인터넷 웹페이지의 음성합성을 위한 엔진 및 플러그-인 설계 및 구현

이 희 만† · 김 지 영††

요 약

본 논문은 인터넷 웹페이지의 텍스트 정보를 추출하여 이를 음성으로 합성하기 위한 음성합성 엔진 및 넷스케이프 플러그-인의 설계 및 구현에 관한 것이다. 인터넷 웹페이지를 음성으로 합성하는 방법은 audio/x-esp MIME 타입을 임베딩한 웹페이지가 발견되면 이에 상응하는 플러그-인이 작동되며 해당 플러그-인은 URL로 지정된 HTML문서를 네트워크에서 가져와 컴맨더 오브젝트에 보내고, 컴맨더 오브젝트는 HTML문서를 파싱하여 합성엔진 제어용 TAG를 추출한다. 제어용 TAG에는 음성합성 데이터베이스 변경 및 합성음의 길이 또는 피치조절 파라미터 등의 정보를 갖고 있어 동적으로 합성음을 제어할 수 있다. 또한 컴맨더 오브젝트는 HTML문서 내부의 특정 테그로 지정된 문장을 추출하여 전처리 과정을 수행한 후 합성엔진을 위한 컴맨드 스트림을 발생한다. 음성합성엔진은 컴맨드 스트림을 훼치(Fetch)하여 명령어를 해석하고 해당 명령어에 상응하는 멤버함수를 실행하여 음성을 합성한다. 컴맨더 오브젝트와 음성합성엔진은 각각 독립적인 객체로 설계하여 이식성과 유연성을 높인다.

Design and Implementation of a Speech Synthesis Engine and a Plug-in for Internet Web Page

Heeman Lee† · Ji-Yeong, Kim††

ABSTRACT

In this paper, the design and the implementation of the netscape plug-in and the speech synthesis engine generating the speech sounds from the text information of the web pages are described. The steps of the generating speech sound from an web pages are: the speech synthesis plug-in is activated when the netscape finds the audio/x-esp MIME data type embedded in the browsed web page; the HTML file referenced in the EMBED HTML tag is down loaded from the referenced URL to send to the commander object located in the said plug-in; The speech synthesis engine control tags and the text characters are extracted from the down loaded HTML document by the commander object; the synthesized speech sounds are generated by the speech synthesis engine. The speech synthesis engine interprets the command streams from the commander objects to call the member functions for the processing of the speech segment data in the data banks. The commander object and the speech synthesis engine are designed as an independent object to enhance the flexibility and the portability.

1. 서 론

멀티미디어 정보처리는 숫자와 문자를 주로 처리하던 기존의 정보처리방식에서 벗어나 인간에게 친숙한 시각, 청각형태의 정보를 처리하여 컴퓨터 사용자에게 편리함을 제공한다. 사람의 주요 의사 소통인 언어를 사용하여 컴퓨터 즉 기계와 인간이 대화할 수 있는 기능(Man-machine Interface)은 멀티미디어 정보시대를

† 정 회 원 : 서원대학교 전자계산학과 교수
†† 종신회원 : 서원대학교 전자계산학과 교수
논문접수 : 1998년 12월 17일, 심사완료 : 1999년 12월 13일

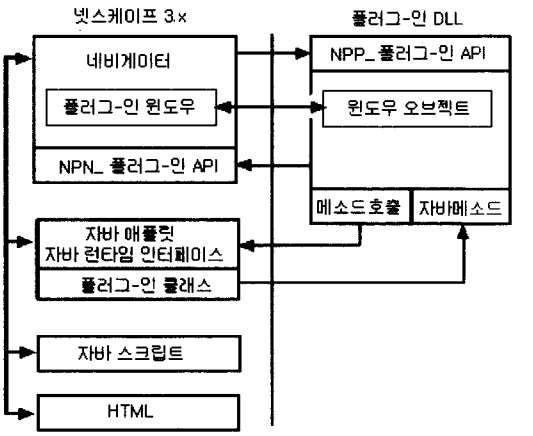
맞아 그 필요성이 어느 때 보다도 요구되고 있다. 인간의 언어를 컴퓨터가 인식하는 음성인식기술과 필요한 음성을 합성하여 인간에게 들려주는 음성합성 기술은 차세대 인터페이스 표준으로 중요한 위치를 차지할 것이다. 그러나 그 동안 많은 노력에도 불구하고 음성을 이용한 인터페이스를 이용하기에는 아직 초보적 단계에 있다. 음성인식은 다양한 화자에 대한 인식 율의 향상이 목표로 연구가 진행되고 있으며 현재 화자종속인 경우 많은 기술적 발전을 하였으나 불특정 다수에 대한 인식 율은 아직도 많은 연구의 필요성이 있다. 음성합성의 경우 합성음의 명료성과 자연성 향상이 요구되지만 현재 부분적 상용화 단계에 접어들고 있다.

최근 인터넷의 열풍과 함께 멀티미디어의 중요성이 부각되면서 멀티미디어의 정보처리와 멀티미디어를 이용한 다양한 응용이 출현하고 있다. 본 논문은 네비게이터를 통해 웹페이지를 항해하는 경우 웹페이지의 특정 태그 안의 텍스트 정보를 음성으로 합성하는 플러그-인 및 음성합성 엔진의 설계 및 구현에 대해 논한다. 인터넷 웹페이지를 음성으로 합성하기 위하여 HTML문서에 audio/x-esp 데이터 타입을 임베딩하면 이에 상응하는 플러그-인이 작동되고 해당 플러그-인에서는 URL로 지정된 HTML문서를 네트워크에서 가져와 특정 태그 부분의 텍스트 정보를 음성으로 합성한다. 플러그-인 내부에 있는 컴맨더 오브젝트는 음성합성엔진을 구동하기위한 명령어를 발생하며 음성합성엔진은 컴맨더 오브젝트가 발생한 컴맨드 스트림을 입력받아 음성을 합성한다. 음성합성엔진은 명료성의 향상을 위해 음절단위의 파형처리 방식을 이용하였으며 아울러 자연성의 향상을 위한 피치조절, 길이 및 에너지 제어를 위해 명령어(command)를 기반으로 쉽게 제어 가능하도록 구현되었다. 컴맨더 오브젝트와 음성합성엔진은 각각 독립적인 객체(Object)로 설계하여 이식성과 유연성을 높인다.

