기하학적 해싱 기법을 이용한 음악 검색

정효숙†· 박성빈††

요 약

본 논문에서는 음악 데이터베이스의 멜로디와 사용자가 기술한 멜로디의 기하학적 구조를 비교하는 음악 검색 시스템을 제안하고 있다. 시스템은 멜로디의 구조적이고 상황적인 특징들을 분석하여 쿼리 멜로디와 데이터베이스의 멜로디가 일치성을 찾고자 한다. 검색 방법은 사전 처리 단계와 인식 단계로 이루어진 기하하적 해싱 알고리즘에 기반을 두고 있다. 사전 처리 단계 동안 구조적 특징을 찾기 위해서 음악의 멜로디를 여러 개의 프래그먼트(fragment)들로 분할하고 그 프래그먼트의 각 음의 높이 및 길이를 분석한다. 상황적 특징을 찾기 위해서 각 프래그먼트의 중심 화음을 찾는다. 인식 단계 동안 사용자가 입력한 쿼리 멜로디를 여러 개의 프래그먼트들로 분할하고 구조적이고 상황적 특성이 유사한 모든 프래그먼트들을 데이터베이스에서 검색한다. 투표는 각 프래그먼트에 대해 이루어지고 총 득표수가 최대인 음악이 쿼리 멜로디와 일치하는 멜로디를 갖는 음악이 된다. 이러한 접근 방법을 이용하여, 음악 데이터베이스에서 유사한 멜로디를 빠르게 찾을 수 있다. 또한 이 방법은 표절 음악을 감지하는데 적용될 수 있다.

키워드 : 컨텐츠 기반 음악 정보 검색, 기하학적 해싱

Music Retrieval Using the Geometric Hashing Technique

Hyosook Jung†· Seongbin Park††

ABSTRACT

In this paper, we present a music retrieval system that compares the geometric structure of a melody specified by a user with those in a music database. The system finds matches between a query melody and melodies in the database by analyzing both structural and contextual features. The retrieval method is based on the geometric hashing algorithm which consists of two steps; the preprocessing step and the recognition step. During the preprocessing step, we divide a melody into several fragments and analyze the pitch and duration of each note of the fragments to find a structural feature. To find a contextual feature, we find a main chord for each fragment. During the recognition step, we divide the query melody specified by a user into several fragments and search through all fragments in the database that are structurally and contextually similar to the melody. A vote is cast for each of the fragments and the music whose total votes are the maximum is the music that contains a matching melody against the query melody. Using our approach, we can find similar melodies in a music database quickly. We can also apply the method to detect plagiarism in music.

Keywords : contents-based music information retrieval, geometric hashing

1. 서 론

컨텐츠 기반 음악 정보 검색(content-based music information retrieval) 분야에서는 사용자가 비록 음악의 일부분만 기억하고 있더라도 원

†정 회 원: 고려대학교 컴퓨터교육학과 박사과정
††종신회원: 고려대학교 컴퓨터교육학과 교수(교신저자)
논문접수: 2005년 7월 21일, 심사완료: 2005년 8월 23일

하는 음악을 찾을 수 있는 다양한 방안을 제안해 왔다. 일반적으로 음악 데이터베이스에서 주어진 쿼리 멜로디(query melody)와 유사한 모든 멜로디를 검색한다. 가장 전통적인 방법은 유사 스트링 매칭 알고리즘(approximate string matching algorithm)을 이용하여 유사도를 측정하는 것이다[1]. 주로 멜로디에 대한 정보를 특별한 문자나 기호의 시퀀스(sequence)로 표현하고 동적 프로그래밍(dynamic programming)이나 n-그램 기법을 이용하여 유사도를 측정한다[2][3]. 이러한 방법들을 이용하는 이유는 사람들이 음악에 대한 정보를 쉽게 잊어버리고, 일정 시간이 지난 후에 음악의 멜로디를 재생하는 능력이 현저하게 떨어지기 때문이다. 따라서 사람들에게 강한 인상을 남기는 음악적 특징을 추출하여 음악 검색에 이용하는 다양한 방법이 시도되고 있다. 특히, 멜로디 곡선(melodic contour)은 멜로디를 기억하는 중요한 특징으로 알려져 있어 멜로디의 높낮이를 표현하는데 활용되고 있다[2][4].

본 논문에서는 멜로디를 구성하고 있는 음들의 높이와 길이 정보를 이용하여 멜로디의 구조적이고 상황적 특징을 추출한 후, 이러한 특징을 기반으로 쿼리 멜로디와 일치하는 멜로디를 데이터베이스에서 검색하기 위해 기하학적 해싱 알고리즘(geometric hashing algorithm)을 적용하고자 한다. 즉, MIDI 포맷을 기준으로 표시된 음의 높이(pitch)와 음의 길이(duration) 데이터를 이용하여 멜로디 곡선을 2차원 좌표에 표현한다. 이러한 기하학적 구조를 통해 멜로디의 구조적 특징을 추출하고, 쿼리 멜로디가 입력되는 순간, 그 멜로디에 내포된 중심 화음을 분석하여 멜로디의 상황적 특징을 추출한다. 이러한 두 가지 특징은 기하학적 해싱 알고리즘을 적용할 때 필요한 불변 특성(invariant property)을 얻는데 사용된다.

기하학적 해싱 알고리즘을 이용한 음악 검색 방법은 멜로디를 사전 처리(preprocessing)하는 과정에서 멜로디의 구조적 특징과 상황적 특징을 추출하여 해시 테이블(hash table)의 인덱스를 계산하여 먼저 해시 테이블을 구성한다. 인식(recognition)과정에서 사용자가 쿼리 멜로디를 입력하면 이를 사전 처리하여 쿼리 멜로디의 인덱스를 계산한다. 그리고 쿼리 멜로디의 인덱스와 동일한 인덱스를 갖는 멜로디들을 해시 테이블에서 먼저 검색한 후, 데이터베이스에 저장된 멜로디 곡선의 길이를 비교하여 그 유사성을 판단한다. 본 논문에서 구현한 시스템을 이용하여 음악 컨텐츠를 사전 처리하고 데이터베이스에 저장한 후, 음악 검색을 실시하여 기하학적 해싱 방법이 어떻게 쿼리 멜로디와 일치하는 멜로디를 찾아내는가 뿐만 아니라, 원곡을 변형한 표절 음악을 어떻게 인식해 내는지도 살펴볼 것이다.

