after attaching.

Ⅲ. 결과 및 고찰

그림 7은 본 연구에서 구현된 음성 인터페이스의 훈련기를 실행 시켰을 때의 초기 화면을, 그림 8은 명령어 선택 다이얼로그 박스를 보인 것이다.

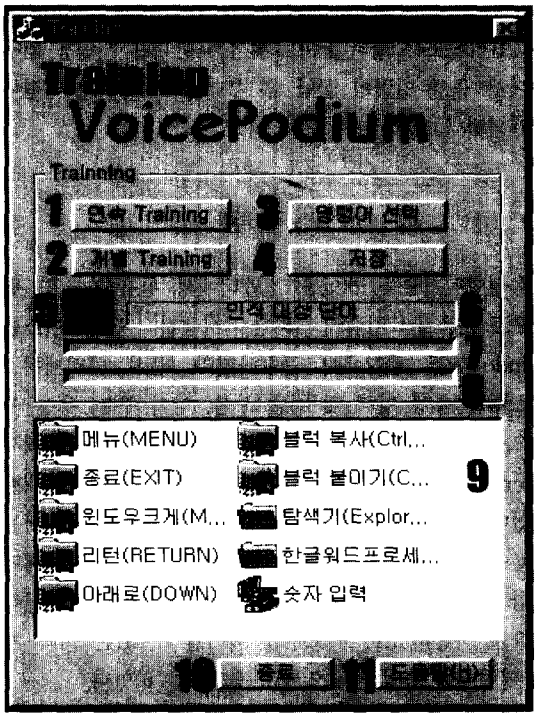


그림 7. 훈련기 초기 화면
Fig. 7. Initial appearance of training system.

그림에서 임의로 표시한 각 번호에 해당하는 기능은 다음과 같다.

- 1 : 연속 훈련 버튼으로 지정된 명령어를 연속적으로 훈련할 수 있도록 한다.
- 2 : 개별 훈련 버튼으로 9번의 리스트 박스에서 선택한 명령어 한 개를 훈련할 수 있도록 한다.
- 3 : 명령어 선택 버튼으로 사용자가 원하는 명령어를 선택할 수 있도록 명령어 선택 다이얼로그 박스를 호출한다.
- 4 : 저장 버튼으로 명령어에 대한 정보와 기준패턴등 인식기에서 사용할 데이타를 형식에 맞춰 저장한다.
- 5 : 남은 훈련 횟수를 LED 형태로 보여준다.
- 6 : 훈련할 단어를 표시해 준다.
- 7 : 음성 입력 레벨을 표시하여 준다.
- 8 : 낮은 입력 레벨을 보다 상세하게 표시하여 준다.
- 9 : 사용자가 선택한 명령어 리스트를 보여주며 선택된 명령어를 두번 마우스 클릭하면 시작점 끝점을 찾은 데이타의 파형과 소리를 스피커를 통해 들을 수 있다. 잘못 입력된 데이타를 검출하는 데 사용될 수 있다.
- 10 : 종료 버튼
- 11 : 도움말 버튼

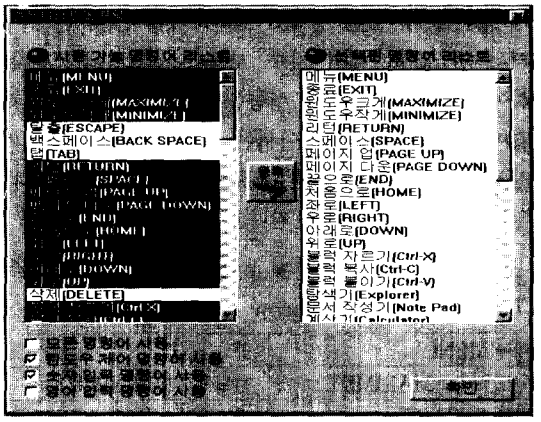


그림 8. 명령어 선택 다이얼로그 박스
Fig. 8. Dialog box for selection of command.