2. 플러그-인의 설계

2.1 플러그-인 아키텍처

넷스케이프 네비게이터는 잘 설계되었고 많은 기능을 가진 소프트웨어로 HTML문서뿐만 아니라 많은 종류의 포맷을 지원한다. 그러나 웹을 사용하는 사람들의 요구사항은 매우 다양하여 모든 종류의 포맷을 지원할 수는 없다. 네비게이터 초기버전에서는 넷스케이프가 처리할 수 없는 미디어에 대해서 헬퍼 어플리케이션을 사용하도록 설계되었다. 헬퍼 어플리케이션은 사용자 시스템에서 동작하는 어플리케이션 프로그램으로 대부분 네트워크와 관련된 일을 하지 않는다. 헬퍼 어플리케이션 프로그램의 단점은 그 자체가 독립적인 별도의 어플리케이션 프로그램으로 사용자는 새로운 프로그램이 로드되는 동안 기다려야 하며, 네비게이터와 다른 위치의 메모리를 사용하므로, 때로는 메모리 부족으로 실행되지 않을 수도 있고 또한 별도의 독립된 윈도우에서 보이게 된다. 일부 네트워크와 관련된 헬퍼 어플리케이션들은 네비게이터와 통신하지 않고 네트워크 소켓 API를 사용하며 어떤 헬퍼 어플리케이션은 넷스케이프의 DDE나 OLE를 이용하여 데이터 교환을 하는 경우도 있다.



(그림 1) 프러그-인 아키텍처

넷스케이프 2.0에서는 헬퍼 어플리케이션의 한계를 극복하는 플러그-인을 사용할 수 있도록 설계되었다. 플러그-인은 플랫폼에 의존적이라는 단점이 있지만 네비게이터와 상호 통신이 가능하고 네비게이터 윈도우 내에서 데이터를 보여 줄 수 있으며, 네비게이터 프로세스에서 실행된다. 플러그-인은 적은 디스크공간을 차지하며 헬퍼 어플리케이션 보다 빨리 로드된다. 넷스케이프 3.0에서는 LiveConnect라는 플러그-인 추가 기능을 지원하여 자바나 자바스크립트와 통합할 수 있도록 플러그-인을 확장하였다. 플러그-인이 클라이언트 플랫폼에 의존적인 반면 자바나 자바 스크립트는 플랫폼 독립적이다. (그림 1)은 넷스케이프 3.x플러그-인 아키텍처이다.

2.2 웹페이지 임베딩

플러그-인은 임베디드, 전체페이지, 히든 모드에서 실행될 수있으며 그중 임베디드 모드는 하나의 웹페이지 결합을 제공하기 때문에 자주 이용된다. 전체 페이지 모드는 플러그-인이 네비게이터 전체 디스플레이 영역을 차지하여 실행되며 히든 모드는 보이지 않는 속성 모드의 플러그-인을 위한 것이다. 웹페이지의 텍스트 부분을 음성으로 합성하기 위해서는 (그림 2)와 같이 audio/x-esp MIME 데이터 타입을 웹페이지 임의의 위치에 임베딩을 한다. 임베디드 플러그-인은 웹페이지 HTML코드에서 EMBED 태그로 시작한다. 즉 플러그-인이 로드될 때 HTML의 부분으로서 나타난다.

```
<EMBED SRC=default.esp RESOURCE=index.html DATABANK=1 HIDDEN>
```

(그림 2) 웹 임베딩

SRC속성은 URL을 이용하여 플러그-인 데이터 파일의 위치를 의미한다. 이 데이터 MIME 타입은 파일을 다루기 위해서 로드되어야 할 플러그-인을 결정한다. 본 연구에서는 .esp 확장자의 타입을 사용한다. default.esp 파일에는 음성합성 엔진의 제어를 위한 기본정보를 갖고 있다. (그림 3)은 defalt.esp 파일에 저장된 내용이다.

```
<ENGIN>
    BANK_LOCATION   "URL of DATABANK"
</ENGIN>
```

(그림 3) default.esp 파일의 예

데이터 뱅크란 음성합성을 위한 음성데이터가 저장된 파일을 의미하며 자신의 로컬 하드디스크에 제한되지 않고 네트워크의 어느 위치에나 존재할 수 있다. 데이터 뱅크가 있는 URL을 BANK_LOCATION에서 지정한다.

RESOURCE속성은 음성합성을 위한 웹페이지를 URL을 이용하여 표시한다. 이 웹페이지는 임베딩을 하는 자기 자신일 수도 있고 또는 네트워크 상의 다른 웹페이지일 수도 있다

2.3 음성합성 제어테그

웹임베딩의 RESOURCE에 지정된 웹페이지에는 음성합성을 위한 제어용 태그를 포함하고 있어야 한다. 현재 사용 가능한 테그는 <표 1>과 같다.

〈표 3〉 음성합성 제어용 테그

TAG	DESCRIPTION
<SYNTHESIS> </SYNTHESIS>	이 태그안에 있는 텍스트만 음성으로 합성한다.
<DATABANK n>	데이터 뱅크를 변경한다. 다양한 화자로 텍스트를 합성할 수 있다.
<SPEED n>	합성음의 속도를 제어한다.
<PITCH n>	합성음의 피치를 제어한다.

