# 선율을 이용한 음악정보 검색 시스템의 설계 및 구현

지 정 규† · 오 해 석††


## 요 약

본 논문은 디지털 음악 도서관에서 임의의 음악정보를 효율적으로 검색하기 위한 시스템의 설계 및 구현에 관한 것이다. 종래의 전형적인 음악정보 검색 항목인 제목이나 작곡자 또는 주제 목록을 입력하는 것이 아니라, 사용자가 음악 데이터베이스로부터 검색하고자 하는 음악의 일부 선율을 마이크를 통해서 노래한다. 그러면 입력된 선율에 대한 음 신호를 처리하여 음표 정보를 인식하고, 이를 바탕으로 음정 곡선을 생성하여 이를 탐색 패턴으로 사용한다. 탐색 패턴을 가지고 제안한 음표열 탐색 알고리즘을 이용하여 근사 탐색을 함으로써 사용자는 노래의 어느 마디를 부르더라도 쉽게 후보곡을 검색하고, 감상할 수 있도록 했다.


# Design and Implementation of Music Information Retrieval System using Melodies

Jeong-gyu Jee† · Heysock Oh††


## ABSTRACT

This paper describes design and implementation of the system that is used to efficiently retrieve music information at a digital music library. Unlike typical music information retrieval systems, this system allows the user to sing a part of the melody through the microphone which he/she wants to find rather than using title, composer or the subject catalog to search. The system then recognizes the musical notes information through the signal processing of the sounds of the entered melody, and the intervals contour is created based on this information and used as a search pattern. By running the proposed notes string search algorithm that uses the musical notes information processed with the user input, and it produces the approximate search results. Therefore, users are able to retrieve and appreciate the music whenever he/she can sing any portion of the desired music.


## 1. 서 론

오늘날 디지털 도서관에 대한 흥미와 관심이 매우 고조되어 국내외의 많은 연구자와 연구기관에서 대규모 지원 프로젝트를 중심으로 연구가 진행되고 있다. 아울러서 단일 주제로서의 국제 학술회의와 전산학 분야는 물론 정보관리학 분야에서의 활발한 학술활동과 새로운 전문잡지의 발간, 그리고 인터넷에서의 많은 활동 등이 두드러진 현상이다[7]. 디지털 도서관은 개인이나 단체에 의해 정보의 생성, 보급, 가공, 저장, 통합, 그리고 재사용 등을 위하여 과감히 장벽들을 없애는 분산 기술 환경으로서, 멀티미디어 기술에 의해서 우리는 이미지, 오디오, 그리고 디지털 비디오 도서관은 물론 WWW 상에 있는 자료를 참조할 수 있는 도서관까지도 생각할 수 있다[4].

† 준 회 원:숭실대학교 전자계산학과
†† 정 회 원:숭실대학교 부총장
논문접수:1997년 11월 6일, 심사완료:1997년 12월 1일

벨기에의 음악학자 페티스는 "음악이란 음의 배합에 의하여 사람의 감정을 감동시키는 예술이다"라고 정의하고 있다[20]. 음악 정보의 검색도 디지털 도서관의 구축과 관련되어 대단히 흥미있는 분야중의 하나로 등장하면서 도전하고 있는 문제이기도 하다. 악보는 전통적으로 제목, 작곡자, 주제분류 등에 의해 목록화되어 있는데, 사서들은 몇 소절의 노래나 콧노래를 기준으로 해당 음악을 찾아 줄 것을 요청 받기도 한다[14]. 그런데 디지털 도서관이 본격화 되면서 보관 중인 음악 정보의 량이 많아지면 더욱 힘든 일이 될 것이다.

그래서 본 논문은 미래의 디지털 음악 도서관의 중요한 구성요소로 정형화 될 수 있는 것으로서 몇 악구를 노래한 것을 기준으로 원하는 음악을 쉽고 빠르게 탐색할 수 있는 시스템의 설계와 구현에 대해 기술한다. 사용자가 노래한 선율을 입력으로 받아서 신호 처리에 의해 음표 정보로 자동변환한 다음, 제안한 탐색 알고리즘을 이용하여 추출된 음표 정보를 바탕으로 효율적인 음악정보 데이터베이스를 검색한다. 이 시스템은 특정 음악의 탐색은 물론 곡의 분석으로 표절의 결과를 예견할 수 있고, 음악 애호가들은 약간 기억나는 악절을 기준으로 곡을 검색할 수 있다. 그리고 혼자서 새로운 노래를 배울 때 정확한 음정을 익히는 학습시스템으로의 활용이 가능하다.

본 논문의 목적 달성을 위해서 설계한 음악정보 검색 시스템의 프로토타입 시스템을 구현하고, 구현된 시스템의 운영 결과를 평가한다. Windows95 상에서 100곡의 동요로 음악 데이터베이스를 구성하고, 구축된 데이터베이스로부터 임의의 곡을 검색한다. 검색을 위해서 곡의 처음과 중간, 그리고 마지막 악절을 골고루 부른 선율을 선율 인식기를 통해 음정 곡선으로 만든 다음, 이를 탐색패턴으로 하여 검색하는 실험을 했다.

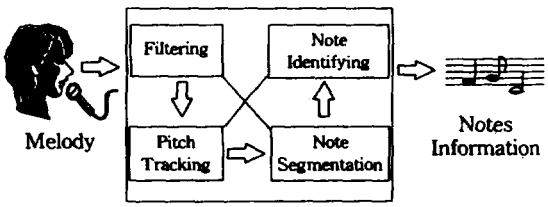
동적 프로그래밍과 상태 대조 알고리즘, 그리고 제안한 근사 탐색 알고리즘을 구현된 시스템에서 실험하여 비교해 본 결과 제안 알고리즘의 탐색 시간이 패턴의 위치에 관계없이 2.2~5배정도 빠르게 나타났다.