그림 9는 음성 인터페이스 프로그램의 인식기를 실행 시켰을 때의 초기 화면을 보인 것으로, 각 번호에 해당하는 기능은 다음과 같다.

- 1 : 사용자 지정 버튼
- 2 : 인식 시작, 중지 버튼
- 3 : 도움말 버튼
- 4 : 인식기 종료 버튼
- 5 : 마이크 입력 레벨 표시
- 6 : 인식된 단어를 출력
- 7 : 인식모드와 숫자입력 모드, 영어 입력 모드인가를 알려준다.(해당되는 부분이 밝게 표시됨)

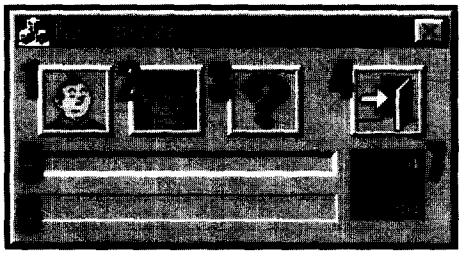


그림 9. 인식기의 초기 화면
Fig. 9. Initial appearance of recognition system.

인식 가능한 명령어는 초기화에서 설정한 명령어 중에서 사용자가 선택하여 원하는 명령어만을 선별하여 훈련시키고 인식하게 할 수 있다. 대체로 너무 많은 명령어는 오히려 사용자로 하여금 기억하기 어렵게 만드는 단점이 있었는데, 이러한 단점은 고립단어 형태의 명령어이기 때문에 사용자가 일상적으로 할 수 있는 명령과는 차이가 많기 때문에 생기는 것이다. 보다 완전한 음성 인터페이스가 되기 위해서는 같은 명령을 수행하는 명령어를 여러 형태로 두어 사용자가 보다 쉽게 접근할 수 있는 방법도 고려되어야 한다. 본 연구에서는 Windows에서 자주 사용하는 명령어 47개를 택하여 구현된 시스템의 성능을 테스트해 보았다. 각 명령어는 내부 처리에 있어서는 각각의 ID로 구분되며 확장을 고려하여 각 ID 번호 부여에 규칙을 두었다. 47개의 명령어는 표 1~3과 같다.

표 1. 범용 윈도우 제어 명령
Table 1. General window control commands.

메뉴(MENU)	종료(EXIT)
윈도우크게(MAXIMIZE)	윈도우작게(MINIMIZE)
탈출(ESCAPE)	백스페이스(BACK SPACE)
탭(TAB)	리턴(RETURN)
스페이스(SPACE)	페이지 업(PAGE UP)
페이지 다운(PAGE DOWN)	끝으로(END)
처음으로(HOME)	좌로(LEFT)
우로(RIGHT)	아래로(DOWN)
위로(UP)	삭제(DELETE)
블럭 잘르기(Ctrl-X)	블럭 복사(Ctrl-C)
블럭 붙이기(Ctrl-V)	

표 2. 응용 프로그램 제어 명령
Table 2. Applications control commands.

탐색기(Explorer)	문서 작성기(Note Pad)
계산기(Calculator)	한글워드프로세서(HWP)
넷스케이프(Netscape)	명령어 리스트
인식 중지	인식
인식기(VoicePodium)	숫자 입력

표 3. 숫자 입력 모드에서 사용 명령
Table 3. Numeric input mode commands.

0	1
2	3
4	5
6	7
8	9
대시(-)	곱하기(*)
나누기(/)	쉼표(,)
마침표(.)	숫자 입력 중지

구현된 시스템으로 성능을 테스트한 결과 인식율에 있어서 사용자에 따라 상당한 차이를 보였다. 처음 사용해 보는 사용자의 경우 상당히 낮은 인식율을 보이다가 점차 사용 환경에 적응하여 인식율이 높아지는 경향을 보였다. 대체로 적응이 어느정도 된 사용자의 경우 실험실 환경에서 95% 이상의 인식율을 보여주었다. 인식속도는 펜티엄 프로세서를 탑재한 개인용 컴퓨터에서 음성 입력이 끝난 후 부터 1초 이내에 명령이 실행되었다.