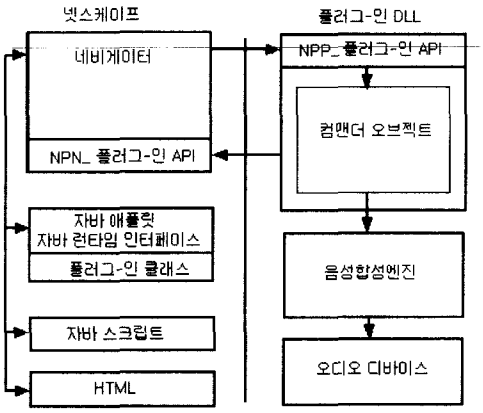
플러그-인은 네비게이터의 스트림으로 받은 HTML 문서를 파싱하고 <SYNTHESIS>...</SYNTHESIS>테그안에 있는 텍스트만을 추출한다. 이 태그 안에는 테이터 뱅크를 변경하거나 합성속도 및 피치를 변경하는 음성합성 제어용 테그가 옵션으로 존재할 수 있다. (그림 4)는 음성합성 제어 태그를 포함하고 있는 HTML 문서의 예이다.

```
<HTML>
<HEAD>
 <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=EUC-KR">
 <META NAME="GENERATOR" CONTENT="Mozilla/4.03 [ko] (Win95; I) [Netscape]">
</HEAD>
<BODY>
<EMBED SRC=default.esp RESOURCE=index.html DATABANK=1 HIDDEN>
<!-- <SYNTHESIS>
 이글은 보이지는 않지만 음성으로 합성됩니다.
 </SYNTHESIS>
-->
 이 글은 화면에 나타나지만 음성으로 합성되지 않습니다.<BR>
 다음 아래의 글은 음성으로 합성 됩니다.<BR>
<SYNTHESIS>
 안녕하세요. 웹페이지를 음성으로 합성합니다. <BR>
 <DATABANK 1> <!-- 음성 데이터 뱅크를 바꾼다 -->
 오늘 하루도 즐거운 시간을 보내세요. <BR>
 <DATABANK 2> <!-- 음성 데이터 뱅크를 바꾼다 -->
 홈페이지를 방문해 주셔서 감사합니다.<BR>
</SYNTHESIS>
</BODY>
</HTML>
```

(그림 4) 음성합성 제어 테그를 포함한 HTML문서

2.3 음성합성 플러그-인 아키텍쳐

음성합성 플러그-인 아키텍쳐는 (그림 5)와 같다. audio/ x-esp MIME데이터 타입이 임베드된 웹페이지를 만나면 음성합성 플러그-인이 자동으로 실행되며 해당 웹페이지를 다운받는다. 플러그-인에 내장된 컴맨더 어브젝트는 HTML에 포함된 음성합성 제어용 태그와 합성할 텍스트를 추출하고 음성합성엔진의 구동을 위한 컴맨드 스트림를 발생하며 음성합성엔진은 컴맨드 스트림을 입력받고 이들 명령에 따라 음성을 합성한다.



(그림 5) 음성합성 플러그-인 아키텍처

2.5 컴맨더 오브젝트

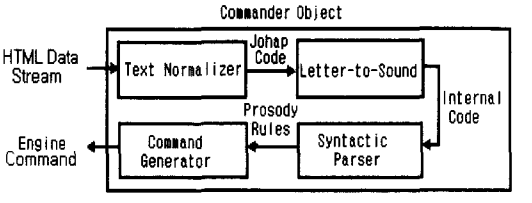
컴맨더 오브젝트는 HTML의 텍스트 문장을 자연스런 음성으로 출력하도록 엔진명령어를 발생한다. 텍스트 문장을 음성합성으로 출력하기 위해서는 여러 단계의 전처리를 하여야 하는데 본 연구에서는 각 단계를 컴맨더 오브젝트의 하위 오브젝트로 상호간 독립성을 유지하도록 설계하여 소프트웨어의 유지보수를 쉽도록 한다.

컴맨더 오브젝트는 플러그-인의 데이터 스트림을 입력으로 한다. 텍스트 문장은 일반 문자 외에도 약어, 축약어, 숫자, 시간, 특수문자 등이 내포되어 있는 데 이를 음성으로 합성하기 전에 일반 텍스트 문장으로 변환할 필요가 있다. 예를 들면, 시간 표시로 "3:30"를 "세시 삼십분", 주소에서 "(우)370-470"를, "우편번호는 삼백칠십 대쉬 사백칠십" 등으로 변경하는 데 이를 텍스트정규화(Text Normalization)이라 하며 텍스트정규화 오브젝트(Text Normalizer Object)에서 처리한다.

컴맨더 오브젝트(Commander Object)는 텍스트정규화 오브젝트에서 처리한 텍스트의 한글코드를 음운변화 등의 처리를 위해 완성형코드에서 조합형 코드로 변환하며 이를 다시 음성 데이터베이스 위치와 관련이 있는 조합형 코드 유형의 내부코드(Internal Code)로 변환한다. 컴맨더 오브젝트 내의 문자소리처리모듈(Letter-to-Sound)에서는 정규 맞춤법에서 소리나는 데로 글자를 표기한다. 텍스트 문장은 글자 그대로 발음되지 않기 때문이다. 즉 "가슴살"이 "가슴쌀", "가슴속" 이 "가슴쏙" 등의 경음화나 또는 구개음화 현상이라든가 또는 음운 환경의 변화에서 오는 음가의 변동, 예를 들면, "부부"의 "ㅂ"이 각각 [p/b]로 변화하기 때문이다. 자연스런 합성음을 생성하기 위해서는 많은 규칙을 내포하고 있어야 한다.