선율 검색 시스템은 적용 범위, 재현율, 정도율, 사용자 노력, 응답시간, 출력 형태 등에 따라서 유용성이 좌우 되는데[2], 전통적인 음악 검색 방법은 음악 주제 사전을 찾는 것으로서, "A Dictionary of Opera and Song Themes"와 "The Directory of Tunes and Musical Themes" 등이 그 예다[12]. 선율 검색에 대한 많은 작업이 이루어 졌지만 주로 시스템의 요소에 관계된 것들로서, 반복주기 성분 검출, 악보화, 정확 또는 비정확 문자열 탐색[8], 그리고 멀티미디어 장비를 이용한 노래 연주 시스템[9]이 속한다. 그리고 비교적 완전한 선율 검색 시스템의 예는 [5]의 것을 들수 있는데, 저역 여과기와 자기상관계수법을 이용하여 반복주기 성분을 검출하고, 음 곡선을 추출했다. 그런데 사용자가 노래할 때 음표 사이에 작은 여백을 두도록 하였으며, 리듬은 전혀 계산치 않았다. 그리고 음표의 장단이나 음표의 시작과 끝을 식별하는 절차에 대해서도 언급하지 않았다. [12]은 선율의 실시간 인식에 중점을 두고 악보화하는 연구를 했는데, 음악정보를 악보로 하여 MIDI 음표에 의해 음표 비교를 하도록 했다.

본 논문의 구성을 보면, 먼저 2장에서 선율이 입력된 음향정보를 분석하여 음표정보를 식별하는 절차에 대해 기술하고, 인식 방법을 제안한다. 3장에서는 설계한 디지털 음악정보 검색 시스템과 함께 해당 음표열을 효율적으로 탐색하기 위한 제안 알고리즘에 대해 기술하고, 4장에서는 음악정보 검색 시스템의 구현 결과와 평가에 대해 기술한다. 마지막으로 5장에서는 본 연구의 결론과 앞으로의 연구 방향에 대해 기술한다.

## 2. 선율의 인식

음 신호의 인식은 아날로그 신호를 획득하여 디지털 형태로 변환하고, 원하지 않는 주파수들을 제거하기 위하여 여과한 다음 주파수를 식별하게 된다. 식

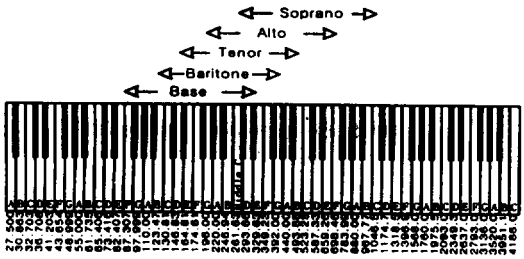



(그림 1) 선율 인식기
(Fig. 1) Melody recognizer

별된 주파수를 분석하여 음표 정보를 인식하게 되는데, 선율 인식기의 구성은 (그림 1)과 같다.

### 2.1 음계와 음표

**음계는 어떤 음을 기점으로 하여 1옥타브 위의 같은 이름의 음에 도달할 때까지 특정된 질서에 의해서 배열된 음렬이고, 옥타브는 음의 높이는 다르나 어떤 음에서 위·아래로 같은 이름의 음인 8번째 음과의 간격으로서[20], 두 음높이간의 간격은 2배로 인식된다.** (그림 2)의 피아노 주파수 범위에서 보듯이 C3은 261.6Hz 주파수를 갖고, 한 옥타브 위인 C4는 523.2Hz, 그리고 한 옥타브 아래인 C2는 130.8Hz이다[1]. 서양음악에서 바하시대 이후의 음계는 한 옥타브를 12개의 같은 간격의 반음으로 나눈 평균율이다[10]. 반음은 서양음악에서 음높이 주기의 가장 작은 단위이지만, 반음의 100분의 1인 센트(cent)로 나누어 사용하기도 하는데, 그러면 1옥타브는 1200센트이다[1]. 이처럼 음높이는 옥타브, 반음, 센트 등으로 구분하여 인식할 수 있는데 비해 주파수는 연속적이기 때문에 주어진 주파수에 반복 주기를 할당하는 것은 양자화에 관련된다.




(그림 2) 피아노의 주파수 범위
(Fig. 2) Frequency range of the piano

**옥타브, 센트 등과 같은 음악적 단위는 상대적인 측정치이기 때문에 센트간의 간격으로 음정을 계산할 수 있으며, 각 음표는 가장 가까운 반음으로 지정되는데, 그것은 음표를 좀 더 세밀하게 표현하기 위해서이다.** 따라서 각 음표는 MIDI 음표 0인 8.176Hz 이상의 센트에서 음표의 간격으로 나타낸다. MIDI는 표준 전자음악 악기로서 표준 서양 음계를 표현하는 기능이 있는데, 음계의 각 음표에 상수를 할당한다. C3는 60을 할당하고, 바로 위 음표인 C#3는 61, 그리고 그 아래인 B2는 59를 할당한다. 평균율로 조율된 음계상의 음표들은 A-440에 상대적으로 100의 배수 센트로 표시한다. 즉, C3는 6000센트이고, A3는 4400센트이다[1].

### 2.2 반복 주기 성분의 검출

음 신호는 1000Hz와 20~40dB에서 여과하고, 22.05KHz로 표본화한 다음 8비트 선형표현으로 양자화 한다. 입력 신호 중에서 기본 주파수의 정수배로 발생하는 배음은 기본음 검출을 혼란스럽게 하므로 가능하면 기본 주파수는 유지한 채 많은 배음을 제거하기 위해 여과한다. 노래하는 음성의 적절한 범위는 보표에 의해 정의되며, 보표의 범위는 F2(87.31Hz)부터 G5(784Hz)까지 이다[11].