2. 관련 연구

2.1. 음악 정보 검색 방법

컨텐츠 기반 음악 검색 방법으로서 [1]에서는 반복되는 멜로디 패턴을 검색하기 위해서 스트링 매칭 기법을 사용한다. 멜로디를 스트링으로 표현한 후, 상관성 행렬(correlative matrix)에 서브스트링 매칭의 결과를 저장하면서 모든 가능한 패턴들을 검색하도록 한다. [2]에서는 MIDI 포맷을 기반으로 음악 정보를 검색하는 아키텍처를 제안하였다. 먼저 음악에서 멜로디를 추출한 후, 멜로디 매칭을 위해 멜로디 정보를 “up", "down", "same"과 같은 스트링으로 표현한다. 그리고 멜로디의 유사성을 측정하기 위해서 동적 프로그래밍과 n-그램 매칭 기법을 사용한다. [3]에서는 주어진 쿼리와 가장 유사한 멜로디를 찾기 위해서 음의 높이와 길이 정보를 이용한다. 변환기를 이용하여 사람의 허밍을 음들로 바꾼 후, 쿼리 멜로디와 음악 데이터베이스에 저장된 멜로디의 각 음들 간의 높이와 길이의 차이를 계산한 후, 동적 프로그래밍을 이용하여 부분적으로 일치하는 모든 멜로디를 검색하여 계산한 결과를 저장한다. [4]에서는 멜로디를 <T P B> 세 가지 특징으로 표현하는데, T는 음의 길이, P는 음의 높이, B는 첫 번째 음으로부터 떨어진 만큼을 비트 수를 의미한다. 특히, P는 0, +, -, ++, --, +++ 등과 같은 스트링으로 표현한다.

이러한 컨텐츠 기반 음악 검색은 주로 텍스트 파일을 검색하는 것과 유사한 방법으로 멜로디를

검색하고 있으나, 본 논문에서는 멜로디의 흐름을 선으로 표현하여 음들이 연결된 기하학적 구조로 표현한 후, 이전 연구에서는 사용된 적이 없는 기하학적 해싱 알고리즘을 이용하여 선분 사이의 각도나 선분의 길이 등 기하학적 특징이 유사한 멜로디를 검색 하고자 한다. 또한 멜로디의 각 음들이 조화롭게 연결되어 형성된 화음을 분석하여 멜로디의 상황적 특성으로 반영하고자 한다. 특히 음악 컨텐츠를 사전 처리한 후, 해시 테이블의 인덱스가 일치하는 멜로디 중에서 멜로디의 일치성을 비교하기 때문에 전체 멜로디를 모두 비교하면서 검색하는 것 보다 빠르게 검색할 수 있을 것이다.

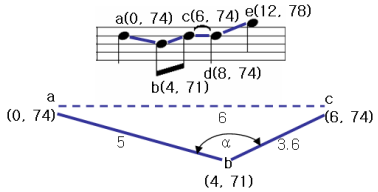
2.2. 기하학적 해싱

기하학적 해싱은 객체를 인식하기 위해 컴퓨터 비전(computer vision)분야에서 개발된 기술이다[5]. 사용자가 찾길 원하는 객체를 효과적으로 인식하기 위한 연구에서 기하학적 해싱 알고리즘이 널리 사용되어 왔다[6][7][8]. 이 알고리즘은 사전 처리 단계에서 객체의 기하학적 정보를 추출하고 해시 함수를 이용하여 이를 인덱싱(indexing)한 후, 해시 테이블에 저장한다. 이러한 기하학적 정보는 객체의 불변 특성이 되고 해시 테이블에 접근하고자할 때 계산된다. 인식단계에서는 찾고자하는 객체와 동일한 인덱싱 정보를 갖는 후보 객체들을 해시 테이블에서 검색한 후, 기하학적 특성이 일치하는지를 비교 검색하게 된다. 비록 두 번의 검색 과정을 수행해야 하지만, 모든 객체를 검색하는 것이 아니라 인덱싱 정보가 일치하는 후보 객체들을 먼저 검색한 후, 실제 일치하는 객체를 찾기 때문에 보다 효율적인 검색을 가능하게 한다[5]. 기하학적 해싱은 단백질 구조를 비교하는 문제와 같은 계산 분자 생물학의 분야에서 많이 이용되고 있다. [7]은 단백질의 3차원 구조를 비교하기 위해서 기하학적 해싱 기법을 사용하고 있으며, [8]은 알파 헬릭스(α-helix)와 베타 스트랜드(β-strand)를 기반으로 단백지의 2차원 구조가 이루는 각의 특성을 이용해 단백질 구조를 비교하는 알고리즘을 개발하였다. 단백질 정보가 저장되는 해시 테이블을 인덱싱하기 위해서 세 개의 조각(알파 헬릭스나 베타 스트랜드)이 이루는 세 개의 코사인 값과 그 조각의 유형을 불변 특성으로 이용한다.