다음단계는 구문분석(Syntactic Parser)으로 텍스트 문장의 문법을 분석하여 단어의 품사를 변별하고 의문문, 평서문 등에 따라 운율제어를 위한 정보를 분석한다. 분석한 정보는 운율제어(Prosody Rules)에 사용된다. 국어의 운율형태를 결정하는 주요자질은 액센트, 리듬, 억양, 휴지 등이 있다[7]. 즉 운율제어라 함은 음높이(Pitch)의 고저, 소리의 크기(Amplitude), 소리의 장단, 리듬 등을 제어하는 것을 말한다. 액센트는 단어내의 상대적인 돋들림으로 음의 고저와 강약, 장단에 의해 결정된다. 리듬은 단어가 모여서 문장을 이룰 때 그 문장 내에서 일정한 운율형태를 반복하는 것을 말하며 국어 연속 음성에서는 앞 뒤 단어들 사이에 피치의 높낮이가 변화하는 주기가 있으며 사람의 호흡시간과도 깊은 관계가 있다. 억양은 연속 음성 중 문장전체의 연속 피치곡선으로 국어의 경우 문장 끝의 억양이 문장전체의 의미적 정보를 나타내며 휴지 또한 중요한 운율요소로 휴지의 유무에 따라 의미가 달라진다.

음성합성의 명료성은 파형연결 처리시 파형의 연속 변화성(Smotheness)을 유지하면 금속성음 등의 불쾌한 음을 쉽게 제거 할 수 있다. 즉 현재의 음성파형은 과거의 N개의 데이터로부터 예측가능하기 때문이다. 일반적으로 피치의 변경 및 연결합성 처리시 금속성음이 들리는 것은 파형이 부드럽게 공명되지 않고 고주파 성분을 많이 함유하기 때문이다. 금속성음은 저역필터 또는 제로 크로싱 연결 및 피치의 불연속변화방지 등의 방법에 의해 비교적 쉽게 해결 가능하다. 그러나 자연성 향상에는 방대한 데이터의 수집과 통계처리 및 모델링 등의 개발을 하여야 하므로 많은 노력과 시간이 필요하므로 팀웍단위의 협력이 요구된다. 컴맨드 발생기(Command Generator)에서는 운율정보에 따라 합성엔진(Speech Synthesis Engine)은 명령어에 따라 음



(그림 6) 컴맨더 오브젝트

성엔진을 가동하기 위한 컴맨드를 생성한다. 음성음성데이터 베이스에 있는 데이터를 가공 연결하여 음성으로 출력한다. 음성합성엔진은 가상장치(Virtual Device)로 복수개의 음성합성엔진을 가동할 수 있다. (그림 6)은 컴맨더 오브젝트의 구성을 보여준다.

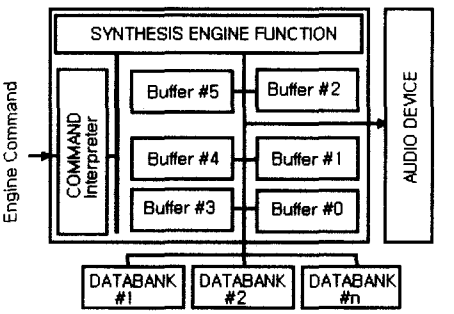
2.6 음성 데이테베이스의 구축

음성의 데이터 베이스를 구축하기 위해서는 합성단위를 선정하여야 한다. 우선 음소를 합성단위로 선정할 경우 국어는 초성 18개, 중성 21개 종성 7개로 46개의 기본 음편이 필요하며, 그 외 변이음을 고려한다 해도 60개미만의 적은 데이터 베이스로 무제한 음성합성기를 작성할 수 있다[9]. 그러나 음소단위의 연결합성의 경우 충분한 품질의 합성음을 얻기가 어려운데 이유는 조음효과(Coarticulatory Effect)에 기인한다. 조음의 효과는 음소파형 중간부분에서 가장 최소가 되는데 이 원리를 이용하여 음성을 합성하는 Diphone방식이 제안되었다[2]. Diphone는 음운 천이부를 포함하므로 비교적 적은 데이터로 합성의 명료성을 확보할 수 있으나 조음결합에 의한 음운변화를 완전히 포함하지는 못한다. 반음절로 데이터 베이스를 구축하는 경우, 초성+중성 조합이 399개 중성+종성 조합이 147개로 기본적으로 546개의 음편이 필요하며 기타 변이음 및 천이구간 음편을 고려하면 약 700여개의 음편으로 비교적 고품질의 음성합성기를 작성할 수 있다. 반음절은 모음의 안정구간을 기준으로 전후로 양분된 데이터를 합성하는 방식으로 Diphone의 연결시 불연속 문제점이 없어 음질이 양호하지만 자음과 자음에 대한 데이터가 없어 음절과 음절사이의 조음결합에 잘 대처하지 못하므로 반음절과 Diphone을 병행하는 방법은 좋은 결과를 얻을 수 있다. 음절단위로 합성시스템을 제작할 경우, 2650자의 기본 음편과 변이음처리를 고려한다면 3600여자 정도의 음편을 녹음하여야 한다.

본 연구에서는 음절 단위의 합성용 음성데이터 베이스를 사용한다. 음성자료를 따로따로 녹음하여 데이터 베이스를 구축하는 경우 F0피치변화 패턴은 초반부에서 중반부까지 피치가 서서히 상승하고 중반부에서 후반부까지는 급격히 하강한다. 이를 연결 합하면 합성음은 연속한 음이 아닌 한음씩 따로따로 들리게 된다. 이를 해결하기 위해서는 파형의 피치를 변경하여야 한다. 그러나 가공처리 단계를 추가할 때마다 음질이 저하되는 문제점이 있다. 본 연구에서 개발한 음성합성엔진은 음절 단위로 구축된 데이터베이스를 사용하고 있으나 음성합성엔진에서 합성단위마다 내부코드를 부여하여 사용하므로 그 어떤 음성단위라도 쉽게 적용하여 사용할 수 있다.