반복 주기 성분의 결정은 신호 처리에서 일반적인 연산이지만 매우 어려운 작업이다. 반복 주기 성분을 검출하는 알고리즘은 대체로 3가지 유형으로 분류되는데, 파형의 표본화를 이용하는 시간 영역 검출과 진폭이나 위상 주파수표를 이용하는 주파수 영역, 그리고 시간과 주파수 영역을 같이 이용하는 방법 등이다[3]. 시간 영역에 의한 음높이 검출은 반복주기를 계산하기 위하여 음의 파형을 직접 연산하는 것으로서 주파수 영역 검출보다 구현이 단순하다.

반복 주기 검출을 위한 알고리즘에는 역주파수표법(Cepstrum Method), 단순화 역여과기법(Simplified Inverse Filtering Technique), 자기상관계수법(Autocorrelation Function), 평균차 함수법(Average Magnitude Difference Function) 등이 있는데, [13]의 실험결과 실행시간이 가장 빠르다는 평균차 함수법을 적용하는 것으로 한다.

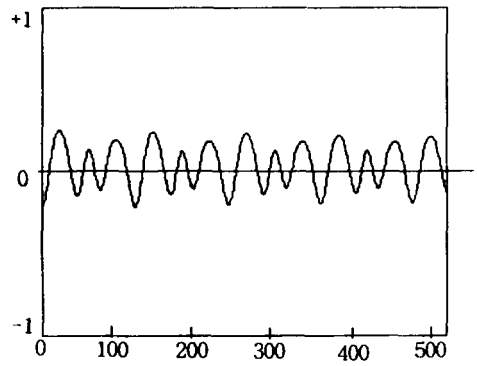
#### (1) 평균차 함수법

음은 주기인 반복 주기 성분을 가지고 있고, 음의 파형은 분절이나 반복 주기 성분의 반복으로 되어 있다[19]. 따라서 음 정보를 추출하기 위해서는 파형의 반복 구조를 발견해 내는 것이 중요하다. 음악에서 반복 주기 성분의 주기는 통상 주기가 시작되는 오름 진폭의 마루에서부터 이므로, 이 마루는 반복 주기 성분 검출기에 의해 사용되는 파형 성분이다. 반복 주기 성분의 검출을 위해 사용하는 평균차 함수는 다음 식

으로 정의된다.

$$R(k)=\sum_{n=1}^{m-k}|x(n)-x(n+k)|$$

m을 512로 하고, 계수 신호 표본 x(n)의 주변에서 서로 k만큼 떨어져 있는 두 신호값의 차들을 구하여 합한 것으로서, 이 값이 작을수록 k만큼 떨어져 있는 두 신호는 서로 유사하다는 의미이다. 따라서 반복 주기를 P라 할 때 k=0, P, 2P, 3P, ... 등의 위치에서 R(k)는 극소값을 가진다. R(k)의 값이 20 이하이면 안정된 음 정보로 볼 수 없으므로 제외한다. (그림 3)은 음이름 "솔"의 파형에서 512Hz를 표본화한 것을 나타낸다.



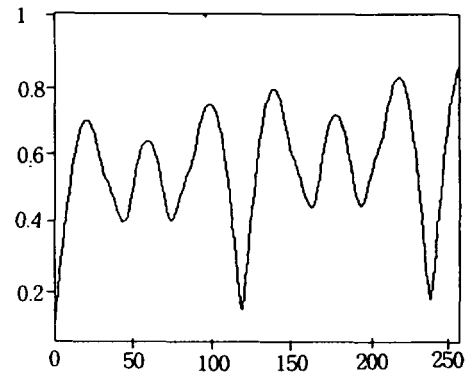

(그림 3) "솔"의 파형
(Fig. 3) Waveform of pitch name "sol"

22.05KHz 표본화 주파수중 512Hz 단위로 구간화하여 연산하는 것은 좀 더 많은 주파수를 단위 연산에 포함하므로서 반복 주기 성분 검출의 정확성과 안정성을 위해서이다.

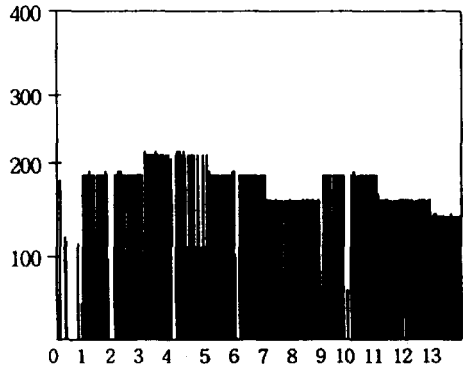
이 파형에서 반복 주기 검출을 위해 평균차 함수를 이용하여 계산한 결과 반복 주기값으로 117이 나왔는데, (그림 4)는 (그림 3)의 파형 중 $\frac{1}{2}$만 표본으로 하여 R(0)부터 R(256)까지 표시되어 있다.

(2) 음표의 분할

완전한 결과를 돌려주는 반복 주기 성분 검출기는 없다. (그림 5)는 반복 주기 성분 검출기에 의해 추출된 음높이 정보를 나타내는데, 모든 표본 위치에서




(그림 4) 평균차 함수에 의한 반복주기 검출
(Fig. 4) Pitch tracking using AMDF



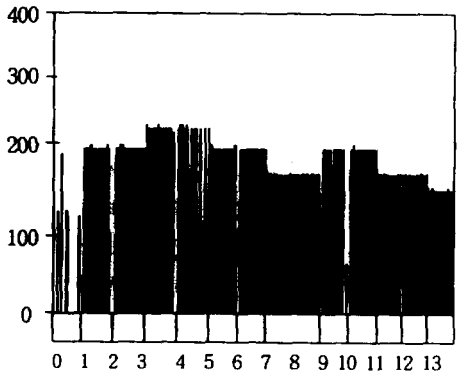

(그림 5) 추출된 음높이 정보
(Fig. 5) Extracted pitch information

반복 주기 성분의 계산으로 이루어 진다.