3. 음악 데이터 모델링

3.1. 음악의 구조적 특성

멜로디는 음악의 기본 요소이며 멜로디의 흐름은 곡선으로 표현될 수 있다. 예를 들어, 초등학교의 음악 수업에서는 어린이들이 멜로디의 흐름을 선으로 표현하여 멜로디를 쉽게 인식할 수 있도록 멜로디 곡선을 그리는 활동이 이루어진다[9]. 또한 멜로디 곡선은 멜로디의 유사성을 측정하기 위한 중요한 특징으로도 이용된다[4]. 본 논문에서는 멜로디의 각 음을 2차원 좌표로 표현한다. x축은 16분 음표를 기준으로 음악이 시작되는 시점으로부터 흘러간 시간이 되고(16분 음표=1), y축은 MIDI 숫자로 표현한 음높이가 된다(C4=60). 특히, 세 개의 음은 하나의 삼각형을 만들 수 있기 때문에, 멜로디 구조를 비교하기 위해서 두 음을 연결한 선의 길이와 선과 선이 이루는 각의 코사인 값을 계산한다. 이 값들은 기하학적 해싱을 이용하기 위한 불변 특성으로 이용된다. [그림 1]은 한 마디 안에 존재하는 음들을 2차원 좌표로 표현한 후, 각각 이웃한 음을 연결한 것이다. 이 중에서 3개의 음(a, b, c)을 연결하였을 때, 각각의 좌표와 선분의 길이를 구할 수 있으며, 선과 선이 이루는 각(α)의 코사인 값을 구할 수 있다.

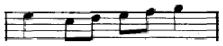


[그림 1] 2차원 좌표로 각 음을 표현

3.2. 음악의 상황적 특성

멜로디의 음들은 조화롭게 연결되어 있기 때문에 멜로디에는 화음들이 존재한다. 화음은 화성

의 기본 요소이며, 화음이 특정 규칙에 따라 연결됨으로써 화성을 이루게 된다. 화음 중에서 주요 3화음은 가장 기본적인 화음으로서 이용되고 있다[10]. 사람들은 임의의 음들을 기억하기 보다는 조화롭게 연결된 음들을 기억하기 때문에 화음은 쿼리 멜로디로 입력되는 순간에 연결된 음들이 갖고 있는 고유한 특성이므로 상황적 특성이라 볼 수 있다. 주요 화음을 분석하는 방법은 음악이 장조인지 단조인지 확인한 후, 각 음의 화음을 표시하고, 음들이 가장 많이 공유하는 화음을 주요 화음으로 결정한다.



화음 \ 음이름	E	C	D	E	F	G
으뜸화음(CEG)	O	O		O		O
버금딸림화음(FAC)		O			O	
딸림화음(GBD)			O			O

[그림 2] 멜로디의 중심 화음 분석

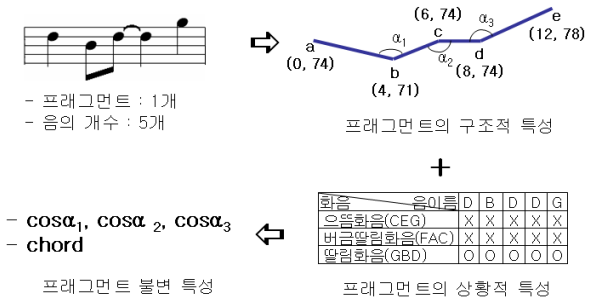
[그림 2]는 다장조 곡에서 특정 마디 안에 존재하는 음들의 화음을 분석한 것이다. 주요 3화음 중에서 6개의 음들이 각각 가질 수 있는 화음을 표시하였을 때, 으뜸화음을 가장 많이 공유하고 있음을 알 수 있다. 따라서 [그림 2]의 멜로디의 주요 화음은 으뜸화음이 된다.

4. 기하학적 해싱 알고리즘을 이용한 음악 검색

4.1. 사전 처리

4.1.1. 불변 특성 추출하기

불변 특성을 정의하기 위해서 멜로디의 두 음을 연결한 선분들이 이루는 각을 이용한다. 멜로디를 검색할 때, 사람들의 기억력에 의존할 수밖에 없다. 일반적으로 단기 기억의 용량은 7±2이므로[11], 본 논문에서는 멜로디 곡선을 구성하는 음의 최소 개수를 5개로 하며, 이를 프래그먼트(fragment)라 부르기로 한다. 5개의 음이 모인 하나의 프래그먼트는 4개의 선분과 세 개의 각을 이루게 된다. 어떤 음악의 멜로디에서 불변 특성을 추출하려면, 멜로디의 구성음을 5개씩 중복하여 분할시킴으로써 여러 개의 프래그먼트를 만들게 된다. 각 프래그먼트의 4개의 선분의 길이와 세 각의 코사인 값을 계산한다. 또한 프래그먼트의 화음을 분석하여 중심 화음을 찾아 불변 특성으로 활용한다. [그림 3]은 임의의 프래그먼트의 불변 특성을 추출하는 과정을 나타내고 있다. 프래그먼트의 음들을 기하학적 구조로 표현하여 α_1, α_2, α_3의 코사인 값을 계산할 수 있으며, 각 음들이 가장 많이 공유하는 주요 화음(딸림화음)을 알 수 있다.



[그림 3] 각 프래그먼트의 불변 특성 추출

4.1.2. 해시 테이블 만들기

프래그먼트의 세 각의 코사인 값과 중심 화음은 해시 테이블의 슬롯 인덱스를 생성할 때 이용된다. 각 슬롯은 각 프래그먼트에 대한 여러 가지 정보(프래그먼트의 ID, 세 각의 코사인 값, 4개의 선분의 길이, 중심 화음 등)를 저장한 데이터베이스의 레코드 하나를 참조하고 있다.

