음악 컨텐츠를 위한 빠른 검색 기법

노승민[O] 황인준
아주대학교 정보통신전문대학원 정보통신공학과
{anycall[O], ehwang}@ajou.ac.kr

A Fast Retrieval Scheme for Music Contents

Seungmin Rho[O] Eenjun Hwang
The Graduate School of Information and Communication, Ajou University

요 약

최근 들어 디지털 음악 컨텐츠의 사용량이 증가하면서 데이터베이스로부터 다양한 포맷의 음악 컨텐츠를 효과적으로 찾을 수 있는 음악 검색 시스템의 필요성이 증가했다. 본 논문에서는 사용자들이 자주 질의하는 멜로디를 이용하여 효율적인 검색을 하기 위한 기법을 제안한다. 사용자의 허밍이나 오선지를 통한 질의로부터 추출된 음높이와 음의 길이를 분석하여 UDR과 LSR 스트링으로 변환하고 자주 검색되는 멜로디의 관리를 통해 검색의 성능을 향상시켰다. 또한 XML을 사용하여 음악 정보를 구조화하는 새로운 데이터 모델을 제안하고 음높이, 음의 길이, 리듬 등의 음악 특징 정보를 이용한 복합 질의를 통하여 제안한 시스템의 성능 평가를 하였다.

1. 서론

웹의 급속한 발전과 함께 음악 컨텐츠의 양이 증가하게 되었고 음악 데이터베이스로부터 특정한 멜로디를 찾아내는 음악 검색 시스템의 필요성이 증가했다.

가장 대표적인 내용 기반 음악 검색 시스템으로는 사용자의 휘파람이나 허밍을 이용한 질의를 사용하는 Query By Humming(QBH) 시스템이 있다. Ghias[1]는 마이크를 통해 입력 받은 사용자의 허밍에서 음높이 변화(pitch contour)를 감지하여 이를 음악 데이터베이스의 컨텐츠와 비교 검색하였다. Wold[2]는 신호 처리를 이용해서 음의 크기(loudness), 밝기(brightness), 높이(pitch), 대역폭(*bandwidth*), 조화도(*harmonicity*)등과 같은 음향 특징을 추출하였다. 또한 이러한 특징들을 벡터로 표현하여 대상 음악의 음향 특징 정보와 질의 음악의 특징 벡터간 거리(vector distance)를 비교하는 방법을 사용하여 음악 컨텐츠를 분류하였다. MELDEX[3]에서는 QBH와 마찬가지로 마이크를 통한 사용자의 허밍으로부터 추출된 멜로디의 음높이 변화나 음의 길이를 이용하여 UDR 스트링 매칭을 하고 동적 프로그래밍(Dynamic Programming) 알고리즘을 이용한 유사 검색 기법을 사용하였다.

본 논문에서는 자주 질의되는 멜로디 패턴의 위치를 FAI에 저장해 두고 사용자의 질의가 들어올 때마다 멜로디 데이터베이스 전체를 검색하는 대신 FAI의 엔트리를 먼저 검색한다. 따라서 사용자가 작성한 질의 멜로디가 FAI 엔트리 안에 있는 엔트리와 같다면 나머지 음악 컨텐츠 모두를 검색할 필요 없이 바로 결과를 도출할 수 있어 검색 속도의 향상을 기대할 수 있다. 또한 다양한 음악 특징의 추출과 XML을 이용한 음악 데이터 모델링을 통하여 확장된 음악 검색 시스템인 MFC(Melody Finder using Climax)를 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 사용된 음향 정보들을 소개한다. 3장에서는 음악 데이터의 저장 구조 설계와 질의 처리 과정 및 FAI에 대해 설명한다. 4장에서는 본 논문에서 구현한 음악 검색 시스템의 구조와 구현 및 실험에 대해서 기술하며 마지막으로 5장에서는 결론과 향후 계획에 대해서 논의한다.