3. 음성합성엔진

음성합성엔진은 오브젝트(Objects)로 구성된 소프트웨어 컴포넌트(Component)로 재사용과 확장이 가능하다. 음성합성엔진은 연결합성을 위한 명령어 기반의 프로그램이 가능한 엔진(Programmable Engine)으로 현재 20여 가지의 명령어가 있으나 추가 및 삭제가 용이하다. 엔진은 컴맨드 스트림을 입력으로 하며 복수개의 음성 데이터뱅크를 사용할 수 있다. 음성합성엔진의 명령어에 의해 화자의 목소리를 음성합성 수행중에 동적으로 변경 가능하여 기존 한사람의 목소리로만 출력되는 방식을 개선한다. 엔진내부에는 컴맨드 인터프리터가 있어 해당 명령어를 해독처리하며 각 명령어에 상응하는 멤버함수를 이용하여 처리하게 된다. 내부의 6개의 독립된 임시메모리(Buffer)는 음성파형을 내부적으로 보관한다. 데이터 뱅크에 있던 데이터는 컴맨드에 의해 내부 버퍼로 불려오게 되고 피치변경 및 지속시간 변경을 위해 가공처리 되면서 다른 버퍼로 이동하게 된다. (그림 7)은 음성합성엔진의 블록다이어그램이다.



(그림 7) 음성합성엔진 아키텍쳐

3.1 음성합성엔진 명령어

명령어는 가변길이의 크기를 가지며 대표적인 명령어를 <표 2>에 요약하였다. <표 2>에서 알 수 있는 바와 같이 음성합성엔진은 데이터뱅크에서 특정 코드

에 해당하는 음성 데이터를 엔진 내부에 있는 버퍼에 적재한 후에 피치, 길이 및 에너지를 변경하며, 이미 처리되어 있는 다른 음성 데이터와 여러 가지 옵션에 의한 연결을 하고, 최종적으로 연주(출력)하는 기능을 한다. 음성합성엔진은 전자악기 MIDI의 시퀀서개념을 도입하였다. 특별히 데이터 뱅크를 변경하지 않으면 기본 데이터뱅크(Bank #1)에 있는 음성파형이 사용되지만 언제든지 가변가능하며 다른 뱅크를 새로 지정하기 전까지는 현재 선택된 뱅크가 계속 사용된다.

〈표 5〉 음성합성엔진의 주요 명령어

Command	Descriptions
0x00	Stop engine(end of command stream) (ex) 0
0x01	Load wave data from the selected databank (ex) 01 <code#> <buff#>
0x02	Clear buffer (ex) 02 <buff#>
0x03	Adjust data to be the specified duration(mili-second) (ex) 03 <S buff> <time> <D-buff>
0x04	Append data from one buffer to another buffer (ex) 04 <S buff> <D buff>
0x05	Stitch data in two buffers (ex) 05 <S %> <D buff> <D%> <# of piches to be m,ixed>
0x07	Modify parts of pitches smoothly (ex) 07 <S buff> <s%> <e%> <s-pitch> <e-pitch>
0x08	Add silience (ex) 08 <S buff> <time>
0x0A	Make the desired pich(Hz) (ex) 0A <S buff> <s%> <e%> <Hz><D buff>
0x0C	Adjust pitch and stitch data (ex) 0C <S1 buff> <s%> <S2 buff> <e%>
0x0D	Mix and append part of data in two buffers with reverve option (ex) 0D <S buff> <s%> <D buff>
0x0E	Energy control (ex) 0E <S buff> <power%> <D buff>
0x10	Select data-bank(speech database) (ex) 10 <bank number>
0x61	Play wave data while clearing buffer (ex) 61 <S buff>

3.2 피치조절

피치조절방법은 TD-PSOLA방식을 사용하였다[2]. 이 합성방식은 다음과 같이 크게 3단계로 구성된다. 제1단계, 피치단위의 분석(Pitch-synchronous Analysis)단계이며, 제2단계, 제1단계에서 분석된 피치단위의 변경단계, 제3단계는 변경된 피치들의 재합성단계이다. 제1단계에서는 피치단위별로 윈도우함수를 곱하여 ST-Signal (Short Term Signal)를 만든다. 윈도우함수로는 일반적으로 해닝윈도우(Hanning Window)를 사용하며, 해닝 윈도우를 이용하는 이유는 깁스현상을 줄이기 위함이나 본 연구에서는 삼각윈도우를 사용하되 제로크로싱 부분을 윈도우의 변두리(Boundary)로 하여 깁스현상(Gibbs Phenomenon)을 줄이면서 처리시간을 향상하였다. 이로 인해 윈도우의 센터는 일반적으로 중심피치(피치마크) 부분을 선정하지만 본 연구에서는 중심피치 파형이 시작되는 제로 크로싱 부분을 선정한다. 무성음의 경우는 일정구간을 윈도우로 선정한다. 피치마킹은 PSOLA방법의 중요한 요소로 잘못 마킹시 명료한 합성음이 나오지 않는다. 본 연구에서는 자동적으로 프로그램에 의해 피치마킹을 하는데 음성신호는 스테이셔너리(Stationary)시그널이 아니므로 피치마킹시에 윈도우의 센터가 원하는 위치에 존재하지 않으므로 위상불결합(Phase Mismatch)을 유발하는 문제점이 있다. 피치마킹을 음성데이터베이스 구축시 핸드마킹 방법에 의해 실시하면 이 문제는 해결할 수 있다고 본다. 그러나 모든 데이터를 핸드 마킹하기에는 많은 시간과 비용이 지출되므로 본 연구에서는 데이터를 녹음하는 프로그램을 별도로 작성하여 피치가 정상적으로 적정 값으로 녹음되었는가를 녹음하는 사람이 확인할 수 있도록 계산하여 보여준다. 그러나 피치불결합(Pitch Mismatch)이나 주파수포락선 불결합(Spectral Envelope Mismatch)은 많은 양의 데이터 베이스를 구축함에 있어 피할 수 없는 PSOLA방법의 한계이다.