[표 1] 해시 테이블의 인덱스 생성 알고리즘

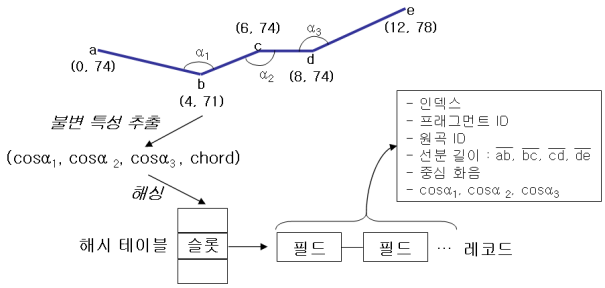
```
Hash(T, N)
for(int i = 0 ; i < N; i++)
  index = h1(cosαi,1, cosαi,2, cosαi,3, chordi)
    if T[index] == NIL
       then T[index] = index
       return index
    else
       A-Hash(index, cosαi,1, cosαi,2, cosαi,3, chordi)
```

```
A-Hash(index, cosαi,1, cosαi,2, cosαi,3, chordi)
index = index + h2(cosαi,1, cosαi,2, cosαi,3, chordi)
    if T[index] = = NIL
       then T[index] = index
       return index
    else
A-Hash(index, cosαi,1, cosαi,2, cosαi,3, chordi)
```

본 논문에서는 해시 테이블을 만들기 위해 더블 해싱[12]을 활용하였으며, 해시 테이블의 인덱

스를 생성하는 알고리즘은 [표 1]과 같다. T는 해시 테이블을 의미하고, N은 프래그먼트의 개수를 의미한다. 해시 테이블의 인덱스는 쿼리 멜로디의 인덱스와 동일한 인덱스를 갖는 모든 프래그먼트를 검색하는데 사용된다. 또한 인덱스는 데이터베이스에 저장된 각 프래그먼트의 정보를 선택할 때 이용된다.

[그림 4]는 각 프래그먼트의 불변 특성을 추출하여 해시 테이블을 만드는 과정을 나타내고 있다. 불변 특성의 값에 해시 함수를 적용하여 인덱스 값을 얻을 수 있으며, 이렇게 생성된 인덱스 값은 해시 테이블의 슬롯 인덱스가 된다. 슬롯 인덱스는 쿼리 멜로디와 동일한 인덱스를 찾거나 해당 프래그먼트의 정보가 저장된 데이터베이스를 참조할 때 사용된다.



[그림 4] 해시 테이블 생성 과정

4.2. 인식

4.2.1. 검색 과정

[그림 5]는 유사한 멜로디를 인식하는 과정이며, 각각에 대한 자세한 설명은 다음과 같다.

① 멜로디의 구성음이 5개보다 적으면 인덱스를 생성하기 위한 불변 특성을 추출할 수 없으므로 사용자의 쿼리 멜로디가 최소 5개 이상의 음들로 구성되어 있는지 확인한다.

① 멜로디의 구성음이 5개보다 적으면 인덱스를 생성하기 위한 불변 특성을 추출할 수 없으므로 사용자의 쿼리 멜로디가 최소 5개 이상의 음들로 구성되어 있는지 확인한다.

② 쿼리 멜로디를 기하학적인 구조로 모델링한다. 멜로디의 각 음의 길이와 높이를 이용하여 2차원 좌표로 표현한다. 예를 들어, MIDI 숫자가 “69”이고 가장 첫 번째 음이라면 (0, 69)이다.

③ 쿼리 멜로디의 음들을 두 개씩 연결하여 생성된 선분의 길이를 계산한다. 각 선분의 길이는 해시 테이블에서 같은 인덱스를 갖는 프래그먼트들을 쿼리 멜로디와 비교할 때 사용된다.

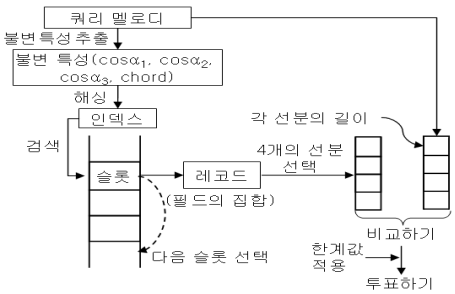
④ 인접한 두 선분사이에 끼인 각의 코사인 값을 계산한 후, 쿼리 멜로디를 5개의 음으로 구성된 프래그먼트 단위로 중첩 방식으로 분할한다. 예를 들어, 멜로디의 구성음이 7개라면 3개의 프래그먼트를 만들게 된다.

⑤ 각 프래그먼트의 중심 화음을 분석한 후, 각 프래그먼트의 세 각의 코사인 값과 중심 화음을 해시 함수에 적용시켜 인덱스를 생성한다. 그리고 해시 테이블에서 같은 인덱스를 갖는 모든 프래그먼트를 검색한다.

⑥ 같은 인덱스를 갖는 프래그먼트들과 쿼리 멜로디가 각자 갖고 있는 4개의 선분들 간의 길이의 차이를 계산한다.

⑦ 각 선분의 길이의 차이가 한계값을 넘는지 확인한 후, 한계값 이하일 때, 해당 프래그먼트에 투표한다.

⑧ 각 음악의 총 득표수를 내림차순으로 정렬하여 결과를 제시한다. 득표수가 상대적으로 많을수록 쿼리 멜로디와 유사한 멜로디를 갖고 있음을 추측할 수 있다.



[그림 5] 쿼리 멜로디와 유사한 멜로디 검색하기

4.2.2. 선분 길이 차이에 대한 한계값 설정

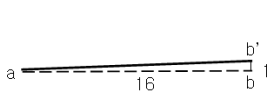
쿼리 멜로디와 인덱스가 일치하는 프래그먼트가 쿼리 멜로디의 프래그먼트와 유사한지 측정하기 위해서 두 프래그먼트간의 선분의 길이의 차이를 비교한다. 이때, 멜로디의 음들이 음의 길이와 높이를 이용하여 좌표에 표현되기 때문에,

두 음 사이의 높이의 차이와 길이의 차이를 고려하여 다음과 같은 기준으로 적용되는 한계값을 설정하였다.

① 두 음 간의 최대 길이의 차이는 박자와 관계가 있다. 본 논문에서는 16분 음표를 기준으로 음의 길이를 측정하였다. 따라서 16분 음표는 '1', 4분 음표는 '4' 등으로 표시된다. 예를 들어, 4/4 박자라면 하나의 음이 가질 수 있는 최대 음의 길이는 4박자(온음표일 때)이므로 '16'이 된다. 따라서 두 음 길이의 최대 차이는 '16'이 된다.