일반적으로 음의 시작과 끝, 또는 음표 조옮김의 경우에는 신호가 불안정하게 나타나서 오류가 발생하기 쉽다. 그리고 반복 주기 성분 추출을 위한 계산 결과가 실제 주파수의 2배 혹은 반으로 나타날 때는 가장 일반적인 옥타브 오류가 발생하는데, 이같은 원인은 반복 주파수의 시작이 첫 번째 배음으로 되었거나, 반복 주기 성분 길이 계산이 실패했을 경우이다[11].

(그림 5)를 보면 음표간의 경계가 불명확한 부분이 있다. 그래서 앞에서 검출된 반복 주기 성분을 이용하여 반복 주기별로 표본 수열의 합을 구하여 다음 반복 구간의 합과 비교한다. 하나의 음이 끝나갈 때는 음파가 0에 가까워 지다가 새로운 음이 시작되면

다시 급격히 높아지는 파형의 성질을 이용하여 다음의 표본 수열의 합이 바로 전의 표본 수열의 합보다 확연히 커지면 다음 음표의 시작으로 간주해서 구별한다. 이와 같이 표본 수열의 계산에 의해 음표간의 경계를 구분한 것이 (그림 6)이다.




(그림 6) 음표 분할
(Fig. 6) Notes segmentation

### 2.3 음표의 인식

반복 주기 성분에 의한 방법을 이용하여 음표를 분할하는데, 이는 검출된 반복 주기값을 이용하여 반복 프레임을 집단화하고 이들을 비교 연산함으로써 음원 정보로부터 직접 음표를 분할하는 것이다.

분할된 음표 구간 내에는 유사한 주파수 값이 다수 놓이게 된다. 이 중에서 하나의 주파수를 결정하는 방법에는 평균을 구하는 법과, 빈도수에 의한 법, 그리고 통상적으로 신호의 안정 단계로 볼 수 있는 3~4번째의 값을 취하는 방법 등이 있는데, 본 연구에서는 음높이가 낮아질 수 있는 경우를 대비하여 빈도수에 의한 방법을 택하였다.

음표에 대한 주파수가 결정되면 해당 주파수의 음높이를 할당해야 한다. 음높이의 할당은 한 옥타브가 12개의 반음으로 구성되므로, 으뜸음을 Cf라 하고 다음 으뜸음을 C'f라하면 Cf와 C'f 사이에 12개의 반음이 놓이게 된다. 따라서 C'f=2Cf이므로 다음 식에 의해 단위음을 구한다.

$$\log_2 C'f = \log_2 Cf + \frac{k}{12}$$

$$C'f = 2^{\log_2 Cf + \frac{k}{12}}$$

$$k = 12 \times (\log_2 C'f - \log_2 Cf)$$

k의 값은 0에서부터 12까지로 옥타브내의 단위음을 나타내는데, C0(0), C#(1), D(2), ..., A#(10), B(11), C1(12) 등이다. 위 식에 의해 으뜸음의 주파수만 결정되면 나머지 음의 음높이를 구할 수 있다.

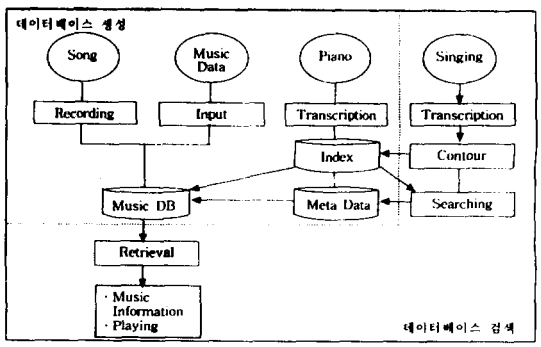
## 3. 음악정보 검색 시스템의 설계

설계한 음악 정보 검색 시스템은 크게 데이터베이스 생성부와 데이터베이스 검색부로 나누어 지며, 검색의 유연성과 효율성을 높이기 위해서 음표열에 의한 근사 탐색을 한다. 그리고 탐색 시간을 보다 빠르게 하기 위하여 색인과 메타 데이터 탐색으로 구분한다.

### 3.1 시스템의 구성

음악 정보 검색 시스템의 구성을 보면 (그림 7)과 같다.

음악 정보 검색 시스템은 크게 데이터베이스 생성부와 데이터베이스 검색부로 이루어 지는데, 데이터베이스 생성부는 음악 데이터베이스와 메타 데이터, 그리고 색인으로 구성된다. 반면에 데이터베이스 검색부는 탐색과 검색, 그리고 검색된 음악 정보의 출력이나 연주 등으로 구성된다.




(그림 7) 음악 정보 검색 시스템의 구성
(Fig. 7) Music information retrieval system

### 3.2 데이터베이스의 생성

음악 데이터베이스를 생성하는데, 탐색을 효율적으로 하기 위해서 메타 데이터와 색인을 추가로 생성한다.