3.3 지속시간 조절

음성의 지속시간은 모음부분의 지속시간에 따라 결정되므로 모음부분의 한 구간 파형을 복사하여 삽입하거나 삭제함으로서 제어한다. 그러나 특정위치의 파형만을 삽입하여 지속시간을 길게 하는 경우 해당피치 값만 일정기간 일정하게 유지되므로 매우 어색하고 볼멘소리가 들리게 된다. 지속시간의 변경시에는 모음의 안정기간 부분의 신호를 균등하게 복사 삽입하여야 자연스럽게 조절이 된다.

3.4 에너지 조절

운율제어라 함은 음높이(Pitch)의 고저, 소리의 크기(Amplitude), 소리의 장단, 리듬 등을 제어하는 것을 말한다. 이 중 가장 인간의 귀에 민감한 것이 피치의

변화, 즉 주파수의 변화이며 그 다음은 소리의 장단 및 리듬이고 가장 덜 민감한 것이 소리의 크기이다. 하지만 에너지도 운율제어요소에 있어 중요한 요소이다. 에너지의 조절은 특정 피치 구간의 최대 값을 산출한 후 원하는 에너지가 되도록 비율을 구하고 같은 구간에 있는 데이터 값에 방금 산출한 비율 값으로 곱한다. TD-PSOLA방식의 피치변경은 피치간격의 변화에 따라 주파수를 조절하는데 이때 에너지의 불균형이 생긴다. 그러나 에너지는 자연성과 명료성에 크게 영향이 없으므로 본 연구에서는 에너지 정규화(Energy Normalization)과정을 수행하지 않는다.

3.5 피치조절 연결합성

음성합성엔진은 음성단위에 무관하게 연결합성할 수 있지만 음절 단위의 음성데이터 베이스를 사용하고 있다. 두 개의 음절을 연결 합성할 때 연결부위에서 피치의 불연속이 생길 수 있다. 이는 음성데이터베이스 구축시 피치를 조절하여 녹음하면 되지만 방대한 음성데이터 베이스 구축시 피할 수 없는 문제점이다. 피치가 서로 다른 모음이 연결되면 두 개의 신호는 주파수 공간상에서 공존하므로 귀에서는 각각의 음성을 구별하게 된다. 이로 인해 부자연스런 합성음이 된다.

3.6 음성 데이테베이스의 구축

음성의 데이터 베이스를 구축하기 위해서는 합성단위를 선정하여야 한다. 우선 음소를 합성단위로 선정할 경우 60개미만의 적은 데이터 베이스로 무제한 음성합성기를 작성할 수 있다[9]. 그러나 음소단위의 연결합성의 경우 충분한 품질의 합성음을 얻기가 어려운데 이유는 조음효과(Coarticulatory Effect)에 기인한다. 조음의 효과는 음소파형 중간부분에서 가장 최소가 되는데 이 원리를 이용하여 음성을 합성하는 Diphone방식이 있다[2]. Diphone는 음운 천이부를 포함하므로 비교적 적은 데이터로 합성의 명료성을 확보할 수 있으나 조음결합에 의한 음운변화를 완전히 포함하지는 못한다. 반음절로 데이터 베이스를 구축하는 경우 기본적으로 546개의 음편이 필요하며 기타 변이음 및 천이구간 음편을 고려하면 약 700여개의 음편으로 비교적 고품질의 음성합성기를 작성할 수 있다. 반음절은 모음의 안정구간을 기준으로 전후로 양분된 데이터를 합성하는 방식으로 Diphone의 연결시 불연속 문제점이 없어 음질이 양호하지만 자음과 자음에 대한 데이터가 없어 음절과 음절사이의 조음결합에 잘 대처하지 못하므로 반음절과 Diphone을 병행하는 방법은 좋은 결과를 얻을 수 있다. 음절단위로 합성시스템을 제작할 경우, 2650자의 기본 음편과 변이음처리를 고려한다면 3600여자 정도의 음편이 필요하다.

본 연구에서는 양질의 합성음을 얻기 위하여 음절단위를 사용한다. 음성자료를 따로따로 녹음하여 데이터 베이스를 구축하는 경우 F0피치변화 패턴은 초반부에서 중반부까지 피치가 서서히 상승하고 중반부에서 후반부까지는 급격히 하강한다. 이를 연결 합하면 합성음은 연속한 음이 아닌 한음씩 따로따로 들리게 된다. 이를 해결하기 위해서는 파형의 피치를 변경하는 처리를 선행한다. 그러나 가공처리 단계를 추가할 때마다 음질이 저하되는 문제점이 있다. 본 연구에서 개발한 음성합성엔진은 음절 단위로 구축된 데이터베이스를 사용하고 있으나 음성합성엔진에서 합성단위마다 내부코드를 부여하여 사용하므로 그 어떤 음성단위라도 쉽게 적용하여 사용할 수 있다.