② 두 음간의 최소 높이의 차이는 반음 간격이며, 각 음의 높이는 MIDI 숫자로 표현되므로, MIDI에서 반음 간격은 '1'이 된다. 예를 들어, 'B4(시)'와 'C5(도)'는 반음 간격이며, 'B4(시)'는 MIDI 숫자가 '71'이고, 'C5(도)'는 '72'이므로 두 음간의 높이 차이는 '1'로 표현된다.

③ [그림 6]은 4/4박자 곡일 때, 적용되는 한계값을 계산하는 과정을 설명하고 있다. 최대 음 길이의 차이는 '16'이 되고, 최소 음 높이의 차이는 '1'이 되므로 한계값은 약 '0.03'이 된다.

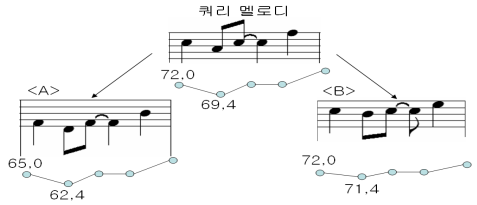


$$L(a,b') = \sqrt{16^2 + 1^2}$$
$$L(a,b) = 16$$
$$d = L(a,b') - L(a,b)$$
$$\therefore threshold = d = \sqrt{16^2 + 1^2} - 16 \approx 0.03$$

[그림 6] 4/4 박자에서 한계값 설정

4.2.3. 쿼리 멜로디와 유사한 멜로디 판단되는 경우

기하학적 해싱 알고리즘의 적용할 경우, 실제 검색하고자 하는 멜로디가 아닌 멜로디 곡선 형태가 유사한 멜로디가 검색될 수 있다. 본 논문에서는 멜로디 곡선을 각 음의 높이와 길이로 표현하였기 때문에 검색된 프래그먼트를 구성하는 음들의 높이와 길이의 차이가 크지 않을 경우 유사할 경우 유사한 멜로디로 판단하게 되는 것이다. 즉, 2차원 좌표에 표현한 멜로디의 기하학적 형태가 유사할 경우, 쿼리 멜로디와 정확하게 일치하지 않더라도 '두 프래그먼트가 유사하다'고 판단하게 된다. [그림 7]은 쿼리 멜로디가 주어졌을 때, A 멜로디와 B 멜로디가 유사한가를 멜로디 곡선을 그려 비교하고 있다. A는 쿼리 멜로디와 음 이름이 정확하게 일치하지 않지만 음 높이의 변화가 동일하고 음 길이도 동일하기 때문에 유사한 멜로디로 검색된다. 예를 들어, 앞의 두 음이 그리는 선분의 길이를 비교해보면 '5'로써 같다. 반면, B는 쿼리 멜로디와 음 길이는 동일하나, 음 높이의 변화가 쿼리 멜로디보다 완만하여 유사한 멜로디로 검색되지 않는다. 앞의 예와 마찬가지로 앞의 두 음이 그리는 선분의 길이를 비교해보면, 약 '0.53'의 차이가 나기 때문에 실험에 적용할 한계값(0.03)을 넘게 된다.



[그림 7] 쿼리 멜로디와 유사한 멜로디 찾기

5. 시스템 구조

본 논문에서 제안한 시스템은 자바 서블릿과 JSP를 이용하여 구현되었으며, 톰캣 서버와 MySQL 데이터베이스 시스템을 이용하였다. 사용자는 웹 서버로 쿼리 멜로디를 전송하고, 웹 서버는 6개의 모듈을 이용하여 사용자의 쿼리를 처리한다. 그리고 음악 데이터베이스에 저장된 멜로디 중에서 유사한 멜로디를 찾은 후, 그 결과를 사용자에게 전송하게 된다. [그림 8]은 시스템의 구조를 나타내고 있으며, 각 모듈에 대해 살펴보면 다음과 같다.

① 사전 처리 모듈 : 멜로디를 기하학적 구조로 모델링한다. 각 음을 2차원 좌표로 표현하고, 두 음을 연결한 선분의 길이를 구하며, 두 선분 사이에 끼인 각의 코사인 값을 계산한다. 또한 중심 화음을 으뜸화음은 '1', 버금딸림화음은 '2', 딸림화음은 '3'으로 표시한다.

② 쿼리 모듈 : 사용자가 전송한 쿼리 멜로디를 음악 멜로디의 사전 처리 작업과 같은 방식으로 사전 처리한다.

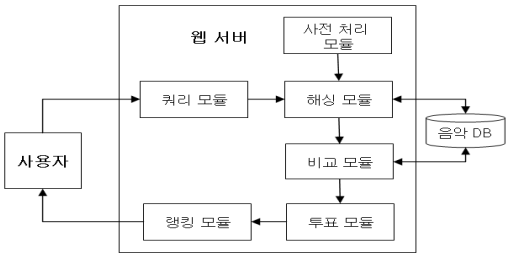
③ 해싱 모듈 : 해시 테이블의 슬롯에 저장될 인덱스를 생성하는 모듈이다. 음악 멜로디나 쿼리 멜로디를 여러 개의 프래그먼트로 분할한 후, 사전 처리 모듈과 쿼리 모듈에서 생성된 세 각의

코사인 값과 중심 화음을 해시 함수에 적용하여 인덱스를 구하고 해시 테이블을 생성한다.

④ 비교 모듈 : 데이터베이스로에서 쿼리 멜로디와 유사한 멜로디를 검색한다. 쿼리 멜로디 프래그먼트들의 인덱스와 같은 인덱스를 갖는 모든 프래그먼트를 데이터베이스에서 검색한다. 쿼리 멜로디 프래그먼트의 4개의 선분과 검색된 프래그먼트의 4개의 선분의 길이를 서로 비교한다.