#### (1) 음악 데이터베이스

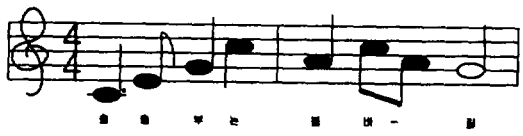
먼저 음악 데이터베이스에 수록 할 대상 동요를 CD 연주기로 연주하면 음악 편집기인 GoldWave로 녹음하여 .WAV 화일을 생성한다. 그리고 해당 동요에 대한 제목, 작곡자, 첫째 악절의 가사 등을 입력하여 .WAV 화일과 함께 (그림 8)의 스키마 형식으로 음악 데이터베이스를 생성한다.

| 순번 | 제목 | 작곡자 | 첫째 악절의 가사 | 전 곡 |
|---|---|---|---|---|

(그림 8) 데이터베이스 스키마
(Fig. 8) Database schema

#### (2) 메타 데이타와 색인

메타 데이터와 색인은 음악 데이터베이스를 효율적으로 검색하기 위해서 생성한다. 음반에 저장된 동요곡은 다중음으로 되어있기 때문에 피아노로 단음 연주한 다음, 이를 음표 인식기를 통해 음표정보로 인식하여 만들어진 음정 곡선을 가지고 메타 데이터를 구성한다. 그리고 메타 데이터 중 첫 악절에 해당하는 부분을 가지고 색인을 구성하는데, 색인의 구조는 근사 키 탐색을 용이하게 하기 위해서 TRIE 구조[6]로 한다. 음정 곡선은 인식된 음표들을 차례로 한 다음, 두 음표 간의 차를 음계에 의해 구한다. 구해진 음정의 값이 0보다 크면 그 수만큼은 A에서부터 M까지 차례로 할당하고, 0보다 작으면 구해진 값만큼은 O에서부터 Z까지 할당하며, 음정의 차가 0, 즉 1도이면 N을 할당하여 차례로 놓는다. (그림 9)는 "봄바람"의 첫 동기인데, 이를 음표(도미솔도라도라솔)들에 대한 음정을 계산하여 음정곡선으로 나타내면 "BBCPBPO"가 된다.



(그림 9) "봄바람"의 첫 동기
(Fig. 9) First motive of "bombaram"

### 3.3 데이터베이스의 검색

사용자가 검색을 원하는 노래 중 알고 있거나 기억나는 부분을 마이크를 통해 노래한다. 그러면 음표 인식기의 선율 인식 절차를 거쳐 음표 정보가 인식되고, 이를 이용하여 음정 곡선을 만들어 탐색 패턴을 구성한다. 구성된 탐색 패턴을 가지고 먼저 색인의 근사 탐색을 시도한다. 탐색이 성공하면 키 정보를 이용하여 바로 음악 데이터베이스를 검색하면 되고, 색인에서 탐색이 실패 했으면 메타 데이터를 기준 음표열로 하여 탐색 패턴에 의한 근사 음표열 탐색을 시도한다. 여기서 근사 음표열이 탐색되면 해당 곡의 키 정보를 이용하여 음악 데이터베이스를 검색한다.

음악 데이터베이스로부터 검색된 음악정보 중 제목, 작곡자, 첫째 악절의 가사, 정확률 등을 정확률의 내림차순으로 화면에 출력한다. 사용자는 화면에 출력된 결과를 토대로 검색하고자 하는 음악을 선택하여 볼 수 있고, 연주가 되게 하여 감상할 수 있다.

### 3.4 음표열 탐색 알고리즘

음표열의 탐색은 정확성과 빠른 탐색 시간이 요구되는데, 먼저 정확한 탐색을 위해서는 탐색코자하는 패턴이 정확해야 한다. 그런데 본 논문에서 다루고자 하는 패턴은 정확성이 낮다고 보아야 한다. 왜냐하면 대부분 전문 음악교육을 전혀 받지 않은 사용자들이 악보도 없이 패턴을 생성해야 하기 때문이다. 따라서 노래하는 형태가 다양하고 원래의 음악에 일치한다는 것이 어렵기 때문에 정확성 탐색을 하게 되면 검색률은 현저히 낮아질 것이다. 이와같이 음표열 탐색을 위한 탐색 패턴의 환경이 다양하기 때문에 디지털 음악 검색 시스템에서는 유연성이 효율성과 깊은 관련이 있다. 따라서 탐색의 유연성을 높이기 위해서 음표열의 정확성 탐색보다 근사 탐색 방법을 택한다.

제안하는 탐색 알고리즘은 먼저 텍스트에 해당하는 전체 음표열을 주사하여 문자 종류별로 발생 위치와 횟수를 조사한다. 문자의 발생 위치 정보를 통해서 음표열을 차례로 비교하지 않아도 탐색 패턴의 처음 문자가 시작되는 위치를 알 수 있으므로 해당 위

치를 직접 찾아서 그 부분만 비교해 보면 그 위치 이외의 다른 곳은 비교할 필요가 없다. 그리고 탐색 패턴의 시작 문자가 나타난 위치를 미리 알고 있으므로 각각의 위치부터 병렬적으로 비교해 나가면 전체 음표열에 걸쳐서 근사 음표열의 발생 여부를 알 수 있다. 이 방법은 패턴 음표열의 시작 문자와 일치하는 문자가 있는 부분만을 직접 찾아서 비교하므로 처음부터 차례로 비교할 필요가 없을뿐만 아니라 비교 대상 위치를 같이 비교해 나감으로써 탐색 시간을 줄일 수 있다. 또한 사전 조사된 위치 정보에 의해 병렬적으로 탐색을 수행하므로 음표열에 있는 패턴의 위치에 따라 탐색 시간이 영향을 받지 않는다. 제안 알고리즘은 음표열 탐색의 특성을 감안하여 대체 연산만을 취급하며, 발생 위치와 횟수 정보 획득을 위해서 탐색 전처리가 필요하다.