2. 내용 기반 분석 및 검색 방법

대부분의 기존 연구에서는 음향 정보 중 음높이의 변화를 비교하여 이를 다음의 세 가지 타입의 UDR 스트링으로 표현한다. 이전 음표의 음높이와 현재 음표의 음높이를 비교해서 현재 음표의 음높이가 상대적으로 높으면 U(p), 낮으면 D(own), 같으면 R(epeat)로 표기한다. 하지만, 기존의 UDR 스트링은 다음과 같은 문제점을 가진다.

첫째, 전혀 다른 음정을 가지는 음악 파일이나 질의를 동일하게 인식한다는 것이다. 그림 1에서 보는 것과 같이 첫 번째 소절과 두 번째 소절이 확연한 차이를 보임에도 불구하고 UDR 스트링 기법에만 의존할 경우 두 소절 모두 "UDU" 스트링으로 변환 된다. 이러한 문제를 해결하고자 본 논문에서는 기존의 UDR 스트링을 세분화하여 검색의 정확도를 높였다. 기존의 Up, Down, Repeat과 두 번째 소절에서 보여지듯이 음정(interval)이 급격하게 높아지거나(Up a lot), 낮아지거나(Down a lot) 하는 경우를 추가하여 u, U, d, D, r과 같이 세분화하였다. 이때, 대문자는 음정의 변화가 급격한 경우를 나타낸다. 음정의 급격한 변화의 정도는 각 음표의 수치 값들의 변화 정도가 일정 임계치를 넘게 되는 경우나 이전의 음정의 변화폭보다 더 크게 변화되는 경우로 판단할 수 있다. 그림 1의 두 소절의 음정 변화를 스트링으로 표기하면 "udu"와 "UDU"로 나타낼 수 있다.

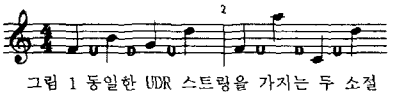

그림 1 동일한 UDR 스트링을 가지는 두 소절

둘째, UDR 스트링 표기법은 음 높이의 변화가 아닌 음표가 가지는 길이 정보와 다른 수식 정보들을 표현할 수 없다

는 것이다. 그림 2의 두 소절을 확장된 UDR 스트링 표기법에 따라 표기하면 둘 다 "udr"로 표현되기 때문에 음의 길이(duration)가 서로 다른 멜로디의 구분이 힘들어 진다. 이와 같은 경우에는 음의 길이 정보를 이용하여 문제를 해결할 수 있다. 음의 길이 정보는 Longer, Shorter, Repeat의 세 가지 타입으로 구분되며 L, S, R의 스트링으로 표현한다. 그림 2의 두 소절의 음의 길이 변화를 스트링으로 표기하면 둘 다 "LSR"로 나타낼 수 있다. 하지만, 그림 2에서 보여지듯이 음정의 변화와 음의 길이를 이용하더라도 두 소절의 정확한 차이를 구분할 수가 없다.

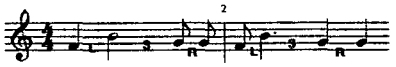

그림 2 동일한 LSR 스트링을 가지는 두 소절

그래서 음표의 길이를 좀 더 확장한 방법으로 아래와 같이 음표의 길이만큼 문자열로 표시한다. 각 음표에 해당하는 문자 "도레미파솔라시"는 "CDEFGAB"로 대응되며 음표의 길이에 대한 가중치를 16분음표(semiquaver), 8분음표(quaver), 4분음표(crotchet), 2분음표(minim), 온음표(semibreve)에 각각 "1", "2", "4", "8", "16"으로 설정한다. 점음표(dot note)의 경우에는 문자열 뒤에 "."을 추가하고 sharp과 flat은 각각 문자열 뒤에 "#", "-"을 추가한다. 그림 2의 8개 음표에 대한 멜로디를 표기하면 아래와 같다.