⑤ 투표 모듈 : 4쌍의 선분의 길이의 차이가 한계값을 넘는지를 확인한 후, 한계값을 넘지 않는 프래그먼트가 존재한다면, 그 프래그먼트는 쿼리 멜로디와 유사한 멜로디를 갖고 있다고 판단할 수 있으므로, 그 프래그먼트에 투표한다.

⑥ 랭킹 모듈 : 각 음악의 프래그먼트들의 투표 상황을 바탕으로 총 득표수를 계산한 후, 내림차순으로 정렬하여 그 결과를 제시한다. 상위에 기록될수록 쿼리 멜로디와 일치하는 멜로디를 포함하고 있는 음악이라고 볼 수 있다.



[그림 8] 시스템의 구조

6. 실험 결과

본 논문에서 제안한 시스템을 실험하기 위해서 UCI Machine Learning Repository Content Summary[13]에서 제공하는 바흐의 코랄 데이터를 이용하였다. 전체 100곡 중에서 장조 19곡과 단조 12곡을 선택하여 사전 처리하였다. 31곡의 전체 프래그먼트 개수는 1258개이다. 실험은 4가지 유형으로 이루어졌다. [유형 1]은 특정 음악 내에서 길이가 다른 멜로디들을 쿼리 멜로디로 입력하여 그 음악에서 원래의 멜로디를 찾아내는지 확인한다. [유형 2]는 길이가 다른 임의의 쿼리 멜로디를 입력하여 전체 음악에서 그 쿼리 멜로디를 갖고 있는 음악을 검색한다. [유형 3]은 임의의 음악의 전체 멜로디를 쿼리 멜로디로 입력하여 실제로 원곡을 검색하는지 그 결과를 확인한다. [유형 4]는 원곡의 멜로디를 변형하여 새로운 멜로디를 구성한 후, 이를 쿼리 멜로디로 입력하여 원곡을 찾는가와 어느 정도 유사한가를 살펴본다.

6.1. 유형별 실험 결과

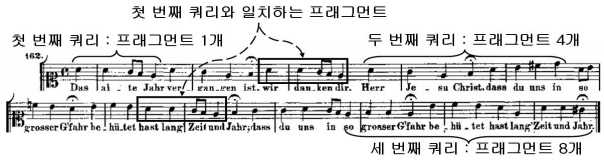
① 유형 1

No. 162에서 다양한 길이(1개, 4개, 8개)의 프래그먼트를 추출한 후, 쿼리 멜로디로 입력하여 검색한다. [표 2]는 길이를 각각 다르게 구성한 프래그먼트를 검색한 후, 일치하는 프래그먼트를 포함한 음악에 투표한 결과를 나타낸다.

[표 2] 각 음악의 총 득표수

쿼리 멜로디 (프래그먼트 #)	음악 ID	검색된 프래그먼트 ID
첫 번째(1개)	162	1, 10, 31
두 번째(4개)	162	6, 15, 16, 17, 18
세 번째(8개)	162	42, 43, 44, 45, 46, 47, 48, 49

검색된 프래그먼트 ID를 살펴보면, 실제 검색하고자 했던 멜로디가 존재하는 프래그먼트를 찾아낼 뿐만 아니라, 위치는 다르지만 동일한 멜로디를 갖고 있는 프래그먼트도 검색함을 확인할 수 있다. 예를 들어, 첫 번째 쿼리 멜로디는 첫 번째 프래그먼트를 검색할 뿐만 아니라, 10번째와 31번째 프래그먼트도 일치하는 프래그먼트라고 검색한다. 실제 악보에서 10번째와 31번째 프래그먼트의 음들과 첫 번째 프래그먼트의 음들을 비교해보면, 음 높이와 음길이가 일치하는 멜로디임을 확인할 수 있다. [그림 9]은 No. 162에서 쿼리 멜로디로 선택된 프래그먼트를 표시하고 있으며, 특히, 첫 번째 쿼리 멜로디와 일치하는 다른 프래그먼트들을 사각형으로 표시하였다.



[그림 9] No. 162의 쿼리 멜로디 검색

② 유형 2

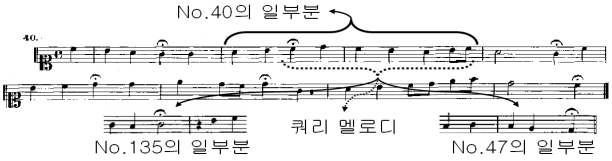
No. 40에서 임의의 7개의 음을 쿼리 멜로디로 입력한 후, 전체 음악에서 유사한 프래그먼트를

비교 검색하고, 각 음악의 총 득표수를 계산한다.

[표 3] No. 40의 쿼리 멜로디의 검색 결과

프래그먼트 개수	음악 ID	검색된 프래그먼트 ID	총 득표수
3개	40	6, 7, 8, 9, 10	5
	47	46	1
	135	39	1

[그림 10]는 No. 40의 5개 프래그먼트가 어떤 부분인지 검색한 결과를 실제 악보에 표시한 것이다. No. 40에서 실제 쿼리 멜로디보다 넓은 범위의 멜로디를 검색하였으며, No. 135와 No. 47에서 검색된 프래그먼트의 멜로디를 살펴보면, 실제 쿼리 멜로디와 유사함을 확인할 수 있다.



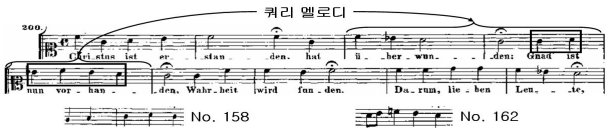
[그림 10] No. 40 에서 쿼리 멜로디 검색 결과

또 다른 음악인 No. 200에서 임의의 멜로디를 추출하여 전체 음악을 대상을 검색을 실시하였다. 쿼리 멜로디의 음의 개수는 10개이며, 총 6개의 프래그먼트를 구성하게 된다. [표 4]는 쿼리 멜로디를 검색하였을 때, 검색된 프래그먼트 ID와 해당 음악 ID를 표시한 것이다.