제안한 알고리즘을 살펴보면 (그림 10)과 같다.

```
/* 음표열의 발생 위치 및 횟수 조사 */
j = OccurrenceCount = 0
WHILE NOT EOL
    PositionTable[String[j] - 'A'][OccurrenceCount[String[j]
                                                - 'A']] = j
    OccurrenceCount[String[j] - 'A'] += 1
    j += 1
END WHILE
/* 음표열 대조 */
FOR i = 0 TO k BY 1
    Location[n] = PositionTable[pattern[i] - 'A'][n]
                        /* n = 0 to StringLength */
    No_of_Found = OccurrenceCount[Pattern[i] - 'A']
    UnmatchCount[m] = PatternLength - 1
                    /* m = 0 to No_of_Found - 1 */
    FOR m = i+1 TO PatternLength - 1 BY 1
        FOR j = 0 TO No_of_Found - 1 BY 1
            IF Pattern[m] = String[Location[j] + m]
                UnmatchCount[j] -= 1
            END IF
        END FOR
    END FOR
/* 일치여부 검사 */
    FOR k = 0 TO No_of_Found - 1 BY 1
        IF UnmatchCount[k] ≤ i
            RETURN UnmatchCount[k]
        END IF
    END FOR
END FOR
```

(그림 10) 제안 알고리즘
(Fig. 10) The proposed algorithm

## 4. 시스템의 구현 및 평가

앞 장에서 설계한 시스템의 구현을 통해 그 성능을 분석해 본다.

### 4.1 구현 환경

시스템 구현에 필요한 하드웨어로는 오디오 매직 II 사운드/오디오 카드가 설치되고, Windows95가 탑재된 166MHz pentium PC를 이용했으며, 녹음 및 연주를 위해서는 goldwave사의 윈도우즈용 디지털 오디오 편집기인 GoldWave를 사용했다. 그리고 선율 인식과 음표열 탐색을 위한 응용 프로그램은 Turbo C 2.0으로 작성했다.

음악 데이터베이스로는 100곡의 동요[17][18]를 대상으로 전체 선율에 대한 음정 곡선을 만든 다음, 음표열 탐색 알고리즘 실험을 위해서 이를 15,360개 레코드의 음표열을 구성했다. 레코드를 구성하는 음표열은 최고 100개로 되어 있는데, 이는 한 곡의 동요를 구성하는 음표의 최대 개수를 기준으로 하였다.

음악의 일부에 해당하는 선율의 입력은 본인이 3곡의 노래 중 7~10개의 음표열이 속하도록 각각 따로 불러 입력했다. 그리고 제안 탐색 알고리즘의 깊이 있는 실험을 위해서 각 곡으로부터 9개의 음표를 선정했는데, 노래별로 처음 악절, 중간 악절, 그리고 마지막 악절로 나누어 선정 대상으로 했다. 각각의 곡에 대해서 알고리즘별로 12회씩 수행했는데, 이는 음표열의 오류가 없는 경우 3회, 음표열의 오류가 1~3개 있는 경우 각 3회씩이다.

탐색 알고리즘으로는 제안한 알고리즘과 함께 문자열의 근사 탐색에 가장 널리 사용되는 동적 프로그래밍 알고리즘[8]과 상태 대조 알고리즘[16]을 이용하는데, 음표열 탐색의 특성상 대체 연산만 고려한다. 동적 프로그래밍 알고리즘은

$$D_{0j} = 0 \qquad 0 \le j \le n$$
$$D_{i0} = D_{i-1,0} + 1 \qquad 1 \le i \le m$$
$$D_{i,j} = D_{i-1,j-1} + \delta_{ij}$$

$$\delta_{ij} = \begin{cases} 0, & pat_i = str_i \\ 1, & otherwise \end{cases}$$ 이고,

상태 대조 알고리즘은

$$T[x]=\begin{cases}0, & pat_i=x\\ 0, & otherwise\end{cases}$$

$$R_0[k]=0 \qquad 1\le k\le m$$
$$R_j[0]=1 \qquad 0\le j\le n$$

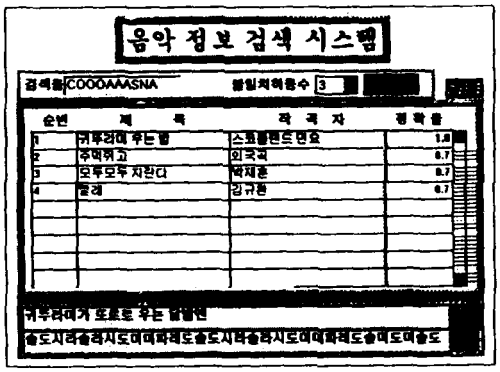
$$R_{j+1}[i]=\begin{cases}1, & R_j[i-1]=1 \ \ AND \ \ p_i=t_{j+1}\\ 0, & otherwise\end{cases}$$

$R_0^d = 11\ldots100\ldots000$ $d$ 1s

$R_{j+1}^d = RShift\ [R_j^d]\ \ AND\ \ T_x\ \ OR$
$RShift\ [R_j^{d-1}]$ 이다.

### 4.2 검색 결과

입력된 선율을 기준으로 음악 데이터베이스를 검색한 결과는 화면으로 출력한다. (그림 11)은 동요 "귀뚜라미 우는 밤"의 처음 부분을 노래하여 검색한 결과 화면이다.




(그림 11) 검색 결과 화면
(Fig. 11) Screen of retrieved results

입력한 선율로 생성된 음정 곡선이 검색음에 나타나 있고, 불일치 허용수는 3으로 했다. 후보곡으로 4곡이 검색되었는데, 이를 정확률에 대한 내림차순으로 제목, 작곡자, 정확률 등을 표시하고, 첫째곡에 대한 첫음절 가사와 계이름을 화면의 아랫부분에 나타낸다. 만일 첫째곡이 원하는 곡이 아니거나, 다음 곡에 대한 정보를 보고 싶으면 내림버튼으로 해당곡을 반전시켜 그 곡에 대한 가사와 계이름을 확인해 볼 수도 있다.

그리고 후보곡 중 감상을 원하는 곡이 선택되었을 때, "감상" 버튼을 누르면 해당 음악을 감상할 수 있다.

### 4.3 성능 평가

〈표 1〉은 음표열의 길이와 불일치 허용수에 따른 음악 데이터베이스의 검색 결과를 나타내고 있다.