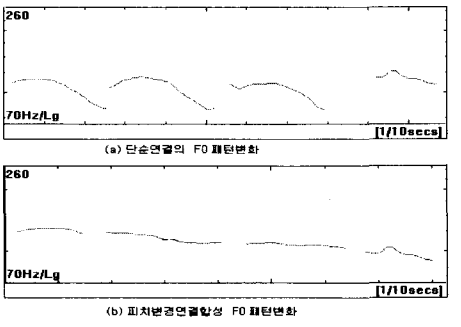
4. 음성합성실험

음성합성은 사운드카드가 장착된 퍼스널 컴퓨터, 윈도우95운영체제에서 소프트웨어만으로 실시간 동작하는 음성합성 엔진을 구동하여 실험하였다. 음성 데이터 뱅크의 제작에는 모든 데이터가 일정한 피치가 되도록 녹음을 하는 것이 중요하다. 전문 성우가 녹음을 하거나 또는 성능 좋은 장비를 이용하면 양질의 녹음 데이터를 얻을 수 있다. 그러나 본 연구에서는 일반인이 평범한 장비를 이용하여 고 품질의 녹음을 할 수 있도록 녹음 전용프로그램을 작성하였다. 즉 일정 시간구간 녹음하도록 신호를 주고 녹음된 데이터의 파형을 디스플레이하며 평균 피치를 산출하여 보여준다. 녹음자는 모든 데이터가 균일한 피치와 적절한 길이(시간)가 되도록 반복녹음을 한다.

데이터는 16비트 데이터 크기와 샘플링 주파수 16KHz를 사용한다. 음성 데이터는 엔진에서 사용하기 위해 에너지의 정규화 등 부분적 처리과정을 거쳐 특정 구조를 갖는 파일에 저장된다. 음절의 합성단위를 사용하며 1890자의 데이터를 이용한다. 음성합성 엔진은 코드번호를 이용하여 음성파형 데이터를 엔진내부에 적재한 후 가공 합성하므로 어떤 단위 즉 음소/반음절/음절 단위의 음성연결합성에도 쉽게 적용될 수 있다.

음성파형 연결합성의 경우 합성음의 명료도는 매우 좋으나 일반적으로 자연성이 나쁘므로 이를 향상시키기 위해서는 음성 파형을 단순히 연결하여 사용할 수 없으며 자연적 운율이 되도록 가공과정이 필요하다. 자연성을 향상시키기 위해 운율을 제어하기 위해서는 피치의 조절, 지속시간의 조절 및 에너지의 조절을 하여야 한다. 의문문이나 감탄문 및 평서문의 경우 어절끝의 피치를 올리거나 내리거나 또는 일정하게 유지하도록 제어한다. 엑센트의 경우 피치를 올려주거나 지속시간을 길게 하거나 또는 에너지를 크기를 조절하며 리듬의 경우 이와 같은 패턴을 일정구간마다 반복함으로서 실현할 수 있다.

문장전체의 F0패턴은 낭독체와 대화체가 매우 다르다. F0패턴이 점차 하강하는 패턴은 거의 모든 언어에서 공통으로 나타나는 현상으로 이는 허파에서 나오는 압력이 발음과정 중 점차 떨어지게 되어 성대의 진동주파수가 내려가는 물리적 현상에 기인한다. 그러므로 휴지 기간 후 처음 발음되는 음절의 피치는 높이고 그 이후는 점차 피치를 내리도록 제어하면 낭독체 문장의 운율을 비교적 쉽게 근사적으로 구현할 수 있다. F0피치 패턴은 전체적으로 불연속이 없도록 제어하는 것이 매우 중요하지만 일반적으로 음성데이터베이스 구축시 한자씩 녹음하면 한 음절내에서 시작부분에서 피치가 증가되다가 모음의 정점에서 최대가 된 후 점차 감소하여 음절이 끝나는 부분에는 피치가 매우 낮아지게 된다(그림 8(a) 참조). 그리하여 2개의 음절을 연결하면 피치가 연속되지 못하므로 외국인이 발음하는 듯이 한자 한자 끊어서 읽는 느낌을 주게 된다. 그림 8(b)는 음절지속시간을 길게 만들어 피치가 서서히 변하게 하며, 그 다음 파형의 후반 부분을 자르고, 다음 음절의 전체 피치를 낮춘 후 연결합성 한 경우의 피치패턴의 변화를 보인 것이다. 이를 통해 자연성은 향상되었으



(그림 8) 피치조절 단어합성

나 신호처리단계를 여러 단계를 거쳤으므로 명료성이 조금 떨어지며 종성이 있는 경우 상기 방법을 사용할 수 없으므로 이 방법은 부분적으로만 적용될 뿐 근본적 해결책은 아니며 데이터 베이스를 잘 구축하는 것이 가장 바람직한 해결책이다.

본 연구에서는 비교적 적은 수의 음운변화 규칙과 운율제어 규칙을 사용하였다. 플러그-인에 있는 컴맨더 오브젝트는 HTML의 데이터 스트림에서 <SYNTHESIS> ...</SYNTHESYS>내부의 텍스트를 정규화 처리한다. 예를 들면 숫자나 영어 스펠링 등 텍스트문장내의 글자를 소리나는 데로 한글로 표기한다. 다음은 한글코드를 내부처리 코드(Internal Code)로 변환한다. 다음 단계는 국어의 경음화나 구개음화등 국어의 문법규칙에 따라 또는 연음처리 등의 문자소리처리(Letter-to-Sound)를 하고 최종적으로 종성의 대표음처리를 한다. 한국어 종성의 종류는 많지만 실제로 소리나는 것은 7개밖에 없기 때문에 데이터베이스에는 종성 7개 종류만 있다. 다음단계는 음성합성엔진을 위한 명령어(Command)를 작성한다. 한 글자에 대한 명령어 작성시에는 낭독체 운율효과를 얻기 위해 최소 3~4개의 명령어가 사용된다.

음성합성엔진의 첫 번째 명령어는 글자(내부코드)에 대한 해당 음성데이터를 메모리에 적제(Loading)하는 명령어이며, 다음 명령은 글자의 문장에서의 역할에 따라 피치 및 길이를 조절하며, 마지막으로 전 문자의 음성데이터와 연결한다. 음절의 피치는 문장의 위치에 따라 달라지지만 낭독체의 경우 한숨구간(One Breathing Period)에서 피치가 서서히 떨어지는 일반규칙만을 적용하였다. 즉 피치는 공백이 없는 문자의 수에 따라 조절하되 두 번째 음절의 경우 첫 번째 글자보다 원래 녹음된 피치에서 5%씩 낮게 조절한다. 사람은 피치에 가장 민감한 반응을 보이므로 피치의 조절은 자연성여부의 가장 중요한 요소이다.