[표 4] 쿼리 멜로디의 검색 결과

프래그먼트 개수	음악 ID	검색된 프래그먼트 ID	총 득표수
6개	200	8, 9, 10, 11, 12, 13	6
	158	12, 13	2
	162	23	1

[그림 11]은 No. 200의 6개 프래그먼트가 어떤 부분인지 검색한 결과를 실제 악보에 표시한 것이다. No. 200의 8, 9, 10, 11, 12, 13 번째 프래그먼트에서 검색하고자 했던 멜로디를 찾아내었다. 사각형 표시 안의 음들은 다른 음악에서 검색된 프래그먼트의 멜로디와 유사한 부분으로써, No. 158의 프래그먼트는 음 높이의 변화가 같고, 음 길이는 동일하다. No. 162는 음 높이가 동일하나, 음 길이는 앞의 두 음이 다르다.



[그림 11] No. 200 에서 쿼리 멜로디 검색 결과

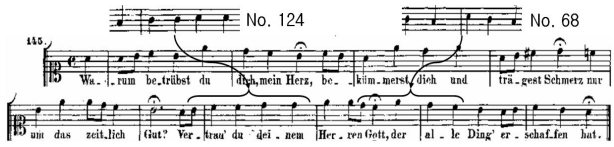
③ 유형 3

특정 음악의 전체 멜로디를 하나의 쿼리 멜로디로 입력한 후, 실제 원곡의 모든 프래그먼트를 검색하는지와 다른 음악에서 유사한 멜로디를 검색해 내는지를 확인한다. No. 145는 모두 41개의 프래그먼트로 구성되어 있으며, [표 5]는 No. 145의 전체 멜로디를 입력한 후, 각 음악의 총 득표수를 계산한 결과이다. [표 5]는 검색을 통해 득표한 음악을 나타내며, No. 145의 총 득표수는 41로서 모든 프래그먼트를 검색함을 하였음을 확인할 수 있다.

[표 5] 음악의 총 득표수

음악ID	총득표수	음악ID	총득표수	음악ID	총득표수
68	1	124	1	145	41

[그림 12]은 No. 145 이외의 다른 음악에서 검색된 프래그먼트가 어떤 부분이며, No. 145의 어떤 부분과 유사한지를 No. 145의 악보에 표시한 것이다. No. 68에서 발견된 프래그먼트(32번째)는 음 높이의 차이는 쿼리 멜로디와 동일하고 음 길이는 첫 번째 하나가 다르다. No. 124(34번째)에서 발견된 프래그먼트는 음 높이 변화가 쿼리 멜로디와 같고 음 길이는 첫 번째 하나가 다르다.



[그림 12] No. 145 에서 쿼리 멜로디 검색 결과

④ 유형 4

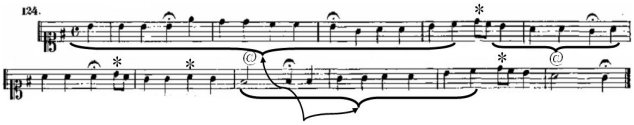
본 시스템이 표절한 음악을 찾아낼 수 있는지 실험하기 위해서 특정 음악을 약간 변형하여 검색을 수행하였다. 본 실험에서는 No. 124의 멜로디에서 2분 음표를 두 개의 4분 음표로 분할하고, 멜로디의 화음을 분석한 후, 화음 밖의 음을 제거하여 변형된 멜로디를 구성하였다. 또한 표절한 멜로디를 광범위하게 검색하기 위해서 선분을 비교하는 한계값을 '0.03'에서 '0.5'로 높여서 적용하였다. 원곡을 변형할 때, 화음 밖의 음을 제거하였으므로, 화음 밖의 음이 없고 주요 3화음의 구성음만 존재할 때 이웃한 음들 간의 높이의 차이는 평균 '4'이다. 유형 1, 2, 3에서는 두 음의 높이의 차이가 가장 적은 '1'로 정하여 계산

하였으나, 표절한 음악을 검색할 때는 '4'로 높여 한계값을 계산하였다. [표 6]은 No. 124에서 쿼리 멜로디와 득표한 프래그먼트를 표시한 것이다.

[표 6] No. 124에서 쿼리 멜로디 검색 결과

ID	Vote	ID	Vote	ID	Vote	ID	Vote	ID	Vote
71	T	80	T	89	T	98	F	107	F
72	T	81	T	90	T	99	F	108	F
73	T	82	F	91	T	100	F	109	F
74	T	83	F	92	T	101	F	110	F
75	T	84	F	93	F	102	T	111	F
76	T	85	F	94	F	103	T		
77	T	86	F	95	F	104	T		
78	T	87	T	96	F	105	T		
79	T	88	T	97	F	106	T		

[그림 13]은 No. 124에서 변형된 멜로디와 유사한 부분을 악보에 표시한 것이다. 악보에서 '*'로 표시된 음은 변형된 멜로디에서 제거된 음이고, '@'은 4분 음표로 분할한 음을 나타낸다.



변형된 멜로디와 유사한 부분

[그림 13] No. 124의 변형 멜로디와 유사한 부분

6.2. 실험 결과 분석

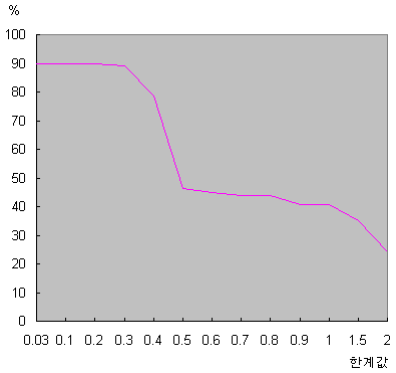
본 논문에서 제안한 음악 검색 방법이 한계값에 따라서 데이터를 어떻게 검색해 내는지 분석하기 위해서 다음과 같은 실험을 실시하였다.