먼저 패턴으로 사용하는 음표열의 길이가 길 수록 검색된 후보곡의 수가 적음을 알 수 있는데, 이는 쉽게 검색하고자 하는 곡에 접근할 수 있음을 뜻한다. 음표열의 길이란 사용자가 검색을 위해 부른 선율의 길이를 말하는데, 너무 짧게 부르면 부르기는 쉽지만 많은 후보곡이 대상으로 검색될 것이고, 반대로 너무 길게 부르면 원하는 곡의 탐색은 용이하지만 부르는 데 힘이 들 것이다. 따라서 보통 7~10개의 음표로 구성되는 2마디 정도를 부르면 적당하리라 생각된다.

〈표 1〉 음표열의 길이별 검색 결과
〈Table 1〉 Retrieved results by length of patterns

| 패턴 음표열의 길이/ 패턴 음표열 | 불일치 허용수 | 검색된 후보곡의 수 ① | ② | ③ |
|---|---|---|---|---|
| 7 | 0 | 1 | 1 | 1 |
| ① BBNNPAA | 1 | 2 | 1 | 2 |
| ② PBPBAOP | 2 | 4 | 4 | 5 |
| ③ NANONPB | 3 | 22 | 16 | 38 |
| 8 | 0 | 1 | 1 | 1 |
| ① BBNNPAAP | 1 | 1 | 1 | 1 |
| ② PBPBAOPB | 2 | 2 | 1 | 3 |
| ③ NANONPBN | 3 | 6 | 5 | 21 |
| 9 | 0 | 1 | 1 | 1 |
| ① BBNNPAAPP | 1 | 1 | 1 | 1 |
| ② PBPBAOPBP | 2 | 1 | 1 | 3 |
| ③ NANONPBNP | 3 | 2 | 2 | 6 |
| 10 | 0 | 1 | 1 | 1 |
| ① BBNNPAAPPN | 1 | 1 | 1 | 1 |
| ② PBPBAOPBPP | 2 | 1 | 1 | 1 |
| ③ NANONPBNPN | 3 | 1 | 1 | 3 |

그리고 불일치 허용수를 0으로 하여 정확성 탐색을 하면 후보곡을 찾는데는 가장 좋다. 그러나 사용자가 탐색하기 위해 노래의 음정을 완전하게 부르는 것이 쉽지 않기 때문에 불일치를 허용하는 것이 타당하다. 불일치 허용수가 커질수록 검색되는 후보곡이 많아지게 되므로 경우에 따라 조절이 가능하도록 하되, 우선은 불일치 허용수는 2 정도로 두는 것이 무난하리라 본다.

〈표 2〉를 살펴보면 대체로 전체 음표열 중 찾고자 하는 음표열이 뒤쪽에 위치할수록 탐색 시간이 많이 걸린다. 그리고 근사 탐색이 정확성 탐색보다 더 많은 시간이 소요되고, 근사 탐색 중에서도 오류 허용수를 늘릴수록 탐색 시간도 늘어남을 볼 수 있다.

제안한 알고리즘은 음표열의 위치에 따른 탐색 시간이 다른 두 알고리즘보다 덜 민감함을 보이고 있다. 그리고 평균 탐색 시간 또한 동적 프로그래밍 알고리즘에 비해서는 5배, 상태 대조 알고리즘에 비해서는 2.2배로 훨씬 빠르다는 것이 증명되었다.

〈표 2〉 탐색 시간 비교
〈Table 2〉 Comparison of search times

| 구 분 | | Dynamic Programming | State Matching | Proposed Algorithm |
|---|---|---|---|---|
| 동기 | 오류 수 | | | |
| 앞 | 0 | 10.86 | 3.26 | 1.83 |
| | 1 | 10.87 | 4.69 | 2.13 |
| | 2 | 10.95 | 6.08 | 2.27 |
| | 3 | 12.49 | 7.78 | 3.84 |
| | 평균 | 11.29 | 5.45 | 2.52 |
| 중간 | 0 | 10.82 | 3.32 | 2.06 |
| | 1 | 11.20 | 4.70 | 2.36 |
| | 2 | 11.35 | 6.00 | 2.75 |
| | 3 | 13.21 | 8.26 | 3.32 |
| | 평균 | 11.65 | 5.57 | 2.62 |
| 뒤 | 0 | 10.87 | 3.31 | 2.07 |
| | 1 | 11.07 | 4.81 | 2.40 |
| | 2 | 11.73 | 6.10 | 2.67 |
| | 3 | 13.88 | 8.49 | 3.28 |
| | 평균 | 15.85 | 5.68 | 2.61 |
| 전체 평균 | | 12.93 | 5.57 | 2.58 |

## 5. 결 론

본 논문은 음향 입력으로부터 음 신호를 처리한 다음 선율을 인식하고, 인식된 음정 곡선을 탐색 패턴으로 하여 적절한 음악 정보를 효율적으로 검색하기 위한 시스템을 설계하고 구현했다. 마이크를 통해서 노래한 음악의 일부분에 해당하는 음 신호를 분석하여 대응되는 음표를 인식하기 위해서 원래의 아나로그 신호는 표본율을 22.05KHz로 하여 양자화하고, 저역 통과 대역과 통과 대역 여과기를 이용하여 1000Hz 이상의 배음과 20~40dB 범위 밖의 음을 잡음으로 간주하여 골라 내었다. 그리고 반복 주기 성분을 검출하기 위하여 시간 영역 중심의 평균차 함수를 이용하였다. 반복 주기표로부터 출력된 음원 정보로부터 음표를 분할하기 위하여 반복 주기 성분을 적용하는데, 표본화 구간별로 표본 수열의 합을 구하여 구간 값을 비교한다. 분할된 음표를 식별하기 위하여 동일한 음표 구간 내에서 발생한 다수의 주파수 값 중에서 빈도수가 가장 많은 값을 해당 음표의 주파수로 결정한다.