F4|B8|G2|G2|F2|B4.|G4|G4

위에서 언급한 방식의 스트링은 음악 컨텐츠를 텍스트로 표현한 정보이므로 필요한 정보를 찾기 위해서는 서브스트링 검색(Substring Matching) 기법이 필요하게 된다. 이때 사용자의 질의가 그들의 기억에 의존하고 휘파람이나, 허밍을 이용하여 질의를 생성할 경우 원곡의 스트링과 조금 다를 수 있다. 예를 들어, 위의 스트링을 질의 멜로디라고 가정하고 데이터베이스내에 저장되어 있는 멜로디가 "F4|C4|G4|G4|F2|C4.|G4|G4"과 같다고 할 때 두 멜로디는 정확 매칭 방법으로는 일치하지 않는다.

따라서 본 논문에서는 이러한 사용자의 부정확한 질의 멜로디를 고려하여 정확 매칭(Exact Matching) 기법보다는 유사 매칭(Approximate Matching) 기법을 이용하며 음악 데이터베이스의 컨텐츠 수의 증가로 인한 시스템 성능 저하를 고려하여 정확 매칭 기법을 병행하여 이용한다. 정확 매칭 기법은 *Boyer-Moore* 알고리즘[6]을 사용하였고 유사 매칭 기법은 동적 프로그래밍 방법을 사용하였다.

3. 음악 저장 공간을 위한 스키마 및 인덱싱

3.1 음악 스키마

최근 인터넷에서의 문서 또는 데이터를 표현하기 위한 표준으로 많이 쓰이고 있는 XML을 이용하여 만들어진 MusicXML[4], ScoreML과 같은 포맷은 악보를 XML로 표현한 것으로 음정, 박자, 조성, 리듬, 화음 등의 모든 정보를 표현한 마크업 언어이다. 본 논문에서는 음계, 음자리표, 박자 정보 등 악보의 속성 정보와 추출된 음정 및 음의 길이 정보 등을 추가한 새로운 데이터 모델을 제안한다. 그림 3은 음악 메타 정보를 XML 스키마를 이용하여 표현한 다이어그램이다. 각 음악 파일은 하나의 *MusicScore* 엘리먼트를 가지며 제목, 장르, 작곡자에 대한 정보를 표현한 *MetaInfo* 엘리먼트, 음계, 음자리표, 박자를 표현한 *Attributes* 엘리먼트, 변환된 음정 및 음의 길이 등에 대한 정보를 스트링을 표현한 *Part* 엘리먼트를 하위 엘리먼트로 가진다.

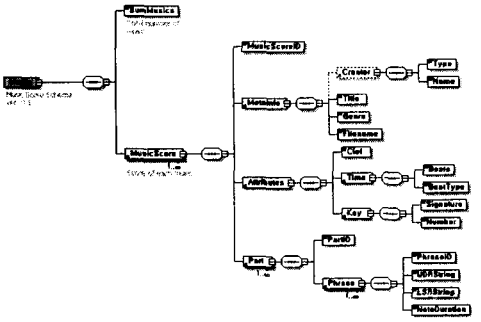

그림 3 음악 메타데이터를 위한 XML 스키마 다이어그램

3.2 인덱싱 메커니즘

그림 4는 본 논문에서 제안하는 FAI 기반 인덱싱 구조를 개괄적으로 보여준다. ①은 FAI의 생성 단계를 보여주며 FAI 생성 초기에는 FAI내에 엔트리가 없이 비어있는 상태로 존재하게 된다. 사용자의 질의 멜로디가 들어오면 해당 멜로디를 음악 데이터베이스에서 검색하고 부합하는 음악 컨텐츠가 있을 경우, FAI 엔트리에 음악 컨텐츠의 링크와 함께 질의 멜로디가 들어갈 공간을 하나 할당해 준다.