Ⅳ. 결 론

음성인식기술을 개인용 컴퓨터에서 활용하는 한 방법으로 음성 입력에 의한 사용자 인터페이스를 Windows95 환경에서 구현하였다. 음성 인터페이스 시스템은 음성 입력 및 검출 부분, 인식 부분, 응용 부분으로 구성되는데 한글 Windows95 운영체제에서 Microsoft Visual C++ 4.0 컴파일러를 사용하여 개발되었다. 음성 입력 및 검출을 위한 사운드카드 제어는 low-level audio API를 사용하였고, 음성인식은 패턴정합 방식의 DTW 알고리즘을 사용한 고립단어 인식 기술을 사용하였고, 윈도우를 제어할 수 있는 여러 가지 명령어를 음성 명령으로 수행하도록 하였다.

앞으로 고려해야 할 점으로는 Windows95에서 지원하는 쓰레드 기반 다중작업을 이용하는 방법으로 음성 입력 및 검출 부분, 인식 부분을 다중처리하여 보다 향상된 인식속도를 기대할 수 있으며, 보다 세밀한 윈도우 제어를 위해 윈도우 제어 명령을 다양화하는 방법에 대한 연구가 수행되어야 한다.

※ 본 연구는 정보통신부의 초고속정보통신 응용기술 개발사업의 연구지원 (과제번호: 96-68)에 의해 수행되었으며, 지원에 감사드립니다.

참 고 문 헌

[1] Christopher Schmandt, *Voice Communication with Computers,* Van Nostrand Reinhold, New York, 1994.

[2] 석 종원, 배 건성, "Wavelet 변환을 이용한 잡음 음성의 끝점 검출." *제9회 신호처리합동학술대회 논문집,* Vol. 9, No. 1, pp. 69-72, 1996

[3] B. H. Juang, L. R. Rabiner and J. G. Wilpon, "On the Use of Bandpass Liftering in Speech Recognition," *IEEE Trans. ASSP*, Vol. ASSP-35, No. 7, pp. 947-954, July 1987.

[4] L. R. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1993.

[5] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. ASSP*, Vol. ASSP-26, No. 1, pp. 550-558, Feb. 1978.

[6] 안 태기, 최 진산, 김 민년, 배 건성, "PC에서의 음성에 의한 사용자 인터페이스 구현," *제7회 신호처리합동학술대회 논문집*, Vol. 7, No. 1, pp. 327-330, 1994

[7] 구 명완, "음성인식기술의 현황과 전망," *전자공학회지*, Vol. 20, No. 5, pp. 41-49, 1993

[8] *Microsoft Multimedia Programmer's Reference*, Microsoft Press, 1993.

[9] *Microsoft Multimedia Programmer's Workbook*, Microsoft Press, 1993.

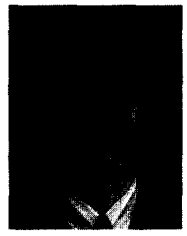
[10] David J. Kruglinski, *Inside Visual C++*^{TM}, Microsoft Press, 1996.

[11] Jeffrey Richter, *Advanced Windows*, Microsoft Press, 1995.

저 자 소 개

韓 永 元(正會員)

1970년 10월 20일생. 1996년 2월: 경북대학교 전자공학과(공학사). 1996년 3월 ~ 현재 경북대학교 전자·전기 공학부 대학원 석사과정. 주관심 분야 : 디지탈신호처리,음성신호처리



裵 建 星(正會員)

1953년 11월 9일생. 1977년 2월 : 서울대학교 전자공학과(공학사). 1979년 2월 : 한국과학기술원 전자공학과(공학석사). 1989년 5월 : University of Florida(공학박사). 1979년 3월 ~ 현재 : 경북대학교 전자·전기 공학부 교수. 주관심 분야:디지탈신호처리, 음성신호처리, 디지탈통신