식별된 음표들간의 음정을 계산하여 음정 곡선을 구한 다음 이를 탐색 패턴으로 이용한다. 피아노로 단음 연주한 곡을 자동 인식하여 메타 데이터를 생성하고, 첫째 동기를 이용하여 Trie 구조의 색인을 구성한다. 탐색 패턴을 가지고 색인을 참조하여 곡을 검색한 다음 검색이 실패했을 경우 메타 데이터를 다시 탐색한다. 색인이나 메타 데이터에서 탐색된 키값을 기준으로 음악 데이터베이스로부터 음악정보가 검색되어 그 결과가 출력되고, 선택에 따라 연주가 이루어 진다. 색인이나 메타 데이터 탐색시에는 근사 탐색을 하여 불완전한 입력 음 신호로부터 효율적인 검색이 이루어 지도록 했다. 그리고 제안한 음표열의 근사 탐색 알고리즘은 기존의 알고리즘보다 탐색 속도가 2배이상 향상 되었다.

이 시스템은 디지털 음악 도서관에서 감상을 필요로 하는 음악을 검색하는 시스템으로 활용하기 위한 것이다. 통상적인 음악 정보 검색 항목인 제목, 작곡자, 주제분야 등에 대한 정보를 알지 못한 상태에서 단지 몇 소절의 노래만 부름으로써 원하는 음악 정보를 검색할 수 있도록 하는 시스템으로서, 수 많은 음악을 저장해 둔 대규모 음악 데이터베이스로부터 사용자는 쉽고 빠르게 음악 감상을 위한 곡의 검색을 할 수 있을 것이다. 또한 개인의 음악 학습시스템으로 활용하면 자신이 부른 음정이 실제 악보상의 음정

과 어떤 차이가 있는지를 확인하면서 틀린 부분을 바르게 부를 수 있을 것이다.

앞으로 선율 인식의 실시간 처리와 함께 가사로 노래를 부를 때 발생하는 연음의 식별을 위한 연구가 필요하며, 복합음으로 된 선율의 인식에 대한 연구도 계속 되어야 할 것으로 생각된다.

## 참 고 문 헌


[1] Backus, J., The Acoustical Foundations of Music, John Murray, 1969.

[2] Baker, S. L., Lancaster, F. W., The Measurement and Evaluation of Library Services, Information Resources Press, 1991.

[3] Eroglu, C., "A Word Spotting Algorithm Based on Pitch Detection and Hidden Markov Models", http://www.ee.bilkent.edu.tr/~eroglu/report.

[4] Fox, E. A., "Digital Libraries", Communication of The ACM, Vol. 38, No. 4, Apr. 1995.

[5] Ghias, A., Logan, J., Chamberlin, D. and Smith, B. C., "Query by Humming: Musical Information Retrieval in an Audio Database", Proc. ACM Multimedia '95, 1995.

[6] Hall, Patrick and Dowling, G. R., "Approximate String Matching", Computing Surveys, Vol. 12, No. 4, pp. 381-402, Dec. 1980.

[7] Harter, S. P., "What is a Digital Library? Definitions, Content, and Issues", Proceedings of the ICDL, 1996.

[8] Landau, G. M., Vishkin, U., "Efficient String Matching with K Mismatches", Theoretical Computer Science, Vol. 43, pp. 239-249, 1986.

[9] Loeb, S., "Architecting Personalized Delivery of Multimedia Information", Communications of the ACM, Vol. 35, No. 12, pp. 39-48, 1992.

[10] Martin, D. W., "Musical Scales since Pythagoras", Sound, Vol. 1, No. 3, pp. 22-24, 1962.

[11] McNab, R. J., Smith, L. A. and Witten, I. H., "Signal Processing for Melody Transcription", Proc Australasian Computer Science Conference, pp. 301-307, 1996.

[12] McNab, R. J., Smith, L. A., Bainbridge, D. and Witten, I. H., "The New Zealand Digital Library MELody inDEX", D-Lib Magazine, May 1997.

[13] Padir, H., "An LPC Vocoder System", M. Sc. Thesis, Middle East Technical University, 1983.

[14] Smith, L. A., McNab, R. J. and Witten, I. H., "Music Information Retrieval Using Audio Input", AAAI Symposium, 1996.

[15] Steiglitz, K., A Digital Signal Processing Primer, Addison-Wesley Publishing, 1996.

[16] Wu, S., Mander, U., "Fast Text Searching Allowing Errors", Communication of the ACM, Vol. 35, No. 10, p. 83-91, 1992.

[17] 김경렬, 동요 피아노 소곡집, 현대음악출판사, 1993.

[18] 김규환, 피아노 동요곡집1, 일신서적공사, 1983.

[19] 박경범, 음성의 분석 및 합성과 그 응용, 도서출판 그린, 1997.

[20] 백병동, 개정 대학음악이론, 현대음악출판사, 1997.



**지 정 규**

1987년 서울산업대학교 전자계산학과(공학사)

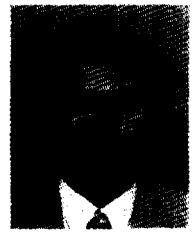
1989년 숭실대학교 전자계산학과(공학석사)

1995년~현재 숭실대학교 전자계산학과 박사과정

1978년~1996년 (주)삼호, 서울시설관리공단 전산실

1996년~현재 한국학술진흥재단 전산실

관심분야: 디지털 도서관, 데이터베이스, 멀티미디어


**오 해 석**

1975년 서울대학교 응용수학과(이학사)

1981년 서울대학교 계산통계학과(이학석사)

1989년 서울대학교 계산통계학과(이학박사)

1982년~현재 숭실대학교 정보과학대학 교수

1976년~1982년 태평양화학(주), (주)삼호 전산실

1990년~1991년 일본 동경대학교 객원교수

1997년~현재 숭실대학교 부총장

관심분야: 멀티미디어, 데이터베이스, 영상처리