첫째, 전체 31곡의 음악에서 첫 번째 프래그먼트를 쿼리 멜로디로 추출하였다.

둘째, 한계값을 점차 증가시키면서 각 한계값마다 31개의 쿼리를 입력하였을 때 시스템이 검색해낸 데이터 중에서 실제 찾고자 했던 멜로디와 정확하게 일치하는 유효 데이터를 찾는다.

셋째, 검색된 데이터 개수 대 유효 데이터 개수의 비율을 계산한다.

각 음악마다 추출한 쿼리 멜로디를 입력하였을 때, 실제 찾고자 했던 프래그먼트를 모두 찾아냈지만, 한계값이 증가함에 따라 실제 찾고자했던 프래그먼트뿐만 아니라, 이와 유사한 프래그먼트도 검색되기 때문에 유효 데이터가 검색될 확률은 점점 떨어지게 된다. [그림 14]은 한계값의 변화에 따른 유효 데이터가 검색되는 비율을 제시하고 있으며, 특히, 한계값이 '0.5'일 때 50%이하로 급격하게 감소하고 있음을 알 수 있다.



[그림 14] 한계값 변화에 따른 유효 데이터의 검색 비율

7. 결론

자신이 들었던 멜로디를 기억하여 정확하게 재생해 것은 어려운 일이기 때문에 사람들이 멜로디의 일부분만 알고 있더라도 원하는 음악을 검색하고자 할 때, 컨텐츠 기반 음악 검색 시스템은 유용하다. 본 논문에서는 음악의 흐름을 이해하는데 도움을 주는 멜로디 곡선을 2차원 좌표에 표현한 후, 기하학적 해싱 알고리즘을 이용하여 선분 사이의 각도나 선분의 길이 등 멜로디의 구조적 특징이 유사한 멜로디를 검색 하였고, 멜로디의 화음을 분석하여 멜로디의 상황적 특성으로 이용하였다. 이러한 특성을 바탕으로 쿼리 멜로디와 일치하는 멜로디를 포함한 음악을 검색할 뿐만 아니라, 표절한 음악을 검색하는데도 활용하였다. 특히, 사전 처리 과정에서 생성한 해시 테이블을 통해 인덱스가 일치하는 멜로디 중에서 멜로디의 일치성을 비교함으로써 전체 멜로디를 모두 검색하는 것 보다 빠르게 검색하고자 하였다. 현재 더욱 다양한 음악 컨텐츠를 확보하고 이를 실험 데이터로 활용함으로써 음악의 유형에 따라 적용될 수 있는 적절한 한계값에 대한 연구 및 해시 테이블을 생성할 때 적용할 알맞은 해시 함수에 대한 연구를 진행하고 있으며, 더 많은 데이터를 수집하여 한계값에 따른 정확률과 재현율을 검증하고자 한다.

참 고 문 헌

[1] Hsu, J. L., Liu, C. C., Chen, A. L. P. : Efficient Repeating Pattern Finding in Music Databases, Proceedings of the seventh international conference on Information and Knowledge Management, (1998) 281-288.

[2] Alexandral, L., Bogerd, U., Zobel, J. : An Architecture of Effective Music Information Retrieval, Journal of the American Society for Information Science and Technology, 55(12), (2004) 1053-1057.

[3] Arentz, W. A., Hetland, M. L., Olstad B. : Methods for Retrieving musical information based on rhythm and pitch correlations, CSGSC, (2003).

[4] Kim, Y., Chai, W., Garcia, R. and Vercoe, B. : Analysis of a Contour-based Representation for Melody, Proceedings of International Symposium on Music Information Retrieval, (2000).

[5] Wolfson, H. J.: Geometric Hashing : An Overview, IEEE Computational Science & Engineering, (1997) 10-21.

[6] Clausen, M. and Kurth, F. : A Unified Approach to Content-Based and Fault-Tolerant Music Recognition, IEEE Transactions on multimedia, Vol. 6, No. 5,(2004) 717-731.

[7] Pennec, X., Ayache, N. : An $O(n^2)$ Algorithm for 3D Substructure Matching of Proteins, In Proceedings of the First International Workshop on Shape and Pattern Matching in Computational Biology, June, (1994) 25-40.

[8] Ferrari, C., Guerra, Cl, Zanotti, G. : A grid-aware approach to protein structure comparison, Journal of Parallel and Distributed Computing, 63, (2003) 728-737.

[9] 교육부, 제7차 음악과 교육과정 해설서, 1997

[10] Fred Plotkin : Classical Music 101 - A Complete Guide to Learning and Loving Classical Music, Hyperion, New York, (2002).

[11] Miller, G. A. : The magical number seven plus or minus two : Some limits on our capacity for processing information. Psychological Review, 63 , (1956) 81-97.

[12] Cormen, T. H., Leiserson, C. E., Stein, C. : Infroduction to Algorithms Second Edition, McGraw-Hill Higher Education, (2003).

[13] http://www.ics.uci.edu/~mlearn/MLSummary.html

정 효 숙

1998 서울교육대학교 교육학과 (교육학학사)

2001 서울교육대학교교육대학원 컴퓨터교육과(교육학석사)

2003 ~ 현재 고려대학교 컴퓨터교육학과 박사과정

관심분야: 컴퓨터교육, 적응형 하이퍼미디어

E-Mail: est0718@comedu.korea.ac.kr

박 성 빈

1990 고려대학교 전산과학과 (이학사)

1993 University of Southern California (전산학 석사)

1999 University of Southern California (전산학 박사)

2003 ~ 현재 고려대학교 컴퓨터교육과 조교수

관심분야: 하이퍼텍스트, 컴퓨터교육, 알고리즘, 계산이론

E-Mail: psb@comedu.korea.ac.kr