5. 결 론

본 연구에서는 웹페이지의 일부를 음성으로 합성하기 위한 넷스케이프 플러그-인과 컴맨드 기반의 음성합성엔진을 설계 및 구현하였다. 음성합성을 위한 플러그-인은 웹페이지에 임베딩된 해당 MIME 타입이 존재하면 실행되며 플러그-인 내부의 컴맨더 오브젝트는 HTML 데이터 스트림에서 특정 테그안에 있는 텍스트를 음성으로 합성하기 위하여 음성합성엔진을 위

한 컴맨드를 발생한다. HTML 태그에는 음성합성 엔진을 제어하기 위한 태그도 포함될 수있다. 음성합성 엔진은 컴맨드 스트림을 인터프리터하여 데이터 뱅크에 있는 음성 파형을 가공 편집하여 음성으로 합성한다. 각각의 처리 모듈들은 오브젝트(Object)로 설계하여 각 모듈이 독립적으로 쉽게 대체가 가능하도록 하였다. 엔진에서 사용 가능한 명령어들로 구성된 컴맨드 스트림을 음성합성엔진에 입력함으로서 음성을 합성하는 컴맨드 기반의 합성방식은 구문분석, 어휘분석 등의 하이레벨과 파형의 편집 가공 등의 로우레벨을 완전히 분리 가능하므로 시스템의 융통성과 확장성을 높인다. 즉 하이레벨과 로우레벨을 분리하는 아키텍쳐는 팀워크 단위 및 공동연구에 있어 효율성을 높이며 또한 공개적 구조(Open Architecture)는 여러 소프트웨어의 조합적 사용(Mix-and-Match)이 가능하도록 하여 확장성과 이식성을 향상시킨다.

참 고 문 헌

[1] Gordon E. Pelton, Voice Processing, McGRAW-HILL, pp.13-32, 1993.

[2] E. Moulines, F.J. Charpentier, "Pitch-Synchronous Waveform Processing Techniques for Text-to-Speech Synthesis Using Diphones," Speech Communication, Vol.9, No.5-6, pp.453-467, 1990.

[3] F.J.Carpentier, M.G. Stella, "Diphone Synthesis Using An Overlap-Add Technique for Speech WaveForms Concattenation," Proc. ICASSP, pp.2015-2018, 1986.

[4] F.J. Carpentier, E. Mouliens, "TTS Algorithms Based on FFT Synthesis," ICASSP, pp.667-670, 1988.

[5] Thierry Dutoit, Henri Leich, "MBR-PSOLA : Text-to-Synthesis Based On FFT An MBE Re-Synthesis of the Segments Database," Speech Communication, Vol.12, 1993

[6] 양진석, 김재범, 이정현, "운율 및 길이 정보를 이용한 무제한 음성합성기의 설계 및 구현", 한국정보처리학회 논문지, Vol.3, No.5, pp.1121-1129, 1996.

[7] 정국, 구희산, 이찬도, 김종미, "음성인식/합성을 위한 국어의 음성-음운론적 특성연구", 한국음향학회지, Vol.13, No.6, pp.31-43, 1994.

[8] 조철우, 김경태, 이용주, "합성음성평가를 위한 다음절 무의미 단어 생성과 이용에 관한 연구", 한국음향학회지, Vol.13, No.5, pp.51-58, 1994.

[9] 박애희, 양진우, 김순협, "음소단위를 이용한 소규모 문 자음성변환 시스템의 설계 및 구현", 한국음향학회지, Vol.14, No.3, pp.49-60, 1995.

[10] A. Rosenberg, "Effects of Glottal pulse Shape on the Quality of Natual Vowels", J. Acost. Soc. Am, No.49, pp.583-590, 1971.

[11] I. Titze, D. Talkin, "A Theoretical Study of the effects of the various Laryngeal Configurations on the Acoustics of Phonation", J. Acoust. Soc. Am. No.66, pp.60-74, 1979.

[12] Zan Oliphant, Programming Netsacpe Plug-Ins, Sams.net , 1996.

[13] Mike Morgan, Developing for Netscape One, Que, 1997.

[14] 이희만,김지영, "TTS 적용을 위한 음성합성엔진", 한국통신학회지, Vol.23, No.6, pp.1443-1453, 1998.

이 희 만

e-mail : hlee@dragon.seowon.ac.kr

1984년 고려대학교 전자공학과 (공학사)

1986년 한국과학기술원 전기 및 전자 공학과(공학사)

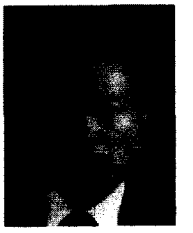
1994년 Texas A&M Univ. Electrical Eng. Ph.D

1986년~1990년 산업연구원(KIET) 연구원

1994년~1996년 삼성중공업 중앙연구소 선임연구원

1996년~현재 (청주)서원대학교 전자계산학과 조교수

관심분야 : 음성합성 및 인식, 가상현실, 스테레오비전



김 지 영

e-mail : jykim@dragon.seowon.ac.kr

1977년 Suny Binghamton Univ. 경영학석사(MBA)

1979년 Suny Binghamton Univ. 전산학 석사

1984년 Suny Binghamton Univ. 전산학박사(Ph.D)

1985년~19988년 Auburn Univ. Auburn AL 전자계산학과 조교수

1988년~현재 (청주)서원대학교 전자계산학과 교수

관심분야 : 시뮬레이션, 컴퓨터비전, HCI