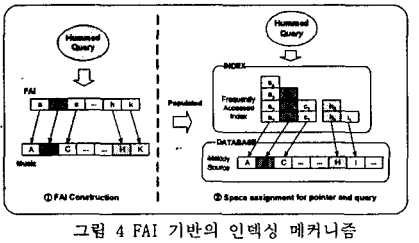

그림 4 FAI 기반의 인덱싱 메커니즘

이 공간에는 질의 멜로디에 대한 정보뿐만 아니라 접속 빈도수와 음악 컨텐츠내의 질의 멜로디 반복수에 대한 정보 역시 포함되어 있다. 이후에 다른 질의 멜로디가 음악 컨텐츠와 매치하면 새로운 FAI 엔트리 공간을 할당해 준다. 초기의 FAI 엔트리에는 이러한 방식으로 음악 컨텐츠와 매칭되는 모든 질의 멜로디를 넣어주게 되며 ②는 이렇게 채워진 FAI의 엔트리들을 보여주고 있다.

4. 음악 검색 시스템

4.1 구현

본 논문에서 제안하는 내용 기반 음악 검색 시스템에 대해서 설명한다. 클라이언트 인터페이스는 웹 기반으로 자바 애플릿과 JSP로 구현하였으며, 서버는 Java Sound와 MIDI 파일 처리를 위해서 jMusic [7] 라이브러리를 이용해서 음향 특징 정보를 추출했다. 음악 파일은 인터넷 사이트들로부터 약 15,000개의 MIDI 파일을 수집 하였고 이들 중 중복되는 파일과 MIDI 포맷이 아닌 파일을 제외한 나머지 12,000개의 MIDI 파일을 실험에 이용하였다.

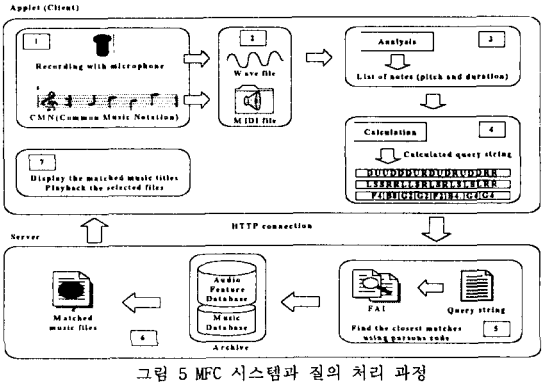

그림 5 MFC 시스템과 질의 처리 과정

그림 5는 MFC 시스템의 개략적인 구조와 질의의 흐름을 순서대로 나열하였다. 그림 6은 CMN(Common Music Notation)과 허밍에 의한 질의 인터페이스를 보여준다. CMN 인터페이스는 사용자가 드래그 앤 드랍(drag and drop)을 통하여 원하는 음표를 오선지 위에 그리는 것이며 음표의 길이 또한 지정할 수 있다. 허밍 인터페이스는 마이크를 통하여 약간의 흥얼거림이나 노래를 입력 받고 이를 웨이브 파일로 저장한다. 이렇게 저장된 웨이브 파일은 다시 미디 파일로 변환된 후에 2장에서 설명한 UDR, LSR 등의 스트링으로 변환되어 음악 특징 데이터베이스에 저장된다.

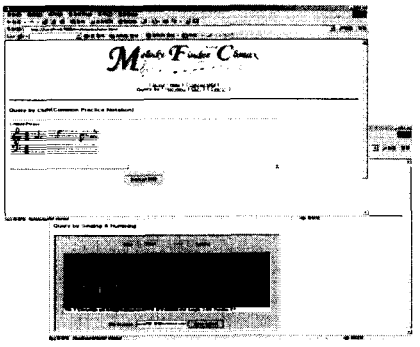

그림 6 CMN 및 허밍에 의한 질의 인터페이스

4.2 실험

실험은 정확 매칭 방법에서 많이 쓰이고 있는 Naive와 KMP(*Knuth-Morris-Pratt*) 그리고 BM(*Boyer-Moore*) 알고리즘을 사용하였으며 유사 매칭을 위해서 동적 프로그래밍 방법을 사용하여 성능 분석을 하였다. 본 논문에서는 세 개의 정확 매칭 알고리즘 중 가장 빠른 *Boyer-Moore* 알고리즘을 사용하여 검색 성능을 높였다. 그림 7은 음악 데이터베이스 내의 음악 컨텐츠 수를 증가시키며 측정한 사용자 질의에 대한 평균 질의 응답 시간을 보여준다. 두 실험 모두 컨텐츠의 수가 증가함에 따라 처리 시간도 같이 증가하였지만 FAI 기반의 검색이 모든 데이터베이스를 검색하는 방법보다 우월함을 알 수 있었고, 음악 컨텐츠의 수가 증가할수록 FAI 기반 검색이 얻는 이득은 증가하는 것으로 나타났다.

그림 8은 사용자 질의에 대한 각 알고리즘별 평균 질의 응답 시간을 보여준다. 첫번째 그래프는 사용자의 질의 멜로디 패턴의 길이가 "3"인 경우이며, 두번째 그래프는 "10"인 경우를 나타낸다. 각각의 알고리즘에 대한 질의 응답 시간은 10번의 질의에 대한 평균값을 적용하였으며, 두 경우 모두 BM 알고리즘이 KMP나 Naive 알고리즘보다 짧은 응답 시간을 가지는 것을 알 수 있다.

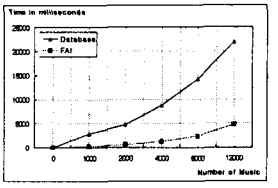

그림 7 FAI 기반 검색의 성능 분석

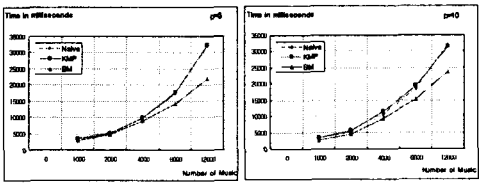

그림 8 Naive, KMP, BM 알고리즘을 이용한 질의 응답 시간

5. 결론 및 향후 계획

본 논문에서는 음악 컨텐츠의 다양한 특징 추출을 통해 멜로디를 검색할 수 있는 내용 기반 검색 시스템을 설계하였다. 사용자의 질의 패턴을 분석하여 자주 검색되는 멜로디의 위치를 FAI라는 새로운 개념의 인덱스에 저장하여 데이터베이스내의 모든 멜로디를 검색하지 않고 FAI를 우선적으로 검색하는 방법으로 검색 성능을 향상시켰다.

향후에는 다성음악(polyphonic) 컨텐츠로부터 특정한 악기의 채널을 제거한 뒤 단성음악이 가지는 멜로디와 유사한 멜로디 생성 후 이를 검색에 이용하는 방법에 대한 연구가 이루어져야 한다.

6. 참고 문헌

[1] A. Ghias, et al., 밠uery by humming – musical information retrieval in an audio database,□ In ACM Multimedia 95 -- Electronic Proceedings, San Francisco, USA, 1995.

[2] E. Wold, et al., 밒ontent-based Classification, Search and Retrieval of Audio,□ IEEE Multimedia 3(3), pp. 27-36, 1996.

[3] R.J. McNab, et al., 밣he New Zealand digital library MELody index,□ D-Lib Magazine, May 1997.

[4] Michael Good, 밠sing XML for Musical Representation," http://www.musicxml.org/stanford.html

[5] E. Hwang and D. Park, 밿opularity-Adaptive Index Scheme for Fast Music Retrieval,□ Proc. of IEEE Multimedia and Expo, Lausanne, Switzerland, August 2002.

[6] R. Boyer and S. Moore, "A fast string searching algorithm," Communications of the ACM, (20):762-772, 1977

[7] jMusic Java library, http://jmusic.ci.qut.edu.au/

[8] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.